



T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü



DİNAMİK ORTAMLARDA ÇALIŞAN ÇOKLU OTONOM ROBOTLAR İÇİN KONTROL ALGORİTMASININ GELİŞTİRİLMESİ

Yüksek Lisans Tezi

Serkan PINAR

Mekatronik Mühendisliği Anabilim Dalı

İzmir
2024

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

**DİNAMİK ORTAMLARDA ÇALIŞAN ÇOKLU
OTONOM ROBOTLAR İÇİN KONTROL
ALGORİTMASININ GELİŞTİRİLMESİ**

Serkan PINAR

Danışman: Dr. Öğr. Üyesi Dilşad ENGİN

Mekatronik Mühendisliği Anabilim Dalı
Mekatronik Mühendisliği Yüksek Lisans Programı

İzmir

2024

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum “Dinamik ortamlarda çalışan çoklu otonom robotlar için kontrol algoritmasının geliştirilmesi” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

12 / 01 / 2024

İmzası

Serkan PINAR



ÖZET**DİNAMİK ORTAMLARDA ÇALIŞAN ÇOKLU OTONOM
ROBOTLAR İÇİN KONTROL ALGORİTMASININ
GELİŞTİRİLMESİ**

PINAR, Serkan

Yüksek Lisans Tezi, Mekatronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Dilşad ENGİN

Ocak 2024, 120 sayfa

Bu tez çalışmasında, endüstri 4.0 kapsamında akıllı fabrika ortamlarını da gözeterek çoklu otonom araçlar için durağan ve hareketli ortamlarda kendi aralarındaki iş yoğunluğunu koordineli bir şekilde görev paylaşımı yapan çoklu otonom mobil robotlar için yük depolama algoritması geliştirilmiştir.

Geliştirilen algoritma, otonom mobil robotlar arasındaki kablosuz haberleşme ile atanan görevlerin sistemin başarımına olan faydalarını otonom mobil robot uygulamaları yapılarak incelemek üzerine geliştirilmiştir. Bu kapsamda yapılan ilk uygulamada konveyör bant üzerindeki sensörlerden gelen bilgiler ile otonom mobil robotun engelsiz bir ortamda haberleşerek istenilen noktaya gitmesi sağlanmıştır. Bir sonraki aşamada istenilen noktaya giden robotun, ürünü alıp kendisi için belirlenen depo alanına yine engelsiz bir ortamda gitmesi sağlanmıştır. Bu uygulamaların üzerine otonom mobil robot, konveyör bant ve depo alanı sayıları iki katına çıkarılarak engelli ortamlarda çalışmaya devam edilerek robotlar arasındaki iletişimin sisteme olan katkısı belirlenmiştir. Devam eden çalışmada ise herhangi bir konveyör bant üzerinde ürün yoğunluğu olduğunda, otonom mobil robotlar koordineli iş paylaşımı yaparak yoğunluğun olduğu banda giderek iş paylaşım görevini gerçekleştirmiştir. Geliştirilen algoritmanın son aşamasında ise gün sonunda hangi bantta ürün fazla ise mobil robot o banda giderek beklemektedir.

Bu çalışmadaki amaç, otonom araçların iş paylaşımını koordineli bir şekilde gerçekleştirerek en kısa yoldan iş görevini gerçekleştirmesi ve üretim hızını ve verimliliği arttırmayı hedeflemektedir.

Bu tez çalışmasında Festo firmasının üretmiş olduğu Robotino mobil otonom robotu kullanıldı. Robotino robotunu programlamak için Festo firmasının geliştirdiği Robotino View yazılımı kullanıldı. Konveyör bantları, Siemens S7-1200 PLC ile kontrol edildi. Robotino otonom robotlar ile konveyör bantları kontrol eden PLC'ler arasındaki veri alışverişi, robot ve PLC üzerlerine harici olarak yerleştirilen ESP32 Wi-Fi modülü ile gerçekleştirildi. Robotino otonom araçlar için konumlandırma ve yön bilgisi durumunu odometri verilerini kullanarak ve Robotino üzerindeki kameranın, mobil robotun hedef noktalara gittiği son noktalardaki kare işaretleyici bilgisini okuyarak eksenlerde konum dengeleme sağlanmaya çalışılmıştır.

Anahtar sözcükler: Otonom mobil robotlar, kablosuz iletişim, akıllı fabrikalar, Robotino, koordineli iş paylaşımı, çoklu otonom robot sistemleri, akıllı üretim, konumlandırma.

ABSTRACT**DEVELOPING A CONTROL ALGORITHM FOR MULTIPLE
AUTONOMOUS ROBOTS IN DYNAMIC ENVIRONMENTS**

PINAR, Serkan

MSc in Mechatronic Engineering

Supervisor: Assist. Prof. Dr. Dilşad ENGİN

January 2024, 121 pages

In this thesis, a product storage algorithm has been developed for multiple autonomous mobile robots that share tasks among themselves in a coordinated manner in static and dynamic environments for multiple autonomous vehicles by considering smart factory settings within the scope of Industry 4.0.

The algorithm was developed to assess the advantages of wireless communication between autonomous mobile robots in improving system performance for autonomous mobile robot applications. In the first application, the autonomous mobile robot was able to navigate to a desired point by receiving information from sensors on the conveyor belt in an unobstructed environment. In the following stage, an autonomous mobile robot is deployed to retrieve the product and transport it to a designated warehouse area in an unobstructed environment. After these applications, the number of autonomous mobile robots, conveyor belts and warehouse areas were doubled. The contribution of robot-to-robot communication to the system was also determined by operating in disabled environments. In the ongoing study, when there was a product density on any conveyor belt, autonomous mobile robots performed coordinated work sharing by going to the belt where the density was located.

The aim of this study is to enable autonomous vehicles to perform the work task in the shortest way by performing work sharing in a coordinated manner and to increase production speed and efficiency.

In this thesis, the Robotino autonomous mobile robot produced by Festo was used. In order to program the Robotino robot, the Robotino View software developed by Festo was used. Conveyor belts were controlled by Siemens S7 1200 PLC. Data exchange between Robotino autonomous robots and PLCs controlling conveyor belts was realized with the ESP32 Wi-Fi module placed externally on the robots and the PLCs. Robotino is used for positioning and orientation information for autonomous vehicles using odometry data, and the camera on Robotino reads frame marker information where the mobile robot travels to target points to provide position compensation in the axes.

Keywords: Autonomous mobile robots, wireless communication, smart factories, Robotino, coordinated work sharing, multi-autonomous robot systems, smart manufacturing, positioning.

İÇİNDEKİLER

Sayfa

1	GİRİŞ.....	1
1.1	Robotların Sınıflandırılması	2
1.1.1	İnsansı robotlar	2
1.1.2	Sürü robotlar	3
1.1.3	Çoklu robotlar	3
1.1.4	İşbirlikçi ve etkileşimli robotlar.....	3
1.1.5	Mikro-nano robotlar.....	4
1.1.6	Biyo-ilhamlı robotlar	4
1.1.7	Haptik sistemler	4
2	İLGİLİ ÇALIŞMALAR	7
2.1	Otonom Mobil Robotlar ile İlgili Yapılan Çalışmalar.....	7
2.2	Festo Robotino Mobil Otonom Robot ile Yapılan Çalışmalar	9
3	FESTO ROBOTİNO MOBİL OTONOM ROBOTU	11
3.1	Robotino Mobil Robot.....	11
3.2	Festo Robotino Mobil Robotun Tekerlek Yapısı	13
3.3	Festo Robotino Robotun Kinematik Modeli	14
3.3.1	Kinematik kontrol ile hız tahmini	16
3.4	Festo Robotino PC Haberleşmesi	18
3.5	Festo Robotino View Programı	19

İÇİNDEKİLER (devam)

3.6 Robotino View Programında Kullanılan Fonksiyonlar	19
3.6.1 Sürücü sistemi	19
3.6.2 Odometri.....	20
3.6.3 Konum sürücüsü (position drive).....	21
3.6.4 Poz ayrıştırıcı (pose decomposer)	21
3.6.5 Matematiksel ve mantıksal işlemler	22
3.6.6 Kamera	23
3.6.7 Renk aralığı bulucu (Color range finder)	24
3.6.8 Segment izleyici (segment tracker)	24
3.6.9 Hat dedektörü (line detector).....	25
3.6.10 Görüntü bilgileri (image information).....	26
3.6.11 Renk uzayı dönüşümü (Colorspace conversion).....	27
3.6.12 ROI.....	28
3.6.13 Filtre	28
3.6.14 İşaretleyici algılama (marker detection).....	29
3.6.15 İşaretleyici konumu (marker position)	29
3.7 ESP32 Mikrodenetleyici Wi-Fi Kartı	29
3.8 ESP32 Kablosuz İletişim Protokolleri	33
3.8.1 ESP-NOW	33
3.9 Siemens S7-1200 PLC	34

İÇİNDEKİLER (devam)

3.9.1	Profinet endüstriyel haberleşme protokolü	35
4	KONTROL, YÖNTEM VE ALGORİTMA.....	37
4.1	Kontrol Noktaları.....	37
4.1.1	Konumlandırma yöntemleri ve sınıflandırılmaları	38
4.1.2	Robot konumlandırma yöntemleri	40
4.1.3	Hareket planlaması	40
4.1.4	Genel konum kontrol şeması	41
4.2	Odometri Kaçıklık Hatası Nedenleri	42
4.3	Laboratuvar Ortamında Robotino Odometri Hataları Belirlenmesi.....	45
4.4	Odometri Kaçıklık Hatası Sonucu Lokalizasyon Hatalarının İyileştirilmesi .	57
4.4.1	Konum sürücüsü (position drive) grafik ayarlamasının yapılması	57
4.4.2	Holonomik ve non-holonomik sürüş	60
4.4.3	Velocity ramp (hız rampası) ve angular velocity ramp (açısal hız rampası)	61
4.5	Dijital Görüntü İşleme	66
4.5.1	Görüntü işleme ile renk algılama	67
4.5.2	Kare işaret işleme.....	68
4.5.3	İşaretçi algılayıcı ile konum tahmini.....	69
4.6	Lua Yazılımı ile Koordinatlandırma	71
4.7	Robotino Mobil Robotun Engelden Kaçınma Kontrolü.....	75
4.8	Programlanabilir Denetleyici ile Yürüyen Bant Kontrolü.....	79

İÇİNDEKİLER (devam)

4.9	ESP 32’ler Arası Haberleşme Kontrolü ve Yazılımı	82
4.10	Algoritma	85
5	SONUÇ	88
6	KAYNAKLAR DİZİNİ	93
7	TEŞEKKÜR	97
8	ÖZGEÇMİŞ	98



ŞEKİLLER DİZİNİ

<u>Şekil</u>	
Şekil 3. 1 Robotino mobil robot önden görünümü	12
Şekil 3. 2 Robotino robotun dijital ve analog giriş/çıkışları	13
Şekil 3. 3 Robotino mobil robotun çok yönlü tekerlek yapısı	14
Şekil 3. 4 Robotun Kinematik Modeli (Masmoudi et al., 2016).	14
Şekil 3. 5 Robotino PC haberleşmesi	18
Şekil 3. 6 (a) Robotino View programında omnidrive sürücüsü sistemi, (b) omnidrive tekerler.....	19
Şekil 3. 7 Robotino View odometri bloğu ve yönelim grafiği	20
Şekil 3. 8 Robotino View odometri bloğu	21
Şekil 3. 9 Robotino View poz ayırıştırıcı bloğu	22
Şekil 3. 10 Robotino View matematiksel ve mantıksal blokları	23
Şekil 3. 11 Robotino View kamera bloğu.....	23
Şekil 3. 12 Robotino View renk aralığı bulucu bloğu	24
Şekil 3. 13 Robotino View segment izleyici bloğu	25
Şekil 3. 14 Robotino View hat dedektörü bloğu.....	25
Şekil 3. 15 Robotino robotun hat çizgisi izlerken kamera görüntüsü.....	26
Şekil 3. 16 Robotino View görüntü bilgileri bloğu ve iletişim kutusu.....	27
Şekil 3. 17 Robotino View renk uzayı dönüşümü bloğu	28
Şekil 3. 18 Robotino View ROI bloğu.....	28
Şekil 3. 19 Robotino View filtre bloğu.....	28
Şekil 3. 20 Robotino View işaretleyici algılama bloğu	29
Şekil 3. 21 Robotino View işaretleyici konum bloğu.....	29
Şekil 3. 22 ESP32 Mikrodenetleyici Wi-Fi Kartı	30
Şekil 3. 23 ESP32 mikrodenetleyici kartı pinleri (http://esp32io.com/).....	31
Şekil 3. 24 ESP32 Mikrodenetleyicilerin PLC'ler ile Robotino mobil robotlar arasındaki haberleşmesi	32
Şekil 3. 24 ESP32 mikrodenetleyicilerin kendi aralarındaki haberleşmesi	33
Şekil 3. 25 OSI modelinin beş katmanı	34
Şekil 3. 26 Siemens S7-1200 PLC.....	35
Şekil 3. 27 PROFINET protokolü	36
Şekil 4.1Klasik geribeslemeli kontrol	37

Şekil 4.2 Mobil robotlarda navigasyon problemi hareket planlaması	38
Şekil 4.3 Konum kontrol şeması	42
Şekil 4.4 (a) Birinci 500mm için oluşan kaçıklıklar ve işaretlemeleri, (b) ikinci 500mm için oluşan kaçıklıklar ve işaretlemeler	46
Şekil 4.5 Robotino x ve y eksen kaçıklıkları	47
Şekil 4.6 Birinci 500mm ileri ve geri hareketleri sırasında odometriden ölçülen x, y ve phi kaçıklıkları.....	47
Şekil 4.7 Başlangıç noktasındaki y eksen kaçıklığı ve Robotino ileri gittiğindeki x ve y eksen kaçıklık ölçümleri.....	48
Şekil 4.8 İkinci 500mm ileri gidiş ve geri dönüş denemesi sırasında odometriden ölçülen x, y ve phi kaçıklıkları.....	49
Şekil 4.9 Başlangıç noktasındaki y eksen kaçıklığı ve ileri gittiğindeki x ve y eksen kaçıklıkları	49
Şekil 4.10 Odometriden alınan x, y ve phi verilerinin karşılaştırması.....	51
Şekil 4.11 İki deneme sonucunda Robotino mobil robottun eksen kaçıklık karşılaştırmaları.....	51
Şekil 4.12 1000 mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar	52
Şekil 4.13 Birinci 1000mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar	53
Şekil 4.14 İkinci 1000mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar	53
Şekil 4.15 Odometriden alınan x, y ve phi verilerinin karşılaştırması.....	55
Şekil 4.16 1000mm iki deneme sonucunda Robotino eksen kaçıklık karşılaştırmaları.....	56
Şekil 4.17 Robotino View konum sürücüsü standart grafiği	58
Şekil 4.18 Konum sürücüsü grafiği yapılandırılmış ayar değerleri	58
Şekil 4.19 (a) Robotino kamera görüntüsü, (b) Robotino kamera kalibrasyonu görüntüsü.....	67
Şekil 4.20 Robotino View programında kamera ile renk işleme	68
Şekil 4.21 İşaretçi konum belirleme bloğu tarafından üretilen işaretleyici örnekleri	68
Şekil 4.22 Üretilen kare işaretleyicinin kamera tarafından görüntülen vektörel gösterimi.....	69
Şekil 4.23 İşaret algılayıcı ile hedef koordinata yönlendirme algoritması deneyi.....	70
Şekil 4.24 Robotino View programı ile işaret algılayıcı konumunu okuma.....	70
Şekil 4.25 Robotino View programındaki Lua bloğu	71

Şekil 4.26 Robotino mobil robotun Lua bloğuna yazılan program ile dengeye getirilmesi	72
Şekil 4.27 Robotino IR sensör yerleşimi	76
Şekil 4.28 Robotino View programında engelden kaçınma şeması	77
Şekil 4.29 Robotino View programında engel gördükten sonra sola dönme şeması	78
Şekil 4.30 Robotino View programında dönme hareketi bittikten sonraki ilerleme hareketi şeması.....	78
Şekil 4.31 Robotino View programında engelden kaçınma algoritmasının çalıştırılması.....	79
Şekil 4.32 Programlanabilir denetleyici ile yürüyen bant kontrolü programı	79
Şekil 4.33 Çalışmanın yürüyen bant kısmının temel yapısı	81
Şekil 4.34 Laboratuvar ortamında çalışan Robotino'ların görüntüsü.....	82
Şekil 4.35 ESP32 mikrodnetleyici kartının Wi-Fi haberleşme için Robotino mobil robot üzerindeki bağlantısı	82
Şekil 4.36 ESP32 mikrodnetleyici kartının Wi-Fi haberleşme için bant üzerindeki PLC ile bağlantısı	84
Şekil 4.37 Robotino mobil robotun engelden kaçarak diğer Robotino mobil robot ile koordineli çalışmasını gösteren algoritma	86

TABLOLAR DİZİNİ

<u>Tablo</u>	<u>Sayfa</u>
Tablo 3. 1 Robotino parametreleri ve değerleri	12
Tablo 3. 2 ESP32 Teknik Özellikleri	31
Tablo 4. 1 Robot konumlandırma yöntemleri ve hareket planlaması	39
Tablo 4. 2 Sistematik ve sistematik olmayan hataların özeti	43
Tablo 4. 3 Birinci 500mm denemesinden elde edilen veriler	48
Tablo 4. 4 İkinci 500mm denemesinden sonra elde edilen veriler	50
Tablo 4. 5 Birinci 1000mm denemesi sonucu odometriden alınan x, y ve phi verilerinin karşılaştırması.....	54
Tablo 4. 6 İkinci 1000mm denemesi sonucu odometriden alınan x, y ve phi verilerinin karşılaştırması.....	55
Tablo 4. 7 Konum sürücüsü hızlanma yapılandırma değerleri	59
Tablo 4. 8 konum grafiği phi yapılandırma değerleri	59
Tablo 4. 9 Pozisyon sürücüsü grafik yapılandırması yapıldıktan sonraki deneme sonuçları 1	63
Tablo 4. 10 Pozisyon sürücüsü grafik yapılandırması yapıldıktan sonraki deneme sonuçları 2	64
Tablo 4. 11 Holonomik sürüş için velocity ramp ve angular ramp süreleri değiştirilmiş deney verileri.....	65
Tablo 4. 12 Robotino robot IR sensörlerinin mesafe - gerilim ilişkisi	76

KISALTMALAR DİZİNİ

Kısaltmalar

Wi-Fi	Kablosuz Bağlantı
SARSA	Durum-Eylem-Ödül-Durum-Eylem
A3C	Avantaj Aktör Eleştirmeni
RL	Pekiştirmeli Öğrenme
GA	Genetik Algoritma
DRL	Derin Öğrenme Algoritması
TCP/IP	İletim Kontrol Protokolü/İnternet Protokolü
ROS	Robot İşletim Sistemi
WLAN	Kablosuz Yerel Alan Ağı
IR	Kızılötesi
AP	Erişim Noktası
IMU	Ataletsel Ölçüm Birimi
GPS	Küresel Konumlandırma Sistemi
PID	Oransal-İntegral-Türevsel
RGB	Kırmızı-Yeşil-Mavi
BGR	Mavi-Yeşil-Kırmızı
ID	Kimlik Numarası

KISALTMALAR DİZİNİ (devam)

HSV	Renk Tonu-Doygunluk-Açıklık
IoT	Nesnelerin İnterneti
ESP NOW	Bağlantısız Wi-Fi İletişim Protokolü
BLE	Bluetooth Low Energy (Bluetooth Düşük Enerjili)
MQTT	Mesaj Kuyruğu Telemetri Aktarımı
GPRS	Genel Paket Radyo Servisi
OSI	Açık Sistemler Ara Bağlantısı
PLC	Programlanabilir Mantıksal Denetleyici
DC	Doğru Akım
QR	Kare Kod
MAC	Ortam Erişim Kontrolü

SİMGELER DİZİNİ

<u>Simgeler</u>	<u>Açıklama</u>
$\dot{\theta}_i$	Tekerleğin açısal hızı
v_i	Tekerleğin doğrusal hızı
r	Tekerleğin yarıçapı
R	Bir tekerlek ile robotun merkezi arasındaki mesafe
ϕ_i	Robotun açısal hızı
φ_i	Tekerleğin açısal konumu
$P(\phi)$	Tekerleklerin açısal hızları ile global hız vektörü arasındaki dönüşüm matrisi
i	Tekerleğin indeksi ($i = 1, 2, 3$)
$\dot{x}, \dot{y}$	Robotun doğrusal hızları

1 GİRİŞ

Otonom mobil robotlar; teknolojinin hızlı ilerlemesiyle birlikte günümüzde fabrikalar, depolar ve dış ortamlar gibi koşulların sürekli değiştiği dinamik ortamlarda giderek daha fazla kullanılmaktadır. İnsan müdahalesi olmaksızın çevrelerini algılayabilen, kararlar alabilen ve iş birliği içinde çalışabilen otonom araçlar, dinamik ortamlarda etkin bir şekilde görevleri yerine getirebilmektedirler. Bu dinamik koşullarda çalışabilmeleri için robotların algılama, konum belirleme, planlama ve karar verme gibi yeteneklerle donatılması gerekmektedir.

Dinamik ortamlar için otonom mobil robotların geliştirilmesindeki en önemli zorluklardan biri lokalizasyon ve sağlam algılama yetenekleri sağlamaktır. Mobil otonom robotlar için lokalizasyon planlama, navigasyon ve görevlerin yürütülmesi için önem taşımaktadır. Robotların çevrelerindeki engeller, dinamik nesneler ile insanlar ve yapılandırılmamış ortamlar gibi değişikliklerle başa çıkmaları gerekir. Bu zorlukların üstesinden gelmek için semantik segmentasyon [1], nesne algılama [2] ve monoküler derinlik tahmini [3] gibi çeşitli bilgisayarla görme tabanlı algılama yöntemleri kullanılmıştır. Kullanılan yöntem ne olursa olsun gerçek ortamda çalışmalar yapıldığında kullanılan sensör verilerin algoritmalar vasıtasıyla işlenerek düzeltmelerin yapılması gerekir.

Dinamik ortamlarda karşılaşılan en önemli zorluk bir robotu ortam içinde doğru şekilde konumlandırmaktır. Konumlandırma hatalarının nedenleri düşük sensör hassasiyeti, ortam gürültüsü, robotun yürüyeceği yüzey alandaki pürüzler, robot tekerlek uzunluk ve çaplarındaki farklılıklardan kaynaklı tekerlerin kalkış, duruş ve dönüşteki odometri kaçıklıkları ve robot sisteminin mekaniksel belirsizlikleri olarak tanımlanabilir. Bu zorlukları aşmak için kullanılan donanımsal ve yazılımsal yöntemlerin bazıları görsel odometri [4], eşzamanlı lokalizasyon ve haritalama (Simultaneous Localization And Mapping – SLAM) [5] ve parçacık filtreleme [6] gibi çeşitli sensör tabanlı lokalizasyon yöntemleridir.

Algılama ve lokalizasyona ek olarak, dinamik ortamlarda çalışan robotlar etkili planlama ve karar verme yeteneklerine de ihtiyaç duyarlar. Bu zorluğun üstesinden gelmek için kullanılan popüler yaklaşımlardan biri pekiştirmeli

öğrenme (Reinforcement Learning – RL) yöntemidir [7]. Bu yöntemde robot, eylemleri için aldığı verilere dayalı olarak karar vermeyi öğrenir. Diğer bir yaklaşım ise robotun çevrenin gelecekteki durumunu tahmin ettiği ve hedeflerine ulaşmak için en iyi eylemi seçtiği model öngörülü kontrol kullanılmasıdır.

Sonuç olarak, dinamik ortamlar için otonom mobil robotların geliştirilmesi, sağlam algılama, konum belirleme, planlama ve karar verme yetenekleri gerektiren zorlu bir problemdir. Bu alandaki araştırmalar, dinamik ortamlarda etkin bir şekilde çalışabilen robotlar geliştirmek için bir araya getirilebilecek çeşitli yöntem ve teknikler üretmiştir.

1.1 Robotların Sınıflandırılması

Robotları, endüstriyel robotlar ve mobil robotlar olarak sınıflandırabiliriz. Endüstriyel robotlar, eklemlerinin yapısına ve işlevlerine göre sınıflandırılır. Endüstriyel robotlar, bir üretim veya imalat ortamında programlanmış bir çerçeve içerisinde tekrarlanan görevleri yerine getirmek üzere tasarlanmıştır. Ürünleri boyamak, monte etmek, kaynak yapmak gibi belirlenmiş görevleri gerçekleştirmek için kullanılırlar. Mobil robotlar ise ofisler, hastaneler, evler, fabrikalar gibi iç ortamlar veya endüstrideki diğer dış ortamlar gibi dinamik ortamlarda hareket etmek ve gezinmek için tasarlanmıştır. Mobil robotlar genel olarak otonomdur ve kendi başlarına karar verebilirler. İşlevlerine, sayılarına ve uygulama alanlarına göre sınıflandırılmaktadır.

1.1.1 İnsansı robotlar

İnsansı robotlar, insan benzeri hareketler, mimikler, ses yığınları veya konuşma tarzı gibi ilerlemeye sahip olabilir. İnsansı robotlar, evde bakım, eğitim, eğlence, sanat veya ticari amaçlar gibi birçok alanda kullanılabilir. İnsansı robotlar ayrıca, insanlar gibi duygusal ve zihinsel yorumlama yapabilir ve bu tanımlayabilir veya yanıt verebilir. İnsansı robotlar, makine öğrenmesi, doğal dil işleme, ses tanıma veya görüntü tanıma gibi teknolojik gelişmelerle birlikte daha da gelişebilir.

1.1.2 Sürü robotlar

Sürü robotları, belirli bir amaca yönelik hareket etme yeteneği olan birden fazla robotun grup olarak çalışmasını amaçlar. Sürü robotlar, ortak bir hedefe ulaşma amacıyla tasarlandıklarından, kendi aralarında iletişim kurarlar ve koordine edilirler, böylece karmaşık görevleri daha verimli bir şekilde gerçekleştirebilirler. Sürü robotların uygulama alanları denizaltı robotları, insansız hava araçları, güvenlik ve savunma, tarım, vb. bulunmaktadır. Sürü robotların performansı, belirli bir amaç doğrultusunda etkin ve verimli çalışmanın yanı sıra ortak bir davranış gerçekleştirme amacını hedefler.

1.1.3 Çoklu robotlar

Çoklu robotlar, birden fazla robottan oluşan bir grup veya sistemdir ve birlikte çalışarak belirli bir görevi gerçekleştirmek veya belirli bir davranışı sergilemek için tasarlanmışlardır. Bu sistemler, birçok farklı amaç ve uygulama alanında kullanılabilir, örneğin hava ve deniz araştırmaları, çevre tarama, güvenlik sağlama, ekonomik veya endüstriyel amaçlar gibi. Çoklu robotlar ayrıca, bir takım tehlikeli veya zorlu görevler için de kullanılabilir, ancak bu tür görevlerin yapılması için robotlar arasında etkileşim ve koordinasyon gereklidir. Çoklu robotlar için pek çok farklı çözüm ve yaklaşım vardır ve bu eylemler genellikle, robotlar arasında veri paylaşma, koordinasyon ve karar verme gereksinimleri gerektirir.

1.1.4 İşbirlikçi ve etkileşimli robotlar

İşbirlikçi ve etkileşimli robotlar, kişiler veya diğer robotlarla beraber çalışan robotlardır. Bu tür robotlar, yapılan işin yüküne göre insanlarla veya diğer robotlarla etkileşim kurarlar. İşbirlikçi robotlar, insanlar tarafından yönetilen veya kontrol edilen robotları da içerir. İşbirlikçi ve etkileşimli robotlar, insanlarla birlikte çalışma gibi zor veya tehlikeli görevlerin yerine getirilmesinde veya insanların fiziksel veya zihinsel yetersizliklerinin aşılmasında kullanılabilir.

1.1.5 Mikro-nano robotlar

Mikro-nano robotlar, son derece küçük boyutlara sahip otonom robotlardır ve bu özellikleri nedeniyle çok özel uygulama alanlarına sahiptirler. Genellikle ölçümler mikrometre ve nanometre arasındadır. Bu robotlar, çok küçük olmaları nedeniyle insan gövdesi içinde veya diğer hassas ortamlarda kullanılabilir. Mikro-nano robotlar, otomatik aktüatörler, piezoelektrik aktüatörler, termal aktüatörler, elektrokimyasal aktüatörler gibi çeşitli hareket metotları kullanarak hareket edebilir. Ayrıca, bu robotlar çeşitli sensörler ve araçlarla donatılabilir ve veri toplamaya, nesne takip etmeye, dokulara veya verilere giriş yapmaya, tedavi uygulamalarına veya endüstriyel kullanımlara yönelik kullanılabilir.

1.1.6 Biyo-ilhamlı robotlar

Biyo-ilhamlı robotlar, doğadaki canlı organizmaların anatomisinden, davranışlarından veya işlevlerinden ilham alınarak tasarlanmış olan robotlardır. Bu robotlar, biyolojik sistemlerin özelliklerini taklit ederek veya esinlenerek daha iyi bir şekilde çalışmak veya belirli görevleri yerine getirmek için tasarlanırlar.

1.1.7 Haptik sistemler

Mekanik algılamaya odaklanan robotik sistemlerdir. Bunlar, robotların hissetme ve tasarım fonksiyonlarını simüle etmek için kullanılan mekanik ve elektronik sistemlerdir. Haptik sistemler, doğal olarak insan ve hayvanların doku algısı gibi davranış taklit etmek amacıyla tasarlanır. Haptik teknolojiler çeşitli alanlarda, özellikle tıp, endüstriyel robotik ve hareketli oyunlarda yaygın olarak kullanılmaktadır.

Haptik sistemler; kuvvet ve darbe sensörleri, aktüatörler ve haberleşme birimlerini içerir ve insan-makine etkileşimini daha dokunsal ve gerçekçi hale getiren önemli bir teknolojik gelişme olarak kabul edilir.

Bu tezde Robotino mobil robotlarının iş birlikçi olarak dinamik ortamlarda etkin ve verimli bir şekilde çalışabilmeleri için yeni algoritmaların ve stratejilerin

geliştirilmesi amaçlanmıştır. Bu doğrultuda, robotların koordinasyonunu sağlayacak, görevleri paylaşacak ve iletişimini etkinleştirecek algoritmaların geliştirilmesi hedeflenmektedir. Tez çalışmasında, iş birlikçi mobil robotların dinamik ortamlardaki etkinliğini artırmak için farklı senaryolarda uygulanan algoritmaların performansını değerlendirilmiş ve literatürdeki benzer çalışmalarla karşılaştırılmıştır. Bu sayede, iş birlikçi mobil robotların dinamik ortamlardaki etkinliğini artırmak için yeni yaklaşımların geliştirilmesine katkı sağlayacağı ön görülmektedir.

Bu tezin önemi, dinamik ortamlarda çalışan Robotino mobil robotların iş birlikçi olarak etkin bir şekilde çalışabilmeleri için geliştirilecek algoritmaların ve stratejilerin araştırılması ve analiz edilmesinde yatmaktadır. Otonom mobil robotlar, çevrelerini algılayabilen, kararlar alabilen ve iş birliği içinde görevleri yerine getirebilen cihazlar olarak önemli bir teknoloji alanını temsil etmektedir. Ancak, dinamik ve karmaşık ortamlarda birden fazla robotun koordinasyonu ve etkin iş birliği sağlaması çeşitli zorlukları beraberinde getirmektedir.

Bu robotların dinamik ve karmaşık ortamlarda işbirlikçi bir şekilde çalışabilme yeteneği, verimlilik ve etkinlik açısından büyük önem taşımaktadır. Özellikle, birbirleriyle haberleşerek koordineli olarak görevleri yerine getirebilen robotların kullanımı, çeşitli sektörlerde verimliliği artırmak ve iş süreçlerini optimize etmek için büyük bir potansiyele sahiptir.

Tezin başka bir önemi ise, Robotino mobil robotunun ESP32 Wi-Fi özellikli mikrodenetleyici kartı ile haberleşerek dinamik ortamlarda işbirlikçi bir şekilde çalışmasını araştırmak ve incelemektir. ESP32 Wi-Fi mikrodenetleyici kartı, kablosuz iletişim ve internet bağlantısı sağlama kabiliyetine sahip olan güçlü bir mikrodenetleyici platformudur. Robotino'nun bu kartla entegrasyonu, robotların diğer Robotino'larla ve merkezi bir kontrol birimiyle iletişim kurmasını ve veri alışverişi yapmasını mümkün kılar. Bu sayede, robotların dinamik ortamlarda koordinasyon ve iş birliği içinde görevleri yerine getirmesi daha etkin ve verimli bir çalışma süreci sağlar. Robotino mobil robotlarının ESP32 Wi-Fi kartı kullanılarak haberleşmesi için geliştirilen algoritmalar ve iletişim protokolleri üzerine odaklanılacaktır. Ayrıca, robotların dinamik ortamlarda engelleri

algılayarak ve diğer robotlarla uyum içinde çalışması için navigasyon ve koordinasyon stratejileri ve algoritmaları geliştirilmiştir. Bu çalışmaların sonucunda, otonom mobil robotların işbirlikçi bir şekilde çalışma yetenekleri artırılarak, endüstriyel otomasyon, lojistik yönetimi, tarım ve sağlık hizmetleri gibi çeşitli alanlarda verimlilik ve etkinlik açısından önemli bir gelişim sağlanması hedeflenmektedir.

Birinci bölümde, mobil robotların sınıflandırılması verilmiştir. İkinci bölümde, otonom mobil robotlar ile ilgili yapılan çalışmalardan bahsedilmiştir. Üçüncü bölüm, Festo firmasının üretmiş olduğu Robotino mobil robotun teknik özellikleri ve kinematik modeli, algoritmasını hazırladığımız Robotino View programı ve programda kullandığımız blok şemaları ile Robotino'lar arasındaki haberleşmeyi sağlayan ESP32 Wi-Fi kartı ve özellikleri yer almaktadır. Dördüncü bölümde, tez kapsamında fiziksel bir deney ortamında tasarlanan algoritma ile çalıştırılan iki Robotino mobil robotla ilgili deneysel sonuçlar detaylandırılarak mobil robot navigasyonunun performanslarının değerlendirilmesi yapılmıştır. Önceki çalışmalarda da karşılaşılan odometri kaçıklık hataları bu çalışmada geliştirilen algoritma ile en aza indirilerek Robotino'ların çalışma sonuçları verilmiştir.

Son bölüm olan beşinci bölümde ise tez çalışması sonucu elde edilen bulgular, tezin literatüre sunduğu katkılar, algoritmalarının literatürde incelenen çalışmalara üstünlükleri açıklanmıştır.

2 İLGİLİ ÇALIŞMALAR

2.1 Otonom Mobil Robotlar ile İlgili Yapılan Çalışmalar

Dinamik ortamlarda çalışan çoklu otonom robotlar için makine öğrenmesi konusunda birçok araştırma ve çalışma yapılmıştır. Bu çalışmalar arasında robotların koordinasyonu, yapay sinir ağları, genetik algoritmalar, veri madenciliği, destek vektör makine öğrenmesi ve kümeleme yöntemleri gibi yöntemlerin kullanılması bulunmaktadır. Ayrıca, dinamik ortamlardaki robotların dağıtık veya merkezi bir yapıda çalışması, veri toplama ve işleme, öğrenme ve adaptasyon gibi konular da bu çalışmalar arasında yer almaktadır. Lozano-Pérez and Wesley, (1979), bir mobil robotun çok yüzölçümlü engeller arasında çarpışması olmadan bir yol bulması için bir algoritma sunmaktadır. Makalede, robotun etrafındaki engelleri ve varış noktasını tanımlamaya ve yolun bu verilerle planlanmasına yönelik bir yöntem sunulmaktadır. Algoritma, robotun mevcut konumunun en yakın noktasına giderek ve orada bulunan çok yüzölçümlü engelleri değerlendirerek hareket etmesini ve engelleri aşmasını içermektedir.

Antonelli et al., (2008), farklı oluşum görev kontrollerini gerçekleştiren çoklu robot sistemlerini kontrol etmek için Boş-Uzay-tabanlı Davranışsal (Null-Space based Behavioral – NSB) yaklaşıma dayalı uygun bir matematiksel model geliştirmiş ve deneysel olarak farklı mobil robotlar üzerinde simülasyon olarak test etmiştir. Varol, (2009), çoklu mobil robot sistemin hareket davranışını kontrol etmek için iki adımlı bir yaklaşım izlemiştir. İlk adımda her robot için ayrı yörüngeler oluşturmak üzere A* algoritmasını geliştirilmiş, ikinci adımda ise oluşturulan yörüngenin doğrusal bir denetleyici ile izlenmesi sağlanmıştır. Bölük et al., (2019), mobil robotların otonom navigasyon probleminin çözümüne nasıl yaklaşıldığına ve pekiştirmeli öğrenmenin ortaya çıkma nedenine odaklanmıştır. Bu kapsamda navigasyon problemi için pekiştirmeli öğrenmenin Q-Learning, SARSA (State-Action-Reward-State-Action), Derin Q-Ağı, İkili Derin Q-Ağı ve A3C algoritmalarını karşılaştırarak incelemişlerdir.

Bayram (2013), çalışmasında robotlar arası iş birliği koordinesi problemini; etkili iletişim ağı topolojilerine ulaşma sorunu, bir yardımcı robot bulma sorunu

ve bu sorunun daha genel bir versiyonunu ele almak üzere problemi üç kritik açıdan incelemiştir. Revelo et al. (2014), iç mekânlardaki konumlandırma için robot kameraları ile alınan görüntülerin kullanılarak üç boyutlu koordinatların algılanabilmesini sağlayan stereoskopik görüş yöntemi, odometri ve Kalman filtresi tekniklerini birleştirmişler ve stereoskopik görüş ile odometri verilerini Kalman filtresi ile optimize etmişlerdir. Bu sayede iç mekânlarda algılama eşliğini artırmışlardır. Geliştirdikleri algoritma ile mobil robotlarda iç mekânlarda daha doğru konumlandırma sağlamışlardır. Budak, (2015), otonom hareket edebilen, birbiriyle kablosuz haberleşebilen ve üzerindeki yük hücresi sensörü ile ağırlığı algılanan yükü platform dışına taşıyabilen robot sistemi üzerine çalışmıştır. Çoklu robot sisteminde yüklerin taşınması görevi eğri uydurma yöntemi ile en kısa yol bulunarak deneylerini gerçekleştirmeyi hedeflemiştir.

Prorok et al. (2017), verilen bir dizi görevleri heterojen robotlar arasında dağıtma sorununu incelemiştir. Görev dağıtım problemini çözmek için, merkezi ve merkezi olmayan yöntemler geliştirmişlerdir. Mevcut robot dağıtımının bir işlevi olarak geçiş hızlarının çevrim içi olarak uygulanmasını sağlayan gerçek zamanlı bir optimizasyon yöntemi önermişlerdir. Dönmez & Kocamaz (2019), çalışmalarında birden fazla robotun çok sayıdaki hedefleri gezinmesi amacıyla bir görev paylaşımı ve yük dengeleme sistemi geliştirmişlerdir. Hedefler ve robotlar renkli şeritlerle etiketlenmiş olup, nesne konumları renk bazlı niceleme ve eşikleme yöntemleri kullanılarak hesaplanmıştır. Wang and de Silva (2008), çok robotlu nesne taşıma görevi için engelli ve hareketli ortamlarda çoklu robotlar için koordine problemi üzerine çalışmışlardır. Robotlar nesneyi, iş birliği içinde engellerden kaçarak istenilen yere götürmeye çalışırken makine öğrenimi olarak RL ve genetik algoritmalar (GA'lar) yöntemi kullanmışlardır. Q-öğrenme algoritması ise robotlar arasındaki iş birliği veya koordinasyon gerektiren görevlerde kullanılarak, bu robotların nasıl davranması gerektiğini öğrenmelerine ve uygun eylemlerde bulunmalarına yardımcı olunması için kullanmışlardır. Park et al., (2021), birden fazla robotun görev tahsisi üzerine çalışmışlardır. Pekiştirmeli öğrenme yöntemi ile robot ve görevlerin sayısı arttıkça kötüleşen kontrolü, Markov karar süreci formülasyonu ile çözüme ulaştırmaya çalışılmıştır. Ortaya çıkacak problemlerdeki yol planlaması için derin pekiştirmeli öğrenme yöntemini önermişlerdir.

Surmann et al. (2020), kapalı ortamlarda gerçek otonom mobil robotlar için derin güçlendirme tekniği olan Derin Öğrenme Algoritması (Deep Learning Algorithm - DRL) kullanılarak otonom mobil robotların kapalı alanlarda hareket edeceği en uygun yolu belirlemek için çalışılmıştır. Makale, DRL yoluyla otonom mobil robotlar için en uygun rotaların nasıl belirleneceğine dair bir analiz sunmaktadır. Ayrıca, DRL yoluyla yapılan en uygun yolu belirleme sonuçlarının diğer yol belirleme yöntemleriyle karşılaştırılması yapılmıştır. Makalede, DRL'nin mobil robotlar için yol belirleme için uygun bir yöntem olduğu değerlendirilmiştir.

2.2 Festo Robotino Mobil Otonom Robot ile Yapılan Çalışmalar

Tang and Khaksar (2016), yapılandırılmamış iç ortamlarda, engellerden kaçınarak, zSlices tabanlı genel tip-2 bulanık kümeler konseptini kullanarak gerçek zamanlı bir yerel yol modifikasyonu ve hareket planlama sistemi ile hedef konfigürasyonu ROMR'ye (Robotino çok yönlü mobil robot) ulaşma yeteneğine sahip olduğunu gösteren bir çalışma yapmışlardır. Abdo et al. (2020), görsel odometrinin kendine ait bazı sorunlarını göz önüne alarak, çeşitli bozuklukların görsel odometri üzerindeki etkileri üzerine bir çalışma gerçekleştirmiştir. Yapılan deneyler, tasarlanan sistem kurulumunun değiştirilmesinin otonom sürüş sisteminin genel kalitesini ve performansını nasıl etkileyeceğini değerlendirmek için tasarlanmıştır.

Tang and Eberhard (2014), Robotino robotun konum kontrolü için Robotino odometrisi verilerine bağlı olarak eksen kaçıklıklarından doğan konumlandırma problemini düzeltebilmek için harici Kuzey Yıldızı sensör sinyallerini temel alan dağıtılmış bir genişletilmiş Kalman filtresi kullanmış ve bu konumlandırmadaki kaçıklıkların çözülmesi için çalışmışlardır. Oltean et al. (2010), her yöne hareket edebilen robotlar için bulanık mantık yöntemini kullanarak dinamik ortamlarda konum kontrolü tasarımını gerçekleştirmiş, Festo Robotino platformunda ve MATLAB ortamında test etmişlerdir. Belda and Jirsa (2021), endüstriyel mobil robotlara uygulanan Endüstri 4.0 kapsamında Wi-Fi ve TCP/IP protokol standardı kullanarak kablosuz olarak haberleşmeyi, kontrol ilkelerini ve araçlarını tanıtmış ve örnek bir uygulama ile işlevselliğini göstermişlerdir.

Anggraeni et al. (2020), Festo Robotino için, ürün teslim etme görevini gerçekleştirmek için bir robotik kol manipülatörü kullanan mobil manipülatör robotlar için kontrol algoritmalarının geliştirilmesine ve bunların Gazebo-ROS (Robot Operating System – Robot İşletim Sistemi) ortamında uygulanmasına odaklanmışlardır. Mukhlisin et al., (2016), bir otomatik depolama ve alım sistemi için Robotino kullanarak depolama ve alma sistemi tasarımı sunmuşlar ve bu sistemin performansını değerlendirmişlerdir.

Literatürde Festo Robotino ile yapılan çalışmaları incelediğimizde, robotun tekerlerini harekete geçiren motorlara bağlı olan enkoderlerden alınan konumlandırma bilgileri odometri tarafından okunduğunda x, y ve phi bilgilerinde kaçıklıklar olduğu ve bu soruna sisteme sonradan eklenen sensörler ile çözüm bulunmaya çalışıldığı görülmüştür.

3 FESTO ROBOTİNO MOBİL OTONOM ROBOTU

3.1 Robotino Mobil Robot

Bu tez çalışmasında Festo Didactic firmasının geliştirdiği bir mobil robot olan Robotino kullanılmıştır. Festo Robotino bir mobil robot platformudur ve kontrol işlevlerini kendi gömülü bilgisayarında 64 bitlik Linux Ubuntu 18.04 LTS işletim sistemi üzerinde çalıştırır. Bu gömülü bilgisayar güçlü bir işlemci olan Intel Atom çift çekirdekli işlemcisi ile çok sayıda sensör ve aktüatörü birleştiren robot platformunu oluşturur. İşlemci, gerçek zamanlı işletim sistemi üzerinde çalışır ve platformdaki sensörler ve aktüatörler arasında güçlü bir iletişim hattı mevcuttur. Bu mobil robotun, her yönde serbestçe hareket etmesine olanak tanıyan birbirinden 120° aralıklı üç adet çok yönlü tekerlekleri (omni-wheel) vardır. Harici modüller eklenerek işlevselliği genişletilebilmektedir. Robotino, açık kaynak bir robottur ve kendi programlama yazılımı kullanılarak programlanabildiği gibi C, C++, Java, .Net, MATLAB, LabVIEW ve Microsoft Robotics Developer Studio kullanarak da yazılım geliştirilebilir ve ROS desteği de vardır. Robotino ile masaüstü bir bilgisayar arasında kablosuz alan ağı (WLAN) kurularak Robotino View platformu aracılığıyla tüm sensör verilerine erişilebilir ve aynı zamanda denetleyici komutları Robotino'ya gönderilebilir. Festo Robotino, mobil robot araştırmaları için bir platform olarak tasarlanmış olup akademik ve endüstriyel uygulamalara uygun bir ortam sağlamaktadır.

Şekil 3.1'de Robotino 3'ün genel görünümü yer almaktadır. “Robotino 3” adı, Robotino serisinin üçüncü nesil bir modeli olduğunu belirtir. Her nesil, öncekine göre daha fazla yetenek ve işlevsellik eklenerek geliştirilmiştir. Tablo 3.1'de Festo Robotino 3 mobil robotun teknik özellikleri, elektronik donanım parametreleri ve değerleri verilmiştir.



Şekil 3. 1 Robotino mobil robot önden görünümü

Robotino robot, dahili bilgisayarına ve diğer işlevlere erişim sağlayan bir ekranla donatılmıştır. Giriş/çıkış arayüz kartına bağlanan ekran ünitesinin altında bir kontrol kartı bulunmaktadır. Her tekerlek, artımlı bir enkodere sahip bağımsız bir motor ile tahrik edilir. Robotino'nun çevresinde 40 derece aralıklarla yerleştirilmiş dokuz adet (IR1- IR9) kızılötesi mesafe sensörleri bulunur. (Festo Didactic SE, 2022a)

Tablo 3. 1 Robotino parametreleri ve değerleri

Parametre	Değer
Besleme Gerilimi	24V DC – 4.5A
Dijital Girişler	8 adet
Dijital Çıkışlar	8 adet
Analog Girişler	8 adet (0 – 10V)
Röleli Çıkışlar	2 adet
Boyutlar	Genişlik: 43 cm, Uzunluk: 43 cm, Yükseklik: 28 cm
Ağırlık	9 kg
İşlemci	Intel Atom çift çekirdekli işlemci
RAM Kapasitesi	1 GB
Depolama Kapasitesi	32 GB dahili depolama
Güç	Dahili akü ile yaklaşık 4 saat çalışma süresi
Bağlantı	Wi-Fi, Ethernet ve diğer bağlantı seçenekleri
Sensörler	Kızılötesi sensörler, sonar sensörler, LIDAR sensörler ve kameralar
Programlama Desteği	C, C++, Java, .Net, MATLAB, LabVIEW

Kızılötesi sensörler, uzaklığı veya nesnenin yerini ölçmek için kızılötesi ışık dalgalarını kullanır. Kızılötesi (infrared – IR) LED'ler ve IR algılayıcılar ile çalışan ve yansıyan ışığı algılayan bu sensörler hareket takibi, mesafe ölçme, nesne tanıma ve evrensel olarak birçok robotik uygulamalarda kullanılır. Festo

Robotino platformundaki kızılötesi sensörler mobil robotların çevresindeki nesneler ve engellere karşı uyarı vermeleri ile engellerden kaçınarak robotun yolunu takip etmesini sağlarlar.

Dijital girişler, robotun diğer cihazlarla iletişim kurmasını sağlar. Robotino 3'te bulunan 8 adet dijital giriş robotun butonlara, sensörlere veya diğer basit girişlere bağlanmasını sağlar. Dijital çıkışlar ile de robotun başka cihazları kontrol etmesi sağlanır.

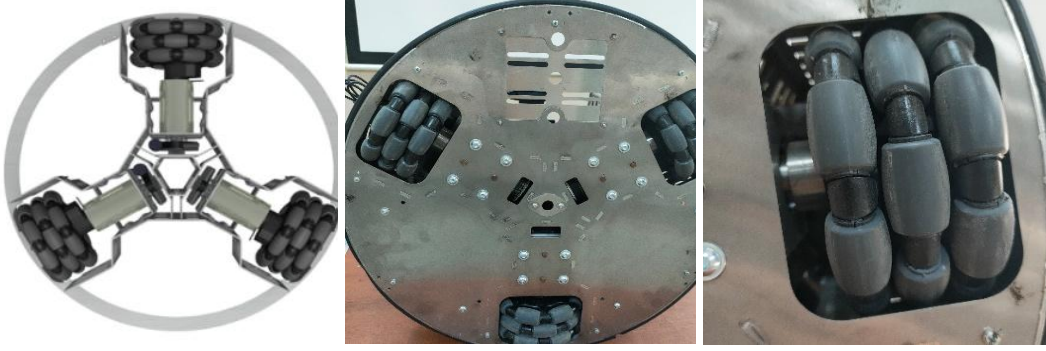
Robotino 3'te bulunan 8 adet analog giriş, robotun ölçülen sıcaklık, ışık, basınç veya diğer çevresel değişkenleri değerlendirmesini sağlar. Analog çıkışlar ise robotun motorlarını veya başka cihazları kontrol etmek için gerekli analog sinyalleri üretir. Robotino 3 robotunun analog ve dijital giriş/çıkışları Şekil 3.2'de gösterilmiştir.



Şekil 3. 2 Robotino robotun dijital ve analog giriş/çıkışları

3.2 Festo Robotino Mobil Robotun Tekerlek Yapısı

Robotino, Şekil 3.3'te görüldüğü üzere üç bağımsız çok yönlü tekerlek (omni-wheel) ile sürülmektedir. Bu tekerlekler birbirleri ile 120° açı yapacak şekilde monte edilmiştir. Üç sürüş ünitesinin her biri DC motor, artımlı enkoder, omni-wheel, 16:1 oranlı redüktör ve tırtıllı (dişli) kayış bileşenlerden oluşmaktadır.

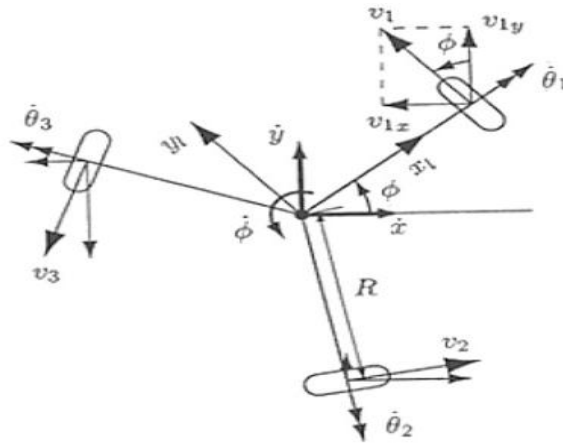


Şekil 3. 3 Robotino mobil robotun çok yönlü tekerlek yapısı

Robotino mobil robotunda kullanılan tekerlek, omni-wheel olarak adlandırılır. Bu tipteki tekerlekler, birkaç adet dairesel tekerleğin özel bir konfigürasyonu kullanılarak oluşturulur ve robotun çeşitli yönlerde hareket etmesini sağlar. Robotino mobil robotunda kullanılan üç adet omni-wheel, robotun 360 derece dönmeye ve yanlara doğru hareket etmesine olanak tanır.

3.3 Festo Robotino Robotun Kinematik Modeli

Çok yönlü robotlar, yan hareketleri mümkün kılan minimum üç motorla üç serbestlik derecesine sahiptirler, kararlıdır ve her bir tekerleğin hızına bağlı olarak herhangi bir yönde hareket edebilir ve dönebilirler. Her üç tekerleğin hareketini birleştirmek; robotun ileri, geri, yana doğru hareketlerine veya dönmeye neden olur. Bu nedenlerden dolayı modellenmesi ve kontrol edilmesi zordur. Şekil 3.4'te robotun kinematik modeli görülmektedir.



Şekil 3. 4 Robotun Kinematik Modeli (Masmoudi et al., 2016).

Festo Robotino otonom mobil robotunun kinematik modeli, robotun hareketinin matematiksel olarak ifade edildiği bir modeldir. Robotun hareketini açıklamak için tipik olarak denklem sistemleri kullanılır. Robotino'nun kinematik modelini açıklamak için yön ve dönüş hızlarına dayalı bir model kullanılmıştır. Kinematik modelin matematiksel gelişimi, tekerlek ve robot hızları arasındaki ilişkiler üzerinden gerçekleştirilmiştir.

Robotino'nun hareketi için kullanılan parametreler:

- $\dot{\theta}_i$: Tekerleğin açısal hızı
- v_i : Tekerleğin doğrusal hızı
- r : Tekerleğin yarıçapı
- R : Bir tekerlek ile robotun merkezi arasındaki mesafe
- $\dot{\phi}_i$: Robotun açısal hızı
- φ_i : Tekerleğin açısal konumu
- $P(\phi)$: Tekerleklerin açısal hızları ile global hız vektörü arasındaki dönüşüm matrisi $(\dot{x}, \dot{y}, \dot{\phi})^T$
- i : tekerleğin indeksi ($i = 1, 2, 3$)

Robotino robotun kinematik modeli Eşitlik 3.1'de verilmiştir (Masmoudi et al., 2016).

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} -\sin(\phi + \varphi_1) & \cos(\phi + \varphi_1) & R \\ -\sin(\phi + \varphi_2) & \cos(\phi + \varphi_2) & R \\ -\sin(\phi + \varphi_3) & \cos(\phi + \varphi_3) & R \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} \quad (3.1)$$

Robot hızları, Eşitlik 3.2'de verildiği gibi $P(\phi)$ matrisini tersine çevirerek hız tekerleklerinden belirlenebilir.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = R \cdot P^{-1}(\phi) \cdot \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} \quad (3.2)$$

Burada, $\dot{x}$, $\dot{y}$ ve $\dot{\phi}$ sırasıyla robotun doğrusal hızları ve açısal hızıdır. Bu eşitlikler, tekerleklerin hızını ve robotun hareketini birbirine bağlar. Robotino'nun kinematik modeli, bu denklemleri kullanarak robotun konum ve hızlarını hesaplamayı sağlar. Robotun belirli bir tekerlek hızı ile ne kadar ilerleyeceğini veya dönme açısının ne kadar olduğunu hesaplamak için bu denklemleri kullanırız. Bu, robotun otonom hareketini ve navigasyonunu anlamak için önemli bir matematiksel temel sağlar.

3.3.1 Kinematik kontrol ile hız tahmini

Kinematik kontrol, robotun hareketini matematiksel denklemler kullanarak kontrol etme yöntemidir. Robotun hedef konumuna veya hedef yönüne göre gerekli hızları tahmin etmek için kinematik kontrol kullanabiliriz.

Robotino diferansiyel tahrikli mobil robottur. Diferansiyel tahrikli robotlar, iki ve daha fazla tekerleği olan mobil robotlardır ve bu tekerlekler farklı hızlarda dönebilirler. Hedef konuma (x_{hedef}, y_{hedef}) ulaşmak ve robotun gerekli hızlarını tahmin etmek için aşağıdaki hesaplamalardan faydalanırız.

3.3.1.1 Hedef yöne doğru açı hesaplaması

Robotun mevcut konumu (x_{robot}, y_{robot}) ve açısı (ϕ_{robot}) ile birlikte, hedef konuma göre robotun yön açısını (ϕ_{hedef}) Eşitlik 3.3 ve 3.4 ile hesaplayabiliriz:

$$\phi_{hedef} = \tan^{-1}(y/x) \quad (3.3)$$

$$\phi_{hedef} = \tan^{-1}(y_{hedef} - y_{robot} / x_{hedef} - x_{robot}) \quad (3.4)$$

Açı farkını hesaplayarak, robotun hedefe yönelik ω açısal hızını Eşitlik 3.5'i kullanarak tahmin edebiliriz.

$$\omega = k_{\phi} \cdot (\phi_{hedef} - \phi_{robot}) \quad (3.5)$$

Burada k_ϕ bir açısal hız kontrol katsayısıdır. Bu katsayı, robotun ne kadar hızlı veya yavaş döneceğini kontrol eder. İdeal olarak, bu katsayı robotun dönme hızını düzenler ve hedef açığa ne kadar hızda ulaşacağını belirler. Katsayının değeri, uygulamanın gereksinimlerine ve robotun mekanik özelliklerine bağlı olarak ayarlanır.

Genellikle bu tür katsayılar deneme ve hata yoluyla belirlenir. Önceden belirlenmiş bir değerle başlanabilir ve ardından robotun davranışı gözlemlenerek katsayı ayarlamaları yapılabilir. Bu nedenle katsayının belirlenmesi, robotun dönme davranışını nasıl kontrol etmek istediğinize ve uygulama gereksinimlerinize bağlıdır.

3.3.1.2 Hedef konuma göre doğrusal hız hesaplaması

Hedefe olan uzaklığı Eşitlik 3.6 ile hesaplayabiliriz.

$$uzaklık = \sqrt{((x_{hedef} - x_{robot})^2 + (y_{hedef} - y_{robot})^2)} \quad (3.6)$$

Hedefe yaklaştıkça robotun doğrusal hızını azaltmak için bir dengeleme yapabiliriz:

$$v = k_d \cdot uzaklık \quad (3.7)$$

$$v = \min(v, v_{max}) \quad (3.8)$$

Uzaklık, robotun mevcut konumu ile hedef konumu arasındaki mesafeyi ifade eder. Genellikle bu mesafe Euclidean uzaklık olarak hesaplanır. Bu uzaklık, robotun hedefe ne kadar yakın olduğunu belirler. Burada v robotun doğrusal hızını ifade eder. Doğrusal hız, bir robotun belirli bir hedefe doğru ilerleme hızını gösterir, k_d ise bir hız dengeleme kontrol katsayısıdır. v_{max} , maksimum doğrusal hızı temsil eder ve Robotino mobil robotunun ulaşabileceği en yüksek doğrusal hızdır. Bu değer, Robotino'nun donanımı ve performansı ile ilgili bir sınırlamadır. Robotun motorları ve tekerlekleri belirli bir hızda çalışabilir ve bu nedenle

maksimum doğrusal hız, robotun güvenli ve kararlı bir şekilde hareket edebileceği en yüksek hızdır.

Hız dengeleme kontrolünde, v_{max} değeri, robotun hedef konuma doğru ilerlerken maksimum doğrusal hızının üzerine çıkmasını engellemek için kullanılır. Bu, robotun ani hareketlerden kaçınmasını ve daha kontrollü bir şekilde ilerlemesini sağlar. Eğer v_{max} değeri belirli bir değerden düşükse, robot hedefe yavaş bir şekilde yaklaşacaktır. Bu değer belirli bir değerden yüksekse, robot hedefe daha hızlı bir şekilde yaklaşacaktır, ancak bu durumda hedef pozisyonundaki eksen kaçıklıkları artacaktır. Bu nedenle, v_{max} değeri, Robotino'nun güvenli ve kararlı bir şekilde hareket etmesini sağlamak için robotun donanım sınırlamaları, çevre koşulları ve hedeflenen uygulama göz önünde bulundurularak uygun bir değer seçilmelidir. Hız dengeleme kontrolünde kullanılan v_{max} değeri, genellikle mobil robotların verimli ve güvenli bir şekilde çalışabilmesi için belirlenen bir parametredir.

3.4 Festo Robotino PC Haberleşmesi

Robotino, kablosuz haberleşme için kendi WLAN sunucusuna sahiptir. Şekil 3.5'teki gibi WLAN bağlantısı kurabilen bir bilgisayar ile Robotino mobil robotu haberleştirmek mümkündür. Aynı anda birden fazla Robotino kontrol edilirken Robotino WLAN sunucusu erişim noktası (AccessPoint – AP) modunda çalıştırılır. Her Robotino kendi ağını kuracağı için sabit IP adresini kullanan bir bilgisayardan her bir Robotino robota ayrı ayrı erişmek mümkün olmaktadır. Robotino View 172.26.1.1 adresi ile Robotino arasındaki bağlantıyı sağlamaktadır.



Şekil 3. 5 Robotino PC haberleşmesi

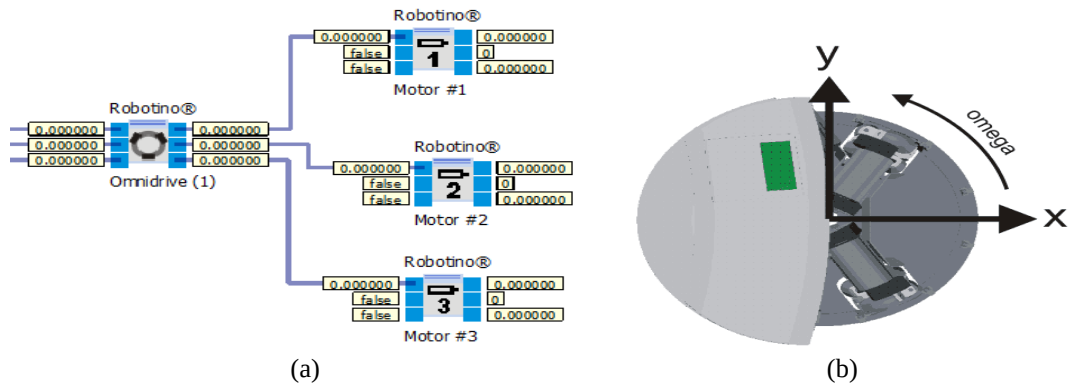
3.5 Festo Robotino View Programı

Robotino, Festo Didactic tarafından geliştirilen Robotino View programını kullanarak programlanabilmektedir. Robotino View ile robotun programlanmasında kullanılan grafik programlama dilinde fonksiyonlar ile yazılım geliştirme yapılmaktadır. Robotino View programı ile yazılan kontrol programlarının benzetimi Robotino Sim programında yapılabilmektedir. Robotino'nun mevcut LabVIEW sürücülerinin kurulumu ile robot LabVIEW üzerinden kontrol edilebilmekte ve yazılım geliştirilebilmektedir (Festo Didactic SE, 2022b).

3.6 Robotino View Programında Kullanılan Fonksiyonlar

3.6.1 Sürücü sistemi

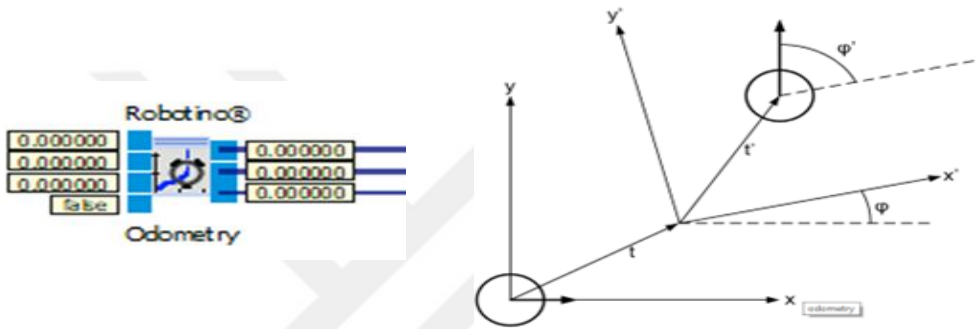
Omnidrive robotlar, 360° hareketleri sayesinde dar alanlarda, kalabalık ortamlarda ve dar yerlerde hareket etme kapasitesinin yanı sıra yerinde dönme ve karmaşık yörüngeleri takip etme yeteneği de dahil olmak üzere mobilite açısından birçok avantaja sahiptir. Ayrıca bu araçlar statik ve dinamik engellerin olduğu ortamlarda ve dar alanlarda görevleri kolaylıkla yerine getirebilmektedir. Şekil 3.6a'da görülen Robotino View programında, sürücü sistemindeki omnidrive sürücü motor 1, 2 ve 3'ün hız ayar noktalarını v_x , v_y ve omega (derece) ayar hızlarına göre hesaplar ve her bir motora erişim sağlar. Motor hızları dev/dak cinsindendir.



Şekil 3. 6 (a) Robotino View programında omnidrive sürücüsü sistemi, (b) omnidrive tekerler

3.6.2 Odometri

Şekil 3.7'deki odometri, robotun hareket ettiği mesafeyi ve yönelim değişimini kullanarak konum tahmini yapar. Odometri genellikle tekerleklerin dönüşlerini ve hareketlerini izleyen kodlayıcı sensörlerini kullanır. Bu sensörler, tekerleklerin dönüş sayılarını veya hareket mesafelerini ölçer ve robotun konumunu tahmin etmek için bu bilgileri kullanır. Odometri, robotun başlangıç konumu ve yönelimini her bir tekerleğin dönüş açıları ve kat edilen mesafeleri ölçerek belirler ve bu bilgileri sürekli olarak günceller.



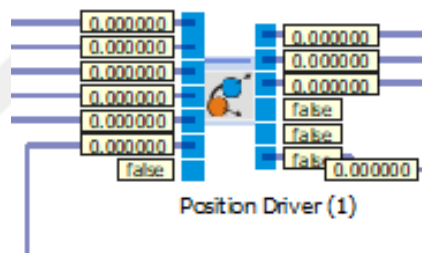
Şekil 3. 7 Robotino View odometri bloğu ve yönelim grafiği

Tekerleklerin dönüşü mümkün olan en yüksek zaman çözünürlüğü ile ölçülür. Her zaman adımında aracın kat ettiği mesafe tekerleklerin dönüş hızından hesaplanır. Bu da başlangıç konumuna göre gerçek konumu verir. Bu yöntem iyi bir yerel performans sağlar. Uzun mesafelerde veya olumsuz koşullar altında (zemindeki toz nedeniyle tekerleklerin kayması, halının tercihli yönü nedeniyle sürüklenme) bu yöntem çok büyük hatalara yol açar. Bu nedenle odometri, açıklanan hataları telafi etmek için her zaman GPS, lazer tarayıcılar, IMU gibi diğer konumlandırma yöntemleriyle birlikte kullanılarak daha doğru konum tahminleri elde etmek için entegre edilir. Odometri, mobil robotlar için konum ve hareket takibi, haritalama, navigasyon ve engel algılama gibi birçok uygulamada kullanılır.

3.6.3 Konum sürücüsü (position drive)

Konum sürücüsü, robotun konumu ile hedef konumu karşılaştırarak bir hedef konum ile mevcut konum arasındaki farkı hesaplar. Bu fark, robotun gitmesi gereken yolu ve yönü hesaplamak için kullanılır. Ardından, robotun tahrik sistemlerine (tekerlekler, eklemler, vb.) uygun kontrol sinyalleri gönderilerek robotun hedeflenen konuma doğru hareket etmesi sağlanır.

Konum sürücüsü, genellikle PID (Proportional-Integral-Derivative) veya benzeri kontrol algoritmalarıyla birlikte kullanılır. PID kontrol, hedeflenen konum ile mevcut konum arasındaki hatayı (farkı) kullanarak kontrol sinyallerini ayarlar ve hatayı azaltarak robotun istenen konuma gitmesini sağlar. Şekil 3.8'deki konum sürücüsü, Robotino'yu gerçek konumdan ayarlanan konuma sürmek için hız ve açısal hız set değerleri üretir.



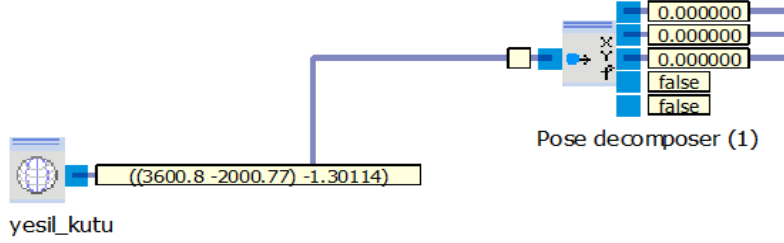
Şekil 3. 8 Robotino View odometri bloğu

3.6.4 Poz ayrıştırıcı (pose decomposer)

Bir koordinat çerçevesinin konumu ve yönelimi pozu olarak bilinir ve grafiksel olarak bir koordinat eksenleri kümesi olarak gösterilir. Kartezyen uzay içinde robotun hareket etmesi gereken bir nokta belirlediğimizde robotun hangi konuma hareket ettiği değil, aynı zamanda robotun ucuna taktığımız bir elemanın hangi yönden bir konuma işaret ettiği de önemlidir.

Pose decomposer, bir robot veya nesnenin konum bilgisini ayrıştırmak için kullanılan bir yöntemdir. Vektörel olarak gelen x, y koordinatları ve dönüş açısı (derece veya radyan) bilgilerini ayrıştırıp konum sürücüsüne gönderir.

Pose decomposer, genellikle karmaşık bir konum bilgisi verisini daha basit bileşenlere ayırarak analiz etmek ve kullanmak için kullanılır. Bu işlem, konumu ve yönelimi belirli bileşenlere bölerek daha kolay manipülasyon ve anlama sağlar. Şekil 3.9’da Robotino View poz ayrıştırıcı bloğu görünmektedir.

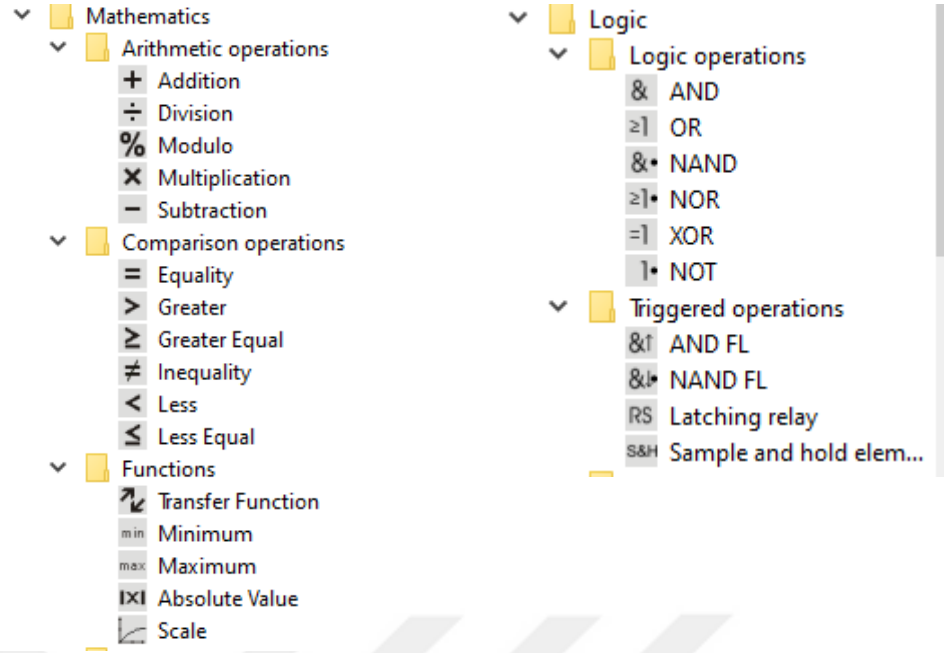


Şekil 3. 9 Robotino View poz ayrıştırıcı bloğu

3.6.5 Matematiksel ve mantıksal işlemler

Matematiksel ve mantıksal işlemler, Robotino'nun çevresini algılaması, karar vermesi, hareket etmesi ve etkileşimde bulunduğu ortama uyum sağlaması için temel bir bileşenidir. Bu işlemler, robotun görevlerini gerçekleştirme yeteneğini artırır ve karmaşık problemleri çözme becerisini sağlar.

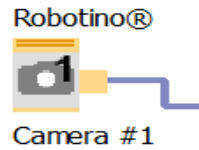
Şekil 3.10’da görülen Robotino View yazılım paketinde matematiksel işlemler için aritmetik işlemler, karşılaştırma işlemleri ile işlevler, mantık işlemleri ve tetikleme işlem blokları dışında sayıcı, multiplexer, demultiplexer blokları da bulunmaktadır.



Şekil 3. 10 Robotino View matematiksel ve mantıksal blokları

3.6.6 Kamera

Robotino'nun kamerası, görüntü işleme ve görüntü tabanlı algılama görevleri için kullanılır. Şekil 3.11'deki Robotino View kamera bloğu aracılığıyla kameradan alınan görüntü bilgileri nesne algılama, engel algılama, çizgi izleme ve renk tanıma gibi görevlerde kullanılır.



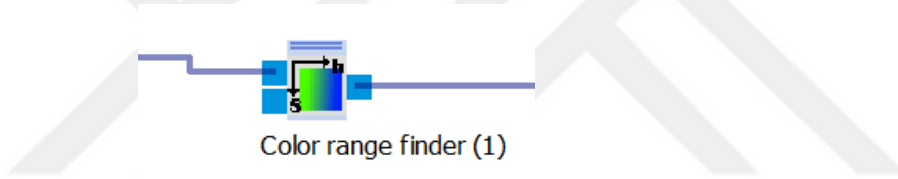
Şekil 3. 11 Robotino View kamera bloğu

Kameralar, Robotino'nun daha karmaşık ve verimli bir şekilde çevresini algılamasına ve etkileşimde bulunmasına yardımcı olur. Bu da robotun çeşitli uygulamalarda daha yetenekli ve verimli çalışabilmesini sağlar.

3.6.7 Renk aralığı bulucu (Color range finder)

Color range finder (renk aralığı bulucu), bir cihaz veya yazılım tarafından kullanılan bir teknolojidir. Bu teknoloji, bir görüntüde belirli bir renk aralığını tespit etmek veya izole etmek için kullanılır. Genellikle görüntü işleme ve görüntü analizi algoritmalarıyla birlikte kullanılır. İlk olarak, kullanıcı belirli bir renk aralığını tanımlar veya belirli bir rengi seçer. Ardından, color range finder, görüntüde bu renk aralığına uyan pikselleri veya nesneleri tespit etmek için belirli bir renk filtresi veya algoritma kullanır.

Şekil 3.12’de görülen renk aralığı bulucu, bir renk aralığı belirlemek ve bu renk aralığına karşılık gelen pikselleri bulmak veya nesnelerin belirlenmesi için kullanılır. Bu teknoloji, görüntü işleme ve renk analizi gibi birçok uygulama için önemli bir araç olarak kullanılır.



Şekil 3. 12 Robotino View renk aralığı bulucu bloğu

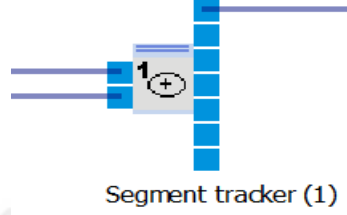
Seçilen renk aralığındaki renge sahip pikseller beyaz olarak ayarlanır. Diğer tüm pikseller siyah olarak ayarlanır. Kameradan gelen girdi görüntüsü; RGB, BGR ve HSV renk formatları geçerli giriş formatlarıdır.

3.6.8 Segment izleyici (segment tracker)

Segment tracker (bölüm izleyici), bir görüntüdeki belirli bir nesneyi veya bölümü izlemek ve takip etmek için kullanılan bir teknolojidir. Bu teknoloji, nesnenin konumunu, hareketini ve şeklini izleyerek sürekli olarak güncellemeler sağlar.

Şekil 3.13’teki segment tracker bloğu, görüntü işleme ve takip algoritmalarıyla birlikte çalışır. İlk olarak, kullanıcı tarafından izlenecek nesnenin

veya bölümün belirlenmesi gerekir. Ardından, segment tracker, nesnenin özelliklerini (renk, şekil, tekstür vb.) kullanarak nesnenin konumunu ve hareketini tespit eder. Genellikle hareketli bir kamera veya video girişi üzerinde çalışır. Kameranın görüntüsü sürekli olarak analiz edilir ve izlenen nesnenin konumu ve hareketi güncellenir. Bu sayede nesne takip edilirken çeşitli durumlara ve görüntü değişikliklerine (örneğin, arka plan değişiklikleri) uyum sağlanabilir.



Şekil 3. 13 Robotino View segment izleyici bloğu

Segment tracker, nesne takibi, güvenlik sistemleri, hareketli hedeflerin izlenmesi, otomotiv uygulamaları, etkileşimli oyunlar, sanal gerçeklik deneyimleri ve video analizi gibi birçok uygulama alanında kullanılabilir. Bu teknoloji, nesnenin veya bölümün sürekli olarak izlenmesini sağlayarak, dinamik bir ortamda hareket eden nesnelerin analizini ve takibini kolaylaştıran bir araçtır.

3.6.9 Hat dedektörü (line detector)

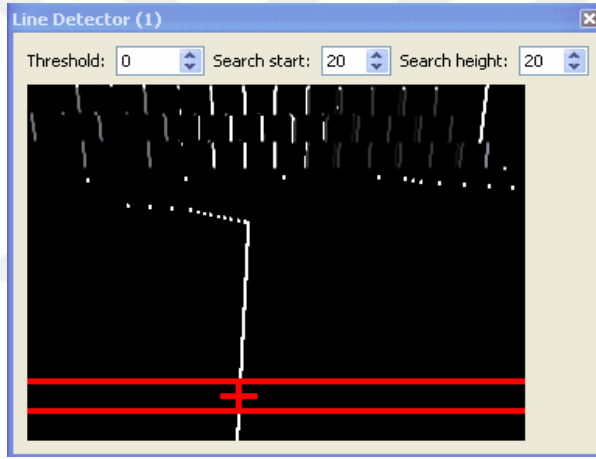
Bu sensör, robotun çevresindeki zemin üzerindeki çizgileri algılamak ve izlemek için kullanılır. Line Detector, genellikle optik sensörler veya kameraları kullanır. Sensör, zemin üzerindeki renk veya kontrast farklılıklarını algılayarak çizgileri tespit eder. Çizgiler genellikle siyah veya farklı bir renkte olabilir ve zemin üzerinde belirli bir desene veya yönlendirmeye sahip olabilir. Şekil 3.14'te Robotino View hat dedektörü bloğu görünmektedir.



Şekil 3. 14 Robotino View hat dedektörü bloğu

Robotino'nun Line Detector sensörü, çizgileri algılayarak robotun çizgi izleme görevlerini gerçekleştirmesine olanak sağlar. Çizgi izleme, robotun belirli bir çizgiyi takip ederek ilerlemesini sağlayan bir hareket kontrolü stratejisidir. Sensör, çizginin konumunu ve yönünü algılayarak robotun çizginin üzerinde doğru bir şekilde hareket etmesini sağlar.

Çizgi izleme, birçok otomatik kontrol uygulamasında kullanılır. Örneğin, sanayide üretim hatlarında, lojistik sistemlerde, otonom robotlarda ve robotik yarışmalarda çizgi izleme önemli bir beceridir. Robotino'nun Line Detector sensörü, bu tür uygulamalarda çizgi izleme görevlerini gerçekleştirmek için kullanılabilir ve Şekil 3.15'te de görüldüğü üzere robotun belirli bir çizgi üzerinde doğru bir şekilde hareket etmesini sağlar.



Şekil 3. 15 Robotino robotun hat çizgisi izlerken kamera görüntüsü

Hat aranan alan yatay kırmızı çizgilerle işaretlenmiştir. Alt çizgi "Arama başlangıcını" işaretler. Üst çizgi "Arama başlangıcı" + "Arama yüksekliği "ni işaretler. Kırmızı + soldan sağa doğru görülen çizginin koyudan açığa kenarını işaretler.

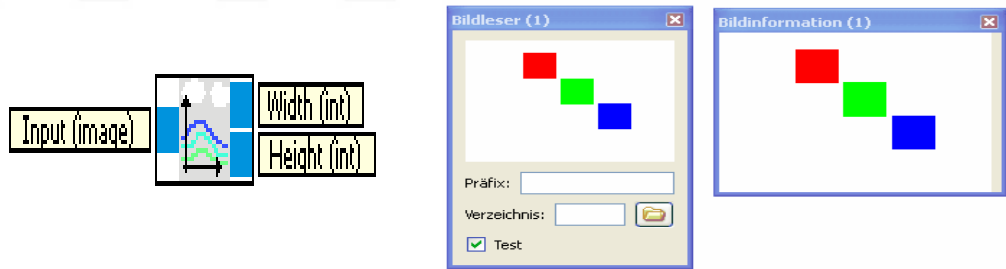
3.6.10 Görüntü bilgileri (image information)

Şekil 3.16'daki görüntü bilgisi bloğu, bir kamera veya görüntü sensörü aracılığıyla alınan görüntülerin içerdiği bilgileri ifade eder. Bu bilgiler, bir nesnenin konumu, şekli, boyutu, rengi, dokusu gibi özellikleri içerebilir. Görüntü

bilgisi, genellikle görüntü işleme teknikleri kullanılarak çıkarılan ve analiz edilen bilgileri içerir.

Robotino'nun bir kamera veya görüntüleme sistemi kullanması durumunda, görüntü bilgisi robotun çevresini algılaması, nesneleri tanınması, hareketi takip etmesi ve görevlerini gerçekleştirmesi için önemli bir kaynaktır. Görüntü bilgisi, nesne tanıma, yol takibi, renk tespiti, engel algılama, haritalama ve navigasyon gibi birçok uygulama alanında kullanılabilir.

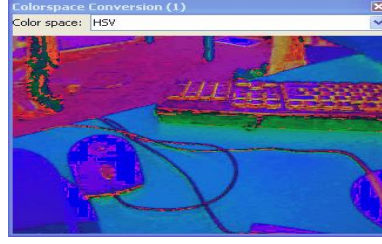
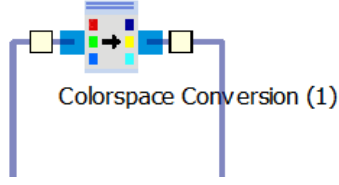
Robotino veya benzer mobil robotlar, görüntü bilgisi kullanarak çevrelerini algılar, durumu analiz eder ve hareketlerini ve görevlerini buna göre uyarlar. Bu, robotun etkileşimde bulunacağı ortama uyum sağlamasını, nesneleri tanınmasını, güvenli bir şekilde hareket etmesini ve verimli görevler gerçekleştirmesini sağlar. Görüntü bilgileri bloğu, kamera ile görüntülenen nesnenin genişliğini ve yüksekliğini baz alarak işlevini gerçekleştirir. Şekil 3.16'daki iletişim kutusu, giriş görüntüsünü gösterir. Görüntü okuyucu tarafından oluşturulan test dizisinin görüntüleri 320×240 piksel çözünürlüğe sahiptir.



Şekil 3. 16 Robotino View görüntü bilgileri bloğu ve iletişim kutusu

3.6.11 Renk uzayı dönüşümü (Colorspace conversion)

Renk uzayı dönüştürme iletişim kutusunda hedef alınacak renk seçimi yapılır. Şekil 3.17'de Robotino View renk uzayı dönüşümü bloğu görünmektedir.



Şekil 3. 17 Robotino View renk uzayı dönüşümü bloğu

3.6.12 ROI

Şekil 3.18’deki ROI (Region Of Interest – ilgilenilen bölge) bloğu ile giriş görüntüsü içinde işlem yapılacak bir bölge seçilir. Çıktı görüntüsü ROI bilgisi ile zenginleştirilir. Daha sonra görüntü işleme yalnızca ROI içinde gerçekleşir. İlgilenilen bölge fare ile işaretlenir.



Şekil 3. 18 Robotino View ROI bloğu

3.6.13 Filtre

ROI bloğunda seçilen imaj bilgisini Şekil 3.19’daki Robotino View filtre bloğu renkli ve gri tonlamalı görüntüler için farklı filtre algoritmaları sağlar.



Şekil 3. 19 Robotino View filtre bloğu

3.6.14 İşaretleyici algılama (marker detection)

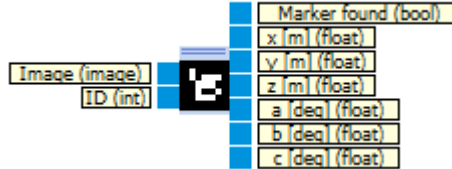
Şekil 3.20'deki Robotino View işaretleyici algılama bloğu bu fonksiyon bloğu giriş kısmından kameradaki bir görüntünün işaretçileri tespit eder. Çıkışta bu görüntünün bilgilerini alır. Bu alınan görüntüye bir kimlik bilgisi verir.



Şekil 3. 20 Robotino View işaretleyici algılama bloğu

3.6.15 İşaretleyici konumu (marker position)

Bir işaretleyici kimliği verildiğinde, bu fonksiyon bloğu kameraya göre Şekil 3.21'deki işaretleyici konum bloğu ile işaretleyici konumunu verir. Image kısmında kimlik bilgisi alınacak görüntüyü okur ve ID kısmında kimlik tespiti yapılır. Eğer tespit doğru ise x, y ve phi bilgileri belirler.



Şekil 3. 21 Robotino View işaretleyici konum bloğu

3.7 ESP32 Mikrodenetleyici Wi-Fi Kartı

ESP32, Espressif Systems tarafından üretilen bir mikrodenetleyici ve Wi-Fi kartıdır. ESP32, çok yönlü bir mikrodenetleyici olması ve internete bağlanma yeteneği ile nesnelerin interneti (IoT) cihazları ve sensör ağları gibi projeler için ideal bir çözümdür.

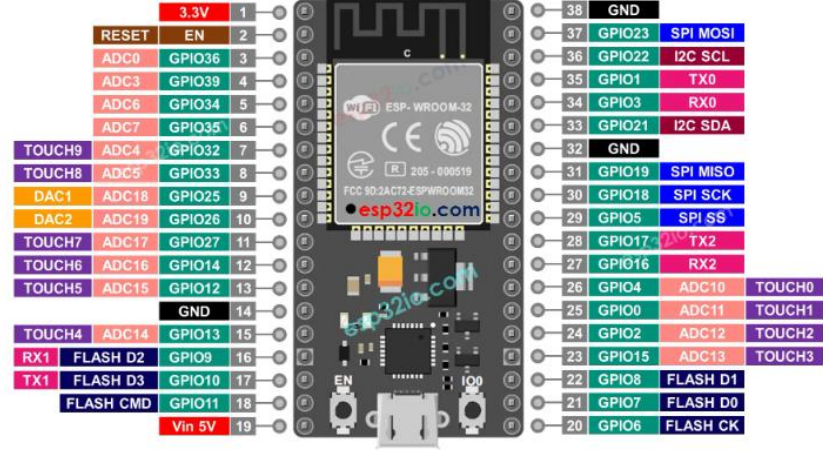
ESP32, çift çekirdekli işlemci, 1 GB RAM, 520 KB ROM ve Wi-Fi, Bluetooth ve Ethernet gibi bağlantı seçenekleri ile öne çıkan özelliklerden bazılarıdır. ESP32 ayrıca sensörler, kameralar ve diğer çevresel cihazlar ile

çalışabildiği için çok yönlü bir çözümdür. ESP32, 100 metre içinde güvenilir şekilde kullanılabilirler ve cihazlar arasında ise 480 metreye kadar uzun menzil modu vardır. ESP32'nin teknik özellikleri Tablo 3.2'de verilmiştir. ESP32 Wi-Fi kartı, uzaktan veri toplama ve kablosuz veri aktarımı için uygun bir çözüm olduğu için bu tez çalışmasında kullanılmıştır. ESP32, 802.11b/g/n HT40 Wi-Fi alıcı-vericiyi içermektedir. Bu özelliğiyle bir Wi-Fi ağı ile internete bağlanmanın dışında kendi ağını kurarak diğer cihazların doğrudan bağlanmasına izin verir. Şekil 3.22'de görülen ESP32 mikrodnetleyici kartı, Arduino 1.8.19 program arayüzü ile programlanmıştır.



Şekil 3. 22 ESP32 Mikrodnetleyici Wi-Fi Kartı

ESP32'nin çalışma gerilim aralığı 2.2V ila 3.6V olduğundan, kartın gerilimini 3.3V'ta sabit tutmak için bir gerilim regülatörü bulunur. 600mA'e kadar güvenilir bir şekilde besleme yapabilir. Radyo frekansı (RF) iletişimlerinde sırasında ESP32 250mA'e kadar akım çektiğinden dolayı bu gerilim regülatörü ve sağladığı akım değeri fazlasıyla yeterli olmaktadır. Regülatörün çıkışı da kartın 3.3V olarak etiketlenen çıkış pininden alınabilir. Bu pin, harici bileşenlere güç sağlamak için kullanılabilir. Şekil 3.23'te ESP32 mikrodnetleyici kartı pinleri görülmektedir.



Şekil 3. 23 ESP32 mikrodnetleyici kartı pinleri (<http://esp32io.com/>)

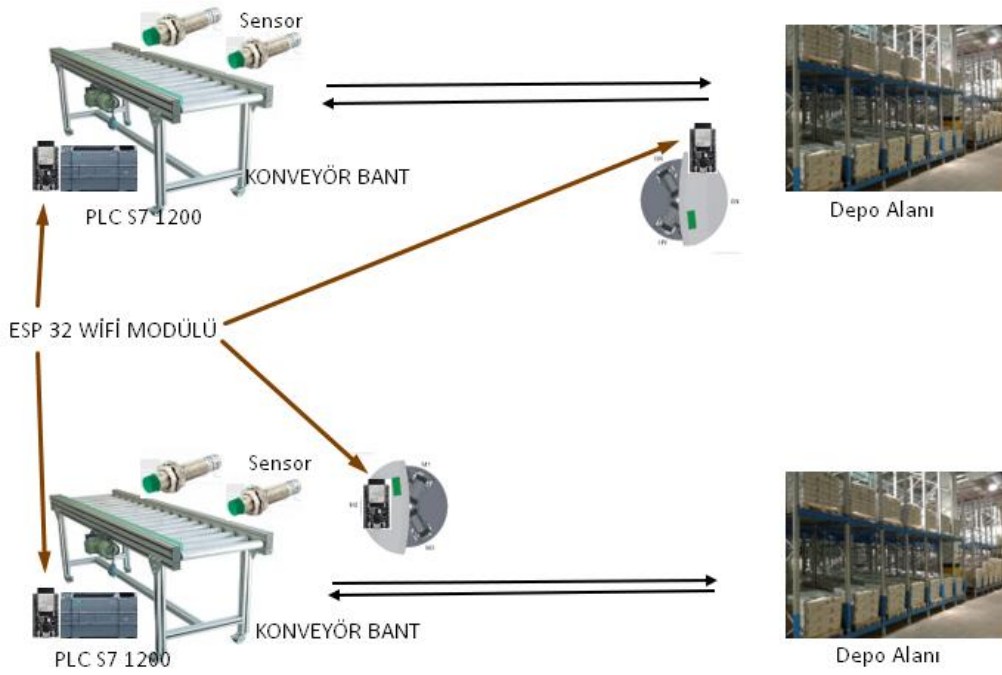
ESP32 mikrodnetleyiciler arasında kablosuz iletiřim kurmak iin birka farklı yntem bulunmaktadır, ancak en yaygın olanı Wi-Fi modl kullanarak ESP32'ler arasında Wi-Fi aėları zerinden haberleřmektir. ESP32 Wi-Fi modlleri, ESP-NOW protokol gibi alternatif iletiřim protokollerini de destekler, ancak Wi-Fi protokol en popler seenektir.

Tablo 3. 2 ESP32 Teknik zellikleri

Parametre	Deėer
İřlemci	ift ekirdekli 32-bit RISC iřlemcisi, 240 MHz
Bellek	520 KB SRAM
Kablosuz Baėlantı	802.11 b/g/n/e/i ve Bluetooth 4.2
Analog ıkıř	2 adet ADC ıkıřı
Giriř/ıkıř (GPIO) Pinleri	34
Seri UART, I2C ve SPI protokolleri Desteklenen İřletim Sistemleri	FreeRTOS, ESP-IDF ve Lua
G Kaynaėı	2.5 V - 3.6 V DC
Dřk G Tketimi	150µA modu, RX veya TX modlarında 2.8mA
alıřma Sıcaklıėı	-40°C ile +85°C

ESP32'ler arasında Wi-Fi iletiřimini kurmak iin, her bir ESP32 mikrodnetleyicinin Wi-Fi modlne sahip olması gerekir. Ardından, her bir ESP32'nin Wi-Fi modllerini bir kablosuz haberleřme aėına baėlamanız ve diėer ESP32'lerle haberleřmek iin aė zerinden IP adreslerini kullanmanız gerekir. Bu

tez çalışmasında kullandığımız ESP32 mikrodnetleyiciler ESP-NOW haberleşme protokolü ile iletişim sağlayarak herhangi bir Wi-Fi ağına bağlanmadan kendi aralarında kablosuz haberleşecek şekilde yapılandırılmıştır. Bu çalışmada iki adet Siemens S7-1200 PLC ve iki adet Festo Robotino mobil robot üzerinde kullanılmak üzere, Wi-Fi modülüne sahip dört adet ESP32 mikrodnetleyicisi kullanılmıştır. Şekil 3.24'te sistemdeki iki adet PLC ile iki adet Robotino mobil robotun aralarındaki ESP32 mikrodnetleyici ile haberleşmesi görülmektedir.



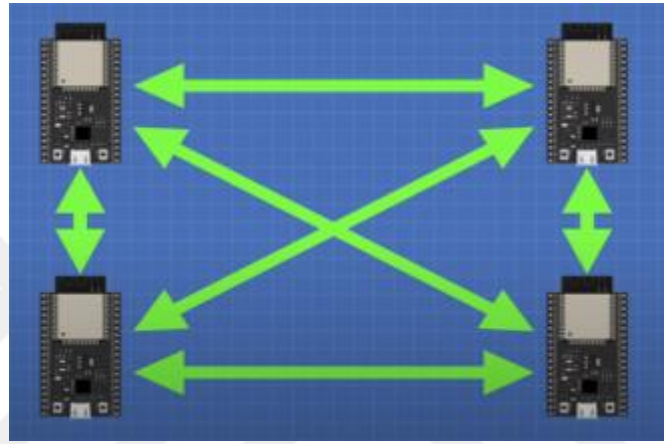
Şekil 3. 24 ESP32 Mikrodnetleyicilerin PLC'ler ile Robotino mobil robotlar arasındaki haberleşmesi

ESP32'ler arasında Wi-Fi iletişimini sağlamak için işlem adımları şöyledir:

- Her ESP32 mikrodnetleyicisinin bir Mac (Media Access Control) adresi vardır.
- Bir ESP32 mikrodnetleyicisi, diğer üç ESP32 mikrodnetleyicisine ESP-NOW protokolü ile bağlanacak ve veri gönderecek bir sunucu olarak yapılandırılır.
- Diğer üç ESP32 mikrodnetleyicisi, sunucu ESP32'ye ESP-NOW protokolünü kullanarak bağlanır ve veri alışverişi yaparlar.

- Aktif olan ESP32 mikrodnetleyicisi sunucu olup, diğr ESP32 mikrodnetleyiciler istemci konumuna geçer.

Bu işlemi gerçekleştirmek için ESP-IDF veya Arduino IDE gibi bir geliştirme ortamı kullanabilir. Bu çalışmada ESP32 mikrodnetleyici kartını programlamak için Wi-Fi ve ESP-NOW kütüphaneleri içermesinden dolayı Arduino IDE programının 1.8.19 versiyonu tercih edilmiştir. Şekil 3.25'te ESP32 mikrodnetleyicilerinin kendi aralarındaki haberleşme sistemi görölmektedir.



Şekil 3. 24 ESP32 mikrodnetleyicilerin kendi aralarındaki haberleşmesi

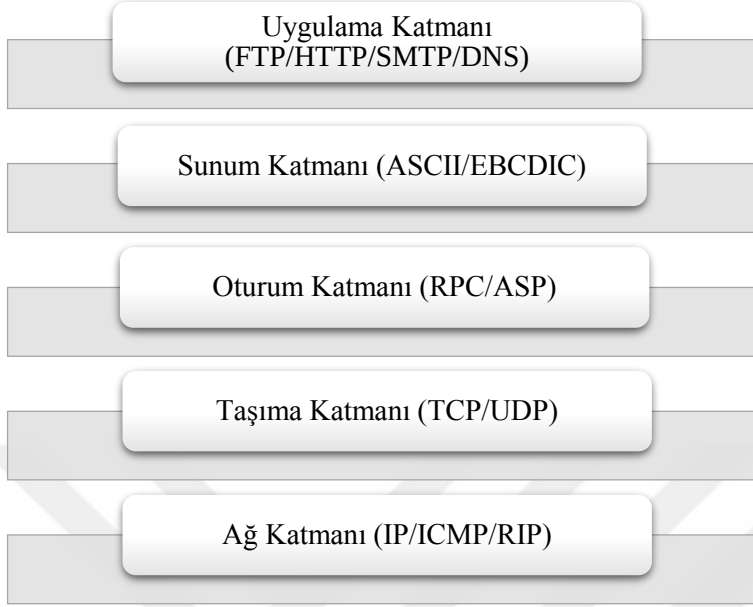
3.8 ESP32 Kablosuz İletişim Protokolleri

ESP32 birkaç farklı kablosuz iletişim protokolünü destekler. Her protokolün avantajları ve dezavantajları vardır ve uygulamaya bağlı olarak biri diğrinden daha uygun olabilir. Bu kablosuz iletişim protokolleri; BLE (Bluetooth Low Energy), Bluetooth Classic, ESP-NOW, Wi-Fi, MQTT, LoRa ve GSM/GPRS/Lee'dir.

3.8.1 ESP-NOW

ESP-NOW, Espressif tarafından tanımlanan veri bağlantısı katmanına dayanan kablosuz bir iletişim protokolüdür. ESP-NOW'ın yarattığı en büyük fark, OSI modelinin beş katmanını yalnızca bir katmana indirgemesidir. Şekil 3.26'da OSI modelinin beş katmanı görölmektedir. Ayrıca, her katmanda paket

başlıklarına veya paketleyicilere gerek yoktur, bu da sıkışık ağlarda paket kaybının neden olduğu gecikmeyi azaltarak hızlı bir yanıt verilmesini sağlar.



Şekil 3. 25 OSI modelinin beş katmanı

Ek olarak, ESP-NOW, geleneksel bağlantı protokollerinden daha az CPU ve flash kaynağı kaplarken, Wi-Fi ve Bluetooth LE ile bulunur. Çeşitli Espressif yonga serilerini destekleyerek, "bire çok" ve "çok-çok" cihazları bağlamak için uygun esnek bir veri iletimi sağlayan ESP-NOW verimli, duyarlı ve geliştirilmesi kolaydır. Bu çalışmada ESP32 mikrodenetleyiciler arası iletişim ESP-NOW kablosuz iletişim protokolü ile sağlandı.

3.9 Siemens S7-1200 PLC

Şekil 3.27’de görülen Siemens S7-1200 PLC, otomasyon endüstrisinde yaygın olarak kullanılan bir programlanabilir denetleyicidir. İşlevleri arasında ölçüm, kontrol ve izleme gibi endüstriyel süreçleri yönetmek yer alır. S7-1200, birçok farklı I/O modülü ile kullanılabilen modüler bir yapıya sahiptir. Bu sayede sistem, farklı I/O modülleriyle genişletilebilir. Ayrıca, S7-1200, Ethernet bağlantısı aracılığıyla veri iletişimi yapabilen bir denetleyici olarak tasarlanmıştır. Bu sayede, uzaktan izleme ve kontrol etme gibi işlemler de yapılabilir.

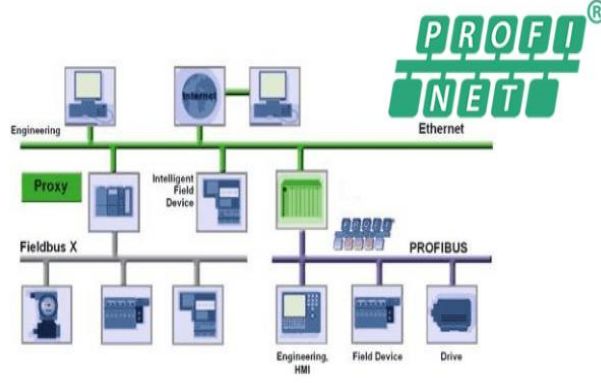


Şekil 3. 26 Siemens S7-1200 PLC

Bu çalışmada S7-1200 PLC ile TIA Portal V13 yazılımını kullanılarak sistemin çalışmasını sağlayan kontrol programı yazılmıştır. S7-1200 PLC'ler ve Festo Robotino mobil robotlar üzerlerine monte edilen ESP32 mikrodnetleyiciler aracılığıyla birbirleri ile iletişim kurmaktadır. Sensörlerden gelen veriler ESP32 mikrodnetleyici aracılığıyla PLC ve Robotino'ların yapacağı eylemlere ve işlemlere yönlendirilmesini sağlar.

3.9.1 Profinet endüstriyel haberleşme protokolü

Siemens S7-1200 programlanabilir denetleyici için kullandığımız PROFINET (Process field network), endüstriyel kontrol sistemlerinde TCP/IP ve IT standartlarını kullanan endüstriyel Ethernet destekli üst düzey açık ağ haberleşme protokolüdür. Bu protokol, otomasyon cihazları, sensörler, hareket kontrol sistemleri ve veri tabanları gibi endüstriyel bileşenler arasında güvenilir ve gerçek zamanlı veri iletişimi sağlayarak cihazlar arasında hızlı ve güvenilir bir şekilde haberleşme avantajı sunar. Farklı otomasyon bileşenleri için esnek bir haberleşme altyapısı sunması sayesinde, endüstriyel ağlar karmaşık yapılarla başa çıkabilir ve farklı cihazlar entegre edilebilir. PROFINET, otomasyon cihazlarının otomatik tanımlanmasını ve yapılandırılmasını destekleyerek sistemdeki bileşenlerin iletişime geçmesini sağlar. Ayrıca, uzaktan erişim ve cihaz yönetimi gibi özellikler de içerir (Şekil 3.28).



Şekil 3. 27 PROFINET protokolü

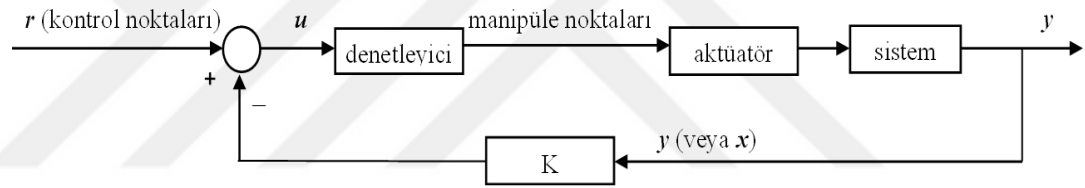
PROFINET, Ethernet tabanlı bir protokol olduğundan, mevcut Ethernet altyapılarıyla uyumludur. Bu protokol, endüstriyel ağların güvenliğini sağlamak için yetkilendirme, veri şifreleme ve ağ trafiği izleme gibi çeşitli güvenlik mekanizmalarını destekler. Aynı zamanda, ağda oluşabilecek kesintilere karşı dayanıklı olma yeteneğine sahiptir. Bu çalışmada, PLC'ler ile PC'ler arasındaki haberleşme endüstriyel Ethernet kablosu ile gerçekleştirildi.

4 KONTROL, YÖNTEM VE ALGORİTMA

Çalışmayı daha iyi anlayabilmek için, kontrol noktaları ve ölçümleri, lokalizasyon gibi kavramları tanıtmak gerekmektedir.

4.1 Kontrol Noktaları

Bu çalışmadaki kontrol noktaları, robot sürüşü için Robotino View ile yazılan algoritmada oluşturulan konumlardır. Bu konumlara giderken Robotino mobil robot, çalışma ortamındaki sabit ve değişken engellere göre engelden kaçınma davranışları gösterir. Bu davranışları ölçmek ve kontrol etmek için sensörlere ihtiyaç duyulmaktadır ve bu sensörlerden alınan bilgiler referans verileri ile karşılaştırılır. Bu çalışmada, robotun dahili sensörlerinin kullanıldığı geri beslemeli kontrol sistemi Şekil 4.1’de görülmektedir.



Şekil 4.1Klasik geribeslemeli kontrol

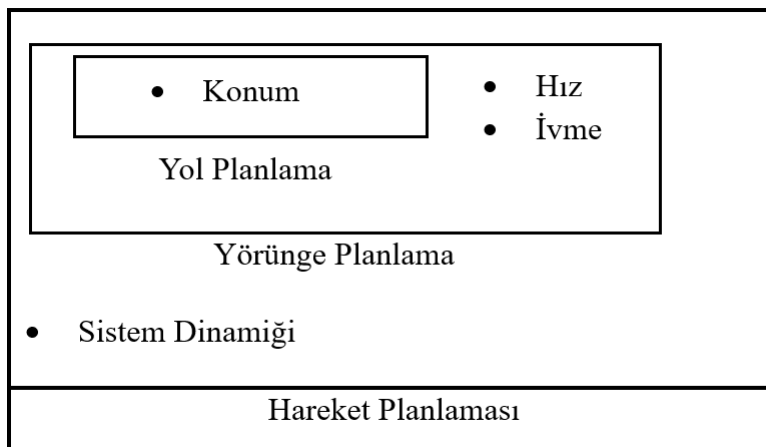
DC motorun klasik geri beslemeli kontrolü, motorun hızını veya konumunu geri bildirim yoluyla ölçerek ve istenen hız veya konum değeri ile karşılaştırarak motor kontrol sinyalini hesaplamayı içeren bir kontrol yöntemidir. Bu kontrol yöntemi, genellikle bir PID denetleyici kullanarak gerçekleştirilir. PID denetleyici, hedef hız veya konum ile geri bildirim arasındaki farka (hata sinyali) dayanarak, P (Oransal–Proportional), I (İntegral–Integral) ve D (Türevsel–Derivative) parametrelerini kullanarak motor kontrol sinyalini hesaplar.

P terimi, hata sinyalinin büyüklüğüne göre doğru kontrol sinyalini hesaplamak için kullanılan oransal kazancı ifade eder. I terimi, hata sinyalinin zaman içindeki toplamını hesaplar ve kontrollü değişkeni sıfır kararlı durum hatası ile referans değerine getirmeyi sağlar. D terimi, hata sinyalinin değişim hızını hesaplar ve yavaş değişen kontrollü değişkenin kararlı durumuna gelmesi

için gerekli sürenin kısılmasını sağlar. PID denetleyici, bu terimleri belirli katsayılarla çarparak kontrol sinyalinin hesaplar. Bu katsayılar, oransal kazanç, integral süre ve türevsel süre parametreleri değerleridir ve Robotino View program arayüzündeki bloklarda hesaplanmaktadır. Robotino mobil robotta, DC motorun hızı geri bildirim yoluyla ölçülür. Konum geri beslemesi için her bir DC motor için enkoder bulunmaktadır.

4.1.1 Konumlandırma yöntemleri ve sınıflandırılmaları

Şekil 4.2’de mobil robotlar için hareket planlaması yer almaktadır. Robotun bir hedef konuma hareket etmesi için gerekli olan yol planını ve hareket yollarını belirleyen bir süreçtir. Hareket planlaması, robotun etrafındaki ortamı algılaması, durumunu tahmin etmesi ve hareket yollarını hesaplamasını gerektirir.



Şekil 4.2 Mobil robotlarda navigasyon problemi hareket planlaması

Önce robotun, kendi konumunu ve yönelimini temsil eden x ve y koordinatları ile ϕ bilgisi gibi temel konumlandırma parametrelerini doğru bir şekilde bilmesi kritik öneme sahiptir. Bu bilgiler, robotun çevresini anlayabilmesi, istenilen hedeflere yönelebilmesi ve diğer robotlarla etkileşimde bulunabilmesi için gereklidir. Bir robot, hareket planlaması yaparken, özellikle konum belirleme, yol planlaması ve engellerden kaçınma gibi temel faktörleri göz önünde bulundurur. Tablo 4.1’de robot konumlandırma yöntemleri ve hareket planlaması verilmiştir.

Tablo 4.1 Robot konumlandırma yöntemleri ve hareket planlaması (Eberhard and Tang, 2013)

LOKALİZASYON	Göreceli	Tümdengelimli hesaplama	
	Mutlak	Görüntüleme, lazer, GPS	
YOL PLANLAMASI	Yerel	APF, Genetik Algoritmalar, Bulanık Mantık	
	Küresel	Çevre Modelleme	Grafik yöntemleri, Serbest alana yöntemler, Izgara yöntemleri
		Yol Arama	A* algoritmaları (Stentz, 1994) D* algoritmalar (Borenstein, 1998)
ENGEL KAÇINMA	VFH (Borenstein et al., 1991), APF(Khatib, 1986),VFH+(Ulrich and Borenstein, 1998),VFH (Ulrich and Borenstein, 2000)		

Lokalizasyon, özellikle robotik ve otonom araçlar gibi hareket eden sistemlerin, kendi konumlarını belirlemesi ve bilinmeyen bir çevre içinde nerede olduklarını tespit etmeleri anlamına gelir. Yerelleştirme, bir robotun, sensör verilerini ve harita bilgilerini kullanarak kendini dünya koordinat sistemi içinde doğru bir şekilde konumlandırmasını sağlayan bir süreçtir. Lokalizasyon, robotların çevrelerinde gezinmelerini ve görevlerini yerine getirmelerini mümkün kılar.

Tümdengelimli hesaplama, mobil robotların akıllı hareket planlamasında bilgi tabanlı mantık kullanarak çevresel koşullara ve hedeflere uygun kararlar almasını sağlar. Bu, robotların karmaşık çevresel koşullar altında etkili ve güvenli bir şekilde hareket etmelerine yardımcı olur. Bir mobil robotun bir engelle temas ettiği noktayı doğru bir şekilde belirlemesi önemlidir. Ancak bu, robotun algılayıcıları ve kontrol sistemlerinin hassasiyetine ve çevresel koşullarına bağlı olarak zor olabilir. Tümdengelimli hesaplama, robotun temas ettiği noktayı çevresel etkiler nedeniyle olması gereken koordinatlar bilgisini yanlış tahmin etmesi durumunda ortaya çıkar. Bu durum, robotun lokalizasyon sürecinin doğruluğunu etkilemektedir. Tümdengelimli hesaplamanın üstesinden gelmek için çeşitli teknikler ve sensörler kullanılır. Özellikle birden fazla sensörün entegrasyonu, robotun konumunu daha doğru bir şekilde belirlemesine yardımcı olabilir. Ayrıca, ileri düzey kontrol algoritmaları ve matematiksel hesaplamalar da bu sorunu azaltabilmekte veya çözebilmektedir.

4.1.2 Robot konumlandırma yöntemleri

Sensör tabanlı konumlandırmada robotlar; LIDAR kameralar, ultrasonik sensörler gibi çeşitli sensörler kullanarak çevrelerini tarar ve bu verileri kullanarak kendi konumlarını tahmin edebilirler. Bu yöntem Eşzamanlı Lokalizasyon ve Haritalama (Simultaneous Localization and Mapping - SLAM) olarak adlandırılır.

İşaretleme tabanlı konumlandırmada, önceden belirlenmiş işaretler (örneğin, QR kodları veya “fiducial” işaretler) kullanılarak robotun konumu belirlenebilir. Fiducial işaret, özellikle görüntü işleme ve konumlandırma uygulamalarında kullanılan bir referans veya kıyaslama işaretidir. Bu işaretler ölçümlerde veya karşılaştırmalarda doğruluk ve tutarlılık sağlamak için izleme, navigasyon, görüntü işleme veya veri senkronizasyonu gibi çeşitli bağlamlarda kararlı ve tanınabilir bir referans noktası olarak hizmet ederler. Bu işaretler, robotun algıladığı görüntü üzerinde bulunur ve bu görüntüden konum tahmini yapılır.

Harita tabanlı konumlandırmada ise önceden oluşturulan bir harita ile gerçek zamanlı sensör verileri birleştirilerek robotun konumu tahmin edilir. Bu yöntem genellikle GPS gibi global konumlandırma sistemleri ile entegre edilir.

4.1.3 Hareket planlaması

Hareket planlaması, robotların belirli görevleri yerine getirirken engelleri aşmasını, güvenli bir şekilde hareket etmesini ve istenen konumlara ulaşmasını sağlar. Bu hareket planlama yöntemleri düşey hareket, yatay hareket, trafik hareket, optimal hareket ve kaçınma hareket planlaması olarak beş sınıfta incelenebilir.

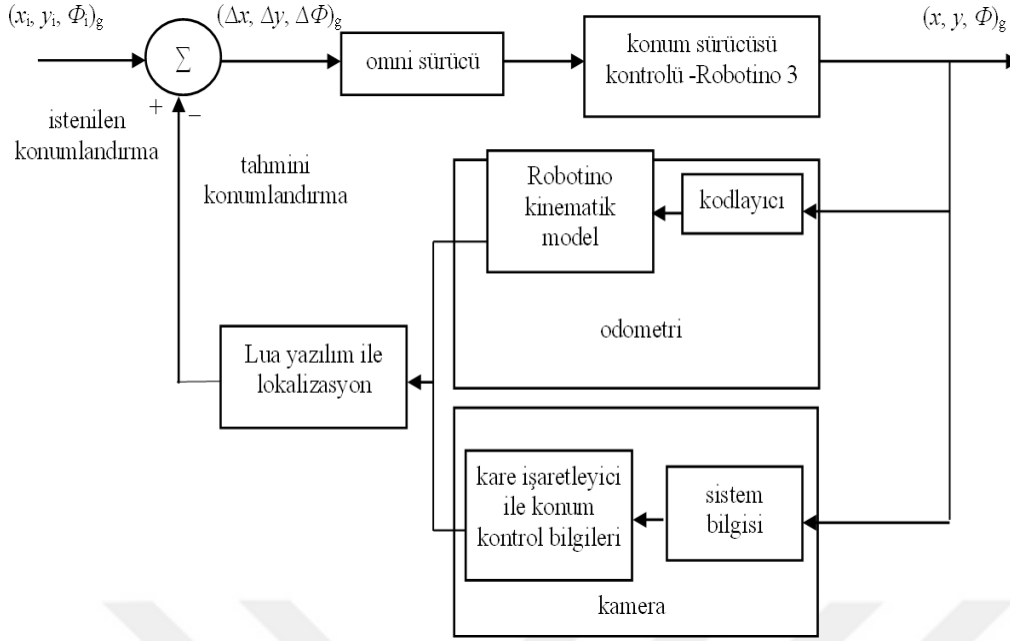
- Düşey hareket planlaması, robotun 3D uzayda yukarı ve aşağı yönlü hareketlerini planlamak için kullanılır. Bu, robotun engelleri aşmasını veya nesneleri kaldırmasını içerebilir.
- Yatay hareket planlaması, robotun yatay düzlemde ileri, geri, sola ve sağa hareketlerini planlamak için kullanılır. Ayrıca çevredeki engelleri de göz önünde bulundurarak güvenli yollar belirler.

- Trafik hareket planlaması, birden fazla robotun aynı alanda hareket ettiği durumlarda kullanılır. Robotlar, çarpışmaları önlemek ve etkili bir şekilde hareket etmek için birbirleriyle iletişim kurarlar.
- Optimal hareket planlaması yönteminde belirli bir hedefe en kısa veya enerji verimli yol üzerinden hareket etmeyi amaçlar. Genellikle matematiksel optimizasyon teknikleri kullanılır.
- Kaçınma hareket planlaması, robotun çevresindeki engelleri algılayarak onlardan kaçınmasını sağlar. Böylece robot, engelleri veya diğer nesneleri etkili bir şekilde atlatabilir.

Robot konumlandırma yöntemleri ise robotun bu hareket planını gerçek dünyada uygulayabilmesi için gereklidir. Bu iki konu bir araya geldiğinde, robotların etkili ve verimli bir şekilde çalışmasını sağlayan temel bileşenler oluşturulur. Bu tez çalışmasında Robotino mobil robotların konumlandırması için Robotino View programındaki bloklardan ürettiğimiz QR kodlarından yararlanarak robot konumu belirlenmiştir. Hareket planlaması kapsamında ise yatay hareket planlaması, trafik hareket planlaması ve kaçınma hareket planlaması uygulanmıştır.

4.1.4 Genel konum kontrol şeması

Konum kontrol şemaları, robotların, otonom araçların veya başka sistemlerin hassas bir şekilde belirli konumlarda olmasını veya belirli konumlara gitmesini sağlar. Şekil 4.3'te Robotino mobil robot için konumlandırma ve konum kontrolü blok şeması görülmektedir.



Şekil 4.3 Konum kontrol şeması

Bu şema dahili bir denetleyici ve harici bir denetleyiciden oluşmaktadır. Dahili denetleyici, Robotino DC motorlarının omni sürücüsünü, konum sürücüsü kontrolü ile doğrudan kontrol edebilen PID denetleyicidir. Harici denetleyici ise, Robotino robotun 'gerçek' konum ve oryantasyonunu Robotino'nun tekerleri üzerindeki enkoderler tarafından alınan verilerin yanında, işaretleme tabanlı konumlandırma yöntemi olan kare işaretleyici kodlarına bağlı olarak alınan konum bilgileri ve robotun odometrisi tarafından okunan konum bilgilerini karşılaştırarak, konumlandırmada oluşan kaçıklıkları Lua dilinde yazılan yazılım ile hedeflenen konumlandırmayı yapmaktadır.

Lua yazılımı ile elde edilen $[\Delta x \ \Delta y \ \Delta \phi]_g$ sapması ile robot istenilen konumlandırma çerçevesinde sürüşe devam eder ve bu ise robotların lokalizasyon doğruluğunu artırır. Her robot kendi başına çalışır ve bu şema tüm robotlar için geçerlidir.

4.2 Odometri Kaçıklık Hatası Nedenleri

Odometri, tekerleklerinin açısal dönme hızının ölçümüne dayanan bir mobil robotun göreceli yer değiştirmesinin periyodik olarak gerçek zamanlı bir tahminini sağlayan bir kilometre sayacıdır. Kilometre sayacı, enkoderin devir

sayısı ile tekerleğin çevresini çarparak doğrusal mesafeyi hesaplar. Odometreler kısa mesafelerde kısmen güvenilirdir, dezavantajı ise zaman içinde mobil robotun mutlak konumunun ve yöneliminin tahmininin doğruluğunu azaltan bir faktör olan sınırsız hata biriktirmesidir. Odometri kaçıklık hataları, farklı kaynaklardan kaynaklanabilmektedir. Tablo 4.2’de mobil bir robot için tüm farklı sistematik ve sistematik olmayan hataların bir incelemesi verilmiştir.

Tablo 4.2 Sistematik ve sistematik olmayan hataların özeti (Borenstein, 1998)

Sistematik Hatalar	Sistematik Olmayan Hatalar
Eşit olmayan tekerlek çapları	Düz olmayan zeminler üzerinde seyahat
Tekerleklerin yanlış hizalanması	Tekerlek kayması
Her iki tekerlek çapının ortalaması nominal çaptan farklı olması	Dış kuvvetler
Dingil mesafesi	İç kuvvetler
Kodlayıcı çözünürlüğü	Zemin ile noktasal olmayan tekerlek teması
Kodlayıcı örnekleme hızı	

- Tekerleklerin kayması veya patinaj yapması, odometri verilerinde hatalara neden olabilir. Bu durum, robotun gerçek mesafesinin tahmin edilenden daha az veya daha fazla olmasına sebep olabilir.
- Robotun yön değiştirmesi, odometri verilerinde hatalara neden olabilir. Özellikle, robotun yön değiştirdiği sırada tekerleklerin kayması, odometri verilerinin yanlış olmasına neden olabilir.
- Zeminin kayganlığı veya eğimi, odometri verilerinde hatalara neden olabilir. Bu durumda, robotun gerçek mesafesi tahmin edilenden daha az veya daha fazla olabilir.
- Farklı boyutlardaki tekerlekler, odometri verilerinde hatalara neden olabilir. Örneğin, farklı boyutlardaki tekerlekler, robotun bir yöne doğru hareket ettiğinde bir tekerleğin diğerinden daha fazla veya daha az dönmesine neden olabilir.
- Enkoderlerin doğru çalışmaması, odometri verilerinde hatalara neden olabilir. Enkoderler, zamanla aşınabilir veya kirlenebilir, bu da hatalara neden olabilir.

- Robotu besleyen akünün enerjisinin azalması da odometri okumasını etkileyen bir faktördür.

Odometri hatasına neden olabilecek dış etmenler, robotun engelden kaçamayarak bir nesneye çarpması veya bir nesnenin robota çarparak onu rotasından çıkarması gibi önceden tahmin edilemeyen olaylardır. Aynı etkiye sahip olabilecek iç kuvvetler ise robotun ileri veya geriye doğru hareket halindeyken tam tersi yön değiştirme hareketinde tekerleklerin zemin üzerinde kayması ile robotu hafifçe rotasından çıkarır. Sistematik hatalardan birinin tamamlayıcısı, tekerleğin zeminle noktasal olmayan temasıdır. Bu durumda, tekerlek ile zemin arasında var olan sürtünme miktarı da kaçıklık hatalarına neden olmaktadır (Epton, 2007).

Odometri hatalarının önlenmesi ve robotun konumunu belirlemek için farklı sensörlerin bir arada kullanılması önerilir. Örneğin, lazer tarayıcılar, jiroskoplar veya manyetik pusulalar gibi sensörler odometri verileri ile birleştirilerek daha doğru bir konum belirleme sağlanabilir. Bu çalışmada Robotino mobil robot üzerinde mevcut olan jiroskopu devreye alarak sistematik olmayan hataları azaltmak hedeflendi.

Birçok araştırmacı belirli hataların azaltılabileceği veya ortadan kaldırılabileceği düşüncesi ile mobil robotların odometri hatalarını incelemişlerdir. Mobil bir robotun konumunu bilmek navigasyon için hayati önem taşıdığından ve odometrinin de hatalı sonuçlar vermesinden dolayı bir sonraki adım, kilometre sayacını düzeltmeye yardımcı olmak için robota başka bir sensör veya sensörler eklemek olmuştur.

Gargiulo et al., (2021), tekerleklerden ve bir Atalet Ölçüm Biriminden (Inertial Measurement Unit – IMU) toplanan bilgileri birleştirerek mobil robotun konumunu ve yönünü tahmin etmişlerdir. IMU, ana işlemciye gönderilen açısız hız ve doğrusal ivme verisini tek bir modülde toplayan elektronik bir birimdir. IMU bileşenleri; ivme ölçer, jiroskop ve bazen de manyetik alan ölçerdir. Yazarlar, bu model tabanlı yaklaşımın deneysel sonuçlarla doğrulandığını ve gezgin konumlandırmasında önemli bir iyileşme sağladığını göstermişlerdir.

Zwierzchowski et al. (2021), sapma oranı gibi robot yönelimlerindeki tüm değişiklikleri hesaplamak için bir IMU sensörünü tekerlek odometrisi ile birleştirerek kullanmalarının yanı sıra çoğunlukla tekerleklerdeki kayma ve IMU sensöründe zamanla oluşan sıfır ve açıklık hatalarından kaynaklanan ek ölçüm hatalarını da azaltmak için bir görüntüleme sisteminden yararlanmayı hedeflemişlerdir. Odometri hata oranını azaltmak için aracın kendi alanında bulunan işaretçilere olan mesafesini sürekli ölçen bir görüş sistemi kullanarak konumlama hatasını 0.32 m'den 0.13 m'ye düşürmüş ve tekerleklerdeki kaymayı engellemişlerdir.

Xue et al., (2019), geleneksel LIDAR odometrisi yaklaşımlarına alternatif olarak, frekans tabanlı bir yöntem önermişlerdir. Yaptıkları çalışmada ilk olarak, aracın kendi hareketinden kaynaklanan LIDAR taramasının çerçeve içi bozulmasını düzeltmek için IMU bilgilerini kullanmışlardır. IMU, LIDAR odometri için daha iyi bir başlangıç tahmini sağlar ve IMU, genişletilmiş Kalman filtre çerçevesinde LIDAR odometri ile kaynaştırılmıştır.

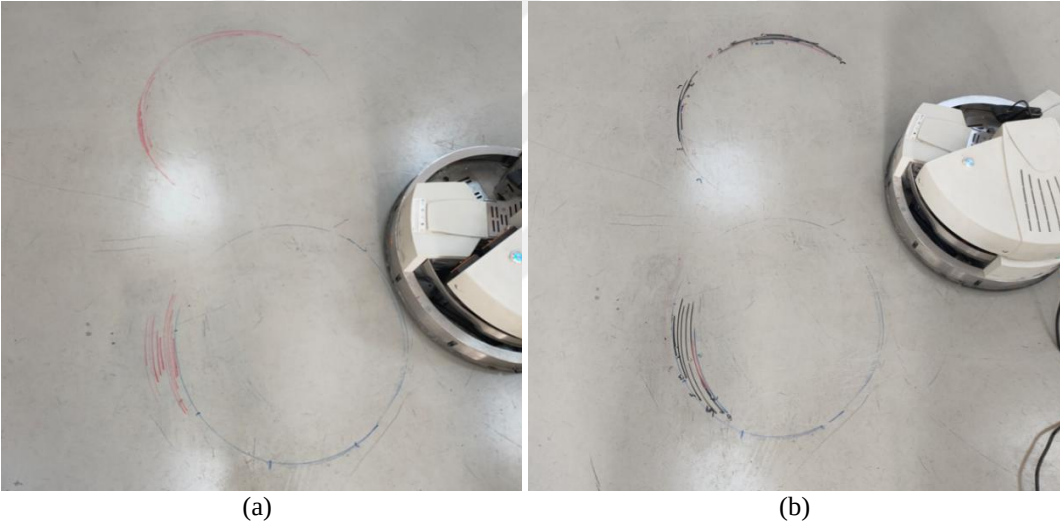
Palacjin et al. (2020), iç ve dış mekân SLAM'ında konumlandırmadaki kaçıklıkların doğruluğunu, haritalama yapılan alanın kullanımı ve çarpışma problemini çözmek için çift LIDAR atalet odometrisini kullanmışlardır. Çift LIDAR verisini, odometrinin hareket bozulmasını ortadan kaldırmak için bir IMU ile birleştirmişlerdir. Önerilen yöntemin daha ayrıntılı olarak daha kesin lokalizasyon ve daha doğru haritalama sonuçları üretebileceğini belirtmişlerdir.

4.3 Laboratuvar Ortamında Robotino Odometri Hataları Belirlenmesi

Robotino View arayüzünde yazılan algoritma simülasyon ortamında beklendiği gibi çalışırken laboratuvar deney ortamında Robotino mobil robot ile gerçek bir sürüşte çalıştırıldığında istenilen koordinatlara ulaşmadığı ve odometriden gözlemlediğimiz x , y ve ϕ (derece) değerlerinde de kaçıklıklar olduğu görülmüştür. Robotino mobil robottaki bu kaçıklıkların nedeni, mobil robotun harekete başlama, durma ve dönüşlerindeki omni-wheel tekerleklerinin sebep olduğu mekaniksel hatalar olabileceği gibi sürüş yüzeyindeki tozlardan

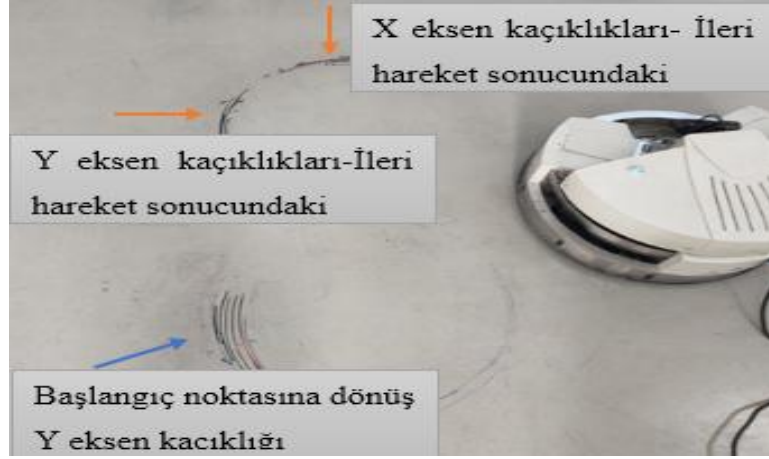
veya tekerlerin hareket ettiği yüzeyin özelliklerinden de kaynaklanabilir. Robotino mobil robotun, Robotino View’da tanımlanan pozisyon ayrıştırıcı bloğuna girilen koordinatlara gidememesinin sebeplerinden bazıları bunlardır.

Bu kaçıklık sapmalarının sistematik olup olmadığını anlamak için, ilk olarak mobil robot Robotino’nun dönüş yaptırılmadan sadece ileri geri hareketleri gözlemlendi. Deney ortamında başlangıç ve bitiş noktaları belirlendi ve önce başlangıç noktası koordinatları olarak x , y ve ϕ değerleri 0,0,0 olarak girildi ve bitiş noktası olarak ise x , y ve ϕ değerleri 500,0,0 olarak belirlendi. x koordinatındaki 500 sayısı 500mm’lik ileri doğru hareketi sağlayacaktır. Şekil 4.4(a)’da birinci 500mm ve Şekil 4.4(b)’de ise ikinci 500mm gidiş dönüş denemelerindeki kaçıklıklardan oluşan işaretlemeler görülmektedir.



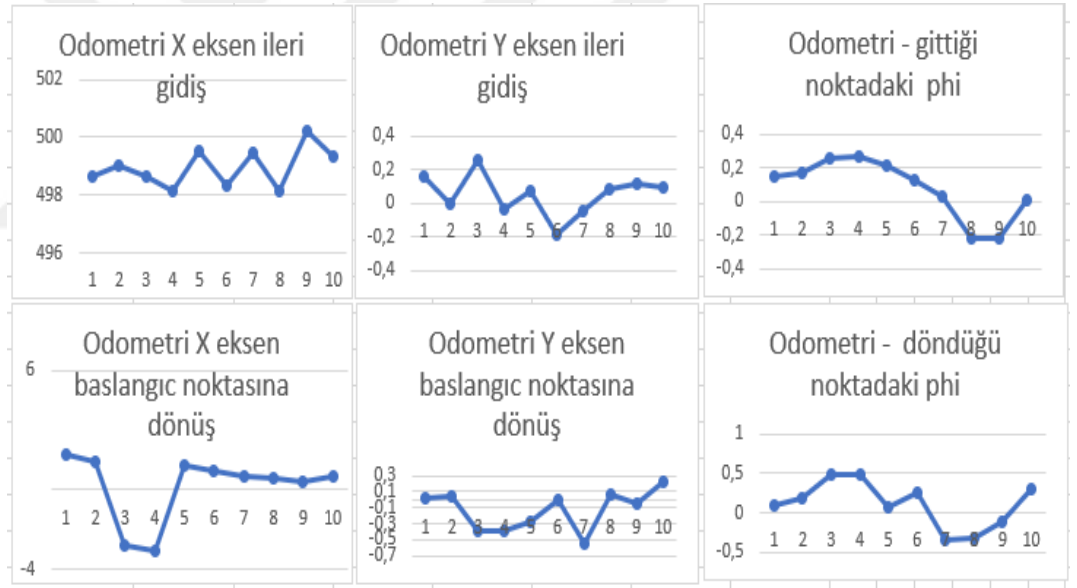
Şekil 4.4 (a) Birinci 500mm için oluşan kaçıklıklar ve işaretlemeleri, (b) ikinci 500mm için oluşan kaçıklıklar ve işaretlemeler

Kaçıklık hatalarının belirlenmesi için yapılan ilk denemede mobil robota toplamda 10 kez 500mm’lik ileri ve geri gidiş-dönüş hareketi yaptırılarak vardığı koordinatlar gözlemlendi. Bu gidiş-dönüş hareketleri bir defa daha tekrarlandı. Her defasında ileri gittiğindeki x eksen ve y eksen kaçıklıkları ve geriye yani başlangıç noktasına vardığı y eksen kaçıklıkları işaretlendi. Karşılaştırma yapmak için Robotino View programındaki odometriden gözlemlenen x , y ve ϕ değerleri de kaydedildi. Şekil 4.5’te başlangıç ve bitiş noktalarındaki ileri-geri hareketindeki kaçıklıklar sonucunda işaretlenen noktalar gösterilmiştir.



Şekil 4.5 Robotino x ve y eksen kaçıklıkları

Birinci 500mm ileri ve geri hareketleri sırasında odometriden ölçülen ve x, y ve phi kaçıklıklarını gösteren grafikler Şekil 4.6'dadır.



Şekil 4.6 Birinci 500mm ileri ve geri hareketleri sırasında odometriden ölçülen x, y ve phi kaçıklıkları

Birinci denemede sonucunda her defasında farklı kaçıklıkların olduğu gözlemlendi. Bu kaçıklıklar sağa ya da sola doğru olmaktadır. Sistematik bir hata olmadığı gözlemlenmiştir. Şekil 4.7'de Robotino mobil robotun ileri hareketindeki x, y eksen kaçıklıkları ve geriye dönüş yani başlangıç noktasına varıştaki y eksen kaçıklıklardan oluşan grafikler verilmiştir.



Şekil 4.7 Başlangıç noktasındaki y eksen kaçıklığı ve Robotino ileri gittiğindeki x ve y eksen kaçıklık ölçümleri

Ayrıca, başlangıç ve gidilecek hedef noktalarındaki odometriden alınan x, y ve phi açıklık hataları Tablo 4.3'te gösterilmiştir.

Tablo 4.3 Birinci 500mm denemesinden elde edilen veriler

KAÇIKLIK												
deneme	Başlangıç noktasından 500 mm ileri gittiğinde x ve y eksen kaçıklıkları(mm)						Robotino View 3 programı- odometri bloğu konum bilgisi					
	Robotino başlangıç kaçıklığı	sol/sağ	y eksen kaçıklığı	sol/sağ	x eksen kaçıklığı ₁	ileri/geri	x (mm)		y (mm)		phi (derece)	
							gidiş	dönüş	gidiş	dönüş	gidiş	dönüş
1	4	sol	2	sol	2	ileri	498.635	1.797	0.155	0.02	0.153	0.1
2	3	sol	2	sol	1	ileri	499.006	1.427	0.0005	0.048	0.169	0.186
3	5	sol	7	sol	1	ileri	498.671	-2.79	0.253	-0.379	0.252	0.48
4	4	sol	3	sol	5	ileri	498.132	-3.03	-0.03	-0.385	0.269	0.477
5	5	sol	1	sol	1	ileri	499.507	1.201	0.074	-0.268	0.217	0.07
6	6	sol	2	sol	1	ileri	498.347	0.977	-0.182	-0.008	0.131	0.253
7	8	sol	1	sol	1	ileri	499.475	0.711	-0.040	-0.538	0.03	-0.35
8	10	sol	1	sol	1	ileri	498.131	0.578	0.087	0.071	-0.22	-0.31
9	6	sol	1	sol	1	ileri	500.189	0.439	0.115	-0.058	-0.22	-0.10
10	9	sol	3	sol	1	ileri	499.35	0.663	0.091	0.228	0.008	0.304
Ortalama	6	sol	2.3	sol	1.5	ileri	498.944	1.361	0.103	0.2	0.167	0.264
Toplam Kayma	60	sol	23	sol	15	ileri						

İkinci 500mm ileri gidiş ve geri dönüş denemesi sırasında odometriden ölçülen ve x, y ve phi kaçıklıklarını gösteren grafikler Şekil 4.8'deki gibidir.



Şekil 4.8 İkinci 500mm ileri gidiş ve geri dönüş denemesi sırasında odometriden ölçülen x, y ve phi kaçıklıkları

İkinci deneme sonucunda Robotino mobil robotun ileri hareketindeki x, y eksen kaçıklıkları ve başlangıç noktasına dönüşteki y eksen kaçıklardan oluşan grafik Şekil 4.9'dadır.



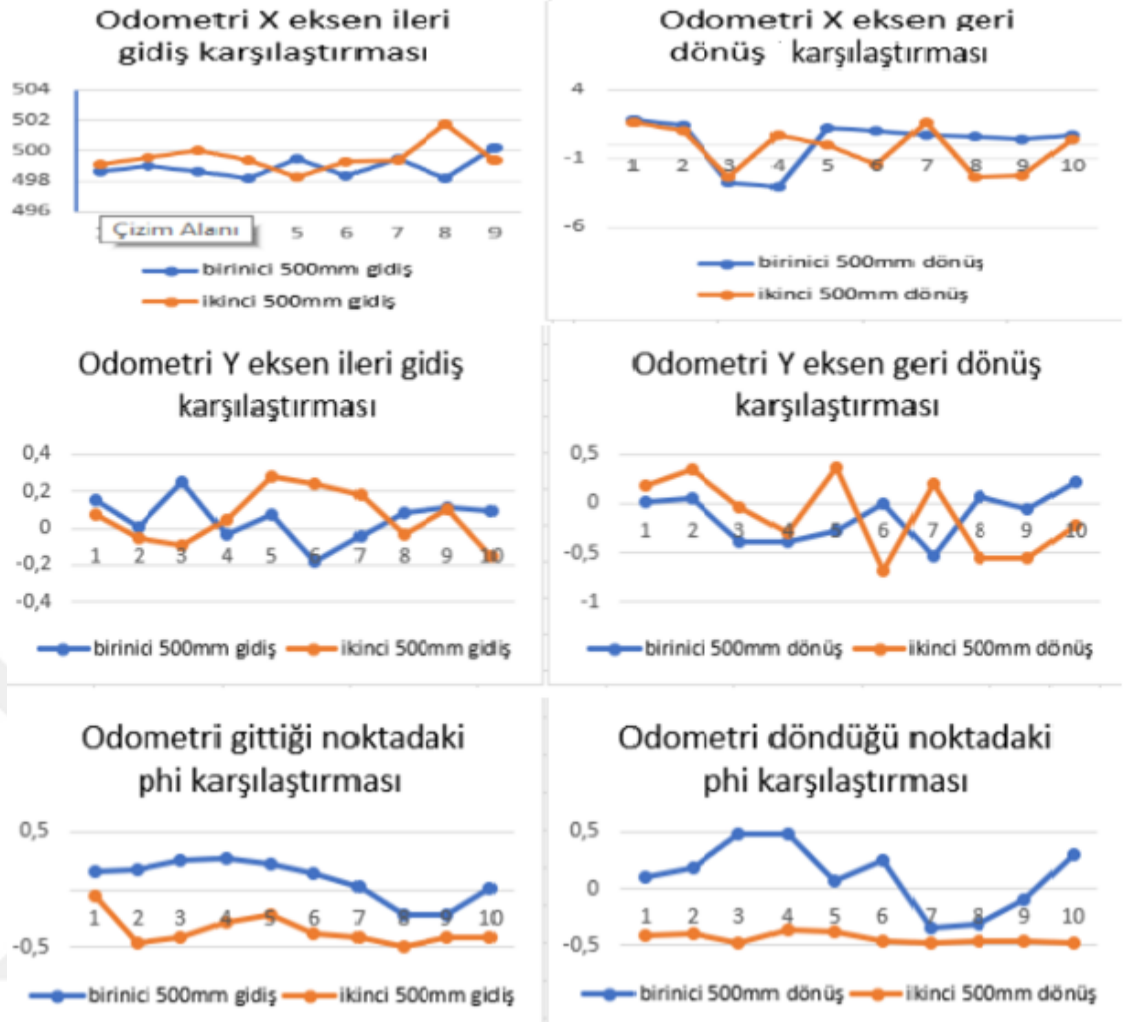
Şekil 4.9 Başlangıç noktasındaki y eksen kaçıklığı ve ileri gittiğindeki x ve y eksen kaçıklıkları

İkinci denemede ileri gidiş ve başlangıç noktasına dönüş noktalarındaki odometriden alınan x, y ve phi verileri ve hesaplanan kaçıklık hataları Tablo 4.4'te gösterilmiştir.

Tablo 4.4 İkinci 500mm denemesinden sonra elde edilen veriler

KAÇIKLIK												
	Başlangıç noktasından 500 mm ileri gittiğinde x ve y eksen kaçıklıkları (mm)						Robotino View 3 programı-odometri bloğu konum bilgisi					
deneme	Robotino başlangıç kaçıklığı	sol/ sağ	y eksen kaçıklığı	sol/ sağ	x eksen kaçıklığı	ileri/ geri	x (mm)		y (mm)		phi (derece)	
							gidiş	dönüş	gidiş	dönüş	gidiş	dönüş
1	2	sol	5	sol	1	ileri	499.047	1.612	0.072	0.19	-0.059	-0.41
2	1	sol	1	sol	5	geri	499.511	0.983	-0.052	0.344	-0.468	-0.395
3	7	sol	2	sol	1	ileri	499.978	-2.313	-0.093	-0.042	-0.421	-0.484
4	6	sol	2	sağ	4	geri	499.349	0.701	0.0488	-0.291	-0.281	-0.361
5	10	sol	1	sağ	6	ileri	498.267	0.0271	0.281	0.367	-0.226	-0.371
6	11	sol	1	sağ	7	geri	499.32	-1.48	0.245	-0.679	-0.379	-0.459
7	13	sol	2	sağ	2	geri	499.355	1.587	0.178	0.194	-0.406	-0.48
8	8	sol	6	sağ	3	geri	501.804	-2.349	-0.033	-0.557	-0.493	-0.467
9	2	sağ	6	sağ	2	geri	499.331	-2.244	0.103	-0.552	-0.417	-0.46
10	5	sağ	2	sol	2	ileri	499.324	0.419	-0.154	-0.225	-0.421	-0.477
Ortalama	6.5		2.888889		3.3		499.529	1.372	0.126	0.344	-0.357	-0.436
Toplam Kayma	65		28		33							

Bu deneylerin sonuçları birleştirilip karşılaştırıldığında sistematik bir hata olmadığı sonucuna varılmıştır. Şekil 4.10'da odometriden alınan x, y ve phi verileri ve başlangıç noktası y eksen, ileri gidiş x ve y eksen kaçıklıklarının karşılaştırmalı grafikleri sunulmuştur.



Şekil 4.10 Odometriden alınan x, y ve phi verilerinin karşılaştırması

Şekil 4.11'deki grafiklerde ise iki deneme sonucunda Robotino mobil robotun başlangıç noktasına geri döndüğündeki y eksen, ileri gittiğinde x ve y eksen kaçıklıklarının ölçümleri karşılaştırılmıştır.



Şekil 4.11 İki deneme sonucunda Robotino mobil robottun eksen kaçıklık karşılaştırmaları

Bundan sonraki denemeler için başlangıç noktası koordinatları x , y ve ϕ değerleri 0,0,0 ve varış noktası koordinatları x , y ve ϕ 1000,0,0 olarak girildi. x koordinatındaki 1000 sayısı 1000mm'lik ileri doğru hareketi belirtir. Bu kaçıklık hata denemesinde toplamda 10 kez 1000mm'lik ileri ve geri hareketlerinin x ve y eksen kaçıklıkları gözlemlendi. Bu deneme 10'ar defadan 2 defa tekrarlandı. Robotino mobil robot ileri gittiğinde her defasında x eksen ve y eksen noktaları ve başlangıç noktasına geri geldiği eksen noktaları işaretlendi. İlk 500mm'lik ileri ve geri hareketlerindeki eksen kaçıklıklarına göre daha fazla mesafede sola veya sağa eksen kaçıklıkları değişimi gözlemlendi. Bu sonuçlara göre de sistematik bir hata olmadığı görüldü. Şekil 4.12'de 1000mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar gösterilmiştir.

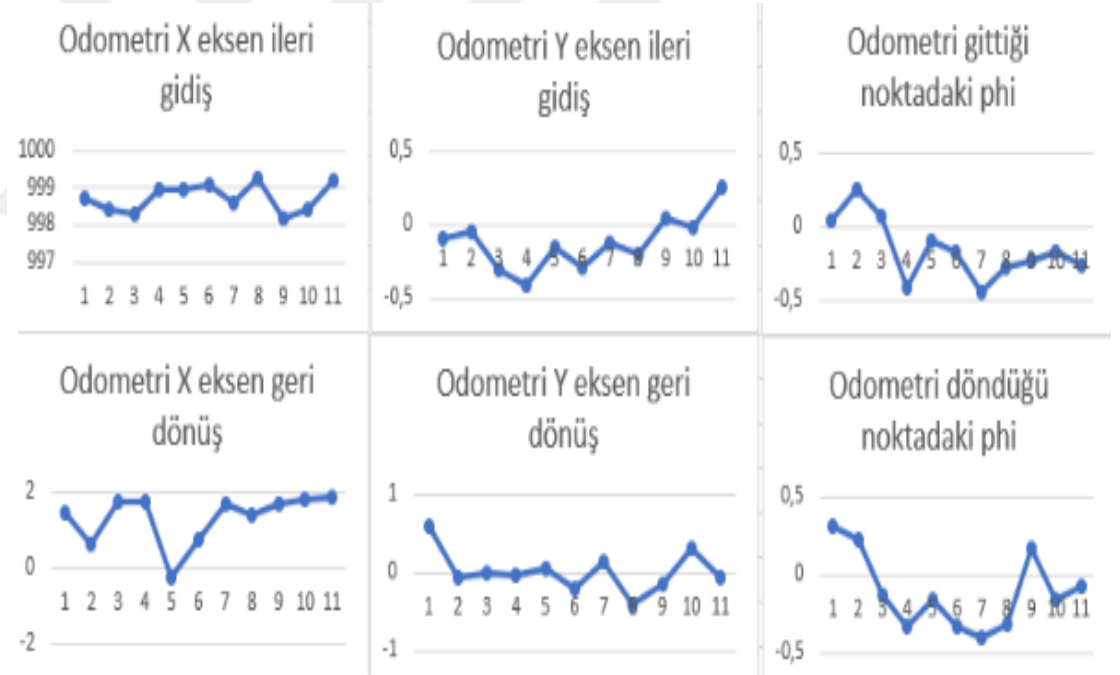


Şekil 4.12 1000 mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar

Şekil 4.13 ve 4.14'te 1000mm'lik ileri gidiş ve başlangıç noktasına dönüş noktalarındaki kaçıklık mesafeleri ve odometriden okunan kaçıklık değerleri verilmiştir.



Şekil 4.13 Birinci 1000mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar



Şekil 4.14 İkinci 1000mm ileri gidiş ve dönüşlerden elde edilen kaçıklıklar

İkişer defa 10 kez 1000mm ileri gidiş ve başlangıç noktasına dönüş deneylerinden elde ettiğimiz sonuçları birleştirerek karşılaştırdığımızda da sistematik bir hata olmadığı görülmüştür. Tablo 4.5 ve 4.6’da odometriden alınan x, y ve phi verileri ve başlangıç noktası y eksen, ileri gidiş x ve y eksen kaçıklıkları karşılaştırılmıştır.

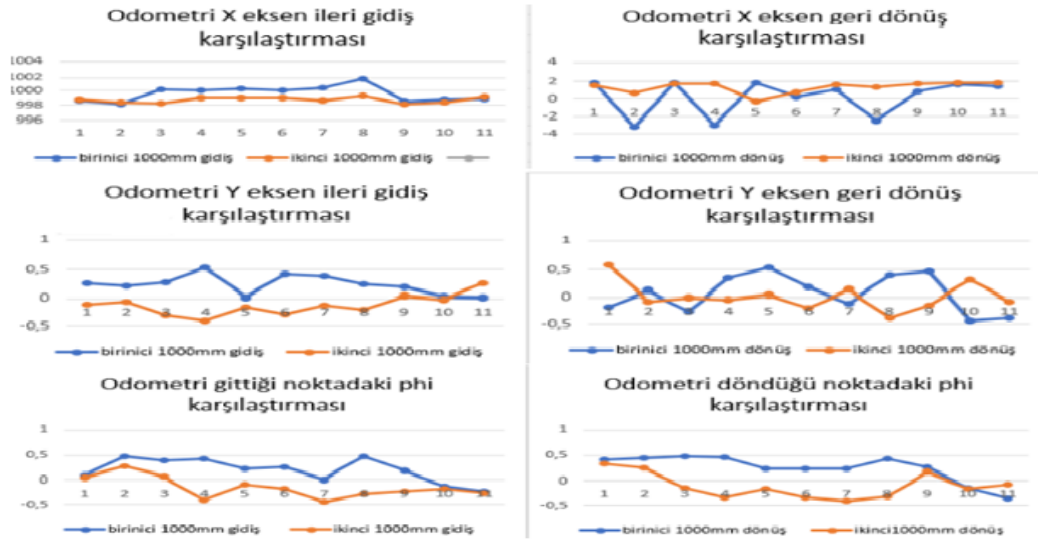
Tablo 4.5 Birinci 1000mm denemesi sonucu odometriden alınan x, y ve phi verilerinin karşılaştırması

KAÇIKLIK												
deneme	Başlangıç noktasından 1000 mm ileri gittiğinde x ve y eksen kaçıklıkları (mm)						Robotino View 3 programı- odometri bloğu konum bilgisi					
	Robotino başlangıç kaçıklığı	sol/ sağ	y eksen kaçıklığı	sol/ sağ	x eksen kaçıklığı	ileri/ geri	x (mm)		y (mm)		phi (derece)	
							gidiş	dönüş	gidiş	dönüş	gidiş	dönüş
1	2	sol	3	sol	1	ileri	998.628	1.841	0.251	-0.172	0.075	0.406
2	15	sol	9	sol	1	ileri	998.097	-3.165	0.2	0.122	0.472	0.441
3	3	sağ	12	sol	6	ileri	1000.19	1.817	0.27	-0.273	0.383	0.474
4	28	sağ	2	sol	6	ileri	1000.07	-2.931	0.545	0.35	0.426	0.471
5	34	sağ	4	sol	6	ileri	1000.33	1.853	0.0137	0.541	0.213	0.211
6	22	sağ	7	sol	6	ileri	1000.09	0.251	0.422	0.165	0.246	0.219
7	16	sağ	8	sol	5	ileri	1000.39	0.985	0.38	-0.114	0.008	0.22
8	32	sağ	2	sol	1	ileri	1001.86	-2.404	0.234	0.403	0.478	0.428
9	12	sol	2	sol	1	ileri	998.59	0.851	0.188	0.468	0.164	0.248
10	22	sol	8	sağ	1	ileri	998.82	1.599	0.024	-0.441	-0.119	-0.132
11	10	sol	5	sol	1	ileri	998.86	1.38	0.017	-0.381	-0.219	-0.359
Ortalama	17.81		5.63		3.18		999.63	1.734	0.2313	0.311	0.254	0.328
Toplam Kayma	196		62		35							

Tablo 4.6 İkinci 1000mm denemesi sonucu odometriden alınan x, y ve phi verilerinin karşılaştırması

KAÇIKLIK												
Başlangıç noktasından 1000 mm ileri gittiğinde x ve y eksen kaçıklıkları (mm)							Robotino View 3 programı- odometri bloğu konum bilgisi					
deneme	Robotino başlangıç kaçıklığı	sol/ sağ	y eksen kaçıklığı	sol/ sağ	x eksen kaçıklığı	ileri/ geri	x (mm)		y (mm)		phi (derece)	
							gidiş	dönüş	gidiş	dönüş	gidiş	dönüş
1	13	sol	17	sol	10	geri	998.743	1.483	-0.1	0.596	0.0475	0.313
2	15	sol	46	sol	25	geri	998.397	0.623	-0.055	-0.073	0.251	0.225
3	20	sol	65	sol	30	geri	998.308	1.753	-0.311	-0.004	0.0687	-0.131
4	17	sol	15	sol	5	geri	998.988	1.74	-0.408	-0.045	-0.41	-0.332
5	36	sol	2	sol	8	ileri	998.985	-0.255	-0.16	0.0447	-0.09	-0.158
6	36	sol	15	sol	3	ileri	999.086	0.744	-0.284	-0.197	-0.17	-0.328
7	22	sol	6	sağ	1	geri	998.617	1.67	-0.13	0.137	-0.439	-0.398
8	14	sol	6	sol	1	ileri	999.245	1.377	-0.2	-0.397	-0.281	-0.318
9	2	sağ	2	sol	2	ileri	998.18	1.702	0.0411	-0.149	-0.229	0.164
10	10	sağ	1	sol	7	geri	998.435	1.801	-0.0192	0.304	-0.169	-0.163
11	20	sağ	8	sol	4	geri	999.176	1.85	0.252	-0.068	-0.253	-0.068
Ortalama	18.63		16.63		8.72		998.7418	1.363	0.178	0.183	0.218	0.236
Toplam Kayma	205		175		96							

Odometriden alınan x, y ve phi verilerinin karşılaştırması ile başlangıç noktası y eksen, ileri gidiş x ve y eksen kaçıklıklarının karşılaştırması Şekil 4.15'te grafiksel olarak verilmiştir.



Şekil 4.15 Odometriden alınan x, y ve phi verilerinin karşılaştırması

Şekil 4.16'da ise 1000mm'lik iki deneme sonucunda Robotino mobil robotun başlangıç noktasına geri döndüğündeki y eksen, ileri gittiğinde x ve y eksen kaçıklıklarının ölçümleri karşılaştırılmıştır.



Şekil 4.16 1000mm iki deneme sonucunda Robotino eksen kaçıklık karşılaştırmaları

Bu denemelerde sadece x eksenini için hareket kontrolü verildi. Bunun üzerine y eksen ve phi (derece) dönüş koordinat bilgilerini de girdiğimizde bu kaçıklıkların daha da artarak devam edeceği aşıkardır. Robotino mobil robotun durağan ve dinamik ortamlarda engellerden de kaçıp istenilen koordinatlara gideceği de göz önüne alındığında, istenilen koordinatlara varışlarda sorunlar oluşacaktır. Robotino simülasyon programında çalışan algoritmamızı laboratuvar ortamında çalıştırdığımızda yukarıda değinilen problemler gözlemlendi. Bu problemi ortadan kaldırmak için aşağıdaki ek algoritmalar eklendi.

Robotino'daki kaçıklık hatalarını gidermek için önce aşağıdaki adımlar izlendi.

- **Sensör kalibrasyonu:** Robotino'nun üzerinde bulunan sensörler doğru şekilde kalibre edildi.
- **Motor kontrolü:** Robotino'nun tekerlek motorları kontrol edilip, motorların hızı ve yönlendirmesi düzgün çalıştırıldı.
- **Daha iyi algoritmalar:** Robotino'nun kaçıklık hatalarını azaltmak için daha iyi algoritmalar ve kontrol stratejileri geliştirildi. Kamera ile renk görüntüsü işlendi, çizgi izleme denemesi yapıldı. Bu iki yöntem istenilen düzeyde başarılı olmadığından vazgeçildi. İşaret algılayıcı ile Lua yazılımı ile algoritma desteklendi.

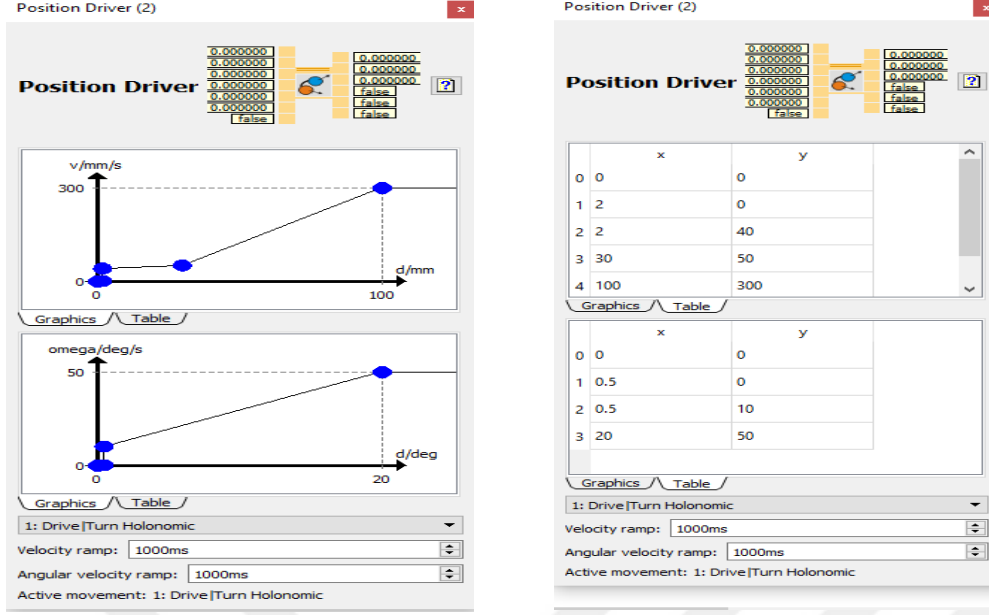
- **Fiziksel ayarlar:** Robotino'nun fiziksel yapılandırması ve ayarları gözden geçirildi. Tekerleklerin ve diğer hareketli parçaların doğru şekilde monte edildiğinden ve ayarlandığından emin olundu.
- **Test ve geri bildirim:** Robotino'nun performansı test edildi ve geri bildirimleri analiz edildi. Robotun nasıl davrandığı ve hangi durumlarda kaçık hatalarının meydana geldiği gözlemlendi. Bu bilgileri kullanarak sistemdeki zayıf noktalar belirlenerek gerekli düzeltmeler yapıldı.

4.4 Odometri Kaçıklık Hatası Sonucu Lokalizasyon Hatalarının İyileştirilmesi

4.4.1 Konum sürücüsü (position drive) grafik ayarlamasının yapılması

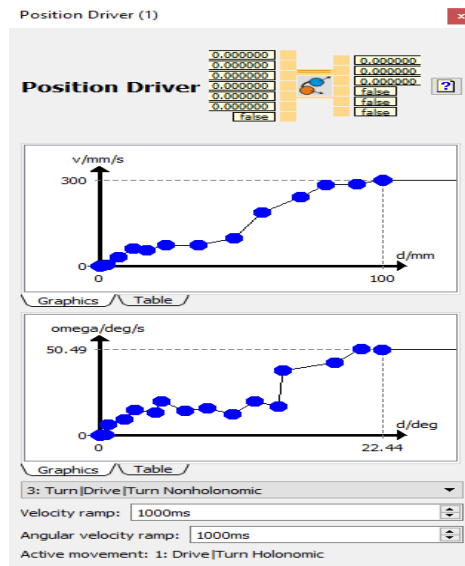
Robotino konum sürücüsü ile robot konumu hedeflenen konumla karşılaştırılarak bir hedef konum ile mevcut konum arasındaki farkı belirlenir, bu farka göre robotun hareket etmesi gereken yol ve yön hesaplanır. Bu hesaplanan değerler ile robotun tahrik sistemlerine uygun kontrol sinyalleri gönderilerek robotun hedeflenen konuma doğru hareket etmesi sağlanır. Şekil 4.17'de Robotino konum sürücüsü grafiği görüntüsü yer almaktadır.

Konum sürücüsü grafiği ile PID kontrolü yapılabilir. PID kontrolü, bir sistemde istenen bir hedef değere yaklaşmak veya bu değeri sabit tutmak için kullanılan yaygın bir geri besleme kontrol yöntemidir. Konum sürücüsü de motorları belirli bir konumda tutmak veya belirli bir konuma hareket ettirmek amacıyla kullanılan bir mekanizmadır.



Şekil 4.17 Robotino View konum sürücüsü standart grafiği

Konum sürücüsü grafiği, motorların veya robotun belirli bir zaman aralığında hangi konumda olduğunu gösteren bir grafikdir. PID kontrolünü konum sürücüsü ile birleştirerek, motorların istenen konumlarda olmasını sağlamak veya belirli bir hedef konuma hassas bir şekilde hareket etmelerini sağlamak mümkündür. Şekil 4.18 Robotino mobil robotun konum sürücüsünün bu çalışmadaki grafik yapılandırma değerlerini vermektedir.



Şekil 4.18 Konum sürücüsü grafiği yapılandırılmış ayar değerleri

Tablo 4.7’de Robotino mobil robotun ileri hızlanmasının kontrolü için konum sürücüsü içerisindeki yapılandırma değerleri, Tablo 4.8’de ise Robotino mobil robotun phi hızlanmasının kontrolü için konum sürücüsü içerisindeki yapılandırma değerleri verilmiştir.

Tablo 4.7 Konum sürücüsü hızlanma yapılandırma değerleri

v (mm/s)		
	x	y
0	0	0
1	1	0
2	26.178	3
3	680.628	30
4	120.419	60
5	167.539	54
6	235.602	72
7	350.785	72
8	47.644	96
9	575.916	186
10	712.042	240
11	801.047	282
12	910.995	285
13	100	300

Tablo 4.8 Konum grafiği phi yapılandırma değerleri

omega (deg/s)		
	x	y
0	0	0
1	0.125	0
2	0.25	0
3	0.5	0
4	0.704843	60.588
5	199.705	90.882
6	281.937	146.421
7	446.401	131.274
8	49.339	196.911
9	681.348	141.372
10	857.559	156.519
11	105.726	121.176
12	123.348	196.911
13	142.143	166.617
14	145.668	378.675
15	186.783	424.116
16	207.929	50.49
17	224.375	50

Konum sürücüsünün grafik ayarlaması Robotino'nun hareketini ve konumunu belirlemek ve özelleştirmek için önemli bir araçtır. Yapılan bu grafik ayarları odometrideki kaçıklıkları azaltarak Robotino'nun daha hassas bir şekilde ve daha verimli çalışmasını sağlamıştır.

4.4.2 Holonomik ve non-holonomik sürüş

Konum sürücüsü grafiğinde holonomik ve non-holonomik sürüş seçeneği bulunmaktadır. Holonomik ve non-holonomik sürüş, bir mobil robotun hareket kabiliyetini tanımlayan terimlerdir.

Holonomik sürüş, bir mobil robotun konum sürücüsü içerisinde, robotun tüm yönlerde bağımsız ve serbestçe hareket edebilme yeteneğini ifade eder. Yani robot ileri, geri, sağa ve sola hareket edebilirken aynı zamanda dönme hareketi de yapabilir. Holonomik sürüş, robotların dar alanlarda veya karmaşık çevrelerde daha fazla manevra kabiliyetine sahip olmalarını sağlar. Düzgün bir zeminde hareket eden bir tekerlekli sandalye, holonomik sürüşe sahip bir örnektir. Holonomik sürüş, genellikle diferansiyel sürüş veya omni-wheel (mecanum veya omni) sistemler gibi platformlarda görülür. Bu tür platformlar, robotun her yöne ayrı bağımsız hareket edebilmesine olanak tanır. Örneğin, omni-wheel bir robotun her bir tekerleği ayrı yönlere dönebilir, böylece robot ileri, geri, sağa ve sola hareket edebilirken aynı zamanda dönme hareketi de gerçekleştirebilir.

Robotino mobil robotunun konum sürücüsü içerisindeki diğer sürüş seçeneği olan non-holonomik sürüş, robotun belli kısıtlamalar nedeniyle tüm yönlerde bağımsız ve serbestçe hareket edememe yeteneğini ifade eder. Bu tür sürüş durumları, robotun belirli hareket yollarını takip etmek veya belirli dönüş yarıçaplarıyla hareket etmek zorunda olduğu durumları ifade eder. Non-holonomik sistemlerde genellikle belirli bir yöne hareket ederken dönme kabiliyetine sahip olmayabilirler veya dönme kabiliyeti sınırlıdır. Arabalar, bu tür bir sürüşe örnek olarak verilebilir. Bir araba doğrudan yanlara veya geriye doğru hareket etme yeteneğine sahip olmazken, dönüşleri sınırlıdır ve bu dönüşleri yaparken belirli bir yarıçap gereklidir.

Non-holonomik sistemler genellikle daha fazla planlama ve kontrol gerektirir, çünkü istenen konuma ulaşmak için daha dikkatli bir yol izlemeleri gerekebilir. Non-holonomik sürüş, cihazın konumunu ve yönelimini belirli bir zamanda ve belirli bir hızda değiştirmesine izin veren kinematik sınırlamalara sahipken, aynı anda konum, hız ve yönelim gibi tüm bağımsız dereceleri hareket etmesine izin vermez.

4.4.3 Velocity ramp (hız rampası) ve angular velocity ramp (açısal hız rampası)

Konum sürücüsü içerisindeki başka bir parametre ise “velocity ramp” (hız rampası) ve “angular velocity ramp” (açısal hız rampası) ayarlamalarıdır. Robotino mobil robotun konum sürücüsü içerisindeki "velocity ramp" terimi, robotun hızını ve hız değişimini kontrol etmek için kullanılan bir kavramı ifade eder. Velocity ramp robotun belirli bir süre içerisinde hızını artırması veya azaltması anlamına gelir. Bu, robotun ani hız değişikliklerinden kaçınarak sarsıntısız ve daha kontrollü bir şekilde hareket etmesini sağlar.

Velocity ramp, genellikle hızlanma ve yavaşlama sürelerini belirtir. Yani, robotun belirli bir hız düzeyine ulaşması veya belirli bir hızdan durmasının ne kadar sürdüğünü kontrol eder. Bu, robotun daha kararlı ve güvenli bir şekilde hareket etmesini sağlarken mekanizmalara veya çevreye zarar verme riskini azaltır.

Robotun aniden yüksek bir hıza çıkması, tekerleklerin kaymasına veya robotun denge kaybına uğramasına neden olabilir. Velocity ramp, bu tür riskleri azaltmak ve daha kontrollü bir şekilde hareket etmek amacıyla kullanılır. Aynı şekilde, dururken veya yavaşlarken aniden durması da robotun denge kaybetmesine veya mekanizmanın zorlanmasına neden olabilir. Velocity ramp, bu durumları da önlemeye yardımcı olur.

Angular velocity ramp, robotun belirli bir süre içerisinde açısal hızını artırması veya azaltması anlamına gelir. Bu, robotun ani açısal hız değişikliklerinden kaçınarak daha pürüzsüz ve kontrollü bir şekilde dönme

hareketi gerçekleştirmesini sağlar. Angular velocity ramp, genellikle dönme hızındaki hızlanma ve yavaşlama sürelerini belirtir. Robotun belirli bir dönme hızına ulaşması veya belirli bir dönme hızından durmasının ne kadar sürdüğünü kontrol eder. Bu, robotun daha kararlı ve güvenli bir şekilde dönme hareketi gerçekleştirmesini sağlar. Robotun aniden yüksek bir dönme hızına çıkması, denge kaybına veya mekanizmanın zorlanmasına neden olabilir. Aynı şekilde, dururken veya dönüşü yavaşlarken aniden durması da robotun denge kaybetmesine veya mekanizmanın zarar görmesine neden olabilir. Bu parametre, bu tür riskleri azaltmak ve daha kontrollü bir şekilde dönme hareketi gerçekleştirmek amacıyla kullanılır.

Bu tez çalışmasında, konum sürücüsündeki PID kontrol için grafik düzenlemesi için gerekli parametre değerleri elde edildi. Holonomik ve non-holonomik sürüş denemeleri velocity ramp ve angular velocity ramp süreleri değiştirilerek gerçekleştirilmiş ve denemeden elde edilen sonuçlar incelenip en uygun sürüş seçeneği için parametreler seçilmiştir. Deneme sonuçlarının bazıları Tablo 4.9 ve Tablo 4.10’da görülmektedir.

Tablo 4.9 Konum sürücüsü grafik yapılandırması yapıldıktan sonraki deneme sonuçları 1

		Konum sürücü grafik ayarı yapıldı					
		Drive holonomik, Velocity ramp=1000ms, Angular velocity ramp=1000ms					
		x_2500mm, y=0mm , Phi=180 derece (2500,0,180)					
		x		y		Açı (phi)	
		0mm noktası	2500mm noktası	0mm noktası	2500mm noktası	0mm noktası	2500mm noktası
1.deneme	Kalkış	0	2499.465	0	-0.181	0	179.886
	Dönüş	-0.145	2500.997	-0.545	0.784	0.153	179.606
	Fark	0.145	-1.532	0.545	-0.965	-0.153	0.28
2.deneme	Kalkış	-0.145	2500.997	-0.545	0.784	0.153	179.606
	Dönüş	0.742	2501.275	-0.281	-0.482	0.212	179.856
	Fark	-0.887	-0.278	-0.264	1.266	-0.059	-0.25
3.deneme	Kalkış	0.742	2501.275	-0.281	-0.482	0.212	179.856
	Dönüş	-0.063	2500.482	1.041	-1.139	0.3	179.873
	Fark	0.805	0.793	-1.322	0.657	-0.088	-0.017
4.deneme	Kalkış	-0.063	2500.482	1.041	-1.139	0.3	179.873
	Dönüş	-1.242	2502.083	-0.236	2.063	0.247	179.794
	Fark	1.179	-1.601	1.277	-3.202	0.053	0.079
5.deneme	Kalkış	-1.242	2502.083	-0.236	2.063	0.247	179.794
	Dönüş	-0.867	2500.875	0.429	-0.433	-0.435	179.797
	Fark	-0.375	1.208	-0.665	2.496	0.682	-0.003
6.deneme	Kalkış	-0.867	2500.875	0.429	-0.433	-0.435	179.797
	Dönüş	-0.764	2500.066	-0.383	-1.133	-0.044	179.736
	Fark	-0.103	0.809	0.812	0.7	-0.391	0.061
7.deneme	Kalkış	-0.764	2500.066	-0.383	-1.133	-0.044	179.736
	Dönüş	-0.345	2499.475	-0.226	-0.352	0.115	179.808
	Fark	-0.419	0.591	-0.157	-0.781	-0.159	-0.072
8.deneme	Kalkış	-0.345	2499.475	-0.226	-0.352	0.115	179.808
	Dönüş	0.994	2499.965	-0.733	-0.142	0.994	179.521
	Fark	-1.339	-0.49	0.507	-0.21	-0.879	0.287

Başlangıç Noktası: x=0 mm , y=0mm , Phi = 0 derece

Hedef Nokta: x=2500 mm , y=0mm , Phi = 180 derece

Tablo 4.10 Konum sürücüsü grafik yapılandırması yapıldıktan sonraki deneme sonuçları 2

		Konum Sürücü grafik ayarı yapılmadı					
		Drive holonomik, Velocity ramp=1000ms, Angular velocity ramp=1000ms					
		x_2500mm , y=0mm , Phi=180 derece (2500,0,180)					
		x		y		Açı (phi)	
		0mm noktası	2500mm noktası	0mm noktası	2500mm noktası	0mm noktası	2500mm noktası
1.deneme	Kalkış	0	2501.608	0	1.717	0	179.973
	Dönüş	-3.316	2504.033	0.797	-0.062	-0.25	179.774
	Fark	3.316	-2.425	-0.797	1.779	0.25	0.199
2.deneme	Kalkış	-3.316	2504.033	0.797	-0.062	-0.25	179.774
	Dönüş	-4.834	2501.658	-0.305	1.222	0.131	179.872
	Fark	1.518	2.375	1.102	-1.284	-0.381	-0.098
3.deneme	Kalkış	-4.834	2501.658	-0.305	1.222	0.131	179.872
	Dönüş	-0.724	2504.202	0.583	0.091	-0.175	179.672
	Fark	-4.11	-2.544	-0.888	1.131	0.306	0.2
4.deneme	Kalkış	-0.724	2504.202	0.583	0.091	-0.175	179.672
	Dönüş	-4.212	2502.776	0.389	-0.049	-0.254	179.814
	Fark	3.488	1.426	0.194	0.14	0.079	-0.142
5.deneme	Kalkış	-4.212	2502.776	0.389	-0.049	-0.254	179.814
	Dönüş	-2.294	2501.194	-0.013	0.214	0.252	179.942
	Fark	-1.918	1.582	0.402	-0.263	-0.506	-0.128
6.deneme	Kalkış	-2.294	2501.194	-0.013	0.214	0.252	179.942
	Dönüş	-2.641	2502.425	0.888	1.389	0.306	179.962
	Fark	0.347	-1.231	-0.901	-1.175	-0.054	-0.02
7.deneme	Kalkış	-2.641	2502.425	0.888	1.389	0.306	179.962
	Dönüş	-2.496	2503.697	-0.635	0.522	0.324	179.897
	Fark	-0.145	-1.272	1.523	0.867	-0.018	0.065
8.deneme	Kalkış	-2.496	2503.697	-0.635	0.522	0.324	179.897
	Dönüş	-3.82	2502.246	-0.715	-0.27	0.193	179.801
	Fark	1.324	1.451	0.08	0.792	0.131	0.096

Tablo 4.11’de, non-holonomik sürüş için velocity ramp ve angular velocity ramp hızlanma süreleri 1000ms olarak ayarlandı. Bu hızlanma sürelerinde hedef konum x=2500 mm, y=0 mm ve phi=180 derece olarak belirlendi. Konum sürücüsündeki mevcut standart grafik ve yapılandırması yapılan grafik için ayrı ayrı yukarıdaki ayarlamalar ile birlikte Robotino robotun x, y ve phi hedefine

gidip gelmesini, odometriden alınan değerleri gözlemleyerek sekiz defa bu sürüş tekrarlanmıştır.

Holonomik sürüş için velocity ramp ve angular velocity ramp süreleri değiştirilerek ve konum sürücüsünde PID kontrollü ve kontrolsüz olmak üzere sekiz deneme yapılmıştır. Tablo 4.11’de başlangıç noktası – hedef nokta – başlangıç noktası olarak toplamda 5000 mm yol için elde edilen değerler bulunmaktadır. Bu deney sonuçları değerlendirilerek non-holonomik sürüş velocity ramp için 1000ms ve angular velocity ramp için 1000ms süreleri kullanılmıştır.

Tablo 4.11 Holonomik sürüş için velocity ramp ve angular ramp süreleri değiştirilmiş deney verileri

Konum sürücü grafik ayarı yapılmadan önce			Konum sürücü grafik ayarı yapıldıktan sonra		
Drive holonomik - Velocity Ramp=1000ms, Angular Velocity Ramp=1000ms			Drive holonomik - Velocity Ramp=1000ms, Angular Velocity Ramp=1000ms		
x=2500mm, y=0mm, Phi=180 derece (2500,0,180)			x=2500mm, y=0mm, Phi=180 derece (2500,0,180)		
x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı	x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı
-24.868	-1.595	0.587	4.45	-1.975	-0.029
x fark toplamı	y fark toplamı	Phi fark toplamı	x fark toplamı	y fark toplamı	Phi fark toplamı
-3.759	-0.344	0.026	0.965	0.269	0.371

Konum sürücü grafik ayarı yapılmadan önce			Konum sürücü grafik ayarı yapıldıktan sonra		
Drive non-holonomik - Velocity Ramp=1000ms, Angular Velocity Ramp=1000ms			Drive non-holonomik - Velocity Ramp=1000ms, Angular Velocity Ramp=1000ms		
X=2500mm, Y=0mm, Phi=180 derece (2500,0,180)			X=2500mm, Y=0mm, Phi=180 derece (2500,0,180)		
X-hedeften dönüşler toplamı	Y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı	X-hedeften dönüşler toplamı	Y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı
-47.238	1.949	1.403	-1.69	1.542	1.542
X fark toplamı	Y fark toplamı	Phi fark toplamı	X fark toplamı	Y fark toplamı	Phi fark toplamı
-4.982	1.675	0.029	0.994	-0.733	0.994

Konum sürücü grafik ayarı yapılmadan önce			Konum sürücü grafik ayarı yapıldıktan sonra		
Drive holonomik - Velocity Ramp=100ms, Angular Velocity Ramp=20ms			Drive holonomik - Velocity Ramp=100ms, Angular Velocity Ramp=20ms		
x=2500mm, y=0mm, Phi=180 derece (2500,0,180)			x=2500mm, y=0mm, Phi=180 derece (2500,0,180)		
x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı	x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı

-25.385	0.034	-2.998	6.047	2.092	-2.531
x fark toplamı	y fark toplamı	Phi fark toplamı	x fark toplamı	y fark toplamı	Phi fark toplamı
-2.5	0.289	-0.258	0.338	-0.155	0.145

Konum sürücü grafik ayarı yapılmadan önce			Konum sürücü grafik ayarı yapıldıktan sonra		
Drive non-holonmik - Velocity Ramp=100ms, Angular Velocity Ramp=20ms			Drive non-holonmik - Velocity Ramp=100ms, Angular Velocity Ramp=20ms		
x=2500mm, y=0mm, Phi=180 derece (2500,0,180)			x=2500mm, y=0mm, Phi=180 derece (2500,0,180)		
x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı	x-hedeften dönüşler toplamı	y-hedeften dönüşler toplamı	Phi- hedeften dönüşler toplamı
-24.327	0.989	0.527	-7.095	3.091	-0.751
x fark toplamı	y fark toplamı	Phi fark toplamı	x fark toplamı	y fark toplamı	Phi fark toplamı
-4.982	1.675	0.029	-0.657	-0.426	0.078

4.5 Dijital Görüntü İşleme

Dijital görüntü işleme, görüntü verilerinin bilgisayar tabanlı yöntemler kullanarak analiz edilmesi, değiştirilmesi ve yorumlanmasını sağlayan bir bilgi işlem disiplini. Bu süreç, bir veya daha fazla dijital görüntü üzerinde çeşitli matematiksel, istatistiksel ve algoritmik işlemleri kullanarak bilgi çıkarma, düzeltilme veya geliştirme amacını taşır. Görüntü elde etmek için kullanılan yazılım ve donanım görüntü kaynağına bağlı olarak farklılık gösterebilir. Bu çalışmada Robotino üzerine taktığımız harici bir kamera kullanılmıştır. Dijital görüntü elde etmek için niteleme (Qualification) ve niceleme (Quantification) işlemlerini güderek sürekli veriyi dijitize etmek gerekmektedir.

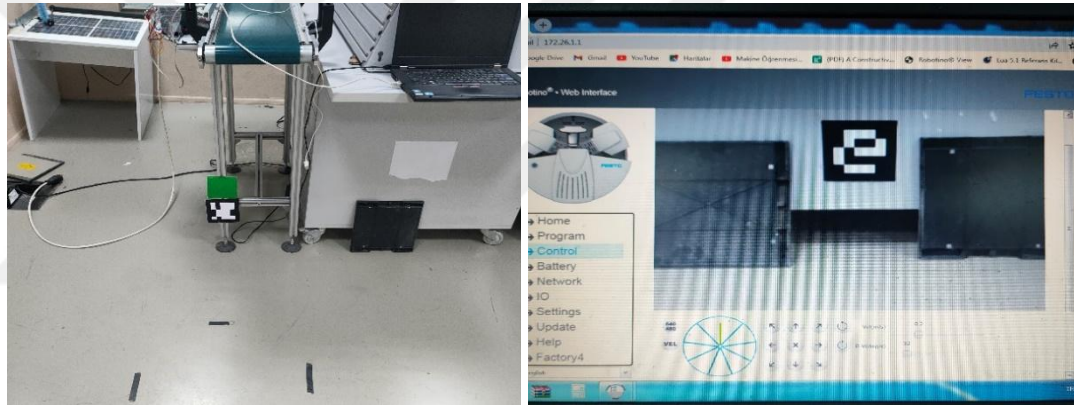
Niteleme, görüntüdeki nesnelerin veya bölgelerin niteliklerinin belirlenmesi anlamına gelir. Bu, görüntülerdeki özelliklerin tanımlanmasını içerir. Bir görüntüdeki nesnelerin renkleri, şekilleri, büyüklükleri gibi özellikler niteleme kapsamına girer. Niteleme işlemi, görüntü verilerini daha anlaşılır ve yorumlanabilir hale getirmek için önemlidir. Niceleme, görüntüdeki nesnelerin veya özelliklerin sayısal olarak ölçülmesini içerir.

Bu çalışmada Robotino robotun hedeflenen konumlandırması yapılabilmesi için Robotino View program arayüzündeki kamera 1'den alınan kare işaretleyicisi görüntüsünü işaretleyici dedektörü bloğu kullanarak iki boyutlu görüntünün

işlenmesi ile konumlandırma kaçıklık hatalarını minimuma indirilmesi hedeflenmiştir. Bunun için de niteleme kapsamında renkli görüntü ve niceleme kapsamında ise işaret işleme, görüntü işleme kapsamında kullanılmıştır.

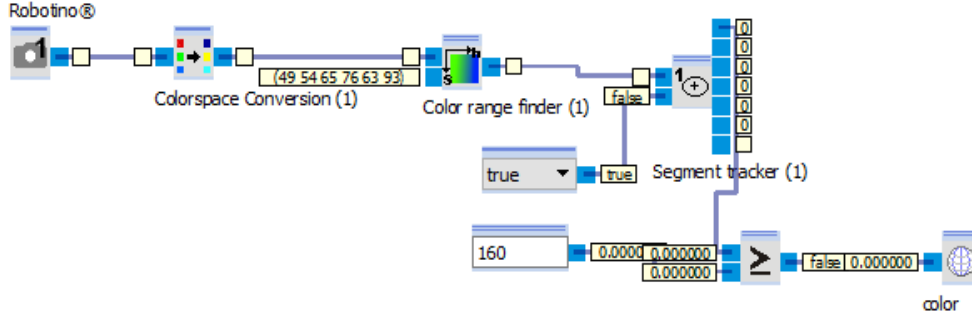
4.5.1 Görüntü işleme ile renk algılama

Robotino View'da kamera 1 ile alınan görüntü renk uzayı dönüştürme bloğundan RGB renk uzayı seçimi yapılarak renk aralığı bulucu bloğu ile istenilen renk blok içinde tanımlanmış, segment izleyici bloğu ile belirlenen rengi bulması sağlanmıştır. Bu aşamada Robotino görevine başlamadan önce kamera kalibrasyonu yapılmalıdır. Şekil 4.19(a)'da Robotino kamera görüntüsü ve Şekil 4.19(b)'de Robotino kamera kalibrasyonu görüntüsü yer almaktadır.



Şekil 4.19 (a) Robotino kamera görüntüsü, (b) Robotino kamera kalibrasyonu görüntüsü

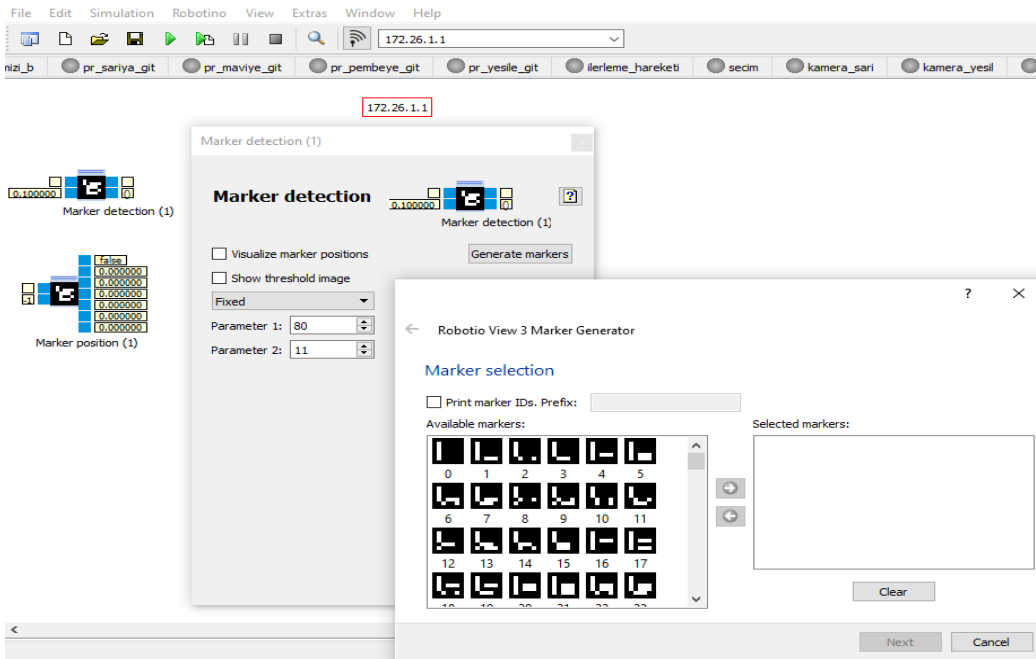
Yürüyen bant ve depo önüne yerleştirdiğimiz kırmızı, mavi, yeşil ve sarı renklerin odometri kaçıklıklarından kaynaklanan eksen kaymalarının hatasını azaltmak için kullanılması düşünüldü. Şekil 4.20'de görülen Robotino View programındaki bloklarla yazılan kamera ile renk işleme işlemi yöntemi kısmen başarılı olsa da ilerleyen sürüşler sonrası verimli olmadığı görülmüştür. Eksen kaçıklıklarından kaynaklanan hatalar Robotino kamerasının rengi sağdan veya soldan görerek yaklaşmasını sağlamıştır ve ilerleyen iş akışlarında renkleri görmemeye başlamıştır. Bu yüzden kamera ile renkli görüntü işleme yönteminden vazgeçilmiştir.



Şekil 4.20 Robotino View programında kamera ile renk işleme

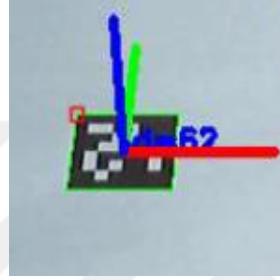
4.5.2 Kare işaret işleme

Robotino View programındaki işaretçi tespiti (marker detection) bloğu ile üretilen kare işaretçilerin kullanılması amaçlanmıştır. Bu kare referans işaretleyiciler, yaygın olarak kullanılan ArUco kütüphanesine dayanmaktadır. ArUco, görüntülerdeki karesel işaretçileri belirlemek için kullanılan açık kaynak bir kütüphanedir. Artırılmış gerçeklik ve robotikte kullanılan 2 boyutlu barkodları okur ve kodları çözer. Şekil 4.21’de görüldüğü üzere her işaret, siyah ve beyaz piksellerin küçük bir bölmesine kodlanmış sayıya karşılık gelir. Kamera kalibrasyonu yapıldıktan sonra ArUco kod çözme algoritması ile kameranın görüş alanındaki herhangi bir ArUco işaretçisinin konumu (yer ve yönü) belirlenebilir.



Şekil 4.21 İşaretçi konum belirleme bloğu tarafından üretilen işaretleyici örnekleri

Üretilen her işaret için tanımlayıcısı (ID) oluşturulur. Siyah kenarlar, görüntüde hızlı algılama sağlar ve ikili kodlama, tanımlama, hata belirleme ve düzeltme tekniklerinin uygulanmasına izin verir. ID numarası bilinen kare işaretleyici basılı çıktısını konumlandırma yapacağımız koordinata kameranın görüş açısına bağlı olarak yerleştiririz. Üretilen koordinat sistemi, Şekil 4.22’de gösterildiği gibi z eksenini işaretleyicinin merkezine yerleştirilir. Robotino kamerası ile görüntülenen işaretçi, kırmızı çizgiyi x eksen, yeşil çizgiyi y eksen ve mavi çizgiyi z eksen olarak kabul etmektedir.



Şekil 4.22 Üretilen kare işaretleyicinin kamera tarafından görüntülenen vektörel gösterimi

4.5.3 İşaretçi algılayıcı ile konum tahmini

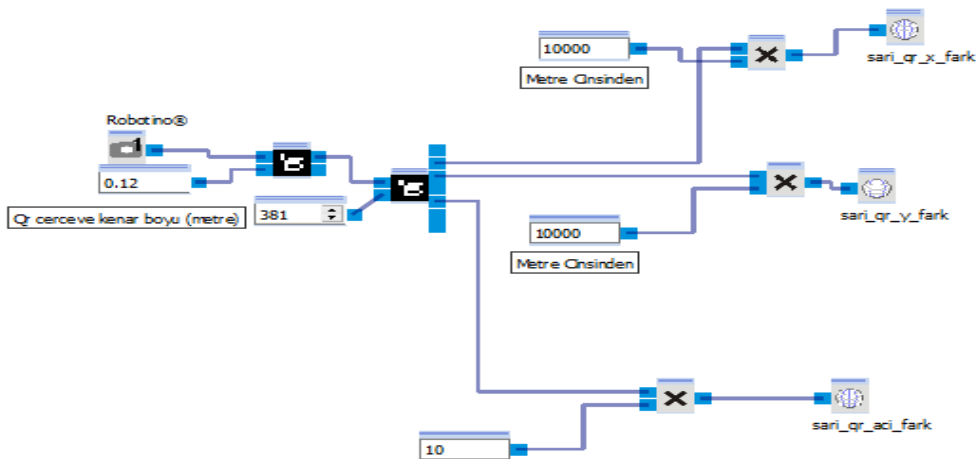
Yapılan deneysel çalışma Şekil 4.23’te görülmektedir. Robotino robotlar kameraları ile başlangıç noktası konumunu, ID’si belirlenmiş bir kare işaretleyici ile okur. Kamera tarafından okunan işaretleyicinin x, y, z ve phi bilgisi kaydedilir. Hedef konuma gidip geldikten sonra odometri kaçıklıklarından dolayı Robotino başlangıç noktasına gelmediği için, geldiği noktada tekrar kare işaretleyici kamera tarafından okutturulur. Bu okunan x, y, z ve phi bilgileri x_fark, y_fark, z_fark ve aci_fark olarak tekrar kaydedilir. İlk okunan değerler ile sonraki değerler karşılaştırılır ve hangi vektörde eksi ya da artı yönde kaçıklık var ise ilk konum referans alınarak Robotino mobil robot referans başlangıç konumuna getirilir. Bunun için ayrı ayrı her vektör için kontrol algoritması oluşturuldu. Phi kontrol, x kontrol ve y kontrol algoritmaları ile ilk okunan işaretleyiciden alınan x, y, z ve phi başlangıç konumlarına getirilmeye çalışıldı. Phi, x ve y kontrol, Bölüm 4.10’daki algoritmada ayrıntılı olarak açıklanmıştır.



Şekil 4.23 İşaret algılayıcı ile hedef koordinata yönlendirme algoritması deneyi

Odometri hatasından kaynaklı konumlandırma düzeltme çalışmasında Robotino mobil robot birden fazla konuma gidip tekrar başlangıç noktasına geldiğinde kamera açısında sapmalar meydana gelmektedir. Bu ise kare işaretleyicisini okurken kameranın açısının da sapmasıyla tekrardan kaçıklıklar meydana getirmiştir. Bunun için yeni bir kontrol algoritması geliştirilmesi amaçlanmış ve Lua yazılım dili yazılan bir program ile çalışan sisteme destek amaçlı ekleme yapılmıştır. Lua yazılım kullanıldığında bu konumlandırma hataları azaltılmış ve düzenli bir şekilde çalışma gerçekleştirilmiştir. Bu çalışma Robotino mobil robotun gideceği her koordinat için gerçekleştirilmiştir.

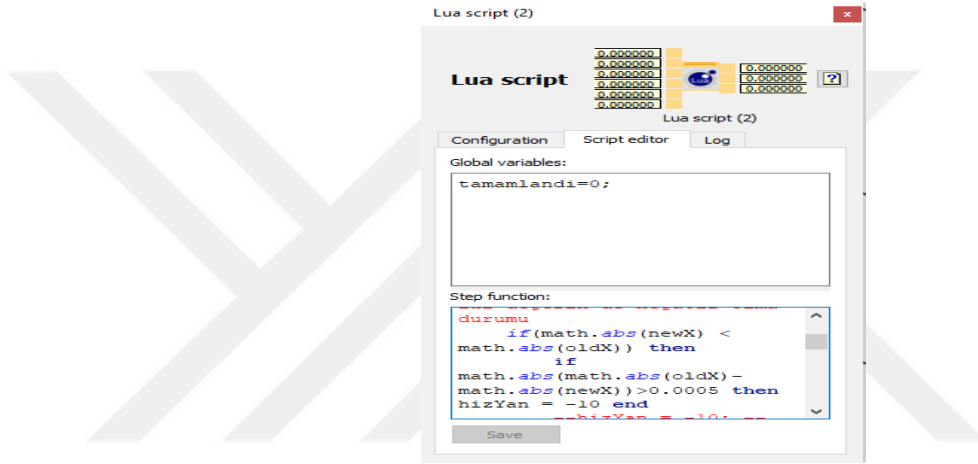
Şekil 4.24'te görülen Robotino View program blokları ile işaret algılayıcı konumu okuma işlemi gerçekleştirilmiştir.



Şekil 4.24 Robotino View programı ile işaret algılayıcı konumunu okuma

4.6 Lua Yazılımı ile Koordinatlandırma

Lua; yüksek seviyeli bir programlama dilidir ve özellikle gömülü yazılım uygulamaları için tasarlanmıştır. 1993 yılında Brezilyalı bir ekibin lideri olan Roberto Ierusalimsky tarafından geliştirilmiştir. Lua, basit ve temiz bir sözdizimine sahiptir ve yüksek derecede esnek ve genişletilebilir bir dil olarak kullanılır. Lua, Robotino'nun yeteneklerini genişletmek ve karmaşık robotik görevleri gerçekleştirmek için güçlü bir araçtır. Şekil 4.25'te Robotino View programındaki Lua bloğu yer almaktadır.



Şekil 4.25 Robotino View programındaki Lua bloğu

Kare işaretleyicileri kullanarak konumlandırmadaki kaçıklıkları x, y ve phi eksenlerinde yapılan düzeltmelere ek olarak Lua yazılımı ile yazılan program vasıtasıyla Robotino mobil robot tekrardan bir koordinat düzeltme işlemi gerçekleştirilmiştir. Yazılan programda, ilk konumlara göre sağa ya da sola, sol yana ya da sağ yana kontrolü yapılmıştır. Şekil 4.26'da işaret algılayıcının durumlarına göre Lua dili ile dengeye getirilmiş Robotino görülmektedir.



Şekil 4.26 Robotino mobil robotun Lua bloğuna yazılan program ile dengeye getirilmesi

Aşağıdaki program Robotino'nun gideceği dört konum için de uygulanmıştır.

Algoritma 4.1 Robotino hız ve yön kontrol programı

```

local oldX = in1;
local oldY = in2;
local oldTheta = in3;
local newX = in4;
local newY = in5;
local newTheta = in6;
local hizYan; --sağa sola giderken kullanılır. Daha sonra
vY'ye bağlanacak.
local hizOne; --ileri geri giderken kullanılır. Daha sonra
vX'e bağlanacak
local hizAci; --Robot dönüşü için phi. Daha sonra Omega'ya
bağlanacak.

out1=nil;
out3=nil;

--önce işaretler belirlenir. 4 kombinasyon için 4 if sorgusu
yazılır.
if(newX==0)then --eğer yeniDeğer okunmadıysa hiçbir şey
yapma
out2=nil;
end

if(oldX<0 and newX<0)then -- iki değerin de negatif olma

```

```

durumu
if(math.abs(newX) < math.abs(oldX)) then
if math.abs(math.abs(oldX)-math.abs(newX))>0.0005 then
hizYan = -10 end
--hizYan = -10; --robot vy ile sağa gidiyor.
out2 = hizYan;
--print('robot saga gidiyor');
end
if(math.abs(newX) > math.abs(oldX)) then
if math.abs(math.abs(oldX)-math.abs(newX))>0.0005 then
hizYan = 10 end
--hizYan = 10; --robot vy ile sola gidiyor.
out2 = hizYan;
--print('robot sola gidiyor');
end
end

if(oldX>0 and newX>0)then -- iki değerin de pozitif olma durumu

if(math.abs(newX) < math.abs(oldX)) then
if math.abs(math.abs(oldX)-math.abs(newX))>0.0005 then
hizYan = 10 end
--hizYan = 10; --robot vy ile sola gidiyor.
out2 = hizYan;
--print('robot sola gidiyor');
end
if(math.abs(newX) > math.abs(oldX)) then
if math.abs(math.abs(oldX)-math.abs(newX))>0.0005 then
hizYan = -10 end
--hizYan = -10; --robot vy ile sağa gidiyor.
out2 = hizYan;
--print('robot saga gidiyor');
end
end

if(oldX>0 and newX<0)then -- eski değer pozitif, yenidoğer negatif. Robot mutlaka sola gitmelidir.
if (oldX - newX)>0.0005 then hizYan = 10 end
--hizYan = 10; --robot vy ile sola gidiyor.
out2 = hizYan;
--print('robot sola gidiyor');
end

if(oldX<0 and newX>0)then -- eski değer negatif, yenidoğer pozitif. Robot mutlaka sağa gitmelidir.
out2 = hizYan;
if (newX - oldX)>0.0005 then hizYan = -10 end
--hizYan = -10; --robot vy ile sağa gidiyor.
out2 = hizYan;
--print('robot sola gidiyor');
end

--Aşağıdaki kısım da ileri ve geri (öne-arkaya) hareket için

```

```

if(newY==0)then --eğer yeniDeğer okunmadıysa hiçbir şey
yapma
out1=nil;
end

if(oldY<0 and newY<0)then -- iki değerin de negatif olma
durumu
if(math.abs(newY) < math.abs(oldY)) then
if math.abs(math.abs(oldY)-math.abs(newY))>0.0005 then
hizOne = 10 end
--hizOne = 10; --robot vy ile ileri gidiyor.
out1 = hizOne;
--print('robot saga gidiyor');
end
if(math.abs(newY) > math.abs(oldY)) then
if math.abs(math.abs(oldY)-math.abs(newY))>0.0005 then
hizOne = -10 end
--hizOne = -10; --robot vy ile geri gidiyor.
out1 = hizOne;
--print('robot geri gidiyor');
end
end

if(oldY>0 and newY>0)then -- iki değerin de pozitif olma
durumu

if(math.abs(newY) < math.abs(oldY)) then
if math.abs(math.abs(oldY)-math.abs(newY))>0.009 then hizOne
= 10 end
--hizOne = 10; --robot vy ile geri gidiyor.
out1 = hizOne;
--print('robot geri gidiyor');
end
if(math.abs(newY) > math.abs(oldY)) then
if math.abs(math.abs(oldY)-math.abs(newY))>0.0009 then
hizOne = -10 end
--hizOne = -10; --robot vy ile ileri gidiyor.
out1 = hizOne;
--print('robot geri gidiyor');
end

end

if(oldY>0 and newY<0)then -- eski değer pozitif, yenideğer
negatif.Robot mutlaka geri gitmelidir.
if (oldY - newY)>0.0005 then hizOne = -10 end
--hizOne = 10; --robot vy ile geri gidiyor.
out1 = hizOne;
--print('robot geri gidiyor');
end

if(oldY<0 and newY>0)then -- eski değer negatif, yenideğer
pozitif. Robot mutlaka ileri gitmelidir.

if (newY - oldY)>0.0005 then hizOne = 10 end

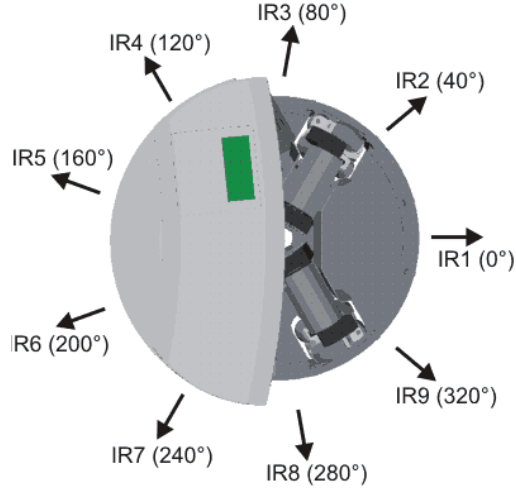
```

```
--hizOne = 10; --robot vy ile ileri gidiyor.
out1 = hizOne;
--print('robot geri gidiyor');
end
```

Bu yazılımda x ve y eksenlerdeki konum bilgilerindeki “hizOne” (ileri/geri) ve “hizYan” (sol/sağ) değerlerindeki işaretler değiştirildi. Bunun nedeni ise Robotino robotun istenilen konumlara geldiğindeki konum durumunun hedeflenen konum durum bilgilerinden farklı olmasından kaynaklanmaktadır. Hedeflenen konumun sağında veya solunda ya da ilerisinde ya da gerisinde olup olmadığının kontrolü yapılarak Robotino mobil robotun hareket yönlerinde hedef konuma getirmek için değişiklikler sağlanmıştır. “hizOne” kapsamında iki kontrol ve “hizYan” kapsamında iki kontrol olmak üzere gidilecek hedef koordinatın durumuna göre işaretler değiştirilmiştir.

4.7 Robotino Mobil Robotun Engelden Kaçınma Kontrolü

Robotino mobil robotun engelden kaçınma kontrolü, robotun çevresini izleyerek engelleri algılaması ve bu engellerden kaçınması için gerekli adımları atmaya dayalı bir dizi işlemi içerir. Şekil 4.27’de Robotino üzerinde gösterilen 40 derece açı ile yerleştirilmiş 9 adet IR sensör yakındaki engelleri algılar ve bu engellere olan mesafeyi ölçer. Sensörler tarafından toplanan veriler, engellerin türünü ve konumunu belirlemek için kullanılan bir kontrol algoritması veya yazılım tarafından işlenir. Bu çalışmada engelden kaçınma algoritması düzenlenirken Robotino mobil robotun 1, 2, 8 ve 9 numaralı IR sensörleri kullanıldı.

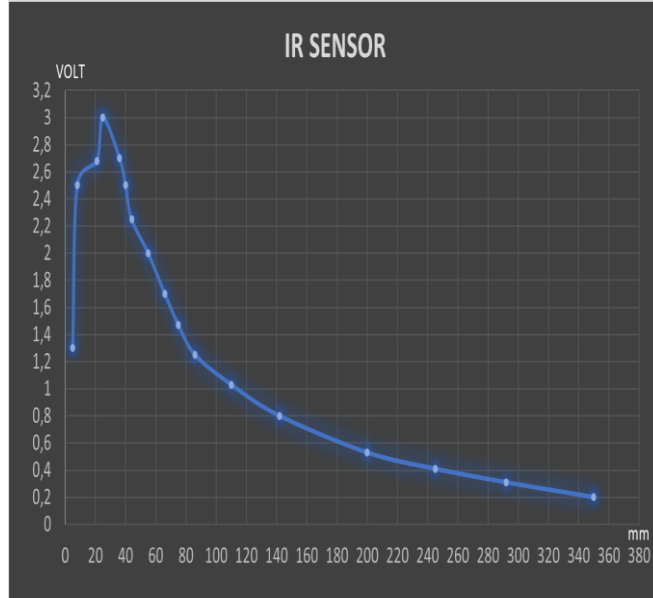


Şekil 4.27 Robotino IR sensör yerleşimi

Robotino IR1 sensör önüne yerleştirilen engel ile 5-350mm arası mesafelerde yapılan gerilim ölçümlerinin sonuçları ve grafiği Tablo 4.12 ile birleşik olarak verilmiştir.

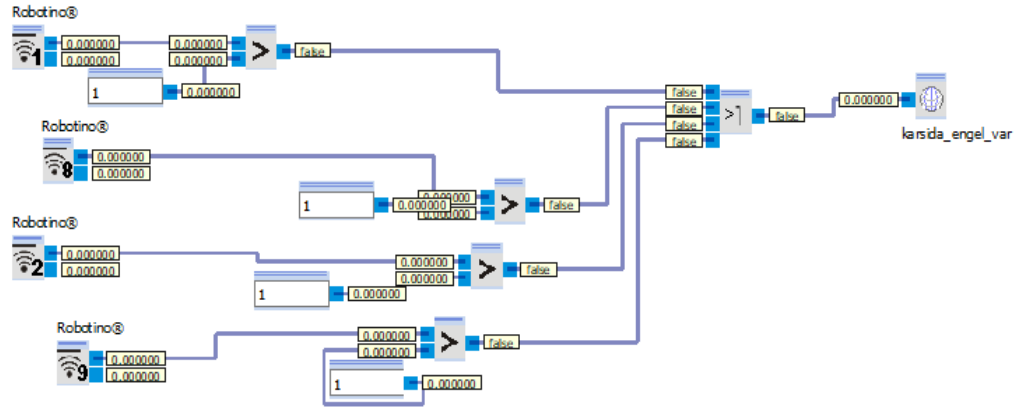
Tablo 4.12 Robotino robot IR sensörlerinin mesafe - gerilim ilişkisi

Mesafe (mm)	Gerilim (V)
5	1.3
8	2.5
21	2,68
25	3
36	2.7
40	2.5
44	2.25
55	2
66	1.7
75	1.47
86	1.25
110	1.03
142	0.8
200	0.53
245	0.41
292	0.31
350	0.2



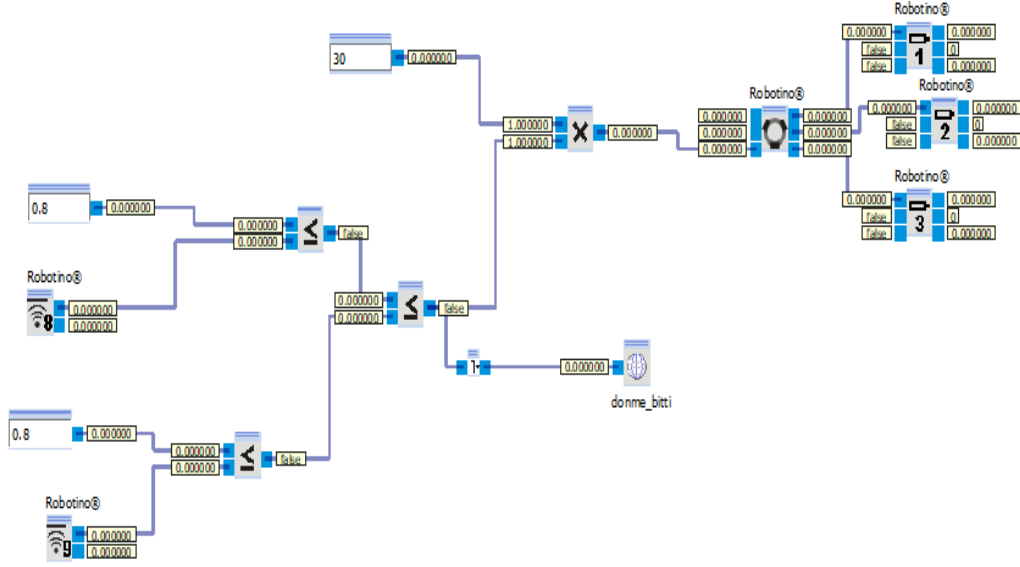
Tablo 4.12'deki grafik incelendiğinde maksimum gerilim 3V ve bu gerilime karşılık gelen mesafe 25mm'dir. En yüksek mesafe ise 0.2V gerilimde 350mm olarak ölçülmüştür.

Şekil 4.28'de görüldüğü gibi Robotino mobil robotun 1, 2, 8 ve 9. sensörlerinin engele yaklaştığındaki mesafeye göre ürettiği gerilimler 1V ile karşılaştırılmıştır. 1V gerilim 110 mm uzaklıktaki bir engeli ifade etmektedir. Robotino önünde 110 mm mesafede bir engel gördüğünde yazılan algoritmaya bilgi gönderecektir. Belirlediğimiz IR sensörlerden gelen veri, “karsıda_engel_var” değişkenini aktif yapacaktır ve algoritma dönme hareketine geçecektir.

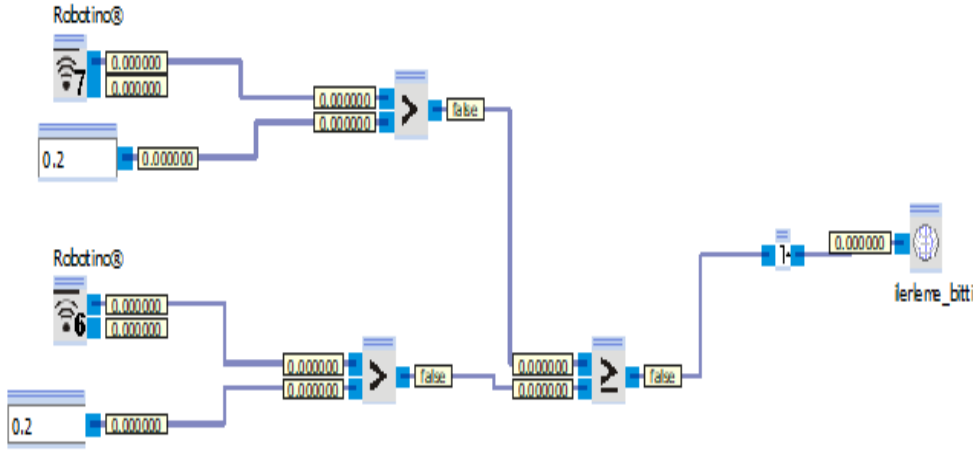


Şekil 4.28 Robotino View programında engelden kaçınma şeması

Şekil 4.29'da görüldüğü üzere 8 ve 9. IR sensörlerden gelen veriler işlenerek dönme hareketini gerçekleştirmektedir. Bu sensörler için 142 mm mesafeye karşılık gelen 0.8V gerilim dikkate alınmıştır. Robotino bir engel gördüğünde belirlenen hızda engel ortadan kalkana kadar sola dönmeye devam edecektir. Engel ortadan kalktığında “dönme_bitti” değişkeni aktif olup ilerleme hareketine geçecektir. İlerleme hareketinde ise 6 ve 7. IR sensörler 0.2V yani 350 mm mesafeyi sorgulayarak bu yöndeki hareketini “ilerleme_bitti” değişkeni ile sonlandırmaktadır.

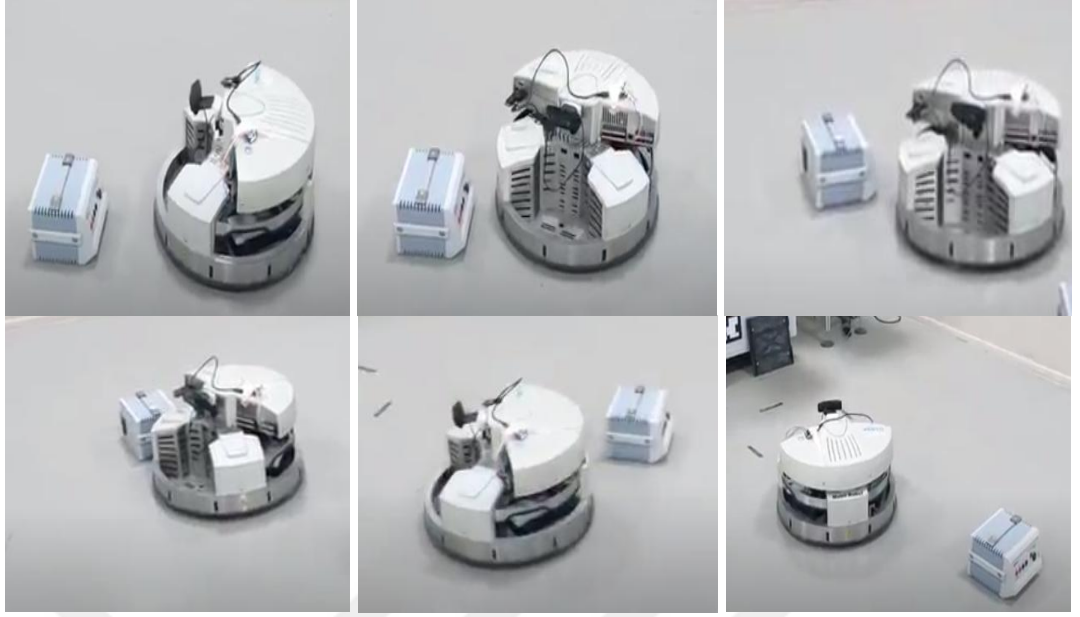


Şekil 4.29 Robotino View programında engel gördükten sonra sola dönme şeması



Şekil 4.30 Robotino View programında dönme hareketi bittikten sonraki ilerleme hareketi şeması

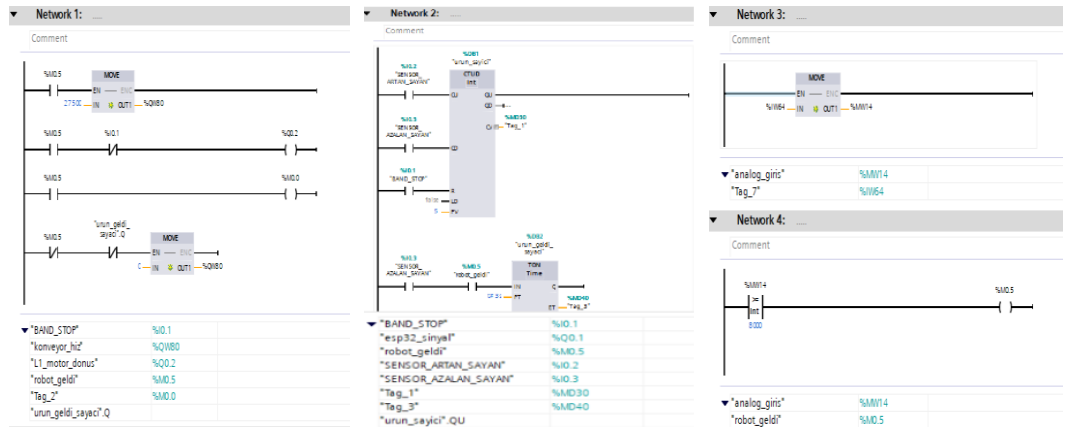
Kısaca açıklayacak olunursa, “engel görüldü→sola dön→hedefe ilerle→tekrar” kontrol edildikten sonra “tekrar→engel gördü→sola dön→hedefe ilerle” mantığı ile engelden kaçınma algoritması oluşturuldu. Şekil 4.30’da ilerleme hareketi bloğu ve Şekil 4.31’de ise deneysel ortamdaki engelden kaçınma algoritması çalıştırıldığında alınan birkaç resim yer almaktadır.



Şekil 4.31 Robotino View programında engelden kaçınma algoritmasının çalıştırılması

4.8 Programlanabilir Denetleyici ile Yürüyen Bant Kontrolü

Bu çalışmada kullanılan S7-1200 Siemens PLC ile yürüyen bant kontrolü yapılmıştır. Yürüyen bandın elektrik motoru 0.09 kW güç, 1500 dev/dk hız ve 1/25 redüktörlüdür. Bu asenkron motor Schneider marka ATV12H075M2 sürücüsü ile sürülmüştür. ESP32 mikrodnetleyiciler ile de Robotino mobil robot ve S7-1200 Siemens PLC aracılığıyla yürüyen bant koordinasyonu gerçekleştirilmiştir. Şekil 4.32’de programlanabilir denetleyici ile yürüyen bant kontrolü programı network1-2-3-4 olarak yer almaktadır.



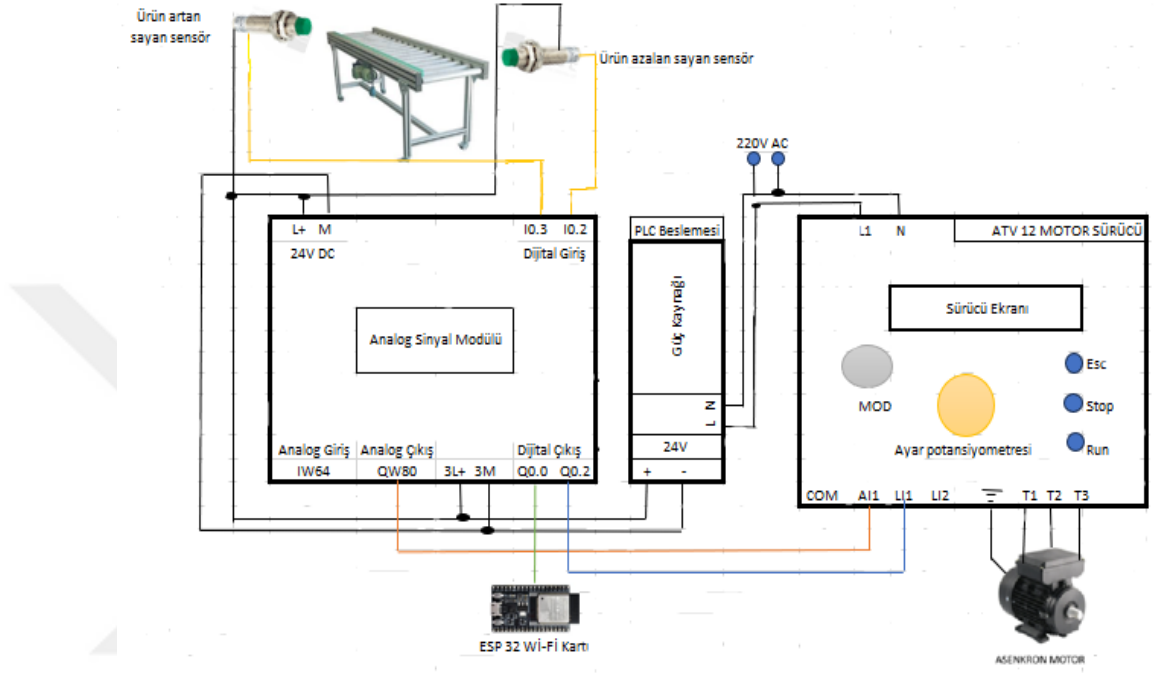
Şekil 4.32 Programlanabilir denetleyici ile yürüyen bant kontrolü programı

Robotino mobil robot, yürüyen bant önüne geldiğinde ve kare işaretleyiciyi okuyup x, y eksen ve phi değerlerine göre lokalizasyon ayarlaması ile dengeleme işlemini gerçekleştirdikten sonra Robotino içerisindeki rölesini çeker ve Network 4'te bulunan %M0.5 bellek biti aktif olur. Robotino, rölesini çektiğinde ESP32 mikrodenetleyicisi PLC'nin %IW64 analog girişine 3.3V gönderir ve bu değer %MW14 bellek sözcüğüne yazılır. S7-1200 PLC'de 27648 sayısı 10V'a karşılık gelmektedir. Karşılaştırma işleminde yaklaşık 3V'a karşılık gelen 8000 sayısı kullanıldı. Buradaki amaç ESP32 kartından gelecek olan 3.3V'u algılamaktır. Belleğe yazılan bu gerilim bilgisinin varlığı karşılaştırma komutu ile belirlendikten sonra %M0.5 bellek biti aktif olur.

Network 1'de ise, aktif olan %M0.5, MOVE komutunun girişine yazılan 27500 değerini PLC'nin %QW80 analog çıkışına yazar ve yürüyen bant çalışmaya başlar. Aynı zamanda %M0.5 motor sürücünün hangi yönde ilerleyeceğini PLC'nin %Q0.2 çıkışını aktif ederek belirler. Bu çıkış motor sürücünün yön bilgisi girişidir.

Network 2'deki sayıcı, yürüyen bandın başındaki ve sonundaki sensörler ile bant üzerine gelen ürünlerin sayım işlemini yapmaktadır. Bant üzerinde sayıcıya atanan ürün sayısından fazla ürün olduğunda sayıcı çıkışı PLC'nin %Q0.1 çıkışını aktif eder. Aktif olan bu çıkışın gerilimi 24V'tur. Bu 24V çıkış bir gerilim çevirici devre ile 3.3V'a çevrilerek PLC üzerindeki ESP32 kartının haberleşme girişine verilir. Bu bilgi ise diğer hattın Robotino mobil robot üzerinde bulunan ESP32 kartının analog1 çıkışını aktif ederek haberleşmesini sağlar. Bu analog veri Robotino View programında seçim kısmında kullanılarak hat değişimi sağlanmıştır. Haberleşmesi sağlanan Robotino şimdi ürün fazlası olan hat için çalışmaya başlayacaktır. Yürüyen bandın çalışma süresini zamanlayıcı sağlamaktadır. Ayrıca yürüyen bant ilerlerken durması için Robotino mobil robotun çeken rölesini bırakması ve dolayısıyla ESP32 kartından gelen bilginin kesilmesi ile %M0.5 bellek bitinin "0" olması gerekmektedir. Aynı zamanda "ürün geldi.Q" sayıcı çıkışının aktif olması gerekmektedir. Bu iki şart sağlandığında yine bir MOVE komutu kullanarak girişine girilen 0 sayısını PLC'nin %QW80 analog çıkışına yüklemesiyle analog çıkış 0V olur ve bant durur.

Şekil 4.33 yapılan çalışmanın yürüyen bant kısmının ana yapısını göstermektedir. Siemens S7-1200 PLC ve PLC'ye bağlı olan sensörler ve ESP32 mikrodnetleyici kartı, analog sinyal kartı ve bağlantıları, PLC güç kaynağı, Schneider marka ATV12H075M2 model AC motor sürücü ve bağlantıları yer almaktadır.



Şekil 4.33 Çalışmanın yürüyen bant kısmının temel yapısı

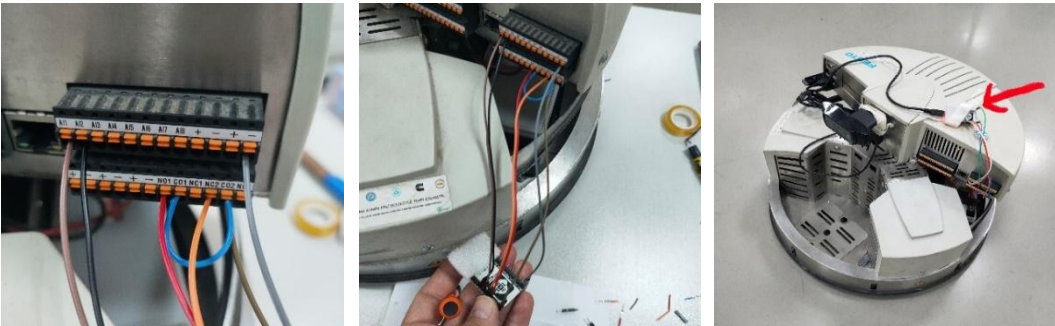
Şekil 4.34'te ise laboratuvar ortamında iki Robotino mobil robotun çalışması görülmektedir.



Şekil 4.34 Laboratuvar ortamında çalışan Robotino'ların görüntüsü

4.9 ESP 32'ler Arası Haberleşme Kontrolü ve Yazılımı

Robotino mobil robotlar üzerinde birer adet ESP32 mikrodnetleyici Wi-Fi kartı ve yürüyen bant üzerinde S7-1200 PLC üzerinde de birer adet ESP32 mikrodnetleyici Wi-Fi kartı kullanılmak üzere 4 adet haberleşme kartı kullanıldı. ESP32 Wi-Fi kartlarından sadece bir tanesi veri gönderici olurken diğer üç tanesi istemci konumundadır. Şekil 4.35'te Robotino robot üzerindeki ESP32 kartı ve bağlantıları görülmektedir.



Şekil 4.35 ESP32 mikrodnetleyici kartının Wi-Fi haberleşme için Robotino mobil robot üzerindeki bağlantısı

ESP32 kartlarının atamaları aşağıdaki şekilde sağlandı.

ESP32 Wi-Fi modülü 1 _____ PLC1

ESP32 Wi-Fi modülü 2 _____ PLC2

ESP32 Wi-Fi modülü 3 _____ Robotino1

ESP32 Wi-Fi modülü 4 _____ Robotino2

PLC1'in bağlı olduğu yürüyen bant1'de birikme var ise Robotino robotun AI1'ine 3.3V gelir. PLC2'in bağlı olduğu yürüyen bant2 'de birikme var ise Robotino robotun AI2'ine 3.3V gelir. Buradaki analog girişlere gelen bilgiler Robotino robotların hat değişimine katkıda bulunur. Robotino robot, yürüyen bant1'in önüne gittiğinde role1 aktif olur. Robotino robot, yürüyen bant2'in önüne gittiğinde role2 aktif olur. Bu her bir Robotino için geçerlidir.

Robotino tarafından değerlendirme yapacak olursak;

Role1 aktif olduğunda “yürüyen bant1 önündeyim” bilgisini, role 2 aktif olduğunda “yürüyen bant2 önündeyim” bilgisini bize vermektedir. Yürüyen bant1'in önünde ürün birikmişse Robotino'nun AI1 analog girişine 3.3V gelir, yürüyen bant2'nin önünde ürün birikmişse Robotino'nun AI2 analog girişine 3.3V gelir. Analog girişler ESP32 kartının 26. pinine bağlıdır. Ürün fazla olduğunda PLC program içindeki sayıcı çıkışının aktif ettiği PLC çıkışı, ESP32 kartının 36. pinine bağlıdır.

ESP32 Wi-Fi modülü ile ilgili bağlantı bilgileri aşağıdaki gibidir.

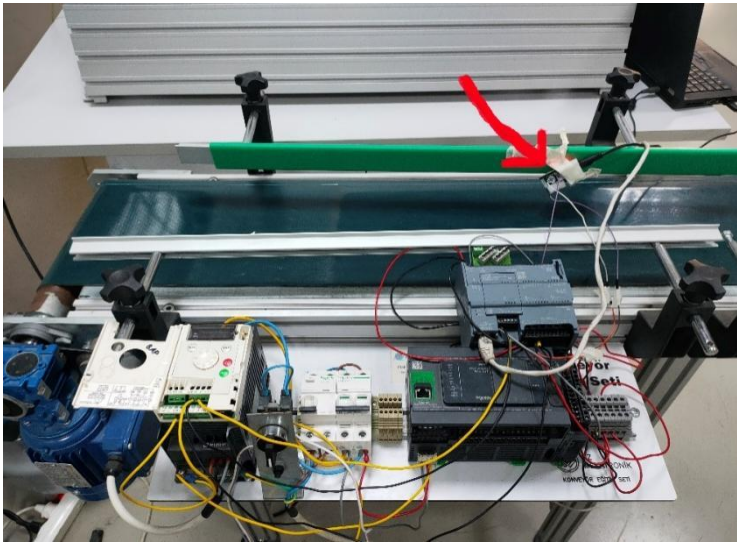
ESP32 Wi-Fi modülü 1: ESP32 modülünün 26.pini (çıkış3) PLC1'in analog girişine bağlanır. 36.pin (giriş1) PLC1'in sayıcı çıkışının aktif ettiği çıkışa bağlanır. PLC1 programındaki sayıcı belirlenen ürün sayısına ulaşıldığında “işlem tamam” bitini kurar ve ESP-NOW ile tüm ağa “1” sinyal verisi gönderilir.

ESP32 Wi-Fi modülü 2: PLC1 sayıcı çıkışının aktif ettiği çıkıştan veri geldiğinde ESP-NOW ağı ile sadece giriş1'e veri girişi olur. ESP32 modülünün

27.pini (çıkış 4) PLC2'in analog girişine bağlanır. 39.pin (giriş 2) PLC2'in sayıcı çıkışının aktif ettiği çıkışa bağlanır.

ESP32 Wi-Fi modülü 3 ve 4: Modül3'ün çıkış1'i Robotino robotun AI1 analog girişine bağlıdır. ESP-NOW ağından PLC1'in Q0.1 çıkış sinyali geldiğinde AI1 aktif olur. Modül3'ün out2 çıkışı Robotino robotun AI2 analog girişine bağlıdır. ESP-NOW ağından PLC2'nin Q0.2 çıkış sinyali geldiğinde AI2 aktif olur. Modül3'ün input1'ine sinyal geldiğinde bütün ağa istasyon 2'ye gittiği sinyali gönderir.

ESP32 modülünün 34.pini (giriş 3) Robotino robotun role1'ine bağlanır. Bu ise “yürüyen bant1 önündeyim” bilgisini verir. ESP32 modülünün 35.pini (giriş 4) Robotino robotun role2'ine bağlanır. Bu ise “yürüyen bant2 önündeyim” bilgisini verir. ESP32 modülünün 33.pini (çıkış1) Robotino robotun AI1 girişine bağlanır ve bu girişe sinyalin gelmesi demek yürüyen bant1'de ürün birikmesi anlamına gelmektedir. ESP32 modülünün 25.pini (çıkış2) Robotino robotun AI2 girişine bağlanır ve bu girişe sinyalin gelmesi yürüyen bant2'de ürünün biriktiği anlamına gelmektedir. Şekil 4.36'da ESP32 mikrodnetleyici kartının Wi-Fi haberleşme için yürüyen bant üzerindeki PLC ile bağlantısı gösterilmiştir.

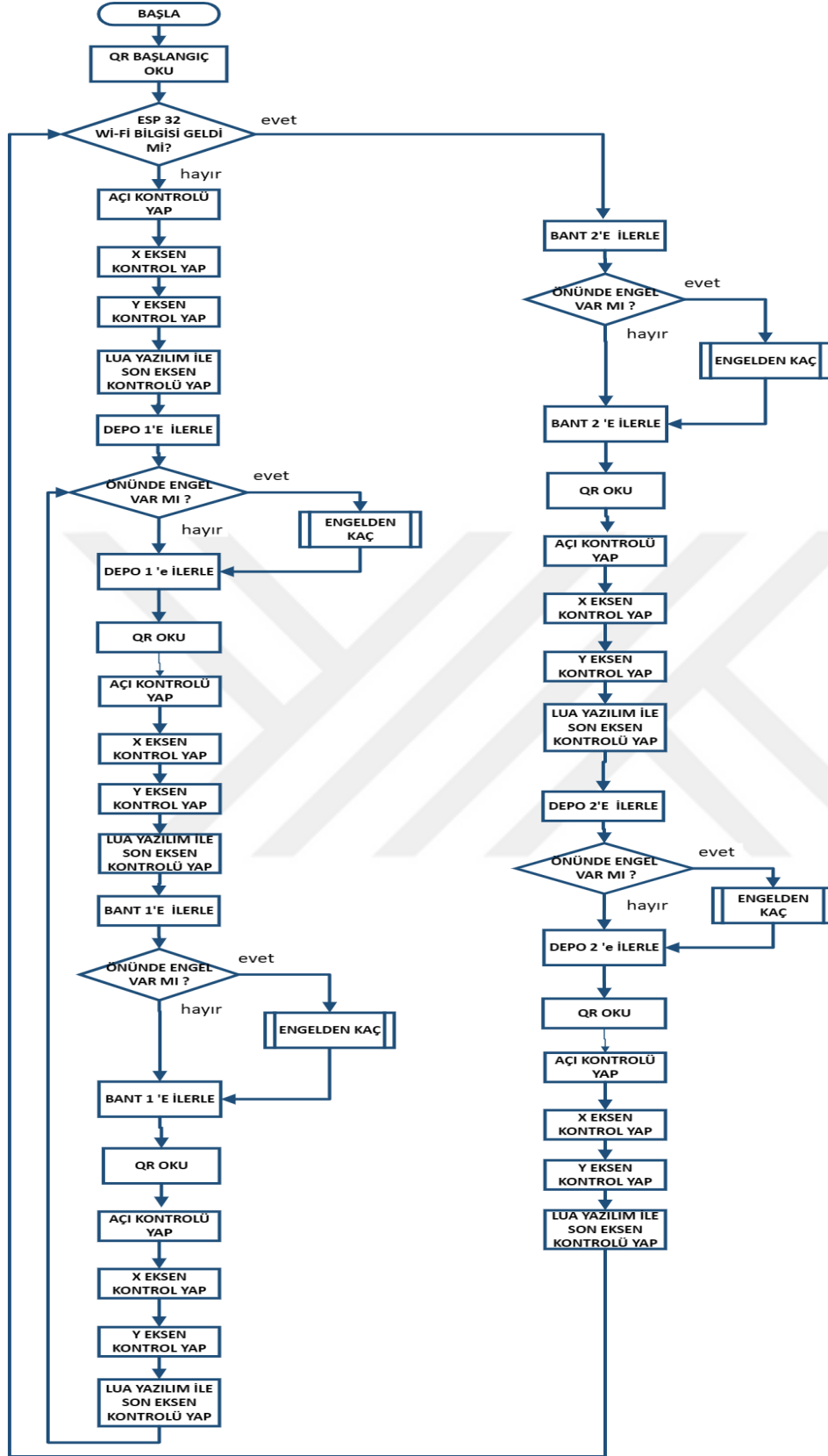


Şekil 4.36 ESP32 mikrodnetleyici kartının Wi-Fi haberleşme için bant üzerindeki PLC ile bağlantısı

4.10 Algoritma

Şekil 4.37'deki algoritma her iki Robotino mobil robot için kullanılmıştır. Algoritma yürüyen bantların önlerinde Robotino robotun konumlandırılması ile başlamaktadır. Robotino robot enkoderi sıfırladıktan sonra QR kodunu okuyarak algoritmaya başlamaktadır. QR kodu okumadaki amaç başlangıç noktasındaki x , y ve ϕ eksen değerlerini daha sonraki QR okuma işlemi sonrası karşılaştırma yapmak için kaydetmektir. QR kodu okunduktan sonra, algoritma ϕ , x ve y eksenleri için ayrı ayrı kontrol yapmaktadır. Yani, her eksen kontrolünde başlangıç konumlarına göre ileride veya geride, yukarıda veya aşağıda, solda veya sağda kontrolleri ayrı ayrı yapılır ve sıfır noktasına göre kaçıklıklar var ise Robotino robotlar kaçıklıkların tam tersi şekilde hareket ederek kaçıklıkları dengeleme işlemini sağlarlar. Ayrıca, Lua bloğu ile yazılan yazılım tekrardan kontrol yapmaktadır. Robotino mobil robotlar, yürüyen bant ve depo önüne her gittiklerinde bu işlemleri tekrarlayarak konumlandırmalarını gerçekleştirmektedir.

ESP32 mikrodnetleyicisi ile Wi-Fi bilgisi kontrolü yapılarak Robotino robotların hangi hatta çalışacağı belirlenir ve hizmet vereceği hattaki görevlerini yürütürler. Hedef noktaya gidecek olan Robotino'lar önlerine çıkan engellerden, engel gördüğünde “sola dön→ilerle”, engel varsa “sola dön→ilerle” hedefe yönel mantığını gözeterek kaçacaktır.



Şekil 4.37 Robotino mobil robotun engelden kaçarak diğer Robotino mobil robot ile koordineli çalışmasını gösteren algoritma

Yürüyen bant önüne gelen ve kaçıklıkları dengeleme işlemi yapan Robotino robot, ESP32 kartı üzerinden yine PLC üzerindeki ESP32 kartı ile Wi-Fi haberleşerek yürüyen bandı çalıştırıp ürünü alacaktır ve depoya doğru ilerleyecektir.

Robotino robotlar yürüyen banttı ürün alıp depoya bıraktıktan sonra her defasında ESP32 kartındaki hat seçimini kontrol ederek hangi hatta çalışacağını belirleyecektir. Hat değişimi olduğunda, Robotino robotlar daha önceki işlem sırasını gözeterek hareket edecektir. Yani eğer ürün almışsa depoya bıraktıktan sonra diğer yürüyen banda doğru harekete geçecektir.



5 SONUÇ

Dinamik ortamlarda çalışan çoklu otonom robotlar için kontrol algoritması geliştirilmesi için yapılan tez çalışmasının ana hedefi Robotino mobil robotların başlangıç konumlarından hedeflenen konumlarına giderken durağan ve hareketli engellerden kaçınması ve robotların kendi arasındaki iş yoğunluğuna göre iş koordinasyonunun sağlanmasıdır. Bilgisayar ile Robotino mobil robot kablosuz haberleşmiş olup aynı zamanda hem Robotino robotların kendi arasındaki hem de yürüyen bantları kontrol eden PLC'ler ile Robotino mobil robotlar arasındaki Wi-Fi haberleşmesi ESP32 Wi-Fi modülü ile sağlanmıştır. Deneyler sırasında Robotino mobil robotun temel hızı 100mm/s ve dönüş hızı 30mm/s olarak ayarlanmıştır. Bu kapsamda tez çalışmasının ana hatları tamamlanmış olup Robotino mobil robotun konumlandırma problemini gidermek için çalışmalar yapılmıştır.

Bu çalışmanın ana hatları tamamlandıktan sonra karşılaşılan en büyük problem Robotino mobil robotun odometri okuma hataları olmuştur. Odometri okuma hataları nedeniyle hedeflenen konumlara gidildiğinde eksen ve phi kaçıklık hataları ile karşılaşılmıştır. Önce bu hataların sistematik bir hata olup olmadığının belirlenmesi için başlangıç noktasından sadece x ekseninde 500mm ve 1000mm için 10'ar kez ileri gidiş ve geriye dönüş denemeleri yapılmış, bu denemelerde x, y ve phi eksen kaçıklıkları gözlemlenmiştir. Alınan sonuçlardan çıkan grafikler odometri kaçıklık hatalarının sistematik hata olmadığını göstermiştir. Robotino View programı içerisindeki konum sürücüsü bloğunun x, y ve phi için hızlanma yapılandırmalarını yaparak eksen kaçıklıklarındaki hataları azaltmak hedeflenmiş ve non-holonomik sürüş tekniği kullanılmıştır. Robotino mobil robot ile yapılan başka çalışmalarda da aynı problemler ile karşılaşılmış ve bunun üstesinden gelebilmek için çalışmalar yapılmıştır.

Mbede et al. (2012), kapalı ortamlarda zSlice Tip-2 bulanık mantık navigasyon yöntemi kullanarak Robotino mobil robotu kontrol etmek için Robotino mobil robot sensörleri olan kızılötesi optik sensör, jiroskop ve gömülü web kamerası tarafından sağlanan sürekli akış verilerini kullanmıştır. Sistem, web kamera tarafından sağlanan video akışı ile başlatılmış ve uzaklık tahmini yapmak

için renkli bir nesneyi tespit etmesi gerekmiştir. Odometri hatalarını azaltmak için sadece doğrusal yer değiştirmeler (yanal ve boylamsal hız kontrolü) ile ilgilenilmiş ve hedef takibi için video ve görüntü işleme sistemindeki sapma oranı kullanılmıştır. Bu çalışmada, robotun hareketli bir engel ile karşılaşması durumunda görüntü boyutu artacağı için yörüngenin doğru bir şekilde tahmin edilmesinin ve mesafe tahmininin zorlaşacağını ifade etmişlerdir. Bu tez çalışmasında ise kamera ile renk okuma ve hedeflenen konuma yerleştirilen 10cm×10cm kare renklerin alanını okuyarak konumlandırma işlemleri başarı olmamıştır.

Eberhard and Tang (2013), çoklu robot işbirlikçi konum belirleme için robot odometrisine ve veri füzyonu için harici Kuzey Yıldızı sinyallerine dayanan genişletilmiş Kalman filtresini (EKF) ilk olarak her bir robotun konumlandırması için kullanmıştır. Robotlar üzerindeki kızılötesi sensörler ve jiroskoplar tarafından alınan verilere göre ve EKF tarafından tahmin edilen konumlar ile konumlandırma sürecini hassaslaştırılmaya çalışmışlardır. Kuzey yıldızı ve jiroskop verilerinin ‘belirsizlik hacmi’ yöntemi ile birleştirilerek konumlandırmayı olumlu yönde etkileyerek x ve y konumlandırma kesinliğini arttırdığını belirtmişlerdir. Çalışmanın dağıtık EKF ve göreceli gözlem sayesinde çeşitli sensörlerden toplanan ölçümleri daha az iletişim yükü ve işlem gereksinimi ile birleştirmek için bir araç oluşturmak amacıyla yapılmış olduğu belirtilmiştir. Robotino Kuzey Yıldızı sistemi oldukça pahalıdır ve çalışma alanı genişledikçe konumlandırma doğruluğu düşmektedir. Yapılan bu tez çalışmasında ise anlık odometri kaçıklık hatalarını belirlemek yerine çalışma alanının boyutlarından bağımsız bir yöntem kullanılarak hedeflenen son noktada odometri kaçıklık hatalarını düzeltmek yoluna gidilmiştir. Robotino mobil robot ile tümleşik kamera ve sensörler kullanılarak düşük maliyetli bir lokalizasyon sağlanmıştır.

Bu tez çalışmasında kullanılan iki adet Robotino mobil robotun başlangıç noktaları ve gideceği hedef noktaları olmak üzere dört adet konum bilgisi mevcuttur. Konumlandırmadaki problemi ortadan kaldırmak için başlangıç ve hedef noktalara farklı renklerde materyal yapıştırılarak web kamerası renk görüntüsü işlenerek konumlandırmadaki hatalar giderilmeye çalışılmış fakat başarılı sonuç vermemiştir. Bu çalışmanın yerine yine renkli materyali kamera ile

okuyup Robotino View programındaki bloğa kaydedilmiş olan alan bilgisini, robot hedeflenen konuma geldiğinde yine robotun kamerası ile istenilen alan bilgisini okuyana kadar konumlandırmasını yaparak odometri kaçıklık hataları düzeltilmek istenilmiş fakat alan okumadaki hatalar, verimli çalışmaya engel olmuştur. Çözüm olarak, Robotino mobil robotların başlangıç noktaları ve gideceği hedef noktalarına 12cm×12cm kare işaretleyiciler yapıştırılmıştır. Robotino'ların başlangıç noktalarındaki kare işaretleyici bilgileri, sürüş başladığı zaman x, y ve phi bilgisi olarak program blokları içine kaydedilmiştir, dolayısı ile artık bu kaydedilen veriler referans noktaları olarak kullanılmıştır. Hedef noktaya gidip tekrar başlangıç noktasına gelen Robotino, kare işaretleyici bilgisini kontrol ederek x, y ve phi odometri kaçıklıklarını Robotino View içindeki Lua yazılımı ile yazılan program vasıtasıyla denetleyerek dengeleme işlemini gerçekleştirmiştir. Bu yazılım, dört noktadaki kare işaretleyiciler için yenilenerek algoritmaya eklenmiştir. Sadece başlangıç noktalarında değil aynı zamanda hedef noktaya varıldığında da kare işaretleyici okunarak Lua yazılımı ile konumlandırmadaki kaçıklıklar giderilmeye çalışılmıştır. Kare işaretleyicilerinden alınan bilgilerin yardımıyla Lua yazılımı ile x, y ve phi kaçıklıklarını gidererek konumlandırmadaki kaçıklık hatalarını en aza indirilmiştir. Uzun çalışma süreleri göz önüne alındığında konumlandırmada kaçıklıkların olduğu görülmüştür fakat bu kaçıklık hataları azalmıştır. Odometri kaçıklık hatalarını gidermek için yapılan diğer çalışmalara harici sensörler eklenmiştir ve bu hatalar tamamen ortadan kaldırılamamış, azaltılmaya çalışılmıştır. Çalışmalara eklenen harici sensörler sistemi iyileştirirken maliyetini arttırmıştır.

Bu tez çalışmasında daha önce çalışılmamış olan Lua yazılım dili kullanılmıştır ve konumlandırmada iyileştirmeler sağlanmıştır. Lua yazılımında, Robotinolar başlangıç noktaları olan yürüyen bant önlerindeki kare işaretleyicilerin x, y ve phi bilgilerini okunur ve Robotino View programı içinde oluşturduğumuz değişkenlere kaydedilir. Robotino robot ürünü depoya bırakıp tekrar başlangıç noktasına döndüğünde kare işaretleyici yeniden okutturulur. Daha önceki değişkenlere kaydettiğimiz x, y ve phi değerleri ile yeniden okunan x, y ve phi değerleri karşılaştırılır. X eksen hız ve yön kontrolü hizOne, y eksen hız ve yön kontrolü hizYan değişkenleri ile ifade edilmiştir. Önce hizYan için yeni okunan x eksen konum bilgisi ilk okunan değerin sağında mı solunda mı olduğu

karşılaştırılır. Sağında ise, robot -10mm/s hızla sola doğru hareket ederek ilk konum bilgisine ulaşana kadar ilerler. Solunda ise 10mm/s hızla sağa doğru hareket ederek ilk okunan konum bilgisine ulaşana kadar ilerler. hizOne için yeni okunan y eksen konum bilgisi ilk okunan değerin ilerisinde mi gerisinde mi olduğu karşılaştırılır. İlerisinde ise -10mm/s hızla geriye doğru hareket ederek ilk konum bilgisine ulaşana kadar ilerler. Gerisinde ise 10mm/s hızla ileri doğru hareket ederek ilk okunan konum bilgisine ulaşana kadar ilerler. hizAci ile konum bilgisi kontrolü Lua ile yapılmamış olup Robotino View programındaki algoritma ile phi açısı kontrolü sağlanmıştır.

Tablo 4.5'teki değerleri incelediğimizde, Robotino mobil robot başlangıç konumlarına göre sadece x ekseninde 1000mm hareket için engelsiz sürüşte yürüyen bant önünden depoya gidip tekrar geri geldiğinde y ekseninde sola doğru toplamda 169 mm ve x ekseninde geriye doğru 82 mm sapmıştır. Deney ortamında x ekseninde 2500mm'lik sürüş mesafesinde ve engelli ortamda Lua yazılımı ile kare işaretleyiciyi okuyarak yapılan sürüşte bu kaçıklık değerleri x ve y ekseninde ortalama 0-18 mm aralığında dengelenerek gerçekleşmiştir. Fakat, IR sensörlerin engeli görmediği kör noktalardaki durumlarda Robotino robotun engele sürterek geçmesinin de etkisiyle ve dolayısıyla odometrinin okuduğu değerin etkilenmesiyle, robotların uzun süreli çalışmasını da göz önüne aldığımızda 0-18 mm aralığı ilerleyen süreçlerde genişlemektedir.

Konumlandırmadaki kaçıklık hatalarını azaltmak için kullanılacak ek sensör maliyetlerinden kaçınılarak bu kaçıklık hataları kare işaretleyici kullanılarak ve Lua yazılım dilinde yazılan program yardımıyla azaltıldı. Konumlandırmadaki yazılım ve ESP32 Wi-Fi haberleşme ile Robotino mobil robotlar ve yürüyen bantlar arası iletişim alt yapısı hazırlandı. Robotino'ların çizgi izleme, iki çizgi arasında hareket etme veya belirlenen alanda hareket etmesi yerine oluşturulan algoritma ile sınırlama olmadan her ortama uyarlabilir olması önemli bir avantajdır.

Robotino mobil robot engelden kaçarken, engelin büyüklüğüne göre IR sensörler engelin kör noktasına gelebilmektedir. Bu ise engele sürtünerek geçme veya dokunarak geçmesine neden olmakta ve odometri sayacının yanlış

okumasına sebebiyet vermektedir. Engelden kaçınma algoritmasında kullanılmış olan IR1, IR2, IR8 ve IR9 sensörleri Robotino mobil robot üzerine 40 derece açı ile yerleştirilmiştir. Daha verimli çalışma için bu sensörler aralarına ek IR sensörler eklenerek engelden kaçınma algoritması geliştirilebilir. Eklenen bu sensörler ile Robotino mobil robotun odometrisinden alınan veriler daha doğru okunarak konumlandırmaya olumlu yönde etki edecektir. Ayrıca, Robotino mobil robot başlangıç noktasından hedef noktaya giderken birkaç noktada tekrar kare işaretleyici okutularak konum sıfırlaması (güncellemesi) yapıp daha uzun vadede kaçıklıklardan etkilenmesi azaltılabilir.



6 KAYNAKLAR DİZİNİ

J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. doi: 10.1109/CVPR.2015.7298965.

J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.690.

C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.699.

D. Scaramuzza and F. Fraundorfer, “Tutorial: Visual odometry,” *IEEE Robot Autom Mag*, 2011, doi: 10.1109/MRA.2011.943233.

H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM): part I The Essential Algorithms,” *Robotics & Automation Magazine*, 2006, doi: 10.1109/MRA.2006.1638022.

A. Doucet, N. De Freitas, N. J. Gordon, and others, *Sequential Monte Carlo methods in practice*, vol. 1, no. 2. Springer, 2001.

V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

T. Lozano-Pérez and M. A. Wesley, “An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles,” *Commun ACM*, 1979, doi: 10.1145/359156.359164.

G. Antonelli, F. Arrichiello, and S. Chiaverini, “The null-space-based behavioral control for autonomous robotic systems,” *Intell Serv Robot*, 2008, doi: 10.1007/s11370-007-0002-3.

Ö. F. Varol, “Kooperatif gezgin robotlarda kontrol tabanlı sezgisel hareket planlama,” 2009.

N. Bölük *et al.*, “Mobil Robotlarda Navigasyon Problemi için Pekiştirmeli Öğrenme,” 2019.

H. Bayram, “Multi Robot Communication and Task Coordination,” *Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi*, 2013.

C. Revelo, M. Trujillo, A. Rosales, and D. Pozo, “Indoor localization by using stereoscopic vision, odometry, and the Kalman Filter,” 2014.

E. Budak, “Yük taşmayı otonom olarak gerçekleştirebilen çoklu mobil robot uygulaması,” *Fen Bilimleri Enstitüsü*, 2015.

A. Prorok, M. A. Hsieh, and V. Kumar, “The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms,” *IEEE Transactions on Robotics*, 2017, doi: 10.1109/TRO.2016.2631593.

E. DÖNMEZ and A. F. KOCAMAZ, “Çoklu Hedeflerin Çoklu Robotlara Paylaştırılması İçin Bir Yük Dengeleme Sistemi,” *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, 2019, doi: 10.17798/bitlisfen.467757.

Y. Wang and C. W. de Silva, “A machine-learning approach to multi-robot coordination,” *Eng Appl Artif Intell*, vol. 21, no. 3, pp. 470–484, 2008, doi: <https://doi.org/10.1016/j.engappai.2007.05.006>.

B. Park, C. Kang, and J. Choi, “Generating Multi-agent Patrol Areas by Reinforcement Learning,” in *2021 21st International Conference*

on Control, Automation and Systems (ICCAS), 2021, pp. 104–107. doi: 10.23919/ICCAS52745.2021.9650047.

H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, “Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments,” *arXiv preprint arXiv:2005.13857*, 2020.

S. H. Tang and W. Khaksar, “Obstacles Avoidance Mobile Robot System in Uncertain and Ever- Changing Surroundings,” *PJSRR Pertanika Journal of Scholarly Research Reviews*, 2016.

A. Abdo, R. Ibrahim, and N. A. Rawashdeh, “Mobile Robot Localization Evaluations with Visual Odometry in Varying Environments Using Festo-Robotino,” in *SAE Technical Papers*, 2020. doi: 10.4271/2020-01-1022.

Q. Tang and P. Eberhard, “Relative observation for multi-robot collaborative localisation based on multi-source signals,” *Journal of Experimental and Theoretical Artificial Intelligence*, 2014, doi: 10.1080/0952813X.2014.930597.

S. E. Oltean, M. Dulau, and R. Puskas, “Position control of Robotino mobile robot using fuzzy logic,” in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2010 - Proceedings*, 2010. doi: 10.1109/AQTR.2010.5520855.

K. Belda and J. Jirsa, “Control Principles of Autonomous Mobile Robots Used in Cyber-Physical Factories,” in *Proceedings of the 2021 23rd International Conference on Process Control, PC 2021*, 2021. doi: 10.1109/PC52310.2021.9447468.

P. Anggraeni, I. Rokhim, and R. M. Salam, “Design and development of multiple mobile manipulator robots using gazebo-ros,” in

2020 *International Conference on Applied Science and Technology (iCAST)*, 2020, pp. 672–676.

S. W. Mukhlisin, H. Rachmat, and T. Mulyana, “Perancangan Sistem Storage And Retrieval Machine Pada Simulasi Automated Storage And Retrieval System Dengan Memanfaatkan Robotino®,” *JRSI (Jurnal Rekayasa Sistem dan Industri)*, vol. 3, no. 01, pp. 27–32, 2016.

M. S. Masmoudi, N. Krichen, M. Masmoudi, and N. Derbel, “Fuzzy logic controllers design for omnidirectional mobile robot navigation,” *Applied Soft Computing Journal*, 2016, doi: 10.1016/j.asoc.2016.08.057.

7 TEŞEKKÜR

Tez çalışması sürecinde fikirlerinden faydalanarak gerekli motivasyonu bulmamı sağlayan değerli arkadaşım ve meslektaşım Öğr.Gör. Okan Duymazlar'a ve beni yönlendiren, akademik disiplini öğreten bunun yanında bu çalışmamda bana güveni sürekli hissettiren değerli hocam Dr. Öğr. Üyesi Dilşad Engin'e teşekkürü bir borç bilirim.

12 / 01 / 2024

8 ÖZGEÇMİŞ

SERKAN PINAR**EĞİTİM BİLGİLERİ**

Lise/Üniversite	Bölüm	Mezuniyet Yılı	Derece
Tuzla Teknik Lisesi	Elektronik	2000	3,05 / 4
Ege Üniversitesi/Ege MYO	Endüstriyel Elektronik	2003	2,92 / 4
Pamukkale Üniversitesi /Teknik Eğitim Fakültesi	Elektronik Öğretmenliği	2007	2,7 / 4

DENEYİM

Kurum/Kuruluş	Bölüm	Görev Türü	Görev Dönemi
Parsan Mak.Parç.San A.Ş.	Elektronik Bakım	Stajyer/Yaz Stajları	1998-1999-2000-2002-2003
Gamak A.Ş.	Elektrik-Elektronik Bakım	Tekniker	2004 ve 2009
Gamak A.Ş.	AR-GE Laboratuvar	Laboratuvar sorumlusu	2010-2013
Ege Üniversitesi/Ege MYO	Kontrol Otomasyon Teknolojisi	Öğretim Görevlisi	2013-

Hakemli konferans/sempozyumların bildiri kitaplarında yer alan yayınlar

Dinamik Ortamda Çalışan Çoklu Mobil Robotlar İçin Görüntü İşleme Tabanlı Gezinme ve Konumlandırma, EGECON 2023