TARİH: 13 / 05/ 2021

ERCİYES ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERSİN ADI: SİSTEM ANALİZİ

GÖRÜNTÜ İŞLEME İLE GERÇEK ZAMANLI NESNE TAKİBİ

ÖĞRETİM ÜYESİ: Doç.Dr. METE ÇELİK

HAZIRLAYANLAR:
1030516734 GÜRBÜZ KAAN AKKAYA
1030516774 ÖMER CAN VURAL
1030516777 BÜŞRA KAYA
1030522911 ALİ NEFVEL ARAS

İÇİNDEKİLER

BÖLÜM 1. ..... **3**
1.1. Problemin Tanımı 3
1.2. Projenin Tanımı 5
1.3. Projenin Amacı 6
1.4. Projenin Gerçekleştirilebilirliği 7
1.5. Projenin Problemin Çözümünde Kullanımı 8

BÖLÜM 2. ..... **8**
2.1. Projenin Organizasyonu 8
2.2. İşlevsel Olmayan Gereksinimler 9
2.3. İşlevsel Gereksinimler 10
2.4. Yöntem Hakkında Genel Bilgi 11
2.5. Uygulamada Kullanılan Araç ve Gereçler 18

BÖLÜM 3. ..... **33**
3.1. Uygulamanın Gerçekleştirilme Aşamaları 33

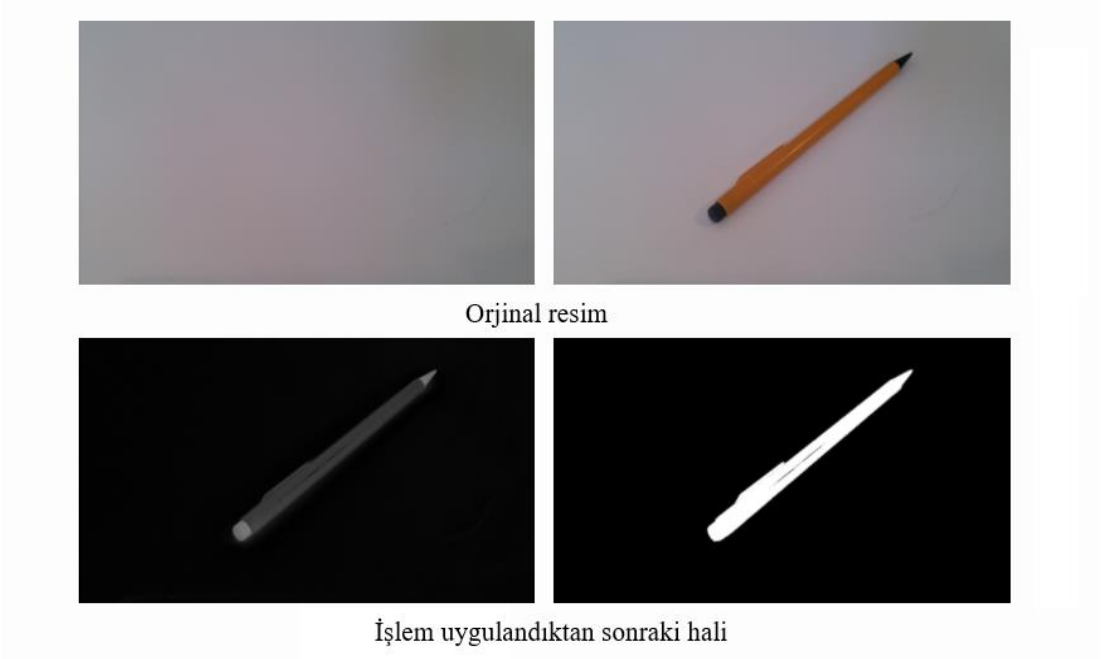
SİMGE VE KISALTMALAR LİSTESİ

İHA	: İnsansız Hava Aracı
MHT	: Çoklu Hipotez Takibi
LBP	: Local Binary Pattern
CSRT	: Kanal ve Mekansal Güvenilirlik Takipçisi
HOG	: Yönelimli Gradyanların Histogramı
KCF	: Çekirdekleştirilmiş İlinti Süzgeci
MIL	: Çoklu Örnek Öğrenme
TLD	: Takip, Öğrenme ve Tespit
NCC	: Düzgelenmiş Çapraz İlinti
OpenCV	: Açık Kaynak Kodlu Görüntü İşleme Kütüphanesi
FPS	: Saniyedeki Kare

1.1.Problemin Tanımı

Nesne takibi yapan çalışmalar 4 farklı aşamadan geçmektedir. Bu aşamalar "Video Ön İşleme">"Nesne Tespiti">"Nesne Sınıflandırma">"Nesne Takibi" şeklinde incelenebilir. Görüntü işlemede, işleme başarısını arttırmak ve işlemci yükünü azaltmak için görüntü önişlemden geçer. Bu işlemi gerçekleştirecek yöntemler, görüntüdeki gürültüyü yok etme, nesne kesimleme, renk temelli filtreleme gibi işlemleri içerir.

Bir videodaki veya görüntüdeki nesnenin takibi tespiti için 2 temel bilgi gerekir. Bunlar; nesnenin rengi, dokusu, şekli gibi özellikleri olan nesnenin özniteliği ve hareket bilgisidir.



Orjinal resim

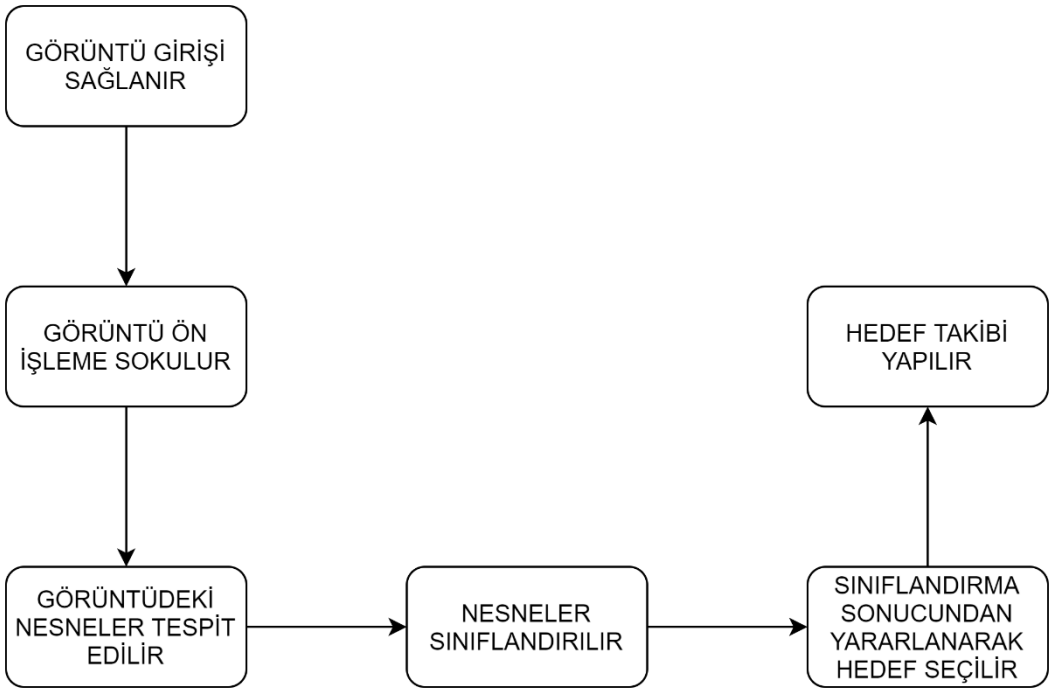
İşlem uygulandıktan sonraki hali

Nesne tespiti yapıldıktan sonra elde kalan nesnenin sınıflandırılması gerekir. Sınıflandırma işlemini cismin özniteliği bakarak yapılır. Nesne takibi için kullanılan en yaygın öznitelikler renk, kenar, doku, derinlik, süper piksel, hareket ve optik akıştır.

Görüntü işleme ile nesne tespiti ve takibi yapmanın 2 yolu vardır: sabit kamera ile, hareketli kamera ile. Sabit kameralı sistemlerde nesne tespitindeki problemler genellikle ışıktaki değişimler, kameradaki titreşimler, gürültüler ve ağaç yapraklı gibi hızlı değişen artalana ait nesnelerden kaynaklanır. Hareketli kamerada ise

sabit kameradaki problemlere ek olarak kameranın hareketi sebebiyle artalanın kayması problemi ile karşılaşılır.

Nesne tespit ve nesne tanıma uzun yıllardır üzerinden çalışılan konulardır. Nesne tespiti ve nesne tanıma için farklı algoritmalar geliştirilmiştir. Son yıllarda grafik işleme birimlerindeki gelişmeler ve derin öğrenme algoritmaları sayesinde daha fazla doğruluk oranına sahip yöntemler geliştirildi. Nesne takip genel olarak dört farklı aşamada ele alınabilir. Bunlar ön işlem, nesne tespiti, nesne sınıflandırması ve nesne takibidir. Bu dört aşamanın her biri bir sonraki aşamanın başarı oranını etkilemektedir. Günümüzde nesne tespit ve takibi uygulamalarını askeri alanda iha, casus uçak, uydu; sosyal hayatta akıllı telefonların yüz tanıması, araç plaka tanıma sistemleri gibi birçok kullanım alanına sahiptir.



Nesne takip uygulamalarının genel blok diyagramı

1.2.Projenin Tanımı

Görüntü işleme dijital bir görüntü içindeki önemli bilgilerin çıkartılması ve işlenmesidir. Görüntülerde bulunan nesneleri sırasıyla tespiti, tanımlanması, sınıflandırılması ve takibi işlemlerini yapacak birçok yöntem geliştirilmiştir. Her bir basamaktaki başarı oranı bir sonraki basamaktaki başarı oranını etkilemektedir. Takip edilecek nesnenin değişken bir alan içinde bulunması bu yöntemleri zorlaştıran temel problemdir. Bu problemi çözmek ve nesnenin başarılı bir şekilde takibini yapabilmek için birçok farklı yöntem geliştirilmiştir. Bu çalışmada kameradan alınan gerçek zamanlı görüntüyü işlemek için Python ve OpenCV kullanılmıştır. Nesnenin ya da kameranın hareket etmesi durumunda hedef nesnenin görüntü çerçevesinden çıkmasını engellemek için Arduino ile kameraya iki eksende bağlı motorları sürerek hedef seçilen nesnenin takibin yapılmıştır. Uygulama sonucunda hedef seçme ve hedefin takip işlemi başarılı olmuştur. Yedi algoritmanında kullanım alanları farklı olduğu için birbirine göre başarı oranları daha farklıdır. Boosting daha yavaş ve zaman zaman hedefi kaybetmektedir. MIL, Boosting'e göre daha iyi doğruluk oranına sahip. KCF, ilk iki algoritmaya göre daha iyi fakat tıkanma durumunda nesneyi takip edemiyor. Median Flow hızlı nesneleri takip ederken bazen nesneyi kaybedebiliyor. TLD tıkanma durumunda en iyi sonucu veriyor. Mosse yüksek hata oranına sahip. CSRT yüksek doğruluk oranına sahip.

1.3. Projenin Amacı

Bu çalışmayı yapmamızdaki amaç Python ve OpenCV ile kameradan aldığımız görüntüyü işleyerek, bir hedef belirlemek ve bu hedefi farklı algoritmaları kullanarak takip etmektir. Cisimlerin kameranın kadrajından çıkması durumunda Python ile Arduino'nun haberleşmesini sağlayarak kamerayı hareket ettirerek takibini sürdürmeye devam etmektir. Bu sayede kamera veya cismin hareket etmesi durumunda bile cismi takip etmeye devam edebilmektir.

Görüntü işleme yöntemleri ile görüntüde bulunan nesneler sırasıyla nesnelerin tanımlanması, tespiti, sınıflandırılması ve takibi için birçok yol vardır. Her bir basamağın kendisine ait algoritmaları ve formülü vardır. Her bir basamaktaki başarı bir sonraki basamağı etkilemektedir. İşlem yapılacak görüntüdeki çözünürlük, ışık durumu, hareketli nesne sayısı, görüntüyü kayda alan eleman gibi etkenler çalışmanın başarısını etkilemektedir. Askeri uygulamalarda kullanılan elektro-optik algılayıcı, son yıllarda yapay zeka tabanlı derin öğrenme algoritmaları ile güçlendirilerek hareketli ve sabit hedeflerin belirlenmesinde ve takibinde hem daha hızlı hem de daha kesin olarak çalışmaktadır. Bu çalışmada Python ve OpenCV ile kameradan alınan gerçek zamanlı görüntüyü işleyerek, nesnenin ya da kameranın hareket etmesi durumunda hedef nesnenin görüntü çerçevesinden çıkmasını engellemek için Arduino ile kameraya iki eksende bağlı motorları sürerek hedef seçilen nesnenin takibini yapmaktır.

1.4. Projenin Gerçekleştirilebilirliği

Çalışmada ekonomik maliyet çıkaracak nesneler bir adet kamera, bir adet arduino ve iki adet motordur.

10 x Jumper Kablo

2 x Micro Servo SG90

1 x 3D printed Pan Tilt

1 x Arduino Uno

1 x BreadBoard

Burada maliyeti en fazla olan nesne kameradır. Kameranın özellikleri olan optik lens kalitesi, video çözünürlüğü ve saniyede tazeleme oranına göre maliyeti değişmektedir. Bu özelliklerinin daha iyi olması sonucu görüntü girdisinin kalitesi artar dolayısıyla çerçevede gözüken cisimlerin tespiti, sınıflandırılması ve takibi daha iyi bir şekilde yapılır. Bu çalışmada kullanılan malzemelerin toplam maliyeti üç yüz Türk lirasıdır.

Çalışmanın üretilebilirliği gayet kolaydır. Üretim aşaması motorların kameraya montajı, motorların arduino ile bağlantısı, kamera ve arduino'nun bilgisayara bağlanmasından oluşur.
Çalışmanın kullanıcıya ya da etrafa sağlık bakımından herhangi bir zararı bulunmamaktadır.

1.5. Projenin Problemin Çözümünde Kullanımı

Python derleyicisine OpenCV kütüphanesi indirildi. OpenCV kütüphanesi kullanılarak nesne takip algoritmaları kullanıldı. Nesnenin çerçevedeki konumu belirlendi. Konum bilgileri Python üzerinden Arduino'ya gönderildi. Nesnenin belirli konumu üzerinden x ve y ekseni olmak üzere 2 adet servo motor bulunan değere göre hareket ederek cisim çerçevenin merkezine getirildi.

2.1.Projenin Organizasyonu

Grup liderleri Gürbüz Kaan AKKAYA ve Ömer Can VURAL projenin kontrolünü yapar. Beklenen ile sonucu karşılaştırır.

Problemin Belirlenmesi Gürbüz Kaan Akkaya, Ömer Can Vural, Büşra Kaya, Ali Nefvel Aras

Gereksinimlerin belirlenmesi araştırma ve ihtiyaçlardan Gürbüz Kaan Akkaya, Ömer Can Vural, Büşra Kaya sorumludur.

Analist Gürbüz Kaan Akkaya, Ömer Can Vural, Büşra Kaya gereksinimlere göre analiz yapar.

Tasarımcı olan Gürbüz Kaan Akkaya ve Ömer Can Vural analizi yapılmış olan programın tasarımını yapmakla sorumludur.

Yazılımcı olan Gürbüz Kaan Akkaya ve Ömer Can Vural tasarımı yapılmış olan programın uygulamasını yapmakla sorumludur.

Donanım ve Fiziksel Uygulamadan Gürbüz Kaan Akkaya ve Ömer Can Vural programın donanım parçasını yapmakla ve çalıştırmakla sorumludur.

2.2.İşlevsel Olmayan Gereksinimler

Kullanılabilirlik :

Projenin kapsamı gereği çalışması için gerekli python ve pip sürümü olmadan gerekli çıktıyı sağlayamayacağı için ve uygulamanın tam işlevsellik ile çalışabilmesi için Arduino gerekli olduğundan uygulamanın kullanılabilirliği düşüktür.

Uygunluk :

Proje Python kodunun derlenebildiği tüm sistemler ile uyumludur

Python 3.7.6 sürümü ve OpenCV 4.2.0 sürümü ve pip 21.0.1 sürümü gerektirir

Arduino kullanılmak zorunludur.

Taşınabilirlik :

Bilgisayarın fiziksel depolama ortamına kopyalanmadan, kurulmadan ve derlenmeden Python kodu, fiziksel devre oluşturulmadan da Arduino kodu çalıştırılamaz

Sürdürülebilirlik :

Sisteme kod açısından bakıldığında if-else gibi koşul yapıları sık kullanılmasına karşın tüm kodlar ve işlevleri yorum satırları ile desteklenmiş ve malzeme kalitesinden kaynaklı hata oranı yüksek olmasına karşım onarım ve bakım hızının da yüksek olması sistemin sürdürülebilirliğini yüksek kılar.

Güvenilirlik :

Bu kalite özelliği,kullanıcının hedef takibi için algoritma seçimi ve hedef belirlemesi ile programın eş zamanlı bir frame oluşturması arasında sürenin yeterli olması ile sağlanır.

2.3.İşlevsel Gereksinimler

Kapsam :

Kullanıcılar sistemi bilgisayarlarına kurulmuş programlar ve fiziksel devre vasıtası ile kullanabilirler.

Veri gereksinimleri :

Girilen veriler (Seçilen hedef) ve sonuç verileri (hedefin kamera görüntüleri)anlık işlenmektedir. Veriler her hangi bir yapay zeka eğitimi amacı ile kullanılmadığı sürece saklanmaz.

Yapısal :

Sistem hedef takip programı çalıştığı sürece açık bulunacaktır. Gereksinimler arası geçişler butonlar yardımıyla yapılacaktır.

1.Yeni sistem, kullanıcılarına hangi yetenekleri sağlamalıdır?

Bu proje savunma sanayisinde hedef takibi, görsel işleme ile alakalı tüm yapay zeka teknolojilerinin gerçek zamanlı eğitilmesinde kullanılabilir.

2.Hangi veriler toplanmalı ve saklanmalıdır?

Kullanıcın seçtiği takip algoritması seçimi, takip edilmesi istenen nesnelerin girdisi ve kullanıcının kaydetmek istediği görüntüleri saklar.

3.Hangi performans seviyesi bekleniyor?

Basit, orta ve üst düzey projelerin her birinde kullanılabilecek bir sistem tasarımı yapıldı bu yüzden üst seviye bir performans bekleniyor.

4.Çeşitli gereksinimlerin öncelikleri nelerdir?

Uygulamada görsel denetim ve nesne takibi yapabilmek için Python ve OpenCV kullanıldı. Python nesne yönelimli bir programlama dilidir. Python ile sistem otomasyonu, web, makine öğrenimi, veri bilimi, uygulama programlama ara yüzü gibi birçok alanda çalışma yapılabilir. OpenCV açık kaynak kodlu görüntü işleme kütüphanesidir. OpenCV kütüphanesi görüntü işleme ve makine öğrenmesine ait algoritmalar barındırır. Nesnenin ekran dışına çıkması durumunda takibini yapabilmek için motor sürme işleminde Arduino ile kullanıldı. Arduino elektronik donanım ve yazılım temelli açık kaynaklı bir mikro kontrolcü platformudur.

2.4. Yöntem Hakkında Genel Bilgi

Görsel denetim ve nesne takibi yapmak için birçok yol vardır. Nesne denetim için 3 farklı yol vardır. Bunlar çerçeveler arasındaki fark, optik akış, arka plan modeli çıkarmadır. Nesne sınıflandırma için ise cismin öznitelikleri olan renk, eğim, doku, hareket ve zamansal değişiklere bakılır. Nesne sınıflandırma yöntemleri ise 4 temel gruba ayrılır şekil tabanlı, hareket tabanlı, renk tabanlı ve doku tabanlıdır. Nesne takip yöntemleri ise nokta tabanlı çekirdek tabanlı ve siluet tabanlı olmak üzere 3 genel gruba ayrılır. Görsel denetim ve nesne takibi yapmak için C++, python, matlab kullanılan programlama dillerinden bazılarıdır. Bu çalışmada Python ile OpenCV kütüphanesin 7 farklı nesne takip algoritmasını kullanıldı.

2.4.1. CSRT Tracker

CSRT takip algoritması, takip için çerçeveden seçilen bölgenin bir kısmını filtre desteğini ayarlamak için uzamsal güvenilirlik haritası kullanır. CSRT, kanallar arasında korelasyon yanıtları elde etmek için yalnızca iki standart özellik kullanır. Bunlar HOG'lar ve renk adlarıdır

2.4.2. KCF Tracker

KCF takipçi algoritması fikri basit olmasına rağmen, son zamanlarda en iyi performansı gösteren takipçiler arasında, en hızlı ve en yüksek performansı elde eder. KCF takipçisinin anahtarı, yüksek verimlilik için sirkülasyon matrisinin yapısını keşfederken, takip detektör şemasının ayırt etme yeteneğini geliştirmek için negatif örneklerin çoğaltılmasını kullanır. Veriler dolaşım matrisleri denklem (1.1) ile ifade edilebilir. F,verileri Fourier alanına dönüştüren bir matristir. F^H F'nin Hermitian transpozudur. Dolaşım matrisinin ayrıştırılması, doğrusal

regresyon çözümünü basitleştirmek için kullanılabilir formüle edilmiş hali denklem (1.2)'de verilmiştir. Denklem (1.3)'te verilen temel örneklemlerin doğrusal kombinasyonu, sırt regresyonunun yakın form çözümü ile denklem (1.4) haline gelir. Denklem (1.1) ile bölerek $\hat{x}$ karmaşık kompleks sayısı elde edilir. Doğrusal olmayan regresyon durumunda, daha güçlü sınıflandırıcı kullanmak için denklem (1.5)'deki çekirdek numarası uygulanır. Bu sayede ikili uzay katsayısı α denklem (1.6)'daki gibi bulunur. Burada k^{xx} çekirdek korelasyonu olarak tanımlanır. Denklem (1.7)'deki dolaşım matris tekniğini kullanarak Gauss çekirdeğini buluruz. Nesnenin bir sonraki çerçevede aynı konumdaki z yaması, Fourier yanıtını hesaplamak için temel örnek olarak kabul edilir ve denklem (1.8) ile hesaplanır. Burada $\tilde{x}$ modelde öğrenilecek verileri gösterir. $\hat{f}(z)$ tekrar uzaysal alana dönüştürüldüğünde maksimum tepkiye göre öteleme, izlenen hedefin hareketi olarak kabul edilir.

$$X = F^H diag(Fx)F \quad \textbf{(1.1)}$$

$$min_w \sum_i^n (f(x_i) - y_i)^2 + \lambda \|w\| \quad \textbf{(1.2)}$$

$$f(x) = w^T x \quad \textbf{(1.3)}$$

$$w = (X^T X + \lambda I)^{-1} X^T y \quad \textbf{(1.4)}$$

$$f(z) = w^T z = \sum_{i=1}^n \alpha_i \kappa(z, x_i) \quad \textbf{(1.5)}$$

$$\hat{\alpha}^* = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \quad \textbf{(1.6)}$$

$$k^{xx'} = \exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|x'\|^2) - 2F^{-1}(\hat{x} \odot \hat{x}'^*)\right) \quad \textbf{(1.7)}$$

$$\hat{f}(z) = (\hat{k}^{\tilde{x}z})^* \odot \hat{\alpha} \quad \textbf{(1.8)}$$

2.4.3. Boosting Tracker

Boosting takipçi bir çerçeve verildiğinde çerçevenin konum mahallesindeki her pikselde çalıştırılır ve sınıflandırma puanı kaydedilir. Nesnenin yeni konumu tekrar sınıflandırma yapıldığında puanın maksimum olduğu yerdedir. Nesne tahmini zayıf sınıflandırıcı tarafından oluşturulan h^{weak} hipotezi öğrenme algoritması kullanılarak elde edilir. Denklem (1.9)'den optimizasyon kriterine göre bir m seçicisi seçilir. Aslında m için Denklem (1.11)'te gösterilen her bir $h_i^{weak} \in H^{weak}$ için beklenen e_i değeri kullanılır. Bir dizi N zayıf sınıflandırıcısı verildiğinde denklem (1.12) ile seçicilerin doğrusal kombinasyonu ile güçlü sınıflandırıcı hesaplanır. Ayrıca marjin ile ilgili olan $conf()$ değeri, güçlü sınıflandırıcının bir güven ölçüsüdü.

$$H^{weak} = \{h_1^{weak}, \dots, h_M^{weak}\} \quad \textbf{(1.9)}$$

$$h^{sel}(x) = h_m^{weak}(x) \quad \textbf{(1.10)}$$

$$m = argmin_i e_i \quad \textbf{(1.11)}$$

$$hStrong(x) = sign(conf(x)) \quad \textbf{(1.12)}$$

$$conf(x) = \sum_{n=1}^{N} \alpha_n \, h_n^{sel}(x) \quad \textbf{(1.13)}$$

2.4.4. MIL Tracker

Mil yani çoklu örnek öğrenme takipçi birden çok pozitif ve negatif örnekleyici kullanarak bunları pozitif ve negatif torbalara koyar. Olabilirlik fonksiyonu sınıflandırır. Bir dizi görüntü kırpılır ve denklem (1.14) ile özellik vektörleri hesaplanır. Denklem (1.15) ile MIL sınıflandırıcısı kullanılır. Denklem (1.16) ile takipçi konumu güncellenir. Denklem (1.17) ile iki grup görüntüsü kırpılır. MIL görünüm modeli X^r pozitif torbası ve $|X^{r,\beta}|$ negatif torbası ile güncellenir. Her biri $X^{r,\beta}$ setinden tek bir görüntü içerir.

$$X^s = \{x|s > \|l(x) - l^*_{t-1}\|\} \quad \textbf{(1.14)}$$

$$p(y = 1|x) \quad \textbf{(1.15)}$$

$$l^*_t = l(argmax_{x \in X^s} p(y|x)) \quad \textbf{(1.16)}$$

$$X^r = \{x|r > \|l(x) - l^*_t\|\} \ , \ X^{r,\beta}\{x|\beta > \|l(x) - l^*_t\| > r\} \quad \textbf{(1.17)}$$

2.4.5. TLD Tracker

TLD yöntemi takip etme problemini üç alt gövdeye ayırır. Bunlar izleme, öğrenme ve algılamadır. Her bir alt görev tek bir modül tarafından adreslenir ve tüm modüller aynı anda çalışır. İzleme modülü medyan akış algoritması tarafından uygulanır. Medyan akış teoremi ilk olarak noktaları sınırlayıcı kutu içine alır ve yer değiştirmelerinin güvenirliğini tahmin eder. En çok yer değiştirmeye sahip noktaların yarısı özellik noktaları olarak seçilir. Nesnenin hareketi, özellik noktalarının hareketinin medyanı ile elde edilir. Yer değiştirmenin güvenilirliği Lucas-Kanade algoritmasından yararlanarak düzgelenmiş çapraz ilinti katsayısı (NCC) ile tahmin edilir. NCC izlenen noktayı çevreleyen yerleri son çerçeve (P_1) ile şuan ki çerçeveyi (P_2) karşılaştırır.

$$R(x,y) = \frac{\sum_{x',y'}(P_1(x',y') * P_2(x+x',y+y'))}{\sqrt{\sum_{x',y'} P_1(x',y')^2 \sum_{x',y'} P_2(x+x',y+y')}} \qquad \textbf{(1.18)}$$

2.4.6. Median Flow Tracker

Medyan akış algoritması birbiri ardına gelen video dizisi karelerinde seyrek optik akış yardımıyla bir nesnenin konumunu tahmin etmek için kullanılır. Nesnenin, tüm nesnenin hareketi ile senkronize hareket eden küçük ve sert bir şekilde bağlı yamalardan oluştuğu varsayılır. Denklem (1.19)'deki durum sağlandığında bitişik noktaların hareket etmesi gerektiği ve kabaca aynı yer değiştirmeye sahip olur. Burada N_i 4 adet komşusu olan i noktalarına sahip bir nokta kümesidir. Parantez içi bağımsız değişkeni doğru ise 1'e aksi taktirde 0'a eşit olan bir fonksiyondur. Δ_i i'inci noktanın yer değiştirmesidir. ϵ yer değiştirme eşiğidir ve $\epsilon < 0.5$ olarak alınır. $S_i^{Nh} < \theta\,, \theta < 1$ koşulu sağlandığında noktanın komşularıyla eş zamanlı olarak hareket ettiği kabul edilir.

$$S_i^{Nh} = \sum_{j \in N_i} (\left\| \Delta_i - \Delta_j \right\|^2 > \epsilon) \qquad \textbf{(1.19)}$$

2.4.7. Mosse Tracker

Mosse takipçi algoritması daha az eğitim görüntüsü ile sentetik tam filtrelerin ortalaması yöntemi benzeri filtreler üretmek için kullanılır. Mosse takipçi algoritması için bir dizi eğitim görüntüsü f_i ve eğitim çıktıları olan g_i ihtiyaç vardır. g_i herhangi bir şekle girebilir. Bu durumda g_i, eğitim görüntüsünde hedef üzerinde merkezlenmiş kompakt 2D Gauss şekilli bir tepe noktasına sahip olacak şekilde temel gerçeklikten üretilir. Girdi ve çıktı arasındaki basit öğe temelli ilişkiden yararlanmak için Fourier bağlantısı kullanılır. Büyük harf değişkenler olan F_i, G_i ve filtre H'yi küçük harf karşılıklarının Fourier dönüşümü olarak tanımlanır. Eğitim girdilerini istenen eğitim çıktılarıyla eşleştiren bir filtre bulmak için, denklem (1.21) ile konvolüsyonun gerçek çıktısı ile istenen evrişim çıktısı arasındaki hata karesinin toplamını en aza indiren bir H filtresi bulunur. Takip etmede hedef her zaman ortalanmaz ve öğrenme çıktılarındaki tepe öğrenme girdilerindeki hedefi takip etmek için hareket eder. Genel olarak öğrenme çıktıları herhangi bir şekle sahip olabilir. Bu optimizasyon problemini çözmek için ilk olarak filtrenin ω ve v tarafından indekslenen her bir elemanı bağımsız olarak çözülebilir çünkü Fourier alanındaki tüm işlemler eleman bazında gerçekleştirilir. Bu, işlevi $H_{\omega v}$ ve $H^*_{\omega v}$ cinsinden yazmayı içerir. Denklem (1.22) yapılarak, $H_{\omega v}$ bağımsız bir değişken olarak ele alınır. H^* için çözüldüğünde, denklem (1.23)'teki Mosse filtresi için bir kapalı form ifadesi bulunur.

$$H_i^* = \frac{G_i}{F_i} \qquad \textbf{(1.20)}$$

$$min_{H^*} \sum_i |F_i \odot H^* - G_i|^2 \qquad \textbf{(1.21)}$$

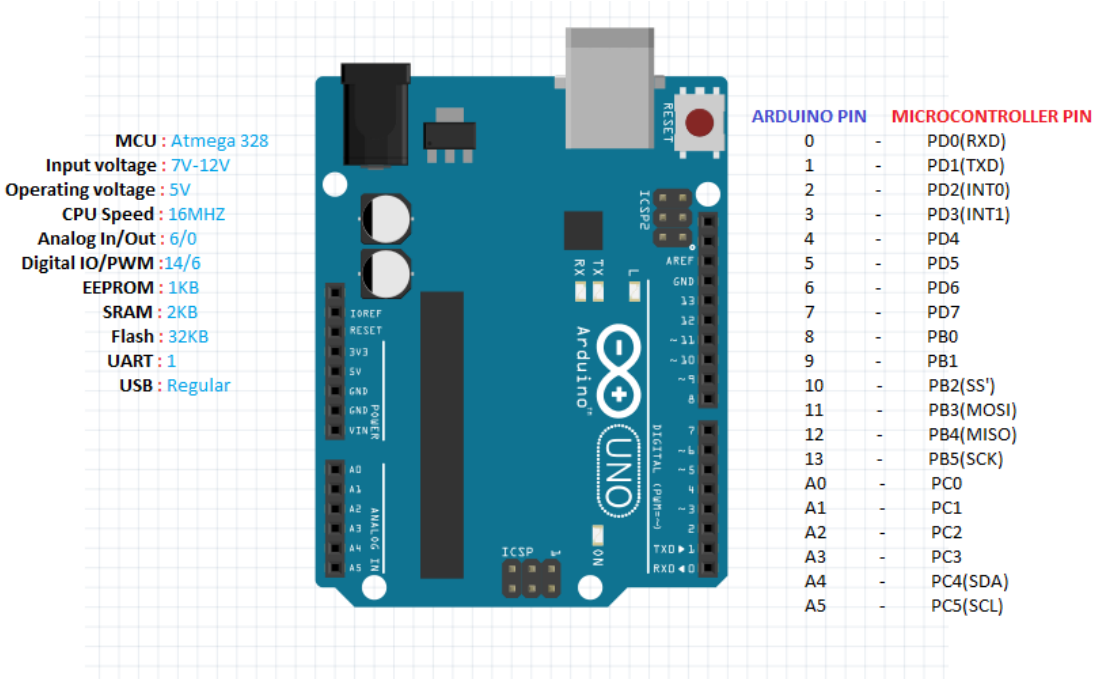
$$0 = \frac{\partial}{\partial H^*_{\omega v}} \sum_i |F_{i\omega v} H^*_{\omega v} - G_{i\omega v}|^2 \qquad \textbf{(1.22)}$$

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*} \qquad \textbf{(1.23)}$$

2.5. Uygulamada Kullanılan Araç ve Gereçler

Uygulamada görsel denetim ve nesne takibi yapabilmek için Python ve OpenCV kullanıldı. Python nesne yönelimli bir programlama dilidir. Python ile sistem otomasyonu, web, makine öğrenimi, veri bilimi, uygulama programlama arayüzü gibi birçok alanda çalışma yapılabilir. OpenCV açık kaynak kodlu görüntü işleme kütüphanesidir. OpenCV kütüphanesi görüntü işleme ve makine öğrenmesine ait algoritmalar barındırır. Nesnenin ekran dışına çıkması durumunda takibini yapabilmek için motor sürme işleminde Arduino ile kullanıldı. Arduino elektronik donanım ve yazılım temelli açık kaynaklı bir mikrokontrolcü platformudur.

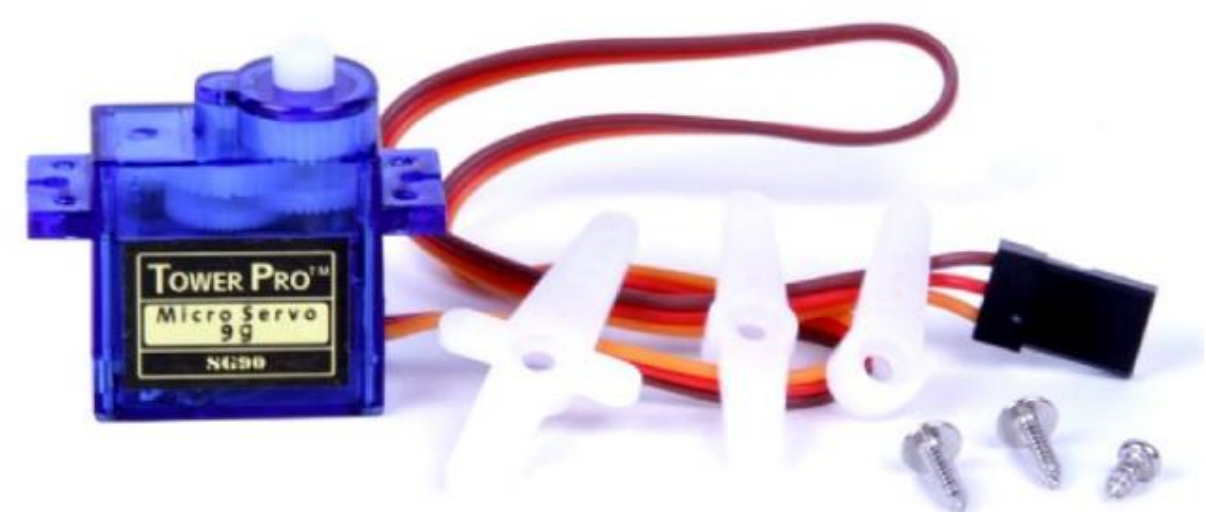
Arduino Data Sheet



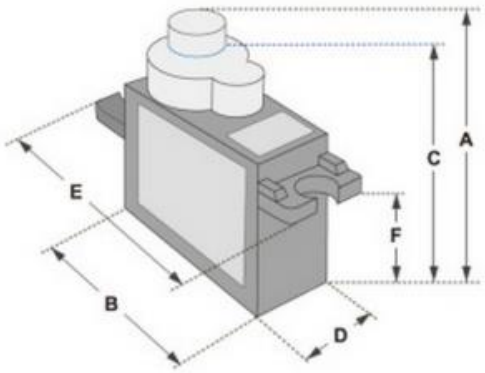
Arduino Uno, ATmega328 tabanlı bir mikrodenetleyici kartıdır. 14 dijital giriş / çıkış pinleri (6 tanesi PWM çıkışı olarak kullanılabilir), 6 analog giriş pini bulnur. Arduino Uno, USB bağlantısı veya harici bir güç kaynağı ile çalıştırılabilir. Güç kaynağı otomatik olarak seçilir.Harici (USB olmayan) güç, AC-DC adaptöründen veya pilden gelebilir.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Servo SG90s micro Data Sheet

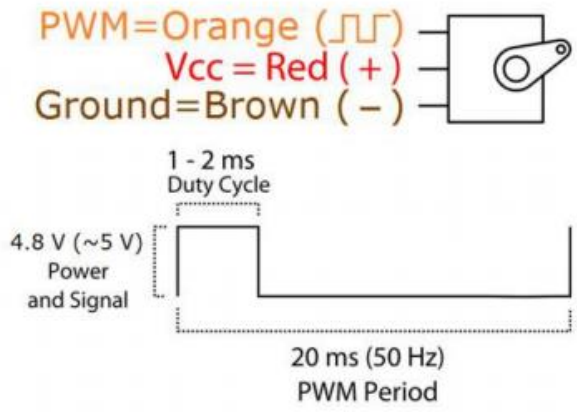


Servo yaklaşık 180 derece dönebilir (her yönde 90) ve tıpkı standart türler gibi çalışır
ama daha küçük. Bu servoları kontrol etmek için herhangi bir servo kodunu, donanımı veya kitaplığı kullanabilirsiniz.

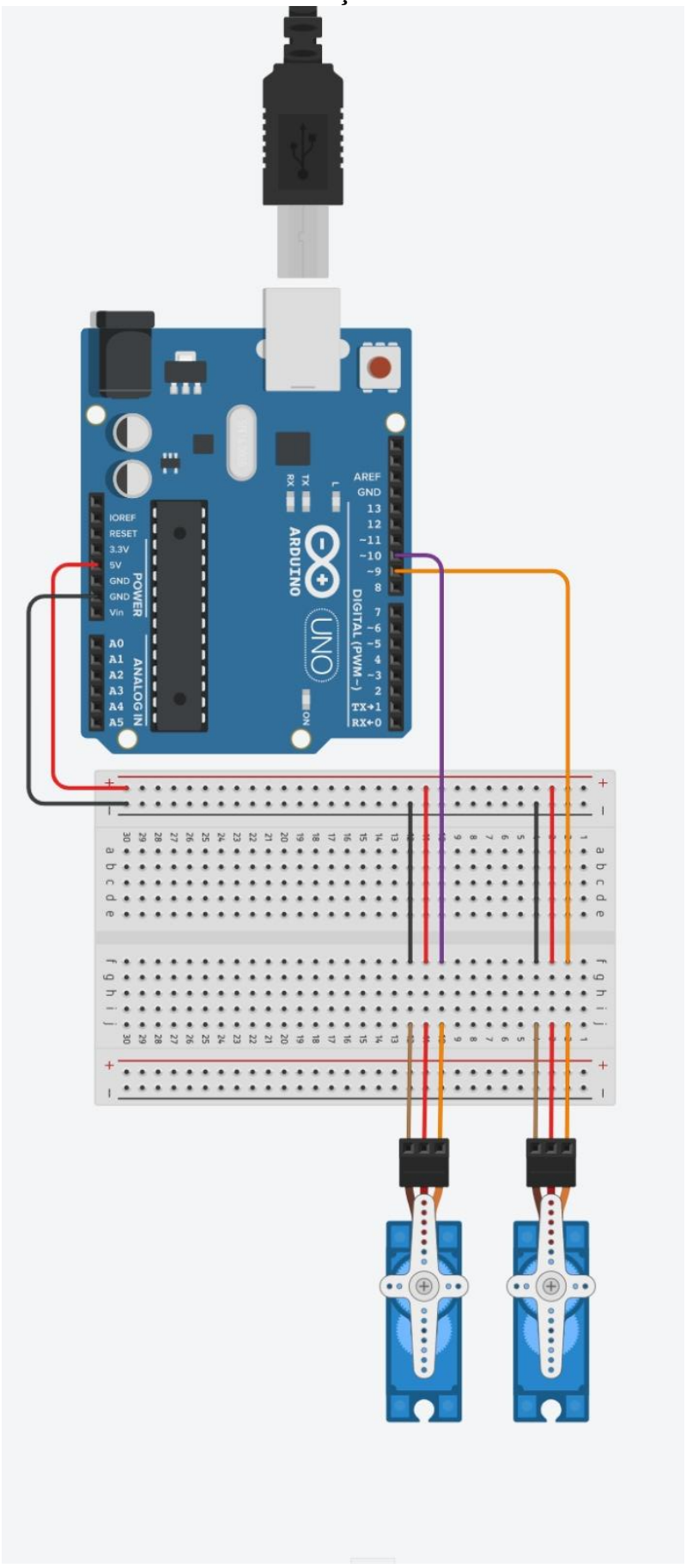


Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

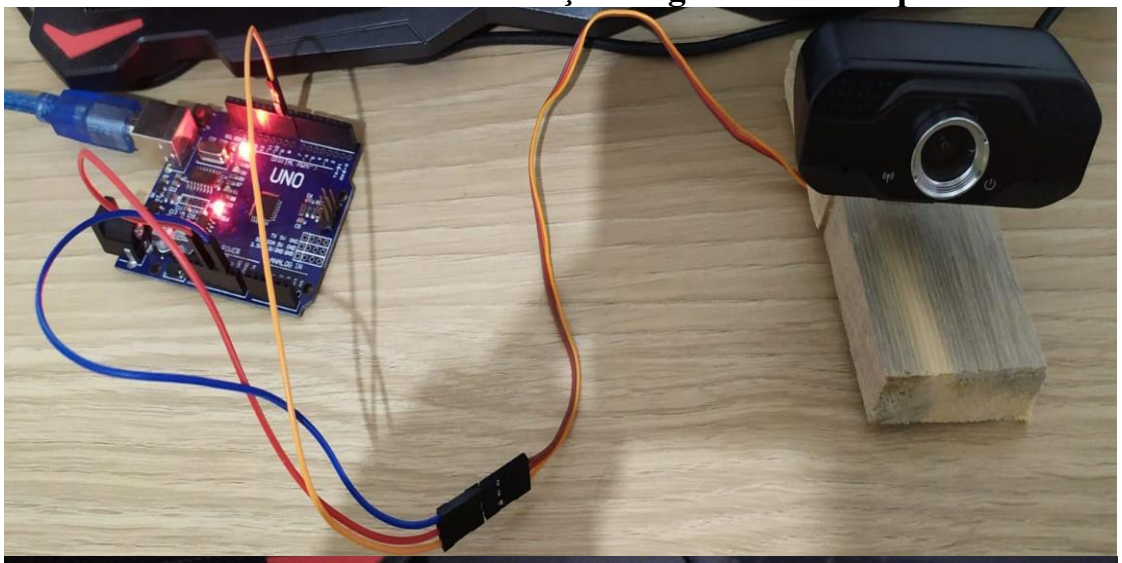
Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



DEVRE ŞEMASI:



Deneme Amacı ile Oluşturduğumuz Prototip



Python Yazılımı ilk Deneme ve Programın tek eksen üzerinde denenmesi

```
import cv2
import time
import serial
import struct

seri = serial.Serial("COM3",9600)
time.sleep(2)

OPENCV_OBJECT_TRACKERS = {"csrt"       : cv2.TrackerCSRT_create,
                          "kcf"        : cv2.TrackerKCF_create,
                          "boosting"   : cv2.TrackerBoosting_create,
                          "mil"        : cv2.TrackerMIL_create,
                          "tld"        : cv2.TrackerTLD_create,
                          "medianflow": cv2.TrackerMedianFlow_create,
                          "mosse"      : cv2.TrackerMOSSE_create}

tracker_name = "boosting"
tracker = OPENCV_OBJECT_TRACKERS[tracker_name]()

cap = cv2.VideoCapture(0)
cap.set(3,960)
cap.set(4,480)

x1=0
y1=0

success, img = cap.read()
def drawBox(img, bbox):
    x, y, w, h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    center_x = int(x+w/2)
    center_y = int(y+h/2)
    cv2.rectangle(img, (x,y), ((x+w), (y+h)), (255,0,255), 3, 1)
    cv2.circle(img, (center_x, center_y), 2,(0,0,255),-1)
    cv2.putText(img,"Object At :",(10,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255
    cv2.putText(img,"X ="+str(center_x),(150,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(
    cv2.putText(img,"Y ="+str(center_y),(250,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(
    cv2.line(img, (320, 240), (center_x, center_y), (255,0,255), 1)

def servo(bbox):
    x1=int(180-((bbox[0]+bbox[2]/2)/3.55))
    print(x1)
    y1=int(180-((bbox[1]+bbox[3]/2)/2.66))
    print(y1)
    seri.write(struct.pack('>BB', x1,y1))

```

Projenin Python dilinde ki yazılım kısmını ilk oluşturduğumuzda “tracker” değeri kullanıcı tarafından seçilemez ve sadece yazılımcı tarafından yazılım üzerinde oynama yapılarak değiştirilebilirdi. Servo motorların dönmesi için Arduino tarafına gönderilen konum verileri ise Ana framede ki x-y koordinat düzleminde elde edilen konum bilgisini 0-180 arasında bir değere eşitlenmesi ile elde ediliyordu, yani seçilen objenin konumu ne ise servo motorlar da ona göre konumlandırılıyor fakat servo motorların aldığı açı, anaframede ki seçili objeyi değil, objenin ana framedeki konumuna göre artıp azalıyordu. “VideoCapture()” değeri servo motorumuzun üstündeki kameradan okuma yaptığında ise obje ana frame merkezinde olduğunda 90 dereceyi gösteriyor, obje merkezden saptığında

konum açıları objeye doğru döndürülüyor fakat kamera objeye doğru baktığında tekrar konum bilgisi 90 derece olduğundan kamera tekrar yanlış tarafa bakıyor ve motor sürekli obje ile merkez arasında dönüp duruyordu.

Python Yazılımı İkinci Deneme ve Programın tek eksen üzerinde denenmesi

```
import cv2
import time
import serial
import struct

seri = serial.Serial("COM3",9600)
time.sleep(2)

OPENCV_OBJECT_TRACKERS = {"csrt"      : cv2.TrackerCSRT_create,
                          "kcf"       : cv2.TrackerKCF_create,
                          "boosting"  : cv2.TrackerBoosting_create,
                          "mil"       : cv2.TrackerMIL_create,
                          "tld"       : cv2.TrackerTLD_create,
                          "medianflow": cv2.TrackerMedianFlow_create,
                          "mosse"     : cv2.TrackerMOSSE_create}
tracker_name = "boosting"
tracker = OPENCV_OBJECT_TRACKERS[tracker_name]()

cap = cv2.VideoCapture(1)
cap.set(3,960)
cap.set(4,480)
y1=0
liste=[0]
xlis=[90]

success, img = cap.read()
def drawBox(img, bbox):
    x, y, w, h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    center_x = int(x+w/2)
    center_y = int(y+h/2)
    cv2.rectangle(img, (x,y), ((x+w), (y+h)), (255,0,255), 3, 1)
    cv2.circle(img, (center_x, center_y), 2,(0,0,255),-1)
    cv2.putText(img,"Object At :",(10,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255
    cv2.putText(img,"X ="+str(center_x),(150,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(
    cv2.putText(img,"Y ="+str(center_y),(250,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(
    cv2.line(img, (320, 240), (center_x, center_y), (255,0,255), 1)

def servo(bbox, liste):
    x1=int(bbox[0]+bbox[2]/2)
    if x1 < 280 and x1 > 360:
        print("kilitlendi")
        xlis.append(xlis[len(xlis)-1])
        xlis.remove(xlis[0])

    elif x1 > 360:
        print("sagda")
        xlis.append(xlis[len(xlis)-1]-1)
        xlis.remove(xlis[0])

    elif x1 < 280:
        print("solda")
        xlis.append(xlis[len(xlis)-1]+1)
        xlis.remove(xlis[0])

    son=xlis[len(xlis)-1]
    print(son,xlis)
    seri.write(struct.pack('>BB', son,y1))

```

İlk denemede anladık ki motorlar objeye döndüğünde konumun objeye döndüğü şekilde kalması için bir hafıza ve durum ilişkisi olmalıydı, bu amaçla motorun başlangıç konumunu aklında tutan ve artıp azaldıkça bunu unutmayan bir değişken yaratmak için liste kullandık. İkinci denememize göre seçili obje, ana frame merkezine göre eksenin belli bir aralığının dışına çıkarsa listede tanımlı başlangıç değeri, obje o aralığa girene kadar artacak ya da azalacak ve bunu anlık konum bilgisi olarak aktaracaktı. Fakat Python komutları sırası ile çalıştıran bir dil olduğundan ve bizim seçme/takip gibi önemli işlemleri yerine getiren komut ve fonksiyonlarımız while döngüsü içerisinde bulunduğundan çalışma esnasında kod sürekli başa döndü ve ana frame de ki konumu anlaşılabilse bile başlangıç konumu sürekli nesnenin konumuna göre 90-91 ya da 90-89 arasında gidip geldi.

```
59  while True:
60      timer = cv2.getTickCount()
61      success, img = cap.read()
62      success, bbox = tracker.update(img)
63      print(bbox)
64      print(img.shape)
65      center = cv2.circle(img, (320, 240), 2,(0,0,255),-1)
66      if success:
67          drawBox(img, bbox)
68          servo(bbox, liste)
69      else:
70          cv2.putText(img,"'t' Tusuna Basarak Nesne Sec",(10,15),cv2.FONT_HERSHEY
71
72      cv2.imshow("Tracking",img)
73
74      key = cv2.waitKey(1) & 0xFF
75      if key == ord("t"):
76              bbox = cv2.selectROI("Tracking", img, False)
77              tracker.init(img,bbox)
78      elif key == ord("q"):
79          break
80
81  cap.release()
82  cv2.destroyAllWindows()
```

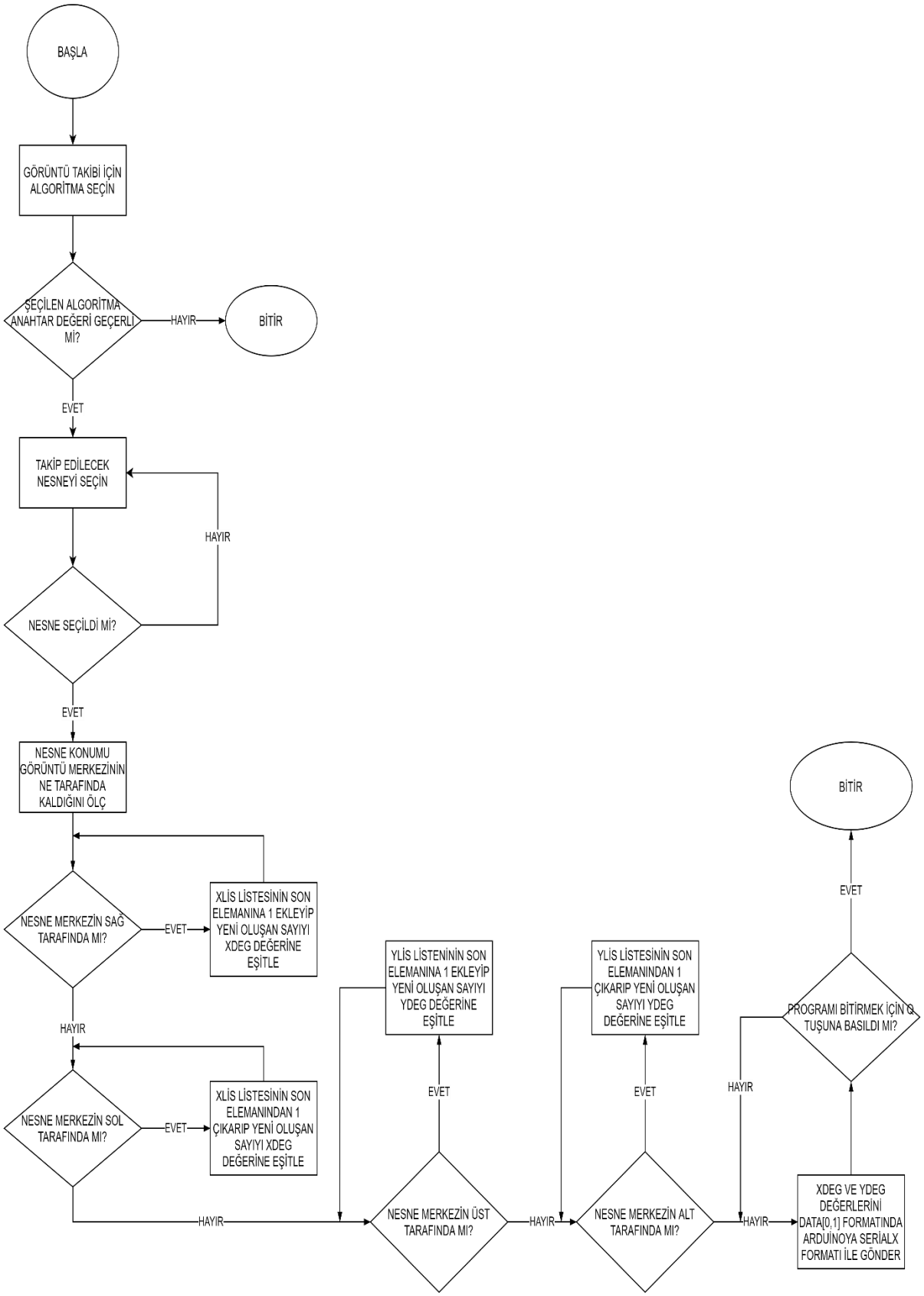
Arduino Yazılım Denemesi

```
#include <Servo.h>
int data_x1 = 0;
int data_y1 = 0;
int data[1];

Servo myservo_x1;
Servo myservo_y1;// Servo motor kontrolü için bir servo objesi oluşturuyoruz

void setup() {
  Serial.begin(9600);
  myservo_x1.attach(9);  // Servo motorları süreceğimiz pinlerin ataması
  myservo_y1.attach(10);
  myservo_x1.write(90); // Motorların başlangıç konumu 90 derece
  myservo_y1.write(90);
}

void loop() {
  while (Serial.available() >= 2) {
    data_x1=Serial.read();
    data_y1=Serial.read();


    myservo_x1.write(data[0]);
    myservo_y1.write(data[1]);
    Serial.println(data[0]);
    Serial.println(data[1]);
  }
}
```

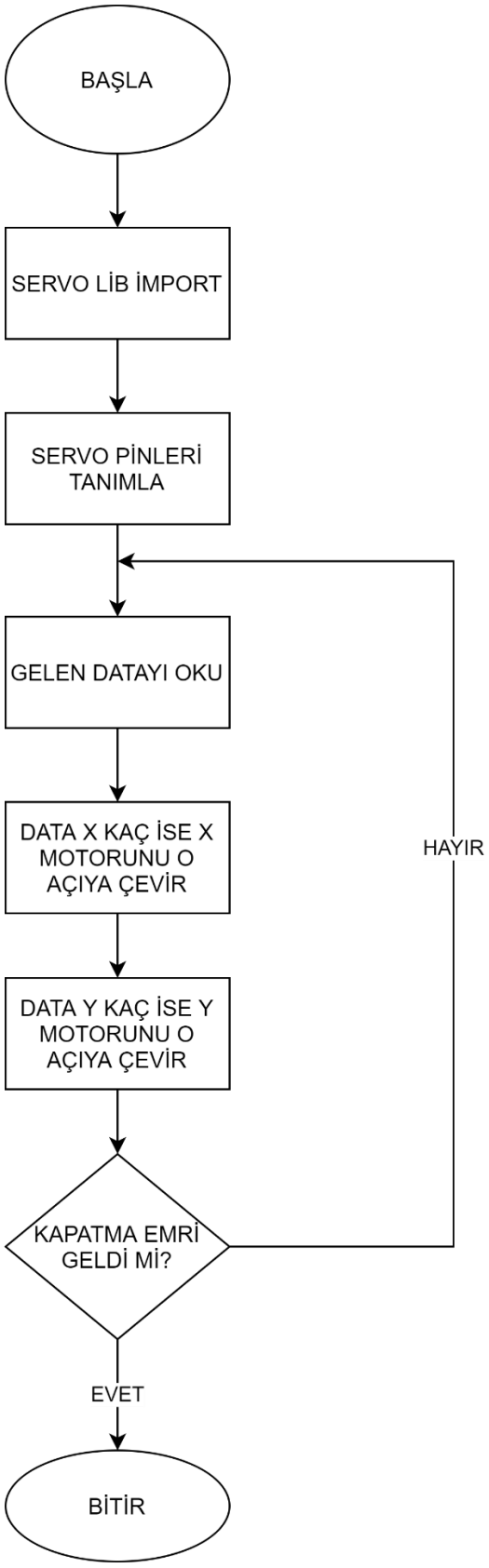
İlk yazdığımız arduino kodunda python struct tan gelen data bilgilerini x1 ve y1 olmak üzere iki ayrı data olarak inceleyip ayrı ayrı okuduk, her hangi bir problem yoktu sadece sade bir kod bütünü idi. İkinci yazdığımız arduino kodu ile

```
void loop() {
  while (Serial.available() >= 2) {
    //    data_x1=Serial.read();
    //    data_y1=Serial.read();
    for (int i = 0; i < 2; i++) {
      data[i] = Serial.read();
    }
    myservo_x1.write(data[0]);
    myservo_y1.write(data[1]);
    Serial.println(data[0]);
    Serial.println(data[1]);
  }
}
```

Gelen konum datalarını “for” döngüsü içine alarak okuduk, bunu yapmamızın amacı daha az değişken belirleyip daha sade bir kod elde etmekti.
Prototip ve denemelerden sonra çalışır program oluşturuldu

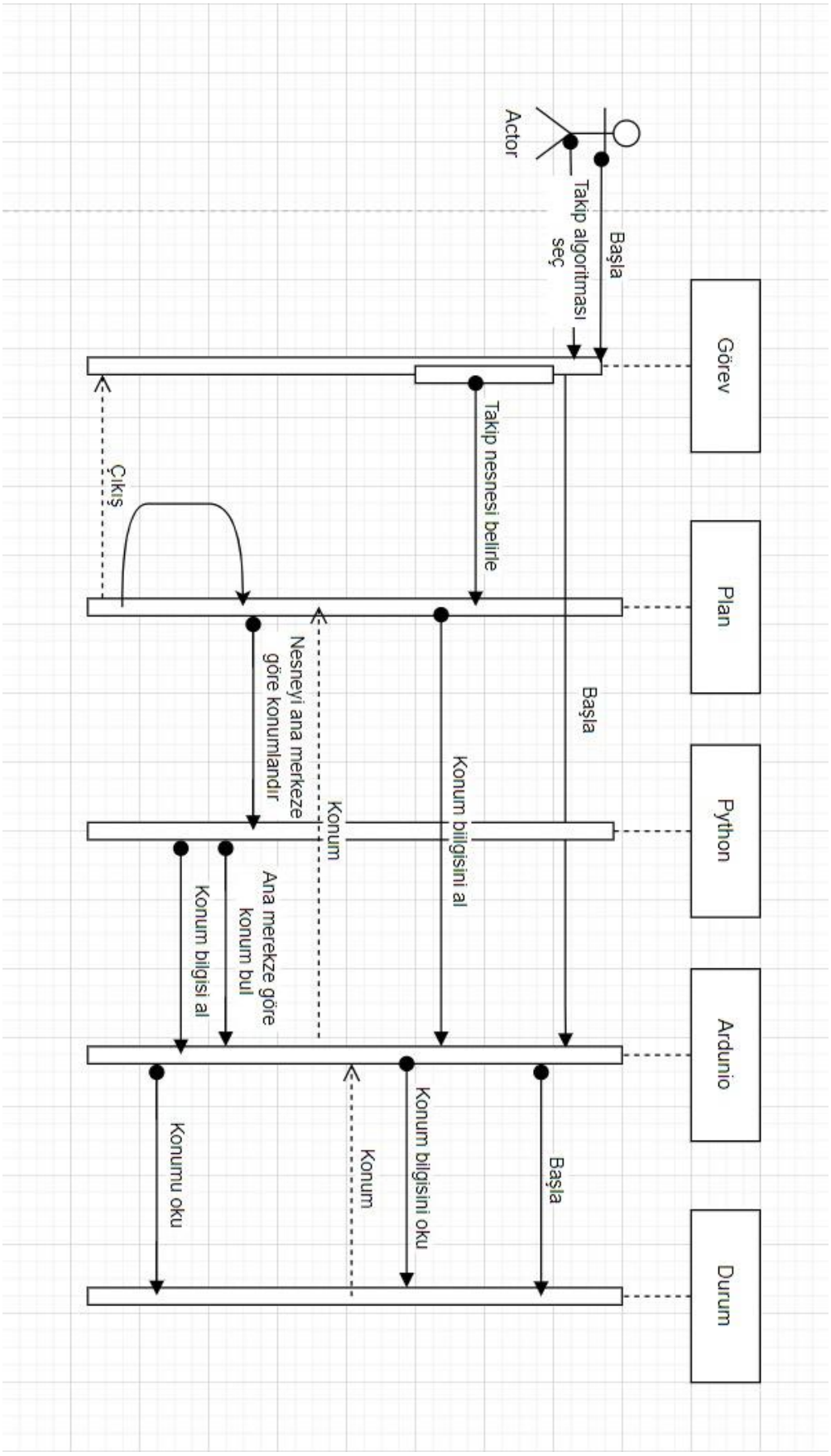


Uygulamanın Python akış diyagramı

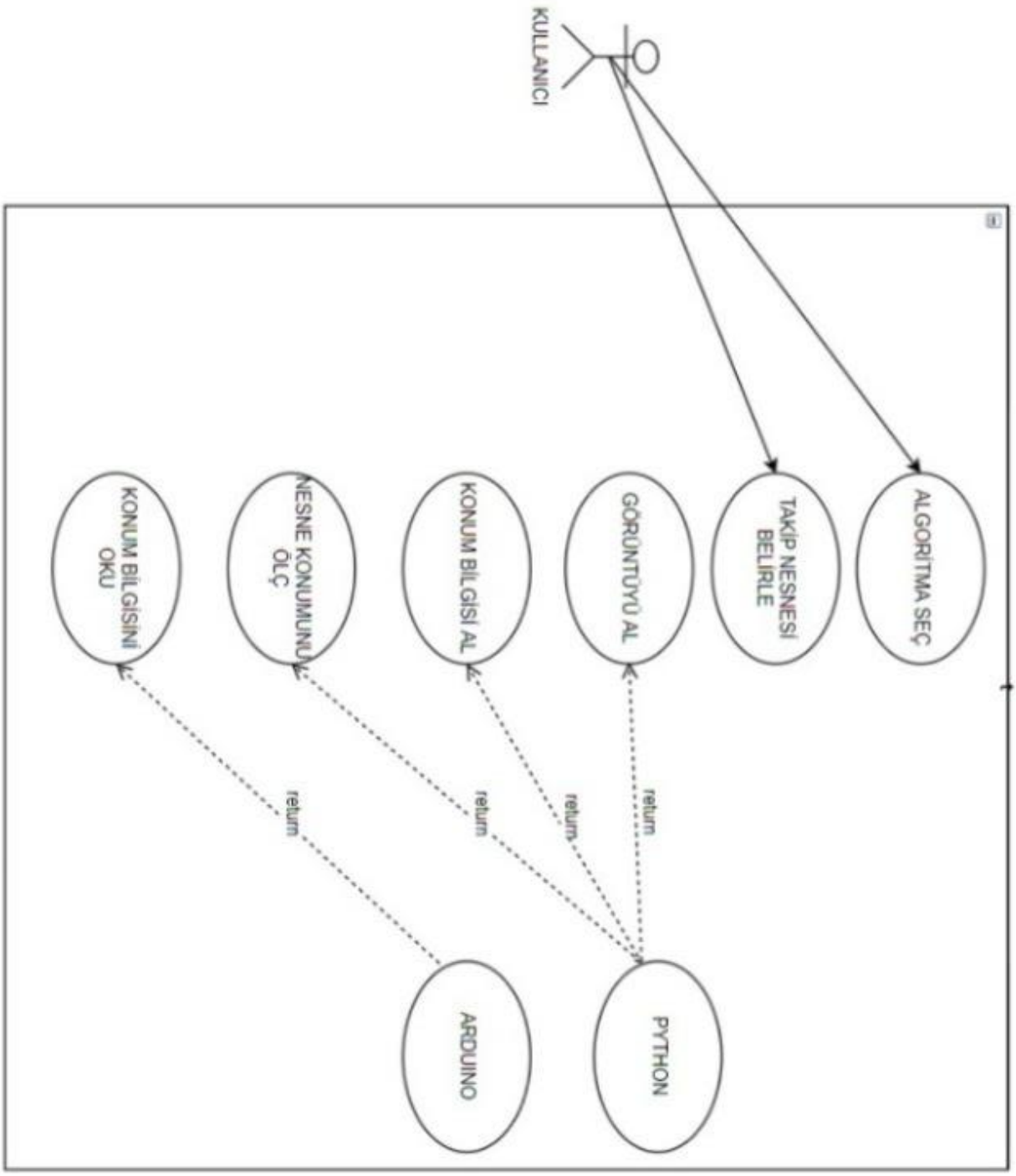


Uygulamanın Arduino akış diyagramı

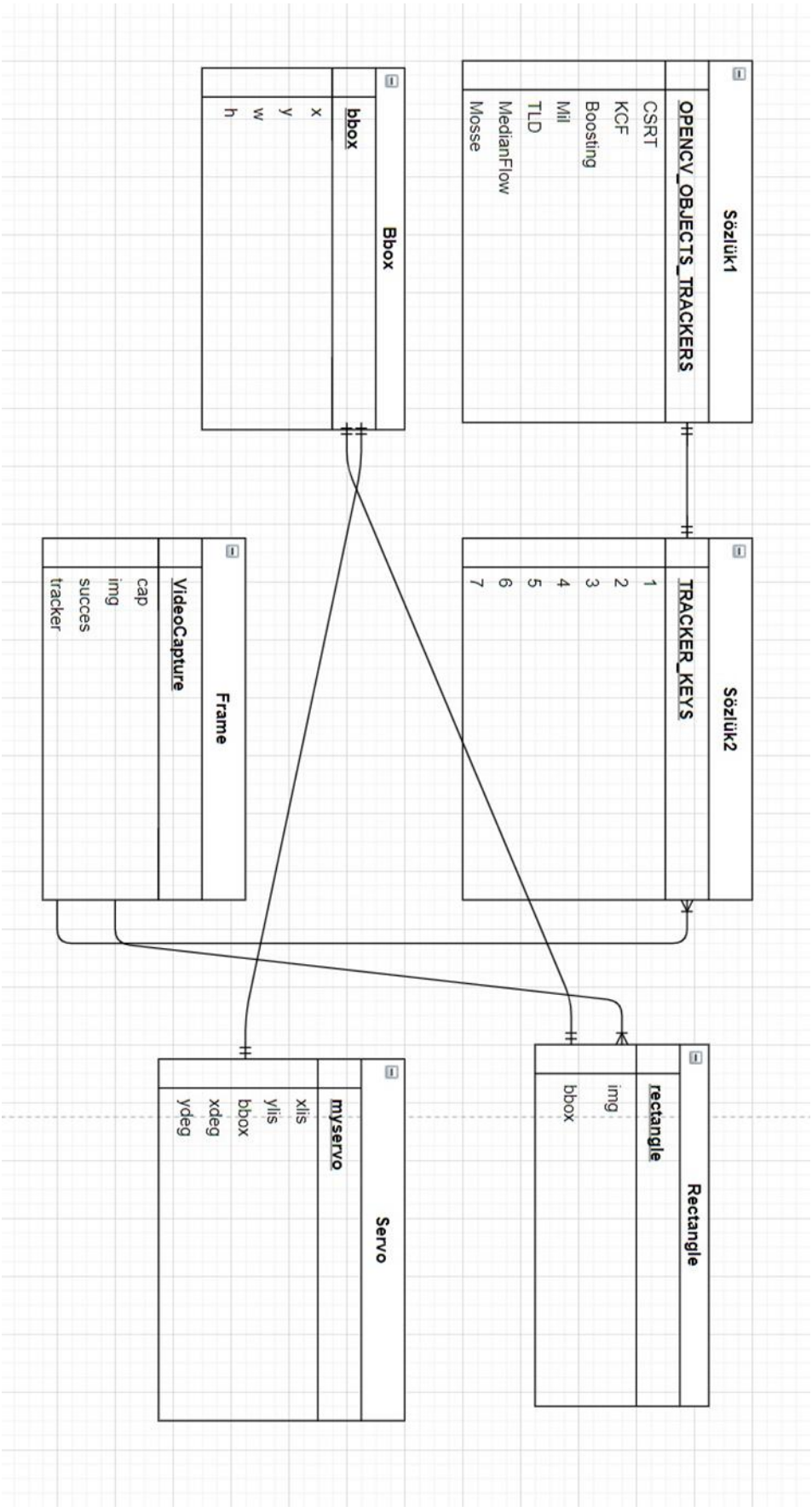
Sequence Diyagramı



Usecase Diyagram



Entity–relationship Diagram



3.1. Uygulamanın Gerçekleştirilme Aşamaları

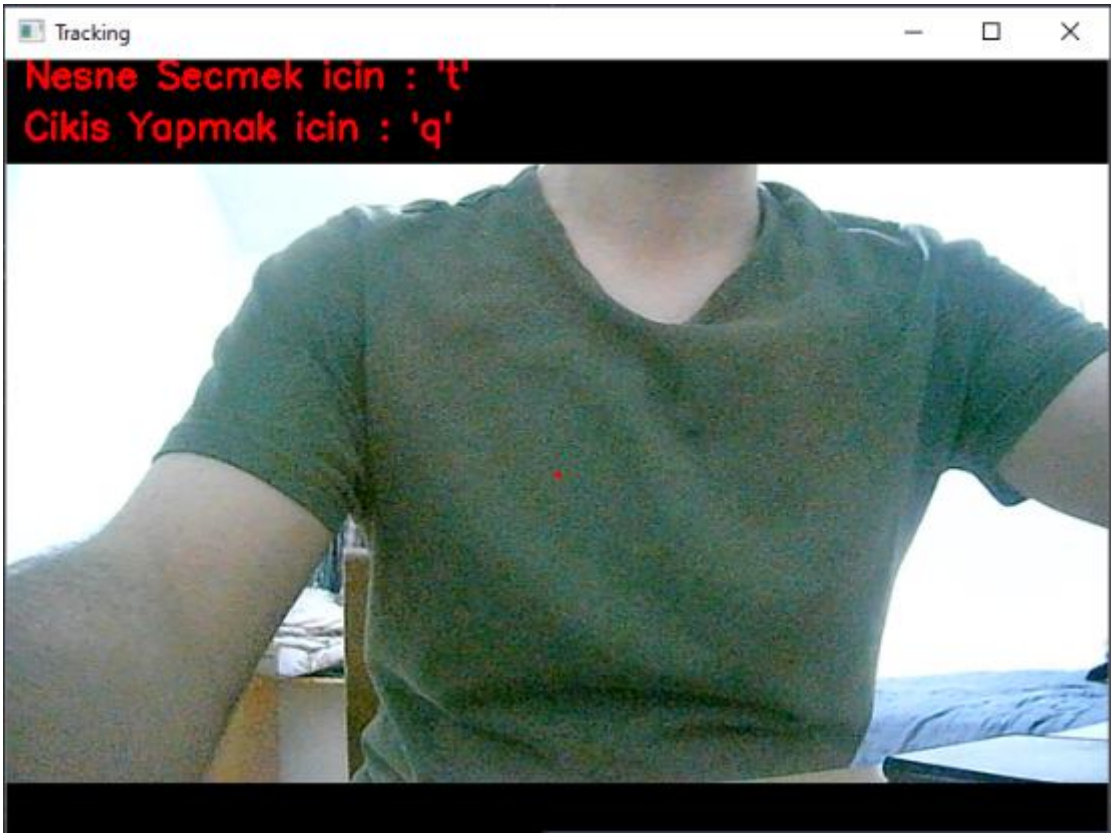
Uygulama nesneyi takip etmek için kullanıcıdan 7 farklı nesne takip algoritmasından birini seçmeni ister. Bu yapı Python'daki anahtar:değer çiftlerinden oluşan sözcükler kullanılarak tasarlandı. Artık nesne takibinde kullanacağımız takip algoritması belirlenmiş ve 'tracker' değişkeni tanımlanmış oldu.

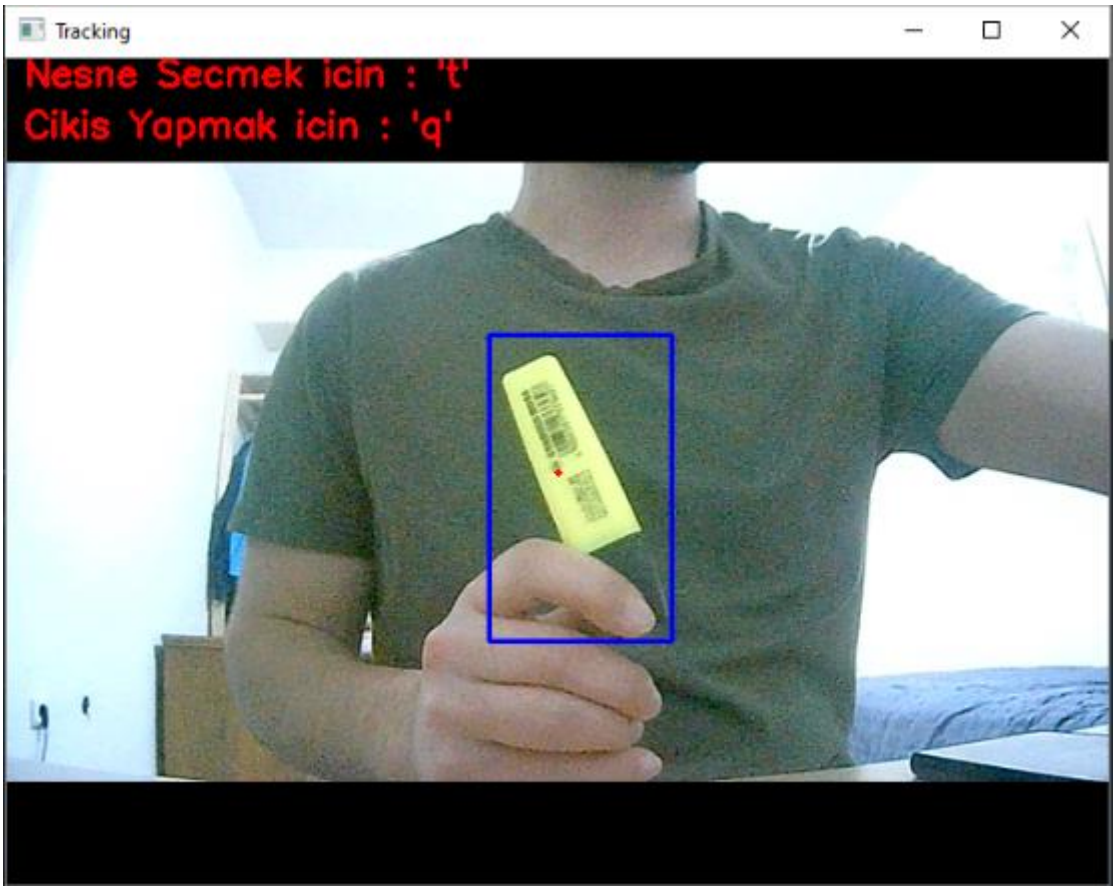
```
In [2]: runfile('C:/Users/Can Omer/Desktop/Tech/Python/Selecting From Camera/
SelectAndTrackSAT.py', wdir='C:/Users/Can Omer/Desktop/Tech/Python/Selecting From Camera')

      1. CSRT
      2. KCF
      3. Boosting
      4. Mil
      5. TLD
      6. MedianFlow
      7. Mosse


Takip algoritmasını sec (1-7)   : 3
3 ['Boosting']
Cikis icin 'q'
```

Algoritma seçimi tamamlandıktan sonra kamera görüntü alma faaliyetine başlar ve alınan her görüntü "capture" değişkenine depolanır. Kullanıcı takip etmek istediği nesneyi seçmek için klavyeden "t" tuşuna basar. Seçme işlemi sırasında kullanıcının seçme işlemini gerçekleştirebilmesi için "capture" değişkenine depolanan son görüntü/frame kullanılır. Kullanıcı seçmek istediği nesneyi görüntü üzerinde bir dikdörtgen içerisine almak için faresini kullanır. Artık takip işleminin gerçekleşmeye başlaması için kullanıcının klavyeden herhangi bir tuşa basması yeterlidir.

```
(170.0, 78.0, 162.0, 240.0)
sol
yukari
108 94

(233.0, 76.0, 162.0, 240.0)
yukari
108 95

(296.0, 102.0, 162.0, 240.0)
sag
107 95

(351.0, 130.0, 162.0, 240.0)
sag
106 95

(402.0, 150.0, 162.0, 240.0)
sag
asagi
105 94
```

Program çalıştığı anda terminale verdiği çıktı





Takip işlemi başladığı zaman kullanılmakta olan takip algoritması, seçilen yani takip edilen objenin frame üzerindeki koordinatları ve servo motorların anlık konumu ekranda görüntülenir. Hem görüntü üzerindeki takip işleminin hem de fiziksel takip işleminin düzgün gerçekleşebilmesi için dikdörtgen içine alınan ve takip edilen nesnenin içine alındığı dikdörtgenin merkez noktası hem de ana frame'in yani çerçevenin merkez noktası hesaplanır.

Seçilen hedefi görüntünün ortasına almak için hedef ile görüntü merkezi arasındaki uzaklık belirlenir. Fiziksel takip işlemi Arduino'ya gönderilen anlık konum bilgileri sayesinde servoların dönüşü ile gerçekleşir. Seçilen hedefin ya da kameranın hareket etmesi durumunda Python üzerinden Arduino ile iletişime geçilir. Arduino kodlarıyla servo motorlar yardımıyla kamera iki eksen üzerinde hareket ederek seçilen hedefi, görüntünün ortasında tutarak takip etmeye çalışır. Eğer takip edilen nesne kadrajdan çıkarsa fiziksel takip işlemi cismin son konumuna kadar devam eder. Kadrajdan çıkan nesne tekrar aynı kadraja girene kadar fiziksel takip işlemi son konuma gitmeye devam eder.
Hedeflenen cismin takip işlemi bittikten sonra q tuşuna basarak program bitirilir.

Python Kodu

```python
# Kütüphanelerin import edilmesi

import cv2
import time
import serial
import struct

# %% Servo motor kontrolü için servoyu bağladığımız arduino ile programımız
#    arasında seri iletişim başlatır

seri = serial.Serial("COM3",9600)   # Arduino ile çalışmak için aç/kapa #
time.sleep(2)                       # Arduino ile çalışmak için aç/kapa #

# %% Takip algoritmalarımızı 'Sözlük' formatında tanımladık ;
# ## Sözlük
# - bir çeşit karma tablo türüdür
# - anahtar ve değer çiftlerinden oluşur, hemen hemen her variable türü
#   olabilir ama genelde sayılar veya dizilerdir
# - { "anahtar": değer }

# Python programlama dilinde Switch-Case yapısı bulunmaz, algoritmalar arasında
# seçim yapabilmek için kendi Switch-Case mantığımızı oluşturuyoruz, tek bir
# 'Sözlük' yapısı yeterliydi ancak programı son kullanıcı tarafından daha
# anlaşılır kılmak için 2. bir Sözlük yapısı daha oluşturduk

# Sözlük 1
OPENCV_OBJECT_TRACKERS = {"CSRT"      : cv2.TrackerCSRT_create,
                          "KCF"       : cv2.TrackerKCF_create,
                          "Boosting"  : cv2.TrackerBoosting_create,
                          "Mil"       : cv2.TrackerMIL_create,
                          "TLD"       : cv2.TrackerTLD_create,
                          "MedianFlow": cv2.TrackerMedianFlow_create,
                          "Mosse"     : cv2.TrackerMOSSE_create}

# Sözlük 2
TRACKERS_KEYS = { "1"    : "CSRT",
                  "2"    : "KCF",
                  "3"    : "Boosting",
                  "4"    : "Mil",
                  "5"    : "TLD",
                  "6"    : "MedianFlow",
                  "7"    : "Mosse"}

# Sözlük yapısı boş bırakılamıyor, bu sebepten default bir anahtar verdik
tracker_name = ["Boosting"]
# Track (Takip) işleminin gerçekleştirileceği algoritmanın temsili
tracker = OPENCV_OBJECT_TRACKERS[tracker_name[0]]()
choice = 0

print('''
      1. CSRT
      2. KCF
      3. Boosting
      4. Mil
      5. TLD
      6. MedianFlow
      7. Mosse
   ''')

```

```
# %% Program başladığında algoritmalar arasından seçim yaptıran fonksiyon
def algoritma(TRACKERS_KEYS, OPENCV_OBJECT_TRACKERS):
    choice = input("Takip algoritmasını sec (1-7)   : ") # klavye değer girişi
    # ilk etapta kolaylık yaratması açısından 2. sözlüğü kullanacağız ve
    # anahtar değerleri "keys" içerisinde tutulacak
    keys = TRACKERS_KEYS.keys()
    if choice in keys: # Eğer klavyeden girilen değer Sözlüğümüzün içinde ise;
        # 2. sözlükten anahtar değerine göre alınan ve aslında 1. sözlüğün
        # anahtarı olan değer
        tracker_name.append(TRACKERS_KEYS[choice])
        # Performansı etkilememesi için listeye 1 değer eklendikten sonra
        # önceki değeri siliyoruz
        tracker_name.remove(tracker_name[0])
        print(choice, tracker_name) # Seçimimizi görelim
        # 1. Sözlüğe göre alınan anahtar ile asıl değerimize ulaştık
        tracker = OPENCV_OBJECT_TRACKERS[tracker_name[len(tracker_name)-1]]()

    else: # Eğer klavyeden girilen değer 2. sözlüğün anahtarları içinde değilse
        print("! Gecersiz secim ! Varsayilan Takip Algoritmasi {}".format(tracker_name))
    print("Cikis icin 'q'")

# yazdığımız fonksiyonu koşturalım
algoritma(TRACKERS_KEYS, OPENCV_OBJECT_TRACKERS)

# kameradan frame yakalama ve bunları "cap" de depolama
# (x) ; Dahili kamera icin x = 0 ; Harici kamera icin x = COMx
cap = cv2.VideoCapture(1)

xlis=[90] # servo fonksiyonu için ; motorlarımızın başlangıç açı değeri ;
ylis=[90] # servo fonksiyonu için ; motorlarımızın başlangıç açı değeri ;
# bunu birnevi motorların son konumunu hatırlaması için yaptık

success, img = cap.read() # "cap" içine frame depolanıyorsa "success" ve "img" değeri dönder
```

```
#%% Obje sınırlarını belirtmek ve kutucuk çizmek için oluşturduğumuz fonksiyon
# seçtğimiz objenin sınırlarını, merkez noktasını ve aldığımız framelerin
# merkez noktasını belirtmek için
def drawBox(img, bbox):
    # bbox değerlerini kolaylık yaratması için isimlendirdik
    x, y, w, h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    center_x = int(x+w/2) # objemizin x eksenine göre merkezi
    center_y = int(y+h/2) # objemizin y eksenine göre merkezi
    # A noktasından B noktasına (x,x,x) renginde dikdörtgen çiz ; obje sınır
    cv2.rectangle(img, (x,y), ((x+w), (y+h)), (255,0,255), 3, 1)
    # Obje merkezine (x,x,x) renginde çember/nokta çiz
    cv2.circle(img, (center_x, center_y), 2,(0,0,255),-1)
    # A noktasına x yazı tipinde "blabla" yaz
    cv2.putText(img,"Object At :",(10,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
    # A noktasına x yazı tipinde x değerini yaz
    cv2.putText(img,"X ="+str(center_x),(140,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
    cv2.putText(img,"Y ="+str(center_y),(240,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
    cv2.putText(img,"Algoritma : "+tracker_name[0],(400,15),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0
    # A noktasından B noktasına (x,x,x) renginde çizgi; Ana Frame merkezimizden Obje merkez n
    cv2.line(img, (320, 240), (center_x, center_y), (255,0,255), 1)
```

```python
# %% Servo motor kontrolü için oluşturduğumuz fonksiyon
def servo(bbox): # Servo motorlarımıza gönderilecek olan komutlar
    x, y, w, h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    center_x = int(x+w/2)
    center_y = int(y+h/2)
    print(" ")
    print(bbox)

    # Ana Frame ekranımızda merkez noktamız (320, 240),
    # o halde kamera merkezini obje merkezine yakınsak oranda ayarlamak için ;

    # obje x merkezi ana frame x ekseni 300-340 arasında ise son konum değeri
    if center_x < 300 and center_x > 340:
        print("kilitlendi_x")
        # bulunduğu açı kaç derecede ise listeye onu ekle
        xlis.append(xlis[len(xlis)-1])
        xlis.remove(xlis[0]) # ilk elemanı/önceki elemanı sil

    elif center_x > 340: # obje kameranın solunda ise
        print("sag")
        # motoru sola döndürmek için bulunduğu son konum açısını azalt
        xlis.append(xlis[len(xlis)-1]-1)
        xlis.remove(xlis[0])

    elif center_x < 300: # obje kameranın sağında ise
        print("sol")
        # motoru sağa döndürmek için bulunduğu son konum açısını arttır
        xlis.append(xlis[len(xlis)-1]+1)
        xlis.remove(xlis[0])

    # listenin son elemanının değerini "xdeg" değişkenine ayarla
    xdeg=xlis[len(xlis)-1]

    if center_y < 220 and center_y > 260:
        print("kilitlendi_y")
        ylis.append(ylis[len(ylis)-1])
        ylis.remove(ylis[0])

    elif center_y > 260:
        print("asagi")
        ylis.append(ylis[len(ylis)-1]+1)
        ylis.remove(ylis[0])

    elif center_y < 220:
        print("yukari")
        ylis.append(ylis[len(ylis)-1]-1)
        ylis.remove(ylis[0])

    ydeg=ylis[len(ylis)-1]

```

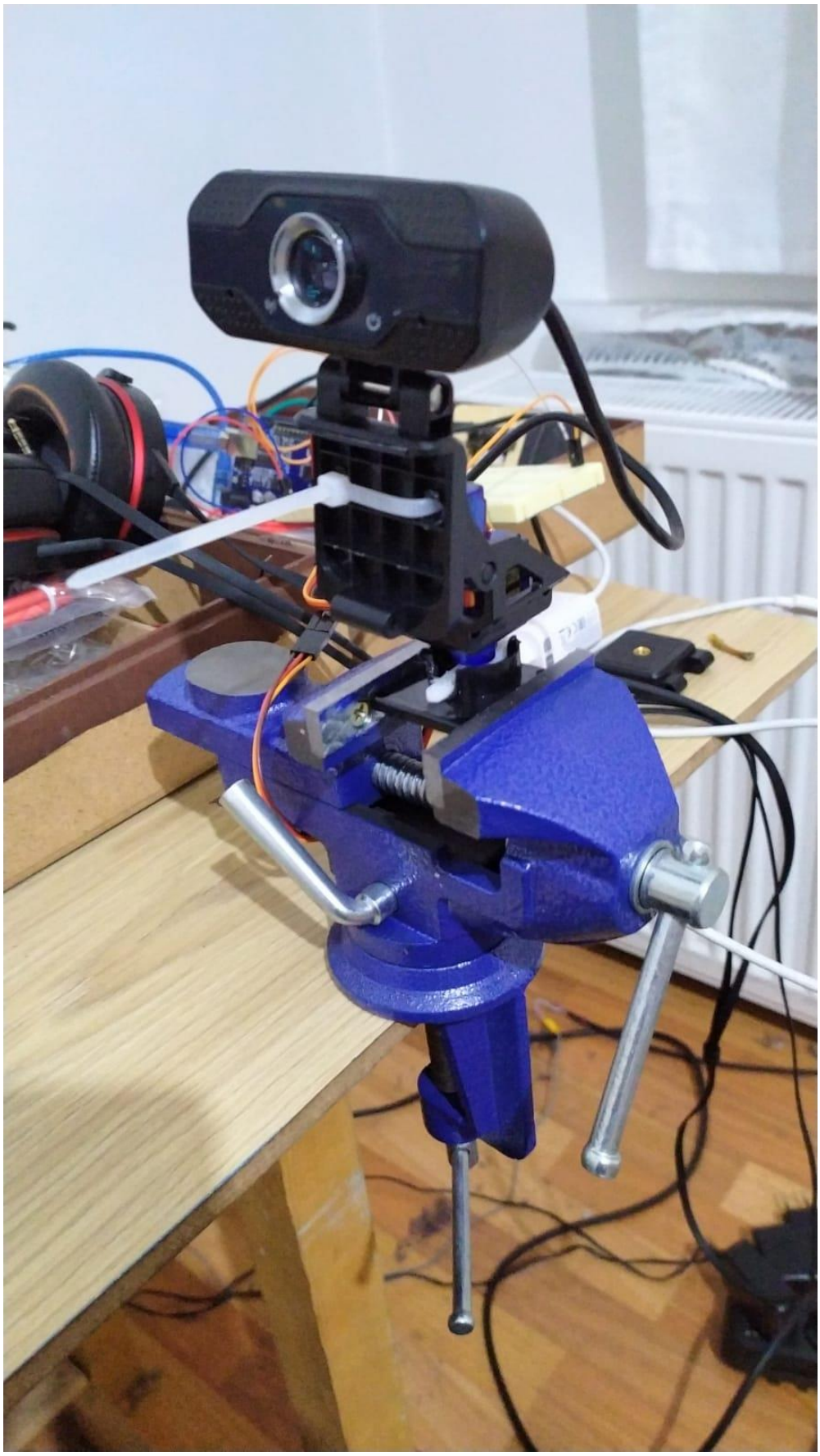
```
    # Seri iletişimde motora gönderilen PWM sinyali 0-255 arasında olduğundan
    # 0 dan az 255 den fazla değeri dönderemeyiz.
    # Ayriyeten elimizdeki servo motorların dönüş açı sınırları 0-180 derece
    # olduğundan 0 ve 180 derece sınırlarını da aşamazlar.
    # Bu nedenle oluşturduğumuz listelerdeki son konumlar 0 dan düşük
    # 180 den fazla olmamalı :
    if xdeg >= 180:
        xlis.append(xlis[len(xlis)-1]-1)
        xlis.remove(xlis[0])
        print("x ekseni sınır acısı 180 derece")
    elif xdeg <= 0:
        xlis.append(xlis[len(xlis)-1]+1)
        xlis.remove(xlis[0])
        print("x ekseni sınır acısı 0 derece")
    if ydeg >= 180:
        ylis.append(ylis[len(ylis)-1]-1)
        ylis.remove(ylis[0])
        print("y ekseni sınır acısı 180 derece")
    elif ydeg <= 0:
        ylis.append(ylis[len(ylis)-1]+1)
        ylis.remove(ylis[0])
        print("y ekseni sınır acısı 0 derece")

    print(xdeg,180-ydeg) # dönderdiğimiz dereceler
    # A noktasına x yazı tipinde x değerini yaz
    cv2.putText(img,"derece X ="+str(xdeg),(50,465),cv2.FONT_HERSHEY_SIMPLEX,0.
    cv2.putText(img,"derece Y ="+str(ydeg),(250,465),cv2.FONT_HERSHEY_SIMPLEX,0

    # Arduino ile çalışmak için aç/kapa #
    seri.write(struct.pack('>BB', xdeg,ydeg))
```

```python
# %% Kameradan görüntü alma "success" değerini dönderdiği sürece ;
while True:
    timer = cv2.getTickCount()
    success, img = cap.read()
    success, bbox = tracker.update(img) # yeni frame geldikçe güncelleme
    # (fps değeri buradan ölçülebilir)

    # Ana frame boyutumuzu/pixel miktarını ve buna bağlı olarak
    # koordinatlarımızı öğrenmek için "img" in shape değerini ekrana basalım
    # print(img.shape)

    # Ana frame merkezine nokta/çember çizdirme
    center = cv2.circle(img, (320, 240), 2,(0,0,255),-1)

    if success: # eğer seçim success değeri dönderirse
        # Obje sınırlarını belirtme fonksiyonu ^
        drawBox(img, bbox)
        # Servo motor kontrolü için gerekli talimatların dönderildiği fonksiyon
        servo(bbox)

    else: # eğer seçim success değeri döndermediyse
        cv2.putText(img,"Nesne Secmek icin : 't'",(10,15),cv2.FONT_HERSHEY_SIMP
        cv2.putText(img,"Cikis Yapmak icin : 'q'",(10,45),cv2.FONT_HERSHEY_SIMP

    cv2.imshow("Tracking",img) # framelerimizin görselleştirilmesi

    key = cv2.waitKey(1) & 0xFF

    # "t" tuşuna basıldığında seçme işlemi talimatlarını izle
    if key == ord("t"):
        bbox = cv2.selectROI("Tracking", img, False)
        tracker.init(img,bbox) # seçilen kısım img ve bbox değerini döndericek

    # "q" tuşuna basıldığında döngüyü kır
    elif key == ord("q"):
        break
# %% Döngü bittiyse tüm pencereleri kapa
cap.release()
cv2.destroyAllWindows()
```

Arduino Kodu

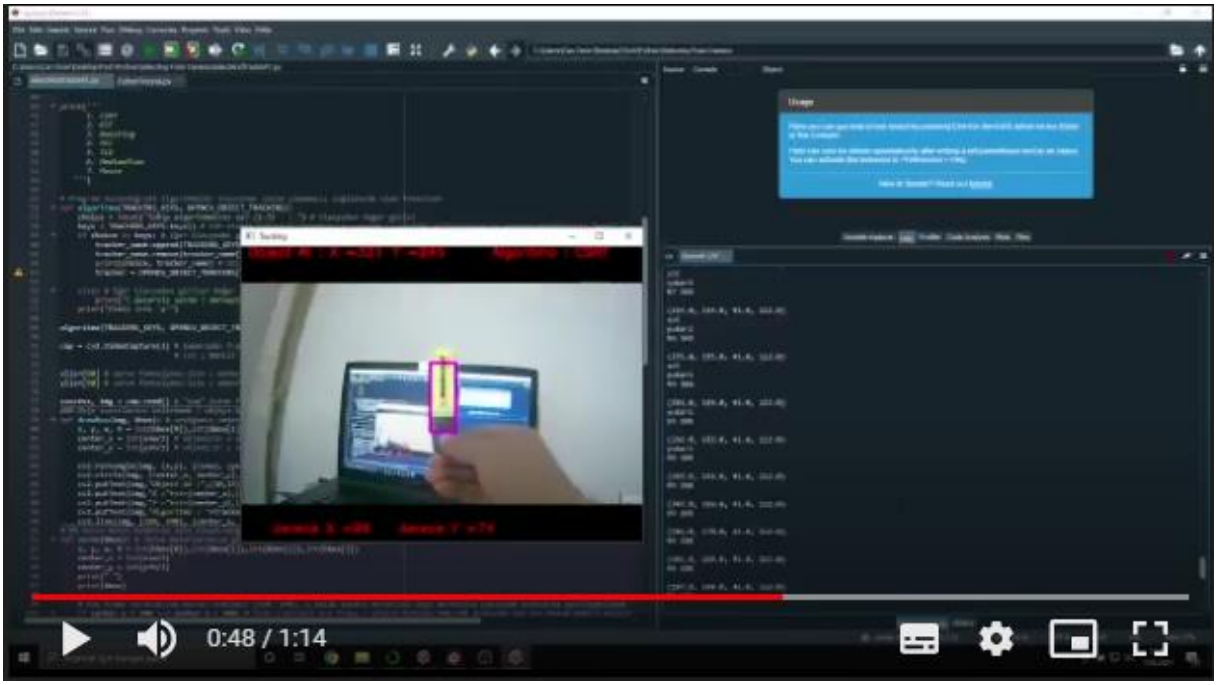
```
#include <Servo.h>
// int data_x1 = 0;
// int data_y1 = 0;
int data[1];

Servo myservo_x1;
Servo myservo_y1;// Servo motor kontrolü için bir servo objesi oluşturuyoruz

void setup() {
  Serial.begin(9600);
  myservo_x1.attach(9);
  myservo_y1.attach(10);
  myservo_x1.write(90);
  myservo_y1.write(90);
}

void loop() {
  while (Serial.available() >= 2) {
    //    data_x1=Serial.read();
    //    data_y1=Serial.read();
    for (int i = 0; i < 2; i++) {
      data[i] = Serial.read();
    }
    myservo_x1.write(data[0]);
    myservo_y1.write(data[1]);
    Serial.println(data[0]);
    Serial.println(data[1]);
  }
}
```

Bitmiş Prototip



Yapmış olduğumuz prototipin çalışma esnasında çekilmiş videoları

https://drive.google.com/file/d/1cEfQJ0En3jDUYZlLh6bAGXaDWkooeqkt/view



https://drive.google.com/file/d/1cFODFqn3Lrw2_SRPcbI7_Z07xWquXr0F/view