DENEYAP

Teknoloji Atölyeleri

HAVACILIK VE UZAY

LİSE

Dr. Öğr. Üyesi Rıfat BENVENİSTE
Doç. Dr. Barış GÖKÇE
Prof. Dr. Hüseyin AKILLI

TÜBİTAK

TÜBİTAK Deneyap Kitapları 26

HAVACILIK VE UZAY
LİSE

Dr. Öğr. Üyesi Rıfat BENVENİSTE
Doç. Dr. Barış GÖKÇE
Prof. Dr. Hüseyin AKILLI

TÜBİTAK Deneyap Kitapları DENEYAP TÜRKİYE Projesi kapsamında hazırlanmıştır.

ISBN 978-605-312-549-5
Yayıncı Sertifika No: 47703

Yayım Tarihi: 2024

TÜBİTAK Başkanı: Prof. Dr. Hasan MANDAL
Bilim ve Toplum Başkanı: Ömer KÖKÇAM
Genel Yayın Yönetmeni: Fatma BAŞAR
Editör: Doç. Dr. Şahin İDİL
Düzelti: Dr. Mustafa ORHAN
Telif İşleri Sorumlusu: Havva Hilal KAÇAR

TÜBİTAK Bilim ve Toplum Başkanlığı
Tunus Caddesi No: 80 Kavaklıdere 06680 Ankara
Tel: (312) 298 96 50
e-posta: deneyap@tubitak.gov.tr
https://yayinlar.tubitak.gov.tr/deneyap-atolyesi

İçindekiler

Sunuş

Havacılık ve uzay, insanoğlunun en eski hayallerinden biridir. Gökyüzüne yükselme, yıldızlara ulaşma ve evrenin gizemlerini keşfetme tutkusu, bizi sürekli yeni teknolojiler geliştirmeye teşvik etmektedir. Ülkemizin başta İnsansız Hava Aracı teknolojileri olmak üzere havacılık ve uzay alanında son on yılda ortaya koyduğu büyük gelişmeler, yerli ve milli imkanlarla geliştirmekte olduğumuz hava araçlarımız gençlerimizde bu tutkuyu daha da fazla ateşlemektedir. Bu kitap, sizleri bu heyecan verici dünyayla tanıştırmak ve geleceğin havacılık ve uzay mühendislerine ilham kaynağı olacak gerekli temel bilgileri ve becerileri kazandırmak için hazırlandı.

Deneyap Teknoloji Atölyelerinde eğitim veren eğitimcilerimize yol gösterici nitelikte olan bu bu kitap, Gözle, Uygula, Tasarla ve Üret felsefesi ile 10 bölümden oluşmaktadır. Kitabın içeriğinde Hava aracı ve uçuşun esasları, havacılıkta kullanılan malzemeler, hava aracı kontrol yüzeyleri, itki sistemleri, aviyonik sistemler, hava aracı simülasyonları, roket tasarımı ve uydu sistemleri gibi konulara değinilmektedir. Her bölümde, havacılık ve uzay teknolojilerinin farklı bir yönüne odaklanılmakta ve teorik bilgiler pratik uygulamalarla pekiştirilmektedir. Bu sayede, sadece ezberden öte, konuya ilişkin derin bir kavrayış ve deneyim kazandırılmaktadır.

Kitapta yer alan uygulamalar, tasarım ve el becerilerini geliştirmenin yanı sıra, malzeme bilimi, elektonik, mekanik, yazılım, robotik ve enerji sistemleri gibi farklı disiplinleri de biraraya getirmektedir. Bu sayede, havacılık ve uzay mühendisliğinin sadece bir mühendislik dalı olmadığını, birçok farklı disiplinin bir sentezi olduğunu da deneyimletmektedir.

İçeriklerde kullanılan bilgilerin niteliği ve uygulamalarda kullanılan yöntemler lise çağındaki çocukların havacılık ve uzay yapılarını kavrayabilecekleri ve uygulamaları başarı ile tamamlayabilecekleri bir temel zorluk seviyesinde hazırlanmıştır. Atölye eğitmenlerinin rehberliğinde kolaylıkla uygulamaları tamamlayacak ve eğlenceli bir öğrenme deneyimi yaşayacaklardır.

Tüm eğitmen ve öğrencilerin kendi havacılık ve uzay projelerine ilham kaynağı olması ve geleceğin havacılık ve uzay mühendislerine yol göstermesi ümidiyle keyifli okumalar dileriz.

HAVACILIK VE UZAY DERSİ ÖĞRETİM KILAVUZU

İçinde bulunduğumuz çağda öğrenme, öğrencilerin sınıf içinde verilen kavram, prosedür ve temel bilgileri istenildiği zaman belirli bir ölçüde sınav veya başka vasıtalar ile yeniden göstermesi olarak algılanmamaktadır. Amerikan Ulusal Araştırma Konseyi (2012), 21. Yüzyıl öğrencileri için bilişsel, kişisel, kişiler arası alan olmak üzere üç temel yeterlik başlığı belirlemiştir. Öğrencilerin öğrendiği kavram, prosedür ve temel bilgileri başka bağlamlara transfer etmesi ve bu bilgileri kendi amaçları doğrultusunda yeniden inşa etmesi gerekmektedir. Bu amaç doğrultusunda öğrenciler problem çözme gibi üst düzey düşünme becerilerini pratik etmelidir.

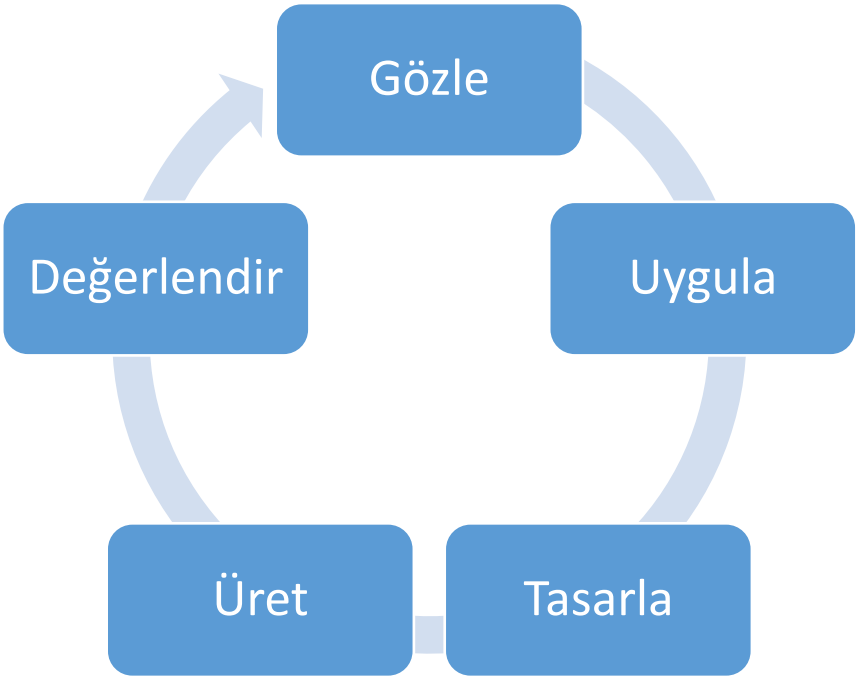
Problem çözme doğal, karmaşık ve anlamlı bir öğrenme/düşünme aktivitesi olarak tanımlanmaktadır. Problemlerin, otantik, öğrencilerin yaşantılarıyla ilişkili, derin öğrenmeyi destekleyen ögeleri içermiş ve içerdiği bilgi ve becerilerin zorluğunun hiyerarşik olarak yapılandırılmış olması gerekmektedir (Jonassen, 2007). Öğrencileri gerçekçi bütünsel görevlere dâhil etmenin, onların uygun şema ve zihinsel modeller oluşturmasına yardımcı olacağına inanılmaktadır (Merrill, 2007). Problemler doğaları gereği yapısallık, karmaşıklık, dinamiklik ve meydana geldikleri bağlam açısından farklılıklar göstermektedirler. Bu yüzden iyi oluşturulmuş içsel temsillerin, öğrencilerin yeni edindikleri bilgi ve becerileri daha sonra farklı durumlarda uygulamalarını kolaylaştırdığına inanılmaktadır. Öğrenciler meslek yaşamlarında hiç karşılaşmadıkları farklı ve ileri seviye problemler ile karşılaşacakları için yaratıcı düşünmek durumundadırlar. Öğrenciler aynı zamanda nasıl öğrenmesi ve kendisine nasıl yön çizmesi gerektiğini de öğrenmelidirler. Tüm bunlara ek olarak öğrenciler, iş birlikli çalışma ortamlarında nasıl çalışacaklarını ve iletişim kurmaları gerektiğini bilmelidirler.

Yaşadığımız çağda bilgi teknolojilerindeki değişimler ve bunun topluma ve günlük yaşama yansıması, mesleklerdeki dönüşümler ve fen bilimleri ve matematik alanlarındaki teknoloji kullanımı bireylerin bilişim teknolojileri okur-yazarı olmasını gerektirmektedir. Fakat bilişim teknolojileri okur-yazarlığı, sadece teknolojik araçları kullanmaktan ibaret değildir. Bu araçların üretimini de içermektedir. Wing (2006), herkesin bilgisayar biliminin temel kavramlarını bilip kullanması gerektiğini söylemiştir. Wing'e göre bilgi işlemsel düşünme, 21. yüzyılda yaşayan bütün bireyler için temel bir beceridir. Cuny, Snider ve Wing (2010), bilgi işlemsel düşünmeyi "Çözümlerin bir bilgi işleme birimi tarafından etkili şekilde yerine getirilebilecek formda sunulması amacıyla problemleri ve çözümleri formülleştirmeyi içeren düşünme süreci" olarak tanımlamışlardır. Bilgi işlemsel düşünme bir problem çözme sürecidir ve beş bileşeni içerir: (i) soyutlama, (ii) algoritmik düşünme, (iii) problemi bileşenlerine ayırma, (iv) çözümü verim açısından analiz etme ve (v) bir çözümü farklı bağlamlar için genelleme.

Bilgi işlemsel düşünme terimini ilk olarak kullanan ve programlamanın bütün öğrenciler tarafından öğrenilmesi gerektiğini bir teorik çerçeve ile ortaya koyan ilk kişi Seymour Papert'tır (Papert, 1991). Papert, Piaget ile çalışmış ve onun yapılandırmacılık (constructivism) teorisinden

etkilenmiştir. Papert (1993), yapılandırmacılığı temel alarak onun bir yorumu olan, inşacılık (constructionism) yaklaşımını geliştirmiştir. Yapılandırmacılık, pedagojik anlamda basit bir şekilde ele alınacak olursa, öğrencinin öğrenme sürecindeki rolünü, pasif olarak bilgiyi alandan aktif bir şekilde bilgiyi inşa eden olarak değiştirir. Yapılandırmacılık, felsefi anlamda basit bir şekilde ele alınacak olursa, dışsal objektif bilginin varlığını kabul etmez. Böyle bir bilgi olsa bile dışsal 'gerçeklik' öğrenci tarafından aynen kavranamaz. Öğrenci her hâlükârda kendi bilgisini inşa etmelidir. Yapılandırmacılık, eğitmeni bilgiyi bir zihinden diğerlerinin (öğrenciler) zihnine aktaran pedagojik aracı olarak görmez. Eğitmen öğrenciye buluş, keşif, inşa veya yeniden inşa sürecinde yardımcı olur ve onu pedagojik olarak destekler. Papert'ın inşacılığı, yapılandırmacılığın bu yorumlarını aynen alır ve bilginin inşa sürecinde öğrencilerin somut bir ürün vasıtası ile soyut kavramsal bilgiye ulaşması gerektiğini savunur. Öğrenciler zihinlerindeki temeli kullanarak somut bir ürün geliştirirler. Bu ürünü geliştirirken soyut kavramları kullanmaları gerekir. Böylece soyut kavramlar somut ürünler vasıtası ile öğrencinin öğrenme alanına dâhil olur. Öğrencinin kendisi, diğer öğrenciler ve eğitmen bu somut ürün hakkında konuşarak öğrencinin soyut bilgiyi içselleştirilmesine yardımcı olabilir. Böylece öğrenciler sosyal bir ortamda somut ürünler vasıtası ile bilgilerini inşa ederler.

Bu dersin tasarımında temel olarak 21. yüzyıl yeterlilikleri ve uygulama eğitiminin esasları kullanılacaktır. Havacılık ve uzay eğitimi ile ilgili alan yazında temel iki eğilim bulunmaktadır. Bunlardan ilki, dersi öğrencinin keşfedeceği şekilde tasarlayıp öğrenci merkezli bir yaklaşım sunmaktır. Bu yaklaşıma göre öğrenci süreçte aktif rol üstlenir. Eğitmen bilgiyi aktarmaktan ziyade, öğrencinin keşfetme veya oluşturma sürecinde ona yardım eden pozisyondadır. İkinci yaklaşım ise dersi eğitmenin anlatımı üzerine kurar. Burada eğitmen asıl sorumluluğu alır ve öğrencilere konuyu aktarmayı hedefler. Havacılık eğitimi alan yazına göre öğrencinin keşfetmesini temel alan yaklaşımda öğrenciler belirli bilgi işlemsel problemlerde yetkin olsalar da temel havacılık kavramlarda ve dolayısıyla bunların farklı bağlamlara transferinde sıkıntılar yaşamaktadır. Bunun karşısında bilgi aktarımını temel alan bilgi işlemsel öğretim yaklaşımları ise çağımızın öğrencilerini yetiştirmek için yetersiz kalmaktadır (Papert, 1991). Bu yüzden, alan yazında tartışıldığı üzere (Brown ve Campione, 1994; Grover, Pea ve Cooper, 2015; Mayer, 2004), bu derste temel kavramların oluşturulmasında aktarım yaklaşımı ön plana çıkarılırken öğrencilerin verilen temel ile ileri seviye zihinsel faaliyetlerde bulunması ve transfer edilebilir bir bilgi oluşturabilmesi için keşfe dayalı yaklaşımlar ön plana çıkarılacaktır. Bu amaç doğrultusunda detayları aşağıda verilecek olan gözle, uygula, tasarla, üret ve değerlendir öğrenme döngüsü kullanılmıştır.



Şekil 1. *Öğrenme Döngüsü*

GÖZLE - UYGULA - TASARLA - ÜRET - DEĞERLENDİR ÖĞRENME DÖNGÜSÜ

Gözle: Bu bölüm iki kısımdan oluşur. Birinci kısımda eğitmen öğrencilerin geçmiş bilgilerini aktive etmek ve onların dikkat ve motivasyonlarını sağlamak ile görevlidir. Eğitmen öğrencilerin geçmiş bilgilerini aktive etmek ve onların dikkatini çekmek için bir önceki derste yapılan etkinlikleri / çalışmaları kısaca özetleyebileceği gibi günlük yaşamdan ilgi çekici örnekler de kullanabilir. Örneğin otomasyona giriş yapmak için Hazerfen Ahmet Çelebi'nin hayatından ve ürettiği cihazlardan bahsedilebilir. Bu bölümün ikinci kısmında eğitmen bir uçuş dinamiğini uygulamalı olarak (göstererek) anlatır. Bu kısımda eğitmen daha aktiftir. Uygulamayı yaparken öğrencilere sorular sorabilir ve öğrencilerin sorularını cevaplayabilir.

Uygula: Bu bölümde eğitmen öğrencilerden bir önceki bölümde gösterilen uygulamaların aynısını veya bir benzerini ister / birlikte yapar. Örneğin, hazırlanmış bir İHA'nın pervanesinin hareket ettirilmesini gösteren eğitmen, öğrencilerden İHA pervanesini belirli bir hızda ve miktarda ters yönde hareket ettirmesini isteyebilir.

Tasarla: Bu bölümde öğrenciler daha aktif rol üstlenir. Eğitmen, rehber pozisyonundadır. Eğitmen, öğrencilere takıldıkları noktalarda destek olacaktır. Öğrencilerin etkinlikten kopup, motivasyonlarının düşmesine izin vermemeye çalışacaktır. Fakat eğitmenin sağladığı destek gereğinden fazla da olmamalıdır. Bu bölümde eğitmen tarafından öğrencilere bir problem verilir. Öğrencilerden öncelikle bu problemin çözümü (tasarlamaları) istenir. Tasarlama aşamasında

öğrenciler temel itibariyle bilinenler ile istenenler arasındaki bağı kurarak bir plan üreteceklerdir. Bu amaçla öğrenciler (Bilgi işlemsel düşünme becerisi bileşenlerini kullanırlar):

- Bilinenleri ve istenilenleri ayrı ayrı belirler,
- İstenilenleri alt bileşenlere ayrılabiliyorsa ayırır,
- Bu problem veya alt problemlerin aynılarına veya benzerlerine daha önceden çözüm ürettiyse bunları tanımlar,
- Bu problemler veya alt problemlerin çözümü için bilgisayar biliminde daha önceden belirlenmiş çözümlerin (sıralama ve arama algoritmaları gibi) olup olmadığını belirler,
- Daha önceki adımlarda ortaya koyduklarını kullanarak bir çözüm planı üretir.

Bu aşamadaki önemli nokta, öğrencilerin çözüme doğrudan başlaması yerine önce çözüm hakkında düşünmesi ve bir çözüm planı üretmesidir. Öğrenciler her defasında yukarıda bahsi geçen beş adımı yapmak istemeyebilir veya bu adımları yaparken sıkılabilir. Bu durumlarda, öğrencilerin adımları bire bir uygulaması yönünde zorlamak yerine onlardan problemi doğrudan çözmeye başlamadan önce problem hakkında düşünmesi ve planlama yapması istenebilir.

Üret: Bu bölümde öğrenciler aktif rol üstlenir. Eğitmen rehber pozisyonundadır. Eğitmen öğrencilere takıldıkları noktalarda destek olacaktır. Destek, Vygotsky'nin (1978) Yakınsal Gelişim Alanı (Zone of Proximal Development) kavramında belirttiği gibi bireyin yardım ile gerçekleştirebileceği, ancak henüz bağımsız olarak yapamayacağı bir durum oluştuğunda sağlanmalıdır. Üret aşamasında, öğrencilerden bir önceki adımda tasarladığı planı kullanarak probleme algoritmik bir çözüm üretmesi istenir. Öğrenciler İHA veya model uçak yapım aşamasında çalışarak gerekli donanımsal ve yazılımsal çözümleri geliştirirler.

Değerlendir: Buradaki değerlendirme ile anlatılmak istenen doğrudan öğrencinin başarısının notlandırılması değildir. Temel hedef, öğrencinin öğrenme sürecinde yaşadıkları ve öğrendikleri üzerine düşünmesini sağlamaktır. Bu sayede öğrenci, problem çözme, dersin konusu ve kendisi ile ilgili gözlemler yaparak yeni öğrenmeler, kendisini değerlendirme ve planlama açısından fırsatlar elde edecektir. Öğrencilerden şu soruları cevaplamaları istenebilir:

- Verilen problemi tanımlayınız (problemi kendi cümleleri ile ifade etme).
- Problemin çözümü için hangi stratejileri kullandınız ve neden bu stratejileri seçtiniz?
- Problemi çözerken ne gibi sıkıntılar yaşadınız ve bunların üstesinden gelmek için neler yaptınız?
- Kullandığınız yöntemler, bu sıkıntıları gidermekte başarılı oldu mu?
- Grup arkadaşınızla ihtilafa düştüğünüz durumlar oldu mu ve bunların üstesinden gelmek için neler yaptınız?
- Grup arkadaşınızdan ne öğrendiniz?

Öğrencilerden buradaki soruların tamamına cevap vermesi beklenmemektedir. Bu sorulardan, verilen etkinlikten elde ettikleri deneyimlere bağlı olarak, kendilerine uyanları cevaplayabilirler. Cevaplar, öğrencilerden yazılı olarak da istenebilir. Fakat öğrenciler, belirli bir süre sonra sürekli aynı sorulara cevap vermekten sıkılabilir/sıkılacaktır. Bu durumda, belirli derslerin sonunda öğrencilerden genel olarak dersteki deneyimlerini değerlendirmeleri istenebilir. Bunun yanında, değerlendirme sürecini daha etkili hâle getirmek amacıyla çeşitli etkinlikler yapılabilir. Örneğin, öğrenciler bir halka şeklinde dizilir ve eğitmen elinde bulunan topu (veya benzeri bir cismi) öğrencilerden birine atarak yukarıdaki sorulardan bir veya birkaçına yanıt vermesini isteyebilir

(veya dersteki deneyimlerini genel olarak değerlendirebilir). Cevap veren öğrenci, elindeki topu bir başka arkadaşına atar ve arkadaşı da bir veya birkaç soruyu cevaplar. Bu durum değerlendirme doyum noktasına ulaşana kadar, öğrencileri sıkmadan, devam ettirilebilir.

Eşli Programlama

Öğrenciler, kendilerine verilen görevlerden uygun olanlarını (yukarıda bahsedilen uygula ve üret adımları) eşli programlama grupları içerisinde yapacaklardır. Eşli programlamada iki öğrenci, bir model uçak veya bilgisayar karşısında yan yana oturarak tasarım, uygulama veya hata ayıklama için iş birlikli çalışır. Eşli programlama eşli araba yarışlarına benzetilebilir. Eşli araba yarışlarında sürücü arabayı kullanırken kılavuz sürücüye yön tayini konusunda yardımcı olur. Eşli programlamada bilgisayarı veya hava aracını kullanan kişiye sürücü denir. Sürücünün görevi hava aracına istenenleri gerçekleştirmesi için tasarımı ve kodlamayı yapmaktır. Sürücünün yanındaki kişiye kılavuz denilir. Kılavuz, bilgisayarı veya hava aracını kullanmaz. Kılavuzun görevi çıkan problemleri veya ana problem için çözüm üretmek, bu süreçte ortaya çıkan hataları belirlemek ve sürücünün nasıl çalıştığını değerlendirmektir. Eşli programlamada, araba yarışından farklı olarak, sürücü ve kılavuz düzenli olarak yer değiştirir. Öğrencilerin her iki görevden de öğreneceği şeyler vardır. Bu yüzden görev değiştirme çok önemlidir. Eğitmen öğrencilerin periyodik olarak görev değiştirmesini sağlamalıdır.

- İki öğrenci bir bilgisayar veya hava aracı karşısına oturur,
- Bir öğrenci kodları yazarken diğeri kodları değerlendirir ve öneride bulunur,
- Belirli zaman aralıklarıyla öğrenciler rollerini değiştirir.

Eşli programlamada bir diğer önemli nokta, hangi iki öğrencinin eş olarak atanacağıdır. Burada, iyi bilen ve az bilen gibi, farklı bilgi veya beceri gruplarında öğrencilerin bir araya getirilmesi iyi bilenin az bilene öğretmesi açısından faydalı olarak görülebilir. Fakat pratikte bu fayda gerçekleşmemektedir. Bundan ziyade, iyi bilen bir müddet sonra diğerini kendine ayak bağı olarak görmeve görece daha az bilen de iyi bilen ile iletişim kurmakta sıkıntı yaşayıp etkinliklerden kopma eğilimi göstermektedir. Benzer bilgi ve beceri düzeyinde olan gruplar daha verimli çalışmaktadır. Bu yüzden benzer bilgi ve beceri düzeyinde olan öğrencilerin eş olarak belirlenmesi önemlidir. Bazı durumlarda eş olarak tayin edilen öğrenciler birbirleriyle ciddi anlaşmazlıklar ve çatışmalar yaşayabilirler. Çatışma yaşayan öğrencilerin aynı grupta kalması sağlıklı olmayacaktır. Bu durumu fark eden eğitmen, çatışma yaşayan eşleri farklı öğrencilerle yeniden eşlemelidir.

Gruplar Arası İletişim ve Yarışmalar

Havacılık etkinlikleri boyunca gruplar arası bilgi alışverişine izin verilmelidir. Gruplar arasında paylaşımcı bir ortam oluşturulmalıdır. Fakat bu paylaşım, komple bir çözümün paylaşımı şeklinde olmamalıdır. Öğrenciler, çözüm yolları, stratejiler ve eksik bilgiler gibi konular için paylaşım yapabilirler. Fakat verilen bir problemin bütün çözümü paylaşılmamalıdır. Bu konu hakkında eğitmen, öğrencileri daha önceden bilgilendirmelidir ve onların ne tür bir paylaşım içerisinde olduğunu takip ederek gereğinden fazla olan paylaşımları engellemelidir.

1. HAFTA: HAVA ARAÇLARI VE UÇUŞUN ESASLARI

Ön Bilgi:

- Temel bilgisayar ve donanım bilgisi

Haftanın Kazanımları:

- Öğrenciler temel düzeyde hava aracını kavrar.
- Hava aracı türlerini sınıflar.
- Hava araçaların uçuş esaslarını yorumlar.
- Bernoulli ilkesini kavrar.
- Kanat yapıları ve kanatların özelliklerini ilişkilendirir.

Haftanın Amacı:

Bu hafta öğretilecek başlıkların amacı, hava araçlarının temel tanımını yaparak bir hava aracının uçmasını sağlayan temel unsurların kavratılmasıdır. Hava aracının uçuş esasları kanat ve katlardaki taşıma unsurları öğretilerek temel bir uçağın nasıl uçtuğunu öğrencilerin kavraması sağlanacaktır. Temel bir lastik motorlu model uçak ile bir uçağın havada kalmasını sağlayan unsurları uygulamalı olarak yorumlatılacaktır.

Kullanılacak Malzemeler:

İki adet balon, ip, a6 kâğıt, lastik motorlu uçak kiti.

Haftanın İşlenişi:

Gözle: Hava aracının tanımı, uçuşun esasları, hava araçlarında kanat yapıları ve taşıma kuvvetinin sağlanması.

Uygula: Bernoulli ilkesinin deneysel olarak gerçekleştirilmesi.

Üret: Lastik motorlu model uçak yapımı.

Değerlendir: Üretilen lastik motorlu uçağın uçuş esaslarına istinaden yapısının incelenmesi ve hareketlerinin değerlendirilmesi.

1. Gözle

1.1. Hava Araçları

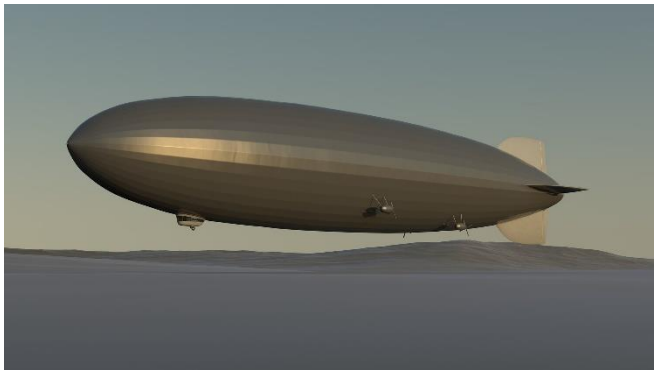
Hava aracını sözlük tanımı ile ifade etmek istediğinizde farklı kaynaklarda farklı açıklamalar karşınıza çıkacaktır. Genel tanımı ile havadan daha ağır veya daha hafif olsun, havada hareket etmek üzere tasarlanmış aerodinamik kuvvetler vasıtasıyla uçabilen herhangi bir yapı veya makineye hava aracı denilmektedir.

Hava araçları tasarımları gereği havada uçabilmek için temelde taşıma kuvvetinden faydalanmaktadır. İlerleyen bölümlerde bu kuvvetler daha detaylı anlatılacaktır. Hava araçları taşıma kuvvetini sağlayan bileşen türlerine göre farklı türlerde sınıflandırılabilmektedir.

1.1.1 Havadan Hafif Hava Araçları (Aerostat)

Havadan hafif hava araçlarına örnek olarak sıcak hava balonu ve zeplin verilebilir. Havanın ısıtılarak genleşmesi veya havanın yoğunluğundan daha düşük yoğunluğa sahip bir gazla hava aracının uçmasının sağlandığı yapıdır. Sıcak hava balonunda sadece dikey yönde kontrol sağlanabilirken, zeplinde ise itici motorlar ve dümenler sayesinde her yönde kontrol sağlanabilmektedir.

Şekil 1. *Ülkemizde Yerli Üretilen Roket Şeklinde Sıcak Hava Balonu*



Şekil 2. *Zeplin Örneği*

1.1.2 Havadan Ağır Hava Araçları (Aerodyne)

Yer çekimi kuvvetini yenecek şekilde yapısal olarak taşıyıcı bir unsura sahip olan hava araçlarıdır. Bu hava aracı taşıma kuvvetini sağlamak için sıcak gaz veya motor ve pervane gibi bir itki bileşenine ihtiyaç duyarlar. Havadan ağır hava araçları taşıyıcı yapılarına göre Sabit Kanat ve Döner Kanat olarak iki sınıfa ayrılırlar.

1.1.2.1 Sabit Kanatlı Hava Araçları

Gündelik hayatta uçak olarak bahsedilen hava araçları kanatlarına karşı havanın dinamik reaksiyonu ile taşıma kuvveti kazanan araçlardır. Kanatlar bütün olarak hareketsizdir ve sabit şekilde gövdeleri üzerinde bulunur. Uçuş hızına bağlı olarak taşıyıcılığı arttırmak için kanat genişliğini değiştirecek hareketli parçalar, kanadın üzerinde bulunabilir. Tasarımına bağlı olarak kanat sayısı, konumu kanat şekli farklılıklar gösterebilir. Örnek olarak bir kargo uçağı yüksek ağırlıkları taşıyacağı için geniş kanatlara ihtiyaç duyarken, bir savaş uçağı hızlı manevra kabiliyeti için dar kanatlara ihtiyaç duyar.

Şekil 3. *Sabit Kanatlı Uçak Örnekleri*

1.1.2.2 Döner Kanatlı Hava Araçları

Döner kanatlı hava araçları, bir milin etrafında dönen rotor kanatları adı verilen kanatlar tarafından üretilen kaldırmayı kullanan havadan ağır bir uçan makinedir. Tek bir mile monte edilmiş birkaç rotor kanadına rotor denir. Uluslararası Sivil Havacılık Örgütü, bir döner kanatlı hava aracını "bir veya daha fazla rotor üzerindeki havanın tepkileriyle uçuşta desteklenen"

olarak tanımlar. Rotorcraft genellikle, helikopterler, siklokopterler, otojirolar ve gyrodynes gibi tüm uçuş boyunca kaldırma sağlamak için bir veya daha fazla rotorun gerekli olduğu hava araçlarını içerir. Bileşik rotorlu taşıt ayrıca ek itme motorları veya pervaneler ve statik kaldırma yüzeyleri içerebilir.

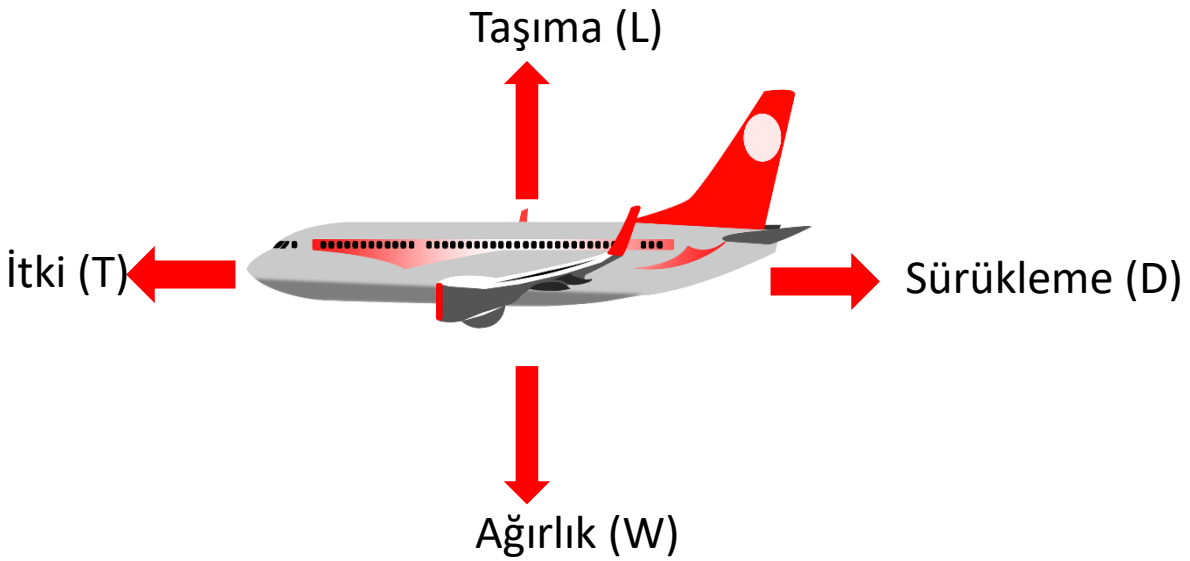
Şekil 4. *Döner Kanat Hava AracıÖrnekleri*

1.2. Uçuşun Esasları

Bu kısımda bir uçağın sabit irtifada düzgün bir simetrik uçuşla nasıl havada kaldığını açıklanacaktır. Uçak üzerindeki kuvvetler ve uçağın bu kuvvetlere bağlı nasıl hareket ettiği öğrencilere öğretilecektir.

1.2.1. Uçağa Etki Eden Temel Kuvvetler

Bir uçağın en temel uçuş hali, sabit irtifada (uçuş yüksekliği) simetrik uçuş halidir. Uçağın bu konumdaki uçuşunu sürdürülebilmesi için kendi ağırlığına (weight) ters yönde bir taşıma (m-lift) kuvvetine ihtiyacı vardır. Taşıma kuvveti uçağın ağırlığından fazla olursa uçak yükselir, az olursa alçalır. Eşit olduğunda ise uçak mevcut irtifasında uçuşunu devam ettirir.



Şekil 5. *Uçağa Etki Eden Temel Kuvvetler*

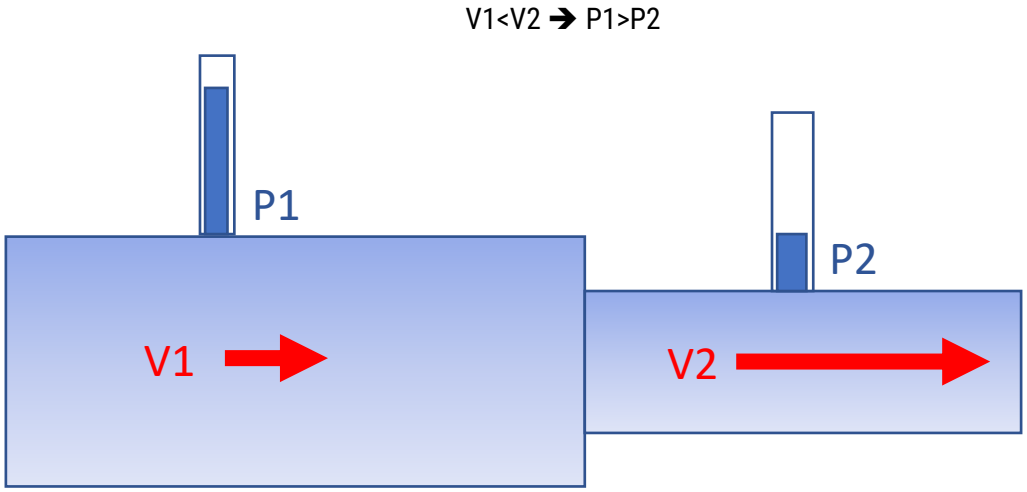
Aerodinamik olarak taşıma ancak uçağın belirli bir yükseklikteki hızla uçması halinde sağlanabilir. Havanın içerisinde yüksek hızla hareket eden cisimlere bir direnç kuvveti etki

eder. Uçakta da bu direnç kuvveti sürükleme (drag) olarak adlandırılır. Uçuşun aynı şekilde devam ettirilebilmesi için sürükleme kuvvetinin de bir itki kuvvetiyle dengelenmesi gereklidir. İtki kuvveti sürüklemeden fazla olduğunda uçak ileri yönlü hızlanır. İtki sürüklemeden az olduğunda ise uçak yavaşlar. Sürükleme ve itki eşit olduğunda uçak mevcut hızıyla ilerlemeye devam eder.

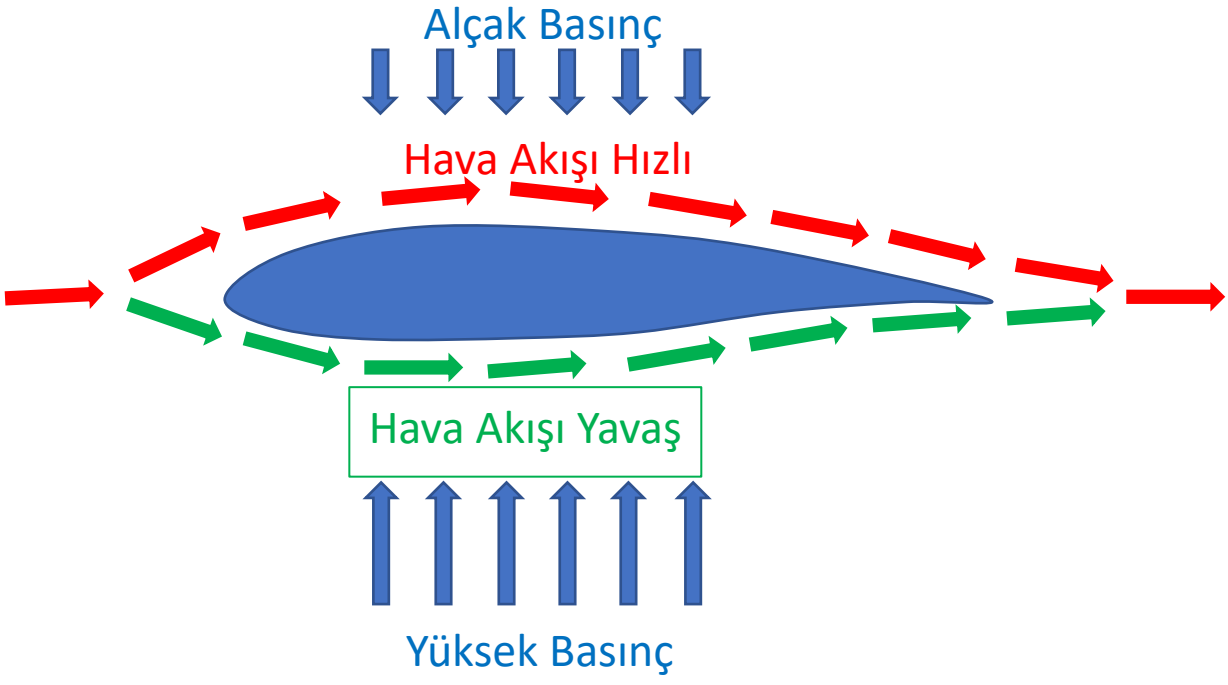
Bir uçağın bütün bileşenlerinin (kanat, kuyruk, gövde gibi) taşımaya olumlu veya olumsuz katkıları vardır. Ancak uçağın taşıma kuvvetini temelde kanatlar sağlar. Sürükleme ise kanatlar, gövde, kuyruk, iniş takımları gibi uçakta bulunan bütün bileşenlerden kaynaklanır.

1.2.2. Bernouilli İlkesi

Uçak kanadındaki taşıma Daniel Bernoulli tarafından bulunan Bernoulli ilkesine dayanmaktadır. Bernoulli ilkesine göre akışkanın hızı ile basıncı arasında ters orantı vardır. Akışkan hızının arttığı bir kesitte hareket ederse akışkanın basıncı azalır. Hızının azaldığı bir kesitte hareket ederse akışkanın basıncı artar.



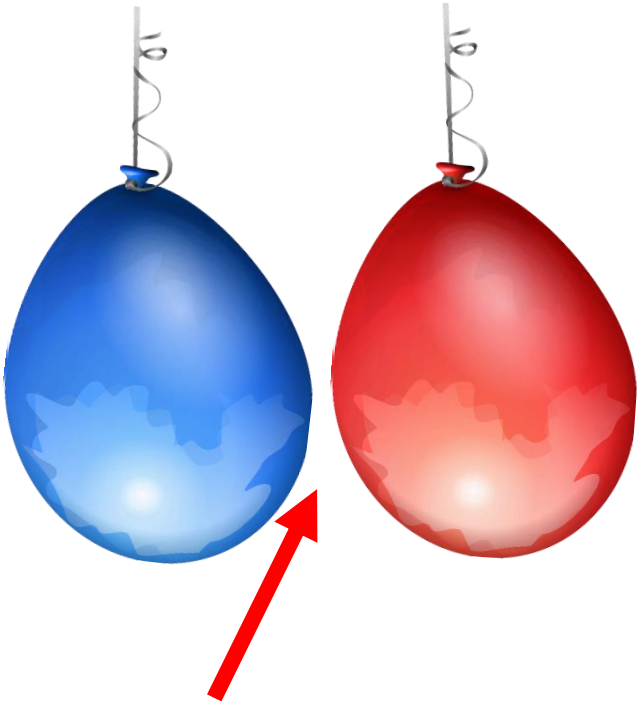
Uçak kanatları da Bernoulli ilkesini sağlayacak şekilde su damlası şeklinde bir profilde tasarlanır. Böylelikle kanadın alt tarafındaki hava üst tarafındakine göre daha yavaş hareket etmektedir. Havanın yavaş hareket ettiği kanadın alt tarafında yüksek basınç, havanın hızlı hareket ettiği kanadın üst tarafında ise alçak basınç meydana gelir. Böylelikle yüksek basınç kanadın uçağın ağırlığını yenecek taşıma kuvvetini sağlar.



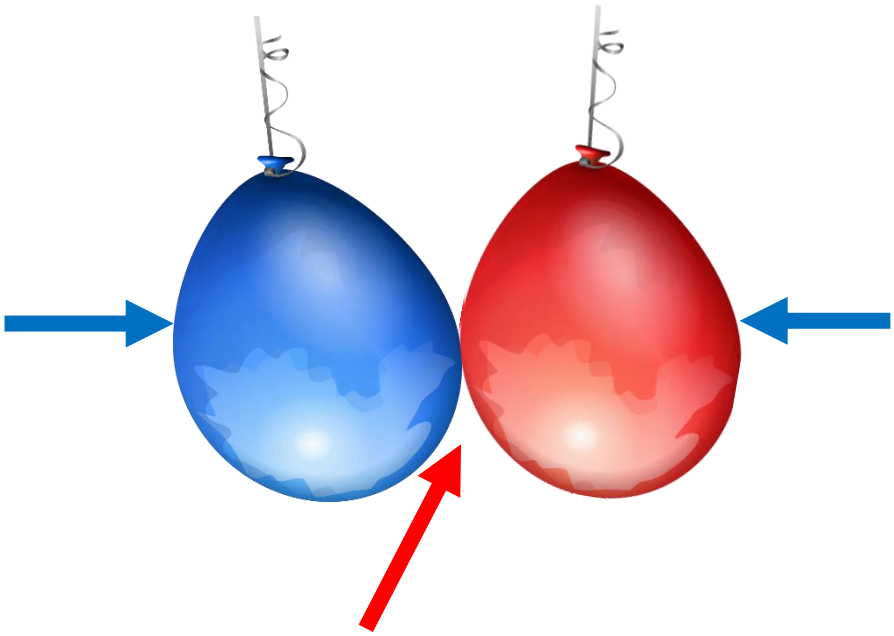
2. Uygula: Bernoulli İlkesi

Öğrencilerden A6 boyunda bir kâğıdı dudak hizasına getirerek üstünden üflemeleri istenir. Bernoulli ilkesi doğrultusunda kâğıdın üst tarafında hava hızlı hareket edeceğinden kâğıdın altındaki basınç üstündeki basınçtan yüksek olacaktır. Böyleleikle kâğıt yukarı doğru hareket etmeye başlaycaktır.

Bernoulli ilkesini gözlemlemek için ikinci uygulamada iki adet balon üstten bir yere bağlanır. Ardından bir pipet yardımı ile iki balonun arasından üflenir. Balonlar arasında hava akışı hızlanacağından aralarındaki basınç düşecek ve balonlar birbirine doğru yaklaşacaktır.



İki balonun arasından bir pipet yardımı ile üfleyin.



Balonlar birbirine doğru yaklaşacaktır.

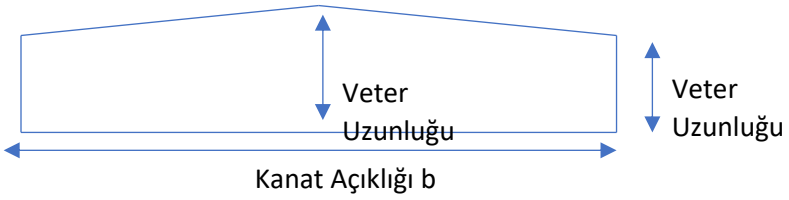
3. Gözle: Uçak Kanatlarının Geometrisi

Uçak kanatlarının geometrisi uçuş şekline (düşük hız, yüksek hız, sesüstü hızlar gibi) bağlı olarak üst görünümlerinin ve kesit geometrilerinin yapısı değişir.

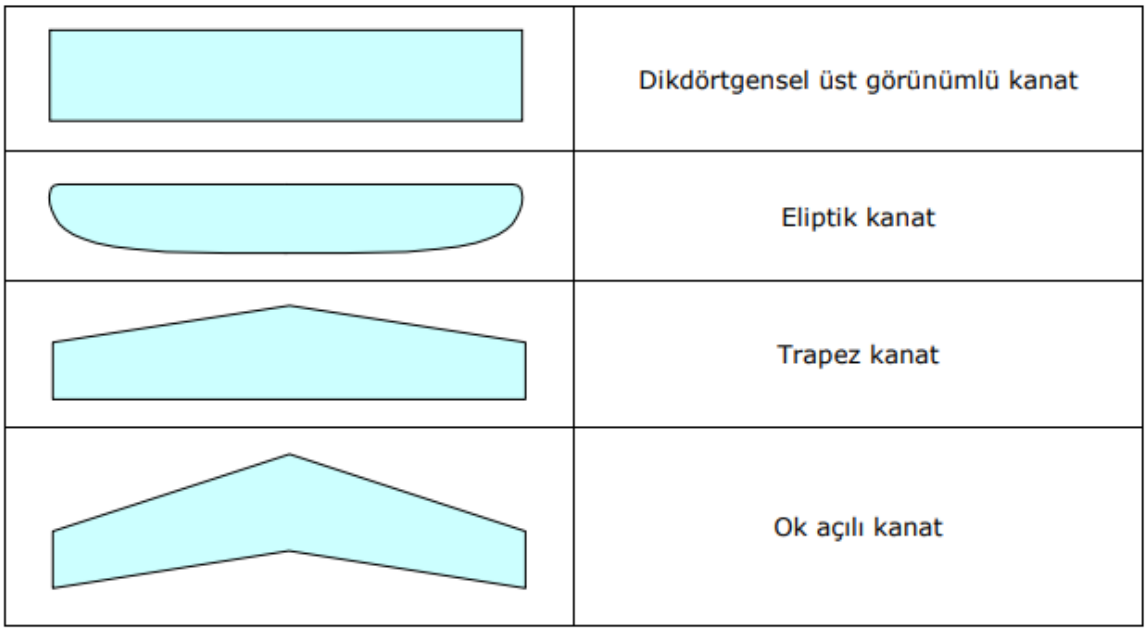
Bir uçak kanadına üstten bakıldığında bir uçtan diğer uca uzunluğuna kanat açıklığı (span), genişliğine ise veter uzunluğu (cord length) adı verilir.

$$Açıklık\ Oranı = \frac{b}{c}$$

Dikdörtgensel üst-görünümlü bir uçak kanadının açıklığının veter uzunluğuna oranı açıklık oranı (aspect ratio) olarak adlandırılır. Trapez şekilli bir kanatta ise veter uzunluğu kanat boyunca değişiklik gösterir. Bu durumda açıklık oranı, kanadın kanat açıklığının üst görünüm yüzey alanının "S" karesine olan oranı olarak hesaplanır.

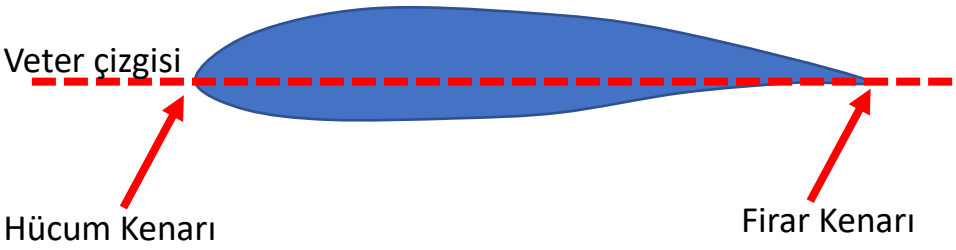


$$Açıklık\ Oranı = \frac{b^2}{S}$$

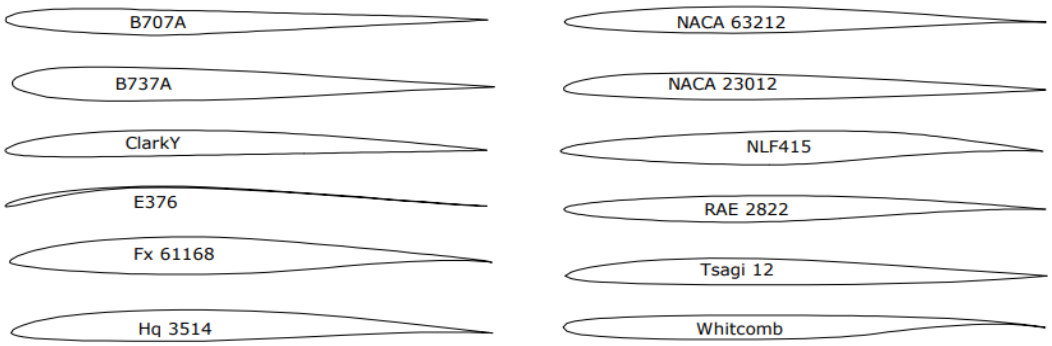


Şekil 6. *Üst Görünümlerine Göre Kanat Tipleri*

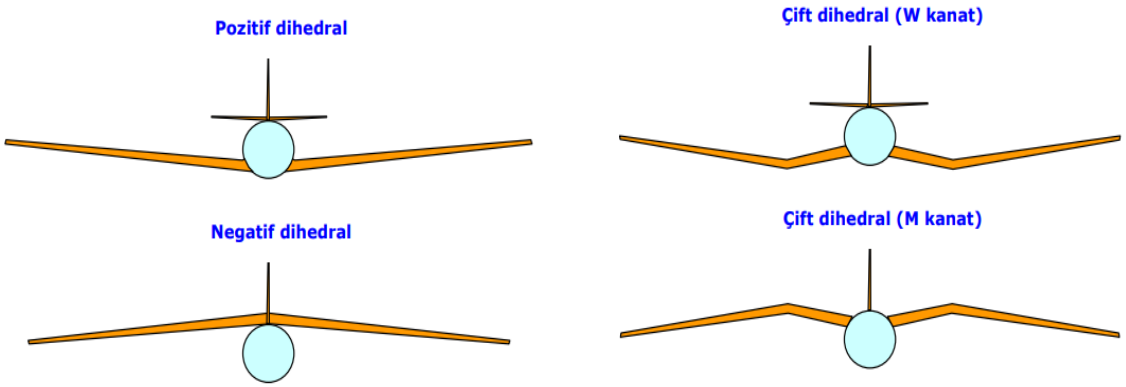
Kanatların gövde doğrultusunda kesilmeleri halinde elde edilen yanal kesit görünümüne kanat profili (airfoil, aerofoil) adı verilir.



Kanat profillerinin uçuş doğrultusu ve yönüne göre hava akımıyla ilk buluştukları en öndeki noktasına hücum kenarı (leading edge), en geride kalan ve hava akışının kanat yüzeyini terk ettiği noktasına ise firar kenarı (trailing edge) adı verilir. Hücum ve firar kenarlarından geçen doğruya veter çizgisi (chord line), hücum ve firar kenarları arasındaki uzaklığa ise veter uzunluğu (chord length) adı verilir. Bütün kanat profillerinin firar kenarları sivridir. Bu durum aerodinamik taşımanın oluşturulması ve kalitesi açısından önemlidir.



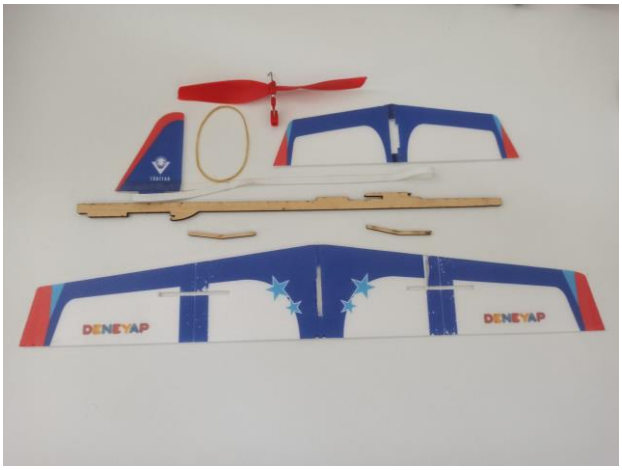
Şekil 7. *Kanat Kesiti Örnekleri*



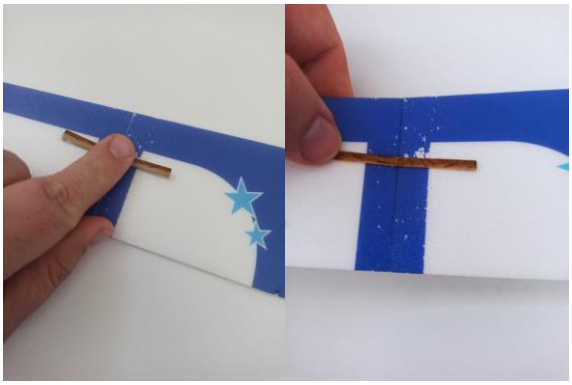
Şekil 8. *Dihedral Açılarına Göre Kanat Tipleri*

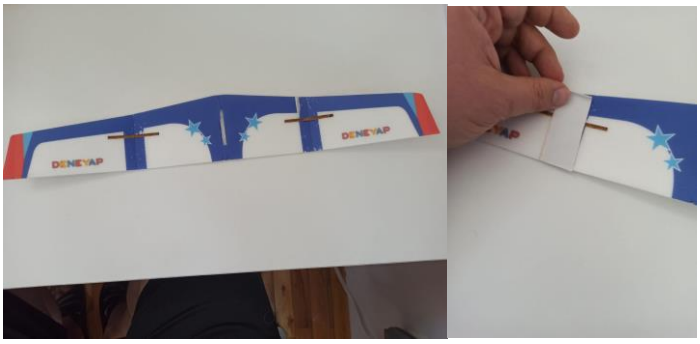
4. Üret: Lastik Motorlu Uçak Yapımı

Öğrenciler, bir uçağın yapısını ve kanadın taşıyıcılığını somut gözlemleyebilmeleri için lastik motorlu bir uçak yaparak uçuracaklardır. Lastik motorlu uçak kitinde strafor kanat parçaları, balsa lazer kesim gövde, plastik pervane ve enerjiyi depolayacak lastik bulunmaktadır.



İlk adımda kanat destek parçaları, kanadın yarıklarına yerleştirilmelidir.

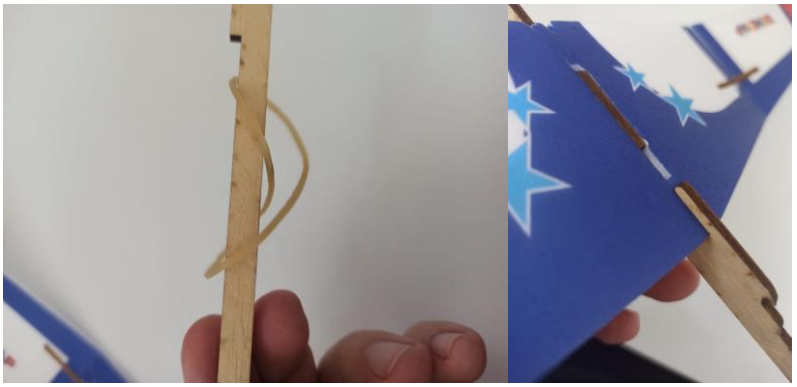


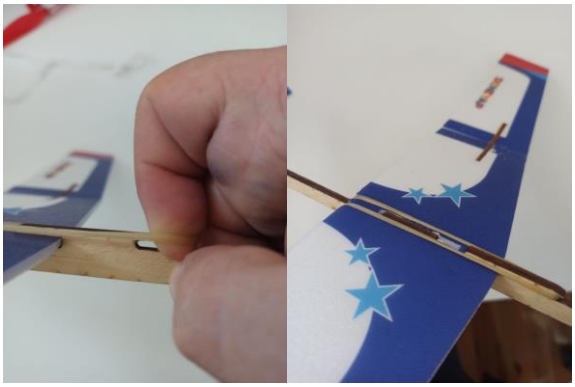


Kanat destekleri montajı sonrasında kanat şekildeki açıyı alacaktır. Desteklerin düşmemesi için kit içerisinde gelen yapışkan bantlar yapıştırılarak destekler sabitlenebilir.

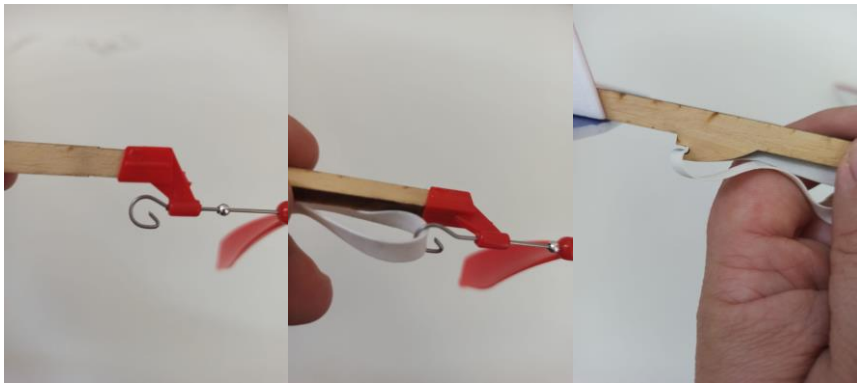
Kuyruk kısmında yatay ve dikey dengeleyici birbirlerine geçirilmelidir. Yapışkan bant sıyrılarak gövdeye sabitlenebilir.

Gövde kısmına paket lastiği çaprazlı şekilde yerleştirilmelidir. Ardından kanat gövdeye geçirilerek lastik üzerinden atlatılarak sabitlenir.

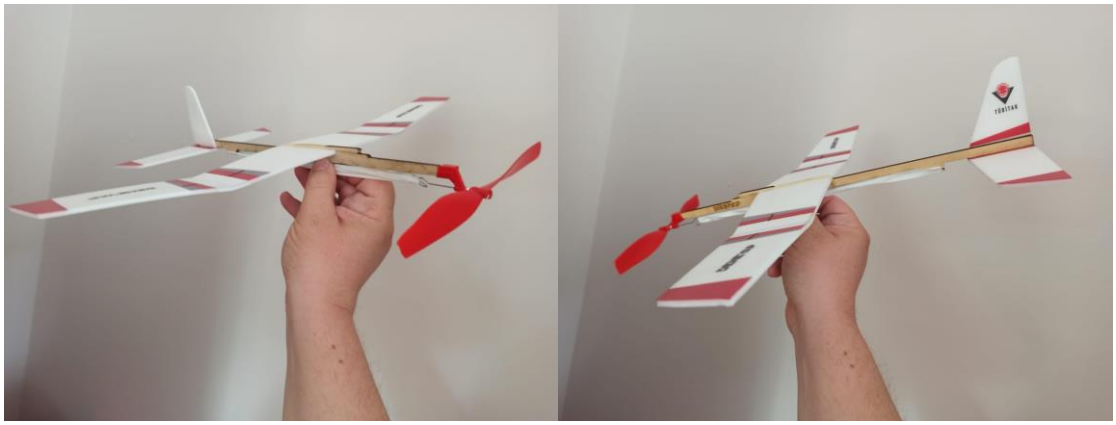




Plastik pervane gödenin önüne geçirilerek sabitlenir. Kurma lastiği pervane kancasına takılır ve gövdenin arka kısmında bulunan çentiğe geçirilir.



Tamamlanan uçak şekildeki gibi olmalıdır.



5. Uygula: Lastik Motorlu Model Uçağın Uçurulması

Bu uygulamanın adımları aşağıdaki gibi olmalıdır:

- Eğitmen öğrencilerden, model uçağın uçuşu için uygun bir uçuş alanına geçmelerini ister. Bu alan kapalı bir spor sahası, salon veya açık bahçe gibi bir alan olabilir.
- Öğrencilerden 200-2500 tur model uçak pervanesi çekiş yönünün tersinde kurmaları istenir.
- Öğrencilerden uçaklarını 5-10 derece yukarı tutarak elden bırakarak uçurmaları istenir.
- Eğer açık alanda uçuruluyorsa rüzgâra karşı ve rüzgârı arkaya alarak uçağı fırlatmaları istenebilir. Rüzgârın uçağa etkisi gözlemlenerek yorumlanabilir.

6. Değerlendir

Günün sonunda öğrencilerle aşağıdaki sorular ve benzerleri üzerinden tartışma ortamı yaratılır.

- Hava aracı nedir?
- Hangi türde hava araçları vardır ve bunların farkları nedir?
- Bir uçak nasıl uçar?
- Uçak üzerinde hangi kuvvetler bulunur?
- Uçağın uçması için kanat taşımayı nasıl sağlar?
- Lastik motorlu uçakta itki kuvveti nasıl oluşur?
- Lastik motorlu uçakta kanatta taşıma nasıl olur?
- Lastik motor olmasaydı uçağımız uçar mıydı?
- Lastik motorlu uçağın kanat dihedral şekli nasıldır?

2. HAFTA: HAVACILIKDA KULLANILAN MALZEMELER

Ön Bilgi:

- Öğrenciler temel düzeyde malzeme kavramını bilir.
- Öğrenciler uzunluk ölçme, alan ve hacim hesaplama, eşitlik ve denklem kurma ile çap (yarıçap) kavramını bilir.
- Öğrenciler doğal malzemeleri bilir.

Haftanın Kazanımları:

- Öğrenciler malzemeyi tanımlar ve malzemeleri sınıflandırır.
- Günümüzde kullanılan havacılık malzemelerini tahmin etmeleri istenir. Bu malzemeleri sınıflandırmaları beklenir.
  1- Kompozit malzemelerin yapı ve özelliklerini keşfeder.
  2- Matris ve takviye malzemelerini bilir.
- Sandviç yapılı kompozit ve bu yapının elemanlarını modeller.
- Öğrenciler el becerileri ile kendi karbon kanat tasarımlarını oluşturarak gelişmiş karbon tasarımları ortaya koyarlar. Aynı zamanda kendi çekirdek ve kalıp malzeme tasarımlarını oluşturulmaları beklenir.
- Deneyler sonucunda malzemelere hafif ama sağlam özelliklerin nasıl kazandırıldığı keşfeder.

Haftanın Amacı:

Bu haftada, öğrencilerin hava ve uzay araçlarının neden hafif ve sağlam malzemelerden üretilmesinin gerektiğini anlamaları sağlanacaktır. Mevcut malzeme türlerine ilave yeni ve endüstriyel malzemeler eklenecektir.. Kompozit ürün içerikleri de öğrencilere öğretilecektir.. Ayrıca, kendi el becerilerini kullanarak bir uçak veya İHA kanat kesiti üretebileceklerdir.

Kullanılacak Malzemeler:

60 gr karbon kumaş 50 mm x 455 şerit şeklinde kesilmiş (öğrenci sayısı kadar),

Kanat kalıbı + kanat profili (core) (öğrenci sayısı kadar),

10 gr epoksi + 4 gr hızlandırıcı set (öğrenci sayısı kadar 2 ayrı tüp içerisinde),

Mayler şerit 65 mm x 455 mm şerit şeklinde kesilmiş (öğrenci sayısı kadar),

Eldiven (öğrenci sayısı kadar),

Fırça (öğrenci sayısı kadar),

Karıştırma çubuğu (öğrenci sayısı kadar),

Pamuk (öğrenci sayısı kadar),

Kalıp ayırıcı vaks 4 adet mini krem kutusu,

Mini tartı 1 adet (plaka ağırlık tespitinde kullanılacak),

Maske (öğrenci sayısı kadar),

100 adet geniş ambalaj lastiği (kalıp sıkıştırma işlemi için her kalıba 5 adet),

100x100 mm ölçüsünde; karbon, kevlar, balsa, cam elyaf (g10) alüminyum plaka (1 sınıf için 2 set).

Haftanın İşlenişi:

Gözle: Malzeme tanımı yapılarak, havacılık ve uzay alanında kullanılan malzemelerden beklenen performanslar tartışılır, takviye malzemeleri ve çekirdek malzemeleri ile ilgili videolar izletilerek matris ve takviye fazı öğrencilere tanıtılır.

Uygula: Havacılıkta kullanılan değişik malzemelerin ağırlık ve fiziksel özelliklerinin incelenmesi

Tasarla: Bir Jet uçağı için hangi parçasını hangi malzemelerden üretilmesinin tasarlanması

Üret: Uçak kanatlarının kalıp yardımı ile üretim adımlarını oluşturma

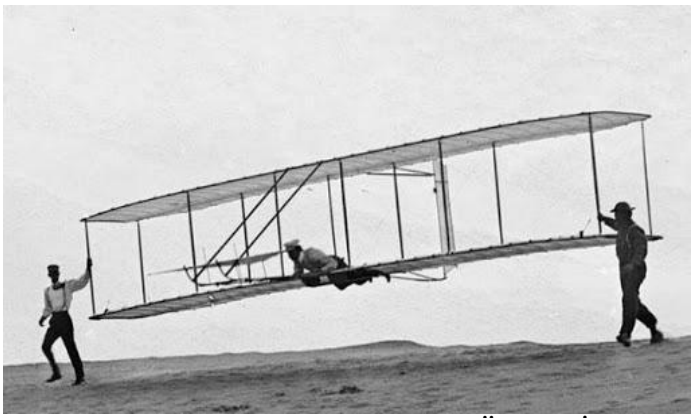
Değerlendir: Haftanın içeriği ile ilgili yansıtma etkinliği

1. GÖZLE VE UYGULA

1.1. Gözle: Havacılıkta Kullanılan Malzemeler

Havacılık endüstrisinin malzemeden beklentisi hem hafiflik hem de yüksek dayanımı aynı anda sağlayabilmesidir. Hava aracı uçuş döngüsü boyunca (kalkış-seyir-iniş) üzerine etki eden kuvvetlere karşı yeterli dayanımı göstermek zorundadır. Bunun yanında hava aracının hafif olması taşınabilecek faydalı yükü artırmakta, uçuş süresini uzatmakta, daha az yakıt tüketimi sağlamakta ve buna bağlı olarak hem karbon salınımını hem de uçuş maliyetini düşürmektedir. Bunlara ek olarak uçak mühendisleri bir parça için malzeme seçimi yaparken o parçanın çalışma koşullarını da dikkate almak zorundadır. Parçanın statik yüklere mi yoksa dinamik yüklere mi maruz kalacağı, hangi sıcaklıklarda çalışacağı, elektriksel ve kimyasal özellikleri malzeme seçiminde önemli parametrelerdir.

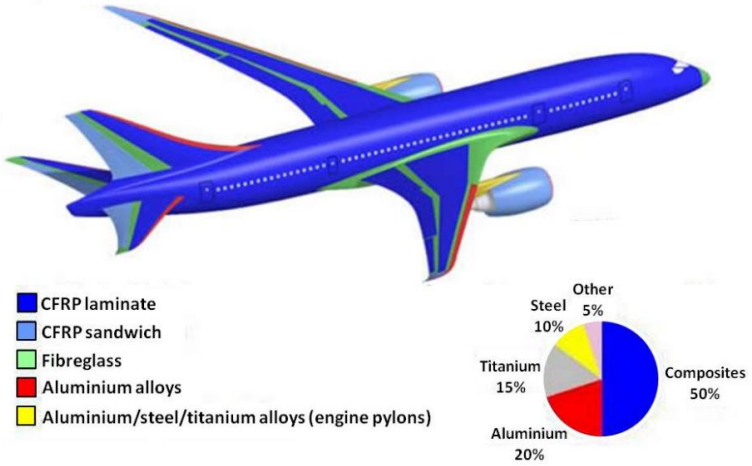
Havacılık tarihi incelendiğinde bu alanda kullanılan malzemelerin zaman içerisinde önemli ölçüde değiştiği göze çarpmaktadır. Şekil 1'de görülebilen Wright kardeşlerin 1903 yılında ürettikleri uçak alüminyum bir motora, ağaç ve çelik karışımı bir konstrüksiyona ve kumaş kanat kaplamalarına sahipken 1915 yılında Hugo Junkers tarafından tamamen metal malzemeler kullanılarak yapılan uçak Şekil 2'de gösterilmiştir. 2. Dünya Savaşı'na kadar uçaklar genellikle metal malzemelerden üretilmiş, savaş sırasında metal ticaretinde yaşanan sorunlarla birlikte polimer parçalar havacılıkta kullanılmaya başlanmıştır.



Şekil 1. *Wright Kardeşler Tarafından Üretilen İlk Uçak*

Şekil 2. *Hugo Junkers Tarafından Üretilen Tamamı Metal Olan İlk Uçak*

Günümüzde polimer malzemeler sahip oldukları yüksek mukavemet ve düşük yoğunluk özellikleri sayesinde havacılıkta oldukça yaygın kullanım alanına sahiptir. Polimerik kompozit malzemelerin havacılıkta birçok parçada kullanımı daha eskiye dayansa da 2009 yılında Boeing 787-8 Dreamliner model yolcu uçağı karbon fiber takviyeli polimer malzemeden üretilmiş kanatlara ve gövdeye sahip ilk uçak olarak uçuşunu yapmıştır. Uçağın malzeme dağılımı Şekil 3'te gösterilmiştir.



Şekil 3. *Boeing 787 Malzeme Dağılımı*

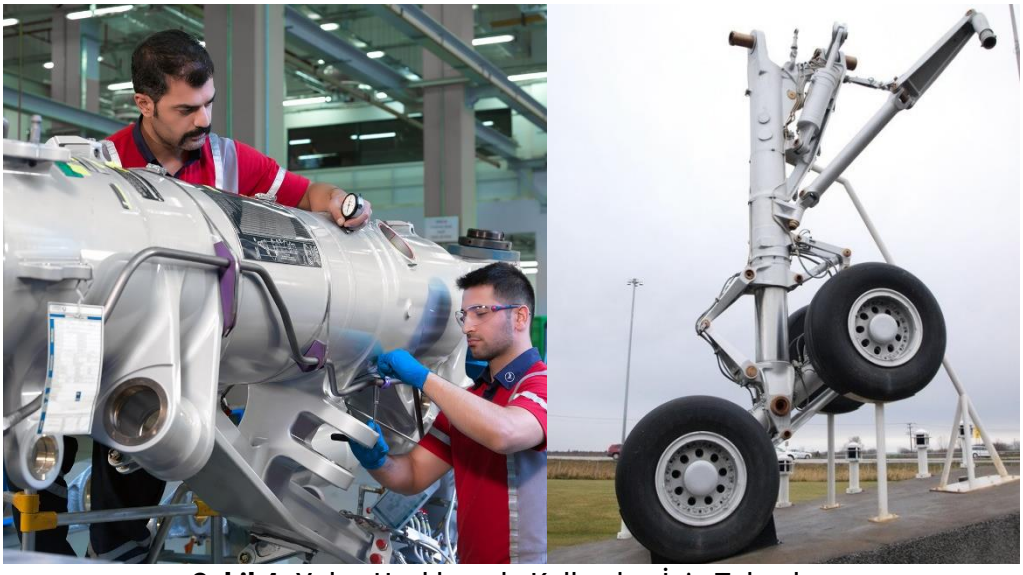
1.2. Gözle: Havacılıkta Kullanılan Metaller

Havacılıkta metal malzemelerin kullanımı havacılık tarihi kadar eskidir. Üretilen ilk uçaklarda gövde ve kanat malzemeleri olarak ahşap gibi doğal malzemeler kullanılsa da özellikle motor gibi parçalarda metal malzemeler kullanılmıştır. İlerleyen zamanlarda da havacılıkta metal malzemelerin kullanımı malzeme teknolojisinin gelişmesi ve yeni malzemelerin bulunmasıyla zaman zaman artıp azalsa da günümüze kadar devam etmiştir. Günümüzde çelik, alüminyum, titanyum gibi metalik malzemeler havacılıkta çeşitli parçalarda kullanılmaktadırlar.

Alüminyum alaşımları sahip oldukları düşük yoğunluk, (2700 kg/m^3) iyi mukavemet ve özellikleri ve korozyon direncinden dolayı günümüz havacılık sektöründe en yaygın kullanılan metallerdendir. Alüminyum alaşımlarının fiyatları çeliklerden daha pahalıdır. Yine de hava aracını daha hafif yapabilmek için ilk tercih edilen metallerdendir. Çünkü hava aracının daha hafif yapılması yakıt tasarrufu sağlayarak işletme maliyetlerini düşürmektedir. Alüminyum parçaların dayanımları yüksek sıcaklıklarda düşmesine rağmen yüksek irtifalardaki çok düşük sıcaklıklarda oldukça iyidir. Gövde ana taşıyıcı ve kaplamaları, kanatlar, koltuk ayakları, tekerler ve stabilizörler gibi çok kritik parçalar yaygın bir şekilde alüminyumdan üretilmektedirler.

Motorun bir uçağın en kompleks parçalarından biri olduğu söylenebilir. Modern bir jet motoru dikkate alındığında motor içerisinde kullanılan malzemelerin 2100°C sıcaklığa kadar çalışabilmektedir. Mühendisler bu sıcaklıklarda çalışacak bileşenler için titanyum alaşımları, nikel alaşımları gibi metalik malzemeler ile seramik gibi metalik olmayan malzemeler kullanmaktadırlar. Bu malzemeler diğer yaygın kullanılan metallerden (çelik ve alüminyum gibi) daha pahalı olmasına rağmen sahip oldukları çok yüksek sıcaklık dayanımı özelliklerinden dolayı tercih edilmektedirler.

Uçaklarda kullanılan bir diğer metal grubu ise çelik malzemelerdir. Çelik malzemeler yüksek yoğunluğa sahip olmalarına rağmen yüksek mukavemet özellikleri nedeniyle bazı kritik parçalarda kullanılırlar. Örneğin iniş takımlarına uçağın inişi sırasında ani olarak oldukça yüksek stresler etki etmektedir. Burada kullanılan malzemenin ani darbe yüklerine karşı yüksek dayanım göstermesi gerekmektedir. Uçağın ana yapısında bulunan bazı birleştirme elemanlarında da yüksek mukavemet özellikleri sebebiyle çelik malzemeler tercih edilmektedir. Ayrıca çelik yapıların orta-yüksek sıcaklıklarda çalışabilmesi bu yapıların motor ve egzoz bileşenlerinde de kullanılmasını mümkün kılmıştır.



Şekil 4. Yolcu Uçaklarında Kullanılan İniş Takımları

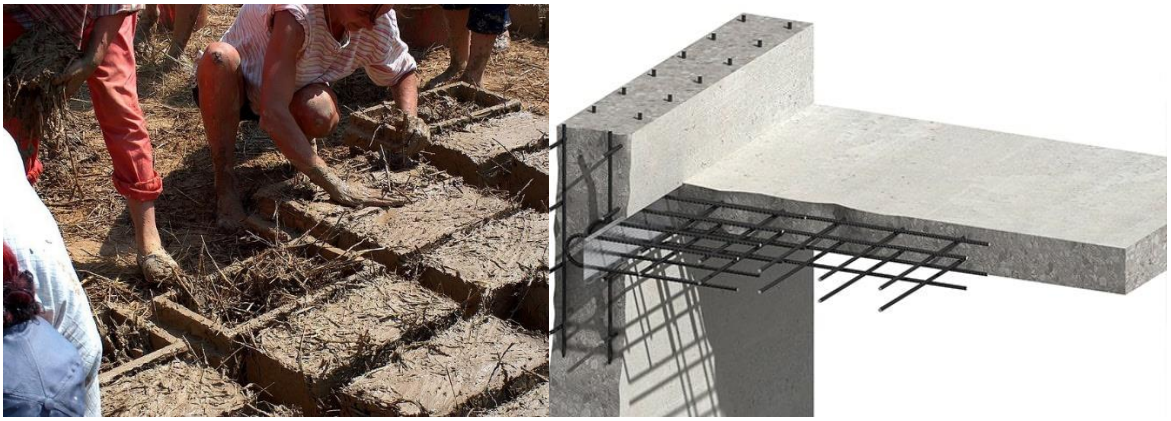
1.3. Gözle: Kompozit Malzemeler

Kompozit malzemelerin günlük hayatta kullanımı yüzyıllar öncesine dayansa da mühendislik alanında yaygınlaşması son yüzyılda meydana gelmiştir. Özellikle havacılık alanında ihtiyaç duyulan yüksek mukavemetli ve düşük yoğunluklu malzeme kullanımı kompozit malzemeleri öne çıkarmıştır. Kompozit malzemelerin kullanımıyla daha hafif hava araçları yapmak mümkün olmuş, bu sayede hem önemli miktarda enerji tasarrufu sağlanmış hem de daha fazla faydalı yük taşımak mümkün olmuştur.

Kompozit malzemeler yük taşıyıcı (takviye) ve bağlayıcı (matris) görevinde birden fazla farklı malzemenin birleştirilmesiyle oluşan malzemelere denilmektedir. Kompozitler, matris malzemelerine göre metal, seramik ve polimer matris kompozitler olarak, takviye malzemelerine göre ise parçacıklı, fiberli ve tabakalı kompozitler olarak sınıflandırılırlar. Ayrıca bir kompozit malzeme çeşidi olan sandviç yapılar da havacılıkta yaygın olarak kullanılmaktadır. Sandviç yapıların yüzey tabakaları ana yük taşıyıcı malzemelerden (karbon fiber kompozit, metalik malzemeler vb.) oluşurken orta kısımda ise yapının kalınlığını artırmak için konulan çok düşük yoğunluklu bir çekirdek malzeme (strafor, PVC veya metalik köpükler, bal peteği yapılar) bulunur.

Kompozit tarihinin çamur ve samanla başladığı düşünülmektedir. Saman çekme yönünde güçlü bir malzemedir ancak kolayca bükülebilir. Çamur ise iyi şekil verilip kurutulabilen bir malzemedir ancak mukavemeti zayıftır. İnsanlar bu iki malzemeyi birleştirerek tuğla yapmışlar ve samanın sağlamlığını kullanarak çamurun mukavemetini iyileştirmişlerdir. Beton da en eski kompozit malzemelerin bir örneğidir. Eski zamanlarda çimento içerisine taş, kum gibi malzemeler

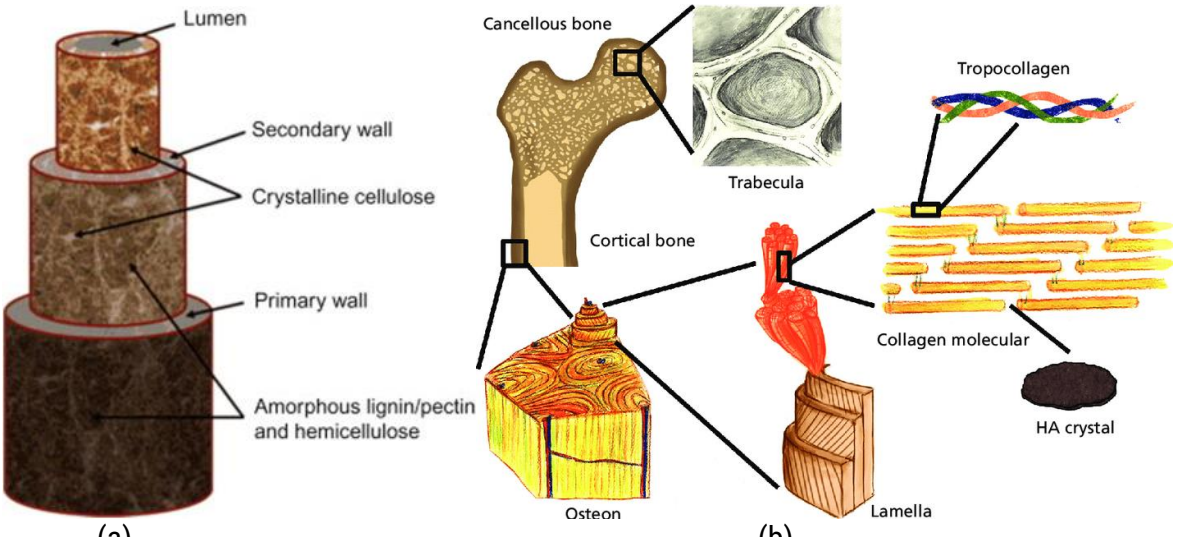
katılarak bir kompozit yapı olan beton üretilirken sonraki zamanlarda içerisine demir çubuklar eklemişler ve yapının mukavemetini artırmışlardır.



Şekil 5. *Saman - Çamur ve Demir - Beton Kompozit Yapılar*

1.4. Gözle: Doğada Kompozit Malzemeler

Kompozit malzemeler doğada sıklıkla karşımıza çıkan malzemelerdendir. Bir ağacın gövdesi incelendiğinde lifli bir yapıda olduğu ve bu liflerin lignin adı verilen malzemeyle bir arada tutulduğu görülmektedir. Bunun sonucunda oldukça sağlam bir yapı olan ağaç gövdesi oluşmaktadır. Kemik yapısı ise doğal kompozit malzemelere bir diğer örnektir. Hafif ve sağlam yapısıyla bilinen kemik dokusu, kalsiyum tuzu ve kollajen malzemelerinin birleşiminden oluşmaktadır. Kemiğin boşluklu yapısı ise hafifliği sağlamaktadır.



Şekil 6. *Doğada Gözlemlenebilen Kompozit Malzemeler; (a): Ağaç Gövdesi, (b): Kemik Yapısı*

1.5. Gözle: Kompozit Malzemelerin Uygulama Teknikleri

Günümüzde birçok sektörde kullanılan kompozit malzemeler takviye ve matris fazlarından oluşmaktadır. Yaygın olarak kullanılan takviye malzemeler karbon fiber, cam fiber, aramid fiber gibi malzemelerdir. Bu malzemeler farklı sayıda lifler halinde kullanılabildiği gibi farklı örgü tiplerinde kumaş halinde de kullanılabilmektedir. Yaygın olarak kullanılan matris malzemeler ise epoksi, polyester ve vinil ester gibi malzemelerdir. Havacılıkta da sıklıkla kullanılan polimer matrisli kompozit yapıların çok çeşitli üretim metotları vardır. Tüm üretim metotlarının ortak özelliği ise takviye yapılarla matris yapıyı uygun bir şekilde bir araya getirmek ve yapının katılaşmasını (kürleşmesini) sağlamaktır. Sıklıkla kullanılan bazı üretim yöntemleri aşağıda verilmiştir.

- **Elle yatırma:** En eski kompozit üretim yöntemlerindendir. Bu yöntemde sıvı formdaki matris malzeme bir fırça aracılığıyla takviye yapılara sürülür. Birleştirme işlemi tamamlandığında yapı kürleşmeye bırakılır. Kürleşme işleminin nasıl olacağı matris olarak kullanılan reçinenin türüne göre değişir. Atmosfer basıncında oda sıcaklığında yapılabileceği gibi vakum basıncında veya fırında yüksek sıcaklıkta da yapılabilir. Vakum basıncında yapılırsa bu yöntem **elle yatırma ve vakum torbalama** yöntemi olarak adlandırılır. Elle yatırma yöntemi esnek ve düşük maliyetli bir üretim yöntemidir. Ancak üretim hızı düşüktür ve üretim kalitesi uygulayıcıya doğrudan bağlıdır.
  (https://www.youtube.com/watch?v=EhAvCqtIo7w)

- **Püskürtme:** Bu yöntemde takviye ve matris malzemeler bir püskürtme tabancası ile kalıba püskürtülür. Takviye malzeme tabancaya kadar sürekli filament halinde gelir ve tabancada kırpılarak dışarı atılırken sıvı formdaki matris malzeme püskürtülerek aynı anda kalıba gönderilir. Kürleşme işlemi elle yatırma yönteminde olduğu gibi reçine tipine göre farklı sıcaklıklarda ve basınçta yapılabilir. Bu yöntem de esnek ve düşük maliyetlidir ancak üretim hızı düşük, üretim kalitesi uygulayıcıya bağlıdır.
  (https://www.youtube.com/watch?v=0iYnfglPkkU)

- **Prepreg yatırma:** Bu yöntemde önceden reçine emdirilmiş prepreg kumaşlar kullanılmaktadır. Prepreg kumaşlar uygulanacak zamana kadar soğutucuda saklanmalıdır. Uygulama anında çıkarılıp istenilen oryantasyonda kalıba dizildikten sonra otoklav fırınlarda sıcaklık ve basınç

altında kürleştirilirler. Bu yöntemde üretim daha hızlıdır ve uygulayıcıdan daha az etkilenir. Ancak bu üretim yönteminde ilk yatırım maliyeti yüksektir.
(https://www.youtube.com/watch?v=Um0_am_XAmw)

- **RTM:** Takviye kumaşlar istenilen dizilimlerde dişi ve erkek kalıplar arasına dizilirler. Kalıplar kapatılır ve sızdırmazlığı sağlanır. Sonrasında ise reçine belirli bir basınçta ve akış hızında kalıba gönderilir. Kalıba yetecek kadar reçine aktarıldıktan sonra reçine akışı kesilir ve yapı kürleşmeye bırakılır. Kürleşme zamanı sonunda kalıplar açılır ve ürün çıkarılır. Bu yöntemle benzerliği olan bir diğer yöntem ise **reçine infüzyon** yöntemidir. Bu yöntemde tek taraflı kalıp kullanılırken yapının diğer yüzü vakum torbasıyla kapatılır. Kalıbın uygun bir noktasından reçine transfer edilirken kalıbın diğer tarafından vakum uygulanır ve reçinenin tüm yapıya ulaşması sağlanır.
(https://www.youtube.com/watch?v=PvYFTdI0z2I)

- **Filament sarma:** Genellikle silindirik parçalar üretmek için kullanılan yöntemdir. Sürekli yapıdaki filamentler önce reçineden geçirilerek bir mandrele (kalıba) sarılır. Sarma açısı ve sıklığı istenilen ölçülerde ayarlanabilir. İşlem bittikten sonra yapı kürleşmeye bırakılır. Kürleşme sonunda mandrel çıkarılır.
(https://www.youtube.com/watch?v=ign6W5ENJAA)

Kompozit malzemelerin konvansiyonel malzemelere kıyasla çeşitli avantaj ve dezavantajları bulunmaktadır. En önemli avantajının hafif olduğu söylenebilir. Bu da bir hava aracı için daha az yakıt tüketimi ve daha uzun menzil gibi çeşitli avantajlar sağlamaktadır. Bunun yanında takviye elemanlarının dizilim yönüne göre farklı doğrultularda farklı mekanik özellikleri ayarlamak mümkün olmaktadır. Ayrıca kompozit malzemeler korozyona, kimyasallara, dış hava şartlarına oldukça dirençlidir. Kompozit malzemelerin önemli dezavantajları için ise pahalı olması, metallere göre daha kırılgan olması, tamirinin daha zor olması, kürleşme sürecinin zaman alması gibi maddeler sıralanabilir.

1.6. Uygula: Kompozit Malzemelerin Ağırlık ve Fiziksel Özelliklerinin Tespiti


Karbon Levha


Cam Elyaf Levha


Alüminyum Levha


Balsa Levha

Yukarıdaki resimde görülen malzemeleri aşağıdaki tabloya göre değerlendiriniz.. Sınıfta bulunan tartı ile levhaları tartarak tabloya yazınız. Diğer bilgileri araştırıp yorumlayıp tabloya yazınız.

Malzeme	Ağırlık 90 mm X 90 mm	Dayanım	Esneklik	Fiyat
Karbon Levha				
Cam Elyaf Levha				
Balsa Levha				
Alüminyum Levha				

2. TASARLA

2.1. İHA'lar İçin Uygun Malzeme Seçimi

Tasarla aşamasında öğrencilerden bir sabit kanatlı İHA'nın parçaları üretilecek şekilde malzeme listesini hazırlamaları istenir. Öğrencilerden malzeme seçiminin nasıl yapılacağı üzerinde düşünmeleri istenir. Öğrenciler grup olarak tartışırlar. Gerektiği noktada rehber eğitmen onlara yardımcı olabilir. Fakat öğrencilere tam bir çözüm verilmemelidir. Gruplar çözümü kendileri üretmelidir. Tam bir çözümün verilmesi öğrencilerin yaratıcılıklarını olumsuz yönde etkileyebileceğinden tavsiye edilmez. Sabit kanat bir İHA tasarlamak için öğrencilerin aşağıda örnek olarak verilen iki adıma benzer bir süreci gerçekleştirmesi gerekir.

Tanımlama: Öncelikle öğrenciler İHA'nın bileşenlerini belirlemelidir. Tasarlanacak İHA mini veya bir uçak olabilir. Öğrenciler gerekli işlemleri maddeler hâlinde yazmalıdır.

Örneğin;

- İHA'nın veya uçağın parçaları,
- Parçaların konumu ve sağlamlık-hafiflik döngüsü,
- Kullanılacak metal olan malzemelerin yerleri,
- Liste halinde bir uçak veya İHA'daki tüm bileşenlerin belirlenmesi.

Fikir üretme: Bu aşamada öğrencilerin tanımlamada belirlenen işlemlerin nasıl yapılabileceği ile ilgili fikir yürütmesi beklenir. Örnek olarak öğrenciler İHA veya uçakların tasarımlarına bakarak fikir üretebilirler.

Bayraktar İHA, F35 savaş uçağı ve Airbus yolcu uçağı.

3. ÜRET

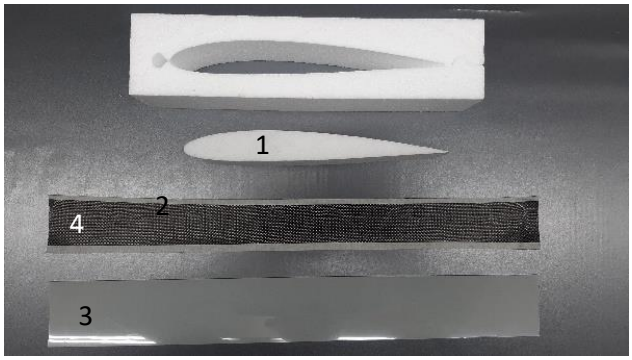
3.1. İHA Karbon Kanat Yapımı

Kanatlar bir uçağın uçuş sırasında çok çeşitli yüklere maruz kalan parçalarından birisidir. Kanatların hem yüksek dayanım göstermesi hem de düşük ağırlığa sahip olması istenir. Bu yüzden günümüz hava araçlarında kompozit malzemelerin sıklıkla tercih edildiği bir bölgedir. Bu kısımda aşağıdaki adımlar takip edilerek öğrencilerin bir kompozit kanat kesitini yapması sağlanacaktır. Kompozit kanat üretiminde öğrencilerin aktif rol üstlenmesi beklenmektedir. Rehber eğitmen yalnızca yönlendirir ve öğrencilere takıldıkları noktalarda destek olur. Yani rehber eğitmen yalnızca ihtiyaç anında destek sağlamalıdır.

Üret aşamasında, öğrencilerden önceden hazırlanmış bir İHA kanat kesitinin karbon malzeme kullanarak kompozit bir kanat yapmaları istenir. Öğrenciler, uygulama aşamasında öğretilen yapım tekniklerini kullanarak el becerileri yardımı ile kompozit bir ürün geliştirirler. Burada öğrencilerin en çok zorlanacağı konulardan biri reçine karışımının doğru oranda hazırlanması ve kalıp içerisine çekirdek kanat kesitinin yerleştirilmesi olacaktır.

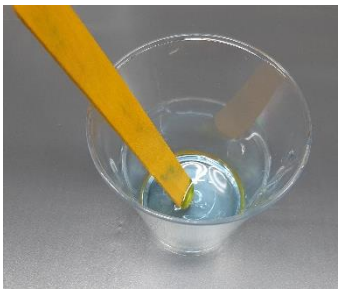
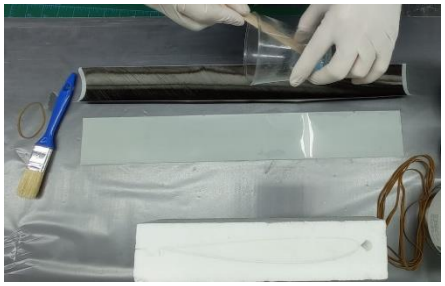
Uygulama esasında kullanılacak malzemeler

1. Erkek köpük kalıp
2. Dişi köpük kalıp
3. Mayler ayırıcı film
4. Karbon kumaş

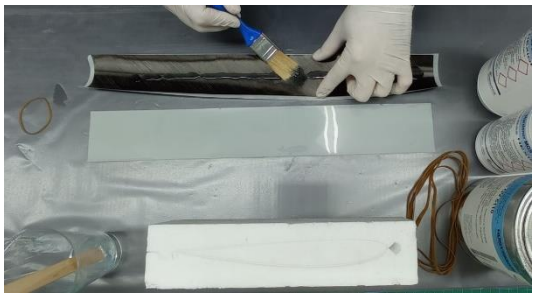


5. Epoksi reçine
6. Epoksi sertleştiricisi
7. Kalıp ayırıcı vaks
8. Fırça
9. Pamuk
10. Bardak
11. Karıştırma çubuğu
12. Eldiven
13. Lastik
14. Ayırıcı kâğıt

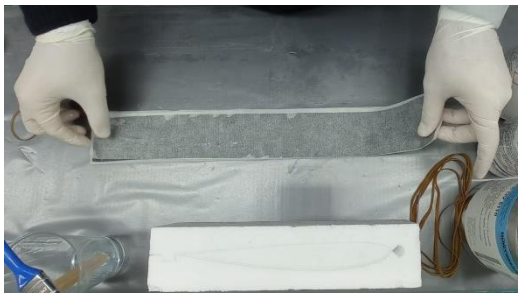
Karbon kanat kesiti uygulama aşamaları

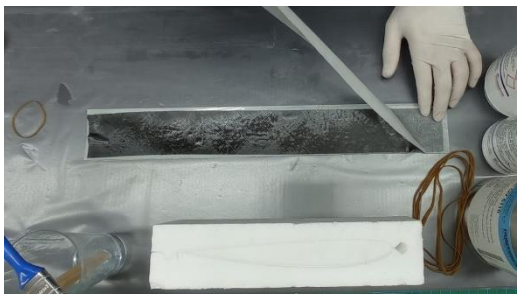




1- Beyaz mayler film tabakasını düz bir şekilde masaya yatırınız. Pamuğu vaks kutusuna batırarak film tabakasının bir yüzüne vaksı sürünüz. Bu işlemi 2 dakika ara ile 3 defa yapınız.
2- Küçük tüpler içerisindeki epoksi reçine ve epoksi sertleştiricisini plastik bardak içerisine boşaltarak karıştırma çubuğu ile iyice karıştırınız.
3- Karbon fiber kumaşı ayırıcı kâğıdın üzerine serip epoksi reçineyi dökünüz.

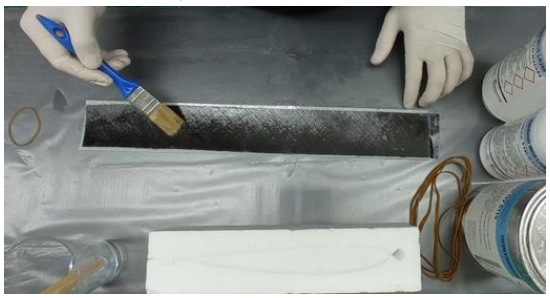



4- Fırça yardımı ile karbon fiber kumaş dokuma yönüne paralel olacak şekilde epoksiyi iyice yayınız.
5- Epoksi sürme işlemi bittikten sonra 5 dakika bekleyiniz.

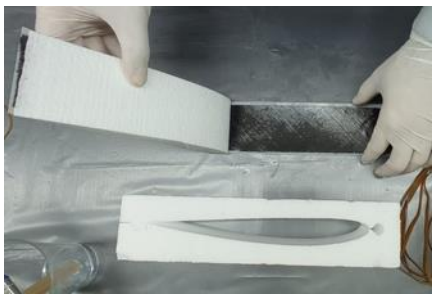


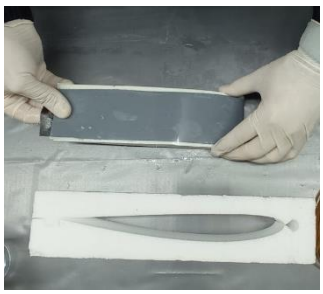


6- Ayırıcı kâğıt ile birlikte reçine sürülen karbon fiber kumaşı ters çevirip mayler filmin üzerine yerleştiriniz.
7- Transfer işlemi tamamlandıktan sonra ayırıcı kâğıdı kumaş üzerinden alınız.

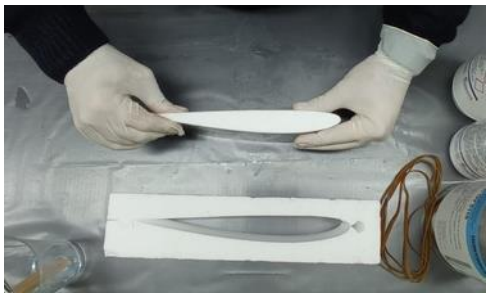


8- Eğer eğrilik, tam yapışmama gibi sorunlar varsa fırça ile düzeltme işlemleri yapılarak kumaşın düz olması sağlanır.
9- Köpük plaka içerisinden kanat profili çıkarılır.

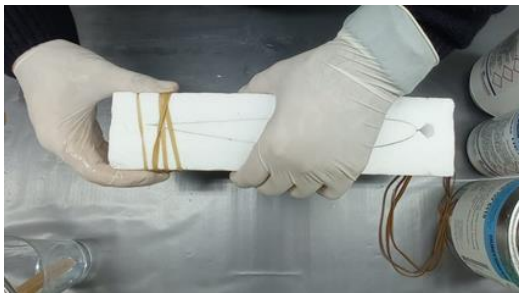


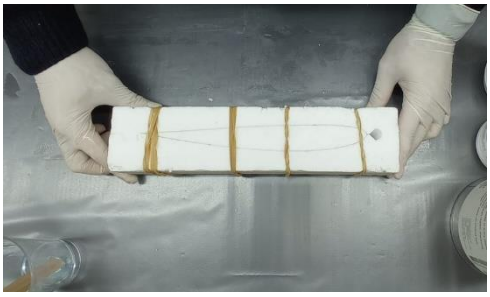


10- Reçine ile ıslatılmış karbon fiber kumaş kanat profiline, karbon fiber kumaş kanat profili tarafına, mayler film dış tarafa gelecek şekilde düzgünce hizalanarak sarılır.
11- Kanadın firar kenarından tutarak düzeltmeler ve hizalamaları yapılır.



12- Kanat profili dişi kalıp içerisine konmadan önce hava boşlukları ve kenar düzeltmeleri tamamlanır.
13- Kanat profili dişi kalıp içerisine düzgünce yerleştirilir.





14- Kalın paket lastikleri ile kalıp sıkıştırılır ve 8 -10 saat oda sıcaklığında beklemeye bırakılır.

Yapımı tamamlanan karbon fiber kanat kesitinin ne kadar sağlam ve hafif olduğunu, günümüz uçaklarından neden kullanıldığını değerlendirebilirsiniz.

4. DEĞERLENDİR

Bu aşamada hedef, öğrencilerin öğrenme sürecinde yaşadıkları ve öğrendikleri üzerine düşünmesini sağlamaktır. Bu sayede öğrenciler problem çözme yetenekleri, dersin konusu ve kendileri ile ilgili gözlemler yaparak yeni bilgiler öğrenme, kendilerini değerlendirme ve sonraki çalışmalarını planlama açısından fırsatlar elde edecektir. Öğrencilerden şu soruları cevaplamaları istenebilir:

- Doğada gördüğünüz hafif ama sağlam olarak bilinen yapılar nelerdir?

  Bal peteği, örümcek ağı, ipek böceği ipliği, yaprak damarı, incinin dış kabuğu…
- İnsan vücudunda böyle bir yapıya örnek verebilir misiniz?

  Kemik dokusu, diş minesi…
- Bütün bu yapıların tasarımsal olarak ortak noktaları nelerdir?

Geometrik olarak düzenli ya da düzensiz, tekrarlı, sürekli, üç boyutlu ve boşluklu olması sebebi ile hafifletilmiş olması gibi cevapların öğrenciler tarafından verilmesi beklenir.

- Bu yapıları taklit ederek oluşturulan gözle görülebilen ve gözle görülemeyen boyutta bildiğiniz mühendislik malzemeleri var mı?

  Metalik köpük (deniz süngeri yapısı), honeycomb (bal peteği taklidi) (gözle görülebilen), grafen ve karbon nanotüp (gözle görülemeyen)…
- Kompozit malzemeler bu yapılarla nasıl güçlendirilir?

  Sandviç yapı (yapısal köpük, honeycomb takviyesi), nano parçacık takviyesi ile güçlendirme…

Değerlendirme, öğrencileri sıkmadan, her bir soru için verilen cevaplar tatmin edici bir düzeye ulaşıncaya kadar devam ettirilebilir.

5. İLAVE ETKİNLİK

5.1. En Hafif Kompozit Kanadın Tespiti

Bu ilave etkinlikteki amaç üretimi yapılan kompozit kanadın en hafif olanını tespit etmektir. Sınıf içerisindeki tüm öğrenciler bir tabloya ölçüleri aynı olan karbon kumaştan üretilmiş kanadın işçilik farklılıklarından kaynaklanan fiziksel değişimleri görmeleri beklenir.

Öğrencilerden bu işlem için kullanılan reçinenin az veya fazla olmasında nasıl bir durumla karşılaşıldığını görmeleri beklenir. Dijital tartı ile tüm öğrencilerin kanatlarını tartıp en hafif ve en ağır olanı belirlenir.

3. HAFTA: HAVA ARAÇLARINDA KONTROL YÜZEYLERİ

Ön Bilgi:

- Temel bilgisayar ve donanım bilgisi

Haftanın Kazanımları:

- Öğrenciler temel hava araçlarının manevra becerilerini kavrar.
- Sabit kanat hava araçlarının kanat bileşenlerini tanımlar.
- Hava aracının dengeli uçuşunu sağlayan unsurları bilir.
- Sabit kanat bir İHA'yı model uçak seviyesinde birleştirir.
- Kontrol yüzeylerinin kumanda ile ilişkisini yorumlar.

Haftanın Amacı:

Bu hafta öğretilecek başlıkların amacı, hava araçlarının uçuş manevralarını sağlatan kontrol yüzeylerini kavratmaktır. Öğrenciler birleştirecekleri sabit kanat model uçak ile bir uçağın kontrol yüzeylerini ve bu yüzeylerin hareketlerini kavrayacaktır. Uçağın ağırlık merkezinin bulunması ve ağırlık merkezinin uçuş dengesi için önemini yorumlayacaklardır.

Kullanılacak Malzemeler:

Model İHA kiti, servo motor fırçasız DC motor, RC kumanda alıcı ve vericisi.

Haftanın İşlenişi:

Gözle: Hava aracının kontrol yüzeylerinin tanıtılması, çalışma prensipleri ve uçağın manevra hareketleri.

Uygula: Ağırlık merkezi hesaplamaları.

Üret: Model İHA kitinin birleştirilerek sabit kanatlı bir model uçak yapılması.

Değerlendir: Birleştirilen model uçağın kontrol yüzeylerini büyük uçaklarla kıyaslayarak öğrenciye yorumlatılması. Kontrol yüzeylerinin isimleri ve işlevlerinin tekrarlanarak pekiştirilmesi.

1. Gözle: Hava Araçlarında Kontrol Yüzeyleri

Tanım

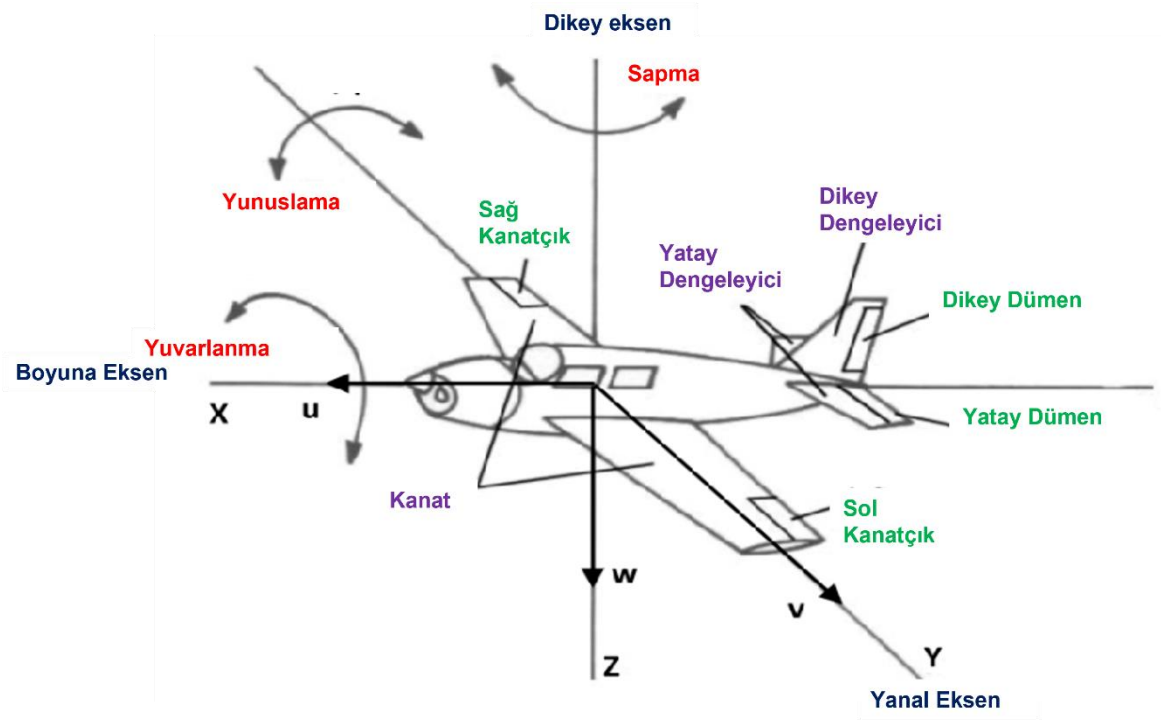
Bir hava aracının uçuş kontrolleri, bir pilotun uçuş halindeki bir uçağın yönünü ve durumunu kontrol ettiği araçlardır.

Uçuş kontrol sistemleri, birincil ve ikincil uçuş kontrolleri olarak ikiye bölünmüştür.

Birincil uçuş kontrolleri, uçuş sırasında bir uçağı güvenli bir şekilde kontrol etmek için gereklidir kanatçıklar (aileron), yatay dümen (elevatör) ve dikey dümenden (rudder) oluşur.

İkincil uçuş kontrolleri, uçağın performans özelliklerini iyileştirmeyi veya aşırı kontrol yükünü hafifletmeyi amaçlar ve slat, flap gibi yüksek kaldırma cihazlarının yanı sıra uçuş spoyleri ve trim sistemlerinden oluşur.

Hava araçları uçuş sırasında üç boyutlu manevralar yapar. Bu hareketi kontrol etmek için pilot, uçağın üç dönme ekseninden biri veya daha fazlası etrafında dönmesini sağlamak için uçuş kontrollerini koordine eder. Boyuna, yanal ve dikey olarak adlandırılan bu üç eksen, birbirine diktir ve uçağın ağırlık merkezinde kesişir.



Boyuna eksen, yanal eksen ve dikey eksen etrafındaki hareket sırasıyla yuvarlanma (roll), yunuslama (pitch) ve sapma (yaw) olarak adlandırılır. Yuvarlanmayı kontrol etmek için kanatçıklar, yunuslamayı kontrol etmek için yatay dümen ve sapma için dikey dümen kullanılır. Bir eksen etrafında dönüş, ikinci bir eksen etrafında komutsuz bir dönüşe neden olabilir. Bu durumda, koordineli uçuşu sürdürmek için ilgili uçuş kontrol yüzeylerinin eşzamanlı kullanımı gereklidir.

Koordineli uçuş, kayma ve irtifa değişiminin olmadığı istenen bir uçuş koşuludur. Koordineli uçuşu sürdürmek için dönüşlere girerken ve dönüşlerden çıkarken karma kanatçık, yatay dümen ve dikey dümen girişleri gereklidir.

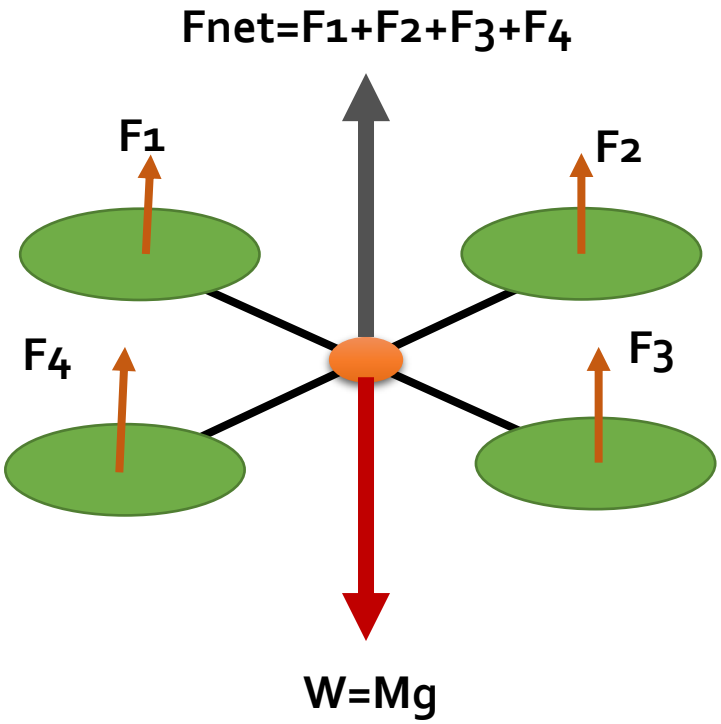
Kanatçık (Aileron)
Kanatçıklar, bir uçağın uzunlamasına ekseni etrafındaki hareketi kontrol eden birincil uçuş kontrol yüzeyidir. Bu harekete "yuvarlanma" denir. Kanatçıklar, her bir kanadın dıştan takmalı arka kenarına takılır ve manuel veya otomatik pilot kontrol girişi yapıldığında birbirinden zıt yönlerde hareket eder. Bazı büyük uçaklarda, her kanatta iki kanatçık bulunur. Bu konfigürasyonda, her bir kanattaki her iki kanatçık, yavaş hızda uçuş sırasında aktiftir. Ancak, daha yüksek hızda, dıştan takmalı kanatçık kilitlenir ve yalnızca iç veya yüksek hız kanatçıkları çalışır.

Yatay Dümen (Elevatör)
Yatay dümen, bir uçağın yan ekseni etrafındaki hareketi kontrol eden birincil uçuş kontrol yüzeyidir. Bu harekete "yunuslama" denir. Çoğu uçakta, biri yatay dengeleyicinin her bir yarısının arka kenarına monte edilmiş iki yatay dümen bulunur. Manuel veya otopilot kontrol girişi yapıldığında yatay dümenler uygun şekilde yukarı veya aşağı hareket eder. Çoğu kurulumda, yatay dümenler simetrik olarak hareket eder, ancak bazı kablosuz kontrollü uçaklarda, kontrol girdi taleplerini karşılamak için gerektiğinde farklı şekilde hareket ederler. Bazı uçak tiplerinde, bir kontrol yüzeyi sıkışması durumunda sağ ve sol yatay dümenlerin birbirinden "bağlantısını kesmek" için hükümler bulunurken, diğer tipler, olayda en az bir yüzeyin çalışır durumda olmasını, sol ve sağ yatay dümene güç sağlamak için farklı hidrolik sistemler kullanır.

Dikey Dümen (Rudder)
Dikey dümen, bir uçağın dikey ekseni etrafındaki dönüşü kontrol eden birincil uçuş kontrol yüzeyidir. Bu harekete "sapma" denir. Dikey dümen, dikey dengeleyicinin veya kanadın arka kenarına monte edilmiş hareketli bir yüzeydir. Bir teknenin aksine, dümen uçağı yönlendirmek için kullanılmaz; bunun yerine dönüşten veya çok motorlu bir uçak olması durumunda motor arızasından kaynaklanan ters yalpalamanın üstesinden gelmek için kullanılır ve ayrıca gerektiğinde uçağın kasıtlı olarak kaymasına izin verilir.

Döner Kanatlı Hava Araçlarında Manevra
Helikopter gibi tek rotorlu döner kanat hava araçlarında yunuslama ve yuvarlanma hareketleri için rotor pallerinin açılı kontrolünden faydalanılır. Sapma hareketi için ise kuyuruk rotorunun hızı değiştirilerek ana rotora karşı oluşan tepkiden faydalanılır.

Quadcopter ve benzeri çok rotorlu araçlarda ise manevra hareketleri rotor hızlarının kontrolü ile yönetilir.



Fnet=W	**Dengede Kalma (Hover)**
Fnet>W	**Yükselme**
Fnet<W	**Alçalma**
$F_4+F_3 > F_1+F_2$	**İleri Yunuslama (Pitch)**
$F_1+F_2 > F_4+F_3$	**Geri Yunuslama (Pitch)**
$F_1+F_4 > F_2+F_3$	**Sağa Yuvarlanma (Roll)**
$F_2+F_3 > F_1+F_4$	**Sola Yuvarlanma (Roll)**
$F_1+F_3 > F_2+F_4$	**Sağa Sapma (Yaw)**
$F_2+F_4 > F_1+F_3$	**Sola Sapmaa (Yaw)**

Her bir motor yukarı yönde bir itki sağlayarak ağırlığı yenecek bir kuvvet oluşturur. Motorlardaki hız artışları ve düşüşleri ile kuvvetler yönetilerek aracın yükselme, alçalma, dengede kalma, yunuslama, yuvarlanma ve sapma hareketleri kontrol edilebilmektedir.

2. Gözle: Ağırlık Merkezi

Bir uçakta, ağırlık merkezi (CG), uçağın o noktada askıya alınması mümkün olsaydı, uçağın dengede kalacağı noktadır. Ağırlık merkezinin konumu, uçağın dengesini etkilediğinden, uçak üreticisi tarafından belirlenen belirli sınırlar içinde kalmalıdır. Hem yanal hem de boylamsal denge önemlidir ancak asıl mesele boylamsal dengedir. Yani, CG'nin uzunlamasına veya uzunlamasına eksen boyunca konumudur.

Boş ağırlık ve boş ağırlık ağırlık merkezi (EWCG), her bir tartım noktası için bir dizi tartım terazisi, üretici tarafından tanımlanan referans verisi, hava aracı tip sertifikası veri sayfasında listelendiği gibi kollar kullanılarak her uçak için hesaplanır. Uçağın boş ağırlığı, her bir tartım noktasından gelen ağırlıkların toplamıdır. EWCG, tartım noktalarının her biri için hesaplanan momentler, hesaplanan boş ağırlık ve Uçak Uçuş El Kitabında (AFM) belirtildiği gibi uçak tipi için uygun formül kullanılarak hesaplanır. Küçük uçaklarda ve helikopterlerde, ağırlık merkezi konumu, referans noktasından belirli bir inç sayısı olarak tanımlanır ve ağırlık merkezi aralığı da aynı şekilde tanımlanır. Daha büyük uçaklarda, ağırlık merkezi ve menzili tipik olarak kanat genişliğine göre tanımlanır. Bu nedenle, boş ağırlık ağırlık merkezi Ortalama Aerodinamik Akor (MAC) yüzdesi olarak ifade edilebilir ve genellikle "temel indeks" olarak adlandırılır. Tartım işleminden sonra uçağa ekipman eklenir veya kaldırılırsa, yenilenmiş boş ağırlık ve EWCG değişiklikle ilişkili moment hesaplanarak ve ardından temel indeks değiştirilerek matematiksel olarak hesaplanabilir.

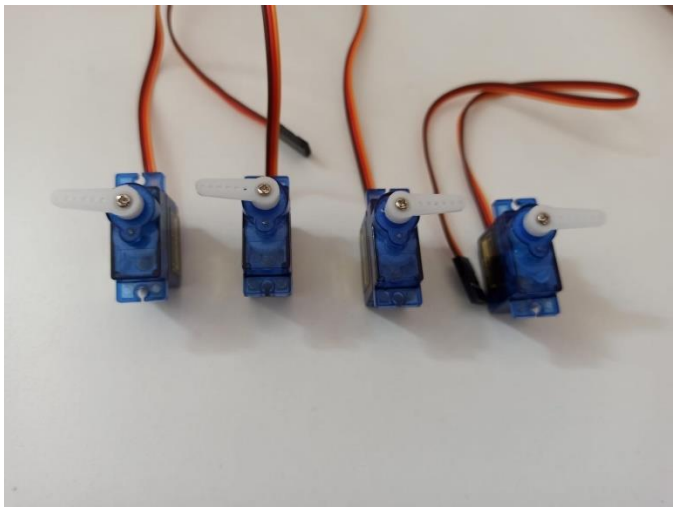
Günlük işlemler için, temel indeks, CG hesaplamaları için bir başlangıç noktası olarak kullanılır. Toplam uçak yükü (mürettebat, yolcu, yük, yakıt, yemek vb.) için ağırlık ve kol bilgileri, momentleri hesaplamak için kullanılır. Yükün toplam momenti, kalkış ağırlık merkeziyle sonuçlanan temel indeksi ayarlamak için kullanılır. Ağırlık merkezi hesaplamaları, operatörün takdirine bağlı olarak matematiksel veya grafiksel olarak yapılabilir.

3. Üret: Model Uçak Yapımı

Bu uygulamada amaç uçuş kontrol yüzeylerinin işleyişini ve uçaktaki konumlamalarını anlamak bir uçağı oluşturan kanat ve gövde bileşenlerinin yapısal kurulumunu kavramaktır. Bu amaçla öğrenciler model uçak setini kullanarak bir sabit kanat uçağı birleştirecektir. Kontrol yüzeylerini yöneten servoları ve bağlantı

kirişlerini bağlayacaktır. Bu uygulamada edinilen bilgi ve beceri ile Teknofest İHA yarışmaları ve dönem sonu etkinliğine bir sabit kanat uçak gövdesi hazırlayabilecektir.
Model uçak gövdesinin birleştirilmesi için üçer kişilik gruplar oluşturulmalıdır. Parçaların montaj sırasına çok dikkat edilmelidir. Yanlış sırada yapılacak montajlar zaman kaybı ve malzemede hasara sebebiyet verecektir.

Uçakta 4 adet mini servo kullanılmaktadır. Bunların 2 tanesi kanatlarda kanatçık kontrolü için, 2 tanesi de gövde içerisinde yatay ve dikey dümen kontrolü için kullanlacaktır.

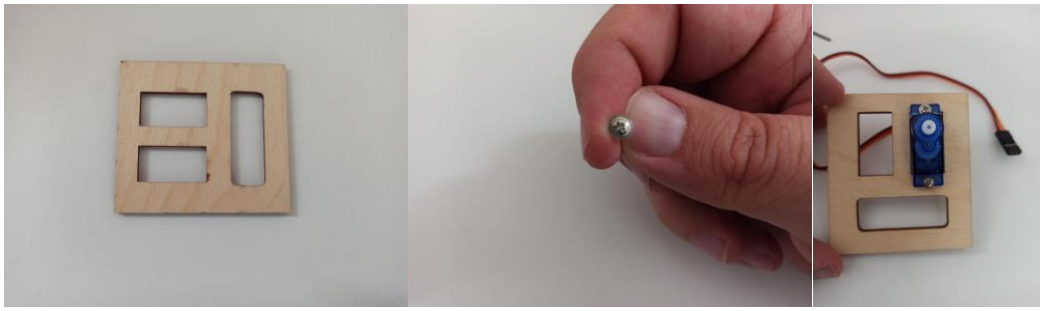


Kontrol yüzeylerini hareket ettirecek bağlantı kirişleri için kısa servo kolları kullanılacaktır. Bu kolların üzerindeki deliklerin çapı metal kiriş çubuklarından küçüktür. Bu nedenle önce delik genişletilerek metal kiriş çubuğunun geçebileceği boyuta getirilmelidir.

Uçağın gövdesinin arka kısmında bulunan yarıklardan uzun metal kiriş çubukları gövde içerindeki yatay ince kanallardan geçecek şekilde geçirilmelidir. Kiriş çubuklarının ucu, kıvrık kısımları uçağın iç tarafında kalacak şekilde konumlandırılmalıdır.

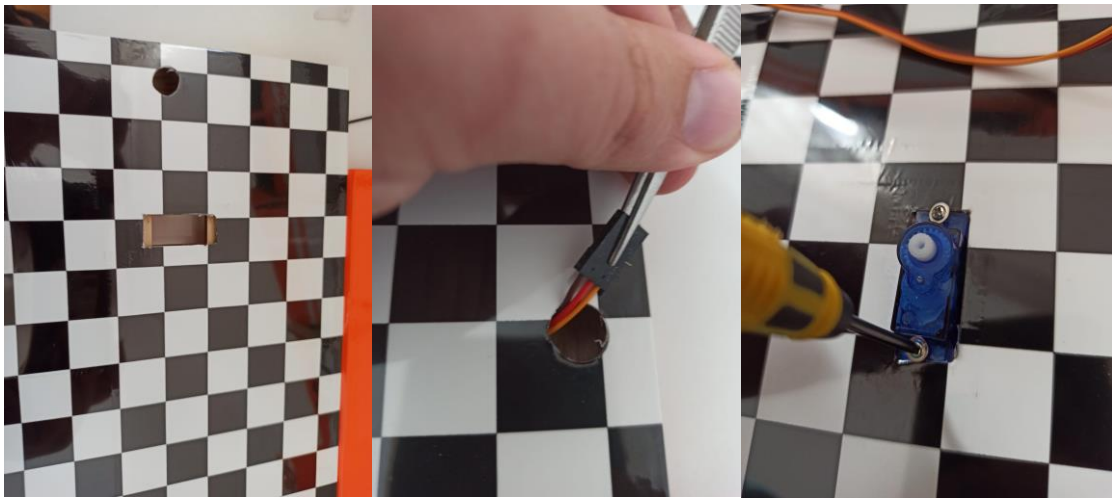
Servo kolları kirişin kıvrık uçlarından geçirilerek hazırlanır.

Yatay ve dikey dümeni yönetecek servolar, kit içerisinde yer alan servo yatağına vida ile monte edilir. Burada servo paketinden çıkan şapka kafalı vidalar kullanılmalıdır.

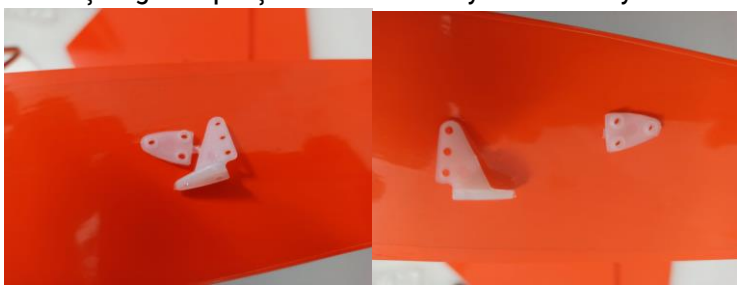


Hazırlanan yatak uçağın gövdesinin içine yerleştirilerek servo kolları, servoların dişlilerine geçecek şekilde yerleştirilir. Burada servonun ileri geri tam 180 derecelik açı hareketi kontrol edilir. Servo kolları tam ortadayken gövdenin boyuna dik şekilde olmalıdır. Servo kolları, servoya vidalanarak sabitlenmelidir.

Benzer işlemi bu sefer kanat altında bulunan servo yuvaları için tekrarlayın. Servonun kablosunu yuvanın içerisinden geçirip bir cımbız yardımıyla kanadın altındaki delikten çıkarın. Ardından servoyu kanada vidalayın.

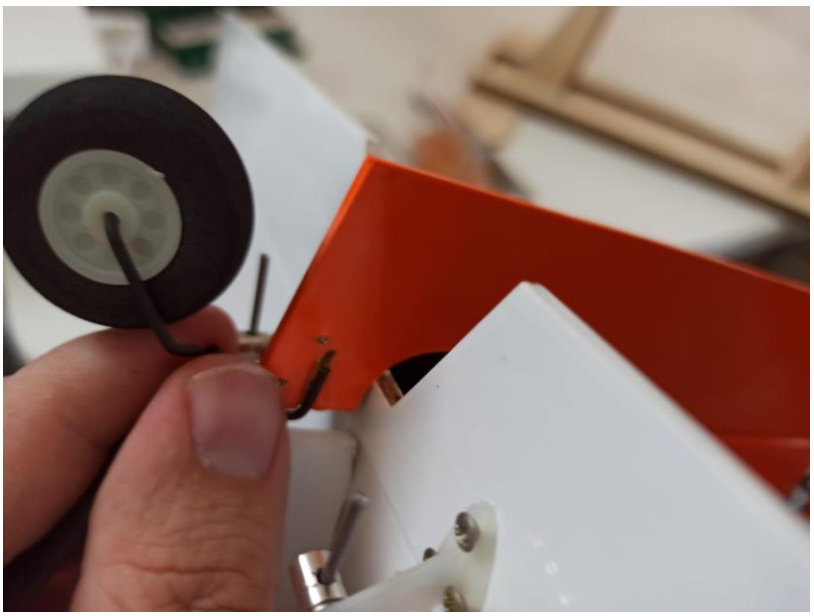


Uçak setinde yer alan kiriş bağlantı parçalarını bir keski yardımı ile ayırın



Dikey dümenin sol alt kenarına dik şekilde vidalayın.

Dikey dümenin sağ yüzünde ince bir yarık olacaktır. Buraya arka iniş takımının metal ucu yerleştirilecektir. Metal uç üzerinden zıt destek parçası vidaya geçirilmelidir.



Aynı uygulama yatay dümen için tekrarlanır. Yatay dümenin alt yüzünde dümene dik şekilde kiriş bağlantısı vidalanır. Dümenin üst yüzünde sabitleyici plastik parça vidaya geçirilerek vida sıkıştırılır.

Ardından plastik dikmelere metal kiriş halkaları vidalanır.

Eğer kit arkadan birleşik şekilde gelmişse bir falçata yardımıyla arka kısımdaki yarık açılır.

Ardından yatay kanat ve dikey sabitleyici yerlerine yerleştirilir. Metal kirişler metal bağlantı vidalarına geçirilerek allen anahtarı yardımı ile sabitlenir.

6

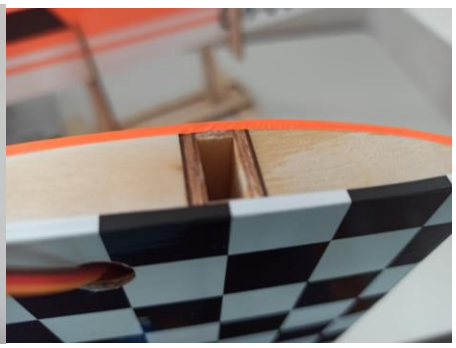
Yatay ve dikey dümen kirişlere sabitlenirken servolar tam orta pozisyonda iken kontrol yüzeylerinin de düz olmasına dikkat edilir. Eğer değilse allen vida gevşetilerek ayar yapılır.

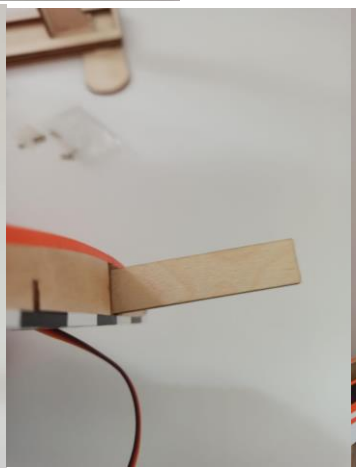
Ayar sonrası arka iniş takımı gövdeye vidalanarak sabitlenir. Böylelikle uçak yerde taksi hareketi yaparken dikey dümen ile yön verilebilecektir.

Yatay ve dikey dümenin hazırlanasının ardından kanat hazırlığına geçilir. Eğitim sırasında grup üyelerinin bir kısmı dümenleri bir kısmı da kanatlarıhazırlayabilir. Kanat üzerindeki kanatçıklara da plastik bağlantı parçaları vidalanır ve kısa kiriş çubukları ile kanat servosuna bağlantıları yapılır.

Burada yine servonun orta konumu ile kanatçıkların düz hizalı olmasına dikkat edilir. Gerekirse allen anahtarı ile metal vida gevşetilerek ayar yapılır.

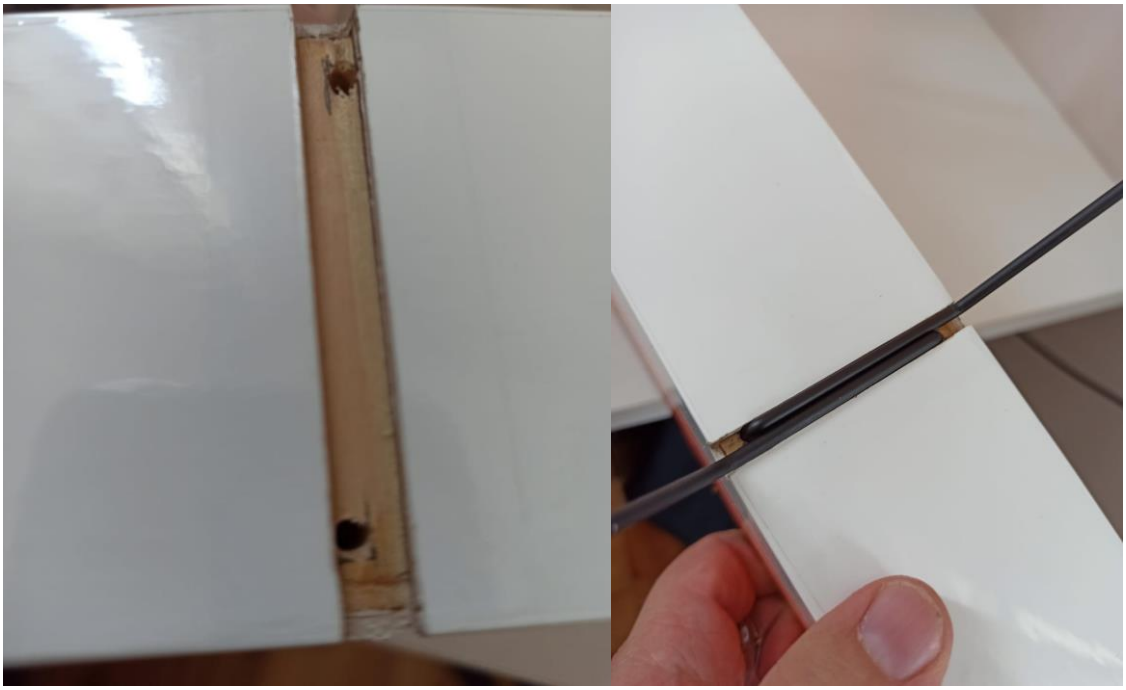
Kontrol yüzeylerinin servo bağlantıları tamamlandıktan sonra sağ ve sol kanat parçaları orta birleştirici kasnak ile bir araya getirilir. Kit içerisinden iki balsa parçasından oluşan kasnak çıkacaktır. Kanatların kenarında bulunan deliklere iki kasnak birleşik şekilde yerleştirilmelidir. Kasnaklardan iki kanat birleştirildiğinde kanat bütünleşik hale gelecektir.




Kanat hazırlandıktan sonra son aşama iniş takımlarının montajıdır. Kit içerisinden çıkan iniş takımı demirlerinin uzun kıvrımlarına tekerlekler geçirilir ve somunları allen anahtarı yardımıyla sıkılır. Uçağın rahat hareketi için tekerlek çok sıkı yapılmamalıdır.





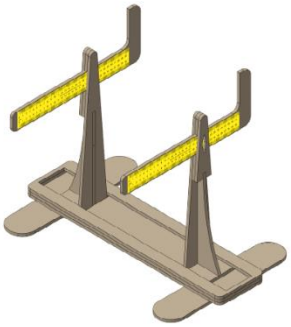
Gövdenin altında yer alan yarıklara ters yönlü olacak şekilde iki metal iniş takımı yerleştirilerek plastik sabitleyicileri vidalanarak iniş takımı monte edilmiş olur.

Uçağı tamamlamak için yapılacak son işlem kanadın gövde üzerine montajıdır. Bunun için gövdenin yanında bulunan deliklere tahta çubuklar geçirilerek lastik tutmakları oluşturulur.

Kanat lasitkler yardımıyla gövdeye sabitlenir. Lastikler çapraz ve dik şekilde yerleştirilir. Böylelikle model uçak İHA şekildeki gibi hazır hale gelmiş olur.

3. Uygula: Ağırlık Merkezi Hesaplaması

Derste yeter sürede uçağın gövdesini tamamlayan takımlar gövdelerinin ağırlık merkezini hesaplayabilir. Bunun için öğrenciler şekildeki gibi denge tahtasına uçaklarını yerleştirmelidir.



Öğrenci kanadın hücum kenarından mesafeyi ayarlayarak uçağın ağırlık merkezini bulur. Uçak bu sırada yere temas etmemeli ve dengede olacak şekilde asılı kalmalıdır. Uçak üzerinde bu ağırlık merkezi bir kalemle işaretlenebilir.

4. Değerlendir

Öğrencilere yaptıkları İHA üzerinde kontrol yüzeyleri sorulur. Aşağıdaki sorularla bilgilerin pekişmesi sağlanır.

1. Kanatçıkları gösterin. Kanatçıklar hangi şekildeyken uçak hangi yöne yatış hareketi yapar?
2. Yatay dümeni gösterin. Yatay dümen hangi pozisyondayken uçağın burnu ne şekilde olur?
3. Dikey dümeni gösterin. Dikey dümenin hareketi neyi sağlatır?
4. Koordineli uçuş için hangi kontrol yüzeyleri kullanılmalıdır?

5. Servo motorların yaptığı görevi büyük uçaklarda hangi parçalar yürütür?
6. Yapılan uçakta ikincil kontrol yüzeyleri var mı? Yoksa hangileri eklenebilir?

5. Ek Faaliyetler

Uçağın kurulumunu erken bitiren gruplar uçağa motor ESC bağlantılarını da yapabilir. Ayrıca 5. Haftada yapılacak uygulamaya hazırlık amaçlı kumanda bağlantısı ve kumanda ile servo kontrolü yapılabilir.

4. HAFTA: İTKİ SİSTEMLERİ

Ön Bilgi:

Öğrencilerin;
İHA sistemleri hakkında temel kavramları bilmeleri beklenir,
Uçuş dinamiğinin temel esaslarını bilmesi beklenir.

Haftanın Kazanımları:

Öğrenciler;
İtkinin temel türlerini tanımlayıp ayırt edebilir.
Uçak motorlarını tanır ve sınıflandırma esaslarını açıklayabilir.
Jet motorlar ile nasıl itki oluşturduğunu anlar ve jet motoru bileşenlerini tanımlar.
Elektrik motoru ve pervane seçimi yapar.
Uçuş için gerekli itkiyi hesaplar.

Haftanın Amacı:

Öğrenciler itki sistemleri hakkında detaylı bilgiye sahip olur. İtki sistemlerinin kullanım gerekçelerini kavrar. Tasarlanan insansız hava aracı için itki sistemi bileşenlerinin (elektrik motoru, pervane, ESC ve batarya) nasıl seçileceğini öğrenir.

Kullanılacak Malzemeler:

Çeşitli Fırçasız DC Motorlar

Çeşitli Ölçülerde Pervaneler

Terazi

ESC

LiPo Batarya

Çeşitli Ölçülerde Kablo ve Bağlantı Elemanları

Haftanın İşlenişi:

Gözle: İtkinin tanımı ve çeşitlerinin öğrenilmesi. İtki sistemi hesabı ve tasarımı yapabilecek teorik bilgiye sahip olma.

Uygula: Örnek bir itki hesabı yapabilme.

Tasarla: Motor-pervane sisteminin ürettiği itkiyi ölçecek deney düzeneği tasarlama. Deney düzeneğini oluşturan bileşenlerin özelliklerini kavrama.

Üret: İtki kuvveti ölçen sistemi kullanarak farklı pervane ve motor çiftlerinin ürettiği itki kuvvetinin ölçülmesi.

Değerlendir: Farklı motor-pervane çiftlerinin ürettiği itki kuvvetleri arasındaki farklar değerlendirilerek bunun sebepleri tartışılır.

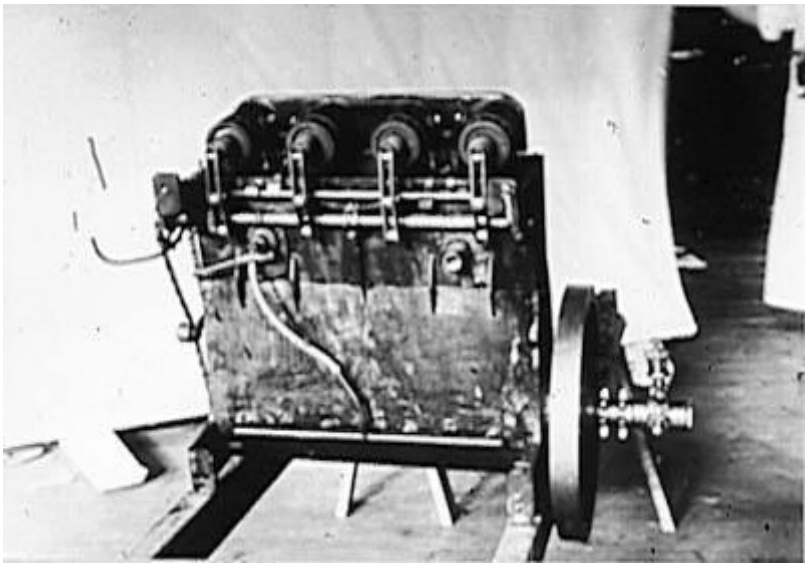
1. GÖZLE VE UYGULA

1.1. Gözle: İtkinin Tanımı ve İtki Üreten Motorlar

İtki kelimesi bir cismi öne doğru ittirmek veya ileriye sürmek anlamına gelmektedir. İtki kuvveti ise (İng. thrust), bir nesneyi ileri iten kuvvet olarak tanımlanmaktadır. Hava araçlarında itki kuvveti daha önceki bölümde de belirtildiği üzere yatay uçuşta araca etki eden sürükleme kuvvetine zıt yönde olacak şekilde üretilen kuvvettir.

Uçaklarda itki kuvveti motorlar tarafından sağlanmaktadır. İlk başarılı uçuşun yapıldığı 1903 yılından bu yana uçaklarda genellikle içten yanmalı motorlar kullanılmaktadır. Uçaklarda kullanılan motorlar prensip olarak iki gruba ayrılır: Pistonlu ve gaz türbinli motorlar. Gaz türbinli motorlar da kendi arasında turboprop, turbofan, turboşaft ve turbojet olarak sınıflandırılmaktadır.

Isı enerjisini mekanik enerjiye çeviren pistonlu motorlar uçakta ilk defa Wright kardeşler tarafından kullanılmıştır. Isı enerjisi kapalı bir silindir içerisinde üretildiği için bu tür motorlar içten yanmalı motorlara örnektir. Wright kardeşler itki kuvvetini üretebilmek amacıyla yüksek hızda dönen pervanelerin gerekliliğini fark etmişlerdi. Pervanelerin gerekli hızda dönebilmesinin ise ancak bir motor yardımıyla yapılabileceğini anlamaları uzun sürmedi. O zamanlar üretilen motorlar arabalar için tasarlandığından kendilerinin kullanabileceği bir tasarım bulamamışlardı. “Uçan makinede” kullanacakları aşağıda resmi bulunan ilk motor Charles Taylor tarafında Wright Kardeşler için özel olarak imal edilmiştir.



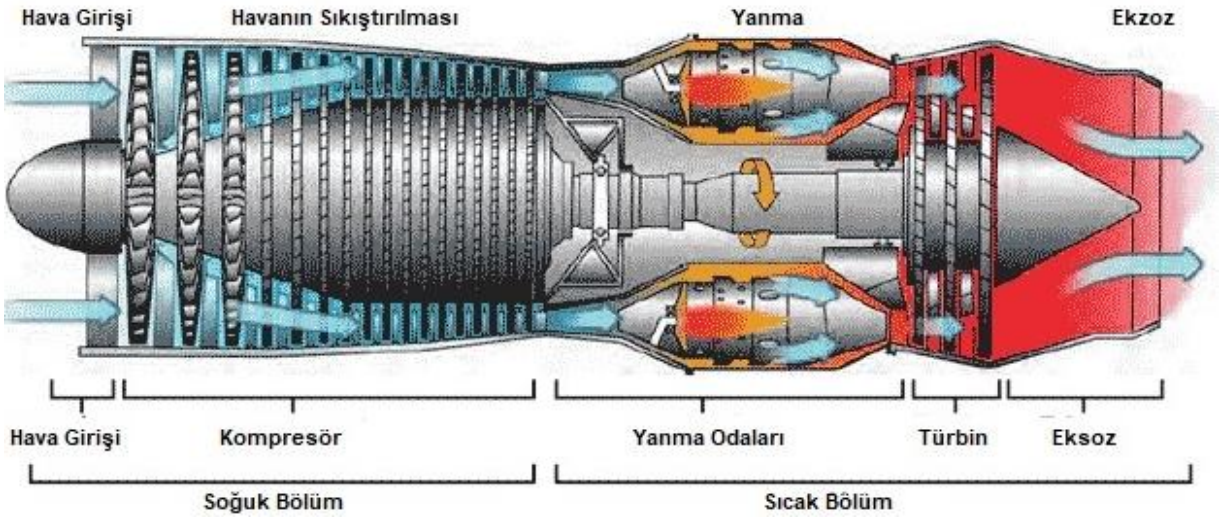
Pistonlu uçak motorlarının silindirleri düz bir hat üzerinde (sıralı tip), V tipi veya silindirleri bir daire (raydal veya yıldız tip) oluşturacak şekilde üretilmektedir. Radyal motorların sıralı motorlara göre iki önemli avantajı bulunmaktadır. Birinci avantaj, radyal motorlarda silindirler motorun ön kısmında yer alması, ikincisi ise hava akımının doğrudan alınması ile motorda verimli bir soğutmanın gerçekleştirilebilmesidir. Aşağıda hava soğutmalı radyal uçak motorunun resmi verilmiştir.

Turbo jet motor tüm gaz türbinli motorların atası olarak tanımlanabilir. Pistonlu motorlarda emme, sıkıştırma, yanma ve egzoz işlemleri silindir içerisinde gerçekleşirken turbo jet motorlarda bu dört işlemin her biri aynı anda motorun farklı kısımlarında gerçekleşmektedir. Bu açından da gaz türbinlerinde bir proses sürekliliğinden söz edilebilmektedir. Bir turbo jet motoru hava alığı, kompresör, yanma odası, türbin ve egzoz kısımlarından oluşmaktadır. Kompresöre kadar hava neredeyse ortam sıcaklığındayken bu kısımdan sonra motor içerisindeki havanın sıcaklığında önemli artışlar olmaktadır.

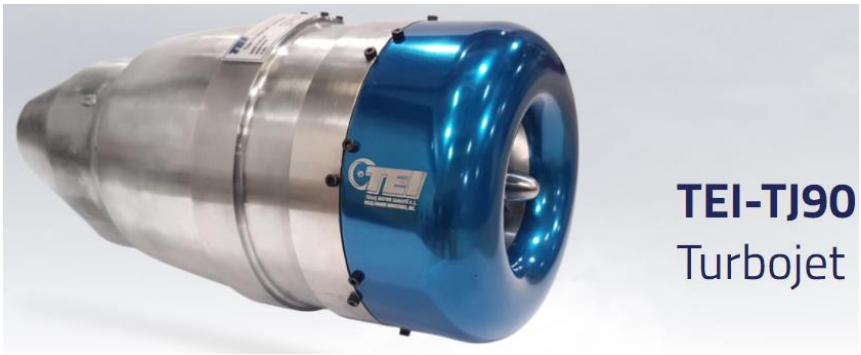
1.1.1. Gaz Türbinli Motorların Çalışma Prensibi:

Turbo, Latince "yüksek hızda dönen", jet ise "fırlatmak" anlamına gelmektedir. Newton'un üçüncü yasasına göre bir cisme etki eden kuvvete karşılık cisim tarafından bu kuvvete aynı şiddette ve zıt yönde bir tepki kuvveti uygulanmaktadır. Turbojet motorlar da bu prensibe istinaden itki kuvveti oluşturmaktadır. Turbojet motorlar gazları yüksek hızda geriye doğru itmekte ve ters yönde tepki kuvveti oluşturmaktadır. Atmosferden hava alığına alınan hava kompresör kısmına girerek belirli bir oranda sıkıştırılmaktadır. Sıkıştırılmış hava buradan yanma odasına girmekte ve burada püskürtülen yakıtla yanması sağlanmaktadır. Yanan ve dolayısıyla ısınan yakıt-hava karışımının oluşturduğu gazlar türbin kısmına gelerek burada genleşerek iş üretmektedir. Türbin ve kompresör birbirine bir şaft aracılığı ile mekanik olarak bağlanarak türbinin ürettiği enerji kompresöre aktarılarak döngü sağlanmaktadır. Son olarak türbinden egzoz

kısmına gelen karışım buradan hızla dışarı atılarak itki elde edilmektedir. Jet motorları pistonlu motorlara nazaran çok yüksek devirlerde çalışarak yüksek güç üretmektedir. Ayrıca yüksek sıcaklıklara ulaşan hava dolayısıyla jet motorlarında yüksek teknoloji gerektiren malzemeler kullanılmaktadır.



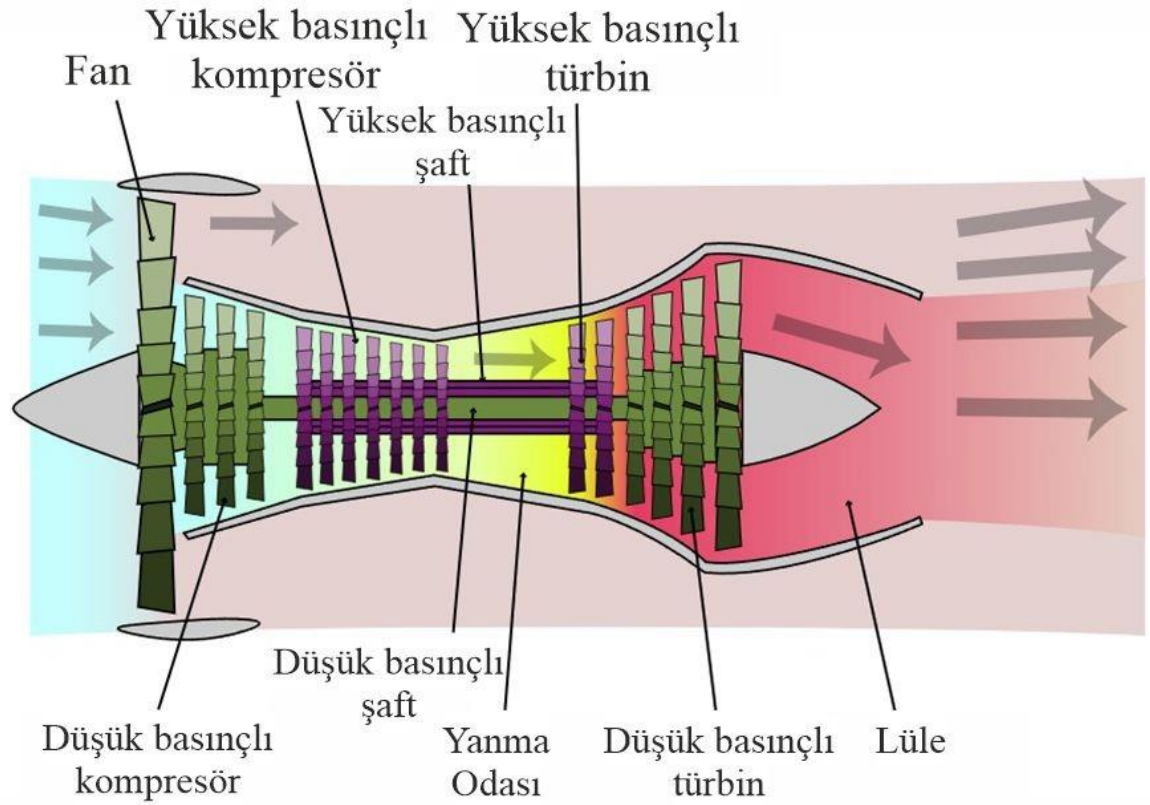
Ülkemizde yerli imkânlarla Türk Havacılık ve Uzay Sanayii (TUSAŞ) Motor Fabrikası'nda üretilen TEI TJ90 Turbojet motorun resmi aşağıda gösterilmiştir.



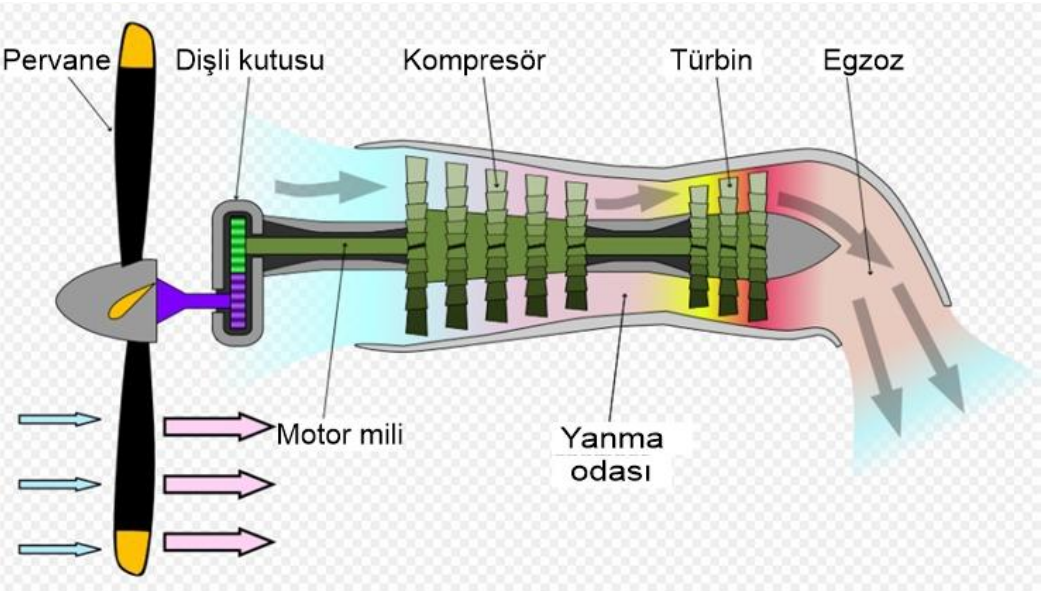
Turbofan motorlar turbojet motorun ön kısmına fan eklenmesiyle oluşmaktadır. Hava alığına giren hava ilk önce fana girmektedir. Fana giren havanın bir kısmı çekirdek olarak adlandırılan kompresör kısmına (turbojet), geri kalan hava ise çekirdek kısmın çevresinden egzoz kısmına by-pass havası olarak gitmektedir. Turbofan motorlarda farklı olarak türbin düşük basınçlı ve yüksek basınçlı olmak üzere iki kademedir. Yüksek basınçlı türbin kompresör ile düşük basınçlı türbin ise fan kısmı ile mekanik olarak bağlantılıdır. Günümüzde birçok yolcu uçağında turbofan motorlar kullanılmaktadır.

https://www.youtube.com/watch?v=bc8h5A-T23g

https://youtu.be/KjYw0GdRpm0



Turboprop motorlar temel olarak turbojet motorların ön kısmına pervane eklenmesiyle oluşturulmaktadır. Turboprop motorlarda oluşan toplam itkinin yaklaşık olarak %80'i pervaneden geri kalan %20 ise jet motorun itmesinden elde edilebilmektedir. Türbin tarafından oluşturulan dönüş hareketinin devrinin yüksek olmasından dolayı pervaneden önce bir dişli kutusu kullanılarak devir düşürülmektedir. Turboşaft tipi motorlarda ise turbojet motor kısmından elde edilen mekanik enerji turboprop motorlara benzer olarak helikopter palelerine aktarılmaktadır.



1.1.2. Hibrit İtki Sistemleri:

Şimdiye kadar anlatılan motor çeşitlerinde yakıt olarak fosil yakıtlar kullanılmaktadır. Fosil yakıtların çevreye olan olumsuz etkileri ve rezervlerinin azalması nedenleriyle alternatif itki sistemleri üzerinde uzun yıllardır çalışmalar gerçekleştirilmektedir. Bu kapsamda elektrikli itki sistemleri akla gelen ilk alternatiftir. İtki oluşturması için yüksek verimli elektrik motorları üretilmesine karşın kullanılan bataryaların kapasitesi ve kapasitedeki artışa bağlı olarak artmak zorunda olan batarya grubu ağırlığı sınırlandırıcı bir etken olarak karşımıza çıkmaktadır. Fosil yakıtların enerji yoğunluğu kilogram yakıt başına 10kWh mertebelerinde iken bataryalarda bu değer 0.2kWh civarındadır. Fosil yakıtlarının yüksek güç yoğunluğundan ve elektrikli itki sistemlerinin yüksek verimliliğinden faydalanmak amacıyla hibrit itki sistemleri geliştirilmiştir. Hibrit itki sistemleri çoğunlukla kara taşıtlarında kullanılmasına karşın hava araçlarında da kullanılmaya başlanmıştır. Hibrit itki sistemlerinin çevreci, daha verimli ve maliyet etkin olması dolayısıyla yakın gelecekte kullanımının daha da artacağı düşünülmektedir.

Hibrit İtki Sistemi Örneği: Elektrik Motoru ve Fosil Yakıtlı Motor

Hibrit itki sistemleri özellikle insansız hava araçlarında (İHA) yaygın kullanım alanı bulabilmektedir. İnsansız hava araçlarının hafif olması ve yüksek verime sahip motorlarla uçuş süresinin artırılması hibrit itki sistemlerinin bu tarz araçlarda kullanımının önünü açmıştır. Özellikle VTOL insansız hava araçlarında hibrit itki sistemleri kullanılarak hem sabit kanatlı uçakların avantajlarından hem de fosil yakıt enerji yoğunluğundan faydalanmak suretiyle uçuş süresi artırılmaktadır. Bunun yanında ticari uygulamalarda kullanım alanı gittikçe artan döner kanatlı araçların havada kalış süresini arttırmak için yine hibrit itki sistemleri kullanılmaktadır. Kullanılan hibrit itki sistemi sayesinde gidilen menzil ve taşınan paralı yük ağırlığında önemli artışlar gözlenmiştir.

Hibrit İtki Sistemli Bir Döner Kanat

1.1.3. Elektrik Motorları ve Bileşenleri

İHA'larda kullanılan elektrikli motorları, Elektronik Hız Kontrol Devresi (ESC), batarya ve pervanelerle birlikte, bir İHA'nın tahrik sistemi içindeki temel bileşenlerdir. ESC ve pervaneler ile ilgili bilgiler ilerleyen kısımlarda verilecektir. İHA'larda kullanılacak motorların seçimi hem verim hem de güvenlik açısından büyük önem arz etmektedir.

İnsansız hava araçlarında (İHA) kullanılan elektrik motorlarının en genel görevi, bu araçların uçma kabiliyetini kazanması için ihtiyacı olan itkiyi pervaneler yardımı ile kazandırmaktır. Ayrıca İHA'lar üstleneceği göreve ve kabiliyetlere göre ek motorlar kullanılabilmektedir. İnsansız hava araçları için itki sağlayan elektrik motorlarının kullanımı üç avantajı beraberinde getirmektedir: Yüksek verim, düşük ağırlık, stabil çalışma ve montaj kolaylığı. Genel olarak yüksek standartlardaki elektrik motorları %90'ın üzerinde verim değerine sahiptir. Düşük titreşimde çalışmalarının yanı sıra montaj-demontaj süresi çok kısadır. Motorların ağırlıklarının düşük olması ağırlık etkin tasarımlar için çok önemli bir faktördür.

Döner Kanat ve Sabit Kanat İHA'lar İçin Motor Montajı

1.1.4. Elektrik Motorunun Temel Bileşenleri

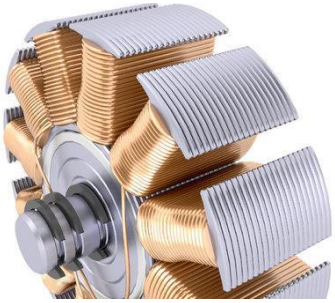
Hem sabit kanat hem döner kanat İHA'lar için genellikle doğru akım (DC) ile çalışan motorlar tercih edilir. Döner ve sabit kanatlı hava araçları için kullanılan doğru akım elektrik motorları, elektrik enerjisini doğrusal veya en yaygın olarak döner hareket şeklinde mekanik enerjiye dönüştüren sistemlerdir. Motorun çalışması, içerisinde bulunan sargılara elektrik akımı uygulandığında yine motorun içerisinde bulunan sabit mıknatıslara zıt yönde oluşan manyetik kuvvetin etkisiyle hareket etme prensibine dayanır. Bu akımın yönünün sürekli olarak sabit mıknatısa ters manyetik alan oluşturacak şekilde değiştirilmesi gerekmektedir. Bu değişim fırçalı motorlarda motorun sarımlarına temas eden fırçalarla, fırçasız motorlarda ise elektronik hız kontrol devresi tarafından sağlanmaktadır. İHA uygulamalarında genellikle fırçasız doğru akım motorları (BLDC) tercih edilmektedir. Fırçasız bir doğru akım motorundan yüksek verim alabilmek ve hafiflik bu tercihteki temel etmenlerdir. Ayrıca, fırçasız yapıları nedeniyle sürtünme olmaması, az ısınması, sessiz çalışmaları, bakıma ihtiyaç duymayacak bir yapıda olmaları, yüksek devirlerde problemsiz çalışmaları, uzun ömürlü olmaları ve boyutlarının küçük olmasına karşın ürettikleri momentin büyük olması gibi avantajlara da sahiptir. Ancak bu motorların maliyeti nispeten yüksektir. Ayrıca, karmaşık bir kontrol devresine sahip olması ve konum sensörüne ihtiyaç duyması fırçasız motorların dezavantajları olarak karşımıza çıkmaktadır. Bu tip motorlar temel olarak 5 parçaya ayrılmaktadır: Stator, iletken tel, rotor, mıknatıs, gövde ve dönme elemanları.

Bir Elektrik Motorunun İç Yapısı

Fırçasız doğru akım motorlarının hareketi, sabit iletken tel sarılı stator etrafında kutupları belirli bir örüntüye göre rotor üzerine sıralanmış mıknatısların dönmesi şeklinde olmaktadır. Mıknatıslar genellikle bir N ; bir S kutupları statora bakacak şekilde dizilmektedir. Bu mıknatısların itme ve çekme özellikleri (manyetik özellikleri) kullanılarak belirli bir örüntüyle karşısındaki konumda bulunan elektromıknatıs görevi yapan bobinleri elektrik enerjisi ile yüklenmekte ve motor dönme hareketi yapmaktadır. Bu hareket bazen göbekten bağlı bir şaft ile bazen de direkt rotora bağlı pervaneler ile itkiye dönüştürülmektedir.

Stator, motor için verimini etkileyen en önemli parçalardan birisidir. Fırçasız doğru akım motorlarında dönmeyen kısımdadır. Statorun konumuna göre içte ve dışta bulunan dişler

içermektedir. Bu dişler üzerinde iletken tel ile sarım yapılarak manyetik alan oluşturulur. İçeriği genellikle birden çok özel silisyumlu çelik sacların birleştirilmesi ile ham halini almaktadır. Bu işlem laminasyon olarak adlandırılmaktadır. Statorun bir diğer temel içeriği iletken teldir. İletken tel olarak genellikle bakır tel kullanılmaktadır. Üretilecek elektrik motorunun gücü de dikkate alınarak kullanılacak bakır telin kalınlığı, diş etrafındaki tur sayısı, paralel sayısı motorun güç, tork, devir, verim gibi temel parametreler belirlenmektedir.



Rotor, motorun dönen kısmıdır ve mıknatısları barındırır. Mıknatısların temel koruyucu elemanıdır. Malzeme olarak verimi artırmak için çelik malzeme kullanılmaktadır. Çok özel durumlarda statorda olduğu gibi silisyumlu sacların üst üste dizilip lamine edilmesiyle de oluşturulabilir.

Bunun haricinde motor içerisinde dönme hareketinde sürtünmeyi en aza indirmek için rulman, dönme hareketini sağlamak için şaft ve ön kapak bulunabilir.

İHA'ların motorunun seçimi, özellikle İHA'nın ağırlığı olmak üzere birçok faktöre bağlıdır. Bir İHA motorunun veya motor grubunun, İHA'nın ağırlığına karşı koyması için yeterli itme gücü üretebilmesi ve dengeli şekilde havalanmasını sağlayabilmesi gerekir. Bir İHA motorunun torku, bir hızdan diğerine geçme yeteneğini temsil eder. Daha büyük çaplardaki pervaneler için daha yüksek tork değerli motorlar gereklidir.

Enerji kaynağı olarak doğru akıma (DC) bağlı enerjinin kullanılacağı batarya ve enerji kaynakları için DC motorlar kullanılır. Piyasada en sık kullanılan elektrikli İHA motor tipleri sabit mıknatıslı motor (permanent magnet) ve drone servo motorlarıdır.

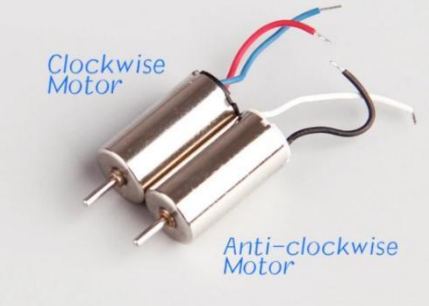
Sabit Mıknatıs Motorlar (Permenant Magnet)

Fırçasız DC motorlara benzer şekilde, sabit mıknatıslı motorlar güvenilirlik, verimlilik ve daha soğuk çalışma sıcaklıkları ile bilinir. Ayrıca diğer İHA motor türlerinden malzeme kalitesine de bağlı olarak daha uzun ömürlü olabilirler. En yaygın sabit mıknatıslı motorlar, sabit bir manyetik alan oluşturmak için neodimyum mıknatıslar kullanır. Ancak neodimyum pahalı ve nadir bir mıknatıs tipi olduğu için sabit mıknatıs motorlar nispeten pahalıdır.

Sabit Mıknatıs DC Motor

Drone Servo Motorları

Drone servo motorları, basit fırçalanmış DC motorlardan karmaşık endüstriyel AC endüksiyon motorlarına kadar çeşitli motor türlerini kullanabilir. Servo motorlar, diğer adıyla servolar, motora motor milinin gerekli konumunu gösteren analog veya dijital elektronik sinyaller kullanılarak hassas bir şekilde kontrol edilebilir. Geri besleme devresi ve bir konum kodlayıcı kullanılarak yüksek derecede hassas konumlandırma elde edilir.



Drone Servo DC Motoru Örneği

1.2. Gözle: Elektrik Motorunun Seçimi

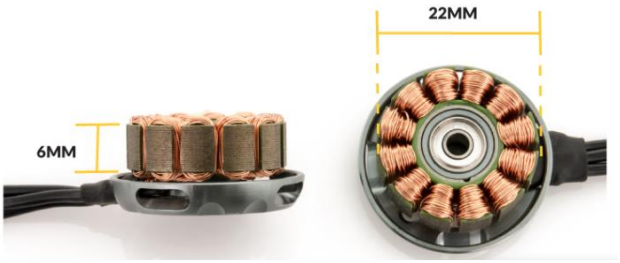
Elekrik motorunun seçiminde dikkat edilmesi gereken en önemli kriter itki-ağırlık oranıdır. Burada ifade edilen değerler İHA'nın dış ortamda uçuş yapacağı düşünülerek verilmiştir. Döner kanat bir İHA yarı gazda yani %50 itki oranında (kumandanın gaz düğmesi sıfır seviyesinden yarı seviyeye getirildiğinde) yerden kalkıp havada asılı kalabilecek şekilde (hover) tasarlanmalıdır. Dolayısıyla, araca tam gaz verdiğimizde üretilen itki miktarının uçak ağırlığının iki katı kadar olması beklenmektedir. Örneğin 4 kg ağırlıkta bir döner kanadın itki sisteminin sağlaması beklenen en yüksek itki kuvveti 8 kg olmalıdır. Dört kanadı olan bir döner kanatta bu durumda her motorda 2 kg itki üretilecek şekilde motor ve pervane seçimi yapılmalıdır. İHA'nın şiddetli rüzgârlı hava koşullarında kararlı bir şekilde uçabilip hızlı manevralar yapabilmesi için itki-ağırlık oranının 2-3 kat arasında seçilmesi önerilmektedir. Sabit kanatlı bir İHA için motor seçimi yapılırken de itki-ağırlık oranının 0.8-1 aralığında seçilmesi önerilmektedir.

Motor seçimi yapılırken üretici firmaların motorları için sunduğu test raporlarından faydalanılır. Genellikle üreticilerin motorla ilgili sunmuş olduğu veriler gaz düğmesinin tam açık olduğu durumdaki değerler olup farklı gaz oranlarındaki motor performansını içeren test raporlarının bulunduğu tabloların kullanımı tasarım açısından daha faydalı olmaktadır. Örnek bir tablo aşağıda verilmiştir.

Propeller	Throttle	Voltage (V)	Current (A)	Input power (W)	RPM	Torque (N*m)	Thrust (g)	Efficiency (g/W)
T-motor P14*4.8CF	40%	48.5	2.9	140.65	5011	0.17	975	6.93
	42%	48.49	3.12	151.19	5182	0.18	1026	6.79
	44%	48.5	3.36	163.01	5337	0.19	1084	6.65
	46%	48.49	3.62	175.73	5508	0.2	1142	6.50
	48%	48.49	3.9	189.06	5667	0.21	1215	6.42
	50%	48.53	4.2	203.63	5834	0.23	1282	6.30
	52%	48.53	4.59	222.80	6035	0.25	1386	6.22
	54%	48.53	4.97	241.34	6213	0.26	1487	6.16
	56%	48.52	5.33	258.66	6383	0.27	1567	6.06
	58%	48.53	5.65	274.29	6413	0.27	1646	6.00
	60%	48.52	5.99	290.44	6655	0.3	1730	5.96
	62%	48.52	6.35	308.30	6829	0.31	1809	5.87
	64%	48.52	6.79	329.45	6993	0.32	1905	5.78
	66%	48.52	7.25	351.77	7161	0.34	1980	5.63
	68%	48.56	7.78	377.65	7340	0.35	2103	5.57
	70%	48.55	8.35	405.25	7534	0.37	2204	5.44
	75%	48.55	9.72	472.05	7947	0.42	2481	5.26
	80%	48.54	11.35	550.83	8376	0.47	2787	5.06
	90%	48.53	14.74	715.24	9161	0.57	3314	4.63
	100%	48.52	18.52	898.64	9870	0.68	3891	4.33

Burada dikkat edilmesi gereken husus verilen motor değerleri ilk sütunda belirtilen pervane boyutu için ölçülen motor performans verileri olduğudur. Motor üreticileri farklı pervane çapları için farklı tablolar sunmaktadır. Bir diğer önemli husus ise bu tabloda elde edilen değerler üretici firmanın testlerde kullandığı pervane için olduğudur. Belirtilen çapta farklı bir pervane kullanılması durumunda belirtilen değerlerde farklılık olabilecektir. İHA'nın üretimi aşamasında kullanılan motor-pervane çiftinin itki performansının ölçülmesi önerilmektedir. Yukarıdaki tabloda 14 inçlik bir pervane kullanılması durumunda motor performansı 48 V gerilimde farklı gaz oranları için ölçülmüştür. Motorun çektiği gerilim, akım, giriş gücü (gerilim x akım), motor devri, üretilen tork, itki kuvveti ve verim (itki kuvveti/giriş gücü) değerleri verilmiştir. Gerilim değeri değiştirildiğinde (seri bağlı hücre sayısının arttırılması veya azaltılması) ölçülen değerlerin değişeceği unutulmamalıdır. Motor seçiminde verim arttıkça uçma süresinin artacağı, düşük gerilimlerde verimin artacağı ancak itki kuvvetinin azalacağı, gerilim yükseldikçe pil ağırlığının artacağı dolayısıyla kalkış ağırlığının artacağı göz önünde bulundurulmalıdır. İHA'nın görev amacına göre itki ile verim arasında bir tercih yapılması gerekebilecektir.

Motor boyutları elektrik motorunun seçiminde kullanılan önemli parametrelerdendir. Motorların boyutlarını ifade etmek için dört haneli sayı kullanılmaktadır. Burada ilk iki hane motor statorunun kalınlığının milimetre cinsinden değerini, son iki hane ise stator yüksekliğinin milimetre cinsinden değerini ifade etmektedir. Stator yüksekliği arttıkça yüksek devirde güç artmaktadır. Buna karşın düşük devirlerde motorun performansı (verimi) düşmektedir. Stator genişliği arttıkça da düşük devirde elektrik motorunun ürettiği tork ve verim artmaktadır. Görev ihtiyacına göre motor boyutlarının seçilmesi gerekmektedir. Örneğin ağır bir nesne taşınacak ise genelde düşük devirde yüksek torklu motorlar, yüksek manevra kabiliyetine sahip bir İHA ihtiyacı varsa bu durumda da yüksek devirde çalışacak motorlar tercih edilmesi gerekmektedir.



2206 Seri Numarasına Sahip BLDC Motor

KV Değeri

KV değeri fırçasız elektrik motorlarında tanımlı bir parametre olup motorun yüksüz halde dakikadaki devir sayısının (RPM) çektiği gerilime oranı olarak ifade edilmektedir. Başka bir ifadeyle, fırçasız bir motorun bir volt ile bir dakikada meydana getirdiği devir KV değeri olarak adlandırılmaktadır. 300 KV değerine sahip bir motora yüksüz durumda 12 volt gerilim uygulandığında 300x12=3600 devir/dakika (RPM) hızla dönecektir. Gerilim değeri iki katına çıkartıldığında (24 V) motor 7200 RPM ile dönecektir. Dolayısıyla yüksek KV'lı motorlar pervaneyi hızlı döndürecek olup genellikle düşük çapa sahip pervaneler kullanılmaktadır. Düşük KV'lı motorlar da pervaneyi düşük hızda döndürmesine karşın yüksek tork kapasitesine sahip olacakları için büyük çapa sahip pervanelerin kullanımı tercih edilmektedir. İmalatçı firmanın vermiş olduğu KV değerinin motorun yüksüz durumda (pervane takılmamış) geçerli olduğu pervaneler takıldığı durumda sürtünme direncinden dolayı beklenen devir değerlerine ulaşılamayacağı hiçbir zaman unutulmamalıdır.

1.3. Gözle: Pervaneler

Pervaneler itki sisteminin en önemli parçalarından olup genel olarak bir kanadın özelliklerini taşırlar. Pervanelerin kanatlar gibi veter uzunluğu, hücum açısı, hücum kenarı ve firar kenarı bulunmaktadır. Ancak pervaneler kanatların aksine dikey yönde bir taşımadan ziyade ileri doğru bir itki oluşturmak üzere tasarlanmışlardır. Pervane, içten yanmalı bir motordan veya türbinli bir motordan aldığı mekanik enerjiyi, önündeki hava kütlesini uçağın hareket yönüne zıt yönde hızlandırmak suretiyle itme veya çekme kuvvetine dönüştüren sistemlerdir. Aşağıda bir pervane kesitine etki eden kuvvetler ve kesitteki hücum açısı gösterilmektedir.

Bir Pervane Kesitine Eki Een Kuvvetler

Pervaneler plastik, karbon fiber ve ahşap gibi malzemelerden imal edilmektedir. Bir pervane, göbeği etrafında eşit açısal aralıklarla konumlandırılmış iki veya daha fazla kanatçıktan (pal) oluşmaktadır. İnsansız hava araçlarında genellikle iki palli pervaneler kullanılmaktadır. Sabit kesitli bir pervane döndüğünde, ucuna yakın kısımlarda çizgisel hız göbeğe yakın kısımlarındaki hızdan daha büyük değere sahip olmaktadır. Böylece pervanenin üzerinden geçen hava akışı uç kısımda en yüksek hıza ulaşmaktadır. Sabit kesitli pervanelerin yüksek devirlerde kullanıldığında bu durum uç kısımlarda önemli sorunlara neden olmaktadır. Kanat boyunca çok farklı itki ve basınç dağılımları oluşur. Bu durum bazı kesitlerde negatif hücum açılarına bazı kesitlerde de akışın yüzeyde tutunamamasına yani taşımanın azalıp sürüklenmenin ani olarak artmasına neden olabilmektedir. Dolayısıyla, palin tamamının eşit miktarda hava çekmesi veya düzgün dağılmış bir itme gücü sağlayabilmesi için pal uçtan pervane göbeğine doğru gittikçe artan bir açı (burulma) verilmek suretiyle tasarlanması ve uca doğru incelmesi gerekmektedir.

Bir pervanenin oluşturduğu itki kuvveti aşağıdaki formül ile hesaplanmaktadır:

$$T = C_T \rho n^2 D^4$$

Burada C_T pervane itki sabiti olup imalatçı firmanın sağladığı boyutsuz bir sayıdır. ρ havanın yoğunluğu (kg/m^3), n saniyedeki pervane devri (devir/saniye) yani bir saniyede döndüğü devir sayısını ve D pervane çapını (m) ifade etmektedir. İHA'nın uçurulacağı yerin deniz seviyesinden yüksekliğinin bilinmesi yoğunluk değerinin doğru olarak kullanılması bakımından önemlidir.

Tasarım için pervanelerimizi belirlerken pervane seçiminde motor kalkış ağırlığının ve motor özelliklerinin bilinmesi gerekmektedir. Yukarıdaki itki formülünden de görüldüğü gibi pervane seçimindeki en önemli parametre pervane çapıdır. Pervane çapının iki kat artması itki kuvvetini 16 kat arttırmaktadır. Çap artışına bağlı olarak pervane salınımının artacağı ve oluşabilecek şok dalgalarının zararları göz ardı edilmemelidir. Pervanelerin yapıldığı malzemelerden daha önce bahsedilmişti. Ağırlık ve mukavemet açısından karbon fiber pervaneler avantaj sağlamakla birlikte bir çarpma anında vermiş olduğu hasarın daha fazla olacağı unutulmamalıdır. Ayrıca karbon fiber pervaneler plastik pervanelere oranla çok daha pahalıdır. Pervanelerin belirlenmesinde dikkat edilmesi gereken diğer önemli bir parametre de optimum devir aralığıdır. İHA'nın uçuşu için gerekli motor devri ile pervanenin optimum devrinin uyumlu olması gerekmektedir. Diğer bir parametre ise pervanenin sağlayabileceği en büyük itki kuvveti değeridir. Tasarım aşamasında her bir motor-pervane çiftinin üretmesi gereken itki değerinin pervane tasarımcısının verdiği en büyük itki kuvveti değerinin altında olması gerekmektedir.

Pervane satın alırken pervanelerin fiziksel boyutları ile alakalı olarak dört haneli bir sayı ile karşılaşılır. Örneğin "1245" ifadesinde ilk iki rakam pervanenin çapının inç biriminden değeridir. Bu örnekte pervane çapı 12 inç yani 30.48 cm'dir. Son iki rakam ise pervanenin bir tur döndüğünde İHA'nın aldığı yolu (geometrik hatveyi) göstermektedir. Örnekteki değer alınan yolun 4.5 inç yani 11.43 cm olduğunu ifade etmektedir.

1.4. Gözle: ESC (Electronic Speed Control – Elektronik Hız Kontrolü) Devresi

Elektronik hız kontrolü devreleri DC (direct current-doğru akım) motorlarının hız ve yönlerini ayarlayan ve kontrol eden elektronik devrelerdir. ESC devreleri döner kanat ve benzeri İHA prototiplerinde sıklıkla kullanılmaktadır. Genellikle motor ve uçuş kontrolcüsü arasına yerleştirilmekte ve enerji dağılım kartı (power distribution board – PDB) ile beslenmektedir. Eğer devrede PDB yok ise doğrudan batarya ile de beslenebilmektedir.

ESC Seçimi

ESC devreleri İHA prototipi hazırlamak üzere seçilirken, kullanılacak olan motorun tipi ve özelliklerine göre seçilir.

D**ikkat Edilmesi Gereken Özellikler**

Motor tipi: Fırçalı ve fırçasız olmak üzere iki farklı ESC devresi kullanılabilir.

Motorun nominal gerilimi ve akımı: Kullanılacak olan motorun maksimum akım değerinin ve nominal gerilim değerinin kullanılacak olan ESC devresi ile uyumlu olması gereklidir. Örneğin motor maksimum 40 amper tüketiyorsa bunun için uygun ESC maksimum motorun çektiği amper değerinin üzerinde; yani 45-50 amperlik olması uygundur. Ayrıca motorun ve bataryanın voltaj değeri ESC'nin voltaj değerinden yukarıda olmamalıdır. Motorlar ilk hareket anındaki moment değerini yenmek için daha yüksek akım çekmektedir. ESC'nin ilk hareket anında yüksek akıma daha az maruz kalması için yumuşak başlatma adı verilen yavaştan hızlıya doğru artan devirle başlamasını sağlayan kontrol programları bulunur. Benzer şekilde bir kaza anında pervanelerin verebileceği hasarı azaltmak için motorun dönüşünü kilitleyecek şekilde ESC akım kesme işlemi yapabilmektedir.

ESC Kullanımında Kaçınılması Gereken Durumlar:

İHA motorlarının hızları ve çektikleri akım değerleri göz önünde bulundurulduğunda çalışma prensibine uygun kullanılmaları uzun ömürlü olmalarını sağlayacaktır. Her ESC'nin bir kapasite ve dayanma toleransı vardır. Bu tolerans akım (amper) ve volt olarak aşılırsa ESC kartınızın yanması ile sonuçlanabilir. Bu yüzden motor ve batarya akım ve gerilim özelliklerinin uyumlu seçilmesi gereklidir.

ESC devresi çalışırken yüksek amperler üzerinden geçeceği için ısınması normaldir. Ancak aşırı ısınma yaşanıyorsa, hız kontrol kartınız zorlanıyor demektir. Soğumak üzere bekletiniz. Motorun mekanik olarak zorlanmadığından emin olmak ve motor – esc uyumunun kontrol edilmesi gereklidir. Sistem, ESC'ler üzerlerindeki alüminyum soğutucular aracılığı ile soğutulmaktadır. Bu nedenle İHA üzerinde hava akışı bulunan alanlarda konumlanması önemlidir.

1.5. Gözle: Batarya

İHA sistemlerinde enerji kaynağı olarak çoğunlukla Lityum Polimer (LiPo) bataryalar kullanılmaktadır. LiPo bataryalar diğer alternatif bataryalara nazaran daha hafif, daha az hacim kaplamasına rağmen daha yüksek enerji depolama kapasitesine sahiptir. LiPo bataryaların enerji kapasitesi 0.36-0.47 MJ/kg aralığında olmaktadır. Ancak maliyeti Li-İyon bataryalara nazaran daha pahalıdır. LiPo piller hücrelerden oluşmakta ve bir hücreli LiPo pil 3.7 V nominal gerilime sahip olmaktadır. Ancak, hücrenin tam dolu olması durumunda hücre 4.2 V gerilim değerine sahip olacaktır. Seri olarak bağlanan hücre sayısı arttıkça gerilim değeri 3.7 V'un katı olacak şekilde artmaktadır. 1S, 2S, 3S, 4S, 5S ve 6S olarak çeşitleri bulunan LiPo pillerde "S" bataryadaki toplam hücre sayısını ifade etmektedir. Ayrıca künyelerinde geçen "C" değeri bataryanın elektrik yükünü ne kadar hızda boşaltabileceğini göstermektedir. Yani "C" değeri bataryanın verebileceği en fazla akım değerini belirlemektedir. "C" değeri batarya seçiminde son derece önemlidir. Bu değer doğru olarak seçilmezse motorun ve ESC'nin anlık olarak talep ettiği akım değerini batarya sağlayamayacak ve bu durum İHA'nın beklenen performansı sergileyememesine neden olacaktır.

Bir örnekle yukarıda ifade edilen büyüklüklere bakacak olursak üzerinde "7.4 V 800 mAh 20C" yazan bir bataryanın nominal geriliminin 7.4 V (2S olduğunu) ve enerji depolama kapasitesinin (bir saatte vereceği akım değeri) saatte 0.8A olduğu anlaşılmaktadır. Ayrıca bu bataryanın verebileceği en fazla akım (anlık akım) 800mAhx20C=16000mA yani 16A'dır. ESC bu değerin üzerinde bir akım talep ettiğinde batarya bu akımı sağlayamayacağı gibi aşırı ısınma ve patlama riski ile karşı karşıya kalınabilecektir. Bataryanın mAh (batarya akımı/saat kapasitesi) değeri aynı zamanda uçuş süremizi de belirlemektedir. Bu değerin artması bataryanın boyutunun büyümesi ve dolayısıyla ağırlığının da artması anlamına gelmektedir. Hücrelerin gerilim değerinin 3V'un altında kullanımı hücreye zarar verebilmektedir. Bu nedenle hücrenin 3V altında gerilim değerinde olması durumunda şarj edilmesi beklenmektedir. Bataryaların raf ömrünü arttırmak için %40-50 şarj seviyesinde oda sıcaklığında saklamak gerekmektedir. LiPo bataryalar çok yüksek akımları anlık verebilirken şarj edilirken düşük akımlarla şarj edilmelidir. İdeal bir LiPo şarjı 1C-2C karşılığı bir akım değeri ile yapılmalıdır. LiPo bataryaların saklanması ve güvenli şarj edilebilmesi için saklama çantalarının kullanımı da önerilmektedir.

7.4V Lipo Batarya 1200mAh 2S Güç Modülü

1.6. Uygula: Örnek İtki Hesabı ve Motor Seçimi

Tasarlanacak aracın ve bileşenlerinin özellikleri aşağıdaki tabloda verilmiştir.

İHA Toplam Kütlesi	4 kg
Motor KV	1200
Pil	3S
Pervane	12x45
Motor Sayısı	4

IHA'nın yarı devirde havada asılı (hover) kalacak şekilde tasarlanması gerekmektedir. Bu durumda gerekli yarı devir aşağıdaki formül yardımıyla hesaplanmaktadır.

Yarı Devir = 1200x12.6/2=7.560 devir/dakika (RPM)

Burada 1200 ve 12.6 değerleri sırasıyla motorun KV değeri ve pilin tam dolu olduğu durumdaki gerilimdir. Bu iki değerin çarpımının ikiye bölünmesinin sebebi İHA'nın yarı güçte çalışıyor olması dolayısıyladır.

Bir sonraki aşamada seçilen motor ve pervane çiftinin 7560 devir/dakika'da ürettiği itki hesaplanacaktır. Bu amaçla pervane üreticisinin ilgili devir için kataloglarda belirtmiş olduğu pervane itki sabiti bulunmalıdır. Pervaneden elde edilecek itki:

$$T = C_T \rho n^2 D^4$$

formülü ile hesaplanmaktadır. Bu formülde geçen her bir terimin ne anlama geldiği daha önceki kısımlarda ifade edilmiştir.

Seçilen parametreler göz önüne alındığında n=7560/60=126 devir/saniye, hava yoğunluğu 1.225 kg/m^3, pervane çapı 0.3048 m ve pervane itki sabiti 0.065'tir.

T=0.065x1.225x126^{2}x0.3048^4=10,91N.

Seçilen motor ve pervane parametrelerine göre tek bir motor-pervane çiftinin sağlayacağı itki 11,75N'dur. Döner kanat araçta her bir motor-pervane için gerekli itki ise

$T_{Gereken}$=4*9.81/4=9,81N.

Yapılan hesaplamalardan seçilen motor-pervane çiftinin hover durumunda İHA için gerekli itkiyi rahatlıkla sağlayabileceği anlaşılmaktadır.

2. Tasarla

Sınıftaki tüm öğrencilerin katılımıyla pervanenin ürettiği itkiyi ölçen bir test düzeneği tasarlaması beklenmektedir. Pervanenin devrini arttırmak suretiyle oluşan itkiyi ölçen bu test düzeneğinde gerekli ölçümleri yapmak üzere hangi cihazlara ihtiyacımız olacaktır?

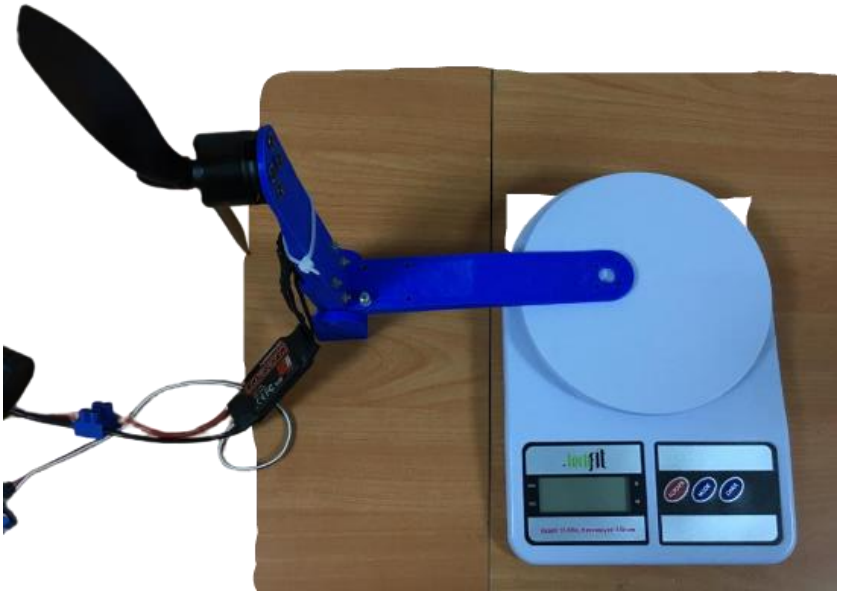
3. Üret

Aşağıda verilen itki ölçme sistemini kurunuz. Farklı motor ve pervane seçenekleri için güç analizöründen ölçülen değerleri (gerilim ve akım) ve gerilim ve akımın çarpımıyla bulunan güç değerini farklı devirler için belirleyiniz.

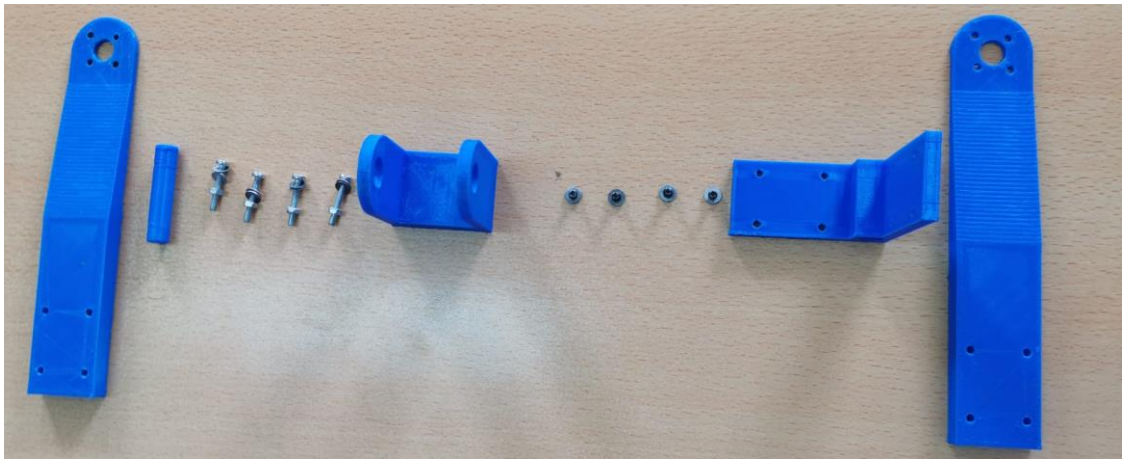
Ölçtüğünüz ve hesapladığınız değerleri makul buluyor musunuz?

İtki Ölçme Sisteminin Kuruluşu

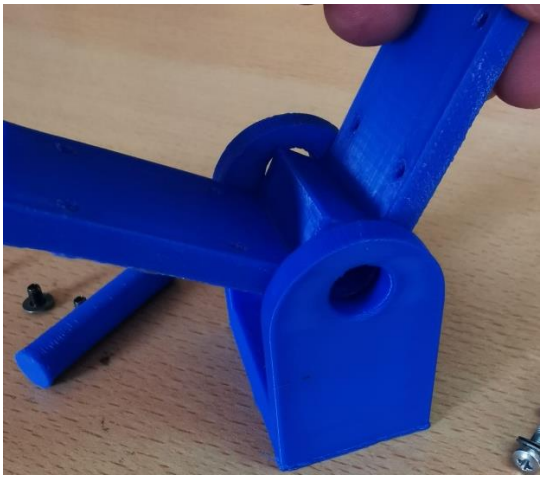
İtki ölçme sistemi motorun çalıştığında geriye doğru yaptığı hareketle teraziye yük bindirerek gram cinsinden motorun itkisini ölçmeyi sağlar. İtki ölçüm sisteminin iki ucu vardır. Bu uçlardan birine motor bağlanırken diğer uç terazi üzerine yerleştirilir.

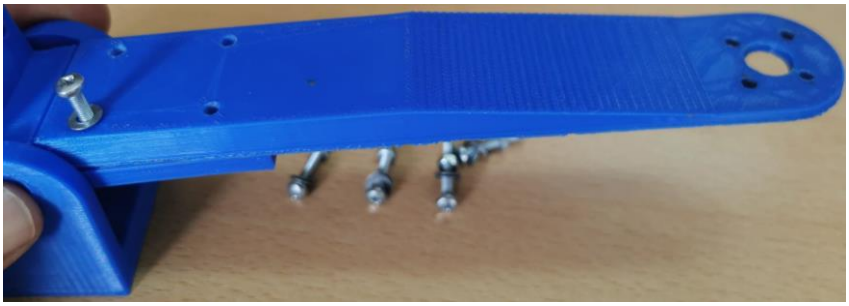


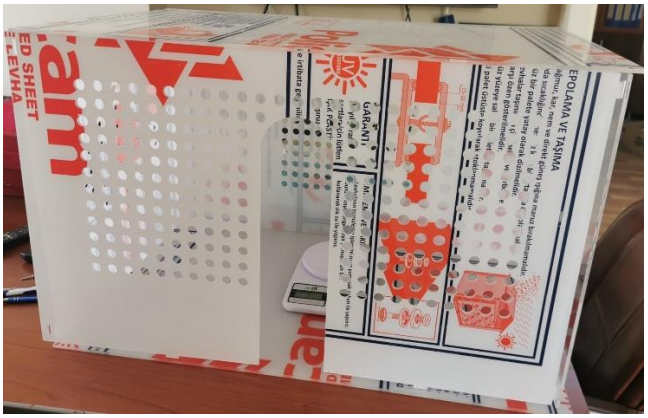
İtki Ölçme Sisteminin Genel Görünüşü



İtki Ölçme Sistemindeki Her Bir Parçanın Montaj Öncesi Görüntüsü







Terazinin darası alınarak pervanenin yönü kontrol edilmelidir. Motorun bağlantıları yapılarak itki sistemine batarya yardımıyla güç verilir. Motor çalıştığında oluşan itki kuvveti

sistemi geriye doğru hareket ettirerek teraziye yük binmesini sağlar. Her iki kolun da uzunluğu aynı olduğundan teraziden ölçülen değer motorun o anki itki değerini göstermektedir.

Motor İtki Ölçme Sistemi Elektronik Bağlantısı

Motor

Motor Sürücü

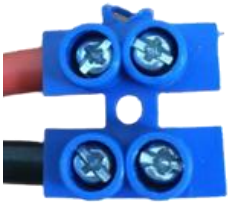
Motordan çıkan 3 kablo motor sürücüsünde (ESC) bulunan 3 kabloya şekildeki gibi bağlanır.

1 ve 3 yön belirler.
1-1 3-3 / 1-3 3-1
şeklinde
bağlanabilir.

İki cihazın da ortasındaki kablolar birbirine bağlanır. Diğer iki kablo motorun dönüş yönünü belirler. Motor çalıştırıldığında saat yönü ya da saat yönünün tersine döner. Motorun istenilen yönde dönmesi için bu iki kablonun kombinasyonları denenir. Motor sürücüsünün diğer tarafında şekildeki gibi iki güç kablosu bir de sinyal kablosu bulunmaktadır.

Hız Kontrolcüsü

Sinyal kablosu hız kontrolcüsünün out kısmına bağlanır. Hız kontrolcüsünün üzerinde bulunan dönebilen parça çevrilerek motorun hızını ayarlar. Kalın kırmızı ve siyah kablolar pile giden güç kablolarıdır. Kırmızı renk pilin artısını temsil ederken siyah renk de pilin eksisini temsil eder.



Bu kablolar iki kabloyu birleştiren klemens aracılığıyla güç analizörünün load kısmına bağlanır.

Klemens

Güç Analizörü

Güç analizörü motorun çektiği akımı voltajı ve pil tüketimini gösteren bir cihazdır. Güç analizörünün source kısmından çıkan güç kabloları bataryaya gitmeden önce yine klemens aracılığıyla güç modülüne bağlanır.

Batarya

Güç Modülü

Güç modülü de güç analizörü gibi akım, voltaj ve pil tüketimini ölçer ama bu verileri göstermez, uçuş kontrol kartına aktarmayı sağlar. Güç modülünün diğer tarafında güç kabloları ve uçuş kontrol kartına akım, voltaj ve pil tüketimi verisi gönderen sinyal kablosu bulunmaktadır. Bu devrede herhangi bir uçuş kontrol kartı kullanılmadığı için güç modülünün sinyal kablosu kullanılmaz.

Kalan güç kabloları konnektör ile bataryaya bağlanır.

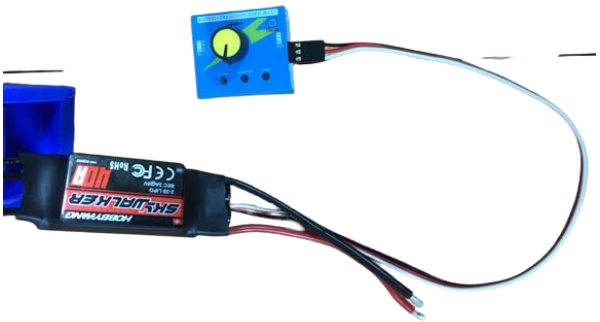
Konnektör

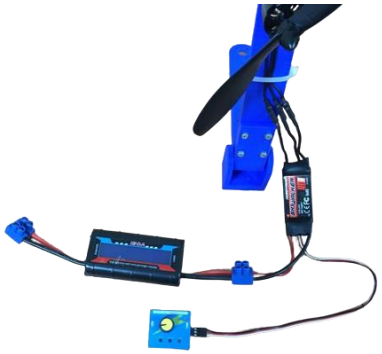
Dikkat: Bu güç kablolarının artı ve eksi yönü çok önemlidir. Ters bağlanması durumunda batarya patlayabilir!

Şekillerle Adım Adım Devre Bağlantısı

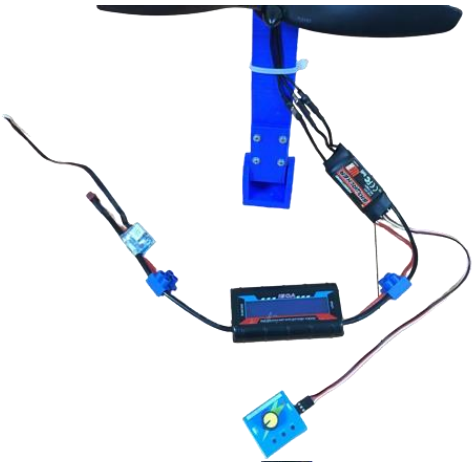
1.

2.

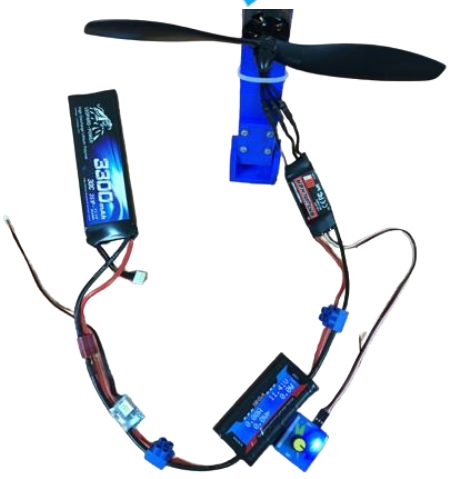




3.



4.



5.

Motor Bilgi Kâğıdı: 11.1 Volt 3S batarya kullanıldığında çekilen akıma göre elde edilen itki değeri tabloda gösterilmiştir.

The voltage (V)	Paddle size	current (A)	thrust (G)	power (W)	efficiency (G/W)	speed (RPM)
11.1	EMAX8045	1	110	11.0	10.0	3650
		2	200	22.0	9.1	4740
		3	270	33.0	8.2	5540
		4	330	44.0	7.5	6200
		5	380	55.0	6.9	6700
		6	420	66.0	6.4	7150
		7.3	470	81.0	5.8	7400
	EMAX1045	1	130	11.0	11.8	2940
		2	220	22.0	10.0	3860
		3	290	33.0	8.8	4400
		4	360	44.0	8.2	4940
		5	430	55.0	7.8	5340
		6	490	66.0	7.4	5720
		7	540	77.0	7.0	5980
		8	580	88.0	6.6	6170
		9	620	99.0	6.3	6410
		9.5	640	105.5	6.1	6530
14.8	EMAX8045	1	130	14.8	8.8	3900
		2	230	29.6	7.8	5180
		3	310	44.4	7.0	6000
		4	390	59.2	6.6	6610
		5	470	74.0	6.4	7200
		6	530	88.8	6.0	7570
		7	580	103.6	5.6	7910
		8	630	118.4	5.3	8230
		9	670	113.2	5.0	8500
		10	700	148.0	4.7	8780
		10.7	720	158.4	4.5	9030

5. HAFTA: AVİYONİK SİSTEMLER

Ön Bilgi:

- Temel elektrik ve elektronik terminolojisini bilmek.
- Elektronik malzemeleri tanımak.
- 3D yazıcıdan baskı alabilmek.
- Arduino ile programlama yapabilmek.
- Deneyap kart temel yapısına hâkim olmak.
- PID kontrol bilgisine hâkim olmak.

Haftanın Kazanımları:

- Hava araçlarının elektrik bileşenlerini bilir.
- Hava aracı güç gerekliliklerini tanımlar.
- Hava araçlarının elektronik bileşenlerini bilir.
- Sensör verilerini yorumlar.
- Hava araçlarının kendi arasında ve yer istasyonları ile iletişimini kavrar.
- Uçuş kumandaları ve kontrol yüzeylerinin çalışma prensiplerini bilir.
- Temel uçuş manevraları için gerekli kumanda ve kontrol ara yüzlerini donanım olarak üretir
- Temel uçuş manevraları için yazılım geliştirir.
- Yazılım ve donanım arasında entegrasyon kurar.

Haftanın Amacı:

Hava araçlarındaki elektrik ve elektronik bileşenleri tanıtmak, kullanım usul ve amaçlarını göstermek. Sensör, servo ve aktüatörler yardımı ile bir hava aracının kararlı uçuşu için gerekli kontrol algoritmaları ve temel uçuş manevralarını elektronik olarak gerçekleştirecek donanım ve yazılım bilgisini kazandırmak.

Kullanılacak Malzemeler:

- Önceden 3 boyutlu çıktıları alınmış itki, kumanda ve kontrol bileşenleri.
- İnsansız hava aracı kiti.
- Servo ve fırçasız DC motorlar.
- Çeşitli ölçülerde pervaneler.
- Çeşitli ölçülerde kablo ve bağlantı elemanları.

- Deneyap kart.
- Arduino IDE arayüzü.

Haftanın İşlenişi:

Gözle: Temel aviyonik bileşenlerin tanıtılması. Uçak üzerindeki ve çalışma usullerinin tartışılması.

Uygula: Deneyap kart üzerinde analog sensör değeri okuma ve servo motor sürme.

Tasarla: Bir sabit kanatlı uçak için kontrol yüzeylerini yönetecek Fly By Wire sistemi tasarlama.

Üret: 3B çıktısı alınmış kumanda kolu bileşenlerini bir araya getirerek model uçak üzerindeki servoları kumanda kolundan gelen komutlara göre konumlandıracak Fly By Wire sistemi oluşturma.

Değerlendir: Modeli oluşturulan Fly By Wire sisteminin gerçek uçaklardaki yapısını düşünerek model ile benzer ve farklı yönlerini değerlendirme.

UYARI

Bu dersin üret aşamasındaki parçalar 3 boyutlu yazıcıda basılmalıdır. Bu nedenle sınıftaki grup sayısı kadar parça ders öncesinde 3 boyutlu yazıcıda basılarak ders sırasında hazır halde olmalıdır.

Parçalara ait 3boyutlu çizim ve STL dosyalarına
https://owncloud.tubitak.gov.tr/index.php/s/pzTAtdzksc2UCBz

Adresinden erişilebilir.

1. GÖZLE

1.1. Aviyonik Nedir?

Eğitmen yönetiminde öğrencilere “aviyonik” terimi tanıtılır. Örnek olabilecek aviyonik bileşenler öğrencilerle soru cevap şeklinde tartışmaları sağlanır. İngilizce kelime köklerine inilerek “aviation” (havacılık) ve “electronic” (elektronik) terimlerinden bahsedilerek öğrencilerin aviyonik kavramını kelime köküyle bağdaştırmaları sağlanır. “Uçakta elektronik hangi parçalar olabilir?”, “Bu parçalar uçağın uçmasına nasıl yardımcı olur?”, “Pilotun kullandığı elektronik parçalar nelerdir?”, “Yerden uçağa destek sağlayan elektronik parçalar var mıdır?” gibi sorularla eğitmen tartışmayı yönetir. Cevaplara öğrencilerin katkı sağlaması beklenir. Tartışmada ortaya çıkan görüşler doğrultusunda eğitmen aviyonik tanımını doğrudan kendisi yapabilir ve bileşenler ile görevlerini detaylandırabilir.

İngilizce havacılık anlamına gelen “aviation” ve elektronik anlamına gelen “eectronics” kelimelerinin bir araya getirilerek oluşturulmuş “avionics” sözcüğü Türkçe olarak da okunduğu şekliyle “Aviyonik” olarak kullanılmaktadır. Adındaki birleşimden de anlaşılacağı üzere havacılık elektroniği anlamına gelmektedir.

Her modern uçak, uzay aracı ve yapay uydu, amaçlarına ve görevlerine uygun bir dizi işlevi yerine getirmek için çeşitli tiplerde elektronik sistemler kullanır. Genel olarak, araç veya görev ne kadar karmaşıksa, kullandıkları elektronik sistemler de o kadar karmaşıktır.

Ticari uçaklar, helikopterler, askeri savaş uçakları, insansız hava araçları (İHA) ve uzay araçlarının tümü görevleri yürütmek, yeni keşifler yapmak, performans önlemlerini izlemek ve raporlamak, belirlenmiş güvenlik parametreleri dahilinde çalışmak için aviyonik sistemler kullanır. Günümüzde gelişmiş aviyonik sistemler uçuş performansını artırmak, bakımı basitleştirmek ve maliyetleri kontrol altına almak için birden çok işlevi entegre eder.

"Havacılık" ve "elektronik" terimlerinin gerçek bir karışımı olan bir uçak veya uzay aracına takılan aviyonikler, motor kontrollerini, uçuş kontrol sistemlerini, seyrüseferi, iletişimi, uçuş kayıt cihazlarını, aydınlatma sistemlerini, tehdit algılamayı, yakıt sistemlerini, elektro-optiği içerebilir. Aviyonik sistemler yalnız hava aracı üzerinde değil onu destekleyici yer sistemlerinde de bulunan radar sistemleri, aletli iniş sistemleri, yer kontrol istasyonu gibi elektronik ekipmanları da kapsamaktadır.

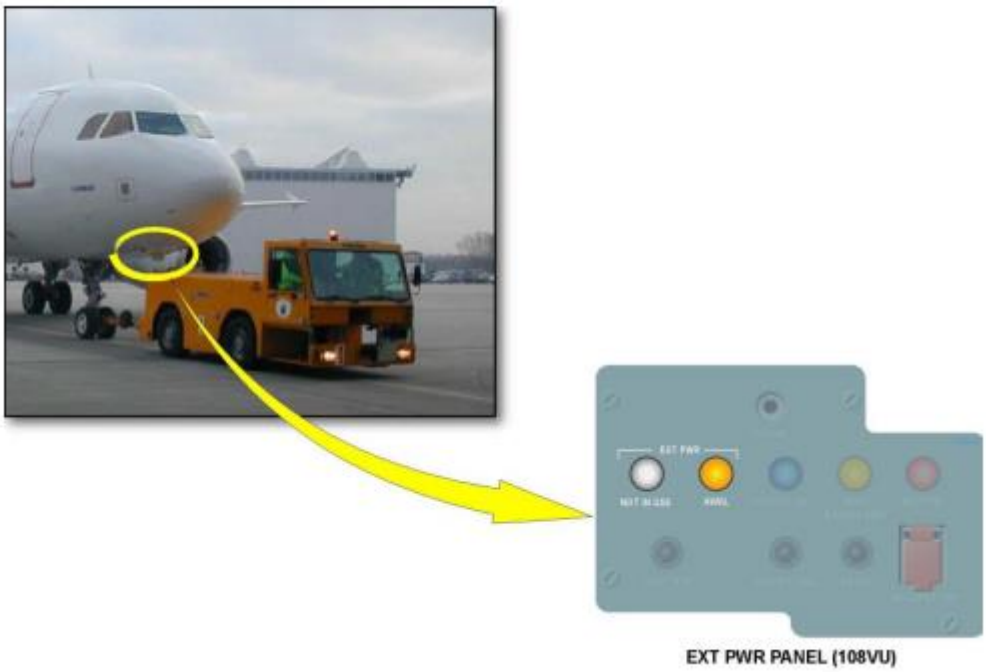
1.2. Hava Aracı Elektrik Bileşenleri

1.2.1 Elektrik Güç Kaynakları

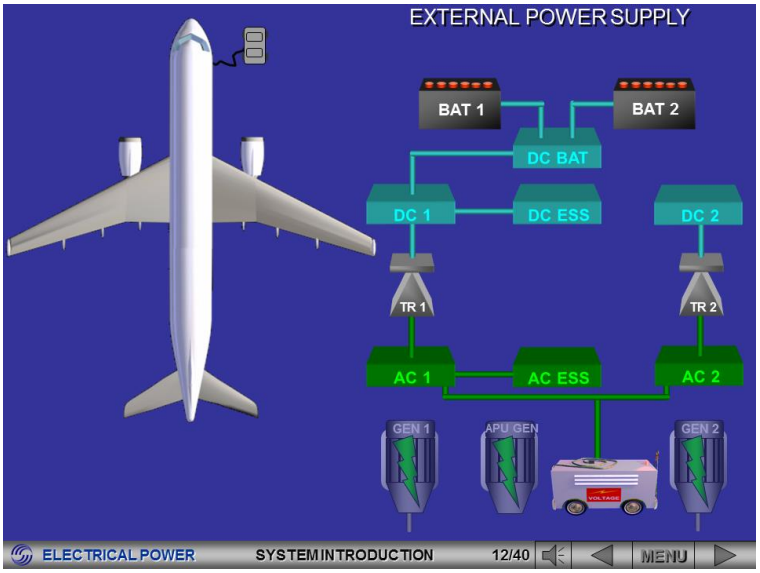
Günümüz modern uçaklarında ana elektrik güç kaynağı uçağın motorundan güç alan elektrik jeneratörleridir. 115 V 400 Hz AC şeklinde üretilen bu enerji uçağın ihtiyaç duyulan bütün alanlarına dağıtılabilmektedir. Ayrıca olası motor arızalarında veya uçak yerde motorları çalışmadığı durumlarda kullanılan Harici güç ünitesi (APU) de uçağa enerji sağlayabilmektedir. Uçaklar yerde durduklarında ayrıca havalimanında yer alan yer güç istasyonlarından enerji sağlanabilmektedir.

Uçaklarda AC enerjinin kesilmesi durumunda kritik bileşenleri beslemeye devam eden batarya sistemleri kullanılmaktadır. Böylelikle uçakta bütün üreteçler dursa dahi uçak bataryadan sağladığı enerji ile uçuşunu devam ettirebilir.

Acil durumlarda uçakta bütün elektrik sistemi arıza verirse devreye giren ve uçağın alt tarafında otomatik olarak açılan bir rüzgâr jeneratörü bulunmaktadır. “RAM AIR TURBINE (RAT)” olarak adlandırılan bu rüzgâr jeneratörü uçuş kontrolü için hayati olan parçalara yetecek enerjiyi üretebilmektedir.



Şekil 1. *Uçak Yer Güç İstasyonu Bağlantı Paneli*



Şekil 2. *Uçak Güç Kaynakları*

Şekil 3. *RAM AIR TURBINE*

1.2.2 Elektrik Güç Dönüştürücüleri

Uçaklarda değişik sistemler değişik gerilim ve akım değerlerinde çalışabilmektedir. Bazı sistemler DC gerilim kullanırken bazıları ise AC gerilim ile beslenmektedir. Bu nedenle uçak içerisinde AC gerilimleri DC gerilimlere çeviren dönüştürücüler yer almaktadır. Ayrıca uçağın ana güç ünitesinde üretilen 115 V AC gerilim ihtiyaç duyulan parçanın gerilimine göre 48 V, 24 V, 12 V, 6 V ve 5 V gibi çeşitli değerlere düşürülebilmektedir. Özellikle uçağın acil durumlarda batarya ile beslenen bileşenleri genellikle 24 V DC gerilim ile beslenmektedir.

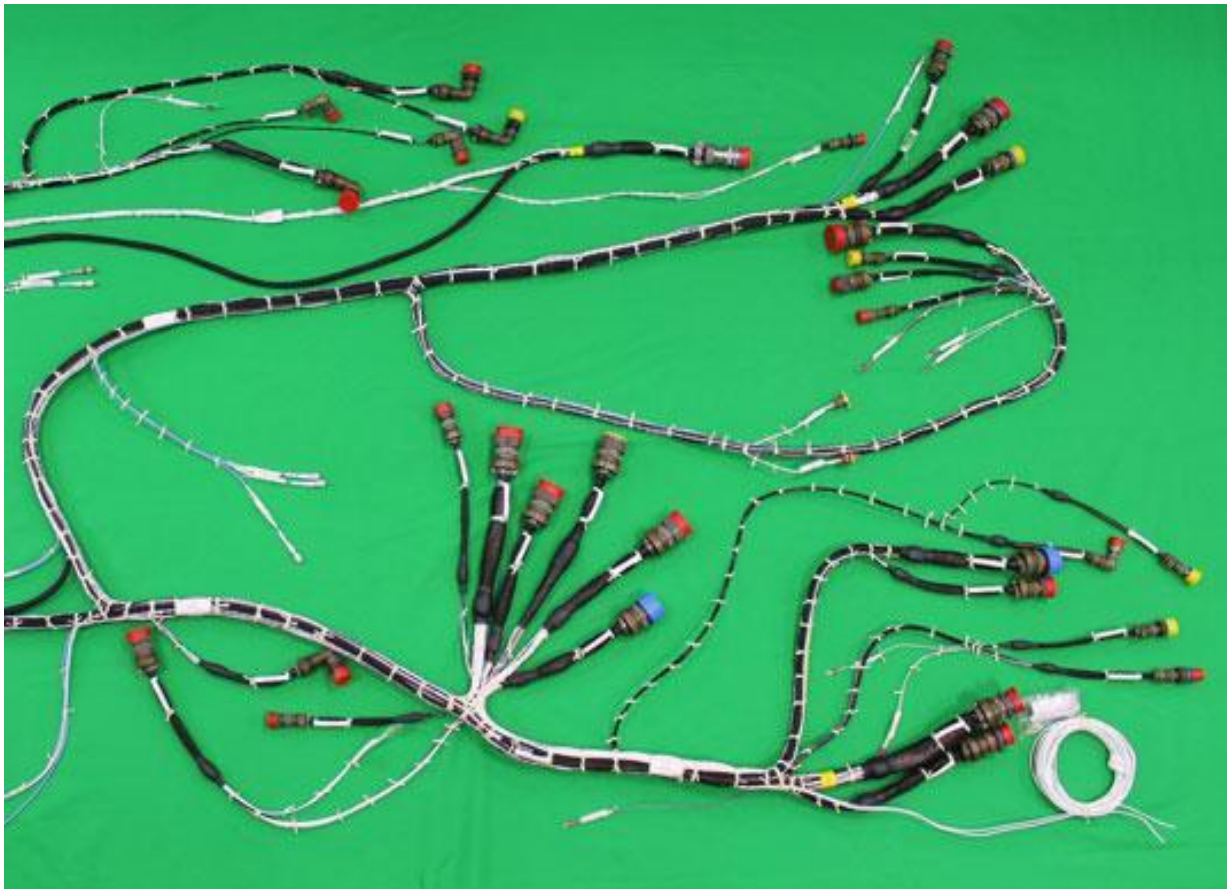
Şekil 4. *Uçaklarda Kullanılan 400Hz Statik Dnüştürücü Örneği*

Fırçasız DC motor kullanan insansız hava aracı sistemlerinde de aynı yolcu uçaklarında olduğu gibi değişik bileşenler değişik gerilim değerlerinde çalışmaktadır. Bir döner kanatlı quadkopter veya sabit kanatlı İHA'nın motorları 12 V, 48 V gibi gerilim değerleri ile beslenirken, uçuş kontrol kartı, radyo alıcısı, telemetri sistemi gibi bileşenler ise genelde 5 V veya 3 V gerilimler ile çalıştırılmaktadır. Bunun için İHA sistemlerinde doğrudan batarya sistemine bağlanan Batarya Eliminasyon Devresi (BEC) kullanılmaktadır. BEC harici olarak bağlanabildiği gibi bazı ESC'lerin içerisinde dâhili olarak da gelebilmektedir.

Şekil 5. *Örnek Bir BEC Elemanı*

1.2.3 Kablo ve Konektörler

Hava araçlarında elektrik ve elektronik bileşenleri modüler yapıda tasarlanır. Böylelikle parçaların bakımı ve arıza durumlarında değiştirilmesi kolayca yapılabilmektedir. Modüllerin birbirleri ile bağlantıları için belirli konnektör standartları belirlenmiştir. Bu standartlar doğrultusunda uçak içerisinde parçalar birbirleri ile uyumlu şekilde bağlanabilmektedir. Modüller arasında birçok farklı sinyal gönderildiğinden genellikle bu konnektörlere büyük kablo demetleri bağlanmaktadır. Kablonun hava aracı içerisinde gideceği mesafe ve taşıyacağı akım düşünülerek farklı kalınlıklarda kablolar seçilmektedir.



Şekil 6. *Hava Araçlarında Kullanılan Konnektör ve Kablo Demeti Örnekleri*

İnsansız hava aracı sistemlerinde de kullanılan batarya türü ve taşıdığı akım değerlerine göre farklı konnektörler ve kablolar kullanılmaktadır. Piyasada yaygın kullanılan LiPo pillerde XT ve DEANS-T veya

JST olarak bilinen konnektörler kullanılmaktadır. Gerilim değeri büyüdükçe konnektör çapı da artmaktadır buna göre XT60 genellikle 3S ve 4S pillerde XT90 ise 6S ve üstü pillerde tercih edilmektedir.

Şekil 7. *Batarya Konnektör Örnekleri*

Uçuş kontrol kartları üzerinde ise servolar, radyo kumanda alıcıları, GPS, telemetri ve benzeri birçok bileşen için her bileşenin desteklediği konnektör çıkışları bulunmaktadır. Servo ve radyo kumanda alıcıları için J ve JR tipi konnektörler kullanılır.

1.3. Hava Aracı Elektronik Bileşenleri

Hava araçlarında kullanılan elektronik sistemlerin geçmişi I. Dünya Savaşı'na dayanmaktadır. 1910 yılında bir uçaktan gemiye doğru yapılan deneysel radyo haberleşmesi ile uçaklarda elektronik bileşenlerin kullanımı başlamıştır. 1917'de vakum lamba sistemlerinin gelişmesi ile iki yönlü radyo telsiz sistemleri kullanılmaya başlamıştır. Radar ve hava trafik kontrol sistemlerinin gelişimi ise II. Dünya Savaşı döneminde hava savunma sistemi ihtiyacı ile ortaya çıkmıştır. Günümüz hava araçlarında kullanılan birçok sistem öncelikle askeri ihtiyaçlarla ortaya çıkmış, ardından sivil ve askeri tüm uçaklarda kullanımına geçmiştir. Günümüz modern aviyonik sistemlerde uçuş emniyetini arttırıcı ve insana dayalı hataları en aza indirecek sistemler kullanılmaktadır.

Bir hava aracının aviyonik sistemleri incelendiğinde aracın durumunu gözlemlemeye yarayan çok sayıda sensör karşımıza çıkmaktadır. Diğer önemli elektronik bileşenler ise hava aracının kumanda ve kontrolünü sağlayan servo ve aktüatör sistemleridir. Bir hava aracı haberleşme sistemleri olmadan kör ve tek başınadır. Bu nedenle diğer önemli bir elektronik bileşen de haberleşme sistemleridir. Günümüz modern uçaklarında artık pilotun yaptığı görevlerin çoğu uçağın uçuş kontrol bilgisayarları ile yapılmaktadır. Bayraktar TB2 İHA gibi sistemler adeta bir robot gibi uçuş bilgisayarına yüklenen görev planları ile

hangarından çıkıp pist başına taksi yapıp otomatik olarak kalkış, seyir ve otomatik inişini gerçekleştirebilmektedir. Bunun başarılı bir şekilde gerçekleşmesindeki önemli unsurlardan bir tanesi seyrüsefer sistemleridir. Küresel Konumlama Sistemi (GPS) ve benzeri birçok sistem hava aracının dünyadaki konumunu anlamasına ve buna göre rotasında kalmasına yardımcı olmaktadır.

Günümüz modern hava araçları bütün bu elektronik bileşenlerin birbirleri ile bütünleşmiş çalıştığı ve yerdeki sistemlerle iletişimde kalarak görev icra eden tam veya yarı otonom robot sistemleridir. İlerleyen bölümlerde bu bileşenleri detaylandırarak hava aracındaki görev ve işlevlerini tanıtacağız.

1.3.1. Sensörler

Hava araçlarının durumunu ve sistemlerinin beklendiği şekilde çalıştığını anlamak için sensörler bize yardımcı olmaktadır. Kendi evimizde yapabileceğimiz minik bir insansız hava aracında bile aracın kararlı şekilde uçmasını sağlatmak için onlarca farklı sensör bilgisine ihtiyaç duyulmaktadır. Günümüz yolcu veya savaş uçaklarındaki karmaşık sistemler düşünüldüğünde çok sayıda sensörden gelen veri işlenmektedir.

Bir hava aracının temel uçuşunu gerçekleştirebilmek için hızını, durumunu, irtifasını ve yönünü bilmek gereklidir. Bir pilot Şekil 1'de gösterilen temel bu dört göstergeyi gözlemleyerek bir hava aracını uçurabilir ve temel uçuş manevralarını sağlatabilir. Bu nedenle en başta bu verileri okuyabilecek elektronik bileşenlere ihtiyaç vardır. Ardından aracın konumu ve üzerindeki diğer bileşenlerin çalıştığını gösterecek sensörlere ihtiyaç duyar.

Şekil 8. *Temel T Yerleşiminde Uçuş Göstergeleri*

1.3.1.1. Ataletsel Ölçüm Birimleri (IMU)

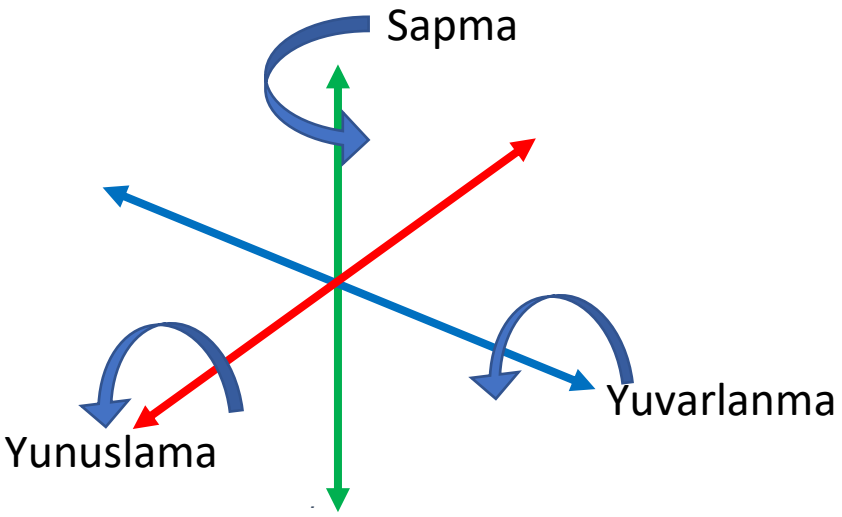
Ataletsel Ölçüm Birimi (Inertial Measurement Unit, IMU), açısal hızı, kuvveti ve bazen manyetik alanı ölçen özel bir sensör türüdür. IMU'lar, 3 eksenli bir ivmeölçer ve 6 eksenli bir IMU olarak kabul edilecek 3 eksenli bir jiroskoptan oluşur. Ayrıca 9 eksenli bir IMU olarak kabul edilebilecek ek bir 3 eksenli manyetometre içerebilirler. Teknik olarak, "IMU" terimi sadece sensörü ifade eder, ancak IMU'lar genellikle yön ve yön ölçümleri sağlamak için birden fazla sensördan gelen verileri birleştiren sensör füzyon yazılımı ile eşleştirilir. Yaygın kullanımda, "IMU" terimi, sensör ve sensör füzyon yazılımının kombinasyonunu belirtmek için kullanılabilir. Bu kombinasyon aynı zamanda bir (AHRS) durum rota referans sistemi olarak da anılır.

Bir IMU, bir nesnenin 3B uzayda hareket edebildiği farklı yolların sayısını ifade eden 2 ila 6 DOF (serbestlik derecesi) sağlar. Mümkün olan maksimum değer, düz bir düzlem boyunca/her eksen boyunca (ön/arka, sağ/sol, yukarı/aşağı) 3 derece öteleme (düz) hareket ve x, y ve z eksenleri boyunca 3 derece dönme hareketi içeren 6 serbestlik derecesidir.

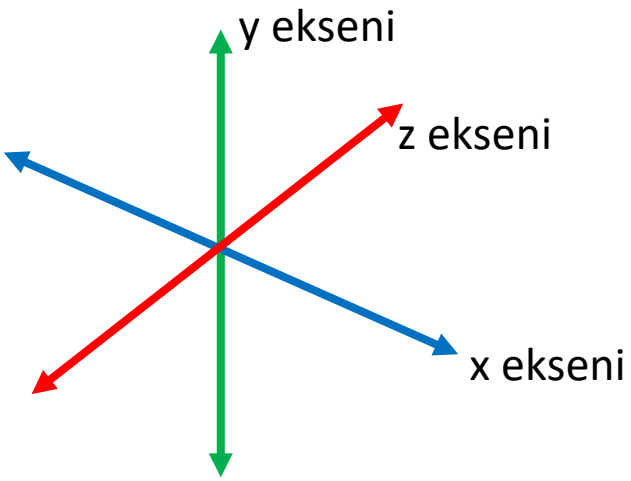
Bir IMU'dan toplanan ham veriler, etrafındaki dünya hakkında bir fikir verir. Ancak bu bilgiler ek matematiksel birleştirme yöntemleri ile işlenebilir. Sensör füzyonu, cihazın oryantasyon ve yönünün daha eksiksiz bir resmini oluşturmak için her sensörden gelen verileri bir IMU'da birleştirme (matematiksel) yöntemidir. Örneğin, dönme hareketi için jiroskop bilgilerine bakarken, bir referans çerçevesi oluşturmak

için bir ivmeölçer yerçekimi hissini dâhil edebilirsiniz. Tüm sensörü Dünya'nın çerçevesine hizalamak için ayrıca Dünya'nın manyetik alanı hakkında bilgi ekleyebilirsiniz.

İvmeölçer: En yaygın kullanılan hareket sensör türü ivmeölçerdir. Arabanızda gaza bastığınızda veya



telefonunuzu düşürdüğünüzde olduğu gibi, tek bir eksende ivmeyi (hız değişimini) ölçer. İvmeölçerler, belirli bir yönde doğrusal ivmeyi ölçer. Yerçekimini aşağı doğru bir kuvvet olarak ölçmek için bir ivmeölçer de kullanılabilir. İvmenin integral verisi hız için bir tahmin ortaya çıkarır ve tekrar integral alındığında ise size konum için bir tahmin verir. İkili integral zamana bağlı olarak yüksek sapma yaratacağından ivmeölçer tek başına önerilen bir mesafe tahmini yöntemi değildir.



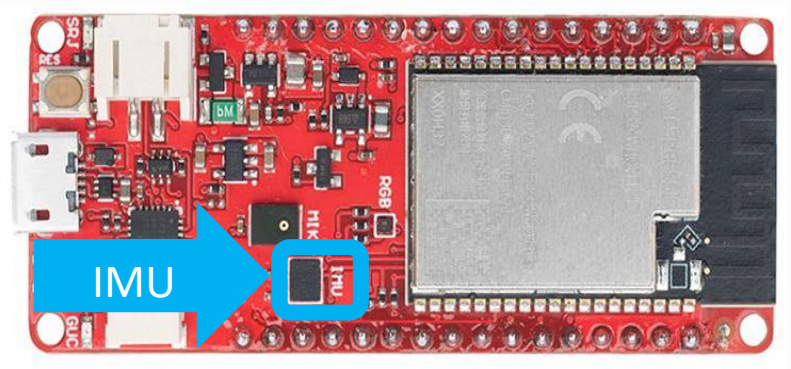
Jiroskop: İvmeölçerler doğrusal ivmeyi ölçebilirken, bükülmeyi veya dönme hareketini ölçemezler. Ancak jiroskoplar açısal hızı üç eksen etrafında ölçer: Yunuslama, yuvarlanma ve sapma.

Sensör füzyon yazılımı ile entegre edildiğinde, bir nesnenin 3B alan içindeki yönünü belirlemek için bir jiroskop kullanılabilir. Bir jiroskopun başlangıç referans çerçevesi olmasa da (yerçekimi gibi), açısal konumu ölçmek için verilerini bir ivmeölçerden alınan verilerle birleştirebilirsiniz.

Manyetometre: Adından da anlaşılacağı gibi bir manyetometre, manyetik alanları ölçer. Sensörün uzaydaki noktasında havanın manyetik akı yoğunluğunu ölçerek, Dünya'nın manyetik alanındaki dalgalanmaları

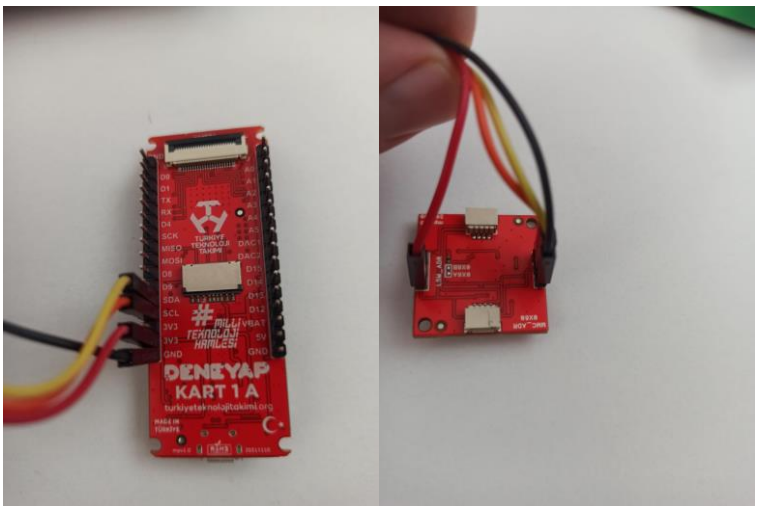
tespit edebilir. Bu dalgalanmalar sayesinde, Dünya'nın manyetik kuzeyine doğru vektörü bulur. Bu, mutlak yönü belirlemek için ivmeölçer ve jiroskop verileriyle birlikte birleştirilebilir. Gördüğünüz gibi, IMU'lar hızlanma, açısal hız ve manyetik alanları ölçmek için kullanılır ve sensör füzyon yazılımıyla birleştirildiğinde hareket, yönelim ve yönü belirlemek için kullanılabilirler.

Deneyap Kart üzerinde yüksek hassasiyetli LSM6DSM sensörü vardır. Bu sensör ile 3-eksen ivme ve 3-eksen jiroskop ölçümü yapabilmektedir. Deneyap kart kullanarak bir hava aracı geliştirdiğinizde bu hava



aracının öteleme ve dönüş bilgilerini okuyabilirsiniz.

Deneyap kart 1A sürümünde ise IMU sensörü kart üzerinde bulunmamaktadır. Farklı IMU sensör modülleri harici olarak karta bağlanabilmektedir. Harici IMU modülleri I2C arayüzü üzerinden fotoğrafta verildiği şekilde bağlantı yapılabilmektedir. Modül üzerindeki “SDA” ve ”SCL” çıkışları deneyap kart “SDA” ve “SCL” çıkışlarına bağlanmalıdır. Kartın “3.3V” besleme ve “GND” çıkışları yine deneyap kart üzerindeki “3.3V “ ve “GND” çıkışlarına bağlanarak bağlantı hazırlanabilir.



2. Uygula: Deneyap Kart IMU Sensörü Değerlerini Okuma

Kullanacağımız program kod parçacığı ile Deneyap Kart üzerinde bulunan IMU verilerini okuyacak ve buna göre kartın 3 eksende ivmelenme değerlerini gözlemleyeceğiz. Örnek kod ile X, Y ve Z ekseninde ivmeölçer verisinde değişim deneyap kartını bu eksenlerde değişik hızlarda sallanarak gözlemlenebilir. Gyro verisindeki değişim ise Deneyap artı üç eksende döndürerek izleyebilirsiniz. Burada daha sağlıklı okuma için dene yap kart diğer eksenlerde sabit tutulurken tek bir eksende hareket ettirilmelidir.

Not

Deneyap Kart ile IMU uygulaması yapacak sınıflar 'arduino ide'ye aşağıdaki kod parçacığını girmelidir:

```
#include "deneyap.h"
#include "lsm6dsm.h"

#define delayms 700

LSM6DSM IMU;                  // IMU icin Class tanimlamasi

void setup() {
  Serial.begin(115200);       // Seri haberlesme baslatildi
  IMU.begin();                // IMU ayarlari konfigure edildi
}

void loop() {
  Serial.println("\nAkselerometre degerleri\n");
  Serial.print("X ekseni\t: ");
  Serial.println(IMU.readFloatAccelX());      // X-eksen akselerometre verisi okuma
  delay(delayms);
  Serial.print("Y ekseni\t: ");
  Serial.println(IMU.readFloatAccelY());      // Y-eksen akselerometre verisi okuma
  delay(delayms);
  Serial.print("Z ekseni\t: ");
  Serial.println(IMU.readFloatAccelZ());      // Z-eksen akselerometre verisi okuma
  delay(delayms);

  Serial.println("\nGyro degerleri\n");
  Serial.print("X ekseni\t: ");
  Serial.println(IMU.readFloatGyroX());       // X-eksen gyro verisi okuma
  delay(delayms);
  Serial.print("Y ekseni\t: ");
  Serial.println(IMU.readFloatGyroY());       // Y-eksen gyro verisi okuma
  delay(delayms);
  Serial.print("Z ekseni\t: ");
  Serial.println(IMU.readFloatGyroZ());       // Z-eksen gyro verisi okuma
  delay(delayms);

}
```

Not

Deneyap Kart 1A sürümü ile IMU uygulaması yapacak sınıflar 'arduino ide'ye aşağıdaki kod parçacığını girmelidir. Kod yazılmadan önce arduino ide programında "SparkFunLSM6DS3" kütüphanesi yüklenmelidir.

```ino
#include "deneyap.h"
#include "SparkFunLSM6DS3.h"
#include "Wire.h"
#include "SPI.h"

LSM6DS3 myIMU; //I2C adresi ayarlanıyor, addr 0x6B

void setup() {

  Serial.begin(9600);
  delay(1000);
  Serial.println("İslemci baslatidi\n");

  //IMU ayarlanıyor
  myIMU.begin();

}


void loop()
{
  //Parametreler okunuyor
  Serial.print("\nİvme Olcer:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatAccelX(), 4);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatAccelY(), 4);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatAccelZ(), 4);

  Serial.print("\nJiroskop:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatGyroX(), 4);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatGyroY(), 4);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatGyroZ(), 4);

  delay(1000);
}
```

3. Gözle Uçuş Kontrol Bilgisayarları

Günümüz modern uçaklarında uçağın birçok bileşeni uçak içerisinde bulunan bilgisayarlar tarafından yönetilebilmektedir. Uçağın üzerinde bulunan sensörlerden gelen veri işlenmekte ve buna bağlı pilot veya otopilot tarafından verilen komutlara göre uçağın uçuşu yönetilebilmektedir. Sabit kanatlı hava araçlarında uçağın kanat taşıyıcılığını yönetecek şekilde uygun hız ve pozisyonda uçmasını uçuş bilgisayarı yönetebilmektedir. Döner kanatlı hava araçlarında bu süreç daha karmaşık bir şekilde gerçekleştirilir. Özellikle çok rotorlu döner kanatlı İHA sistemlerinde hava aracının kararlı kalması için uçuş kontrol bilgisayarı hassas bir kontrol algoritması ile her bir motora verilmesi gereken güç miktarını kontrol ederek İHA'nın havada kararlı bir şekilde durması ve manevralarını gerçekleştirmesini sağlatır.

Uçuş kontrol bilgisayarlarının temel girişleri sensörler ve kumanda girişleridir. Temel çıkışları ise servo ve aktüatörlerdir. Bunlar elektrik motorları, hidrolik pompalar ve çeşitli elektromekanik sistemler olabilir.

Uçuş kontrol bilgisayarı ayrıca GPS, telemetri, haberleşme üniteleri gibi bileşenlerle de iletişime geçerek uçağın konumlaması ve rota yönetimine de yardımcı olmaktadır. Otopilot yazılımı ile uçağın belirlenen rotada belirlenen hız ve irtifada uçurulması da yine uçuş kontrol bilgisayarı tarafından yönetilmektedir.

Model uçak ve İHA'ların kontrolünde de yine uçuş kontrol bilgisayarları bulunur. Daha küçük boyutlu olan bu bilgisayarların bir kısmı açık kaynak kod yazılım kullanırken azıları ise ticari lisanslı yazılımlarla yönetilmektedir. En yaygın açık kaynak kodlu uçuş kontrol bilgisayarı PIXHAWK olarak bilinmektedir. Geliştirildikçe çeşitli versiyonları ortaya çıkan bu bilgisayarda donanım olarak 32 bitlik ARM tabanlı bir işlemci, IMU sensörü, barometre gibi bileşenler dâhili olarak bulunmaktadır. 2 GPS, telemetri ve diğer sensör çıkışları ile çevrebirimlerle haberleşecek portları bulunmaktadır. 8 adet motor ve ayrıca çeşitli görevler için kullanılabilecek servo çıkışları ile standart kumanda alıcılarla haberleşecek girişler barındırmaktadır. Bu uçuş kontrol bilgisayarları üzerinde yine açık kaynak kodlu geliştirilen PX4 yazılımı koşmaktadır. Bu yazılım aracılığı ile uçağın kumanda ve kontrolleri yapılabilmekte rota ve görev planları yüklenerek otonom olarak uçması sağlanabilmektedir.

Şekil 12. *Örnek Bir Pixhawk Uçuş Kontrol Bilgisayarı*

3.1. Hava Araçlarında Bulunan Diğer Sensörler

Bu kısımda derste uygulaması yapılmayıp bir hava aracında yer alan diğer sensörler hakkında kısa bilgilendirme verilecektir. Eğitmen dersin süresini göz önüne alarak diğer sensörlerle ilgili verilen bilgileri öğrencilerle paylaşabilir ve üzerinde karşılıklı tartışabilir.

3.2. İrtifa Ölçerler (Altimetre)

Uçaklarda irtifa ölçümü radyo sinyalleri ve basınç farklılıklarından faydalanılarak yapılmaktadır. Belirli bir irtifa değerine ulaşana kadar uçaklar alt taraflarında yerleşik olan radyo altimetre vasıtasıyla yerden yüksekliklerini ölçer. Radyo altimetre yaydığı radyo sinyallerinin geri yansıma süresini ölçerek uçağın

yerden yüksekliğini belirler. Belirli bir yükseklikten sonra sinyaller bozulmaya uğradığı için radyo altimetre sağlıklı sonuç vermez.

İkinci ölçüm aracı ise barometredir. Uçaklar yükseldikçe hava basıncındaki değişikliği ölçerek bir önceki irtifa değerlerine göre ne kadar yükselip alçaldıklarını hesaplayabilir. Bunun için uçağın bulunduğu yerdeki yerel basınç değerlerini bilmesi gereklidir. Hava trafik kontrol görevlileri pilotlara bulundukları yerin basınç değerlerini bildirir. Pilotlar buna göre uçağın altimetresine düzeltme değeri girerek doğru irtifa değerlerini okur. Drone ve model uçaklarda da coğrafi olarak konum değiştirildiğinde barometre kalibrasyonu yapılmalıdır. Aksi takdirde İHA kendini yanlış irtifade görecektir. Model uçak ve drone larda yaygın kullanılan barametrik sensör BMP180'dir. Dersin ana uygulamasında bu sensör kullanılmayacaktır. Ancak ilave etkinliklerde bu sensör kullanılarak deneyap kart ile irtifa okuma uygulaması gerçekleştirilebilir.

3.3. Hız Ölçerler

Sabit kanatlı uçaklarda kanattaki taşıma uçağın hızı ile doğrudan ilişkilidir. Bu nedenle pilotların uçağın hızını bilmeleri uçağın STALL (taşımayı kaybetme) durumuna girmesini önlemek için çok önemlidir. Uçaklarda hızı okumak için pitot tüpü kullanılır. Uçağının önden aldığı havanın oluşturduğu dinamik basınç uçağın bulunduğu ortamdaki statik basınç ile karşılaştırılır. Böylelikle uçağın hangi hızda uçtuğu hesaplanabilir. Pitot tüpü uçak hızını okumakta önemli bir parça olduğundan tıkanmamalıdır. Bu nedenle içerisine böcek girmemesi için yerde iken pitot tüpleri kılıf ile kapatılır. Havada da içerisinin donmasını engellemek için pitot ısıtıcısı kullanılır. Tarihte pitot tüpü tıkanması sebebiyle yaşanmış uçak kazaları bulunmaktadır. Uçağın hızının okunması için bir diğer araç da GPS tır. Uçağın konumundaki zamana göre yer değiştirme hesaplanarak uçağın hızı hesaplanır.

3.4. Servo ve Aktüatörler

Uçaklar havada üç temel hareketi yaparak manevra gerçekleştirmektedir. Yunuslama, Yuvarlanma ve Sapma. Yunuslama hareketi uçağın burnunun aşağı yukarı hareket etmesi ile irtifasını değiştirmesini saplar. Yuvarlanma hareketi ise uçağın sağa ve sola yatışı ile dönme hareketi yapmasını sağlar. Sapma hareketi ise dönüşlerde uçağın burnunu yöneterek dönüşlerin koordineli ve stabil yapılmasına yardımcı olur. Ayrıca uçağın ana rotasından sapmasını önleyecek küçük burun düzeltmelerini yönetir.

3.5. Uçuş Kumanda Sistemleri

Temel uçuş kumanda yüzeyleri olarak bilinen kanatçık, yatay dengeleyici ve dümen uçağın sapma, yuvarlanma ve yunuslama hareketlerini yapmasını sağlamaktadır. Yüksek hava direnci sebebiyle bu üç temel kumanda yüzeyinin güçlü motorlarla yönetilebilmesi gereklidir. Büyük uçaklarda bu yüzeyleri yönetmek için, hidrolik aktüatörler, çelik halat ile doğrudan levye bağlantısı veya elektrik servo motorları kullanılabilmektedir.

Model uçak ve İHA'larda ağırlıklı olarak servo motorlar tercih edilmektedir. Uçuş kumanda yüzeylerinde servo motor kullanılma sebebi yüzeyin istenilen açıda sabit tutulabilmesini sağlamaktır. Kumanda yüzeyinde bir derecenin altında olabilecek bir sapma bile uçağın dengeli uçuşunu etkileyebilir. Bu nedenle servo motorlar hassas bir şekilde konumlandırılmalı ve kalibre edilmelidir. Bir model uçak üzerinde kanatçık için bir veya iki adet, yatay dengeleyici için bir adet ve dümen için bir adet olmak üzere toplamda en az üç adet kontrol yüzeyi servosu bulunur. Kanatçıkta ters açı ile hareket sağlayarak uçağın yuvarlanma hareketini yapmasını sağlar. Yatay dengeleyici servo ile aşağı yukarı hareket ettirilerek uçağın yunuslama hareketini kontrol eder. Dümen servosu koordineli bir dönüş hareketi yapılması için sapma hareketinin kontrolünde kullanılır. Genelde üç servodaki hareket her iki yönde 0-30 derece arasında gerçekleşir.

4. Uygula: Deneyap Kart ile Servo Motor Sürme

Uçuş kumanda yüzeylerinde servo hareketlerini anlamak için İHA kiti üzerinde bulunan servoları Deneyap kart tarafından kontrol ederek uçuş kontrol yüzeylerinin ne şekilde hareketler yaptığını inceleyeceğiz. Deneyap kart üzerinden vereceğimiz servo sinyal komutları ile servoyu istediğimiz açıda hareket ettirerek kontrol yüzeyini istediğimiz açıya konumlandıracağız. Servolar 20ms'lik kare dalga işaretinin darbe genişliği yönetilerek çalıştırılmaktadır. 1ms ile 2ms arasında değişen darbe genişlikleri ile servo açısı 0 ile 180 derece arasında değiştirilebilir.

Deneyap kartta bu kontrolü sağlamak için PWM (Pulse Widh Modulation) çıkışları kullanılmaktadır. Servo kablolarında kırmızı renkli +5v besleme ucu üçlü konnektörün orta kısmında bulunur. Sarı renkli kablo sinyal, siyah renkli kablo da – kutuplu toprak için kullanılır. Bu nedenle İHA kitimizdeki kontrol yüzeyi servolarından birini Deneyap kart üzerindeki 31 numaralı PWM0 çıkış pinine bağlıyoruz. +5v kablosunu Deneyap kart +5V çıkışına toprak kablosunu da GND çıkışına bağlıyoruz. Ardından aşağıda verilen kodu çalıştırarak Seri ekrandan verdiğimiz açı değerleri ile servomuzu kontrol ediyoruz. Eğitmen değişik değerler vererek servo değerine göre kontrol yüzeyinin hareketini öğrencilerin gözlemlemesini sağlar.

Not

Burada çok büyük açılar verilmesi kontrol yüzeyinin kilitli kalmasına sebep olabilir. 30 dereceyi aşmayacak şekilde değerler verilmesi kontrol yüzeyinin sağlıklı çalışmasını sağlayacaktır.

```
#include "deneyap.h"
#include "ServoESP32.h"

#define servoPin PWM0                  // PWM pini tanimlamasi
#define potentiometerPin A0             // Analog giris pini tanimlamasi

Servo servo1;                          // Class'tan nesne turetimi
int incomingByte = 0;           // Gelen veriyi tutacak degisken tanimi

void setup() {
   Serial.begin(115200);              // Seri haberlesme baslatildi
   servo1.attach(servoPin);  // Servo baglantisi yapilacak pin ile iliski kuruldu
}

void loop() {
 if (Serial.available() > 0)       // Seri terminalden veri bekleniyor
 {

   incomingByte = Serial.parseInt();  // Seri terminalden asagida belirtilen sinir degerler arasi veri girisi yapilir
   int servoPosition = map(incomingByte, 0, 150, 30, 180);
   servo1.write(servoPosition);  // Pozisyon kontrolu yapliyor
   Serial.println(servoPosition);   // Motor pozisyonuna iliskin derece bilgisi seri terminale yaziliyor
   delay(20);
 }   }
```

5. TASARLA

Tasarla ve üret bölümlerinde öğrenciler aktif rol üstlenerek verilen problemi çözer. Eğitmen sadece zorlanılan noktalarda destek olur. Eğitmen uçaklarda kulanılan kontrol yüzeyleri ve kumanda kontrol sistemlerini tekrar hatırlamalarını ister. Ardından elektrik sinyalleri ile pilotun kumanda komutlarını uçağın kontrol yüzeyine aktaracak bir sistem (fly by wire sistemi) tasarlamalarını ister.

Donanım tasarımı: Öğrencilerden öncelikle sistemin donanım bileşenlerini düşünmelerini ve bir kâğıtta şablon halinde çizmelerini ister. Bu aşamada öğrenci;

- Bir kumanda kolu,
- Kumanda kolunda manevra eksenlerini okuyacak sensör yerleşimleri,
- Sensör değerlerini okuyacak bir mikroişlemci,
- Kontrol yüzeyleri üzerindeki servolar,
- Servo ve sensör kablo yerleşimleri,

kısımlarını tasarlamalı ve çizebilmelidir.

Yazılım tasarımı: Bu aşamada öğrencilerin Deneyap Kart üzerinde koşacak program algoritmasını tasarlamaları beklenmektedir. Kumanda kolundan eksen hareket verilerinin okunacağı bir fonksiyon tasarlanmalıdır. Bu hareketlerin servolara aktarılacağı başka bir fonksiyon daha tasarlanmalıdır. Ardından bu iki fonksiyonun birbirleri ile ardışıl ve hızlı bir şekilde veri aktaracağı döngü oluşturulmalıdır.

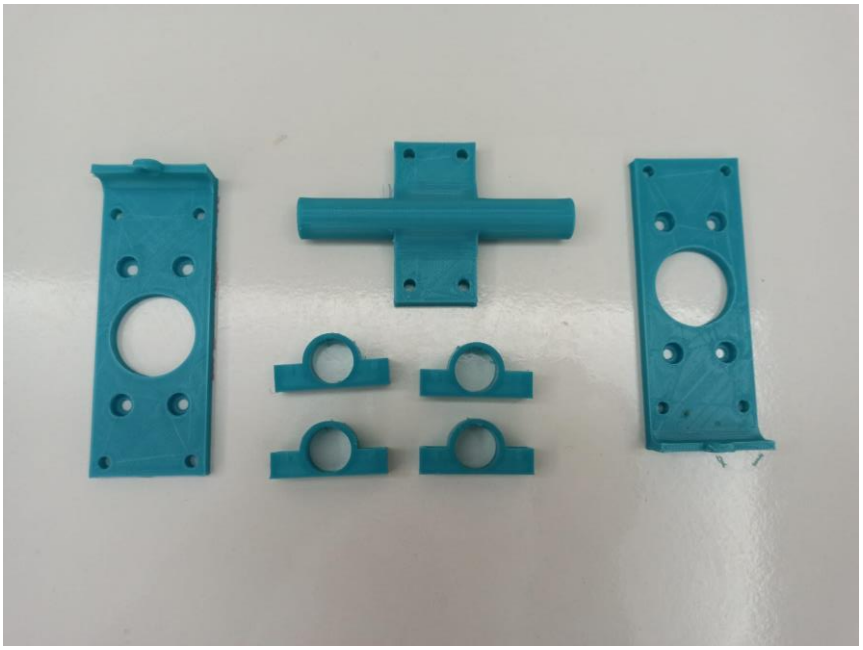
Eğitmen başta algoritmanın şematik gösterimde pseudo kod halinde tasarlanmasını ister. Ardından kullanılacak kütüphaneler ve Deneyap Kart port numaraları belirlenmelidir. Bu şekilde yazılımın kavramsal tasarımı tamamlanmış olacaktır.

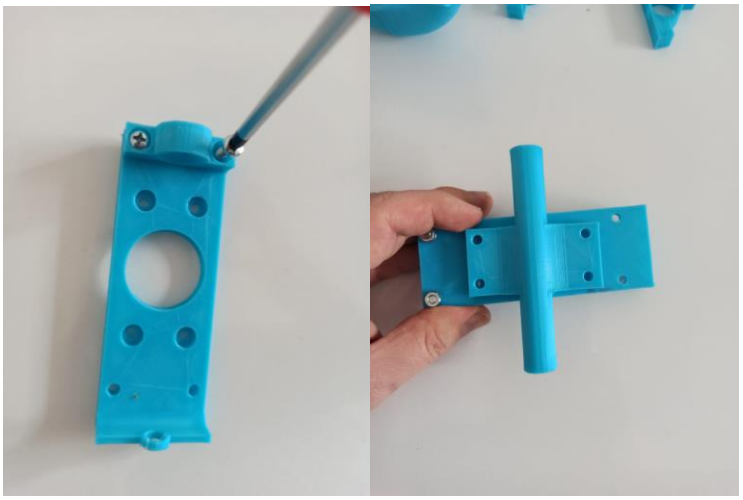
6. ÜRET

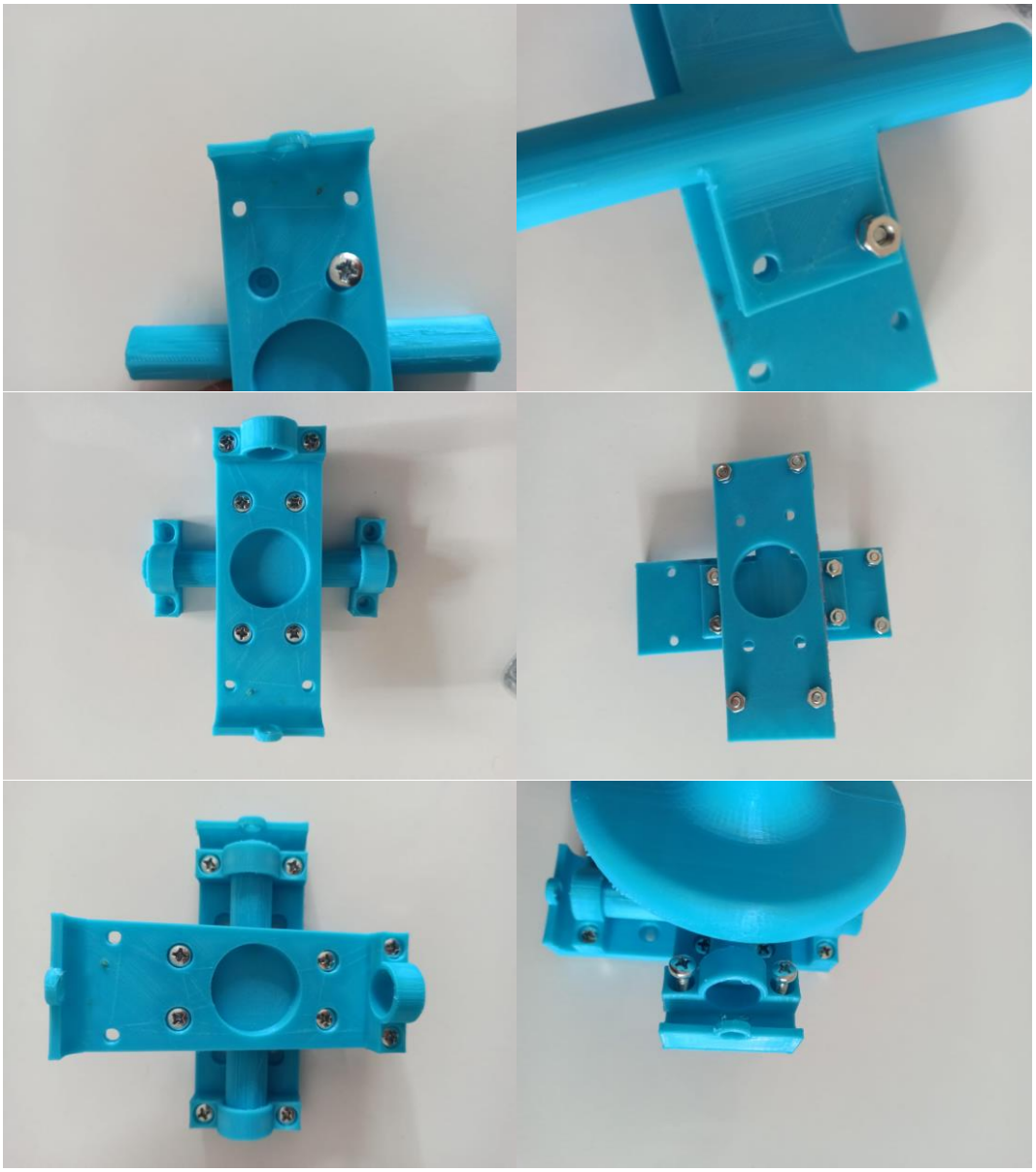
Bu bölümde öğrenciler tasarlanan fly by wire sistemini model bir uçak üzerinde uygulayacaktır. Eğitmen önceden hazırlanmış kumanda kolu 3D parçalarını öğrencilerle paylaşır. Montaj süreçlerinde yardımcı olarak öğrencilerle birlikte kumanda kolunu görsel anlatımdaki şekilde gibi kullanılır hale hazır ederler.

Kumanda kolunun iki parçalı bileşen metrik 3 vida ve somun kullanılarak birleştirilir.

Ardından görseldeki eksen mil yatakları sırasıyla vidalanarak bir araya getirilir.

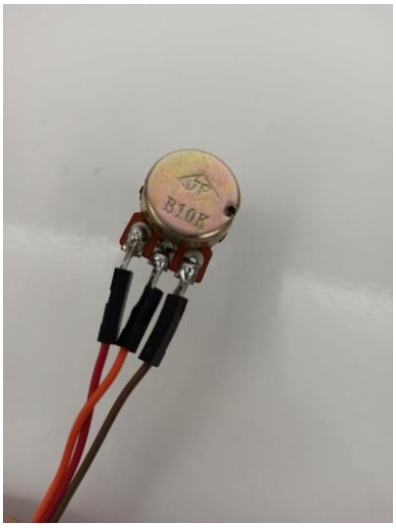




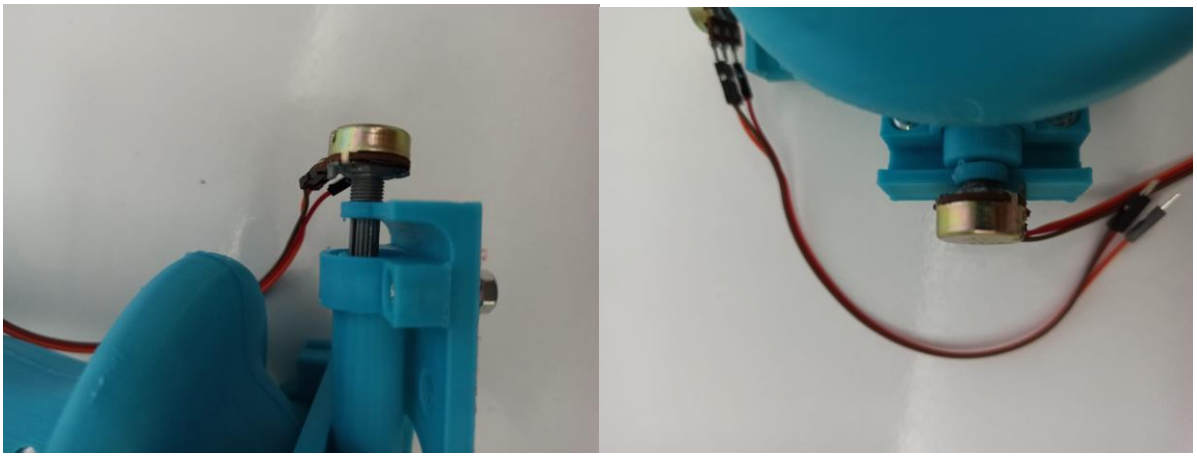


Bütün bileşenler bir araya geldiğinde kumanda kolu yapısal olarak görseldeki gibi görünmelidir.

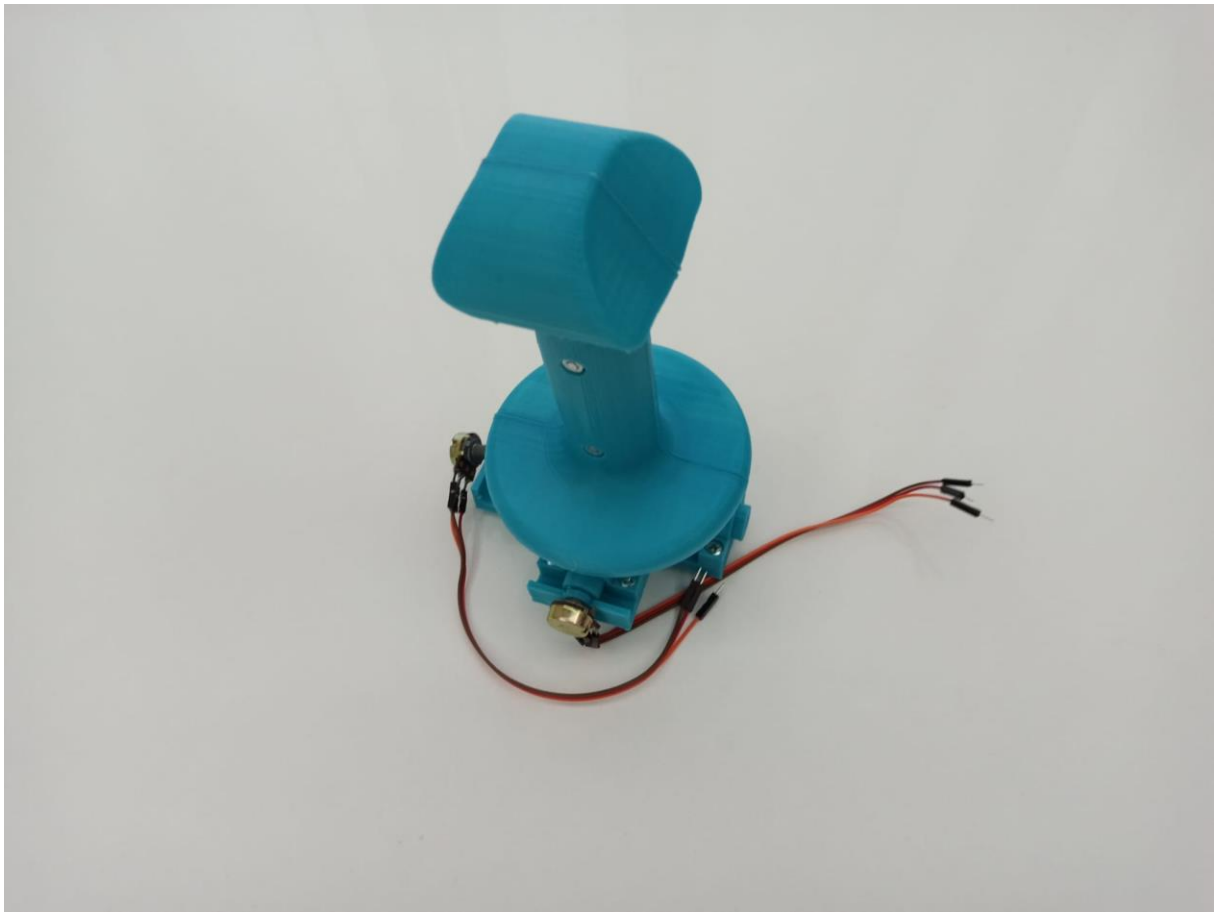
Plastik parçaları birleştirilen kumanda koluna bağlanacak potansiyometrelere Jumper kabloları lehimlenerek bağlantı için hazır hale getirilir.



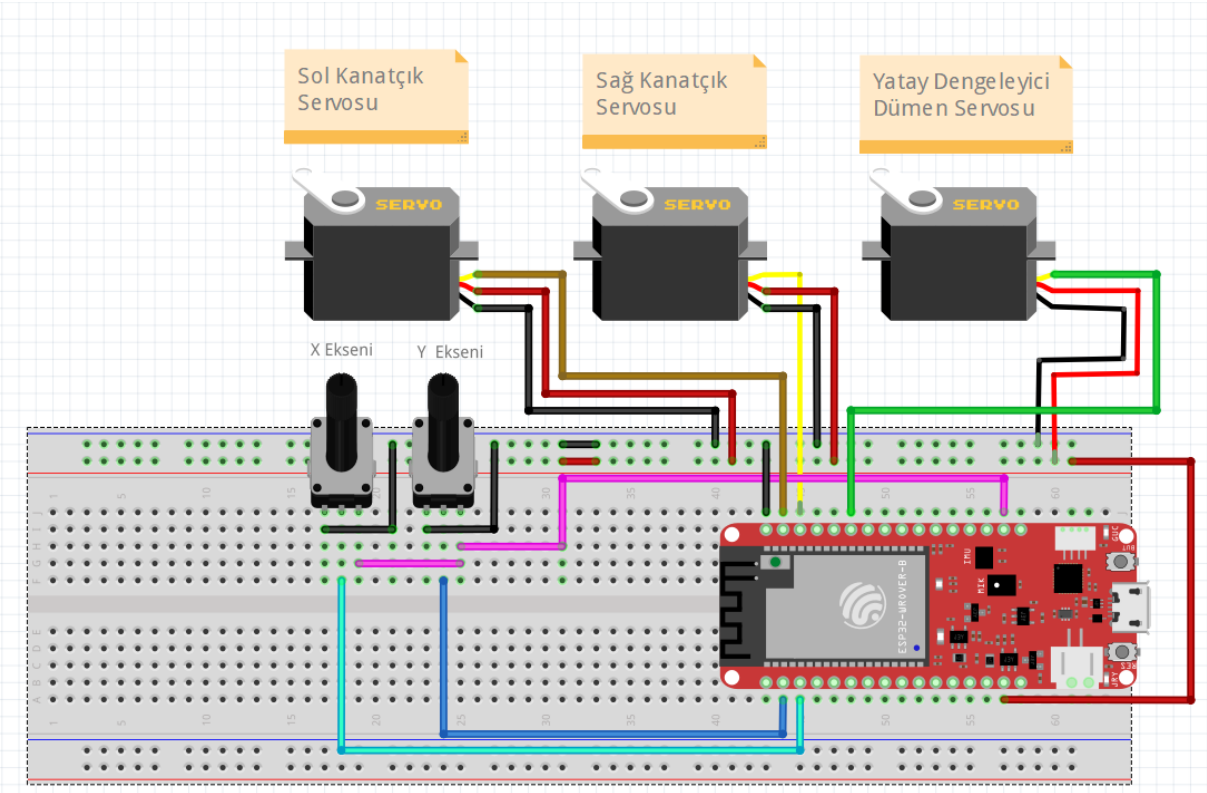
Kumanda kolunun iki eksenine denk gelecek şekilde potansiyometreler vida gibi çevrilerek yerleştirilir. Bu sırada potansiyometrelerden her iki yönde de değer okuyabilmek için kontrol ucunun orta seviyede konumlanmasına dikkat edilir.



Potansiyometreler de bağlandıktan sonra kumanda kolu görseldeki gibi deneyap karta bağlanmaya hazır hale gelmiş olur.



Sırada mikro işlemci ile sensör ve servo bağlantıları yapılır. Bu noktada eğitmen Şekil 5'te gösterilen şemada yer alan tasarımı paylaşıp öğrencilerin tasarladıkları modeller ile ne kadar benzeştiği üzerine tartışabilir.



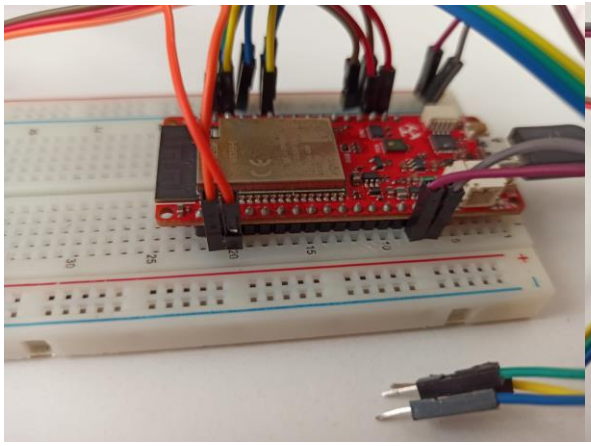
Şekil 13. *Deneyap Kart Potansiyometre ve Servo Bağlantı Şeması*

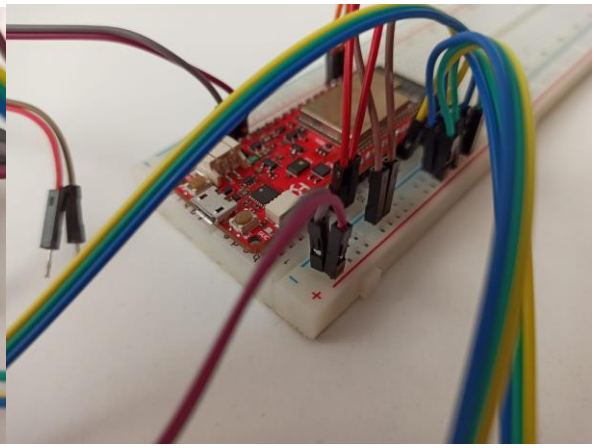
Öğrenciler şemada gösterilen devre bağlantılarını tamamladıktan sonra yazılımı hazırlama aşamasına geçer. Bu aşamada arduino ino üzerinde daha önce uygula aşamasında gördükleri potansiyometre okuma ve servo sürme kodlarını bir araya getirerek kumanda kolu hareketine göre servoları hareket ettirecek yazılımı üretirler. Bu aşamada eğitmen aşağıda verilen etkinlik kodunu doğrudan paylaşmaz. Kodlarını hazırlayan öğrencileri gözlemleyerek verilen koda benzerliğine göre yorumlarla öğrencilerin hazırladıkları kodun çalışır hale gelmesine yardımcı olur.

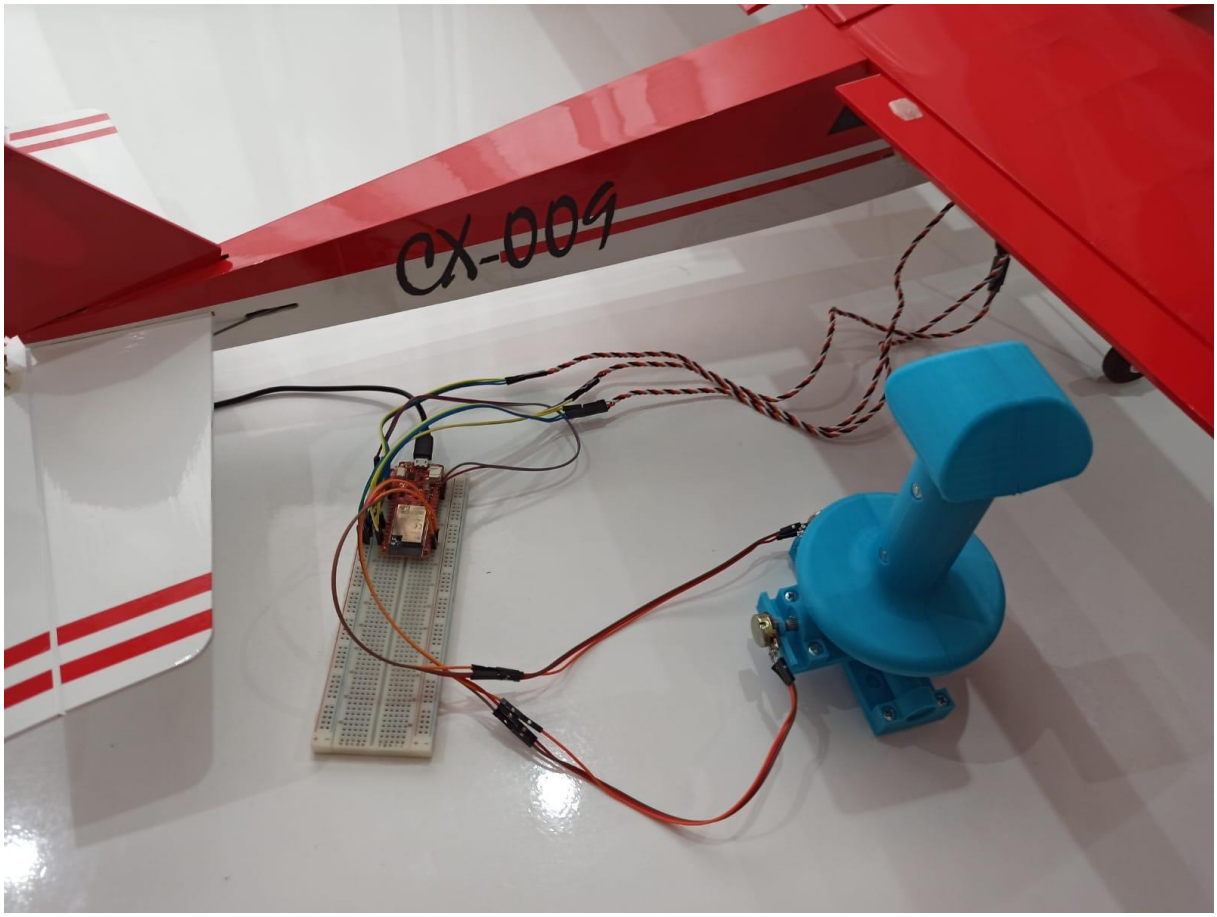
```
#include "deneyap.h"
#include "deneyap.h"
#include "ServoESP32.h"

#define servoPin PWM0
Servo aileron_sag;  // Kanatçık servosu
Servo aileron_sol;  // Kanatçık servosu
Servo elevator;  // Yatay dengeleyici servosu

int x_deger=0; //x ekseni okanacak değer değişkeni
int y_deger=0; //y ekseni okanacak değer değişkeni

void setup() {


  Serial.begin(9600);
  elevator.attach(D4);  // elevatör servosunu 27 numaralı pine bağladık
  aileron_sag.attach(PWM1);  // sağ kanatçık servosunu 30 numaralı pine bağladık
  aileron_sol.attach(PWM0); // sol kanatçık servosunu 31 numaralı pine bağladık

}
```

```
void loop() {
  x_deger = analogRead(A1); // A1 analog girişinden x ekseni değerini okuyoruz
  y_deger = analogRead(A0); // A0 analog girişinden y ekseni değerini okuyoruz
  Serial.println(y_deger);
  x_deger = map(x_deger, 2500, 3200, 0, 180);   // okuduğu x ekseni değerini uçağın servo açılarına göre ölçekliyor
  y_deger = map(y_deger, 2600, 1900, 0, 180);   // okuduğu y ekseni değerini uçağın servo açılarına göre ölçekliyor

  aileron_sag.write(x_deger);  //Sag kanatçık servosunu x_ekseni değerinde çevirir
  aileron_sol.write(x_deger);  //Sol kanatçık servosunu 180 - x_ekseni değerinde çevirir

  elevator.write(y_deger);    // yatay dengeleyici servosunu y_ekseni değerinde çevirir


  delay(15);   // servo hareketini tamamlamasını bekler
```

Öğrenciler yazılımı da hazır hale getirdikten sonra eğitmen takibinde çalıştırarak kumanda kolu ile uçak kontrol yüzeylerini kontrol etmeye başlar. Bu sırada kumanda kolu ile kontrol yüzeylerinin uyumlu çalışıp çalışmadığını kontrol ederler. Ortam ısısı ve potansiyometrenin konumlamasına bağlı olarak kumanda kolu girişlerinde beklenen çıkışlar alınamıyorsa öğrenciler kodda bulunan ölçekleme değerlerini değiştirerek kumanda kolunu kalibre edebilirler.







Not

Uygulama örnek videosuna aşağıdaki linkten ulaşılabilir

https://owncloud.tubitak.gov.tr/index.php/s/pzTAtdzksc2UCBz

7. DEĞERLENDİR

Öğrenciler fly by wire uygulamasında Deneyap karta verilen giriş ve çıkışların kontrol yüzeyini ne şekilde yönettiğini değerlendirir? Kumanda kolu ileri itildiğinde yatay dengeleyici nasıl bir hareket yaptı? Buna göre uçağın burnu hangi pozisyonu alır? Kanatçık kumanda koduna göre nasıl hareket etti? Sağ ve sol kanatçık, kanatçıkları neden birbirine ters yönlü hareket etti? Gibi sorular yönelterek eğitmen öğrencilerin kumanda kolu ile kontrol yüzeyi hareketleri arasındaki ilişkiyi kavramalarına yardımcı olur. Kumanda kolundaki hareketin kontrol yüzeyine aktarımının elektrik sinyali aracılığı ile nasıl sağlandığını açıklamalarını ister. Bunun uçakta nasıl bir kazanım sağladığını olumlu ve olumsuz yönlerini, riskli ve emniyetli taraflarını sorar. Böylelikle günümüz uçaklarında yaygın kullanılan bu teknolojiyi kavradıklarından emin olur.

8. İLAVE ETKİNLİK

Dersin işlenişine bağlı olarak zaman artması durumunda eğitmen, öğrencilerin aşağıdaki ilave etkinliklerin yapılmasını isteyebilir.

8.1. Kablosuz Kumanda Kolu Yapımı

Öğrenciler ikinci bir Deneyap kart kullanarak İHA ile kumanda kolu arasındaki bağlantıyı kablosuz hale getirebilir. Birinci Deneyap karta kumanda kolu potansiyometre girişleri yapılır. İkinci Deneyap kart üzerinden ise servo motor çıkışları alınır. Birinci Deneyap kart ile ikinci Deneyap kart arasında WiFi bağlantı kurulur. Böylelikle birinci Deneyap kart üzerinden alınan kumanda kolu bilgisi servoları yöneten Denyap karta kablosuz olarak aktarılmış olur. Gerçek bir İHA'nın yer istasyonu ve uçak arası kablosuz kontrolünü gösteren etkinlik gerçekleştirilebilir.

8.2. BMP180 Basınç Ölçer ile İrtifa Bilgisi Okuma

Deneyap kart ile BMP180 basınçölçer bağlanarak uçağın irtifa bilgisinin okunacağı bir uygulama gerçekleştirilebilir. Bunun için BMP180 I2C bağlantısı, Deneyap Kart I2C girişine yapılır. Arduino BMP180 kütüphanesi yardımıyla basınç değişimine bağlı yükseklik değeri okunabilir.

8.3. RC Kumanda Alıcı Değerlerini Okuma

Dersin üret kısmında kullanılan 3D baskılı kumanda kolu yerine RC kumanda alıcısında gelen komutlar okuyarak bir yazılım oluşturulabilir. Bu yazılım ile RC sinyalleri servoların kumandası için kullanılabilir. Böylelikle öğrenci Denyap Kart ile RC kumanda alıcı sinyallerini okuyarak uçuş kontrol kartı girişi oluşturabilir.

9. ÖRNEK PROJE ÖNERİLERİ

Bu dersten edinilen bilgi ve beceriler ile öğrenciler Deneyap kart kullanarak bir İHA uçuş kontrol kartı projesi gerçekleştirebilir. Uçuş kontrollerini sağlayacak PID kontrol algoritmalarını geliştirerek sabit kanatlı veya döner kanatlı bir İHA'nın manuel veya otonom uçuşunu yönetecek bir uçuş kontrol bilgisayarı oluşturabilirler.

Benzer şekilde bir İHA'yı yerden kontrol edebilecek bir yer kontrol istasyonu gerçekleştirilebilir. Deneyap kart WiFi özelliği veya bağlanacak bir haberleşme modülü ile İHA uçuş bilgisayarı ile Deneyap kart haberleştirilebilir. Telemetri verileri çekilebilir. Uçuş kumanda komutları İHA'ya iletilebilir.

6. HAFTA: TEMEL ROS BİLGİSİ

Ön Bilgi:

- Temel Linux ve programlama bilgisi

Haftanın Kazanımları:

- Öğrenciler temel düzeyde bir ROS paketi oluşturur.
- Öğrenciler ROS düğümlerini (node) ve konuları (topic) açıklar.
- Öğrenciler ROS mesajlarını açıklar ve bir örnek oluşturur.
- Öğrenciler ROS servislerini ve eylem kütüphanelerini tanır.
- Öğrenciler ROS başlatma (launch) dosyalarını oluşturur.
- Öğrenciler ROS paketlerini yayınlar.

Haftanın Amacı:

Bu haftanın amacı eğitim süresince kullanılacak Robotik İşletim Sistemi (ROS) kavramının tüm öğrenciler tarafından genel bir hatırlatmasının yapılmasını sağlamak ve robot programlamaya girmeden önce kodlamaların yapılacağı Linux üzerindeki ROS ara yüzünü tanıtmaktır. Ayrıca, temel ROS bileşenleri kullanılarak ROS programlamaya giriş yapılması, öğrencilerin ROS mimarisini tanıması ve basit bir algoritma oluşturabilmesi de hedeflenmektedir. Öğrencilere bir robotun belirli bir mesafeyi alabilmesi için gerekli programlama adımlarını öğretmek ve öğrencilerin ölçme, uzunluk, denklem, çap ve çevre konularını verilen problemin çözümünde kullanmalarını sağlamak bu haftanın bir diğer amacıdır.

Kullanılacak Malzemeler:

Linux işlerim sistemi üzerine ROS

Haftanın İşlenişi:

Gözle: ROS temel bileşenleri üzerine tartışma, ROS bileşenlerini tanıma ve temel ROS uygulamalarının yazıldığı ***catkin*** ara yüzünü tanıma

Uygula: Temel haberleşme/mesajlaşma, temel ROS programlama

Tasarla: İstenilen haberleşme ve mesajlaşma algoritmasını tasarlama

Üret: Bir ROS paketi programlama

Değerlendir: Haftanın içeriği ile ilgili yansıtma etkinliği ve değerlendirme

1. GÖZLE VE UYGULA

1.1. Gözle: ROS Temel Kavramları ve ROS Bileşenleri

Rehber öğretmen yönetiminde öğrencilerin ROS temel kavramları ve bileşenleri üzerinde tartışmaları sağlanır. "Robotik İşletim Sistemi (ROS) nedir?", "Robotik işletim sistemini ayıran temel özellikler nelerdir?", "ROS nerelerde kullanılır?", "Gerçek bir robot ile simülasyon için tasarlanmış bir robot arasında ne gibi farklılıklar vardır?", "Simülasyondaki robotlar hangi parçalardan oluşur?", "Catkin çalışma alanı (catkin workspace) neden gereklidir?" gibi sorularla rehber öğretmen tartışmayı yönetir. Cevaplara öğrencilerin katkı sağlaması beklenir. Rehber ROS için tanımları dikkate alarak öğrencileri yönlendirebilir veya gerektiği yerde (öğrenciler kavramı yanlış tanımladıklarında veya doğru olandan çok farklı tanım yaptıklarında) bu tanımları doğrudan kendisi yapabilir.

Robot İşletim Sistemi kısaca ROS, robot uygulamaları geliştirmek amacı ile bir dizi yazılım kütüphanesi ve bir robot geliştirme standardı oluşturmaktadır. Yeni bir robotik tasarlama ve üretme sürecinde yazılım geliştirirken üretimi tecrübe etmenize gerek yoktur. Bu yüzden ROS kullanılır ve ROS kullanma sebeplerimiz kısaca şunlardır;

- ROS genelleştirilmiş bir yapıdadır. Aynı bilgi birçok farklı robota uygulanabilir. Örneğin robot kolu, insansız kara aracı, insansız hava aracı, robotik raylı sistemler vs.
- Esnek ve globalleştirilmiş ROS paketleri: ROS için yazılmış kontrol paketleri global ve esnek yapıda olduğundan çok kolay bir şekilde değiştirilebilmektedir.
- Esnek programlama dilleri: ROS esas olarak C++ ve Python dilleri referans alınarak geliştirilmiştir. Böylece bu dilden biri ile veya her ikisini de kullanarak robotik kodlama yapabilirsiniz.
- ROS farklı simülasyon araçlarına sahiptir. ROS'da geliştirilen robotlar RVIZ ve GAZEBO gibi simülasyon ortamlarında gerçek fiziksel şartlar için görselleştirilebilmektedir.
- ROS ile birden fazla robotu kontrol edebilirsiniz. Aynı anda hem bir insansız aracı hem de üzerindeki bir robot kolunu kontrol edebilirsiniz.
- ROS sistem çekirdeğinde fazla yer kaplamaz ve kaynak tüketmez. Raspberry Pi gibi düşük kaynağa sahip tek kartlı bilgisayarlar üzerinde bile çalışabilir.
- ROS çok geniş geliştirici ve uyumlu bileşenlere sahiptir. Böylece hem yaygınlaştırılması sağlanmış olmakta hem de sorunlar hızlı çözülmektedir.
- ROS açık kaynaklı bir projedir. Herhangi bir lisans ücreti ödemeden kullanabilir ve geliştirilmesine katkı sağlayabilirsiniz.

1.2. Gözle: ROS Kurulumu ve ROS Paketleri

Aynı anda yayında olan desteklenen birden fazla ROS dağıtımı vardır. Bazı ROS dağıtımları uzun süreli desteğe sahip sürümlerdir, bu da onları daha kararlı hale getirirken, diğerleri daha kısa süreli destek olmaları nedeni ile daha yenidir. ROS dağıtımları ile ilgili olarak ayrıntı için ROS Dağıtımlar sayfasını bakabilirsiniz.

Not

Temel ROS Dağıtımlarına aşağıdaki bağlantıdan ulaşılabilirsiniz.
i) http://wiki.ros.org/Distributions

ROS Ubuntu işletim sistemi üzerinde resmi olarak desteklenmektedir. Bunun yanında diğer işletim sistemleri için de tecrübe edilmiştir. ROS kurulumları için aşağıda verilen resmi kurulum kılavuzu bağlantılarını takip ederek kullanabilirsiniz. Ya da hazır bir sanal makinede kurulmuş versiyonunu indirebilirsiniz. ROS kurulu hazır sanal makineyi indirmek için http://github.com/brgokce/deneyap/ adresini ziyaret ediniz. Bunun dışında herhangi bir ROS kurulumu olmadan sıfırdan ROS öğrenmek için oluşturulmuş bir platform olan https://www.theconstructsim.com sitesini ziyaret ederek ücretsiz bir hesap oluşturup hemen kullanmaya başlayabilirsiniz. Bu platformda giriş seviyesinden ileri seviyeye kadar, temel robot teorisinden ROS tabanlı robot programlamaya kadar her şeyi uygulamalı olarak yapabilirsiniz.

Not

Temel ROS dağıtımlarına ve kurulumlarına aşağıdaki yollarla ulaşılabilir:
i) http://wiki.ros.org/ROS/Installation
ii) http://wiki.ros.org/Installation
iii) http://wiki.ros.org/Installation/Ubuntu
iv) https://www.theconstructsim.com
v) http://github.com/brgokce/deneyap/

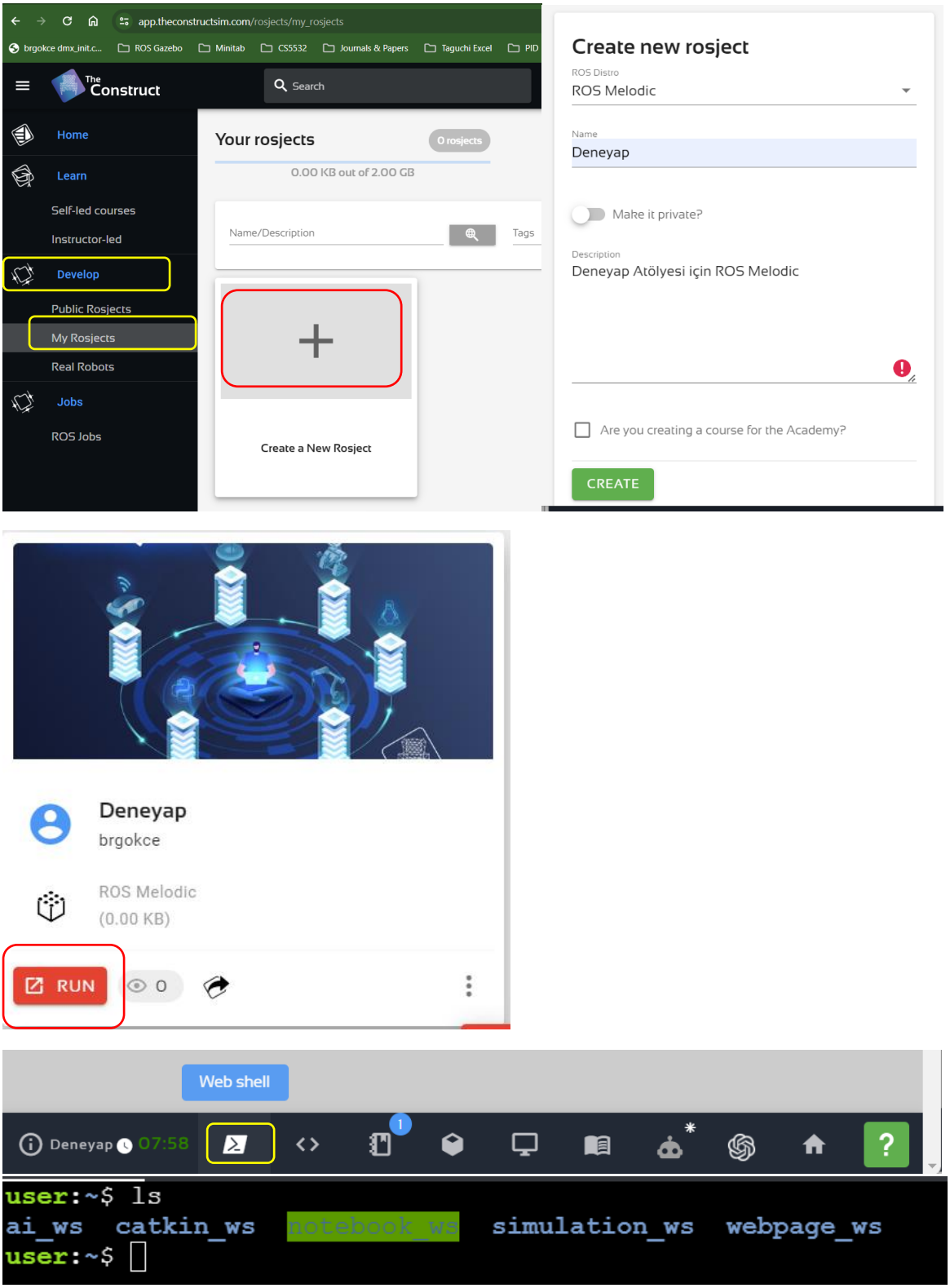
ROS paketleri, robotik İşletim sisteminin en temel bileşenidir. Her kullanıcı bir ROS paketi geliştirebilir. Bir ROS paketi oluşturmak için **catkin** adı verilen bir sistem kullanılır. Catkin adı verilen bu sistem, çalıştırılabilir uygulama veya kütüphane gibi temel hedef çıktılar üretmek için geliştiricinin ham kaynak kodundan bir yapı inşa eder. Rosbuild'de zamanla ortaya çıkan eksiklikler ve hatalar nedeni ile CMake yapısını kullanan catkin sistemi kullanılmaya başlandı.

Bir ROS paketi oluşturmanın ve çalışmanın en temek şartı bir catkin çalışma alanının oluşturulmasıdır. ROS, Linux işletim sistemine kurulduktan sonra ister projenize uygun (deneyap_ws) veya isterseniz sıklıkla kullanılan catkin_ws adında bir çalıma alanı oluşturulabilirsiniz.

1.3. Uygula: TheConstructSim Üzerinde ROS Paketi Oluşturuyorum

Theconstructsim.com sitesini ziyaret ederek site içerisinde “SIGN UP” diyerek bir kullanıcı oluşturun veya hazır epostalarınız üzerinden bir oturum açın. Şekil 1’de theconstructsim sistemi üzerindeki kullanıcı işlemleri gösterilmektedir. Soldaki menü üzerinden “Develop” daha sonra “My Rosjects” diyerek yeni bir ROS projesi açın. Burada ROS dağıtım versiyonlarından Melodic veya Noetic’i seçip “CREATE” diyerek bir proje oluşturabilirsiniz. Proje oluşturulduktan sonra “RUN” diyerek projenize ait bir linux işletim sistemi arayüzü gelecektir. Daha sonra alt simgelerden “Web Shell” simgesine

tıklayarak bir linux terminal ekranı açılacaktır. Burada catkin_ws çalışma alanı oluşturulmuş ve tüm yapılandırmalar tamamlanmıştır. Sizden sadece kullanmanız istenir.



Şekil 1. *theconstructsim.com Sistemindeki catkin_ws Çalışma Alanı*

Eğer bilgisayara kurulu bir linux işletim sistemi üzerinde ROS kurulumu gerçekleştirildikten sonra bir catkin çalışma alanı oluşturmak istiyorsanız, önce linux terminal ara yüzü açılır ve temel ROS

komutları sırası ile uygulanarak gerekli ayarlamalar ve ön kurulumlar yapılır. Bu işlemleri aynı zamanda Theconstructsim.com sitesindeki projeniz içinde yapabilirsiniz. Aşağıdaki komutlar ile dizin listelemesi ve **deneyap_ws/src** adında iç-içe iki klasör oluşturulmaktadır.

```
$ ls
```

```
$ mkdir -p ~/deneyap_ws/src
```

ROS uygulamaları ve ROS fonksiyonlarını kullanmak için çalışma alanının sıklıkla derlenmesi gerekmektedir. Bunun için ROS ortamının linux komut satırına tanıtılması (kaynaklanması) kullanım pratikliği açısından önemlidir.

```
$ source /opt/ros/melodic/setup.bash
```

Bu komutu bash başlangıç dosyasına (~ /.bashrc) eklemediğiniz sürece, bu komutu ROS komutlarına erişebilmek için açtığınız her yeni kabukta (shell) çalıştırmanız gerekecektir. Aşağıdaki komutlar ile oluşturulan **deneyap_ws/src** klasörüne geçiş yapın.

```
$ cd ~/deneyap_ws/src
```

Src klasörü altında iken **deneyap_ws** klasörü catkin çalışma alanı ROS sistemine tanıtılmalıdır.

```
$ catkin_init_workspace
```

ROS üzerinde tanımlanmış bir paket olmasa bile çalışma alanı oluşturulmalıdır. Linux komutları kullanılarak çalışma alanı klasörüne geçiş yapabilirsiniz.

```
$ cd ~/deneyap_ws
```

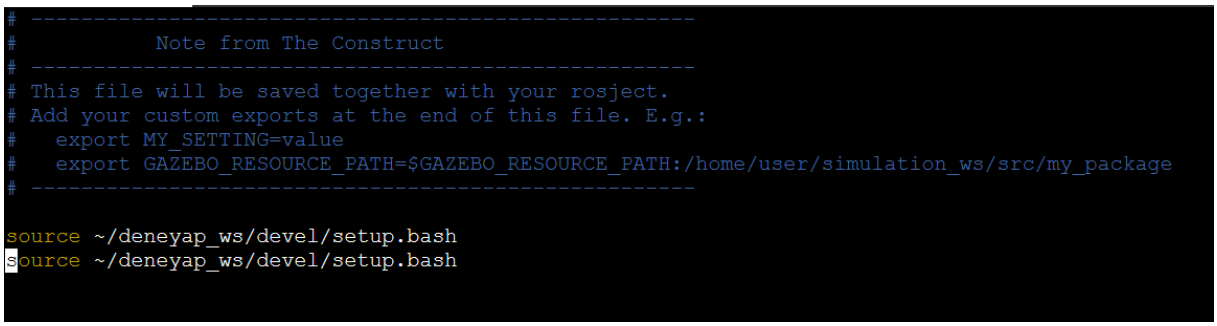
catkin_make komutu ile çalışma alanının ilk derlemesi ve yapılandırılması oluşturulur.

```
$ catkin_make
```

catkin_make, komutu oluşturulmuş olan çalışma alanında bir sistem geliştirme ve bir derleme dizini oluşturur. “devel” klasörü hedef derlemelerin kurulmadan önceki yerleştirildiği yerdir ve içerisinde farklı kurulum dosyaları bulunur. Şimdi bu çalışma alanının kaynaklanması yani komut satırı kabuğunda tanıtılmasını yapabiliriz.

```
$ echo "source ~/deneyap_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
$ sudo vim ~/.bashrc
```

Şekil 2’de yukarıda yapılan işlem sonrasında “vim” görüntüleme arayüzünde görüntülenmektedir. Şekilde görülen ve “source …” ile başlayan satırlar 157’inci satırdan sonra gelmektedir. Sayfada aşağıya doğru inmek için aşağı ok tuşlarını kullanabilirsiniz. Herhangi bir değişiklik yapmadan kabuk (Shell) ekranını kapatabilirsiniz.



Şekil 2. *.bashrc Dosyasındaki Kaynak Oluşturma Komut Satıları görülmektedir.*

Not

Vim arayüzü kullanımı çok zor olabilir. Bunun yerine nano veya gedit kullanılabilir ancak sisteme gerekli izinler alınarak yüklenmesi gerekmektedir.

Catkin çalışma alanı ile ilgili temel ayarları yaptıktan sonra ROS paketlerini oluşturulabilirsiniz. catkin_create_pkg komutu, bir ROS paketi oluşturmak için kullanılır. Yeni bir komut satırı ekranı (shell) açarak aşağıdaki komutları yazalım.

```
$ cd ~/deneyap_ws/src
$ catkin_create_pkg ilk_paket std_msgs roscpp rospy actionlib actionlib_msgs
```

catkin_create_pkg komutu ile ROS içerisinde bir paket oluştururken kullanılan std_msgs, rospy, roscpp, actionlib, actionlib_msgs komut parametreleri bir paket bağlılığıdır. Bu bağlılıklar oluşturulan paket için gerekli ve kullanılacak ROS bağlılıklarını içermektedir.

std_msgs: Temel veri türlerini ve çoklu diziler gibi diğer temel mesaj yapılarını temsil eden ortak ve mesaj türlerini içeren Standart ROS Mesajlarıdır.

roscpp: ROS'un C++ uygulamasıdır. ROS konuları, hizmetleri, parametreleri vb. ile ROS düğümleri oluşturmak için C++ geliştiricilerine API'ler sağlayan bir ROS kütüphanesidir. Bu bağlılığı C++ tabanlı düğüm yazılacağı için kullanıyoruz.

actionlib: actionlib meta paketi, ROS düğümlerinde öncelikli görevler oluşturmak için ara yüzler sağlar. Bu pakette actionlib tabanlı düğümler oluştururuz. Bu yüzden ROS düğümlerini oluşturmak için bu paketi eklemeliyiz.

actionlib_msgs: Bu paket, eylem (action server) sunucusu ve eylem istemcisi (action client) ile etkileşim kurmak için gereken standart mesaj tanımlarını içerir.

Şekil 3'de bir paket için oluşturulan dosyalar ve mesajlar gösterilmektedir. Paket oluşturulduktan sonra, CMakeLists.txt ve package.xml dosyalarını düzenleyerek ek bağımlılıkları manuel olarak ekleyebilirsiniz. Paket başarıyla oluşturulduysa aşağıdaki mesajı alırız:

```
user:~$ cd ~/deneyap_ws/src
user:~/deneyap_ws/src$ catkin_create_pkg ilk_paket std_msgs roscpp rospy actionlib actionlib_msgs
Created file ilk_paket/package.xml
Created file ilk_paket/CMakeLists.txt
Created folder ilk_paket/include/ilk_paket
Created folder ilk_paket/src
Successfully created files in /home/user/deneyap_ws/src/ilk_paket. Please adjust the values in pac
user:~/deneyap_ws/src$
```

Şekil 3.*Catking ROS Paketi Oluşturma*

catkin_make komutu kullanılarak oluşturulan paket derlenir. Bu komut, mutlaka catkin çalışma alanı ana dizini içerisinde gerçekleştirilmelidir. Şekil 4'de derleme işlemi ve sonucu gösterilmektedir.

```
$ cd ..
$ catkin_make
```

```
user:~/deneyap_ws/src$ cd ..
user:~/deneyap_ws$ catkin_make
Base path: /home/user/deneyap_ws
Source space: /home/user/deneyap_ws/src
Build space: /home/user/deneyap_ws/build
Devel space: /home/user/deneyap_ws/devel
Install space: /home/user/deneyap_ws/install
```

Şekil 4. *Catkin Çalışma Alanını Derleme*

Eğer catkin_make başarılı bir şekilde yapıldıysa derleme başarılı demektir ve terminal ekranı şekil 5'teki gibi bir çıktı verir.

```
WARNING: package "moveit_opw_kinematics_plugin" should not depend on metapa
ckage "moveit_resources" but on its packages instead
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- ~~  traversing 1 packages in topological order:
-- ~~  - ilk_paket
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- +++ processing catkin package: 'ilk_paket'
-- ==> add_subdirectory(ilk_paket)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/deneyap_ws/build
####
#### Running command: "make -j8 -l8" in "/home/user/deneyap_ws/build"
####
user:~/deneyap_ws$
```

Şekil 5. *Catkin Çalışma Alanı Derleme Sonucu*

1.4. Gözle: ROS Düğümleri (ROS Nodes)

ROS wiki'ye göre bir ROS düğümü, temelde birçok hesaplamaları ve işlemleri gerçekleştiren bir program veya kod dizisidir. Yani uygulamanın içinde çalışan yürütülebilir bir programdır. Bir uygulama için birçok düğüm yazılır ve bunlar paketlere projenin parçası olarak eklenir.

Düğümlerin grafiksel gösterimleri vardır ve bir akış grafiğinde görsel olarak birleştirilir. Akış grafiği üzerinde ROS konuları, hizmetleri, eylemleri vb. temel bileşenler de gösterilebilir. ROS programlamada dikkat edilmesi gereken en önemli konulardan biri iki düğümün aynı isme sahip olmamasıdır. ROS içerisinde birden fazla düğüm aynı anda çalıştırılmak istendiğinde ön ek veya son ek gibi tanımlamalar eklenerek düğümler çoğaltılabilir. Düğümler kod karmaşıklığını azaltır. Uygulama paketlere ve düğümlere doğru şekilde bölünürse, uygulamayı taşımak ve ölçeklemek çok daha kolay olmaktadır. Düğümler yalnızca ROS aracılığıyla iletişim kurduğundan, düğümlere doğrudan erişim yoktur. Bu yüzden bir düğüm cevap vermez veya çökerse, diğer düğümler bunlardan olumsuz etkilenmez, sadece haberleşmesi kesilir.

1.5. Gözle: ROS Konu Başlıkları (ROS Topics)

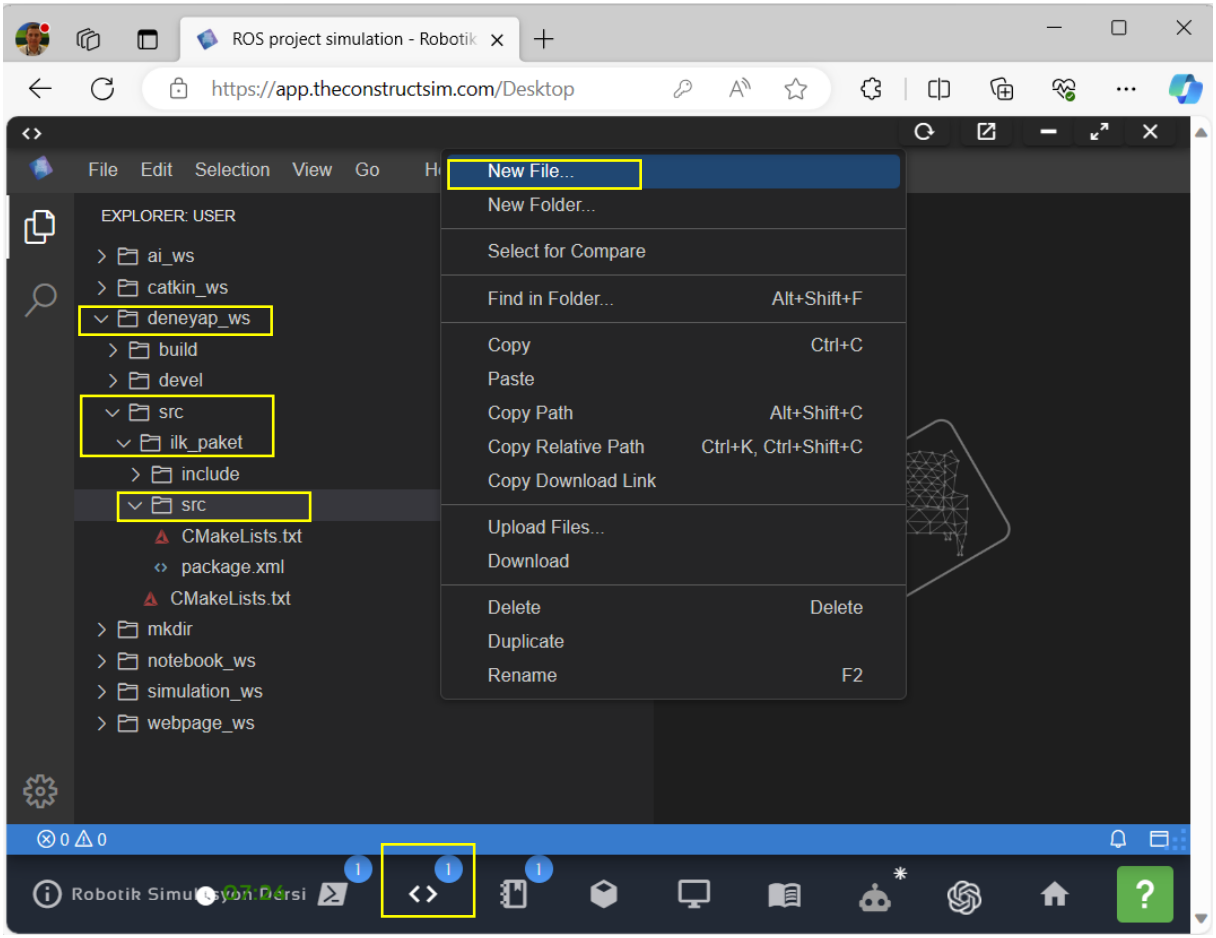
ROS'da konu başlıkları düğümler arasında haberleşmek için kullanılmaktadır. Bunun için ROS'da konu başlığı (topic) yönetiminin nasıl çalıştığının bilinmesi gerekmektedir. ROS'da bir düğüm tarafından belirli konu başlığı altında bilgi yayınlaması yapılır. Yine diğer bir düğüm tarafından yayınlanan ilgili bilgi aynı konu başlığına abone olunarak bilgi alınır.

Not

Temel ROS komutlarına aşağıdaki yollarla ulaşılabilir:
i) http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes
ii) http://wiki.ros.org/ROS/Tutorials/CreatingPackage
iii) http://github.com/brgokce/deneyap/

1.5. Uygula: ROS Konu Başlıkları (ROS Topics) Oluşturuyorum

Theconstructsim.com uygulamasına eriştikten sonra ilgili ROS projenizi "RUN" diyerek çalıştırın. Açık erişim lisansa sahip olan Visual Studio Code yazılımı sistem üzerinde kurulu gelmektedir. Visual Studio Code ara yüzünü kullanarak **ilk_paket** klasörü içerisindeki src klasörü içerisine **ilk_dugum.cpp** adında bir dosya oluşturun ve aşağıdaki kod dizesini yazın. Böylece çalışmalar catkin çalışma alanı içerisine kaydedilir. Düğümler arası haberleşme için **/src** alt dizinine yeni düğümler (nodes) ekleyerek örnek uygulamalar yapabilirsiniz. Bunun için çalışma alanı içerisindeki **deneyap_ws/src/ilk_paket/src** klasörü içerisine ***.cpp** dosyalarını oluşturabilirsiniz. Şekil 6'da Visual Studio Code kullanarak **ilk_paket** klasörü içerisindeki "src" dizini altına bir **ilk_dugum.cpp** dosyası ekleme işlemi gösterilmiştir. Dosya ekleme işlemi tamamlandıktan sonra aşağıdaki kodları kopyalayıp bu ***.cpp** dosyası içerisine yapıştırın.



Şekil 6. *Paket İçerisindeki "src" Dizini Altında bir C++ Dosyası Ekleme*

```
#include "ros/ros.h"
int main(int argc, char **argv)
{
	ros::init(argc, argv, "MERHABA_DUGUMU");
	ros::NodeHandle nh; // tanımlanmazsa zamanlama hatası döndürür.
	ros::Rate loop_rate(10);
	int sayi = 0;
	while (ros::ok()) // Kullanıcı Ctrl + C tuşlarına basana kadar döngüyü sürdür
	{
		ROS_INFO_STREAM("Merhaba Dunya : " << sayi);
		ros::spinOnce(); // ROS'un gelen mesajları işlemesine izin ver
		loop_rate.sleep(); // Döngünün geri kalanı için uykuda kalır
		sayi++;
	}
	return 0;
}
```

Bu yazılım kodunu derlemek için paketteki CMakeLists.txt dosyasının düzenlenmesi gerekmektedir. Çalışma alanı içerisindeki "deneyap_ws" altındaki ilk_paket klasörü içerisindeki CMakeLists.txt dosyasını görüntüleyerek aşağıdaki düzenlemeleri yapın. Burada sadece 138'inci satırdaki "add_executable" ve 151'inci satırdaki "target_link_libraries" komutları düzenlenecek.

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(ilk_paket)

## Find catkin macros and libraries
find_package(catkin REQUIRED COMPONENTS actionlib actionlib_msgs roscpp rospy std_msgs)

## Declare catkin package
catkin_package()

## Specify additional locations of header files
include_directories(${catkin_INCLUDE_DIRS})

## Declare a C++ executable
add_executable(MERHABA_DUGUMU src/ilk_dugum.cpp)

## Specify libraries to link a library or executable target against
target_link_libraries(MERHABA_DUGUMU ${catkin_LIBRARIES})
```

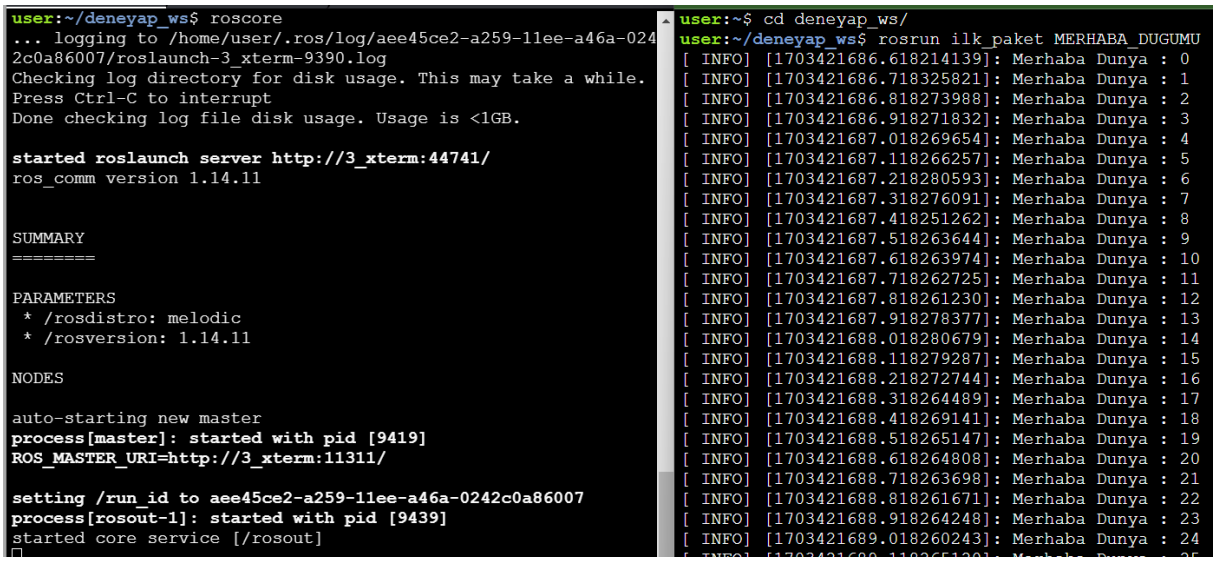
Daha sonra ana çalışma alanı olan "deneyap_ws" içerisinde **catkin_make** komutunu yazarak derleme işlemini yapın ve daha sonra sırası ile **roscore** yazın ve çalıştırın. Ayrı bir terminal açın ve "**rosrun ilk_paket MERHABA_DUGUMU"** komutunu yazarak paketteki düğümü çalıştırın. Kodlar aşağıda verilmiştir sırası ile uygulayabilirsiniz.

```
$ cd ~/deneyap_ws/
$ catkin_make
$ roscore
```

Ayrı bir terminal açılarak aşağıdaki kod yazılır.

```
$ rosrun ilk_paket MERHABA_DUGUMU
```

Şekil 7'de iki ayrı terminal ekranı görülmektedir. Bu ekranlardan solda olanda roscore çalıştırdığımız ROSMASTER ve sağda ise oluşturduğumuz pakete ait olan "MERHABA_DUGUMU" gösterilmektedir. Eğer yaptığınız kodlamalar doğru ise aşağıdaki gibi bir ekran görüntüsü alacaksınız.



Şekil 7. *Catkin Çalışma Alanının Derlenmesi, Roscore, ilk_paket ve merhaba_dunya Düğümünün Çalıştırılması*

Daha sonra ilk_paket klasörü içerisinde "**konu_yayinla_publisher.cpp**" adında bir dosya oluşturun ve aşağıdaki komut dizesini bu dosyaya ekleyin.

konu_yayinla_publisher.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

int main(int argc, char **argv)
{
   //bir_konu_yayinla bir adıyla ROS düğümü başlatılır.
   ros::init(argc, argv,"bir_konu_yayinla_publisher");

   //Bir düğüm tanıtıcı nesnesi oluşturulur
   ros::NodeHandle dugum_nesnesi;

   //Bir yayıncı nesnesi oluşturulur
   ros::Publisher sayi_nesnesi;
   sayi_nesnesi = dugum_nesnesi.advertise<std_msgs::Int32>("/SaymaDegeri",10);

   //Bir hız nesnesi oluşturulur
   ros::Rate loop_rate(10);

   //sayi adında bir değişken tanımlanır
   int sayi = 0;

   //Sayıyı artırmak ve konuyu yayınlamak için while döngüsü oluşturulur
   while (ros::ok())
   {
      std_msgs::Int32 msg;

      msg.data = sayi;

      //Mesaj verisi ekrana yazdırılır
      ROS_INFO("Gonderilen Deger: %d",msg.data);

      //konu başlığı yayınlanır.
      sayi_nesnesi.publish(msg);

      //Tüm işlemi bir kez yapmak için en az bir kez işlem döndürülür
      ros::spinOnce();

      //Sleeping for sometime
      loop_rate.sleep();

      //sayi degeri arttırılır
      ++sayi;
   }
return 0;
}
```

Daha sonra yine **ilk_paket** klasörü altındaki src klasörünün içine "**konuya_abone_ol_subscriber.cpp**" adında bir dosya oluşturun ve aşağıdaki komut dizesini ekleyin.

konuya_abone_ol_subscriber.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

//Konu başlığı için çağrılan fonksiyon
void abone_bildirim_fonksiyonu(const std_msgs::Int32::ConstPtr& msg)
{
   ROS_INFO("Alinan Deger: [%d]",msg->data);
}
```

```
int main(int argc, char **argv)
{
   //Konuya abone ol adıyla bir ROS düğümü başlatılır
   ros::init(argc, argv,"konuya_abone_ol_subscriber");
   //Bir düğüm tanıtıcı nesnesi oluşturulur
   ros::NodeHandle dugum_nesnesi;
   //Bir yayınlama nesnesi oluşturulur
   ros::Subscriber abone_ol;
   abone_ol = dugum_nesnesi.subscribe("/SaymaDegeri",10,abone_bildirim_fonksiyonu);
   //Düğümün tekrarlı çevrimi sağlanır
   ros::spin();
   return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(ilk_paket)

## Find catkin macros and libraries
find_package(catkin REQUIRED COMPONENTS actionlib actionlib_msgs roscpp rospy std_msgs)

## Declare catkin package
catkin_package()

## Specify additional locations of header files
include_directories(${catkin_INCLUDE_DIRS})

## Declare a C++ executable
add_executable(merhaba_dunya src/ilk_dugum.cpp)
add_executable(konu_yayinla_publisher src/konu_yayinla_publisher.cpp)
add_executable(konuya_abone_ol_subscriber src/konuya_abone_ol_subscriber.cpp)

## Specify libraries to link a library or executable target against
target_link_libraries(merhaba_dunya ${catkin_LIBRARIES})
target_link_libraries(konu_yayinla_publisher ${catkin_LIBRARIES})
target_link_libraries(konuya_abone_ol_subscriber ${catkin_LIBRARIES})
```

$ cd ~/deneyap_ws/
$ catkin_make
$ roscore

Yeni bir terminal açın ve aşağıdaki kodu uygulayın.
$ rosrun ilk_paket konuya_abone_ol_subscriber

Yeni bir terminal açın ve aşağıdaki kodu uygulayın.
$ rosrun ilk_paket konu_yayinla_publisher

Şekil 8'de iki ayrı düğümün çalıştığı terminal ekranları verilmektedir. Burada konu yayınlayan ve konuyu dinleyen iki düğümün de ayrı ayrı çalıştığını ve bir mesaj aktarım (sayma değeri) işleminin olduğunu göreceksiniz.

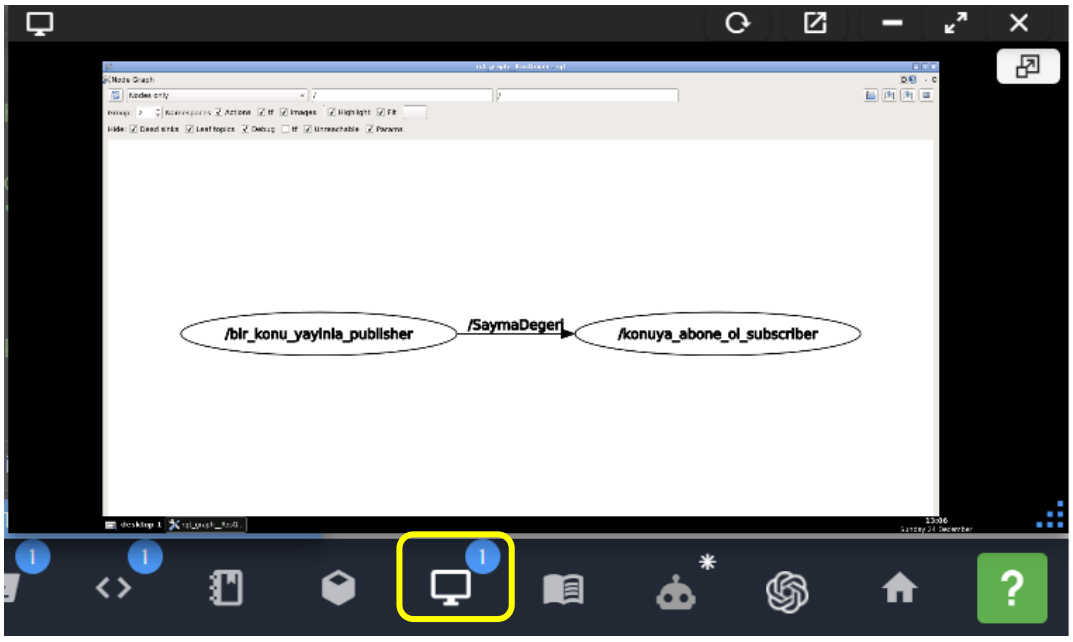
```
user:~$ rosrun ilk_paket konu_yayinla_publisher
[ INFO] [1703422774.531908560]: Gonderilen Deger: 0
[ INFO] [1703422774.631999014]: Gonderilen Deger: 1
[ INFO] [1703422774.731983867]: Gonderilen Deger: 2
[ INFO] [1703422774.831958005]: Gonderilen Deger: 3
[ INFO] [1703422774.931964597]: Gonderilen Deger: 4
[ INFO] [1703422775.031959813]: Gonderilen Deger: 5
[ INFO] [1703422775.131955832]: Gonderilen Deger: 6
[ INFO] [1703422775.231956411]: Gonderilen Deger: 7
[ INFO] [1703422775.331953765]: Gonderilen Deger: 8
[ INFO] [1703422775.431952469]: Gonderilen Deger: 9
[ INFO] [1703422775.531960016]: Gonderilen Deger: 10
[ INFO] [1703422775.631951948]: Gonderilen Deger: 11
[ INFO] [1703422775.731956834]: Gonderilen Deger: 12
user:~/deneyap_ws$ rosrun ilk_paket konuya_abone_ol_subscriber
[ INFO] [1703422774.832346538]: Alinan Deger: [3]
[ INFO] [1703422774.932272218]: Alinan Deger: [4]
[ INFO] [1703422775.032244991]: Alinan Deger: [5]
[ INFO] [1703422775.132241812]: Alinan Deger: [6]
[ INFO] [1703422775.232258193]: Alinan Deger: [7]
[ INFO] [1703422775.332206788]: Alinan Deger: [8]
[ INFO] [1703422775.432229681]: Alinan Deger: [9]
[ INFO] [1703422775.532254507]: Alinan Deger: [10]
[ INFO] [1703422775.632230574]: Alinan Deger: [11]
[ INFO] [1703422775.732254770]: Alinan Deger: [12]
```

Şekil 8. *Catkin Çalışma Alanı Derlendikten Sonra Her İki Düğümün Çalıştırılması*

Çalışan terminalleri kapatmadan yeni bir terminal açalım ve aşağıdaki komutu yazalım

```
$ rosrun rqt_graph rqt_graph
```

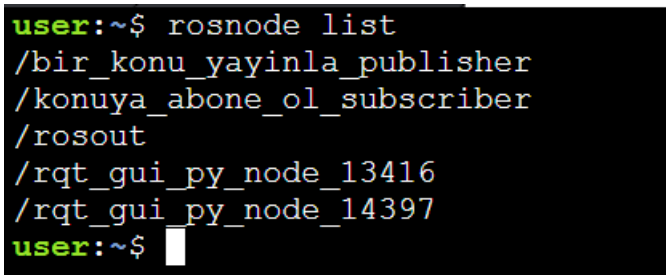
Şekil 9'da iki ayrı düğümün çalıştığı sırada iki düğümün hangi konu başlığı üzerinden haberleştiğini gösteren "RQT" grafiği verilmektedir. Bu grafik sayfanın altı kısmındaki sarı çerçeve ile gösterilen "Graphical tool" sekmesi altındadır.



Şekil 9. *ROS Düğümlerinin Birbirleri ile Bağlantısının Grafiksel Gösterimi*

ROS ortamında düğümler ile ilgili daha fazla bilgi almak ve hataları gözlemlemek için rosnode ve rostopic komutları ve parametrelerini kullanılabilirsiniz. Şekil 10'da ROSMASTER'da çalışan düğümlerin listesi verilmiştir. Burada sadece düğümler verilmektedir. Düğümlerin kullandığı mesajlar (topic) ise farklı bir komut ile listelenmektedir.

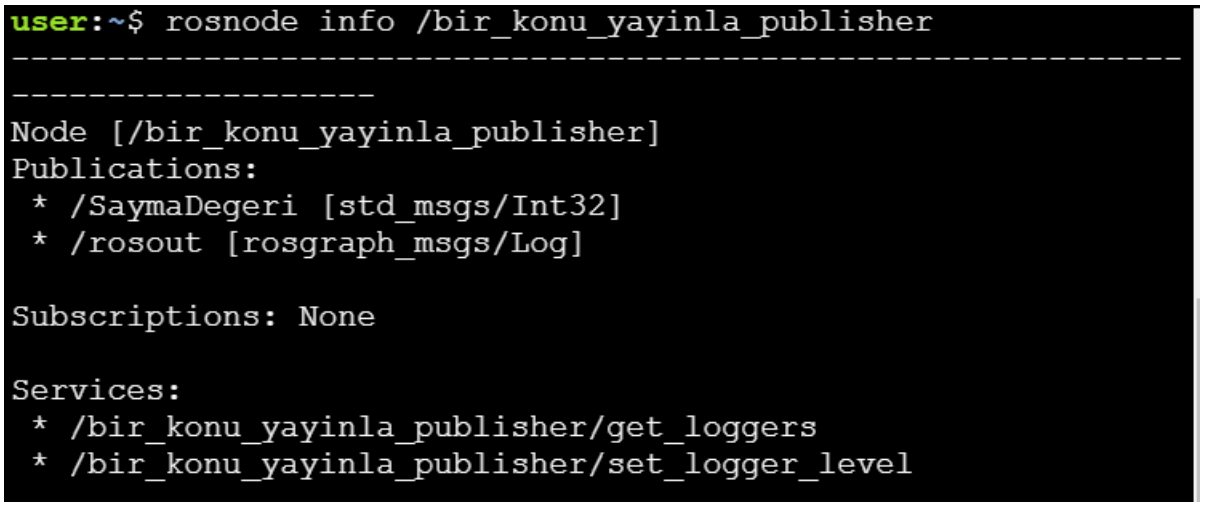
$ rosnode list
O anda aktif olan düğümleri listeler.



Şekil 10. *ROS Düğümlerinin Listelenmesi*

$ rosnode info /bir_konu_yayinla_publisher

Bu komut İlgili düğüme ait detaylı bilgi verir. Şekil 11'de bir düğümün detayları görülmektedir.



Şekil 11. *Bir Düğümün İçeriğine Bakma*

$ rostopic list
$ rostopic echo /SaymaDegeri

Şekil 12'de görüldüğü gibi öncelikle mevcut ROS çekirdeğine ait aktif mesajlar listelenir. Daha sonra echo komutu ilgili konunun (topic) içeriği ekrana yazdırılır.

```
user:~$ rostopic list
/SaymaDegeri
/rosout
/rosout_agg
/statistics
user:~$ rostopic echo /SaymaDegeri
data: 6927
---
data: 6928
---
```

Şekil 12. *Bir Düğümün Değerlerini Ekrana Yazdırma*

$ rostopic type /SaymaDegeri

Şekil 13'de İlgili konunun mesaj tipinin yazdırıldığı ekran görülmektedir.

```
user:~$ rostopic type /SaymaDegeri
std_msgs/Int32
```

Şekil 13. *Bir Mesaj Bağlığını Türünü Sorgulama*

1.6. Gözle: ROS Mesajları

ROS, mesajların iletilmesi için zengin bir veri türü kümesi vardır. std_msgs paketinin standart türleri aşağıda verilmiştir. Bu standart türler, ROS ortamında tüm haberleşme mesajlarını oluşturmak için kullanılır. Örneğin çoğu lazerli telemetre sensörleri **sensor_msgs/LaserScan** mesajları yayınlar.

İlkel Veri Tipleri	Serileştirme	C++	Python2	Python3
bool (1)	işaretsiz (unsigned) 8-bit int	uint8_t (2)	bool	
int8	işaretli (signed) 8-bit int	int8_t	int	
uint8	işaretsiz (unsigned) 8-bit int	uint8_t	int (3)	
int16	işaretli (signed) 16-bit int	int16_t	int	
uint16	işaretsiz (unsigned) 16-bit int	uint16_t	int	
int32	işaretli (signed) 32-bit int	int32_t	int	
uint32	işaretsiz (unsigned) 32-bit int	uint32_t	int	
int64	işaretli (signed) 64-bit int	int64_t	long	int
uint64	işaretsiz (unsigned) 64-bit int	uint64_t	long	int
float32	32-bit IEEE float	float	float	
float64	64-bit IEEE float	double	float	
string	ascii string (4)	std::string	str	bytes
time	secs/nsecs unsigned 32-bit ints	ros::Time	rospy.Time	
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration	

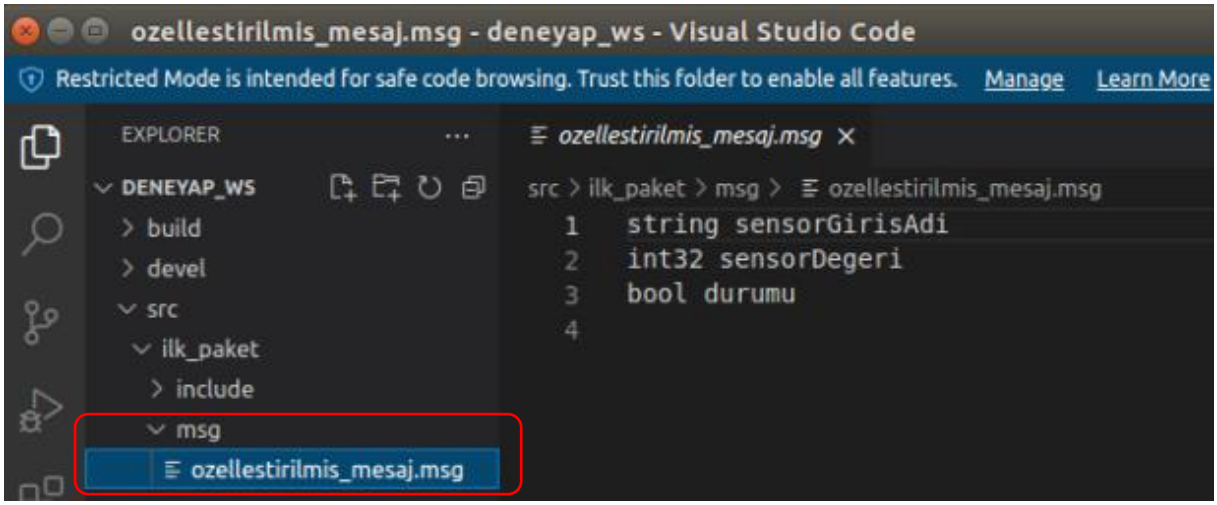
Lazer taramaları ve konum tahminleri için standartlaştırılmış mesaj türlerini kullanmak, farklı robotlar için navigasyon ve haritalama sağlayan düğümlerin yazılmasını sağlayabilir. Ancak, hazır mesaj türlerinin yeterli olmadığı ve kendi mesajlarımızı tanımlamamız gereken durumlar vardır. ROS mesajları, bir paketin msg dizinindeki özel mesaj tanımlama dosyaları ile tanımlanır. Bu dosyalar daha sonra kodunuzda kullanılabilecek dile özgü uygulamalar halinde derlenir. Dosyadaki her satır bir tür ve bir alan adı belirtir.

Not

ROS mesajları ile ilgili detaylı bilgilere
http://wiki.ros.org/msg adresinden ulaşılabilir.
https://github.com/brgokce/deneyap adresinden ise örnek uygulamalara ulaşılabilir.

1.7. Uygula: ROS Mesajları Oluşturuyorum

Özel amaçlı mesaj tanımları için catkin çalışma alanı içerisindeki src klasörü altındaki ilgili paket içerisinde (ilk_paket) msg adında bir klasör oluşturun. Şekil 14'te bu ekran görülmektedir.



Şekil 14. *Özelleştirilmiş Mesaj İçin Bir msg Dosya İçeriği Oluşturma*

Bu klasör içerisine **ozellestirilmis_mesaj.msg** adında bir dosya oluşturun. Dosya içerisine aşağıdaki değişkenleri ve türlerini yazın.

```
string sensorGirisAdi
int32 sensorDegeri
bool durumu
```

Burada yapılan tanımlama bir başlık dosyası veya bir kütüphane değildir. Bu yüzden .msg dosyasının C++, Python ve diğer diller için kaynak koduna dönüştürüldüğünden emin olmalıyız. Yani bu msg dosyayı ROS tarafından bir başlık dosyasına dönüştürülecektir. Üzerinde işlem yaptığımız mevcut paketin **package.xml** dosyası aşağıdaki şekilde yeninden düzenlenir.

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

CMakeLists.txt dosyasında message_generation bağımlılığını find_package çağrısına eklemeliyiz. Böylece mesaj oluşturabilir.

```
find_package(catkin REQUIRED COMPONENTS actionlib actionlib_msgs roscpp rospy std_msgs message_generation)

## Generate added messages and services with any dependencies listed here
 generate_messages(
   DEPENDENCIES
   actionlib_msgs
   std_msgs
 )
```

Ayrıca mesaj çalışma zamanı bağımlılığını dışa aktardığımızdan da emin olmalıyız. CMakeLists.txt dosyası içerisindeki 111'inci satırı açıklama satırından program koduna dönüştürünüz.

```
catkin_package(CATKIN_DEPENDS actionlib actionlib_msgs roscpp rospy std_msgs message_runtime )
```

Aşağıdaki mesaj oluşturma kod bloğunu bulun ve yorum satırlarını kaldırın. Kod bloğu CMakeLists.txt dosyası içerisindeki 53. Satırda olması gerekmektedir. Eğer satırlar kaydı ise lütfen kod üzerinden takip ediniz.

```
## Generate messages in the 'msg' folder
add_message_files( FILES ozellestirilmis_mesaj.msg )
```

Şimdi, **generate_messages()** fonksiyonunun çağrıldığından emin olmalıyız. Aşağıda gösterildiği şekilde satırların yorumunu kaldıralım.

```
## Generate added messages and services with any dependencies listed here
generate_messages( DEPENDENCIES actionlib_msgs std_msgs )
```

Aşağıda CMakeList.txt dosyasının son hali görülmektedir.

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(ilk_paket)

## Find catkin macros and libraries
find_package(catkin REQUIRED COMPONENTS actionlib actionlib_msgs roscpp rospy std_msgs message_generation)

## Generate messages in the 'msg' folder
add_message_files(FILES ozellestirilmis_mesaj.msg)

## Generate added messages and services with any dependencies listed here
```

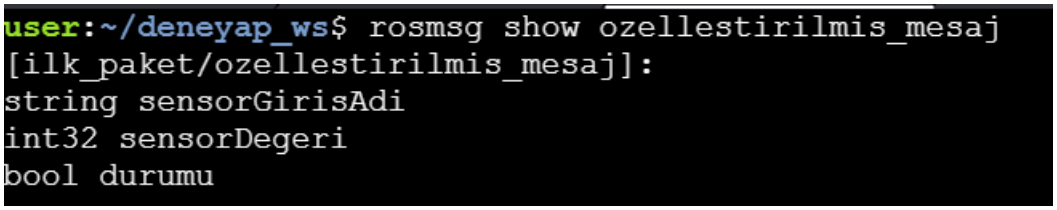
```
generate_messages(DEPENDENCIES actionlib_msgs std_msgs )

## Declare catkin package
catkin_package(CATKIN_DEPENDS actionlib actionlib_msgs roscpp rospy std_msgs message_runtime)

## Specify additional locations of header files
include_directories(${catkin_INCLUDE_DIRS})

## Declare a C++ executable
add_executable(MERHABA_DUGUMU src/ilk_dugum.cpp)
add_executable(konu_yayinla_publisher src/konu_yayinla_publisher.cpp)
add_executable(konuya_abone_ol_subscriber src/konuya_abone_ol_subscriber.cpp)

## Specify libraries to link a library or executable target against
target_link_libraries(MERHABA_DUGUMU ${catkin_LIBRARIES})
target_link_libraries(konu_yayinla_publisher ${catkin_LIBRARIES})
target_link_libraries(konuya_abone_ol_subscriber ${catkin_LIBRARIES})
```

Terminal ekranına aşağıdaki komutları sırası ile yazarak uygulayalım.

```
$ cd ~/deneyap_ws
$ catkin_make
```

Derleme sırasında, kaynak dosyalar yani referans başlık dosyaları **.msg** dosyasından oluşturulur. Eğer derleme sırasında bir hata oluşmuş ise CMakelist.txt dosyasına geri dönerek düzenlemelerin doğru yapılıp yapılmadığını kontrol edin. Terminal üzerinde **rosmsg show** komutunu kullanarak oluşturduğumuz özel mesajı görebildiğimizden emin olmalıyız. Şekil 15'te verilen görsel oluşturulan özel mesajın ROS tarafından tanındığını gösteriyor.

```
$ rosmsg show ozellestirilmis_mesaj
```

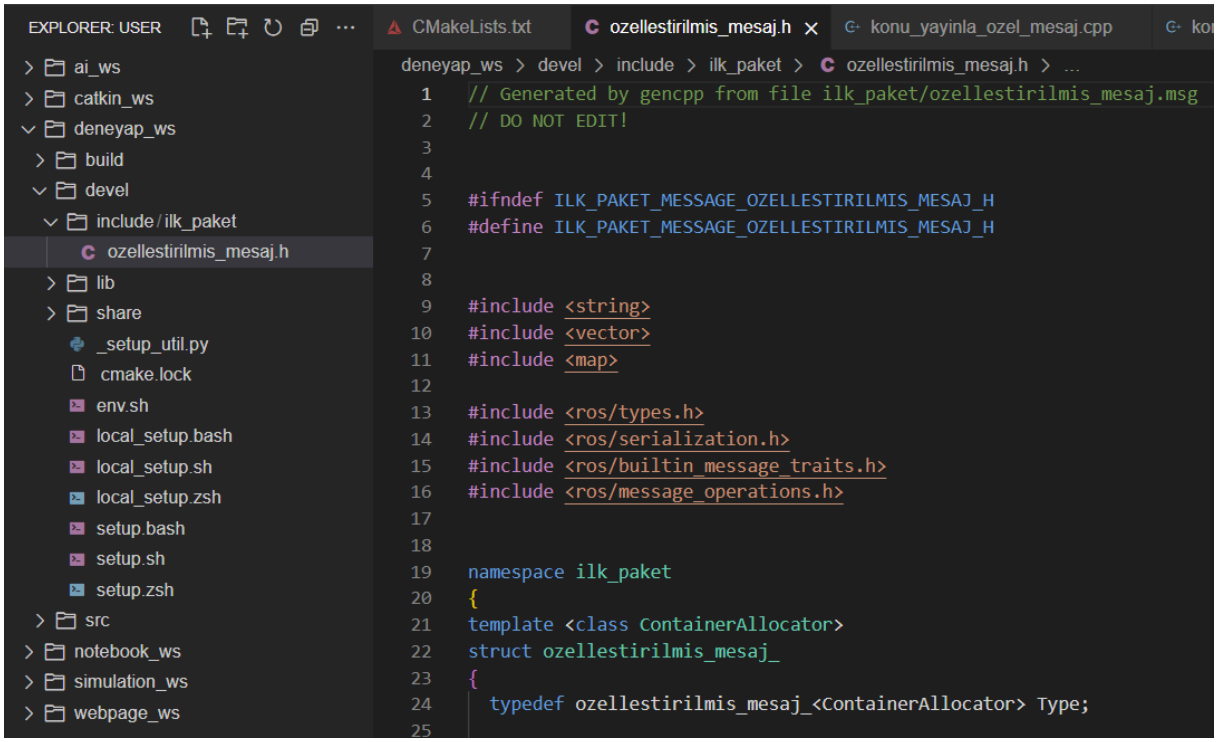


Şekil 15. *ROS İçerisine Eklenmiş Özel Mesaj Türünü Görüntüleme*

Msg dizinindeki herhangi bir .msg dosyası, desteklenen tüm dillerde kullanılması için bir kod oluşturulur. C ++ mesaj başlık dosyası

~/deneyap_ws/devel/include/ içinde oluşturulur.

Şekil 16'da bir msg dosyasından oluşturulan başlık dosyası verilmiştir. Catkin_make ile derleme sonrasında devel/include klasörü altında bir başlık dosyası otomatik olarak oluşur.



Şekil 16. *ROS Tarafıdan Otomatik Olarak Oluşturulmuş Özelleştirilmiş Mesaj Başlık Dosyası*

Şimdi ozellestirilmis_mesaj başlık dosyasını aşağıdaki satırı ekleyerek bir kod bloğuna çağırabilirsiniz. Mesajların paketin adıyla eşleşen bir ad alanına yerleştiği görülmektedir.

```
#include "ilk_paket/ozellestirilmis_mesaj.h"
```

Başlık dosyası proje içerisindeki dosyalara eklenerek tanımlanan değişkenler çağrılabilir. ilk_paket içerisindeki SRC klasörü içerisine aşağıdaki düğümler oluşturulur.

konu_yayinla_ozel_mesaj.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include "ilk_paket/ozellestirilmis_mesaj.h"
#include <iostream>
using namespace std;
int main(int argc, char **argv)
{
   //bir_konu_yayinla bir adıyla ROS düğümü başlatılır.
   ros::init(argc, argv,"ozel_mesaj_yayinla");
   //Bir düğüm tanıtıcı nesnesi oluşturulur
   ros::NodeHandle dugum_nesnesi;
   //Bir yayıncı nesnesi oluşturulur
   ros::Publisher sayi_nesnesi;
   sayi_nesnesi = dugum_nesnesi.advertise<ilk_paket::ozellestirilmis_mesaj>("/Ozel_Mesaj",10);
   //Bir hız nesnesi oluşturulur
   ros::Rate loop_rate(10);
   //sayi adinda bir değişken tanımlanır
   int sayi = 0;
   ilk_paket::ozellestirilmis_mesaj mesaj;
   mesaj.sensorGirisAdi = "ENKODER";
   mesaj.durumu = true;
   mesaj.sensorDegeri = sayi;
   ROS_INFO("Sensor Adi : %s", mesaj.sensorGirisAdi.c_str());
   ROS_INFO("Sensor Durumu : %d",mesaj.durumu);
```

```
    while (ros::ok())
    {
        //Mesaj verisi ekrana yazdırılır
        mesaj.sensorDegeri = sayi;
        ROS_INFO("Sensor Degeri: %d", mesaj.sensorDegeri );
        //konu başlığı yayınlanır.
        sayi_nesnesi.publish(mesaj);
        //Tüm işlemi bir kez yapmak için en az bir kez işlem döndürülür
        ros::spinOnce();
        //Sleeping for sometime
        loop_rate.sleep();
        //sayi degeri arttırılır
        ++sayi;
    }
    return 0;
}
```

konuya_abone_ol_ozel_mesaj.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include "ilk_paket/ozellestirilmis_mesaj.h"
#include <iostream>

//Konu başlığı için çağrılan fonksiyon
void abone_bildirim_fonksiyonu(const ilk_paket::ozellestirilmis_mesaj::ConstPtr& mesaj)
{
    ROS_INFO("Sensor Adi: [%s]",mesaj->sensorGirisAdi.c_str());
    ROS_INFO("Sensor Durumu: [%d]",mesaj->durumu);
    ROS_INFO("Alinan Deger: [%d]",mesaj->sensorDegeri);
}

int main(int argc, char **argv)
{
    //Konuya abone ol adıyla bir ROS düğümü başlatılır
    ros::init(argc, argv,"konuya_abone_ol_ozel_mesaj");
    //Bir düğüm tanıtıcı nesnesi oluşturulur
    ros::NodeHandle dugum_nesnesi;
    //Bir yayınlama nesnesi oluşturulur
    ros::Subscriber abone_ol;
    abone_ol = dugum_nesnesi.subscribe("/Ozel_Mesaj",10,abone_bildirim_fonksiyonu);
    //Düğümün tekrarlı çevrimi sağlanır
    ros::spin();
    return 0;
}
```

CMakeLists.txt dosyasına aşağıdaki kod satırlarını ekleyerek mesajı test edebiliriz.

```
add_executable(konu_yayinla_ozel_mesaj src/konu_yayinla_ozel_mesaj.cpp)
add_executable(konuya_abone_ol_ozel_mesaj src/konuya_abone_ol_ozel_mesaj.cpp)
target_link_libraries(konu_yayinla_ozel_mesaj ${catkin_LIBRARIES})
target_link_libraries(konuya_abone_ol_ozel_mesaj ${catkin_LIBRARIES})
```

Farklı terminaller açın, önce catkin_make ile derleyin, roscore çalışmıyor ise çalıştırın ve sırası ile aşağıdaki komutları kullanarak düğümleri başlatın. RQT Graph üzerinden düğüm ve mesaj başlıkları gözlemleyin. Şimdi her bir terminale aşağıdaki komutları yazalım. Şekil 17'de düğümlerin terminal çıktıları ve şekil 18'de ise rqt_graph çıktısı gösterilmektedir.

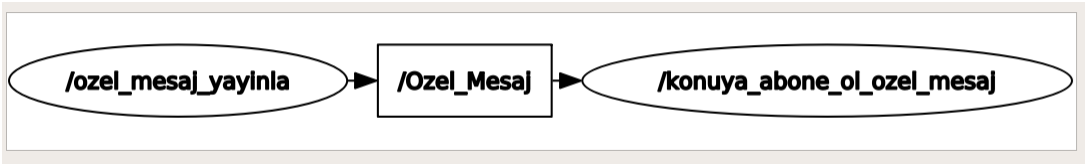
```
$ rosrun ilk_paket konuya_abone_ol_ozel_mesaj
$ rosrun ilk_paket konu_yayinla_ozel_mesaj
```

```
user:~$ rosrun ilk_paket konu_yayinla_ozel_mesaj
[ INFO] [1703428309.758316792]: Sensor Adi : ENKODER
[ INFO] [1703428309.760377338]: Sensor Durumu : 1
[ INFO] [1703428309.760419884]: Sensor Degeri: 0
[ INFO] [1703428309.858380680]: Sensor Degeri: 1
[ INFO] [1703428309.958362206]: Sensor Degeri: 2
[ INFO] [1703428310.058370855]: Sensor Degeri: 3
[ INFO] [1703428310.158354728]: Sensor Degeri: 4
[ INFO] [1703428310.258755106]: Sensor Degeri: 5
[ INFO] [1703428310.358375826]: Sensor Degeri: 6
[ INFO] [1703428310.458362868]: Sensor Degeri: 7
[ INFO] [1703428310.558374794]: Sensor Degeri: 8
[ INFO] [1703428310.658355291]: Sensor Degeri: 9
[ INFO] [1703428310.758360274]: Sensor Degeri: 10
```

```
user:~/deneyap_ws$ rosrun ilk_paket konuya_abone_ol_ozel_mesaj
[ INFO] [1703428326.859191433]: Sensor Adi: [ENKODER]
[ INFO] [1703428326.864057107]: Sensor Durumu: [1]
[ INFO] [1703428326.865303659]: Alinan Deger: [171]
[ INFO] [1703428326.959031930]: Sensor Adi: [ENKODER]
[ INFO] [1703428326.959182506]: Sensor Durumu: [1]
[ INFO] [1703428326.959207142]: Alinan Deger: [172]
[ INFO] [1703428327.064233153]: Sensor Adi: [ENKODER]
[ INFO] [1703428327.064288360]: Sensor Durumu: [1]
[ INFO] [1703428327.064308621]: Alinan Deger: [173]
[ INFO] [1703428327.158985671]: Sensor Adi: [ENKODER]
[ INFO] [1703428327.159041667]: Sensor Durumu: [1]
[ INFO] [1703428327.159064062]: Alinan Deger: [174]
[ INFO] [1703428327.263168729]: Sensor Adi: [ENKODER]
```

Şekil 17. *Yayın Yapan ve Yayına Özel Mesaj Üzerinden Abone Olan Dinleyici Düğümün Durumu*

```
$ rosrun rqt_graph rqt_graph
```



Resim 18. *Özel Mesaj Yayıncısı ve Abone Olan Düğüm Arasındaki İletişimin Grafiği*

```
$ rosmsg info ilk_paket/ozellestirilmis_mesaj
```

Özel olarak oluşturulan ROS mesajının özellikleri listelenebilir ve istenilen değişiklikler yapılarak yeninden mesajlar derlenir.

Not

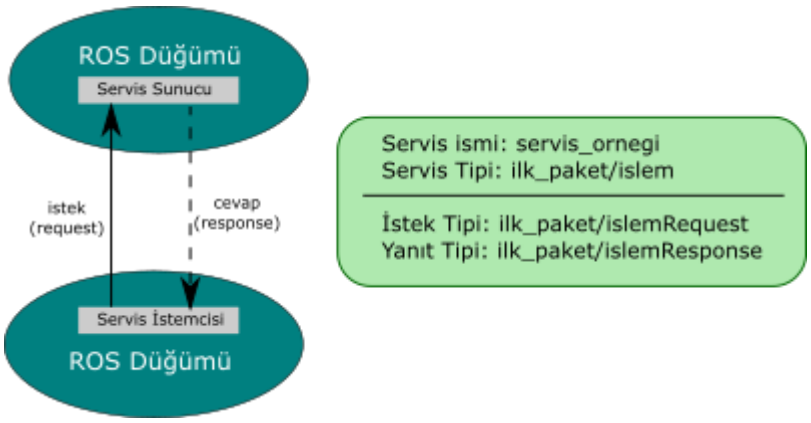
Msg dosyasında bir değişiklik yapıp derleme yaptığınız zaman bir hata alabilirsiniz. Bu durumda cmake.txt dosyasında bu msg dosyasını kullanan düğüm dosyasının add_executable ve target_link_libraries satılarını derleme sonuna kadar kapatın. Derleme işlemi tamamlandıktan sonra tekrar açarak derleyin.

1.8. Gözle: ROS Servisleri (ROS Services)

ROS servisleri eşzamanlı çalışan uzak prosedür çağrılarıdır. Bir düğüm tarafından başka bir düğümde yürütülen bir fonksiyonun çağırılmasını sağlar. Bu fonksiyonun giriş ve çıkışları, ros mesaj tipine benzer şekilde tanımlanır.

Önceki bölümde öğrendiğimiz gibi ROS mesajları, ROS düğümü olarak tanımlanan yayıncılar (publisher) ve abonelerdir (subscriber). Yayıncılar ve aboneler düğümler arası iletişimi sağlamak için kullanılan nesnelerdir ve haberleşme tek taraflıdır. Servisi sağlayan sunucu bir diğer ifade ile yayıncı bir kanala mesaj gönderirken, aboneler yayıncılar tarafından gönderilen mesajları almak için bu kanalları dinler.

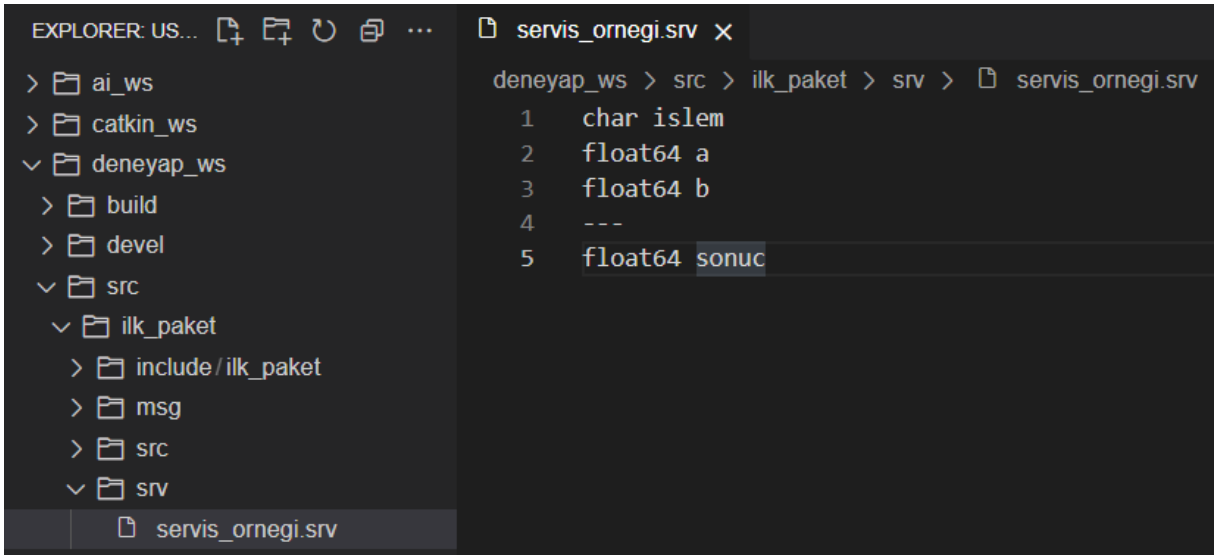
ROS servislerinde ise yayıncılar, talep edilen bir isteğe bağlı olarak farklı türdeki verileri tutan mesajları kullanarak bir kanala yayın yaparlar. Servis aboneleri ise bu mesajları alır ve farklı türlerdeki bu verilerle bir işlem yapmak için bir geri çağırma fonksiyonu yürütür. Bir diğer ifade ile sunucu (servisi sağlayan), istenen servis talebine cevap vermek için bir geri çağırma adı verilen bir bildirim yapar ve böylece servisin yayınını gerçekleştirir. İstemci (servisi isteyen) ise daha sonra bu servise yerel bir aracı (proxy) vasıtası ile erişir. Aşağıdaki resimde bu durum görsel olarak anlatılmaktadır. Şekil 19'da servis istemcisi ve sunucusu arasındaki ilişki gösterilmiştir.



Şekil 19. *ROS Servisleri Sunucu ve İstemci Arasındaki İstek ve Cevap (https://www.mathworks.com/help/ros/ug/call-and-provide-ros-services.html)*

1.9. Uygulama: ROS Servisleri Oluşturuyorum

Çalışma alanı içerisindeki **ilk_paket** klasörü altında **srv** adlı yeni bir klasör oluşturalım ve **.srv** uzantılı (**servis_ornegi.srv)** bir metin dosyası oluşturarak aşağıdaki kodları ekleyelim. Şekil 20'de srv klasörü ve servis dosyası görüntüsü verilmiştir.



Şekil 20. *ROS Servisi İçin "srv" Uzantılı Dosya Oluşturulması*

servis_ornegi.srv

```
char islem
float64 a
float64 b
---
float64 sonuc
```

ROS mesajlarında olduğu gibi **CmakeList.txt** üzerinde aşağıdaki satırların yorumunu kaldırmamız gerekiyor. Bir önceki yapılandırmadaki cmakelist.txt yapısını bozmadan ekleme yaparak ilerliyoruz. Eğer daha önceden bu işlem yapıldı ise tekrar yapmanıza gerek yoktur.

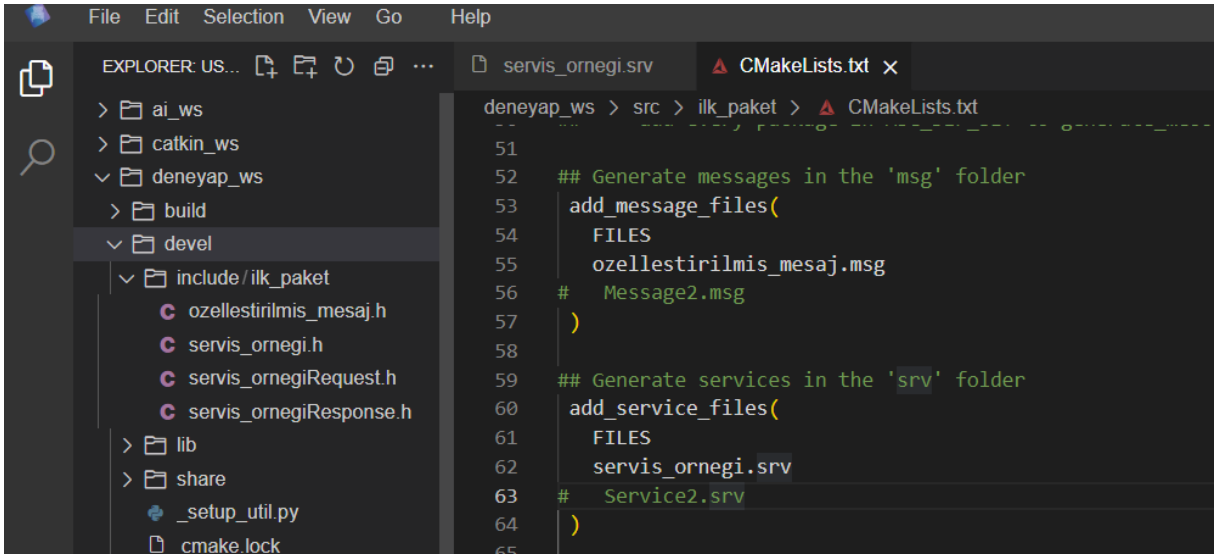
```
add_service_files(
  FILES
  servis_ornegi.srv
)
```

package.xml dosyasını açın ve daha önce eklenmedi ise bu iki satırı ekleyin

```
<build_depend>message_generation</build_depend>

<exec_depend>message_runtime</exec_depend>
```

```
$ cd ~/deneyap_ws
$ catkin_make
```

Catkin derlemesi yapıldıktan sonra şekil 20'de gösterildiği gibi **devel/include** klasörü altında üç adet başlık (*.h) dosyasının oluştuğu görülmektedir. Bunlar servisin istemci(request) ve sunucu (response) tarafı içindir.



Şekil 21. *ROS İçerisinde Oluşturulan Servis Mesajları Başlık Dosyaları ve CMakeList.txt dosyası*

```
$ rossrv list
$ rossrv show ilk_paket/servis_ornegi
```

Komutları kullanılarak ros servis listesi ve mesaj detayları görüntülenebilir. Servis mesajlarını paketin isminden “servis_ornegi.h” başlık dosyasını ilgili kaynak dosyasına ekleyerek çağırabilirsiniz.

#include "ilk_paket/servis_ornegi.h"
Başlık dosyası proje içerisindeki dosyalara eklenerek tanımlanan değişkenler çağrılabilir. ilk_paket içerisindeki SRC klasörü içerisine aşağıdaki düğümler oluşturulur.

ros_servis_istemci.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include "ilk_paket/servis_ornegi.h"
#include <iostream>
#include <sstream>

char op;
float num1, num2;

using namespace std;

int main(int argc, char **argv)
{
    ros::init(argc, argv, "ros_servis_istemci");
    ros::NodeHandle dugum_nesnesi;
    ros::Rate loop_rate(10);

    ros::ServiceClient istemci = dugum_nesnesi.serviceClient<ilk_paket::servis_ornegi>("ros_servis");

    while (ros::ok())
    {
        ilk_paket::servis_ornegi srv;
        cin >> num1;
        cin >> op;
        cin >> num2;
        srv.request.islem=op;
        srv.request.a=num1;
        srv.request.b=num2;
        if (istemci.call(srv))
```

```
        {
            ROS_INFO("%.2f  %c  %.2f = %.2f ",num1,op,num2,srv.response.sonuc);
        }
        else
        {
            ROS_ERROR("Servis Cagrilamadi");
            return 1;
        }

        ros::spinOnce();
        loop_rate.sleep();
    }

    return 0;
}
```

ros_servis_sunucu.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include "ilk_paket/servis_ornegi.h"
#include <iostream>
#include <sstream>

bool ros_servis_bidirimi(ilk_paket::servis_ornegi::Request  &req,
                         ilk_paket::servis_ornegi::Response &res)
{
    switch (req.islem) {
        case '+':
            res.sonuc= req.a + req.b;
            ROS_INFO("%.2f %c %.2f = %.2f",req.a,req.islem,req.b,res.sonuc);
            break;
        case '-':
            res.sonuc= req.a - req.b;
            ROS_INFO("%.2f %c %.2f = %.2f",req.a,req.islem,req.b,res.sonuc);
            break;
        case '*':
            res.sonuc= req.a * req.b;
            ROS_INFO("%.2f %c %.2f = %.2f",req.a,req.islem,req.b,res.sonuc);
            break;
        case '/':
            res.sonuc= req.a / req.b;
            ROS_INFO("%.2f %c %.2f = %.2f",req.a,req.islem,req.b,res.sonuc);
            break;
        default:
            break;
    }
    return true;
}

int main(int argc, char **argv)
{
    //bir_konu_yayinla bir adıyla ROS düğümü başlatılır.
    ros::init(argc, argv,"ros_servis_sunucu");
    //Bir düğüm tanıtıcı nesnesi oluşturulur
    ros::NodeHandle dugum_nesnesi;
    //Bir servis sunucu nesnesi oluşturulur
    ros::ServiceServer servis = dugum_nesnesi.advertiseService("ros_servis", ros_servis_bidirimi);
    ROS_INFO("ROS servis sunucusu istemciden talep almaya hazir.");
    ros::spin();
    return 0;
}
```

CMakeLists.txt dosyasına aşağıdaki kod satırlarını ekleyerek mesajı test edebiliriz.

```
add_executable(ros_servis_sunucu            src/ros_servis_sunucu.cpp)
add_executable(ros_servis_istemci           src/ros_servis_istemci.cpp)
```

```
target_link_libraries(ros_servis_sunucu     ${catkin_LIBRARIES})
target_link_libraries(ros_servis_istemci    ${catkin_LIBRARIES})
```

İki farklı terminal açarak önce catkin_make ile derleyin, roscore çalışmıyor ise çalıştırın, ve sırası ile aşağıdaki komutlar ile düğümleri başlatın. Daha sonra rqt grafiği üzerinden düğüm ve mesaj başlıkları gözlemleyin. Şekil 22 ve 23'deki gibi her bir terminale aşağıdaki komutları yazalım. Daha sonra istemci terminaline toplama, çıkarma, çarpma ve bölme işlemlerinden oluşan işlemleri girelim ve sonucu servis penceresinden gözlemleyelim.

```
$ rosrun ilk_paket ros_servis_sunucu
$ rosrun ilk_paket ros_servis_istemci
```

```
user:~/deneyap_ws$ rosrun ilk_paket ros_servis_sunucu
[ INFO] [1703430778.690336950]: ROS servis sunucusu istemciden talep almaya hazir.
[ INFO] [1703430828.728659521]: 8.00 + 7.00 = 15.00
[ INFO] [1703430885.945431701]: 9.00 * 5.00 = 45.00
[ INFO] [1703430890.054520876]: 12.00 / 4.00 = 3.00
[ INFO] [1703430897.138478200]: 87.00 - 17.00 = 70.00
```

Şekil 22. *ROS Servisi İçin Sunucu Düğümü*

```
user:~$ rosrun ilk_paket ros_servis_istemci
8+7
[ INFO] [1703430828.728844538]: 8.00  +  7.00 = 15.00
9*5
[ INFO] [1703430885.945585584]: 9.00  *  5.00 = 45.00
12/4
[ INFO] [1703430890.054722009]: 12.00  /  4.00 = 3.00
87-17
[ INFO] [1703430897.139202602]: 87.00  -  17.00 = 70.00
```

Şekil 23. *ROS Servisi İçin İstemci Düğümü*

Aşağıda ROS servisleri ile ilgili olarak ek komutlar verilmiştir. Daha fazlası için lütfen ros.org sitesini ziyaret edin.

```
$ rosservice list: Bu komut ile mevcut ROS hizmetlerini listelenir
$ rosservice type /ros_servis: Bu komut servis mesaj tipini yazdırır
$ rosservice info /ros_servis: ros_servis hakkında bilgi verir.
```

1.10. Gözle: ROS Eylem Kütüphaneleri

ROS servisinde, iki düğüm arasında bir istek ve yanıt etkileşimi uygulanır. Ancak yanıt çok uzun sürerse veya sunucu verilen işi zamanında bitirmediyse çok uzun süre beklenmek zorunda kalınabilir. Böyle durumda verilen görevin sonlandırılması istenebilir. Böyle durumlar için ROS eylem kütüphaneleri kullanılır. Eylem kütüphanelerinde bir istek istediğimiz zamanda bitmedi veya istediğimiz sonuç aralığından farklı bir değer aldı ise yine işlemi yarıda kesebiliriz. Sistemi bu şekilde yönetmek daha da hızlı ve esnek olabilmektedir. "Actionlib" adı verilen eylem kütüphaneleri birtakım önleyici görevleri uygulamak için kullanılır. Eylem kütüphaneleri otonom mobil robot ve İHA uygulamalarında ağırlıklı olarak kullanılır.

ROS servislerinde olduğu gibi eylem kütüphanelerinde de (actionlib) bir eylem veya iş tanımlaması yapılmalıdır. Eylem (action) tanımlaması ".action" uzantısına sahip bir metin dosyası içerisinde yapılır.

Bu dosya, ROS paketinin içindeki **action** klasörü içinde tutulur. Bir eylem kütüphanesi dosyası şu bölümlerden oluşur:

Hedef: Bir eylem sunucusu aktif bir şekilde istemcilere hizmet vermek için çalıştırılır. Daha sonra istemci (end node client) bir eylem isteği gönderir, bu durum aynı ROS servisindeki servis isteğine benzer. Örneğin "1,1" noktasında 45 derecelik yönelimde duran bir mobil robot 135 derecelik açı yaparak (1,0) noktasına hareket ettirmek istenir.

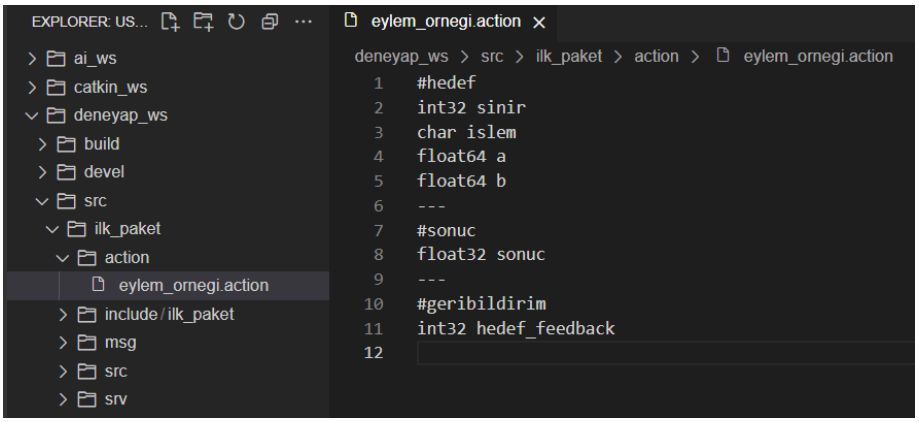
Sonuç: Robot hedefe ulaştıktan sonra, eylem sunucusu nihai bir tamamlama veya işlem sonucu gönderir. Bu bir matematiksel hesaplama sonucu veya bir alındı bilgisi de olabilir. Önceki örnekte robot eylemi 135 dereceye ve (1,0) noktasına ulaşırsa, hedefe ulaşmış olur ve sonuç olarak robotun hedefe ulaştığını gösteren herhangi bir bilgi gönderir.

Geri bildirim: Bir istemci, eylem sunucusuna bir işlem talebi gönderdiğinde, bir geri çağırma fonksiyonu yürütülmeye başlanır. Geri bildirim, geri çağırma fonksiyonu içindeki mevcut işlemin ilerlemesini vermektedir. Geri bildirim tanımını kullanarak mevcut ilerlemeyi elde edebiliriz. Örneğin bir robot kolu önceki durumda, robot kol eklemi 90 derece hareket etmelidir. Bu durumda geri besleme, kolun hareket ettiği 45 ile 90 derece arasındaki ara değer olabilir.

1.11. Uygula: ROS Eylem (Action) Kütüphaneleri Oluşturuyorum

ROS ortamında bir eylem kütüphanesi örneği için ilk_paket klasörü içerisine **action** adında bir klasör oluşturulur. Klasör içerisine **eylem_ornegi.action** adında bir metin dosyası oluşturulur. Bu metin dosyası içerisine aşağıdaki komut dizesi eklenir. Şekil 24'te eylem kütüphanesi dosyası ve mesaj içeriği görülmektedir.

```
#hedef
int32 sinir
char islem
float64 a
float64 b
---
#sonuc
float32 sonuc
---
#geribildirim
int32
hedef_feedback
```



Şekil 24. *Eylem Kütüphanesi Mesaj İçeriği*

ROS servislerine olduğu gibi **CmakeList.txt** üzerinde aşağıdaki satırların yorumunu kaldırmamız gerekiyor. Bunlar;

```
## Generate actions in the 'action' folder
add_action_files( FILES    eylem_ornegi.action )
```

Daha sonra yine CmakeList.txt üzerinde aşağıdaki eklemeleri yapalım.

```
find_package(catkin REQUIRED COMPONENTS
  actionlib
  actionlib_msgs
  roscpp
```

```
  rospy
  std_msgs
  message_generation
)

generate_messages(DEPENDENCIES actionlib_msgs std_msgs)
Catkin_package(CATKIN_DEPENDS actionlib actionlib_msgs roscpp rospy
std_msgs message_runtime)
```

package.xml dosyasını kontrol edelim ve aşağıdaki satırlar yok ise ekleyelim.

```
<build_depend>actionlib</build_depend>
<build_depend>actionlib_msgs</build_depend>
```

Daha sonra derleme işlemlerini aşağıdaki kodlarla gerçeleştirelim. CMakeList.txt ve Package.xml dosyalarının son helleri aşağıda verilmiştir.

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(ilk_paket)

find_package(catkin REQUIRED COMPONENTS
actionlib actionlib_msgs roscpp rospy std_msgs message_generation)

## Generate messages in the 'msg' folder
add_message_files(FILES ozellestirilmis_mesaj.msg)

## Generate services in the 'srv' folder
add_service_files(FILES servis_ornegi.srv )

## Generate actions in the 'action' folder
add_action_files(FILES eylem_ornegi.action )

## Generate added messages and services with any dependencies listed here
generate_messages(DEPENDENCIES actionlib_msgs std_msgs)
catkin_package(CATKIN_DEPENDS actionlib actionlib_msgs roscpp rospy std_msgs
message_runtime)

include_directories(${catkin_INCLUDE_DIRS})

add_executable(MERHABA_DUGUMU src/ilk_dugum.cpp)
add_executable(konu_yayinla_publisher src/konu_yayinla_publisher.cpp)
add_executable(konuya_abone_ol_subscriber src/konuya_abone_ol_subscriber.cpp)
add_executable(konu_yayinla_ozel_mesaj src/konu_yayinla_ozel_mesaj.cpp)
add_executable(konuya_abone_ol_ozel_mesaj src/konuya_abone_ol_ozel_mesaj.cpp)
add_executable(ros_servis_sunucu            src/ros_servis_sunucu.cpp)
add_executable(ros_servis_istemci           src/ros_servis_istemci.cpp)

target_link_libraries(MERHABA_DUGUMU ${catkin_LIBRARIES})
target_link_libraries(konu_yayinla_publisher ${catkin_LIBRARIES})
target_link_libraries(konuya_abone_ol_subscriber ${catkin_LIBRARIES})
target_link_libraries(konu_yayinla_ozel_mesaj ${catkin_LIBRARIES})
target_link_libraries(konuya_abone_ol_ozel_mesaj ${catkin_LIBRARIES})
target_link_libraries(ros_servis_sunucu     ${catkin_LIBRARIES})
target_link_libraries(ros_servis_istemci    ${catkin_LIBRARIES})
```

```
$ cd ~/deneyap_ws
$ catkin_make
```

Aşağıdaki örnekte istemci (client), hedefe iki adet sayı, bir işlem karakterini ve bir de sınır değeri gönderir. Bir eylem sunucusu gönderilen hedefi aldığında, iki sayısı gelen işleme tipine göre işler ve sonucu bir sınır değer ile karşılaştırır. Eğer sonuç hedef değerden küçük ise cevabı geri döndürür. Ancak sonuç hedef değerden büyükse o zaman hata mesajı döndürerek sonucun yanlış olduğunu bildirir. Böylece işlemin sonucundan kesin olarak haberdar oluruz. Diğer türlü olsaydı işlemin yapılıp yapılmadığından ve sonucun nasıl olduğundan haberdar olamayacaktık. Buradaki geri bildirim, sonucun eşik değerden büyük ya da küçük olma durumudur. Bu eylem servislerinin komutları aşağıdaki gibidir.

Aşağıda verilen başlık dosyaları istemci ve sunucu dosyalarına ayrı ayrı eklenir. ROS eylem düğümlerini yazmak için **ilk_paket/src** klasörü içerisine "**ros_eylem_istemci.cpp**" ve "**ros_eylem_sunucu.cpp**" adında birer cpp dosyası oluşturalım.

```
#include <actionlib/client/simple_action_client.h>
#include <actionlib/server/simple_action_server.h>
#include "ilk_paket/eylem_ornegi_Action.h"
```

ros_eylem_istemci.cpp

```
#include "ros/ros.h"
#include <iostream>
#include <actionlib/client/simple_action_client.h>
#include <actionlib/client/terminal_state.h>
#include "ilk_paket/eylem_ornegiAction.h"

char op;
float num1, num2;
using namespace std;

int main (int argc, char **argv)
{
  ros::init(argc, argv, "ros_eylem_istemci");

   if(argc != 2){
  ROS_INFO("%d",argc);
  ROS_WARN("En az bir sayi girmeniz gerekiyor!!!");
  return 1;
  }
// eylem istemcisini oluşturuluyor ve istemcinin kendi iş parçacığını döndürür
  actionlib::SimpleActionClient<ilk_paket::eylem_ornegiAction> ac("ros_eylem_sunucu", true);

  ROS_INFO("Eylem sunucusunun baslatilmasi bekleniyor");

  // Eylem sunucusunun başlatılması bekleniyor
  ac.waitForServer(); //Sürekli bekliyor
  ROS_INFO("Eylem sunucusu baslatildi");
  ROS_INFO("a islem b olacak sekilde islemi giriniz");
  // ROS_INFO("hesap icin 1. sayiyi giriniz");
  cin >> num1;
  // ROS_INFO("hesap icin islem turu giriniz");
  cin >> op;
  // ROS_INFO("hesap icin 2. sayiyi giriniz");
  cin >> num2;
  ROS_INFO("Action sunucusu baslatildi, Hesap icin gonderiliyor.");
  // send a goal to the action
  ilk_paket::eylem_ornegiGoal goal;
  goal.islem = op;
  goal.a=num1;
  goal.b=num2;
  goal.sinir=atoi(argv[1]);
```

```
  ROS_INFO("islem( %.2f %c %.2f ) sinir(%d)",goal.a, goal.islem, goal.b, atoi(argv[1]));
  ac.sendGoal(goal);

  bool sinir_asimi = ac.waitForResult(ros::Duration(atoi(argv[1])));

  ilk_paket::eylem_ornegiResultConstPtr result=ac.getResult();

  ac.cancelGoal();
  if (result->sonuc <= atoi(argv[1]))
  {
    actionlib::SimpleClientGoalState state = ac.getState();
    ROS_INFO("Eylem Sonlandi!!!: %s ",state.toString().c_str());

    //Öncelik Alma Gerçekleşiyor
    ac.cancelGoal();
  }
  else
    ROS_INFO("Hesaplanan deger sinirdan buyuk!!!");

  //çıkış
  return 0;
}
```

ros_eylem_sunucu.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <actionlib/server/simple_action_server.h>
#include "ilk_paket/eylem_ornegiAction.h"
#include <iostream>
#include <sstream>

class eylem_ornegi_action
{
protected:
  ros::NodeHandle nh_;
  // NodeHandle yapisi bu satirdan once oluşturulmalıdır. Aksi takdirde bir hata oluşabilir.
  actionlib::SimpleActionServer<ilk_paket::eylem_ornegiAction> as;
  // yayınlanan geri bildirim/sonuç için kullanılan mesajlar oluşturur
  ilk_paket::eylem_ornegiFeedback feedback;
  ilk_paket::eylem_ornegiResult result;

  std::string action_name;
  int hesap;
  float hesap_sonuc;

public:
  eylem_ornegi_action(std::string name) :
    as(nh_, name, boost::bind(&eylem_ornegi_action::executeCB, this, _1), false),
    action_name(name)
  {
  as.registerPreemptCallback(boost::bind(&eylem_ornegi_action::preemptCB, this));
  as.start();
  }

  ~eylem_ornegi_action(void){}

  void preemptCB()
  {
  ROS_WARN("%s oncelikli oldu!", action_name.c_str());

  as.setPreempted(result,"Ben onlendim");
  }

  void executeCB(const ilk_paket::eylem_ornegiGoalConstPtr &hesap)
  {
    if(!as.isActive() || as.isPreemptRequested()) return;
    ros::Rate rate(5);
    ROS_INFO("%s calisiyor sinir= %d", action_name.c_str(), hesap->sinir);

    switch (hesap->islem) {
```

```
        // If user enter +
        case '+':
            hesap_sonuc = hesap->a + hesap->b;
            // printf("%.2f  ,%.2f   ,%.2d",hesap->a,hesap->b,hesap_sonuc);
            ROS_INFO("%.2f %c %.2f = %.2f",hesap->a,hesap->islem,hesap->b,hesap_sonuc);
            break;

        // If user enter -
        case '-':
            hesap_sonuc = hesap->a - hesap->b;
            //printf("%.2f  ,%.2f   ,%.2f",hesap->a,hesap->b,res.sonuc);
            ROS_INFO("%.2f %c %.2f = %.2f",hesap->a,hesap->islem,hesap->b,hesap_sonuc);
            break;

        // If user enter *
        case '*':
            hesap_sonuc= hesap->a * hesap->b;
            //printf("%.2f  ,%.2f   ,%.2f",hesap->a,hesap->b,res.sonuc);
            ROS_INFO("%.2f %c %.2f = %.2f",hesap->a,hesap->islem,hesap->b,hesap_sonuc);
            break;

        // If user enter /
        case '/':
            hesap_sonuc= hesap->a / hesap->b;

            ROS_INFO("%.2f %c %.2f = %.2f",hesap->a,hesap->islem,hesap->b,hesap_sonuc);
            break;

        // operator +, -, * veya / 'den farkli bir isaret girerse hata mesaji gosterilir
        default:
            break;
          }

    if(!as.isActive() || as.isPreemptRequested()){
      return;
    }

    if(hesap->sinir >= hesap_sonuc){
      ROS_INFO("%s basarili hesaplanan sayi= %.2f", action_name.c_str(), hesap_sonuc);
      result.sonuc = hesap_sonuc;
      as.setSucceeded(result);

    }else{
      ROS_INFO("Hesaplanan sayi sinirdan daha buyuk (%.2f > %d)",hesap_sonuc,hesap->sinir);
      feedback.hedef_feedback = result.sonuc;
      as.publishFeedback(feedback);
      as.setAborted(result);
    }
    rate.sleep();
  }
};

int main(int argc, char** argv)
{
  ros::init(argc, argv, "ros_eylem_sunucu");
  ROS_INFO("Eylem Sunucusu Baslatildi");
  eylem_ornegi_action action_ornegi_obj(ros::this_node::getName());
  ros::spin();
  return 0;
}
```

Bu iki dosya src klasöründe oluşturulduktan sonra, düğümleri ROS ortamına tanıtmak için package.xml ve CMakeLists.txt dosyasını düzenlememiz gerekiyor. “package.xml” dosyası, ROS hizmeti ve mesajları için yaptığımız gibi mesaj oluşturma ve çalışma zamanı paketlerini içermelidir.

Bu düğümleri ROS ortamına tanıtmak için CMakeLists.txt dosyasına dahil etmeliyiz. Ayrıca bu örnek için yazdığımız eylem dosyalarını da eklememiz gerekiyor:

```
## Generate actions in the 'action' folder
```

```
add_action_files( FILES eylem_ornegi_.action )

include_directories( include ${catkin_INCLUDE_DIRS} ${Boost_INCLUDE_DIRS} )

add_executable(ros_eylem_sunucu      src/ros_eylem_sunucu.cpp)
add_executable(ros_eylem_istemci     src/ros_eylem_istemci.cpp)

target_link_libraries(ros_eylem_sunucu  ${catkin_LIBRARIES})
target_link_libraries(ros_eylem_istemci ${catkin_LIBRARIES})
```

Yeni bir terminal açarak aşağıdaki 'roscore'u çalıştıralım.

$ roscore

Yine iki farklı terminal açarak aşağıdaki ROS düğümlerini ayrı ayrı başlatalım:

$ rosrun ilk_paket ros_eylem_sunucu

$ rosrun ilk_paket ros_eylem_istemci 50

Not: 50 sınır değer olarak giriliyor.

ROS eylem kütüphanesi kullanılarak oluşturulan örneğin çıktısı şekil 25 ve şekil 26'da gösterilmektedir. Burada önce eylem sunucusu başlatılır ve daha sonra istemci başlatılarak bir işlem için sayı girilmesi istenir. Sonuç olarak sunucu işlemi yapar ve sonucu eşik değer ile karşılaştırır. Eğer sonuç eşik veya referans değerin altında ise işlem başarılıdır. Eğer sonuç eşik veya referans değerin üstünde ise işlem hatalı olarak belirlenir.

```
user:~$ rosrun ilk_paket ros_eylem_sunucu
[ INFO] [1703444353.553048496]: Eylem Sunucusu Baslatildi
[ INFO] [1703444366.318696310]: /ros_eylem_sunucu calisiyor sinir= 50
[ INFO] [1703444366.350235455]: 12.00 + 24.00 = 36.00
[ INFO] [1703444366.350300148]: /ros_eylem_sunucu basarili hesaplanan sayi= 36.00
[ INFO] [1703444426.903348118]: /ros_eylem_sunucu calisiyor sinir= 50
[ INFO] [1703444426.903409435]: 25.00 * 5.00 = 125.00
[ INFO] [1703444426.903440565]: Hesaplanan sayi sinirdan daha buyuk (125.00 > 50)
```

Şekil 25. *ROS Eylem Kütüphanesi İçin Sunucu Düğümü*

```
user:~$ rosrun ilk_paket ros_eylem_istemci 50
[ INFO] [1703444361.338056456]: Eylem sunucusunun baslatilmasi bekleniyor
[ INFO] [1703444361.638999967]: Eylem sunucusu baslatildi
[ INFO] [1703444361.639055299]: a islem b olacak sekilde islemi giriniz
12+24
[ INFO] [1703444366.318218870]: Action sunucusu baslatildi, Hesap icin gonderiliyor.
[ INFO] [1703444366.318284329]: islem( 12.00 + 24.00 ) sinir(50)
[ INFO] [1703444366.350617508]: Eylem Sonlandi!!!: SUCCEEDED
user:~$ rosrun ilk_paket ros_eylem_istemci 50
[ INFO] [1703444415.628052564]: Eylem sunucusunun baslatilmasi bekleniyor
[ INFO] [1703444415.974318033]: Eylem sunucusu baslatildi
[ INFO] [1703444415.974373819]: a islem b olacak sekilde islemi giriniz
25*5
[ INFO] [1703444426.902863590]: Action sunucusu baslatildi, Hesap icin gonderiliyor.
[ INFO] [1703444426.902927854]: islem( 25.00 * 5.00 ) sinir(50)
[ INFO] [1703444426.903900308]: Eylem Sonlandi!!!: ABORTED
```

Şekil 26. *ROS Eylem Kütüphanesi İçin İstemci Düğümü*

1.12. Gözle: ROS Hızlı Başlatma Dosyaları

ROS'ta düğümleri, servisleri ve diğer bileşenleri ayrı ayrı başlatmak bir ekran üzerinde çok fazla terminal penceresinin artmasına ve yönetimin zorlaşmasına sebep olmaktadır. Bu çalışmada şuana kadar çok fazla ROS düğümü görmedik. Bir insansız hava aracı veya bir insansız kara aracı için 15-20 farklı düğüm ve servislerin eş zamanlı çalışması gerekebilmektedir. Böyle bir sistemde her bir düğüm ve servisin ayrı ayrı terminallerde başlatılması yönetimi zorlaştıracaktır. Bu sorunu ortadan kaldırmak ve yönetilebilirliği kolaylaştırmak için başlatma/yükleme (launch) dosyası oluşturulur. Başlatma dosyaları, birden fazla düğümü başlatmak için çok verimli ve esnek bir yöntemdir. Projedeki düğümler ve servisler XML yapısında oluşturulan başlatma dosyaları ".launch" uzantılı bir dosya içine yazılır. Daha sonra terminalden **roslaunch** komutu kullanılarak proje tek bir launch dosyası kullanılarak başlatılabilir.

Roslaunch komutu, ROS Core'u ve parametre sunucusunu otomatik olarak başlatır. Yani ayrı bir terminal üzerinden **roscore** komutunu kullanmamıza gerek yoktur. Tüm işlemler tek bir "**roslaunch**" komutu ile yapılabilir.

1.13. Uygula: ROS Hızlı Başlatma Dosyaları (ROS Launch) Oluşturuyorum

Terminal ekranında aşağıdaki komutları sırası ile uygulayalım veya Visual Studio Code ara yüzü ile görsel olarak yapalım.

```
$ roscd ilk_paket
$ mkdir launch
$ cd launch
```

launch klasörü Visual Studio Code arayüzü ile görüntüleyin ve klasör içerisine ros_servis_ornegi.launch adında bir dosya oluşturun ve aşağıdaki kodları yazın.

```
<launch>
    <node name="ros_servis_sunucu" pkg="ilk_paket"
     type="ros_servis_sunucu" output="screen" />
    <node name="ros_servis_istemci" pkg="ilk_paket"
     type="ros_servis_istemci" output="screen" />
</launch>
```

Kodu inceleyelim: <launch></launch> etiketleri, bir başlatma dosyasındaki kök öğedir. Yani komut dizesi bu etiketler arasında yazılır. <node> etiketi, başlatılması istenen düğümü belirtir.

```
<node name=" ros_servis_sunucu" pkg="ilk_paket" type=" ros_servis_sunucu" output="screen"/>
```

<node> içindeki "name" etiketi düğümün adını gösterir, pkg ise paketin adıdır ve type ise başlatacağımız yürütülebilir dosyanın adıdır.

ros_servis_ornegi.launch başlatma dosyasını oluşturduktan sonra, aşağıdaki komutu kullanarak başlatabiliriz:

```
$ roslaunch ilk_paket ros_servis_ornegi.launch
```

Yükleme “launch” dosyası başarılı bir şekilde yüklendiğinde Şekil 27’de verilen terminal gibi görünür.

```
started roslaunch server http://4_xterm:35909/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.11

NODES
  /
    ros_servis_istemci (ilk_paket/ros_servis_istemci)
    ros_servis_sunucu (ilk_paket/ros_servis_sunucu)

ROS_MASTER_URI=http://4_xterm:11311

process[ros_servis_sunucu-1]: started with pid [10967]
[ INFO] [1703445107.051600447]: ROS servis sunucusu istemciden talep almaya hazir.
process[ros_servis_istemci-2]: started with pid [10968]
12+15
[ INFO] [1703445112.487807123]: 12.00 + 15.00 = 27.00
[ INFO] [1703445112.488121747]: 12.00  +  15.00 = 27.00
```

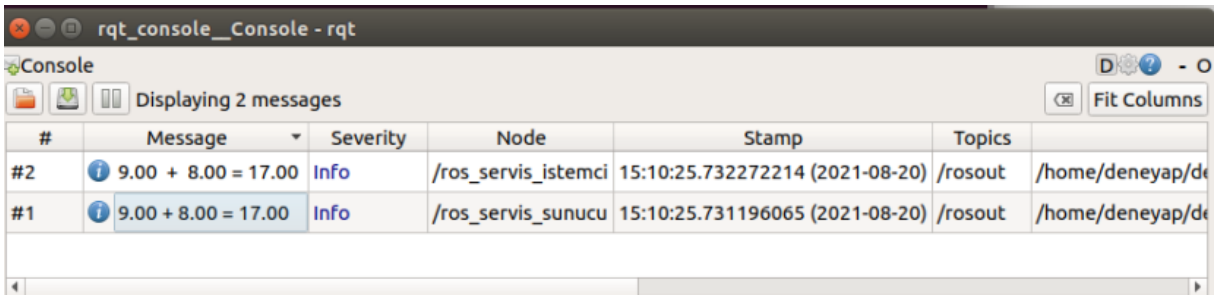
Şekil 27. *ROS Sunucu Düğümünde Terminal Cevabı*

Düğüm listesini görüntüleyelim:
`$ rosnode list`

“rqt_console” adlı bir GUI aracı kullanarak mesajları görüntüleyebilir ve düğümlerdeki hatayı ayıklayabiliriz.

`$ rqt_console`

Şekil 28’de düğümler tarafından oluşturulan günlükler görülmektedir. Burada her bir servis işleminin günlüklendiği görülmektedir. Böylece robot kontrolünde süreç denetimi yapılabilmektedir.



Şekil 28. *rqt_console Aracı ile Günlükleri Görüntüleme*

2. TASARLA

Öğrencilerden, gün içerisinde 80-150 arasında sayı üretip bunu 10 hz'de yayınlayan bir yayıncı düğümü, bu düğümden gelen sayılardan 90-110 arasındaki sayıları sayan ve sayıların adetleri toplamları 20 olduğunda mesaj veren bir abone düğümü ve bunları otomatik olarak başlatan başlatma dosyasını yazılımsal olarak tasarlamaları istenir. Tasarlama aşamasında öğrencilerin derste verilen örnekten faydalanması istenir.

Tanımlama: Öğrencilerden öncelikle problemin bileşenlerine ayrılması istenilenlerin maddeler hâlinde yazmaları beklenir. Bu aşamada kâğıt ve kalem kullanarak programın algoritmasını veya akış diyagramını hazırlamaları istenir.

Örneğin bir düğüm için;

- Yeni bir paket oluşturma
- ROS Bağlılıklarını kontrol etme
- Yayıncı/abone düğüm oluşturma
- Rastgele sayı üretme fonksiyonunu araştırma
- Kodlama için sözdizimini gözden geçirme
- CMake yapısını gözden geçirme
- Derleme ve düğümleri çağırma

Fikir üretme: Bu aşamada öğrencilerin tanımlamada belirlenen işlemlerin nasıl yapılabileceği ile ilgili fikir yürütmesi beklenir. Örnek olarak öğrenciler aşağıdaki maddelere benzer fikirler üretebilir:

- Öncelikle yayıncı ve abone düğümler arasındaki iletişim sağlanması planlanmalıdır. Planlanan her bir işlemin tanımı yapılmalıdır (rastgele sayı üretme, kullanılacak mesaj tipi). Bu aşamada öğrenciler ROS ve C++ sözdizimlerini hızlıca yazılımsal olarak deneyebilirler.
- RQT ile yayıncı ve abone düğümleri ilişkisi gözlemlenebilir.
- Aktarılan verilerin işlenmesi sağlanabilir.
- Karşılaştırma çıktılarının ekrana yazdırılması sağlanabilir.

ROS içerisinde yayıncı ve abone düğümleri planlanan şekillerde oluşturulması için gözetmen desteği ile bilgisayar bileşenleri ve Visual Studio yazılım ortamı kullanılabilir. Düğümler arası iletişim sağlanması ve verilerin işlenmesi için oluşabilecek muhtemel sorunlarla ilgili öğrencilerden örneklere ve işlem algoritmasına bakarak çözüm üretmeleri istenir.

3. ÜRET

Tasarla ve üret bölümlerinde öğrenciler aktif rol üstlenerek verilen problemi çözer. Rehber öğretmen öğrencilere yalnızca zorlandıkları noktalarda destek olur. Öğrenciler bilgisayar başında çalışarak ROS için gerekli yazılım çözümlerini geliştirirler (Eğitim programındaki bu ve benzeri bütün örnekler/programlar rehber öğretmenlere verilecektir).

4. DEĞERLENDİR

Günün sonunda öğrencilerle halka oluşturulur. Aşağıdaki sorular üzerinden tartışma ortamı yaratılır.

- ROS ortamında yayıncı ve abone mantığında haberleşmenin avantajları ve dezavantajları ne olabilir?
- Sabit mesaj tipinden farklı olarak özel mesaj tipi ne avantajlar sağlar?
- ROS servisi ile yayıncı/abone düğümü arasında ne fark vardır? İkisi birbirinin alternatifi olabilir mi?
- ROS eylem yapısı ile ROS servis arasında ne fark vardır? Yine ikisi birbirinin alternatifi olabilir mi?

Bu soruların cevaplarına göre farklı farklı uygulamaların tasarlanabileceğine dair örnekler verilebilir.

5. İLAVE ETKİNLİK

5.1. Örnek Proje Önerileri ve Uygulamaları

Öğrencilerin ROS temel bilgilerini ilerletmek için ROS servis ve eylem yapısının kullanarak farklı uygulamalar yapılabilir. Örneğin bir robotun hareket sınırlaması sağlamak için hangi ROS bileşeni daha verimli olabilir? Bunun için örnek bir proje üzerinde çalışma gerçekleştirilebilir.

7. HAFTA: ROBOT TANIMLAMAYA GİRİŞ

Ön Bilgi:

- Öğrenciler, XML kodu biçimlendirmesini bilir.
- Öğrenciler, Linux ve ROS üzerinde program geliştirme ara yüzünü kullanmıştır.
- Öğrenciler, Linux ve ROS üzerinde paketler ve düğümler oluşturmuştur.
- Öğrenciler, ROS mesaj, servis ve eylem kütüphanelerini ve bunlar ile ilgili kavramları öğrenmiştir.
- Öğrenciler, ROS ortamında robot tanımlama için gerekli adımlarını oluşturmuştur.

Haftanın Kazanımları:

- Öğrenciler bir robotun tanımlanması için gerekli temel kavramları bilir.
- Öğrenciler eklem ve uzuv, görünüm, çarpışma, kinematik ve dinamik tanımlamaları bilir.
- Öğrenciler eklemlerin farklı özelliklerde hareketlendirilmesini bilir ve uygular.
- Öğrenciler uzuvlar ile eklemler arasındaki dönüşüm mantığını anlar ve uygular.
- Öğrenciler bir robotu tanımlarken gerekli uzuv eklem döngü mantığını kurup uygulayabilir.

Haftanın Amacı:

Bu haftanın amacı robot tanımama kavramının tüm öğrenciler tarafından doğru ve benzer şekilde anlaşılmasını sağlamaktır. Ayrıca robot tanımlama etiketlerini ve robot tanımlamasının yapılacağı yazılım geliştirme ara yüzü ile robot uzuv ve eklemlerin görselleştirildiği RVIZ ortamını tanıtmaktır. Robot tanımlama etiketleri kullanılarak bir robotun uzuvları ve eklemleri tanımlandıktan sonra atalet tanımlamalarının yapılması ve robotun uzuvların hareket verilmesi hedeflenmektedir. Öğrencilere tasarlanan robotların belirli bir dönme veya öteleme hareketini yapabilmesi için sınırlama tanımlamasının yapılması, robotların uzuvlarının seri ve paralel hareketlerinin sağlanması bu haftanın bir diğer amacıdır.

Kullanılacak Malzemeler:

Linux işlerim sistemi üzerine ROS

Haftanın İşlenişi:

Gözle: Bir robotun uzuv ve eklemleri üzerine tartışma, iki uzuv arasındaki eklemin hareketi ve dönüşüm ekseni kavramını tartışma. Robot tanımlamadaki temel bileşenleri ve etiketleri tanıma ve uzuv ve eklemlerin görselleştirildiği ***RVIZ*** ara yüzünü tanıma.

Uygula: Robot tanımlama etiketlerini kullanarak uzuv ve eklem oluşturma.

Tasarla: Bir robot tasarlamak için hiyerarşik uzuv ve eklem bağlantı algoritması tasarlama.

Üret: Bir robot tasarımı yapmak.

Değerlendir: Haftanın içeriği ile ilgili yansıtma etkinliği ve değerlendirme.

1. GÖZLE VE UYGULA

1.1. Gözle: Robotik Simülasyon

Robotik simülasyon süreci ile bir robot üretilmeden önce tasarım aşamasının kritik kusurları kontrol edilebilir ve çalışması doğrulanarak ar-ge süreçleri çok kısa tutulabilir. Simülasyon sürecinde hazırlanan sanal robot modeli gerçek donanımın tüm özelliklerine sahip olmalıdır. Robotun şekli gerçek bir robot gibi görünebilir veya görünmeyebilir, ancak gerçek robotun tüm fiziksel özelliklerine sahip sanal bir modeli olmalıdır.

Bir robotun üç boyutlu modelini oluşturarak ROS ortamında simüle etmeyi istiyorsak, robot tasarımını sağlayan ROS paketleri hakkında bilgi sahibi olmamız gereklidir. ROS, robot_model adı verilen ve collada_parser, collada_urdf, joint_state_publisher, kdl_parser, resource_retriever, urdf, urdf_parser_plugin gibi paketlerin olduğu standart bir meta paketlere sahiptir. Bu paketleri kullanarak robot modelleri tasarlanabilir veya oluşturulabilir. Bu paketler, gerçek robotun bileşenlerine ve temel özelliklerine sahip üç boyutlu robot modeli oluşturmamıza yardımcı olur. Daha fazla bilgi için http://wiki.ros.org/robot_model sitesini ziyaret edebilirsiniz.

1.2. Gözle: Robot Modelleme İçin Gerekli ROS Paketleri

ROS, üç boyutlu robot modelleri oluşturmak için bazı paketler sunar. Bir robot modeli oluşturmak ve tasarlamak için yaygın olarak kullanılan bazı ROS paketleri şunlardır.

robot_model: ROS, üç boyutlu (3D) robot modellerinin oluşturulmasına yardımcı olan önemli paketler içeren robot_model adında bir **meta pakete** sahiptir. Bu meta paketin içindeki tüm önemli paketleri görebiliriz:

URDF: robot_model meta paketi içindeki önemli paketlerden biri ‘urdf'dir. URDF paketi, bir robot modelini temsil eden bir XML dosyası olan Birleşik Robot Tanımlama Biçimi (URDF) için bir C++ ayrıştırıcısı içerir.

URDF kullanarak bir robot modeli, sensörler ve bir çalışma ortamı tanımlayabilirsiniz ve URDF çözümleyicileri kullanarak ayrıştırılabilirsiniz. URDF'de yalnızca katı uzuvları ağaç benzeri hiyerarşik bir yapıya sahip bir robotu tanımlayabiliriz. Yani robotun katı uzuvları olacaktır ve eklemler kullanılarak bağlanacaktır. Esnek uzuvlar URDF kullanılarak temsil edilemez. URDF özel XML etiketleri kullanılarak oluşturulur ve bu XML etiketlerini daha fazla işlem için ayrıştırıcı programları kullanarak ayrıştırabiliriz.

joint_state_publisher: Bu araç, URDF ile robot modelleri tasarlarken çok kullanışlıdır. Bu paket, robot modeli tanımlamasını okuyan, tüm eklemleri bulan ve grafik kullanıcı ara yüzü (GUI) kaydırıcılarını (değer değiştiricilerini) kullanarak sabitlenmemiş tüm eklemlere değerleri yayınlayan joint_state_publisher adlı bir **düğüm** içerir. Kullanıcı bu aracı kullanarak her bir robot eklemiyle etkileşime girebilir ve **RViz** kullanarak görselleştirebiliriz. URDF tasarlarken, kullanıcı bu aracı kullanarak her bir eklemin dönüşü/hareketi ile eksenel dönüşümünü doğrulayabilir.

robot_state_publisher: Bu paket, mevcut robot eklem durumlarını okur ve URDF'den kinematik ağaç yapısını kullanarak her bir robot uzvunun 3D pozlarını/pozisyonlarını yayınlar. Robotun üç boyutlu

(3B) pozu ROS tf (dönüşüm) olarak yayınlanır. ROS tf, bir robotun koordinat çerçeveleri arasındaki ilişkiyi yayınlar.

kdl_parser: Kinematik ve Dinamik Kütüphanesi (KDL), URDF'yi temsil eden yapıdan bir KDL ağacı oluşturmak için çözümleyici (parser) araçları içeren bir ROS paketidir. Kinematik ağaç, eklem durumlarını yayınlamak ve ayrıca robotun ileri ve ters kinematiği için kullanılabilir.

xacro: Xacro, XML Makroları anlamına gelir ve xacro'nun URDF'nin ilave eklentilerine eşit olduğunu tanımlayabiliriz. Bu makrolar, URDF'yi daha kısa, okunabilir hale getirmek için bazı eklentiler içerir ve karmaşık robot tanımlamaları oluşturmak için kullanılır. Bazı ROS araçlarını kullanarak xacro'yu her zaman URDF'ye dönüştürebiliriz. Önümüzdeki bölümlerde xacro ve kullanımı hakkında daha fazla bilgi ve uygulama göreceğiz.

1.3. Gözle: Üniversal Robot Tanımlama Formatını (URDF) Anlama

URDF, robotun kinematik ve dinamik tanımını, robotun görsel temsilini ve robotun çarpışma modelini temsil eder. Bir URDF robot modeli oluşturmak için yaygın olarak kullanılan URDF etiketleri bilinmelidir. Bu bölümde, robotun modellenmesine yardımcı olan URDF XML etiketlerine daha ayrıntılı bakacağız. Bu bölümde bir dosya oluşturacak ve robottaki her bir uzuv ve eklem arasındaki ilişkiyi yazarak dosyayı "**.urdf**" uzantısıyla kaydedeceğiz. Daha fazla bilgi için http://wiki.ros.org/urdf bağlantısını ziyaret edebilirsiniz.

1.4. Gözle: URDF XML Yapısı ve Özellikleri

Bir robot tanımlamada kullanılan üst seviye temel etiketler ve özellikleri şunlardır:

Robot: Bir robotun tüm özelliklerini tanımlar.

Sensor/proposals: Kamera, lazer sensörü vb. gibi bir sensörleri tanımlar.

Link: Bir uzvun kinematik ve dinamik özelliklerini tanımlar.

Transmission: Transmisyon bir diğer ifade ile aktarıcılar, şanzımanla aktüatörleri uzuvlara bağlar ve sistemin mekanik bağlantılarını temsil eder.

Joint: Bir eklemin kinematik ve dinamik özelliklerini tanımlar.

Gazebo: Hareket, sönümleme, sürtünme vb. gibi simülasyon özelliklerini tanımlar.

Model_state: Bir modelin durumunu belirli bir zamanda tanımlar.

Model: Bir robot yapısının kinematik ve dinamik özelliklerini tanımlar.

1.5. Gözle: <robot> Etiketi

Bu etiket, URDF kullanılarak temsil edilebilecek tüm robot modelini kapsar. Robot etiketinin içinde robotun adını, uzuvlarını ve robotun eklemlerini tanımlayabiliriz. Bir robot tanımlama dosyasındaki kök eleman bir robot olmalı ve diğer tüm elemanları içine almalıdır. Bir robot modeli kendine bağlı uzuvlardan ve eklemlerden oluşur. Robot modelinin bir tanımlaması şu şekildedir:

```
<robot name=”robotum”>
  <link> ..... </link>
  <link> ..... </link>
```

```
  <joint> ....... </joint>
  <joint> ........</joint>
</robot>
```

<robot> etiketinin alt elemanları şunlardır:

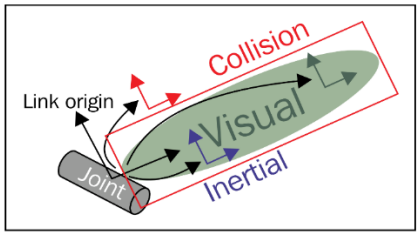
1. **<link>:** Robotun bir uzvudur ve kendi çerçevesiyle tanımlanır.
2. **<joint>:** Uzvun eklemidir ve zorunlu olarak bir eklem tanımı yapılmalıdır.
3. **<transmission>:** İlgili uzvun hareket ettirilmesi için tanımlanır.
4. **<gazebo>:** Gazebo, robotun simülasyon eklentisidir.

1.6. Gözle: <link> (Uzuv) Etiketi/Elemanı:

Link etiketi bir robotun tek bir uzvunu temsil eder. Bu etiketi kullanarak bir robot uzvunu ve özelliklerini modelleyebiliriz. Modelleme boyut, şekil ve renk içerir, şekilsel model aynı zamanda robot uzvunu temsil etmek için üç boyutlu (3D) ağ (mesh) ile robot model içerisine aktarılabilir. Ayrıca atalet matrisi ve çarpışma özellikleri gibi uzvun dinamik özelliklerini de sağlalar. Link etiketinin sözdizimi aşağıdaki gibidir:

```
<link name="uzuv_adi">
    <inertial>...........</inertial>
    <visual> ............</visual>
    <collision>..........</collision>
</link>
```

Şekil 29'da verilen görselde tek bir uzvun görünümü verilmiştir. Bu görsel robotun gerçek uzvunu temsil eder ve gerçek uzvu çevreleyen kırmızı alan ise çarpışma sınırlarıdır. Çarpışma sınırları, gerçek uzva çarpmadan önce çarpışmayı tespit etmek için oluşturulan bir çerçevedir ve gerçek uzvu içine alır.



Şekil 29. *URDF Uzuv (link) ve Eklem (joint) Özellikleri Gösterimi (http://wiki.ros.org/urdf/XML/link)*

```
<link name="kol_uzvu">
   <inertial>
      <origin xyz="0 0 0.5" rpy="0 0 0"/>
      <mass value="1"/>
      <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
   </inertial>
   <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
            <box size="1 1 1" />
      </geometry>
      <material name="Cyan">
            <color rgba="0 1.0 1.0 1.0"/>
     </material>
```

```
   </visual>
   <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
             <cylinder radius="1" length="0.5"/>
      </geometry>
    </collision>
 </link>
```

<link> (Eklem) Etiketi ve Alt Bileşenleri Özellikleri

"name" uzva verilen addır ve mutlaka tanımlanmalıdır.

```
<link name="kol_uzvu">
…….
</link>
```

<link> <inertial>

<inertial> etiketi ile uzvun ataletsel özellikleri tanımlanır ve bu tanımlama isteğe bağlıdır.

<origin> etiketi uzuv referans çerçevesine göre atalet referans çerçevesinin pozisyonudur. Atalet referans çerçevesinin merkezi, ağırlık merkezinde olmalıdır. Atalet referans çerçevesi eksenlerinin, ataletin ana eksenleri ile hizalanması gerekmez.

xyz: x, y, z doğrultusundaki ofseti temsil eder ve varsayılan olarak sıfır vektöründedir. Bu etiket isteğe bağlıdır.

rpy: Radyan cinsinden sabit eksenli yuvarlanma, yunuslama ve sapma (roll, pitch, yaw) açılarını temsil eder. Bu etiket isteğe bağlıdır.

<mass> etiketi ile uzvun kütlesi tanımlanır. Bu öğe değer özelliğiyle (value attribute) temsil edilir.

<inertia>: Atalet çerçevesinde temsil edilen 3x3'lük rotasyonel atalet matrisidir. Rotasyonel atalet matrisi simetrik olduğundan, burada bu matrisin sadece 6 üst köşegen elemanı ixx, ixy, ixz, iyy, iyz, izz özellikleri kullanılarak belirtilir. Geometrik simetriye sahip basit nesneler için nesne boyunca sabit yoğunluğa sahip simetrik kütle dağılımları ile ilgili detaylı bilgi için "https://en.wikipedia.org/wiki/List_of_moments_of_inertia#List_of_3D_inertia_tensors" bağlantısını ziyaret edebilirsiniz.

<link> < visual>

<visual> Bir uzvun görsel özelliklerini tanımlar. Bu etiket, görselleştirme amacıyla nesnenin şeklini (kutu, silindir vb.) belirtir. Bazı durumlarda aynı uzuv için birden fazla <visual> etiketi bulunabilir. Bu durumda tanımlanan geometrinin birleşimi uzvun görsel temsilini oluşturur.

name: Visual etiketinin isteğe bağlı parametresidir. Uzvun geometrisinin bir kısmı için bir ad belirtir. Bu bir uzvun geometrisinin belirli bölümlerine atıfta bulunmak için kullanılabilir.

<link> <visual> <origin>

<origin> Referans uzvun çerçevesine bağlı olarak görsel elemanın referans çerçevesini belirlemek için kullanılır. İsteğe bağlı bir etikettir. Bu etiketin parametreleri şunlardır:

xyz: x, y, z doğrultusundaki ofseti temsil eder. İsteğe bağlı bir tanımlamadır ve varsayılan olarak sıfır vektörüdür.

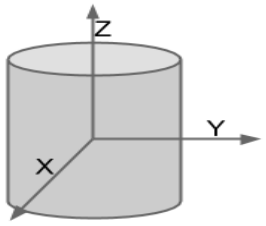
rpy: Radyan cinsinden sabit eksenli yuvarlanma, yunuslama ve sapma (roll, pitch, yaw) açılarını temsil eder. İsteğe bağlı bir tanımlamadır.

<link> <visual> <geometry>

<geometry> Tanımlaması zorunlu bir alandır. Nesnenin görsel şeklini tanımlar. Bu şekiller şunlardan biri olabilir;

<box>: Bir kutu tanımlamasında kullanılır, "size" parametresi ile birlikte tanımlanır ve kutunun üç kenar uzunluğunu içerir. Kutunun orijini merkezindedir.

<cylinder>: Bir silindir tanımlamasında kullanılır, "radius" (yarıçap) ve "length" (uzunluk) parametresi ile birlikte tanımlanır. Silindirin orijini merkezindedir. Şekil 30'da oluşturulacak silindir nesnesi örneği verilmiştir.



Şekil 30. *Silindir Nesnesi Eksen Tanımlaması (http://wiki.ros.org/urdf/XML/link)*

<sphere>: Bir küre tanımlamasında "radius" (yarıçap) parametresi ile birlikte kullanılır. Kürenin orijini merkezindedir.

<mesh>: Farklı bir tasarım programında tasarlanmış uzuvların mesh adı verilen bir ağ yapısı oluşturularak bir şekil görseli dışarıdan eklenir. "scale" parametresi ile ölçekleme yapılabilir. "filename" parametresi ile dışarıdaki bir ağ yapısına sahip (mesh) dosya yolu belirtilir. En iyi doku ve renk desteği için Collada.dae dosyalarının kullanılması önerilir bununla birlikte ".stl" dosyaları da desteklenir. Mesh dosyası proje içerisinde kayıtlı yerel bir dosya olmalıdır.

<link> <visual> <material>

<material> İsteğe bağlı bir tanımlamadır. Görsel elemanların malzeme tanımlaması yapılır. "link" yani uzuv nesnesinin dışında, üst düzey 'robot' öğesinde bir malzeme öğesi olarak belirtilebilir. Daha sonra bir uzuv öğesinin içinden ilgili malzeme adı ile çağrılabilir.

name: Malzeme adı parametresidir. Malzemeyi tanımlamak için kullanılır.

<color> Malzemenin rengini tanımlamada kullanılır ve isteğe bağlı bir etikettir. Bu etiket rgba parametresini alır. RGBA her biri [0,1] aralığında kırmızı/yeşil/mavi/alfa'yı (red/green/blue/alpha) temsil eden dört sayı kümesiyle belirtilen bir rengi tanımlar.

<texture> Malzemenin dokusunu tanımlamada kullanılır ve isteğe bağlı bir etikettir. Bu etiket "filename" parametresini alır. Bir malzemenin dokusu" filename" parametresindeki dosya adıyla çağrılır.

<link> <collision>

<collision> Bir uzvun çarpışma özelliklerini tanımlar ve isteğe bağlı bir etikettir. Çarpışma etiketi genellikle uzvun görsel (visual) etiketindeki özellikler ile aynı olabildiği gibi farklı olarak da

tanımlanabilir. Örneğin uzvun tamamı için değil sadece bir kısmı için de tanımlanabilir. Bunun yanında hesaplama süresini azaltmak için genellikle daha basit çarpışma modelleri kullanılır. Bazı durumlar için bir uzuv için birden fazla <collision> (çarpışma) etiketi örneği bulunabilir. Bu durumda tanımlanan geometrinin birleşmesi, uzvun çarpışma geometrisini temsil eder.

name: <collision> etiketi isteğe bağlı olarak uzuv geometrisinin bir kısmı için bir ad belirtilebilir. Bu parametre bir uzuv geometrisinin belirli bölümlerini tanımlamak için kullanılır.

<link> <collision> <origin>

<origin> etiketi isteğe bağlı bir etikettir. Uzvun referans çerçevesine göre çarpışma elemanının referans çerçevesini belirler.

xyz: İsteğe bağlıdır ve varsayılan olarak sıfır vektörüdür. x, y, z doğrultundaki ofseti temsil eder.

rpy: İsteğe bağlıdır. Radyan cinsinden sabit eksenli yuvarlanma, yunuslama ve sapma (roll, pitch,yaw) açılarını temsil eder.

<link> <collision> <geometry>

<geometry> etiketi için <link> <visual> <geometry> etiketindeki aynı özellikler kullanılmaktadır.

Önerilen Ağ (Mesh) Çözünürlüğü

ROS hareket planlama paketlerini kullanarak çarpışma kontrolü için, URDF'ye yerleştirdiğiniz çarpışma ağları (mesh) için (ideal olarak 1000'den az) uzuv başına mümkün olduğunca az yüzey alanı veya ağ (mesh) önerilir.

Çoklu Çarpışma Parçaları

URDF'lerin, bazen arzu edilmesine rağmen, birden fazla çarpışma parçası grubunu desteklememesi gerektiğine karar verilmiştir. Bir URDF'de, <visual> elemanları gerçek robot için olabildiğince doğru olmalı ve <collision> elemanları ise ağlarda (mesh) çok daha az üçgensel yapı ile yakın bir yaklaşım içermelidir.

Çarpışma kontrolü ve denetimi için aşırı büyük boyutlu çarpışma geometrilerine ihtiyacınız varsa, bu ağları/geometrileri özel XML öğelerine taşıyabilir ya da ayrıştırabilirsiniz. Örneğin denetleyicilerinizin özel bir kaba çarpışma kontrol geometrisine ihtiyacı varsa, <collision> öğesinden sonra <collision_checking> etiketini ekleyebilirsiniz:

```
<link name="robot_kolu">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
             <mesh filename="package://robot_modeli/meshes/base_link.DAE"/>
      </geometry>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="-0.065 0 0.0"/>
      <geometry>
        <mesh filename="package://robot_ modeli/meshes/base_link_simple.DAE"/>
      </geometry>
    </collision>
    <collision_checking>
      <origin rpy="0 0 0" xyz="-0.065 0 0.0"/>
      <geometry>
```

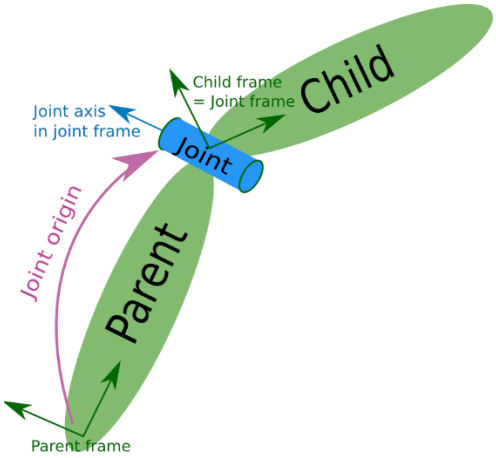
```
            <cylinder length="0.7" radius="0.27"/>
      </geometry>
    </collision_checking>
    <inertial>
      ...
    </inertial>
  </link>
```

1.7. Uygula: <link> Etiketi ve Özelliklerini Kullanarak Bir Robot Uzvu Tasarlama

Öğrencilerden <link> uzuv tanımlama etiketi ve bu etiketin bileşenlerini kullanarak silindirik ve kare şeklinde uzuvlar tasarlamaları istenir. Uzuv tasarımları yapılırken her bir alt etiketin robot uzvu üzerindeki etkisinin görülmesi istenir. Öğrencilerden robot uzuvlarının RVIZ ortamında görüntülenmesi istenir ve RVIZ ortamının sağladığı faydaların anlaşılması sağlanır.

1.8. Gözle: <joint> (Eklem) Etiketi/Elemanı

Joint etiketi bir robot uzvunun eklemini temsil eder. İki uzuv arasında bir URDF eklemi oluşturulur. İlki ebeveyn (parent) uzvu, ikincisi çocuk (child) uzvudur. Aşağıda bir eklemin ve iki adet uzvun bir resmi verilmiştir. Eklemin kinematik ve dinamikleri ile eklem hareketinin ve hızının sınırlarını belirleyebiliriz. Eklem elemanı, eklemin kinematiğini ve dinamiğini tanımlar ve ayrıca eklemin güvenlik sınırlarını belirtir. Eklem etiketi döner, sürekli, prizmatik, sabit, yüzer ve düzlemsel (revolute, continuous, prismatic, fixed, floating, planar) gibi farklı eklem hareket tiplerini destekler. Şekil 31'de uzuvlar arası eklemin konumu ve temsili verilmiştir.



Şekil 31. *Uzuvlar Arası Eklem (Joint) Bağlantısı (http://wiki.ros.org/urdf/XML/joint)*

Etiketin sözdizimi aşağıdaki gibidir:

```
<joint name="eklem_adi">
  <parent link="uzuv1"/>
  <child link="uzuv2"/>
  <calibration .... />
  <dynamics damping ..../>
  <limit effort .... />
</joint>
```

Örnek Sözdizimi

```
<joint name="eklem1" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
```

```
    <parent link="uzuv1"/>
    <child link=" uzuv2"/>
    <calibration rising="0.0"/>
    <dynamics damping="0.0" friction="0.0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
    <safety_controller k_velocity="10" k_position="15" soft_lower_limit="-2.0"
     soft_upper_limit="0.5" />
 </joint>
```

Özellikler

Eklem elemanının iki özelliği vardır. Bunlar eklemin adı olan “name” ve eklemin hareket tipi olan “type” ’dir. “name” özelliği zorunludur ve mutlaka benzersiz bir ad ile tanımlanmalıdır. “type” özelliği ise yine zorunlu bir alandır ve eklemin türünü belirtir. Eklem türü olarak aşağıdakilerden biri belirlenir.

Revolute (döner): Eksen boyunca dönen ve üst ve alt sınırlarla belirtilen sınırlı bir aralığa sahip bir menteşe (hinge) eklemidir.

Continuous (sürekli): Eksen etrafında sürekli dönen, üst ve alt sınırları olmayan sürekli menteşe eklemidir.

Prismatic (prizmatik): Eksen boyunca kayan ve üst ve alt sınırlarla belirtilen sınırlı bir aralığa sahip kayan bir eklemdir.

Fixed (sabit): Bu normalde gerçek bir eklem değildir. Çünkü hareket etmez ve sabittir. Bütün serbestlik dereceleri kilitlidir. Bu tür bir eklem ekseni, kalibrasyonu, dinamikleri, limitleri veya güvenlik kontrolü (safety_controller) gerektirmez.

Floating (yüzer): Bu eklem 6 serbestlik derecesinin tamamı için harekete izin verir.

Planar (düzlemsel): Bu eklem, eksene dik bir düzlemde harekete izin verir.

<Joint><origin>

<origin> etiketi isteğe bağlıdır. Bu etiket ebeveyn eklemden çocuk ekleme dönüşümü tanımlar. Eklem, yukarıdaki şekilde gösterildiği gibi çocuk eklemin orijininde bulunur.

xyz özelliği isteğe bağlıdır ve varsayılan olarak sıfır vektöründedir. Bu etiket x, y ve z ofsetlerini temsil eder. Tüm pozisyonlar metre cinsinden belirtilir.

rpy özelliği yine isteğe bağlıdır ve aksi belirtilmedikçe varsayılan olarak sıfır vektöründedir. İlgili eksen etrafındaki dönüşü temsil eder. Sırasına göre önce x etrafında döner, yani yuvarlanma (roll), sonra y etrafında eğilir, yunuslama yapar (pitch) ve son olarak z etrafında sapar (yaw). Tüm açılar radyan cinsinden belirtilir.

<Joint><parent>

<parent> etiketi zorunlu olarak tanımlanır. Bu etiket ebeveyn (parent) uzvunu belirtir.

<parent link="ebeveyn_uzuv"/> şeklinde tanımlanır.

<Joint><child>

<child> etiketi zorunlu olarak tanımlanır. Bu etiket çocuk (child) uzvunu belirtir.

< child link="cocuk_uzuv"/> şeklinde tanımlanır.

<Joint><axis>

<axis> etiketi isteğe bağlıdır ve varsayılan olarak (1,0,0) konumu tanımlıdır. Bu, döner (revolute) eklemler için dönme eksenini, prizmatik eklemler için öteleme eksenini ve düzlemsel (planar) eklemler için normal yüzeyi tanımlar. Eksen, ortak referans çerçevesinde belirtilir. Sabit ve hareketli eklemler eksen alanını kullanmaz.

xyz özelliği zorunlu bir alandır. Bir vektörün x, y ve z bileşenlerini temsil eder. Bu vektörler normalleştirilmelidir.

<Joint><calibration>

<calibration> etiketi isteğe bağlıdır. Eklemin mutlak konumunu kalibre etmek için kullanılan eklemin referans konumlarıdır ve aşağıdaki özellikler ile birlikte kullanılır.

Rising (yükselen): isteğe bağlı bir özelliktir. Eklem pozitif yönde hareket ettiğinde, referans konumu yükselen bir kenarı tetikler.

Falling (düşen): isteğe bağlı bir özelliktir. Eklem negatif yönde hareket ettiğinde, referans konumu düşen bir kenarı tetikler.

<Joint><dynamics>

<dynamics> etiketi isteğe bağlı bir etikettir. Eklemin fiziksel özelliklerini belirten bir elemandır. Bu değerler, özellikle simülasyon için yararlı olan eklemin modelleme özelliklerini belirtmek için kullanılır ve aşağıdaki özellikler ile beraber kullanılır.

Damping: isteğe bağlı bir özelliktir ve varsayılan olarak 0'dır. Eklemin fiziksel sönümleme değeridir. ((N.s)/m prizmatik eklemler, (N.m.s)/rad döner eklemler içindir.

Friction (sürtünme): isteğe bağlı bir özelliktir ve varsayılan olarak 0'dır. Eklemin fiziksel statik sürtünme değeridir ve N prizmatik eklemler için ve N*m döner eklemler içindir.

<Joint><limit>

<limit> etiketi sadece döner (revolute) ve prizmatik eklem için zorunludur. Diğer eklem türleri için zorunlu değildir. Bu etiket aşağıdaki özellikleri içerir.

Lower (alt limit): isteğe bağlı bir özelliktir ve varsayılan olarak 0'dır. Eklem alt limit sınırını belirten bir özelliktir (döner eklemler için radyan, prizmatik eklemler için metredir). Eklem sürekli (continuous) ise bu özellik kullanılmaz.

Upper (üst limit): isteğe bağlı bir özelliktir ve varsayılan olarak 0'dır. Eklem üst limit sınırını belirten bir özelliktir (döner eklemler için radyan, prizmatik eklemler için metredir). Eklem sürekli (continuous) ise bu özellik kullanılmaz.

Effort (efor, çaba) özelliği zorunlu olarak tanımlanır. Maksimum eklem eforu uygulamak için kullanılan bir özelliktir. Velocity (hız) özelliği zorunlu olarak tanımlanır. Maksimum eklem hızını tanımlamak için kullanılan bir özelliktir.

<Joint><mimic>

<mimic> isteğe bağlı bir özelliktir ve ROS Groovy ile birlikte kullanılmaya başlanmıştır. Bu etiket, tanımlanan eklemin başka bir mevcut eklemi taklit ettiğini belirtmek için kullanılır. Bu eklemin değeri

“değer = katsayı * diğerEklemDeğeri + ofset” (value = multiplier * other_joint_value + ofset) olarak hesaplanabilir. Bu etiket ile birlikte kullanılan özellikler şunlardır:

‘joint’ özelliği zorunlu olarak tanımlanır. Bu taklit edilecek eklemin adını belirtir.

‘multiplier’ isteğe bağlı bir özelliktir. İlgili formülde çarpım faktörünü belirtir.

‘offset’ isteğe bağlı bir özelliktir. İlgili formüle eklenecek ofseti belirtir. Varsayılan değer 0'dır ve döner eklemler için radyan, prizmatik eklemler için metre cinsinden tanımlama yapılır.

<Joint><safety_controller >

<safety_controller> etiketi isteğe bağlı bir etikettir. Bu etiket ile birlikte aşağıdaki özellikler kullanılır.

soft_lower_limit: İsteğe bağlı bir özelliktir ve varsayılan olarak 0'dır. Güvenlik denetleyicisinin eklemin konumunu sınırlamaya başladığı alt eklem sınırını belirten bir özelliktir. Bu sınırın alt eklem sınırından daha büyük olması gerekir.

soft_upper_limit: İsteğe bağlı bir özelliktir ve varsayılan 0'dır. Güvenlik kontrolörünün eklemin pozisyonunu sınırlamaya başladığı üst eklem sınırını belirten bir özellik. Bu sınırın üst eklem sınırından daha küçük olması gerekir.

k_position: İsteğe bağlı bir özelliktir ve varsayılan 0'dır. Konum ve hız sınırları arasındaki ilişkiyi belirten bir özelliktir.

k_velocity: Bu özelliği tanımlamak zorunludur. Efor ve hız sınırları arasındaki ilişkiyi belirten bir özelliktir.

1.9. Uygula: <joint> Etiketi ve Özelliklerini Kullanarak Bir Eklem Tanımlama

Öğrencilerden <link> uzuv tanımlama etiketi ve bu etiketin bileşenlerini kullanarak oluşturulmuş olan uzuvlar arasında bir eklem <joint> ve eklem türü tanımlamaları istenir. Uzuvlar arasında eklem tanımlamaları her ebeveyn uzuv ve çocuk uzuv arasındaki etkileşimin görülmesi sağlanır. Öğrencilerden tanımladıkları robot uzuvlarının, eklemlerinin ve eklem hareketlerinin RVIZ ortamında görüntülenmesi istenir.

1.10. Gözle: <transmission> (Transmisyon/Aktarım) Etiketi/Elemanı

Transmisyon elemanı, bir aktüatör ile bir eklem arasındaki ilişkiyi tanımlamak için kullanılan bir URDF etiketidir. Bu etiket dişli oranları ve paralel bağlantılar gibi kavramların modellenmesini sağlar. Bir aktarım/iletim, efor/akış değişkenlerini, çıkış gücü sabit kalacak şekilde dönüştürür. Birden fazla aktüatör, karmaşık transmisyon yoluyla birden fazla ekleme bağlanabilir.

Bir transmisyon sözdizimine örnek:

```
<transmission name="motor_1_aktarim">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="sag_motor_eklemi">
         <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="eklem1_motor">
    <mechanicalReduction>50</mechanicalReduction>
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </actuator>
</transmission>
```

<transmission> etiketinin özellikleri şunlardır.

Transmisyon etiketinin “name” özelliği bir aktarımın/transmisyonun benzersiz adını belirtir ve zorunlu olarak tanımlanmalıdır.

Örnek: `<transmission name="motor_1_aktarim">`

<transmission> <type>

<type> etiketi aktarım/transmisyon türünü belirtir.

Örnek: `<type>transmission_interface/SimpleTransmission</type>`

<transmission><joint>

<joint> etiketi aktarımın/transmisyonun bağlı olduğu bir eklemi tanımlar ve aşağıdaki alt etiketler ve özellikler ile tanımlanır.

<hardwareInterface> etiketi desteklenen bir eklem alanı donanım arabirimini belirtir. Bu aktarım/transmisyon Gazebo'ya yüklendiğinde bu etiketin değerinin EffortJointInterface ve bu transmisyon RobotHW'a yüklendiğinde Hardware_interface/EffortJointInterface olması gerekmektedir.

<transmission><actuator>

<actuator> etiketi aktarımın/transmisyonun bağlı olduğu bir aktüatördür ve aşağıdaki alt etiketler ve özellikler ile tanımlanır.

<mechanicalReduction> alt etiketi isteğe bağlı bir etikettir. Mafsal / aktüatör aktarımında mekanik bir redüksiyon belirler. Bu etiket tüm aktarım/transmisyon tipleri için gerekli olmayabilir.

<hardwareInterface> alt etiketi isteğe bağlı bir etikettir. Desteklenen bir eklem alan donanım arabirimini belirtir. <hardwareInterface> etiketinin burada yalnızca Indigo'dan önceki ROS sürümleri için belirtilmesi gerektiğini unutmayın. Bu etiketi belirtmek için doğru yer <joint> etiketidir.

1.11. Uygula: <transmission> Etiketini Kullanarak Aktarım Tanımlama

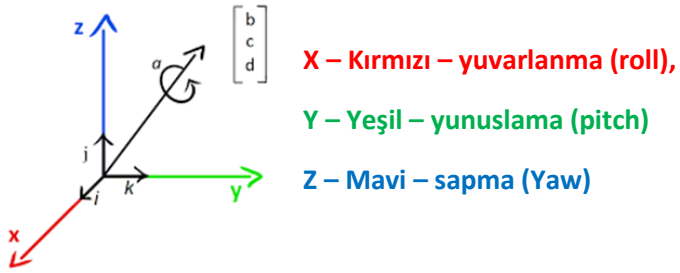
Öğrencilerden uzuv ve eklem tanımlaması yapılmış bir robot koluna eklem tanımlama etiketi ve bu etiketin bileşenlerini kullanarak oluşturulmuş olan uzuvlar arasında bir eklem ve eklem türü tanımlamaları istenir. Uzuvlar arasında eklem tanımlamaları her ebeveyn uzuv ve çocuk uzuv arasındaki etkileşimin görülmesi sağlanır. Öğrencilerden tanımladıkları robot uzuvlarının, eklemlerinin ve eklem hareketlerinin RVIZ ortamında görüntülenmesi istenir.

İlk URDF modelimiz

Tasarlayacağımız ilk robot mekanizması, aşağıdaki şekilde gösterildiği gibi bir robot kolu mekanizmasıdır.

Bu mekanizmada üç uzuv ve iki eklem vardır. Ana uzuv (base_link) diğer tüm uzuvların monte edildiği statik bir uzuvdur. İlk uzuv kendi ekseni üzerinde kaydırma yapabilir, ikinci uzuv birinci uzva monte

edilir ve kendi ekseni üzerinde eğilebilir. Bu sistemdeki iki eklemde döner (revolute) tiptedir. Şekil 32'de URDF için eksen tanımları verilmiştir.



Şekil 32. *Dönüşüm Eksenleri (TF)(https://spl.hevs.io/spl-docs/tools/ros/tf2.html)*

Robot Modeli

```
<?xml version="1.0" encoding="utf-8"?>
<robot name="robot_kolu">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.01" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <material name="yellow">
        <color rgba="1 1 0 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <cylinder length="0.03" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </collision>
    <inertial>
      <mass value="1"/>
      <inertia ixx="0.5" ixy="0.0" ixz="0.0" iyy="0.5" iyz="0.0" izz="0.5"/>
    </inertial>
  </link>
  <joint name="govde_eklemi" type="revolute">
    <parent link="base_link"/>
    <child link="govde_uzvu"/>
    <origin xyz="0 0 0.1"/>
    <axis xyz="0 0 1"/>
    <limit effort="300" lower="-3.14" upper="3.14" velocity="0.1"/>
    <dynamics damping="50" friction="1"/>
  </joint>
  <link name="govde_uzvu">
    <visual>
      <geometry>
        <box size="0.08 0.08 0.5"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.15"/>
      <material name="red">
        <color rgba="0 0 1 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <box size="0.08 0.08 0.5"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.05"/>
    </collision>
    <inertial>
      <mass value="1"/>
      <inertia ixx="0.5" ixy="0.0" ixz="0.0" iyy="0.5" iyz="0.0" izz="0.5"/>
    </inertial>
```

```
  </link>

  <joint name="kol_eklemi" type="revolute">
    <parent link="govde_uzvu"/>
    <child link="kol_uzvu"/>
    <origin xyz="0.04 0 0.36"/>
    <axis xyz="0 1 0"/>
    <limit effort="300" lower="1.57" upper="-1.57" velocity="0.1"/>
    <dynamics damping="50" friction="1"/>
  </joint>

  <link name="kol_uzvu">
    <visual>
      <geometry>
        <cylinder length="0.3" radius="0.04"/>
      </geometry>
      <origin rpy="0 1.57 0" xyz="0.15 0 0"/>
      <material name="green">
        <color rgba="1 0 0 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <cylinder length="0.4" radius="0.24"/>
      </geometry>
      <origin rpy="0 1.5 0" xyz="0 0 0"/>
    </collision>
    <inertial>
      <mass value="1"/>
      <inertia ixx="0.5" ixy="0.0" ixz="0.0" iyy="0.5" iyz="0.0" izz="0.5"/>
    </inertial>
  </link>
</robot>
```

İnceleme

Kodu kontrol ettiğimizde tanımlamanın en üstünde bir <robot> etiketi eklendiğini görüyoruz.

```
<?xml version="1.0"?>
<robot name="robot_kolu">
```

<robot> etiketi, oluşturduğumuz robotun adını tanımlar. Burada robot adına “robot_kolu” adını verdik.

<robot> etiket tanımından sonraki bölümleri kontrol edersek, gövde ve kol mekanizmasının uzuv ve eklem tanımlarını görebiliriz:

Önceki kod parçacığı, gövde ve kol mekanizmasının base_link tanımıdır. Base_link genel olarak ilk ana uzuv olarak tanımlanır.

<visual> etiketi, robot simülasyonunda gösterilen uzvun görsel görünümünü tanımlar. Bu etiketi kullanarak uzuv geometrisini (silindir, kutu, küre veya ağ (mesh)) ve uzvun malzemesini (renk ve doku) tanımlayabiliriz:

```
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.01" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <material name="yellow">
```

```
      <color rgba="1 1 0 1"/>
    </material>
  </visual>
```

Önceki kod parçacığında, benzersiz bir ad'a ve eklem tipine sahip bir eklem tanımladık. Burada kullandığımız eklem türü döner eklemdir (revolute) ve üst/ebeveyn uzuv ve alt/çocuk uzuv sırasıyla base_link ve govde_uzvu'dur. Eklem orijini de bu etiketin içinde belirtilir.

Önceki URDF kodunu ***robot_kolu_urdf.xml*** veya ***robot_kolu.urdf*** olarak kaydedin ve URDF'nin aşağıdaki komutu kullanarak hata içerip içermediğini kontrol edin:

```
$ check_urdf robot_kolu_urdf.xml
```

Check_urdf komutu urdf'yi çözümler ve varsa hatayı gösterir. Her şey yolundaysa, Şekil 33'de verilen bir çıktı gösterecektir:

```
ubuntu@ubuntu1804:~/catkin_ws/src/robot_ornegi/urdf$ check_urdf robot_kolu_urdf.xml
robot name is: robot_kolu
---------- Successfully Parsed XML ---------------
root Link: base_link has 1 child(ren)
    child(1):  govde_uzvu
        child(1):  kol_uzvu
```

Şekil 33. *URDF Dosyasını Kontrol Etme*

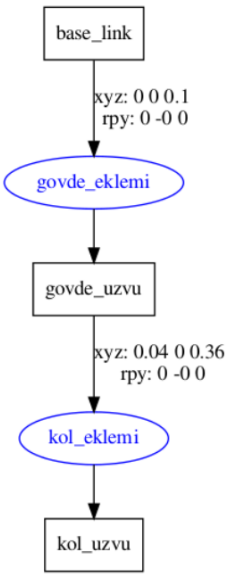
Robot uzuvlarının ve eklemlerinin yapısını grafik olarak görüntülemek istiyorsak, urdf_to_graphiz adlı bir komut aracını kullanabiliriz:

```
 $ urdf_to_graphiz robot_kolu_urdf.xml
```

Bu komut iki dosya oluşturur: Robot_kolu.gv ve robot_kolu.pdf. Aşağıdaki komutu kullanarak bu robotun yapısını görebiliriz:

```
$ evince robot_kolu.pdf
```

Dosya içerisinde aşağıdaki çıktı alınır. Şekil 34'te robotun uzuv ve eklem hiyerarşisi verilmiştir. Buna göre **base_link** ana uzvu temsil etmekte ve eklemler ile diğer uzuvların birbirine bağlandığı görülmektedir.



Resim 34. *Robotun Uzuv ve Eklem Hiyerarşisi*

RViz'de robot 3D modelini görselleştirme

URDF'yi tasarladıktan sonra, robot modelimizi RViz'de görüntüleyebiliriz. Bunun için bir “launch” adında bir klasör oluşturalım ve içerisine "robot_kolu_rviz.launch" adında bir dosya oluşturalım. Aşağıdaki kodları “robot_kolu_rviz.launch” dosyasına ekleyerek bir başlatma dosyası oluşturabiliriz.

```
<?xml version="1.0"?>
<launch>
<arg name="model" />
<param name="robot_description" textfile="$(find robot_ornegi)/urdf/robot_kolu_urdf.xml" />
<node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui" type="joint_state_publisher_gui"
/>
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find robot_ornegi)/rviz/robot_kolu.rviz"
required="true" />
</launch>
```

Modeli aşağıdaki komutu kullanarak başlatabiliriz.

```
$ roslaunch robot_ornegi robot_kolu_rviz.launch
```

Gövde ve kol eklemlerindeki hareketleri kontrol etmek için RViz ara yüzü açılırken kaydırıcılar (slider) içeren ekstra bir grafik ara yüzünün geldiğini görebiliriz. Bu grafik ara yüzüne joint_state_publisher paketinden Joint State Publisher düğümü adı verilir. Bu paketi ve düğümü, başlatma (launch) dosyasında inceleyebiliriz.

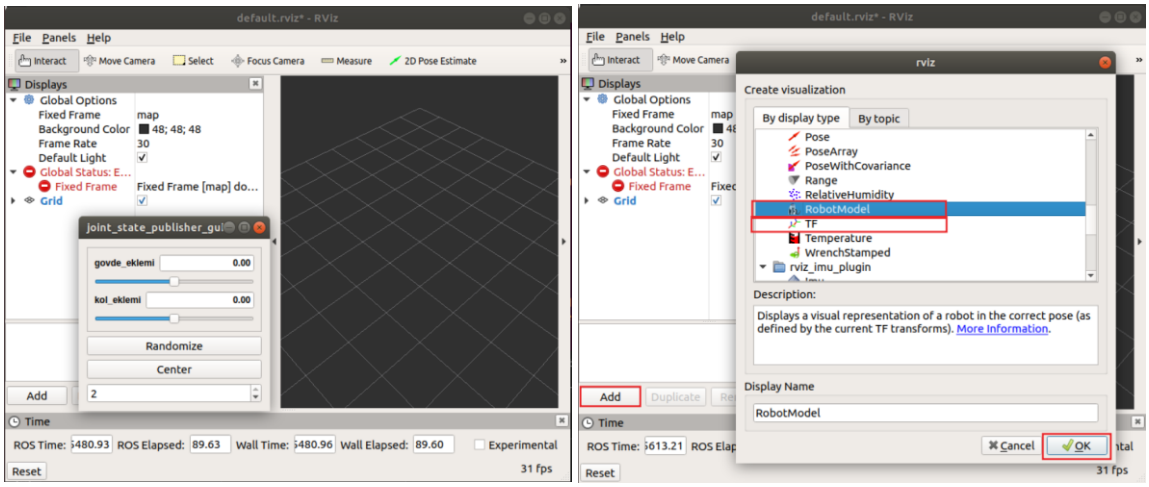
```
<node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui"
type="joint_state_publisher_gui" />
```

Gövde ve kol sınırları, eklem etiketinin içinde belirtilmelidir.

```
  <joint name="govde_eklemi" type="revolute">
    <parent link="base_link"/>
    <child link="govde_uzvu"/>
    <origin xyz="0 0 0.1"/>
    <axis xyz="0 0 1"/>
    <limit effort="300" lower="-3.14" upper="3.14" velocity="0.1"/>
```

```
  <dynamics damping="50" friction="1"/>
</joint>
```

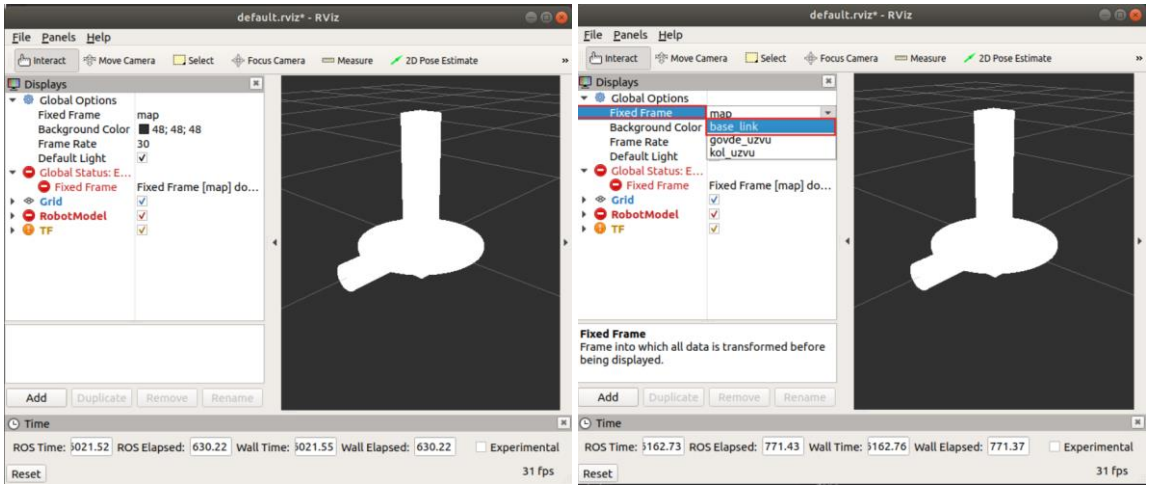
Bu eklem etiketi içerisinde tanımlanır: Kod, efor, hız ve açı sınırlarını tanımlar. Efor, bu eklem tarafından desteklenen maksimum kuvvettir, alt ve üst, döner tip eklem için radyan cinsinden eklemin alt ve üst (lower and upper) sınırını ve prizmatik eklemler için sayaçları ve değerleri gösterir. Hız, maksimum eklem hızıdır. Şekil 35 robot modelinin RVIZ içerisinde ana ekrana eklenmesi gösterilmektedir.



Şekil 35. *Robot Modelinin RVIZ İçerisinde Ana Ekrana Eklenmesi*

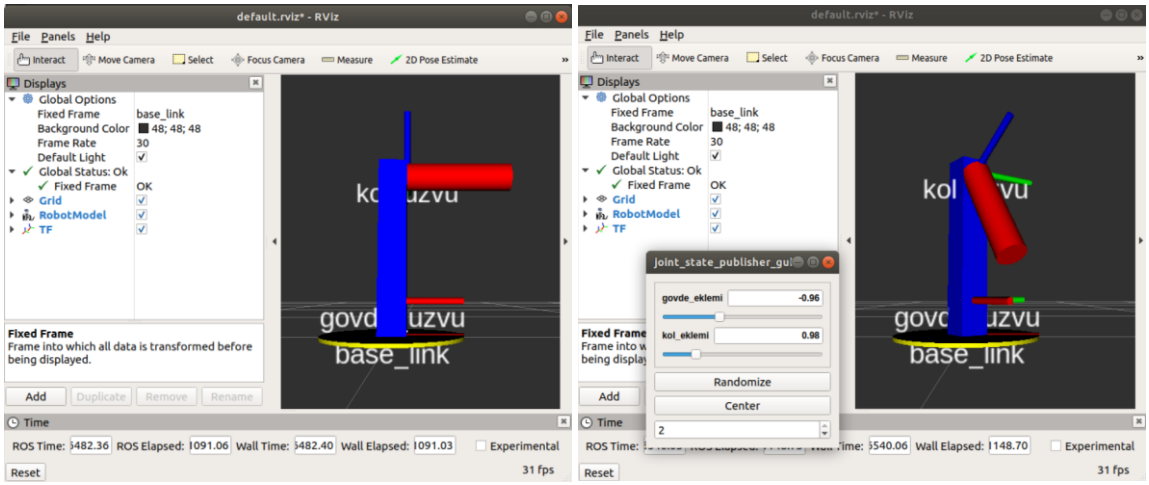
Yukarıdaki ekran görüntüsünde, kaydırıcıların (sliders) ve robotun görselleştirilmesi için açılan RVIZ ara yüzü görülmektedir. Kaydırıcı ekranı ise geçerli eklem değerlerini içeren Joint State Publisher'ın grafik ara yüzünü (GUI) göstermektedir.

RVIZ ekranı karşımıza geldiğinde robot modelimizin ekranda olmadığını görüyoruz. Bunun için hafızada yüklü olan robot modelinin RVIZ ekranına çağrılması gerekmektedir. Bunun için RVIZ ekranındaki “Add” butonuna tıklayarak görüntüleme ağacından “RobotModel” seçilir. Ardından aynı işlemleri tekrar ederek “TF” ara yüzü de seçilir. Böylece robotumuz ekranda aşağıdaki gibi görüntülenir. Resimlerde de görüldüğü gibi robot modeli beyaz renkte ve ayırt edilemeyecek şekilde görülmektedir. Bunun sebebi robotun sabit çerçevesinin (fixed frame) tanımlanmamış olmasındandır. Şekil 36’da robotun sabit çerçeve (Fixed Frame) tanımının yapıldığı görülmektedir.



Şekil 36. *Robotun Sabit Çerçeve (Fixed Frame) Tanımının Yapılması*

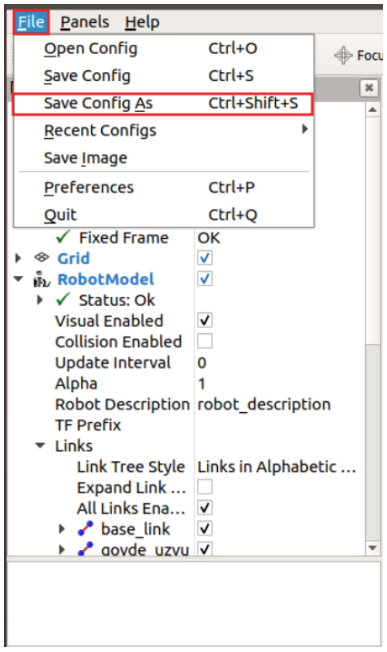
Robotun sabit çerçevesi (fixed frame) “base_link” olarak yani ilk uzuv seçildikten sonra robotun tam olarak görüntüsü alınabilmektedir. Robot, kaydırıcılar vasıtası ile eksen sınırları içerisinde hareket ettirilebilmektedir. Böylece robotun tüm sınırları simülasyon öncesi kontrol edilebilmektedir. Şekil 37’de robot uzuvlarının joint_state_publisher ile hareket ettirilmesi gösterilmektedir.



Şekil 37. *Robot Uzuvlarının joint_state_publisher ile Hareket Ettirilmesi*

Robot modelini RVIZ ortamında görüntüledikten sonra bu görüntüleme konfigürasyonunu ve robot modelini tekrar tekrar yapmak zorunda kalmayacağız. Bunun için “File” menüsünden “Save Config” veya “Save Config As”’i tıklayarak RVIZ uzantısında paket klasörü içerisine “RVIZ” adında bir klasör oluşturarak içerisine kaydedilir. Başlatma dosyası içerisindeki “launch” dosyasının en alt satırına aşağıdaki kod eklenir. Bu RVIZ konfigürasyonunun robot açılışında yüklenmesini sağlamaktadır. Şekil 38’de RVIZ konfigürasyonun sonraki görüntülemeler için kaydedilmesi aşaması gösterilmektedir.

```
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find robot_ornegi)/rviz/robot_kolu.rviz"
required="true" />
```



Şekil 38. *RVIZ Konfigürasyonun Sonraki Görüntülemeler İçin Kaydedilmesi*

Gazebo, V-REP ve benzeri gibi bir robot simülatöründe bir robotu simüle etmeden önce robot uzvunun geometri, renk, kütle ve atalet gibi fiziksel özelliklerini ve eklemin çarpışma özelliklerini tanımlamamız gerekir. Tüm bu özelliklerin robot modelinde eksiksiz şekilde tanımlanması iyi bir simülasyon sonucu elde etmemize yardımcı olmaktadır. Bunun yanında her bir uzvun çarpışma ve atalet (collision, inertia) parametreleri mutlaka olmalıdır. Aksi halde Gazebo robot modelini düzgün yükleyemez.

XACRO

XACRO Kullanarak Robot Modellemesini Anlama

Karmaşık robot modelleriyle çalışırken URDF'nin temel tanımlamalar için esnek olduğu görülmektedir. Ancak URDF tekrar kullanılabilirlik, modülerlik ve programlanabilirlik gibi parametrik unsurları taşımamaktadır. Örneğin bir aracın tekeri için URDF bloğunu robot tanımlamasında dört kez yeniden kullanmak istenirse, bloğu üç kez daha kopyalayıp yapıştırmak gerekir. Oysa bu işlem parametrik olursa 1 kez kod yazıp 4 kez parametrik olarak çağrılabilir. Bu yüzden bir kod bloğunu kullanma ve farklı ayarlarla birden fazla çağırma seçeneği robot tanımlamasını oluşturulurken işi çok kolaylaştırır.

URDF tek bir dosyadır ve içerisine başka URDF dosyaları ekleyemeyiz veya çağıramayız. Bu durum kodun modüler yapısını azaltır. Tüm kodlar tek bir dosyada bulunursa kod basitliğini azaltır ve hata ayıklama sürecini zorlaştırır. Bu yüzden değişken tanımlamalar, sabitler, matematiksel ifadeler, koşullu deyimler ve benzeri eklemeler programlanabilir robot tanımını daha da esnek hale getirmektedir. Xacro aslında makrolar kullanarak URDF'nin biraz daha programlanabilir yapıda olmasını sağlamaktadır. Xacro'nun bazı temel özellikleri şunlardır.

URDF'yi Basitleştirme: Xacro, URDF'nin daha temiz bir sürümüdür. Yaptığı şey, robot tanımlamasında makrolar oluşturur ve makroları yeniden kullanmaya izin verir. Bu durumda kod uzunluğunu azaltır. Ayrıca diğer dosyalardan makro kodlarını okuyabilir, daha basit ve daha modüler hale getirebiliriz.

Programlanabilir Özellik: Xacro dili, açıklamasında basit bir programlama ifadesini destekler. İçerisinde tanımlamayı daha akıllı ve verimli yapan değişkenler, sabitler, matematiksel ifadeler, koşullu ifadeler vardır. Xacro'nun URDF'nin güncellenmiş bir sürümü olduğunu söyleyebiliriz ve bazı ROS araçlarını kullanarak xacro tanımını gerektiğinde URDF'ye dönüştürebiliriz.

XACRO Etiketi

Paket içerisindeki "urdf" klasörü içerisine "robot_kolu_xacro.xml" adında bir dosya oluşturun. https://github.com/brgokce/deneyap/ adresinden "robot_ornegi/urdf/robot_kolu_xacro.xml" dosyasından kodları kopyalayarak yapıştırın veya yazın.

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro"  name="robot_kolu">
    <xacro:property name="base_link_uzunluk" value="0.01" />
    <xacro:property name="base_link_cap" value="0.2" />
    <xacro:property name="govde_x" value="0.08" />
    <xacro:property name="govde_y" value="0.08" />
    <xacro:property name="govde_z" value="0.5" />
    <xacro:property name="kol_uzvu_uzunluk" value="0.4" />
    <xacro:property name="kol_uzvu_cap" value="0.04" />
    <xacro:macro name="atalet_matrisi" params="kutle">
                <inertial>
                        <mass value="${kutle}" />
                        <inertia ixx="0.5" ixy="0.0" ixz="0.0"
                                 iyy="0.5" iyz="0.0"
                                 izz="0.5" />
                </inertial>
    </xacro:macro>

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="${base_link_uzunluk}" radius="${base_link_cap}"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <material name="yellow">
        <color rgba="1 1 0 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
    <cylinder length="${base_link_uzunluk+0.02}" radius="${base_link_cap}"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </collision>
    <xacro:atalet_matrisi kutle="1"/>
  </link>

  <joint name="govde_eklemi" type="revolute">
    <parent link="base_link"/>
    <child link="govde_uzvu"/>
    <origin xyz="0 0 0.1"/>
    <axis xyz="0 0 1" />
    <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/>
    <dynamics damping="50" friction="1"/>
  </joint>

  <link name="govde_uzvu">
    <visual>
      <geometry>
         <box size="${govde_x} ${govde_y} ${govde_z}"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.15"/>
      <material name="red">
        <color rgba="0 0 1 1"/>
      </material>
```

```
    </visual>
    <collision>
      <geometry>
        <box size="${govde_x} ${govde_y} ${govde_z}"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.09"/>
    </collision>
    <xacro:atalet_matrisi kutle="1"/>
  </link>

  <joint name="kol_eklemi" type="revolute">
    <parent link="govde_uzvu"/>
    <child link="kol_uzvu"/>
    <origin xyz="0.04 0 0.36"/>
    <axis xyz="0 1 0" />
    <limit effort="300" velocity="0.1" lower="1.57" upper="-1.57"/>
    <dynamics damping="50" friction="1"/>
  </joint>

  <link name="kol_uzvu">
    <visual>
      <geometry>
        <cylinder length="${kol_uzvu_uzunluk}" radius="${kol_uzvu_cap}"/>
      </geometry>
      <origin rpy="0 1.57 0" xyz="0.15 0 0"/>
      <material name="green">
        <color rgba="1 0 0 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <cylinder length="${kol_uzvu_uzunluk}" radius="${kol_uzvu_cap}+0.01"/>
      </geometry>
      <origin rpy="0 1.57 0" xyz="0.15 0 0"/>
    </collision>
    <xacro:atalet_matrisi kutle="1"/>
  </link>
</robot>
```

Bu satırlar içerisinden en baştaki tanımlama, xacro dosyasını ayrıştırmak için tüm xacro dosyalarında gereken bir ad alanını belirtir. Bu ad alanını belirledikten sonra, xacro dosyasının adını eklememiz gerekir.

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro"  name="robot_kolu">
```

XACRO'da özellikleri (property) kullanma

Xacro içerisinde "`xacro:property`" etiketini kullanarak, isimlendirilmiş değerleri olan sabitleri veya özellikleri tanımlayabiliriz. Başta tanımladığımız bu özellikleri kodun herhangi bir yerinde kullanılabiliriz. Bu sabit tanımların ana kullanımı, uzuvlar ve eklemler üzerinde sabit kodlanmış değerler vermek yerine, sabitleri bu şekilde tutabiliriz. Sabit kodlanmış değerleri bulmak ve değiştirmek yerine bu değerleri kullanmak daha kolay olacaktır. Özelliklerin kullanımına bir örnek burada verilmiştir. Taban uzvu ve gövde uzvunun uzunluğunun temel ölçülerini burada belirtiriz. Bu nedenle, her birindeki değerleri değiştirmek yerine sadece buradaki değerleri değiştirmek kolay olacaktır:

```
    <xacro:property name="base_link_uzunluk" value="0.01" />
    <xacro:property name="base_link_cap" value="0.2" />
    <xacro:property name="govde_x" value="0.08" />
```

Değerleri değişkenlere atama

Sabit kodlanmış bir değeri bir değişkenin değerini ile aşağıdaki tanımlamayı kullanarak değiştirerek kullanabiliriz:

```
<box size="${govde_x} ${govde_y} ${govde_z}"/>
```

Burada kutunun x,y ve z boyutları önceden tanımlanmış özellik isimleri değiştirilir. Böylece kullanım parametrik hale gelir.

Matematiksel ifadeleri kullanma

+, -, *, /, işaretler, "eksi" ve parantez gibi temel işlemleri kullanarak ${ } içinde matematiksel ifadeler oluşturabiliriz. Üstelleştirme ve mod alma henüz desteklenmemektedir. Aşağıdaki kod içinde kullanılan basit bir matematik ifadesidir:

```
<cylinder length="${kol_uzvu_uzunluk}" radius="${kol_uzvu_cap}+0.01"/>
```

Makroları kullanma

Xacro'nun ana özelliklerinden biri makroları desteklemesidir. Xacro kullanarak uzun karmaşık tanımlamaları büyük ölçüde azaltabiliriz. İşte atalet için kodumuzda kullandığımız bir xacro:macro tanımı:

```
    <xacro:macro name="atalet_matrisi" params="kutle">
            <inertial>
                    <mass value="${kutle}" />
                    <inertia ixx="0.5" ixy="0.0" ixz="0.0"
                             iyy="0.5" iyz="0.0"
                             izz="0.5" />
            </inertial>
    </xacro:macro>

<xacro:atalet_matrisi kutle="1"/>
```

Xacro tanımı, kod okunabilirliğini geliştirdi ve URDF ile karşılaştırıldığında satır sayısını azalttı. Ardından xacro'nun urdf dosyasına nasıl dönüştürüleceğini görebiliriz.

Xacro'nun URDF'ye d0önüştürülmesi

Xacro'yu UDRF'ye dönüştürmek için ROS başlatma dosyasında aşağıdaki satırı kullanabiliriz ve bunu robot_description parametresi olarak kullanabiliriz:

```
$ rosrun xacro xacro.py robot_kolu_xacro.xml> robot_kolu_xacro.urdf
```

XACRO dosyasını görüntülemek için "launch" klasörü içerisine "robot_kolu_xacro.launch" adında bir dosya ekleyin ve aşağıdaki kodları ekleyin veya https://github.com/brgokce/deneyap/ klasörü içerisinden "robot_ornegi/launch/robot_kolu_xacro.launch" dosyasını indirin.

```
<?xml version="1.0"?>
<launch>
    <arg name="model" />
    <param name="robot_description" command="$(find xacro)/xacro.py $(find
robot_ornegi)/urdf/robot_kolu_xacro.xml" />
    <node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui"
type="joint_state_publisher_gui" />
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find robot_ornegi)/rviz/robot_kolu.rviz"
required="true" />
</launch>
```

Burada Xacro dosyasını çağıran komut ile
```
command="$(find xacro)/xacro.py $(find robot_ornegi)/urdf/robot_kolu_xacro.xml"
```
URDF dosyasını çağıran komutların farklı olduğunu görüyoruz.
```
textfile="$(find robot_ornegi)/urdf/robot_kolu_urdf.xml" />
```

"robot_kolu_xacro.launch" başlatma dosyasını çağırarak XACRO ile hazırlanmış dosyanın görüntülenmesini sağlayalım.

```
$ roslaunch robot_ornegi robot_kolu_xacro.launch
```

2. TASARLA

2.1. Sabit Kanat Bir İHA'nın Robot Olarak Tanımlanması

Tasarla aşamasında öğrencilerden sabit kanat bir insansız hava aracının robot tanımlama kodlarını kullanarak tasarlamaları istenir. Sabit kanat bir İHA tanımlarken özellikle 'kanatcık'ın senkron bir şekilde çapraz hareket etmesi gerektiğini ve tanımlamalarda buna dikkat edilmesi gerektiği unutulmamalıdır.

Öğrencilerden sabit kanat bir İHA'nın tasarımında kaç uzuvdan ve eklemden oluştuğu, hangi uzuvların sabit, hangi uzuvların hareketli olması gerektiği konuları üzerinde düşünmeleri istenir. Öğrenciler grup olarak tartışırlar. Gerektiği noktada rehber öğretmen onlara yardımcı olabilir. Fakat öğrencilere tam bir çözüm verilmemelidir. Gruplar çözümü kendileri üretmelidir. Tam bir çözümün verilmesi öğrencilerin yaratıcılıklarını olumsuz yönde etkileyebileceğinden tavsiye edilmez. Sabit kanat bir İHA'ı tasarlamak için öğrencilerin aşağıda örnek olarak verilen iki adıma benzer bir süreci gerçekleştirmesi gerekir.

Tanımlama: Öncelikle öğrenciler sabit kanat bir İHA'nın uzuvlarının belirlenmesi gerekmektedir. Daha sonra uzuvların hangileri nasıl hareket etmektedir. Uzuv hareketlerinde sınırlama veya kısıtlama var mıdır? Eklemlerde kısıtlama tanımlamaları nelerdir? Senkron hareket eden uzuvlar var mıdır? Varsa nasıl tanımlanmalıdır? Tüm bu süreçler için öğrenciler gerekli işlemleri maddeler hâlinde yazmalıdır.

Örneğin;

- Bir sabit kanat İHA'nın ana uzvu hangi parçası olmalıdır,
- Kanat yerleşimleri nasıl olmalıdır,
- Eklem ve hareket tanımlamaları nasıl olmalıdır.

Fikir üretme: Bu aşamada öğrencilerin tanımlamada belirlenen işlemlerin nasıl yapılabileceği ile ilgili fikir yürütmesi beklenir. Örnek olarak öğrenciler aşağıdaki maddelere benzer fikirler üretebilir:

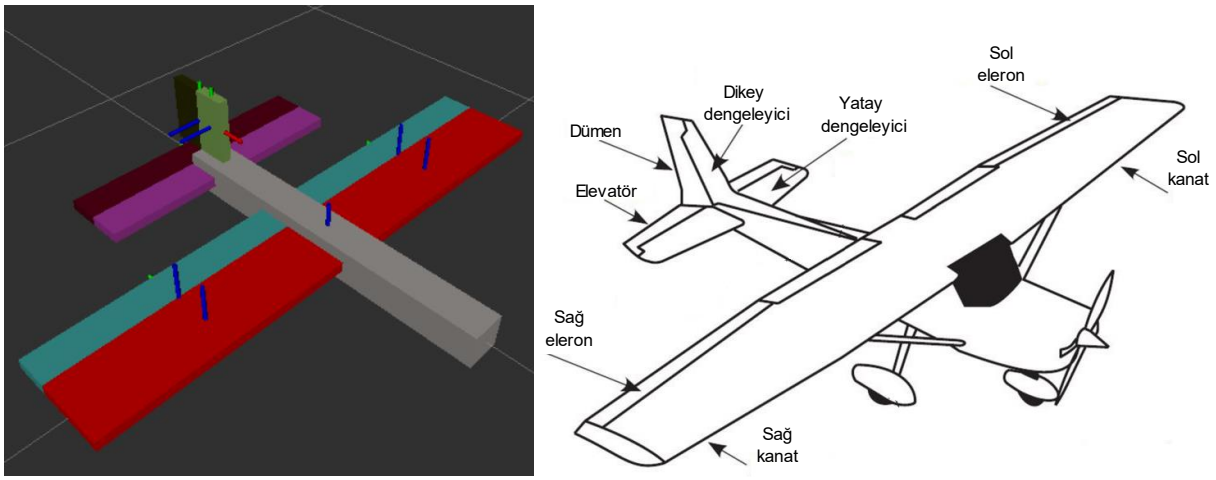
- Sabit kanat bir İHA'da belirlenen uzuvların yerleşme koordinatları ve dönüşüm eksenleri neler olmalıdır? Bu konu öğrenciler arasında tartışılır ve dönüşüm eksenlerinin yerleşme ve hareket üzerindeki etkisi belirlenir.
- Sabit kanat bir İHA'da kanatçık, yatay dengeleyici dümeni ve dikey dengeleyici dümeninin hangi açısal hareketleri yapacağı ve hareket kısırları belirlenmelidir. Belirlenen her bir hareketin

tanımı yapılmalıdır. Bu aşamada öğrenciler hareket tanımlamalarını ve kısıtlarını hızlıca deneyebilirler.

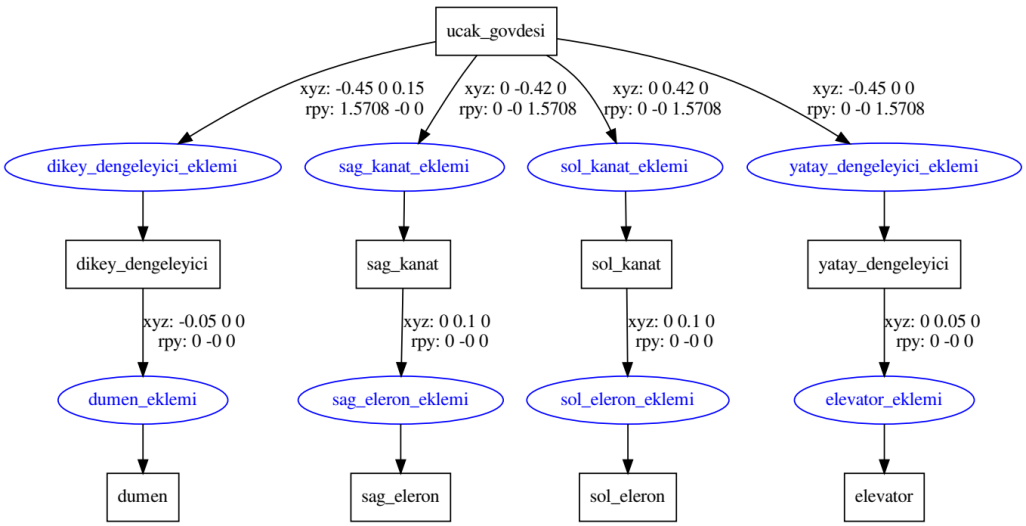
3. ÜRET

3.1. Sabit KANAT İHA Tasarımı

Bu bölümde de öğrenciler bir İHA tasarımında etkin rol üstlenir. Rehber öğretmen yalnızca yönlendirir ve öğrencilere takıldıkları noktalarda destek olur. Yani rehber öğretmen yalnızca ihtiyaç anında destek sağlamalıdır. Üret aşamasında, öğrencilerden bir önceki adımda tartıştıkları temel parametreleri dikkate alarak daha doğru yapılandırılmış bir İHA tasarımı geliştirmeleri istenir. Öğrenciler bilgisayar başında çalışarak gerekli tasarım çözümlerini geliştirirler. Burada öğrencilerin en çok zorlanacağı konular uzuv ve eklemlerdeki eksen dönüşüm kavramlarıdır. Bu durumda birçok kez deneme ile çözebileceklerdir. Şekil 39'da sabit kanat bir uçağın temel sabit ve hareketli uzuvları (https://core.ac.uk/download/pdf/43637015.pdf) ve Şekil 40'da ise sabit kanat bir uçağın uzuv ve eklemler arasındaki hiyerarşisi verilmiştir.



Şekil 39. *Sabit Kanat Bir Uçağın Temel Sabit ve Hareketli Uzuvları (https://core.ac.uk/download/pdf/43637015.pdf)*



Resim 40. *Sabit Kanat Bir Uçağın Uzuv ve Eklemler Arasındaki Hiyerarşisi*

Paket içerisindeki “urdf” klasörü içerisine “ucak_xacro.xml”, “materials.xml” ve “ucak_kanat_eleron.xml” adında bir dosya oluşturun. https://github.com/brgokce/deneyap/ adresinden “robot_ornegi/urdf/ucak_xacro.xml”, “robot_ornegi/urdf/materials.xml” ve “robot_ornegi/urdf/ucak_kanat_eleron.xml” dosyalarından kodları kopyalayarak yapıştırın veya yazın.

“materials.xml” dosyası

```
<?xml version="1.0"?>
<robot>
  <material name="black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>

  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>

  <material name="green">
    <color rgba="0.0 1.0 0.0 1.0"/>
  </material>

  <material name="grey">
    <color rgba="0.2 0.2 0.2 1.0"/>
  </material>

  <material name="orange">
    <color rgba="${255/255} ${108/255} ${10/255} 1.0"/>
  </material>

  <material name="brown">
    <color rgba="${222/255} ${207/255} ${195/255} 1.0"/>
  </material>

  <material name="red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>

  <material name="white">
    <color rgba="1.0 1.0 1.0 1.0"/>
  </material>

  <material name ="yellow">
    <color rgba="1 1 0 1"/>
</material>
</robot>
```

“ucak_kanat_eleron.xml” dosyası

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
<xacro:property name="kanat_uzvu_x" value="0.75" />
<xacro:property name="kanat_uzvu_y" value="0.2" />
<xacro:property name="kanat_uzvu_z" value="0.03" />

<xacro:property name="eleron_uzvu_x" value="0.75" />
<xacro:property name="eleron_uzvu_y" value="0.1" />
<xacro:property name="eleron_uzvu_z" value="0.03" />

<xacro:macro name="kanat_eleron_atalet" params="kutle xx yy zz">
    <inertial>
        <mass value="${kutle}"/>
       <inertia ixx="${xx}" ixy="0.0" ixz="0.0" iyy="${yy}" iyz="0.0" izz="${zz}"/>
    </inertial>
</xacro:macro>

<xacro:macro name="kanat" params="sag_sol otelemeY dondurmeZ">
    <link name="${sag_sol}_kanat">
        <visual>
```

```
            <geometry>
                <box size="${kanat_uzvu_x} ${kanat_uzvu_y} ${kanat_uzvu_z}"/>
            </geometry>
            <origin xyz="0.0 0.0 0" rpy="0.0 0.0 0.0"/>
            <material name ="sari">
                <color rgba="0 0 1 1"/>
            </material>
        </visual>
        <collision>
            <geometry>
                <box size="${kanat_uzvu_x} ${kanat_uzvu_y} ${kanat_uzvu_z}"/>
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
        </collision>
        <xacro:kanat_eleron_atalet kutle="0.2" xx="0.0" yy="0.0" zz="0.0"/>
    </link>

    <joint name="${sag_sol}_kanat_eklemi" type="fixed">
        <origin xyz="0.0 ${otelemeY} 0.0" rpy="0.0 0.0 ${dondurmeZ}"/>
        <parent link="base_link"/>
        <child link="${sag_sol}_kanat"/>
        <calibration rising="0.0"/>
    </joint>
</xacro:macro>

<xacro:macro name="eleron" params="sag_sol otelemeEklemY otelemeKonumY">
    <link name="${sag_sol}_eleron">
        <visual>
            <geometry>
                <box size="${eleron_uzvu_x} ${eleron_uzvu_y} ${eleron_uzvu_z}"/>
            </geometry>
            <origin xyz="0.0 ${otelemeKonumY} 0" rpy="0.0 0.0 0.0"/>
            <material name ="orange">
                <color rgba="${255/255} ${108/255} ${10/255} 1.0"/>
            </material>
        </visual>
        <collision>
            <geometry>
                <box size="${eleron_uzvu_x} ${eleron_uzvu_y} ${eleron_uzvu_z}"/>
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
        </collision>
        <xacro:kanat_eleron_atalet kutle="0.2" xx="0.0" yy="0.0" zz="0.0"/>
    </link>

    <xacro:if value="${sag_sol=='sag'}">
        <joint name="${sag_sol}_eleron_eklemi" type="revolute">
            <origin xyz="0.0 ${otelemeEklemY} 0.0" rpy="0.0 0.0 0"/>
            <parent link="${sag_sol}_kanat"/>
            <child link="${sag_sol}_eleron"/>
            <axis xyz="1.0 0.0 0.0"/>
            <limit lower="-1.14" upper="1.14" effort="17000" velocity="45"/>
            <calibration rising="0.0"/>
        </joint>
    </xacro:if>
    <xacro:if value="${sag_sol=='sol'}">
        <joint name="${sag_sol}_eleron_eklemi" type="revolute">
            <origin xyz="0.0 ${otelemeEklemY} 0.0" rpy="0.0 0.0 0"/>
            <parent link="${sag_sol}_kanat"/>
            <child link="${sag_sol}_eleron"/>
            <axis xyz="1.0 0.0 0.0"/>
            <limit lower="-1.14" upper="1.14" effort="17000" velocity="45"/>
            <calibration rising="0.0"/>
            <mimic joint="sag_eleron_eklemi" multiplier ="-1" offset="0"/>
        </joint>
    </xacro:if>

</xacro:macro>


</robot>
```

“ucak_xacro.xml” dosyası

```
<?xml version="1.0"?>
<robot name="ucak" xmlns:xacro="http://www.ros.org/wiki/xacro">

 <!-- Import Rviz colors -->
 <xacro:include filename="$(find robotik_simulasyon)/urdf/materials.xml" />

<xacro:property name="base_link_x" value="1" />
<xacro:property name="base_link_y" value="0.1" />
<xacro:property name="base_link_z" value="0.1" />

<xacro:property name="yatay_dengeleyici_uzvu_x" value="0.8" />
<xacro:property name="yatay_dengeleyici_uzvu_y" value="0.1" />
<xacro:property name="yatay_dengeleyici_uzvu_z" value="0.03" />

<xacro:property name="elevator_uzvu_x" value="0.8" />
<xacro:property name="elevator_uzvu_y" value="0.1" />
<xacro:property name="elevator_uzvu_z" value="0.03" />

<xacro:property name="dikey_dengeleyici_uzvu_x" value="0.1" />
<xacro:property name="dikey_dengeleyici_uzvu_y" value="0.2" />
<xacro:property name="dikey_dengeleyici_uzvu_z" value="0.03" />

<xacro:property name="dumen_uzvu_x" value="0.1" />
<xacro:property name="dumen_uzvu_y" value="0.2" />
<xacro:property name="dumen_uzvu_z" value="0.03" />

<xacro:property name="atalet_govde_xx" value="${0.83*0.1*0.1}"/>
<xacro:property name="atalet_govde_yy" value="${0.83*1*0.1}" />
<xacro:property name="atalet_govde_zz" value="${0.83*1*0.1}" />
<xacro:property name="atalet_govde_kutle" value="1.46" />

<xacro:macro name="atalet_matrisi" params="kutle xx yy zz">
    <inertial>
        <mass value="${kutle}"/>
       <inertia ixx="${xx}" ixy="0.0" ixz="0.0" iyy="${yy}" iyz="0.0" izz="${zz}"/>
    </inertial>
</xacro:macro>
    <link name="base_link">
        <visual>
            <geometry>
                <box size="${base_link_x} ${base_link_y} ${base_link_z}"/>
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
            <material name="yellow"/>
        </visual>
        <collision>
            <geometry>
                <box size="${base_link_x} ${base_link_y} ${base_link_z}"/>
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
        </collision>
        <xacro:atalet_matrisi kutle="${atalet_govde_kutle}" xx="${atalet_govde_xx}"
yy="${atalet_govde_yy}" zz="${atalet_govde_zz}" />
    </link>
<!-- Bu kisim ucagin kanat ve eleron konfigürasyonu icin eklendi -->
    <xacro:include filename="$(find robotik_simulasyon)/urdf/ucak_kanat_eleron.xml" />

<kanat sag_sol="sag" otelemeY="-0.425" dondurmeZ="1.5707"/>
<kanat sag_sol="sol" otelemeY="0.425" dondurmeZ="1.5707"/>
<eleron sag_sol="sol" otelemeEklemY="0.10" otelemeKonumY="0.05"/>
<eleron sag_sol="sag" otelemeEklemY="0.10" otelemeKonumY="0.05"/>

<!-- YATAY DENGELEYICI VE ELEVATOR TANIMLAMASI-->

<link name="yatay_dengeleyici">
    <visual>
        <geometry>
            <box size="${yatay_dengeleyici_uzvu_x} ${yatay_dengeleyici_uzvu_y}
${yatay_dengeleyici_uzvu_z}"/>
        </geometry>
```

```
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
        <material name="green"/>
    </visual>
    <collision>
        <geometry>
            <box size="${yatay_dengeleyici_uzvu_x} ${yatay_dengeleyici_uzvu_y}
${yatay_dengeleyici_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    </collision>
    <xacro:atalet_matrisi kutle="${0}" xx="${0}" yy="${0}" zz="${0}" />
</link>
<joint name="yatay_dengeleyici_eklemi" type="fixed">
    <origin xyz="-0.45 0.0 0.0" rpy="0.0 0.0 1.5707"/>
    <parent link="base_link"/>
    <child link="yatay_dengeleyici"/>
    <calibration rising="0.0"/>
</joint>

<link name="elevator">
    <visual>
        <geometry>
            <box size="${elevator_uzvu_x} ${elevator_uzvu_y} ${elevator_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.05 0.0" rpy="0.0 0.0 0.0"/>
        <material name="red"/>
    </visual>
    <collision>
        <geometry>
            <box size="${elevator_uzvu_x} ${elevator_uzvu_y} ${elevator_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    </collision>
    <xacro:atalet_matrisi kutle="${0}" xx="${0}" yy="${0}" zz="${0}" />
</link>
<joint name="eklemi_eklemi" type="revolute">
    <origin xyz="0.0 0.05 0.0" rpy="0.0 0.0 0"/>
    <parent link="yatay_dengeleyici"/>
    <child link="elevator"/>
    <axis xyz="1.0 0.0 0.0"/>
    <limit lower="-1.14" upper="1.14" effort="17000" velocity="45"/>
    <calibration rising="0.0"/>
</joint>

<!-- DIKEY DENGELEYICI VE DUMEN TANIMLAMASI-->

<link name="dikey_dengeleyici">
    <visual>
        <geometry>
            <box size="${dikey_dengeleyici_uzvu_x} ${dikey_dengeleyici_uzvu_y}
${dikey_dengeleyici_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
        <material name="black"/>
    </visual>
    <collision>
        <geometry>
            <box size="${dikey_dengeleyici_uzvu_x} ${dikey_dengeleyici_uzvu_y}
${dikey_dengeleyici_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    </collision>
    <xacro:atalet_matrisi kutle="${0}" xx="${0}" yy="${0}" zz="${0}" />
</link>
<joint name="dikey_dengeleyici_eklemi" type="fixed">
    <origin xyz="-0.35 0.0 0.15" rpy="1.5707 0.0 0"/>
    <parent link="base_link"/>
    <child link="dikey_dengeleyici"/>
    <calibration rising="0.0"/>
</joint>

<link name="dumen">
    <visual>
```

```
        <geometry>
            <box size="${dumen_uzvu_x} ${dumen_uzvu_y} ${dumen_uzvu_z}"/>
        </geometry>
        <origin xyz="-0.05 0.0 0.0" rpy="0.0 0.0 0.0"/>
        <material name="white"/>
    </visual>
    <collision>
        <geometry>
            <box size="${dumen_uzvu_x} ${dumen_uzvu_y} ${dumen_uzvu_z}"/>
        </geometry>
        <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    </collision>
    <xacro:atalet_matrisi kutle="${0}" xx="${0}" yy="${0}" zz="${0}" />
</link>
<joint name="dumen_eklemi" type="revolute">
    <origin xyz="-0.05 0.0 0.0" rpy="0.0 0.0 0"/>
    <parent link="dikey_dengeleyici"/>
    <child link="dumen"/>
    <axis xyz="0 1 0"/>
    <limit lower="-1.14" upper="1.14" effort="17000" velocity="45"/>
    <calibration rising="0.0"/>
</joint>
</robot>
```

4. DEĞERLENDİR

Bu aşamada hedef, öğrencilerin öğrenme sürecinde yaşadıkları ve öğrendikleri üzerine düşünmesini sağlamaktır. Bu sayede öğrenciler problem çözme yetenekleri, dersin konusu ve kendileri ile ilgili gözlemler yaparak yeni bilgiler öğrenme, kendilerini değerlendirme ve sonraki çalışmalarını planlama açısından fırsatlar elde edecektir. Öğrencilerden şu soruları cevaplamaları istenebilir:

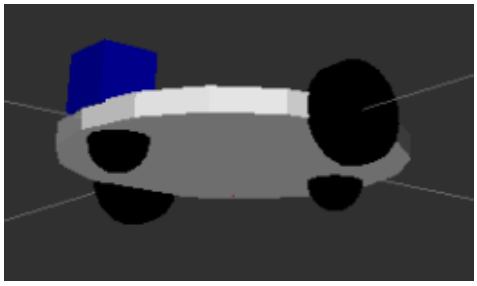
- Bir tasarım problemi nasıl tanımlarsınız? (Problemi kendi cümleleri ile ifade etme)
- Bir tasarımı bileşenlerine nasıl ayırırsınız?
- Tasarımda en çok hangi noktada zorlandınız?
- Grup arkadaşınızla fikir ayrılıkları yaşadınız mı? Bunların üstesinden gelmek için neler yaptınız?
- Grup arkadaşınızdan ne/neler öğrendiniz?

Değerlendirme, öğrencileri sıkmadan, her bir soru için verilen cevaplar tatmin edici bir düzeye ulaşıncaya kadar devam ettirilebilir.

5. İLAVE ETKİNLİK

5.1. İnsansız Bir Mobil Robot Tasarımı

Bu ilave etkinlikteki amaç farklı şekillere ve özelliklere sahip robot tiplerinin tasarımını ve robot tanımlamalarını göstermektir. Bu yeni tasarım yukarıda verilen etkinliğin farklı bir versiyonu şeklinde yapılacaktır. Aşağıda mobil bir robotun temel bileşenleri ve kabaca ölçüleri verilmiştir. Bu bileşenler ve ölçüler bir mobil robot tasarımı için yeterlidir. Ancak tasarım detaylarının düşünülmesi, tartışılması ve hiyerarşik sıralamanın yapılması gereklidir. Şekil 41'de örnek bir tasarım verilmiştir.



Şekil 41. *Örnek Tasarım*

8. HAFTA: ROBOT TASARIMI

Ön Bilgi:

- Öğrenciler sabit kanat veya döner kanat İHA temel bileşenlerini bilir.
- Öğrenciler ROS üzerinde URDF robot tanımlama etiketlerini kullanmıştır.
- Öğrenciler eklem ve uzuv kavramlarını ve ilişkilerini anlamıştır.
- Öğrenciler robot görselleştirme ara yüzü olan RVİZ ortamını kullanmıştır.
- Öğrenciler ROS ortamında bir İHA tasarlamak ve tanımlama için gerekli adımlarını oluşturmuştur.

Haftanın Kazanımları:

- Öğrenciler bir döner kanat İHA tasarımı ve tanımlanması için gerekli temel kavramları bilir.
- Öğrenciler eklem, uzuv, görünüm, çarpışma, atalet, kinematik ve dinamik tanımlamalarını bilir.
- Öğrenciler eklemlerin farklı özelliklerde hareketlendirilmesini bilir ve uygular.
- Öğrenciler uzuvlar ile eklemler arasındaki dönüşüm mantığını anlar ve uygular.
- Öğrenciler bir robotun görselleştirilmesini ve simülasyon yapabilir.

Haftanın Amacı:

Bu haftanın amacı, öğrenciler tarafından döner kanat bir İHA'nın XACRO tanımlamaları ile tasarlanmasını sağlamaktır. Robot tanımlama etiketleri ve yazılım geliştirme ara yüzü ile İHA'nın gövde, kol, motor gibi uzuv ve eklemlerinin RVIZ ortamında görselleştirilmesi sağlanır. Ayrıca, XACRO etiketleri kullanılarak İHA'nın uzuvları ve eklemleri tanımlandıktan sonra atalet tanımlamalarının yapılması ve İHA robotuna hareket verilmesi hedeflenmektedir. Öğrencilere tasarlanan İHA'nın uçuşunun yapabilmesi için sınırlama tanımlamalarının yapılması, İHA'nın uzuvlarının seri ve paralel hareketlerinin sağlanması yine bu haftanın bir diğer amacıdır.

Kullanılacak Malzemeler:

Linux işlerim sistemi üzerine ROS ve GAZEBO ortamları kullanılacaktır.

Haftanın İşlenişi:

Gözle: Döner kanat bir İHA'nın uzuv ve eklemleri üzerine tartışma, uzuvlar ve eklemler arasındaki ilişki ve dönüşüm eksenleri.

Uygula: Robot tanımlama etiketlerini kullanarak bir döner kanat bir İHA'nın uzuv ve eklem tanımlamalarını gerçekleştirme.

Tasarla: Döner kanat bir İHA tasarlamak için hiyerarşik uzuv ve eklem bağlantı oluşturma.

Üret: Bir döner kanat İHA tasarımı yapmak.

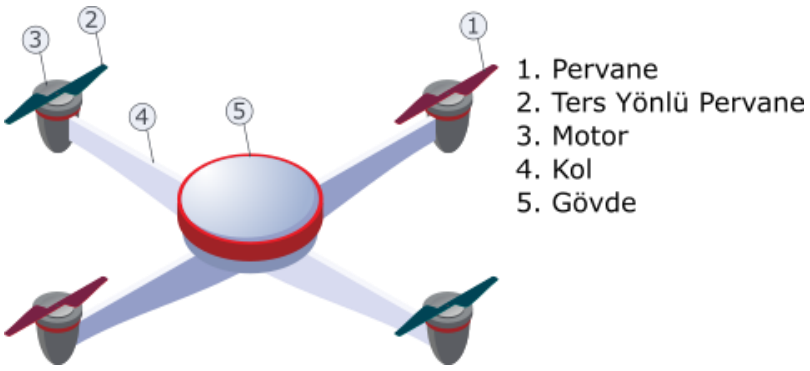
Değerlendir: Haftanın içeriği ile ilgili yansıtma etkinliği ve değerlendirme.

1. GÖZLE VE UYGULA

1.1. Gözle: Döner Kanat Bir İHA'nın Mekaniksel Analizi

Döner kanat İHA olarak adlandırılan dört rotorlu bir diğer ifade ile dört motorlu bir helikopter veya dron olarak da adlandırılan bir quadrotor İHA, dönme ekseni boyunca bir itme kuvveti ve bir moment üreten dört bağımsız rotor/motor tarafından tahrik edilir. Dört rotor tarafından tahrik edilen, ancak üç öteleme serbestlik derecesine (DOF) ve üç dönme/döner serbestlik derecesine sahip olan döner kanat İHA, gerekenden daha az kontrollü serbestlik derecesinin olduğu robotik bir sistemdir.

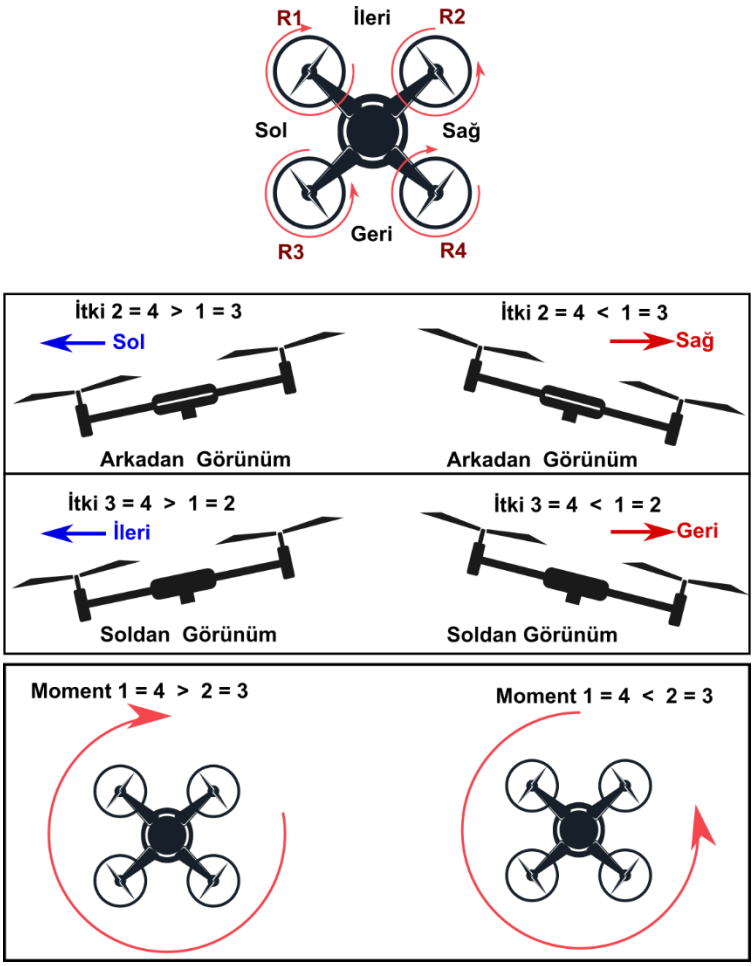
Şekil 42'de bir simülasyon için gerekli olan temel bileşenler verilmiştir. Burada görüldüğü gibi en temelde gövdenin ve kolların bulunduğu ve çerçeve (frame) adı verilen bir mekanik yapı vardır. Motorlar ile kolların bağlandığı bir bağlantı aparatı ve bu bağlantı aparatına bağlanmış motorları görüyoruz. Yine son olarak motorlara bağlı ve iki farklı yönlü pervaneleri görüyoruz. Pervanelerin farklı olması İHA'nın ileri-geri, sağa-sola ve kendi ekseninde bir moment oluşturmasını sağlamaktadır. Şekil 42'de döner kanat iha'nın simülasyon için temel bileşenleri verilmiştir.



Şekil 42. *Döner Kanat İHA'nın Simülasyon İçin Temel Bileşenleri*

Şekil 43'de Döner kanar bir İHA'nın davranış modelleri verilmiştir. Burada görüldüğü gibi iki motorun saat yönünde (R1-R4) ve diğer iki motorunda saat yönünün tersine (R2-R3) döndüğü görülmektedir. İHA'nın arkadan görünümüne bakıldığında İHA'nın 'sol'a hareket etmesi için 2 ve 4 numaralı motorların 1 ve 3 numaralı motorlardan daha fazla itki üretmesi gerektiği görülmektedir. Bunun tam tersi durumda ise yani 'sağ'a gitmesi gerektiği durumda 2 ve 4 numaralı motorların 1 ve 3 numaralı motorlardan daha az itki üretmesi gerektiği görülmektedir. Benzer bir durumu aracın soldan görünümüne bakarak ileri ve geri gitme durumları içinde söyleyebiliriz.

Ancak İHA'nın kendi ekseninde bir sapma açısı (yaw) yani bir moment uygulaması durumunda aynı saat yönünde dönen motorların itki farkları oluşturması gerektiği görülmektedir. Örneğin saat yönünde bir sapma açısı ile moment oluşturması durumunda 1 ve 4 numaralı motorların 2 ve 3 numaralı motorlardan daha fazla itki üretmesi gerektiği görülmektedir. Şekil 43'de döner kanat İHA'nın davranış modelleri görülmektedir.



Şekil 43. *Döner Kanat İHA'nın Davranış Modelleri*

1.2. Döner Kanat Bir İHA'nın Robot Olarak Tanımlanması

Tasarla aşamasında öğrencilerden döner kanat bir insansız hava aracının robot tanımlama kodlarını kullanarak tasarlamaları istenir. Sabit kanat bir İHA tanımlarken özellikle kolların simetrik bir şekilde gövdeye bağlanması gerektiği ve tanımlamalarda buna dikkat edilmesi gerektiği unutulmamalıdır. Öğrencilerden döner kanat bir İHA'nın tasarımında kaç uzuvdan ve eklemden oluştuğu, hangi uzuvların sabit hangi uzuvların hareketli olması gerektiği konuları üzerinde düşünmeleri istenir. Öğrenciler grup olarak tartışırlar. Gerektiği noktada rehber öğretmen onlara yardımcı olabilir. Fakat öğrencilere tam bir çözüm verilmemelidir. Gruplar çözümü kendileri üretmelidir. Tam bir çözümün verilmesi öğrencilerin yaratıcılıklarını olumsuz yönde etkileyebileceğinden tavsiye edilmez. Döner Kanat bir İHA'ı tasarlamak için öğrencilerin aşağıda örnek olarak verilen adıma benzer bir süreci gerçekleştirmesi istenir.

Tanımlama: Öncelikle öğrencilerin döner kanat bir İHA'nın uzuvlarını belirlemeleri istenir. Daha sonra uzuvların hangileri nasıl hareket etmektedir? bunların tanımlamaları istenir. Uzuv hareketlerinde sınırlama veya kısıtlama var mıdır? Eklemlerde kısıtlama tanımlamaları varsa nelerdir yoksa neden kısıtlama konulmamıştır? Senkron hareket eden uzuvlar var mıdır? Varsa nasıl tanımlanmalıdır. Tüm bu süreçler için öğrenciler gerekli işlemleri maddeler hâlinde yazmalıdır.

Örneğin;

- Bir döner kanat İHA'nın ana uzvu hangi parçası olmalıdır?

- Kolların gövdeye yerleşimleri nasıl olmalıdır?
- Eklem ve hareket tanımlamaları nasıl olmalıdır?
- Atalet tanımlamaları nasıl olmalıdır?

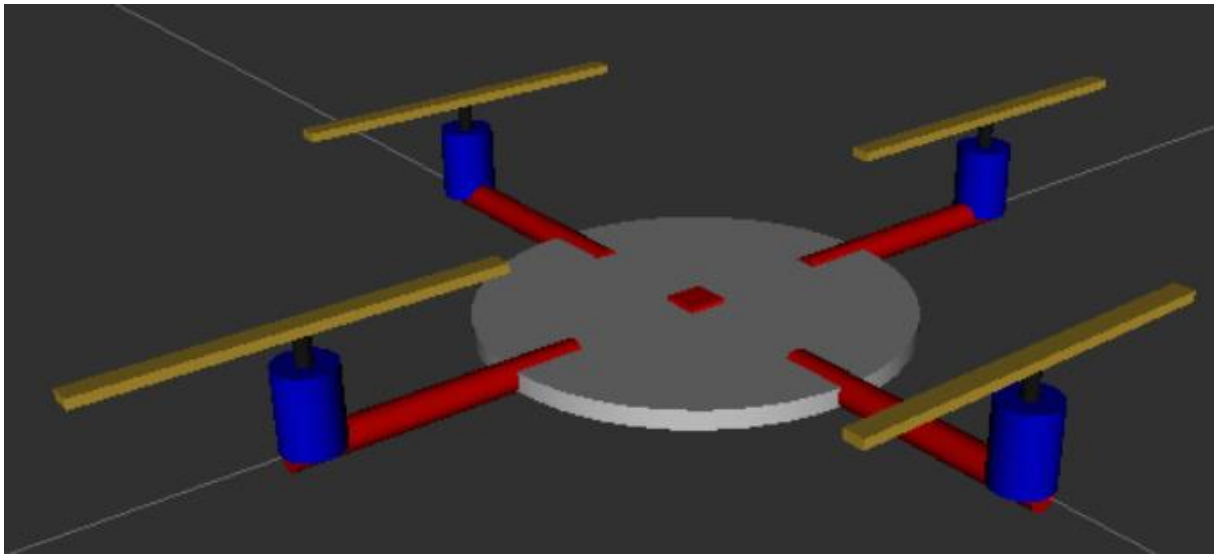
Fikir üretme: Bu aşamada öğrencilerden tanımlamada belirlenen işlemlerin nasıl yapılabileceği ile ilgili fikir yürütmesi beklenir. Örnek olarak öğrenciler aşağıdaki maddelere benzer fikirler üretebilir.

- Döner kanat bir İHA'da belirlenen uzuvların yerleşme koordinatları ve dönüşüm eksenleri neler olmalıdır? Bu konu öğrenciler arasında tartışılır ve böylece dönüşüm eksenlerinin yerleşme ve hareket üzerindeki etkisi belirlenir.
- Döner kanat bir İHA'da motorların kollar üzerine yerleşimi ve pervanelerin de motor mili üzerine yerleşimi ve ters yön hareketleri belirlenmelidir. Belirlenen her bir hareketin tanımı yapılmalıdır. Bu aşamada öğrenciler hareket tanımlamalarını deneyebilirler.

2. TASARLA

2.1. Döner KANAT İHA Tasarımı

Bu bölümde de öğrenciler bir döner kanat İHA tasarımında etkin rol üstlenir. Rehber öğretmen yalnızca yönlendirir ve öğrencilere takıldıkları noktalarda destek olur. Yani rehber öğretmen yalnızca ihtiyaç anında destek sağlamalıdır. Üret aşamasında, öğrencilerden bir önceki adımda tartıştıkları temel parametreleri dikkate alarak daha doğru yapılandırılmış bir döner kanat İHA tasarımı geliştirmeleri istenir. Öğrenciler bilgisayar başında çalışarak gerekli tasarım çözümlerini geliştirirler. Burada öğrencilerin en çok zorlanacağı konular uzuv ve eklemlerdeki eksen dönüşüm kavramlarıdır. Bu durum da birçok kez deneme ile çözülebilir. Şekil 44'de döner kanat bir iha'nın temel sabit ve hareketli uzuvlarının görseli verilmiştir.

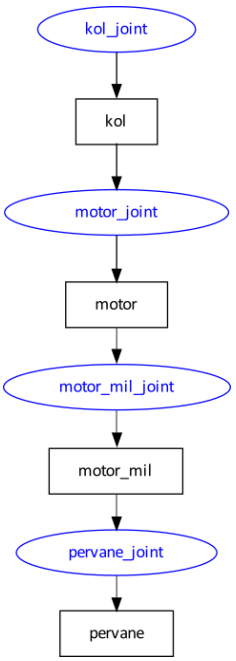


Şekil 44. *Döner Kanat Bir İHA'nın Temel Sabit ve Hareketli Uzuvları*

Robot tasarımına başlamak için ROS ortamında yeni bir paket oluşturulur. Bunun için aşağıdaki kodlar kullanılabilir.

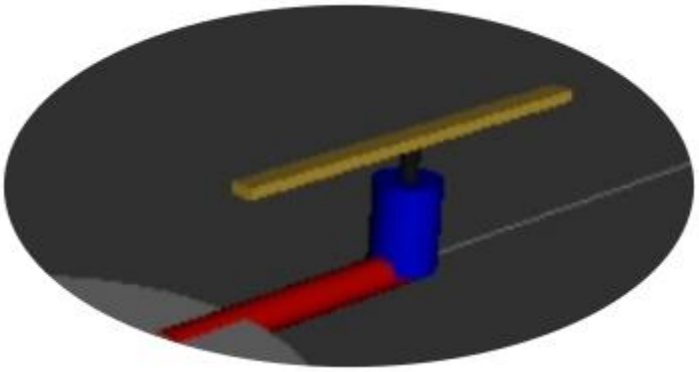
```
$ cd ~/deneyap_ws/src
$ catkin_create_pkg iha std_msgs roscpp rospy actionlib actionlib_msgs
```

ROS paketi oluşturulduktan sonra ilk olarak **deneyap_ws/src/iha/** klasörü altında "urdf" adında boş bir klasör oluşturulur. Bu klasör içerisine üç boş üç adet xml dosyası eklenir. Bunlar "drone.xml", "drone_malzeme.xml" ve drone_kol.xml" dosyalarıdır. Şekil 45'te İHA kolunun uzuv ve eklem ilişkileri verilmiştir. Burada öncelikli olarak silindirik bir kol ve bu kol ile motor arasında bir eklem, sonra bu kol uzvu üzerinde silindirik bir motor, motor ile motor mili arasında bir eklem ve motor milinin silindirik olarak kendisi, motor mili ile pervane arasındaki eklem ve son olarak pervanenin kendisi bulunmaktadır.



Şekil 45. *İHA Kolunun Uzuv ve Eklem İlişkileri*

Aşağıdaki Şekil 46'da bir İHA kolu, motoru, motor mili ve pervanesi verilmiştir. Buna göre tek bir kol ve bu kola ait bileşenler oluşturulmuştur. Daha sonra ana dosya üzerinde makrolar ile çoğaltılması yapılacaktır. Şekil 46'da da görüldüğü gibi İHA kolunun sabit ve hareketli uzuvları ayrı ayrı renklendirilmiştir.



Şekil 46. *İHA Kolunun Sabit ve Hareketli Uzuvları*

Oluşturulan dosya içine aşağıdaki kodlar sırası ile aktarılır. Bunlardan ilki dronun bileşenlerinin renklendirilmesi için **drone_malzeme.xml** dosyası içerisine aşağıdaki kodlar yazılır.

```
<?xml version="1.0" encoding="utf-8"?>
<robot>
    <material name="white">
        <color rgba="0.8 0.8 0.8 1" />
    </material>
    <material name="black">
        <color rgba="0.1 0.1 0.1 1"/>
    </material>
    <material name = "red">
        <color rgba="0.8 0.01 0.01 1"/>
    </material>
    <material name = "blue">
        <color rgba="0.01 0.01 0.8 1"/>
    </material>
    <material name = "green">
        <color rgba="0.01 0.8 0.01 1"/>
    </material>
    <material name = "golden">
        <color rgba="${212/255} ${175/255} ${55/255} 1"/>
    </material>
</robot>
```

Daha sonra dron kol ve bileşenlerinin parametrik olarak oluşturulması için **drone_kol.xml** dosyası içerisine aşağıdaki kodlar yazılır. Burada dikkat edilmesi gereken en önemli şey ölçülerin sabit olmaması ve makrolar ile değişkenlere atanmasıdır.

```
<?xml version="1.0" encoding="utf-8"?>
<robot name="kol" xmlns:xacro="http://ros.org/wiki/xacro">
    <xacro:property name="ixy_olcegi" value ="1.782311429"/>
    <xacro:property name="iz_olcegi" value ="50.140814167"/>
    <xacro:property name="pi" value ="3.14159263"/>
    <xacro:property name="kol_yaricap" value ="0.01"/>
    <xacro:property name="kol_uzunluk" value ="0.15"/>
    <xacro:property name="kol_kutle"   value ="0.1"/>
    <xacro:property name="motor_yaricap" value ="0.015"/>
    <xacro:property name="motor_uzunluk" value ="0.04"/>
    <xacro:property name="motor_kutle"   value ="0.1"/>
    <xacro:property name="motor_mili_yaricap" value ="0.004"/>
    <xacro:property name="motor_mili_uzunluk" value ="0.015"/>
    <xacro:property name="motor_mili_kutle"   value ="0.005"/>
    <xacro:property name="pervane_x" value ="0.02"/>
    <xacro:property name="pervane_y" value ="0.25"/>
    <xacro:property name="pervane_z"   value ="0.002"/>
    <xacro:property name="pervane_kutle"   value ="0.005"/>
    <xacro:macro name="varsayilan_atalet" params="kutle r p y ixx iyy izz">
```

```
        <inertial>
            <origin xyz="0 0 0" rpy="${r} ${p} ${y}" />
            <mass value="${kutle}"/>
            <inertia ixx="${ixx}" ixy="0" ixz="0" iyy="${iyy}" iyz="0" izz="${izz}"/>
        </inertial>
    </xacro:macro>

    <xacro:macro name="silidir_ataleti" params="kutle r h">
        <inertial>
        <origin xyz ="0 0 0" rpy="0 0 0" />
        <mass value="${kutle}"/>
        <inertia ixx="${(kutle/12)*((3*r*r)+(h*h))}"
                 ixy="0"
                 ixz="0"
                 iyy="${(kutle/12)*((3*r*r)+(h*h))}"
                 iyz="0"
                 izz="${(kutle*r*r)/2}" />
        </inertial>
    </xacro:macro>

    <xacro:macro name="kutu_ataleti" params="kutle x y z">
        <inertial>
        <origin xyz ="0 0 0" rpy="0 0 0" />
        <mass value="${kutle}"/>
        <inertia ixx="${kutle*((y*y)+(z*z))/12}"
                 ixy="0"
                 ixz="0"
                 iyy="${kutle*((x*x)+(z*z))/12}"
                 iyz="0"
                 izz="${kutle*((y*y)+(x*x))/12}" />
        </inertial>
    </xacro:macro>

    <xacro:macro name="kol" params="on_arka_sag_sol oteleme_x oteleme_y tersleme_z donmekatsayisi_z
                                    motor_oteleme_x motor_oteleme_y motor_oteleme_z motor_donme_z
                                    motor_mili_oteleme_x motor_mili_oteleme_y motor_mili_oteleme_z
motor_mili_donme_z
                                    pervane_oteleme_x pervane_oteleme_y pervane_oteleme_z
pervane_donme_z">
        <link name="${on_arka_sag_sol}_kol">
        <visual>
            <origin xyz="0 0 0" rpy="${pi/2} 0 ${tersleme_z*pi/4}"/>
            <geometry>
               <cylinder radius="${kol_yaricap}" length="${kol_uzunluk}"/>
            </geometry>
            <material name="red"/>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="${pi/2} 0 ${tersleme_z*pi/4}"/>
            <geometry>
               <cylinder radius="${kol_yaricap}" length="${kol_uzunluk}"/>
            </geometry>
        </collision>
        <xacro:silidir_ataleti kutle="${kol_kutle}" r="${kol_yaricap}" h="${kol_uzunluk}"/>
        </link>

        <joint name="${on_arka_sag_sol}_kol_eklemi" type="fixed">
            <parent link="govde"/>
            <child link="${on_arka_sag_sol}_kol"/>
            <origin xyz="${oteleme_x} ${oteleme_y} 0.0" rpy="0.0 0.0 ${donmekatsayisi_z}"/>
        </joint>

  <!-- Motor uzvu ve eklemi baglantisi -->
        <link name="${on_arka_sag_sol}_motor">
            <visual>
                <origin xyz="0 0 0" rpy="0 0 0"/>
                <geometry>
                   <cylinder radius="${motor_yaricap}" length="${motor_uzunluk}"/>
                </geometry>
                <material name="blue"/>
            </visual>
            <collision>
                <origin xyz="0 0 0" rpy="0 0 0"/>
```

```
                <geometry>
                    <cylinder radius="${motor_yaricap}" length="${motor_uzunluk}"/>
                </geometry>
            </collision>
            <xacro:silidir_ataleti kutle="${motor_kutle}" r="${motor_yaricap}" h="${motor_uzunluk}"/>
            </link>

            <joint name="${on_arka_sag_sol}_motor_eklemi" type="fixed">
                <parent link="${on_arka_sag_sol}_kol"/>
                <child link="${on_arka_sag_sol}_motor"/>
                <origin xyz="${motor_oteleme_x} ${motor_oteleme_y} ${motor_oteleme_z}" rpy="0.0 0.0
${motor_donme_z}"/>
            </joint>

  <!-- Motor mili uzvu ve eklemi baglantisi -->
  <link name="${on_arka_sag_sol}_motor_mili">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
           <cylinder radius="${motor_mili_yaricap}" length="${motor_mili_uzunluk}"/>
        </geometry>
        <material name="golden"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <cylinder radius="${motor_mili_yaricap}" length="${motor_mili_uzunluk}"/>
        </geometry>
    </collision>
    <xacro:silidir_ataleti kutle="${motor_mili_kutle}" r="${motor_mili_yaricap}"
h="${motor_mili_uzunluk}"/>
    </link>

    <joint name="${on_arka_sag_sol}_motor_mili_eklemi" type="fixed">
        <parent link="${on_arka_sag_sol}_motor"/>
        <child link="${on_arka_sag_sol}_motor_mili"/>
        <origin xyz="${motor_mili_oteleme_x} ${motor_mili_oteleme_y} ${motor_mili_oteleme_z}" rpy="0.0
0.0 ${motor_mili_donme_z}"/>
    </joint>


  <!-- Pervane uzvu ve eklemi baglantisi -->
  <link name="${on_arka_sag_sol}_pervane">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
           <box size="${pervane_x} ${pervane_y} ${pervane_z}"/>
        </geometry>
        <material name="white"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${pervane_x} ${pervane_y} ${pervane_z}"/>
        </geometry>
    </collision>
    <xacro:kutu_ataleti kutle="${pervane_kutle}" x="${pervane_x}" y="${pervane_y}" z="${pervane_z}"/>
    </link>

    <joint name="${on_arka_sag_sol}_pervane_eklemi" type="continuous">
        <parent link="${on_arka_sag_sol}_motor_mili"/>
        <child link="${on_arka_sag_sol}_pervane"/>
        <origin xyz="${pervane_oteleme_x} ${pervane_oteleme_y} ${pervane_oteleme_z}" rpy="0.0 0.0
${pervane_donme_z}"/>
        <axis xyz="0.0 0.0 1"/>

    </joint>


    <transmission name="${on_arka_sag_sol}_pervane_eklemi">
        <type>transmission_interface/SimpleTransmission</type>
        <joint name="${on_arka_sag_sol}_pervane_eklemi">
```

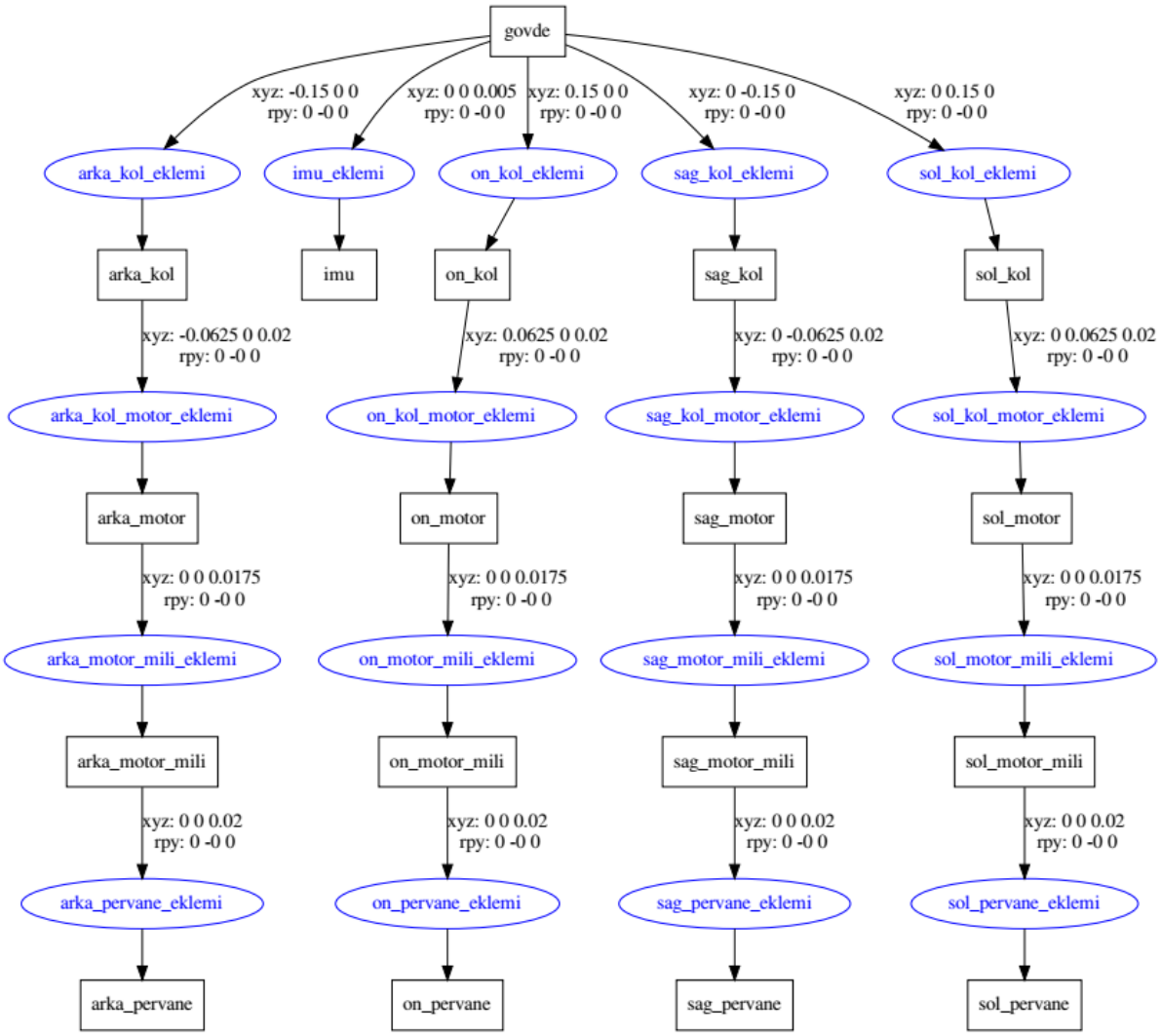
```
            <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
        </joint>
        <actuator name="${on_arka_sag_sol}_pervane_motoru">
            <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
             <mechanicalReduction>1</mechanicalReduction>
        </actuator>
    </transmission>
    <gazebo reference="${on_arka_sag_sol}_motor">
        <material>Gazebo/Blue</material>
    </gazebo>
    <gazebo reference="${on_arka_sag_sol}_motor_mili">
        <material>Gazebo/White</material>
    </gazebo>
    <gazebo reference="${on_arka_sag_sol}_pervane">
        <material>Gazebo/Gold</material>
    </gazebo>
    <gazebo reference="${on_arka_sag_sol}_kol">
        <material>Gazebo/Red</material>
    </gazebo>
    </xacro:macro>
</robot>
```

Malzeme ve dron kol tanımlamaları yapıldıktan sonra dron gövdesi ve kol parçasının makrolar ile çoğaltılması ve birleştirilmesi için **dron.xml** dosyası içerisine aşağıdaki kodlar aktarılır. Burada öncelikli olarak temel değişkenler makro olarak değerleri ile birlikte tanımlanır. Daha sonra gövde tanımlaması yapılır ve üzerine IMU bir uzuv olarak eklenir. Sonra olarak dört adet dron kolu parametrik olarak eklenir.

```
<?xml version="1.0" encoding="utf-8"?>
<robot name="drone" xmlns:xacro="http://ros.org/wiki/xacro">
    <xacro:include filename="$(find iha)/urdf/drone_malzeme.xml"/>
    <xacro:include filename="$(find iha)/urdf/drone_kol.xml"/>
    <xacro:property name="govde_yaricap" value ="0.1"/>
    <xacro:property name="govde_uzunluk" value ="0.02"/>
    <xacro:property name="govde_kutle"   value ="0.3"/>
    <xacro:property name="pi" value ="3.14159263"/>
    <link name="govde">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
               <cylinder radius="${govde_yaricap}" length="${govde_uzunluk}"/>
            </geometry>
            <material name="white"/>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <cylinder radius="${govde_yaricap}" length="${govde_uzunluk}"/>
            </geometry>
        </collision>
        <xacro:silidir_ataleti kutle="${govde_kutle}" r="${govde_yaricap}" h="${govde_uzunluk}"/>
    </link>
    <kol on_arka_sag_sol="on" oteleme_x="0.0" oteleme_y="0.15" tersleme_z="0.0" donmekatsayisi_z="0.0"
    motor_oteleme_x="0.0" motor_oteleme_y="0.06" motor_oteleme_z="0.03" motor_donme_z="0.0"
    motor_mili_oteleme_x="0.0" motor_mili_oteleme_y="0.0" motor_mili_oteleme_z="0.028"
motor_mili_donme_z="0.0"
    pervane_oteleme_x="0.0" pervane_oteleme_y="0.0" pervane_oteleme_z="0.0075" pervane_donme_z="0.0" />

    <kol on_arka_sag_sol="arka" oteleme_x="0.0" oteleme_y="-0.15" tersleme_z="0.0"
donmekatsayisi_z="${pi}"
    motor_oteleme_x="0.0" motor_oteleme_y="0.06" motor_oteleme_z="0.03" motor_donme_z="${pi}"
    motor_mili_oteleme_x="0.0" motor_mili_oteleme_y="0.0" motor_mili_oteleme_z="0.028"
motor_mili_donme_z="0.0"
    pervane_oteleme_x="0.0" pervane_oteleme_y="0.0" pervane_oteleme_z="0.0075" pervane_donme_z="0.0" />

        <!-- Sag ve sol kolun dönüşüm ekseninde X ve Y lerin dogrultulari degistigi icin on ve arkadaki
x yonundaki otelenme
        bu kisimda  y yonunde verilmistir. Dogrulamak icin RVIZ'de TF donusumlerine bakilabilir-->
    <kol on_arka_sag_sol="sag" oteleme_x="0.15" oteleme_y="0.0" tersleme_z="0.0"
donmekatsayisi_z="${pi/2}"
```

```
    motor_oteleme_x="0.0" motor_oteleme_y="-0.06" motor_oteleme_z="0.03"  motor_donme_z="${pi}"
    motor_mili_oteleme_x="0.0" motor_mili_oteleme_y="0.0"
motor_mili_oteleme_z="0.028"  motor_mili_donme_z="0.0"
    pervane_oteleme_x="0.0" pervane_oteleme_y="0.0" pervane_oteleme_z="0.0075" pervane_donme_z="0.0"/>

    <kol on_arka_sag_sol="sol" oteleme_x="-0.15" oteleme_y="0" tersleme_z="0.0"
donmekatsayisi_z="${1.5*pi}"
    motor_oteleme_x="0.00" motor_oteleme_y="-0.06" motor_oteleme_z="0.03" motor_donme_z="${pi}"
    motor_mili_oteleme_x="0.0" motor_mili_oteleme_y="0.0"
motor_mili_oteleme_z="0.028"  motor_mili_donme_z="0.0"
    pervane_oteleme_x="0.0" pervane_oteleme_y="0.0" pervane_oteleme_z="0.0075" pervane_donme_z="0.0"/>

    <link name="imu">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
               <box size="0.02 0.02 0.005"/>
            </geometry>
            <material name="red"/>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <box size="0.02 0.02 0.005"/>
            </geometry>
        </collision>
        <xacro:kutu_ataleti kutle="0.002" x="0.02" y="0.02" z="0.02"/>
        </link>

        <joint name="imu_eklemi" type="fixed">
            <parent link="govde"/>
            <child link="imu"/>
            <origin xyz="0.0 0.0 0.012" rpy="0.0 0.0 0"/>
        </joint>
</robot>
```

Şekil 47'de döner kanat İHA'nın uzuv ve eklemler arasındaki hiyerarşisi verilmiştir. Burada görüldüğü gibi tüm uzuvlar ana uzva eklemler aracılığı ile bağlanmaktadır.



Şekil 47. *Döner Kanat İHA'nın Uzuv ve Eklemler Arasındaki Hiyerarşisi*

9. HAFTA: ROKET SİSTEMLERİ

"Ön Bilgi:

- Uçuşun esasları hakkında temel bilgi

Haftanın Kazanımları:

- Öğrenciler, atmosferin katmalarını ve özelliklerini açıklar.
- Öğrenciler, roketçilik tarihini açıklar.
- Öğrenciler, roketçilik ile ilgili önemli kişileri bilir.
- Öğrenciler, roketlerin kullanım amacını bilir.
- Öğrenciler, roketle ilgili temel tanımları bilir.
- Öğrenciler, roketlerin parçalarını ve kullanım gerekçelerini açıklar.
- Öğrenciler, rokete uçuşları esnasında etki eden kuvvetleri bilir.
- Öğrenciler, uzay mekiğini tanımlar.

Haftanın Amacı:

Öğrencilerin havacılık tarihini kronoljisine göre teknolojik gelişmeler göz önüne alınarak öğrenmeleri amaçlanır. Ülkemizdeki havacılık alanındaki gelişmeler özellikle vurgulanılır. Ayrıca ülkemizdeki havacılık ve uzayla ilgili kurum ve kuruluşlar öğrenilir. Uçuşun esas veprensiplerinin öğrenilmesi hedeflenir. Havacılık tarihindeki ilklerin öğretilmesi amaçlar arasında yer alır.

Kullanılacak Malzemeler:

Powerpoint sunum, bilgisayar, model roket kiti

Haftanın İşlenişi:

Gözle: Atmosfer ve katmaları, Roket nedir ve kullanım yerleri nerelerdir, Roket tarihçesi, Roketin kısımları ve görevleri, Rokete etki eden kuvvetler ve roketin uçuşu, Ağırlık merkezi ve basınç merkezi kavramları, Uzay mekiği ve uzay serüveni

Uygula: Kalemin bir ucuna belirli bir ağırlık yapıştırdıktan sonra kalemin yeni ağırlık merkezinin bulunması ve tartışma

Tasarla: Kâğıt, makas ve yapıştırıcı kullanarak roket tasarlanması

Üret: Bir model roket kiti kullanarak 100 metre yüksekliğe çıkabilecek roket üretimi

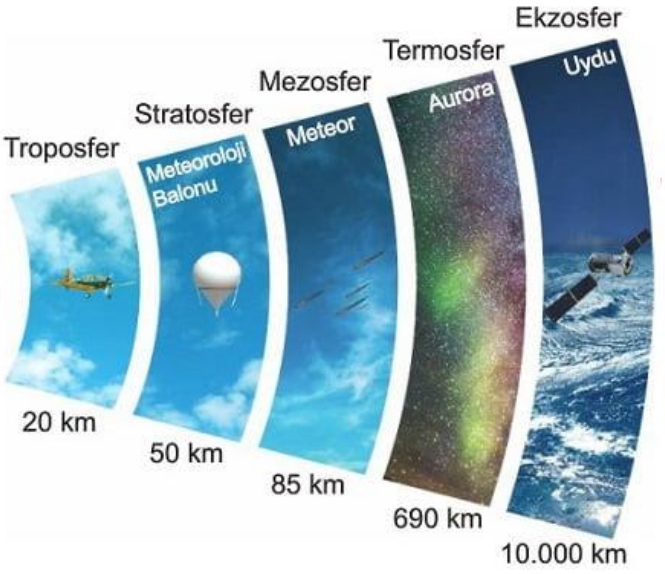
Değerlendir: Haftanın içeriği ile ilgili yansıtma etkinliği

1. GÖZLE VE UYGULA

1.1. Gözle: Atmosfer ve Katmanları

Yeryüzünü saran, gök cisimleri ve zararlı ışınlardan koruyan canlıların yaşamını sürdürmesini sağlayan hava tabakasına atmosfer denir. Yaklaşık %71'i azottan oluşan atmosferin %28'i oksijen ve geri kalanı da su buharı, argon, karbondioksit, neon, helyum, metan, kripton, hidrojen, ozon ve ksenon elementlerinden oluşmaktadır. Atmosferin alt sınırı yeryüzü iken, üst sınırını belirlemek oldukça zordur. Atmosferin kalınlığı yerden itibaren 560 km yüksekliğe kadar ulaştığı ifade edilmekle birlikte atmosferi oluşturan hava kütlesinin %99'u ilk 32 km'lik kısmın içerisinde yer almaktadır. Atmosferi oluşturan gazların temel özelliklerinin değişmediği yerden 80 km'ye kadar olan bölge homosfer, bu bölgenin üzerindeki gazların molekül ağırlıklarına göre ayrıştığı tabaka ise heterosfer olarak adlandırılmaktadır.

Atmosfer özelliklerine göre beş katmana ayrılmaktadır. Bu katmanlar sırasıyla troposfer, stratosfer, mezosfer, termosfer ve egzosferdir.



1.1.1. Troposfer

Atmosferi oluşturan gazların yaklaşık %75'i bu tabaka içerisinde yer almaktadır. Daha çok yeryüzünden yansıyan ışınlarla ısındığı için bu katmanda yerden yükseldikçe sıcaklık her 100 metrede 0.5°C azalmaktadır. Kalınlığı ortalama 12 km olan bu bölgenin bitiminde sıcaklık -56.5°'ye düşmektedir. Atmosfer içerisindeki yatay ve dikey sıcaklık değişimlerinin ve hava olaylarının büyük bölümü bu tabaka içerisinde gerçekleşmektedir. Sıcaklık ortalamasının, yer çekiminin ve çizgisel hızın ekvatorda ve kutuplarda farklılık göstermesi sebebiyle troposferin kalınlığı kutuplarda 8, ekvator üzerinde ise 16 km civarındadır. Kuvvetli hava akımları (rüzgârlar) bu seviyelerde gözlenmektedir. Yapısı tamamen yer radrasyonuna bağlı olarak değişmektedir. Su buharının %99'u troposfer tabakasında yer almaktadır. Su buharı solar enerjiyi ve yerden gelen termal radyasyonu soğurarak sıcaklığın ayarlanmasında önemli bir rol oynamaktadır. Sıcak hava yükselme, soğuk hava çökme eğiliminde olması dolayısıyla

troposferde ciddi hava hareketleri gözlenmektedir ve bu da türbülans anlamına gelmektedir. Bundan dolayı meteorolojistler troposferi mükemmel karışım olarak tanımlamaktadır.

1.1.2. Stratosfer

Atmosferin %19.9'u bu tabaka içerisinde yer alır. Stratosfer, atmosferde 12 km'den 50 km'ye kadar uzanmaktadır. Bu bölge içerisinde iki farklı sıcaklık davranışı görülmektedir. Stratosferin ilk 10 km'sinde sıcaklık -56.5^{o}C'de sabit kalmakta olup troposfer ile stratosfer arasındaki bu geçiş katmanı tropopoz olarak adlandırılmaktadır. Yolcu uçakları, troposferde yoğun olarak görülen türbülans olayından kaçınmak için sıcaklığın değişmediği bu katmanda uçmayı tercih etmektedir. Tropopozun üstündeki katmanda sıcaklık yükseklikle artarak -2.5^{o}C'ye çıkmaktadır. Bu sıcaklık artışı Güneş ışınlarının, yeryüzündeki temel ısı dengesine yardımcı olan ve zararlı ultraviyole ışınlarının yeryüzüne ulaşmasına engel olan ozon tabakası tarafından emilerek bu katmanın ısınması sonucu oluşmaktadır. Bu sıcaklık katmanlaşması nedeniyle, stratosferde çok az konveksiyon ve karışım (az türbülans) vardır. Bu nedenle oradaki hava katmanları oldukça kararlıdır. Troposferden stratosfere geçen partiküller veya stratosfere enjekte edilen kirleticiler uzun süre yeryüzüne dönmeden stratosferde kalabilmektedir. Bu durum küresel iklim değişikliği açısından önem arz etmektedir.

1.1.3. Mezosfer

Atmosferde 50 ila 80-90 km arasında yer almakta olup sıcaklık yükseklikle azalmakta ve -100^{o}C'ye kadar düşmektedir. Mezosferdeki hava basıncı ve yoğunluğu en düşük seviyededir. Mezosfer tabakası yeryüzünü uzaydan gelen meteorlar bu tabakaya girdiklerinde yanmaktadır. Bu seviyede nefes alacak oksijen neredeyse yoktur. Mezosferin en alt seviyesini stratosfer ısıtır, ısı yavaş dönüşümle mezosfere geçmektedir. Stratosfer ile mezosfer arasındaki geçiş katmanı stratopoz, mezosfer ile termosfer arasındaki geçiş katmanı ise mezopoz olarak adlandırılır.

1.1.4. Termosfer

Termosfer 80-90 km'nin üzerinde uzanır. Hava çok incedir. Katmanlar içerisinde sıcaklığın en yüksek olduğu katman olup sıcaklığı yaklaşık 1.000-1.650°C arasında değişmektedir. Kutup ışıkları (aurora) bu katmanda görülmektedir. Gazlar burada iyonlara ayrışmış hâlde bulunur. Bu yüzden haberleşme sinyalleri ve radyo dalgaları çok iyi iletilmektedir. Gaz partikülleri güneşten gelen ultraviyole ve X-ray radrasyonunu bu katmanda absorbe eder. Radyo dalgaları bu seviyeden yeryüzüne dönmektedir.

1.1.5. Egzosfer

Atmosferin en dış katmanıdır. Yer çekiminin en az olduğu katmandır. Haberleşme uyduları bu katmana yerleştirilir. Gazlar çok seyreldiği için katmanın kesin sınırları belirlenemez. Bu nedenle sınır olarak 10.000 km yüksekliğe kadar ulaştığı kabul edilmiştir. Egzosfer 550

km'den binlerce kilometreye kadar uzanır. Bu bölge yeryüzü atmosferi ile gezegenler arası uzayda bir geçiş bölgesi olarak adlandırılmaktadır.

1.2 Gözle: Roket Nedir ve Kullanım Yerleri Nerelerdir?

Oksijen ve yakıtın yanması sonucu oluşarak genleşen sıcak ve basınçlı gazı arkaya püskürtmek suretiyle ileri yönde itki oluşturabilen motorlara ve bu motorların bağlı olduğu, yanma için gerekli yakıt ve oksijeni taşıyan yatay ve dikey uçuş yapabilen hava ve uzay araçlarına roket denmektedir. Belirli bir hedefi olan ve uçuşu esnasında yerçekimi, motor itkisi ve aerodinamik sürüklenmeye maruz kalan roketler, günümüzde sivil veya askeri amaçlı olarak kullanılmaktadır. Uçaklardaki jet motorları havadaki oksijeni kullandığından uzay uçuşlarında kullanılamazlar. Bu nedenle uzay uçuşları ancak roketlerle gerçekleştirilebilmektedir.

Roketler genel olarak uzay çalışmaları, askeri ve hobi amaçlı olarak üretilmektedir. Askeri amaçlı roketler genellikle atmosfer dışına çıkmayan roketlerdir. Uzay çalışmaları için kullanılan roketler ise yörüngeye (atmosfer dışına) araç veya insan göndermek için kullanılmaktadır.

Füzeler belirli hedeflere ulaşabilmek amacıyla roketlerin daha geliştirilmiş güdümlü hâlidir. Füzeler askeri amaçlı olarak hem saldırı hem de savunma amacıyla kullanılabilirler. Füzeler temel olarak birbirlerine benzeseler de uçuş profillerine, taktik kullanımlarına, menzillerine, itki sistemlerine ve güdüm yöntemlerine göre gruplandırılmaktadır. Füzelerin tasarımları daha karmaşık olup, alt bileşenlerinin sayısı roketlere nazaran nispeten daha fazladır.

Sivil bakış açısına göre ise güdümlü olup olmadığına bakılmaksızın her şey rokettir ve füze askeri bir teçhizattır. Bu nedenle hem sivil havacılıkta ve uzaya uydu göndermek için kullanılan araçlar hem de model roketçilikte kullanılan her türlü sistem roket olarak tanımlanmaktadır.

Roketlerin yer çekimine karşı koyabilecek güçtü yükselmesi gerekmektedir. Bir roketin uydu yörüngesine oturabilmesi için 28.000 km/saat, aynı roketin yer çekimi kuvvetinden kurtulması için de yaklaşık olarak 40.000 km/saat hıza ulaşması gerekmektedir. Bu hızlara ulaşan roketlerin yüzeyleri sürtünme dolayısıyla çok yüksek sıcaklıklara maruz kalmaktadır. Bu nedenle roketlerin üretiminde kullanılan malzemeleri hem çok yüksek hem de çok düşük sıcaklıklara dayanabilmesi için yüksek teknoloji gerektiren üretimler ve araştırma-geliştirme çalışmaları yapılmaktadır.

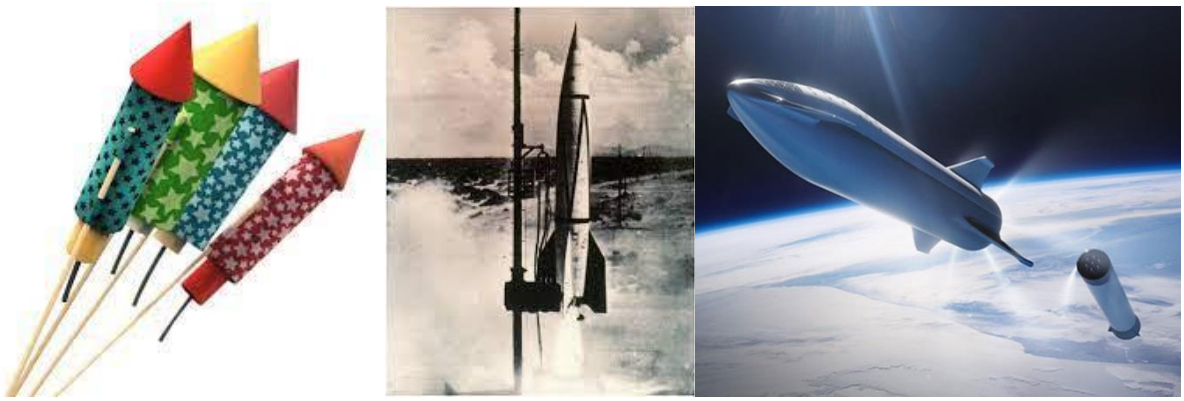
1.3 Gözle: Roketin Tarihçesi

İlk roket yapımı 13. Yüzyıl'da Çin'de gerçekleşmiştir. Yakıt olarak barut kullanılan ve havai fişeklere benzeyen bu roketler, önceleri bayram ve dini törenlerde, sonraları da savaşlarda oklara tutturularak kullanılmıştır. Roketin, Avrupa'ya ulaşması 1241 yılında Moğolların basit rokete benzeyen silahları Macarlara karşı kullanmasıyla olmuştur.

Şekil: *Roketlerin Tarih Boyunca Gelişimi (Solda) ve Çin Roket Tasviri (Sağda)*

Moğolların 1258 yılında Bağdat'ı işgal ederken roket kullanmasıyla roketler Arap literatüründe yerini almıştır. Araplar kendi ürettikleri roketleri 7. Haçlı Seferi'nde Kral IX. Louis'e karşı kullanmıştır. Osmanlılarda IV. Murat zamanında yaşayan Türk mühendis Lagari Hasan Çelebi tarafından ilk roket denemesi yapılmıştır. Lagari Hasan Çelebi, 1633 yılında IV. Murat'ın kızı Kaya Sultan'ın doğduğu gece, Sarayburnu'nda düzenlenen şenliklerde ilk uçuş denemesini gerçekleştirilmiştir. Bu uçuş ilk insanlı roketin icadı ve ilk roketli uçuş denemesi olmuştur. Uzaya roket göndermek için çalışan ABD'li Robert Goddard bu iş için katı yakıt kullanılamayacağını düşündüğünden sıvı yakıtlı ilk roketi 1926 yılında uzaya fırlatmıştır. Alman V2 savaş roketi 1942 yılında Avrupa'da yapılan ve uzaya ulaşan ilk rokettir. 1957 yılında bir Sovyet roketi uzaya ilk uyduyu fırlatmıştır. Yüksekliği 110 metre, kütlesi kalkış esnasında 3.000 ton olan Amerikan Satürn 5 roketi aya ayak basan ilk astronotları taşımıştır.

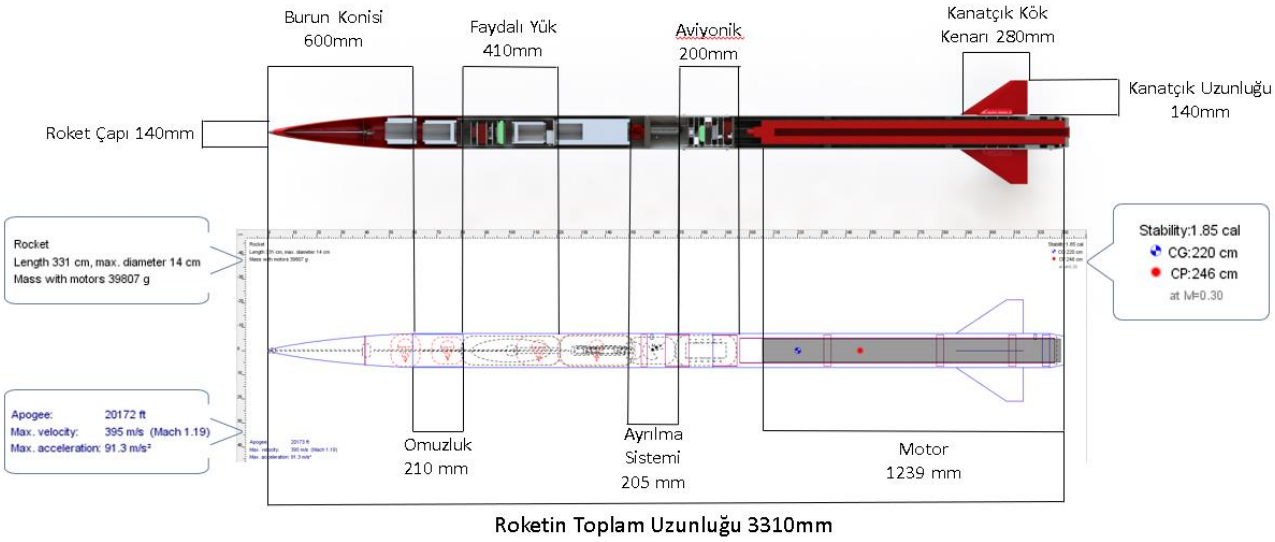
Dünya'da 1950'li yıllarda başlayan model roketçilik çalışmaları ilerleyen yıllarda büyük gelişme göstermiş, model roketler gerçek roketlerin problemlerinin çözümünde kullanıldığı gibi eğitim ve hobi amaçlı olarak üretilerek uçuşlar gerçekleştirilmektedir. Günümüzde iki binin üzerinde aktif grup model roketçiliği ile ilgilenmektedir.



Şekil: *Roketlerin Şekil ve Teknoloji Olarak Geçmişten Günümüze Gelişimi (Soldan Sağa Doğru)*

1.4 Gözle: Roketin Kısımları ve Görevleri

Roket; itme kuvveti ile hareket eden, ön kısmı sivri silindirik bir gövdeye ve bu gövde üzerinde çoğunlukla üçgen şeklinde kanatlara sahip bir hava aracıdır. Roketler gerçekleştirecekleri görevlere bağlı olarak farklı sistemler içerseler de genel olarak bir rokette olması gereken bölümler şunlardır: Gövde, motor, kanatçık, burun konisi, faydalı yük, kurtarma sistemi ve ayırma sistemi.



Şekil: *Türkiye 2021 Teknofest Roket Yarışları Şampiyonu 1,5 Adana WOF Roketi Görünümü (Altta) ve Roket Kesit Görünümü (Üstte)*

1.4.1. Gövde

Roketin ana parçası olan gövde, tüm sistemleri içerisinde barındıran, dış atmosferle olan etkileşimi sağlayan, silindirik yapıda olan ve roketin birçok alt parçalarını içerisinde barındıran kısımdır. Roketin hedeflenen görevi tamamlayabilmesi için dış koşullara dayanacak şekilde tasarlanması önem arz etmektedir. Bu açıdan gövde malzemesi olarak genellikle titanyum, alüminyum, karbon fiber veya cam elyaf gibi hafif ve yüksek dayanımlı malzemeler tercih edilmektedir. Gövdenin kalınlığı gövde malzemesi ile doğrudan orantılı olarak değişmektedir. Gövde genellikle tek parça olmayıp görevi gereği birkaç parçadan oluşmaktadır. Roketin gövdesi tasarlanırken ve diğer parçaların gövde içerisine yerleşimi gerçekleştirilirken roketin ağırlık merkezinin konumuna dikkat edilmelidir. Ayrıca alt parçaların uçuş sırasında ağırlık merkezini değiştirmeyecek ve titreşimden etkilenmeyecek şekilde yerleştirilmesi beklenmektedir. Aksi takdirde uçuşu esnasında roketin kararlılığında oluşacak sorunlar, roketin belirlenen rotada uçuşuna engel olacaktır. Gövdenin geometrik tasarımında en önemli iki parametre gövde çapı ve uzunluğudur. Model roketçilikte fazla sayıda ve karmaşık alt sistemlere ihtiyaç olmaması nedeniyle gövdenin çapının belirlenmesindeki en önemli parametre motor ve montaj yöntemidir. Gövde uzunluğu ise en az tüm alt sistemleri içerisine alacak şekilde seçilmesi beklenmekle birlikte roketin uçuşu esnasında en iyi performansı göstermesi için burun konisi ve kanatçıklarla birlikte optimizasyonu gerekmektedir.

Şekil: *Model roketçilikte kullanılan gövde tipleri*

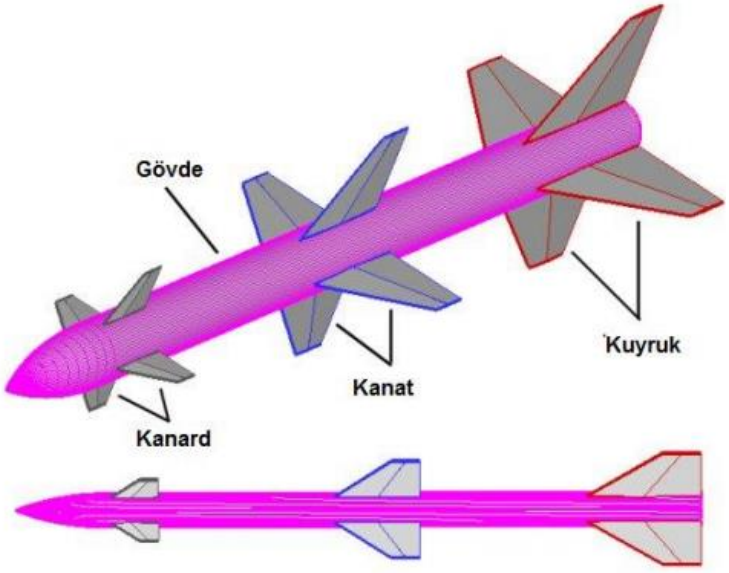
1.4.2. Kanatçıklar

Roketlerin uçuşları esnasında kararlı ve düzgün bir şekilde hareket etmelerini sağlamak amacıyla kanatçıklar kullanılmaktır. Roketlerin hareketleri esnasında uçuşu bozacak bir durum ortaya çıktığında kanatçıklar roketin tekrar kararlı hale getirilmesine yardımcı olurlar. Aksi hâlde kararsızlaşma nedeniyle arzu edilmeyen bir rotanın takip edilmesi söz konusu olabilmektedir. Düzgün tasarlanıp konumlandırılmış sabit kanatçıklar uçuş esnasında aerodinamik kuvvetler oluşturarak kararlı bir uçuşa önemli katkı sağlarlar. Kullanılan kanatçıkların sayıları, boyutları ve geometrileri tasarıma özgü değişiklik göstermektedir. Genellikle üç veya dört kanatçıklı roketler kullanılmaktadır. Kanatçık sayısının belirlenmesinde artan kanatçık sayısı ile birlikte rokete etki eden hava direncinin artıracağı unutulmamalıdır.

Şekil: *1,5 Adana WOF Roketine Ait Kanatçık Görünümleri (Solda ve Ortada), Kanatçıkların Üretilmiş ve Montajlanmış Hâli (Sağda)*

Kanatlar (kanard kontollü, kanat kontrollü ve kuyruk kontrollü) özellikle füze sistemlerinde uçuşu kontrol ve yönlendirme amaçlı olarak da kullanılmaktadır. Kontrol amaçlı kullanılan kanatların üzerinde hareketli yüzeyler bulunmaktadır. Aşağıdaki görsel, her üç kontrol yüzeyini de göstermektedir. Kararlılık amacıyla kullanılan kanatlar sabit kanatçıklar olup üzerlerinde

hareketli kontrol yüzeyi bulunmaması dolayısıyla çok basit bir yapıya sahiptirler. Günümüzde güdümsüz roketlerde gerçekleştirilecek tasarımlarla kararlılık yeterince sağlanabildiği için sabit kanatçıkların kullanımı yeterli olmaktadır. Model roketçilikte kanatçıklar genellikle metal, kompozit, plastik ya da balsa ağacı gibi malzemelerden yapılmaktadır.



Şekil: *Füze Kontrol Yüzeyleri*

1.4.3. Burun Konisi

Burun konisinin amacı roketin üzerindeki aerodinamik yükleri (sürüklenme kuvveti) azaltmak olup rokette havayı ilk karşılayan konik kısımdır. Genellikle konik veya küresel şekillerde geometriye sahiptir. Literatürde çok sayıda burun geometrisi bulunmakta olup bu geometrilerin avantaj ve dezavantajları karşılaştırmalı olarak belirtilmektedir. Ses altı hızlarda küresel bir burun aerodinamik açıdan daha verimliyken, ses üstü hızlarda daha sivri ve konik geometriler tercih edilmektedir. Burun konisinin gövde ile eş merkezli olması gerekmektedir. Burun konisi, üzerine gelen yüklere karşı dayanıklı, roketin ağırlık merkezini olabildiğince az etkilemek ve genel uçuş performansını bozmamak için ise hafif olmalıdır. Burun konisi tasarımındaki önemli parametrelerden biri burun konisinin boyunun taban çapına oranı olarak tanımlanan incelik oranıdır. Roket tasarımında bu oranın roketin tamamı göz önüne alınarak optimize edilmesi gerekmektedir.

Şekil: *Burun Konisi*

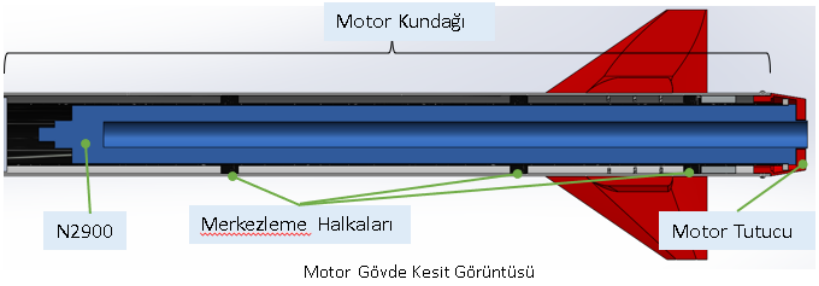
1.4.4. Faydalı Yük

Roketlerin temel amacı, uçuşları esnasında beraberlerinde taşıdıkları faydalı yük olarak adlandırılan teçhizat ve donanımları hedef noktaya güvenli bir şekilde ulaştırmaktır. Faydalı yük belirli bir görevi yerine getirmek için tasarlanan sistemler olup genellikle bilimsel araştırma araçları veya deney cihazlarıdır. Roketler irtifa görevlerinde tercih edildikleri için bu araştırmalar çoğunlukla atmosfer ya da uzay ile ilgilidir. Roketler seyirleri esnasında taşıdıkları faydalı yüke dış ortam şartlarından zarar gelmeyecek şekilde tasarlanırlar. Faydalı yük rokette genellikle uç kısma yerleştirilmekle birlikte uzay mekiği gibi roketin üzerine yerleştirilen faydalı yükler de vardır.

Şekil: *Faydalı yük örneği*

1.4.5. Motor

Motorlar, rokete yerçekimi ve aerodinamik sürüklenmeyi yenecek şekilde itki vererek yukarı yönde hareketini sağlayan sistemlerdir. Motorlar roketleri yüksek ivme ile hareketlendirerek çok kısa zaman içerisinde çok yüksek hızlara ulaştırabilirler. Roketlerin en alt kısmında yer alan motorlar gücünü gazlardan almaktadır. Basit bir anlatımla kapalı bir ortamda yakıtın oksitlenmesi ile yanma odasındaki reaksiyon sonucu oluşan yüksek sıcaklık ve yüksek basınca sahip gazların yanma odasının ucunda yer alan lüle yardımıyla yüksek hızda tahliyesi neticesinde bir kuvvet oluşur. Oluşan kuvvet gazın çıkış hızı ve debisiyle doğru orantılıdır. Bu kuvvet etkiye tepki prensibi ile roketin ileri doğru hareket etmesine neden olur. Roket motorları hareketli parça içermezler. Ancak çok yüksek ivme, yüksek sıcaklık ve yüksek basınçta çalıştıkları için teknolojik açıdan gelişmiş sistemlerdir.



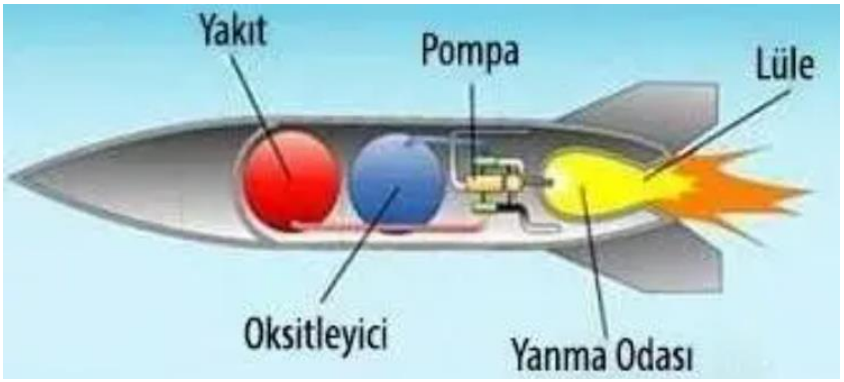
Şekil: *Motor Gövde Kesit Görüntüsü*

Roket motorları yakıtı ve oksitleyiciyi roketin içerisinde bulundurur. Bu sayede motorların dışarıdan hava almaya ihtiyaçları kalmadığından roketler havanın ince olduğu ya da hiç olmadığı yerlerde (atmosferin üst katmanlarında veya uzayda) kullanılabilirler. Yanma sonucu oluşan gaz sıcaklığı ne kadar yüksekse gaz roketten o oranda hızlı çıkar ve roket de hızlı itilir.

Roketlerde kullanılan yakıtlar genellikle sıvı yakıtlar ve katı yakıtlar olmak üzere ikiye ayrılır. Bunların yanı sıra her iki yakıt sisteminin kullanıldığı hibrit sistemler de kullanılmaktadır. Katı yakıtlı motorlar barutun bulunmasıyla birlikte kullanılmaya başlanan en eski motorlardır. Katı yakıtlı roketler nispeten daha basit yapıya sahip roketlerdir. Yakıtın yanmasını sağlayan ve yanmayı kontrol eden bir motor düzeneği bulunmamaktadır. Belirtilen bu iki durumdan dolayı sıvı yakıtlı roketlere nazaran hacim ve büyüklük açısından daha küçük olabilirler.

Katı yakıtlı roketlerde yanıcı ve yakıcı maddeler homojen olarak karıştırılmaları gerekmektedir. Zira, karışımın homojen olmaması roketin yanma odasında ve eksoz çıkışında yerel patlamalara neden olmaktadır. Bu da roketin hızında düzensizliklere sebebiyet vermektedir. Katı yakıtlı roketlerin en önemli avantajı sıvı yakıtlı roketlere nazaran daha yüksek itme gücüne sahip olmalarıdır. Lüleden çıkan gazın sıcaklığı çok yüksek olduğundan, yakıt tankı ve lülenin üretildiği malzemenin yüksek sıcaklık dayanımına sahip olması gerekmektedir. Katı yakıtlı roketlerde genellikle oksitleyici madde olarak; potasyum nitrat, sodyum nitrat, amonyum nitrat, potasyum perklorat, amonyum perklorat, yakıt olarak; sakkaroz, glikoz, sorbitol, alüminyum ve bağlayıcı madde olarak; dekstrin, parafin, HTPB (Hydroxyl-terminated Polybutadiene), epoksi, çeşitli polimer türevli reçine kullanılmaktadır.

Sıvı yakıtlı roketlerde sıvılaştırılmış yakıcı ve yanıcılar depolanmış oldukları tanktan pompa yardımıyla kontrollü olarak yanma odasına taşınmaktadır. Yakıcı olarak sıvı oksijen, nitrik asit, nitrojen peroksit, flor, oksijen biflorit kullanılırken yanıcı olarak sıvı hidrojen, kerosen, hidrozin, propan, metan, hipergolik yakıtlar (monometil hidrozin, antisimetrik dimetil hidrozin) kullanılmaktadır. Savaş füzeleri, uzay mekiklerinin çoğu, uzun menzilli balistik füzeler ve yüksek irtifa araştırma roketlerinde sıvı yakıtlı motor kullanılmaktadır. Ayarlanabilir itki seviyesi, yanma odasında kontrol edilebilir yanma hızı ve sıvı formdaki iticilerin yüksek yoğunluğu sebebiyle tank hacimlerinin küçük olması sıvı yakıtlı motorların avantajları arasındadır. Yüksek bakım maliyeti ve karmaşık akış kontrol sistemi ve yanma odasının soğutulması için gerekli sistemler sıvı yakıtlı motorların dezavantajı olarak karşımıza çıkmaktadır.



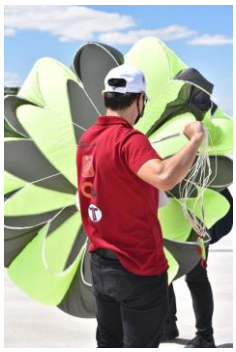
Şekil: *Sıvı Yakıtlı Roket*

Hibrit roket motorunda, yanıcı ve yakıcılardan (oksitleyici) biri sıvı veya gaz diğeri de katı haldedir. Genellikle yakıt katı halde, oksitleyici ise daha güvenli ve verimli olması dolayısıyla sıvı veya gaz halde olmaktadır. Yakıt olarak parafin, kauçuk veya alüminyum tozuyla takviye edilmiş polimerler kullanılmaktadır. Oksitleyici olarak ise sıvılaştırılmış hidrojen veya hidrojen peroksit tercih edilmektedir. İtki gerektiğinde bir kontrol edilebilen bir vana yardımıyla sıvı ya da gaz yakıcı yanma odasına sevk edilmekte ve katı yanıcı ile etkileşime girmektedir. Böylelikle yanmanın şiddetinin ayarlanması mümkün olmaktadır. Hibrit yakıtlı roket motorları karmaşık olmamaları dolayısıyla daha güvenli olması ve sıvı yakıtlı motorlara nazaran daha az hacim kaplamaları bakımından avantajlara sahiptir. En önemli dezavantajı ise sıvı yakıtlı roketlere nazaran verimlerinin az almasıdır. Hibrit roket motorları mikro veya küçük ölçekli uyduların yörüngeye çıkartılması için tercih edilirler.

1.4.6. Kurtarma

Çok sayıda uçurulması hedeflenen ve bu doğrultuda tasarlanan roketleri güvenli bir şekilde kurtarılmak amacıyla paraşüt kullanılmaktadır. Paraşütler ayrıca roketin inişi (düşüşü) esnasında kontrolsüz bir şekilde çevreye zarar vermelerini de engellemektedir. Roketin içerisinde burun konisine şok kordonu vasıtasıyla bağlı olan paraşüt genellikle motorun fırlatma barutu tarafından burun konisiyle birlikte dışarıya doğru itilmektedir. Atmosferde açılan paraşüte uygulanan hava direnci sayesinde roketin tahribatsız bir şekilde inişi sağlanmaktadır. Paraşütün doğru zamanda açılması kurtarmanın en önemli hususudur. Zira erken açılan paraşüt roketin kararlılığını veya uçuşunu olumsuz yönde etkilerken geç (düşüşe geçtikten sonra) açılan paraşütler yapıya zarar verebildiği gibi paraşütün kendisi de zarar görebilmektedir. Paraşütlerde dikkat edilmesi gereken önemli konulardan biri paraşütün serbest düşüş hızını belirleyen paraşüt yüzey alanıdır. Paraşütün yüzey alanının düşük olması serbest düşüş hızını arttıracaktır. Yüzey alanının yüksek olması ise roketi ağırlaştıracağı gibi roketin içerisine yerleştirilen paraşütün hacmini de arttıracaktır. Diğer bir önemli husus ise paraşüt ipleridir (kolonlar). İplerin özellikle ilk açılışta paraşüt bağlantı noktalarından veya rokete bağlandığı noktadan kopmayacak şekilde seçilmesi gerekmektedir.

Şekil: *Örnek Kurtarma Planlaması ve Örnek Kurtarma Sistemi*



Paraşütler, kurtarılması hedeflenen parçaların yere inişini istenen aralıktaki hızda düşürmeyi sağlayan kurtarma elemanlarıdır. Çeşitli hesaplamalar ile çap, boy, sürtünme katsayısı gibi parametreler göz önünde bulundurularak paraşüt hesapları yapılır. Paraşütlerin açıldığı andaki şoka dayanması başarılı kurtarma için çok önemlidir.

Şok kodu

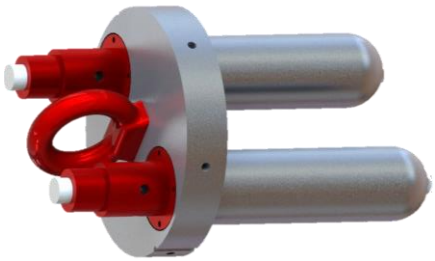
Şok kordonları, kurtarılacak parçaların birbirlerine ve bu parçaların paraşüt bağlantı noktalarına bağlanmasını sağlar. Ayrıca açılma anında oluşacak şok yüküne dayanıklı olmaları gerekmektedir.

Şekil: *Örnek bir paraşüt ve şok kordu*

1.4.7. Ayırma Sistemi

Roket istenilen irtifaya ulaştığında gerek paraşütün ve gerekse faydalı yükün roketten dışarı atılması için ayırma sistemleri kullanılmaktadır. Ayırma sistemleri zamanlayıcı veya sensörlerle aktif hale getirilebilir. Zamanlayıcı sistemler genellikle motorun yanması bittikten sonra alevin geciktirilmesi ile aktive edilir. Yakıtın sonunda bulunan ve yavaş bir yanma hızına sahip olan başka bir yakıt, motorun itki profili bitince ateşlenir. Yavaş yanan bu yakıt başka bir yakıta bağlıdır ve bu yakıt geciktirici bittikten sonra devreye girer, ortamı basınçlandırır. Balistik hesaplamalar yapılarak geciktirme 30 süresi hesaplanabilir. Zamanlayıcı ve soğuk sistemlerde ise yavaş yanan yakıt bitince basınçlı bir tüpü aktive eder ve ortam soğuk bir şekilde basınçlanır. Sensörlü sistemler ise kullanılmak istenilen veriye göre -basınç, ivme, hız,

irtifa vs. uygun sensörler kullanılarak çalışır. Uçuş istenilen profile geldiğinde sensörler ayırma sistemini aktive ederler.



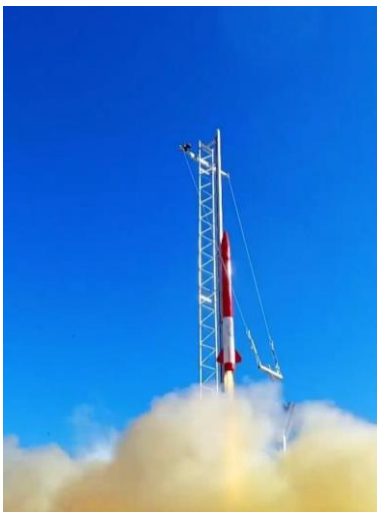
Şekil: *Raptor Peregrine*

Şekil: *Mekanik Açılma Mekanizması*

Şekil: *Barutla Açılma (Barut)*

1.4.8. Fırlatma Rampası

Roketin atışının sağlandığı, çevresele koşullara ve hedef koşullarına göre yerden belirli bir açı ile eğim verilen atış platformuna fırlatma rampası denmektedir.



Şekil: *Atış Rampasına Yüklenmiş 2021 1,5 Adana WOF Roketi (Solda) ve Ateşlenmiş Çukurova Rocketry Roketi (Sağda)*

1.5. Gözle: Rokete Etki Eden Kuvvetler ve Roketin Uçuşu

Roketlere etki eden kuvvetler uçaklara etki eden kuvvetlerle hemen hemen aynıdır. Bu kuvvetler; sürüklenme, taşıma, itki ve roketin ağırlığıdır. Ancak bu kuvvetlerin motorlu bir uçak veya planörün aksine bir model roket üzerindeki etkilerinde önemli farklılıklar bulunmaktadır. Bir uçakta, aerodinamik kuvvetler genelde kanatlar ve kuyruk yüzeyleri tarafından üretilmekteyken model roket için, aerodinamik kuvvetler (sürüklenme ve taşıma) hem gövde hem de kanatlar tarafından üretilmektedir. Taşıma kuvveti uçağın ağırlığını yenmek için kullanılmaktayken model rokette, ağırlığa karşı itme gücü kullanılır ve uçuş yönünü stabilize etmek ve kontrol etmek için taşıma kuvveti kullanılır. Çoğu uçağın taşıma/sürüklenme katsayısı oranı yüksek olmasına rağmen roketin sürüklenmesi genellikle taşımadan çok daha yüksektir. Etki eden kuvvetlerin büyüklüğü ve yönü bir uçak için neredeyse sabit kalırken, bir

rokete etki eden kuvvetlerin büyüklüğü ve yönü tipik bir uçuş sırasında önemli ölçüde değişikliğe uğramaktadır. Bir roket uçuşu esnasında ağırlık merkezi etrafında dönerek ilerler. Rokette daha az bileşen bulunması nedeniyle ağırlık merkezinin belirlenmesi uçağa nazaran daha basittir.

Newton'un birinci yasası, bir dış kuvvetin etkisiyle durumunu değiştirmek zorunda kalmadıkça, her nesnenin düz bir çizgide durağan veya düzgün hareket halinde kalacağını ifade etmektedir. Başka bir ifadeyle bir cisme etki eden net bir kuvvet sıfırsa, sabit hızla hareket eden cisim hızını koruyacak, sıfır hızda ise hareketsiz kalmaya devam edecektir. Newton'un ikinci yasasına göre ise bir cisme etki eden net kuvvet sıfırdan farklıysa bu net kuvvet cismin kütlesi ve ivmesinin çarpımına eşittir. Başka birifadeyle, sıfırdan farklı olan net kuvvet cismin hızlanarak hareketine devam etmesine neden olmaktadır.

Fırlatma rampasından kalkan bir roketin hareketi dikkate alındığında motorun ateşlenmesinden hemen önce roketin hızı sıfırdır ve roket hareketsizdir. Motor ateşlendiğinde, motorun sağladığı itki kuvveti ağırlığa karşı ek bir kuvvet oluşturur. İtki ağırlıktan daha az olduğu sürece, kanatlar boyunca itki ve tepki kuvvetinin kombinasyonu ağırlığı dengeler, net bir dış kuvvet yoktur ve roket rampa üzerinde kalmaya devam eder. İtki kuvveti ağırlığa eşit olduğunda, roket üzerindeki net kuvvet hâlâ sıfırdır. Ancak, itki kuvveti roketin ağırlığından büyük olduğunda, itme kuvveti ile ağırlığın farkı kadar net bir dış kuvvet vardır ve roket hareket ederek yükselmeye başlar. Roket net dış kuvvet tarafından üretilen ivme dolayısıyla sıfır hızdan başlayarak çok yüksek hızlara sahip olacak şekilde yükselişini sürdürür. Roketin hızı arttıkça, harekete zıt yönde oluşan ve roket hızın karesi ile doğru orantılı olarak artan aerodinamik sürüklenme kuvveti oluşmaktadır. Uçuş esnasında roket ekseniyle uçuş rotası ile aynı hizada olduğunda roketin simetrisinden dolayı herhangi bir taşıma kuvveti oluşmayacaktır. Ancak roket ekseni ile uçuş rotası arasında sapma oluşması durumunda üretilecek taşıma kuvveti roketi tekrar hizaya getirmeye çalışacaktır. Roketin ivmelenmeye devam etmesi için artık roketin itki kuvvetinin ağırlık ve artan sürüklenme kuvvetinin toplamından daha büyük olması gerekmektedir. İtki kuvveti ağırlık ve sürüklenme kuvvetinin toplamına eşit olursa, roket artık sabit bir hızla tırmanmaya devam edecektir. İtki kuvveti artık bu iki kuvveti dengeleyemeyecek duruma geldiğinde yani yakıt tükendiğinde, rokete etki eden net kuvvet hareket yönünün tersi yönde olduğunda ise roket yavaşlayacaktır.

Sonunda roketin dikey hızının sıfır olduğu ve roketin hareketi boyunca çıkabileceği en yüksek irtifaya çıkılacaktır. Bu irtifada rokete etki eden tek kuvvet yeryüzüne doğru olan (sürüklenme kuvveti ve itki kuvveti olmadığından) roket ağırlığıdır. Dolayısıyla roketin yukarı yönlü hareketi sona erecek ve artık aşağı yönlü hareketine hızlanarak başlayacaktır. Aşağı yönlü harekette bu sefer rokete etki eden sürüklenme kuvveti ile ağırlık birbirlerine zıt yönlü olarak oluşacaktır. Roket ağırlığı düşüş esnasında sabit kalırken sürüklenme kuvveti roketin hızına bağlı olarak artacaktır. Sürüklenme kuvveti ile ağırlığın birbirine eşit olduğu duruma gelindikten sonra rokete etki eden net kuvvet sıfır olacağından artık roket sabit hızla düşüşüne devam edecektir. Roketin düşüşü esnasında her iki kuvvetin de birbirine eşit olduğu durumdaki hız serbest düşme hızı (terminal hız) olarak adlandırılmaktadır. Ancak roketin düşüşü esnasında serbest düşme hızına ulaşmasını beklemeden mümkünse roketin uçuşu esnasında ulaştığı en yüksek

konumda gövde içerisindeki paraşütün açılarak kurtarma işleminin başlatılması gerekmektedir.

Üretim maliyetleri düşünüldüğünde roketlerin bir defalık kullanılmaması gerektiği anlaşılmaktadır. Roketi görevi sonunda kurtarmak ve yeniden kullanabilmek için roket en yüksek irtifaya ulaştığında veya hemen akabinde paraşütün açılması gerekmektedir. Kullanılacak paraşütün yüzey alanı hesabındaki önemli parametrelerden birisi de yukarıda tanımı verilen serbest düşme hızıdır. Uygun bir paraşüt yüzey alanı seçimi neticesinde roketin yere çarpması (serbest düşme hızında) anında zarar görmemesi ve yeniden kullanılması sağlanır.

1.6. Gözle: Ağırlık Merkezi ve Basınç Merkezi

Kütle merkezi veya ağırlık merkezi roket üzerine etki eden ve yukarıda ifade ettiğimiz kuvvetlerin etki ettiği düşünülen noktadır. Roketi yatayda asılı kalacak şekilde durmasını sağlayacak bir ipin bağlandığı nokta yaklaşık olarak kütle merkezinin yerini ifade eder. Roketin her bir bileşeninin ağırlığı ve konumunun bilinmesinden ziyade roketin ağırlık merkezi ve toplam ağırlığının bilinmesi birçok hesaplama için yeterli olmaktadır. Serbest bir cisim dönme hareketini ağırlık merkezi etrafında gerçekleştirir. Bu dönme hareketini oluşturacak aerodinamik kuvvetler ise roketin basınç merkezine etki eder.

1.7. Uygula: Ağırlık Merkezinin Tespiti

Elinize bir kalem alın. Kalemin ağırlık merkezini bulmak için bir yöntem geliştirin. Daha sonra kalemin herhangi bir ucuna bozuk para bantlayın. Yeni durum için ağırlık merkezinin yerini tespit edip ağırlık merkezinin konumunun değişimini tartışın.

1.8. Gözle: Uzay Mekikleri

1.8.1. Uzay Mekiği Nedir?

Uzay mekikleri, astronot ve uyduları taşımak, içerisinde bulunan laboratuvarlarda yer çekimsiz ortamda deney yapmak ve uzaktan erişimi sayesinde bünyesindeki faydalı yükü istediği irtifada bırakmak gibi görevleri yerine getirmek üzere kullanılmış araçlardır. Amerika Birleşik Devletleri Ulusal Havacılık ve Uzay Dairesi (NASA) uzay mekikleri ile ilgili çalışmalara 1968 yılında başlamıştır. İlk mekik olan Columbia Kennedy Uzay Merkezi'nde 12 Nisan 1981 tarihinde fırlatılarak iki kişilik astronot ekibiyle uzay serüvenine başlamıştır. Uzay mekikleri, üzerlerinde bulunan büyük hacimli yakıt tanklarıyla roket gibi kalkarken uzaydaki görevlerini tamamladıktan sonra uçak gibi yeryüzüne inerler.

Uzay mekikleri, Dünya'ya geri dönerken kütle çekim kuvveti nedeni ile atmosferde çok yüksek hızlara ulaşıştır. Bu hız (yaklaşık olarak ses hızının 20 katı) nedeniyle oluşan sürtünme nedeniyle mekik yüzeyinde sıcaklık 2000^{o}C'ye kadar çıkabilir. Bunun yanı sıra mekik yüzeyinin

uzayda yaklaşık -120ºC sıcaklığa dayanması gerekmiştir. Bu sıcaklıklara dayanabilmesi amacıyla uzay mekiklerinin yüzeyi genel olarak bor fiberleriyle güçlendirilmiş alüminyum alaşımlar ve ana yapısını matris metalin oluşturduğu ve takviye olarak da çoğu zaman seramik bir takviye fazının kullanıldığı seramiklerden üretilmişlerdir. Uzay mekiğinin üretimi ile kazanılan bilgi birikimi daha sonraki dönemlerde birçok alanda kullanılmış, otomotiv sanayinden tekstil sektörüne kadar çok sayıda teknolojik cihazın tasarım ve üretiminde öncü rol üstlenmiştir.

Şekil: *Uzay mekiğinin kalkış ve iniş anı*

1.8.2. Uzay Mekiğinin Geçmişi

Uzay mekikleri, 1981 yılında NASA tarafından kullanılmaya başlanmış olup 2011 yılında uçuşlar sona erdirilmiştir. Bu süreç içerisinde 5 adet uzay mekiği üretilerek 135 uzay seyahati gerçekleştirilmiştir.
Bunlar;

1. Columbia Uzay Mekiği (1981-2003): Yukarıda da bahsedildiği üzere Columbia, uzaya çıkan ilk uzay mekiğidir. İlk uçuş görevindeki iki astronot uzay mekiğinden çıkarak uzay yürüyüşü yapmıştır. Oluşan bir arıza sebebiyle Dünya'ya dönüşü esnasında (1 Şubat 2003) infilak etmiştir.

2. Challenger Uzay Mekiği (1983 - 1986): Columbia'dan sonra kullanılmaya başlanan ikinci uzay mekiğidir. İlk uçuşunu 4 Nisan 1983'te yapan Challenger, başarılı dokuz uçuş gerçekleştirmiştir. 28 Ocak 1986'da son uzay serüveni başladıktan 73 saniye sonra içerisinde altı astronot ve bir öğretmenden oluşan mürettebatıyla birlikte infilak etmiştir. Bu kaza en trajik kazalardan biri olarak uzun süre hafızalardan silinmemiştir.

3-Discovery Uzay Mekiği (1984-2010): Discovery, NASA'ya ait olan uzay mekiği filosunun geriye kalan üç uzay aracından biridir. Challenger ve Columbia facialarından sonra ara verilen uzay uçuşları Discovery ile yeniden başlamıştır. Discovery'nin ilk uçuşu 1984 yılında gerçekleşmiştir. Uluslararası Uzay İstasyonu'nun yapımında ve uzay araştırmalarında etkin olarak kullanılmıştır. Son uçuşunu 24 Şubat 2011 tarihinde yaparak emekliye ayrılmıştır.

4-Atlantis Uzay Mekiği (1985-2011): Atlantis inşa edilen dördüncü uzay mekiğidir. İlk uçuşunu Ekim 1985'te gerçekleştirmiştir. Atlantis son uçuşunu ise 8 Temmuz 2011'de tamamlamıştır. Bu uçuş ile NASA'nın 30 yıllık uzay mekiği programı sonlanmıştır.

5. Endeavour Uzay Mekiği (1991-2011): Endeavour, Challenger Uzay Mekiği faciasından sonra kullanılmıştır. İlk uçuşunu 12 Mayıs 1992'de yapmıştır.16 Mayıs 2011'de ise son uçuşunu yaparak emekliye ayrılmıştır. Endeavour ilk kez gece fırlatılan mekik olması açısından önemlidir.

2. TASARLA

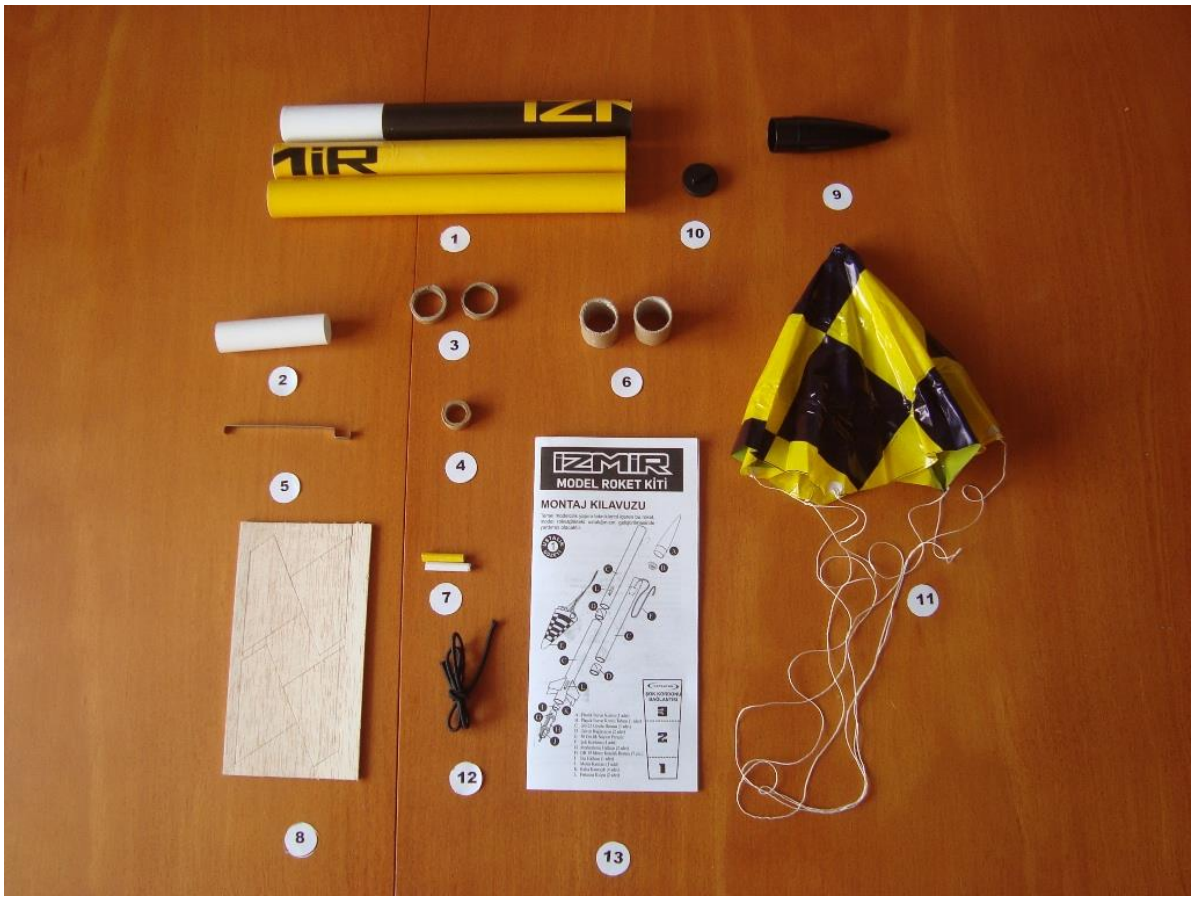
A4 kâğıdı kullanarak roket tasarlayalım. Makas, tutkal ve kâğıda ihtiyacımız var. Roketin gövdesini, burun konisini ve kanatçıkları ayrı ayrı tasarlayıp ürettikten sonra üç parçayı da birleştirerek kâğıt roketi tamamlıyoruz.

3. ÜRET

Model roket kiti kullanarak 100 metre irtifaya çıkabilecek 75 cm boyunda ve 25 mm çapında bir roket üretimi gerçekleştirilecektir. Model roket kitinde bulunan parçalar:

1. Gövde Borusu (3 adet)
2. Motor Kundak Borusu (1 adet)
3. Merkezleme Halkası (2 adet)
4. İtki Halkası (1 adet)
5. Motor Kancası (1 adet)
6. Gövde Bağlayıcı (2 adet)
7. Fırlatma Kulpu (2 adet)
8. Lazer Kesim Balsa Kanatçık (4 adet)
9. Plastik Burun Konisi (1 adet)
10. Plastik Burun Konisi Tabanı (1 adet)

11. Şok Kordonu (1 adet)
12. 30 cm'lik Naylon Paraşüt (1 adet)
13. Montaj Kılavuzu (1 adet)
14. Plastik Pipet (1 adet)
15. Roket Motoru (1 adet)

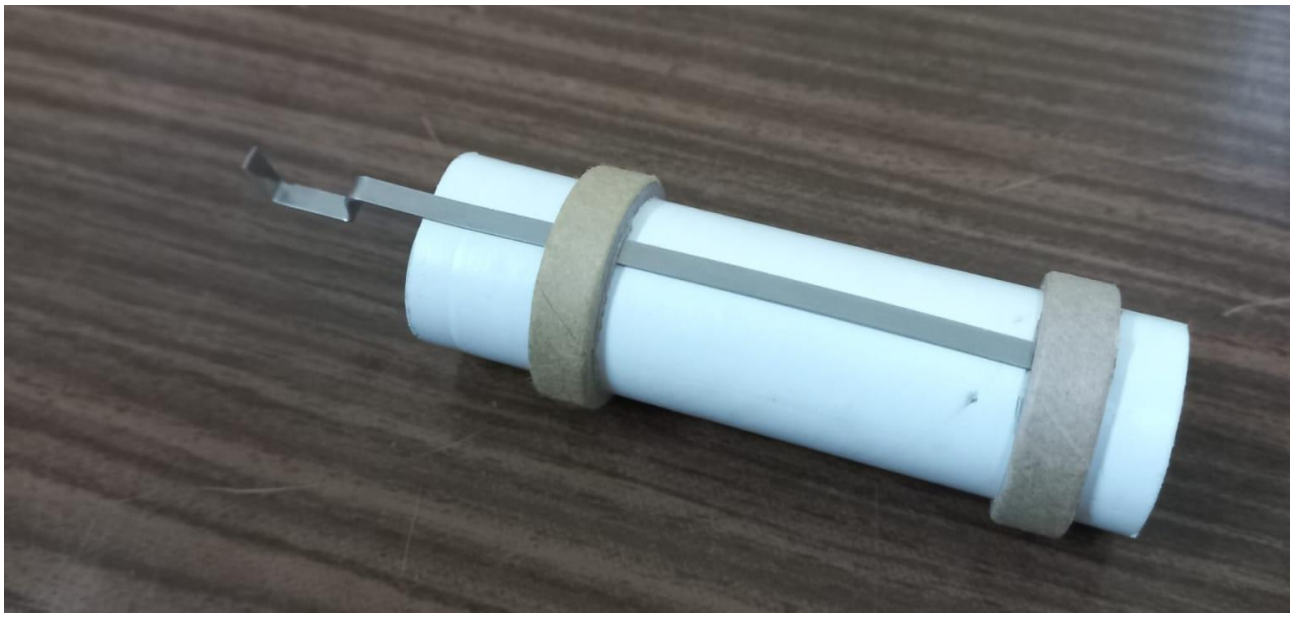


Şekil: *Roket kiti elemanları*

Aşağıda yer alan ve montaj kılavuzunda da belirtilen adımlar takip ederek üreteceğiniz roket tamamlandıktan sonra açık havada atış rampasına yerleştirerek atabilirsiniz.

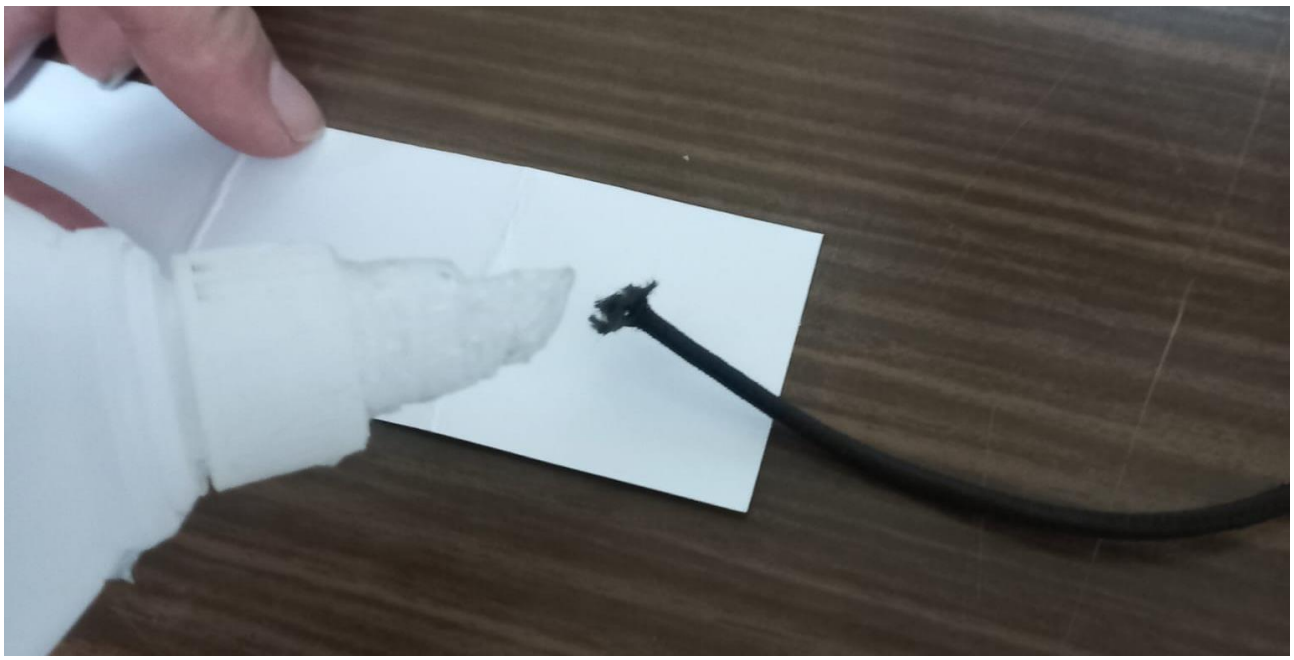
1. Bir cetvel kullanarak motor kundak borusunun ucundan itibaren 13 mm, 57 mm ve 64 mm uzaklıkta üç ayrı çizgi çiziniz. Motor kancasını yerleştirmek için 64 mm uzaklıktaki çizginin üzerine modelci bıçağını kullanarak 3mm genişlikte bir yarık açınız.

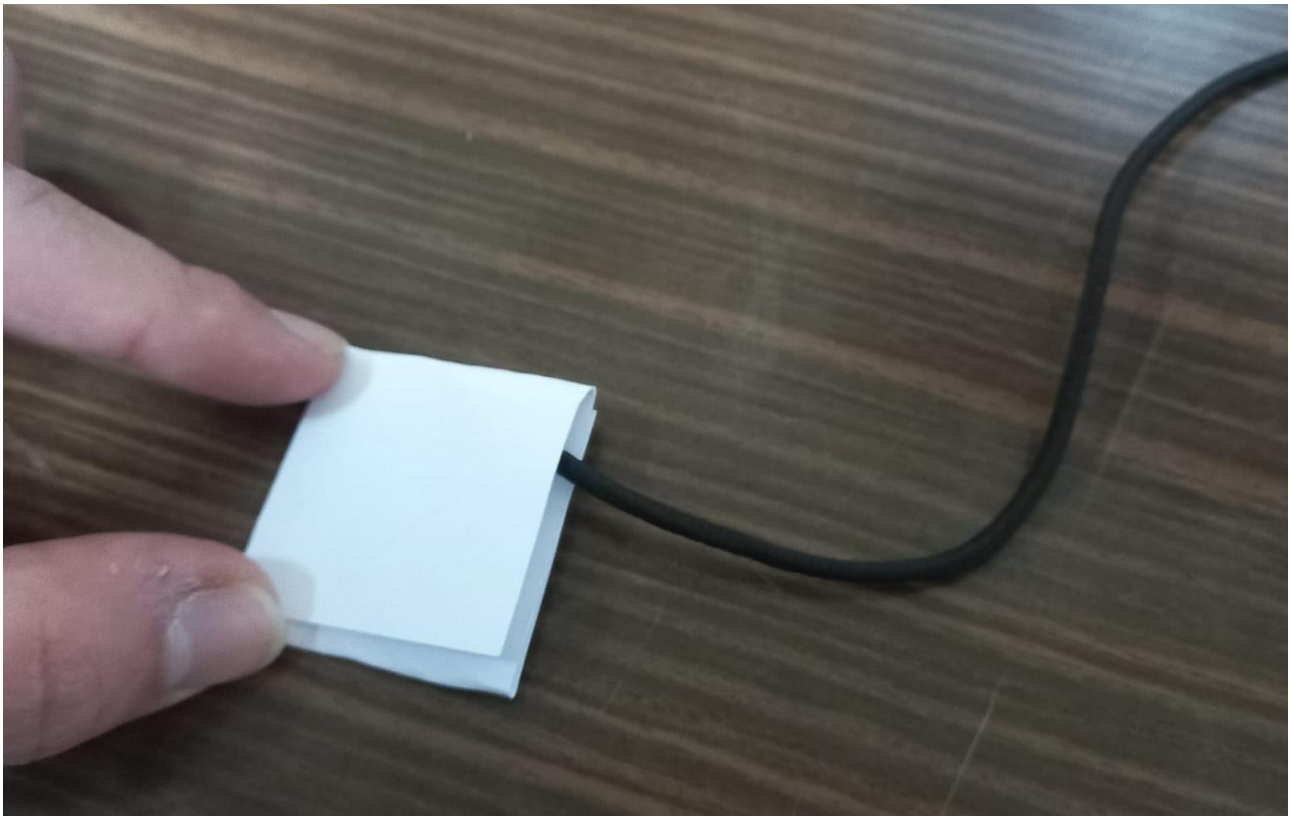
2. Çok az kavis vermiş olduğunuz motor kancasını açmış olduğunuz bu yarığa yerleştirip motor borusunun üzerine yatırınız.



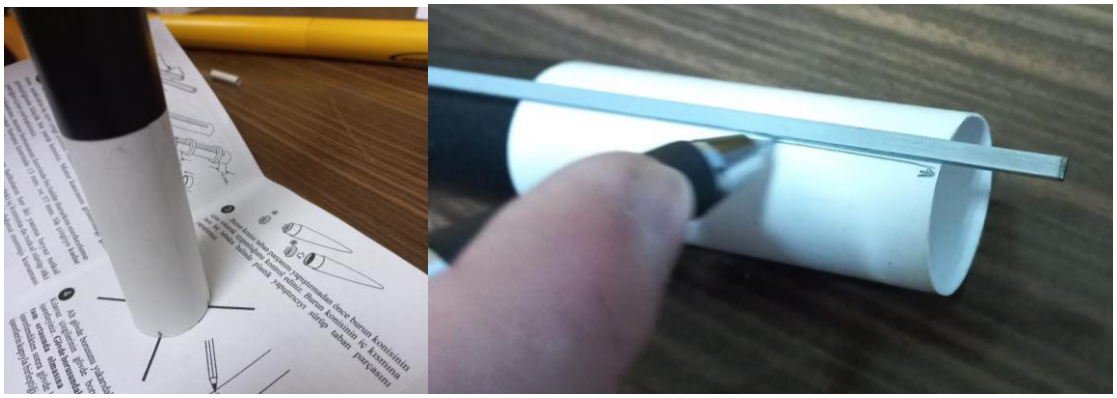
3. Motor kancası, motor borusunun üzerinde bu halde dururken merkezleme halkalarını motor borusu üzerinde 13 mm ve 57 mm'lik çizgiye kadar dikkatlice kaydırınız.

4. Kestiğiniz şablonu noktalı çizgilerden katlayarak kırıştırınız. 2 ile gösterilen yere çok az tutkal sürüp şok kordonunu hafif çapraz olacak şekilde yayınız. 1 ile gösterilen kısmı öne doğru bu kısmın üzerine katlayıp bastırınız. Aynı işlemi 3 ile gösterilen yere tutkal sürüp tekrar katlayarak yineleyiniz. Montajı parmaklarınızın arasında sıkıca bastırınız. Şok kordonu bağlantısı montajınız tamamlanmış olup kuruması için bir kenara bırakınız.

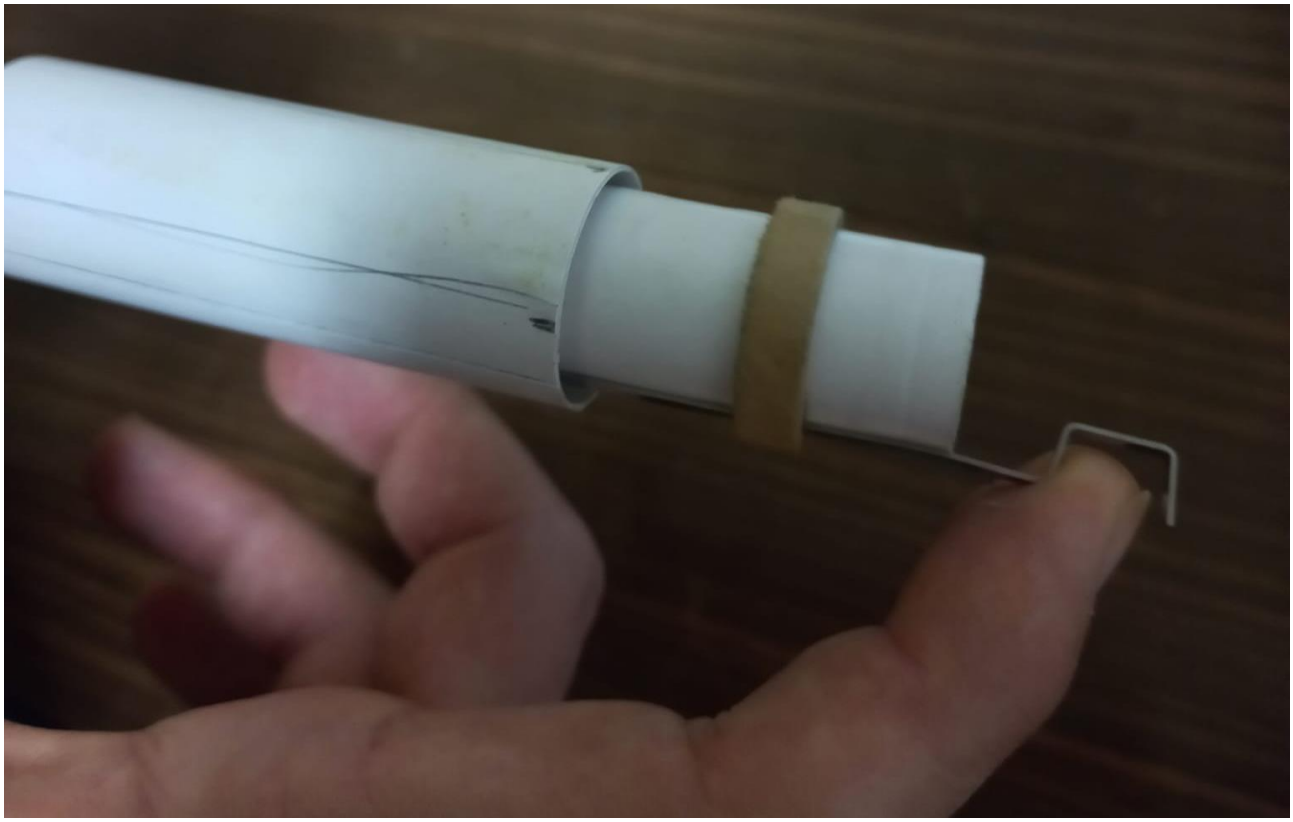




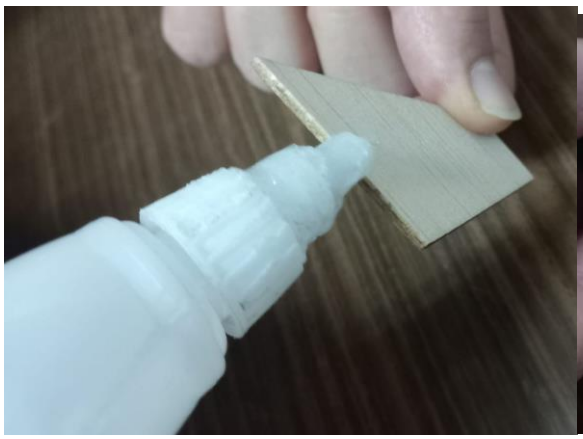
5. Alt gövde borusu üzerinde kanatçık yerlerinin belirlenmesi için montaj kılavuzunun ikinci sayfasındaki *boru işaretleme* kullanmak gerekmektedir. Motor kundağını gövde borusuna yerleştirmeden evvel gövde borusunu kılavuz üzerinde dik bir pozisyonda tutunuz.



6. Gövde borusunun iç tarafındaki ölçüm yerini kalemle çepeçevre çiziniz. Bir parça balsa artığını ya da bir çubuk kullanarak, gövde borusunun işaretlenmiş iç tarafına çepeçevre yapışkan sürünüz. Motor kundak montajının ön tarafını gövde borusunun yapıştırılmış kısmından içeriye doğru hafifçe itiniz.



7. **Levhadaki diğer kanatçıklara zarar vermemek için bu ayırma işlemini çok dikkatli yapınız.** Balsa levhadan ayırdığınız 4 adet kanatçığın bütün kenarlarını eşleştirip bir araya getiriniz.

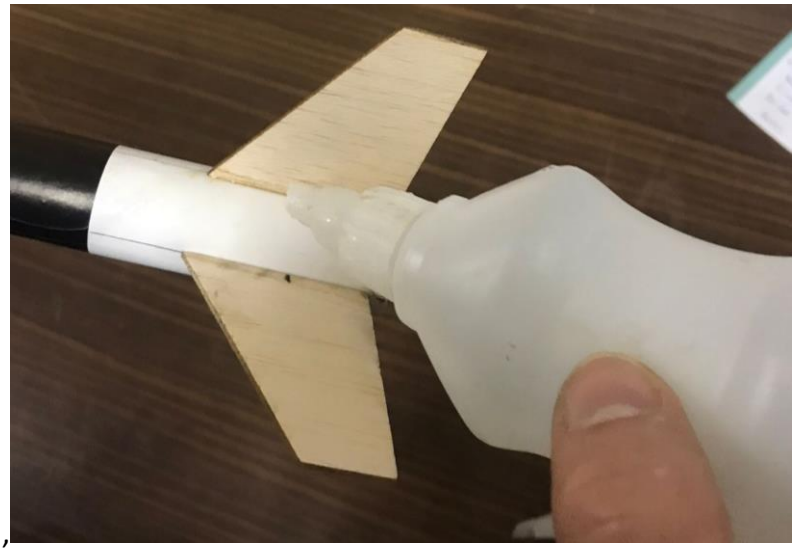


8. Her iki gövde bağlayıcısının ucundan itibaren 13mm ölçüp işaret koyunuz. Entegrasyon borusu ve gövde borularının içine çepeçevre yapıştırıcı sürünüz.

9. Orta ve üst gövdeyi birleştirdikten sonra aynı işlemi kanatçıkların olduğu motor gövde için de uygulayınız.

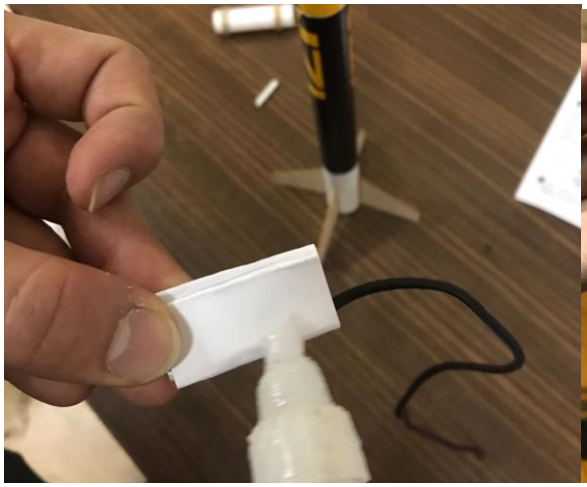
10. Motor kancasının tam arkasını referans alarak, motor gövde arkasından ve 2 kanatçığın ortasından 38mm ve 317 mm olarak ölçtükten sonra fırlatma kulplarını (ray butonunu) yapıştırıcı ile yerleştiriniz.

11. Montajlanan kanatçıklara yapıştırıcı sürerek montajı güçlendirmelisiniz.

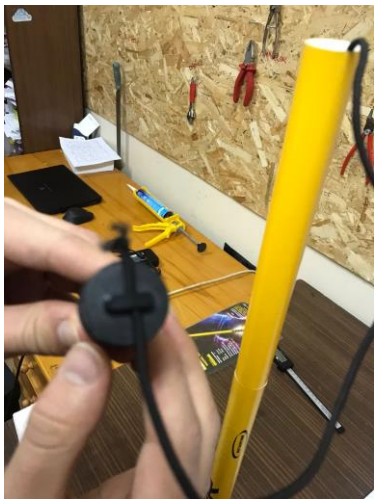


,

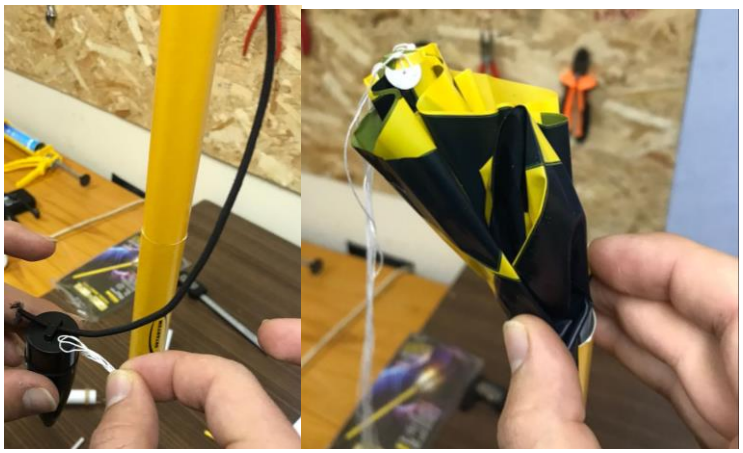
12. Şok kordu için hazırlamış olduğunuz parçayı gövde içerisine yapıştırıcı ile sabitleyiniz.



13. Önce burun konisi kapağını, burun konisinin arka kısmına ortalı bir şekilde yapıştırınız. Daha sonra bu kapağa şok kordunuzu bağlayınız.

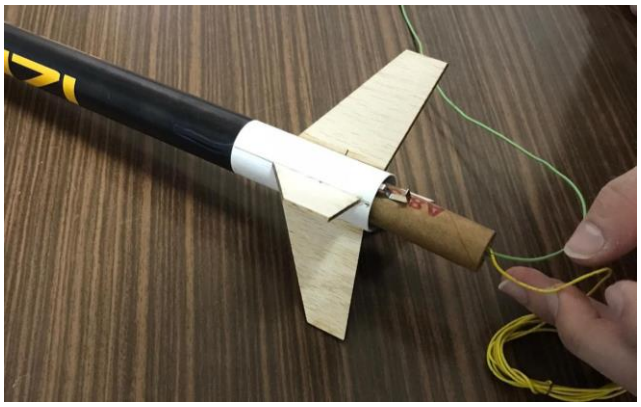


14. Şok kordonu bağlantısının olduğu kısma paraşüt montajınızı da bağlayarak yapınız (Paraşüt iplerinizin karışmadığından emin olunuz).

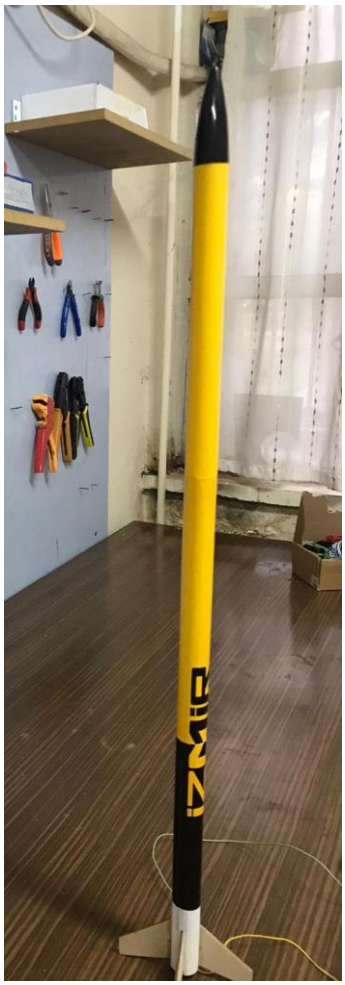


15. Ateşleme fitilinizi motorun nozul kısmındaki delikten içeriye yerleştiriniz.

16. Ateşleme telinizin motorun içerisinde sağlam bir şekilde durduğundan emin olduktan sonra, motorunuzu görseldeki gibi motor kundağının içerisine yerleştiriniz.



17. Tebrikler! Model roket kitini tamamlamış bulunmaktasınız. Roketinizi fırlatma rampasına, fırlatma kollarından geçirerek ateşlemenizi yapabilirsiniz.



4. DEĞERLENDİR

- Model roket tasarımında kullanılan paket programlar var mıdır?
- Üretilen roket daha yüksek irtifalara nasıl çıkartılabilir?
- Roketler içerisinde taşınabilecek faydalı yükler neler olabilir?
- Ticari roketlerin üretiminde hangi mesleklere sahip mühendislerin görev alması beklenir?

10. HAFTA: UYDU SİSTEMLERİ

Ön Bilgi:

- Temel elektrik ve elektronik terminolojisini bilmek.
- Elektronik malzemeleri tanımak.
- 3D yazıcıdan baskı alabilmek.
- Arduino ile programlama yapabilmek.
- Deneyap kart temel yapısına hâkim olmak.

Haftanın Kazanımları:

- Uydu sistemlerini tanır
- Uyduların uzayda yerleşimi ve çalımasını prensiplerini kavrar.
- Uydu geliştirme malzemelerini ilişkilendirir.
- Uydu çeşitlerini sınıflar.
- Uydu haberleşme usulleri kavrar.
- Model yapay uydu oluşturur.
- Model uydu için temel yazılım üretir.
- Yenilenebilir enerji kaynakları ile uyduların enerji sistemlerini ilişkilendirir.

Haftanın Amacı:

Uydu sistemleri ve uyduların haberleşme altyapılarını öğretmek.

Model uydu oluşturma becerisini kazandırmak.

Kullanılacak Malzemeler:

- Önceden 3 boyutlu çıktıları alınmış uydu bileşenleri.
- Güneş paneli.
- Solar pil şarj modülü.
- Lipo pil.
- Çeşitli ölçülerde kablo ve bağlantı elemanları.
- Deneyap kart.
- Arduino IDE arayüzü.

Haftanın İşlenişi:

Gözle: Temel uydu sistemlerinin tanıtılması. Yapay uydu çeşitleri ve çalışma şekillerinin anlatılması.

Uygula: Deneyap kart üzerinde güneş panelinden beslenerek çalışan bir gözlem uydusu çalıştırma.

Tasarla: Deneyap kart üzerinde bir yapay gözlem uydusu tasarlama.

Üret: 3B çıktısı alınmış uydu gövde bileşenleri ile güneş paneli, Lipo batarya kullanarak Deneyap kartı besleyecek ve aldığı görüntüyü yer istasyonuna aktaracak bir yapay uydu üretilecek.

Değerlendir: Modeli oluşturulan yapay gözlem uydusu sisteminin gerçek yapay uydularla benzer ve farklı yönleri değerlendirme.

UYARI

Bu dersin üret aşamasındaki parçalar 3 boyutlu yazıcıda basılmalıdır. Bu nedenle sınıftaki grup sayısı kadar parça ders öncesinde 3 boyutlu yazıcıda basılarak ders sırasında hazır halde olmalıdır.

Parçalara ait 3 boyutlu çizim ve STL dosyalarına
https://owncloud.tubitak.gov.tr/index.php/s/pzTAtdzksc2UCBz

adresinden erişilebilir.

1. GÖZLE

Dersin bu bölümünde eğitmen öğrencilere “Yapay Uydu” kavramını açıklar ve uydu sistemleri hakkında genel bilgiler verir. Uzayda bildiğiniz uydu türleri nelerdir? Uydular uzayda nasıl duruyor? Uydular ne kadar süre çalışır? Ömrü biten uydu ne olur? gibi sorular yönelterek eğitmen öğrencilerle konuyu soru cevap şeklinde sınıf içi tartışarak detaylandırır.

1.1. Uydu Sistemleri

Uydu, uzayda daha büyük bir nesnenin etrafında konumlanan veya dönen bir nesnedir. İki tür uydu vardır: Doğal (Dünya yörüngesindeki Ay gibi) veya yapay (Dünya yörüngesindeki Uluslararası Uzay İstasyonu gibi).

Güneş sisteminde düzinelerce doğal uydu vardır ve hemen hemen her gezegenin en az bir uydusu/ayı vardır. Örneğin Satürn'ün en az 53 doğal uydusu var. 2004 ile 2017 arasında yapay bir uydusu da vardı: Halkalı gezegeni ve uydularını keşfeden Cassini uzay aracı.

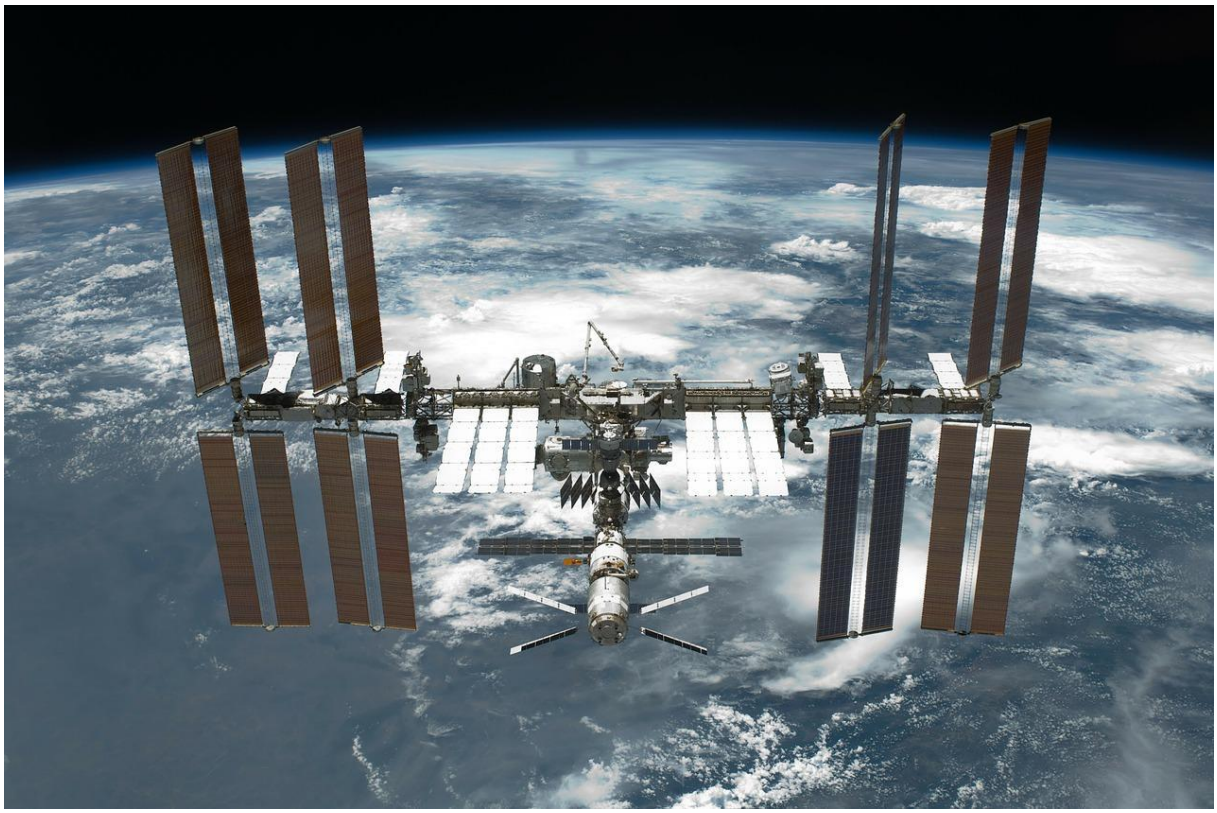
Yapay bir uydu, haberleşme, gözlem, ölçüm vb. işler yapmak üzere tasarlanmış Dünya'dan sinyal alma ve bu sinyalleri bir transponder (entegre bir radyo sinyalleri alıcısı ve vericisi) kullanarak geri iletme yeteneğine sahip bağımsız bir iletişim sistemidir. Bir uydu, fırlatma sırasında saatte 28.100 km (17.500 mil) yörünge hızına kadar hızlanma şokuna ve öngörülen operasyonel ömrü boyunca radyasyona ve aşırı sıcaklıklara maruz kalabileceği düşmanca bir uzay ortamına dayanmalıdır. Ayrıca uydu fırlatmanın maliyeti ve ağırlığına bağlı olarak oldukça maliyetli olduğu için uyduların hafif olması gerekir. Bu zorlukların üstesinden gelmek için uydular küçük olmalı, hafif ve dayanıklı malzemelerden yapılmalıdır. Bakım veya onarım olasılığı olmadığından uzay boşluğunda yüzde 99,9'dan fazla çok yüksek bir güvenilirlikle çalışmalıdırlar.

Şekil 1 Bir Gözlem Uydusu Örneği

1.2. Uydu Sistemlerinin Kısa Tarihçesi

İlk yapay uydu, 4 Ekim 1957'de havalanan bir şişme top büyüklüğünde olan Rusya yapımı Sputnik'ti. Bu başarının ardından, 3 Kasım 1957'de Sovyetler, Laika adında bir köpek taşıyan daha büyük bir uydu olan Sputnik 2'yi fırlattı. Amerika Birleşik Devletleri'nin ilk uydusu 31 Ocak 1958'de Explorer 1'di. Uydu, Sputnik 2'nin kütlesinin yalnızca yüzde 2'si kadar ve 13 kg ağırlığındaydı.

Sputnikler ve Explorer 1, Amerika Birleşik Devletleri ile Sovyetler Birliği arasında en azından 1960'ların sonlarına kadar süren bir uzay yarışının açılış çekimleri oldu. Sovyetler Birliği Dünya'nın ilk uzay istasyonunu, 1971'de fırlatılan Salyut 1'i inşa etti. Bunları Amerika Birleşik Devletleri'ndeki Skylab ve Sovyetler Birliği'ndeki Mir gibi diğer istasyonlar izledi.



Şekil 2 Uluslararası Uzay İstasyonu

Bu hızlı uzay yarışında diğer ülkeler de kendi toplumlarının faydalanması için kendi uydularını uzaya göndermeye başladı. Hava durumu uyduları, uzak bölgeler için bile tahminleri iyileştirdi. Landsat serisi gibi kara gözlemci uyduları, zaman içinde ormanlar, su ve Dünya yüzeyinin diğer kısımlarındaki değişiklikleri izledi. Telekomünikasyon uyduları uzun mesafeli telefon görüşmeleri yaptı ve nihayetinde dünyanın dört bir yanından canlı televizyon yayınları hayatın normal bir parçası oldu. Daha sonraki nesiller internet bağlantılarına yardımcı oldu.

Türkiye'nin ilk haberleşme uydusu Türksat 1B, 11 Ağustos 1994 yılında uzaya fırlatıldı ve 42° Doğu yörüngesine başarıyla yerleştirildi. 10 Temmuz 1996'da Türksat 1C, 11 Ocak 2001'de Türksat 2A, 13 Haziran 2008'de Türksat 3A, 14 Şubat 2014 Türksat 4A, 16 Ekim 2015'te Türksat 4B uzaya fırlatıldı. En güncel uzaya gönderilen haberleşme uydumuz ise Türksat 5B, 19 Aralık 2021'de Space X firmasına ait Falcon 9 roketi ile uzaya gönderildi.

Haberleşme uydularımızın yanı sıra Rasat, Göktürk-1 ve Göktürk-2 uyduları gözlem amacıyla kullanılmaktadır.

Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından desteklenen İMECE Uydu Projesi, Ocak 2017'de başlamıştır. Haziran 2020'de uzay aracının termal yapısal verimlilik modülünün montaj entegrasyon faaliyetleri tamamlanmış ve uzay aracında titreşim testleri başlamıştır. Bu testleri uçuş modülünün montaj faaliyetleri izlemiştir. Uydu, Transporter-7 görevinin bir parçası olarak SpaceX'in Falcon 9 Blok 5 roketiyle 15 Nisan 2023'te fırlatılmıştır.

Bilgisayarların ve diğer donanımların minyatürleştirilmesiyle, yörüngede bilim, telekomünikasyon veya diğer işlevleri yapabilen çok daha küçük uydular göndermek artık mümkün. Şirketler ve üniversiteler için "CubeSats" veya sıklıkla düşük Dünya yörüngesini dolduran küp şeklindeki uydular oluşturmak artık çok yaygın bir uygulama haline geldi. Bu kapsamda Türkiye'nin ilk küp uydusu 2009 yılında İstanbul Teknik Üniversitesi tarafından hazırlanarak uzaya gönderilmiştir.

1.3. Uyduların Uzaya Taşınması

Uzaya bir uydu fırlatmak, onu doğru yörüngeye itmek için çok güçlü çok aşamalı bir roket gerektirir. Yörünge fırlatma sistemleri roketler ve faydalı yükleri Dünya yörüngesinin içine veya ötesine yerleştirebilen sistemlerdir. Roket sistemleri dersinde detaylarını gördüğünüz yapıdaki araçlar uyduları istenilen yörünge seviyesine taşır. Uydu fırlatma sağlayıcıları, Cape Canaveral, Florida'daki Kennedy Uzay Merkezi, Kazakistan'daki Baikonur Uzay Üssü, Fransız Guyanası'ndaki Kourou, Kaliforniya'daki Vandenberg Hava Kuvvetleri Üssü, Çin'deki Xichang ve Japonya'daki Tanegashima Adası gibi sitelerden uyduları fırlatmak için tescilli roketler kullanır.

Şekil 9 SpaceX Roketi

1.4. Yörüngede Durma

Bir uydu en iyi şekilde bir mermi veya üzerinde hareket eden tek bir kuvvete (yer çekimi) sahip bir nesne olarak anlaşılır. Teknik olarak konuşursak, Karman Hattını 100 kilometre yükseklikte geçen her şey uzayda kabul edilir. Bununla birlikte, bir uydunun Dünya'ya geri düşmesini hemen durdurmak için saniyede en az 8 km hızla gitmesi gerekir.

Bir uydu yeterince hızlı hareket ediyorsa, sürekli olarak Dünya'ya "düşecektir", ancak Dünya'nın eğriliği, uydunun yüzeye çarpmak yerine gezegenimizin etrafına düşeceği anlamına gelir. Atmosferik moleküllerin sürüklenmesi uyduları yavaşlatacağından, Dünya'ya daha yakın seyahat eden uydular daha fazla düşme riski altındadır.

Dünya çevresinde kabul edilen birkaç yörünge bölgesi vardır. Biri yaklaşık 160 ila 2.000 km arasında uzanan düşük Dünya yörüngesi olarak adlandırılır. Burası, Uluslararası Uzay İstasyonu ISS'nin yörüngede döndüğü ve uzay mekiğinin işini yaptığı bölgedir. Aslında Apollo'nun aya uçuşları dışında tüm insan görevleri bu bölgede gerçekleşti. Çoğu uydu da bu bölgede çalışır.

Bununla birlikte, coğrafi sabit veya jeosenkron yörünge, iletişim uydularının kullanması için en iyi noktadır. Bu, 35.786 km yükseklikte, Dünya'nın ekvatorunun üzerinde bir bölgedir. Bu yükseklikte, Dünya etrafındaki "düşme" oranı, Dünya'nın dönüşü ile yaklaşık olarak aynıdır. Bu da uydunun neredeyse sürekli olarak Dünya üzerindeki aynı noktanın üzerinde kalmasını sağlar. Böylece uydu, yerdeki sabit bir antenle sürekli bir bağlantı kurarak güvenilir iletişim sağlar. Jeo-durağan uydular ömürlerinin sonuna ulaştığında, protokol, yerlerini yeni bir uydunun alması için uzaydaki konumlarından ayrılırlar.

Bazı uydular ekvator çevresinde en iyi şekilde kullanılırken, diğerleri kutupsal yörüngeler için daha uygundur. Dünya'yı bir kutuptan diğerine çevreleyenler, böylece kapsama bölgeleri kuzey ve güney kutuplarını içerir. Kutup yörüngeli uydulara örnek olarak hava uyduları ve keşif uyduları söylenebilir.

Bir uydunun ana bileşenleri, sinyalleri alan ve yeniden ileten antenler ve transponderleri içeren iletişim sistemi, güç sağlayan güneş panellerini içeren güç sistemi ve uyduyu hareket ettiren roketleri içeren tahrik sisteminden oluşur. Bir uydu, kendisini doğru yörünge konumuna getirmek ve bu konumda ara sıra düzeltmeler yapmak için kendi tahrik sistemine ihtiyaç duyar. Jeostatik yörüngedeki bir uydu, Ay ve Güneş'in yerçekimi kuvveti nedeniyle her yıl kuzeyden güneye veya doğudan batıya bir dereceye kadar sapabilir. Bir uydunun, konumunda ayarlamalar yapmak için ara sıra ateşlenen iticileri vardır. Bir uydunun yörünge konumunun korunmasına “istasyon tutma”, uydunun iticileri kullanılarak yapılan düzeltmelere ise “tutum kontrolü” denir. Bir uydunun ömrü, bu iticilere güç sağlamak için sahip olduğu yakıt miktarına göre belirlenir. Yakıt bittiğinde, uydu sonunda uzaya sürüklenir ve çalışma dışı kalır ve uzay enkazı haline gelir. Bir uydunun izleme telemetri ve kontrol (TT&C) sistemi, uydu ile yerdeki TT&C arasında iki yönlü bir iletişim bağlantısıdır. Bu, bir yer istasyonunun bir uydunun konumunu izlemesine ve uydunun tahrik, termal ve diğer sistemlerini kontrol etmesine olanak tanır. Ayrıca bir uydunun sıcaklığını, elektrik voltajlarını ve diğer önemli parametrelerini de izleyebilir.

1.5. Uydularda Enerji

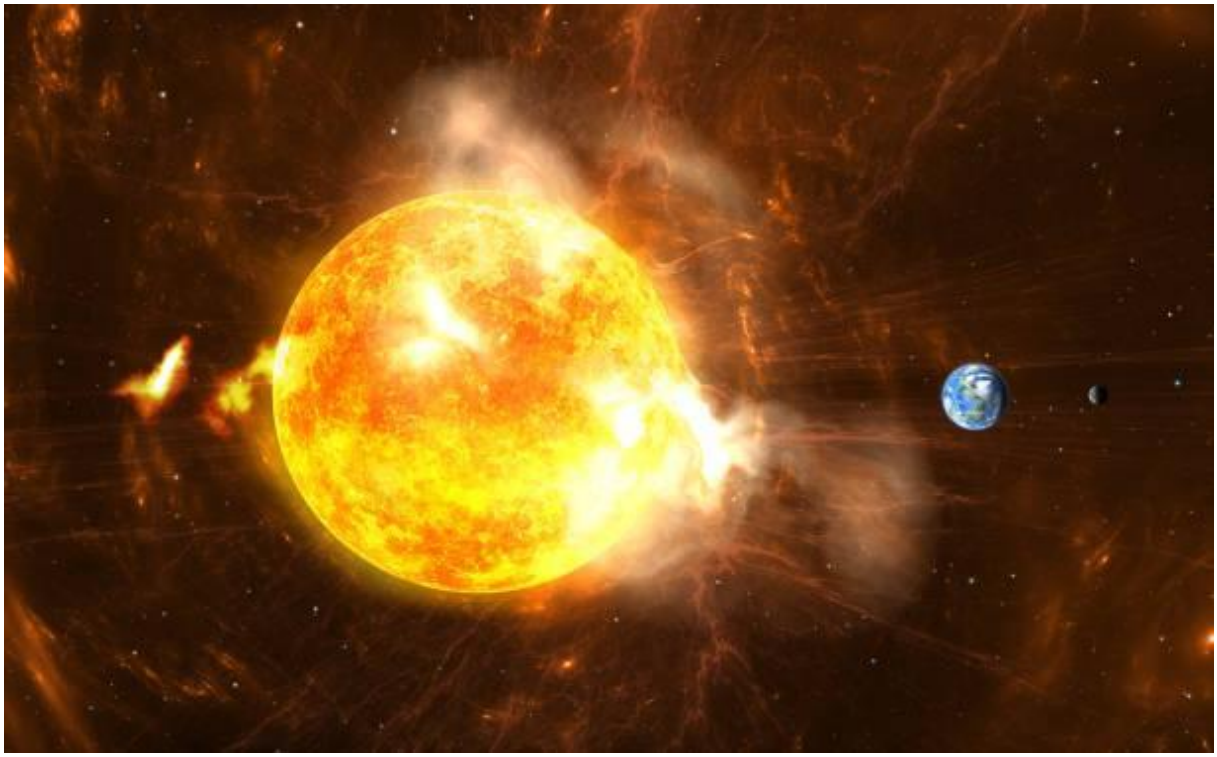
Yörüngedeki bir uydu, tüm ömrü boyunca sürekli çalışmak zorundadır. Elektronik sistemlerini ve iletişim yükünü çalıştırabilmek için dâhili güce ihtiyaç duyar. Ana güç kaynağı, uydunun güneş panelleri tarafından kullanılan güneş ışığıdır. Bir uyduda ayrıca Güneş, Dünya tarafından engellendiğinde güç sağlamak için piller bulunur. Piller, güneş ışığı olduğunda güneş panelleri tarafından üretilen aşırı akımla yeniden şarj edilir.

Bir yapay uydudaki elektrik güç sisteminin (EPS) birincil rolü, uydudaki diğer sistemlerin etkin bir şekilde çalışması için gerekli elektrik gücünü sağlamaktır. Gücün kaynağı, doğrudan güneş radyasyonuna veya albedodan dolaylı radyasyona maruz kalan güneş panellerinden toplanan enerjidir. Uydunun kendisi, farklı sensörler ve alt sistemler için birden fazla voltaj seviyesine ihtiyaç duyabilir. Bu seviyeleri yönetmek, sistemin başka bir işlevidir; EPS, çeşitli voltajlarda gerekli miktarda elektrik gücü sağlayabilen bir güç koşullandırma ünitesine sahiptir. Ayrıca uzay aracı durumunun izlenmesinde önemli bir rol oynar.

Şekil 4 Küp Uydularda Kullanılan EPS Örnekleri

1.6. Uzay Şartlarına Dayanım

Uydular −150 °C (−238 °F) ila 150 °C (300 °F) arasındaki aşırı sıcaklıklarda çalışır ve uzayda radyasyona maruz kalabilir. Radyasyona maruz kalabilen uydu bileşenleri, alüminyum ve diğer radyasyona dayanıklı malzemelerle korunmaktadır. Bir uydunun termal sistemi, hassas elektronik ve mekanik bileşenlerini korur ve sürekli çalışmasını sağlamak için onu optimum çalışma sıcaklığında tutar. Bir uydunun termal sistemi çok ısındığında soğutma mekanizmalarını veya çok soğuduğunda ısıtma sistemlerini etkinleştirerek hassas uydu bileşenlerini sıcaklıktaki aşırı değişikliklerden korur.



Şekil 5 Uzayda Güneş Patlamaları

2. Kullanım Amaçlarına Göre Uydu çeşitleri

Uydular, uygulamalarına göre veya yerleştirildikleri yörüngeye göre sınıflandırılabilir. Yörüngeler, yüksekliklerine, ekvator düzlemine göre eğimlerine, eksantriklik, senkronizasyon parametreleri, gibi değişik türlerde sınıflandırılır. Uygulamalarına göre yaygın kullanılan uydu türleri şunlardır:

1. Haberleşme uyduları
2. Meteoroloji uyduları
3. Seyrüsefer uyduları
4. Dünya gözlem uyduları
5. Astronomik uydular
6. Minyatürleştirilmiş uydular.

2.1. Haberleşme Uyduları

İletişim uydusu, telefon, televizyon, radyo, internet gibi sinyalleri uzun mesafelere ileten bir uydudur. Bilgiyi tel üzerinden iletmek için elektrik sinyali kullanılır. Ancak, hava saf bir yalıtkan olduğu için elektrik sinyali veya elektrik akımı kablosuz olarak iletilmez. Bu nedenle, elektrik akımının veya elektrik sinyalinin akışına direnir. Bilgileri kablosuz olarak iletmek için ışık sinyalleri veya elektromanyetik dalgalar kullanılır. Ancak ışık sinyalleri veya elektromanyetik dalgalar dünyanın eğriliği etrafında bükülemezler. Bu nedenle, bilgiyi veya sinyali uzun mesafelerde iletmek için, sinyalleri yeniden yönlendirmek için tek bir uydu veya birden fazla uydu kullanılır.

İletişim uyduları, 1 kg'dan daha hafif olan mikro uydulardan, 6,500 kg'ın üzerindeki büyük uydulara kadar çeşitlilik gösterir. Minyatürleştirme ve dijitalleştirmedeki gelişmeler, yıllar içinde uyduların kapasitesini önemli ölçüde artırdı. Early Bird, yalnızca bir TV kanalı gönderebilen tek bir aktarıcıya sahipti. Buna karşılık Boeing 702 serisi uydular, 100'den fazla transpondere sahip olabilir ve dijital sıkıştırma teknolojisinin kullanılmasıyla her transponder, bir uydu üzerinden 1.600'den fazla TV kanalı sağlayan 16 kanala sahip olabilir.

Türkiye'nin faydalı yük kapasitesi en yüksek uydusu Türksat 5B, 4,5 ton fırlatma ağırlığına ve 15 kilovat güç kapasitesine sahip bulunuyor. Türkiye'nin yanı sıra Orta Doğu'nun tamamı, Basra Körfezi, Kızıldeniz, Akdeniz, Kuzey ve Doğu Afrika, Nijerya, Güney Afrika ve yakın komşu ülkeleri içeren geniş kapsama alanında hizmet verecek Türksat 5B, frekansın tekrar kullanımı ve çoklu hüzme kapsama konseptlerinin kullanıldığı Ka-Bant faydalı yüküyle toplamda 55 Gbps'den daha fazla veri iletim kapasitesi sağlayacak. Türksat 5B ile Türkiye'nin uzayda aktif haberleşme uydusu sayısı 5'tir

Şekil 6 Haberleşme Uydu Yer İstasyonu

2.2. Meteoroloji Uyduları

Meteoroloji uyduları hava olaylarını küresel olarak inceleme olanağı sağlayan uzaktan algılama cihazlarıdır. Dünya çevresindeki yörüngelerinde hareket ederlerken, sensörleri (radyometre) tarafından kaydedilen verileri belirli aralıklarla yer istasyonlarına gönderirler. Uyduların en önemli faydalarından biri, yer gözlem istasyonları kurulamadığı için verilerin toplanamadığı okyanus, çöl, dağlık alanlar, kutup bölgeleri vs. gibi çok geniş alanlardan meteorolojik bilgilerin elde edilmesidir.

Uyduların uzaktan algılama sistemleri cisimler tarafından yansıtılan ve cisimlerin vücut sıcaklığına bağlı olarak yaydıkları elektromagnetik radyasyonun, uzaya yerleştirilen platformlar (uydu) üzerinde bulunan radyometreler (pasif algılama) ve radarlar (aktif algılama) tarafından ölçülmesi prensibine dayanır. Bulutluluk, ozon miktarı ve konsantrasyonu, buzul alanlarının, atmosferik sıcaklık ve nem profillerinin, yağış miktarının tespiti, kara ve deniz yüzeyi sıcaklıklarının belirlenmesi pasif algılama ile okyanus dalga boyu, dalga yüksekliği, deniz yüzeyi rüzgâr hızı ve yönünün tespiti aktif algılama ile yapılır.

Meteorolojik uydular yörüngelerine göre temel olarak iki kısma ayrılırlar:

Geostationary (Sabit Yörüngeli) Uydular

Polar (Kutupsal Yörüngeli) Uydular

2.3. Seyrüsefer Uyduları

Uydu navigasyonu, orta dünya yörüngesinden radyo sinyallerini ileten küresel bir uydu ağına dayanmaktadır. Uydu navigasyonu kullanıcıları en çok Amerika Birleşik Devletleri tarafından geliştirilen ve işletilen 31 Küresel Konumlandırma Sistemi (GPS) uydusuna aşinadır. Diğer üç sağlayıcı de benzer hizmetler sunar. Toplu olarak, bu sistemler Küresel Navigasyon Uydu Sistemleri (GNSS) olarak adlandırılır. Diğer sistemler, Rusya Federasyonu tarafından geliştirilen ve işletilen GLONASS, Avrupa Birliği tarafından geliştirilen ve işletilen Galileo ve Çin tarafından geliştirilen ve işletilen BeiDou'dur. Tüm sağlayıcılar, uluslararası topluluğa kendi sistemlerinin ücretsiz kullanımını sunmaktadır.

Temel GPS hizmeti, kullanıcılara dünyanın herhangi bir yerinde veya yakınında, zamanın %95'inde yaklaşık 7,0 metre doğruluk sağlar. Bunu başarmak için 31 uydunun her biri, alıcıların en az dört uydudan gelen sinyallerin bir kombinasyonu aracılığıyla konumlarını ve zamanlarını belirlemelerini sağlayan sinyaller yayar. GPS uyduları, son derece doğru zaman sağlayan atomik saatler taşır. Zaman bilgisi, uydu tarafından yayınlanan kodlara yerleştirilir. Böylece bir alıcı, sinyalin yayınlandığı zamanı sürekli olarak belirleyebilir. Sinyal, bir alıcının uyduların konumlarını hesaplamak ve doğru konumlandırma için gereken diğer ayarlamaları yapmak için kullandığı verileri içerir. Alıcı, alıcıdan uyduya olan mesafeyi veya aralığı hesaplamak için sinyal alım zamanı ile yayın zamanı arasındaki zaman farkını kullanır. Alıcı, iyonosfer ve troposferin neden olduğu sinyal hızındaki yayılma gecikmelerini veya düşüşlerini hesaba katmalıdır. Üç uydunun menzili ve sinyal gönderildiğinde uydunun konumu hakkında bilgi ile alıcı kendi üç boyutlu konumunu hesaplayabilir. Bu üç sinyalden mesafeleri hesaplamak için GPS ile senkronize edilmiş bir atomik saat gereklidir. Ancak, alıcı dördüncü bir uydudan ölçüm alarak atom saatine olan ihtiyacı ortadan kaldırır. Böylece alıcı enlem, boylam, yükseklik ve zamanı hesaplamak için dört uydu kullanır.

Şekil 7 GPS IIR(M) Uydusu

2.4. Gözlem Uyduları

Yer Gözlem uyduları, sahip oldukları yörünge tipine, taşıdıkları faydalı yüke ve görüntüleme cihazlarının bakış açısından, sensörlerin uzamsal çözünürlüğüne, spektral özelliklerine ve alan genişliğine göre farklılık gösterir. Tüm bu parametreler, uydu görevinin hedeflediği uygulamaya bağlı olarak görev tanımının başında tasarlanmaktadır.

Hava durumunu büyük ölçeklerde ve yüksek frekansta izlemek için, bir uydunun sabit bir yörüngede olması uygundur. Böyle bir yörüngede bir uydu, neredeyse tüm yarım küreyi sürekli olarak görüntüleyebilir. Ancak yörünge çok yüksek olduğundan (Dünya'nın yaklaşık 36 000 km üzerinde), yüksek bir uzaysal çözünürlüğün elde edilmesi zordur. Ancak kıtalar üzerinde bulutların izlenmesi gibi uygulamalar için yüksek bir uzaysal çözünürlük gerekli değildir.

Bir buzul gölünün izlenmesi veya bir depremde yıkılan binaların haritalanması gibi çok özel bir alanın yüksek çözünürlüklü görüntülenmesini gerektiren uygulamalar için, yüksek çözünürlüklü bir sensör gerekli olacaktır. Böyle bir sensör genellikle dar bir alana sahip olacak ve LEO (QuickBird uydusu durumunda Dünya'nın 600 km üzerinde olduğu gibi) olarak adlandırılan Alçak Dünya Yörüngesinde bir uydu üzerinde olacaktır. Böyle bir yörüngede, uydunun Dünya'ya göre göreceli hareketi nedeniyle aynı alanı sürekli olarak izlemek mümkün değildir. Görüntüler yalnızca belirli bir alan üzerinden uydu üzerinden geçtiğinde alınabilir.

RASAT Araştırma Uydusu, Türkiye'nin ve TÜBİTAK UZAY'ın BiLSAT uydusundan sonra sahip olduğu ikinci uzaktan algılama uydusudur. Yüksek çözünürlüklü optik görüntüleme sistemine ve Türk mühendislerce tasarlanıp geliştirilen yeni modüllere sahip olan RASAT, Türkiye'de tasarlanıp üretilen ilk yer gözlem uydusudur.

Şekil 8 RASAT Uydusu

TÜBİTAK Uzay Enstitüsü ve TUSAŞ ortaklığında geliştirilen GÖKTÜRK-2 Uydusu, 18 Aralık 2012'de Çin'in Jiuguan fırlatma üssünden Long March-2D fırlatma sistemi ile yörüngesine yerleşti. GÖKTÜRK-2 Uydumuz, uzayda geçirdiği 9 yıllık süreçte dünya etrafında 48 bin 200 tur attı. Dünyanın her yerinden kaydettiği görüntüleri Ankara'daki yer istasyonuna ileten uydumuz ile 18 bin 700 defa başarılı haberleşme sağlanırken toplam 29,4 milyon km2'de görüntüleme yapıldı. Yüksek çözünürlüklü Elektro-Optik (E/O) kamera taşıyan GÖKTÜRK-1 Yer Gözlem ve Keşif Uydusu Aralık 2016'da Fransız Guyanası'ndan uzaya başarıyla fırlatılmıştır. 70 kilogram kütleye ve 0,50 metre çözünürlüğe sahip Göktürk-1, Güneş eş zamanlı görev yapıyor ve ömrü 7 yıl olarak öngörülüyor. Göktürk-1 uydusunun tasarım ömrünün 2025'te dolması bekleniyor. Bu yüzden Türkiye Göktürk-1 uydusunun yerini alacak Göktürk Yenileme Gözlem Uydusu Projesi'ni başlattı.

Şekil 9 İMECE Uydusu

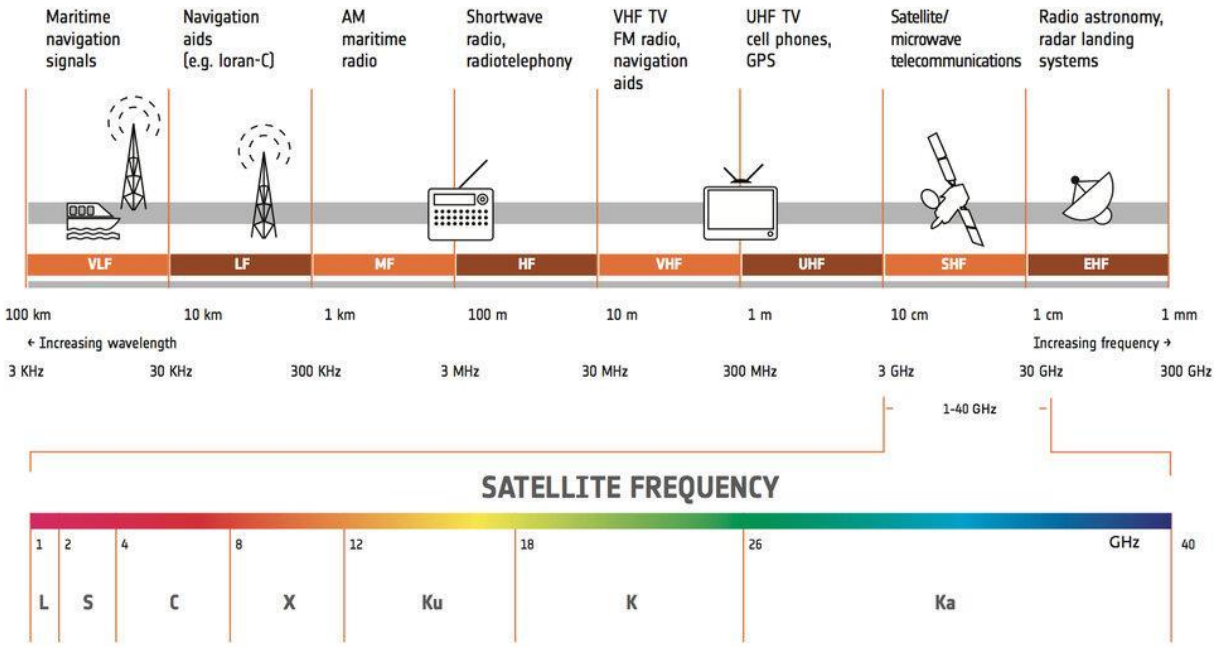
Metre Altı Çözünürlüklü Elektro Optik Uydu Kamerası, Haberleşme Sistemi, Yıldız İzler, Güneş Algılayıcı sistemleri yerli ve milli olarak geliştirilen İMECE uydusu da 15.04.2023 tarihinde uzaya fırlatıldı. Türkiye'nin yüksek çözünürlüklü uydu görüntüsü ihtiyacını karşılayan İMECE, 680 kilometre irtifada güneşe eş zamanlı yörüngede görev yapmaktadır. Coğrafi kısıt olmaksızın dünyanın her yerinden görüntü çekebilen uydu, hedef tespit ve teşhis, doğal afet, haritalama, tarımsal uygulamalar gibi birçok alanda Türkiye'ye hizmet veriyor.

Elektro-optik kamera dışında, elektrikli itki sistemi, güneş algılayıcı, yıldız izler, tepki tekeri, küresel konumlandırma sistemi alıcısı, manyetometre, X bant haberleşme ekipmanı ve anteni, S bant haberleşme ekipmanları ve antenleri, güç düzenleme ve dağıtım ekipmanları, uçuş bilgisayarı, uçuş yazılımları, yer istasyonu anteni, yer istasyonu yazılımları İMECE Projesi kapsamında yerli olarak geliştirildi. Yaklaşık 700

kilogram ağırlığındaki İMECE'nin boyutları 2 metre x 3,1 metre ölçüsünde. İMECE, 1000 kilometre uzunluğunda 16,73 kilometre genişliğinde bir alanı tek seferde çekebilirken, çektiği görüntüleri 320 megabayt/saniye brüt veri hızıyla yer istasyonuna indirebiliyor.

3. Uydu Haberleşmesi

Uydu iletişimleri, sinyalleri iletmek ve almak için 1-50 gigahertz (GHz; 1 gigahertz = 1.000.000.000 hertz) gibi çok yüksek frekans aralığını kullanır. Uydu haberleşme bantları Frekans aralıkları veya bantları harflerle tanımlanır: (düşük frekanstan yüksek frekansa doğru sırayla) L-, S-, C-, X-, Ku-, Ka- ve V-bantları. Uydu frekans spektrumunun alt aralığındaki (L-, S- ve C-bantları) sinyaller düşük güçle iletilir ve bu nedenle bu sinyalleri almak için daha büyük antenlere ihtiyaç vardır. Bu spektrumun üst ucundaki sinyaller (X-, Ku-, Ka- ve V-bantları) daha fazla güce sahiptir; bu nedenle, çapı 45 cm (18 inç) kadar küçük tabaklar bunları alabilir. Bu, Ku-bandı ve Ka-bandı spektrumunu doğrudan eve (DTH) yayın, geniş bant veri iletişimi ve mobil telefon ve veri uygulamaları için ideal hale getirir.



Şekil 10 Uydu Haberleşmesinde Kullanılan Frekanslar

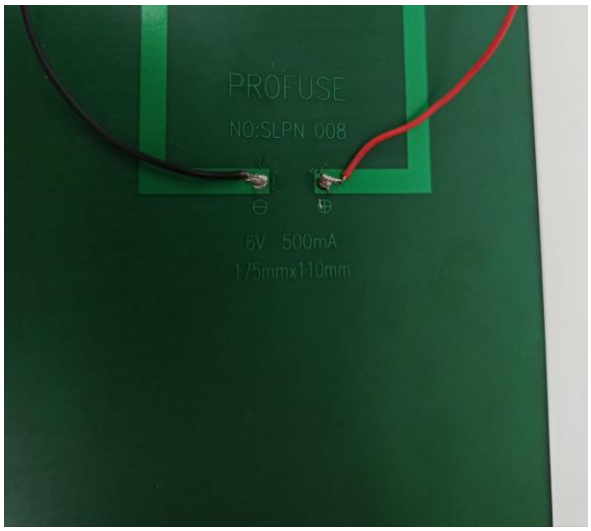
2. TASARLA

Bu bölümde öğrenciler bir gözlem uydusu modeli için gerekli parçaları listeleyerek bir gözlem uydusu tasarlamaya çalışır. Eğitmen bu sırada üretecekleri uydu modeli örneğini öğrencilerle paylaşmaz. Böylelikle öğrenciler gözle kısmında öğrendikleri bilgiler ışığında bir gözlem uydusu tasarlamaya çalışır. Eğitmen bu sırada öğrencilere uydunun enerji için neye ihtiyaç duyduğu, haberleşme için ne yapması gerektiği, gözlem için nasıl bir ekipmana ihtiyaç duyduğu gibi sorular sorarak onları yönlendirebilir.Eğitmen böyle bir uydunun Deneyap kart ile nasıl yapılabileceğini sorarak öğrencileri üretcekleri uydu modeline yönlendirebilir. Deneyap kart üzerine takılabilen kamera ve üzerinde bulunan Wifi alıcı-vericisi'ni hatırlatır. Böylelikle öğrencilerin kafasında üretecekleri model uydu için bir fikir oluşmasını sağlar.

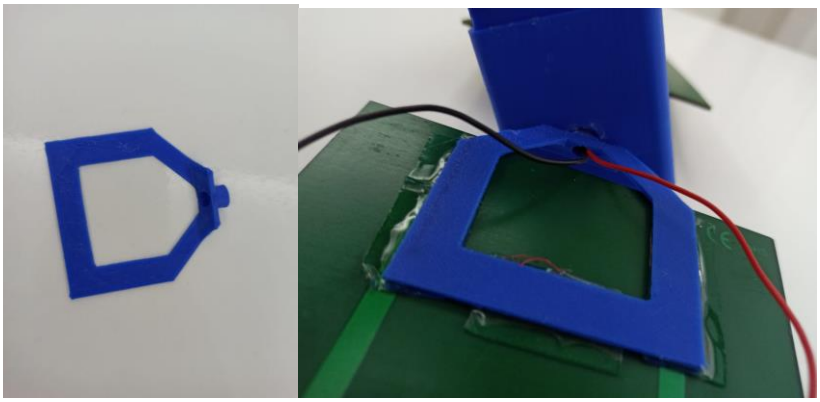
3. ÜRET

Bu aşamada eğitmen önceden hazırlanmış 3D basılı model uydu parçalarını sınıftaki öğrenci grupları ile paylaşır. Güneş paneli, Li-Po pil ve solar şarj modülünü de dağıtarak öğrencilerin bu parçalar yardımı ile Deneyap kartı kullanarak model bir gözlem uydusu yapmalarını ister.

Öncelikle öğrenciler güneş panellerine kablo lehimlemesi yapmalıdır.



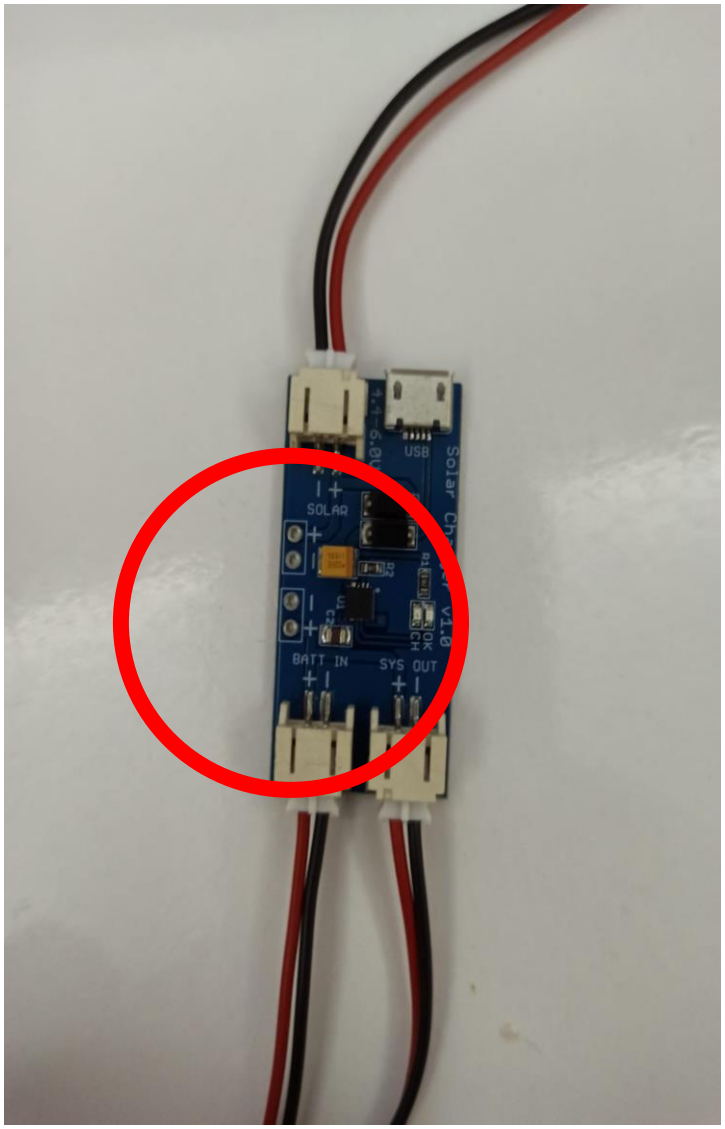
Üç boyutlu yazıcıda basılmış olan güneş paneli tutucuları yine 3 boyutlu yazıcıda basılmış olan uydu gövdesine şekildeki gibi yerleştirilerek yapıştırılmalıdır.



Güneş paneline lehimlenen kablolar tutucuların ortasındaki delikten geçirilerek uydu gövdesinin içerisinden alınır. İki kenardan gelen kırmızı ve siyah kablolar paralel bağlantı yapılarak lehimlenir. Paralel bağlanmış güneş paneli kablolarının ucuna JT2 konnektör ucu bağlanır. Ardından bu uç solar şarj kartının "solar" girişine yerleştirilir.

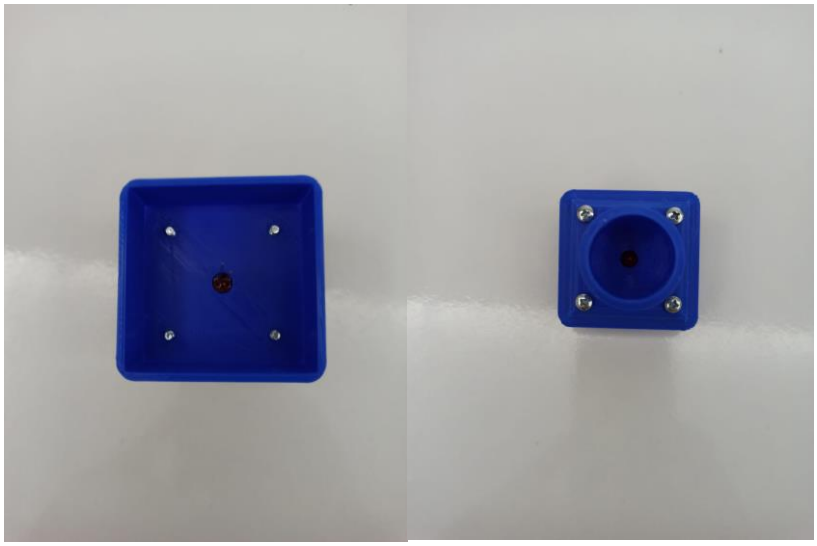
UYARI

JT2 konnektör yönleri Deneyap kart ile ters olabilir. Bu durumda konnektördeki kablo uçları (kırmızı ve siyah) yer değiştirilmelidir. Aksi takdirde Deneyap kart zarar görebilir.



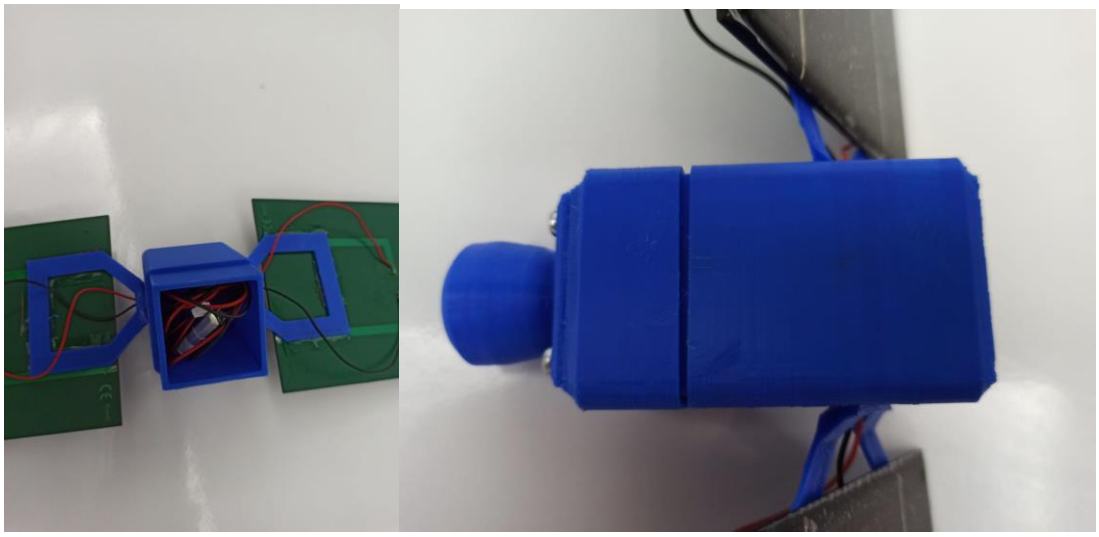
Li-Po pil batarya girişine deneyap kart da sys out girişine bağlanır. Böylelikle gerçek uydularda olduğu gibi artık Deneyap kart ve pilimiz güneş panelinden beslenmeye başlar.

Uydunun arka kapağına bir led yerleştirilir ve deneyap kartın GPIO çıkışlarından uygun bir tanesine bağlanır.

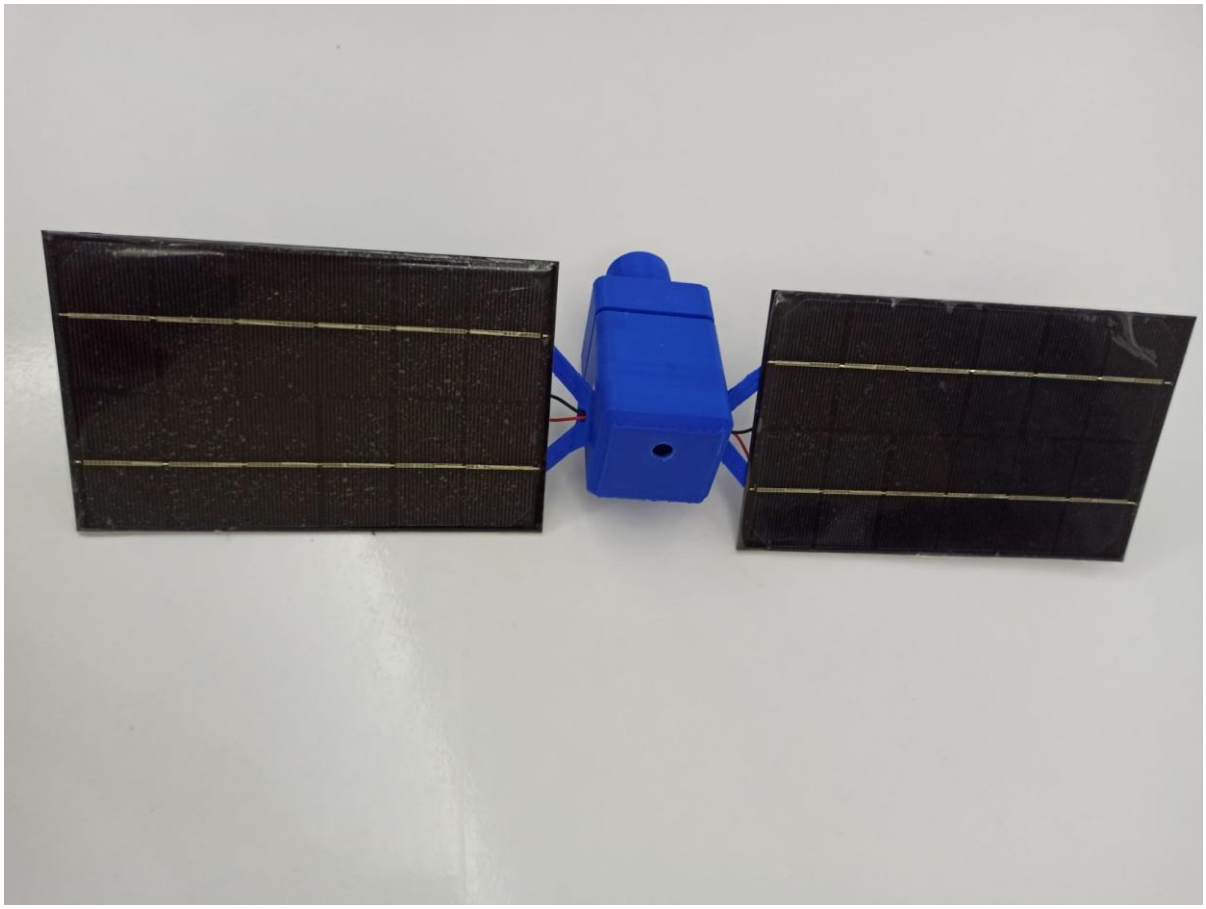


Arka ateşleyici parçası da kapağa vidalanır. Böylelikle LED yakıp söndürerek uydulardaki hareketi sağlaya ateşlemeler simüle edilmiş olur.

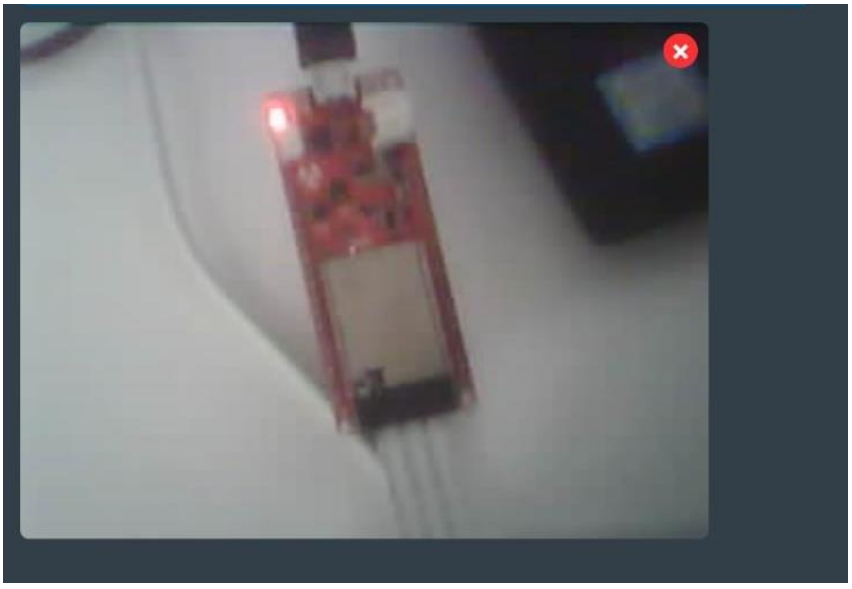
Deneyap kart kamerası uydu gövdesinin ön tarafına oturacak şekilde yerleştirilir. Ardından pil, solar panel ve deneyap kart uydunun içerisine yerleştirilerek arka kapak kapatılır.



Çalışır haldeki gözlem uydumuzun son hali aşağıdaki görseldeki gibi olur.



Bu aşamada bir grup öğrenci de deneyap karttan gelen görüntülerin alınıp gösterileceği yazılımı hazırlamalıdır. Öğrenciler kamera görüntüsünü bir web sayfasında izleyebilmelidir. Burada Deneyap kart bir WiFi bağlantı noktası gibi davranmaktadır (Kodda SSIDAP kısmına gruplar kendileri belirledikleri bir isim verebilirler). Bilgisayar bu bağlantı noktasına bağlanmalıdır, ardından 192.168.4.1 adresinden kamera görüntüsüne erişilebilir.



Bu aşamaya gelebilen öğrenciler bir sonraki aşamada uyduyu yörüngesinde hareket ettirme işlemini simüle etmek için uydu modelinin arka kısmına yerleştirilen bir LED'i web sayfasından komut vererek yakıp söndürmelidir. Bu sayede sanki uyduya yer istasyonundan itki sağlatılıyormuş gibi bir etki yaratılır.

Not

Arduino ide arayüzünde

Arayüz Türkçe ise: Araclar->Partition Scheme->Huge APP seçilmelidir.
Arayüz İngilizce ise: Tools->Partition Scheme->Huge APP seçilmelidir.

```
#include <WiFi.h>

const char* ssidAP     = "Deneyapkart";     // Wi-Fi ag adi
const char* passwordAP  = NULL;             // Wi-Fi ag sifresi

//IPAddress apip(192,168,1,1);
//IPAddress gateway(192,168,1,1);
//IPAddress subnet(255,255,255,0);

void setup() {
  Serial.begin(115200);   // Hata ayiklamak icin seri terminal baslatildi
  Serial.setDebugOutput(true);
  Serial.println();

  cameraInit();                     // Kamera konfigurasyonu yapildi

  WiFi.softAP(ssidAP, passwordAP);
//  WiFi.softAPConfig(apip, gateway, subnet);
  delay(100);

  startCameraServer();              // Kamera server baslatildi

  Serial.print("Kamera hazir! Baglanmak icin 'http://");
  Serial.print(WiFi.softAPIP());
  Serial.println("' adresini kullaniniz");
}

void loop() {
  delay(10000);
}
```

4. DEĞERLENDİR

Öğrenciler sınıf ortamında yaptıkları model uydu ile öğrendikleri uyduları kıyaslar. Benzerlik ve farklılıklarını değerlendirir. Bu uydu modelini gerçekten uzaya göndermek isteseydik nasıl bir yapıda olmalıydı ve de nasıl parçalardan oluşmalıydı yorumlar. Eğitmen bu yorumlamayı sözlü veya yazılı şekilde isteyebilir.

5. İLAVE ETKİNLİK

Sınıfta gerçekleştirilen uygulama zamanı artarsa aşağıdaki ileve etkinlikler gerçekleştirilebilir.

Öğrencilerden kamera verisi ile uydudan telemetri verilerini alacak kodu hazırlamaları istenebilir. IMU ve sıcaklık bilgisini Deneyap karttan alarak ekranda gösterecek kodu hazırlayabilirler. Atölye ortamındaki ışık vs. sensörler uydu üzerine ilave edilerek uydunun kapsamı arttırılabilir.

KAYNAKÇA

International Civil Aviation Authorization ANNEX I, Annex I 'Definitions for terms used in Annexes II-VIII

Uçuşa Başlangıç - Introduction To Flight. John D. Anderson, Jr. Çeviren Adil YükselenNobel Akademik Yayıncılık Eylül 2020

https://knilt.arcc.albany.edu/images/0/0e/Paper_activity2.jpg

UCK 111 Uçak Mühendisliğine Giriş ve Etik 2006-2007 Güz yarıyılı ders notları Prof. Dr. M. Adil Yükselen

Beygi, Nima et al. "Design of Fuzzy self-tuning PID controller for pitch control system of aircraft autopilot." *ArXiv* abs/1510.02588 (2015): n. pag.

TÜBİTAK Ulusararası İnsansız Hava Araçları Yarışması Eğitim Sunumları İtki. Dr. Öğr. Üyesi Rıfat Benveniste

Hareide, Odd Sveinung & Ostnes, Runar. (2017). Scan Pattern for the Maritime Navigator. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation. 11. 39-47. 10.12716/1001.11.01.03.

T.C. Milli Eğitim Bakanlığı Uçak Bakım, Uçak Elektriksel Güç Üniteleri, Ankara 2011

AIRBUS A320 Series Electrical System Training Manual 24.10.2015

https://www.aviationnepal.com/what-is-ram-air-turbine-rat/ Son erişim 12.01.2021

https://www.avionicinstruments.com/part-sub-family/static-inverter Son erişim 12.01.2021

https://www.interconnect-wiring.com/aerospace-wiring-products/ Son erişim 12.01.2021

PX4 Autopilot User Guide (v1.9.0) PX4 Dev Team. GITBook 28.10.2020, https://docs.px4.io/v1.9.0/en/

https://mgm.gov.tr/genel/meteorolojikuydular.aspx Son erişim 12.01.2022

https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks Son erişim 12.01.2022

https://uzay.tubitak.gov.tr/tr/uydu-uzay/rasat

https://www.esa.int/SPECIALS/Eduspace_EN/SEM7YN6SXIG_0.html

https://www.tusas.com/urunler/uzay/yer-gozlem-ve-kesif-uydulari/gokturk-1

https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands

https://www.gps.gov/multimedia/images/ Son erişim 01.04.2022

Beason, D. Aircraft Started with the Wright Materials. MRS Bulletin 20, 67 (1995). https://doi.org/10.1557/S0883769400044936

https://aipprecision.com/the-history-of-aviation-materials/

Yashpal, Jawalkar, C. S., & Kant, S. (2015). A review on use of aluminium alloys in aircraft components. I-Manager's Journal on Material Science, 3(3), 33-38.

Zhao, J.C., Westbrook, J.H. Ultrahigh-Temperature Materials for Jet Engines. MRS Bulletin 28, 622–630 (2003). https://doi.org/10.1557/mrs2003.189

Tuttle, M. E. (2004). Structural Analysis of Polymeric Composite Materials. Retrieved from http://books.google.com/books?hl=es&lr=&id=uvrMzybfbiAC&pgis=1

Joseph, L., & Cacace, J. (2018). Mastering ROS for Robotics Programming - Second Edition: Design, build, and simulate complex robots using the Robot Operating System. Packt Publishing. ISBN: 9781788478953.

Joseph, L., & Cacace, J. (2021). Mastering ROS for Robotics Programming - Third Edition: Best practices and troubleshooting solutions when working with ROS. Packt Publishing. ISBN: 1801071020.

Ramkumar, G., & Joseph, L., (2019). ROS Robotics Projects- Second Edition. Packt Publishing. ISBN: 1838649328.

Quigley, M., Gerkey, B., & Smart, W. D. (2015). Programming Robots with ROS: A Practical Introduction to the Robot Operating System (1st ed.). O'Reilly Media, Inc. ISBN: 9781449323899

Joseph, L., & Johny, A. (2022). Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy (2nd ed.). Apress. ISBN: 9781484277508

Abdunabi T. (2006) Modelling and Autonomous Flight Simulation of a Small Unmanned Aerial Vehicle. Sheffield, UK: The University of Sheffield, August 2006, 61 p

Koubaa, A. (2016) Robot Operating System (ROS): The Complete Reference (Volume 1). Springer; 1st ed. February 2016, ISBN: 9783319260525

Koubaa, A. (2017) Robot Operating System (ROS): The Complete Reference (Volume 2). Springer; 1st ed. June 2017, ISBN: 9783319549262

Koubaa, A. (2018) Robot Operating System (ROS): The Complete Reference (Volume 3). Springer; 1st ed. July 2018, ISBN: 3319915894

https://www.theconstructsim.com/

https://gazebosim.org/home

https://github.com/ros/ros_tutorials

https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition

https://github.com/ros/urdf_tutorial

http://wiki.ros.org/hector_quadrotor_urdf

https://github.com/noshluk2/ROS-Drone-Basic-Course-for-Beginners