T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

MERKEZİ KONTROLLÜ BİREYSEL TAŞIT SİSTEMİ

YÜKSEK LİSANS TEZİ

ERKAN AKIN

DENİZLİ, ARALIK - 2018

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

MERKEZİ KONTROLLÜ BİREYSEL TAŞIT SİSTEMİ

YÜKSEK LİSANS TEZİ

ERKAN AKIN

DENİZLİ, ARALIK - 2018

KABUL VE ONAY SAYFASI

Erkan Akın tarafından hazırlanan "**Merkezi Kontrollü Bireysel Taşıt Sistemi**" adlı tez çalışmasının savunma sınavı 24.12.2018 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Prof. Dr. Abdullah Tahsin TOLA	.....
Üye Prof. Dr. Abdurrahman ÜNSAL Kütahya Dumlupınar Üniversitesi	.....
Üye Doç. Dr. Selim KÖROĞLU Pamukkale Üniversitesi	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 09/01/2019 tarih ve ..02../..14... sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Uğur YÜCEL

Fen Bilimleri Enstitüsü Müdürü

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

ERKAN AKIN

ÖZET

MERKEZİ KONTROLLÜ BİREYSEL TAŞIT SİSTEMİ
YÜKSEK LİSANS TEZİ
ERKAN AKIN
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, ARALIK - 2018

Şehirlerdeki en büyük sorunlardan birisi trafik yoğunluğudur. Trafik yoğunluğu nedeniyle birçok insan trafikte boşa zaman harcamaktadır. Bu tezde son zamanlarda trafikte geçen sürelerin daha verimli kullanılabilmesi için önerilen MetroCar Sisteminin test edilmesi amaçlanmıştır. Bunun için özel olarak yapılmış yollarda otonom olarak ilerleyen tek kişilik arabalardan oluşan bir sistem düşünülmüştür. Sistemdeki her araç sistem tarafından kontrol edilmektedir. MetroCar Sistemi için yeni önerilen takip ve birleşme algoritmaları merkezi kontrollü bireysel taşıt sistemi için test edilmiştir. Bu testler 1:10 oranında küçültülmüş araçlar ve 1:10 oranında küçültülmüş yollar ile yapılmıştır. Testler yapılırken araçlardan alınan anlık hız ve konum bilgileri kullanılmıştır. Her bir araç hız ve konum bilgilerini wireless modüller aracılığıyla merkezi bilgisayara göndermiştir. Geliştirilen görsel program yardımıyla bu veriler işlenmiştir ve araçlara hızlanma veya yavaşlama komutları gönderilmiştir. Araçların tüm hareketlerinde sabit ivmeli hareket kullanılmıştır. Bu algoritmaların MetroCar Sistemlerinde kullanılabileceği gösterilmiştir.

ANAHTAR KELİMELER: MetroCar Sistemler, Otonom Araçlar, Akıllı Araçlar, Trafik Sistemleri

ABSTRACT

CENTRALLY CONTROLLED INDIVIDUAL VEHICLES SYSTEM
MSC THESIS
ERKAN AKIN
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
(SUPERVISOR:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, DECEMBER 2018

One of the biggest problems in cities is traffic density. Due to traffic density, many people are wasting time in traffic. In this thesis, it is aimed to test recently proposed MetroCar System to use the time spent in traffic more efficiently. For this purpose, a system consisting of autonomous single-seat cars which runs on specially built roads is considered. Each vehicle in the system is controlled by the system. Recently proposed tracking and merging algorithms for MetroCar Systems have been tested for a centrally controlled individual vehicles. These tests were carried out with 1:10 minimized vehicles and 1:10 minimized roads. The instantaneous speed and position information from the vehicles are used during the tests. Each vehicles sends speed and location information to the central computer via wireless modules. The data is processed by developed visual program and acceleration or deceleration commands are sent to vehicles. In all movements of the vehicles, constant acceleration movement is used. It has been shown that these algorithmes can be used for MetroCar Systems.

KEYWORDS: MetroCar System, Autonomous Vehicles, Smart Vehicles, Traffic System

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

ÖNSÖZ

Uzun süredir üzerinde çalıştığım, kafa yorduğum, araştırdığım projemi bitirmiş olmanın mutluluğunu yaşıyorum.

Tez projem boyunca her türlü sorunumda yanımda olan danışmanım Sayın Prof. Dr. Abdullah Tahsin TOLA'ya sonsuz teşekkürlerimi ve şükranlarımı sunuyorum.

1. GİRİŞ

Günümüz dünyasında en büyük problemlerden birisi olarak trafik yoğunluğu gösterilebilir. Bu yoğunluğa bağlı olarak ortaya çıkan zaman kaybı da aynı şekilde büyük bir sorun olarak göze çarpmaktadır. Bu yoğunluğun azaltılması ve insanların trafikte geçirdikleri süreleri daha verimli kullanabilmeleri üzerine birçok çalışma yapılmaktadır. İlk zamanlarda toplu taşıma ile bu sorun bir nebze olsun çözülmüştür. Ancak sonrasında nüfusun hızlı artışı ile toplu taşıma da konfor ve zaman yönetimi konusunda olumlu sonuçlar vermemeye başlamıştır. Sonrasında tamamen kendi kararları ile hareket eden otonom arabalar piyasaya çıkmaya başlamıştır ve hâlâ bu konuda birçok çalışma yapılmaktadır. Ancak meydana gelen kazalar ve ölümler otonom araçlara olan güveni sarsmaya başlamıştır. Hem toplu taşımanın hem de otonom araçların iyi yönlerini alarak ortaya çıkarılabilecek bir bireysel taşıt sistemi ile kişilerin belli güzergâhlarda olan özel yollarda araç kontrolünü sisteme devrederek kendi özel işlerini konforlu bir yolculukta yapmaları sağlanabilecektir. Yapılan bu çalışma ile MetroCar Sistemi adı verilen böyle bir sistemde araçların takip algoritmaları ve tali yoldan ana yola bağlanma algoritmaları test edilmiş ve olumlu sonuçlar elde edilmiştir.

1.1 Literatür Taraması

Dünya üzerindeki araştırmacılardan birçoğu otonom araçlar konusunda yapılan çalışmaların 1939 yılında General Motors tarafından düzenlenen Futurama Dünya Fuarı ile başladığını kabul etmektedir. Fuarda tanıtılan aracın, şehir içerisinde insanları güvenli bir şekilde otomatik olarak taşıyan bir araç olduğu belirtilmiştir (Özgüner 2011).

1960'lı yıllarda İngiltere'de United Kingdom's Transport ve Road Research Laboratory tarafından ilk kez sürücüsüz bir araç testi gerçekleştirileceği duyurulduğunda bu fikrin gerçekleşebileceğine inanan insan sayısı oldukça az olmasına rağmen sürücüsüz araç manyetik kablolarla döşenmiş olan yolda saatte 130

km hıza kadar çıkarak hareketini tamamlayınca ilk sürücüsüz araç denemelerinden birisi gerçekleşmiştir (WEB_1).

1980 yılında Mercedes-Benz mühendisi Ernst Dickmanns'ın tasarlamış olduğu robotik araç ise ilk sürücüsüz araçlardan birisi olarak gösterilebilir. Dickmanns'ın tasarlamış olduğu bu araç kamera görüntülerinin işlenmesi ile hareketini sağlamaktadır. Projede görüntü işlemesi için kullanılan bilgisayar ile araç arasında uzun manyetik kablolar kullanılmıştır. Yapılan bu çalışmalar ile sürücüsüz araçların geleceğe yön verecek bir teknoloji olduğunun kanıtlanması sağlanmıştır (Özgüner 2011).

1986 yılında NevLab 1 ilk kez kablo ihtiyacı olmaksızın bilgisayarlı aracı yapmayı başarmıştır. Mavi bir karavan içerisine jeneratör kurulması ile araç üzerine bilgisayar ve sensörler yerleştirilmiştir. Maliyeti 1 milyon doları bulan bu araç sürücüsüz olarak saatte 32 km hızla gitmeyi başarmıştır (WEB_1).

1995 yılında ise NevLab projesi araçları %98.2 verim ile 5000 km yol almıştır. Ancak bu araçlarda gaz ve fren pedalları insan gücü ile kontrol edildiği için yarı sürücüsüz araçlar olarak tanımlanmaktadırlar (Lari 2014).

1995 yılında Mercedes-Benz mühendisi Ernst Dickmanns S-sınıfı bir Mercedes-Benz'in 1600 km'lik yolu almasını sağlamıştır. Yol boyunca araç %95 oranında otonom olarak hareket etmiş ve saatte 175 km hıza kadar çıkabilmiştir (Lari 2014).

2001 yılında Paruchuri ve arkadaşları tarafından yapılan çalışma ile kavşakta düzensiz trafik için simülasyon gerçekleştirilmiştir. Bu projede amaç araçların trafik ışıkları ya da trafik polisleri gibi bir sistem olmadan kavşakta sorunsuz olarak hareket edebilmeleridir (Lari 2014).

2002 yılında ABD Savunma Bakanlığı tarafından Grand Challenge adında bir otonom araç yarışması düzenlenmiştir. Yapılan bu yarışmaya katılan hiçbir takım parkuru tamamlamayı başaramamıştır (Lari 2014).

Yine ABD Savunma Bakanlığı tarafından 2005 yılında yapılan yarışmada ise 217 km'lik rotayı 5 takım tamamlamayı başarmıştır (Lari 2014).

Parma Üniversitesi bünyesinde çalışmalar yapan Alberto Broggi kendi yapmış olduğu 4 adet sürücüsüz elektrik arabasını, İtalya'dan Çin'e 13.000 kilometre yol yaparak göndermeyi başarmıştır (Forret 2007).

1999 yılında Hollanda'nın Rotterdam kentinde ParkShuttle adında bir sistem ile sürücüsüz araçlar ile yolcu taşımacılığı yapılmaktadır. Günde 2000 yolcu taşıma kapasiteli bu sistem sürücüsüz araçların gündelik hayatta kullanımına en büyük örnek olarak gösterilebilir (CDIAC 2008).

2001 yılında Paruchuri ve arkadaşları kavşak bazında bir çalışma yapmışlardır. Yapılan bu çalışma kavşak bazında olup düzensiz trafik için çoklu etmenli simülasyon gerçeklemişlerdir. Bu çalışmalarında asıl amaç araçların trafik ışığı, trafik polisi vb. gibi etmenler olamadan hareket edebilmelerini sağlamaktır (Paruchuri 2001).

2002 yılında Hernández tarafından yapılan çalışmada Barcelona çevresindeki otoyol ağında gerçek zamanlı trafik için çok ajanlı bir sistem önerilmiştir. Bu sistemlerde trafik sorunlarına karşı akıl yürütme tekniklerini kullanmaktadır. Değerlendirme sonucunda yapay zeka tekniklerinin kullanımının trafik yönetim sistemleri geliştirmek için yararlı olabileceğine dair sonuçlar elde edilmiştir (Hernández 2002).

2005 yılında Oliveira ve Duarte tarafından yapılan çalışma ile çoklu etmen sistemi gerçekleştirilmiştir (Oliveira 2005).

2006 yılında Chuang ve Kung tarafından yapılan çalışma ile en kısa yol problemi için yeni bir algoritma çalışması yapılmıştır. Bu algoritmada en kısa yol hesabı için bulanık benzerlik ölçümü kullanılmaktadır (Chuang 2006).

2012 yılında Güvez ve arkadaşları il içerisinde tıbbi atık toplama problemi için uygun rota bulma konusunda çalışmalar yapmışlardır (Güvez 2012).

2012 yılında Zohdy ve Rakha ışıksız ve kontrollü bir kavşakta araçların bekleme sürelerini düşüren bir çalışma yapmışlardır. Bu çalışmada Monte Carlo yöntemi kullanılmıştır (Zohdy 2012).

2013 yılında Bayzan tarafından araç konumunun belirlenmesi için çalışma yapılmıştır. GPS ve merkezi veritabanı ile yapılan bu çalışma başarılı sonuçlar vermiştir (Bayzan 2013).

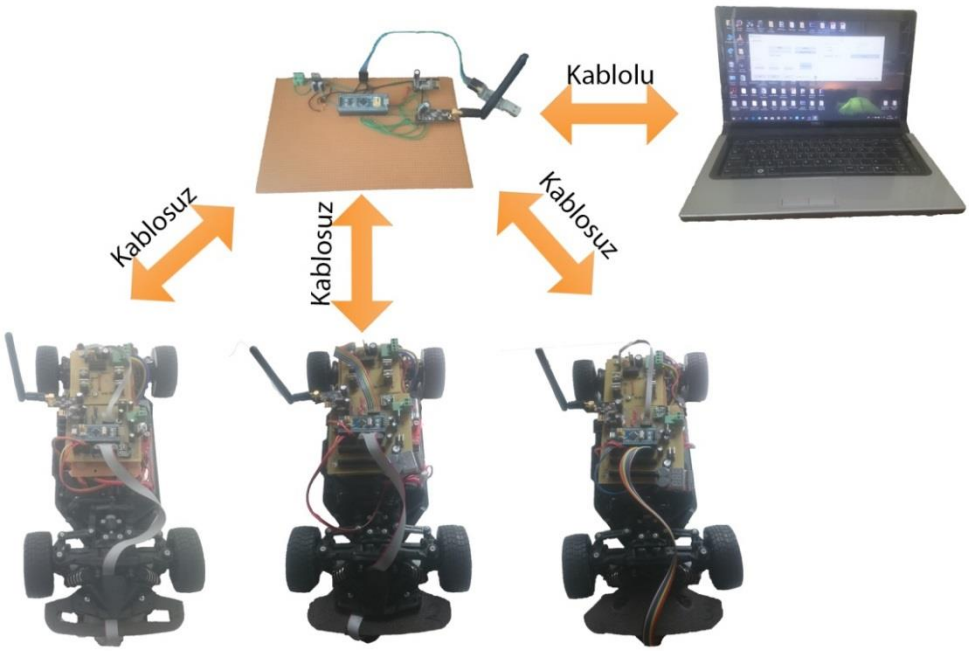
2018 yılında Gökozan ve Taştan tarafından akıllı araçlar ve bu akıllı araçların kontrolleri konusunda çalışmalar yapılmıştır (Gökozan 2018).

2018 yılında Gökaşar ve Dündar tarafından sürücüsüz taşıtların trafik akım hızına etkileri konusunda çalışma yapılmıştır. Yapılan bu çalışmada yapay sinir ağları kullanılmıştır (Gökaşar 2018).

Sürücüsüz araçlar konusunda başta Tesla, General Motors, Volkswagen, Audi, BMW, Volvo ve Google gibi büyük kuruluşlar olmak üzere birçok büyük otomobil ve teknoloji şirketleri tarafından çalışmalar yapılmaktadır (WEB_2).

1.2 Tez Tanıtımı

Proje süresince MetroCar Sisteminde araçların takip ve birleşme algoritmalarının testi üzerine çalışmalar yapılmıştır. Bunun için gerçek araçlara göre 1:10 oranında küçültülmüş 3 adet RC model araba kullanılmıştır. Araçların haberleşmesi için bir adet haberleşme kartı ve tüm kontrol işlemlerinin yapılabilmesi için bir adet bilgisayar kullanılmıştır.



Şekil 1.1: Projede Kullanılan Bilgisayar, Haberleşme Kartı ve Araçların Bağlantı Şekli

Şekil 1.1'de kullanılan araçlar, haberleşme kartı ve bilgisayar ile genel olarak haberleşmenin nasıl yapılacağı gösterilmiştir.

Şekil 1.1'den de anlaşılacağı üzere bilgisayarda yazılmış olan C# programı ile haberleşme kartına kablolu bağlantı gerçekleştirilecek ve bu haberleşme kartı ile araçların konum ve hız bilgileri alınacaktır. Bilgisayarda işlenecek olan bu veriler neticesinde araçlara hızlanma, yavaşlama vb. gibi bilgiler kablosuz olarak gönderilecektir. Böylelikle araçların takip ve birleşme algoritmalarını yapmaları sağlanacaktır.

Projede ikinci bölümde öncelikle MetroCar Sisteminden bahsedilecektir. Sonrasında ise projede kullanılan malzemeler ve malzeme özellikleri konusunda bilgilere yer verilecektir. Bir sonraki bölüm olan üçüncü bölümünde projede kullanılan sensör, devre kartı vb. gibi elemanların hazırlanması ve araç üzerine montajı anlatılacaktır. Dördüncü bölümde proje için oluşturulan C# programından ve kullanılan programın yazılımından, haberleşme kartı yazılımından ve araç kontrol kartı yazılımından bahsedilecektir. Son olarak beşinci bölümde ise proje boyunca elde edilen verilerden bahsedilecektir.

2. TEMEL KAVRAMLAR

2.1 Giriş

Bu bölümde öncelikle MetroCar Sistemlerinin tanıtımı yapılacaktır. Sonrasında ise projede kullanılan malzemeler ile ilgili genel bilgilere yer verilecektir.

2.2 MetroCar Sistemleri

MetroCar Sistemi olarak tanımladığımız sistem insanların trafikte geçirdikleri süreleri daha verimli geçirmelerini sağlayan ve toplu taşıma ile şahsi araçlarla seyahatin verimli taraflarını içerisinde barındıran bir sistemdir. Şehir içerisinde belli bir güzergâhta kurulabilecek olan bu sistem ile insanlar evden şahsi araçlarıyla yola çıkacaklar, belli bağlantı noktalarından bu yola bağlanabilecekler ve gitmek istedikleri yere en yakın bağlantı noktasından çıkarak hedefledikleri yere ulaşabileceklerdir. Araçların MetroCar Sistemine girmesi ile kontrol, sistem tarafından devralınacak ve yol boyunca araçlar otonom araçlar gibi hareket edeceklerdir. Böylelikle araç sürücüleri trafikte geçen bu zamanı kitap okuyarak, yemek yiyerek hatta uyuyarak geçirebileceklerdir.

MetroCar Sisteminde sisteme girebilecek olan araçların tek tip ve normal araçlara göre daha küçük boyutlarda, tek kişilik araçlardan oluşması planlanmaktadır. Aynı şekilde bu araçların elektrik enerjisini kullanacağı düşünülmektedir. Elektrikli olan bu araçlar sisteme girdiklerinde şarj edilmelerine de olanak sağlanacaktır.

MetroCar Sistemlerinin neden bir ihtiyaç olarak ortaya çıktığını anlayabilmek için öncelikle günümüzde kullanılan sistemlere göz atmamız gerekmektedir. Günümüzde günlük şehir içi seyahatlerde en çok kullanılan seyahat araçları olarak şahsi araçlar, minibüsler, otobüsler, taksiler, tramvaylar ve metrolar gösterilebilir.

Otobüsler ve minibüsler sağladıkları yararlar ve meydana gelebilecek zararlar yönünden ele alındığında aynı kategoride gösterilebilirler. Otobüsle yapılan

yolculuklarda ulaşım giderinin azlığı, park sorunu olmaması pozitif olarak öne çıkmaktadır. Ancak özellikle büyük şehirlerde çok fazla insan tarafından kullanılan otobüsler ve minibüslerde aşırı yoğunluktan dolayı ayakta yolculuk etme durumu dâhi ortaya çıkmaktadır. Eğer oturulacak yer bulunabilse kitap okumak, gazete okumak, maillere bakmak gibi günlük işler de yerine getirilebilir. Ancak bununla birlikte çok sayıda insanla yolculuk edileceği için kiminle yan yana oturulacağı bile belli değildir. Bu da beraberinde hijyen sorunlarını ortaya çıkarmaktadır. Bu araçların kendilerine has bir yolu olmadığı için şehir içi trafiğine girmek zorunda kalacaklar ve zaman yönünden kayıplar yaşanacaktır.

Tramvaylar da aynı minibüs ve otobüsler gibi şehir içinde belli bir yol üzerinde hareket eden toplu taşıma araçlarıdır. Artı ve eksi yönlerine bakıldığında ulaşım için ödenen ücret düşük olmakla birlikte park sorunu gibi strese neden olan bir problemi de bulunmamaktadır. Otobüs ve minibüslerde olduğu gibi oturulacak yer bulunabildiğinde ufak tefek günlük işlerin halledilebilmesi de pozitif yönlerinden birisi olarak öne çıkmaktadır. Bu yararlarının yanında trafik sorunu, hijyen sorunu ve kurulum maliyetleri negatif yönleri olarak gösterilebilmektedir.

Metro ile yapılan toplu taşımacılığa baktığımızda diğer toplu taşıma sistemlerinin pozitif yanlarını aldığını ve negatif yönlerini azaltmayı amaçlandığı görülmektedir. Diğer toplu taşımalarla ortak olan taşıma ücretinin makul seviyelerde olması pozitif yönü olarak öne çıkmaktadır. Kendisine ait bir yola sahip olması trafik karmaşasından ayrılmasını ve diğer taşıma sistemlerinde olan trafikte kaybedilen sürelerin ortadan kalkmasını sağlamaktadır. Bu pozitif taraflarına rağmen yine çok sayıda insanla yolculuk edileceği için ortaya çıkan hijyen problemi ve çok büyük boyutlarda olan kurulum maliyetleri negatif yönler olarak ön planda yer almaktadır.

Toplu taşıma yerine normal araçlarla seyahat etmek istersek de iki seçenek karşımıza çıkmaktadır. Bunlar taksi ile seyahat ve şahsi araçlar ile seyahattir. Taksiler konfor ve hijyen yönünden bakıldığında neredeyse şahsi araçlar kadar iyidirler. Taksi ücretleri negatif olarak öne çıksa da en önemli olumsuzluğu trafik karmaşası ve trafikte kaybedilen zamanın çok fazla olması olarak öne çıkmaktadır.

Aynı şekilde şahsi araçların şehir içi ulaşımda kullanılmaları trafik yoğunluğunu arttırıcı bir durumdur. Aracın yıllık bakımları, vergileri ve diğer

giderleri ile birlikte yakıt masrafı da hesaba katıldığında maddi olarak negatif olduğu görülmektedir. Toplu taşımada ve takside olduğu gibi gazete, kitap, dergi okumak ya da maillere bakmak gibi günlük işlerini yapma fırsatı da olmayacaktır. Yoğun trafikte araç kullanmanın getirdiği stres ile birlikte trafikte kaybedilen zaman da negatif yönler olarak öne çıkmaktadır. Aynı şekilde park sorunu da negatif bir yön olarak sayılabilir. Neredeyse tek pozitif taraf olarak konfor ve hijyen gösterilebilir.

Bütün bu şehir içi seyahat yöntemlerine baktığımızda hepsinin artı ve eksi yönleri olduğunu görebilmekteyiz. Bu yüzden bu sorunları ortadan kaldıracağı düşünülen MetroCar Sisteminde insanlar araç kontrolü sisteme devredildikten sonra günlük özel işlerini yapabileceklerdir. Tek kişilik araçlar kullanılacağı için hijyen yönünden de bir sorun teşkil etmeyecektir. Yol boyunca araçların şarj edilmesi ile seyahat esnasında yol masrafı en aza indirgenmiş olacaktır. Kendisine ait yollarda sistem kontrolünde araçların hareketi sağlanacağından hem trafik karmaşası olmayacak hem de bir kaza durumundan bahsedilemeyecektir. Tek negatif yönü olarak gösterilebilecek kurulum maliyetleri göz ardı edildiğinde şehirde yaşayan insanlar için oldukça yararlı bir sistem olacağı düşünülmektedir.

2.3 RC Model Arabalar

Projenin en önemli parçası olarak kullanılan arabalar gösterilebilir. Öncelikle model arabalar hakkında bilgi vermek doğru olacaktır. Model arabalar genellikle hobi amaçlı olarak kullanılmaktadır. Projede kullanılan arabanın belirlenmesi için yapılan araştırma sırasında model arabaların birçok çeşidi ve geniş bir kullanım alanı olduğu görülmüştür. 1:10, 1:8 veya 1:5 gibi ölçeklerde üretimi yapılabilmektedir. Bu ölçekler model arabanın gerçek arabaya göre boyutunu belirtmektedir. Örneğin 1:10 boyutu gerçek bir arabanın onda bir oranında küçültülmüş hâli olarak düşünülebilir. Yakıt olarak benzinli, nitro yakıtlı veya elektrikli türleri mevcuttur. Genel anlamda oyuncak arabalar ile sık sık karıştırılırlar fakat RC araçlar oyuncak sınıfına girmez. Oyuncak arabalara göre daha profesyonel sistemlere sahiptirler. Montajı yapılmamış şekilde kit olarak alınır ve parça parça birleştirilirler. RC denmesinin sebebi de radyo kontrollü olmasındandır. Genel anlamda RC model arabalar yol arabaları ve arazi arabaları şeklinde iki ana başlığa ayrılabilir.

2.3.1 Arazi Tipi RC Model Arabalar

Arazi tipi RC model arabalar genel olarak zorlu ve engebenin çok olduğu arazi şartlarına göre tasarlanmış ve üretilmiş araçlardır. Ulaşabildikleri hızlar doğal olarak kısıtlıdır. Çok yüksek hızlara çıkamazlar. Bu tip araçlarda yüksek tork değerine sahip motorlar kullanılmaktadır. Tekerlekleri ve süspansiyonları da arazi şartlarına daha uygun şekilde tasarlanmıştır. Tekerlekleri büyük dişlere sahip şekilde üretilmiştir. Bunun nedeni ise arazide kullanılacağı için kavrama gücünün daha iyi olması isteğindendir. Bu tip araçlarda şasenin üretiminde genellikle alüminyum veya titanyum kullanılmaktadır. Bu malzemeler hafif olmasına rağmen oldukça dayanıklıdırlar.

2.3.2 Yol Tipi RC Model Arabalar

Yol tipi RC model arabalar kullanım amacı olarak asfaltta kullanılmak üzere tasarlanmıştır. Engebeli yüzeylerde kullanılması teknik sorunlara yol açabilmektedir. Arazi tipi RC model arabalara göre farklılıkları mevcuttur. Öncelikle motor seçimi olarak yüksek tork yerine yüksek hızlı motorlara ihtiyaç duyulmaktadır. Aynı şekilde süspansiyon sistemi de yüksek hız yapmaya uygun şekilde tasarlanmıştır. Tekerlek yapısı olarak daha az dişli veya dişsiz tekerlekler kullanılmaktadır.

Projede kullanılan arabalar da bu kategoriye girmektedir. Ancak kullanılan arabaların yol tipi model araçların tüm özelliklerini sağladığı söylenemez. Tekerlek seçimi yaparken dişli tekerlekler seçilmiştir. Bunun nedeni projede araçların hız ve konum bilgilerinin tekerlek üzerinden alınmasıdır. Eğer dişsiz bir tekerlek seçimi yapılmış olsaydı ve tekerleğin sürtünmenin az olmasından dolayı boşa dönmesi yani pati atması gibi bir durum olsaydı alınan konum bilgisinde yanlışlıklar olabilirdi. Örneğin alınan verilere göre tekerleğin 10 kere döndüğü ve buna bağlı olarak da arabanın yaklaşık 2.5 m ilerlediği düşünülmesine rağmen gerçekte boşa dönmeden dolayı araba 2.4 m gitmiş olabilirdi. Bu durum takip algoritmasının özünü oluşturan takip formülünde yanlışlıklar yapılmasını, dolayısıyla bir kaza durumuyla karşı karşıya kalınmasına sebep olabilirdi (WEB_3).

Şekil 2.2'de projede kullanılan RC araca ait bir gösrel sunulmuştur.

Şekil 2.2: Projede Kullanılan RC Arabalardan Bir Tanesine Ait Görsel

2.4 DC Motorlar

Motorlar, elektrik enerjisini mekanik enerjiye dönüştüren makineler olarak düşünülebilir. DC motorlar da isminden de anlaşılabileceği gibi DC voltajları yani düz elektrik enerjisini mekanik enerjiye dönüştürürler. Motorun içinde yer alan sargılara elektrik akımı uygulanmasıyla hareketlerini sağlarlar. DC Motorlar fırçalı ve fırçasız DC motorlar olmak üzere iki ana başlıkta toparlanabilmektedir.

2.4.1 Fırçasız DC Motorlar

İsminden de anlaşılacağı gibi fırçasız DC motorlarda motor sarımlarına temas eden fırçalar bulunmamaktadır. Bu motorların rotor yani motorun dönen kısmında güçlü doğal mıknatıslar kullanılmaktadır. Statorda yani motorun sabit kısmında ise bobinli sargılar mevcuttur.

Fırçasız DC motorları kullanabilmek için harici bir elektronik hız kontrol ünitesine ihtiyaç bulunmaktadır. Elektronik hız kontrol devresi sayesinde DC gerilim, sanal üç fazlı gerilim hâline dönüştürülür ve bu şekilde motor sürülür.

Fırçasız DC motorlar gelişen teknoloji ile birçok alanda kendisine yer bulmaktadır. Bunların başında robot projeleri, insansız hava araçları, fotokopi makineleri, yazıcılar, optik tarayıcılar ve tıp cihazları gelmektedir.

Fırçasız DC motorların birçok avantajı ve dezavantajı mevcuttur. Yüksek verimli olmaları, doğrusal moment - hız ilişkileri, sessiz çalışmaları, fırça ve kollektör olmadığı için daha az bakım masrafına sahip olmaları ve uzun ömürleri avantaj olarak değerlendirilebilir. Ancak buna karşın harici güç elektroniği gerektirmeleri ve maliyetleri dezavantajları olarak öne çıkmaktadır.

2.4.2 Fırçalı DC Motorlar

Piyasada kullanılan motor türlerinden birçoğu fırçalı DC motordur. El matkaplarından hobi sektörüne kadar çok geniş bir kapsamda kullanılmaktadırlar. DC motorların ana milinin üzerinde bobinler yer alır. Mil üzerinde yer alan bobin sarımlarına kömür adı da verilen fırçalar vasıtasıyla elektrik akımı uygulanır. Bu sayede motor mili hareket etmiş olur.

Projede de kullanılan fırçalı DC motorların yaygın olarak kullanılmasının en büyük nedeni nispeten maliyetinin düşük olmasıdır. Fırçasız DC motorlardaki gibi elektronik hız devresi gerekmemesi maliyetin düşmesinin en büyük sebeplerindendir. Buna karşın fırçalı olmasından dolayı zaman içerisinde bakım yapılması gerekmektedir. Bakım masraflarının çokluğu bir dezavantaj olarak gösterilebilir. Aynı zamanda elektriksel bir gürültü oluşturmaları ve fırça nedeniyle oluşan verim kaybı dezavantajları olarak öne çıkmaktadır.

2.4.3 Redüktörlü Fırçalı DC Motor

DC motorlar genel olarak yüksek devirde çalışabilmektedirler. Ancak projenin ihtiyaç duyduğu özelliklere göre bazen yüksek hız yerine yüksek tork daha fazla önem arz etmektedir. Yüksek devir yani yüksek hız yerine yüksek güce ihtiyaç duyulan uygulamalarda motorun miline bağlanan bir dişli seti ile hızı belli ölçüde düşürerek elde edilen tork aynı ölçüde yükseltilebilir.

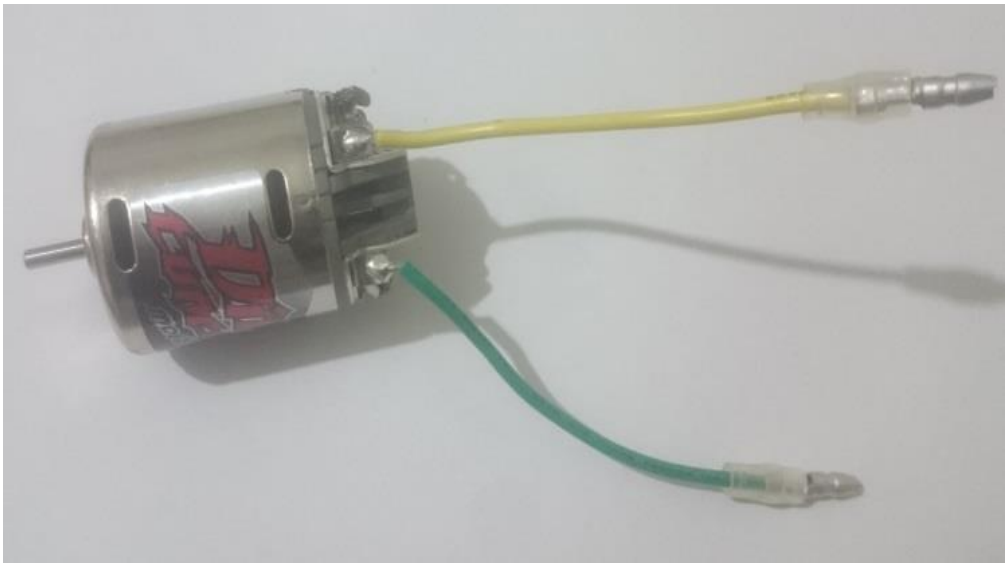
Projede tam da ihtiyaç duyulan motor olarak öne çıkmasına rağmen RC model arabaların redüktörlü motora uygun şekilde tasarlanmamasından dolayı redüktörlü DC motor kullanılamamıştır.

2.4.4 Redüktörsüz Fırçalı DC Motor

Redüktörsüz fırçalı DC motorlar yüksek devire ihtiyaç duyulan ama torkun çok da fazla önemli olmadığı fan, dramel ve yol tipi RC model araba gibi uygulamalarda karşımıza çıkmaktadırlar. Motor mili üzerinde bir dişli sistemi yoktur.

Projede kullanılan araç için 540 model redüktörsüz fırçalı DC motor kullanılmıştır. Burada model isminde yer alan 540 ifadesi motor çapını belirtmektedir. Kullanılan fırçalı DC motorun motor çapı 5.4 cm'dir. Motor seçiminde projenin gerekliliği olan sabit ivmeli harekete iyi tepki verebilecek bir motor kullanılmasına ihtiyaç duyulmuştur. Bunun için de torku yüksek bir motor tercih edilmiştir. Torku yüksek olması nedeniyle hızdan kayıp yaşanması beklenen bir durum olmasına rağmen projenin gerektirdiği maksimum hız olan 2 m/sn hızına rahatça çıkılabildiği için bu durum proje için sorun olarak görülmemiştir (WEB_4).

Şekil 2.3'de projede kullanılan DC motor yer almaktadır.



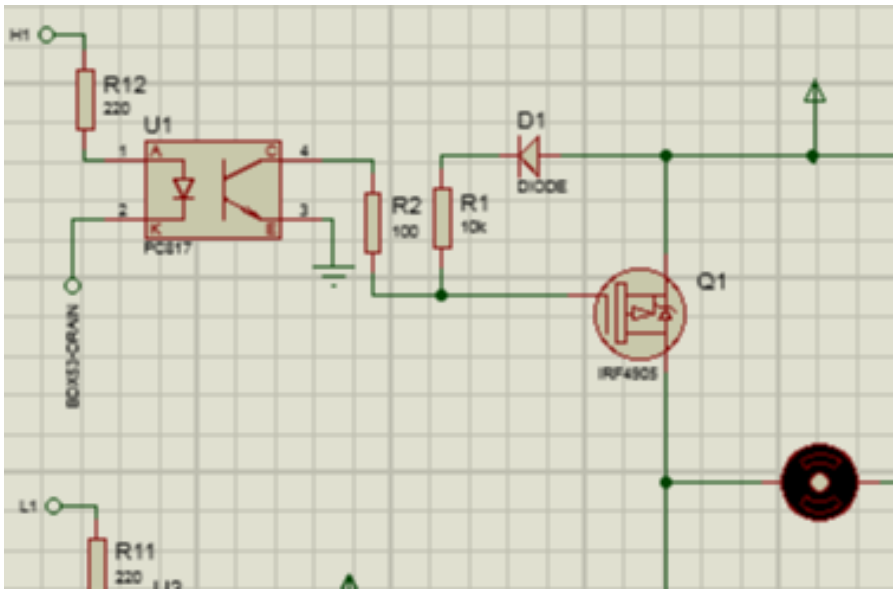
Şekil 2.3: Projede Kullanılan DC Motorlardan Bir Tanesine Ait Görsel

2.5 DC Motor Sürücüsü

DC motor sürücülerine baktığımızda birçok motor sürme tekniği ortaya çıkmaktadır. Projede kullanılan motor sürücüsünde MOSFET sürme tekniği kullanılmıştır.

2.5.1 MOSFET Sürme Teknikleri

MOSFET bir anahtarlama elemanıdır. MOSFET gibi anahtarlama elemanları yüksek frekans altında çalıştırıldıkları durumlarda anahtarlama işlemleri oldukça hızlı şekilde yapılabilmektedir. Kullanılan MOSFET elemanının anahtarlama hızını arttırmak ve MOSFET elemanının anahtarlanması için uygulanan PWM sinyalinin MOSFETin *gate* ucuna daha düzgün bir şekilde gelmesi için bazı teknikler de kullanılmaktadır.



Şekil 2.4: MOSFET Elemanının Anahtarlamasında Kullanılan Malzemelerin İSİS Görüntüsü

MOSFETin anahtarlama hızını arttırmak için MOSFETin GS uçları arasına MOSFET elemanını saturasyona sokacak bir gerilim uygulanmalıdır. Bunun yanında yeteri kadar hızlı deşarj edebilmek için entegre ile MOSFET arasındaki direnç değerinin düşürülmesine ihtiyaç vardır. Şekil 2.4'de de görüldüğü üzere bu direnç değeri 100 Ω olarak belirlenmiştir.

MOSFET anahtarlanırken bir kapasitörün şarj ve deşarj işlemleri yapılıyormuş gibi düşünülebilir. MOSFETin gate ucuna bağlı direnç, anahtarlama hızını belirleyen faktörlerden birisi olduğu için *gate* direncinin büyük olması hiç istenmeyen bir durum olan anahtarlama kayıplarını artırır ve hatta osilasyona sebep olabilir. MOSFET sürme teknikleri de kendi içerisinde iki ana grupta incelenebilir.

2.5.2 Bipolar Totem Pole Sürme Tekniği

Bu yöntem en çok kullanılan sürme tekniklerinden birisi olarak göze çarpmaktadır. Bu yöntemde çıkış ucundaki sinyal olduğu gibi *gate* ucuna aktarılmaktadır. Gürültüyü azaltmak için dirençler ve ayrıca *bypass* kapasitörleri de bu sürme tekniğinde kullanılan diğer devre elemanları olarak göze çarpmaktadır. Bu teknik ani akım sıçramalarına karşı dayanıklı bir sürme tekniğidir. Ani akım sıçramalarının dolaştığı döngü nispeten daha küçüktür ve bu nedenle parazitik endüktans da daha küçük olmaktadır.

Bu tekniğin avantajlarından birisi de iki bipolar MOSFETin ters gerilim ve ters akımlardan dolayı birbirlerini korumasıdır. Ayrıca bu yöntem entegrenin yeterli *gate* akımı sağlayamadığı durumlarda da kullanılabilmektedir. Ancak yüksek frekanslı devrelerde bu yöntem önerilmemektedir.

2.5.3 PWM Direkt Sürme Tekniği

PWM direct sürme tekniği de *bipolar totem pole* sürme tekniği gibi oldukça popüler bir sürme tekniğidir. Bu sürme tekniğinde GND ucu ile MOSFETin *source* bacağı arasındaki mesafe parazitik endüktans değerinin belirlenmesinde önemli rol oynamaktadır. Parazitik endüktans anahtarlama hızının düşmesine neden olduğu için çok istenen bir durum değildir. Kullanılan PCB yolları genişletilerek parazitik endüktans değeri düşürülebilmektedir. Bu sürme tekniğinde PWM entegresinin verebileceği akım sınırının düşük olması bir dezavantaj olarak göze çarpmaktadır. Eğer entegrenin karşılayabileceği akım sınırı *gate* kapasitansının ihtiyacı olan akımı karşılayamayacak düzeydeyse entegrenin harcayacağı güç artacaktır ve bu durum entegrenin ısınmasına yol açacaktır. *Gate* akımındaki ani yükselmeler akım

sıçramalarına yol açabileceği için entegre üzerinde hasara da yol açabilirler. Bu durumun önüne geçebilmek için pozitif ve negatif gerilimler arasına bir adet *bypass* kapasitörü atılmaktadır. *Gate* akımındaki ani yükselmelerin karşılandığı bu kapasitör sayesinde entegre üzerine ekstra oluşacak güçler engellenmekte ve entegrenin sağlıklı çalışması sağlanmaktadır (WEB_4).

2.6 Servo Motorlar

Servo motorlar, gönderilen kodlanmış sinyaller ile şaftları özel bir açısal pozisyonda döndürebilen motorlardır. En yaygın kullanım alanı olarak robotik sektörü gösterilebilir. Aynı şekilde RC uygulamalarında da sıkça karşımıza çıkmaktadırlar. Servolar, istenilen pozisyonu alması ve yeni bir komut gelmediği sürece bulunduğu pozisyonu değiştirmemesi amacıyla üretilmişlerdir.

Servo motorların iç kısmında motorun hareketini sağlaması amacıyla bir DC motor yer almaktadır. Servo motor içyapısında, DC motorla birlikte bir dişli mekanizması, bir motor sürücü devresi ve bir de potansiyometre yer almaktadır. Bu malzemelerden potansiyometre motor milinin dönüş miktarını ölçmektedir. Servo motor içyapısında bulunan DC motor döndükçe potansiyometre de döner ve böylece motor kontrol devresi motorun bulunduğu pozisyonu öğrenmiş olmaktadır. Motor sürücü devresi öğrendiği bu konum bilgisi ile ulaşılmak istenilen pozisyonu karşılaştırarak motoru sürme işlemini yapar.

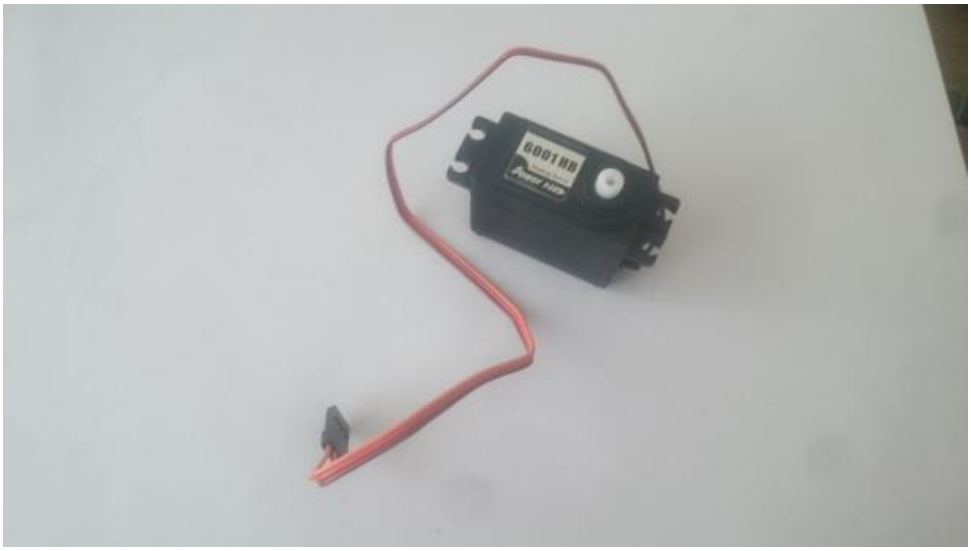
Servo motorlar içyapısında bulunan motor sürücü devresi sayesinde sürüldükleri için diğer motorlar gibi ekstra bir motor sürücü devresine ihtiyaç duymadan çalışabilmektedirler. Servo motorların birçoğu 4.8V ile 6V aralığındaki gerilim değerlerinde çalışmaktadırlar. Ancak 6V'dan daha yüksek gerilim değerlerinde çalışan servo motorlar da piyasada bulunmaktadırlar.

Servo motorların çalışabilmesi için sinyal genişlik modilasyonuna ya da daha bilinen adıyla PWM sinyaline ihtiyaç vardır. Bir kontrol ünitesi yardımıyla üretilen bu PWM sinyalleri ile servo motor pozisyonu ayarlanır. Servo motorlar her 20 ms içerisinde bir pals değerini okuyabilmektedirler. Okudukları her bir pals değerinin uzunluğu motorun dönüşünü belirler. Örneğin 1 ms'lik bir pals okuduğunda sıfır

derece, 1.5 ms'lik bir pals okuduğunda doksan derece ve 2 ms'lik bir pals değeri okuduğunda 180 derece pozisyonunu almaktadırlar. Servo motorlar hareket etmeleri için bir komut okuduklarında öncelikle istenilen pozisyona hareketi gerçekleştirirler. Sonrasında ise o pozisyonda kalmaya çalışırlar. Tabi ki bulundukları konumu sürekli koruyabilmeleri için aldıkları pals değerinin tekrarlanmasına ihtiyaç duyarlar.

Projede servo motor RC model arabada sağa sola dönüş mekanizması için kullanılmıştır. Servo motor seçimi yaparken 4.8V ile 6V arasındaki değerlerde çalışmasına dikkat edilmiştir. Aracın yolda giderken tam istenilen doğrultuda gidebilmesi için servo motorun seçiminde yüksek torklu bir servo motor seçimi yapılmıştır. Böylelikle istenilen pozisyona gelmesi ve karşı gelen kuvvetlere karşı konumunu koruyabilmesi daha kolay olabilecektir (WEB_5).

Projede şekil 2.5'de yer alan servo motor aracın yön tayinini sağlaması için kullanılmıştır.



Şekil 2.5: Projede Kullanılan Servo Motorlardan Bir Tanesine Ait Görsel

2.7 Lipo Piller

Lipo piller kısaca özetlemek gerekirse yapısında hem lityum hem de polimer kimyasallarını barındıran pillerdir. İsimlerini de lityum ve polimer isimlerinin kısaltmalarından alırlar. Modelcilik sektöründe sıvı yakıtlardan elektriğe geçilmesinde en büyük paylardan birisi bu pillere aittir. Çünkü modelcilikte

kullanılacak olan bir elektrik motoru çok yüksek düzeylerde akım çekebilmektedirler. Eski tip piller ile ulaşılabilecek anlık ve sürekli akım değerleri çok kısıtlı olduğu için eski tip bir pil ile modelcilik sektörünün elektrikli motorlara geçmesi ne yazık ki mümkün değildi. Lipo piller hem sürekli olarak verebildiği akımın yüksek olması hem de anlık olarak sürekli verdiği akımın kat kat üzerine çıkabilmesi bakımından bu soruna çözüm olmuşlardır.

Lipo piller hücrelerden oluşan pillerdir. Lipo pilde bulunan her bir hücrenin voltaj değeri 3.7V'tur. Bu hücreler seri bağlanarak Lipo Pilin Voltaj değeri değiştirilir. Örneğin 2 hücresi seri bağlı piller yani 2S piller 3.7 x 2 = 7.4V değerine sahiptirler. Lipo piller üretilirken genellikle sadece seri bağlı hücrelerden oluşan piller üretilirler. Ancak bazı pillerde paralel bağlı hücreler de yer almaktadır. Hücrelerin paralel bağlanması durumunda voltaj değeri değişmemektedir fakat pilin akım kapasitesi yükseltilebilir. Örneğin 3S2P bir pil tanımı kullanılmışsa 3.7 x 3 = 11.2V değerine sahip bir lipo pil ve paralel 2 hücre olduğu için pilin akım değeri iki kat olarak düşünülebilir.

Lipo pilin her bir hücresinin boş hâldeki voltajı 3V, tam dolu hâldeki voltajı ise 4.2V'tur. Lipo pilin herhangi bir hücresinin 3V altına düşmesi ile pilin kullanılamaz hâle gelmesi sağlanmış olabilir. O yüzden sürekli olarak hücrelerdeki voltaj değerleri kontrol edilmeli ve belli bir değerin altına düştüğünde ise şarj işlemi gerçekleştirilmelidir. Hücrelerdeki bu voltaj değerlerini sürekli olarak ölçen ve belli değerlerin altına düştüğünde üzerinde bulunan buzzer ile uyarı yapan devreler bulunmaktadır. Herhangi bir sıkıntı yaşamamak adına böyle bir devre kullanılması mantıklı bir seçenek olarak ortaya çıkmaktadır. Şekil 2.6'de projede kullanılan buzzer ve 2S yani 7.4V voltaj değerine sahip lipo pil yer almaktadır. Kullanılan pil 4000mAh değerine sahiptir ve anlık olarak 100A akım değerine kadar çıkabilmektedir.

Şekil 2.6: Projede Kullanılan Lipo Pil ve Lipo Buzzer

Lipo pillerin birçok avantajından ve dezavantajından bahsedilebilir. Lipo pillerin sağlayabildikleri akım değerleri çok yüksek değerlere çıkabilmektedir. Bu da büyük bir avantaj olarak göze çarpmaktadır. Diğer pillere oranla hafif olmaları da önemli bir özelliktir. Hobi sektöründe ve özellikle insansız hava araçlarında sıklıkla diğer pillerin yerine hafif oldukları için lipo piller tercih edilmektedirler. Kullanım süreleri olarak baktığımızda da yine birçok pile göre daha uzun kullanım süreleri olduğundan bahsedebiliriz. Yine göze çarpan başka bir özelliği ise istenen şekilde ve ölçüde imal edilebilme imkânına sahip olmasıdır. Örneğin artık telefon veya laptop pili olarak dahi kullanılmaya başlanmıştır.

Tüm bu avantajlarının yanında bazı dezavantaj sayılabilecek özellikleri de mevcuttur. En büyük dezavantajlarından birisi özel şarj aletine ihtiyaç duymalarıdır. Lipo piller için özel olarak üretilmiş şarj aletleri balans yapabilme olanağı sağlar. Böylelikle her bir hücrenin eşit voltajlarda doldurulması sağlanır. Lipo piller ile ilgili birçok patlama haberi de kullanıp kullanmama konusunda tereddüt yaşanmasına sebep olmaktadır. Lipo piller kullanılırken ve şarj edilirken çok dikkatli olunmalıdır. Yine pillerin kullanılmadığı zamanlarda saklanma koşullarına da çok önemlidir. Piller çok sıcak olmayan bir ortamda, su ve nemden uzak karanlık bir yerde saklanmalıdır. Lipo piller kesinlikle ve kesinlikle kısa devre yapılmamalıdır. Herhangi bir kısa devre durumunda pil üzerinden çok büyük akımlar çekileceği için pilin bozulması gibi durumlar ortaya çıkabileceği gibi patlama, yanma gibi daha kötü sonuçları da olabilir.

Lipo piller yanlış şarj veya çok yüksek akımlarda kullanıldıklarında şişme olayları görülebilir. Lipo pil üzerindeki koruyucu kısım yırtılmadığı sürece bu şişme olayları çok da önemli değildir. Fakat şişmedeki boyutlar dış korumaya zarar verecek düzeye gelmişse artık o lipo pilin ömrü tükenmiş demektir. Kullanılmaması uygun ve yerinde bir karar olacaktır.

Lipo pillerin içinin açılması bazı olumsuz durumların yaşanmasına neden olabilir. İçi açılan lipo piller hava veya su ile temas ettiğinde patlama ve yanma gibi sorunlara neden olabilir. Bu yüzden kesinlikle lipo pillerin koruyucu kısımları yırtılarak içi açılmaya çalışılmamalıdır (WEB_6).

2.8 Mikrodenetleyiciler ve STM Ailesi

Mikroişlemciler ya da mikrodenetleyiciler en temel anlamda transistörlerden oluşurlar ve sadece 1 ve 0 değerleriyle çalışmaktadırlar. Mikrodenetleyiciler isminden de anlaşılacağı üzere çok küçük olarak üretilmişlerdir. Mikrodenetleyicilerin nasıl çalıştıklarını anlayabilmek için öncelikle transistör kavramı üzerinde durmamız gerekmektedir. Sonrasında ikili sayı sistemleri hakkında da bilgiye sahip olmamız gerekmektedir.

20. yüzyılın en önemli buluşlarından birisi olarak gösterilen transistör günümüzde hemen hemen bütün elektronik cihazların temelini oluşturmaktadır. 1947 yılında geliştirilmesinin ardından elektronik biliminin çok pozitif yönde etkilendiği ve çok büyük ilerleme kaydettiği savunulabilir.

Transistörler en temel anlamda bir gerilim veya akım kaynağı ile başka bir gerilim veya akım kaynağını kontrol etmeye imkân sağlayan elektronik devre elemanları olarak düşünülebilir. Anahtarlama yapma, yükselteç ya da üreteç gibi de kullanabilmektedir. En çok kullanılan türleri BJT veya FETlerdir. Transistörlerin üç bağlantısı vardır. Bunlar BJT transistörlerde *base, emitter* ve *collector* olarak adlandırılırken FET transistörlerde *gate, drain* ve *source* olarak adlandırılırlar. Transistörlerin genel çalışma mantığı iki bacağı arasındaki gerilim veya akımı kesmesi veya iletmesi şeklinde tanımlanabilir. Örneğin bir BJT için ele alırsak *collector* ucuna bağlı 5V voltaj değerine sahip enerji base ucundan aldığı bilgiye

göre *emitter* ucuna iletilebilir veya iletimi kesilebilir. Transistörün çalışma mantığı kabaca böyle olsa da transistör içerisinde gerçekleşen olaylar daha karmaşık olmaktadır.

Bütün transistörler elektrik direncinin değişmesine dayalı olarak çalışırlar. *Base* akımı ya da *gate* gerilimi olmadığı sürece *collector* ve *emitter* arasındaki direnç çok yüksek değerlerde olduğu için bu iki bağlantı arasında bir akımdan bahsedilemez. Ancak *base* bağlantısından çok küçük bir akım dâhi gelse *collector* ile *emitter* arasındaki dirençte çok büyük azalma meydana geleceğinden akım geçişine izin verilmiş olur. Böylece transistör ile çok küçük bir akım kullanılarak çok büyük akımlar denetlenebilir.

Transistörlerin birleşmesiyle bir bitlik veriyi tutmaya yarayan flip flop yapıları meydana gelir. Flip flop yapıları basit bir şekilde bir sonraki bit gelene kadar çıkışındaki biti saklayan, girişte tekrar veri geldiğinde *clock* darbesiyle birlikte çıkıştaki değeri de değiştiren lojik devre olarak tanımlayabiliriz. Kısaca flip flop yapısının yaptığı şey bir biti saklamaktır.

Flip flopların da birleşmesi ile daha fazla verinin saklanmasını sağlayan register ya da Türkçe adıyla saklayıcılar oluşmaktadırlar. Bir saklayıcı mikroişlemci veya mikrodenetleyicinin yapısına göre 8, 16 ya da 32 bit olarak karşımıza çıkabilir. Saklayıcılar da birleşerek işlemcinin hafıza birimlerini oluştururlar.

Mikroişlemci veya mikrodenetleyicilerde hafıza birimleri dışında bu birimlerdeki veriyi işleyen, okuyan ve yazan farklı birimler de mevcuttur. Aslına bakarsak sistem içerisinde yer alan tüm bileşenler, mantıksal olarak saklayıcılardan meydana gelmektedirler. Mikroişlemci veya mikrodenetleyiciler içerisinde yapılan her işlem en basit mantıkta saklayıcılarda tutulan bilgilerin mantıksal devre yapıları aracılığıyla birbirleri arasında taşınması ve belirli işlemlere tabi tutulması ile gerçekleşmektedirler. Bu mantıksal devre yapılarından bazılarını saymak gerekirse ALU matematiksel işlemleri gerçekleştirirken giriş çıkış birimleri ise çıkış işlemlerinde mikroişlemciden gelen veriyi tutarken giriş işlemlerinde ise dış dünyadan gelen bilgileri tutma amacıyla kullanılırlar.

Mikroişlemci ve mikrodenetleyiciler genellikle birbiri ile karıştırılan terimlerdir. Aralarındaki farka bakılacak olursa mikroişlemciler sadece işlem ve hafıza birimlerinden oluşmaktadırlar. Mikrodenetleyiciler ise hafıza, analog-dijital çeviriciler ve zamanlayıcı gibi diğer çevre birimlerini bünyesinde bulundururlar. Mikroişlemciler günümüzde genel olarak kişisel bilgisayarlarda kullanılmaktadırlar.

Mikrodenetleyicileri genel manada endüstriye yönelik olarak kontrol ve otomasyon işlemlerini gerçekleştirmek için tasarlanmış özel mikroişlemciler olarak tanımlayabiliriz. İçyapılarına baktığımızda mikroişlemcilere göre daha fazla birime sahip olmalarına göre hızları mikroişlemcilere kıyasla daha düşük seviyelerdedir.

Bir gömülü sistem tasarımı yapıyorsak mikrodenetleyiciler işimizi çok daha kolaylaştıracaktır. Çünkü gömülü sistem için mikroişlemci kullanmamız durumunda ram, osilatör, hafıza elemanları veya analog dijital dönüştürücü birimlerini ayrı olarak satın alıp bu birimler ve mikroişlemci arasında veri yolu, adres yolu bağlantılarını kurmamız gerekecekti. Mikrodenetleyiciler ise tim bu birimlerin birleştirilmiş hâli olarak karşımıza çıkmaktadır.

Günümüzde birçok mikrodenetleyici üreticisi çeşitli modellerde üretim yapmaktadırlar. Ülkemizde en çok kullanılan bazı mikrodenetleyicilere göz gezdirmek gerekirse 8051, Microchip, Atmel, Texas Instruments ve ARM tabanlı TI, ST ve Atmel işlemciler karşımıza çıkmaktadır. Ülkemizde 8051, Microchip firmasının ürettiği PIC ve Atmel firmasının AVR mikrodenetleyicileri uzun zamandır popülerliğini korumaktadırlar. Ancak son zamanlarda Texas Instruments firmasının MSP430 ailesi ve ARM tabanlı olarak karşımıza çıkan TI, ST ve Atmel mikrodenetleyiciler giderek popülerliğini artırmaktadır.

2.8.1 ARM Tabanlı Mikrodenetleyiciler

ARM mimarisi 1983 yılında Acorn Computers Ltd. tarafından ARM1 adı ile geliştirilmeye başlanmıştır. 2 yıl süre sonunda 1985 yılında ilk kez piyasaya sürülmüştür. Bir yıl sonrasında ise aynı firma tarafından 32 bitlik ARM2 modeli piyasaya çıkmıştır. 1990 yılında firma adını Advanced RISC Machine Ltd. olarak değiştirmiştir ve 1998 yılına kadar bu isim ile ticari faaliyetlerine devam etmiştir.

Son olarak 1998 yılında ise günümüzdeki sahip olduğu isim olan ARM Ltd. ismini almıştır. ARM firması 32 bitlik işlemci çekirdekleri üretip, bu ürettiği çekirdeklerin mimarilerini Philips, Samsung, Atmel, Intel gibi firmalara lisanslı olarak satmaktadır.

ARM işlemciler 32 bitlik yapısı sayesinde 8 bitlik işlemcilere göre çok daha hızlıdır. ARM işlemci yapısı düşük güç tüketimi, yüksek performans gibi özelliklerinden dolayı sektörde açık ara lider konumdadır. Sektörde açık ara lider olmasındaki en önemli etken olarak düşük güç tüketimleri nedeniyle cep telefonu, PDA ve taşınabilir cihazlarda tercih edilmesi gösterilebilir. Hem düşük güç tüketimi hem de yüksek performansları nedeniyle günümüzde yaklaşık %75'lik oranla gömülü sistemler üzerinde en çok kullanılan işlemcilerdendir. ARM mimari ailesi 3 temel gruba ayrılarak incelenebilirler.

Klasik ARM işlemcileri grubunda yer alan işlemcilere baktığımızda Cortex serisi öncesindeki işlemciler olarak gösterilebilirler. Bu grubun üyeleri ARM 7, ARM 9 ve ARM 11 işlemcilerdir. ARM 7 serisi işlemciler daha çok motor kontrolü, sinyal işleme gibi mikrodenetleyici uygulamalarında kullanılan mikrodenetleyicilerin çekirdeğini oluşturmaktadırlar. ARM 9 ve ARM 11 serileri uygulama seviyesinde ve daha çok mobil cihazlarda kullanılmaktadır. Klasik ARM çekirdekleri görece daha eski ARM ürünlerindendir ve kullanımları giderek azalmaktadır.

ARM Cortex Embedded işlemciler grubunda Cortex M ve Cortex R serileri yer almaktadır. Bu seriler deterministik bir şekilde çalışması gereken gerçek zamanlı ve düşük güç tüketimi gerektiren uygulamalarda kullanılmaktadır. NXP Semiconductors, Texas Instruments, STMicroelectronics ve Toshiba gibi mikrodenetleyici üreticileri bu çekirdeğe sahip işlemcileri kullanmaktadırlar.

ARM Cortex uygulama işlemcileri grubunda ise son dönemlerde hayatımızda oldukça büyük bir yer bulan akıllı telefonlar, tablet bilgisayarlar gibi ürünlerin birçoğunda bu serideki çekirdeklere sahip işlemciler bulunmaktadır. Örnek olarak bakacak olursak iPhone, iPad, Samsung Galaxy Tablet, RIM Playbook gibi ürünlerde Cortex-A serisi ARM tabanlı mikroişlemciler bulunmaktadır. Bu seri daha çok yüksek performans gerektiren son kullanıcı uygulamalarının üzerinde çalışacağı platformlarda kullanılmaktadırlar.

Piyasada kullanılan birçok ARM tabanlı mikrodenetleyici bulunmaktadır. Projede kullanılan mikrodenetleyici STM32F1 serisi içerisinde yer alan STM32F103C8T6 model mikrodenetleyicidir.

ST firmasının STM32F1 serisi ana mikrodenetleyiciler, tıbbi ve tüketici pazarlarındaki çok çeşitli uygulamalarda kullanılmaktadırlar. STM32F1 serisi ile ST firması, ARM Cortex M mikro denetleyicileri dünyasına öncülük etmektedir. Gömülü uygulamaların tarihine baktığımızda STM32F1 serisi çok önemli bir noktadadır. Birinci sınıf çevre birimlerine sahiptir ve düşük güç tüketimi, düşük voltajlı çalışma ve yüksek performansa sahiptir. Basit bir mimarisinin ve kolay kullanımının yanı sıra düşük fiyatları sayesinde çok geniş bir kitleye ulaşmaktadır. STM32F1 serisi mikrodenetleyicileri STM32F100, STM32F101, STM32F202, STM32F103 ve STM32F105 şeklinde 5 ana gruba sahiptirler.

STM32F103 serisi mikrodenetleyiciler maksimum 72 MHz'e kadar CPU hızı ile Cortex M3 çekirdeği kullanmaktadır. Motor kontrol çevre birimlerinin yanı sıra USB tam hızlı arayüz ve CAN ile 16 KB'tan 1 MB flasha kadar geniş bir alanı kapsamaktadır. Projede kullanılan geliştirme kartları da bu seride yer alan STM32F103C8T6 mikrodenetleyiciye sahip aynı isimdeki geliştirme kartıdır.

Projede kullanılan geliştirme kartına baktığımızda 72 MHz'lik ARM Cortex M3 çekirdeğe sahiptir ve üzerinde 20 KB ram bulunmaktadır. 64 KB ile 128 KB flasha sahip olan geliştirme kartında 32 kHz kristal bulunmaktadır (WEB_7).

2.9 Renk Sensörleri

Projenin 1:10 boyutlu araçlardan oluşacağından ve kendine ait özel yollarda ilerleyeceğinden bahsedilmişti. Projede 30 cm enindeki ve 1 m boyundaki sunta malzemesinden parçalar birleştirilerek yol olarak kullanılmıştır. Araçların bu yolda düz şekilde ilerleyebilmeleri için yol üzerine 2 cm eninde siyah şerit çizilmiştir.

Şekil 2.7: Projede Kullanılan Yola Ait Görsel

Şekil 2.7'da projede kullanılan yol görülmektedir. Şekil 2.6'daki yol üzerinde görülen şeritte aracın düz bir şekilde ilerleyebilmesi için aracın ön kısmında QTR-8RC modelinde çizgi sensörü kullanılmıştır. 75x13x3 mm boyutlarına sahip olan bu sensör 3.3V ile 5V arası gerilimlerde çalışabilmektedir ve yaklaşık 100mA akım çekmektedir. En iyi algılama için zemin üzerinden yaklaşık 3 mm yükseklikte olması gerekmektedir. Ancak yaklaşık 10 mm'ye kadar mesafelerde de algılama yapabilmektedir.

Üzerinde 8 adet IR alıcı-verici çifti bulunan QTR-8RC 11 pinli bir sensördür. 8 tanesi alıcı-verici çiftlerden gelen bilgilerin alındığı giriş-çıkış pinleri, iki tanesi besleme pinleri ve bir tanesi ledleri kullanılmadığı zamanlarda kapatmaya yarayan *led on* pinidir. Projenin her aşamasında sensörlerin kullanılması gerektiği için *led on* pini kullanılmamıştır.

Bu sensörlerin genel yapısına baktığımızda temel olarak kızılötesi gönderilen bilginin yerden ne kadar sürede yansıdığının hesaplanarak zemin rengini belirleyebildikleri görülmektedir. Projede kullanılan sensör dijital bir sensör olduğu için sadece siyah-beyaz gibi iki renk ayrımı yapabilmektedir.

Projede renk sensörleri sadece yol takibi için kullanılmamışlardır. Aynı zamanda bu sensörler tekerlekten konum bilgisini almak için de bir adet kızılötesi sensör kullanılmıştır. Tekerlek üzerine yerleştirilen siyah ve beyaz zemin farklarını algılaması için kullanılan sensör ile aracın ne kadar yol aldığı konusunda bilgi sahibi olunmuştur. Tekerlek üzerinde yer alan 4 adet siyah zemin ile birlikte aracın yaklaşık her 2.5 cm yol alışında bilgi elde edilmiştir (WEB_8).

Şekil 2.8’de tekerlek üzerine yerleştirilen siyah – beyaz şeritler ve bu şeritlerden bilgi alan renk sensörü görülmektedir.

Şekil 2.8: Tekerleğe Yerleştirilen Siyah – Beyaz Şeritler ve Renk Sensörüne Ait Görsel

2.10 SPI Haberleşme Protokolü ve Wireless Haberleşme Modülü

SPI Motorola firması tarafından geliştirilen çift taraflı ve senkron bir seri haberleşme standardıdır. Günümüzde pek çok tümdevre tarafından donanımsal olarak desteklenmektedirler. SPI haberleşme protokolünde veri transferi yapılırken *master* - *slave* ilişkisi yani efendi - köle mantığı vardır. Master cihaz veri haberleşmesini başlatan cihazdır. Master tarafından veri iletiminin başlaması ile birlikte veri her iki yönde de eşzamanlı olarak aktarılabilir.

SPI haberleşmede 4 adet sinyal hattı mevcuttur. Bunlar sırasıyla SCK, SDI, SDO ve CS sinyallerini taşımaktadırlar. SCK hattındaki saat sinyali *master* cihaz tarafından üretilir ve master cihaz tarafından üretilen bu sinyal tüm diğer *slave* cihazlara giriş verisi olarak uygulanır. *Master* cihaz hangi *slave* cihazın CS ucunu aktif hâle getirmiş ise ilgili *slave* cihaz haberleşme için seçilmiş olur ve *master* - *slave* haberleşmesi başlamaktadır.

SPI cihazlarda bulunan MISO ve MOSI hatlarından da anlaşılacağı üzere SPI haberleşmede veri hatları tek yönlü olarak kullanılmaktadırlar. Diğer haberleşme

protokollerindeki gibi *slave* cihazların adreslerinin olmasına da gerek yoktur. Her çevresel cihazın yani *slave* cihazın seçim ayağı vardır. Bu ayağa *slave select* teriminin kısaltılmış hâli olan SS hattı denir. Bu hattın sayısı kullanılan çevresel cihazların sayısı ile aynıdır. Her *slave* cihaz için *master* cihazdan ayrı SS hatları çıkmaktadır. SS hattı LOW olan yani 0 V gerilime sahip olan çevresel cihaz *master* cihaz ile iletişim hâline geçer. *Master* cihazdan *slave* cihaz sayısı kadar SS çıkışı sağlanmıştır. *Master* cihaz iletişime geçmek istediği çevresel cihazın SS pinini LOW düzeyine çeker.

SPI haberleşme protokolünde pratikte baktığımızda dört adet farklı çalışma modu bulunmaktadır.

- **Mode 0:** Haberleşme henüz başlamamışken veya haberleşme yokken saat sinyali LOW düzeyindedir. Veriler saat darbesinin düşen kenarında değiştirilirken yükselen kenarında okuma işlemi yapılmaktadır.
- **Mode 1:** Haberleşme başlamadan önce ve haberleşme yok iken saat sinyali LOW düzeyindedir. Veriler saat darbesinin yükselen kenarında yazılırken düşen kısmında okuma işlemi yapılmaktadır.
- **Mode 2:** Haberleşme başlamadan önce ve haberleşmenin olmadığı zamanlarda saat sinyali HIGH düzeyinde sinyaller göndermektedir. Mode 2 çalışma modunda saat darbesinin yükselen kenarında yazma işlemi yapılırken düşen kenarında okuma yapılmaktadır.
- **Mode 3:** Haberleşme başlamadan önce ve haberleşme yok ise saat sinyali HIGH seviyededir. Veriler saat darbesinin düşen kenarında yazılırken düşen kenar durumunda okuma işlemi gerçekleşmektedir.

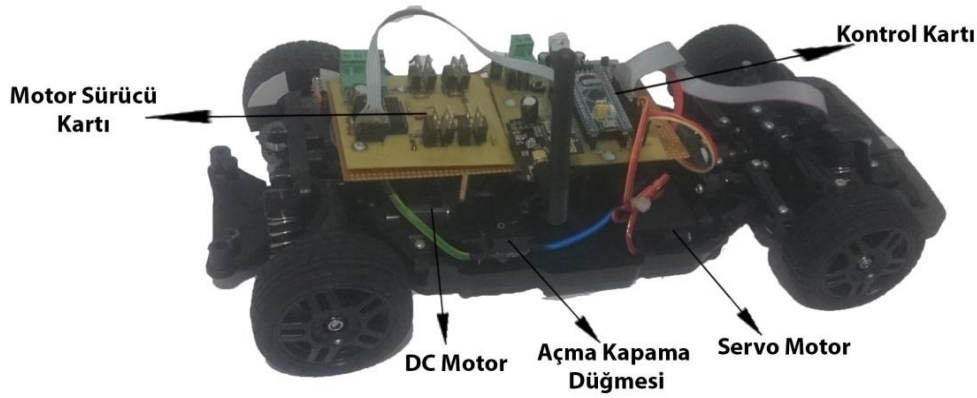
Proje gerçekleştirirken SPI haberleşme protokolünün kullanabileceği bir haberleşme modülü kullanılması uygun görülmüştür ve yapılan araştırmalar sonucunda hem fiyat konusunda hem de kullanım konusunda oldukça yeterli olarak ortaya çıkan nrf24l01 modülünün kullanılmasına karar verilmiştir. Nrf24l01 kablosuz haberleşme modülü Nordic firması tarafından geliştirilmiştir. 2.4 GHz frekansında haberleşme yapılmasına imkân sağlamaktadır. Çok düşük güç tüketimine sahiptir ve 2 MBps hızında haberleşme yapılmasına imkân sağlamaktadır.

Nrf24l01 haberleşme modülü çift yönlü olarak haberleşme yapılmasına olanak sağlamaktadır. Haberleşme yönünün değiştirilmesi durumunda kablo bağlantılarının değiştirilmesine gerek olmaması projede büyük kolaylık sağlamıştır (WEB_9).

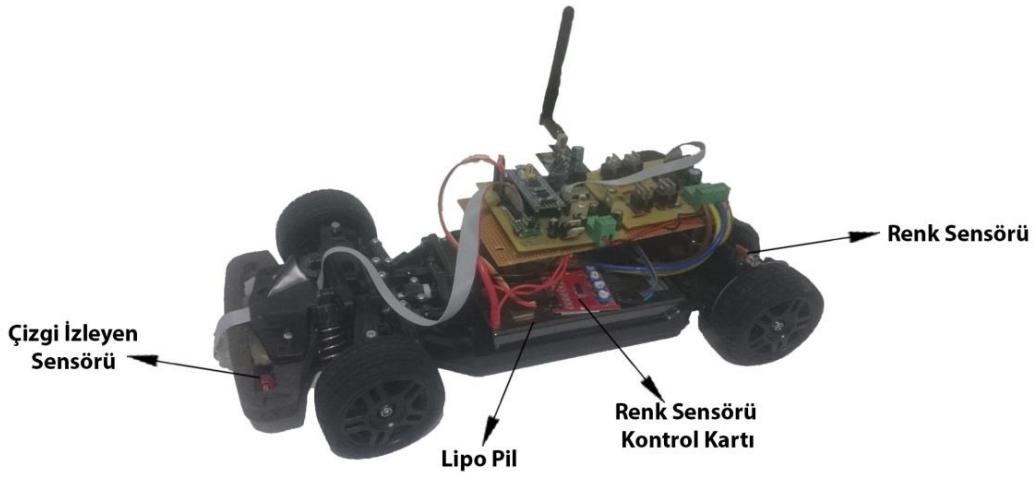
3. DONANIM

3.1 Giriş

Bu bölüm boyunca aracın hareketi için gerekli olan tüm devreler ve ekipmanlarla ilgili bilgiler verilecektir. Şekil 3.9 ve şekil 3.10'da aracın sağ ve sol tarafından çekilen resimleri ve devre kartları ile devre malzemelerinin araç üzerindeki yerleri gösterilmektedir.



Şekil 3.9: Aracın Sağ Tarafından Bakıldığında Devre Kart ve Elemanlarının Yerlerini Belirten Görsel



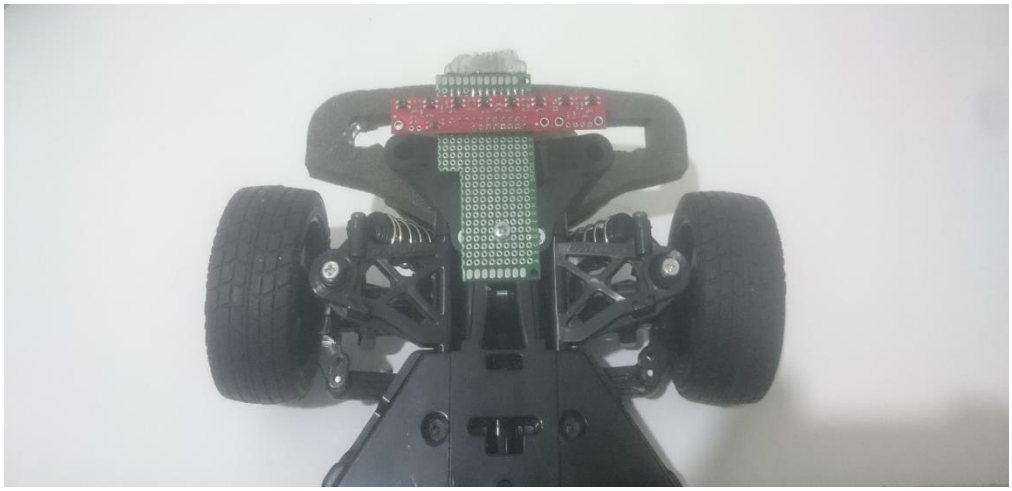
Şekil 3.10: Aracın Sol Tarafından Bakıldığında Devre Elemanlarının Yerlerini Belirten Görsel

Şekil 3.9 ve 3.10 ile motor sürücü kartının, kontrol kartının, DC ve servo motorun, açma kapama düğmesinin, lipo pilin, çizgi izleyen sensör ile renk sensörünün konumları gösterilmiştir. Bu bölüm boyunca bu elemanlardan, devre kartlarından ve araç üzerine yerleştirilmelerinden bahsedilecektir.

3.2 Çizgi İzleyen Sensörün Montajı

Projede QTR-8RC model çizgi izleyen sensörü kullanıldığından daha önce bahsetmiştik. Bu sensör üzerinde 8 adet renk algılayıcılar yer almaktadır. Projede kullanılan sensör dijital bir sensör olduğu için başta kalibre esnasında tanıtılan iki rengi algılayarak bir renkte *high* diğer renkte ise *low* gerilim vermektedir. Tabi bu iki rengin siyah ve beyaza yakın renkler seçilmesi önem arz etmektedir.

Renk algılama için kullanılan bu sekiz algılayıcı birbirine eşit mesafede olacak şekilde sensör üzerine yerleştirilmişlerdir. Aracın çizgi izleyerek ilerlemesinden dolayı çizgi sensöründen alınan verilerin doğruluğu oldukça önemlidir. Sensörden verilerin doğru alınabilmesi için de bu sensörün tam düz şekilde ve yerden belli bir yükseklikte konumlandırılmış olması gerekmektedir.



Şekil 3.11: Araç Üzerinde Çizgi Sensörünün Yerini Gösteren Görsel

Şekil 3.11'de görüldüğü gibi aracın altında yer alan bir vida sökülmüş ve araya çizgi sensörü için yapılmış olan devre koyularak daha uzun bir vida ile çizgi sensörünün montajı yapılmıştır. Araya pul koyularak sensörün daha aşağıda olması veya sensörün uç tarafından yukarı doğru kablo bağı ile çekilmesi gibi işlemlerle de

deneme yapılmasına rağmen şu andaki hâliyle en optimum sonuçların elde edildiği görülmüştür.

3.3 Renk Sensörü Montajı

Proje için önem arz eden konulardan birisi de aracın aldığı yolun ve buna bağlı olarak da aracın anlık hızının ölçülmesidir. Bu ölçümler birçok şekilde yapılabilmektedir. GPS kullanılmasının uygun olacağı düşünülse de küçük araçlarla ve kapalı bir alanda çalışma yapıldığı için bu düşünceden bu aşamada vazgeçilmiştir. GPS yerine tekerlekten dönüş bilgisini alarak bu ölçümlerin yapılması tercih edilmiştir.

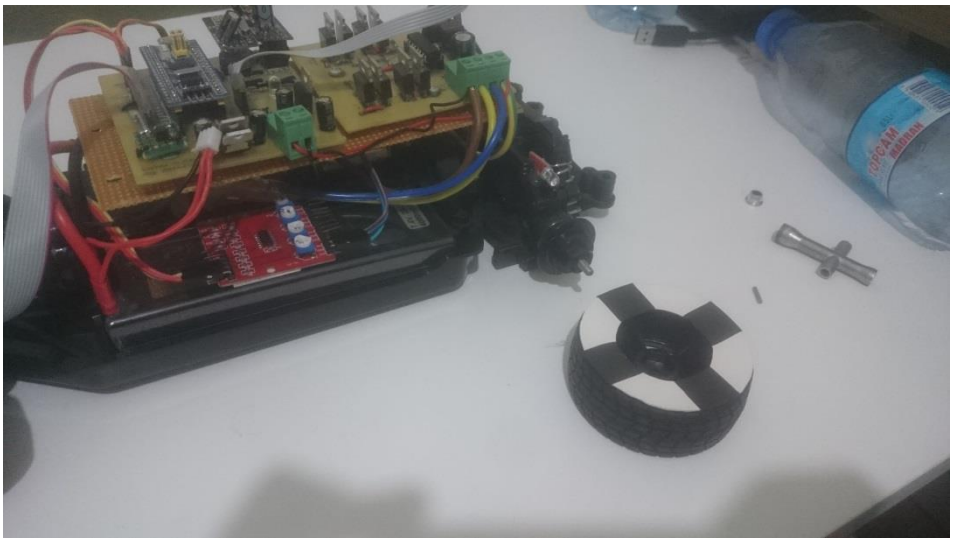
Tekerler üzerinden aracın konum ve hız verisini alabilmek için çeşitli yöntemler denenmiştir. Öncelikli olarak teker üzerinde 5 noktaya mıknatıs yerleştirilmiş ve *hall effect* sensör yardımı ile aracın konum bilgisi alınmıştır. Yapılan denemelerde *hall effect* sensöründen gelen veriler ile gerçekte olan veriler arasında bazı sapmalar olduğu gözlenmiş ve bu sorun çözülemediği için başka bir yöntem arayışına girilmiştir.

Bir diğer yöntem olarak dairesel enkoder kullanılması düşünülmüştür. Kullanılan enkoder bir tam turda 30 adet veri almaktadır ve bu da arabanın her 0.75cm hareketinde bilgi alınabilmesi anlamına gelmektedir. Enkoder araç tekerine dış taraftan bir devre yardımıyla bağlanmış ve uzun süre bu enkoder ile alınan veriler test edilmiştir. Sonuç olarak ise enkoderden alınan veriler ile gerçek verilerin birbirine uymadığı gözlemlenmiştir. Bunun sonucunda ise enkoder kullanılmasından da vazgeçilerek başka bir yol aranmaya başlanmıştır.

Sonuç olarak tekerlek üzerine yerleştirilen siyah ve beyaz bölümlerin renk sensörü yardımıyla okunması fikri öne çıkmıştır. Şekildeki gibi tekerleğin iç kısmına 4 adet siyah ve 4 adet beyaz şerit çizilmiştir. Yaklaşık olarak tekerleğe 0.5cm mesafede yerleştirilen renk sensörü ile de her renk geçişinde veri okunarak konum bilgisi elde edilmiştir. Tekerleğin bir tur dönüşünde 20.5cm yol aldığı bilindiği için her bir veri alışında aracın 2.56cm yol aldığı anlaşılmıştır. Yapılan denemeler neticesinde en büyük hata olarak %2'lik bir hata tespit edilmiştir. Ortalama olarak

bakıldığında ise %1'lik bir hata ile karşılaşılmıştır. Uygulama sonuçlarına göre bakıldığında 10 metrelik bir mesafede en fazla hata olarak 20 cm'lik bir hata ile ölçüm sonuçları elde edilmiştir. Ortalama olarak ise 10 cm'lik bir hata payı ortaya çıkmıştır. Projenin bu aşamasında bu büyüklükteki bir hatanın yok sayılabileceği düşüncesiyle yapılan denemelerde en iyi sonucu veren renk sensörü kullanılması kararlaştırılmıştır.

Şekil 3.12'de renk sensörünün montaj yeri ve tekerlek iç kısmındaki şeritler görülmektedir.

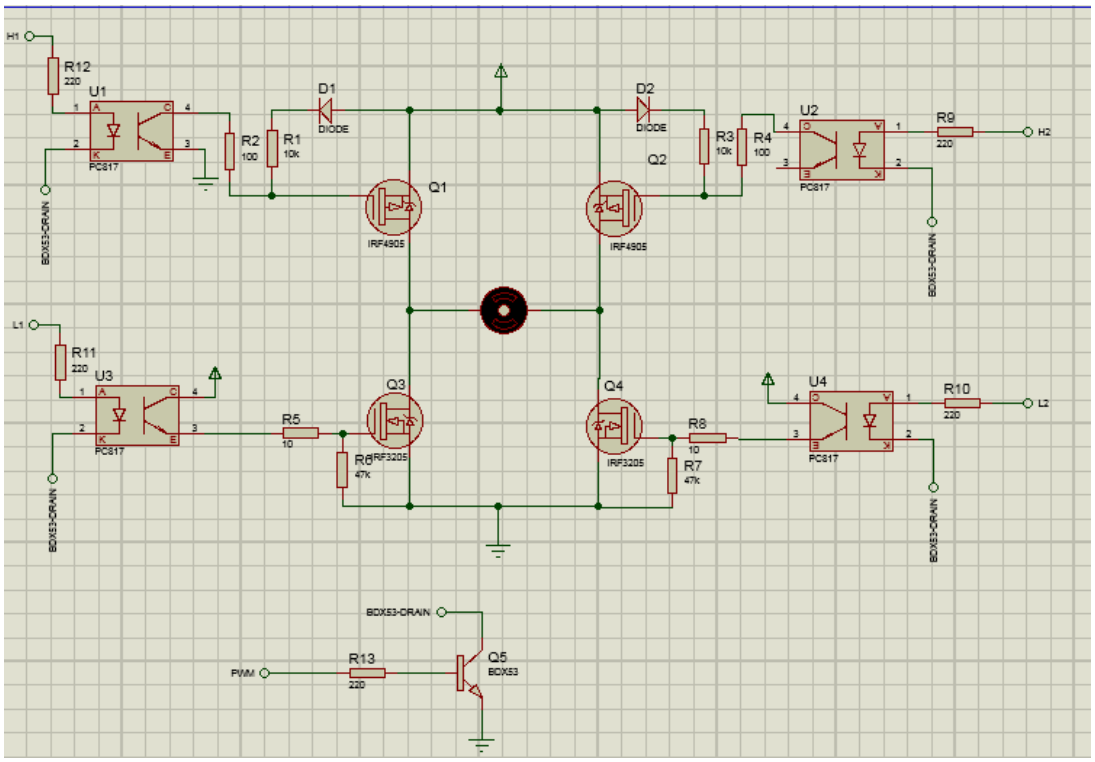


Şekil 3.12: Renk Sensörü ve Tekerlek İç Kısmına Ait Görsel

3.4 Motor Sürücü Kartı ve Devresi

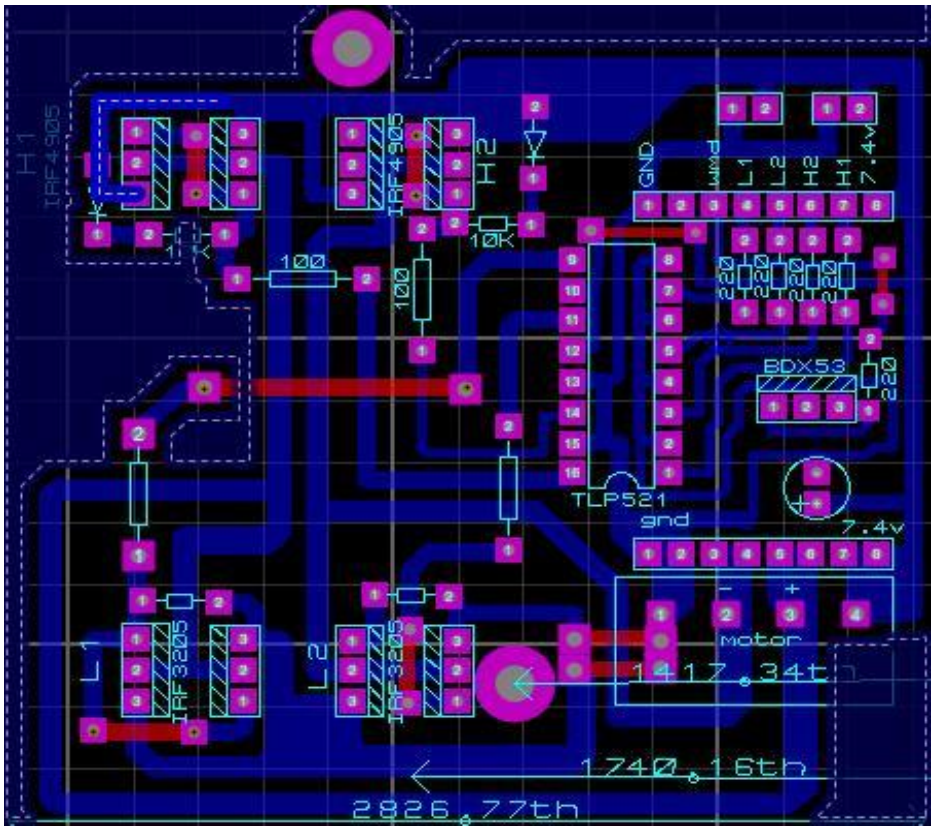
Projede motor sürücü kartı ile aracın hareketini sağlayan DC motorun sürülme işlemi gerçekleştirilmiştir. Bu kart üzerinde yer alan H1, H2, L2, L1 ve PWM pinleri ile kontrol kartından dönüş yönü ve PWM bilgisi alınmış ve DC motorun alınan bu verilere göre sürülmesi gerçekleştirilmiştir.

Şekil 3.13 ile motor sürücü kartına ait ISIS devre çizimi gösterilmiştir. Bu çizime bakıldığında kontrol kartından gelen H1, H2, L2, L1 ve PWM pinlerinin bağlantı noktaları da görülmektedir.



Şekil 3.13: Motor Sürücü Kartı ISIS Çizimi

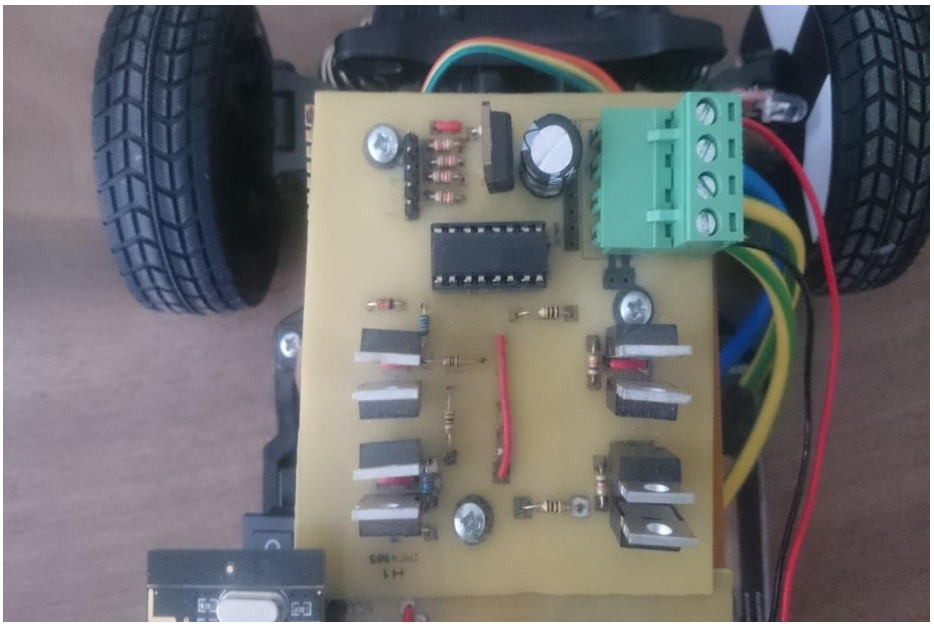
Motor sürücü devresinde daha önce de belirtildiği gibi MOSFET sürme tekniği kullanılarak devre çizilmiştir. Bu devrede çizim yapılırken H köprüsü modeli örnek alınmıştır. Devrede malzeme olarak ise 4 adet IRF4905 MOSFET, 4 adet IRF3205 MOSFET, 1 adet TLP521 optakuplör, 1 adet BDX53 transistör, 4 adet 100 Ω direnç, 4 adet 10 kΩ direnç, 5 adet 220 Ω direnç ve 2 adet 1n4148 diyot kullanılmıştır.



Şekil 3.14: Sürücü Kartı ARES Çizimi

ARES programı ile çizilen devre kartının çizim görüntüsü şekil 3.14'deki gibidir. Çizimi yapılan sürücü devresinde MOSFETlerin daha iyi anahtarlanabilmesi için 15 ya da 20V gibi bir voltaja ihtiyaç bulunmaktadır. Bu gerilimin altındaki gerilimlerde yapılan anahtarlamada ise MOSFETlerde ısınmalar meydana gelmektedir. Projeye baktığımızda besleme için kullanılan lipo pil iki hücrelidir. Bu da 7 – 8V gibi bir gerilim anlamına gelmektedir. Bundan dolayı MOSFETlerde ısınma sorunu ortaya çıkmıştır. İlk yapılan devrelerde alüminyum soğutucu kullanılarak bu sorun çözülmeye çalışılmış olsa da uzun süreli kullanımlarda ortaya çıkan olumsuz durumlar nedeniyle soğutucu da yetersiz kalmıştır.

Isınma sorunun çözümü için yapılan çalışmalarda her kolda 2 adet MOSFET kullanılması düşünülmüş ve son hâlde kullanılan devre her kolda 2 adet olmak üzere toplam 8 adet MOSFET kullanılarak çizim gerçekleştirilmiştir. Anahtarlama geriliminde değişiklik olmamasına rağmen her kolda yer alan 2 adet MOSFET ile ısınma sorunu ortadan kaldırılmıştır. Böylece soğutucu kullanılmasına da gerek kalmamıştır. Bu durum soğutucu maliyetini azaltmış olmasına rağmen MOSFET maliyetini arttırdığı için toplam maliyeti olumsuz yönde etkilemiştir.

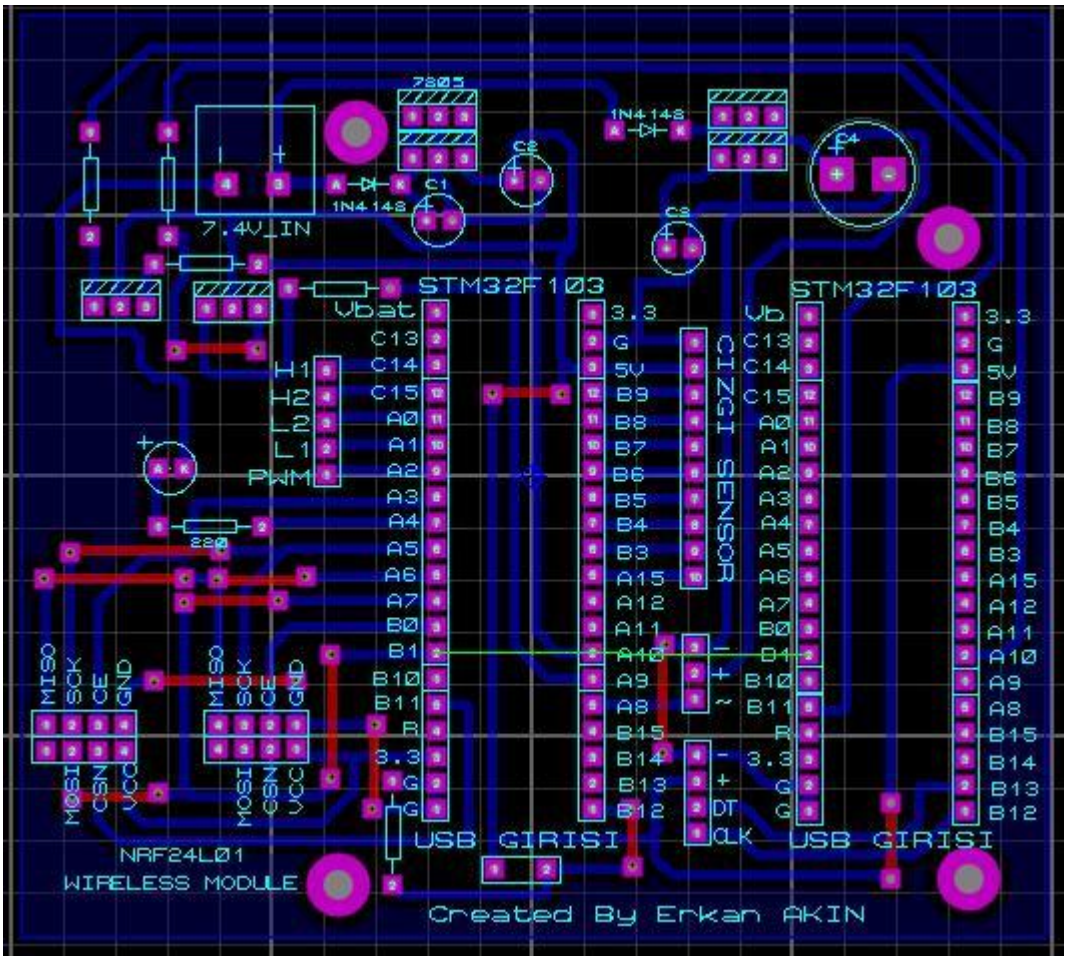


Şekil 3.15: Sürücü Kartı PCB Devresi

Şekil 3.15'de sürücü devresinin baskısı yapılmış ve malzemeleri yerleştirilmiş hâli görülmektedir.

3.5 Kontrol Kartı ve Devresi

Kontrol kartı araçlar üzerinde yer almaktadır. Bu kart ile haberleşme kartı tarafından araca gönderilen bilgilerin alınması ve alınan bu verilere göre de aracın hareketinin sağlaması amaçlanmıştır. Kontrol kartına genel olarak baktığımızda 7.4V voltaj ile beslendiği görülmektedir. Bu gerilimin mikrodenetleyici gerilimi olarak kullanılabilmesi için LM7805 model voltaj regülatörü kullanılmıştır. Aynı şekilde NRF24L01 model haberleşme modüllerinin beslemesi için ihtiyaç olan 3.3V gerilim elde edilmesi için de 2N3904 model regülatöre devrede yer verilmiştir. Kontrol kartında en önemli eleman olarak mikrodenetleyici gösterilebilir. Mikrodenetleyici olarak da daha önce bahsedildiği gibi STM mikrodenetleyici ailesi üyesi STM32F103C8T6 model mikrodenetleyici kullanılmıştır. Kontrol kartında bahsedilen elemanların dışında kullanılan elemanlar ise giriş – çıkış pinleri, buton, led, diyot, kondansatörler, dirençler, transistörlerdir.



Şekil 3.16: Kontrol Kartı ARES Çizimi

Şekil 3.16'da ARES çizimi yer alan kontrol kartı, üzerinde 2 adet STM32F103C8T6 mikrodenetleyici yer alacak şekilde tasarlanmıştır. İkinci mikrodenetleyici modüler olarak devreye alınıp kullanılabilecektir. Fakat projenin gelişimi ile birlikte ikinci mikrodenetleyicinin kullanılmasına gerek kalmamıştır. Kontrol kartı üzerinde iki adet NRF24L01 kablosuz haberleşme modülü kullanılmıştır. Bu modüllerden birisi verici olarak görev yaparken diğer modül alıcı olarak çalışmaktadır. Haberleşme modüllerinin CLK, MISO ve MOSI pinleri ortak olarak sırasıyla mikrodenetleyicinin A5, A6 ve A7 pinine bağlanmıştır. Haberleşme Modüllerinin CE pinleri sırasıyla B0 ve A3 pinine bağlanırken CSN pinleri sırasıyla B1 ve B11 pinlerine bağlanmışlardır. Haberleşme modülleri mikrodenetleyici üzerinde yer alan 3.3V gerilim ile beslenmişlerdir.

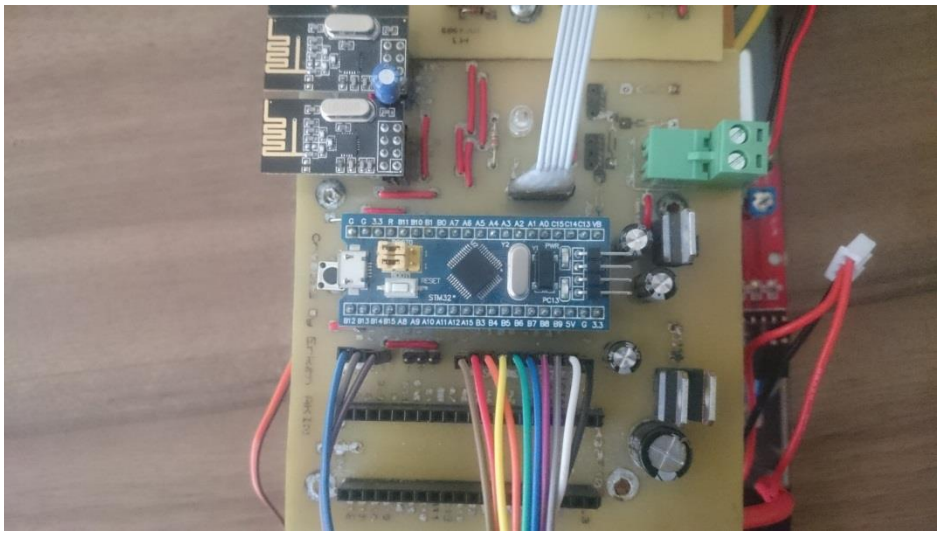
Kontrol kartı ile sürücü devresinin kontrol edilebilmesi için 5 adet pine ihtiyaç duyulmuştur. Bunlar sırasıyla H1, H2, L1, L2 ve PWM pinleridir. Burada H1,

H2, L2 ve L1 pinleri motorun dönüş yönünü ayarlayan pinlerdir. Bu pinler mikrodenetleyici üzerinde sırasıyla C14, C15, A0 ve A1 pinlerine bağlanmışlardır. Motor hızını ayarlayan PWM pini ise mikrodenetleyici üzerinde A2 pinine bağlanmıştır.

Teker üzerine yerleştirilen renk sensörü için 3 adet pine ihtiyaç duyulmuştur. Bu 3 pinden iki tanesi besleme pini olup üçüncü pin ise bilginin okunacağı OUTPUT pinidir. Bu pin de mikrodenetleyici üzerinde B13 pinine bağlanmıştır.

Servo motor kontrolü için de renk sensöründe olduğu gibi 3 adet pine ihtiyaç vardır ve bunlardan 2 tanesi besleme pinleridir. SGN isimli sinyal pini ise mikrodenetleyicinin A8 pinine bağlanmıştır.

Şerit takibinin sağlanması için aracın ön bölümünde yer alan QTR 8RC model sensör bilgilerinin okunabilmesi gerekmektedir ve bunun için 10 adet pine ihtiyaç duyulmuştur. Bu pinlerden 2 tanesi besleme pinidir. Kalan 8 adet pin ise sırasıyla mikrodenetleyicinin B9, B8, B7, B6, B5, B4, B3 ve B14 pinlerine bağlanmışlardır.

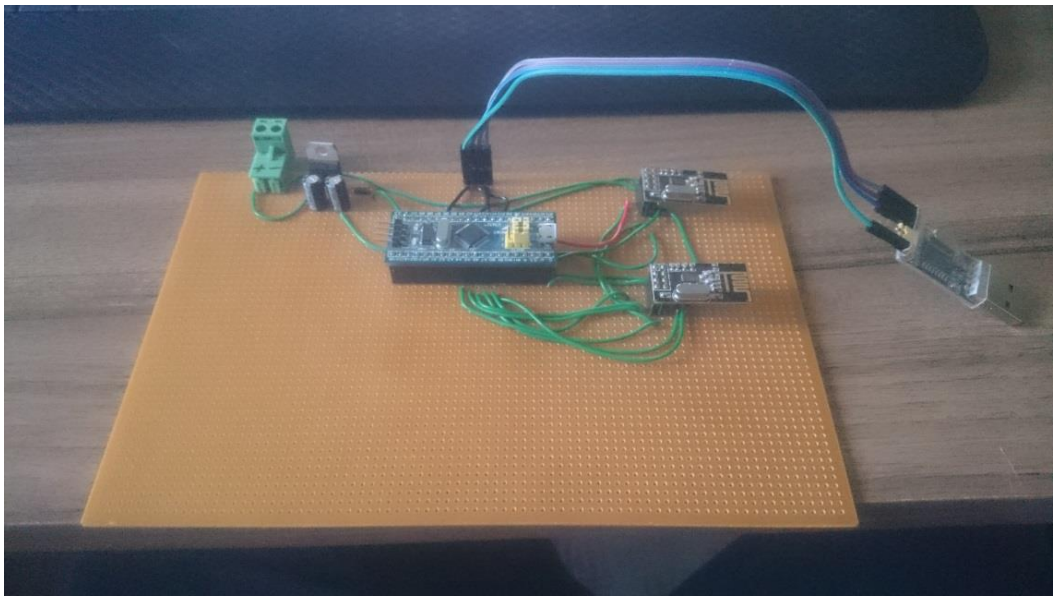


Şekil 3.17: Kontrol Kartı PCB Devresi

Şekil 3.17’de kontrol kartının baskı yapılmış ve malzemeleri yerleştirilmiş hâli yer almaktadır.

3.6 Haberleşme Kartı

Projede haberleşme kartı olarak kullanılan kart için ARES çizimi ve baskı devre yapılmamıştır. Üzerinde sadece bir adet mikroişlemci, ona bağlı 2 adet kablosuz haberleşme modülü ve voltaj regülatörü bulunacağı için devre delikli plaket üzerine kurulmuştur. Plaket üzerine kurulan haberleşme kartı devresi şekil 3.18'de görülmektedir.



Şekil 3.18: Haberleşme Kartı Devresi

4. YAZILIM

4.1 Giriş

Bu bölümde Projede kullanılan algoritmalardan, aracın bilgisayardan kontrolünü sağlayan C# programı ve kodlarından, haberleşme kartı ve alıcı kartının yazılımından bahsedilecektir.

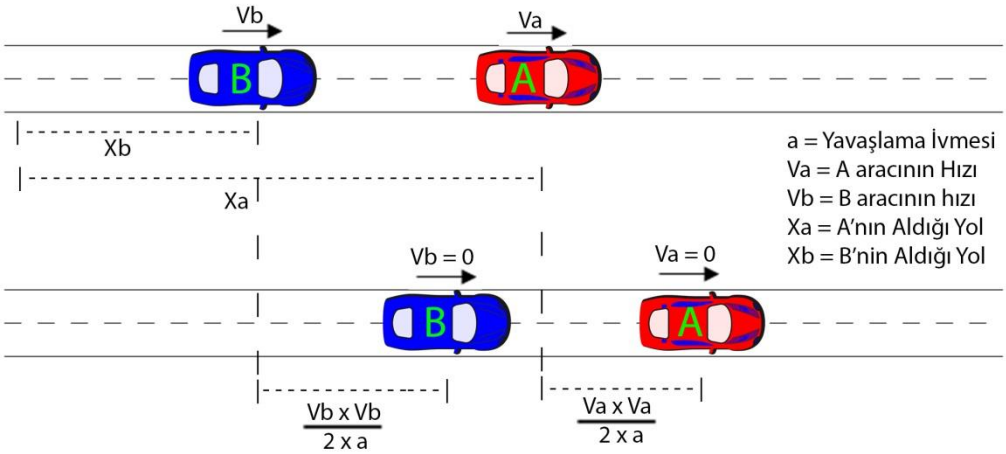
4.2 Yazılımda Kullanılan Şifrelemeler ve Algoritmalar

Kablosuz haberleşme yapılırken mesajlar *string* ifade türünde gönderilmektedir. Bu *string* ifade içinde gönderilecek mesajlar iki farklı harf veya karakter arasına yazılarak gönderilmiş ve karşı tarafta yine bu harfler ve karakterler arasında kalan kısımlar mesaj olarak kabul edilmiştir. Böylelikle haberleşmede istenmeyen karışıklıkların ya da dışarıdan araçlara müdahale edilmesi durumları ortadan kaldırılmıştır. Proje süresince kullanılan şifrelemeler Tablo 4.1'deki gibidir.

Tablo 4-1: Araçlar Arası Haberleşmede Şifreleme Tablosu

C#'dan Haberleşme Kartına	A + 1. Durum + B + 2. Durum + 3. Durum + D + E + 1. Sıra + F + 2. Sıra + G + 3. Sıra + H
Haberleşme Kartından 1. Araca	E + 1. Durum + F
	H + 1. Sıra + I
Haberleşme Kartından 2. Araca	F + 2. Durum + G
	I + 2. Sıra + J
Haberleşme Kartından 3. Araca	G + 3. Durum + H
	J + 3. Sıra + K
1. Araçtan Haberleşme Kartına	L + Hız + M + Alınan Yol + N
2. Araçtan Haberleşme Kartına	O + Hız + P + Alınan Yol + R
3. Araçtan Haberleşme Kartına	S + Hız + T + Alınan Yol + U
Haberleşme Kartından C#'a	V + 1. Hız + Y + 1. Konum + Z + 2. Hız + '+' + 2. Konum + % + 3. Hız + / 3. Konum + *

Yazılım kısmının daha net anlaşılabilmesi için öncelikle projede kullanılan algoritmalardan bahsedilecektir. MetroCar Sistemi temel olarak iki temel algoritma üzerine kurulmuştur. Bu algoritmalar takip algoritması ve birleşme algoritmasıdır.



Şekil 4.19: Takip Algoritması Genel Görünümü

Öncelikle takip algoritmasına bakalım. Şekil 4.19'da görüldüğü üzere önde yer alan araç A aracı ve arkada yer alan araç B aracı ve her iki aracın da sıfır konumunda hareketlerine başladıkları kabul edilmiştir. A aracının şu ana kadar aldığı yol Xa ve B aracının şu ana kadar almış olduğu yol Xb şeklindedir. Algoritmaların tam ve düzgün olabilmesi için araçların sabit ivmeli hareket yapmaları projenin en önemli kısmı olarak gösterilebilir. Sabit ivmeli hareket yapıldığı için kaç saniye sonra arabaların nerede olacaklarına ilişkin tam ve net bilgiler elde edilebilmektedir. Algoritmaya dönülecek olursa takip algoritması 4 kısım hâlinde incelenebilir. İlk olarak araçların arasındaki mesafe kontrol edilir. Eğer mesafe takip mesafesi olarak seçilen mesafeden kısa ise arkadaki aracın yavaşlaması istenir. Eğer ki aradaki mesafe güvenli mesafeden yüksek ise ikinci şart olarak araçların hızları kıyaslanır ve öndeki aracın hızı arkadaki aracın hızından daha yüksekse arkadaki aracın hızını artırması istenir. Bu şart da sağlanmıyorsa 3. Şart olarak takip mesafesi formülü uygulanır. Bu noktada takip mesafesi için uygulanan formülün nasıl elde edildiğinden bahsedilecektir.

Takip mesafesi için uygulanan formül esas olarak öndeki aracın ani olarak durması durumunda arkadaki aracın öndeki araca çarpmadan durup duramayacağının kontrolü olarak söylenebilir. Va hızıyla giden öndeki araç sabit a ivmesi ile durmaya kalkarsa;

$$\frac{\mathrm{Va} * \mathrm{Va}}{2a} \tag{4.1}$$

Uzaklıkta durabilecektir. Böylelikle sıfır konumundan bu yana aldığı toplam yol;

$$\mathrm{Xa} + \frac{\mathrm{Va} * \mathrm{Va}}{2a} \tag{4.2}$$

Şeklinde hesaplanabilecektir. Aynı şekilde Vb hızıyla giden arkadaki araç da aynı a ivmesi ile durmaya kalktığında;

$$\frac{\mathrm{Vb} * \mathrm{Vb}}{2a} \tag{4.3}$$

Mesafede durabilecek ve B aracının sıfır konumundan bu yana aldığı toplam yol;

$$\mathrm{Xb} + \frac{\mathrm{Vb} * \mathrm{Vb}}{2a} \tag{4.4}$$

Şeklinde hesaplanabilecektir.

$$\Delta \mathrm{X} = \mathrm{Xa} - \mathrm{Xb} \tag{4.5}$$

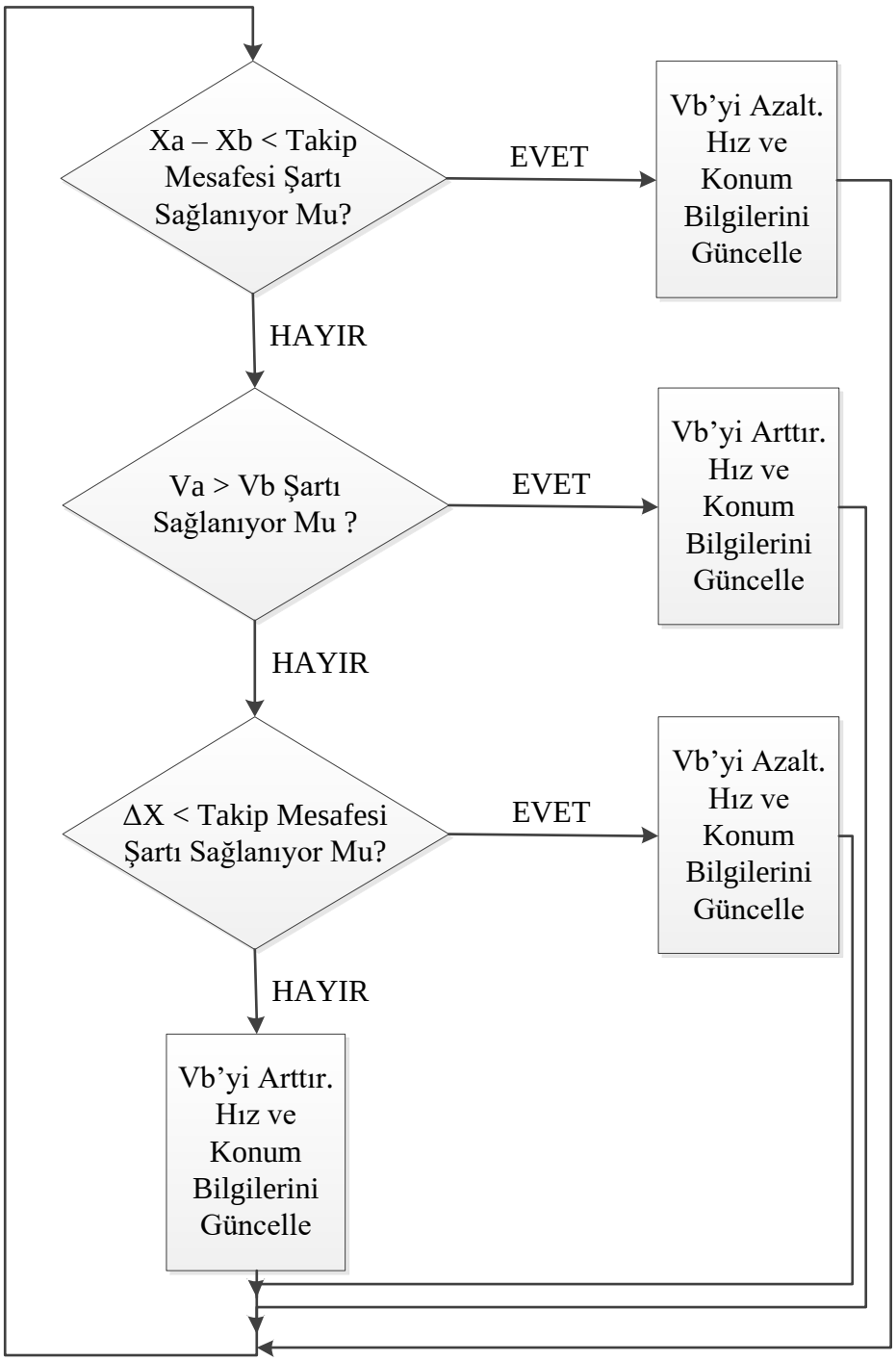
Formülü ile araçların durdukları andaki konumları arasındaki fark hesaplanacaktır. Bunun neticesinde yukarıdaki denklemlerin birleştirilmesi ile;

$$\Delta \mathrm{X} = \mathrm{Xa} + \frac{\mathrm{Va} * \mathrm{Va}}{2a} - Xb - \frac{\mathrm{Vb} * \mathrm{Vb}}{2a} \tag{4.6}$$

Formülü takip formülü olarak ortaya çıkmıştır.

Algoritmaya geri dönülecek olursa ilk iki şartın sağlanmamasının ardından kontrol edilecek üçüncü şart olarak ΔX mesafesi ile takip mesafesi kıyaslanır. Eğer ki ΔX mesafesi takip mesafesinden kısa ise arkadaki aracın yavaşlaması istenir. Algoritmanın son kısmında ise hiçbir şartın sağlanmadığı durum olarak arkadaki aracın hızlanması istenir ve algoritma bu şekilde tamamlanmaktadır (Bozuyla 2018).

Yukarıda yazılı olarak anlatımı yapılan algoritmanın blok diyagram olarak ifadesi şekil 4.20'deki gibi gösterilebilir.

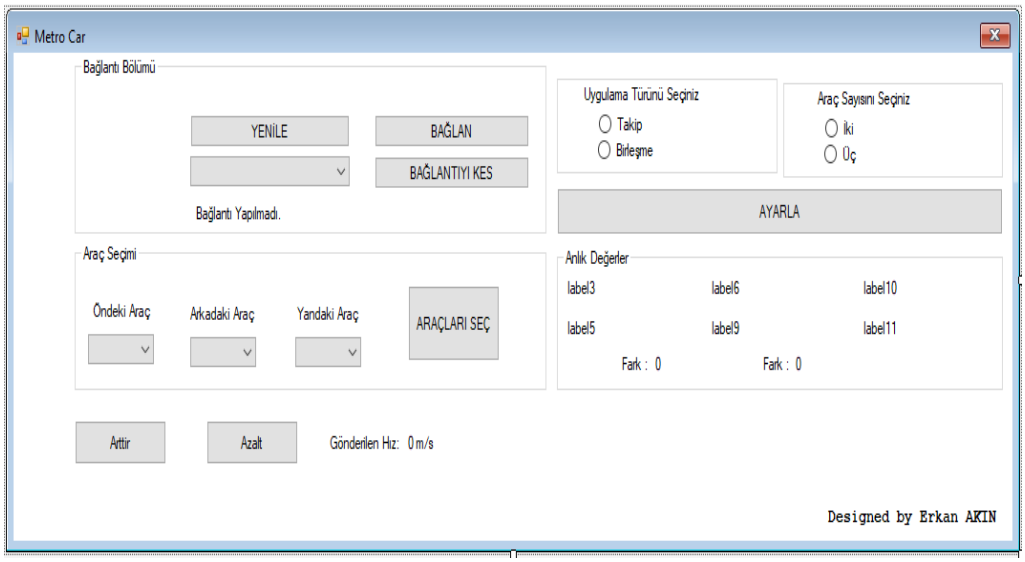


Şekil 4.20: Takip Algoritması Blok Diyagramı

İkinci temel algoritma olan birleşme algoritması ise yan yoldan gelen aracın ana yola nasıl bağlanacağının kararının verilmesini sağlamaktadır. Yan yoldan gelen araç ana yola bağlanana kadar 3 bölgeden geçmektedir. Bu bölgeler sırasıyla serbest bölge, kontrol bölgesi ve kritik bölgedir. Serbest bölge boyunca araba hızlanır ve kontrol bölgesine geldiğinde artık ana yoldan gelen arabalarla aynı hıza ulaşmış

olması gerekmektedir. Ana yoldan gelen araçların hızlarına ulaşan araç ana yoldan gelen araçlarla arasındaki mesafelere bakar ve yola hangi arabaların arasından bağlanacağına karar vermektedir. Kritik bölgede araçlar karar verilen şekilde sıralanırlar ve aralarında yeterli takip mesafesi olacak şekilde bağlantı noktasından sırasıyla geçerler. (Bozuyla 2018)

4.3 C# Programı ve Yazılımı



Şekil 4.21: Proje İçin Hazırlanmış Olan C# Programına Ait Görsel

Visual Studio ile hazırlanmış olan C# programının ekran görüntüsü Şekil 4.21'deki gibidir. Öncelikle programın genel olarak tanıtımı yapılacaktır. Programın ilk açılış ekranında sadece bağlantı bölümü aktif olarak gelmektedir. Öncelik olarak bu bölümde haberleşme kartı ile haberleşmenin yapılması gerekmektedir. *Combobox* içerisinde listelenen portlardan haberleşme kartının bağlı olduğu port bulunur ve bağlan butonuna tıklanır. Eğer port bulunamıyorsa yenile butonuna tıklanarak portlar yenilenir.

Bağlan butonuna tıklandığında uygulama türünü ve araç sayısını seçtiğimiz bölüm aktif olacaktır. Bu bölümde uygulama türü olarak takip veya birleşme seçimi yapılır. Araç sayısı olarak 2 ve 3 seçeneğinden birisi tercih edilir. Ayarla butonuna tıklanması ile birlikte araç seçimi bölümü düzenlenerek karşımıza gelmektedir.

Eğer araç sayısı bölümünde 2 araba seçilmişse araç seçimi bölümünde ekrana iki adet *combobox* çıkacaktır. Eğer 3 araba seçilmişse 3 adet *combobox* ile seçim yapılacaktır. Araçların konumuna göre 1, 2 veya 3 numaralı araçlar seçilir ve araçları seç butonuna tıklanır. Araçları seç butonuna tıkladığımız anda artık hangi araba hangi konumda olduğunu, takip mi yapacağını ya da önde mi gideceğini bilmektedir. Aynı şekilde araçlardan alınan anlık hız ve konum bilgileri de anlık değerler bölümünde ekrana yazdırılmaktadır.

Programın görünen kısımlarında bunlar yaşanırken arka planda ise sürekli olarak *timer* içerisinde araçlardan veriler alınır ve işlenerek araçlara hızlanması veya yavaşlaması gerektiği bilgisi gönderilmektedir.

Şekil 4.22: Proje İçin Hazırlanmış Olan C# Program Kodlarının Bölüm Bölüm Ayrılmış Görseli

Şekil 4.22'de C# programına ait kodların bölüm bölüm ayrılmış olarak genel görünümüne ait ekran görüntüsü yer almaktadır. Bu bölümlere tek tek değinerek programın genel çalışma mantığını anlatılmaya çalışılacaktır.

Değişkenler bölümü isminden de anlaşılacağı üzere program içerisinde kullanılan tüm değişkenlerin yer aldığı bölümdür.

Değişkenler bölümünden sonra programın devamında *form_load* bölümü yer almaktadır. Bu bölüm program çalıştığı anda yapılması gereken işlemlerin yer aldığı bölümdür. Bu bölümde öncelikle 1 numaralı seri porttan 30 numaralı seri porta kadar olan tüm portlar açılıp kapatılmaya çalışılarak hangilerinin kullanılabileceği

belirlenmiştir. Sonrasında ise kullanılabilecek olan portlar *combobox* nesnesiiçine eklenmiştir. Bu işlemlerin yapıldığı kodlar aşağıdaki gibidir:

```
for(int i=0; i<30; i++)
{try
{serialPort1.PortName = "COM" + i.ToString();
serialPort1.Open();
combobox1.Items.Add(serialPort1.PortName);
serialPort1.Close(); }
catch(Exception)
{continue; } }
```

Sonrasında yine *form_load* içerisinde uygulama ve araç seçimi yapılan kısımlar, araç seçimi bölümü ve anlık değerler bölümleri pasif hâle getirilmiştir. Araç seçimi yapılabilmesi için yer alan *combobox* nesnelerinin içerisine "1" , "2" ve "3" de bu bölümde eklenmiştir.

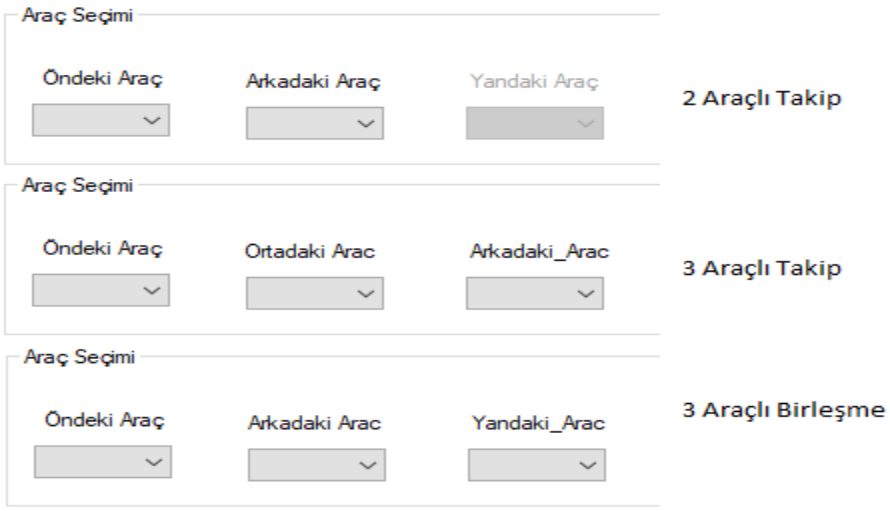
Bir sonraki bölüm olan yenile bölümü içerisinde yenile butonuna tıklandığında işlenecek kodlar yer almaktadır. Bu bölümde öncelikle port numaralarının yer aldığı *combobox* içerisindeki değerler sıfırlanır ve *form_load* bölümğnde olduğu gibi açık olan portların bulunup *combobox* içerisine eklenmesi işlemi yapılır.

Bağlan bölümünde bağlan isimli butona tıklandığı anda yapılacak olan işlemler yer almaktadır. Öncelikle *combobox* ile seçilen port adresine bağlantı gerçekleştirilir. Ekrana "COMx butonuna bağlantı gerçekleşti." şeklinde bir mesaj yazılır ve uygulama türü ile araç seçiminin yapılacağı bölüm aktif hâle getirilir.

Bağlantı kes bölümünde bağlantı kes isimli butona tıklandığı durumda yapılacak olan işlemler yer almaktadır. Öncelikle seri port kapatılarak haberleşme kesilir. Bağlantı bölümü haricindeki tüm kısımlar pasif yapılır. Ekrana bağlantının kesildiği yazılarak kullanıcıya mesaj verilir.

Başlat bölümü aslında ayarla bölümü olarak düşünülebilir. Programda daha sonra yapılan değişiklikler nedeniyle buton ismi değiştirilmediği için başlat butonu olarak kalmıştır. Fakat bu bölümde yer alan işlemler ayarla butonuna basıldığında yapılan işlemlerdir. Bu kısımda öncelikle *radio* butonlarının durumlarına bakılır. Uygulama türü seçimine göre uygulama isimli değişkene 1 ve 2 değerleri atanır. Uygulama değişkenine 1 değerinin atanması takip yapılacağı anlamına gelirken 2 değerinin atanması birleşme yapılacağı anlamına gelir. Araç sayısı kısmında da seçilen değerler okunarak *arac_sayisi* isimli değişkene 2 ya da 3 değeri atanır.

Sonrasında yapılan seçimlere göre araç seçimi kısmı düzenlenir. Şekil 4.23'de farklı seçim durumlarına göre araç seçimi bölümünün durumu yer almaktadır.



Şekil 4.23: Farklı Durumlara Göre Araç Seç Bölümü Ekran Görüntüsü

Başlat bölümünde son olarak araç seçimi bölümü aktif hâle getirilir. Program icra ederken karışıklıklara yer vermemek için de uygulama seçimi ve araç seçimi bölümleri pasif hâle getirilir.

Sıradaki bölüm araç seçimlerinin yapılacağı bölümdür. Bu bölümde öncelikle araç seçimleri için yukarıdaki şekilde de yer alan *combobox*lar kullanılarak hangi arabanın nereden harekete başlayacağı belirtilir. Programın modüler bir yapıda olmasından dolayı bu bölüm 3 ayrı parçada değerlendirilmiştir. 1. durum 2 araçlı takip, 2. durum 3 araçlı takip ve son durum birleşme durumdur. Her durumda yapılan işlemler benzer olduğu için burada sadece 2 araçlı takip durumunda işlenen kodlardan bahsedilecektir. Öncelikle *on* ve *arka* değişkenlerine *combobox*larda

seçilen ifadeler atanır. Sonrasında *on* değişkeni eğer, eğer ise ifadeleri ile kontrol edilerek on değerine 1 atanmışsa *sira_1* değişkenine "ii", *on* değişkenine 2 atanmışsa *sira_2* değişkenine "ii" ve son olarak *on* değişkenine 3 atanmışsa *sira_3* değişkenine "ii" değeri atanır. Aynı şekilde bir kontrol yapısıyla *arka* değişkeni de kontrol edilir ve *sira_1, sira_2* ya da *sira_3* değişkenlerinden birisine "aa" değeri atanır. Sorasında mesaj olarak bu *sira_1* ve *sira_2* değişkenleri haberleşme kartına gönderilerek haberleşme kartı yardımıyla araçlara iletilirler. Böylelikle araçlar hangi konumda olduklarına karar verirler. Sonrasında ise *timer* nesnesi aktif hâle getirilir.

Araç seçimi bölümünden sonra yer alan bölümler buton artı ve buton eksi bölümleridir. Bu bölümlerde buton_arti ve buton_eksi butonlarına basıldığında yapılacak olan işlemler yer alır. Arttırma ya da azaltma butonu önde yer alan aracın hızının merkez tarafından belirlenmesi için kullanılırlar. Arttırma ya da azaltma ile belirlenen hız değişkeni bu bölümde önde yer alan araca gönderilir.

Timer bölümü programın asıl kodlarının icra edildiği bölüm olarak gösterilebilir. Bu bölümde öncelikle arabalardan haberleşme kartına gönderilen hız ve konum bilgileri C# programı tarafından aşağıdaki kodlar kullanılarak alınır:

```
String gelen_mesaj = serialPort1.ReadExisting();
int indis_1 = gelen_mesaj.IndexOf('V');
int indis_2 = gelen_mesaj.IndexOf('Y');
int indis_3 = gelen_mesaj.IndexOf('Z');
int indis_4 = gelen_mesaj.IndexOf('+');
int indis_5 = gelen_mesaj.IndexOf('%');
int indis_6 = gelen_mesaj.IndexOf('/');
int indis_7 = gelen_mesaj.IndexOf('*');
if (indis_2 - indis_1 > 1 && indis_3 - indis_2 > 1 && indis_4 - indis_3 > 1 &&
indis_5 - indis_4 > 1 && indis_6 - indis_5 > 1 && indis_7 - indis_6 > 1)
{ try {
alinan_hiz1 = gelen_mesaj.Substring(indis_1 + 1, indis_2 - indis_1 - 1);
alinan_hiz1 = alinan_hiz1.Replace(".", ",");
hiz1 = Convert.ToDouble(alinan_hiz1);
konum1 = Convert.ToDouble(gelen_mesaj.Substring(indis_2 + 1, indis_3 - indis_2 -
1));
```

```
alinan_hiz2 = gelen_mesaj.Substring(indis_3 + 1, indis_4 - indis_3 - 1);
alinan_hiz2 = alinan_hiz2.Replace(".", ",");
hiz2 = Convert.ToDouble(alinan_hiz2);
konum2 = Convert.ToDouble(gelen_mesaj.Substring(indis_4 + 1, indis_5 - indis_4 - 1));
alinan_hiz3 = gelen_mesaj.Substring(indis_5 + 1, indis_6 - indis_5 - 1);
alinan_hiz3 = alinan_hiz3.Replace(".", ",");
hiz3 = Convert.ToDouble(alinan_hiz1);
konum3 = Convert.ToDouble(gelen_mesaj.Substring(indis_6 + 1, indis_7 - indis_6 - 1));  }
catch (Exception)
{ }}
```

Bundan sonraki kodlar iki ana başlık olarak incelenecektir. İlk ana başlığımız takip kodlarının yapıldığı kısımdır. Eğer uygulama olarak takip yapılmışsa bu kodlar icra edilir. Takip bölümü de yine araç sayısına bağlı olarak iki bölümden oluşmaktadır. Öncelikle 2 araçlı takip programının nasıl işlediğinden bahsedilecektir.

Programın araç seçme kısmında *on* ve *arka* değişkenlerine 1, 2 veya 3 değerlerinden birisinin atandığı belirtilmişti. Bu değişkenler bu bölümde tekrar kontrol edilir ve *hiz_on*, *konum_on*, *hiz_arka* ve *konum_arka* değişkenlerine uygun arabalardan gelen veriler atanır. Örnek olarak öndeki arabanın 1 ve arkadaki arabanın 2 olarak seçildiğini kabul edelim. Bu durumda *on* değişkeni "1" ve *arka* değişkeni "2" olur. Bu değerlerin kontrol edilmesiyle birlikte *hiz_on* değişkenine *hiz1* değeri, *konum_on* değişkenine *konum1* değeri atanırken *hiz_arka* değişkenine *hiz2* değeri ve *konum_arka* değişkenine *konum2* değeri atanır. Sonrasında *double* türünde *formul1* değişkeni tanımlanır ve değer olarak;

*formul1 = konum_on * 0.01 + hiz_on * hiz_on * 2 – konum_orta * 0.01 – hiz_orta * hiz_orta * 2;*

işleminin sonucu atanır. Tam sayı türünde tanımlanan *fark1* değişkenine ise; $konum\_on - konum\_arka$ işleminin sonucu atanır. Sonrasında ise daha önce bahsedilmiş olan takip algoritmasına göre kontroller yapılarak arkadaki aracın

hızlanması ve yavaşlaması gerektiği belirtilir. Yine verilen örnek üzerinden bakılacak olursa bu algoritma aşağıdaki gibi düşünülebilir:

```
if(fark1 < takip_mesafesi)
{ durum2 = "-"; }
else if(hiz_on > hiz_arka)
{ durum2 = "+"; }
else if(formul1 < takip_mesafesi * 0.01)
{ durum2 = "-"; }
else
{ durum2 = "+"; }
```

Sonrasında gerekli olan *label* nesnelerine bu hız ve konum değerleri yazdırılarak anlık değerler bölümünde araçların hız ve konumları kullanıcıya sunulur.

Üç araçlı bir takip seçilmişse daha önceden *on, orta* ve *arka* değişkenlerine atanan değerlere göre *hiz_on, konum_on, hiz_orta, konum_orta, hiz_arka* ve *konum_arka* değişkenlerine uygun arabalardan gelen hiz ve konum değerlerini atayarak işe başlanır. Yine modüler olan programın bu bölümünün de kolay anlaşılabilmesi için bir örnekle anlatım yapılacaktır.

Örnek olarak bakarsak öndeki araç 1, ortadaki araç 2 ve arkadaki araç 3 numaralı araç olarak seçilmiş olsun. Bu durumda *hiz_on* ve *konum_on* değişkenlerine *hiz1* ve *konum1* değerleri, *hiz_orta* ve *konum_orta* değişkenlerine *hiz2* ve *konum2* değerleri ve son olarak *hiz_arka* ve *konum_arka* değişkenlerine *hiz3* ve *konum3* değerleri atanır. Sonrasında *formul1* ve *formul2* değişkenleri *double* türünde tanımlanarak;

```
formul1 = konum_on * 0.01 + hiz_on * hiz_on * 2 – konum_orta * 0.01 – hiz_orta * hiz_orta * 2;
formul2 = konum_orta * 0.01 + hiz_orta * hiz_orta * 2 – konum_arka * 0.01 – hiz_arka * hiz_arka * 2;
```

değerleri bu değişkenlere atanırlar. Tam sayı türünde tanımlanan *fark1* ve *fark2* değişkenlerine de sırasıyla *konum_on – konum_orta* ve *konum_orta – konum_arka*

işlemlerinin sonuçları atanır. Sonrasında ise takip algoritmaları ön ve ortadaki için ayrı, orta ve arkadaki için ayrı olacak şekilde aşağıdak gibi uygulanır:

```
if(fark1 < takip_mesafesi)
{ durum2 = "-"; }
else if(hiz_on > hiz_orta)
{ durum2 = "+"; }
else if(formul1 < takip_mesafesi * 0.01)
{ durum2 = "-"; }
else
{ durum2 = "+"; }
if(fark2 < takip_mesafesi)
{ durum3 = "-"; }
else if(hiz_orta > hiz_arka)
{ durum3 = "+"; }
else if(formul2 < takip_mesafesi * 0.01)
{ durum3 = "-"; }
else
{ durum3 = "+"; }
```

İki araçlı takip durumunda olduğu gibi yine gerekli olan *label* nesnelerine anlık hız ve konum bilgileri yazılarak kullanıcıya anlık değerler bölümünde gösterilmektedir.

Timer bölümündeki son kısım olarak birleşme bölümü gösterilebilir. Öncelikle birleşme olayının nasıl gerçekleştiğinden bahsedilmelidir. Birleşme durumunda önde bir araç, arkada bir araç ve öndeki arabanın konumuna denk yan yolda bir araç ile harekete başlanmaktadır. Önce serbest bir bölgede öndeki ve yandaki araç kendisine verilen hız değerine göre hareket ederken arkadaki araç önündeki aracı takip edecek şekilde hareket etmektedir. Sonra araçlar kontrol bölgesine ulaşacaklar ve bu bölgede birleşmeden sonra hangi sıra ile hareketlerine devam edeceklerine karar vereceklerdir. Sonra kritik bölgede artık takip algoritmasını yaparcasına sıralanma gerçekleşecek ve birleşme yapacaklardır. Birleşmeden sonra da 3 araçlı takip algoritması işlemeye devam edecektir.

Daha önceden değerleri atanan *on, arka* ve *yan* değişkenleri öncelikle kontrol edilir ve hangi arabadan gelen bilgilerin hangi değişkenlere atanacakları belirlenir. Yine örnek vererek devam edilecektir. Öndeki araç 1 numara, arkadaki araç 2 numara ve yandaki araç 3 numara olacak şekilde seçilsin. Böylelikle *hiz_on* ve *konum_on* değişkenlerine *hiz1* ve *konum1* değerleri, *hiz_arka* ve *konum_arka* değişkenlerine *hiz2* ve *konum2* değerleri ve son olarak *hiz_yan* ve *konum_yan* değişkenlerine *hiz3* ve *konum3* değerleri atanmaktadır. Sonrasında öndeki araç ile arkadaki araç arasında 2 araçlı takip algoritması işletilerek arkadaki aracın hızlanması ve yavaşlaması istenir. Öndeki ve yandaki araca ise artır ya da azalt butonları ile belirlenen hızlarda gitmesi istenmektedir. Bu hareketler serbest bölge boyunca devam etmektedir.

Belirlenen kontrol bölgesine geldiğimizde yandaki arabanın nereye girerek bağlanacağı karar verilir. Yandaki aracın bağlanma ihtimali olan 3 varyasyon vardır. Yandaki araç en öne, ortaya ya da en arkaya geçecek şekilde bağlanma işlemi gerçekleşebilir. Yine 3 durumu da tek tek anlatmak yerine bir örnek üzerinden devam edilecektir. Yandaki aracın iki aracın ortasına girdiği duruma bakalım. Bu durumda *orta* adlı değişkene *yan* değişkeninin değeri atanır. Böylelikle yandaki arabanın artık ortadaki araba olduğu araca bildirilir. Artık öndeki, ortadaki ve arkadaki şeklinde 3 aracımız belirlenmiş olmaktadır. Bundan sonraki süreç 3 araçlı takip ile aynıdır.

4.4 Haberleşme Kartı Yazılımı

Haberleşme kartında kontrolcü olarak STM32F103C8T6 model mikrodenetleyici kullanmaktadır. Bu mikrodenetleyici kartın programlanması için Arduino programı kullanılmıştır. Haberleşme kartı üzerinde yer alan *wireless* modüller yardımı ile araçlar ile haberleşmekte ve aynı zamanda seri port yardımı ile C# programı ile haberleşme gerçekleştirilmektedir. Haberleşme kartında yer alan kodlar ile ilgili bazı bölümler program ekran görüntüleri yardımıyla anlatılacaktır.

Arduino yazılımı ile yazılan kodlar üç bölümden oluşmaktadır. İlk bölüm olarak henüz işlemler başlamadan önce yapılan tanımlamalar, kullanılan değişkenlerin tanımlanması vb. gibi kodların yer aldığı kısımdır.

Bu kısımda öncelikle gerekli olan kütüphaneler eklenmiştir. Sonrasında haberleşme ile ilgili olan CS ve CSN pinlerini sisteme tanıtılmış ve RF24 kütüphanesi sayesinde *radio* ve *radio2* ifadeleri bu CS ve CSN'ye bağlı olarak oluşturulmuştur. Sonraki bölümde de programda kullanılan değişkenler tanımlanmıştır. Belirtilen işlemlerin yapılması için gerekli olan kodlar aşağıdaki gibidir:

```
#include <SPI.h>
#include "nRF24L01_STM32.h"
#include "RF24_STM32.h"
#define RF_CS1 PB0
#define RF_CS2 PB12
#define RF_CSN1 PB1
#define RF_CSN2 PB11
RF24 radio(RF_CS1, RF_CSN1);
RF24 radio2(RF_CS2, RF_CSN2);
const uint64_t pipes[2] = { 0xe7e7e7e7e7LL, 0xc2c2c2c2c2LL };
char verig[50];
char veria[50];
String C_gelen_mesaj, durum1, durum2, durum3, hiz1 = "0", konum1 = "0", hiz2 = "0", konum2 = "0", hiz3 = "0", konum3 = "0";
```

Sonrasında program akışı içerisinde yer alan kısım *setup* bölümüdür. Bu bölüm program içerisinde gerekli ayarlamaların yapıldığı kısımdır. Şekil 4.29'da görüldüğü üzere seri haberleşme 9600 bandında başlatılmıştır. *Radio*'lar başlatılarak haberleşmenin başlaması sağlanmıştır. Ayrıca *radio* nesneleri ile ilgili kanal, aktarım hızı vb. gibi ayarlamalar yapılmıştır.

```
void setup()
{ Serial1.begin(9600);
 radio.begin();
 radio.setChannel(60);
radio.setDataRate(RF24_2MBPS);
radio.setPALevel(RF24_PA_MAX);
 radio2.begin();
```

radio2.setChannel(50);
radio2.setDataRate(RF24_2MBPS);
radio2.setPALevel(RF24_PA_MAX);}

Programın son bölümü ise *loop* yani döngü bölümüdür. Bu bölümde yer alan kodlar sürekli olarak işlenmektedir. Bu bölümde karmaşıklığı azaltmak için *serial, deger_alma, gönder, al, değerleri_coz* isimli fonksiyonlar tanımlanmış ve bu fonksiyonlar kullanılmıştır. Sonrasında da bir şifreleme ile araçlardan gelen hız ve konum bilgilerini seri porta yazılmıştır. Bu bölümde yer alan kodlar aşağıdaki gibidir:

void loop()
{ serial();
değer_alma();
gönder();
al();
değerleri_coz();
Serial1.println("V" + hiz1 + "Y" + konum1 + "Z" + hiz2 + "+" + konum2 + "%" + hiz3 + "/" + konum3 + ""); }*

Bu bölümde yer alan kodlara değinmek gerekirse *serial* fonksiyonu C# programından gelen verileri almak için kullanılmaktadır. *deger_alma* fonksiyonu ise C#'dan alınan bu ham verinin işlenerek hangi arabaya hangi bilginin gideceğinin ayrıştırıldığı kısımdır. *gonder* fonksiyonu ile bu veriler araçlara gönderilmektedir. *al* fonksiyonu ile araçlardan gelen hız ve konum bilgileri ham olarak alınırken *değerleri_coz* fonksiyonu ise alınan bu verilerin çözülmesi için kullanılmaktadır.

4.5 Araç Kontrol Kartı Yazılımı

Projede kullanılmakta olan üç araba için de kontrol kartı birbirine denk yazılımlara sahiptir. Sadece şifreleme konusunda ve aracın servo açıları konusunda temel ayrılıklar bulunmaktadır.

Haberleşme kartında olduğu gibi kontrol kartlarında da Arduino ile programlama yapılmıştır. Bu yüzden kodların dağılımı haberleşme kartına oldukça benzerdir. Öncelikle kütüphanelerin eklenmesi ve değişkenlerin tanımlanması yapılmıştır. Sonrasında *setup* kısmında gerekli ayarlamalar yapılmış ve son olarak *loop* bölümüne ise asıl kodlar yazılmıştır. Bu bölümler haberleşme kartı ile neredeyse aynı olduğu için çok detaya girilmeyecektir ancak haberleşme kartında bulunmadığı hâlde burada yer alan bazı kodlara değinilecektir. Bunlara bir örnek olarak kesme fonksiyonunun kullanımı gösterilebilir.

Arduino kesme işlemi için öncelikle mikrodenetleyicide kesme olarak kullanılacak pinin belirlenmesi gerekmektedir. Projede bu pin B13 pini olarak belirlenmiştir. *Setup* içerisinde "*attachInterrupt(PB13,Rotaty, CHANGE)* "kodu kullanılarak kesme pini sisteme tanıtılmıştır. Burada PB13 kesme pinini, *Rotaty* kesme durumunda kullanılacak fonksiyonu ve CHANGE pinde değişme olduğunda kesmeye girileceğini belirtmektedir.

Öncelikle mikrodenetleyici çalışmaya başladığından bu yana geçen zaman *son_zaman* değişkenine atanır. Sonrasında *sayac* isimli değişkenin değeri bir arttırılır. *zaman_hiz* değişkeninin değeri *son_zaman – ilk_zaman* işlemi yapılarak bulunur ve arada geçen zaman kullanılarak arabanın hızı hesaplanır. En son olarak *ilk_zaman* değişkenine *son_zaman* değişkeninin değeri atanarak fonksiyon sonlandırlır. Bu işlemlerin yapıldığı kodlar aşağıdaki gibidir:

```
void Rotaty() {
son_zaman = micros();
if (son_zaman - ilk_zaman > 5000)
{ sayac = sayac + 1;
son_zaman_hiz = micros();
zaman_hiz = son_zaman_hiz - ilk_zaman;
alinan_yol = 2.56 * sayac ;
olculen_hiz = 25600 / zaman_hiz; }
ilk_zaman = son_zaman; }
```

5. SONUÇ VE ÖNERİLER

Proje süresince MetroCar Sistemlerinin takip ve birleşme algoritmaları konusunda 1:10 oranında küçültülmüş araçlar ve aynı şekilde 1:10 oranında küçültülmüş yollar ile çalışmalar yapılmıştır. Bu çalışmalar 5, 10 ve 15 metrelik yollarda 2 ve 3 araba ile defalarca tekrar edilmiş ve elde edilen sonuçlar neticesinde ortalama olarak yapılan hatalar belirlenmiştir.

Proje süresince tek araçla, iki araçla ve üç araçla gerçekleştirilen denemelerde renk sensöründen alınan veriler ile gerçek veriler arasında ortalama %1'lik bir hata olduğu gözlemlenmiştir. Hesaplanan bu hata gidilen yoldan ve araç sayısından bağımsız olarak ortaya çıkmaktadır. Yani alınan yol arttıkça veya araç sayısı fazlalaştıkça bu hatada bir artış görülmemektedir. 5 metre mesafe ile gerçekleştirilen denemelerde ortalama olarak 5 cm'lik bir hata ile karşılaşılmıştır. Bu hata maksimum olarak 10 cm'e kadar çıkabildiği gibi hatasız denemeler de gerçekleştirilmiştir. Aynı şekilde 10 metrelik denemelerde 10 cm ve 15 metrelik denemelerde 15 cm'lik ortalama hata değeri tespit edilmiştir.

Haberleşme kaynaklı olarak da bazı hataların olduğu proje süresince gözlemlenmiştir. Bu hatalar araç sayısına göre değişiklik göstermektedir. Tek araçla yapılan denemelerde gözlemlenen hata değeri oldukça düşük olduğu için (yaklaşık %0.1) gözardı edilebileceğine karar verilmiştir. Araç sayısının artması ile gözlemlenen hatada bir yükseliş olduğu tespit edilmiştir. 2 araç ile yapılan denemelerde ortalama %0.5'lik bir hata değeri elde edilmiştir. 3 araç ile yapılan denemelerde ise hata değerinin daha da yükseldiği gözlemlenmiştir. Üç araç ile birleşme ve takip algoritmalarında yaklaşık olarak %1'lik bir hata oranı tespit edilmiştir.

Yapılan denemeler neticesinde haberleşme kaynaklı hataların sadece araç sayısına bağlı olmadığı da ortaya koyulmuştur. Alınan mesafenin de haberleşme kaynaklı hatalara neden olduğu düşünülmektedir. 5, 10 ve 15 metrelik denemelerde ortaya çıkmayan bu hata daha uzun mesafeli denemelerde kendisini göstermiştir. Alınan mesafenin artması ile araçtan haberleşme kartına gönderilen mesafe değeri

büyümüş ve haberleşmeyi zorladığı için hata değerlerinde artışlar gözlemlenmiştir. Ancak proje süresince çoğunlukla denenmiş olan 15 metrelik yolda bu hatadan bahsedilmesi doğru olmayacaktır.

Genel olarak bakıldığı zaman elde edilen veriler değerlendirildiğinde projenin başarılı sonuçlar verdiği görülmektedir. MetroCar Sistemlerinin takip ve birleşme algoritmalarının başarılı olarak test edilmesiyle birlikte bu sistemin gerçeklenebilirliğine olan inanç daha da kuvvetlenmiştir.

MetroCar Sistemlerinin günümüz trafik karmaşası ve buna bağlı oluşan büyük zaman kayıpları düşünüldüğünde zaruri bir ihtiyaç olarak ortaya çıktığı da ayrıca görülebilmektedir. İlerleyen süreçte sistemin diğer parçalarının da denenmesi ile birlikte günümüz trafik sistemine entegre edilebilmesi ihtimali de daha da güçlenecektir.

Yapılan bu çalışmada kullanılan araçlar takip algoritmasının test edilmesi konusunda olumlu sonuçlar vermesine karşın birleşme algoritmasının tam olarak test edilebilmesi için ilerleyen süreç içerisinde daha fazla araç kullanılarak denemelerin yapılmasına ihtiyaç duyulmaktadır. Yine bundan sonra yapılacak listesine bakılacak olursa sabit ivmeli hareket konusunda daha fazla çalışma yapılmasına da ihtiyaç bulunmaktadır.

Yapılan çalışmalarda çizgi izleme ve tekerlikten veri alınmasında renk sensörleri kullanılmasına rağmen projenin bundan sonraki aşamalarında bu işlemler için manyetik sensörlerin kullanılmasının daha uygun olacağı düşünülmektedir. Aynı şekilde haberleşme üzerine geliştirme çalışmalarının yapılması gerekmektedir.

MetroCar Sistemlerine bakıldığında sistem için özel tasarlanmış elektrikli araçların olacağından ve bu araçların şarj ihtiyaçlarını sistem üzerinden karşılayacaklarından daha önce bahsedilmiştir. Araçların şarj edilebilmesi için araçlara bir şarj ünitesi eklenmesi ve kullanılacak olan yolların, araçların şarj edilebilmesine uygun şekilde yeniden tasarlanması da MetroCar Sisteminin geleceği için oldukça önemlidir.

6. KAYNAKLAR

Bayzan Ş, “Mobil Sistemlerde Gerçek Zamanlı Trafik Bilgisi Kullanarak Alternatif Araç Güzergâhlarının Belirlenmesi Ve Uygulaması”, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı, (2013)

Bozuyla, M., Tola,AT., Murat,YS., “A Novel Safe Merging Algorithm For Connected Vehicles Using NetLogo”, *Elektronika ir Elektrotechnika,* 24 (3), 3-8, (2018).

CDIAC, “The United States Department of Energy's Carbon Dioxide Information Analysis Center for the United Nations”. (2008)

Chuang TN., Kung JY., “A New Algorithm For the Discrete Fuzzy Shortest Path Problem in a Network”, *Applied Mathematics and Computation*, 174(1), 660-668, (2006)

Forret, A., Konca M., “Autonomous Cars and Society”. (2007)

Gökaşar I., Dündar S ., "Sürücüsüz Taşıtların Trafik Akım Hızına Etkisinin Yapay Sinir Ağları ile İncelenmesi", *Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi*, 56-71, (2018)

Gökozan H., Taştan M., "Akıllı Taşıtlar ve Kontrol Sistemleri". *Mesleki Bilimler Dergisi,* 1-5, (2018)

Güvez H., Dege M., Eren T., “Kırıkkale’de Araç Rotalama Problemi ile Tıbbi Atıkların Toplanması”. *International Journal of Engineering*, 4, 1, 41-45, (2012).

Hernández J., Ossowski S., Garcia-Serrano A., “Multiagent Architectures For Intelligent Traffic Management Systems”, *Transportation Research Part C,* 10(5-6), 473-506, (2002)

Lari A., Douma F., Onyiah I., “Self-Driving Vehicles:Current Status of Autonomous Vehicle Developmentand Minnesota Policy Implications”, Minnesota Üniversitesi, (2014)

Oliveira E. ve Duarte N., “A Multi-Agent System for Simulation of Traffic Control”, *Proceedings of the ESM* 2005 (European Simulation and Modelling), 128 – 135, (2005)

Ozguner, U., Acatman, T., Redmil, “Autonomous Ground Vehicles”, (2011)

Paruchuri P., Pullalarevu A.R., Karlapalem K., “Multi Agent Simulation of Unorganized Traffic”, *International Conference on Autonomous Agents*,176 – 183, (2001)

Zohdy I., Rakha H, “Optimizing Driverless Vehicles At Intersections”, *19th ITS World Congress, Vienna, Austria,* (2012)

WEB_1, https://ontrava.com

WEB_2, https://www.endustri40.com

WEB_3, https://www.bilgiustam.com

WEB_4, https://www.elektrikport.com

WEB_5, http://www.elektrikrehberiniz.com

WEB_6, http://www.rcturka.com

WEB_7, http://www.mcu-turkey.com

WEB_8, http://otomasyondergisi.com.tr

WEB_9, https://gelecegiyazanlar.turkcell.com.tr

7. ÖZGEÇMİŞ

Adı Soyadı	:Erkan Akın
Doğum Yeri ve Tarihi	:Denizli – 07.11.1991
Lisans Üniversite	:Pamukkale Üniversitesi
Elektronik posta	:eakin101@posta.pau.edu.tr
İletişim Adresi	:İncilipınar Mahallesi İncilipınar Caddesi No:2/B Tarım İl Müdürlüğü Lojmanları B Blok Daire:6 Pamukkale/DENİZLİ
Yayın Listesi	**:**
Konferans listesi	**:**