FIRÇASIZ DOĞRU AKIM MOTOR SÜRÜCÜ

2023

LİSANS TEZİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

Muhammed Enes AKTAŞ

Burak İsmail IŞIK

FIRÇASIZ DOĞRU AKIM MOTOR SÜRÜCÜ

Muhammed Enes AKTAŞ
Burak İsmail IŞIK

Karabük Üniversitesi
Mühendislik Fakültesi
Elektrik-Elektronik Mühendisliği Bölümü

Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK
Ocak 2024

Muhammed Enes AKTAŞ ve Burak İsmail IŞIK tarafından hazırlanan “Fırçasız Doğru Akım Motor Sürücü” başlıklı bu tezin Lisans Tezi olarak uygun olduğunu onaylarım.

06/06/2024

Doç. Dr. Mustafa GÖKDAĞ ..……...

Tez Danışmanı,

Karabük Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü

Doç. Dr. Ozan Gülbudak ..……...

Karabük Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü

Dr. Öğr. Üyesi Ahmet Kaymaz ..……...

Karabük Üniversitesi Mekatronik Mühendisliği Bölümü

2007 KARABÜK ÜNİVERSİTESİ

ÖZET

Lisans Tezi

Fırçasız Doğru Akım Motor Sürücü

Muhammed Enes AKTAŞ-Burak İsmail IŞIK

Karabük Üniversitesi
Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği Bölümü

Tez Danışmanı
Doç. Dr. Mustafa Gökdağ
Ocak 2024, XX sayfa

Bu çalışmada, mikrodenetleyici yardımı ile 6 step metodu kullanılarak Hall etkisi (Hall Effect) sensörü ile 3 fazlı fırçasız doğru akım motorunun (BLDC) hız kontrolü sağlanmaktadır.

Çalışma kapsamında fırçasız doğru akım motorunu (BLDC) çalışma metodu ve nasıl kontrolü sağlanacağı araştırıldı. Belirlenen şartlar ve kriterlere karşılayan bir tasarım yapıldı. Yapılan tasarım ile 60V'a kadar çalışma aralığında yüksek güçlü ve yüksek verimli BLDC motor sürücü uygulaması gerçekleştirildi. Matlab Simulink kullaılara yapılan testler sonucunda Stator Back-EMF sensörlerinin 60^{o} gecikme ile adımlarını gerçekleştirdiği incelendi. 50V giriş uygulandığı durumda stator akımı ve Back-EMF değerlerinin analizi yapıldı. BLDC motor sürücü olarak tasarlanan ESC, STM32F103C8T6 ve birim çevreler ile motora gerekli olan PWM değerlerini sağlamaktadır. Şematik ve PCB tasarımları eklenerek ne malzeme kullanıldığı anlatıldı.

Sonuç olarak, bu tez çalışması ile belirlenen parametreleri gerçekleştirerek BLDC motor sürücüsü tasarımı yapılabilmektedir.

Anahtar Kelimeler **:** ESC, BLDC, 3 fazlı fırçasız motor sürücü, sensörlü BLDC.

ABSTRACT

B. Sc. Thesis

Brushless Direct Current Motor Driver

Muhammed Enes AKTAŞ-Burak İsmail IŞIK

Karabük University
Faculty of Engineering Department of Electrical-Electronics Engineering

Thesis Advisor
Assoc. Prof. Dr. Mustafa Gökdağ
January 2024, XX pages

In this study, speed control of a three-phase brushless direct current (BLDC) motor is achieved using a microcontroller and the 6-step method with the assistance of a Hall Effect sensor. Within the scope of the research, the operating method of the brushless direct current motor (BLDC) and how its control can be ensured were investigated. A design meeting the specified conditions and criteria was developed.

With the designed system, a high-power and efficient BLDC motor driver application was implemented within the operating range of up to 60V. Through tests conducted using Matlab Simulink, it was observed that the Stator Back-EMF sensors executed their steps with a 60-degree delay. An analysis of stator current and Back-EMF values was carried out when a 50V input was applied. The Electronic Speed Controller (ESC) designed as the BLDC motor driver provides the necessary PWM values to the motor using STM32F103C8T6 and peripheral circuits. Schematic and PCB designs were provided, and the materials used were explained.

In conclusion, through this thesis work, the design of a BLDC motor driver can be achieved by realizing the specified parameters.

Key Words **:** ESC, BLDC, Three Phase Brushless Motor Driver, Sensor-Equipped BLDC

TEŞEKKÜR

Lisans tezi konusu seçimi ve çalışması süresince bizimle ilgilenen ve bilgilerini paylaşan değerli hocamız Doç. Dr. Mustafa GÖKDAĞ'a en içten dileklerimiz ile teşekkür ederiz.

Kardemir yardımcı işletmeler direktörü sayın Mehmet Lütfi NALÇABASMAZ' a teşekkür ederiz.

2007 KARABÜK ÜNİVERSİTESİ

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Muhammed Enes AKTAŞ
Burak İsmail IŞIK

2007
KARABÜK
ÜNİVERSİTESİ

İÇİNDEKİLER

Sayfa

2007 KARABÜK ÜNİVERSİTESİ

TABLO LİSTESİ

Tablo **Sayfa**

2007 KARABÜK ÜNİVERSİTESİ

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

2007 KARABÜK ÜNİVERSİTESİ

BÖLÜM 1

GENEL BİLGİLER

1.1. Giriş

Günümüzde elektriksel enerjiyi mekani enerjiye dönüştüren elektrik motorlarının icadı teknolojinin geleceği için çok büyük bir önem taşımaktadır. Faraday'ın bir iletkenin içinden akım geçirerek manyetik alan oluştuğunu gözlemlemesi doğru akım makinelerinin miladıdır. Temelde bütün DC makinelerinin çalışma mantığı aynıdır. İlk üretilen motordan günümüzde üretilen motorlara kadar. [1] Tabii ki, DC motorlar, doğru akım (DC) kullanarak dönme hareketi üreten elektromekanik cihazlardır. Bu makineler temel olarak bir manyetik alan ve bu alana maruz kalan bir iletken tel ile eylemini gerçekler. 2 ana bölümden oluşur Stator ve Rotor;

- Stator: Motorun hareket etmeyen kısmını oluşturur. Manyetik alanın üretilmesinde görevlidir.
- Rotor: Manyetik alan içinde dönen kısımdır.

Yukarıda da belirtildiği gibi BLDC motorlar diğer DC motorlar ile benzer çalışma prensibine sahiptir. Elektrik akımının rotor içindeki bobinlere uygulanması bobinlerin manyetik alan üretmesini sağlar. Üretilen manyetik alan Stator ile etkileşime girer ve Rotorun dönmesine sebep olur. Endüstriyel piyasada kullanılan makinelerin büyük bir çoğunluğunda doğrudan akım (DC) motorları kullanılır. Kullanım amaçlarına göre 4 çeşit DC motor vardır;

- Fırçalı DC motor
- Fırçasız DC (BLDC) motor
- Step motor
- Servo motor

BLDC motorların endüstrideki kullanımı gün geçtikçe artıyor bunun başlıca 3 sebebi var;

- Düşük hızlarda $(\omega|r)$ yüksek Tork(τ) ürettikleri için BLDC motorlar endüstri de vinçler, koşu bantları ve konveyör sistemlerde kullanılır. BLDC motorlar tüm hız aralıklarında yüksek tork üretebilmesi,
- Daha uzun ömürlü çalışma süresi, daha az ve daha kolay bakım gereksinimi olması,
- Eski jenerasyon fırçalı motorlara göre güç tüketiminin (W en az %10 daha verimli, daha sessiz çalışıyor olması ve nominal hız değerine kısa sürede ulaşması,

tercih sebebidir. [2]

BLDC motorlar sabit mıknatıs yapısına sahip oldukları için rotorda elektriksel bir yön değiştirme gerçekleşir. Bu işlem motorun dönme yönünü kontrol edebilmek için kullanılır. Fırçalı DC motorlarından ayıran en büyük farkı komütasyonu fırça yerine sürücü yardımı ile elektronik olarak yapmasıdır. Motorların etkili bir şekilde çalışabilmesi için kontrol ve sürücü bloğuna ihtiyaç duymaktadır. Bu sistemler, motorun nominal değere istenilen sürede çıkmasını ve doğru bir şekilde hızlanıp yavaşlamasını sağlar. Bu sistemin komütasyon çalışmasını Elektronik hız kontrolü (ESC) kullanılarak gerçeklenir.

ESC, anahtarlama elemanı (MosFet) kontrol edilmesini sağlayan sürücü (MosFet Driver), istenilen komütasyon hızına ulaşabilmek için MosFet sürücüyü kontrol eden mikroişlemci(μC)ve gerekli çalışma Voltaj'ı (V) için gerilim regülatörü içeren bir kontrol sistemidir.

1.2. Çalışmanın Amacı

Bu tez çalışmasında yapılmak istenen BLDC motorların tam kontrolünü sağlayabilecek Elektronik hız kontrolcüsü (ESC) tasarlanmakdır. Bu tasarım 4 ana konuyu ele almaktadır:

1)Optimizasyon ve Verimlilik:

- Sürücü tasarımının temel unsuru, motorun istenilen şekilde çalışabilmesini sağlamaktır. Bu kapsamda, tasarım sürecinin kullanılacak kontrolcü algoritması ve parametreleri üzerinde durulacaktır.

2)Hız ve Tork Kontrolü:

- Kullanım alanı nedeniyle hız ve tork kontrolü birçok uygulama için kritiktir. Tez çalışmasının amacı sürücünün istenilen hız ve tork seviyelerine hassas bir şekilde ulaşabilmesidir.

3)Dijital Kontrol ve Mikrodenetleyici:

- Kontrolün hassas bir şekilde sağlanması için mikrodenetleyici tabanlı sürücü tasarlanmalı ve bu tasarımın performansı incelenmelidir.

4)Enerji Verimliliği ve Sürdürülebilirlik:

- Bu bağlamda tasarımın enerji tüketimi, kayıpları ve uzun vadedeki etkisi incelenmelidir.

2007 KARABÜK ÜNİVERSİTESİ

BÖLÜM 2

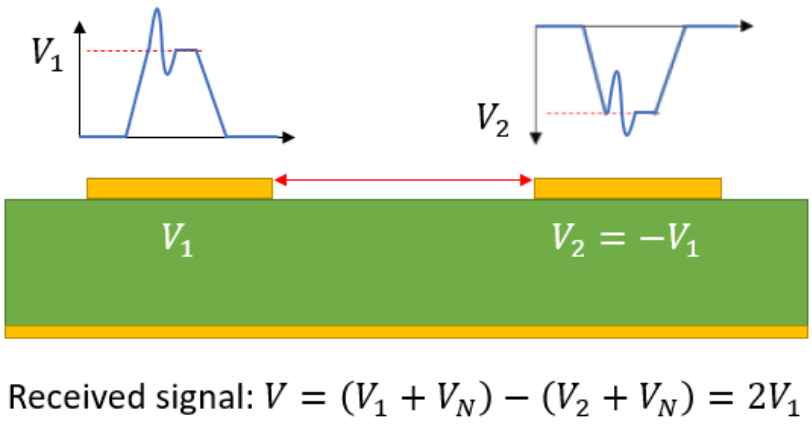
LİTERATÜR TARAMASI

- Chapman, S. J. (2005). Electric Makinelerinin Temelleri, BLDC motorların çalışma prensibi, iç yapısı ve motor performansına etkileri analiz edildi.

- Texas Instruments. (2016). "Brushless DC (BLDC) Motor Fundamentals.", [3] motor sürücünün malzemeleri ve çalışma prensibi incelendi.

- Uğur Dereli. (2020). "Fırçasız Doğru Akım Motoru (BLDC) ve Sürücüsü Tasarımı."(Yüksek Lisans Tezi), [4].

- Zachariah Peterson. (2021). “What Are Differential Pairs and Differential Signals?”. [5]



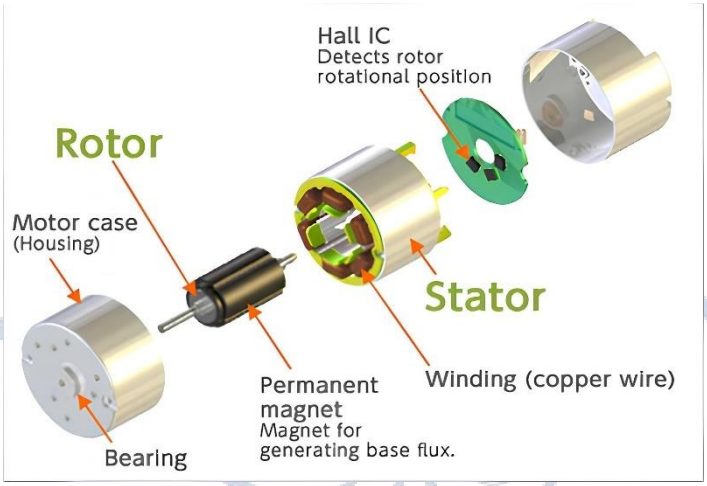
Şekil 1 Diferansiyel Çift

- Texas Instruments.(2023). “Bootstrap Circuitry Selection for Half-Bridge Configurations.” (Application Note) [6]

BÖLÜM 3

FIRÇASIZ DOĞRU AKIM MOTORLARI

3.1. Fırçasız Doğru Akım Motorların Yapısı



Şekil 2 BLDC Motor Yapısı

Fırçasız Doğru Akım Motorlarının ön planda olmasının en büyük sebebi hem azalan maliyetleri hem de kullanım işlevselliği bakımından avantajlı olmasıdır. Şekil 3.1'de görülen yapılar sırası ile motor şasesi, rotor, stator, hall sensörü şeklindedir. [4]

Motorlardaki komütasyon, motorun rotorunun dönme hareketinin manyetik alandaki değişikliklere bağlı olarak akımın doğru bir şekilde yönlendirilmesini ifade eder. [7]

Geleneksel DC motorlarda, motorun komütatör adı verilen parçası rotorun poziyonunu algılayarak komütasyon işlemini gerçekleştirir. Ancak BLDC motorlarda fırça bulunmadığı için komütatör de bulunmaz bu sebeple sensörlü veya sensörsüz komütasyon gerçekleştirilir.

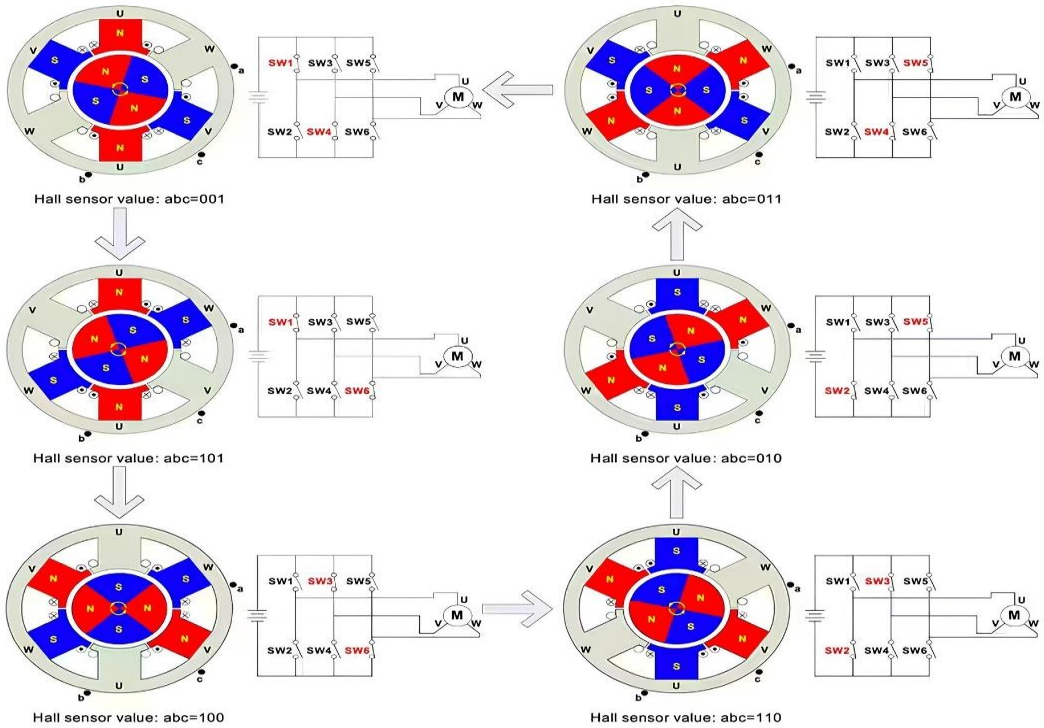
- Sensörlü Komütasyon: Sensörlü Fırçasız Doğru Akım Motorları, rotorun pozisyonunu anlayabilmek için sensörler kullanır. Bu sensörler genellikle Hall Effect (Hall Etkisi) sensörleri olur.

BÖLÜM 4

BLDC KONTROLÜ VE SÜRÜCÜ DEVRE TASARIMI

4.1.Sensörlü Kontrol

Sensörlü BLDC motorlarda hem rotorun pozisyonunu bulmak hem de rotorun çembersel hareketini sağlamak amacıyla Hall Effect sensörleri kullanılır. Statorun etrafına 60^o açı aralığı ile yerleştirilen 3 adet hall efect sensör ile rotorun konumunu öğrenip mikrodenetleyici vasıtasıyla bir sonraki aşamada enerjilendirilmesi gereken mosfet çiftini enerjilendirmek için mosfet sürücüye gerekli komutlar gönderilir. Bu sayede doğru zamanda anahtarlanan 2 faz ile motor sürülmüş olur. Anahtarlama sıralaması Şekil 4.1'de gösterilmektedir.



Şekil 3 BLDC Motorun Çalışma Aşamaları

4.2.Sensörsüz Kontrol

Sensörsüz BLDC motorlar, Back EMF ölçümleri kullanarak rotoroun pozisyonunu kontrol ederler. Rotorun çembersel hareketine bağlı olarak, motorda oluşan Back EMF mikrodenetleyici tarafından incelenerek rotorun pozisyonu hakkında bilgi verilir. Sistem maliyeti ve karmaşıklığı düşürme açısından sensörsüz kullanım daha avantajlıdır. Ancak daha hassas ve pozisyonun ölçümünün daha önemli olduğu kullanım alanlarında sensörlü kullanım avantajlıdır.

4.3.BLDC Motorların Üstün ve Zayıf Yönleri

Tablo 4.1'de BLDC motorların endüstriyel piyasadaki diğer motorlardan üstün ve zayıf tarafları açıklanmıştır.

Tablo 1 BLDC Motorun Üstün ve Zayıf Yönleri

Özellik	Üstün Yönler	Zayıf Yönler
Verimlilik	- Yüksek verimlilik, fırça olmaması ve enerji kaybının düşük olması.	- Kontrol sistemi karmaşıklığı, sürücü devresi gereksinimi.
Bakım Gereksinimi	- Düşük bakım gereksinimi, fırçasız tasarım nedeniyle aşınma olmaz.	- Kontrol sistemini anlamak ve bakımı yapabilmek için özel bilgi gerektirir.
Hız ve Tork Kontrolü	- Yüksek hız ve tork kontrol kapasitesi.	- Kontrol algoritmalarının daha karmaşık olması.
Hızlanma ve Tepki Süresi	- Hızlı hızlanma ve tepki süreleri.	- Yavaş hızlanma ve tepki süreleri, özellikle düşük hızlarda.
Boyut ve Ağırlık	- Yüksek güç yoğunluğu, küçük boyut ve hafif ağırlık.	- Kontrol elektroniği ve sürücü devreleri, fiziksel hacim ve ağırlığı artırabilir.
Çalışma Ömrü	- Uzun çalışma ömrü, fırçasız tasarım aşınma olmadığı için.	- Kontrol elektroniği arızaları veya başarısızlıkları.
Maliyet	- Uzun vadede düşük maliyet, bakım ve aşınma maliyetlerinin azalması.	- Başlangıç maliyeti yüksek, özellikle sensörlü sistemlerde.

BLDC motorların en büyük zayıflığı maliyettir. Maliyetin sebebi fırçalı DC motorlardan farklı olarak sürücü elektroniği olmadan çalışamamasındandır. [8] [9]
Piyasada kullanılan CD sürücüden takım tezgahlarına ve CNC tezgahlarına kadar hassas iş yükünü karşılayabilen hız ve pozisyon kontrolü açısından vazgeçilmezdir. Dezavantaj olarak gözüken sürücü zorunluluğu hassas işler için BLDC motorları kaçınılmaz kılmaktadır.

4.4. Mosfet

MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor), yarı iletken cihazların bir türü olan bir transistör çeşididir. MOSFET'ler, elektronik devrelerde anahtarlama, amplifikasyon ve sinyal işleme gibi birçok uygulamada kullanılan önemli bileşenlerdir.

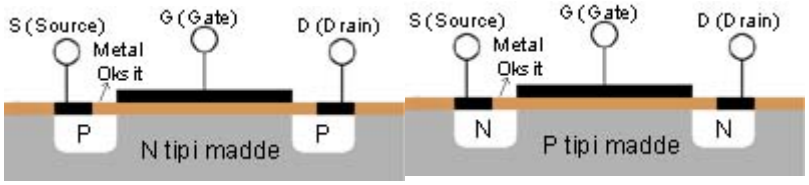
1. **Yapısı:**
    - MOSFET'in temel yapısı, metal (M), yalıtkan (O), ve yarı iletken (S) katmanlarından oluşur.
    - Metal kapı (gate), yalıtkan bir tabaka (genellikle silikon dioksit) ve yarı iletken bir kanal içerir.

2. **Çalışma Prensibi:**
    - MOSFET, kapı gerilimi (Vgs) ile kontrol edilen bir kanal üzerinden akım taşır.
    - Kapı gerilimi, yarı iletken kanalındaki taşıyıcıların (genellikle elektronlar) hareketini kontrol eder.

3. **N-Channel ve P-Channel:**
    - N-Channel MOSFET'lerde, kanal negatif taşıyıcıları (genellikle elektronlar) ile iletken hale gelir.

2007 KARABÜK ÜNİVERSİTESİ

- P-Channel MOSFET'lerde, kanal pozitif taşıyıcıları (genellikle boşluklar veya "holler") ile iletken hale gelir.



Şekil 4 N kanal ve P kanal mosfet [8]

4. **Çalışma Modları:**
    - MOSFET, üç temel çalışma modunda kullanılabilir: Kesim (Off), Doymuş (Saturation), ve Kesme Bölgesi (Triode).

5. **Kullanım Alanları:**
    - MOSFET'ler, bilgisayarlar, güç amplifikatörleri, güç kaynakları, radyo frekans devreleri, entegre devreler ve daha birçok elektronik uygulamada kullanılır.

MOSFET'ler, düşük güç tüketimi, hızlı anahtarlamalar, yüksek giriş direnci gibi avantajları nedeniyle birçok elektronik devre tasarımında tercih edilmektedir. N-Channel ve P-Channel MOSFET'ler, devre tasarımındaki ihtiyaçlara göre seçilir ve kullanılır.

4.5. Projede Kullanılan Mosfetin Özellikleri:

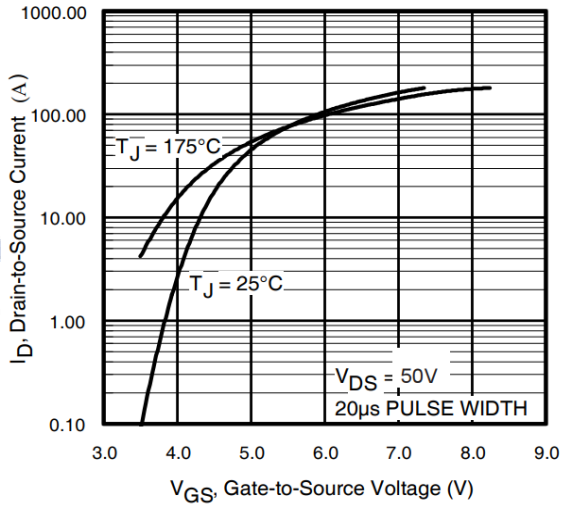
Fırçasız motor sürücü devremizde anahtarlama elemanı olarak kullandığımız mosfet; IRF3710L model mosfettir.

Bu mosfeti seçme nedenimiz olarak; [9]

- Yüksek bara gerilimi. Vgs = 100V
- Yüksek çalışma akımı. Oda sıcaklığında 57A, 100°C sıcaklık altında 40A
- Yüksek anlık akım taşıma kabiliyeti. Anlık 180A
- Düşük $R_{DS(on)}$ değeri. $R_{DS(on)}$= 23 mΩ
- Türkiye içinde kolay tedarik

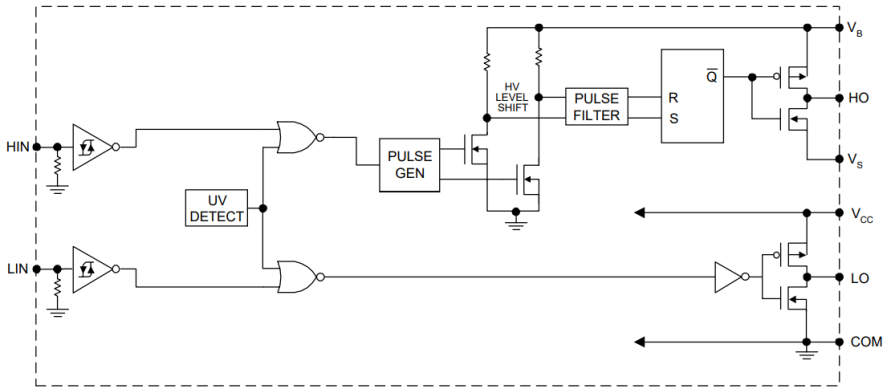
4.6. MosFET Sürücü

Mosfetleri iletime geçirmek için V_{th} voltajını vermemiz yeterlidir ama bu gerilim ile mosfet üzerinden geçecek drain akımı çok küçük kalır. Yüksek akımlar için yüksek gate-source voltajına ihtiyacımız vardır.



Şekil 5 Mosfet VGS-ID grafiği [9]

Mikroişlemcinin çıkışı ise en fazla 3.3 volt verebilir. Bu durumda mikroişlemciden gelen voltaj ile yüksek voltajı anahtarlama yapmamız gerekiyor. Bu işlemi ayrı transistör çiftleri ile yapmak yerine bu iş için özel olarak geliştirilmiş ve paket haline getirilmiş entregre kullandık.



Şekil 6 Mosfet sürücü iç yapısı [10]

Low-side mosfeti anahtarlamak bulunduğu konum yüzünden kolaydır. Mosfetin source ucu toprağa bağlı. Gate ucuna uygulanacak herhangi bir sıfırdan yüksek gerilim mosfetin tetiklenmesini sağlar. High-side mosfette ise source ucu toprağa bağlı olmadığı için source

ucundan daha yüksek bir gerilime sahip gate gerilimi ile tetiklenmelidir. Bu işlemi ise bootstrap denilen metot ile yapılır.

BootStrap Kapasitörleri: Mosfet sürücülerde bulunan BootStrap kapasitörü yüksek frekanslı anahtarlama (High-Side Switching) uygulamlarında kullanılan bir yöntemdir. Şekil 4.5'de de görüldüğü üzere BootStrap kapasitörü, Mosfetin yüksek taraflı gate sürücüsünü, Mosfetin düşük tarafındaki bir anahtar tarafından bir kaynaktan beslenir. Bu anahtar yüksek taraflı mosferin çalışma döngüsüne bağlı olarak düşük taraflı mosfeti anahtarlmaa işleminde kullanılır. Düşük taraflı mosfet kapalı olduğunda, BootStrap kapasitörü yüksek taraflı mosfetin gate'ini beslemekte. Böylece gate geriliminin source geriliminden her zaman BootStrap kapasitörünün gerilimi kadar fazla olmasını sağlar. [6]

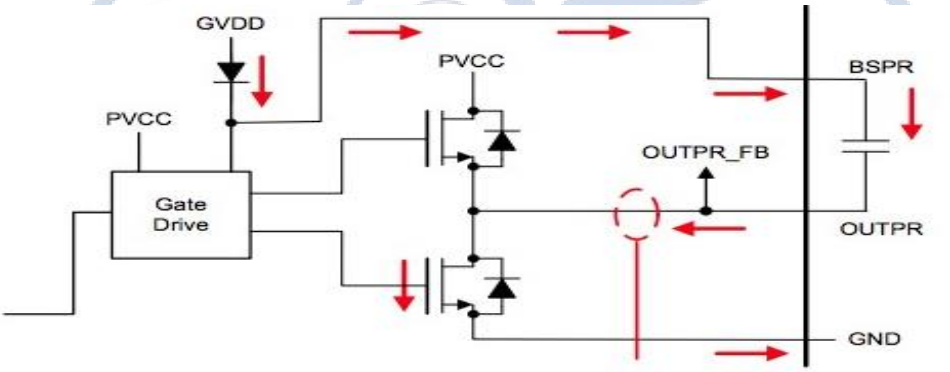


Figure 1. Bootstrap Capacitor Charging

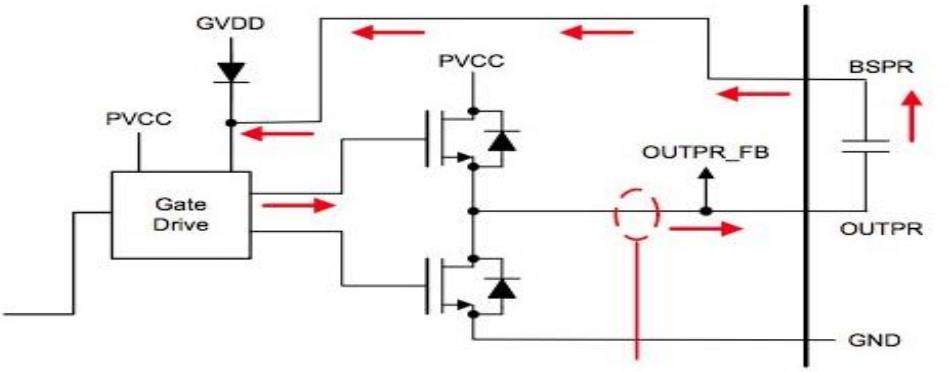


Figure 2. Bootstrap Capacitor Discharging

Şekil 7 Bootstrap çalışma mantığı

4.7 Sürücü Devresinde Kullanılan Mikrodenetleyici ve Özellikleri

Sürücü devresinin kontrolcüsü olarak tercih edilen STM32 Blue Pill, STMicroelectronics firması tarafından üretilen bir STM32 mikrodenetleyici tabanlı bir geliştirme kartıdır.

STMicroelectronics firması tarafından üretilen mikro denetleyiciler genellikle ARM Cortex-M çekirdekleri üzerine kurulmaktadır ve geniş çalışma alanında, endüstriyel ve gömülü sistem uygulamalarında kullanılmaktadır.

4.7.1. STM32 Blue Pill mikrodenetleyicisinin tercih edilme sebepleri:

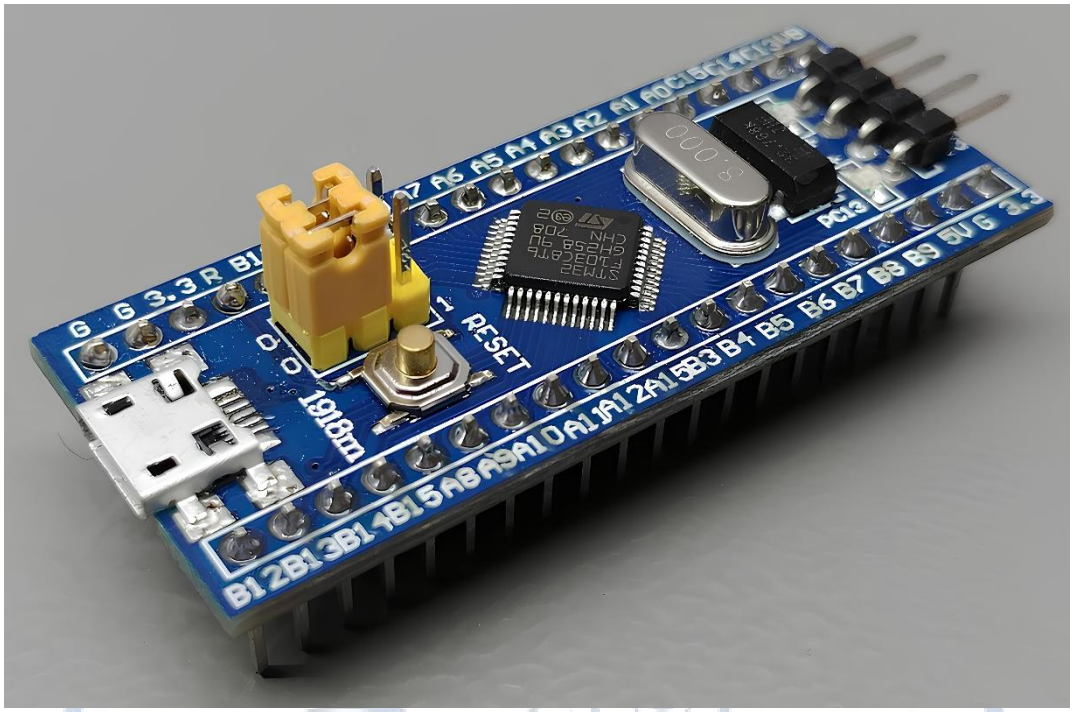
I. **Kolay montaj imkanı:**
Çalışması için gereken komponentleri üzerinde barındırması ve through-hole olması ve baskı devre kartına (PCB) kolay montajlanabilmesi sebebiyle tercih edildi.

II. **Kolay programlanabilme ve Hata ayıklama (Debuging):**
Üzerinde bulunan USB portunun yanı sıra UART haberleşme protokolü ile mikrodenetleyici içine program yüklenebilir.

III. **Ayarlanabilir giriş çıkış pinleri:**
Mikrodenetleyici üzeirnde bulunan GPIO (General Purpose Input/Output) pinlerini gereken konfigürasyonlar için kolay bir şekilde istenen amaca uygun bir şekilde kullanılır.

IV. **Dökümantasyon ve topluluk desteği:**
STMicroelectronics firmasının topluluk desteği olarak kullanıcıların birbirlerine yardımcı olabilmesi adına oluşturulan forumlar ve topluluk destek sayfaları ile yazılım konusunda yardımcı olmaktadır. Aynı şekilde STMicroelectronics firması tarafından üretilen kartların bilgi kitapçığı (datasheet), kullanım kitapcığı (user manual) ve uygulama notları sayesinde tasarım ve programlama kolaylaşmaktadır.

V. **Programlama arayüzü (IDE):**
Üretici firma tarafndan sunulan ve desteklenen STM32CubeMX ve STM32CubeIDE yanı sıra KeiluVision programları ile kolay bir şekilde mikrodenetleyici programlanır.



Şekil 8 Sürücüde kullanılan mikrodenetleyici [11]

4.7.2. Mikrodenetleyici içerisinde bulunan mikroişlemcinin başlıca özellikleri:

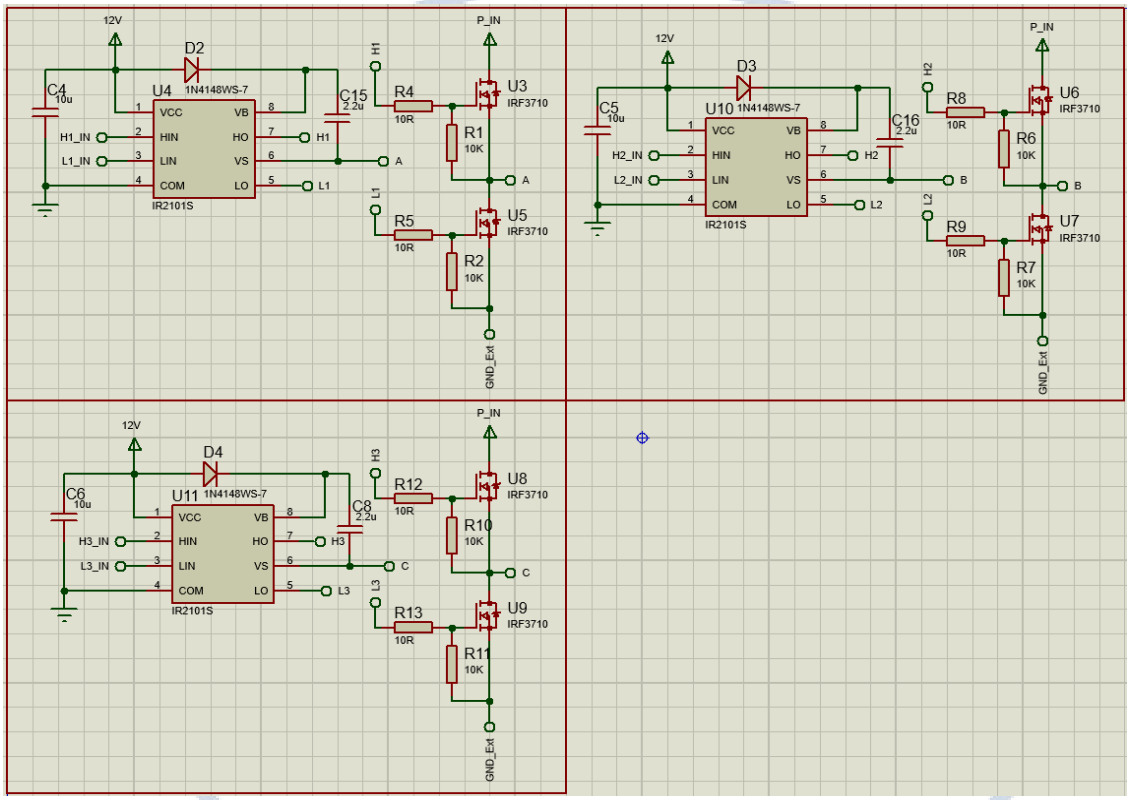
- ARM Cortex-M3 çekirdeği, 32-bit RISC mimarisi.
- 72 MHz çalışma frekansı.
- 64 Kbyte hafıza.
- 20 Kbyte SRAM
- 12 bit ADC
- I^2C, SPI, USART ve CAN-bus haberleşme protokol desteği

KARABÜK ÜNİVERSİTESİ

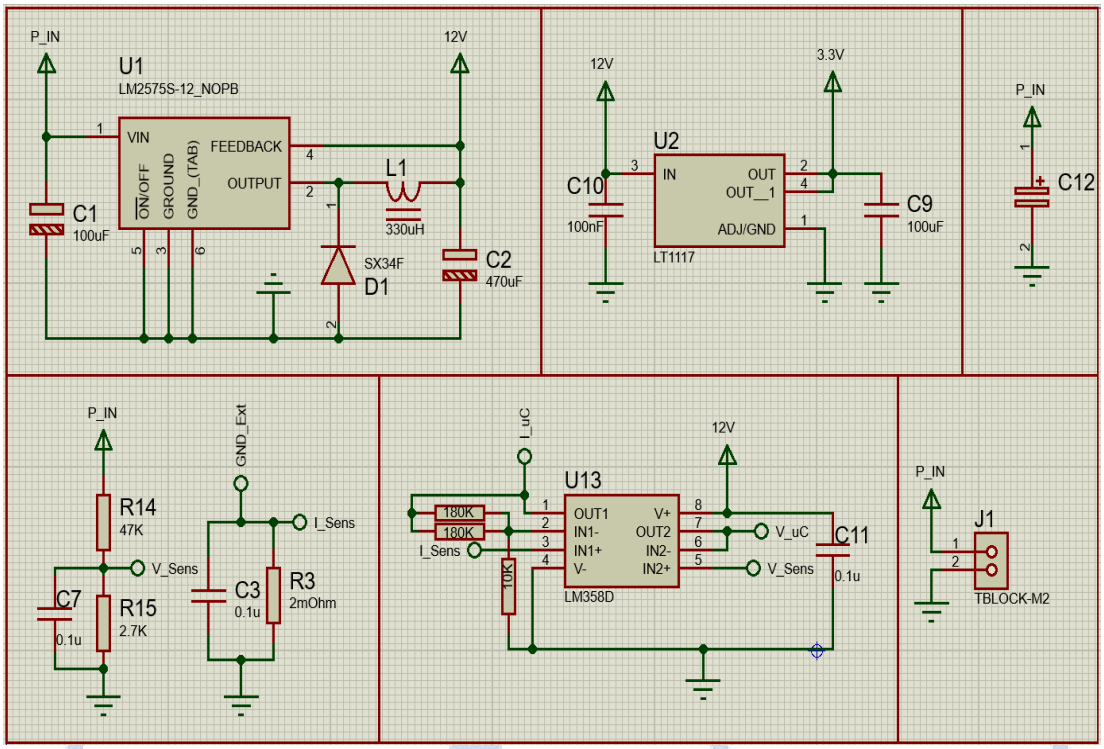
4.8. Devre Şematiği ve PCB çizimleri

IR2101S mosfet sürücü kullanıldı. Mosfet Sürücü Mikrodenetleyiciden gelen PWM sinyalleri ile IRF3710 mosfetlerinin çalışmasını sağlar. Şekil 4.7'de gözüken C8, C15 ve C16 Kapasitörleri Bootstrap görevi görmektedir.

R1, R2, R6, R7, R10 ve R11 dirençleri boşta iken mosfetin kapanmasını sağlamak için yerleştirilen gate-source dirençleridir. [6]



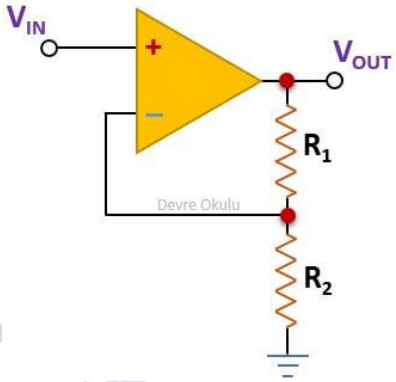
Şekil 9 Mosfet Sürücü ve Mosfetlerin Devre Şeması



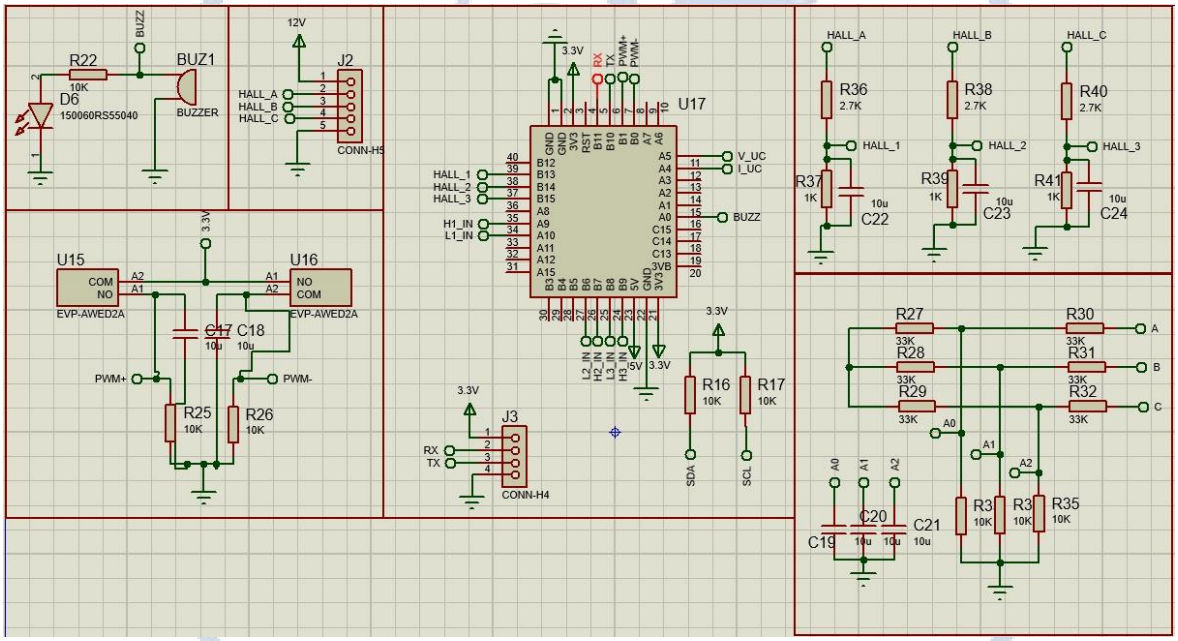
Şekil 10 Şematiğin Güç Bloğu

- LM2575 Beslemeden gelen gerilimi mosfet sürücü ve opampların çalışma voltajı olan 12 volta indirir. LM7812 gibi basit modül yerine bu entegreyi kullanma sebebimiz ise yüksek giriş gerilimlerinde de çalışmaktır. 60 volta kadar olan gerilimlerde rahatlıkla çalışmaktadır.
- LT1117 voltaj regülatörünün çıkışı mikrodenetleyicinin beslemesi olarak kullanılmaktadır. 12V'u 3.3V'a indirerek STM32F103C8T6 mikrodenetleyicisinin gerilim seviyesinin aşmamaktadır.
- LM358D OP-AMP'ı ile motorun gerilim ve akım seviyesini yaklaşık olarak 10 katına çıkararak düşük seviyedeki değerleri ölçmeyi sağlar.
- R14 ve R15 ile oluşturulan voltaj bölücü ile besleme gerilimi mikroişlemci ile kontrol edilip istenilen değerin altına düştüğünde sistemi durdurmayı amaçladık.
- R3 direnci ile çekilen akımı opamp ile 10 kat yükseltip mikroişlemci ile okumayı planladık. Bu sayede istenilen değeri geçtiğimiz zaman sistem çalışmayı durdurabilecek
- C12 besleme kondastörü ile anlık akım çekme durumunda sistem geriliminin sabit kalmasını sağladık.

$$A_v = 1 + \frac{R_1}{R_2}$$

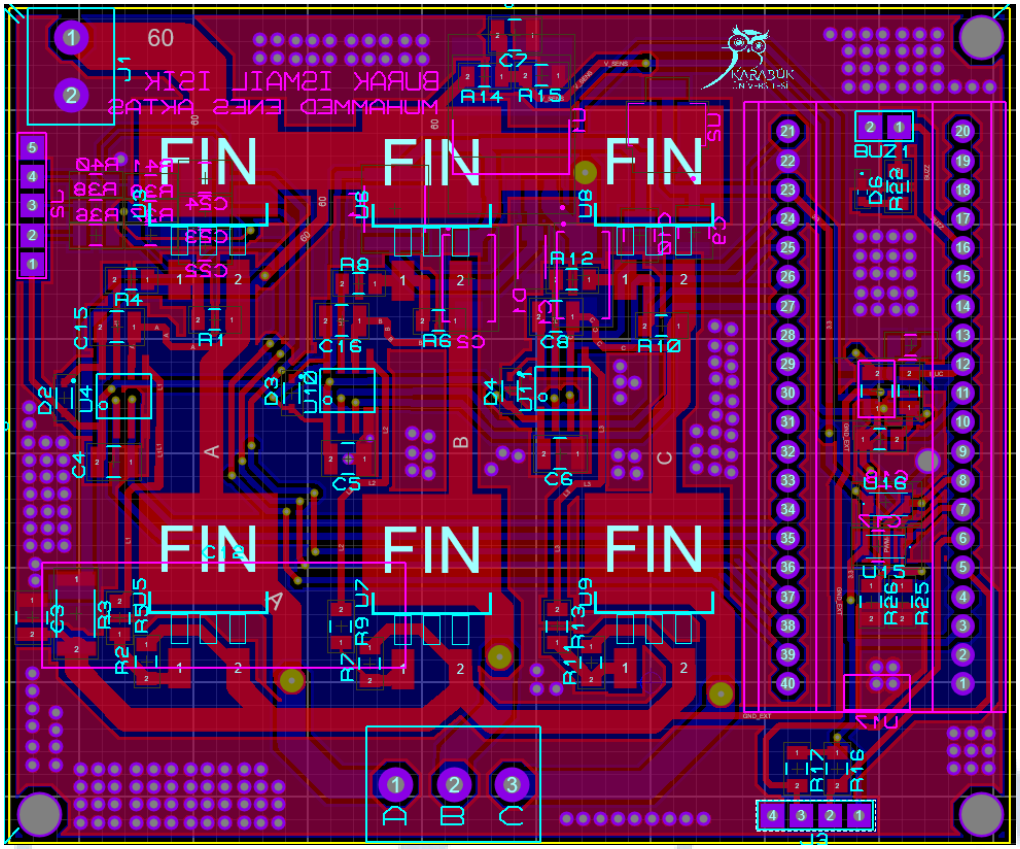


Şekil 11 Evirmeyen OP-AMP



Şekil 12 Mikrodenetleyici Bloğu

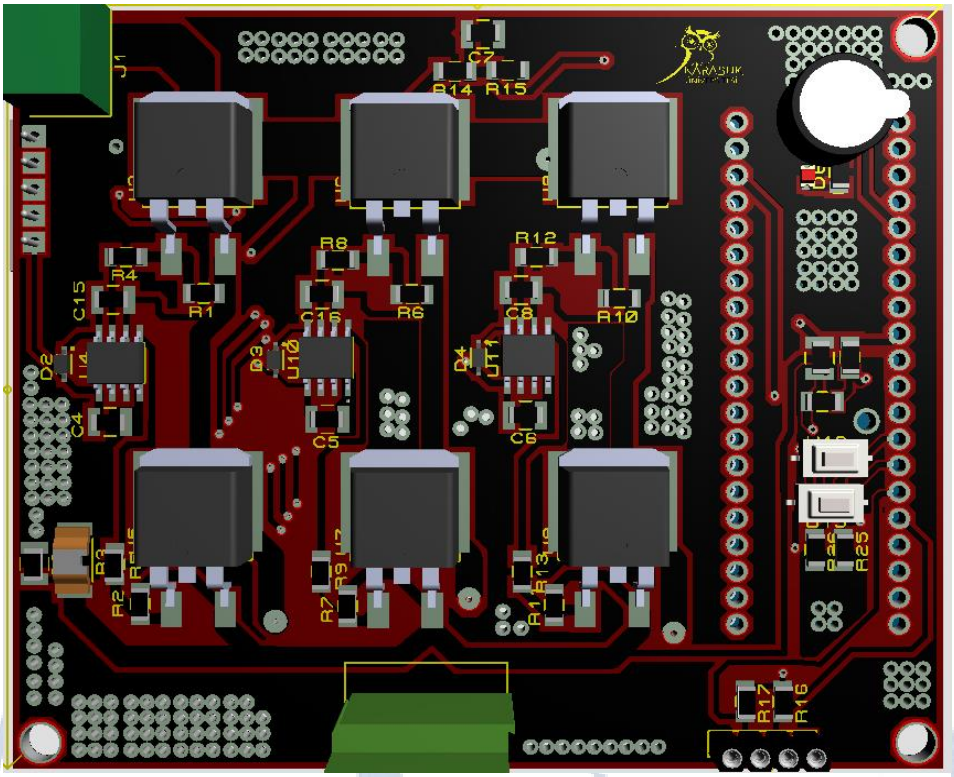
- D6 ledi ve buzzer ile hem sesli hem de görsel olarak hata bilgilendirmesi yapıldı.
- U15 ve U16 butonları ile motor hız kontrolünü dijital olarak kontrol edildi.
- R36 ile R41 aralığındaki dirençler ile oluşturduğumuz gerilim bölücü ile hall-effect sensöden gelen değerleri mikroişlemciye girdik.
- J3 headder aracılıyla seri port üzerinden debug ve kullanım sırasında RPM, gerilim, çekilen akım, duty cycle gibi değerleri yazılımla işleyip göstermeyi amaçladık.



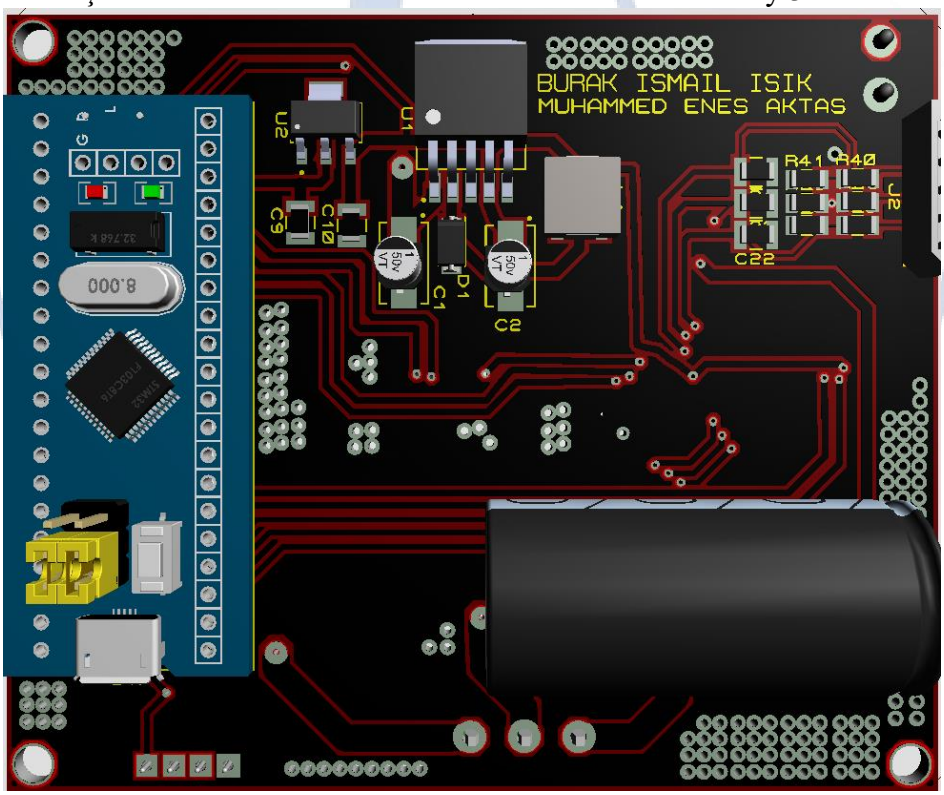
Şekil 13 ESC'nin PCB tasarımı

- Elektrostatik etkileri en aza indirmek için ground plane kullanıldı.
- Alt katman ile üst katman arasındaki iletimi sağlamak için via kullanıldı
- Mosfetler bir yüzde toplandı. Bu sayede eğer ihtiyaç duyulursa pasif soğutma kullanılabilecek.
- Motor, besleme, hall-efect ve seri port bağlantıları headder ve klemens ile yapıldı.

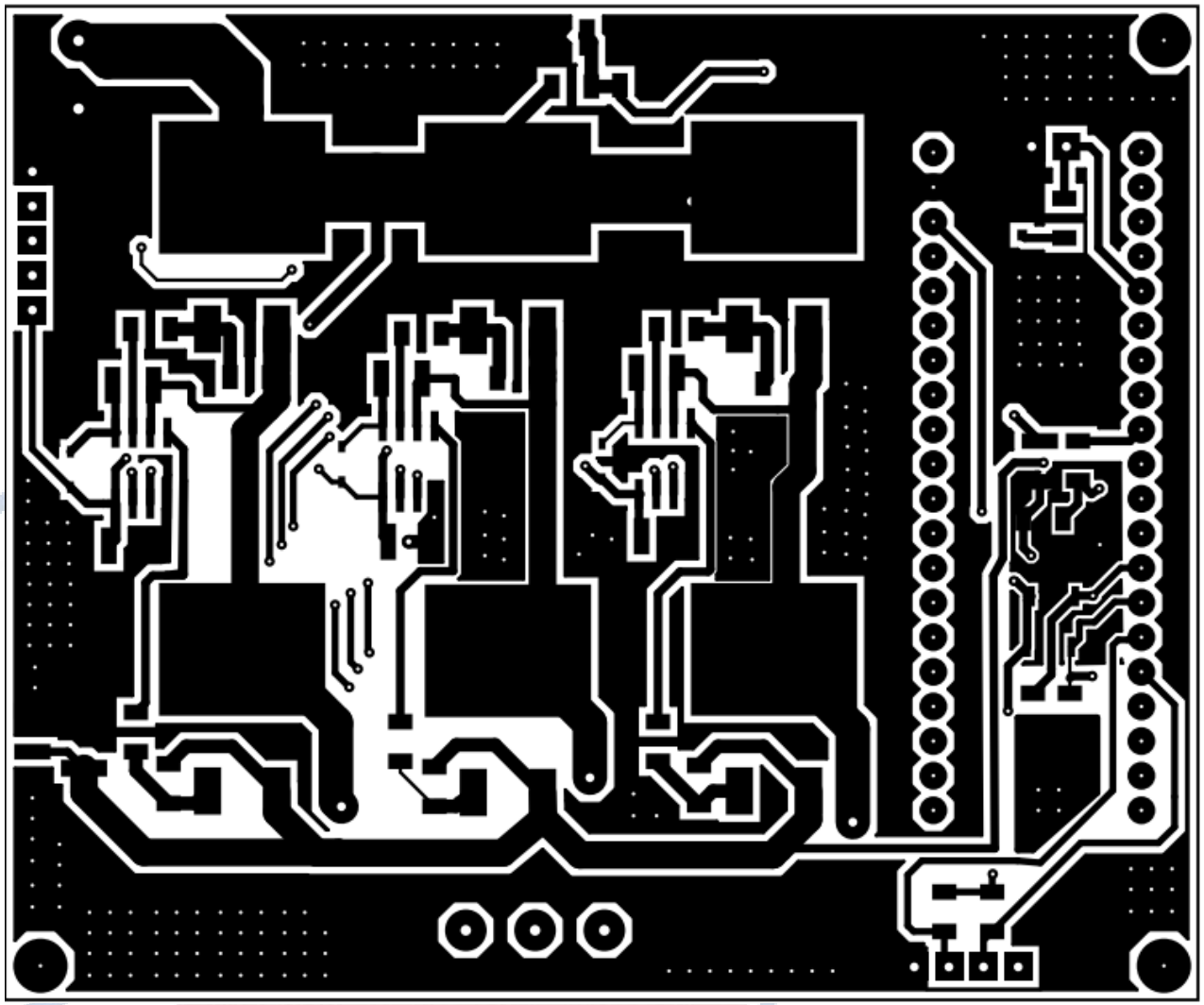
Daha profesyonel bir görünüş ve en önemlisi minimum hataya sahip bir kart elde edebilmek için elle baskı yerine PCB baskı sektöründe öncü firmalardan olan PCBway ile kartın baskısı yapıldı. [10]



Şekil 14 BLDC Motor Hız Kontrolcüsünün Üst Yüzey 3D Görünümü

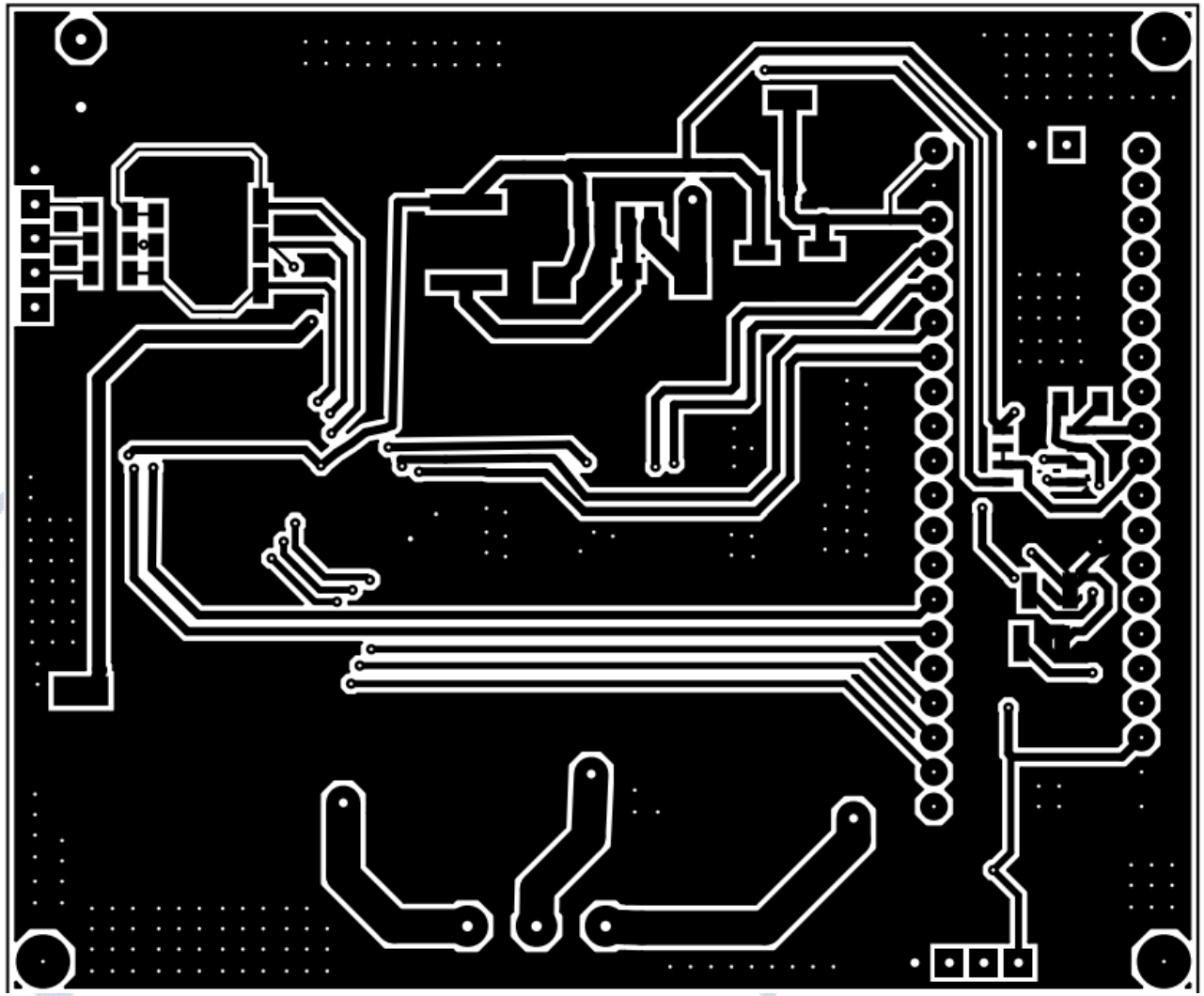


Şekil 15 BLDC Motor Hız Kontrolcüsü Alt Yüzey 3D Görünümü



Şekil 16 PCB üst katman

ÜNİVERSİTESİ



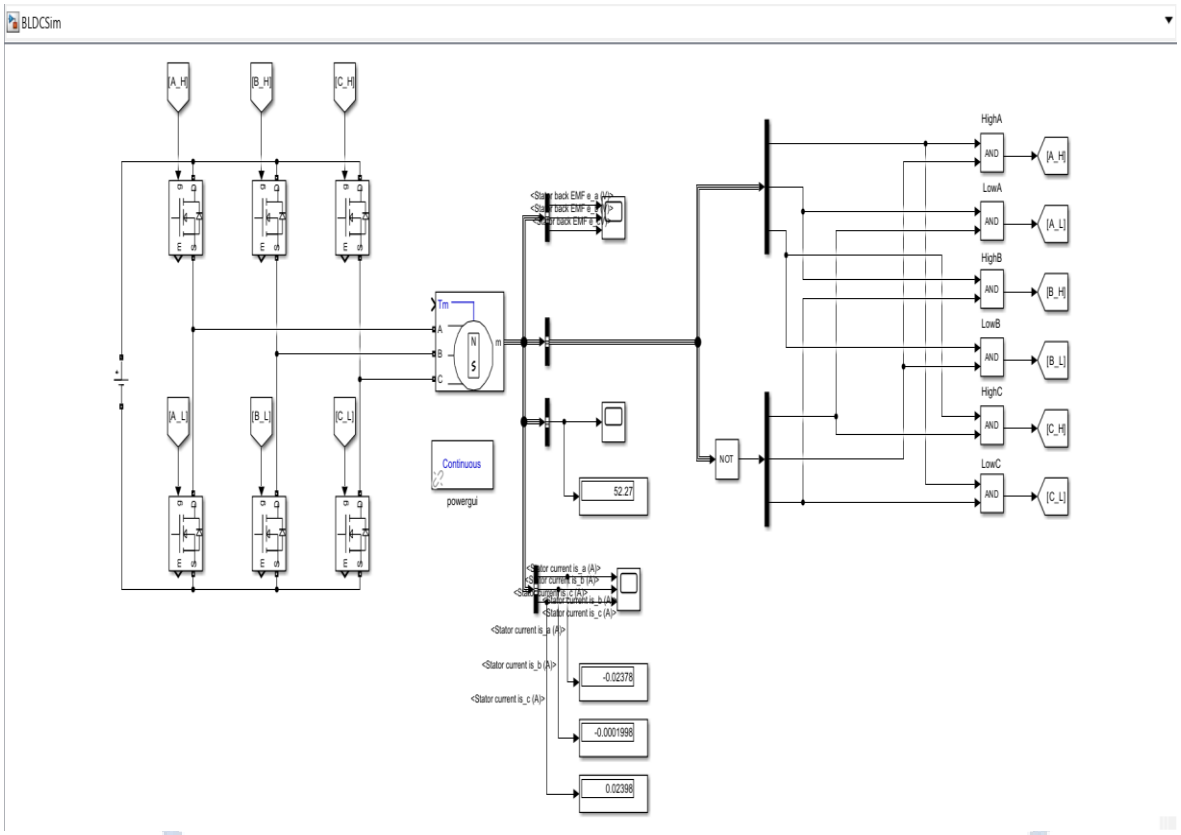
Şekil 17 PCB alt katma

ÜNİVERSİTESİ

BÖLÜM 5

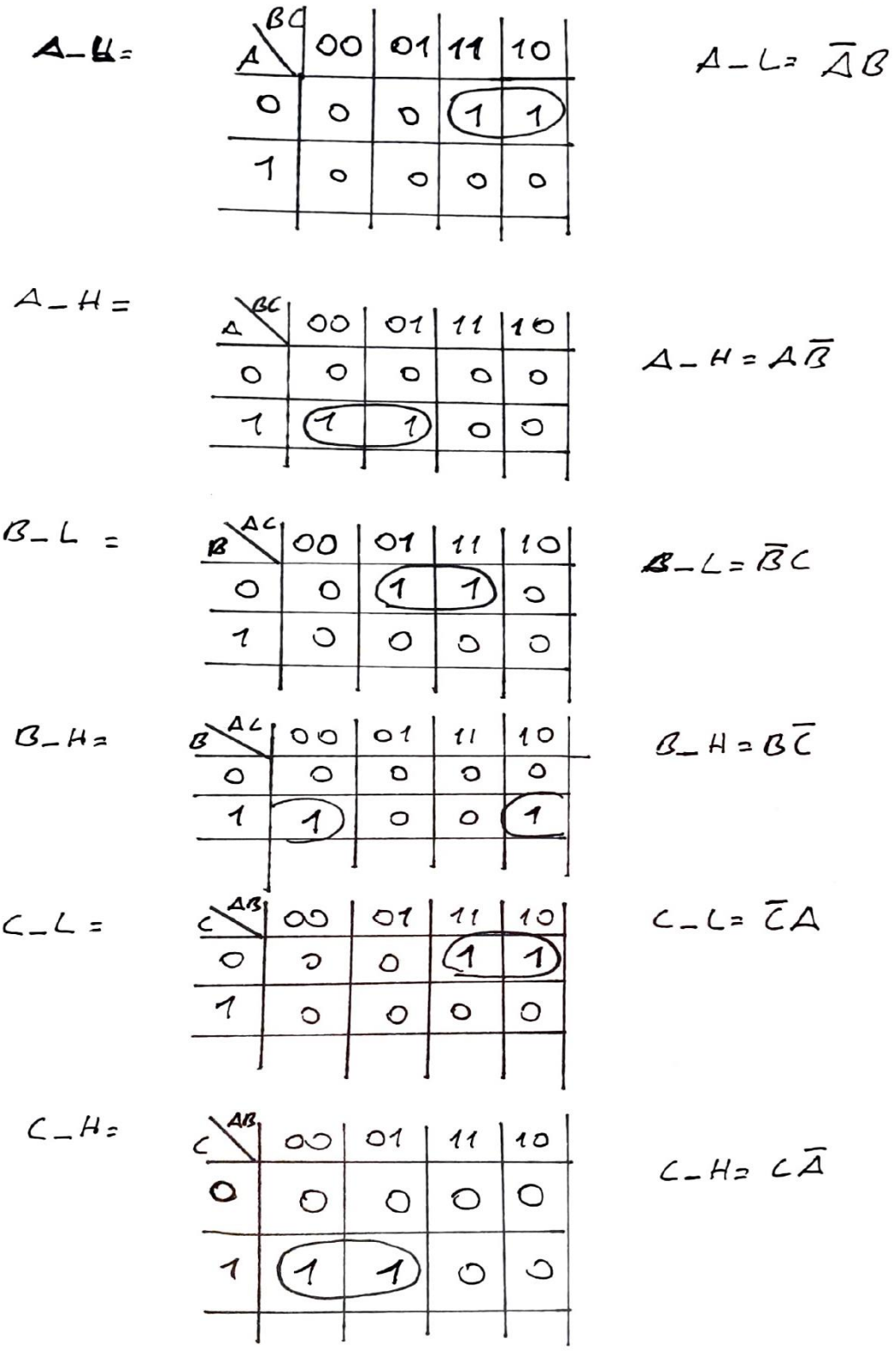
SİMÜLASYON VE SONUÇLARI

5.1.Simulink ile BLDC Motor Hız Kontrolü



Şekil 18 Simulink BLDC Hız Kontrol Simülasyonu

Şekil 5.1'de görülen Blok diagramda lojik kapılar kullanılarak ve karno tablosu oluşturularak her sensör için stator akımları ve Back EMF değerleri bulunarak analiz edildi.



Şekil 19 Hall Sensörünün Karno Tablosu

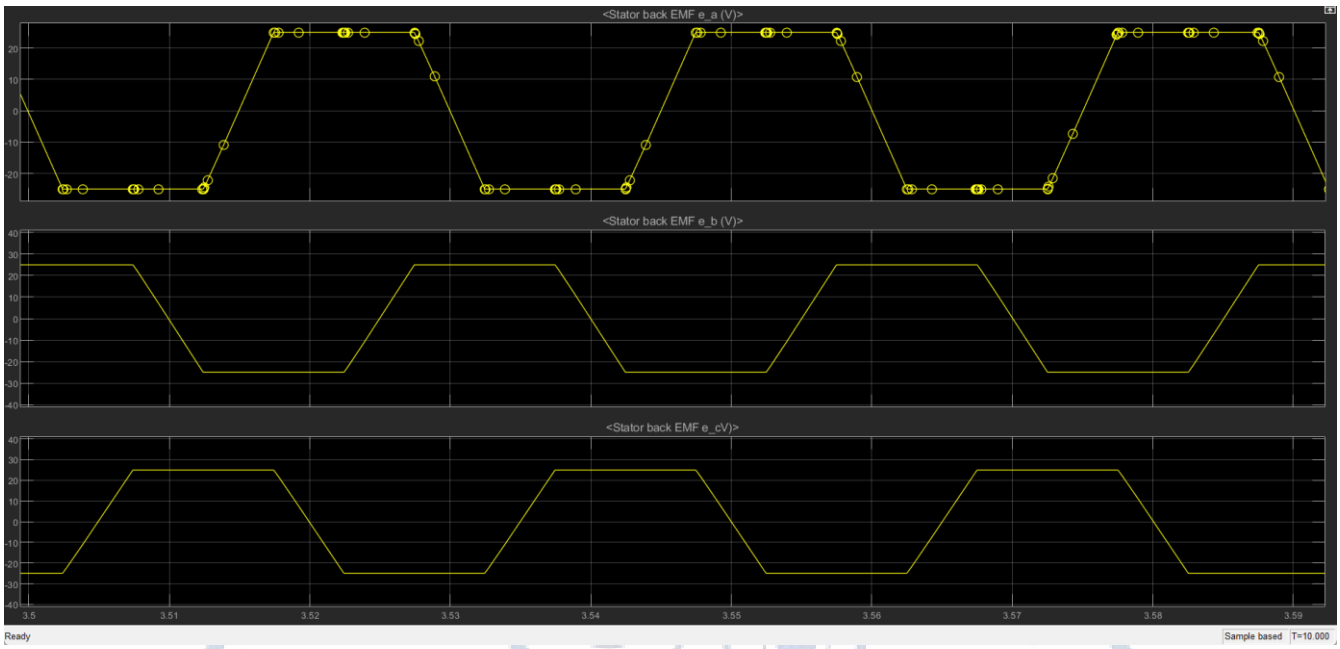
Karno tablosu kullanarak motorun saat yönünde dönmesini sağlayacak olan Hall Sensör giriş durumları hesaplandı. Şekil 5.2' de görüldğü gibi her faz için yüksek ve düşük olacak şekilde 1'er tane tablodan oluşan hesaplama Hall Sensörlerinin hangi sıra ile tetikleneceğini göstermektedir.

	Hall sensors			Switchs						Phases			Windings		
Phase	H3	H2	H1	Q1L	Q1H	Q2L	Q2H	Q3L	Q3H	P1	P2	P3	V_{1-2}	V_{2-3}	V_{3-1}
I	1	0	1	0	1	1	0	0	0	$+V_m$	Gnd	NC	$-V_m$	-	-
II	0	0	1	0	1	0	0	1	0	$+V_m$	NC	Gnd	-	-	$+V_m$
III	0	1	1	0	0	0	1	1	0	NC	$+V_m$	Gnd	-	$-V_m$	-
IV	0	1	0	1	0	0	1	0	0	Gnd	$+V_m$	NC	$+V_m$	-	-
V	1	1	0	1	0	0	0	0	1	Gnd	NC	$+V_m$	-	-	$-V_m$
VI	1	0	0	0	0	1	0	0	1	NC	Gnd	$+V_m$	-	$+V_m$	-

Şekil 20 Komütasyonun Gerçekleşme Sırası

Şekil 5.3'de görüldüğü üzere Motorun hareketinin ve komütasyonunun saat yönünde sağlanabilmeesi için yukarıdaki adımların takip edilmesi gerekmektedir. Fazların enerjilendirilme sıralaması, Hall Sensör bilgileri ve mosfetlerin durumları yukarıdaki tabloda görülmektedir.

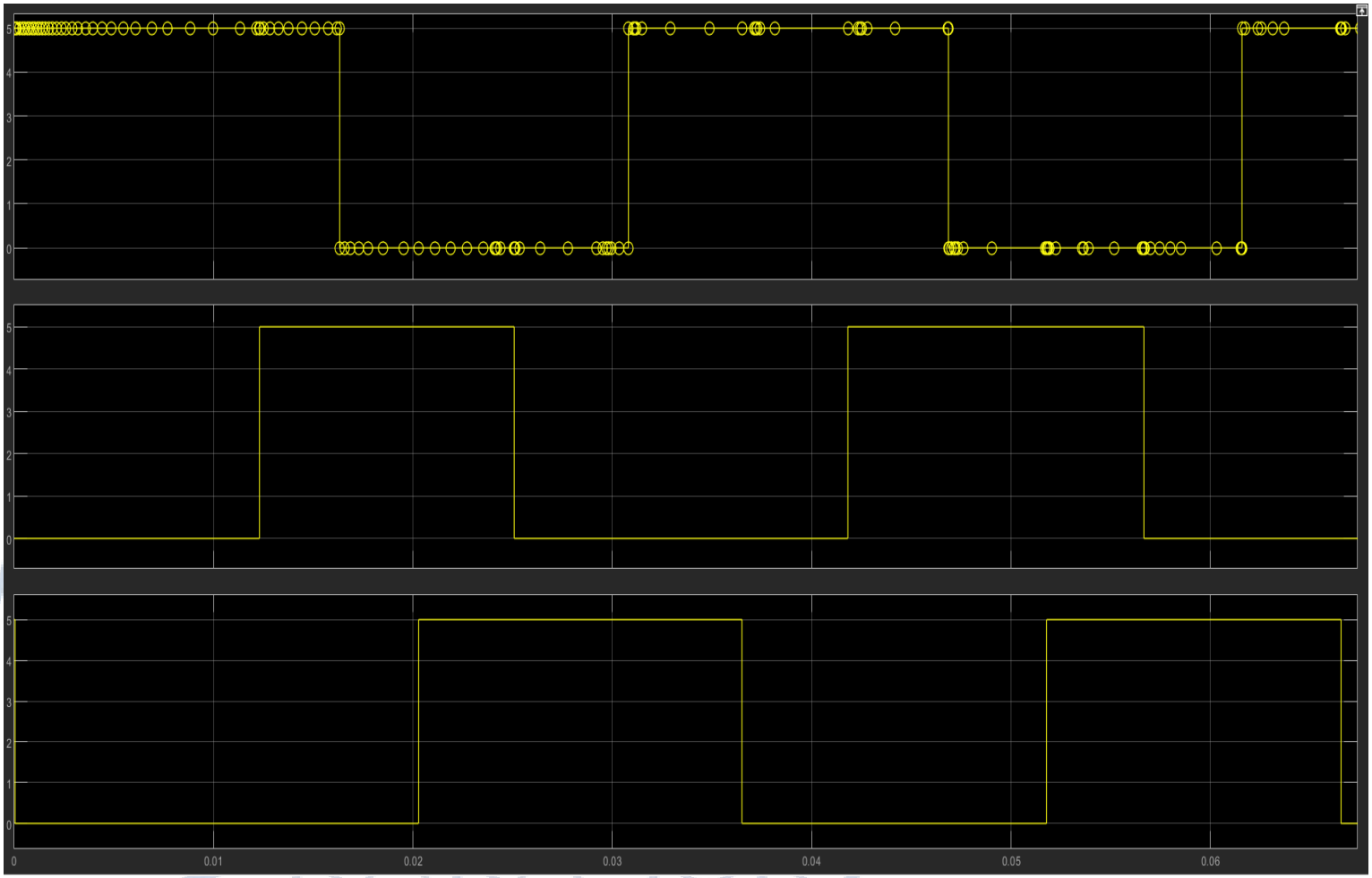
2007 KARABÜK ÜNİVERSİTESİ



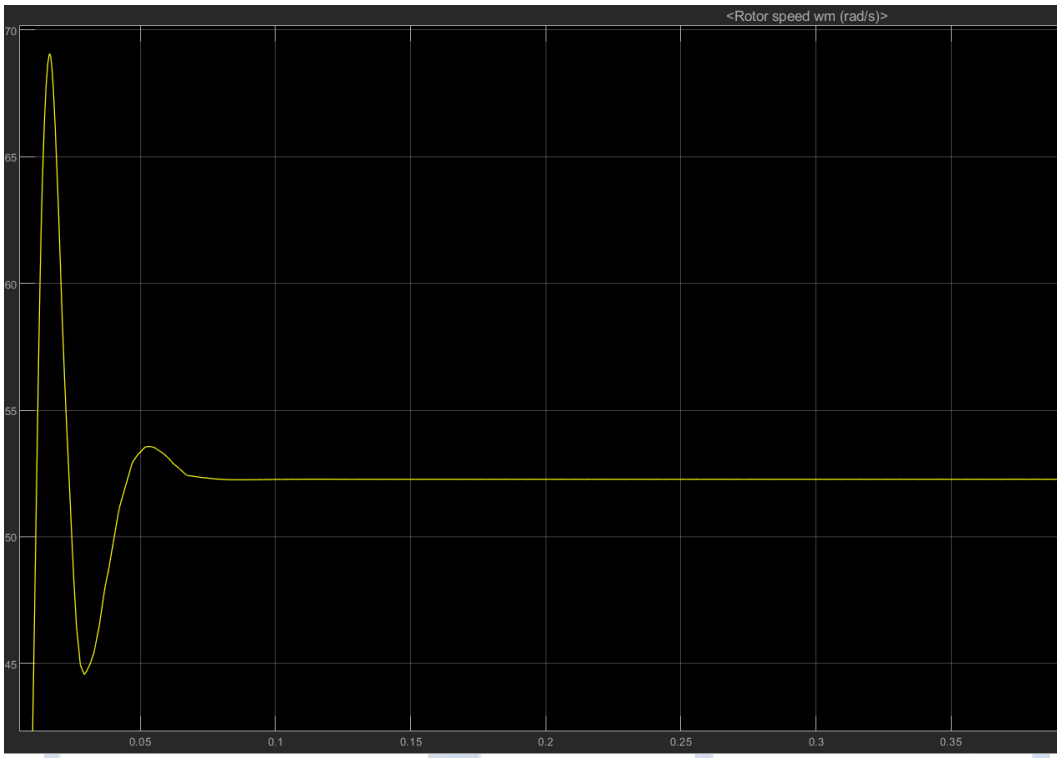
Şekil 21 Stator Back-EMF Değerleri

Şekil 5.4'de de görüleceği üzere motorun uçlarından alınan Back-EMF gerilimleri görülmektedir. Motor uçlarının enerjilendirilebilmesi için aralarında faz farkı bulunması gerekmektedir.

2007 KARABÜK ÜNİVERSİTESİ



Şekil 22 Hall-Effect Sensörü Çıkış Değerleri



Şekil 23 Rotor Hızı

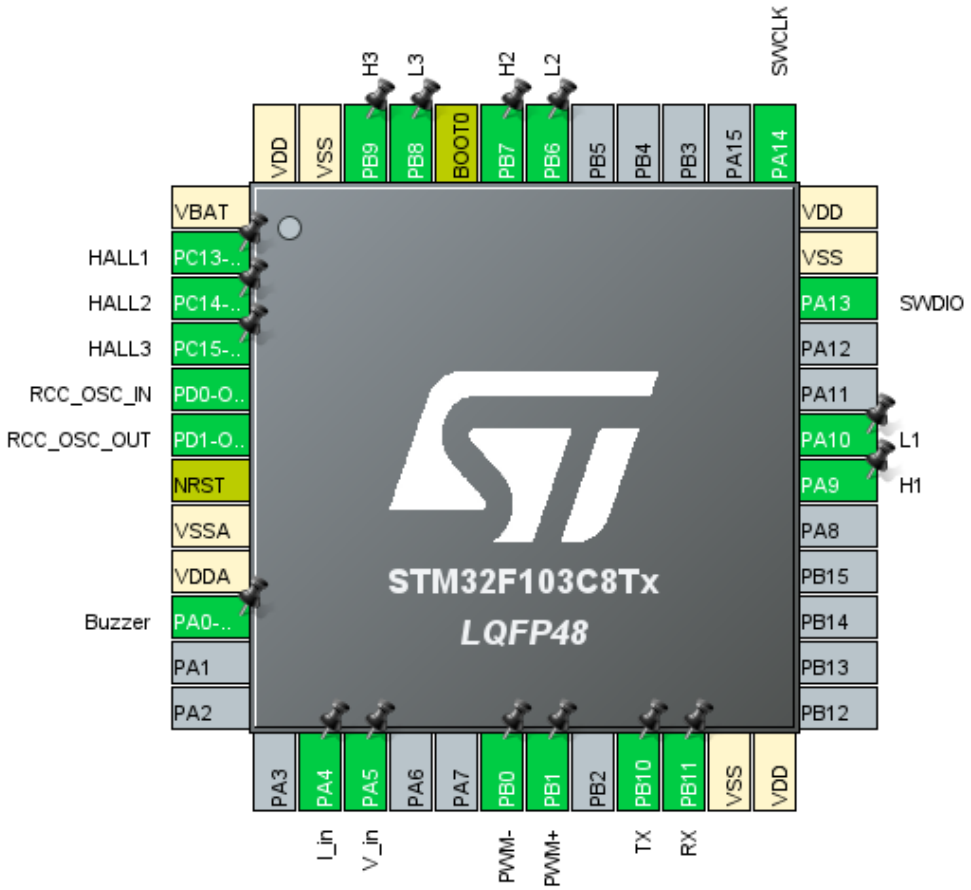
Yukarıdaki grafikte rotor hızı görülmektedir.

$$N = \frac{120 \times f}{P}$$

Bu denklemde;

- *N*, rotor hızını temsil eder (devir/dakika cinsinden),
- *f*, motorun çalışma frekansını temsil eder (Hz cinsinden),
- *P*, kutup sayısını temsil eder.

5.2.Mikroişlemci ve Yazılım Sonuçları



Şekil 24 STM32CubeIDE Programı ile Pin Konfigürasyonu

- PC13, PC14, PC15 Hall sensör interrupt pinleri
- PD0, PD1 İşlemci harici kristal pinleri
- PA0 Kullanıcı bilgilendirmesi için kullanılan Buzzer
- PA4, PA5 ADC pinleri devredeki akım ve gerilimi hesaplar
- PB0, PB1 GPIO Duty Ratio ayar pinleri
- PB10, PB11 İşlemci ile kullanıcı habeleşmesini sağlayan seri port
- PA9, PA10, PB6, PB7, PB8, PB9 Mosfet sürücü pinleri
- PA13, PA14 Debug pinleri

```
void basla(void){
//1.Blok
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0); //L1
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);  //L2
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);  //L3
  __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0); //H1
  __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,0); //H2
  __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0); //H3

//2.Blok
  __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,100); //H1
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);  //L2
  HAL_Delay(10);

  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);  //L2
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);  //L3
  HAL_Delay(10);

  __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0); //H1
  __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,100); //H2
  HAL_Delay(10);

  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);  //L3
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1); //L1
  HAL_Delay(10);

  __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,0); //H2
  __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,100); //H3
  HAL_Delay(10);

  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0); //L1
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);  //L2
}
```

Tablo 2 Motora Manuel Yol Verme

Kullanılacak motorun Hall sensör durumları bilinmediği için mosfetleri manuel bir şekilde başlatarak Hall sensör durumları tespit edilebilir.
Tablo 2'de bulunan kod bloğunda mosfetler manuel olarak başlatılmıştır.

1.Blokdaki kodların amacı, devre kartında bulunan mosfetleri kontrol eden mosfet sürücülerinin girişlerine dijital '0' vererek pasif duruma getirilmesini sağlar. Bu durum mosfetlerin iletime kapalı olmasını sağlar.

1.Blok Mosfetlerin önceki durumları bilinmediği için ilk sürüş durumunda bütün mosfetler iletime kapatılarak kontrol altında istenilen mosfetin iletime açılması ile motora yol verme aşaması tamamlanır.

H1	L2
H1	L3
H2	L3
H2	L1
H3	L1
H3	L2

Tablo 3 Mosfet Enerjilendirme Sıralaması

2.Blok da ise motorun saat yönünde harekete başlayabilmesi gereken sargı enerjilendirme sıralaması Tablo 3'te gösterilmiştir. Durum değişikliği meydana gelen mosfet grupları aynı renk ile gösterilmiştir. Low-side mosfetleri PWM sinyali ile kontrol edilmesine gerek duyulmadığı için dijital 1 veya 0 kullanarak kontrolü sağlanmıştır. High-side mosfetler de ise motor kontrolü için PWM sinyali şarttır. Her komütasyon ardından kullanılan **Hal_Delay(10)** fonksiyonu ile mosfetlerin açılma ve kapanma anında çakışmaması için kullanılan bir gecikmedir.

```
__HAL_TIM_SetCompare(&htim,TIM_CHANNEL,100);
```

High-side mosfetlere uygulanan PWM sinyal değerinin Duty ratio değerini belirlemek için kullanılan fonksiyondur. Çözünürlük değerini 0-299 arasında tutarak %33'lük oranında bir PWM sinyal üretilmesini sağlar.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
//1.Blok
   HALL1=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
   HALL2=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_14);
   HALL3=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);   //L1
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);   //L2
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);   //L3
   __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);   //H1
   __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,0);   //H2
```

```
   __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);   //H3
//2.Blok
   if(HALL1&&!HALL2&&HALL3){//101
         step=4;
      }
   if(HALL1&&!HALL2&&!HALL3){//100
         step=5;
      }
   if(HALL1&&HALL2&&!HALL3){//110
         step=6;
      }
   if(!HALL1&&HALL2&&!HALL3){//010
         step=1;
      }
   if(!HALL1&&HALL2&&HALL3){//011
         step=2;
      }
   if(!HALL1&&!HALL2&&HALL3){//001
         step=3;
      }

   switch(step)
   {
      case 1:
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
           __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,0);//BH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//BL=H
          __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,pwm_value);//AH=H PWM
      break;
      case 2:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
         __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,0);//BH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);//CL=H
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,pwm_value);//AH=H PWM
      break;
      case 3:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
           __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
           __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);//CL=H
           __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,pwm_value);//BH=H PWM
      break;
```

```
      case 4:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
          __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);//AL=H
          __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,pwm_value);//BH=H PWM
      break;
      case 5:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=H
         __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,0);//BH=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);//AL=H
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,pwm_value);//CH=H PWM
      break;
      case 6:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
         __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_2,0);//BH=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//BL=H
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,pwm_value);//CH=H PWM
      break;
   }
}
```

Tablo 4 Hall Sensör ile Kontrol

Tablo 2 kodlarının çalışmasının ardından motor hareketi ile Hall sensörlerinin durumları öğrenilir. Interrupt pinlerine bağlı olan motorun Hall sensör uçları, algınalarak istenilen duruma geldiğinde. Çalışması gereken fonksiyon vasıtası ile gerekli olan mosfet grupları tetiklenir.

2.kod bloğunda saat yönünde dönüş sağlanabilmesi için durum sorguları gerçeklenir. Ardından Switch-Case fonksiyonu ile de doğru sıradaki mosfetler tetiklenir.

"**pwm_value**" değeri ile Şekil 24'te gösterilen PB0 ve PB1 pinlerine bağlı butonlar yardımı ile Duty Ratio değiştirlerek motor hızı kontrol edilebilir. Ayrıca seri port üzerinden de Duty Ratio değiştirilebilir.

```
void uart_gonder(char *deger)
{
  HAL_UART_Transmit(&huart3, (uint8_t*)deger, strlen(deger), 100);
}
```

Tablo 5 Seri Port Haberleşmesi

```
int main(void)
{
//1.Blok
  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);//A PWM START
  HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);// B PWM START
  HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_4);// C PWM START
  basla();

  while (1)
  {
//2.Blok
    if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0)==1)
        {
          pwm_value=pwm_value-30;
          if(pwm_value<20)
          {
            pwm_value=20;
          }
          HAL_Delay(500);
        }
        if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_1)==1){
          pwm_value=pwm_value+30;
          if(pwm_value>300){
            pwm_value=280;
          }
          HAL_Delay(500);
        }
//3.Blok
      HAL_UART_Receive (&huart3, &Rx_data, 8, 100);
      if(Rx_data=='1')
      {
          pwm_value=pwm_value+30;
          if(pwm_value>280){
            pwm_value=280;
          }
          HAL_Delay(500);
      }
      if(Rx_data=='0'){
```

```
          pwm_value=pwm_value-30;
          if(pwm_value<20){
            pwm_value=20;
          }
          HAL_Delay(500);
      }
//4.Blok
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 100);
        raw =HAL_ADC_GetValue(&hadc1);
        volt =(((( raw *3.3)/4096)*53.8)/2.7);
        HAL_ADC_Stop (&hadc1);
        HAL_Delay(2);
        HAL_ADC_Start(&hadc2);
        HAL_ADC_PollForConversion(&hadc2, 100);
        raw1 = HAL_ADC_GetValue(&hadc2);
        akim =((( raw1 *3.3)/2048)/96)*100;
        HAL_ADC_Stop (&hadc2);
        sprintf(buffer, "%f*%f*%d\r\n", volt, akim, pwm_value);
        uart_gonder(buffer);
  }
}
```

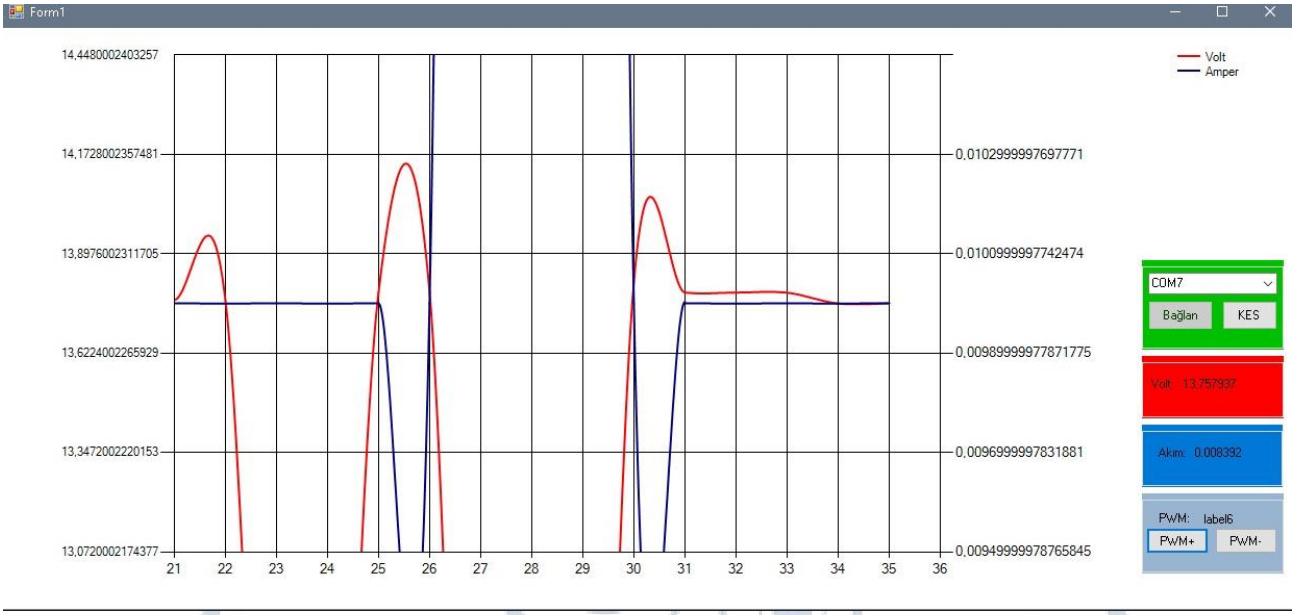
Tablo 6 Main Fonksiyonu

Tablo 6'te işlemcinin çalışması ile işleyecek olan kodlar görülmektedir. 1.kod bloğunda High-Side mosfetleri kontrol edebilmek için gerekli olan PWM sinyallerini oluşturabilmek için Timer pinleri başlatılır. Ardından "**basla();**" kodu ile Tablo 2'de anlatılan "**void basla(void)**" fonksiyonu çağrılarak motor manuel olarak çalışmaya başlar.

2.kod bloğunda sistemde bulunan butonlar aracılığı ile Duty Ratio ayarlanır. 3.blokta ise aynı işlem seri port üzerinden gerçekleştirilebilir.

4.kod bloğunda ADC kanalları kullanılarak hesaplanan motorun gerilim ve akım değerleri anlamlandırılarak Tablo 5'te bahsedilen fonksiyon vasıtası ile kullanıcı arayüzüne gönderilir.

5.3. C# ile Tasarlanan Kullanıcı Arayüzü



Şekil 25 Kullanıcı Arayüzü

```
private void Form1_Load(object sender, EventArgs e)
        {
            Control.CheckForIllegalCrossThreadCalls = false;
            foreach (var seri in SerialPort.GetPortNames())
            {
                comboBox1.Items.Add(seri);
            }
            comboBox1.SelectedIndex = 0;
            button2.Enabled = false;
            button4.Enabled = false;
            button3.Enabled = false;
            label1.Text = "";
            label2.Text = "";
        }
```

Tablo 7 Main Fonksiyonu

Tablo 7'de bulunan kod bloğunun çalışması ile "**foreach()**" fonksiyonu sayesinde bilgisayara bağlı olan seri portları comboBox'a kaydeder. comboBox içindeki ilk değeri varsayılan olarak gösterir.

```
private void button1_Click(object sender, EventArgs e)
        {
            serialPort1.PortName = comboBox1.Text;
            serialPort1.BaudRate = 115200;
            serialPort1.Parity = Parity.None;
            serialPort1.StopBits = StopBits.One;
            serialPort1.DataBits = 8;
            try
            {
                serialPort1.Open();
            }
            catch(Exception ex)
            {
                MessageBox.Show($"HATA! \n Sebep:{ex.Message}","de-
neme",MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            if(serialPort1.IsOpen)
            {
                button1.Enabled = false;
                button2.Enabled = true;
                button4.Enabled = true;
                button3.Enabled = true;
                timer1.Start();
            }
        }
```

Tablo 8 “Bağlan” Butonu

comboBox içerisindeki seçilen seri porta 115200 baudrate hızında bağlanarak 8 bitlik seri port haberleşmesini başlatır.

```
private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            button1.Enabled = true;
            button2.Enabled=false;
            timer1.Stop();
        }
```

Tablo 9 “Kes” Butonu

Bağlanılan seri port ile olan haberleşmeyi sonlandırır.

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedE-
ventArgs e)
        {

            veri= serialPort1.ReadLine();
            ayrik_veri = veri.Split('*');
            volt = ayrik_veri[0];
            akim = ayrik_veri[1];
            pwm  = ayrik_veri[2];
            Thread.Sleep(30);

            string data = serialPort1.ReadLine();// bufferdan verileri oku

            if (textBox1.InvokeRequired)
            {
                textBox1.Invoke(new MethodInvoker(delegate { textBox1.Text
= volt + "\r\n"; }));

            }
            if (textBox1.InvokeRequired)
            {
                textBox2.Invoke(new MethodInvoker(delegate { textBox2.Text
= akim + "\r\n"; }));

            }
            label1.Text = String.Format("{0:0.00}", textBox1.Text.ToSt-
ring());
            label2.Text = String.Format("{0:0.00}", textBox2.Text.ToSt-
ring());
            label6.Text = pwm;
        }
```

Tablo 10 Veri Alma

Seri porttan arayüze gelen değerleri veri adındaki diziye kaydedilerek “*” ile ayrılarak gerilim, akım ve PWM olarak kaydedilir. Değerler değişiklik meydana geldiği zaman textBox1 ve textBox2 içerisinde saklanır.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            if(volt != null)
                {
                akim1 = (float)Convert.ToDecimal(akim, CultureInfo.GetCul-
tureInfo("en-US"));
                volt1 = (float)Convert.ToDecimal(volt, CultureInfo.GetCul-
tureInfo("en-US"));
                volt2 = (float)System.Math.Round(volt1, 2);
                akim2 = (float)System.Math.Round(akim1, 2);


                //int tepe = Int32.Parse(volt);
                chart1.ChartAreas[0].AxisX.Minimum = min;
                chart1.ChartAreas[0].AxisX.Maximum = max;
                chart1.ChartAreas[0].AxisY.Minimum = volt2 * .95;
                chart1.ChartAreas[0].AxisY.Maximum = volt2 * 1.05;
                chart1.ChartAreas[0].AxisX.ScaleView.Zoom(min, max);
                this.chart1.Series[0].Points.AddXY((max - 1), volt2);
                chart1.ChartAreas[0].AxisY2.Minimum = akim2 * .95;
                chart1.ChartAreas[0].AxisY2.Maximum = akim2 * 1.05;
                this.chart1.Series[1].Points.AddXY((max - 1), akim2);
                max++;
                min++;
            }
            }
```

Tablo 11 Akım ve Gerilim Grafiği

X ekseni son 15 saniyeyi gösterirken Y ekseni ise akım ve gerilim değerlerini gösterir. Y ekseni ayrıca iki eksen de çalışmaktadır. Sağ taraf gerilimi verirken sol bölge ise akım değerlerini kullanıcı ile paylaşmaktadır. Max ve min değeri okunan değerin %5 fazlası ve %5 eksiği olacak şekilde tolerans ile çalışmaktadır.

Arayüz kendini saniye de bir kez güncelleyecek şekilde tasarlanmıştır. Bu sayede sisteme fazla yük binmesi engellenmiştir.

```
private void button3_Click(object sender, EventArgs e)
        {
            int pwm_suan = Int32.Parse(pwm);
            if(pwm_suan<280){
                pwm_suan++;
            }
            else{
                pwm_suan = 280;
            }
            byte[] b = BitConverter.GetBytes(pwm_suan);
            serialPort1.Write(b, 0, 4);
        }
```

Tablo 12 “PWM+” Butonu

Seri port üzerinden alan PWM değerini arttırarak yine seri port üzerinden işlemciye gönderir.

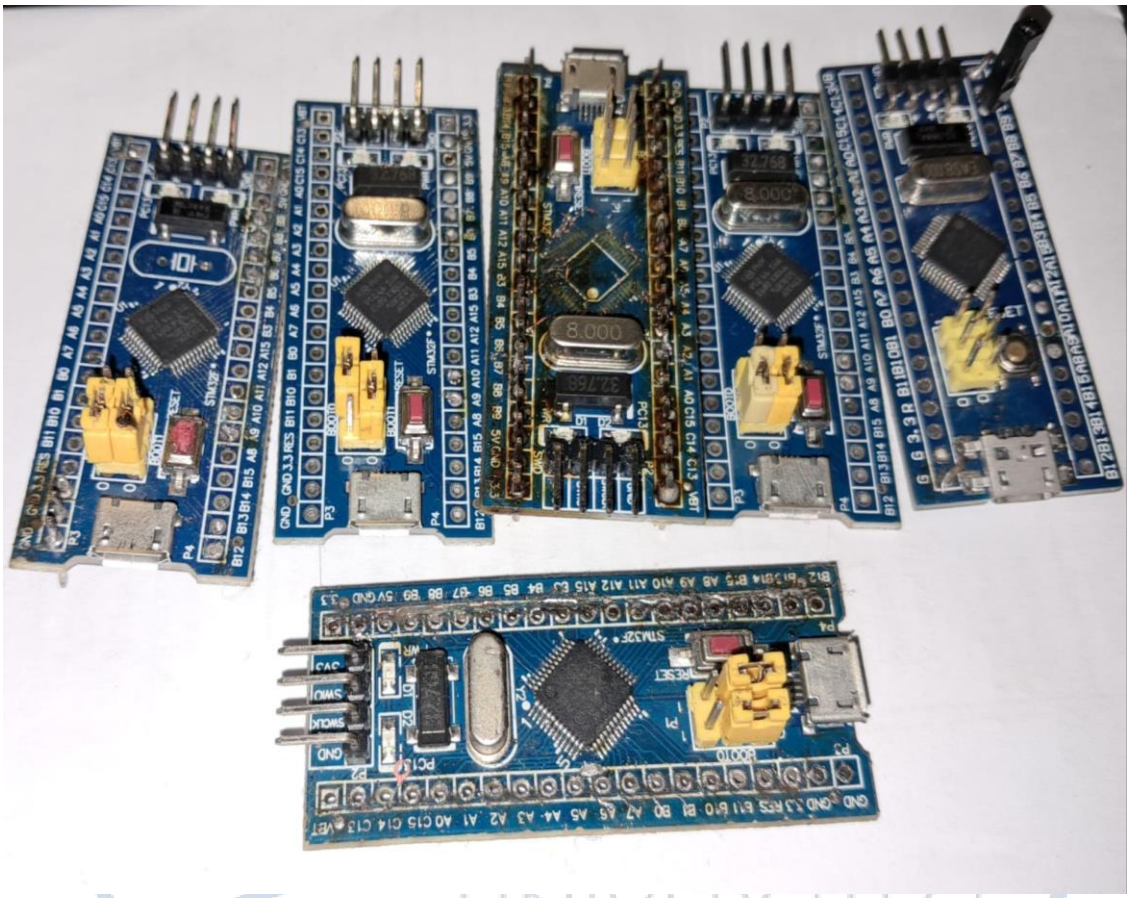
```
private void button4_Click(object sender, EventArgs e)
        {
            int pwm_suan = Int32.Parse(pwm);

            if (pwm_suan > 20)
            {
                pwm_suan--;
            }
            else
            {
                pwm_suan = 20;
            }
            byte[] b = BitConverter.GetBytes(pwm_suan);
            serialPort1.Write(b, 0, 4);
        }
```

Tablo 13 “PWM-“ Butonu

Seri port üzerinden alan PWM değerini azaltarak yine seri port üzerinden işlemciye gönderir.

5.4.Test Sonuçları

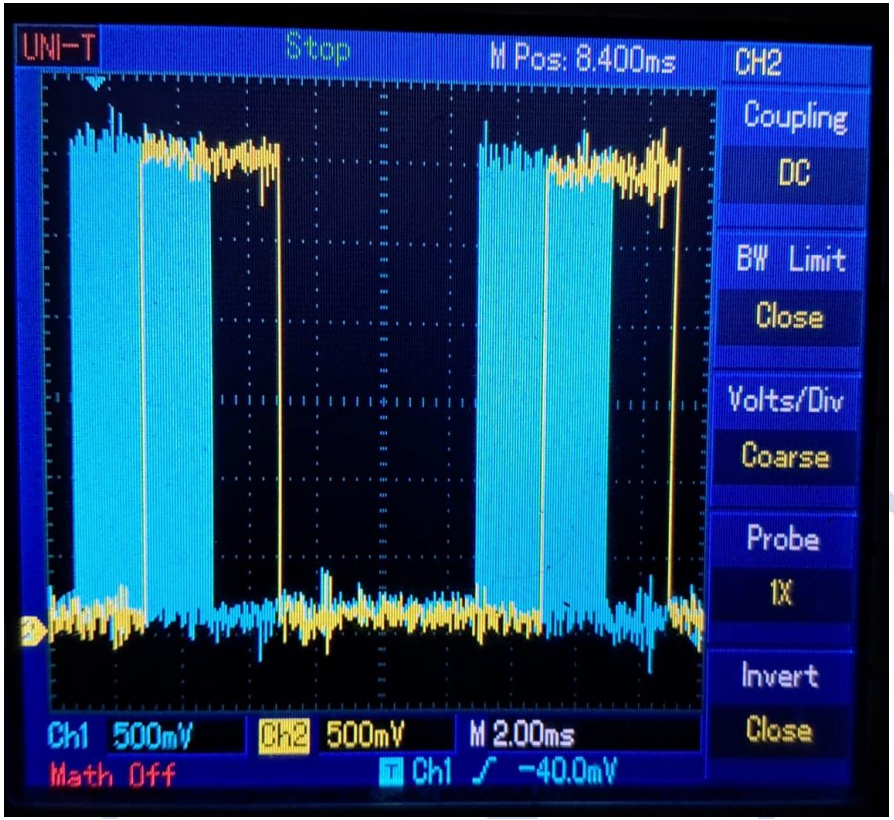
5.4.1 Prototip Aşaması



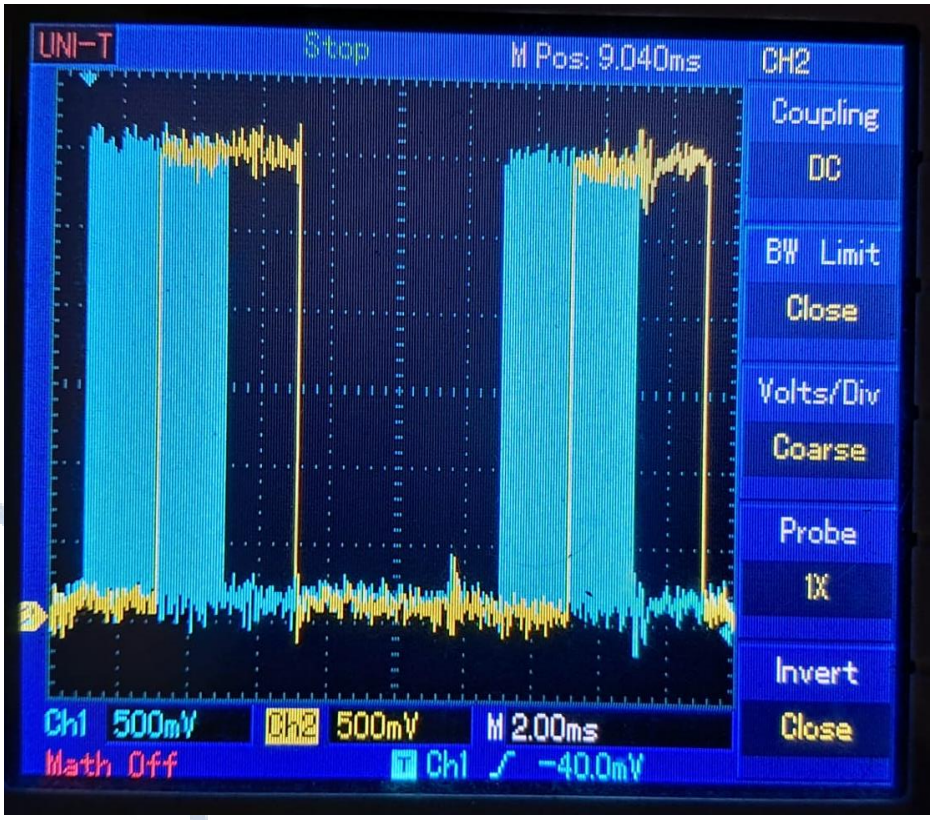
Şekil 26 Proje Boyunca Bozulan Mikrodenetleyiciler

Proje süresince birçok mikrodenetleyici kullanılmaz hale geldi bu durumun başlıca sebepleri;

- Kullanılan LT1117 3V3 voltaj regülatörünün ters gerilim korumasına sahip olmaması ve regülatör çıkışının diyot ile korunmaması sebebiyle, entegrenin bozularak mikrodenetleyiciyi besleyen gerilimin istenenin gerilimin üstünde olması nedeni ile mikrodenetleyici çalışamaz duruma gelmiştir.
- Ölçümler sırasında GPIO pinleri ile toprak hattının istenmeyen bir şekilde kısa devre olması.
- Kullanıcı kaynaklı olmayan, seri üretim ve orijinal olmaması sebebiyle oluşan kalitesiz ürünler.

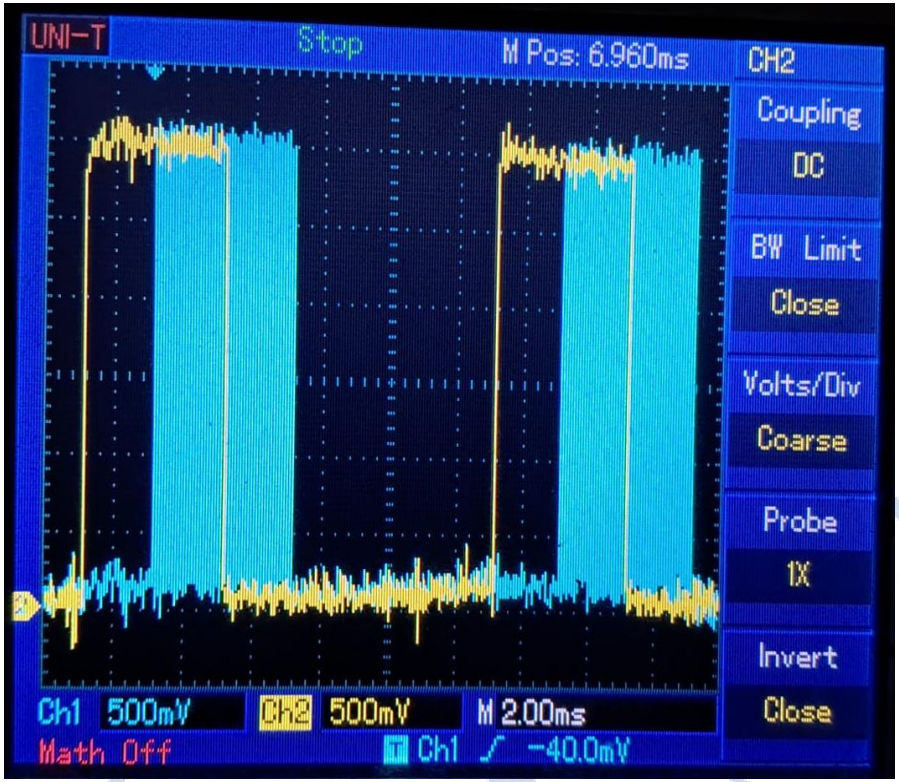


Şekil 27 H1-L2 Mikrodenetleyici Çıkışları

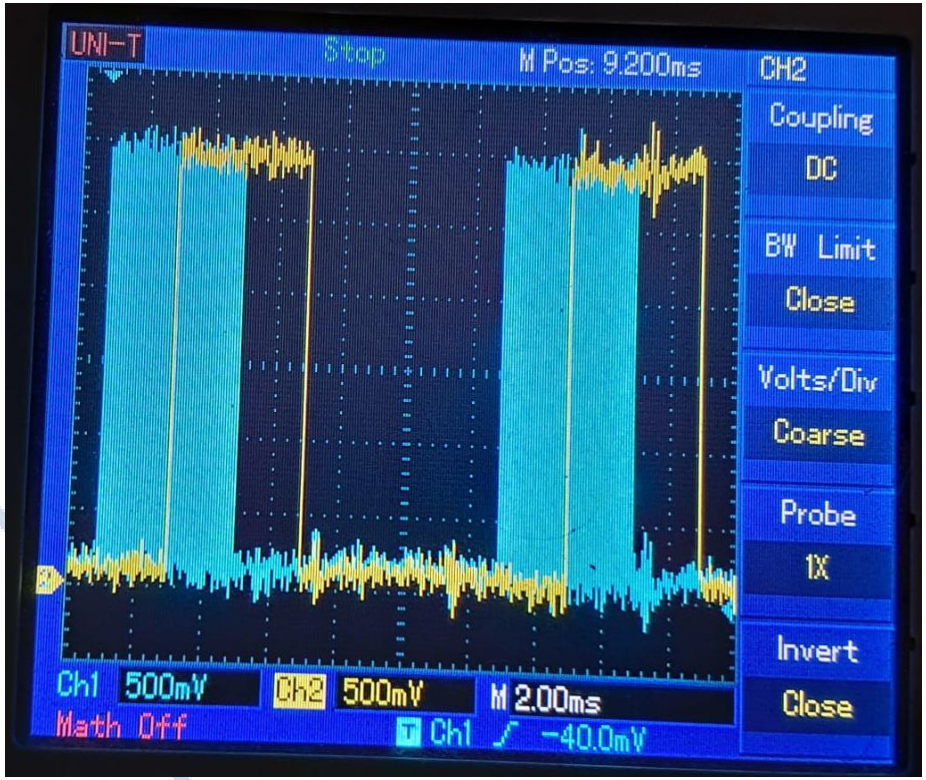


Şekil 28 H1-L3 Mikrodenetleyici Çıkışları

2007 KARABÜK ÜNİVERSİTESİ

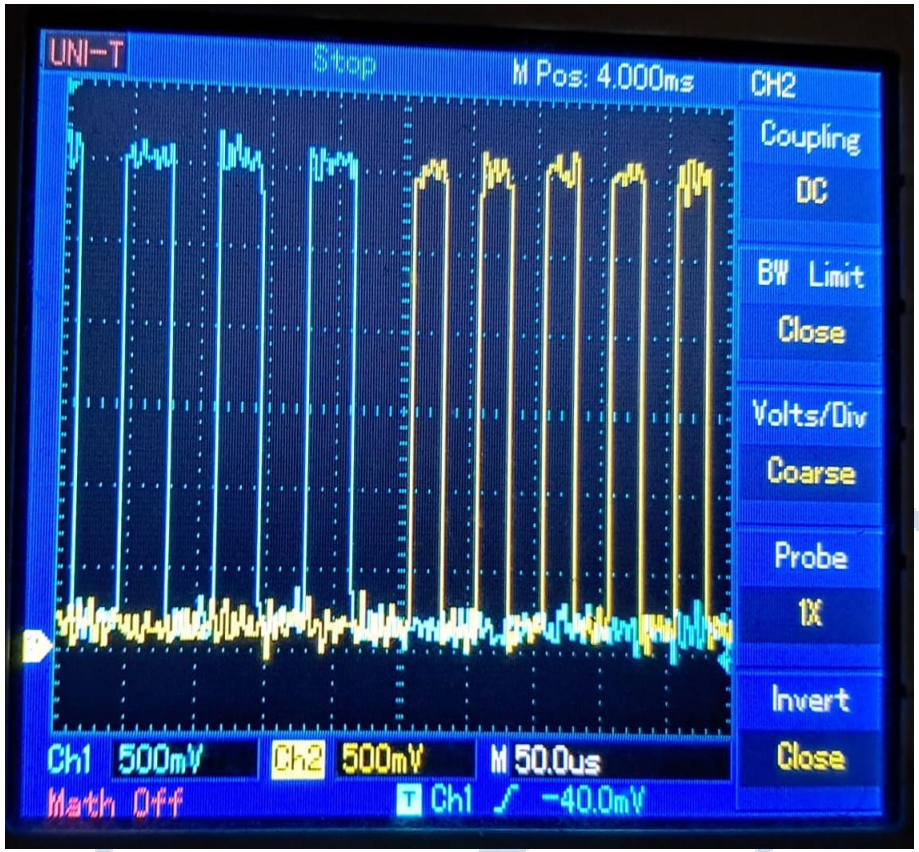


Şekil 29 H2-L3 Mikrodenetleyici Çıkışları

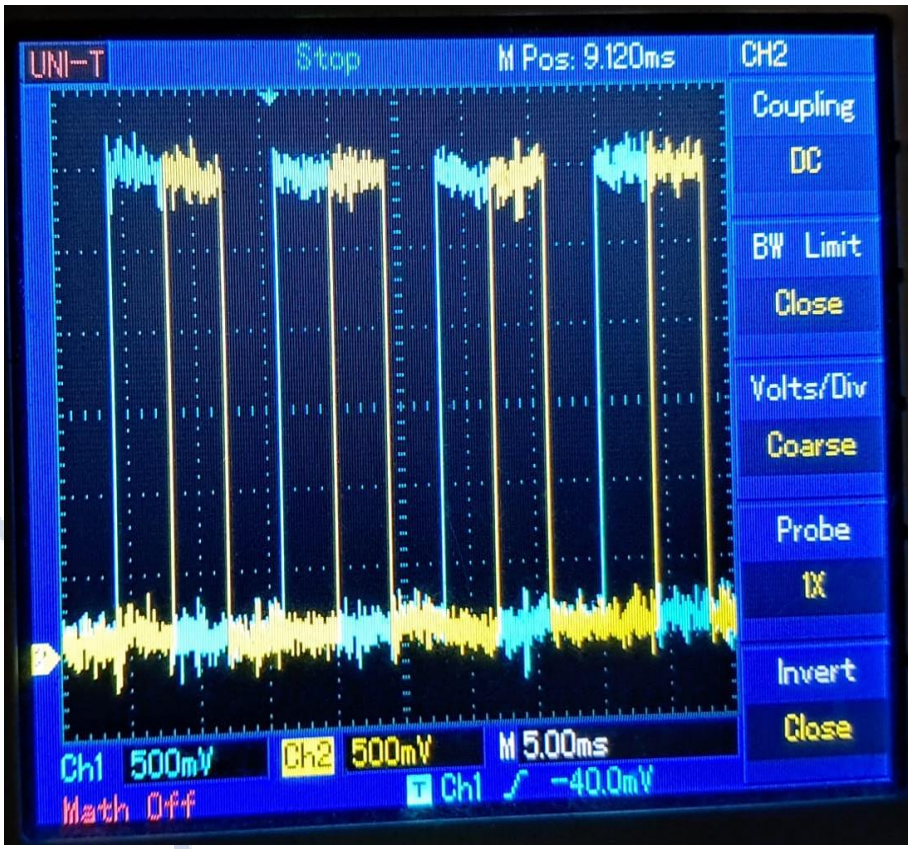


Şekil 30 H2-L1 Mikrodenetleyici Çıkışları

2007 KARABÜK ÜNİVERSİTESİ



Şekil 31 H1'in kapanıp H2'nin açılma anı

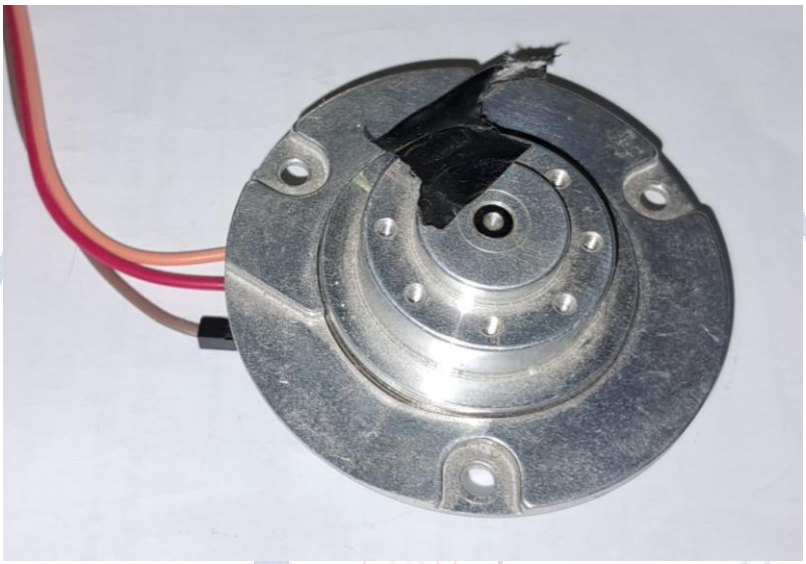


Şekil 32 L2'nin kapanıp L3'ün açılma anı

2007
KARABÜK
ÜNİVERSİTESİ

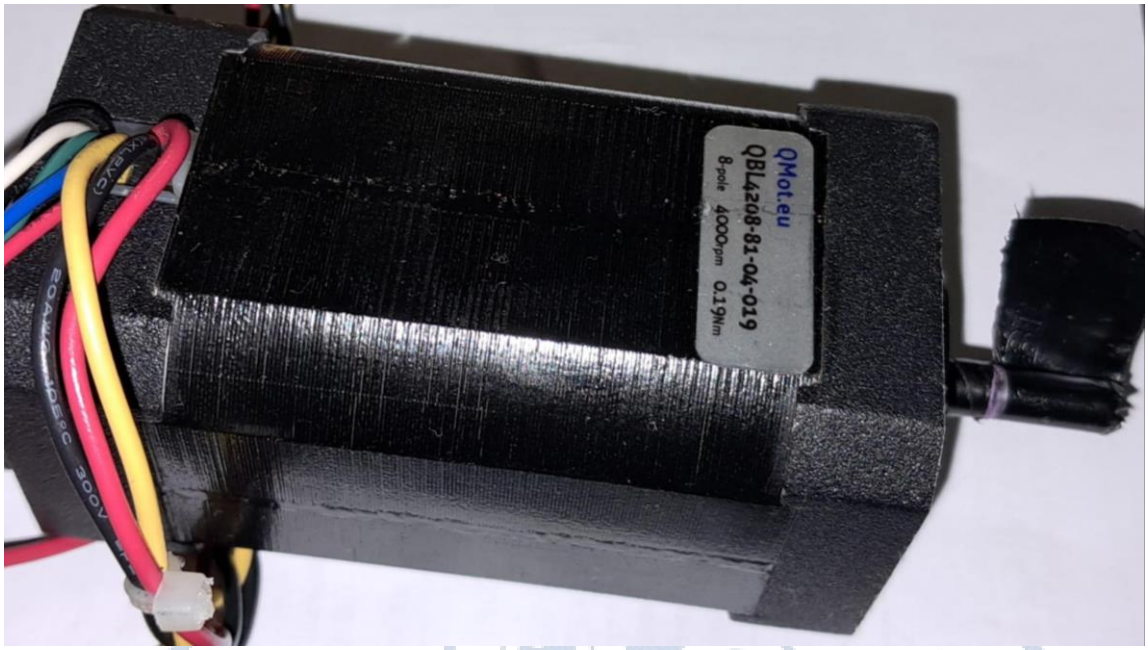
Şekil 27, 28, 29, 30 da mikrodenetleyicinin mosfet sürücüleri kontrol eden time pinlerinin 6 adımlı komütasyona göre açılıp kapanması gösterilmiştir. H3-L1 ve H3-L2 adımları önceki adımlar ile aynı olduğu için yer verilmemiştir.

Şekil 31 ve 32 de oluşan grafikler tablo 3 de anlatılan 6 adımlı komütasyonun H1 kapanıp H2 açılma ve L2 kapanıp L3 açılma anındaki osiloskop çıktısını göstermektedir.



Şekil 33 Hall Sensör Bulunmayan HDD Motoru

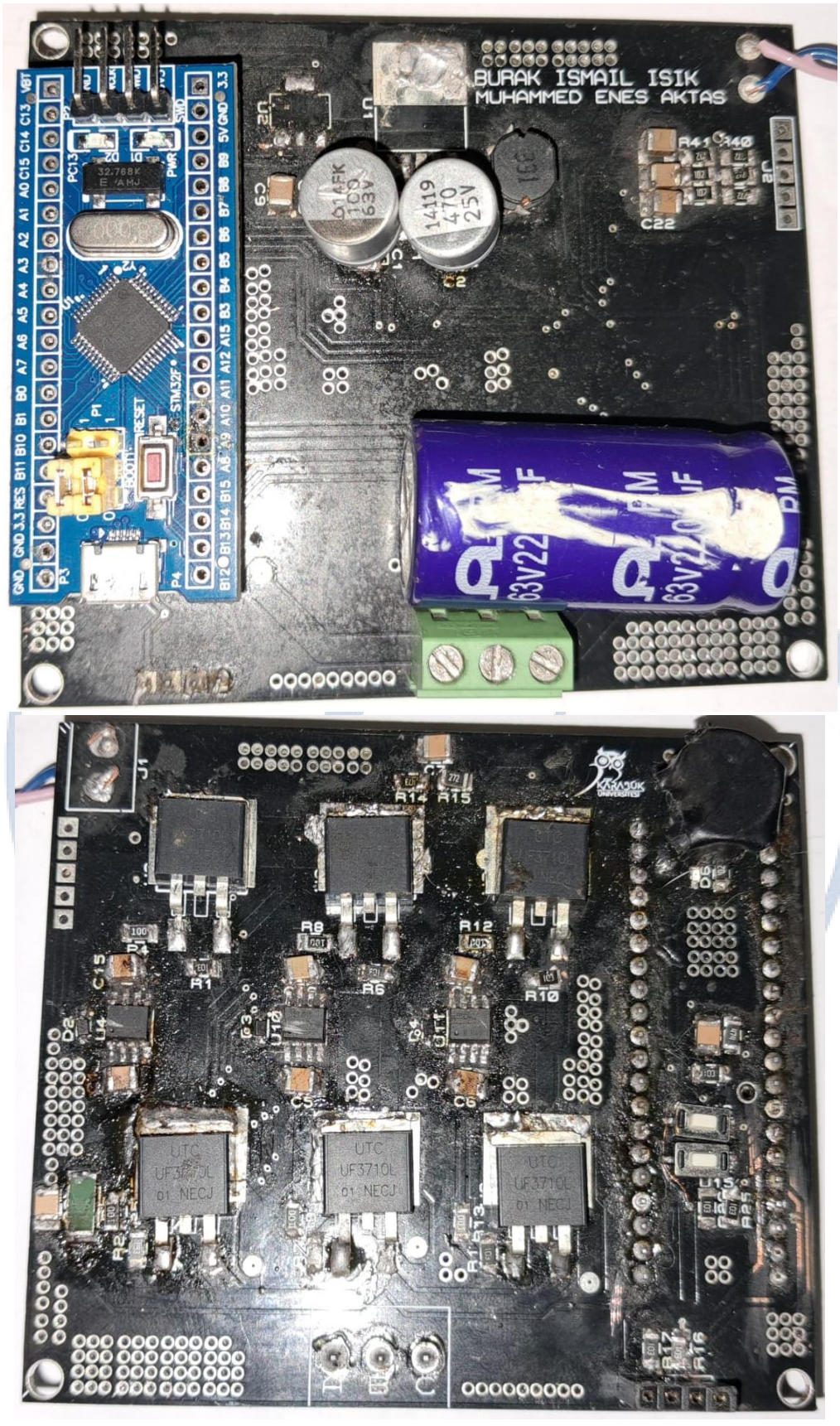
Şekil 34 Hall Sensöründe Hata Bulunan Motor



Şekil 35 Sürücü Tasarımında Refereans Alınan Motor (QBL 4208)

Specifications		QBL 4208			
		-41-04-006	-61-04-013	-81-04-019	-100-04-025
No. of Pole		8	8	8	8
No. of Phase		3	3	3	3
Rated Voltage	V	24	24	24	24
Rated Phase Current	A	1.79	3.47	5.14	6.95
Rated Speed	RPM	4000	4000	4000	4000
Rated Torque	Nm	0.0625	0.125	0.185	0.25
Max Peak Torque	Nm	0.19	0.38	0.56	0.75
Torque Constant	Nm/A	0.035	0.036	0.036	0.036
Line to Line Resistance	Ohm	1.8	0.72	0.55	0.28
Line to Line Inductance	mH	2.6	1.2	0.8	0.54
Max Peak Current	A	5.4	10.6	15.5	20
Length (L_{MAX})	mm	41	61	81	100
Rotor Inertia	$kgm^2 \times 10^{-6}$	24	48	72	96
Mass	kg	0.3	0.45	0.65	0.8

Şekil 36 QBL 4208 Referans Motor Özellikleri [12]



Şekil 37 Prototip Aşamasındaki Devre Kartı

KODLAR

Tablo 14 STM32CUBEIDE Kodları

```c
#include "main.h"
#include "stdio.h"
#include <stdbool.h>

ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc2;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim4;

UART_HandleTypeDef huart3;


uint8_t Buffer[25] = {0};
uint8_t Space[] = " - ";
uint8_t StartMSG[] = "Starting I2C Scanning: \r\n";
uint8_t EndMSG[] = "Done! \r\n\r\n";
int pwm_value=30;
bool HALL1 = false;
bool HALL2 = false;
bool HALL3 = false;
uint32_t step=1;
int pwm_eski;
bool saat_yon = true;
int rpm;
bool rpm_ok=false;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM4_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_ADC2_Init(void);


void uart_gonder(char *deger)
{
        HAL_UART_Transmit(&huart3, (uint8_t*)deger, strlen(deger), 100);
}

void basla (void) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);       //L1
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//L2
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//L3
        __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);   //H1
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);   //H2
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);   //H3

        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_Delay(10);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_Delay(950);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_Delay(10);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_Delay(950);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        HAL_Delay(10);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_Delay(950);
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);   //H3
        __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,100); //H1
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//L2
        HAL_Delay(10);
```

```
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//L2
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);//L3
        HAL_Delay(10);
        __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);    //H1
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,100);  //H2
        HAL_Delay(10);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//L3
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);        //L1
        HAL_Delay(10);
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);    //H2
        __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,100);  //H3
        HAL_Delay(10);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);        //L1
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//L2
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){

   HALL1=HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_13);
   HALL2=HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14);
   HALL3=HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15);
if(saat_yon==true)
{
   if(HALL1&&!HALL2&&HALL3){//101
         step=4;
      }
   if(HALL1&&!HALL2&&!HALL3){//100
         step=5;
      }
   if(HALL1&&HALL2&&!HALL3){//110
         step=6;
      }
   if(!HALL1&&HALL2&&!HALL3){//010
         step=1;
      }
   if(!HALL1&&HALL2&&HALL3){//011
         step=2;
      }
   if(!HALL1&&!HALL2&&HALL3){//001
         step=3;
      }
}
else if(saat_yon == false)
{
        if(HALL1&&!HALL2&&HALL3){//101
                        step=1;
                }
        if(HALL1&&!HALL2&&!HALL3){//100
                        step=2;
                }
        if(HALL1&&HALL2&&!HALL3){//110
                        step=3;
                }
        if(!HALL1&&HALL2&&!HALL3){//010
                        step=4;
                }
        if(!HALL1&&HALL2&&HALL3){//011
                        step=5;
                }
        if(!HALL1&&!HALL2&&HALL3){//001
                        step=6;
                }
}
   switch(step)
   {
      case 1:
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
           __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);//BH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//BL=H
          __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,pwm_value);//AH=H PWM
```

```
      break;
      case 2:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);//BH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);//CL=H
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,pwm_value);//AH=H PWM
      break;
      case 3:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL=L
           __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
           __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1);//CL=H
           __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,pwm_value);//BH=H PWM
      break;
      case 4:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
          __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,0);//CH=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);//AL=H
          __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,pwm_value);//BH=H PWM
      break;
      case 5:
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);//BL=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL=L
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=H
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);//BH=L
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 1);//AL=H
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,pwm_value);//CH=H PWM
      break;
      case 6:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);//AL
           HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0);//CL
         __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,0);//AH=L
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_2,0);//BH=L
         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);//BL=H
         __HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_4,pwm_value);//CH=H PWM
         rpm++;
                  if(rpm==100)
                  {
                         rpm_ok=true;
                         rpm=0;
                  }
      break;
   }
}


int main(void)
{

           int sira=1;
           int i;
           uint8_t acilis[] = "Sistem Basliyor\r\n";
           uint8_t hazir[] = "Sistem Hazir\r\n";
        uint8_t Rx_data=0;
        uint8_t MSG2[] = "Sistem Hazir\r\n";
        uint16_t raw;
        uint16_t raw1;
        float volt;
        float akim;
        char buffer[32]={0};

  HAL_Init();


  SystemClock_Config();
```

```
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_ADC1_Init();
  MX_TIM4_Init();
  MX_USART3_UART_Init();
  MX_ADC2_Init();

  printf("\n");
  HAL_UART_Transmit(&huart3, MSG2, sizeof(MSG2), 100);
  HAL_Delay(1000);
  HAL_UART_Transmit(&huart3, MSG2, sizeof(MSG2), 100);


  HAL_UART_Transmit(&huart3, acilis, sizeof(acilis), 100);
  HAL_Delay(20);
  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);//A PWM START
  HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);// B PWM START
  HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_4);// C PWM START
  HAL_Delay(50);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
  HAL_Delay(500);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
  HAL_Delay(500);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
  HAL_Delay(10);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
  HAL_Delay(10);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
  HAL_Delay(10);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
  HAL_Delay(100);
  HAL_UART_Transmit(&huart3, hazir, sizeof(hazir), 100);

  basla();

  while (1)
  {



          if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0)==1)
          {
                  if(pwm_value==0){
                        pwm_value=pwm_value+10;
                        basla;
                  }
                  else{
                          pwm_value=pwm_value+10;
                  }
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
                HAL_Delay(10);
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
                  if(pwm_value>290)
                  {
                        pwm_value=290;
                  }
                  HAL_Delay(100);
          }
          if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_1)==1)
                  {
                  pwm_value=pwm_value-10;
                        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
                        HAL_Delay(10);
                        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
                          if(pwm_value<0)
                          {
                                pwm_value=0;
                          }
                          HAL_Delay(100);
                  }

                HAL_UART_Receive (&huart3, &Rx_data, 2, 100);
```

```
		if(Rx_data=='1')
		{
			if(pwm_value==0){
				pwm_value=pwm_value+10;
				basla;
			}
			else{
				pwm_value=pwm_value+10;
			}
			if(pwm_value>290)
			{
				pwm_value=290;
			}
			HAL_Delay(100);
		}
		if(Rx_data=='3')
		{
		pwm_value=pwm_value-10;
			if(pwm_value<0)
			{
				pwm_value=3;
			}
			HAL_Delay(100);
		}
		if(Rx_data=='0')
		{
				pwm_eski=pwm_value;
				pwm_value=0;
				HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
				  HAL_Delay(100);

			}
			if(Rx_data=='2'){
				HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
				pwm_value=pwm_eski;
				basla;
				  HAL_Delay(100);

			}
			if(Rx_data=='4'){
				basla;
				saat_yon=false;
				  HAL_Delay(100);
			}
			if(Rx_data=='5'){
				basla;
				saat_yon=true;
				  HAL_Delay(100);
			}
			if(Rx_data=='6'){
				pwm_value=pwm_eski;
				pwm_value==0;
			  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
			  HAL_Delay(500);
			  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
			  HAL_Delay(500);
			}


	HAL_ADC_Start(&hadc1);
	HAL_ADC_PollForConversion(&hadc1, 100);
	raw = HAL_ADC_GetValue(&hadc1);

	HAL_ADC_Stop (&hadc1);
	HAL_Delay(2);
	HAL_ADC_Start(&hadc2);
	HAL_ADC_PollForConversion(&hadc2, 100);
	raw1 = HAL_ADC_GetValue(&hadc2);

	HAL_ADC_Stop (&hadc2);
	if(rpm_ok==true)
	{
```

```
                    uint32_t len=sprintf(buffer, "%d*%d*%d*1\r\n", raw, raw1, pwm_value);
                    rpm_ok=false;
                    HAL_UART_Transmit(&huart3, (uint16_t*)buffer, len, 100);
            }
            else{
                    uint32_t len=sprintf(buffer, "%d*%d*%d*0\r\n", raw, raw1, pwm_value);
                    HAL_UART_Transmit(&huart3, (uint16_t*)buffer, len, 100);
            }

    }

}


void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};


  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }


  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV4;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}


static void MX_ADC1_Init(void)
{


  ADC_ChannelConfTypeDef sConfig = {0};


  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  sConfig.Channel = ADC_CHANNEL_5;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_71CYCLES_5;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

}


static void MX_ADC2_Init(void)
{


  ADC_ChannelConfTypeDef sConfig = {0};



  hadc2.Instance = ADC2;
  hadc2.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc2.Init.ContinuousConvMode = ENABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
    Error_Handler();
  }


  sConfig.Channel = ADC_CHANNEL_4;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_71CYCLES_5;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

}


static void MX_TIM1_Init(void)
{


  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};


  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 11;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 299;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) != HAL_OK)
  {
    Error_Handler();
  }

  HAL_TIM_MspPostInit(&htim1);

}


static void MX_TIM4_Init(void)
{



  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 11;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 299;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
```

```c
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */
  HAL_TIM_MspPostInit(&htim4);

}

/**
  * @brief USART3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    Error_Handler();

  }
  /* USER CODE BEGIN USART3_Init 2 */

  /* USER CODE END USART3_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, Buzzer_Pin|L1_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, L2_Pin|L3_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Buzzer_Pin */
  GPIO_InitStruct.Pin = Buzzer_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
```

```
  HAL_GPIO_Init(Buzzer_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PA6 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PWM__Pin PWM_B1_Pin */
  GPIO_InitStruct.Pin = PWM__Pin|PWM_B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : PB13 PB14 PB15 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : L1_Pin */
  GPIO_InitStruct.Pin = L1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(L1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : L2_Pin L3_Pin */
  GPIO_InitStruct.Pin = L2_Pin|L3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  __disable_irq();
  while (1)
  {
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
                HAL_Delay(10);
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
```

```
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Tablo 15 Arayüz Tasarımı Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Globalization;
using System.IO.Ports;
using System.Linq;
using System.Media;
using System.Reflection.Emit;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;


namespace ESC1
{

    public partial class Form1 : Form
    {
        string raw_volt;
        string raw_akim;
        ulong max = 15;
        ulong min = 0;
        float volt;
        double akim;
        double akim_max;
        double akim_max2;
        double akim_max3;
        double akim_max4;
        double akim_max5;
```

```
        float akim1;
        float volt1;
        float akim2;
        float volt2;
        string veri;
        string[] ayrik_veri;
        string pwm;
        string rpm;
        double rpm1;
        int rpm2;
        bool dur = false;
        bool saat_yon = true;
        float volt_kes=10;
        float akim_kes=500;
        float volt_kes_text;
        float akim_kes_text;
        bool rpm_basla = false;
        Stopwatch stopwatch = new Stopwatch();
        public Form1()
        {
            InitializeComponent();

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            stopwatch.Start();
            Control.CheckForIllegalCrossThreadCalls = false;
            foreach (var seri in SerialPort.GetPortNames())
            {
                comboBox1.Items.Add(seri);
            }
            if (comboBox1.SelectedIndex == -1)
            {
                comboBox1.Text = "Seri Port Yok";
            }
            else
            {
                comboBox1.SelectedIndex = 0;
            }
            button2.Enabled = false;
            button4.Enabled = false;
            button3.Enabled = false;
            button6.Enabled = false;
            button7.Enabled = false;
            label1.Text = "";
            label2.Text = "";
```

```
            button6.Text = "CCW dön";
            textBox3.Text = "10";
            textBox4.Text = "500";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            serialPort1.PortName = comboBox1.Text;
            serialPort1.BaudRate = 115200;
            serialPort1.Parity = Parity.None;
            serialPort1.StopBits = StopBits.One;
            serialPort1.DataBits = 8;
            try
            {
                serialPort1.Open();
            }
            catch(Exception ex)
            {
                MessageBox.Show($"HATA! \n Sebep:{ex.Message}","(,)))",MessageBoxButtons.OK, Mes-
sageBoxIcon.Error);
            }
            if(serialPort1.IsOpen)
            {
                button1.Enabled = false;
                button2.Enabled = true;
                button4.Enabled = true;
                button3.Enabled = true;
                button5.Enabled = true;
                button6.Enabled = true;
                button7.Enabled = true;
                timer1.Start();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            button1.Enabled = true;
            button2.Enabled=false;
            button4.Enabled = false;
            button3.Enabled = false;
            button5.Enabled = false;
            button6.Enabled = false;
            timer1.Stop();
        }
```

```
        private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {

            veri= serialPort1.ReadLine();
            ayrik_veri = veri.Split('*');
            raw_volt = ayrik_veri[0];
            raw_akim = ayrik_veri[1];
            pwm = ayrik_veri[2];
            rpm = ayrik_veri[3];

            rpm1 = (float)Convert.ToDecimal(rpm, CultureInfo.GetCultureInfo("en-US"));
            if (rpm1 == 1 && rpm_basla == false)
            {
                stopwatch.Start();
                rpm_basla = true;
            }
            else if (rpm1 == 1 && rpm_basla == true)
            {
                stopwatch.Stop();
                label9.Text = stopwatch.Elapsed.ToString();
                stopwatch.Restart();
                rpm_basla = false;
            }

                //label9.Text = stopwatch.Elapsed.ToString();

            Thread.Sleep(100);

            string data = serialPort1.ReadLine();       // bufferdan verileri oku

            if (textBox1.InvokeRequired)
            {
                textBox1.Invoke(new MethodInvoker(delegate { textBox1.Text = volt2 + "\r\n"; }));

            }
            if (textBox1.InvokeRequired)
            {
                textBox2.Invoke(new MethodInvoker(delegate { textBox2.Text = akim2*1000 + "\r\n";
}));

            }

            label6.Text = pwm;
            label1.Text = textBox1.Text.ToString();
            label2.Text = textBox2.Text.ToString();
            akim = (float)Convert.ToDecimal(raw_akim, CultureInfo.GetCultureInfo("en-US"));
            volt = (float)Convert.ToDecimal(raw_volt, CultureInfo.GetCultureInfo("en-US"));
```

```
            if (volt < 0.00000001)
            {
                volt = 0;
            }
            if (akim < 0.0000000000000000000000000001)
            {
                akim = 1;
            }
            volt1 = (float)Convert.ToDecimal(((volt * 3.3) / 4095) * 48 / 2.41);
            volt2 = (float)System.Math.Round(volt1, 3, MidpointRounding.ToEven);
            akim1 = (float)Convert.ToDecimal(((akim * 3.3) / 2048));//bu neden 10 bit?
            akim2 = (float)System.Math.Round(akim1, 4, MidpointRounding.ToEven);
            aGauge1.Value = Convert.ToInt16(volt2);
            aGauge2.Value = Convert.ToInt16(akim2 * 1000);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if(raw_volt != null)
                {

                //int tepe = Int32.Parse(volt);
                chart1.ChartAreas[0].AxisX.Minimum = min;
                chart1.ChartAreas[0].AxisX.Maximum = max;
                chart1.ChartAreas[0].AxisY.Minimum = volt2 * .5;
                chart1.ChartAreas[0].AxisY.Maximum = volt2 * 1.5;
                chart1.ChartAreas[0].AxisX.ScaleView.Zoom(min, max);
                this.chart1.Series[0].Points.AddXY((max - 1), volt2);
                chart1.ChartAreas[0].AxisY2.Minimum = akim2 * .5;
                chart1.ChartAreas[0].AxisY2.Maximum = akim2 * 1.5;
                this.chart1.Series[1].Points.AddXY((max - 1), akim2);
                max++;
                min++;
                if (volt2 < volt_kes)
                {
                    serialPort1.Write("6\r\n");
                    button5.Enabled = false;
                    button5.Text = "Düşük Gerilim";
                    SystemSounds.Beep.Play();
                }
                else if (akim2*1000 > akim_kes)
                {
                    serialPort1.Write("6\r\n");
                    button5.Enabled = false;
                    button5.Text = "Yüksek Akım";
                    SystemSounds.Beep.Play();
                }
```

```
            else
            {
                button5.Enabled = true;
                button5.Text="DUR!!!";
            }

        }
        }

    private void chart1_Click(object sender, EventArgs e)
    {

    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void label3_Click(object sender, EventArgs e)
    {

    }

    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void button3_Click(object sender, EventArgs e)
    {

        int pwm_suan = Int32.Parse(label6.Text);
        if(pwm_suan<290)
        {
            pwm_suan++;
            serialPort1.Write("1\r\n");
        }
        else
        {
            pwm_suan = 290;
        }
        //byte[] b = BitConverter.GetBytes(pwm_suan);
        //serialPort1.Write(b, 0, 4);

        //serialPort1.Write("1\r\n");
        Thread.Sleep(30);
```

```
        }

        private void button4_Click(object sender, EventArgs e)
        {

            int pwm_suan = Int32.Parse(label6.Text);
            if (pwm_suan > 0)
            {
                pwm_suan--;
                serialPort1.Write("3\r\n");
                //Thread.Sleep(30);
            }
            else
            {
                pwm_suan = 0;
            }
            Thread.Sleep(30);

            //serialPort1.Write("0\r\n");
            //byte[] b = BitConverter.GetBytes(pwm_suan);
            //serialPort1.Write(b, 0, 4);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (dur == false)
            {
                serialPort1.Write("0\r\n");
                dur = true;
                Thread.Sleep(30);
            }
            else if(dur==true)
            {
                serialPort1.Write("2\r\n");
                dur = false;
                Thread.Sleep(30);
            }
        }

        private void aGauge1_ValueInRangeChanged(object sender, AGaugeApp.AGauge.ValueInRangeC-
hangedEventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {
```

```
        }

        private void button6_Click(object sender, EventArgs e)
        {
            if (saat_yon == true)
            {
                serialPort1.Write("4\r\n");
                saat_yon = false;
                Thread.Sleep(30);
                button6.Text = "CW dön";
            }
            else if (saat_yon == false)
            {
                serialPort1.Write("5\r\n");
                saat_yon = true;
                Thread.Sleep(30);
                button6.Text = "CCW dön";
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {

            if (textBox4.Text == "")
            {
                MessageBox.Show($"Akım değeri girmelisin! \n", "Hata", MessageBoxButtons.OK, Mes-
sageBoxIcon.Error);
            }
            if (textBox3.Text == "")
            {
                MessageBox.Show($"Gerilim değeri girmelisin! \n", "Hata", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            akim_kes_text = (float)Convert.ToDecimal(textBox4.Text);
            volt_kes_text = (float)Convert.ToDecimal(textBox3.Text);
            if (volt_kes_text != volt_kes)
            {
                volt_kes = volt_kes_text;
            }
            if (akim_kes_text != akim_kes)
            {
                akim_kes = akim_kes_text;
            }
        }

        private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
```

```
        {
            e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar) && e.KeyChar != 
',';
        }

        private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar) && e.KeyChar != 
',';
        }

        private void timer2_Tick(object sender, EventArgs e)
        {

        }
    }
}
```

2007 KARABÜK ÜNİVERSİTESİ

KAYNAKÇA

[1] Chapman, %1 içinde *Electric Machinery Fundamentals*, 2005, pp. 606-609.

[2] K. Huang, 28 Kasım 2019. [Çevrimiçi]. Available: https://www.linkedin.com/pulse/3-benefits-brushless-dc-motors-industrial-engineering-kris-huang.

[3] «Three-Phase Brushless DC (BLDC) Power Tool Motor Driver».

[4] U. Dereli, «Fırçasız Doğru Akım Motoru ve Sürücüsü Tasarımı,» Yozgat, 2020.

[5] Z. Peterson, «What Are Differential Pairs and Differential Signals?,» 2021. [Çevrimiçi]. Available: https://resources.altium.com/p/what-are-differential-pairs-and-differential-signals.

[6] «Bootstrap Circuitry Selection for Half-Bridge,» 2023.

[7] H. İ. Seçen, «Doğru Akım Elektrik Makinelerinde Komütasyon ve Endüvi Reaksiyonu Nedir?,» [Çevrimiçi]. Available: https://www.elektrikport.com/universite/dogru-akim-elektrik-makinelerinde-komutasyon-ve-enduvi-reaksiyonu-nedir/9024.

[8] [Çevrimiçi]. Available: https://aspiringyouths.com/advantages-disadvantages/bldc-motors/.

[9] «What is an Electric Motor,» 2022. [Çevrimiçi]. Available: https://www.cencepower.com/blog-posts/ac-electric-motors-vs-dc-brushless-motors.

[10] [Çevrimiçi]. Available: https://www.pcbway.com.

[11] [Çevrimiçi]. Available: https://stm32-base.org/boards/STM32F103C8T6-Blue-Pill.html.

[12] F. Electronic, «QMot QBL4208,» [Çevrimiçi]. Available: https://www.farnell.com/datasheets/2623630.pdf.

2007 KARABÜK ÜNİVERSİTESİ