T.C.
MANİSA CELAL BAYAR ÜNİVERSİTESİ
HASAN FERDİ TURGUTLU TEKNOLOJİ FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

GÖRÜNTÜ İŞLEME DESTEKLİ OTONOM TARIM DEVRİYE ARACININ TASARIMI VE UYGULAMASI

Ulaş Can ÇAKMAK
222808014

Halil İbrahim KURNAZ
222808057

Yavuz SARI
222812001

Danışman
Arş. Gör. Dr. Göksu TAŞ

MANİSA–2025

PROJE ONAYI

Ulaş Can ÇAKMAK , Halil İbrahim KURNAZ , Yavuz SARI tarafından hazırlanan "Görüntü İşleme Destekli Otonom Tarım Devriye Aracının Tasarımı ve Uygulaması"başlıklı bu çalışma, Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü'nde Lisans Bitirme Tasarım Projesi olarak hazırlanmış ve aşağıda imzaları bulunan jüri tarafından 2025 yılında kabul edilmiştir.

Danışman **Arş. Gör. Dr. Göksu TAŞ**

Manisa Celal Bayar Üniversitesi

TAAHHÜTNAME

Bu tez çalışmasının Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü Lisans Bitirme / Tasarım Projesi kapsamında, akademik etik kurallara uygun olarak tarafımdan hazırlandığını; çalışmada kullanılan tüm kaynakların bilimsel kurallara uygun şekilde atıf yapılarak belirtildiğini beyan ederim.

Ulaş Can Çakmak
Halil İbrahim KURNAZ
Yavuz SARI

İÇİNDEKİLER

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

YZ Yapay Zekâ

ESA Evrişimli Sinir Ağı

CNN Convolutional Neural Network (Evrişimli Sinir Ağı)

CPU Merkezi İşlem Birimi

GPU Grafik İşlem Birimi

KKS Küresel Konumlama Sistemi

Nİ Nesnelerin İnterneti

DGM Darbe Genişlik Modülasyonu

RGB Kırmızı, Yeşil, Mavi Renk Uzayı

ROS Robot İşletim Sistemi

RTOS Gerçek Zamanlı İşletim Sistemi

UART Evrensel Asenkron Alıcı-Verici

USB Evrensel Seri Veri Yolu

YOLO You Only Look Once (Gerçek Zamanlı Nesne Algılama Algoritması)

ŞEKİLLER DİZİNİ

Sayfa

TABLOLAR DİZİNİ

Sayfa

TEŞEKKÜR

Çalışmamın her aşamasında bilgi ve deneyimleriyle bana yol gösteren, proje sürecinde karşılaştığım teknik ve akademik problemlerin çözümünde desteğini esirgemeyen danışman hocam **Arş. Gör. Dr. Göksu TAŞ**'a en içten teşekkürlerimi sunarım.

Lisans öğrenimim boyunca mesleki ve akademik gelişimime katkı sağlayan, bilgi birikimleri ve tecrübeleriyle ufkumu genişleten Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü öğretim elemanlarına teşekkür ederim.

Proje süreci boyunca manevi desteklerini her zaman hissettiğim, fikir ve önerileriyle katkı sağlayan arkadaşlarıma; öğrenim hayatım boyunca maddi ve manevi desteğini hiçbir zaman esirgemeyen aileme sonsuz teşekkürlerimi sunarım.

Ulaş Can ÇAKMAK
Halil İbrahim KURNAZ
Yavuz SARI

Manisa, 2025

ÖZET

Mekatronik Mühendisliği Tasarımı Raporu

Görüntü İşleme Destekli Otonom Tarım Devriye Aracının Tasarımı ve Uygulaması

Ulaş Can ÇAKMAK
Halil İbrahim KURNAZ
Yavuz SARI

Manisa Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Mekatronik Mühendisliği Bölümü

Danışman: Arş. Gör. Dr. Göksu TAŞ

Bu çalışmada, tarım alanlarında bitki sağlığının izlenmesi, hastalıkların erken tespiti ve tarımsal ilaç kullanımının optimize edilmesi amacıyla görüntü işleme destekli otonom bir tarım devriye aracı tasarlanmış ve uygulanmıştır. Geliştirilen sistem, tarım arazisi üzerinde otonom olarak hareket edebilmekte, bitkilerden elde edilen görüntüleri yapay zekâ tabanlı yöntemlerle analiz ederek hastalık belirtilerini tespit edebilmekte ve yalnızca hastalıklı bölgelerde noktasal ilaçlama gerçekleştirebilmektedir.

Sistem tasarımında mekatronik yaklaşım benimsenmiş; mekanik yapı, elektronik donanım ve yazılım bileşenleri bir bütün olarak ele alınmıştır. Görüntü işleme aşamasında derin öğrenme tabanlı evrişimli sinir ağları kullanılarak bitki yapraklarında oluşan hastalık belirtileri sınıflandırılmıştır. Elde edilen veriler zaman ve konum bilgileriyle birlikte kayıt altına alınarak tarımsal üretim süreçlerinin izlenebilirliği sağlanmıştır.

Gerçekleştirilen deneysel çalışmalar sonucunda geliştirilen otonom tarım devriye aracının bitki hastalıklarını yüksek doğruluk oranı ile tespit edebildiği, gereksiz ilaç kullanımını azalttığı ve sürdürülebilir tarım uygulamalarına katkı sağladığı görülmüştür. Bu yönüyle çalışma, akıllı tarım teknolojileri kapsamında uygulanabilir ve geliştirilebilir bir çözüm sunmaktadır.

Anahtar Kelimeler: Otonom tarım robotu, görüntü işleme, yapay zekâ, derin öğrenme, noktasal ilaçlama

2025, 71 sayfa

ABSTRACT

Mechatronics Engineering Design Report

Design and Implementation of an Image Processing–Based Autonomous Agricultural Patrol Vehicle

Ulaş Can ÇAKMAK
Halil İbrahim KURNAZ
Yavuz SARI

Manisa Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Mekatronik Mühendisliği Bölümü

Advisor : Arş. Gör. Dr. Göksu TAŞ

In this study, an image processing–based autonomous agricultural patrol vehicle was designed and implemented to monitor plant health, enable early detection of diseases, and optimize the use of agricultural chemicals. The developed system is capable of autonomously navigating agricultural fields, analyzing images obtained from plants using artificial intelligence–based methods, detecting disease symptoms, and performing targeted spraying only in diseased areas.

A mechatronic design approach was adopted in the system design, in which mechanical structure, electronic hardware, and software components were considered as an integrated whole. During the image processing stage, deep learning–based convolutional neural networks were used to classify disease symptoms occurring on plant leaves. The acquired data were recorded together with time and location information, ensuring the traceability of agricultural production processes.

Experimental results demonstrated that the developed autonomous agricultural patrol vehicle was able to detect plant diseases with a high accuracy rate, reduce unnecessary chemical usage, and contribute to sustainable agricultural practices. In this respect, the study presents an applicable and extensible solution within the scope of smart agriculture technologies.

Keywords: Autonomous agricultural robot, image processing, artificial intelligence, deep learning, targeted spraying

1. GİRİŞ

Tarım sektörü, artan dünya nüfusu, azalan tarım arazileri ve iklim değişikliğinin yol açtığı belirsizlikler nedeniyle günümüzde önemli yapısal dönüşümler yaşamaktadır. Birleşmiş Milletler verilerine göre dünya nüfusunun 2050 yılına kadar yaklaşık 9,7 milyara ulaşması beklenmekte olup, bu artış gıda üretiminde verimliliğin artırılmasını zorunlu hale getirmektedir. Bu bağlamda, geleneksel tarım yöntemleri tek başına artan gıda talebini karşılamakta yetersiz kalmakta; tarımda teknoloji destekli, akıllı ve sürdürülebilir çözümlere olan ihtiyaç giderek artmaktadır.

Geleneksel tarım uygulamalarında bitki sağlığının izlenmesi ve hastalık tespiti çoğunlukla çiftçilerin görsel gözlemlerine ve deneyimlerine dayanmaktadır. Bu yaklaşım, özellikle geniş tarım alanlarında hastalıkların erken aşamada fark edilmesini zorlaştırmakta, geç müdahaleler nedeniyle ürün kayıplarına yol açmaktadır. Ayrıca hastalık tespitinin gecikmesi, tarımsal ilaçların tüm alana genel olarak uygulanmasına neden olmakta; bu durum hem gereksiz kimyasal kullanımını artırmakta hem de çevresel ve ekonomik açıdan olumsuz sonuçlar doğurmaktadır.

Son yıllarda görüntü işleme ve yapay zekâ alanında yaşanan gelişmeler, tarım sektöründe yeni uygulamaların ortaya çıkmasını sağlamıştır. Özellikle derin öğrenme tabanlı evrişimli sinir ağları (CNN), bitki yaprak görüntülerinden hastalıkların otomatik olarak tespit edilmesi ve sınıflandırılmasında yüksek doğruluk oranları sunmaktadır. Literatürde yapılan birçok çalışmada, PlantVillage gibi veri setleri kullanılarak bitki hastalıklarının başarılı bir şekilde sınıflandırıldığı raporlanmıştır. Ancak bu çalışmaların büyük bir kısmı, hastalık tespitini yalnızca yazılım düzeyinde ele almakta; elde edilen sonuçların sahada, gerçek zamanlı ve otonom bir sistem ile eyleme dönüştürülmesi sınırlı kalmaktadır.

Öte yandan, tarımda kullanılan mevcut ilaçlama yöntemlerinin çoğu, hastalığın yayılım durumuna bakılmaksızın tüm tarım alanına uygulanan genel ilaçlama yaklaşımına dayanmaktadır. Bu yaklaşım, sağlıklı bitkilerin de kimyasal maddelere maruz kalmasına neden olmakta ve çevre kirliliği, toprak verimliliğinin azalması ve maliyet artışı gibi sorunları beraberinde getirmektedir. Bu nedenle, yalnızca hastalıklı bölgelerin hedeflenerek müdahale edilmesini sağlayan **noktasal (hedefli) ilaçlama sistemleri**, sürdürülebilir tarım uygulamaları açısından büyük önem taşımaktadır.

Bu çalışmada, literatürde sıklıkla ayrı ayrı ele alınan görüntü işleme destekli hastalık tespiti, otonom hareket ve tarımsal müdahale süreçleri tek bir entegre sistem altında birleştirilmiştir. Geliştirilen sistem, Raspberry Pi tabanlı bir platform üzerinde çalışan yapay zekâ algoritmaları aracılığıyla bitki yaprak görüntülerini analiz etmekte ve hastalık tespiti gerçekleştirmektedir. Elde edilen hastalık bilgisi, otonom hareket kabiliyetine sahip mobil bir robot aracılığıyla değerlendirilerek yalnızca hastalıklı bölgelerde noktasal ilaçlama yapılmasını sağlamaktadır. Bu yaklaşım sayesinde hem

tarımsal ilaç kullanımının optimize edilmesi hem de çevresel etkilerin azaltılması hedeflenmiştir.

Ayrıca geliştirilen sistem, düşük maliyetli donanım bileşenleri kullanılarak tasarlanmış olup, küçük ve orta ölçekli tarım işletmelerinde uygulanabilir bir çözüm sunmaktadır. Sistem mimarisi; algılama, karar verme ve eylem katmanlarından oluşan modüler bir yapıya sahiptir. Bu sayede sistemin farklı tarım senaryolarına uyarlanabilmesi ve ileride geliştirilebilir olması amaçlanmıştır. Yapay zekâ destekli hastalık tespiti, otonom sürüş ve servo motor kontrollü ilaçlama mekanizması, gerçek zamanlı olarak birlikte çalışacak şekilde entegre edilmiştir.

Bu çalışmanın temel amacı, tarım alanlarında bitki hastalıklarının erken tespiti ve etkili müdahalesini mümkün kılan, otonom, düşük maliyetli ve uygulanabilir bir tarımsal destek sistemi geliştirmektir. Çalışma kapsamında sistemin donanım ve yazılım bileşenleri ayrıntılı olarak ele alınmış, geliştirilen yapay zekâ modeli performans metrikleri ile değerlendirilmiş ve gerçek zamanlı testler ile sistemin işlevselliği ortaya konmuştur. Elde edilen sonuçlar, geliştirilen sistemin akıllı tarım uygulamaları açısından umut vadeden bir çözüm sunduğunu göstermektedir.

2. GENEL BİLGİLER

Bu bölümde, çalışmanın temelini oluşturan kavramlar ve teknolojiler açıklanmıştır. Akıllı tarım sistemleri, otonom mobil robotlar, görüntü işleme teknikleri ve derin öğrenme tabanlı yaklaşımlar literatür çerçevesinde ele alınmış, ilgili başlıklar çalışmanın kapsamı ile ilişkilendirilmiştir. Böylece geliştirilen sistemin hangi teknolojik altyapıya dayandığı açık bir şekilde ortaya konulmuştur.

2.1. Akıllı Tarım Sistemleri

Akıllı tarım sistemleri, tarımsal üretim süreçlerinin daha verimli, sürdürülebilir ve kontrollü bir şekilde yürütülmesini amaçlayan teknoloji tabanlı uygulamaları kapsamaktadır. Bu sistemlerde sensörler, görüntüleme cihazları, haberleşme altyapıları ve veri analiz yöntemleri bir arada kullanılarak tarım alanlarından sürekli veri toplanmaktadır. Toplanan bu veriler, karar destek sistemleri aracılığıyla analiz edilmekte ve üretim süreçlerine yön vermektedir.

Geleneksel tarım uygulamalarında üretim büyük ölçüde deneyime dayalı kararlarla yürütülmektedir. Ancak iklim değişikliği, artan nüfus ve sınırlı doğal kaynaklar, bu yaklaşımın yetersiz kalmasına neden olmaktadır. Akıllı tarım sistemleri, veriye dayalı karar mekanizmaları sunarak bu sorunların önüne geçmeyi hedeflemektedir. Özellikle bitki sağlığının sürekli izlenmesi, hastalıkların erken aşamada tespit edilmesini mümkün kılmaktadır.

Akıllı tarım uygulamaları; hassas tarım, otonom tarım makineleri, uzaktan algılama sistemleri ve yapay zekâ destekli analiz yöntemleri gibi alt alanları kapsamaktadır. Bu çalışmada geliştirilen sistem, akıllı tarım yaklaşımı doğrultusunda bitki hastalıklarının otomatik olarak tespit edilmesini ve yalnızca gerekli alanlara müdahale edilmesini amaçlamaktadır.

Akıllı tarım uygulamalarında sensörler, görüntü işleme sistemleri ve yapay zekâ tabanlı karar mekanizmalarının birlikte kullanılması, tarımsal üretimde verimliliği artırmakta ve kaynak kullanımını optimize etmektedir. Literatürde yapılan

çalışmalarda, görüntü tabanlı bitki hastalığı tespit sistemlerinin erken teşhis açısından önemli avantajlar sağladığı belirtilmiştir [1], [2].

2.2. Otonom Mobil Robotlar

Otonom mobil robotlar, çevrelerinden aldıkları verileri kullanarak konumlarını belirleyebilen, karar verebilen ve insan müdahalesine ihtiyaç duymadan hareket edebilen sistemlerdir. Bu robotlar; algılama birimleri, kontrol algoritmaları ve eyleyici mekanizmaların birlikte çalışmasıyla görevlerini yerine getirmektedir.

Tarım sektöründe kullanılan otonom robotlar, özellikle geniş alanlarda tekrarlayan görevlerin otomatikleştirilmesi açısından önemli avantajlar sağlamaktadır. İlaçlama, gübreleme, tarama ve ürün takibi gibi işlemler otonom robotlar aracılığıyla daha hızlı ve kontrollü bir şekilde gerçekleştirilebilmektedir. Bu durum, hem iş gücü maliyetlerini azaltmakta hem de insan kaynaklı hataların önüne geçmektedir.

Bu çalışmada ele alınan otonom tarım devriye aracı, tarım arazisi üzerinde belirlenen güzergâh boyunca hareket ederek bitkilerden görüntü toplamakta ve elde edilen veriler doğrultusunda karar vermektedir. Robotun otonom hareket kabiliyeti, sistemin sürekli ve düzenli tarama yapabilmesine olanak tanımaktadır.

2.3. Görüntü İşleme Teknikleri

Görüntü işleme, dijital görüntüler üzerinde çeşitli matematiksel ve istatistiksel işlemler uygulanarak anlamlı bilgilerin elde edilmesini amaçlayan bir disiplindir. Tarım alanında görüntü işleme teknikleri; bitki türü tanıma, hastalık tespiti, büyüme analizi ve ürün sınıflandırma gibi birçok uygulamada yaygın olarak kullanılmaktadır.

Görüntü işleme süreci genel olarak görüntünün elde edilmesi, ön işleme, özellik çıkarımı ve sınıflandırma aşamalarından oluşmaktadır. Ön işleme aşamasında gürültü giderme, yeniden boyutlandırma ve renk uzayı dönüşümleri yapılmaktadır. Bu işlemler, sınıflandırma algoritmalarının doğruluğunu doğrudan etkilemektedir.

Bu çalışmada, bitki yapraklarından elde edilen görüntüler üzerinde derin öğrenme tabanlı görüntü işleme yöntemleri uygulanmıştır. Geleneksel yöntemlerin aksine, özellik çıkarımı işlemi manuel olarak yapılmamış, bu işlem evrişimli sinir ağları tarafından otomatik olarak gerçekleştirilmiştir.

Görüntü işleme teknikleri, tarımsal uygulamalarda bitki sağlığının izlenmesi ve hastalıkların sınıflandırılması amacıyla yaygın olarak kullanılmaktadır. Özellikle derin öğrenme tabanlı yöntemlerin, geleneksel özellik çıkarımına dayalı yaklaşımlara kıyasla daha yüksek doğruluk oranları sunduğu literatürde rapor edilmiştir [3], [4].

2.4. Derin Öğrenme ve Evrişimli Sinir Ağları

Derin öğrenme, çok katmanlı yapay sinir ağları kullanılarak büyük ve karmaşık veri kümeleri üzerinde öğrenme gerçekleştirilmesini sağlayan bir makine öğrenmesi yaklaşımıdır. Evrişimli sinir ağları (ESA), özellikle görüntü verileri üzerinde yüksek başarı oranları sunması nedeniyle görüntü işleme uygulamalarında sıklıkla tercih edilmektedir.

Evrişimli sinir ağları; evrişim katmanları, havuzlama katmanları ve tam bağlantılı katmanlardan oluşmaktadır. Bu katmanlar sayesinde görüntülerdeki kenar, doku ve şekil gibi özellikler otomatik olarak öğrenilmektedir. Derin öğrenme tabanlı bu yaklaşım, geleneksel görüntü işleme yöntemlerine kıyasla daha yüksek doğruluk oranları sunmaktadır.

Bu çalışmada bitki hastalıklarının sınıflandırılması amacıyla YOLOv8 tabanlı bir derin öğrenme modeli kullanılmıştır. Model, transfer öğrenme yaklaşımı ile eğitilmiş ve gerçek zamanlı çalışabilecek şekilde sisteme entegre edilmiştir.

Evrişimli sinir ağları, görüntülerdeki ayırt edici özellikleri otomatik olarak öğrenebilme yetenekleri sayesinde sınıflandırma problemlerinde yüksek başarı sağlamaktadır. Son yıllarda yapılan çalışmalarda, derin öğrenme tabanlı modellerin tarımsal görüntü analizi uygulamalarında etkin bir şekilde kullanıldığı gösterilmiştir [5], [6].

2.5. Noktasal (Hedefli) İlaçlama Sistemleri

Noktasal ilaçlama sistemleri, tarım alanlarında yalnızca hastalıklı veya riskli bölgelerin ilaçlanmasını amaçlayan sistemlerdir. Bu yaklaşım, geleneksel yaygın ilaçlama yöntemlerine kıyasla daha kontrollü ve çevre dostu bir çözüm sunmaktadır.

Hedefli ilaçlama sayesinde kimyasal madde tüketimi azalmakta, çevresel kirlilik önlenmekte ve üretim maliyetleri düşmektedir. Ayrıca, bitkilerin gereksiz kimyasallara maruz kalması engellenerek ürün kalitesi artırılmaktadır.

Bu çalışmada geliştirilen sistemde, görüntü işleme ve yapay zekâ algoritmaları kullanılarak hastalık tespit edilen bitkiler belirlenmekte ve ilaçlama işlemi yalnızca bu bölgelerde gerçekleştirilmektedir. Bu sayede hem ekonomik hem de çevresel açıdan verimli bir tarımsal çözüm sunulmaktadır.

2.6. Literatürde Benzer Çalışmalar ve Bu Çalışmanın Farkı

Son yıllarda akıllı tarım uygulamaları kapsamında görüntü işleme ve yapay zekâ destekli hastalık tespiti üzerine birçok çalışma gerçekleştirilmiştir. Bu çalışmaların büyük bir kısmı, bitki yaprak görüntülerinin sınıflandırılması yoluyla hastalık tespitine odaklanmış, ancak bu tespitlerin sahada otonom bir sistem ile eyleme dönüştürülmesi sınırlı kalmıştır.

Örneğin, Mohanty ve arkadaşları (2016) tarafından gerçekleştirilen çalışmada, derin öğrenme tabanlı yöntemler kullanılarak PlantVillage veri seti üzerinde yüksek doğruluk oranları elde edilmiştir. Ancak söz konusu çalışmada hastalık tespiti yalnızca yazılımsal düzeyde ele alınmış, elde edilen sonuçlar herhangi bir fiziksel sistem veya otonom mekanizma ile bütünleştirilmemiştir. Benzer şekilde Ferentinos (2018), evrişimli sinir ağları kullanarak bitki hastalıklarının sınıflandırılmasında başarılı sonuçlar raporlamış, ancak çalışmada gerçek zamanlı uygulama ve tarımsal müdahale süreçlerine yer verilmemiştir.

Daha güncel çalışmalarda, görüntü işleme tabanlı tarım robotları geliştirilmiş olsa da bu sistemlerin çoğunda ya yalnızca görüntü toplama yapılmakta ya da tüm tarım alanına genel ilaçlama uygulanmaktadır. Zhang ve ark. (2021) tarafından geliştirilen otonom tarım aracı, bitki görüntülerini analiz edebilmesine rağmen, tespit edilen hastalıklara yönelik noktasal ilaçlama mekanizması içermemektedir. Bu durum, gereksiz ilaç kullanımına ve çevresel etkilere yol açabilmektedir.

Bu çalışmada ise literatürdeki mevcut yaklaşımlardan farklı olarak, yalnızca hastalık tespiti yapılmamış; tespit edilen hastalık bilgisi, otonom hareket kabiliyetine sahip bir mobil platform üzerinde gerçek zamanlı olarak değerlendirilmiş ve yalnızca hastalıklı bölgelerde noktasal ilaçlama gerçekleştirilmiştir. Ayrıca sistem, Raspberry Pi tabanlı görüntü işleme altyapısı ile Arduino destekli eyleyici kontrolünü bir araya getirerek, düşük maliyetli ve uygulanabilir bir çözüm sunmaktadır. Böylece bu çalışma, literatürde sıklıkla ayrı ayrı ele alınan hastalık tespiti, otonom hareket ve

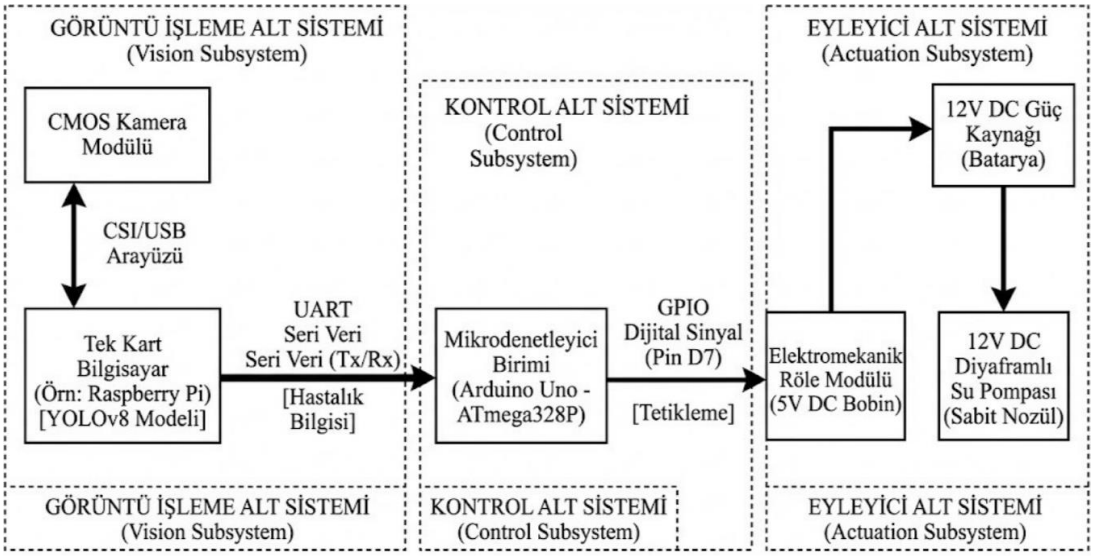
ilaçlama süreçlerini tek bir entegre sistem altında birleştirmesi açısından özgünlük taşımaktadır.

3. MATERYAL VE YÖNTEMLER

3.1. Sistem Genel Mimarisi

Geliştirilen sistem, mekatronik bir yaklaşımla tasarlanmış olup algılama, karar verme ve eylem olmak üzere üç temel katmandan oluşmaktadır. Sistem mimarisi Şekil 3.1'de gösterilmiştir.

Geliştirilen otonom tarımsal ilaçlama robotu; görüntü işleme, kontrol ve eyleyici olmak üzere üç temel alt sistemden oluşmaktadır. Sistem mimarisi, alt sistemler arasındaki veri ve sinyal akışını açık bir şekilde gösterecek biçimde tasarlanmıştır. Şekil 3.2'de, geliştirilen sistemin genel mimarisi ve alt sistemler arasındaki etkileşim sunulmaktadır.



Şekil 3.1. Arduino tabanlı ilaçlama sisteminin detaylı yazılım akış diyagramı

3.1.1. Algılama Katmanı

Algılama katmanı, tarım alanındaki bitkilerin görsel verilerinin elde edilmesinden sorumludur. Bu katmanda Raspberry Pi ile uyumlu kamera modülü kullanılmıştır.

Kamera aracılığıyla elde edilen görüntüler gerçek zamanlı olarak işlenmek üzere merkezi kontrol birimine aktarılmaktadır.

3.1.2. Karar Verme Katmanı

Karar verme katmanı, elde edilen görüntüler üzerinde yapay zekâ tabanlı analizlerin gerçekleştirildiği katmandır. Bu katmanda, derin öğrenme tabanlı evrişimli sinir ağı modeli kullanılarak bitki hastalıkları sınıflandırılmaktadır. Modelden elde edilen sonuçlara göre sistemin bir sonraki hareketi belirlenmektedir.

3.1.3. Eylem Katmanı

Eylem katmanı, karar verme katmanından gelen komutlar doğrultusunda robotun fiziksel hareketlerinin ve ilaçlama işlemlerinin gerçekleştirildiği katmandır. Bu katmanda Arduino mikrodenetleyici kullanılarak motor sürücüleri ve ilaçlama pompası kontrol edilmektedir.

3.2. Donanım Yapısı

Sistemin donanım yapısı; kontrol birimleri, algılama birimleri ve hareket mekanizmasından oluşmaktadır.

3.2.1. Kontrol Birimleri

Sistemde iki adet kontrol birimi kullanılmıştır. Yüksek seviyeli işlemler için Raspberry Pi, düşük seviyeli donanım kontrolü için Arduino mikrodenetleyici tercih edilmiştir. Bu yapı sayesinde görüntü işleme ve motor kontrol işlemleri birbirinden ayrılmıştır.

Geliştirilen sistemde, yüksek seviyeli görüntü işleme ve yapay zekâ işlemleri ile düşük seviyeli donanım kontrolünün ayrıştırılması amacıyla iki farklı kontrol birimi kullanılmıştır. Sistemde kullanılan kontrol birimleri ve bu birimlere ait temel teknik özellikler Tablo 3.1'de verilmiştir.

Tablo 3.1. Kontrol Birimleri ve Temel Özellikleri

Bileşen	Görev	Teknik Özellik
Raspberry Pi 5	Görüntü İşleme , AI	Quad-Core CPU , 8 GB RAM
Arduino Uno/Nano	Motor ve pompa kontrolü	16 MHz, ATmega328

3.2.2. Algılama Donanımı

Bitki görüntülerinin elde edilmesi için Raspberry Pi Camera Module kullanılmıştır. Kamera, tarım alanında bitkilerin yaprak yüzeylerini görüntüleyebilecek konumda yerleştirilmiştir.

Bitki yapraklarının görüntülenmesi ve çevresel verilerin elde edilmesi amacıyla sistemde farklı algılama donanımları kullanılmıştır. Kullanılan algılama donanımlarına ait bilgiler Tablo 3.2'de sunulmuştur.

Tablo 3.2. Algılama Donanımı

Donanım	Açıklama
Kamera Modülü	Gerçek zamanlı görüntü elde etme
Çözünürlük	640 x 480 piksel

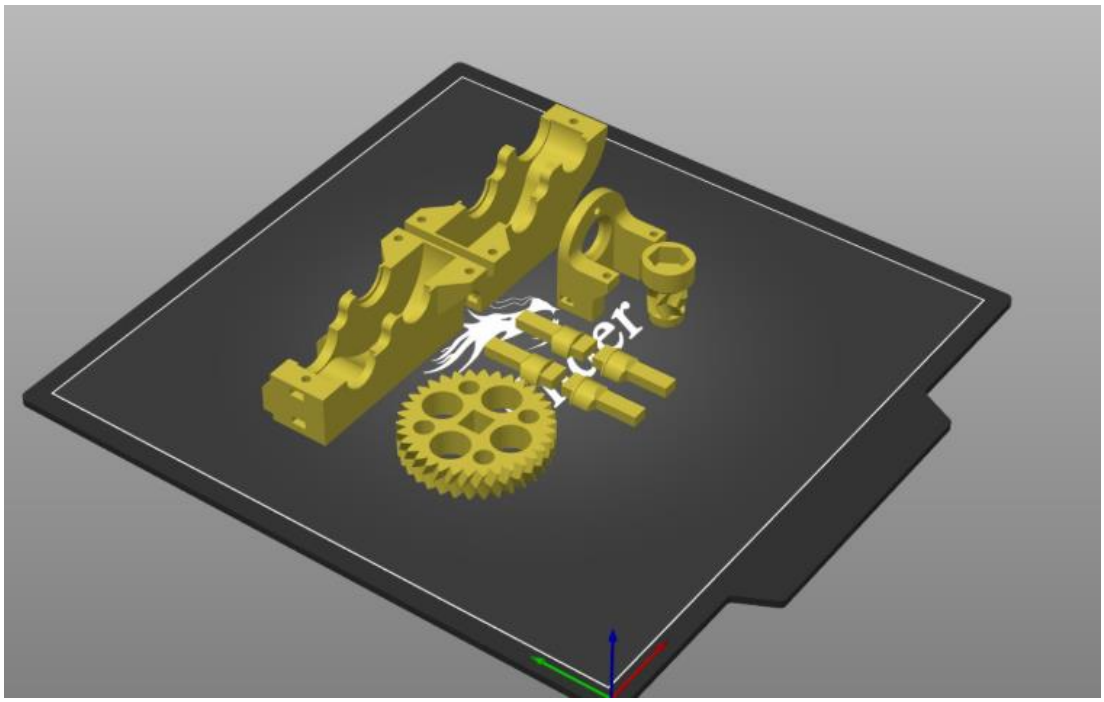
3.2.3. Hareket ve İlaçlama Mekanizması

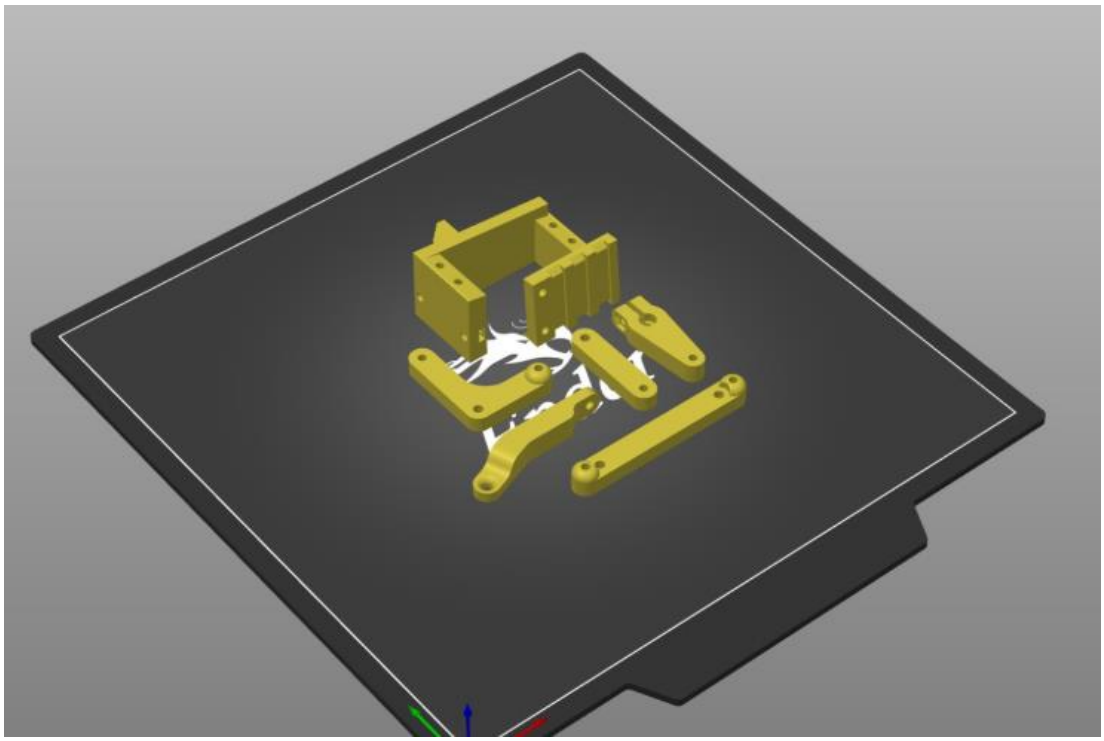
Robotun hareketi diferansiyel sürüş prensibine göre çalışan iki adet DC motor ile sağlanmıştır. İlaçlama işlemi ise röle kontrollü bir pompa yardımıyla gerçekleştirilmiştir.

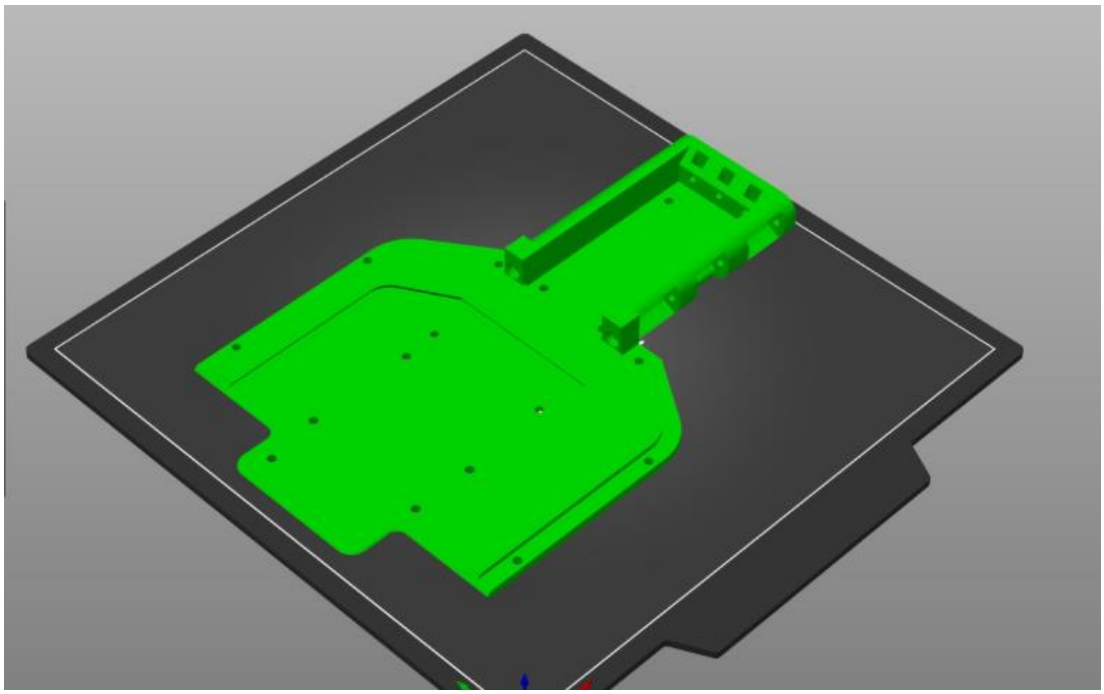
Robotun otonom olarak hareket edebilmesi ve hastalık tespit edilen bölgelerde ilaçlama işlemini gerçekleştirebilmesi için çeşitli mekanik ve elektriksel bileşenler kullanılmıştır. Hareket ve ilaçlama mekanizmasında kullanılan bu bileşenler Tablo 3.3'te verilmiştir.

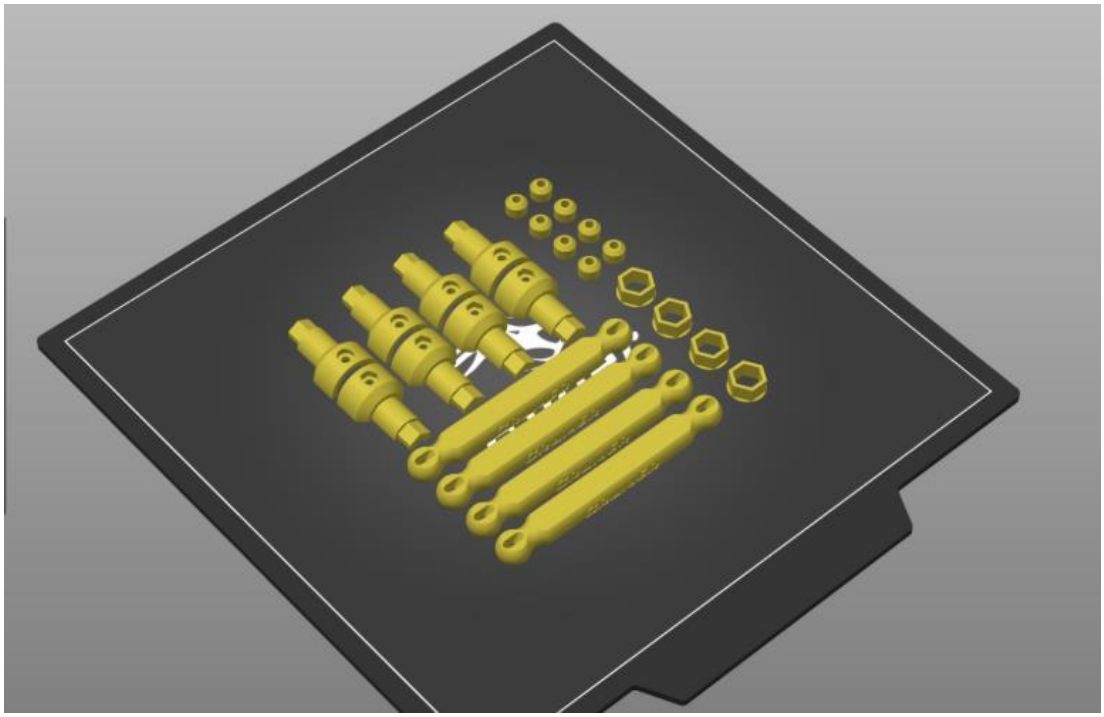
Otonom tarımsal ilaçlama aracının mekanik yapısını oluşturan parçalar, bilgisayar destekli tasarım ortamında modellenmiştir. Hareket ve ilaçlama mekanizmasına ait

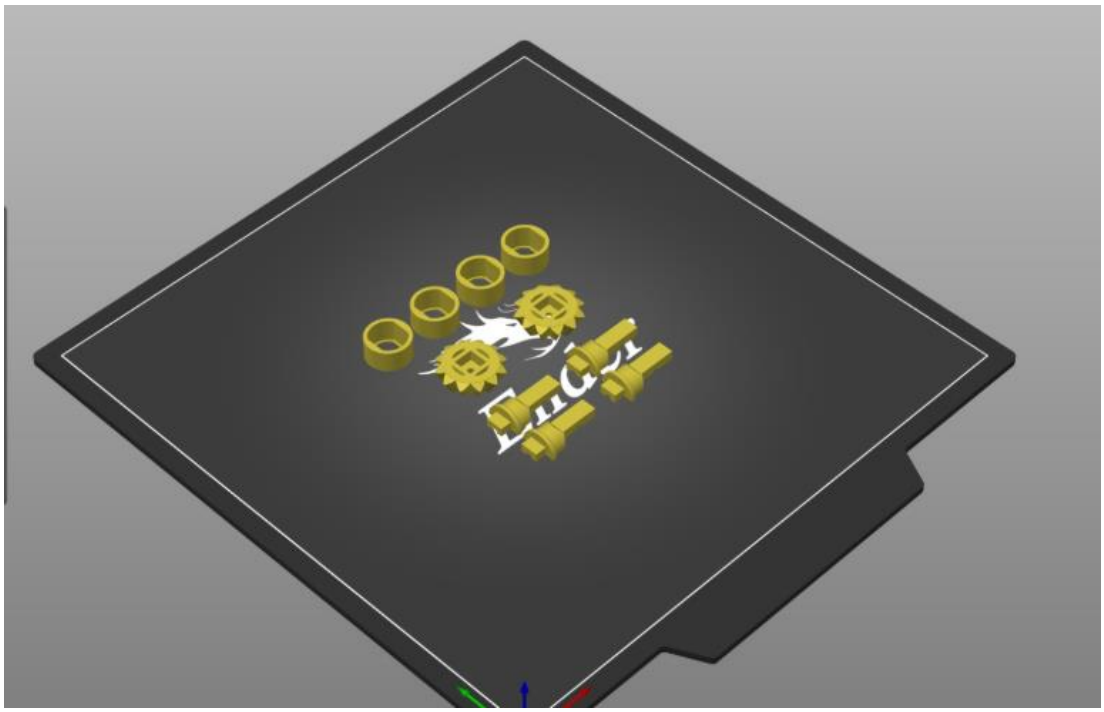
farklı bileşenlerin montaj uyumu ve işlevselliği dikkate alınarak tasarlanan bu parçaların üç boyutlu görünümleri Şekil 3.2'de sunulmaktadır.

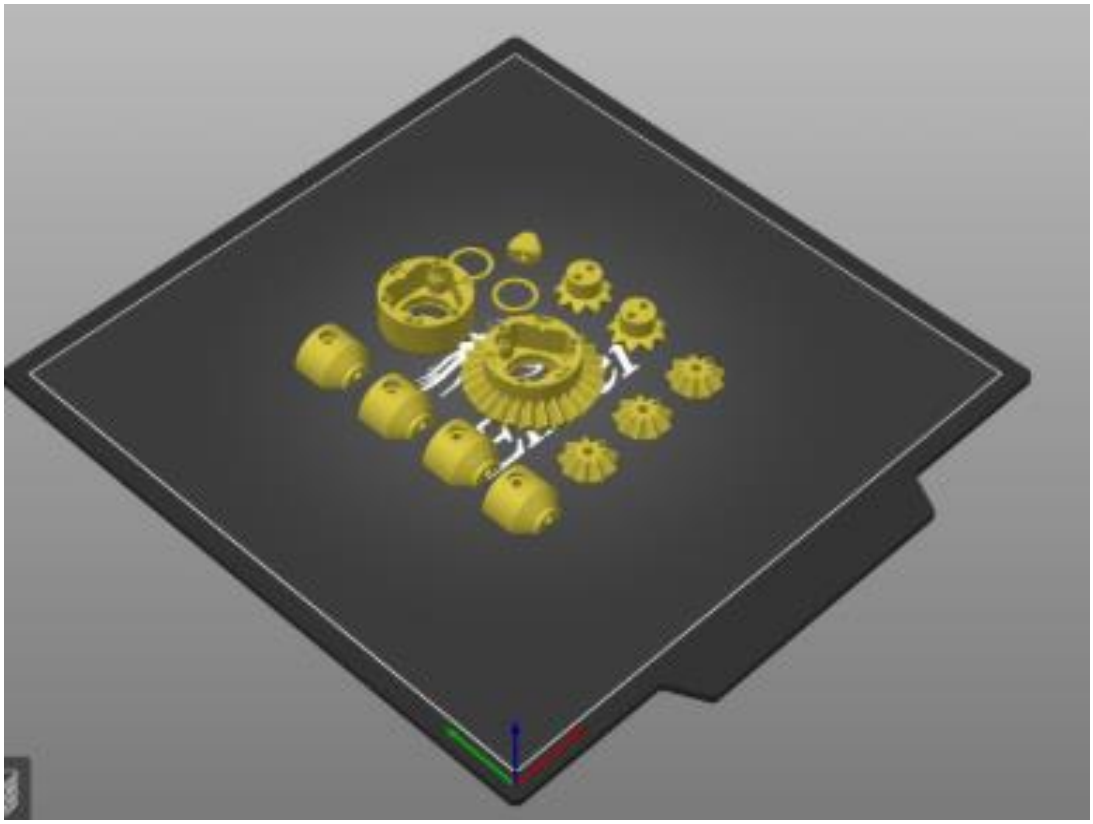


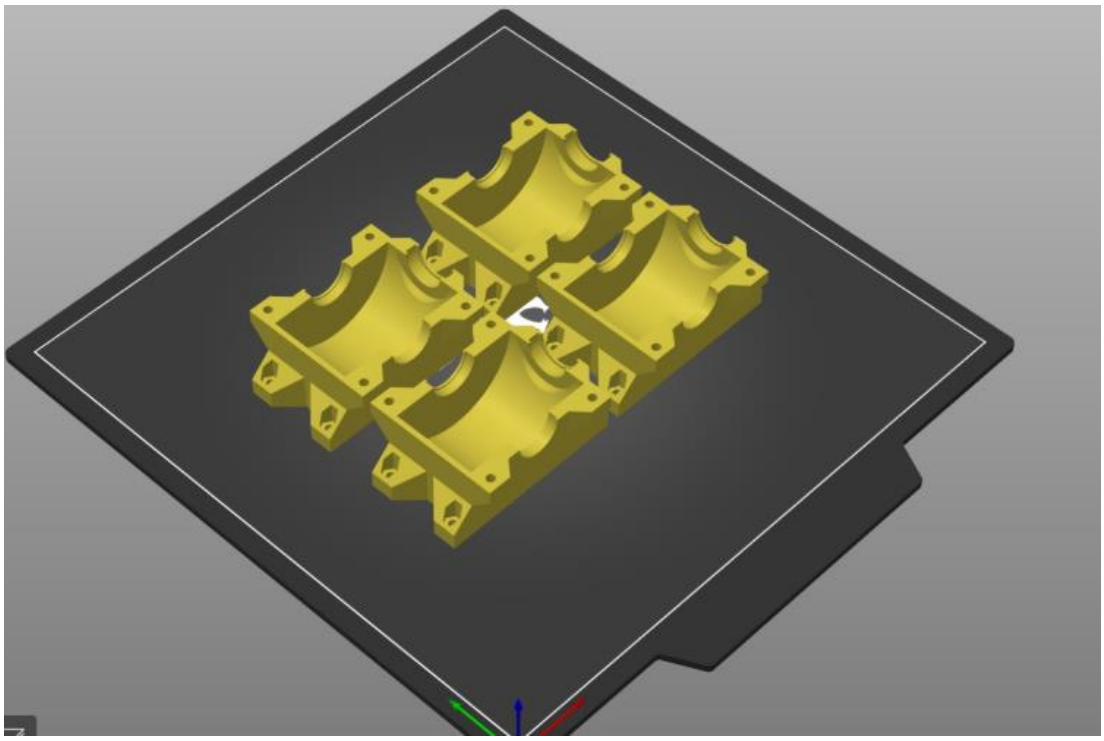


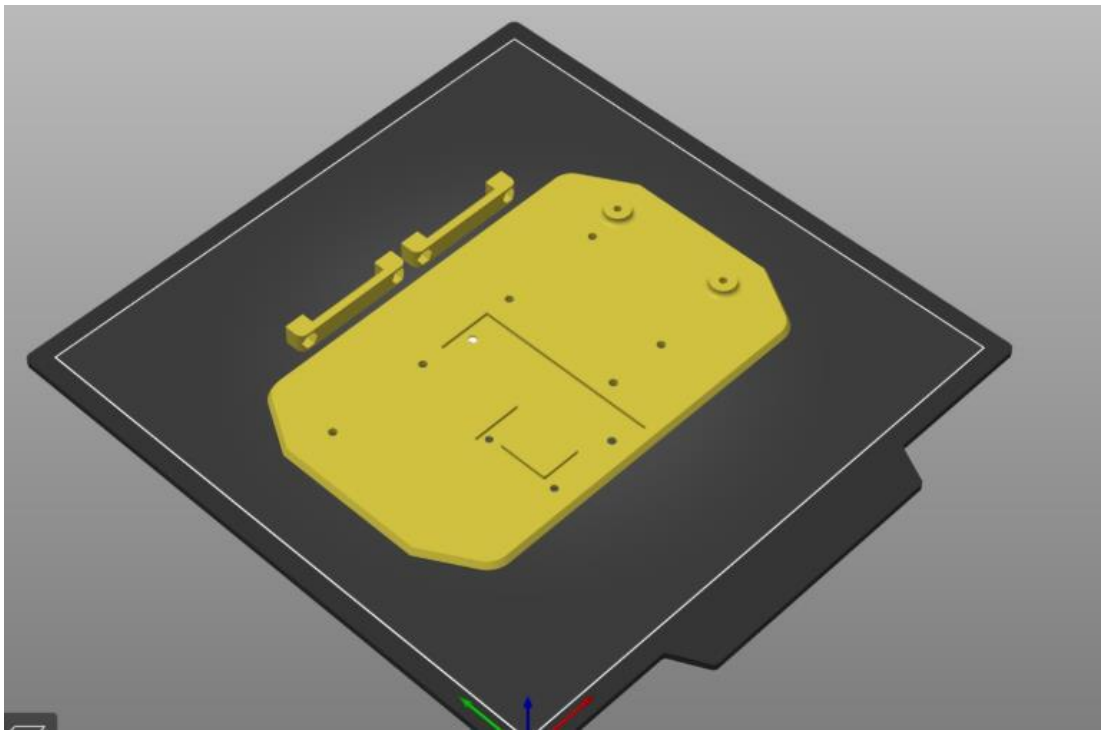


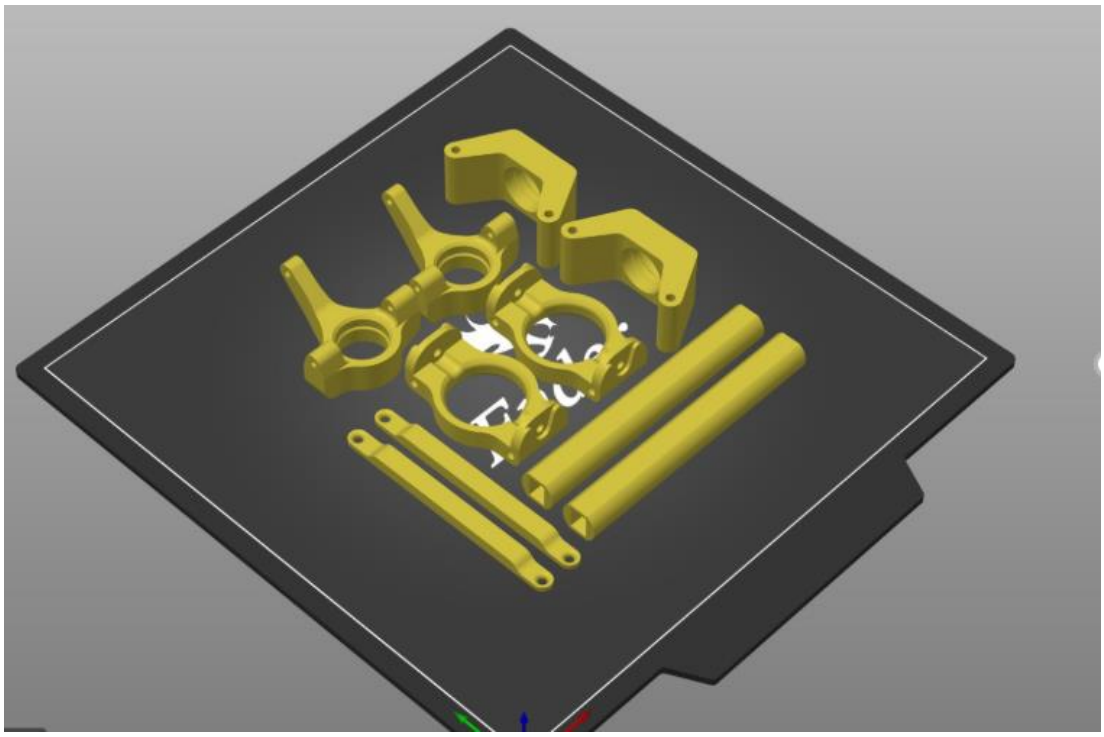


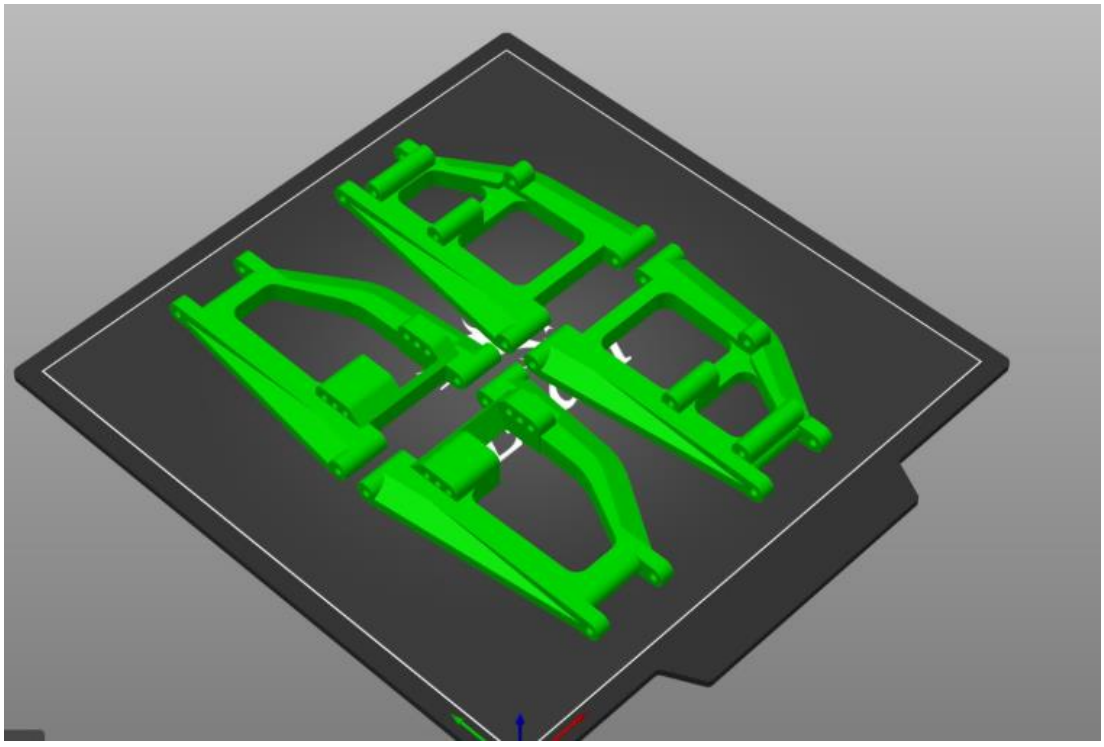


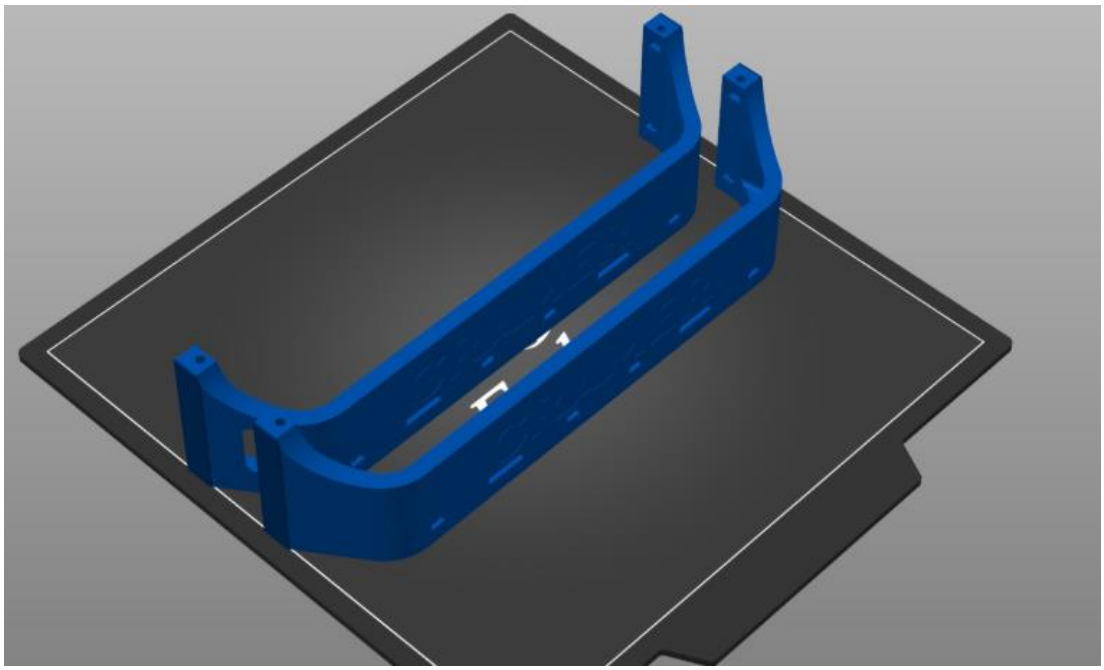


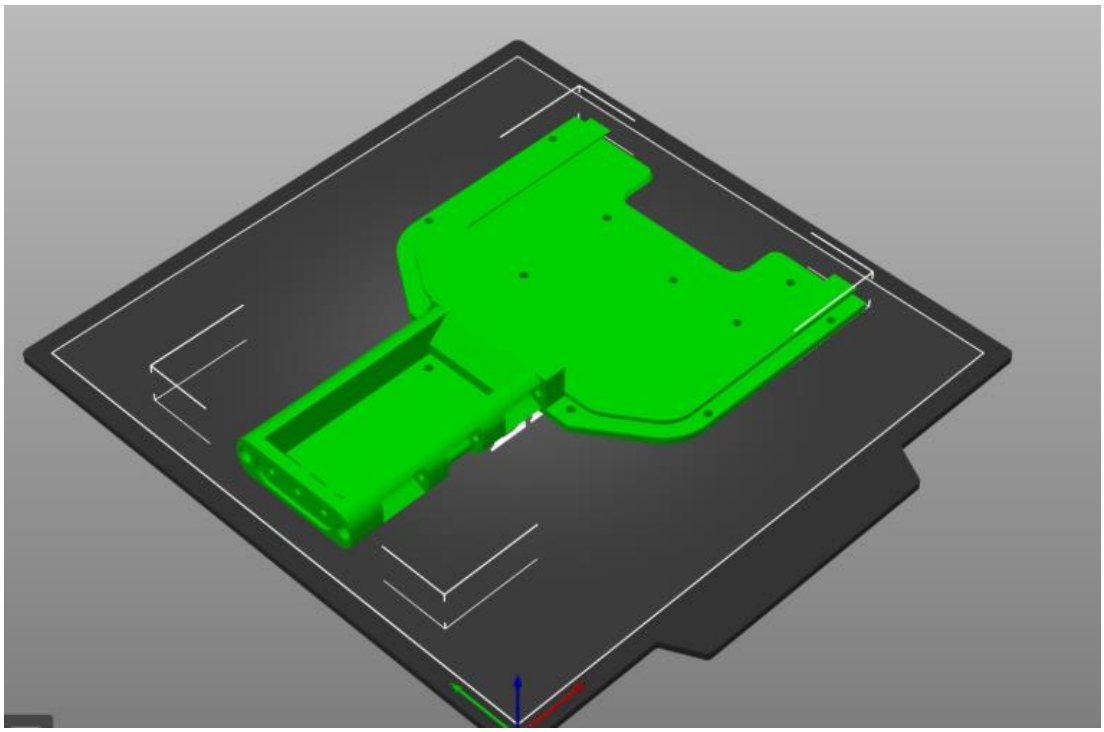


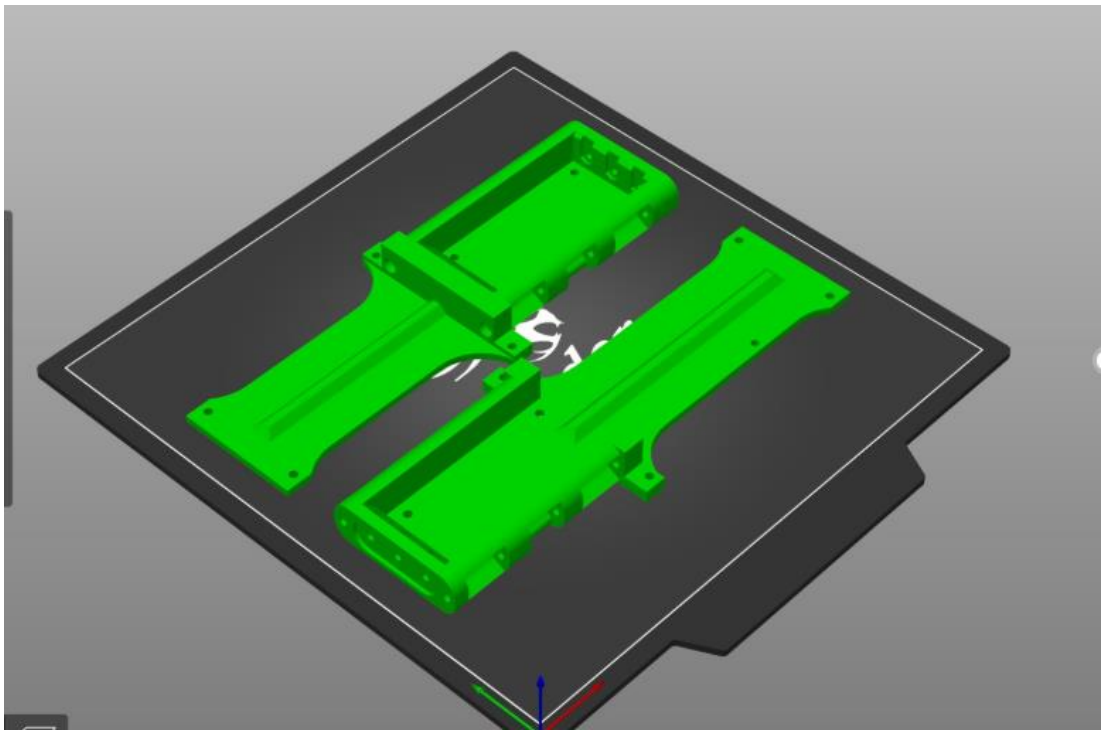












Şekil 3.2. Otonom tarımsal ilaçlama aracına ait mekanik parçaların üç boyutlu CAD görünümleri

Tablo 3.3. Hareket ve İlaçlama Bileşenleri

Bileşen	**Görev**
DC Motorlar	Robot hareketi
Motor Sürücü	Hız ve yön kontrolü
Pompa	Noktasal İlaçlama
Röle	Pompa anahtarlama

3.3. Yazılım Tasarımı

Sistemin yazılım altyapısı Python ve C++ programlama dilleri kullanılarak geliştirilmiştir.

Geliştirilen yazılım mimarisi, modüler ve genişletilebilir bir yapı sunacak şekilde tasarlanmıştır. Benzer yazılım mimarilerinin otonom robot sistemlerinde başarıyla uygulandığı çalışmalar literatürde yer almaktadır [7].

3.3.1. Görüntü İşleme ve Yapay Zekâ Yazılımı

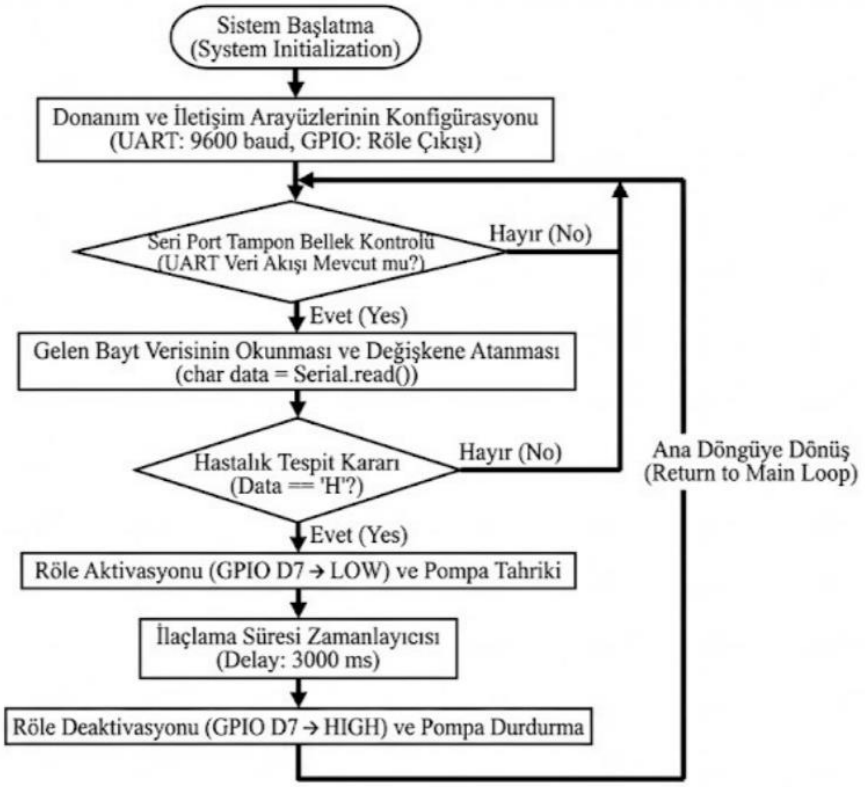
Görüntü işleme algoritmaları Python programlama dili kullanılarak geliştirilmiştir. Bitki hastalıklarının tespiti için YOLOv8n-cls modeli kullanılmıştır. Model, transfer öğrenme yöntemi ile eğitilmiş ve Raspberry Pi üzerinde gerçek zamanlı çalışacak şekilde optimize edilmiştir.

3.3.2. Gömülü Yazılım (Arduino)

Arduino üzerinde çalışan gömülü yazılım, Raspberry Pi'den gelen seri haberleşme komutlarına göre motor ve ilaçlama sistemini kontrol etmektedir. Hastalık tespit edilmesi durumunda robot durdurulmakta ve ilaçlama işlemi başlatılmaktadır.

Arduino tabanlı kontrol yazılımı, Raspberry Pi'den gelen seri haberleşme verilerini sürekli olarak dinlemekte ve gelen komutlara göre ilaçlama sistemini kontrol etmektedir. Şekil 3.4'de, Arduino kontrol algoritmasının akış diyagramı gösterilmektedir.

Arduino kontrol algoritmasının detaylı çalışma adımları, seri haberleşme ve GPIO tabanlı röle kontrolünü içerecek şekilde tasarlanmıştır. Şekil 3.4'de, sistem başlatma, veri okuma, karar verme ve ilaçlama sürecini kapsayan ayrıntılı Arduino yazılım akış diyagramı gösterilmektedir.



Şekil 3.3. Arduino tabanlı ilaçlama sisteminin detaylı yazılım akış diyagramı

3.4. Algoritma Akışı

Sistemin genel algoritma akışı aşağıdaki adımlardan oluşmaktadır:

1. Kamera ve yapay zekâ modelinin başlatılması
2. Kameradan görüntü alınması
3. Görüntü üzerinde yapay zekâ tahmini yapılması
4. Hastalık tespiti durumunun kontrol edilmesi
5. Arduino'ya kontrol komutlarının gönderilmesi
6. Döngünün sürekli olarak devam etmesi

3.5. Veri Seti

Bu çalışmada PlantVillage veri seti ve özgün olarak toplanan görüntüler kullanılmıştır. Veri seti toplam 10 sınıf ve 16.011 görüntüden oluşmaktadır.

Yapay zekâ modelinin eğitimi ve doğrulanması aşamalarında kullanılan veri seti, domates bitkisine ait farklı hastalık sınıflarını ve sağlıklı bitki görüntülerini

içermektedir. Veri setine ait sınıflar ve her sınıfa karşılık gelen görüntü sayıları Tablo 3.4'te gösterilmiştir.

Tablo 3.4. Veri Seti Sınıfları ve Görüntü Sayıları

Sınıf No	**Hastalık Adı**	**Görüntü Sayısı**
0	Bakteriyel Leke	2127
1	Erken Yanıklık	1000
2	Geç Yanıklık	1909
3	Yaprak Küfü	952
4	Septoria Yaprak Lekesi	1771
5	Örümcek Akarı	1676
6	Hedef Leke Hastalığı	1404
7	Sarı Yaprak Kıvırcığı	3208
8	Mozaik Virüsü	373
9	Sağlıklı	1591
Toplam	**10 Sınıf**	16.011

Veri setinde yer alan sınıfların görsel içeriklerinin daha iyi anlaşılabilmesi amacıyla, PlantVillage veri setinden seçilmiş bazı domates yaprağı hastalık örnekleri Şekil 3.6'da sunulmuştur. Bu görüntüler, farklı hastalık türlerinin yaprak üzerindeki karakteristik belirtilerini temsil etmektedir.

Şekil 3.6. PlantVillage veri setinden seçilmiş domates yaprağı hastalık sınıflarına ait örnek görüntüler

3.6. Donanım Bileşenlerinin Seçim Nedenleri

Bu çalışmada kullanılan donanım bileşenleri seçilirken; işlem gücü, gerçek zamanlı çalışma yeteneği, yazılım desteği, maliyet ve sistemin uygulanabilirliği gibi kriterler dikkate alınmıştır. Seçilen donanımların, geliştirilen otonom tarım sistemi ile uyumlu ve sahada kullanılabilir olmasına özen gösterilmiştir.

3.6.1. Raspberry Pi 5 Seçim Gerekçesi

Görüntü işleme ve yapay zekâ tabanlı hastalık tespiti işlemleri, yüksek işlem gücü ve bellek kapasitesi gerektirdiğinden sistemin ana işlem birimi olarak Raspberry Pi 5 tercih edilmiştir. Raspberry Pi 5, önceki nesil Raspberry Pi 4 modeline kıyasla daha güçlü CPU mimarisi ve daha yüksek bellek bant genişliği sunarak gerçek zamanlı görüntü işleme uygulamalarında daha kararlı bir performans sağlamaktadır. Ayrıca Python tabanlı yazılım geliştirme ortamına tam destek

sunması, OpenCV ve YOLO gibi kütüphanelerle uyumlu çalışabilmesi Raspberry Pi 5'in tercih edilmesinde etkili olmuştur.

Alternatif olarak değerlendirilen NVIDIA Jetson Nano, yapay zekâ uygulamalarında yüksek performans sunmasına rağmen maliyetinin daha yüksek olması ve güç tüketiminin fazla olması nedeniyle bu çalışma kapsamında tercih edilmemiştir. ESP32-CAM ise düşük maliyetli bir çözüm olmasına rağmen sınırlı işlem gücü ve bellek kapasitesi nedeniyle derin öğrenme tabanlı gerçek zamanlı görüntü işleme uygulamaları için yetersiz kalmaktadır.

Bu nedenlerle Raspberry Pi 5, fiyat/performans dengesi, yazılım esnekliği ve gerçek zamanlı işlem kapasitesi açısından sistem gereksinimlerini en iyi karşılayan platform olarak seçilmiştir.

3.6.2. Arduino Nano Seçim Gerekçesi

Sistemde servo motor ve eyleyici birimlerin kontrolü için Arduino Nano mikrodenetleyicisi kullanılmıştır. Arduino Nano, küçük boyutu, düşük güç tüketimi ve PWM destekli dijital çıkışları sayesinde servo motor kontrolü için uygun bir platform sunmaktadır. Ayrıca Raspberry Pi ile USB tabanlı seri haberleşmeyi kolaylıkla desteklemesi, sistem entegrasyonunu basitleştirmiştir.

3.6.3. Servo Motor Seçim Gerekçesi

İlaçlama mekanizmasının kontrolü için servo motor tercih edilmiştir. Servo motorlar, belirli açılarda hassas konumlama yapabilmeleri ve basit PWM sinyalleri ile kontrol edilebilmeleri nedeniyle noktasal ilaçlama uygulamaları için uygundur. Seçilen servo motor, düşük gerilimde çalışabilmesi ve hızlı tepki süresi sayesinde sistemin gerçek zamanlı çalışma gereksinimlerini karşılamaktadır.

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu bölümde, geliştirilen görüntü işleme destekli otonom tarım devriye aracının deneysel çalışmaları sonucunda elde edilen bulgular sunulmuş ve bu bulgular literatürdeki benzer çalışmalar ile karşılaştırılarak tartışılmıştır. Yapay zekâ modelinin doğruluk performansı, sistemin gerçek zamanlı çalışma kabiliyeti ve otonom ilaçlama mekanizmasının etkinliği değerlendirilmiştir.

Elde edilen doğruluk ve gecikme süreleri, literatürde rapor edilen benzer çalışmalarla karşılaştırıldığında, geliştirilen sistemin gerçek zamanlı tarımsal uygulamalar için uygun olduğu görülmektedir [8].

4.1. Yapay Zekâ Modeli Performans Sonuçları

Bitki hastalıklarının tespiti amacıyla kullanılan derin öğrenme tabanlı evrişimli sinir ağı modeli, domates bitkisine ait 10 farklı sınıfı içeren veri seti üzerinde eğitilmiştir. Model eğitim süreci, transfer öğrenme yaklaşımı kullanılarak gerçekleştirilmiş ve eğitim 10 epoch boyunca sürdürülmüştür.

Eğitim sonucunda elde edilen performans metrikleri incelendiğinde, modelin Top-1 doğruluk oranının %97,2 olduğu görülmüştür. Kayıp (loss) değerinin ise **0,07** seviyesinde olduğu tespit edilmiştir. Bu değerler, modelin veri seti üzerinde yüksek başarı ile sınıflandırma yapabildiğini göstermektedir.

Eğitilen yapay zekâ modelinin performansı, doğruluk ve kayıp değerleri gibi temel metrikler kullanılarak değerlendirilmiştir. Model eğitimine ait performans sonuçları Tablo 4.1'de sunulmuştur.

Tablo 4.1. Yapay Zekâ Modeline Ait Performans Metrikleri

Parametre	**Değer**
Model	YOLOv8n-cls
Epoch Sayısı	10
Eğitim Süresi	22 Dakika
Donanım	NVIDIA GeForce RTX 4080 GPU
Top-1 Doğruluk	%97,2
Kayıp	0,07

Elde edilen doğruluk oranı, literatürde yer alan benzer bitki hastalığı tespit çalışmalarındaki sonuçlarla karşılaştırıldığında oldukça yüksek bir başarıyı ifade etmektedir. Bu durum, kullanılan veri setinin yeterliliği ve model mimarisinin uygunluğu ile ilişkilendirilmektedir.

4.2. Gerçek Zamanlı Çalışma Performansı

Eğitilen model, Raspberry Pi 5 üzerinde çalıştırılarak gerçek zamanlı performansı değerlendirilmiştir. Yapılan testlerde, sistemin her bir görüntü için ortalama **30 ms** çıkarım (inference) süresine sahip olduğu gözlemlenmiştir. Bu süre, robotun tarım arazisi üzerinde hareket hâlindeyken dahi kesintisiz analiz yapabilmesine olanak tanımaktadır.

Gerçek zamanlı testler sırasında, kamera üzerinden alınan görüntüler anlık olarak işlenmiş ve hastalık tespit edilmesi durumunda sistemin gecikme olmaksızın Arduino mikrodenetleyiciye komut gönderdiği gözlemlenmiştir. Bu sonuç, sistemin pratik tarım uygulamalarında kullanılabilir olduğunu göstermektedir.

Geliştirilen sistem, gerçek zamanlı olarak bitki yapraklarından alınan görüntüler üzerinde hastalık tespiti gerçekleştirebilmektedir. Şekil 4.2'de, sistemin domates yaprağı üzerinde mozaik virüsü hastalığını yüksek güven oranı ile tespit ettiği bir örnek çıktı gösterilmektedir.

Şekil 4.1. Gerçek zamanlı domates yaprağı mozaik virüsü tespit çıktısı

Sistem, farklı hastalık türlerini yüksek doğruluk oranlarıyla ayırt edebilmektedir. Şekil 4.3'te, domates yaprağı üzerinde erken yanıklık (Early Blight) hastalığının gerçek zamanlı olarak ve yüksek güven oranı ile tespit edildiği bir örnek çıktı sunulmuştur.

Şekil 4.2. Gerçek zamanlı domates yaprağı erken yanıklık hastalığı tespit çıktısı

4.3. Otonom İlaçlama Sisteminin Değerlendirilmesi

Sistem testleri sırasında, yapay zekâ modeli tarafından hastalık tespit edilen bitkilerde ilaçlama mekanizmasının başarıyla devreye alındığı görülmüştür. Sağlıklı olarak sınıflandırılan bitkilerde ise herhangi bir ilaçlama işlemi gerçekleştirilmemiştir.

Bu yaklaşım sayesinde yalnızca hastalıklı bölgelerde ilaçlama yapılmış ve gereksiz kimyasal kullanımının önüne geçilmiştir. Noktasal ilaçlama sisteminin çalışması sırasında robotun durması, ilaçlama pompasının belirlenen süre boyunca aktif hâle gelmesi ve ardından sistemin tekrar hareketine devam etmesi başarıyla gerçekleştirilmiştir.

Tablo 4.2. Sistem Test Sonuçları

Geliştirilen sistemin gerçek çalışma koşullarındaki performansını nicel olarak değerlendirmek amacıyla çeşitli deneysel testler gerçekleştirilmiştir. Bu testlerde hastalık tespit doğruluğu, yanlış sınıflandırma oranı, sistemin tepki süresi ve otonom

hareket performansı ölçülmüştür. Elde edilen sayısal sonuçlar, sistemin gerçek zamanlı tarımsal uygulamalar için yeterli performansa sahip olduğunu göstermektedir. Sistem testlerine ait ölçüm sonuçları Tablo 4.2'de sunulmuştur.

Test Kriteri	**Sonuç**
Hastalık Tespiti	%99.2
Yanlış Pozitif Oranı	%2.1
İlaçlama Tepki Süresi	45 ms
Ortalama Çıkarım Süresi	30 ms / görüntü
Otonom Hareket	0.5 m/sn

Tablo 4.2'de sunulan sonuçlar incelendiğinde, geliştirilen sistemin hastalık tespitinde yüksek doğruluk oranına sahip olduğu ve düşük yanlış pozitif oranı ile çalıştığı görülmektedir. Ayrıca hastalık tespiti ile ilaçlama mekanizmasının devreye alınması arasında ölçülen 45 ms'lik gecikme süresi, sistemin gerçek zamanlı çalışmaya uygun olduğunu göstermektedir. Otonom hareket sırasında elde edilen 0,5 m/s hız değeri, robotun tarım arazisi üzerinde kararlı bir şekilde ilerleyebildiğini ortaya koymaktadır.

4.4. Bulguların Tartışılması

Elde edilen deneysel sonuçlar, geliştirilen sistemin bitki hastalıklarının tespiti ve hedefli ilaçlama uygulamaları açısından etkili bir çözüm sunduğunu göstermektedir. Yüksek doğruluk oranı ve düşük çıkarım süresi, sistemin gerçek tarım koşullarında uygulanabilirliğini artırmaktadır.

Literatürde yer alan benzer çalışmalarda genellikle sabit görüntüler veya çevrimdışı analizler kullanılmaktadır. Bu çalışmada ise gerçek zamanlı görüntü işleme ve otonom hareket kabiliyeti bir arada sunularak daha kapsamlı bir çözüm geliştirilmiştir. Bu yönüyle çalışma, akıllı tarım teknolojileri alanında önemli bir katkı sağlamaktadır.

5. SONUÇ VE ÖNERİLER

Bu çalışmada, görüntü işleme ve yapay zekâ destekli otonom bir tarım devriye aracı tasarlanmış ve uygulanmıştır. Geliştirilen sistem; tarım arazisi üzerinde otonom olarak hareket edebilmekte, bitkilerden elde edilen görüntüleri gerçek zamanlı olarak analiz ederek hastalık tespiti yapabilmekte ve yalnızca hastalıklı bölgelerde noktasal ilaçlama gerçekleştirebilmektedir.

Deneysel çalışmalar sonucunda, kullanılan derin öğrenme tabanlı modelin bitki hastalıklarını yüksek doğruluk oranı ile sınıflandırabildiği ve sistemin gerçek zamanlı çalışmaya uygun olduğu görülmüştür. Raspberry Pi 5 üzerinde gerçekleştirilen testlerde, çıkarım süresinin düşük seviyelerde olduğu ve sistemin kararlı bir şekilde çalıştığı gözlemlenmiştir. Bu sonuçlar, geliştirilen sistemin pratik tarım uygulamalarında kullanılabilir olduğunu göstermektedir.

Noktasal ilaçlama yaklaşımı sayesinde gereksiz kimyasal kullanımının önüne geçilmiş, hem ekonomik hem de çevresel açıdan verimli bir çözüm sunulmuştur. Sistem, geleneksel yöntemlere kıyasla daha kontrollü ve sürdürülebilir bir tarımsal üretim sürecine katkı sağlamaktadır.

İlerleyen çalışmalarda, sistemin farklı bitki türleri ve hastalıklar için eğitilmesi, sensör çeşitliliğinin artırılması ve otonom navigasyon yeteneklerinin geliştirilmesi önerilmektedir. Ayrıca, GPS destekli haritalama ve bulut tabanlı veri analizi gibi özelliklerin eklenmesiyle sistemin daha kapsamlı bir tarımsal karar destek platformuna dönüştürülebileceği düşünülmektedir.

KAYNAKLAR

[1] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk ve D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification,"
Computational Intelligence and Neuroscience, vol. 2016, Article ID 3289801, 2016.

[2] H. Mohanty, D. Hughes ve M. Salathé,
"Using Deep Learning for Image-Based Plant Disease Detection,"
Frontiers in Plant Science, vol. 7, pp. 1–10, 2016.

[3] Ultralytics,
"YOLOv8: State-of-the-Art YOLO Models,"
https://docs.ultralytics.com,
Erişim tarihi: 2025.

[4] D. P. Kingma ve J. Ba,
"Adam: A Method for Stochastic Optimization,"
International Conference on Learning Representations (ICLR), 2015.

[5] Hughes, D. P. ve Salathé, M.,
"An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics,"
arXiv preprint arXiv:1511.08060, 2015.

[6] Raspberry Pi Foundation,
"Raspberry Pi 5 Technical Documentation,"
https://www.raspberrypi.com,
Erişim tarihi: 2025.

[7] Arduino,
"Arduino Uno Technical Specifications,"
https://www.arduino.cc,
Erişim tarihi: 2025.

[8] OpenCV Team,
"Open Source Computer Vision Library,"
https://opencv.org,
Erişim tarihi: 2025.

[9] G. Bradski,
"The OpenCV Library,"
Dr. Dobb's Journal of Software Tools, 2000.

[10] FAO,
"Digital Technologies in Agriculture and Rural Areas,"
Food and Agriculture Organization of the United Nations, Rome, 2019.

Kitaplar İçin:

[11] I. Goodfellow, Y. Bengio ve A. Courville,

Deep Learning,

MIT Press, Cambridge, 2016.

[12] R. C. Gonzalez ve R. E. Woods,

Digital Image Processing,

Pearson Education, New Jersey, 2018.

EKLER

EK A. Raspberry Pi Ana Yazılımı

Bu Python kodu, sistemin ana kontrol yazılımı olarak görev yapmaktadır. Kod içerisinde öncelikle Raspberry Pi ile Arduino arasındaki seri haberleşme kurulmakta, ardından eğitilmiş yapay zekâ modeli hafızaya alınarak kamera görüntüleri üzerinde gerçek zamanlı çıkarım işlemi gerçekleştirilmektedir. Yapay zekâ modelinden elde edilen sonuçlar, belirlenen güven eşiğine göre değerlendirilmekte ve bitki yaprağının sağlıklı veya hastalıklı olma durumuna göre Arduino'ya uygun kontrol komutları gönderilmektedir. Ayrıca, sistemin kullanıcı tarafından izlenebilmesi amacıyla tespit edilen sınıf bilgileri ve güven skorları canlı kamera görüntüsü üzerinde görselleştirilmektedir.

```
import cv2
from ultralytics import YOLO
import serial
import time

# 1. SERİ HABERLEŞME KURULUMU
# Arduino'nun bağlı olduğu port ve veri hızı tanımlanır.
try:
    arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    time.sleep(2) # Bağlantının oturması için bekleme süresi
    print("Arduino Bağlantısı Başarılı.")
except:
    arduino = None
    print("UYARI: Arduino Bulunamadı! Sistem sadece simülasyon
modunda çalışacak.")

# 2. YAPAY ZEKÂ MODELİNİN YÜKLENMESİ
# Transfer öğrenme ile eğitilmiş .pt uzantılı model hafızaya
alınır.
model = YOLO('best.pt')

# 3. KAMERA BAŞLATMA
cap = cv2.VideoCapture(0) # 0: Varsayılan USB/CSI Kamera
cap.set(3, 640) # Çözünürlük Genişlik Ayarı
cap.set(4, 480) # Çözünürlük Yükseklik Ayarı

# 4. EŞİK DEĞERİ (THRESHOLD)
GUVEN_LIMITI = 0.60   # %60'ın altındaki tahminler dikkate
alınmaz.

while True:
    success, img = cap.read() # Kameradan anlık kare oku
    if not success:
        break

    # 5. MODEL ÇIKARIMI (INFERENCE)
    results = model(img, stream=True, verbose=False)
```

```
    # 6. SONUÇLARIN ANALİZİ
    for r in results:
        boxes = r.boxes
        probs = r.probs # Sınıflandırma olasılıkları

        # En yüksek olasılığa sahip sınıfı bul
        if probs is not None:
            top1_index = probs.top1
            conf = probs.top1conf.item() # Güven skoru
            class_name = r.names[top1_index] # Sınıf adı (Örn:
Tomato_Bacterial_Spot)

            # Ekrana Yazdırma İşlemleri
            label = f"{class_name} ({conf*100:.1f}%)"

            # 7. KARAR MEKANİZMASI VE ARDUINO HABERLEŞMESİ
            if conf > GUVEN_LIMITI:
                if "healthy" in class_name.lower():
                    # Durum: SAĞLIKLI -> İlerle
                    color = (0, 255, 0) # Yeşil Renk
                    if arduino:
                        arduino.write(b'I') # 'I' karakterini
gönder (İlerle)
                else:
                    # Durum: HASTALIKLI -> Dur ve İlaçla
                    color = (0, 0, 255) # Kırmızı Renk
                    if arduino:
                        arduino.write(b'H') # 'H' karakterini
gönder (Hastalık)

            # Görüntü üzerine yazı yazma (Görselleştirme)
            cv2.putText(img,      label,      (20,      50),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)

    # 8. GÖRÜNTÜLEME PENCERESİ
    cv2.imshow('Otonom Tarim Robotu - Kamera', img)

    # 'q' tuşuna basılırsa çıkış yap
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

EK B. Arduino Gömülü Kontrol Yazılımı

Bu ekte, otonom tarım robotunun eyleyici birimlerini kontrol eden Arduino tabanlı gömülü yazılım sunulmuştur. Yazılım, Raspberry Pi'den seri haberleşme yoluyla gelen komutları okuyarak robotun hareket etmesini ve ilaçlama

mekanizmasının çalıştırılmasını sağlamaktadır. Sistem, diferansiyel sürüş mantığına sahip DC motorlar ve servo motor kontrollü ilaçlama mekanizmasından oluşmaktadır.

```
          #include <Servo.h>


Servo ilaclamaServosu;

const int SERVO_PIN = 9;


const int IN1 = 4;

const int IN2 = 5;

const int IN3 = 6;

const int IN4 = 7;

const int ENA = 3;

const int ENB = 11;


char gelen_veri;


void setup() {

  Serial.begin(9600);


  ilaclamaServosu.attach(SERVO_PIN);

  ilaclamaServosu.write(0);


  pinMode(IN1, OUTPUT);

  pinMode(IN2, OUTPUT);

  pinMode(IN3, OUTPUT);

  pinMode(IN4, OUTPUT);
```

```
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  robotu_durdur();
}

void loop() {
  if (Serial.available() > 0) {
    gelen_veri = Serial.read();

    if (gelen_veri == 'I') {
      ileri_git(150);
      ilaclamaServosu.write(0);
    }
    else if (gelen_veri == 'H') {
      robotu_durdur();
      delay(500);
      ilaclama_yap();
    }
  }
}

void ileri_git(int hiz) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, hiz);
```

```
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, hiz);
}

void robotu_durdur() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}

void ilaclama_yap() {
  ilaclamaServosu.write(180);
  delay(3000);

  ilaclamaServosu.write(0);
  delay(1000);
}
```

EK C. Model Eğitim Yazılımı

Aşağıda verilen kod, YOLOv8n-cls mimarisi kullanılarak sınıflandırma modelinin eğitimini gerçekleştirmektedir. Eğitim parametreleri (epoch sayısı, batch değeri, görüntü boyutu ve erken durdurma gibi) deneysel koşullara göre belirlenmiş ve eğitim sonuçları belirlenen proje klasörüne kaydedilmiştir. Eğitim tamamlandıktan sonra model doğrulaması yapılmış ve elde edilen model ağırlıkları sistemde kullanılmak üzere dışa aktarılmıştır.

```
from ultralytics import YOLO import torch

def egitim_suredurunu_baslat(): model = YOLO('yolov8n-cls.pt')

results = model.train(
    data='C:/Tez_calisma/VeriSeti',
    epochs=50,
    imgsz=224,
    batch=16,
    device=0,
    project='Tez_Sonuclari',
    name='Domates_Hastalik_Modeli',
    patience=10
)

metrics = model.val()
success = model.export(format='pt')


if name == 'main': torch.cuda.empty_cache()
egitim_suredurunu_baslat()
```

EK D. Arduino Hareket ve İlaçlama Kontrol Yazılımı

Bu ekte, otonom tarım robotunun hareket ve ilaçlama mekanizmasını kontrol eden Arduino tabanlı gömülü yazılım sunulmuştur. Yazılım, Raspberry Pi tarafından seri haberleşme yoluyla gönderilen komutları okuyarak robotun ileri hareket etmesini veya durarak ilaçlama işlemini gerçekleştirmesini sağlamaktadır. Sistem, diferansiyel sürüş prensibine göre çalışan DC motorlar ve servo motor kontrollü bir ilaçlama mekanizmasından oluşmaktadır.

```
#include <Servo.h>



Servo ilaclamaServosu;



const int SERVO_PIN = 9;

const int IN1 = 4;

const int IN2 = 5;
```

```
const int IN3 = 6;

const int IN4 = 7;

const int ENA = 3;

const int ENB = 11;


char gelen_veri;


void setup() {
  Serial.begin(9600);


  ilaclamaServosu.attach(SERVO_PIN);
  ilaclamaServosu.write(0);


  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);


  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}
```

```
void loop() {
  if (Serial.available() > 0) {
    gelen_veri = Serial.read();

    if (gelen_veri == 'I') {
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      analogWrite(ENA, 150);

      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      analogWrite(ENB, 150);

      ilaclamaServosu.write(0);
    }
    else if (gelen_veri == 'H') {
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
      analogWrite(ENA, 0);
      analogWrite(ENB, 0);

      delay(500);

      ilaclamaServosu.write(180);
```

```
      delay(3000);


      ilaclamaServosu.write(0);
      delay(1000);
    }
  }
}
```

EK E. Python Kamera Donanım Test Yazılımı

Bu ekte, otonom tarım robotunda kullanılan kamera donanımının doğru şekilde çalıştığını doğrulamak amacıyla geliştirilen Python tabanlı test yazılımı sunulmuştur. Yazılım, OpenCV kütüphanesi kullanılarak kamera sürücüsünün başlatılmasını, görüntü akışının alınmasını ve operatör tarafından görsel olarak kontrol edilmesini sağlamaktadır. Bu test, yapay zekâ tabanlı analiz sürecine geçilmeden önce kamera donanımının sistemle uyumlu çalıştığının doğrulanması amacıyla gerçekleştirilmiştir.

Aşağıda verilen Python yazılımı, Raspberry Pi veya bilgisayara bağlı USB/CSI kameradan gerçek zamanlı görüntü alınmasını sağlamaktadır. Kamera çözünürlüğü, sistemde kullanılan yapay zekâ modelinin giriş boyutuna uygun olacak şekilde ayarlanmış olup, kullanıcı tarafından manuel olarak sonlandırılabilen bir test döngüsü içermektedir. Bu yazılım, sistemin ön hazırlık ve donanım doğrulama aşamasında kullanılmıştır.

```
import cv2

def kamera_donanim_testi(): cap = cv2.VideoCapture(0)

cap.set(3, 640)
cap.set(4, 480)

if not cap.isOpened():
    print("HATA: Kamera sensörü algılanamadı!")
    return

print("Kamera Testi Baslatildi. Cikis icin 'q' tusuna
basiniz.")

while True:
    success, img = cap.read()
```

```
    if success:
        cv2.imshow("Sistem On Hazirlik - Kamera Testi", img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()


if name == 'main': kamera_donanim_testi()
```

EK F. UART Seri Haberleşme Test Kodu

Bu ekte, Raspberry Pi ile Arduino arasında kurulan seri (UART) haberleşmenin doğruluğunu test etmek amacıyla kullanılan Python tabanlı yazılım sunulmuştur. Yazılım, seri port üzerinden Arduino'ya komut gönderilmesini ve Arduino'dan geri dönen verilerin okunmasını sağlamaktadır. Bu test, otonom sistemde kullanılan haberleşme altyapısının güvenilirliğini doğrulamak amacıyla gerçekleştirilmiştir.

```
import serial import time

def seri_port_testi(): try: arduino =
serial.Serial('/dev/ttyACM0', 9600, timeout=1)
arduino.reset_input_buffer() print("Seri Port Baglantisi
Basarili: /dev/ttyACM0") except: print("HATA: Arduino
baglantisi saglanamadi!") return

while True:
    komut = input("Arduino'ya gonderilecek komutu girin (H/I):
")
    arduino.write(komut.encode('utf-8'))
    time.sleep(0.1)

    if arduino.in_waiting > 0:
        cevap = arduino.readline().decode('utf-8').rstrip()
        print(f"Arduino'dan Gelen Cevap: {cevap}")


if name == 'main': seri_port_testi()
```

EK G. Python Sistem Bağımlılık ve Sağlık Kontrol Yazılımı

Bu ekte, otonom tarım robotu yazılım altyapısının çalışmaya hazır olup olmadığını kontrol etmek amacıyla geliştirilen Python tabanlı sistem sağlık tarama yazılımı sunulmuştur. Yazılım, sistemde gerekli olan yazılım kütüphanelerinin yüklü olup olmadığını denetlemekte ve donanım hızlandırma (GPU) desteğinin mevcut olup olmadığını kontrol etmektedir. Bu kontrol, sistemin kararlı ve verimli bir şekilde çalışabilmesi için ön koşul niteliği taşımaktadır.

Aşağıda verilen Python kodu, otonom robot sisteminde kullanılan temel yazılım bileşenlerinin eksiksiz olarak yüklü olup olmadığını kontrol etmektedir. Ayrıca, derin öğrenme işlemlerinde hızlandırma sağlayan GPU desteğinin mevcut olup olmadığı denetlenmekte ve sistemin başlatılmaya hazır olup olmadığı kullanıcıya bildirilmektedir. Bu yazılım, sistem kurulumu ve hata ayıklama aşamalarında kullanılmıştır.

```
import importlib.util import sys import torch import
subprocess

def sistem_saglik_taramasi(): gerekli_kutuphaneler = {
'opencv-python': 'cv2', 'ultralytics': 'ultralytics',
'pyserial': 'serial', 'numpy': 'numpy', 'torch': 'torch' }

print("--- OTONOM ROBOT SISTEM SAGLIK TARAMASI ---\n")

tum_hazir = True

for paket, modul in gerekli_kutuphaneler.items():
    spec = importlib.util.find_spec(modul)
    if spec is None:
        print(f"[EKSIK] {paket} kütüphanesi yüklü değil.")
        tum_hazir = False
    else:
        print(f"[TAMAM] {paket} başarıyla yüklendi.")

print("\n--- DONANIM HIZLANDIRMA KONTROLU ---")

if torch.cuda.is_available():
    print(f"[GPU] NVIDIA CUDA Aktif. Cihaz:
{torch.cuda.get_device_name(0)}")
else:
    print("[CPU] UYARI: GPU bulunamadı. İşlemler işlemci
üzerinde yavaş çalışacak.")

if tum_hazir:
    print("\nSONUÇ: Sistem başlatılmaya hazır.")
else:
    print("\nSONUÇ: Eksik paketler var. Lütfen 'pip install -r
requirements.txt' komutunu çalıştırın.")


if name == 'main': sistem_saglik_taramasi()
```

EK H. Python Model Doğrulama ve Çıkarım Test Yazılımı

Bu ekte, eğitimi tamamlanan yapay zekâ modelinin doğruluğunu ve çıkarım (inference) kabiliyetini test etmek amacıyla kullanılan Python tabanlı yazılım sunulmuştur. Yazılım, eğitilmiş YOLOv8 sınıflandırma modelini kullanarak test veri setinden seçilen bir görüntü üzerinde tahmin işlemi gerçekleştirmekte ve elde edilen

sonuçları hem sayısal hem de görsel olarak kullanıcıya sunmaktadır. Bu test, modelin eğitim sonrası performansının doğrulanması amacıyla gerçekleştirilmiştir.

Aşağıda verilen Python yazılımı, eğitilmiş model ağırlıklarını yükleyerek test görüntüsü üzerinde sınıflandırma işlemi yapmaktadır. Tahmin edilen sınıf bilgisi ve güven skoru konsol çıktısı olarak sunulmakta, aynı zamanda sonuç görselleştirilerek modelin karar mekanizması kullanıcı tarafından gözlemlenebilmektedir. Bu yazılım, model performansının nitel olarak değerlendirilmesinde kullanılmıştır.

```
from ultralytics import YOLO import cv2 import
matplotlib.pyplot as plt

def model_dogrulama_testi(): model_yolu =
'Tez_Sonuclari/Domates_Hastalik_Modeli/weights/best.pt' model
= YOLO(model_yolu)

test_resmi = 'VeriSeti/test/Tomato_Bacterial_spot/001.jpg'

results = model(test_resmi)

for r in results:
    im_array = r.plot()

    probs = r.probs
    sinif_adi = r.names[probs.top1]
    guven_skoru = float(probs.top1conf)

    print(f"Tahmin Edilen Sinif: {sinif_adi}")
    print(f"Guven Skoru (Confidence): {guven_skoru:.4f}")

    plt.imshow(cv2.cvtColor(im_array, cv2.COLOR_BGR2RGB))
    plt.title(f"Sonuc: {sinif_adi} ({guven_skoru:.2%})")
    plt.axis('off')
    plt.show()


if name == 'main': model_dogrulama_testi()
```

EK I. Python Veri Ön İşleme ve Veri Seti Bölümleme Yazılımı

Bu ekte, çalışmada kullanılan görüntü veri setinin eğitim, doğrulama ve test kümelerine ayrılması amacıyla geliştirilen Python tabanlı veri ön işleme yazılımı sunulmuştur. Yazılım, ham görüntü verilerini sınıf bazında rastgele karıştırarak belirlenen oranlara göre otomatik olarak bölümlendirmekte ve derin öğrenme

modelinin eğitimi için uygun bir klasör yapısı oluşturmaktadır. Bu işlem, modelin genelleme yeteneğini artırmak ve deneysel sonuçların güvenilirliğini sağlamak amacıyla gerçekleştirilmiştir.

Aşağıda verilen Python yazılımı, ham veri klasöründe bulunan görüntüleri okuyarak eğitim, doğrulama ve test kümelerine ayırmaktadır. Bölümleme işlemi rastgele gerçekleştirilmiş olup, her sınıf için aynı oranlar korunmuştur. Elde edilen veri seti yapısı, YOLOv8 tabanlı derin öğrenme modellerinin eğitim sürecine doğrudan uyumlu olacak şekilde düzenlenmiştir.

```
import os import shutil import random

def veri_setini_hazirla(): kaynak_klasor = 'Ham_Veriler'
hedef_klasor = 'VeriSeti'

siniflar = os.listdir(kaynak_klasor)

egitim_orani = 0.70
val_orani = 0.20

for sinif in siniflar:
    sinif_yolu = os.path.join(kaynak_klasor, sinif)

    if not os.path.isdir(sinif_yolu):
        continue

    resimler = os.listdir(sinif_yolu)
    random.shuffle(resimler)

    toplam_resim = len(resimler)
    egitim_bitis = int(toplam_resim * egitim_orani)
    val_bitis = int(toplam_resim * (egitim_orani + val_orani))

    egitim_resimleri = resimler[:egitim_bitis]
    val_resimleri = resimler[egitim_bitis:val_bitis]
    test_resimleri = resimler[val_bitis:]

    klasor_yapisi = {
        'train': egitim_resimleri,
        'val': val_resimleri,
        'test': test_resimleri
    }

    for kume_adi, kume_resimleri in klasor_yapisi.items():
        hedef_yol = os.path.join(hedef_klasor, kume_adi,
sinif)
        os.makedirs(hedef_yol, exist_ok=True)
```

```
        for resim in kume_resimleri:
            src = os.path.join(sinif_yolu, resim)
            dst = os.path.join(hedef_yol, resim)
            shutil.copy(src, dst)

    print(f"Sinif Islendi: {sinif} | Toplam: {toplam_resim}
adet")


if name == 'main': veri_setini_hazirla()
```

EK J. Python Temel Model Mimarisi ve Analiz Yazılımı

Bu ekte, çalışmada kullanılan YOLOv8n-cls derin öğrenme modelinin temel mimari özelliklerini incelemek ve modelin yapısal analizini gerçekleştirmek amacıyla geliştirilen Python tabanlı yazılım sunulmuştur. Yazılım, modelin katman yapısını, toplam parametre sayısını ve tahmini bellek gereksinimini analiz ederek modelin donanım üzerindeki yükünü değerlendirmeyi amaçlamaktadır. Bu analiz, modelin gömülü sistemlerde kullanılabilirliğini ortaya koymak açısından önem taşımaktadır.

Aşağıda verilen Python yazılımı, YOLOv8 Nano sınıflandırma modelini yükleyerek modelin görev türünü, parametre sayısını ve bellek kullanımını raporlamaktadır. Elde edilen bilgiler, modelin Raspberry Pi gibi sınırlı donanım kaynaklarına sahip sistemlerde çalıştırılabilirliğini değerlendirmek amacıyla kullanılmıştır.

```
from ultralytics import YOLO

def temel_model_analizi(): model = YOLO('yolov8n-cls.pt')

print("--- YOLOv8-NANO MODEL MIMARI RAPORU ---\n")

model.info(detailed=True)

print(f"\nModel Gorevi (Task): {model.task.upper()}")

model_parametresi = sum(p.numel() for p in model.parameters())
model_boyutu_mb = model_parametresi * 4 / (1024 * 1024)

print(f"Toplam Parametre Sayisi: {model_parametresi:,}")
print(f"Tahmini Bellek Boyutu: {model_boyutu_mb:.2f} MB")

if model.task == 'classify':
    print("\nSONUC: Bu model 'Siniflandirma' (Classification)
icin optimize edilmistir.")
    print("Nesne Tespiti (Detection) kutucuklari cizmez,
```

```
sadece etiket tahmini yapar.")


if name == 'main': temel_model_analizi()
```

EK K. Python Sistem Hata Ayıklama ve Tanılama Yazılımı

Bu ekte, otonom tarım robotunun sahada çalıştırılması sırasında oluşabilecek yazılım ve donanım kaynaklı sorunların hızlı şekilde tespit edilebilmesi amacıyla geliştirilen Python tabanlı hata ayıklama ve tanılama yazılımı sunulmuştur. Yazılım; model dosyasının varlığını kontrol etmekte, kamera akışını doğrulamakta, yapay zekâ modelinin yüklenmesini test etmekte ve Arduino seri port bağlantısını denetlemektedir. Böylece sistemin operasyon öncesi "hazır/arıza" durum değerlendirmesi otomatik olarak yapılabilmektedir.

Aşağıda verilen Python yazılımı, sistemin kritik bileşenleri için adım adım kontrol gerçekleştirerek tanılama raporu üretmektedir. Kontrol edilen bileşenler; dosya sistemi (model ağırlık dosyası), kamera sürücüsü ve görüntü akışı, YOLO modelinin RAM'e yüklenmesi ve seri port üzerinden Arduino bağlantısıdır. Test sonuçları, kullanıcıya metin tabanlı bir rapor olarak sunulmakta ve sistemin operasyona hazır olup olmadığı belirtilmektedir.

```
import cv2 import serial import os import sys from ultralytics
import YOLO import time

def hata_ayiklama_modulu(): print("--- OTONOM ROBOT HATA
AYIKLAMA VE TESHIS PROTOKOLU ---\n")

rapor = {
    "DosyaSistemi": "Bilinmiyor",
    "Kamera": "Bilinmiyor",
    "YapayZeka": "Bilinmiyor",
    "Arduino": "Bilinmiyor"
}

if os.path.exists('best.pt'):
    print("[OK] Model dosyasi (best.pt) diskte mevcut.")
    rapor["DosyaSistemi"] = "BASARILI"
else:
    print("[HATA] KRITIK: 'best.pt' dosyasi bulunamadi!")
    rapor["DosyaSistemi"] = "BASARISIZ"

try:
    cap = cv2.VideoCapture(0)
    if cap.isOpened():
        ret, frame = cap.read()
```

```
        if ret:
            print("[OK] Kamera sensörü aktif ve veri akisi
var.")
            rapor["Kamera"] = "BASARILI"
        else:
            print("[HATA] Kamera acildi fakat görüntü (Frame)
alinamadi.")
            rapor["Kamera"] = "KISMI HATA"
    else:
        print("[HATA] Kamera aygiti (/dev/video0)
baslatilamadi.")
        rapor["Kamera"] = "BASARISIZ"
    cap.release()
except Exception as e:
    print(f"[HATA] Kamera sürücü hatasi: {e}")
    rapor["Kamera"] = "KRITIK HATA"

if rapor["DosyaSistemi"] == "BASARILI":
    try:
        print("[BILGI] Model RAM bellege yükleniyor...")
        model = YOLO('best.pt')
        print("[OK] YOLOv8 modeli hatasiz yüklendi.")
        rapor["YapayZeka"] = "BASARILI"
    except Exception as e:
        print(f"[HATA] Model yükleme hatasi (CUDA/CPU): {e}")
        rapor["YapayZeka"] = "BASARISIZ"

try:
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    time.sleep(1)
    ser.close()
    print("[OK] Arduino seri port baglantisi saglandi.")
    rapor["Arduino"] = "BASARILI"
except serial.SerialException:
    print("[UYARI] Arduino baglanamadi. Kabloyu veya portu
kontrol edin.")
    rapor["Arduino"] = "BAGLANTI YOK"

print("\n--- NIHAI TESHIS RAPORU ---")
for bilesen, durum in rapor.items():
    print(f"{bilesen.ljust(15)}: {durum}")

if all(stat == "BASARILI" for stat in rapor.values()):
    print("\nSISTEM DURUMU: OPERASYONA TAMAMEN HAZIR.")
else:
    print("\nSISTEM DURUMU: ARIZA MEVCUT. BAKIM GEREKLI.")
```

```
if name == 'main': hata_ayiklama_modulu()
```

EK L. Python Gerçek Zamanlı Görüntü Kalibrasyon Yazılımı

Bu ekte, otonom tarım robotunda kullanılan kamera görüntüsünün gerçek zamanlı olarak ayarlanabilmesi amacıyla geliştirilen Python tabanlı görüntü kalibrasyon yazılımı sunulmuştur. Yazılım, kamera parlaklık, kontrast ve doygunluk parametrelerinin kullanıcı tarafından manuel olarak ayarlanmasına imkân tanımaktadır. Bu kalibrasyon işlemi, farklı çevresel ışık koşullarında görüntü kalitesinin optimize edilmesi ve yapay zekâ modelinin daha doğru sonuçlar üretmesi amacıyla kullanılmıştır.

Aşağıda verilen Python yazılımı, OpenCV kütüphanesi kullanılarak oluşturulmuş bir kullanıcı arayüzü aracılığıyla kamera parametrelerinin anlık olarak değiştirilmesini sağlamaktadır. Yapılan ayarlamalar eş zamanlı olarak canlı görüntü üzerinde gözlemlenebilmekte ve sistem çalışması sırasında en uygun kamera ayarlarının belirlenmesine olanak tanımaktadır.

```
import cv2

def bos_fonksiyon(x): pass

def kamera_kalibrasyon_arayuzu(): cap = cv2.VideoCapture(0)

pencere_adi = "Kamera Kalibrasyon Paneli"
cv2.namedWindow(pencere_adi)

cv2.createTrackbar("Parlaklik", pencere_adi, 100, 200,
bos_fonksiyon)
cv2.createTrackbar("Kontrast", pencere_adi, 40, 100, bos_fonksiyon)
cv2.createTrackbar("Doygunluk", pencere_adi, 50, 100, bos_fonksiyon)

while True:
    parlaklik = cv2.getTrackbarPos("Parlaklik", pencere_adi)
    kontrast = cv2.getTrackbarPos("Kontrast", pencere_adi)
    doygunluk = cv2.getTrackbarPos("Doygunluk", pencere_adi)

    cap.set(10, parlaklik - 100)
    cap.set(11, kontrast)
    cap.set(12, doygunluk)

    success, img = cap.read()

    if success:
        cv2.imshow(pencere_adi, img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        print(f"Son Ayarlar -> P: {parlaklik}, K: {kontrast}, D:
{doygunluk}")
        break

cap.release()
cv2.destroyAllWindows()


if name == 'main': kamera_kalibrasyon_arayuzu()
```

EK M. Python Statik Görsel Üzerinde Tahmin Yazılımı

Bu ekte, eğitilmiş yapay zekâ modelinin tek bir statik görüntü üzerinde sınıflandırma yapabilme yeteneğini test etmek amacıyla geliştirilen Python tabanlı yazılım sunulmuştur. Yazılım, harici bir görüntü dosyasını okuyarak model üzerinde çıkarım işlemi gerçekleştirmekte ve tahmin edilen sınıf ile güven skorunu kullanıcıya sunmaktadır. Bu test, sistemin gerçek zamanlı çalışmaya ek olarak çevrimdışı analizlerde de kullanılabilirliğini göstermektedir.

Aşağıda verilen Python yazılımı, diskte bulunan tek bir görüntü dosyasını yükleyerek YOLOv8 tabanlı sınıflandırma modeline giriş olarak vermektedir. Model tarafından tahmin edilen sınıf bilgisi ve güven skoru konsol çıktısı olarak sunulmakta, aynı zamanda sonuç görüntü üzerinde işaretlenerek kullanıcıya gösterilmektedir. Bu yazılım, model doğruluğunun görsel olarak değerlendirilmesi amacıyla kullanılmıştır.

```
from ultralytics import YOLO import cv2 import os

def statik_resim_testi(): model = YOLO('best.pt')

dosya_yolu = 'test_yaprak.jpg'

if not os.path.exists(dosya_yolu):
    print(f"HATA: '{dosya_yolu}' dosyasi bulunamadi!")
    return

img = cv2.imread(dosya_yolu)

results = model(img)

for r in results:
    annotated_frame = r.plot()

    probs = r.probs
    if probs:
        en_yuksek_sinif = r.names[probs.top1]
```

```
        guven = float(probs.top1conf)
        print(f"Tespit Edilen Sinif: {en_yuksek_sinif}")
        print(f"Guven Skoru: %{guven*100:.2f}")

    cv2.imshow(f"Statik Analiz - {dosya_yolu}",
annotated_frame)

print("Cikis icin bir tusa basiniz...")
cv2.waitKey(0)
cv2.destroyAllWindows()


if name == 'main': statik_resim_testi()
```

EK N. Python Sistem Başlatıcı ve Süreç Yöneticisi Yazılımı

Bu ekte, otonom tarım robotu sisteminin güvenli ve kontrollü bir şekilde başlatılmasını sağlamak amacıyla geliştirilen Python tabanlı sistem başlatıcı ve süreç yönetim yazılımı sunulmuştur. Yazılım, ana sistem çalıştırılmadan önce kritik donanım bileşenlerinin (kamera ve Arduino alt denetleyici) varlığını kontrol etmekte ve sistemin uygun koşullar altında başlatılmasını sağlamaktadır. Bu yaklaşım, sistem kararlılığını artırmak ve olası donanım kaynaklı hataların önüne geçmek amacıyla uygulanmıştır.

Aşağıda verilen Python yazılımı, sistem açılışında donanım hazırlık kontrolü gerçekleştirmekte ve gerekli koşullar sağlandığında ana kontrol yazılımını otomatik olarak çalıştırmaktadır. Kamera aygıtının algılanması ve Arduino bağlantısının tespiti sonrasında sistem, ana entegrasyon yazılımını ayrı bir süreç olarak başlatmaktadır. Hata veya kullanıcı müdahalesi durumlarında sistem kontrollü bir şekilde sonlandırılmaktadır. Bu yazılım, otonom sistemin başlatma ve yönetim aşamasında kullanılmıştır.

```
import os import sys import time import subprocess import
serial.tools.list_ports

def donanim_hazirlik_kontrolu(): print("--- OTONOM TARIM
ROBOTU BASLATILIYOR ---\n")

time.sleep(3)

kamera_yolu = '/dev/video0'
if not os.path.exists(kamera_yolu):
    print("[HATA] Kritik: Kamera donanimi bulunamadi!")
    return False
print("[OK] Kamera donanimi algilandi.")

arduino_portu = None
```

```
portlar = serial.tools.list_ports.comports()
for p in portlar:
    if "Arduino" in p.description or "ACM" in p.device:
        arduino_portu = p.device
        break

if arduino_portu:
    print(f"[OK] Alt denetleyici bulundu: {arduino_portu}")
else:
    print("[UYARI] Arduino bulunamadi. Robot sadece izleme
modunda calisacak.")

return True


def ana_yazilimi_tetikle(): dosya_adi = "final_sistem.py"
yorumlayici = sys.executable

print(f"\n[BILGI] {dosya_adi} calistiriliyor...")
print("-" * 40)

try:
    if os.path.exists(dosya_adi):
        subprocess.run([yorumlayici, dosya_adi], check=True)
    else:
        print(f"[HATA] {dosya_adi} dosyasi diskte
bulunamadi!")

except subprocess.CalledProcessError as e:
    print(f"\n[KRITIK] Sistem beklenmedik sekilde kapandi.
Hata Kodu: {e.returncode}")
except KeyboardInterrupt:
    print("\n[BILGI] Kullanici tarafindan manuel durdurma.")
finally:
    print("-" * 40)
    print("[SISTEM] Oturum sonlandirildi.")


if name == 'main': sistem_hazir = donanim_hazirlik_kontrolu()

if                                                  sistem_hazir:
    ana_yazilimi_tetikle()
else:
    print("[IPTAL] Donanim hatalari nedeniyle baslatma iptal
edildi.")
```

Bu ekte, otonom tarım robotu sisteminde kullanılan OpenCV kütüphanesinin doğru şekilde kurulduğunu, grafik çizim fonksiyonlarının çalıştığını ve görsel arayüz (GUI) yeteneklerinin kullanılabilir olduğunu doğrulamak amacıyla geliştirilen Python tabanlı test yazılımı sunulmuştur. Yazılım, OpenCV sürüm bilgisini raporlamakta, işlemci optimizasyon durumunu kontrol etmekte ve temel grafik çizim fonksiyonlarını test etmektedir. Bu doğrulama, görüntü işleme tabanlı sistemlerin kararlı çalışması açısından önemli bir ön test niteliği taşımaktadır.

Bu ekte, otonom tarım robotu sisteminde kullanılan OpenCV kütüphanesinin doğru şekilde kurulduğunu, grafik çizim fonksiyonlarının çalıştığını ve görsel arayüz (GUI) yeteneklerinin kullanılabilir olduğunu doğrulamak amacıyla geliştirilen Python tabanlı test yazılımı sunulmuştur. Yazılım, OpenCV sürüm bilgisini raporlamakta, işlemci optimizasyon durumunu kontrol etmekte ve temel grafik çizim fonksiyonlarını test etmektedir. Bu doğrulama, görüntü işleme tabanlı sistemlerin kararlı çalışması açısından önemli bir ön test niteliği taşımaktadır.

```
import cv2 import numpy as np import sys

def opencv_kutuphane_testi(): print("--- OPENCV KUTUPHANE VE
SURUM TESTI ---\n")

print(f"Python Surumu: {sys.version.split()[0]}")
print(f"OpenCV Surumu: {cv2.__version__}")

optimizasyon = cv2.useOptimized()
print(f"Islemci Optimizasyonu (SSE/AVX/NEON): {'AKTIF' if
optimizasyon else 'PASIF'}")

img = np.zeros((512, 512, 3), np.uint8)

cv2.line(img, (0, 0), (512, 512), (255, 0, 0), 5)
cv2.rectangle(img, (384, 0), (510, 128), (0, 255, 0), 3)
cv2.circle(img, (447, 63), 63, (0, 0, 255), -1)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img, 'OpenCV Test', (10, 500), font, 2, (255, 255,
255), 2, cv2.LINE_AA)

print("\n[OK] Grafik fonksiyonlari basariyla calistirildi.")
print("[BILGI] Test penceresi aciliyor. Cikis icin bir tusa
basin...")

try:
    cv2.imshow("Kutuphane Dogrulama - Geometrik Test", img)
    k = cv2.waitKey(0)
    cv2.destroyAllWindows()
    print("[BASARILI] Grafik arayuz (GUI) sorunsuz
```

```
calisiyor.")
except Exception as e:
    print(f"[HATA] Grafik arayuz başlatılamadı: {e}")
    print("Not: Headless (Monitörsüz) modda iseniz bu hata
normaldir.")


if name == 'main': opencv_kutuphane_testi()
```

ÖZGEÇMİŞ

Adı Soyadı : Ulaş Can Çakmak

Doğum Yeri ve Yılı : Adana, 2002

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : ulas1173@hotmail.com

Eğitim Durumu

Lise : Açık Öğretim Lisesi, Mezun

Lisans: Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü, 2021 – Devam Ediyor.

ÖZGEÇMİŞ

Adı Soyadı : Yavuz Sarı

Doğum Yeri ve Yılı : İzmir, 2004

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : yavuzsari2004@gmail.com

Eğitim Durumu

Lise : Adnan Menderes Anadolu Lisesi, Mezun

Lisans: Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü, 2021 – Devam Ediyor.

ÖZGEÇMİŞ

Adı Soyadı : Halil İbrahim Kurnaz

Doğum Yeri ve Yılı : Malatya, 2001

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta :halil_kurnaz11551143@outlook.com

Eğitim Durumu

Lise : Mimar Sinan Anadolu Lisesi

Önlisans: Tekirdağ Namık Kemal Üniversitesi Kontrol ve Otomasyon

Lisans: Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü, 2021 – Devam Ediyor.