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

METROCAR ARAÇLARIN SABİT İVMELİ HAREKETİ

YÜKSEK LİSANS TEZİ

HÜSEYİN EMİK

DENİZLİ, AĞUSTOS - 2021

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

METROCAR ARAÇLARIN SABİT İVMELİ HAREKETİ

YÜKSEK LİSANS TEZİ

HÜSEYİN EMİK

DENİZLİ, AĞUSTOS - 2021

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.

HÜSEYİN EMİK

ÖZET

METROCAR ARAÇLARIN SABİT İVMELİ HAREKETİ
YÜKSEK LİSANS TEZİ
HÜSEYİN EMİK
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

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

DENİZLİ, TEMMUZ - 2021

Trafikte artan araç sayısı ile birlikte kaynağı sürücü olan kazalar da bir hayli artmıştır. Sürücünün dikkatsiz davranması, trafik kurallarına uymaması ve kuralları dikkate almamasından kaynaklı kazalar hem can kayıplarına hem de maddi hasarlara neden olmuştur. Günümüzde ilerleyen teknoloji ile birlikte can ve mal kayıplarının önüne geçmek adına uzaktan kontrol edilebilen araçlara veya sürücüsüz araçlara ilgi de bu nedenle artmıştır. Sürücüsüz araçlar ile birlikte yaşanan olumsuzluklar en aza indirgenecek aynı zamanda kişilere zaman yönünden büyük avantaj sağlayacaktır. Yapılan tez çalışmasında tek şeritli özel bir yolda ister araca binen kişi tarafından isterse de bir operatör tarafından hızlanma ve yavaşlama ivmesi belirlenerek, sabit ivme ile hızlanması veya sabit ivme ile yavaşlaması sağlanmıştır. Bu özel yolda araç ilerlerken kişi araç içerisinde sosyal faaliyetlerini ya da fiziksel ihtiyaçlarını çok kolay bir şekilde gerçekleştirebilir. Yapmış olduğumuz bu çalışmada 1/10 boyutunda bir RC model araç tasarlanmış, bu araca uygun yazılım geliştirilmiş ve araç özel bir yolda denenmiştir.

ANAHTAR KELİMELER: Metrocar Araçlar, Otonom Araçlar, Sabit İvme, İvmelİ Hızlanma ve İvmeli Yavaşlama

ABSTRACT

FIXED ACCELERATION OF METROCAR VEHICLES
MSC THESIS
HÜSEYİN EMİK
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

(SUPERVISOR:PROF.DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, JULY 2021

Along with the increasing number of vehicles in traffic, the accidents whose source is the driver have also increased considerably. The driver's inattention, non-compliance with traffic rules and ignoring the rules caused both dies and material damages. Today, with advanced technology, the interest in vehicles that can be remote controlled to prevent loss of life and property or driverless vehicles has increased. Disadvantages of self-driving vehicles will be minimize, also it will be provide great advantage to people. In the thesis study, the acceleration and deceleration were determined by either the person riding the vehicle or an operator on a single lane special road, and it was accelerated with constant acceleration or decelerated with constant acceleration. While the vehicle is moving on this special road, the person can easily perform her social activities or physical need in the vehicle. In this study we have done, an RC model vehicle has been designed with a size of 1/10, software suitable for this vehicle has been developed and the vehicle has been tested on a special road.

KEYWORDS: Metrocar Vehıcles, Autonomous Vehicle, Accelerated Acceleration and Accelerated Deceleration, Fixed Acceleration

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

Ω	**:**	Ohm
τ	**:**	Zaman Aralığı
Zn	**:**	Çinko
C	**:**	Karbon
Ni-Cad	**:**	Nikel-Kadmiyum
Li-İon	**:**	Lityum-İyon
Pb	**:**	Kurşun

KISALTMALAR LİSTESİ

DC	**:**	Doğru Akım
RC	**:**	Radyo Kontrol
ESC	**:**	Elektronik Hız Kontrol
CNC	**:**	Bilgisayar Sayısal Kontrol
PWM	**:**	Darbe Genişliği Modülasyonu
STEP	:	Adım
VCC	**:**	Kollektör Besleme Gerilimi
GND	**:**	Toprak
NPN	**:**	Negatif Pozitif Negatif
PNP	**:**	Pozitif Negatif Pozitif
ARM	**:**	İşlemci
STM	**:**	STMikroelektronik
AVR	**:**	Atmel VR
CPU	**:**	Merkezi İşlem Birimi
EEPROM	**:**	Elektronik Olarak Silinebilir Programlanabilir Salt Okunur Bellek
SRAM	**:**	Durağan Rastgele Erişimli Bellek
SDRAM	**:**	Eş Zamanlı Dinamik Rastgele Erişimli Bellek
RAM	**:**	Rastgele Erişimli Hafıza
ADC	**:**	Analog Dijital Konvertör
DAC	**:**	Dijital Analog Konvertör
OPAMP	**:**	Operasyonel Amplifikatör
DMA	**:**	Doğrudan Bellek Erişimi
CAN	**:**	Denetleyici Alan Ağı
SPI	**:**	Seri Çevre Arayüzü
I2C	**:**	Entegre Devre
USART	**:**	Evrensel Asenkron Alıcı / Verici
RTC	**:**	Gerçek Zamanlı Saat
OTP	**:**	Tek Seferlik Parola
MAC	**:**	Medya Erişim Kontrolü
LCD	**:**	Sıvı Kristal Ekran
FPU	**:**	Kayan Nokta Birimi
DSP	**:**	Dijital Sinyal İşlemcisi
DSC	**:**	Dijital Ses Kod Çözücü
PGA	**:**	Programlanabilir Kazanç Amplifikatörü
TFT	**:**	İnce Film Transistör
ONEWİRE	**:**	Tek Hatlı İletişim
INPUT	**:**	Giriş
OUTPUT	**:**	Çıkış
OPTICAL	**:**	Optik
MAGNETIC	**:**	Manyetik
THERMAL	**:**	Termal
VOLTAGE	**:**	Voltaj
CURRENT	**:**	Akım
MASTER	**:**	Kontrolcü
SLAVE	**:**	Kontrol Edilen

USB	**:**	Evrensel Seri Veriyolu
USB OTG	**:**	Hareket Halinde Evrensel Seri Veri Yolu
Wİ-Fİ	**:**	Kablosuz Bağlantı Alanı
SWD	**:**	Seri Kablo Hata Ayıklama
GHz	**:**	GigaHertz
KHz	**:**	KiloHertz
Hz	**:**	Hertz
KBps	**:**	Saniyede Kilobit
MBps	**:**	Saniyede Megabit
KB	**:**	Kilo Byte
MB	**:**	Mega Byte
IO	**:**	Giriş/Çıkış
PID	**:**	Oransal İntegral Türev
Kp	**:**	PID Parametreleri
Ki	**:**	PID Parametreleri
Kd	**:**	PID Parametreleri
aa	**:**	Hızlanma İvmesi
ad	**:**	Yavaşlama İvmesi
to	**:**	Zaman Sabiti
k	**:**	k Sabiti, Bir Siyah-Beyazın Yol Karşılığı
CLOCK	**:**	Saat
GPIO	**:**	Genel Amaçlı Giriş/Çıkış
NVIC	**:**	İç İçe Vektörlü Kesme Denetleyicisi
TIMER	**:**	Zamanlayıcı
PLL	**:**	Faz Kilitlemeli Çevrim
HSE	**:**	Yüksek Hızlı Harici
HCLK	**:**	İşlemci Saati
CE	**:**	Çip Etkinleştirme
CSN	**:**	Çip Seçme
PCB	**:**	Baskı Devre Kartı
CS	**:**	Çip Seçim
EN	**:**	Etkinleştirme
INA	**:**	Giriş A
INB	**:**	Giriş B
TIM1	**:**	Zamanlayıcı1
TIM2	**:**	Zamanlayıcı2
TIM3	**:**	Zamanlayıcı3
TIM6	**:**	Zamanlayıcı6
ARR	**:**	Otomatik Yükleme Sayacı
PSC	**:**	Ön Bölücü
APB1	**:**	Gelişmiş Çevresel Veri Yolu 1
APB2	**:**	Gelişmiş Çevresel Veri Yolu 2

ÖNSÖZ

Bu çalışmanın yürütülmesi sırasında desteğini, bilgi birikimini esirgemeyen lisans hocam, danışmanım Prof. Dr. Abdullah Tahsin Tola'ya, tüm iş yoğunluğuna rağmen bana zaman zaman vakit ayıran sıra arkadaşım Gökhan Tokmak'a, moral ve motivasyon anlamında yanımda olan değerli eşim Zinet Emik'e ve annem Fadime Emik'e, çalışmalarım sırasında ümit veren Arş. Gör. Engin Doğan'a teşekkür ederim.

1. GİRİŞ

1.1 Tezin Literatür Taraması

1939 yılında Genaral Motors tarafından otonom araçlar üzerine yapılan Futurama Dünya Fuarı çalışması bu alanda yapılan çalışmaların başlangıcı saylmaktadır. Genaral Motors ortaya çıkardığı bu araç sayesinde insanları şehir içinde güvenli ve otonom olarak taşıyan bir araç meydana getirmiştir (Özgüner 2011).

1953 yılında labaratuvar zeminde döşenen kablolarla kontrol edilen ve yön verilen minyatür araç RCA Labs tarafından üretildi (WEB_1).

1980 yılında Münih kentindeki Bundeswehr Üniversitesinde bir Mercedes Benz minibüs tasarlandı. Bu minibüs Ernst Dickmanns ve ekibi tarafından tasarlanıp trafiğe kapalı bir alanda 63km/h hıza ulaşmıştır. (WEB_1).

1987 yılında Darpa'nın desteklemiş olduğu otonom kara aracı Amerikada ilk kez tamamen sensör bazlı bir arazi ortamında sürüşünü gerçekleştirmiştir. Bu araçla saatte 30km hız yapılırken kamera görüntüleri, robot kontrol mekanizmaları ve lazer radarı kullanılmıştır (WEB_2).

1991 yılında Amerika Birleşik Devletleri Kongresi, USDOT'a "1997 yılına kadar otomatik bir araç ve otoyol sistemi gösterme" emrini veren ISTEA ulaşım yetki belgesini kabul etti (WEB_1).

1995 yılında NAVLAB isimli projesiyle Carnegie Mellon Üniversitesi, No Hands Across America ile 5000 km'yi %98.2 başarı oranı ile tamamlamıştır. Yine 1995 yılında Münih'ten Kopenhag'a 1600 km'lik mesafeyi S-segmenti bir Mercedes Benz ile gidip dönmeyi tamamlamıştır. Araç %95'lik bir otonom verimliliğini sağlarken aynı zamanda 175km/h'lık bir hıza ulaşmıştır (WEB_2).

2001 yılında düzensiz trafik için çoklu etmenli similasyonu Paruchuri ve arkadaşları gerçekleştirmiştir. Bu similasyonun asıl amacı trafikte kendilerini yöneten bir sistem olmadan (trafik polisi, trafik işaretleri gibi) trafiğin akışını sağlamaktır (Paruchuri 2001).

2001 yılında Wahle ve arkadaşları yürüttükleri çalışmada, kullanıcıya uygun yol güzergahlarını bulmak, trafikte oluşan sıkışıklığı minimuma indirmek ve trafik alt yapısını verimli kullanabilmek için sürücü bilgi sistemini (ATIS) geliştirmişlerdir. Bu çalışmada seyahat süresini ve trafik yoğunluğunu hesaplamak amacıyla kullanılan online simülasyonların gerçek trafik bilgileri ile desteklenmesi önerilmiş, elde edilen verilerin kullanıcıların bireysel tercihleri ile uyumlu bir yol haritası olarak sunulması gerektiği vurgulanmıştır. (Wahle 2001).

Çok kriterli optimizasyon problemini çözmek için; dinamik güzergah problemine önerilen uygulama fuzzy set teorisi olmuşur. Belman ve Zadeh çok kriterli bu optimizasyon probleminin simetrik karar modeli kullanılarak çözülebileceğini belirtmişlerdir. Lineer maliyet fonksiyonlu modeli ile fuzzy güzergah modelin karşılaştırılması yapılmıştır. Bulanık küme teorisini çok kriterli optimizasyon problemini çözmek için dinamik güzergah belirleme sorununa uygun görmüşlerdir. Bu sistem ile gelecekte trafiğin durumuna göre internet üzerinden en uygun yolu belirleme imkanı sağlayacağını belirtmiştir (Wahle 2001).

2002 yılında otonom araçlar için ABD Savunma Bakanlığının Grande Challenge yarışında hiçbir grup final çizgisine varamazken 2005 yılında 5 takım final çizgisine ulaşarak 217 km rotayı tamamlamıştır. Bu başarının ödülü olarak Stanford Üniversitesini temsil eden takım 2 milyon doları kazanmıştır. 2007 yılında yarışmayı tamamlayan 6 sürücüsüz araç 3,5 milyon dolara çıkarılan ödülü aralarında paylaşmıştır Bu noktada gelinmiş en büyük başarılardan biri de Parma Üniversitesinden Alberto Broggi'nin yaptığı çalışmadır. Broggi İtalyadan 13 bin km'deki Çin'e 4 adet sürücüsüz elektrik arabasını 2010 yılında yapılan Şangay Expo'ya yetiştirerek kıtalar arası gönderimini başarıyla tamamlamıştır (Fornet 2007).

Akıllı trafik sistemleri için ajan tabanlı mimariler sunan diğer bir çalışma da Hernandes tarafından 2002 yılında Barcelona çevresinde kentsel otoyol ağındaki gerçek zamanlı trafik yönetimi için karar vermeyi sağlayan çok ajanlı bir sistem

önerilmiştir. Yerel trafik sorunlarını çözebilmek için bu iki sistem benzer bilgilere dayalı akıl yürütme tekniklerini kullanmaktadır. Bu çoklu ajan tabanlı mimarinin akıllı trafik sistemleri, avantajları ve dezavantajları, uygulanabilirlik açısından incelenmiştir. Bu inceleme sonucunda trafik yönetimini geliştirmek için yapay zeka tekniklerinin kullanımı, önemli bir katkı sağlamıştır (Hernandes 2002).

Oliveira ve Duarte tarafından 2005 yılında yapılan çalışmada trafiğin temelini oluşturan araçlar nesne, kavşaklar, trafik ışıkları etmen olarak belirtilmiştir (Oliveira 2005).

Başkaya ve Öztürk 2005 yılında bir ekmek fabrikasının ekmek dağıtım sorununu dal kesme yöntemiyle çözmeye çalışmışlardır (Başkaya 2005).

2006 yılında Chong ve arkadaşları akıllı trafik sistemi konulu bir çalışmada trafik sorununu çözmek için, karar destek sistemlerinin ve bilgi analiz sistemlerinin eksik olması nedeni ile veri madenciliği ve grid bilgi işlemeye dayalı yeni analiz sistemlerini tavsiye etmişlerdir. Yararlı bilgileri tanımlamak bu sistemde kolay olmaktadır. Bilgi analizi ve karar desteğini zeki yapacağının savunulmuş olmasının sebebi, bu sistemin yararlı bilgilere dayanarak tavsiyelerde bulunmasıdır. Çok fazla trafik verisi kullanılarak yapılan analiz sonucunda, tavsiye edilen akıllı ulaşım sisteminin vereceği kararın trafik yönetimi için daha uygun olduğu ve sistemin trafik sinyal kontrolü, komut yönetimi, trafik yönlendirmesi, trafik bilgi servisi ve olay tespiti açısından verim artırabileceği sonucuna varılmıştır (Chonge 2006).

Güzergahın en uygun parametrelerin seçildiği rasyonel güzergah araştırma yöntemleri, 2006 yılında Jakimavicius ve Macerinskiene tarafından yapılan araçlar için rasyonel güzergahların GIS tabanlı modellemesi çalışmasında sunulmaktadır. Sunulan çalışmada alternatif güzergahın önerilmesi en hızlı güzergahı ve en kısa güzergahı hesaplaması aktarılmıştır. Elde edilen analizlere bakılarak, ulaşım akış modellemesi için GIS araçları toplanmakta ve yol trafik yoğunluğunun gün içerisinde saatlere bağlı olarak değerlendirilmesi ile güzergah araştırma algoritması ortaya konulmuştur. Ortaya konulan bu yöntem Vinius şehrinin ulaşım alanının bir kısmında gerçekten uygulanabilir bir örneğe dayanmaktadır (Jakimavicius 2006).

2008 yılında Du ve Gao yoldaki işaret çizgisini takip etme işlemini AGU'nin önünde bir CDD kamera kullanarak başarmışlardır. Bu çalışmayı tasarlarken tekerleklerde de kodlayıcı kullanarak, AGU'nin aldığı mesafeyi ölçmüşlerdir. Sistemin pozisyon hatalarından kaçınabildiğini yol takibinde görüntü işleme tekniklerini kullanarak gözlemlemişlerdir (Du 2008).

Keshavars ve Khorram tarafından 2009 yılında yapılan yüksek güvenirliliğe sahip bulanık bir en kısa yol çalışmasında kesin olmayan sayılardan oluşan bir ağ üzerindeki en kısa yol sorununa odaklanılmıştır. Bulanık en kısa yol problemi karışık bir doğrusal tam sayı olmayan programlama problemi gibi önerilen maksimum ve minimum kriterler doğrultusunda ele alınmıştır. İki seviyeli programlama problemini çözmek için parametrik en kısa yol problemine dayanan algoritma sunulurken, kolay çözülebilir iki seviyeli programlama problemine indirgenebildiği gösterilmiştir (Keshavars 2009).

Aynı başlangıç ve varış noktalarına rağmen sürücüleri farklı sebeplere ve bu sebeplerin önem derecesine bağlı olarak güzergah tercihlerinde bulundukları için genellikle aynı yoldan gitmediklerini ifade eden çalışmada güzergah kılavuz sistemlerinin farklı sürücüler için yolları sürücülere daha iyi anlatan güzergahlar seçim kılavuzları için sürücülerin gösterdiği güzergah seçimindeki davranışlarını dikkate alan bir model yaklaşımı 2009 yılında Lin ve arkadaşları tarafından bulunmuştur. Bulanık nöral kılavuz sistemi ismindeki bu yaklaşımı uygulayabilmek adına sürücülerin geçmiş sürüş kayıtlarından elde edilen anlaşılması zor tutumları ile karar mantıklarından faydalanılmıştır. Sistemin kullanıcı müdahalesini önleyerek kendi kendisini ayarlayabilmesi amacıyla ANFIS kullanılmıştır. Farklı tercilere dayanarak farklı en iyi güzergahları otomatik olarak sunan, sürücülere uyarlanabilen bir sürüş güzergahı kılavuzu bu yaklaşımla sunulmuştur (Lin 2009).

Genetik algoritma tabanlı yol plan algoritmasını 2009 yılında Taha ve arkadaşlarının yaptığı farklı bir çalışmada AGV ve Robot Navigasyon Waypoint'lerini kullanarak uygulamaya çalışmıştır. Genetik algoritma problemi NP-Hart ve Multi Objektive uygulamada robot navigasyonu için önerilmiştir. Waypoint navigasyon sisteminin avantajı çevre hakkında kapsamlı bilgiye ihtiyaç duymaması ve kullanılan geleneksel navigasyon sistemlerinin yerine kullanılabilmesidir. Waypoint sayısının çok olduğu sistemlerde önerilen bu algoritmanın kromozom satır

sayısını azalttığı, daha hızlı yakınsadığı ve yakın optimum sonuç getirdiği ispatlanmıştır (Taha 2009).

Fuzzy kontrol tekniği uygulayarak çoklu AGV'nin bilinmeyen bir ortamda hybird kullanarak istenen hedefe ulaşması 2010 yılında Khanmohammadi ve Mirnia tarafından yapılan bir çalışmayla sağlanmıştır. Hybird tabanlı metod fuzzy tabanlı metod kullanılarak hareketli engelleri tanımlama hariç tesis edilen yolu robotun bulması, merkezi kontrollerde karmaşık hesaplamaları indirgemiş ve robotta yeteri kadar hafıza olduğu düşüldüğünde yerel plan problemlerini ve AGU'nin istenen hedefe doğru giderken farklı şekil ve pozisyondaki engelleri aşmasını sağlamışlardır (Khanmohammadi 2010).

2010 yılı ile birlikte önde gelen birçok araç firması sürücüsüz araç sistemlerini test etmeye başlamışlardır (WEB_1).

2010 yılında büyük bir firma olan GOOGLE kaza sayılarını yarıya indirecek bir çözüm bulma amacı ile bu sektöre girdiğini ve gizli gizli bu çalışmaları yürüttüğünü belirtmiştir (WEB_1).

2010 yılında AUDİ, yarış hızlarına yakın hıza çıkabilen sürücüsüz bir Audi TTS modeli aracı Pike's Peak'in zirvesine gönderdi (WEB_1).

2011 yılında General Motors'un otonom aracı olan EN-V'yi geliştirdi (WEB_1).

2012 yılında Volkswagen 130 km/h hıza kadar sürüş izni olan "Geçici Otomatik Pilot" (TAP) sistemini test etmeye başladı (WEB_1).

2012 yılında Ford firması sürücüsüz araçlar ve iletişim sistemleri alanında geniş kapsamlı araştırmalara parmak basmıştır (WEB_1).

Işıksız kontrollü bir kavşakta dört ayrı sürücüsüz arabanın Monte Karlo yöntemiyle bekleme sürelerini düşüren çok ajanlı bir çalışmayı Zohdy ve Rakha 2012 yılında yapmıştır (Zohdy 2012).

Başarılı sonuçlar elde eden Bayzan, 2013 yılında yapılan tez çalışmasında araç konumunun belirlenmesini GPS ve Merkezi Veritabanı ile yapmıştır (Bayzan 2013).

2013 yılının başlarında Toyota iletişim sistemlerine sahip ve çok sayıda sensörü bulunan ayrıca kısmen kendi kendini sürebilen aracını tanıttı (WEB_1).

2013 yılının sonlarına doğru Nissan firması 2000'li yıllarda birçok otonom aracın üretimine geçeceklerini açıkladı (WEB_1).

Otonom araçların gelişimi ve mevcut durumları Lari ve arkadaşları tarafından 2014 yılında raporlanmıştır (Lari 2014).

2014'ün sonlarına gelindiğinde Tesla Motors ilk Auto Pilot sürümünü açıkladı. Bu sistemde model S araçları, şerit kontrolü, hız limiti, çeşitli görüntü işleme algortimaları, fren kontrolü ve otomatik direksiyon yapabildiğini açıkladı. Bu sistemde aynı zamanda otomatik park olduğunu belirtti (WEB_1).

2015 yılında yazılımı sürekli güncelleyen Tesla Motors Sanfrancisco ile Seattle arasındaki otoyolu bir sürücü ile Auto Pilot sistemini neredeyse hiç yardımsız şekilde test etti. (WEB_1).

Şubat 2015'te UBER otonom araç sektörüne girmek istediğini ve filo oluşturmak istediğini açıkladı. UBER Carnagie Mellon Üniversitesinden bir grup araştırmacı kiralayarak sektöre giriş yaptı ve filosuna 20 adet Ford Fucions araç kazandırdı. UBER'in araçlarından 1 GPS, Radar, Lidar ve 20 kamera mevcuttu. (WEB_1).

2016 yılının Mayıs ayında Florida'daki Wiliston kentinde Model-S aracı Auto Pilot modunda iken önüne kontrolsüz bir traktör çıkarak fren sisteminin başarısız olması sonucunda Tesla Motors'un Model-S aracındaki kişi kurtarılamamıştır (WEB_1).

2017 yılında Audi, yeni A18'in saatte 60 km hızla tamamen kendi gidebilecek bir araç olduğunu belirtti. Audi kamera ve ultrasonik sensörlere ek olarak 3-D Lidar sistemi kullanan ilk üretici olacaktır (WEB_1).

2. TEMEL KAVRAMLAR

2.1 Giriş

Bu kısımda projede kullandığımız malzemeler hakkında temel bilgilere değinilecektir. Bunlardan bazılarına aşağıda yer verilmiştir.

- Doğru akım motorları ve sürme teknikleri,
- Servo motorlar ve sürme teknikleri,
- Batarya çeşitleri,
- Mikrodenetleyiciler,
- Sensörler.

2.2 Doğru Akım Motorları

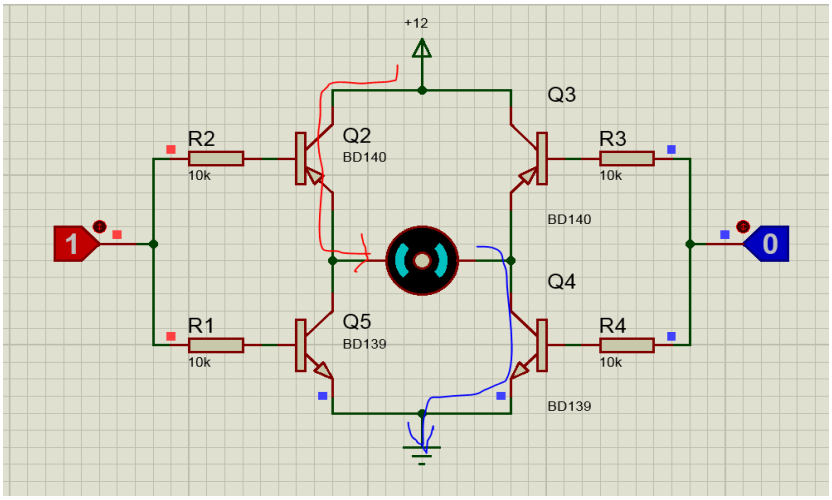
DC motorlar doğru akım elektrik enerjisinin mekanik enerjiye dönüşmüş halidir.

2.2.1 Doğru Akım Motorlarının Çalışma Prensibi

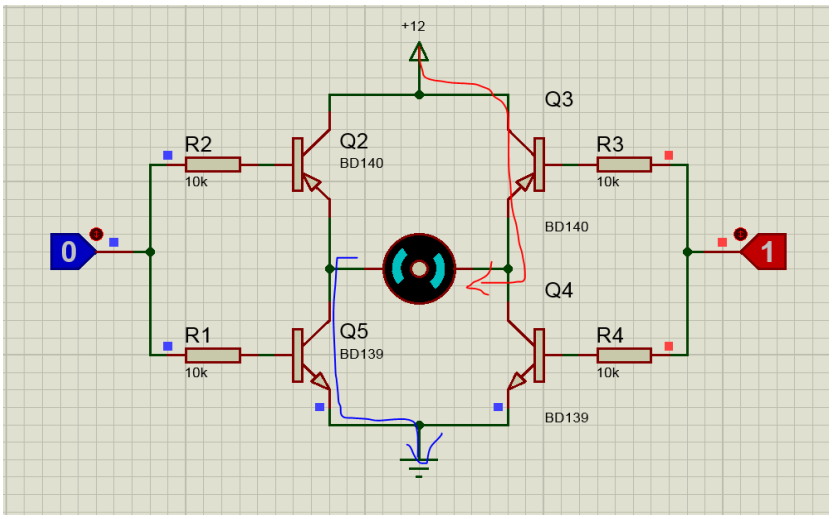
DC motorun hareket etme prensibi; motor içinde bulunan sargılara elektrik akımı uygulandığında sabit mıknatıslara zıt yönde oluşan elektrik manyetik kuvvetin etkisiyle gerçekleşir. Sargılara uygulanan akımın yönü sabit mıknatıslara ters manyetik alan sağlayacak şekilde değiştirilmesi gerekir. Bu değişim fırçalı motorlarda motorun sargılarına temas eden fırçalar tarafından, fırçasız motorlarda ise hız kontrol devresi tarafından sağlanır.

2.2.2 Fırçalı Doğru Akım Motorları

En yaygın ve en sık kullanım alanına sahip DC motor tipi, fırçalı DC motordur. Birçok kullanım alanı vardır. Bunlardan bazıları; şarjlı el matkapları, oyuncak arabalar vb. Fırçalı DC motorun mili üzerinde bobinleri vardır. Motor gövdesinin iç kısmında ise güçlü mıknatıslar yer alır. Mil üzerindeki bobinlere fırçalar vasıtası ile elektrik akımı uygulanır. Milin hareket etmesini sağlamak için uygulanan elektrik akımının bobinlerde oluşturduğu manyetik alan, mıknatısların manyetik alanları ile sürekli çakışacak şekilde ayarlanır ve mil hareket etmiş olur. Fırçalı DC motorların hız kontrollerini sağlamak için iki terminali (+ -) arasındaki gerilim değiştirilir. Yani gerilimi artırılarak motor hızı artırılır, gerilim azaltılarak motor hızı yavaşlatılır. Eğer motor yönünü değiştirmek istersek uygulanan gerilimin yönünü de değiştirmek gerekecektir. Bu işlem için H-köprüsü denilen devrelere ihtiyaç vardır. Şekil 2.1 ve Şekil 2.2'de H – Köprü devresi ile DC motorun sola ve sağa döndürülmesi verilmiştir.



Şekil 2.1: H - Köprü Devresi ile Sola Döndürme



Şekil 2.2: H - Köprü Devresi ile Sağa Döndürme

Motora uygulanan gerilimin büyüklüğü ve yönü değiştirilerek kolay bir şekilde hız ve yön kontrolü yapılabilmesi, fırçalı DC motorların en büyük avantajıdır. Diğer avantajları ise basit sürücü devresi ile sürülmesi, kullanım kolaylığı ve uygun fiyatta piyasada bulunabilmeleridir. Avantajları olduğu gibi sürekli şafta sürtünen fırçaların aşınması da fırçalı motorların en büyük dezavantajıdır. Bu olay esnasında sürtünmeden kaynaklı ısı meydana çıkacaktır, bu ısı da fırçalı DC motorların verimini düşürecektir. Bu olay da fırçalı DC motorlarda bakım gereksinimlerini doğuracaktır. Şekil 2.3'de Fırçalı DC Motor görülmektedir.

Şekil 2.3: Fırçalı Doğru Akım Motoru

2.2.3 Fırçasız Doğru Akım Motorları

Fırçasız doğru akım motorlarının yapısı, fırçalı doğru akım motorlarının tam tersi şekilde düşünülebilir. Burada sargılar sabit dururken motor milinde mıknatıs bulunur.

Fırçasız doğru akım motorlarının bağlantı terminallerinde üç adet kablo bulunur. Bu kablolar motor içinde farklı fazlara bağlıdır. Fazlara farklı sıra ile elektrik akımı verilirse motor içindeki mıknatıslarda zıt elektro manyetik alan oluşur ve bu durumda motor dönmüş olur. Fırçalı doğru akım motorları ile kıyaslandığında sürtünmeye bağlı verim kaybı ve bakım onarım gerektiren parça sayısı daha azdır. Bunun nedeni ise, bobinlere uygulanan gerilim için aşınan bir parça barındırmamasıdır.

Fırçasız doğru akım motorları yüksek performans ve yüksek verimleri ve bakım kolaylığı sebebi ile dronelerde ve RC (Radio Control) model araçlarda sık kullanılır.

Bu motorların tek büyük dez avantajı maliyetli olması ve doğrudan gerilim uygulanarak kullanılamamasıdır. Fırçasız doğru akım motorlarını çalıştırabilmek için mutlaka sürücü devreye ihtiyaç vardır. Bu sürücü devreyi modellendiren kişiler bu sürücü devresine ESC (Electroic Speed Control) adınıvermişlerdir. Şekil 2.4'de Fırçasız DC Motor görülmektedir.

Şekil 2.4: Fırçasız Doğru Akım Motoru

2.2.4 Redüktörlü Doğru Akım Motorları

Redüktör kelimesi, hareketli parçaların bir arada bulunmasını sağlayan ve motor gövdesinin içine yerleştirilmiş millerden oluşan yapıdır. Yani redüktörlü motor demek bu motorun daha işlevsel ve daha kabiliyetli olması anlamına gelmektedir. Redüktörlü motorların tercih edilmesinin sebebi, hareket ve güç iletimine imkan vermesidir. Bu motorlar az güçle yüksek moment üretirler.

Redüktörlü motorlar daha çok otomotiv sanayinde, pres atölyelerinde, yüksek basınç gerektiren alanlarda, paketleme tesislerinde, eşya taşıma şirketlerinin eşyalarını taşıma mekanizmalarında tercih edilirler. Şekil 2.5'de Redüktörlü DC Motor görülmektedir.

Şekil 2.5: Redüktörlü Doğru Akım Motoru

2.2.5 Servo Motorlar

Servo motorların tasarlanma amacı istenilen pozisyonu alması ve başka bir komut gelene kadar bu pozisyonda kalmasıdır. Servo motorlar ilk

olarak uzaktan kumandalı araçlarda kullanılmış olsa da robot teknolojilerinde de sıklıkla görülmektedir.

2.2.5.1 Servo Motorların Çalışma Prensibi

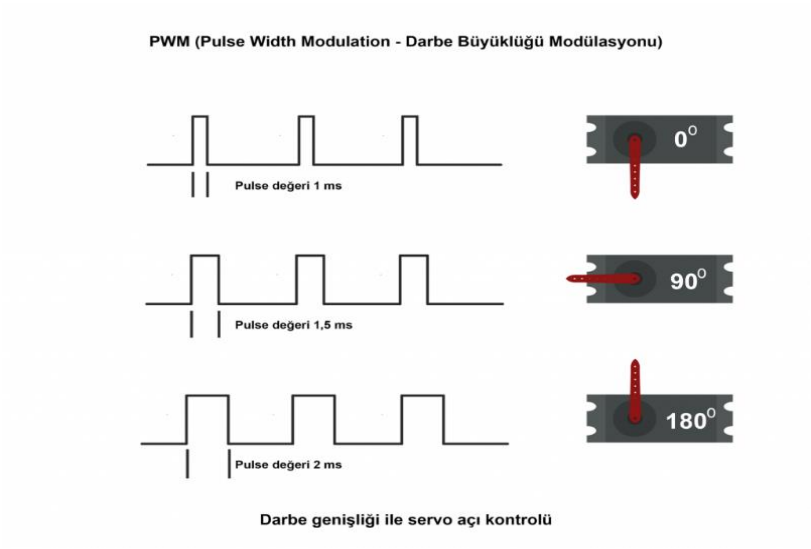
Servo motor içerisinde bir DC motor bulunur ve bu motor sayesinde servo motorlar hareket ederler. Bu motor ile birlikte bir dişli mekanizması, bir motor sürücü devresi ve potansiyometre bulunmaktadır. Buradaki potansiyometre dönüş miktarını ölçmektedir. Genellikle çalışma açıları 180 derecedir. Fakat ihtiyaç durumuna göre 360 derece açı yapabilen servo motorlar da piyasada mevcuttur.

Servo motorlar ilk olarak uzaktan kumandalı model araçlarda kullanılmıştır. Aynı zamanda uçağın kanat açısını ayarlamada kullanılmıştır.

Servo motorların avantajı, hassas pozisyon kontrolü sağlamaları ve sürücü devreye ihtiyaç duymamasıdır. Dezavantajı ise, kısıtlı hareket imkanı sunmasıdır. Şekil 2.6'da Servo Motor görülmektedir.

Şekil 2.6: Servo Motor

Çalışma gerilimleri genellikle 4,8-6 volttur. Bu motorları çalıştırmak için PWM (Palse Width Modulation-Sinyal Genişlik Modülasyonu) sinyaline ihtiyaç vardır. Servo motorlar her 20ms içerisinde bir pals değeri okumaktadır. Pals uzunluğu motorun yaptığı açıyı belirlemektedir. Örneğin; 1,5ms servo motor için 90 dereceye karşılık gelmektedir. Genellikle minimum pals genişliği 1ms, maksimum pals genişliği 2ms'dir. Şekil 2.7'de Servo Motorun açı kontrolü gösterilmektedir.



Şekil 2.7: Servo Motor Açı Kontrolü

2.2.6 Adım Motorları

Step motorları hassas konum ayarı imkanı sağlayan ve fırçalı DC ya da fırçasız DC motorlar gibi sürülebilen motorlardır. En çok CNC tezgahları ve üç boyutlu yazıcılarda kullanılır. Step motorların avantajları şu şekilde sıralanır;

- Çok hassas pozisyon ayarı,
- Hız kontrol ayarı,
- Düşük devirde yüksek tork üretmeleri,

Dezavantajları ise;

- Geri bildirim mekanizması bulundurmadığı için harici konum limitlemesine ihtiyaç duyulmasıdır.

Şekil 2.8'de Adım Motoru gösterilmektedir.

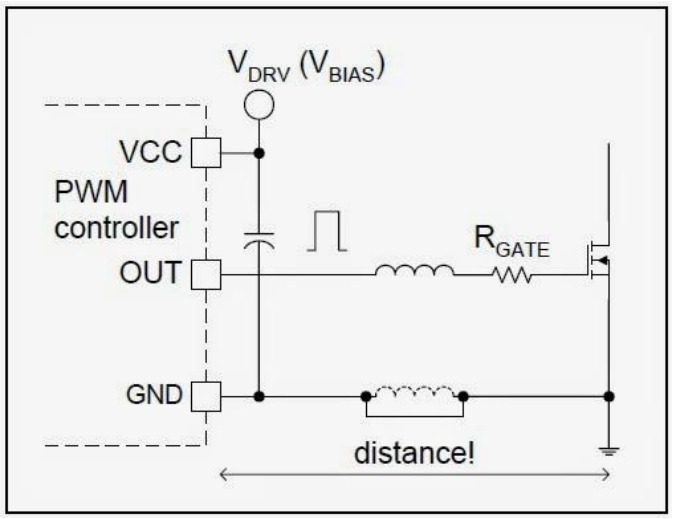
Şekil 2.8: Adım Motoru

2.3 Doğru Akım Motor Sürücüleri

DC motorların terminalleri arasındaki gerilimi tersine çevirmeye yarayan birleşik devrelere motor sürücüleri denir. Motor sürücülerin yapımında birçok elektronik eleman kullanılabilir. Bunlarda başlıcaları; transistör, role, ve anahtarlama görevi yapan bir çok lojik entegre kullanılabilir. Fakat burada önemli olan, motor sürücünün, devrede kullanılacak olan DC motora uygun, akıma dayanıklı ve aşırı ısınmayan cinsten olmalıdır. Endüstride kullanılan motorlar için birçok motor sürücü devreleri vardır.

2.3.1 PWM ile Direkt Sürme

Yer ve fiyat açısından tasarruf etmek istenilen durumlarda tek çıkıştan tek MOSFET ile direkt sürme tekniği kullanılabilir. Şekil 2.9'da Direkt Sürme Devresi gösterilmektedir.

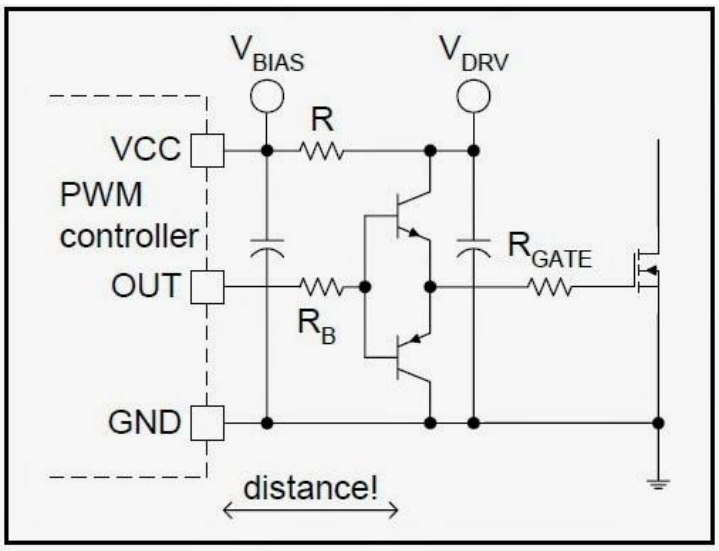


Şekil 2.9: Direkt Sürme Devresi

Burada belirtilen GND ucu ile *MOSFET*'in source ucu arasındaki uzaklık parazitik endüktansın değerinde önemli bir faktöre sahiptir. Bu parazitik endüktans OUT terminalinden GATE terminaline ve oradan da GND terminaline gelen döngünün içinde yer alır. GATE terminali ile GND terminali arasındaki uzaklık anahtarlama hızını düşürerek GATE terminalindeki sinyale istenmeyen osilasyonlara sebebiyet verebilir. Bu endüktansı düşürmek için PCB devrede yollar daha geniş tercih edilmelidir. Diğer bir problem ise sinyal genişlik modülasyon (PWM) entegresinin akım kapasitenin düşük olabilmesidir. Bu da GATE terminalinin taşıdığı akımı kaldıramayacağı anlamına gelebilir. Ayrıca bu akımın yüksek olması durumunda entegre içinde harcanan güç de yüksek olur. Bu da entegrenin ısınması anlamına gelir. GATE terminalinden çekilen ani akım entegre çıkışında akım sıçramaları meydana getirip PWM entegresine zarar verebilir. Bu durumu engellemek için VCC terminali ile GND terminali arasına by-pass kapasitörü konulur.

2.3.2 Bipolar Totem Pole Sürme Tekniği

En çok tercih edilen sürme tekniklerinden Bipolar Totem Pole None-Inverting sürme tekniği aşağıdaki gibidir. OUT terminalindeki sinyalin hiçbir faz farkı olamadan GATE terminaline aynen aktarılmasına Non-Inverting denir. Bu teknik aynı zamanda akım sıçramalarına dayanabilir. Şekil 2.10'da Bipolar Totem Pole Sürme Devresi gösterilmektedir.



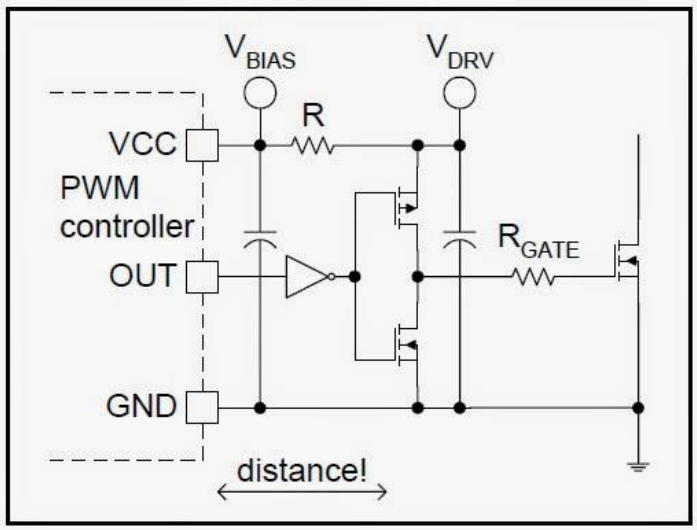
Şekil 2.10: Bipolar Totem Pole Sürme Devresi

Parazitik endüktansın daha küçük olma sebebi, ani akım sıçramalarının dolaştığı döngünün daha küçük olmasıdır. Aynı zamanda bu teknik VCC terminali ile GND terminali arasında ekstradan by-pass kapasitörü bulundurmaktadır ve bu da yukarıdaki gibi bağlanmaktadır. Burada R direnci opsiyonel olarak seçilir. Görevi gürültüyü azaltmaktır.

2.3.3 MOSFET Totem Pole Sürme Tekniği

MOSFET Totem Pole sürme tekniği Bipolar tekniğinin mosfetlerle yapılmış halidir denebilir. Bipolar teknikteki NPN, PNP çifti gibi bu teknikte de P kanal ve N kanal *MOSFET*ler kullanılmıştır. Bu teknik bazı dezavantajlara sahip olduğu gibi Bipolar teknikteki tüm avantajlara sahiptir. Bu tekniğin az kullanılma sebebi de bu dezavantajlardır. Bu teknikte

*MOSFET*lerden önce OUT sinyalinin tersi alınmalıdır. Çünkü, OUT terminalindeki sinyal GATE terminaline ters olarak gelir. Bir diğer dezavantajı ise; Totem Pole mosfetlerinin Bipolar Totem Pole transistörlerinden daha pahalı oluşudur. Şekil 2.11'de MOSFET Totem Pole Sürme Devresi gösterilmektedir.

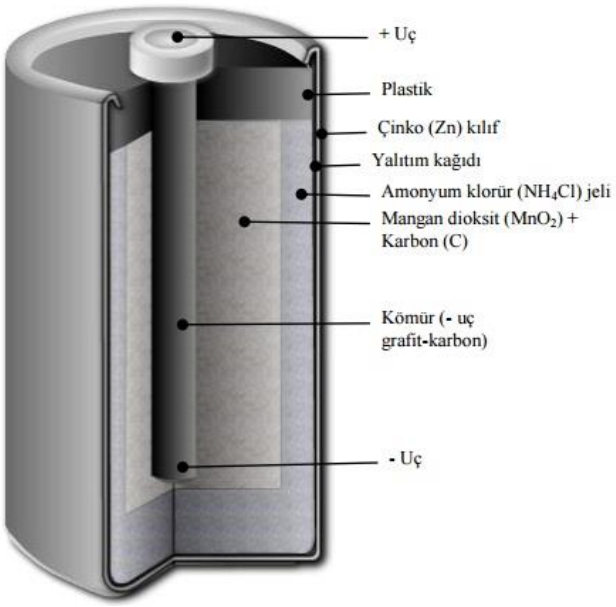


Şekil 2.11: MOSFET Totem Pole Sürme Devresi

2.4 Pil Çeşitleri

2.4.1 Çinko Karbon Pilleri

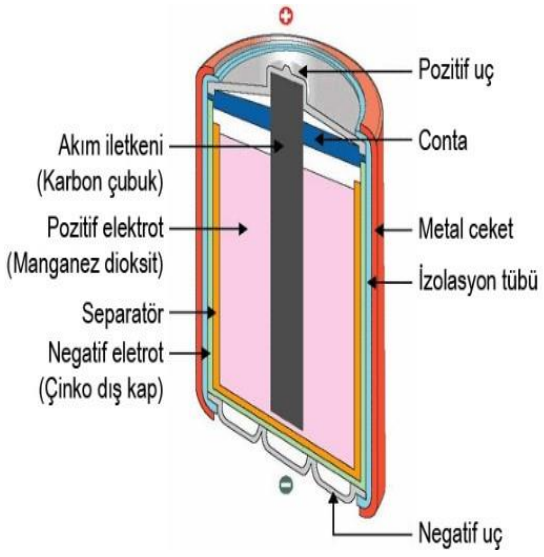
Çinko karbon piller iki kısımdan oluşur. Burada Çinko (Zn) negatif elektrodu oluştururken, Karbon (C) ise pozitif elektrodu oluşturur. Çinko karbon piller, pil çeşitleri arasındaki en ucuz pildir. Fakat şarj edilip tekrar tekrar kullanılamazlar. Kapasiteleride düşük olduğu için kullanım alanları da kısıtlıdır. Bir diğer dezavantajı da elektrolitlerinin zamanla pilin dışına akıp kullanıldıkları cihaza zarar vermeleridir. Şekil 2.12'de Çinko Karbon Pil gösterilmektedir.



Şekil 2.12: Çinko Karbon Pil

2.4.2 Alkaline Piller

Alkaline piller Çinko (Zn) ve Manganezden (Mn) oluşur. Alkaline piller de Çinko Karbon piller gibi şarj edilemezler. Çinko Karbon pillere nazaran daha maliyetlidirler. Avantajları ise kolaylıkla bulunabilmeleri, yüksek güçte olmaları ve boşalma hızlarının oldukça düşük olmasıdır. Şekil'2.13'de Alkaline Pil gösterilmektedir.



Şekil 2.13: Alkaline Pil

2.4.3 Nikel-Kadmiyum Piller

Nikel kadmiyum pillerde elektrodlar Nikel ve Kadmiyumdan oluşurken, elektrolit ise Potasyum Hidroksitten oluşur. Pilleri şarj etmeden önce tamamen deşarj olduğundan emin olunup, 1000 defaya kadar da şarj edilebilirler. Eğer tam olarak deşarj olmadan şarj edilirse hem pil kapasitesi düşer hem de pilin ömrü kısalır. Bu olaya da bellek etkisi denir. Şekil 2.14'de Nikel-Kadmiyum Pil gösterilmektedir.

Şekil 2.14: Nikel - Kadmiyum Pil

2.4.4 Nikel-Metal Hidrit Piller

Nikel-Metal Hidrit piller uzun ömürlü ve defalarca şarj edilebilen pillerdir. Yapılarında ise; Nikel, Titanyum, Kobalt, Magnezyum, Vanadyum, Alüminyum, Demir, Zirkonyum ve Krom bileşenleri bulunur. Bu pillerin bir pil birimi 1,2 volttur ve enerji kapasiteleri yüksektir. Nikel-Kadmiyum piller gibi bellek etkisi bu pillerde de geçerli olduğu için tamamen deşarj edilmeden

şarj edilmesi uygun olmayacaktır. Avantajının yanı sıra dezavantajı ise kullanılmadığı zaman kendi kendine deşarj olmalarıdır. Şekil 2.15'de Nikel-Metal Hidrit Pil gösterilmektedir.

Şekil 2.15: Nikel – Metal Hidrit Pil

2.4.5 Lityum İyon Piller

Lityum Iyon pillerde elektrotlar Karon ve Metal Oksiti oluştururken elektrolit ise Lityum tuzu çözeltisini oluşturur. Lityum Iyon pillerde şarj edilebilen piller arasındadır. Enerji yoğunluğu bakımından diğer pillerden çok daha yüksektir. Bu pillerin bir pil birimi 3,6 volttur. Lityum Iyon piller diğer pillere göre hafif, enerji kapasiteleri yüksek, şarj tutma süreleri de uzundur. Robot projelerinde çok tercih edilme sebebi de budur. Dezavantajı ise; diğer pillere göre maliyetli oluşudur. Bu pillerde bellek etkisi olmadığı içi tamamen deşarj olmadan şarj edilmelerinde bir sakınca yoktur. Şekil 2.16'da Lityum-Ion Pil gösterilmektedir.

Şekil 2.16: Lityum İyon Pil

2.4.5.1 Lityum İyon Pil Çeşitleri

Lityum Iyon piller Lityum Polimer pil ve Lityum Titanat pil olmak üzere iki başlık altında incelenir.

2.4.5.1.1 Lityum İyon Polimer Pil

Lityum Iyon Polimer pil, Lityum Iyon pil ile aynı özellikleri taşır. Tek farkı daha az şarj yoğunluğuna sahip olmasıdır. Kişinin ihtiyacına göre Lityum Iyon Polimer piller kullanıldıkları yerde avantaj sağlayabilirler.

2.4.5.1.2 Lityum İyon Tatanat Pil

Lityum Iyon Tatanat piller nano teknolojide kullanılmak amacı ile hazırlanmış, yeni geliştirilen bir pil teknolojisidir. Lityum Iyon Tatanat pillerin en büyük ve en güzel özelliği yüksek akım sayesinde çok seri şekilde şarj edilebilmesi ve 25.000 kez yeniden doldurulabilme kapasitesine sahip olmasıdır.

2.4.5.2 Lityum İyon Pillerin Avantajları

- Diğer pillere nazaran çok daha hafiftirler.
- Küçük ve taşınması kolaydır.
- Bellek etkisi yoktur (Yani tamamen deşarj olmadan şarj edilebilirler).
- Uzun ömürlüdürler.

2.4.5.3 Lityum İyon Pillerin Dezavantajları

- Kullanım ömürleri üretildikleri tarihten itibaren başlar.
- Kullanım yerindeki sıcaklık arttıkça pilin kullanım ömrü azalacaktır.(0 santigrat derecede %6, 25 santigrat derecede %20 ve 40 santigrat derecede %35 kayıplar oluşur.)
- Yüksek ısıya maruz kaldıklarında kolayca patlayabilirler. Sıcak ortamlarda ve direkt güneş ışığı alan yerlerde bekletilmemelidirler.

2.4.6 Kurşun Asit Aküler

1950'li yılların başında kullanılmaya başlayan ve kullanımı günümüze kadar devam eden akü tipidir. Pil birimleri 6 volt, 12 volt ve 24 volt olarak piyasada yaygın şekilde bulunabilirler. Yüksek akım sağladıklarından büyük güç gerektiren projelerde kullanılırlar. En büyük dezavantajı ise ağır olmalarıdır. Şekil 2.17'de Kurşun Asit Akü gösterilmektedir.

Şekil 2.17: Kurşun Asit Akü

2.5 STM32 Ailesine Genel Bakış

Günümüzde 8-bit mikrodenetleyiciler yerlerini ARM tabanlı mikrodenetleyicilere ve 32-bitlik mikrodenetleyicilere bırakmaktadır. Sağladığı özellikler bakımından ve ucuz olmalarından kaynaklı ilgiyi üzerlerinde tutmaktadırlar. 8-bit mikrodenetleyicilerde bulunmayan yüksek işlemci hızı, kapasiteleri ve ayrıca üstün çevre birimleri bulunmaktadır. Bu denetleyicilerden de en yaygın olarak kullanılanlardan birisi ise STM32 ailesidir.

STMicroElectronic firması STM32 ailesini üretti ve bu aile mikrodenetleyiciler pazarında yoğun bir ilgi görmüştür. Bu pazarda ilgi görmesinin sebebi fiyat anlamıyla diğer mikrodenetleyicilerden daha ucuz olması ve oldukça geniş yelpazesi ile kullanıcılara kolaylık sunmasıdır. Mikrodenetleyici ailesi içinde yer alan AVR veya PIC mikodenetleyici fiyatının yarısına 32-bit ARM tabanlı bir mikrodenetleyici alınabilir.

2.5.1 STM32 Serisi Çeşitleri

STM32 ailesinin ihtiyaçlara göre ürettiği birçok mikrodenetleyici çeşidi bulunmaktadır. Burada STM32 ailesinin bazılarına değilecektir.

2.5.1.1 STM32L0

Cortex-M0+ tabanlı mikrodenetleyicilerin en büyük özelliği çok düşük güç tüketimi gerçekleştirmelerinden kaynaklıdır. Burada "L" harfinin *Low Power* yani düşük güç anlamına gelmektedir. Minyatür ve kolay kullanımı sayesinde mikrodenetleyici kullanıcılarına başlangıç anlamında önerilebilirler. Kapasite ve hız beklentisi olan kullanıcılara bu mikrodeneteyiciler önerilmemelidir. Yine de 8-bit AVR ve PIC mikrodenetleyicilere nazaran çok çok üstündürler. Şekil 2.18'de STM32L0 Teknik Detay Tablosu verilmiştir.

Arm® Cortex®-M0+ (32 MHz with MPU)

- Low voltage 1.65 to 3.6V
- - 40 to 125°C oper. temp.
- 14 to 100 pins
- Dynamic voltage scaling
- 5 clock sources
- Advanced RTC w/ calibration
- Multiple USART, SPI, I²C
- Multiple 16-bit timers
- 5V tolerant I/Os
- 2 watchdogs
- Programmable voltage detector (PVD)
- Reset circuitry POR/PDR
- Brown-out Reset
- DMA
- Comparators
- Temperature sensor
- AES-128

Product	Flash (KB)	RAM (KB)	EEPROM (KB)	12-bit ADC 1.14 MSPS	LP[1] UART	LP[1] 16-bit timer	12-bit DAC	Touch sense	True RNG	USB 2.0 FS Crystal-less	Segment LCD Driver
STM32L0x1 Access	Up to 192	Up to 20	Up to 6	•	•	•					
STM32L0x2 USB	Up to 192	Up to 20	Up to 6	•	•	•	•	•	•	•	
STM32L0x3 USB & LCD	Up to 192	Up to 20	Up to 6	•	•	•	•	•	•	•	Up to 4x52 or 8x48

Note 1: Low-power peripherals available in ultra-low-power modes

Şekil 2.18: STM32L0 Teknik Detay Tablosu

Minimum 14 pin bağlantısı olan bu mikrodenetleyicilerde dahili olarak EEPROM mevcuttur. Kendi içinde üç alt serisi olan bu mikrodenetleyici 100 pinlik kılıfa kadar çıkabilmektedir.

2.5.1.2 STM32L1

STM32 ailesinin küçük güç tüketimine sahip bir diğer mikrodenetleyicisi ise STM32L1'dir. Diğer seriden ayıran özelliği ise Cortex-M3 işlemci çekirdeğinin olmasıdır. Bu çekirdeğin sıfırdan yükseldiğini görmek mümkündür. 230 mikro amperi Mega Hertz başına tüketmektedir. Bu da çok çok düşük bir güç tüketimi anlamına gelmektedir. Minimum işlemde güç tüketimi 0,27 mikroamperlere kadar düşmektedir. Önceki seriye göre özellik anlamında daha kapsamlıdır. Bunlarda birkaçını maddeler halinde sıralarsak;

- ARM Cortex-M3 32MHz > CPU(İşlemci),
- 32-384 Kb arası hafiza,
- 48 Kb'a kadar SRAAM,
- 12 Kb'a kadar EEPROM,
- 1,63-3,6 Volt arası çalışma gerilimi.

AVR ailesindeki mikrodenetleyicilere göre program hafızası hemen hemen aynı olsada RAM ve EEPROM bellekleri aşırı yüksektir. Bir diğer avantajı ise AVR'de olmayan çevre birimlerine sahip olmalarıdır. Örneğin bu mikrodenetleyici ailesinde OPAMP ve DAC bulunmaktadır ve bu özellikler AVR mikrodenetleyicisinde mevcut değildirler. Şekil 2.19'da STM32L1 Teknik Detay Tablosu verilmiştir.

Arm® Cortex®-M3 – 32 MHz	STM32 L1 Product lines	Flash (Kbytes)	RAM (Kbytes)	EEPROM (Kbytes)	Memory I/F	Op-Amp	Comp.	Temp. Sensor	Capacitive Touch	Segment LCD Driver	AES 128-bit
• Reset POR/PDR • 2x watchdogs • Hardware CRC • Internal RC • Crystal oscillators • PLL • RTC calendar • 16- and 32-bit timers • 1x12-bit ADC • Temperature sensor • Multiple-channel DMA • Single-wire debug • Unique ID • USB 2.0 (with internal 48 MHz PLL)	**STM32L100 Value line**	32 to 256	4 to 16	2						Up to 8 x 28	
	STM32L151 STM32L152	32 to 512	16 to 80	4 to 16	SDIO FSMC	•	•	•	•	Up to 8 x 40	
	STM32L162	256 to 512	32 to 80	8 to 16	SDIO FSMC	•	•	•	•	Up to 8 x 28	•

Şekil 2.19: STM32L1 Teknik Detay Tablosu

2.5.1.3 STM32L4

Küçük güç tüketimi başlığı adı altında incelediğimiz bu seri özellik bakımından giriş seviyesi demek pek de mümkün değildir. Cortex-M4 tabanına sahip bu seri yüksek performans ve küçük güç tüketimi sunmaktadır. Bu mikrodenetleyicide 16-bit ADC, OPAMP, KARIŞTIRICI, 12-bit DAC gibi çevre birimleri mevcut iken aynı zamanda 80 MHz'de 100 DMIPS performansına sahiptir. (DMIPS: DhryStone MIPS'nin kısaltmasıdır ve işlemci performans bilgisi anlamına gelir.) Şekil 2.20'de STM32L4 Teknik Detay Tablosu verilmiştir.

Arm® Cortex®-M4 (DSP + FPU) – 80 MHz

- ART Accelerator™
- USART, SPI, I²C
- Quad-SPI
- 16- and 32-bit timers
- SAI + audio PLL
- SWP
- 2x CAN
- 2x 12-bit DACs
- Temperature sensor
- Low voltage 1.71 to 3.6 V
- V_{BAT} mode
- Unique ID
- Capacitive touch sensing
- AES-128/256* and SHA-256**

Product line	Flash (KB)	RAM (KB)	Memory I/F FSMC	Op-Amp	CAN	Sigma Delta interface	12-bit ADC 5 Msps 16-bit HW oversampling	DAC	SAI	USB2.0 OTG FS	USB Device	Segment LCD driver	Chrom-ART Accelerator™
STM32L4x6 - USB OTG + Segment LCD Lines													
STM32L496**	512 to 1024	320	•	2	2	8x ch	3	2	2	•		Up to 8x40	•
STM32L476*	256 to 1024	128	•	2	1	8x ch	3	2	2	•		Up to 8x40	
STM32L4x5 - USB OTG lines													
STM32L475	256 to 1024	128	•	2	1	8x ch	3	2	2	•			
STM32L4x3 - USB Device + Segment LCD lines													
STM32L433*	128 to 256	64		1	1		1	2	1		•	Up to 8x40	
STM32L4x2 - USB Device lines													
STM32L452*	256 to 512	160		1	1	4x ch	1	1	1		•		
STM32L432*	128 to 256	64		1	1		1	2	1		•		
STM32L412*	64 to 128	40		1			2				•		
STM32L4x1 - Access lines													
STM32L471	512 to 1024	128	•	2	1	8x ch	3	2	2				
STM32L451	256 to 512	160		1	1	4x ch	1	1	1				
STM32L431	128 to 256	64		1	1		1	2	1				

Note: * HW crypto/hash functions are available on STM32L486, STM32L443, STM32L462, STM32L442 and STM32L422 - ** on STM32L4A6

Şekil 2.20: STM32L4 Teknik Detay Tablosu

2.5.1.4 STM32F0

Bu seri STM32 ailesinin giriş seviyesindeki en temel mikrodenetleyicisidir. ARM Crortex-M0 tabanlı olmasından dolayı, ARM Cortex-M0+ tabanlı mikrodenetleyicilerden bile daha alt seviyededirler. 8-bit mikrodenetleyicilerin kullanıldığı yerlerde rahatlıkla kullanılabilirler. Özelliklerinin seviye anlamında düşük olması fiyat anlamında da daha uygun hale getirmektedir. Fakat 8-bit mikrodenetleyicilere göre daha fazla özellikleri ya da üstünlükleri vardır. Bunları şöyle sıralayabiliriz;

- 32-bit ARM Cortex-M0 çekirdek 48MHz 12 DMA kanallı,
- 1,8-3,6 Volt çalışma gerilimi,
- 2 x SPI (8MHz'den 96MHz'e kadar frekans örneklemesi),
- 3 x I2C,
- USB tam hız,
- 8'e kadar çıkabilen USART birimi,
- 11'e kadar çıkabilen zamanlayı/sayıcı (16-32-bit),
- RTC,
- 12 MHz hızında giriş/çıkış birimleri,
- Saniyede bir milyon örnek alabilen 12-bit ADC,
- 12-bit DAC,
- 2 x Analog karşılaştırıcı.

Şekil 2.21'de STM32F0 Teknik Detay Tablosu verilmiştir.

ARM® Cortex®-M0 – 48 MHz

- Reset POR/PDR
- 2x watchdogs
- Hardware CRC
- Internal RC
- Crystal oscillators
- PLL
- RTC calendar
- 16- and 32-bit timers
- 1x12-bit ADC
- Temperature sensor
- Multiple-channel DMA
- Single-wire debug
- Unique ID

STM32 F0 Product lines	Flash (Kbytes)	RAM (Kbytes)	Power supply	20-byte backup data	12-bit DAC	Comp.	Touch sense	Up to 2x SPI/ PS, 2x I²C	USART	CEC	CAN	USB
STM32F0x0 Value line	16 to 256	4 to 32	2.4 to 3.6 V					•	6			•
STM32F0x1 Access line	16 to 256	4 to 32	2.0 to 3.6 V	•	•	•	•	•	8	•	•	
STM32F0x2 USB line	16 to 128	4 to 16	2.0 to 3.6 V	•	•	•	•	•	4	•	•	• (crystal-less)
STM32F0x8 Low-voltage line	32 to 256	4 to 32	1.8 V ± 8%	•	•	•	•	•	8	•		• (crystal-less)

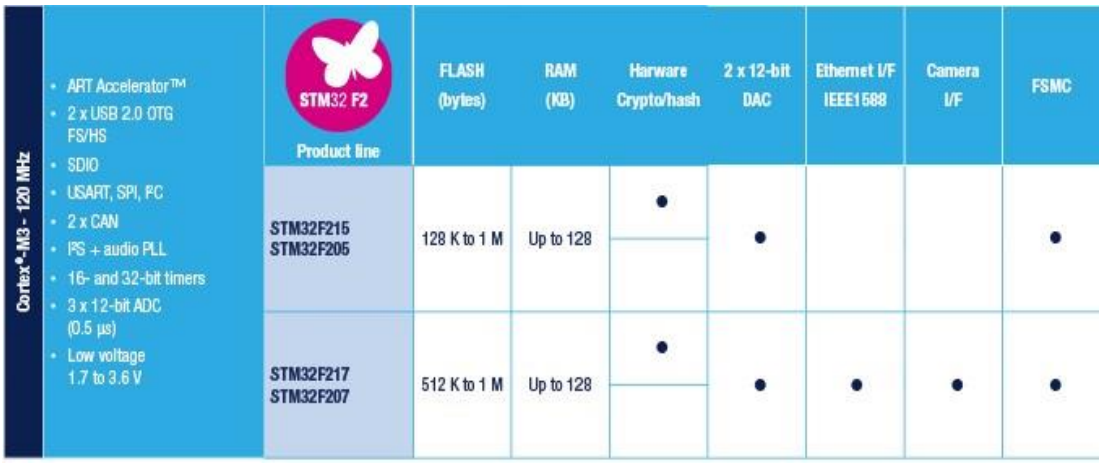
Şekil 2.21: STM32F0 Teknik Detay Tablosu

2.5.1.5 STM32F2

Bu mikrodenetleyiciler ARM Cortex-M3 tabanına sahiptirler ve F0 serisine göre işlem hızları daha yüksek ve daha fazla kapasiteye sahiptirler. Daha iyi bir performans sunmak için Cortex-M4 kullanılmaya başlanmıştır. Belli başlı özelliklerini şu şekilde sıralayabiliriz;

- 120 MHz'de 150 DMIPS İşlem hızı,
- 16 DMA kanalı,
- 528 Byte OTP (bir defa programlanabilir) bellek. Önemli bilgileri MAC adresine ya da şifre çözme anahtarına yazmak için kullanılabilir.
- 2 x USB OTG,
- 1 x Ethernet MAC,
- 2 x CAN 2.0 B,
- 3 x SPI (30Mbit/s' e kadar),
- 6 x USART (7,5Mbit/s'e kadar),
- 3 x I2C,
- 2 x I2C,
- LCD Sürücüsü,
- 2 x Motor Sürücü Zamanlaıcısı,
- 12 x Zamanlayıcı ve Sayıcı,
- 2 x 32-bit Zamanlayıcı ve Sayıcı,
- 3 x 12-bit ADC,
- 2 x 12-bit DAC,
- 60MHz hızında giriş/çıkış,
- RTC,
- Şifreleme,
- Gerçek rastgele sayı üretici.

Özelliklere bakılınca üst düzey bir mikrodenetleyici olduğu görülmektedir. AVR ve PIC çevre birimleri bu işlemciye göre çok geri planda kalmaktadır. Şekil 2.22'de STM32LF2 Teknik Detay Tablosu verilmiştir.



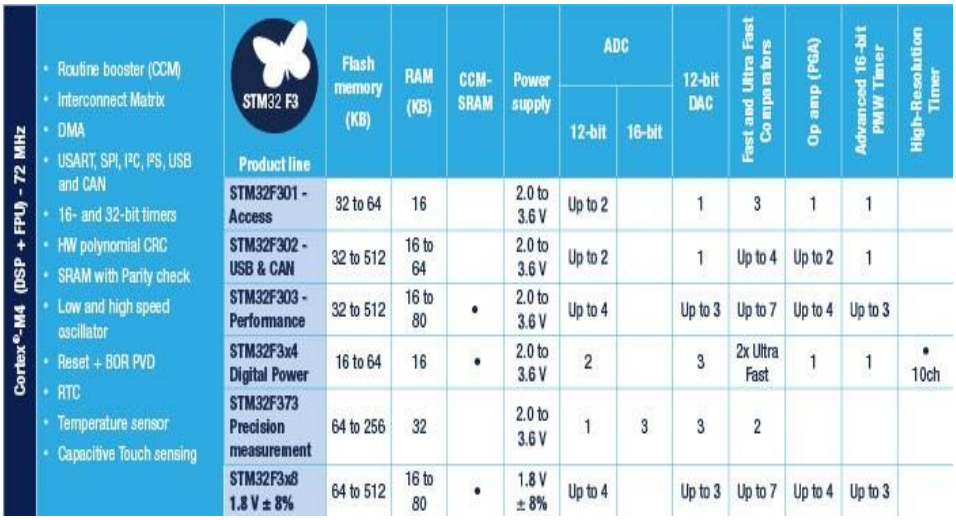
Şekil 2.22: STM32LF2 Teknik Detay Tablosu

2.5.1.6 STM32F3

Bu mikrodenetleyici serisi Cortex-M4 çekirdeğine sahiptir ve kullanım alanı bakımından sinyal işleme ve analog çevre birimleri ile ön plandadır. Yani analog sistemler ile çalışacaklar için büyük avantajlar sunmaktadır. Bu seride ondalık işlemler yapmak DSP ve FPU (Floazing Point Unit) sayesinde mümkündür. Gömülü olarak gelen DSC 12-bit 5 MSPS hızda çalışmaktadır ve programlanabilir amplifikatörler, karşılaştırıcılar,16-bit ADC bulunmaktadır. Teknik özellikleri daha detaylı incelersek;

- 4 x 12-bit ADC (saniyede 5 milyon örnekleme),
- 3 x 16-bit Sigma Delta ADC (saniyede 50 bin örnekleme),
- 12-bit DAC,
- 25 nanosaniye ve 50 nanosaniyelik karşılaştırıcılar,
- PGA (Programlanabilir Kazanç Amplifikatörü),
- 17 Adete kadar Zamanlayıcı/Sayıcı,
- 5 x USART,
- 3 x SPI/12s,
- 2 x I2C,
- 1 x CAN,
- USB,
- RTC.

Şekil 2.23'de STM32F3 Teknik Detay Tablosu verilmiştir.



Cortex®-M4 (DSP + FPU) - 72 MHz

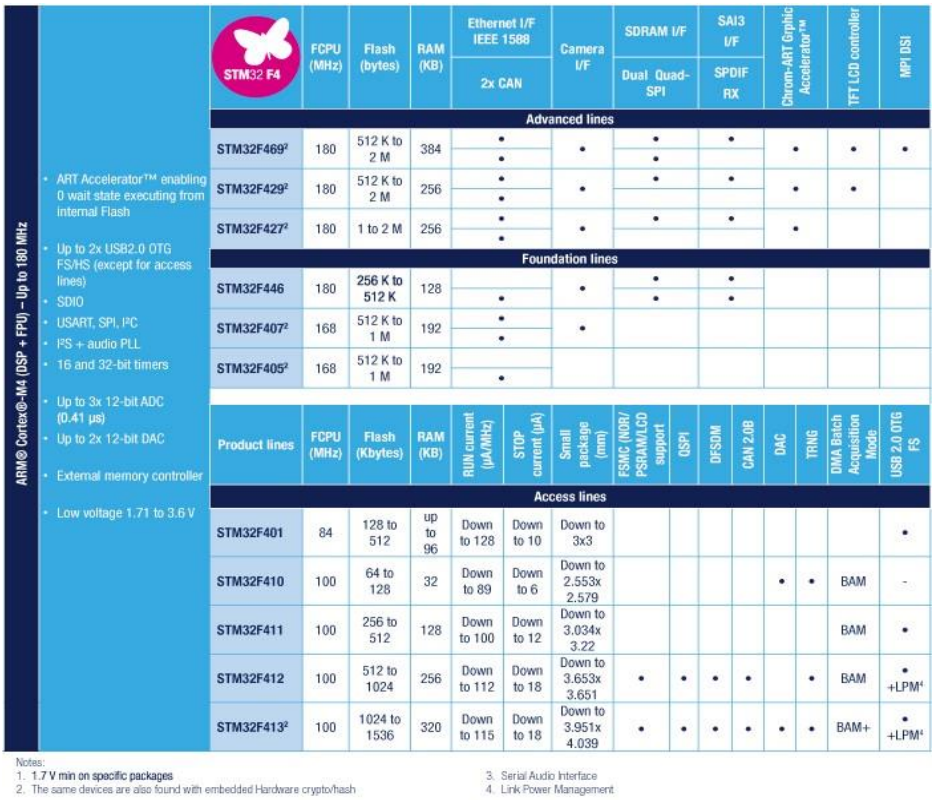
- Routine booster (CCM)
- Interconnect Matrix
- DMA
- USART, SPI, I²C, IrS, USB and CAN
- 16- and 32-bit timers
- HW polynomial CRC
- SRAM with Parity check
- Low and high speed oscillator
- Reset + BOR PVD
- RTC
- Temperature sensor
- Capacitive Touch sensing

STM32 F3 Product line	Flash memory (KB)	RAM (KB)	CCM-SRAM	Power supply	ADC 12-bit	ADC 16-bit	12-bit DAC	Fast and Ultra Fast Comparators	Op amp (PGA)	Advanced 16-bit PMW Timer	High-Resolution Timer
STM32F301 - Access	32 to 64	16		2.0 to 3.6 V	Up to 2		1	3	1	1	
STM32F302 - USB & CAN	32 to 512	16 to 64		2.0 to 3.6 V	Up to 2		1	Up to 4	Up to 2	1	
STM32F303 - Performance	32 to 512	16 to 80	•	2.0 to 3.6 V	Up to 4		Up to 3	Up to 7	Up to 4	Up to 3	
STM32F3x4 Digital Power	16 to 64	16	•	2.0 to 3.6 V	2		3	2x Ultra Fast	1	1	• 10ch
STM32F373 Precision measurement	64 to 256	32		2.0 to 3.6 V	1	3	3	2			
STM32F3x8 1.8 V ± 8%	64 to 512	16 to 80	•	1.8 V ± 8%	Up to 4		Up to 3	Up to 7	Up to 4	Up to 3	

Şekil 2.23: STM32F3 Teknik Detay Tablosu

2.5.1.7 STM32F4

Bu mikrodenetleyicide Cortex-M4 çekirdeği kullanılmaktadır ve içyapısında bir sistem, bir hiyerarşi barındırırlar. Diğer mikrodenetleyicilerden farklı olarak TFT (Thin Film Transistor) sürücüsü bulunmaktadır. Thin Flim Transistör için mini bir LCD ekran diyebiliriz. Burada her piksel 4 transistör tarafından kontrol edilir. Ayrıca 180 MHz hızında işlem yapabilirler. Üst seviye bir denetleyicidir. Teknik özellikleri şekil 2.24'de belirtilmiştir.



ARM® Cortex®-M4 (DSP + FPU) – Up to 180 MHz

- ART Accelerator™ enabling 0 wait state executing from internal Flash
- Up to 2x USB2.0 OTG FS/HS (except for access lines)
- SDIO
- USART, SPI, I²C
- I²S + audio PLL
- 16 and 32-bit timers
- Up to 3x 12-bit ADC (0.41 µs)
- Up to 2x 12-bit DAC
- External memory controller
- Low voltage 1.71 to 3.6 V

STM32 F4	FCPU (MHz)	Flash (bytes)	RAM (KB)	Ethernet I/F IEEE 1588	Ethernet I/F 2x CAN	Camera I/F	SDRAM I/F	SDRAM I/F Dual Quad-SPI	SAI3 I/F	SAI3 I/F SPDIF RX	Chrom-ART Graphic Accelerator™	TFT LCD controller	MPI DSI
Advanced lines													
STM32F469[2]	180	512 K to 2 M	384	•	•	•	•	•	•		•	•	•
STM32F429[2]	180	512 K to 2 M	256	•	•	•	•		•		•	•	
STM32F427[2]	180	1 to 2 M	256	•	•	•	•		•		•		
Foundation lines													
STM32F446	180	256 K to 512 K	128		•	•	•	•	•	•			
STM32F407[2]	168	512 K to 1 M	192	•	•	•							
STM32F405[2]	168	512 K to 1 M	192	•									

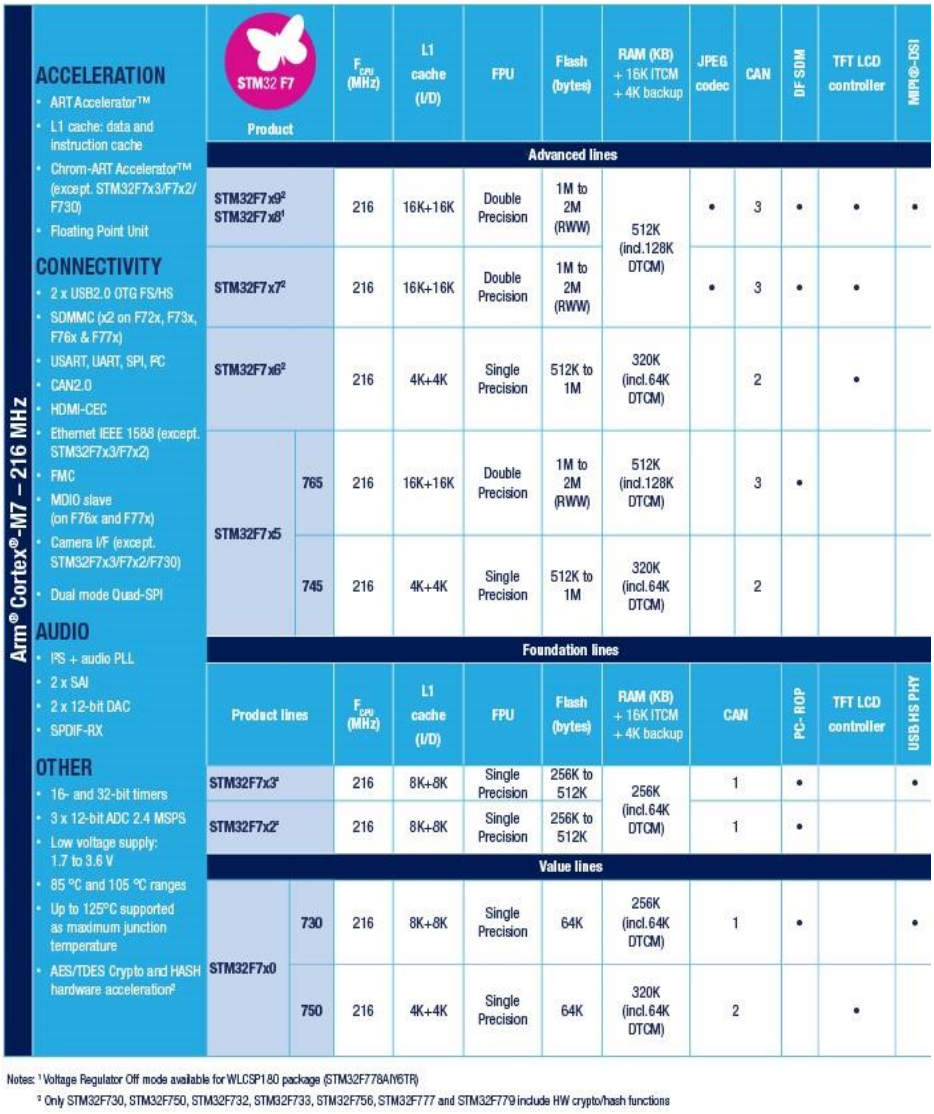
Product lines	FCPU (MHz)	Flash (Kbytes)	RAM (KB)	RUN current (µA/MHz)	STOP current (µA)	Small package (mm)	FSMC (NOR/ PSRAM/LCD support	QSPI	DFSDM	CAN 2.0B	DAC	TRNG	DMA Batch Acquisition Mode	USB 2.0 OTG FS
Access lines														
STM32F401	84	128 to 512	up to 96	Down to 128	Down to 10	Down to 3x3								•
STM32F410	100	64 to 128	32	Down to 89	Down to 6	Down to 2.553x 2.579					•	•	BAM	-
STM32F411	100	256 to 512	128	Down to 100	Down to 12	Down to 3.034x 3.22							BAM	•
STM32F412	100	512 to 1024	256	Down to 112	Down to 18	Down to 3.653x 3.651	•	•	•	•		•	BAM	• +LPM[4]
STM32F413[2]	100	1024 to 1536	320	Down to 115	Down to 18	Down to 3.951x 4.039	•	•	•	•	•	•	BAM+	• +LPM[4]

Notes:
1. 1.7 V min on specific packages
2. The same devices are also found with embedded Hardware crypto/hash
3. Serial Audio Interface
4. Link Power Management

Şekil 2.24: STM32F4 Teknik Detay Tablosu

2.5.1.8 STM32F7

STM32 serisinin en üst seviye mikrodenetleyici modellerindendir. ARM Cortex-M7 çekirdeği bulunur. Hız gerektiren ve işlem gücü bakımından yüksek performans gerektiren yerlerde kullanılır. 100-216 arasında pin sayısı vardır. 400MHz'e kadar hıza sahiptirler. Bu serinin teknik özellikleri aşağıdaki şekil 2.25'de belirtilmiştir.



Arm® Cortex®-M7 – 216 MHz

ACCELERATION
- ART Accelerator™
- L1 cache: data and instruction cache
- Chrom-ART Accelerator™ (except. STM32F7x3/F7x2/F730)
- Floating Point Unit

CONNECTIVITY
- 2 x USB2.0 OTG FS/HS
- SDMMC (x2 on F72x, F73x, F76x & F77x)
- USART, UART, SPI, I²C
- CAN2.0
- HDMI-CEC
- Ethernet IEEE 1588 (except. STM32F7x3/F7x2)
- FMC
- MDIO slave (on F76x and F77x)
- Camera I/F (except. STM32F7x3/F7x2/F730)
- Dual mode Quad-SPI

AUDIO
- I²S + audio PLL
- 2 x SAI
- 2 x 12-bit DAC
- SPDIF-RX

OTHER
- 16- and 32-bit timers
- 3 x 12-bit ADC 2.4 MSPS
- Low voltage supply: 1.7 to 3.6 V
- 85 °C and 105 °C ranges
- Up to 125°C supported as maximum junction temperature
- AES/TDES Crypto and HASH hardware acceleration[2]

STM32 F7

Product		F_{CPU} (MHz)	L1 cache (I/D)	FPU	Flash (bytes)	RAM (KB) + 16K ITCM + 4K backup	JPEG codec	CAN	DF SDM	TFT LCD controller	MIPI®-DSI
Advanced lines											
STM32F7x9[2] STM32F7x8[1]		216	16K+16K	Double Precision	1M to 2M (RWW)	512K (incl.128K DTCM)	•	3	•	•	•
STM32F7x7[2]		216	16K+16K	Double Precision	1M to 2M (RWW)		•	3	•	•	
STM32F7x6[2]		216	4K+4K	Single Precision	512K to 1M	320K (incl.64K DTCM)		2		•	
STM32F7x5	765	216	16K+16K	Double Precision	1M to 2M (RWW)	512K (incl.128K DTCM)		3	•		
	745	216	4K+4K	Single Precision	512K to 1M	320K (incl.64K DTCM)		2			
Foundation lines											
Product lines		F_{CPU} (MHz)	L1 cache (I/D)	FPU	Flash (bytes)	RAM (KB) + 16K ITCM + 4K backup	CAN		PC- ROP	TFT LCD controller	USB HS PHY
STM32F7x3[3]		216	8K+8K	Single Precision	256K to 512K	256K (incl.64K DTCM)	1		•		•
STM32F7x2[3]		216	8K+8K	Single Precision	256K to 512K		1		•		
Value lines											
STM32F7x0	730	216	8K+8K	Single Precision	64K	256K (incl.64K DTCM)	1		•		•
	750	216	4K+4K	Single Precision	64K	320K (incl.64K DTCM)	2			•	

Notes: [1] Voltage Regulator Off mode available for WLCSP180 package (STM32F778AIY6TR)

[2] Only STM32F730, STM32F750, STM32F732, STM32F733, STM32F756, STM32F777 and STM32F779 include HW crypto/hash functions

Şekil 2.25: STM32F7 Teknik Detay Tablosu

2.5.1.9 STM32H7

Bu denetleyici serisinde de ARM Cortex-M7 işlemcisi kullanılmakta ve yine üst seviye mikrodenetleyici ailesi arasında yerini almaktadır. İlgili teknik detayı şekil 2.26'da incelenebilir.

Arm® Cortex®-M7 – 400 MHz

CORE, MEMORIES AND ACCELERATION
- Cortex-M7 core @ 400 MHz
- 16KB+16KB I/D L1 Cache
- Double-precision FPU
- 4 x DMA

CONNECTIVITY
- 2 x USB2.0 OTG FS/HS
- 2 x SDMMC
- USART, UART, SPI, I²C
- 2 x CAN (1 x FD and 1 x TT/FD)
- HDMI-CEC
- FMC
- Analog (comp, AOP)

AUDIO
- 3 x I²S + audio PLL
- 4 x SAI
- 2 x 12-bit DAC
- SPDIF-RX

GRAPHIC
- LCD TFT controller
- JPEG Codec
- Chrom-ART Accelerator™

OTHER
- AES/TDES Crypto and HASH hardware acceleration[a]
- Software firmware install[b]
- TRNG
- DFSDM
- 16- and 32-bit timers
- 3 x 14-bit ADC (2 Msps)
- Low voltage 1.7 to 3.6V
- Multi power domains
- -40 to +85 °C temperature range

STM32 H7 Product line	F_{CPU} (MHz)	Core	Flash (bytes)	RAM (KB)	Dual Quad-SPI	Ethernet I/F IEEE 1588	Camera I/F
Advanced lines							
STM32H7x7	Available in 2019						
STM32H7x5	Available in 2019						
Foundation lines							
STM32H753	400	Cortex-M7	Up to 2MB (dual bank)	1MB (incl.128K DTCM) + 64K ITCM + 64KB bckup1 + 4K bckup2	•	•	•
STM32H743	400	Cortex-M7	Up to 2MB (dual bank)	1MB (incl.128K DTCM) + 64K ITCM + 64KB bckup1 + 4K bckup2	•	•	•
Value line							
STM32H750	400	Cortex-M7	128K	1MB (incl.128K DTCM) + 64K ITCM + 64KB bckup1 + 4K bckup2	•	•	•

Notes:
- (a) Available on STM32H750 and STM32H753 only
- (b) Available in 2019 on STM32H750 and STM32H753 only

Şekil 2.26: STM32H7 Teknik Detay Tablosu

2.6 Sensörler

Burada sensörlerin görevi ve sensör çeşitlerine değinilecektir.

2.6.1 Sensörlerin Görevi

Sensörler projelerde mesafe, ışık, sıcaklık gibi fiziksel büyüklükleri elektrik sinyallerine dönüştürmek ve bu bilgileri işleyecek karar mekanizmalarını kurabilme gibi görevleri vardır.

2.6.2 Sensör Teknolojisi

Otomasyon ve akıllı projeler sensör ve sensörlerin bir araya gelmesiyle oluşturulur. Endüstri 4.0 sensör teknolojisi ile ortaya çıkmıştır ve sensör teknolojisi birçok projenin gelişimine zemin hazırlayarak her geçen gün daha çok önem kazanmaktadır.

Sensör teknolojilerinin çoğalması hayatımızda büyük önem taşır çünkü hayatımızın her noktasında vardır.

2.6.3 Sensör Çeşitleri

Sensörleri analog ve dijital olmak üzere iki başlık adı altında incelenir.

2.6.3.1 Analog Sensörler

Ölçtükleri fiziksel büyüklüğe bağlı olarak akım ve gerilim bilgisi verirler. Kontrol kartlarına bu sensörleri bağlayabilmek için analog dijital çeviriciler (ADC – Analog Digital Converter) kullanılır. Analog dijital çeviriciler hassasiyetlerine göre harici olarak bağlanabilirken aynı zamanda mikrodenetleyici kartlarının içinde de yer alabilirler (STM32, Arduino, PIC vb).

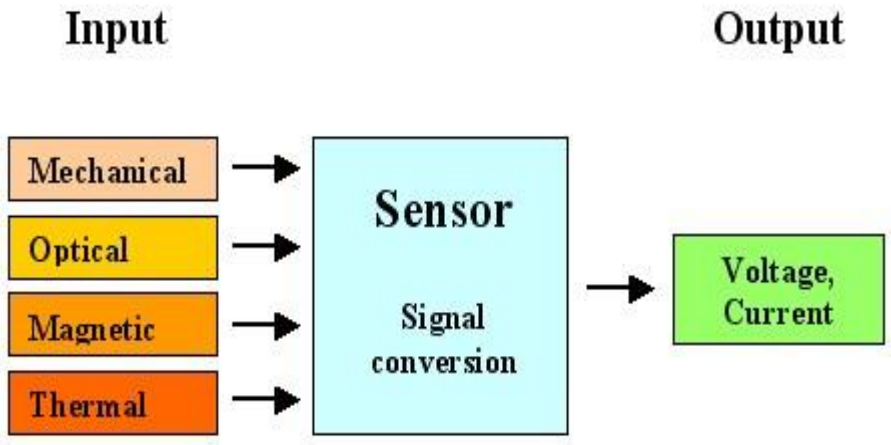
Son dönemde yaygın olarak kullanılan Rasberry Pi'de ise dahili olarak analog dijital çevirici bulunmaz. Bu nedenle analog girişe ihtiyaç duyulduğunda harici bir devre elemanı kullanılması gerekir.

2.6.3.2 Dijital Sensörler

Dijital sensörler haberleşme protokolleri (I2C, SPI, One Wire vb.) aracılığı ile bilgisayar ile iletişim kurarlar. Opamp devre elemanı ile kullanılan analog sensörler lojik bir sinyal üretirler. Rasberry Pi gibi dahili analog dijital çeviriciye sahip olmayan kartlar analog çıkışlı sensörler ile

kullanılırlar.

Şekil 2.27'de Analog Dijital Çevirici Mantığı blok diyagramı verilmiştir.



Şekil 2.27: Analog Dijital Çevirici Mantığı

2.6.4 Giriş Büyüklüklerine Göre Sensör Çeşitleri

Giriş büyüklüğüne göre sensörler 6 başlık altında incelebilirler;

- Mekanik Sensörler; İvme, hız, miktar, alan, uzunluk, kuvvet, tork, basınç, akış, pozisyon vb.
- Termal Sensörler; Isı ve sıcaklık vb.
- Elektriksel Sensörler; Akım, gerilim, kapasitans, endüktans, drenç, frekans, elektrik alan vb.
- Manyetik Sensörler; Manyetik moment, geçirgenlik, alan yoğunluğu, akı yoğunuğu vb.
- Işıma Sensörleri; Dalga boyu, polarizasyon, yansıtma, gönderme, yoğunluk vb.
- Kimyasal Sensörler; Yoğunlaşma, reaksiyon hızı, Ph miktarı vb.

Elektrik projlerinde ve robot sistemlerinde en yaygın şekilde kullanılan sensörleri biraz daha yakından incelersek;

- Mesafe Sensörü; Kızılötesi, endüktif, kapasitif, pır, ultrasonik vb.
- Kuvvet Ağırlık Basınç Sensörleri;
- Manyetik Sensörler; Hall effect, read role vb.
- Sıcaklık Sensörleri; PTC, NTC vb.
- Ses Sensörleri; Mikrofon, kristal, karbon vb.
- Işık ve Renk Sensörleri; LDR, RGB, fototransistör, fotodiyot vb.

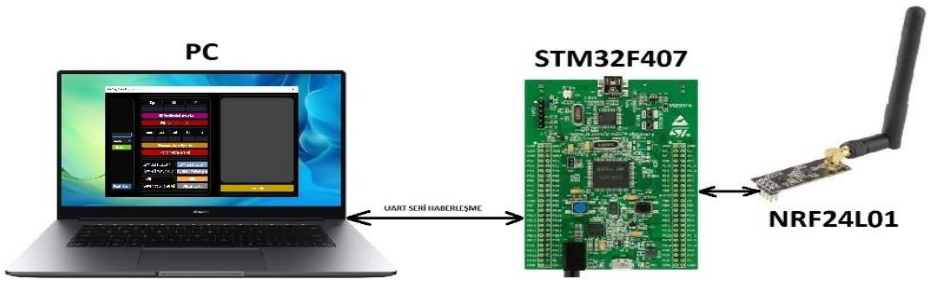
3. DONANIM

3.1 Giriş

Tezin bu kısmında Metrocar Araçların Sabit İveli Hareketi iki alt başlık olarak incelenecektir. Projenin Master yani kontrolcü kısmı, kullanıcıdan aldığı verileri ve komutları kablosuz şekilde araca iletecek ve araçtan gelen verileri kullanıcı arayüzüne aktaracaktır. Projenin Slave yani kontrol edilen kısmı, kontrolcüden gelen emirleri ve bilgileri değerlendirerek yerine getirecek ve geribildirimde bulunacaktır. Yani Master ve Slave bölümünde kullanılan bütün donanımsal parçalara detaylı şekilde yer verilecektir.

3.2 Kontrolcü Blok Yapısı

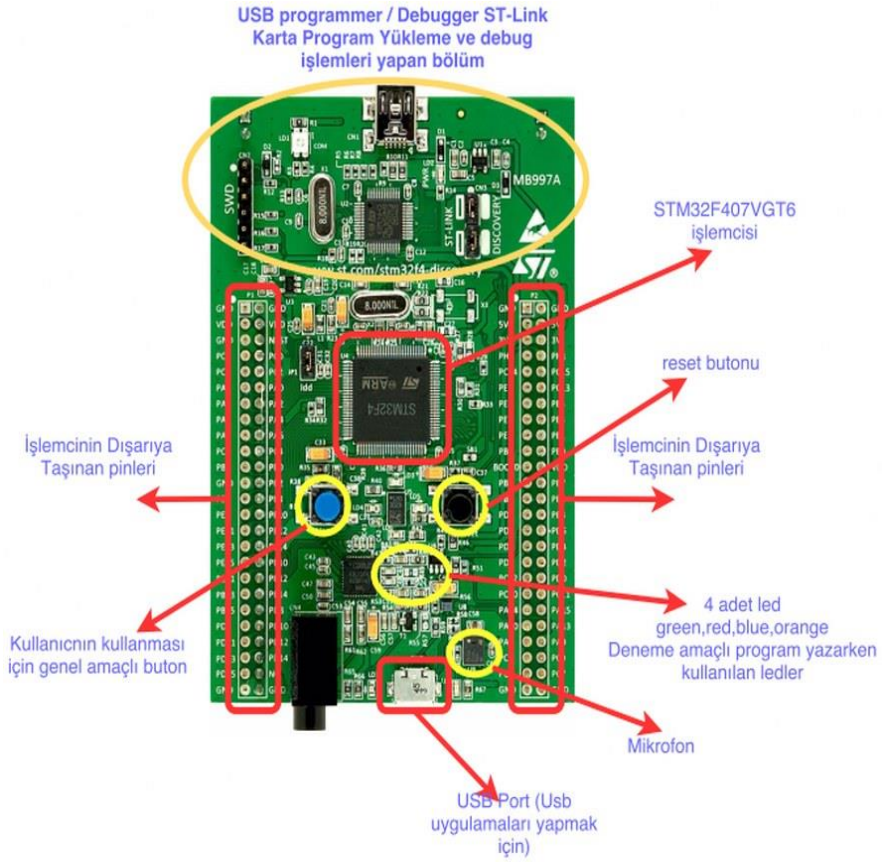
Tezin bu kısmında Master yapısına değinilecektir. Burada uzaktan komut göndereceğimiz ve gerekli ayarları yapabileceğimiz bir bilgisayar mevcuttur. Bilgisayarımızın USB portu UART dönüştürücü vasıtası ile STM32F407 Discovery Kartına bağlantısı yapılmıştır. STM32F407 mikrodenetleyicinin de SPI haberleşme çıkışları aktif edilerek NRF24L01 Wi-Fi modülün bağlantısı yapılmıştır. Şekil 3.1'de Kontrolcü Blok Yapısı verilmiştir.



Şekil 3.1: Kontrolcü Blok Yapısı

3.2.1 STM32F407 Mikrodenetleyici Kit

Tezin Master bölümünde STM32F4 ailesinden STM32F407 mikrodenetleyicisi seçilmiştir. Bu seri 168 MHz saat çalışma frekansına sahip 32-bit'lik bir mikrodenetleyicidir. Üzerinde STM32F4VG çekirdeğini barındırmaktadır. 3-eksen ivme ölçer, dahili ses girişi ve buna bağlı dijital analog dönüştürücü, 4 adet programlanabilir LED, 2 adet buton barındırır. Aşağıdaki şekil 3.2'de sarı yuvarlak içine alınan bölge kitin kendi çekirdeğine program yükleme ve debug işlemlerini yapabileceğiniz st-linkV2 kısmıdır (WEB_3).



Şekil 3.2: *Master*'da Kullanılan STM32F407 Discovery Modülü

Başlıca teknik özelliklerine değinecek olursak;

- 32-bit ARM Cortex-M4F çekirdeğine sahip STM32F407VGT6 mikrodenetleyicisi, 1 MB Flash ve 192 KB RAM,
- Dahili ST-LINK/V2 JTAG Debugger,

- USB veya harici kaynaktan doğrudan 5 Volt ile çalışabilme,
- 3 Volt ve 5 Volt çıkış pinleri,
- 3-Eksenli ivme ölçer,
- D sınıfı yükselteçli ses sürücü çipi,
- Sekiz adet led,
    - LD1 (kırmızı/yeşil)(USB haberleşmesi için),
    - 3,3 Volt power on/of ledi,
    - Dört adet kullanıcı ledi, LD3 (turuncu), LD4 (yeşil), LD5 (kırmızı), LD6 (mavi),
    - 2 adet USB OTG LEDi, LD7 (yeşil) ve LD8 (kırmızı),
- Bir adet Reset ve bir adet kullanıcı tanımlı buton,
- USB OTG için mikro-AB konnektör,
- 100 pin'in tamamını kullanabilmeye imkan tanıyan çıkışlar.

3.2.2 NRF24L01 Wi-Fi Haberleşme Modülü

Tezimizde Master ile Slave arasında kablosuz haberleşmeyi sağlamak amacı ile NRF24L01 Wi-Fi modülü seçilmiştir. NRF24L01 kablosuz haberleşme modülü, 2.4GHz frekansında kablosuz haberleşme yapmanıza imkan sağlayan düşük güç tüketimine sahip modüldür. Ayrıca 2MBps haberleşme hızına sahip olup, SPI arabirimini destekler (WEB_4).

Başlıca teknik özelliklerine değinecek olursak;

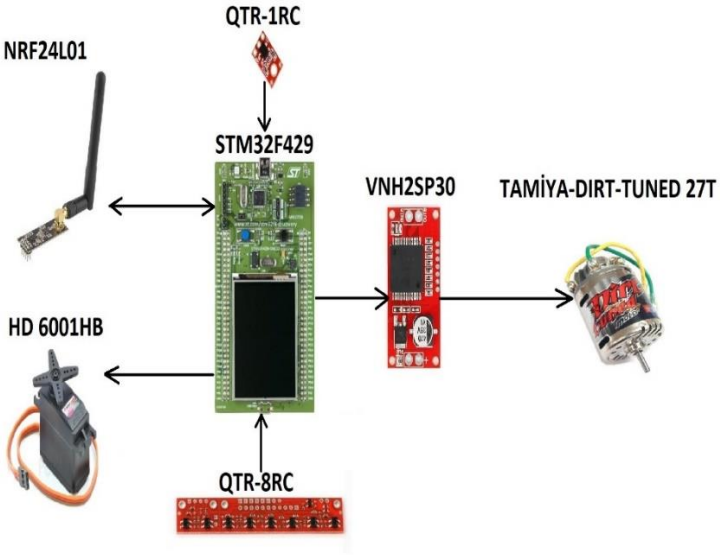
- 2.4GHz bandında yayın yapabilir.
- 250KBps, 1MBps ve 2MBps gibi hızlarda haberleşme hızı seçilebilir.
- Gelişmiş ShockBurst™ hızlandırma protokolünü desteklemektedir.
- Ultra düşük güç tüketimi.
- Çalışma Voltajı: 1.9-3.6 Volt.
- IO Portları Çalışma Voltajı:0-3.3 Volt/5 Volt.
- Verici Sinyal Gücü: +7 dB.

- Alıcı Hassasiyeti ≤ 90dB.
- Haberleşme Mesafesi: Açık Alanda 250m.
- Boyutları: 15x29mm.

Şekil 3.3: *Master*'de Kullanılan NRF24L01 Wi-Fi Modülü

3.3 Araç Blok Yapısı

Tezin bu kısmında Slave yapısına değineceğiz. Burada Master yani STM32F407'nin pinlerine bağlı olan NRF24L01 Wi-Fi haberleşme modülü ile veriler kablosuz şekilde Slave gönderilir. Slave yani STM32F429'un pinlerine bağlı olan NRF24L01 ile Master tarafından gönderilen veriler ve komutlar alınır. Bu veriler ve komutlar program tarafından değerlendirilerek bağlı olduğu çevre birimlerini (DC motor, servo motor, teker kızılötesi sensör, çizgi takip sensör vb.) yönetir. Şekil 3.4'de Araç Blok Yapısı verilmiştir.



Şekil 3.4: Araç Blok Yapısı

3.3.1 STM32F429 Discovery Kit

İvme hareketli aracımızın, gerekli komutlarını yerine getirmesini sağlayan STM32F4 ailesinden biri olan 32-bit STM32F429 Discovery modülü araç üzerinde yerleştirilmiştir.

Başlıca teknik özelliklerine değinecek olursak;

- 2 MB Flash bellek, LQFP144 paketinde 256 KB RAM bulunan STM32F429ZIT6 mikrodenetleyici,
- Seti bağımsız olarak kullanmak için seçim modu anahtarı bulunan araç ST-LINK / V2,
- ST-LINK / V2 (programlama ve hata ayıklama için SWD konnektörlü),
- Kart güç kaynağı: USB veri yolu üzerinden veya harici bir 3 Volt veya 5 Volt besleme gerilimi,
- 2.4 'QVGA TFT LCD,
- SDRAM 64 Mbits,

- L3GD20, ST MEMS hareket sensörü, 3 eksenli dijital çıkış jiroskop,
- Altı LED,
- USB iletişimi için LD1 (kırmızı / yeşil),
- LD2 (kırmızı) 3.3 Volt güç açıkken,
- İki kullanıcı LED'si: LD3 (yeşil), LD4 (kırmızı),
- İki USB OTG LED'i: LD5 (yeşil) VBUS ve LD6 (kırmızı) OC (aşırı akım),
- İki basma düğmesi (kullanıcı ve sıfırlama),
- Micro-AB konnektörlü USB OTG,
- Prototipleme kartına hızlı bir bağlantı sağlamak ve kolay programlama için LQFP144 I/O'lar için uzatma başlığı.

Şekil 3.5'de Araçta Kullanılan STM32F429 Discovery Modülü gösterilmiştir.

Şekil 3.5: Araçta Kullanılan STM32F429 Discovery Modülü

3.3.2 DC Motor ve Sürücüsü

RC aracımıza uygun model olan yüksek performanslı Tamiya'nın fırçalı 27T modeli seçilmiştir.

Başlıca teknik özellikleri;

- Çalışma gerimi 7,2 Volt,
- Yüksüz devir 17.000 Rpm,
- En iyi verimlilikte tork 37,24 nM.m,
- Değiştirilebilir fırçalar.

Şekil 3.6'da Araçta Kullanılan DC Motor gösterilmektedir.

Şekil 3.6: Araçta Kullanılan DC Motor

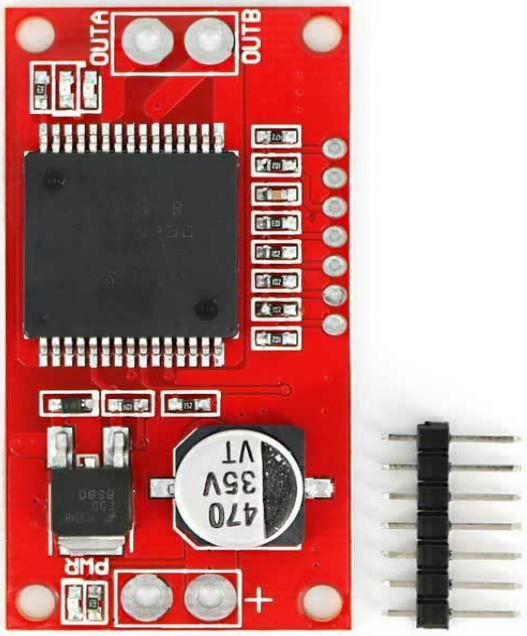
Aracımızda DC motor sürücüsü olarak VNH2SP30 H-köprülü MOSFET sürücü modülü seçilmiştir. Bu modülü seçmemizdeki temel neden 30 Ampere kadar çıkış verebilmesi ve termal kapatma özelliğine sahip olmasıdır.

Başlıca teknik özelliklerine değinecek olursak;

- Maksimum voltaj: 16 Volt.
- Maksimum akım derecelendirmesi: 30 Amper.
- Pratik Sürekli Akım: 14 Amper.

- Analog pin için mevcut akım algılama,
- MOSFET dayanıklılığı: 19 mΩ (bacak başına).
- Maksimum PWM frekansı: 20 kHz.
- Termal kapatma,
- Düşük Gerilim ve Aşırı Gerilim Kapatma.

Şekil 3.7'de Araçta Kullanılan DC Motor Sürücü Kartı gösterilmektedir.



Şekil 3.7: Araçta Kullanılan DC Motor Sürücü Kartı

3.3.3 Servo Motor

Aracımızın yön kontrolünde kullanılan HD-6001HB servo motor modeli seçilmiştir. Başlıca teknik özellikleri aşağıda yer almaktadır.

- Çalışma Gerilimi 4,8 Volt,
- Kontrol Sistemi: PWM.
- Tork: 5.8kg/cm.
- Hız: 0.16sn/60°.

Şekil 3.8'de Araçta Kullanılan Servo Motor gösterilmektedir.

Şekil 3.8: Araçta Kullanılan Servo Motor

3.3.4 NRF24L01 Wi-Fi Haberleşme Modülü

Nordic firmasınca geliştirilen NRF24L01 2.4 GHz kablosuz haberleşme modülü, 2.4GHz frekansında kablosuz haberleşme yapmanıza imkan sağlayan düşük güç tüketimine sahip modüldür. Çeşitli hobi, robotik ve endüstriyel projelerde sıklıkla kullanılabilecek 2MBps haberleşme hızına sahip olup, SPI arabirimini destekler (WEB_4).

Başlıca teknik özelliklerine değinecek olursak;

- 2.4GHz bandında yayın yapabilir.
- 250KBps, 1MBps ve 2MBps gibi hızlarda haberleşme hızı seçilebilir.
- Gelişmiş ShockBurst™ hızlandırma protokolünü desteklemektedir.
- Ultra düşük güç tüketimi,
- Çalışma Voltajı: 1.9-3.6 Volt.
- IO Portları Çalışma Voltajı: 0-3.3 Volt / 5 Volt.

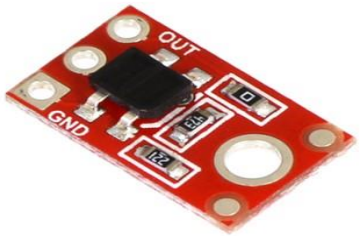
- Verici Sinyal Gücü: +7 dB.
- Alıcı Hassasiyeti ≤ 90dB.
- Haberleşme Mesafesi: Açık Alanda 250m.
- Boyutları: 15x29mm.

3.3.5 QTR-1RC Kızılötesi Sensör

Aracımızın aldığı yolu hesaplayabilmek için Autocad programında teker çapında bir daire çizilmiştir. Bu dairenin içine 25 adet siyah-beyaz eş parça oluşturulmuştur. Bu içi dolu daire kuşe kağıda çıktı alınıp aracın sol arka tekerine monte edilmiştir. Bu siyah-beyaz eş parçaları saymak için QTR-1RC Kızılötesi Sensör uygun görülmüştür. Başlıca teknik özellikleri aşağıda yer almaktadır;

- Boyutlar: 0,3"x 0,5" x 0,1" (isteğe bağlı başlık pimleri takılı olmadan).
- Modül üzerinde 1 adet kızılötesi sensör bulunmaktadır.
- Çalışma gerilimi: 5 Volt.
- Besleme akımı: 17 mA.
- Optimum algılama mesafesi: 0.125" (3 mm).
- Maksimum tavsiye edilen algılama mesafesi: 0,375" (9,5 mm).

Şekil 3.9'da Araçta Kullanılan QTR-1RC Kızılötesi Sensörü gösterilmektedir.



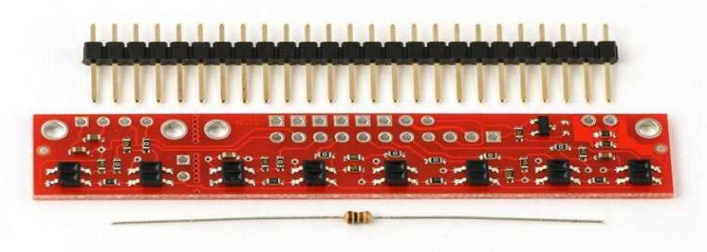
Şekil 3.9: Araçta Kullanılan QTR-1RC Kızılötesi Sensörü

3.3.6 QTR-8RC Kızılötesi Sensör

Tezin giriş bölümünde belirttildiği üzere aracın gerçek dünyada metrocar araç için ayrılmış özel tek şeritli yolda ivme hareketli şekilde yol alacağı için aracın şeritten çıkmaması adına şerit takip özelliği yani çizgi takip özelliği eklenmiştir. Bu özelliği araca kazandırabilmek için QTR-8RC Kızılötesi Sensörü tercih edilmiştir. Başlıca teknik özellikleri aşağıda yer almaktadır;

- Boyutlar: 2.95" x 0.5" x 0.125" (Header Hariç).
- Modül üzerinde 8 adet kızılötesi sensör bulunmaktadır.
- Çalışma voltajı: 3.3-5.0 Volt.
- Besleme akımı: 100 mA.
- Optimum algılama mesafesi: 0.125" (3 mm).
- Maksimum tavsiye edilen algılama mesafesi: 0,375" (9,5 mm).

Şekil 3.10'da Araçta Kullanılan QTR-8RC Kızılötesi Sensörü gösterilmektedir.



Şekil 3.10: Araçta Kullanılan QTR-8RC Kızılötesi Sensörü

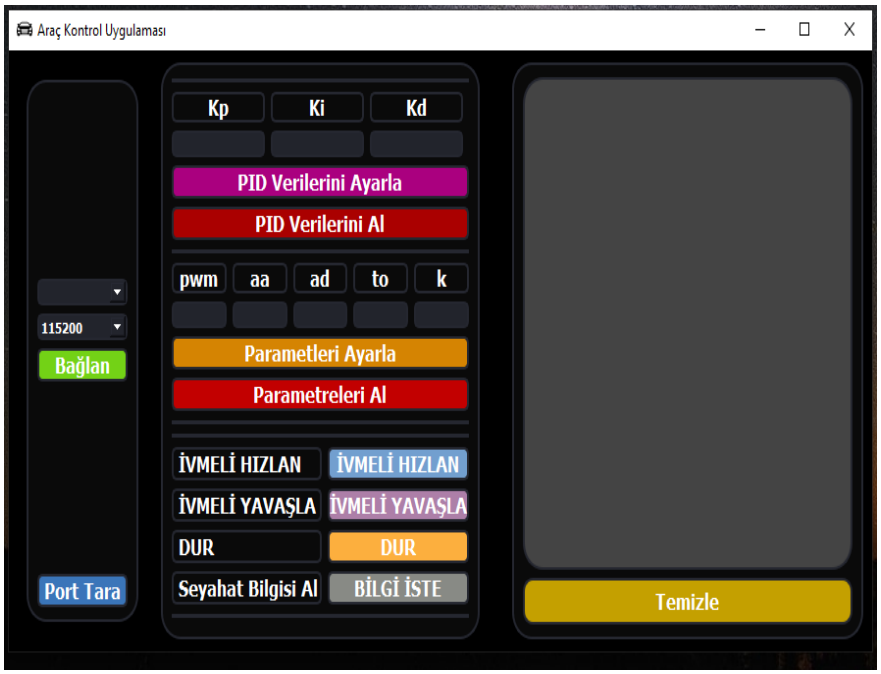
4. YAZILIM

4.1 Kullanıcı Arayüz Yazılımı

Kullanıcı arayüzü, nesne tabanlı Python programlama dili ile PyQt5 grafiksel kullanıcı arayüzü kütüphanesi kullanılarak geliştirilmiştir. Python, nesne yönelimli, modüler, etkileşimli, yorumlamalı ve daha da önemlisi platformdan bağımsız (Mac, Windows, Linux,) bir dil olduğu için tercih edilmiştir. Kullanıcı arayüzünde kullanılan PyQt5 kütüphanesi, QT'nin çok gelişmiş grafiksel kullanıcı arayüz geliştirme ortamı sunması ve yine Python gibi platformdan bağımsız çalışabilmesi nedeniyle tercih edilmiştir.

Öncelikle STM32F407 Discovery Modülü bir USB-UART dönüştürücü modülü ile bilgisayara bağlanır. Bağlantı yapıldıktan sonra Araç Kontrol Uygulaması açılır, sonrasında COM listesinden ilgili COM portu seçilerek "Bağlan" butonu ile bağlantı yapılır. Bağlantı başarılı ise "Bağlan" butonu "Kapat" olarak değişir ve diğer butonlar aktif duruma geçer.

Başarılı bir bağlantı yapıldıktan sonra, eğer ki aracın gücü bu bağlantıdan önce açıksa PID ve Parametre alanları otomatik şekilde aracın içindeki varsayılan değerler alınır ve alanlar doldurulur değilse aracın gücü açılır ve "Parametreleri Al" butonu ve "PID Verilerini Al" butonu ile ilgili alanlar araçtan talep edilerek doldurulur.
Şekil 4.1'deAraç Arayüz Görüntüsü verilmiştir.



Şekil 4.1: Araç Arayüz Görüntüsü

Varsayılan değerler araçtan alındıktan sonra, gerek duyulması halinde ilgili alanlara yeni değerler girilerek "PID Verilerini Ayarla" veya "Paremetreleri Ayarla" butonu ile aracın içindeki değerler güncellenir.

Arayüzde yer alan parametreler şu şekildedir.

- PID Parametreleri
  - Kp: Oransal Sabit
  - Ki: İntegral Sabiti
  - Kd: Türev Sabiti
- İvme Parametreleri
  - PWM: Araç Sabit Hız Değeri
  - aa: Araç Hızlanma İvmesi (m/s^2)
  - ad: Araç Yavaşlama İvmesi (m/s^2)
  - τ: Zaman Sabiti (100ms)
  - k: Teker Üzerindeki Bir Siyah-Beyazın Yol Karşılığı (0,00832m)

Parametreler ayarlandıktan sonra aracımızı “İVMELİ HIZLAN”, “İVMELİ YAVAŞLA” ve “DUR” butonları ile hareketine başlatıp durdurabiliriz. Aracımızın seyahat bilgisini “BİLGİ İSTE” butonu ile araçtan alıp görüntüleyebiliriz.

Kullanıcı arayüz programın açık kaynak kodunun bir kısmı Ek-1’de yer almaktadır.

4.2 Yazılım İçerisindeki Formüller

Araç içerisinde iki farklı formül hesaplaması yapılmaktadır. İlk formül hesaplaması, araç kontrol uygulaması üzerinden girilen değerler ile referans hız ve referans konum değerleri hesaplatılmaktadır. Diğer formüller ile aracın pist üzerinde almış olduğu yol, hız ve ivme bilgileri hesaplatılmaktadır.

Araç kontrol uygulaması üzerinden girilen hızlanma ve yavaşlama ivmelerine göre referans hızın ve referans konumun belirlendiği formüller aşağıdaki gibidir.

$$V_{Referans}[n+1] = V_{Referans}[n] + aa * \tau \quad (1.1)$$

Burada referans alınan hız formülünde τ =0,1 ms sabit, ilk hız olarak da $V_{Referans}$ [0]=0 m/s belirlenmiştir. Burada hızlanma ivmesi olan aa değeri araç kontrol uygulaması üzerinden kullanıcının giriş yapmasına bırakılmıştır. Bütün değerleri girildikten sonra araç kontrol uygulaması üzerindeki ivmeli hızlan butununa basıldığında araç sabit ivmeli şekilde ilerleyecektir. Belirli bir hıza ulaştıktan sonra araç kontrol uygulaması üzerinden ivmeli yavaşla butonuna basıldığında $V_{Referans}$ formülü aşağıdaki gibi hesaplanmaktadır.

$$V_{Referans}[n+1] = V_{Referans}[n] - ad * \tau \quad (1.2)$$

Bu değerler hesaplandıktan sonra ilgili konum referans formülüne geçilmektedir.

$$X_{Referans}[n+1] = X_{Referans}[n] + V_{Referans}[n] * \tau \quad (1.3)$$

Burada aracın ilk hıza karşılık gelen konumu $X_{Referans}$ [0]=0 m’dir

Aracın pist üzerindeki hız ve konum bilgisi aşağıdaki denklemler yardımıyla hesaplanır.

Araç yazılımını oluştururken, aracımızın aldığı yol (X) üzerinden hesaplamalar yapılmıştır. Bunu hesaplarken şu adımlar uygulanmıştır.

- Aracın sol arka tekerinde 25 adet Siyah – Beyaz şerit bulunmaktadır.
- Teker çapı hassas kumpas ile ölçülmüştür ve R=6,62084cm bulunmuştur.
- Tekerin 1 turda aldığı yol bulunursa;
- $ÇEVRE = 2\pi r = \pi R = \pi * 6{,}62084 = 20{,}7998\ cm$ Olarak bulunur

Aracın sol arka tekerindeki sensör okuduğu siyah beyaz sayısını vereceği için 1 Siyah – Beyaz sayısında kaç santim ya da kaç metre olduğunu bulmak gerekecektir. Bu değere k sabiti adı verilmiştir.

$$k = \frac{ÇEVRE}{25} = \frac{0{,}207998}{25} = 0{,}0083199 \cong 0{,}00832\ m \tag{1.4}$$

Bu değerler ışığında aracın aldığı yol, hız ve ivme aşağıdaki gibi hesaplanır.

$$X = sb * k = (sb[n] - sb[n-1]) * k \tag{1.5}$$

$$V = \frac{X[n] - X[n-1]}{\tau} = \frac{X[n] - X[n-1]}{0{,}1} \tag{1.6}$$

$$A = \frac{V[n] - V[n-1]}{\tau} = \frac{V[n] - V[n-1]}{0{,}1} \tag{1.7}$$

4.3 Kontrolcü Yazılımı

Kontrolcü yazılımı, kullanıcı arayüzünden alınan verilerin ve komutların USB-UART dönüştürücü vasıtasıyla STM32F407 Discovery kartına iletilmesi ve araçtan gelen verilerinde tekrar kullanıcı arayüzüne iletilmesinde görev yapmaktadır. Gönderilen ve alınan, veri ve komutlar NRF24L01 Wi-Fi modülü ile sağlanmaktadır.

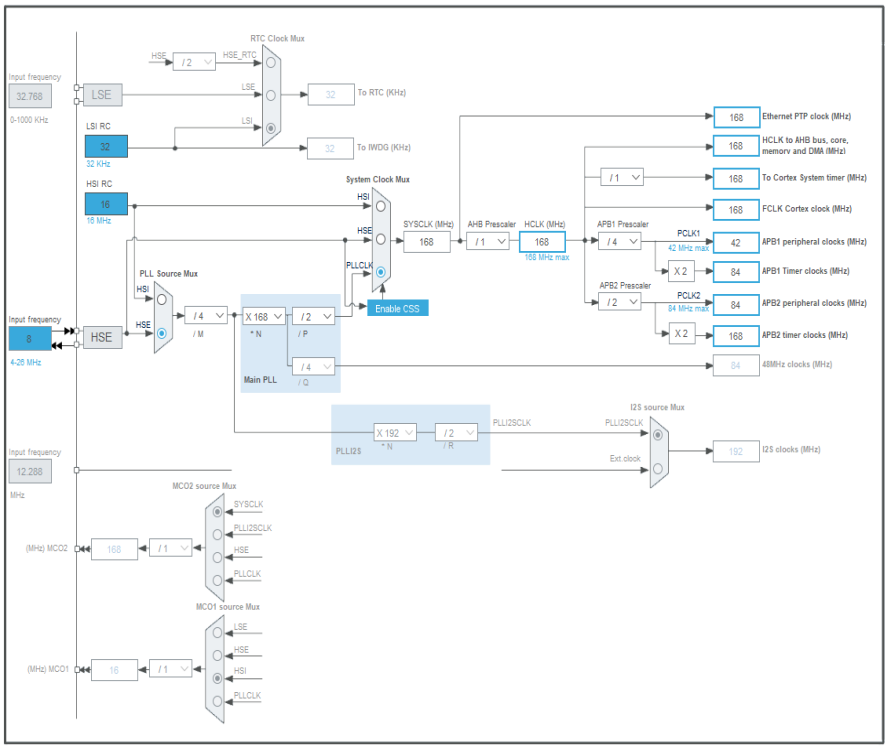
Geliştirilen yazılım Stmicroelectronics firmasının STM32 için sağlamış olduğu ücretsiz Entegre Geliştirme Ortamı (Integrated Development Environment) olan STM32CubeIDE kullanılarak geliştirilmiştir.

4.3.1 STM32F407 Mikrodenetleyici Kit Ayarları

Kullanıcı arayüzünden, kontrolcüye iletilecek olan bilgi USB-UART dönüştürücü modülü ile alabilmek için mikrodenetleyicinin UART çevre birimi, NRF24L01 Wi-Fi modülü ile araca iletilecek ve araçtan alınacak bilgi için mikrodenetleyicinin SPI haberleşme çevre birimi, çip seçim pini ve çip aktif pinleri bazı GPIO yapılandırmaları yapılmıştır. Haberleşme çevre birimleri aktif yapılmadan önce sistemin saat (clock) yapılandırması yapılarak haberleşme hızları bu saat yapılandırmasına uygun seçilmiştir.

4.3.1.1 Saat Yapılandırması

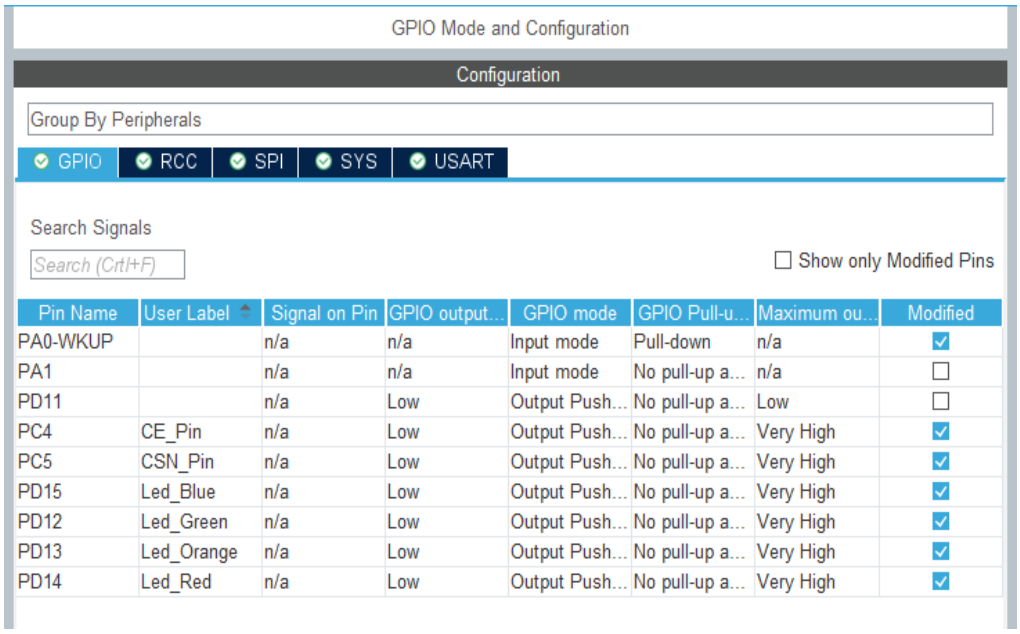
Saat yapılandırmasında STM32F407 Discovery kartı üzerinde bulunan 8 MHz kristal osilatör dikkate alınarak PLL kaynağı olarak HSE (High Speed External) seçilmiştir. Mikrodenetleyicimizin çıkabileceği en yüksek çalışma frekansı olan 168Mhz, HCLK parametre olarak verilerek, en uygun saat ayarlaması otomatik olarak yapılmıştır.



Şekil 4.2: STM32F407'nin Saat Yapılandırma Ayarı

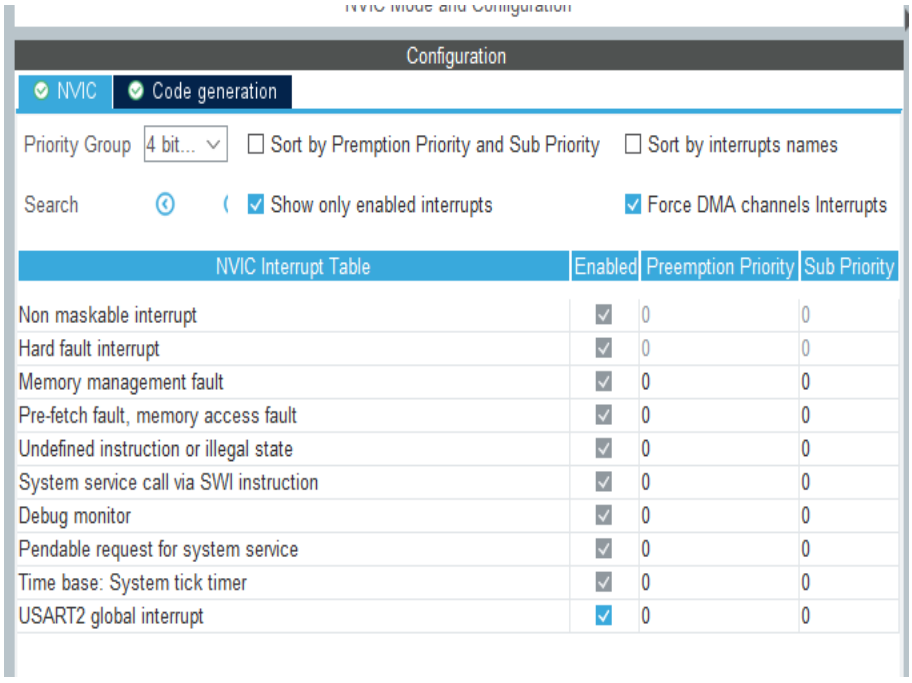
4.3.1.2 GPIO ve NVIC Ayarları

NRF24L01 Wi-Fi modülünün SPI haberleşmesini başlatabilmek için gerek duyduğumuz CE (Chip Enable) ve CSN (Chip Select Not) GPIO yapılandırmaları yanı sıra ihtiyaç dahilinde kullanılmak üzere STM32F407 Discovery kartı üzerinde bulunan kullanıcı ledleri ve butonları da yapılandırılmıştır.



Şekil 4.3: STM32F407'nin GPIO Ayarları

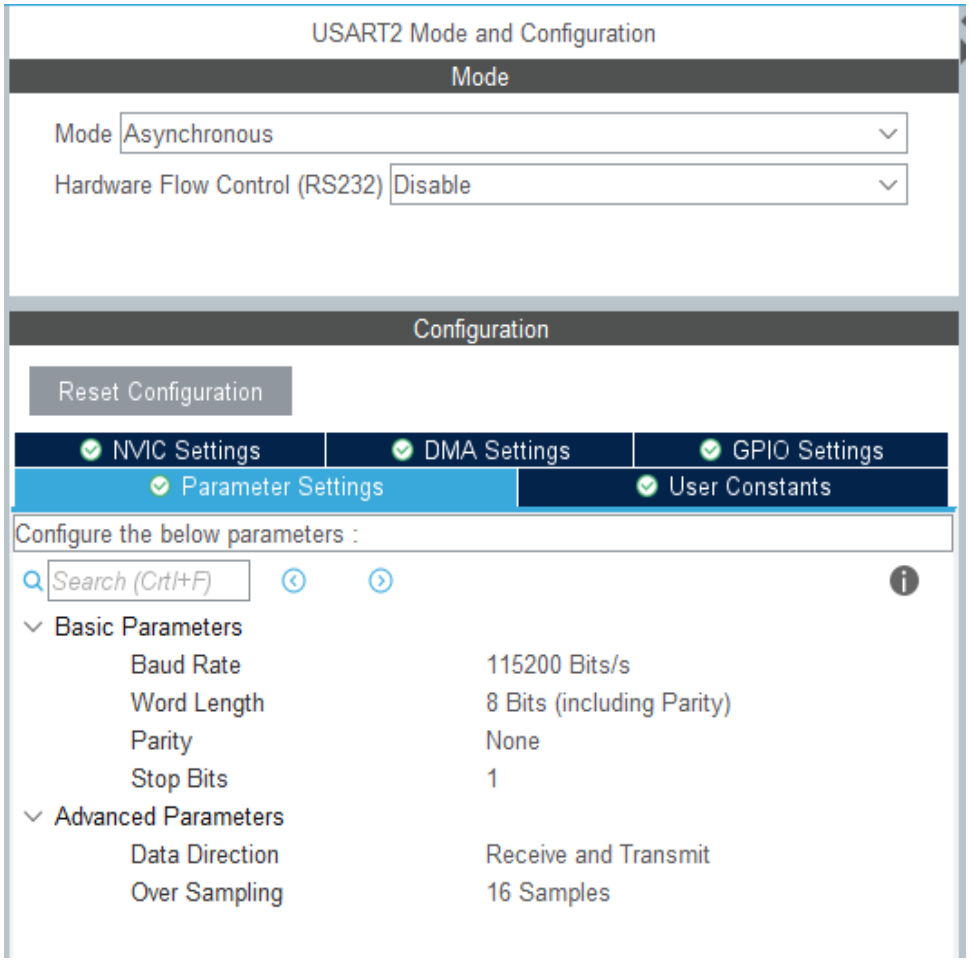
UART çevre birimlerinin yapılandırmasından sonra, UART global kesmesi, kullanıcı arayüzünden gelecek verilerin kayba uğramaması için aktif hale getirilmiştir. Böylece kullanıcı ara yüzünden veri ya da komut geldiğinde kesme gerçekleşecek ve araç yazılımına bu bilgiler iletilecektir.



Şekil 4.4: STM32F407'nin NVIC Ayarları

4.3.1.3 UART Haberleşme Ayarları

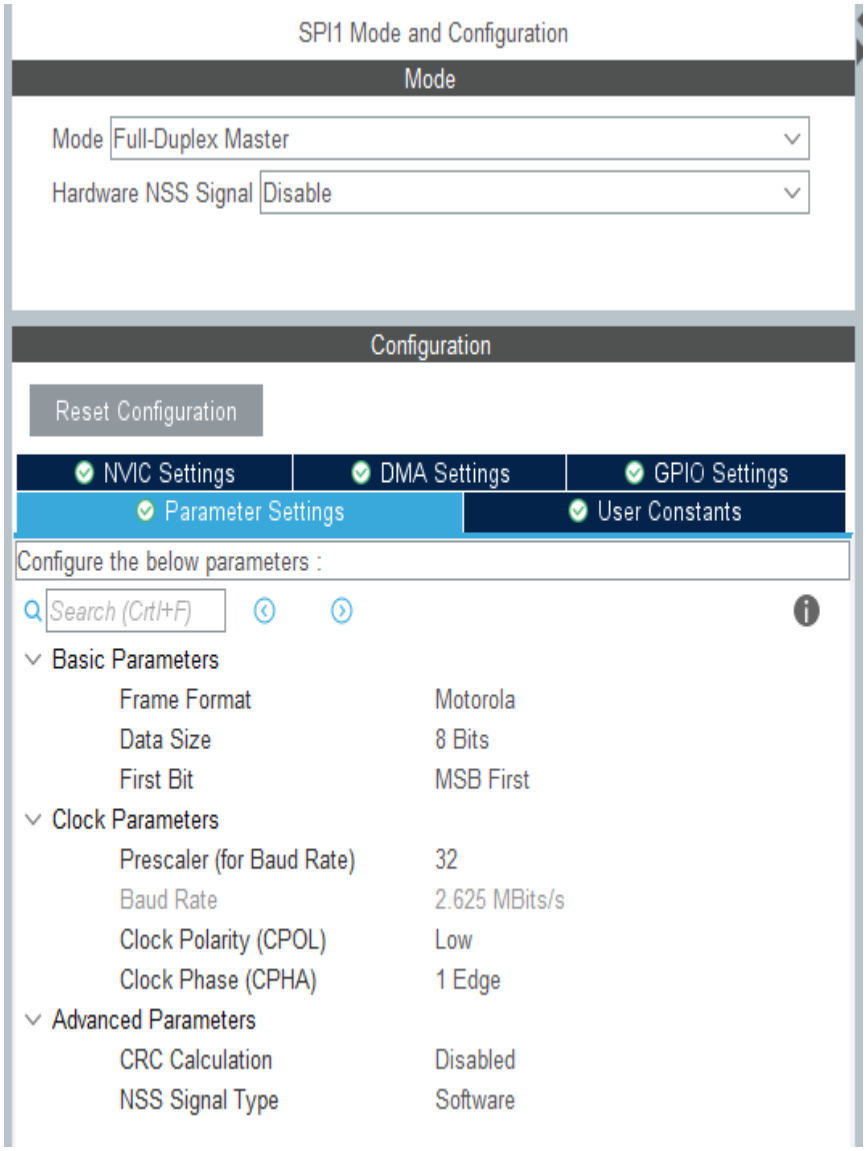
Kullanıcı arayüzünden, kontrolcüye iletilecek olan veri ve komutları USB-UART dönüştürücü modülü ile alabilmek için mikrodenetleyicinin UART çevre birimi olarak USART2 seçilmiş olup, veri iletiminde asenkron modda, 115200 Bit/s haberleşme hızına sahip 8 bit veri uzunluğu ve 1 bit stop biti olacak şekilde yapılandırılmıştır.



Şekil 4.5: STM32F407'nin UART Haberleşme Ayarları

4.3.1.1 SPI Haberleşme Ayarlar

NRF24L01 Wi-Fi modülü için mikrodenetleyicinin SPI haberleşme çevre birimi, NRF24L01'in veri sayfasından alınan bilgiler doğrultusunda gerçekleştirilmiştir.



Şekil 4.6: STM32F407'nin SPI Haberleşme Ayarları

4.3.2 Kontrolcü Yazılımı Açıklaması

STM32F407 Discovery kartı üzerinde bulunun mikrodenetleyicimiz sonsuz döngünün içinde kullanıcı arayüzünden veri ya da komut iletildiğinde "flag" değeri '1' olmakta ve gelen bilgiyi "setMessage()" fonksiyonu NRF24L01 Wi-fi modülü ile aracımıza iletmektedir. Ayrıca sonsuz döngümüzün içinde araçtan iletilen veri ve komutlar NRF24L01 Wi-Fi modülünden alındığında aktif olan "NRF24_available()"

fonksiyonu tespit edilip “uartSend()” fonksiyonu ile kullanıcı arayüzüne iletilmektedir.

```
while (1)
{
	if (flag == 1)
	{
		printf(UartRxBuffer);
		setMessagge(UartRxBuffer);
	}
	if (NRF24_available())
	{
				char string[32] = "";
				NRF24_read(string, 32);
				uartSend(string);
				//printf(string);
	}
}
```

Kullanıcı arayüzden iletilen veriler 1 bayt şeklinde HAL_UART_RxCpltCallback kesme fonksiyonu ile alınmaktadır. Gelen bilgi ‘\r\n’ içeriyorsa flag değeri “1” yapılarak NRF24L01 Wi-Fi modülü ile araca gönderilmek üzere hazır komut ya da veri olduğunu bildirilmektedir.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
if (huart->Instance == USART2) {
	UartRxBuffer[strlen(UartRxBuffer)] = UART_RxData[0];
	HAL_UART_Receive_IT(&huart2, (uint8_t*) UART_RxData, 1);
	if (strstr((char*) UartRxBuffer, "\r\n") != NULL)
		flag = 1;
	}
}
```

4.4 Araç Yazılımı

Araç yazılımı, kullanıcının iletmiş olduğu komutlar araç üzerinde bulunan STM32F429 Discovery kartına bağlı NRF24L01Wi-Fi modülü ile alınmaktadır. Alınan komutlar değerlendirilerek gerekli aşama araç tarafından alınmaktadır.

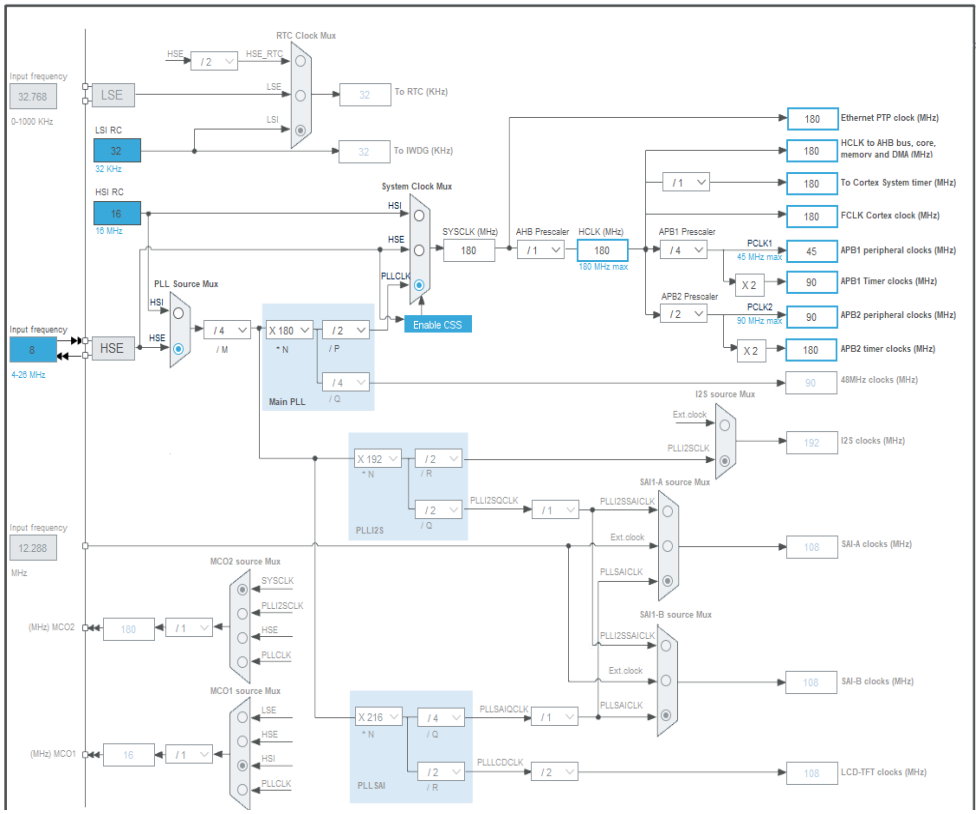
Geliştirilen yazılım Stmicroelectronics firmasının STM32 için sağlamış olduğu ücretsiz Entegre Geliştirme Ortamı (Integrated Development Environment) olan STM32CubeIDE kullanılarak geliştirilmiştir.

4.4.1 STM32F429 Discovery Kit

Kontrolcü tarafından iletilecek olan veri ve komutları NRF24L01 Wi-Fi modülü ile alınacağı için mikrodenetleyicimizin SPI haberleşme çevre birimi, şerit takibinde kullanılacak olan Çizgi Takip Sensör (QTR-8RC), aracın aldığı yolu hesaplamada kullanılacak teker Sensor (QTR-1RC) , servo ve DC motor kontrollerinde kullanılacak olan pinlerin GPIO yapılandırmaları, servo ve DC motorların kontrolünde kullanılacak PWM ayarları ve zamanlayıcıda kullanılacak zamanlayıcıların yapılandırmaları bu bölümde anlatılacaktır.

4.4.1.1 Saat Yapılandırması

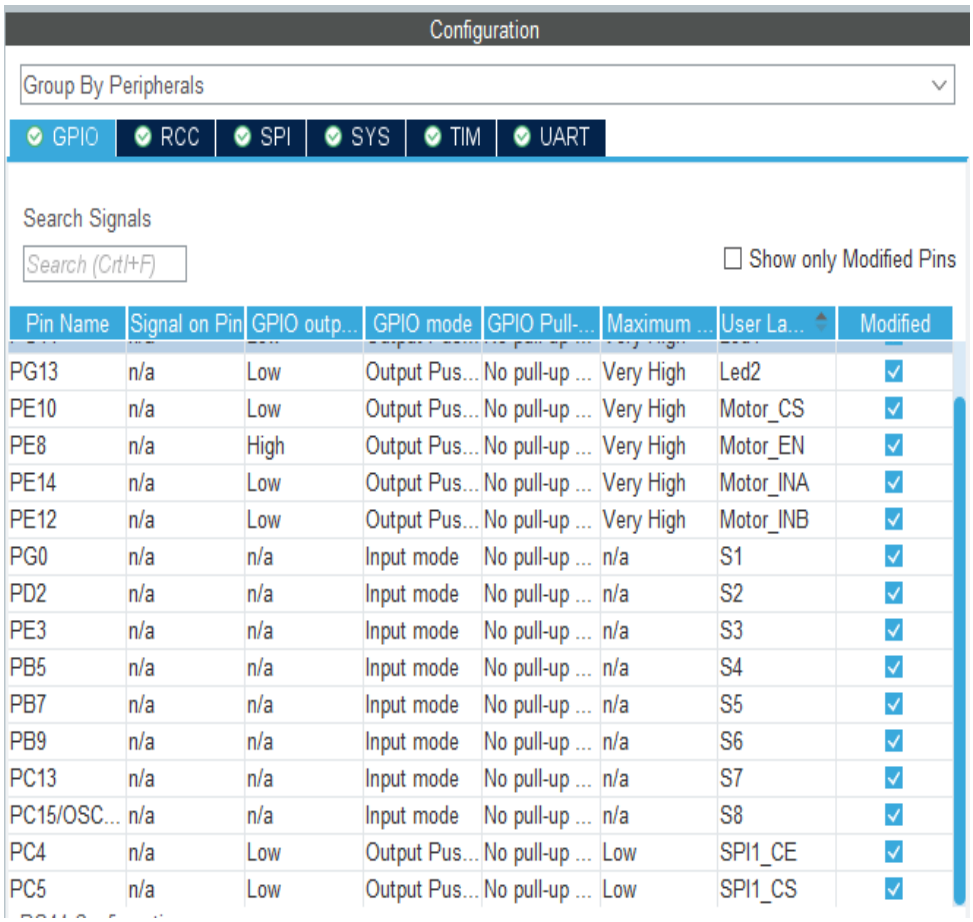
Saat yapılandırmasında STM32F429 Discovery kartı üzerinde bulunan 8 MHz kristal osilatör dikkate alınarak PLL kaynağı olarak HSE (High Speed External) olarak seçilmiştir. Mikrodenetleyicimizin çıkabileceği en yüksek çalışma frekansı olan 180Mhz, HCLK' parametre olarak verilerek, en uygun saat ayarlaması otomatik olarak yapılmaktadır.



Şekil 4.7: STM32F429'un Saat (Clock) Yapılandırma Ayarı

4.4.1.2 GPIO ve NVIC Yapılandırması

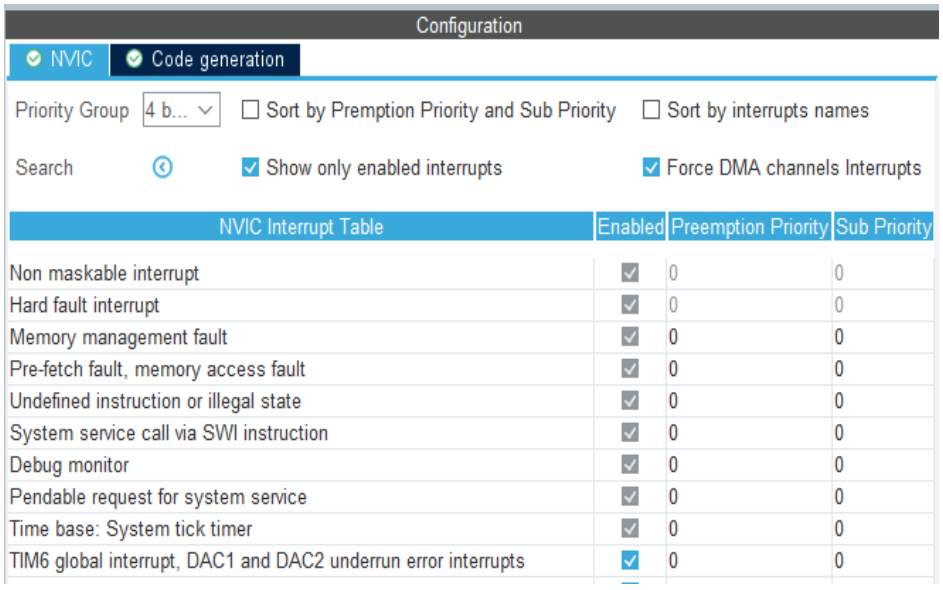
DC motor sürücüsü olarak VNH2SP30 kullandığımız modülün CS, EN, INA, INB pinlerinin bağlantı yapılacağı pinler çıkış modunda, çizgi takibinde kullanacağımız sensörün pinleri S1, S2, S3, S4, S5, S6, S7, S8 giriş modunda ve NRF24L01 Wi-fi modülünün SPI haberleşmesini başlatabilmek için gerek duyduğumuz CE (Chip Enable) ve CSN (Chip Select Not) pinleri çıkış modunda yapılarak GPIO yapılandırmaları tamamlanmıştır.



Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO Pull-...	Maximum ...	User La...	Modified
PG13	n/a	Low	Output Pus...	No pull-up ...	Very High	Led2	☑
PE10	n/a	Low	Output Pus...	No pull-up ...	Very High	Motor_CS	☑
PE8	n/a	High	Output Pus...	No pull-up ...	Very High	Motor_EN	☑
PE14	n/a	Low	Output Pus...	No pull-up ...	Very High	Motor_INA	☑
PE12	n/a	Low	Output Pus...	No pull-up ...	Very High	Motor_INB	☑
PG0	n/a	n/a	Input mode	No pull-up ...	n/a	S1	☑
PD2	n/a	n/a	Input mode	No pull-up ...	n/a	S2	☑
PE3	n/a	n/a	Input mode	No pull-up ...	n/a	S3	☑
PB5	n/a	n/a	Input mode	No pull-up ...	n/a	S4	☑
PB7	n/a	n/a	Input mode	No pull-up ...	n/a	S5	☑
PB9	n/a	n/a	Input mode	No pull-up ...	n/a	S6	☑
PC13	n/a	n/a	Input mode	No pull-up ...	n/a	S7	☑
PC15/OSC...	n/a	n/a	Input mode	No pull-up ...	n/a	S8	☑
PC4	n/a	Low	Output Pus...	No pull-up ...	Low	SPI1_CE	☑
PC5	n/a	Low	Output Pus...	No pull-up ...	Low	SPI1_CS	☑

Şekil 4.8: STM32F429'un GPIO Ayarı

τ zamanlayıcısı olarak kullanacağımız TIM6 biriminin global kesmesi aktif hale getirilmiştir.

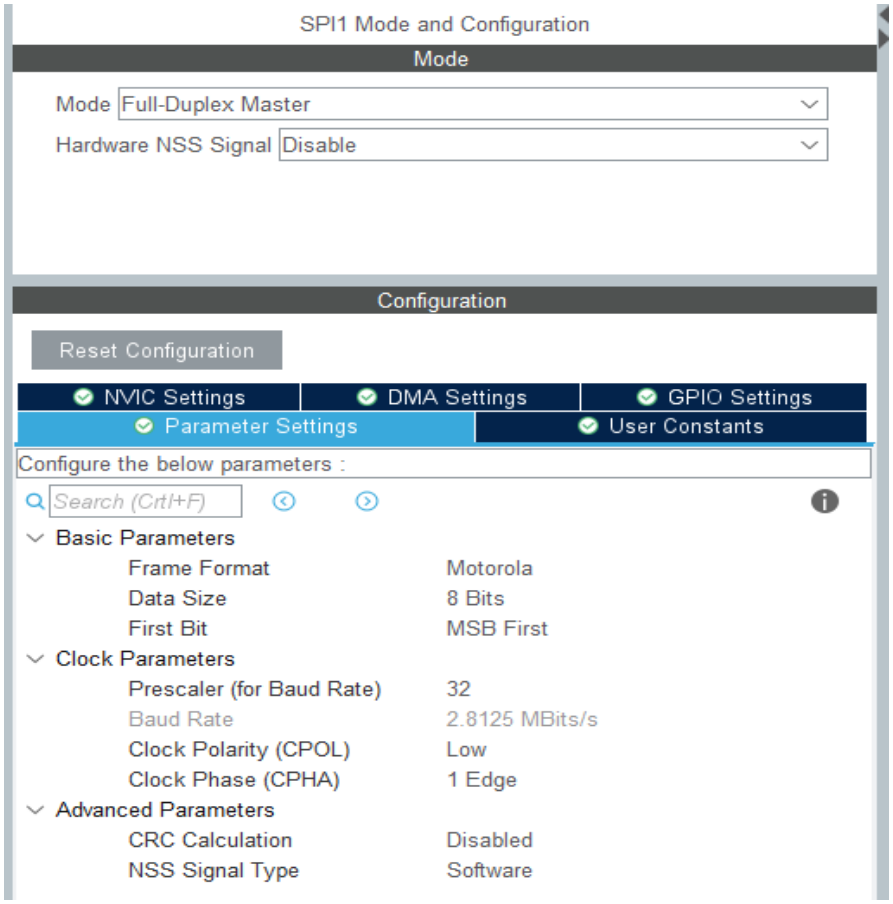


NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	☑	0	0

Şekil 4.9: STM32F429'un NVIC Ayarı

4.4.1.3 SPI Haberleşme Ayarlar

NRF24L01 Wi-Fi modülü için mikrodenetleyicimizin SPI haberleşme çevre birimi, NRF24L01'in veri sayfasından alınan bilgiler doğrultusunda gerçekleştirilmiştir.



Şekil 4.10: STM32F429'un SPI Haberleşme Ayarı

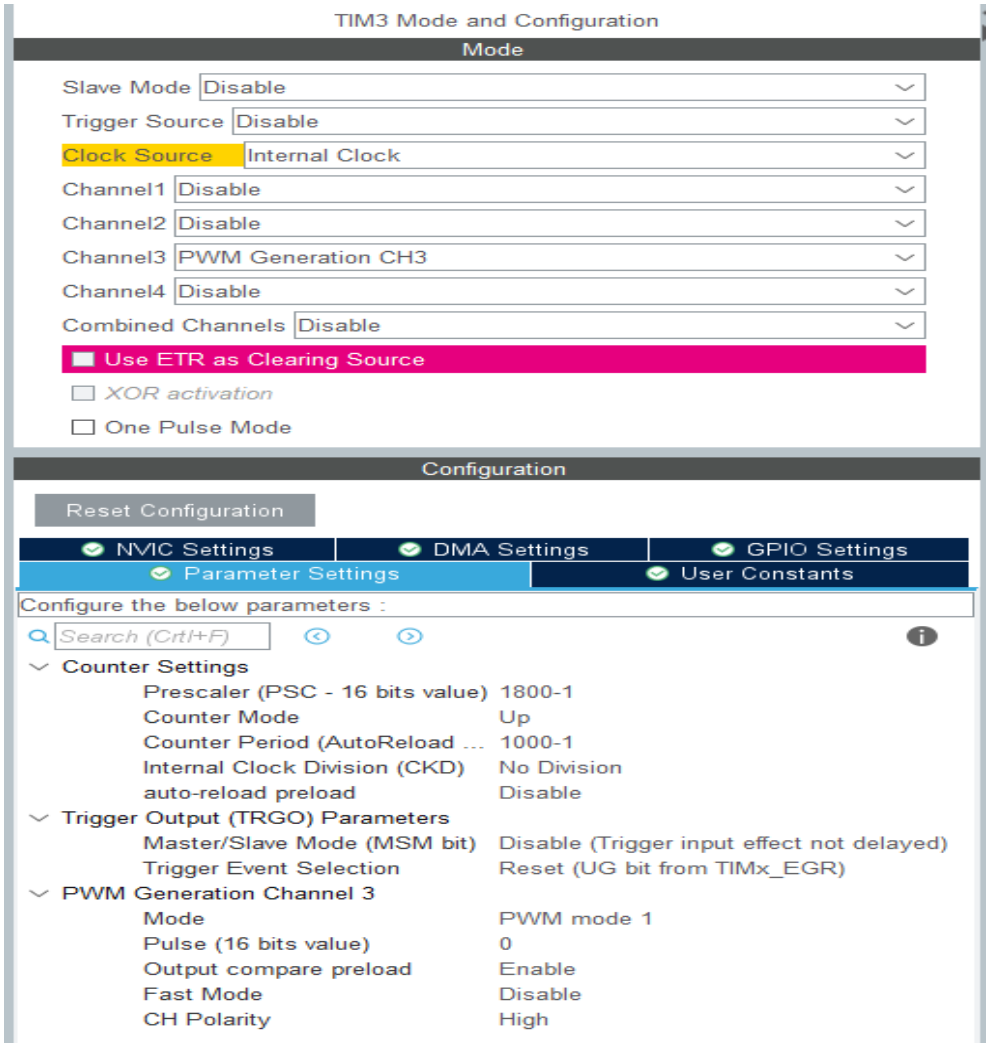
4.4.1.4 TIMER Ayarları

STM32F429 mikrodenetleyicinin gelişmiş zamanlayıcı özelliklerinden; PWM giriş modu (Input Capture) özelliğini kullanarak tekerleğimizin dönüş sayısını, genel amaçlı sayaç zamanlayıcı özelliklerinden; servo motoru ve DC motoru sürmek için darbe genişlik modülasyonu (PWM) özelliğinden, basit zamanlayıcı özelliğini kullanarak da 100 ms'de bir kesme üreten τ zamanlayıcımızda kullanılmıştır.

4.4.1.4.1 Servo Motor PWM Ayarları

Servo motorların kontrolünde PWM sinyaline ihtiyaç vardır. PCB dizaynını dikkate alarak bize en uygun noktada olan TIM3 zamanlayıcı biriminin kanal 3 PWM'i tercih edildi.

PWM sinyalimiz saat kaynağı olarak dahili saat seçildikten sonra servo motorumuzun çalışma frekansı olan 50 Hz ayarlayabilmek için öncelikle TIM3 zamanlayıcı birimimizin bağlı olduğu saat hattı, veri sayfasında APB1 olarak tespit edilmiştir. APB1 saat veri yolu hızı 90 Mhz, önbölücü (PSC Prescaler) değeri olan 1800 bölünerek 50 kHz zamanlayıcı çalışma frekansı elde edilmiştir. Otomatik yeniden yükleme kaydı (ARR Auto Reload Register) değeri de 1000 seçilerek 50 Hz PWM çalışma frekansı elde edilmiş oldu.

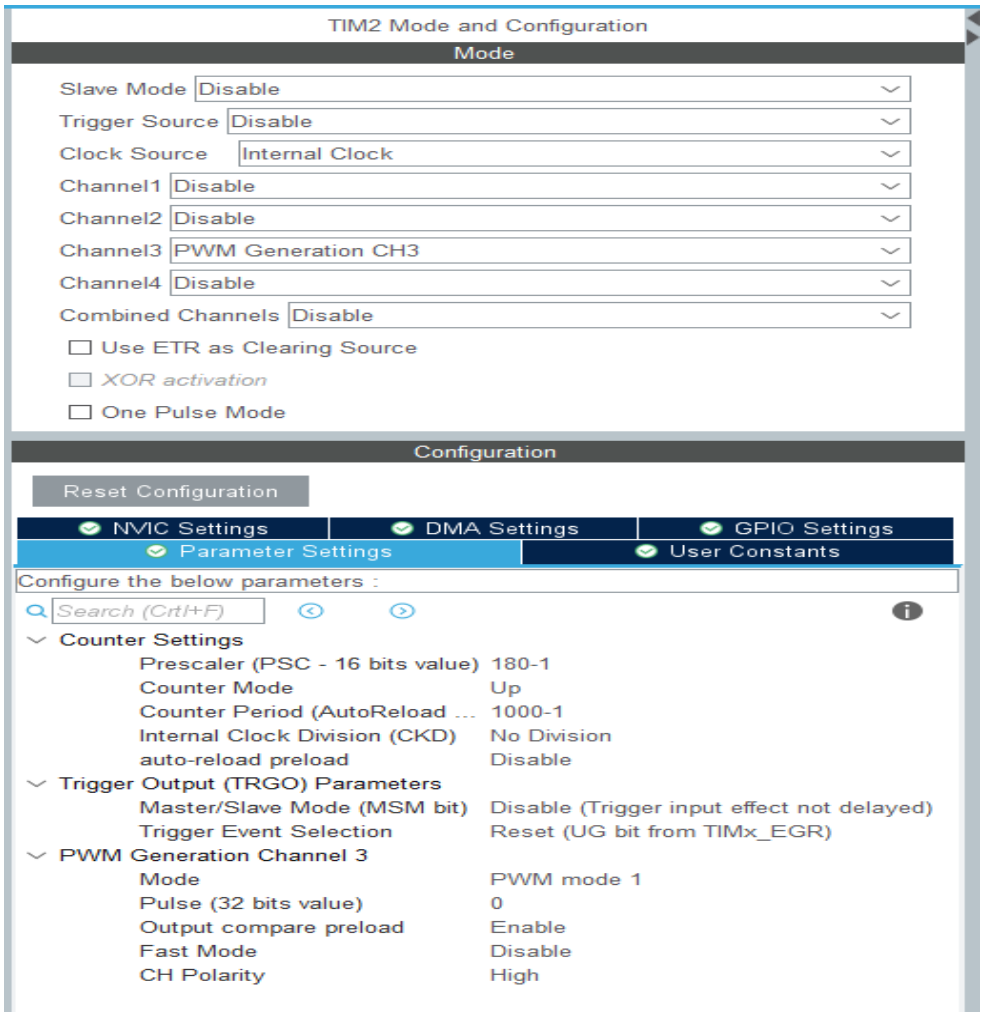


Şekil 4.11: Servo Motor PWM Ayarı

4.4.1.4.2 DC Motor PWM Ayarları

DC motorun kontrolünde PWM sinyaline ihtiyaç vardır. PCB dizaynını dikkate alarak en uygun noktada olan TIM2 zamanlayıcı biriminin kanal 3 PWM'i tercih edildi.

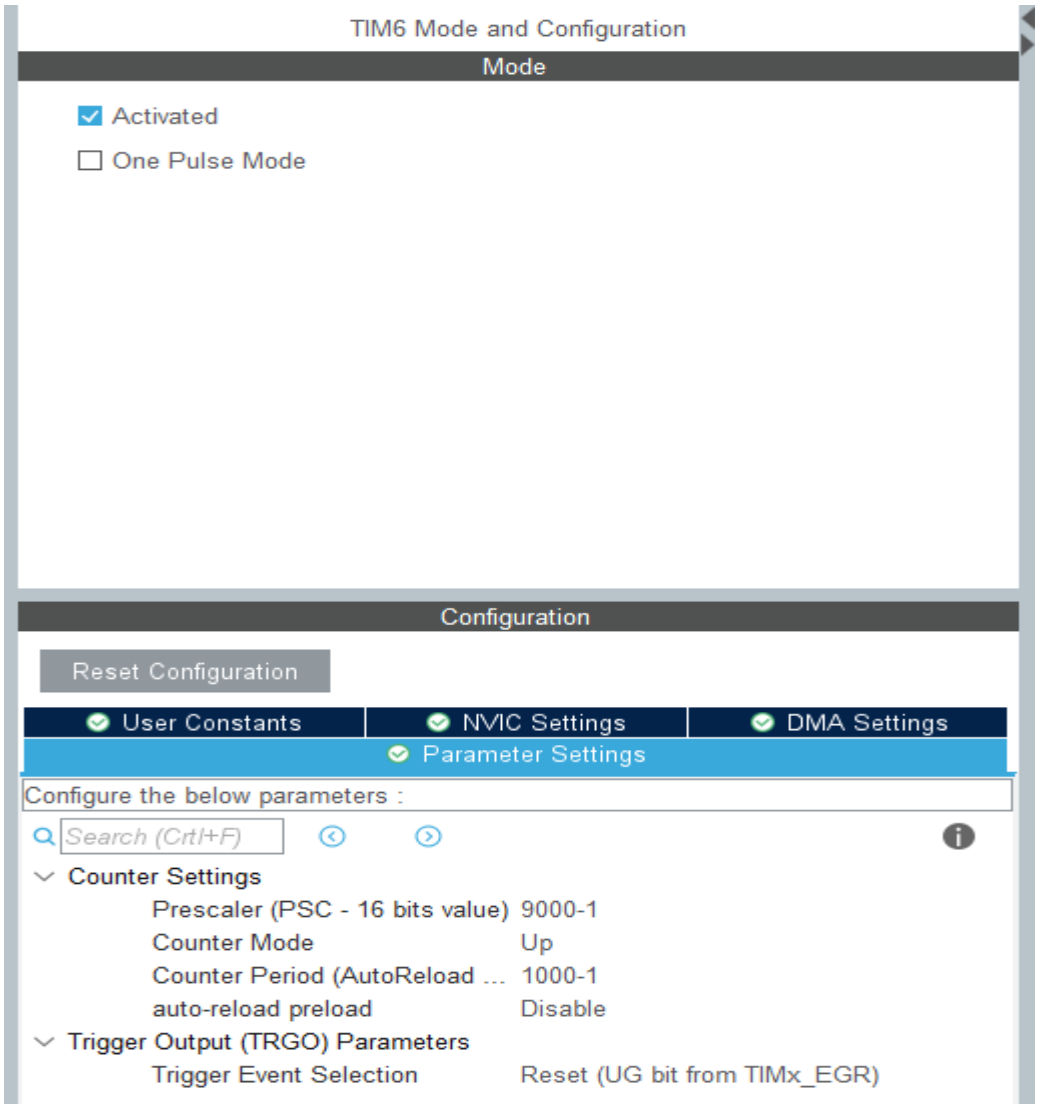
PWM sinyalimiz saat kaynağı olarak dahili saat seçildikten sonra DC motorumuzun en iyi performans vermiş olduğu çalışma frekansı 500 Hz frekans değerini elde edebilmek için TIM2 zamanlayıcı birimimizin bağlı olduğu saat hattı, APB1 önbölücü (PSC Prescaler) değeri olan 180 bölünerek 500 kHz zamanlayıcı çalışma frekansı elde edilmiştir. Otomatik yeniden yükleme kaydı (ARR Auto Reload Register) değeri de 1000 seçilerek 500 Hz PWM çalışma frekansı elde edilmiş oldu.



Şekil 4.12: DC Motor PWM Ayarı

4.4.1.4.3 τ Zamanlayıcısı Ayarları

τ zamanlayıcımızı basit zamanlayıcı biriminden TIM6'yı tercih ettik. τ zamanlama frekansımız olan 10 Hz ayarlayabilmek için TIM6 zamanlayıcı birimimizin bağlı olduğu saat hattı, veri sayfasında APB1 olarak tespit edilmiştir. APB1 saat veri yolu hızı 90 Mhz, önbölücü (PSC Prescaler) değeri olan 9000 bölünerek 10 kHz zamanlayıcı çalışma frekansı elde edilmiştir. Otomatik yeniden yükleme kaydı (ARR Auto Reload Register) değeri de 1000 seçilerek 10 Hz'lik yani 100 ms'lik bir periyot elde edilmiştir. TIM6 global kesmesi de aktif hale getirildiği için her 100 ms'de bir kesme meydana gelecek ve gerekli yol, hız, ivme gibi değerlerimiz bu fonksiyonun içinde hesaplancaktır.



Şekil 4.13: τ Zaman Ayarı

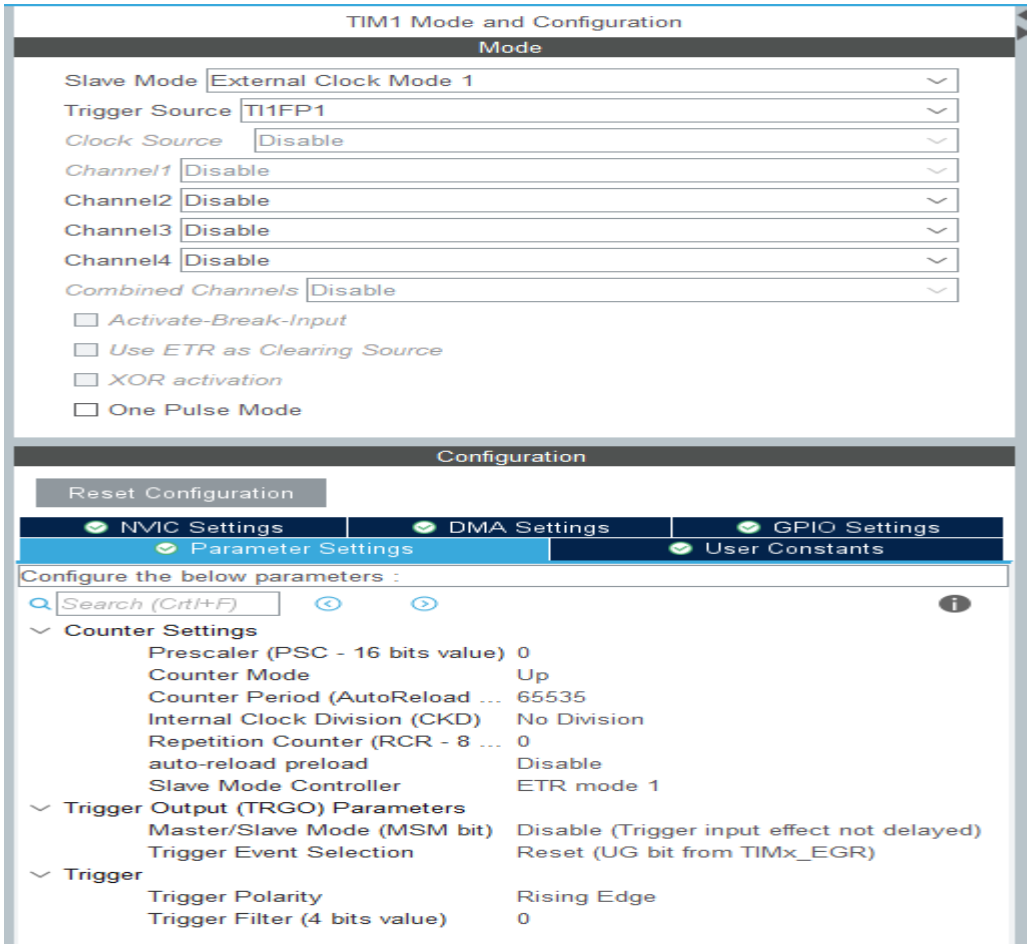
4.4.1.4.4 Teker Sensör Ayarları

Teker dönüş sayısını tespit edebilmek için teker üzerine monte edilen belirli bir aralıkla siyah beyaz döşenmiş zeminimizdeki her değişimi QTR-1RC sensörüyle PWM sinyal çıkışı elde edilmiştir. Bu çıkıştaki PWM sinyali, gelişmiş zamanlayıcı özelliği olan PWM giriş modu (Input Capture) özelliğini kullanarak tespit edilecektir. PCB dizaynına en uygun noktada olan TIM1 zamanlayıcı birimi tercih edilmiştir.

Giriş yakalama modunda yükselen kenar tercih edilmiştir. Her yükselen kenar oluştuğunda yakalama olayı meydana gelecektir ve mikrodenetleyicinin

*register*ındaki değer bir artacaktır. Her 100 ms'de bir bu değer alınarak aracın ne kadar siyah beyaz saydığı tespit edilmiştir.

Saat kaynağı olarak harici saat modu seçilmiş olup tetikleme kaynağı olarak TI1FP1 ve tetikleme yükselen kenar seçilmiştir.



Şekil 4.14: Kızılötesi Sensör Ayarı

4.4.1.4.5 Çizgi Takip Sensörü

QTR-8RC yansıma sensörü çizgi sensörü olarak kullanılıyor. QTR-8RC modülü üstünde 8 adet IR alıcı ve verici çifti var. QTR-8RC modülü kullanmak için öncelikle STM32F429 mikrodenetleyicinin bağlı olduğu pinler çıkış olarak yapılandırılarak yüksek (High) yapılır ve belirli bir süre sensör üzerinde bulunan kondansatörlerin dolması beklenir. Sonrasında STM32F429 mikrodenetleyicimize

bağlı olduğu pinler giriş olarak yapılandırılarak kondansatörlerin deşarj olma zamanları ölçülerek çizginin varlığı hangi sensörde olduğu tespit edilir.

Mikrodenetleyicinin çıkış pinleri PCB dizaynı dikkate alınarak en uygun pinler seçilmiş olup GPIO olarak yapılandırılmıştır. QTR-8RC modülün 8 adet IR sensörleri S1, S2, S3, S4, S5, S6, S7, S8 olarak isimlendirilerek program içersinde kullanılmıştır.

4.4.2 Araç Yazılımı Kodları

STM32F429 Mikrodenetleyicinin içindeki yazılım Ek-2'de verilmiştir.

5. DENEYSEL SONUÇLAR

Yapılan proje çalışması iki temel ana unsurdan oluşmaktadır. Ana unsurlar Master ve Slave başlıkları adı altında toplanmıştır. Master kısmı bir dizüstü bilgisayar, STM32F407 discovery kartı ve NRF24L01 Wi-Fi modülünden oluşmaktadır. Python programı ile Master ve Slave kısımlarını kontrol eden bir arayüz tasarlanmıştır. Bu arayüz vasıtası ile Slave kısmına yani RC aracı sabit hız komutu ile çalıştırabilmekte ayrıca hızlanma ivmesi (aa) ve yavaşlama ivmesi (ad) girilerek aracın ivmeli olarak hareketlenmesi ya da ivmeli olarak yavaşlaması sağlanmıştır. Slave kısmı bir RC model araç ve bu araca bağlı STM32F429 discovery kartı ve bu karta bağlı DC motor, DC motor sürücü kartı, Servo motor, aracın tek şeritlik yolda şerit takibi yapabilmesi için QTR-8RC Kızılötesi sensör ve bu aracın aldığı yolu bulmak için tekere yerleştirilen QTR-1RC Kızılötesi sensör ayrıca tüm bu verileri Master kısmına aktarabilmek için NRF24L01 Wi-Fi modünden oluşmaktadır.

Yapılan çalışmada seyehat eden aracın yolu ayrı ve tek şerit olarak düşünülmüştür. Ayrıca bu yol üzerindeki aracın insansız kabul edilerek değerlendirmeleri yapılmıştır

Yapılan proje çalışmasında Metrocar Araçların Sabit İvmeli Hareketi konusunda 10 kat oranında küçültülmüş RC model araç ile gene 10 kat boyutunda küçültülmüş parkurlar hazırlanarak tek araç ile defalarca kez ölçümler alınıp değerlendirmeleri yapılmıştır.

Aracın hız sınır değerlerini belirleyebilmek için sabit PWM ile aracımız pistte birçok kez çalıştırılmış olup aşağıdaki Tablo 5.1 elde edilmiştir.

Tablo 5.1: Hız Sınır Değerleri Tablosu

PWM	HIZ (m/sn)	SB (adet)	İVME (m/s²)
114	0,4160	5,00	0,8320
122	0,4992	6,00	0,8320
132	0,5824	7,00	0,8320
142	0,6656	8,00	0,8320
154	0,7488	9,00	0,8320
160	0,8320	10,00	0,8320
166	0,9152	11,00	0,8320
172	0,9984	12,00	0,8320

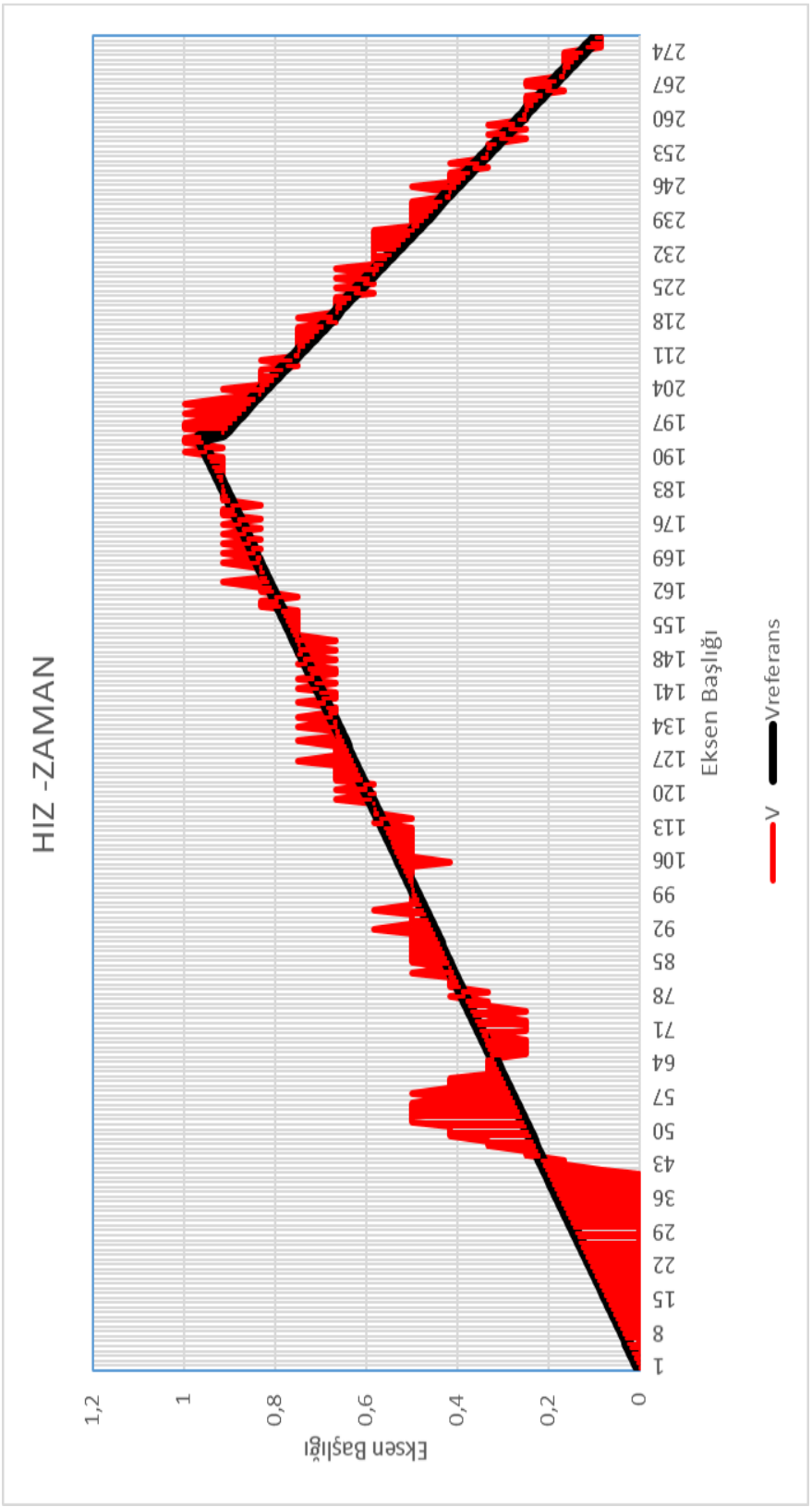
Tablo 5-1'de aracın ilk hareketi yapabilmesi için minimum 114 PWM ile çalıştırılması gerekmektedir. Bu PWM değerine karşılık gelen hız, minimum hıza karşılık gelir. Bu minimum hızın değeri de birçok ölçüm sonucunun ortalaması olan 0,416 m/s'ye karşılık gelmektedir. Aracın 14 metrelik parkurda güvenli bir şekilde yol alabilmesi için aracın hız değeri maksimum 1 m/s'ye sınırlandırılmıştır. Aracın maksimum hıza ulaştığı andaki PWM değeri yaklaşık 172'dir.

Tablo 5-1'deki hız sınırları dikkate alınarak farklı hızlanma ve yavaşlama ivmeleri PID kontrol ile pistte birçok kez denenmiştir. Tablo 5-2 araçtan alınan örnek bir sonuç tablosu görülmektedir. Burada N=0'dan başlayıp birer ardışık olarak artmaktadır ve tablonun tamamı verilemeyeceği için her 10 işlem adımındaki değerler tabloda verilmiştir. Araç kontrol uygulaması üzerinde hızlanma ivmesi, aa=0,05 m/s², yavaşlama ivmesi ise, ad=0,1 m/s² girilip aşağıdaki tablo 5.2 Sonuç Tablosu elde edilmiştir.

Tablo 5.2: Örnek Sonuç Tablosu

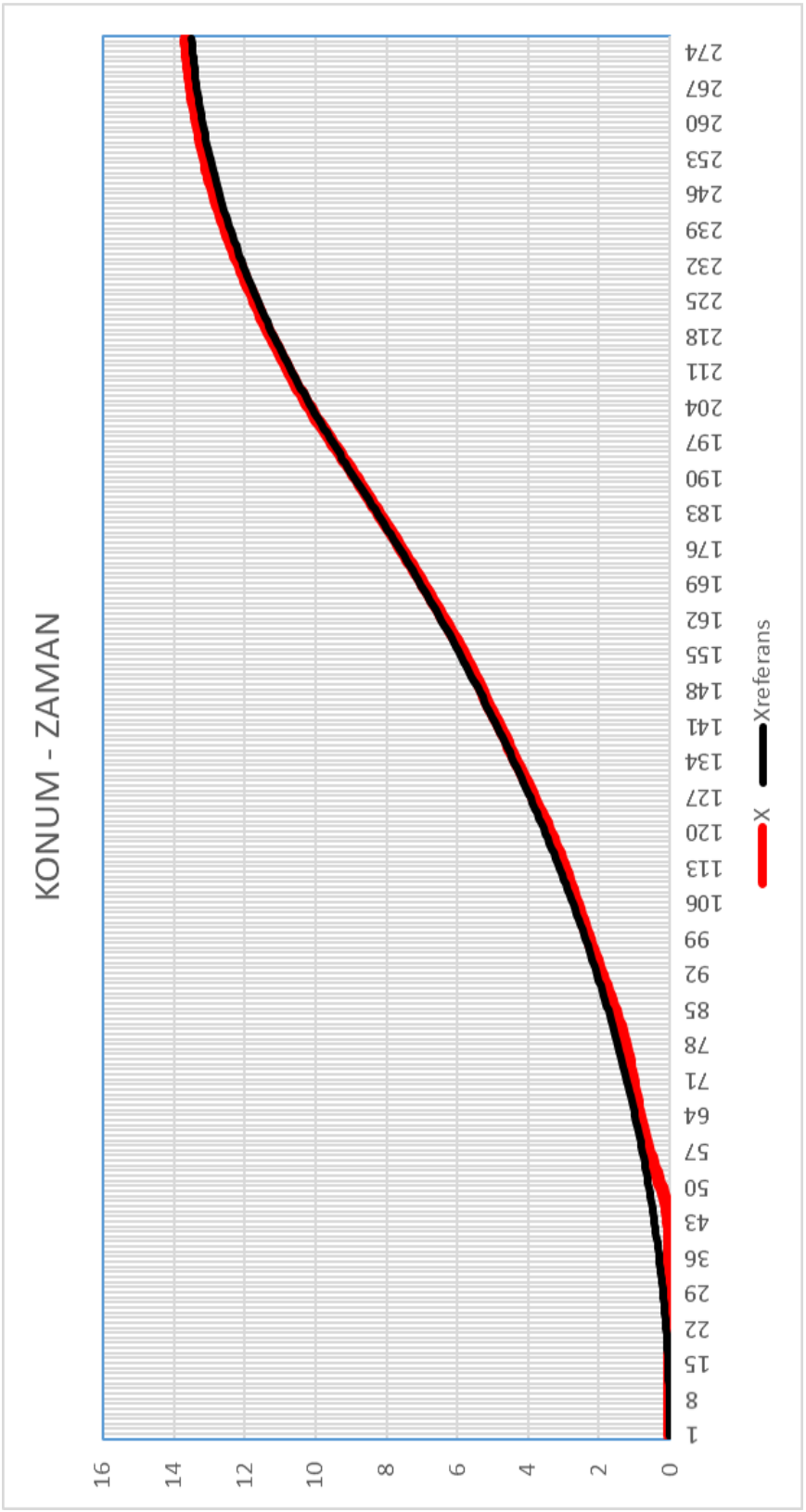
N	Zaman (sn)	Xaraç (m)	Xreferans (m)	Varaç (m/s)	Vreferans (m/s)	Sb	PID-PWM
10	1	0	0,023	0	0,05	0	7,1063
20	2	0	0,095	0	0,1	0	31,5875
30	3	0	0,218	0	0,15	0	75,9438
40	4	0	0,39	0	0,2	0	142,675
50	5	0,283	0,612	0,416	0,25	5	145,1239
60	6	0,749	0,885	0,416	0,3	5	100,0524
70	7	1,04	1,207	0,2496	0,35	3	117,7081
80	8	1,373	1,58	0,416	0,4	5	140,9789
90	9	1,855	2,002	0,4992	0,45	6	133,9477
100	10	2,371	2,475	0,4992	0,5	6	128,9891
110	11	2,862	2,997	0,4992	0,55	6	146,3483
120	12	3,436	3,57	0,6656	0,6	8	155,5167
130	13	4,11	4,192	0,7488	0,65	9	148,5395
140	14	4,801	4,865	0,6656	0,7	8	149,4883
150	15	5,508	5,587	0,7488	0,75	9	160,4879
160	16	6,265	6,36	0,7488	0,8	9	172,9054
170	17	7,122	7,182	0,832	0,85	10	169,9843
180	18	8,004	8,055	0,9152	0,9	11	173,5215
190	19	8,927	8,977	0,9984	0,95	12	179,638
200	20	9,892	9,844	0,9984	0,8552	12	154,6421
210	21	10,724	10,654	0,7488	0,7552	9	148,4966
220	22	11,448	11,365	0,6656	0,6552	8	143,6441
230	23	12,072	11,975	0,5824	0,5552	7	137,7592
240	24	12,621	12,485	0,4992	0,4552	6	122,4745
250	25	13,054	12,895	0,416	0,3552	5	110,1062
260	26	13,354	13,205	0,2496	0,2552	3	106,9415
270	27	13,57	13,416	0,1664	0,1552	2	98,4582

Tablo 5.2'nin tüm işlem adımlarını içeren değerler aşağıdaki Şekil 5.1'de Hız-Zaman grafiği ve Şekil 5.2'de Konum-Zaman grafiğinde verilmiştir.



Şekil 5.1: Sabit İvmeli Hareketin Hız-Zaman Grafiği

Şekil 5.1’deki grafikte siyah renk ile belirtilen eğri aracın formül hesaplaması ile çıkabileceği maksimum hızı ve minimum hızı göstermektedir. Kırmızı renk ile belirtilen eğri ise aracın pistte ulaşabildiği maksimum hızı ve minimum hızı göstermektedir. Şekil 5.1’deki grafikte aracın pistteki hız sonuçları ile formül ile hesaplanan hız sonucuyla benzerlik gösterdiği görülmektedir.



Şekil 5.2: Sabit İvmeli Hareketin Konum-Zaman Grafiği

Şekil 5.2’deki grafikte siyah renk ile belirtilen eğri aracın formül hesaplaması ile aldığı yolu göstermektedir. Kırmızı renk ile belirtilen eğri ise aracın pistte aldığı yolu göstermektedir. Şekil 5.2’deki grafikte aracın pistteki aldığı yol sonuçları ile formül ile hesaplanan aldığı yol sonucunun benzerlik gösterdiği görülmektedir.

6. SONUÇ ve ÖNERİLER

Tezin oluşturulan bu sistem içerisindeki hata oranı bir araç boyunu geçmemiştir. Oluşan hatayı azaltmak adına kullanılan Discovery kartı yerine yüksek kapasiteli ve daha yüksek çalışma frekanslarına sahip mikrodenetleyici seçilebilir. Aynı zamanda hız ölçümünde hassas sensörler kullanılıp hata oranı minimize edilebilir.

İleride bu çalışmanın geliştirilebilmesi için araç sayısının artırılması planlanmakta ve bu araçların birbirlerine olan emniyet mesafesi için emniyet amaçlı mesafe sensörleri ilave edilmesi düşünülmektedir. Ayrıca pil bataryasının bitme durumu göz önünde bulundurularak tek şerit yola ve araca uygun seyahat ederken elektrik enerjisini zemine yerleştirilen mekanizmadan alması ve mekanizmanın arıza yapması durumunda bataryadan elektrik enerjisini alması planlanmaktadır.

7. KAYNAKLAR

Baldacci R., Christofides N., Mingozzi A., "An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts". Mathematical Programming, 115, 2, 351-385 (2008).

Başkaya Z., Öztürk B. A., "Dal Kesme Yöntemi ve Bir Ekmek Fabrikasında Oluşturulan Araç Rotalama Problemine Uygulanması". Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 24, 1, 101-114 (2005).

Bayzan Ş., "Mobil Sistemlerde Gerçek Zamanlı Trafik Bilgisi Kullanarak Alternatif Araç Güzergahlarının Belirlenmesi Ve Uygulaması", Kocaeli Üniversitesi, (2013)

Caixia L., Anavatti S. G., Ray T., "Adaptive Route Guidance System with Realtime Traffic Information, Intelligent Transportation Systems (ITSC)", 2012 15th International IEEE Conference on, Anchorage, USA, (2012)

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

Carlino D., Depinet M., Khandelwal P., Stone P., "Approximately Orchestrated Routing and Transportation Analyzer:Large-scale Traffic Simulation for Autonomous Vehicles", 15th IEEE Intelligent Transportation Systems Conference (ITSC 2012),Anchorage, Alaska, USA,(2012)

Chong L., Huapu L., "Study of Traffic Information Analysis and DecisionSupport System Based on Grid Computing, International Journal of InformationTechnology", 12(6), 69-77, (2006)

Chuang T. N., Kung J. Y., "A New Algorithm for the Discrete Fuzzy Shortest Path Problem in a Network", Applied Mathematics and Computation, 174(1), 660-668, (2006)

Du Y., Gao X., "The new Navigation System for Automatic Guided Vehicle", (2008)

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

Fukasawa R., Longo H., Lysgaard J., Aragão M. P., Reis M., Uchoa E., Werneck R. F., "Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem". Mathematical Programming, 106, 3, 491-511, (2006).

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 İntelligent Traffic Management Systems", Transportation Research Part C, 10(5-6), 473-506, (2002)

Jakimavičius M., Mačerinskienė A., "A GIS-Based Modelling Of Vehicles Rational Routes", Journal of Civil Engineering and Management, 12(4), 303-309, (2006)

Khanmohammadi M., Mirnia K., "Multi AGV Hybrid Path Planning using Fuzzy Inference Systems", (2010)

Keshavarz E., Khorram E., "A Fuzzy Shortest Path with the Highest Reliability",Journal of Computational and Applied Mathematics, 230(1), 204-212, (2009)

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

Lin C, Chou S.Y., Hsu H.Y., "Developing adaptive driving route guidance systems based on fuzzy neural network", IEEE International Conference on Systems, Man and Cybernetics (SMC 2009),San Antonio, USA, (2009).

Taha M., Ibrahim S., Ragavan S. V., Ponnambalam S. G., "Way Point Based Deliberative Path Planner for Navigation", IEEE/ASME International Conference on Advanced Intelligent Mechatronics Suntec Convention and Exhibition Center Singapore, (2009)

Mukherjee S., "Dijkstra's Algorithm for Solving the Shortest Path Problem on Networks Under Intuitionistic Fuzzy Environment", Journal of Mathematical Modelling and Algorithms, 11(4), 345-359, (2012)

Oliveira E. ve Duarte N., "A Multi-Agent System for Simulation of Traffic Control", Proceedings of the ESM 2005 (European Simulation and Modelling), Sayfalar: 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,Sayfalar: 176 – 183, (2001)

Wahle J., Annen O., Schuster C., Neubert L., Schreckenberg M., “A Dynamic Route Guidance System Based On Real Traffic Data”, Europen Journal of Operation Research, 131(2), 302-308, (2001)

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

WEB_1, http://www.tekyolbilim.com

WEB_2, http://www.autonomouscars.com

WEB_3, https://maker.robotistan.com/stm32f407-discovery-baslangic-rehberi/

WEB_4,https://www.robocombo.com/NRF24L01-24GHz-RF-Alici-Verici-Modul-Harici-Antenli,PR-1059.html

WEB_5, https://www.st.com/content/st_com/en.html

WEB_6, https://www.direnc.net/

EKLER

8. EKLER

Ek-1

```
import time
from PyQt5.QtWidgets import QMainWindow, QMessageBox, QApplication, QDialog
from PyQt5.QtGui import QIntValidator, QRegExpValidator, QDoubleValidator,
QTextCursor
from _form_python import Ui_MainWindow
from PyQt5.QtCore import QTime, QRegExp, Qt
from serialThreadQtClass import serialThreadClass


class HomeWindow(QMainWindow, QDialog):
    def __init__(self):
        try:
            super(HomeWindow, self).__init__()
            self.runProg = True
            self.ui = Ui_MainWindow()
            self.ui.setupUi(self)
            #self.setWindowFlag(Qt.FramelessWindowHint)
            self.setAttribute(Qt.WA_MacAlwaysShowToolWindow)
            self.buttonSetVisible(False)
            self.mySerial = serialThreadClass()
            self.mySerial.mesaj.connect(self.serialReader)
            self.ui.comboBox_com.addItems(self.mySerial.serial_ports())
            self.validator()
        except Exception as err:
            print(f"HomeWindow: {err}")

    def __del__(self):
        if self.mySerial.seriport is not None or
self.mySerial.seriport.isOpen():
            self.mySerial.close()


    def baglan(self):
        try:
            port = self.ui.comboBox_com.currentText()
            baudRate = int(self.ui.comboBox_baud.currentText())
            if port != "":
                if self.mySerial.seriport is None or not
self.mySerial.seriport.isOpen():
                    try:
                        self.mySerial.open(port=port, baudrate=int(baudRate))
                        self.ui.textBrowser.append(f"{port} bağlandı.")
                        self.buttonSetVisible(True)
                        self.mySerial.start()
                        self.ui.pushButton_baglan.setStyleSheet("background-
color: rgb(205, 0, 0);")
                        self.ui.pushButton_baglan.setText("Kapat")
                    except Exception as e:
                        print(f"baglanma hatası: {e}")
                        self.ui.textBrowser.append(f"baglanma hatası: {e}")
                        self.ui.pushButton_baglan.setStyleSheet("background-
color: rgb(115, 210, 22);")
                else:
```

```
                    if self.mySerial.seriport.isOpen():
                        self.mySerial.close()
                        self.ui.textBrowser.append(f"{port} bağlantı
kapatıldı")
                        self.ui.pushButton_baglan.setText("Bağlan")
                        self.buttonSetVisible(False)
                        self.ui.pushButton_baglan.setStyleSheet("background-
color: rgb(115, 210, 22);")
                    if self.mySerial.seriport.isOpen():
                        self.ui.pushButton_baglan.setStyleSheet("background-
color: rgb(205, 0, 0);")
                        self.ui.pushButton_baglan.setText("Kapat")
            else:
                self.ui.textBrowser.append(f"Port Seçiniz.")
        except Exception as err:
            print(f"hata_baglan: {err}")

    def serialReader(self, line):
        try:
            if line != "":
                if line == "seriPortClose":
                    self.portTara()
                    self.ui.pushButton_baglan.setText("Bağlan")
                    self.buttonSetVisible(False)
                    self.ui.pushButton_baglan.setStyleSheet("background-color:
rgb(115, 210, 22);")

                self.available = True
                if line.endswith('-_pidVeriAl\r\n'):
                    self.pid = line.split('-')
                    self.ui.lineEdit_kp.setText(self.pid[1])
                    self.ui.lineEdit_ki.setText(self.pid[2])
                    self.ui.lineEdit_kd.setText(self.pid[3])
                elif line.endswith('-_paramVeriAl\r\n'):
                    self.param = line.split('-')
                    self.ui.lineEdit_pwm.setText(self.param[1])
                    self.ui.lineEdit_aa.setText(self.param[2])
                    self.ui.lineEdit_ad.setText(self.param[3])
                    self.ui.lineEdit_to.setText(self.param[4])
                    self.ui.lineEdit_k.setText(self.param[5])
                if line.endswith('sendMessagge_successful\r\n'):
                    self.ui.textBrowser.insertHtml(f'<font size=2
color="green"><strong>MASTER: Komutu gönderdi.</strong></font>')
                elif line.endswith('sendMessagge_unsuccessful\r\n'):
                    self.ui.textBrowser.insertHtml(f'<font size=2
color="red"><strong>MASTER:Komutu gönderemedi.</strong></font> ')
                elif line.endswith('_S_successful\r\n'):
                    self.ui.textBrowser.insertHtml(f'<br/> <font size=2
color="lime"><strong>SLAVE: Yanıt alındı.</strong></font><hr> </hr>')
                else:
                    line = line[:-2]
                    line = line.replace('.', ',')

                    self.ui.textBrowser.setTextColor(Qt.black)
                    self.ui.textBrowser.setFontPointSize(10)
                    self.ui.textBrowser.append(line)
                self.ui.textBrowser.moveCursor(QTextCursor.End)
        except Exception as err:
            print(f"hata_serialReader: {err}")

    def sendPCCommand(self, metin):
        self.ui.textBrowser.append(f'<hr></hr><font size=2
color="blue"><strong>PC: {metin}</strong></font> ')
```

```
    def veriAl(self):
        try:
            self.available = False
            i = 0
            self.ser.write(b'_pidVeriAl\r\n')
            self.sendPCCommand("PID Veri al komutu gönderildi.")
            QApplication.processEvents()
            while not self.available:
                i += 1
                if i == 10:
                    break
                time.sleep(0.1)
            self.available = False
            i = 0
            self.ser.write(b'_paramVeriAl\r\n')
            self.sendPCCommand("PARAMETRE al komutu gönderildi.")
            QApplication.processEvents()
            while not self.available:
                i += 1
                if i == 10:
                    break
                time.sleep(0.1)
        except Exception as err:
            print(f"hata_veriAl: {err}")

    def portTara(self):
        try:
            self.ui.comboBox_com.clear()
            serialPortList = self.mySerial.serial_ports()
            self.ui.comboBox_com.addItems(serialPortList)
        except Exception as err:
            print(f"hata_portTara {err}")

    def parametreAl(self):
        try:
            self.mySerial.sendSerial('_paramVeriAl\r\n')
            self.sendPCCommand("PARAMETRE al komutu gönderildi.")
            QApplication.processEvents()
        except Exception as err:
            print(f"hata_parametreAl {err}")

    def pidVeriAl(self):
        try:
            self.mySerial.sendSerial('_pidVeriAl\r\n')
            self.sendPCCommand("PID Veri al komutu gönderildi.")
            QApplication.processEvents()
        except Exception as err:
            print(f"hata_parametreAl {err}")

    def pid_ayarla(self):
        try:
            self.pid_kp = float(self.ui.lineEdit_kp.text())
            self.pid_ki = float(self.ui.lineEdit_ki.text())
            self.pid_kd = float(self.ui.lineEdit_kd.text())

self.ui.textBrowser.append(f"kp:{self.pid_kp}\tki:{self.pid_ki}\tkd:{self.pid_k
d} ")
            pid = f"{self.pid_kp}-{self.pid_ki}-{self.pid_kd}-_pd\r\n"
            self.mySerial.sendSerial(pid)
            time.sleep(0.5)
            self.pidVeriAl()
        except Exception as e:
            print(f"hata_pid_ayarla: {e}")
            self.ui.textBrowser.append(f"pid ayar hatası: {e}")
```

```
    def parametre_ayarla(self):
        try:
            self.parametre_pwm = int(self.ui.lineEdit_pwm.text())
            self.parametre_aa = float(self.ui.lineEdit_aa.text())
            self.parametre_ad = float(self.ui.lineEdit_ad.text())
            self.parametre_to = float(self.ui.lineEdit_to.text())
            self.parametre_k = float(self.ui.lineEdit_k.text())
            self.ui.textBrowser.append(

f"pwm:{self.parametre_pwm}\taa:{self.parametre_aa}\tad:{self.parametre_ad}\t"
                f"to:{self.parametre_to}\tk:{self.parametre_k} ")

            parametre = f"{self.parametre_pwm}-{self.parametre_aa}-
{self.parametre_ad}" \
                        f"-{self.parametre_to}-{self.parametre_k}-_prm\r\n"
            self.mySerial.sendSerial(parametre)
            time.sleep(0.5)
            self.parametreAl()
        except Exception as e:
            print(f"hata_parametre_ayarla: {e}")
            self.ui.textBrowser.append(f"parametre_ayarla hatası: {e}")

    def ivmeliHizlan(self):
        try:
            self.mySerial.sendSerial('_ivmeliHizlan\r\n')
            self.sendPCCommand("Ivmeli hızlan komutu gönderildi.")
        except Exception as err:
            print(f"_ivmeliHizlan {err}")

    def ivmeliYavasla(self):
        try:
            self.mySerial.sendSerial('_ivmeliYavasla\r\n')
            self.sendPCCommand("Ivmeli yavaşla komutu gönderildi.")
        except Exception as err:
            print(f"_ivmeliYavasla {err}")

    def dur(self):
        try:
            self.mySerial.sendSerial('_dur\r\n')
            self.sendPCCommand("Dur komutu gönderildi.")
        except Exception as err:
            print(f"hata_dur {err}")

    def bilgi_iste(self):
        try:
            self.mySerial.sendSerial('_bilgi\r\n')
            self.sendPCCommand("Bilgi iste komutu gönderildi.")
        except Exception as err:
            print(f"hata_bilgi_iste {err}")

    def temizle(self):
        try:
            self.ui.textBrowser.clear()
        except Exception as err:
            print(f"hata_temizle {err}")

    def buttonSetVisible(self, enabled):
        try:
            self.ui.pushButton_pidVerisiAl.setEnabled(enabled)
            self.ui.pushButton_pidVerisiAyarla.setEnabled(enabled)
            self.ui.pushButton_parametreVerisiAl.setEnabled(enabled)
            self.ui.pushButton_parametreVerisiAyarla.setEnabled(enabled)
            self.ui.pushButton_ivmeliHizlan.setEnabled(enabled)
            self.ui.pushButton_ivmeliYavasla.setEnabled(enabled)
            self.ui.pushButton_dur.setEnabled(enabled)
```

```
            self.ui.pushButton_bilgi.setEnabled(enabled)
            self.ui.comboBox_com.setEnabled(not enabled)
            self.ui.comboBox_baud.setEnabled(not enabled)
            self.ui.pushButton_portlar.setEnabled(not enabled)
        except Exception as err:
            print(f"hata_buttonSetVisible {err}")

    def validator(self):
        try:
            self.ui.lineEdit_kp.setValidator(QDoubleValidator(0.00000, 1, 5,
self))
            self.ui.lineEdit_kd.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
            self.ui.lineEdit_ki.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
            self.ui.lineEdit_pwm.setValidator(QIntValidator(0, 999, self))
            self.ui.lineEdit_aa.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
            self.ui.lineEdit_ad.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
            self.ui.lineEdit_to.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
            self.ui.lineEdit_k.setValidator(QDoubleValidator(0.00000, 100, 5,
self))
        except Exception as err:
            print(f"hata_validator {err}")
```

Ek-2

```
#include "MY_NRF24.h"
#include "PID.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

/* NRF ile ilgili Degiskenler Baslangic */

uint64_t address = 0x11223344AA;
char myTxData[32];
char myRxData[32];

/* NRF ile ilgili Degiskenler Bitis */

/* Motor ile ilgili Degiskenler Baslangic */

volatile int motorPwm = 0;
PIDController motorPid;

/* Motor ile ilgili Degiskenler Bitis */

/* Servo ile ilgili Degiskenler Baslangic */

volatile double servoCenterPwm = 73;

/* Servo ile ilgili Degiskenler Bitis */

/* Sensor ile ilgili Degiskenler Baslangic */

volatile uint32_t sbSayisiOnceki = 0, sbSayisiSimdiki = 0;

/* Sensor ile ilgili Degiskenler Bitis */

/* Hesaplama ile ilgili Degiskenler Baslangic */

volatile float  aa = 0.2f, ad = 0.2f;
volatile uint32_t       N = 0; //Her 100ms bir artacak olan degisken
volatile float  to = 0.1f, yol[1], hiz[1], k = 0.00832,
                        V[2048], X[2048], a[2048], Vexcel[2048], Xexcel[2048],
        Sb[2048], sbToplam[2048],PID[2048];

/* Hesaplama ile ilgili Degiskenler Bitis */


/* NRF CEVAP GONDERME Baslangic*/
void sendResponse(const void *string) {

        NRF24_stopListening();
        NRF24_openWritingPipe(address);
        while(!NRF24_write(string, 32)){
                NRF24_resetStatus();
                NRF24_flush_tx();
                HAL_GPIO_TogglePin(Led1_GPIO_Port, Led1_Pin);
                HAL_Delay(750);
        }
        for (int x = 0; x < sizeof(myRxData); x++)
                myRxData[x] = (int) NULL;
        NRF24_openReadingPipe(1, address);
        NRF24_startListening();

}
/* NRF CEVAP GONDERME Bitisi */

/* Zamanlayici Fonksiyonlari Baslangic*/

void timerStart() {
        __HAL_TIM_SetCounter(&htim6,0);
        while(HAL_TIM_Base_Start_IT(&htim6) != HAL_OK);
}
```

```
void timerStop() {
	HAL_TIM_Base_Stop_IT(&htim6);
}
/* Zamanlayici Fonksiyonlari Bitis*/

/* Sensor Fonksiyonlari Baslangic*/
void stepSensorStart() {
	__HAL_TIM_SetCounter(&htim1,0);
	HAL_TIM_Base_Start(&htim1);
}

void stepSensorStop() {
	HAL_TIM_Base_Stop(&htim1);
}
/* Sensor Fonksiyonlari Bitis*/

/* SERVO MOTOR Fonksiyonlari Baslangic*/
void servoPwmUpdate(volatile double pwm) {
	__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, pwm);
}

void servoPwmStart() {
	HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
}

void servoPwmStop() {
	HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_3);
}
/* SERVO MOTOR Fonksiyonlari Bitis*/

/* DC  MOTOR Kontrol Fonksiyonlari Baslangic*/
void motorPwmUpdate(uint16_t pwm) {
	__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_3, pwm);
}

void motorPwmStart() {
	HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_3);
}

void motorPwmStop() {
	__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_3, 0);
	HAL_TIM_PWM_Stop_IT(&htim2, TIM_CHANNEL_3);
}

void motorForward() {
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INB_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(Motor_EN_GPIO_Port, Motor_EN_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INA_Pin, GPIO_PIN_SET);
}
void motorBack() {
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INA_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(Motor_EN_GPIO_Port, Motor_EN_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INB_Pin, GPIO_PIN_SET);
}
void motorStop() {
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INA_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(Motor_EN_GPIO_Port, Motor_EN_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INB_Pin, GPIO_PIN_RESET);
}

void motorBrake(int frenSure) {
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INA_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INB_Pin, GPIO_PIN_SET);
	delay_us(frenSure);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INA_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(Motor_INA_GPIO_Port, Motor_INB_Pin, GPIO_PIN_RESET);
}
/* DC MOTOR Kontrol Fonksiyonlari Bitis*/

/* Ivmeli Hareket Hesaplama Baslangic*/

void excelHizKonumHesapla(float aaIvme, float adIvme) {

	if (aaIvme != 0 ){
```

```
                Vexcel[N] = V[N];
                Xexcel[N] = X[N];
                for (int _N = N; _N < N + 1024; ++_N) {
                        if(1 <= Vexcel[_N ])
                                Vexcel[_N + 1] = 1;
                        Else
                                Vexcel[_N + 1] = aaIvme * to + Vexcel[_N];
                        Xexcel[_N + 1] = Vexcel[_N] * to + Xexcel[_N];
                }
        }

        if (adIvme != 0 ){
                Vexcel[N] = V[N];
                Xexcel[N] = X[N];
                for (int _N = N; _N < N + 1024; ++_N) {
                        Vexcel[_N + 1] = -adIvme * to + Vexcel[_N];
                        Xexcel[_N + 1] = Vexcel[_N] * to + Xexcel[_N];
                        if(Vexcel[_N + 1] <= 0)
                                Break;
                }}

}
/* Ivmelı Hareket Hesaplama Bitis*/

/* 100ms bir oluşan kesme fonksiyonu Baslangıc*/

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {

        if (htim->Instance == TIM6) {
                volatile float Dx = 0, motorPidValue=0;
                HAL_GPIO_TogglePin(Led2_GPIO_Port, Led2_Pin);
                ++N;
                sbSayisiOnceki = sbSayisiSimdiki;
                sbSayisiSimdiki = __HAL_TIM_GetCounter(&htim1); //Timr1 Her
        siyah beyaz okumasında 1 artırır.
                sbToplam[N] = (float)(sbSayisiSimdiki);
                Sb[N] = (float)(sbSayisiSimdiki - sbSayisiOnceki);
                Dx = Sb[N] * k; //100ms'de alınan yol

                X[N] = X[N - 1] + Dx;
                V[N] = (X[N] - X[N - 1]) / to;
                a[N] = (V[N] - V[N - 1]) / to;
                if((N > 60 && V[N] <= 0) || Vexcel[N] <= 0){
                        HAL_GPIO_WritePin(Led1_GPIO_Port, Led2_Pin, GPIO_PIN_RESET);
                        motorPwmStop();
                        motorStop();
                        timerStop();
                        servoPwmStop();
                        stepSensorStop();
                }

                motorPidValue = PIDController_Update(&motorPid, Xexcel[N], X[N]);
                if (motorPidValue > 1000)
                        motorPwmUpdate(1000);
                Else if (motorPidValue < 0)
                        motorPwmUpdate(0);
                Else
                        motorPwmUpdate(motorPidValue);
                PID[N] = motorPidValue;

                }

}

/* 100ms bir oluşan kesme fonksiyonu Bitis*/

/* Mikrosaniye cinsinden gecikme fonksiyonu Baslangıc*/

void delay_us(uint16_t us)
{
        __HAL_TIM_SET_COUNTER ( &htim7, 0 );   // sayaç değerini 0 olarak ayarlandı
        while  (__HAL_TIM_GET_COUNTER ( &htim7)  <  us);   // sayacın parametredeki us
girişine ulaşmasını bekleniyor
}
```

```
/* Mikrosaniye cinsinden gecikme fonksiyonu Bitis */

/* Cizgi takip sensoru fonksiyonu Baslangıc */
void QTR8_RC_READ() {

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  GPIO_InitStruct.Pin = S1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(S1_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(S2_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(S3_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S4_Pin|S5_Pin|S6_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S7_Pin|S8_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, S4_Pin|S5_Pin|S6_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, S7_Pin|S8_Pin, GPIO_PIN_SET);
  delay_us(500);

  GPIO_InitStruct.Pin = S1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(S1_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(S2_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(S3_GPIO_Port, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S4_Pin|S5_Pin|S6_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = S7_Pin|S8_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  delay_us(750);

        uint8_t sensor_durum =0;

        sensor_durum |= HAL_GPIO_ReadPin(S1_GPIO_Port, S1_Pin);
```

```
        sensor_durum |= HAL_GPIO_ReadPin(S2_GPIO_Port, S2_Pin) << 1;
        sensor_durum |= HAL_GPIO_ReadPin(S3_GPIO_Port, S3_Pin) << 2;
        sensor_durum |= HAL_GPIO_ReadPin(S4_GPIO_Port, S4_Pin) << 3;
        sensor_durum |= HAL_GPIO_ReadPin(S5_GPIO_Port, S5_Pin) << 4;
        sensor_durum |= HAL_GPIO_ReadPin(S6_GPIO_Port, S6_Pin) << 5;
        sensor_durum |= HAL_GPIO_ReadPin(S7_GPIO_Port, S7_Pin) << 6;
        sensor_durum |= HAL_GPIO_ReadPin(S8_GPIO_Port, S8_Pin) << 7;

        switch (sensor_durum) {
                case 1: // en sağ
                        break;
                        servoPwmUpdate(servoCenterPwm + 7);
                case 3:
                        servoPwmUpdate(servoCenterPwm + 6);
                        break;
                case 2:
                        servoPwmUpdate(servoCenterPwm + 5);
                        break;
                case 6:
                        servoPwmUpdate(servoCenterPwm + 4);
                        break;
                case 4:
                        servoPwmUpdate(servoCenterPwm + 3);
                        break;
                case 12:
                        servoPwmUpdate(servoCenterPwm + 2);
                        break;
                case 8:
                        servoPwmUpdate(servoCenterPwm + 1);
                        break;
                case 24: // orta
                        servoPwmUpdate(servoCenterPwm);
                        break;
                case 16:
                        servoPwmUpdate(servoCenterPwm - 1);
                        break;
                case 48:
                        servoPwmUpdate(servoCenterPwm - 2);
                        break;
                case 32:
                        servoPwmUpdate(servoCenterPwm - 3);
                        break;
                case 96:
                        servoPwmUpdate(servoCenterPwm - 4);
                        break;
                case 64:
                        servoPwmUpdate(servoCenterPwm - 5);
                        break;
                case 192:
                        servoPwmUpdate(servoCenterPwm - 6);
                        break;
                case 128:       //en sol
                        servoPwmUpdate(servoCenterPwm - 7);
                        break;
                default:
                        break;
        }


}
/* Cizgi takip sensoru fonksiyonu Bitis */


int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
```

```
  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_UART7_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_TIM6_Init();
  MX_TIM7_Init();
  /* USER CODE BEGIN 2 */

          HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
          HAL_NVIC_SetPriority(RCC_IRQn, 0, 0);

          HAL_NVIC_SetPriority(TIM1_TRG_COM_TIM11_IRQn, 2, 0);
          HAL_NVIC_SetPriority(TIM1_BRK_TIM9_IRQn, 2, 0);
          HAL_NVIC_SetPriority(TIM1_CC_IRQn, 2, 0);
          HAL_NVIC_SetPriority(TIM1_UP_TIM10_IRQn, 2, 0);

          HAL_NVIC_SetPriority(TIM6_DAC_IRQn, 3, 0);
          HAL_NVIC_SetPriority(TIM7_IRQn, 4, 0);

          HAL_NVIC_SetPriority(USART2_IRQn, 2, 0);
          HAL_NVIC_SetPriority(USART3_IRQn, 2, 0);

          HAL_TIM_Base_Start(&htim7);


        NRF24_begin(SPI1_CE_GPIO_Port, SPI1_CS_Pin, SPI1_CE_Pin, hspi1); // NRF MODÜLÜNÜN CE
CSN PİNLERİNİ TANIMLADIK ve NRF YE HANGİ SPI ile HABERLEŞECEĞİ BİLDİRİLDİ.
        nrf24_DebugUART_Init(huart7); // NRF den GELEN VERİLERİ PC ye GÖNDERMEK İÇİN UART2
HABERLEŞME MODÜLÜ KULLANILDI.
        //printRadioSettings(); // PC ye NRF BİLGİLERİNİ GÖNDERME , GÖSTERME
        NRF24_setAutoAck(true);
        NRF24_setChannel(52);
        NRF24_setPayloadSize(32);
        NRF24_openReadingPipe(1, address);
        NRF24_enableDynamicPayloads();
//      NRF24_enableAckPayload();
        NRF24_startListening();

        motorPid.Kp = 300.0f;
        motorPid.Ki = 50.0f;
        motorPid.Kd = 65.0f;
        motorPid.T = 0.1f;
        motorPid.tau = 15.4f;
        motorPid.limMin = 0;
        motorPid.limMax = 1000;
        PIDController_Init(&motorPid);

        servoPwmStart();
        servoPwmUpdate(servoCenterPwm);

//*********************************************************************
        // ILK acilista verileri kullanıcı arayuzune gönder
        char cevapGonder[32];
        sprintf(cevapGonder, "-%.2f-%.2f-%.2f", motorPid.Kp, motorPid.Ki,
                        motorPid.Kd);
        sendResponse(cevapGonder);
        HAL_Delay(5);
        sendResponse("-_pidVeriAl\r\n");
        HAL_Delay(5);
        sendResponse("_S_successful\r\n");
```

```
        sprintf(cevapGonder, "-%d-%.3f-%.2f-%.2f-%.5f", motorPwm,
                        aa, ad, to, k);
        sendResponse(cevapGonder);
        HAL_Delay(5);
        sendResponse("-_paramVeriAl\r\n");
        HAL_Delay(5);
        sendResponse("_S_successful\r\n");
//*********************************************************************

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
        while (1) {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */


                QTR8_RC_READ();  //Cizgi takip fonksyonu
                /* NRF'DEN GELEN VERİLERİN DEGERLENDIRILMESI */
                if (NRF24_available()) {
                        NRF24_read(myRxData, 32);
                        if (strstr((char*) myRxData, "_ivmeliHizlan\r\n") != NULL) {

                                N = 0;
                                sbSayisiSimdiki = 0, sbSayisiOnceki = 0, yol[0] = 0.0f,
hiz[0] = 0.0f;
                                V[0] = 0, X[0] = 0;
                                PIDController_Init(&motorPid);
                                sendResponse("_S_successful\r\n");
                                excelHizKonumHesapla(aa, 0);
                                servoPwmStart();
                                stepSensorStart();
                                __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_3, 0);
                                motorPwmStart();
                                motorForward();
                                timerStart();

                        } else if (strstr((char*) myRxData, "_ivmeliYavasla\r\n") != NULL) {
                                sendResponse("_S_successful\r\n");
                                excelHizKonumHesapla(0, ad);

                        } else if (strstr((char*) myRxData, "_dur\r\n") != NULL) {
                                sendResponse("_S_successful\r\n");
                                motorPwmStop();
                                motorBrake(50000);
                                servoPwmStop();
                                stepSensorStop();
                                motorStop();
                                delay_us(65000);
                                timerStop();

                        } else if (strstr((char*) myRxData, "_bilgi\r\n") != NULL) {
                                char cevapGonder[32];
                                if(motorPwm == 0){
                                                                sprintf(cevapGonder,
"N\tX\tXexcel\tV\tVexcel\tA\tSb\t");
                        //                                      HAL_UART_Transmit(&huart7, (uint8_t*)
cevapGonder, strlen(cevapGonder), 100);
                                                                sendResponse(cevapGonder);
                                                                HAL_Delay(5);
                                                                sprintf(cevapGonder, "PID-PWM\r\n");
                                                                sendResponse(cevapGonder);

                                                                for (int n = 0; n <= N; ++n) {
                                                                        sprintf(cevapGonder,
"%d\t%.3f\t%.3f\t",n, X[n], Xexcel[n]);
                        //                                              HAL_UART_Transmit(&huart7,
(uint8_t*) cevapGonder, strlen(cevapGonder), 100);
                                                                        sendResponse(cevapGonder);
                                                                        HAL_Delay(5);
```

```
                                                        sprintf(cevapGonder,
"%.4f\t%.4f",V[n],Vexcel[n]);
                        //
                                                        HAL_UART_Transmit(&huart7,
(uint8_t*) cevapGonder, strlen(cevapGonder), 100);
                                                        sendResponse(cevapGonder);
                                                        HAL_Delay(5);
                                                        sprintf(cevapGonder,
"\t%.3f\t%.4f\t%.4f\r\n",a[n], Sb[n],PID[n]);
                        //
                                                        HAL_UART_Transmit(&huart7,
(uint8_t*) cevapGonder, strlen(cevapGonder), 100);
                                                        sendResponse(cevapGonder);
                                                        HAL_Delay(5);
                                                }
                                }else{
                                                        sprintf(cevapGonder,
"N\tX\tV\tA\tSb\t");
                        //                              HAL_UART_Transmit(&huart7, (uint8_t*)
cevapGonder, strlen(cevapGonder), 100);
                                                        sendResponse(cevapGonder);
                                                        HAL_Delay(5);
                                                        sprintf(cevapGonder, "PWM\r\n");
                                                        sendResponse(cevapGonder);

                                                        for (int n = 0; n <= N; ++n) {
                                                        sprintf(cevapGonder,
"%d\t%.3f\t%.4f",n, X[n], V[n]);
                        //                              HAL_UART_Transmit(&huart7,
(uint8_t*) cevapGonder, strlen(cevapGonder), 100);
                                                        sendResponse(cevapGonder);
                                                        HAL_Delay(5);
                                                        sprintf(cevapGonder,
"\t%.3f\t%.4f\t%.4f\r\n",a[n], Sb[n],PID[n]);
                        //                              HAL_UART_Transmit(&huart7,
(uint8_t*) cevapGonder, strlen(cevapGonder), 100);
                                                        sendResponse(cevapGonder);
                                                        HAL_Delay(5);
                                                }

                                }
                                sendResponse("\r\n_S_successful\r\n");

                        } else if (strstr((char*) myRxData, "_pidVeriAl\r\n") != NULL) {

                                char cevapGonder[32];
                                sprintf(cevapGonder, "-%.2f-%.2f-%.2f", motorPid.Kp,
motorPid.Ki,
                                                motorPid.Kd);
                                sendResponse(cevapGonder);
                                HAL_Delay(5);
                                sendResponse("-_pidVeriAl\r\n");
                                HAL_Delay(5);
                                sendResponse("_S_successful\r\n");

                        } else if (strstr((char*) myRxData, "_paramVeriAl\r\n") != NULL) {

                                char cevapGonder[32];
                                sprintf(cevapGonder, "-%d-%.3f-%.2f-%.2f-%.5f", motorPwm,
                                                aa, ad, to, k);
                                sendResponse(cevapGonder);
                                HAL_Delay(5);
                                sendResponse("-_paramVeriAl\r\n");
                                HAL_Delay(5);
                                sendResponse("_S_successful\r\n");

                        } else if (strstr((char*) myRxData, "-_prm\r\n") != NULL) {

                                char strprm[6][7];
                                char *token = strtok(myRxData, "-");
                                int i = 0;
                                while (token != NULL) {
                                        sprintf(strprm[i], "%s", token);
                                        token = strtok(NULL, "-");
                                        i++;
```

```
                                }
                                motorPwm = atoi(strprm[0]);
                                motorPwmUpdate(motorPwm);
                                aa = strtof(strprm[1], NULL);
                                ad = strtof(strprm[2], NULL);
                                to = strtof(strprm[3], NULL);
                                k = strtof(strprm[4], NULL);
                                char cevapGonder[32];
                                sprintf(cevapGonder, "%s", myRxData);
                                sendResponse(cevapGonder);
                                sendResponse("_S_successful\r\n");

                        } else if (strstr((char*) myRxData, "-_pd\r\n") != NULL) {

                                char strpid[5][6];
                                char *token = strtok(myRxData, "-");
                                int i = 0;
                                while (token != NULL) {
                                        sprintf(strpid[i], "%s", token);
                                        token = strtok(NULL, "-");
                                        i++;
                                }
                                motorPid.Kp = strtod(strpid[0], NULL);
                                motorPid.Ki = strtod(strpid[1], NULL);
                                motorPid.Kd = strtod(strpid[2], NULL);
                                char cevapGonder[32];
                                sprintf(cevapGonder, "%s", myRxData);
                                sendResponse(cevapGonder);
                                sendResponse("_S_successful\r\n");

                        }
                } //NRF24_available end}
  /* USER CODE END 3 */
}
```