İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SABİT KANATLI İNSANSIZ HAVA ARACININ DİKEY UÇUŞ KONTROLÜ

YÜKSEK LİSANS TEZİ

Kadir AKIN

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

OCAK 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SABİT KANATLI İNSANSIZ HAVA ARACININ DİKEY UÇUŞ KONTROLÜ

YÜKSEK LİSANS TEZİ

Kadir AKIN
(504081116)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Doç. Dr. M. Turan SÖYLEMEZ

OCAK 2012

İTÜ, Fen Bilimleri Enstitüsü'nün 504081116 numaralı Yüksek Lisans Öğrencisi **Kadir AKIN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"SABİT KANATLI İNSANSIZ HAVA ARACININ DİKEY UÇUŞ KONTROLÜ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. M. Turan SÖYLEMEZ**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. Hakan TEMELTAŞ**
İstanbul Teknik Üniversitesi

Doç. Dr. Gökhan İNALHAN
İstanbul Teknik Üniversitesi

Teslim Tarihi : **26 Aralık 2011**
Savunma Tarihi : **27 Ocak 2012**

Aileme, nişanlıma ve dostlarıma,

ÖNSÖZ

Bu çalışmada bana yol gösteren, zamanını ayıran, değerli görüşlerini benimle paylaşan hocam sayın Doc. Dr. M. Turan SÖYLEMEZ'e teşekkür ederim. Ayrıca çalışmamın her evresinde bana yardımcı olan Fatih Şekeroğlu'na Bahadır Taşkıran'a ve nişanlım N. Tuğbagül ALTAN'a teşekkürü borç bilirim.

Tüm eğitim hayatım boyunca beni destekleyen, bugünlere gelmemi sağlayan anneme, babama, kardeşlerime ve nişanlıma çalışmam süresince bana sabır gösterdikleri ve moral verdikleri için sevgi ve teşekkürlerimi sunarım.

Ocak 2012

Kadir Akın

(Kontrol Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
ÇİZELGE LİSTESİ ..... **xiii**
ŞEKİL LİSTESİ ..... **xv**
ÖZET ..... **xvii**
SUMMARY ..... **xix**
1. GİRİŞ **1**
1.1 Tezin Amacı 2
1.2 Hipotez 3
2. TERS SARKAÇ SİSTEMİNİN MODELLENMESİ VE KONTROLÜ **5**
2.1 Amaç 5
2.2 Ters Sarkaç Sisteminin Modellenmesi 5
2.3 Ters Sarkacın LQR İle Kontrolü 10
2.3.1 Durum denklemlerinin lineerleştirilmesi ve durum uzayı modeli 10
2.3.2 Sistemin kotrol edilebilirliği ve gözlenebilirliği 12
2.3.3 Sistemin kotrolü 13
2.4 Simülasyon Sonuçları 15
3. SABİT KANATLI İNSANSIZ HAVA ARACI TASARIMI **19**
3.1 SKİHA'nın Gövde Tasarımı 19
3.1.1 Servo motorlar 19
3.1.2 Fırçasız DC motor ve pervane 20
3.1.3 Elekrik hız kontrolörü 21
3.2 Batarya 22
3.3 Sürücü ve Haberleşme Kartı 22
3.4 Ataletsel Ölçüm Birimi 23
4. SABİT KANATLI İNSANSIZ HAVA ARACININ MODELLENMESİ **27**
4.1 Kuaternion Yönelim Tanımı 27
4.2 SKİHA'nın Dikey Denge Modeli 30
5. SABİT KANATLI İNSANSIZ HAVA ARACININ KONTROLÜ **37**
5.1 Uçak Eksenleri 37
5.1.1 Roll kontrol 37
5.1.2 Pitch kontrol 39
5.1.3 Yaw kontrol 39
5.2 Dikey Denge Uçuşu PD - PID Kontrolü 39
6. WEB TABANLI MİKRO DENETLEYİCİ İLE ARAYÜZ TASARIMI **49**
6.1 Olimex PIC-WEB kartını web server olarak kullanma 50
7. SONUÇLAR **55**
KAYNAKLAR **57**
EKLER **59**
ÖZGEÇMİŞ **101**

KISALTMALAR

AÖB **:** Ataletsel Ölçüm Birimi
İHA **:** İnsansız Hava Aracı
İTÜ **:** İstanbul Teknik Üniversitesi
EHK : Elektronik Hız Kontrolörü
LQR : Linear Quadratic Regulator
PWM : Pulse Width Modulation
SKİHA : Sabit Kanatlı İnsansız Hava Aracı
UAV **:** Unmanned Aerial Vehicle

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Quanser IP02 sistemi parametrelerinin değerleri. 6
Çizelge 3.1 : Sbach 342'nin özellikleri..... 19
Çizelge 3.2 : Gerilim akım ve itme kuvveti ilişkisi..... 20
Çizelge 3.3 : Kullanılan elemanların yaklaşık ağırlıkları 26

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : General atomics aeronautical systems predator 2
Şekil 1.2 : Aerovironment raven..... 2
Şekil 1.3 : Ters sarkaç ve dikey dengede sabit kanatlı İHA. 3
Şekil 2.1 : Quanser IP02 ters sarkaç sistemi..... 5
Şekil 2.2 : Ters sarkaç serbest cisim diyagramı..... 6
Şekil 2.3 : Non-lineer model..... 16
Şekil 2.4 : Matlab virtual reality toolbox ile hazırlanan arayüz 16
Şekil 2.5 : Bozucusuz konum kontrolü..... 16
Şekil 2.6 : Bozucusuz konum kontrolü sırasında sarkaç açısının değişimi 17
Şekil 2.7 : Bozuculu komun kontrolü..... 17
Şekil 2.8 : Bozuculu konum kontrolü sırasında sarkaç açısının değişimi 17
Şekil 3.1 : Hextronix HXT900 9g servo..... 19
Şekil 3.2 : Turnigy 3.7g 1370 servo 20
Şekil 3.3 : PWM kontrol sinyali servo açısı 20
Şekil 3.4 : TowerPro brushless outrunner 2410-08T 890kv..... 21
Şekil 3.5 : APC marka pervaneler 21
Şekil 3.6 : Hobbyking 25-30A ESC 21
Şekil 3.7 : Turnigy 1000mAh 3S lityum polimer batarya 22
Şekil 3.8 : Sürücü ve haberleşme kartı 22
Şekil 3.9 : Haberleşme arayüzü 23
Şekil 3.10 : Analog devices ADIS16365 AÖB 24
Şekil 3.11 : dsPIC33FJ128MC804 kontrollü AÖB..... 24
Şekil 3.12 : dsPIC33FJ128MC804 bağlantı şeması 25
Şekil 3.13 : Gerçeklenen model uçak 25
Şekil 4.1 : Refertans ve gövde eksenleri 29
Şekil 5.1 : Uçuş eksenleri 37
Şekil 5.2 : SKİHA'nın matematiksel modeli..... 40
Şekil 5.3 : SKİHA'nın dikey uçuş x ekseni modeline ait sisotool analizi..... 40
Şekil 5.4 : Sisotool ortamında elde edilen kontrolörün zaman cevabı 40
Şekil 5.5 : Simulink otomatik ayarlama özelliği kullanılarak optimize edilen PID katsayıları 41
Şekil 5.6 : Dikey uçuşta SKİHA istenilen x konumu kontrolünün birim basamk yanıtı..... 41
Şekil 5.7 : Dikey uçuş x konumu kontrolü sırasında pervanenin oluşturduğu hava akımı..... 42
Şekil 5.8 : Roll açısı PD kontrololü köklerin yer eğrisi ve frekans analizi 43
Şekil 5.9 : Roll açısı PD kontrololü zaman domeni cevabı 43
Şekil 5.10 : Sisotool roll açısı PD kontrolörü..... 44
Şekil 5.11 : Simulink otomatik ayarlamasonucu optimize edilmiş roll açısı PD kontrolörü..... 44
Şekil 5.12 : Simulink otomatik ayarlamasonucu optimize edilmiş pitch açısı PD kontrolörü..... 44

Şekil 5.13 : Simulink otomatik ayarlamasonucu optimize edilmiş yaw açısı PD kontrolörü 44
Şekil 5.14 : Simulink kontrol bloğu 45
Şekil 5.15 : Senaryo 1 kontrolör sonuçları 46
Şekil 5.16 : Senaryo 2 kontrolör sonuçları 47
Şekil 6.1 : Olimex PIC-WEB geliştirme kartı 49
Şekil 6.2 : Olimex PIC-WEB devre şeması 50
Şekil 6.3 : Oluşturulan web sitenin tüm elemanları bir klasör altına toplanmalıdır .. 51
Şekil 6.4 : Microchip MPFS generator 51
Şekil 6.5 : Micro denetleyici WEB server ana sayfa 52
Şekil 6.6 : Micro denetleyici WEB server ayarlama sayfası 52
Şekil 6.7 : Yapılması gereken IP ayarları 53

SABİT KANATLI İNSANSIZ HAVA ARACININ DİKEY UÇUŞ KONTROLÜ

ÖZET

İnsansız hava araçlarının (İHA) sınırlı kapalı mekanlarda kontrol ve uçuş planlaması bazı spesifik zorluklar içermektedir. Bu çalışmada, İHA'ların kapalı mekân otonom uçuşlarını deneysel ve teorik olarak incelenecektir.

Küçük ölçekli bir üç kanatlı bir İHA tasarlanmış ve kablolu bir bilgisayar haberleşmesi ile kapalı mekan uçuşları yapabilecek donanıma ulaşması sağlanmıştır.

Öncelikle dikey denge uçuş modelini daha iyi anlamak için, genel bir kontrol problemi olan "Ters Sarkaç" üzerinde çalışılmıştır. Serbest cisim diyagramı kullanılarak ters sarkacın matematiksel modeli hesaplanmış ve ters sarkacı kontrol etmek üzere bir optimal kontrol yöntemi olan Lineer Kuadratik Regülatör kullanılmıştır. Aynı zamanda bozuculu ve bozucusuz modellere ait sonuçlar Matlab Virtual Realty Toolbox kullanılarak gözlemlenmiştir.

Ters sarkaç sisteminden elde edilen bilgi ile tasarlanan kapalı ortam uçağının dikey denge uçuş modeli elde edilimştir. Burada geribesleme açılarında oluşacak tekillik problemini ortadan kaldırmak amacıyla kuaternion yaklaşımı kullanılmıştır.

Sonuç olarak, İHA için kontrolör optimizasyon kriterleri tanımlanmış ve araç için LQR kontrol yapısı tasarlanmıştır.

HOVER MODE FLIGHT CONTROL OF FIXED WING UNMANNED AIR VEHICLE

SUMMARY

The Unmanned aerial vehicles (UAVs) are expected to become a major part of the aviation industry over the next years, primarily enabled by developments in computer science, automatic control, communications, and sensor technologies. UAVs will likely make a major impact in surveillance and remote data collection. Examples of applications include fireground surveillance, crop and vegetation surveying, emergency data communications, and maintaining the security of people and assets against terrorist related threats. For these reasons UAV is a great interest in the research and academic circles in recent years, using it as a platform for many applications and for pure academic research.

Unmanned aerial vehicles (UAVs) are being used to perform a wide variety of missions in different environments. The ability to perform transitions between hover and flight (and vice-versa) in a confined environment enables fixed-wing UAVs to perform a multitude of missions previously limited to rotary-wing UAVs. Reconnaissance, search and rescue, surveillance, and other missions scenarios benefit from having a sensory unit that can maintain a stationary position. In an open environment, a fixed-wing UAV would fly a loiter pattern to remain on station for extended time periods. However, with added spatial constraints imposed by most urban environments, a loiter pattern may not be feasible.

In addition, using fixed-wing UAVs for sensory missions in enclosed spaces can provide additional operational advantages. For example, a fixed-wing UAV can perform quick dashes from one position to another, allowing for quick relocation. Additionally, the capability to hover enables vertical take-offs and landings, which do not require runways or facilities with large spaces. These combined factors make fixed-wing UAVs with transition capabilities ideal for missions where spatialconstraints are dominant.

The operation of unmanned aerial vehicles (UAVs) in constrained indoor environments presents many unique challenges in control and planning. This thesis investigates modeling and control methods as applied to indoor autonomous flight vehicles in both a theoretical and experimental context.

A small-scale UAV, including a custom-built three-wing tailsitter is combined with wired computer network to form a testbed capable of indoor autonomous flight.

First of all a general control problem “Inverted Pendulum” is studied to understand the hover flight model better.

The inverted pendulum system is a standard problem in the area of control systems. They are often useful to demonstrate concepts in linear control such as the stabilization of unstable systems. Since the system is inherently nonlinear, it has also been useful in illustrating some of the ideas in nonlinear control.

The inverted pendulum system is a popular demonstration of using feedback control to stabilize an open-loop unstable system. The first solution to this problem was described by Roberge in his thesis, "The Mechanical Seal." Subsequently, it has been used in many books and papers as an example of an unstable system. Siebert does a complete analysis of this system using the Routh Criterion, by multiplying out the characteristic equation as a polynomial of s and studying the coeficients. Although correct, this approach is unnecessarily abstruse. This system is the ideal root-locus analysis example.

When balancing a ruler in your hand, to move the ruler to the right, you must first move your hand sharply to the left, pointing the ruler to the right, so that when you catch the ruler, you have moved both your hand and ruler to the right. Physically, the pendulum is stabilized at a small angle from vertical, such that it always points toward the center of the track. Thus, the pendulum is always "falling" toward the center of the track, and the only possible equilibrium is a vertical pendulum in the middle of the track. If the cart is to the left of the track center, the control will stabilize the pendulum pointing to the right, such that it then falls a little more to the right. To catch the falling pendulum, the cart must move to the right (back toward the center). That motion is the desired behavior for inverted pendulum and also hover mode control of UAV.

In this system, an inverted pendulum is attached to a cart equipped with a motor that drives it along a horizontal track. The user is able to dictate the position and velocity of the cart through the motor and the track restricts the cart to movement in the horizontal direction. Sensors are attached to the cart and the pivot in order to measure the cart position and pendulum joint angle, respectively.

Measurements are taken with a quadrature encoder connected to a general purpose data acquisition and control board. Matlab/Simulink is used to implement the controller and analyze data.

The inverted pendulum system inherently has two equilibria, one of which is stable while the other is unstable. The stable equilibrium corresponds to a state in which the pendulum is pointing downwards. In the absence of any control force, the system will naturally return to this state. The stable equilibrium requires no control input to be achieved and, thus, is uninteresting from a control perspective.

The unstable equilibrium corresponds to a state in which the pendulum points strictly upwards and, thus, requires a control force to maintain this position. The basic control objective of the inverted pendulum problem is to maintain the unstable equilibrium position when the pendulum initially starts in an upright position. The control objective for this project will focus on starting from the stable equilibrium position (pendulum pointing down), swinging it up to the unstable equilibrium position (pendulum upright), and maintaining this state.

By using free body diagram of Inverted Pendulum mathematical model is computed. And Linear Quadratic Regulator which is an optimal control algoritm aplied to our sytem. Also results are observed in Matlab Virtual Realty world for with and without disturbances.

By using knowledge of Inverted Pendulum, mathematical model of our indoor plane in hower is obtained. Quaternion axis set is used to avoid singularty in feedback angles.

Finally, controller optimization criterias are defined for UAV. By using LQR control technique an optimal linear controller is designed for the vehicle.

The hover controller uses full state feedback from the sensing system, which is made possible by accurate measurement data being available at 100 Hz. In order to improve position keeping, integrators are incorporated into all four control schemes. The controller gains are optimized using linear quadratic (LQ) control techniques. Large penalties are assigned to pitch, roll, and their respective derivatives to ensure that the vehicle maintains a hover attitude. Small penalties on the position deviations prevent the vehicle from making sudden movements should the position change.

The penalty on altitude deviation is large to ensure that the airplane accurately maintains an altitude, facilitating "perching." Due to the low structural rigidity of foam airplanes, quick control surface actuation will cause the airframe to twist. This twisting moment can be perceived by the motion capture system as a movement of the airplane, adding noise to the position and attitude measurements. This undesirable effect is partially mitigated by the fact that the servos cannot instantly deflect the control surfaces. Since the servo time constants are small enough to not completely resolve the issue, the LQ design is focused on minimizing rapid changes in control surface position.

The model of the airplane in flight can be divided into four LTI systems. However, the systems are not fully decoupled. Depending on the roll angle, the rudder and elevator will influence pitch and lateral acceleration to different extents respectively.

Forward velocity of the aircraft is influenced solely by the throttle. Similarly, roll is controlled by the ailerons. Full state feedback with an integrator is used to control the forward velocity of the airplane using throttle. The control loop uses full state feedback with an integrator term to produce a desired pitch angle. Depending on the roll angle of the airplane at a specific instant, a mixing of elevator and rudder is used to get the airplane to the desired pitch attitude.

Lateral acceleration is commanded by PID controller. The lateral acceleration is produced by a mixing of elevator and rudder in the same manner as the pitch angle. The two contributions to elevator and rudder deflection are then aggregated to produce the true commands.

For roll pitch yaw control PD control algorithm is used. All PD parameters designed and calculated by using Matlab and Sisotool. Also calculated controllers optimized by ysing simulink model.

1. GİRİŞ

Günümüzde insansız hava araçları (İHA) arama ve kurtarma, haritalama, jeolojik keşif, askeri keşif akıllı bilgi toplama gibi birçok zorlayıcı görevde kullanılmaktadır. Bu araçların insansız uçabilmeleri sayesinde tehlikeli görevler yapabilmeleri yanında üretim maliyetlerinin de insanlı hava araçlarına nazaran düşük olması yaygınlaşmalarını sağlamaktadır. Genellikle uzaktan kontrollü ve otonom olmak üzere iki tip İHA üretilmektedir. Uzaktan kontrollü İHA'lar bir pilot tarafından uzaktan konrol edilirken otonom İHA'lar ise uçuş öncesi programlandıkları görevi yaparlar. Otonom İHA'ların yapıları uzaktan kontrollü İHA'lara nazaran daha karmaşıktır.

Şekil 1.1'de görülmekte olan resim 2000 yılında üretilmiş olan "General Atomics Aeronautical Systems Predator" a aittir. Çokça terçih edilen bu İHA 20 m kanat açıklığına, 4536 kg ağırlığa sahip, 50000 ft yükseklik için tasarlanmış 30 saatten fazla uçabilen, askeri standartları sağlayan bir araçtır. Şekil 1.2'de görülmekte olan AeroVironment Raven isimli araç ise daha alçak uçuş yapan hafif ve küçük bir modeldir. Bu araç hem uzaktan kontrollü olarak hemde programlanabilir otonom olarak çalışabilmektedir.

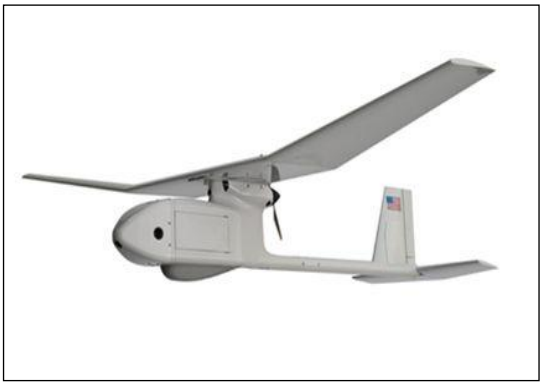
Sabit kanatlı İHA'larırn normal uçuş ve dikey dengede durma modları arasında geçiş yapabilmeleri döner kanatlı İHA'lara göre avantaj sağlamaktadır. Sabit kanatlı İHA'lar açık ortamda normal uçuş sayesinde hızlı hareket ederlerken, kapalı ortamlarda dikey denge durumuna geçerek hasas görevleri yerine getirebilirler. Ayrıca dikey denge modu, dikey kalkış inişe olanak sağlar böylece kalkış ve inişte geniş alanlara ihtiyaç duyulmaz [1].

Sabit kanatlı İHA'ların normal uçuş - dikey denge arası geçiş yetenekleri kullanım alanlarını ve faydalarını arttırmıştır. İlk başarılı manuel kontrollü geçişler 1954 yılında Convair XFY-1 isimli araç ile yapıldı. Sonraki yıllarda da araç geliştirmek için çalışan pek çok araştırma grupları vardır. Drexel üniversitesinde geliştirilmiş olan radyo sinyalleri ile haberleşen insan kontrollü araç yatay uçuş yapabildiği gibi

üzerindeki işlemci ve ataletsel ölçüm ünitesi sayesinde dikey dengede de durabilmektedir [2].



Şekil 1.1 : General atomics aeronautical systems predator



Şekil 1.2 : Aerovironment raven.

Aracın genel anlamda işlevselliğinin artması için yüksek seviyede uçuş ve dikey salınım yapması gerekmektedir. Yapılan çalışmalar genelikle kapalı alan uçuş denemeleri ile sınırlı kalmıştır. Ayrıca yapılan denemler de kalkışlar ve inişler için ya özel tasarlanmış bir platform tasarlanmış yada el ile sağlanmıştır bu da değişik fiziksel çevrelerde meydana gelebilecek iniş kalkış sorunlarını kapsamamaktadır [3].

1.1 Tezin Amacı

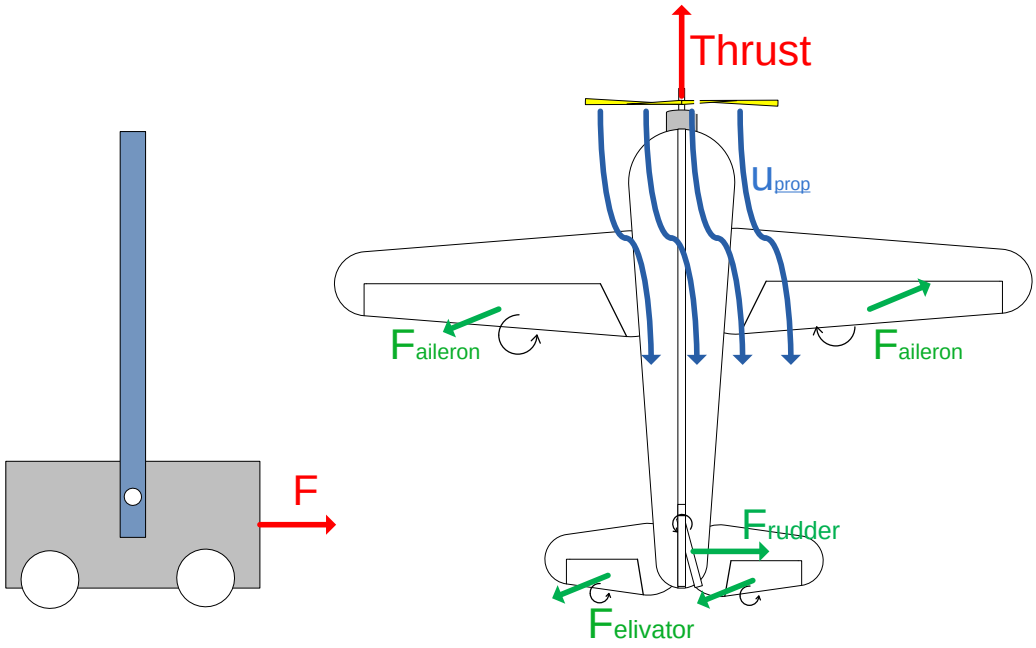
Bu çalışmada kontrol mühendisliğinin temel problemlerinden biri olan ters sarkacın dikey dengede durmasından ilham alınarak bir sabit kanatlı insansız hava aracının dikey olarak dengede durması sağlayacak kontrolör tasarlanmış ve gerçeklenmiştir. İHA'nın yatay olarak dengede durabilmesi için uygun bir kontrolör tasarlamak amacıyla ters sarkaçın modeli elde edilmiş ve elde edilen modele yönelik

kontrolörler tasarlanmıştır. Daha sonra kapalı ortamlarda uçabilecek bir İHA üretilmiş ve yatay uçuş ve dikey denge modelleri elde edilmiş, ters sarkaçın kontrolünde kullanılan kontrolör yapıları uygulanmıştır.

1.2 Hipotez

Ters sarkaç düzeneği lineer olmayan bir sistemdir ve bu sistemin kontrolü kontrol mühendisliğinin temel problemlerinden biridir. Özellikle robotik sistemlerde eklemlerin modellenmesi ve kontrolü sırasında karşılaşın non-lineerliğe benzer non-lineer terimler içermesi ters sarkaç sistemini günümüzde de üzerinde çalışılan konulardan biri haline getirmektedir.

Sabit kanatlı bir İHA'nın dikey denge modeli temel anlamda ters sarkaç modeliyle paralellik göstermektedir. Şekil 1.3'te de görüldüğü gibi en basit anlamda denge durumundaki İHA üst ucuna yay bağlanmış iki eksenli bir ters sarkaç gibi düşünülebilir. Bu durumda ters sarkaçın dikey dengede tutulması için tasarlanan kontrolör yapıları SKİHA için de kullanılabilir.



Şekil 1.3 : Ters sarkaç ve dikey dengede sabit kanatlı İHA.

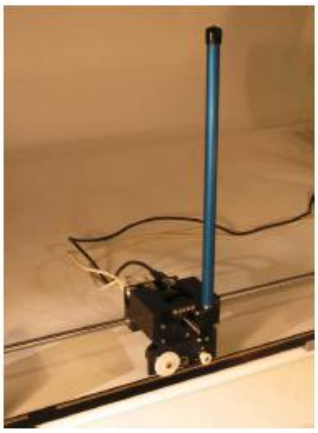
2. TERS SARKAÇ SİSTEMİNİN MODELLENMESİ VE KONTROLÜ

2.1 Amaç

Bir aracın üzerine monte edilmiş ters sarkacı serbest konumdan yukarı denge konumu civarına getirmek ve sarkacı dik olarak dengede tutmak amaçlanmıştır. Sarkacın kararsız denge konumunda dik olarak tutulabilmesi için sürekli aktif olarak kontrol edilmesi gerekmektedir [4].

2.2 Ters Sarkaç Sisteminin Modellenmesi

Şekil 2.1' de görülmekte olan Quanser firmasın IP02 ters sarkaç deney setinin donanımsal yapısı ve parametreleri göz önünde bulundurularak sistem modellenmiştir.



Şekil 2.1 : Quanser IP02 ters sarkaç sistemi

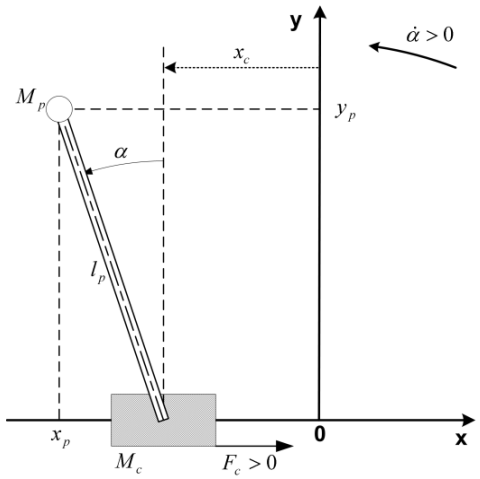
IP02 ters sarkaç sistemine ilişkin parametreler ve değerleri Çizelge 2.1'de verilmiştir.

Sistem dinamiklerini tanımlayan denklemleri bulabilmek için enerji tabanlı lagrangian yöntemi kullanılmıştır [5,6]. Öncelikle sistemdeki kinetik ve potansiyel enerji hesaplanarak lagrangian bulunur. Lagrangian kullanılarak sistemin hareket denklemleri elde edilebilir.

Çizelge 2.1 : Quanser IP02 sistemi parametrelerinin değerleri.

Sembol	Tanım	Değer	Birim
V_{nom}	Motor Nominal Giriş Gerilimi	6.0	V
$f_{V\max}$	Maksimum Motor Giriş Frekansı	50	Hz
R_m	Motor Armatür Direnci	2.6	Ohm
L_m	Motor Armatür Endüktansı	0.18	mH
K_t	Motor Tork Sabiti	0.00767	N.m/A
η_m	Motor Kazanç Oranı	100	%
K_m	Ters EMK Sabiti	0.00767	V.s/rad
J_m	Rotor Eylemsizlik Momenti	3.9E-007	kg. m^2
K_g	Dişli Kutusu Oranı	3.71	
η_g	Dişli Kutusu Kazanç Oranı	100	%
M_{c2}	IP02 Araç Kütlesi	0.57	kg
M_w	IP02 Araç Ek Ağırlığının Kütlesi	0.37	kg
r_{mp}	Motor Dişlisinin Yarıçapı	0.0063	m
B_{eq}	Motor dişlisi için eşdeğer viskos sürtünme katsayısı	5.4	
M_p	Sarkaç Kütlesi	0.127	kg
L_p	Sarkaçın Tüm Uzunluğu	0.3365	m
l_p	Sarkaçın Ağırlık Merkezi Uzunluğu	0.1778	m
I_p	Sarkaç Atalet Momenti	0.0012	kg. m^2
B_p	Sarkaç Sürtünmesi	0.0024	N.m.s/rad

Ters sarkaç sistemin serbest cisim diyagramı Şekil 2.2'de görülmektedir. Ters sarkaç sisteminin bileşenleri araç ve sarkaç şeklinde ayrıştırılırsa model daha anlaşılır olacaktır. Öncelikle araca ait öteleme ve dönme kinetik enerjileri aşağıdaki gibidir.



Şekil 2.2 : Ters sarkaç serbest cisim diyagramı

$$EK_{ct} = \frac{1}{2} M_c \left(\frac{d}{dt} x_c(t) \right)^2 \quad \textbf{(2.1)}$$

$$EK_{cr} = \frac{1}{2} \frac{J_m \, K_g^2 \left(\frac{d}{dt} x_c(t) \right)^2}{r_{mp}^2} \quad \textbf{(2.2)}$$

$$EK_c = EK_{ct} + EK_{cr} = \frac{1}{2} \left(M_c + \frac{J_m \, K_g^2}{r_{mp}^2} \right) \left(\frac{d}{dt} x_c(t) \right)^2 \quad \textbf{(2.3)}$$

Denkelm 2.3'ü basitleştirmek amacıyla $M = M_c + \frac{J_m \, K_g^2}{r_{mp}^2}$ yazılabilir. Sarkaca ait öteleme ve dönme kinetik enerjileri ise aşağıda verilmiştir.

$$EK_{pt} = \frac{1}{2} M_p \left(\left(\frac{d}{dt} x_p(t) \right)^2 + \left(\frac{d}{dt} y_p(t) \right)^2 \right) \quad \textbf{(2.4)}$$

$$x_p(t) = x_c(t) - l_p \sin \alpha(t) \; ; \quad \frac{d}{dt} x_p(t) = \frac{d}{dt} x_c(t) - l_p \cos \alpha(t) \frac{d}{dt} \alpha(t) \quad \textbf{(2.5)}$$

$$y_p(t) = l_p \cos \alpha(t) \; ; \quad \frac{d}{dt} y_p(t) = -l_p \sin \alpha(t) \frac{d}{dt} \alpha(t) \quad \textbf{(2.6)}$$

$$EK_{pr} = \frac{1}{2} I_p \left(\frac{d}{dt} \alpha_c(t) \right)^2 \quad \textbf{(2.7)}$$

$$\begin{aligned} EK_p &= EK_{pt} + EK_{pr} \\ &= \frac{1}{2} M_p \left(\frac{d}{dt} x_c(t) \right)^2 - M_p l_p \cos \alpha(t) \frac{d}{dt} \alpha(t) \frac{d}{dt} x_c(t) \\ &+ \frac{1}{2} \left(I_p + M_p l_p^2 \right) \left(\frac{d}{dt} \alpha_c(t) \right)^2 \end{aligned} \quad \textbf{(2.8)}$$

Sonuç olarak sistemin toplam kinetik enerjisi $EK = EK_c + EK_p$ denklem 2.9'da verilmiştir.

$$
\begin{aligned}
EK_p &= EK_c + EK_p \\
&= \frac{1}{2}\left(M + M_p\right)\left(\frac{d}{dt}x_c(t)\right)^2 - M_p l_p \cos\alpha(t)\frac{d}{dt}\alpha(t)\frac{d}{dt}x_c(t) \\
&+ \frac{1}{2}\left(I_p + M_p l_p^2\right)\left(\frac{d}{dt}\alpha_c(t)\right)^2
\end{aligned}
\tag{2.9}
$$

Şekil 2.2'de görüldüğü üzere aracın ağırlık merkezi x-ekseni üzerinde seçilmiştir bu durumda sistemin toplam potansiyel enerjisi sarkaçın potansiyel enerjisine eşit olacaktır.

$$
EP = EP_p = M_p g l_p \cos\alpha(t) \tag{2.10}
$$

Sistemin toplam kinetik ve potansiyel enerjisi kullanılarak lagrangian denklemi elde edilir (2.11).

$$
\begin{aligned}
L &= EK - EP \\
&= \frac{1}{2}\left(M + M_p\right)\left(\frac{d}{dt}x_c(t)\right)^2 - M_p l_p \cos\alpha(t)\frac{d}{dt}\alpha(t)\frac{d}{dt}x_c(t) \\
&+ \frac{1}{2}\left(I_p + M_p l_p^2\right)\left(\frac{d}{dt}\alpha_c(t)\right)^2 - M_p g l_p \cos\alpha(t)
\end{aligned}
\tag{2.11}
$$

Lagrangian denklemi kullanılarak sistemin durum denklemleri aşağıdaki gibi hesaplanır.

$$
F_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_\iota}\right) - \frac{\partial L}{\partial \dot{x}_\iota} \tag{2.12}
$$

Sistemin durumları $x_d = \begin{bmatrix} x_c \\ \alpha_p \end{bmatrix}$ aracın konumu x_c ve sarkacın açısı α_p olarak seçilmesi durumunda 2.12'deki genel lagrangian eşitliği kullanılarak durum denklemleri 2.13 ve 2.14'te görüldüğü gibi hesaplanır.

$$F_{x_c} = (M + M_p)\left(\frac{d^2}{dt^2}x_c(t)\right) - M_p l_p \cos\alpha_p(t)\left(\frac{d^2}{dt^2}\alpha_p(t)\right) + M_p l_p \sin\alpha_p(t)\left(\frac{d}{dt}\alpha_p(t)\right)^2 \quad \textbf{(2.13)}$$

$$\tau_{\alpha_p} = -M_p l_p \cos\alpha_p(t)\left(\frac{d^2}{dt^2}x_c(t)\right) + \left(I_p + M_p l_p^2\right)\left(\frac{d^2}{dt^2}\alpha_p(t)\right) - M_p g l_p \sin\alpha_p(t) \quad \textbf{(2.14)}$$

Yukarıdaki denklemlerde görülmekte olan F_{x_c} ve τ_{α_p} değerleri sisteme uygulanan kuvvet ve tork değerlerini göstermektedir. Burada F_{x_c} sisteme motor dan aktarılan yatay kuvvetten oluşan yatay sürtünmelerin çıkartılmış halini τ_{α_p} ise α_p boyunca herhangi bir dış tork etkilemediği için yalnızca sarkaçta oluşan dönme sürtünmelerini ifade etmektedir. Bu kuvvet ve tork ifadeleri aşağıdaki gibi formülize edilebilir.

$$F_{x_c} = F_c - B_x\left(\frac{d}{dt}x_c(t)\right) \quad \textbf{(2.15)}$$

$$\tau_{\alpha_p} = -B_\alpha\left(\frac{d}{dt}\alpha_p(t)\right) \quad \textbf{(2.16)}$$

2.13 ile 2.15 ve 2.14 ile 2.16 denkelemleri eşitlenip düzenlenirse aşağıdaki non-lineer durum denklemleri elde edilir.

$$\ddot{x_c} = \frac{-\left(I_p + M_p l_p^2\right)B_x\dot{x_c} - \left(I_p M_p l_p + {M_p}^2 l_p^3\right)\sin\alpha_p(t)\left(\dot{\alpha_p}\right)^2}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2 + {M_p}^2 l_p^2 \sin^2\alpha_p(t)} + \frac{-M_p l_p \cos\alpha_p(t)B_p\,\dot{\alpha_p} + \left(I_p + M_p l_p^2\right)F_c + {M_p}^2 l_p^2 g\cos\alpha_p(t)B_p\sin\alpha_p(t)}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2 + {M_p}^2 l_p^2 \sin^2\alpha_p(t)} \quad \textbf{(2.17)}$$

$$\ddot{\alpha_p} = \frac{-M_p l_p B_x \cos\alpha_p(t)\,\dot{x_c} - {M_p}^2 l_p^2 \cos\alpha_p(t) B_p \sin\alpha_p(t)\left(\dot{\alpha_p}\right)^2}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2 + {M_p}^2 l_p^2 \sin^2\alpha_p(t)}$$

$$+\frac{-\left(M_c + M_p\right)B_p\,\dot{\alpha_p} + \left(M_c M_p g l_p + {M_p}^2 g l_p\right)\sin\alpha_p(t) + M_p l_p \cos\alpha_p(t) F_c}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2 + {M_p}^2 l_p^2 \sin^2\alpha_p(t)} \quad \textbf{(2.18)}$$

2.3 Ters Sarkacın LQR İle Kontrolü

2.3.1 Durum denklemlerinin lineerleştirilmesi ve durum uzayı modeli

Lineer kontrol yöntemlerinin uygulanabilmesi için öncelikle (2.16) ve (2.17)'de bulunan ifadelerin lineerleştirilmesi gerekmektedir. Sarkacın çalışma bölgesi olan $\alpha \approx 0$ civarında sistem lineerleştirilmelidir. $\alpha \approx 0$ için:

$$\cos(\alpha) \approx 0\ ,$$

$$\sin(\alpha) \approx \alpha\ ,$$

$$\left(\frac{d}{dt}\,\alpha(t)\right)^2 = 0\ ,$$

yazılabilir. Bu yaklaşıklıklar kullanılarak (2.16) ve (2.17) eşitlikleri yeniden düzenlenirse lineerleştirilmiş hareket denklemleri

$$\ddot{x_c} = \frac{-\left(I_p + M_p l_p^2\right)B_{eq}\dot{x_c} - M_p l_p B_p \dot{\alpha_p} + M_p^2 l_p^2 g \alpha_p + \left(I_p + M_p l_p^2\right)F_c}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2} \quad \textbf{(2.19)}$$

$$\ddot{\alpha_p} = \frac{-B_{eq} M_p l_p \dot{x_c} - \left(M_c + M_p\right)B_p \dot{\alpha_p} + M_p l_p g\left(M_c + M_p\right)\alpha_p + M_p l_p F_c}{\left(M_c + M_p\right)I_p + M_c M_p l_p^2} \quad \textbf{(2.20)}$$

olarak elde edilir.

Sisteme ait durumlar lineerleştirildikten sonra aşağıdaki ifadeler kullanılarak durum uzayında gösterime geçilebilir:

$$\frac{d}{dt}X = AX + BU \quad \textbf{(2.20)}$$

$$Y = CX + DU \tag{2.21}$$

(2.18) ve (2.19)'dan görüleceği üzere sistemin durumları olarak aracın konumu, sarkacın açısı, aracın hızı ve sarkacın açısal hızı alınması gerekmektedir. Bu durumda Durum vektörü X (2.22) de gösterilmiştir.

$$X = \begin{bmatrix} x_c(t) & \alpha_p(t) & \frac{d}{dx}x_c(t) & \frac{d}{dx}\alpha_p(t) \end{bmatrix} \tag{2.22}$$

(2.18) ve (2.19) ifadeleri kullanılarak A,B,C ve D matrisleri aşağıdaki gibi bulunur.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{M_p^2 l_p^2 g}{(M_c + M_p)I_p + M_c M_p l_p^2} & \frac{-(I_p + M_p l_p^2)B_{eq}}{(M_c + M_p)I_p + M_c M_p l_p^2} & \frac{-M_p l_p B_p}{(M_c + M_p)I_p + M_c M_p l_p^2} \\ 0 & \frac{M_p l_p g(M_c + M_p)}{(M_c + M_p)I_p + M_c M_p l_p^2} & \frac{-B_{eq} M_p l_p}{(M_c + M_p)I_p + M_c M_p l_p^2} & \frac{-(M_c + M_p)B_p}{(M_c + M_p)I_p + M_c M_p l_p^2} \end{bmatrix} \tag{2.23}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{I_p + M_p l_p^2}{(M_c + M_p)I_p + M_c M_p l_p^2} \\ \frac{M_p l_p}{(M_c + M_p)I_p + M_c M_p l_p^2} \end{bmatrix} \tag{2.24}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.26}$$

2.3.2 Sistemin kotrol edilebilirliği ve gözlenebilirliği

Bir sistemi herhangi bir $x_i \in \mathbb{R}^n$ başalangıç durumundan $x_f \in \mathbb{R}^n$ son durumuna sonlu bir T süresinde getirebilecek bir kontrol işareti $u(t)$ bulunabiliyorsa sistem kontrol edilebilirdir [7].

$$x_f = e^{AT}x_i + \int_0^T e^{A(T-s)}Bu(s)ds \qquad \textbf{(2.27)}$$

(2.28) de ifade edilen kontrol edilebilirlik matrisinin rankının durum vektörünün boyutu olan *n*'e eşit olması sistemin kontrol edilebir olduğu anlamına gelmektedir.

$$Kont = [B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B] \qquad \textbf{(2.28)}$$

Sisteme belli bir zaman aralığında etki eden *u(t)* kontrol işareti ve *y(t)* çıkış işaretine bakılarak başlangıç durumu hesaplanabiliyor ise bu sistem gözlenebilirdir denir [8].

$$y(t) = Ce^{AT}x(0) + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t)\,; \qquad \forall 0 \le t \le T, \qquad \textbf{(2.29)}$$

Bir sistemin gözlenebilirliğini belirlemek için (2.30) da verilen gözlenebilirlik matrisinin rankına bakılır, rankın durum vektörünün boyutu olan *n*'e eşit olması sistemin gözlenebilir olduğu anlamına gelir.

$$Gözl = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \qquad \textbf{(2.30)}$$

Sistemin durum matrislerinin sayısal değerleri aşağıda verilmiştir.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.6013 & -9.0121 & -0.0173 \\ 0 & 49.4271 & -39.0353 & -0.5355 \end{bmatrix} \qquad \textbf{(2.31)}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1.6689 \\ 7.2288 \end{bmatrix} \quad \textbf{(2.32)}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.33)}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \textbf{(2.34)}$$

Sonuç olarak sistem kontrol edilebilirdir. Sistem durumlarından araç konumu x_c gözlenebilir, sarkaç açısı α_p ise gözlenebilir değildir.

2.3.3 Sistemin kotrolü

Sarkacın dik konumda tutulması oldukça hassas bir kontrol gerektirmektedir. Sistem yapısı gereği kararsız olduğundan dolayı ters sarkaç sistemi tasarlanan yeni kontrolör yapılarının ne kadar etkili olduğunu denemek amacıyla sıklıkla kullanılır. Sarkacın açı değişimlerine gerektiğinden fazla tepki vermesi çok küçük bozucu etkilerinde bile sarkacın konumunun değişmesine neden olacaktır. Açı değişimlerine çok yavaş cevap verildiğinde ise sarkaç dik konumda tutulamayacaktır. Bu nedenlerden dolayı bu bölümde sarkacı yukarı denge konumunda tutabilmek ve bu noktada oluşacak bozucu etkilerini bastırabilmek için bir optimal kontrol yöntemi olan "Linear Quadratic Regulator" kontrol yapısı kullanılacaktır.

LQR yöntemi ile yapılmak istenen aslında belirlenen bir performans ölçütünü minimize eden kontrolörü belirlemektir [9]. Örneğin performans ölçütü olarak

$$J_{LQR} = \int_0^\infty \|x(t)\|^2 + \rho\|u(t)\|^2 dt \quad \textbf{(2.35)}$$

seçilebilir. Bu performans ölçütü ile kontrol edilen durumların karesel integrali $\int_0^\infty \|x(t)\|^2\, dt$ ve sisteme uygulanan kontrol işaretinin karesel integrali

$\int_0^\infty \|u(t)\|^2\, dt$ nin ağırlıklandırılmış toplamı minimize edilmek istemektedir. Burada bir anlamda $\int_0^\infty \|x(t)\|^2\, dt$ ifadesi kontrol edilen durumların enerjisini, $\int_0^\infty \|u(t)\|^2\, dt$ ifadesi de kontrol işaretinin enerjisini göstermektedir. Diğer bir deyişle bu tarz bir kontrol yapısı ile sistem durumlarının minimum enerji ile istenen değerlerine ulaşması amaçlanmaktadır.

Tasarımda LQR için daha genel bir performans ölçütü olan (2.35) ifadesini kullanacağız.

$$J_{LQR} = \int_0^\infty x(t)^T Q\, x(t) + \rho u(t)^T R u(t) dt \qquad \textbf{(2.36)}$$

En genel halde Q simetrik negatif olmayan tanımlı ve R simetik pozitif tanımlı matrislerdir. Bu matrislerin belirlenmesi için Bryson kuralını kullanacağımızdan dolayı Q matrisi diagonal matris olarak elde edilecektir. R ise sistem bir girişi olduğu için skaler olacaktır.

Burada $u = -Kx$ olarak seçilecektir. Böylece amacımız performans ölçütünü minimize elden K kazanç vektörünü belirleme problemine indirgenmiş olur. K vektörünün analitik yolla hesaplanabilmesi için Matris Riccati Denklemi çözülmedir.

K kazanç vektörünün belirlenebilmesi için öncelikle Q ve R 'nin belirlenmesi gerekir. Yukarıda da değinildiği gibi burada Bryson kuralı uygulanacaktır. Bu kural bazı durumlarda iyi sonuç verse de genellikle yeterli değildir ancak bir başlangıç noktası bulma görevi görür. Bu kurala göre

$$Q_{ii} = \frac{1}{x_i^{2\prime} nin\ maximum\ kabul\ edilen\ değerleri} \qquad \textbf{(2.29)}$$

$$R_{ii} = \frac{1}{u_i^{2\prime} nin\ maximum\ kabul\ edilen\ değerleri} \qquad \textbf{(2.30)}$$

ifadeleriyle Q ve R matrisleri belirlenebilir [10, 11]. Bu aşamada hangi durumun ve kontrol işaretinin maksimum değişim aralığını belirlememiz gerekir. Sistemimizde

sarkacın üzerinde hareket ettiği dişli sistemin uzunluğu yaklaşık olarak 1m'dir. Yine sistemin lineerleştirme hatalarının kabul edilebilir olması için sarkaç açısının $\pm 25°$ arasında olması öngörülmüştür.

$$x_{max} = 1\text{m}$$

$$\alpha_{max} = 25^o = 0.4363\, rad.$$

$$u_{max} = 50\text{V};$$

Bu değerler kullanılarak Q ve R matrisleri

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5.2525 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R = [4 \cdot 10^{-4}]$$

olarak belirlenebilir. Bu değerler belirlendikten sonra K kazanç vektörünün belirlenebilmesi için matris Riccati denklemi çözülmelidir. Matlab programı kullanılarak hesaplanan kaznç değeri aşağıda verilmiştir.

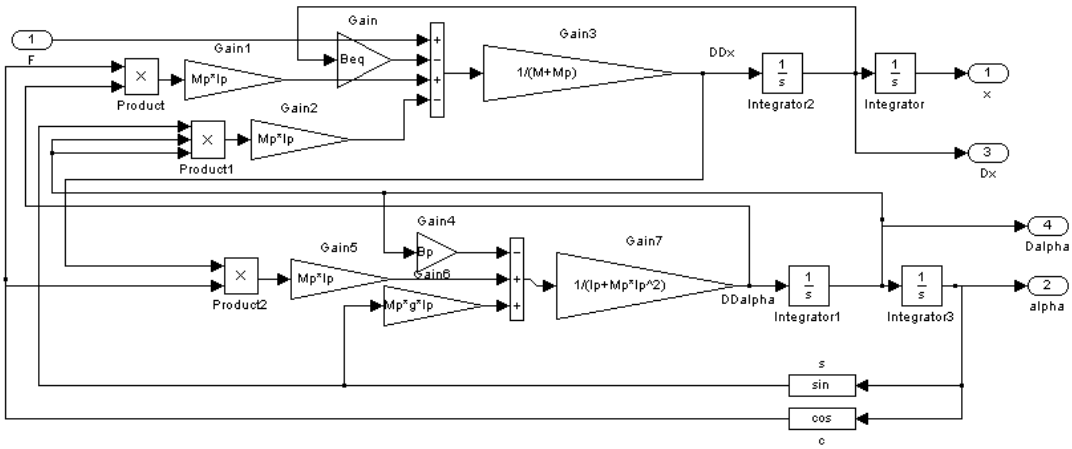
K = [-50.0000 157.4838 -44.9575 15.4461]

2.4 Simülasyon Sonuçları

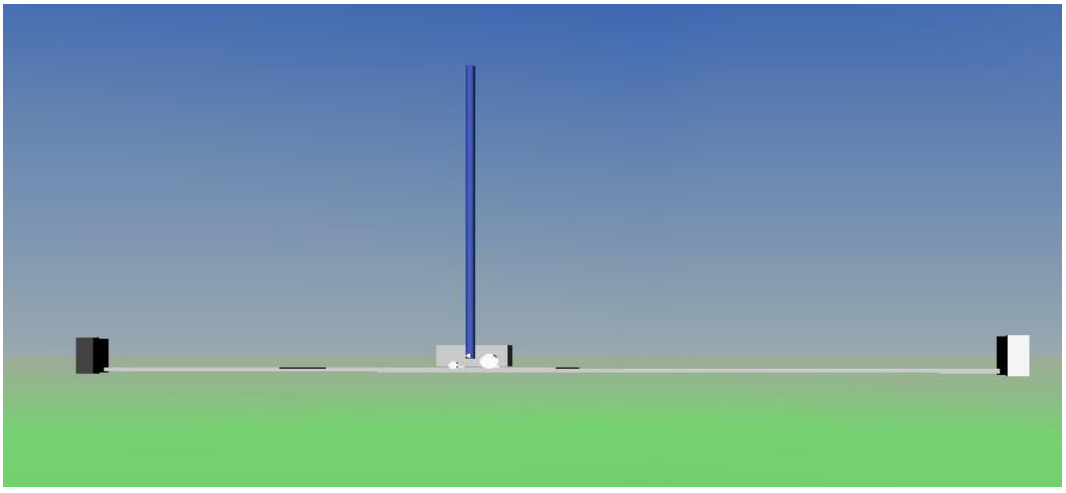
Matlab Simulin kullanılarak oluşturulan sistem modeli Şekil 2.3'de görülmektedir. Ayrıca Matlab Virtuel Reality Toolbox kullanılarak Şekil 2.4 görülen üç boyutlu arayüz hazırlanmıştır.

Sisteme herhangi bir bozucu etki etmediği durumda sistemin konum cevabı ve sarkaç açısı Şekil 2.5 v4 Şekil 2.6'da görüldüğü gibi olmuştur.

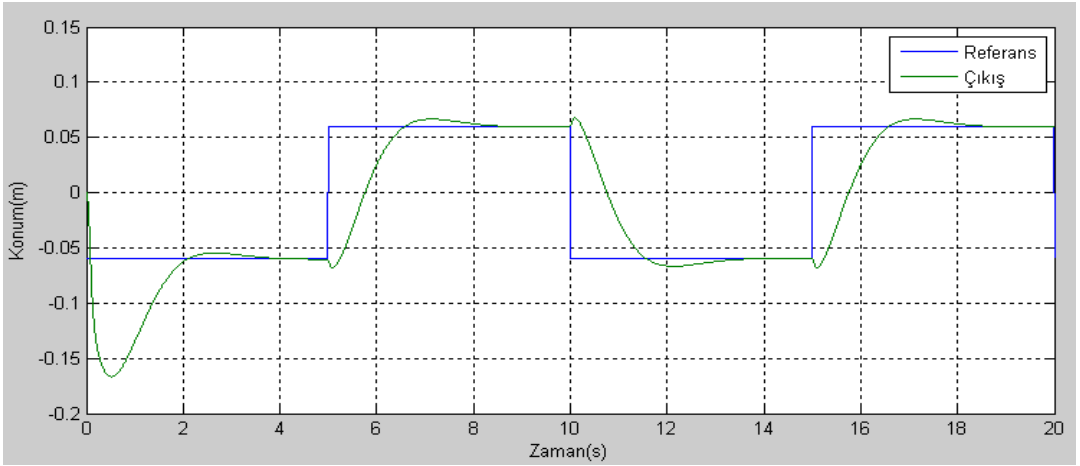
Sisteme 4. Saniyede bir bozucu verildiği durumda ise sistemin konum cevabı ve sarkaç açısı Şekil 2.7 ve Şekil 2.8'de görüldüğü gibi olmuştur.



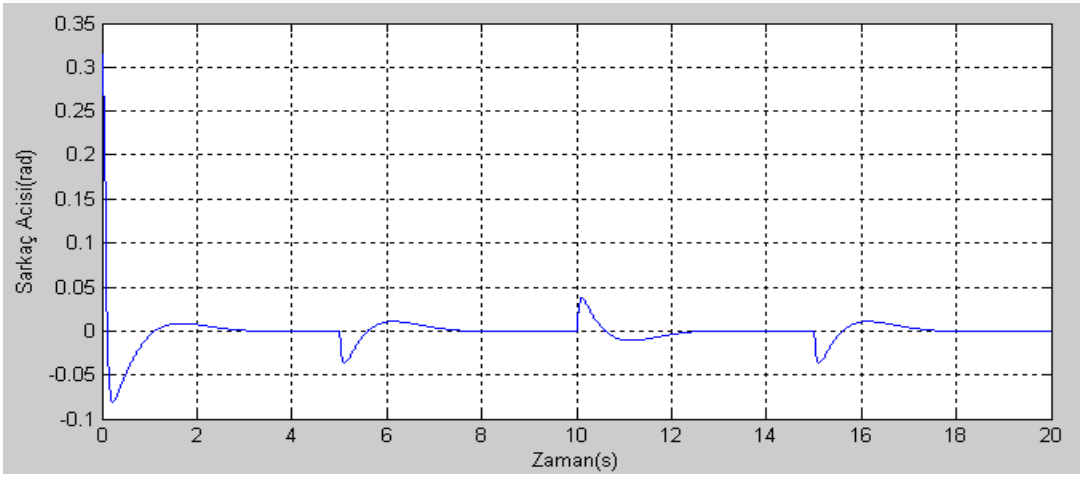
Şekil 2.3 : Non-lineer model



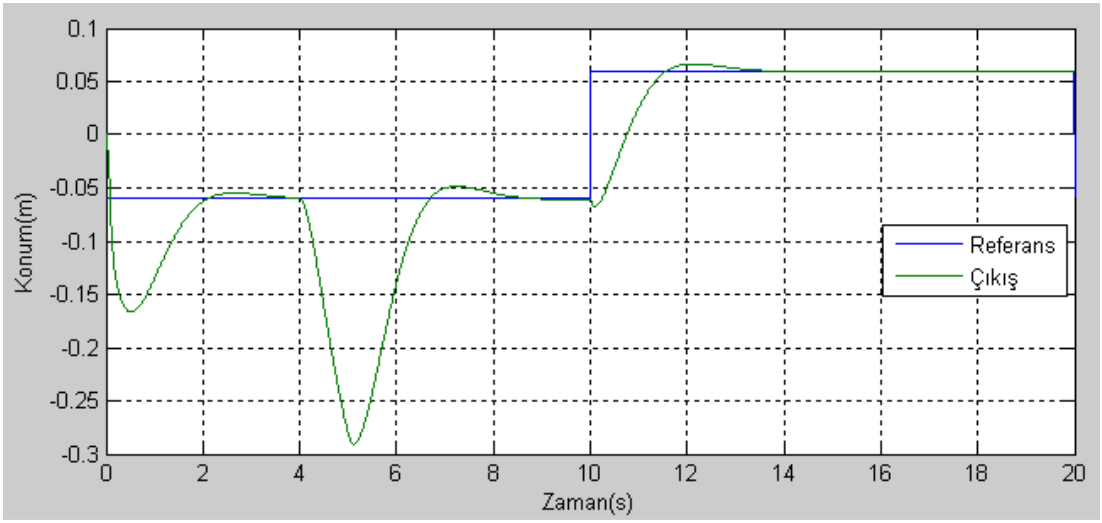
Şekil 2.4 : Matlab virtual reality toolbox ile hazırlanan arayüz



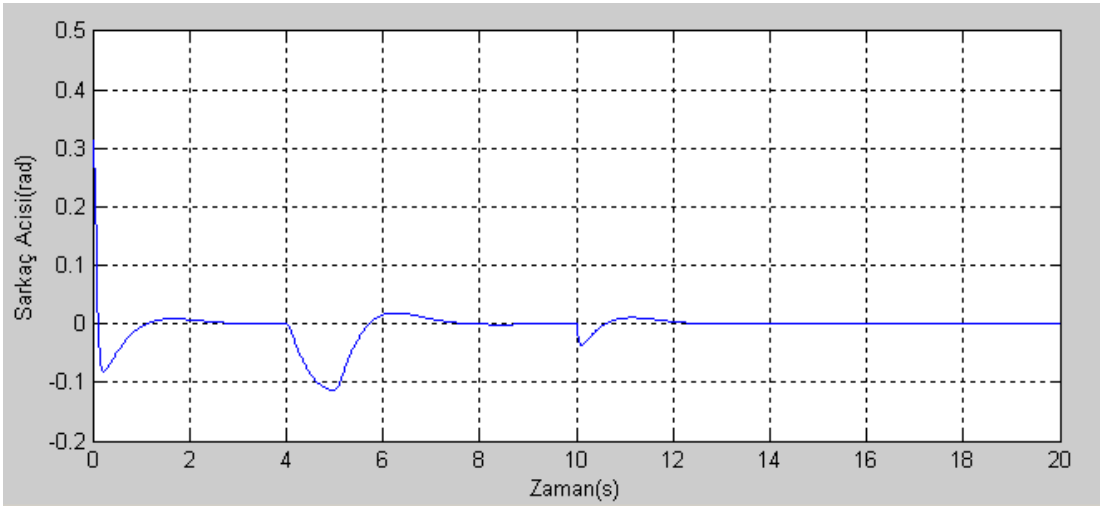
Şekil 2.5 : Bozucusuz konum kontrolü



Şekil 2.6 : Bozucusuz konum kontrolü sırasında sarkaç açısının değişimi



Şekil 2.7 : Bozuculu komun kontrolü



Şekil 2.8 : Bozuculu konum kontrolü sırasında sarkaç açısının değişimi

3. SABİT KANATLI İNSANSIZ HAVA ARACI TASARIMI

3.1 SKİHA'nın Gövde Tasarımı

Tasarım amacıyla bir çok kapalı ortam uçak modeli incelenmiştir. Sonuç olarak genel olarak depron köpük ile gerçeklenen hobi uçakların başlangıç düzeyinde kontrol yapısını denemeye uygun olacağı görülmüştür. SKİHA'da gövde tasarımında daha önce uçuş denemeleri yapılmış olan bir model olan Sbach 342 kullanılmıştır [12]. Bu model 6mm'lik eppron ile gerçeklenmiştir. Model'e ait özellikler Çizelge 3.1'de verilmitir.

Çizelge 3.1 : Sbach 342'nin özellikleri

Sembol	Tanım	Değer	Birim
w_{sp}	Kanat Açıklığı	90	cm
L_p	Uzunluk	95	cm
M_p	Ağırlık	0.127	kg

3.1.1 Servo motorlar

Eppron köpük kullanarak üretilen uçak modelinde kanatçık açıklıklarını kontrol etmesi amacıyla dört adet servo motor kullanılmıştır. Bu servo motorların ikisi eleronların kontrolünde kullanılmıştır. Hextronix firmasını 9 g'lık HXT900 model numaralı bu motorlar Şekil 3.1'de görülmektedir. Uçağın kuyruk kısmında bulunan kaldırıcı ve dümen kanatçıklarının kontrolünde ise Şekil 3.2'de görülmekte olan Turnigy firmasının 3.7g'lık 1370A servoları kullanılmıştır.

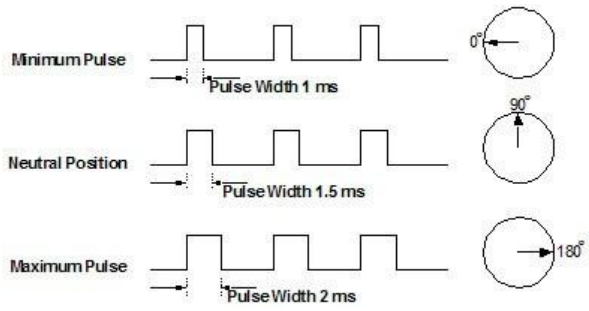


Şekil 3.1 : Hextronix HXT900 9g servo



Şekil 3.2 : Turnigy 3.7g 1370 servo

Bu servo motorların her ikisi de DC 3.6V-6V ile çalışabilen 50Hz'lik PWM sinyali ile kontrol edilen servolardır. Kontrol sinyali ile servonun açısı arasındaki lineer ilşki Şekil 3.3'te görülmektedir [13].



Şekil 3.3 : PWM kontrol sinyali servo açısı

3.1.2 Fırçasız DC motor ve pervane

Fırçasız DC motor uçağı uçuracak ve yatay denge durumunda tutacak olan itiş kuvvetini oluşturacak ana motordur. Seçilmiş olan fırçasız DC motor

3.4'te görülmekte olan "TowerPro Brushless Outrunner 2410-08T 890kv" motorudur. Bu motorun gerilim akım ve oluşturduğu itme kuvvetine ait veriler Çizelge 3.2'de görülmektedir.

Çizelge 3.2 : Gerilim akım ve itme kuvveti ilişkisi

Pevane Tipi	Gerilim	Akım	İtme Kuvveti
1047	7.4V	6.0A	344.4 g
1047	8.4V	7.3A	411.6 g
1047	9.6V	8.6A	492.8 g
1047	11.1V	10.9A	602 g



Şekil 3.4 : TowerPro brushless outrunner 2410-08T 890kv

Çizelge 3.2'de görülen itme kuvveti değerini arttırmak amacıyla 20 cm çapında pervane kullanmak yerine Şekil 3.5'te görülen 30 cm'lik APC marka pervane kullanılmıştır.



Şekil 3.5 : APC marka pervaneler

3.1.3 Elekrik hız kontrolörü

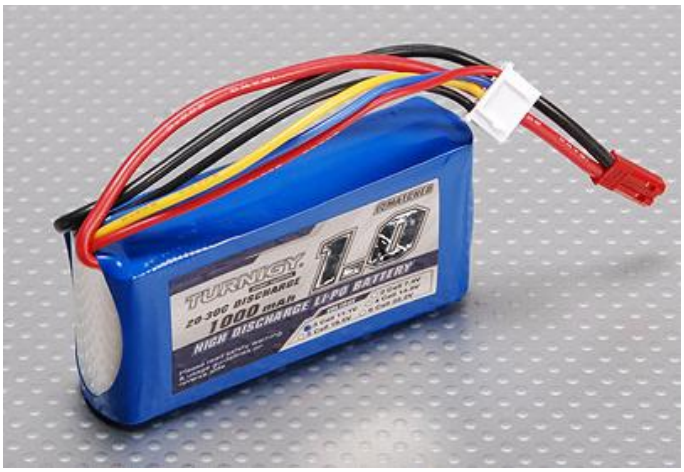
Fırçasız DC motorun kontrol edilmesi için 25-30A'lik HobbyKing firmasından Şekil 3.6'da görülen elektrik hız kontrolörü (EHK) kullanılmıştır. Bu elemanın da kontrol işareti servo motorlarda olduğu gibi 50Hz'lik PWM'dir. Servo motorlardan farklı olarak bu eleman kontrol edilirken güvenliği sağlamak adına öncelikle EHK'ya minimum PWM verilir. Bu durumda EHK üzerindeki bir buzzer uzun bir bip sesi çıkararak hazır olduğunu bildirir. Daha sonra PWM çalışma oranı arttırılarak motorun hızlanması sağlanır.



Şekil 3.6 : Hobbyking 25-30A ESC

3.2 Batarya

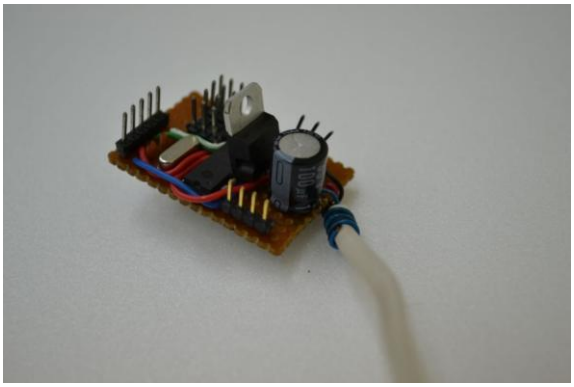
İHA'da bulunan tüm elektronik ekipmanı ve motorları beslemek için 1000mAh'lik üç hücreli lityum polimer batarya kullanılmıştır.Şekil 3.7'de görülmekte olan üç hücreli bataryada çıkış gerilimi 11.1V'tur.



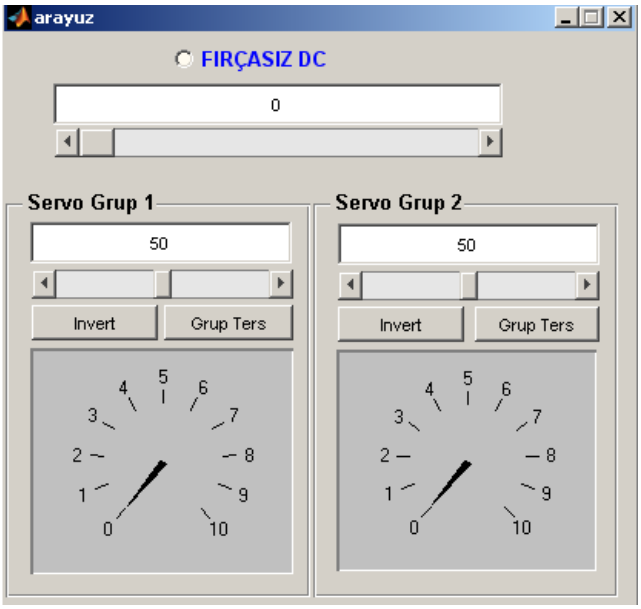
Şekil 3.7 : Turnigy 1000mAh 3S lityum polimer batarya

3.3 Sürücü ve Haberleşme Kartı

Sistemdeki servo motorları sürmek ve bilgisayar ile haberleşmek amacıyla Şekil 3.8'de görülen kart tasarlanmıştır. Pertinaks üzerinde hazırlanan bu devrede bulunan PIC16F628A mikrodenetleyicisi üzerinden 20 ms periyotla 5 adet PWM üretilmektedir. Bu 5 PWM işareti ile dört servo ve EHK kontrol sürülmektedir. Aynı zamanda RS232 seri iletişim ile servo ve fırçasız motorlar hazırlanan arayüz üzerinden manuel olarak kontrol edilebilmektedir. Matlab'da hazırlanan arayüze ait bir görünüm Şekil 3.9'de verilmiştir ayrıca arayüze ve mikro denetliyiciye ait kaynak kod EK.A'da mevcuttur.



Şekil 3.8 : Sürücü ve haberleşme kartı



Şekil 3.9 : Haberleşme arayüzü

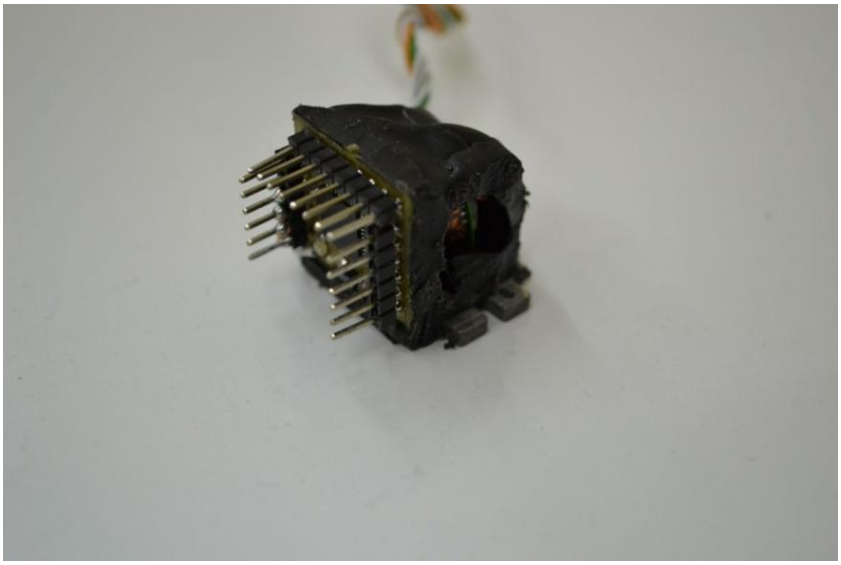
3.4 Ataletsel Ölçüm Birimi

Ataletsel ölçüm birimi (AÖB) içerdiği 3 eksenli doğrusal ivme sensörü ve üç eksendeki jiroskop sayesinde lineer ivmelenmeleri ve açısal değişimleri algılar. Yine içerdiği işlemci birimi sayesinde sensörlerden alınan analog değerleri filtreleyip seçilen bir referans sistemine göre üç eksenli pozisyon ve yönelim değerlerini verebilir.

İstanbul Teknik Üniversitesi (İTÜ) Kontrol Mühendisliği Robotik Labarotuarında kullanılmakta olan Analog Devices'a ait AÖB Şekil 3.9'da görülmektedir. Bu AÖB bağlantı kablosunun kopması sonucu bozulmuş ve bir lisans bitirme projesi kapsamında içerdiği ivme sensörleri ve jireskoplar kullanılarak yeni bir yapı oluşturulmuştur. AÖB'nin içerdiği işlemci birimi Analog Devicesa ait özel bir yapı olduğu için bunun yerine Microchip firmasının dsPIC serisinden dsPIC33FJ128MC804 mikrodenetleyicisi kullanılmıştır. Yeni tasarlanan devreye ait mikrodenetleyici bağlantı şeması Şekil 3.10'da görülmektedir. Yeni tasarlanan devreye elektronik komponentler lehimlendikten sonra komponentlerin zarar görmemesi için üzerleri sıcak silikon ile kaplanmıştır. Yapının son hali şekil 3.11'de görülmektedir [14]. Tasarlanan devreye ait bağlantı şeması da Şekil 3.12'de görülmektedir.



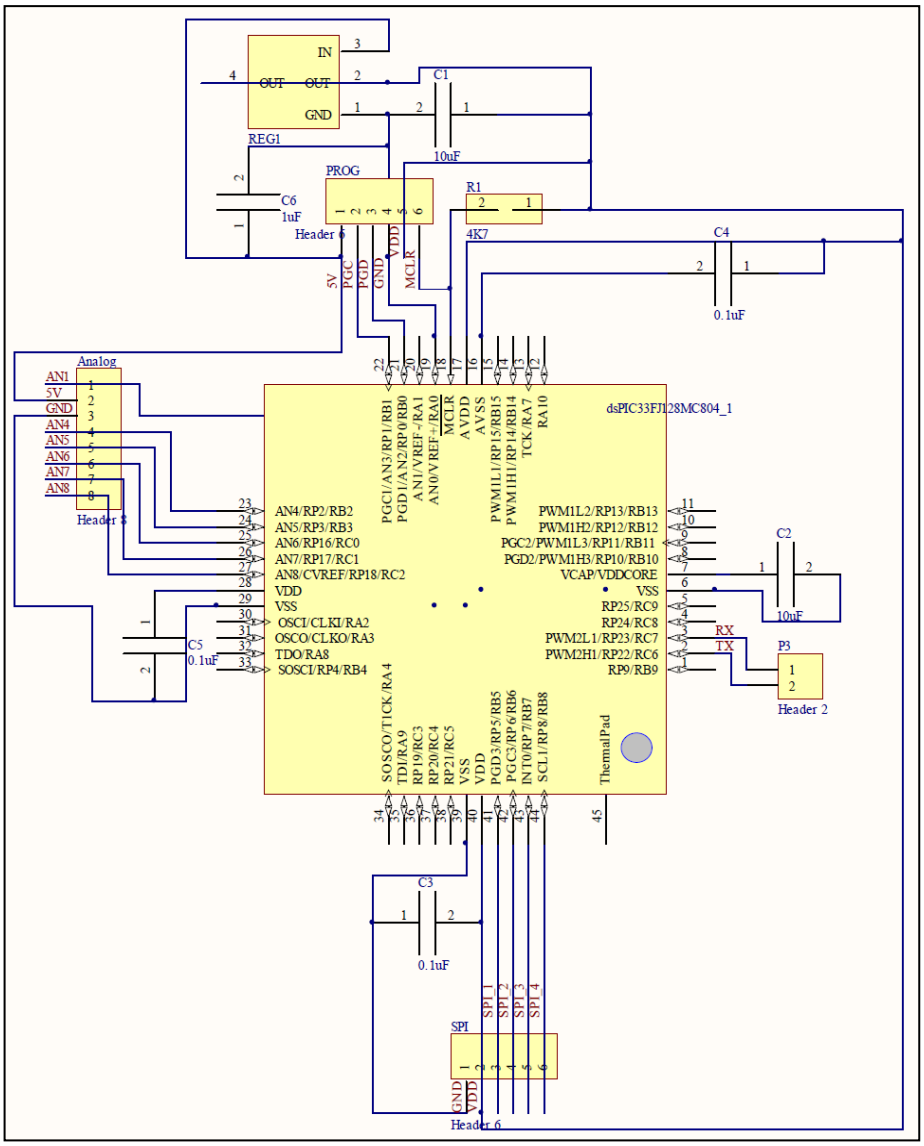
Şekil 3.10 : Analog devices ADIS16365 AÖB



Şekil 3.11 : dsPIC33FJ128MC804 kontrollü AÖB

Bu devrede sensörlerden okunan 6 değer dsPIC'in analog kanallarından okunmakta bu değerler dsPIC'te işlenmekte ve elde edilen yönelim değerleri Euler açıları olarak hesaplanmaktadır. dsPIC'e ait kaynak kod EK.B'de verilmiştir.

Sonuç olarak Şekil 3.13'te görülen uçak gerçeklenmiştir. Uçağın tüm elmanları üzerinde iken yapılan manuel uçuş denemesinde fırçasız motor %40 oranında çalıştırıldığında uçağı kaldırmaya yetecek kadar thrust oluştuğu görülmüştür.



Şekil 3.12 : dsPIC33FJ128MC804 bağlantı şeması



Şekil 3.13 : Gerçeklenen model uçak

Modelde kullanılan elemanların ağırlıkları Çizelge 3.3'te verilmiştir.

Çizelge 3.3 : Kullanılan elemanların yaklaşık ağırlıkları

Eleman	Ağırlık
Depron Köpük	137 g
Servo Motorlar	25.4g
Fırçasız DC Motor	66g
Fırçasız DC Motor Sürücü	87g
Ataletsel Ölçüm Birimi	24g
Haberleşme Kontrol Kartı	18g
Kablo ve Bağlantı Elemanları	50g
TOPLAM	407g

4. SABİT KANATLI İNSANSIZ HAVA ARACININ MODELLENMESİ

4.1 Kuaternion Yönelim Tanımı

Kuaternion karmaşık sayılar cisminin değişmesiz genişletmesidir. İlk defa İrlanda'lı matematikçi Sir William Rowan Hamilton tarafından 1843 yılında tanımlanmış, ve 3 boyutlu uzaydaki matematiğe uygulanmıştır. İlk başta, kuaterniyonlar değişme kuralına (ab = ba) uymadıkları için sorunlu kabul edilmişlerdir. Her ne kadar pek çok uygulamada vektörler yerlerini almış olsa da, hala kuramsal ve uygulamalı matematikte kullanılmaktadırlar. Başlıca kullanım alanları, 3 boyutlu uzayda dönme hareketinin hesaplanmasıdır.

Yatay uçuş ve dikey denge konumları arasında geçişlerin olacağı bir İHA'nın yöneliminin tanımlanması sırasında tekil durumların oluşmamasını sağlanmalıdır. Bir hava aracının yönelimini tanımlayan birkaç yöntem vardır. Bunlardan en yaygını euler açılarıdır. Euler açıları İHA'nın yalpalama (roll), yunuslama (pitch) ve dümen (yaw) açılarını olacağı için kullanımı kolaydır. Fakat, uçuş durumları arasındaki geçiş sırasında yunuslama açısı $\pm 90°$ civarında iken euler açı tanımı yalpalama ve dümen açılarını ayırt edemez.

İHA'nın yatay uçuşu sırasında yer düzlemi yaklaşık olarak yunuslamaya paralel olacağı için tekillik oluşmaz. Bu çalışmada dikey dengede durma kontrolü yapılacağı için yunuslama ekseni yer düzlemine dik olacak ve euler açıları için tekillik problemi ortaya çıkacaktır. Bu nedenle tekillik içermeyen bir yönelim tanımına ihtiyaç vardır.

Euler açıları yaklaşımında oluşan tekilliği gidermek adına global bir yönelim yaklaşımı olarak kuaternion yaklaşımı kullanılabilir. Tanımlanan global yönelim referansı üzerine yerel yönelim tekillik olmaksızın euler açıları olarak belirlenebilir.

Kuaternion 3 serbestlik derecesine sahip 4 reel sayıdan oluşan bir birim vektördür. Daha açıklayıcı olmak gerekirse kuaternion, denklem 4.1'de görüldüğü gibi üç boyutlu bir $\bar{E}$ birim vektöründe oluşan $\zeta/2$ radyanlık bir rotasyon olarak tanımlanabilir.

$$q = \begin{bmatrix} \cos(\zeta/2) \\ \bar{E}\cos(\zeta/2) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad \textbf{(4.1)}$$

Buna ek olarak sıfır kuaternionu $q^{(0)} = [1 \quad 0 \quad 0 \quad 0]^T$ olarak tanımlananır, bu durum euler açılarının $\phi = \theta$ ve $\psi = 0^o$ olduğu yönelimi ifade eder. Bu çalışmada sıfır kuaternionunun ifade ettiği yönelim yatay uçuşta ucunun kuzeye baktığı andaki durum olarak kabul edilmiştir. Kuaternion çarpımı değişmeyen bir sıfır kuaternionu referansına göre yönelimleri ifade eder ve matris çarpım kurallarına uyar. Kuaternion çarpımı denklem 4.2. de tanımlanmıştır, burada p ve q kuaternionları M(p) ve M(q) ise denklem 4.3'te tanımlanmış olan olan kuaternion matrislerini ifade etmektedir.

$$p * q = \mathrm{M(p)} \cdot q = \mathrm{M(p)} \cdot \mathrm{M(q)} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \textbf{(4.2)}$$

$$\mathrm{M}(\eta) = \begin{bmatrix} \eta_0 & -\eta_1 & -\eta_2 & -\eta_3 \\ \eta_1 & \eta_0 & -\eta_3 & \eta_2 \\ \eta_2 & \eta_3 & \eta_0 & -\eta_1 \\ \eta_3 & -\eta_2 & \eta_1 & \eta_0 \end{bmatrix} \quad \textbf{(4.3)}$$

Kuaternion matrisi, matris işlemleri kurallarına uyar. Kuaternionlar birim vektörler olduğu için kuaternion matristeri için $\mathrm{M}^{-1} = \mathrm{M}^{\mathrm{T}}$ eşitliği geçerlidir.

İHA'nın kontrolü sırasında aracın anlık yöneliminden $q^{anlık}$ çok istenen referans yöneliminde q^{ref} oluşan değişim yönelimi $q^{degisim}$ daha önemlidir. Kuaternion çarpımı tanımı gereğince $q^{anlık}$ aşağıdaki gibi ifade edilebilir.

$$q^{anlık} = q^{degisim} * q^{ref} \quad \textbf{(4.4)}$$

Aynı zamanda $q^{degisim}$ aşağıdaki gibi parçalanabilir.

$$q^{degisim} = q^x * q^y * q^z \quad \textbf{(4.5)}$$

Denkelm (4.5) te tanımlanan q^x, q^y, q^z kuaternionları Şekil 4.1'de görülen x_B, y_B, z_B eksenlerinde oluşan rotasyonlardır. Bu kuaternionların euler açıları cinsinden aşağıdaki bileşenleri vardır.

$$q^x = [q_0^x \; q_1^x \; 0 \; 0]^{\;T} \tag{4.6}$$

$$q_0^x = cos(\phi) = \frac{1 - 2(q_1^2 + q_2^2)}{\sqrt{1 - 4(q_0 q_2 - q_1 q_3)^2}} \tag{4.7}$$

$$q_1^x = sin(\phi) = \frac{2(q_0 q_1 + q_2 q_3)}{\sqrt{1 - 4(q_0 q_2 - q_1 q_3)^2}} \tag{4.8}$$

$$q^y = \left[q_0^y \; 0 \; q_2^y \; 0\right]^T \tag{4.9}$$

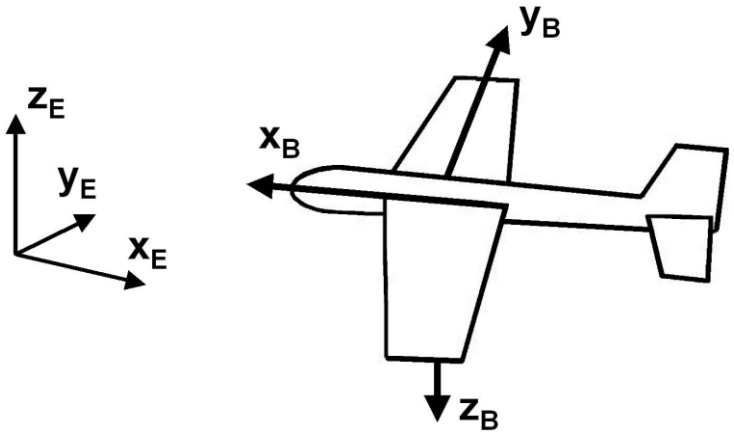
$$q_0^y = cos(\theta) = \sqrt{1 - 4(q_0 q_2 - q_1 q_3)^2} \tag{4.10}$$

$$q_2^y = sin(\theta) = 2(q_0 q_1 + q_2 q_3) \tag{4.11}$$

$$q^z = [q_0^z \; 0 \; 0 \; q_3^z]^T \tag{4.12}$$

$$q_0^z = cos(\psi) = \frac{1 - 2({q_2}^2 + {q_3}^2)}{\sqrt{1 - 4(q_0 q_2 - q_1 q_3)^2}} \tag{4.13}$$

$$q_3^z = sin(\psi) = \frac{2(q_1 q_2 + q_0 q_3)}{\sqrt{1 - 4(q_0 q_2 - q_1 q_3)^2}} \tag{4.14}$$



Şekil 4.1 : Refertans ve gövde eksenleri

Denklem (4.5) gösterilen ayrıştırmaya tekil olma durumlarını ortadan kaldırmamıştır. Tekilliği ortadan kaldırmak için yatay uçuştan dikey uçuşa geçildiği

sırada yunuslama açısı 90^o'e yaklaşırken kuaternion referans düzlemi değiştirilerek tekillikten kurtarılmış olur.

4.2 SKİHA'nın Dikey Denge Modeli

Şekil 4.1'de de görüldüğü üzere yeryüzü x_E, y_E, z_E vektörleri ile tanımlanan eksen sistemi ile ifade edilirken rijit olarak tanımlanan uçak eksen sistemi ise x_B, y_B, z_B vektörleri ile tanımlanmıştır.

Tasarlanan İHA tek pervaneli bir uçaktır. Pervanede dönüş hızının değiştirilmesi hem x_B yönünde itiş kuvvetinin hem de yalpalama eksenindeki momentin değişmesine neden olacaktır. Pervanenin oluşturduğu momentin iki temel bileşeni vardır. Bunlardan biri pervane dönüş hızının değişmesidir. Pervane ivmesinden kaynaklanan moment (4.15)'te verilmiştir.

$$M_{ivme} = \frac{I_{per}\dot{\omega}}{I_x} \tag{4.15}$$

I_{per} ve I_x, pervane ve hava çerçevesinin x_B yönündeki atalet momentleri iken ω pervanenin dönüş hızıdır.

Pervane sürükleme momenti ise toplam pervane momentinin ikinci temel bileşenidir. NACA tarafından standardilize edilen itiş (thrust) ve güç (power) katsayıları C_T ve C_P [15].

$$C_T = \frac{T}{pn^2d^4} \tag{4.16}$$

$$C_P = \frac{P}{pn^3d^5} \tag{4.17}$$

Dikey uçuş modunda uçağın denge kalması hedeflendiği için Newton'un ikinci yasası gereği $T = m_u g$ olmalıdır. Burada m_u uçağın kütlesini g ise yerçekimi ivmesini ifade etmektedir. Bu çıkarımda oluşan itme kuvvetinin (T) sadece yerçekimine karşı iş yaptığı kabul edilmiştir. Ayrıca pervaneden geçen havanın sıkıştırılamadığı ve hava yoğunluğu ρ'nun sabit olduğu kabul edilmiştir. Yukarıdaki denklemlerde geçen diğer parametreler d ve n ise pervane çapı ve pervanenin bir saniyedeki dönüş

sayısıdır. Pervane sürükleme momenti ile güç arasındaki bağıntı (4.18) de verilmiştir. Bu durumda sürükleme momenti (4.19) daki gibi olur.

$$P = M_{sürükleme} 2\pi n \tag{4.18}$$

$$M_{\ sürükleme} = \frac{T d C_P}{2\pi C_T} \tag{4.19}$$

Dikey denge uçuşu sırasında kontrol yüzeylerinin sadece bir kısmı hava akışına maruz kalır. Pervanenin oluşturduğu hava akışı u_{perv} (4.20) de verilmiştir. Burada aktüatör disk alanı $A_{disk} = \pi d^2/4$ olarak kullanılabilir.

$$u_{perv} = \sqrt{\frac{2T}{\rho A_{disk}}} \tag{4.20}$$

Kontrol yüzeylerindeki itme katsayıları eleron, kuyruk ve dümen için $C_{L_{\delta_{eleron}}}, C_{L_{\delta_{kuyruk}}}, C_{L_{\delta_{dümen}}}$ olarak tanımlanmıştır. Kanatçıklarda oluşacak olan itme kuvvetleri aşağıdaki gibi genelleştirilebilir.

$$L_{kanat} = \frac{1}{2}\ \rho u_{\ \infty}{}^2 A_{\ kanat} C_{\ L_\alpha} \alpha \tag{4.21}$$

Pervanenin oluşturduğu rüzgarın kapakçıklara etki açısı küçük açı yaklaşımıyla $\alpha = \dot{z}_B/u_\infty$ şeklinde ifade edilebilir. Eleronlarda oluşan sürüklenme kuvveti (4.22) ki gibi formülize edilebilir.

$$D_{eleron} = \left(C_{D,0} + \frac{C^2{}_{L_{\delta_{eleron}}} \delta^2{}_{eleron}}{\pi e G_o} \right) \frac{1}{2} \rho u_{perv}{}^2 A_{eleron,perv.} \tag{4.22}$$

Yunuslama ve dümen itme momenteri q_{damp} ve r_{damp} aşağıda verilmiştir. L_{perv_kuyruk} ve $L_{perv_dümen}$ pervane ile kuyruk ve dümen arasındaki uzaklığı, α_{kuyruk} ve $\alpha_{dümen}$ pervane rüzgarının etki açılarını, C_{kuyruk} ve $C_{dümen}$ ise moment katsayılarıdır.

$$q_{damp} = \frac{L_{kuyruk} \alpha_{kuyruk} A_{kuyruk} \frac{1}{2} \rho u^2{}_{perv}}{I_y} \tag{4.23}$$

$$r_{damp} = \frac{L_{dümen}\alpha_{dümen}A_{dümen}\frac{1}{2}\rho u^2{}_{perv}}{I_z} \quad \textbf{(4.24)}$$

$$\alpha_{kuyruk} = \frac{L_{perv_kuyruk}q}{u_{perv}} \quad \textbf{(4.25)}$$

$$\alpha_{dümen} = \frac{L_{perv_dümen}r}{u_{perv}} \quad \textbf{(4.26)}$$

$$C_{kuyruk} = \frac{q_{damp}}{qu_{perv}} \quad \textbf{(4.27)}$$

$$C_{dümen} = \frac{r_{damp}}{ru_{prop}} \quad \textbf{(4.28)}$$

Kuaternion yönelim tanımına göre genel hareket denklemleri aşağıdaki gibi olacaktır [16].

$$x_E = 2(q_1q_2 + q_0q_3)x_B + [1 - 2({q_1}^2 + {q_3}^2)]y_B + 2(q_2q_3 + q_0q_1)z_B$$

$$y_E = [1 - 2({q_2}^2 + {q_3}^2)]x_B + 2(q_1q_2 - q_0q_3)y_B + 2(q_0q_2 + q_1q_3)z_B \quad \textbf{(4.29)}$$

$$z_E = -2(q_1q_3 - q_0q_2)x_B - 2(q_0q_1 + q_2q_3)y_B - [1 - 2({q_1}^2 + {q_2}^2)]z_B$$

$$\ddot{x}_B = 2(q_1q_3 - q_0q_2)g + \frac{1}{m}\delta_{throttle} - \left(C_{D,0} + \frac{{C_L}^2}{\pi e G_o}\right)\frac{1}{2}\rho {u_\infty}^2 A_{kanat} - \frac{D_{eleron}}{m}$$

$$\ddot{y}_B = 2(q_0q_1 + q_2q_3)g + C_{L_{\delta_{dümen}}}\frac{1}{2}\rho\left({u^2}_{perv}A_{dümen,perv} + {u_\infty}^2 A_{dümen}\right)\delta_{dümen} \quad \textbf{(4.30)}$$

$$\ddot{z}_B = [1 - 2({q_1}^2 + {q_2}^2)]g + C_{L_{\delta_{kuyruk}}}\frac{1}{2}\rho\left({u^2}_{perv}A_{kuyruk,perv} + {u_\infty}^2 A_{kuyruk}\right)\delta_{kuyruk} - L_{\ kanat}$$

$$\dot{p} = qr\left(\frac{I_y - I_z}{I_x}\right) + C_{L_{\delta_{ail}}} \frac{1}{2}\rho\left(u^2{}_{perv} A_{eleron,perv} L_{eleron,perv} + u_\infty{}^2 A_{eleron} L_{eleron}\right)\frac{1}{I_x}\delta_{eleron} - \frac{M_{acc} + M_{drag}}{I_x}$$

$$\dot{q} = pr\left(\frac{I_z - I_x}{I_y}\right) + C_{L_{\delta_{kuyruk}}} \frac{1}{2}\rho\left(u^2{}_{perv} A_{kuyruk,perv} + u_\infty{}^2 A_{kuyruk}\right)\frac{L_{kuyruk}}{I_y}\delta_{kuyruk} - C_{kuyruk} u_{perv} q \quad \textbf{(4.31)}$$

$$\dot{r} = pq\left(\frac{I_x - I_y}{I_z}\right) + C_{L_{\delta_{dümen}}} \frac{1}{2}\rho\left(u^2{}_{perv} A_{dümen,perv} + u_\infty{}^2 A_{dümen}\right)\frac{L_{dümen}}{I_z}\delta_{dümen} - C_{dümen_d} u_{perv} r$$

$$\begin{aligned} \dot{q_0} &= \frac{1}{2}(-pq_1 - qq_2 - rq_3) \\ \dot{q_1} &= \frac{1}{2}(pq_0 - qq_3 + rq_2) \\ \dot{q_2} &= \frac{1}{2}(pq_3 + qq_0 - rq_1) \\ \dot{q_3} &= \frac{1}{2}(-pq_2 + qq_1 + rq_0) \end{aligned} \quad \textbf{(4.32)}$$

Bu denklemlerde geçen $[q_0 \quad q_1 \quad q_2 \quad q_3]^T$ vektörü kuaternion yönelimi ifade etmektedir. A_{eleron_perv}, A_{kuyruk_perv}, $A_{dümen_perv}$ pervanenin oluşturduğu hava akışının taradığı kontrol kapakçıkları alanlarını, A_{eleron}, A_{kuyruk}, $A_{dümen}$ dış hava akışlarından etkilenen kontrol kapakçıkları alanlarını, L_{eleron_perv}, L_{eleron}, L_{kuyruk}, $L_{dümen}$ ise çeşitli momen kolu uzunluklarını ifade etmektedir.

Uçakta oluşacak sürüklenmenin temel iki bileşeni vardır. Endüklenen sürüklenme ve parazit sürüklenme. Bu sürüklenmelerin hesaplanması endüklenen sürüklenme katsatısı $C_{Di} = \frac{C_L^2}{\pi\, e\, GO}$ ve parazit sürüklenme katsayısı $C_{D,0}$ kullanılır. Burada $C_L = C_{L_\alpha}\alpha$ rüzgarın kaldırma katsayısını, e rüzgar verimini GO ise rüzarın görüş oranını ifade etmektedir. İHA dikey denge uçuşu yaparken etki eden sürüklenme ihmal edilebilir. Bunun nedeni dikey denge uçuşunda yatay uçuşta olduğu gibi gövde çerçevesinin gibi yüksek hava akışını etkilememesidir. Yine yanal eksenlerden

gelecek çevresel rüzgarların az olması nedeniyle y_B ve z_B eksenlerinde oluşacak kuvvet ifadeleri ihmal edilebilir.

Pervaneye ait atalet momenti I_{perv} uçağın x esenindeki atalet momenti I_x'den çok küçük olduğu için ihmal edilebilir. Uçak gövdesi dengede olacağı için açışal hızlar *p, q* ve *r* küçük değerlerde kalacağı için bunların kareleri ve çarpımları çok küçük olur ve ihmal edilebilir.

Yatay denge uçuşunda pervanenin sabit hızla döndüğü düşünülebilir. Bu durumda hava akışı sabit olacaktır, uçağında sabit durması nedeniyle $u_\infty = 0$ olacaktır.

$$\delta_{eleron} = \frac{2}{C_{L_{\delta_{eleron}}} \rho {u_{perv}}^2 A_{eleron,perv} L_{eleron,perv}} M_{sürüklenme} + \hat{\delta}_{eleron} \qquad \textbf{(4.33)}$$

$$\delta_{throttle} = mg + D_{eleron} + \hat{\delta}_{throttle} \qquad \textbf{(4.34)}$$

(4.34) te geçen $\hat{\delta}_{throttle}$ ifadesi İHA dengede durduğu andaki pervane kaldırma kuvvetinde oluşacak küçük değişimlerdir. D_{ail} aileronların oluşturduğu sürüklenme kuvvetidir. Bu kuvvet uçağın ağırlığından küçüktür ve uçağımızın yanal hareketlerinde ihmal edilebilir.

Uçağın kontrolü sırasında oluşacak sapmalar kuaternionlar ifade edilmiştir. Denge durumunda oluşacak sapma kuaternionu (q^{dev}) anlık kuaternionun (q^{curret}), yönelme kuaternionuna ($q^{hdg} = [cos(hdg),sin(hdg),0,0]^T = [c^*, s^*, 0, 0]^T$; *hdg: yönelme referansı*) ve dikey kuaterniona ($q^v = [\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 0]$) iz düşümü olarak hesaplanabilir. Burada sapma kuaternionu yatay quaterniondan oluşacak sapma miktarlarını ifade eden bir vektör olacaktır.

$$q^{current} = q^{dev} * \mathrm{q}^{\mathrm{hdg}} * \mathrm{q}^{\mathrm{v}}$$

$$= \frac{\sqrt{2}}{2} \begin{bmatrix} \bar{c}{q_0}^{dev} - \bar{s}{q_1}^{dev} - \bar{c}{q_2}^{dev} + \bar{s}{q_3}^{dev} \\ \bar{s}{q_0}^{dev} + \bar{c}{q_1}^{dev} + \bar{s}{q_2}^{dev} + \bar{c}{q_3}^{dev} \\ \bar{c}{q_0}^{dev} - \bar{s}{q_1}^{dev} + \bar{c}{q_2}^{dev} + \bar{s}{q_3}^{dev} \\ -\bar{s}{q_0}^{dev} - \bar{c}{q_1}^{dev} + \bar{s}{q_2}^{dev} + \bar{c}{q_3}^{dev} \end{bmatrix} \qquad \textbf{(4.35)}$$

Sonuç olarak sapma quaternonları kullanılarak aşağıdaki kuaternion modeli elde edilir.

$$\ddot{x_B} = \frac{1}{m}\hat{\delta}_{throttle}$$

$$\ddot{y}_B = \ddot{g}{q_3}^{dev} \tag{4.36}$$

$$\ddot{z}_B = -g{q_2}^{dev}$$

$$\dot{p} = C_{L_{\delta_{eleron}}}\frac{1}{2}\rho{u_{perv}}^2 A_{eleron,perv} L_{eleron,perv}\frac{1}{I_x}\hat{\delta}_{eleron}$$

$$\dot{q} = C_{L_{\delta_{kuyruk}}}\frac{1}{2}\rho{u_{perv}}^2 A_{kuyruk,perv}\frac{L_{kuyruk}}{I_y}\delta_{kuyruk} - C_{kuyruk}u_{perv}q \tag{4.37}$$

$$\dot{r} = C_{L_{\delta_{dümen}}}\frac{1}{2}\rho{u_{perv}}^2 A_{dümen,perv}\frac{L_{dümen}}{I_z}\delta_{dümen} - C_{dümen_d}u_{perv}r$$

$$\dot{q}^{dev} = \begin{bmatrix} 0 \\ \frac{p}{2} \\ \frac{q}{2} \\ \frac{r}{2} \end{bmatrix} \tag{4.38}$$

Basitleştirilmiş model iki adet dördüncü iki adet ikinci mertebeden lineer zamanla değişmeyen sisteme ayrıştırılabilir. Dördüncü dereceden sistemlerden biri yunuslama açısı ve z_B konumunun diğeri ise dümen açısının ve y_B konumunu dinamik modellerini ifade eder. İkinci mertebeden sistemler ise ayrı ayrı x_B konumu ve yuvarlanma açısının dinamik modellerini ifade eder.

5. SABİT KANATLI İNSANSIZ HAVA ARACININ KONTROLÜ

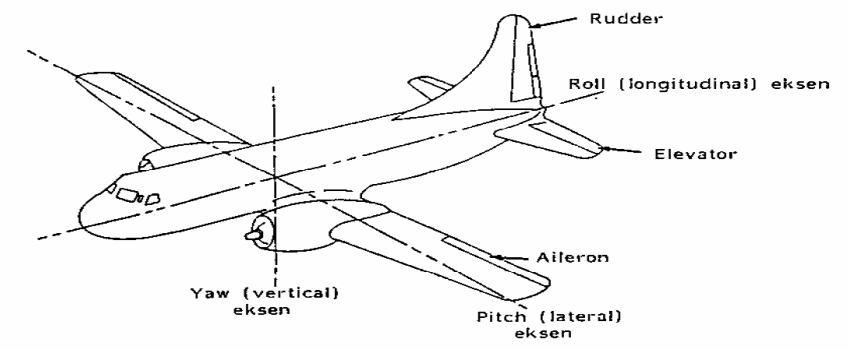
5.1 Uçak Eksenleri

Bir uçak; uçuş esnasında; tüm hareketlerini üç ayrı eksen üzerinde yapar.Şekil 5.1'de görülmekte olan bu eksenler birbirlerine 90° açı yaparlar ve longitudinal eksen, lateral eksen, vertical eksen olarak isimlendirilirler.

Longitudinal (Roll hareketlerinin yapıldığı) eksen : Uçağın burnu ile kuyruk konisini birleştiren hayali hattır. Uçağın sağa ve sola yatış hareketleri bu eksen üzerinde olur.

Lateral (Pitch hareketlerinin yapıldığı) eksen : Uçağın iki kanat ucunu birleştiren hayali hattır. Uçağın burun aşağı ve yukarı hareketleri bu eksen üzerinde olur.

Vertical (Yaw hareketlerinin yapıldığı) eksen : Yukarıda bahsedilen bu iki eksene 90° dik, uçağı üstlen alta doğru delen eksendir. Uçağın yönünün belirlenmesini sağlayan sağa ve sola dönüş hareketleri bu eksen üzerinde olur.



Şekil 5.1 : Uçuş eksenleri

5.1.1 Roll kontrol

Uçağın roll kontrolü; yani sağa ve sola yatışları her bir kanadın firar kenarında bulunan eleron (aileron) ile sağlanır. eleron 'lara; pilot kabininde bulunan her iki pilot kumanda volanlarının sağa veya sola döndürülmesi ile kumanda edilir. Aileron kumanda volanının sağa döndürülmesinde; sağ aileron'un firar kenarı yukarı, sol "aileron"'un firar kenarı ise aşağı doğru hareket eder. Sağ eleron 'un firar kenarının

yukarı hareketi; kanat ile beraber alt tarafta bir kamburluk oluşturur. Diğer bir deyimle kanat airfoil şekli değiştirilmiş olur. Airfoil üzerinde kaldırma kuvvetinin elde edilişinden hatırlayacağımız gibi; kamburluk kanat altından akan hava filelerinin hızını arttıracak, dolayısıyla basınç düşecektir.

Kanat üzerinde ise; hava hızında bir değişme olmayacağından; basınç; yaklaşık ortam basıncına eşit olacaktır. İşte aileron ile kanadın beraberce oluşturacakları alanda meydana gelen bu basınç farkı ile oluşan kuvvet; kanat üzerinde meydana gelen kaldırma kuvvetine karşı bir güç oluşturur. Bu nedenle kanat kaldırma kuvveti azalır. Azalan kanat kaldırma kuvveti nedeniyle kanadın roll ekseni etrafında hareket başlar ve sağ kanat aşağı doğru hareket eder.

Kısacası; kumanda volanı ile verilen yatış kumandası; her iki kanattaki "aileron"'ların birbirlerine ters olarak çalışmalarını sağlar. Bu çalışma da kanatların kaldırma kuvvetlerini etkileyerek uçağın roll ekseni etrafında sağa ve sola yatışlarını sağlar. Böylece roll kontrolü elde edilmiş olur.

Bazı uçaklarda iç ve dış olmak üzere her bir kanatta iki adet aileron bulunur. Uçağın düşük hızlarında her iki aileron birlikte kullanılarak uçağın roll kontrolü etkin olarak sağlanır. Uçağın yüksek hızlarında ise; dış "aileron"'lar kilitlenerek nötr pozisyonunda kalırlar. Bu durumda uçağın roll kontrolü sadece iç aileronlar ile sağlanır.

Uçakların roll kontrolü kanatların kaldırma kuvvetlerinin arttırılması veya azaltılması ile sağlandığına göre; yatış yapılacak taraftaki kanat kaldırma kuvveti ne kadar azaltılırsa yatış o kadar çabuk ve etkili olur. Uçakların kanatlarındaki kaldırma kuvvetlerini azaltmak için "spoiler"'ler kullanılır. Normal durumlarda kanat üst yüzeyini oluşturan "spoiler"'lerin açılmaları ile oluşturdukları açısal pozisyon; kanat üzerinden akan hava filelerinin akışını bozarlar ve kanat kaldırma kuvvetinin azalmasına neden olur.

"Aileron"'larla uçağa yatış kumandası verildiği zaman; yatış yapılan taraftaki kanat üzerinde bulunan spoiler'ler açılır, diğer kanattakiler kapalı kalırlar. "Spoiler"'i açılan kanattaki bozulan kaldırma kuvveti uçağın yatışına yardımcı olur.

5.1.2 Pitch kontrol

Uçağın pitch ekseni etrafındaki burun aşağı ve yukarı hareketleri elevatorler ile sağlanır. Elevatorler; uçağın yatay stabililizer'inin firar kenarındadır.

Pilot kabininde bulunan her bir levyenin ileri ve geri hareket ettirilmesi ile elevatorlerin firar kenarlarının aşağı ve yukarı hareket eder. Elevator'lerin yatay stabilizemle açı yapması; altta ve üstte kamburluk oluşumuna neden olur. Daha önceden öğrendiğimiz gibi hava filelerinin hızı; kamburluğun olduğu tarafta daha fazla, basıncı ise; daha azdır Bu basınç farklılığı ile yatay stabilizer üzerinde meydana gelen kuvvet"' uçağı burun aşağı ve yukarı hareket ettirir.

5.1.3 Yaw kontrol

Uçağın yaw ekseni etrafındaki sağa ve sola dönüş hareketleri; dikey stabilizenin firar kenarında bulunan rudder ile sağlanır. “Rudder”'in sağa ve sola hareketleri; pilot kabininde bulunan her bir pilota ait birer çift pedal ile sağlanır. İki pedal çiftinden birini kullanarak kumanda edildiğinde; pedalın birinin ileri gittiği, diğerinin ise; geri geldiği görülür.

“Rudder”’in dikey stabilizer ile yapmış olduğu kamburluk; iki tarafta basınç farklılığı yaratır. Bu basınç farklılığı uçağın sağa veya sola dönmesini sağlar. Böylece uçağın yönü değişir.

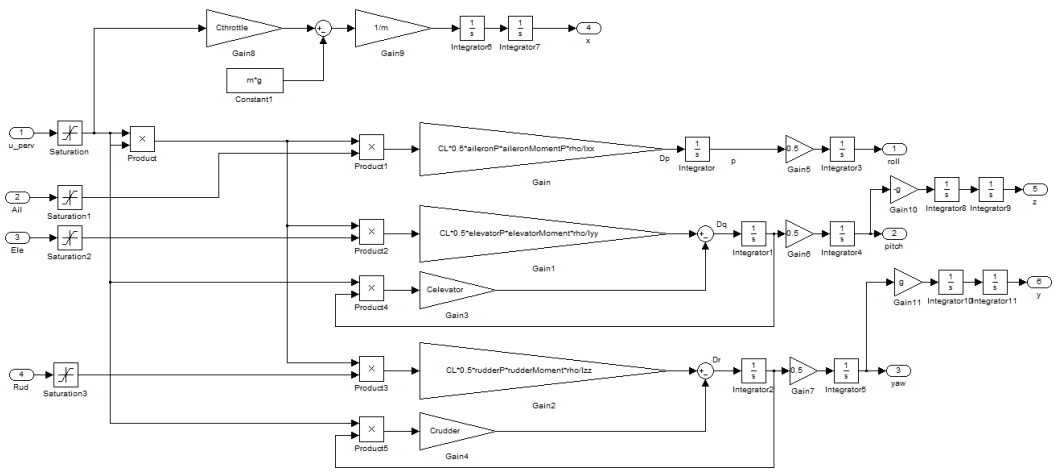
5.2 Dikey Denge Uçuşu PD - PID Kontrolü

Bu bölümdeMatlab Simulink ortamında tasaralanan PID ve PD kontrolörleri ve simülasyon sonuçları verilecektir. Öncelikle Bölüm 4.2’de matematiksel modelini elde etmiş olduğumuz SKİHA’ya ait simülasyon bloğu Şekil 5.2’de ki gibi oluşturulmuştur.

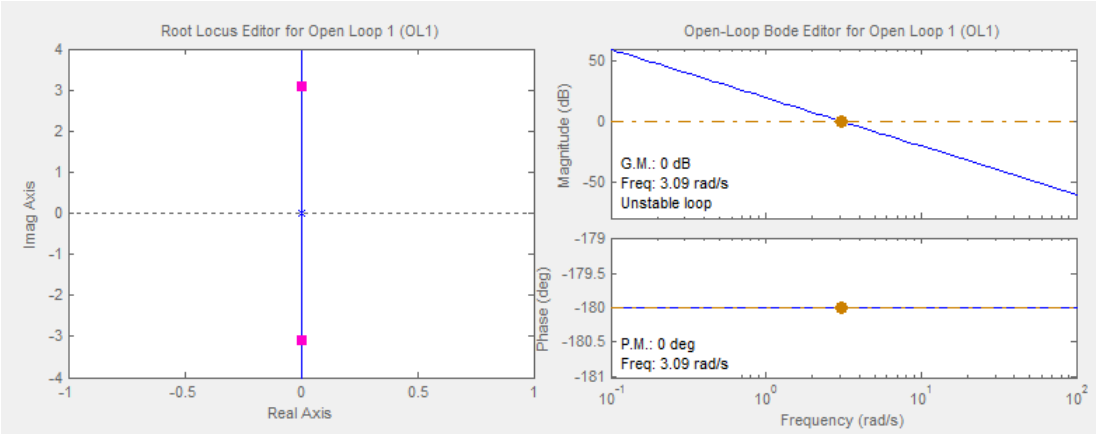
Bu modelde kullanılan parametrelerin değerleri EK-C ‘de belirtildiği gibi verilmiştir.

SKİHA’nın dikey denge uçuşu sırasındaki altı eksenine ait hareket modeli yukarıdaki bigi oluşturulduktan sonra rol, pitch, yaw ve x eksenlerine ait modeller yine EK-C’de gösterildiği gibi bulunmuştur.

SKİHA’nın x eksenindeki hareketini kontrol etmek üzere bir PID kontrolör tasarlanması ön görülmüştür. Sistemin x eksenindeki hareket denklemlerinden elde edilen model Matlab Sisotool ortamına Şekil 5.3’de görüldüğü gibi aktarılmıştır.

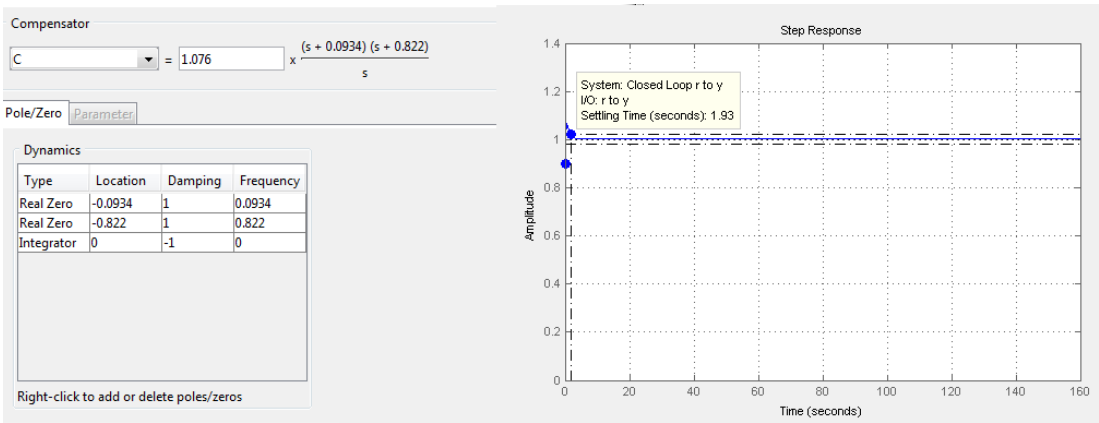


Şekil 5.2 : SKİHA'nın matematiksel modeli

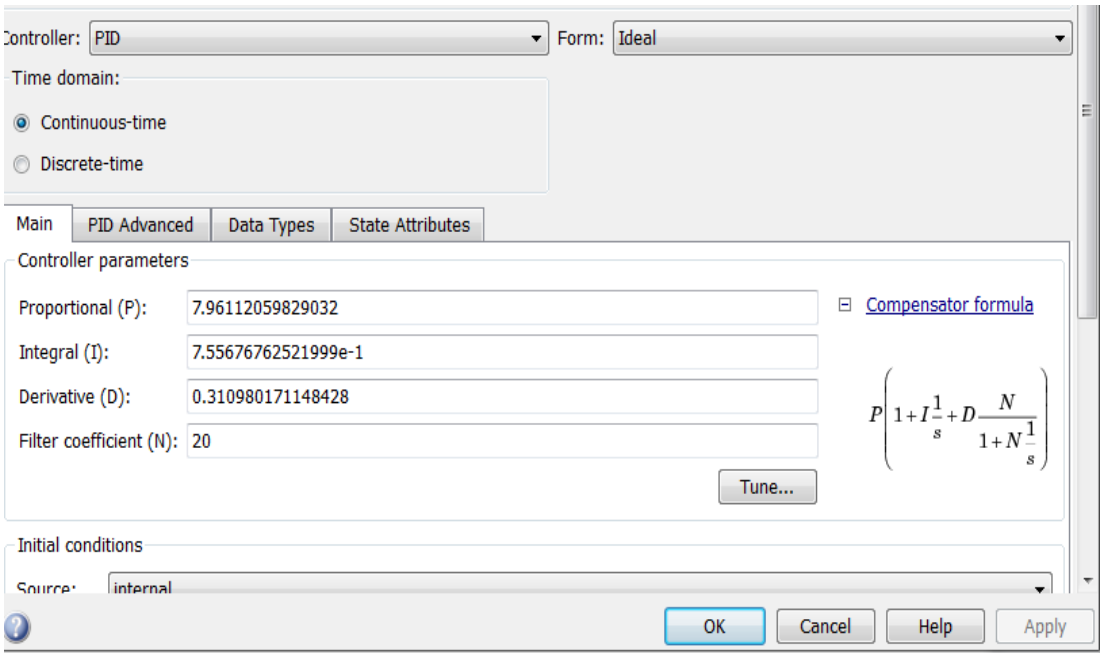


Şekil 5.3 : SKİHA'nın dikey uçuş x ekseni modeline ait sisotool analizi

Sisotool ortamında elde edilen detayları Şekil 5.4'te verilmiş olan PID kontrolör model üzerinde denenmiştir. Sistemde bulunan saturasyonlar nedeniyle kontrol sonuçları beklenen iyilikte olmamıştır dolayısı ile kontrolörün PID katsayıları Simulink Ototune özelliği kullanılarak optimize edilmiştir. Optimizasyon sonucu elde edilen PID yapısı ve katsayıları Şekil 5.5'tegörüldüğü gibi elde edilmiştir.

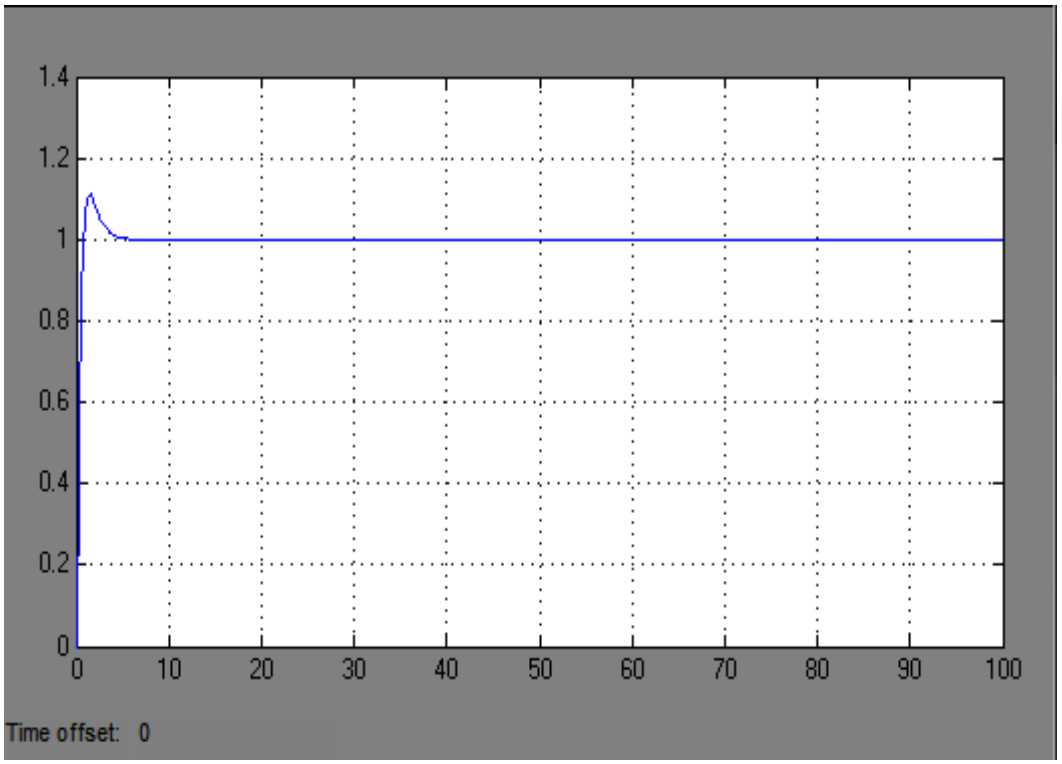


Şekil 5.4 : Sisotool ortamında elde edilen kontrolörün zaman cevabı



Şekil 5.5 : Simulink otomatik ayarlama özelliği kullanılarak optimize edilen PID katsayıları

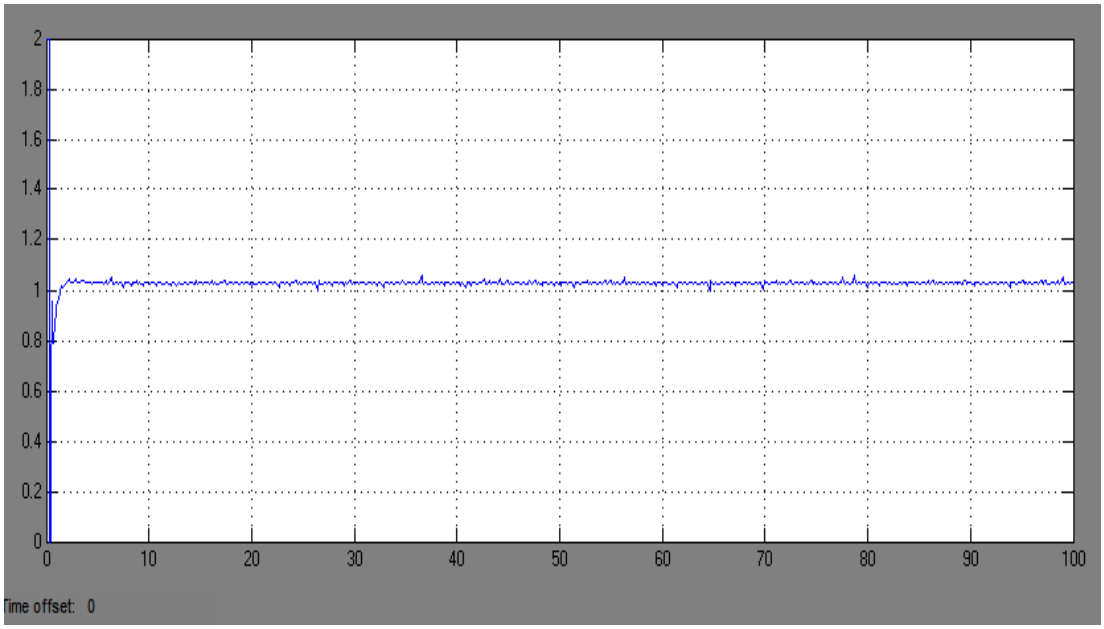
Sonuç olarak ulaşılan PID kontrolör simülasyon ortamında denendiğinde birim basamak girişi için Şekil 5.6'da görülen cevabı vermiştir.



Şekil 5.6 : Dikey uçuşta SKİHA istenilen x konumu kontrolünün birim basamk yanıtı

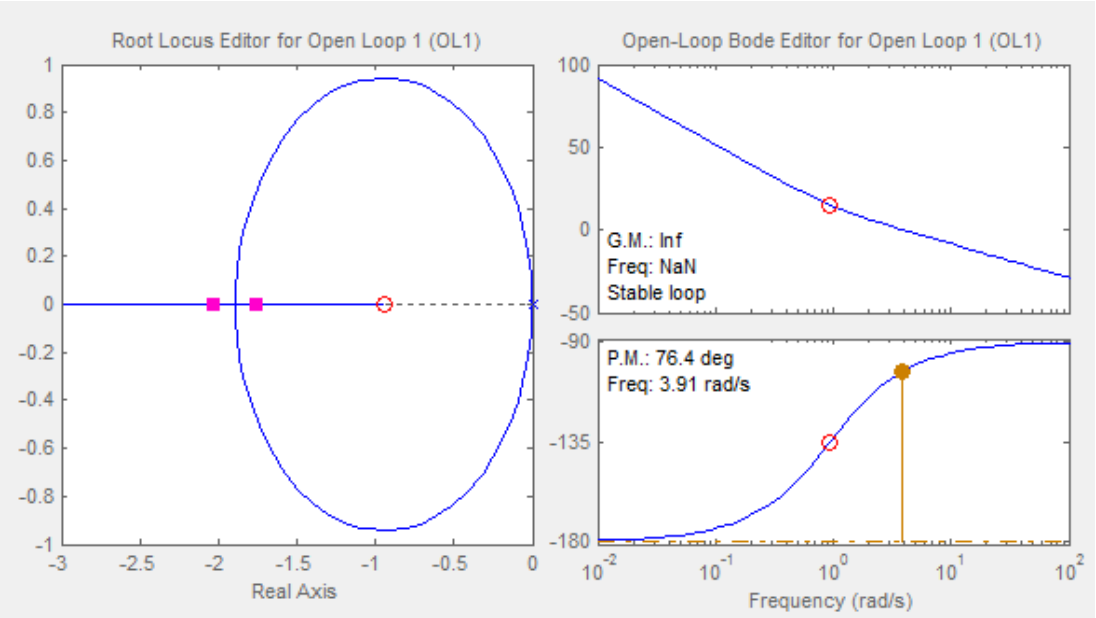
Bu durumda pervanenin oluşturduğu hava akımına ait grafik aşağıdaki gibi olacaktır.

Dikey denge uçuşu sırasında Şekil 5.7'den de anlaşılacağı üzere genellikle sabit bir Thurst kuvveti dolayısı ile sabit bir pervane dönüş hızı ve sabit bir hava akımı oluşur. SKİHA'nın rotasyonel hareket denklemlerinde bulunan hava akımının karesi ifadesi modellemedeki non-lineerlikten kurtulmak adına bu çlışma bölgesi civarında linerleştirlmiştir. Dolayısı ile SKİHA'nın rotasyonel hareketlerinin sadece kanatçıklara bağlı değiştiği kabul edilerek EK-C'de verilen matematiksel modeller elde edilmiştir.

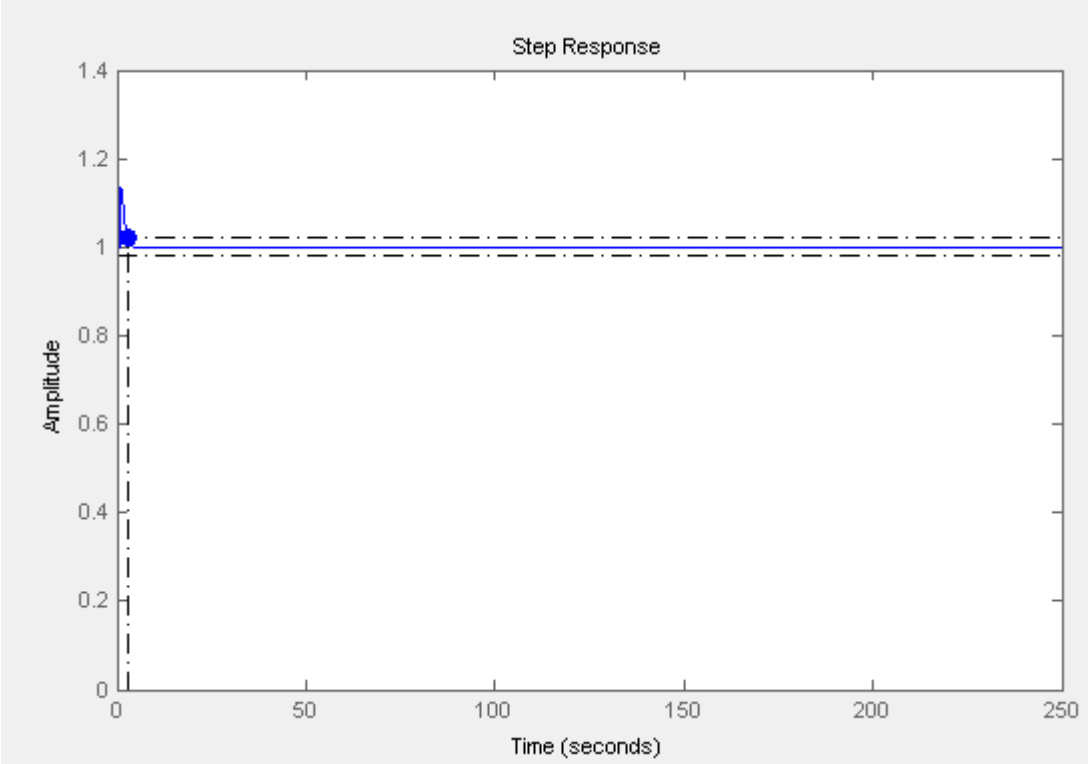


Şekil 5.7 : Dikey uçuş x konumu kontrolü sırasında pervanenin oluşturduğu hava akımı

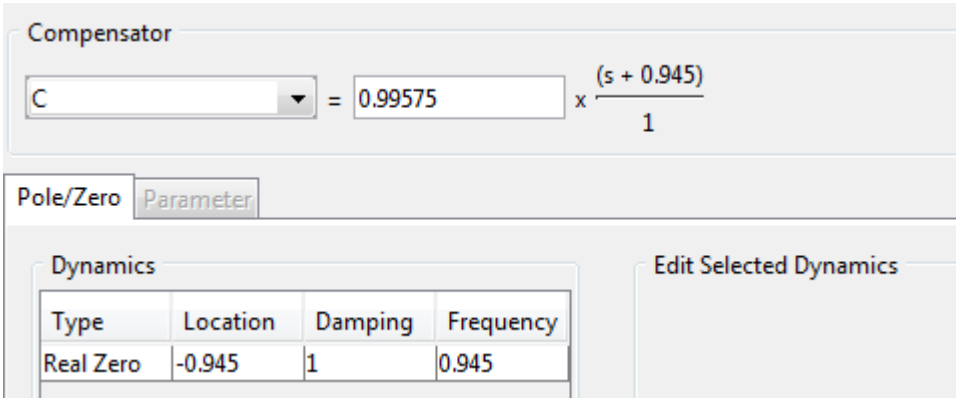
SKİHA'nın x konum kontrolünde olduğu gibi roll - pitch – yaw kontrolünde de Matlab Sisotool ve Simulink kullanılmıştır. X- konum kontrolünden farklı olarak roll – pitch – yaw kontrolünde PD kontrolör yeterli görülmüştür. Roll açısının kontrolü için tasarlanan PD kontrolör Şekil 5.8'de mevcuttur. Bu durumda roll açısının basamak cevabı Şekil 5.9'da görüldüğü gibi olacaktır, Şekil 5.10'da bu kontrolörü modeli açık bir şekilde verilmiştir. Elde ettiğimiz kontrolörü optimize ettiğimizde Şekil 5.11'de verilmiş olan PD kontrolör elde edilir. Pitch ve yaw açıları için tasarım adımları uzun uzun anlatılmamış sonuç olarak elde edilen PD kontrolöerler Şekil. 5.12 ve Şekil 5.13'de verilmiştir.



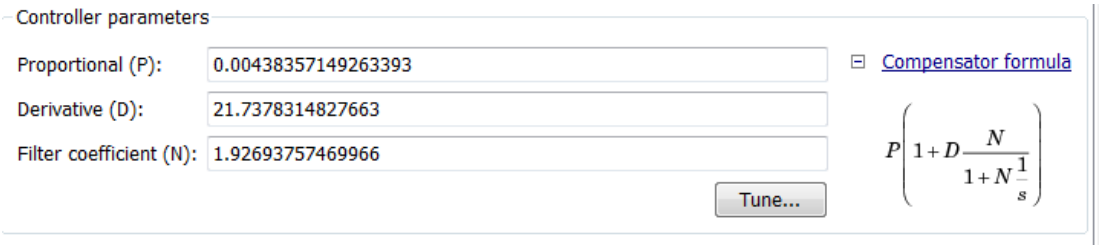
Şekil 5.8 : Roll açısı PD kontrololü köklerin yer eğrisi ve frekans analizi



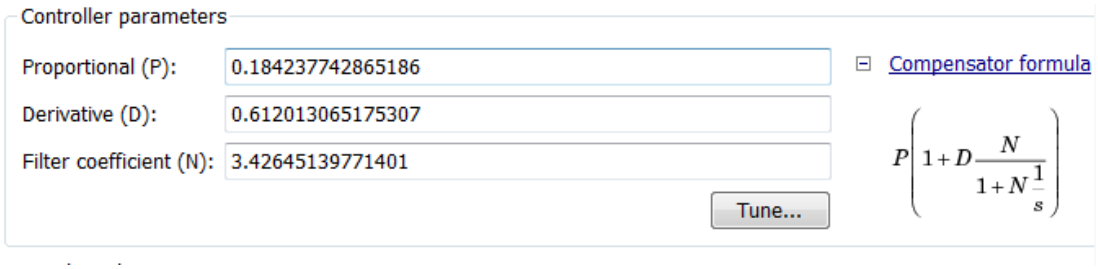
Şekil 5.9 : Roll açısı PD kontrololü zaman domeni cevabı



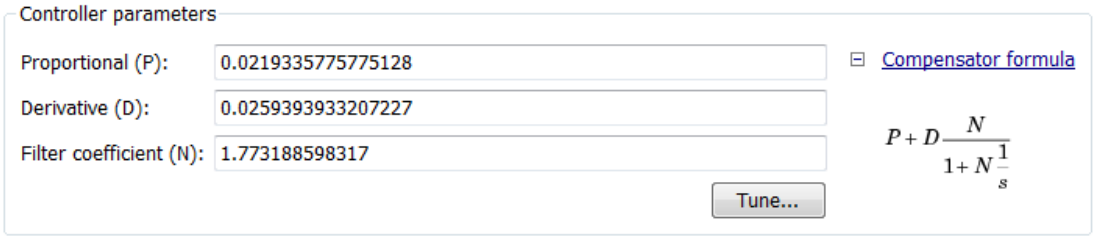
Şekil 5.10 : Sisotool roll açısı PD kontrolörü



Şekil 5.11 : Simulink otomatik ayarlamasonucu optimize edilmiş roll açısı PD kontrolörü

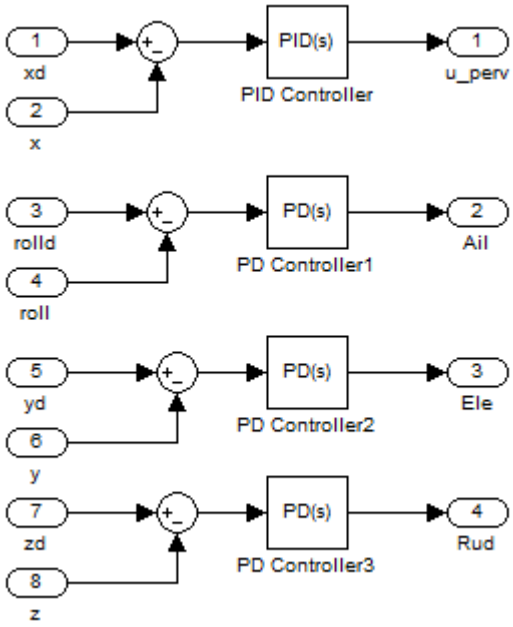


Şekil 5.12 : Simulink otomatik ayarlamasonucu optimize edilmiş pitch açısı PD kontrolörü



Şekil 5.13 : Simulink otomatik ayarlamasonucu optimize edilmiş yaw açısı PD kontrolörü

Elde ettiğimiz PID ve PD kontrolörleri model üzerinde bir alt bloğa Şekil 5.14'teki yerleştirilmiştir.

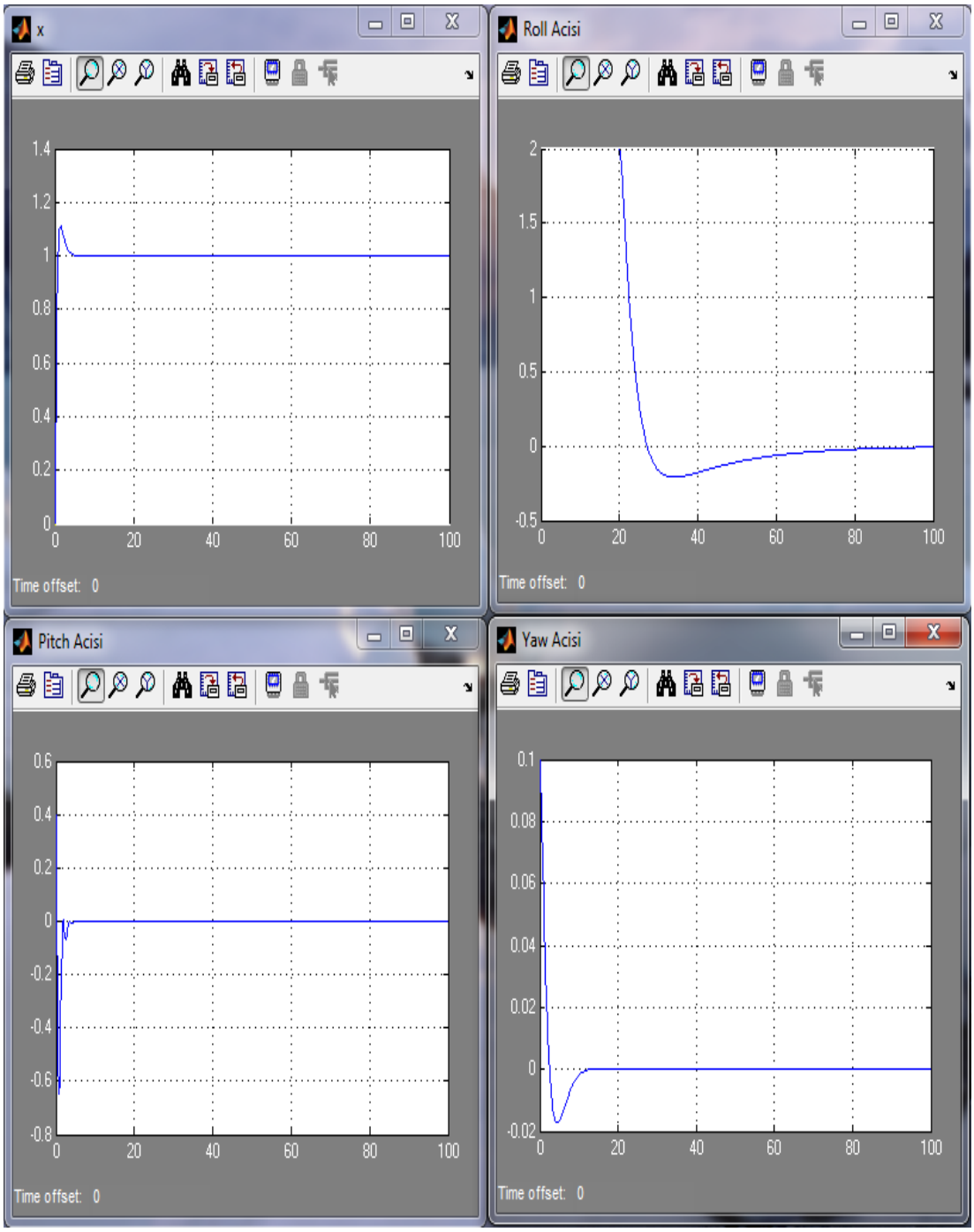


Şekil 5.14 : Simulink kontrol bloğu

Elde edilen kontrolörlerin çalışmasını test etmek amacıyla iki adet senaryo oluşturulmuştur.

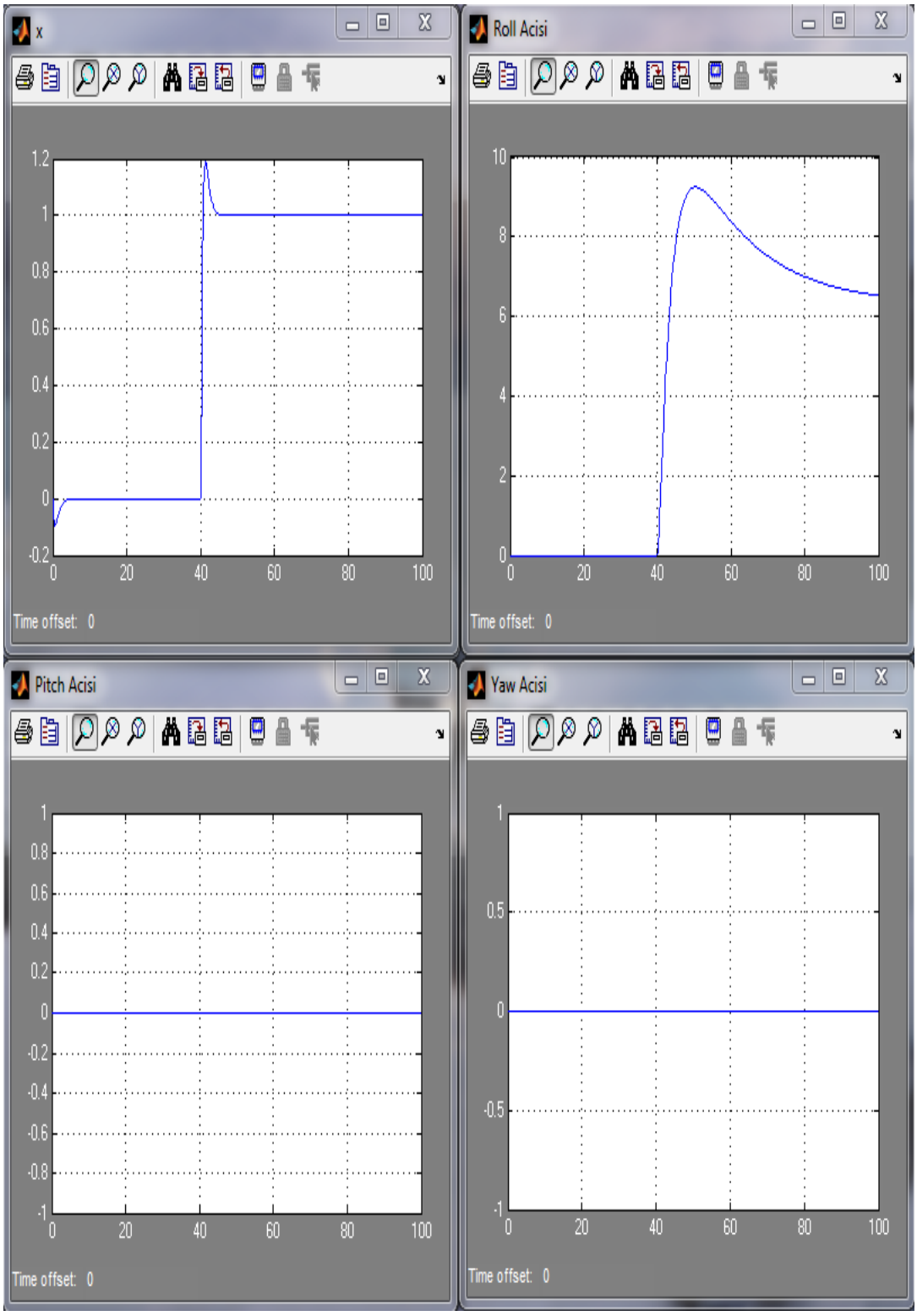
Birinci senaryoda SKİHA'nın x = 0 konumunda roll = 2 rad, pitch = 0.4 rad ve yaw = 0.1 rad başlangıç durumunda olduğu varsayılmış ve x = 1 konumunda rotasyonel bileşenleri 0 olacak şekilde dengede durmasını hedefleyen referams değerleri giirlmiştir.

Bu seneryoda yalnızca roll açışına ait referans işareti 20. Saniyede verilmişitr. Elde edilen sonuçlar şekil 5.15'te görülmektedir.



Şekil 5.15 : Senaryo 1 kontrolör sonuçları

İkinci senaryoda ise 0 konumunda 0 rotasyon ile durmakta olan bir uçağa 40. Saniyede 2π Roll yaparak 1m yükselme referansı girilmiştir. Bu senaryoya ait kontrolör sonuçları da Şekil 5.16'da mevcuttur.

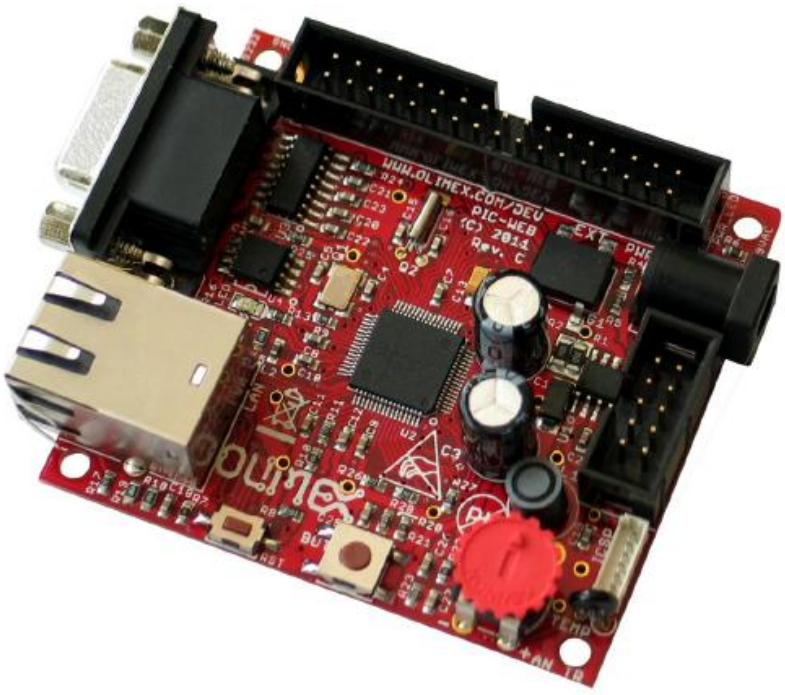


Şekil 5.16 : Senaryo 2 kontrolör sonuçları

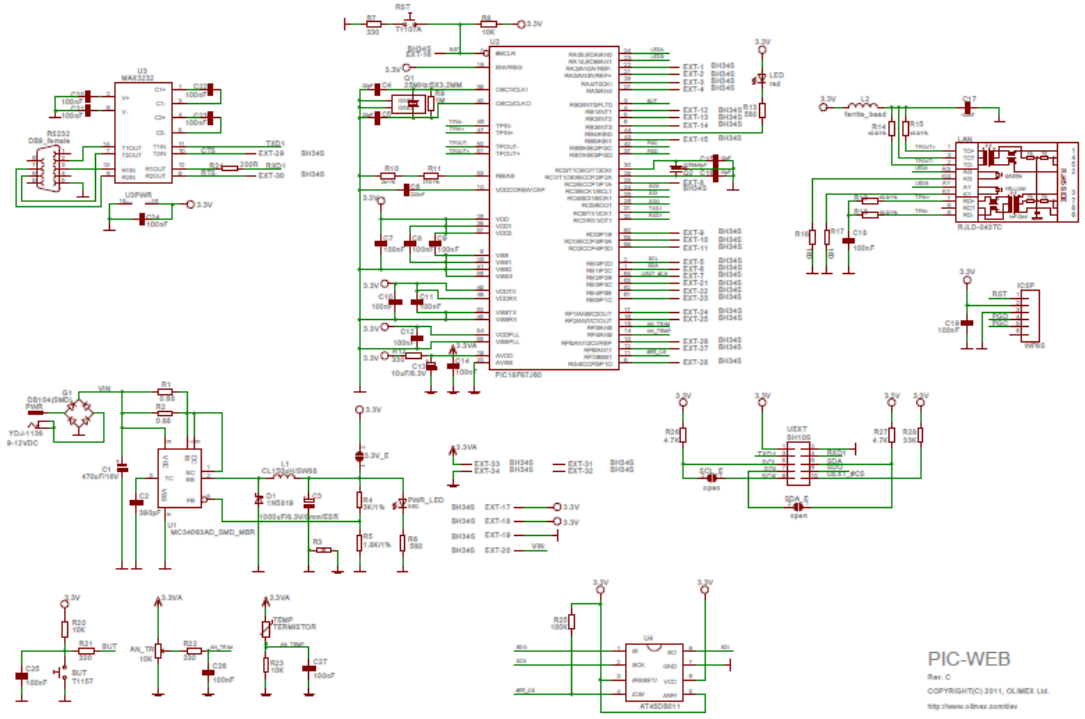
6. WEB TABANLI MİKRO DENETLEYİCİ İLE ARAYÜZ TASARIMI

Sistemin genel kontrolü amacıyla PIC 18F67J60 mikro denetleyicisi seçilmiştir. Bu mikro denetleyicinin seçilmesindeki temel amaç Ethernet ile haberleşme yapmak için fiziksel katmanının bulunmasıdır. Ayrıca Microchip firmasının gelistirmiş olduğu TCP/IP Stack sayesinde yazılımsal katmanda da kolaylık sağlanmış olur. 128 kByte'lık flash program belleği sayesinde web server olarak da çalışabilmektedir.

DIP pakette üretilmeyen bu mikro denetleyici ile kolayca çalışabilmek amacıyla Olimex firmasının geliştirmiş olduğu PIC-WEB geliştirme kartı temin edilmiştir. Bu kartın Şekil 6.1 ve Şekil 6.2'de bu kartın resmi ve devre şeması mevcuttur.



Şekil 6.1 : Olimex PIC-WEB geliştirme kartı



Şekil 6.2 : Olimex PIC-WEB devre şeması

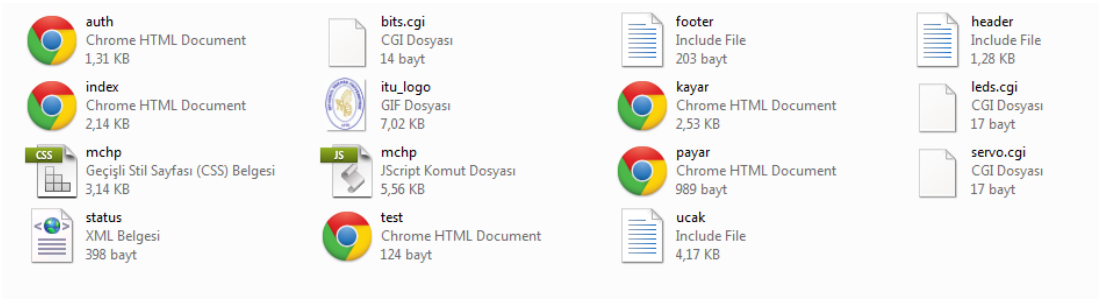
6.1 Olimex PIC-WEB kartını web server olarak kullanma

Olimex PIC-WEB kartının program belleğine istenirse web sayfası yüklenebilir. Özel bir formatta yüklenen bu web sayfası Microchip TCP/IP stack yardımıyla ekstra bir donanıma ithiyaç durmadan yayımlanabilecektir. Bunun için sırasıyla aşağıdaki işlemler yapılmalıdır.

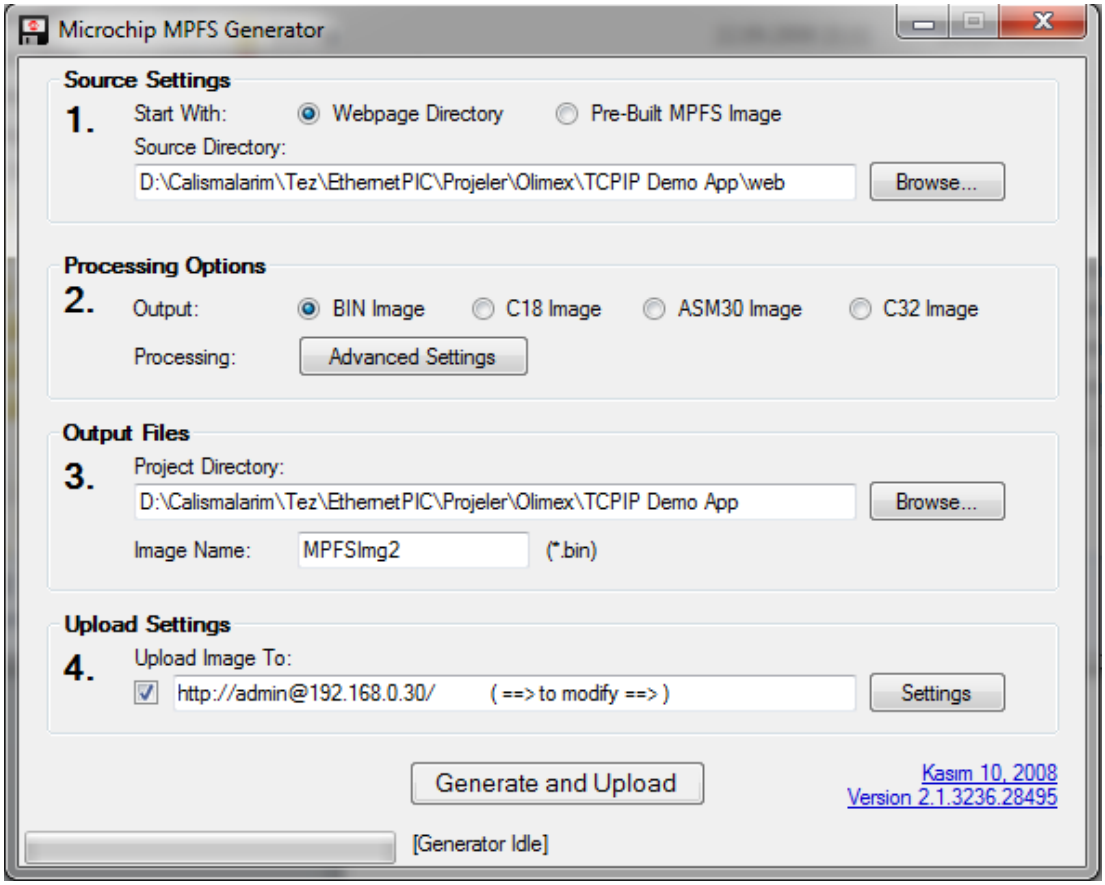
Öncelikle Html dili ile web sayfaları oluşturulur. Oluşturduğumuz web sayfalarını htm uzantısıyla kaydederiz. Donanımız 128 kByte'lık bir bellek alanı ile sırlı olduğundan mümkün olduğunca optimum bir tasarım yapılmalıdır. Bunun için Microchip TCP/IP Stack'in sunduğu ~inc: fonksiyonundan faydalınılabir. Burada temel amaç uygulamamızda birden fazla kez tekrar eden html kodlarının tekrar tekrar yazılmadan kullanılarak bellek kullanımını azaltmaktır.

Mikro denetleyicimizin içine gömeceğimiz web sayfasında görsellik sağlamak için JavaScript ve CSS dillerinden faydalanılabilir.

Web sitesini oluşturacak olan tüm bileşenler Şekil 6.3'teki gibi bir klasörde toplandıktan sonra Şekil 6.4'te görülmekte olan Microchip MPFS generator kullanılarak web sitesi mikro denetleyiciye uygun bir formatta derlenir ve yüklenir.

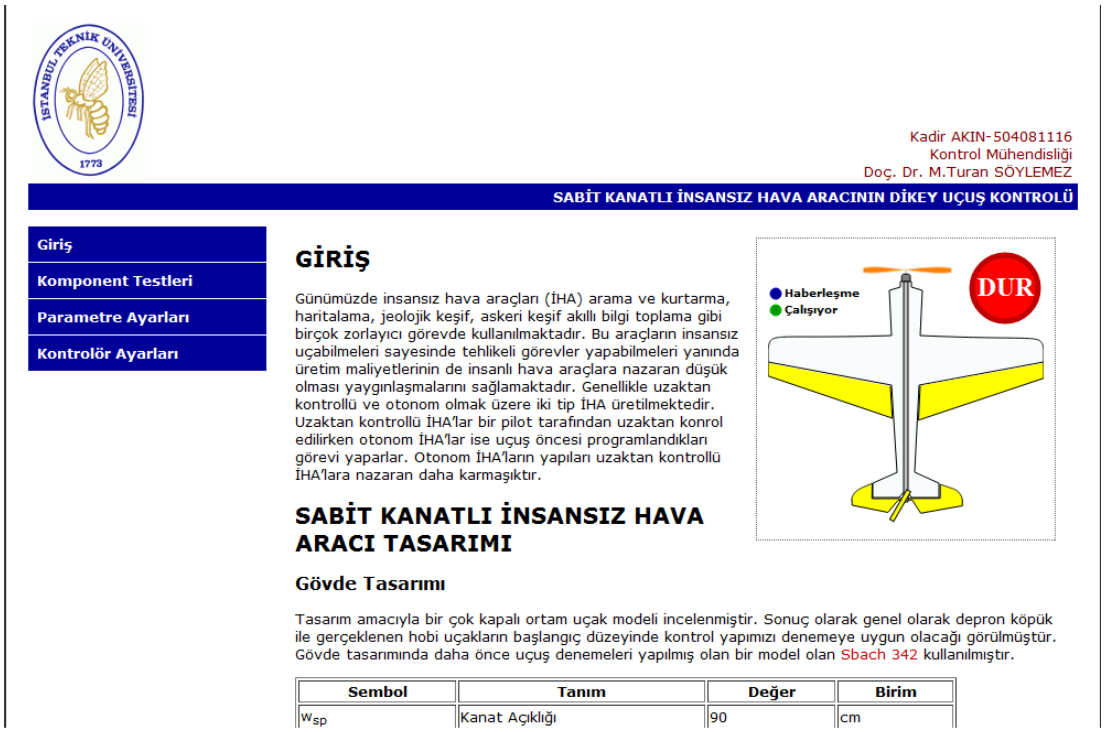


Şekil 6.3 : Oluşturulan web sitenin tüm elemanları bir klasör altına toplanmalıdır

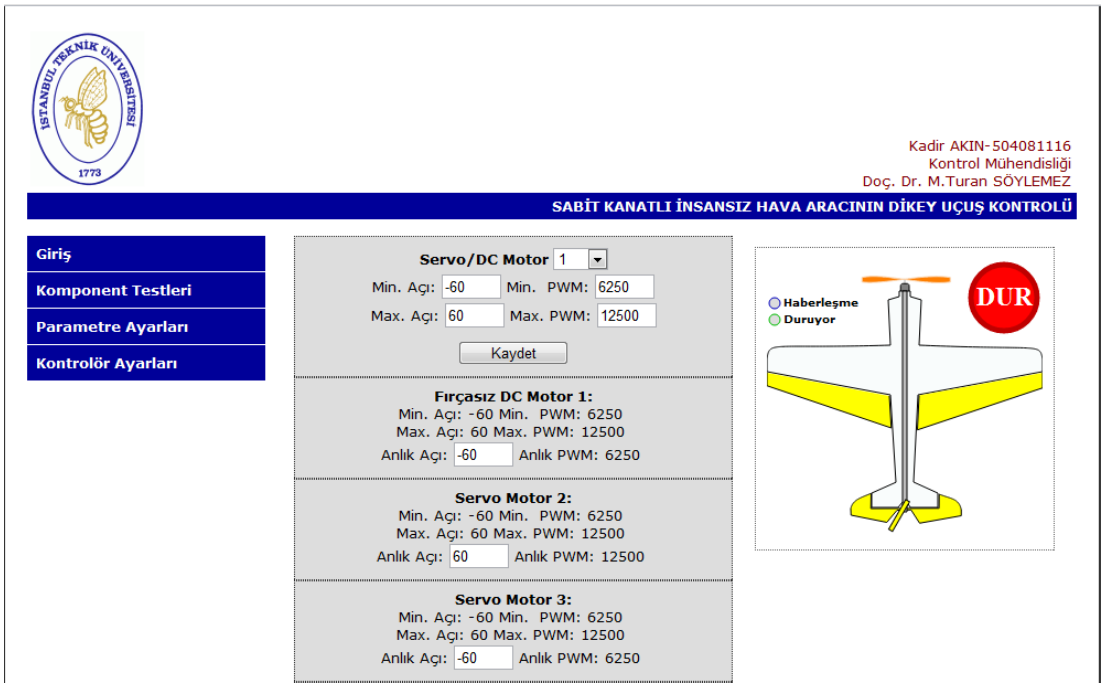


Şekil 6.4 : Microchip MPFS generator

Bu uygulamada Şekil 6.5 ve Şekil 6.6'da görülmekte olana web sayfaları oluşturulmuştur. Bu arayüz sayesinde SKİHA'I üzerindeki ekipmanların durumarı gözlenebilmekte, bu ekipmanlara açık çevrimde test ve kalibrasyon amaçlı sinyal ve veriler gönderilebilmekte ve kontrolör parametreleri ayarlana bilmektedir.



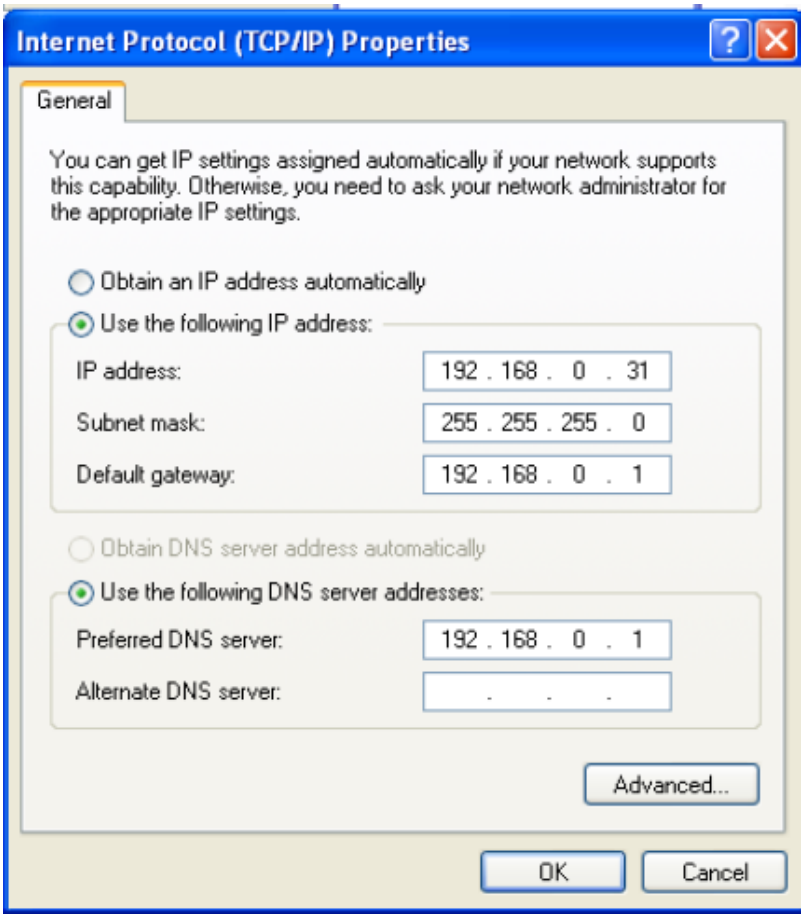
Şekil 6.5 : Micro denetleyici WEB server ana sayfa



Şekil 6.6 : Micro denetleyici WEB server ayarlama sayfası

Mikro denetleyisi içine gömülü olana bu sayfalara ulaşamak için tk yapılması gereken bilgisayarın IP ayarlarının mikro denetleyici ile aynı domaine çekilmesidir.

Bu uygulama için Şekil 6.7'de görülen ayarlar yapılabilir. Ip ayarları yapıldıktan sonra bir web tarayıcı kullanılarak 192.168.0.30 adresiye ulaşılabilir.



Şekil 6.7 : Yapılması gereken IP ayarları

Bu uygulamada tasarlanan web arayüzüne ilişkin kaynak kodlar EK-D'de verilmiştir.

7. SONUÇLAR

Bu çalışmada sabit kanatlı insansız hava aracının dikey konumda dengede uçuş modeli elde edilmiştir. Matematiksel yapı olarak bu sistemin daha basit bir benzeri olan ters sarkaç probleminden faydalanılarak sisteme ait kontrolör tasarlanmıştır. LQR yapısı kullanılarak tasarlanan kontrolöre ait sonuç matlab ortamında gözlemlenmiştir.

Kontrolörü gerçek sistemde denemek amacıylar bir model uçak yapılmış. Bu modele uygun motor, sensör seçilip elektronik kart tasarlanarak uçabilecek duruma getirilmiştir.

KAYNAKLAR

[1] **R Hugh Stone,** 2004, Control Architecture for a Tail-Sitter Unmanned Air Vehicle, School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney, NSW, Australia

[2] **William E. Green and Paul Y. Oh**, 2005, A MAV That Flies Like an Airplane and Hovers Like a Helicopter, *International Conference on Advanced Intelligent Mechatronics Monterey*, California, USA, 24-28 July, 2005

[3] **Takaaki Matsumoto, Koichi Kita, Ren Suzuki, Atsushi Oosedo,** 2010, A Hovering Control Strategy for a Tail-Sitter VTOL UAV that Increases Stability Against Large Disturbance, 2010 IEEE *International Conference on Robotics and Automation Anchorage Convention* District May 3-8, 2010, Anchorage, Alaska, USA

[4] **Ahmad Nor Kasruddin Bin Nasir**, 2007, Modeling and Controller Design for an Inverted Pendulum System, *Master Theses of Electrical Engineering*

[5] **Khalil Sultan**, 2003, Inverted Pendulum Analysis, design and Implemantation, IEEE Visionaries

[6] **Christopher Mayhew,** 2003, Robust Control of an Inverted Pendulum, Department of Electrical and Computer Engineering University of California, Santa Barbara

[7] **Roger W. Brockett**, 1970, Finite Dimentional Lineer Systems, Jhon Willey and Sons, Inc. New York - London – Sydney - Toronto

[8] **G. Basile, G.Marro**, 1969, On the Observability of Linear, Time-Invariant Systems with Unknown Inputs, *Journal Of Optimization Theory and Applications*, Vol.3 No. 6, June 1969, Belgium

[9] **Brian D. O. Anderson, John B. Moore**, 1971, Linear Optimal Control Prentice-Hall, Inc., New Jersey

[10] **Karl Johan Aström,Richard M. Murray**, 2008, Feedback Systems An Introduction For Scientists And Engineers, Princeton University Press Princeton And Oxford

[11] **Boris J. Lurie, Paul J. Enright**, 2000, Classical Feedback Control with Matlab, Marcel Dekker, Inc. New York -Basel

[12] **MEGEP**, 2006, Uçak Bakım Otomatik Uçuş, Ankara 2006

[13] **Url-2** <*http://www.pitt.edu/~sorc/robotics/handbook/ServoControl.htm*>, alındı ğı tarih 01.05.2011

[14] **Gökhan Erünlü**, 2010, Sensor Data Fusion With Kalman Filter, *Undergraduate Thesis*, İstanbul Teknik Üniversitesi 2010

[15] **D. Biermann and R. N. Conway.**, 1942, Propeller Charts for the Determination of the Rotational Speed for the Maximum Ratio of the Propeller Efficiency to the Specific Fuel Consumption., *Technical report, National Advisory Committee on Aeronautics*, 1942

[16] **Adrian Frank, James McGrew, Mario Valenti, Daniel Levine, Jonathan P. How**,2007, Hover, Transition, and Level Flight Control Design for a Single-Propeller Indoor Airplane,*Technical Report Aerospace Controls Laboratory Department of Aeronautics and Astronautics* Massachusetts Institute of Technology

EKLER

EK A : Sürücü ve Haberleşme Kartı Arayüz ve Mikro Denetleyici Kodları
EK B : AÖB dsPIC'inine Ait Kaynak Kod
EK C: SKİHA'nın modelleme parametreleri
EK D: Mikro Denetleyiciye gömülen web sayfasının kaynak kodları

EK A.

Seri Haberleşme ile bilgisayardan kontrol edilebilen haberleşme ve motor sürücü kartına ilişkin kaynak kod aşağıda verilmiştir.

```
#include <pic.h>
#include "delay.h"
__CONFIG(0x3F0A);

bank1 unsigned char counter=0, i=0, j=0, s1_ters = 0, s2_ters = 0, servo_aci[8] =
{0,0,0,0,0,0,0,0}, gelen_veri = 0;
bank1 unsigned int servo_sureler[16];

void main(void){
   TRISA = 0xFF;
   TRISB = 0b00000111;

   TXSTA = 0b00100100;
   RCSTA = 0b10010000;
   SPBRG = 64;

   GIE=1;
   PEIE=1;
   TMR1IE=1;
   TMR1ON=1;
   RCIE = 1;
   while(1)
   {
      for(i=0;i<8;i++){
         servo_sureler[2*i] = (60536 - (servo_aci[i]*50) + 33);
         servo_sureler[2*i+1] = (48036 + (servo_aci[i]*50) + 33);
      }

   }
}

void interrupt abc(void){
// Timer Interrupt
   if(TMR1IF){
      TMR1IF=0;
      TMR1H=(servo_sureler[counter])>>8;
      TMR1L=servo_sureler[counter];

      if(counter == 0) RB3 = 1;
      else if(counter == 1) RB3 = 0;
      else if(counter == 2) RB4 = 1;
      else if(counter == 3) RB4 = 0;
      else if(counter == 4) RB5 = 1;
      else if(counter == 5) RB5 = 0;
      else if(counter == 6) RB6 = 1;
```

```
    else if(counter == 7) RB6 = 0;
    else if(counter == 8) RB7 = 1;
    else if(counter == 9) RB7 = 0;

    if(++counter == 16) counter = 0;

}

if(RCIF){
    gelen_veri = RCREG;

    if(gelen_veri>=100){
        if(j == 10){
            servo_aci[0] = gelen_veri - 100;
        }else if(j == 20){
            servo_aci[1] = gelen_veri - 100;
            if(s1_ters==0) servo_aci[2] = gelen_veri - 100;
            else servo_aci[2] = 200 - gelen_veri;
        }else if(j == 30){
            servo_aci[3] = gelen_veri - 100;
            if(s2_ters==0) servo_aci[4] = gelen_veri - 100;
            else servo_aci[4] = 200 - gelen_veri;
        }
        j = 0;
    }else if(gelen_veri == 10){
        j = 10;
    }else if(gelen_veri == 20){
        j = 20;
    }else if(gelen_veri == 30){
        j = 30;
    }else if(gelen_veri == 21){
        s1_ters = 0;
        servo_aci[2] = servo_aci[1];
        j = 0;
    }else if(gelen_veri == 22){
        s1_ters = 1;
        servo_aci[2] = 100 - servo_aci[1];
        j = 0;
    }else if(gelen_veri == 31){
        s2_ters = 0;
        servo_aci[4] = servo_aci[3];
        j = 0;
    }else if(gelen_veri == 32){
        s2_ters = 1;
        servo_aci[4] = 100 - servo_aci[3];
        j = 0;
    }

    i = 50;
    while(--i);
```

```
        while(!TXIF);
        TXREG = gelen_veri;
        }


}
```

Bilgisayarda Matlab üzrinden PIC ile haberleşen arayüze ait kaynak kod aşağıdaki gibidir.

```
function varargout = arayuz(varargin)
% ARAYUZ M-file for arayuz.fig%      ARAYUZ, by itself, creates a new
ARAYUZ or raises the existing
%      singleton*.
%
%      H = ARAYUZ returns the handle to a new ARAYUZ or the handle to
%      the existing singleton*.
%
%      ARAYUZ('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ARAYUZ.M with the given input arguments.
%
%      ARAYUZ('Property','Value',...) creates a new ARAYUZ or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before arayuz_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to arayuz_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help arayuz

% Last Modified by GUIDE v2.5 19-Apr-2011 03:13:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @arayuz_OpeningFcn, ...
                   'gui_OutputFcn',  @arayuz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before arayuz is made visible.
function arayuz_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to arayuz (see VARARGIN)

% Choose default command line output for arayuz
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes arayuz wait for user response (see UIRESUME)
% uiwait(handles.figure1);

handles.com = serial('COM5','BaudRate',19200);
set(handles.com,'TimeOut',5);
set(handles.com,'InputBufferSize',1);
% get(handles.com)
fopen(handles.com);
fprintf('Port Açıldı\n')

veri_gonder(handles.com,22);
veri_gonder(handles.com,32);
veri_gonder(handles.com,20);
veri_gonder(handles.com,150);
veri_gonder(handles.com,30);
veri_gonder(handles.com,150);

% fprintf(handles.com, '*IDN?')
% idn = fscanf(handles.com);
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = arayuz_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;

function veri_gonder(file,value)
% Veri gönderme fonksiyonu
fwrite(file,value);
read_val = fread(file);
if(read_val ~= value)
fprintf('ERROR\n');
else
fprintf('%d\n',value);
end




% --- Executes on slider movement.
function S1_slider_Callback(hObject, eventdata, handles)
% hObject    handle to S1_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
v = get(handles.S1_slider,'Value');
v = round(v);
set(handles.S1_edit,'String',v);
veri_gonder(handles.com,20);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function S1_slider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to S1_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function S2_slider_Callback(hObject, eventdata, handles)
% hObject    handle to S2_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
```

```
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
v = get(handles.S2_slider,'Value');
v = round(v);
set(handles.S2_edit,'String',v);
veri_gonder(handles.com,30);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function S2_slider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to S2_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function FDC_slider_Callback(hObject, eventdata, handles)
% hObject    handle to FDC_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
v = get(handles.FDC_slider,'Value');
v = round(v);
set(handles.FDC_edit,'String',v);
veri_gonder(handles.com,10);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function FDC_slider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FDC_slider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function S1_edit_Callback(hObject, eventdata, handles)
% hObject    handle to S1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of S1_edit as text
%        str2double(get(hObject,'String')) returns contents of S1_edit as a double
v = str2double(get(handles.S1_edit,'String'));
if(isnan(v))
   v = get(handles.S1_slider,'Value');
elseif(v>100)
   v = 100;
elseif v<0
   v = 0;
end
v = round(v);
set(handles.S1_slider,'Value',v);
set(handles.S1_edit,'String',v);
veri_gonder(handles.com,20)
veri_gonder(handles.com,100+v)
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function S1_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to S1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function S2_edit_Callback(hObject, eventdata, handles)
% hObject    handle to S2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of S2_edit as text
%        str2double(get(hObject,'String')) returns contents of S2_edit as a double
v = str2double(get(handles.S2_edit,'String'));
if(isnan(v))
   v = get(handles.S2_slider,'Value');
elseif(v>100)
   v = 100;
elseif v<0
```

```
    v = 0;
end
v = round(v);
set(handles.S2_slider,'Value',v);
set(handles.S2_edit,'String',v);
veri_gonder(handles.com,30);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function S2_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to S2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FDC_edit_Callback(hObject, eventdata, handles)
% hObject    handle to FDC_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FDC_edit as text
%        str2double(get(hObject,'String')) returns contents of FDC_edit as a double
v = str2double(get(handles.FDC_edit,'String'));
if(isnan(v))
    v = get(handles.FDC_slider,'Value');
elseif(v>100)
    v = 100;
elseif v<0
    v = 0;
end
v = round(v);
set(handles.FDC_slider,'Value',v);
set(handles.FDC_edit,'String',v);
veri_gonder(handles.com,10);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function FDC_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FDC_edit (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in S1_togglebutton.
function S1_togglebutton_Callback(hObject, eventdata, handles)
% hObject    handle to S1_togglebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of S1_togglebutton
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
   veri_gonder(handles.com,21);
   set(hObject,'String', 'Grup Düz');
elseif button_state == get(hObject,'Min')
   veri_gonder(handles.com,22);
   set(hObject,'String', 'Grup Ters');
end

% --- Executes on button press in S2_togglebutton.
function S2_togglebutton_Callback(hObject, eventdata, handles)
% hObject    handle to S2_togglebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of S2_togglebutton
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
   veri_gonder(handles.com,31);
   set(hObject,'String', 'Grup Düz');
elseif button_state == get(hObject,'Min')
   veri_gonder(handles.com,32);
   set(hObject,'String', 'Grup Ters');
end

% --- Executes on button press in S1_radiobutton.
function S1_radiobutton_Callback(hObject, eventdata, handles)
% hObject    handle to S1_radiobutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of S1_radiobutton
```

```
% --- Executes on button press in S2_radiobutton.
function S2_radiobutton_Callback(hObject, eventdata, handles)
% hObject    handle to S2_radiobutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of S2_radiobutton


% --- Executes on button press in FDC_radiobutton.
function FDC_radiobutton_Callback(hObject, eventdata, handles)
% hObject    handle to FDC_radiobutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of FDC_radiobutton


% --- Executes on button press in S1_pushbutton.
function S1_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to S1_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
v = get(handles.S1_slider,'Value');
v = round(100 - v);
set(handles.S1_slider,'Value',v);
set(handles.S1_edit,'String',v);
veri_gonder(handles.com,20);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);



% --- Executes on button press in S2_pushbutton.
function S2_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to S2_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
v = get(handles.S2_slider,'Value');
v = round(100 - v);
set(handles.S2_slider,'Value',v);
set(handles.S2_edit,'String',v);
veri_gonder(handles.com,30);
veri_gonder(handles.com,100+v);
guidata(hObject, handles);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
```

```
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
fclose(handles.com);
delete(handles.com);
guidata(hObject, handles);
fprintf('Port Kapatıldı')
delete(hObject);
```

EK B.

```
#include "p33FJ128MC804.h"
#include "dsp.h"
#include <stdio.h>

#define UART_BUFFER_SIZE 100 //UART verilerini yollamak için kullanılan fonksiyonun kaç veri tutabileceği
#define TO 0.02 //sampling time
#define TOTAL_SIMULATION_TIME 4 //total simulation time
#define pi 3.14
   //Radyan cinsinden kalibre edicen. 10mv 1 dereceye karşılık gelse, 1 derece de 3.14/180.0 = 0.01744 radyana denk geldiğinden
   //1mv da 0.001744 radyana denk gelir, 1 bit yaklaşık 1mV olsa;
#define GYRO_X_CALIBRATION 0.003314 //DİKKAT! bu değer atmasyon!
#define GYRO_Y_CALIBRATION 0.003314 //DİKKAT! bu değer atmasyon!
#define GYRO_Z_CALIBRATION 0.003314 //DİKKAT! bu değer atmasyon!
#define OPERATION_TIME_STEP 0.001
#define ACCUMULATION_COUNT 100

#define TIMER1_PERIOD_R 156
#define TIMER2_PERIOD_R 1000
#define TIMER3_PERIOD_R 1000
#define TIMER4_PERIOD_R 1000
#define TIMER5_PERIOD_R 1000

int MEASUREMENT__ACCELEROMETER_X = 1;
int MEASUREMENT__ACCELEROMETER_Y = 2;
int MEASUREMENT__ACCELEROMETER_Z = 3;
int MEASUREMENT__GYRO_X = 4;
int MEASUREMENT__GYRO_Y = 5;
int MEASUREMENT__GYRO_Z = 6;
unsigned char GL_Thraed_1Hz = 0;
unsigned char GL_Thraed_50Hz = 0;
unsigned char GL_UART_Buffer[UART_BUFFER_SIZE] = {0};
unsigned char GL_UART_Buffer_Finish = 0;
unsigned char GL_UART_Buffer_Start = 0;
unsigned char GL_UART_Ek_Veri[50] = {0};
unsigned char GL_UART_Ek_Veri_Size = 0;
unsigned int GL_ADC_Olcum_UI16[8] __attribute__((space(dma),aligned(16)));
   //Allocate 32 word (64 byte) from the DMA area
unsigned int BufferA[4][8] __attribute__((space(dma),aligned(64)));
unsigned int BufferB[4][8] __attribute__((space(dma),aligned(64)));
unsigned int GL_Switch__Velocity_Calculate_UI16 = 0; //EU_CASE__WAIT
unsigned int GL_Accumulator_Counter_UI16 = 0;
float GL_ADC_Olcum_Milivolt_Offset[8] = {0};
float GL_ADC_Olcum_Milivolt_Accumulator[8] = {0}; //
```

```
float GL_ADC_Olcum_Milivolt[8] = {0}; //
float GL_Operation_Time = 0; // Hassasiyet: Operation_Time_Step = 0.001 saniye

//if Simulation_Time == TOTAL_SIMULATION_TIME -> Simulation ends
//else -> Simulation_Time += TO
float Simulation_Time = 0.0; //recent simulation time

//arrays of angular velocity measurements
float measurements_w_alfa[] = {0.1, 0.0, 0.0, 0.0, 0.0};
float measurements_w_beta[] = {0.0, 0.1, 0.0, 0.0, 0.0};
float measurements_w_gama[] = {0.0, 0.0, 0.4, 0.7, 0.2};

//angular velocities
float GL_W_alfa;
float GL_W_beta;
float GL_W_gama;

//angular rotations (angular pozition change) per sempling
float GL_R_alfa;
float GL_R_beta;
float GL_R_gama;

//new Rotation matrix
float R_zyx[4][4] = {{0}};

//we do not use the firtst elements of the rotation matrix
//matrix of total rotations, starting value is identity matrix
float R[4][4] = {{0,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}};
float R_ex[4][4] = {{0}};

//estimation of Euler angles
float GL_alfa_Estimation[3];
float GL_beta_Estimation[3];
float GL_gama_Estimation[3];
//*****************************

void ADC_Ayarlarini_Yap( void );

void DMA0_Ayarlarini_Yap(void);

void Portlari_Ayarla( void );

void SPI_Ayarlarini_Yap( void );

void Timer1_Ayarlarini_Yap( void );

void UART_Ayarlarini_Yap( void );

void Calculate_New_Angular_Velocities();
```

```
void Take_New_Angular_Velocities();

void Calculate_Rotation_Matrix_Rzyx();

void Calculate_Euler_Angles_From_Rotation_Matrix();

void Multiply_3_3_Matrix(float M1[4][4],float M2[4][4],float M_[4][4]);

void Copy_3_3_Matrix(float M1[4][4], float M2[4][4]);

void Print_3_3_Matrix(float M[4][4]);

void UART_Veri_Depola_Ve_Yolla( void );

_FOSCSEL(FNOSC_FRC); // Internal FRC Oscillator

_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF  & POSCMD_XT); //  Clock Switching is enabled

_FWDT(FWDTEN_OFF);  // Watchdog Timer Enabled/disabled by user software

int main(void)
{
   unsigned long Baslangic_Sayaci_UI32 = 2000000; //for döngüsü ile kullanıldığında yaklaşık yarım saniye beklemek için 2milyona kadar say
   unsigned int i = 0;
   unsigned int j = 0;

   PLLFBD=237;
   CLKDIVbits.PLLPOST = 0;
   CLKDIVbits.PLLPRE = 9;

   __builtin_write_OSCCONH(0x01);      // Clock switch to incorporate PLL // Initiate Clock Switch to
   __builtin_write_OSCCONL(0x01);      // FRC with PLL (NOSC=0b001) // Start clock switching
   while (OSCCONbits.COSC != 0b001);   // Wait for Clock switch to occur
   while(OSCCONbits.LOCK!=1) {};   // Wait for PLL to lock
   OSCCONL = 0x46;
   OSCCONL = 0x57;
   OSCCONbits.IOLOCK = 0;
   Portlari_Ayarla();
   OSCCONL = 0x46;
   OSCCONL = 0x57;
   OSCCONbits.IOLOCK = 1;
   ADC_Ayarlarini_Yap();
   DMA0_Ayarlarini_Yap();
   Timer1_Ayarlarini_Yap();
   UART_Ayarlarini_Yap();
   IPC0bits.T1IP = 1;
```

```
IPC1bits.T2IP = 2;
IPC2bits.T3IP = 3;
IPC6bits.T4IP = 4;
IPC7bits.T5IP = 5;
IPC3bits.U1TXIP = 1;
IFS0bits.T1IF = 0;
IFS0bits.T2IF = 0;
IFS0bits.T3IF = 0;
IFS1bits.T4IF = 0;
IFS1bits.T5IF = 0;
IFS0bits.DMA0IF = 0;
IEC0bits.T1IE = 1;
IEC0bits.T2IE = 0;
IEC0bits.T3IE = 0;
IEC1bits.T4IE = 0;
IEC1bits.T5IE = 0;
IEC0bits.U1TXIE = 1;
IEC0bits.DMA0IE = 1;

//Baslangıç değerlerini ata
LATCbits.LATC7 = 1;//==
LATCbits.LATC6 = 0;//==

while(1)
{
   if( GL_Thraed_50Hz == 1 )
   {
      LATCbits.LATC7 ^= 1;//==
      LATCbits.LATC6 ^= 1;//==

      Take_New_Angular_Velocities();

      GL_R_alfa = GL_W_alfa * TO;
      GL_R_beta = GL_W_beta * TO;
      GL_R_gama = GL_W_gama * TO;

      Calculate_Rotation_Matrix_Rzyx();

      Copy_3_3_Matrix(R_ex,R);
      Multiply_3_3_Matrix(R_ex,R_zyx,R);//R=R_ex*R_zyx;

      Simulation_Time = Simulation_Time + TO;

      GL_Thraed_50Hz = 0;
   }//endif thread 50Hz

   if((GL_Thraed_1Hz == 1) && (GL_Operation_Time >= 5.0))
   {
      Calculate_Euler_Angles_From_Rotation_Matrix();
```

```
            GL_UART_Ek_Veri_Size = sprintf( GL_UART_Ek_Veri, "\n %.2f %.2f
%.4f\r\n",GL_alfa_Estimation[1],GL_beta_Estimation[1],GL_gama_Estimation[1]*
57.32);
            UART_Veri_Depola_Ve_Yolla();

            GL_Thraed_1Hz = 0;
        }

    }

    return 0;
}


//ADC_Ayarlarini_Yap----------------------------------------------------------
ADC_Ayarlarini_Yap---------------
void ADC_Ayarlarini_Yap( void )
{

    AD1CON1bits.ADON = 0;
    AD1CON1bits.ADSIDL = 0;
    AD1CON1bits.ADDMABM = 0;
    AD1CON1bits.AD12B = 1;
    AD1CON1bits.FORM = 0b00;
    AD1CON1bits.SSRC = 0b111;
    AD1CON1bits.ASAM = 1;
    AD1CON2bits.VCFG = 0b000;
    AD1CON2bits.CSCNA = 1;
    AD1CON2bits.SMPI = 3;
    AD1CON2bits.BUFM = 0;
    AD1CON2bits.ALTS = 0;
    AD1CON3bits.ADRC = 0;
    AD1CON3bits.SAMC = 0b00011;
    AD1CON3bits.ADCS = 0b00111111;
    AD1CON4bits.DMABL = 0b011;
    AD1CHS0bits.CH0NA = 0;
    //Bizim kullandıklarımız: AN1,4,5,6,7,8
    AD1CSSLbits.CSS0 = 0;
    AD1CSSLbits.CSS1 = 0;
    AD1CSSLbits.CSS2 = 0;
    AD1CSSLbits.CSS3 = 0;
    AD1CSSLbits.CSS4 = 0;
    AD1CSSLbits.CSS5 = 0;
    AD1CSSLbits.CSS6 = 0;
    AD1CSSLbits.CSS7 = 0;
    AD1CSSLbits.CSS8 = 1;
    AD1PCFGLbits.PCFG0 = 0;
    AD1PCFGLbits.PCFG1 = 0;
    AD1PCFGLbits.PCFG2 = 0;
    AD1PCFGLbits.PCFG3 = 0;
```

```
    AD1PCFGLbits.PCFG4 = 0;
    AD1PCFGLbits.PCFG5 = 0;
    AD1PCFGLbits.PCFG6 = 0;
    AD1PCFGLbits.PCFG7 = 0;
    AD1PCFGLbits.PCFG8 = 0;
    IFS0bits.AD1IF = 0; // Clear the A/D interrupt flag bit
    IEC0bits.AD1IE = 0; // Do Not Enable A/D interrupt
    AD1CON1bits.ADON = 1;   //ADC modülünü aktif et

}

void DMA0_Ayarlarini_Yap(void)
{

    DMA0CONbits.AMODE = 2; // Configure DMA for Peripheral indirect mode
//==
    DMA0CONbits.MODE = 2; // Configure DMA for Continuous Ping-Pong mode
    DMA0PAD = 0x0300; // Point DMA to ADC1BUF0
    DMA0CNT = 31; // 32 DMA request
    DMA0REQ = 13; // Select ADC1 as DMA Request source

    DMA0STA = __builtin_dmaoffset(&BufferA);
    DMA0STB = __builtin_dmaoffset(&BufferB);

        //CHEN: Channel Enable bit
        //1 = Channel enabled
        //0 = Channel disabled
    DMA0CONbits.CHEN=1; // Enable DMA Channel

}

//Portlari_Ayarla-------------------------------------------------------------
Portlari_Ayarla-----------------------
void Portlari_Ayarla(  )
{
    TRISA = 0;
    TRISB = 0;
    TRISC = 0;
    LATA = 0;
    LATB = 0;
    LATC = 0;

    //Input-Output
Settings*****************************************************
    //Bizim kullandıklarımız: AN1,4,5,6,7,8
    _TRISA0 = 1;
    _TRISA1 = 1;
    _TRISB0 = 1;
    _TRISB1 = 1;
    _TRISB2 = 1;
```

```
    _TRISB3 = 1;
    _TRISC0 = 1;
    _TRISC1 = 1;
    _TRISC2 = 1;

    //pin41:RB5:RP5: UART_TX (output)
    //_LATB5 = 0; //output
    //RPn tied to UART1 Transmit
    RPOR0bits.RP0R0 = 0b1;
    RPOR0bits.RP0R1 = 0b1;
    RPOR0bits.RP0R2 = 0b0;
    RPOR0bits.RP0R3 = 0b0;
    RPOR0bits.RP0R4 = 0b0;

}

//Timer1_Ayarlarini_Yap-----------------------------------------------------------------
Timer1_Ayarlarini_Yap------------------------
void Timer1_Ayarlarini_Yap(  )
{

    T1CONbits.TON = 1;
    T1CONbits.TSIDL = 0;
    T1CONbits.TGATE = 0;
    T1CONbits.TCKPS = 0b11;
    T1CONbits.TSYNC = 0;
    T1CONbits.TCS = 0;
    TMR1 = 0x00;
    PR1 = TIMER1_PERIOD_R; //

}

//UART_Ayarlarini_Yap-------------------------------------------------------------------
UART_Ayarlarini_Yap------------------------
void UART_Ayarlarini_Yap(  )
{

    U1MODEbits.UARTEN = 1;
    U1MODEbits.UEN = 0b00;//bu ayarı sonradan ekledim
    U1MODEbits.ABAUD = 0;
    U1MODEbits.BRGH = 0;
    U1MODEbits.PDSEL = 0b00; //no parity
    U1MODEbits.STSEL = 0; //one stop bit
    U1STAbits.UTXISEL0 = 0b0;
    U1STAbits.UTXISEL1 = 0b0;
    U1STAbits.UTXEN = 1; //U1TXIF'i set eder
    U1BRG = ((40000000/38400)/16) - 1; //baud rate is 38400

}
```

```
//Fonksiyonlar************************************************************
************************************************************
void Calculate_New_Angular_Velocities()
{

   GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_X] = 0.0; //== Sensörden
okunan değerler
   GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_Y] = 0.0; //==
   GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_Z] = ADC1BUF0;
//==ADC1BUF0

   switch( GL_Switch__Velocity_Calculate_UI16 )
   {
      case 0:
      {
         //Gyronun kendine gelmesini bekle
         if(GL_Operation_Time >= 2.0)
         {
            GL_Switch__Velocity_Calculate_UI16 = 1;
         }

         break;
      }
      case 1:
      {
         GL_ADC_Olcum_Milivolt_Accumulator[MEASUREMENT__GYRO_Z]
+= GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_Z];

         //Eğer sonuncu ölçümün akümülasyon işlemiyse
         GL_Accumulator_Counter_UI16++;
         if( GL_Accumulator_Counter_UI16 == ACCUMULATION_COUNT)
         {
            GL_Switch__Velocity_Calculate_UI16 = 2;
         }
         break;
      }
      case 2:
      {
         GL_ADC_Olcum_Milivolt_Offset[MEASUREMENT__GYRO_Z] =
GL_ADC_Olcum_Milivolt_Accumulator[MEASUREMENT__GYRO_Z]/ACCUM
ULATION_COUNT;

         GL_Switch__Velocity_Calculate_UI16 = 3;
         break;
      }
      case 3:
      {
         GL_W_alfa = (GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_X] -
GL_ADC_Olcum_Milivolt_Offset[MEASUREMENT__GYRO_X])*GYRO_X_CA
LIBRATION;
```

```
        GL_W_beta = (GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_Y] -
GL_ADC_Olcum_Milivolt_Offset[MEASUREMENT__GYRO_Y])*GYRO_Y_CA
LIBRATION;
        GL_W_gama = (GL_ADC_Olcum_UI16[MEASUREMENT__GYRO_Z] -
GL_ADC_Olcum_Milivolt_Offset[MEASUREMENT__GYRO_Z])*GYRO_Z_CA
LIBRATION;

      break;
      }
      default:
      {

      break;
      }

   }//end_switch

}

void Take_New_Angular_Velocities()
{

   int Measurement_No = (int)(Simulation_Time/TO);

   Calculate_New_Angular_Velocities();

   //GL_W_alfa = measurements_w_alfa[Measurement_No]; //==
   //GL_W_beta = measurements_w_beta[Measurement_No]; //==
   //GL_W_gama = measurements_w_gama[Measurement_No]; //==

}

void Calculate_Rotation_Matrix_Rzyx()
{
   float Ca,Sa,Cb,Sb,Cg,Sg;

   Ca = cos(GL_R_alfa);
   Sa = sin(GL_R_alfa);
   Cb = cos(GL_R_beta);
   Sb = sin(GL_R_beta);
   Cg = cos(GL_R_gama);
   Sg = sin(GL_R_gama);

   R_zyx[1][1] = Cb*Cg;
   R_zyx[1][2] = Cg*Sa*Sb-Ca*Sg;
   R_zyx[1][3] = Ca*Cg*Sb+Sa*Sg;
   R_zyx[2][1] = Cb*Sg;
   R_zyx[2][2] = Ca*Cg+Sa*Sb*Sg;
   R_zyx[2][3] = -Cg*Sa+Ca*Sb*Sg;
   R_zyx[3][1] = -Sb;
```

```
   R_zyx[3][2] = Cb*Sa;
   R_zyx[3][3] = Ca*Cb;

}

void Calculate_Euler_Angles_From_Rotation_Matrix()
{
   //Find Euler Angles from given Rotation Matrix
   //Given rotation marix is R

   //if cos(beta)!=0
   //if((R[3][1] != 1) && (R[3][1] != -1))
   if((R[3][1] <= 0.99) && (R[3][1] >= -0.99))
   {
      //calculation of beta
      GL_beta_Estimation[1] = -asin(R[3][1]);
      GL_beta_Estimation[2] = pi - GL_beta_Estimation[1];

      //calculation of alfa
      if(cos(GL_beta_Estimation[1]) > 0)
      {
         GL_alfa_Estimation[1] = atan2(R[3][2],R[3][3]);
      }
      else
      {
         GL_alfa_Estimation[1] = atan2(-R[3][2],-R[3][3]);
      }

      if(cos(GL_beta_Estimation[2]) > 0)
      {
         GL_alfa_Estimation[2] = atan2(R[3][2],R[3][3]);
      }
      else
      {
         GL_alfa_Estimation[2] = atan2(-R[3][2],-R[3][3]);
      }

      //calculation of gama
      if(cos(GL_beta_Estimation[1]) > 0)
      {
         GL_gama_Estimation[1] = atan2(R[2][1],R[1][1]);
      }
      else
      {
         GL_gama_Estimation[1] = atan2(-R[2][1],-R[1][1]);
      }

      if(cos(GL_beta_Estimation[2]) > 0)
      {
         GL_gama_Estimation[2] = atan2(R[2][1],R[1][1]);
```

```
        }
        else
        {
           GL_gama_Estimation[2] = atan2(-R[2][1],-R[1][1]);
        }
     }
     //if cos(GL_beta_Estimation) = 0
     else
     {
        GL_gama_Estimation[1] = 0;
        if (R[3][1] == -1)
        {
           GL_beta_Estimation[1] = pi / 2.0;
           GL_alfa_Estimation[1] = atan2(R[1][2],R[1][3]);
        }
        else //R(3,1) == 1
        {
           GL_beta_Estimation[1] = -pi / 2.0;
           GL_alfa_Estimation[1] = atan2(-R[1][2],-R[1][3]);
        }

        GL_alfa_Estimation[2] = GL_alfa_Estimation[1];
        GL_beta_Estimation[2] = GL_beta_Estimation[1];
        GL_gama_Estimation[2] = GL_gama_Estimation[1];
     }


}

void Multiply_3_3_Matrix(float M1[][4], float M2[][4], float M_[][4])
{

   int i, j, k;

   for(i=1; i<4; i++)//next row of M1
   {
     for(j=1; j<4; j++)//next column of M2
     {
        M_[i][j] = 0; //Clear the ex content of matrix elemnt
        for(k=1; k<4; k++)//row_of_M1*column_of_M2
        {
          M_[i][j] += M1[i][k] * M2[k][j];
        }
     }
   }

}

void Copy_3_3_Matrix(float M1[4][4], float M2[4][4])
{
```

```
    int i,j;

    for(i=1;i<4;i++)
    {
       for(j=1;j<4;j++)
       {
          M1[i][j] = M2[i][j];
       }
    }

}

void Print_3_3_Matrix(float M[4][4])
{

    GL_UART_Ek_Veri_Size = sprintf( GL_UART_Ek_Veri, "\n%.2f %.2f
%.2f\r\n",M[1][1],M[1][2],M[1][3]);
    UART_Veri_Depola_Ve_Yolla();
    GL_UART_Ek_Veri_Size = sprintf( GL_UART_Ek_Veri, "%.2f %.2f
%.2f\r\n",M[2][1],M[2][2],M[2][3]);
    UART_Veri_Depola_Ve_Yolla();
    GL_UART_Ek_Veri_Size = sprintf( GL_UART_Ek_Veri, "%.2f %.2f
%.2f\r\n\n",M[3][1],M[3][2],M[3][3]);
    UART_Veri_Depola_Ve_Yolla();

}

void UART_Veri_Depola_Ve_Yolla( void )
{
    unsigned char Sayac = 0;
    //veri depola ve yolla

    //Yeni verilerin sonuna gelene kadar döngü içinde kal
    //yeni verileri UART buffer'ına kalındığı yerden itibaren ekle
    while( GL_UART_Ek_Veri[Sayac] != '\0' )
    {

       GL_UART_Buffer[GL_UART_Buffer_Finish] = GL_UART_Ek_Veri[Sayac];

       Sayac++;

       //GL_UART_Buffer_Finish sayacını artır, taşma olmaması için sona
gelindiğinde başa dön
       GL_UART_Buffer_Finish++;
       if(GL_UART_Buffer_Finish == UART_BUFFER_SIZE)
       {
          GL_UART_Buffer_Finish = 0;
       }

    }
```

```
    _U1TXIE = 1;
    _U1TXIF = 1;
}


//Kesmeler*************************************************************
*************************************************************
void __attribute__((interrupt, no_auto_psv)) _ADC1Interrupt(void)
{

    IFS0bits.AD1IF = 0; //Clear Timer1 interrupt Flag

}

//DMA0_kesmesi----------------------------------------------------DMA0_kesmesi
//ADC1 ile tetiklenir
void __attribute__((__interrupt__)) _DMA0Interrupt(void)
{


    IFS0bits.DMA0IF = 0; //Clear the DMA0 Interrupt Flag

}

//T1Interrupt-----------------------------------------------------T1Interrupt
//1kHz
void __attribute__((interrupt, no_auto_psv)) _T1Interrupt(void)
{
    static unsigned int SC_Sayac_50Hz_UI16 = 0;
    static unsigned int SC_Sayac_1Hz_UI16 = 0;

    PR1 = TIMER1_PERIOD_R;

    GL_Operation_Time = GL_Operation_Time + OPERATION_TIME_STEP;

    SC_Sayac_50Hz_UI16++;
    if(SC_Sayac_50Hz_UI16 == 20)
    {
        GL_Thraed_50Hz = 1;
        SC_Sayac_1Hz_UI16++;

        SC_Sayac_50Hz_UI16 = 0;
    }

    if(SC_Sayac_1Hz_UI16 ==50 )
    {
        GL_Thraed_1Hz = 1;

        SC_Sayac_1Hz_UI16 = 0;
```

```
    }

    IFS0bits.T1IF = 0; //Clear Timer1 interrupt Flag
}

//U1TXInterrupt----------------------------------------------------U1TXInterrupt
void __attribute__((__interrupt__, no_auto_psv)) _U1TXInterrupt(void)
{

    IFS0bits.U1TXIF = 0; // clear TX interrupt flag

    //Tampondan sıradaki veriyi yolla
    if(GL_UART_Buffer_Start != GL_UART_Buffer_Finish)
    {
        U1TXREG = GL_UART_Buffer[GL_UART_Buffer_Start]; //Transmit one
character

        //GL_UART_Buffer_Start sayacını artır, taşma olmaması için sona gelindiğinde
başa dön
        GL_UART_Buffer_Start++;
        if(GL_UART_Buffer_Start == UART_BUFFER_SIZE)
        {
            GL_UART_Buffer_Start = 0;
        }

    }
    else
    {
        _U1TXIE = 0;
    }

}
```

EK C.

```
%% Uçak Parametreleri
% Burada uçağa ait sabitler ve parametreler tanımlanmaktadır
rho = 1; % Hava yoğunluğu

%Kontrol Alanları:
aileron =   0.017;     %0.0245        [m^2]
elevator = 0.027;      %0.0293        [m^2]
rudder = 0.016;    %0.0179        [m^2]

%Pervane Rüzgarının Taradığı Kontrol Alanları:
 aileronP = 0.0101;    %0.0125        [m^2]
 elevatorP =    0.027;     %0.0293        [m^2]
 rudderP = 0.016;      %0.0179        [m^2]

 %Kanat ve Uçak Gövdesi:
 wingArea = 0.16;   %0.08    [m^2]

%  Moment Kolları:
    aileronMomentInfty = 0.40;     %0.235     [m]
    aileronMomentP = 0.20;     %0.13      [m] (to be verified!)
    elevatorMoment = 0.54;   %0.48      [m]
    rudderMoment = 0.62;       %0.55      [m]

%   Kütle ve Atalet:
    m = 0.175;                 %0.27          [kg]
    g = 9.81;
    Ixx = 1.1E-3;                %2.35E-3       [kg m^2]
    Iyy = 0.78E-3;               %1.72E-3       [kg m^2]
    Izz = 1.6E-3;                %3.53E-3       [kg m^2]

% Pervane:
    propDiameter = 0.205;   %0.2286     [m]
    propArea = 0.03;       %0.033  [m^2]
    propWash = 1.15;    %[m/s]

% Lift-Drag Katsayıları:
   CL = (2*3.141);  %Prandt
   Celevator = 0.5;
   Crudder = 0.45;
   Cthrottle =1.67;

% Thrust:
    Tmax  = 4;   %5         [N]
    TmaxSafeHov = 3.0;   %3.2          [N]
    TminSafeHov = 1.0;    %2.0         [N]
    TmaxSafeLev = 2.0;    %2.0         [N]
    TminSafeLev =  0.0;     %0.0           [N]
```

```
% Maksimum Komutlar:
   AilMax = 0.6;       %   [rad]
   AilMin = -0.6;      %   [rad]
   EleMax  = 0.6;      %   [rad]
   EleMin  = -0.6;     %   [rad]
   RudMax = 0.6;       %   [rad]
   RudMin  = -0.6;     %   [rad]
   Umax = 2;
   Umin = 0;

% Tork Hesaplama:  (v_induced^2)*A_p + (v_\infty)^2*A_\infty          %A_p= [0 0
_a1P; 0 _a2P 0; _a3P 0 0]    A_\infty= [0 0 _a1Infty; 0 _a2Infty 0; _a3Infty 0 0]
   a1P = (0.5*rho*CL*rudderP*rudderMoment);
   a2P = (0.5*rho*CL*elevatorP*elevatorMoment);
   a3P = (0.5*rho*CL*aileronP*aileronMomentP);
                                                    %B_p= [0; 0; _b3P]
   a1Infty = (0.5*rho*CL*rudder*rudderMoment);
   a2Infty = (0.5*rho*CL*elevator*elevatorMoment);
   a3Infty = (0.5*rho*CL*aileron*aileronMomentInfty);

   b3P = (0.09/150);              %TO BE ESTIMATED!!!


   %
   hdg = pi;
   s = sin(hdg/2);
   c = cos(hdg/2);
   Kcurrent_quaternion = (sqrt(2)/2)*[ c -s -c  s
                                       s  c  s  c
                                       c -s  c -s
                                      -s -c  s  c];


%% MATLAB SİSOTOOL KULLANILARAK KONTROLÖR
%%TASARLAMAK AMACIYLA ELDE EDİLEN MODELLER

Is = tf(1,[1 0]);    %İntegratör

Kx = Cthrottle/m;
Gx = Kx*Is*Is;   %Dikey Konumda x ekseni hareket modeli
sisotool(Gx)

Kale = CL*0.5*aileronP*aileronMomentP*rho/Ixx*0.5*propWash^2;
Gale = K1*Is*Is;   %Dikey Konumda roll ekseni hareket modeli
step(Gale),grid
sisotool(Gale)

Kele = CL*0.5*elevatorP*elevatorMoment*rho/Iyy*0.5*propWash^2;
```

```
Hele = Celevator*propWash;
Gele = Kele*Is*feedback(Is,Hele,-1);   %Dikey Konumda pitch ekseni hareket
modeli
step(Gele),grid
sisotool(Gele)

Krud = CL*0.5*rudderP*rudderMoment*rho/Iyy*0.5*propWash^2;
Hrud = Crudder*propWash;
Grud = Krud*Is*feedback(Is,Hrud,-1);  %Dikey Konumda yaw ekseni hareket
modeli
step(Grud),grid
sisotool(Grud)
```

EK D.

İndex Sayfası

```
~inc:header.inc~
<div id="content">

<div id="status">

~inc:ucak.inc~
</div>

<h2>GİRİŞ</h2>
<p>Günümüzde insansız hava araçları (İHA) arama ve kurtarma, haritalama, jeolojik
keşif,
askeri keşif akıllı bilgi toplama gibi birçok zorlayıcı görevde kullanılmaktadır. Bu
araçların insansız uçabilmeleri sayesinde tehlikeli görevler yapabilmeleri yanında
üretim
maliyetlerinin de insanlı hava araçlara nazaran düşük olması yaygınlaşmalarını
sağlamaktadır.
Genellikle uzaktan kontrollü ve otonom olmak üzere iki tip İHA üretilmektedir.
Uzaktan kontrollü İHA'lar bir pilot tarafından uzaktan konrol edilirken otonom
İHA'lar ise uçuş öncesi programlandıkları görevi yaparlar.
Otonom İHA'ların yapıları uzaktan kontrollü İHA'lara nazaran daha
karmaşıktır.</p>
<h2>SABİT KANATLI İNSANSIZ HAVA ARACI TASARIMI</h2>
<h3>Gövde Tasarımı</h3>
<p>
Tasarım amacıyla bir çok kapalı ortam uçak modeli incelenmiştir.
Sonuç olarak genel olarak depron köpük ile gerçeklenen hobi uçakların
başlangıç düzeyinde kontrol yapımızı denemeye uygun olacağı görülmüştür.
Gövde tasarımında daha önce uçuş denemeleri yapılmış olan bir model olan
<a
href="http://www.rcgroups.com/forums/attachment.php?attachmentid=3011697">Sb
ach 342</a> kullanılmıştır.
</p>
<table width="600" border="1">
<tr
align="center"><td><b>Sembol</b></td><td><b>Tanım</b></td><td><b>Değer</
b></td><td><b>Birim</b></td></tr>
<tr><td>w<sub>sp</sub></td><td>Kanat
Açıklığı</td><td>90</td><td>cm</td></tr>
<tr><td>L<sub>p</sub></td><td>Uzunluk</td><td>95</td><td>cm</td></tr>
<tr><td>M<sub>p</sub></td><td>Ağırlık</td><td>0.127</td><td>kg</td></tr>
</table>

<h3>Kullanılan Ekipmanlar</h3>
<table width="400" border="1">
```

```
<tr align="center"><td><b>Eleman</b></td><td><b>Ağırlık</b></td></tr>
<tr><td>Depron Köpük</td><td>137g</td></tr>
<tr><td>Servo Motorlar</td><td>25.4g</td></tr>
<tr><td>Fırçasız DC Motor</td><td>66g</td></tr>
<tr><td>Fırçasız DC Motor Sürücü</td><td>87g</td></tr>
<tr><td>Ataletsel Ölçüm Birimi</td><td>24g</td></tr>
<tr><td>Haberleşme Kontrol Kartı</td><td>18g</td></tr>
<tr><td>Kablo ve Bağlantı Elemanları</td><td>50g</td></tr>
<tr><td align="right"><b>TOPLAM </b></td><td>407g</td></tr>
</table>
</div>
~inc:footer.inc~
```

Ayarlama Sayfası

```
~inc:header.inc~
<div id="content">

<div id="status">
~inc:ucak.inc~
</div>
<form method="get" action="kayar.htm">
<div class="examplebox" >
<b>Servo/DC Motor </b><select name="no" style="width:50px;"><option
value="1">1</option><option value="2">2</option><option
value="3">3</option><option value="4">4</option></select><br />
Min. Açı: <input type="text" name="dmin" style="width:50px;" value="-60"/> Min.
 PWM: <input type="text" name="tmin" style="width:50px;"
value="6250"/><br />
Max. Açı: <input type="text" name="dmax" style="width:50px;" value="60"/> Max.
PWM: <input type="text" name="tmax" style="width:50px;" value="12500"/><br />
<input type="submit" value="Kaydet" style="margin-top:10px; width:100px;">
</div>
</form>

<div class="examplebox" >
<b>Fırçasız DC Motor 1:</b><br />
 Min. Açı: ~Servo_Ayar(1,1)~   Min.  PWM: ~Servo_Ayar(1,2)~ <br />
 Max. Açı: ~Servo_Ayar(1,3)~   Max. PWM: ~Servo_Ayar(1,4)~ <br />
Anlık Açı: <input type="text" id="pwm1" onchange="servo_sp(1,this)"
style="width:50px;" value="~Servo_Ayar(1,5)~"/>
Anlık PWM: ~Servo_Ayar(1,6)~ <br />
</div>

<div class="examplebox" >
<b>Servo Motor 2:</b><br />
 Min. Açı: ~Servo_Ayar(2,1)~   Min.  PWM: ~Servo_Ayar(2,2)~ <br />
 Max. Açı: ~Servo_Ayar(2,3)~   Max. PWM: ~Servo_Ayar(2,4)~ <br />
```

```
Anlık Açı: <input type="text" id="pwm2" onchange="servo_sp(2,this)"
style="width:50px;" value="~Servo_Ayar(2,5)~"/>
Anlık PWM: ~Servo_Ayar(2,6)~ <br />
</div>

<div class="examplebox" >
<b>Servo Motor 3:</b><br />
 Min. Açı: ~Servo_Ayar(3,1)~   Min.  PWM: ~Servo_Ayar(3,2)~ <br />
 Max. Açı: ~Servo_Ayar(3,3)~   Max. PWM: ~Servo_Ayar(3,4)~ <br />
Anlık Açı: <input type="text" id="pwm3" onchange="servo_sp(3,this)"
style="width:50px;" value="~Servo_Ayar(3,5)~"/>
Anlık PWM: ~Servo_Ayar(3,6)~ <br />
</div>

<div class="examplebox" >
<b>Servo Motor 4:</b><br />
 Min. Açı: ~Servo_Ayar(4,1)~   Min.  PWM: ~Servo_Ayar(4,2)~ <br />
 Max. Açı: ~Servo_Ayar(4,3)~   Max. PWM: ~Servo_Ayar(4,4)~ <br />
Anlık Açı: <input type="text" id="pwm4" onchange="servo_sp(4,this)"
style="width:50px;" value="~Servo_Ayar(4,5)~"/>
Anlık PWM: ~Servo_Ayar(4,6)~ <br />
</div>

<div class="examplebox" >
<b>Servo Motor 5:</b><br />
 Min. Açı: ~Servo_Ayar(5,1)~   Min.  PWM: ~Servo_Ayar(5,2)~ <br />
 Max. Açı: ~Servo_Ayar(5,3)~   Max. PWM: ~Servo_Ayar(5,4)~ <br />
Anlık Açı: <input type="text" id="pwm5" onchange="servo_sp(5,this)"
style="width:50px;" value="~Servo_Ayar(5,5)~"/>
Anlık PWM: ~Servo_Ayar(5,6)~ <br />
</div>
</div>
~inc:footer.inc~
```

JavaScript Kodu

```
/*********************************************************************
**
 * Microchip TCP/IP Stack Javascript Library

*********************************************************************
**
 *
 * Software License Agreement
 *
 * Copyright  2002-2007 Microchip Technology Inc.  All rights
 * reserved.
 *
```

```
 * Microchip licenses to you the right to use, modify, copy, and
 * distribute:
 * (i)  the Software when embedded on a Microchip microcontroller or
 *      digital signal controller product (Device) which is
 *      integrated into Licensees product; or
 * (ii) ONLY the Software driver source files ENC28J60.c and
 *      ENC28J60.h ported to a non-Microchip device used in
 *      conjunction with a Microchip ethernet controller for the
 *      sole purpose of interfacing with the ethernet controller.
 *
 * You should refer to the license agreement accompanying this
 * Software for additional information regarding your rights and
 * obligations.
 *
 * THE SOFTWARE AND DOCUMENTATION ARE PROVIDED AS IS
WITHOUT
 * WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING
WITHOUT
 * LIMITATION, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR
A
 * PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO
EVENT SHALL
 * MICROCHIP BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR
 * CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF
 * PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES,
ANY CLAIMS
 * BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE
 * THEREOF), ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR
OTHER
 * SIMILAR COSTS, WHETHER ASSERTED ON THE BASIS OF CONTRACT,
TORT
 * (INCLUDING NEGLIGENCE), BREACH OF WARRANTY, OR OTHERWISE.
 *
 *
 * Author              Date       Comment

*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~
 * Elliott Wood        6/25/07    Original
 * Elliott Wood        12/5/07    Updated newAJAXCommand

********************************************************************
/

/**
 * Determines when a request is considered "timed out"
 */
var timeOutMS = 5000; //ms

/**
```

```
 * Stores a queue of AJAX events to process
 */
var ajaxList = new Array();

/**
 * Initiates a new AJAX command
 *
 * @param   the url to access
 * @param   the document ID to fill, or a function to call with response XML
(optional)
 * @param   true to repeat this call indefinitely (optional)
 * @param   a URL encoded string to be submitted as POST data (optional)
 */
function newAJAXCommand(url, container, repeat, data)
{
   // Set up our object
   var newAjax = new Object();
   var theTimer = new Date();
   newAjax.url = url;
   newAjax.container = container;
   newAjax.repeat = repeat;
   newAjax.ajaxReq = null;

   // Create and send the request
   if(window.XMLHttpRequest) {
      newAjax.ajaxReq = new XMLHttpRequest();
      newAjax.ajaxReq.open((data==null)?"GET":"POST", newAjax.url, true);
      newAjax.ajaxReq.send(data);
   // If we're using IE6 style (maybe 5.5 compatible too)
   } else if(window.ActiveXObject) {
      newAjax.ajaxReq = new ActiveXObject("Microsoft.XMLHTTP");
      if(newAjax.ajaxReq) {
         newAjax.ajaxReq.open((data==null)?"GET":"POST", newAjax.url, true);
         newAjax.ajaxReq.send(data);
      }
   }

   newAjax.lastCalled = theTimer.getTime();

   // Store in our array
   ajaxList.push(newAjax);
}

/**
 * Loops over all pending AJAX events to determine
 * if any action is required
 */
function pollAJAX() {

   var curAjax = new Object();
```

```
   var theTimer = new Date();
   var elapsed;

   // Read off the ajaxList objects one by one
   for(i = ajaxList.length; i > 0; i--)
   {
      curAjax = ajaxList.shift();
      if(!curAjax)
         continue;
      elapsed = theTimer.getTime() - curAjax.lastCalled;

      // If we suceeded
      if(curAjax.ajaxReq.readyState == 4 && curAjax.ajaxReq.status == 200) {
         // If it has a container, write the result
         if(typeof(curAjax.container) == 'function'){
            curAjax.container(curAjax.ajaxReq.responseXML.documentElement);
         } else if(typeof(curAjax.container) == 'string') {
            document.getElementById(curAjax.container).innerHTML =
curAjax.ajaxReq.responseText;
         } // (otherwise do nothing for null values)

         curAjax.ajaxReq.abort();
         curAjax.ajaxReq = null;

         // If it's a repeatable request, then do so
         if(curAjax.repeat)
            newAJAXCommand(curAjax.url, curAjax.container, curAjax.repeat);
         continue;
      }

      // If we've waited over 1 second, then we timed out
      if(elapsed > timeOutMS) {
         // Invoke the user function with null input
         if(typeof(curAjax.container) == 'function'){
            curAjax.container(null);
         } else {
            // Alert the user
            alert("Command failed.\nConnection to development board was lost.");
         }

         curAjax.ajaxReq.abort();
         curAjax.ajaxReq = null;

         // If it's a repeatable request, then do so
         if(curAjax.repeat)
            newAJAXCommand(curAjax.url, curAjax.container, curAjax.repeat);
         continue;
      }

      // Otherwise, just keep waiting
```

```
        ajaxList.push(curAjax);
    }

    // Call ourselves again in 10ms
    setTimeout("pollAJAX()",10);

}// End pollAjax

/**
 * Parses the xmlResponse returned by an XMLHTTPRequest object
 *
 * @param   the xmlData returned
 * @param   the field to search for
 */
function getXMLValue(xmlData, field) {
    try {
        if(xmlData.getElementsByTagName(field)[0].firstChild.nodeValue)
            return xmlData.getElementsByTagName(field)[0].firstChild.nodeValue;
        else
            return null;
    } catch(err) { return null; }
}

//kick off the AJAX Updater
setTimeout("pollAJAX()",500);
```

CSS Stil Fonksiyonları

```
body {
    font-family: Verdana, Arial, sans-serif;
    background: #5b5b5b;
    margin: 0px auto;
    padding: 0 0 20px 0;
}

#shadow-one {
    width: 1000px;
    border: 1px solid #555;
    border-top: 0;
    margin: 0px auto;
}
#shadow-two {
    border: 1px solid #4b4b4b;
    border-top: 0;
}
#shadow-three {
    border: 1px solid #444;
```

```
    border-top: 0;
}
#shadow-four {
    border: 1px solid #3a3a3a;
    border-top: 0;
}

#page {
    border: 1px solid #333;
    background: #fff;
    border-top: 0;
    padding: 20px;
    font-size: 10pt;
}

#page td {
    font-size: 10pt;
}

#title {
    background: #009;
    padding: 4px;
    font-weight: bold;
    color: #fff;
    margin-bottom: 15px;
    height: 16px;
}

#title .right {
    float: right;
    width: 800px;
    text-align: right;
}

#menu {
    float: left;
    width: 210px;
    padding-right: 20px;
}

#menu a {
    width: 200px;
    display: block;
    background: #009;
    color: white;
    padding: 8px;
    font-weight: bold;
    border-bottom: 1px solid #fff;
    text-decoration: none;
}
```

```
#menu a:hover {
   background: #ecc42b;
   color: #009;
}

#content {
   width: 700px;
   float: right;
   padding-right: 10px;
}

#content a {
   color: #c00;
   text-decoration: none;
}

#content a:hover {
   color: #d33;
   text-decoration: underline;
}

#content h1 {
   margin-top: 0px;
}

.spacer {
   clear: both;
}

#footer {
   font-size: 0.8em;
   color: #666;
   text-align: center;
   margin: 10px 50px 0 50px;
   padding-top: 10px;
   border-top: 1px dotted #666;
}

#status {
   width: 250px;
   float: right;
   padding: 10px;
   margin: 10px;
   border: 1px dotted #666;
   font-weight: bold;
}

#status p {
   padding: 5px 0 5px 10px;
```

```
    text-indent: -10px;
    margin: 0;
}

#status .leds {
    font-size: 14pt;
    text-align: left;
}

#status .leds a {
    color: #ddd;
    cursor: pointer;
}

#status .leds a:hover {
    color: #ddd;
    text-decoration: none;
}

.examplebox {
    width: 380px;
    margin:0px 0px 0px 0px;
    padding: 8px;
    border: 1px dotted #333;
    text-align:center;
    background:#ddd;
}

.code {
    font-family: Courier New, Courier, fixed-width, fixed;
}

.exercise {
    background: #fe9;
    padding: 5px;
    margin-left: 10px;
    border-left: 2px solid #eb3;
}

.error {
    margin-left: 10px;
    padding: 10px;
    background: #fdd;
    border-left: 2px solid #900;
}

#content h6 {
    border-top: 1px dashed #333;
    margin: 15px;
    padding: 0px;
```

```
    height: 1px;
}

.examplebox td {
    text-align:left;
    padding-right: 15px;
}

fieldset {
    margin:10px 40px 10px 40px;
    padding: 8px;
    border: 1px dotted #333;
    background:#ddd;
}

fieldset div {
    padding: 2px 0px 2px 150px;
}

fieldset div label {
    margin-left: -140px;
    padding-top: 2px;
    width: 135px;
    font-weight: bold;
    position: absolute;
}

fieldset input {
    width: 250px;
}

fieldset input.sm {
    width: auto;
}

fieldset textarea {
    width: 250px;
}

.result
{
    margin: 10px 40px 10px 40px;
    padding: 8px;
    border: 1px solid #333;
    text-align: center;
    font-weight: bold;
    background: #eee;
}

    .ok {
```

```
    border-color: #6b6;
    background: #efe;
 }

 .fail {
    border-color: #b66;
    background: #fee;
 }

 .unk {
    border-color: #cc6;
    background: #ffd;
}
```

ÖZGEÇMİŞ

Ad Soyad: Kadir Akın

Doğum Yeri ve Tarihi: Bursa 14.03.1983

Lisans Üniversite: İstanbul Teknik Üniversitesi Kontrol Mühendisliği Bölümü

Yüksek Lisans Üniversite: İstanbul Teknik Üniversitesi Kontrol ve Otomasyon Mühendisliği Bölümü