T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

İHA'lar için Uçuş Kontrol Kartı Tasarımı ve Kalman Filtre Tabanlı Uçuş Konum Kontrolü

Nur KIKHIA

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Mayısa-2024
KONYA

TEZ KABUL VE ONAYI

Nur KIKHIA tarafından hazırlanan "**İHA'lar için Uçuş Kontrol Kartı Tasarımı ve Kalman Filtre Tabanlı Uçuş Konum Kontrolü**" adlı tez çalışması 17/05/2024 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Dr. Öğr. Üyesi Hüseyin DOĞAN	……..
Danışman Prof. Dr. Ömer AYDOĞDU	……..
Üye Doç. Dr. Akif DURDU	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mevlüt UYAN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Nur KIKHIA

Tarih:17.05.2023

ÖZET

YÜKSEK LİSANS

İHA’lar için Uçuş Kontrol Kartı Tasarımı ve Kalman Filtre Tabanlı Uçuş Konum Kontrolü

Nur KIKHIA

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ömer AYDOĞDU

2024,104 Sayfa

Jüri
Prof. Dr. Ömer AYDOĞDU
Doç. Dr. Akif DURDU
Dr. Öğr. Üyesi Hüseyin DOĞAN

Teknolojinin gelişimiyle birlikte İnsansız Hava Araçlarının (İHA) sivil ve askeri alanlarda yaygın olarak kullanımının arttığı bilinmektedir. Özellikle dikey iniş kalkış yapabilen 4 motorlu İHA'ların yüksek manevra kabiliyeti, küçük boyutu ve otonom uçuş yeteneği gibi önemli özelliklere sahip olması, kullanım alanını genişletmiştir. Bu tez çalışmasında, İHA’lar için özgün bir kontrol kartı tasarımı ve Kalman Filtreleri ile yüksek doğrulukla ölçme ve denge kontrol işlemlerinin gerçekleştirilmesi amaçlanmıştır. Çalışmada ilk olarak genel amaçlı bir uçuş kontrol kartı tasarlanmış ve gerçekleştirilmiştir. Uçuş kontrol kartında ARM Cortex-M4 tabanlı bir mikrodenetleyici olan STM32F401RCT6 mikrodenetleyici kullanılmış ve denge ölçümleri çoklu sensör grubu içinde ataletsel ölçüm birimi (IMU sensörlerle) sağlanmıştır. Çalışmada daha sonra, İHA’lar için oluşturulan deneysel kontrol donanımının ardından, kontrol yazılımında IMU sensörden alınan sinyaller Kalman filtreleri ile anlamlı ve kararlı sinyallere dönüştürülmüş ve kontrol döngüsü için hazır hale getirilmiştir.

Özgün olarak geliştirilen uçuş kontrol kartı, donanım olarak bir İHA için gerekli tüm otonom uçuş kontrol gereksinimlerini sağlayabilecek özelliktedir. Geliştirilen sistemde uçuş kontrol yazılımı iki ana bölümden oluşmaktadır. Birincisi, sensörlerden gelen gürültülü ve gecikmeli verileri filtreleyerek ve tahmin ederek kontrolcüye daha anlamlı ve daha doğru bilgiler sağlayan tahmin edici bölümdür. Burada sensörlerden gelen verileri birleştirerek daha doğru ve güvenilir bir durum tahmini sağlayan Kalman filtreler ve sensör füzyonu algoritmaları birlikte kullanılmıştır. Uçuş kontrol yazılımı ikinci kısımda, İHA'nın açısal hız, açı, doğrusal hız ve konum gibi değişkenlerini kontrol eden denetleyici algoritmaları bulunmaktadır.

Deneysel çalışmada, tasarlanan uçuş kontrol kartının kapalı çevrim denge kontrolü testi için en temel denetleyici yapısı olan PID kontrol kullanılmıştır. Test çalışmalarında sabit bir referans açıya yerleşme, kare dalga formunda düzenlenen referans açılara yerleşme, bozucu etkilere karşı davranışlar gözlenmiştir. Ayrıca Kalman Filtresinin etkileri deneysel

olarak incelenmiştir. Deneysel sonuçlardan, tasarlanan uçuş kontrol kartının denge deneylerinde oldukça başarılı olduğu, başarılı bir uçuş kontrol için filtreleme, özellikle Kalman Filtresi, Sensör Füzyonu gibi tamamlayıcı algoritmaların kararlılığı etkilediği görülmüştür.

Anahtar Kelimeler: Ataletsel Ölçüm Birimi (IMU), Elektronik Hız Denetleyici (ESC), Fırçasız DC Motor (BLDC), İnsansız Hava Aracı (İHA), Kalman Fitre (KF), Oransal-İntegral-Türevsel (PID) Denetleyici.

ABSTRACT

MS THESIS

Flight Control Board Design and Kalman Filter Based Flight Position Control for UAVs

Nur KIKHIA

Konya Technical University
Institute of Graduate Studies
Department of Electrical and Electronical Engineering

Advisor: Prof. Dr. Ömer AYDOĞDU

2024, 104 Pages

Jury
Prof. Dr. Ömer AYDOĞDU
Assoc. Prof. Dr. Akif DURDU
Asst. Prof. Dr. Hüseyin DOĞAN

It is known that the widespread use of unmanned aerial vehicles (UAVs) in civil and military areas has increased with the development of technology. Especially the 4-engine UAVs, which can take off and land vertically, have important features such as high maneuverability, small size and autonomous flight ability, and have expanded their usage area. In this thesis study, it is aimed to design a unique control card for UAVs and to perform high accuracy measurement and balance control operations with Kalman Filters. In the study, firstly, a general purpose flight control card was designed and implemented. STM32F401RCT6 microcontroller, an ARM Cortex-M4 based microcontroller, was used in the flight control card and balance measurements were provided with IMU sensors. Later in the study, after the experimental control hardware created for UAVs, the signals received from the IMU sensor in the control software were converted into meaningful and stable signals with Kalman filters and made ready for the control cycle.

The indigenously developed flight control card is hardware capable of providing all the autonomous flight control requirements required for a UAV. In the developed system, the flight control software consists of two main parts. The first is the predictive section, which provides more meaningful and accurate information to the controller by filtering and predicting noisy and delayed data from sensors. Here, Kalman filters and sensor fusion algorithms are used together, which provide a more accurate and reliable situation estimation by combining data from sensors. In the second part of the flight control software, there are controller algorithms that control the UAV's variables such as angular speed, angle, linear speed and position.

In the experimental study, PID control, the most basic controller structure, was used for the closed-loop stability control test of the designed flight control card. In the test studies, settling to a fixed reference angle, settling to reference angles arranged in a square wave form, and behavior against disturbing effects were observed. Additionally, the effects of the Kalman Filter were examined experimentally. From the experimental

results, it has been seen that the designed flight control card is quite successful in stability experiments, and complementary algorithms such as filtering, especially Kalman Filter, Sensor Fusion, affect the stability for a successful flight control.

Keywords: Inertial Measurement Unit (IMU), Electronic Speed Controller (ESC), Brushless DC Motor (BLDC), Unmanned Aerial Vehicle (UAV), Kalman Filter (KF), Proportional-Integral-Derivative (PID) Controller.

ÖNSÖZ

Tez çalışmalarım süresinde her aşamada çalışmalarımı büyük bir titizlikle yönlendiren ve yardımlarını benden eksik etmeyen danışman hocam Prof. Dr. Ömer AYDOĞDU'ya teşekkürlerimi sunarım. Bilgi ve tecrübelerinden yararlandığım bölümümüz değerli öğretim üyeleri ve elemanlarına ayrıca teşekkür ederim. Çalışmalarım boyunca her türlü destekleriyle yanımda olan eşim Hisham'a, Muhammed kardeşim'e ve güzel aileme teşekkürü borç bilirim.

Nur KIKHIA
KONYA-2024

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

D	Türevsel
F	Frekans
I	İntegral
K_k	Kalman kazancı
P	Tahmin hata kovaryans matrisi
P	Oransal
Q	İşlem hatası kovaryans matrisi
R	Ölçüm hatası kovaryans matrisi
T	Periyot
u_k	Kontrol giriş işareti
v	Ölçüm gürültüsü
w	İşlem gürültüsü
$\hat{X}$	Sistemin durumu tahmin
X_k	k Zamanındaki sistem durum vektörü
$\hat{Y}$	Sistemin gerçek çıkışı için tahmini
y_k	Gürültülü sistem çıkış işareti
y	Ölçülen gürültülü çıkışa
Z_k	k Zamanındaki ölçülen değer vektörü

Kısaltmalar

ADC	Analog-dijital dönüştürücü
ARM	Gelişmiş RISC makineleri
AÖB	Ataletsel ölçüm birimi
ESC	Elektronik hız sürücü
EDA	Elektronik tasarım otomasyonu
HAL	Donanım soyutlama katmanı
KF	Kalman filtresi
MCU	Mikro Kontrol Ünitesi
I2C	Seri bir haberleşme protokolüdür
IHA	İnsansız hava aracı
IWDT	Bağımsız Saat izleyici
IoT	Nesnelerin interneti
LQR	Doğrusal karesel regülatör
LSE	Düşük hızlı osilatör
LSI	düşük hızlı saat
PCB	Baskı devre kartları
PD	Oransal- türevsel denetleyici
PI	Oransal- integral denetleyici
PID	Oransal integral türevsel
PWM	Darbe genişlik modülasyonu
RCC	Sıfırlama ve saat kontrolü
RISC	Azaltılmış komut seti bilgisayarı
RTC	Gerçek Zamanlı Saat
SDIO	Güvenli dijital giriş/çıkış
SPI	Seri çevresel arayüz
SYS	Sistem konfigürasyonu
SCL	Seri veri hattı
SDA	Seri saat hattı
USART	Evrensel senkron/asenkron alıcı/verici
UART	Evrensel asenkron alıcı/verici

1. GİRİŞ

Son yıllarda hava araçları içerisinde kullanımı giderek artan İnsansız Hava Araçları (İHA), günümüzde daha popüler hale gelmiş ve dolayısıyla İHA ile ilgili yapılan çalışmalarda yaygınlaşmıştır. Özellikle yüksek hareket kabiliyeti ve esnekliği nedeniyle sivil uygulamalar için giderek daha fazla İHA istihdam edilmektedir. İHA'lar arama-kurtarma (Alotaibi ve ark., 2019), savunma sanayi (Roberge ve ark., 2018), yangın söndürme (Spurny ve ark., 2021), kargo hizmetleri (Faust ve ark., 2017), uzaktan algılama (Osco ve ark., 2021), haritalama (Rossi ve ark., 2018), hassas tarım uygulamaları (Ye ve ark., 2020), vb. gibi çok geniş bir uygulama alanı yakalamıştır (Liu ve ark., 2020).

Çok rotorlu İHA sistemleri, savunma sanayisi başta olmak üzere birçok farklı kullanım alanına sahiptir. Son yıllarda İHA sistemlerinin kullanımının artmasıyla birlikte, bu sistemler üzerinde yapılan çalışmalar da önemli bir ivme kazanmıştır. Özellikle yerli kaynaklarla geliştirilen İHA sistemlerinin önemi giderek artmaktadır. Yerli kaynaklarla geliştirilen İHA sistemleri, bir ülkenin savunma sanayisindeki bağımsızlığını ve yeteneklerini artırır. Ayrıca, bu tür projelerin tamamlanmasıyla yerli mühendislik ve teknoloji kapasitesinin geliştirilmesine de katkı sağlanmış olur. Bu nedenle, İHA sistemlerinin yerli kaynaklarla geliştirilebilmesi büyük önem taşır (Osman ve ve diğ.,2022).

İnsansız hava araçlar ülkemiz için stratejik öneme sahip hava araçlarıdır. Bu hava araçlarının ana kontrol bileşenlerinin yerli imkânlarla üretilebilir olması stratejik bir önem arz etmektedir. Bu tür araçların araştırma ve geliştirme süreçleri yüksek maliyetlidir. Bu süreçlerde uygun maliyetli ve esnek otopilot bileşenlerinin kullanımı, sektörde geliştirme maliyetlerini azaltacak ve süreçlerin daha hızlı ilerlemesini sağlayacaktır (Selma.,2022).

Kalman filtreleri 1960 yılındaki ilk sunumundan bu yana, geliştirilen binlerce askeri ve sivil yön bulma sistemlerinin ayrılmaz bir parçası olmuştur. Bu görünüşte basit, dijital, algoritmik filtre, sistemin genel performansını elde etmek için navigasyon verileri üzerinde uygun bir kaynaşma sağlaması ile ünlü olmuştur. Sistem durumlarının ölçülen andaki değerlerini tahmin etmek için, filtre bir önceki bilgilere bakarak her bir yeni ölçümü uygun bir şekilde ağırlıklandırıp, istatistiksel bir yöntemle yeni tahminleri gerçekleştirir. Filtre aynı zamanda, gerçek zamanlı kalite değerlendirmeleri için ya da çevrimdışı sistem tasarım çalışmaları için yapılan tahminlerin güncel belirsizliklerini tespit içinde kullanılabilir (Miseon, 2013).

Kalman filtresi, sensör verilerini tahmin etmek ve gürültüyü azaltmak için kullanılan gelişmiş bir yöntemdir. Bu filtre, sensör modelini ve ölçüm modelini kullanarak, geçmiş ölçümlere ve mevcut ölçüme dayalı olarak en olası durumu tahmin eder. İHA sistemlerinde sensör uygulamalarında doğru bir şekilde kullanıldığında, IMU sensörlerinin performansını ve doğruluğunu önemli ölçüde artırabilir.

Literatürde, İHA için açısal konum, irtifa ve yörünge takibi kontrol algoritmaları üzerine çok fazla çalışma yapılmıştır. Çalışmaların büyük bir kısmında lineer kontrol teknikleri kullanılmıştır. Literatür çalışmalarında PID gibi kontrol tekniklerinin bu sistemlere başarıyla uygulandığı görülmüştür. Yapılan çalışmalar genellikle laboratuvar ve simülasyon ortamında gerçekleşmiştir. Saha uçuş testleri için birçok çalışma devam etmektedir.

Oransal-İntegral-Türevsel (PID) kontrol, endüstriyel sistemlerin kontrolünde en yaygın kullanılan kontrol tekniğidir ve endüstride bir standart olarak kabul edilmiştir (Bhatti ve ark., 2015). PID kontrol cihazlarının tercih edilmesinin en önemli sebepleri olarak, çok çeşitli çalışma koşullarındaki gürbüz yapılı kontrol kabiliyetleri ve bir sorun oluştuğunda bunun çözülmesini kolaylaştıracak, fonksiyonel olarak sade yapıda olmaları gösterilebilir (Li ve ark., 2007). PID kontrolü, uçuş kontrol kartlarında da oldukça yaygın olarak kullanılan bir kontrol algoritmasıdır. PID kontrol, istenen bir hedefe ulaşmak için mevcut durumla istenen durum arasındaki farkı değerlendirir ve bu farka göre uygun bir kontrol sinyali üretir. Bu kontrol sinyali genellikle motor hızları veya yönleri gibi çıkışlara uygulanır.

Bu tezde İHA platformlarına yönelik mevcut durumda yurtdışından ithal edilmesi gereken otomatik uçuş kontrol sistemleri yerine, yurtiçinde yenilikçi otomatik uçuş kontrol sistemleri (yerli donanım ve yazılım) ve alt bileşenlerinin geliştirilmesi amaçlanmıştır. Piyasada yerli donanıma ve yazılıma sahip İHA sayısı çok azdır. Bu eksiklikten yola çıkarak yerlileşmeyi desteklemek amacıyla böyle bir tez üzerine çalışma gereksinimi doğmuştur.

Çalışmada tez hedefi bu doğrultuda belirlenmiş ve yerli kaynaklarla geliştirilen bir İHA kontrol kartı ve yazılımı üzerinde çalışarak, yerli sanayinin güçlenmesine katkıda bulunma hedeflenmiştir. Bu tez kapsamında, yerli teknoloji ve mühendislik yeteneklerini kullanarak bir İHA kontrol sistemi geliştirilmesi ve bu sistemin başarılı bir şekilde çalıştırılması amaçlanmaktadır.

1.1. Çalışmanın Amacı

İHA'lar için özgün bir kontrol kartı tasarımı ve gerekli kontrol yazılımlarının geliştirilmesi amaçlanmıştır. İHA kontrolünde, istenilen yörüngede hareket için sürekli İHA konumu ve denge durumunun ölçülmesi ve kullanıcıdan aldığı komut sinyallerini de dikkate alarak uçuş kontrol kartı üzerinden İHA'nın konum ve denge kontrolünün sağlanması gerekir. Ayrıca tasarlanan İHA'lar ile çift taraflı olarak veri iletimi şeklinde haberleşme ve otomatik takip özelliklerinin yapılabilmesi gerekmektedir. Ülkemizde bu tip çalışmaların hızla artmasıyla birlikte yerli ve milli bir uçuş kontrol kartı gereksinimi görülmüştür. Güvenliği sağlamak için yerli bir kontrol kartı tasarlaması ve yurt dışına herhangi bir bağımlılık olmaması amaçlanmıştır.

1.2. Çalışmanın Önemi

Proje kapsamında güncel işlemci ve yazılımlar dikkate alınarak özgün bir uçuş kontrol kartının tasarımı amaçlanmıştır. Projede tasarlanacak ve üretilecek olan kontrol kartında, ARM Cortex-M4 tabanlı bir mikrodenetleyici olan STM32F401RCT6 mikrodenetleyicisi seçilmiştir. Ayrıca IMU ve barometre gibi donanımsal sensörler kullanılmıştır. IMU sensör kartından gelen veriler kalman filtre kullanılarak daha kararlı anlamlı verilere dönüştürülmüş ve bu veriler PID kontrol yardımıyla değerlendirilerek motorlar için gerekli PWM sinyalleri üretilmiştir. Böylece İHA yörünge takibi ve denge kontrolü sağlanmıştır.

Çalışmada bir diğer önemli nokta, bilimsel makaleler (kaynak taraması) göz önünde bulundurularak daha önce yapılmış çalışmalardan farklı, maliyeti düşük ve işlemci hızı daha yüksek bir kontrol kartı gerçekleştirilmiştir.

Teknolojinin gelişmesi ve ilerlemesiyle birlikte insansız hava araçları (İHA) sivil ve askeri birçok alanda kullanılmaya başlanmıştır. Çok rotorlu İHA sistemlerinin başlıca savunma sanayisi olmak üzere birçok kullanım alanları mevcuttur. İHA sistemlerin kullanımının son yıllarda artmış olması, bu sistemler üzerine yapılan çalışmaların yoğunlaşmasına neden olmuştur. Bu doğrultuda insansız hava aracı sistemlerinin yerli kaynaklarla geliştirilebilmesi büyük önem arz etmektedir. Bu ihtiyaç göz önünde bulundurularak projenin sonuçlanması hedeflenmiştir.

2. KAYNAK ARAŞTIRMASI

Sinan E., (2007) İstanbul Teknik Üniversitesi'nde gerçekleştirilen "Bir İnsansız Hava Aracı ve Uçuş Kontrol Sistem Tasarımı" tezinde, dikey iniş kalkış özelliğine sahip bir İHA kavramsal tasarımı ve bu tasarıma göre şekillenen kontrol tahrik sistemi (KTS) uygulaması anlatılmıştır. KTS aynı zamanda İHA'nın itki sistemidir ve aynı eksen üzerinde birbirine zıt yönde dönen iki pervaneden oluşan bir fan şeklindedir. İtki sisteminin kontrolü İHA'nın yönelimini ve havada dengede kalmasını sağlamakta bu yüzden tasarımdaki ilk ve en büyük problem olarak görülmüştür (Sinan Ekinci, 2007).

KTO Karatay Üniversitesi'nde gerçekleştirilen "Belli Bir Rota Üzerinde Helikopterin Otonom Uçuşu için Yaklaşım Geliştirilmesi" tezinde, belli bir rota üzerinde otonom ve insansız helikopter uçuş yöntemi önerilmiş, gerekli olan tüm donanım ve yazılım bileşenlerinin uyum ve bütünleşme süreçleri detaylı bir biçimde anlatılmıştır. Helikopter, çok karmaşık ve kontrolü zor olan bir dinamik yapıya ve kararsız uçuşa sahip olduğundan insansız hava aracı olarak tercih edilmemektedir. Bu çalışmada helikopter yönteminin seçilmesinin sebebi; helikopterin diğer hava araçlarına göre daha atik ve manevra kabiliyetinin daha yüksek olmasıdır. Sabit kanatlı hava araçlarında bulunmayan havada asılı kalma ve dikey iniş-kalkış yapma, çok rotorlu araçlarda olmayan hız ve atiklik avantajlarını helikopter mümkün kılmaktadır. Buna karşın diğer hava araçlarına göre daha karmaşık bir uçuş dinamiğine sahip olan helikopterin otonom uçuşu için gerekli yöntemin uygulanması da bir o kadar özen ve dikkat gerektirmektedir. Bu tez çalışması ile insansız ve otonom helikopter teknolojisine farklı bir bakış açısı getirilmesi düşünülmüştür (Ayhan, 2019).

2021'de yayınlanan "Bir İnsansız Hava Aracı Sisteminin Tasarımı, Benzetimi ve Gerçekleştirilmesi "bir çalışmada, İHA olarak bir dört kanatlı araç gerçekleştirilmiştir. Bu dörtuçarın teorik çalışmaları ve performans özelliklerini çıkarmak için "eCalc" programı kullanılmıştır. eCalc'te tasarım süreci ve benzetim işlemleri etkileşimli olup, mümkün olduğunca gerçek elemanların özelliklerinin kullanıldığı gerçeğe çok yakın ortamlar oluşturulabilmektedir. eCalc ile elde edilen sonuçlar MATLAB uygulamalarından elde edilen sonuçlarla karşılaştırılarak, bir İHA genel sistemi ve alt sistemleri için gerekli olan donanım ve yazılım mimarileri tasarlanarak uygulamada kullanılacak dörtuçar gerçekleştirilmiştir. Yer kontrol istasyonu, olarak Mission Planner kullanılmıştır (Elmas ve Alkan, 2021).

2021'de yayınlanan "Üç Döner kanatlı ve Döner-Rotorlu İnsansız Hava Aracının Tasarımı" bir çalışmada, tüm döner kanat rotorları eğimlenebilen (tilt mekanizmasına sahip), üç döner kanatlı, dikey kalkış iniş yapabilen insansız hava aracının tasarım süreci sunulmuştur. Modelleme alt yapısının kurulması, malzeme ve bileşen seçimi, enerji tüketimi, titreşim analizi, statik yük analizi ve fiziksel sistem testleri gibi çalışmalara değinilmiştir. Sürecin sonunda ilk prototip ortaya çıkarılmış ve uçuş testlerine başlanmıştır (Kaçar ve ark., 2021).

2021'de yayınlanan "İnsansız Bir Hava Aracı Modelinin Üç Boyutlu Tasarımı, Analizi ve Simülasyonu" bir çalışmada, üç boyutlu tasarlanan bir taktik İHA modelinin akış analizleri yapılarak benzetim sonuçları verilmiş. Tasarlanmış olan 3B modelin yüksek basınçlara maruz kaldığı, malzeme seçiminde bu hususun göz önünde bulundurulması gerektiği, bununla birlikte uçağın yapısının aerodinamik açıdan başarılı olduğu tespit edilmiştir. Ayrıca yüksek statik basınç noktaları için önlemler alındığı takdirde tasarlanan insansız hava aracımız güvenli ve uygun aerodinamik yapıya sahip olduğunu saptanmıştır (Çuhadar ve ark., 2021).

2014'te yayınlanan "Taktik İnsansız Hava Aracı Tasarımı ve Üretimi" çalışmasında, Hava aracının bütün mekanik ve elektromekanik sistem entegrasyonu başarı ile tamamlanmıştır. Hava aracına takılı bir kamera ve verici sistemi vasıtası ile gerçek zamanlı görüntü aktarımı yer testleri sırasında sağlanmıştır. Yer testleri başarı ile gerçekleştirilmiş ve hava aracı tasarımında iyileştirmesi gereken noktalar saptanmıştır. Bir sonraki adımda belirlenen tasarım iyileştirmeleri yapıldıktan sonra uçuş denemelerine geçilecektir (Kayran ve Seber., 2014).

2016'da Karadeniz Teknik Üniversitesi'nde gerçekleştirilen" Quadcopter Uçuş Kontrolcüsünün Tasarlanması" Tasarım Projesinde, genel olarak Quadcopterin dengeli bir biçimde hareket etmesi, Android uygulamasından ve Jiroskoptan gelecek verilerin koordineli bir biçimde Raspberry Pi'ye aktarılarak motor sürücüye bu verilere göre gerekli frekansları göndermekle alakalıdır. Proje geliştirilirken jiroskoptan gelen verilerin filtrelenmesi ve seçilen motorlar fazla akıma gerek duyduğundan güç sorunu yaşanmış. Sorunun çözülmesi için iki motor bir güç kaynağı ile diğer iki motorda başka bir güç kaynağı ile beslenmiş. Proje geliştirilirken Raspberry Pi kullanımı ve genel ayarlamaların yapılması, Raspberry Pi 'de GPIO arayüzü kullanımı, Linux ortamında yazılım geliştirme Android Programlama, Client-Server Modeli, I2C Protokolü gibi konular incelenmiştir (Topsakal ve ark., 2016).

2021'de yayınlanan "Döner Kanatlı Özgün Bir İHA Tasarımı ve Uçuş Kontrol Kartının Yerli Yazılım ile Optimizasyonu" bir çalışmada, 4 rotorlu döner kanatlı bir İnsansız Hava Aracı (İHA) tasarlanmıştır. Gövdesi karbon fiberden kare şeklinde iki plakadan, kolları ise alüminyum kare profilden yapılmıştır. X-tipi bir İHA olan tasarımın ağırlığı toplamda görev mekanizması hariç 1300 gr, görev mekanizması dâhil 1700 gr ve 300 gr su ile görev sırasında maksimum 2000 gr olacak şekilde optimize edilmiştir. Yaklaşık boyut motordan motora 50cm olup tasarıma ait uçuş kontrol kartı, özgün yazılımı ve yer kontrol yazılımı takım tarafından gerçekleştirilmiş, acil durumda elektrik aksamının kapatılması için önlemler alınmıştır (Gönger ve ark., 2021).

2016'de yayınlanan Karabük Üniversitesi'nde gerçekleştirilen "Quadcopter (Drone) Tasarım" tezinde, dört pervaneli bir quadcopterin tasarım, üretim ve programlamasına ait tüm detaylar verilerek, kişisel olarak bir quadcopterin nasıl tasarlanıp geliştirileceği anlatılmıştır. Çalışmada geliştirilen quadcopter havada uçarken üzerinde bağlı bulunan mini kameradan aldığı görüntüyü yedi ekrana anlık olarak aktarım yapmaktadır. Üzerindeki GPS sisteminden okumuş olduğu konum bilgilerini, sensörlerden almış olduğu yükseklik ve basınç bilgilerini de göndererek konum takibi ve uçuş güvenlik imkanını da sağlamaktadır (Eryıldız, 2016).

2016'da yayınlanan Karabük Üniversitesi'nde gerçekleştirilen "Android İşletim Sistemli Cihazlar ve 2.4ghz Kumanda ile Arm Tabanlı Mikrodenetleyici Üzerinden dört rotor İnsansız Hava Aracının Tasarımı, Otonom Kontrolü ve Denetimi" tezinde, Arm mimarisine sahip bir mikrodenetleyici kullanılarak, Android işletim sistemli cihazlar ve 4 kanallı RC kumanda ile dört rotor aracın mekanik, elektronik tasarımı, otonom kontrol ve denetimi yapılmıştır. Kumandadan veya android işletim sistemli cihazlardan alınan kullanıcı isteklerinin mikrodenetleyiciye aktarılması, jiroskop, ivmeölçer, pusula ve barometrik basınç sensörlerinden alınan bilgiler mikrodenetleyicide işlenerek elektronik hız kontrolcü sistemlerine aktarılarak dört rotor r araca istenilen hareketler verilmektedir. Android işletim sistemli cihazlardan quadcopterin kalibrasyon ayarları yapılabilmektedir (Özer, 2016).

2016'de yayınlanan Karabük Üniversitesi'nde gerçekleştirilen" Öz Ayarlamalı Bulanık PID Denetimli Dört Rotorlu İnsansız Hava Aracıyla Otonom Güzergâh ve Nesne Takibi" tezinde, dört rotorlu insansız hava aracı (DRİHA) ile bir referans güzergahın ve bir nesnenin otonom takibi öz ayarlamalı bulanık PID denetleyici ile gerçek zamanlı olarak yapılmıştı. Çalışmada, üzerinde kablosuz haberleşme, kamera ve uçuş için gerekli olan algılayıcı donanımlarına sahip AR.Drone dört rotorlu hava aracı kullanılmıştı. Bu

DRİHA ile bilgisayar arasında kablosuz bağlantı kurulabilmektedir. Bu bağlantı sayesinde istemci konumundaki bilgisayar ile sunucu konumundaki hava aracı arasında çift taraflı veri iletimi yapılabilmektedir. Güzergâh ve nesne takibi için geliştirilen MATLAB/Simulink modellerine aktarılan veriler ile dört rotorlu hava aracının denetimi sağlanmıştı. DRİHA denetiminde PID denetleyici ve PID denetleyicinin kazançlarını ayarlamada bir bulanık mantık denetleyicisi kullanılmıştı. Önerilen kontrol yöntemi ile klasik PID denetim yönteminin karşılaştırıldığı kapalı ortamda yapılan güzergâh takibi uygulamaları neticesinde, önerilen yöntem daha az hatayla referans güzergahı takip etmişti (Demir, 2016).

2020'de yayınlanan "Uyarlamalı Bulanık Mantık Denetleyici Tabanlı İnsansız Hava Aracı (İHA)'nın Rota Takibi ve Faydalı Yük Taşıma Performansı" adlı makalesinde, yük alma bırakma sistemi kullanılarak belirlenen bölgedeki farklı renkli iki özdeş faydalı yükün dört pervaneli İHA tarafından sırasıyla alınarak belirlenen konuma bir rota üzerinden minimum hata ile bırakılabilmesi için bulanık PID denetleyicisi tasarlanmıştı. Faydalı yük alma, taşıma ve bırakma görevi esnasında elde edilen sonuçlar PID denetleyici performansı ile kıyaslamalı olarak analiz edilmiştir (Belge ve ark., 2021).

2021' de yayınlanan"Döner Kanatlı İnsansız Hava Aracının Sistem Tasarımı ve Kontrolü" Araştırma Makalesinde, dört rotorlu insansız hava aracı ile aynı faydalı yük kapasitesi ve aynı uçuş süresine sahip şekil değiştirebilen dört rotorlunun sistem tasarımı ve kontrolü gerçekleştirilmiş. Kararsız yapıda olan dört rotorlu PID (oransal integral türev) kontrolör ile kontrol edilebilmiş. Hava aracı üzerinde bulundurduğu MEMS (Mikro Elektro-Mekanik Sensör)'ler ve engel algılama sensörleri sayesinde dört rotorlunun kolları arasındaki kesişim açısını değiştirecek aktüatörün enerjilendirilmesi ile açı azaltılarak kapalı ortamda engellerden sakınması ve seyrine devam edebilmesi amaçlanmış. Hava aracının seyir halinde şekil değiştirmesi neticesinde meydana gelen konfigürasyon değişikliğinin uçuş karakteristiğine olan etkileri incelenmiş (Oktay ve Özen, 2021).

2021'de yayınlanan "Mikrodenetleyicili İHA Uçuş Test Düzeneği Tasarımı" Araştırma Makalesinde, döner kanatlı insansız hava araçları için üç eksen kontrollü (x, y ve z eksenlerindeki denetim) uçuş test düzeneği hazırlanmıştır. Uçuş denetim parametreleri test düzeneği üzerinde bulunan mikroişlemci kontrollü devre ve test sistemi için tasarlanan mobil uygulama sayesinde izlenebilmektedir. Eksen bilgisi "IMU" sensörleri sayesinde elde edilirken uçuş test düzeneğinin akıllı telefon ile olan iletişimi "bluetooth" teknolojisi ile sağlanmıştır. Bu düzenek özellikle proje geliştirme

aşamasındaki döner kanatlı insansız hava araçların kontrolü için tasarlanmıştır. Bu uçuş test sistemi ile insansız hava aracı, uçuş esnasındaki temel senaryoları gerçekleştirilebilmektedir. Kalibrasyon hataları, itki sistemindeki dengesizlik, uçuş esnasındaki denge ve kararlılık, ağırlık merkezinin sapması gibi olası durumların uçuş öncesinde gözlemlenmesine imkân sunmaktadır (Küçüksezer ve Sancaktar, 2021).

3.MATERYAL VE YÖNTEM

3.1. Giriş

Hava araçlarının gövde yapıları tasarlanırken, bir dizi önemli faktör dikkate alınır. Bu faktörler arasında hafiflik, darbe dayanımı, termal kararlılık, korozyona dayanıklılık ve maliyet gibi hususlar bulunmaktadır. Dolayısıyla, havacılık endüstrisinde genel olarak, bu gereksinimleri karşılamak için çeşitli malzemeler tercih edilmektedir.

Karbon fiber, seramik elyaf, fiberglas gibi kompozit malzemeler, havacılık endüstrisinde yaygın olarak kullanılan malzemeler arasındadır. Bu malzemeler, yüksek mukavemetleri ve düşük ağırlıkları sayesinde hafiflik ve dayanıklılık sağlamaktadır. Aynı zamanda, epoksi, polyester, fenolik gibi reçineler ve çeşitli termoplastik malzemeler de gövde yapılarının üretiminde sıkça kullanılan malzemeler arasındadır. Özellikle İHA gövdelerinde, hafiflik ve darbe dayanımı önemli bir rol oynar. Bu nedenle, karbon fiber kompozit malzemesi İHA gövdelerinde sıklıkla tercih edilir. Karbon fiber, yüksek mukavemeti ve düşük ağırlığıyla İHA'ların performansını artırırken, aynı zamanda enerji verimliliğini artırır. Bu avantajlar, İHA'ların uzun süreli uçuşlarında ve operasyonel verimliliğinde önemli bir rol oynar (Gminside, 2021).

İnsansız Hava Araçları, genellikle görevleri yönetmek, uçuşu kontrol etmek, verileri işlemek ve görev gereksinimlerini yerine getirmek için entegre bir bilgisayar sistemi veya hafif, enerji verimli ve yüksek performanslı işlemci kullanmaktadır. İşlemci türü İHA'nın görevine ve karmaşıklığına bağlı olarak değişmektedir. İHA'ların kontrolü ve operasyonu için çeşitli işlemci tipleri kullanılabilmektedir, ancak en yaygın kullanılanalar; ARM İşlemciler, PIC Mikrodenetleyiciler, DSP (Dijital Sinyal İşleme) işlemcileri ve FPGA (Alan Programlanabilir Kapı Dizisi) İşlemcileridir.

Yapılan çalışmada, aşağıda bahsedilen avantajlarından dolayı ARM işlemcileri kullanılmıştır. ARM mimarisi günümüzde oldukça yaygın bir mikroişlemci mimarisidir ve çeşitli elektronik cihazlarda geniş bir kullanım alanına sahiptir. Cep telefonları, tabletler, akıllı saatler, dijital kameralar, IoT (Nesnelerin İnterneti) cihazları ve birçok

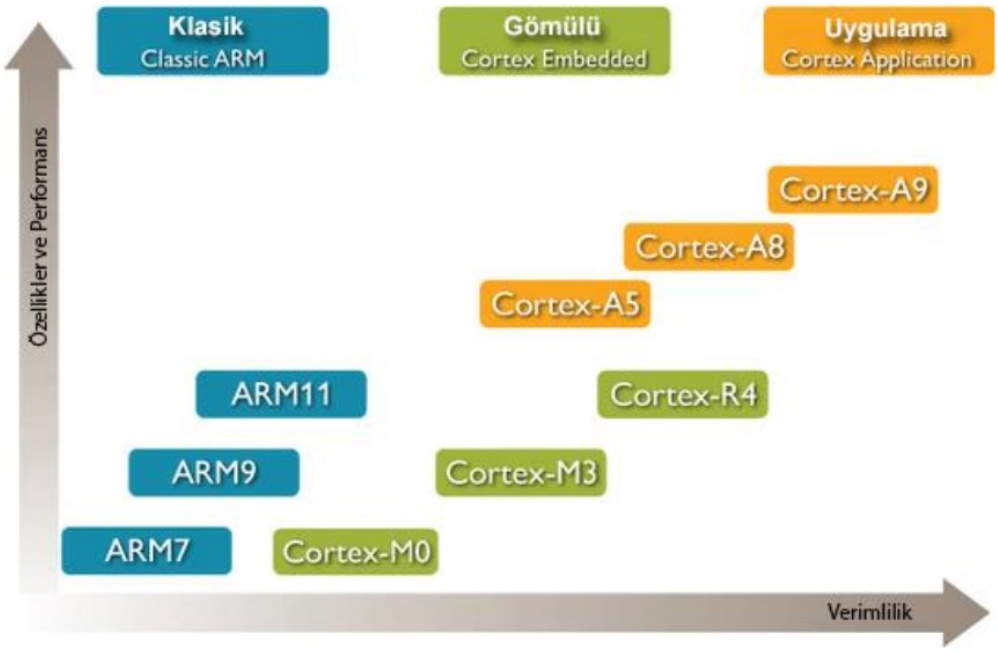
tüketici elektroniği ürününde sıkça bulunmaktadır. ARM mimarisi, düşük güç tüketimi, yüksek performans, küçük boyut ve çeşitli uygulama alanları için esneklik gibi avantajlarıyla bilinir.

ARM mimarisi, lisanslama modeliyle işler; yani ARM Holdings tarafından geliştirilir ve diğer şirketlere lisanslanır. Bu da elektronik cihaz üreticilerinin kendi ihtiyaçlarına uygun özelleştirilmiş ARM tabanlı işlemciler geliştirmelerine olanak tanır.

Makine öğrenme algoritmalarıyla uyumlu ve yüksek performanslı çalışması için tasarlanan ARM işlemciler, yüz tanıma sistemleri, nesne takibi uygulamaları, sanal gerçeklik gibi çalışmalarda da tercih edilirler.

Şekil 3.1'de gelişim evresi verilen ARM-Cortex serisi, üst düzey güvenlik gerektiren uygulamalar için özel olarak tasarlanmış bir işlemci serisidir. Savunma sektörü, ödeme sistemleri, SIM kart uygulamaları gibi sistemler için tercih edilir. Bu nedenle, savunma sektöründe kullanılan İHA projeleri için ARM mimarisi tercih edilmektedir.

Sonuç olarak, literatür taramasından ve yapılan araştırmalardan İHA'lar için kullanılabilecek en uygun işlemci mimarisinin ARM mimarisi olduğunu görülmektedir. ARM mimarisi, üst düzey güvenlik, makine öğrenme ve yüksek performans gibi özellikleriyle öne çıkar ve bu nedenle tercih edilmiştir.

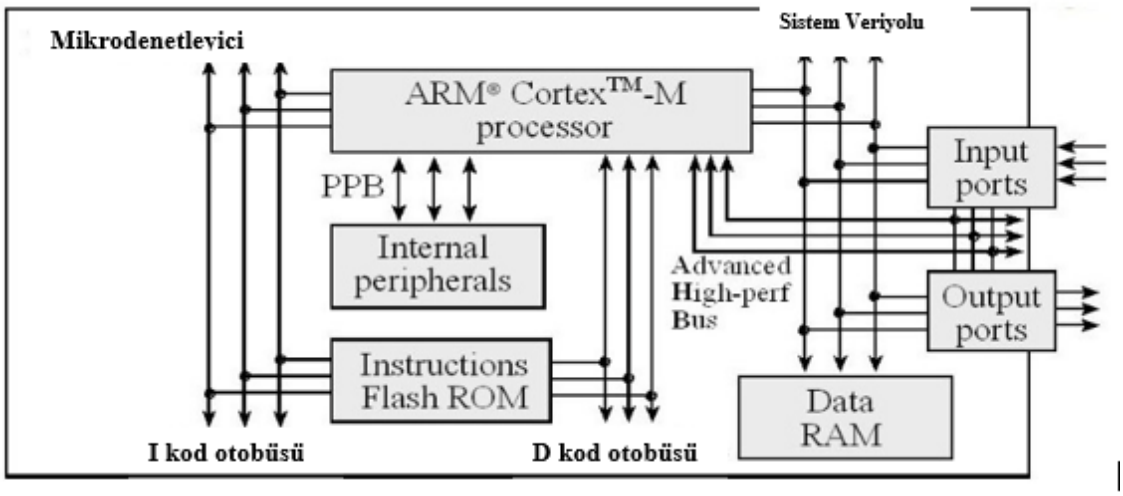


Şekil 3.1. ARM Mimarisi

Cortex-M3, ARM tarafından geliştirilen bir mikrodenetleyici çekirdeğidir ve gerçek zamanlı ve düşük güç tüketimi gerektiren uygulamalarda kullanılmak üzere tasarlanmıştır. Bu çekirdek, genellikle gömülü sistemlerde, endüstriyel kontrol sistemlerinde, otomotiv uygulamalarında, tıbbi cihazlarda ve diğer benzer uygulamalarda tercih edilir.

Cortex-M3 çekirdeği, ARM'nin Thumb-2 komut seti mimarisini kullanır ve 32-bitlik bir yapıya sahiptir. Bu çekirdek, düşük güç tüketimi ile yüksek performans sağlar ve aynı zamanda deterministik bir şekilde çalışarak gerçek zamanlı sistemler için uygun bir seçenektir.

Cortex-M3 çekirdeği, düşük güç tüketimi, hızlı işlemci hızları, gelişmiş kesme denetimi ve geniş bir periferik seti gibi özellikler sunarak, çeşitli gömülü sistem uygulamaları için ideal bir seçenek sunar. Bu nedenle, bu çekirdek, mikrodenetleyici pazarında yaygın olarak kullanılan bir çözümdür. Arm Cortex-M İletişim Kanalları aşağıda Şekil 3.2'de gösterilmiştir.



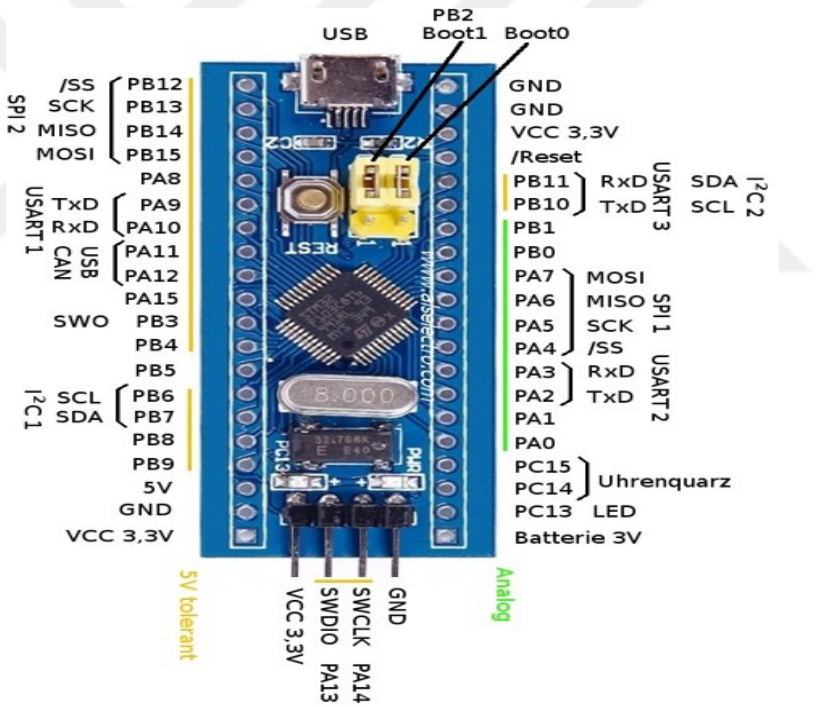
Şekil 3.2. Arm Cortex-M iletişim kanalları

3.2. STM32F103C8T6 Mikrodenetleyici

Arduino, elektronik projeler geliştirmek isteyen birçok kişinin tercih ettiği bir geliştirme kartıdır. Ancak, proje geliştikçe ve yeni bilgiler edinildikçe, Arduino'nun bazı sınırlılıkları fark edilebilir. Bu yüzden, STM32F103 gibi mikrodenetleyicilerin kullanılması önerilmektedir. Çünkü bu tür mikrodenetleyiciler, Arduino'ya göre daha

fazla pin çıkışına sahip olmalarının yanı sıra entegre yapısı sayesinde birden fazla görevi hızlı bir şekilde yerine getirebilme yeteneğine sahiptir.

STM mikroelektronik firması tarafından geliştirilen 32bit mikrodenetleyici ailesinde ARM mimarisi kullanılmaktadır. Bu aile, F0, F1, F2, F3, F4, F7 gibi serilere ayrılmıştır. Bu uygulamada kullanılan kart, STM32F103C8T6 mikrodenetleyicili "blue pill" adını taşır ve F1 serisine aittir. Kart, Arduino Nano'ya güçlü bir alternatif sunar; aynı yapıya sahip olması ve diğer serilere kıyasla daha uygun fiyatlı ve erişilebilir olması öne çıkmaktadır. Mikrodenetleyicide 32-bit ARM Cortex-M3 çekirdeği bulunur ve RISC mimarisiyle çalışır. Bu kart, gömülü sistemlerde kendini geliştirmek isteyenler için güçlü ve ekonomik bir seçenektir. 72 MHz işlemci frekansıyla yüksek performans sağlarken, I2C, USART-UART ve SPI gibi iletişim protokollerini destekler. Kullanılan mikrodenetleyici görseli Şekil 3.3'te yer almaktadır.

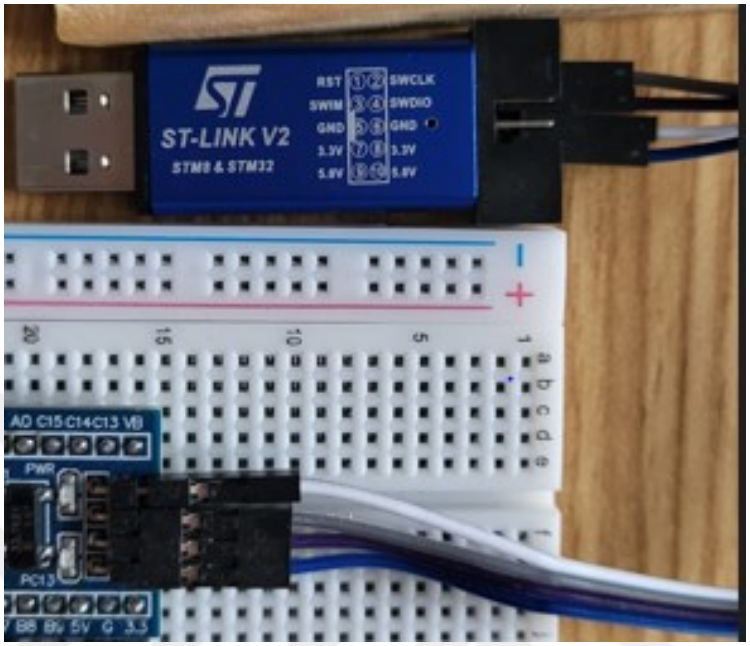


Şekil 3.3. STM32F103C8T6 Mikrodenetleyicisi

3.2.1. ST-link V2

ST-link V2, düşük maliyetli bir çözüm olarak STM8 ve STM32 mikrodenetleyici aileleri için bir devre hata ayıklayıcı ve programlayıcı olarak hizmet vermektedir. STM32F103C8T6 kartının alt kısmındaki dörtlü çıkışları ST-LINK karşılıklarına bağlanarak kablolama işlemi tamamlanır. Ardından, yazılım uygulamalarının hex uzantılı dosyaları program yardımıyla yüklenir ve bu dosyaların yerleri not edilir. Program

yükleme işlemi için ST-LINK Utility yazılımı kullanılır. ST-link V2'nin görüntüsü Şekil 3.4'te verilmiştir.



Şekil 3.4. ST-Link V2

3.3. Programlama ve Tasarım Araçları

3.3.1.STM32cubeMX

STM32CubeMX, STMikroelektronik tarafından sunulan bir grafiksel konfigürasyon aracıdır ve STM32 serisi mikrodenetleyicilerle çalışan projelerin başlatılması ve yapılandırılması için kullanılmaktadır. Bu araç, mikrodenetleyici tabanlı projelerin hızlı bir şekilde başlatılmasını sağlar ve karmaşık konfigürasyon ayarlarını kullanıcı dostu bir arayüz üzerinden kolayca yapılmasını sağlar. STM32CubeMX STM32 mikroişlemcisi çok kolay bir konfigürasyon, aynı zamanda ARM için karşılık gelen başlatma C kodu üretilmesini sağlayan bir grafik aracıdır. Kullanıcıların genel amaçlı giriş/çıkış pinlerini, Zamanlayıcı ayarlarını, güç tüketimini hesaplamalarını, MCU da bulunan gerekli çevre birimlerinin yapılandırılmasını, UART, SPI gibi gerekli çevre birimlerini yapılandırmalarını ve USB, TCP/IP gibi ara katman yığınlarını hazırlamalarını sağlamaktadır. Bu süreçler, daha az emek harcanarak ve kısa bir sürede projelerin tasarlanmasını sağlamaktadır.

Burada proje oluşturma ve tamamlama aşamaları beş adıma ayırılabilmektedir:

1. MCU mikrodenetleyici modelinin seçimi,
2. İhtiyaç duyulan çevre birimlerinin etkinleştirilmesi,
3. Saat sinyalinin konfigürasyonu,
4. Ara katman yığınlarının konfigürasyonu
5. Proje için uygun derleyici seçiminin yapılması ve projenin oluşturulması.

Bu aşamaları takip ederek, kullanıcılar STM32CubeMX aracılığıyla projeleri hızlı ve verimli bir şekilde oluşturabilmektedir.

3.3.2.Keil

Keil programı, ARM işlemcileri için program yazma, derleme, derlenmiş kodların çipe aktarılması, yazılımın çalıştırılması ve kodlardaki hataların ayıklanması gibi işlemleri gerçekleştirmek için kullanılmaktadır. Ayrıca, IAR ve Atollic TrueStudio gibi diğer yazılım araçları da bu tür işlemleri gerçekleştirmek için kullanılabilir.

Vision IDE, proje yönetimini, çalışma zamanı ortamını, derleme olanaklarını, kaynak kodu düzenlemesini ve program hata ayıklamasını tek bir güçlü ortamda birleştirir. µVision'ın kullanımı kolaydır ve gömülü yazılım geliştirmesini hızlandırır. µVision birden fazla ekranı destekler ve görsel yüzeyin herhangi bir yerinde ayrı pencere düzenleri oluşturmasına olanak tanımaktadır.

Vision Debugger, testlerin doğrulanmasını ve uygulama kodunun optimize edilmesini sağlayabilen tek bir ortam sunmaktadır. Bu hata ayıklayıcı, basit ve karmaşık kesme noktaları, izleme pencereleri ve yürütme kontrolü gibi geleneksel özellikleri içerirken, aynı zamanda aygıt çevre birimlerine tam görünürlük sağlar. Keil gibi kullanışlı bir derleyiciye sahip olmasına rağmen, kullanıcıya ücretsiz sınırlı kullanım sunmaktadır.

3.3.3.Stm32studio

STM mikroelektronik tarafından geliştirilen güçlü bir araçtır, STM32 mikrodenetleyiciler (MCU'lar) ile çalışan uygulamalarda hata ayıklama ve tanılama işlemlerini kolaylaştırır. Gerçek zamanlı olarak değişkenleri okuma ve görüntülemesini sağlamaktadır. STM Studio, karmaşık hata ayıklama işlemlerini basitleştirir. Değişkenleri tek tek incelemek yerine, bu araç tüm değişkenleri tek bir pencerede gerçek zamanlı olarak görüntülemesine olanak tanır. Bu sayede, programın hangi bölümünde sorun olduğunu hızlı ve kolay bir şekilde belirleyebilmektedir.

STM Studio, kullanımı kolay bir arayüze sahiptir. Sezgisel tasarımı sayesinde, bu aracı yeni başlayanlar da kolayca kullanabilir.Bir bilgisayarda çalışan STM Studio, STM32 MCU'larla standart ST-LINK geliştirme araçlarıyla arayüz oluşturur. STM Studio, uygulamaların gerçek zamanlı davranışını koruyan müdahaleci olmayan bir araçtır.

STM Studio, uygulamalara ince ayar yapmak için geleneksel hata ayıklama araçlarını mükemmel bir şekilde tamamlar. Motor kontrol uygulamaları gibi durdurulamayan hata ayıklama uygulamaları için çok uygundur. Hata ayıklama ve tanılama gereksinimlerini karşılamak veya uygulama davranışını göstermek için farklı grafik görünümler mevcuttur.

3.3.4.KiCad

KiCad, açık kaynaklı bir elektronik tasarım otomasyon (EDA) yazılımıdır. Elektronik devrelerin şematik çizimlerini oluşturmak, devre kartları tasarlamak ve bunları üretim için hazırlamak için kullanılmaktadır. KiCad, şematik çizimleri, devre kartı tasarımı ve PCB (baskılı devre kartı) üretim süreçlerini tek bir entegre ortamda birleştirir.

KiCad'in özellikleri beş ana modülde toplanmıştır:

- kicad: Proje yönetimi işlevini yerine getirir.
- eeschema: Elektronik devre şemalarının yakalanmasını ve oluşturulmasını sağlayan şematik editörüdür.
- cvpcb: Bileşenlerin ayak izi seçimini yapar ve devre tasarımında kullanılan bileşenlerin ayak izlerini yönetir.
- pcbnew: PCB çizimi ve düzenlemesi için kullanılır; ayrıca 3D görüntüleme özelliğine sahiptir.
- gerbview: Gerber dosyalarını görüntüleyen bir araçtır; PCB üretimi için gereken çıktı dosyalarını incelemek için kullanılır.

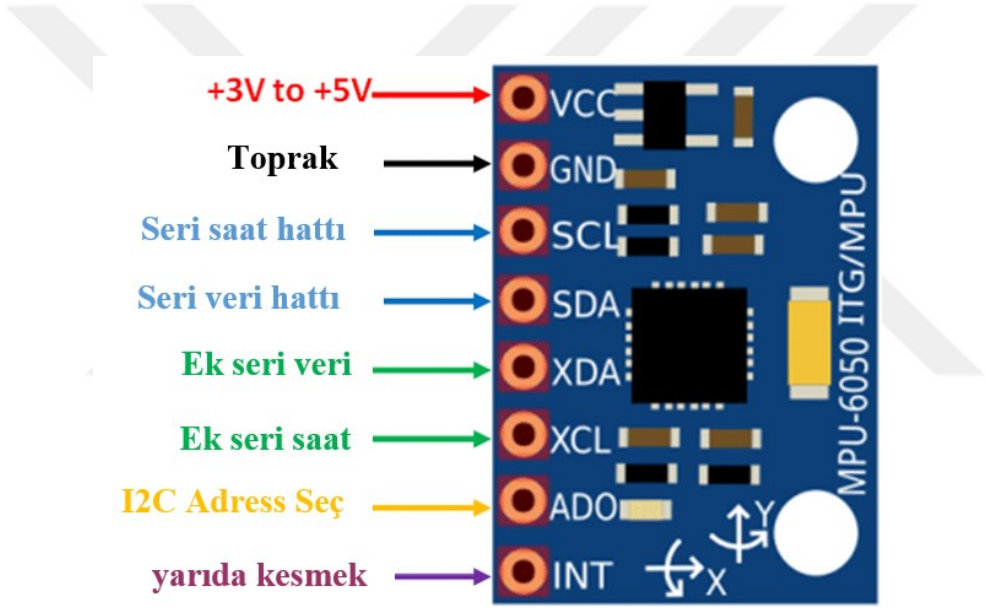
3.4. IMU Sensörler

Jiroskoplar ve ivmeölçerler tek başlarına yeterince anlamlı bilgi sağlamazlar. Bu nedenle, bu iki sensörün birleştirilmesiyle, yönelim, hız, pozisyon gibi bilgileri tek bir üniteden alınabilir. Şekil 3.5 ile verilen çoklu sensör ünitesine IMU (Ataletsel Ölçüm

Ünitesi) denir. IMU'nun temel görevi, bir cihazın veya aracın hızını, dönme açısını ve eğim miktarlarını belirlemektir. Bu sensör içerisinde 3 eksenli jiroskop ve 3 eksenli ivmeölçer bulunur. Bu sayede, cihazın 6 eksendeki hareketi algılanabilir.

İvmeölçer, üç eksende üç ayrı analog sinyal üretir. Bu sinyaller, cihazın lineer ivmesini (yani, yerçekimi ve diğer kuvvetlerin etkisiyle oluşan ivmeyi) ölçmek için kullanılır.

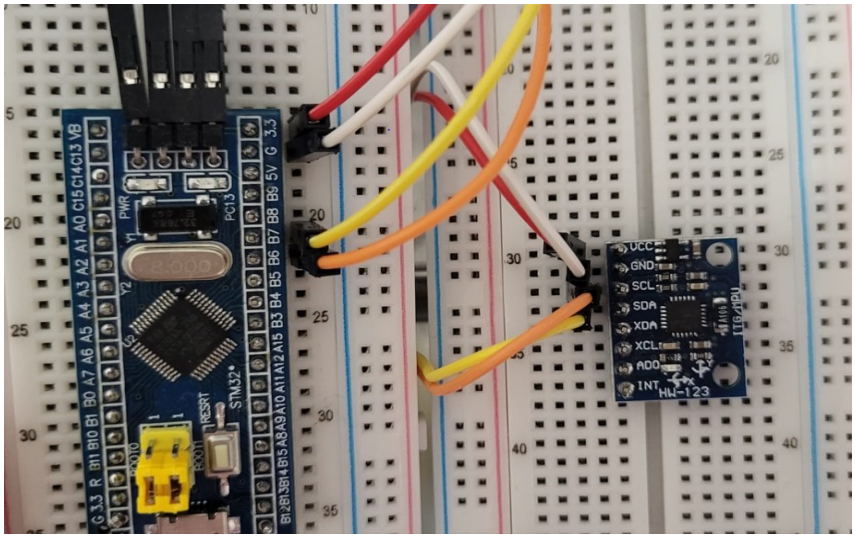
Jiroskopun temelde iki önemli özelliği vardır. Birincisi, yatay eksende dönen bir jiroskop, yatay eksende bir kuvvet uygulandığında eksen etrafında dönmeye başlar. Diğer özelliği ise jiroskopun dönüş ekseninin sabit kalmasıdır. Bu özellikler sayesinde jiroskop, cihazın açıklık ve dönme hızını ölçmek için kullanılır.



Şekil 3.5. Altı Eksen IMU sensörü

3.4.1. IMU Sensörden Veri Okuma

Uygulamada IMU sensörden veri okuma yani IMU sensör ham değerlerini okuma işlemi için şu adımları yürütebiliriz. Öncelikle Şekil 3.6'da görüldüğü gibi uygun bağlantının yapılması gereklidir. Bunun için içerisinde MPU6050 elemanı bulunan IMU sensör bağlantısının STM32F103C6T8 mikrodenetleyiciye yapabilmek için, IMU sensördeki V_{cc} beslemesinin 3.3V´a, toprağı ise kaynağın toprağına bağlamalıyız, daha sonra SCL pinini STM32F103C8T6'daki B7 pinine ve SDA pinini de B6 pinine bağlamalıyız. Tüm bağlantılar aşağıda Şekil 3.6'da verilmektedir.

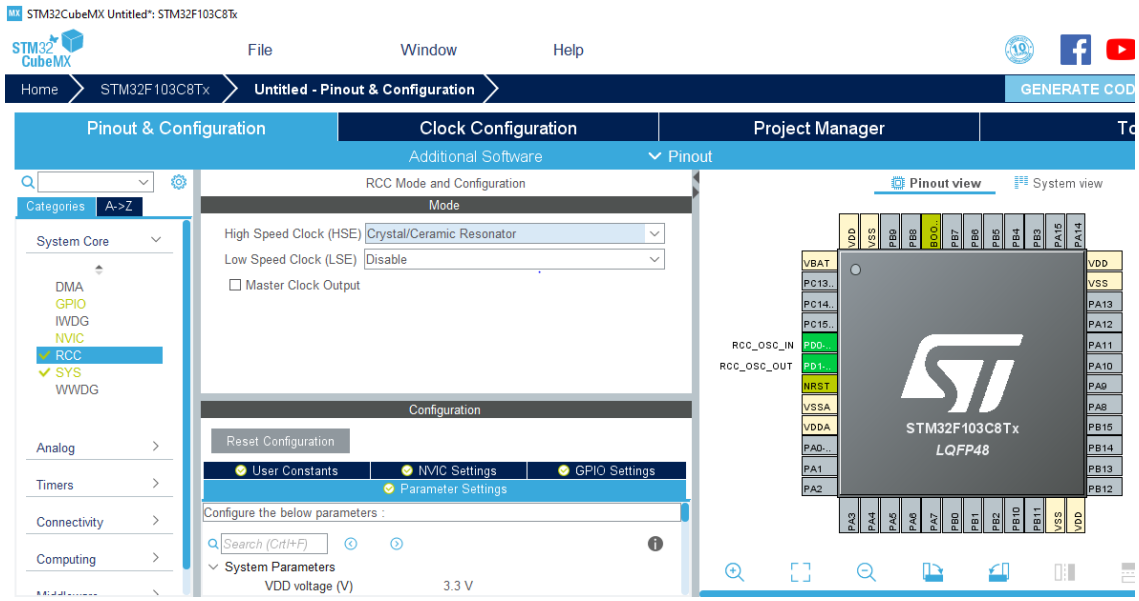


Şekil 3.6. IMU sensörü ile STM32F103C6T8 bağlantısı

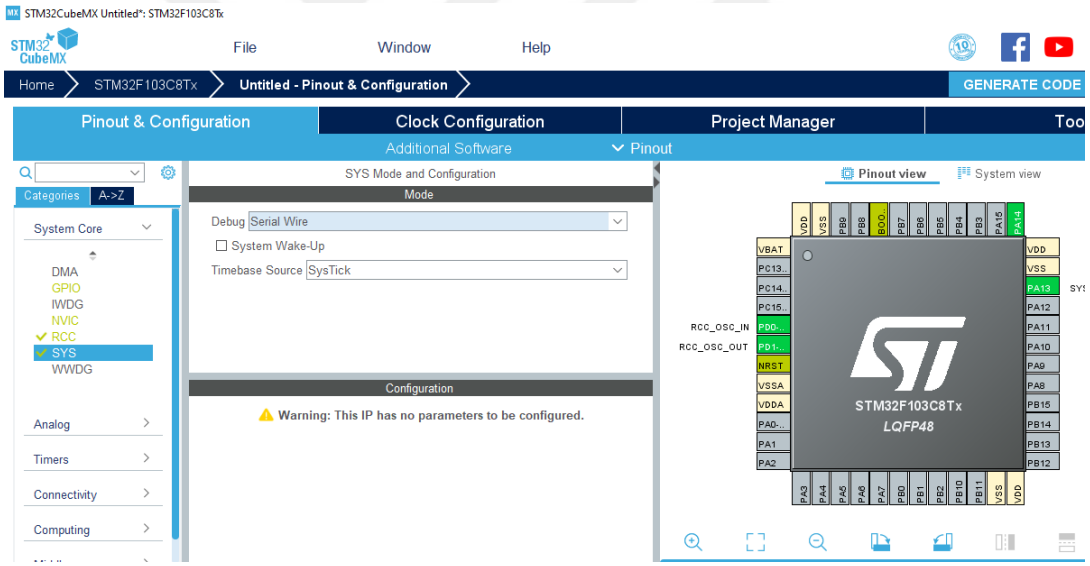
Mikrodenetleyiciyi programlamak ve I2C haberleşmesini sağlamak için STM32CubeMX ve HAL kütüphanesini kullanılmaktadır.

Şekil 3.7'de görüldüğü gibi öncelikle STM32CubeMX yazılımında proje açıp STM32F103C6T8 kartın seçilir ve stm32 ortamını aşağıdaki gibi kurulur.

- "System Core" kategorisinden "RCC" (Saat konfigürasyonu) seçeneğine tıklayıp, High speed clock (HSE) de: Crystal/Ceramic Resonator seçilmesi gerekir.
- Sıfırlama ve Saat Kontrolü (RCC) ayarları yapılır. Mikrodenetleyicinizin saat konfigürasyonlarını düzenleme işlevini yerine getirir. Burada mikrodenetleyicinizin saat konfigürasyonlarını ayarlayabilir. Hızları, kaynakları ve diğer clock ayarlarını bu bölümde düzenleyebilirsiniz.
- Daha sonra "System Core" kategorisinden "SYS" seçeneğine tıklayıp Bebug (Serial Wire) Şekil 3.8'de gibi seçilmesi gerekir.
- Sistem Konfigürasyonu (SYS) yapılmalıdır. SYS bağlantıları, mikrodenetleyici projenizin genel sistem konfigürasyonlarını içermektedir. Bu konfigürasyonlar, mikrodenetleyicinin temel özelliklerini ve çalışma koşullarını belirlemeye yöneliktir.



Şekil 3.7.RCC Sıfırlama ve Saat Kontrolü ayarlanması



Şekil 3.8. SYS, Sistem konfigürasyonu ayarlanması

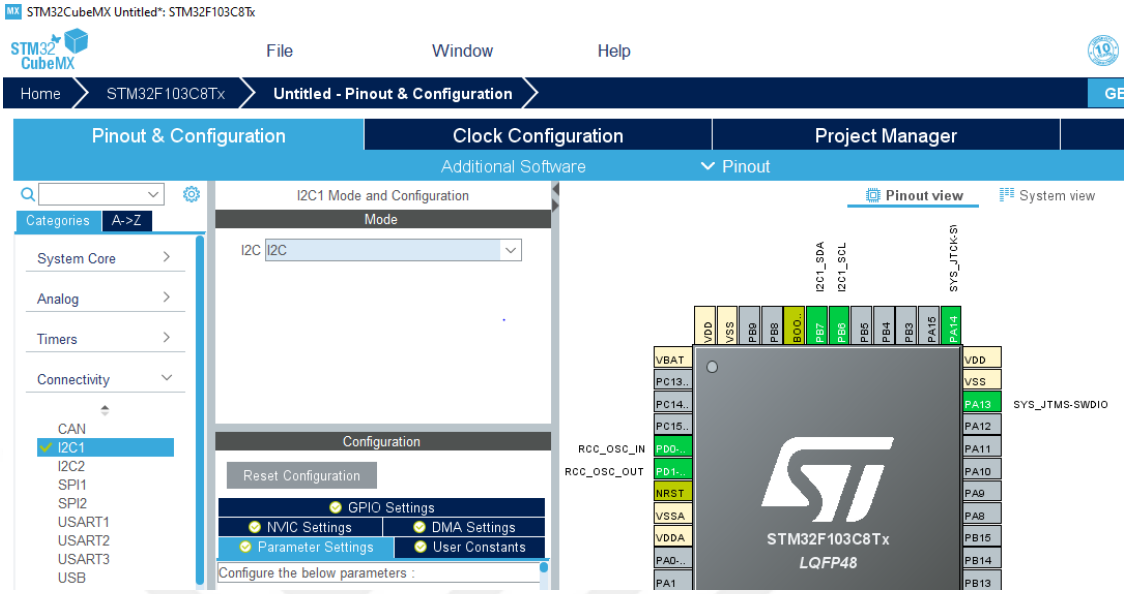
STM32 ile MPU6050'nin I2C Haberleşmesi:

STM32 mikrodenetleyicileri ile MPU6050 sensörünü kullanmak için I2C haberleşmesi gereklidir. I2C, kısa mesafeli seri bir haberleşme protokolüdür ve birden fazla cihazın aynı veri hattını paylaşmasına izin verir. I2C protokolü, iki hatlı bir sistem üzerinde çalışmaktadır. Bunlar;

SDA (Seri Veri): Veri hattıdır ve cihazlar arasında seri veri iletimini sağlamaktadır.

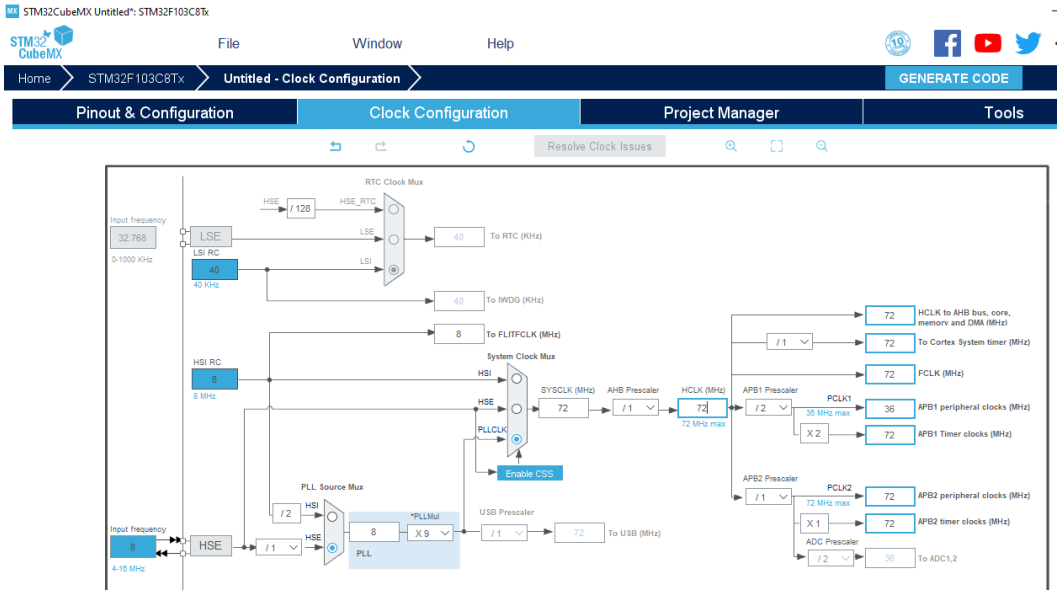
SCL (Seri Saat): Saat hattıdır ve cihazlar arasında senkronizasyonu sağlamaktadır.

I2C haberleşmesini yapabilmek için aşağıdaki Şekil 3.9'da gösterildiği gibi Connectivity kısmında I2C modu etkinleştirilmelidir.



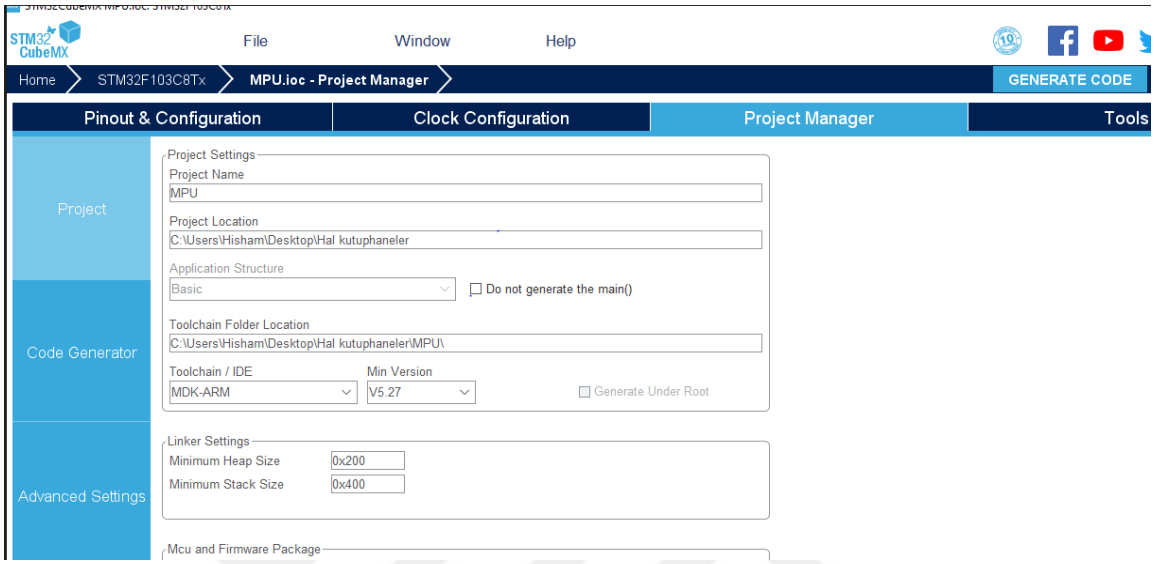
Şekil 3.9. STM32 ile MPU6050'nin I2C uygulaması

GPIO ayarlarında I2C'nin SDA ve SCL hatları mevcuttur. Kullanıldığı mikrodenetleyiciyi kartında PB6 ve PB7 pinleri mevcuttur. Uygulamada bu pinler ilgili MPU6050 hatlarına bağlanmıştır. Saat konfigürasyon ayarları Şekil 3.10'da gösterildiği gibi olmaktadır.

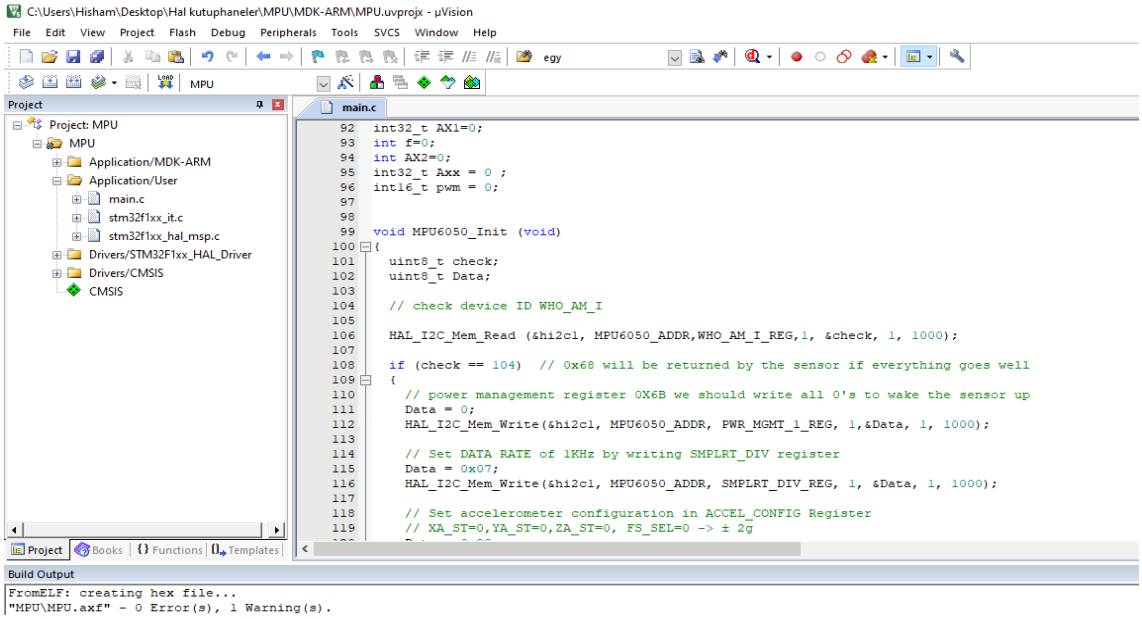


Şekil 3.10. Saat konfigürasyon ayarları

Saat konfigürasyon ayarları yapıldıktan sonra STMCubeMX programındaki project manager kısmında isimlendirilen proje için kod üret kısmına basarak Keil µVision yazılım programına dönüştürme yapılmış olur. Şekil 3.11 ve 3.12'de gösterilmiştir.



Şekil 3.11. Proje yöntemi

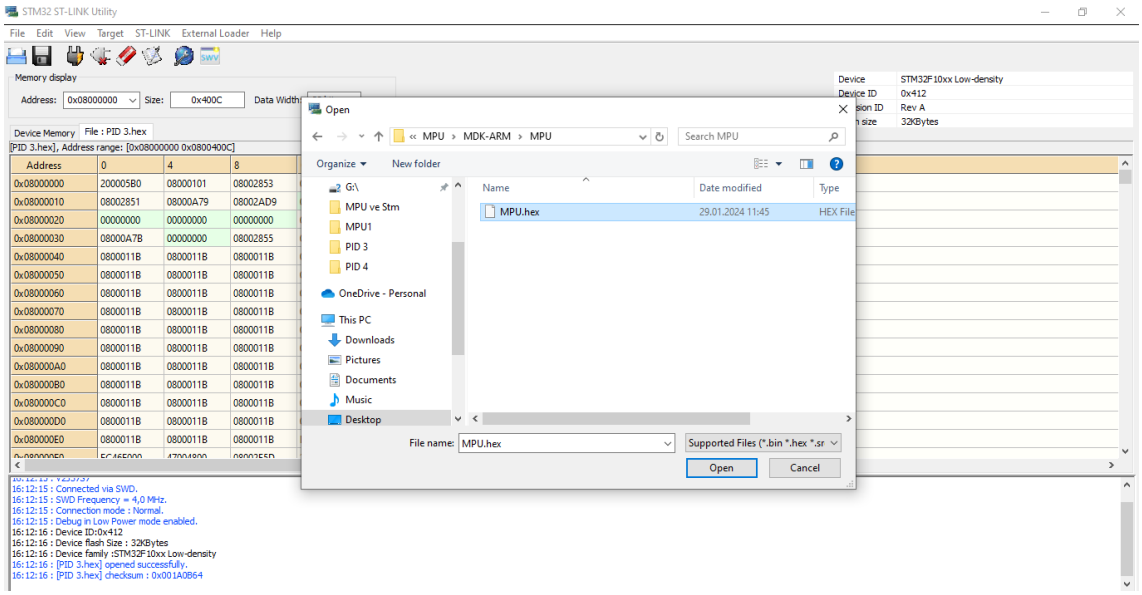


Şekil 3.12.Keil uvision projesini oluşturma

Kod oluşturma işlemi tamamlandığında µVision projesinin açılıp açılmayacağı sorulur. STM32CubeMX'den konfigürasyon aracından elde edilen kod dosyaları Keil

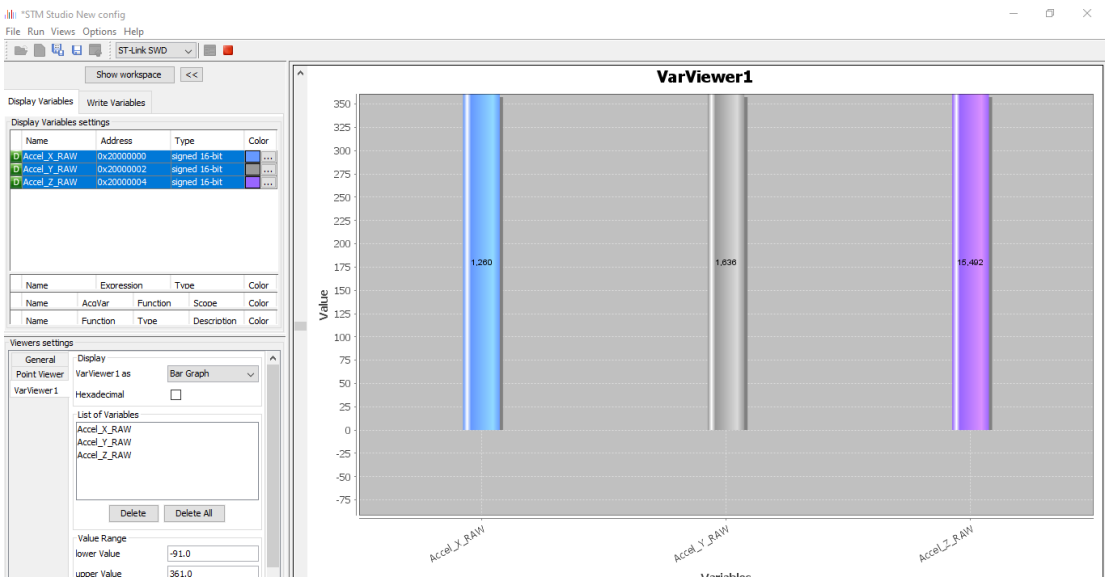
uVision projesine eklenmiş olur. Bu dosyalar genellikle main.c, stm32fxxx_hal_conf.h, stm32fxxx_hal_msp.c ve diğer sistem dosyalarını içerir. main.c ana dosya kodunu içermektedir. Bu dosya projenizin ana kod dosyasıdır. Mikrodenetleyici üzerinde yapmak istenilen işlemler bu kısımda programlanmaktadır. MPU6050 sensörüne ait kodlar eklenerek program çalıştırılmaktadır.

Keil'de yazılım kısmına, STM kartımız ile MPU6050 sensörü bağlantıları yaptıktan sonra Şekil 3.13'te görülen St-link Utility programını açıp hex dosyası eklenmiştir.



Şekil 3.13. St-Link utility programı

Son aşamada STMstudio programı açılarak axf dosyası eklenmiş ve STMStudio'da "Start Recording" düğmesine tıklayıp sensör verilerini izlemek için eklenilen değişkenler seçilerek sensörün x eksenli ve y eksenli hareket verileri görülebilmektedir. Yani IMU sensörü ham değerleri okunmuştur ve sensörden gelen değerler STMStudio arayüzünde Şekil 3.14'te görünmektedir. Burada, sensör X eksenini sola kaydırırsak pozitif tarafa doğru değer artacak, sağa kaydırırsak negatif tarafa doğru değer azalacaktır. Sensör Y eksenini ileriye kaydırırsak negatif tarafa doğru değer artacak, geriye kaydırırsak pozitif tarafa doğru değer azalacaktır. Sensörü döndürdüğümüzde yaptığımız harekete bağlı olarak Z değeri değişir.



Şekil 3.14. MPU6050 Sensöründen ham değerlerin okunması

3.4.2. Sensör Füzyonu

Sensörler, elektronik cihazların dışarıdan veri almasına yardımcı olan önemli bileşenlerdir ve bu alanda büyük bir çeşitlilik gösterirler. Her uygulama için farklı sensör çeşitleri geliştirilmiştir; sıcaklık, renk, eğim, ivme gibi çeşitli ölçümler için kullanılır. Askeri, endüstriyel ve eğitim alanlarında, sensör verileri geniş bir yelpazede kullanılmaktadır.

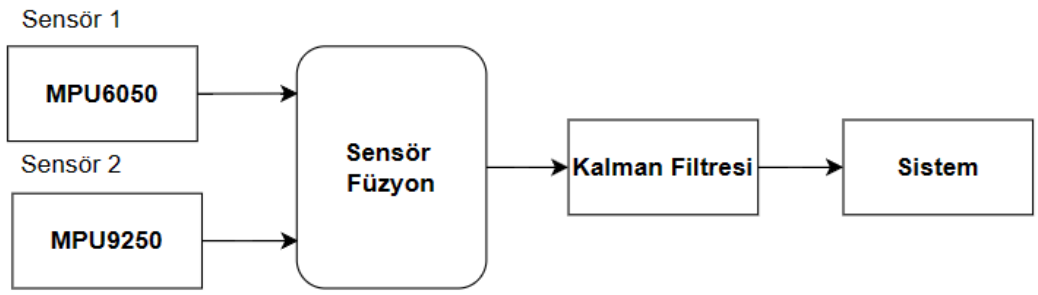
Çeşitli sensör gruplarının kullanılmasıyla elde edilen veriler, tek bir sensörden alınacak verilere göre daha doğru sonuçlar sağlar. Sensörlerin birleştirilmesi ve oluşturulan sensör füzyonu ile elde edilen veriler, algoritma, filtre ve yapay zekâ gibi tekniklerle değerlendirilir. Bu sayede sensör verilerinin daha doğru ve anlamlı bir şekilde yorumlanması sağlanır. Sensör füzyonu, sensörlerden gelen verilerin karşılaştırılması ve en doğru sonuçların elde edilmesi için önemli bir yöntemdir ve bu yöntemle sensör verilerinin etkili bir şekilde kullanılması sağlanır.

İnsansız hava aracı kontrol sistemi, geri beslemeli kontrol yöntemini kullanmaktadır. Bu yöntemde, kontrolcüler uçuş esnasında kontrol ettikleri durum değişkenine ilişkin anlık verilere ihtiyaç duyarlar. Dolayısıyla, hava aracının kararlı bir şekilde kontrol edilebilmesi için yüksek doğruluk ve kesinlikte davranış ve konum verilerine ihtiyaç vardır. İnsansız hava araçları, davranış ve pozisyon durum değişkenlerini uçuş kontrol sistemi için IMU sensör verileri ile temin etmektedir.

Tasarımda kullanılan MPU6050 ve MPU9250 iki farklı IMU sensörü birleştirerek sensör füzyonu ve Kalman filtresi kullanarak veri toplama ve işleme, daha doğru konum ve oryantasyon tahminleri elde edilmişti. MPU6050 ve MPU9250, ivmeölçerler ve jiroskoplar gibi farklı sensörlerin bir araya getirildiği popüler IMU (İvmeölçer ve Jiroskop Birimi) sensörleridir. Bu sensörler, hareket ve oryantasyon bilgilerini ölçmek için kullanılır.

Sensör füzyonu işleminde ilk adım, her bir sensör verilerinin alınması gerekmektedir. MPU6050 ve MPU9250 sensörlerinden birini yükselen kenar, diğerini ise düşen kenar bağlayarak veri çakışması riskini azaltabilir ve verileri daha güvenilir bir şekilde toplanabilir ve ortalamasını alınabilir. MPU6050 ve MPU9250'den gelen ivmeölçer ve jiroskop verileri bir araya getirilerek, hava aracının konumunu ve oryantasyonunu daha doğru bir şekilde tahmin edilir. Sensör füzyon sisteminin blok diyagramı aşağıdaki Şekil 3.15'te gösterilmiştir.

Son olarak, elde edilen veriler Kalman filtresi ile işlenebilir. Kalman filtresi, sensör verilerini bir tahmin modeliyle birleştirir ve gerçek zamanlı olarak hata düzeltmeleri yaparak daha hassas verileri elde etmek için kullanılır. Bu sayede, hava aracının konumu ve oryantasyonu daha hassas bir şekilde kontrol edilebilir.



Şekil 3.15. Sensör Füzyon blok diyagramı

Bu yöntemlerin bir araya getirilmesi, insansız hava araçlarının daha kararlı ve güvenilir bir şekilde uçmasını sağlayabilir. Özellikle hassas ve doğru konum ve oryantasyon bilgileri gerektiren uygulamalarda, sensör füzyonu ve Kalman filtresi kullanmak önemlidir.

3.5. Fırçasız DC (BLDC) Motorlar

Döner Kanatlı İnsansız hava araçlarının uçmasını sağlayacak temel parçalardan biri fırçasız motorlardır. Fırçasız motor seçiminde dikkat edilmesi gereken özellikler KV (rpm/volt) değeri, pervane uygunluğu, kaldırma kuvveti, verimlilik, gereken voltaj değeri vb. özelliklerdir. Bu özellikler tasarımın amacına göre seçilmektedir. KV değeri, BLDC motorlarının özelliklerinden biridir ve motorun dönme hızını belirtir. Bu değer, her bir voltajda motorun devir sayısını ifade eder. Yani, her bir voltajda motorun ne kadar devir yapacağını gösteren bir ölçüdür. Daha yüksek KV değerine sahip bir motor, daha yüksek bir devir hızına sahip olurken, daha düşük bir KV değerine sahip motor daha yavaş bir devir hızına sahip olacaktır. KV değeri, motorun özellikle model uçak ve drone uygulamalarda hangi pil gerilimine uygun olduğunu belirlemede önemlidir.

2200KV değere sahip bir fırçasız motora 1 V gerilim yüklendiğinde dakikada 2200 devir dönebilmektedir. Fırçasız motorların seçiminde dikkat edilmesi gereken ikinci konu ise motorların tekrarlanabilir ve stabil olmasıdır. KV değeri motora her bir volt verildiğinde motorun dakikada kaç tur atacağını gösterir.

Bir fırçasız motor iki ana bölümden oluşur bunlar rotor ve stator kısımlarıdır. Rotor mıknatıstır stator ise bobinlerden oluşmaktadır. Bobinlere akım uygularsak bobinler bir manyetik alan oluşturur, manyetik alan çizgileri ve rotor kutupları sürekli değiştirilerek hareket sağlanır.

Uygulamada kullanılan A2212/6T 2200KV Fırçasız DC motorlar; Uçaklar, tekneler, araçlar ve DIY kitleri için uygundur. 6T (BLDC) fırçasız DC motor, özellikle RC hobileri için tasarlanmış en popüler yüksek hızlı fırçasız motorlardan biridir. Motor, dış kasanın döndüğü, iç kısmın ise sabit kaldığı öncü tipindedir. Düşük maliyetli olması nedeniyle piyasadaki popüler modellerden biridir. Küçük drone ve uçaklar için tercih edilmektedir. Motor, düzgün ve istikrarlı bir dönüş hareketi için 3,17 mm sertleştirilmiş çelik şafta ve çift bilyalı rulmanlara sahiptir. Mermi tipi pervane adaptörü ve çapraz motor montajı dahildir. Üç fazlı motor olduğundan hızı ve verimi yüksektir. Bir motor montajı, kubbe tipi alaşımlı pervane döndürücü/adaptör, vidalar ve 3,5 mm muz konnektörler (Kurşun fişler) ile gelmektedir.

Aşağıda Şekil 3.16'da uygulamada kullanılan BLDC motoru gösterilmektedir. Her bir motor 10A çekmektedir ve deneysel düzenekte yan yana iki motor kullanıldığı için toplam 20A çekilmektedir. Pervaneler, model uçağın uçuş performansında önemli bir rol oynar. Her motor için bir pervane olmak üzere toplamda dört pervane kullanılmaktadır.

Her motorun şaftına bir pervane takılmaktadır. Pervaneler, motorların kaldırma kuvveti ve verimliliği üzerinde doğrudan etkili olduğundan, büyüklüklerine göre titizlikle seçilmelidir. Pervane seçimi yaparken dikkat etmemiz gereken önemli faktörler şunlardır:

Boyut ve Çap: Pervanenin boyutu ve çapı, uçağın hava akışını nasıl yönlendireceğini ve motorun sağlayabileceği itici gücü belirler. Uçağın tasarımına ve motorun özelliklerine uygun boyutta bir pervane seçilmelidir.

Malzeme ve Yapı: Pervanelerin yapısı ve malzemesi, dayanıklılık, ağırlık ve aerodinamik özellikler açısından önemlidir. Hafif ancak sağlam ve aerodinamik olarak etkili malzemeler tercih edilmelidir.

Dönme Hızı: Pervanelerin dönme hızı, motorun verimliliği ve kaldırma kuvveti üzerinde etkilidir. Doğru dönme hızı seçimi, optimum uçuş performansını sağlar.

Denge ve Dönme Özellikleri: Pervanelerin dengesi ve dönme özellikleri, uçakta titreşimleri ve uçuş stabilitesini etkiler. Dengeli ve düzgün dönen pervaneler tercih edilmelidir.

Şekil 3.16. Fırçasız DC motor

Gerçekleştirilen sistemde yan yana kullanılan iki motor için iki pervane kullanılmaktadır ve kullandığımız pervanelerin çapları motorlara uygun bir şekilde oturması sağlanmaktadır.

3.5.1.BLDC Motor Sürücü (ESC)

Elektronik Hız Kontrol Cihazı (ESC), İHA sistemlerinde önemli bir parçadır ve genellikle en çok kaza sebebi olan parçalardan biridir. Bu nedenle, yüksek kaliteli ve önerilen ürünlerin kullanılması büyük önem taşır. Profesyonel sistemlerde, kaliteli ve yüksek amper değerine sahip ESC'ler tercih edilir.

ESC'ler genellikle servo sinyal çözücü, işlemci (MCU), ve motor akım yükseltme devresi gibi bileşenleri içerirler. İşlevsel olarak, ESC'ler, kullanıcının verdiği komutlara (genellikle bir pilot tarafından verilen kontrol sinyallerine) yanıt olarak motor hızını kontrol eder. Bu, bir uçak, helikopter, multikopter veya diğer elektrik motorlu araçların hareketini sağlamak için önemlidir. Elektronik Hız Kontrol Cihazları (ESC'ler), fırçasız motorları kontrol etmek için kullanılan devre sistemleridir. ESC'lerin çalışma prensibi, motorların sargılarına PWM dalgaları göndermektir. Fırçasız motorlar, trifaze motorlardır ve üç ayrı uçtan sürülürler. Pilin iki ucundan aldıkları DC gerilimi, üç kablo ile AC olarak motora gönderirler. Bu şekilde, motorun dönme yönü ve hızı kontrol edilir.

Fırçasız motorlar için kullanılan ESC'ler, bir frekans konvertörü gibi çalışırlar. Yani, devir sayısını değiştirmek için gerilime bağlı değil, frekansa bağlı olarak PWM dalgalarının sıklığını ve genişliğini ayarlarlar. Bu sayede motorun hızını ve dönüş yönünü kontrol edebilirler. Bu karmaşık yapıya sahip ESC'ler, fırçasız motorların daha verimli ve hassas bir şekilde kontrol edilmesini sağlarlar. Bu sayede, uçuş sistemlerinde daha iyi performans ve güvenilirlik elde edilir.

Bu çalışmada kullanılan 20 amper değerli ESC'nin üst kısımdaki üç çıkış uçları motora bağlanmakta, diğer iki giriş ucu (kırmızı ve siyah) besleme adaptörünün eksi ve artı ucuna bağlanmaktadır. Ortadaki beyaz ve siyah uçlar ise STM32F103FC8 mikrodenetleyici kartındaki toprak ve zamanlama uçları olarak kullanılan pinlere bağlanmıştır (Şekil 3.17).

Şekil 3.17. 20A ESC

3.5.2. BLDC Motor Hız Kontrolü Kalibrasyonu

ESC'ler, fırçasız motorların kontrolünde önemli bir role sahiptir ve aldıkları PWM sinyallerini bir mikrodenetleyici yardımıyla işlerler. PWM darbelerinin genişliklerine bağlı olarak, ESC'ler fırçasız motorların hızlarını ayarlayarak motorun devrini kontrol ederler. Bu sayede, kullanıcı istediği hız ve yönde motoru kontrol edebilir.

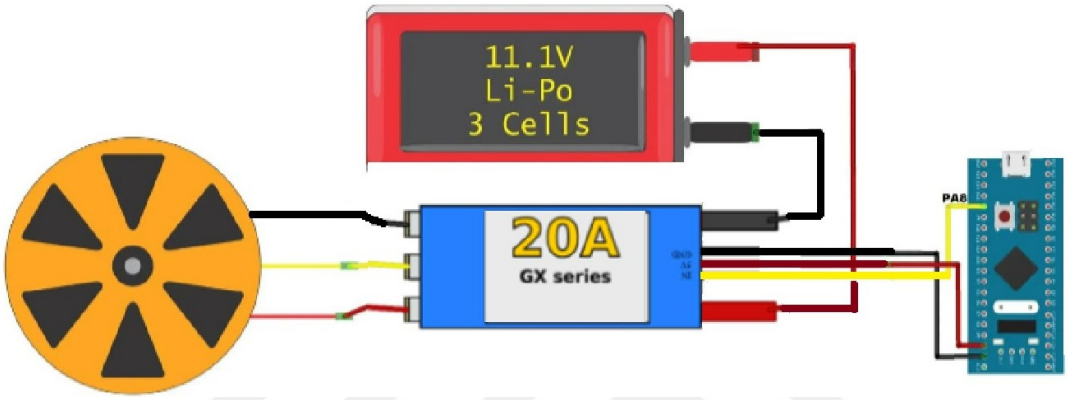
Fırçasız motorları kontrol etmek için kullanılan ESC'ler aldığı PWM sinyallerini içerisinde bulunan mikrodenetleyici MCU ile işleyerek gerçekleştirirler. Bu işlemi gerçekleştirmek için, genellikle mikrodenetleyicinin pinlerine alternatif fonksiyon ataması yapılır. STM32F103 gibi mikrodenetleyicilerde bulunan zamanlayıcı öğeleri, PWM sinyalleri üretmek için kullanılabilir. Bu zamanlayıcı öğelerini, mikrodenetleyicinin çıkış pinlerine alternatif fonksiyon olarak bağlayarak, farklı doluluk oranlarına sahip PWM sinyalleri üretebilmektedir. STM32F103 mikrodenetleyicisinde, bir zamanlayıcı öğesini kullanarak dört farklı pinin PWM sinyallerini üretebilmektedir. Bu pinler, aynı zamanlayıcı öğesini kullanarak farklı doluluk oranlarına sahip PWM sinyalleri üretebilir. Bu şekilde, ESC'ye gidecek olan PWM sinyalleri oluşturulmuş olur.

Bu işlem genellikle mikrodenetleyici programlama ortamlarında, örneğin Arduino IDE veya STM32CubeIDE gibi yazılım geliştirme araçları kullanılarak gerçekleştirilir. PWM sinyallerinin oluşturulması ve ESC'ye gönderilmesi, fırçasız motorun doğru şekilde kontrol edilmesini sağlar.

STM32, ESC ile fırçasız motorun hız kontrolünü yapmak için Stm32f103c8 kartı,20 A Esc, 3S adaptör, kablolar ve fırçasız motor gerekmektedir. Aşağıdaki özelliklere sahip bir fırçasız dc motor, 2200 KV derecesine sahip, 3S adaptör ve 20A ESC kullanılarak çalıştırılabilir.

Fırçasız motordaki KV derecesi, motorun volt başına devir/dakika cinsinden devir sayısını tanımlar. Örneğin, Bir volt uygulandığında motor, 2200 RPM hızında dönecektir. 2 volt uygulandığında motor 4400 RPM hızında dönecektir. 3 volt uygulandığında motor 6600 RPM hızında dönecektir. Buradan hareketle, "Adaptör 3S", bir pilin voltajını ifade eder ve genellikle lityum polimer (LiPo) piller için kullanılan bir terimdir. Bir LiPo pilin “S” sayısı, pilin kaç tane hücresi olduğunu ve her bir hücrenin 3.7V'lik bir gerilimi olduğunu gösterir, dolayısıyla, "3S" terimi, üç adet hücrenin seri olarak bağlandığı anlamına gelir. Bu nedenle, 3S LiPo pil, üç hücrenin seri olarak bağlanmasıyla elde edilir ve toplam nominal voltajı 3.7V x 3 = 11.1V olur. Böylece motorun maksimum 24.420 RPM'ye (11.1 x 2200) ulaşmasını sağlayabiliriz.

ESC'nin sinyal kablosu, genellikle bir üçgen sembolüyle işaretlenir. Bu kablo işlemciden kontrol sinyali alır ve gelen komutlara göre motor hızını kontrol eder. ESC'nin sinyal kablosunu işlemcinin PWM uyumlu bir pinine bağlayabiliriz. ESC'nin pozitif ve negatif güç kablosu, pil veya güç kaynağından gelen elektrik gücünü sağlar. Pozitif kırmızı, negatif siyah kabloya bağlanmaktadır (Şekil 3.18).



Şekil 3.18. STM32f103c8 ile DC motor hız kontrolü

Bağlantılar yaptıktan sonra STM kartına-kodu yüklemek gerekmektedir. Kontrol kartı çalışır durumdayken, ESC'nin güç bağlantısı yapılarak motorun tepki verip vermediği kontrol edilmelidir. Motorun düzgün bir şekilde çalıştığını ve hızının değiştiğini görmek için biraz beklemek gerekmektedir. Bu kod, STM kartın Zamanlayıcı kullanarak 50Hz PWM sinyali üretir ve bu sinyali ESC'ye gönderir. Kodu kartına yükledikten sonra, ESC'nin motorun hızını bu PWM sinyallerine göre kontrol etmesi gerekmektedir. PWM sinyalinin motor hızını doğru bir şekilde kontrol ettiğinden emin olmalı. 1000 ile 2000 mikro saniye arasındaki değerler verip değişen PWM sinyalleri göndererek motorun hızını ayarlayabilmektedir.

Deneysel uygulama için aşağıda Şekil 3.19 ile verilen adaptör S-480-12 kullanılmıştır. Adaptör S-480-12, 12V, 40A anahtarlamalı bir güç kaynağıdır.

Şekil 3.19. Adaptör S-480-12, 12V, 40A

3.6. Kalman Filtreleri

Kalman Filtreleri, bir dinamik sistemdeki durumları öngörmek için giriş ve çıkış bilgilerini kullanarak tahminlerde bulunan bir filtredir. Filtre modeli, önceki bilgilerle birlikte mevcut giriş ve çıkış bilgilerini dikkate alarak sistemdeki değişiklikleri izler ve gelecekteki durumları tahmin etmeye çalışır.

Kalman filtresi, en önemli ve yaygın kullanılan tahmin algoritmalarından biridir. Kalman filtresi, gizli değişkenlerin gürültülü ve belirsiz ölçümlerden yola çıkarak tahminini üreten güçlü bir algoritmadır (Gaeid 2013).

Kalman filtresi, bir sürecin belirsiz ve gürültülü ölçümlerle izlenmesi için kullanılan bir matematiksel algoritmadır. Bu filtre, bir sürecin durumunu belirlemek ve bu durumu optimize etmek için ölçümleri ve sistemin dinamik modelini birleştirir. Kalman Filtresi, 1960'larda Rudolf E. Kálmán tarafından geliştirilmiştir. Kalman Filtreleri günümüzde birçok alanda yaygın olarak kullanılmaktadır, özellikle kontrol sistemleri, uçuş sistemleri, hedef izleme sistemleri, konum ve navigasyon sistemleri ve bilgisayar grafikleri gibi alanlarda kullanılmaktadır.

Kalman filtresi, tahmin etmek, karşılaştırmak, güncellemek ve yeni bilgilerle daha iyi bir tahmin yapmak için sürekli olarak çalışan bir filtredir. Sürekli olarak tahmin hatalarını minimize ederek gerçek değere ulaşmayı amaçlar.

Kalman filtresi, matematiksel olarak karmaşık bir yapıya sahiptir, ancak bu yapı, sistemin durumunu tahmin etmek ve hataları minimize etmek için etkili bir araç sağlar.

Pratikte kullanılan birçok sistem dinamiktir ve modellenirken sistemi etkileyen birçok faktör ölçülemez. Bu ölçülemeyen faktörler nedeniyle, sınırlı bilgilere sahip bir

sistem hakkında çıkarım yapmak için Kalman filtresi oldukça uygun bir yöntemdir. Gerçekte, Kalman filtresi bilinmeyen bilgilerle bile sistem hakkında tahminler sunabilen ve bu tahminleri sürekli olarak istenilen değerlere yaklaştırmaya çalışan bir filtredir. Kalman Filtresi, geleneksel tahmin edicilerde bulunan filtreleme özelliğine ek olarak, sistemin ölçülemeyen durumlarını tahmin etmek için son derece güçlü ve yeteneklidir. Bu özellikleriyle, Kalman filtresi gerçek dünya sistemlerinin karmaşıklığını anlamak ve yönetmek için önemli bir araçtır. (Nedime 2019).

Kalman filtresi, gürültülü sistemlerde gerçek zamanlı hassasiyet sunan bir filtreleme yöntemidir. Ölçümleri ve matematiksel modeli kullanarak sistemin gerçek durumunu tahmin eder. Bunu, en küçük kareler yöntemi ile eğri uydurma ve sistemin matematiksel modelini kullanarak yapar.

Kalman filtresi, elde edilen tahminleri gerçek gözlemlerle karşılaştırarak çalışır. Tahminler ile gözlemler arasındaki fark, Kalman kazancı adı verilen bir parametre ile ölçeklendirilir. Bu kazanç, sistemin performansını iyileştirmek için ayarlanabilir ve büyük bir değere sahip olduğunda, filtre gözlemleri daha yakından takip ederek daha doğru tahminler sunabilir. Kalman kazancı, filtrenin ne kadar hızlı veya yavaş uyarlandığını belirler ve bu sayede sistemdeki belirsizliği azaltır.

Kalman filtresi, sensör füzyonu ve veri füzyonu gibi uygulamalarda da kullanılır. Gerçek zamanlı sistemler genellikle bir sistemin durumunu elde etmek için tek bir ölçüm yerine ardışık bir dizi ölçüm üretir. Bu ölçümler, matematiksel olarak birleştirilerek zaman döneminde sistemin durumunu tahmin etmek için kullanılır.

Kalman filtresi, doğrusal kuadratik Gauss problemini çözen bir tahminci olarak tanımlanmaktadır. Bu, lineer bir sistemdeki anlık durumu, lineer olarak doğrusal ancak Gauss gürültüsü ile bozulan bir ölçümden tahmin etme problemidir.

Kalman filtresi, minimum kareler yönteminin verimli bir özyinelemeli hesaplama çözümüdür. Elde edilen tahmin edici, tahmin hatasının herhangi bir kuadratik fonksiyonuna göre statik olarak optimaldir. Bu, tahminlerin gerçek duruma en iyi şekilde uyum sağladığı anlamına gelir.

Fiziksel bir sisteme uygulandığında, Kalman filtresi iki temel gürültü kaynağının etkisi altında olacaktır: işlem gürültüsü ve ölçüm gürültüsü. Bu gürültüler, sistemin gerçek durumunu doğru bir şekilde tahmin etmeyi zorlaştırabilir, ancak Kalman filtresi bu belirsizlikleri minimize etmeye çalışır.

Aşağıda Kalman filtresine dair temel özellikler sıralanmıştır:

a) Tahmin ve filtreleme: KF, sadece bir filtreleme aracı olmanın ötesinde, sistem durumunun tahmin edilmesi konusunda da etkilidir. Tahmin aşaması, filtreleme sürecinin önemli bir parçasıdır.

b) 20. Yüzyılın önemli Buluşu: KF, 1960'larda Rudolf E. Kalman tarafından geliştirilmiştir ve bu alandaki en önemli buluşlardan biri olarak kabul edilir. Kontrol teorisi, izleme ve tahminleme uygulamalarında geniş bir etki yaratmıştır.

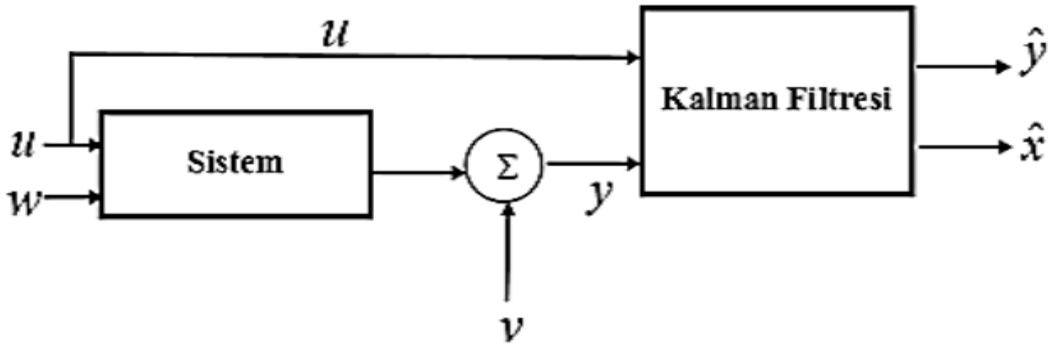
c) Bazı matematik alt yapılarla Kalman filtresine tam olarak hâkim olmak mümkündür:

d) Kalman filtresi, matematiksel temele dayanır ve matris hesaplamalarını içerir. Ancak, temel prensipleri anlamak ve uygulamak için özel bir matematiksel bilgi seviyesine ihtiyaç duyulmaktadır.

e) Yazılım uyumluluğu: KF, yazılım uygulamalarında sıklıkla kullanılan bir algoritma türüdür. Özellikle gerçek zamanlı uygulamalarda ve gömülü sistemlerde yaygın olarak kullanılır.

f) Rekürsif yapı: Kalman filtresi, rekürsif bir metot kullanır. Yani, her bir zaman adımında geçmiş tahminlerden türetilen yeni tahminler kullanılarak sürekli olarak güncellenir.

g) Denklemleri ve matrisleri karmaşık olabilir, ancak bazı durumlarda ihmal edilebilir veya göz ardı edilebilir, çünkü bazı basitleştirme ve yaklaştırma teknikleri kullanılabilir ve uygulanabilir. Sistemin en genel blok diyagramı aşağıdaki Şekil 3.20'de verilmiştir.



Şekil 3.20. Kalman filtresi blok diyagramı (Nedime Aydın, 2019)

Kalman Filtresi, sistemdeki girişe (u) ve ölçülen gürültülü çıkışa (y) bakarak, sistemin gerçek çıkışını ve durumunu tahmin etmeye çalışır. İşlem gürültüsü olarak adlandırılan (w), idealden sapmaları temsil ederken, ölçüm gürültüsü olarak adlandırılan (v), ölçüm sırasında meydana gelen gürültüleri ifade eder. Kalman Filtresi, bu giriş ve çıkış bilgileri ile sistemin gerçek çıkışını tahmin etmek için $\hat{Y}$ ve sistemin durumunu tahmin etmek için $\hat{X}$ değerlerini hesaplar. Bu tahminler, işlem ve ölçüm gürültülerini dikkate alarak, sistemin gerçek durumuna en yakın değerleri elde etmeyi amaçlar.

3.6.1. Kalman Filtresinin Matematiksel Modeli

Kalman filtresinin en basit matematiksel gösterimi aşağıdaki gibi ifade edilebilir. Burada; ifadelerdeki alt indis olan k harfi, durum denkleminde ayrık zaman aralıklarını temsil eder. Örneğin, k=1 olduğunda 1. Örnek, k=5 olduğunda ise 5. Örnek olarak kabul edilebilir. Buradaki amaç, x sinyalinin tahmini olan $\hat{X}_K$ değerini bulmaktır. Her bir k için bu değer bulunmaya çalışılır, Z_k ölçülen değeri ifade eder. Ancak bu değerin doğruluğu kesin olarak bilinemez, aksi halde tüm bu hesaplamalara gerek kalmazdı. Burada en önemli parametre K_K Kalman kazancıdır ve $\hat{X}_{k-1}$ değeri ise sinyalin önceki durumdaki tahminidir. Denklemin bilinmeyen elemanı K_K Kalman kazancıdır. Zira ölçülen değer (Z_k) ve önceki tahmin edilen sinyal $\hat{X}_{k-1}$ zaten bilinmektedir. Her bir durum için Kalman kazancı Denklem (3.1) ile hesaplanmalıdır.

$$\hat{X}_K = K_k Z_k + (1 - K_k)\hat{X}_{k-1} \tag{3.1}$$

Kalman filtresi ile durumları $X \epsilon R^n$ tahmin etmek ve $Z \epsilon R^m$ ölçüm değeri için aşağıdaki doğrusal fark denklemleri kullanılmaktadır.

$$X_k = A_{k-1} + Bu_k + W_{k-1} \tag{3.2}$$

$$Z_k = H_K x_K + v_k \quad (3.3)$$

Burada;

- X_k K zamanındaki sistem durum vektörü
- Z_k K zamanındaki ölçülen değer vektörü
- W_k Giriş (işlem) gürültüsü
- u_k Kontrol sinyali
- v_k Ölçüm gürültüsü hareket tahmini sırasında gerçekleyebilecek hataları kapsamaktadır.
- W_k ve v_kBirbirinden bağımsız normal dağılıma sahip, beyaz gürültü olduğu varsayılır.

$$P(w) \approx N(0, Q) \quad (3.4)$$
$$P(v) \approx N(0, R) \quad (3.5)$$

Pratikte, giriş gürültü kovaryansı Q ve ölçüm gürültü kovaryansı R matrisleri her ölçüm aralığında değişir fakat bunların sabit olduğunu varsayılır.

Sinyalin her bir X_k değeri, Denklem (3.4) kullanılarak bulunur. Bu denklemde, herhangi bir X_k değeri önceki değerinin (X_{k-1}) üzerine kontrol sinyali (u_k) ve önceki işlem gürültüsü (W_{k-1}) eklenerek oluşturulan lineer bir kombinasyonla elde edilir.

Herhangi bir ölçüm değeri (Z_k), sinyalin değeri (X_k) ile ölçümün gürültüsünün (v_k) lineer bir kombinasyonu olarak Denklem (3.5) ile ifade edilir. Bu denklemde, ölçüm değeri (Z_k), sinyalin gerçek değeri (X_k) ve ölçüm gürültüsü (v_k) arasındaki ilişkiyi açıklar ve ölçümün gerçek değerden sapmasını temsil eder.

A, B ve H elemanları matrislerin genel gösterimidir. Bu değerler genellikle sabit varsayılır ve değişebilmesine rağmen, sabit kabul edilerek kolaylık sağlanır.

A Matrisi: Doğrusal fark denkleminde kullanılır. Sistemin bir önceki K-1 durumdan, k durumuna geçişini temsil eder. Boyutu n×n'dir. Pratikte sabit kabul edilir, ancak her k durum değişiminde değişmesi beklenir.

B Matrisi: Kontrol girişini u temsil eder. Boyutu nx1'dir. U ϵ R her k durumu ile ilgilidir.

H Matrisi: Ölçüm denkleminde kullanılır. Sistemin K durumundaki ölçüm değerlerini temsil eder. Boyutu m×n'dir. Pratikte sabit kabul edilir, ancak her k durum değişiminde değişmesi beklenir.

Bilindiği gibi hiçbir sinyal Gauss değildir. Ancak, Kalman Filtresi doğru tahminlere doğru yakınsamaya çalışır. Bu nedenle, sinyali bir miktar yaklaşık Gauss olarak kabul etmek genellikle mümkündür.

3.6.2. Kalman Filtreleri ile Durum Tahmini

Kalman filtresi, Şekil 3.21'de görüldüğü gibi iki temel adımdan oluşur: tahmin ve düzeltme güncellemedir. İşlem ve ölçüm gürültüsünü içeren stokastik sistemlerde, bu filtre, belirli bir durumun gelecekteki tahminini yapmak ve ardından gerçek ölçümlerle bu tahmini güncellemek için kullanılmaktadır. Bu sayede, sistemdeki belirsizlikler ve gürültülerle başa çıkabilir ve daha hassas durum tahminleri elde edebilir.

Şekil 3.21. Ayrık kalman filtre döngüsü (Gayıroğlu,2012)

Durum uzay modeli, sistemdeki temel değişkenleri ve bu değişkenler arasındaki ilişkileri ifade eder. Kalman filtresi, bu modeli kullanarak durumu tahmin eder. Kovaryans matrisleri, işlem ve ölçüm gürültülerinin özelliklerini ifade eder ve bu matrislerin bilinmesi önemlidir.

Eğer durum uzay modeli ve kovaryans matrisleri zamanla değişiyorsa, zamanla değişen Kalman filtresi tercih edilir. Bu durumda, sistemdeki değişikliklere daha iyi uyum sağlayabilir. Ancak, eğer bu değerler sabitse, sürekli-durum Kalman filtresi kullanmak daha uygun olabilir.

3.6.3. Sürekli Zamanlı Kalman Filtresi

Sürekli zamanlı Kalman filtreleri, zamanla sürekli olan denklemlerle ifade edilir. Bu nedenle, önceki ve sonraki güncellemeler gibi ayrık zamanlı kontrol sistemlerindeki gibi belirgin adımlar bulunmaz. Ancak, sürekli zamanlı filtreyi ayrıklaştırarak, Kalman

filtre denklemleri elde edilebilir (Lewis ve ark., 2017). Bu işlem, sürekli zamanlı sistemlerin ayrık zamanlı sistemlere dönüştürülmesini sağlar ve bu sayede Kalman filtrelerinin uygulanabilirliğini artırır. Aşağıda verilen sistem dinamikleri varsayımı yapılırsa;

$$\dot{X} = AX + Bu + Fw \tag{3.6}$$

$$Y = CX + Du + v \tag{3.7}$$

burada w işlem gürültüsünü, v ise ölçüm gürültüsünü temsil etmektedir.

Sürekli zamanlı kalman filtresi, durum tahmini ve güncelleme adımlarını içeren aşağıdaki gibi dört diferansiyel denklemle ifade edilmektedir.

$$\dot{\hat{X}} = A\hat{X} + Bu + K\left(\dot{y} - C\hat{X} + Du\right) \tag{3.8}$$

$$\dot{P} = AP + PA^T + FQF^T - KRK^T \tag{3.9}$$

$$K = PC^T R^{-1} \tag{3.10}$$

$$\dot{P} = AP + PA^T + FQF^T - PC^T RCP \tag{3.11}$$

Burada:

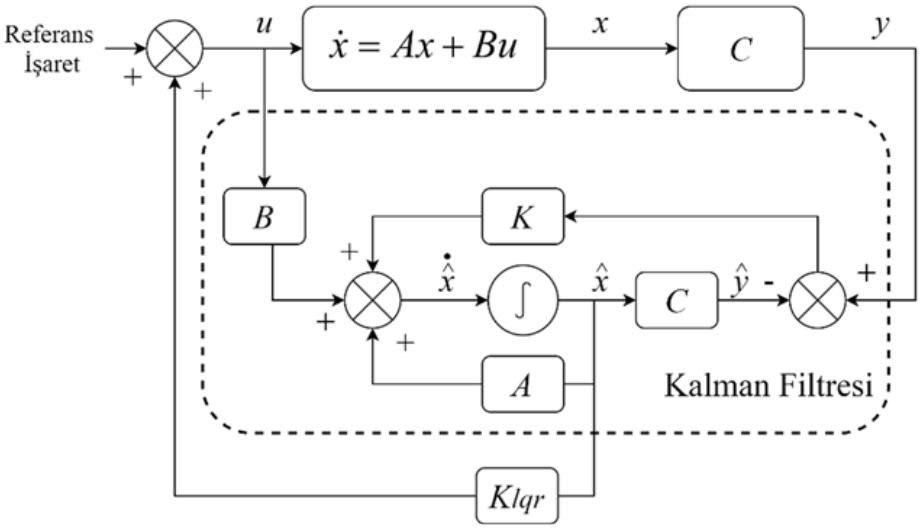
- $\hat{X}$Sürekli zamanlı durum tahmini,
- P tahmin hata kovaryans matrisi,
- K Kalman kazanç matrisi,
- R ölçüm hatası kovaryans matrisi,
- Q işlem hatası kovaryans matrisi.

$$0 = AP + PA^T + FQF^T - PC^T RCP \tag{3.12}$$

Sürekli zamanlı Kalman filtre denklemleri Riccati denklemlerine benzer bir yapıya sahiptir. Özellikle, Denklem (3.12)'de gösterildiği gibi, Riccati denklemlerinde olduğu gibi A ve B matrislerinin yerine A^T ve B^T matrisleri kullanılarak sürekli zamanlı Kalman filtresi çözümlenmektedir.

Denklem (3.12), bir Riccati denklemi gibidir ve genellikle bir lineer zamanla değişmeyen kontrol sistemlerinde sürekli zamanlı kalman filtresi tasarımında kullanılır. Bu denklem, durum tahmin hata kovaryans matrisini P ve diğer sistem matrislerini içerir. Riccati denklemleri, belirli bir kontrol problemi için en uygun kontrol kazancını hesaplamak için kullanılır.

Kalman filtresi, tahmin edilen durumları LQR kontrol bloğunda kullanarak, zamanla değişmeyen kapalı çevrim kontrol sistemlerinde optimal kontrol sağlar. Sürekli zamanlı Kalman filtre denklemleri kullanılarak elde edilen kapalı çevrim kontrol blok şeması aşağıda Şekil 3.22'de verilmiştir. Bu şemada, en uygun geribesleme kazanç matrisi LQR kontrolü ile hesaplanmaktadır. (Mehmet Latif 2021). Kalman filtresi, sistem durumlarını tahmin ederken LQR kontrolü, bu tahmin edilen durumları gerçek zamanlı olarak kullanarak sistemi kontrol eder. Bu şekilde, sistemdeki belirsizlikler ve gürültüler minimize edilerek, istenen hedeflere daha yakın ve optimal bir kontrol elde edilir.

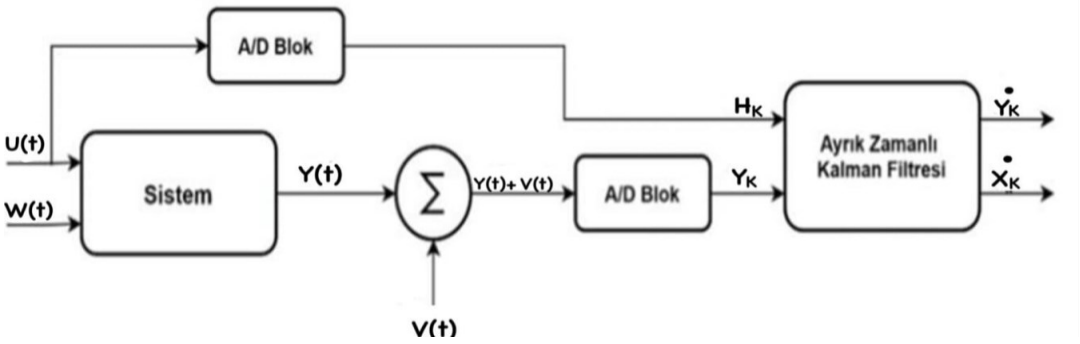


Şekil 3.22 Sürekli kalman filtresi ile tasarlanan kapalı çevrim kontrol şeması(Latif Levent,2021)

3.6.4. Ayrık Zamanlı Kalman Filtresi

Ayrık zamanlı Kalman filtresi, kontrol girişi işareti U_k ve gürültülü sistem çıkışı işareti Y_k verilerini kullanarak sistemin hedeflenen çıkışını ve durumlarını tahmin eden bir yöntemdir. Bu yöntemin temel blok diyagramı Şekil 3.23'te sunulmuştur. Kalman filtresi, sistemin içsel durumlarını ve çıkışını gözlemleyerek, giriş verisiyle birlikte sistemin gelecekteki durumlarını tahmin etmeye çalışır. Bu tahminler, sistemin belirsizliklerini ve dış etkileri hesaba katarak, istenen çıkışın ve sistem durumlarının

doğru bir şekilde tahmin edilmesini sağlar. Bu sayede, sistem kontrolü ve tahmini istenen hedeflere daha uygun ve doğru bir şekilde gerçekleştirilebilir.



Şekil 3.23. Ayrık zamanlı filtresi ile durum tahmini blok diyagramı

Ayrık kalman filtre iki temel aşamadan oluşan bir yapıya sahiptir. Bunlardan birinci tahmin-zaman güncelleme ikincisi ise ölçüm-zaman güncellemedir.

Kalman filtresi, geri besleme, yani sistemin gürültülü ölçüm değerlerinin kullanarak, bir sistemin durumlarını tahmin eder. Kalman filtresi denklemleri iki grupta; zaman güncelleme denklemleri ve ölçüm güncelleme denklemleri olarak toplanabilir. Zaman güncelleme denklemleri, bir sonraki durumun değeri tahmin etmek için, güncel durumu ve hata kovaryansını kullanır. Zaman güncelleme denklemleri, tahmin edici denklemler olarak adlandırılır. Ölçüm güncelleme denklemleri ise, geliştirilmiş sonraki tahmini elde etmede kullanılır. Ölçüm güncelleme denkleri ise doğrulayıcı, düzeltici denklemler olarak adlandırılır.

Ayrıca Kapalı çevrim geribeslemeli kontrol sistemlerinde, geribesleme bloğunda ihtiyaç duyulan durumların tahmini ise ölçüm güncelleme denklemleri ile sağlanmaktadır.

Tahmin edici güncelleme denklemleri:

$$\hat{X}_k^- = A\hat{X}_{k-1} + Bu_k \tag{3.13}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.14}$$

Denklem (4.13) ile düzeltme aşamasında kullanılacak durumların önceki tahmin değerleri bulunmaktadır. (K) değişkeni zamanı temsil etmektedir. (X) değişkeni durum matrisi olarak adlandırılır ve içeriği sistemde gözlemlenmek istenen konum veya hız gibi herhangi bir değer olabilir. (u) değişkeni kontrol değişkeni matrisi olarak adlandırılır.

(A) değişkeni geçiş matrisi olarak adlandırılır ve (B) değişkeni kontrol girdi matrisi olarak adlandırılır. A ve B matrisleri "1" olarak alınır ve kontrol sinyalinin olmadığı varsayılır. Fakat sistemde durum matrisleri ve kontrol sinyali varsa bunlarda işleme dâhil edilmelidir. (3.13) İfade herhangi bir durum tahmin değeri, bir önceki durumun tahmin değeri ile kontrol sinyalinin lineer bir kombinasyonudur.

Başlangıç koşulları bilinmiyorsa, genellikle k=1 için $\hat{X}_{k-1}$yani(X_0) değeri sıfır olarak alınabilir. Başlangıç durum tahmininin sıfır alınması, sistemin çalışmasını engellemez; ancak başlangıç koşullarında sistemin durumu biliniyorsa, bu bilgilerin kullanılması tahminlerin gerçek değerlere daha hızlı yakınsamasını sağlar.

Başlangıç durumunun sıfır alınması, sistemin ilk tahmininin belirsizliği azaltır ve filtreleme sürecinin başlangıcını belirler. Ancak, eğer başlangıç durumları biliniyorsa (örneğin, sistemin başlangıç durumu ölçülmüş veya bilinmektedir), bu gerçek değerlerin kullanılması, Kalman filtresinin daha hızlı ve doğru bir şekilde çalışmasını sağlar. Bu sayede, sistemin başlangıç koşullarına daha iyi bir şekilde uyum sağlanır ve tahminler, gerçek durum değerlerine daha hızlı yaklaşır.

Denklem (3.14) de P değişkeni hata kovaryans matrisi olarak adlandırılır ve her bir değişkenin hata oranını tutar. Q değişkeni işlem gürültü kovaryans matrisi olarak adlandırılır.

A matrisi ile önceki duruma ait P matrisi çarpılarak güncel duruma ait tahmini bir hata oranı üretilir. Bu hata oranına öngörülemeyen çevresel gürültüler de varyansı alınmış olarak eklenir.

Eğer hata kovaryansının başlangıç koşulu (k = 1 için P_{k-1}=Po) bilinmiyorsa sıfırdan farklı bir değer alınabilir. Genellikle 1 olarak alınmaktadır.

Bu işlemin sonunda AKF'nin tahmin aşaması biter ve güncelleme aşamasına geçilerek ölçümler de hesaba katılır.

Düzeltici güncelleme denklemleri:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (3.15)$$

$$\widehat{X_k} = \hat{X}_k^- + K_k (z_k - H\hat{x}_k^-) \quad (3.16)$$

$$P_k = (1 - K_k H) P_k^- \quad (3.17)$$

Kalman kazancı hesaplandıktan sonra sonraki durum değeri z_k değeri ölçülerek, güncellenmektedir. Sonraki adımda ise hata kovaryansı elde edilir. Bu, tahmin edici ve düzeltici döngü her tekrarlandığında, yani bir önceki değerler kullanılarak, sonraki değerler (tahminler) elde edilmektedir.

Ölçüm güncelleme basamağında tasarıma ait belirlenmesi gereken en önemli bilinmeyen kalman kazancı (K) ve 0 ile 1 arasında değer almaktadır.
Kalman Kazancı'nın değeri ölçüm ya da kestirim değerlerinden hangisine daha çok güvenileceğini gösterdiği için, kalman filtrede kilit rol oynar. K değerinin sıfıra yaklaşması kestirimlerin daha doğru olduğu, bire yaklaşması ise ölçümlerin daha doğru olduğu anlamına gelmektedir. Kalman kazancı hesabını, tahmini z_k ile güncellemesi ve hata kovaryansını güncellemesi ölçüm güncelleme denklemleri ile gerçekleştirilmektedir. Kalman kazancı filtre performansını daha iyi bir hale getirmek için değiştirilebilir ve yapılan ölçümü göz önüne alarak tahmin edilecek durumların ne kadar değişmesi gerektiğine belirmektedir.

Filtre tasarımı sırasında, Kalman kazancı üzerinde en fazla etkiye sahip değişken hata kovaryansıdır (P_k). Hata kovaryansının büyük bir değere sahip olması durumunda, sistem durumlarının tahminleri büyük değişiklikler gösterebilir. Bu nedenle, elde edilen yeni ölçümlerle tahmin güncellemesi yapılması gereklidir. Hata kovaryansı ile Kalman kazancı arasında doğru orantı vardır; yani hata kovaryansı arttığında Kalman kazancı da artar. Eğer hata kovaryansı küçük ise, durum tahminlerinin çok fazla değişmediği gözlemlenir. Bu durumda, tahmin değerini sürekli olarak değiştirmek gereksiz olabilir. Hata kovaryansının azalması durumunda ise Kalman kazancı da azalır. Bu işlemin ardından, güncellenen hata kovaryansı matrisi geri besleme ile Denklem (3.14)'te gönderilir. Bu süreç, Kalman filtresinin güncellenmiş hata kovaryansı ve geri bildirimle sistemin durumunu daha doğru bir şekilde tahmin etmesini sağlar.

Kalman kazancı üzerinde etkisi olan bir diğer faktör ise ölçüm gürültüsü matrisidir (R). Bu matris, ölçüm sırasında ortaya çıkan belirsizlikleri ve tutarsızlıkları belirtmek için kullanılır. Ölçüm gürültüsü matrisinin artması, ölçüm değerlerinin doğru olmadığının bir işaretidir. Ölçüm gürültüsü ve Kalman kazancı arasında ters orantılı bir ilişki vardır. Yani ölçüm gürültüsü arttıkça, Kalman kazancı azalır.

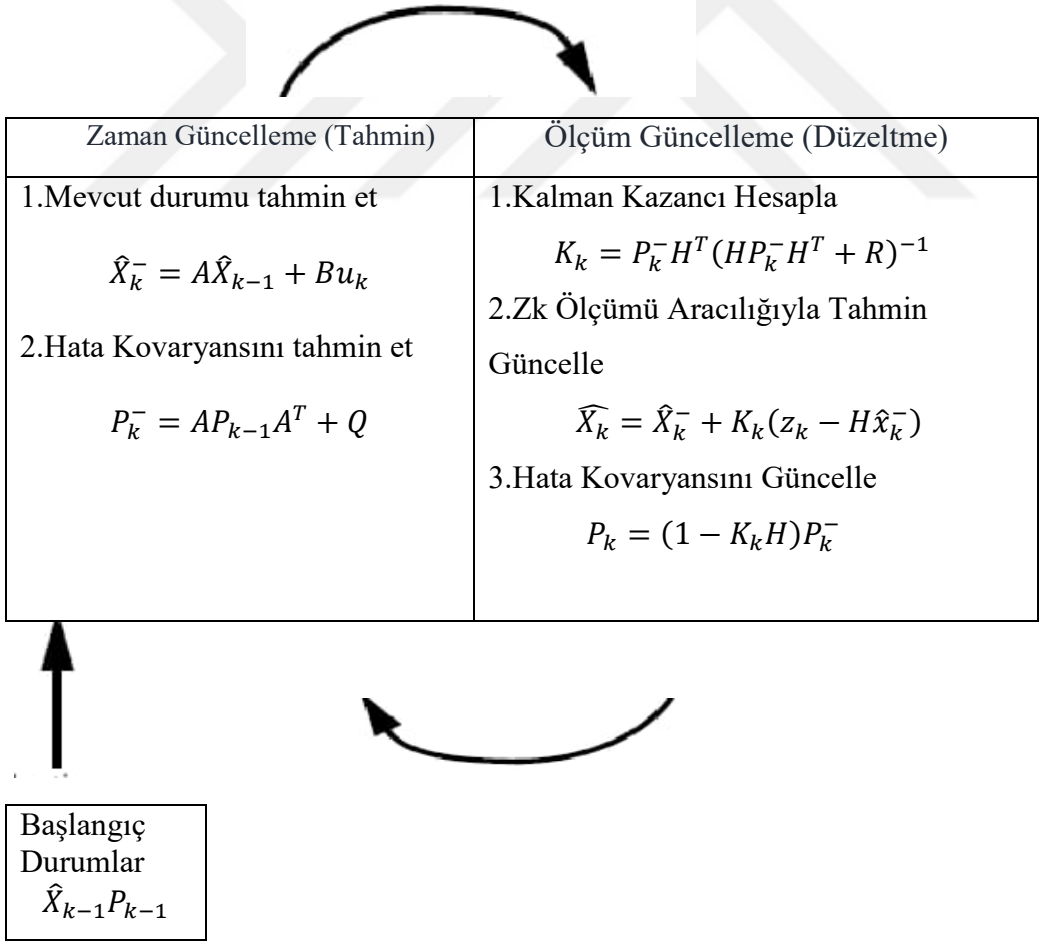
H matrisi adaptasyon matrisidir ve bazı durumlarda P matrisinin boyutu ile K matrisinin boyutu uyuşmayabilir. Bu gibi durumlarda, H matrisi, P matrisini K matrisine dönüştürecek şekilde seçilir ve Kalman filtresinin sorunsuz çalışmasını sağlar. Aksi durumlarda H matrisi birim matristir.

Denklem (3.16), durum tahminlerinin elde edildiği denklemdir. Bu denklemde, durumların tahmin edilmesi için Denklem (3.15)'te bulunan Kalman kazancı (K_k) ve k alt indisindeki ölçüm değerleri kullanılır. Bu işlem sonucunda elde edilen durum tahminleri, zaman güncelleme basamağında girdi olarak kullanılır.

Denklem (3.17)'de ise hata kovaryansının bir önceki tahmin değeri kullanılarak zaman güncelleme basamağında kullanılacak olan güncel hata kovaryansı bulunur. Bu denklem, Kalman filtresinin güncellenmiş hata kovaryansını hesaplamasına yardımcı olur ve filtreleme işleminin doğru bir şekilde gerçekleşmesini sağlar.

Yukarıda anlatıldığı gibi, ölçüm güncelleme denklemleri ve zaman güncelleme denklemleri gibi farklı iki denklem kullanılarak ayrık zamanlı kalman filtre tasarımları gerçekleştirilmektedir. Mevcut durumların ve hata değerlerinin kovaryans tahmini, zaman güncelleme denklemleri ile gerçekleştirilmektedir.

Kapalı çevrim geribeslemeli kontrol sistemlerinde, geribesleme bloğunda ihtiyaç duyulan durumların tahmini ise ölçüm güncelleme denklemleri ile sağlanmaktadır. Ölçüm güncelleme denklemleri ve zaman güncelleme denklemleri sırasıyla ‘düzeltme denklemleri’ ve ‘tahmin denklemleri’ olarak da tanımlanmaktadır. Bu doğrultuda çalışan ayrık zamanlı kalman filtre çalışma döngüsü Şekil 3.24'te verilmiştir. Her bir K için hesaplanan çıktı, (K+1) sonraki adım için girdi olacaktır



Şekil3.24. Ayrık zamanlı kalman filtre çalışma döngüsü

3.6.5. Kalman Filtresi Kavramsal Uygulama Örnekleri

Kalman Filtreleri ile ilgili aşağıda bir dolaylı ölçüm uygulaması verilmektedir. Dolaylı ölçümlerden gerçek ölçüm değerlerine ulaşmak için Kalman Filtresinden yararlanılmıştır. Örnekte APOLLO uzay aracı uçuşunda Kalman Filtresi ile ölçüm yapılmıştır.

Problem-1 Tanım:

Uzay gemisinin yakıt olarak sıvı hidrojen kullanılmakta ve motorun iç ısısı 5500(F °) Fahrenhayt seviyelerine kadar çıkmaktadır. Bu 5500 °F çıkan sıcaklık mekanik parçalara zarar verebilmekte ve sonuçta roket zarar görebilir.

Doğal olarak APOLLO roketinin bütün bu sıcaklığının motor iç ısısının ölçülmesi gerektiği için mekanik aksamlarda herhangi bir sıkıntı olup olmadığını görebilmeleri gerekiyor, yani APOLLO roketinin ateşleyicilerinin iç ısısının ölçülmesi gerekmektedir.

Bu Ölçüm için dayanıklı algılayıcılar yok. Bu sebeple, sıcaklık dışarıdan ölçülmelidir. Dışarıdan ölçülen sıcaklık üzerinden iç sıcaklık tahmin edilmelidir.Asıl değeri ölçülmüyor ama asıl değere Endirek ölçümler yapabilir. Bu endirek ölçümler üzerine nasıl değeri tahminleme çalıştığında kalman filtresi bir çözüm olarak öngörmektedir. İlk olarak matematiksel modeli ortaya koyup Matematiksel modeli ortaya koyduktan sonra bu endirek ölçümlerden asıl ölümlere gitmeye çalışır.

Problem-2 Tanım:

Bir aracın gerçek konumunun ölçümü ve konum takibinin gerektiği bir problemi ele alalım. Bilindiği gibi konum ölçümü için;

1. GPS kullanılmaktadır, ama GPS'in sıkıntıları var: Güncelleme frekansı 10 Hz'lere çıksa bile genelde 1 Hz civarında düşünebilir. Saniyede bir bin güncelleme hızı vardır yani GPS güncelleme hızı (frekansı) diğer algılayıcılara (AÖB ataletsel ölçüm birimi) göre düşüktür. Ayrıca GPS dış ortamlarda çalışırken kapalı ortamlarda çalışmamaktır.

2. Bunun dışında konum takibi için ataletsel ölçü birimi jiroskop, ivmeölçer ve manyetometre ölçü birimleri kullanılmaktadır. Bunların GPS'e göre daha yüksek frekansta sonuç elde etse de çok basit bir şekilde ölçtüğünden ivmeden yolu hesaplarken, çift integral da aldığı için doğal olarak hata negatif olarak artmaktadır $(x = \iint at * dt)$.

3. Konum takibi için kara araçlarında ayrıca odometre kullanılmaktadır. Odometre yola bağlı olarak yoldaki çukurlar ve setlerin fiziksel özelliklerine bağlı olarak hatalı ölçüm yapılabilir ya da lastiğin havasının durumuna bağlı olarak yine fazla ya da eksik ölçüm yapabilir.

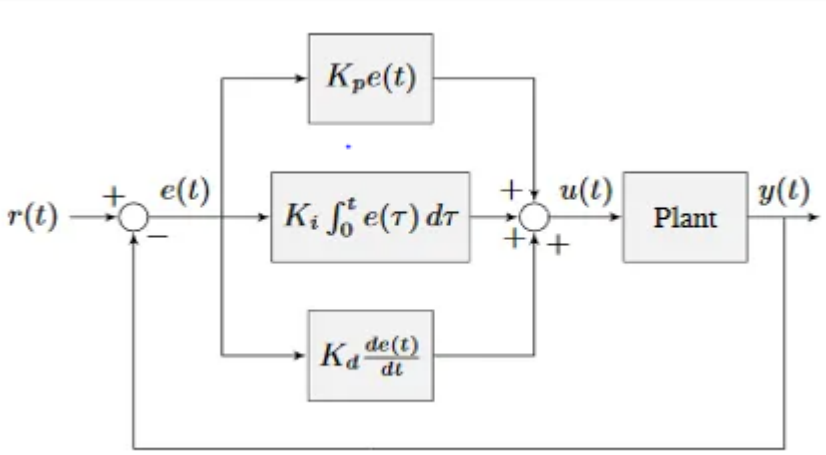
Sonuç olarak yukarıdaki gibi farklı tipte sensörler (GPS jiroskop, ivmeölçer, manyetometre ve odometre) ile konum ölçümü yapılabileceği görülmektedir. Ama üç sensöründe her birinin kendine özgü farklı belirsizlikleri vardır. Yani herhangi biriyle mükemmel ölçüm yapılamadığı görülmektedir. Bu durumda, bu belirsizlikleri daha aza indirmek için bu üç farklı tipte sensör aynı anda kullanarak belirsizliği azaltacağı yönünde bir çalışma yapılabilir. Yani GPS, odometre ve manyometre üçünün bir arada kullanılarak Kalman filtresi ile sensör füzyon yapılarak daha doğru bir sonuç elde edilebilir.

3.7. PID Denetleyiciler

PID denetleyici kendi ismini de oluşturan üç temel kontrol biriminin birleşiminden oluşmaktadır: Oransal kontrol (P), İntegral kontrol (I) ve Türevsel kontrol (D). Bu üç kontrol birimi, kontrol edilen sistemin durumunu izleyen hata sinyaline göre kontrol sinyalini hesaplamaktadır.

PID kontrol yöntemi, sistemin sürekli kontrol edilmesini gerektiren robotik, endüstri ve havacılık gibi alanlarda sıkça kullanılan bir geri beslemeli kontrol yöntemidir. PID kontrol sistemlerinin yapı itibari ile basit olmaları, her türlü sisteme kolay entegre edilebilmeleri bu kontrol sisteminin popülerliğini arttırmaktadır

PID denetleyici, sistemin hedef değeri ile mevcut durumu arasındaki farkı (hatayı) hesaplar ve bu hatayı oransal, integral ve türevsel kontrol birimlerinde işleyerek kontrol sinyalini üretir. PID denetleyici sürekli olarak referans değer ile ölçülen değer arasındaki hatayı alır ve PID parametrelerini kullanarak oluşan hatayı en aza indirmeyi amaçlamaktadır. Bu sayede sistemin istenilen referans değerde kalması sağlanmış olmaktadır. Aşağıda Şekil 3.25'te temel kapalı çevrim PID Kontrol sistemi blok diyagramı verilmektedir.



Şekil 3.25. PID Kontrol Sistemi

PID denetleyici kontrol işareti, aşağıda denklem (3.18) ve (3.19) sırasıyla zaman dönemi ve frekans (Laplas) dönemi için verilmektedir.

$$u(t) = K_p e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt} \tag{3.18}$$

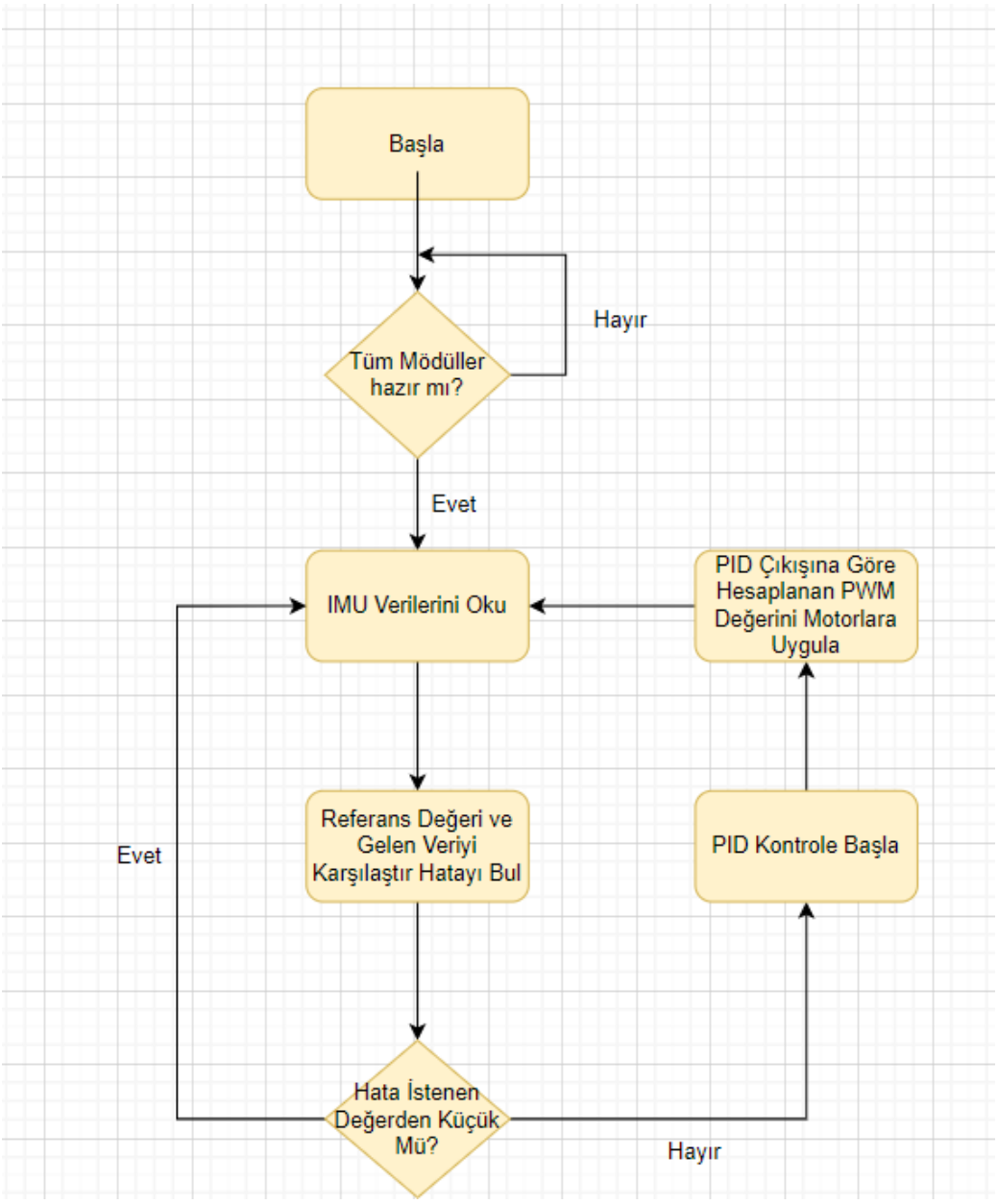
$$U(s) = K_p + \frac{K_i}{s} + K_d s \tag{3.19}$$

Denklemlerde görülen K_p: oransal kazancı, K_i: integral kazancı, K_d: türev kazancını ifade etmektedir. $e(t)$ veya E(s): referans ile gerçek değer arasındaki hatayı, $u(t)$veya U(s): Kontrol sinyalini, s sembolü ve 1/s sembolü ise sırasıyla frekans düzleminde türev ve integral işlemlerini ifade etmektedir.

PID denetleyicide her bir bileşeninin karakteristikleri vardır: Oransal denetleyici, yükselme zamanını kısaltır ancak sürekli hal hatasını tamamen ortadan kaldırmaz. İntegral denetleyici, kararlı hal hatasını ortadan kaldırır ancak geçici tepkinin daha kötü olmasına neden olabilir. Türevsel denetleyici, sistemin kararlılığını artırır, aşımı azaltır ve geçici tepkiyi iyileştirir. Bu bileşenlerin etkileri, kapalı çevrimli sistemlerde önemlidir ve her birinin sistem performansına farklı bir katkısı vardır. Bu etkiler, genellikle bir PID denetleyicisinin ayarlanmasında dengelenmelidir. Bu dengeyi sağlamak için, PID kontrolörünün her bir bileşeninin katsayıları dikkatlice ayarlanmalıdır. Bu ayarlama işlemi, deneysel yaklaşımlar veya sistem modellemesi ile hesaplamalarla yapılabilir. Ya da en kaba biçimde deneme-yanılma yöntemiyle yapılabilir.

3.7.1 Ayrık Zamanlı PID Denetleyiciler

Sistemin denge kontrolü için kullanılan hata değeri, referans değer ile IMU'dan okunan değer arasındaki farktan oluşmaktadır. PID kontrol sistemi, bu hata değerini ilgili algoritmalar kullanarak referans değere getirmeyi amaçlamaktadır. İki motorlu sistemde PİD kontrolünü sağlamak için aracın roll eksen değerine karşılık gelen PWM değerleri kaydedilerek sistemin matematiksel modellemesi çıkarılmalıdır. PID kontrol algoritması, şekil 3.26'de verilmiştir.



Şekil 3.26. PID Kontrol algoritması

4. UÇUÇ KONTROL KARTI TASARIMI

Bu bölümde, çok rotorlu bir İHA'ya uçuş kontrol fonksiyonu sağlayacak uçuş kontrol kartı (UKK) tasarımında kullanılan materyaller ve yöntemlerden bahsedilecektir. Donanım tasarımında kullanılan tasarım programları, çizim teknikleri ve kullanılan elektronik komponentler ile ilgili bilgiler verilecektir.

Bu çalışmada, hazır PCB veya geliştirme kartı kullanmak yerine özgün ve yerli bir PCB tasarımı ve üretimi yapılmıştır. Baskı devre kartları ve elektronik devre şema tasarımı için açık kaynaklı EDA programı olan KiCad kullanılmıştır. KiCad, elektronik devre şemalarını çizmek ve PCB tasarımlarını oluşturmak için yaygın olarak kullanılan bir programdır.

UKK tasarımında kullanılan bazı önemli elektronik komponentler aşağıdaki gibi sıralanmıştır:

- Mikrodenetleyici: Uçuş kontrol algoritmalarını çalıştıran işlemci birimidir.
- IMU: İHA'nın konumu, hızı ve yönelimi hakkında bilgi sağlayan sensör birimidir.
- Barometre: İHA'nın yüksekliğini ölçen sensördür.
- ESC: Motorların hızını kontrol eden elektronik devredir.
- MOSFET: Motorlara güç sağlayan elektronik anahtarlarıdır.
- Diyotlar, kondansatörler ve dirençler: Devredeki diğer komponentleri desteklemek için kullanılan pasif komponentlerdir.

4.1. STM32F401 Mikrodenetleyicisi

Gömülü sistemlerde ARM mimarisi, genellikle yüksek performans, düşük güç tüketimi ve geniş bir ekosistem sunması nedeniyle tercih edilmektedir. Bu nedenle, kart üzerinde ARM Cortex-M4 tabanlı bir mikrodenetleyici olan STM32F401 kullanılması uygun görülmüştür. STM32F401 mikrodenetleyicisi, harici bir osilatör kullanarak 84 MHz'e kadar bir çalışma frekansına sahiptir ve bu da yüksek performanslı uygulamalar için idealdir.

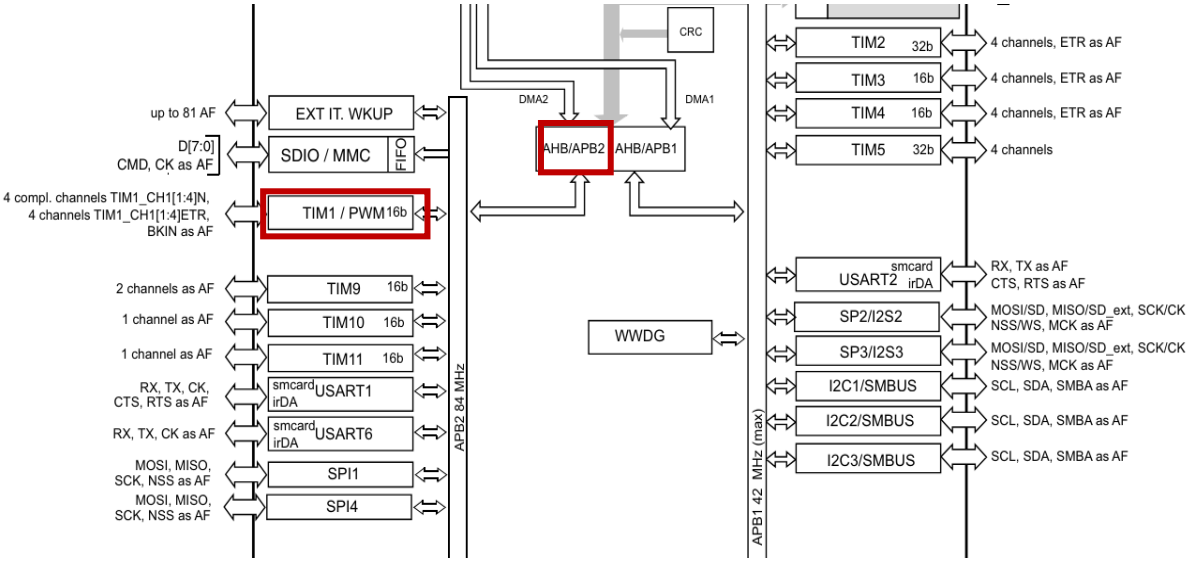
STM32F401 mikrodenetleyicisi birçok farklı iletişim protokolünü destekler. Bunlar arasında I2C, USART, UART, SDIO ve SPI bulunmaktadır. Bu protokoller, farklı cihazlar arasında iletişim kurmak için kullanılır ve çeşitli sensörler, ekranlar ve diğer harici bileşenlerle entegrasyon sağlamaktadır.

Aynı anda birden fazla işlemi gerçekleştirebilmemizi sağlayacak olan bir gerçek zamanlı işletim sistemi (RTOS) desteğine sahiptir. RTOS desteği, karmaşık sistemlerin daha iyi organize edilmesine ve daha güvenilir çalışmasına yardımcı olmaktadır. Kullanılan mikrodenetleyici görseli Şekil 4.1'de verilmektedir.

Şekil 4.1. STM32F401 Mikrodenetleyicisi

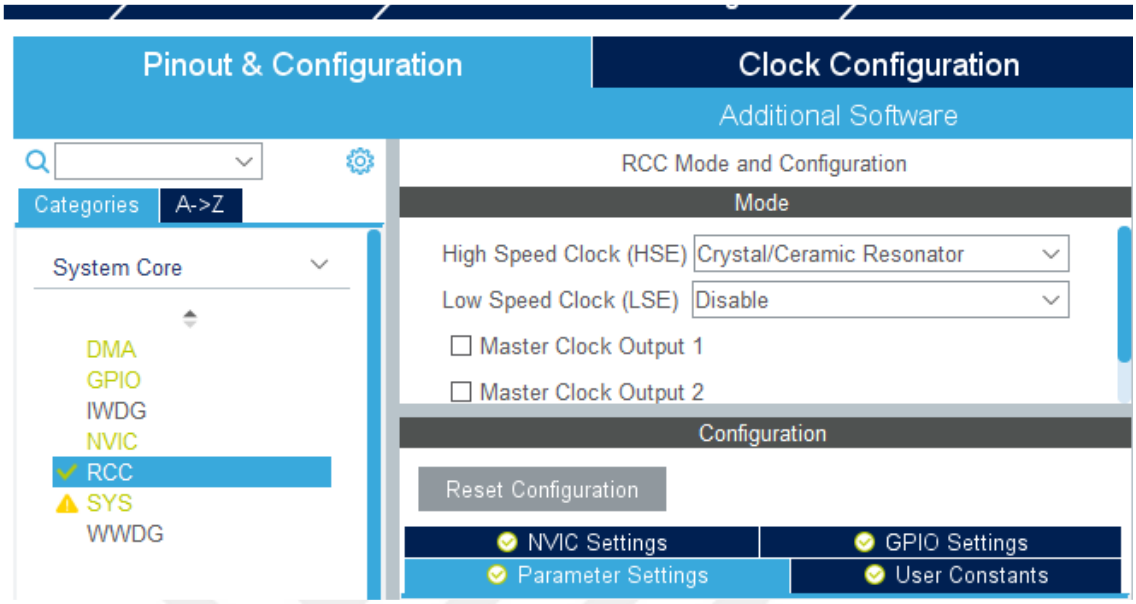
4.2. Zamanlayıcı (Timer) Ayarları

Şekil 4.2'de görüldüğü gibi TIM1 biriminin saat frekansı olarak beslendiği hat PLKC2 APB2 hattıdır. Genel saat frekansı maksimum çalışma frekansı olan 84 MHz olarak tercih edildi. TIM1 birimini besleyen hatta 84 MHz değerine sahiptir.



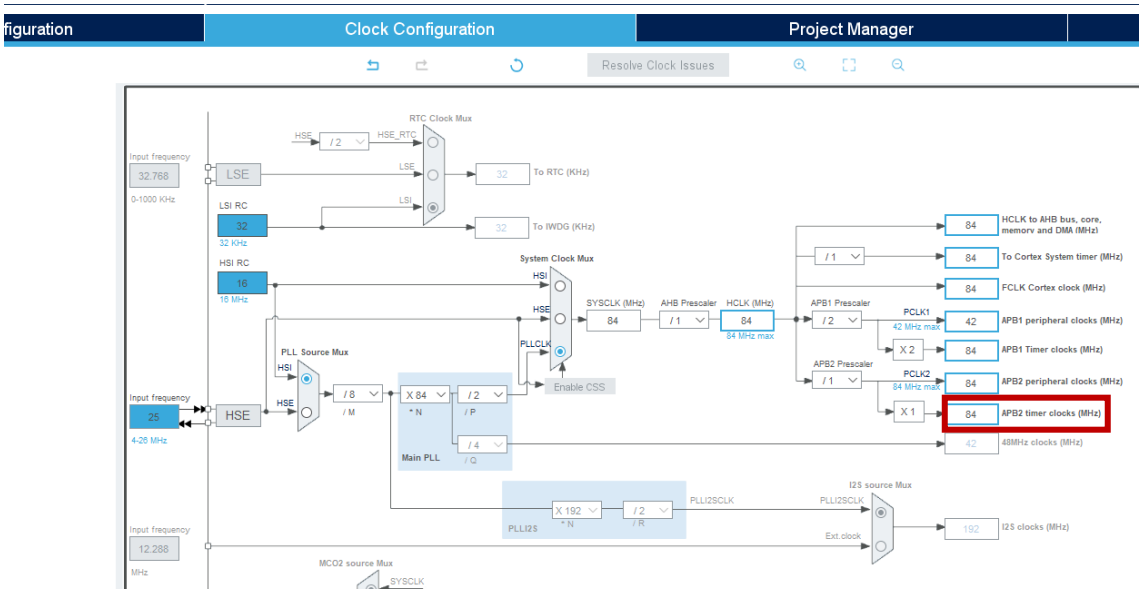
Şekil 4.2. STM32F401 Performans hattı blok şeması

STM32CubeMX programında devre için belirli bir frekansta elektrik sinyali oluşturmak için RCC ayarlarından Şekil 4.3'te harici osilatör (HSE) aktif edilir.



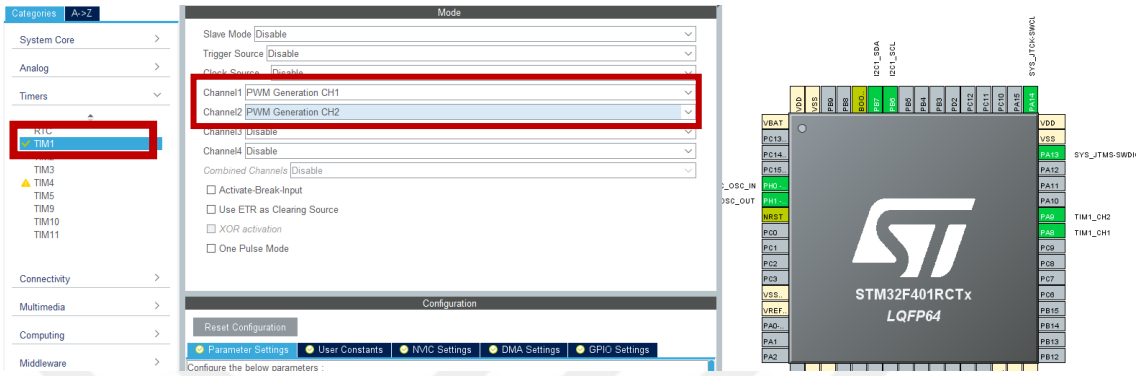
Şekil 4.3. Harici Osilatör Konfigürasyon ayarları

Saat konfigürasyon kısmında, TIM1 birimini besleyen APB2 Zamanlayıcı Saat hattı 84 MHz değere sahiptir. Şekil 4.4'te gösterilmektedir.



Şekil 4.4. Saat konfigürasyon ayarları

STM32CUBEMX programında Pinout configuration kısmından Timer kısmındaki Timer1 seçip ve 2 farklı kanal için çalışma modu PWM Generation seçilir. Şekil 4.5'te gösterilmiştir. PA8 ve PA9 pinleri TIM1_CH1 ve TIM1_CH2 olarak ayarlanmaktadır.

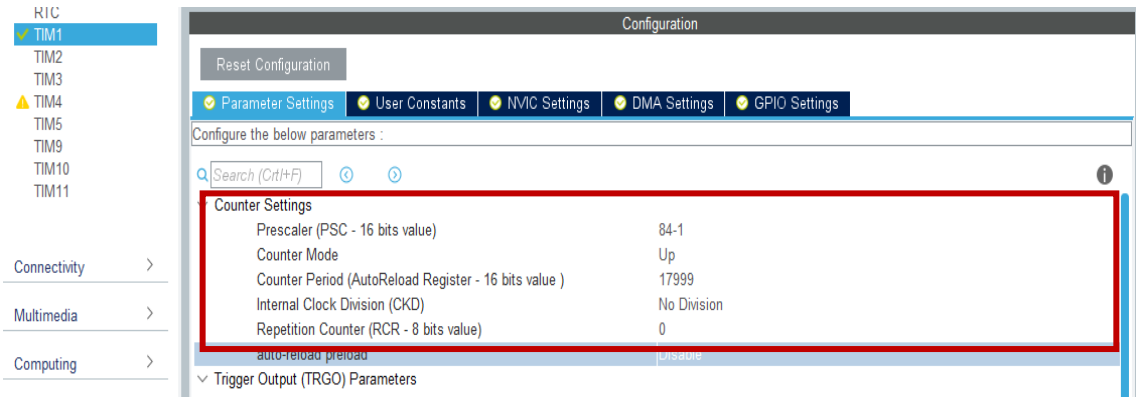


Şekil 4.5. Konfigürasyon ayarları

Zamanlayıcının 1MHz değerinde çalışmasını istiyoruz. Zamanlayıcının her sayımında geçecek süre böylece 1 us oluyor (1/1Mhz = 1us).

Zamanlayıcının 1MHz ile çalışmasını sağlamak için, zamanlayıcıyı besleyen frekans hattını 84 değerine bölmemiz gerekiyor (84Mhz/84 = 1 Mhz)

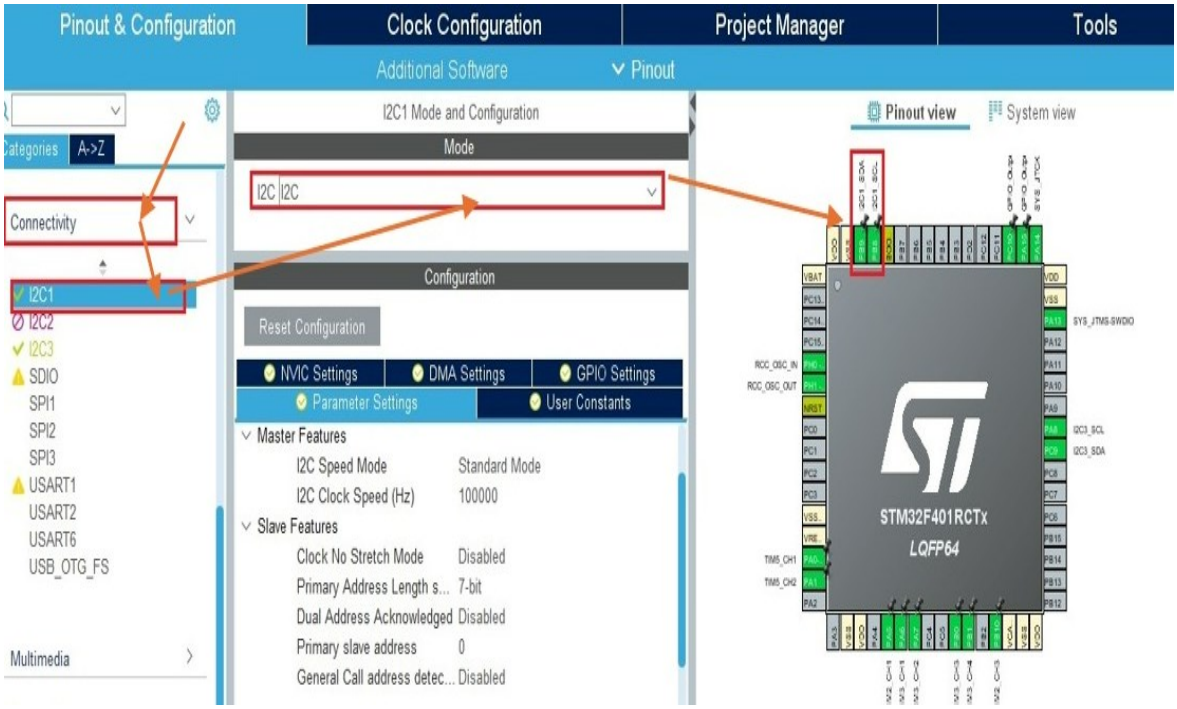
Bölme işlemini, zamanlayıcının PSC Registeri ile gerçekleştiriyoruz. Parametre settings kısmında prescaler değeri 84-1 değeri girilir. Counter Periyod ise zamanlayıcının maksimum sayacağı miktardır.16 Bit bir zamanlayıcı olduğu için sayacağı maksimum değer 65536'dır. Yapılan işlemler Şekil 4.6'da gösterilmiştir.



Şekil 4.6. Zamanlayıcı Konfigürasyon Ayarları

4.3. MPU6050 IMU Sensör Kullanımı

Tasarlanan özgün UKK için yaygın kullanılan MPU6050 IMU sensör tercih edilmiştir. Sensör ile denetleyici iletişimi için I2C haberleşme protokolü kullanılacaktır ve ilgili pinlere atamalar Şekil 4.7'de görüldüğü gibi yapılmıştır.



Şekil 4.7. IMU Konfigürasyon Ayarları

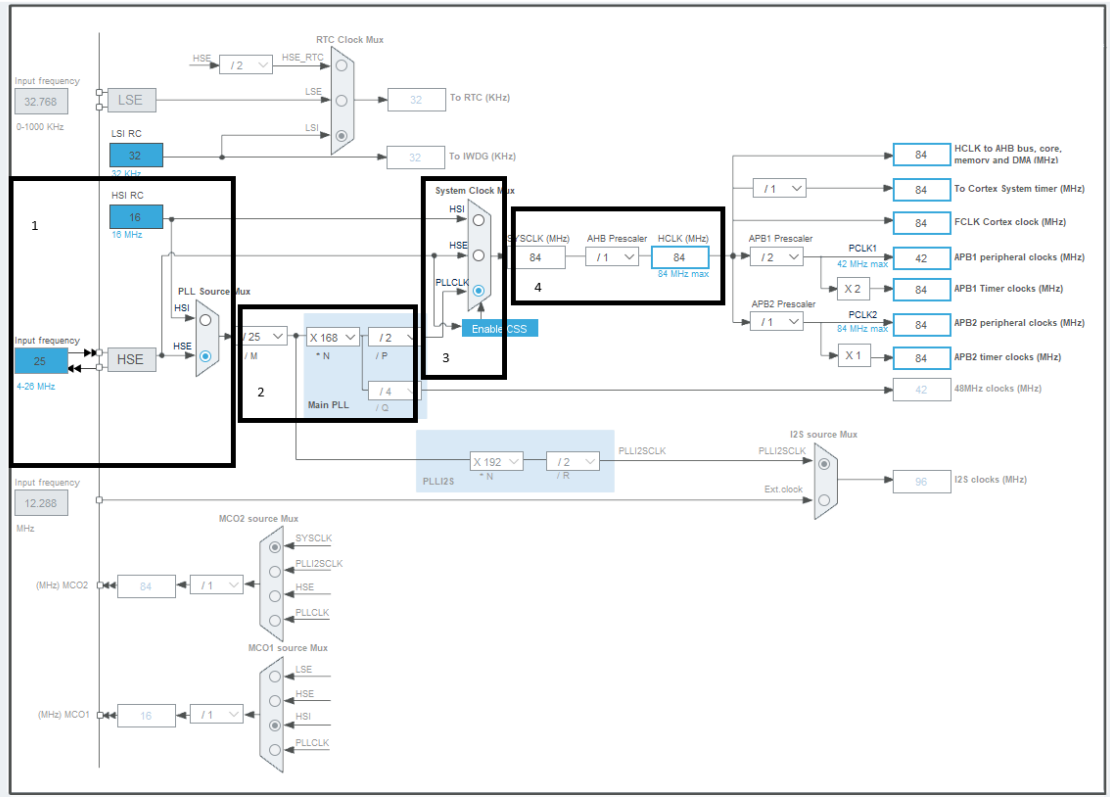
STM32 kartlarında Saat konfigürasyonu 4 bölümden oluşmaktadır. Şekil 4.8'de görüldüğü gibi 4 farklı blok ve bu bloklar aşağıda tek tek incelenecektir.

1. Bu kısımda kullanılacak osilatörü seçmemiz gerekmektedir. HSI, yüksek hızlı dahili osilatördür, HSE ise yüksek hızlı harici osilatördür. HSE osilatörler, HSI osilatörlere göre daha yüksek doğruluk oranına sahip frekanslar sağlamaktadır. Bu nedenle, doğruluk ve performans açısından daha avantajlı olduğu için HSE kullanılacaktır.

2. PLL, bir saat frekansının katları olan çok daha yüksek saat frekanslarını elde etmek için kullanılan bir mekanizmadır. PLL katsayıları ayarlanarak işlemcinin

frekansı, 84 katına kadar yükseltilebilir. Bu sayede, işlemcinin performansı artırılarak daha hızlı ve verimli çalışması sağlanır.

3. Burada ise bölümde Sistem saat frekans kaynağının seçildiği kısımdır.



Şekil 4.8. Saat Konfigürasyon Ayarları

4. Bu kısımda AHB (HCLK) ve APB (PCLK) bus hatlarının saatlerinin ayarlanması işlemi yapılır. Dikkat edilmesi gereken nokta AHB hattının saat sinyali belli bir bölme oranıyla sistem saat kaynağından alınırken, APB hattının sinyali AHB hattı üzerinden belli bir bölme oranıyla alınır.

4.4. Uçuş Kontrol Kartı Devre Şematiği

İnsansız hava araçlarının kontrolünü sağlamak amacıyla tasarlamış olduğumuz IHA kontrol kartının tasarımı, KiCad üzerinde gerçekleştirilmiştir. Tasarlanan kontrol kartının bileşenlerini ve bağlantılarını Şekil 4.12'de elektronik devre şematiği üzerinde gösterilmiştir. İHA için KiCad programı üzerinde gerçekleştirilen kontrol kartının PCB

tasarımı, bileşen yerleşimi, güç yönetimi, sinyal yolları ve termal yönetim gibi kritik faktörlere odaklanarak Şekil 4.13'te gerçekleştirilmiştir.

Tasarlanan uçuş kontrol kartı ön ve arka yüzü Şekil 4.14'te gösterilmiştir üzerinde UKK tasarımında kullanılan tüm önemli elektronik komponentler bulunmaktadır.

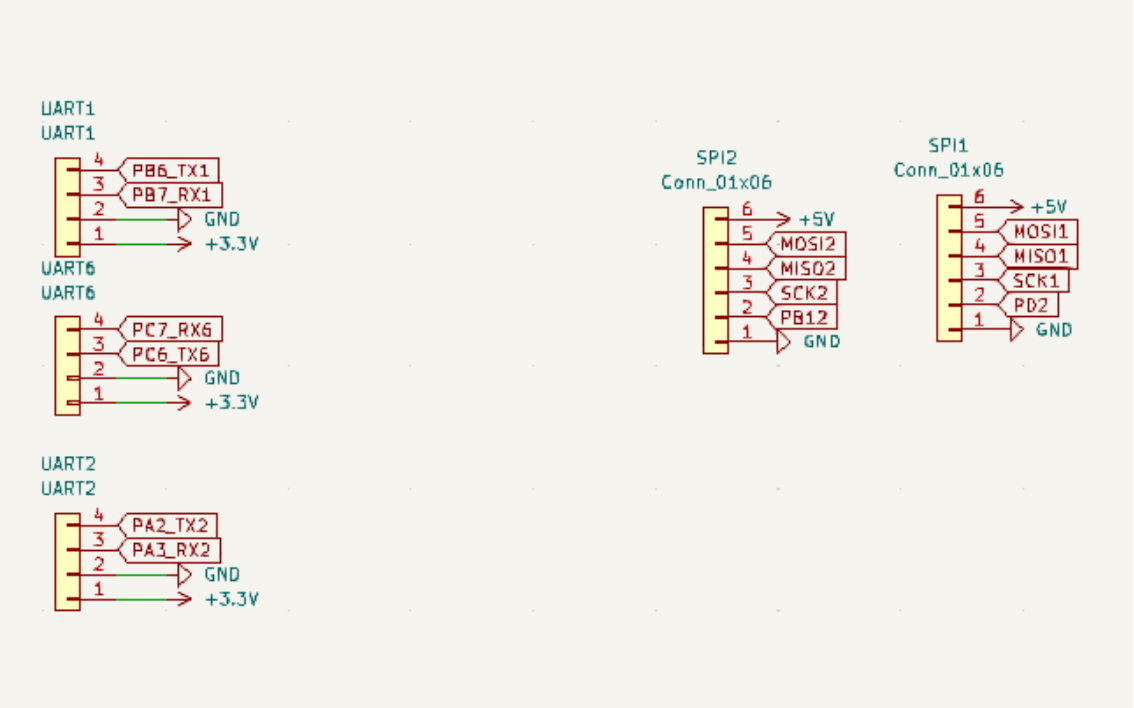
Kontrol kartının üzerinde bulunan güç devresi, ayarlanmış güç pinleri veya USB portun üzerinden alınan 5V gelirimi AMS1117 regülatörü sayesinde 3.3V'a dönüştürerek mikrodenetleyici ve diğer bileşenlerin çalışması için uygun gerilim sağlanmıştır.

Kullanılan regülatörün giriş ve çıkış pinlerinde 2 adet 11uF kondansatör yerleştirilmiştir. Bu kondansatörler gerilim dalgalanmalarını azaltma ve kararlılık sağlamak için kullanılmaktadır.

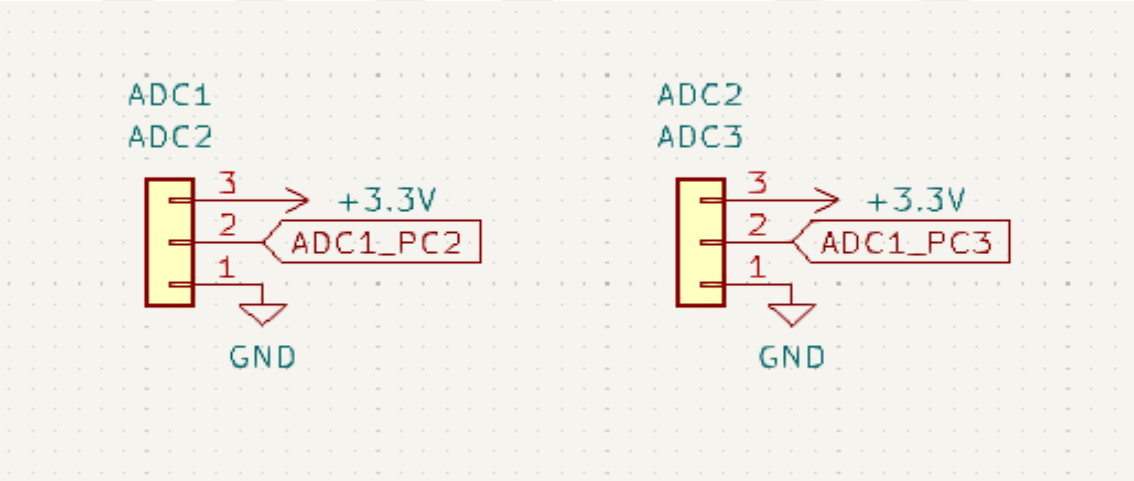
Sensörler ve çıkışlar ise kontrol kartının önemli birer bileşenidir. IMU, barometre ve GPS gibi sensörler, İHA'nın stabilizasyonunu ve kontrolünü sağlamak için şarttır. Bu donanımlar olmadan IHA'nın stabilizasyonunu ve kontrolü imkânsızdır. Bu nedenle tasarlanan kontrol kartı üzerine 2 adet IMU ve 1 adet barometre sensörü entegre edilmiştir. GPS, kumanda ve telemetri kontrolü için 3 adet UART çıkışı ayarlanmıştır. Uart pinleri Şekil 4.9'da gösterilmiştir. İki adet ADC girişi ayarlanmıştır. ADC şematiği Şekil 4.10'de gösterilmiştir. Motorların bağlantıları için sekiz adet PWM sinyali ayarlanmıştır. Sekiz adet PWM çıkışı Şekil 4.11 gösterilmiştir ve kullanım sebebi ise tasarlanan kontrol kartı tüm IHA Airframelere uyumlu olmasıdır. Yani, 4, 6 veya 8 motorlu İHA'lar aynı kontrol kartı üzerinden kontrol edilebilir. Bu özellik, kontrol kartının esnek ve çok yönlü kullanımını sağlar ve farklı İHA tipleri için uygun olacaktır.

Kontrol kartının üzerinde yer alan iletişim protokolleri ve özel olarak ayarlanmış pinlerin açıklamaları aşağıda belirtilmiştir.

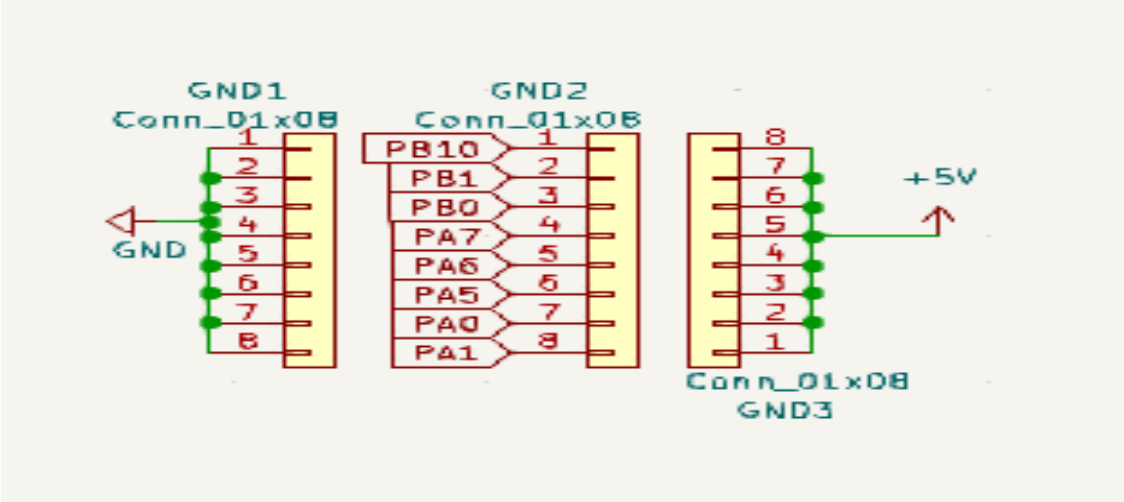
- 8 x PWM Sinyal çıkışı (PB0, PB1, PB10, PA0, PA1, PA5, PA6, PA7)
- 2 x I2C_1 protokolü I2C_1 (SCL = PB11, SDA = PB12)
    - I2C_2 (SCL= PB3, SDA = PB10)
- 2 x SPI protokolü (SPI_SCL = PB13, MIOS = PB14, MOIS = PB15)
- 3 x UART protokolü TX1 = PA2, RX1 = PA3,
    - TX2 = PC6, RX2 = PC7,
    - TX3 = PB6, RX3 = PB7
- 2 x ADC (PC3, PC2)
- 2 x LED (PC0, PC1)



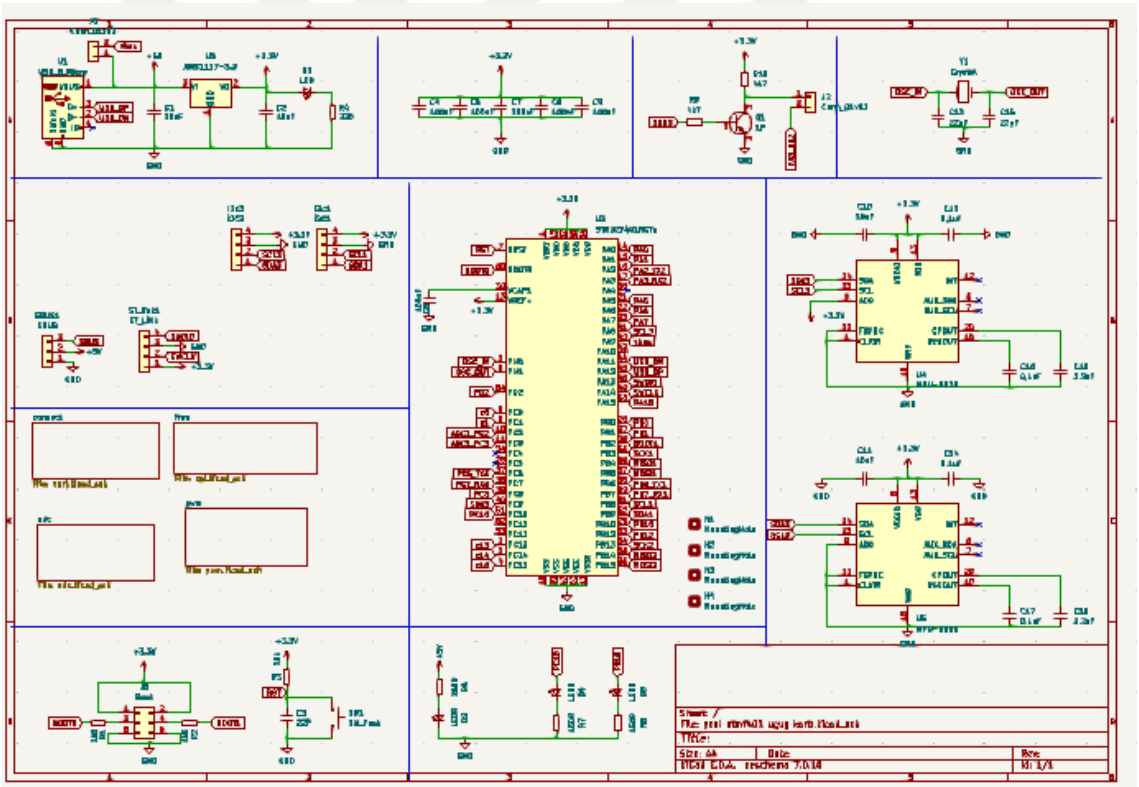
Şekil 4.9. Uart Şematiği



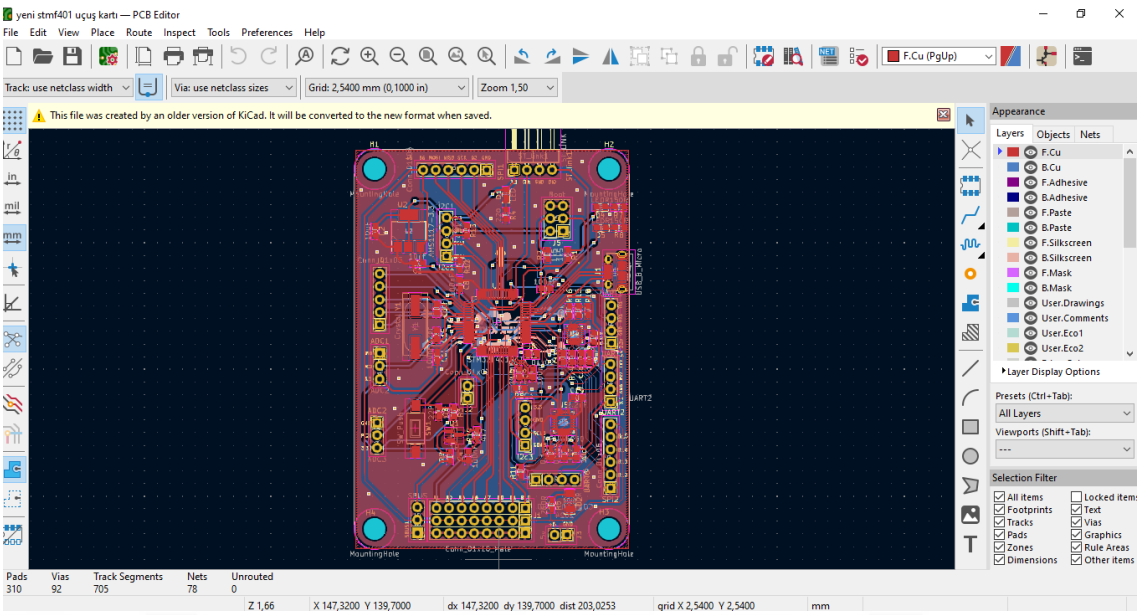
Şekil 4.10.ADC Şematiği



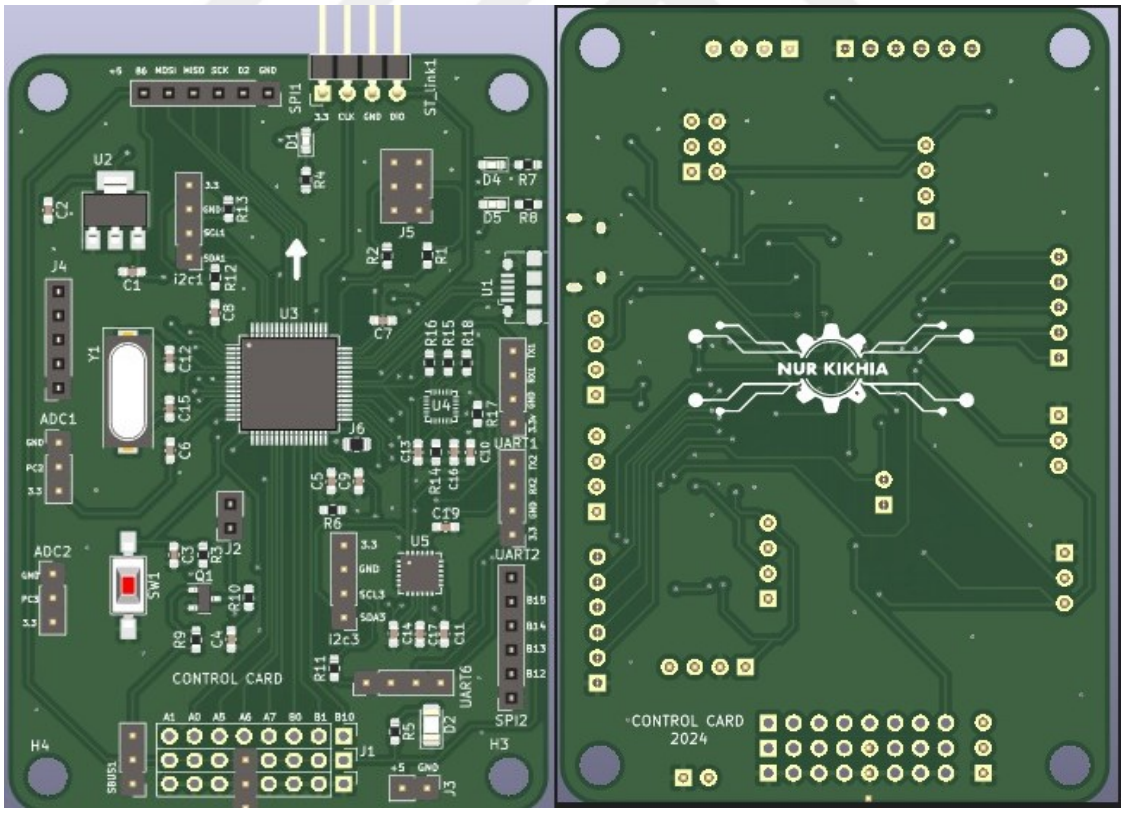
Şekil 4.11. PWM Şematiği



Şekil 4.12. Yerli Uçuş Kontrol Kartın Elektronik Devre Şematiği



Şekil 4.13. PCB Kartın şekli

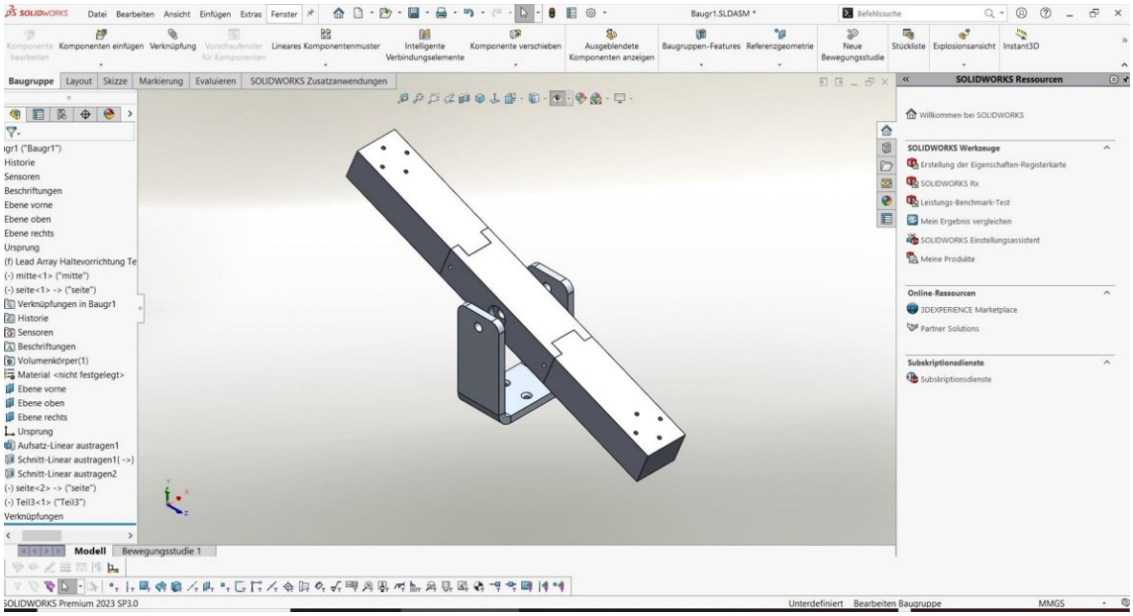


Şekil 4.14. Tasarlanan uçuş kontrol kartının ön ve arka yüzü

5. DENEYSEL DONANIM TASARIMI VE PROGRAMLAMA

5.1. Deney Düzeneği Mekanik Tasarımı

Tasarlanan uçuş kontrol kartının deney çalışmaları için iki boyutlu deney düzeneği oluşturulmuştur. Deney düzeneğinin mekanik kısımları aşağıda Şekil 5.1'de görüldüğü gibi Solidworks'te tasarlanmıştır. Daha sonra bu tasarım G koduna dönüştürülmüş ve 3B yazıcı ile çıkarılmıştır.



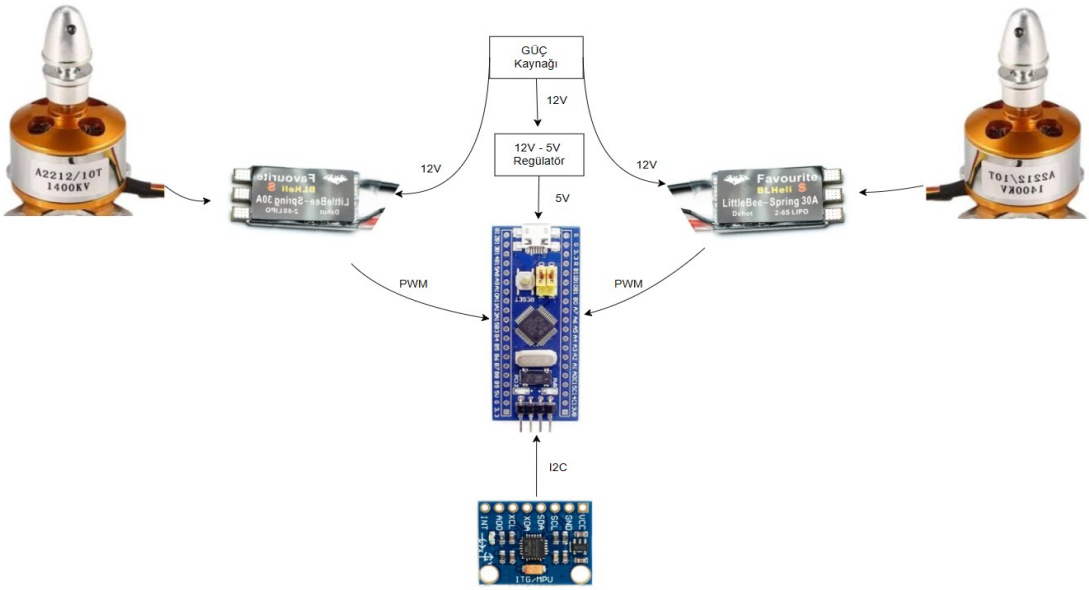
Şekil 5.1. Solidworks'te tasarlanan düzeneği

5.2. Deney Düzeneği Elektronik Donanım

Sistemin içerisinde yer alacak tüm elektronik donanımların, güç bağlantılarının ve haberleşme bağlantılarının detayları, aşağıda bulunan Şekil 5.2'de gösterilmiştir.

5.2.1 STM32 ile PWM Sinyali Oluşturma

PWM kullanarak analog bir elektriksel sinyal elde etmek oldukça yaygın bir tekniktir Şekil 5.3'te gösterilmiştir. PWM'in temel prensibi, darbe genişliğini (yani darbenin yüzde oranını) değiştirerek belirli bir zamanda ortalama bir değer elde etmektir Şekil 5.4'te görülmektedir. Bu, dijital bir sinyalin analog bir değere dönüştürülmesini sağlamaktadır. PWM sinyalleri tamamen anahtarlama elemanları ile sağlanmaktadır. Anahtarlama (transistor, mosfet vs..) ne kadar hızlı yapılırsa, PWM ile aktarılan sinyalin gücü o kadar da artmaktadır.



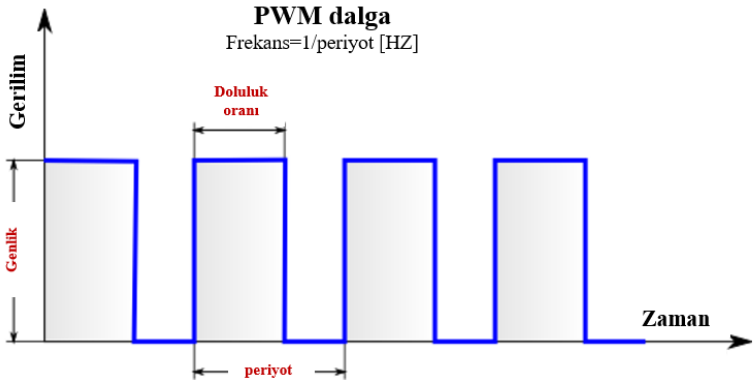
Şekil 5.2. Elektronik Bağlantı Şematığı

PWM Önemi,

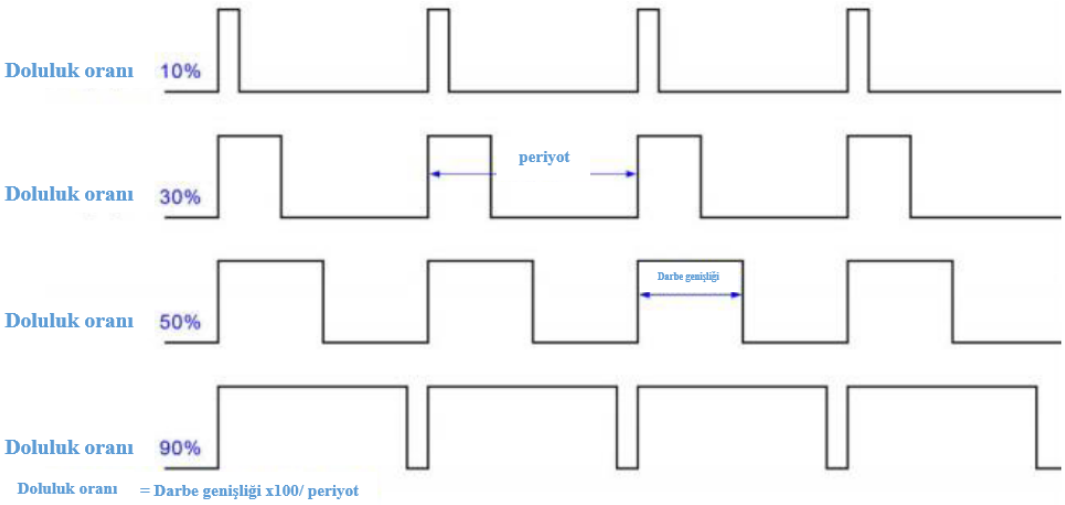
1. Güç tüketimi az.
2. Sinyalleri uzun mesafelere bozulmadan taşımak için yüksek verimli.
3. Uygulaması kolay.

PWM sinyalinin Doluluk oranı olarak adlandırılan görev döngüsü, sinyalin bir periyot boyunca açık (yüksek) kalma süresinin, toplam periyota oranıdır. Doluluk oranı genellikle yüzde olarak ifade edilmektedir. Doluluk oranı, sinyalin genişliğini temsil eder ve çıkış sinyalinin ortalama gücünü kontrol eder. Doluluk oranı göstermesi Şekil 5.5'te verilmiştir ve formülü şu şekildedir:

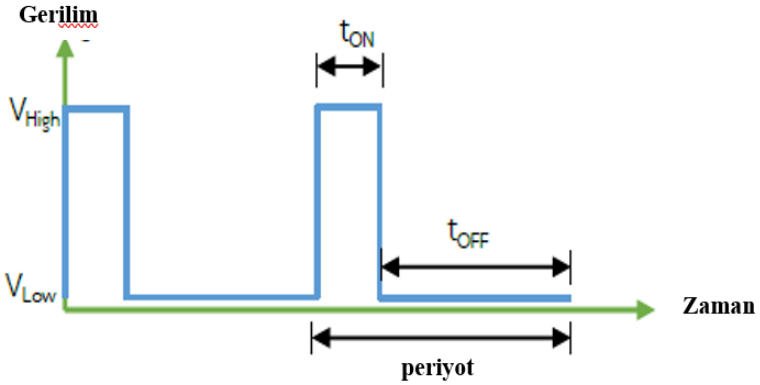
$$Doluluk\ Oranı(\%) = \frac{t_{on}}{t_{on} + t_{off}} \times 100\% \qquad (5.1)$$



Şekil 5.3 PWM ile Doluluk Oranı



Şekil 5.4. %10 Doluluk Oranında minimum, %90 Doluluk Oranında maksimum hız sağlamaktadır



Şekil 5.5. Doluluk oranı göstermesi

5.3. Kontrol Yazılımı

Uçuş kontrol yazılımı, uçuş kontrol kartının mikrodenetleyicisi üzerinde koşan, hava aracının tam kontrolü için sensör ve kumanda sistemleri ile iletişim sağlayarak gelen sinyallerin anlamlandırılması ve kontrol sinyali üretim algoritmalarını içeren yapıdır. Uçuş kontrol yazılımı, hava aracının tam kontrolünü sağlamak için tasarlanmıştır. Uygulama kapsamında kullanılan mikrodenetleyici olan STM32F103C8'in lisanslı yazılım programı olan Keil IDE 5.4 geliştirme ortamı kullanılarak yazılmıştır. Bu mikrodenetleyici, yüksek performanslı ve gömülü sistemler için yaygın olarak kullanılan bir modeldir. Ayrıca, C programlama dili kullanılarak geliştirilmiş ve mikrodenetleyici komponentlerine erişimi kolaylaştırmak için HAL kütüphaneleri ve STM32CubeMx programı kullanılmaktadır.

STM32CubeMx, STM firması tarafından lisanslanmış başlatıcı kod üretici bir programdır. Bu program, mikrodenetleyici başlatıcı kodlarının otomatik olarak oluşturulmasını sağlar. Bu sayede yazılım geliştirme sürecini hızlandırır ve başlatıcı kodlarının doğru şekilde yapılandırılmasını sağlamaktadır.

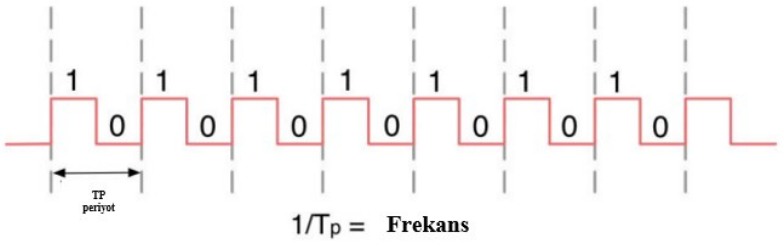
Kontrolcünün performansını artırmak ve işlemleri düzenlemek için gerçek zamanlı işletim sistemi (RTOS) kullanılmıştır. RTOS, belirli zaman aralıklarında çalışan ve önceliklerine göre işlem yapabilen çoklu görevlerin yönetimini sağlamaktadır. Bu, kontrol döngüsü ve alt işlemlerin sıralama ve yönetimini kolaylaştırmaktadır.

Düzeni tasarladıktan sonra motorlar hız kontrollü yapıp ve kullanacağımız tüm elektronik malzemeleri hazır bulunduğunda yazılım işlemi adım adım yapılacaktır. İlk önce STMCUBEMX programında tasarımız ile başlamak gerekmektedir.

5.3.1. Saat Ayarları

Saat, periyodik sinyaller üreten bir cihazdır. Bir bakıma mikrodenetleyicinin kalbidir. Bununla birlikte saat sinyali mikrodenetleyici de sadece tüm bileşenleri ve çevre birimlerini beslemek için kullanılamaz. Ayrıca, güç tüketimi, belirli bir çevre biriminin saat hızı ile doğrudan bağlantılı kritik bir hususdur. Bazı MCU parçalarının saat hızını seçici olarak devre dışı bırakma veya azaltma yeteneğine sahip olmak, genel cihaz güç tüketimini optimize etmeyi sağlamaktadır. Bu, saatin hiyerarşik bir yapıda düzenlenmesini gerektirir ve geliştiriciye farklı hızlar ve saat kaynakları seçme olanağı vermektedir. Saat, genellikle %50 görev döngüsüyle (doluluk oranı) kare dalga sinyali üreten bir cihazdır.

Bir tam dalganın oluşması için geçen süreye periyot denir. Periyot T_p harfi ile gösterilmektedir. Birim zamanda oluşan dalga sayısına da frekans denir. Frekans f harfi ile gösterilmektedir. Frekans Hertz olarak ifade edilir. Periyot ile frekans arasında T = 1/f veya f = 1/T şeklinde bir bağıntı vardır Şekil 5.6'da gösterilmiştir.



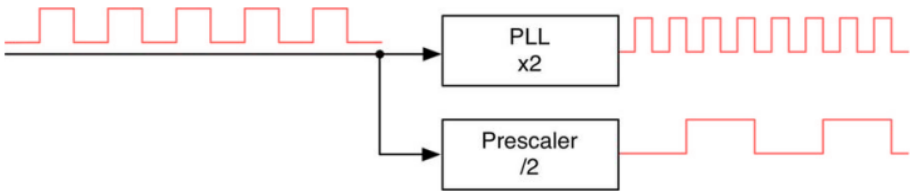
Şekil 5.6. Frekans ve periyot

STM32 mikrodenetleyiciler genellikle dahili bir RC osilatörü (Yüksek Hızlı Dahili (HSI olarak adlandırılır)) ve harici bir özel kristal osilatör (Yüksek Hızlı Harici (HSE olarak adlandırılır)) olmak üzere iki farklı saat kaynağına sahiptir.

Harici kristal osilatörler, genellikle daha yüksek hassasiyet sağlarlar ve çeşitli sıcaklık ve çevresel koşullarda daha istikrarlı bir performans sunarlar. Bu nedenle, harici kristal osilatörler, özellikle PCB'nin çalışma sıcaklıkları ortam sıcaklığının dışına çıktığında, dahili RC osilatörlere göre tercih edilmektedir. Bazı çevre birimleri, özellikle yüksek hızlı olanlar, yalnızca belirli bir frekansta çalışan özel bir harici kristal tarafından senkronize edilir.

STM32 mikrodenetleyicilerinde düşük hızlı osilatör (LSE) ve düşük dahili hız (LSI) gibi düşük hızlı saat kaynakları da bulunur. Bu kaynaklar genellikle Gerçek Zamanlı Saat (RTC) ve Bağımsız Saat izleyici (IWDT) gibi çevre birimlerini beslemek için kullanılır.

Yüksek hızlı osilatörün (HSE) frekansı, Cortex-M çekirdeğinin ve diğer çevre birimlerinin gerçek frekansını oluşturmaz. Saat ağacı olarak adlandırılan karmaşık bir dağıtım ağı, saat sinyalinin STM32 mikrodenetleyicisine yayılmasından sorumludur. Programlanabilir Faz Kilitli Döngü (PLL) ve ön ölçekleyiciler gibi bileşenler kullanılarak, belirli bir çevre birimi veya veri yolu için maksimum hıza ulaşılabilir. İhtiyaç duyulan kaynak frekansını artırmak veya azaltmak, bu yapılandırmalar aracılığıyla mümkündür. Bu, sistem tasarımcısına uygulama gereksinimlerine uygun performans ve güç tüketimi dengesini sağlama esnekliği sağlar. PLL metodu aşağıdaki Şekil 5.7'de göstermiştir.



Şekil 5.7. PLL Metodu

Saat ağacı yapılandırmasının RCC birimi aracılığıyla gerçekleştirilmesi için genellikle üç adımda işlemler yürütülür. Bu adımlar;

1) Yüksek Hızlı Osilatör Seçimi ve Yapılandırılması (HSE/HSI): İlk adım, yüksek hızlı osilatör kaynağını seçmek ve uygun şekilde yapılandırmaktır. Eğer harici bir kristal osilatörü (HSE) kullanılacaksa, bu osilatörün uygun şekilde

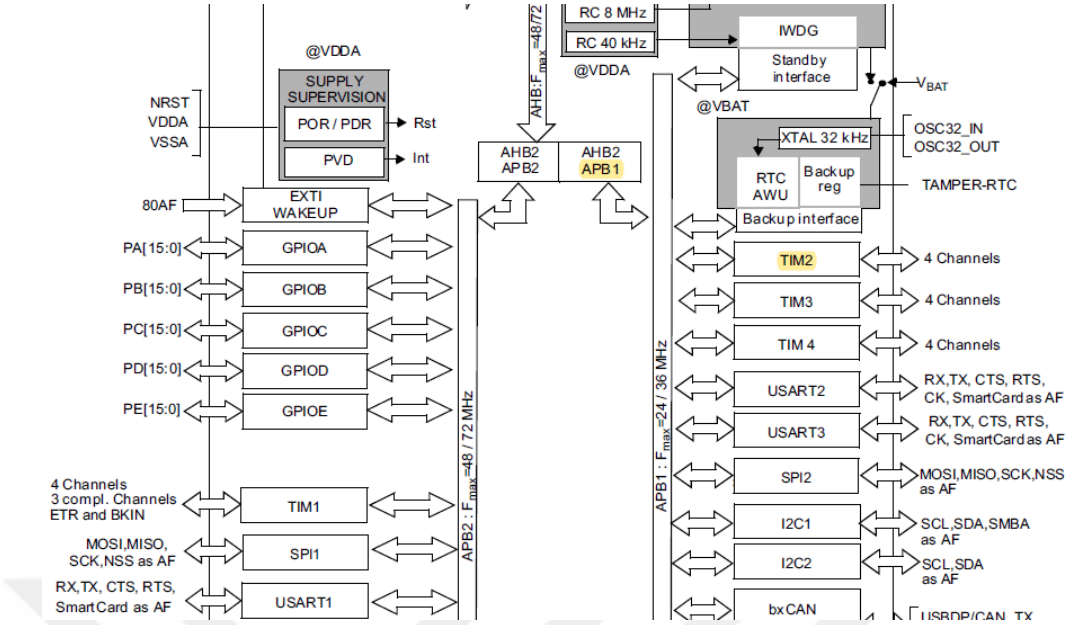
yapılandırılması gerekir. Eğer dahili bir osilatör (HSI) kullanılacaksa, HSI'nin yapılandırılması gerekir. Bu adım, genellikle mikrodenetleyicinin sistem gereksinimlerine ve harici bağlantılara bağlı olarak gerçekleştirilir.

2) Ana PLL Yapılandırması (Opsiyonel): İkinci adım, ana PLL'nin yapılandırılmasını içerir. Ana PLL, sistem saatinin (SYSCLK) frekansını kontrol eder. Eğer SYSCLK'nin yüksek hızlı osilatör tarafından sağlanan frekansla aynı olmasını isteniyorsa, bu adımı atlanmaktadır. Ancak, SYSCLK'nin yüksek hızlı osilatör frekansından daha yüksek bir frekansla beslenmesi gerekiyorsa, ana PLL yapılandırılmalıdır.
3) AHB, APB1 ve APB2 Ön Ölçekleyici Ayarlarının Seçimi: Üçüncü adım, yüksek hızlı saatin (HCLK) istenen frekansına ve gelişmiş frekanslarına ulaşmak için doğru AHB, APB1 ve APB2 ön ölçekleyici ayarlarının seçilmesini içermektedir. Bu adım, mikrodenetleyicinin performansını ve çevre birimlerinin frekanslarını belirlemektedir.

STM32F103 mikrodenetleyici kullanılarak yapılan uygulamada saat kontrol sistemi, yüksek frekansta kontrol sinyalleri üretmek için PLL (Faz Kilitleme Döngüsü) mekanizması kullanılarak ayarlanabilir. PLL mekanizması, 8 MHz osilatörden gelen sinyali 72 MHz'e kadar artırabilir. Sistem saat frekansı 72 MHz olarak belirlenmiş ve PLL mekanizmasının M, N ve P terimleri sırasıyla 8, 9 ve 1 olarak ayarlanmıştır.

Frekansları belirlenen ayarlar dahilinde aşağıdaki gibi ifade edilebilir ve STM32FS103C8 Performans hattı blok şeması Şekil 5.8'de göstermiştir.

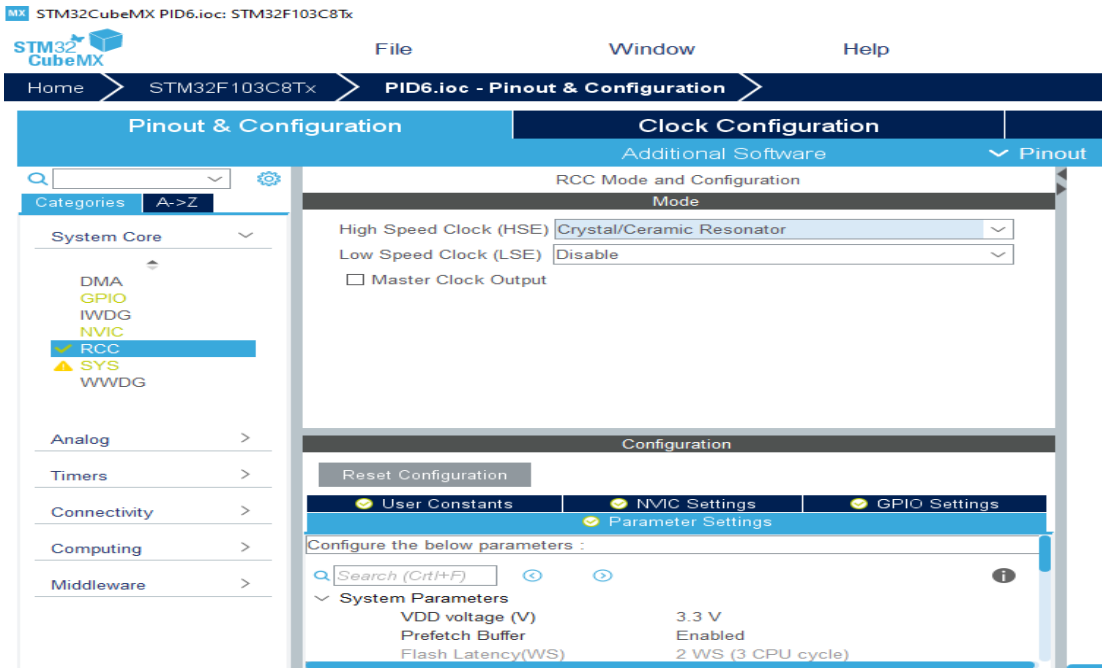
- Cortex Sistem Zamanlayıcı (Cortex System Timer): 72MHz
- APBI Zamanlayıcı (TIM2, TIM3, TIM4) Saat Frekansı: 72MHz
- APB1 Donanım (USART2, USART3, SPI2, I2C1, I2C2) Saat Frekansı :36 MHz
- APB2 Zamanlayıcı (TIM1) Saat Frekansı: 72MHz
- APB2 Donanım (SDIO, USART1, USART6, SPI1, ADC) Saat Frekansı: 72MHz
- Görüldüğü üzere TIM2 biriminin saat frekansı olarak beslendiği hat PLKC2 APB1 hattıdır.
- Genel saat frekansı maksimum çalışma frekansı olan 72 MHz olarak tercih edildi.
- TIM2 birimini besleyen hatta 36 MHz değerine sahiptir.



Şekil 5.8. STM32F103C8 Performans hattı blok şeması

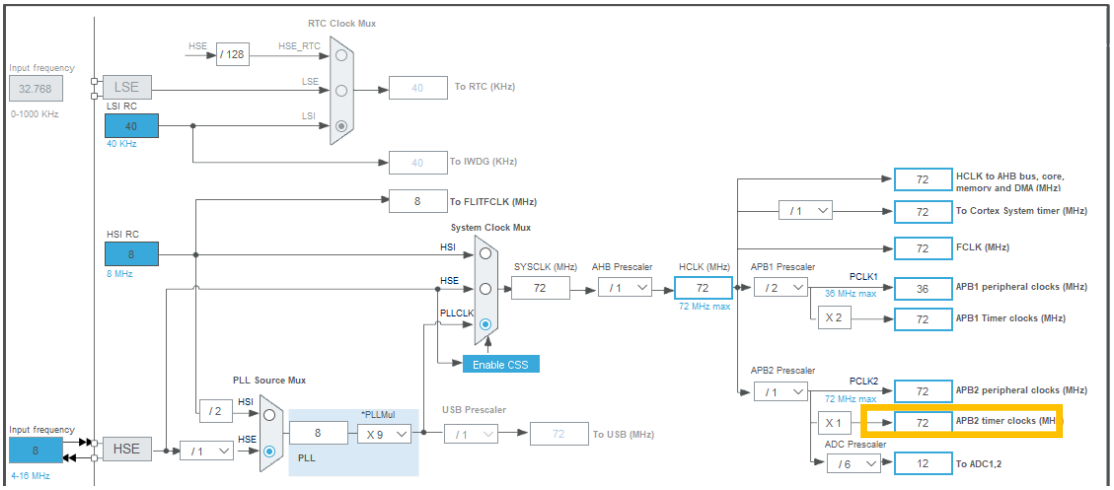
5.3.2. CubeMX Ayarları

STM32CubeMX programında devre için belirli bir frekansta elektrik sinyali oluşturmak için RCC ayarlarından Şekil 5.9'de harici osilatör (HSE) aktif edilir.



Şekil 5.9. Konfigürasyon ayarı

Saat Konfigürasyon kısmında, TIM2 birimini besleyen APB2 Timer Saat hattı 72 MHz değere sahiptir. Şekil 5.10'da gösterilmiştir.



Şekil 5.10. Saat konfigürasyon ayarları

5.3.3. Zamanlayıcı Ayarları

Zamanlayıcı (Timer), basit anlamda bir sayıcıdır(sayaç). Belirli bir sayıdan aşağı birer birer düşer ya da istediğiniz noktaya çıkartabilmektedir. Buna ek olarak aralıklarında seçebilmekte ve uygulayabilmektedir. Bu sayma işlemini saat darbesi hızına (clock) göre yapabildikleri gibi dışarıdan verilen kesmelerle yönetilebilmektedir. Sayma işlemini işlemcimizin saat hızına göre yapmaktadır**.**

Genellikle, bir zamanlayıcı sıfırdan belirli bir değere kadar sayar, bu da çözünürlüğü için maksimum işaretsiz değerden daha yüksek olamaz (örneğin, sayaç 65535'e ulaştığında 16 bitlik bir zamanlayıcı taşar), ancak tam tersi de sayılabilir.
Bir STM32 mikrodenetleyicideki zamanlayıcıların çeşitli özellikleri vardır:

- Zamanlayıcı olarak kullanılabilirler (tüm STM32 zamanlayıcılarında ortak olan özelliktir).
- Harici bir olayın frekansını ölçmek için kullanılabilirler (giriş yakalama modu).
- Bir çıkış dalga formunu kontrol etmek veya bir sürenin ne zaman geçtiğini belirtmek için (çıkış karşılaştırma modu).
- Her kanalda bağımsız olarak kenara hizalı veya merkez hizalı modda PWM sinyalleri oluşturmak için (PWM modu).

STM32 mikrodenetleyicilerinde PWM sinyali üretmek için ZAMANLAYICI birimleri kullanılır. Bu işlem genellikle şu adımlarla gerçekleştirilir:
Ön Ölçekleyici (Prescaler): Zamanlayıcın sayım frekansını belirlemek için kullanılan bir faktördür. Zamanlayıcı, mikrodenetleyicinin ana saatinden gelen bir sinyali kullanarak

sayım yapar. Ancak bu sinyal genellikle çok yüksek frekansta olabilir. Ön Ölçekleyici, bu yüksek frekansı daha düşük bir frekansa bölerek Zamanlayıcın daha düşük bir frekansta sayım yapmasını sağlar. Ön ölçekleyici değeri, Zamanlayıcın sayma frekansını kontrol eder ve 1 ile 65535 arasında değer alabilir.

Zamanlayıcı birimini besleyen saat sinyali, belirli bir frekansta çalışır. Bu frekanstaki sinyali, PWM sinyali üretmek için istenen frekansa dönüştürmek için bir ön ölçekleyici kullanılır. Ön ölçekleyici, zamanlayıcı biriminin giriş saatini belirli bir değere böler ve bu sayede zamanlayıcının çalışma frekansını düşürür.

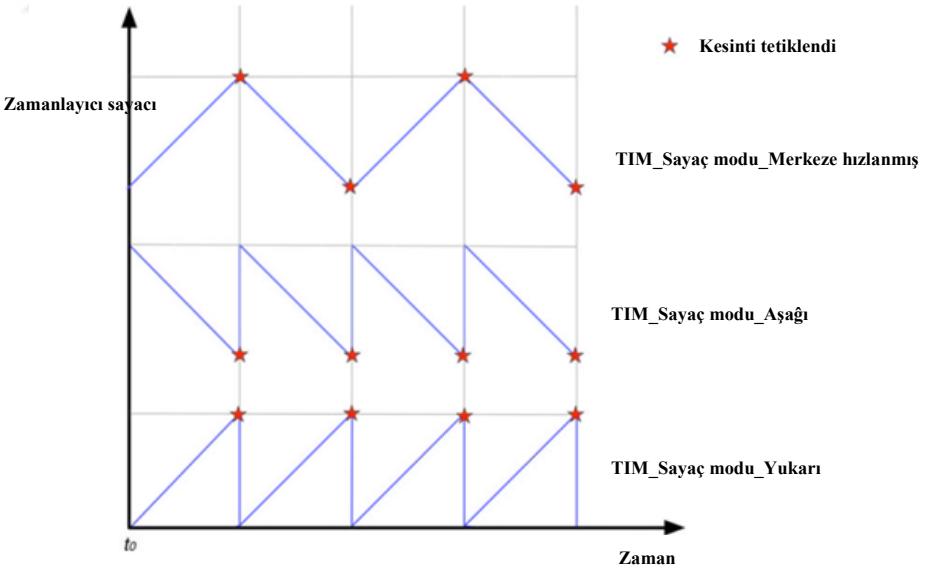
Sayıcı Modu: Zamanlayıcının sayma yönünü ve davranışını belirler.

Üç temel sayıcı modu vardır:

a) TIM_Sayaç modu yukarı: Zamanlayıcı, belirli bir değerden başlayarak yukarı doğru sayar. Başlangıç değerinden maksimum değere ulaştığında sıfırlanır.

b) TIM_ Sayaç modu _Aşağı: Zamanlayıcı, belirli bir değerden başlayarak aşağı doğru sayar. Başlangıç değerinden minimum değere ulaştığında sıfırlanır.

c) TIM_ Sayaç modu _Merkeze hızlanmış: Zamanlayıcı, belirli bir değerden başlayarak yukarı ve aşağı yönde sırayla sayar. Başlangıç değerine ulaşıldığında, sayma yönü otomatik olarak tersine çevrilir ve işlem tekrarlanır.



Şekil 5.11. Sayıcı (Counter) modu

Periyot, bir zamanlayıcının tekrar sayıcıyı yeniden başlatmadan önce ulaşması gereken maksimum değeri belirtir. Bu değer, zamanlayıcı sayacının başlangıç değeri ile karşılaştırılır ve eşit olduğunda zamanlayıcı sıfırlanır veya yeniden başlatılır. Periyot

değeri, Zamanlayıcının çalışma frekansı ve istenilen zamanlama gereksinimlerine bağlı olarak ayarlanır.

Örneğin, bir 16 bitlik zamanlayıcı için periyot değeri 0x1 ile 0xFFFF (65535) arasında olabilir. Bu, 16 bitlik bir zamanlayıcı için 0'dan 65535'e kadar olan bir aralıktır. Benzer şekilde, 32 bitlik bir zamanlayıcı için periyot değeri 0x1 ile 0xFFFFFFFF arasında olabilir.

PWM sinyalinin frekansı ve dolayısıyla darbe süresi, zamanlayıcı biriminin otomatik yeniden yükleme değeriyle belirlenir. Bu değer, zamanlayıcının belirli bir sayıya kadar saymasını ve ardından sıfırlanmasını sağlar. Bu, PWM sinyalinin bir periyodu (darbe süresi) boyunca olan süresidir.

Zamanlayıcı güncelleme olayı hesaplanması:

Zamanlayıcı Güncelleme Olayı (Update Event), zamanlayıcının otomatik yeniden yükleme değerine ulaştığında meydana gelir. Bu olay, zamanlayıcının bir tam devir tamamladığı zamanlama döngüsünü temsil eder. Zamanlayıcı güncelleme olayı hesaplanma formülü aşağıdaki verilmiştir.

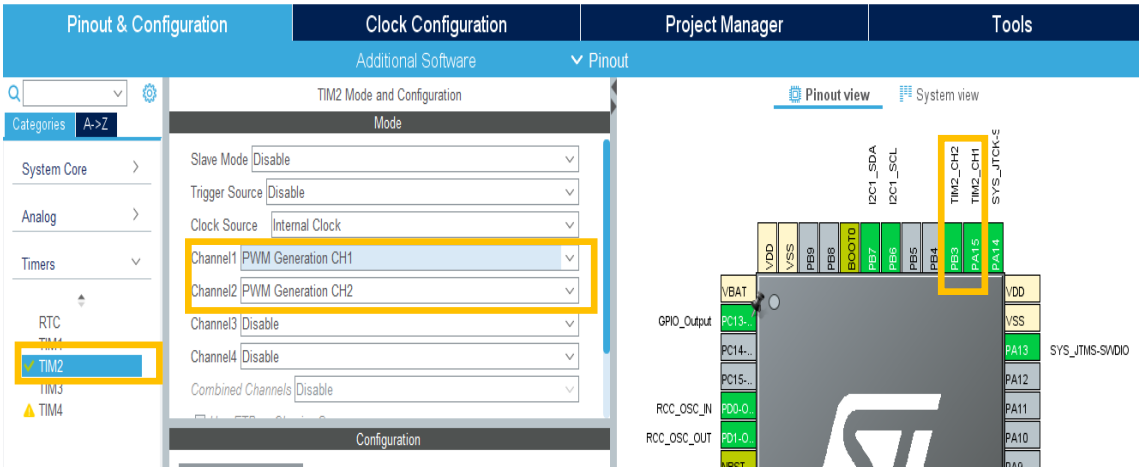
$$Güncelleme\ Olayı = \frac{Zamanlayıcı_{saat}}{(\text{ön ölçekleyici}+1)(periyot+1)} \tag{5.2}$$

Zamanlayıcı Güncelleme Olayın hesaplanması yani olayın gerçekleşme zamanı, zamanlayıcı yapılandırması ve çeşidine bağlıdır. Genellikle, zamanlayıcının ön ölçekleyici değeri, otomatik yeniden yükleme değeri ve ana saat hızı (mikrodenetleyicinin çalışma frekansı) dikkate alınır.

STM32CUBEMX programında Pinout configuration kısmından Timer kısmındaki Timer2 seçip ve 2 farklı kanal için çalışma modu PWM Generation seçilir. Şekil 5.12'de gösterilmiştir. PA15 ve PB3 pinleri TIM2_CH1 ve TIM2_CH2 olarak ayarlanmaktadır.

Bu PA15 üzerinde birinci motora gönderilen PWM sinyali kullanırken TIMER2 zamanlayıcısı ve CH1 kanalı kullanmaktadır.

Bu PB3 üzerinde ikinci motora gönderilen PWM sinyali kullanırken TIMER2 zamanlayıcısı ve CH2 kanalı kullanmaktadır.



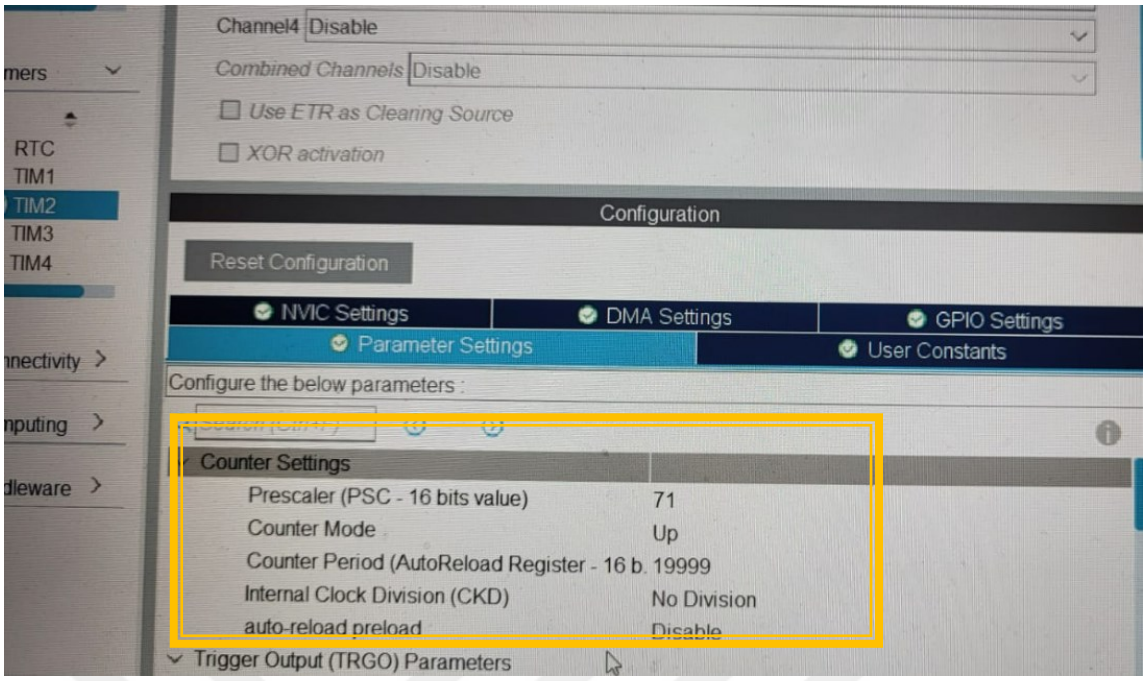
Şekil 5.12. Konfigürasyon ayarları

İhtiyaç duyulan PWM frekans ve çözünürlüğü uygulamadan uygulamaya değişebilmektedir. Uygulamamızda fırçasız dc motorun frekansın 50Hz olması şartıdır yani bu tip motora PWM sinyali üretebilmek için 50Hz frekansa ihtiyacımız var.

Zamanlayıcı Güncelleme Olayın denklemi kullanarak motorların durumunu karşılamak için 50 Hz sonucunu elde etmek istiyoruz. APB1 Zamanlayıcı Saat değeri 72 MHz idi. Parametre settings kısmında prescaler değeri 72 değeri girilir. Counter Periyod ise zamanlayıcının maksimum sayacağı miktardır. 16 Bit bir zamanlayıcı olduğu için sayacağı maksimum değer 65536'dır burada Counter Periyod değeri 19999 değeri girilir. Yapılan işlemler Şekil 5.13'te gösterilmiştir. Zamanlayıcı Güncelleme Olayın denklemi şu şekilde olur.

$$Güncelleme\ Olayı = \frac{72000000}{(19999+1)(71+1)} = 50HZ = \frac{1}{50} = 0.02s \qquad (5.3)$$

Denklemin sonucu yaklaşık 50HZ Dolayısıyla 50'nin tersini alırsak, 0,02s yani 0,02 saniyede 50 Hz frekans elde edilir, yani gönderdiğimiz PWM sinyali her 20 milisaniyede yaklaşık 50 titreşimdir.

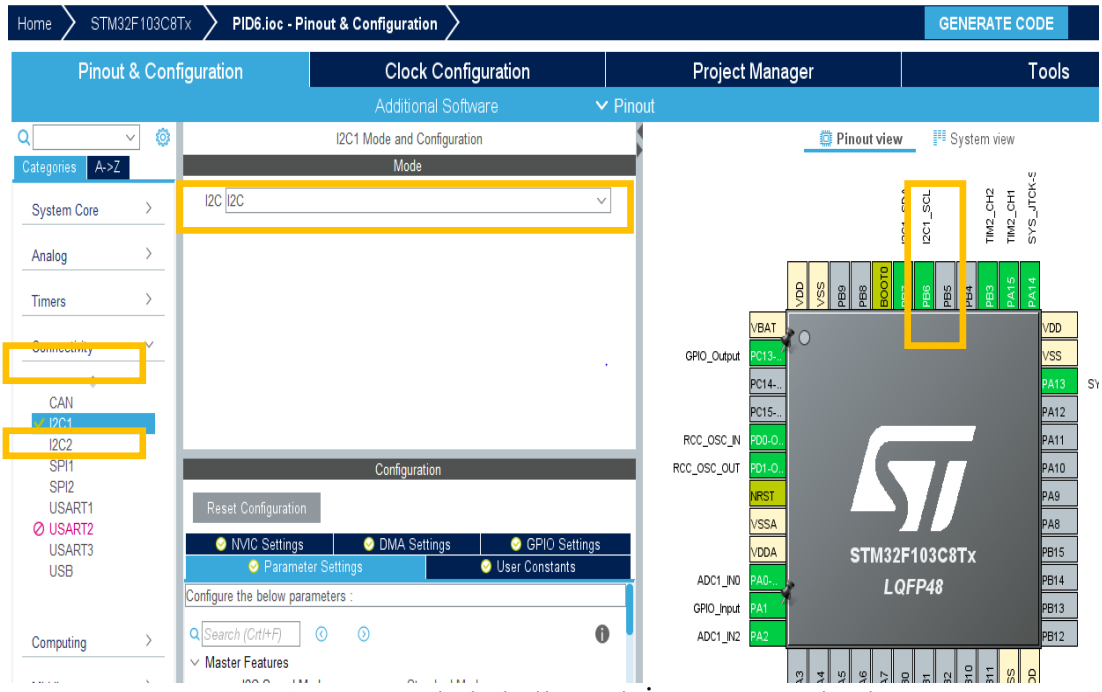


Şekil 5.13. TIM2 PWM Parametrelerinin girilmesi

5.3.4. I2C Protokolü Kullanarak MPU6050 IMU Sensörü Kullanımı

I2C haberleşme protokolünü tanımlamaktadır. I2C, elektronik cihazlar arasında seri iletişim kurmak için kullanılan bir haberleşme protokolüdür. Bu protokolde, SDA ve SCL adında iki temel pin bulunmaktadır.

I2C haberleşme protokolünü kullanacağımızı belirtiyor ve ilgili pinlere atamaları yapılmıştır. Şekil 5.14'te gösterilmiştir.



Şekil 5.14. I2C Protokolü kullanarak İMU sensörü bağlantısı

5.3.5.Map Fonksiyonu Kullanımı

Fırçasız DC motor hızını kontrol etmek için STM32 mikrodenetleyicilerinde PWM sinyali kullanılmaktadır. PWM sinyal değerinin kontrolü, bir potansiyometreden alınan verileri "map" fonksiyonu yardımıyla sağlanır.

Öncelikle, bir "map" fonksiyonu oluşturulur, ardından bu fonksiyonu kullanarak

```
uint32_t MAP(uint32_t au32_IN, uint32_t au32_INmin, uint32_t au32_INmax, uint32_t au32_OUTmin, uint32_t au32_OUTmax)
{
    return ((((au32_IN - au32_INmin)*(au32_OUTmax - au32_OUTmin))/(au32_INmax - au32_INmin)) + au32_OUTmin);
}
```

motor hızını belirli bir giriş aralığından belirli bir çıkış aralığına dönüştürebilmektedir.Proje Keil'de main dosyası içinde yazılan kodda Map fonksiyonu bulunmaktadır.

Yukarıdaki MAP fonksiyonu, belirli bir giriş aralığından başka bir çıkış aralığına dönüşüm yapmak için kullanılabilir. Bu durumda, bu fonksiyon, bir DC fırçasız motora PWM sinyali vermek için kullanılmaktadır.

Bu fonksiyonun giriş ve çıkış aralıklarını ve değişkenleri aşağıdaki gibi sıralanmıştır:

au32_IN: Dönüştürülecek giriş değeri.

au32_INmin: Giriş aralığının minimum değeri.

au32_INmax: Giriş aralığının maksimum değeri.

au32_OUTmin: İstenilen minimum değeri, PWM sinyalinin minimum darbe genişliği (duty cycle) olarak düşünülür.

au32_OUTmax: İstenilen maksimum değeri, PWM sinyalinin maksimum darbe genişliği (duty cycle) olarak düşünülür.

Bu fonksiyon, giriş aralığından alınan değeri, giriş aralığındaki minimum ve maksimum değerler arasında normalize eder. Daha sonra, bu normalize edilmiş değeri çıkış aralığına dönüştürmek için kullanılır. Bu şekilde, belirli bir giriş değeri, belirli bir çıkış aralığında bir PWM sinyali oluşturmak için kullanılmaktadır.

ADC'den okunan değeri bir "map" fonksiyonu aracılığıyla (0 ile 4095) arasından (1000 ila 2000) arasına dönüştürdük ve PWM sinyali oluşturmak için TIM2_CH1 kanalında PWM çıkışını ayarlanmaktadır. Bu PWM sinyali, fırçasız DC 1.motorun hızını kontrol etmek için kullanılmaktadır.

Sonuç olarak bu map fonksiyonu kullanarak, belirli bir motor hızı aralığına karşılık gelen PWM sinyali değerlerini elde edebilmektedir.

Stm32cubemx programını tamamlayıp Generate Code diyip keil üzerinde kodları açılmaktadır. Kodlar arasında IMO sensöre ait kütüphaneleri, PID ve Kalman filtresini de kütüphaneleri ayrı ayrı eklenmektedir, kodları projenin çalışacağı şekilde doğru şekilde olmalıdır.

5.4. Uygulamada PID Kontrol Kullanılması ve Ayarlanması

Uygulamada kontrol ve ayar işlemleri aşaması tamamladıktan sonra deneysel denemeler aşamasına geçilmiştir. Burada ilk olarak motorları Esc'leriyle uygun şekilde bağlayıp mikrodenetleyiciye ve aküye doğru ve hassas bir şekilde bağlanacaktır. Daha sonra ST-link V2 bilgisayara bağlayıp St.-link Utility programını kullanarak mikrodenetleyiciye kodlar göndermeye başlanacaktır. Üçüncü aşama olarak stm32studio programını kullanarak istediğimiz veriyi izleyip değerlendirilecektir.

Mikrodenetleyicinizde PID kontrol algoritmasını uygulamak için gerekli kodu yazılmaktadır. Bu, hedeflenen açıları veya hızları belirlemek ve IMU sensöründen gelen verilerle geri bildirim almak için uygun matematiksel işlemleri içerecektir.

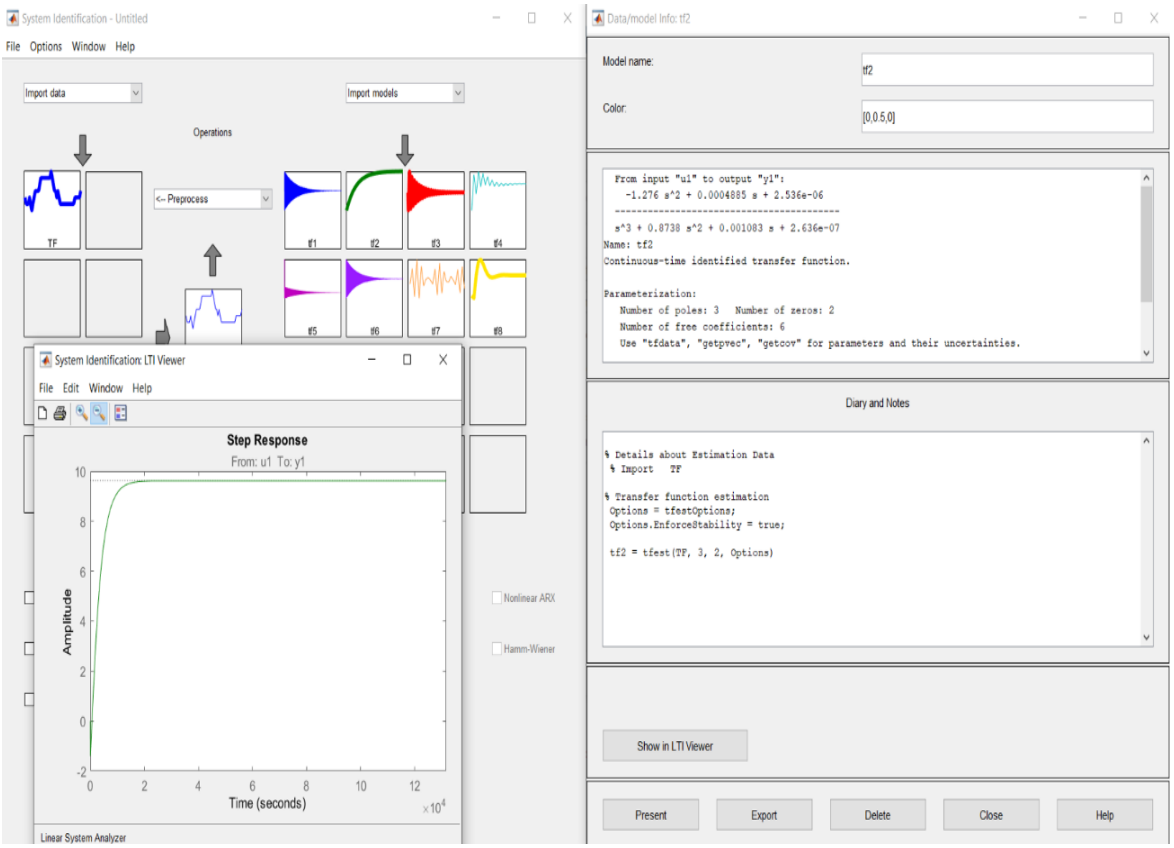
PID kodu, STM32 mikrodenetleyici kullanarak drone kontrolünü sağlamak için tasarlanmaktadır. Kod, sensör verilerini okuyarak PID kontrol algoritması kullanarak cihazın istenen açıya sabitlenmesini sağlar.

PID kontrolünün etkinliği, kullanılan PID parametrelerine (K_P, K_i, K_d) bağlıdır. Bu parametrelerin doğru bir şekilde ayarlanması, istenen performansın elde edilmesi için kritik öneme sahiptir. PID kontrolör katsayıları, manuel olarak ayarlanabilir veya Matlab/Simulink gibi otomatik ayarlama yöntemleri kullanılarak ayarlanabilir. Ancak, her iki yöntemin de kendine özgü avantajları ve dezavantajları bulunmaktadır.

Birinci yöntemde, PID kontrol parametrelerini belirlemek için manuel bir yaklaşım kullanılmıştır. Bu yöntemde, PID programlama bölümünü içeren kodlar Keil programına eklenmiş ve STM32 mikrodenetleyici kartına gönderilmiştir. Ardından, mikrodenetleyici kartı üzerinde çeşitli deneyler yapılmış ve K_P, K_i, K_d gibi PID parametreleri kod üzerinden sürekli olarak değiştirilerek istenen performans elde edilmeye çalışılmıştır. Kararlı bir durum elde edildiğinde, bu parametreler kod içinde kaydedilmiştir. Bu yöntemle, manuel ayarlama yöntemleri kullanılarak sistemin uygun açıya deneylerle kalibre edilmesi sağlanmıştır.

İkinci yöntemde ise, Matlab Simulink kullanılarak otomatik bir yaklaşım benimsenmiştir. Bu yöntemde, STM32 üzerinden giriş ve çıkış verileri UART protokolü ve TTL dönüştürücüsü kullanılarak bilgisayara aktarılmıştır. Ardından, Matlab üzerinde

bu veriler işlenmiş ve giriş-çıkış değişkenleri kaydedilmiştir. Daha sonra, Sistem Tanılaması (System Identification) kullanılarak sistemin matematiksel modeli belirlenmiş ve transfer fonksiyonu elde edilmiştir. Bu transfer fonksiyonu üzerinde katsayılar hesaplanmış ve Matlab üzerinde yapılan denemelerle doğruluk oranı %95 gibi yüksek bir seviyede tespit edilmiştir doğruluk oran şekil. Ancak, elde edilen transfer fonksiyonu üzerindeki katsayılar, yazılım koduna eklenerek yapılan denemelerde istenen performansı sağlamamıştır. Bu nedenle, yerleşme zamanının yaklaşık olarak 7 saniyeye kadar çıktığı gözlenmiştir. Matlab Simulink üzerinde hesaplanan transfer fonksiyonun Şekil 5.15'te gösterilmiştir.



Şekil 5.15. Sistem tanılaması

Sistem istenilen açıda (0) stabil durum gelmelidir. Motor hareket ettiğinde tekrar stabil hale gelmesi birkaç saniye alır ve salınımlar olur. PID katsayıları deneyerek daha hızlı sürede ve daha az salınımlar ile stabilizasyona gelmesine yardımcı olmaktadır. Manuel ayarlama yöntemleri, deneyimli bir kullanıcının sistemi dikkatlice izleyerek ve katsayıları el ile ayarlayarak yukarıda bahsedildiği gibi gerçekleştirilmiş. Bu yöntem,

sistem davranışını anlamak ve kontrolör parametrelerini elle optimize etmek için gereklidir.

Yapılan testler sonucunda, birinci yöntemle elde edilen PID parametrelerinin daha hassas ve daha hızlı bir şekilde sistemi yerleştirdiği tespit edilmiştir. Bu nedenle, projenin devamında birinci yöntemle belirlenen PID parametreleri kullanılarak ilerleme kararı alınmıştır

5.5. Kalman Filtre Kullanmanın Etkisi

Durum uzayı modeli ile gösterilen bir dinamik sistemde, modelin önceki bilgileriyle birlikte giriş ve çıkış bilgilerinden sistemin durumlarını tahmin edilebilen filtredir. Kalman filtresi, önceki bilgileri, giriş verilerini ve çıkış verilerini kullanarak bir sistemin durumunu tahmin etmek için kullanılan bir tür tahmin ve düzeltme filtresidir. Durum uzayı modeli, bir sistemi matematiksel olarak ifade etmek için kullanılan bir modeldir. Bu model, sistemin içsel durumlarını ve bu durumların zamanla nasıl değiştiğini tanımlar.

IMU sensörü, hız, ivme ve oryantasyon gibi bilgiler sağlar. Ancak, bu sensörlerden gelen veriler genellikle gürültülü olabilir ve doğru oryantasyon açılarını elde etmek için filtreleme gerektirebilir. Sensörden gelen veriler daha doğru ve stabil bir değerleri elde edebilmemiz için kalman filtre kullanmasına karar verilmiştir. Kalman filtresi, IMU sensörlerinden gelen verileri işleyerek sistem durumunu tahmin edebilir ve gürültüyü azaltabilir.

Kalman filtresi, sensör verilerini tahmin etmek ve gürültüyü azaltmak için kullanılan gelişmiş bir yöntemdir. Bu filtre, sensör modelini ve ölçüm modelini kullanarak, geçmiş ölçümlere ve mevcut ölçüme dayalı olarak en olası durumu tahmin eder. Doğru şekilde kullanıldığında, IMU sensörlerinin performansını ve doğruluğunu önemli ölçüde artırabilir.

IMU sensörlerinden gelen verileri işlemek için Kalman filtresi kullanılırken, durum uzayı modeli sisteminin matematiksel ifadesi Kalman filtresinin içine entegre edilir. Bu, sistemin durumlarını ve durumların zamanla nasıl değiştiğini temsil eden bir dizi denklem olarak ifade edilir. Kalman filtresi, bu denklemleri kullanarak IMU sensörlerinden gelen verileri işleyerek sistem durumunu tahmin eder.

IMU Sensör Verileri için Kalman Filtresi Kullanmanın Avantajları:

- Gürültülü ve hatalı verilerle çalışabilir.
- Sensör verilerini tahmin etmek için kullanılabilir.

- Zamanla daha doğru ve stabil sonuçlar elde eder.

Kalman filtre projeye dahil edilmediğinde sistem nasıl çalışacağını görmek için deneme yapıldı. Kalman filtre denklemlerini genel koda eklenmediği zaman projeyi çalıştığımızda, sistemin çalışmaya başlayacağını ve motorların dönmeye başlayacağını fark edilmekte ancak sistemin dengesiz olduğunu ve çok fazla titreşim bulunduğunu gözlenmiştir. Kısacası sistemde çok fazla titreşim ve kararsızlık gözleniyorsa, bu durum Kalman filtresinin eksikliğinden kaynaklanabilir.

Kalman filtresi, gürültülü verileri azaltarak ve önceki bilgilerle birleştirerek daha doğru bir tahmin sağlar. Bu nedenle, filtre olmadan sistem, giriş verilerine ve dinamik sistemdeki gürültülere daha duyarlı olabilir.

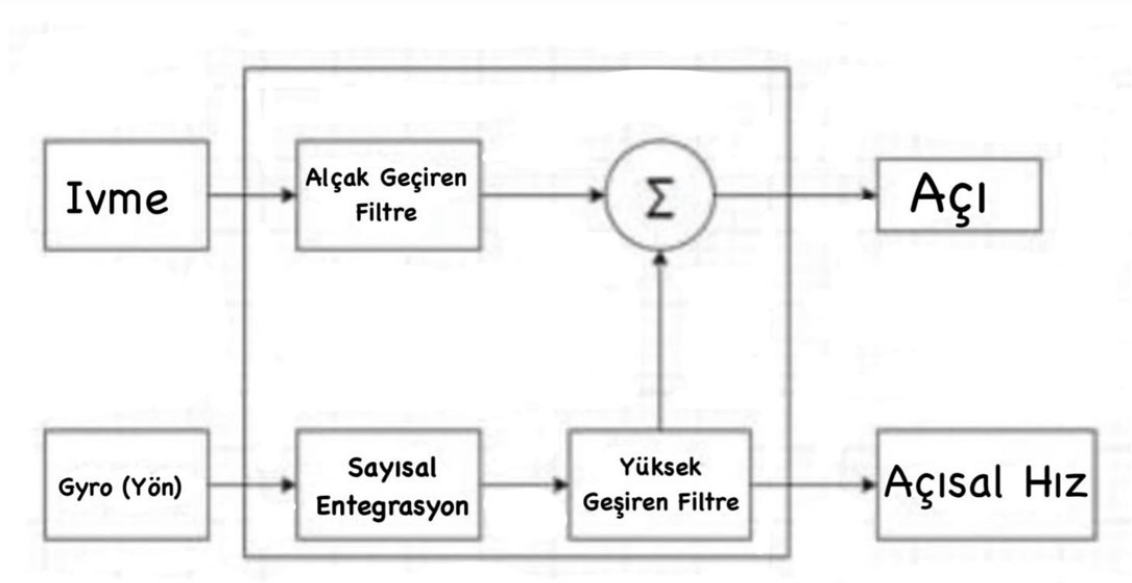
Tekrardan motorları Esc'leriyle uygun şekilde bağlayıp mikrodenetleyiciye ve aküye bağlanacaktır.Keil programındaki koda Kalman filtre denklemlerini ekleyip ve ST-link V2 bilgisayara bağlayıp St-link Utility programını kullanarak hex dosayası eklenecek ve sistemi tekrar çalıştırmaktadır ve mikrodenetleyiciye kodlar göndermeye başlanacaktır.

STMstudio programı açıp axf dosyası eklenip STMStudio'da "Start Recording" düğmesine tıklayıp görmek istendiği veriler (sensör verilerini, açıları, RMPWM ve LMPWM) vs. izlemek için eklendiğini değişkenleri seçip açı hareketler göre nasıl değiştiği görülmektedir. Sensörün x eksenli ve y eksenli hareket ettiğine göre değerleri değiştiğini görülmektedir.

Motorlardan motorların iyi çalıştığını gösteren çınlamalar duymaya başlanmaktadır. Sekiz saniye sonra motorlar dengeli bir şekilde dönmeye başlıyor ve sistem stabil hale geliyor. Denge açısını biraz değiştirerek sistemin eski konumuna (stabil moda) döndüğünü kısa sürede ve çok az salınımlar ile fark edilecek.

5.6. Tamamlayıcı Filtre Kullanılması

İHA’nın eğim açısını elde etmek için üzerinde bulunan ivmeölçer ve yön belirleyen MPU6050 sensöründen gelen verilerin hassasiyeti arttırmak amacıyla bu yöntem gerçekleştirilecektir. Sabit bir ağırlıklı ortalama oranıyla IMU verileri ile referans verilerin ortalaması alınması yöntemidir ve kullanılabilecek en basit algoritmadır. Tamamlayıcı filtre hem alçak geçiren hem de yüksek geçiren filtresi uygulanılmaktadır. Tamamlayıcı bir filtrenin diyagramı Şekil 5.16'de gösterilmektedir.



Şekil 5.16. Tamamlayıcı filtre blok diyagramı

Aşağıdaki denklemin mantığına göre yazılımsal olarak sensörlere filtrelenmesi uygulanılacaktır.

$$\theta = \alpha(Gyro\ Açısı \times Örnek\ zamanı) + (1 - \alpha).ivme ölçer\ açısı \quad (5.4)$$

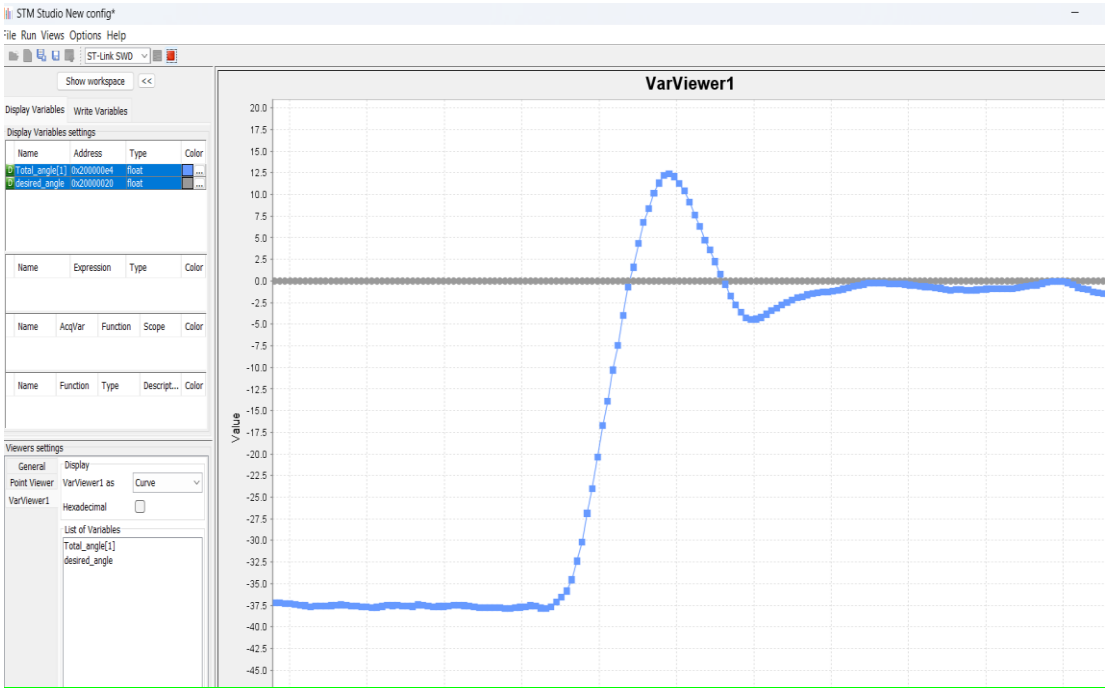
Burada α, ölçümünün her bir sensöre hangi derecede bağlı olacağını belirleyen filtre katsayısıdır.

6. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

6.1. Farklı Referans Girişli Uygulama Sonuçları

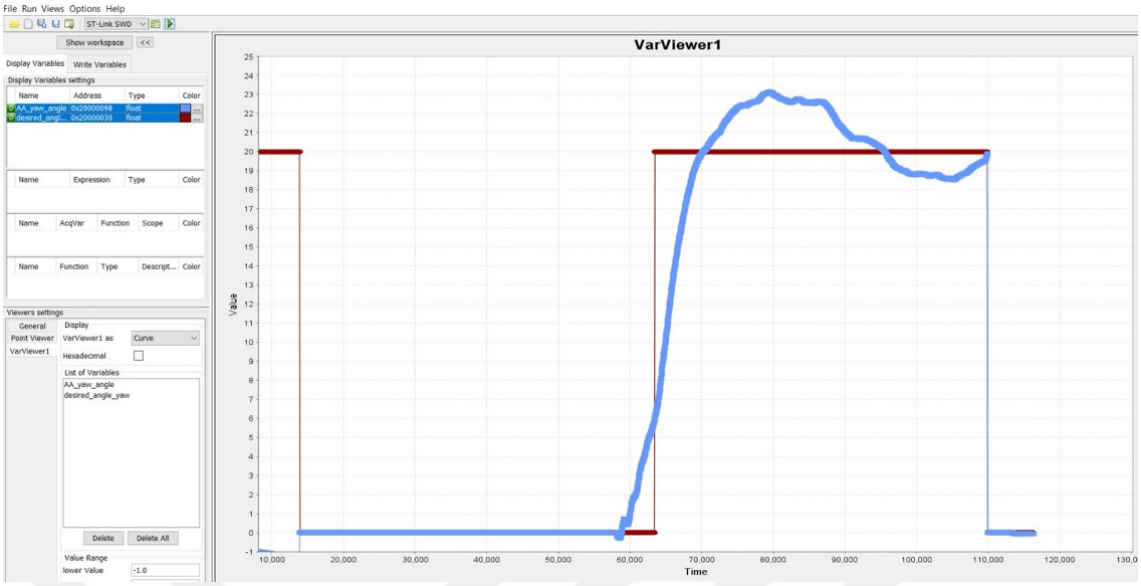
Değişen giriş referans açıları ile farklı deneyler gerçekleştirilmiş, elde edilen sonuçları aşağıdaki gibi sıralanmıştır:

Deney (1): giriş referans açısı (0) olarak seçildiğinde ve dışarıdan bozucu etki uygulandığında, sistem kendiliğinden sıfır açısına stabil hale gelip sinyalin karalı bir şekilde kaldığını tespit edilmiştir. İstenilen referans açısı ile sistemin sinyalleri Şekil 6.1'de gösterilmiştir.



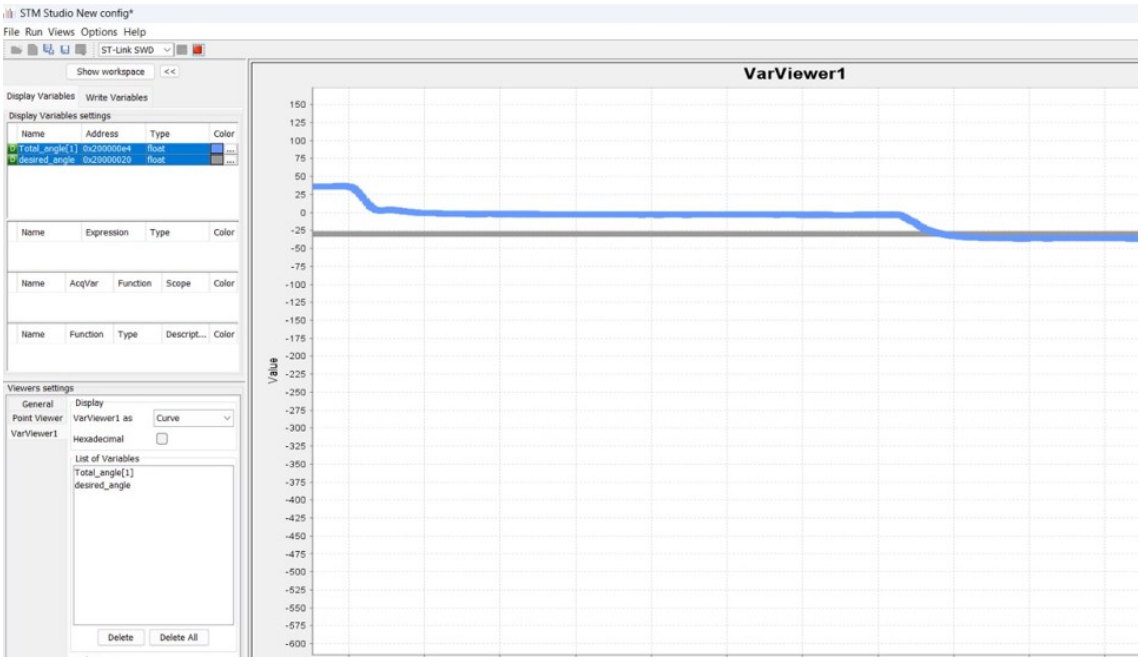
Şekil 6.1.Sıfır derece giriş referansın grafiği

Deney (2): giriş referans açısı 20 derece seçildiğinde dışarıdan (el ile hareket ettiğinde) bir etki uyguladığında sistemin otomatik olarak stabil şeklinde giriş referans açısına yani kararlılık durumuna yaklaştığını fark edilmektedir. Zaman ile istenilen açı ve sistemin açısı eksenleri arasındaki grafik üzerinde Şekil 6.2'de gösterilmiştir.



Şekil 6.2. Yirmi derece giriş referansın grafiği

Deney (3): giriş referans açısı -30 derece seçildiğinde dışarıdan (el ile hareket ettiğinde) bir etki uyguladığında sistemin otomatik olarak stabil şeklinde giriş referans açısına yaklaşıp sistem kararlılık hale geldiğini fark edilmektedir. Zaman ile istenilen açı ve sistemin açısı eksenleri arasındaki grafik üzerinde Şekil 6.3'te gösterilmiştir.

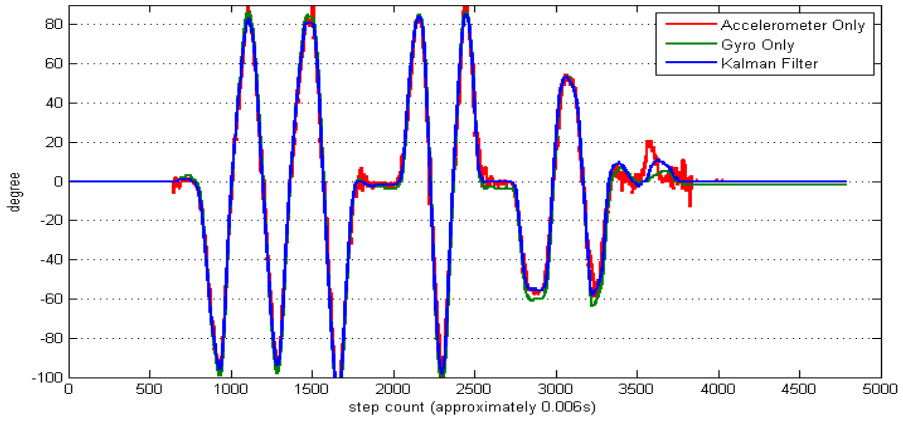


Şekil 6.3. Eksi otuz derece giriş referansın grafiği

6.2. PID Kontrol ve Kalman filtre Uygulama Sonuçları

Uçuş kontrol kartı tasarımı için yapılması gereken birçok uygulama vardır. Bunların en önemlisi PID kontrol ve Kalman Filtre uygulamasıdır. Uygulama sonuçları şu şekilde açıklanacaktır;

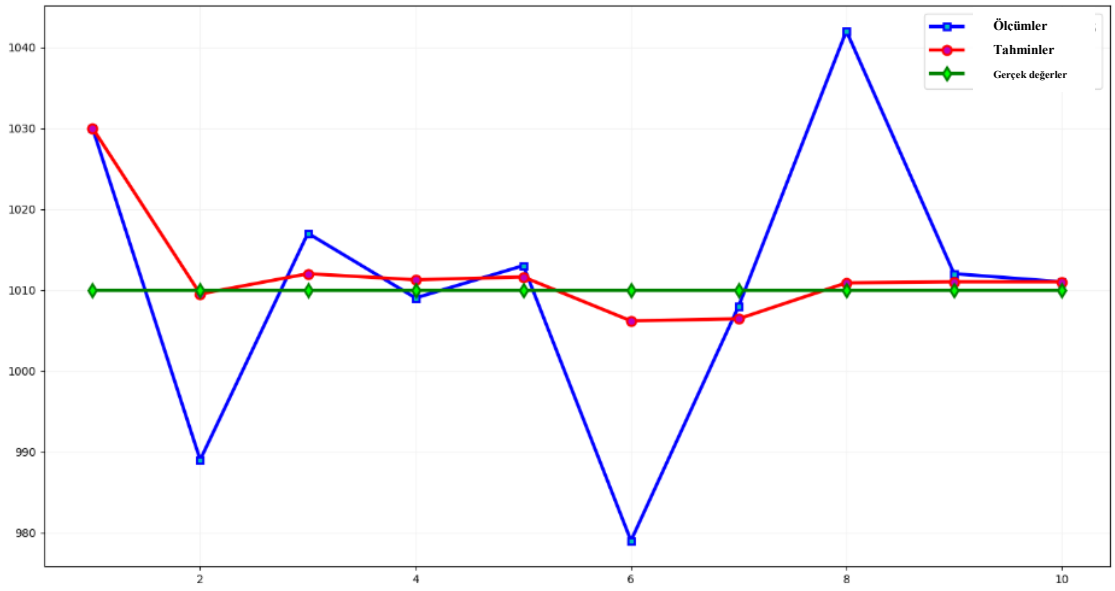
1. Verileri ayarlama ve okuma: Sensörlerden ham ivmeölçer ve jiroskop verilerini okunmuştur ve bunların uygun birimlere (açı dereceleri) dönüştürülmüştür. MPU6050 sensördeki bunan ivmeölçer ve jiroskop ünitesinden gelen verileri birleştirerek Kalman filtresi kullanarak açısal tutumu tahmin etmeye çalışmıştır. İvmeölçer ve jiroskopun her biri avantajları ve dezavantajları vardır. İvmeölçerler, sabit ivmeleri ölçme eğilimindedir ve kısa vadeli titreşimlerden etkilenirken, jiroskoplar daha hassas bir açısal hız ölçümü sağlar ancak zamanla kayma eğilimindedir. Kalman filtresi, bu iki sensörün avantajlarını birleştirerek daha doğru bir sonuç elde etmek için sensör verilerini sürekli olarak değerlendirir ve günceller. Bu sayede, kısa vadeli titreşimlerden kaynaklanan ivmeölçer hataları düzeltilirken, jiroskopun zamanla kayma eğilimi de telafi edilebilir. Kalman filtresi kullanılarak birleştirilen verilerle elde edilen açısal tutum tahminleri, her iki sensörün de eksikliklerini azaltarak daha doğru hale getirilmiştir. Sensörden gelen değerler STMStudio arayüzünde görünmektedir. Aşağıdaki Şekil 6.4'te Kalman filtre yardımıyla IMU sensörden gelen veriler daha doğru ve stabil bir değerleri elde edilmiştir.



Şekil 6.4. Kalman filtre yardımıyla IMU sensörü verilerin okunması

2. Hata hesaplama ve PID kontrolü: Mevcut açı (total angle [1]) ile istenen açı (desired_angle) (motorun ayar Y ekseni için) arasındaki hatayı hesaplanmıştır. Daha sonra motor çıkışını hataya göre ayarlamak için Oransal İntegral Türevsel (PID) denetleyiciyi uygulanmıştır. Hataya ve önceden tanımlanmış sabitlere (K_p , K_i, K_d) dayalı olarak orantı terimini (pid_p), integral terimini (pid_i) ve türev terimini (pid_d) hesaplanmıştır. Nihai PID çıkışı bu üç terimin toplamıdır. PID = pid_p + pid_i + pid_d: Bu satır, bireysel terimleri toplayarak genel PID çıktısını hesaplamıştır.

Sensörün kalman filtre ve ortalama filtresi uygulandıktan sonra IMU verilerin Şekil 6.5'te gösterilmiştir.



Şekil 6.5. Kalman filtre ve ortalama filtresi yardımıyla IMU sensörü verilerin okunması

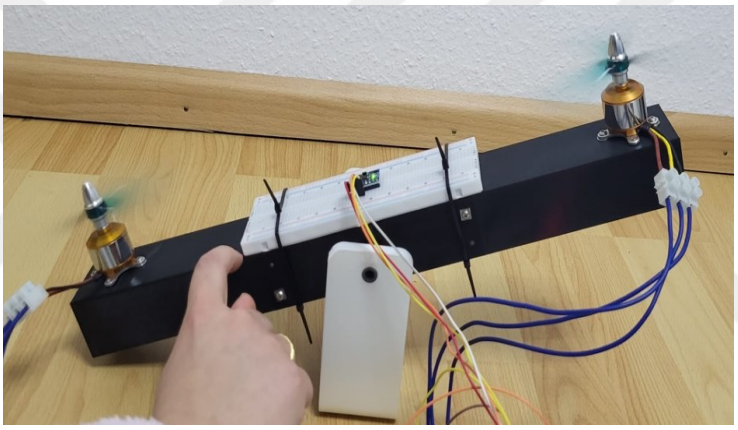
Yukarıdaki şekilde görülen mavi rengi IMU ham verisi ve kırmızıda gösterilen çizgiyi ortalama filtresi uygulandıktan sonrası veri ayrıca yeşil renkte gösterilen çizgiyi kalman filtresi kullanıldıktan sonrası veridir.

6.2.1 Denge Kontrol Uygulama Sonuçları

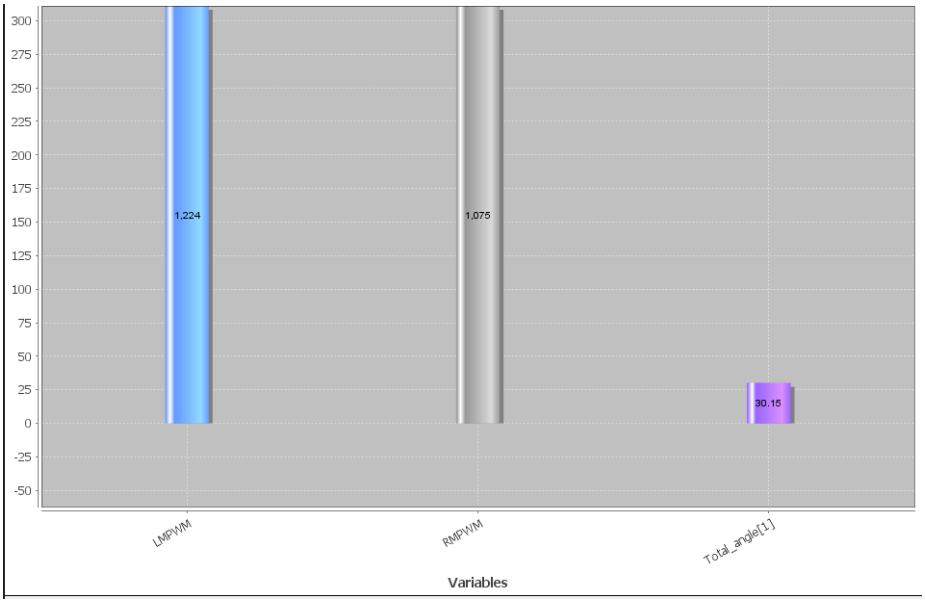
İki motorlu sol ve sağ bir sistem için bir kontrol döngüsü uygulanmıştır. Sol ve sağ motorlar için ayrı PWM sinyalleri oluşturmaktadır LMPWM ve RMPWM: Bu değişkenler sırasıyla sol ve sağ motorlara gönderilen PWM değerlerini tutmaktadır.

Motor çıkışlarının sınırlandırılması: PWM değerleri, motorların zarar görmesini önlemek ve uygun kontrolü sağlamak için belirli bir aralıkla (1020-1350) sınırlandırılır.

Sisteme sola doğru basıldığında (Şekil 6.6) sol taraftaki motor yükselmeye ve stabil bir pozisyona ulaşmaya çalışacaktır dolayısıyla sol motor hızını artmak için daha fazla çalışacak ve motora gönderilen LMPWM sinyalinin değeri artacaktır çünkü motorun hızı artmak için doluluk oranı değeri artmak gerekmektedir. Sol motorun pwm değeri arttığında ikinci motorun RMPWM değeri de aynı oranda azalacak böylece kendileri belirlenen değeri arasında korutmaktadır. Ayrıca sistem sola doğru bastığınızda IMU sensörü X eksenli açısı artı değerde olduğunu okunur ve hareketlere göre değişmektedir. Şekil 6.7'de göstermiştir.

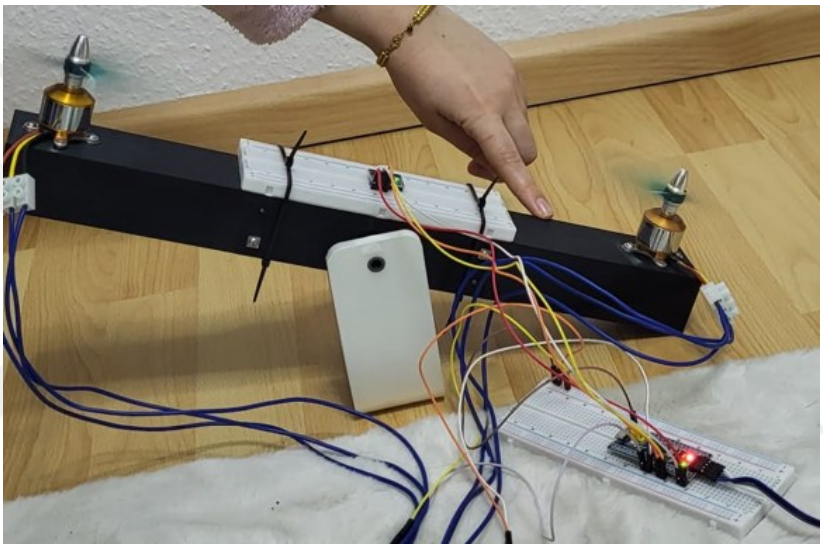


Şekil 6.6. Sisteme sola doğru bastığında

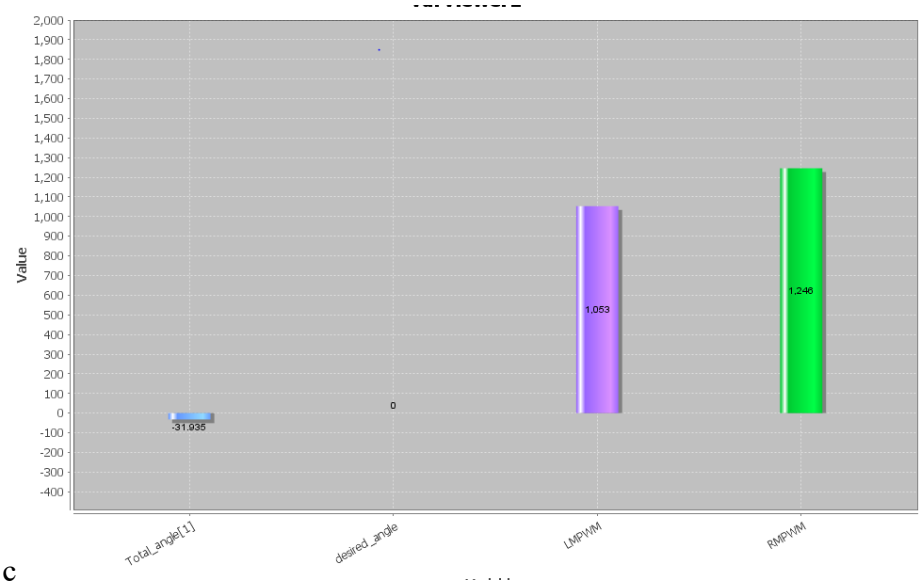


Şekil 6.7. Sisteme sola doğru bastığında IMU sensöründen okunan değerler

Aynı şekilde Sistem sağa doğru basıldığında (Şekil 6.8) sağ taraftaki bulunan motor yükselmeye ve diğer motor inmeye başlanmaktadır ve sağ motor kararlı duruma ulaşmaya çalışacaktır dolayısıyla sağ motor hızını artmak için daha fazla çalışacak ve motora gönderilen RMPWM sinyalinin değeri artacaktır .Sağ motorun RMPWM değeri arttığında ikinci motorun LMPWM değeri de aynı oranda azalacak böylece kendileri belirlenen değeri arasında korutmaktadır .Ayrıca sistem Sağ doğru bastığınızda IMU sensörü X eksenli açısı eksi değerde olduğunu okunur ve hareketlere göre değişmektedir. Şekil 6.9'da göstermiştir.



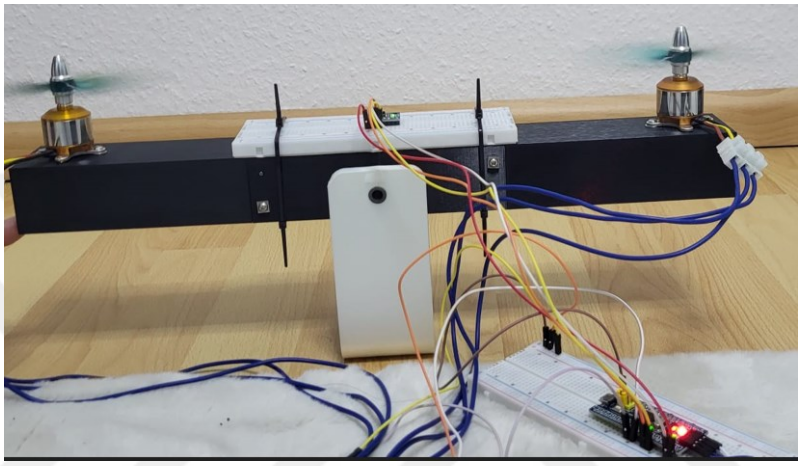
Şekil 6.8. Sisteme sağa doğru bastığında



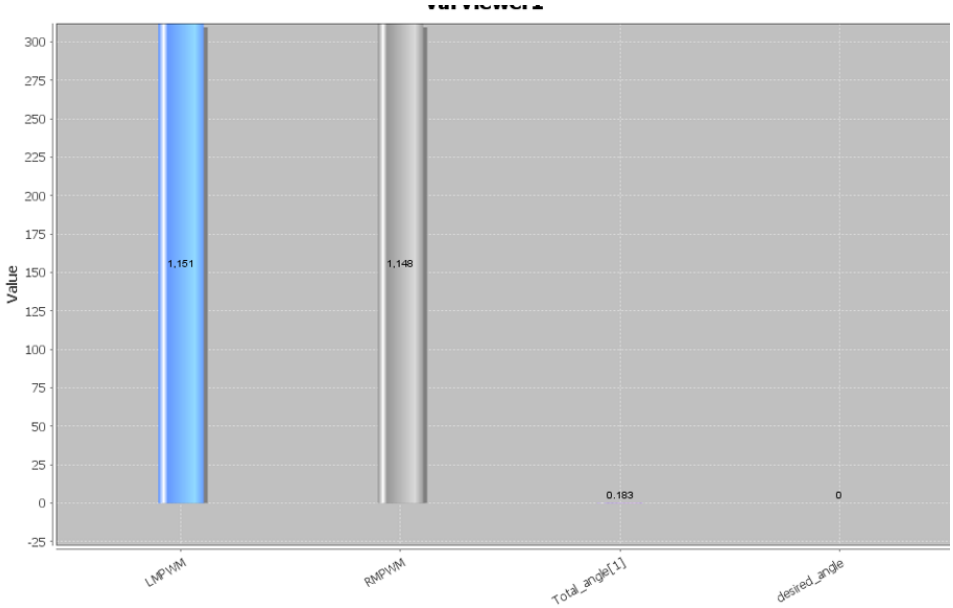
c

Şekil 6.9. Sisteme sağa doğru bastığında IMU sensöründen okunan değerler

Sistem denge durum olduğunda ve karalı aşamasına ulaştığında (Şekil 6.10) iki motor aynı seviyede sıfır açıda stabildir ve sol ve sağ motorların LMPWM ve RMPWM değerleri hemen hemen eşittir. Ayrıca X eksenli açısı istenilen açıya (yaklaşık sıfır değeri) eşit olmakta ve kararlı hale gelmektedir İstenilen açı işe sistem kararlı durumda olduğundaki açıdır (Şekil 6.11). Sistem bu denge durumuna gelmesi için hem Kalman filtresi hem de PID kontrolün etkilerinden oluşmuştur.



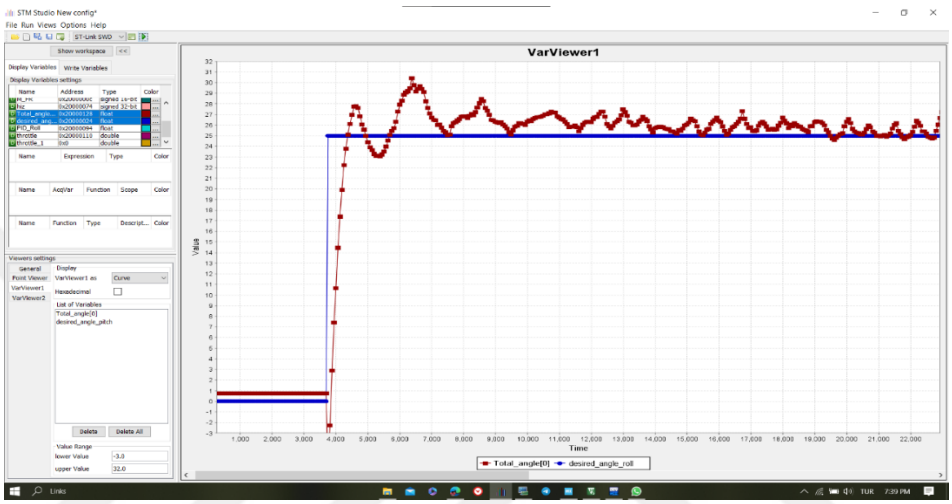
Şekil 6.10. Sistem Denge Durum Olduğunda



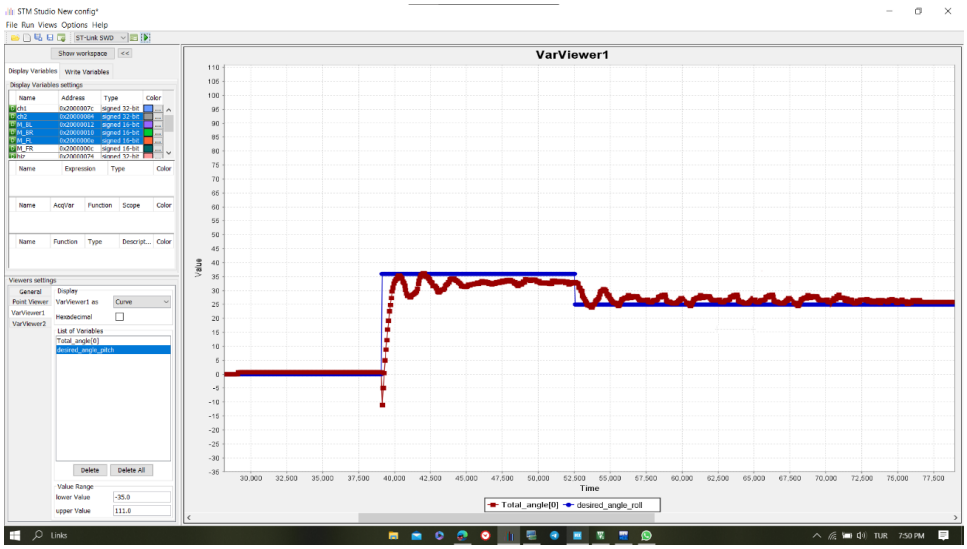
Şekil 6.11. Sistem denge durum olduğunda IMU sensöründen okunan değerler

Sistemin kararlı hale gelebilmesi için kalman filtresi olmadan sistem yaklaşık olarak 7 saniye gibi bir yerleşme süresine sahiptir, bu da sistemde istenmeyen

dalgalanmaların sönmesi için geçen süredir ve sistem yerleştikten sonra basit bir titreşimi ile devam etiğini gözlenmiştir. Yerleşme süresi ve sensör verileri, Şekil 6.12'de grafiksel olarak gösterilmiştir. Kalman filtresi uygulandıktan sonra, sistemdeki yerleşme süresi azaldığını gözlenmiştir ve daha önce görülen parazitler kalman filtresi sayısında kaldırılmıştır. Bu sistem daha hızlı bir şekilde kontrolünü sağlayarak daha kararlı bir duruma gelmektedir. Şekil 6.13'te yerleşme süresi ve sensör verileri ile görsel olarak belirtilmiştir.



Şekil 6.12. Kalman Filtre uygulamadan



Şekil 6.13. Kalman Filtre uyguladıktan sonra

6.2.2. Bozucu Tepkisi

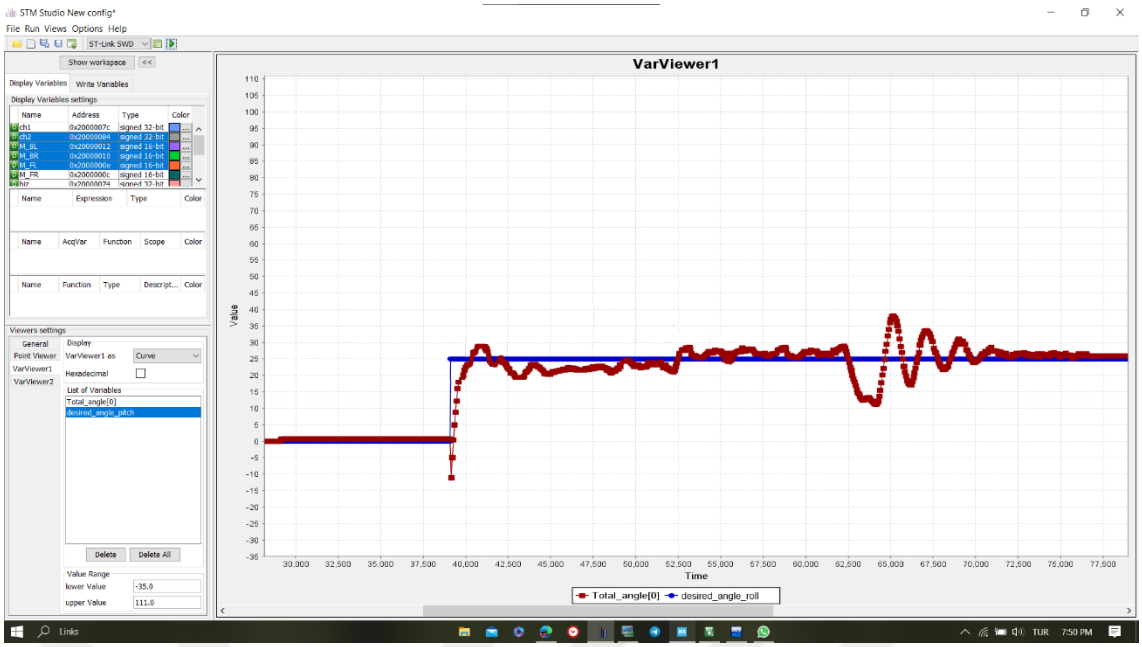
PID sistemi tasarımımızın tamamlanması ve yerleştirilme süresinin en aza indirilmesi için gerekli adımlar atılmıştır. Sistemin harici etkenlerden etkilenme, stabilizasyon bozulması, yük ekleme ve tekrar kararlı hale gelme süreleri ile ilgili deneyler gerçekleştirilmiştir.

Deneye geçmeden önce, kullanılan motorların taşıyabileceği yükün hesaplanması gerekmektedir. Kullandığımız motorlar Şekil 3.16 gösterildiği gibi A2212/6T 2200KV modelidir. Motorun veri tablosunu incelediğimizde kullanılan pervanenin türüne bağlı olarak farklı ağırlıkları kaldırabilirler. Kullandığımız pervane 6030 modelidir ve her bir motor, 732g ağırlığı kaldırabilir. Dolayısıyla, test aşamasında kullanılacak yük, motorun taşıyabileceği maksimum yükün yarısını geçmemelidir. Bu doğrultuda, yük 350g olarak belirlenmiştir. Motorun veri tablosu aşağıda gösterilmiştir.

Modelli	KV (rpm/V)	Gerilim (V)	pervane	Yük Akımı (A)	Çekmek (g)	Güç (W)	Yeterlik (g/W)	Lipo Hücre	Ağırlık (g) Yaklaşık
	930		1060	9.8	660	109	6.1		
	1000		1047	15.6	885	173	5.1	3S	
A2212	1400	11.1	9050	19.0	910	210	4.3		52
	2200		6030	21.5	732	239	3.1	2-3S	
	2450		6*3	25.2	815	280	2.9		

Tablo 6.1. A2212/6T Veri Tablosu

Deneyin daha verimli bir şekilde gerçekleştirilmesi amacıyla, sistemin roll ekseni 25 derecede stabil olacak şekilde gerçekleştirilmiştir. Sistem, stabil hale geldikten sonra, sağdaki motorun üzerine 300 gram ağırlığında bir yük eklenmiştir. Yük, Şekil 6.14'te görüldüğü gibi, 62,5 saniye sonra eklenmiştir. Sistemin tekrar kararlı hale dönmesi yaklaşık 6 saniye sürmüştür. Yükün eklenmesiyle birlikte sistemdeki yerleşme süresinin arttığı gözlemlenmiştir. Daha hafif yüklerin eklenmesi durumunda ise sistemdeki kararlı hale dönme süresinin daha az olduğu tespit edilmiştir. Bu durum, yük ile yerleşme süresi değişimi arasında doğru orantılı bir ilişki olduğunu göstermektedir.



Şekil 6.14. PID Sinyal Grafiği

7. SONUÇLAR VE ÖNERİLER

7.1 Sonuçlar

Çalışmanın temel amacı, yerli uçuş kontrol kartlarının tasarım ve geliştirme süreçlerini inceleyerek, havacılık endüstrisinin ulusal bağımsızlığını artırma potansiyelini araştırmaktır. Bu amaç doğrultusunda benzer çalışmalar literatürde incelenmiş ve yerli uçuş kontrol kartlarının mevcut durumu ve eksiklikleri belirlenmiştir. İnceleme sonuçlarına dayanarak, donanım ve yazılım bileşenlerinin tasarımı, prototip üretimi ve testleri detaylı bir şekilde ele alınmıştır.

Tasarlanan uçuş kontrol kartının testleri için iki boyutlu bir deney düzeneği oluşturulmuş ve 2 BLDC motor ve 2 pervane için denge testleri başarıyla gerçekleştirilmiştir. Bunun için birçok çalışma ve denemeler yapılmıştır. İlk olarak, IMU sensöründen gelen ham veriler okunmuş ve bu değerleri İHA'nın açılarının ve hareketlerinin nasıl etkilediği incelenmiştir. IMU sensörleri, İHA'nın konumunu ve hareketini algılamak için kullanılan önemli bileşenlerdir. Ham verilerin doğru bir şekilde okunması ve yorumlanması, İHA'nın stabil bir uçuş gerçekleştirmesi için kritik öneme sahiptir.

İkinci olarak, motor hızının kalibrasyonu yapılmıştır. Bu süreçte, motorlara 1000 ile 2000 mikro saniye arasında değişen PWM sinyalleri gönderilerek motor hızları ayarlanmıştır. PWM, motor hızını kontrol etmek için sıkça kullanılan bir tekniktir. Bu işlem, İHA'nın farklı hızlarda ve yönlere doğru hareket etmesini sağlayan temel bir adımdır. Motor hızının doğru bir şekilde kalibre edilmesi, İHA'nın istenilen manevraları yapabilmesi için gereklidir.

Bu aşamalar, İHA'nın temel kontrol sistemlerinin geliştirilmesine yönelik önemli adımları temsil etmektedir. IMU sensörünün doğru bir şekilde okunması ve motor hızının kalibrasyonu, İHA'nın stabil bir şekilde uçabilmesi ve istenilen görevleri yerine getirebilmesi için hayati öneme sahiptir. Bu aşamaların başarılı bir şekilde tamamlanması, İHA'nın daha ileri düzeyde kontrol ve yönlendirme sistemlerinin geliştirilmesine olanak sağlayacaktır.

Kalman filtresi, IMU sensöründen gelen ham verileri kullanarak açı değerlerini daha doğru ve kesin bir şekilde tahmin etmek için yaygın olarak kullanılan bir yöntemdir. IMU sensörleri ivmeölçerler ve jiroskoplardan oluşur ve İHA'nın pozisyonunu ve hareketlerini izlemek için kullanılır. Ancak, IMU sensörlerinden gelen veriler genellikle

gürültülü ve hatalıdır, bu da doğru açı ölçümlerini zorlaştırır. Kalman filtresi, bu tür gürültülü verileri temizleyerek daha doğru sonuçlar elde etmeye yardımcı olmaktadır. Uygulanan Kalman Filtre yapısı, IMU sensöründen gelen ham verilerin değerlendirilmesinde kullanılarak, sistemdeki belirsizlikler ve gürültüler başarıyla minimize edilmiştir. Genel olarak Kalman Filtre algoritması iki adımdan oluşmaktadır; tahmin adımı ve güncelleme adımı. Tahmin adımında, sistem modeli ve kontrol girişleri kullanılarak bir sonraki durum tahmini yapılmıştır. Güncelleme adımında ise, IMU sensöründen gelen gerçek ölçümler kullanılarak durum tahmini güncellenmiş ve daha doğru bir tahmin elde edilmiştir.

PID denetleyici ise, sistemin istenilen durumuna hızlı ve hassas bir şekilde yaklaşmasını sağlayan bir kontrol yöntemidir. PID denetleyici kazanç katsayıları uygun bir şekilde ayarlanarak sistemin istenilen duruma daha hızlı ve kararlı bir şekilde gelmesi başarıyla sağlanmıştır. Deneysel sonuçlar kısmında, sistemin sabit bir referans açısına hızla yerleştiği, kare dalga formunda düzenlenen referans açılara başarışı bir şekilde yerleştiği, bozucu durumlara hızla tepki verdiği gözlemlenmiştir.

Ayrıca Kalman Filtresinin etkileri deneysel olarak incelenmiştir. Kalman filtresi ve PID denetleyici birlikte kullanılarak, IMU sensöründen gelen verilerin doğru bir şekilde işlenmesi ve sistemin istenilen duruma hızlı bir şekilde gelmesi sağlanmıştır. Kalman filtresi gürültülü verileri temizlerken, PID denetleyici sistemi istenilen duruma yönlendirmiştir. Deneysel sonuçlardan görüldüğü gibi, sistemin hızlı bir şekilde kararlı duruma gelmesi ve az salınımlarla istenilen görevleri başarılı bir şekilde yerine getirmesi sağlanmıştır. Son olarak, başarılı bir uçuş kontrol için filtreleme, özellikle Kalman Filtresi, Sensör Füzyonu gibi tamamlayıcı algoritmaların kararlılığı etkilediği görülmüştür.

7.2 Öneriler

Yapılan çalışmada, tasarlanan kontrol kartı 2 boyutlu denge sisteminde başarılı bir şekilde kullanılmıştır. Bundan sonra gerçek bir İHA üzerinde testlerinin tamamlanması gerekir. Ayrıca, sistemin kinematik ve dinamik denklemleri çıkarılarak sistemin teorik tasarımının da yapılması, denetleyici tasarımının en doğru şekilde ortaya konulması gereklidir. Deneme yanılma yoluyla elde edilen PID denetleyici katsayıları, modelleme ile elde edilmelidir.

Bu süreçte MATLAB ve Simulink gibi araçlar kullanılarak, İHA'nın kinematik ve dinamik denklemleri modele aktarılarak PID katsayılarının otomatik olarak hesaplanması mümkündür. Bu işlem, Auto-Tune veya diğer otomatik ayarlama yöntemleri kullanılarak gerçekleştirilebilir.

Bu yaklaşımın uygulanmasıyla, İHA'nın yükseklik koruma ve manevra kabiliyetlerinde daha kararlı uçuşlar gerçekleştirmesi sağlanabilir. PID kontrolcülerinin katsayıları, sistem dinamiklerine uygun şekilde ayarlandığında, istenilen uçuş performansını elde etmek daha mümkün olacaktır.

Bu şekilde, matematiksel modelleme ve PID katsayılarının otomatik ayarlanması kullanılarak, İHA'nın stabilite ve performansını artırmak için daha sistematik ve doğru bir yaklaşım benimsenmiş olur. Bu da İHA'nın güvenliği ve etkinliği açısından önemli bir adım olacaktır.

İHA'nın uçuş denetim parametrelerinin yeterince iyi bir şekilde ayarlanamaması uçuş sırasında istenmeyen kazalara neden olabilmektedir. Kullanılan donanımsal birimlerin yüksek maliyetli ve elde edilmesinin uzun sürmesi sebebiyle bu tip hava araçlarında denetleyicinin parametre ayarlarının doğru yapılması büyük önem arz etmektedir.

Bu nedenle bu çalışmaya ek olarak döner kanatlı insansız hava araçları için kontrol algoritmalarının ve uçuş denetim parametrelerinin test edilebileceği bir düzenek tasarımı gerçekleştirilebilir.

Saha uçuş testleri sırasında yüksek hassasiyet ve doğruluk elde edebilmek için farklı hava koşullarında çok sayıda deneme uçuşu yapılabilir. Kullanıcının ihtiyaçları da göz önünde bulundurarak ürün maliyetini düşürmek için daha düşük performanslı mikroişlemci kullanılabilir.

Üretilen uçuş kontrol kartının ilk prototip olması nedeniyle donanımsal eksikliklerin bulunması oldukça doğaldır. Özellikle güvenlik açısından acil durum butonu ve motorları ARM durumuna geçirmeden önce basılması gereken bir güvenlik butonu gibi önemli özelliklerin eksik olması, potansiyel riskleri artırabilir. Bu eksikliklerin giderilmesi için aşağıdaki adımlar önerilebilir.

Acil Durum Butonu Eklenmesi: Sisteme acil durum butonu eklenerek, beklenmedik durumlarda İHA'nın hızlı bir şekilde durdurulması sağlanabilir. Acil durum butonu, ani tehlikeler veya sistem hataları durumunda İHA'nın kontrolünün kaybedilmesini önlemek için hayati öneme sahiptir.

Motorları Arm Durumuna Geçirmeden Önce Basılması Gereken Güvenlik Butonu Eklenmesi: Motorları arm durumuna geçirmeden önce basılması gereken bir güvenlik butonu eklenerek, İHA'nın istenmeyen bir şekilde hareket etmesini önleyebilir. Bu butonun basılması, operatörün İHA'nın kontrolünü almadan önce sistemin kontrolünün güvence altına alınmasını sağlar.

KAYNAKLAR

Ahn, K. and Truong, D. Q., 2009, Online tuning fuzzy PID controller using robust extended Kalman filter, Journal of Process Control, 19 (6): 1011–1023.

Akyüz, S., 2013, Dört Rotorlu İnsansız Hava Aracı (Quadrotor)'ın PD ve Bulanık Kontrolcü Tasarımı ve Benzetim Uygulaması, Yüksek Lisans Tezi*, Ege Üniversitesi, Fen Bilimleri Enstitüsü*, İzmir.

Altın, C. and Er, O., 2015, Complementary Filter Application for Inertial Measurement Unit. Electronic Letters on Science and Engineering, 11(2), 20-25.

Demir, B. E., Duran, F., and Bayir, R., 2016, "Real-time trajectory tracking of an unmanned aerial vehicle using a self-tuning fuzzy proportional integral derivative controller", International Journal of Micro Air Vehicles, 8 (4): 252–268.

Doğan, İ., PIC C ile Sıcaklık Projeleri. Bilişim Yayınları, İstanbul (2003) Altınbaşak, O., Mikrodenetleyiciler ve PIC Programlama. Altaş Yayınları, İstanbul.

Ekmen, M. İ., Aydoğdu, Ö., 2020, İnsansız Hava Araçları İçin Görüntü İşleme Tabanlı Otonom İniş. Avrupa Bilim ve Teknoloji Dergisi, (Özel Sayı), 297-303.

Falanga, D., Floreano, D., Kleber, K., Mintchev, S., 2018, Scaramuzza, D. (2018) The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly. IEEE Robotics and Automation Letters. Preprint Version. Accepted November.

Fantoni, I., Lozano R., 2002, Nonlinear control for underactuated mechanical systems Springer, Communications and Control Engineering Series.

Fourati, H., Manamanni, N., 2013, Position Estimation Approach by Complementary Filteraided IMU for Indoor Environment.

Güçlü, A., 2012, "Attitude and Altitude Control of an Outdoor Quadrotor", Yüksek Lisans Tezi, Atılım Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

Gürgöze, G., ve Türkoğlu, İ., 2022, Mobil robotlarda kullanılan DC motorların parametrelerinin belirlenmesi için deney düzeneği geliştirilmesi, Politeknik Dergisi.

Huang, H., Tomlin, J., Waslander, L., and Hoffmann, M., 2009, Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering, IEEE International Conference on Robotics and Automation, 3277–3282.

Ibrahim, B., Hat, M., Hassan, M., Mohd, T., 2012, MODEL BASED DESIGN OF PID CONTROLLER.

Ismail, M., Cao Yihua, Abu Bakar and Zhenlong Wu, March 2013., "Aerodynamic Efficiency Study of 2D Airfoils and 3D Rectangular Wing in Heavy Rain Via Two-Phase Flow Approach", Journal of Aerospace Engineering.

Kahveci, M., ve Can, N., 2017, İnsansız hava araçların: Tarihçesi, tanımı, dünyada ve Türkiye'deki yasal durumu, Selçuk Üniversitesi Mühendislik Bilim ve Teknik Dergisi.

Karaahmetoğlu, A., 2000, Dört Rotorlu Hava Aracının Durum Kestirimine Dayalı LQG ile Yörünge Kontrolü. İstanbul Teknik Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü Bitirme Tasarım Projesi, İstanbul, Türkiye.

Keçecioğlu, Ö. F., Açıkgöz, H., Güneş, M., and Şekkeli, M., 2015, Öz ayarlamalı bulanık-PID denetleyici ile hidrolik türbinin benzetim çalışması, Academic Platform Journal of Engineering and Science, (1): 7–15.

Köse, O., Oktay, T., 2021, İnovatif Yöntemlerle Kuadkopter Modellenmesi, Kontrolü ve Gerçek Zamanlı Uygulamaları. Doktora Tezi, Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, KAYSERİ.

Liu, Y. J., Gao Y., 2016, Adaptive Fuzzy Optimal Control using Direct Heuristic Dynamic Programming for Chaotic Discrete-Time System. Journal of Vibration and Control, 22 (2), 595-603.

Mettler B. and Kundak, N., 2007, Experimental Framework for Evaluating Autonomous Guidance and Control Algorithms for Agile Aerial Vehicles, European Control Conference.

Nice, B. E., 2004, Design of a Four Rotor Hovering Vehicle, Yüksek Lisans Tezi, Cornell University.

Okudan, M. E., 2019, Navigasyon hassasiyetini arttırmak için ataletsel ölçüm birimine tamamlayıcı filtre uygulanması. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, *Fen Bilimleri Enstitüsü,* İstanbul.

Özer, F., 2016, Android İşletim Sistemli Cihazlar ve 2.4GHz Kumanda ile ARM Tabanlı Mikrodenetleyici üzerinden Quadrotor İnsansız Hava Aracının Tasarımı, Otonom Kontrolü ve Denetimi, Karabük.

Pounds, E., Bersak, R., Dollar, M., 2012, Stability of Small-Scale UAV Helicopters and Quadrotors with Added Payload Mass under PID Control. Autonomous Robots, 33 (1-2), 129-142.

Pezzuolo, A., Chiumenti, A., Marinello, F., Sartori, L., 2016, Technical analysis of unmanned aerial vehicles (drones) for agricultural applications. Engineering for Rural Development.

Pınar, A., Uğur, K., Serkan, Ö., Melin, Ş., Ercan G., Yosheph, Y., Yavuz, Y., 2012, "Büyük Oranda Şekil Değiştirebilen Kanat Yüzeylerinin Aerodinamik Yükler Altındaki Davranışları", Combined Morphing Assesment Software Using Flight Envelope Data and Mission Based Morphing Prototype Wing Development Project / CHANGE.

Restas, A., Bodnar, L., Qiang, X., 2018, Conceptual Approach of Measuring the Profession and Economic Effectiveness of Drone Applications Supporting Forest fir Management, Procedia Engineering.

Tavakol, F., Binazadeh, T., 2017, Robust Control Design for Path Tracking of Non-Affine UAV. Systems Science and Control Engineering, 5 (1), 474-480.

Uğurlu, B., ve Özçınar, E. C., 2021, “Tepki kuvveti gözetleyici tabanlı tork kontrolü”,Politeknik Dergisi.

Vervoorst, W.J., 2016, A Modular Simulatıon Envıronment For The Improved Dynamic Simulation Of Multirotor Unmanned Aerıal Vehicles”, Master Thesis, University of Illinois at Urbana- Champaign.

Welch, G. and Bishop G., Jack, K., 2004, Handbook for the Digital Engineer, Burlington, USA.An Introduction To The Kalman Filter, University of North Carolina, North Carolina.

Yusuf B., 2012, “Bir Sesaltı Rüzgâr Tüneli İçerisine Yerleştirilmiş NACA0012 Kanadının Sayısal Analizi”, Dokuz Eylül Üniversitesi Mühendislik Fakültesi Makina Mühendisliği Bölümü Bitirme Projesi, Haziran.

EKLER

EK-1 STM kodu

```
#include "main.h"
#include "stdio.h"

#define MAX_SIGNAL 2000
#define MIN_SIGNAL 1000
#define MOTOR_PIN 9
int DELAY = 2000;


TIM_HandleTypeDef htim1;
UART_HandleTypeDef huart3;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
   HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
      HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);

      __HAL_TIM_SET_COMPARE(&htim1 , TIM_CHANNEL_1, MAX_SIGNAL);
      HAL_DELAY(delay);

      __HAL_TIM_SET_COMPARE(&htim1 , TIM_CHANNEL_1, MIN_SIGNAL);
      HAL_DELAY(delay);

  while (1)
  {
  }
}

static void MX_TIM1_Init(void)
{

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

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 23;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 19999;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
```

```
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1)
!= HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_2)
!= HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_3)
!= HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_4)
!= HAL_OK)
  {
    Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput =
TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  HAL_TIM_MspPostInit(&htim1);

}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
```

```
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }
}

#ifdef  USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{
}
#endif
```

EK-2 İMU ham değerleri okuması kodu

```
//MPU sensrou ham kutuphaneleri tanitimi

#define MPU6050_ADDR 0xD0
#define SMPLRT_DIV_REG 0x19
#define GYRO_CONFIG_REG 0x1B
#define ACCEL_CONFIG_REG 0x1C
#define ACCEL_XOUT_H_REG 0x3B
#define TEMP_OUT_H_REG 0x41
#define GYRO_XOUT_H_REG 0x43
#define PWR_MGMT_1_REG 0x6B
#define WHO_AM_I_REG 0x75

float Acceleration_angle[2];
float Gyro_angle[2];
float Total_angle[2];
float AngX, AngY;

int F = 0;
int A,A1,A2  = 0;


int32_t Acc_rawX,Acc_rawY,Acc_rawZ ;
int32_t Gyr_rawX,Gyr_rawY,Gyr_rawZ ;


int16_t Accel_X_RAW = 0;
int16_t Accel_Y_RAW = 0;
int16_t Accel_Z_RAW = 0;

int16_t Gyro_X_RAW = 0;
int16_t Gyro_Y_RAW = 0;
int16_t Gyro_Z_RAW = 0;
```

```
float Ax, Ay, Az, Gx, Gy, Gz;
int32_t AX1=0;
int f=0;
int AX2=0;
int32_t Axx = 0 ;
int16_t pwm = 0;


void MPU6050_Init (void)
{
  uint8_t check;
  uint8_t Data;

  // check device ID WHO_AM_I

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR,WHO_AM_I_REG,1, &check, 1, 1000);

  if (check == 104)  // 0x68 will be returned by the sensor if everything goes well
  {
    // power management register 0X6B we should write all 0's to wake the sensor up
    Data = 0;
    HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, PWR_MGMT_1_REG, 1,&Data, 1, 1000);

    // Set DATA RATE of 1KHz by writing SMPLRT_DIV register
    Data = 0x07;
    HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, SMPLRT_DIV_REG, 1, &Data, 1, 1000);

    // Set accelerometer configuration in ACCEL_CONFIG Register
```

```
void MPU6050_Read_Gyro (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from GYRO_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Gyro_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Gyro_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Gyro_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /* convert the RAW values into dps (°/s)
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 131.0
       for more details check GYRO_CONFIG Register              **/

  Gx = Gyro_X_RAW/131.0;
  Gy = Gyro_Y_RAW/131.0;
  Gz = Gyro_Z_RAW/131.0;
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
void MPU6050_Read_Gyro (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from GYRO_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Gyro_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Gyro_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Gyro_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /* convert the RAW values into dps (°/s)
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 131.0
       for more details check GYRO_CONFIG Register              **/

  Gx = Gyro_X_RAW/131.0;
  Gy = Gyro_Y_RAW/131.0;
  Gz = Gyro_Z_RAW/131.0;
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
```

```
    // XG_ST=0,YG_ST=0,ZG_ST=0, FS_SEL=0 -> ± 250 °/s
    Data = 0x00;
    HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, GYRO_CONFIG_REG, 1, &Data, 1, 1000);
  }

}

void MPU6050_Read_Accel (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from ACCEL_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, ACCEL_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Accel_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Accel_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Accel_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /* convert the RAW values into acceleration in 'g'
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 16384.0
       for more details check ACCEL_CONFIG Register              **/

  Ax = Accel_X_RAW/16384.0;
  Ay = Accel_Y_RAW/16384.0;
  Az = Accel_Z_RAW/16384.0;
}

void MPU6050_Read_Gyro (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from GYRO_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Gyro_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Gyro_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Gyro_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /* convert the RAW values into dps (°/s)
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 131.0
       for more details check GYRO_CONFIG Register              **/

  Gx = Gyro_X_RAW/131.0;
  Gy = Gyro_Y_RAW/131.0;
  Gz = Gyro_Z_RAW/131.0;
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
```

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

  /* USER CODE END 1 */

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

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

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

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

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_ADC1_Init();
  /* USER CODE BEGIN 2 */
  //icinde sensr baslmasi
MPU6050_Init();
  /* USER CODE END 2 */
  /* USER CODE BEGIN WHILE */
  //while icinde sensr calmasi
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    MPU6050_Read_Accel();
    MPU6050_Read_Gyro();


     }

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

EK-3 MPU6050 ile PID Kontrol C Kodu

```
//////////////////// pid

float elapsedTime, time, timePrev;
float rad_to_deg = 180/3.141592654;

float PID, error, previous_error;

float pid_p=0;
float pid_i=0;
float pid_d=0;

double kp=3.00;//3.55
double ki=0.00;//0.005
double kd=1.0;//2.05

double throttle = 1150; //initial value of throttle to the motors

float desired_angle = 5; //This is the angle in which we whant the

float Acceleration_angle[2];
float Gyro_angle[2];
float Total_angle[2];
//////////////////// pid
int but;
int32_t Acc_rawX,Acc_rawY,Acc_rawZ ;
int32_t Gyr_rawX,Gyr_rawY,Gyr_rawZ ;


int16_t Accel_X_RAW = 0;
int16_t Accel_Y_RAW = 0;
int16_t Accel_Z_RAW = 0;
```

```
int i =0;
int q =0;

int32_t Axy = 0 ;
int32_t Axx = 0 ;
int32_t Axz = 0 ;

int32_t  Axxl = 0;
int32_t  Axyl = 0;
int32_t  Axzl = 0;

float egy =0.2;
int16_t Gyro_X_RAW = 0;
int16_t Gyro_Y_RAW = 0;
int16_t Gyro_Z_RAW = 0;

int32_t Gxy = 0 ;
int32_t Gxx = 0 ;
int32_t Gxz = 0 ;

int32_t  Gxxl = 0;
int32_t  Gxyl = 0;
int32_t  Gxzl = 0;

float Ax, Ay, Az, Gx, Gy, Gz;


int32_t ADC =0;
int32_t ADCy =0;
int16_t ADCx =0;
int16_t ADCz =0;
```

```
int16_t LMPWM  = 1000;
int16_t RMPWM = 1000;

ADC_ChannelConfTypeDef sConfigPrivate = {0};


void  Read_ADC()
{
    sConfigPrivate.Rank = ADC_REGULAR_RANK_1;
    sConfigPrivate.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
    sConfigPrivate.Channel = ADC_CHANNEL_0;
    HAL_ADC_ConfigChannel(&hadc1, &sConfigPrivate);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,1000);
    ADC = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);

}

uint32_t MAP(uint32_t au32_IN, uint32_t au32_INmin, uint32_t au32_INmax, uint32_t au32_OUTmin, uint32_t au32_OUTmax)
{
    return ((((au32_IN - au32_INmin)*(au32_OUTmax - au32_OUTmin))/(au32_INmax - au32_INmin)) + au32_OUTmin);
}
```

```
void MPU6050_Read_Accel (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from ACCEL_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, ACCEL_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Accel_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Accel_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Accel_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /*** convert the RAW values into acceleration in 'g'
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 16384.0
       for more details check ACCEL_CONFIG Register              ****/

  Ax = Accel_X_RAW/16384.0;
  Ay = Accel_Y_RAW/16384.0;
  Az = Accel_Z_RAW/16384.0;
}



void MPU6050_Read_Gyro (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from GYRO_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, 1000);
```

```
  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, ACCEL_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Accel_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Accel_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Accel_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);

  /*** convert the RAW values into acceleration in 'g'
       we have to divide according to the Full scale value set in FS_SEL
       I have configured FS_SEL = 0. So I am dividing by 16384.0
       for more details check ACCEL_CONFIG Register              ****/

  Ax = Accel_X_RAW/16384.0;
  Ay = Accel_Y_RAW/16384.0;
  Az = Accel_Z_RAW/16384.0;
}



void MPU6050_Read_Gyro (void)
{
  uint8_t Rec_Data[6];

  // Read 6 BYTES of data starting from GYRO_XOUT_H register

  HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6, 1000);

  Gyro_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data [1]);
  Gyro_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data [3]);
  Gyro_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data [5]);
```

```
  Gx = Gyro_X_RAW/131.0;
  Gy = Gyro_Y_RAW/131.0;
  Gz = Gyro_Z_RAW/131.0;
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  time = HAL_GetTick();


  /* USER CODE END 1 */

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

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_I2C1_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
   MPU6050_Init();

  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
     while (q < 1)
    {
    LMPWM = 1000;
    RMPWM = 1000;
    __HAL_TIM_SET_COMPARE(&htim2 , TIM_CHANNEL_2, RMPWM);
    __HAL_TIM_SET_COMPARE(&htim2 , TIM_CHANNEL_1, LMPWM);
    HAL_Delay(6000);
    q++;
    }

    /////// sensor

    MPU6050_Read_Accel();
    MPU6050_Read_Gyro();
```

```
while( i<=5)
{
MPU6050_Read_Accel();
MPU6050_Read_Gyro();

Axx += Accel_X_RAW;
Axy += Accel_Y_RAW;
Axz += Accel_Z_RAW;

Gxx += Gyro_X_RAW;
Gxy += Gyro_Y_RAW;
Gxz += Gyro_Z_RAW;

i++;
}

Axxl = Axx/6;
Axyl = Axy/6;
Axzl = Axz/6;

Gxxl = Gxx/6;
Gxyl = Gxy/6;
Gxzl = Gxz/6;

Acc_rawX = ((Axxl *egy)+(Acc_rawX*(1-egy)));
Acc_rawY = ((Axyl *egy)+(Acc_rawY*(1-egy)));
Acc_rawZ = ((Axzl *egy)+(Acc_rawZ*(1-egy)));

Gyr_rawX = ((Axxl *egy)+(Gyr_rawX*(1-egy)));
Gyr_rawY = ((Axyl *egy)+(Gyr_rawY*(1-egy)));
Gyr_rawZ = ((Axzl *egy)+(Gyr_rawZ*(1-egy)));
```

```
///// PWM
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);

timePrev = time;  // the previous time is stored before the actual time read
time = HAL_GetTick();  // actual time read
elapsedTime = (time - timePrev) / 1000;


/*---X---*/
Acceleration_angle[0] = atan((Acc_rawY/16384.0)/sqrt(pow((Acc_rawX/16384.0),2) + pow((Acc_rawZ/16384.0),2)))*rad_to_deg;
/*---Y---*/
Acceleration_angle[1] = atan(-1*(Acc_rawX/16384.0)/sqrt(pow((Acc_rawY/16384.0),2) + pow((Acc_rawZ/16384.0),2)))*rad_to_deg;

Gyro_angle[0] = Gyr_rawX/131.0;
/*---Y---*/
Gyro_angle[1] = Gyr_rawY/131.0;

/*---X axis angle---*/
Total_angle[0] = 0.98 *(Total_angle[0] + Gyro_angle[0]*elapsedTime) + 0.02*Acceleration_angle[0];
/*---Y axis angle---*/
Total_angle[1] = 0.98 *(Total_angle[1] + Gyro_angle[1]*elapsedTime) + 0.02*Acceleration_angle[1];


error = Total_angle[1] - desired_angle;

pid_p = kp*error;

if(error > -3 && error < 3)
{
  pid_i = pid_i+(ki*error);
}

pid_d = kd*((error - previous_error)/elapsedTime);
```

```
  PID = pid_p + pid_i + pid_d;


  if(PID < -1000)
  {
    PID=-1000;
  }
  if(PID > 1000)
  {
    PID=1000;
  }

  LMPWM = throttle + PID;
  RMPWM = throttle - PID;

  if(RMPWM < 1020)
  {
    RMPWM= 1020;
  }
  if(RMPWM > 1350)
  {
    RMPWM=1350;
  }
  //Left
  if(LMPWM < 1020)
  {
    LMPWM= 1020;
  }
  if(LMPWM > 1350)
  {
    LMPWM=1350;
  }

  previous_error = error;

 if (i > 5)
 {
 i = 0;
 ADCy =0;
 Axx = 0;
 Axy = 0;
 Axz = 0;
 Gxx = 0;
 Gxy = 0;
 Gxz = 0;
 }
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
 __HAL_TIM_SET_COMPARE(&htim2 , TIM_CHANNEL_2, RMPWM);
 __HAL_TIM_SET_COMPARE(&htim2 , TIM_CHANNEL_1, LMPWM);

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

EK-4 MPU6050 Sensörün Filtre Kullanarak veriler okuma kodu

```
//////////////////////////////////////////////IMU
float Acceleration_angle[2];
float Gyro_angle[2];
float Total_angle[2];

int32_t Acc_rawX,Acc_rawY,Acc_rawZ ;
int32_t Gyr_rawX,Gyr_rawY,Gyr_rawZ ;


int16_t Accel_X_RAW = 0;
int16_t Accel_Y_RAW = 0;
int16_t Accel_Z_RAW = 0;

float x,yaw,yaw_angle, kroll, kpitch;

//////////////////////////////////////////////IMU END
```

```
MPU6050_t MPU6050;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

float AngX, AngY;

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

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

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

  /* USER CODE BEGIN Init */
```

```
MX_GPIO_Init();
MX_LPUART1_UART_Init();
MX_I2C1_Init();
MX_I2C2_Init();
/* USER CODE BEGIN 2 */
MPU6050_Init(&hi2c1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  MPU6050_Read_All(&hi2c1, &MPU6050);

  AngX = MPU6050.KalmanAngleX;
  AngY = MPU6050.KalmanAngleY;

  HAL_Delay (500);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
////////////////////////////////////////////////// FILTRE
char str[20];

int32_t Axy = 0 ;
int32_t Axx = 0 ;
int32_t Axz = 0 ;

int32_t  Axxl = 0;
int32_t  Axyl = 0;
int32_t  Axzl = 0;

float egy =0.2;
int16_t Gyro_X_RAW = 0;
int16_t Gyro_Y_RAW = 0;
int16_t Gyro_Z_RAW = 0;

int32_t Gxy = 0 ;
int32_t Gxx = 0 ;
int32_t Gxz = 0 ;

int32_t  Gxxl = 0;
int32_t  Gxyl = 0;
int32_t  Gxzl = 0;

float Ax, Ay, Az, Gx, Gy, Gz;

int32_t ADC =0;
int32_t ADCy =0;
int16_t ADCx =0;
int16_t ADCz =0;

////////////////////////////////////////////////// FILTRE END
```