Araçlarda Kendi Kendine Yol Bulma Sistemi

Selim Göksu [1] Nihan Kahraman [2] Tülay Yıldırım[3]

[1,2,3] Elektronik ve Haberleşme Mühendisliği Bölümü, Yıldız Teknik Üniversitesi, İstanbul
[1]e-posta: slmgks@gmail.com [2,3] e-posta: {nicoskun, tulay}@yildiz.edu.tr

Özetçe

Bu çalışmada boyutları bilinen bir harita üzerinde noktalar arası kendi kendine gezinebilen robot araç tasarlanmıştır. Harita, üzerinde gezinen aracın rahatça hareket edebildiği boş alanlara ve aracın çarpmaması gereken engellere sahiptir. Amaç, daha önceden belirlenen harita üzerinde aracın belirlenen bir noktadan istenen herhangi bir noktaya engellere çarpmadan gidebilmesini sağlamaktır. Aracın istenen konuma gideceği güzergahlar arasında en kısa yolu bulabilmesi önemli bir özelliktir. Bir diğer özellik ise aracın güzergah üzerinde hareketi sırasında haritada belirlenmeyen bir engelin sonradan önüne çıkarılması veya engelin sonradan kaldırılabilmesi durumunda araç kendi kendine yeni güzergahını tekrar belirleyebilmektedir. RF iletişim çift yönlü yapılarak aracın her adımda çevresindeki engel bilgisini kullanıcıya göndermesi sağlanabilir. Aracın gezindiği harita bilgisi, haritaya hakim bir kamera yardımıyla belirli anlarda fotoğraflanıp, görüntü işleme teknikleri ile bu çalışmada ele alınan hücresel harita görünümü ve bilgisi elde edilebilir.

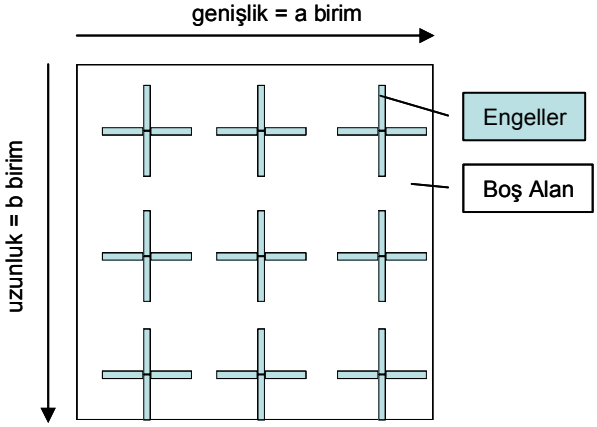
1. Giriş

Günümüzde kullanılan ve kimi zaman tehlike de içerebilen bir takım araçların (evdeki robotlardan fabrikalardaki forkliftlere, sokaktaki arabalardan askeri alanda kullanılan diğer araçlara) idare ve denetimlerini uzaktan yapılabilir hale getirerek yada bu araçları tamamen kendi kendilerine iş yapılabilir kılarak, insanların üzerine düşen idare ve denetim yükünün ve de tüm bunların getirdiği tehlikelerin azaltılması için yapılan çalışmalar teknoloji dünyasının başlıca çalışmaları arasında yer almaktadır.

Kendi kendine hareket edebilen araçlar, sabit bir yüzeye monte edilmiş olmadıkları için bütün koordinatları bağıldır ve ufak bir sapma yanlış veri işlenmesine ve dolayısıyla yanlış karar vermeye neden olabilir. Bu çalışmada, önce herhangi bir düzlem, boyutları belirli hücreler ile kaplanarak, o düzlemi temsil edebilecek bir harita meydana getirilmiş, bu harita üzerinde herhangi bir konumdan diğerine giden en kısa yol bulma algoritması geliştirilmiştir. Çalışırken daima haritanın neresinde olduğunu bilen araç, gitmesi istenen hedef bilgisini aldıktan sonra belleğinde bulunan harita üzerinde o hedefe varan en kısa yolu bulur ve o yol üzerinden harekete başlar. Araç, hareketi sırasında bulundurduğu engel algılayıcılar yardımı ile çevresindeki engellere çarpmadan hedefe varır. Hareket halinde, engel algılayıcılardan elde edilen bilgiler bellekteki harita ile çelişirse, algılayıcılardan elde edilen bilgiye göre bellekteki harita değiştirilerek, yeniden o konumdan hedef noktaya giden en kısa yol hesaplanır ve bu yol üzerinden harekete devam edilir.

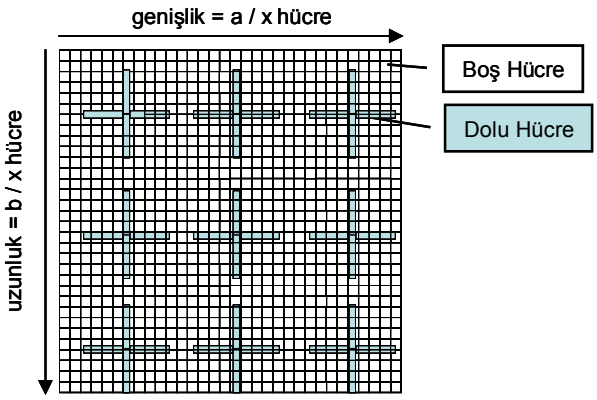
2. Harita

Çalışmada kullanılan harita, üzerinde gezinen aracın rahatça hareket edebildiği boş alanlara ve aracın çarpmaması gereken engellere sahiptir. *Şekil 1* de üzerinde engelleri barındıran boyutları a birim ve b birim olan bir örnek harita görülebilmektedir.



Şekil 1: Engellere Sahip Örnek Harita

Harita düzlemi, x birimlik kenar uzunluğuna sahip, birbirine eşit karesel alanlara (hücrelere) ayrıştırılırsa Şekil 2 deki gibi sayısal bir harita görünümü elde edilir. Engellere denk gelen hücreler dolu hücre, diğerleri boş hücre olarak isimlendirilir. Burada, hücrenin boyutunun küçülmesi (x değerinin azalması) harita üzerinde gezinen araca daha hassas hareket yeteneği kazandırırken, haritanın bellekte tutulması sırasında daha fazla bellek ihtiyacını veya en kısa yol bulma algoritmasının döndürülmesi sırasında daha fazla zaman ihtiyacını beraberinde getirmektedir.

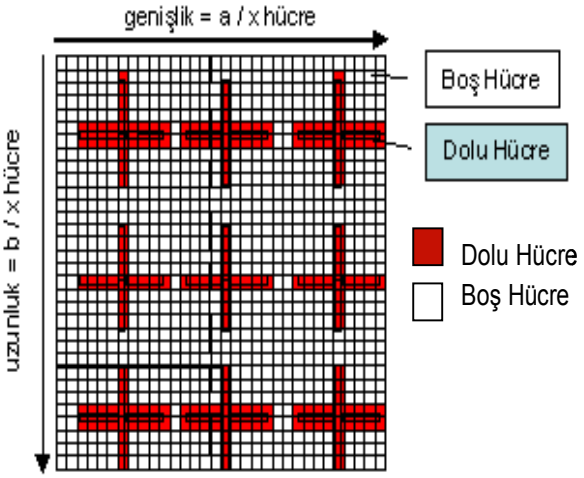


Şekil 2: Hücrelere Ayrıştırılmış Harita Örneği

Haritayı kaplayan toplam hücre sayısı (boyut) aşağıdaki formülle hesaplanabilir.

$$\text{boyut} = \frac{ab}{x^2} \quad (1)$$

Hücrelere ayrıştırılmış haritada dolu hücreler ve boş hücrelere göre tekrar şekillendirilirse *Şekil 3* 'teki gibi bir harita elde edilir. Dolu ve boş hücrelerden oluşan harita aracın belleğine bu şekilde yüklenerek, aracın harita üzerinde engellere çarpmadan hareket etmesi sağlanır.



Şekil 3: Hücrelerle Temsil Edilen Örnek Harita

3. En Kısa Yol Bulma

En kısa yol bulma problemlerinin en bilinen çözümleri Dijkstra, Bellman-Ford ve Floyd algoritmalarıdır. [1] numaralı çalışmada üretilen yeni bir algoritma ile (D* algoritması), değişebilen haritalarda optimum yolu bulabilme işlemi gerçekleştirilmiştir. Dijkstra'nın algoritmasında kaynak düğümden hedef düğüme doğru, her düğüm için kaynak düğüme göre maliyet hesaplaması yapıldıktan sonra, maliyetlerle oluşturulan komşuluk matrisi yardımıyla, kaynaktan hedefe varan tüm yollar taranarak içinden en kısa olan yol seçilir [2]. Bu çalışmada ise maliyet verme işlemi hedef düğüm merkez alınarak gerçekleştirilmiştir. Böylece kaynak düğümün hedef düğüme göre olan maliyeti elde edildiği anda hedefe olan en kısa yol, sürekli bir sonraki en düşük maliyete sahip olan düğüme adım atılarak elde edilir. Geliştirilen algoritmanın ilk aşaması, graf veri modeli yardımı ile *Şekil 4* ve (4) numaralı denkleme göre doldurulan *Tablo 1* 'de görülebildiği üzere, merkez düğümden komşu düğümlere, kenarların maliyetleri üst üste eklenerek yayılan bir yapıya sahiptir.

0,1,2,3… : düğüm numaraları (m veya n)

$K_{m,n}$: m düğümü ile n düğümü arasındaki kenar

$M_{m,n}$: $K_{m,n}$ kenarının maliyeti.

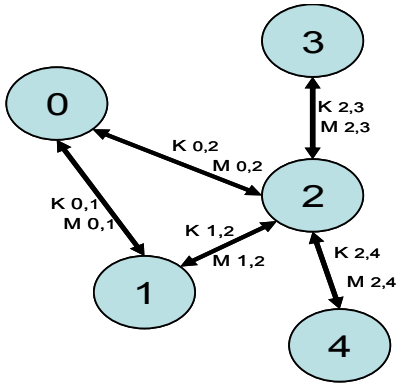
deger(n) : n düğümünün merkez düğüme göre maliyeti.

Kenarlar çift yönlüdür ve maliyetleri birbirine eşittir.

$$K_{m,n} = K_{n,m} \quad (2)$$

$$M_{m,n} = M_{n,m} \quad (3)$$

$$\text{deger}(n) = \text{deger}(m) + M_{m,n} \quad (4)$$



Şekil 4: Düğüm, Kenar ve Maliyetler

Merkez düğüm olarak 0 numaralı düğüm seçilirse her bir düğümün 0 numaralı düğüme göre değerleri (4) denklemine göre belirlenir. Değerler hesaplanırken, her bir düğüm tek tek taranır. Başlangıç durumunda, merkez düğüm dışında tüm düğümlerin değerleri sonsuza eşitlenir. Eğer bir düğüm birden fazla kenara sahipse her bir kenar için bir değer hesaplanır ve en küçük olan değer o düğümün merkez düğüme göre değeri olur. Bu şekilde tüm düğümler için değerler hesaplanarak *Tablo 1* oluşturulabilir.

Tablo 1: Merkez Düğüme Göre Maliyetler

deger(n)		**hesaplama**	**açıklama**
deger(0)	=	0	Merkez düğüm 0 maliyetlidir.
deger(1)	=	deger(0) + $M_{0,1}$ deger(2) + $M_{2,1}$	İki değerden en az maliyete sahip olan seçilir.
deger(2)	=	deger(0) + $M_{0,2}$ deger(1) + $M_{1,2}$ deger(3) + $M_{3,2}$ deger(4) + $M_{4,2}$	Dört değerden en az maliyete sahip olanı seçilir.
deger(3)	=	deger(2) + $M_{2,3}$	3 numaralı düğüm bir kenara sahiptir.
deger(4)	=	deger(2) + $M_{2,4}$	4 numaralı düğüm bir kenara sahiptir

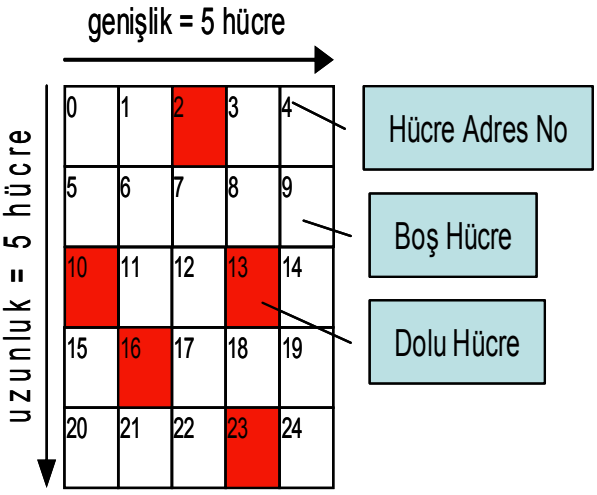
Araç, en kısa yol bulma algoritmasını belleğinde bulunan hücrelerden oluşan harita üzerinden döndürür. Graf veri modelindeki düğümler hücre merkezleri ile, kenarlar da hücreler arasındaki geçişler ile eşleştirilirse aracın belleğindeki haritanın da graf veri modeli çıkarılabilir. *Şekil 5* 'teki örnek harita modellendiğinde görülür ki; boş hücreler arası geçişler, haritanın tüm hücreleri için eşit maliyete sahipken herhangi bir dolu hücreye geçiş olanağı yoktur ya da maliyeti sonsuzdur.

Genişliği; a = 250 cm, uzunluğu; b = 250 cm olan bir örnek harita, bir kenar uzunluğu; x = 50 cm olan birim hücrelerle kaplanırsa ve her bir hücreye adres numarası verilirse *Şekil 5* 'teki örnek harita elde edilir. Harita, araç belleğinde dizi şeklinde tutulduğundan, *Şekil 5* 'teki haritada, hücrelerin adres numaraları bellekteki dizi indislerine, hücrelerin içerikleri (dolu veya boş olması) dizideki değerlerine denk gelmektedir.

$$\text{genislik} = \frac{a}{x} = \frac{250\,\text{cm}}{50\,\text{cm}} = 5 \ldots \text{hücre}$$

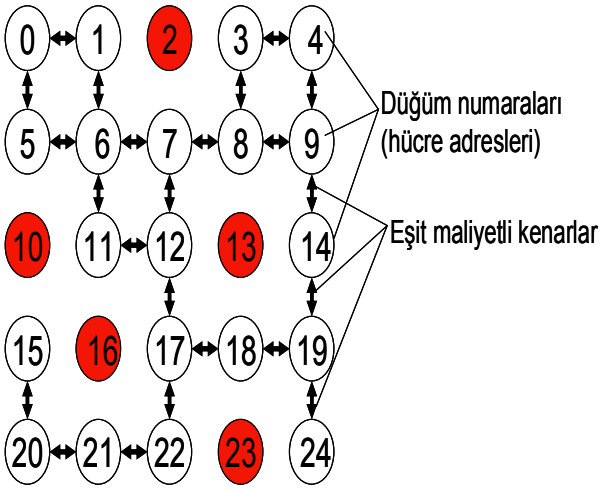
$$\text{uzunluk} = \frac{b}{x} = \frac{250\,\text{cm}}{50\,\text{cm}} = 5 \ldots \text{hücre}$$

$$\text{boyut} = \frac{ab}{x^2} = \frac{62500\,\text{cm}^2}{2500\,\text{cm}^2} = 25 \ldots \text{hücre}$$



Şekil 5: Örnek harita

Şekil 5'teki örnek haritanın graf modeli şekil 6 'daki gibidir.

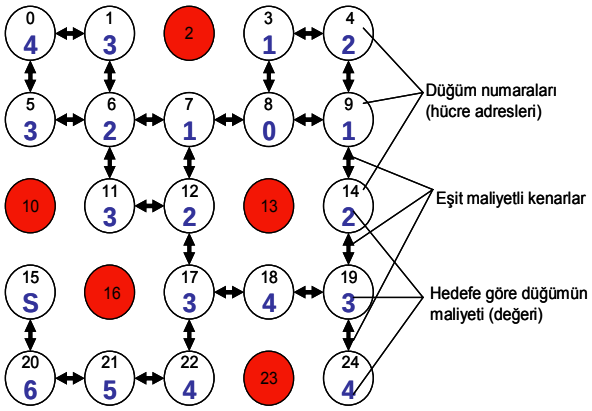


*Şekil 6 :*Örnek Haritanın Graf Veri Modeli

Araç, kullanıcıdan, gitmesi istenilen hedef hücre bilgisini aldıktan sonra hedefe varan en kısa yolu bulmak için önce, hedef hücreyi merkez düğüm seçerek haritadaki diğer hücrelere (düğümlere) hedefe göre maliyet (değer) verir. Bu işlem, aracın içinde bulunduğu hücrenin hedefe olan maliyeti belirlenene kadar devam eder.

Örneğin, *Şekil 5*'teki örnek harita üzerinde 20 numaralı hücrede bulunan araçtan, 8 numaralı hücreye gitmesi istenilirse, öncelikle, 8 numaralı düğüm merkez olmak şartıyla diğer komşu düğümler, hedefe göre değerlendirilir. Bu işlem 20 numaralı düğümün merkez düğüme göre değeri belli olduğunda son bulur. Haritadaki her kenarın maliyeti birbirine eşittir.

$M_{m,n} = M_{n,m} = 1$ kabul edilirse, (4) denklemi yardımı ile, düğüm değerleri hedef düğüme göre düzenlenen harita modeli *Şekil 7* 'deki gibi olur.

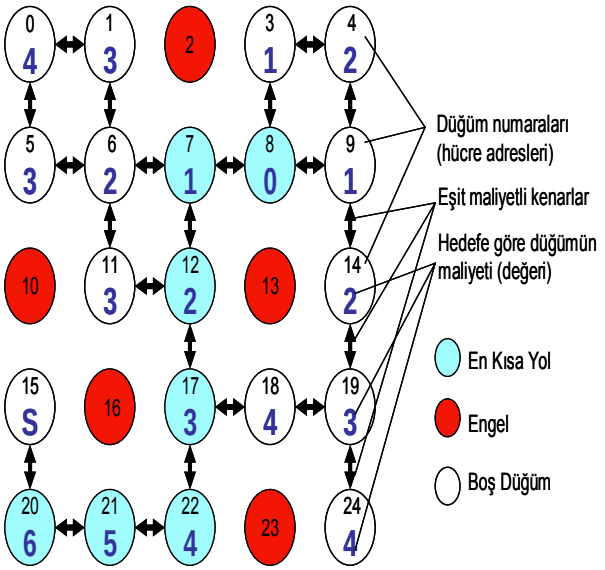


Şekil 7: En Kısa Yol Bulma-1

Şekilden de görüldüğü gibi aracın üzerinde bulunduğu konum hücrenin (20 numara) hedefe ne kadar uzakta olduğu bilgisi artık elimizdedir. Bir hücreden diğerine geçiş adım olarak isimlendirilirse 20 numaralı hücreden 8 numaralı hücreye 6 adımda varılmaktadır. 15 numaralı hücrenin hedefe göre maliyeti sonsuz anlamında S harfi ile gösterilmiştir. Çünkü 20

numaralı hücrenin hedefe göre durumu bulunduktan sonra maliyet verme işlemi sona ermiştir.

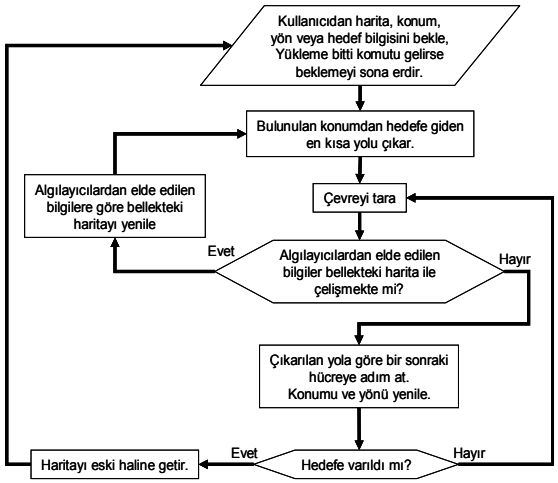
En kısa yolu hesaplamanın ikinci aşaması, üzerinde bulunulan konum hücreden, maliyeti en düşük olan hücreye adım atarak hedefe varmaktan ibarettir. Bu şekilde hedefe en kısa yoldan varılır (*Şekil 8*).



Şekil 8: En Kısa Yol Bulma-2

4. Hareket

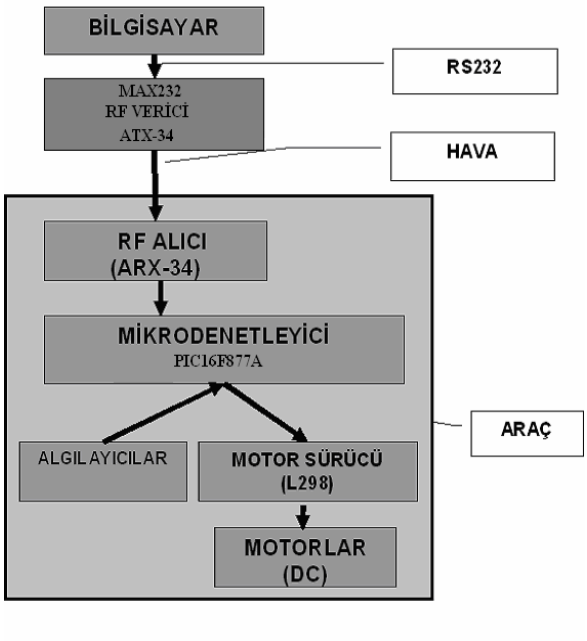
Araç harekete başlamadan önce kullanıcıdan, gezindiği haritanın şekli, sahip olduğu yön bilgisi, bulunduğu konum hücre bilgisi ve gideceği hedef hücre bilgisi gibi bir takım komut veya veriler bekler. Kullanıcının yüklemeyi bitirdim anlamındaki komutuyla birlikte araç bulunduğu konumdan hedefe giden en kısa yolu çıkarır. Sonra bulundurduğu sensörler aracılığıyla çevresindeki engelleri tarar. Eğer sensörlerden elde ettiği engel bilgileri belleğindeki harita ile çelişirse belleğindeki haritada geçici olmak üzere yenileme yaparak tekrar bulunduğu konumdan hedefe yeni bir yol çıkarır, çelişki olmadığında ise daha önce çıkardığı yola göre sıradaki hücreye adım atarak, belleğinde konum ve yön bilgilerini yeniler. Bu işlem her adımda tekrarlanır. Her adımda hedefe varıp varmadığını kontrol eden araç, hedefe vardığında belleğindeki harita üzerinde yapmış olduğu geçici değişikleri iptal ederek haritayı eski haline getirir. Aracın hareket algoritması aşağıdaki gibidir.



Şekil 9: Hareket Algoritması

5. Donanım

Kullanıcı tarafında, bilgisayarın RS232 portu çıkışına, MAX232 tümdevresi ile bağlanan ATX-34 RF verici birimi [2], bilgisayardan gelen verileri modüle ederek ortama yayar. Araç tarafında ise ARX-34 RF alıcı birim[3] ile alınan veriler demodüle edilerek, mikrodenetleyici olarak kullanılan PIC16F877A 'nın Usart girişine gönderilir [4]. Mikrodenetleyici aldığı verilere göre hesaplamalar yaptıktan sonra DC motor sürücü tümdevre; L298 aracılığı ile hareketi başlatır ve yönetir. Aynı zamanda aracın çevresine yerleştirilen SHARP GP2D120X sensörleri [5], aracın etrafındaki engel bilgisini mikrodenetleyiciye iletir.



Şekil 10: Donanım

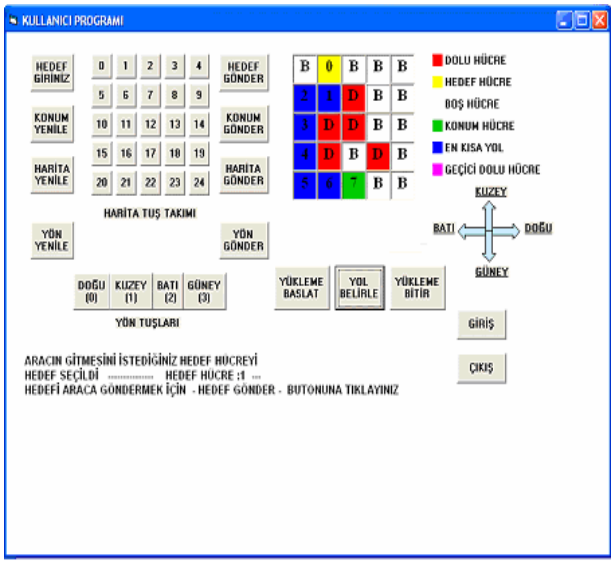
6. Yazılım

Çalışma gerçekleştirilirken iki farklı yazılım dilinden yararlanılmıştır. Kullanıcı tarafında arayüz programı oluşturulurken, visual basic kullanılmıştır. Mikrodenetleyiciyi çalıştırmak için de MikroC dilinden yararlanılmıştır [6]. Öğrenim ve kullanım kolaylığı, PIC16F877A ile uyumlu geniş bir kütüphaneye sahip olması MikroC'nin tercih edilme nedenidir.

6.1. Kullanıcı Arayüz Ekranı

Kullanıcının, değişebilen harita bilgisini, konum, yön ve hedef bilgilerini, bilgisayar ekranında görsel olarak takip edip araca gönderebilmesi için visual basic kullanılarak bir arayüz ekranı oluşturulmuştur.

En kısa yol bulma algoritması, kullanıcı tarafındaki programa da yazılmıştır. Böylece, haritada kullanıcının haberi olmayan bir değişiklik olmadığı takdirde, kullanıcı, aracın gideceği en kısa yolu kullanıcı arayüz ekranındaki haritada görebilmektedir. Şekil 11'de kullanıcı arayüz ekranı görülmektedir.



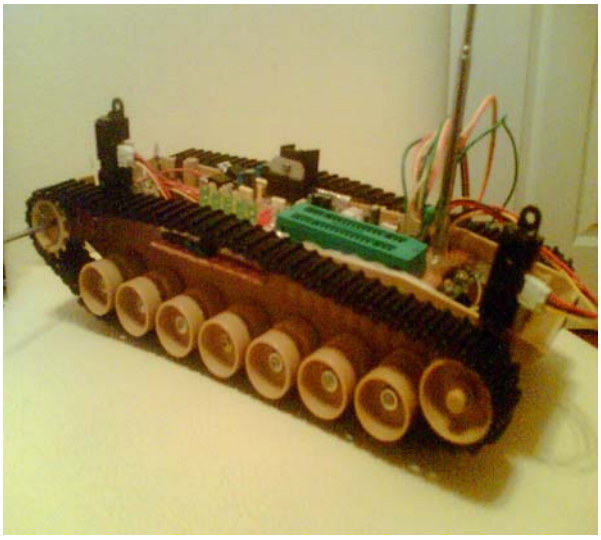
Şekil 11: Kullanıcı Arayüz Ekranı

7. Tartışma

Bundan sonraki aşamalarda aracın çevresindeki engel bilgisini daha detaylı algılaması için algılayıcılar artırılabilir veya değiştirilebilir. RF iletişim çift yönlü yapılarak aracın her adımda çevresindeki engel bilgisini kullanıcıya göndermesi sağlanabilir. Aracın gezindiği harita bilgisi, haritaya hakim bir kamera yardımıyla belirli anlarda fotoğraflanıp, görüntü işleme teknikleri ile bu projede ele alınan hücresel harita görünümü ve bilgisi elde edilebilir. Harita karesel değil de altıgen hücrelerle kaplanabilir böylece bir hücreden daha fazla hücreye bir adımda geçiş olanağı sağlanabilir. Araçların hareket matematiği elde edilip projeye yüklendikten sonra herhangi bir araç için kullanılabilir halde proje genişletilebilir.

8. Sonuçlar

Bu çalışmada, daha önce belirlenmiş bir harita üzerinde belirlenen herhangi bir konumdan istenilen bir konuma kendi kendine en kısa yolu bularak gidebilen bir araç için farklı bir yazılım yazılmış, baskı devresi gerçekleştirilmiş ve tank benzeri model bir araç üzerinde de uygulanmıştır (Şekil 12). Aracın güzergah üzerinde hareketi sırasında haritada belirlenmeyen bir engelin sonradan önüne çıkarılması veya engelin sonradan kaldırılabilmesi durumunda araç kendi kendine yeni güzergahını tekrar belirleyebilmektedir, bu da aracın akıllı bir sistem olabilme yeteneğini göstermektedir.



Şekil 12: Model Araç

9. Kaynakça

[1] "Stentz, A, Optimal and Efficient Path Planning for Partially-Known Environments, IEEE International Conference on Robotics and Automation, 1994. Proceedings., 1994, 8-13 May 1994 Page(s):3310 - 3317 vol.4"

[2] Çölkesen R., "Veri Yapıları ve Algoritmalar", Papatya Yayınları, İstanbul, 2006.

[3] ATX34 UHF ASK Data Transmitter Ürün Kılavuzu, Eylül 2006, www.udea.com.tr

[4] ARX34 UHF ASK Data Receiver Ürün Kılavuzu, Aralık 2004, www.udea.com.tr

[5] Şahin H., Dayanık A., Altınbaşak C., "PIC Programlama Teknikleri ve PIC16F877A", Altaş Yayıncılık, İstanbul 2006.

[6] SHARP GP2D120X sensör ürün kılavuzu http://www.acroname.com/robotics/parts/GP2D120_SS.pdf.

[7] MikroC User's Manual www.mikroe.com/pdf/**mikroc**/**mikroc**_manual.pdf