İSTANBUL TOPKAPI ÜNİVERSİTESİ

Mobil Uygulama ile Drone Kontrolü

Yazılım Mühendisliği 3. Sınıf

Yusuf Utku Öztürk 22040301013

Ahmet Buğra Kaplan 22040301028

Emirhan Sertel 22040301064

Alper Ceylan 22040301071

Danışman: Doç. Dr. Cevat Rahebi

12 Haziran 2025

İçindekiler

Özet

Bu projede, dört motorlu bir insansız hava aracı (drone) tasarlanarak, Pixhawk uçuş kontrol kartı ve ESP8266 Wi-Fi modülü kullanımıyla uzaktan kontrol edilebilir yenilikçi bir sistem geliştirilmiştir. Projede öncelikle drone'un mekanik ve elektronik bileşenleri bir araya getirilmiş, ardından Pixhawk uçuş kontrol kartı aracılığıyla stabil ve güvenli bir uçuş sağlanması hedeflenmiştir. Drone'un uzaktan kontrolü, Python programlama diliyle geliştirilen özel bir API üzerinden gerçekleştirilmekte olup, bu API'nin aktif hale gelmesiyle birlikte, aynı Wi-Fi ağına bağlı olan mobil cihazlar ile drone arasında çift yönlü haberleşme mümkün olmaktadır. Python tabanlı bu arayüz, komutların hızlı ve güvenilir biçimde Pixhawk kontrol kartına iletilmesine olanak tanır.

Mobil uygulama ise Kotlin programlama diliyle geliştirilmiş ve kullanıcı dostu bir arayüzle donatılmıştır. Uygulama üzerinden kullanıcılar, drone'un kalkış, iniş, yönlendirme ve hız gibi temel kontrollerini gerçek zamanlı olarak gerçekleştirebilmektedir. ESP8266 Wi-Fi modülü sayesinde, kablosuz bağlantı kurularak mobil cihazlar ile Pixhawk arasındaki iletişim sorunsuz biçimde sağlanmakta ve böylece uçuş esnasında anlık kontrol ve izleme imkânı sunulmaktadır. Projede, uçuş kontrol sistemlerinin, kablosuz haberleşmenin ve mobil yazılımların etkin bir şekilde entegrasyonu üzerinde durulmuş ve bu bileşenlerin bir arada çalışabilirliği başarıyla gösterilmiştir.

Sonuç olarak, bu çalışma, insansız hava aracı teknolojilerinde donanım ve yazılım bileşenlerinin bütünleşik bir şekilde kullanılmasına dair uygulamalı bir örnek teşkil etmektedir. Pixhawk uçuş kontrol kartı, ESP8266 Wi-Fi modülü, Python tabanlı API ve Kotlin ile geliştirilen mobil uygulama arasında kurulan bu iletişim ağı, ileri düzeyde bir uzaktan kontrol deneyimi sunmakta ve gelecekteki benzer projeler için kapsamlı bir referans niteliği taşımaktadır.

***Anahtar Kelimeler**: Drone, Esp8266, Mobil, Python API*

1. Giriş

Son yıllarda insansız hava araçlarının (drone) uzaktan ve akıllı sistemlerle kontrolüne yönelik ilgi önemli ölçüde artmıştır. Özellikle mobil uygulamalar aracılığıyla yapılan temassız ve kablosuz kontrol yöntemleri, hem erişilebilirlik hem de kullanıcı deneyimi açısından klasik kumanda tabanlı sistemlere güçlü bir alternatif sunmaktadır. Bu proje, Wi-Fi tabanlı haberleşme

altyapısı ile geliştirilen, gerçek zamanlı ve düşük maliyetli bir insansız hava aracı kontrol sistemi oluşturmayı hedeflemektedir.

Sistem, mobil cihazlar üzerinde çalışan ve Kotlin programlama diliyle geliştirilen özel bir uygulama ile kullanıcıya sezgisel bir kontrol arayüzü sunar. Kullanıcıdan alınan komutlar, Python tabanlı bir API üzerinden, Wi-Fi bağlantısı kurabilen ESP8266 modülü aracılığıyla drone'un uçuş kontrol kartı olan Pixhawk'a iletilmektedir. Böylece, geleneksel joystick veya fiziksel uzaktan kumanda ihtiyacı ortadan kalkmakta, modern akıllı cihazlar üzerinden daha esnek ve erişilebilir bir uçuş deneyimi mümkün olmaktadır. Temel amaç, kablosuz ve dijital altyapı sayesinde kullanıcıların drone'u güvenli, hızlı ve verimli şekilde yönlendirebilmesini sağlamaktır.

Projede, mobil uygulama üzerinden kalkış, iniş, yön değiştirme ve hız ayarlama gibi temel uçuş komutlarının gerçek zamanlı iletimi hedeflenmiştir. Bu komutların Wi-Fi ağı üzerinden ESP8266 aracılığıyla iletilmesi sayesinde, uçuş sırasında düşük gecikmeli ve güvenilir bir kontrol sağlanmaktadır. Haberleşme sürecinde, Python API sayesinde komutlar doğrulanmakta ve yalnızca geçerli talimatlar Pixhawk uçuş kontrol kartına aktarılmaktadır. Böylece olası iletişim hataları ve yanlış yönlendirme riskleri minimuma indirilmektedir.

Ek olarak, sistemin modüler yazılım mimarisi, ileride farklı sensör veya kontrol seçeneklerinin kolaylıkla entegre edilebilmesine olanak sunar. Donanım ve yazılım bileşenlerinin birbiriyle uyumlu çalışması, hem kullanıcı güvenliğini hem de sistem kararlılığını ön planda tutmaktadır. Sonuç olarak, bu proje; kablosuz haberleşme ve mobil uygulama desteği ile donatılmış yeni nesil bir drone kontrol teknolojisi geliştirmekte ve klasik fiziksel kontrol yöntemlerine yenilikçi, düşük maliyetli ve uygulanabilir bir alternatif sunmaktadır. Geliştirilen sistem, başta insansız hava araçları olmak üzere, akıllı otomasyon, uzaktan erişimli robotik uygulamalar ve eğitim alanlarında geniş bir kullanım potansiyeline sahiptir.

2. Amaç ve Kapsam

Amaç

Bu projenin temel amacı, Wi-Fi üzerinden kablosuz haberleşme altyapısı kullanılarak, mobil cihazlar aracılığıyla insansız hava aracı (drone) üzerinde gerçek zamanlı ve kullanıcı dostu bir kontrol sistemi geliştirmektir. Geliştirilen sistem, klasik RC kumandalar yerine, modern mobil uygulamalar ve kablosuz veri aktarım teknolojilerinden yararlanarak drone kontrolünü daha

erişilebilir ve esnek hale getirmektedir. Kullanıcılar, Kotlin diliyle geliştirilen mobil uygulama üzerinden, aynı ağda bulundukları drone'a anlık komutlar gönderebilmekte ve uçuşu canlı olarak yönlendirebilmektedir. ESP8266 Wi-Fi modülü ile kurulan bağlantı, mobil uygulamadan gelen komutların Python tabanlı API aracılığıyla Pixhawk uçuş kontrol kartına aktarılmasını sağlamakta, böylece hızlı ve güvenilir bir haberleşme altyapısı oluşturulmaktadır. Bu sayede, drone kontrol süreçlerinin dijitalleştirilmesi ve mobil platformlar üzerinden yönetilmesi amaçlanmıştır.

Kapsam

Proje, mobil cihazlardan gönderilen uçuş komutlarının (kalkış, iniş, yön değiştirme, hız ayarlama vb.) Wi-Fi üzerinden drone'a aktarılması üzerine odaklanmaktadır. Mobil uygulama, kullanıcıya sezgisel ve pratik bir arayüz sunarken; uygulama ile drone arasındaki veri iletişimi, ESP8266 Wi-Fi modülü üzerinden sağlanmakta ve Python ile geliştirilen özel bir API aracılığıyla Pixhawk kontrol kartına iletilmektedir. Sistem, sadece tek bir drone'un temel uçuş komutlarının mobil cihaz üzerinden kontrol edilmesiyle sınırlıdır ve GPS ile konum takibi, otomatik uçuş rotası, kamera entegrasyonu veya çoklu drone yönetimi gibi ileri seviye fonksiyonlar bu çalışmanın kapsamı dışında tutulmuştur. Proje sürecinde hem donanım (ESP8266, Pixhawk, dört motorlu drone bileşenleri) hem de yazılım (Python API, Kotlin tabanlı mobil uygulama) geliştirmeleri yapılmış, tüm sistemin birlikte uyumlu ve güvenilir şekilde çalışması hedeflenmiştir. Bu kapsamda elde edilen sistem, kablosuz haberleşme, uçuş kontrolü ve mobil yazılım entegrasyonu konusunda bütünleşik ve uygulanabilir bir örnek ortaya koymaktadır.

3. Materyal ve Metotlar

Bu bölümde, projenin gerçekleştirilmesinde kullanılan donanım ve yazılım bileşenleri ayrıntılı olarak tanıtılacak; sistemin genel mimarisi, haberleşme altyapısı ve geliştirilen mobil uygulamanın işlevsel yapısı adım adım açıklanacaktır.

3.1 Donanım

Projede kullanılan tüm donanımlar, uçuş stabilitesi, maliyet ve enerji verimliliği ve kablosuz bağlantı gereksinimleri göz önünde bulundurularak seçilmiştir.

Donanım bileşenleri arasında, drone'un taşıyıcı gövdesi olarak F450 drone frame'i, dört adet A2212 fırçasız motor ve bu motorların kontrolü için dört adet 30A ESC (Electronic Speed Controller) bulunmaktadır. Uçuşun hassas şekilde yönetilebilmesi için Pixhawk uçuş kontrol

kartı ve ona bağlı bir GPS modülü kullanılmıştır. Sisteme enerji sağlayan ana birim ise 4200 mAh kapasiteli 3S Lipo bataryadır. Kablosuz haberleşme altyapısı için ise ESP8266 Wi-Fi modülü tercih edilmiş olup, sistemin iletişim ihtiyaçlarını düşük gecikme ve yüksek güvenilirlikle karşılamaktadır. Ayrıca kamera görüntüsü aktarımı için ESP32-CAM modülü projeye entegre edilmiş, ancak bu çalışmada aktif olarak kullanılmamıştır.

Bu bölümde, projede yer alan merkezi ve önemli bileşenler; teknik özellikleri ve sistem içindeki işlevleriyle birlikte ayrıntılı olarak tanıtılmaktadır.

3.1.1. Pixhawk Uçuş Kontrol Kartı

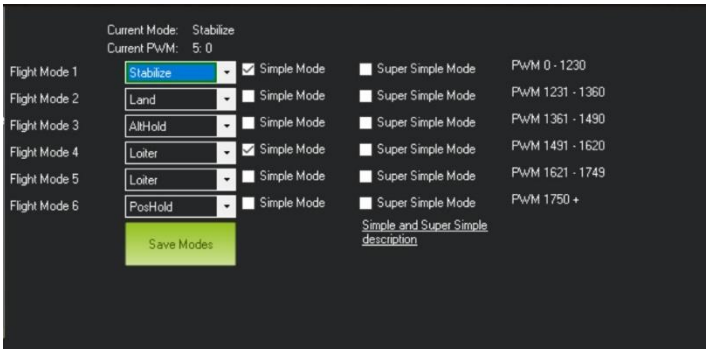
Pixhawk(1) uçuş kontrol kartı, açık kaynaklı donanım ve yazılım desteğine sahip, çok yönlü ve yüksek performanslı bir otopilot platformudur. Gelişmiş sensör seti, genişletilebilir bağlantı portları ve esnek yazılım desteği sayesinde, hem amatör hem de profesyonel insansız hava aracı uygulamalarında yaygın olarak kullanılmaktadır. Pixhawk, entegre mikrodenetleyicisi (STM32F427 Cortex-M4 işlemci, 168 MHz hızında) ve hassas IMU (jüroskop, ivmeölçer, pusula ve barometre) birimleriyle uçuş sırasında yüksek doğrulukta veri toplama ve analiz imkânı sunar.

Şekil 1. Pixhawk Uçuş kartı

Bu projede Pixhawk, drone'un tüm uçuş dinamiklerinin yönetilmesinden sorumludur. ESC'lere iletilen motor hız kontrol sinyallerinin üretilmesi, denge ve yönelim kontrolü, GPS verisiyle konum sabitleme, ve uçuş modlarının (Stabilize, Alt Hold, Loiter, RTL gibi) yönetimi Pixhawk üzerinden sağlanmaktadır. Mobil uygulamadan veya Python tabanlı API'den gelen komutlar, ESP8266 modülü aracılığıyla Pixhawk'a iletilir ve burada uygun uçuş komutlarına dönüştürülerek drone'un anlık hareketi sağlanır.

Pixhawk'ın en önemli avantajlarından biri, farklı uçuş modları ve geniş çevre birimi desteğiyle esnek bir kontrol altyapısı sunmasıdır. Stabilize modu, manuel kontrol ve temel dengeyi sağlarken; Alt Hold modu, yükseklik koruması sunar. GPS destekli modlar (örneğin, Loiter ve RTL - Return to Launch) ise otonom uçuş ve eve dönüş gibi işlevlere imkân tanır. Pixhawk'ın bu yetenekleri, projenin güvenilir ve hassas bir şekilde uzaktan kontrol edilmesini mümkün kılar.

Pixhawk uçuş kontrol kartının sunduğu farklı uçuş modları, kullanıcıların deneyim seviyelerine ve uçuş ihtiyaçlarına göre drone üzerinde tam hakimiyet sağlamalarına olanak tanır. Özellikle Loiter modu, drone uçurmaya yeni başlayanlar için büyük kolaylık ve güvenlik sunar. Loiter modunda, drone GPS yardımıyla bulunduğu noktada ve yükseklikte sabit kalır; kullanıcı stickleri bıraktığında otomatik olarak pozisyonunu ve irtifasını korur. Bu sayede acemi kullanıcılar, karmaşık manevralar sırasında veya ani kontrol kaybı yaşadıklarında dronun sabitlenmesini sağlayarak riski en aza indirebilirler.



Pixhawk'ın öne çıkan başlıca uçuş modları şunlardır:

Loiter: GPS tabanlı konum ve irtifa sabitleme modudur. Kullanıcı komut vermediğinde drone bulunduğu konumu otomatik olarak korur. Özellikle yeni başlayanlar ve hassas pozisyon gerektiren görevler için idealdir.

AltHold (Altitude Hold): Bu modda drone, barometre sensörüyle mevcut yüksekliğini korur ancak yatay hareketlerde konum sabitlemesi yapılmaz. Pilot yalnızca irtifa sticki ile yükselme/alçalma kontrolünü yapar; diğer hareketlerde manuel uçuş gerçekleşir. AltHold, orta seviye kullanıcılar için hem kolaylık hem de temel uçuş alışkanlığı sağlar.

PosHold (Position Hold): Loiter'a benzer şekilde, hem GPS ile konum hem de barometre ile irtifa sabitlenir. Ancak PosHold modunda sticklere hafif müdahalelerde drone yavaşça hareket eder ve bırakıldığında yine sabit pozisyona döner. Hem yeni başlayanlar hem de deneyimli pilotlar için güvenli ve akıcı bir uçuş deneyimi sunar.

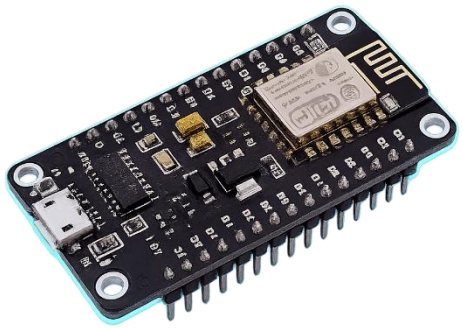
Land: İniş modudur. Drone bulunduğu konumda otomatik olarak kontrollü ve güvenli bir şekilde alçalarak iniş gerçekleştirir. Bu mod, manuel iniş hatalarını önlemeye yardımcı olur ve özellikle acil durumlarda emniyet sağlar.

Stabilize: Tamamen manuel uçuş modudur. Pilot, tüm eksenlerde drone'u doğrudan kontrol eder; sistem yalnızca temel dengeyi sağlar fakat pozisyon veya irtifa koruması sunmaz. Stabilize modu, uçuş tecrübesi kazanan kullanıcıların tam kontrol kabiliyetini geliştirmeleri için uygundur.

Sonuç olarak, Pixhawk'ın bu modları hem güvenli hem de esnek bir uçuş deneyimi sunarken, özellikle Loiter ve PosHold modları yeni başlayanlar için hata toleransı yüksek, öğrenme sürecini kolaylaştırıcı özellikler taşır. Her bir mod, farklı seviyedeki kullanıcıların ihtiyaçlarına yanıt verecek şekilde tasarlanmıştır.Sonuç olarak, Pixhawk uçuş kontrol kartı, projede uçuşun tüm temel yönetim, denge ve güvenlik süreçlerinden sorumlu çekirdek bileşen olarak görev yapmaktadır ve kablosuz komutların pratik biçimde uygulanabilmesi için gelişmiş bir altyapı sunmaktadır.

3.1.2. ESP-8266 Wifi Geliştirme Kartı

ESP8266, düşük maliyetli, kompakt ve enerji verimli bir Wi-Fi modülüdür. Üzerinde entegre TCP/IP protokolü ve 32-bit Tensilica L106 mikrodenetleyici barındıran ESP8266, kablosuz veri iletiminde esnek ve pratik bir çözüm sunar. 80 MHz'e kadar saat hızında çalışabilen bu modül, harici mikrokontrolcülere ihtiyaç duymadan Wi-Fi ağına doğrudan bağlanabilme yeteneğine sahiptir. Üzerinde yer alan GPIO pinleri sayesinde, sensör ve çeşitli çevresel birimlerle kolayca entegre edilebilir.



Şekil 2. ESP-8266

Bu projede ESP8266 modülü, mobil uygulama ile drone arasında kablosuz haberleşme köprüsü olarak görev yapmaktadır. Kullanıcıdan mobil uygulama aracılığıyla alınan uçuş komutları, Python tabanlı API üzerinden ESP8266'ya iletilir. ESP8266 ise bu komutları, Pixhawk uçuş kontrol kartına aktarmak için uygun protokol ve formatta iletir. Böylece tüm uçuş kontrolleri (kalkış, iniş, yönlendirme, hız değişimi vb.) Wi-Fi ağı üzerinden, düşük gecikme ile ve güvenilir şekilde gerçekleştirilebilmektedir.

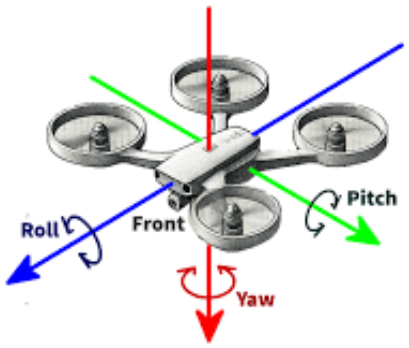
ESP8266'nın önemli avantajlarından biri, programlanabilir yapısı ve farklı modlarda (ör. Access Point, Station veya her ikisi) çalışabilmesidir. Projede kullanılan modül, sistemin Wi-Fi ağına dahil olmasını ve uzaktan gelen verileri alıp işleyebilmesini sağlar. Ayrıca HTTP veya TCP/IP protokolleri üzerinden veri alışverişi yapabilmesi sayesinde Python API ile sorunsuz bir entegrasyon sunar.

Sonuç olarak, ESP8266 Wi-Fi modülü, projede mobil cihaz ile uçuş kontrol kartı arasında güvenilir, hızlı ve kesintisiz bir kablosuz iletişim altyapısı sağlayan temel bileşen olarak kritik bir rol üstlenmektedir.

3.2. Yazılım ve Yöntemler

3.2.1. Drone Dengesi

İnsansız hava araçları (İHA'lar), modern teknoloji ile entegre edilmiş şekilde dengeli ve kontrollü uçabilmesi için bazı temel uçuş dinamiklerinin anlaşılması ve doğru biçimde yönetilmesi gerekmektedir. Bu dinamikler, temel olarak üç ana yönsel hareket etrafında gerçekleşir(Şekil 2.):



Şekil 3. Yönsel Terimler Görseli

Pitch: Drone'un ön veya arka kısmının yukarı veya aşağı hareket etmesini ifade eder. Yani basitçe ileri-geri eğilme hareketidir.

Roll: Drone'un sağ veya sol yana yatmasını ifade eder. Bu yönler drone'un sağa yattığında sağa gitmesini sola yattığında sola gitmesini aynı zamanda çapraz yöne de yönelmesini sağlar.

Yaw: Drone'un dikey ekseni etrafında dönmesi, yani yön değiştirmesi anlamına gelir. Drone'un ön tarafının nereye baktığının önemi roll ve pitch hareketlerini kullanırken çok önemi bulunur. Drone'un baktığı tarafa referans alınarak yön verilir.

Bir drone'un uçabilmesi için ilk olarak yerçekimine karşı koyması gerekir. Dört motorlu (quadrotor) bir drone'un havada sabit şekilde durduğu duruma hover (havada asılı kalma) denir. Hover durumunu sağlamak için, drone'un ağırlığı kadar bir kaldırma kuvveti üretmesi gerekir. Yerçekimi, drone'u aşağı doğru bir kuvvetle çeker. Drone'un her motoru yukarıya doğru bir itme kuvveti oluşturur. Dört motor birlikte çalıştığında bu kuvvetler toplanır ve aşağıdaki gibi formüle edilir:

$$F_{\text{motor\_toplam}} = F_1 + F_2 + F_3 + F_4$$

Hover durumunun gerçekleşebilmesi için şu koşul sağlanmalıdır:

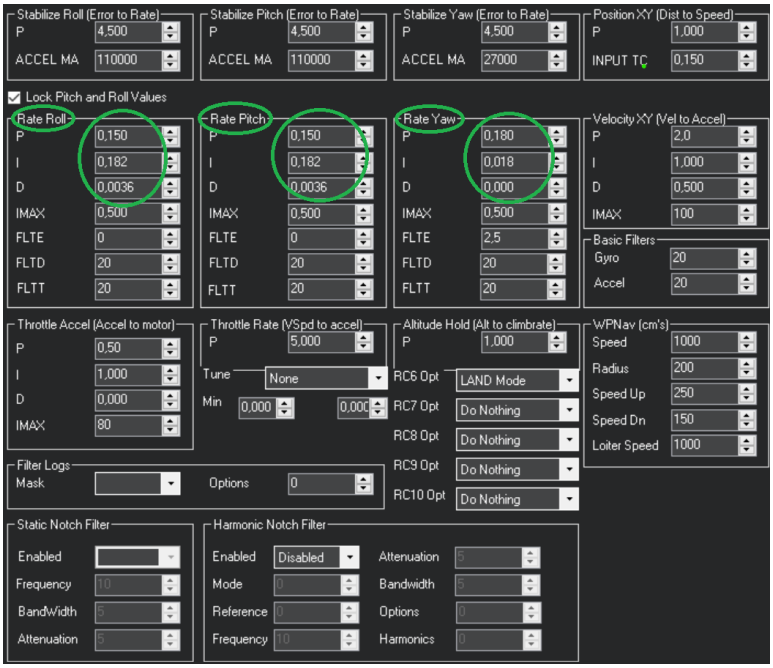
$$F_{\text{motor\_toplam}} = F_{\text{yercekimi}}$$

Yani motorların ürettiği toplam kaldırma kuvveti, yerçekiminin uyguladığı kuvvete eşit olmalıdır. Eğer bu kuvvet daha düşükse drone düşer, daha yüksekse yükselir. Bu dengeyi sağlamak, sistemin kontrol edilebilirliğinin temelidir.

Ancak yalnızca havada sabit kalmak yeterli değildir; drone'un yönünün ve dengesinin de korunması gerekir. İşte bu noktada PID kontrol sistemi devreye girer.

PID Kontrolcü ile Yönsel Dengeleme

PID, Proportional–Integral–Derivative (Oransal–Türevsel–İntegral) kelimelerinin baş harflerinden oluşur ve otomatik kontrol sistemlerinde kullanılan en yaygın algoritmalardan biridir. Bir sistemin hedeflenen (istenilen) değere ulaşmasını sağlamak amacıyla hata payını (gerçek değer - hedef değer farkı) sürekli olarak hesaplar ve bu hatayı minimize etmeye çalışır.



Drone'un dengede kalması için yönelme hareketlerini (pitch, roll, yaw) sürekli olarak kontrol etmek gerekir. Bunu sağlamak için PID, gerçek zamanlı olarak hata değerlerini hesaplayarak motorlara uygulanacak düzeltme sinyallerini üretir.

PID'in Bileşenleri:

P (Proportional – Oransal)

Hatanın büyüklüğüne göre anlık tepki üretir.

Hata büyükse, düzeltici sinyal de büyük olur.

Örnek: Drone öne doğru çok eğildiyse, arkadaki motorlara daha fazla güç vererek karşı koyar.

I (Integral – İntegral)

Zamana yayılan küçük hataların birikimini dikkate alır.

Sürekli küçük sapmaları zamanla sıfırlamaya çalışır.

Örnek: Drone sürekli hafif sola kayıyorsa, I bileşeni bunu algılayarak motorlara uzun vadeli düzeltme uygular.

D (Derivative – Türevsel)

Hatanın değişim hızını ölçer, yani hata ne kadar hızlı artıyor ya da azalıyorsa buna göre tepki verir.

Ani değişikliklere karşı sistemin "yavaşlamasını" sağlar, aşırı tepkiyi engeller.

Örnek: Drone bir yöne doğru aniden eğiliyorsa, D bileşeni bu eğilimin hızını görür ve ani düzeltme uygulayarak aşırı salınımı engeller.

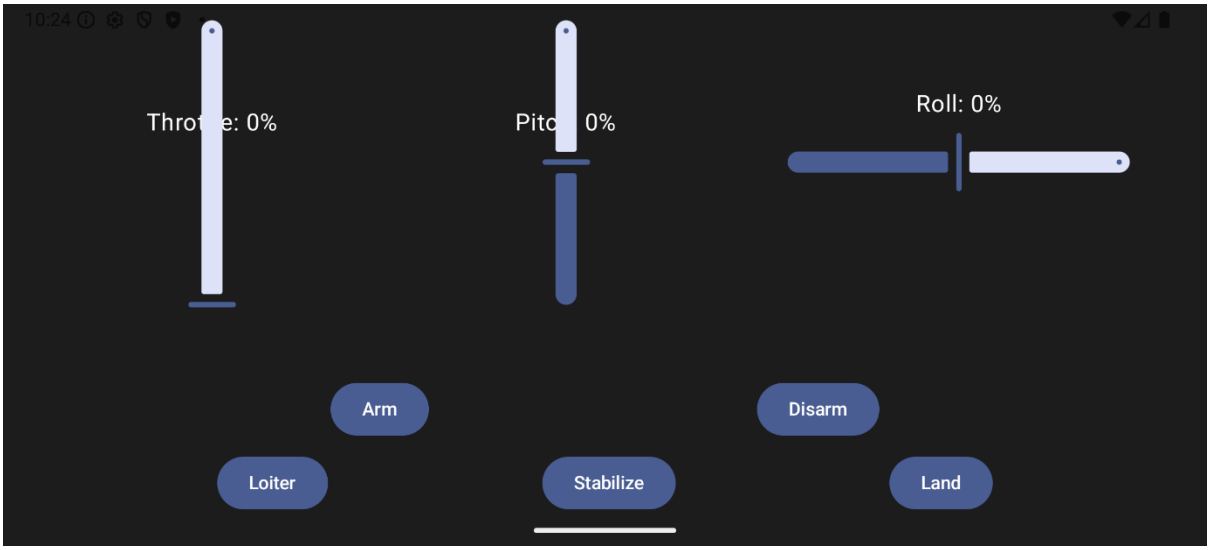
Bu terimlerin hepsinin detaylı matematiksel ifadeleri bulunmaktadır. Ancak bu raporda bu kadar detaylı teknik matematik bilgisine inmeden basit anlatım ile anlatılmıştır.

PID algoritması, her bir motorun hızını bu hareketlere göre ayarlayarak drone'un dengede kalmasını ve istenilen manevraların eksenlere göre gerçekleşmesini sağlar. PID parametrelerinin doğru biçimde ayarlanması, drone'un kararlı uçuş kabiliyetini doğrudan etkiler. Bu ayarlar genellikle deneysel olarak yapılmakta ve uçuş testleriyle optimize edilmektedir. Ayrıca pixhawk gibi uçuş kontrol kartlarında bulunan AutoTune modu ile uçuş yapıldıktan sonra otomatik olarak 3 eksen için PID ayarları otomatik belirlenilebilir.

Bu temeller üzerine kurulu sistemde, drone stabil bir şekilde uçuşunu sağlayabilmektedir. Ağ üzerinden kontrol komutlarını alabilmekte; sistemin mobil uygulama üzerinden kontrolü sağlanabilmesi için önünde artık bir engel bulunmamaktadır. ESP8266 ve Kotlin ile yazılmış mobil uygulama tarafı ve Python API dair detaylara sonraki bölümlerde yer verilecektir.

3.2.2. Mobil Uygulama Tasarım ve Yazılımı

Genel Tasarım: Projenin temel amacı doğrultusunda, insansız hava aracının (drone) mobil bir uygulama aracılığıyla uçurulması ve komut verilmesi sağlanmıştır. Bu amaçla, Kotlin programlama dili kullanılarak geliştirilen ve Şekil 4'te gösterilen mobil uygulama arayüzü tercih edilmiştir.



Şekil 4.

Uygulama arayüzünde iki adet **slider** (kaydırıcı) bulunmaktadır. Bu slider'lar, kullanıcının hareket miktarına bağlı olarak drone'a throttle (gaz) ve pitch (öne/arkaya eğim) komutlarının iletilmesini sağlar (Şekil 4 ve 5). Slider'ların pozisyonuna göre oluşturulan komutlar, uygulama tarafından işlenerek drone'un hareket parametrelerinin hassas bir şekilde ayarlanmasına olanak tanır.

```
@Composable
fun HorizontalSliderControl(label: String, value: Float, onValueChange: (Float) -> Unit) {
    Column(horizontalAlignment = Alignment.CenterHorizontally) {
        Text(
            text = "$label: ${(value * 100).roundToInt()}%",
            color = Color.White,
            fontSize = 18.sp
        )
        Slider(
            value = value,
            onValueChange = { onValueChange(it) },
            valueRange = -1f ≤ .. ≤ 1f,
            modifier = Modifier
                .width(250.dp)
                .padding(8.dp)
        )
    }
}
```

Şekil 5.

```
Slider(
    value = value,
    onValueChange = { onValueChange(it) },
    valueRange = 0f ≤ .. ≤ 1f,
    modifier = Modifier.fillMaxWidth()
)
```

Şekil 6.

Bunun yanı sıra, uygulamada üç adet buton yer almaktadır. Bu butonlar, drone'a arm (motorları aktifleştirme), disarm (motorları devre dışı bırakma) ve mod değişikliği komutlarını string veri tipinde kaydeder (Şekil 6). Son olarak, mobil uygulamada oluşturulan tüm komutlar, sendCommandToPython fonksiyonu kullanılarak Python tabanlı API sunucusuna iletilmektedir. Bu sayede komutların hızlı ve güvenilir şekilde drone'a aktarılması sağlanmaktadır (Şekil 7).

```
) {
    Button(onClick = { sendCommandToPython( command: "arm") }) {
        Text( text: "Arm")
    }
    Spacer(modifier = Modifier.height(16.dp))
    Button(onClick = { sendCommandToPython( command: "disarm") }) {
        Text( text: "Disarm")
    }
    Spacer(modifier = Modifier.height(16.dp))
    Button(onClick = { sendCommandToPython( command: "cmode") }) {
        Text( text: "Change Mode")
    }
}
```

Şekil 7.

3.2.3. Mobil Uygulama ile PythonAPI Veri Aktarımı

Mobil uygulamadan Python tarafına veri aktarımı, projede oluşturulan yerel (lokal) bir web sunucusu aracılığıyla gerçekleştirilmektedir. Bu amaçla iki farklı sendToPython fonksiyonu kullanılmıştır. Fonksiyonlardan ilki, slider bileşenleriyle elde edilen ve float veri tipinde olan x ve y parametrelerini iletmekte; diğeri ise sadece string tipinde taşınan arm, disarm ve mod değiştirme (change mod) komutlarının aktarımında kullanılmaktadır.

Uygulamada, verilerin doğru adrese yönlendirilmesi için API_IP değişkeni tanımlanmış ve bu değişken aracılığıyla Python tarafında oluşturulan API sunucusunun adresi belirlenmiştir (Şekil 8). Komutlar, mobil uygulamadan alınarak ilgili API adresine gönderilmekte; sunucudan yanıt alınıp alınmamasına bağlı olarak uygun callback fonksiyonları tetiklenmektedir. Böylece, kullanıcının gerçekleştirdiği işlemlerin başarılı veya başarısız olduğu duruma göre sistemde log kaydı tutulmaktadır.

3.2.4. Python API ile Veri Alma ve İşleme

```
fun sendXYtoPython(x: Float, y: Float) {
    val client = OkHttpClient()
    val json = """{"x":$x, "y":$y}"""
    val body = json.toRequestBody("application/json; charset=utf-8".toMediaTypeOrNull())

    val request = Request.Builder()
        .url("http://$API_IP:5000/update")
        .post(body)
        .build()

    client.newCall(request).enqueue(object : Callback {
        override fun onFailure(call: Call, e: java.io.IOException) {
            Log.e( tag: "HTTP",  msg: "Hata: ${e.message}")
        }

        override fun onResponse(call: Call, response: Response) {
            val bodyStr = response.body?.string()
            Log.d( tag: "HTTP",  msg: "Başarılı cevap: $bodyStr")
        }
    })
}
```

Şekil 8.

```
app = Flask(__name__)

x = 0.0  # Roll
y = 0.0  # Throttle
command = None
master = None
```

Şekil 9.

Python tarafında, Flask çatısı kullanılarak yerel bir API sunucusu yapılandırılmıştır (Şekil 8). Mobil uygulamadan alınacak x, y ve command (Şekil 9). değişkenlerini geçici olarak saklamak üzere global değişkenler oluşturulmuştur.

```
@app.route('/data', methods=['POST'])
@app.route('/update', methods=['POST'])
@app.route('/command', methods=['POST'])
def handle_all():
    global x, y, command

    data = request.get_json()
    if data is None:
        print("[ERROR] JSON yok!")
        return {"status": "error", "reason": "No JSON"}, 400
```

Şekil 10.

Sunucu üzerinde tanımlı olan handle_all fonksiyonu (Şekil 10), data, update ve command endpoint'lerine gelen tüm verileri dinlemektedir. Tüm komutların bu adreslerden alınması hedeflendiğinden, yalnızca POST yöntemi desteklenmektedir. Projede, Python tarafından veri çekilmesine ihtiyaç duyulmadığı için GET yöntemi aktif edilmemiştir.

```
data = request.get_json()
if data is None:
    print("[ERROR] JSON yok!")
    return {"status": "error", "reason": "No JSON"}, 400

if 'x' in data:
    x = data['x']
    print(f"[LOG] Gelen X (Roll): {x} -> PWM: {pwm_from_slider_roll(x)}")

if 'y' in data:
    y = data['y']
    print(f"[LOG] Gelen Y (Throttle): {y} -> PWM: {pwm_from_slider_throttle(y)}")

if 'command' in data:
    command = data['command']
    print(f"[LOG] Gelen komut: {command}")

return {"status": "ok"}
```

Şekil 11.

API sunucusuna ulaşan veriler JSON formatında işlenmekte ve data değişkenine aktarılmaktadır (Şekil 11). Eğer veri gelmezse, ilgili koşul blokları ile hata mesajı terminale yazdırılmaktadır. Veri başarılı bir şekilde ulaştığında ise, ilgili PWM değeri hem terminalde görüntülenmekte hem de sırasıyla pwm_from_slider_roll veya pwm_from_slider_throttle fonksiyonlarına iletilmektedir.

```
def pwm_from_slider_roll(x):
    return int(1500 + (x * 500))

def pwm_from_slider_throttle(y):
    return int(1050 + (y * 1000))
```

Şekil 12.

PWM değerleri, slider bileşenlerinden alınan değerlere göre dinamik olarak hesaplanmakta ve minimum değer sınırları 1050 ile 1500 arasında tutulmaktadır (Şekil 12). Böylece, drone'a gönderilecek motor sinyalleri hem güvenli hem de kontrollü bir şekilde belirlenmektedir.

```python
def pixhawk_main():
    global master, command, x, y

    print("[MAIN] Pixhawk'a UDP ile bağlan□l□yor...")
    master = mavutil.mavlink_connection('udp:192.168.4.2:14550')
    master.wait_heartbeat()
    print("[MAIN] Bağlant□ kuruldu (heartbeat al□nd□).")

    while True:
        if command == "arm":
            print("[KOMUT] ARM komutu al□nd□.")
            master.arducopter_arm()
            master.motors_armed_wait()
            print("[KOMUT] ARM tamamland□.")
            command = None

        elif command == "disarm":
            print("[KOMUT] DISARM komutu al□nd□.")

            master.mav.rc_channels_override_send(
            master.target_system,
            master.target_component,
            roll_pwm,        # CH1: Roll
            65535,           # CH2: Pitch
            1000,    # CH3: Throttle
            65535, 65535, 65535, 65535, 65535, 65535
        )
            master.arducopter_disarm()
```

Şekil 13.

Şekil 13'te gösterildiği üzere, ESP8266 modülü üzerinden yerel hosta UDP protokolüyle bağlantı kurulmaktadır. Bu bağlantı üzerinden, Pixhawk uçuş kontrol kartı ile iletişimin sağlanabilmesi için MAVLink protokolü kullanılmaktadır. MAVLink protokolü, Python tarafından oluşturulan komutların Pixhawk'ın anlayabileceği formata dönüştürülmesine olanak tanır. Sistem başlatıldığında, master.wait_heartbeat() fonksiyonu ile Pixhawk ile bağlantı sağlanmakta ve bağlantının başarılı olması durumunda kullanıcı terminal aracılığıyla bilgilendirilmektedir.

Mobil uygulamadan gelen string tabanlı veriler, global bir değişken olan command üzerinde saklanmaktadır. Bu komutlar, sistemde ilgili if-elif blokları ile kontrol edilmekte ve hangi işlemin yapılacağı belirlenmektedir.

Örneğin, kullanıcı tarafından arm komutu verildiğinde, master.arducopter_arm() fonksiyonu çağrılarak drone'un motorları aktif hale getirilmektedir. Disarm komutu ile, öncelikle motorlara gönderilen PWM oranı 1000 olarak ayarlanmakta (bu değer motorun durma değerine eşittir), ardından master.arducopter_disarm() fonksiyonu çağrılarak motorlar güvenli şekilde devre dışı bırakılmaktadır. Bu yöntem, drone'un mümkün olan en hızlı ve emniyetli biçimde disarm edilmesini sağlamak amacıyla uygulanmıştır.

```
roll_pwm = pwm_from_slider_roll(x)      # CH1: Roll
throttle_pwm = pwm_from_slider_throttle(y)  # CH3: Throttle

print(f"[RC OVERRIDE] X (Roll): {x:.2f} -> PWM: {roll_pwm}, Y (Throttle): {y:.2f} -> PWM: {throttle_pwm}")

master.mav.rc_channels_override_send(
    master.target_system,
    master.target_component,
    roll_pwm,        # CH1: Roll
    65535,           # CH2: Pitch
    throttle_pwm,    # CH3: Throttle
    65535, 65535, 65535, 65535, 65535, 65535
)

time.sleep(0.1)
```

Şekil 14.

Şekil 14'te görüldüğü gibi, roll ve throttle değerleri uygulama üzerinden anlık olarak algılanmakta ve bu değerler

```
master.mav.rc_channels_override_send(
```

fonksiyonuna parametre olarak iletilmektedir. Bu fonksiyon aracılığıyla roll komutu Pixhawk'ın 1. kanalına, throttle komutu ise 3. kanalına aktarılmaktadır. Böylece, kullanıcının mobil uygulama arayüzünden yaptığı hareketler gerçek zamanlı olarak drone'un uçuş dinamiklerine yansıtılmakta ve hassas bir kontrol sağlanmaktadır..

4. Bulgular ve Tartışma

Projenin tamamlanmasıyla gerçekleştirilen testler ve senaryolar sonucunda, Wi-Fi üzerinden mobil cihazlarla drone kontrolünün başarılı bir şekilde sağlanabildiği görülmüştür. Mobil uygulama aracılığıyla gönderilen throttle, pitch ve temel kontrol komutları, Python tabanlı bir API ile ESP8266 modülü üzerinden Pixhawk uçuş kontrol kartına güvenilir şekilde aktarılmıştır. Genel olarak, sistem düşük gecikme süresiyle kararlı bir iletişim sağlamıştır.

Uygulamanın arayüzü üzerinden yapılan komut gönderimlerinde herhangi bir veri kaybı veya gecikme sorunu yaşanmamış, özellikle throttle ve pitch gibi değerlerin anlık olarak güncellenebilmesi sayesinde kullanıcıya hassas bir kontrol deneyimi sunulmuştur. ESP8266'nın UDP tabanlı bağlantı mekanizması ve MAVLink protokolü, komutların istikrarlı bir şekilde işlenmesini mümkün kılmıştır.

Deneysel çalışmalarda, drone'un arm/disarm işlemleri ve uçuş modları hızlı ve güvenli bir şekilde değiştirilmiş, verilen komutlara yanıt süresinin bir saniyenin altında olduğu tespit edilmiştir. Python tabanlı sunucu sistemi, komutları filtreleyerek doğrulama işlemini gerçekleştirmiş ve hatalı veri girişlerinin önüne geçerek güvenliği ön planda tutmuştur.

Ancak sistemin performansı, testlerin yapıldığı ağ ortamına ve donanım uyumluluğuna bağlı olarak değişkenlik göstermiştir. Özellikle Wi-Fi bağlantısının zayıf olduğu durumlarda kısa süreli gecikmeler ya da kopmalar gözlemlenmiştir. Bunun yanında, ESP8266 ile Pixhawk arasındaki veri iletiminin doğruluğu, kullanılan protokollerin stabilitesi ve donanım bileşenlerinin uyumu, sistemin genel başarısını doğrudan etkilemiştir.

Gelecekte, projenin kamera tabanlı görüntü işleme desteğiyle genişletilmesi, birden fazla drone ile senkronize çalışma senaryolarının oluşturulması ve GPS bazlı otomasyon sistemlerinin entegrasyonu gibi yenilikler, sistemin yeteneklerini daha ileri taşıyabilir. Mobil uygulamanın daha fazla kontrol seçeneği sunacak şekilde geliştirilmesi ve farklı ağ koşulları altında test edilmesi, sistemin genellenebilirliğini ve esnekliğini artıracaktır.

Sonuç olarak, geliştirilen Wi-Fi tabanlı drone kontrol sistemi, hedeflenen uçuş komutlarını hızlı ve güvenli bir şekilde yerine getirirken, sezgisel ve kullanıcı dostu bir arayüz sunmaktadır. Kablosuz haberleşme, Python API ve MAVLink entegrasyonunun başarılı bir şekilde kullanılması, gelecekte ek modüller ve yeni işlevlerle daha da geliştirilmeye açık bir yapı oluşturmaktadır.

5. Sonuç

Bu proje, Wi-Fi tabanlı haberleşme altyapısı kullanılarak geliştirilen mobil uygulama destekli bir drone kontrol sistemiyle, insansız hava aracı yönetiminde yeni nesil bir yaklaşım sunmayı hedeflemiştir. Geleneksel kumanda ve fiziksel kontrol araçlarına bağımlılığı ortadan kaldıran bu sistem, özellikle erişilebilirlik, pratiklik ve uzaktan gerçek zamanlı müdahale gibi alanlarda önemli avantajlar sağlamaktadır. Geliştirilen çözümde, Kotlin diliyle tasarlanmış mobil uygulama üzerinden verilen throttle, pitch ve temel uçuş komutları, Python tabanlı API ile ESP8266 Wi-Fi modülü aracılığıyla Pixhawk uçuş kontrol kartına başarıyla iletilmiştir.

Sistemin bütünleşik yapısı sayesinde, kullanıcıdan alınan komutlar düşük gecikmeyle, güvenli ve kararlı bir şekilde drone'a aktarılabilmiştir. Özellikle throttle ve pitch gibi hassas değerlerin slider üzerinden gerçek zamanlı güncellenmesi, uçuş kontrolünde yüksek doğruluk ve kullanıcı deneyimi sunmuştur. Ayrıca arm, disarm ve mod değişikliği gibi kritik fonksiyonların mobil uygulama üzerinden güvenle yönetilebildiği gözlemlenmiştir.

Testler sırasında, sistemin hem yazılım hem de donanım katmanında yüksek uyumluluk ve kararlılık sergilediği belirlenmiştir. Komutların iletiminde kayda değer bir gecikme veya veri kaybı yaşanmamış; tüm işlemler hem sunucu tarafında hem de uçuş kontrol kartında başarıyla sonuçlandırılmıştır. Ayrıca, Python API'nin güvenlik ve hata filtreleme mekanizmaları sayesinde istenmeyen ya da hatalı komutların drone'a ulaşması büyük ölçüde engellenmiştir.

Bu kablosuz kontrol altyapısı, yalnızca teorik açıdan değil, pratik uygulama performansı ve donanım esnekliği açısından da başarılı sonuçlar vermiştir. Özellikle drone teknolojisinin yaygınlaştığı günümüzde, mobil uygulama tabanlı ve kolay erişilebilir kontrol sistemlerine olan ihtiyaç giderek artmaktadır. Bu proje, sunduğu altyapı ve mimariyle; eğitim, araştırma, robotik, insansız sistemler ve erişilebilirlik teknolojileri başta olmak üzere geniş bir uygulama alanına hitap etmektedir.

Sonuç olarak, geliştirilen sistem; hızlı kurulum, düşük maliyet, kolay entegrasyon ve kullanıcı dostu arayüzü ile klasik kontrol yöntemlerine alternatif, yenilikçi ve uygulanabilir bir çözüm sunmaktadır. İlerleyen çalışmalarda kamera entegrasyonu, otonom görevler veya çoklu drone yönetimi gibi gelişmiş modüllerin eklenmesiyle sistemin kapsamı ve işlevselliği daha da artırılabilir.

Gelecekte Yapılabilecek Geliştirmeler

Projenin mevcut halinde, temel uçuş komutlarının mobil uygulama üzerinden başarıyla drone'a iletildiği ve güvenli bir şekilde uygulandığı bir kontrol altyapısı sunulmaktadır. Ancak sistemin daha işlevsel, yaygın ve endüstriyel ölçekte kullanılabilir bir ürüne dönüşmesi için aşağıdaki geliştirme alanları değerlendirilmelidir:

- **Otonom Görev ve Kusur Tespiti:** Sistemin bir sonraki aşamasında, drone'un güneş paneli yüzeylerinde kusur tespiti ve raporlaması amacıyla otomatik görevler icra etmesi hedeflenmektedir. Bu amaçla, drone üzerinde yüksek çözünürlüklü kamera ve görüntü işleme algoritmalarının entegrasyonu ile, panel yüzeyindeki çatlak, kirlilik veya arıza gibi kusurların tespit edilip otomatik olarak raporlanması sağlanabilir.
- **Komut ve Fonksiyon Çeşitliliğinin Artırılması:** Mobil uygulama üzerinden sadece temel uçuş komutlarının değil, aynı zamanda belirli otonom tarama modlarının, panel bölgesi seçimlerinin ve fotoğraf/video çekimi gibi ek görevlerin de kontrol edilmesi için uygulamaya yeni fonksiyonlar eklenebilir.
- **Veri Analizi ve Bulut Tabanlı Raporlama:** Drone ile toplanan kusur verilerinin gerçek zamanlı olarak bulut tabanlı bir platforma aktarılması, kullanıcıların her yerden erişebileceği otomatik raporlar ve bakım önerileri üretmesine olanak tanıyabilir.
- **Uzun Menzilli ve Kararlı Haberleşme:** Mevcut Wi-Fi altyapısına ek olarak, açık alanlarda daha kararlı ve uzun menzilli iletişim için LTE/5G gibi alternatif haberleşme protokolleri değerlendirilebilir. Böylece geniş güneş enerjisi tarlalarında kesintisiz görev yürütmek mümkün olacaktır.
- **Gelişmiş Sensör ve Kamera Sistemleri:** Görüntü işleme başarısını artırmak amacıyla kızılötesi kamera, termal kamera veya LIDAR gibi ek sensörler entegre edilebilir. Bu sensörler ile panel verimliliği, sıcaklık analizi veya detaylı yüzey taraması yapılabilir.
- **Çoklu Drone Yönetimi:** Geniş alanlarda tarama verimliliğini artırmak için birden fazla drone'un koordineli olarak çalışabileceği, merkezi bir kontrol ve görev paylaşım sistemi geliştirilebilir.
- **Endüstriyel ve Pratik Kullanım Senaryoları:** Sistem, güneş paneli taramasının ötesinde; elektrik hatları, tarım arazileri veya diğer büyük ölçekli altyapı kontrolleri gibi farklı endüstriyel uygulama alanlarına uyarlanabilir.

Sonuç olarak, mevcut sistem güçlü bir temel sunmakla birlikte; donanım, yazılım ve otomasyon alanlarında yapılacak bu iyileştirme ve eklemeler sayesinde, güneş panellerinin bakım ve izlenmesinde akıllı ve etkili bir insansız hava aracı çözümüne dönüştürülebilir.

6. Kaynakça

- https://ardupilot.org/plane/docs/roll-pitch-controller-tuning.html
- https://ardupilot.org/copter/docs/tuning-process-instructions.html#tuning-process-instructions
- https://docs.px4.io/main/en/config_mc/pid_tuning_guide_multicopter.html
- https://ardupilot.org/copter/docs/autotune.html
- https://ardupilot.org/copter/docs/common-esp8266-telemetry.html
- https://ardupilot.org/dev/docs/esp32-autopilot.html
- https://github.com/ArduPilot/mavesp8266/tree/master
- https://youtu.be/Jkh-J3SWR1I?si=dYrZMpEXASDyw3HW
- https://youtu.be/V_mZsiZcy7s?si=9m5P62LgfxLFAccT
- https://youtu.be/dMRDzicSvXk?si=XtL3FYPLZLNCWZbC