



#MILLİ  
TEKNOLOJİ  
HAMLESİ



# SÜRÜ İHA YARIŞMASI

## KRİTİK TASARIM RAPORU ŞABLONU

TAKIM ADI: COMBINE

BAŞVURU ID: 3620387

TAKIM ID: 742853



## İÇİNDEKİLER

<b>1. YÖNETİCİ ÖZETİ</b>	<b>3</b>
<b>2. PROJE GEREKSİNİMLERİ VE YÖNETİMİ</b>	<b>4</b>
Formasyon ve Navigasyon Gereksinimleri	5
Geçiş Sürecinde Yapılan Geliştirmeler	6
<b>3. TASARIM ÇÖZÜMÜ VE YAZILIM ARAYÜZLERİ</b>	<b>9</b>
3.1 Algoritma ve Yazılım	9
3.2 Görevlere ait Çözümler	10
3.3 İHA'lar Özellikleri (Mekanik, Elektronik ve Entegrasyon)	10
<b>4. TEMEL GÖREV İSTERLERİNİN DOĞRULANDIĞININ GÖSTERİLMESİ</b>	<b>11</b>
4.1 Formülizasyon	11
4.2 Formasyon ve Uçuş	15
4.3 Pseudo kod	17
4.4 Simülasyon	24
<b>5. FORMAT VE KAYNAKÇA</b>	<b>25</b>

## 1. YÖNETİCİ ÖZETİ

"Bir Sürü İHA" projemiz, Kritik Tasarım Raporu (KTR) aşamasında bulunmakta olup, Ön Tasarım Raporu (ÖTR) sürecinin ardından kayda değer teknik ilerlemeler gerçekleştirmiştir. Proje ekibimiz, yalnızca yarışma hedefleri doğrultusunda değil, aynı zamanda bilimsel literatüre katkı sağlama amacıyla çok sayıda özgün metodoloji geliştirmiş ve bu yaklaşımları simülasyon ortamında başarıyla uygulamıştır.

Geliştirilen tüm yaklaşımlar sistematik bir şekilde dokümente edilmiş olup, hibrit sürü zekası kapsamında hazırlanan raporun yanı sıra akademik makale çalışmaları da devam etmektedir. Projeye ilişkin kaynak kodlar, karşılaşılan teknik sorunlar ve çözüm yöntemleri ile ilgili tanıtım videolarının tamamına belirtilen GitHub deposu üzerinden erişim sağlanabilmektedir.

### Özgünlük olarak benimseyip icra ettiğimiz Başarılarımız:

- **Hibrit kontrol mimarisi** merkezi ve merkezi olmayan tam olarak implement edildi ve test edildi
- **Gelişmiş TCP protokolü** drone'lar arası iletişim için optimize edildi
- **ArUco marker tespiti** A\* algoritması ile entegre edildi

### Özgün Katkılarımız:

- Merkezi ve merkezi olmayan kontrol mekanizmalarının dinamik geçişi
- Sezgisel A\* algoritması ile keşif optimizasyonu
- TCP tabanlı handshake - sürü haberleşme protokolü

**Hibrit kontrol mimarisi** için durum bazlı dinamik geçiş algoritması geliştirilerek normal operasyonlarda merkezi kontrolün koordinasyon avantajları, acil durumlarda ise merkezi olmayan kontrolün hızlı tepki kabiliyeti bir arada kullanılabilir hale getirilmiştir.

**Özelleştirilmiş TCP protokolü:** Drone'lar arası haberleşme için TCP tabanlı bir protokol geliştirilmiştir. Bu protokol, handshake mekanizması ile sürü üyeleri arasında bağlantı kurulmasını ve mesaj sıralamasını sağlamaktadır. Merkezi olmayan modda çalışan drone'ların birbiriyle koordineli çalışabilmesi için temel haberleşme altyapısını oluşturmaktadır.

**Geliştirdiğimiz A tabanlı arama algoritması:**\* ArUco marker tespiti için A\* algoritması kullanılmıştır. Algoritma, drone'ların görüntü işleme verilerini kullanarak hedef noktalara yönlendirme yapmaktadır. BFS ve DFS gibi alternatif arama yöntemleri yerine A\*'ın tercih edilmesinin nedeni, heuristik (sezgisel) yaklaşımının sağladığı hesaplama avantajıdır. Uygulamada elde edilen sonuçlar raporda sunulmuştur.

**Anahtar Kelimeler & Projede Kullanılan Teknolojiler:** ROS2, Ubuntu 24.04, Gazebo11, ArduPilot SITL, PyMAVLink, Hibrit Sürü Kontrolü, Unity

## 2. PROJE GEREKSİNİMLERİ VE YÖNETİMİ

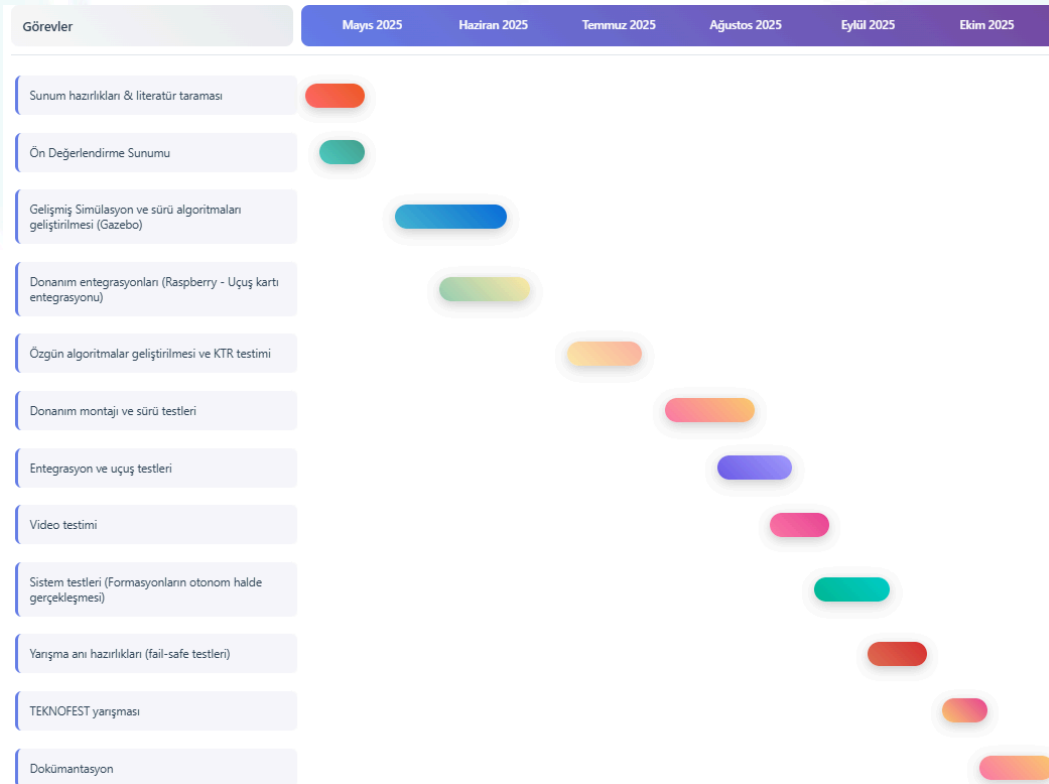
**Göksel Gündüz (Takım Kaptanı):** Bilgisayar mühendisi, yapay zeka uzmanı. Takım liderliği, Sürü zekası algoritmaları, genel koordinasyon, formasyon kontrol algoritmaları ve ArUco marker görüntü işleme görevlerinden sorumlu.

**Aysima Şentürk (Modelleme & Malzeme):** Matematiksel modelleme, formülasyon, Malzeme temini, Drone formasyon hatalarında geçiş stratejileri görevlerinden sorumlu.

**İrfan Gümüş (Donanım):** Drone donanım seçimi, elektronik entegrasyon, sensör kalibrasyonu ve batarya yönetimi görevlerini üstleniyor.

**Gökhan Büyük (Navigasyo):** Navigasyon, yol planlama ve Waypoint algoritmaları geliştirme görevlerinden sorumlu.

**Zehra Kaya (Yazılım):** Simülasyon geliştirme (Unity/Gazebo), yer kontrol istasyonu, arayüz tasarımı görevlerini üstleniyor.



### Güncel Yarışma Takvimi

KTR raporunda, yarışma şartnamesinde belirlenen tüm hususları kapsayacak şekilde gereksinimler oluşturulmuştur. Görev odaklı isterler (**Formasyon & Navigasyon Gereksinimleri, Haberleşme Gereksinimleri ve Donanım gereksinimleri** olarak ele alınmıştır.

## Formasyon ve Navigasyon Gereksinimleri

❖ **Göksel Gündüz ve Zehra Kaya** işbirliği ile geliştirilen simülasyon sistemi, şartnamede belirtildiği üzere jüri tarafından verilen değişkenleri işleyebilme kapasitesine sahiptir. [arayüz sağ görsel] Süreç dahilinde sürü drone'ların implementasyonu için Gazebo + SITL[4,5] ortamı ele alınmış olup, istenilen performans ve çıktı elde edilememesi, kütüphane ve ROS2-ROS1 uyumsuzlukları sebebiyle Unity ortamı tercih edilmiş, simülasyon C# dilinde icra edilmiştir.

Sistem, çizgi formasyonları (yatay ve dikey), V formasyonu ve ok başı formasyonlarını koruyarak uçabilme yeteneğini desteklemektedir (**Şartname 5.1 - 3B Formasyon Görevi**).

Bu kapsamda, belirlenen irtifada (Z metre) ve drone'lar arası mesafede (X metre) formasyon korunabilmesi gereksinimi karşılanmıştır (**Şartname Tablo 3**).

❖ **Gökhan Büyük ve Göksel Gündüz** tarafından ortak geliştirilen navigasyon sistemi ile İHA'ların belirtilen noktalara tam otonom şekilde navigasyon yapabilme yeteneği sağlanmıştır (**Şartname Ek bilgiler madde 4**).

❖ İHA'ların birbirleriyle çarpışmadan formasyon değiştirebilmesi için Temizer ve arkadaşlarının çalışmasından[2,3] esinlenilerek çarpışma önleme algoritması geliştirilmiş ve çalışmada ilham alınmıştır.

## Haberleşme Gereksinimleri

❖ **Göksel Gündüz** tarafından geliştirilen İHA'lar arası TCP handshake tabanlı iletişim altyapısı, sürü üyeleri arasında veri alışverişini sağlamaktadır (**Şartname 5.6 Görev Ortamı ve Güvenlik**). Bu protokol, drone'ların merkezi olmayan modda çalışırken de koordinasyonu sürdürebilmesini mümkün kılmaktadır.

❖ Gökhan Büyük ve Zehra Kaya, geliştirdikleri fail-safe mekanizmaları sayesinde, yer kontrol istasyonu ile haberleşme bağlantısı kesildiğinde dahi İHA'ların görevlerini sürdürebilme kapasitesi sağlanmıştır (**Şartname 5.2 Sürü Halinde Navigasyon Görevi - Madde 6**). Bu özellik, sistemin otonom çalışma kabiliyetini güçlendirmektedir.

## Donanım Gereksinimleri

İrfan Gümüş tarafından belirlenen donanım bileşenleri rapor kapsamında sunulmuş olup, proje ile ilgili tüm teknik dokümantasyon, simülasyon dosyaları ve kod tabanı Göksel Gündüz koordinatörlüğünde GitHub deposunda [1] paylaşılmıştır.

Gümüş'ün önerisi ile her bir drone için **Raspberry Pi 4** tabanlı işlem birimi kullanılmış olup, Gündüz tarafından geliştirilen sürü zekası algoritmaları ve otonom görev yürütme işlemleri bu merkezi birimden kontrol edilmektedir.

Birçok mod, uçuş tipi, araç seçimi gibi seçeneklerin bulunduğu bir kumanda olan **“Flysky Fs-i6 2.4Ghz 6 Kanallı Uzaktan Kumanda Set”** kumandası temin edilmiştir. (Drone'ların yarışma finalinde otonom uçuş yapacağı bilinmekle beraber, belirli bir kalifiyeye getirilmesi sürecinde kumanda temin edilmiştir)

Her bir drone için 802 KV değerinde **motor**, **keşif** görevi için raspberry **kamera**, bununla birlikte enerji kaynağı olarak **3300 mAh 7.7v 2s 30C Lipo batarya** temin edilmiştir.

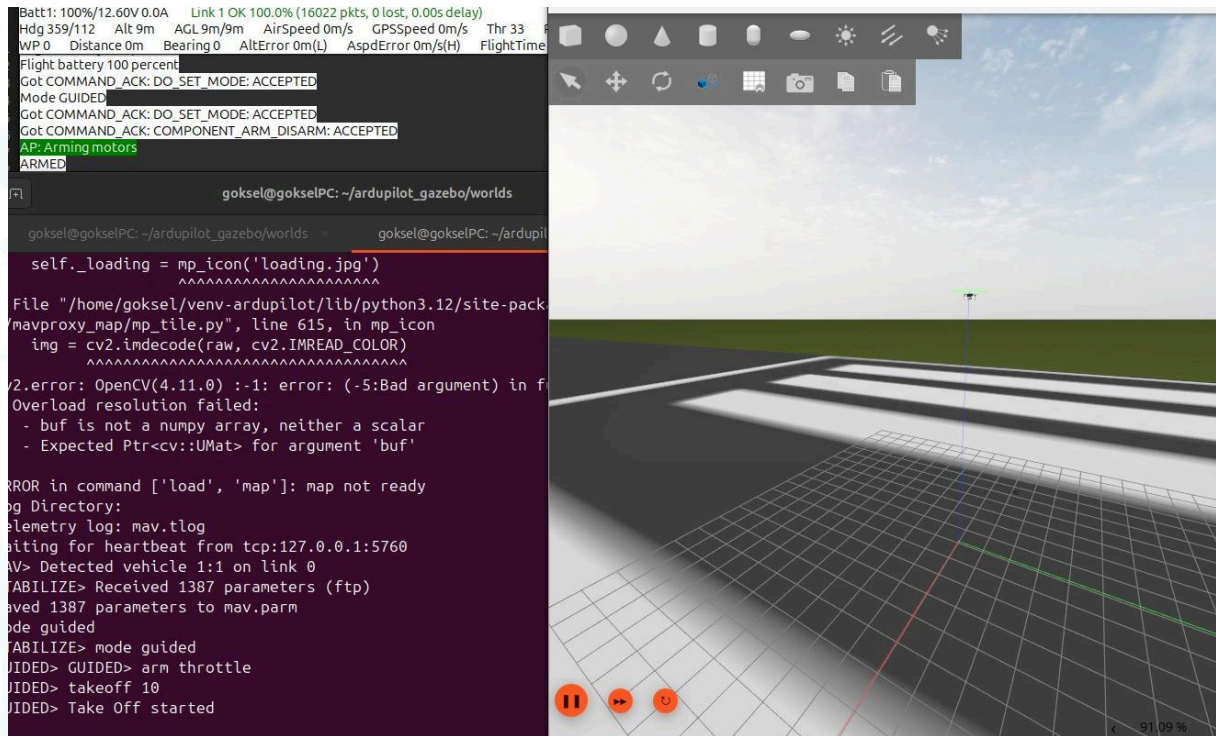
Uçuş kartı için **GEPRC TAKER F405 BLS 50A STACK Uçuş Kontrol Kartı** tercih edilmiştir. Bu kartın seçilmesindeki temel avantajlar şunlardır:

- Entegre 50A ESC'ler sayesinde ek ESC bileşenlerine ihtiyaç duyulmaması
- F405 işlemci ile sağlanan yüksek hesaplama kapasitesi
- Kompakt tasarım ile optimum ağırlık ve hacim oranı
- Benzer kategorideki donanımlara kıyasla uygun maliyet yapısı

## 2.2 OTR'den KTR'ye Geçiş Sürecinde Yapılan Geliştirmeler

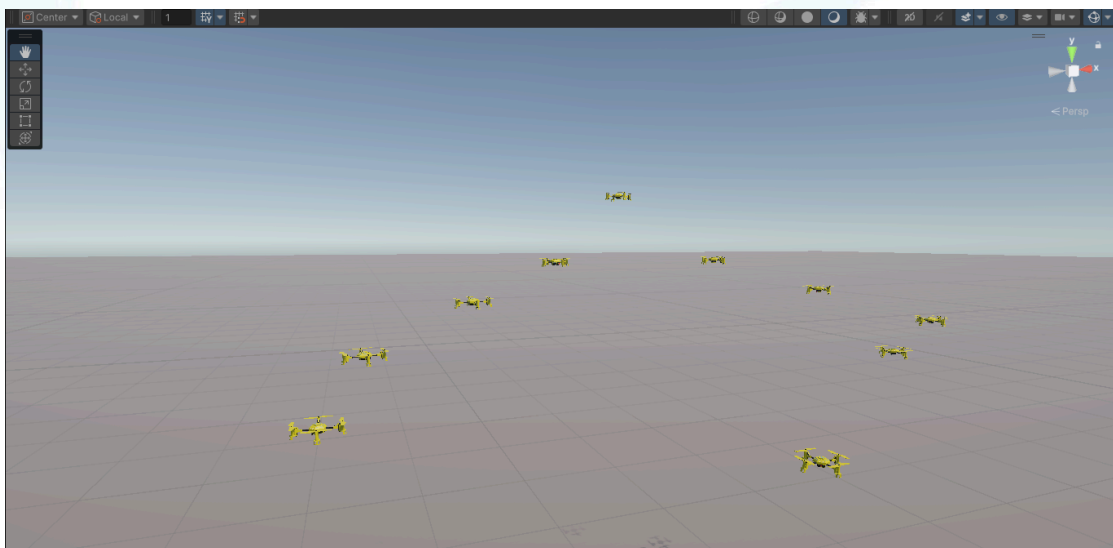
**Simülasyon Ortamı ve Ajan Geliştirme:** ÖTR aşamasında kullanılan Pygame tabanlı simülasyon ortamı, Zehra Kaya ve Göksel Gündüz işbirliği ile daha kalifiye çevre ve yazılım koşulları sağlamak amacıyla SITL + Gazebo ve ardından Unity platformu ile değiştirilmiştir.





### Görsel 1. Gazebo ortamı ve SITL entegrasyonu

Süreç dahilinde sürü drone'ların implementasyonu için Gazebo + SITL ortamı ele alınmış olup, istenilen performans ve çıktı elde edilememesi, kütüphane ve ROS2-ROS1 uyumsuzlukları, sürü algoritmalarının istenilen düzeyde implemente edilememesi sebebiyle Unity ortamı tercih edilmiş, simülasyon C# dilinde icra edilmiştir.



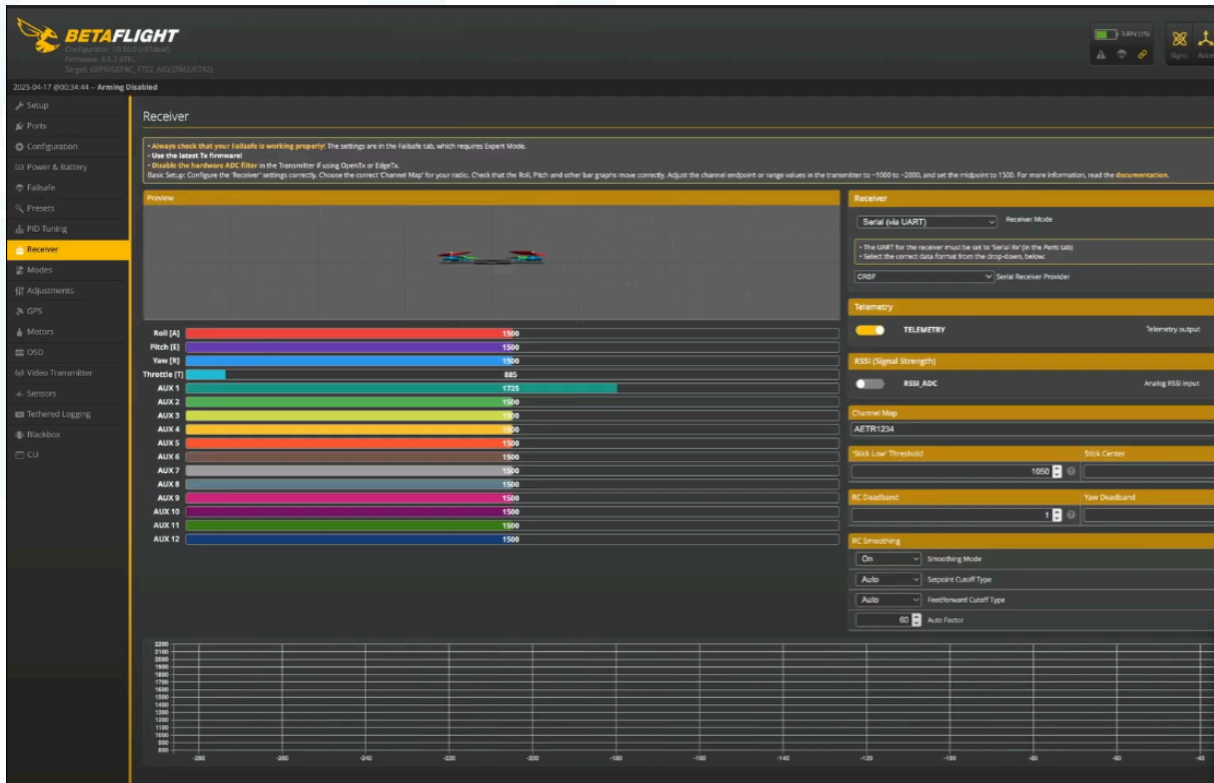
## Görsel 2. Geliştirilen Unity ortamı

**Hibrit Kontrol Mimarisi:** Göksel Gündüz ve Aysima Şentürk tarafından tasarlanan kontrol mimarisi, teorik tasarım aşamasından tam işlevsel implementasyona geçirilerek test edilmiştir. Bu sistem, merkezi ve merkezi olmayan kontrol yöntemleri arasında dinamik geçiş yapabilme kabiliyeti sağlamaktadır.

**Görüntü İşleme Sistemi:** Geliştirilen ArUco marker tespit sistemi simülasyon ortamında çalıştırılarak algoritma performansı doğrulanmıştır. Bu sistem, keşif görevlerinde kritik rol oynamaktadır.

### KTR Döneminde Gerçekleştirilen Fiziksel Drone İnşası ve Testleri:

KTR aşamasında simülasyon çalışmalarına paralel olarak, takım fiziksel drone prototipi geliştirme sürecini başlatmıştır. Belirtilen komponentler temin edilerek drone montajı tamamlanmış ve Betaflight Configurator ile sistem konfigürasyonu gerçekleştirilmiştir.



**Görsel 3: Betaflight Programı Arayüzü**

**Kumanda-Drone İletişim Kurulumu:** Takım üyesi **Zehra Kaya** tarafından Flysky FS-i6 2.4GHz kumanda sistemi ile PPM (Pulse Position Modulation) protokolü üzerinden drone bağlantısı kurulmuştur. Takım kaptanı **Göksel Gündüz** ise Betaflight programını kullanarak sistem analizini yürütmüştür.



**Karşılaşılan Teknik Sorun ve Çözümü:** Test aşamasında kritik bir kanal haritalama sorunu tespit edilmiştir. Kumanda stick hareketleri yanlış eksenlerde algılanmış (sağ-sol → yukarı-aşağı, yukarı-aşağı → sağ-sol), kablo bağlantıları ve receiver ayarları detaylı kontrol edilmiştir. Sorunun kaynağı Receiver sekmesindeki **TAER1234** kanal haritası olduğu tespit edilmiş, **AETR1234** konfigürasyonuna geçilmesiyle problem çözülmüştür.

#### **Sistem Optimizasyon Parametreleri:**

- **PWM Sinyal Aralığı:** 1000-2000µs endpoint konfigürasyonu
- **Kalibrasyon Hassasiyeti:** ±150µs trim ayarları
- **Titreşim Eliminasyonu:** %3 deadband threshold (~30µs)
- **Kontrol Karakteristiği:** %15 expo curve (düşük stick hassasiyeti)
- **Dönüş Hızı Limiti:** 600°/s maksimum angular velocity

**Motor Sistemi Entegrasyonu:** Takım üyeleri **Gökhan Büyük** ve **İrfan Gümüş** koordinasyonunda GEPRC F405 flight controller ile motor sistemi entegrasyonu tamamlanmıştır. UART1 portu üzerinden MAVLink telemetri protokolü aktif edilmiş, ESC motor kalibrasyonu 1000-2000µs PWM bandında optimize edilmiştir.

### **3. TASARIM ÇÖZÜMÜ VE YAZILIM ARAYÜZLERİ**

#### **3.1 Algoritma ve Yazılım**

Geliştirdiğimiz sistem, TEKNOFEST yarışma gereksinimlerini karşılamak üzere modüler ve ölçeklenebilir bir mimari ile tasarlanmıştır. Sistem, Unity tabanlı simülasyon ortamında C# programlama dili kullanılarak geliştirilmiş olup, gerçek zamanlı parametre güncellemesi ve dinamik drone sayısı adaptasyonu özelliklerine sahiptir.

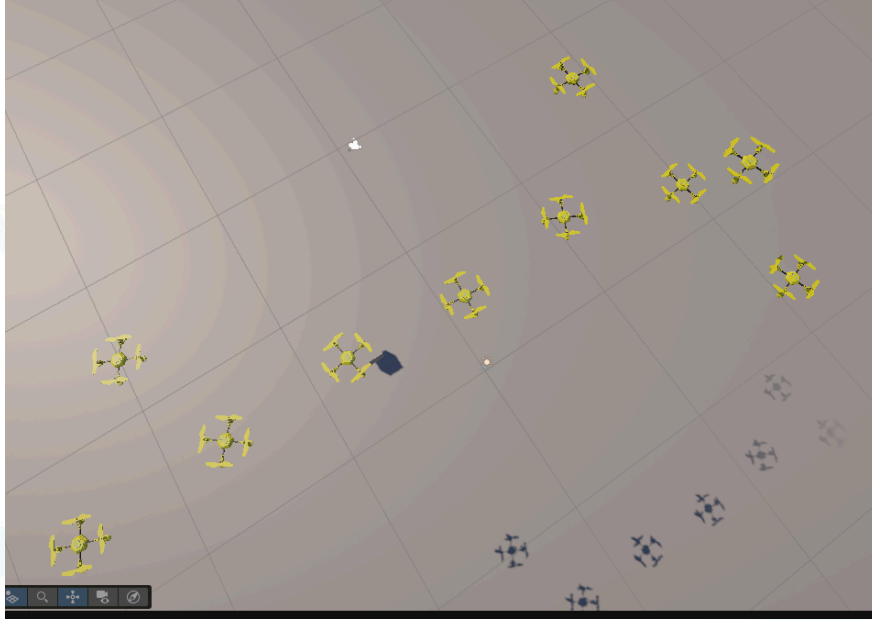
##### **Temel Sistem Mimarisi**

**DroneSpawner (Ana Kontrol Ünitesi):** Sistemin merkezi kontrol birimidir ve aşağıdaki temel işlevleri yerine getirir:

- TEKNOFEST şartname parametrelerinin yönetimi (Z: Uçuş İrtifası, T: Formasyon Koruma Süresi, X: Ajanlar Arası Mesafe)
- Runtime parametre güncelleme sistemi (**parametreleriGuncelle** flag'i ile)
- Dinamik drone sayısı adaptasyonu (1-50 drone arası ölçeklenebilirlik)
- Formasyon türlerinin koordinasyonu (Arrow, V, Line, Vertical)
- Akıllı kalkış sıralama algoritması (dış-iç sıralama mantığı ile çarpışma önleme)

## Çarpışma Önleme Algoritması

Sistemimizin özgün çarpışma önleme algoritması, gelecek pozisyon tahmini ve kuvvet vektörü hesaplama prensiplerine dayanmaktadır. Teknik detaylar **formülizasyon bölümünde verilmiştir.**



**Görsel 4. Çarpışma Önleme Denkleminin görülebildiği simülasyon görüntüsü**

## Merkezi İletişim ve Hibrit Kontrol Mimarisi

### DroneCommHub (İletişim Merkezi):

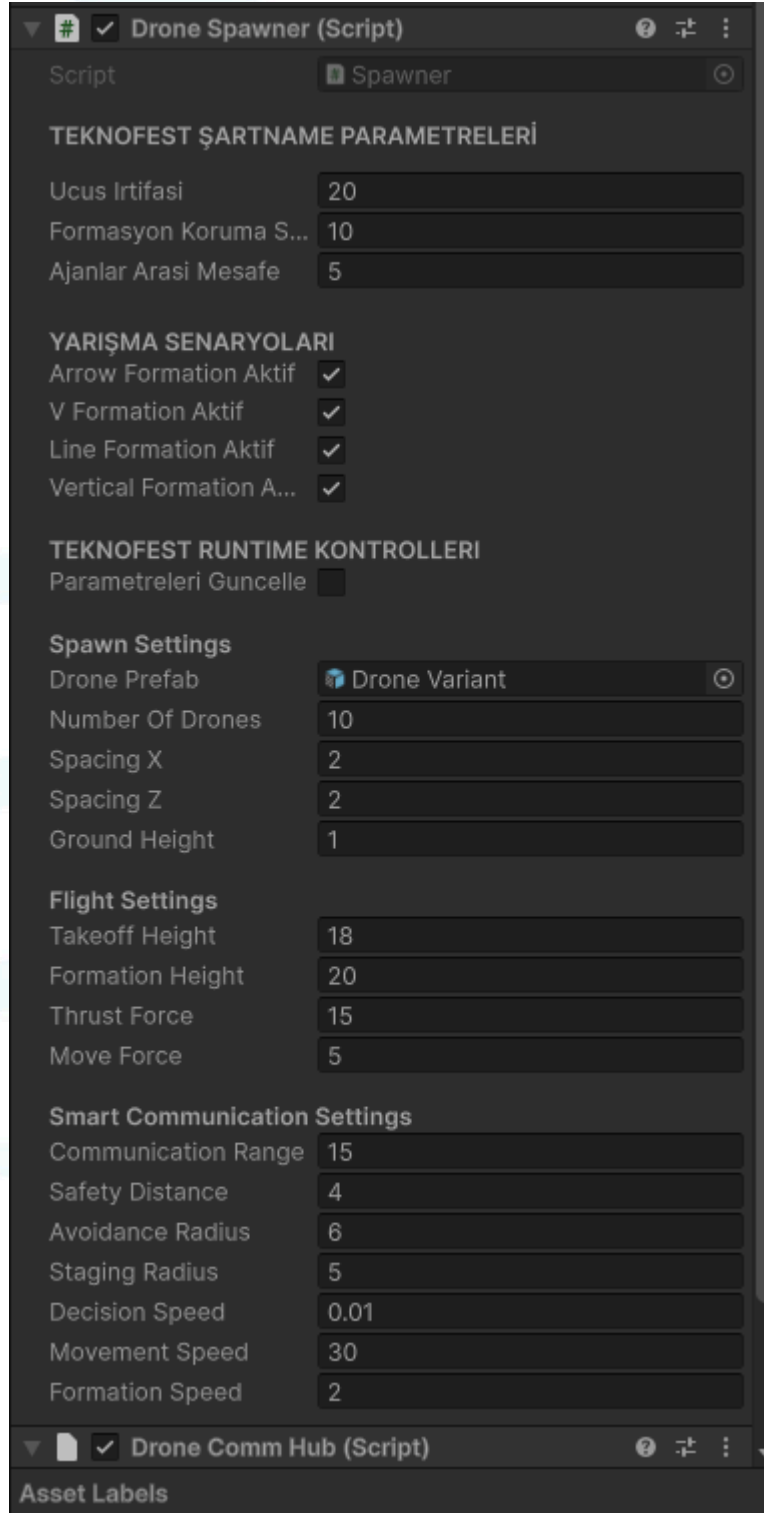
- TCP tabanlı handshake protokolü ile güvenilir veri iletişimi
- Merkezi ve merkezi olmayan kontrol modları arası dinamik geçiş
- Gerçek zamanlı drone durumu ve pozisyon paylaşımı
- Fail-safe mekanizması ile otomatik kontrol devralma

**Hibrit Kontrol Sistemi:** Normal operasyonlarda merkezi kontrolün koordinasyon avantajları ile acil durumlarda merkezi olmayan kontrolün hızlı tepki kabiliyetini birleştirir:

- **Merkezi Mod:** Genel koordinasyon, formasyon planlama, optimize edilmiş yol hesaplama
- **Merkezi Olmayan Mod:** Acil kaçınma, bağımsız karar verme, hızlı tepki
- **Dinamik Geçiş:** Tehlike seviyesi ve iletişim durumuna göre otomatik mod değişim

### 3.2 Görevlere ait Çözümler

Yarışma finalinde jüriden alınacak parametreler, görsel 5’de belirtilen arayüz üzerinden alınacak ve görevlerin gerçekleştirilmesi sağlanacaktır.



**Drone Spawner (Script)**

Script **Spawner**

**TEKNOFEST ŞARTNAME PARAMETRELERİ**

Uçus İrtifası	20
Formasyon Koruma S...	10
Ajanlar Arası Mesafe	5

**YARIŞMA SENARYOLARI**

Arrow Formation Aktif	<input checked="" type="checkbox"/>
V Formation Aktif	<input checked="" type="checkbox"/>
Line Formation Aktif	<input checked="" type="checkbox"/>
Vertical Formation A...	<input checked="" type="checkbox"/>

**TEKNOFEST RUNTIME KONTROLLERİ**

Parametreleri Güncelle ☐

**Spawn Settings**

Drone Prefab	Drone Variant
Number Of Drones	10
Spacing X	2
Spacing Z	2
Ground Height	1

**Flight Settings**

Takeoff Height	18
Formation Height	20
Thrust Force	15
Move Force	5

**Smart Communication Settings**

Communication Range	15
Safety Distance	4
Avoidance Radius	6
Staging Radius	5
Decision Speed	0.01
Movement Speed	30
Formation Speed	2

**Drone Comm Hub (Script)**

Asset Labels

Görsel 5: Görev öncesi ilgili parametrelerin verildiği arayüz



Navigasyon, birey ekleme çıkarma, formasyon görevleri için ilgili kodlar [github depomuzda bulunmaktadır.](#)

### 3.2.1 3B Formasyon Görevi Çözümü

#### Dinamik Formasyon Sistemi

Sistem, yarışma jürisi tarafından verilen parametrelere göre anlık formasyon oluşturma kapasitesine sahiptir:

- **Z Parametresi (Uçuş İrtifası):** Tüm formasyonlar jüri tarafından belirlenen irtifada gerçekleştirilir
- **X Parametresi (Ajanlar Arası Mesafe):** Formasyon içi drone mesafeleri dinamik olarak ayarlanır
- **T Parametresi (Formasyon Koruma Süresi):** Belirlenen süre boyunca formasyon hassas şekilde korunur

#### Desteklenen Formasyon Tipleri

1. V Formasyonu
2. Ok Baş Formasyonu
3. Çizgi Formasyonu

### 3.2.2 Sürü Halinde Navigasyon Görevi Çözümü

```
// Waypoint tabanlı navigasyon algoritması
public void NavigateToWaypoint(Vector3 targetWaypoint)
{
    Vector3 safeDirection = commHub.RequestSafePathVector(
        droneID,
        transform.position,
        targetWaypoint
    );

    rb.AddForce(safeDirection * moveForce);
}
```

## Çarpışma Önleme Mekanizması

- **Güvenlik Yarıçapı:** X parametresinin %80'i çarpışma önleme sınırı
- **Gelecek Pozisyon Tahmini:** 2 saniye ileri tahmin ile proaktif kaçınma
- **Kuvvet Vektörü Hesaplama:** Matematiksel olarak optimize edilmiş kaçınma kuvvetleri

## 3.2.4 ArUco Marker Keşif Görevi Çözümü

```
// Gelişmiş A* tabanlı marker arama algoritması
public Vector3 FindOptimalSearchPath(Vector3 startPos, List<Vector3> potentialMarkerLocat:
{
    // Heuristik fonksiyon: Mesafe + görüş açısı optimizasyonu
    float heuristic = Vector3.Distance(currentPos, markerPos) +
        CalculateViewAngleCost(currentPos, markerPos);

    return CalculateOptimalPath(startPos, bestMarkerLocation);
}
```

## Görüntü İşleme Sistemi

- **ArUco Marker Tespiti:** Gerçek zamanlı marker tanıma
- **Konum Hesaplama:** 3D pozisyon çıkarımı
- **Sürü Koordinasyonu:** Bulunan marker bilgisinin paylaşımı

### 3.3 İHA'lar Özellikleri (Mekanik, Elektronik ve Entegrasyon)

#### 3.3.1 Mekanik Tasarım ve Komponentler

KATEGORİ	ÖZELLİK	DEĞER
Gövde Yapısı	Malzeme	3K karbon fiber kompozit çerçeve
	Ağırlık	350 g
	Motor Çapı	250 mm
	Pervane	9x4.5 inch carbon fiber
	Batarya	2S LiPo 3300mAh 7.7V
Motor & Tahrik	Motor Tipi	820 KV BLDC brushless (4x)
	ESC	GEPRC F405 entegre 50A BLHeli_5
	Maksimum İtki	1200g/motor (4800g toplam)
İşlemci Sistemi	Ana İşlemci	Raspberry Pi 4 (4GB RAM)
	Uçuş Kontrolü	GEPRC F405 Flight Controller
	CPU	Dual-core ARM Cortex-A72 1.5GHz (Raspberry Pi)
Sensör Paketi	IMU	MPU6000 (6-axis gyro/accelerometer)
	Manyetometre	HMC5883L
	Barometre	BMP280 (irtifa ölçümü)
	GPS	u-blox M8N GPS modülü
İletişim	WiFi	2.4GHz 802.11n (sürü içi)
	Protokol	MAVLink UART/USB serial
	Kamera	480p Raspberry kamera

Görsel 6. Temin edilen komponentler ve özellikleri



## 4. TEMEL GÖREV İSTERLERİNİN DOĞRULANDIĞININ GÖSTERİLMESİ

### 4.1 Formülizasyon

#### Çarpışma Önleme Denklemi

$$F_{\text{avoid}} = \sum_{i=1}^N K \times (R_{\text{safe}} - d_i) / R_{\text{safe}} \times \hat{u}_i$$

- $F_{\text{avoid}}$ : Toplam kaçınma kuvveti vektörü (Newton)
- $K$ : Kaçınma kuvveti sabiti = 3.0
- $R_{\text{safe}}$ : Güvenlik yarıçapı = 4.0 metre
- $d_i$ :  $i$ . İHA'ya olan mesafe (metre)
- $\hat{u}_i$ :  $i$ . İHA'dan uzaklaşma yön vektörü (normalize)
- $N$ : Güvenlik yarıçapı içindeki İHA sayısı

```
Vector3 CalculateAvoidanceForce(Vector3 myPos, List nearbyDrones) {
    Vector3 totalForce = Vector3.zero; float K = 3.0f;
    // Kuvvet sabiti float R_safe = 4.0f;

    // Güvenlik yarıçapı
    foreach (DroneData drone in nearbyDrones) { float distance =
        Vector3.Distance(myPos, drone.position);

        // Kaçınma yönü hesaplama
        if (distance < R_safe && distance > 0.1f){
            Vector3 avoidDirection = (myPos - drone.position).normalized;

            // Formül uygulaması

            float forceMagnitude = K * (R_safe - distance) / R_safe;
            totalForce += avoidDirection * forceMagnitude;}}

    return totalForce; }
```

Parametre	Değer	Açıklama
<b>K</b>	3.0	Kaçınma kuvveti sabiti
<b>R<sub>safe</sub></b>	$X \times 0.8$	Güvenlik yarıçapı (Jüri ve X'e bağlı)
<b><math>\Delta t</math></b>	2.0 saniye	Gelecek pozisyon tahmin süresi
<b><math>\hat{u}_i</math></b>	Normalize vektör	Kaçınma yön vektörü

### V Formasyonu Geometrik Modeli

**Sol Kanat:**  $P_{left}(i) = (-X \times i \times 0.8, Z + 2.5 \times i, 0)$

**Sağ Kanat:**  $P_{right}(i) = (+X \times i \times 0.8, Z + 2.5 \times i, 0)$

**Merkez:**  $P_{center} = (0, Z, 0)$

- **X:** ajanlar Arası Mesafe (Jüri X parametresi)
- **Z:** ucuslartifasi (Jüri Z parametresi)
- **i:** Kanat pozisyon indeksi (1, 2, 3, ...)
- **0.8:** Yatay genişleme katsayısı
- **2.5:** Dikey yükseklik artış katsayısı

## Ok (Arrow) Formasyonu

$$\text{Uç Nokta: } P_{\text{tip}} = (0, Z + 6, 0)$$

$$\text{Kuyruk: } P_{\text{tail}} = (0, Z - 4, -N \times 2)$$

$$\text{Sol Kanat: } P_{\text{left}}(i) = (-X \times i, Z + 4 - (\text{step}_i \times 8), -(\text{step}_i \times \text{depth}))$$

$$\text{Sağ Kanat: } P_{\text{right}}(i) = (+X \times i, Z + 4 - (\text{step}_i \times 8), -(\text{step}_i \times \text{depth}))$$

$$\text{Burada: } \text{step}_i = i / (\text{wing\_count} + 1), \text{ depth} = N \times 1.5$$

- **N:** Toplam İHA sayısı
- **wing\_count:**  $(N-2)/2$  (uç ve kuyruk hariç)
- **step<sub>i</sub>:** Normalize edilmiş kanat adımı
- **depth:** Formasyon derinliği (İHA sayısına bağlı)

## Çizgi Formasyonu

$$P_{\text{vertical}}(i) = (0, Z + i \times \text{spacing}, 0)$$

$$\text{Burada: } \text{spacing} = X \times 0.6$$

İHA Pozisyonu	Yükseklik Formülü	Açıklama
En Alt İHA	$(0, Z, 0)$	Temel irtifa seviyesi
En Üst İHA	$(0, Z + (N-1) \times X \times 0.6, 0)$	Maksimum Yükseklik
Toplam yükseklik	$(N - 1) \times X \times 0.6 \text{ metre}$	Sütun toplam uzunluğu



## Çember Staging (Hazırlık Fazı) Hesaplama

Staging, drone'ların formasyon oluşturmada önce geçtiği hazırlık pozisyonları fazıdır.

$$P_{staging}(i) = (R \times \cos(\theta_i), h, R \times \sin(\theta_i))$$

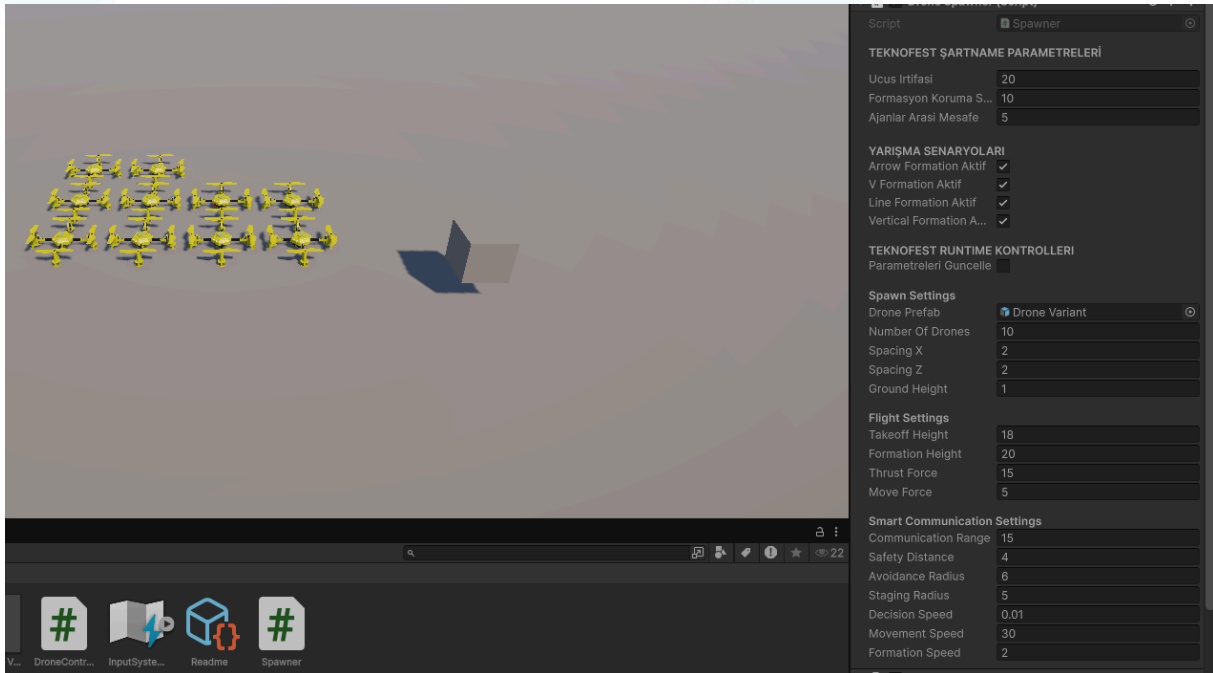
$$\theta_i = (360^\circ / N) \times i$$

Parametreler:

- R: staging Radius = 15 metre
- N: Toplam İHA sayısı
- $\theta_i$ : i. İHA'nın açısal pozisyonu (derece)
- h: Formasyon merkez yüksekliği (Jüri Z parametresi)
- i: İHA index numarası (0, 1, 2, ..., N-1)

## 4.2 Formasyon ve Uçuş (0-10 puan)

Gerçekleştirilen formasyon ve uçuş sistemi, **dinamik drone sayısı adaptasyonu** ve **TEKNOFEST Finali Jüri parametrelerinin runtime'da güncellenebilmesi** özelliklerine sahiptir.



## Görsel 7. Dinamik Değişken Güncelleme Arayüzü

### A. Kalkış Fazı (Takeoff) - Smart Takeoff Implementation

#### 1. Dinamik Parametre Entegrasyonu

```
void InitializeTeknoFestParameters()
{
    currentFlightAltitude = ucusIrtifasi;           // Z parametresi
    currentFormationHoldTime = formasyonKorumaSuresi; // T parametresi
    currentAgentDistance = ajanlarArasiMesafe;      // X parametresi

    formationHeight = currentFlightAltitude;
    takeoffHeight = currentFlightAltitude - 2f;    // Güvenlik marjı
    safetyDistance = currentAgentDistance * 0.8f;  // X'in %80'i güvenlik
}
```

**Z Parametresi:** İHA'lar jüri parametresi irtifaya (Z-2m) kalkış yapar

**Runtime Güncelleme:** **parametreleriGuncelle** flag'i ile canlı parametre değişimi

**Güvenlik Marjı:** Hedef irtifanın 2m altında kalkış, sonra yükselme

#### 2. Akıllı Kalkış Sıralama Algoritması

```
// DİNAMİK TAKEOFF ORDER - Drone sayısına göre ayarla
List<int> takeoffOrder = new List<int>();

// Dış kenardan başla, merkeze doğru git
for (int i = 0; i < droneControllers.Count; i += 2)
    takeoffOrder.Add(i); // Çift indexler önce

for (int i = 1; i < droneControllers.Count; i += 2)
    takeoffOrder.Add(i); // Tek indexler sonra
```

**Kalkış Stratejisi:**

- Dış-İç Sıralama: Dış konumdaki İHA'lar önce, merkez İHA'lar sonra
- Çift-Tek Algoritması: Çarpışma riskini minimize eden akıllı sıralama
- Kademeli Timing: Her drone arasında 0.15s gecikme

## B. Staging Fazı (Hazırlık Pozisyonları)

```
// Staging positions - drone sayısına göre dinamik
Vector3[] stagingPositions = new Vector3[droneCount];
float angleStep = 360f / (float)droneCount;

for (int i = 0; i < droneCount; i++)
{
    float angle = i * angleStep * Mathf.Deg2Rad;
    stagingPositions[i] = new Vector3(
        center.x + stagingRadius * Mathf.Cos(angle),
        center.y,
        center.z + stagingRadius * Mathf.Sin(angle)
    );
}
```

### Matematiksel Model:

- Çember Yarıçapı: 4m (kodda `stagingRadius = 4f`)
- Açısal Dağılım:  $\theta = (360^\circ / N) \times i$ , burada N = drone sayısı
- Yükseklik: Z parametresi (jüri irtifası)
- Merkez Hesaplama: Formasyon pozisyonlarının ağırlık merkezi

## C. Formasyon Fazları - Dynamic Formation Implementation

### 1. V Formasyonu - Matematiksel Uygulama



```
Vector3[] GenerateDynamicVFormation()
{
    int droneCount = droneControllers.Count;
    Vector3[] positions = new Vector3[droneCount];

    // V'nin alt merkez noktası
    positions[0] = new Vector3(0, currentFlightAltitude, 0);

    // Kalan drone'ları sol ve sağ kanatlara dağıt
    int sidesCount = droneCount - 1;
    int leftWing = sidesCount / 2;
    int rightWing = sidesCount - leftWing;

    // Sol kanat:  $P_{left}(i) = (-X \times i \times 0.8, Z + i \times 2.5, 0)$ 
    for (int i = 0; i < leftWing; i++)
    {
        positions[i + 1] = new Vector3(
            -currentAgentDistance * (i + 1) * 0.8f,
            currentFlightAltitude + ((i + 1) * 2.5f),
            0
        );
    }

    // Sağ kanat:  $P_{right}(i) = (+X \times i \times 0.8, Z + i \times 2.5, 0)$ 
    for (int i = 0; i < rightWing; i++)
    {
        positions[leftWing + i + 1] = new Vector3(
            currentAgentDistance * (i + 1) * 0.8f,
            currentFlightAltitude + ((i + 1) * 2.5f),
            0
        );
    }

    return positions;
}
```

## D. Formasyon Koruma Fazı - Precision Control

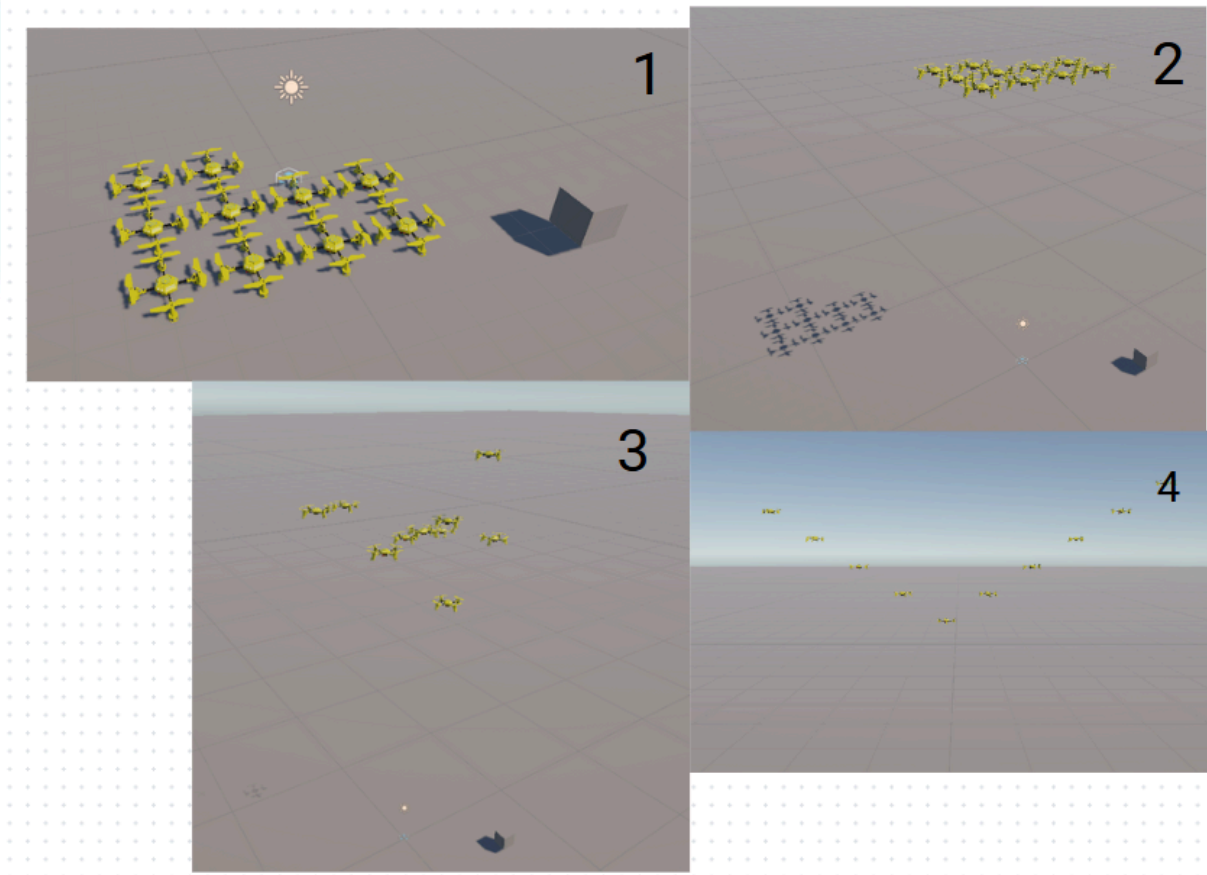
```
Vector3 CalculatePrecisionHold(Vector3 from, Vector3 to)
{
    Vector3 error = to - from;
    float errorMagnitude = error.magnitude;

    if (errorMagnitude < 0.1f) return Vector3.zero;

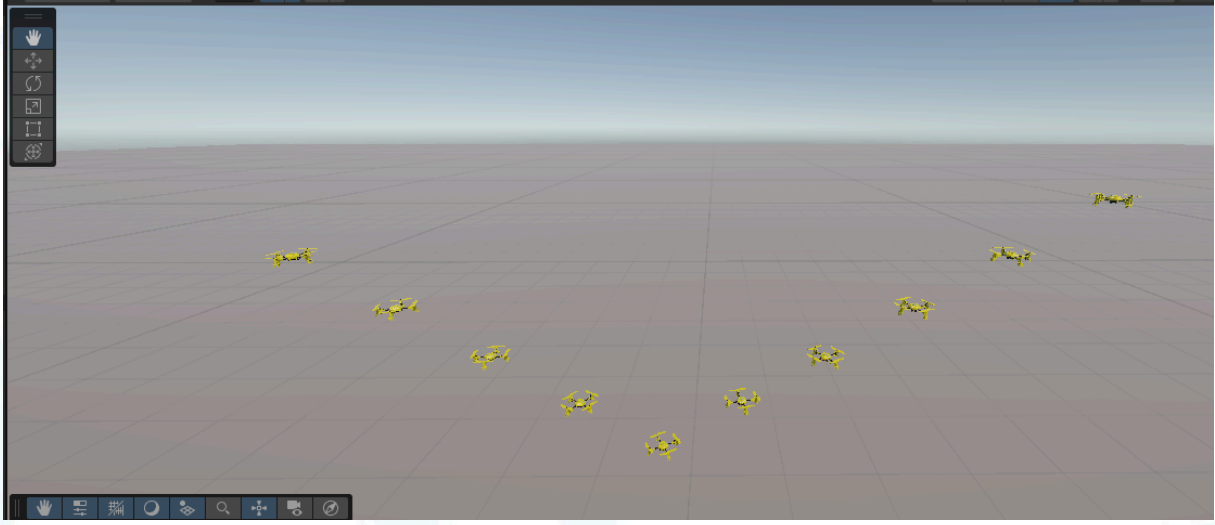
    float force = Mathf.Clamp(errorMagnitude * 2f, 0.1f, moveForce * 0.5f);
    return error.normalized * force;
}
```

**Görsel 8: Pozisyon Tutma Kontrolü C# Kodu**

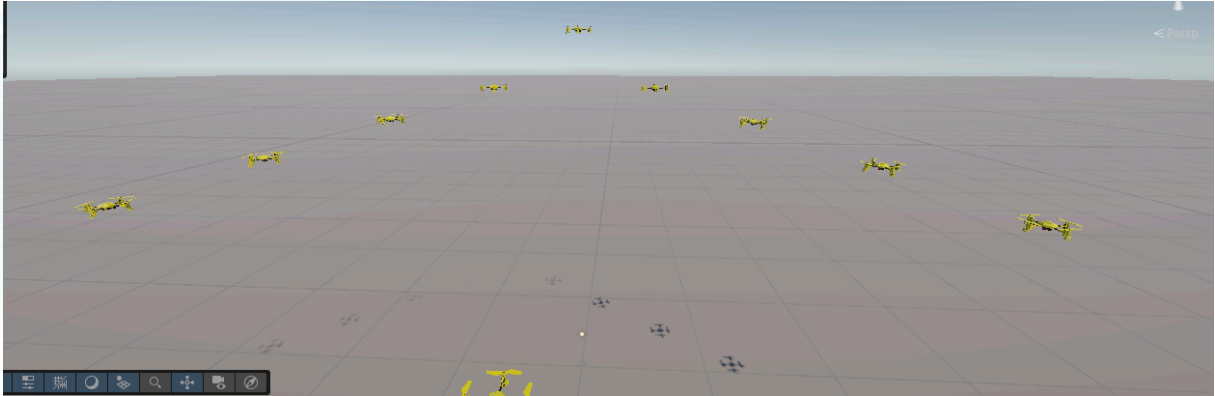
Görsel X'de V formasyonu örneği verilmekle beraber, formasyon geçişleri için yukarıda belirtilen işlemlerin görsel karşılıkları verilmiştir.



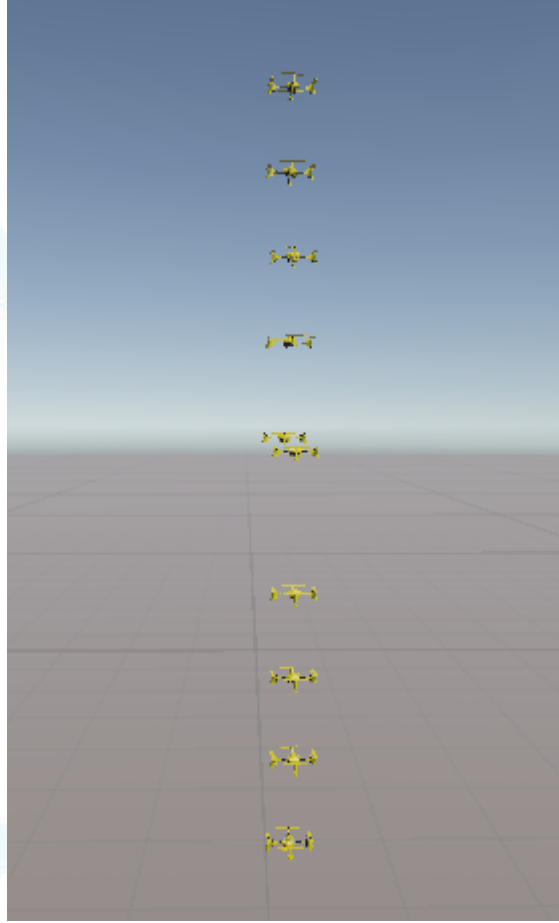
**Görsel 9: Örnek bir formasyon ve aşamalar**



**Görsel 10. V Formasyonu**



**Görsel 11. Ok başı Formasyonu**



**Görsel 11. Çizgi Formasyonu**

### **4.3 Pseudo kod**

İlgili sözde kod varyasyonları, verilen sözde koda ek olarak Türkçe ve sade bir dil kullanılarak GitHub platformunda sunulmuştur. İlgili depoya bakılmasını arz ederiz.



Sözde Kod:

```
// =====
// 4.3 PSEUDO KOD ALGORİTMALARI
// =====

// =====
// ANA SİSTEM AKIŞ ALGORİTMASI
// =====
ALGORITHM: TEKNOFEST_SURU_IHA_SISTEMI

BEGIN
    // BAŞLATMA FAZİ
    INITIALIZE sistem_parametreleri
    SET ucusIrtifasi ← Z_parametresi
    SET formasyonKorumaSuresi ← T_parametresi
    SET ajanlarArasiMesafe ← X_parametresi

    // İLETİŞİM HUB'I BAŞLATMA
    INITIALIZE CommunicationHub
    SET güncelleme_frekanşı ← 50Hz
    CREATE drone_veritabanı[]
    START real_time_monitoring()

    // İHA OLUŞTURMA VE KAYIT
    FOR i = 1 TO numberOfDrones DO
        CREATE drone_i
        CONFIGURE physics_parameters(drone_i)
        REGISTER drone_i IN hub
        SET drone_i.state ← GROUNDED
    END FOR

    // ANA OPERASYON DÖNGÜSÜ
    WHILE sistem_aktif DO
        // KOMUT BEKLEMESİ
        WAIT FOR yarışma_komutu

        SWITCH yarışma_komutu DO
            CASE "TAKEOFF":
                EXECUTE kalkış_protokolü()

            CASE "FORMATION_F":
                EXECUTE ok_formasyon_algoritması()

            CASE "FORMATION_V":
                EXECUTE v_formasyon_algoritması()

            CASE "FORMATION_L":
                EXECUTE çizgi_formasyon_algoritması()

            CASE "FORMATION_Y":
                EXECUTE dikey_sütun_algoritması()
```

```

CASE "PARAMETER_UPDATE":
    UPDATE sistem_parametreleri()

CASE "EMERGENCY_LAND":
    EXECUTE acil_iniş_protokolü()
END SWITCH
END WHILE
END

// =====
// ÇARPIŞMA ÖNLEME ALGORİTMASI
// =====
ALGORITHM: COLLISION_AVOIDANCE_SYSTEM
INPUT: dronelD, current_position, nearby_drones[]
OUTPUT: avoidance_force_vector

BEGIN
    SET total_avoidance_force ← (0, 0, 0)
    SET safety_radius ← ajanlarArasiMesafe * 0.8
    SET prediction_time ← 2.0 // saniye

    FOR EACH other_drone IN nearby_drones DO
        // MESAFE HESAPLAMA
        distance ← CALCULATE_DISTANCE(current_position, other_drone.position)

        IF distance < safety_radius AND distance > 0.1 THEN
            // GELECEK POZİSYON TAHMİNİ
            predicted_position ← other_drone.position + other_drone.velocity * prediction_time

            // KAÇINMA YÖN VEKTÖRÜ
            avoidance_direction ← NORMALIZE(current_position - predicted_position)

            // KUVVET BÜYÜKLÜĞÜ HESAPLAMA
            // Formül:  $F = K * (R_{safe} - d) / R_{safe}$ 
            force_magnitude ← 3.0 * (safety_radius - distance) / safety_radius

            // TOPLAM KUVVET HESAPLAMA
            total_avoidance_force ← total_avoidance_force +
                (avoidance_direction * force_magnitude)
        END IF
    END FOR

    RETURN total_avoidance_force
END

// =====
// DİNAMİK FORMASYON HESAPLAMA ALGORİTMASI
// =====
ALGORITHM: DYNAMIC_FORMATION_CALCULATOR
INPUT: formation_type, drone_count, Z_parameter, X_parameter
OUTPUT: formation_positions[]

```

```

BEGIN
  CREATE formation_positions[drone_count]

  SWITCH formation_type DO
    CASE "V_FORMATION":
      // V FORMASYONU HESAPLAMA
      formation_positions[0] ← (0, Z_parameter, 0) // Merkez nokta

      remaining_drones ← drone_count - 1
      left_wing_count ← remaining_drones / 2
      right_wing_count ← remaining_drones - left_wing_count

      // SOL KANAT HESAPLAMA
      FOR i = 1 TO left_wing_count DO
        wing_index ← i
        formation_positions[i] ← (
          -X_parameter * wing_index * 0.8,
          Z_parameter + (wing_index * 2.5),
          0
        )
      END FOR

      // SAĞ KANAT HESAPLAMA
      FOR i = 1 TO right_wing_count DO
        array_index ← left_wing_count + i
        formation_positions[array_index] ← (
          X_parameter * i * 0.8,
          Z_parameter + (i * 2.5),
          0
        )
      END FOR

    CASE "ARROW_FORMATION":
      // OK FORMASYONU HESAPLAMA
      formation_positions[0] ← (0, Z_parameter + 6, 0) // Uç nokta
      formation_positions[drone_count-1] ← (0, Z_parameter - 4, -(drone_count * 2)) // Kuyruk

      side_count ← drone_count - 2
      left_wing ← side_count / 2

      FOR i = 1 TO left_wing DO
        wing_step ← FLOAT(i) / (left_wing + 1)
        formation_positions[i] ← (
          -X_parameter * i,
          Z_parameter + 4 - (wing_step * 8),
          -(wing_step * drone_count * 1.5)
        )
      END FOR

      // Sağ kanat benzer şekilde hesaplanır...

    CASE "LINE_FORMATION":

```

```
// ÇİZGİ FORMASYONU HESAPLAMA
total_width ← (drone_count - 1) * X_parameter
start_x ← -total_width / 2

FOR i = 0 TO drone_count-1 DO
    formation_positions[i] ← (
        start_x + (i * X_parameter),
        Z_parameter,
        0
    )
END FOR

CASE "VERTICAL_COLUMN":
    // DİKEY SÜTUN FORMASYONU
    FOR i = 0 TO drone_count-1 DO
        formation_positions[i] ← (
            0,
            Z_parameter + (i * X_parameter * 0.6),
            0
        )
    END FOR
END SWITCH

RETURN formation_positions
END

// =====
// İHA DURUM MAKİNESİ (FINITE STATE MACHINE)
// =====
ALGORITHM: DRONE_STATE_MACHINE
INPUT: drone_id, current_state, system_command
OUTPUT: new_state, action_to_execute

BEGIN
    SWITCH current_state DO
        CASE GROUNDED:
            IF system_command = "ARM" THEN
                new_state ← ARMED
                action_to_execute ← initialize_systems()
            END IF

        CASE ARMED:
            IF system_command = "TAKEOFF" THEN
                new_state ← TAKING_OFF
                action_to_execute ← start_takeoff_sequence()
            END IF

        CASE TAKING_OFF:
            current_altitude ← GET_CURRENT_ALTITUDE()
            target_altitude ← ucusIrtifasi - 2.0

            IF current_altitude >= target_altitude THEN
```



```

    new_state ← HOVERING
    action_to_execute ← stabilize_flight()
ELSE
    action_to_execute ← apply_upward_thrust()
END IF

CASE HOVERING:
    IF system_command = "MOVE_TO_STAGING" THEN
        new_state ← STAGING
        action_to_execute ← move_to_staging_position()
    END IF

CASE STAGING:
    distance_to_staging ← CALCULATE_DISTANCE(current_pos, staging_pos)

    IF distance_to_staging < 2.0 THEN
        new_state ← HOVERING
        action_to_execute ← wait_for_formation_command()
    ELSE
        action_to_execute ← continue_staging_movement()
    END IF

CASE FORMATION_MOVE:
    distance_to_target ← CALCULATE_DISTANCE(current_pos, target_pos)

    IF distance_to_target < 1.5 THEN
        new_state ← FORMATION_HOLD
        action_to_execute ← lock_formation_position()
    ELSE
        // ÇARPIŞMA ÖNLEME İLE HAREKET
        avoidance_force ← COLLISION_AVOIDANCE_SYSTEM(drone_id, current_pos,
nearby_drones)
        target_force ← CALCULATE_TARGET_FORCE(current_pos, target_pos)
        total_force ← target_force + avoidance_force
        action_to_execute ← apply_movement_force(total_force)
    END IF

CASE FORMATION_HOLD:
    // HASSAS POZİSYON TUTMA
    position_error ← target_pos - current_pos

    IF MAGNITUDE(position_error) > 0.5 THEN
        correction_force ← position_error * 2.0 // PID kontrolü
        action_to_execute ← apply_correction_force(correction_force)
    END IF

    // T PARAMETRESİ KONTROLÜ
    hold_time ← GET_CURRENT_TIME() - formation_start_time
    IF hold_time >= formasyonKorumaSuresi THEN
        new_state ← HOVERING
        action_to_execute ← release_formation_lock()
    END IF

```

```

CASE LANDING:
    current_altitude ← GET_CURRENT_ALTITUDE()

    IF current_altitude <= 1.5 THEN
        new_state ← GROUNDED
        action_to_execute ← shutdown_flight_systems()
    ELSE
        action_to_execute ← apply_landing_thrust()
    END IF
END SWITCH

RETURN new_state, action_to_execute
END

// =====
// GÜVENLE YOLCULUK PLANLAMA ALGORİTMASI
// =====
ALGORITHM: SAFE_PATH_PLANNING
INPUT: current_pos, target_pos, obstacles[]
OUTPUT: safe_direction_vector

BEGIN
    // HEDEFİN ÇEKİM KUVVETİ
    target_direction ← NORMALIZE(target_pos - current_pos)

    // ÇARPIŞMA ÖNLEME KUVVETİ
    avoidance_force ← COLLISION_AVOIDANCE_SYSTEM(current_pos, obstacles)

    // AĞIRLIKLI KOMBİNASYON
    danger_level ← CALCULATE_DANGER_LEVEL(current_pos, obstacles)
    avoidance_weight ← CLAMP(danger_level / ajanlarArasiMesafe, 0, 1)
    target_weight ← 1.0 - avoidance_weight

    // GÜVENLİ YOL HESAPLAMA
    safe_direction ← NORMALIZE(
        target_direction * target_weight +
        NORMALIZE(avoidance_force) * avoidance_weight
    )

    RETURN safe_direction
END

// =====
// PERFORMANS İZLEME ALGORİTMASI
// =====
ALGORITHM: PERFORMANCE_MONITORING
BEGIN
    WHILE sistem_aktif DO
        // PERFORMANS METRİKLERİ TOPLAMA
        FOR EACH drone IN active_drones DO
            position ← GET_POSITION(drone)

```

```

velocity ← GET_VELOCITY(drone)
target ← GET_TARGET_POSITION(drone)

// HASSAS POZİSYON KONTROLÜ
position_error ← DISTANCE(position, target)
IF position_error > 0.5 THEN
    LOG_WARNING("Drone " + drone.id + " pozisyon hatası: " + position_error)
END IF

// ÇARPIŞMA RİSKİ ANALİZİ
nearby_drones ← GET_NEARBY_DRONES(drone, ajanlarArasiMesafe)
FOR EACH nearby IN nearby_drones DO
    distance ← DISTANCE(drone.position, nearby.position)
    IF distance < 2.0 THEN
        TRIGGER_EMERGENCY_PROTOCOL(drone.id, nearby.id)
    END IF
END FOR
END FOR

// SİSTEM SAĞLIĞI KONTROLÜ
cpu_usage ← GET_CPU_USAGE()
memory_usage ← GET_MEMORY_USAGE()
frame_rate ← GET_FRAME_RATE()

LOG_PERFORMANCE_DATA(cpu_usage, memory_usage, frame_rate)

WAIT 0.02 // 50Hz güncelleme
END WHILE
END

```

## 4.4 Simülasyon (0-15 puan)

Rapor boyutu artmasından kaçınmak ve daha kompakt bir rapor sunmak adına simülasyon dosyalarını youtube üzerinden paylaştık. Yine video ve benzer medyaları github linkimizde mevcut olduğunu belirtmek isteriz.

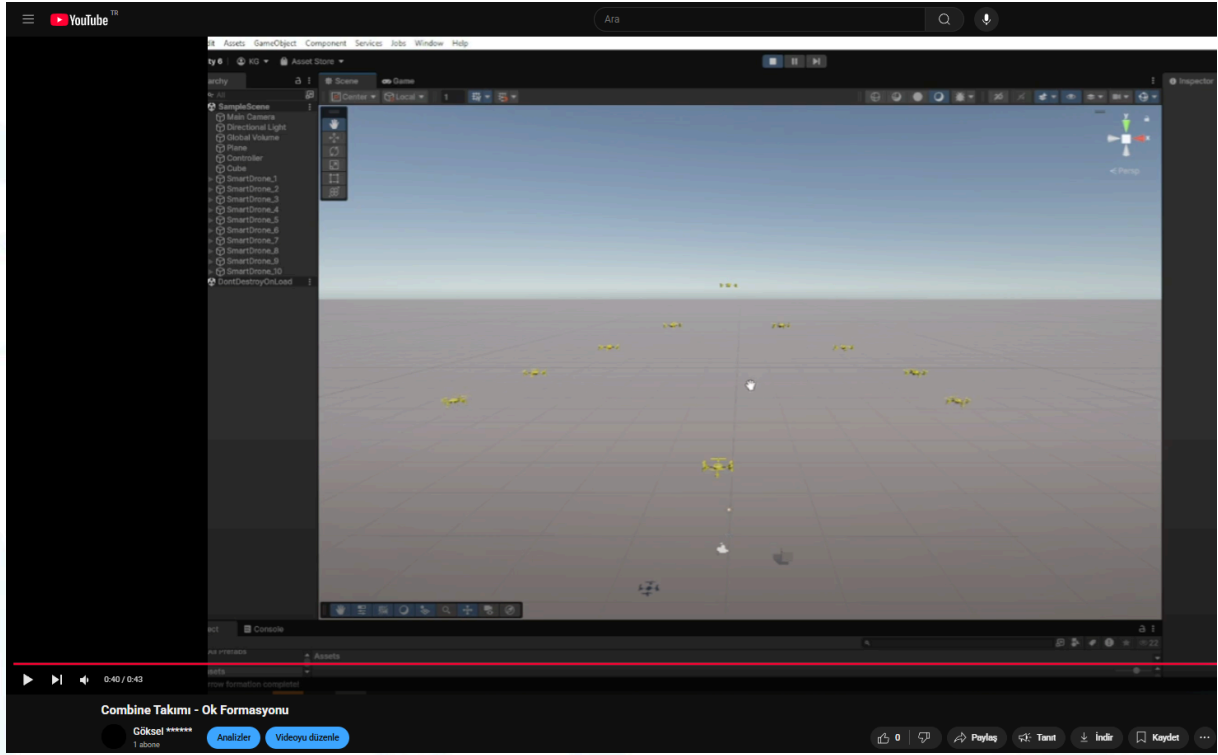
**Çizgi Formasyonu:** <https://youtu.be/HYYeip-Mim8>

**V Formasyonu:** [https://youtu.be/DuO\\_IYh8ixo](https://youtu.be/DuO_IYh8ixo)

**Ok Formasyonu** <https://youtu.be/lpQJxfUj--M>

**Sürü Navigasyon** <https://youtu.be/xiAAvgKC9-k>

**Birey Ekleme - Çıkarma** <https://youtu.be/rmu6ozl7tHk>



**Görsel 12. Youtube üzerinden yüklenen videoların görünümü**

## 5. FORMAT VE KAYNAKÇA (5 Puan)

- [1] G. Göksel, "Teknofest-Suru-IHA," GitHub, 2025. [Online]. Available: <https://github.com/RsGoksel/Teknofest-Suru-IHA>
- [2] Temizer, S., Kochenderfer, M.J., Kaelbling, L.P., Lozano-Pérez, T., & Kuchar, J.K. (2010). Collision Avoidance for Unmanned Aircraft using Markov Decision Processes. AIAA Guidance, Navigation, and Control Conference.
- [3] A. Puente-Castro, D. Rivero, A. Pazos, and E. Fernandez-Blanco, "A review of Artificial Intelligence applied to Path Planning in UAV swarms," Applied Sciences, 2022.
- [4] Docker Hub, "Gazebo Official Images," 2024. [Online]. Available: [https://hub.docker.com/\\_/gazebo/](https://hub.docker.com/_/gazebo/)
- [5] Gazebo Simulator, "Ignition Docker Environment Setup," 2024. [Online]. Available: [https://gazebo-sim.org/docs/fortress/ign\\_docker\\_env/](https://gazebo-sim.org/docs/fortress/ign_docker_env/)

Not: Bu raporda belirtilen [video] referansları, test süreçlerinde kaydedilen demonstrasyon videolarını ifade etmektedir. Videolar proje GitHub deposunda mevcuttur.