T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

PID KONTROLLÜ TOP-ÇUBUK SİSTEMİNİN TASARIMI VE KONTROLÜ ÜZERİNE BİR ARAŞTIRMA

Sıtkı KOCAOĞLU

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ ANA BİLİM DALI

Yrd.Doç.Dr. Hilmi KUŞÇU

EDİRNE 2013

PID KONTROLLÜ TOP-ÇUBUK SİSTEMİNİN

TASARIMI VE KONTROLÜ ÜZERİNE BİR ARAŞTIRMA

SITKI KOCAOĞLU

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ ANA BİLİM DALI

2013

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof. Dr. Mustafa ÖZCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

Prof. Dr. Taner TIMARCI
Anabilim Dalı Başkanı

Bu tez tarafımızca okunmuş, kapsamı ve niteliği açısından bir Doktora tezi olarak kabul edilmiştir.

Yrd.Doç.Dr.Hilmi KUŞÇU
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Makine Mühendisliği Anabilim Dalında bir yüksek lisans tezi olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad):	İmza
Doç.Dr. Yılmaz ÇAN	
Yrd.Doç.Dr.Hilmi KUŞÇU	
Yrd.Doç.Dr. Hayrettin TOYLAN	

Tarih: 15 /08 / 2013

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI

DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.

15/08/2013

Sıtkı KOCAOĞLU

Yüksek Lisans Tezi
PID Kontrollü Top-Çubuk Sisteminin Tasarımı ve Kontrolü Üzerine Bir Araştırma
T.Ü. Fen Bilimleri Enstitüsü
Makine Mühendisliği Anabilim Dalı

ÖZET

Top- Çubuk sisteminin kontrolü havacılık ve uzay alanlarında, uçuş, iniş yada türbülans durumlarında aracın kontrolüne çok benzer olması sebebiyle önemlidir. Bu alanlarda gerçek deneme yanılma uygulamalarının mümkün olmaması Top- Çubuk sisteminin kontrol laboratuvarlarında kullanılmasını gerekli hale getirmiştir. Sistem topun çubuk üzerinde seçilecek herhangi bir noktada tutulmasını amaçlamaktadır. Topun mevcut durumu değişik sensörler kullanılarak ölçülür ve çeşitli motorlar vasıtasıyla çubuk açısı değiştirilerek topun yer değiştirmesi sağlanır. Bu sistemin matematiksel modeli lineer değildir. Değişik kontrol algoritmaları uygulanarak topun istenilen noktaya yerleşme işlemi uygun şekilde sağlanmaya çalışılır. Burada yerleşme zamanı, maksimum aşma ve kalıcı durum hatası gibi değişkenler optimum halde tutulmaya çalışılır. Bu çalışmada PID kontrol yöntemi uygulanmış ve sistem bir deney seti olarak hazırlanmıştır. Amaç kontrol laboratuvarlarında öğrencilere PID katsayılarının etkilerini uygulamalı olarak anlatmaktır.

Yıl **: 2013**

Sayfa Sayısı **: 107**

Anahtar Kelimeler **: Top-Çubuk, PID, Kontrol, Mikrodenetleyici**

M. Sc. Thesis
A Research About Design and Control of Ball and Beam System with PID Algorithm
Trakya University Institute of Natural Sciences
Mechanical Engineering Department

ABSTRACT

The control of The Ball and Beam system is very important in the aviation and space fields because of its similarity to the control of aircraft during the flight, landing and turbulent,. Since real experiments are not possible in these areas, using Ball and Beam systems in the control laboratories has become necessary. The system aims to keep the ball on the beam in desired position. Current position of the ball is measured by using various sensors and the location of the ball is changed by changing the angle of beam by using various motors. The mathematical model of this system is not linear. By using several control algorithms, system tries to move the ball to the desired position appropriately. Here system tries to set the rising time, the overshoot and offset in optimum level. In this study the PID control method is applied and system is fixed as an experimental set. The aim is lecturing the effects of PID constants to students in control laboratories.

Year : **2013**

Number of Pages : **107**

Keywords : **Ball and Beam, PID, Control, Microprocessor**

TEŞEKKÜR

Yüksek lisans eğitimim süresince desteğini benden esirgemeyen, kıymetli fikir ve tecrübeleri doğrultusunda bana yol gösteren ve ivme kazandıran değerli hocam Yrd.Doç.Dr. Hilmi KUŞÇU'ya teşekkür ederim.

Bu süreçte benimle birlikte adeta göreviymiş gibi çalışan arkadaşım Ercan KURAK'a ve çalışmalarıma katkı sağlayıp bana sabır gösteren tüm mesai arkadaşlarıma teşekkürü bir borç bilirim.

Bu zor süreçte her zaman yanımda olan eşim Burcu KOCAOĞLU'na ve varlığıyla bana güç veren sevgili oğlum Atakan KOCAOĞLU'na gönülden teşekkür ederim.

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
İÇİNDEKİLER iv
SİMGELER vii
KISALTMALAR ix
ŞEKİLLER x
TABLOLAR xii
BÖLÜM 1 1
GİRİŞ 1
BÖLÜM 2 3
LİTERATÜR TARAMASI 3
BÖLÜM 3 5
PID KONTROLÖR TASARIMI 5
3.1. Oransal (P) Kontrolör 5
3.2. Oransal-İntegral (PI) Kontrolör 6
3.2.1 PI Kontrolörün Zaman Tanım Bölgesinde İncelenmesi 8
3.2.2. PI Kontrolörün Frekans Tanım Bölgesinde İncelenmesi 8
3.3. PD Kontrolör 9
3.4. PID Kontrolör 12
3.5. Ziegler- Nichols Kuralları 12
BÖLÜM 4 15
DİJİTAL PID KONTROLÖR TASARIMI 15
4.1. Dijital PD Kontrolör 16
4.2. Dijital PI Kontrolör 17
4.3. Dijital PID Kontrolör 17
BÖLÜM 5 18
STEP MOTORLAR 18

5.1. Step Motorların Genel Özellikleri 18
5.2. Step Motorların Karakteristik Özellikleri 20
5.2.1. Statik Karakteristikler 20
5.2.1.1. Sargı Direnci (R) 20
5.2.1.2. Sargı İndüktansı (L) 20
5.2.1.3. Rotor Atalet Momenti (JM) 20
5.2.1.4. Adım Açısı (θa) 20
5.2.1.5. Temel Adım Açısı (θt) 20
5.2.1.6. Nominal Sargı Akımı (IR) 21
5.2.1.7. Histerezis Hatası (Δθh) 21
5.2.2. Dinamik Karakteristikler 21
5.2.2.1. Tutma Momenti (T_d) 21
5.2.2.2. Darbe Oranı (f_p) 21
5.2.2.3. Maksimum Başlama Frekansı (f_s) 21
5.2.2.4. Başlama Torku (T_s) 21
5.2.2.5. Çekme Torku (T_o) 22
5.2.2.6. Tek Adım Tepkisi ve Rezonansı 22
5.2.3. Step Motor Çalışma Eğrileri 23
5.2.3.1. Darbe Oranı ve Eylemsizlik Momenti Karakteristiği 23
5.2.3.2. Darbe Oranı- Tork Karakteristiği 23
5.2.3.3. Moment- Akım Karakteristiği 24
5.3. Hibrit Step Motorlar 25
5.3.1. Hibrit Step Motorların Genel Özellikleri 25
5.3.2. Hibrit Step Motorların Çalışma Karakteristikleri 27
5.3.2.1. Açısal Pozisyon 27
5.3.2.2. Tork ve Hız 27
5.3.3.3. Rezonans 29
5.3.3.4. Mikro Adımlama (Microstepping) 29
BÖLÜM 6 31
SİSTEMİN MODELLENMESİ VE MALZEME SEÇİMİ 31
6.1. Serbest Cisim Diyagramı ve Transfer Fonksiyonu 31
6.2. Mekanik Dizayn ve Malzeme Seçimi 33

6.3. Elektronik Dizayn ve Malzeme Seçimi....34
6.3.1. Mikrodenetleyici....36
6.3.2. Sensörler ve Konum Belirleme....38
6.3.3. Step Motor....42
6.3.4. LCD Ekran ve Kontrol Paneli....43
6.4. Yazılım....44
6.5. Blok Diyagram....45
6.6. Sistemin Çalışması....47
BÖLÜM 7....53
SONUÇLAR VE ÖNERİLER....53
EKLER....56
EK-A MİKRODENETLEYİCİ CCS-C KODU....56
KAYNAKLAR....74
ÖZGEÇMİŞ....77

SİMGELER

s	Saniye
(t)	Zaman Domeni
(s)	Frekans Domeni
e	Hata
Gc	Kontrolör
Gp	Kontrol Edilen Sistem
ω_n	Doğal Frekans
$\boldsymbol{\zeta}$	Amortisman Oranı
ω_g	Faz Sınırı Payı
J	Atalet Momenti
M_r	Rezonans Tepesi
M_p	Maksimum Aşma
t_r	Yükselme Zamanı
t_o	Yerleşme Zamanı
u	Kontrol İşareti
y	Kontrol Edilen Değişken, Çıkış Vektörü
R	Cevap Eğrisinin En Yüksek Eğimi, Direnç
L	Cevap Eğrisinin Zaman Eksenini Kestiği Nokta, Endüktans
K_{pu}	Ziegler-Nichols İçin Osilasyon Yapan Minimum Kazanç
P_u	Ziegler-Nichols İçin Osilasyon Yapan Minimum Periyot
K_v	Dijital PID Kontrolör İçin Hata Sabiti
θ_a	Adım Açısı

θ_t	Gerçek Adım Açısı
$\Delta\theta h$	Histerezis Hatası
f_p	Darbe Oranı
f_s	Maksimum Başlama Frekansı, Stator Akıları Arasındaki Uzaklık
f_r	Rotor Pozisyonu
f_{SL}	Yüklü durumdayken başlama frekansı
T_s	Başlama Torku
T_o	Çekme Torku
T_d	Tutma Torku
J_L	Yükün Atalet Momenti
J_M	Rotorun Eylemsizlik Momenti
λ	Tam Adım Modunda Toplam Adım Açısı
v	Hız
T	Motor Tarafından Üretilen Tork
F	Kuvvet
g	Yerçekimi İvmesi
m	Kütle
α	Açısal Yer Değiştirme
τ	Moment
Ω	Ohm

KISALTMALAR

ADC	Analog- Dijital Çevirici
B.W.	Bant Genişliği
dB	Desibel
DC	Doğru Akım
H	Henry
I	İntegral Kontrol
K_d	Türevsel Parametre Katsayısı
K_i	İntegral Parametre Katsayısı
K_p	Oransal Parametre Katsayısı
N	Newton
PID	Oransal İntegral Türevsel Kontrol
P	Oransal Kontrol
PD	Oransal Türevsel Kontrol
PI	Oransal İntegral Kontrol
rad	Radyan
SA	Adım Açısı
SPR	Dönüş Başına Adım Sayısı

ŞEKİLLER

Şekil 3.1 Oransal Kontrolörün Birim Basamak Cevabı ve Blok Diyagramı 6
Şekil 3.2 I ve PI Kontrolörlerin Birim Basamak Cevabı ve Blok Diyagramları 7
Şekil 3.3 PI Kontrolörlü Kontrol Sistemi Blok Diyagramı 8
Şekil 3.4 PD Kontrolörlü Kontrol Sistemi Blok Diyagramı 10
Şekil 3.5 PD ve PID Kontrolörlerin Birim Basamak Cevapları ve Blok Diyagramları.. 11
Şekil 3.6 Ziegler- Nichols Teoremi Açık Çevrim Cevabı..... 13
Şekil 5.1 Step Motorun Tek Adım Tepkisi ve Rezonansı..... 22
Şekil 5.2 Step Motorların Darbe Oranı- Tork Karakteristiği 23
Şekil 5.3 Step Motorların Akım- Moment Grafiği 24
Şekil 5.4 Hibrit Step Motorların İç Yapısı 26
Şekil 5.5 a) Hibrit Step Motor Rotoru..... 26
Şekil 5.6 b) Hibrit Step Motor Statoru 26
Şekil 5.7 Hibrit Step Motorların Akıma Bağlı Hız- Tork Grafiği 28
Şekil 5.8 Hibrit Step Motorların Gerilime Bağlı Hız- Tork Karakteristiği..... 29
Şekil 5.9 Tam Adım ile Mikro Adım Tekniklerinin Açısal Pozisyon, Tork ve Hız Bakımından Karşılaştırılması..... 30
Şekil 6.1 Sistem Bileşenleri Çizimi 31
Şekil 6.2 Cihazın Görünüşü 34
Şekil 6.3 Proteus ISIS Programı İle Devre Modellenmesi..... 35
Şekil 6.4 Proteus ARES Programı İle Baskı Devre 35
Şekil 6.5 PIC18F4550 Pin Diyagramı..... 37
Şekil 6.6 DS1302 Entegresi Bağlantı Şeması 37
Şekil 6.7 Sharp GP2Y0AO Analog Kızılötesi Mesafe Sensörü 38

Şekil 6.8 Dijital Kızılötesi Sensör 39

Şekil 6.9 1. Sensöre Ait Top Konumu- Çıkış Gerilimi Grafiği 40

Şekil 6.10 2. Sensöre Ait Topun Konumu- Çıkış Gerilimi Grafiği 40

Şekil 6.11 GP2Y0A0 Topun Konumu- Çıkış Gerilimi Grafiği 41

Şekil 6.12 Nema 23 Step Motor 42

Şekil 6.13 Leadshine M542 Step Motor Sürücüsü 43

Şekil 6.14 Kontrol Paneli 44

Şekil 6.15 Sistemin Blok Diyagramı 45

Şekil 6.16 Sinyal Akış Grafiği 46

TABLOLAR

Tablo 3.1Ziegler- Nichols Katsayılarının Hesaplanması 14
Tablo 4.1 Step Motorların Diğer Motorlara Göre Avantaj ve Dezavantajları 19
Tablo 6.1 Sensörlerden Okunan Değerler 39
Tablo 6.2 Sistem Parametreleri 48
Tablo 6.3 Deney Seti Üzerinde Yapılan P Kontrolör Denemeleri 48
Tablo 6.4 Deney Seti Üzerinde Yapılan PD Kontrolör Denemeleri 49
Tablo 6.5 Deney Seti Üzerinde Yapılan PI Kontrolör Denemeleri 50
Tablo 6.6 Deney Seti Üzerinde Yapılan PID Kontrolör Denemeleri 51

BÖLÜM 1

GİRİŞ

Bu çalışmada bir çubuk üzerinde serbestçe hareket eden topun konum kontrolünün yapılması amaçlanmıştır. Çubuğun yatayla yapmış olduğu açı kendisini kontrol eden motorun hareketine bağlıdır. Böylece kullanıcı arabirimi yardımıyla topun seçilen konuma getirilmesi sensörlerle desteklenen kapalı çevrim bir kontrol sistemi yardımıyla step motorun konum kontrolü yapılarak sağlanmıştır. Bir deney seti olarak dizayn edilmiş olan sistem aslında otomatik kontrolün popüler modellerinden biri olan top-çubuk sistemi için yeni bir materyal seçim ve kontrol yaklaşımı getirmektedir.

Kararlı olmayan sistemlerin kontrolü birçok kontrol problemi açısından çok önemlidir. Bu tür sistemlerin çoğu tehlikeli olduğundan ancak laboratuvar ortamında modelleme yöntemiyle üzerinde çalışma yapılabilir durumdadır. Ekzotermik kimyasal reaksiyonların açık döngü sıcaklık kontrolünde, güç üretiminde kararsız manyetik alan kontrolünde ve havacılık ve uzay araçlarında dikey konum kontrolünde benzer durumun olduğu görülmektedir.

Bu projede alüminyum L profil bir çubuk, bir step motor, bir pinpon topu, topun konumunu algılamak üzere iki adet kızılötesi mesafe sensörü ve bir adet dijital kızılötesi sensör kullanılmıştır. Burada ölçülmekte olan değişkenler çubuğun açısı ve topun konumudur. Çubuğun yatayla yaptığı açı motorun bir adımda yaptığı dönüş açısı bilindiği için istenirse hesaplanabilmektedir. Ölçülemeyen değişken ise dış etkenlere de maruz kalacak olan topun çizgisel hızıdır.

Kontrolör tasarımı yapılırken P, PD, PI, PID kontrolörler birbirlerinden bağımsız olarak kullanılabilir şekilde tasarlanmıştır. Kp, Kd ve Ki katsayıları kullanıcı tarafından serbestçe seçilebilir şekilde ayarlanmış, böylece bu katsayıların sistem davranışına etkilerinin gözlemlenmesi ve kontrolör yapılarının karşılaştırılması olanaklı hale getirilmiştir.

Kararlı olmayan sistemlerin kontrolünde PID yönteminin kullanımı günümüzde oldukça yaygındır. Bu projenin benzer çalışmalara göre en önemli farkı ise otomatik kontrol eğitiminde PID kontrolörde katsayıların etkileri açısından, bilgisayardan bağımsız gözlemlenebilir bir örnek teşkil etmesidir.

BÖLÜM 2

LİTERATÜR TARAMASI

Top çubuk sistemi çok sayıda bilim adamı tarafından incelenmiş ve yeni kontrol metotlarının denenmesinde algoritmaların test edilmesi için kullanılmıştır. P.Wellstead'ın yayınında kontrol yöntemleri ve sistem modellemesinden genel olarak bahsedilmiştir [1]. W. Yu ve F. Ortiz PD kontrolör ile top-çubuk sisteminin kararlılık analizini gerçekleştirmişlerdir [2]. Y.Jiang, C. McCorkell ve R.B. Zmood top-çubuk sistemi için konvansiyonel kutup yerleştirme ve sinir ağları yöntemlerini uygulamışlardır [3].

K. C. Ng ve M. M. Trivedi sinirsel bulanık kontrol üzerinde çalışmışlardır [4]. M. A. Marra, B. E. Boling ve B. L. Walcott genetik algoritma ile top-çubuk sistemi kontrolünü gerçekleştirmişlerdir [5]. P. H. Eaton, D.V. Prokhorov ve D.C. Wunsch top-çubuk sisteminin sinirsel bulanık kontrolünü düzgün ve doğrusal olmayan sürtünmede analiz etmişlerdir [6]. H. S. Ramirez top-çubuk sisteminin yörünge planlama yaklaşımını çalışmıştır [7]. J.Yi, N. Yubazaki ve K. Hirota top-çubuk sisteminin kararlılığı için yeni bir bulanık denetleyici üzerinde çalışmışlardır [8]. F. Song ve S. M. Smith top-çubuk sisteminde bulanık mantık kontrolörün optimizasyonunu sağlamışlardır [9]. J. M. Lemos, R. N. Silva ve J. S. Marques düzenek üzerinde sensör çıkış değerlerine göre adaptif kontrol üzerinde çalışmışlardır [10]. R. M. Hirschorn artımsal kaydırma yöntemi ile düzeneği kontrol etmeye çalışmıştır [11].

Q. Wang, M. Mi, G. Ma ve P. Spronck yeni bir sinirsel kontrolör geliştirmiştir [12]. Burada genetik algoritma yardımıyla en uygun şekle sokmuştur. D.A. Voytsekhovsky ve R. M. Hirschorn kayan kipli kontrol ile sistemin karalı hale getirilmesi konusunda önemli bir çalışma yapmışlardır [13]. K. C. Ng ve M. M. Trivedi sistemin bulanık mantıkla ve gerçek zamanlı kontrolünü yapmışlardır [14]. J. Huang ve W. J. Rugh lineer olmayan servo mekanizma problemi için yeni bir yaklaşım metodu geliştirmişlerdir [15]. N. H. Jo ve J. H. Seo top- çubuk sistemi için durum gözlemcisi üzerinde çalışmışlardır [16].

I. Petroviç, M. Brezak ve R. Cupec sistemin yapay görme temelli kontrolünü yapmışlardır [17]. Benzer bir çalışma bulanık mantık kullanılarak P. Dadios ve R. Baylon tarafından yapılmıştır [18]. J. Whelan ve J. V. Ringwood aynı sistemi PID kontrolör ile kameradan konum takibi yaparak çalıştırmışlardır [19]. S. Sridharan ve G. Sridharan silindir üzerinde kayan top-çubuk sistemi adı ile yeni bir kontrol laboratuvarı cihazı geliştirmişlerdir. Bilim adamları burada PID kontrolör kullanmışlardır [20].

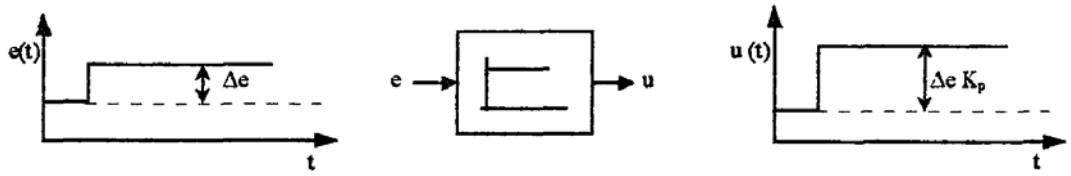
BÖLÜM 3

PID KONTROLÖR TASARIMI

PID kontrolörlerin yapısında oransal(P), integral(I) ve türevsel(D) kontrol elemanları bulunur. İsteğe bağlı olarak sistem P, I, PI, PD veya PID kontrolör olarak kullanılabilir. Hangi kontrol elemanından ne oranda kullanılacağı da tasarımcı tarafından seçilebilir. Uygulamada katsayıların ayarlanması ile en iyi sonuç PID kontrolörden alınır [22].

3.1. Oransal (P) Kontrolör

Kontrolör çıkışındaki sinyalin girişteki sinyalle orantılı olduğu kontrolör çeşididir. Lineer olmayan sistemlerde birim basamak girişine yanıt olarak çıkış sabit bir değere yaklaşırsa sistem kararlı kabul edilir ve oransal katsayı ayarlanarak kontrol edilebilir. K_p artırılırsa cevap süresi kısalır fakat bu durumda sistem geri beslemeli ise kararsız hale gelebilir. Açık çevrim kontrol sistemlerinin ise kararlılık problemi yoktur. Oransal kontrol kullanarak çıkış asla istenilen değere ayarlanamaz ve mutlaka az ya da çok kararlı hal hatası oluşur. Kontrol elemanının %100 çıkış vermesi için gerekli olan hatadaki yüzde değişim oransal bant olarak tanımlanır. Oransal bant daraltılırsa kazanç artacaktır [21].



Şekil 3.1 Oransal Kontrolörün Birim Basamak Cevabı ve Blok Diyagramı [21]

$$u(t) = Kp.e(t) \tag{3.1}$$

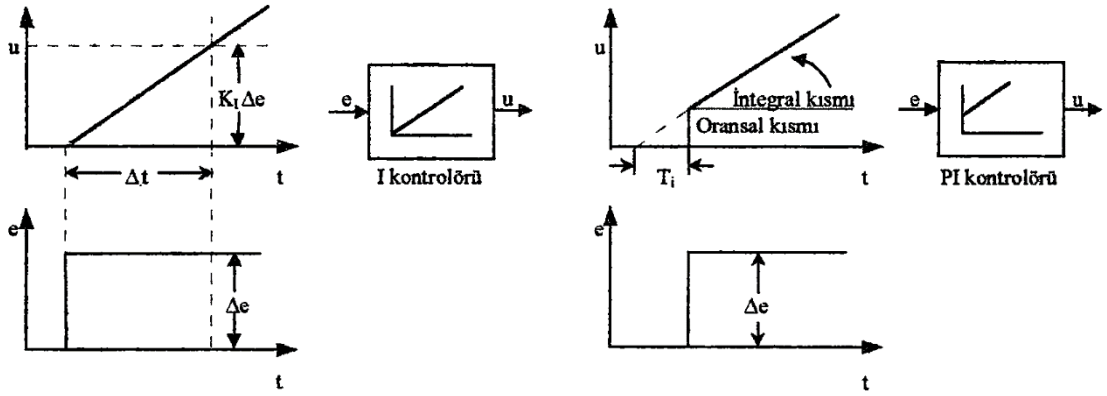
$$U(s) = Kp.E(s) \tag{3.2}$$

$$Kp = \frac{100}{Oransal\ Bant} \tag{3.3}$$

3.2. Oransal-İntegral (PI) Kontrolör

Oransal kontrolde kararlı hal hatası çıkış kontrol işaretinin değişmediği noktaya ulaştığı anda işaret ile set değer arasındaki farktır. Kontrolör kararlı hal hatası sıfırlanana kadar değişen bir sinyal üretirse bu hata giderilebilir. İntegral kontrolör kararlı hal hatasının sıfır yapılması için kullanılır fakat geçici cevabı kötü etkiler. Bu yüzden tek başına kullanılması tercih edilmez ve her zaman oransal kontrolörle birlikte kullanılır. Integral kontrolör kullanımıyla giriş sinyalinin zamana göre integrali ile orantılı kontrol sinyali üretilir. PI kontrolörün transfer fonksiyonu denklem 3.4'de verilmiştir [22].

$$Gp(s) = Kp + \frac{Ki}{s} \tag{3.4}$$



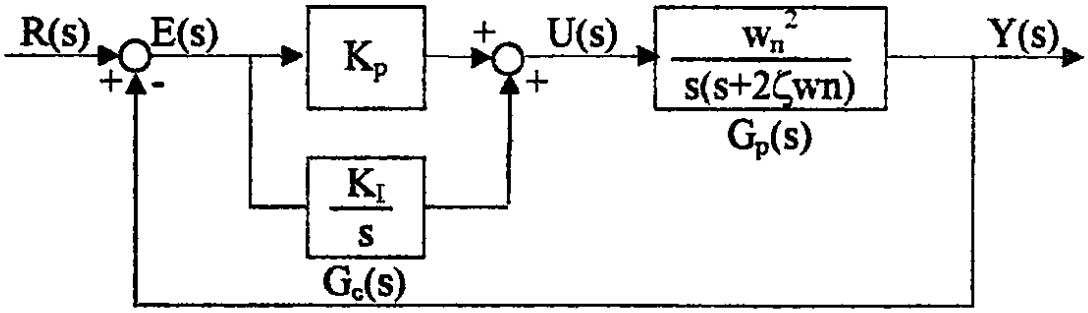
Şekil 3.2 I ve PI Kontrolörlerin Birim Basamak Cevabı ve Blok Diyagramları [21]

Ti integral teriminin oransal terime ulaşması için gereken zamandır. Oransal kontrolöre integral kontrolör eklendiğinde oransal katsayı azaltılmalıdır. İntegral kontrolör salınımlar yaparak zamanla kararlı hal hatasını sıfırlar.

Şekil 3.3'de görülen sistemin açık çevrim transfer fonksiyonu denklem 3.5'de verilmiştir.

$$G(s) = G_c(s).G_p(s) = \frac{w_{n^2}.(K_P s + K_i)}{s^2.(s + 2\zeta w_n)} \tag{3.5}$$

PI kontrolör ileri yönlü transfer fonksiyonuna $s = \frac{K_i}{K_p}$ 'de bir sıfır ve s=0'da bir kutup ekler. Böylece sistemin tipi artarken kararlı hal hatası bir derece düzelir. Örneğin hata sabitse sıfıra düşer. PI kontrollü birinci dereceden sistemler için oransal katsayı önemlidir çünkü rampa girişi uygulandığında kararlı hal hatasının genliği girişle doğru orantılıdır fakat K_p çok büyük seçilirse sistem kararsız hale gelebilir [21].



Şekil 3.3 PI Kontrolörlü Kontrol Sistemi Blok Diyagramı [21]

3.2.1 PI Kontrolörün Zaman Tanım Bölgesinde İncelenmesi

$G_c(s)$'nin sıfırının yeri uygun seçildiğinde sönüm iyileştirilebilir. PI kontrolör alçak geçiren filtre olduğundan sisteme eklendiğinde daha uzun yerleşme ve yükselme zamanına sebep olur. $s = \frac{K_i}{K_p}$ sıfırı kutuplardan uzak ve orijine çok yakın seçilmelidir. $s = \frac{K_i}{K_p}$ sıfırı G(s)'nin en yakın kutbuna göre çok küçük seçilirse kök yer eğrisinde orijinin civarlarında küçük bir daire oluşur. Kararlı hal hatası kıstasına uygun K_I katsayısı belirlenir ve gerekli sönüm oranını veren K_p seçilir. Yükselme zamanı ve yerleşme zamanını minimum yapmak için baskın kutuplar karmaşık eşlenik olarak seçilir [21].

3.2.2. PI Kontrolörün Frekans Tanım Bölgesinde İncelenmesi

Transfer fonksiyonu denklem 3.6'da ki şekilde yazılır.

$$G_c(s) = K_P + \frac{K_i}{s} = \frac{K_\text{I}.(1+\frac{K_p.s}{K_\text{I}})}{s} \tag{3.6}$$

PI kontrolörün tasarımı yapılırken önce kompanze edilmemiş sistemin ileri yol transfer fonksiyonunun ($G_p(s)$) Bode diyagramı kararlı hal hatası uygun olacak şekilde seçilen kazançla çizilir. Sonra buradan faz ve kazanç bulunur. Belirli bir faz payına ulaşmak için yeni faz sınırı payı da (ω_g) buradan bulunur. Bu frekansta kompanze edilmiş transfer fonksiyonun genlik eğrisi x ekseninde 0dB'den geçmelidir. Kompanze edilmemiş transfer fonksiyonu genlik eğrisinin yeni kazanç payı frekansında 0dB olması için PI kontrolörün kazancına eşit miktarda zayıflatma yapması gerekmektedir. K_P değeri şu şekilde bulunur;

$$\left|G_p.(j\omega_g)\right|_{dB} = -20.\log_{10} K_p\ dB \qquad K_P < 1 \tag{3.7}$$

$$K_p = 10^{-\left|G_p.\left(jw_{g'}\right)\right|_{dB/20}} \qquad K_P < 1 \tag{3.8}$$

K_p belirlendikten sonra uygun bir K_i değerinin seçilmesi gerekir. $\frac{K_\text{İ}}{K_P}$ oranı seçilirken özgün fazı çok değiştirmeyecek ve sistemin bant genişliği çok dar olmayacak şekilde denklem 3.9'da ki eşitliğe göre belirlenmelidir.

$$\frac{K_i}{K_p} = \frac{\omega_g}{10}\ rad/s \tag{3.9}$$

Bu belirlemeden sonra gerekli denemeler yapılır ve ilgili katsayılar tespit edilir. Kontrol edilen sistem sıfır tipinde ise $K_\text{İ}$ rampa hatasını düzeltecek şekilde seçilir. Faz payı, kazanç payı, rezonans tepesi ve kapalı çevrim bant genişliği hesaplanarak en uygun K_p değeri seçilir [21].

PI kontrolör maksimum aşmayı (M_p) azaltır, sönümü iyileştirir, yükselme zamanını (t_r) arttırır, bant genişliğini (B.W.) azaltır, kazanç payını iyileştirir, faz payını iyileştirir, rezonans tepesini iyileştirir ve yüksek frekanslı gürültüleri filtre eder [21].

3.3. PD Kontrolör

PI kontrolörün yapısında integral bileşeni olduğu için anlık hata sıfır olsa bile kontrol işareti üretilmeye devam eder. Osilasyonları engellemek için hatanın türevi alınıp kontrolöre bildirilmelidir. Türevsel kontrol yalnız başına kullanılamaz ve ancak

oransal kontrolörle birlikte çalışması mümkündür. Şekil 3.4'de PD kontrolörlü kontrol sistemi blok diyagramı gösterilmektedir.

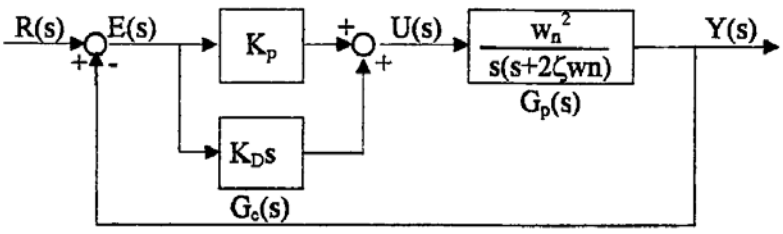
$$G_P(s) = \frac{{\omega_n}^2}{s.(s+2\zeta w_n)} \tag{3.10}$$

$$G_c(s) = K_p + K_d(s) \tag{3.11}$$

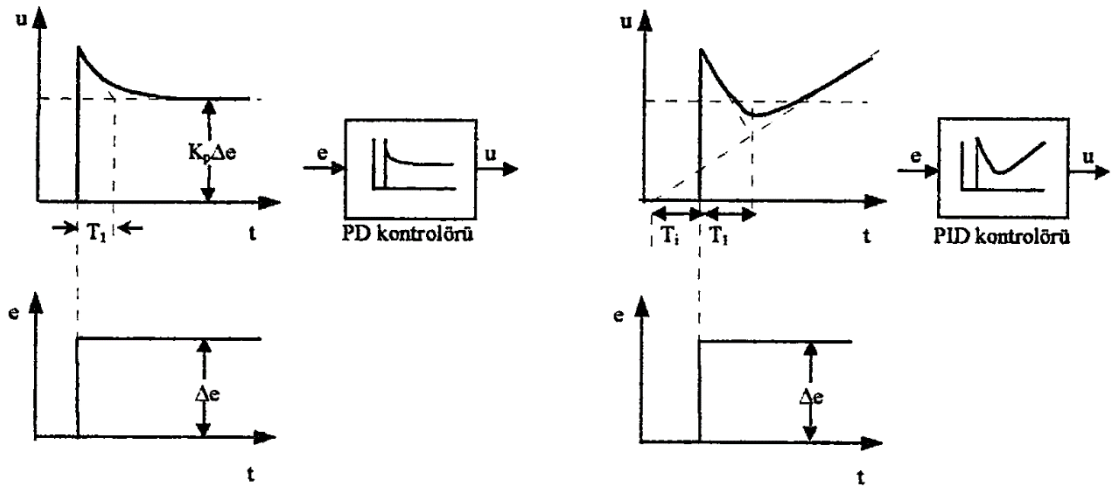
$$u(t) = K_P.e(t) + K_d.\frac{de(t)}{dt} \tag{3.12}$$

$$G(s) = \frac{Y(s)}{E(s)} = G_c(s).G_p(s) = \frac{{\omega_n}^2.(K_p+K_d(s)}{s.(s+2\zeta w_n)} \tag{3.13}$$

PD kontrolör t domeninde hata işaretinin eğimini barındırdığından sezgisel bir kontrolör yapısıdır. Bu şekilde hata yönü önceden hissedilir ve önlem alınabilir. Doğrusal sistemlerde basamak cevabına ilişkin e(t) ya da y(t)'nin eğimi büyük olursa yüksek aşma meydana gelir. Türevsel kontrolör hata sinyalinin ani eğimini ölçer ve böylece aşımın yüksek olup olmayacağını tahmin eder. Bu öngörü sonucu uygun bir düzeltme sinyali gönderilir. Sistemin kalıcı durum hatası zamanla değişiyorsa PD kontrolör devreye girer ve durumu düzeltme yönünde çaba gösterir. Eğer hata zamanla değişim göstermiyorsa türevsel kısım etkisiz hale gelecektir. Bu kontrolör sistem tipini değiştirmezken geçici hal hatasını düzenler. Şekil 3.5'de PI ve PID kontrolörlerin birim basamak cevapları verilmiştir.



Şekil 3.4 PD Kontrolörlü Kontrol Sistemi Blok Diyagramı [21]



Şekil 3.5 PD ve PID Kontrolörlerin Birim Basamak Cevapları ve Blok Diyagramları [21]

$$u(t) = K_p.e + K_d.\frac{de}{dt} \tag{3.14}$$

$$u(t) = K_p.e + K_d.\frac{de}{dt} + K_i \int e.dt \tag{3.15}$$

s domeninde Bode diyagramının kolay çizilebilmesi için PD kontrolör şu şekilde tanımlanır;

$$G_c(s) = K_p + K_d s = K_p(1 + \frac{K_d}{K_p} s) \tag{3.16}$$

PD kontrolör yüksek geçiren kontrolördür. Faz ilerletmeli özelliği ile kontrol sisteminin faz payını iyileştirir [21]. Ancak faz payı frekansını yükseltir. K_p/K_d oranı değiştirilerek sönümü iyileştirecek en iyi çözüm bulunmaya çalışılır.

PD kontrolör sönümü iyileştirir, maksimum aşmayı (M_p) azaltır, yükselme zamanını (t_r) ve yerleşme zamanını (t_o) düşürür, bant genişliğini (B.W.) arttırır, kazanç payı, faz payı ve M_r'yi iyileştirir. Yüksek frekanslarda gürültüyü ön plana çıkarabilir, az

sönümlü ve kararsız sistemlerde etkisizdir. PD kontrolör değişken kararlı hal hatalarına etki eder [21].

3.4. PID Kontrolör

PD kontrolör sisteme sönüm eklemekte fakat kararlı hal hatasını değiştirmemektedir. PI kontrolör göreli kararlılığı ve kararlı hal hatasını iyileştirmekte fakat yükselme zamanını arttırmaktadır. Bu yüzden bunların iyi yönlerini kullanmakta olan PID kontrolör yapısı birçok uygulamada gerekli hale gelir. PD ve PI bileşenlerinin birbirine seri olarak bağlandığı varsayımıyla aşağıdaki denklemeler oluşturulur.

$$G_c(s) = K_p + K_d s + \frac{K_i}{s} = (1 + K_{d1} s).(K_{p2} + \frac{K_{i2}}{s}) \quad (3.17)$$

$$K_p = K_{p2} + K_{d1}.K_{i2} \quad (3.18)$$

$$K_d = K_{d1}.K_{P2} \quad (3.19)$$

$$K_{\mathrm{i}} = K_{i2} \quad (3.20)$$

Burada üç parametre yeterli olacağından PD kontrolörün oransal katsayısı 1 alınmıştır. İlk olarak sadece PD kontrolörün etki ettiğini düşünülür. İstenilen bağıl kararlılığa ulaşabilmek için K_{d1} değeri seçilir. T domeninde bağıl kararlılık maksimum aşma s domeninde ise faz payı ile ölçülür. Daha sonra K_{i2} ve K_{p2} ihtiyaca göre belirlenir. İlk önce PI kontrolörü seçip PD'yi sona bırakmak da mümkündür.

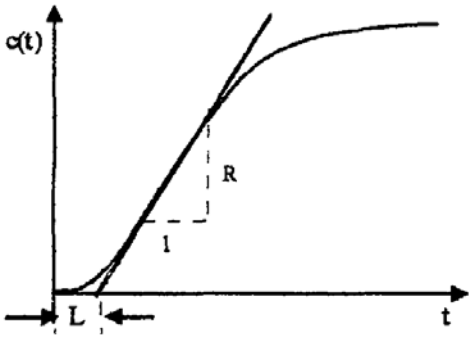
3.5. Ziegler- Nichols Kuralları

Matematiksel modeli bilinmeyen sistemlerin transfer fonksiyonunun çıkarılmasında en yaygın olarak kullanılan metot iki ayrı yöntemden oluşan Ziegler-Nichols yöntemidir. Sistemin matematiksel modeli bulunabilse dahi bu yöntemi kullanmak işi kolaylaştırır. Bu yöntemde kriterlere uygun olarak çalışacak sistemin salınımlı fakat ikinci aşma ilk aşmanın %25'inden küçük olacak şekilde tasarlandığı bilinmektedir. Çeyrek sönümleme ölçütü olarak bilinen denklem aşağıdaki formdadır.

$$J = \int_0^\infty |e(t)|.dt = \int_0^\infty |r(t) - y(t)|.dt \quad (3.21)$$

Birçok sistem açık çevrim birim basamak cevabına sahiptir. Şekil 3.6'da R cevap eğrisinin en yüksek eğimi, L ise bu eğrinin zaman eksenini kestiği noktadır. Birinci ve ikinci dereceden doğrusal sistemlere bu yöntem uygulanamaz. Çünkü bunlarda L değeri pozitif çıkmaz. Daha yüksek dereceli sistemler içinse tabloda verilen değerler kullanılarak bu yöntem uygulanabilir.

Kapalı çevrim yöntemi uygulanırken ise oransal katsayı hariç diğer katsayılar sıfırlanır ve oransal katsayı sıfırdan başlanarak sistem osilasyon yapana kadar arttırılır. Burada kazanç K_{pu} ve periyot P_u olacak şekilde, tabloda verilen değerler uygulanır.



Şekil 3.6 Ziegler- Nichols Teoremi Açık Çevrim Cevabı [21]

Tablo 3.1 Ziegler- Nichols Katsayılarının Hesaplanması

Kontrolör Tipi	1.Yöntem (Açık Çevrim)	2. Yöntem (Kapalı Çevrim)
P Kontrolör	$K_p = 1/R.L$	$K_P = 0{,}5.K_{pu}$
PI Kontrolör	$K_p = 0{,}9/R.L \quad T_i = 3{,}3.L$	$K_P = 0{,}45.K_{pu}$ $T_i = 0{,}83.P_u$
PID Kontrolör	$K_p = 1{,}2/R.L$ $T_i = 2.L \quad T_d = 0{,}5.L$	$K_P = 0{,}6.K_{pu}$ $T_i = 0{,}5.P_u \quad T_d = 0{,}125.P_u$

BÖLÜM 4

DİJİTAL PID KONTROLÖR TASARIMI

Günümüzde dijital kontrolörlerin kullanımı mikrodenetleyici teknolojisindeki gelişmeler ile birlikte çok yaygın hale gelmiştir. Dijital PID kontrol sistemi tasarımı mantık olarak PID kontrol sistem tasarımı ile aynıdır. Burada kontrolör dijital bilgiye göre tasarlanır. İnceleme zaman domeninde ve frekans domeninde gerçekleştirilir. s tanım bölgesindeki tasarım kriterleri z tanım bölgesine uyarlanır. Tasarım mikrodenetleyici kodunun değiştirilmesi ile en az sayıda deneme yapılarak gerçekleştirilebilir. Burada K_P oransal katsayısı sabit kazanç olarak alınır.

f(t) fonksiyonun t=kT anında zamana göre türevi, geri dikdörtgen fark kuralı ile t=kT ve t=(k-1)T zamanlarında ölçülen değerleri kullanarak dijital türev alıcının z-transfer fonksiyonu;

$$G_d(z) = K_d.\frac{z-1}{Tz} \tag{4.1}$$

şeklinde bulunur. Burada örnekleme periyodu çok önemlidir. Sürekli bilgi dijital bilgiye çevrilirken bilgi kaybı çok olmayacak şekilde T değeri yeterince küçük alınmalıdır. Bu da mikrodenetleyicinin işlem sayısını arttıracağından optimum örnekleme zamanı seçilmelidir.

İntegral kontrolörün dijitalleştirilmesinde yamuk, ileri veya geri integrasyon kurallarından herhangi biri kullanılabilir.

Yamuk integrasyon;

$$u(kT) = u[(k-1)T] + \frac{T}{2}.[\text{f(kT)+f(k-1)T}] \quad (4.2)$$

$$G_i(z) = K_i.\frac{U(z)}{F(z)} = \frac{K_i T(z+1)}{2.(z-1)} \quad (4.3)$$

$$G_c(z) = \frac{\left(K_p + T.\frac{K_i}{2} + \frac{K_d}{T}\right).z^2 + \left(T.\frac{K_i}{2} - K_p - \frac{2K_d}{T}\right).z + \frac{K_d}{T}}{z.(z-1)} \quad (4.4)$$

İleri dikdörtgen integrasyon;

$$u(kT) = u[(k-1)T] + T.\,\text{f(kT)} \quad (4.5)$$

$$G_i(z) = K_i.\frac{U(z)}{F(z)} = \frac{K_i T.z}{(z-1)} \quad (4.6)$$

$$G_c(z) = \frac{\left(K_p + \frac{K_d}{T} + T.K_i\right).z^2 - \left(K_p + \frac{2K_d}{T}\right).z + \frac{K_d}{T}}{z.(z-1)} \quad (4.7)$$

Geri dikdörtgen integrasyon;

$$u(kT) = u[(k-1)T] + T.\,\text{f [(k-1)T]} \quad (4.8)$$

$$G_i(z) = K_i.\frac{U(z)}{F(z)} = \frac{K_i T}{(z-1)} \quad (4.9)$$

$$G_c(z) = \frac{\left(K_p + \frac{K_d}{T}\right).z^2 + \left(T.K_i - K_p - \frac{2K_d}{T}\right).z + \frac{K_d}{T}}{z.(z-1)} \quad (4.10)$$

4.1. Dijital PD Kontrolör

Kurulacak sisteme de bağlı olarak en çok kullanılan dijital kontrol metodudur. 1 tipi sistemlerde birim basamak girişi için kalıcı hata sıfır olacağından z=1'de kutup yerleştirilmez. PD kontrolör yeterli gelebilir ve integral katsayısı sıfır seçilir. PD kontrolörün z=0'da kutbu ve pozitif gerçel eksende birim daire içinde $z = K_d/(K_pT + K_{d)}$'de sıfır vardır [22].

$$D(z) = \frac{(K_pT + K_d).z - K_d}{T_z} \quad (4.11)$$

$$D(z) = K_{p.}\frac{z-z_1}{z-p_1} \quad |z_1| > |p_1| \tag{4.12}$$

$$K_p = \frac{1-p_1}{1-z_1} \tag{4.13}$$

4.2. Dijital PI Kontrolör

Sıfır tipindeki sistemin kararlı hal hatasını sıfırlamak için PI kontrolör kullanılır. Sistem tipini artırarak kararlı hal hatasını iyileştirir ve bağıl kararlılığı artırır. PI kontrolörün z=1'de kutbu ve $z = \frac{K_p - K_i.T}{K_p}$ 'de sıfırı vardır. z=1 seçilmesiyle $K_v = \infty$ olur ve kararlı halde K_P'nin önemi kalmaz [22]. K_p ayarlanarak göreli kararlılık düzeltilir. İleri dikdörtgen integrasyona göre ayrık PI kontrolör şu şekilde gösterilir.

$$D(z) = \frac{(K_p + K_i.T).z - K_p}{z-1} \tag{4.14}$$

$$D(z) = K_p.\frac{z-z_1}{z-p_1} \quad |z_1| < |p_1| \tag{4.15}$$

4.3. Dijital PID Kontrolör

Dijital PID kontrolör z düzlemi kök eğrisine z=0 ve z=1'de iki adet kutup ve bu iki kutbun ortasında birim dairenin sağ tarafında iki adet sıfır ekler. Küçük K değerlerinde kapalı çevrim kutbu z=1'e çok yakın olduğundan aşım çok büyük olur [21].

Önce K_v hata sabiti değerinden, K_i hesaplanır, sıfırlar tespit edilir sonra K_p ve K_d değerleri bulunur. İleri dikdörtgen integrasyona göre transfer fonksiyonu şu şekilde bulunur.

$$G_c(z) = \frac{\left(K_p + \frac{K_d}{T} + T.K_i\right).z^2 - \left(K_p + \frac{2K_d}{T}\right).z + \frac{K_d}{T}}{z.(z-1)} \tag{4.16}$$

BÖLÜM 5

STEP MOTORLAR

5.1. Step Motorların Genel Özellikleri

Step motorlar yapısal özelliklerine, kullanıldıkları yerlere ve çalışma prensiplerine göre çok farklı gruplara ayrılırlar. Temelde yapılarına göre Sabit Mıknatıslı, Değişken Relüktanslı ve Hibrit Step Motorlar olarak gruplandırılırlar. Hassas dönme ya da öteleme hareketi gerektiren uygulamalarda sık tercih edilirler.

Step motorların sargılarına her sinyal uygulanışında yapmış oldukları dönme miktarına adım, bunun açısına adım açısı denir. Bu motorlar otomatik kontrol sistemlerinde geri beslemesiz olarak çalıştırılabilecek kadar güvenilir hassasiyete ve doğruluğa sahiptirler. Adım açıları $1,8^{o}$ hatta $0,72^{o}$ olanları vardır.

Çalışmaları incelendiğinde step motorların birçok üstün özelliği ile karşılaşılır. Step motorlar geri beslemesiz devre kontrol sisteminde; dönüş açısıyla giriş darbelerinin oranını sabit tutarak, dönüş hızıyla darbe frekansı yani giriş darbe oranını sabitleyerek ve açısal hatayı hemen hemen hesaplanmayacak düzeye getirerek tam pozisyon doğruluğunu sağlarlar [23]. Step motorlar; son sargının enerjili kalması durumunda motor sargılarının uyarılması veya sistemin enerjisiz kalması durumunda sabit mıknatıs kullanması itibariyle ürettikleri tutma torku sayesinde son pozisyonlarını korurlar [23]. Step motorların çalışması faz sayısına göre sahip oldukları stator sargılarının ardı ardına uyarılması prensibine dayandığı için sargı enerjileri doğru verildiği sürece başlama, durma ve herhangi bir çalışma periyodunda yön değiştirme olaylarına hızlı tepki

verebilirler [24]. Birçok motor yapısında var olan ve çalışma esnasında elektrik arkı yaratan fırça sistemine sahip değillerdir [24]. Bu özelliklerden farklı olarak step motorların açık çevrim davranışlarının ±1 step doğruluk pozisyonuna sahip olması, mekanik sistemde mil hareketinin yeterli ölçüde olması ve yüke yeterli momenti sağlayabilmeleri diğer üstün özelliklerindendir [25]. Step motorlar endüstride kullanılan diğer motorlarla karşılaştırılırsa aşağıdaki tablo elde edilir [26].

Tablo 5.2 Step Motorların Diğer Motorlara Göre Avantaj ve Dezavantajları

	STEP MOTORLAR	DC SERVO MOTORLAR	FIRÇASIZ DC MOTORLAR	AC İNDÜKSİYON MOTORLARI
AVANTAJLARI	*Açık döngü kontrol *Düşük hızda dönme *Düşük hız-yüksek tork karakteristiği *Sıralı darbe girişi *Yüksek hızda adım alma kabiliyeti	*Kapalı döngü kontrolü *Yüksek tork *Yüksek dönme verimi	*Yüksek hız-yüksek tork *Yüksek dönme verimi	*Yüksek hız *Yüksek verim
DEZAVANTAJLARI	*Adım hatası *Manyetik gürültü	*Fırçalı çalışma *Doğrulama limiti *Düşük güvenilirlik	*Servo motora göre yüksek maliyet	*Alternatif frekans limiti *Hız kontrol zorluğu

5.2. Step Motorların Karakteristik Özellikleri

5.2.1. Statik Karakteristikler

Bu karakteristikler zamana göre değişim göstermeyen veya motorun yapısal özellikleriyle ilgili olan karakteristikler olup aşağıdaki gibi gruplandırılabilir [26].

5.2.1.1. Sargı Direnci (R)

Step motorun bir fazı için stator sargısının DC direnç değeridir. Birimi Ω (ohm)'dır.

5.2.1.2. Sargı İndüktansı (L)

Step motorun bir fazı için stator sargısının maksimum indüktans değeridir. Birimi H (Henry)'dir.

5.2.1.3. Rotor Atalet Momenti (J_M)

Rotor eksenlerine göre değişen eylemsizlik momentidir. Birimi $kg.m^2$'dir.

5.2.1.4. Adım Açısı (θ_a)

Bir darbe geldiğinde motor milinin teorik dönme açısıdır. Birimi ^{o} (derece)'dir.

5.2.1.5. Temel Adım Açısı (θ_t)

Bir darbe geldiğinde motor milinin gerçek dönme açısıdır. Birimi ^{o} (derece)'dir.

5.2.1.6. Nominal Sargı Akımı (I_R)

Çalışma anında çekilen akımdır. Uygulanan gerilime bağlıdır. Birimi A(amper)'dir.

5.2.1.7. Histerezis Hatası (Δθh)

Motorun saat yönünde ve saat yönünün tersinde döndüğünde yaptığı açısal farklılıktır. Birimi ^{o} (derece)'dir.

5.2.2. Dinamik Karakteristikler

Motor çalışırken değişen karakteristiklerdir ve şu şekilde listelenebilir.

5.2.2.1. Tutma Momenti (T_d)

Motor uyarılmışken rotorun pozisyonunu korumasını sağlayan momenttir. Birimi N.m'dir.

5.2.2.2. Darbe Oranı (f_p)

Motora uygulanan sinyalin frekansıdır. Birimi Hz (Hertz)'dir.

5.2.2.3. Maksimum Başlama Frekansı (f_s)

Motor yüksüzken harekete geçebileceği en yüksek frekans değeridir. Birimi Hz (Hertz)'dir.

5.2.2.4. Başlama Torku (T_s)

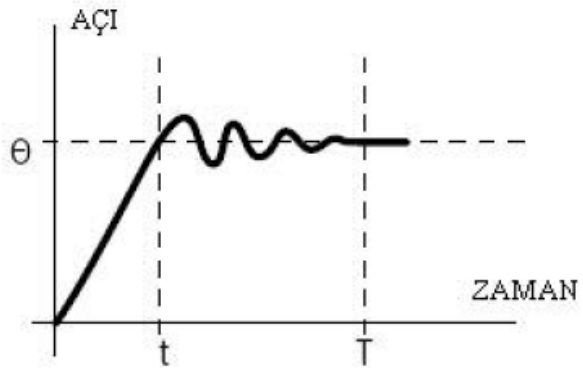
Belli bir frekansta motorun kendisini sürebilmesi için gerekli olan maksimum yük momentidir. Birimi N.m'dir.

5.2.2.5. Çekme Torku (T_o)

Belli bir frekans değerinde senkron çalışan maksimum moment değeridir. Birimi N.m'dir.

5.2.2.6. Tek Adım Tepkisi ve Rezonansı

Step motora tek bir adım gönderildiğinde adımı attıktan sonra yaptığı salınımdır. Şekil 5.1'de gösterilmiştir.



Şekil 5.1 Step Motorun Tek Adım Tepkisi ve Rezonansı [27]

5.2.3. Step Motor Çalışma Eğrileri

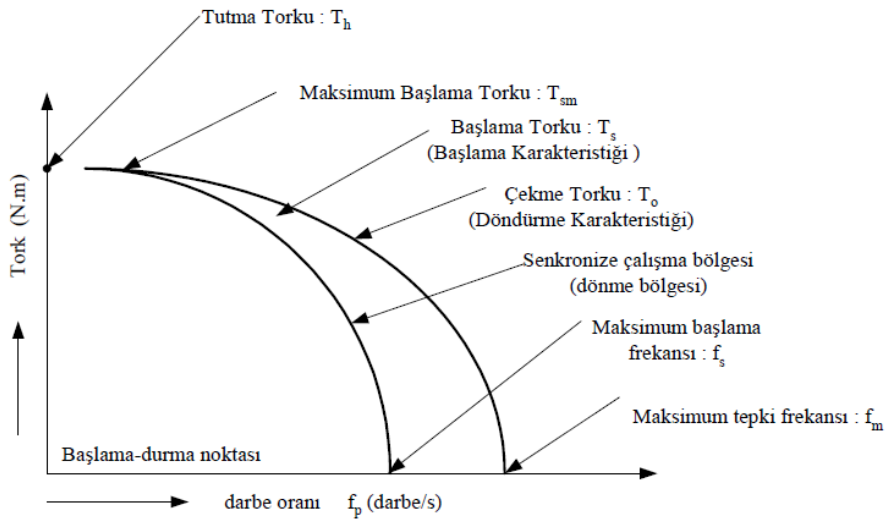
5.2.3.1. Darbe Oranı ve Eylemsizlik Momenti Karakteristiği

Yükün eylemsizlik momenti ile başlama frekansı arasındaki ilişkiyi veren eğridir. Yükün eylemsizlik momenti yükseldikçe başlama frekansı da yükselecektir.

$$f_{SL} = \frac{f_S}{1+\frac{J_L}{J_M}} \tag{5.1}$$

5.2.3.2. Darbe Oranı- Tork Karakteristiği

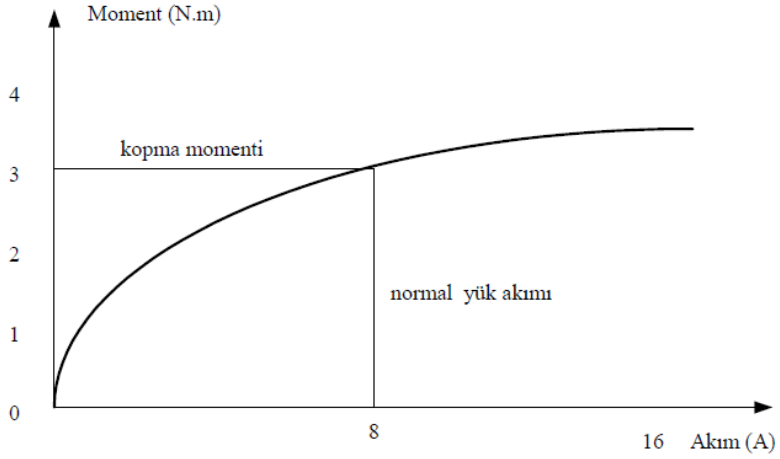
Giriş darbe frekansı ile tork arasındaki ilişkiyi gösteren eğridir (Şekil 5.2).



Şekil 5.2 Darbe Oranı- Tork Karakteristiği [27]

5.2.3.3. Moment- Akım Karakteristiği

Step motorun akımına bağlı olarak bir moment üretir. İlgili eğri Şekil 5.3'de gösterilmiştir.



Şekil 5.3 Step Motor Akım- Moment Grafiği [27]

5.3. Hibrit Step Motorlar

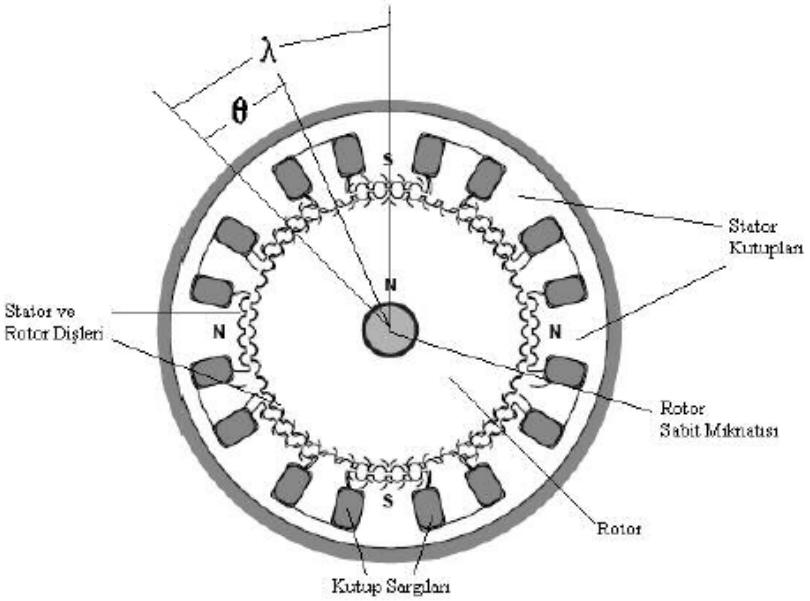
Teknolojinin gelişmesi insanoğlunu günden güne farklı yapı ve donanımlarla uğraşmaya sevk etmiştir. Bu durum çok yüksek güçlü anahtarlama transistörlerinin ve devrelerinin yapılabilme olanağını da beraberinde getirmiş bu gelişmeler sonrası step motor daha etkin bir şekilde kullanılmaya başlanmıştır. 1930'lu yılların ardından step motor alanındaki gelişmeler, 1960'lı yıllarda sabit mıknatıslı ve değişken relüktans motorlarının düzgün bir yapı içerisinde birleştirilerek yeni bir motor türünün ortaya çıkmasını sağlamıştır [27]. Bu yeni motora karma anlamına gelen Hibrit step motor (HSM) adı verilmiştir.

5.3.1. Hibrit Step Motorların Genel Özellikleri

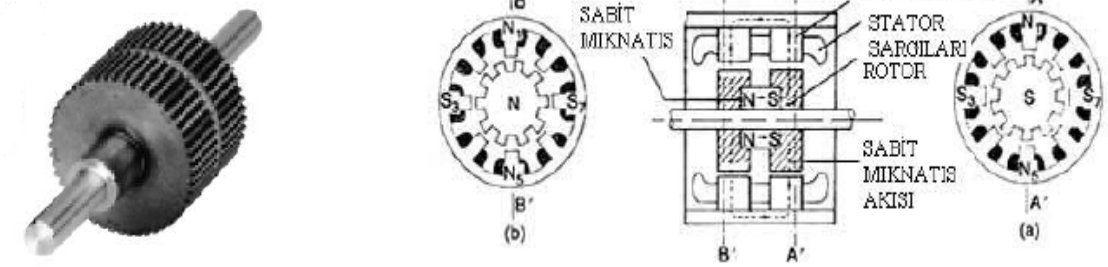
Hibrit step motorlar servo motorlara göre daha iyi hız ve pozisyon doğruluğu sağlamaktadırlar. Sabit mıknatıslı step motorlar ve Relüktanslı step motorlardaki bir takım olumsuz durumlar bu yeni yapıda giderilmiştir. Hibrit step motorlar kullanılacakları sistem ve yük özelliklerine göre farklı yapılarda üretilebilirler. Bu tip motorlar, aynı mil üzerine monte edilmiş birbirinin aynısı olan iki yumuşak demir statora sahiptir. Hibrit adım motorları yapısal olarak rotor ve stator şeklinde iki kısımdan oluşmaktadır [26, 27]. Şekil (5.4)'de kesit görünüşü verilmiştir.

Burada 8 stator kutbuna(4 faz) ve 50 rotor dişine sahip bir hibrit step motorun kesit görünüşü verilmiştir. θ yarım adım modunda stator kutupları arasındaki toplam adım açısını, λ ise tam adım modunda stator kutupları arasındaki toplam adım açısını vermektedir [27].

Hibrit step motorlarda rotor sabit mıknatıslıdır. Her dişli üzerinde değişik sayılarda çıkıntılar bulunmaktadır ve bu çıkıntıların arası disklerle yalıtılmıştır.



Şekil 5.4 Hibrit Step Motorların İç Yapısı [27]



Şekil 5.5 a) Hibrit Step Motor Rotoru [27] b) Hibrit Step Motor Statoru [27]

Hibrit step motorlarda stator, çok parçalı değişken relüktanslı yapıdadır. Statorda 8 kutup ve her kutupta 2 ila 8 arasında diş bulunur.

Hibrit step motorlar hassas pozisyon kontrolleri için kullanılır. Bu tip motorlar özellikle çift kutuplu sürücü devreleriyle sürüldüklerinde yüksek moment üretme yeteneğine sahiptirler [27].

5.3.2. Hibrit Step Motorların Çalışma Karakteristikleri

5.3.2.1. Açısal Pozisyon

Diğer motorların aksine step motorlar süreksiz kontrol işareti ile sürülürler. Piyasada farklı adım çözünürlüğünde ya da adım açısında motorlar mevcut olmasına rağmen genel olarak hibrit step motorlar her adımda $1{,}8^{o}$ dönerler. Başka bir ifadeyle 360^{o}'lik bir tam turu toplam 200 adımla tamamlarlar. SPR (Step Per Rotation) dönüş başına adım olarak tanımlanır.

$$SPR = \frac{Adım\ Dizisi}{Rotor\ Diş\ Sayısı} \tag{5.2}$$

SA - adım açısıdır.

$$SA = \frac{360}{SPR} \tag{5.3}$$

5.3.2.2. Tork ve Hız

Hibrit step motorlarda tork rotor pozisyon değiştirirken minimum değerdedir ve bu değer çalışma torku olarak adlandırılır. Maksimum hız ise tork değerinin sıfır olduğu andaki dakika başına devirdeki hızdır ve aşağıdaki denklemle hesaplanabilir.

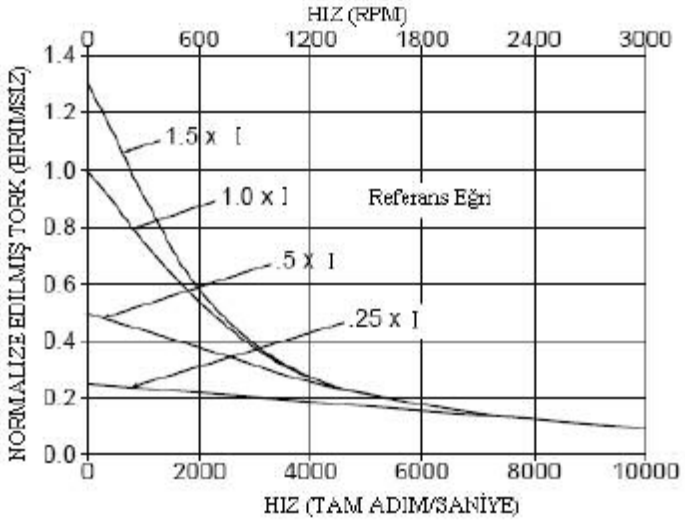
$$v = \frac{60.f}{SPR} \tag{5.4}$$

Bundan farklı olarak radyan cinsinden açısal hız (saniye başına) şöyle hesaplanabilir;

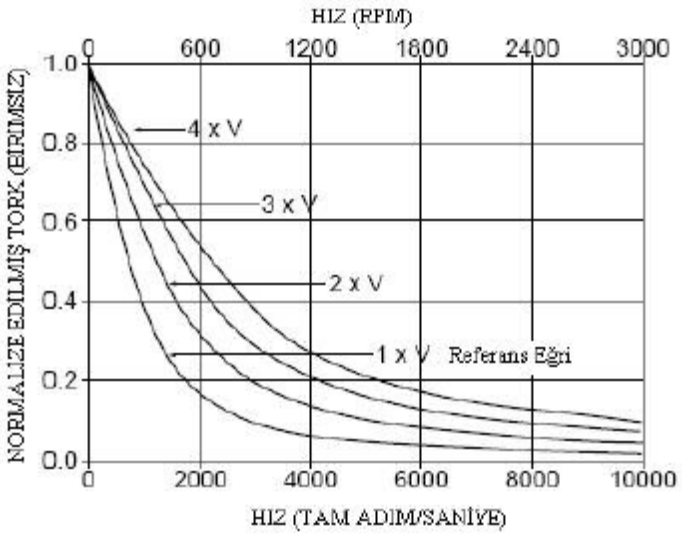
$$\omega = \frac{2.\pi.f}{SPR} \tag{5.5}$$

Şekil 5.6'da hibrit step motorların akıma bağlı hız-tork karakteristiği verilmiştir.

Grafik incelendiğinde akım değişikliğinde hız tork eğrisinin bariz bir şekilde etkilendiği görülmektedir. Farklı olarak gerilime bağlı hız-tork karakteristiği incelenirse gerilim değişimlerinden motorun fazlaca etkilenmediği ortaya çıkmaktadır.



Şekil 5.6 Hibrit Step Motorların Akıma Bağlı Hız- Tork Grafiği [27]



Şekil 5.7 Hibrit Step Motorların Gerilime Bağlı Hız- Tork Karakteristiği [27]

5.3.3.3. Rezonans

Adım motorları özellikli adım oranlarında "rezonans" olarak adlandırılan ve arzu edilmeyen bir tepki durumu oluştururlar. Bu durum ani tork kayıpları ve atlama adımları adını verdiğimiz durumları oluşturarak senkronizasyonun kaybolmasına neden olur. Rezonans tipik olarak 100-200 adım arasında adım oranı rotor dönmesiyle çakıştığında yükselir. Bu durumda motor yük eylemsizliği ve dönme etkilerini karşılayamadığı için atlama adımları ortaya çıkacak ve tork kaybı senkronizasyon bozulması meydana getirecektir. Bu durumu ortadan kaldıracak en pratik ve basit yol "microstepping" adı verilen küçük adımlama tekniği kullanmaktır [27].

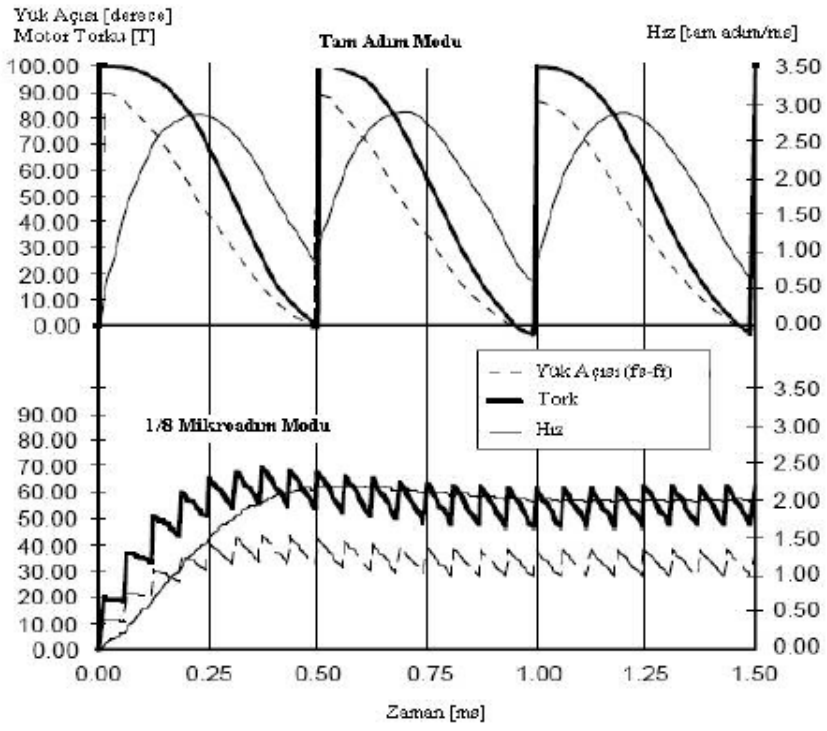
5.3.3.4. Mikro Adımlama (Microstepping)

Step motorun stator akısını taşımasında kullanılan tam ve yarım adımlama gibi başka bir yoldur. Mikro adımlama daha az titreşim ve gürültü probleminin ortadan kalkması, kararlılığın iyileşmesi gibi avantajlar sağlar. Step motor kontrolünde

kullanılan çok sayıda mikro adımlama tekniği vardır. Motor tarafından üretilen tork T, tutma torku T_d, stator akıları arasındaki uzaklık f_s ve rotor pozisyonu f_r iken;

$$T = T_d . \sin(f_s - f_r) \tag{5.6}$$

Tam ve yarım adım ile mikro adım durumları karşılaştırıldığında, belirlenen açısal pozisyonlarda her iki durumda üretilen akı miktarı ve yönleri arasında bariz farklar bulunmaktadır. Belirlenen bu farklar motor performansı, karmaşıklık ve fiyat gibi nitelikleri doğrudan etkileyecektir [27].



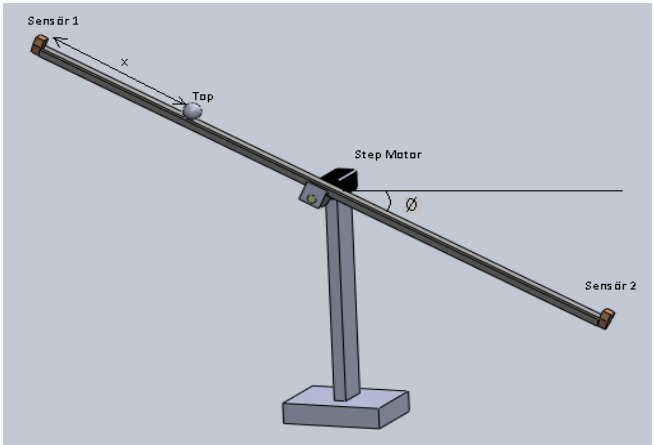
Şekil 5.8 Tam Adım ile Mikro Adım Tekniklerinin Açısal Pozisyon, Tork ve Hız Bakımından Karşılaştırılması [27]

BÖLÜM 6

SİSTEMİN MODELLENMESİ VE MALZEME SEÇİMİ

6.1. Serbest Cisim Diyagramı ve Transfer Fonksiyonu

Şekil 6.1'de gösterildiği gibi sistem üç temel mekanik bileşenden oluşmaktadır. Bunlar top, topun üzerinde yer değiştirdiği çubuk ve bu çubuğu kontrol eden step motordur. Fonksiyonu sadeleştirmek için motor mili ve çubuk rijit, merkezkaç kuvveti ise sıfır kabul edilmiştir.



Şekil 6.1 Sistem Bileşenleri Çizimi

Çubuk ekseni top temas düzlemine çok yakın olduğu ve topun çubuk üzerinde kaymadığı varsayılmıştır. Hareket denklemlerini çıkarmak için parçaların birleşim noktalarındaki kuvvetlerin hesaplanması gerekmektedir.

Öncelikle çubuk üzerinde topun durumunu değerlendirilecek olursa topa etkiyen iki adet kuvvet olduğu söylenebilir. Birincisi çubuğun açısına bağlı olarak topa etkiyen yer çekimi kuvveti ikincisi ise sürtünme katsayısına bağlı topla çubuk arasındaki sürtünme kuvvetidir.

$$\sum F_{top} = m.g.sin\emptyset - F_s \tag{6.1}$$

Topun çubuk üzerindeki pozisyonu;

$$x = \alpha.r \tag{6.2}$$

bağıntısıyla bulunur. Burada α açısal yer değiştirme r ise topun dönme ekseniyle çubukla topun temas ettiği nokta arasındaki mesafedir ve yaklaşık olarak topun yarıçapına eşittir. Topun momenti;

$$\tau_{top} = F_s.r = J_{top}.\ddot{\alpha} \tag{6.3}$$

şeklinde hesaplanır. Burada J_{top} topun eylemsizlik momentidir ve;

$$J_{top} = \frac{2}{5}.m.r^2 \tag{6.4}$$

formülüyle hesaplanır. Projede kullanılan pinpon topunun yarıçapının 0,63 cm ve kütlesinin 2 g olduğu bilinmektedir. Hesaplandığında eylemsizlik momentinin değeri $0{,}3.10^{-7} kg\ m^2$olarak bulunmuştur ve yaklaşık olarak sıfır alınabilir. Denklem (6.3)'de yerine koyduğumuzda sürtünme kuvvetinin yaklaşık sıfır olduğu ve denklem (6.1) yeniden düzenlendiğinde;

$$\sum F_{top} = m.g.sin\emptyset = 0{,}002.9{,}81.sin\emptyset = 0{,}019.sin\emptyset \tag{6.5}$$

olarak hesaplanır. Burada ∅ açısı 0^0 ile 30^0 arasında değişecektir. $sin\emptyset$ ise 0 ile 0,5 arasında değişecektir. Minimum ve maksimum açı durumlarında top üzerine etkiyen kuvvet sıfır ile 0,0095 N arasında değişecektir.

Motor çubuk sisteminin çalışma anında ürettiği tork motor kataloğundan;

$$\sum \tau_{çm} = \tau_{in} = 0{,}8Nm \tag{6.6}$$

olarak tespit edilmiştir.

Sistemin transfer fonksiyonu çıkarılırken kontrolde kullanılacak olan değişken adım sayısı ya da motorun her adımının 1,8^{o}'ye karşılık geldiği bilindiğine göre açı temel alınabilir.

$$\frac{mesafe}{açı} = \frac{x(s)}{\emptyset(s)} = \frac{g}{s^2} \tag{6.7}$$

$$T.F. = \frac{mesafe}{adım\ sayısı} = \frac{x(s)}{a.s.(s)} = \frac{g}{1{,}8.s^2} = \frac{5{,}45}{s^2} \tag{6.8}$$

Burada transfer fonksiyonunu tamamlamak için kontrolör tipi ve çalışma şartlarına da karar verilmesi gerekir. Çalışma bir deney seti olduğundan yerleşme zamanı ya da maksimum aşma gibi kısıtlamalar ile kontrolör tipi kısıtlaması yapmak uygun olmayacaktır.

6.2. Mekanik Dizayn ve Malzeme Seçimi

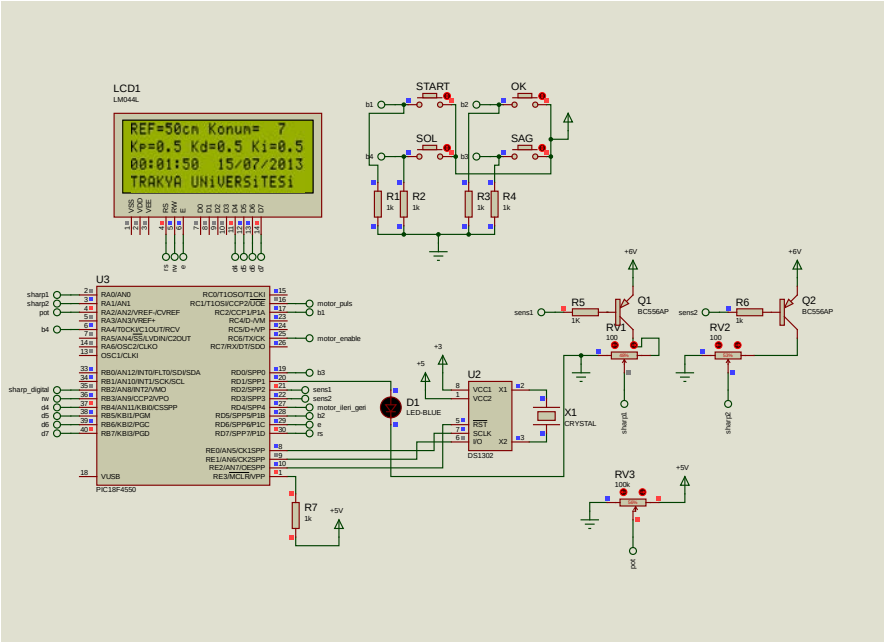
Projeye başlanırken yapılmış benzer örnekler incelenmiş ve bunların iyi ve kötü özellikleri analiz edilmiştir. Topun üzerinde hareket edeceği çubuk sensör ölü bölgeleri de dikkate alınarak 90 cm boyunda alüminyum L profil malzemeden seçilmiştir. Çubuğun ortasına açılan delik yardımıyla altına polyetilen malzemeden V kanallı bir takoz imal edilmiş ve monte edilmiştir. Takozun içinden geçecek şekilde motor miline denk bir mil imal edilmiş sonrasında bunların tamamı cıvata ile sabitlenmiştir. Bu mil bir bağlantı aparatı kullanmak suretiyle motor miline tutturulmuş ve gövde üzerine rulmanlar kullanarak yataklanmıştır. Böylece sürtünme kuvveti en aza indirilmiştir. Mekanizma ahşap gövde ile yükseltilmiş ve masa üzerinde sabit durması sağlanmıştır. Şekil 6.2'de görüldüğü gibi gövdenin alt kısmındaki bölgenin ön kısmına kontrol paneli yerleştirilmiş arka kısmına ise motor sürücü ve güç kaynağı gibi kısımlar monte edilmiştir.



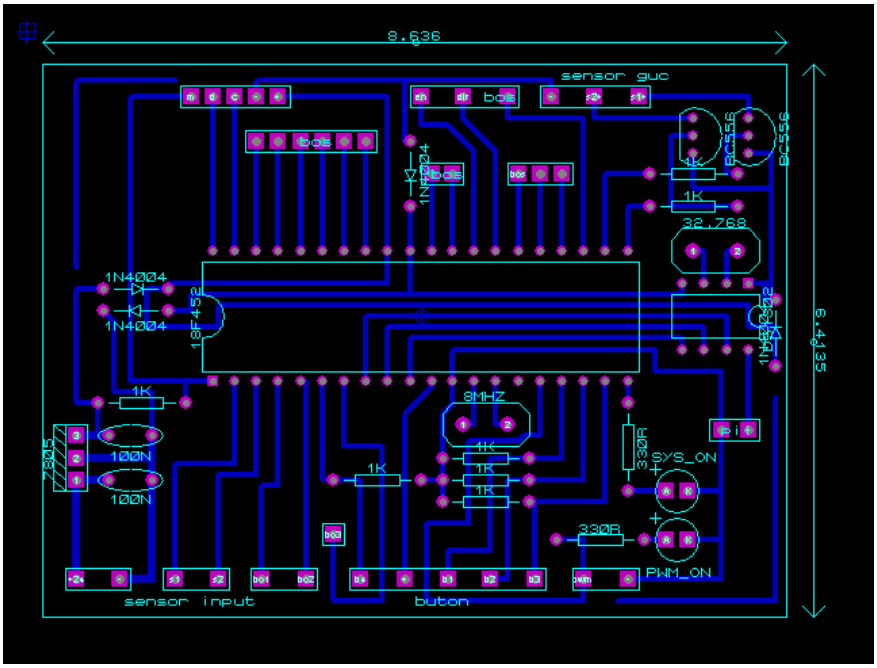
Şekil 6.2 Cihazın Görünüşü

6.3. Elektronik Dizayn ve Malzeme Seçimi

Şekil 6.3'de görüldüğü gibi öncelikle devre Proteus Isis programı yardımıyla modellenmiş ve ilgili kodlama yapılarak test edilmiştir. Modelleme işlemi bittiğinde Proteus ARES programı yardımıyla baskı devre oluşturulmuştur (Şekil 6.4).



Şekil 6.3 Proteus ISIS Programı İle Devre Modellenmesi

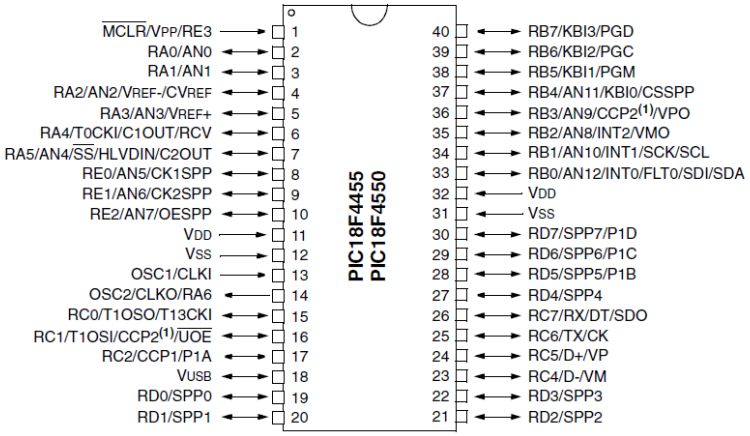


Şekil 6.4 Proteus ARES Programı İle Baskı Devre

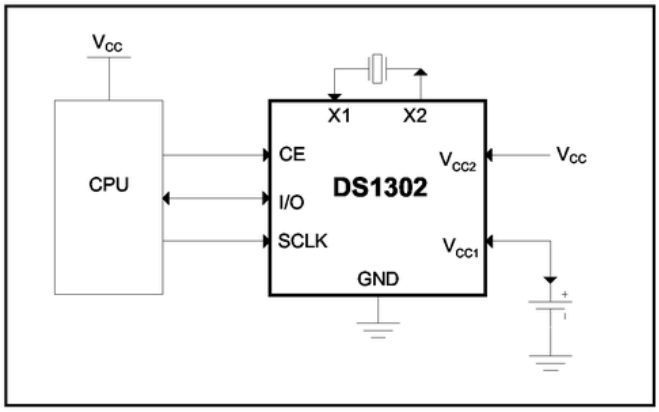
6.3.1. Mikrodenetleyici

Projede mikrodenetleyici olarak PIC18F4550 tercih edilmiştir. PIC18F4550 diğer tüm PIC18 ailesi mikrodenetleyicilerinin sahip olduğu gibi ucuz olmasının yanında yüksek hesaplama performansı ile birlikte yüksek dayanıklılığa ve geliştirilmiş flash program hafızasına sahiptir. Aynı zamanda PIC18F4550'nin tasarımında yapılan geliştirmelerle, yüksek performanslı ve güç hassasiyetli uygulamalarda seçilmesi sağlanmıştır [21].

Mikrodenetleyicinin tüm portları projede aktif olarak kullanılmıştır. Çalışma frekansı 8MHz seçilmiş ve PIC18F4550'nin dahili osilatörü kullanılmıştır. Bu mikrodenetleyici A portunda 5 adet analog girişe sahiptir ve bu girişlerin ilk ikisi kızıl ötesi sensörler için üçüncüsü ise potansiyometre için ayrılmıştır. Analog dijital dönüştürücü çözünürlüğü 10 bit olarak ayarlanmıştır. B portunda bulunan 3 adet kesme girişinden bir tanesi dijital olarak çalışan kızıl ötesi sensör için kullanılmıştır. Yazılımda sadece buradan gelen sinyalle kesmeye gidilmekte başka herhangi bir kesme kullanılmamaktadır. B portunun diğer pinleri ve D portunun bazı pinleri LCD bağlantıları için kullanılmıştır. Motor enable, yön ve darbe çıkışları ile buton bağlantıları ve cihazda tarih ve saati göstermek amacıyla kullanılan DS1302 (Şekil 6.6) entegresi diğer portlara ve boşta kalan pinlere bağlanmıştır.



Şekil 6.5 PIC18F4550 Pin Diyagramı [28]



Şekil 6.6 DS1302 Entegresi Bağlantı Şeması

6.3.2. Sensörler ve Konum Belirleme

Sensörlerden alınan sinyaller yüksek bozuluma sahip olduğu için uygun değerli kondansatör kullanılarak süzülmüştür. Alınan sinyaller mikrodenetleyicinin ADC portları kullanarak okunmaktadır. Burada 10 bitlik ADC kullanılmıştır. Buradan alınan bilgi tuş takımı yardımıyla girilen referans değerle karşılaştırılmakta ve motora gerekli sinyal uygulanmaktadır. Kullanılmış olan sensör çeşidi şekil (6.7)'de görülen Sharp GP2Y0A0 modeli kızıl ötesi mesafe sensörüdür.

Bu serinin en uzun menzilli ölçüm yapabilen sensörü olup 1.5m mesafeye kadar algılayabildiği katalog bilgisinde belirtilmiştir. Fakat sensör 30cm'den daha uzak mesafelerde çok küçük değişimlere sahip çıkış vermektedir. Bu bölgeden sonra sensör çıkışındaki gerilim değeri değişken olarak bazen artış bazen azalış göstermiştir. Bu sebeple toplam ihtiyaç duyulan mesafe iki eş parçaya bölünmüş ve bu sensörden iki adet kullanılmıştır. Hangi sensörün aktif hale geleceğine karar verilmesi adına ise çubuğun orta kısmına dijital çıkış veren standart bir kızılötesi sensör monte edilmiştir (Şekil 6.8).

Sensörlerden alınan değerler MS Excel programında grafik olarak konum-gerilim ilişkisine dönüştürülmüş ve buradan elde edilen dördüncü dereceden dönüşüm denklemleriyle okunan ADC değerine karşılık gelen mesafe değeri tespit edilmiştir. İlgili grafikler aşağıda verilmiştir. Grafikler incelendiğinde sensörlerin birbirinden farklı çıkış değişimlerine sahip oldukları görülmektedir. 1.sensöre ait dönüşüm denklemi $y = 435.x^4 - 3130.x^3 + 8405.x^2 - 10010.x + 4484$ ve 2. sensöre ait dönüşüm denklemi ise $y = 442.x^4 - 3481.x^3 + 10209.x^2 - 13253.x + 6444$ olarak bulunmuştur.



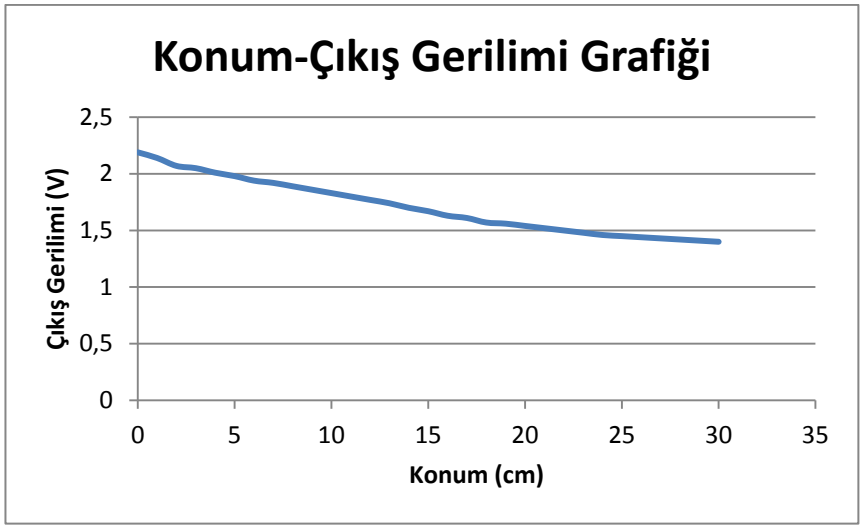
Şekil 6.7 Sharp GP2Y0AO Analog Kızılötesi Mesafe Sensörü [29]



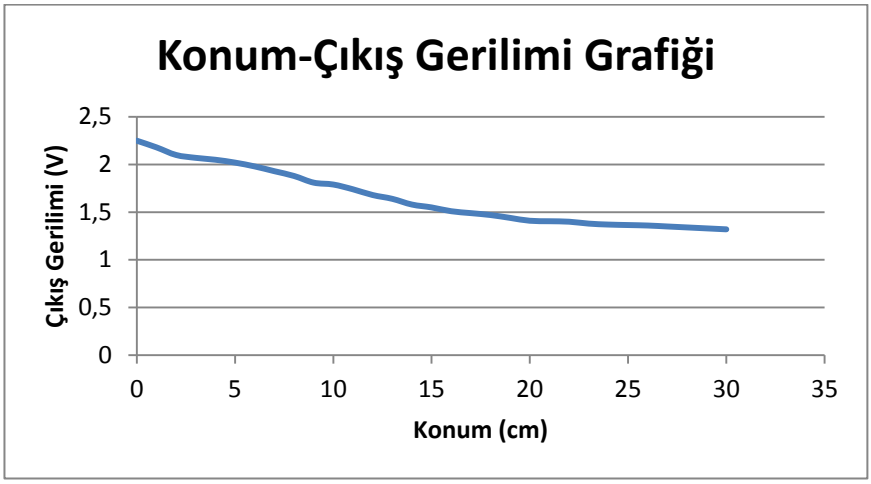
Şekil 6.8 Dijital Kızılötesi Sensör

Tablo 6.1 Sensörlerden Okunan Değerler

Konum (cm)	Gerilim (V)	Konum (cm)	Gerilim (V)	Konum (cm)	Gerilim (V)	Konum (cm)	Gerilim (V)	Konum (cm)	Gerilim (V)	Konum (cm)	Gerilim (V)
1	2,14	11	1,80	21	1,52	31	1,33	41	1,44	51	1,81
2	2,07	12	1,77	22	1,50	32	1,34	42	1,47	52	1,88
3	2,05	13	1,74	23	1,48	33	1,35	43	1,49	53	1,93
4	2,01	14	1,70	24	1,46	34	1,36	44	1,51	54	1,98
5	1,98	15	1,67	25	1,45	35	1,365	45	1,55	55	2,02
6	1,94	16	1,63	26	1,44	36	1,37	46	1,58	56	2,05
7	1,92	17	1,61	27	1,43	37	1,385	47	1,64	57	2,07
8	1,89	18	1,57	28	1,42	38	1,40	48	1,68	58	2,10
9	1,86	19	1,56	29	1,41	39	1,405	49	1,74	59	2,18
10	1,83	20	1,54	30	1,40	40	1,41	50	1,79	60	2,25

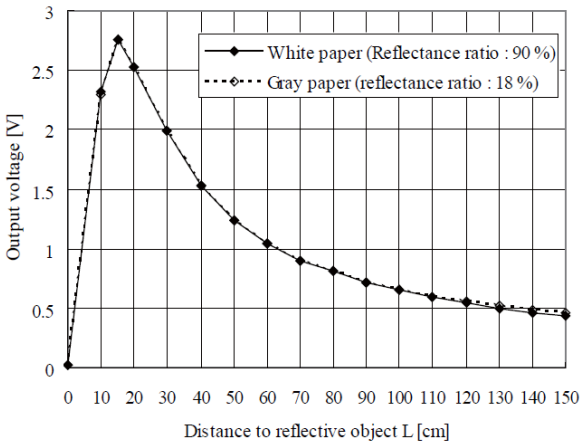


Şekil 6.9 1. Sensöre Ait Grafik



Şekil 6.10 2. Sensöre Ait Grafik

Sensörün katalog bilgilerinde verilen grafiği ise Şekil 6.11'de verilmiştir. Buradan da anlaşılabileceği gibi bu sensörler ilk 15 cm'lik mesafeyi okuyamamaktadır. Bundan dolayı uygulamada sensörler ile topun gezinebileceği menzil başlangıç bitiş noktaları arasında 15 cm'lik boşluklar bırakılmıştır. Sensörlerden alınan gerilim ile topun konumu arasındaki ilişki aşağıdaki tabloda verilmiştir.



Şekil 6.11 GP2Y0A0 Gerilim- Mesafe Grafiği [29]

6.3.3. Step Motor

Projede Nema 23 modeli hibrit step motor kullanılmıştır. Şekil 6.12'de görülen bu motor 1,8^{o}'lik adım açısına sahiptir. Bu motor 20000Hz'e kadar frekanslarda rahatlıkla sürülebilmektedir. Projede sürücüye 2000 Hz'lik kare dalga uygulanmış ve sürücü 25600 pals/devir konumunda kullanılarak yaklaşık 128 kat daha yüksek çözünürlükle, bir adım yaklaşık 0,014^{0}'ye karşılık gelecek şekilde mikro adımlama tekniğiyle motor sürülmüştür.

Topun konumu ve referans konum arasındaki fark ve ayrıca Kp, Kd, Ki katsayılarının büyüklüğüne göre motorun kaç adım atacağı MS Excel programı kullanılarak elde edilmiş dönüşüm denklemi ile hesaplanmıştır. Motorun sürülmesinde MS542 modeli hazır bir motor sürücü kullanılmıştır (Şekil 6.13). Adımlar yaklaşık olarak her 300 milisaniyede bir PID hesabının sonucu olarak sürücüye verilmektedir.



Şekil 6.12 Nema 23 Step Motor



Şekil 6.13 M542 Step Motor Sürücüsü

6.3.4. LCD Ekran ve Kontrol Paneli

Şekil 6.14'de görüldüğü üzere 4 satır 20 karaktere sahip LCD ekran kullanıcı ara yüzü olarak tercih edilmiştir. Ekran üzerinde kullanıcı tarafından seçimi yapılan referans konum, Kp, Kd, Ki gibi değişkenlerin yanı sıra topun gerçek konumu da gösterilmektedir. Ayrıca LCD'nin 3. Satırı gerçek zamanı göstermek için ayrılmıştır. Menü içerisinde gezinme ve başlatma durdurma için 4 adet buton da panele eklenmiştir.



Şekil 6.14 Kontrol Paneli

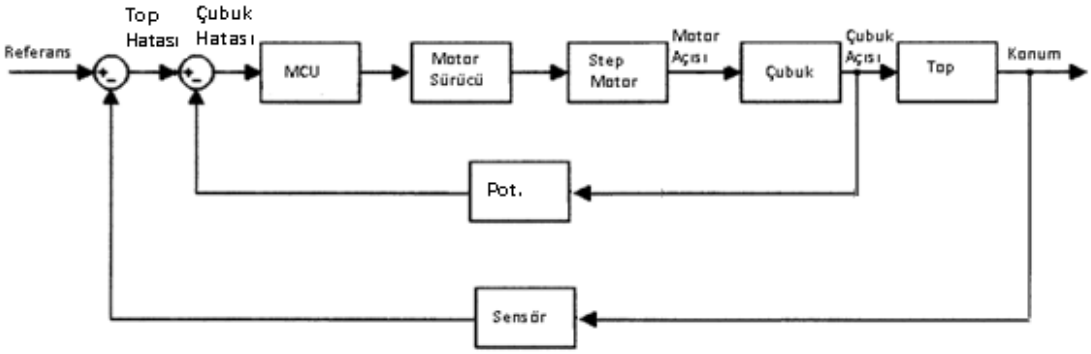
6.4. Yazılım

Mikrodenetleyicinin yazılımı CCS C kullanılarak hazırlanmıştır. Diğer muadil yazılımlara göre daha esnek ve gerçek uygulamalarda hatasız oluşu tercih sebebi olmuştur.

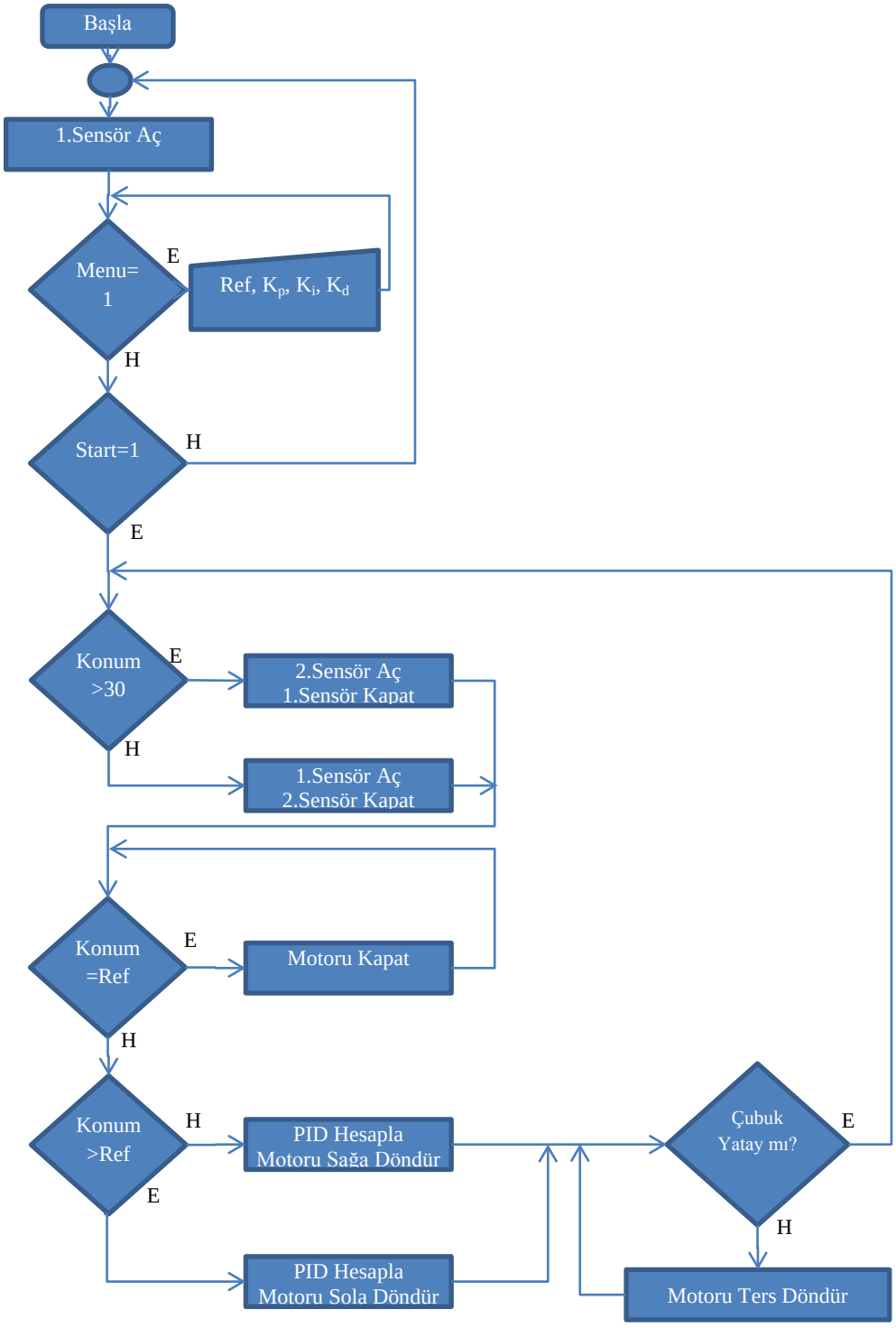
Menü tasarımı ile başlayan yazılım ADC portlarının ayarlanması, sensör okuma-dönüştürme rutini, PID hesapları, motora gerekli sinyalin gönderilmesi şeklinde devam etmektedir.

6.5. Blok Diyagram

Sistemin blok diyagramı Şekil 6.15'de görülmektedir. Burada topun konumuna bakan sensörler ve çubuğun konumuna bakan potansiyometre bulunduğu için iki kapalı döngü halinde kontrol sağlanmaktadır. Böylece katsayıların sistem davranışına etkisi daha az sayıda değişken ile net olarak gözlemlenebilmektedir. Sisteme ait sinyal akış diyagramı Şekil 6.16'da verilmiştir.



Şekil 6.15 Sistemin Blok Diyagramı



Şekil 6.16 Sinyal Akış Diyagramı

6.6. Sistemin Çalışması

Aç/Kapat (On/Off) butonuna basılması ile birlikte cihaz çalışmaya başlar. Panelde mevcut olan 4 adet buton ile tüm ayarlar yapılabilmektedir. İlk olarak Menü / Ok butonuna basılarak ana menüye giriş yapılır. Burada sırayla değiştirilebilecek olan büyüklükler İleri ve Geri butonlarına basılarak ekranda görüntülenebilir. Tekrar Menü / Ok butonuyla seçilen büyüklük İleri ve Geri butonları ile imleç kaydırılarak seçilip değiştirilebilir. Tüm ayarlamalar bittiğinde Ana Sayfa seçilerek menüden çıkılır. Burada ayarlanabilecek büyüklükler ve aralıkları Tablo 6.2'de verilmiştir. Bunların dışında saat ve tarih ayarı da menüden yapılmaktadır.

İstenilen çalışma büyüklükleri seçildikten sonra Start/ Stop butonuna basılarak motor çalıştırılır. Motor miline bağlı bulunan çubuk üzerindeki top, çubuğun açı değiştirmesi ile konum değiştirmeye başlar. Motorun dönüş yönü belirlenirken istenilen referans konum ile anlık konum karşılaştırılır. Bundan sonra ise motorun kaç adım atması gerektiğinin hesabı yapılacaktır. Bu hesap dijital PID kodu ile tespit edilmektedir. Çalışma esnasında topun çubuğun orta noktasından her geçişi sensörlerden birini aktif hale getirirken diğerinin enerjisini kesmektedir. Bu geçiş başka bir kızılötesi sensör tarafından takip edilmektedir. Çalışma esnasında sürekli olarak çubuğun konumuna da bakılmaktadır. Burada geri besleme elemanı olarak bir potansiyometre kullanılmıştır. Potansiyometre mili motor milinin ucuna bağlanmış, birlikte dönmeleri sağlanmıştır. Top istenilen referans değerini bulduğunda duracaktır. Bundan sonra sistem yeniden çalıştırılmak istenirse topun konumu referanstan uzaklaştırılmalıdır. Top durmaktayken herhangi bir bozucu giriş ile konumu değişirse yeniden aynı noktaya dönmek üzere sistem kendiliğinden çalışmaya başlayacaktır.

Tablo 6.2 Sistem Parametreleri

Büyüklük	Kısaltması	En Düşük	En Yüksek	Birim
Referans Konum	x	0	60	cm
Oransal Katsayı	Kp	0,1	0,9	-
Türevsel Katsayı	Kd	0	0,9	-
İntegral Katsayı	Ki	0	0,9	-

Tablo 6.3 Deney Seti Üzerinde Yapılan P Kontrolör Denemeleri

Kp	Yerleşme Zamanı (s)	Offset (cm)	Sonuç
0,2	36	+2	Başarılı
0,4	74	+1,5	Başarılı
0,6	11	+0,5	Başarılı
0,8	-	-	Kararsız

Tablo 6.4 Deney Seti Üzerinde Yapılan PD Kontrolör Denemeleri

Kp	Kd	Yerleşme Zamanı (s)	Offset (cm)	Sonuç
0,2	0,3	9	+9	Yüksek Kararlı Hal Hatası- Başarılı
0,2	0,6	22	+9	Yüksek Kararlı Hal Hatası- Başarılı
0,2	0,9	30	-1,5	Başarılı
0,4	0,3	15	0	Başarılı
0,4	0,6	5	-0,5	Başarılı
0,4	0,9	-	-	Kararsız
0,6	0,3	51	-2	Başarılı
0,6	0,6	118	-1	Gecikmeli- Başarılı
0,6	0,9	-	-	Kararsız
0,8	0,3	-	-	Kararsız
0,8	0,6	77	+0,5	Başarılı
0,8	0,9	107	+0,5	Gecikmeli- Başarılı

Tablo 6.5 Deney Seti Üzerinde Yapılan PI Kontrolör Denemeleri

Kp	Ki	Yerleşme Zamanı (s)	Offset (cm)	Sonuç
0,2	0,3	30	+2	Başarılı
0,2	0,6	47	-1	Başarılı
0,2	0,9	-	-	Kararsız
0,4	0,3	-	-	Kararsız
0,4	0,6	-	-	Kararsız
0,4	0,9	91	-0,5	Başarılı
0,6	0,3	-	-	Kararsız
0,6	0,6	-	-	Kararsız
0,6	0,9	-	-	Kararsız
0,8	0,3	-	-	Kararsız
0,8	0,6	-	-	Kararsız
0,8	0,9	-	-	Kararsız

Tablo 6.6 Deney Seti Üzerinde Yapılan PID Kontrolör Denemeleri

Kp	Kd	Ki	Yerleşme Zamanı (s)	Offset (cm)	Sonuç
0,2	0,3	0,3	11	+1	Başarılı
0,2	0,3	0,6	44	0	Başarılı
0,2	0,3	0,9	-	-	Kararsız
0,2	0,6	0,3	9	-2	Başarılı
0,2	0,6	0,6	27	+1	Başarılı
0,2	0,6	0,9	-	-	Kararsız
0,2	0,9	0,3	37	-0,5	Başarılı
0,2	0,9	0,6	-	-	Kararsız
0,2	0,9	0,9	20	0	Başarılı
0,4	0,3	0,3	-	-	Kararsız
0,4	0,3	0,6	-	-	Kararsız
0,4	0,3	0,9	-	-	Kararsız
0,4	0,6	0,3	15	0	Başarılı
0,4	0,6	0,6	25	0	Başarılı
0,4	0,6	0,9	-	-	Kararsız
0,4	0,9	0,3	-	-	Kararsız
0,4	0,9	0,6	29	+1	Başarılı
0,4	0,9	0,9	-	-	Kararsız
0,6	0,3	0,3	31	-0,5	Başarılı
0,6	0,3	0,6	59	-0,5	Başarılı
0,6	0,3	0,9	88	+0,5	Başarılı
0,6	0,6	0,3	-	-	Kararsız
0,6	0,6	0,6	40	-0,5	Başarılı
0,6	0,6	0,9	-	-	Kararsız
0,6	0,9	0,3	52	+0,5	Başarılı
0,6	0,9	0,6	30	+1,5	Başarılı

Kp	Kd	Ki	Yerleşme Zamanı (s)	Offset (cm)	Sonuç
0,6	0,9	0,9	-	-	Kararsız
0,8	0,3	0,3	-	-	Kararsız
0,8	0,3	0,6	-	-	Kararsız
0,8	0,3	0,9	-	-	Kararsız
0,8	0,6	0,3	131	+0,5	Gecikmeli-Başarılı
0,8	0,6	0,6	-	-	Kararsız
0,8	0,6	0,9	-	-	Kararsız
0,8	0,9	0,3	-	-	Kararsız
0,8	0,9	0,6	-	-	Kararsız
0,8	0,9	0,9	-	-	Kararsız

BÖLÜM 7

SONUÇLAR VE ÖNERİLER

Bu çalışmada otomatik kontrolün popüler problemlerinden biri olan Top-Çubuk Sistemi için yeni bir modelleme ve imalat yapılmıştır. Burada özellikle kullanılan malzemelerin önceki çalışmalardan çok farklı olması diğer örneklerden farklı bir kontrol modelinin ortaya çıkmasını zorunlu kılmıştır. Literatürde benzer çalışmalar incelendiğinde genellikle DC servo motor ya da boyut olarak daha küçük sistemler için RC servo motor kullanıldığı gözlemlenmektedir. Bu çalışmada ise bunların yerine hibrit step motor kullanılmıştır. Servo motorlar yapıları gereği kendi içlerinde geri beslemeli oldukları için kullanımları step motorlara göre daha kolaydır. Çünkü gönderilen sinyalin tam olarak gereğinin yapıldığı bilinir. Fakat step motorlarda adım kaçırma sorunu özellikle yüksek frekanslarda çok yaygındır. Ayrıca motor bir yöne hareket etmekteyken aniden ters yöne çevirme esnasında da aynı sorunla karşılaşılır. Bu sorunun üstesinden gelebilmek için sistemde iç kapalı döngü olarak potansiyometre kullanılmış ve motorun, dolayısıyla çubuğun konumu sürekli olarak ölçülmüştür. Step motor bu uygulama için çok uygundur. Çünkü çubuğun sürülmesinde ihtiyaç duyduğu mantık ile step motorun sürülme mantığı aynıdır. Fakat step motoru sürmek bir DC motor sürmekten çok daha zor ve karmaşıktır.

Malzeme seçimi açısından ikinci fark topun konumunu algılamakta kullanılan sensördür. Geçmişteki çalışmalara bakıldığında genellikle bir çubuk çifti üzerinde hareket etmekte olan metal bilyenin konumunun çubuklardan birinin üzerine sarılmış olan gerilim altındaki direnç tellerinin ilgili noktasından gerilim ölçülmesiyle yani basit bir gerilim bölücü esasıyla tespit edildiği görülmektedir. Bu çalışmada ise topun konum

tespiti için kızılötesi sensör kullanılmıştır. Sensör ile çalışmanın daha kolay olacağı düşünülse de konumu tespit edilecek malzemenin düz bir yüzeye sahip olmaması kararlılık problemini ortaya çıkarmıştır. Aynı nedenle sensör çıkış gerilimi katalogda belirtildiği gibi bir eğime sahip olamamıştır. Ayrıca potansiyometrik yapıda en fazla ikinci dereceden bir dönüşüm denklemiyle karşılaşıldığı halde burada dönüşüm denklemleri 4. veya 5. dereceden olmaktadır. Bu durum mikrodenetleyicinin işlem yükünü artırmakta ve çalışmayı olumsuz etkilemektedir. Buna önlem olarak mikrodenetleyici kodu üzerinde uzun süre iyileştirme çalışmaları yapılmıştır.

Sensör menzillerinin yetersiz kalması yeni bir problemi ortaya çıkarmış ve bu yüzden çubuğun iki ucuna monte edilen birer adet sensör kullanmak zorunda kalınmıştır. İki sensörün aynı anda enerjili olma durumunda top ile çubuk arasında milimetrik bir boşluk olsa bile birbirlerini etkiledikleri görülmüş ve buna önlem olarak sensörleri gerektiğinde devreye almak üzere topu takip eden üçüncü bir sensör kullanılmıştır. İmalata başlamadan önce yapılan araştırmalarda kızılötesi sensöre alternatif olarak kullanılabilecek iki sensör olduğu görülmüştür. Bunlardan birincisi ultrasonik sensördür. Ultrasonik sensörler ses sinyalleriyle çalışmaktadır ve bu tip çalışmalarda ortam değişkenlerinden çok fazla etkileneceği için tavsiye edilmemektedir. Diğer alternatif ise lazer sensörlerdir. Lazer sensörler 100 metreye kadar olan mesafeleri milimetrik çözünürlükte çok yüksek bir doğruluk payıyla okuyabilmektedir. Menzili ve güvenilirliği bakımından diğer sensörlerden bir hayli iyi olması fiyatının da o denli yüksek olmasına sebep olmaktadır. Sorunsuz bir çalışma için kullanılması çok faydalı olacaktır.

Çubuğun açısını takip etmekte genellikle artımsal enkoderler tercih edilmektedir. Enkoder kullanımı mikrodenetleyici için çok uygun olan dijital sinyaller üretmesi bakımından avantajlıdır. Bizim sistemimizde ise bunun yerine motor miline bağladığımız potansiyometre kullanılmıştır. Potansiyometrenin tek dezavantajı analog dijital dönüşüm zorunluluğu getirmesidir.

Geliştirilen ürün otomatik kontrol laboratuvarlarında kullanılabilecek bir deney setidir. Burada amaç topu en iyi şekilde istenilen noktaya getirmek değil oransal, türevsel ve integral katsayıların sistem davranışına olan etkisini net bir şekilde gözlemlenebilir kılmaktır. Bu yüzden çalışma bilgisayar ortamından bağımsız tutulmuş,

oluşturulan kontrol paneliyle rahatça deneme yapılabilir, gözlemlenebilir halde mikrodenetleyici kontrollü olarak sunulmuştur.

Sistem tam anlamıyla çalışır hale geldikten sonra üzerinde çok sayıda deneme yapılmış ve katsayıların etkileri gözlemlenmiştir. Denemelerde katsayılar artırılıp azaltılarak her durumda bunların etkileri gözlemlenmiştir. Yapılan denemelerin yaklaşık olarak %52'si (Tablo 6.3, 6.4, 6.5, 6.6) başarıyla sonuçlanmış dolayısıyla top kabul edilebilir bir sürede uygun bir kararlı durum hatası kalacak şekilde yerleşmiştir.

Bu denemeler esnasında P, PD, PI ve PID kontrolörler ayrı ayrı denenmiştir. P kontrolörlerden %75 oranında başarı sağlanmış ancak Kp katsayısının çok büyük seçilmesi durumunda osilasyonlar devam etmiştir (Tabo 6.3). PD kontrolör denemelerinde başarı oranı %83 olmuş oransal katsayının artışı ile bu kontrolörde de kötüleşme görülmüştür (Tablo 6.4.). PI kontrolör örneklerinde başarı yalnızca %25 olmuş, hiçbir denemede offset sıfırlanamamış ve oransal katsayı yarı değeri geçtikten sonra hiç başarılı örnek görülmemiştir (Tablo 6.5). PID kontrolör denemelerinin ise %48'i başarılı olmuş, bunların %25'inde offset sıfırlanmıştır (Tablo 6.6). Uygulamada en başarılı kontrolör yapısı PD olmuştur. Bunun sebebi integral kontrolörün yapısından kaynaklı olarak top istenilen noktaya varsa da hala kontrol işareti üretiyor olmasıdır.

Gelecekteki çalışmalarda aynı deney seti üzerinde farklı kontrolör yapıları denenmelidir. Böylece birbirlerine göre avantaj ve dezavantajları ortaya çıkacaktır. Kullanılan model özellikle havacılık ve uzay araçlarının yörünge takibi ve dış etkenlerde karşılaşacağı durumlarla çok örtüştüğünden ilgili sistem üzerinde yapılacak çalışmalar ve geliştirmelerle bu alanda önemli adımlar atılabilir.

Top- Çubuk sistemi lineer olmayan bir sistem olduğu için aslında kontrolünün de lineer olmayan metotlarla yapılması gerekmektedir. PID kontrol sistemi lineer bir kontrol sistemidir ve bu yüzden bu mekanizmanın kontrolü için en uygun metot değildir. Bulanık mantıkla kontrol, gürbüz kontrol ya da durum uzay metodu ile kontrol gibi yöntemler daha iyi sonuçlar verecektir. Kesirli üslü PID kontrolör de bu çalışma için daha uygun bir metot olabilir.

EKLER

EK-A MİKRODENETLEYİCİ CCS-C KODU

```
#include <18F4550.h>
#device adc=10
#use delay(INT=8000000)
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES INTRC_IO                 //Internal RC Osc, no CLKOUT
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOVREGEN                 //USB voltage regulator disabled
#FUSES NOSTVREN                 //Stack full/underflow will not cause reset
#FUSES NODEBUG                  //No Debug mode FOR ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18)

                                // used FOR I / O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOLPT1OSC                //Timer1 configured FOR higher power operation
#FUSES NOIESO                   //Internal External SWITCH Over mode disabled
#FUSES NOFCMEN                  //Fail-safe clock monitor disabled
#FUSES NOPBADEN                 //PORTB pins are configured as digital I/O on

                                // RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOMCLR                   //Master Clear pin used FOR I/O
#FUSES NOXINST                  //Extended set extension and Indexed Addressing

                                // mode disabled (Legacy mode)
#FUSES PLL1                     //No PLL PreScaler
#FUSES NOUSBDIV                 //USB clock source comes from primary oscillator
#FUSES CPUDIV1                  //No System Clock Postscaler
#FUSES ICPRT                    //ICPRT enabled
#include "flex_lcd.c"
#include "math.h"
#include <DS1302e.c>  // DS1302.c dosyası programa ekleniyor
#define start pin_c2
#define OK pin_d5
#define sag pin_d0
#define sol pin_a4
#define sens1 pin_d2
#define sens2 pin_d3
#define motor_ileri_geri pin_d4
#define motor_enable pin_c6
#byte UCON=0xF6D
#bit USBEN=UCON.3

byte saniye,dakika,saat,gun,ay,yil,haftanin_gunu; // byte türünde değişkenler
unsigned INT16 men=0;
float kp,kd,ki;
int1 don=0,don1=0,don2=0,sens1_on=1,sens2_on=0,start1=0, k=0, degisken;
signed INT16 p_error = 0, i_error = 0, d_error = 0, pid_error = 0, m=0;
signed INT16 error_old, hesap=0, top=0, derivative=0;
int8 saat_say, i, ref;
int16 integral=0;
unsigned LONG mesafe, konum, u;

saat_ayarlari()
{
   rtc_get_time (saat, dakika, saniye);

   lcd_gotoxy (1, 3) ;
   printf (lcd_putc, " % 02d: % 02d: % 02d", saat, dakika, saniye) ;
   rtc_get_date (gun, ay, yil, haftanin_gunu) ;
   lcd_gotoxy (11, 3) ;
   printf (lcd_putc, " % 02d / % 02d/20 % 02d", gun, ay, yil);
}

menu()
{
   IF(input(ok)==1)
   {
      men=0; i=0;
```

```
 75:       delay_ms(300);
 76:       don1=1;
 77:       lcd_init();
 78:
 79:       WHILE(don1)
 80:       {
 81:          ////////////////////////////////////
 82:          IF(input(sag)==1)
 83:          {
 84:             lcd_gotoxy(17,3);
 85:             lcd_putc("-->>");
 86:             men=men+1;
 87:             IF(men==5) men=0;
 88:             delay_ms(300);
 89:             lcd_gotoxy(17,3);
 90:             lcd_putc("    ");
 91:          }
 92:
 93:          IF(input(sol)==1)
 94:          {
 95:             lcd_gotoxy(1,3);
 96:             lcd_putc("<<--");
 97:
 98:             IF(men>0)
 99:                men = men - 1;
100:
101:             ELSE
102:             men = 5;
103:
104:             delay_ms(300);
105:             lcd_gotoxy(1,3);
106:             lcd_putc("    ");
107:          }
108:
109:          ////////////////////////////////////
110:          SWITCH(men)
111:          {
112:             CASE 0:
113:             lcd_gotoxy(1,1);
114:             lcd_putc("                ");
115:             lcd_gotoxy(1,2);
116:             lcd_putc("   **REF DEGERi** ");
117:             BREAK;
118:
119:             CASE 1:
120:             lcd_gotoxy(1,1);
121:             lcd_putc("                ");
122:             lcd_gotoxy(1,2);
123:             lcd_putc("   **KP DEGERi** ");
124:             BREAK;
125:
126:             CASE 2:
127:             lcd_gotoxy(1,1);
128:             lcd_putc("                ");
129:             lcd_gotoxy(1,2);
130:             lcd_putc("   **KD DEGERi** ");
131:             BREAK;
132:
133:             CASE 3:
134:             lcd_gotoxy(1,1);
135:             lcd_putc("                ");
136:             lcd_gotoxy(1,2);
137:             lcd_putc("   **Ki DEGERi** ");
138:             BREAK;
139:
140:             CASE 4:
141:             lcd_gotoxy(1,1);
142:             lcd_putc("                ");
143:             lcd_gotoxy(1,2);
144:             lcd_putc("      **SAAT**    ");
145:             BREAK;
146:          }
147:
148:          ////////////////////////////////////////////////////////REFERANS
```

```
149:          IF(men==0&&input(ok)==1)
150:          {
151:             delay_ms(300);
152:             lcd_init();
153:             don=1; i=0;
154:             WHILE(don)
155:             {
156:                lcd_gotoxy(2,4);
157:                lcd_putc("MENU");
158:                lcd_gotoxy(10,4);
159:                lcd_putc("ANA SAYFA");
160:                lcd_gotoxy(3,1);
161:                lcd_putc("REFERANS AYARLARI");
162:                lcd_gotoxy(1,2);
163:                printf(lcd_putc,"REFERANS=%d",ref);
164:
165:                IF(input(sag)==1)
166:                {
167:                   i++;
168:                   delay_ms(300);
169:                }
170:
171:                IF(input(sol)==1)
172:                {
173:                   i--;
174:                   delay_ms(300);
175:                }
176:
177:                IF(i>=99)
178:                {
179:                   i=2;
180:                }
181:
182:                IF(i==3)
183:                {
184:                   i=0;
185:                }
186:
187:                SWITCH(i)
188:                {
189:                   CASE 0:
190:                   lcd_gotoxy(1,4);
191:                   lcd_putc(" ");
192:                   lcd_gotoxy(10,3);
193:                   lcd_putc("^^");
194:                   //lcd_putc("^^^^");
195:                   lcd_gotoxy(9,4);
196:                   lcd_putc(" ");
197:                   BREAK;
198:
199:                   CASE 1:
200:                   lcd_gotoxy(9,4);
201:                   lcd_putc(" ");
202:                   lcd_gotoxy(10,3);
203:                   lcd_putc("    ");
204:                   lcd_gotoxy(1,4);
205:                   lcd_putc(">");
206:                   BREAK;
207:
208:                   CASE 2:
209:                   lcd_gotoxy(1,4);
210:                   lcd_putc(" ");
211:                   lcd_gotoxy(10,3);
212:                   lcd_putc("    ");
213:                   lcd_gotoxy(9,4);
214:                   lcd_putc(">");
215:                   BREAK;
216:                }
217:
218:                IF(i==0&&input(ok)==1)
219:                {
220:                   delay_ms(300);
221:                   don2=1;
222:
```

```
223:                    WHILE(don2)
224:                    {
225:                       lcd_gotoxy(1,2);
226:                       printf(lcd_putc,"REFERANS=%d",ref);
227:
228:                       IF(input(sag)==1)
229:                       {
230:                          ref=ref+1;
231:                          delay_ms(300);
232:                       }
233:
234:                       IF(input(sol)==1)
235:                       {
236:                          ref=ref-1;
237:                          delay_ms(300);
238:                       }
239:
240:                       IF(ref>=100)
241:                       {
242:                          ref=100;
243:                       }
244:
245:                       IF(input(ok)==1)
246:                       {
247:                          don2=0;
248:                          delay_ms(300);
249:                       }
250:
251:                       IF(ref<100)
252:                       {
253:                          lcd_gotoxy(12,2);
254:                          lcd_putc("  ");
255:                       }
256:
257:                       IF(ref<10)
258:                       {
259:                          lcd_gotoxy(11,2);
260:                          lcd_putc("  ");
261:                       }
262:                    }
263:                 }
264:
265:                 IF(input(ok)==1&&i==1)
266:                 {
267:                    don=0;
268:                    delay_ms(300);
269:                    lcd_init();
270:                 }
271:
272:                 IF(input(ok)==1&&i==2)
273:                 {
274:                    don=0;
275:                    don1=0;
276:                    delay_ms(300);
277:                    lcd_init();
278:                 }
279:              }
280:           }
281:
282:           ////////////////////////////////////////////////////KP AYARLARI
283:           IF(men==1&&input(ok)==1)
284:           {
285:              delay_ms(300);
286:              lcd_init();
287:              don=1;
288:              i=0;
289:
290:              WHILE(don)
291:              {
292:                 lcd_gotoxy(2,4);
293:                 lcd_putc("MENU");
294:                 lcd_gotoxy(10,4);
295:                 lcd_putc("ANA SAYFA");
296:                 lcd_gotoxy(6,1);
```

```
297:                lcd_putc("KP AYARLARI");
298:                lcd_gotoxy(1,2);
299:                printf(lcd_putc,"KP=%1.1f          ",kp);
300:
301:                IF(input(sag)==1)
302:                {
303:                   i++;
304:                   delay_ms(300);
305:                }
306:
307:                IF(input(sol)==1)
308:                {
309:                   i--;
310:                   delay_ms(300);
311:                }
312:
313:                IF(i>=99)
314:                {
315:                   i=2;
316:                }
317:
318:                IF(i==3)
319:                {
320:                   i=0;
321:                }
322:
323:                SWITCH(i)
324:                {
325:                   CASE 0:
326:                   lcd_gotoxy(1,4);
327:                   lcd_putc(" ");
328:                   lcd_gotoxy(4,3);
329:                   lcd_putc("^ ^");
330:                   lcd_gotoxy(9,4);
331:                   lcd_putc(" ");
332:                   BREAK;
333:
334:                   CASE 1:
335:                   lcd_gotoxy(9,4);
336:                   lcd_putc(" ");
337:                   lcd_gotoxy(4,3);
338:                   lcd_putc("   ");
339:                   lcd_gotoxy(1,4);
340:                   lcd_putc(">");
341:                   BREAK;
342:
343:                   CASE 2:
344:                   lcd_gotoxy(1,4);
345:                   lcd_putc(" ");
346:                   lcd_gotoxy(4,3);
347:                   lcd_putc("   ");
348:                   lcd_gotoxy(9,4);
349:                   lcd_putc(">");
350:                   BREAK;
351:                }
352:
353:                IF(i==0&&input(ok)==1)
354:                {
355:                   delay_ms(300);
356:                   don2=1;
357:
358:                   WHILE(don2)
359:                   {
360:                      lcd_gotoxy(1,2);
361:                      printf(lcd_putc,"KP=%1.1f          ",kp);
362:
363:                      IF(input(sag)==1)
364:                      {
365:                         kp=kp+0.1;
366:                         delay_ms(300);
367:                      }
368:
369:                      IF(input(sol)==1)
370:                      {
```

```
371:                        kp=kp-0.1;
372:                        delay_ms(300);
373:                     }
374:
375:                     IF(kp>=0.9)
376:                     {
377:                        kp=0.9;
378:                     }
379:
380:                     IF(kp<=0.1)
381:                     {
382:                        kp=0.1;
383:                     }
384:
385:                     IF(input(ok)==1)
386:                     {
387:                        don2=0;
388:                        delay_ms(300);
389:                     }
390:                  }
391:               }
392:
393:               IF(input(ok)==1&&i==1)
394:               {
395:                  don=0;
396:                  delay_ms(300);
397:                  lcd_init();
398:               }
399:
400:               IF(input(ok)==1&&i==2)
401:               {
402:                  don=0;
403:                  don1=0;
404:                  delay_ms(300);
405:                  lcd_init();
406:               }
407:            }
408:         }
409:
410:         ////////////////////////////////////////////////////KD DEĞER
411:         IF(men==2&&input(ok)==1)
412:         {
413:            delay_ms(300);
414:            lcd_init();
415:            don=1;
416:
417:            WHILE(don)
418:            {
419:               lcd_gotoxy(2,4);
420:               lcd_putc("MENU");
421:               lcd_gotoxy(10,4);
422:               lcd_putc("ANA SAYFA");
423:               lcd_gotoxy(6,1);
424:               lcd_putc("KD AYARLARI");
425:               lcd_gotoxy(1,2);
426:               printf(lcd_putc,"KD=%1.1f          ",kd);
427:
428:               IF(input(sag)==1)
429:               {
430:                  i++;
431:                  delay_ms(300);
432:               }
433:
434:               IF(input(sol)==1)
435:               {
436:                  i--;
437:                  delay_ms(300);
438:               }
439:
440:               IF(i>=99)
441:               {
442:                  i=2;
443:               }
444:
```

```
445:                IF(i==3)
446:                {
447:                   i=0;
448:                }
449:
450:                SWITCH(i)
451:                {
452:                   CASE 0:
453:                   lcd_gotoxy(1,4);
454:                   lcd_putc(" ");
455:                   lcd_gotoxy(4,3);
456:                   lcd_putc("^ ^");
457:                   lcd_gotoxy(9,4);
458:                   lcd_putc(" ");
459:                   BREAK;
460:
461:                   CASE 1:
462:                   lcd_gotoxy(9,4);
463:                   lcd_putc(" ");
464:                   lcd_gotoxy(4,3);
465:                   lcd_putc("   ");
466:                   lcd_gotoxy(1,4);
467:                   lcd_putc(">");
468:                   BREAK;
469:
470:                   CASE 2:
471:                   lcd_gotoxy(1,4);
472:                   lcd_putc(" ");
473:                   lcd_gotoxy(4,3);
474:                   lcd_putc("   ");
475:                   lcd_gotoxy(9,4);
476:                   lcd_putc(">");
477:                   BREAK;
478:                }
479:
480:                IF(i==0&&input(ok)==1)
481:                {
482:                   delay_ms(300);
483:                   don2=1;
484:
485:                   WHILE(don2)
486:                   {
487:                      lcd_gotoxy(1,2);
488:                      printf(lcd_putc,"KD=%1.1f         ",kd);
489:
490:                      IF(input(sag)==1)
491:                      {
492:                         kd=kd+0.1;
493:                         delay_ms(300);
494:                      }
495:
496:                      IF(input(sol)==1)
497:                      {
498:                         kd=kd-0.1;
499:                         delay_ms(300);
500:                      }
501:
502:                      IF(kd>=0.9)
503:                      {
504:                         kd=0.9;
505:                      }
506:
507:                      IF(kd<=0.0)
508:                      {
509:                         kd=0.0;
510:                      }
511:
512:                      IF(input(ok)==1)
513:                      {
514:                         don2=0;
515:                         delay_ms(300);
516:                      }
517:                   }
518:                }
```

```
519:
520:                IF(input(ok)==1&&i==1)
521:                {
522:                   don=0;
523:                   delay_ms(300);
524:                   lcd_init();
525:                }
526:
527:                IF(input(ok)==1&&i==2)
528:                {
529:                   don=0;
530:                   don1=0;
531:                   delay_ms(300);
532:                   lcd_init();
533:                }
534:             }
535:          }
536:
537:          ////////////////////////////////////////////////// Kİ AYARLARI
538:          IF(men==3&&input(ok)==1)
539:          {
540:             delay_ms(300);
541:             lcd_init();
542:             don=1;
543:
544:             WHILE(don)
545:             {
546:                lcd_gotoxy(2,4);
547:                lcd_putc("MENU");
548:                lcd_gotoxy(10,4);
549:                lcd_putc("ANA SAYFA");
550:                lcd_gotoxy(6,1);
551:                lcd_putc("Ki AYARLARI");
552:                lcd_gotoxy(1,2);
553:                printf(lcd_putc,"Ki=%1.1f         ",ki);
554:
555:                IF(input(sag)==1)
556:                {
557:                   i++;
558:                   delay_ms(300);
559:                }
560:
561:                IF(input(sol)==1)
562:                {
563:                   i--;
564:                   delay_ms(300);
565:                }
566:
567:                IF(i>=99)
568:                {
569:                   i=2;
570:                }
571:
572:                IF(i==3)
573:                {
574:                   i=0;
575:                }
576:
577:                SWITCH(i)
578:                {
579:                   CASE 0:
580:                   lcd_gotoxy(1,4);
581:                   lcd_putc(" ");
582:                   lcd_gotoxy(4,3);
583:                   lcd_putc("^ ^");
584:                   lcd_gotoxy(9,4);
585:                   lcd_putc(" ");
586:                   BREAK;
587:
588:                   CASE 1:
589:                   lcd_gotoxy(9,4);
590:                   lcd_putc(" ");
591:                   lcd_gotoxy(4,3);
592:                   lcd_putc("   ");
```

```
593:                    lcd_gotoxy(1,4);
594:                    lcd_putc(">");
595:                    BREAK;
596:
597:                    CASE 2:
598:                    lcd_gotoxy(1,4);
599:                    lcd_putc(" ");
600:                    lcd_gotoxy(4,3);
601:                    lcd_putc("    ");
602:                    lcd_gotoxy(9,4);
603:                    lcd_putc(">");
604:                    BREAK;
605:                }
606:
607:                IF(i==0&&input(ok)==1)
608:                {
609:                   delay_ms(300);
610:                   don2=1;
611:
612:                   WHILE(don2)
613:                   {
614:                      lcd_gotoxy(1,2);
615:                      printf(lcd_putc,"Ki=%1.1f          ",ki);
616:
617:                      IF(input(sag)==1)
618:                      {
619:                         ki=ki+0.1;
620:                         delay_ms(300);
621:                      }
622:
623:                      IF(input(sol)==1)
624:                      {
625:                         ki=ki-0.1;
626:                         delay_ms(300);
627:                      }
628:
629:                      IF(ki>=0.9)
630:                      {
631:                         ki=0.9;
632:                      }
633:
634:                      IF(ki<=0.0)
635:                      {
636:                         ki=0.0;
637:                      }
638:
639:                      IF(input(ok)==1)
640:                      {
641:                         don2=0;
642:                         delay_ms(300);
643:                      }
644:                   }
645:                }
646:
647:                IF(input(ok)==1&&i==1)
648:                {
649:                   don=0;
650:                   delay_ms(300);
651:                   lcd_init();
652:                }
653:
654:                IF(input(ok)==1&&i==2)
655:                {
656:                   don=0;
657:                   don1=0;
658:                   delay_ms(300);
659:                   lcd_init();
660:                }
661:             }
662:          }
663:
664:          ////////////////////////////////////////////////////////////////
665:          IF(men==4&&input(ok)==1)
666:          {
```

```
667:              delay_ms(400);
668:              lcd_init();
669:              saat_say=1;
670:              don=1;
671:
672:              WHILE(don)
673:              {
674:                 lcd_gotoxy(4,1);
675:                 lcd_putc("SAAT AYARLARI");
676:                 lcd_gotoxy(2,4);
677:                 lcd_putc("MENU");
678:                 lcd_gotoxy(10,4);
679:                 lcd_putc("ANA SAYFA");
680:                 rtc_get_time(saat,dakika,saniye);
681:                 lcd_gotoxy(1,2);
682:                 printf(lcd_putc,"%02d:%02d:%02d",saat,dakika,saniye);
683:                 rtc_get_date(gun,ay,yil,haftanin_gunu);
684:                 lcd_gotoxy(11,2);
685:                 printf(lcd_putc,"%02d/%02d/20%02d",gun,ay,yil);
686:                 lcd_gotoxy(5,2);
687:
688:                 IF(input(sag)==1)
689:                 {
690:                    saat_say++;
691:                    delay_ms(400);
692:                 }
693:
694:                 IF(input(sol)==1)
695:                 {
696:                    saat_say--;
697:                    delay_ms(400);
698:                 }
699:
700:                 IF(saat_say==9)
701:                 {
702:                    saat_say=1;
703:                 }
704:
705:                 IF(saat_say==0)
706:                 {
707:                    saat_say=8;
708:                 }
709:
710:                 SWITCH(saat_say)
711:                 {
712:                    CASE 1:
713:                    {
714:                       lcd_gotoxy(1,3);
715:                       lcd_putc("^^      ");
716:                       lcd_gotoxy(9,4);
717:                       lcd_putc(" ");
718:                       lcd_gotoxy(1,4);
719:                       lcd_putc(" ");
720:                       BREAK;
721:                    }
722:
723:                    CASE 2:
724:                    {
725:                       lcd_gotoxy(1,3);
726:                       lcd_putc("   ^^   ");
727:                       BREAK;
728:                    }
729:
730:                    CASE 3:
731:                    {
732:                       lcd_gotoxy(11,3);
733:                       lcd_putc("  ");
734:                       lcd_gotoxy(1,3);
735:                       lcd_putc("      ^^");
736:                       BREAK;
737:                    }
738:
739:                    CASE 4:
740:                    {
```

```
741:                     lcd_gotoxy(1,3);
742:                     lcd_putc("         ");
743:                     lcd_gotoxy(11,3);
744:                     lcd_putc("^^         ");
745:                     BREAK;
746:                  }
747:
748:                  CASE 5:
749:                  {
750:                     lcd_gotoxy(11,3);
751:                     lcd_putc("   ^^      ");
752:                     BREAK;
753:                  }
754:
755:                  CASE 6:
756:                  {
757:                     lcd_gotoxy(1,4);
758:                     lcd_putc(" ");
759:                     lcd_gotoxy(11,3);
760:                     lcd_putc("      ^^^^");
761:                     BREAK;
762:                  }
763:
764:                  CASE 7:
765:                  {
766:                     lcd_gotoxy(11,3);
767:                     lcd_putc("          ");
768:                     lcd_gotoxy(9,4);
769:                     lcd_putc(" ");
770:                     lcd_gotoxy(1,4);
771:                     lcd_putc(">");
772:                     BREAK;
773:                  }
774:
775:                  CASE 8:
776:                  {
777:                     lcd_gotoxy(1,3);
778:                     lcd_putc("         ");
779:                     lcd_gotoxy(1,4);
780:                     lcd_putc(" ");
781:                     lcd_gotoxy(9,4);
782:                     lcd_putc(">");
783:                     BREAK;
784:                  }
785:               }
786:
787:               ////////////////////////////////////////////////////////GÜN
788:               IF(saat_say==4&&input(ok)==1)
789:               {
790:                  delay_ms(400);
791:                  don2=1;
792:
793:                  WHILE(don2)
794:                  {
795:                     rtc_get_time (saat, dakika, saniye);
796:                     lcd_gotoxy (1, 2) ;
797:               printf (lcd_putc, " % 02d: % 02d: % 02d", saat, dakika, saniye) ;
798:                     rtc_get_date (gun, ay, yil, haftanin_gunu);
799:                     lcd_gotoxy (11, 2) ;
800:                     printf (lcd_putc, " % 02d / -- / ----", gun) ;
801:
802:                     IF(input(sag)==1)
803:                     {
804:                        gun=gun+1;
805:
806:                        IF(gun==33)
807:                        {
808:                           gun=01;
809:                        }
810:
811:                        write_ds1302(0x86,get_bcd(gun)); //saniye degeri gonder.
812:                        delay_ms(400);
813:                     }
814:
```

```
815:                     IF(input(sol)==1)
816:                     {
817:                        gun=gun-1;
818:
819:                        IF(gun==0)
820:                        {
821:                           gun=33;
822:                        }
823:
824:                        write_ds1302(0x86,get_bcd(gun)); //saniye degeri gonder.
825:                        delay_ms(400);
826:                     }
827:
828:                     IF(input(ok)==1)
829:                     {
830:                        don2=0;
831:                        delay_ms(400);
832:                     }
833:                  }
834:               }
835:
836:               ////////////////////////////////////////////////////////AY
837:               IF(saat_say==5&&input(ok)==1)
838:               {
839:                  delay_ms(400);
840:                  don2=1;
841:
842:                  WHILE(don2)
843:                  {
844:                     rtc_get_time (saat, dakika, saniye);
845:                     lcd_gotoxy (1, 2) ;
846:            printf (lcd_putc, " % 02d: % 02d: % 02d", saat, dakika, saniye) ;
847:                     rtc_get_date(gun,ay,yil,haftanin_gunu);
848:                     lcd_gotoxy (11, 2) ;
849:                     printf (lcd_putc, "-- / % 02d / ----", ay);
850:
851:                     IF(input(sag)==1)
852:                     {
853:                        ay=ay+1;
854:
855:                        IF(ay==13)
856:                        {
857:                           ay=01;
858:                        }
859:
860:                        write_ds1302(0x88,get_bcd(ay)); //saniye degeri gonder.
861:                        delay_ms(400);
862:                     }
863:
864:                     IF(input(sol)==1)
865:                     {
866:                        ay=ay-1;
867:
868:                        IF(ay==0)
869:                        {
870:                           ay=12;
871:                        }
872:
873:                        write_ds1302(0x88,get_bcd(ay)); //saniye degeri gonder.
874:                        delay_ms(400);
875:                     }
876:
877:                     IF(input(ok)==1)
878:                     {
879:                        don2=0;
880:                        delay_ms(400);
881:                     }
882:                  }
883:               }
884:
885:               ////////////////////////////////////////////////////////YIL
886:               IF(saat_say==6&&input(ok)==1)
887:               {
888:                  delay_ms(400);
```

```
889:                    don2=1;
890:
891:                    WHILE(don2)
892:                    {
893:                       rtc_get_time (saat, dakika, saniye);
894:                       lcd_gotoxy (1, 2) ;
895:                printf (lcd_putc, " % 02d: % 02d: % 02d", saat, dakika, saniye) ;
896:                       rtc_get_date(gun,ay,yil,haftanin_gunu);
897:                       lcd_gotoxy (11, 2) ;
898:                       printf (lcd_putc, "-- / -- / 20 % 02d", yil) ;
899:
900:                       IF(input(sag)==1)
901:                       {
902:                          yil=yil+1;
903:                          write_ds1302(0x8c,get_bcd(yil)); //saniye degeri gonder.
904:                          delay_ms(400);
905:                       }
906:
907:                       IF(input(sol)==1)
908:                       {
909:                          yil=yil-1;
910:                          write_ds1302(0x8c,get_bcd(yil)); //saniye degeri gonder.
911:                          delay_ms(400);
912:                       }
913:
914:                       IF(input(ok)==1)
915:                       {
916:                          don2=0;
917:                          delay_ms(400);
918:                       }
919:                    }
920:                 }
921:
922:                 ////////////////////////////////////////////////////SAAT
923:                 IF(saat_say==1&&input(ok)==1)
924:                 {
925:                    delay_ms(400);
926:                    don2=1;
927:
928:                    WHILE(don2)
929:                    {
930:                       rtc_get_time (saat, dakika, saniye);
931:                       lcd_gotoxy (1, 2) ;
932:                       printf (lcd_putc, " % 02d:--:--", saat) ;
933:
934:                       IF(input(sag)==1)
935:                       {
936:                          saat=saat+1;
937:
938:                          IF(saat==25)
939:                          {
940:                             saat=00;
941:                          }
942:
943:                          write_ds1302(0x84,get_bcd(saat)); //saniye degeri gonder
944:                          delay_ms(400);
945:                       }
946:
947:                       IF(input(sol)==1)
948:                       {
949:                          saat=saat-1;
950:
951:                          IF(saat>=99)
952:                          {
953:                             saat=24;
954:                          }
955:
956:                          write_ds1302(0x84,get_bcd(saat)); //saniye degeri gonder
957:                          delay_ms(400);
958:                       }
959:
960:                       IF(input(ok)==1)
961:                       {
962:                          don2=0;
```

```
966:                }
967:
968:                ////////////////////////////////////////////////////DAKİKA
969:                IF(saat_say==2&&input(ok)==1)
970:                {
971:                   delay_ms(400);
972:                   don2=1;
973:
974:                   WHILE(don2)
975:                   {
976:                      rtc_get_time (saat, dakika, saniye);
977:                      lcd_gotoxy (1, 2) ;
978:                      printf (lcd_putc, "--: % 02d:--", dakika) ;
979:
980:                      IF(input(sag)==1)
981:                      {
982:                         dakika=dakika+1;
983:
984:                         IF(dakika==60)
985:                         {
986:                            dakika=00;
987:                         }
988:
989:                         write_ds1302(0x82,get_bcd(dakika)); //saniye degeri
990:                         delay_ms(400);
991:                      }
992:
993:                      IF(input(sol)==1)
994:                      {
995:                         dakika=dakika-1;
996:
997:                         IF(dakika>=99)
998:                         {
999:                            dakika=59;
1000:                        }
1001:
1002:                        write_ds1302(0x82,get_bcd(dakika)); //saniye degeri
1003:                        delay_ms(400);
1004:                     }
1005:
1006:                     IF(input(ok)==1)
1007:                     {
1008:                        don2=0;
1009:                        delay_ms(400);
1010:                     }
1011:                  }
1012:               }
1013:
1014:               ////////////////////////////////////////////////////SANİYE
1015:               IF(saat_say==3&&input(ok)==1)
1016:               {
1017:                  delay_ms(400);
1018:                  don2=1;
1019:
1020:                  WHILE(don2)
1021:                  {
1022:                     rtc_get_time (saat, dakika, saniye);
1023:                     lcd_gotoxy (1, 2) ;
1024:                     printf (lcd_putc, "--:--: % 02d", saniye) ;
1025:
1026:                     IF(input(sag)==1)
1027:                     {
1028:                        saniye=saniye+1;
1029:
1030:                        IF(saniye==60)
1031:                        {
1032:                           saniye=00;
1033:                        }
1034:
1035:                        write_ds1302(0x80,get_bcd(saniye)); //saniye degeri
1036:                        delay_ms(400);
```

```
                        }

                        IF(input(sol)==1)
                        {
                           saniye=saniye-1;

                           IF(saniye==0&&saniye>60)
                           {
                              saniye=60;
                           }

                           write_ds1302(0x80,get_bcd(saniye)); //saniye degeri
                           delay_ms(400);
                        }

                        IF(input(ok)==1)
                        {
                           don2=0;
                           delay_ms(1000);
                        }
                     }
                  }

                  IF(saat_say==7&&input(ok)==1)
                  {
                     don=0;
                     delay_ms(400);
                     lcd_init();
                  }

                  IF(saat_say==8&&input(ok)==1)
                  {
                     delay_ms(400);
                     don=0;
                     don1=0;
                     lcd_init();
                  }
               }
            }
         }
      }
   }

 ////////////////////////////////////////////////////////////////
#INT_EXT2                              //Top çubuğun diğer yarısına geçtiğinde

void Ext2_interrupt()
 {
   top=1-top;                            //Top değişkenini tersle
 }

////////////////////////////////////////////////////////////////
void konumum()
{

   IF (top == 0)      //Top = 0 ise değişkenleri ayarla 1.sensörü aç
   {
      sens1_on = 1; output_high (sens1) ;
      sens2_on = 0; output_low (sens2);
   }

   IF (top == 1)      //Top = 1 ise değişkenleri ayarla 2. sensörü aç
   {
      sens1_on = 0; output_low (sens1) ;
      sens2_on = 1; output_high (sens2) ;
   }


   set_adc_channel (top);  //adc kanalı açık sensör çıkışına ayarlandı
   delay_us (10) ;
   konum = READ_ADC ();   //sensör değerini oku
   IF (sens1_on == 1)     //1. sensör açıksa mesafeyi belirle
(mesafe=435*pow(konum,4)-3130*pow(konum,3)+8405*pow(konum,2)-10010*konum+4484);
else if (sens2_on==1)
```

```
1111: (mesafe=442*pow(konum,4)-3481*pow(konum,3)+10209*pow(konum,2)-13253*konum+6444);
1112:
1113:
1114:       IF (sens2_on == 1) mesafe = 60 - mesafe; // 60 - okunan
1115:    }
1116:
1117:
1118:    lcd_gotoxy (10, 1);      //LCD'de yazılacak yere git
1119:    printf (lcd_putc, "Konum = %3lu", mesafe); //LCD'ye mesafe bilgisini gönder
1120: }
1121:
1122: ////////////////////////////////////////////////////////////////////////
1123: void hareket()
1124: {
1125:    IF (k == 1)
1126:    {
1127:       //Start tuşuna basıldıysa
1128:       konumum ();        //konum okuma kısmına git
1129:
1130:       set_adc_channel (2);      //adc kanalını pota ayarla
1131:       delay_us (10) ;
1132:       u = read_adc ();       //potansiyometreyi oku
1133:       pid_error = ref - mesafe;     //Hata hesaplanıyor
1134:       IF (pid_error < 0) pid_error = - pid_error; //Hatanın yönü önemli değil
1135:       derivative = pid_error - error_old;  //Son hata ile yeni hatanın farkı
1136:       integral = error_old + (pid_error); //Son hata ile yeni hatanın toplamı
1137:       p_error = (Kp) * pid_error * 18;    //Oransal kazanç hesaplanıyor
1138:       d_error = (Kd) * (derivative) * 18;  //Türevsel kazanç hesaplanıyor
1139:       i_error = (Ki) * (integral) * 18;  //İntegral kazanç hesaplanıyor
1140:       hesap = ( p_error + i_error + d_error); //PID kazancı hesaplanıyor
1141:       error_old = pid_error;     //Son hata güncelleniyor
1142:
1143:       IF (hesap < 0) hesap = 0;      //Alt sınır 0
1144:       IF (hesap > 118 * 3) hesap = 118 * 3;    //Üst sınır 30 derece
1145:       IF (ref < mesafe)       //Top set değerin sağındaysa
1146:       {
1147:          output_high (motor_ileri_geri); //Motor pals geldiğinde sola dönecek
1148:          output_low (motor_enable);  //Enable açık
1149:       }
1150:
1151:       IF (ref > mesafe)       //Top set değerin sağındaysa
1152:       {
1153:          output_low (motor_ileri_geri); //Motor pals geldiğinde sağa dönecek
1154:          output_low (motor_enable);  //Enable açık
1155:       }
1156:
1157:
1158:       FOR (m = 0; m <= hesap; ++m)     //PID kadar adım at
1159:       {
1160:          output_high (pin_c1) ;
1161:          delay_us (500) ;
1162:          output_low (pin_c1) ;
1163:          delay_us (500) ;
1164:       }
1165:
1166:
1167:       DO            //Çubuk yatay olana kadar
1168:       {
1169:          IF (u < 647) output_high (motor_ileri_geri); //Çubuk sağa yatıksa sola
1170:          IF (u > 647) output_low (motor_ileri_geri); //Çubuk sola yatıksa sağa
1171:          output_high (pin_c1) ;
1172:          delay_us (500);
1173:          output_low (pin_c1) ;
1174:          delay_us (500) ;
1175:          set_adc_channel (2);       //ADC çubuğa ayarlanıyor
1176:          delay_us (10) ;
1177:          u = read_adc ();        //Pot. değerini oku
1178:       }
1179:
1180:       WHILE (u != 647);         // çevir
1181:       //delay_ms (100);
1182:
1183:       DO            //Durdurma kısmı
1184:       {
```

```
1185:          konumum ();          //konumum () etiketine git
1186:          pid_error = ref - mesafe;      //ref ile mesafe karşılaş.
1187:          IF (pid_error < 0) pid_error = - pid_error;   //hata yönü önemsiz
1188:          delay_ms (5) ;
1189:       }
1190:
1191:       WHILE (pid_error < 4);        //hata 4cm'den küçükse
1192:    }
1193: }
1194:
1195: ////////////////////////////////////////////////////////////////
1196: void main()
1197: {
1198:    output_b (0x00);          //B portu low yapıldı
1199:    output_c (0x00);          //C portu low yapıldı
1200:    output_d (0x00);          //D portu low yapıldı
1201:    setup_oscillator (OSC_8MHZ);       //Osialtör frekansı ayar
1202:    enable_interrupts (GLOBAL);       //Global kesmeler açık
1203:    setup_adc_ports (AN0_TO_AN2|VSS_VDD);    //ADC PORTLARI 0 1 2 aktif
1204:    setup_adc (ADC_CLOCK_DIV_32);       //ADC clock ayarlandı
1205:    SETUP_PSP (PSP_DISABLED);        //Paralel Slave Port kapat
1206:    setup_wdt (WDT_OFF);          //Watchdog timerkapatıldı
1207:    setup_timer_1 (T1_DISABLED);       //Timer1 kapatıldı
1208:    setup_timer_3 (T3_DISABLED);       //Timer3 kapatıldı
1209:    setup_ccp2 (CCP_OFF);         // CCP2 kapatıldı
1210:    setup_ccp1 (CCP_OFF);         //CCP1 kapatıldı
1211:    disable_interrupts (INT_ad);       //ADC kesmeleri kapalı
1212:    disable_interrupts (INT_ext);       //EXT0 kesmeleri kapalı
1213:    disable_interrupts (INT_ext1);      //EXT1 kesmeleri kapalı
1214:    ENABLE_INTERRUPTS (INT_EXT2);       //EXT2 kesmeleri açık
1215:    EXT_INT_EDGE (2, L_TO_H);        //EXT2 kesmesi yükselen kn
1216:    setup_comparator (NC_NC_NC_NC);      //Karşılaştırıcı kapatıldı
1217:    rtc_init ();           // DS1302 başlangıç ayar
1218:    lcd_init ();           //LCD ayarlanıyor
1219:    sens1_on = 1;           //Başlangıçta 1.sensör on
1220:    sens2_on = 0;           //Başlangıçta 2.sensör off
1221:    ref = 50;            //Başlangıç değerleri
1222:    kp = 0.5;
1223:    kd = 0.5;
1224:    ki = 0.0;
1225:    degisken = input (pin_b2);        //B2 pini giriş
1226:    WHILE (true)           //Gidilecek etiketler
1227:    {
1228:       menu () ;
1229:       konumum () ;
1230:       hareket () ;
1231:
1232:       ust:
1233:
1234:       IF (input (start) == 1)        //Start butonu
1235:       {
1236:          delay_ms (400);        //İkinci basışsa
1237:
1238:          IF (start1 == 1)
1239:          {
1240:             output_high (pin_c6);      //enable kapalı
1241:             k = 0;
1242:             start1 = 0;
1243:             GOTO ust;
1244:          }
1245:
1246:
1247:          IF (start1 == 0)        //İlk basılışsa hareket
1248:          {
1249:             output_low (pin_c6);      //enable açık
1250:             k = 1;
1251:             start1 = 1;
1252:             GOTO ust;
1253:          }
1254:       }
1255:
1256:       saat_ayarlari ();
1257:       lcd_gotoxy (1, 1);         //LCD standart ekran
1258:       printf (lcd_putc, "REF = %dcm", ref);
```

KAYNAKLAR

[1] P. WELLSTEAD, ***http://www.control-systems-principles.co.uk/whitepapers/ball-and-beam1.pdf***, [05 06 2013 tarihinde erişilmiştir].

[2] W. Yu ve F. Ortiz, ***Stability Analysis of PD Regulation for Ball and Beam System***, Proceedings of IEEE Conference on Control Applications, pp. 517-522, 2005.

[3] Y. Jiang, C. McCorkell ve R. Zmood, ***Application of Neural Networks for Real Time Control of Ball and Beam***, Proceedings of IEEE International Conference on Neural Networks, Cilt 5, pp. 2397-2402, 1995.

[4] K. C. Ng ve M. M. Trivedi, ***Neural Integrated Fuzzy Controller(NiF-T) and Real Time Implementation of a Ball Balancing Beam(BBB)***, Proceedings of IEEE International Conference on Robotics and Automation, Cilt 2, pp. 1590-1595, 1996.

[5] M. A. Marra, B. E. Boling ve B. L. Walcott, ***Genetic Control of a Ball-Beam System***, IEEE International Conference on Control Applications, pp. 608-613, 1996.

[6] P. H. Eaton, D. V. Prokhorov ve D. C. Wunsch, ***Neurocontroller Alternatives for Fuzzy Ball and Beam Systems With Nonuniform Nonlinear Friction***, IEEE Transactions on Neural Networks, cilt 11, no. 2, pp. 423-435, 2000.

[7] H. S. Ramirez, **On the Control of the Ball and Beam System: A Trajectory Planning Approach**, Proceedings of the 39th IEEE Conference on Decision and Control, Cilt 4, pp. 4042-4047, 2000.

[8] J. Yı, N. Yubazaki ve K. Hirota, ***Stabilization Control of Ball and Beam Systems***, IFSA World Congress and 20th NAFIPS International Conference, Cilt 4, no. 9, pp. 2229-2234, 2001.

[9] F. Song ve S. M. Smith, ***Applying Incremental Best Estimate Directed Search to Optimize Fuzzy Logic Controllers for Ball and Beam Systems***, Proceedings of the

IEEE International Conference on Fuzzy Systems, Cilt 2, pp. 1132-1137, 2002.

[10] J. M. Lemos, R. N. Silva ve J. S. Marques, ***Adaptive Control of the Ball and Beam Plant in the Presence of Sensor Measure Outliers***, Proceedings of the American Control Conference, Cilt 6, pp. 4612-4613, 2002.

[11] R. M. Hirschorn, ***Incremental Sliding Mode Control of the Ball and Beam***, IEEE Traansactions on Automatic Control, Cilt 47, No. 10, pp. 1696-1700, 2002.

[12] Q. Wang, M. Mi, G. Ma ve P. Spronck, ***Evolving a Neural Controller for a Ball and Beam System***, Proceedings of the Third International Conference on Machine Learning and Cybernetings, Cilt 2, pp. 757-761, 2004.

[13] D. A. Voytsekhovsky ve R. M. Hirschorn, ***Stabilization of Single Input Nonlinear Systems Using Higher Order Compensating Sliding Mode Control***, Proceedings of the 44th IEEE Conferenceon Decision and Control and the European Control Conference, pp. 566-571, 2005.

[14] J. Huang ve W. J. Rugh, ***An Approximation Method for the Nonlinear Servomechanism Problem***, IEEE Transactions of Automatic Control, Cilt 37, No. 9, pp. 1395-1398, 1992.

[15] N. H. Jo ve J. H. Seo, ***A State Observer for Nonlinear Systems and its Application to Bball and Bam System***, IEEE Transactions of Automatic Control, Cilt 45, No. 5, pp. 968-973, 2000.

[16] I. Petroviç, M. Brezak ve R. Cupec, ***Machine Vision Based Control of the Ball and Beam***, 7th International Workshop on Advanced Motion Control, pp. 573-577, 2002.

[17] E. P. Dadios, R. Baylon, R. D. Guzman, A. Florentino, R. M. Lee ve Z. Zulueta, ***Vision Guided Ball-Beam Balancing System Using Fuzzy Logic***, 26th Annual Conference of the IEEE Industrial Electronics Society, Cilt 3, pp. 1973-1978, 2000.

[18] J. Whelan ve J. W. Ringwood, ***A Demonstration Rig for Control Systems Based on the Ball and Beam with Vision Feedback***, Proc.3rd IFAC Symposium on Control Education, *Tokyo*, 1994.

[19] S. Sridharan ve G. Sridharan, ***Ball and Beam on Roller: A New Control Labrotary Device***, Proceedings of the 2002 IEEE International Symposium on Industrial Electronics, Cilt 4, pp. 1318-1321, 2002.

[20] T. Tez, ***Manyetik Askı Sisteminde Kullanılan Kontrol Algoritmalarının Deneysel ve Teorik Araştırılması***, Yüksek Lisans Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Makine Mühendisliği Ana Bilim Dalı, 2011, p. 64.

[21] H. N. Hurma, ***PID Kontrolör ve PLC Uygulaması***, Yüksek Lisans Tezi İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, pp. 11-30, 1998.

[22] B. Kuo, Digital ***Control Systems***, Sounders College Publishing, 1992.

[23] Quin Tep;Tamagawa Seiki Co., Ltd., ***5 Phase Step Motor and Driver***, Motro Compo, No. T12-1596N2, 1989.

[24] Jones, D.W.;Microchip Technology Inc., ***Stepping Motor Fundamentals***, University of Iowa, *USA*, 2004.

[25] L. Taştan, ***Bilgisayar Yardımıyla Step Motorun Hareket Kontrolü***, Yüksek Lisans Tezi*, İstanbul Üniversitesi*, İstanbul, 1997.

[26] Final Report, ***Stepping Motor Tutorial***, Electrical Machinery, 2004.

[27] D. Uygun, ***Hibrit Adım Motorunun Sayısal Kontrolü***, Marmara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, İstanbul, 2006.

[28] Microchip Technology Inc., ***PIC18F2455/2550/4455/4550 Data Sheet***, No: DS39632C, 2005

[29] Sharp Corporation, ***Sharp GP2Y0A02YK0F Data Sheet***, No: E4-A00101EN, 2006

ÖZGEÇMİŞ

Sıtkı KOCAOĞLU 1984 yılında Ankara'da doğdu. İlk, orta ve lise öğrenimini Ankara'da tamamladı. 2002 yılında Ankara Yıldırım Beyazıt Anadolu Lisesi'nden mezun olduktan sonra aynı yıl eğitimine Selçuk Üniversitesi Mühendislik Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümü'nde devam etti. Lisans bitirme tezini "İnternet Üzerinden Akvaryum Otomasyonu" başlığıyla PIC mikrodenetleyicisi kullanarak yaptı.

Üniversite öğrenimi sırasında Botaş Genel Müdürlüğü ve Eren Elektrik A.Ş. kurumlarında staj yaptı. Mezun olduktan sonra Selen-Y Elektrik A.Ş., Kart-Sis A.Ş., Teknosis Ltd. Şti. gibi şirketlerde üretim mühendisi, proje mühendisi ve büro mühendisi olarak çalıştı.

Sıtkı KOCAOĞLU evli ve bir çocuk babasıdır.