KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

UZAKTAN ERİŞİMLİ SABİT MIKNATISLI DOĞRU AKIM MOTOR KONTROLÜ DENEY DÜZENEĞİ

Elektronik Müh. Ahmet AKTOĞAN

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"ELEKTRONİK YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 07. 06. 2011
Tezin Savunma Tarihi : 23. 06. 2011

Tez Danışmanı : Yrd. Doç. Dr. Halil İbrahim OKUMUŞ

Trabzon 2011

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü

Elektrik Elektronik Anabilim Dalında

Ahmet AKTOĞAN tarafından hazırlanan

UZAKTAN ERİŞİMLİ SABİT MIKNATISLI DOĞRU AKIM MOTOR KONTROLÜ DENEY DÜZENEĞİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 07 / 06 / 2011 gün ve 1408 sayılı kararıyla oluşturulan jüri tarafından 23 / 06 / 2011 tarihinde yapılan sınavda

YÜKSEK LİSANS TEZİ

olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Adem Sefa AKPINAR ……

Üye : Yrd. Doç. Dr. H. İbrahim OKUMUŞ ……

Üye : Yrd. Doç. Dr. Hüseyin PEHLİVAN) ……

Prof. Dr. Sadettin KORKMAZ

Enstitü Müdürü

ÖNSÖZ

Günümüzde; uzaktan eğitim dikkate değer bir şekilde yaygınlaşmış ve önemi artmıştır. Bu sebeple; laboratuarlarda yapılan birçok deneyin internet üzerinden de yapılabilmesi gerekli hale gelmiştir. Yüksek Lisans Tezi olarak hazırlanmış olan bu çalışmada bir doğru akım motorunun akım, gerilim, hız… gibi karakteristiklerinin izlenmesi için uzaktan erişimli bir deney düzeneği (UEDD) geliştirilmiştir. Çalışmalar; donanım ve yazılım olarak iki ana kısımda yapılmıştır. Donanım olarak mikro denetleyici ünite ve motor sürücü ünitesi tasarlanmıştır. Mikro denetleyici olarak Atmel firmasının ATMega328 mikro denetleyicisi kullanılmıştır. Yazılım geliştirme çalışmaları üç ana kısımda yapılmıştır. Mikro denetleyici yazılımı C dili, internet sunucu ve istemci yazılımları ise Java dili ile geliştirilmiştir. İnternet sunucu ve istemci yazılımlarında kanal (thread) teknolojisi kullanılarak haberleşme performansı artırılmıştır. Çalışmanın tüm kısımları fiziksel olarak gerçekleştirilmiş ve sonuçlar elde edilmiştir.

Bu çalışmanın uzaktan erişim ve kontrol altyapısı K:T:Ü. Bilimsel Araştırma Projeleri Fonu'nun desteklemiş olduğu 2005.116.004.2 kodlu GSM-GPRS Sistemini Kullanarak Askeri, Meteorolojik, Turizm ve Güvenlik Amaçlı Taşınabilir Uzaktan İzleme ve Kontrol Sistemi isimli proje kapsamında yapılan çalışmalara dayanmaktadır. Verdiği destekten dolayı K.T.Ü. Bilimsel Araştırma Projeleri Fonu'na teşekkür ederim.

Lisansüstü çalışmalarım boyunca desteklerini gördüğüm Prof. Dr. İsmail Hakkı ALTAŞ, Prof. Dr. Temel KAYIKÇIOĞLU, Prof. Dr. İsmail Hakkı ÇAVDAR, Doç. Dr. Hasan KARAL, Doç. Dr. Ali GANGAL, Yrd. Doç. Dr. İsmail KAYA, tez danışmanım Yrd. Doç. Dr. Halil İbrahim OKUMUŞ, Dr. Gökce HACIOĞLU, Öğr. Gör. Mehmet KALKIŞIM, Öğr. Gör Saffet KAHVECİ, Arş. Gör. Emre ÖZKOP ve hayatın her durumunda bana destek olan aileme ve arkadaşım Sefa ÇİÇEK'e teşekkür ederim.

Bu çalışmayı aileme, özellikle çalışmalarım boyunca ihmal ettiğim ve bana sabırla katlanan kızım Ayşe Zeynep ile oğlum Ömer Faruk'a ve bu alanda çalışmalar yapmış ve yeni çalışmalar yapacak olan herkese ithaf ediyorum.

Ahmet AKTOĞAN
Trabzon 2011

TEZ BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum *"UZAKTAN ERİŞİMLİ SABİT MIKNATISLI DOĞRU AKIM MOTOR KONTROLÜ DENEY DÜZENEĞİ"* başlıklı bu çalışmayı baştan sona kadar danışmanım Yrd. Doç. Dr. Halil İbrahim OKUMUŞ'un sorumluluğunda tamamladığımı, örnekleri kendim topladığımı, deneyleri ilgili laboratuarlarda yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

05/08/2011

Ahmet AKTOĞAN

İÇİNDEKİLER

Sayfa No

ÖNSÖZ.....III
TEZ BEYANNAMESİ.....IV
İÇİNDEKİLER..... V
ÖZET.....VII
SUMMARY VIII
ŞEKİLLER DİZİNİ.....IX
TABLOLAR DİZİNİ X
SEMBOLLER DİZİNİ XI
1. GENEL BİLGİLER..... 1
1.1. Giriş 1
1.2. Doğru Akım Motorları..... 4
1.2.1. Doğru Akım Motorlarının Yapıları 5
1.2.1.1. Endüktör (Kutup)..... 6
1.2.1.2. Endüvi (Rotor)..... 6
1.2.1.3. Kolektör (Commutator) 6
1.2.1.4. Fırçalar (Brushes) 7
1.2.1.5. ataklar ve Diğerleri 7
1.3. Sabit Mıknatıslı Fırçalı Doğru Akım Motorunun Eşdeğer Devresi..... 8
1.4. Darbe Genişlik Modülasyonu (DGM, PWM) ile Hız Kontrolü 8
2. YAPILAN ÇALIŞMALAR..... 11
2.1. Mikrodenetleyici Seçimi..... 11
2.1.1. Dâhili Analog – Sayısal dönüştürücüsü olmalı. 11
2.1.2. USB Veya Seri (RS-232) Haberleşme Bağlantı Noktası Olmalı. 12
2.1.3. Dâhili Darbe Genişlik Modülasyon (DGM) Birimi Olmalı 13
2.1.4. Yüksek Seviyeli Diller İçin Derleyici ve Dönüştürücü Yazılımı Olmalı 13
2.1.5. Yazılım Geliştirme Ortamı Ücretsiz Olmalı..... 13
2.1.6. Programlama Donanımı Ucuz ve Kolay Elde Edilebilir Olmalı 14
2.1.7. Seçilen Mikrodenetleyici..... 14
2.2. Motor Sürücü Kartı Tasarımı 15

2.3. Yazılım Geliştirme 20
2.3.1. Mikrodenetleyici Yazılımı: 22
2.3.2. Sunucu Bilgisayar Yazılımı: 23
2.3.3. İstemci Bilgisayar Yazılımı 23
3. BULGULAR 24
3.1. Uygulanan Yöntemin Başarısı 24
4. TARTIŞMA ve SONUÇLAR 27
5. ÖNERİLER 28
6. KAYNAKLAR 29
7. EKLER 31

ÖZET

UZAKTAN ERİŞİMLİ SABİT MIKNATISLI DOĞRU AKIM
MOTOR KONTROLÜ DENEY DÜZENEĞİ

Ahmet AKTOĞAN

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Halil İbrahim OKUMUŞ
2011, 41 Sayfa Ekler 37 Sayfa

Uygulamalı eğitimin gerekli olduğu alanlarda uzaktan eğitimin gelişmesinin önündeki en önemli engellerden biri laboratuar deneylerinin internet ortamında gerçekleştirilmesinin zorluğudur. Bu sebeple; internet tabanlı deney düzeneklerinin uzaktan eğitimdeki önemi çoktur. Bu çalışmada; bir sabit mıknatıslı fırçalı doğru akım elektrik motoru deney düzeneği internet tabanlı olarak tasarlanmış ve internet üzerinden deney yapılarak veriler gerçek zamanlı olarak izlenmiştir.

Bu çalışmanın özü fiziksel bir deney ortamının internete açılmasıdır. Deney düzeneğinin bu hali ile birçok eksiği vardır. Fakat bu alanda yapılacak çalışmalara referans olacaktır. İstemci ve sunucu yazılımları bir Java teknolojisi olan bloklamasız, kanal tabanlı giriş çıkış tekniği ile geliştirilmiş, böylece veri istemci ve sunucunun veri alma, işleme ve gösterme performansı önemli ölçüde artırılmıştır. Sunucunun, deney düzeneği ile haberleşmesi RS-232 standardında seri haberleşmedir. Sunucu ile istemcinin internet üzerinden haberleşmesi de TCP/IP teknolojisi ile gerçekleştirilmiştir. Motor hız kontrolü; Darbe Genişlik Modülasyonu (PWM=Pulse Width Modulation) tekniği ile yapılmıştır. Bu çalışma; **gerçek zamanlı** uzaktan erişimli laboratuar çalışmalarına katkı olması bakımından önemlidir.

Anahtar Kelimeler: Uzaktan Eğitim, Uzaktan Erişimli Laboratuar, Motor Kontrol, Darbe Genişlik Modülasyonu, DGM, Bloklamasız Giriş Çıkış.

SUMMARY

EXPERIMENTAL CONTRIVANCE OF REMOTELY ACCESSIBLE PERMANENT MAGNET DC MOTOR CONTROLLING

Ahmet AKTOĞAN

Karadeniz Technical University
Graduate School of Natural & Applied Sciences
Electrical and Electronics Engineering

Supervisor: Asist. Prof. Dr. Halil İbrahim OKUMUŞ
2011, 41 Pages, 37 Pages Appendix

One of the big obstacles against the development of distance learning is the difficulty of realizing laboratory experiments via internet. Composing experimental contrivance that can be used remotely via internet is the most important stage to provide developing distance learning for applied sciences.

In this thesis; we designed an experimental contrivance for permanent magnet brushed dc motor. We used the designed experimental contrivance and made experiments via internet and observed the real time results. We used a Java technology that is named as nonblocking input/output to develop client and server software. The used Java technology provided a remarkably increase of performance of getting, processing and displaying capabilities of client and server. The server communicates with the experimental contrivance by RS-232 standard. The server and client communicate with each other by TCP/IP protocol over the internet. We used pulse width modulation (PWM) to control speed of the motor. The designed experimental contrivance may need a lot of enhancements but it constitutes a basis for making laboratory experiments in distance learning. This thesis gives a contribution to real-time internet based laboratory studies.

Key Words: Distance Learnig, Remote Access Laboratory, Motor Control, Pulse Width Modulation, PWM, Non Blocking Input Output

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 1. Doğru akım motorunun ilke şeması 4
Şekil 2. Fırçalı bir doğru akım motorunun içyapısı 5
Şekil 3. Sürekli mıknatıslı DA motorunun eşdeğer devresi 8
Şekil 4. DGM (PWM) işaretin dalga şekli 9
Şekil 5. ATmega328 programlama ve uygulama kartı 14
Şekil 6. Motor sürücü devre şeması 16
Şekil 7. TIP42C Güç yolu devresi 17
Şekil 8. TIP42C için V_{CEsat}-V_{BEsat}-I_C eğrileri 18
Şekil 9. 2SD1207 V_{CEsat}-I_C grafiği 19
Şekil 10. Motor gerilimi için alçak geçiren süzgeç 20
Şekil 11. Rö akım ölçü direnci gerilimi için alçak geçiren süzgeç 20
Şekil 12. Sistemin donanım mimarisi 21
Şekil 13. Sistemin yazılım mimarisi 21
Şekil 14. Mikrodenetleyici kartı, motor sürücü kartı ve motor 22
Şekil 15. Sistem çalışırken 22
Şekil 16. Motorun, %50 oranında DGM ile başlatılması 24
Şekil 17. Motorun, %50 oranında DGM ile çalışırken durdurulması 25
Şekil 18. Motorun %100 oranında DGM ile başlatılması 25
Şekil 19. Motorun %100 DGM ile çalışırken durdurulması 26

TABLOLAR DİZİNİ

Sayfa No

Tablo 1 . Motor Özellikleri 15

SEMBOLLER DİZİNİ

D	DGM işaretinin doluluk boşluk oranı
DGM	Darbe Genişlik Modulasyonu
E_m	Zıt EMK gerilimi.
f	Bir sinyalin frekansı
fs	Örnekleme frekansı
$f(t)$	DGM (Darbe Genişilik Modulasyonu=PWM) işareti
I_B	Transistörün Base akımı
I_C	Transistörün Kolektör akımı
I_a	Motorun endüvi devresinin akımı.
K_E	Zıt EMK sabiti.
K_T	Moment sabiti
L_m	Endüvi sargısının endüktansı.
M_m	Motorun ürettiği moment
P	Aktif elektriksel güç
R_m	Motorun endüvi devresinin direnci.
T	Bir peryodik işaretin peryod süresi
Ts	Örnekleme peryodu
V_{CEsat}	Transistörün doyum durumunda Emetör – Kolektör arası gerilimi
V_{BEsat}	Transistörün doyum durumunda Emetör – Base arası gerilimi
V_m	Motora uygulanan gerilim·
ω_m	Açısal hız
$\bar{y}$	DGM işaretinin ortalama genlik değeri
$y_{\min}$	DGM işaretinin en düşük genlik değeri
$y_{\max}$	DGM işaretinin en yüksek genlik değeri

1. GENEL BİLGİLER

1.1. Giriş

İnternet erişiminin yaygınlaşması ile internetin eğitim öğretim alanında kullanımı da yaygınlaşmış ve bu alanda internet eksenli birçok yeni yaklaşım ve yöntem geliştirilmiştir. Eskiden mektup, radyo ve televizyon ile yapılan uzaktan eğitimin yerini son zamanlarda internet tabanlı uzaktan eğitim almış ve böylece uzaktan eğitimde çok büyük ve hızlı gelişmeler olmuştur. Uzaktan eğitimle ilgili birçok tanıma rastlamak mümkündür. Bu tanımlar genel olarak birbirlerine benzese de bazıları farklı yaklaşımları içermektedir.

Uzaktan eğitim, geleneksel öğrenme-öğretme yöntemlerindeki sınırlılıklar nedeniyle, sınıf içi etkinliklerin yürütülme olanağı bulunmadığı durumlarda, eğitim öğretim çalışmalarını planlayanlar ve uygulayanlar ile öğrenenler arasında iletişim ve etkileşimin özel olarak hazırlanmış öğretim üniteleri ve çeşitli ortamlar yoluyla belli bir merkezden sağlandığı bir öğretim yöntemidir [1]. Uzaktan Öğretim, fiziksel olarak öğrencilerin bulundukları yerlerde olmasını gerektirmeksizin, teknolojinin imkânlarından yararlanılarak, öğrenci ve öğretmenlerin bir sanal dershane ortamında değişik şekillerde karşı karşıya getirildikleri, planlı bir öğretim şeklidir [2]. Bir başka tanıma göre ise uzaktan eğitim, öğrenci ile öğretenin birbirinden uzakta olmalarına karşın, eş zamanlı (Senkron) ya da ayrı zamanlı (Asenkron) olarak bir araçla iletişim kurdukları bir eğitim sistemidir [3]. Senkron uzaktan eğitim, öğreten ile öğrenenin aynı zamanda etkileşime girmesi durumundaki eğitime verilen isimidir. Asenkron uzaktan eğitim ise öğreten ile öğrenenin ayrı zamanlarda etkileşime girmesi durumudur. Günümüzde uzaktan eğitim, neredeyse tamamen bilgisayar yazılım ve donanım teknolojileri ile internet haberleşme altyapısı üzerinde yükselmektedir [4]. İnternet haberleşmesinin yaygın, kolay ulaşılabilir, ucuz ve hızlı olması; deney düzeneklerinin, ölçüm ve kontrol cihazlarının, hatta laboratuarların uzaktan erişilebilir hale getirilmesi konusundaki çalışmalara ivme kazandırmış, böylece uygulamalı teknik eğitimin uzaktan yapılabilmesinin de önü açılmıştır [5]. Yazılım ve donanım üreticileri de, uzaktan erişilebilen ürünler geliştirip pazara sunmakta ve mevcut ürünlerine veri haberleşme donanım ve yazılımları ekleyerek onları güncelleştirmektedirler.

Ethernet bağlantı noktasına sahip osiloskoplar, ölçü aletleri, veri toplama kart ve sistemleri, PLC (Programlanabilir Mantık Denetleyicisi)'ler ve kontrol üniteleri gibi ürünler pazardaki donanımlara örnek olarak verilebilir. Bu ürünlerin bir kısmı sahip oldukları Ethernet bağlantı noktasından hizmet veren bir gömülü web sunucu yazılımını da içermektedirler. Ayrıca çeşitli şirketler ve araştırmacılar, laboratuar cihazlarına internetten erişime imkân veren yazılım paketleri sunmaktadırlar.

Bu paketlerden biri, National Instruments şirketinin, LabView uygulaması ile birlikte çalışan InternetToolkit yazılımıdır ve bu paket G Web Server adında bir web sunucu yazılımını da içermektedir. Bu paket sayesinde, osiloskop, ölçü aleti ve sinyal üreteci gibi cihazların ön panellerine standart bir internet gezgini penceresinden erişme imkânı sunulmaktadır. Erişilen bu ön paneller gerçek cihazların görüntüleri olup bu görüntülere VI (Virtual Instruments) yani Sanal Cihaz adı verilmektedir. Bu görüntüler iki biçimde sunulmaktadır. Birincisi cihaz ön panellerinin statik görüntülerinin istemciye gönderilmesi (snapshot mode) şeklinde diğeri ise izleme (monitor mode) biçimindeki hizmettir. İzleme biçiminde; cihaz ön panel görüntüleri animasyon etkisi ile sunulmaktadır. Bu, Server-Push teknolojisi ile başarılmaktadır. Server-Push teknolojisi sayesine sunucu, sanal cihazın ön panel görüntülerini; istemciden istek beklemeksizin ardışık çerçeveler şeklinde istemciye göndermektedir. Çerçeve gönderme hızı, G-Web Server'ın yapılandırma dosyasında ayarlanabilmektedir. Fakat Server-Push teknolojisinin tüm internet gezginleri tarafından desteklenmemesi [6] önemli bir sakıncadır.

Diğer yazılım paketi ise MathWorks şirketinin MATLAB uygulaması ile entegre çalışan MATLAB Web Server (MWS) yazılımıdır. MWS sayesinde, MATLAB tabanlı benzetimlere ve uygulamalara internet üzerinden erişebilmek bunları çalıştırabilmek ve sonuçları yine web tabanlı olarak görüntüleyebilmek mümkün olmaktadır. Fakat MWS'in e-öğrenme ortamları için olumsuzluklarından biri, benzetim için gerekli giriş verilerinin paket halinde istemciden alınması ve sonuçların aynı şekilde bir bütün olarak kullanıcıya gönderilmesidir. Bu nedenle anlık parametre değişimlerine karşın sistemin tepkisi gözlenememektedir. Özellikle sürgü (slider) olarak isimlendirilen ve bir parametrenin değerini sürgü kolunun yatay ya da dikey olarak hareket ettirilmesi ile değiştirmeye yarayan ara yüz öğelerinin, MWS ile kullanılamaması son derece ciddi bir sakıncadır [7]. MWS, http protokolünü kullandığı için istemci ile sunucu arasındaki haberleşme istek-cevap mekanizması şeklinde cereyan eder. Kullanıcı, çalıştırılacak uygulamanın giriş parametrelerini web sayfası aracılığı ile sunucuya iletir, sunucu da uygulamayı çalıştırır ve

elde ettiği grafikleri JPEG formatında bir görüntü dosyasına dönüştürerek sunucuda saklar ve istemciye geri gönderir.

MATLAB ve LabView altyapılarının, istemci ile sunucu arasında, birçok deneyin gerektirdiği kesintisiz veri akışı ihtiyacını karşılayamamaları, lisans ücretlerinin yüksek olması, özel, pahalı bilgisayar donanımları ve veri toplama kartları ile birlikte çalışıyor olmaları önemli sakıncalardır. Ayrıca bu altyapılar elektronik deneyler için değil endüstriyel uygulamalar için tasarlanmışlardır[8]. Literatürde LabView/InternetToolkit ve MATLAB/MWS altyapılarının kullanıldığı birçok çalışma bulunmaktadır [9-24].

Bu çalışmada; platform bağımsız, hiçbir özel yazılım ve donanım paketi gerektirmeyen, ucuz, gerçek zamanlı ve internet tabanlı bir deney düzeneği yazılımı ve donanımı tasarlanmıştır. Teknik eğitimde laboratuar deneylerinin çoğu basit ve ucuz elektronik devreler ve düzeneklerle gerçekleştirilir. Deney düzeneğinden alınan işaretler ve ölçümler de genelde herhangi bir işleme ve analize tabi tutulmadan gösterilir. Yani deneylerin genelinde sinyal işleme, sayısal analiz ihtiyacı olmadığı gibi çok hassas ölçümlere de gerek duyulmamaktadır. Ayrıca her deney düzeneğinin kendine özgü bir donanımı ve ölçüm prosedürü vardır. Bu sebeple MATLAB ve LabView gibi çok yüksek veri işleme ve analiz kapasitesine sahip yazılımlar ve bu yazılımların kullandığı hassas ve pahalı veri toplama donanımları çoğu deney için gereksizdir.

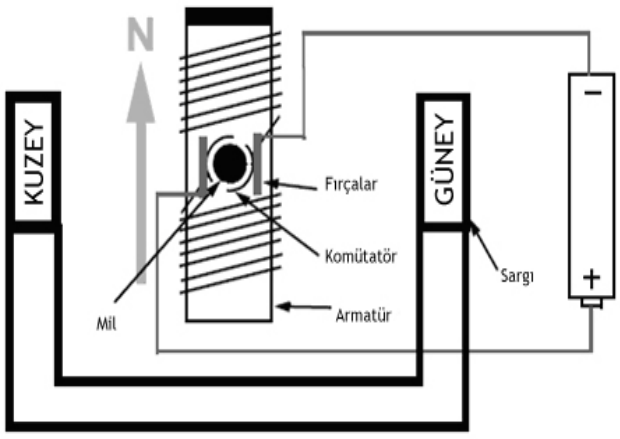
Bu çalışmanın literatürde yer alan benzer çalışmalara göre en önemli farkı, istemci ile sunucu arasında gerçek zamanlı ve kesintisiz veri akışı sağlayan, kurulum ve kullanımı kolay olan ucuz bir yazılım-donanım altyapısı sunmasıdır. Çalışmada örnek deney olarak, Sabit Mıknatıslı bir DA motorunun hızı, Darbe Genişlik Modülasyonlu (DGM=PWM) işaret ile denetlenmiş, motorun akım, gerilim ve hız değerleri anlık olarak gerçek zamanlı bir şekilde internet üzerinden izlenmiştir. Sistem hem uzaktan izleme hem de uzaktan kontrol yapmaya uygundur. Sistem GeZİLab (**Ge**rçek **Z**amanli **İ**nternet **Lab**oratuarı) olarak adlandırılmıştır.

Sabit mıknatıslı fırçalı doğru akım motorları yaygın bir kullanıma sahiptir. Bu sebeple; fırçalı doğru akım motorlarının yapısının, davranışlarının ve kontrolünün öğrenilmesi ve uygulamalı olarak analiz edilmesi önemlidir. Genel olarak elektrik makinelerinin akım ve gerilim değişimlerine karşı verdikleri tepki yavaş olduğundan bu tepkinin ölçülmesinden elde edilen verilerin değişim hızları da yavaştır. Veri değişim hızının yavaş oluşu bu verilerin gerçek zamanlı olarak iletiminde düşük hızlı haberleşme kanallarının kullanılmasına imkân vermektedir. Bu çalışmada tasarlanan deney

düzeneğinde sabit mıknatıslı fırçalı bir doğru akım motorunun hızı, Darbe Genişlik Modülasyonu yöntemi ile kontrol edilmekte ve motorun akım, gerilim ve hız gibi parametleri internet veya yerel ağ üzerinden gerçek zamanlı olarak izlenebilmektedir. Ayrıca motorun başlama ve durma gibi geçici durumlarına ilişkin verilerin de izlenme imkânı vardır. Bu çalışmanın esası herhangi bir deney düzeneğinin gerçek zamanlı olarak internete açılabilmesine imkân olup olmadığının araştırılmasıdır. Bu sebeple nispeten kolay kontrol edilebildiği ve kontrolu için basit donanım gerektiği için sabit mıknatıslı fırçalı doğru akım motoru seçilmiştir. Ayrıca sayısal olarak üretimi ve izlenmesi kolay olduğu için de kontrol yöntemi olarak Darbe Genişilik Modülasyonu (PWM= Pulse Width Modulation) kullanılmıştır.

1.2. Doğru Akım Motorları

Elektrik enerjisini mekanik enerjiye dönüştüren aygıtlara Elektrik Motorları denir. Her elektrik motoru biri sabit (Stator) ve diğeri kendi çevresinde dönen (Rotor ya da Endüvi) iki ana parçadan oluşur. Doğru akım motorlarının çalışma ilkesi ilk kez Faraday tarafından ortaya konulmuştur. Faraday'ın bulduğu doğru akım motorlarının çalışma ilkesi, "Manyetik alan içerisinde bulunan bir iletkenden elektrik akımı geçirilirse iletken hareket eder." şeklinde ifade edilebilir.



Şekil 1. Doğru akım motorunun ilke şeması

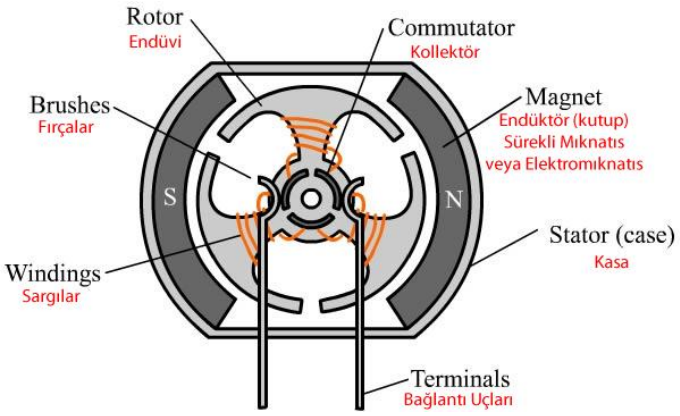
Kuzey ve Güney (N-S) kutuplarından meydana gelen mıknatısın manyetik hatlarının yönünün N'den S'ye doğru olduğu kabul edilir. Diğer taraftan, bir iletkenden elektrik akımı geçerse bu iletkenin etrafında bir manyetik alan meydana gelir. Doğru akım

motorlarının çalışma ilkesi, mıknatısın ve üzerinden elektrik akımı geçen iletkenin etrafında meydana gelen manyetik hatların, birbirini itmesi ile gerçekleşmektedir. Rotorun sürekli dönmesi, her yarım devirde kolektör dilimleri ile temas durumunda olan fırçalar aracılığı ile sağlanmaktadır. Fırçalar ve komütatör, sargılardan geçen akımın yönünü, her yarım devirde bir değiştirmektedir.

Günümüzde elektrik enerjisini mekanik enerjiye çeviren makinelerden biri olan DA motoru ve sürücü sistemleri, endüstrinin birçok kolunda kullanılmaktadır. Yıllar önce, konum kontrolü için kullanılan servo motorların birçoğu alternatif akım (AA) ile çalıştırılmaktaydı. AA motorların kontrolünün zor ve doğrusal olmayan özelliklerinin baskın olması sebebiyle DA motorlar birçok uygulamada tercih edilir. Diğer yandan DA motor içindeki fırça ve komütatör, bakımı zorlaştırmakta ve masrafı artırmaktadır. DA motor ve güç elektroniği teknolojisindeki gelişmeler sayesinde motor hacmi başına düşen moment artırılmakta ve sürekli mıknatıslı motor türleri gelişme göstermektedir. Böylece fırça ve komütatör bakım sakıncalarını büyük ölçüde azamiye indirilmekte ve DA motor uygulama alanının daha da genişlemesine imkân sağlamaktadır.

1.2.1. Doğru Akım Motorlarının Yapıları

Doğru akım motorları, endüktörün yapısına bağlı olarak elektromıknatıslı ve sabit mıknatıslı olmak üzere iki şekilde imal edilir. Bu ikisi arasında endüktör haricinde yapı bakımından farklı bir özellik yoktur.



Şekil 2. Fırçalı bir doğru akım motorunun içyapısı

1.2.1.1. Endüktör (Kutup)

Doğru akım motorlarında manyetik alanın meydana geldiği kısımdır. Endüktöre kutup da denilmektedir. Kutup uzunluğu yaklaşık olarak endüvi uzunluğuna eşittir. Endüktörler tabii mıknatıslarla yapıldığı gibi kutuplara sargılar sarılarak, bu sargıların enerjilendirilmesiyle mıknatıslık özelliği kazandırılmış elektromıknatıslardan da yapılabilir. Çok küçük doğru akım motorlarında kutuplar (tabii mıknatıslı) sabit mıknatıslıdır. Doğru akım motorlarında kutup sayısı makinenin gücüne ve devir sayısına göre değişir.

1.2.1.2. Endüvi (Rotor)

Gerilim indüklenen ve iletkenleri taşıyan kısma endüvi denir. Endüvi, kalınlığı 0,30-0,70 mm arasında değişen dinamo saclarının üst üste konulması ile elde edilir. Tabakalar Eddy akım hatlarını sınırlar kayıplarını azaltır. Dinamo sacları, istenen şekil ve ölçüde preslerle kesildikten sonra tavlanır ve birer yüzeyleri yalıtılır. Yalıtma fuko kayıplarını azaltır ve endüvinin tabakalı yapısını korur. Yalıtma işleminde kâğıt, lak kullanılır ve oksit tabakası oluşturulur. Endüvi sacları üzerine iletkenleri yerleştirmek için oluklar açılır. Bu olukların şekil ve sayıları makinenin büyüklüğüne, sarım tipine, sarım şekline ve devir sayısına göre değişir. Oluklar, küçük güçlü makinelerde yuvarlak veya oval büyük güçlü makinelerde ise tam açık olarak yapılırlar.

1.2.1.3. Kolektör (Commutator)

Motor uçlarına uygulanan gerilimin endüviye iletilmesini sağlar. Doğru akım motorlarında endüviye uygulanacak gerilimin iletilmesini kolektörler sağlar. Kolektör dilimleri, haddeden geçirilmiş sert bakırdan pres edilerek yapılır. Bakır dilimleri arasına 0,5-1,5 mm kalınlığında mika veya mikanit yalıtkan konur. Bu kalınlık, kolektörün çapına ve komşu dilimler arasındaki gerilim farkına göre değişir. Kolektör dilimleri ile bunlara temas eden fırçalar, bağlama elemanlarını teşkil ederler. Kolektör, hareketli olduğundan doğru akım motorlarının en önemli ve en çok arıza yapan parçasıdır. Bu nedenle kolektör

dilimleri, özenle yapılır ve dilimler arası gerilim farkı 15 voltu geçmeyecek şekilde ayarlanır.

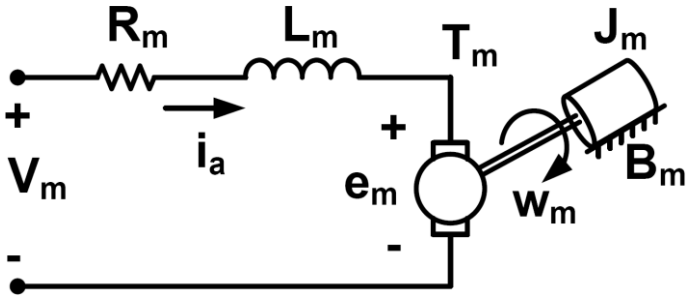
1.2.1.4. Fırçalar (Brushes)

Doğru akım motorlarında dış devredeki akımı endüviye iletebilmek için fırçalar kullanılır. Doğru akım makinelerinde aşınma ve iyi komütasyon elde etmek için saf bakır fırça kullanılmaz. Fırçalar; makinenin akım şiddeti ve gerilimine göre sert, orta sert ve yumuşak karbon veya karbon alaşımından yapılır. Genel olarak küçük güçteki (10 kW'a kadar) doğru akım motorlarında bütün fırça çeşitleri ile iyi çalışabilir. Mümkün olduğu kadar bir motorda aynı cins fırçalar kullanılmalı ve fırça boyları da eşit olmalıdır. Fırçalar, dik ve yatay olarak yapılırlar. Çok küçük güçlü motorlarda fırçalar, kapak üzerine açılmış ve yalıtılmış yuvalara konulur. Büyük güçlü motorlarda ise fırça yuvaları, sac veya dökümden yapılmış olup fırça tutucularına monte edilir. Fırçaların kolektör yüzeyine oturup, işletme boyunca durumunu muhafaza edebilmelerini fırça tutucuları sağlar. Fırça tutucuları, eğik ve dik olmak üzere iki tiptir. Fırça tutucuları, fırça taşıyıcılarıyla monte edilir.

1.2.1.5. Yataklar ve Diğerleri

Yatakların görevi, motorun hareket eden kısımlarının mümkün olduğu kadar az kayıpla gürültüsüz ve bir eksen etrafında rahatça dönmesini sağlamaktır. Doğru akım motorlarında bilezikli yataklar ve rulmanlı (bilyeli ve makaralı) yataklar kullanılır. Kolay değiştirilebilir olması ve sürtünme kayıplarının ihmal edilecek derecede olması nedeniyle küçük ve orta güçlü motorlarda hemen hemen yalnız rulmanlı yataklar kullanılmaktadır. Bu yatakların en büyük sakıncası, fazla gürültü yapmasıdır. Elektrik motorlarının en önemli parçalarından biri de yataklardır. Yataklar, çok arıza yapan ve bakım isteyen kısımdır. Yataklarda meydana gelen aşınmalar, sürtünmeler komütasyonun bozulmasına ve en büyük arızaların doğmasına neden olur.

1.3. Sabit Mıknatıslı Fırçalı Doğru Akım Motorunun Eşdeğer Devresi



Şekil 3. Sürekli mıknatıslı DA motorunun eşdeğer devresi

Buradaki seri bağlanmış L_m endüktansı, endüvi sargısının endüktansını, R_a motorun iki terminali arasındaki direnci temsil eder. Motor döndüğü zaman, bir çıkış momenti ve besleme gerilim kaynağına zıt bir gerilim üretilir. Bu zıt gerilim burada e_m olarak gösterilmiştir. Bu zıt gerilime, Ters Elektromotor Kuvvet (Back Electromotor Force =BEMF) de denir.

$$v_m(k) = R_m i_a(k) + L_m \frac{di_a(k)}{dt} + e_m(k) \quad \textbf{(1)}$$

$$e_m(k) = K_E \omega_m(k) \quad \textbf{(2)}$$

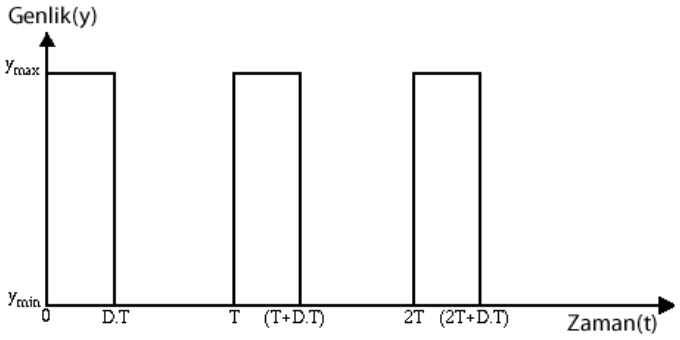
Motorun ürettiği tork (m_m), endüvinin sargısından geçen akım (i_a) ile doğru orantılıdır.

$$m_m(k) = K_T i_a(k) \quad \textbf{(3)}$$

1.4. Darbe Genişlik Modülasyonu (DGM, PWM) ile Hız Kontrolü

Doğru akım motorunun hızı direkt olarak kaynak gerilimi ile doğru orantılıdır. Dolayısı ile eğer kaynak gerlimini yarıya düşürürsek motor da yarı hızda çalışacaktır. Bu yüzden doğru akım motorlarında hız kontrolünün esası motora uygulanan ortalama gerilimi değiştirmeye dayanır. Motora uygulanan ortalama gerilimi değiştirmenin en verimli yolu ise kaynak gerilimini anahtarlayarak kıymaktır. Eğer anahtarlama hızı yeterli olursa motor anahtarlamayı hissetmeyecek fakat motora uygulanan ortalama gerilim değişmiş olacaktır. Bu, anahtarlamalı hız kontrolünün prensibidir. Böylece motorun hızı motora uygulanan darbenin süresi (genişliği) ile değişmiş olacaktır.

Darbe Genişlik Modülasyonlu (DGM) bir işaretin dalga şekli aşağıdaki gibidir.



Şekil 4. DGM (PWM) işaretin dalga şekli

Yukarıdaki $f(t)$ kare dalgasını göz önüne alalım. Bu işaretin en düşük genlik değerine $y_{\min}$, en yüksek genlik değerine $y_{\max}$ ve doluluk boşluk oranına (duty cycle) D diyelim. Bu sinyalin ortalama değeri:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)dt \qquad \textbf{(4)}$$

$f(t)$ peryodik bir kare dalga olduğu için bu dalganın genlik değeri $0 < t < D.T$ aralığında $y_{\max}$ ve $D.T < t < T$ aralığında $y_{\min}$ olacaktır. Dolayısı ile bu sinyalin ortalama değeri:

$$\bar{y} = \frac{1}{T}\int_0^{DT} y_{\max}dt + \int_{DT}^{T} y_{\min}dt \qquad \textbf{(5)}$$

$$\bar{y} = \frac{D.T.y_{\max} + T(1-D)y_{\min}}{T} \qquad \textbf{(6)}$$

$$\bar{y} = D.y_{\max} + (1-D)y_{\min} \qquad \textbf{(7)}$$

$y_{\min}$ =0 olduğu için:

$\bar{y} = D.y_{\max}$ olur. **(8)**

Son eşitlikten görüleceği gibi sinyalin ortalama değeri olan $\bar{y}$, direkt olarak doluluk boşluk oranı olan D'ye bağlıdır. Yani sinyalin doluluk boşluk oranı değiştirilirse o sinyalin ortalama değeri değiştirilmiş olur.

2. YAPILAN ÇALIŞMALAR

2.1. Mikrodenetleyici Seçimi

Mikrodenetleyici seçilirken aşağıdaki kıstaslar göz önüne alınmıştır.

2.1.1. Dâhili Analog – Sayısal dönüştürücüsü olmalı.

Mutlaka gerekli olmamakla birlikte, dâhili analog-sayısal dönüştürücünün olması, geliştirme işinde çok kolaylık sağlayacaktır. Donanım açısından bakıldığında, harici analog-sayısal dönüştürücü ek donanım tasarımı gerektirecektir. Ayrıca harici analog-sayısal dönüştürücünün mikrodenetleyiciye bağlanması için mikrodenetleyicinin birçok giriş-çıkış ucunun kullanılması gerekecek ve ek olarak arayüz tasarlanması gerekecektir. Yazılım açısından bakıldığında; harici dönüştürücünün mikrodenetleyici ile haberleşmesi için ilgili dönüştürücüye ait haberleşme protokolünün öğrenilmesi ve bu protokolde haberleşme yapabilmek için ek kod yazılması gerekecektir. Bu sebeple dâhili analog sayısal dönüştürücüye sahip bir mikrodenetleyici seçilmelidir.

Analog-sayısal dönüştürücünün hassasiyetinin 5mV olması yeterli olacaktır. 5 Volt beslemeye sahip bir sistemde 5mV'luk hassasiyet için 10 Bit'lik dönüşüm gereklidir. Bu sebeple analog sayısal dönüştürücünün 10 bit çözünürlükte olması yeterlidir. Yapılan araştırmalarda sabit mıknatıslı fırçalı küçük doğru akım motorlarının darbe işaretine tepki süresinin 10mS civarında olduğu görülmüştür. Elimizdeki motorla yapmış olduğumuz deneyler de bunu doğrulamıştır. Bu sebeple motorun herhangi bir parameteresinin gözlenebilmesi için o parametreden her 10 mS içinde en az iki örnek almak gereklidir. Parametre başına örnek peryodu $Ts = 5mS$ olacaktır. Bir kanal için örnekleme hızı:

$$f_{1s} = \frac{1}{Ts} = \frac{1}{5.10^{-3}} = 200\,{örnek}/{saniye} \qquad \textbf{(9)}$$

Motorun, akım, gerlim ve hız olmak üzere üç parametresi gözlenecektir. Bu sebeple; motorun başlatma anındaki geçici durumunu gözleyebilmek için her parametreden her 10mS içinde en az iki ölçüm değeri alınmalı bu üç farklı analog değer eş zamanlı olarak

veya birbirine çok yakın zamanlarda sayısala dönüştürülmelidir. Bu sebeple en az üç kanallı bir analog sayısal dönüştürücüye ihtiyaç vardır. Bu üç kanallı analog sayısal dönüştürücünün her kanalının her 10mS içinde en az iki örnekleme yapabilmesi gerekir. Gerilim, akım ve hız değerlerinin her biri için saniyede 200'er örnek alınması gerektiğinden, bir saniyedeki toplam örnek sayısı aşağıdaki gibi olacaktır:

$$Toplam-örnek = (200+200+200) = 600 \quad \textbf{(10)}$$

$$fs = \frac{6*1000}{10} = 600\, örnek/saniye \quad \textbf{(11)}$$

Yani kulanacağımız analag sayısal dönüştürücünün kanal başına dönüştürme süresi en fazla 1,66 mS olmalıdır:

$$Ts = \frac{1}{fs} = \frac{1}{600} = 1,66mS \quad \textbf{(12)}$$

Ayrıca analog – sayısal dönüşümün tamamlandığını gösteren bir kesme işareti üretmelidir. Aksi halde dönüşümün tamamlandığını anlamak için yazılımda bir bekleme ve gözetleme mekanizması kurmak gerekecektir ki bu da zaman kritik uygulamalr için önemli bir dezavantajdır.

2.1.2. USB Veya Seri (RS-232) Haberleşme Bağlantı Noktası Olmalı.

Mikrodenetleyici ünitenin sunucu bilgisayar ile haberleşebilmesi için uygun bir bağlantı noktası olamsı gerekir. Aksi halde ek donanım ve yazolım tasarımına ihtiyaç olacaktır. Günümüz bilgisayarlarında yaygın olarak kullanılan haberleşme bağlantı noktalrı USB ve RS-232 bağlantı noktalarıdır.

2.1.3. Dâhili Darbe Genişlik Modülasyon (DGM) Birimi Olmalı

Motor hız kontrolünde darbe genişlik modülasyonu kullanılacağı için, mikrodenetleyicinin dâhili bir DGM birimi olması tercih edilir. Aksi halde ya harici bir DGM donanımının kullanılması gerekecek veya mikrodenetleyicinin giriş çıkış uçlarından biri DGM sinyalini üretmek için kullanılacaktır. Mikrodenetleyiciler işlerini sıra ile yapmaktadırlar. Mikrodenetleyicinin giriş-çıkış uçlarından birinin DGM sinyali üretmek için kullanılması durumunda mikrodenetleyicinin bu sinyali üretmesi için zaman harcaması gerekecek ve analog – sayısal dönüşüm işlemi ve veri haberleşme işleminin zamanlaması aksayarak beklenen sonucu vermeyecektir. Çünkü buradaki uygulama zaman kritik bir uygulamadır.

2.1.4. Yüksek Seviyeli Diller İçin Derleyici ve Dönüştürücü Yazılımı Olmalı

Makine dili ile programlama küçük projeler için tercih edilebilir. Fakat mikrodenetleyici üreticilerinin birçoğu veya üçüncü parti sağlayıcılar, mikrodenetleyiciler için yüksek seviyeli dil derleyicileri geliştirmişleridir. Bu derleyiciler sayesinde; mikrodenetleyici yazılımları için artık C, C++, Java, .NET, Python… gibi yüksek seviyeli diller kullanılabilmektedir. Eğer mikrodenetleyicinin yüksek seviyeli dillerden biri için bir derleyicisi varsa o mikrodenetleyicinin makine veya assembly dilini öğrenmek zorunda kalmadan etkili sistemler geliştirilebilir.

2.1.5. Yazılım Geliştirme Ortamı Ücretsiz Olmalı

Birçok mikrodnetleyici firması sistem geliştirme ortmlarını çok yüksek fiyatlara satmaktadırlar. Seçeceğimiz mikrodenetleyicinin yazılım geliştirme ortamının ücretsiz olamsı gerekir.

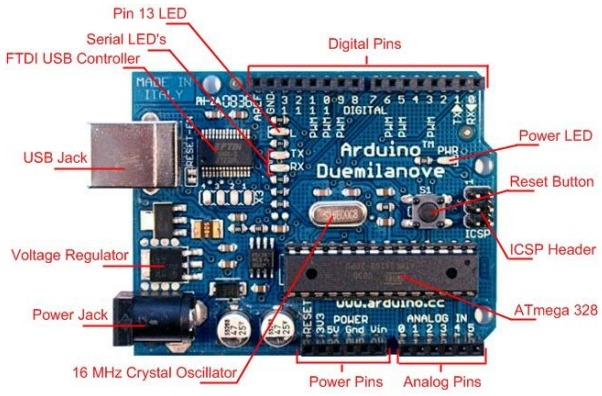
2.1.6. Programlama Donanımı Ucuz ve Kolay Elde Edilebilir Olmalı

Birçok mikrodenetleyici için özel ve pahalı programlama donanımlarına ihtiyaç duyulmaktadır. Bu proje için seçeceğimiz mikrodenetleyicinin programlama donanımı ucuz ve kolay elde edilebilir olmalıdır.

2.1.7. Seçilen Mikrodenetleyici

Yukarıdaki kıstaslar göz önüne alınarak yapılan araştırmalar neticesinde Atmel firmasının ATMega328 mikrodenetleyicisine karar verilmiştir. Bu mikrodenetleyici kullanımı kolay ve ucuz bir programlama kartı piyasada var olduğundan yeni bir kart tasarımı gereksiz görülerek kart satın alındı. Bu mikrodenetleyici için geliştirilmiş ücretsiz bootloader yazılımı da vardır. Bootloader yazılımı, bilgisayarda yazılımış olan kodların seri bağlantı noktası üzerinden mikrodenetleyiciye yüklenmesine imkân tanımaktadır. Mikrodenetleyicinin programlanması ve deneylerin yapılması aynı kart ile mümkündür. Yani mikrodenetleyici, devrede programlanabilme (ICP=In Circuit Programmable) özelliğine sahiptir. Bu mikrodenetleyicinin bir de C dili dönüştürücüsü vardırdır. Bu dönüştürücü sayesinde C dili ile yazılmış programlar makine diline dönüştürülerek mikrodenetleyicide koşulabilmektedir. Mikrodenetleyici, programlama ve deneme kartı, bootloader yazılımı ve geliştirme ortamı bütünleşmiş bir platform oluşturmakta ve bu platform piyasada Arduino platformu olarak bilinmektedir.

ATmega328 mikrodenetleyicisinin programlama ve uygulama kartı aşağıdadır.



Şekil 5. ATmega328 programlama ve uygulama kartı

2.2. Motor Sürücü Kartı Tasarımı

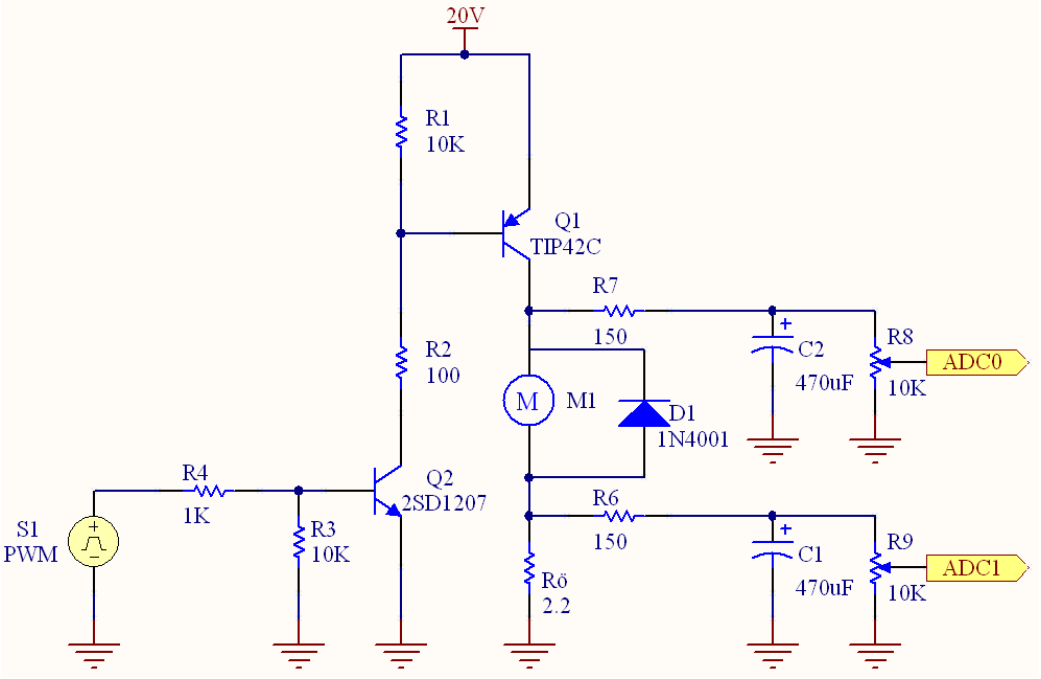
Elde bulunan iki kutuplu sabit mıknatıslı fırçalı bir doğru akım motorunun kullanılmasına karar verildiği için bu motoru sürebilecek bir kart tasarlanmıştır. Elde bulunan motor HP/Mabuchi DC/PM C2162-60006 / DN505728 model sabit mıknatıslı fırçalı bir doğru akım motorudur. Motorun özellikleri aşağıdaki tabloda verilmiştir. Sürücü kartının tasarımında göz önüne almamız gereken motor parametreleri şunlardır:

- Motor besleme gerilimi (V_m): 19V
- Motor maksimum akımı (I_a): 2.5A
- Motor Direnci (R_m):8.2Ω

Motor akımını ölçmek için düşük değerli bir direnç motorun akım yoluna bağlanacaktır. Ölçü devresinin basit olması açısından ölçü direncinin bir ucu sıfır seviyesinde olmalıdır. Böylece mikrodenetleyici kart ile sürücü kart arasında referans seviye bakımından fark oluşmayacak, akım ve gerilim ölçümleri daha kolay olacaktır.

Tablo 1. Motor Özellikleri

Besleme Gerilimi (Volt)	19 (Maks. 24)
Maksimum yüklü iken akımı (Stall Current) (Amper)	2.5
Yüksüz iken akımı (Amper)	0.15
Başlatma Gerilimi (Volt)	2
Direnç (Ohm)	8.2
Motor Sabiti (N-cm/√watt)	1.8
Tork Sabiti (N-cm/amper)	4.8
Maksimum Yükteki Tork (N-cm)	28.7
Yüksüz Hızı (rpm)	4550
Maks. Güç (Watt)	34.2
Maks. Güç (mili-HP)	45.83
Energy (Joule)	1026
Ağırlık (Gram)	224



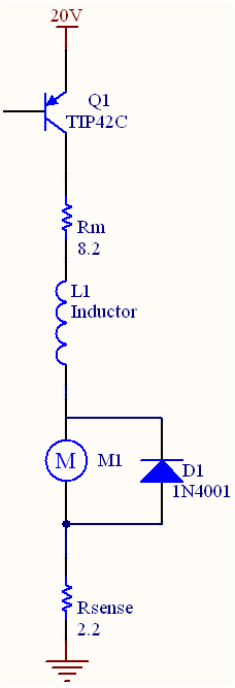
Şekil 6. Motor sürücü devre şeması

Motor geriliminin ölçümünün ve hesaplanmasının kolay olması için High-Side sürme tekniği kullanılmıştır.

Sürücü devrenin güç transistörü olarak TIP42C seçilmiştir. TIP42C'ni kolektör akım sınırları sürekli olarak 6A, darbeli olarak 10A dir. Ayrıca C-E arası kırılma gerilimi V_{CEO}=100V tur. Bu sınır değerler kullandığımız motorun akım ve gerilim sınır değerleri için uygundur.

Eldeki motorun maksimum akımı 2.5A, maksimum gerilimi de 24V'tur. Motor sürücü devresi 20 V' kaynak ile beslenecektir. Motor direnci 8.2Ω dur.

Bu veriler göz önüne alındığında Q1 güç transistörünün emetör-toprak arasındaki eşdeğer devresi aşağıdaki gibi olacaktır.



Şekil 7. TIP42C Güç yolu devresi

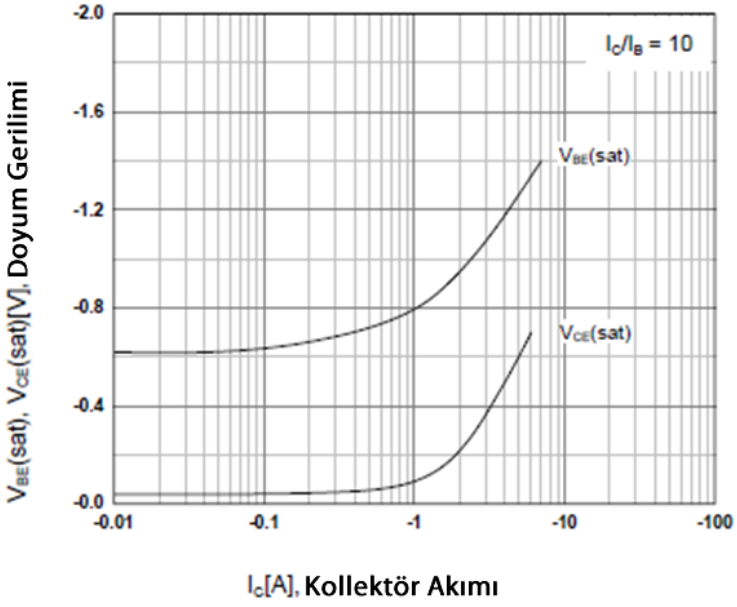
Şekil 8 deki devreye göre doğru gerilim denklemi aşağıdaki gibi olur.

$$V_{CE1sat} + \mathrm{I}_{C1} * (R_m + R_{ö}) = 20 \qquad \textbf{(13)}$$

TIP42C'nin veri sayfalarından elde edilen Şekil 9 deki grafik incelendiğinde 2-3A civarındaki kolektör akımına karşılık V_{CE1sat} değerinin 0.3V civarında olduğu görülecektir. Böylece TIP42C'nin kolektör akımı aşağıdaki gibi hesaplanır:

$$\mathrm{I}_{c1} * (R_m + R_{ö}) = 20 - V_{CE1sat} \qquad \textbf{(14)}$$

$$\mathrm{I}_{c1} = \frac{19.7}{(R_m + R_{ö})} = 1,89A \qquad \textbf{(15)}$$



Şekil 8. TIP42C için V_{CEsat}-V_{BEsat}-I_C eğrileri

Yukarıdaki grafikten elde edilen bilgiye göre TIP42C için $\frac{I_{C1}}{I_{B1}} = 10$ ' dur. Bu bilgiden yola çıkarak 1.89A kollektör akımına karşılık base akımı aşağıdaki şekilde hesaplanır.

$$I_{B1} = 1{,}89 / 10 = 0{,}189A = 189mA \qquad \textbf{(16)}$$

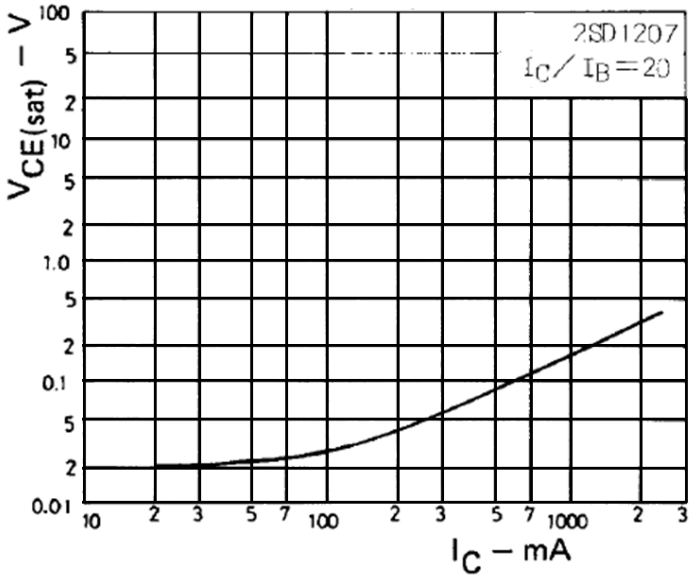
Yani Q1 güç transistörünün kolektöründen 1,89A akım akması için base akımının $I_{B1} = 189mA$ olması gerekir.

Q1 güç transistörünün base akımını belirleyen, R2 direnci ile Q2 transistörünün V_{CE2sat} değeridir. Ayrıca yaklaşık olarak $I_{B1} = I_{C2}$ dir. Bu bilgiler ışığında Q2 transistörünün gerilim denklemini aşağıdaki gibi yazıp R2 direncinin değerini hesaplayabiliriz.

$$V_{BE1sat} + V_{CE2sat} + I_{C2} * R_2 = 20 \qquad \textbf{(17)}$$

V_{BE1sat} değeri, yine Şekil 7 den bulunabilir. Şekil 7'ye göre 2-3A civarındaki kolektör akımına karşılık V_{BE1sat} değerinin 1V civarında olduğu görülecektir.

V_{CE2sat} değerini bulmak için 2SD1207 transistörünün veri sayfasından alınan aşağıdaki grafik incelenmelidir.



Şekil 9. 2SD1207 V_{CEsat}-I_C grafiği

Şekil 10 incelendiğinde 180mA kolektör akımına karşılık $V_{CE2sat} = 0,04V$ civarında olduğu görülür. Bu bilgiler ışığında Q2 ve Q1 için gerilim denklemini yeniden yazarsak:

$$V_{BE1sat} + V_{CE2sat} + \mathrm{I}_{C2} * R_2 = 20 \quad \textbf{(18)}$$

Değerleri yerine koyup hesapladığımızda:

$$1 + 0,04 + 0,189 * R_2 = 20 \quad \textbf{(19)}$$

$$R_2 = 100\Omega \quad \textbf{(20)}$$

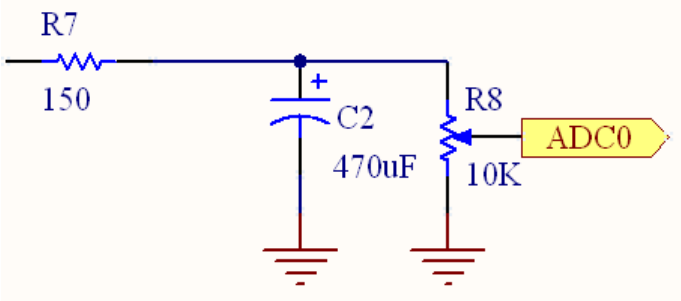
R_2 direncinde harcanan güç aşağıdaki gibi hesaplanır.

$$P_{R2} = {I_{R2}}^2 * R_2 \quad \textbf{(21)}$$

$$P_{R2} = (0,189)^2 * 100 = 3,57W \quad \textbf{(22)}$$

Bu değerlere göre R_2 direnci 4W'lık seçilebilir.

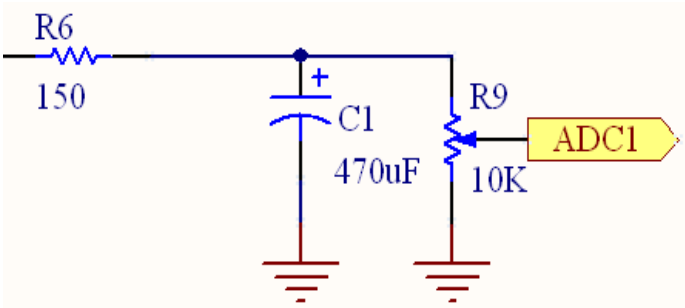
Motor, DGM sinyali ile sürüleceğinden darbe geçişleri sırasında motor akım ve geriliminde sıçrama parazitleri oluşacaktır. Bu parazitlerin sönümlenmesi için motor akım ve gerilimleri RC alçak geçiren süzgeçlerinden geçirilmiştir. Motorun üst ucundan okunacak gerilim için aşağıdaki filtre kullanılmıştır.



Şekil 10. Motor gerilimi için alçak geçiren süzgeç

Motor çıkış gerilimi R8 ayarlı direnci ile bölünerek analog sayısal dönüştürücü giriş geriliminin maksimum değerini aşmaması sağlanır. Buradaki bölme işleminden elde edilen katsayı yazılımda kullanılarak verinin gösterim sırasında normalize edilmesi sağlanmıştır.

Aynı şekilde motor akımının hesaplanması için kullanılan ölçü direncinden elde edilen gerilim de aşağıdaki gibi alçak geçiren filtreden geçirilmiş ve çıkış gerilimi R9 ayarlı direnci ile bölünmüştür.

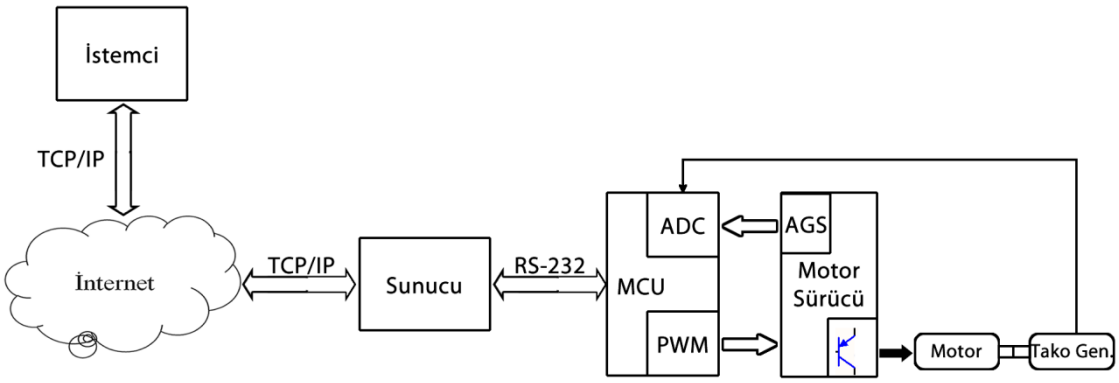


Şekil 11. Rö akım ölçü direnci gerilimi için alçak geçiren süzgeç

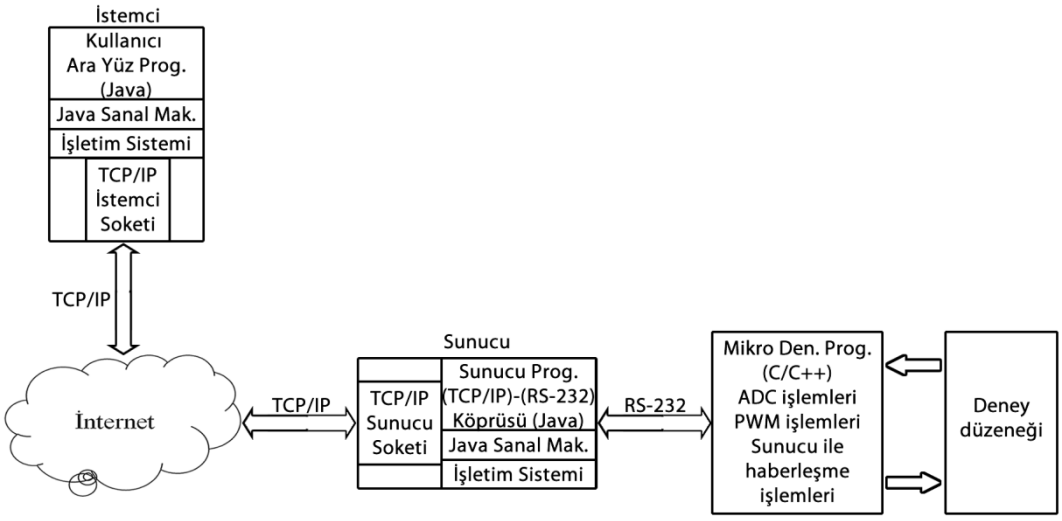
2.3. Yazılım Geliştirme

Yazılım geliştirme çalışmaları üç kısımda yapılmıştır. Birinci kısım mikrodenetleyici tarafındaki yazılım, ikinci kısım sunucu bilgisayarda çalışacak olan

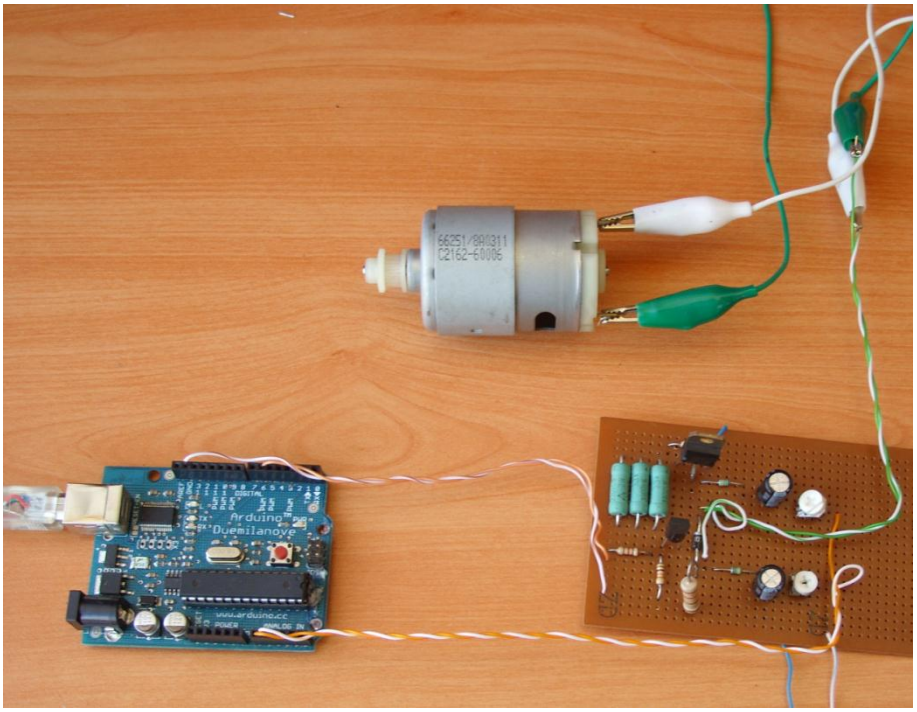
yazılım, üçüncü kısım ise istemci tarafında çalışacak olan yazılımdır. Bu yazılım modüllerinin ayrıntılarına girmeden önce sisteme genel olarak bakmakta fayda var.



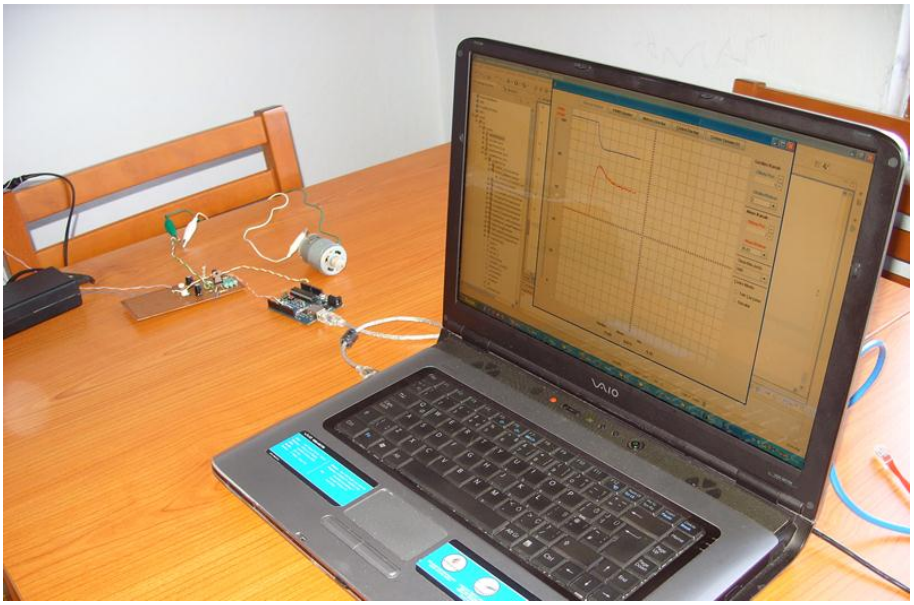
Şekil 12. Sistemin donanım mimarisi



Şekil 13. Sistemin yazılım mimarisi



Şekil 14. Mikrodenetleyici kartı, motor sürücü kartı ve motor



Şekil 15. Sistem çalışırken

2.3.1. Mikrodenetleyici Yazılımı:

Mikrodenetleyicideki yazılım üç ana görevi yerine getirmektedir.

- Kullanıcıdan gelen parametrelere uygun DGM işaretini üretmek

- Sürücü karttan analog verileri okuyup sayısala dönüştürmek
- Sunucu bilgisayarla RS-232 bağlantı noktası üzerinden haberleşerek sunucu bilgisayardan aldığı komutları yerine getirmek ve sayısala dönüştürmüş olduğu analog verileri sunucu bilgisayara göndermek.

Mikrodenetleyici üzerindeki yazılım Arduino yazılım geliştirme platfotmu üzerinde C dili ile yazılmıştır.

2.3.2. Sunucu Bilgisayar Yazılımı:

Sunucu bilgisayarda çalışan yazılım istemci ile mikrodenetleyici ünite arasında köprü olarak çalışmakta iki ana görevi yerine getirmektedir.

- İstemciden TCP/IP protokolü ile gelen komutları RS-232 protokolü ile mikrodenetleyici birime göndermek
- Mikrodenetleyiciden RS-232 protokolü ile aldığı verileri TCP/IP protokolü ile istemciye göndermek.

Sunucu tarafındaki yazılım Java dili ile geliştirilmiştir. Görevlerini eş zamanlı olarak yapabilmesi için yazılımda Java Thread (kanal) teknoloji kullanılmıştır.

2.3.3. İstemci Bilgisayar Yazılımı

İstemci bilgisayarda çalışan yazılım üç ana görevi yerine getirmektedir.

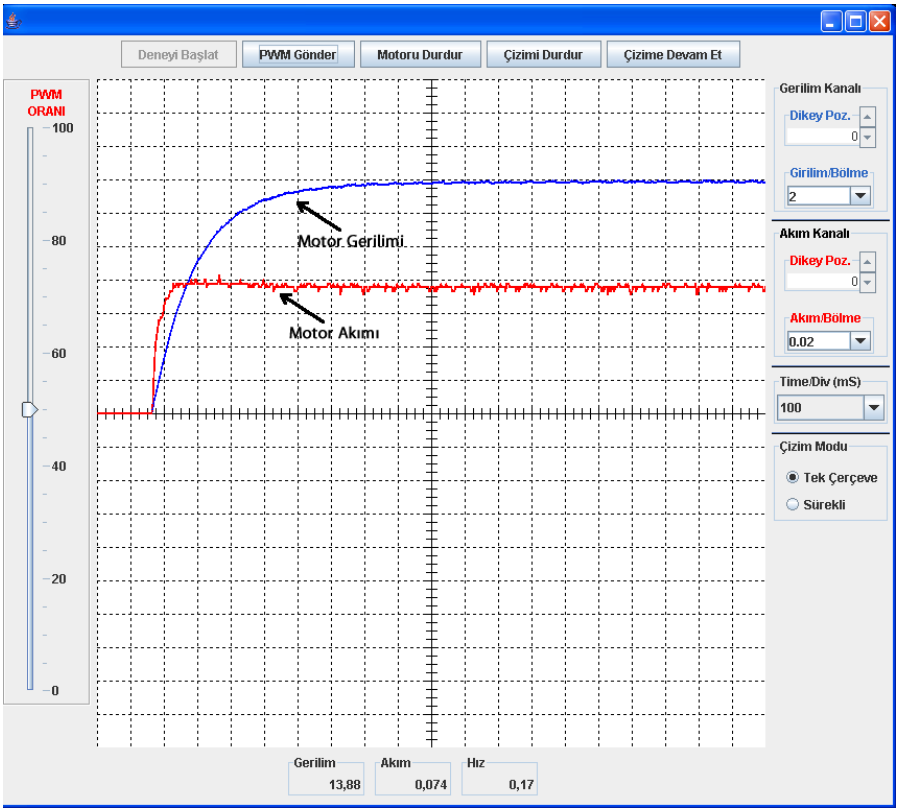
- Grafik kullanıcı arayüzü vasıtası ile kullanıcıdan komut almak
- Kullanıcı komutlarını TCP/IP protokolü ile sunucu bilgisayara göndermek
- Sunucu bilgisayardan TCP/IP protokolü ile gelen veri paketlerini çözümleyerek grafikleri çizmek

İstemci tarafındaki yazılım Java dili ile geliştirilmiştir. Veri haberleşme performansını artırmak için bir Java'nın bloklamasız giriş çıkış (Non Blocking IO) teknolojisi ve bellek görüntü kaynağı (Memory Image Source) teknolojisi kullanılmıştır.

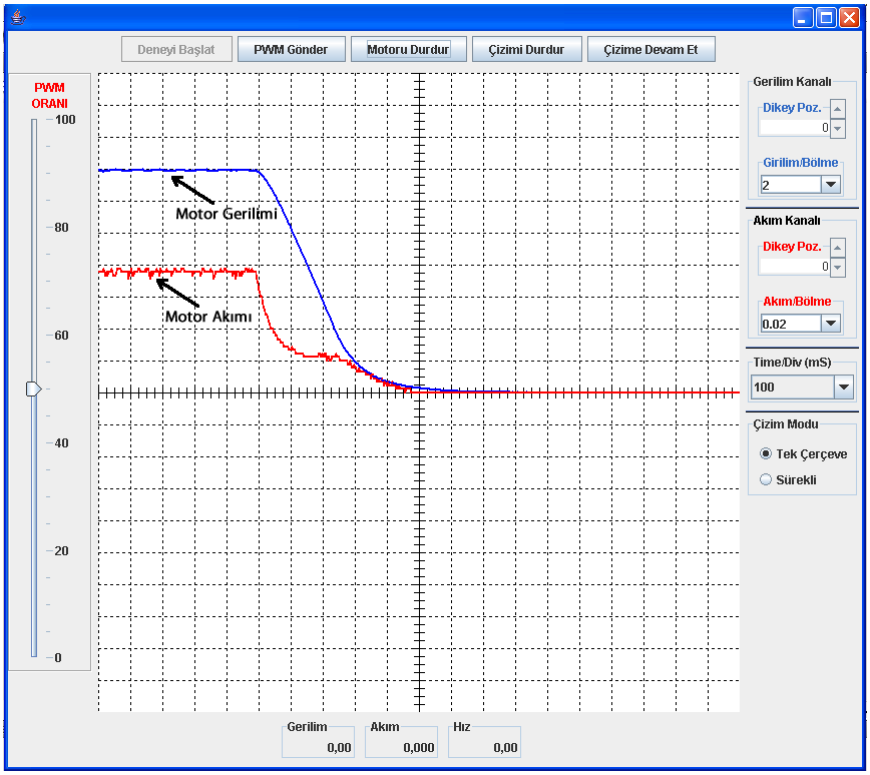
3. BULGULAR

3.1. Uygulanan Yöntemin Başarısı

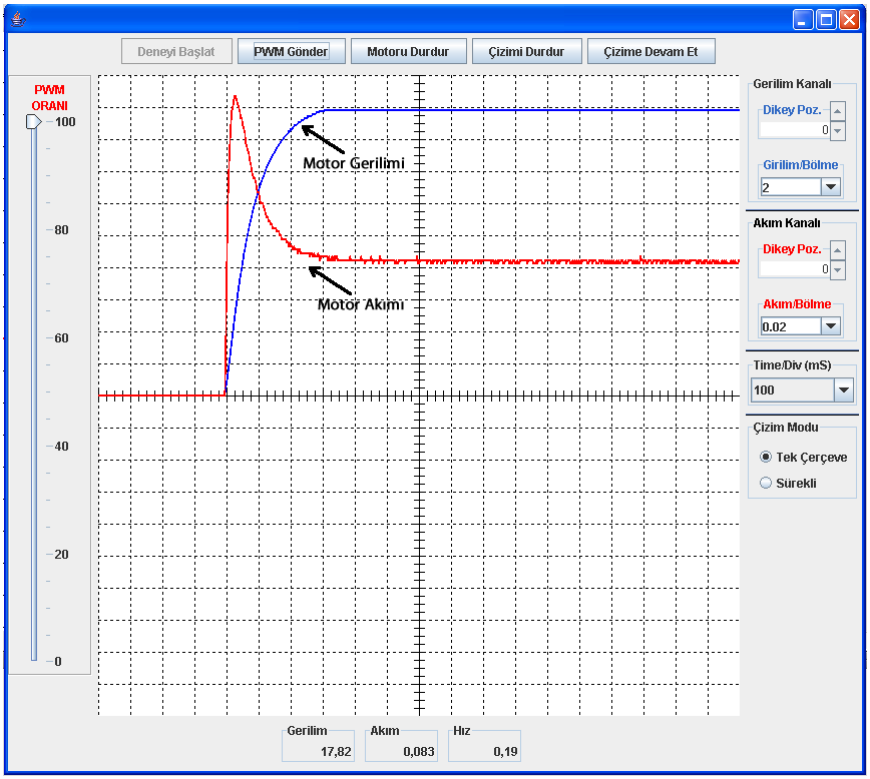
Yapılan çalışma başarı ile sonuçlanmıştır. Elde bulunan motor farklı oranlarda DGM sinyali ile sürülerek motorun akım ve gerilim grafikleri eş zamanlı şekilde istemci bilgisayar ekranında çizdirilmiştir. Aşağıdaki şekillerde, motorun farklı DGM oranlarında başlatılması ve durdurulması ile elde edilen akım gerilim grafikleri gösterilmiştir.



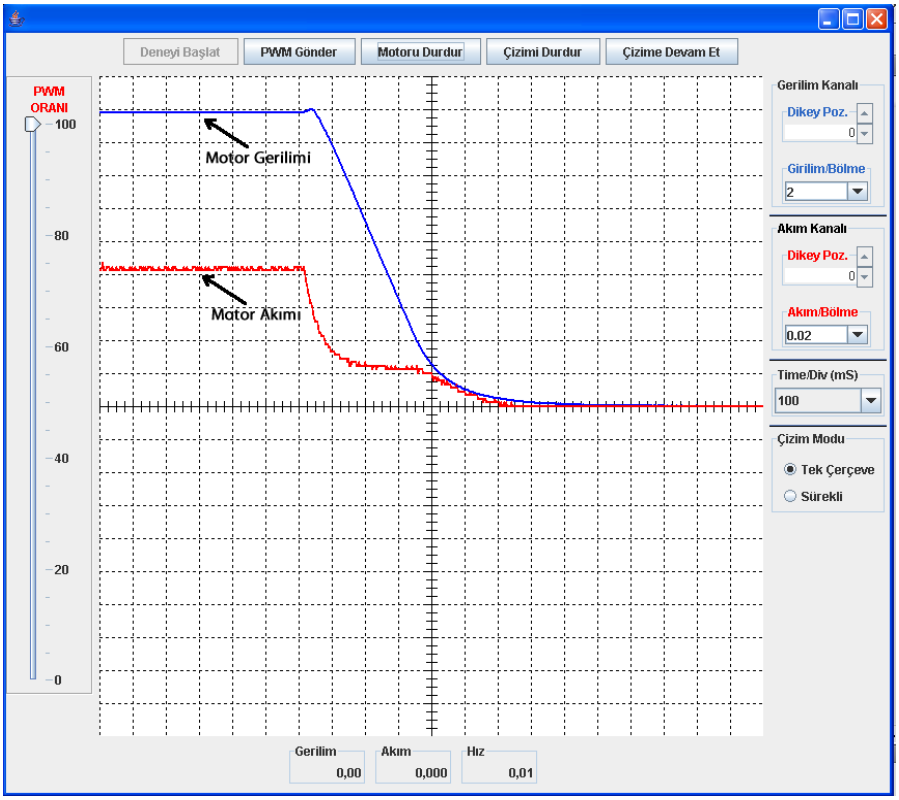
Şekil 16. Motorun, %50 oranında DGM ile başlatılması



Şekil 17. Motorun, %50 oranında DGM ile çalışırken durdurulması



Şekil 18. Motorun %100 oranında DGM ile başlatılması



Şekil 19. Motorun %100 DGM ile çalışırken durdurulması

Şekil. 16, 17,18,19 motor başlatılırken ve durdurulurken motor akımının ve geriliminin değişim grafiklerini göstermektedir. Motorun armatür akımı aşağıdaki denklem ile ifade edilir.

$$i_a = \frac{Vm - e_a}{Ra} \qquad \textbf{(23)}$$

Zıt EMK e_m Denklem 2'de gösterildiği gibi motorun açısal hızı ile doğru orantılıdır.

Motor'a gerilim uygulandığı anda motor hızı sıfır olduğu için zıt EMK e_a'da sıfırdır. Bu yüzden başlangıç anında motor akımı: $i_a = \frac{Vm}{Ra}$ şeklinde olup en yüksek değerindedir. Hız arttıkça zıt EMK e_a'da artacağından hız sabit bir değere ulaşıncaya kadar motor akımı da düşmektedir.

4. TARTIŞMA ve SONUÇLAR

Bu çalışmada uzaktan erişimli laboratuarlarda kullanılabilecek gerçek zamanlı uzaktan erişimli bir deney düzeneği geliştirilmiştir. Bir deney düzeneğinin uzaktan erişimli olabilmesinin önündeki en ciddi engelin uzaktan erişimde kullanılacak olan haberleşme kanallarının veri haberleşme hızlarıdır. Uzaktan erişimli deney düzenekleri tasarlanırken bu kısıtlama göz önüne alınmalıdır. Mevcut kullanımdaki intermet erişim hızı zamandan zamana çok değişiklik göstermektedir. Bu sebeple veri haberleşme kanalı olarak internetin kullanılacağı uygulamalarda istatistiksel hız verileri dikkate alınarak uygulama geliştirilmelidir.

Geliştirilmiş olan sistem gerçek saha şartlarında test edilmiş ve başarı ile çalıştırılmıştır. Sistem bir prototip olduğu için sisteme, kullanıcı erişim kontrolü, kullanıcı oturum takibi, ses ve video aktarımı gibi özellikler eklenmemiştir. Bu sistemin; geliştirme aşamasında kullanılan programlama dilleri ve mikro denetleyici bakımından da bir bağımlılığı yoktur. Zira sunucu ve istemci yazılımları, Java dışında herhangi bir dil kullanılarak da yazılabilir. Çünkü güncel programlama dillerinin hemen hepsi TCP/IP soketlerine dil düzeyinde erişim imkânı sağlamaktadırlar. Deney düzeneğinin kontrol birimindeki mikro denetleyici yerine başka bir mikro denetleyici de kullanılabilir Sistemden elde edilen yüksek performans sebebi ile bu yöntem uzak erişimli laboratuar tasarımları için bir referans teşkil edebilir.

5. ÖNERİLER

Yapılan bu çalışmada gerçek zamanlı uzaktan erişimli deney düzeneklerinin başarılı olabileceği görülmüştür. Veri haberleşme alt yapıları çok hızlı gelişmektedir. Bu gelişmeler sayesinde, bu tür çalışmalar için en ciddi kısıtlama olan veri haberleşme hızı engeli ortadan kalkacaktır. Bu sebeple bu alanda yapılacak çalışmalara devam edilmeli ve yeni sistemler geliştirilmelidir. Böylece, uzaktan eğitimin gelişmesi ve daha yararlı olmasına katkıda bulunulmuş olacaktır.

6. KAYNAKLAR

[1] http://egitek.meb.gov.tr/KapakLink/UzaktanEgitim/UzaktanEgitim.html, [3Haziran2011]

[2] http://www.uzem.sakarya.edu.tr/index.php/uzaktanegretim.html, [5Haziran 2011]

[3] http://www.drcetiner.org/uzaktan_egitim, [9Haziran 2011]

[4] Xuemin Chen, Gangbing Song, Yongpeng Zhang, "Virtual and Remote Laboratory Development: A Review", Earth and Space 2010: Engineering, Science, Construction, and Operations in Challenging Environments: 3843-3852, 2010 ASCE

[5] Raymond Berntzen, Jan Olav Strandman, Tor A. Fjeldly, Michael S. Shur, "Advanced Solutions For Performing Real Experıments Over The Internet", International Conference on Engineering Education , Oslo, Norway: 6B121-6B126, August 6–10, 2001

[6] K.K. Tan, K.N. Wang, K.Z. Tang.. "Mechatronıc Experiment On Remote Vibration Signature Analysis Via The Internet", Proceedings of the 2002 IEEE Intemational Conference on Control Applications Glasgow, Scotland, U.K., 310-315, September 18-20, 2002

[7] Irmak, E., "E-Öğrenme Ortamları İçin Matlab Web Sunucu Kullanımı", Gazi Üniv. Müh. Mim. Fak. Der. Cilt 23, No 2, 495-506, 2008

[8] Ying-Wen Bai and Cheng-Yu Hsu, "Design and Implementation of an Embedded Remote Electronic Measurement System", IMTC 2006 – Instrumentation and Measurement Technology Conference Sorrento, Italy: 1311-1316, 24-27 April 2006

[9] Ján MOLNÁR, Irena KOVÁČOVÁ, "Distance Remote Measurement Of Magnetic Field", Acta Electrotechnica et Informatica No. 4, Vol. 7, 2007

[10] Christof Rohrig, Andreas Jochheim "Java-Based Framework For Remote Access To Laboratory Experiments", In Proc. of the IFAC/IEEE Symposium on Advances in Control Education, Gold Coast, Australia, 2000

[11] Peng Yan, Mikko Valkama, and Markku Renfors, "Distance Learning in Communications Signal Processing using MATLAB Web Server", Proceedings of the 6th Nordic Signal Processing Symposium – NORSIG 2004 Espoo, Finland:244-247, June 9 - 11, 2004

[12] Ewald, H., and Page, G.F., "Performing Experiments by Remote Control Using the Internet. Global Journal of Engineering Education. 4(3):287-292.

[13] Hong Shen, Zheng Xu, B. Dalager, V. Kristiansen, Ø. Strøm, Michael S. Shur, Tor A. Fjeldly, Jian-Qiang Lü, and T. Ytterdal.. "Conducting Laboratory Experiments over the Internet", IEEE Transactıons On Educatıon, Vol. 42, No. 3: 180-185, August 1999

[14] Ersin Bozkurt, Ahmet Sarıkoç ," Fizik Eğitiminde Sanal Laboratuar, Geleneksel Laboratuarın Yerini Tutabilir Mi?", Selçuk Üniversitesi Ahmet Keleşoğlu Eğitim Fakültesi Dergisi, Sayı: 25, Sayfa 89 -100, 2008

[15] J.L. Díez, M. Vallés, A.Valera, "A Global Approach For The Remote Process Simulation And Control", 15th Triennial World Congress, Barcelona, Spain, 2002

[16] Colak İ, Demırbas Ş.,, Seref Sagıroglu, Erdal Irmak, "A Novel Web-Based Laboratory for DC Motor Experiments", © 2009 Wiley Periodicals, Inc. Comput Appl Eng Educ 19: 125–135, 2011

[17] Çolak, İ., Irmak, E., Demirbaş Ş., Sağıroğlu, Ş., "A Novel İntegrated Web Based Learning System For Electrical Machines Education", IEEE International Conference on Power Engineering, POWERENG '07, Energy and Electrical Drives Setubal, Portugal: 265-269, 2007

[18] Christof Rohrig, Andreas Jochheim, "The Virtual Lab for Controlling Real Experiments via Internet", Proceedings of the 1999 EEE, International Symposium on Computer Aided Control System Design, MP3-4 5:20 Kohala Coast-lsland of Hawai'i, Hawai'i, USA: 279-284, August 22-27, 1999

[19] C. C. KO, Ben M. Chen, S. H. Chen, V. Ramakrishnan, R. Chen, S. Y Hu, Y Zhuang, "A Large-Scale Web-Based Virtual Oscilloscope Laboratory Experiment", Engıneerıng Scıence And Educatıon Journal: 69-76, April 2000

[20] Chi Chung Ko, Senior Member, IEEE, Ben M. Chen, Shaoyan Hu, Vikram Ramakrishnan, Chang Dong Cheng, Yuan Zhuang, and Jianping Chen, "A Web-Based Virtual Laboratory on a Frequency Modulation Experiment", IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 31, No. 3: 295-303, August 2001

[21] Rumen Iv. Amaudov , Ivo N. Dochev, "Functional Generator and Data Acquisition System Controlled by Internet", IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sofia, Bulgaria: 276-278, 5-7 September 2005

[22] Akif Kutlu, Kubilay Tasdelen,"Remote Electronic Experiments Using LabView Over Controller Area Network", Scientific Research and Essays Vol. 5(13) ISSN 1992-2248 ©2010 Academic Journals: 1754-1758, 4 July, 2010

[23] Ahmet Tekin, Fikret Ata, "Sanal Bir Laboratuar: Asenkron Motorların Hız Denetimi", Fırat Üniv. Mühendislik Bilimleri Dergisi, 22 (1): 73-82, 2010

[24] Şevki Demirbaş, "İnternet Tabanlı PI Kontrollü Bir Doğru Akım Motoru Deney Seti", Gazi Üniv. Müh. Mim. Fak. Der. Cilt 22, No 2: 401-410, 2007

7. EKLER

Ek-1 : Mikrodenetleyici Tarafındaki Yazılımın Kodları

```
int inByte;
int i=0;
int j=0;
int baud = 19200;
int k=0;
int volt=0, cur=0, spd=0;
String komut;
unsigned long t;
unsigned long t1,t2,t3,t4,t5;
int markSize=0;
int samples=6;
byte dataBuffer[6]; //dataBuffer must be=samples+markSize
void setup() {
  Serial.begin(baud);
  pinMode(6,OUTPUT);
}
void loop() {
  while (Serial.available() == 0) {
    ;
  }
  readPort();
}
void readPort(){
  inByte = Serial.read();
  procCommand(inByte);
}
void procCommand(byte comm){
  switch(comm)  {
  case 80:
    pwm();
    break; //P
  case 68:
    _stop();
    break; //D
default:
    inByte=68;
    break;

  }
}
void pwm(){
  while (Serial.available() == 0){
    ;
```

```
  }
  inByte=Serial.read();
  analogWrite(6, (int)inByte);
  captureData();
}

void captureData(){

  do {

    i=0;

    while(i<samples){

      volt=analogRead(0);

      cur=analogRead(1);

      spd=analogRead(2);

      dataBuffer[i]=lowByte(volt);
      i++;
      dataBuffer[i]=highByte(volt);
      i++;
      dataBuffer[i]=lowByte(cur);
      i++;
      dataBuffer[i]=highByte(cur);
      i++;
      dataBuffer[i]=lowByte(spd);
      i++;
      dataBuffer[i]=highByte(spd);
      i++;

    }

   Serial.write(dataBuffer, samples+markSize);
  }
  while(Serial.available() == 0);
  readPort();
}

void _stop(){
  analogWrite(6, 0);
}
```

Ek-2: Sunucu Tarafındaki Yazılımın Kodları

```
import processing.serial.*;
import processing.net.*;

Server server;
Serial myPort;
int xPos = 0;
int p=0;
int command;
int f=0;
int i=0;
int s=0;
int raw=0;
float v1,i1,s1;
int markSize=0;
int bufferLen=6;
byte[] inBuffer = new byte[6];
int g1,g2;
int timeStep=1;

int sd=0;

Client cl;

void setup () {

  size(800, 600);
  smooth();

  println(Serial.list());

  myPort = new Serial(this, Serial.list()[1], 19200);

  myPort.buffer(bufferLen);

  server = new Server(this, 5204);

  drawGrid();
}
void draw () {

  cl=server.available();
  if (cl != null&&cl.available()>0) {
    // Receive the message
    command= cl.read();
    println(command);
if(command=='P') {
```

```
myPort.write('P');
      myPort.write(cl.read());
    }
  }
}

void serialEvent (Serial myPort) {

  if(raw>255) raw=0;
  int dataread=myPort.readBytes(inBuffer);
  server.write(inBuffer);
  if(sd==255) {
    sd=0;
  }
  //  println("data read:"+dataread);
  // println("time:"+readTwoBytePackFloat());
  // println("last:"+readTwoBytePackFloat());
  i=0;
  if (inBuffer != null) {
    while(i<(bufferLen-markSize)) {
      float v2= byteToFloat(i);
      float i2= byteToFloat(i+2);
      float s2= byteToFloat(i+4);
      if(s==0) {
        v1=v2;
        i1=i2;
        s1=s2;
        s=1;
      }
      stroke(0,0,255);
      line(xPos, 225 - round(v1), xPos+timeStep, 225 - round(v2));
      stroke(255,0,0);
      line(xPos, 450 - round(i1), xPos+timeStep, 450 - round(i2));
      stroke(0,0,0);
      line(xPos, 675 - round(s1), xPos+timeStep, 675 - round(s2));
      v1=v2;
      i1=i2;
      s1=s2;
      i=i+6;
      xPos=xPos+timeStep;
      if (xPos >= width) {
        xPos = 0;
        println("server frame: "+ f++);
        drawGrid();
      }
    }
  }
}
```

```
float byteToFloat (int i) {
  int lowByte =int( inBuffer[i]);

  int hiByte = int(inBuffer[i+1]);
  hiByte=hiByte<<8;

  float inValuef = float(hiByte|lowByte);
  inValuef = map(inValuef, 0, 1023, 0, 200);
  return inValuef;
}

void mousePressed() {

  println("-clic"+i);
  myPort.write('P');
  myPort.write(p++);
  if(p>99) {
    p=0;
  }
}

void drawGrid() {
  background(132,247,173);
  for(g1=0;g1<height;g1=g1+20) {
    stroke(255,255,0);
    line(0, g1, width, g1);
  }
  for(g2=0;g2<width;g2=g2+20) {
    stroke(255,255,0);
    line(g2, 0, g2, height);
  }
}
```

Ek-3: İstemci Tarafındaki Yazılımın Kodları

```
package master;

import java.nio.*;

import java.nio.channels.*;

import java.net.*;
import java.text.NumberFormat;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.io.IOException;

import javax.swing.JLabel;
import javax.swing.JPanel;

public class nbClient extends Thread {
  private int xPos = 1;
  private int p = 0;
  private Graphics2D g2d;

  private int k = 0;
  private int s = 0;

  private float v1 = 0, i1 = 0, s1 = 0;
  private int timeStep = 1;

  private int byteCount;

  private JPanel box;
  private JLabel speed;
  private JLabel cur;
  private JLabel volt;
  drawGrid dg = new drawGrid();

  byte[] inBuffer = new byte[6];
  private int port = 5204;

  private SocketAddress address;
  private SocketChannel client;
  private ByteBuffer buffer;
  private ByteBuffer recBuffer;

  private WritableByteChannel out;
private boolean rec = false;
```

```
  private double voltScale = 1;
  private int voltPos = 0;

  private double curScale = 1;
  private int curPos = 0;

  private boolean drawStatus = true;
  private boolean contMode = true;

  private double rsVal = 2.2;

  private float vAvg = 0, iAvg = 0, sAvg = 0;
  private int avg = 0;
  private int avgStep = 8;

  public nbClient(JPanel b, JLabel v, JLabel c, JLabel s) {

          address = new InetSocketAddress("localhost", port);

          try {
                  client = SocketChannel.open(address);
          } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
          }

          buffer = ByteBuffer.allocate(6);

          out = Channels.newChannel(System.out);

          box = b;
          speed = s;
          cur = c;
          volt = v;

          dg.draw(box);
  }

  float byteToFloat(int i) {

          int lowByte = (inBuffer[i]) & (0XFF);

          int hiByte = (inBuffer[i + 1]) & (0XFF);
          hiByte = hiByte << 8;
float inValuef = (hiByte | lowByte);
inValuef = (inValuef / 1023) * 5;
return inValuef;
  }
```

```
  void startMotor(int val) {
        ByteBuffer outBuffer = ByteBuffer.allocate(2);
        if (val > 255) {
              val = 255;
        }
        outBuffer.put((byte) 80);
        outBuffer.put((byte) val);
        outBuffer.flip();
        try {
              client.write(outBuffer);
        } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
        }

  }

  public void run() {
        g2d = (Graphics2D) box.getGraphics();
        while (true) {
              scope(g2d);
        }

  }

  public void scope(Graphics2D g2d) {

        g2d.setStroke(new BasicStroke((float) 1.3));
        int bytesRead = 0;
        try {
              buffer.clear();
              bytesRead = client.read(buffer);

        } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
        }
        if (drawStatus) {
              if (bytesRead >= 6) {
                    buffer.flip();
                    // buffer.get(inBuffer);
                    inBuffer = buffer.array();
int rem = bytesRead % 6;
  if (rem != 0) {
        bytesRead = bytesRead - rem;
                    }
                    k = 0;

                    while (k < bytesRead) {
```

```
                                        float v2 = (float) ((float) byteToFloat(k)); // motorun
üst

                                        // ucundaki

                                        // gerilim
                                        float i2 = (float) (byteToFloat(k + 2) / rsVal); // ölçü

                                                // direnci

                                                // üzerindeki

                                                // gerilim
                                        // ölçü direncine bölünerek akım bulunur
                                        float s2 = byteToFloat(k + 4); // hız gerilimi

                                        float i2s = (float) (i2 / curScale) * box.getHeight() /
20;

                                        float vm = (float) ((float) (v2 - (i2 * rsVal)) * 18.5 /
5); // motor

                                                                        // gerilimi=v2-Vrs
                                        float v2s = (float) (vm / voltScale) * box.getHeight() /
20;
                                        float s2s = (float) (s2 / curScale) * box.getHeight() /
10;

                                        iAvg = iAvg + i2;

        vAvg = vAvg + vm;
        sAvg = sAvg + s2;
        avg++;
          if (avg == avgStep) {
          vAvg = vAvg / avgStep;
          iAvg = iAvg / avgStep;
          sAvg = sAvg / avgStep;
          volt.setText("" + formatNumber(vAvg, 2));
          cur.setText("" + formatNumber(iAvg, 3));
          speed.setText("" + formatNumber(sAvg, 2));

                                                avg = 0;
                                                vAvg = 0;
                                                iAvg = 0;
                                                sAvg = 0;
                                        }

                                        if (s == 0) {
```

```
                                        v1 = v2s;
                                        i1 = i2s;
                                        s1 = s2;
                                        s = 1;
                                }

                                g2d.setColor(new Color(0, 0, 255));
                                g2d.drawLine(xPos, box.getHeight() / 2 - voltPos
                                        - Math.round(v1), xPos + timeStep,
box.getHeight()
                                        / 2 - voltPos - Math.round(v2s));

                                // System.out.println("v2 :"+v2);
                                v1 = v2s;

                                g2d.setColor(new Color(255, 0, 0));
                                g2d.drawLine(xPos, box.getHeight() / 2 - curPos
                                        - Math.round(i1), xPos + timeStep,
box.getHeight()
                                        / 2 - curPos - Math.round(i2s));

                                // System.out.println("i2 :"+i2);
                                i1 = i2s;

                                k = k + 6;

                                xPos = xPos + timeStep;

                                if (xPos >= box.getWidth() && contMode) {

                                        xPos = 0;

                                        dg.draw(box);
                                }
                        }

                }
        }
    }

    public void setVoltScale(double d) {
        voltScale = d;
    }

    public void setVoltPosition(int pos) {
        voltPos = pos;
    }

    public void setCurScale(double d) {
```

```
            curScale = d;
    }

    public void setCurPosition(int pos) {
            curPos = pos;
    }

    public void setTimeStep(int ts) {
            switch (ts) {
            case 100:
                    timeStep = 1;
                    break;
            case 50:
                    timeStep = 2;
                    break;
            case 30:
                    timeStep = 3;
                    break;

            }
}

    public void setDrawStatus(boolean ds) {
            drawStatus = ds;
    }

    public void setContMode(boolean cm) {
            contMode = cm;
    }

    public void setRsVal(double rs) {
            rsVal = rs;
    }

    public String formatNumber(float num, int dec) {
            NumberFormat nf = NumberFormat.getNumberInstance();
            nf.setMinimumFractionDigits(dec);
            nf.setMaximumFractionDigits(dec);
            return nf.format(num);

    }

}
```

```
package master;

	import java.awt.BasicStroke;
	import java.awt.Color;
	import java.awt.Graphics;
	import java.awt.Graphics2D;
	import java.awt.image.BufferedImage;

	import javax.swing.JPanel;

	public class drawGrid {
	  JPanel gp;
	  BufferedImage grid; // declare the image

	  final static float dash1[] = { 3.0f };

	  final static BasicStroke dashed = new BasicStroke(1.0f,
			BasicStroke.CAP_BUTT, BasicStroke.JOIN_MITER, 1.0f, dash1,
0.0f);
	  private int gap = 8;

	  public void draw(JPanel gp) {
		this.gp = gp;
		// paint background
		Graphics2D g2 = (Graphics2D) gp.getGraphics(); // we need a Graphics2D

		// context

		if (grid == null) {
			// Compute the grid only one time
			int w = gp.getWidth();
			int h = gp.getHeight();
			grid = new BufferedImage(w, h,
BufferedImage.TYPE_INT_ARGB);
			// grid = (BufferedImage)(this.createImage(w,h));
			Graphics2D gc = grid.createGraphics();
			gc.setStroke(dashed);
			gc.setColor(new Color(0, 0, 0));

			for (int x = 0; x < w; x += gap) {
				int subVer=x%8;
				int mainVer=x%32;
				if (subVer == 0) {
					gc.setStroke(new BasicStroke(1));
					gc.drawLine(x, 315, x, 325);
				}

	if(x==320)
				{
```

```
                        gc.setStroke(new BasicStroke(1));
                        gc.drawLine(x, 0, x, h);
                    }
                    if(mainVer==0){
                        gc.setStroke(dashed);
                        gc.drawLine(x, 0, x, h);
                    }

                }

                for (int y = 0; y < h; y += gap) {
                    int subHor=y%8;
                    int mainHor=y%32;
                    if(subHor==0){
                        gc.setStroke(new BasicStroke(1));
                        gc.drawLine(315, y, 325, y);
                    }
                    if (y == 320) {
                        gc.setStroke(new BasicStroke(1));
                        gc.drawLine(0, y, w, y);
                    }
                    if(mainHor==0) {
                        gc.setStroke(dashed);
                        gc.drawLine(0, y, w, y);
                    }
                }

            }

            // Draw the grid from the precomputed image

            g2.drawImage(grid, 0, 0, new Color(132, 247, 173), null);

      }

    }
```

```
package master;


        import java.io.*;
        import java.net.*;

        /**
         * A client connects to a server and sends data back and forth.
         * If anything goes wrong with the connection,
         * for example the host is not there or is listening on a different port,
         * an exception is thrown.
         *
         * @webref
         * @brief The client class is used to create client Objects which connect to a server to
exchange data.
         * @instanceName client any variable of type Client
         * @usage Application
         */
        public class netThread implements Runnable {

          Thread thread;
          Socket socket;
          int port;
          String host;

          public InputStream input;
          public OutputStream output;

          byte buffer[] = new byte[32768];
          int bufferIndex;
          int bufferLast;

          /**
           *
           * @param parent typically use "this"
           * @param host address of the server
           * @param port port to read/write from on the server
           */
          public netThread(String host, int port)
        {
            this.host = host;
            this.port = port;

            try {
              socket = new Socket(this.host, this.port);
              input = socket.getInputStream();
              output = socket.getOutputStream();
            thread = new Thread(this);
              thread.start();
```

```
    } catch (ConnectException ce) {
      ce.printStackTrace();
      dispose();

    } catch (IOException e) {
      e.printStackTrace();
      dispose();
    }
  }

  /**
   * @param socket any object of type Socket
   * @throws IOException
   */
  public netThread(Socket socket) throws IOException {
    this.socket = socket;

    input = socket.getInputStream();
    output = socket.getOutputStream();

    thread = new Thread(this);
    thread.start();
  }


  /**
   * Disconnects from the server. Use to shut the connection when you're finished
with the Client.
   * =advanced
   * Disconnect from the server and calls disconnectEvent(Client c)
   * in the host PApplet.
   * <P/>
   * Use this to shut the connection if you're finished with it
   * while your applet is still running. Otherwise, it will be
   * automatically be shut down by the host PApplet
   * (using dispose, which is identical)
   * @webref
   * @brief Disconnects from the server
   */
  public void stop() {
    dispose();
    }
  /**
   * Disconnect from the server: internal use only.
   * <P>
   * This should only be called by the internal functions in PApplet,
   * use stop() instead from within your own applets.
   */
  public void dispose() {
```

```
    thread = null;
    try {
      // do io streams need to be closed first?
      if (input != null) input.close();
      if (output != null) output.close();

    } catch (Exception e) {
      e.printStackTrace();
    }
    input = null;
    output = null;

    try {
      if (socket != null) socket.close();

    } catch (Exception e) {
      e.printStackTrace();
    }
    socket = null;
  }


  public void run() {
    while (Thread.currentThread() == thread) {
      try {
        while ((input != null) &&
               (input.available() > 0)) {  // this will block
          synchronized (buffer) {
            if (bufferLast == buffer.length) {
              byte temp[] = new byte[bufferLast << 1];
              System.arraycopy(buffer, 0, temp, 0, bufferLast);
              buffer = temp;
            }
            buffer[bufferLast++] = (byte) input.read();
          }
        }
/*
       try {
          // uhh.. not sure what's best here.. since blocking,
          // do we need to worry about sleeping much? or is this
    // gonna try to slurp cpu away from the main applet?
          Thread.sleep(10);
        } catch (InterruptedException ex) { }
*/
      } catch (IOException e) {
        //errorMessage("run", e);
        e.printStackTrace();
      }
```

```
    }
  }


  /**
   * Return true if this client is still active and hasn't run
   * into any trouble.
   */
  public boolean active() {
    return (thread != null);
  }


  /**
   * Returns the IP address of the computer to which the Client is attached.
   * @brief Returns the IP address of the machine as a String
   * @webref
   */
  public String ip() {
    return socket.getInetAddress().getHostAddress();
  }


  /**
   * Returns the number of bytes available. When any client has bytes available from
the server, it returns the number of bytes.
   * @webref
   * @brief Returns the number of bytes in the buffer waiting to be read
   */
  public int available() {
    return (bufferLast - bufferIndex);
  }


  /**
   * Empty the buffer, removes all the data stored there.
   *
   * @webref
  * @brief Clears the buffer
   */
  public void clear() {
    bufferLast = 0;
    bufferIndex = 0;
  }


  /**
   * Returns a number between 0 and 255 for the next byte that's waiting in the buffer.
```

```
     * Returns -1 if there is no byte, although this should be avoided by first cheacking
<b>available()</b> to see if any data is available.
     *
     * @webref
     * @brief Returns a value from the buffer
     */
    public int read() {
      if (bufferIndex == bufferLast) return -1;

      synchronized (buffer) {
        int outgoing = buffer[bufferIndex++] & 0xff;
        if (bufferIndex == bufferLast) {  // rewind
          bufferIndex = 0;
          bufferLast = 0;
        }
        return outgoing;
      }
    }


    /**
     * Returns the next byte in the buffer as a char.
     * Returns -1, or 0xffff, if nothing is there.
     *
     * @webref
     * @brief Returns the next byte in the buffer as a char
     */
    public char readChar() {
      if (bufferIndex == bufferLast) return (char)(-1);
      return (char) read();
    }


    /**
     * Reads a group of bytes from the buffer.
```

```
  * The version with no parameters returns a byte array of all data in the buffer.
        * This is not efficient, but is easy to use.
        * The version with the <b>byteBuffer</b> parameter is more memory and time
efficient.
        * It grabs the data in the buffer and puts it into the byte array passed in and returns
an int value for the number of bytes read.
        * If more bytes are available than can fit into the <b>byteBuffer</b>, only those
that fit are read.
        * =advanced
        * Return a byte array of anything that's in the serial buffer.
        * Not particularly memory/speed efficient, because it creates
        * a byte array on each read, but it's easier to use than
        * readBytes(byte b[]) (see below).
        *
        * @webref
        * @brief Reads everything in the buffer
        */
       public byte[] readBytes() {
         if (bufferIndex == bufferLast) return null;

         synchronized (buffer) {
           int length = bufferLast - bufferIndex;
           byte outgoing[] = new byte[length];
           System.arraycopy(buffer, bufferIndex, outgoing, 0, length);

           bufferIndex = 0;  // rewind
           bufferLast = 0;
           return outgoing;
         }
       }


       /**
        * Grab whatever is in the serial buffer, and stuff it into a
        * byte buffer passed in by the user. This is more memory/time
        * efficient than readBytes() returning a byte[] array.
        *
        * Returns an int for how many bytes were read. If more bytes
        * are available than can fit into the byte array, only those
        * that will fit are read.
        *
        * @param bytebuffer passed in byte array to be altered
        */
       public int readBytes(byte bytebuffer[]) {
         if (bufferIndex == bufferLast) return 0;
```

```
 synchronized (buffer) {
          int length = bufferLast - bufferIndex;
          if (length > bytebuffer.length) length = bytebuffer.length;
          System.arraycopy(buffer, bufferIndex, bytebuffer, 0, length);

          bufferIndex += length;
          if (bufferIndex == bufferLast) {
            bufferIndex = 0;  // rewind
            bufferLast = 0;
          }
          return length;
        }
      }


      /**
       * Reads from the port into a buffer of bytes up to and including a particular
character.
       * If the character isn't in the buffer, 'null' is returned.
       * The version with no <b>byteBuffer</b> parameter returns a byte array of all data
up to and including the <b>interesting</b> byte.
       * This is not efficient, but is easy to use. The version with the <b>byteBuffer</b>
parameter is more memory and time efficient.
       * It grabs the data in the buffer and puts it into the byte array passed in and returns
an int value for the number of bytes read.
       * If the byte buffer is not large enough, -1 is returned and an error is printed to the
message area. If nothing is in the buffer, 0 is returned.
       *
       * @webref
       * @brief Reads from the buffer of bytes up to and including a particular character
       * @param interesting character designated to mark the end of the data
       */
      public byte[] readBytesUntil(int interesting) {
        if (bufferIndex == bufferLast) return null;
        byte what = (byte)interesting;

        synchronized (buffer) {
          int found = -1;
          for (int k = bufferIndex; k < bufferLast; k++) {
            if (buffer[k] == what) {

      found = k;
              break;
            }
          }
          if (found == -1) return null;
      int length = found - bufferIndex + 1;
          byte outgoing[] = new byte[length];
          System.arraycopy(buffer, bufferIndex, outgoing, 0, length);
```

```
    bufferIndex += length;
    if (bufferIndex == bufferLast) {
      bufferIndex = 0; // rewind
      bufferLast = 0;
    }
    return outgoing;
  }
}


/**
 * Reads from the serial port into a buffer of bytes until a
 * particular character. If the character isn't in the serial
 * buffer, then 'null' is returned.
 *
 * If outgoing[] is not big enough, then -1 is returned,
 *   and an error message is printed on the console.
 * If nothing is in the buffer, zero is returned.
 * If 'interesting' byte is not in the buffer, then 0 is returned.
 *
 * @param byteBuffer passed in byte array to be altered
 */
public int readBytesUntil(int interesting, byte byteBuffer[]) {
  if (bufferIndex == bufferLast) return 0;
  byte what = (byte)interesting;

  synchronized (buffer) {
    int found = -1;
    for (int k = bufferIndex; k < bufferLast; k++) {
      if (buffer[k] == what) {
        found = k;
        break;
      }
    }
    if (found == -1) return 0;

    int length = found - bufferIndex + 1;
    if (length > byteBuffer.length) {
      System.err.println("readBytesUntil() byte buffer is" +
                         " too small for the " + length +
                         " bytes up to and including char " + interesting);
      return -1;
  }
    //byte outgoing[] = new byte[length];
    System.arraycopy(buffer, bufferIndex, byteBuffer, 0, length);

    bufferIndex += length;
    if (bufferIndex == bufferLast) {
```

```
        bufferIndex = 0;  // rewind
        bufferLast = 0;
      }
      return length;
    }
  }


  /**
   * Returns the all the data from the buffer as a String.
   * This method assumes the incoming characters are ASCII.
   * If you want to transfer Unicode data,
   * first convert the String to a byte stream in the representation of your choice
   * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
   *
   * @webref
   * @brief Returns the buffer as a String
   */
  public String readString() {
    if (bufferIndex == bufferLast) return null;
    return new String(readBytes());
  }


  /**
   * Combination of <b>readBytesUntil()</b> and <b>readString()</b>. Returns
<b>null</b> if it doesn't find what you're looking for.
   * =advanced
   * <p/>
   * If you want to move Unicode data, you can first convert the
   * String to a byte stream in the representation of your choice
   * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
   *
   * @webref
   * @brief Returns the buffer as a String up to and including a particular character
   * @param interesting character designated to mark the end of the data
   */
  public String readStringUntil(int interesting) {
    byte b[] = readBytesUntil(interesting);
    if (b == null) return null;
  return new String(b);
  }

  /**
   * Writes data to a server specified when constructing the client.
   *
   * @webref
   * @brief       Writes bytes, chars, ints, bytes[], Strings
   * @param data data to write
```

```
   */
  public void write(int data) {  // will also cover char
    try {
      output.write(data & 0xff);  // for good measure do the &
      output.flush();   // hmm, not sure if a good idea

    } catch (Exception e) { // null pointer or serial port dead
      //errorMessage("write", e);
      //e.printStackTrace();
      //dispose();
      //disconnect(e);
      e.printStackTrace();
      stop();
    }
  }


  public void write(byte data[]) {
    try {
      output.write(data);
      output.flush();   // hmm, not sure if a good idea

    } catch (Exception e) { // null pointer or serial port dead
      //errorMessage("write", e);
      //e.printStackTrace();
      //disconnect(e);
      e.printStackTrace();
      stop();
    }
  }


  /**
   * Write a String to the output. Note that this doesn't account
   * for Unicode (two bytes per char), nor will it send UTF8
   * characters.. It assumes that you mean to send a byte buffer
   * (most often the case for networking and serial i/o) and
 * will only use the bottom 8 bits of each char in the string.
   * (Meaning that internally it uses String.getBytes)
   *
   * If you want to move Unicode data, you can first convert the
   * String to a byte stream in the representation of your choice
   * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
   */
  public void write(String data) {
    write(data.getBytes());
  }
```

```
 /**
  * Handle disconnect due to an Exception being thrown.
  */
 /*
   protected void disconnect(Exception e) {
   dispose();
   if (e != null) {
   e.printStackTrace();
    }
    }
 */


 /**
  * General error reporting, all corraled here just in case
  * I think of something slightly more intelligent to do.
  */
 //public void errorMessage(String where, Exception e) {
 //parent.die("Error inside Client." + where + "()", e);
 //e.printStackTrace(System.err);
 //}
}
```

```
package master;
	import java.awt.BorderLayout;
	import java.awt.Color;
	import java.awt.Component;
	import java.awt.Dimension;
	import java.awt.EventQueue;
	import java.awt.FlowLayout;
	import java.awt.Graphics;
	import java.awt.GridBagConstraints;
	import java.awt.GridBagLayout;
	import java.awt.GridLayout;
	import java.awt.Insets;
	import java.awt.SystemColor;
	import java.awt.event.ActionEvent;
	import java.awt.event.ActionListener;
	import java.awt.event.ItemEvent;
	import java.awt.event.ItemListener;
	import java.awt.event.KeyAdapter;
	import java.awt.event.KeyEvent;
	import java.awt.event.MouseAdapter;
	import java.awt.event.MouseEvent;
	import javax.swing.Box;
	import javax.swing.BoxLayout;
	import javax.swing.ButtonGroup;
	import javax.swing.JButton;
	import javax.swing.JComboBox;
	import javax.swing.JComponent;
	import javax.swing.JFrame;
	import javax.swing.JLabel;
	import javax.swing.JLayeredPane;
	import javax.swing.JPanel;
	import javax.swing.JRadioButton;
	import javax.swing.JSeparator;
	import javax.swing.JSlider;
	import javax.swing.JSpinner;
	import javax.swing.JTextField;
	import javax.swing.SpinnerNumberModel;
	import javax.swing.SpringLayout;
	import javax.swing.SwingConstants;
	import javax.swing.border.BevelBorder;
	import javax.swing.border.EtchedBorder;
	import javax.swing.border.TitledBorder;
	import javax.swing.event.ChangeEvent;
	import javax.swing.event.ChangeListener;
	import org.jdesktop.layout.GroupLayout;
	import org.jdesktop.layout.LayoutStyle;

	public class scopeMainGui extends JFrame {
```

```
        private ButtonGroup buttonGroup = new ButtonGroup();
        private SpringLayout springLayout_2;
        private SpringLayout springLayout;
        private final JPanel panel = new JPanel();
        private final JButton button = new JButton();
        private final JButton durdurButton = new JButton();
        private final JPanel panel_1 = new JPanel();
        private final JSlider slider = new JSlider(SwingConstants.VERTICAL,0,100,0);


        private final JLabel pwmLabel = new
JLabel("<html><center>PWM<br>ORANI</center></html>");
        private final JLabel label_speed = new JLabel();
        private final JLabel label_cur = new JLabel();
        private final JLabel label_volt = new JLabel();
        private final Component component_1 = Box.createHorizontalStrut(5);
        private Graphics g;
        private final JPanel panel_3 = new JPanel();
        private final drawGrid dg=new drawGrid();
        nbClient nbc;
        private final JButton button_1 = new JButton();
        private final JPanel panel_4 = new JPanel();
        private final JPanel panel_2 = new JPanel();
        private final JSpinner voltSpinner = new JSpinner();

        private String[] voltValues = { "10","5", "2", "1", "0.5", "0.2", "0.1",
"0.05","0.02","0.01","0.001"};
        private final JComboBox voltComboBox = new JComboBox(voltValues);

        private final JPanel panel_6 = new JPanel();
        private final JSpinner curSpinner = new JSpinner();
        private String[] curValues = { "10","5", "2", "1", "0.5", "0.2", "0.1",
"0.05","0.02","0.01","0.001"};
        private final JComboBox curComboBox = new JComboBox(curValues);

        private String[] timeValues = { "100","50", "30"};
        private final JComboBox comboBox_timeDiv = new JComboBox(timeValues);
        private final JSeparator separator = new JSeparator();
        private final JSeparator separator_1 = new JSeparator();

       private final JButton dondurButton = new JButton();
        private final JButton button_2 = new JButton();
```

```
private final JSeparator separator_2 = new JSeparator();
	private final JPanel panel_5 = new JPanel();
	private final JRadioButton radioButton_singleFrame = new JRadioButton();
	private final JRadioButton radioButton_1 = new JRadioButton();
	private final JTextField textField_Rs = new JTextField();
	/**
	 * Launch the application
	 * @param args
	 */
	public static void main(String args[]) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					scopeMainGui frame = new scopeMainGui();

					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame
	 */
	public scopeMainGui() {
		super();
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		try {
			jbInit();
		} catch (Throwable e) {
			e.printStackTrace();
		}
		//
	}
	private void jbInit() throws Exception {

		getContentPane().setFocusCycleRoot(true);
		getContentPane().setLayout(null);
		setSize(860, 771);
		panel.add(button);
		panel.setBounds(90, 0, 640, 36);
		panel.setLayout(new FlowLayout());
		getContentPane().add(panel);
```

```
button.addActionListener(new ButtonActionListener());
		button.setText("Deneyi Başlat");
		panel.add(button_1);
		button_1.addActionListener(new Button_1ActionListener());
		button_1.setText("PWM Gönder");
		panel.add(durdurButton);
		durdurButton.addActionListener(new DurdurButtonActionListener());
		durdurButton.setText("Motoru Durdur");
		panel.add(dondurButton);
		dondurButton.addActionListener(new DondurButtonActionListener());
		dondurButton.setText("Çizimi Durdur");
		panel.add(button_2);
		button_2.addActionListener(new Button_2ActionListener());
		button_2.setText("Çizime Devam Et");

		final GridBagLayout gridBagLayout = new GridBagLayout();
		gridBagLayout.rowHeights = new int[] {0,0,0};
		panel_1.setLayout(gridBagLayout);
		panel_1.setBounds(0, 42, 84, 600);
		getContentPane().add(panel_1);
		panel_1.setPreferredSize(new Dimension(100, 300));
		panel_1.setBorder(new EtchedBorder(BevelBorder.LOWERED));
		panel_1.setMinimumSize(new Dimension(100, 300));

		final GridBagConstraints gridBagConstraints_1 = new
GridBagConstraints();
		gridBagConstraints_1.fill = GridBagConstraints.VERTICAL;
		gridBagConstraints_1.insets = new Insets(0, 0, 0, 0);
		gridBagConstraints_1.gridx = 0;
		gridBagConstraints_1.gridy = 1;

		component_1.hashCode();
		final GridBagConstraints gridBagConstraints_2 = new
GridBagConstraints();
		gridBagConstraints_2.gridx = 0;
		gridBagConstraints_2.gridy = 0;
		panel_1.add(component_1, gridBagConstraints_2);
		panel_1.add(pwmLabel, gridBagConstraints_1);
		pwmLabel.setForeground(Color.RED);
		pwmLabel.setHorizontalTextPosition(SwingConstants.CENTER);
		pwmLabel.setHorizontalAlignment(SwingConstants.CENTER);

		slider.setMajorTickSpacing(10);
		slider.addChangeListener(new SliderChangeListener());
		slider.setName("slider");
```

```
slider.setPaintLabels(true);
                slider.setPaintTicks(true);
                slider.setOrientation(SwingConstants.VERTICAL);
                slider.setMinorTickSpacing(1);

                final GridBagConstraints gridBagConstraints = new GridBagConstraints();
                gridBagConstraints.fill = GridBagConstraints.VERTICAL;
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.ipady = 520;
                gridBagConstraints.insets = new Insets(0, 2, 0, 0);
                panel_1.add(slider, gridBagConstraints);
                slider.setPreferredSize(new Dimension(50, 350));


                getContentPane().add(panel_3);

                panel_3.setBackground(Color.BLACK);
                panel_3.setBounds(90, 42, 640, 640);

                getContentPane().add(panel_4);
                panel_4.setLayout(null);
                panel_4.setBounds(90, 688, 640, 49);

                label_cur.setBorder(new TitledBorder(null, "Akım",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
                label_cur.setBounds(160, 0, 77, 42);
                panel_4.add(label_cur);
                label_cur.setHorizontalAlignment(SwingConstants.RIGHT);

                label_speed.setBorder(new TitledBorder(null, "Hız",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
                label_speed.setBounds(243, 0, 77, 42);
                panel_4.add(label_speed);
                label_speed.setHorizontalAlignment(SwingConstants.RIGHT);
                label_speed.setHorizontalTextPosition(SwingConstants.CENTER);

                label_volt.setBorder(new TitledBorder(null, "Gerilim",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
                label_volt.setBounds(77, 0, 77, 42);
                panel_4.add(label_volt);
                label_volt.setHorizontalAlignment(SwingConstants.RIGHT);
        panel_4.add(textField_Rs);
                textField_Rs.setText("2.2");
                textField_Rs.addKeyListener(new TextField_RsKeyListener());
```

```
                textField_Rs.setBorder(new TitledBorder(null, "Rö",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
                textField_Rs.setBounds(10, 0, 57, 38);

                getContentPane().add(panel_2);
                springLayout = new SpringLayout();
                panel_2.setLayout(springLayout);
                panel_2.setBorder(new TitledBorder(null, "Gerilim Kanalı",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
                panel_2.setBounds(736, 42, 113, 129);
                panel_2.add(voltSpinner);
                voltSpinner.addChangeListener(new VoltSpinnerChangeListener());
                voltSpinner.setBorder(new TitledBorder(null, "Dikey Poz.",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
SystemColor.textHighlight));
                springLayout.putConstraint(SpringLayout.SOUTH, voltSpinner, 45,
SpringLayout.NORTH, panel_2);
                springLayout.putConstraint(SpringLayout.NORTH, voltSpinner, 5,
SpringLayout.NORTH, panel_2);
                springLayout.putConstraint(SpringLayout.EAST, voltSpinner, 95,
SpringLayout.WEST, panel_2);
                springLayout.putConstraint(SpringLayout.WEST, voltSpinner, 5,
SpringLayout.WEST, panel_2);
                panel_2.add(voltComboBox);
                voltComboBox.setSelectedIndex(3);
                springLayout.putConstraint(SpringLayout.EAST, voltComboBox, 0,
SpringLayout.EAST, voltSpinner);
                springLayout.putConstraint(SpringLayout.WEST, voltComboBox, 0,
SpringLayout.WEST, voltSpinner);
                voltComboBox.setDoubleBuffered(true);
                voltComboBox.addActionListener(new VoltComboBoxActionListener());

                voltComboBox.setEditable(true);
                voltComboBox.setBorder(new TitledBorder(null, "Girilim/Bölme",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
SystemColor.textHighlight));
```

```
springLayout.putConstraint(SpringLayout.SOUTH, voltComboBox, 105,
SpringLayout.NORTH, panel_2);
		springLayout.putConstraint(SpringLayout.NORTH, voltComboBox, 60,
SpringLayout.NORTH, panel_2);

		getContentPane().add(panel_6);
		panel_6.setBorder(new TitledBorder(null, "Akım Kanalı",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
Color.BLACK));
		final SpringLayout springLayout_1 = new SpringLayout();
		panel_6.setLayout(springLayout_1);
		panel_6.setBounds(736, 181, 113, 129);

		panel_6.add(curSpinner);
		curSpinner.addChangeListener(new CurSpinnerChangeListener());
		curSpinner.setBorder(new TitledBorder(null, "Dikey Poz.",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
Color.RED));
		springLayout_1.putConstraint(SpringLayout.SOUTH, curSpinner, 45,
SpringLayout.NORTH, panel_6);
		springLayout_1.putConstraint(SpringLayout.NORTH, curSpinner, 5,
SpringLayout.NORTH, panel_6);
		springLayout_1.putConstraint(SpringLayout.EAST, curSpinner, 95,
SpringLayout.WEST, panel_6);
		springLayout_1.putConstraint(SpringLayout.WEST, curSpinner, 5,
SpringLayout.WEST, panel_6);

		panel_6.add(curComboBox);
		curComboBox.setSelectedIndex(3);
		curComboBox.addActionListener(new CurComboBoxActionListener());
		curComboBox.setBorder(new TitledBorder(null, "Akım/Bölme",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
Color.RED));
		springLayout_1.putConstraint(SpringLayout.SOUTH, curComboBox, 105,
SpringLayout.NORTH, panel_6);
		springLayout_1.putConstraint(SpringLayout.NORTH, curComboBox, 60,
SpringLayout.NORTH, panel_6);
		springLayout_1.putConstraint(SpringLayout.EAST, curComboBox, 0,
SpringLayout.EAST, curSpinner);
		springLayout_1.putConstraint(SpringLayout.WEST, curComboBox, 0,
SpringLayout.WEST, curSpinner);
		curComboBox.setEditable(true);
		curComboBox.setDoubleBuffered(true);

		getContentPane().add(comboBox_timeDiv);
```

```
comboBox_timeDiv.addActionListener(new ComboBox_timeDivActionListener());
		comboBox_timeDiv.setBorder(new TitledBorder(null, "Time/Div (mS)",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
		comboBox_timeDiv.setBounds(736, 323, 113, 51);

		getContentPane().add(separator);
		separator.setBackground(Color.BLACK);
		separator.setBounds(736, 316, 113, 2);

		getContentPane().add(separator_1);
		separator_1.setBackground(Color.BLACK);
		separator_1.setBounds(736, 176, 113, 2);

		getContentPane().add(separator_2);
		separator_2.setBackground(Color.BLACK);
		separator_2.setBounds(736, 380, 113, 8);

		getContentPane().add(panel_5);
		springLayout_2 = new SpringLayout();
		panel_5.setLayout(springLayout_2);
		panel_5.setBorder(new TitledBorder(null, "Çizim Modu",
TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.DEFAULT_POSITION, null,
null));
		panel_5.setBounds(736, 385, 113, 82);

		panel_5.add(radioButton_singleFrame);
		radioButton_singleFrame.addActionListener(new
RadioButton_singleFrameActionListener());
		buttonGroup.add(radioButton_singleFrame);
		springLayout_2.putConstraint(SpringLayout.SOUTH,
radioButton_singleFrame, 29, SpringLayout.NORTH, panel_5);
		springLayout_2.putConstraint(SpringLayout.NORTH,
radioButton_singleFrame, 5, SpringLayout.NORTH, panel_5);
		springLayout_2.putConstraint(SpringLayout.EAST,
radioButton_singleFrame, 101, SpringLayout.WEST, panel_5);
		springLayout_2.putConstraint(SpringLayout.WEST,
radioButton_singleFrame, 5, SpringLayout.WEST, panel_5);
		radioButton_singleFrame.setText("Tek Çerçeve");

		panel_5.add(radioButton_1);
		radioButton_1.addActionListener(new RadioButton_1ActionListener());
		buttonGroup.add(radioButton_1);
		springLayout_2.putConstraint(SpringLayout.EAST, radioButton_1, 65,
SpringLayout.WEST, radioButton_singleFrame);
```

```
springLayout_2.putConstraint(SpringLayout.WEST, radioButton_1, 0,
SpringLayout.WEST, radioButton_singleFrame);
		springLayout_2.putConstraint(SpringLayout.SOUTH, radioButton_1, 55,
SpringLayout.NORTH, panel_5);
		springLayout_2.putConstraint(SpringLayout.NORTH, radioButton_1, 31,
SpringLayout.NORTH, panel_5);
		radioButton_1.setText("Sürekli");




	}
	private class ButtonActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			button_actionPerformed(e);
		}
	}
	private class SliderChangeListener implements ChangeListener {
		public void stateChanged(ChangeEvent e) {
			slider_stateChanged(e);
		}
	}
	private class DurdurButtonActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			durdurButton_actionPerformed(e);
		}
	}
	private class Button_1ActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			button_1_actionPerformed(e);
		}
	}

	private class VoltComboBoxActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			voltComboBox_actionPerformed(e);
		}
	}
	private class VoltSpinnerChangeListener implements ChangeListener {
		public void stateChanged(ChangeEvent e) {
			voltSpinner_stateChanged(e);
		}
	}
	private class CurSpinnerChangeListener implements ChangeListener {
		public void stateChanged(ChangeEvent e) {
	curSpinner_stateChanged(e);
		}
```

```
	}
	private class CurComboBoxActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			curComboBox_actionPerformed(e);
		}
	}

	private class ComboBox_timeDivActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			comboBox_timeDiv_actionPerformed(e);
		}
	}
	private class DondurButtonActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			dondurButton_actionPerformed(e);
		}
	}
	private class Button_2ActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			button_2_actionPerformed(e);
		}
	}

	private class RadioButton_singleFrameActionListener implements ActionListener
{
		public void actionPerformed(ActionEvent e) {
			radioButton_singleFrame_actionPerformed(e);
		}
	}
	private class RadioButton_1ActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			radioButton_1_actionPerformed(e);
		}
	}
	private class TextField_RsKeyListener extends KeyAdapter {
		public void keyTyped(KeyEvent e) {
			textField_Rs_keyTyped(e);
		}
		public void keyReleased(KeyEvent e) {
			textField_Rs_keyReleased(e);
		}
	}
```

```
protected void button_actionPerformed(ActionEvent e) {
		nbc =new nbClient(panel_3,label_volt,label_cur,label_speed);
		int vs=Integer.parseInt(voltComboBox.getSelectedItem().toString());



		nbc.start();
	  nbc.startMotor(0);
		button.setEnabled(false);

	  }
	  protected void slider_stateChanged(ChangeEvent e) {




	  }
	  protected void durdurButton_actionPerformed(ActionEvent e) {
		if (radioButton_singleFrame.isSelected()){
			nbc.setContMode(false);

		}else if(!radioButton_singleFrame.isSelected()){
			nbc.setContMode(true);
		}
				nbc.startMotor(0);




	  }
	  protected void button_1_actionPerformed(ActionEvent e) {
	 if (radioButton_singleFrame.isSelected()){
	  nbc.setContMode(false);

	 }else if(!radioButton_singleFrame.isSelected()){
	  nbc.setContMode(true);
	 }
		nbc.startMotor(Math.round(slider.getValue()*255/100));



	  }

	  protected void voltComboBox_actionPerformed(ActionEvent e) {
```

```
nbc.setVoltScale(Double.parseDouble(voltComboBox.getSelectedItem().toString()));
	}
	protected void voltSpinner_stateChanged(ChangeEvent e) {
		nbc.setVoltPosition(Integer.parseInt(voltSpinner.getValue().toString()));
	}
	protected void curSpinner_stateChanged(ChangeEvent e) {
		nbc.setCurPosition(Integer.parseInt(curSpinner.getValue().toString()));
	}
	protected void curComboBox_actionPerformed(ActionEvent e) {

	nbc.setCurScale(Double.parseDouble(curComboBox.getSelectedItem().toString()))
;
	}

	protected void comboBox_timeDiv_actionPerformed(ActionEvent e) {

	nbc.setTimeStep(Integer.parseInt(comboBox_timeDiv.getSelectedItem().toString())
);
	}
	protected void dondurButton_actionPerformed(ActionEvent e) {
		nbc.setDrawStatus(false);
	}
	protected void button_2_actionPerformed(ActionEvent e) {
		nbc.setDrawStatus(true);
	}

	protected void radioButton_singleFrame_actionPerformed(ActionEvent e) {
		//nbc.setContMode(false);
	}
	protected void radioButton_1_actionPerformed(ActionEvent e) {
		nbc.setContMode(true);
	}
	protected void textField_Rs_keyTyped(KeyEvent e) {

	}
	protected void textField_Rs_keyReleased(KeyEvent e) {
		int key=e.getKeyCode();
		if(key==KeyEvent.VK_ENTER){

	nbc.setRsVal(Double.parseDouble(textField_Rs.getText().toString()));
		}		}

	}
```

ÖZGEÇMİŞ

Ahmet AKTOĞAN 03.01.1973 Trabzon doğumludur. Lise öğrenimini Gümüşhane Lisesinde tamamladıktan sonra Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği bölümünde lisans öğrenimini 1997 yılında tamamlamıştır. 2004 yılına kadar serbest çalışmıştır. 2004-2011 yılları arasında KTÜ Beşikdüzü Meslek Yüksekokulu Elektronik Haberleşme Programında öğretim görenlisi olarak çalışmış 2011 yılndan buyana da Gümüşhane Üniversitesi Gümüşhane Meslek Yüksekokulu Mekatronik Programında öğretim görevlisi olarak çalışmaktadır. Yabancı dil olarak iyi derecede İngilizce bilmektedir. Evli ve iki çocuk babasıdır.