KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI ANABİLİM DALI

PID KONTROLLÜ OTONOM QUADCOPTER TASARIMI

YÜKSEK LİSANS TEZİ

Zekeriya HACIMUHAMMED

AĞUSTOS 2019
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

PID KONTROLLÜ OTONOM QUADCOPTER TASARIMI

Zekeriya HACIMUHAMMED

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

Tezin Enstitüye Verildiği Tarih : 22 / 07 /2019
Tezin Savunma Tarihi : 08 / 08 /2019

Tez Danışmanı : Dr.Öğr.Üyesi İbrahim SAVRAN

Trabzon 2019

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı
Zekeriya HACIMUHAMMED

PID KONTOLLU OTONOM QUADCOPTER TASARIMI

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 25 / 06 / 2019 gün ve 1808 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Abdulsamet HAŞILOĞLU

Üye : Dr. Ogr. Uye. Ibrahim SAVRAN

Üye : Dr. Ogr. Uye. Hüseyin PEHLİVAN

Prof. Dr. Asim KADIOĞLU
Enstitü Müdürü

ÖN SÖZ

ALLAH’a teşekkür ettikten sonra, proje kapsamında hazırlamış olduğum yüksek lisans tezinde tüm iyi niyetiyle yardımlarını esirgemeyen, bana kendisiyle çalışma fırsatı sunan Sayın hocam Prof. Dr. İbrahim SAVRAN'a teşekkür eder, saygılarımı sunarım.

Ayrıca tez çalışmam süresince desteklerinden dolayı Muslem DAMKHİ ve Cem BAYAR teşekkürlerimi sunarım. Sosyal ve arkadaşlık ilişkileri bağlamında sürekli yanımda olan beni motive eden, her fırsatta farklı bakış açılarıyla değişik fikirleri düşünmemi sağlayan arkadaşlarım Raşit MUSTAFA ve Muhammed ALKASIM’a, fırsatları değerlendirme ve insanını başına gelebilecek olası olumsuz olayları bizzat yaşayıp tecrübelerini benimle paylaşan ağabeyim Bekir HACIMUHAMMED’a ayrıca teşekkürlerimi sunarım.

Eğitimim süresince ufkumu genişletmiş olan tüm öğretmenlerime, hocalarıma ve son olarakta bugünlere gelmemde en büyük emeğe sahip olan maddi ve manevi desteklerini esirgemeyenailemesonsuz teşekkür etmeyi borç bilirim.

Zekeriya HACIMUHAMMED

Trabzon 2019

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Özel Quadcopter” başlıklı bu çalışmayı baştan sona kadar danışmanım Bilg.Müh ZEKERİYA HACIMUHAMMED‘ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 10/09/2019

Zekeriya HACIMUHAMMED

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

ÖZEL QUADCOPTER

Zekeriya HACIMUHAMMED

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üye. İbrahim SAVRAN
2019, 49 Sayfa

Quadcopter bu günlerde çok yayındı ayrıca şeçit sektorlarda birçok amaç için kullanmaya başlanmıştır, özellikle quadcopterler arama ve ilaç teslim hizmeti hatta askeri amaçlarda, onun için kendi otopilot kartımız tasarlardığımızda yapabileceklerimiz ufuğu hazır otopilot kullanmaktan daha çok genişleşir.

Quadcopterin en önemli kısmı uçuş kontrollörü. Arduino, ATMEGA işlemçi sahip olan mikrokontrolör benim kullanımı ile uçuş kontrollörün bütün işlevleri yapılabilir. Arduino ile manula uçuşte, hem RC vericinin sinyaları hem de sensörlardan dengeleme değerleri alınır ve PID yöntemi kullanarak rotorlar olması dönme hızı ve şimdeki quadcopterin durumu arasında hata işleyip her rotorun yeni dönme hızı bulur. rota yapabilmesi için GPS ve Barometre olması gerekiyor ve bu halde bir daha PID kullanılacak, bu PID rotanın noktaların arasındaki mesafe, irtifa ve yön hatası işlerek quadcopteri yönlandırır.

Kullandığı dil c++ arduino uzerinde ve simülasyonu için bir matlab quadcopterin simülatörüde yapıldı sistem ugulayıp test edildi.

Anahtar Kelimeler: Quadcopter, Uşuç sistemi, Görev, Drone, arduino

Master **Thesis**

SUMMARY

ÖZEL QUADCOPTER

Zekeriya HACIMUHAMMED

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Asst. Prof. Dr. Ibrahim SAVRAN
2019, 49 Pages

The Quadcopter has been widely circulated these days and has begun to be used in many industries for many purposes, especially autonomous quadcopter search and drug delivery service, even for military purposes.when we design our own autopilot card we can do the more and more thinks wider the use of the ready-made autopilot.

The most important part of the autonomous quadcopter is the flight controller. In this study by using Arduino that has a ATMEGA micro-controller as processor, all functions of flight controller rightly can be done. The Arduino at the manuel flight take both the balance values from the sensors and the RC transmitters signals and use the PID method to calculate the error between the current state of the quadcopter and the needed state then finds a new rotation speed for each rotor. In order to be able to do autonomously route but needs to GPS and Barometer, and in this case one more PID will be used, this PID will direct the quadcopter to the distance between the points of the route by solving the altitude and direction errors. The language used is c++ on arduino and a quadcopter simulator for simulation was run on the matlab.

Key Words: Quadcopter, Ucar system, GPS, Task, Drone, autonomous, arduino.

ŞEKİLLER DİZİNİ

Sayfa No

KISALTMALAR LİSTESİ

Quadcopter	:Sabit kanatlı uçak dört prevaneden oluşuyor..
Drone	:İnsansız hava aracı.
MPU	:(Motion Proccessing Unit) hareket işleme ünitesi.
PID	:(Proportional-İntegra-Derivative) Oransal-İntegral-Türevsel denetleyici

1. GENEL BİLGİLER

1.1. Giriş

Otonom otopilot arştırımda hem manula hem otonom arduino ile bir quadcopter yapmaya hedefledim ve quadcopter'in temellerini tarif edip ve nasıl inşa edeceğimiz konusunda rehberlik edeceğim.

Bu günlerde otonom quadcopter çok önemli bir araç ortaya çıkmıştır, onudan profesyonel bir çekilde faydalanbilmek için kendi quadcopterimizin otopilot ana kartı sıfırdan programlayarak bütün görevleri kolay olacaktır, hatta nesnelerin takipi için hemen uçuş emirleri control edebiliriz.

Bu arştırmada bir kaç sıkıntı olabilir onlardan gps hatası 2 ile 3 metre arası oldu için bir hassasiyet sıkıntısı yaratır hem de ruzgar, quadcopter hafif olduğu için etkiler.

Hazır otopilot kartleri bir kaç tane büyük şerketler tarafından tsarlandı ve onlar bir kaç tür uçaklara kullanılabilir, Bazları arduino kullandı ve bazları başka bir mikroişlemci kullanmış. Bu kartların bazları açık kaynak ve ona rağmen otonom aldıgı zaman harıcı bir yerden uçuşu kontrol edilmiyor. Turkiye'de otonom otopilot kapsayan bir araştırma bulamadım, ancak dışında da oranı azdır.

1.1.1. Araştırmanın Amacı

Aslında bu araştırmanın ana amacı quadcopter'in otonom otopilot anakartı yapmaktır. Bir otobilot kartı sahip olup ve istedimiz şekilde kullanabileceğiz, özellikle onun kodları biz yaptık ve her sensör nasıl çalıştığı ve ne faydası bilmekteyiz. Diğer amacı quadcopter'in tsarımı ve programlaması yapıp kolaylaştırp ele taşımak.

1.1.2. Otonom Quadcopter Önemi

Quadcopter ağırlığı hafif, hızlı, düşük maliyetli, havada kalabilir, trafikte takılmaz ve kısa yoldan uşabilir gibi özellikler sahip olduğu için çok önemli bir aractır, özellikle otonom olduğu zaman:

1.1.2.1. Arama

Her hangi bir şey bulmak için quadcopter ile üsten baktığı için daha kolaydır, özellikle bir arama programı kurursak şu zamanda aradığımız nesne komandasız quadcopteri onu buluyor.

1.1.2.2. Küçük Ürünler Teslimati

Quadcopter, birçok büyük şirket tarafından müşterilere ürün teslimatı için kullanılmaktadır.

1.1.2.3. Askeri Alanında

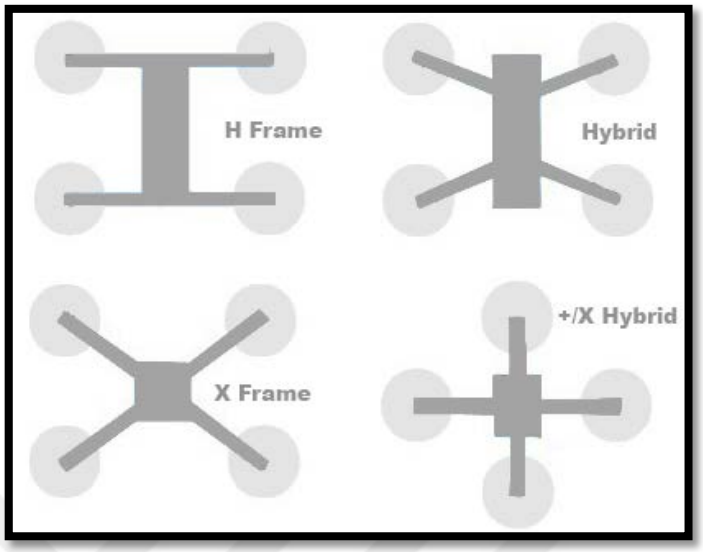
Quadcopter bu alanda çok önemli ve hassastır. Quadcopter çok görevler yapabilir onlardan duşmen bölgeleri görüntü almak, duşman asker bulup onlara vurmak yada küçük bomba atmak.

1.2. Temel Quadcopter'in Parçalari

Otonom otopilot quadcopter için yapılacak ve quadcopter genellikle bir kaç temel parçadan oluşur . Otopilot bu temel parçaların standartları değişirse bile uyumlu olacak çünkü hepsini göz önünde alındı. Bu parçaları:

1.2.1. Uçak Gövdesi

Her şeyi bir arada tutar. Genellikle güçlü ve hafif şekilde tasarlanır. Ana uçuş kontrol çipi ve sensörlerinin monte edildiği bir merkez plakasından ve motorların monte edildiği kollardan oluşur. Gövdenin şekli (+) ,(X) yada (H) olabilir şekil 1 onları gösterir. En çok karbon fiber, fiberglas, alüminyum veya çelikten yapılır. Gövdenin standartları ağırlık ve kolların uzunluğu.



Şekil 1. Quadcopter gövdesinin şeklileri

1.2.2. Motor

Motorları kullanmanın ana amacı pervaneleri döndürmektir. Bu halde pervaneleri, yerçekimi ve sürükleme karşı itme sağlar. quadcopter'in dört motoru var ve her motor bir hız kontrol cihazı tarafından ayrı ayrı kontrol edilmelidir.

Quadcopter kullandığı motorları fırçasız motorlar şekil 2. Fırçasız motorlar çok daha hızlı döner ve DC motorlara göre aynı hızda daha az güç tüketir.

Motorun genelliklerrr iki standartları var

- Boyut: yükseldikçe daha ağırlık kaldırabilir yanısıra daha göç tüketir.
- KV (kilovolt): motorun volt başına ne kadar çeşitli RPM'lerin (her an devir) nasıl olacağını gösterir, KV derecesi ne kadar yüksek olursa, motor daha hızlı bir şekilde sabit bir voltajda döner.

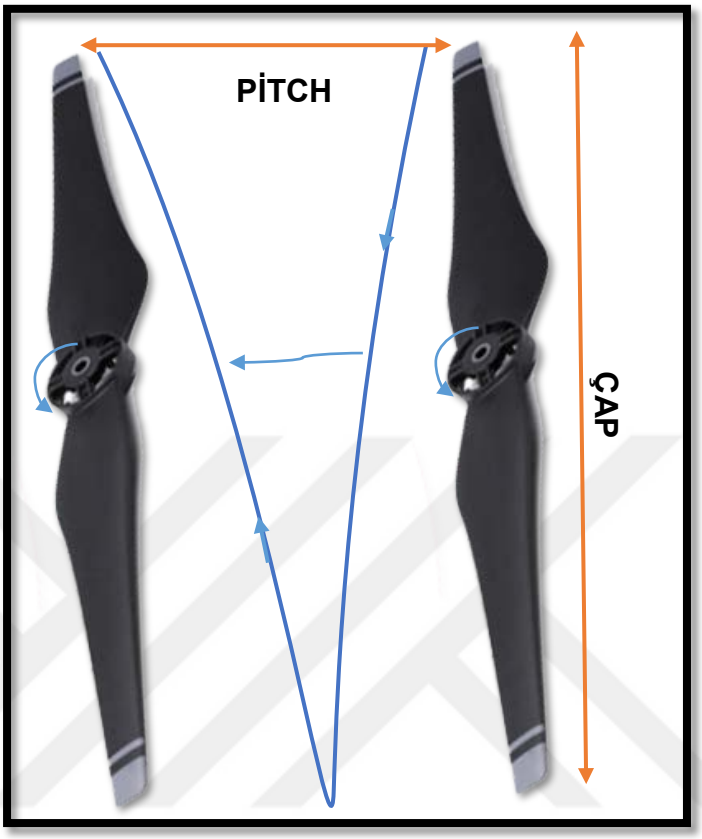
Şekil 2. Fırçasız motorlar

1.2.3. Pervaneler

Bir quadcopter'in dört pervanesi vardır, iki pervane saat yönünde dönüyor ve diğeri saat yönünün tersine döner. Pervaneler bir motor ile döndüğü zaman, rüzgarı aşağı doğru üfler. Yeterince hızlı döndüğünde uçağı yükseler ve yavaşladığında inder.

Pervanele iki standartı var

- Pitch: bir pervanenin bir devrede hareket edeceği mesafe. *Pitch* uzun olduğunda daha uzak bir mesafe keser ama bu halde titreşim oranı yükselir. Görüntü çekim yapan quadcopterler az *pitch* sahip olan pervaneleri kullanır ama yarışma amaca tsarlanmış olanlar büyük *pitch* sahip olan pervaneleri tercih eder şekil 3.
- Çap: büyük motorlar büyük çaplı bir pervaneler kullanır ki daha yük taşıyabilmek. Pervane çap quadcopter'in gövdesini boyutuna karşılık gelir şekil 3.

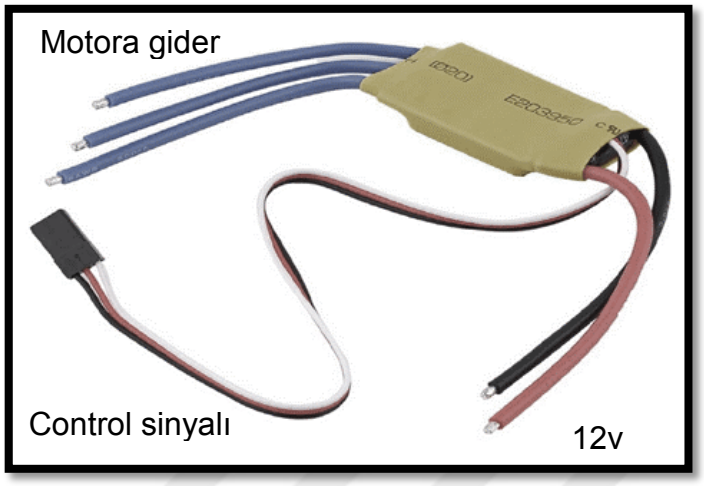


Şekil 3. Pervanlerin standartleri

1.2.4. ESCs (Electronic Speed Controller)

Elektronik hız kontrol cihazı veya ESC, uçuş kontrolörüdan gelen bir PWM sinyali kullanarak ortalama bir voltaj / akım yaratır ve motorun hızını kontrol eder. Üç dizi tel vardır; Bir set iki kablu 12 v almak için bile gider, bir set üç lablu PWM sinyalari taşır uçuş kontrolörüne gider ve bir set üç kablu ESC'dan cıkıp motorlara gider şekil 4.

Her motora ayrı ayrı ESC’si var onunla kontrol edelir. ESC’lar programlanabilir ve kullanmadan önce kalibre edilmelidir.



Şekil 4. ESC (Elektronik hız kontrol cihazı)

1.2.5. Alıcı ve Verici Cihazları

Uzaktan pilotlu quadcopters, bir çeşit verici ve alıcı kullanarak bir yer istasyonundan kontrol edilir. Bu iletişim sistemi basit, güvenilir ve kurulumu kolaydır. Yer istasyonu çok daha fazla sistemden yapılabilir, ancak basitleştirilmiş bir iletişim modeli radyo 2.4GHz gibi kullanarak tercih edilir. Vericiler genellikle şekil 5 gibi.

Şekil 5. ESC (Elektronik hız kontrol cihazı)

Vereci alıcı cihazının en önemli standartlarından iki tanedir

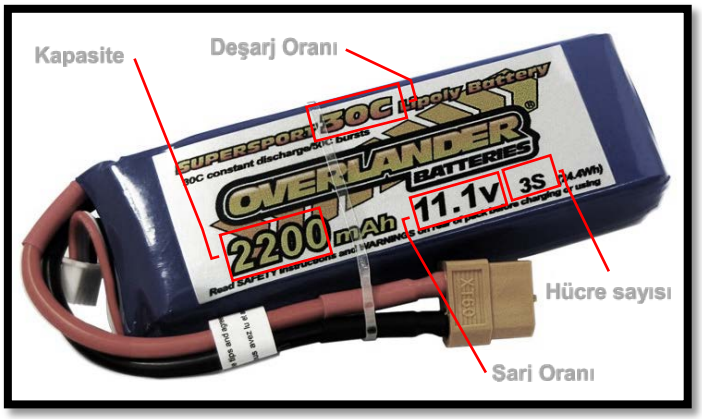
- **Kapsama:** en iyi 2.4 Ghz radyo 300m ila 1.5Km aralığında olabilir.
- **Kanal sayısı**: en basit radyo cihazı dört kanala sahiptir, Ana verici kanalları dört kanaldır, *throttle* (gaz kelebeği), *yaw* (rudder), *roll* (aileron), and *pitch* (elevator) bu kanallar aracılığıyla tamamen kontrol edilir. Ayrıca, bir verici ilave fonksiyonlar için iki veya daha fazla kanal ile donatılmıştır; Örnek, bir pilotun bazı ışıkları yakması, bir sistemi aktive etmesi veya bir hava fotoğrafı çekmesi gerekebilir. Tüm bunlar, bir pilotun herhangi bir ekstra kanal üzerinden etkinleştirebileceği ekstra fonksiyonlardır.

1.2.6. Bil

Lityum Polimer (LiPo) biller, quadcopter için en uygun seçenek haline getiren birçok özelliğe sahiptir onalardan hafif, büyük kapasitesi ve uzun zaman sürer.

En önemli standartları şekil 6 :

- **Hücre sayısı:** her hücrenin voltajı 3,7 v ,
  3 hücre x 3,7 volt (3S) = 12,6V
- **Kapasite:** Kapasite pil tutabilir ne kadar güç / enerji belirtir ve miliamper saat (mAh) belirtilir.
- **Maksimum şarj oranı:** Bataryanın güvenli bir şekilde şarj edilebileceğini bildiren en yüksek şarj akımıdır.
- **Deşarj oranı:** Bir Bilin ne kadar hızlı güvenle boşaltılabilir. 10C deşarj derecesine sahip bir bil, 10 kat daha hızlı bir şekilde boşaltabileceğiniz anlamına gelir.
  bir 15C = 15 kat daha fazla, bir 20C = 20 kat daha fazla, vb.



Şekil 6. Lityum Polimer (LiPo) biller

1.2.7. Uçuş Kontrolörü

Uçuş kontrolörü quadcopter'in beynidir ve her şeyi kontrol ediyor. Uçuş kontrolörü ana bileşimileri yeterince hızlı bir mikro kontrolcü ve onun yanında bazı sensörler. Uçuş kontrolörü amaçlara göre bileşimi değişir, manuel uçuş etmesini istediğimiz zaman bir mikro kontrolcünün ve onun yanında denge sensörü ve barometre sensörüden oluşur, otonom ise manuel içerdiği bileşimileriden oluşur yanlız GPS cihazına ıhtiyacı olacak yanı sıra mikro kontrolcü daha hızlı olması gerekiyor.

mikro kontrolcü olarak en yanyın Arduino kullanılıyor ve diğerleri çok, denge sensörü ise en çok MPU6050 kullanılıyor.

Hem manuel hem otonom quadcopterlerde mikro kontrolcü dört emir eşleyecek, onları quadcopter'in hareketi sağlıyacak:

- *Throttle*: Yukarı ve aşağı hareket için.
- *Yaw*: Dönme hareketi için.
- *Pitch*: İleri yada geriye doğru hareket için.
- *Roll*: Sağ yada sola hareket için.

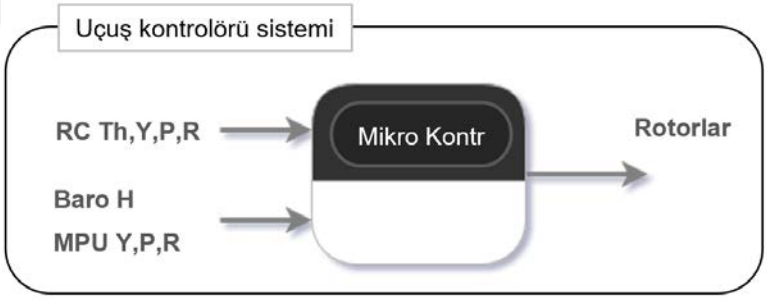
Manuel uçuşta mikro kontrolcü üçuş emirleri kumandadan alır hem de denge sensörüden (mpu) quadcopter'in üç eksanla yaptığı açıları okuyor sonrası PID kullanarak motorlara verecek hızı hesaplar.

Otonom uçuşte komandadan değil bulunduğu nokta hedef noktası ile çıkmış uzaklık ve açıdan uçuş emileri hesaplanır.

1.3. Manuel Uçuş Kontrolörü Sistemi

Manuel uçuşta mikro kontrolcü (kullanılan Arduino uno) RC vericisinden gelen kumanda komutlarını alır, RC sinyalleri PWM darbesi olarak alınıp daha sonra zamandan uygun değer aralığına dönüştürülür, bizim durumumuzda *Throttle*, *Pitch* ,*Roll* ve *Yaw* değerlerine dönüştürülür. Uçuş kontrolörün mikro kontrolcü ayne zamanda MPU'dan (Hareketli İşleme Ünitesi) alınan uçağın gerçek açıları okuyor ve Barometre'den yükseklik okuyor sonrası PID işlemi kullanarak RC vericisinden alınan arzu edilen hareket bağlı olarak her motorun hız komutunu hesaplayıp verir.

Şekil 7. Genel manuel uçuş kontrolörü sistemi, (Y,P,R) değerleri quadcopter'in *Yaw*,*Pitch*,*Roll* açılarına, H yüksekliğine ve TH Throttle değerine işaret eder.

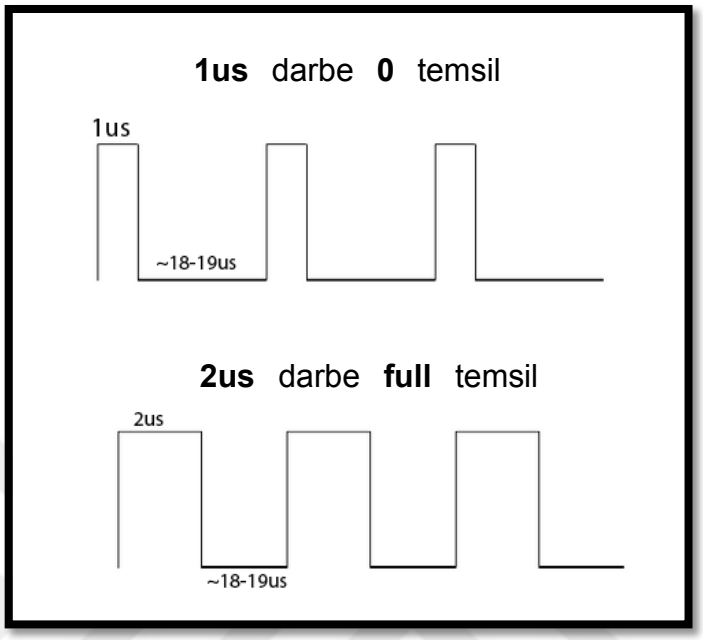


Şekil 7. Genel manuel uçuş kontrolörü sistemi

Manuel uçuş kontrolörü sistemin içininden daha deteyli bir şekilde göstermek için bu sistemin çalışma açamalarına birer uğrarak anltılacak:

1.3.1. RC Değeri Okumak:

Alıcının sinyallerinin temel prensipleri: Her bir radyo çıkışı 50Hz'de bir darbe gönderir yani 20us uzunluğu sahip bir sinyal ve her 20us tekralanacak. Tipik olarak, darbe 1us ve 2us uzunluğundadır ve bir sonraki 18us - 19us arasında duraklama vardır. Darbenin uzunluğu 1us ise yani 0 damak ve uzunluğu 2us olunca full damaktır şekil 8 onu gösterir.



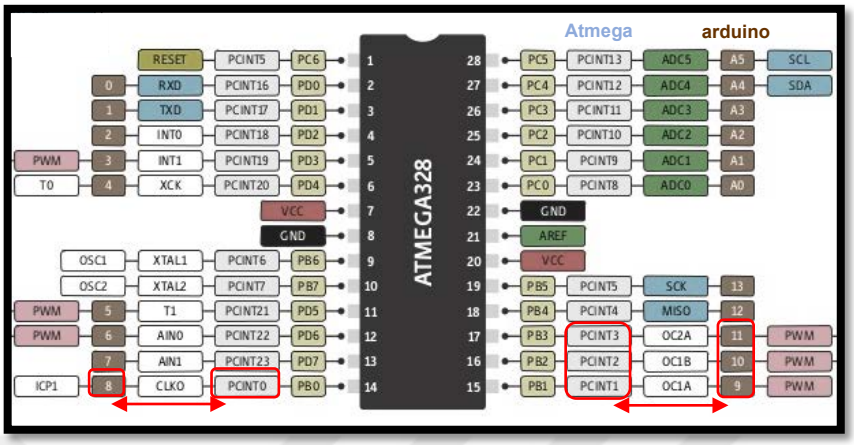
Şekil 8. Alıcı sinyalı

Arduino uno kullanacak ve onun işlemcisi ATMEGA 328P. Alıcının sinyalları Arduino trafından en iyi şekilde okumak için interrupt kullanacktır, çünkü alıcının sinyalı yenileme hızı çok düşüktür, 1 – 2 us arası bir darbe genişliği okunacak sonra 18-19 us bekleme süresi var ve arduino bu boş zamanda çok şeyler işleyebilir.

Programda Arduino komutları çok az kullanack ve en çok atmega komutları kullanacak çünkü o programın hafızası daha az yer alır.

Quadcopter'i kontrol etmek için dört ana kanal lazımdır *Throttle* ,*Yaw*, *Pitch*, *Roll*. Bu kanallar dört pine ihtiyacı olacak. Atmega 328P veri Sayfasına göre eğer interrupt kullanılacak, etkinleştirilmiş pin değişimi interrupt PCINT[7..0] pinleriden herhangi bire geçiş yapılırsa pin değişimi interruptı rutini PCINT0 etiklenir. Atmega 328P pin diyagramında bakacak olursak PCINT 0, PCINT 1, PCINT 2, PCINT 3 Arduino'da şekil 9 gösterdiği gibi şu pinlere eşlesir 8, 9, 10, 11.

Arduino onu ile alıcının pinleri bağlama diyagramı Şekil 10 gösterilecektir.



Şekil 9. Atmega 328P pino diyagramı

1.3.1.1. İnterrupt Pinleri Hazırlama

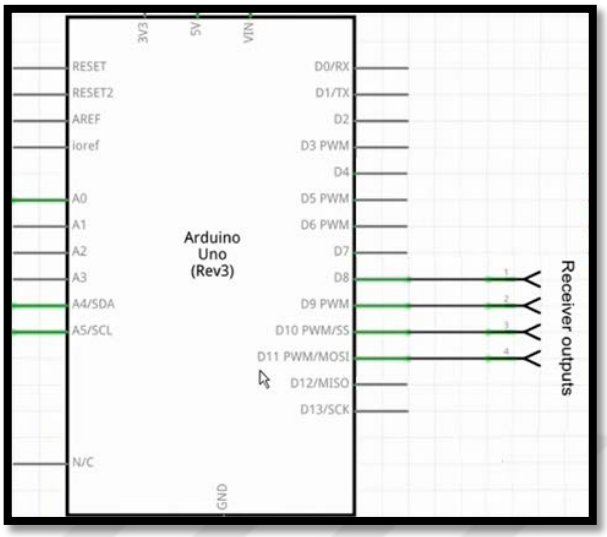
Pin değişimi interrupt PCINT çalışması için ayarlamamız gereken iki şey:

Pin değişimi interrupt etkinleştirme sıfır PCIE0 bit değeri 1 yapmak, bu bit pin değişimi interrupt control sicillinde PCICR,

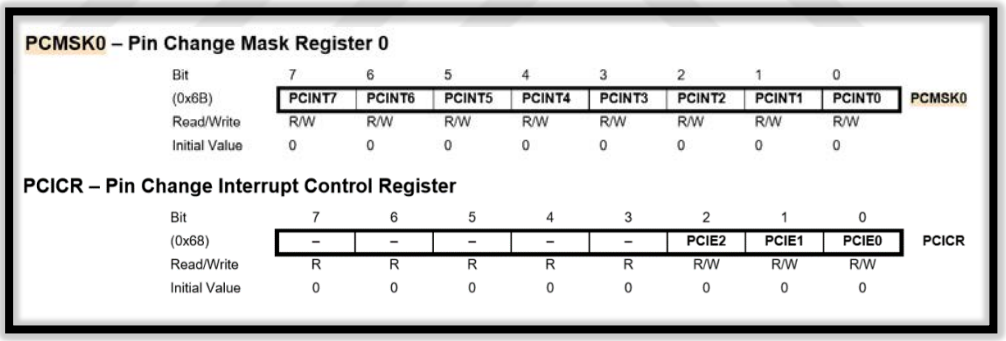
Pin değişimi maskesi sıfır PCMSK0 sicilinde PCINT [7..0] gerekli pinleri seçmemiz gerekiyor, şekil 11 PCICR, PCMSK0 sicilleri gösterir.

Bu işlemleri kodlamak için hem arduino hem de Atmega dilinde bu şekilde olacak:

```
// PCIE0 akivasiyonu
PCICR |= (1 << PCIE0);
// interrupt için PCINT 0,1,2,3 pinleri seçmek
PCIMSK0 |= (1 << PCINT0);
PCIMSK0 |= (1 << PCINT1);
PCIMSK0 |= (1 << PCINT2);
PCIMSK0 |= (1 << PCINT3);
```



Şekil 10. Arduino uno alıcı ile bağlantısı



PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Şekil 11. PCICR, PCMSK0 sicilleri

1.3.1.2. İnterrupt Rutini

Bu durumda her 8,9,10 ve 11 pinlere hangi bir değişlik interrupt rutini PCINT0 çalışacak yanı rutinin içindeki işlemler şuanda işlenecek şöyle:

```
ISR(PCINT0_vect)
{
  // işlemler
}
```

Şuanda kaldığı işlem bu interrupt rutini çağırıldığı her anda hangi pinden geldiği bilmek ve geldiği sinyalın darbesinin zamanı hesaplamaktır.

PCINT [7..0] pinleri Atmega diyagramında port B'ya bağlı ve PINB şekil 12 sicilinde port B'nin giriş pin adresleri var, PCINT0 ile PINB0 bağlı ...vp.

Her giriş için darbe geldiğinde zaman değeri alırız va darbe bitiğinde da zaman değeri alırız ve aralardaki zaman hesapladığında darbenin zamanı olacak. Her kanalın sinyaları okumak için interrupt rutininde şöyle bir kod olacak.

```
ISR(PCINT0_vect){
  current_time = micros();
  //kanal 1 -------------------
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time - timer_1;
  }
 //kanal 2 -------------------
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  .
  .
  .
```

1.3.1.3. *Throttle*, *Yaw*, *Pitch* ve *Roll* Değerlerin Temsili

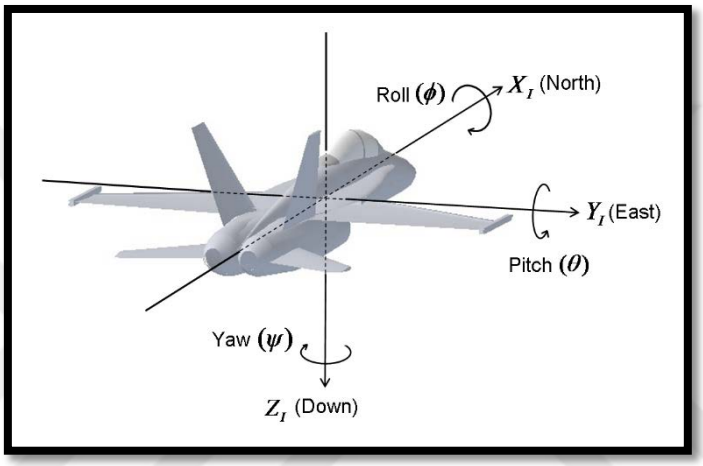
Şimdi *Throttle*, *Yaw*, *Pitch* ve *Roll* sırayla 8, 9, 10 ve 11 bağlı halinde timer_1, timer_2, timer_3 ve timer_4 sırayla da değeri olacak. Bütün değerler 1000 ile 2000 olacak. Bu değerleri standar olarak şekil 12 temsil itiği gibi olacak yanı:

Throttle değer ise 1000, 2000 arasından 0, 100 arasına dünüştürecek, *throttle* varsayılan yeri 0'da sıfır gaz damaktır 100'e kada yükselenebilir ve 100 max gaz olacak.

Yaw değeri 1000, 2000 aynı alanda kalacak ama 1500 damak ki *Yaw* eşittir 0 hiç dönme yok, 1000 sola max ve 2000 sağa max.

Pitch değeri 1000, 2000 aynı alanda kalacak ama 1500 damaki *Pitch* eşittir 0 hiç dönme yok, 1000 cuadcopter'in önü açağya max yanı ileriye hareket ve 2000 cuadcopter'in önü ukarıya max yanı geriye hareket.

Roll değeri 1000, 2000 aynı alanda kalacak ama 1500 damaki *Roll* eşittir 0 hiç dönme yok, 1000 max sağa hareket ve 2000 max sola hareket.



Şekil 12. *Yaw*, *Pitch* ve *Roll* Standart Yönleme

1.3.2. MPU Değerleri Almak

MPU (Motion Processing Unit) damak hareket işleme ünitesi, yanı onu her hangi bir ederse uyumlu bir veri verecek. MPU verdiği değerleri quacoter için bir refrens gibi olacak.

MPU sensörü bu projede modeli MPU6050 Üç Eksenli bir jiro (Gy) ve Üç Eksenli bir akselerometre (ACC) sahiptir.

MPU ana amacı Pitch ve Roll açıları almaktır.

1.3.2.1. Jiro

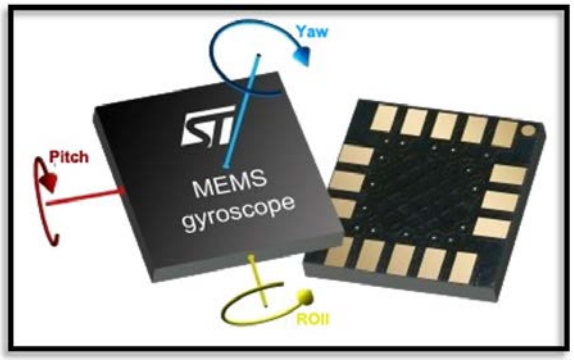
bir cismin ekseni etrafında dönme oranı ölçer RPM (dakikda devir) ile hesaplanır. MPU-6050 veri sayfasına göre bir parametresi var o Tam Ölçekli Aralık, dört değer alabilir (±250, ±500, ±1000, ±2000 °/s) ve onunla bağlı dört Hassasiyet Ölçeği Faktörü var (131, 65.5, 32.8, 16.4), bu projede Tam Ölçekli Aralık ±500 olacak ve bu halde Hassasiyet Ölçeği Faktörü 65.5

olacak. jironun değei her okumakta Tam Ölçekli Aralığna bölerek jiro değeri (º/s) olacak, şekil 13.

Kullanmadan önce jironun kalibre edilmesi gerektiğini bilmek önemlidir. Bu, sadece çoklu okumalardan ortalama sapmayı alıp bunu yeni değerlerden çıkartarak yapılır. 2000 gyro okumalarının ortalamasını hesaplanacak ve daha sonra kullanmak için bu değerleri ofset olarak saklanacak .

kalibre ettilten sonra jiron verdiği verileri okumaya başlayacağız ama quadcopter'in üç eksen etrafında yaptığı dönme oranı ölçmek değil yaptığı açıları bilmemiz gerekiyor. Jironun bütün değerlerin zamana eklenmesiyle, entegrasyon denir, jironun toplam ettiği açıyı hesaplamak mümkündür. Bu projede yenileme hızı 250Hz veya her 4us'dur. Bu yüzden her bir 4us'un jironun çıkışı entegre olacak. Bu halde jironun değei her okumakta Tam Ölçekli Aralığna(65.5) hem de yenileme hızıa(250) bölenecek, Pitch ve Roll açıları jironun Gx ve Gy değerlerinden hisapalanabilir.

Kodu ise iki kütüphane kullanılacak MPU6050.h parametreleri ayarlamak ve Wire.h MPU'dan okumak için. Aşağdaki kod yukardaki açıkaladığım noktaları temsil edecek, yanlız bazı fonksyonlar içindek kodlar burada değiller.



Şekil 13. Fizik olarak akselerometre çalışması

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

#define MPU6050_GFS_500 0x01

void setup() {
  Wire.begin();
  Serial.begin(57600);

  mpu.setFullScaleGyroRange(MPU6050_GYRO_FS_500);

  get_offset();

  loop_timer = micros();
}

void loop(){
  read_mpu();
  gx -= gx_offset;
  gy -= gy_offset;
  gz -= gz_offset;
  pitch_gyro += gx / 250 / 65.5;
  roll_gyro  += gy / 250 / 65.5;
}
```

Şimdi biz jirodan Pitch ve Roll açıları alabildik ama iki sıkıntı var jirode

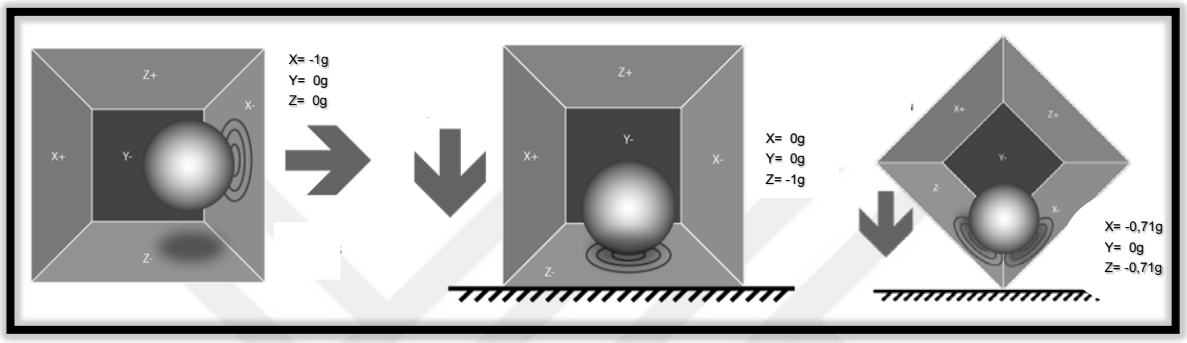
- Sürüklenme: Jironun MPU çiplerde zamanla bir açının sürüklenmesi yaşıyor.
- Varsayalan değer: MPU açılı bir yüzeyde başlatılır ve seviyesinin ne olduğu ile ilgili bir referansı yoktur ve sıfır olarak alınır. Bu sorun ancak başlangışta MPU italik olmayan bir yüzey üstüne koymak.

1.3.2.2. Akselerometre

Akselerometre yönü ve doğrusal hareketi ölçer (m/s^2). Bütün yönlere doğrusal hareketi fark edip hız ve yön verebilir bir sensördir.

Fizik olarak akselerometre nasıl çalıştığını anlamak için bir örnek vereceğim, bir küp merkezde bir top tutar ve top, bu dizelerle küpün her tarafındaki sensörlere bağlanır şekil 14.

Yerçekimi, toprağın yüzeyine dik olan topu hızlandırmaya çalışır. Topun kütlesi olduğundan ve bu küpün içinde sabitlendiğinden, bir kuvvet yaratılır. Bu kuvvet ayrıca yüzeyine diktir. Bağlı diziler, topun düşmesini önlemek için ters yönde bir kuvvet yaratacaktır. Bu, akselerometre sensör tarafından ölçülen bu kuvvettir ve bu her eksen için yapılır.



Şekil 14. Fizik olarak akselerometre çalışması

Akselerometre sensörlerin çıkışı, yerçekimi kuvvetinin yönüne dayanır. Böylece MPU-6050 sensörü kullanarak açıların değerlerini hesaplamak mümkündür. MPU-6050 veri sayfasına göre bir parametresi var o Tam Ölçekli Aralık, dört değer alabilir (±2, ±4, ±8, ±16 g) ve onunla bağlı dört Hassasiyet Ölçeği Faktörü var (16,384, 8,192, 4,096, 2,048), bu projede Tam Ölçekli Aralık ±8 olacak ve bu halde Hassasiyet Ölçeği Faktörü 4,096 olacak. *Pitch* ve *Roll* değerleri hesaplamak esnasında kendi değerleri arasında bir bölme nedeniyle bu Faktörü ile bölme olmayacak.

Pythagorean teorem'e göre *Pitch* ve *Roll* değerleri hesaplamak için ilk önce toplam yerçekimi vektörü hesaplaması sonrası (asin) fonksyonu ile *Pitch*, Y eksenle akselerometre değeri ile toplam yerçekimi vektörü bölerek bulunur, *Roll* ise, X eksenle akselerometre değeri ile toplam yerçekimi vektörü bölerek bulunur. Aşağdaki kodu temsil eder.

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

#define MPU6050_ACCEL_FS_8 0x02

void setup() {
  Wire.begin();
  Serial.begin(57600);

  mpu.setFullScaleAccelRange(MPU6050_ACCEL_FS_8);
}

void loop(){

  read_mpu_6050_data();

  calculate_acc();

  acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));

  pitch_acc= asin((float)acc_y/acc_total_vector)* 180/PI;
  roll_acc = asin((float)acc_x/acc_total_vector)* -180/PI;
}
```

1.3.2.3. Kullanılabilir Değerleri Çıkartmak

Akselerometre ve jirodan *Pitch* ve Roll açıları alabildik ama bir sıkıntısı var

- Akselerometre titreşimlere çok duyarlı ve motorları çalıştığında değerinlerini bir bozulma yaşıyor.
- Jiro değerleri zamanla değişiyor, bir sürüklenme yaşıyor ve başladığında düz yerde olması gerek ki başlangıç değerlerinde yanlış olmasın o da zayıflık bir noktadır.

Bu halde *Pitch* ve *Roll* açıları ne jirodan ne de sadece akselerometreden alınacak, ikisindendir, onun için :

- Önce, *Pitch* ve *Roll* açıların başlangıç değerleri akselerometreden alınacak çunkü akselerometrenin değerleri cesmin bulunduğu seviyele alakası yok ve düz olmasına gerek yok hem de sabit olduğu için titreşime yok.
- Sonra, akselerometrenin titreşimini ve jironun sürüklenmesini atlamak için *Pitch* ve Roll değerleri yüzde 90 jiroden ve yüzde 10 akselerometreden alınacak.

Son olarak *Pitch* ve *Roll* için LPF filtre uyglacağız. Aşağdaki kod bunları acıklar.

```
if(angles_seted){
  pitch = pitch_gyro * 0.9 + pitch_acc * 0.1;
  roll = roll_gyro * 0.9 + roll_acc * 0.1;
}
else{
  pitch = pitch_acc;
  roll = roll_acc;
  angles_seted = true;
}

// low pass filter
 pitch= pitch_old + alpha * (pitch - pitch_old);
 roll= roll_old + alpha * (roll - roll_old);

pitch_old=pitch;
roll_old=roll;
```

1.3.3. RC ile MPU PID Hesaplamak

PID kullanmak amacı hemen olamayan yada hemen yapılması istenmeyen bir yapı zamanla ve hesaplanmış adımlarla yapılır.

PID yapısı ise istediğimiz durum, şimdiki durumu ile zamanla yeni bir ara durum bulur ,bazı parametrelere ile bu hesaplamayı istediğimiz ve şimdiki durumu aynı olana kadar devam edecek. İki girişi var biri istediğimiz açıyı girişi diğeri şimdiki açıyı , bir cıkışı var ara açı değeridir.

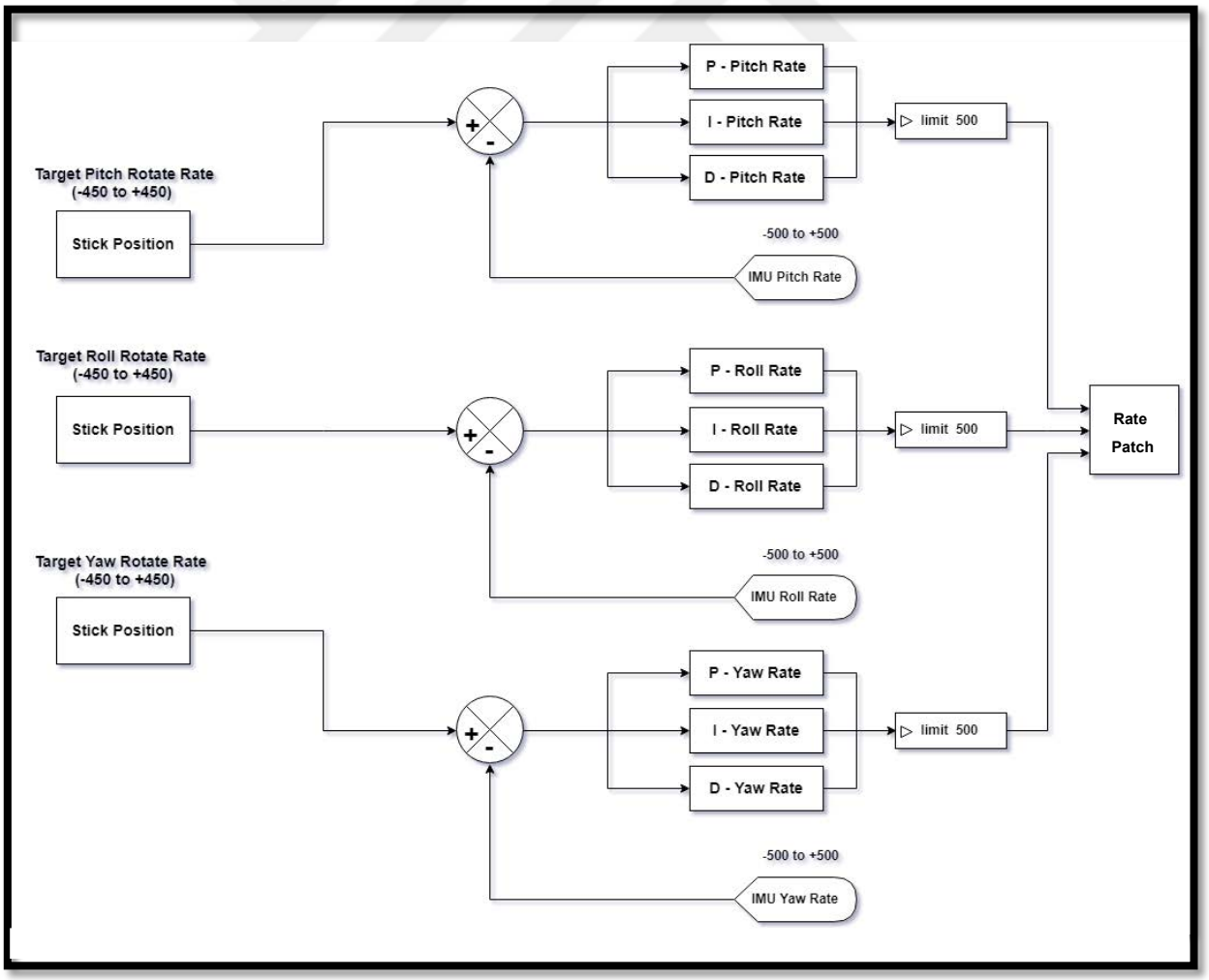
Quadcopter istediğimiz yöne ve açıya hemen dönemez ancak, bir zaman alacak ve onun için PID kullanması gerek, ancak yapılacak işlemler uçuş moduna göre değişiyor. Bu araştırmada iki manul uçuş modu kullanacağım.

.

1.3.3.1. Rate Modu

Vericideki çubukların quadcopter'in belirli bir hızda (ör. 50 deg/sn) dönmesini söyler, ve çubukları merkeze döndüğüde, quadcopter'in dönmesini duracak ancak mevcut tutumunu koruyacak ve seviyesine dönmeyecektir. Bu uşuçta sadece jiro değerleri lazım.

Aşağıdaki Rate modunun çalışma akış şeması şekil 15 gösterdiği gibi RC'dan aldığımız *Pitch*, *Roll* ve *Yaw* değerlerin alanları 1000, 2000 alanından -450, +450 alanına dönüştürme olacak, bu halde bu değerleri istenilen dönme hız oranı temsil edecek. Quadcopter'e aynı istediğimiz dönme hız oranı hemen verilmez ,yoksa bir titreşim yaşacak çünkü quadcopter 1 deg/s ile döndü zaman heme istedimiz 100 deg/sec hemen dönmez ve bunun için ilk önce IMU'dan şimdiki dönme hız alacağız sonra bir PID kullanacagız.



Şekil 15. Rate Modu için çalışma akış şeması

Bu PID'nın ilk girişi RC'dan istenelen dönme hızı ve diğeri IMU'dan şimdiki döme hızdır. P,I ve D parametreleri ayarlarak yeni dönme hızı elde edeceğiz ve bazan PID çıkış büyük olur ve onun için bir limit koyması lazım. *Pitch*, *Roll* ve *Yaw* aldımız yeni değerleri bir denklam ile motorlar iletilecek.

Bu rate modu normalde kolay değil çünkü bir sağa yada sola dömek için çubukları ilk önce bir yan çekeceğiz daha sonra çok dömemek için sabit tutacağız ve sonra istediğimiz yönde olduğu zaman çubuklar ters yönde dönme hızı sıfır olana kadar çekeceğiz.

Aşağdaki kodu bu modu açıklar

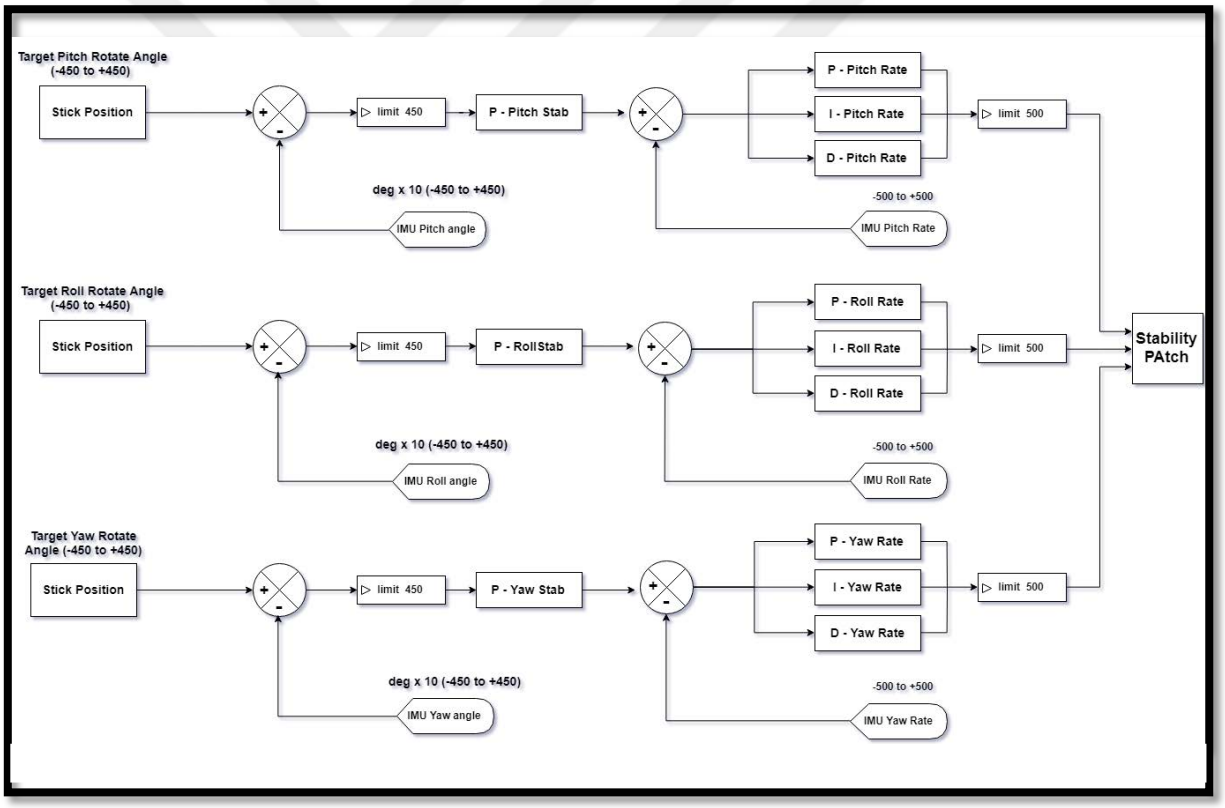
```
// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
pid_Roll_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Roll olamasın.
  if(receiver_input_channel_1 > 1508)pid_Roll_setpoint = (receiver_input_channel_1 - 1508);
  else if(receiver_input_channel_1 < 1492)pid_Roll_setpoint = (receiver_input_channel_1 - 1492);
}
// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
 pid_Pitch_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Pitch olamasın.
  if(receiver_input_channel_2 > 1508)pid_Pitch_setpoint = (receiver_input_channel_2 - 1508);
  else if(receiver_input_channel_2 < 1492)pid_Pitch_setpoint = (receiver_input_channel_2 - 1492);
}
// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
 pid_yaw_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Yaw olamasın.
  if(receiver_input_channel_4 > 1508)pid_yaw_setpoint = (receiver_input_channel_4 - 1508);
  else if(receiver_input_channel_4 < 1492)pid_yaw_setpoint = (receiver_input_channel_4 - 1492);
}

calculate_pid();
```

1.3.3.2. Stable Mode

Pilotun *pitch* ve *roll* girişi, quadcopterin açısını kontrol eder. Pilot, *roll* ve *pitch* çubuklarını bıraktığında araç otomatik olarak kendini dengeler. ANGLE_MAX, varsayılan olarak 450 şekil 16. (yani 45 derece) olan maksimum eğri açısını kontrol eder.

Pilot'un *yaw* girişi, başlığın değişim oranını kontrol eder. Pilot *yaw* çubuğunu serbest bıraktığında araç mevcut rotasını koruyacaktır.

Pilot'un *throttle* girişi, ortalama motor hızını kontrol eder, yani rakımı korumak için throttlenin sabit ayarlanması gerekir.



Şekil 16. Stable Modu için çalışma akış şeması

Roll / *Pitch* P, quadcopterin *roll* ve *pitch* pilot girişine olan duyarlılığını ve istenen ve gerçek *roll* ve *pitch* açıları arasındaki hataları kontrol eder. 4.5 varsayılanı, açıdaki her 1 derecelik hata için bir 4.5deg / sn dönüş hızı komutu verecektir.

- 7 veya 8 gibi daha yüksek bir kazanç daha duyarlı bir quadcopter olmasına.
- Düşük bir stabilize edici P, quadcopterin çok yavaş dönmesine neden olur ve quadcopterin tepkisizleşmesine neden olabilir.

Rate *Roll*/*Pitch* P, I ve D terimleri, komandadan istenen dönüş hızına dayanarak motorlara giden çıkışı kontrol eder. Bu terimler genellikle, düşük hızda PID değerleri gerektiren daha güçlü kopyalayıcılara sahip olan quadcopterin güç / ağırlık oranına ilişkindir.

- Rate *Roll* / *Pitch* P, quadcopterine için doğru ayarı yapmak için en önemli değerdir.P yükseldikçe istenen dönüş hızını elde etmek için motor tepkisi artar. Standart quadcopter için varsayılan P = 0.15'tir.
- Rate *Roll* / *Pitch* I, quadcopterini istenen hızı daha uzun süre korumaz hale getirecek dış kuvvetleri telafi etmek için kullanılır. Yüksek I terimi, istenen oranı elde etmek için hızlı bir şekilde rampalaşacak ve aşınmayı önlemek için hızlı bir şekilde rampalanacaktır.
- Rate *Roll* / *Pitch* D, quadcopterin istenen ayar noktasına doğru hızlanmalara tepkisini azaltmak için kullanılır. Yüksek D, alışılmadık titreşimlere ve kontrollerin yavaş veya tepkisiz gibi hissettirdiği bir “hafıza” etkisine neden olabilir. Düzgün monte edilmiş bir kontrol cihazı, 0,011 D Hızı değerine izin vermelidir.

Aşağdaki kodu bu modu açıklar:

```
// level adjust deg * 10  değeri -500 to +500 aralarında olmak üzere
pitch_level_adjust = angle_pitch*10;
roll_level_adjust = angle_roll*10;


// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
pid_Roll_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Roll olamasın.
  if(receiver_input_channel_1 > 1508)pid_Roll_setpoint = (receiver_input_channel_1 - 1508);
  else if(receiver_input_channel_1 < 1492)pid_Roll_setpoint = (receiver_input_channel_1 - 1492);
}
pid_roll_setpoint -= roll_level_adjust;

// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
 pid_Pitch_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Pitch olamasın.
  if(receiver_input_channel_2 > 1508)pid_Pitch_setpoint = (receiver_input_channel_2 - 1508);
  else if(receiver_input_channel_2 < 1492)pid_Pitch_setpoint = (receiver_input_channel_2 - 1492);
}
pid_Pitch_setpoint -= Pitch_level_adjust;

// Derece / saniye  PID setpoint, alıcı RC girişi tarafından belirlenir.
 pid_yaw_setpoint = 0;
//Daha iyi sonuçlar elde etmek için küçük bir ölü 16us değeri almayacağız.
if(receiver_input_channel_3 > 1050){ //Motorları çalışmadıkları zaman Yaw olamasın.
  if(receiver_input_channel_4 > 1508)pid_yaw_setpoint = (receiver_input_channel_4 - 1508);
  else if(receiver_input_channel_4 < 1492)pid_yaw_setpoint = (receiver_input_channel_4 - 1492);
}

calculate_pid();
```

PID sonuçları pid_output_pitch , pid_output_roll ve pid_output_yaw ve diğer değeri throttle, bunları motorlara uygun bir şekilde aktarılacak ancak quadcopter'in çalışma durumunun bir kaç durum tanımlanacak.

1.3.4. Motorlara Dönme Hızı Vermek

Quadcopter motorları maksimum dönebilmesi için ESC'lara 2000 us bir darbe verilmesi gerekken durdurmak için 1000 us verilmesidir.

1.3.4.1. ESC'ye Dönme Hız Değeri Verme

ESC yenileme hızı (50-400Hz) arası olabilir biz 250Hz seçeceğiz yani her 4000us ESC'lar yeni değeri girişlerinden alıyour, bunu için motorların istenelen hızı bu 4000us aralıgında bulnması gerek ki hemen ESC'lara aktarılmalıdır. Bir zamanlama kodları ile eğer motorların istenelen hızı erken bulunursa program 4000us dolmasına kadar bekleyecek.

ESC'ye verilecek dönme hız değeri şekil ise, girişine 1 veriliyor ve istenilen dönme hızı kadar bir zaman tutuluyor sonrası 0 veriliyour. Ancak ESC'ların en uzun değerli zamanı bitine kadar program sadece ESC'ye değerleri aktarıyou ve bittiğinde yeniden programın hesapları işlenecekt.

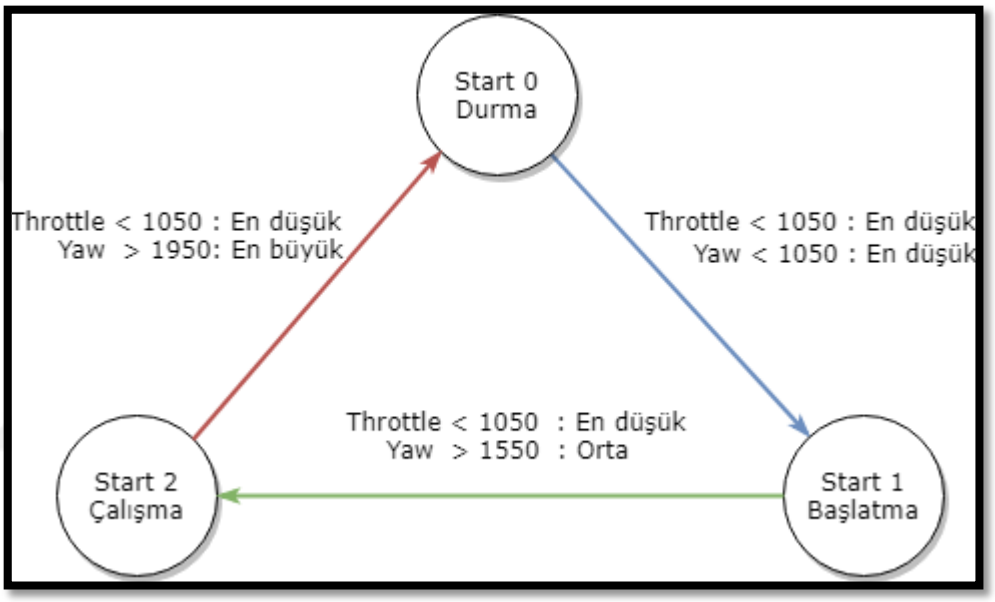
Motorlar her zaman dömeyecek, bazan duruması gerek bazan çalışması da gerek olacak, onu blirleten quadcopter çalışma durumdur.

```
if(micros() - loop_timer > 4050)digitalWrite(12, HIGH);

//The refresh rate is 250Hz..
while(micros() - loop_timer < 4000);
loop_timer = micros();

PORTD |= B11110000;
timer_channel_1 = esc_1 + loop_timer;
timer_channel_2 = esc_2 + loop_timer;
timer_channel_3 = esc_3 + loop_timer;
timer_channel_4 = esc_4 + loop_timer;


while(PORTD >= 16){
  esc_loop_timer = micros();
  if(timer_channel_1 <= esc_loop_timer)PORTD &= B11101111;
  if(timer_channel_2 <= esc_loop_timer)PORTD &= B11011111;
  if(timer_channel_3 <= esc_loop_timer)PORTD &= B10111111;
  if(timer_channel_4 <= esc_loop_timer)PORTD &= B01111111;
}
}
```

1.3.4.2. Zamanlama ve esc çalışma durumda nasıl alır

Quadcopter çalışma durumlarına göre motorına ayrı bir şekilde dönme hızı verilir, bunun için ilk önce quadcopter'in çalışma durumları tanımlayacağız.

Çalışma durumları üç durum olarak tanımlanacaktır. Şekil 17 bu çalışma durumları açıklıyou.



Şekil 17. Quadcopter çalışma durumları

- Start 0 - Durma Durumu

Bu durumda quadcopter duruyor ve motorlara 0 dönme hızı verilir yani *Pitch* yada *Roll* ne olursa olsun bütün ESC'lara 1000 us veriyor. Bu varsayılan durumu, quadcopter çalıştığımızda bu durumdan başlıyor.

Code olarak ESC'lara isenelen dönme hızı vereceğmiz zaman quadcopter durumuna bakarak eğer start 0 .

Bu durumda eğer RC'den bu iki değer aynı anda gelirse quadcopter durumu start 1'e çeverilecek:

Throttle çubuğu aşağıya çektik yani en düşük (1050 us) kadar,
Yaw çubuğu sola çektik yani en büyük (1950 us) kadar.

- Start 1 - Başlatma Durumu

Bu durumda quadcopter hala duruyor ve motorlara 0 dönme hızı veriliyor da veriyor ama bu durum blunmanın ana amaclar:

PID parametreleri sıfırlamak,
MPU'dan *pitch* ve *roll* aşıları ilk önce akselerometreden almak.

Bu durumda eğer RC'den bu iki değer aynı anda gelirse quadcopter durumu start 2'e çeverilecek:

Throttle çubuğu hala aşağıya çekli yani en düşük (1050 us) kadar,
Yaw çubuğu soldan ortaya döndu çektik (1550 us) kadar.

- Start 2 - Çalışma Durumu

Bu durumda quadcopter motorlarını hesaplanmış dönme hızı alır ve istenilen hareketi etmeye başlar.

Bu durumda eğer RC'den bu iki değer aynı anda gelirse quadcopter durumu start 0'e çeverilecek:

Throttle çubuğu aşağıya çekli yani en düşük (1050 us) kadar,
Yaw çubuğu sağa çektik yani en düşük (1050 us) kadar.

Aşağdaki kodları hem çalışma durumları bilirtir

```
//For starting and initilize: throttle low and yaw left, move to (start 1).
if(receiver_input_channel_3 < 1050 && receiver_input_channel_4 > 1950)start = 1;
//For the running mode: yaw stick is back in the center position, move to (start 2).
if(start == 1 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 > 1550){
  start = 2;
  //set the angle from accelerometr for starting
  angle_pitch = angle_pitch_acc;
  angle_roll = angle_roll_acc;
  gyro_angles_set = true;

  //Reset the PID controllers for a bumpless start.
  pid_i_mem_roll = 0;
  pid_last_roll_d_error = 0;
  pid_i_mem_pitch = 0;
  pid_last_pitch_d_error = 0;
  pid_i_mem_yaw = 0;
  pid_last_yaw_d_error = 0;
}
// Stop mode: throttle low and yaw right, return to (start 0)
if(start == 2 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 < 1050)start = 0;
```

1.3.4.3. Amirler Kombinasyon

Son olarak ve bütün geçmiş hesapların sonucuna göre dört değer çıkıyor: *Throttle*, *Yaw*, *Pitch* ve *Roll*. Bu değeler motorlara aktarabilmek için açağdaki formulalar kullanacak:

MR_1 = throttle - pitch - roll + yaw (1)

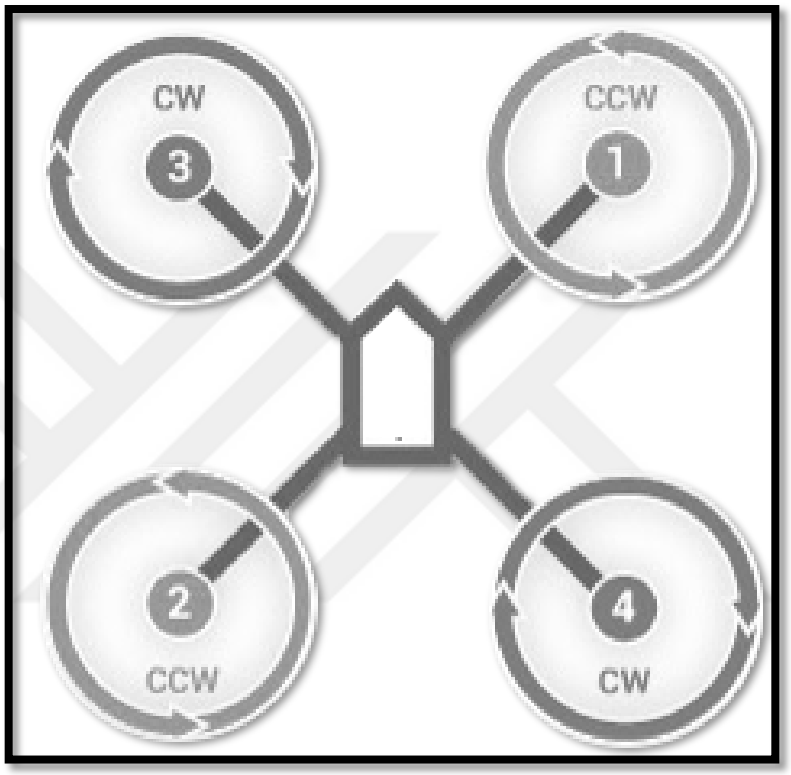
MR_2 = throttle + pitch + roll + yaw (2)

MR_3 = throttle - pitch + roll - yaw (3)

MR_4 = throttle + pitch - roll - yaw) (4)

(1), (2) ,(3) ve (4) formulalar aşağdaki şekil.18 gösterdiğine göre yazılmıştır ve aşağda açılanacaktır:

- THR değerine göre yükselme oluyor ve onun için bütün motorlar uygulanacak.



Şekil 18. Motorlar durumu

- *YAW* değeri pozitif ise MR_1, MR_2 motorları daha hızlı döner ve MR_1, MR_2 aynı çapta bulunduğu için quadcopter döndikleri yönüne döner yani CCW yönde, negatif ise MR_3, MR_4 motorları daha hızlı döner ve MR_3, MR_4 aynı çapta bulunduğu için quadcopter döndikleri yönüne döner yani CW yönde.
- *PİTCH* değeri pozitif ise MR_1, MR_3 motorların hızını azalanacak ve MR_2, MR_4 motorların hızını artıyacak bu çekilde quadcopter önü aşağıya eğilim edip ileriye gider, PİTCH değeri negatif ise quadcopter arkaya gider.

- *ROLL* değeri pozitif ise MR_1, MR_4 motorların hızını azalanacak ve MR_3, MR_2 motorların hızını artıyacak bu çekilde quadcopter sağa eğilim edip gider, PİTCH değeri negatif ise quadcopter sola eğilim edip gider.

1.3.5. Zamanlama ve Code Optimizasyonu

Arduino işlediği işlemlerin süreleri var, arduino ile pağlayan MPU, ESC ve RC alıcı olan parçaların okuma ve yazma süreleri var ve yenileme hızı var, yanı eğer bu süreleri hesaplarsak ve okuma ve yazma fonksiyonları yenileme hızlarına dikkat edersek bir önceliklendirme yaparsak hem hataları silinecek hem de daha hızlı işlenecek.

1.3.5.1. Program Ana İşlemlerin Zamanlaması

- RC değerleri almak (RC interrupt oldüğü için blirli bir süre alamaz).
- MPU değerleri almak, yaklaşık 300us alır.
- PID hesaplamak ve diğer işlemler, yaklaşık 600us alır.
- ESC'lere değerleri verme, maksimum 2000us alabilir.

Program yenileme hızı 250hz diye alınırsa yani program döngüsü, 4000us bir zaman alır, bu halde her program döngüsüda

(4000us – 300us – 600us – 2000us =1100us)

Bu 1100us boş zaman olan RC interrup kullanablir.

1.3.5.2. Code Optimizasyonu

Code optimizasyonu alınacak alanı azaltmak için ve hızı yükseltmek için kullanacaktır.

1.3.5.2.1. Alanı Azaltmak

- iki porta çalışma modu vermek için eğer Arduino emri yerine Atmega amri kullanırsak çok az alan kullanılacak:

```
DDRB |= B00110000 ; // Atmega amri 12,13 portlar çıkış olarak kullanacak - 50k
/*---------*/
```

```
pinMode(12,OUTPUT); // Arduino amri 12 portu çıkış olarak kullanacak - 100k
pinMode(13,OUTPUT); // Arduino amri 13 portu çıkış olarak kullanacak - 100k
```

1.3.5.2.2. Hızı Yükseltmek

- iki porta çıkış değeri vermek için eğer Arduino emri yerine Atmega amri kullanırsak çok az zaman yanı çok hızlı olacak:

```
DDRB |=  B00110000 ; // Atmega amri 12,13 portlara 1 vermek – 50ns
DDRB &= B11001111 ;
/*---------*/
pinMode(12,OUTPUT); // Arduino amri 12,13 portlara 1 vermek – 4us
pinMode(13,OUTPUT);
```

- Delay kullanmak yerine bir sayıcı kullanmak performans olarak daha yüksek çünkü interrup kullanıldığı için bir zamanlamda eşitsizlik olacak :

```
delayMicroseconds(timer_ms); // interrupt çalıstığında zamanlamda eşitsizlik oluyour
/*---------*/
timer = micros()+ timer_ms;          // aynı zaman bekleyecek ama zamanlamda eşitsizlik yok
while(micros() < timer );
```

- ESC'ya dönme hızı verdiğimiz zaman 1000us bir bekleme olacak çünkü alacağı 1 logic süresi 1000 ile 2000 arası, onun için bu 1000us aranında giro değerleri okumakla kullanılacak.

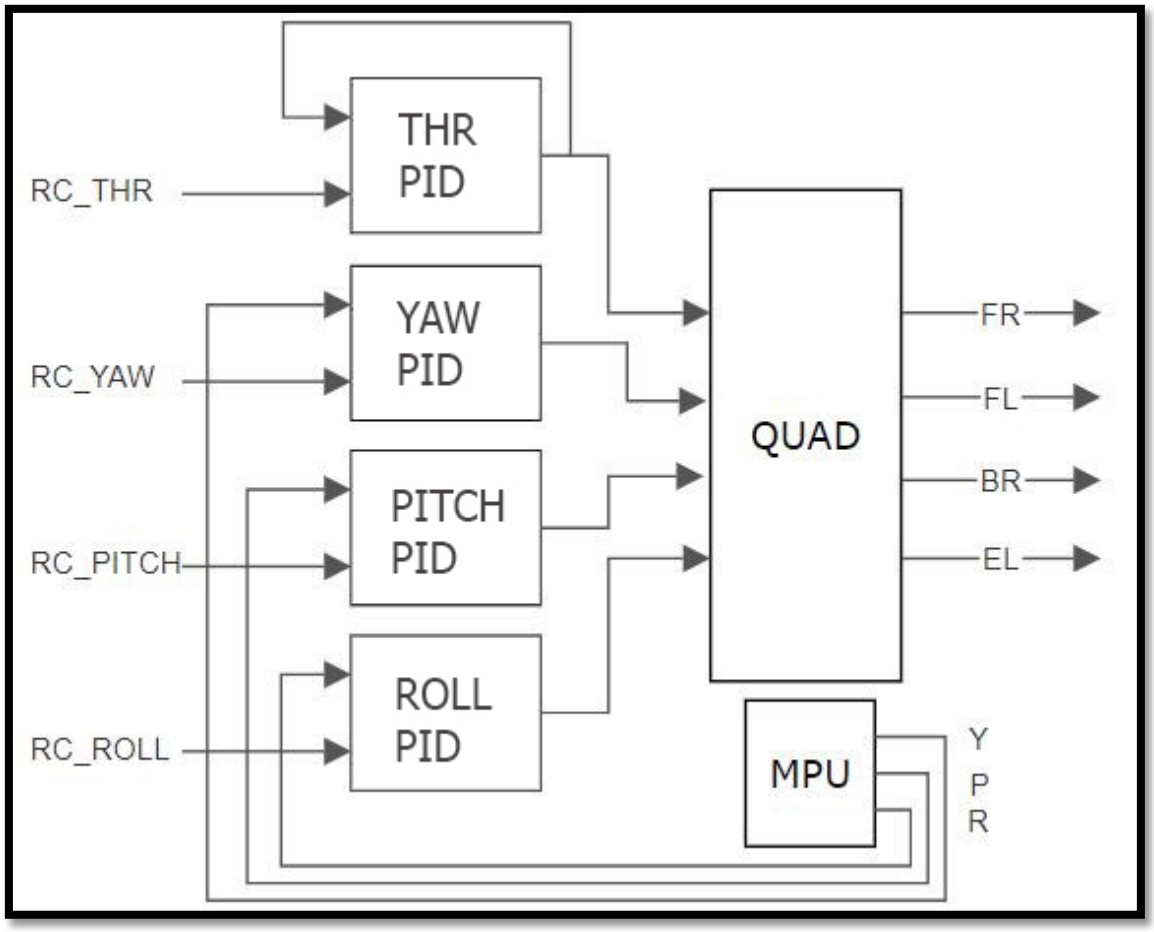
1.4. Otmatik Uçuş Kontrolörü Sistemi

Otomatik uçuş yalnız kumandadan RC değerine ihtiyacımız kalmıyor, ancak RC değerinin yerinde otomatik olarak uçuş kontrol eden değerleri (*Throttle*, *Pitch*, *Roll*, *Yaw*) üretilecek, onu yapabilmek için GPS VE Barometre kullanmasını gerek olacaktır. Otomatik uçuş yapmak için quadcopter'e hedef noktası verilmesi gerekiyor, hedef noktası iki şekilde:

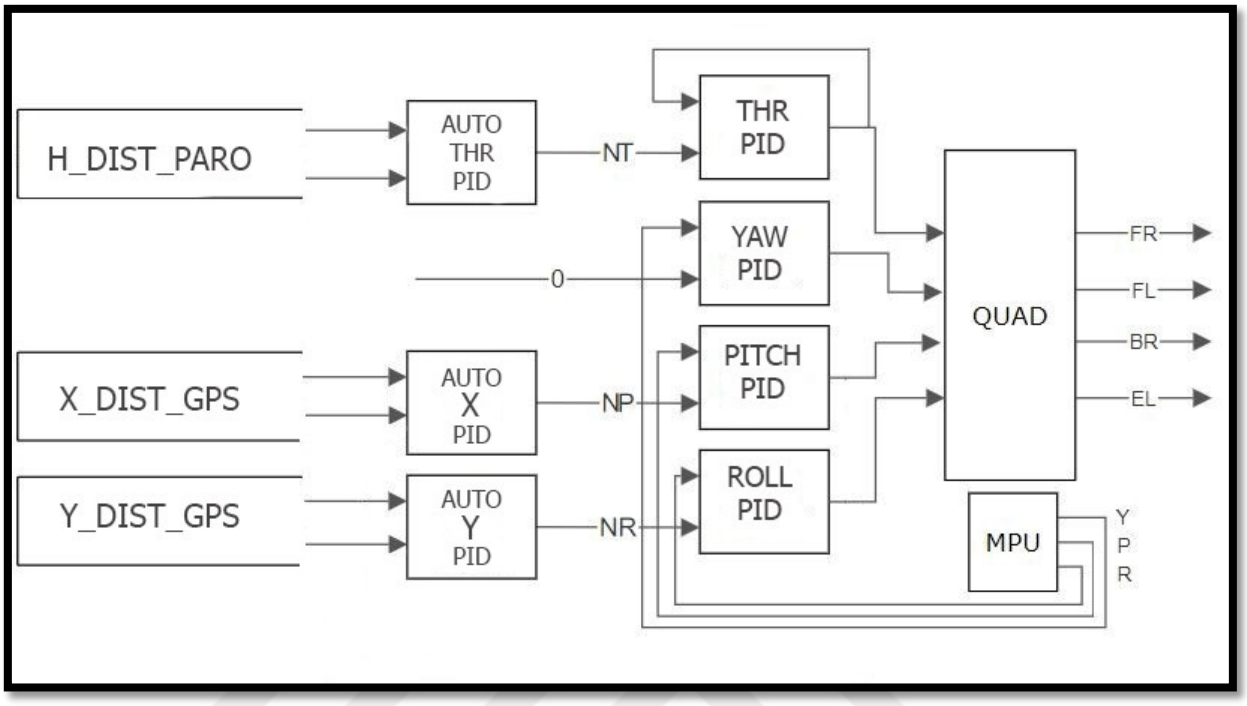
- Hedef noktası daha önce rota ile bilirlenmiş
- Hefed noktası dinamik olarak kamiradan yada harekt eden bir hedef konumundan.

Quadcopter, Barometre sensöründen okunan değerlere dayanarak bilinen yüksekliğe ulaşmak için önce kalkacak, daha sonra GPS algılayıcısının okumalarına dayanarak sürekli olarak değişen rulo ve eğim açılarıyla bilinen hedef konuma gidecektir, Bundan sonra dört açıcı yere inecek.

Manuel üçuşte ise genel çalışmas aşağdaki şekil.19 göstediği gibi oldu, ama otomatik haline gelince şekil.20 çalşması gösterecek.



Şekil 19. Manuel çalışma şekli



Şekil 20. Otomatik çalışma şekli

Bu otomatik uçuşu için lazım olan GPS ve Barometre okumalarını ve kullamnmalarını ilk önce açıklanacak

1.4.1. Otomatik Üçüş İçin Lazım Olan Değerleri Okumak

1.4.1.1. Gps Okuma

GPS gihazı UBLOX kullanacatır ve bu gıhaz Arduino'ya aşağıdaki gibi bağlayın:

GND ==> GND

TX ==> Dijital pin (D3)

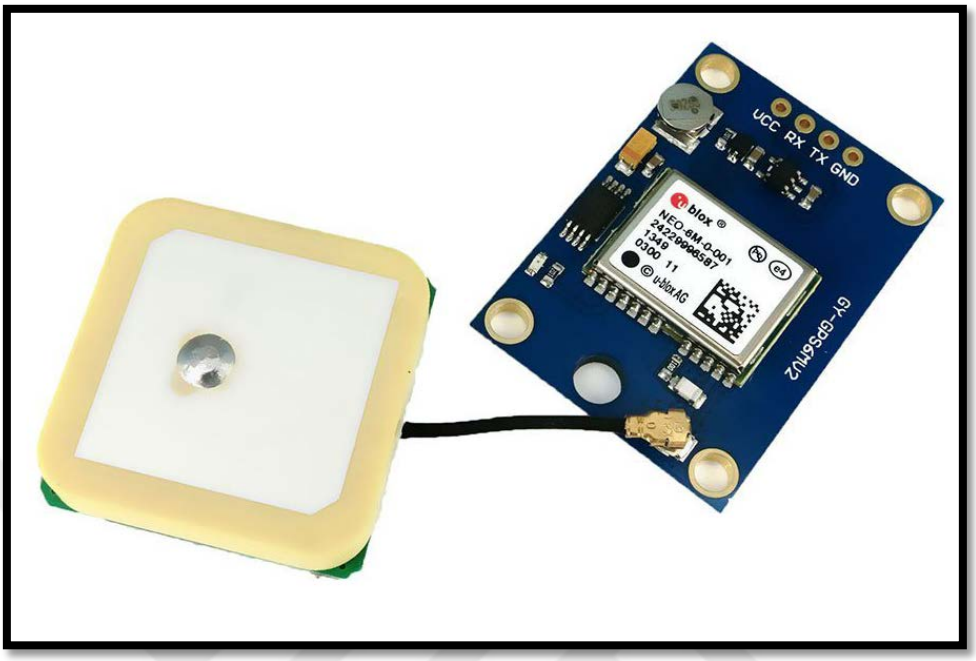
RX ==> Dijital pin (D4)

Vcc ==> 3,3 V

Bu gıhaz bir kaç parameter hisaplar ancak bize lazım olan iki tanetir: **Enlem**, **Boylam** bu iki parametre, quadcopter'in konumunu belirtir. Bir program kullanarak zaman aralığında gelen seri okuyoruz, eğer sonuç -1 işittir ise damaki daha gerçek değerler daha gelemedi, geçek değerleri geldiğinde bir diziye byte byte alıyoruz eğer geldiği byte değeri 13 işittir ise damak bu göderi bitmiş ve istediğimiz değerleri elde edilmiştirç.
bu dizi içinden enlam ve boylam değerleri alabilirs. Aşagıdaki kodu iki kütüphane kullanarak bu değerleri alabiliriz.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

while (gpsSerial.available() > 0)
        {
        if (gps.encode(gpsSerial.read()))
        {
                if (gps.location.isValid())
                {
                        Lat = gps.location.lat();
                        Lng = gps.location.lng();
                }
        }
}
```

Ve bu iki kütüphanesi kullanmadan üzün bir kod ile de Enlem ve Boylam değerleri alabiliriz, ben programımda çok az kütüphaneler kullandım ki başk bir işlemcide bu kod ullanabilmek, FPGA örnek olarak.



Şekil 21. UBLOX GPS gihazı

1.4.1.2. Barometre Okuma

BMP180 barometrik sensör kullanacağım, Etrafındaki havanın mutlak basıncını ölçer. 0.02 hPa'ya kadar doğruluk ile 300 ila 1100hPa arasında bir ölçüm aralığına sahiptir. BMP180 barometrik sensörü I2C arayüzü üzerinden haberleşir. Bu, sadece 2 pin kullanarak Arduino ile iletişim kurduğu anlamına gelir.

Pin: Arduino Uno'ya Kablolama

SCL: A5

SDA: A4

değerleri almak için SFE_BMP180.h kütüphanesi kullanılacak ve bu fonksıyonları işlenecek:

Tamamlanan sıcaklık ölçüsünü almak, başarılı olursa 1, başarısız olursa 0 döndürür.

```
Status = pressure.getTemperature(T);
```

Start a pressure measurement. The parameter is the sampling setting with 0 to 3 (highest resolution, longest standby).

```
Status = pressure.startPressure(3);
```

Tamamlanan basınç ölçümünü almak. ölçümün P değişkeninde saklandığını,

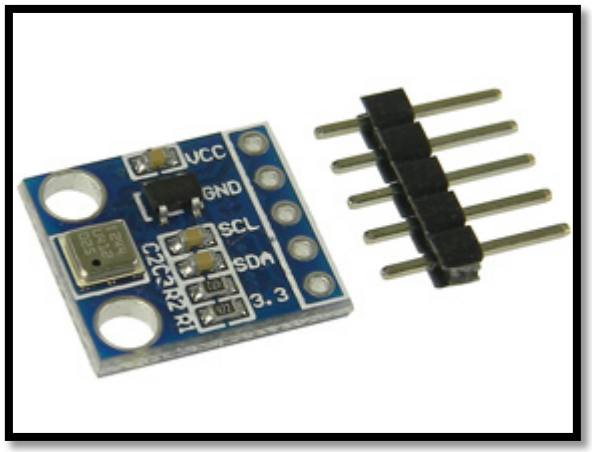
```
Status = pressure.getPressure(P,T);
```

Yüksekliğin etkilerini kaldırmak için, deniz seviyesi işlevini ve geçerli yüksekliği kullanmak.

```
p0 = pressure.sealevel(P, 1655.0);
```

a = yükseklik meter cinsinden

```
a = pressure.altitude(P,p0);
```



Şekil 22. Parometre gihazı

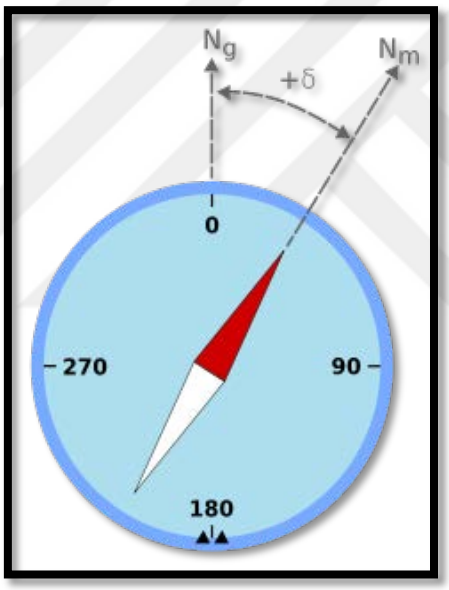
1.4.1.3. Pusula'dan (Compass) Yön Bilirlemek

Manyetometreler (Pusula) etrafındaki manyetik alanı ölçer ve cihazın kuzeydeki gibi sorulara cevap verir. Quadcopter, uzayda yönlendirmenin belirlenmesine yardımcı olmak için

bunu kullanır. Bu projede bi rota yapabilmek için yönü çok gerekli. Quadcopter bulunduğu lokasiyon hedef lokasiyon ile bir açı hesaplanacak, ancak bu açı, kuzey için fark değeridir. quadcopter'deki takılan pusulanın kuzeyi quadcopterin yönü ile takılacak ve bu şekilde quacopter'i bu açı sıfır olana kadar dönmeye devem edecek.

Pusulanın kuzeyden fark değeri alması iki aşamada yapacağız:

- Manyetik kuzeyi Nm hesaplaması.
- coğrafi (gerçek) kuzeyi Ng hesaplaması, manyetik sapma δ manyetik değerinden çıkardında.



Şekil 23. Pusulanın kuzeyden fark

coğrafi (gerçek) kuzeyi Ng, sapma δ, manyetik değeri Nm temsili.

HMC5883L şipi kullandım doğru değerler alabilmek için önce bir ayarlama programı ile ofset değerleri bulması lazm, sonra heading bulmak için heading formül kullanacak.
Bu halde Heading

$$\text{atan2}\left(\frac{mg.YAxis}{mg.XAxis}\right)$$

ama bu halde quadcopteri düz ve İtalik değil olması gerek yoksa değerinde hata olacak, bunun için başka bir formül kullandım, bu formül *Pitch* ve *Roll* açıların değerleri göz önünde koyuyor Heading:

$$\operatorname{atan2}\left(\frac{mg.ZAxis * \sin(roll) - mg.YAxis * \cos(roll)}{(mg.XAxis * \cos(pitch) + mg.YAxis * \sin(pitch) * \sin(roll) + mg.ZAxis * \sin(pitch) * \cos(roll))}\right)$$

Sonara 180 , -180 alana dönüş yaptıktan sonra LPF filtre uyglanacak .

1.4.2. Otomatik Üçüşü Simülasyonu

Automatic üçüşü için ilk önce bir simülasyon kullanılacak, bütün quadcopter'in parametreleri eklenecek ve otomatic bir rota ile uçumak için gerkli emirleri yazıp denerek quadcopter'e zarar olmadan olacak.

1.4.2.1. QRSim Quadrotors Simülatörü

Simülasyon için QRSim quadrotors simülatörü kullanacak, Quadrotor helikopter modellerinin örnekleri, yalnızca hava platformunun dinamik kullanım alanlarının çoğaltılmasına odaklanma eğilimindedir ve birincil kullanımları kapalı döngü uçuş kontrolü alanındadır.

1.4.2.1.1. QRSim Parametreleri

QRSim quadrotors simülatörü quadcopter'in bütün sensörlerine bir parametre var. Bu şekilde siymlatore, quadcopter'in tam yapısı uygluyor. Bu paramtreleri ayarlam imkani da var (açma, kapatma) , çevre alanı sınırları da ayralanabilir, parametreleri:

- GPS Alıcısı, gürültülü veya gürültüsüz konumu döndürür.
- IMU+Barometre, attitude - heading - altitude.

Quadcopter fizik olarak parametreleri da ayarlanabilir, bu paramtreler:

- AL kol uzunluğu m
- AW kol genişliği m

- Kol kalınlığı m
- BW gövde genişliği m
- BT gövde kalınlığı m
- R yarıçapı m

1.4.2.1.2. Qrsim Modül Fonksiyonları

QRSim quadrotors simülatörü bir kaç fonksiyon sahiptir, bu projeye lazım onlarından:

Durum, Herhangi bir zamanda simülatör, simülasyonun parçası olan tüm nesnelerin durumunu korumalıdır, Örnek, state.platforms{1}.getX() bu fonksiyonu ilk platformun durum değişkenlerine erişim sağlar. Verdiği değerleri:

- px, py, pz (pozisyon parametreleri)
- φ, θ, ψ (hız)
- u, v, w (oryantasyon)
- p, q, r (dönme hızı)

Controllers, çeşitli görevlerde genellikle yararlı olan basit denetleyicilerin temel uygulamalarını içerir. Örnek, WaypointPID bu fonksiyonu PID denetleyicisi oluştur.

Genel fonkisyonlar, quadcoptere lazım olan bazı eşlemleri için yapıldı. Örnek, llaToUtm bu fonksiyonu gps'ten (enlem, boylam) XY poszisyouna dönüştürür.

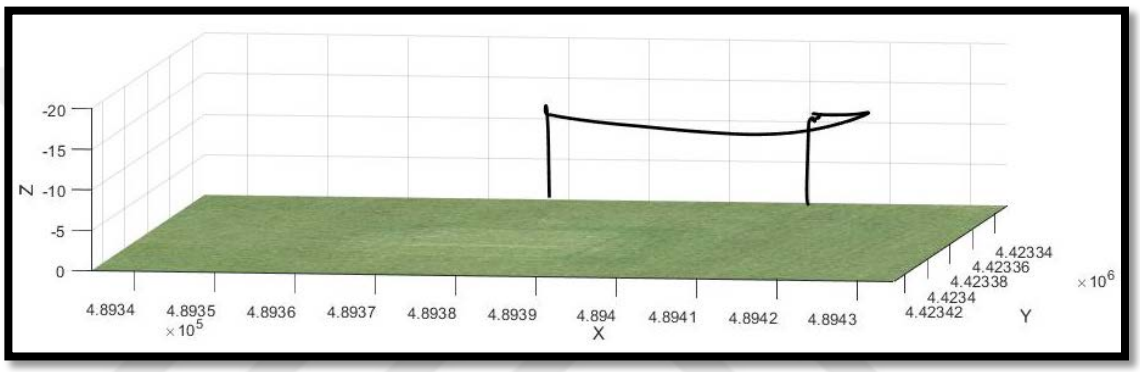
1.4.2.2. Otomatik Uçuş

Otomatik uçuş genellikle üç aşamadan oluşu: kalkış, uçuş sonra da iniş. Bu simülatörü ise bu üç aşamada bu şekilde gerçekleşti:

- Kalkış, istediğimiz irtifaya ulaştıktan sonra ve bilirlenmiş süre içinde bu irtifada kalmak.
- Uçuş, bulunduğu XY'dan hedef XY'e ulaştıktan sonra ve bilirlenmiş süre içinde bu pozisyonda kalmak.
- İniş, bulunduğu irtifadan 0 irtifaya ulaştıktan sonra olacaktır.

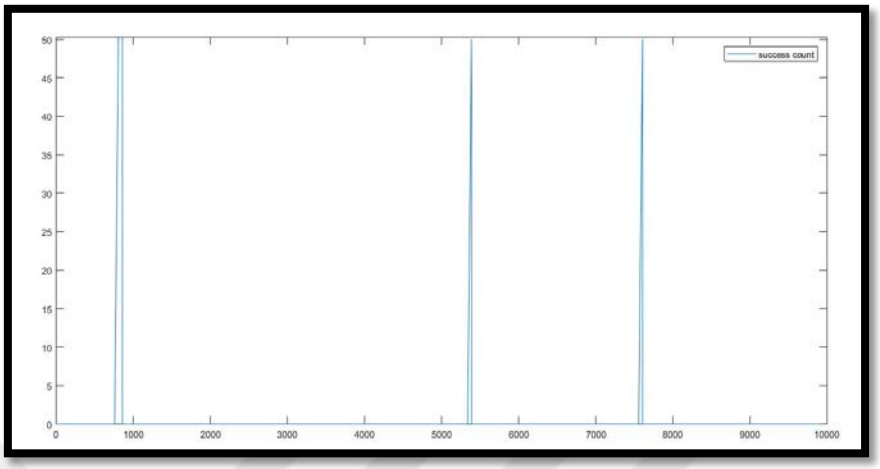
Bu programdaki bütün eğri çizimleri ile anlatılacak:

- Şekil .24, Üç buyotlu rota çizimi gösteriyour, bu şekilde üç haraket var (kalıkş, gidiş ve iniş) ve üç duruma var (kalkıştan sonra istenmiş irtifa varana kadar, gidişten sonra istenmiş hedef pozisyonu varana kadar ve inişten sonra yerde hafifçe varana kadar).



Şekil 24. Üç buyotlu rota çizimi

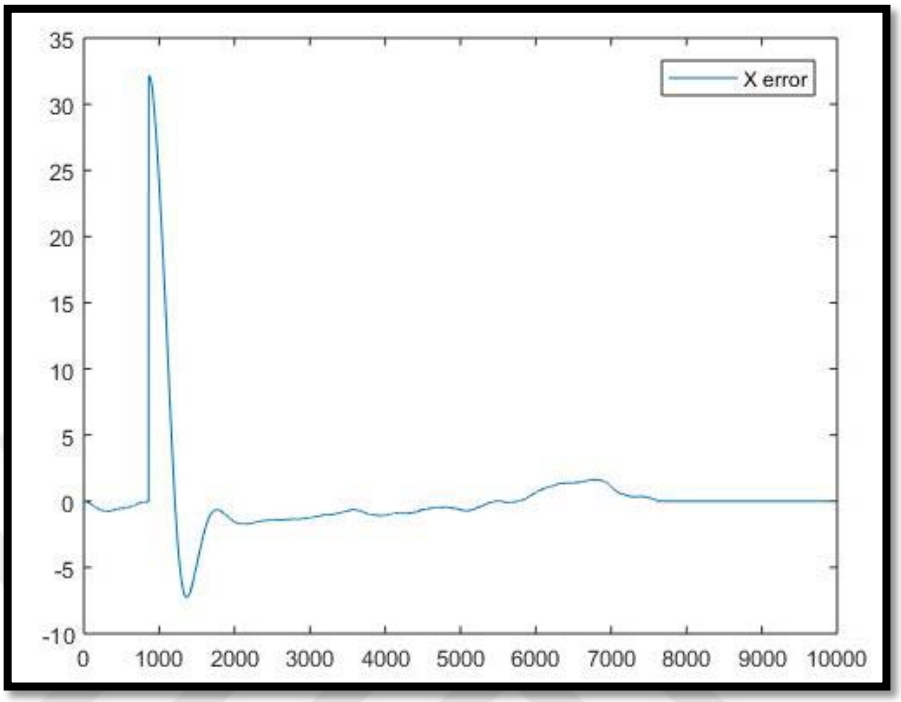
- Şekil .25, üç haraket ile arasındaki üç duruma bittiğini gösteriyour. Durum, 50 kere bütün hataları <0.2 olunca bitiyour. [0 s, 800] **kalkış**, [800 s, 5400 s] **gidiş**, [540, 760] **iniş**.



Şekil 25. Başarlı sayısı

1.4.2.2.1. X Eksanı İle Hata Değeri, Şekil .26.

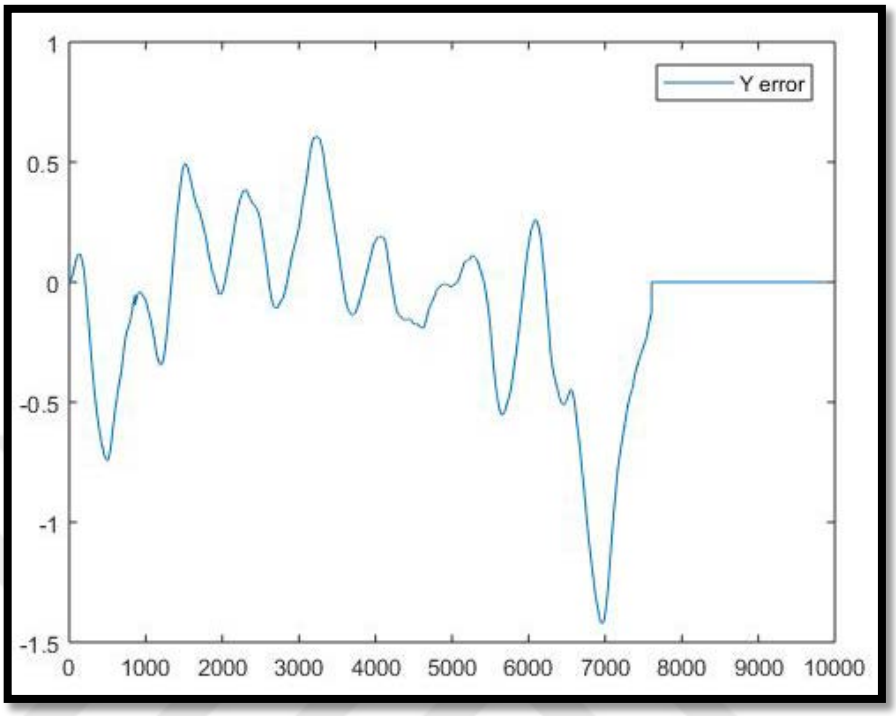
- Kalkış aralarında sıfıra yakın, gidiş başladığında yanı zaman olarak(800 s) en yüksek değeri sahip old ve hedefe yaklaşdıkça azalıyour.
- gidiş bittiğinde yanı zaman olarak((5400 s) sıfia yakın değer sahip oldu,
- iniş aralarında sıfıra yakın.



Şekil 26. EX (X eksanı ile hata miktarı)

1.4.2.2.2. Y Eksanı İle Hata Değeri, Şekil .27.

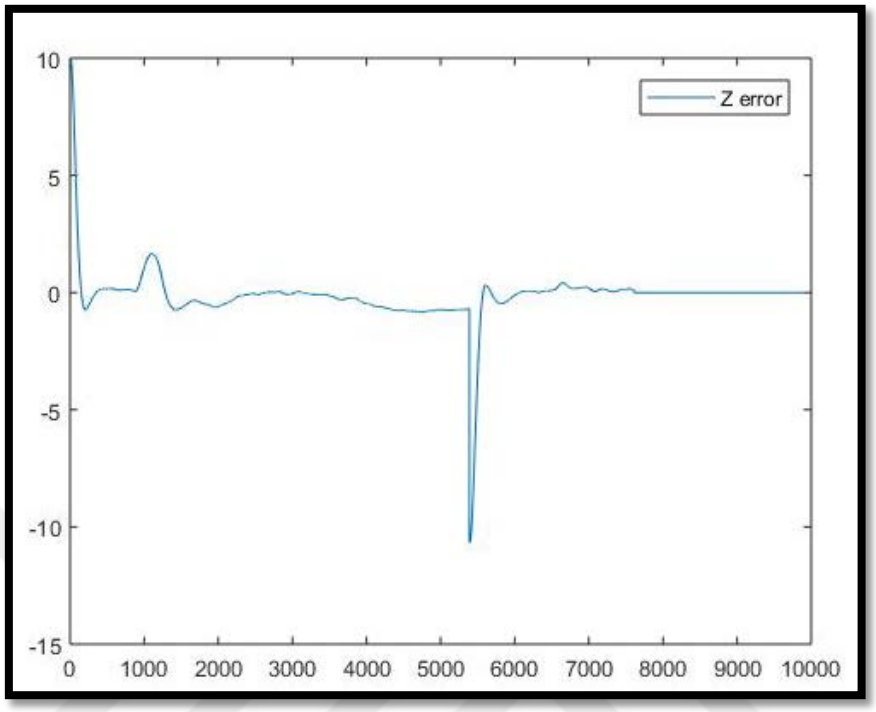
- Kalkış, gidiş ve iniş aralarında sıfıra yakın çünkü bu rotada Y sabittir.



Şekil 27. EY(Y eksanı ile hata miktarı)

1.4.2.2.3. Z Eksanı İle Hata Değeri, Şekil .28.

- Kalkış başladığında yanı zaman olarak(0 s) en yüksek değeri sahip old ve istenmiş irtifaya yaklaşdıkça azalıyour ve kalkış bittiğinde yanı zaman olarak((800 s) sıfıa yakın değer sahip oldu.
- gidiş aralarında sıfıra yakın.
- İniş başladığında yanı zaman olarak(540 s) en yüksek değeri ama negatif (iniş) sahip old ve yere yaklaşdıkça azalıyour ve iniş bittiğinde yanı zaman olarak((760 s) sıfia yakın değer sahip oldu.



Şekil 28. EZ(Z eksanı ile hata miktarı)

2. YAPILAN ÇALIŞMALAR

Stability analysis and autonomous control of conventional and tilted quadcopters / Klasik ve eğik rotorlu dört rotorlu insansız hava aracının otomatik kontrolü ve kararlılık analizi.

Modeling and attitude control of a quadcopter using model predictive controller / Dört rotorlu bir hava aracı modellemesi ve model öngörülü kontrolör ile yönelim kontrolü.

Quadcopter trajectory tracking control using reinforcement learning / Pekiştirmeli öğrenme ile quadcopter yörünge takibi kontrolü.

3. SONUÇLAR

- Manuel bir üçüş Arduino ile gerçekleştirmek.
- Otomatik üçüş simülasyon olarak gerçeklerştirmek şekil 24.

4. ÖNERİLER

- Bir FPGA ile otopilot kartı tasarlamak.
- Birden fazla komutlar gerçekleşmesi (konumu sapitlamak, eve dönme).
- Hafif bir quadcopter ile başlamak ki hasar az olmasına.

5. KAYNAKLAR

1. Duncan S., How to build a quadcopter drone A complete guide to building a radio controlled quadcopter, 2015.
2. Ian Cinnamon, DIY Drones for the Evil Genius - Design, Build, and Customize Your Own Drones Paperback, 2016.
3. David McGriffy, Make Drones, Teach an Arduino to Fly, 1st Edition.
4. Johnson W., Helicopter Theory, New York, Dover Publications, 1994.
5. http://www.droneybee.com/arduino-quadcopter-guide/, February 18th, 2019.
6. https://uavcoach.com/how-to-fly-a-quadcopter-guide/, 2019.
7. http://ardupilot.org/, 07/05/2019.
8. Renzo De N., CompLACS Helicopters Scenarios, University College, London 2013.

6. ÖZ GEÇMİŞ

Zekeriya HACIMUHAMMED 1990 yılında Süriye'da Halep şehirinde doğdum, Liseyi Halep Alavda Lisesinde tamamladım, lisans Halep Üniversitesinden mezun oldum, 2016 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında yüksek lisans öğrenimine başladım. Ana dilim arapça ve onun yanında iki dil biliyorum englizce ve arapça.