1. **PROJE KAPSAMINDA YAPILAN**
   1. Benzetim (simülatör) geliştirildi.

Burada amaç mevcut olan simülatörlerin araştırılması ve öncelikle çalışma prensiplerinin öğrenilmesidir. Daha sonra bu simülatörlerin üzerine ne konulabileceğinin araştırılıp uygulanmasıdır.

* 1. Geliştirilen benzetim araba benzeri robota uygulandı.

İstenilen bu benzetim algoritmaları günlük hayatta karşılaştığımız sorunları sembolize etmesi amaçlı araba benzeri robota uygulanacaktır. Günümüzdeki araçların hareket kabiliyetlerine sahip olacaktır. Bu hareketler belli kısıtlamalar içerisinde gerçekleştirilecektir. Bu hareket kısıtlamaları belli kinematik denklemlere göre gerçekleştirilecektir.

* 1. Benzetim ortamında araba benzeri robot, duvar izleme, köşe dönme, paralel park etme gibi görevleri yerine getirebildi.

Simülasyon ortamında araba benzeri robot, belli görevlere sahip olacaktır. Bu görevler önceden belirlenecekdir. Görevler bir arabanın hareket kabiliyetine göre otomatik bir şekilde simülatör başlatılmadan kullanıcı tarafından seçilerek gerçekleştirilecektir. Bu görevler duvar takibi, algılanan 90 derecelik köşeleri dönme, paralel park etme gibi hareketlerdir.

* 1. Robotlarda kullanılacak olan algılayıcılar araştırıldı.

Bu proje ne kadar yazılım uygulaması olsada, bu yazılım günlük hayatta kullanılan cihazların algılama kısıtlamalarıyla eş değerde olmalıdır. Çünkü simülatörün gerçekçiliği buna bağlıdır.

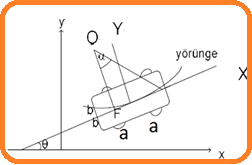
Bu projede kullanılıcak algılayıcılar araba benzeri robotun dönen tekerleklerinin hangi açı ile tutulacağı, cisim ile araba arasında ne kadar mesafe olduğunu anlayan algılayıcılar ve köşe dönme uygulamasında ki gibi köşeyi algılayan algılayıcılar baz alınarak gerçekleştirilecektir.

* 1. Benzetim ortamı oluşturmak için C ,C++ ,C# ,java gibi yazılım dilleri kullanıldı.

Yapılacak olan bu benzetimler yazılımsal olacağı için nesne yönelimli yazılım dilleri kullanılacaktır. Kullanılan bu diller kullanıcı açısından değişikliklere esneklik gösterebileceği için bu diller seçilmiştir. Bu dilleri alt yapısında barındıran simülasyon programları kullanılacaktır.

1. **ARABA BENZERİ GEZGİN ROBOTLARIN KİNEMATİK DENKLEMLERİ**

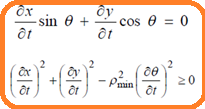
Araç boyutları 2a ve 2b olan dikdörtgen bir nesne olarak kabul edilmiştir. Robotun ön tekerlerinin yönlenmeyi sağlayacak şekilde sınırlı olarak döndüğü ve sürüşün arka tekerler aracılığıyla sağlandığı varsayılmıştır.



Şekil 2.1 (araba benzeri gezgin robot modeli)

Şekil 1’ de robotun modellenmesinde arka tekerleri birleştiren aksın orta noktası (F) referans noktası olarak alınmıştır. Bu referans noktası bazı uygulamalarda arabanın sağ arka köşesi alınacaktır.

Araba benzeri robot iki kısıtlama ile karşı karşıyadır. Bunlarda ilki hızın 0 radyal bileşeni iken ikincisi dönme yarı çapı içeren bileşenidir.

[[1]](#footnote-1)

Yukarıda ki bu denklemler koordinat sisteminde sol alt köşeyi referans alarak hesaplanacak ve açısı bu referans noktasına göre ölçülecektir. Kullanılacak olan bu denklemler aracın açısal hareket gerektiren bölümlerinde kullanılacaktır. Bu açısal hareketleri öncesi belli parametrelere göre bu denklemleri gerçekleştireceği için önceliklebuparametreler algılayıcılar tarafından okunur ve belli değişkenlere atanır. Örnek olarak bu değişkenler yapılan Visual Studio programında C# dili kullanılarak Windows Forms Application ‘da duvar takip etme ve köşe dönme özellikleri simüle edilmiştir.

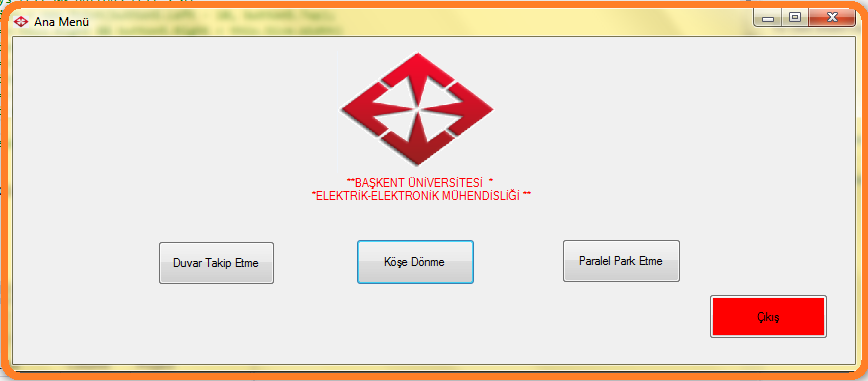
Duvar takip etme ve köşe dönme simülasyonları Visual Studio ‘da kısmen gerçekleştirilmiştir. Gerçekleştirilen bu simülasyonların matematiksel algoritmaları yazılıma uyarlanmıştır.

Uyarlama kısmı gerçek hayata benzetmek amaçlı olarak duvar ve araç benzeri görseller içermektedir.

Projenin ilerleyen kısımlarında bu hareketler geliştirilerek bir aracın paralel park etmesi simülasyonu gerçekleştirilecektir.

1. **KİNEMATİK DENKLEMLERİN YAZILIM ORTAMINDA GERÇEKLEŞTİRİLMESİ VE GÖRSELLEŞTİRİLMESİ**
2. **İLK DÖNEM YAPILANLAR**

İlk dönem gerçekleştirilen 2 boyutta proje belli iç içe windows çerçevelerinden oluşmaktadır. Program ilk açıldığında karşımıza duvar takip etme, köşe dönme ve paralel park etme butonları çıkacaktır.

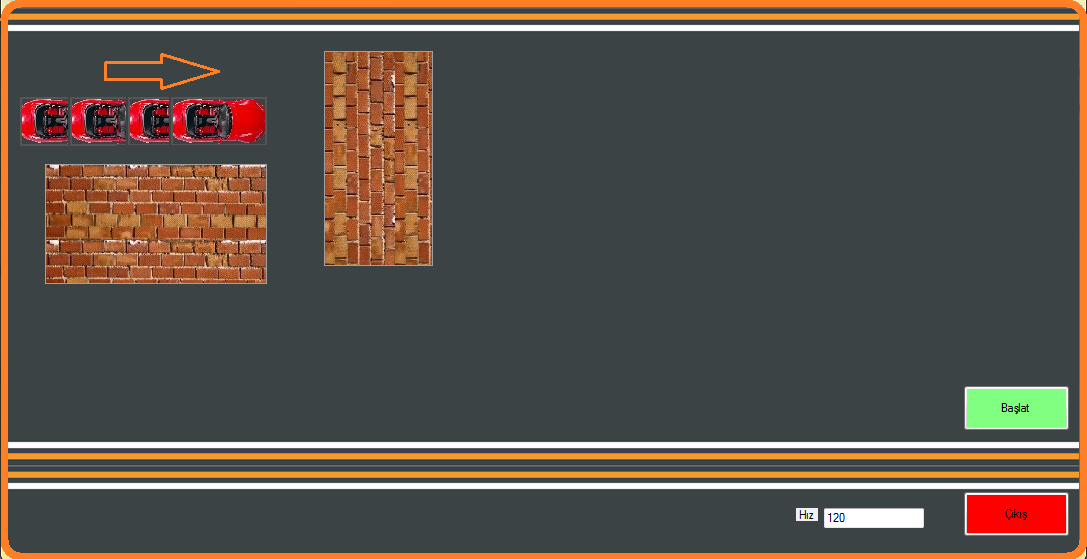
[[2]](#footnote-2)

Şekil 3.1 (Program Ana Menüsü)

Şekil 3.1 ‘den de görüldüğü gibi çıkan bu butonlar ile alt pencerelere ulaşılacaktır.

* 1. **Duvar Takip Etme**

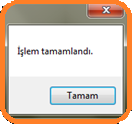
Duvar takip etme butonuna basıldığı zaman çıkıcak olan ekran Şekil 3.2 ‘de gösterilmiştir.



Şekil 3.1.1[[3]](#footnote-3) (Duvar Takip Etme)

Duvar takip etme görevinde araba benzeri robot algılayıcıları tarafından yakın bulunduğu duvarı takip edicek ve önüne başka bir duvar veya cisim çıkma durumunda hareketini sonlandıracaktır.

Başlat butonuna basıldığı zaman araç ok yönünde hareket edecektir. Takip ettiği duvar bitiminde ,boşluk algılanılacak ve aracın hareketi sonlanacaktır.



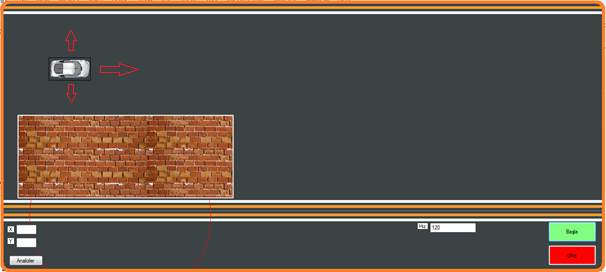
Şekil 3.1.2 (Hareket tamamlandığı zaman kullanıcıya bildirimi)

* 1. **Köşe Dönme**

Köşe dönme butonuna basılğı zaman, araç benzeri robotumuz bu aşamada hareket yönünde ilerleyip köşe algılandığı zaman duvar ile arasındaki mesafeye göre köşe ile arka tekeri arasında ki mesafeyi ayarlıcak ve dönme yarıçapına göre hareket rotası çizilecektir.

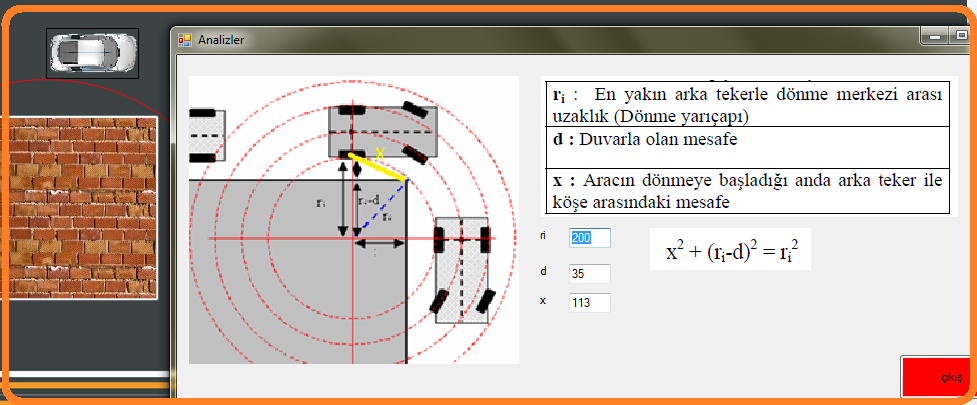
Dönüş eylemi sırasında gerçekleştirilen işlemler aşağıdaki gibi sıralanabilir.

* Robot sağ duvarı paralel olarak izler.
* Sağ ön sensör tarafından köşe algılanır.
* Duvara olan uzaklığa göre dönme noktası ve alınacak yol hesaplanır.
* Dönme noktasında, maksimum direksiyon açısıyla dönüş yapılır.
* Ön ve arka algılayıcı tarafından ölçülen
* Mesafe eşitlenince dönme işlemi tamamlanır.



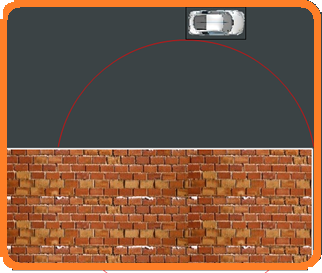
Şekil 3.2.1[[4]](#footnote-4) (Köşe dönme)

Şekil 3.2.1 ‘de gösterilen ok yönleri aracın hareket yönleridir. Bilgisayar faresi yardımıyla aracın konumunu belirlenebilir ve her durum için aracın hareket yörüngesi çizdirilebilir. Değişkenlerin değerlerini görmek isteyen kullanıcı açılan ekranın sol alt kısmında yer alan “Analizler” butonuna basarak görebilir.

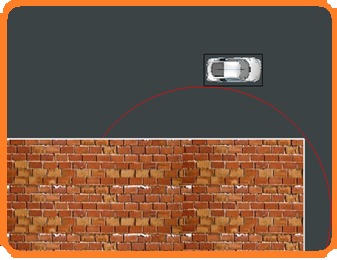
[[5]](#footnote-5)

Şekil 3.2.2[[6]](#footnote-6) (Analizler)

Analiz bölümünde olduğu gibi araç duvar ile arasında ki uzaklığa göre belli kinematik denklemler de değişkenleri yerine koyarak hesaplama yapar ve ona göre hareket edecektir.



Şekil 3.2.3 (Köşe dönme - örnek1)



Şekil 3.2.4 (Köşe dönme - örnek2)



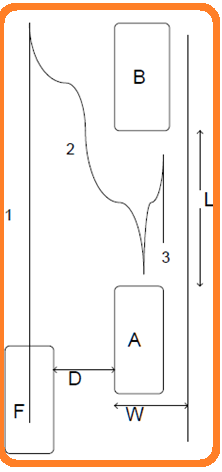
Şekil 3.2.5 (Köşe dönme - örnek3)

* 1. **Paralel Park Etme**

Paralel park etme hareketinin temelleri duvar takip etme ve köşe dönme hareketlerinden gelir. Davar takip etme hareketinden iki cisim veya araç arasındaki boşluğun algılanması ve köşe dönme hareketinden ise aracın çembersel hareket kabiliyeti kazanılmıştır. Temel olarak paralel park etme süreci üçe bölünmüştür.

* Tarama
* Posizyon Alma
* Park Etme

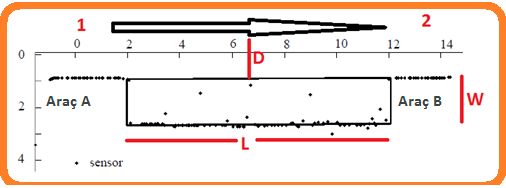
Öncelike temel olarak belli değişkenlerimiz vardır. Bunlar algılayıcılar tarafından ölçülür ve sistem bu ölçümler sonucu hareketlerini kısıtlar.



Şekil 3.3.1 (Sağ taraf paralel park etme durumu)

* + 1. Tarama Fazı

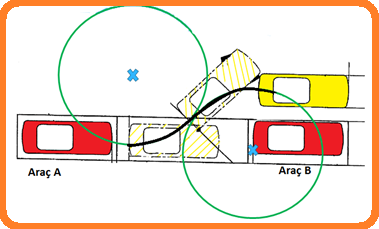
Park alanı genelde iki park etmiş araç arasında kalmış olan dikdörtgen bir alan olarak varsayılır. Araç, 1 ve 2 pozisyonu arasında düz bir şekilde giderken algılayıcılar ortam hakkında bilgi toplar ve bu fazın sonunda L, W ve D hesaplanmış olur.

[[7]](#footnote-7)

Şekil 3.3.1.1 (Tarama fazı)

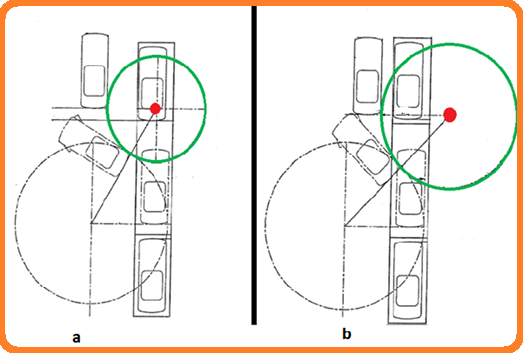
* + 1. Pozisyon Alma Fazı

Pozisyon alma fazı paralel park etmek için oldukça önemlidir ve tarama ve park etme fazı arasında bir geçiş görevi görür. Pozisyon alma fazında sadece bir tane hareket vardır. Aracın F noktası (Şekil 3.3.1) birbiriyle kesişen iki benzer çemberi takip eder . Çemberlerin üzerinde eşit uzunlukta yaylar takip edilecektir.



Şekil 3.3.2.1 (Paralel park etme)

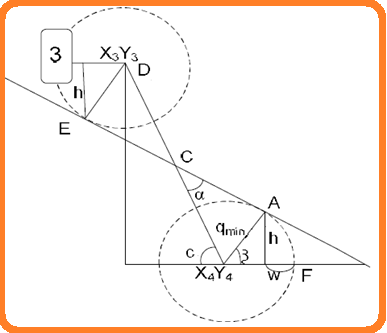
Algılanan değişken değerlerine göre farklı paralel park etme teknikleri kullanılabilir. Aracın dönme yarıçapına göre örnek alınıcak çemberlerin merkezle koordinatları değişebilir.



Şekil 3.3.2.2 [[8]](#footnote-8) (a ve b oluşturulacak farklı merkezli çemberler)

* + 1. Park Etme Fazı

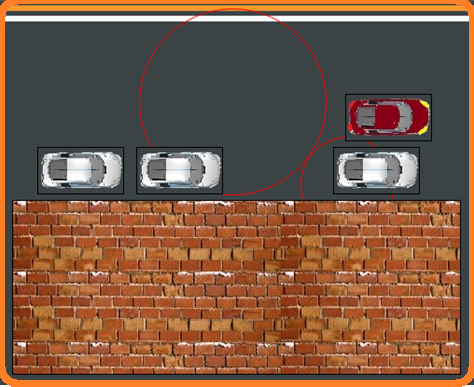
Park etme fazında iki türlü hareket kullanılmıştır. Bunlardan bir tanesi park alanının robottan çok daha büyük olduğu durumlarda tek hareketle istenilen yere park edilmesidir . Bu hareket 3[[9]](#footnote-9) numaralı pozisyonda başlar ve F[[10]](#footnote-10) noktası alttaki aracın orta noktasıyla aynı dikey hizaya geldiğinde hareket biter.

[[11]](#footnote-11)

Şekil 3.3.3.1[[12]](#footnote-12) (Park alanının büyük olduğu durumda kullanılan hareket)

Park etme fazında kullandığımız diğer hareket ise park alanının robota göre çok büyük olmadığı yani tek hareketle park edemediğimiz durumlarda kullanılacak harekettir. Bu hareket pozisyon alma fazındaki hareketle hemen hemen aynıdır.

Yapılan programın paralel park bölümünden ekran görüntüleri :

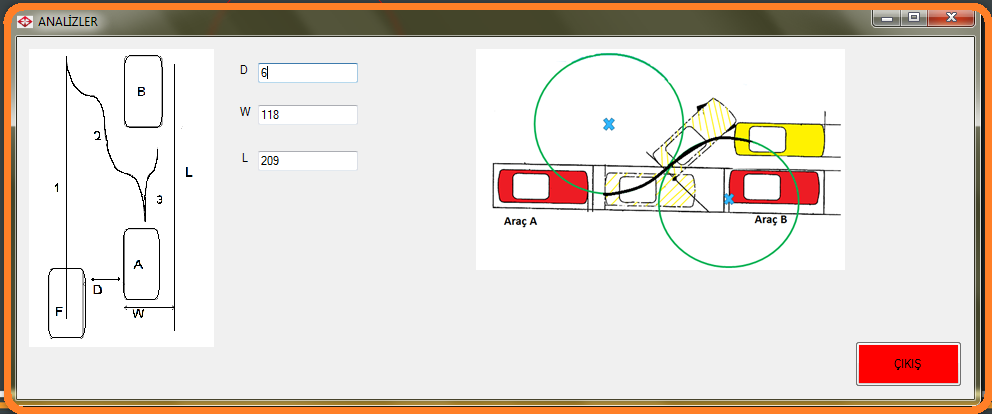


Şekil 3.3.3.2[[13]](#footnote-13) (Paralel park etme)



Şekil 3.3.3.3 (Tuş kısmı)

Paralel park etme için programını çaşıltıran kullanıcı Başlat tuşuna bastıktan sonra Analizler tuşu ile değişkenleri görülebilircektir.

[[14]](#footnote-14)

Şekil 3.3.3.4 (Analizler)

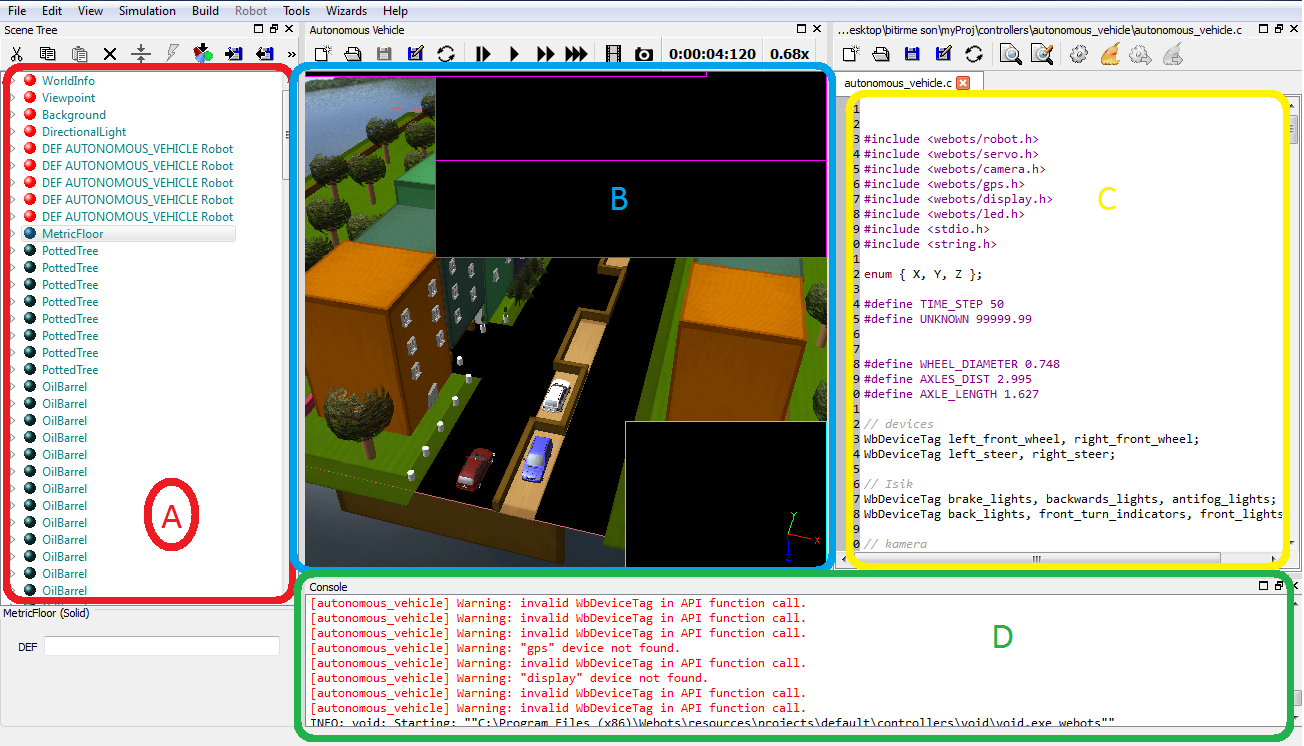
1. **İKİNCİ DÖNEM YAPILANLAR**

İlk dönem yapılan 2 boyuttaki çalışma 3 boyuta taşınmıştır. 3 boyutta 3 düzlem yer almıştır. Aynı kinematik denklemler 3 boyutta ki araba benzeri gezgin robota uygulanmıştır. 3 boyutta uygulama “Webots” adın da bir program yardımıyla olmuştur.



Şekil 3.4 (Webots programı sembolü)

1. **Webots programının tanıtımı**



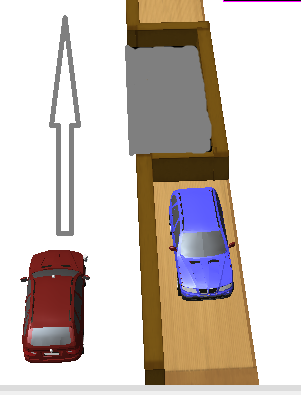
Şekil 4.1 (Webots programı kullanıcı arayüzü)

1. Arayüz obje ekleme
2. Tasarım ve robot kontrol gösterimi
3. Yapılan tasarımda hareket ettirilen robotun kontrolü için yazılan kod kısmı. Bu kod kısmı C, C++, JAVA gibi yazılım dilleri ile yazılmaktadır.
4. Console bölümü

Projenin ikinci kısmı tamamen 3 boyutta gerçekleştiği için Webots programı kullanılmıştır. Programda robot araba benzeri robot olmuştur.

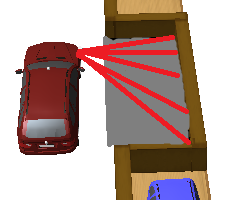


Şekil 4.2 (araba benzeri gezgin robot)



Şekil 4.3 (Robotu park etme sisteminin çalıştırma anı)

Kırmızı araç ok yönünde ilerliyecek ve sağında bulunan kendisine paralel olan gri renkli bölgeye park edicektir. Bu işlemler 2 boyutta olduğu gibi üç aşamada gerçekleşecektir. Tarama fazı ilk olarak gerçekleşen fazdır.



Şekil 4.4 (Tarama fazının gerçekleşmesi)

* Kısmen gerçekleştirilen benzetim, tamamlanacak.
* 2 boyutta yapılan benzetim 3 boyut görselleştirmesi yapılacak.
* Araç benzeri robot istenilen hareketleri gerçektirebileceği ortamlar oluşturulacak ve geliştirilecek.
* Yazılım ortamında robot farklı görevleri yerine getirebilecektir.

**KAYNAKLAR**

**[1]** PID CONTROLLER FOR AN AUTONOMOUS PARALLEL SELF PARKİNG SYSTEM - TEOH WEN XIN

**[2]** UNİTED STATES PATENT APPLİCATION PUBLİCATİON – JUI HUNG WU

**[3]** UNISINOS IEEE COMPUTER SOCİETY

**[4]** AUTOMATIC VEHICLE PARKING USING AN EVOLUTION-OBTAINED NEURAL CONTROLLER - Franco Ronchetti, Laura Lanzarini

**[5]**<http://www.intelligence.tuc.gr/~>[robots/ARCHIVE/2011w/projects/Theodoropoulos/Presentation\_final.pdf](http://www.intelligence.tuc.gr/~robots/ARCHIVE/2011w/projects/Theodoropoulos/Presentation_final.pdf)

**[6]**<http://www.idsc.ethz.ch/Research_Guzzella/Automotive_Applications/Hybrid_Powertrains/archives/Semi-Autonomous_Parking>

1. min kullanılıcak olan arac benzeri robotun minimum dönme yarı çapı [↑](#footnote-ref-1)
2. Kod kısmı için bakınız EK-1 [↑](#footnote-ref-2)
3. Kod kısmı için bakınız EK-2 [↑](#footnote-ref-3)
4. Kod kısmı için bakınız EK-3 [↑](#footnote-ref-4)
5. ri Dönme yarıçapı

   d Duvar ile araç arası mesafe

   x Araç dönmeye başladığı anda arka teker ile köşe arası mesafe [↑](#footnote-ref-5)
6. Kod kısmı için bakınız EK-4 [↑](#footnote-ref-6)
7. D = park edilecek araç ile araç A arası mesafe

   W = araç A ve araç B ‘nin eni veya park edilecek araç ile kaldırım arası mesafe

   L = park edilecek boşluk uzunluğu [↑](#footnote-ref-7)
8. United State Patent Application Publication (US 2009/0085771 A1) [↑](#footnote-ref-8)
9. Bakınız Şekil 3.3.3.1 [↑](#footnote-ref-9)
10. Bakınız Şekil 3.3.3.1 [↑](#footnote-ref-10)
11. min Aracın dönme açısı [↑](#footnote-ref-11)
12. ARABA BENZERİ GEZGİN ROBOTUN OTOMATİK PARK ETMESİ İÇİN BİR YÖNTEM (Eskişehir Osmangazi Üniversitesi) [↑](#footnote-ref-12)
13. Kod kısmı için bakınız EK-5 [↑](#footnote-ref-13)
14. Kod kısmı için bakınız EK-6 [↑](#footnote-ref-14)