YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİR GEZGİN ROBOT İÇİN ELEKTRONİK DENETİM YAZILIMININ TASARIMI VE UYGULAMASI

Elektronik ve Hab. Müh. Ekin YILDIZ

F.B.E. Elektronik ve Haberleşme Mühendisliği Anabilim Dalında Elektronik Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı :Yrd. Doç. Dr. Tuncay UZUN

Prof. Dr. Oruç Bilgiç

İSTANBUL, 2000

İÇİNDEKİLER

SİMGE LİSTESİ

L_u	Dümen mekanizmalı gezgin robotun sürüş ve dümen tekerlekleri arası uzunluğu
$r_{dönme}$	Diferansiyel sürüşlü gezgin robotun dönüş yarıçapı
r_{min}	Dümen mekanizmalı gezgin robotun minimum dönüş yarıçapı
$\theta_{dönme}$	Diferansiyel sürüşlü gezgin robotun dönme açısı
$\theta_{dümen}$	Dümen mekanizmalı gezgin robotun tekerlek açısı
θ_{servo}	Dümen mekanizmalı gezgin robotun servo motor açısı
$V_{sürüş}$	Dümen mekanizmalı gezgin robotun sürüş hızı

KISALTMA LİSTESİ

AGV	Automated Ground Vehicles (Otomat Yer Taşıtı)
API	Application Program Interface (Uygulama Programı Arabirimi)
AR-GE	Araştırma-Geliştirme
ASCII	American Standard Code for Information Interchange (Bilgi Aktarımı için Amerikan Standart Kodu)
CD	Carrier Detect (Taşıyıcı Algılama)
DC	Direct Current (Doğru Akım)
GUI	Graphic User Interface (Görsel Kullanıcı Arabirimi)
I/O	Input/Output (Giriş/Çıkış)
MS	Microsoft
PC	Personal Computer (Kişisel Bilgisayar)
PWM	Pulse Width Modulation (Darbe Genişlik Modülasyonu)
RTS	Request to Send (Gönderme İsteği)
SAIL	Stanford Artificial Intelligence Language (Stanford Yapay Zeka Dili)

ŞEKİL LİSTESİ

ÖNSÖZ

Başta; bu uzun yolda yardımlarını ve desteklerini hiç esirgemeden, karşılaştığım tüm zorluklara ve sıkıntılara benimle birlikte göğüs geren arkadaşlarım, Arş. Gör. Seçil Özen ve Arş. Gör. Aydın Melek'e; tez danışmanım Yrd. Doç. Dr. Tuncay Uzun'a; yardımlarından dolayı Arş. Gör. Bülent Bolat ve Arş. Gör. Refet Ramiz'e, anlayış ve desteği için Yrd. Doç. Dr. Herman Sedef'e ve motivasyonu için Prof. Dr. Ertuğrul Eriş'e teşekkür ederim.

ÖZET

Gezgin robot; verilen bir yön ve konum boyunca hareket edebilme yeteneği olarak tanımlanabilen gezinmesine göre, otomatik olarak hareket edebilen, serbest programlanabilir veya özerk bir araçtır. Bu tezde, verilen bir görevi yerine getirmek için gerekli adımları gerçekleyecek bilgisayar kontrollü bir robot prototipinin denetim yazılımı tasarlanıp, gerçekleştirilmiştir. Bu amaçla diferansiyel sürüşlü ve dümen mekanizmalı olmak üzere iki adet gezgin robot modelinin mekanikleri incelenmiş ve uzaktan kontrolleri için yazılımları gerçekleştirilmiştir.

Bu bağlamda; ilk bölümde, ülkemizde de yeni bir araştırma konusu olan gezgin robot kavramı tanımlanarak, tarihçesi verilmiştir. Robot kol ve gezgin robot ile ilgili yapılan robot programlama dilleri incelenmiştir.

İkinci bölümde; tezde tasarlanıp, gerçekleştirilen gezgin robot sisteminin denetim yazılımı ile ilgili bilgi verilmiştir. Nasıl bir sistemin tasarlandığı, bu bölümde anlatılmıştır. Ayrıca diferansiyel sürüşlü ve dümen mekanizmalı gezgin robotların mekanik denklemleri verilmiş ve kişisel bilgisayardan robotta bulunan denetim kartına gönderilen komutların yapısı, bu bölümde açıklanmıştır. Üçüncü bölümde; yazılımın gerçeklenmesinde kullanılan Pascal temelli görsel bir dil olan Delphi kısaca anlatılmış ve yazılımda kullanılan fonksiyonlar, kontroller ve özellikleri ilgili bilgi verilmiştir. Dördüncü bölümde ise, tasarlanıp gerçekleştirilen gezgin robot kontrol programının özellikleri verilmiş, çalışması ayrıntılı bir biçimde anlatılmıştır.

Tezin son kısmında tasarlanan ve uygulaması yapılan gezgin robot yazılımı değerlendirilerek sonuçlar ve gerçekleştirilen programın ayrıntılı dökümü verilmiştir.

Anahtar Kelimeler: Gezgin robot, diferansiyel sürüş, dümen mekanizması, denetim yazılımı.

ABSTRACT

In this thesis, the control software of a computer controlled mobile robot prototype that will realize the steps, which are necessary to achieve the goal procedure, is designed and implemented. For this purpose the mechanical parts of a differential mobile robot and a mobile robot which has a steering mechanism is examined and software controls for these robots is designed and realized. The main object is investigation and realization of the design of software control module. From this point of view, first section is an introduction to the concept and historical development of mobile robot. This section also gives the research results about software controls of robot manipulators and mobile robots to give an idea of robot programming languages. Second section gives the mechanical equations of a differential mobile robot and a mobile robot which has a steering mechanism. The command structure that will be transmitted to the control card of the system, is explained. This section has attempted to give some insight as how computational elements and software come together to form a robot controller. The use of computers and computational elements is essential just as the presence of the brain in an intelligent animal is essential. In the third section, Delphi is explained, which is used in the program. Pascal is not mentioned, but you'll see that Pascal programming skills is used to develop Windows applications with Delphi. A basic knowledge of Pascal is beneficial, but is only necessary to understand relatively small blocks of code. This chapter discusses about all the components that are used in the robot application. The fourth section gives an explanation about the mobile robot program that is designed. The communication standards that are used and command string adjustments for both differential mobile robot and a mobile robot which has a steering mechanism, is explained. At last the results and the print out of the program is given.

Keywords: Mobile robot, differential mobile robot, steering mechanism, control software.

1. GİRİŞ

Günümüzde, bilim ve teknoloji alanında çok boyutlu gelişmelere tanık olmaktayız. Geçtiğimiz yüzyılın birinci yarısının, genellikle bir donanım egemenliği altında geçtiğini söyleyebiliriz. Bu sürede; endüstrideki gelişmeler genellikle endüstriyel tesislerin yapısında bulunan mekanik kısmın daha karmaşık, daha hassas ve daha hızlı bir duruma gelmesi şeklinde olmuştur. Bu nedenle bu döneme "donanım çağı" adını verebiliriz. Yüzyılın daha sonraki yılları benzer bir şekilde "yazılım çağı" olarak adlandırılabilir, çünkü bu yıllarda teknolojik gelişmelerin öncüsü yazılım konusundaki gelişmeler olmuştur. Başlangıçta; o günlere kadar iletişim alanı uygulamaları ile sınırlı kalmış elektronik kullanımı, endüstriyel tesislere de yaygın olarak girmiştir. Tümleşik devreler ve özellikle mikroişlemciler alanında 1970'li yıllarda görülen gelişmeler endüstriyel evrime yeni bir ivme kazandırmış ve karmaşık işlevlerin uygun bir yazılımla, basit ve ekonomik bir şekilde gerçekleştirilmesi olası olmuştur.

Daha sonraki yıllarda ise farklı disiplinlerin, özellikle mekanik ve elektronik teknolojilerinin bilgisayar teknolojisi ile işbirliği içinde entegrasyonuna tanık olduk. Sonuçta belirli bir derecede otonomi ve zeka içeren bilgisayar denetimli süreçler ortaya çıktı ve insanların IQ'sundan bahsettiğimiz gibi makinelerin IQ'sunda da bahseder duruma geldik. Böylece mekatronik çağına girmiş olduk (Kaynak, 1998).

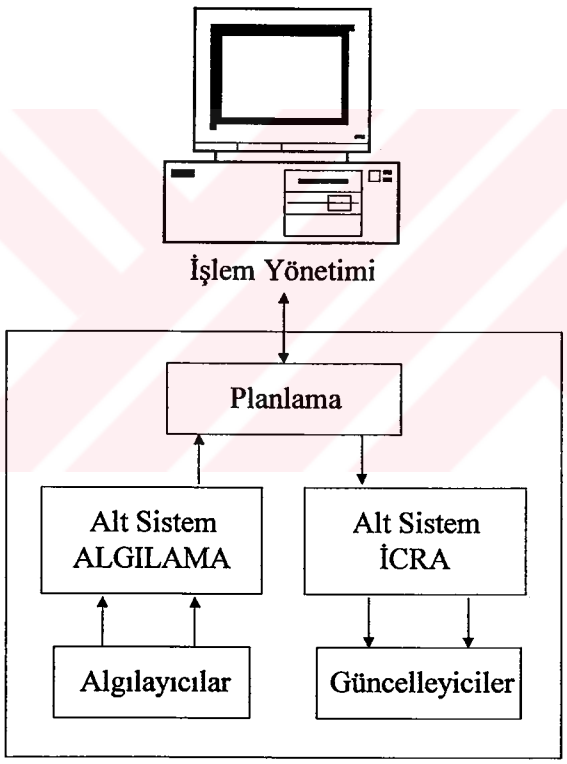
1960'larda mikroelektronik teknolojisinde çok önemli gelişmeler yaşandı. Bu gelişmeler sonucunda da bazı ülkelerde robot araçlara yönelik çalışmalar başladı. İnsanlar için tehlikeli, zahmetli ve sıkıcı işlerde kullanılmak amacıyla ilk robotlar yapıldı. Çoğu yalnızca bir robot kolu ve el yerine takılmış bir aletten oluşmaktaydı. Gerçekte, günümüz robotlarının da büyük bir bölümü aynı biçimdedir. Her geçen yıl gelişen teknolojiyle birlikte robotların yetenekleri de gelişmektedir. Artık basit robot kollarından, bilgisayar denetimli karmaşık aygıtlara değin, çok değişik boyut ve yapıda robotlar vardır. Robotlar bazı işlerde insanlardan daha başarılıdır. Robotlar en çok fabrikalarda kullanılmaktadır, ama bir yandan da laboratuarlar, enerji santralleri ve hastanelerde de robot kullanımı giderek yaygınlaşmaktadır. Ayrıca her geçen gün robotlar için yeni kullanım alanları ortaya çıkmaktadır.

Robotik sistemlerin bir özelliği, zeki davranışlar gösterebilmeleridir. Sistemde gömülü (embedded) bilgi işlem elemanlarının ortaya çıkardıkları zekanın spektrumu, programlanmış davranış; öz-tanıma, öğrenme ve öz-organizasyona kadar değişen bir alanı kapsayabilir. Robotik bir sistem, bir denetim sistemi ve bir denetlenen sistemden oluşmaktadır. Denetim sistemi; (Şekil 1.1) planlama, algılama ve icra olmak üzere en az üç işlevsel alt sistemi içerir.

Planlama, sisteme verilen görevlere, kullanıcının isteklerine ve kavrama sistemince elde edilen çevre durum bilgisine dayalı olarak yapılması gereken eylemlerin planlanması ile ilgilenir. Klasik mikroişlemciler, yapay sinir ağları veya bulanık işlemciler bilgi işleyebilecek, öğrenebilecek ve karar alabilecek kabiliyete sahip elemanlara örnek teşkil ederler.

Algılama, gerekli bilgilerin toplanması ve sistemin bulunduğu çevrenin bir iç modelinin oluşturulması ile ilgilenir. Bu alt sistemin ana bileşenleri, bilgi depolama ve bilgi işleme elemanları ve haberleşme alt birimleridir.

İcra, planlama alt sistemince karar verilen eylemlerin uygulanmasından ve bu alt sisteme sonuç hakkında bilgi verilmesinden sorumludur. Güncelleyiciler, manipülatörler ve denetimciler icra alt sisteminin ana bileşenleridir (Sunay, 1998).



Şekil 1.1 Bir robotik sistemin yapısı

Gezgin robot; verilen bir yön ve konum boyunca hareket edebilme yeteneği olarak tanımlanabilen gezinmesine göre, otomatik olarak hareket edebilen, serbest programlanabilir veya kendi başına hareket edebilen özerk bir araçtır. Endüstriyel robotlar, sabit bir tabana monte edilir. Çalışma alanları robot eklemlerinin ulaşabileceği çevreyle sınırlıdır. Bu nedenle, bu tip robotlarda yapacakları işin kendilerine getirilmesi gerekmektedir. Gezgin bir robot ise, belirli bir görevi yerine getirebilmek amacıyla işin bulunduğu alana gidebilmektedir (Aksun, 1996).

Gezinme konusunda endüstriyel uygulamalardaki ilk adım, 1975 yılında Olivetti tarafından sunulan kızaklı robot tipidir. Bundan sonraki gelişme, fabrikaların büyük bir bölümüne ulaşabilen tekerlekli araçlardır. Uzaktan kumandalı olarak bir operatör tarafından dolaylı biçimde yönlendirilen (Teleoperasyon) ve hareketi kontrol edilen robotlardır. Teleoperasyon işlemi ile çalışan bu robotlar tehlikeli bölgelerde çok iyi bir performans sağlamışlardır.

Teleoperasyonla çalışan robotlardan sonra kendiliğinden yönlendirilmiş araçlara (AGV) geçilmiştir. Bu araçlar, fabrikanın bir köşesinden diğer bir köşesine yükleri götürebilmekte ve genellikle sabit bir yol izlemekle sınırlıdırlar. İzleyeceği yörünge, aracın izlemek üzere programlandığı yere gömülü bir tel veya yere çekilen boyalı bir hattır. Kendiliğinden yönlendirilmiş araçlar, gömülü teli izlemek üzere radyo frekanslı algılayıcı veya boyalı hattı izlemek üzere optik algılayıcılar kullanırlar.

Kendiliğinden yönlendirilmiş araçlardan sonraki en arzulanan aşama, tamamiyle özerk robotlardır. Bu aşamada, belirli bir görevi insan müdahalesi gerektirmeden yerine getirmek amacıyla tasarlanmış tekerlekli, paletli veya bacaklı hareket sistemlerine sahip özerk robotlar yer almaktadır. Özerk gezgin robotlar, robotik sistemlerdeki gelişmenin bir sonucu olarak ortaya çıkmıştır. Burada özerklik, robotun değişken durumlara karşı kendi kendine karar verebilme ve bu kararları uygulayabilme yeteneğidir. Özerk bir gezgin robotun, dinamik değişkenlik gösteren bir ortamda gerekli kararları verip, uygulayabilmesi için algılayıcıları vasıtasıyla çevresinden bilgi toplaması gerekmektedir.

Gezgin robotlar konusundaki ilk çalışmalar, 1970'lerin başlarına rastlamaktadır. Bu konudaki ilk çalışma, Standford Üniversitesinde gerçekleştirilen SHAKEY gezgin robotudur. İlk çalışmalarda planlama ve problem çözme yetenekleri üzerinde durulmuştur. Algılayıcılar ve kontrol teknolojisi fazla gelişmediği için, robotun çevresini algılaması konusunda istenen sonuçların alınamaması, gezgin robotlar üzerinde yapılan çalışmaları zorlaştırmıştır. 1980'lerden itibaren, bilgisayarlardaki ve görme sistemlerindeki gelişmeler neticesinde endüstriyel robotlardaki çalışmalar yeniden canlanmakta ve günümüzde yüksek teknolojinin en önemli araştırma alanlarından biri olarak yer almaktadır.

- Gezgin robotlar; fabrika otomasyon projelerinde, sabit tabanlı robotlar veya makineler arasında malzemeleri ulaştırma amaçlı,
- Yangın, deprem veya nükleer santraller gibi tehlikeli ortamlarda kurtarma, tetkik, bakım, vb.., görevleri yerine getirme amaçlı,
- Sualtı ve uzay çalışmalarında, geniş bir alanda, çevre hakkında veri toplama amaçlı,

- Hastanelerde hastalara ilâç, yemek ve mektupların dağıtımı gibi görevlerde veya özürlü insanlara yardım amaçlı,
- Askeri alanda mayın temizleme, döşeme, cephane ve diğer gereçlerin savaş alanı birimlerine iletimi, vb.., amaçlı,
- Evlerde temizlik, yemek servisi ve günlük işlere yardım amaçlı olarak kullanılabilirler.

Gezgin robotun hareket sistemi, robotun gezinmesine imkan veren alt sistemdir. Kara, deniz hava veya uzay gibi değişik ortamlarda hareket etmek üzere tasarlanmış gezgin robot sistemleri vardır. Yer yüzeyinde hareket eden gezgin robotlar hareket sistemlerine göre;

1. Tekerlekli
2. Paletli
3. Bacaklı
4. Eklemsi yapılı
5. Temel hareket sistemlerinin kombinasyonu şeklinde sınıflandırılabilirler.

Bir uygulamada kullanılacak gezgin robotun seçimi; güç ihtiyacına, maliyete, istenen hareket karakteristiklerine ve robotun hareket etmesi gereken yüzeylere göre yapılmalıdır. Ancak pratikte seçim genellikle tekerlekli robotlar olmaktadır. Tekerlekler, düz veya sınırlı bir eğime sahip, az pürüzlü yüzeylerde iyi performans ve çekme kabiliyetine sahiptirler. Bu sebeple de, en fazla kapalı alanlarda tercih edilmektedirler. Aksi takdirde özellikle engebeli arazilerde ve açık alanlarda paletli robotlar tercih edilmektedir. Bacaklı robotlar, birçok arazi tipinde çalışabilecek ve merdiven tırmanabilecek yeteneğe sahiptirler. Fakat tasarımları, bacak kontrolleri, denge için gereken hesaplamalar oldukça uzun ve karmaşıktır. Çektikleri güç çok büyük olduğundan deneysel modeller üniversitelerde veya AR-GE laboratuarlarında görülmektedir. Eklemsi yapılı gezgin robotlar ise, düzgün olmayan arazi ve dar çalışma alanlarında uzun ve eklemli gövdelerini yerin topografisine adapte edebilmektedirler. Ayrılabilir bir yapıya sahip olmaları nedeniyle de, arıza halinde olan parçaları sökülerek yerlerine sağlam parçalar monte edilebilmektedir. Temel hareket sistemleri olan tekerlek, palet ve bacak mekanizmalarının yetersiz kaldığı koşullarda ise tekerlek-bacak, tekerlek-palet gibi kombine hareket sistemlerinden faydalanılabilmektedir.

Tezimizde, tekerlekli yapıya sahip gezgin robot mekanikleri incelenerek, yazılım için denetim kartına gönderilecek komutlar oluşturulacaktır. Görsel kullanıcı arabirimi ve kullanıcının bu robotları kontrol etmesi için gerekli yazılım tasarlanıp, gerçekleştirilecektir. Fakat daha önce literatürde yapılan çalışmalar incelenecektir.

1.1 Robot Kol Programlama Dilleri

Robot kol programlama dilleri üç başlıca üç ana gruba ayrılabilir (Craig, 1989);

1. Robot kola özel diller: Bu tür programlama dilleri, tamamen yeni bir dil geliştirilerek oluşturulmuştur. Robota özel alanlara hitap ederken, genel bir bilgisayar programlama dili olarak düşünülebilir.

2. Var olan bir dile robot kütüphanesi ekleyerek yaratılmış diller: Bu tür programlama dilleri, popüler bir programlama dili (Pascal) ile başlayarak ve robota özel alt rutin kütüphanesi ekleyerek geliştirilmiştir. Kullanıcı, daha önceden tanımlanmış robota özel alt rutin paketini sık sık çağırarak bir Pascal programı yazar.

3. Yeni bir genel amaçlı dile robot kütüphanesi ekleyerek yaratılmış diller: Bu tür programlama dilleri, önce programlama tabanı olarak genel amaçlı bir dil yaratıp, daha önceden tanımlanmış robota özel alt rutin kütüphanesi ekleyerek geliştirilirler.

Robot kontrolü için geliştirilmiş bazı yeni programlama dilleri: AL, AML, MHI, Ada.

Değişiklik yapılarak oluşturulanlar: WAVE, VAL, PAL, RAIL, JARS, RPL ve MCL, HELP.

1.1.1 AL

Stanford Üniversitesi Yapay Zeka Laboratuarında geliştirilmiş bir robot programlama dilidir. Pascal temellidir. Çok kollu robotun birleşik hareketinin kontrolü için yapılar oluşturur. Endüstriyel kollar AL sistemine entegre edilmiştir. Uygulama büyük bir anaçatı bilgisayara (mainframe) ihtiyaç duyar. Neredeyse tamamı OMSI Pascal'da yazılmıştır. Sistemde iki bilgisayar vardır. AL sisteminde programlar, denetçi bilgisayar üzerinde geliştirilir ve derlenir. Sonuçta elde edilen p-kodu, diğer bilgisayara yüklenir. Bu bilgisayar da robotun kollarındaki her eklemi direkt olarak kontrol eder ve komutları mikroişlemciye gönderir. Yüksek seviyeli kod da SAIL (Stanford Artificial Intelligence Language) ile yazılmıştır. Denetçi bilgisayar kayan noktalı işlemciye sahiptir. Tampon belleği yoktur ve tek bir terminali ve 128 KB RAM belleği vardır.

Skaler sayılar ve VECTOR, ROTATION, FRAME, TRANS, STRING, EVENT veri tipleri için özellikleri vardır. Bir VECTOR, vektörün x, y ve z doğrultularındaki üç bileşenine özel üç gerçek sayıdan oluşur. ROTATION, FRAME ve TRANS veri tipleri homojen matris temellidir. ROT, bir yön vektörü ve dönme miktarını belirleyen bir açıdan oluşur. FRAME, bir yön vektörü ve konumu tanımlayan bir yönlendiriciden oluşur. TRANS, bir koordinat

sisteminin diğerine göre konumunu tanımlar. Bütün bu veri tipleri dizi formunda kullanılabilir.

AL, objeler arasındaki ilişkiyi, ekleme (affixment) mekanizmasını kullanarak gösterebilir. Bir objeyi temsil eden bir FRAME, başka bir FRAME'e takılabilir. Böylece nesneler arasındaki ilişki AL tarafından otomatik olarak sağlanır. AL; algılayıcı girişi, durum görüntülenmesi ve giriş/çıkış dönüşümü için kontrol seti içermektedir.

AL programları yaratabilmek ve ifade temelli kontrol yapabilmek için etkileşimli bir AL sistemi ya da kaynak kodu çeviricisine (POINTY) sahiptir. POINTY her AL komutunu çalıştırabilir. Kendine ait etkileşimli işlemlerde yararlı olabilecek pekçok ek komutu vardır.

1.1.2 AML

IBM tarafından tasarlanmış bir programlama dilidir. Bir çevirici, taban dili uygular ve basit işlemleri tanımlar. AML, RS/1 robotunu kontrol etmek için kullanılır. Çevirici, komutların analiz edilmesine ve satır satır uygulanmasına izin verir. Depolama yönetimi, dosyaların ve belleğin otomatik kontrolünü sağlar. Veri parçası programları, statik değişkenleri, yığını ve arabirim tamponlarını (buffer) saklar. Yığın, programın çalışmasını kontrol eder. Değerler yığının en tepesine konur ve sırayla alınır. Arabirim tamponları da, çevre birimleriyle ve robot kol kontrolörleri ile haberleşmek için veri saklar. Statik değişkenler, her program koşturulduğunda atanır ve saklanır. İş İstasyonu arabirimi; algılayıcı, giriş/çıkış görüntüleme, hareket planlama, aygıt arabirim çıkışı ve güvenlik denetimini sağlar.

1.1.3 MHI

Programlanabilir komutlar ile işlemleri tanımlayabilme kapasitesine sahip ilk robot dilidir. Massachusetts Institute of Technolgy tarafından (1960-1961) geliştirilmiştir. Bu dil MHI robotunu kontrol etmek için geliştirilmiştir. MHI'nın birkaç ikili (binary) algılayıcısı olduğundan dil, algılayıcı işlemler ile kontrol edilen hareketler için yazılmıştır. Mevcut bazı komutlar;

MOVE: Yön ve hız belirtir.
UNTIL: Belirli bir algılayıcı durumu oluşana kadar işler.
IFGOTO: Belirli bir durumda programda yeni bir yere dallan.
IFCONT: Belirli bir durumda harekete devama dallan.

Bu komutlar, bir nesneye uzanma, onu tutma, başka bir bölgeye hareket ettirme ve nesneyi istenilen yere bırakma hareketlerini içeren birkaç işlemi gerçekleştirmek için yeterlidir.

Bununla birlikte dil, aritmetik ve kontrol komutları için yeterli değildir.

1.1.4 Ada

1978'de Amerikan Savunma Bakanlığı tarafından geliştirilmiştir. Gerçek zamanlı yerleşik sistemlerin programlanması için taban sistem uygulama dilidir. Ada'nın avantajı, sağlanan güçlü veri yapıları ve veri işlemlerinden kontrol işlemlerini ayırabilme kabiliyetidir.

1.1.5 WAVE

Standford Üniversitesinde 1970-1975'te geliştirilmiş ilk genel amaçlı robot programlama sistemidir. Robotun kontrolü için geliştirilen algoritmalar kompleks ve işlemsel olarak kuvvetlidir. Gerçek zamanda anlık olarak koşturulamazlar. WAVE, birkaç önemli özelliğe sahiptir. Bunlar;

- Kartezyen koordinatlara uyma kapasitesi özelliği,
- Hız ve ivme devamlılığını sağlamak için eklem hareketlerinin koordinasyonu,
- Kartezyen koordinatlarda dokunma algılayıcı konumlarının tanımı,
- Korumalı hareket kapasitesi (Böylece algılayıcıdan gelen bilgi ile bir hareketi bitirmek için kullanılabilir.)

1.1.6 VAL

Victor Sheinman tarafından, Unimation Anonim Şirketi için Unimation robotlar ile kullanılmak üzere tasarlanmış ve geliştirilmiş programlama dili ve kontrol sistemidir. Tasarım amacı, robot görevlerini kolaylıkla tanımlayabilme kabiliyetini sağlamaktır. VAL, robot programlamak için eklenmiş birçok yeni komut kelimeleri ile BASIC yapısına sahiptir. Kendi işletim sistemi vardır. Bu sistem kullanıcı arabirimi, editör ve dosya yöneticisine sahiptir. Sistem; kullanıcı terminal, floppy disk, öğrenme kutusu, ayrık bir I/O modülü ve bir görüntü sistemi ile haberleşir. VAL, C dili ve 6502 ASSEMBLY dili kullanarak uygulanmıştır. Yazılım, PUMA robotları, Unimate 2000 ve 4000 robot serileri ile kullanılabilir. VAL; hız kontrolü, kavrama, kol hareketi ve sinyalleşmeyi basit ve kolay anlaşılır bir çerçevede sağlar. Basma düğmesine ve kolu hareket ettirmek için denetim kolu (joystick) kontrolüne sahiptir. Yazılım, gerçek sayıları veya karakter dizilerini desteklememektedir.

1.1.7 RAIL

RAIL Automatix, Inc.of Bilerica, Massachusetts tarafından, görüntü ve manevra kontrolü için yüksek seviyeli bir dil olarak geliştirilmiştir. Pascal temelli bir çeviricidir. Denetlemeyi desteklemek ve endüstride kaynak yapmak için birçok yapı katılmıştır. RAIL sisteminin merkezi işlemcisi Motorola 68000'dür. Çevre birimleri bir terminal ve bir öğrenme kutusu içerir. RAIL, üç değişik sistemle desteklenmiştir; görüntü, montaj işlemleri için geleneksel tasarımlı kartezyen kol ve kaynak yapmak için Hitachi işlemcili robot. Geniş bir Pascal komut kümesi içerir ve birkaç değişik veri tipini kullanabilir. Denetim, kaynak yapma ve görüntü işlemleri, dil kapasitesi ile desteklenebilir.

1.1.8 JARS

1979'da NASA'nın Jet Tahrik Laboratuarında, Robotik ve Teleoperatör Grubu tarafından geliştirilmiştir. Pascal temellidir. Robot kontrolü için birçok tipler, değişkenler ve alt rutinler eklenmiştir. Yüksek seviyeli bir programlama kapasitesi sağlar. Kayan nokta işlemler ve 28 KB bellek kullanarak çalışır. Standford Scheinman kolu ve PUMA 600 robotunun kontrolü için kullanılmaktadır.

1.1.9 MCL

MCL, 1978'de Amerikan Hava Kuvvetleri Projesi için Cincinnati Milicron ile Mc Donell-Douglas tarafından geliştirilmiştir. CAM'de kullanılan nümerik kontrollü makineler için geliştirilmiş kontrol dilidir.

1.1.10 HELP

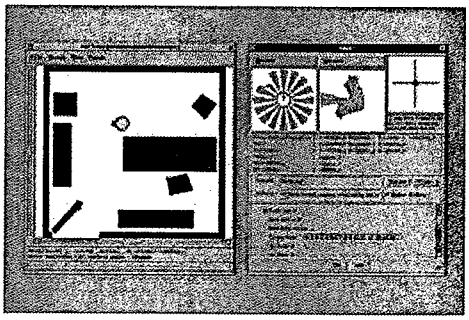
General Electric Şirketi tarafından, kendi robot ürünleri için geliştirilmiştir. Pascal temellidir. İki robot kolunu aynı anda kontrol edebilmek için işlemleri desteklemektedir (Critchlow, 1985; Klafter vd., 1989).

1.2 Gezgin Robot Programlama Dilleri

1.2.1 XRDev

XTREME ROBOTICS Geliştirme Sistemi (XRDev) (Şekil 1.2), Nomad ailesi robotları için tasarlanmış nesne tabanlı gezgin robot geliştirme paketidir. Sistem, simülasyon veya gerçek robot için grafik kullanıcı arabirimine sahiptir. Direkt veya uzaktan erişim yapılabilir. Gezgin

yol bulma ve algılama algoritmalarına sahiptir. Tüm algılayıcı verinin zaman/konum uyarlaması yapılabilir. C programlama dilinde komut seti arabirimi vardır.



Şekil 1.2 XRDev yazılımı

XRDev sistemi dört bölümden oluşur: Uygulama Programı Arabirimi (API), Olay Programlayıcı, bir robot simülatörü ve Görsel Kullanıcı Arabirimi (GUI).

Uygulama Programı Arabirimi (API): Sistem içinde her modüle erişimi sağlar. API, kullanıcının birden fazla robotu eşzamanlı olarak kontrol etmesine imkan sağlar. Her robotun kontrolü C programlama dilinde temin edilmiş klasörden yapılır. Komut seti; algılayıcı kontrolü, algılayıcı bilgi alma ve motor kontrolü için gerekli verileri içerir. Algılayıcı kontrolü; sıra değiştirmeyi, hızlanmayı, sonar ve kızılötesi algılayıcıların ateşleme arası gecikmelerini sağlar. Algılayıcılardan bilgi alma komutları, programlayıcının algılayıcı verilerini istek üzerine düzenlemesine izin verir. İşlenmemiş algılayıcı verinin yanı sıra kullanıcı, motor konumları ve her algılayıcı verisinin kaydedildiği zaman hakkında bilgi isteyebilir. Bu veri genel olarak "poz yerleştirme" (pose stamping) olarak adlandırılır. Motor kontrol seçeneği kullanıcıya, gerçek ortamda Nomad'in konumunu belirleme ve motorları çeşitli durumlara ayarlama olanağı sağlar. Nomad'in konumu hakkındaki bilgi, kartezyen koordinat sisteminde bir değerler kümesine çevrilir. Kullanıcı tarafından ek robot komutları da eklenebilir. Komut işleme kullanıcı tercihine göre senkron veya asenkron olabilir. Bu; herbirinin tamamlanması beklenmeden, birden fazla dizi komutunun işlenebilmesini sağlar.

Programlayıcı: İki işlevi vardır. Birincisi, çevredeki her robot arasında merkezi haberleşmeyi sağlamaktır. Kullanıcı listeleyiciyi başlatır ve robotları buna bağlar. İkincisi de; bu robotların birbirleriyle olan etkileşimini senkronize etmek için zaman artırımını düzenlemektir. Kullanıcı gerçek zamandaki işlemler ile simülasyon arasında seçim yapabilir.

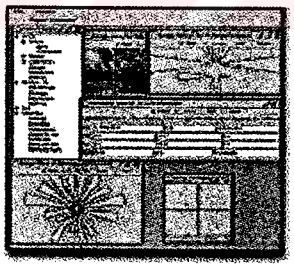
Simülatör: Her Nomad robotunun hareketini ve algılama kapasitesini modelleyebilir. Geliştirilmiş kullanıcı programlarını test etme imkanı sağlar. Simülasyon zaman-sürmelidir

yani, hesaplama olay tabanlı simülatörler gibi komutlar işlendiği zaman değil, sürekli olarak yapılır. Ses üstü, kızılötesi, dokunma ve lazer bilgilerini modelleyebilir. Yapı ve renk gibi malzeme karakteristiklerinin kullanıcı çevresinde tanımlanmasını sağlar. Böylece gerçek ortam malzeme özellikleri temsil edilebilir.

Görsel Kullanıcı Arabirimi (GUI): Robot çevresinin görsel bir perspektifini sağlar. Bu arabirim Linux, SGI, ve Sun İşletim Sistemleri ile çalıştırılabilen, X/Motif tabanlı bir sistemdir. Simüle edilmiş çevreyi ve içindeki robotları gösteren bir ortam penceresi gösterilir. Kullanıcı ortamda engeller yaratıp, yerleştirebilir ve her engelin karakteristiğini tanımlayabilir. Ortam penceresi ayrıca robot ve algılayıcı işaretlerini simüle edilmiş çevre ile ilişkili olarak gösterir. Çevredeki her robot, algılayıcı bilgilerini gösterebildiği gibi komutların doğrudan çalışmasına izin verebilen kendi penceresi ile temsil edilebilir. Bir komut listesi daha önce çalıştırılmış komutların kaydını tutar ve bir denetim kolu (joystick) robot konum kontrolünün uzaktan yapılmasını sağlar.

1.2.2 MOBILITY (Mobility Object Manager)

"MOM" (Şekil 1.3) ile (Mobility'nin Java-tabanlı grafik kullanıcı arabirimi) program koşturulurken, robot kontrol çevresi gözlemlenebilir, düzenlenebilir, ve derlenebilir. MOM Arabirimi; programı başlatır, nesne yaratır, nesne özelliklerini değiştirir, nesneleri birbirine bağlar ve düzenler ve hangi nesnelerin aktif olduğunu kontrol eder.



Şekil 1.3 MOBILITY yazılımı

Merkezi bir yönetim noktasından, robot hareketlendiricilerinin, algılayıcıların, algoritma çıkışlarının ve derleme bilgisinin görüntülenmesini sağlar.

1.2.3 Saphira

*Activ*Media ROBOTICS firmasının ürettiği bir yazılımdır. Pioneer 2 DX gezgin robotu için tasarlanmıştır. Sistem, Siemens C166 temelli mikrokontrolöre sahiptir. Ayrıca çok özellikli bir GUI'ye sahiptir. GUI, müşteri ve sunucu işlemlerinin ve durumlarının görsel ve metinsel

temsilini verir. Saphira ANSI C tabanlıdır. WindowsNT, Windows95, Windows98, UNIX, SunOS, Solaris, IRIX, OSF/1, FreeBSD, RedHat, Linux altında çalışabilmektedir.

Saphira yazılımı, çok seviyeli bir müşteri/sunucu (client/server) çevresinde işletilir. Robot sunucunun; sürüş motorları, konum kod çözücüleri ile tekerlekleri, uzaklık algılayıcıları ve bu kaynakları yönetebilmek için kontrolörleri vardır. Gerçek çevre algılaması ve akıllı gezgin robot aktiviteleri için yön bulma elemanlarını taşır. Robot sunucu, algılayıcılardan aldığı bilgileri işler. Fakat ona kılavuzluk edecek bir müşteri olmadan robot sunucu işlevsizdir. Müşteri robota akıllılık sağlar.

Saphira'nın alt seviyesi; (robot ile arayüzü) uygun bir metod ve robot sunucu ile haberleşme ve kontrol için robottan gerçek zamanlı işletim verisi alıp, robot kontrol etmek için komutlar göndererek protokoller sağlar. Saphira'nın orta seviyesi; yön bulma kontrolü ve algılayıcı yorumlaması için yüksek seviyeli fonksiyonları destekler. Yüksek seviyelerinde Saphira, robotun davranışlarını belirlemek için bulanık mantık tabanlı kontrol sağlar. Harita tabanlı yön bulma gibi sistemleri destekler. Kullanıcıya kendi istediği davranışları davranış derleyicisi ile tanımlayabilme olanağı sağlar. Bu derleyici basit bulanık kontrolleri C tabanlı koda dönüştürür.

Saphira gerçek veya sanal bir çevrenin modellemesini yapabilecek bir simülatöre sahiptir. Bu simülatör sonar algılayıcılar ve tekerleklerin kod çözücüleri için hata modellerine sahiptir. Tek adım modu vardır böylece yazılan programın her adımı detaylı olarak incelenebilir.

1.2.4 Ayllu

*Activ*Media ROBOTICS firmasının ürettiği bir yazılımdır. Ayllu, gezgin robot grupları için kontrol geliştiren bir araçtır. Dağılmış kontrol sistem elemanları ile görev programlama arasındaki haberleşmeyi sağlar. İşletim sistemleri arasında taşınabilir şekilde tasarlanmıştır. Windows, UNIX, Macintosh, QNX gibi platformlarda çalışabilmektedir. C tabanlı bir dildir. Ayllu'nun başlıca görevleri, gürültülü ve uzaklık limitli haberleşme, hızla değişen gerçek ortam ve değişik donanım hataları ile başa çıkmak zorunda olan dayanıklı çoklu robot sistemlerinin yürütülmesidir. Pioneer gezgin robotları için tasarlanmıştır. Bu robotlar için temel motor kontrol ve algılayıcı yorumlaması için bileşenleri vardır. Yüksek seviyeli planlama ve görüntü işleme gibi işlem yoğunluğu olan görevlerin koordinasyonunu sağlar.

Araştırma sonucunda, incelenen gezgin robot programlama dilleri karşılaştırılmış ve Çizelge 1.1 oluşturulmuştur.

Çizelge 1.1 Gezgin robot programlama dilleri karşılaştırması

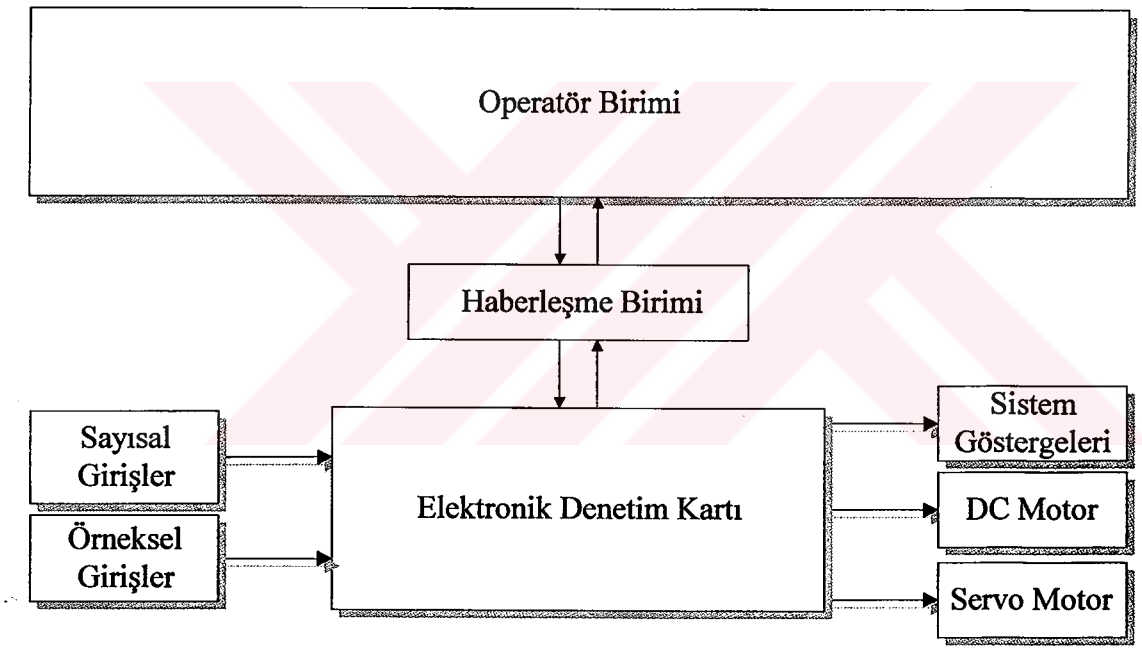
Özellikler	**XRDev**	**MOBILITY**	**Saphira**	**Ayllu**
Simülatör	√	X	√	X
Görsel Kullanıcı Arabirimi	√	√	√	√
Algılayıcı Kontrolü	√	√	√	√
Çoklu Robot Kontrolü	√	X	X	√
Programlama Dili	C	-	ANSI C	C

2. GEZGİN ROBOT YAZILIMI

2.1 Genel Tanım

Tez kapsamında gerçekleştirilmesi düşünülen gezgin robot bir "teleoperated" (uzaktan işletilen) robottur. Bu robot, operatör tarafından uzaktan gönderilen komutları yerine getirir. Bu amaçla, gezgin robot sistemi (Şekil 2.1) dört ana bölümden oluşmaktadır:

1. Gezgin taban
2. Operatör (komuta) birimi
3. Gezgin taban işletim ve denetim birimi
4. Haberleşme birimi



Şekil 2.1 Gezgin robot sistemi blok diyagramı

Gezgin taban ve mekanik aksamları, robotun mekaniğini oluşturmaktadır. Hareketi sağlayan birim, sürüş mekanizması, güç ünitesi, haberleşme ünitesi ve denetim birimini taşıyabilecek kadar güçlü olmalıdır. Gezgin taban, manevra kabiliyeti ve hız bakımından çalışma ortamına uygunluk göstermelidir.

Operatör birimi, mimari hiyerarşinin en üstünde yer almaktadır. Robotun kumandası bu birimden yapılır. Operatör, robotun gerçekleştirmesini istediği komutları bu birimden girer. Birimin donanım ve yazılımı, kullanıcı dostu bir yapıya sahip olmalıdır. Görsel bir arayüze sahip olmalı ve karmaşık olmayan, anlaşılır komutlarla çalıştırılabilmelidir.

Gezgin platform işletim ve denetim birimi, robot mekaniğinin operatör tarafından istenen komutları gerçekleştirmesini sağlar. Operatör birimi ve mekanik sistem arasında bir arayüz oluşturur ve komutların doğru bir şekilde gerçekleştirilmesinden sorumludur. Operatörün, robotun durumundan haberdar olmasını da bu birim sağlar. Operatöre iletilmesi gereken verileri, haberleşme birimi yardımıyla operatöre gönderir.

Haberleşme birimi, komuta birimi ile denetim birimi arasında bilgi alışverişini sağlar. Robotun çalışacağı ortama ve komuta biriminin robottan uzaklığına uygun sistemler seçilmelidir.

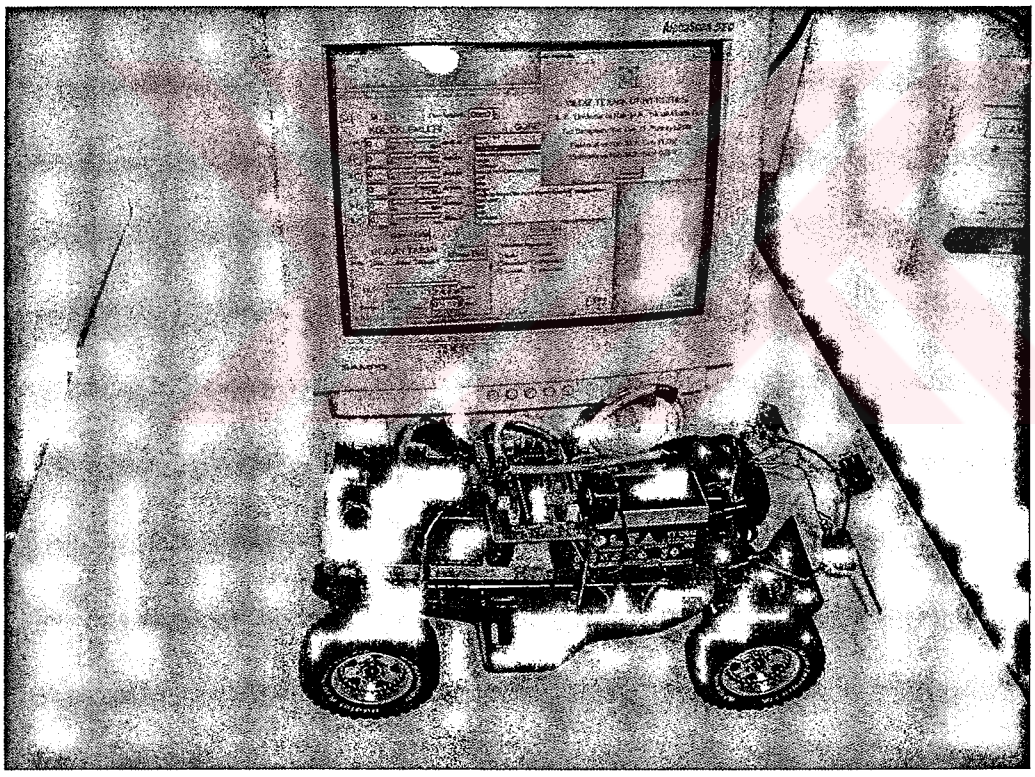
Gezgin robotun, komutları doğru bir şekilde yerine getirmesinin gereklerinden biri de çevre koşullarını algılayabilmesidir. Örneğin; ileri yol alması istenilen bir robot, gerekli algılayıcıları yoksa, ilerlerken bir engelle karşılaştığında, verilen komutu yerine getirmek için çalışmasına devam edecek ancak komutu hiçbir zaman yerine getiremeyecektir. Ancak, uygun algılayıcılara sahip olduğu takdirde çalışmayı durdurabilecek ve durumundan operatörü haberdar edebilecektir. Bu durumda robot yalnız "teleoperated" değil, aynı zamanda kılavuzlu "supervised" olacaktır (Everett, 1995).

Tez kapsamında tasarlanan robot yukarıda açıklandığı şekilde "supervised-teleoperated" bir yapıya sahiptir. Komuta birimi olarak IBM uyumlu bir kişisel bilgisayar (PC) seçilmiştir. Bir komuta birimi olarak PC, hem donanım hem de yazılım açısından çok geniş olanaklar sunmaktadır. Görsel bir birimdir ve kullanıcılar için de bilindik ve kullanımı kolay bir arayüz oluşturmaktadır. Mekanik bir sistemin idaresi için yeterli hıza ve uygulama geliştirmek için esnek bir yapıya sahiptir. Geliştirilen yazılım sayesinde gezgin robot, daha önceden yazılmış bir görev listesini veya kullanıcı tarafından anında girilen komutları işleyebilmektedir. Görev listesindeki görevler bir kereye mahsus yerine getirilebildiği gibi, çevrimsel bir şekilde tekrarlanan bir rutin gibi de yerine getirilebilmektedir. Ayrıca çevresel değişikliklerin varlığı da göz önüne alınarak programlama yapılabilmekte, algılayıcılardan alınan bilgiler robotun hareketi için koşul olarak kullanılabilmektedir. Bu durumda robot, sınırlı hareket bölgesi tahmin edilebilir engeller göz önünde tutularak programlanabilmektedir. Örneğin kapılar bu şekilde engeller oluştururlar. Dolayısıyla, gezgin robot programlanırken kapının açık veya kapalı olması durumu öngörülmek zorunda değildir. Her iki koşulda da robotun nasıl hareket edeceği önceden belirlenebilir. PC, kullanıcı tarafından kapalı bir şekilde girilen komutları gezgin platform işletim ve denetim biriminin işleyebileceği verilere dönüştürerek bunları denetim kartına gönderir.

Robot, aynı zamanda AGV Otomat Yer Taşıtı (Automated Ground Vehicles) sınıfına da girmektedir. Çünkü önceden programlandığı görevi, görevin yerine getirilmesi esnasında herhangi bir operatör müdahalesine gerek duymadan bir otomat gibi yerine getirebilmektedir. Yani bir tipik teleoperated robot gibi her an operatörün denetimi altında değildir. Programlandıktan sonra operatöre ancak beklenmedik durumlarla karşılaşıldığında ihtiyaç duymaktadır.

2.2 Tasarladığımız Gezgin Robot Sistemi

Çalışmamızda gezgin robot, bir gezgin mekanik yapı, operatör komuta birimi olarak PC, gezgin platform işletim ve denetim birimi olarak mikrodenetleyicili bir arayüz kartı ve motor ve algılayıcı sürücülerinden oluşmaktadır.



Şekil 2.2 Tasarlanarak gerçekleştirilen gezgin robot sistemi

Arayüz kartı, PC'den aldığı verileri mekanik sistem için gerekli elektriksel işaretlere dönüştürür. PC'den komutlar iki ayrı biçimde gelmektedir. Denetim kartı, komut biçimini ve daha sonra da komutları ayırt ederek, hareket organlarını komuta uygun şekilde hareket ettirir. Yapılan çalışmada, diferansiyel sürüşlü ve dümen mekanizmalı olmak üzere iki tür gezgin platform incelenmiş ve kontrolü gerçekleştirilmiştir.

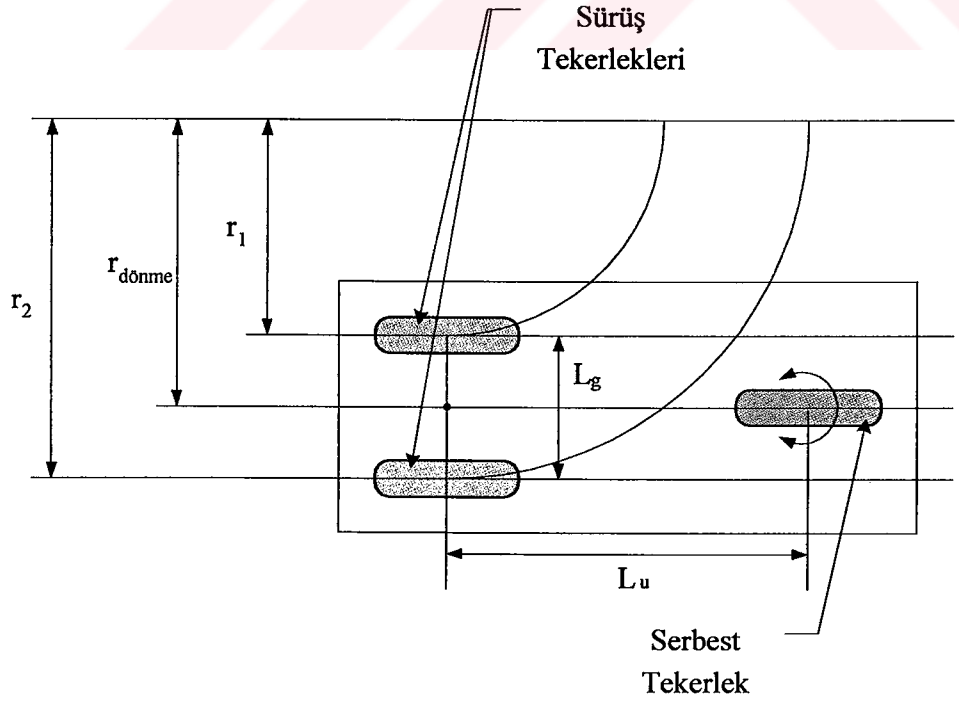
Diferansiyel sürüşlü robotta, sürüş ve dümen mekanizması için iki DC motor; dümen mekanizmalı robotta, sürüş için bir DC motor ve dümen mekanizması için de bir servo motor kullanılmaktadır. Kart, komutları çözümleyerek DC motoru istenen hız ve yönde hareket ettirirken, servo motoru da uygun konuma getirir.

2.2.1 Uygulamada kullanılan gezgin robot mekanikleri

2.2.1.1 Diferansiyel sürüşlü gezgin robot mekaniği

Yapılan gezgin robot çalışmalarına ilk olarak diferansiyel sürüşlü bir gezgin robot mekaniği üzerinde başlanmıştır. Bu robot üzerindeki çalışmalarda PC ve denetim kartı arasındaki haberleşme sistemi oluşturulmuş, komut dizgesi ve sistemin komutları işleme mantığı belirlenmiştir.

Bu robot iki sürüş tekerleğine ve dengeyi sağlamak üzere yerleştirilmiş üçüncü bir serbest tekerleğe sahiptir. İki tekerleğin ayrı ayrı sürülmesi nedeniyle, robot tekerlek ekseninin orta noktası üzerinde 0 yarıçapla dönme kabiliyetine sahiptir. Bu özellik, robota dar alanlarda hareket serbestliği vermektedir. Üçüncü tekerleğin, gerek düz ilerlerken gerekse manevra yaparken robotun hareketine engel olmayacak bir yapısı vardır. Tek bir mille robotun alt tabanına sabitlenmiştir ve itiş yönünde doğrultu alacak serbestliğe sahiptir (Şekil 2.3).



Şekil 2.3 Diferansiyel sürüşlü robotta $\theta_{dönme}$ kadar yönlenme için sağ ve sol tekerleklerin çizdiği yaylar

Diferansiyel sürüşlü robotta hareket, özdeş olan sürüş motorlarının hızlarının ve dönüş yönlerinin kontrol edilmesi ile sağlanır. Bu nedenle DC motorların her iki yönde ve farklı hızlarda çalışmasına izin verecek şekilde sürülmeleri gerekmektedir. Yine aynı nedenle, operatörün verdiği komutların yalnız motor ismi, hızı ve dönüş yönü bilgilerini taşıması yeterlidir. Robot hareketi için gerekli kinematik denklemler aşağıda verilmiştir.

Her iki tekerlek ayrı ayrı sürüş motorlarına bağlanmıştır. Bu nedenle iki tekerlek farklı yön ve hızlarda hareket edebilme kabiliyetine sahiptir. Sol tekerleği 1, sağ tekerleği ise 2 ile numaralandırırsak; sol tekerleğin $\theta_{dönme}$ kadar yönlenme için çizdiği yay uzunluğu, tekerleğin çizgisel hızı V_1 (m/sn) ve hareket süresi t (sn) cinsinden belirlenebilir:

$$V_1 \cdot t = \pi \cdot r_1 \cdot \frac{\theta_{dönme}}{180} \tag{2.1}$$

Sağ tekerleğin $\theta_{dönme}$ kadar yönlenme için çizdiği yay uzunluğu da benzer şekilde yazılabilir:

$$V_2 \cdot t = \pi \cdot r_2 \cdot \frac{\theta_{dönme}}{180} \tag{2.2}$$

Robotun mekanik yapısı nedeniyle bu yarıçaplar arasında aşağıdaki gibi bir ilişki vardır:

$$r_1 = r_{dönme} - \frac{L_g}{2} \tag{2.3}$$

$$r_2 = r_{dönme} + \frac{L_g}{2} \tag{2.4}$$

Bu denklemlerden, iki tekerlek hızı arasındaki ilişkiyi formüle edersek; denklem aşağıdaki gibi olacaktır.

$$\frac{V_1}{V_2} = \frac{r_1}{r_2} = \frac{r_{dönme} - \frac{L_g}{2}}{r_{dönme} + \frac{L_g}{2}} \tag{2.5}$$

Bu denklemden yola çıkarak robotun hareketi için motorların hız ve yönlerinin ne olması gerektiğini söyleyebiliriz. Tasarladığımız robotun, yalnızca tekerlek ekseninin orta noktası üzerinde dönmesi planlanmıştır. Bu nedenle, dönüş esnasında her iki sürüş motoru da hareket halinde olacak ve aynı hızda, fakat ters yönde hareket edeceklerdir. İleri ve geri harekette de aynı hızda olacaklarından, komutta tek bir hız değerinin yer almasının yeterli olacağı görülür. Bu, komut uzunluğunu etkileyen önemli bir faktördür. Bu durumda oluşturulan komut

dizgesinde gezgin robot komutlarının uzunlukları bir bayttır.

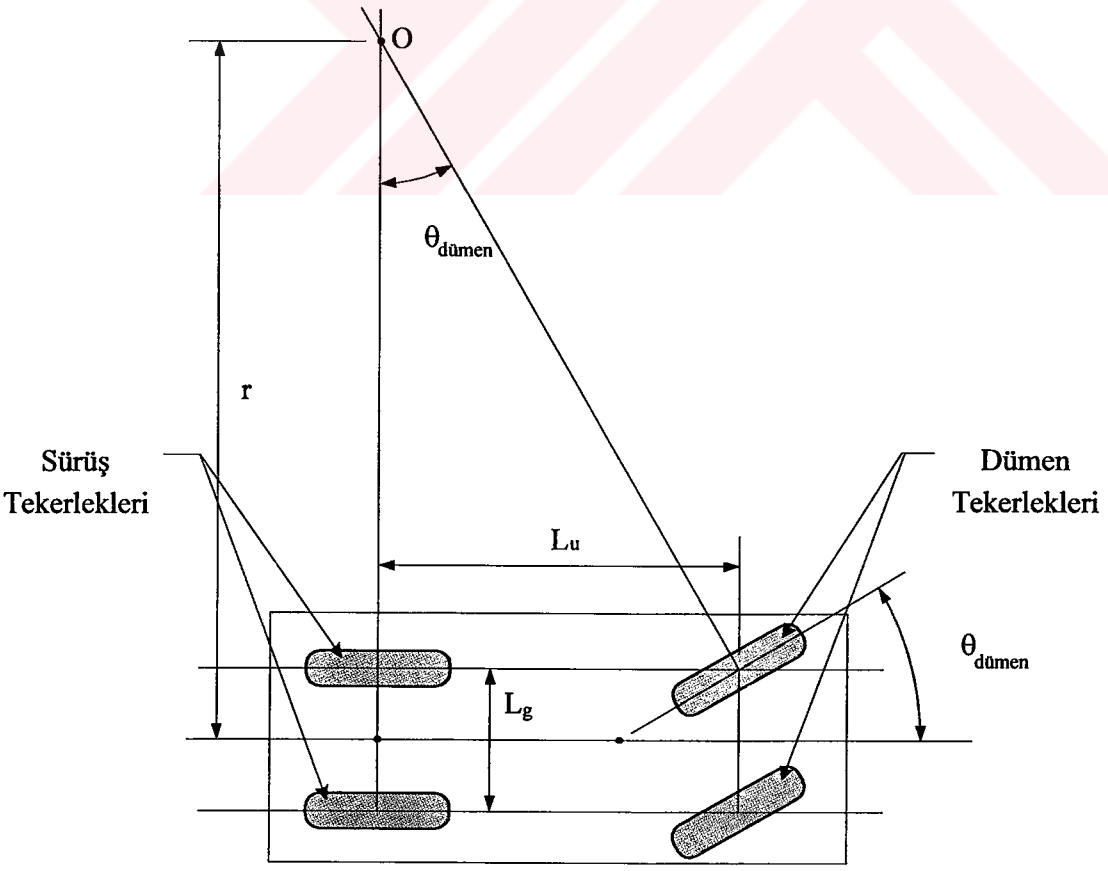
$$r_{dönme} = 0 \text{ için} \qquad \frac{V_1}{V_2} = -1 \qquad V_1 = -V_2 \tag{2.6}$$

2.2.1.2 Dümen mekanizmalı gezgin robot mekaniği

Çalışmalara dümen mekanizmalı bir gezgin robot mekaniği üzerinde devam edilmiştir. Komut dizgesi ve sistemin komutları işleme mantığı diferansiyel robottan farklı olarak belirlenmiştir.

Dümen mekanizmalı robotta hareket, sürüş motorunun hız ve dönüş yönü ile dümen mekanizmasını hareket ettiren servo motor açısının kontrol edilmesi ile sağlanır. Bu nedenle DC ve servo motorların ayrı ayrı sürülmeleri gerekmektedir. Aynı nedenle, operatörün verdiği komutların; DC motor hızı ve yönü, dönüşün olup olmadığı, dönüş varsa servo açısı bilgilerini taşıması gerekmektedir. Dönüş yönü, servo açı bilgisinin içine gömülmüştür.

Robot, mekanik yapısı nedeniyle diferansiyel sürüşlü modelde olduğu gibi tekerlek ekseninin orta noktası üzerinde dönemez. Belli bir "*r*" yarıçapıyla dönmesi ve tekerleklerin bir yay çizmesi gerekmektedir (Şekil 2.4) (Rajagopalan, 1997).



Şekil 2.4 Dümen mekanizmalı robot için $\theta_{dönme}$ kadar yönlenme için dönüş yarıçapı

İki tekerlek birbirinden farklı uzunluklarda yaylar çizmelidir. Ancak, sürüş ve dümen tekerlekleri aksları üzerinde birer diferansiyel dişli sistemi taşımaktadır ki, bu şekilde aynı aksa bağlı tekerlekler birbirinden farklı hız ve yönlerde hareket etme serbestliğine sahip olurlar. Bu nedenle hesapların yalnız bir tekerlek için yapılmasında bir sakınca yoktur.

Robot, PC'den komutları sıra ile almakta ve bir sonraki komutu alıncaya kadar eski komutu işlemektedir. Bu nedenle komutların gönderilmeleri arasında geçen süre uygun mesafe ve yönlenmenin sağlanabilmesi için önemlidir. Dönüş komutları için bu süre robotun yönlenme açısı, dönüş yarıçapı ve çizgisel hızı cinsinden hesaplanabilir. Yönlenme açısı ve yarıçap ile robotun üzerinde hareket edeceği yayın uzunluğu belirlenecek ve belirlenen hızda bu yayın ne kadar sürede çizileceği de bize komutun işlenme süresini saniye cinsinden verecektir.

$$\text{Komutun İşlenme Süresi} = \text{yuvarla}\left(\frac{\pi \cdot r \cdot \theta_{dümen}}{180 \cdot V_{sürüş}}\right) \tag{2.7}$$

Robot farklı yarıçaplarda yaylar çizerek aynı yönlenme açısını sağlayabilmektedir. Farklı yarıçaplar çizebilmek için ise dümen tekerleklerinin farklı açılarda yönlenmesi gerekmektedir. Bu açı $\theta_{dümen}$ olarak adlandırılmıştır. $\theta_{dümen}$ derece cinsinden aşağıdaki şekilde hesaplanabilir;

$$\theta_{dümen} = \left[ArcTan\left(\frac{L_u}{r}\right)\right]\left(\frac{180}{\pi}\right) \tag{2.8}$$

Dümen tekerlekleri bir servo motor yardımıyla hareket ettirilir. Gerekli dümen açısını ($\theta_{dümen}$) sağlamak amacıyla servo motorun hangi konuma geleceğini saptamak için bir dönüşüm yapılır. Bu dönüşüm dümen mekanizmasının yapısına bağlıdır. Kullandığımız yapı için θ_{servo} derece cinsinden aşağıdaki şekilde hesaplanabilir;

$$\theta_{servo} = \pm 3 \cdot \text{yuvarla}(\theta_{dümen}) \tag{2.9}$$

Servo açısı, komut uzunluğunu etkileyen önemli bir faktördür. Bu durumda oluşturulan komut dizgesinde gezgin robot komutlarının uzunlukları; dönüş yoksa bir bayt, dönüş varsa, üç bayt olmaktadır.

Denetim kartı servo motoru uygun konuma getirebilmek için PWM modülasyonlu bir işaret kullanmaktadır. Bu nedenle, servo açısı PC'den denetim kartına açı olarak değil, darbe boşluk oranı olarak gönderilir. Servo kontrol işaretinin periyodu 20ms'dir. Ancak darbenin uzunluğu 0.75 ile 2,25ms arasında doğrusal bir şekilde değişmektedir. Bu aralıkta bir darbe uzunluğu servo açısında ±90°'lik bir aralıkta değişime denk düşer. Buradan hareketle, servo açısının

darbe boşluk oranı karşılığı PC tarafından, aşağıdaki şekilde hesaplanmaktadır.

$$\text{Komut Kelimesi} = 64 \cdot \text{yuvarla}\left(\frac{750 + (90 - \theta_{servo})\left(\frac{1625}{180}\right)}{4}\right) \tag{2.10}$$

Robot, dümen mekanizmalı olduğundan belli bir yarıçap ile dönecektir. Bu nedenle, kullanıcının ayrıca yarıçap bilgisini girmesi gerekmektedir. Bu bilgi PC'nin servo açısını ve komut işleme süresini hesaplamasında kullanılacaktır. Fakat kullanıcı minimum olarak, robotun boyutlarından dolayı müsaade edeceği yarıçap değerini girebilmektedir. Bu yarıçap, kullandığımız mekanik yapı için cm cinsinden aşağıdaki şekilde hesaplanabilir;

$$r_{\min} = \frac{L_u}{\tan(\theta_{dümen\max})} \tag{2.11}$$

$\theta_{dümen}$ en fazla ±30°'lik konum alabilmektedir. Robotun sürüş ve dümen tekerlekleri arası uzunluğu (L_u), 26 cm olduğundan, dönme yarıçapı (r_{min}) (2.11)'den yaklaşık en az 46 cm olmaktadır.

Gezgin robotun aldığı doğrusal yol "mesafe" olarak alınırsa, robotun ileri veya geri hareket ettiği durumlarda komutun işlenme süresi saniye cinsinden aşağıdaki şekilde hesaplanabilir;

$$\text{Komutun İşlenme Süresi} = \frac{mesafe}{V_{sürüş}} \tag{2.12}$$

2.2.2 Operatör birimi

Operatör komuta birimi bir kişisel bilgisayardan oluşmaktadır. Sistemde programlar, bu denetçi bilgisayar üzerinde geliştirilir ve derlenir. Sistem; kullanıcı terminal, floppy disk, ayrık bir I/O modülü ile haberleşir. Komut seti; algılayıcı kontrolü, algılayıcı bilgi alma ve hareket kontrolü için gerekli verileri içerir. GUI; kullanıcının, robotu izlemesini ve kontrol edebilmek için gerekli verileri rahatlıkla girebilmesini sağlar

Program, Borland Delphi 3'te geliştirilmiş olup, Windows 95 veya Windows 98 işletim sistemleri altında çalışmaktadır. Seri haberleşme için, MS Visual Basic 5.0 yazılım programının TMSComm kontrolü kullanılmıştır. PC ve denetim kartı arasındaki haberleşme seri port üzerinden kablo bağlantısı ile, dört farklı bit/saniye hızında, 1 başla, 1 dur biti ve eşlik biti kullanılmayan, 8 bit veri biçiminde yapılmaktadır.

Robot komuta programı, diferansiyel sürüşlü ve dümen mekanizmalı gezgin tabanları kontrol edebilecek şekilde tasarlanmıştır. Bununla birlikte; program, görsel arayüze 6 ekleme kadar bir kol eklenebilecek şekilde geliştirilmiştir.

Program, Doğrudan Çalışma Modu ve Kayıttan Çalışma Modu olmak üzere iki modda çalıştırılabilmektedir. Doğrudan Çalışma Modu, basit bir denetim kolu (joystick) kontrolüne benzemektedir. Kayıttan Çalışma Modu'nda ise, komutlar bir görev listesine yazılmakta ve bu görev listesinden sıra ile işlenmektedir. Daha sonra istendiğinde yazılan komutlar diferansiyel sürüşlü robot için bir ".dir" dosyasına, dümen mekanizmalı robot için bir ".dur" dosyasına kaydedilmekte veya daha önce kaydedilmiş bir komut listesi, görev listesine yüklenebilmektedir. Program ayrıca dokunma algılayıcılarından aldığı bilgileri kullanıcıya aktarabilmektedir. İleri-geri hareket, dönme, bekleme görevleri dışında robot; kullanıcının ekleyebileceği döngü içinde de hareket edebilmektedir.

2.2.3 Gezgin robot mekaniği işletim ve denetim kartı

Gezgin platform işletim ve denetim birimi, operatör komuta birimi ile robotun mekanik sistemi arasında bir arabirim oluşturmaktadır. Programda kullanılan TMSComm modülü ancak karakter biçiminde veri iletişimi sağlayabilmektedir. Bu nedenle, PC'den bu birime, karakter biçiminde gelen komutlar, yorumlanarak gerekli işaretlere dönüştürülür ve motorlar için sürüş işaretleri üretilir.

2.2.4 Haberleşme birimi

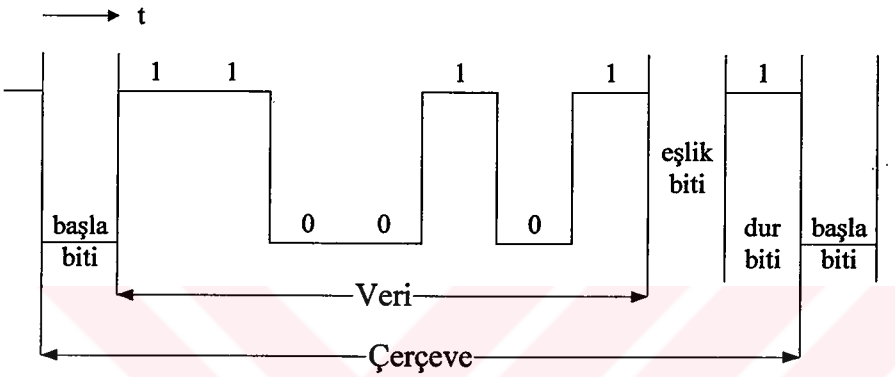
Operatör komuta birimi ile işletim ve denetim birimi arasındaki haberleşme, RS232 portu üzerinden kablo bağlantısı ile yapılmaktadır. Haberleşme 9600 bit/saniye hızında 1 başla, 1 dur biti ve eşlik biti kullanılmayan, 8 bit veri biçiminde yapılmaktadır.

PC ve denetim kartı asenkron seri veri biçiminde haberleşmektedir. Her temel komut ve denetim kartından PC'ye gönderilen algılayıcı durum bilgisi 1 bayt veri şeklindedir. Komutlar ve algılayıcı durum bilgileri, gönderen birim tarafından düzenlendikten sonra alıcı birim tarafından çözümlenerek değerlendirilir.

2.2.4.1 Asenkron haberleşme

Asenkron haberleşme kişisel bilgisayarlar, anaçatı bilgisayarlar (mainframe) ve her türlü terminal ile iletişim sağlamakta kullanılır. Bu biçim (Şekil 2.5), başla biti, veri, eşlik ve dur bitinden oluşan bir çerçeve yapısıdır.

İletişimde veri hattı boşken, bir bit süre ile boşluk konumuna çekilerek, başla biti verilmiş olunur. Sonra 7 veya 8 bit veri gönderilir. Verinin ardından hata algılamada kullanılan eşlik biti gönderilir. Çerçeve, içersindeki tek sayıda bit değişimlerini algılayabilecek yapıdadır. İletişim sırasında meydana gelebilecek iki bitlik değişimleri algılayamaz. Ayrıca asenkron veri iletişiminde karşılaşılan taşma hatası ve çerçeveleme hatası gibi hatalar vardır. Hızlı çerçeve iletiminde ve çerçeve başla ve dur bitlerinin kaçırılması durumunda bu hatalar ortaya çıkar (Güvercin, 1999).



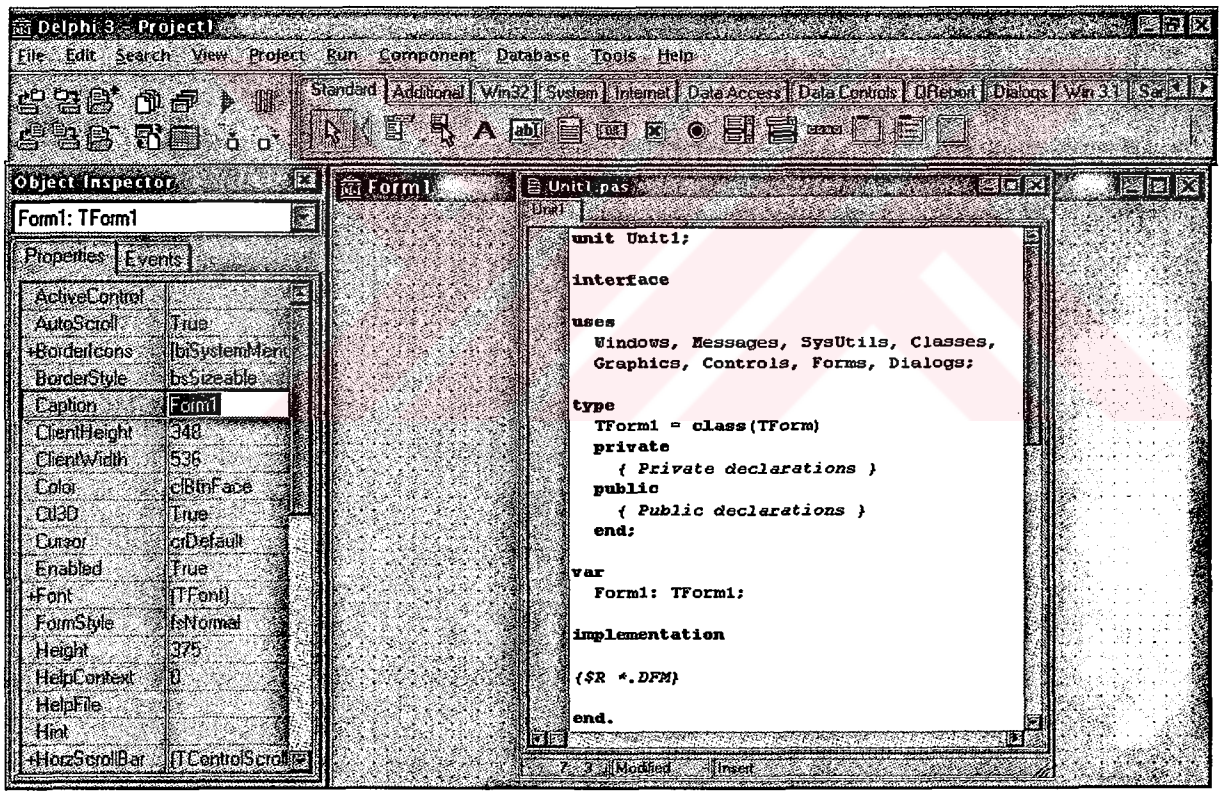
Şekil 2.5 Asenkron veri biçimi

Asenkron yapıda overload denilen veri dışındaki kontrol bitleri fazladır. Bu yüzden veri iletim verimi senkron yapıya nazaran daha düşüktür. Fakat hata oluşumu durumunda senkron yapıya göre paket yapısı kısa olduğu için bunun telafisi çabuk olabilmektedir.

3. GEZGİN ROBOT PROGRAMINDA KULLANILAN YAZILIM

Borland, Windows işletim sistemi üzerinde çalışan görsel tabanlı programların yaygınlaşması üzerine; daha önce Visual Pascal olarak adlandırılacağı öngörülen bir Pascal versiyonu üzerinde çalışmaya başladı. Visual Basic benzeri Pascal'ın kod adı Delphi idi. Ancak gün gelip Delphi Projesi tamamlanınca; Borland, Visual Pascal adından vazgeçip, hazırladığı Pascal tabanlı nesne yönetimli görsel program geliştirme aracını Delphi adıyla pazarlamaya başladı. Delphi birçok yönüyle Pascal'a benzemektedir. Yüksek performanslı, dayanıklı kontrollere sahiptir (Karagülle vd., 1996; Yanık, 1996; Matcho vd., 1996).

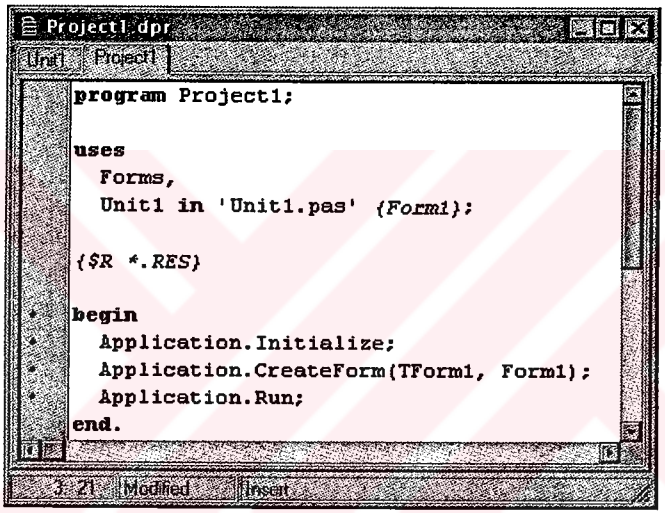
Delphi ilk açıldığında (Şekil 3.1) ekrana dört pencere gelir. Bunlar; "Speedbar" ve "Component Palette" araç çubuklarını içeren "Delphi 3–Project1" penceresi, "Object Inspector", "Form1" ve "Unit1.pas" pencereleridir.



Şekil 3.1 Delphi açıldığında ekrandaki durumu

Bütün Windows uyumlu programlar ekrana bir pencere içinde gelirler. Bu açıdan Windows ortamı içinde bir program yazılırken, yazılan programın çalışması sırasında, kullanıcıya nasıl bir pencere içinde yansıyacağını belirlemek gerekir Delphi'de bu pencerelere "Form" adı verilmektedir. Delphi başlatıldığında "Unit1.pas" adında Pascal program kodu içeren bir ünite hazırlanmakta ve Delphi ile birlikte otomatik olarak hazırlanan projeye dahil edilmektedir.

Burada program dahilinde kullanılacak olan üniteler "uses" deyimi ile programa dahil edilmektedir. Değişkenler "var" deyimi içinde tanımlanmakta ve program "end" deyimi ile bitirilmektedir. "type" deyimi içersinde yeni tipler, ünite dahilinde kullanılacak nesne ve prosedürler ve varsa "public" ve "private" özellikli diğer tipler tanımlanmaktadır. Delphi projesine otomatik olarak dahil edilen form, gerçekte bir nesnedir. Bu nesne "class" deyimi ile TForm nesne tipinden yararlanılarak tanımlanmaktadır. "TForm", formları tanımlamada kullanılan nesne tipidir. "implementation" deyimi içersinde de, ünitenin asıl program satırlarına geçilmektedir. Ünitelerin herbiri, ayrı bir ".pas" uzantılı dosyaya kaydedilir. Projeler kaydedilirken, o ana kadar forma kaydedilmiş her form için, diskte otomatik olarak ".dfm" uzantılı bir Form dosyası hazırlanır. Projeler (Şekil 3.2) ise, ".dpr" uzantısı ile kaydedilir.



Şekil 3.2 Delphi proje sayfası

Projeye verilen isim, "program" deyiminin yanına yazılmaktadır. "uses" deyimi ile, proje dahilinde kullanılmak istenen hazır Delphi üniteleri projeye dahil edilmektedir. Delphi başlatılması sırasında otomatik olarak hazırlanan projenin, asıl program satırlarının yazıldığı "begin-end" işlem bloğunun arasında bulunan program satırları ile "CreateForm()" metoduyla proje için bir form hazırlanmaktadır. "Run" metodu ile üzerinde çalışılan proje çalıştırılır.

Bir form üzerine oluşturulacak ekran görüntüsü ise "Component Palette" üzerindeki kontrollerle yapılır. "Object Inspector" adlı pencerede ise; formun özellikleri hakkında bilgi bulunmaktadır. Bu pencerenin "properties" kısmı ile forma yerleştirilmiş seçili elemanın tasarım zamanı özellikleri değiştirilirken, "events" kısmı ile de seçili elemana ait olaylar görüntülenir.

3.1 Yazılımda Kullanılan Fonksiyonlar

Length(): Bu fonksiyonla, kendisine parametre olarak verilen karaktersel bir bilginin uzunluğu bulunur.

Pos(): Karaktersel bir bilgide, başka bir karaktersel bilgiyi aramak amacıyla kullanılır.

IntToStr() ve StrToInt(): IntToStr(), tam sayısal bir bilgiyi, karaktersel bilgi tipine; StrToInt() ise karaktersel bir sabit bilgiyi tam sayısal bilgi tipine çevirmek için kullanılır.

Chr(): 0 ile 255 arasındaki herhangi bir sayıya karşılık gelen ASCII karakterini elde etmek için kullanılır.

Ord(): ASCII karakter kümesindeki herhangi bir karakterin kodunu bulmak için kullanılır.

Round(): Verilen kesirli bir sayının 0.5 ve üzerindeki küsuratını bire tamamladıktan sonra, küsüratı atarak sayıyı tam sayıya çevirir.

Val(): Karaktersel bir veriyi, sayısal veri tipine çevirmek için kullanılır. Dönüştürme işlemi yapılırken meydana gelen hatalar ile ilgili sayısal değer, fonksiyonun üçüncü parametresi olarak verilen değişkene aktarılır.

Copy(): Kendisine parametre olarak verilen karaktersel bir bilginin, soldan itibaren istenilen kadarını ayırmak veya çıkarmak amacıyla kullanılmaktadır.

3.2 Yazılımda Kullanılan Kontroller

TEdit (Metin Kutusu): Kullanıcının bilgi girişi yapmasına imkan veren ve istenen bir bilgiyi tek satır halinde ekrana yazdırabilen bir kontroldür.

TListBox (Liste Kutusu): Elemanları listelemek, sıralamak gibi özellikleri olan genel amaçlı bir kontroldür. Programda, görev listesi olarak kullanılmıştır.

TTabbedNotebook: Birden fazla kendisine has kontrolleri barındıran sayfalar içeren bir kontroldür. Programda, gezgin tabanın görevini seçmek için kullanılmıştır.

TButton (Komut Düğmesi): Programda, "Görevi Ekle" düğmeleri olarak kullanılmıştır.

TBitBtn (Resimli Komut Düğmesi): Bu kontrol standart komut düğmesinin resimli halidir. "Kind" özelliği sayesinde hazır düğmelerden biri seçilebilir. Programdaki "Çıkış" düğmeleri, bu kontrol kullanılarak yaratılmıştır.

TSpeedButton: Tek başına kullanıldığında komut düğmesinden bir farkı yoktur. Grup halinde çalıştırılabilmektedir. Üzerine resim yüklenebilir. Programda; araç çubuğundaki düğmeler, bu kontrol kullanılarak yaratılmıştır.

TLabel (Etiket): Form üzerinde açıklama yazmak veya başka bir kontrolün ne işe yaradığını belirtmek için kullanılır. Programda, başlıkları oluşturmak için kullanılmıştır.

TTrackBar (Ayar Çubuğu): Değişik ayarlama işlemlerinde görsel arabirim olarak kullanılır. Programda, hız ve açı verilerinin girişi için kullanılmıştır.

TCheckBox (İşaret Kutusu): Kullanıcının bir seçeneği aktif ya da pasif durumuna getirmesi için kullanılan bir kontroldür. Programda, robot kol eklemlerinin seçimi için kullanılmıştır.

TRadioGroup (Radyo Kutusu): İşaret kutusundan farklı olarak birkaç seçenekten birini seçme imkanı veren bir kontroldür. Gruptaki düğmelerden biri seçildiğinde diğeri kendiliğinden seçilmiş özelliğini kaldırır. Programda; sağ-sol, ileri-geri girişleri ve algılayıcı çıkışları için kullanılmıştır.

TComboBox (Aşağıya Doğru Açılan Liste): Genellikle değerleri daha önceden belli olan elemanların seçimi için kullanılırlar. Programda, port seçimi için kullanılmıştır.

TStatusBar (Durum Çubuğu): Windows'un durum çubuğunu temsil eder.

TMainMenu (Menü): Delphi formlarına menü eklemeyi, menülere belirli bir biçim vermeyi ve oluşturulan menülere program kodu yazmayı mümkün kılan bir kontrol elemanıdır. Programın menüsü bu kontrol vasıtası ile tasarlanmıştır.

TOpenDialog (Dosya Aç Penceresi): Standart aç diyalog penceresinin kullanılmasını sağlar.

TSaveDialog (Dosya Kaydet Penceresi): Standart kaydet diyalog penceresinin kullanılmasını sağlar.

TTimer: Zamanın belli aralıklarında, "OnTimer" olayını meydana getiren bir kontroldür. Programda, görev listesindeki komutların, denetim kartı tarafından komut süresi kadar işlenmesini sağlar.

TMSComm: MSComm kontrolü; verinin bir seri port üzerinden alışverişine izin vererek, uygulama için seri haberleşme sağlar. Birçok durumda, bir olayın vuku bulduğu an (bir karakterin gelmesi ya da CD veya RTS hatlarında bir değişiklik olduğunda) haberdar edilmek istendiğinde, bu haberleşme olaylarını idare etmek için MSComm kontrolünün "OnComm"

olayı kullanılır. Bu olay haberleşme hatalarını da bulur. Programda, PC ile denetim kartının haberleşmesi için kullanılmıştır. Özellikleri şunlardır:

CommPort: Hangi porttan haberleşileceği bu özellik vasıtasıyla belirlenir.

Settings: Haberleşme hızı, eşlik biti, veri bitleri, başla ve dur bitleri belirlenir.

PortOpen: Bir portu açar veya kapatır.

Input: Alıcı tamponundan karakter alır.

Output: Verici tamponuna gönderilmek istenen karakterleri yazar.

Handshaking: Donanımın el sıkışma protokolünü belirler.

InBufferCount: Alıcı tamponunda bekleyen karakter sayısını verir.

InBufferSize: Alıcı tamponunun büyüklüğünü bayt olarak belirler.

InputLen: "Input" özelliğinin alıcı tamponundan okuduğu karakter sayısını belirler.

OutBufferCount: Verici tamponunda bekleyen karakter sayısını belirler.

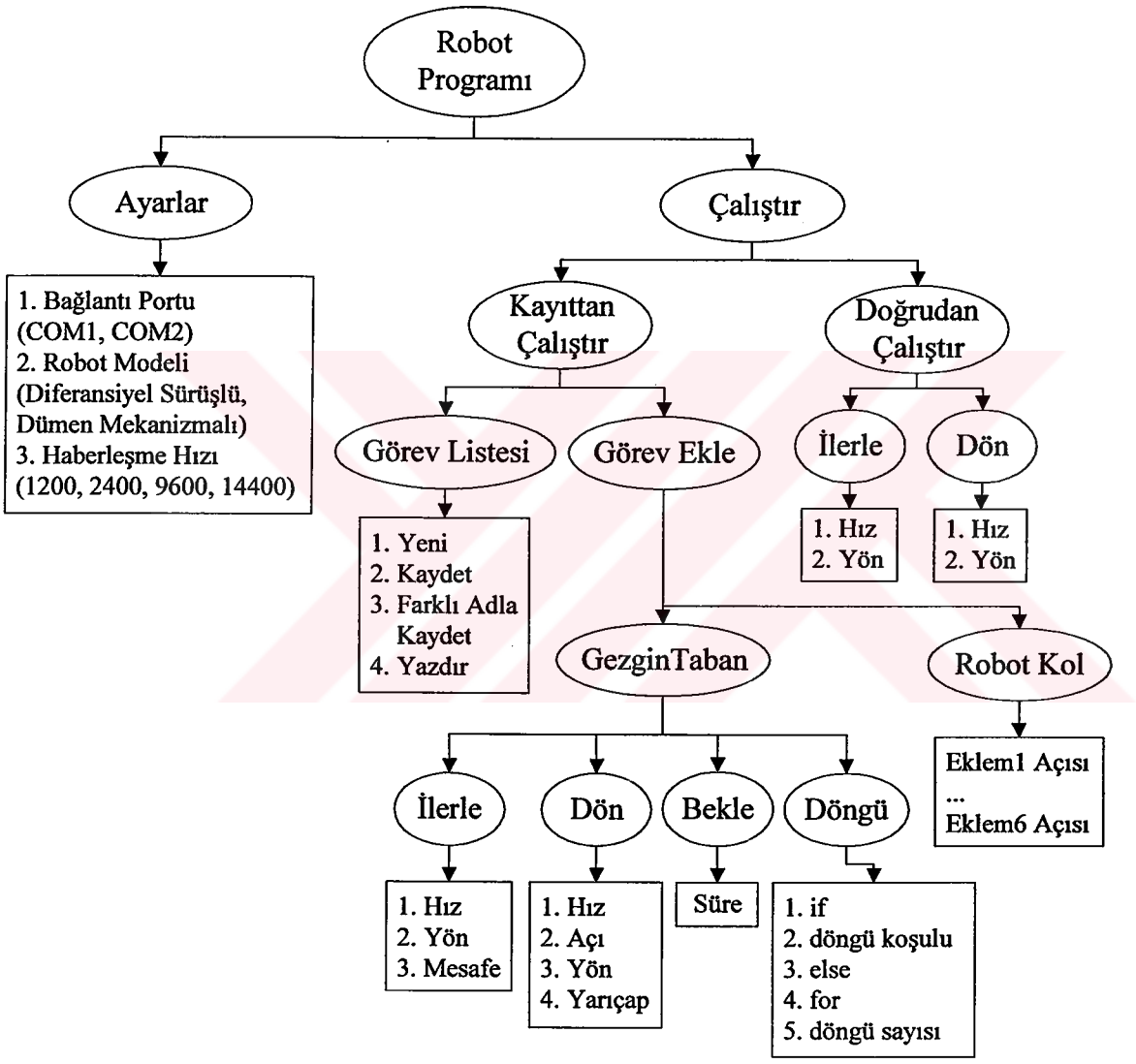
OutBufferSize: Verici tamponunun büyüklüğünü bayt olarak belirler.

RThreshold: MSComm kontrolü "OnComm" olayını oluşturmadan alınacak karakter sayısını belirler.

SThreshold: Verici tamponunda, "OnComm" olayı oluşmadan izin verilen minimum sayıdaki karakteri belirler.

4. TASARLANARAK GERÇEKLEŞTİRİLEN GEZGİN ROBOT PROGRAMI

Tasarlanarak gerçekleştirilen gezgin robot programı sistemin beynini oluşturmaktadır. Program, diferansiyel sürüşlü ve dümen mekanizmalı iki tür gezgin robotu çalıştırabilecek şekilde tasarlanmıştır. Bu iki tür gezgin robot Kayıttan Çalışma Modu ve Doğrudan Çalışma Modu olmak üzere iki tür modda çalışabilmektedir. Bu modlar ilerleyen kısımlarda anlatılmıştır. Şekil 4.1'de programın çalışma şeklini anlatan blok diyagram verilmiştir.



Şekil 4.1 Programın blok diyagramı

Yazılımın bir özelliği de Türkçe olmasıdır. Genel olarak yazılım, "ileri git", "geri git", "sağa dön", "sola dön" gibi temel, basit ve kullanıcının anlayabilmesi kolay komutlardan oluşmuştur. Program bu temel komutların görüntülenebileceği bir editöre sahiptir. Bu editör salt okunur yapılarak kullanıcı hataları ortadan kaldırılmıştır. Komut girişleri programdaki "Görev Ekle" düğmeleri ile yapılmaktadır. Program denetimi sayesinde eksik komut girişleri

de engellenmektedir. Eğer istenmeyen komutlar girilmişse, bu komutlar mouse ile seçilerek "Görev Sil" düğmesi ile silinebilmektedir.

Kayıttan Çalışma Modunda kullanıcı kendi komut listesini oluşturabilmektedir. Böylece temel komutlar kullanılarak robotun istenen bir görevi yerine getirmesi sağlanır. Programa "for" ve "if" döngüleri eklenerek görevin devamlı ya da belli bir koşul altında yapılması sağlanır. Ayrıca editöre yazılan robot görevi kaydedilerek, robotun aynı görevi tekrar yapması sağlanabilir. Program geliştirmeye açık bir şekilde tasarlanmıştır.

Aşağıda diferansiyel sürüşlü ve dümen mekanizmalı gezgin robotlar ve robot kol için örnek üç görev listesi verilmiştir. Görüldüğü gibi komutlar basit ve kolay anlaşılır olmakla birlikte, mevcut programlama dillerinden herhangi birini bilmeyen biri için de kolay kullanılabilir bir özellik taşımaktadır.

Diferansiyel sürüşlü gezgin robot için örnek görev listesi:

İleri Git, Hız=6, Mesafe=4m
Sağa Dön, Hız=4, Dönme Açısı=234°
5sn bekle
Sola Dön, Hız=4, Dönme Açısı=113°

Dümen mekanizmalı gezgin robot için örnek görev listesi:

for i = 1 to 2
begin
1sn bekle
if arkasağda engel varsa then
begin
2sn bekle
İleri Git, Hız=4, Mesafe=1m
end
else
begin
Sağa ve Geriye Dön, Hız=4, Dönme Açısı=148°, Yarıçap=45cm
2sn bekle
end
end
1sn bekle

Robot kol için örnek görev listesi:

M1=22° M3=60° M5=46°
M2=11° M4=90° M6=-90°

Programda DC motorlar on ayrı hız kademesinde hareketlendirilebilirler. Dokunma algılayıcılarından gelen bilgi kullanıcı tarafından gözlemlenebilmektedir. Böylece kullanıcı robotun çevresi hakkında bilgi alabilmektedir. Veri giriş-çıkışı yapılacak olan port kullanıcı tarafından değiştirilebilmektedir. Kullanıcı COM1 veya COM2 portlarından birini seçebilmektedir. Programın menüsünden dört iletişim hızından biri (1200 bit/saniye, 2400 bit/saniye, 9600 bit/saniye, 14400 bit/saniye) seçilebilmektedir. Ayrıca kullanıcı istediğinde, görev listesini yazıcıya gönderebilmektedir.

Programın görsel kullanıcı arabirimi, sisteme bir robot kol eklenebilecek şekilde tasarlanmıştır. Kullanıcı, bu robot kolun eklem sayısını seçebilmekte ve eklemlerde kullanılan servo motorlara göre servo açılarını değiştirebilmektedir.

Programın görsel kullanıcı arabiriminde (GUI), diferansiyel sürüşlü ve dümen mekanizmalı robotların mekanik özelliklerinden gelen küçük farklar olmakla birlikte, genel olarak yaptıkları işler aynıdır. Fakat aynı işleri yapabilmeleri için PC'den denetim kartına gönderilecek komut dizgeleri fark etmektedir. Aşağıda diferansiyel sürüşlü ve dümen mekanizmalı iki tür gezgin robot için PC'den denetim kartına gönderilecek komut dizgeleri açıklanmıştır.

4.1 Diferansiyel Sürüşlü Gezgin Robot için Komut Dizgesi

Gezgin robotun istenen şekilde hareket etmesi özdeş olan sürüş motorlarının hızlarının ve dönüş yönlerinin kontrol edilmesi ile sağlanır. Bu nedenle DC motorların her iki yönde ve farklı hızlarda çalışmasına izin verecek şekilde sürülmeleri gerekmektedir. Yine aynı nedenle, operatörün verdiği komutların yalnız motor ismi, hızı ve dönüş yönü bilgilerini taşıması yeterlidir.

Robot, yalnızca tekerlek ekseninin orta noktası üzerinde dönecektir. Bu nedenle, dönüş esnasında her iki sürüş motoru da hareket halinde olacak ve aynı hızda, fakat ters yönde hareket edeceklerdir. İleri ve geri harekette de aynı hızda olacaklarından, komutta tek bir hız değerinin yer almasının yeterli olacağı görülür. Bu, komut uzunluğunu etkileyen önemli bir faktördür. Bu durumda oluşturulan komut dizgesinde (Şekil 4.2), gezgin robot komutlarının uzunlukları bir bayttır.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Şekil 4.2 Diferansiyel sürüşlü robot için komut yapısı

Bit0 : Sol tekerlek için dönüş yönü (1 ise ileri, 0 ise geri)

Bit1 : Sağ tekerlek için dönüş yönü (1 ise ileri, 0 ise geri)

Bit2 : Hareket göstergesi (1 ise hareket var, 0 ise hareket yok)

Bit3 : Kullanılmıyor, 0 okunur

Bit4-7 : Motorlar için hız kademeleri

Robot, on ayrı hız kademesinde gidebilmektedir. Komutlarda bu hız değerleri, kademe numaralarıyla belirtildiğinden dört bitlik bir bölge yeterli olmuştur. Kontrol kartı motora göndereceği hız bilgisini aldığı hız kademe verisinden hesaplayacaktır.

Denetleyici kart; işleyeceği komutu alınca ilk olarak bit2'yi kontrol ederek bunun bir bekleme komutu mu, yoksa hareket komutu mu olduğunu belirler. Eğer bir bekleme komutu ise motorları durdurur. Ancak eğer bir hareket komutu ise, iki motor için ayrı ayrı yön şartlamalarını ve hız ayarlamalarını yapar. Buna göre robot, bit1 ve bit0; 00 ise geri, 11 ise ileri, 01 ise sola ve 10 ise sağa doğru hareket yapacaktır.

4.2 Dümen Mekanizmalı Gezgin Robot için Komut Dizgesi

Gezgin robotun istenen şekilde hareket etmesi, sürüş motorunun hız ve dönüş yönü ile dümen mekanizmasını hareket ettiren servo motor açısının kontrol edilmesi ile sağlanır. Bu nedenle DC ve servo motorların ayrı ayrı sürülmeleri gerekmektedir. Aynı nedenle, operatörün verdiği komutların; DC motor hızı ve yönü, dönüşün olup olmadığı, dönüş varsa servo açısı bilgilerini taşıması gerekmektedir. Dönüş yönü servo açı bilgisinin içine gömülmüştür.

Robot, dümen mekanizmalı olduğundan belli bir yarıçap ile dönecektir. Bu nedenle, kullanıcının ayrıca yarıçap bilgisini girmesi gerekmektedir. Bu bilgi PC'nin servo açısını hesaplamasında kullanılacaktır. Fakat kullanıcı minimum olarak, robotun boyutlarından dolayı müsaade edeceği yarıçap değerini girebilmektedir. Servo açısı, komut uzunluğunu etkileyen önemli bir faktördür. Bu durumda oluşturulan komut dizgesinde (Şekil 4.3), gezgin robot komutlarının uzunlukları; dönüş yoksa bir bayt, dönüş varsa üç bayt (Şekil 4.4) olmaktadır.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Şekil 4.3 Dümen mekanizmalı robot için komut yapısı

Bit0 : Kullanılmıyor, 0 okunur

Bit1 : Kullanılmıyor, 0 okunur

Bit2 : Yön göstergesi (1 ise ileri, 0 ise geri yönde hareket var)

Bit3 : Dönüş göstergesi (1 ise dönüş var, 0 ise dönüş yok)

Bit4-7 : DC motor için hız kademeleri

Robot, on ayrı hız kademesinde gidebilmektedir. Komutlarda bu hız değerleri, kademe numaralarıyla belirtildiğinden dört bitlik bir bölge yeterli olmuştur. Kontrol kartı motora göndereceği hız bilgisini aldığı hız kademe verisinden hesaplayacaktır.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	Bit5	Bit4	0	0	0	0

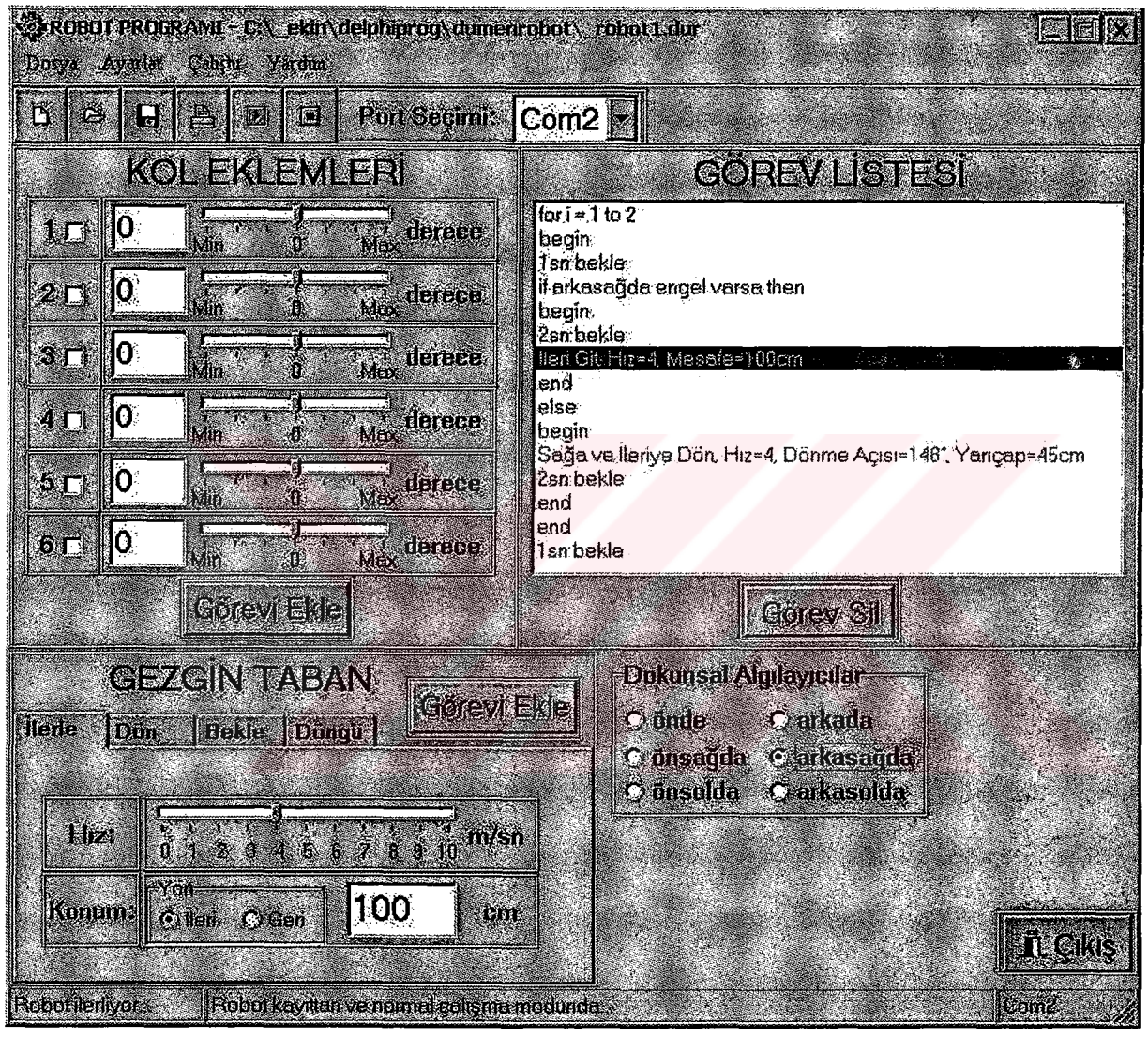
Şekil 4.4 Dümen mekanizmalı robotun dönüş komutu için 2 baytlık ek komut yapısı

Denetleyici kart; işleyeceği komutu alınca ilk olarak bit3'ü kontrol ederek bunun bir dönüş komutu mu, yoksa ileri ya da geri komutu mu olduğunu belirler. Eğer ileri ya da geri komutu ise (bit3: 0); PC, yön şartlaması ve hız ayarlamasını içeren bir baytlık veri gönderir. Ancak eğer bir dönüş komutu ise (bit3: 1); PC, yön şartlaması ve hız ayarlamasını içeren bir baytlık veriye ek olarak, on bitlik servo açı bilgisine ait iki baytlık veri ile birlikte, üç bayt veri gönderir. Servo açı verisi daha hassas bir dönüş sağlamak amacıyla on bit olarak belirlenmiştir. Bu on bitin, en ağırlıklı sekiz biti ikinci baytta, ağırlıksız iki biti ise üçüncü baytın dört ve beşinci bitlerine yerleştirilir. Bunun nedeni denetim kartında bu bitlerin bir kontrol yazmacının aynı numaralı bitlerine yazılmasıdır.

4.3 Programın Açıklaması

Program (Şekil 4.5) Borland Delphi 3'te geliştirilmiş olup, Windows işletim sistemi altında çalışmaktadır. Seri haberleşme için, MS Visual Basic 5.0 kontrolü olan Microsoft Comm Control 6.0 (Sürüm 1.1) TMSComm ActiveX kontrolü kullanılmıştır. PC ve denetim kartı arasındaki haberleşme seri port üzerinden kablo bağlantısı ile, 9600 bit/saniye hızında, 1 başla, 1 dur biti ve eşlik biti kullanılmayan, 8 bit veri biçiminde yapılmaktadır. Donanım el

sıkışması yoktur. Alıcı tamponu büyüklüğü 1024 bayt, verici tamponu büyüklüğü 512 bayt olarak seçilmiştir. Bazı ActiveX kontrolleri ancak Windows'a kayıt edilirse çalışabilmektedir. Bu yüzden, programın çalışabilmesi için gerekli dosyalar kullanılarak bu kontrolün Windows'a kayıt edilmesi gerekmektedir.



Şekil 4.5 Robot programı

Kullandığımız robotlar diferansiyel sürüşlü ve dümen mekanizmalı gezgin tabanlardan oluşmaktadır. Bununla birlikte; programın görsel arabirimi, sisteme 6 eklemli bir kol eklenebilecek şekilde geliştirilmiştir. Robotun, gezgin taban ve kol görevleri, görev listesine ayrı ayrı girilmektedir yani ayrı ayrı görev ekle düğmeleri vardır.

Program robotu iki ayrı modda çalıştırabilecek şekilde tasarlanmıştır. Kullanıcı bu modları programdaki Çalıştır menüsünden seçerek çalıştırabilmektedir. Bunlar;

- Doğrudan Çalışma Modu
- Kayıttan Çalışma Modu

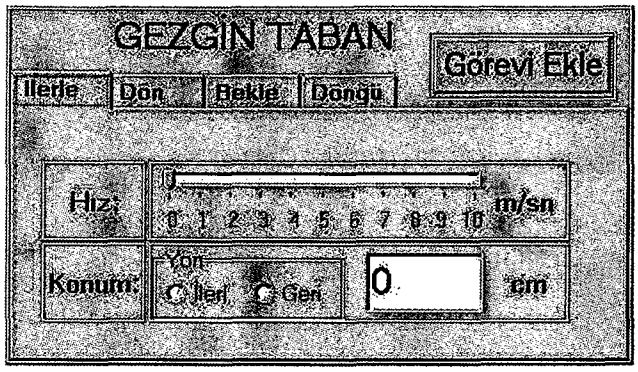
Doğrudan Çalışma Modunda komutlar robota direkt olarak verilmektedir. Robot, kullanıcının görsel arabirimdeki kontroller üzerinde yaptığı her değişikliğe hemen cevap verebilmektedir. Fakat bu modda komutların herhangi bir yere yazılması, kaydedilmesi veya saklanması söz konusu değildir. Komutlar anlık olarak işlenir ve cevap alınır. Bu mod denetim kolu (joystick) kontrolüne benzemektedir.

Kayıttan Çalışma Modunda ise komutlar, robota kayıttan verilmektedir. Bunun için programda bir görev listesi mevcuttur. Kullanıcı önce; görev listesine, daha önce kaydetmiş olduğu komut dizisini mi yükleyeceğine, yoksa yeni bir komut dizisi mi gireceğine karar verecektir. Sonuçta; görev listesine girilen veya yüklenen komutlar sıra ile çalıştırılıp, robota gönderilir. Kullanıcı programdan çıkmak istediğinde ise program, yaratılan komut dizisinin kaydetme sorgusunu yapar.

Her iki modda da arayüz kartı PC'den komutu alır ve bir sonraki komut gelinceye dek o komutu çalıştırır. Komut işleme süresi her komut için robotun kinematik denklemlerinden yararlanılarak hesaplanır.

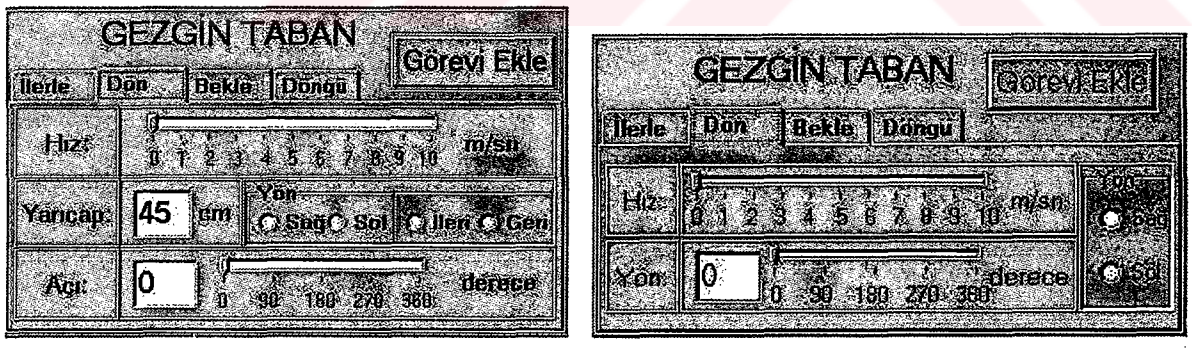
Kullanıcı, görev listesine yeni bir komut girmek istiyorsa, önce robotun yapması gereken görevi seçmelidir. Kullanıcı gezgin tabanın görevini girmek isterse, öncelikle Sayfalı Diyalog Kutusundan (TabbedNotebook) tabanın dört görevinden birini seçer. Bunlar; "İlerle", "Dön", "Bekle" ve "Döngü" dür. "İlerle" sayfasında (Şekil 4.6), robotun ileri veya geri hareketine ait kontroller; "Dön" sayfasında (Şekil 4.7) ise, sağa veya sola hareketine ait kontroller bulunur. "Bekle" sayfasında (Şekil 4.8), robotun bekleme süresine ait kontrol bulunurken; "Döngü" sayfasında (Şekil 4.9) ise görev listesine for veya if döngüsü koymak için kullanılacak kontroller bulunur.

"İlerle" sayfasında (Şekil 4.6), kullanıcı robotu hareket ettirebilmek için "Hız" etiketli ayar çubuğundan hız bilgisini girmelidir. "Yön" etiketli radyo kutusundan robotun ileri mi yoksa geri mi gideceği ve gideceği mesafe bilgisi santimetre cinsinden belirlenmelidir. Bunlardan herhangi biri eksik girildiği takdirde, hangi bilgi girilmemişse; o bilgiye ait hata kutusu ekrana gelir ve görev, görev listesine eklenmez. Veriler doğru olarak girildiğinde görev, "Görevi Ekle" düğmesi ile görev listesine eklenir ve yeni görev için liste kutusunda bir alt satıra geçilir.



Şekil 4.6 "İlerle" komut sayfası

"Dön" sayfasında (Şekil 4.7) kullanıcı, robotu hareket ettirebilmek için dümen mekanizmalı robotta "Hız" etiketli ayar çubuğundan hız bilgisini girmelidir. "Yön" etiketli radyo kutularından robotun ileri mi yoksa geri mi ve sağa mı yoksa sola mı gideceği belirlenmelidir. "Yarıçap" etiketli metin kutusundan robotun hangi yarıçapta dönmesi gerektiği ve "Açı" etiketli ayar çubuğundan ise kaç derece dönmesi gerektiği bilgileri girilmelidir. Bunlardan herhangi biri eksik girildiği takdirde, hangi bilgi girilmemişse; o bilgiye ait hata kutusu ekrana gelir ve görev, görev listesine eklenmez. Veriler doğru olarak girildiğinde görev, "Görevi Ekle" düğmesi ile görev listesine eklenir ve yeni görev için liste kutusunda bir alt satıra geçilir. Diferansiyel robotta dümen mekanizmalı robottan farklı olarak yarıçap girişi ve ileri-geri seçimi bulunmamaktadır.

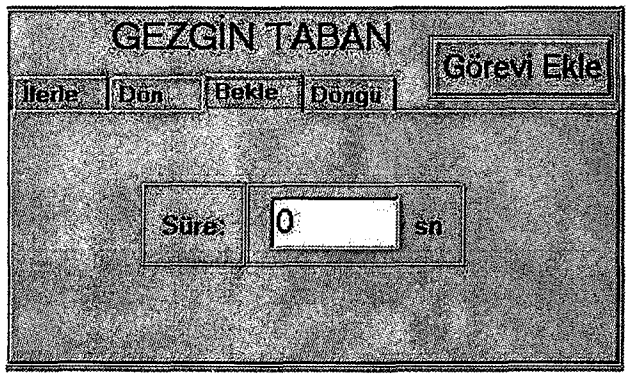


a) b)

Şekil 4.7 a) Dümen mekanizmalı robot için "Dön" komut sayfası
b) Diferansiyel robot için "Dön" komut sayfası

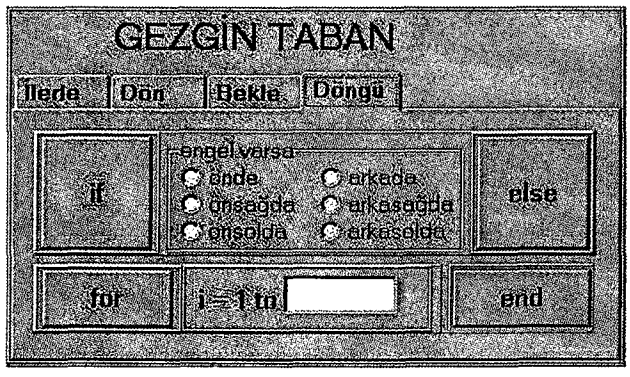
"Bekle" sayfasında (Şekil 4.8) kullanıcı, robotu bekletme işlemini yapabilmek için "Süre" etiketli ayar çubuğundan robotun kaç saniye bekleyeceğine dair bilgiyi metin kutusuna girmelidir. Bu bilgi eksik girildiği takdirde, bilgiye ait hata kutusu ekrana gelir ve görev, görev listesine eklenmez. Veri doğru olarak girildiğinde görev, "Görevi Ekle" düğmesi ile

görev listesine eklenir ve yeni görev için liste kutusunda bir alt satıra geçilir.



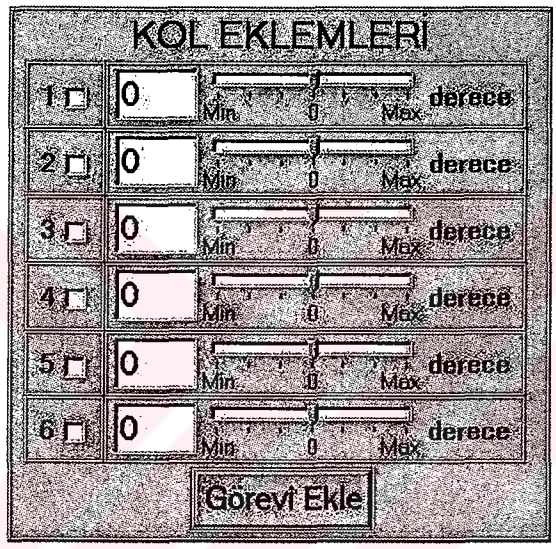
Şekil 4.8 "Bekle" komut sayfası

"Döngü" sayfası (Şekil 4.9), robot hareketlerini for veya if döngüsü ile düzenleyebilmek için tasarlanmıştır. "Döngü" sayfasında kullanıcı, görev listesine "for" veya "if" döngülerinden herhangi birini ekleyebilmektedir. "if" döngüsü ekleyebilmek için önce "engel varsa" etiketli radyo kutusundan robotun hangi durumda döngüye gireceği belirlenmelidir. Sonrada "if" etiketli düğmeye basılarak görev, görev listesine eklenir. Gerekli durumlarda "else" etiketli düğme ile "if" döngüsü tamamlanır. "for" döngüsü ekleyebilmek için önce döngüye ait panel içindeki metin kutusuna döngü sayısı girilir ve "for" etiketli düğmeye basılarak görev listesine eklenir. Bu bilgiler eksik girildiği takdirde, hangi bilgi girilmemişse; o bilgiye ait hata kutusu ekrana gelir ve görev, görev listesine eklenmez Bu düğmeler her "for", "if" ve "else" döngülerinden sonra görev listesine begin kelimesini otomatik olarak eklemektedir ve kullanıcı her döngüyü bitirmek amacıyla belirlediği yere "end" kelimesini eklemelidir. Bu da "end" etiketli düğmeye basılarak yapılır. Veriler doğru olarak girildiğinde, görev, görev listesine eklenir ve yeni görev için liste kutusunda bir alt satıra geçilir.



Şekil 4.9 "Döngü" komut sayfası

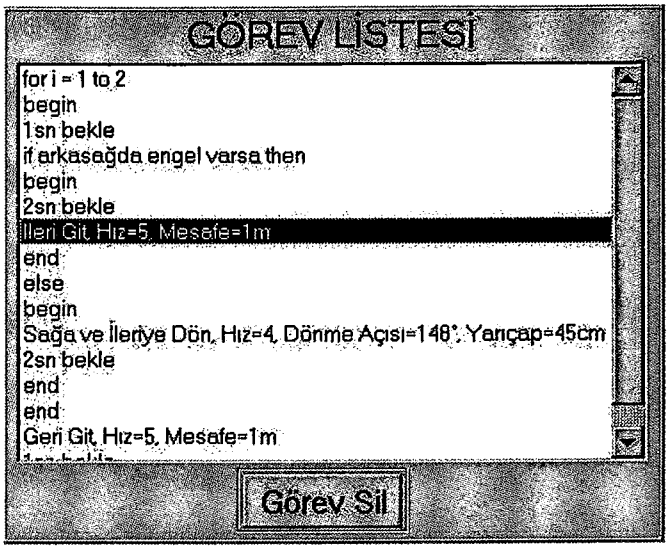
Görsel kullanıcı arabirimi, robota altı eklemli bir kol eklenebilecek şekilde (Şekil 4.10) tasarlanmıştır. Kola ait bir görevin görev listesine eklenebilmesi için, önce hangi eklem veya eklemler hareket ettirilmek isteniyorsa, o ekleme ait işaret kutusu seçilmelidir. İşaret kutusu seçilmeden ekleme ait panele müdahale edilememektedir. Kullanıcının, seçtiği işaret kutusuna ait eklemin paneli aktif hale gelecek ve böylece açı girişi yapılabilecektir. Seçilen eklem veya eklemlerin hangi açıyla döneceği ayar çubuğundan ayarlanmalı veya metin kutusuna direkt olarak yazılmalıdır. Eğer işaret kutusu seçilip, herhangi bir giriş yapılmazsa, o eklemin açı bilgisi sıfır olarak alınır. Bu da eklemin belirlenen referans konuma geri dönmesi demektir. Veri doğru olarak girildiğinde görev "Görevi Ekle" düğmesi ile görev listesine eklenir ve yeni görev için liste kutusunda bir alt satıra geçilir.



Şekil 4.10 Robot kola ait komut sayfası

Programın görev listesi Şekil 4.11'daki gibidir. Listede silinmek istenen görevler "Görev Sil" düğmesi ile silinebilir. Bunun için mouse ile silinmek istenen görev veya görevler işaretlenip, silinebilir. Fakat, döngü komutları kendilerine ait begin kelimeleri ile silinmek zorundadır. Tek başına bir döngü komutu ya da tek başına bir begin kelimesi silinemez. Silinmeye çalışıldığında, program hata verir ve silme işlemine izin vermez.

Yeni görevler, eğer herhangi bir görev işaretlenmemişse, listenin en altına eklenir. Fakat görevler arasına yeni bir görev eklenmek isteniyorsa, listede belirlenen görev mouse ile işaretlenir ve görev ekleme işlemi bu seçilen görevin üzerine yapılır. Döngü komutları ile begin arasına görev ekleme yapılamaz. Yapılmaya çalışıldığında program hata verir ve ekleme işlemi yapılmaz. Görev listesi salt okunur olduğundan klavye ile veri girişi yapılamaz, müdahale edilemez. Böylece kullanıcı hataları en aza indirgenmiştir.



Şekil 4.11 Görev listesi

Son yıllarda, Windows uyumlu programlarda açılan menüler ve komut vermede kullanmak için ekranın bir satırı boyunca bir araya getirilen komutları temsil eden küçük şekiller kullanılmaktadır. Bu ekranın bir satırı boyunca bir araya getirilmiş şekilli düğmelerin hepsine birden "Araç Çubuğu" adı verilir. Menüler yerine, bu düğmelerle komut verilebilir. Şekil 4.12'de programın menüsü ve araç çubuğu gösterilmiştir.

Şekil 4.12 Araç çubuğu

Programda "Dosya", "Ayarlar", "Çalıştır" ve "Yardım" ana menüleri bulunmaktadır. "Dosya" ana menüsü içinde; "Yeni", "Aç", "Kaydet", "Farklı Adla Kaydet", "Yazdır" ve "Çıkış" menüleri bulunmaktadır. "Ayarlar" ana menüsü; "Robot Modeli", "Haberleşme Hızı", "Değerleri Sıfırla" ve "Robot Kol Servo Ayarları" menülerini içermektedir. "Çalıştır" ana menüsü içinde ise; "Çalıştır", "Doğrudan Çalıştır", "Kayıttan Çalıştır" ve "Durdur" menüleri bulunmaktadır. "Yardım" ana menüsü içinde programın kim tarafından ve ne zaman yazıldığı hakkında bilgi veren "Hakkında" menüsü vardır.

"Yeni" menüsü, araç çubuğundaki ilk düğmeyle, "Aç" menüsü ikinci düğmeyle, "Kaydet" menüsü üçüncü düğmeyle (dosya yeni oluşturulmuş ise bu düğme Farklı Adla Kaydet menüsü ile), "Yazdır" menüsü dördüncü düğmeyle ve "Çalıştır" menüsü araç çubuğundaki beşinci düğmeyle aynı işi yapmaktadır.

"Dosya" ana menüsü, oluşturulan görev listesindeki robot komutlarını kaydetmek veya daha önce oluşturulmuş bir dosyayı görev listesine yüklemek, yeni bir görev listesi açmak veya varolan bir görev listesini yazıcıya göndermek için kullanılır. Bu menü grubunun görevi, standart Windows uygulamalarında olduğu gibidir.

"Araçlar" ana menüsündeki "Robot Modeli" menüsü, diferansiyel sürüşlü ve dümen mekanizmalı gezgin robot modellerinden birini seçmek için kullanılır. Bu iki model arasında birinden diğerine geçerken, görev listesinde o modele ait bir komut listesi varsa, program kullanıcıya bu listenin kaydetme sorgusunu yapar ve görev listesini temizler ve bütün değerleri sıfırlar. Her iki robot modeline ait komut listeleri, farklı uzantılı dosyalara kaydedilmektedir. Diferansiyel sürüşlü gezgin robota ait komut listesi ".dir" uzantılı, dümen mekanizmalı gezgin robota ait komut listesi de ".dur" uzantılı bir dosyaya kaydedilmektedir. Program dümen mekanizmalı robot modeli seçili olarak açılmaktadır.

"Araçlar" ana menüsündeki "Haberleşme Hızı" menüsü, verilerin denetim kartına hangi hızda iletileceğini belirlemek için kullanılmaktadır. Kullanıcı 1200 bit/saniye, 2400 bit/saniye, 9600 bit/saniye ve 14400 bit/saniye hızlarından birini seçebilmektedir. Program 9600 bit/saniye hızı seçili olarak açılmaktadır. Çalıştır düğmesine basıldığında bu menü işlevsiz hale gelmektedir. Bu ana menüdeki "Değerleri Sıfırla" menüsü de, görev listesi haricindeki değerleri başlangıç durumuna getirmek için kullanılır.

Programın "Ayarlar" ana menüsünde bulunan, "Robot Kol Servo Ayarları" menüsü robot kola ait ayarları yapmak için kullanılmaktadır. Bu pencerede (Şekil 4.13) yer alan "Eklem Sayısı" girişi, var olan robot kolun eklem sayısına göre ayarlanmalıdır. Pencerede yer alan "Servo Açı Sınırları" minimum ve maksimum olmak üzere iki girişe sahiptir. Bu girişler robot kol eklemlerine ait servo motorların dönme miktarlarına göre ayarlanmalıdır.

Şekil 4.13 Robot kol servo ayarları

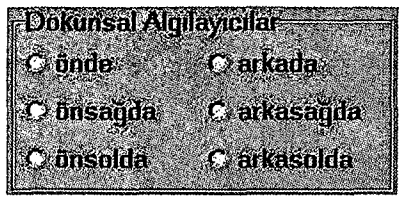
"Çalıştır" ana menüsündeki "Çalıştır" ya da araç çubuğundaki beşinci düğme, görev listesine yazılmış programı çalıştırmak, yani komutları teker teker robota göndermek için kullanılır. "Çalıştır" ana menüsündeki "Durdur" ya da Araç çubuğundaki altıncı düğme, programın çalışmasını durdurmak için kullanılır. Program ilk açıldığında, Çalıştır ve Durdur düğmeleri işlevsizdir. Görev listesine döngü komutları haricinde herhangi bir komut eklendiğinde, Çalıştır düğmesi aktif hale gelir. Görev listesindeki komutlar, robota Çalıştır düğmesi ile gönderilmeye başlandığında, Çalıştır düğmesi işlevsiz hale gelirken, Durdur düğmesi aktif hale gelir. Bu düğmelerin ikisi birden hiçbir zaman aktif olmamaktadır.

"Çalıştır" ana menüsündeki "Doğrudan Çalıştır" ve "Kayıttan Çalıştır" menüleri daha önce anlatılan Doğrudan Çalıştır ve Kayıttan Çalıştır Modlarına geçmek için kullanılır. Program, Kayıttan Çalıştır Modunda açılmaktadır. Menüden Doğrudan Çalıştır Moduna geçildiğinde; görev listesi, gezgin tabanın "İlerle" sayfasındaki mesafe girişi, "Dön" sayfasındaki açı ve yarıçap girişleri, "Bekle" sayfasındaki süre girişi ve gezgin tabana ve robot kola ait "Görevi Ekle" düğmeleri görünmez hale gelirken, "Döngü" sayfasındaki girişler, Çalıştır ve Durdur düğmeleri işlevsiz hale gelir. Tekrar Kayıttan Çalıştır Moduna geçildiğinde, bu girişler ve görev listesi eski haline (görünür hale) gelmektedir.

Araç Çubuğunda Durdur düğmesinden sonraki panel port seçimine aittir. Kullanıcı bu paneldeki liste kutusu ile robotla hangi porttan (COM1 veya COM2) haberleşeceğini seçebilir. Donanım ayarlamaları bu seçime göre program tarafından yapılmaktadır. Yapılması gereken denetim kartı bağlantısının PC'nin seçilmiş portuna yapılmasıdır. Program COM2 ile açılmaktadır.

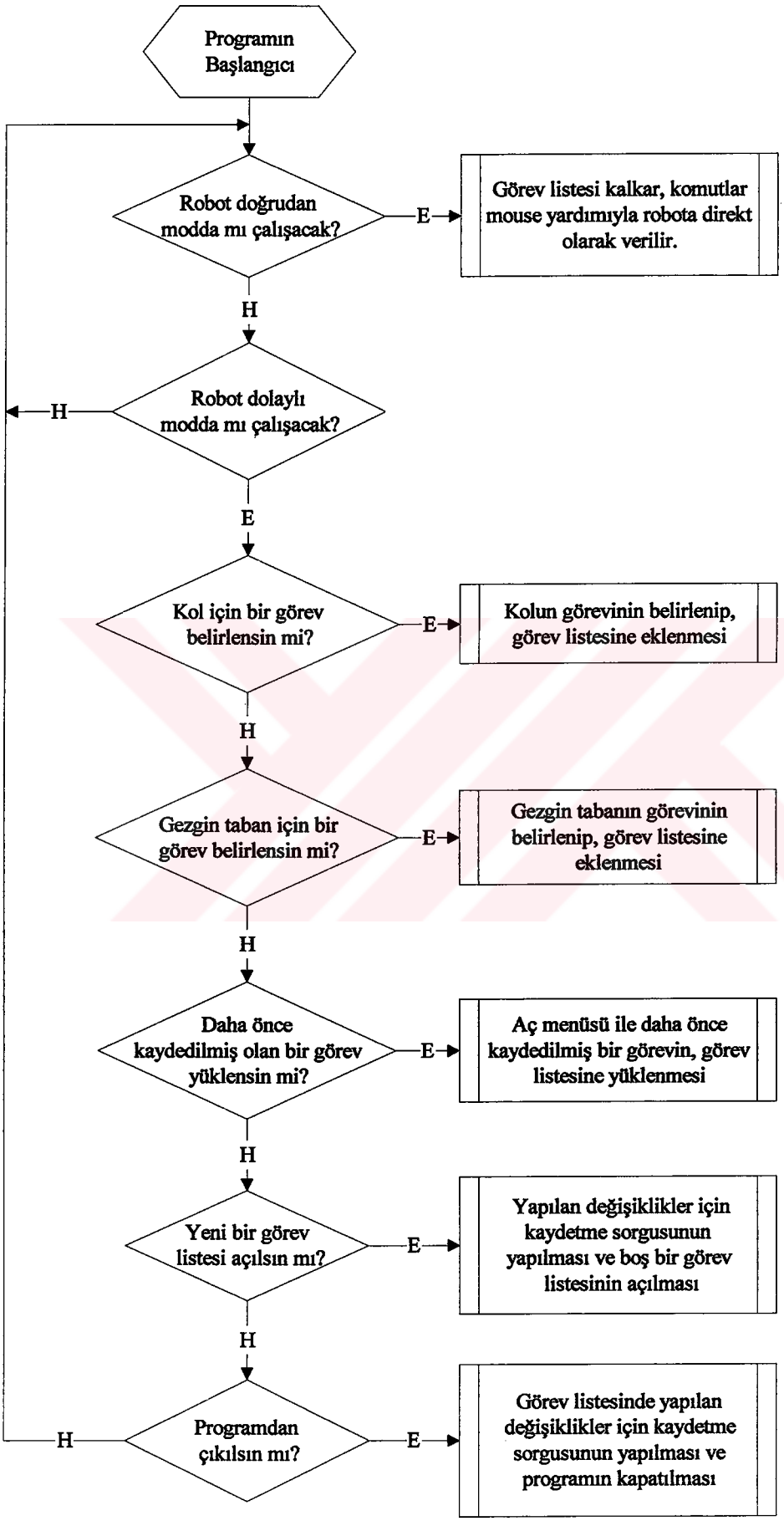
Şekil 4.14 Durum çubuğu

Programda, pencerenin en alt satırında, o sırada yapılan işlem hakkında bilgi veren bir satır bulunmaktadır. Bu satıra "Durum Çubuğu" adı verilir. Şekil 4.14'de programın durum çubuğu gösterilmiştir. Programımızda durum satırı üç parçadan oluşmaktadır. Birinci parça, görev listesindeki komutlar çalıştırılmaya başlandığında, robotun hangi görevi yaptığına dair bilgi vermektedir. İkinci parça, robotun hangi modda çalıştığını (Doğrudan Çalıştır Modu ve Kayıttan Çalıştır Modu) göstermektedir. Üçüncü parça ise, verinin hangi porttan iletildiğini gösterir. Şekil 4.15'de görülen radyo grubu ise robota bağlı altı adet dokunma algılayıcılarından gelen işaretleri kullanıcıya yansıtmak için kullanılır. Salt okunurdur.

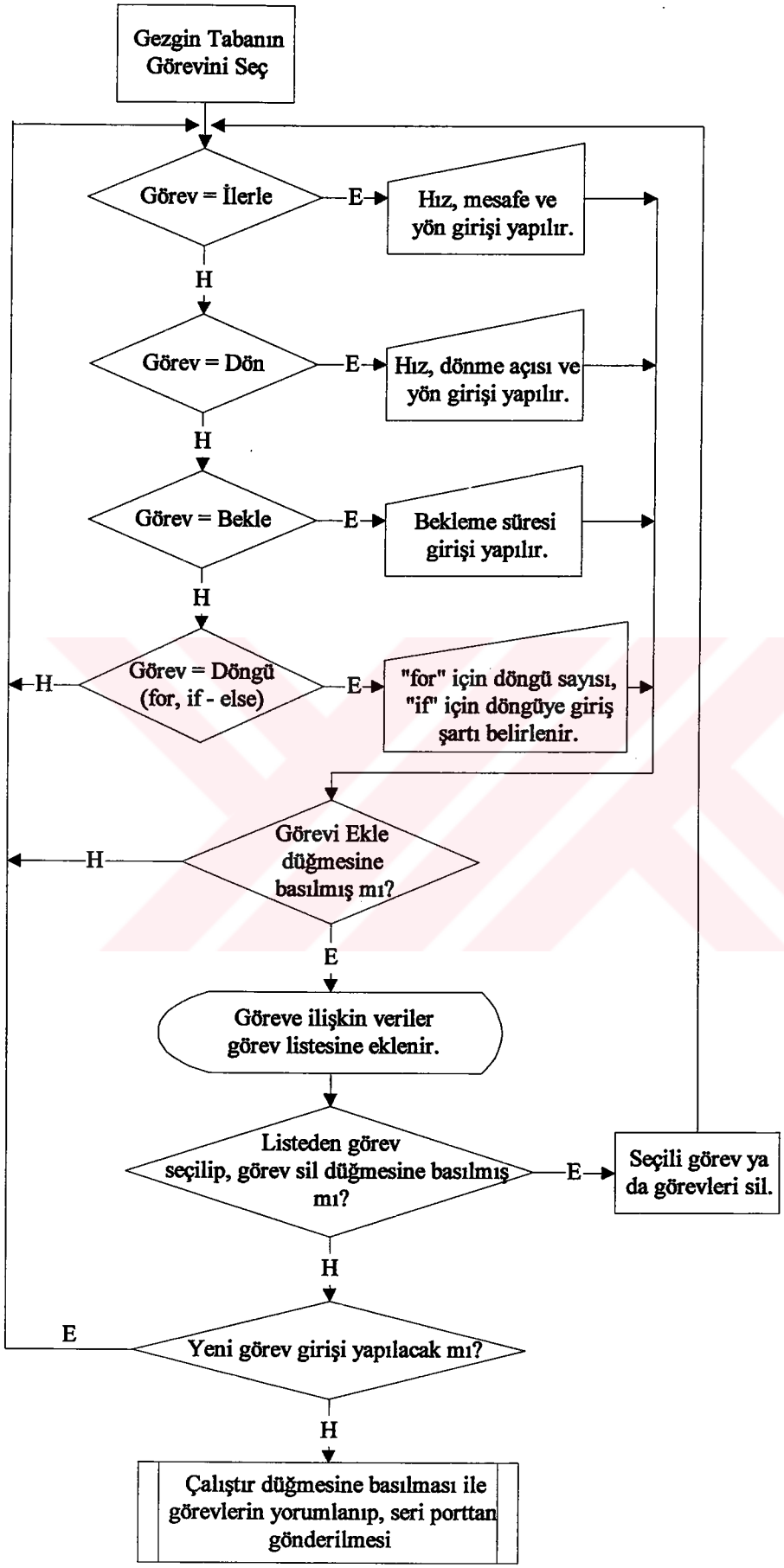


Şekil 4.15 Dokunma algılayıcısı durum göstergesi

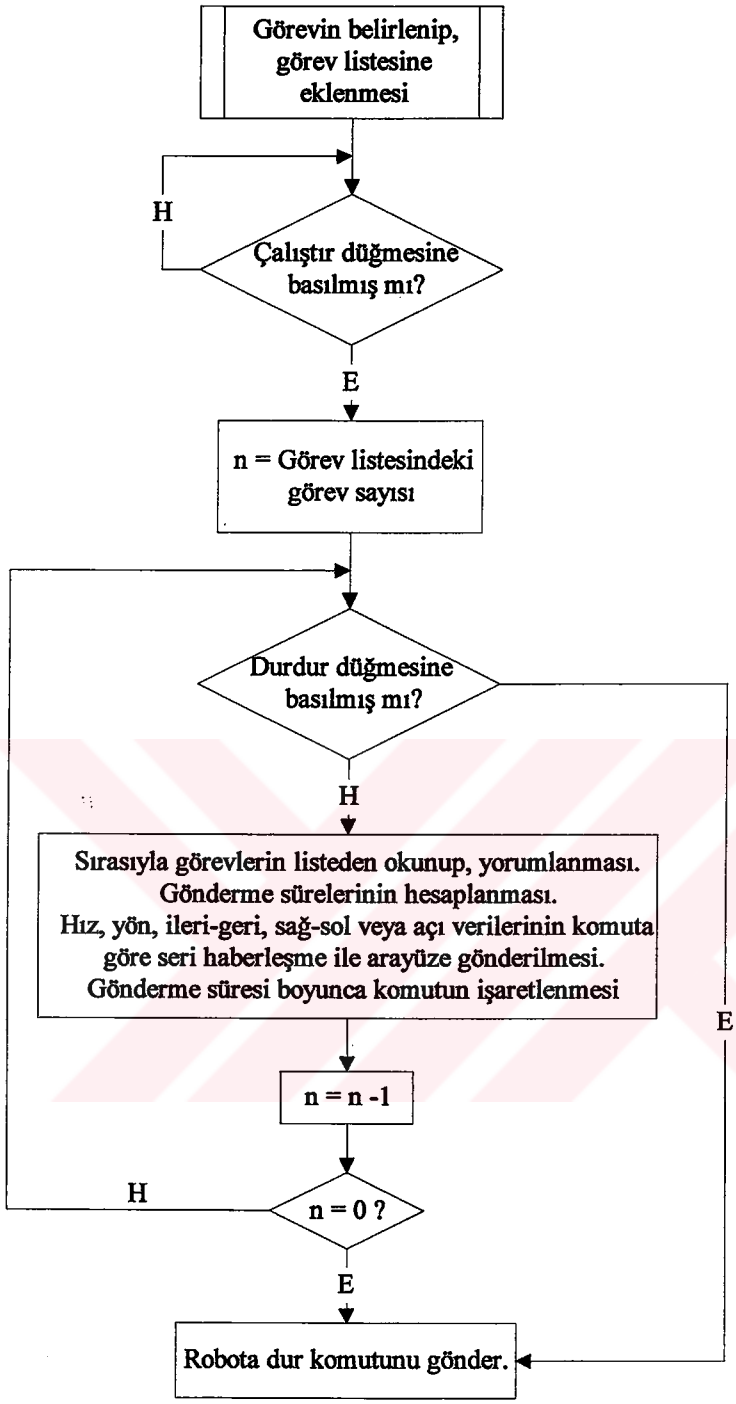
Programda Çalıştır düğmesine basıldığında, görev listesindeki komutlar robota gönderilir. Bunun için yazılan program, görev listesindeki verileri yorumlar. İlk önce listedeki metin halindeki bilgi, sayısal verilere dönüştürülür. Bunun için robotun kinematik denklemlerinden yararlanılır. Sonuçta komutlar başta anlatılan veri paketleri halinde denetim kartına gönderilir. Denetim kartı da aldığı verileri yorumlayıp, PWM işaretlere dönüştürerek motorları sürer. Denetim kartı ilk veriyi aldıktan sonra ikinci veri gelene kadar ilk veriyi işler. Komutların gönderilme süreleri PC tarafından belirlenir. Bunu için Delphi'de TTimer kontrolünün "Timer.Interval" özelliği kullanılmıştır. Görev listesindeki komutlar sıra ile gönderilir. İlk komut denetim kartına gönderildikten sonra, gönderilen göreve ait veriler ile TTimer kontrolü şartlanır (Timer.Interval süresi belirlenir.) ve ikinci görev bu süre dolduktan sonra gönderilir. Listedeki ilk komuta ait veri gönderildiğinde, listede o komut işaretlenir ve bir sonraki komuta ait veri gönderilinceye kadar işaretli kalır. Yani her komut kendine ait "Timer.Interval" süresi boyunca işaretli kalır. Bu işlem listedeki bütün komutlar bitinceye kadar sürer. Böylece kullanıcı, komutların iletimini kontrol edebilmektedir. Fakat kullanıcı istediğinde Durdur düğmesine basarak veri gönderme işlemini, dolayısıyla robotu durdurabilmektedir. Programdan çıkmak içinse, "Dosya" menüsü içindeki "Çıkış" menüsü veya program penceresi üzerindeki "Çıkış" düğmesi kullanılmaktadır. Program kapanmadan önce görev listesindeki bilgiyi kaydetme sorgusunu yapmaktadır. Şekil 4.16'te programın akış diyagramı görülmektedir.


Programın Başlangıcı
Robot doğrudan modda mı çalışacak?
E
Görev listesi kalkar, komutlar mouse yardımıyla robota direkt olarak verilir.
H
Robot dolaylı modda mı çalışacak?
H
E
Kol için bir görev belirlensin mi?
E
Kolun görevinin belirlenip, görev listesine eklenmesi
H
Gezgin taban için bir görev belirlensin mi?
E
Gezgin tabanın görevinin belirlenip, görev listesine eklenmesi
H
Daha önce kaydedilmiş olan bir görev yüklensin mi?
E
Aç menüsü ile daha önce kaydedilmiş bir görevin, görev listesine yüklenmesi
H
Yeni bir görev listesi açılsın mı?
E
Yapılan değişiklikler için kaydetme sorgusunun yapılması ve boş bir görev listesinin açılması
H
Programdan çıkılsın mı?
H
E
Görev listesinde yapılan değişiklikler için kaydetme sorgusunun yapılması ve programın kapatılması

a)


Gezgin Tabanın Görevini Seç
Görev = İlerle
E
Hız, mesafe ve yön girişi yapılır.
H
Görev = Dön
E
Hız, dönme açısı ve yön girişi yapılır.
H
Görev = Bekle
E
Bekleme süresi girişi yapılır.
H
Görev = Döngü (for, if - else)
H
E
"for" için döngü sayısı, "if" için döngüye giriş şartı belirlenir.
Görevi Ekle düğmesine basılmış mı?
H
E
Göreve ilişkin veriler görev listesine eklenir.
Listeden görev seçilip, görev sil düğmesine basılmış mı?
E
Seçili görev ya da görevleri sil.
H
Yeni görev girişi yapılacak mı?
E
H
Çalıştır düğmesine basılması ile görevlerin yorumlanıp, seri porttan gönderilmesi

b)



c)

Şekil 4.16 Programın akış diyagramı

5. SONUÇLAR

Gezgin robotlar üzerine, özellikle son yıllarda yapılmış pek çok çalışma bulunmaktadır. Bu çalışmaların önemli bir bölümü yörünge planlama, engellerden sakınma, görev planlama gibi gezinme ağırlıklı çalışmalardır. Ayrıca algılayıcıların kullanımı, gelen verilerin değerlendirilmesi ve esnek sistem tasarımı konularında da pekçok çalışma yapılmıştır. Bu çalışmalar arasında pratik uygulamaya dökülmüş olanları da mevcuttur.

Yapılan çalışmalara bakıldığında planlama biriminin genelde bir PC olarak seçildiğini görmekteyiz. Bunun başlıca sebebi, PC'nin robot kontrol algoritmalarının geliştirmesinde büyük yazılım desteği sağlaması, kullanıcı için monitör, klavye ve mouse gibi birimlerle kullanması kolay ve bilindik bir ortam sunmasıdır. Engellerden sakınma, algılayıcı planlama, haritalama, konumlandırma ve benzeri pekçok işlevi sağlamak için adaptif yöntemler, bulanık mantık veya yapay sinir ağları gibi ileri algoritmaların uygulanmasına da imkan sağlamaktadır. Görüntü işleme gibi yoğun işlem gerektiren uygulamalarda da PC kullanılması kaçınılmazdır. Ayrıca PC, robot-kullanıcı ilişkisini sağlayacak bir görsel arabirim oluşturulmasına izin verir.

PC'nin getirdiği en önemli avantajlardan biri de maliyetin düşük olmasıdır. Hemen her yerde bir PC bulunduğu düşünülürse, bu tip robotlar için yalnız gezgin robotun kendisinin ve yazılımının alınması yeterli olacaktır. Aynı işi yapan ve kendi yüksek seviye kontrol birimini içeren bir gezgin robotun ise maliyeti diğerine göre çok daha fazla olacaktır.

Yalnız, PC'nin varlığı robotun gezginliğini kısıtlayan bir faktördür. PC'nin taşınabilir olması bu kısıtlamayı ortadan kaldırabilmektedir. Yeterince güçlü bir mekanik yapı ile dizüstü bilgisayarlar bu ihtiyacı karşılayabilir. Ancak bu da, güç tüketimini arttırması bakımından dezavantajlar taşır. PC104 kartları bu noktada iyi bir çözüm olabilir. Fakat, bu durumda da birimin kullanıcı ile ilişkisi sınırlı olmaktadır.

Tezde, kullanıcı hatalarına izin vermeyen yapısıyla kullanması kolay bir robot yazılımı geliştirilmiştir. Yazılımın en önemli özelliği ise, diferansiyel sürüşlü ve dümen mekanizmalı olmak üzere iki ayrı gezgin robotu tipini kontrol edebilmesidir. Programın görsel kullanıcı arabiriminde, altı ekleme kadar olan bir robot kolu kontrol edebilmek için gerekli birimlere de yer verilmiştir. Yazılım; gerekli altyapısı tamamlandığında, altı ekleme kadar olan robot kolları kontrol edebilecektir. Programın farklı modlara sahip olması kullanıcının, yazılımı kullanım amacını genişletmiştir.

Bu modlardan biri Kayıttan Çalışma Modu'dur. Burada kullanıcı kendi komut listesini oluşturabilmektedir. Böylece temel komutlar kullanılarak, robotun istenen bir görevi yerine getirmesi sağlanır. Ayrıca oluşturulan komut listesi kaydedilerek, robotun farklı işler için programlanabilmesi sağlanmaktadır.

Diğer mod ise, Doğrudan Çalışma Modu'dur. Burada komutlar robota anlık olarak verilmektedir. Robot, kullanıcının görsel arabirimdeki kontroller üzerinde yaptığı her değişikliğe hemen cevap verebilmektedir. Komutlar anlık olarak işlenir ve cevap alınır. Bu mod denetim kolu (joystick) kontrolüne benzemektedir.

İncelenen çalışmalarla geliştirilen gezgin robot programı karşılaştırıldığında, programın diğerleri gibi kullanıcı-robot ilişkisini sağlayan bir görsel kullanıcı arabirimine sahip olduğunu görülür. Ayrıca program gezgin tabanın sırayla komutları alıp işleyebileceği ve kullanıcının robotun o an hangi komutu işlediğini görüp, gerektiğinde müdahale edebileceği bir komut listesine sahiptir.

Araştırmalarda kontrol algoritmalarının üretilebilmesi için algılayıcıların sıklıkla kullanıldığı görülmüştür. Dokunma algılayıcıları da basit kullanımları nedeniyle çalışmaların hemen hepsinde yer almıştır. Geliştirilen yazılım da diğer çalışmalarda olduğu gibi dokunma algılayıcılar için algılayıcı kontrolü yapmaktadır.

Gelinen aşamada yazılımın; çoklu robot kontrolü, görüntü işleme ve akıllı gezgin robot aktiviteleri desteği yoktur. Fakat, geliştirilen program ilerde bu konularda yapılabilecek çalışmalar için iyi bir temel oluşturmaktadır. İlerleyen safhalarda programa eklenecek, farklı kullanım alanları olan algılayıcılar sayesinde yeni kontrol algoritmaları geliştirilebilecektir.

KAYNAKLAR

Aksun, B., Mobile Robots, M. Sc. Thesis, January 1996.

Craig, J. J., (1989), Introduction to Robotics, Addison-Wesley Publishing Company, Inc.

Critchlow, A. J., (1985) Introduction to Robotics, Macmillan Publishing Company, New York, Collier Macmillan Publishers, London.

Everett, H. R., (1995), Sensors for Mobile Robots : Theory and Application, A K Peters Ltd.

Güvercin, A., (1999), Bir DC Motorun Hız, Devir Yönü, Akım, ve Sıcaklık Parametrelerinin RS232 Arabirimi ve Telefon Hatları Aracılığıyla Kontrolü ve İzlenmesi, Yüksek Lisans Tezi, YTÜ Fen Bilimleri Enstitüsü.

Karagülle, İ., Pala, Z. (1996), Borland Delphi 16 & 32 Bit ile Visual Programlama, Türkmen Kitabevi.

Kaynak, O., "Mekatronik Kavramı ve Uygulama Alanları", Otomasyon Dergisi, Kasım 1998, 77: 40-45.

Klafter, R. D., Chmielewski, T. A. ve Negin, M. (1989), Robotic Engineering An Integrated Approach, Prentice Hall, Englewood Cliffs, New Jersey 07632.

Matcho, J., Salmanowitz, B., Strool, S., Biely, B., Jurkouich, S. T., Berry, S., Sleeper, L., Dumbrill, D., Uber, E., (1996), Using Delphi 2, Que Corporation, Indiana.

Rajagopalan, R., (1997), "A Generic Kinematic Formulation For Wheeled Mobile Robots", Journal of Robotic Systems, 1997, 14(2): 77-91.

Sankur, B., İstefanopulos, Y., Elektrik - Elektronik Bilgisayar Mühendisliği Terimleri Sözlüğü, Boğaziçi Üniversitesi Yayınları, Mart 1997.

Sunay, Ç., "21. Yüzyılın Robotları, Androidler", Bilim ve Teknik Dergisi, Ağustos 1998, 369: 34-41.

Yanık, M., Borland Delphi ile Görsel Programcılık, Sistem Yayıncılık, Haziran 1996.

EKLER

Ek 1 Robot Programı Yazılım Dökümü

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Program menüsünden haberleşme hızının seçimi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.Baud1200Click(Sender: TObject);
begin
   SetCheck(Sender);
   Baud1200.Checked:= True;
   Baud2400.Checked:= False;
   Baud9600.Checked:= False;
   Baud14400.Checked:= False;
   habhizi:= 1200;
   Haberhizi(Sender);
end;
procedure TForm1.Baud2400Click(Sender: TObject);
begin
   SetCheck(Sender);
   Baud1200.Checked:= False;
   Baud2400.Checked:= True;
   Baud9600.Checked:= False;
   Baud14400.Checked:= False;
   habhizi:= 2400;
   Haberhizi(Sender);
end;

procedure TForm1.Baud9600Click(Sender: TObject);
begin
   SetCheck(Sender);
   Baud1200.Checked:= False;
   Baud2400.Checked:= False;
   Baud9600.Checked:= True;
   Baud14400.Checked:= False;
   habhizi:= 9600;
   Haberhizi(Sender);
end;

procedure TForm1.Baud14400Click(Sender: TObject);
begin
   SetCheck(Sender);
   Baud1200.Checked:= False;
   Baud2400.Checked:= False;
   Baud9600.Checked:= False;
   Baud14400.Checked:= True;
   habhizi:= 14400;
   Haberhizi(Sender);
end;
```

```
procedure TForm1.Haberhizi(Sender: TObject);
var habhizistr: string;
begin
   habhizistr:=IntToStr(habhizi);
   MSComm1.Settings:= habhizistr + ',n,8,1';
end;
```

* *

Programı ilk açılış durumuna getiren program;

* *

```
procedure TForm1.Initiate(Sender: TObject);
begin
   Listboxflag:= False;
   if DegerlerSifir = False then
      ListBox.Items.Clear;
   DegerlerSifir:= False;
   Bekledonilerle.PageIndex:= -1;
   Yaz7.Text:= '0';
   Yaz8.Text:= '0';
   Yaz9.Text:= '0';
   Yaz10.Text:= '46';
   foryaz.Text:= '';
   TrackBar7.Position:= 0;
   TrackBar8.Position:= 0;
   TrackBar10.Position:= 0;
   Sagsol.Itemindex:= -1;
   Geriileri.Itemindex:= -1;
   IleriGeri.Itemindex:= -1;
   Check1.Checked:= False;
   Check2.Checked:= False;
   Check3.Checked:= False;
   Check4.Checked:= False;
   Check5.Checked:= False;
   Check6.Checked:= False;
   Check(Sender);
   CalisButton.Enabled:= False;
   dizino:= 0;
   Form1.Caption:= 'ROBOT PROGRAMI - Adsız.dur'
end;
```

* *

Portun açılıp, kapatılması ve port seçimi;

* *

```
procedure TForm1.portopen;
begin
   MSComm1.RThreshold:= 1;
   MSComm1.SThreshold:= 0;
   MSComm1.CommPort:= 2;
```

```
    portsecim.ItemIndex:= 1;
    MSComm1.Settings:= '9600,n,8,1';
    MSComm1.OutBufferSize:= 512;
    MSComm1.Handshaking:= ComNone;
    MSComm1.PortOpen:= True;
    alinanveri:='';
end;

procedure TForm1.portclose;
begin
    MSComm1.PortOpen:= False;
end;
procedure TForm1.ComboChange(Sender: TObject);
begin
if MSComm1.CommPort <> (PortSecim.ItemIndex+1) then
    begin
    MSComm1.PortOpen:= False;
    MSComm1.CommPort:= PortSecim.ItemIndex + 1;
    StatusBar1.Panels[2].Text:= PortSecim.Items.Strings[PortSecim.ItemIndex];
    MSComm1.PortOpen:= True;
    end;
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Program menüsünden robot modelinin (diferansiyel sürüşlü robot, dümen mekanizmalı robot)
seçimi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.DifRobotClick(Sender: TObject);
begin
    YeniClick(Sender);
    SetCheck(Sender);
    DifRobot.Checked:= True;
    DumRobot.Checked:= False;
    YarLabPanel.Visible:= False;
    YarPanel.Visible:= False;
    GeriileriPanel.Visible:= False;
    Form1.Caption:= 'ROBOT PROGRAMI - Adsız.dir';
    Open:= OpenDialogDif.Filename;
    Save:= SaveDialogDif.Filename;
end;

procedure TForm1.DumRobotClick(Sender: TObject);
begin
    YeniClick(Sender);
    SetCheck(Sender);
    DifRobot.Checked:= False;
    DumRobot.Checked:= True;
    YarLabPanel.Visible:= True;
    YarPanel.Visible:= True;
    GeriileriPanel.Visible:= True;
```

```
    Form1.Caption:= 'ROBOT PROGRAMI - Adsız.dur';
    Open:= OpenDialogDum.Filename;
    Save:= SaveDialogDum.Filename;
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Program menüsünden Kayıttan Çalışma Modunun seçimi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.NormalModClick(Sender: TObject);
begin
    SetCheck(Sender);
    Kayittan.Checked:= True;
    SurekliMod.Checked:= False;
    Dogrudan.Checked:= False;
    Initiate(Sender);
    TabGorEkPanel.Visible:= True;
    KolGorEkPanel.Visible:= True;
    GorListPanel.Visible:= True;
    YarLabPanel.Visible:= True;
    AciLabPanel.Visible:= True;
    SureLabPanel.Visible:= True;
    SurePanel.Visible:= True;
    YarPanel.Visible:= True;
    Yaz8.Visible:= True;
    AciPanel.Visible:= True;
    KonumPanel.Caption:= 'm  ';
    StatusBar1.Panels[1].Text:= 'Robot kayıttan ve normal çalışma modunda...';
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Gezgin tabanın "bekle", "dön" veya "ilerle" görevlerinden birinin görev listesine eklenmesi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.GezTabGorevEkleClick(Sender: TObject);
var
saha1, saha2: string;
a, b, n, code, code1, uzun, sec: integer;
begin
    Listboxflag:=True;
    saha1:= IntToStr(TrackBar8.Position);
    saha2:= IntToStr(TrackBar10.Position);
    if bekledonilerle.PageIndex = 2 then
        begin
        Val(Yaz9.Text, a, code);
        if a = 0 then
            ShowMessage('Kaç saniye bekleyecek?')
        else
            begin
            if Listbox.SelCount = 1 then
```

```
        begin
        for sec:= (Listbox.Items.Count - 1) downto 0 do
          if listbox.Selected[sec] then
            if (strpos(pchar(Listbox.Items.Strings[sec]),'begin')) <> nil then
              ShowMessage('Döngü ile begin arasına görev eklenemez!')
            else ListBox.Items.Insert(sec,Yaz9.Text+'sn'+' bekle');
        end
      else ListBox.Items.Add(Yaz9.Text+'sn'+' bekle')
      end
    end

else if bekledonilerle.PageIndex = 1 then
    begin
    if DumRobot.Checked = True then
      begin
      Val(Yaz10.Text, a, code);
      ss:= '';
      gi:= '';
      if Sagsol.ItemIndex = 0 then
        ss:= 'Sağa'
      else if Sagsol.ItemIndex = 1 then
        ss:= 'Sola'
      else
        ShowMessage('Sağ mı, sol mu?');
      if Geriileri.ItemIndex = 0 then
        gi:= 'İleri'
      else if Geriileri.ItemIndex = 1 then
        gi:= 'Geri'
      else
        ShowMessage('İleri mi, geri mi?');
      if saha2 = '0' then
        ShowMessage('Hız sıfırdan farklı olmalı!');
      if a < 46 then
        begin
        ShowMessage('Yarıçap 46"ten küçük olamaz!');
        Yaz10.Text:= '46';
        end
      else if a > 300 then
        begin
        ShowMessage('Yarıçap 300""den büyük olamaz!');
        Yaz10.Text:= '300';
        end;
      if ss <> '' then
        if gi <> '' then
          if saha2 <> '0' then
            begin
            Val(Yaz10.Text, a, code);
            Val(Yaz7.Text, b, code1);
            if b = 0 then
              ShowMessage('Dönme açısı sıfırdan farklı olmalı!')
            else
```

```
            begin
            if Listbox.SelCount = 1 then
               begin
               for sec:= (Listbox.Items.Count - 1) downto 0 do
                  if listbox.Selected[sec] then
                     if (strpos(pchar(Listbox.Items.Strings[sec]),'begin')) <> nil then
                        ShowMessage('Döngü ile begin arasına görev eklenemez!')
                     else
                        ListBox.Items.Insert(sec,ss+' '+'ve'+' '+gi+'ye'+' Dön'+',
                        '+'Hız='+saha2+', '+'Dönme Açısı='+IntToStr(b)+'°'+',
                        '+'Yarıçap='+IntToStr(a)+'cm');
               end
            else
               ListBox.Items.Add(ss+' '+'ve'+' '+gi+'ye'+' Dön'+', '+'Hız='+saha2+',
               '+'Dönme Açısı='+IntToStr(b)+'°'+', '+'Yarıçap='+IntToStr(a)+'cm')
            end;
         end;
   end

else if DifRobot.Checked = True then
   begin
   ss:= '';
   if Sagsol.ItemIndex = 0 then
      ss:= 'Sağa'
   else if Sagsol.ItemIndex = 1 then
      ss:= 'Sola'
   else
      ShowMessage('Sağ mı, sol mu?');
   if saha2 = '0' then
      ShowMessage('Hız sıfırdan farklı olmalı!');
   if ss <> '' then
      if saha2 <> '0' then
         begin
         Val(Yaz7.Text, a, code);
         if a = 0 then
            ShowMessage('Dönme açısı sıfırdan farklı olmalı!')
         else
            begin
            if Listbox.SelCount = 1 then
               begin
               for sec:= (Listbox.Items.Count - 1) downto 0 do
                  if listbox.Selected[sec] then
                     ListBox.Items.Insert(sec,ss+' Dön'+', '+'Hız='+saha2+', '+'Dönme
                     Açısı='+IntToStr(a)+'°');
               end
            else
               ListBox.Items.Add(ss+' Dön'+', '+'Hız='+saha2+', '+'Dönme
               Açısı='+IntToStr(a)+'°')
            end;
         end;
   end
```

```
      end
   else if bekledonilerle.PageIndex = 0 then
      begin
      ig:= '';
      if Ilerigeri.ItemIndex = 0 then
         ig:='İleri'
      else if Ilerigeri.ItemIndex = 1 then
         ig:='Geri'
      else
         ShowMessage('İleri mi, geri mi?');
      if saha1 = '0' then
         ShowMessage('Hız sıfırdan farklı olmalı!');
      if ig <> '' then
         if saha1 <> '0' then
            begin
            Val(Yaz8.Text, a, code);
            if a = 0 then
               ShowMessage('Mesafe sıfırdan farklı olmalı!')
            else
               begin
               if Listbox.SelCount = 1 then
                  begin
                  for sec:= (Listbox.Items.Count - 1) downto 0 do
                     if listbox.Selected[sec] then
                        if (strpos(pchar(Listbox.Items.Strings[sec]),'begin')) <> nil then
                           ShowMessage('Döngü ile begin arasına görev eklenemez!')
                        else
                           ListBox.Items.Insert(sec,ig+' Git'+', '+'Hız='+saha1+',
                           '+'Mesafe='+Yaz8.Text+'cm');
                  end
               else

                  ListBox.Items.Add(ig+' Git'+', '+'Hız='+saha1+',
                  '+'Mesafe='+Yaz8.Text+'cm');
               end;
            end;
      end;
   if ListBox.Items.Text <> '' then
      CalisButton.Enabled:= True;
      Calistir.Enabled:= True;
      Listbox.ItemIndex:= -1;
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
"for" döngüsünün görev listesine eklenmesi, döngünün sayısının belirlenmesi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.forClick(Sender: TObject);
var sec:integer;
begin
   Listboxflag:=True;
```

```
    if foryaz.Text <> '' then
      if foryaz.Text <> '0' then
        if Listbox.SelCount = 1 then
          begin
          for sec:= (Listbox.Items.Count-1) downto 0 do
            if Listbox.Selected[sec] then
              if (strpos(pchar(Listbox.Items.Strings[sec]),'begin')) <> nil then
                ShowMessage('Döngü ile begin arasına görev eklenemez!')
              else
                begin
                Listbox.Items.Insert(sec,'begin');
                ListBox.Items.Insert(sec,'for i = 1 to '+ foryaz.Text);
                end;
          end
        else
          begin
          Listbox.Items.Add('for i = 1 to '+ foryaz.Text);
          Listbox.Items.Add('begin');
          end;
end;

procedure TForm1.foryazChange(Sender: TObject);
var i:integer; s:string;
begin
   s:=foryaz.Text;
   for i:=1 to length(s) do
      if not (s[i]in['0'..'9']) then System.Delete(s,i,1);
   foryaz.Text:= s;
end;
```

* *

"if" döngüsünün, döngü içine girme durumuyla birlikte görev listesine eklenmesi ve "else" döngüsünün görev listesine eklenmesi;

* *

```
procedure TForm1.ifbtnClick(Sender: TObject);
var sec:integer;
begin
   Listboxflag:= True;
   if engel.ItemIndex <> -1 then
      if Listbox.SelCount = 1 then
        begin
        for sec:= (Listbox.Items.Count-1) downto 0 do
          if Listbox.Selected[sec] then
            if (strpos(pchar(Listbox.Items.Strings[sec]),'begin')) <> nil then
              ShowMessage('Döngü ile begin arasına görev eklenemez!')
            else
              begin
              Listbox.Items.Insert(sec,'begin');
              ListBox.Items.Insert(sec,'if '+ engel.Items.Strings[engel.ItemIndex]+' engel
              varsa then');
```

```
                end;
        end
    else
        begin
        Listbox.Items.Add('if '+ engel.Items.Strings[engel.ItemIndex] +' engel varsa then');
        Listbox.Items.Add('begin');
        end;
end;
```

* *

"else" döngüsünün görev listesine eklenmesi;

* *

```
procedure TForm1.elsebtnClick(Sender: TObject);
var sec:integer;
begin
   Listboxflag:=True;
   if Listbox.SelCount = 1 then
      begin
      for sec:=(Listbox.Items.Count-1) downto 0 do
         if listbox.Selected[sec] then
            if (strpos(pchar(Listbox.Items.Strings[sec]),'begin'))<> nil then
               ShowMessage('Döngü ile begin arasına görev eklenemez!')
            else
               begin
               Listbox.Items.Insert(sec,'begin');
               ListBox.Items.Insert(sec,'else');
               end;
      end
   else
      begin
      Listbox.Items.Add('else');
      Listbox.Items.Add('begin');
      end;
end;
```

* *

Görev listesindeki "bekle", "dön", "ilerle" veya "döngü" komutlarının yorumlanıp denetim kartına gönderilebilecek hale getirilmesi;

* *

```
procedure TForm1.Gorevyorumla;
var
r:Real;
n, code, i, don, carpan:integer;
dummy, dummy2, str: string;
konum,aci,sure,hiz, bul, bak:integer;
hesap:real;
olayolmus:boolean;
a, b: word;
```

```
ilerigit, gerigit:integer;
begin
   r:=0.1;
   olayolmus:= False;
   n:=sirano-1;
   dummy:=ListBox.Items[n];

   if strpos(pchar(dummy),'bekle') <> nil then
      begin
      gorevlist[n+1].gorev:= bekle;
      dummy:=copy(dummy,1,pos('sn',dummy)-1);
      val(dummy,sure,code);
      gorevlist[n+1].gondermesuresi:= sure*1000;
      gorevlist[n+1].ayar:= 0;
      StatusBar1.Panels[0].Text:= 'Robot bekliyor...';
      end;

   if strpos(pchar(dummy),'Dön') <> nil then
      begin
      if DumRobot.Checked = True then
         begin
         if strpos(pchar(dummy),'Sağa') <> nil then
            gorevlist[n+1].gorev:= dumsagadon
         else gorevlist[n+1].gorev:= dumsoladon;
            dummy:=copy(dummy,9,length(dummy));
         if strpos(pchar(dummy),'İleriye')<> nil then
            gorevlist[n+1].yon:= ileridon
         else if strpos(pchar(dummy),'Geriye') <> nil then
            gorevlist[n+1].yon:= geridon;
         if strpos(pchar(dummy),'Hız') <> nil then
            begin
            dummy2:= copy(dummy,pos('z=',dummy)+2,2);
            val(dummy2,hiz,code);
            gorevlist[n+1].ayar:= gorevlist[n+1].yon + (hiz*16);
            end;
         if strpos(pchar(dummy),'Dönme Açısı=') <> nil then
            begin
            dummy2:= copy(dummy,pos('ı=',dummy)+2,3);
            val(dummy2,aci,code);
            end;
         if strpos(pchar(dummy),'Yarıçap=')<> nil then
            begin
            dummy2:= copy(dummy,pos('p=',dummy)+2,3);
            val(dummy2,yarcap,code);
            end;

         gorevlist[n+1].gondermesuresi:= Round((2*pi*(yarcap/100)*aci*carpan/(360*hiz)));
         tekeracisi:= ArcTan(26/yarcap)*(180/pi);
         servoacisi:= gorevlist[n+1].gorev*3* round(tekeracisi);
         donkomutu:= round((750+(90 - servoacisi)*(1625/180))/4)*64;
         a:= (donkomutu) and (255);
```

```
        b:= (donkomutu) and (65280);
        a:= (a) shr (2);
        donkomutu:= a+b;
        StatusBar1.Panels[0].Text:= 'Robot dönüyor...';
        end

    else if DifRobot.Checked = True then
        begin
        if strpos(pchar(dummy),'Sağa') <> nil then
            gorevlist[n+1].gorev:= difsagadon
        else gorevlist[n+1].gorev:= difsoladon;
            dummy:=copy(dummy,11,length(dummy));
        if strpos(pchar(dummy),'Hız') <> nil then
            begin
            dummy2:= copy(dummy,5,pos(',',dummy)-5);
            val(dummy2,hiz,code);
            gorevlist[n+1].ayar:= gorevlist[n+1].gorev + (hiz*16);
            end;
        if strpos(pchar(dummy),'Dönme Açısı=') <> nil then
            begin
            dummy:= copy(dummy,pos('ı=',dummy)+2,3);
            val(dummy,aci,code);
            end;
        gorevlist[n+1].gondermesuresi:= Round((pi*r*r*aci*carpan/(360*hiz*k)));
        StatusBar1.Panels[0].Text:= 'Robot dönüyor...';
        end;
    end;

if strpos(pchar(dummy),'Git') <> nil then
    begin
    if DumRobot.Checked = True then
        begin
        ilerigit:= dumilerigit;
        gerigit:= dumgerigit;
        end
    else if DifRobot.Checked = True then
        begin
        ilerigit:= difilerigit;
        gerigit:= difgerigit;
        end;
    if strpos(pchar(dummy),'İleri') <> nil then
        gorevlist[n+1].gorev:= ilerigit
    else gorevlist[n+1].gorev:= gerigit;
    dummy:= copy(dummy,pos('H',dummy),length(dummy));
    if strpos(pchar(dummy),'Hız') <> nil then
        begin
        dummy2:= copy(dummy,5,pos(',',dummy)-5);
        val(dummy2,hiz,code);
        {gorevlist[n+1].hiz:= HizAyarlari(hiz); }
        gorevlist[n+1].ayar:= gorevlist[n+1].gorev + (hiz*16);
        end;
```

```
    if strpos(pchar(dummy),'Mesafe=') <> nil then
        begin
        dummy2:= copy(dummy,pos('e=',dummy)+2,6);
        val(dummy2,konum,code);
        end;
    gorevlist[n+1].gondermesuresi:= Round((Konum/(100*hiz))*k*carpan);
    servoacisi:= 0;
    StatusBar1.Panels[0].Text:= 'Robot ilerliyor...';
    end;

if strpos(pchar(dummy),'for') <> nil then
    begin
    dummy:= copy(dummy,pos('to ',dummy)+3,5);
    dizino:= dizino + 1;
    val(dummy,fordongu[dizino].defa,code);
    sirano:= sirano + 2;
    fordongu[dizino].baslano:= sirano;
    GorevYorumla;
    exit;
    end;

if strpos(pchar(dummy),'if') <> nil then
    begin
        if (onde.Checked = True) or (onsag.Checked = True) or (onsol.Checked = True) or
        (arkada.Checked = True) or (arkasag.Checked = True) or
        (arkasol.Checked = True) then
            begin
            dummy:= copy(dummy,pos('if ',dummy)+3,pos(' engel',dummy)-4);
            if dummy = onde.Caption then
                if onde.Checked = True then
                    begin
                    olayolmus:= True;
                    sirano:= sirano + 2;
                    gorevyorumla;
                    exit;
                    end
                else olayolmus:= False
            else if dummy = onsag.Caption then
                if onsag.Checked = True then
                    begin
                    olayolmus:= True;
                    sirano:= sirano + 2;
                    gorevyorumla;
                    exit;
                    end
                else olayolmus:= False
            else if dummy = onsol.Caption then
                if onsol.Checked = True then
                    begin
                    olayolmus:= True;
                    sirano:= sirano + 2;
```

```
            gorevyorumla;
            exit;
            end
        else olayolmus:= False
    else if dummy = arkada.Caption then
        if arkada.Checked = True then
            begin
            olayolmus:= True;
            sirano:= sirano + 2;
            gorevyorumla;
            exit;
            end
        else olayolmus:= False
    else if dummy = arkasag.Caption then
        if arkasag.Checked = True then
            begin
            olayolmus:= True;
            sirano:= sirano + 2;
            gorevyorumla;
            exit;
            end
        else olayolmus:= False
    else if dummy = arkasol.Caption then
        if arkasol.Checked = True then
            begin
            olayolmus:= True;
            sirano:= sirano + 2;
            gorevyorumla;
            exit;
            end
        else olayolmus:= False
    end
else olayolmus:= False;

if olayolmus = False then
    begin
    olayolmus:= False;
    i:= 1;
    while endayarla[i].iffor <> (sirano-1) do
        i:= i + 1;
    sirano:= (endayarla[i].endbul + 2);
    if strpos(pchar(Listbox.Items.Strings[sirano-1]),'else') <> nil then
        begin
        sirano:= sirano + 2;
        gorevyorumla;
        exit
        end
    else
        begin
        gorevyorumla;
        exit;
```

```
            end;
        end;
    end;

if strpos(pchar(dummy),'end') <> nil then
    begin
    i:= 1;
    while endayarla[i].endbul <> (sirano-1) do
        i:= i + 1;
    don:= endayarla[i].iffor;
    if (strpos(pchar(Listbox.Items.Strings[don]),'if')) <> nil then
        if strpos(pchar(Listbox.Items.Strings[sirano]),'else') <> nil then
            begin
            i:=1;
            while  endayarla[i].iffor <> (sirano) do
                i:= i + 1;
            sirano:= (endayarla[i].endbul+2);
            if sirano <> (ListBox.Items.Count+1) then
                begin
                gorevyorumla;
                exit;
                end;
            exit;
            end
        else
            begin
            sirano:= sirano + 1;
            if sirano <> (ListBox.Items.Count+1) then
                begin
                gorevyorumla;
                exit;
                end;
            exit;
            end;

    if (strpos(pchar(Listbox.Items.Strings[don]),'else')) <> nil then
        begin
        sirano:= sirano + 1;
        if sirano <> (ListBox.Items.Count+1) then
            begin
            gorevyorumla;
            exit;
            end;
        exit;
        end;

    if (strpos(pchar(Listbox.Items.Strings[don]),'for')) <> nil then
        begin
        fordongu[dizino].defa:= fordongu[dizino].defa - 1;
        if fordongu[dizino].defa <> 0 then
            begin
```

```
            sirano:= fordongu[dizino].baslano;
            GorevYorumla;
            exit;
            end
         else
            begin
            sirano:= sirano + 1;
            dizino:= dizino - 1;
            if sirano <> (ListBox.Items.Count+1) then
               begin
               gorevyorumla;
               exit;
               end;
            end;
         end;
      end;
end;
```

* *

Görev listesindeki "end" satırlarının hangi döngü komutuna ("for", "if", "else") ait olduğunu bulan program. Programdaki "end" ve "begin" sayılarını karşılaştıran program;

* *

```
procedure TForm1.endleribul(Sender: TObject);
begin
   eslestir:= 1;
   for bul:= 0 to (ListBox.Items.Count - 1) do
      if (strpos(pchar(Listbox.Items.Strings[bul]),'end')) <> nil then
         begin
         endayarla[eslestir].endbul:= bul;
         ifforbul(Sender);
         end;
end;

procedure TForm1.ifforbul(Sender: TObject);
var
a: string;
bul1, i: integer;
mevcut: Boolean;
begin
   for bul1:= bul downto 0 do
      begin
      mevcut:= False;
      a:= Listbox.Items.Strings[bul1];
      if (strpos(pchar(a),'for')) or (strpos(pchar(a),'if')) or (strpos(pchar(a),'else')) <> nil then
         begin
         if eslestir > 1 then
            for i:= 1 to (eslestir-1) do
               if endayarla[i].iffor = bul1 then
                  mevcut:= True;
```

```
            if mevcut = False then
               begin
               endayarla[eslestir].iffor:= bul1;
               eslestir:= eslestir+1;
               exit
               end;
            end;
         end;
end;

procedure TForm1.Kural(Sender: TObject);
var a:integer;
begin
   beginsayisi:= 0;
   endsayisi:= 0;
   ifsayisi:= 0;
   elsesayisi:= 0;
   for a:= 0 to (ListBox.Items.Count - 1) do
      begin
      if (strpos(pchar(Listbox.Items.Strings[a]),'begin')) <> nil then
         beginsayisi:= beginsayisi + 1;
      if (strpos(pchar(Listbox.Items.Strings[a]),'end')) <> nil then
         endsayisi:= endsayisi + 1;
      if (strpos(pchar(Listbox.Items.Strings[a]),'if')) <> nil then
         ifsayisi:= ifsayisi + 1;
      if (strpos(pchar(Listbox.Items.Strings[a]),'else')) <> nil then
         elsesayisi:= elsesayisi + 1;
      end;
   if beginsayisi > endsayisi then
      ShowMessage('Programda end eksik!')
   else if beginsayisi < endsayisi then
      ShowMessage('Programda fazla end var!');
   if ifsayisi < elsesayisi then
      ShowMessage('Programda fazla else var!');
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Çalıştır menüsüne basıldığında, çağrılan program;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.CalisButtonClick(Sender: TObject);
var
d1, hour, min, sec, msec, msec2: word;
begin
   veriyok:= True;
   d1:=0;
   repeat
      if (d1 mod 15) = 0 then
         MSComm1.Output:= chr(255);
      inc(d1);
      DecodeTime(Time,Hour, Min, Sec, MSec);
```

```
      msec2:= msec;
      repeat
         DecodeTime(Time,Hour, Min, Sec, MSec);
      until msec2 <> msec;
      verial(Sender);
   until (veriyok = False) or (d1 >= 45);

   if veriyok = True then
      begin
      StatusBar1.Panels[0].Text:= 'Haberleşme yok';
      DurdurButtonClick(Sender);
      exit;
      end;
   gorevsira:= ListBox.Items.Count;
   Kural(Sender);
   endleribul(Form1);
   if (beginsayisi <> endsayisi) or (ifsayisi < elsesayisi) then
      exit;
   sirano:=1;
   Timer1.Enabled:= True;
   p:=0;
   btnDelete.Enabled:= False;
   ListBox.MultiSelect:= False;
   DurdurButton.Enabled:= True;
   Durdur.Enabled:= True;
   Calistir.Enabled:= False;
   CalisButton.Enabled:= False;
   RobModel.Enabled:= False;
   Habhiz.Enabled:= False;
   GezTabGorevEkle.Enabled:= False;
   KolGorevEkle.Enabled:= False;
   Timer1Timer(Sender);
end;
```

* *

Timer'ın şartlanması ve görevyorumla prosedüründe elde edilen verilerin seri porttan denetim kartına karakterler halinde gönderilmesi;

* *

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
d: integer;
a: byte;
begin
   if sirano <> (ListBox.Items.Count+1) then
      begin
      ListBox.Enabled:= False;
      btnDelete.Enabled:= False;
      GezTabGorevEkle.Enabled:= False;
      gorevyorumla;
      if sirano = (ListBox.Items.Count+1) then
```

```
        begin
        DurdurButtonClick(Sender);
        exit;
        end;
      Timer1.Interval:= gorevlist[sirano].gondermesuresi;
      if gorevlist[sirano].gondermesuresi = 0 then
        begin
        ShowMessage('Programda hata var!');
        DurdurButtonClick(Sender);
        exit;
        end
      else
        begin
        if DumRobot.Checked = True then
          begin
          a:= (gorevlist[sirano].ayar) and (8);
          if a = 8 then
            begin
            MSComm1.Output:= chr(gorevlist[sirano].ayar);
            MSComm1.Output:= chr(hi(donkomutu));
            MSComm1.Output:= chr(lo(donkomutu));
            end
          else MSComm1.Output:= chr(gorevlist[sirano].ayar);
          end
        else if DifRobot.Checked = True then
          MSComm1.Output:= chr(gorevlist[sirano].ayar);
        ListBox.ItemIndex:= sirano-1;
        sirano:= sirano+1;
        end;
      end
    else
      DurdurButtonClick(Sender);
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Denetim kartından veri alan program;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.verial(Sender: TObject);
var
sayi: String;
sonuc: integer;
sayi1: Longint;
begin
   if veriyok = true then
      begin
      sayi:= MSComm1.Input;
      if strpos(pchar(sayi),chr(255)) <> nil then
        begin
        StatusBar1.Panels[0].Text:= 'Haberleşme var';
        veriyok:= False;
```

```
            exit;
            end;
        end;
    sayi:= MSComm1.Input;

    while sayi <> '' do
        begin
        sonuc:= sayi1 and 128;
        if sonuc = 128 then
            onde.Checked:= True;
        sonuc:= sayi1 and 64;
        if sonuc = 64 then
            onsag.Checked:= True;
        sonuc:= sayi1 and 32;
        if sonuc = 32 then
            onsol.Checked:= True;
        sonuc:= sayi1 and 16;
        if sonuc = 16 then
            arkada.Checked:= True;
        sonuc:= sayi1 and 8;
        if sonuc = 8 then
            arkasag.Checked:= True;
        sonuc:= sayi1 and 4;
        if sonuc = 4 then
            arkasol.Checked:= True;
        if sayi1 = 2 then
            begin
            onde.Checked:= False;
            onsag.Checked:= False;
            onsol.Checked:= False;
            arkada.Checked:= False;
            arkasag.Checked:= False;
            arkasol.Checked:= False;
            end;

        edit1.Text:= edit1.Text + IntToStr(sayi1)+' ';
        alinanveri:= alinanveri + sayi[1];
        sayi:= copy(sayi,2,length(sayi));
        end;
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Ana menüden "Durdur" menüsüne basıldığında çağrılan program;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.DurdurButtonClick(Sender: TObject);
begin
    Listbox.Enabled:= True;
    MSComm1.Output:= chr(0);
    ListBox.ItemIndex:= -1;
    Timer1.Enabled:= False;
```

```
    CalisButton.Enabled:= True;
    Calistir.Enabled:= True;
    DurdurButton.Enabled:= False;
    Durdur.Enabled:= False;
    btnDelete.Enabled:= True;
    Listbox.MultiSelect:= True;
    GezTabGorevEkle.Enabled:= True;
    KolGorevEkle.Enabled:= True;
    RobModel.Enabled:= True;
    Habhiz.Enabled:= True;
    StatusBar1.Panels[0].Text:= '';
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Program menüsünden Doğrudan Çalışma Modunun seçimi;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.DogrudanClick(Sender: TObject);
begin
    SetCheck(Sender);
    YeniClick(Sender);
    SurekliMod.Checked:= False;
    Kayittan.Checked:= False;
    NormalMod.Checked:= False;
    Initiate(Sender);
    Form1.Caption:= 'ROBOT PROGRAMI';
    TabGorEkPanel.Visible:= False;
    KolGorEkPanel.Visible:= False;
    GorListPanel.Visible:= False;
    YarLabPanel.Visible:= False;
    AciLabPanel.Visible:= False;
    SureLabPanel.Visible:= False;
    SurePanel.Visible:= False;
    Yaz8.Visible:= False;
    AciPanel.Visible:= False;
    YarPanel.Visible:= False;
    KonumPanel.Caption:= '';
    StatusBar1.Panels[1].Text:= 'Robot doğrudan çalışma modunda...';
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Doğrudan Çalışma Modunda robota dönme hareketinin yaptırılması;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.SagsolClick(Sender: TObject);
begin
    TrackBar10Change(Sender);
end;
```

```
procedure TForm1.GeriileriClick(Sender: TObject);
begin
   TrackBar10Change(Sender);
end;

procedure TForm1.TrackBar10Change(Sender: TObject);
var
hiz, sagsolyon, ilerigeriyon, yon, ayar, r: integer;
a, b: word;
begin
   r:= 45;
   if Dogrudan.Checked = True then
      begin
      if DumRobot.Checked = True then
         begin
         if (Sagsol.ItemIndex = -1) and (Trackbar10.Position <> 0) then
            begin
            Trackbar10.Position:= 0;
            ShowMessage('Sağ mı, sol mu?');
            end;
         if (Geriileri.ItemIndex = -1) and (Trackbar10.Position <> 0) then
            begin
            Trackbar10.Position:= 0;
            ShowMessage('İleri mi, Geri mi?');
            end
         else if Sagsol.ItemIndex = 0 then
            sagsolyon:= dumsagadon
         else if Sagsol.ItemIndex = 1 then
            sagsolyon:= dumsoladon;
         if Geriileri.ItemIndex = 0 then
            ilerigeriyon:= ileridon
         else if Geriileri.ItemIndex = 1 then
            ilerigeriyon:= geridon;
         if TrackBar10.Position = 0 then
            begin
            MSComm1.Output:= chr(0);
            StatusBar1.Panels[0].Text:= 'Robot duruyor...';
            end
         else
            begin
            hiz:= TrackBar10.Position;
            tekeracisi:= ArcTan(26/r)*(180/pi);
            servoacisi:= sagsolyon*3* round(tekeracisi);
            donkomutu:= round((750+(90 - servoacisi)*(1625/180))/4)*64;
            a:= (donkomutu) and (255);
            b:= (donkomutu) and (65280);
            a:= (a) shr (2);
            donkomutu:= a+b;
            ayar:= ilerigeriyon + (hiz*16);
            MSComm1.Output:= chr(ayar);
            MSComm1.Output:= chr(hi(donkomutu));
```

```
            MSComm1.Output:= chr(lo(donkomutu));
            StatusBar1.Panels[0].Text:= 'Robot dönüyor...';
            end;
         end

      else if DifRobot.Checked = True then
         begin
         if (sagsol.ItemIndex = -1) and (Trackbar10.Position <> 0) then
            begin
            Trackbar10.Position:= 0;
            ShowMessage('Sağ mı, sol mu?');
            end
         else if Sagsol.ItemIndex = 0 then
            yon:= difsagadon
         else if Sagsol.ItemIndex = 1 then
            yon:= difsoladon;
         if TrackBar10.Position = 0 then
            begin
            MSComm1.Output:= chr(0);
            StatusBar1.Panels[0].Text:= 'Robot duruyor...';
            end
         else
            begin
            hiz:= TrackBar10.Position;
            ayar:= yon + (hiz*16);
            MSComm1.Output:= chr(ayar);
            StatusBar1.Panels[0].Text:= 'Robot dönüyor...';
            end;
         end;
      end;
end;


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Doğrudan Çalışma Modunda robota ileri-geri gitme hareketinin yaptırılması;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.ilerigeriClick(Sender: TObject);
begin
   TrackBar8Change(Sender);
end;

procedure TForm1.TrackBar8Change(Sender: TObject);
var
hiz, yon, ayar, ilerigit, gerigit: integer;
begin
   if Dogrudan.Checked = True then
      begin
      if DumRobot.Checked = True then
         begin
         ilerigit:= dumilerigit;
         gerigit:= dumgerigit;
```

```
        end
      else if DifRobot.Checked = True then
        begin
        ilerigit:= difilerigit;
        gerigit:= difgerigit;
        end;
      if (ilerigeri.ItemIndex = -1) and (Trackbar8.Position <> 0) then
        begin
        Trackbar8.Position:= 0;
        ShowMessage('İleri mi, geri mi?');
        end
      else if ilerigeri.ItemIndex = 0 then
        yon:= ilerigit
      else if ilerigeri.ItemIndex = 1 then
        yon:= gerigit;
      if TrackBar8.Position = 0 then
        begin
        MSComm1.Output:= chr(0);
        StatusBar1.Panels[0].Text:= 'Robot duruyor...';
        end
      else
        begin
        hiz:= TrackBar8.Position;
        ayar:= yon + (hiz*16);
        MSComm1.Output:= chr(ayar);
        StatusBar1.Panels[0].Text:= 'Robot ilerliyor...';
        end;
      end;
end;
```

* *

Robor kolun her ekleminin yanındaki işaret kutularının işaretlenmesi durumunda eklemlerin açı giriş yapılabilir duruma getirilmesi;

* *

```
procedure TForm1.Check (Sender: TObject);
begin
   if Check1.Checked = False then
      begin
      Eklem1Panel.Enabled := False;
      Yaz1.Text := '0';
      TrackBar1.Position := 0;
      end
   else
      Eklem1Panel.Enabled := True;
   Yaz1.Text := IntToStr(TrackBar1.Position);

   if Check2.Checked = False then
      begin
      Eklem2Panel.Enabled := False;
      Yaz2.Text := '0';
```

```
      TrackBar2.Position:= 0;
      end
   else
      Eklem2Panel.Enabled := True;
   Yaz2.Text := IntToStr(TrackBar2.Position);

   if Check3.Checked = False then
      begin
      Eklem3Panel.Enabled := False;
      Yaz3.Text := '0';
      TrackBar3.Position:= 0;
      end
   else
      Eklem3Panel.Enabled := True;
   Yaz3.Text := IntToStr(TrackBar3.Position);

   if Check4.Checked = False then
      begin
      Eklem4Panel.Enabled := False;
      Yaz4.Text := '0';
      TrackBar4.Position:= 0;
      end
   else
      Eklem4Panel.Enabled := True;
   Yaz4.Text := IntToStr(TrackBar4.Position);

   if Check5.Checked = False then
      begin
      Eklem5Panel.Enabled := False;
      Yaz5.Text := '0';
      TrackBar5.Position:= 0;
      end
   else
      Eklem5Panel.Enabled := True;
   Yaz5.Text := IntToStr(TrackBar5.Position);

   if Check6.Checked = False then
      begin
      Eklem6Panel.Enabled := False;
      Yaz6.Text := '0';
      TrackBar6.Position:= 0;
      end
   else
      Eklem6Panel.Enabled := True;
   Yaz6.Text := IntToStr(TrackBar6.Position);
end;
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Robor kolun eklemlerine ait metin kutularına, harf girişi, mininmum ve maksimum değerler dışında veri girişi yapılmasını engelleyen program. Bu program altı eklem için tekrarlanmıştır.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.Yaz1Change(Sender: TObject);
var
gir, veri, code:integer;
s:string;
begin
   MinAciDegeri:= StrToInt(Form3.AciMin.Text);
   MaxAciDegeri:= StrToInt(Form3.AciMax.Text);
   s:=Yaz1.Text;
   Val(s, gir, code);
   if code <> 0 then
   begin
   System.Delete(s,code,1);
   Yaz1.Text:= s;
   end
   else if gir > MaxAciDegeri then
   begin
   Yaz1.Text:= '90';
   Val(Yaz1.Text, gir, code);
   end
   else if gir < MinAciDegeri then
   begin
   Yaz1.Text:= '-90';
   Val(Yaz1.Text, gir, code);
   end;
   Trackbar1.Position:= gir;
   end;
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Robor kolun görevinin görev listesine eklenmesi;

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
procedure TForm1.KolGorevEkleClick(Sender: TObject);
var koldizi:string;
begin
   Listboxflag:= True;
   koldizi:= '';

   if Check1.Checked = True then
      koldizi:= koldizi+'M1='+IntToStr(Trackbar1.Position)+'°'+' '
   else koldizi:= koldizi+'';

   if Check2.Checked = True then
      koldizi:= koldizi+'M2='+IntToStr(Trackbar2.Position)+'°'+' '
   else koldizi:= koldizi+'';
```

```
    if Check3.Checked = True then
       koldizi:= koldizi+'M3='+IntToStr(Trackbar3.Position)+'°'+' '
    else koldizi:= koldizi+'';

    if Check4.Checked = True then
       koldizi:= koldizi+'M4='+IntToStr(Trackbar4.Position)+'°'+' '
    else koldizi:= koldizi+'';

    if Check5.Checked = True then
       koldizi:= koldizi+'M5='+IntToStr(Trackbar5.Position)+'°'+' '
    else koldizi:= koldizi+'';

    if Check6.Checked = True then
       koldizi:= koldizi+'M6='+IntToStr(Trackbar6.Position)+'°'+' '
    else koldizi:= koldizi+'';

    if koldizi <> '' then
       ListBox.Items.Add(koldizi);
    if ListBox.Items.Text <> '' then
       CalisButton.Enabled:= True;
end;
```

* *
Görev sil düğmesine basıldığında, görev listesindeki seçili satır veya satırları silen program;
* *

```
procedure TForm1.btnDeleteClick(Sender: TObject);
var a:integer; s:string; sakla:Boolean;
begin
    sakla:= False;
    if Listbox.SelCount <> 0 then
       for a:= (Listbox.Items.Count-1) downto 0 do
          if Listbox.Selected[a] then
             begin
             s:=Listbox.Items.Strings[a];
             if (strpos(pchar(s),'begin')) <> nil then sakla:= True;
             else if (strpos(pchar(s),'if')) or (strpos(pchar(s),'for')) or
             (strpos(pchar(s),'else')) <> nil then
                begin
                if sakla = False then
                   ShowMessage('Döngü komutları begin ile birlikte silinmelidir!')
                else
                   begin
                   ListBox.Items.Delete(a);
                   ListBox.Items.Delete(a);
                   Listboxflag:= True;
                   end;
                end
             else
                begin
                Listboxflag:= True;
```

```
                ListBox.Items.Delete(a);
                end;
            end;
    if Listbox.Items.Text = '' then
        CalisButton.Enabled:= False;
    DefocusControl(Listbox,True);
end;
```

* *

Programın araç çubuğu üzerindeki "Yeni", "Aç", "Kaydet", "Yazdır" "Çalıştır" ve "Durdur" düğmelerine basıldığında çağrılan programlar;

* *

```
procedure TForm1.YeniButtonClick(Sender: TObject);
begin
    YeniClick(Sender);
end;

procedure TForm1.AcButtonClick(Sender: TObject);
begin
    AcClick(Sender);
end;
procedure TForm1.KaydetButtonClick(Sender: TObject);
begin
    KaydetClick(Sender);
end;

procedure TForm1.YazdirButtonClick(Sender: TObject);
begin
    YazdirClick(Sender);
end;

procedure TForm1.CalistirClick(Sender: TObject);
begin
    CalisButtonClick(Sender);
end;

procedure TForm1.DurdurClick(Sender: TObject);
begin
    DurdurButtonClick(Sender);
end;
```

* *

Yeni görev listesi açan program ("Yeni" menüsü tarafından çağrılır.);

* *

```
procedure TForm1.YeniClick(Sender: TObject);
var sor:integer;
begin
    if Listboxflag = True then
```

```
      begin
      if Open <> '' then
         begin
            case Application.MessageBox(PChar(Open +'e son değişiklikler
            kaydedilsin mi?'),'Uyarı',mb_YesNoCancel+mb_IconQuestion) of
            IDYes:   begin
                     ListBox.Items.SaveToFile(Open);
                     Initiate(Sender);
                     Open := '';
                     end;
            IDNo:    begin
                     Initiate(Sender);
                     Open := '';
                     end;
            end;
         end
      else
         begin
            case Application.MessageBox(PChar('Değişiklikler kaydedilsin mi?')
            ,'Uyarı',mb_YesNo+mb_IconQuestion) of
            IDYes:   begin
                     FarkliAdlaKaydetClick(Sender);
                     Initiate(Sender);
                     end;
            IDNo:    Initiate(Sender);
            end;
         end;
      end
   else Initiate(Sender);
end;
```

* *

Daha önce kaydedilmiş bir görev listesini açan program ("Aç" menüsü tarafından çağrılır.);

* *

```
procedure TForm1.AcClick(Sender: TObject);
begin
   if Listboxflag = True then
      begin
         case Application.MessageBox(PChar('Değişiklikler kaydedilsin mi?')
         ,'Uyarı',mb_YesNo+mb_IconQuestion) of
         {IDCancel:}
         IDYes:   begin
                  if Open = '' then
                     begin
                     FarkliAdlaKaydetClick(Sender);
                     Acma(Sender);
                     end
                  else ListBox.Items.SaveToFile(Open);
                  Acma(Sender);
                  end;
```

```
        IDNo:   begin
                Acma(Sender);
                Listboxflag:= False;
                end;
        end;
      end
   else Acma(Sender);
end;
```

* *

Diferansiyel sürüşlü robot için ".dir", dümen mekanimalı robot için ".dur" dosyalarını açan program;

* *

```
procedure TForm1.Acma(Sender: TObject);
begin
   if DumRobot.Checked = True then
      if OpenDialogDum.Execute then
         begin
         Open:= OpenDialogDum.Filename;
         ListBox.Items.LoadFromFile(Open);
         Form1.Caption:= 'ROBOT PROGRAMI - '+Open;
         CalisButton.Enabled:= True;
         Calistir.Enabled:= True;
         Listbox.ItemIndex:= -1;
         end

      else if DifRobot.Checked = True then
         if OpenDialogDif.Execute then
            begin
            Open:= OpenDialogDif.Filename;
            ListBox.Items.LoadFromFile(Open);
            Form1.Caption:= 'ROBOT PROGRAMI - '+Open;
            CalisButton.Enabled:= True;
            Calistir.Enabled:= True;
            Listbox.ItemIndex:= -1;
            end;
end;
```

* *

Görev listesini, diferansiyel sürüşlü ve dümen mekanizmalı robotlar için ayrı ayrı kaydeden program ("Kaydet" ve "Farklı Adla Kaydet" menüleri tarafından çağrılır.);

* *

```
procedure TForm1.KaydetClick(Sender: TObject);
begin
   if Save = '' then
      FarkliAdlaKaydetClick(Sender)
   else
      begin
```

```
      ListBox.Items.SavetoFile(Save);
      Listboxflag:= False;
      end;
end;

procedure TForm1.FarkliAdlaKaydetClick(Sender: TObject);
begin
   if DumRobot.Checked = True then
      if SaveDialogDum.Execute then
         begin
         Save:= SaveDialogDum.Filename;
         ListBox.Items.SavetoFile(Save);
         Form1.Caption:= 'ROBOT PROGRAMI - '+Save;
         Listboxflag:= False;
         end
   else if DifRobot.Checked = True then
      if SaveDialogDif.Execute then
         begin
         Save:= SaveDialogDif.Filename;
         ListBox.Items.SavetoFile(Save);
         Form1.Caption:= 'ROBOT PROGRAMI - '+Save;
         Listboxflag:= False;
         end;
end;
```

* *

Programdan çıkmak için yazılmıştır. İlk prosedür, "Çıkış" menüsü tarafından çağrılır. İkinci prosedür, çıkarken kaydetme sorgusunu yapar.

* *

```
procedure TForm1.cikisClick(Sender: TObject);
begin
   Close;
end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
   if Listboxflag = True then
      begin
         case Application.MessageBox(PChar('Değişiklikler kaydedilsin mi?')
         ,'Uyarı',mb_YesNoCancel+mb_IconQuestion) of
         IDCancel: CanClose:= False;
         IDYes:  begin
                 requesttoclose:= True;
                 if Open = '' then
                    begin
                    FarkliAdlaKaydetClick(Sender);
                    CanClose:= False;
                    end
                 else ListBox.Items.SaveToFile(Open);
                 end;
```

```
        end;
      end;
end;
```

* *

Görev listesini yazıcıya gönderen program;

* *

```
procedure TForm1.YazdirClick(Sender: TObject);
begin
   PrintDialog.Execute;
   RichEdit.Lines:= ListBox.Items;
   RichEdit.Print('Robot Görev Listesi');
end;
```

* *

Program menüsü üzerine "seçildi" işareti koyan program;

* *

```
procedure TForm1.SetCheck(Sender: TObject);
var Item: TMenuItem;
begin
   Item:= Sender as TMenuItem;
   Item.Checked:= True;
end;
```

ÖZGEÇMİŞ

Doğum tarihi	19.03.1975	
Doğum yeri	İstanbul	
Lise	1986-1993	Hüseyin Avni Sözen Anadolu Lisesi
Lisans	1993-1997	Yıldız Teknik Üniversitesi, Elektrik-Elektronik Fak., Elektronik ve Haberleşme Mühendisliği Bölümü
Yüksek Lisans	1997-2000	Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme Müh. Anabilim Dalı, Elektronik Programı
Çalıştığı kurum		
	1998-	YTÜ, Elektrik Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü, Devreler ve Sistemler Anabilim Dalı, Araştırma Görevlisi