IMU KALMAN FİLTRESİ UYGULAMASI

GÖKHAN GÖL

HEE 592 SEMİNER DERSİ

FİNAL RAPORU

HAVACILIK ELEKTRİK ELEKTRONİĞİ ANA BİLİM DALI

MAYIS,2015

1. Giriş

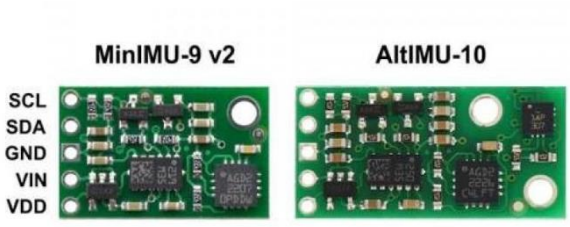
Cep telefonlarından, robotlara, hava araçlarından, stabilizasyon sistemlerine kadar hemen hemen her alanda kullanılan ataletsel ölçüm üniteleri (IMU-inertial measurement unit) birden fazla atalet sensörünü içinde barındıran sensör birimidir. Üzerinde bulunan gyroskop ve ivmeölçer sensörleri ile elde edilen veriler sayesinde, cisimlerin açısal hızlarını, yerçekimi yönlerini ve hareketten kaynaklanan ivmeleri ölçebilmekteyiz. Bu çalışmada 10 serbestlik derecesine sahip bir adet imu kullanılarak iki eksen kamera stabilizasyon sistemi yapılmıştır. Imu sensöründen alınan ve titreşimler dolayısı ile gürültülü sinyalleri barındıran verilere Kalman Filtresi uygulanarak kamera stabilizason sisteminin hassasiyeti artırılmıştır.

2. IMU

Imu atalet/eylemsizlik durumunu ölçemeye yarayan sensörleri tek bir kart içerisinde barındıran ve birçok alanda uygulama bulan analog ve dijital tipleri olan bir sensör birimidir. Aslında dünya referansına göre cisimlerin uzaydaki konumlarını belirlemek için kullanılırlar. Basit bir imu sensöründe açısal hızı ölçen gyroskop ve ivmelenmeyi ölçen ivmeölçer sensörü bulunmaktadır. Ancak üzerinde, irtifa kontrolü için kullanılan barometrik basınç sensörü ve yön tayini için pusula sensörlerini barındıran tipleri de mevcuttur.

Özellikle havacılık ve uzay sistemlerinde yaygın olarak kullanılan imu sensörleri, insansız hava, kara ve deniz araçlarında, füze sistemlerinde, otopilot sistemlerinde, kamera stabilizasyon sistemlerinde kullanılmaktadır.

Bu çalışmada kullanılan Pololu AltImu 10 sensör ünitesinde bulunan sensörler şunlardır (Şekil 1):



Şekil 1. AltImu 10 serbestlik dereceli imu

a) İvmeölçer :

Havacılıkta kullanılan akselerometreler 3 eksendeki ivmelenmeyi ölçmektedirler. Bunu üzerlerine düşen statik veya dinamik basıncı yada manyetik alanı ölçerek yaparlar. Statik basınç durma anındaki değerdir yani yerçekimidir. Sensörden alınan değerler m/s2 veya g kuvveti diye isimlendirilmektedir. Dolayısıyla durgun posizyonda sensörden okunacak 1g değeri 9.8 m/s2' ye yani yerçekimi kuvvetine denk gelmektedir. Genellikle statik basınç ölçümü yapan bu sensörlerden alınan değerler bizim için şu aşamada yeterlidir. Ancak uçaklarda bulunan ve ani hızlanmalarda acil durum sinyali yayan ELT (emergency locator trasmitter) içerisinde dinamik basıncı ölçen sensör mevcuttur. Birçok akıllı telefonda bulunan

ivmeölçer sensörleri statik basınç sayesinde, yerçekimi kuvveti etkisi altında kaldığından eğimi ölçerek telefonunuzun ekran oryantasyonunu değiştirmeyi sağlar. Aslında sensörü 3 boyutlu bir küp içerisindeki yuvarlak cisim olarak düşünebilirsiniz. Siz küpü hızlıca +x yönüne çektiğinizde yuvarlak cisim merkezkaç kuvveti etkisi ile –x yönüne basınç uygulayacaktır. Yani zıt yönde. Yada sabit durduğunu düşündüğümüzde yerçekimi etkisi ile küpün tabanına basınç uygulayacaktır. Dolayısıyla çalışma mantığı oldukça basittir.

Bu çalışmada kullanılan sensörden alınan ivme bilgileri ham verilerdir ve bu veriler işlemcinin ADC (analog digital converter) sayısına bağlı olarak belli değerler arasında sınırlandırılmışlardır.

Eğer kullanılan sensörün direk x,y,z pinlerinden arduinonun analog pinleri ile değer okunabiliyorsa, yani sensör analog bir sensör ise alınan analog değerler açı cinsinden ifade edilmelidir. Bu işlem şu şekilde yapılmaktadır:

Sensörün 3.3 V ile beslendiğini ve ADC (analog digital converter) nin 10 bit (yani $2^{10} - 1$ =1023) olduğunu varsayalım (arduino geliştirme kartlarında ADC 10 bittir).

*akselerometre_Değeri =(akselerometre_ADC_degeri *V_{ref}/1023-akselerometre_Zero_degeri) / Duyarlılık*

Bu formülde akselerometre_ADC_degeri okunan analog değerdir. V_{ref} sensörün besleme voltajıdır. Zero değeri ise sensörün datasheetinde yazan 0g anındaki akselerometrenin verdiği analog çıkış değeridir. ADXL335 sensörü için bu değer datashetten 1.5 V olarak okunmaktadır. Bu değerin analog değerde karşılığı şu şekilde bulunur. 3.3 V değerinde analog çıkış 1023 ise 1.5 V değerinde kaçtır diye oranlamak yeterli olacaktır. Dolayısıyla 1.5/3.3*1023 = 465 dir. Duyarlılık sensörün datashetinde verilen sensörün hassasiyet değeridir. Artık akselerometre degerini bilinenleri yerine koyarak bulabiliriz. Bunun 3 eksen içinde (x,y,z) yapılması gerekir. Sonrasında ise havacılık tabirindeki pitch, roll ve yaw eksenlerindeki açılar bulunacaktır. Bu da şu işlemlerle yapılır:

*Pitch_degeri = atan2(akselerometre_**Y**_degeri, akselerometre_**Z**_degeri)+PI*

*Roll_degeri = atan2(akselerometre_**X**_degeri, akselerometre_**Z**_degeri)+PI*

*Yaw_degeri = atan2(akselerometre_**X**_degeri, akselerometre_**Y**_degeri)+PI*

Bu formüllerde atan2 fonksiyonu, eksene göre açıyı hesaplar. Bu fonksiyondan -180 ile +180 arasında bir değer döner. PI değeri ise 180 dereceye denk gelmektedir.

Eğer kullanılan sensör ivmeölçer ve gyro değerlerini I2C protokolü (seri veri yolu haberleşme protokolü) ile alıyorsa yani imu ünitesine dahil edilmişse ve kütüphane kullanılıyorsa zaten veriler size bu hesaplamalar yapılıp açı şeklinde ilgili fonksiyonları kullandığınızda gelmektedir. Kütüphane kullanıldığında alınabilecek değerler fonksiyonlar ile birlikte şu şekildedir:

- *getAngles(angles) ;* Açı değerlerini verir (angles değişkeni *float angles[3];*) diye tanımlanmalıdır.
- *getRawValues(angles);* Sensörün ham verilerini gönderir. (angles değişkeni *int angles[5];*) diye tanımlanmalıdır. Sensörün 3 eksendeki gyro ve ivmeölçer verileri ayrı ayrı alınır.
- *getEuler(angles);* Sensörün euler açılarını gönderir. (angles değişkeni *float angles[3];*) diye tanımlanmalıdır
- *getAngles(angles);* Sensörün açı değerlerini gönderir. (angles değişkeni *float angles[3];*) diye tanımlanmalıdır. Euler açılarının düzenlenmiş halidir.
- *getQ(angles);* Sensörün quaternion açılarını gönderir. (angles değişkeni *float angles[3];*) diye tanımlanmalıdır.
- *getYawPitchRoll(angles);* Sensörün sırasıyla yaw, pitch, roll açılarını gönderir. (angles değişkeni *float angles[3];*) diye tanımlanmalıdır.

b) Gyroskop :

Gyroskop sensörü bir cismin bir eksen etrafında ne kadar hızda döndüğünü ölçmeye yarayan sensördür. Yani açısal hızı ölçen sensördür. Ölçülen değer derece/saniye yada rpm (dakikadaki devir sayısı) cinsinden ifade edilmektedir. Mekanik jiroskoplar topaç hareketine benzemektedir. Topaç hızla döndürüldüğünde aynı eksende hızlıca döndüğü müddetçe devrilmeden hareketine devam eder, ancak yavaşlaması durumunda yalpalayarak eksendeki dönme hareketi bozulur.

Bu projede analog çıkış veren 3 eksen gyroskop kullanılmıştır. Sensörün 3.3 V ile beslendiğini ve ADC (analog digital converter) nin 10 bit (yani $2^{10} - 1 = 1023$) olduğunu varsayalım (arduino geliştirme kartlarında ADC 10 bittir). Bu sensörden okunan analog değerler aşağıdaki işlemlerden geçirilerek eksenler arasındaki dönüş hızı ve dönüş açısı elde edilir:

$$XZdönüş\_hızı = (GyroADCxzDegeri * V_{ref}/1023 - V_{zeroRate})/Duyarlılık$$

$$YZdönüş\_hızı = (GyroADCyzDegeri * V_{ref}/1023 - V_{zeroRate})/Duyarlılık$$

$$ZXdönüş\_hızı = (GyroADCzxDegeri * V_{ref}/1023 - V_{zeroRate})/Duyarlılık$$

Bu formülde *GyroADCxzDegeri* gyroskop sensörünün X pininden okunan analog değerdir. V_{ref} sensörün besleme gerilimi yani 3.3 V' tur. $V_{zeroRate}$ sensörün datasheetinde yazan sensörün sabit konumdayken (yani hareketsiz konum) ürettiği analog değeri göstermektedir. *Duyarlılık* sensörün datasheetinde yazan sensörün hassasiyetidir.

Yukarıdaki formülasyon yapısı kullanılarak elde edilen açısal hızlar, iki örneklem arasındaki zaman farkı ile çarpılırsa dönüş açıları da bulunmuş olur.

c) Barometrik Basınç Sensörü:

Genellikle piezzo-resistif teknolojisine sahip olan bu sensörler irtifaya göre basınç değişimi kuralından yararlanarak irtifanın belirlenmesi için kullanılırlar. Yani basınç ölçümü ile irtifanın belirlenmesi amacıyla kullanılırlar. Basınç ile irtifa arasında ters orantı vardır ve her 30 metre irtifa artışında basınç 3.5 mb azalmaktadır.

d) Pusula :

Pusula yada magnetometre diye isimlendirilen bu sensör manyetik kuzeye göre yön tayini için kullanılmaktadır. Özellikle hava araçlarında sapma (yaw) açısı hatalarının belirlenmesi ve düzeltilmesi için ihtiyaç duyulan pusula sensörü otomatik kara ve deniz araçlarında, navigasyon sistemlerinde ve hedef takip sistemlerinde yaygın bir şekilde kullanılmaktadır.

Yukarıda anlatılan 4 farklı sensörün tek bir kart üzerinde toplanması bir bir eksikliklerini tamamlamaları içindir.

Kamera stabilizasyonu için gimbal tasarımı ve kontrolü ile ilgili bu çalışmada bu sensörlerden sadece ivmeölçer ve gyroskop sensörlerinin verileri okunup filtrelenecektir. Aslında elde edilen ivme bilgisinin zamana göre integrali açısal hız bilgisini vermektedir. Yada elde edilen açısal hız bilgisinin zamana göre türevi ivmeyi vermektedir. Yani tek bir sensörle bu değerler elde edilirken bu iki sensörün aynı anda kullanılma amacı olumsuz yönlerinin birlikte kullanılarak ortadan kaldırılmasıdır.

Özellikle bu iki bilgiden elde edilecek pozisyon bilgisinin çok önemli olduğu hava araçlarında akselerometre yanında jiroskop ve pusula sensörüne ihtiyaç duyulmaktadır. Çünkü bize sadece ivmelenme bilgisi yetmez, bunun yanında açısal hız ve yön tayin bilgisi de gerekir. Aslında bir diğer ve önemli sebepte, ivmeölçer sensörünün kuvvete karşı çok duyarlı olması ve bunun neticesinde en ufak titreşimlerde yüksek gürültü sinyallerinin oluşmasıdır. Bunun yanı sıra ivmeölçer sensöründe kayma (drfit) çok fazla gözükmez. Ancak jiroskop sensöründe kayma çok fazla gözükürken hassasiyette bir o kadar yüksektir. Ayrıca gyroskop üzerinde

oluşan kaymalar pusula yardımı ile de kompanse edilebilmektedir. Dolayısıyla birbirlerini tamamlayarak İMU denilen yapı içerisinde toplanmışlardır.

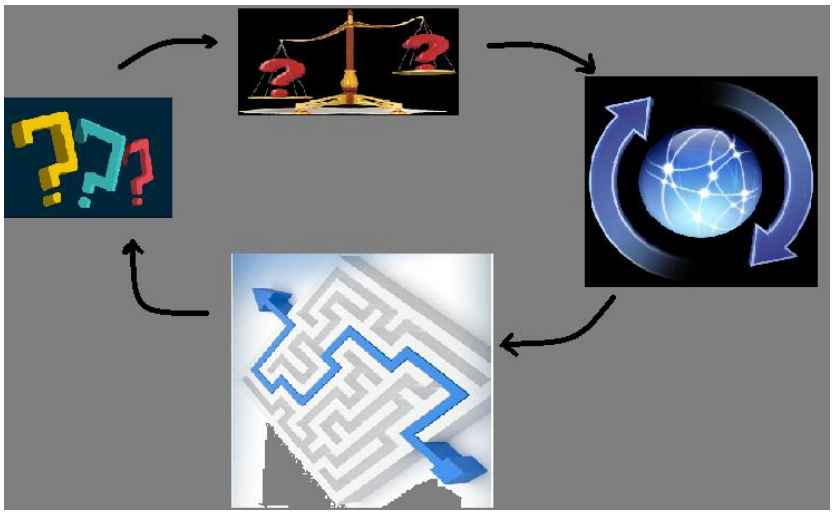
3. Kalman Filtresi

Kalman Filtresi 1950' li yıllarda matematiksel sistem teoristi olan Rudolf Emil Kalman tarafından bulunmuş ancak 1960'lı yıllara kadar duyulan şüpheler dolayısı ile kullanımı yaygınlaşamamıştır[1]. Ancak 1960 yılında ilk kez araç navigasyon sisteminde ve Apollo programında kullanıldıktan sonra çok fazla yaygınlaşmış ve özellikle havacılık, uzay ve kontrol mühendisliği alanlarında çok önemli bir gelişme sağlamıştır.

Kalman filtresi, yinelemeli olarak bir yandan bir durumun sürecini kestirirken (yani tahmin ederken), diğer yandan yapılan hatayı minimize eden yani sürekli gerçek değere ulaşmayı hedefleyen bir formülasyondur. Biraz daha açık söylemek gerekirse sistemin önceki bilgileri ile birlikte giriş ve çıkış bilgilerini de kullanarak bir sonraki anın durum tahminini yapan matematiksel ifadelerden oluşan bir filtredir. Matematiksel olarak çok karmaşık bir yapıya sahip ve uygulamada zorlanılan bu filtrenin yaptığı işi hayvanlar ve insanlar kendiliğinden yapmaktadırlar. Bunu şu örneklerle açıklayabilirim: bir çita bir antilopu avlamaya çalışırken, antilopun nasıl hareket edeceğini, dönüşlerinin nasıl olacağını tahmin etmektedir ve ona göre kendi hareketini kontrol etmektedir. Yani antilop dönüşünü yapmadan bir önceki hareketi ile çitaya ipucunu vermiştir. Yada sivri sineği avlamaya çalıştığınızda, tam vuracakken sineğin ortadan kaybolması buna bir örnektir.

Kalman filtresi özellikle sensör füzyonu ve veri füzyonu için kullanılır[2]. Bu çalışmada da kamera stabilizasyon sistemi için ivmeölçer ve gyroskop sensörlerinden oluşan bir sensör füzyonu kullanılacaktır. Çoğu zaman, gerçek zamanlı sistemler bir sistemin durumunu elde etmek için tek bir ölçüm yapmak yerine yada tek bir sensör kullanmak yerine bir çok ardışık ölçüm üreterek ve aynı yada farklı birçok sensör kullanır. Bu farklı yada aynı sensörlerden alınan veriler matematiksel işlemlerle birleştirilerek ölçümlerin gerçek değere yakınsaması sağlanır. Örneğin bir oda sıcaklığının tespit edilmesi probleminde, odanın içerisine yerleştirilen 10 sıcaklık sensöründen okunan değerler muhtemelen farklı olacaktır. Bu durumda tüm bu sensör değerleri bazı matematiksel işlemlerden geçirilerek ölçümün, gerçek sıcaklık değerine yaklaşması amaçlanır.

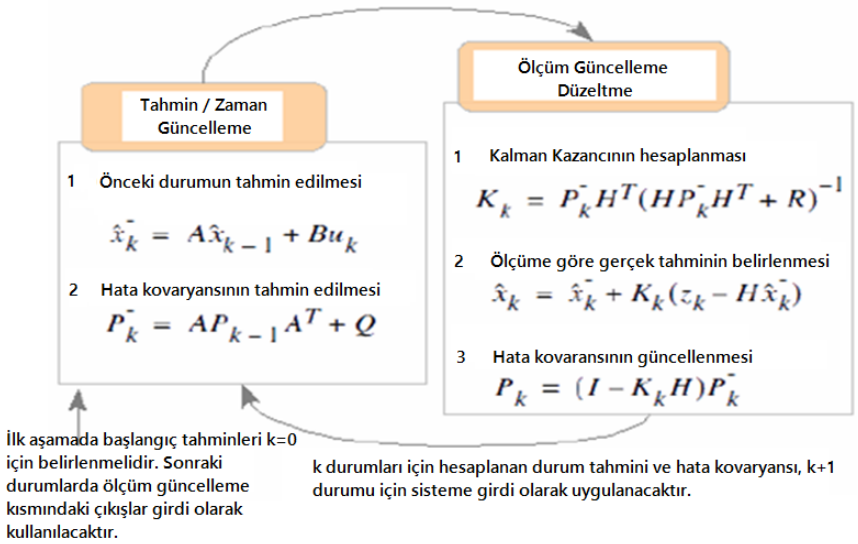
Kalman filtresinin temel mantığı, tahmin eder, karşılaştırır, günceller ve daha iyi tahmin eder. Yani sürekli tahmini iyileştirme yönünde çalışan kuvvetli bir filtredir (Şekil 2).



Şekil 2. Kalman Filtresi çalışma mantığı

Çevremizdeki birçok sistem dinamiktir ve bu sistemleri modellerken bu sistemlere etkiyen tüm faktörler ölçülemez yada ölçülenlerin doğruluğundan emin olunamaz. Dolayısıyla modellenmek istenen bu sistemlere ait yetersiz bilgilerden tüm sistem hakkında çıkarım yapabilmek için kalman filtresi birebirdir. Yani kalman filtresi eldeki yetersiz bilgilerden tahminler oluşturan ve tahminleri sürekli istenen/hedeflenen değerlere yaklaşan kuvvetli bir filtredir.

Kalman filtresi iki evreden oluşan bir yapıya sahiptir. Bunlardan ilki tahmin/zaman güncelleme diğeri ölçüm güncelleme yada düzeltmedir (Şekil 3). Tahmin yapılacak veriler bu iki yapı arasında sürekli bir döngü içerisinde çalışırlar ve sürekli tahminin gerçek değere yakınsamasını sağlarlar.



Şekil 3. Kalman Filtresi çalışma şeması

Tahmin ölçüm güncelleme denklemlerindeki bilinmeyenler:

k: durumları gösterir.

X_k^-: k durumundaki önceki durum tahmini

X_{k-1}: k-1 durumundaki tahmin

U_k : sisteme dışarıdan uygulanacak kontrol sinyalini simgelemektedir.

A,B,H : matrislerin genel gösterimidir. Aslında durum matrisleridir ve sistemin transfer fonksiyonundan elde edilmektedirler. Çoğunlukla işlem kolaylığı açısından bu matrisler sabit sayı alınmaktadır. Hatta çoğunlukla 1 gibi sabit bir değer alabilir.

P_k^- : hata kovaryansının önceki durumu.

P_k : hata kovaryansı (hatanın ortak değişme miktarı)

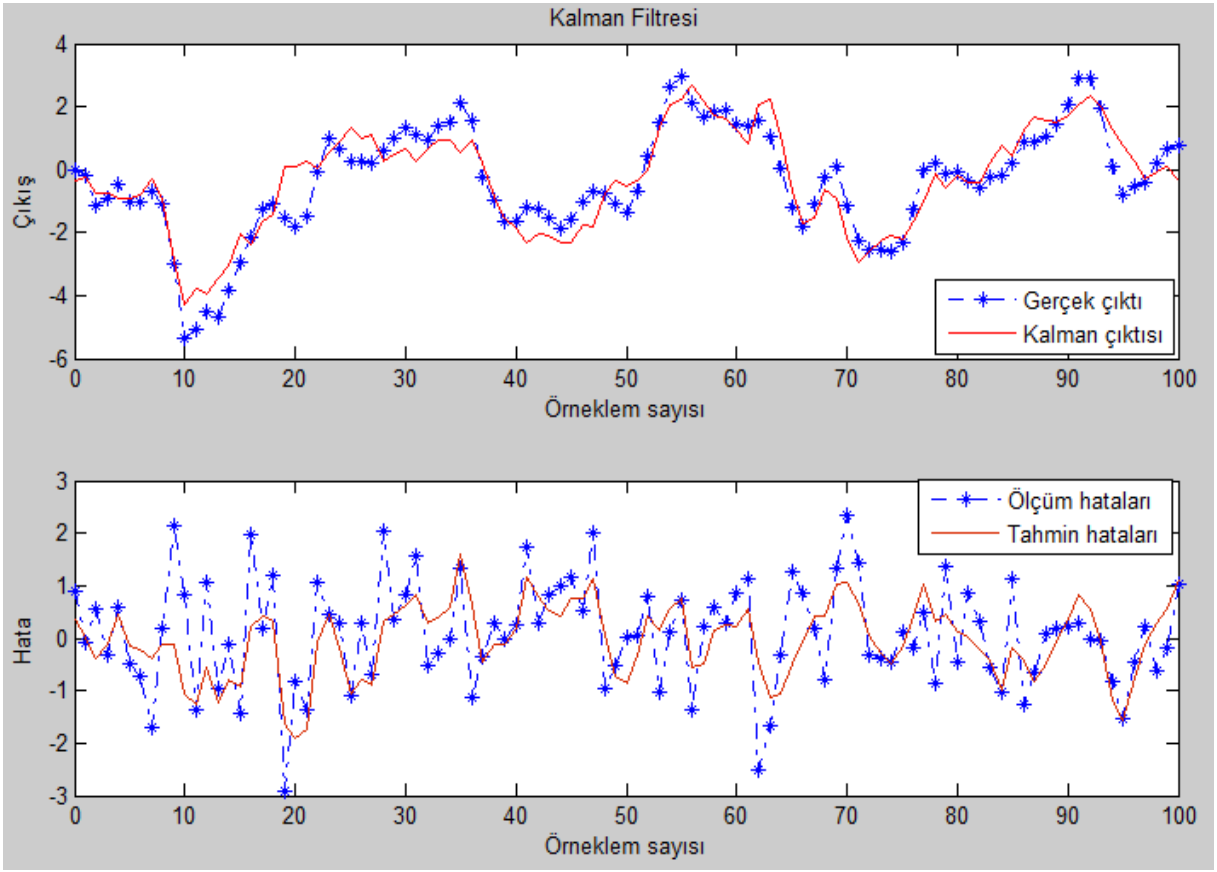
K_K : kalman kazancı

Z_k : k durumundaki ölçüm değeri

Denklemler arasındaki R ve Q parametreleri, gürültü kovaryans matrisleridir. R ölçüm gürültüsünü temsil etmektedir ve küçük olması ölçümlerin doğruluğunu artırmaktadır, Q ise işlem/süreç gürültüsüdür. Bu gürültüler gerçekte dinamik ve non-lineardir. Ancak işlem kolaylığı ve kalman filtresine uygulayabilmek için bu gürültü değerleri gaussian dağılıma uydurularak doğrusallaştırılır. Normalde bu gürültüleri modellemek oldukça zordur bu yüzden genellikle değerler eşit yada birbirlerini 1'e tamamlayacak şekilde alınır.

Kalman filtresi uygulanan sistemde model tahmini, gözlem ile karşılaştırılır ve bu fark, kalman kazancı olarak bilinen bir çarpan ile ölçeklendirilir ki bu daha sonra sıradaki tahminleri iyileştirmek için modele bir girdi olarak geri beslenir. Böylece yöntem, gerçek bilinmeyen değerlere, tek bir ölçüme veya sadece model tahminlerine dayanarak elde edilebilecek tahminlerden daha yakın tahminler üretmeye yakınsar.

Kalman filtresinin daha iyi anlaşılması için matlab programında üretilen bir sinyale gürültü eklenmiştir. Eklenen gürültü sinyalleri, gerçek sinyal ve kalman tahmin sinyalinin çıktısı Şekil 4' de gözükmektedir. Eklenen gürültü sinyaline rağmen kalman çıktısının gerçek değere ne kadar yakınsadığı gösterilmiştir. Ayrıca ölçüm hataları ve tahmin hataları arasındaki fark oldukça göze çarpmaktadır.



Şekil 4. Kalman Filtresi uygulanan matlab çıktısı

3.1.Kalman Filtresi' nin Matematiksel Açıklaması

a) Zaman Güncelleme / Tahmin

$$X_k^- = AX_{k-1} + BU_k \quad (1)$$

1 numaralı formüldeki ifadelerin tanımlamaları yukarıda yapılmıştı. Bu eşitlik ile ölçüm güncelleme aşamasında kullanılacak durumun önceki tahmin değeri bulunmaktadır. Kalman filtresinin ilk aşaması olan bu işlem de durum matrisleri (A, B) 1 olarak alınacak, kontrol sinyalinin olmadığı varsayılacaktır. Eğer sistemin durum matrisleri ve kontrol sinyali mevcut ise işleme katılmalıdır. İfadeden de anlaşıldığı üzere herhangi bir durum tahmin değeri, bir önceki durumun tahmin değeri ile kontrol sinyalinin lineer bir kombinasyonudur.

Başlangıç koşulları bilinmiyorsa k=1 için X_{k-1} yani $X_0 = 0$ olarak alınabilir. Başlangıç durum tahmininin 0 alınması sistemin çalışmasını engellemeyecektir. Ancak başlangıç koşullarında sistemin durumu biliniyorsa bu değerleri kullanmak, tahminlerin gerçek değerlere daha hızlı yakınsamasını sağlayacaktır.

Dolayısıyla yapılan değerlendirmeler sonucunda $X_k^- = AX_{k-1} = 0$ eşitliği karşımıza çıkmaktadır.

$$P_k^- = AP_{k-1}A^T + Q \quad (2)$$

2 numaralı formülde tahminin hata kovaryansı yani hataların birlikte ne kadar değiştiğinin ölçüsü, ölçüm güncelleme aşamasında kullanılmak üzere bulunmaktadır. Eğer hata kovaryansının başlangıç koşulu (k = 1 için $P_{k-1} = P_0$) bilinmiyorsa sıfırdan farklı bir değer alınabilir. Genellikle 1 olarak alınmaktadır.

b) Ölçüm Güncelleme / Düzeltme

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (3)$$

Ölçüm güncelleme kısmında bulunması gereken en önemli faktör kalman kazancıdır. Genellikle bu kazanç değeri performansın iyileştirilmesi için değiştirilebilir yapılmaktadır. Kalman kazancı, bize ölçüme göre durum tahmininin ne kadar değişmesi gerektiğini söyler. Yani bir çarpan olarak durum tahminine eklenir. Burada kalman kazancı üzerinde en fazla etkiye sahip değişken hata kovaryansıdır (P_k). Eğer hata kovaryansı büyük bir değer ise durum tahminleri çok fazla değişkenlik gösterir. Bu yüzden yeni ölçümlerle tahmin güncellemesi yapılmalıdır. Aynı zamanda hata kovaryansının artması kalman kazancının da artmasına sebep olacaktır. Eğer hata kovaryansı küçük ise, durum tahminin çok fazla değişmediği gözlenir. Bu yüzden tahmin değerini sürekli olarak değiştirmek gereksizdir. Hata kovaryansının küçük olması kalman kazancının azalmasına sebep olur.

Kalman kazancını etkileyen bir diğer faktör ölçüm gürültüsüdür (R). Ölçüm gürültüsünün artması ölçüm değerlerinin doğru olmadığını gösterirken kalman kazancının da azalmasına sebep olur. Böyece model tahminleri daha yakından takip edilebilmektedir.

$$X_k = X_k^- + K_k(Z_k - HX_k^-) \quad (4)$$

4 numaralı formül kalman filtresinin amacı olan durum tahminin elde edildiği kısımdır. Burada 3. formülde elde edilen kalman kazancı (K_k) ve k durumundaki ölçüm değeri kullanılır. Bulunan durum tahmini zaman güncelleme kısmında girdi olarak kullanılacaktır.

$$P_k = (1 - K_k H)P_k^- \quad (5)$$

5 numaralı formül ile zaman güncelleme kısmında kullanılacak olan güncel hata kovaryansı bulunmaktadır.

4. Kamera Stabilizasyon Sistemi (Gimbal)

Gimbal, bir nesnenin tek eksende dönüşünü sağlayan tek pivotlu sistemdir. Yani bir nesnenin monte edildiği yerin hareketine rağmen sabit kalmasını sağlayan servo yada dc motor kullanılan ve kontrol ünitesi bulunan bütünleşik yapıdır (Şekil 5).

Şekil 5. 2 eksen gimbal

Gimbal sistemleri 2 yada 3 eksen olarak imal edilirler. Özellikle hava araçlarında kullanılan bu sistemler hareket halindeki araçlar üzerinde titreşimsiz görüntü elde edilmesi amacıyla kullanılırlar. Ancak bu sistemler araç amortisör sistemlerinde de yaygın olarak kullanılmaktadır.

Doğadaki en iyi gimbal örneği tavukların boyunlarıdır. Bir tavuğu gövdesinden tutup sağa / sola, yukarı / aşağı hareket ettirdiğinizde kafasının sabit kaldığını gözlemlersiniz. Aslında bu doğanın kusursuz gimbalına bir örnektir.

4.1.Gimbal Yapımı ve Kontrolü

Bu çalışmada 2 adet Futaba S14B model dc servo motor, 1 adet arduino mega, 1 adet Pololu AltImu 10 imu sensörü kullanılarak iki eksende kontrol edilebilen bir gimbal uygulaması yapılmıştır. Ancak çalışmada istenirse, arduino geliştirme kartlarının diğer modelleri de kullanılabilir.

Arduino, C++ tabanlı Processing/Wiring dili ile uygulamaların oluşturulduğu fiziksel programlama platformudur. Arm mimariye sahip atmel işlemcilerin kullanıldığı bu geliştirme

kartları üzerinde, haberleşme için tanımlı seri port, i2c protokol destekli çıkış ve girişler, dijital ve analog giriş/çıkış pinleri ve harici kesme pinleri mevcuttur.

AltImu 10 imu sensörü üzerinde 3 eksen gyroskop (LG3D20), 3 eksen ivmeölçer (LSM303DLHC), 3 eksen pusula (LSM303DLHC) ve 1 eksen barometrik basınç sensörü (LPS331AP) bulunmaktadır. Yani toplamda10 serbestlik derecesine sahip bu imu sensörü sayesinde, hız, pozisyon, yönelim, yükseklik ve yön tayini bilgileri elde edilebilmektedir. Ancak bu çalışmada sadece gyroskop ve ivmeölçer bilgileri kullanılmıştır. Bu imu'nun kullanılma amacı ileriki çalışmalar için alt yapının oluşturulmasıdır.

Çalışma kapsamında dizayn edilen basit gimbal sistemi Şekil 6' da gösterilmektedir.

Şekil 6. 2 eksen basit gimbal düzeneği

Gimbal kontrolü için kullanılan kontrol ünitesi arduino mega diye isimlendirilen geliştirme kartıdır. İstenirse arduino geliştirme kartlarının farklı versiyonları kullanılabilir. İmu sensöründen arduino geliştirme kartına bilgiler I^2C data hattı olan SDA (serial data) ve I^2C clock hattı olan SCL (serial clock) pinleri kullanılarak alınır.

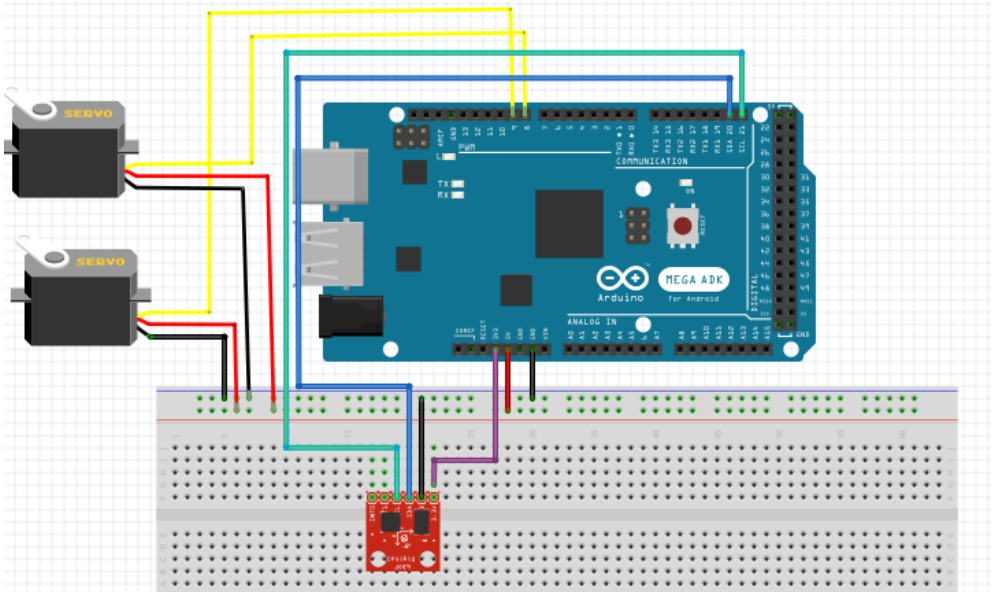
I^2C haberleşme protokolü: En basiti ile söylemek gerekirse, SDA (serial data line) ve SCL (serial clock line) pinleri kullanılarak yapılan düşük bant genişliğine sahip haberleşme protokolüdür. Aslında bir seri veri yoludur. Bu protokolde de veri gönderimi bir start biti ile başlar ve stop biti ile biter. Veri alışverişi clock (SCL) bitinin lojik 0 olduğu durumlarda gerçekleşir. Start ve stop durum bitleri ise clock bitinin lojik 1 olduğu durumlarda gönderilir. Dolayısıyla verinin gönderilmesi yada alınması SDA hattı üzerinden gerçekleşir. Genellikle mikroişlemci tarafı master diğer taraf (ki bu duruma göre mikro işlmeci tarafıda olabilir) slave diye isimlendirilir. Bu protokol iki nokta arası haberleşme dışında multi point

haberleşme imkanı da sunmaktadır[3]. Tam olarak veri alışverişi şu şekilde gerçekleşmektedir:

- Mikrodenetleyici veri yolu üzerinden (SDA hattı) start bitini gönderir (SCL hattı lojik 1 iken, lojik 1'den lojik 0'a geçiş start biti, tam terside stop biti anlamına gelir). Start biti gönderildiğinde slave modüller adres bilgilerini hazırlar.
- Mikrodenetleyici tarafından haberleşmek istenilen slave modülün adresi ve veri alınacak mı yoksa gönderilecek mi bilgisi *(read/write) 1 bayt şeklinde veri yolu (SDA) üzerinden gönderilir. Slave olarak kullanılan cihazlar SDA üzerinden gelen adresin kendilerine ait olup olmadıklarını kontrol ederler.
- Adresin kendine ait olduğunu doğrulayan slave modülü, haberleşmeye hazırım mesajını iletmek için veri yolu üzerinden ACK (kabul) sinyali gönderir. Bu arada adresi uymayan diğer slave modüller stop bitine kadar veri yoluna bilgi basmazlar. Böylece veri alışverişi gerçekleşmiş olur. Stop biti master tarafından gönderildiği an ise bitmiş olur.

*read/write: işlem yazma masterdan slave'e doğru iken işlem okuma tersi yöndedir.

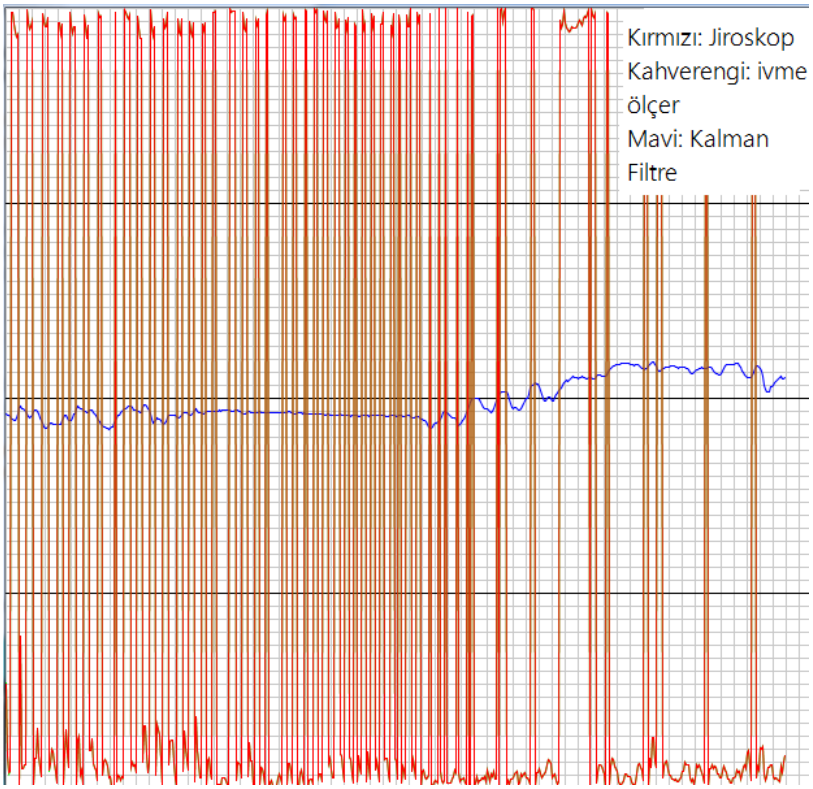
İmu sensörü, servolar ve arduino mega bağlantı şeması Şekil 7' de gösterilmiştir:



Şekil 7. Uygulama bağlantı şeması

Gimbal kontrol kodu Arduino Ide arayüzü kullanılarak arduino programlama dili ile yazılmıştır. Kalman filtresi imu sensöründen okunan değerlere, C dili ile yazılmış kütüphane vasıtasıyla uygulanarak servoların kontrolü sağlanmıştır.

Şekil 8’ de imu sensöründen alınan ham datalar ve kalman filtresinin uygulandığı çıkış gösterilmiştir.



Şekil 8. Imu verileri ve kalman çıktısı

5. Kaynaklar

[1] Wikipedia, http://tr.wikipedia.org/wiki/Kalman_Filtresi

[2] Çayıroğlu ,İ.“*Kalman Filtresi ve Programlama*” , Fen ve Teknoloji Bilgi Paylaşımı Sayı: 2012/1

[3] http://www.gokhangol.com/arm/tiva-c-series-project-1/