



T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

OTONOM RC ARABA

Onur Ali KORKMAZ

BİTİRME ÇALIŞMASI

DANIŞMANI : Dr. Öğr. Üyesi Hakan ÜÇGÜN

BİLECİK
26 Haziran 2024



T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

OTONOM RC ARABA

Onur Ali KORKMAZ

BİTİRME ÇALIŞMASI

DANIŞMANI : Dr. Öğr. Üyesi Hakan ÜÇGÜN

BİLECİK
26 Haziran 2024

BİLDİRİM

Bu çalışmada bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

İmza

AD-SOYAD

Tarih: 26 Haziran 2024

ÖZET

BİTİRME ÇALIŞMASI

OTONOM RC ARABA

Onur Ali KORKMAZ

**Bilecik Şeyh Edebali Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü**

Danışman: Dr. Öğr. Üyesi Hakan ÜÇGÜN

2024, 56 Sayfa

Jüri Üyeleri

.....
.....
.....

İmza

Bu çalışma, görüntü işleme ve ultrasonik sensörlerin entegrasyonu ile şerit takip ve engel algılama yeteneklerine sahip bir otonom RC araba geliştirmeyi amaçlamaktadır. Sistem, Raspberry Pi ve Arduino Nano platformları kullanılarak oluşturulmuştur. Kameradan alınan görüntüler, Raspberry Pi tarafından işlenerek şerit çizgileri tespit edilmekte ve araç yönlendirme için gerekli açılar hesaplanmaktadır. Ayrıca, TensorFlow Lite ile eğitilmiş yapay zeka modeli sayesinde trafik işaretleri gibi nesneler algılanmakta ve bu bilgiler Arduino Nano'ya iletilerek aracın hareketi kontrol edilmektedir. Sonuçlar, sistemin belirli şartlar altında stabil çalıştığını ve otonom sürüş için potansiyel taşıdığını göstermektedir.

ABSTRACT

THESIS

AUTONOMOUS RC CAR

Onur Ali KORKMAZ

**Bilecik Şeyh Edebali University
Engineering Faculty
Department of Computer Engineering**

Advisor : Asst. Prof. Hakan ÜÇGÜN

2024, 56 Pages

Jury

.....
.....
.....

Sign

This study aims to develop an autonomous RC car with lane tracking and obstacle detection capabilities through the integration of image processing and ultrasonic sensors. The system is built using Raspberry Pi and Arduino Nano platforms. Images captured by the camera are processed by Raspberry Pi to detect lane lines and calculate the necessary angles for vehicle steering. Additionally, a TensorFlow Lite-trained artificial intelligence model is used to detect objects such as traffic signs, and this information is sent to Arduino Nano to control the vehicle's movements. The results demonstrate that the system operates stably under certain conditions and holds potential for autonomous driving applications.

ÖNSÖZ

Bitirme çalışmam boyunca bana rehberlik eden ve yönlendiren saygıdeğer hocam ve danışmanım Sayın Dr. Öğr. Üyesi Hakan ÜÇGÜN'e sonsuz teşekkürlerimi sunarım. Ayrıca, çalışma arkadaşlarından Ahmet Faruk GÜMÜŞTAŞ, Muhammet Eren BÜYÜK ve Yusuf GÜZELDERE'ye verdikleri destek için minnettarım. Projedeki şasiye ek olarak çizdiği tüm parçalarla büyük katkı sağlayan Hasan Mert KOŞ'a da şükranlarımı sunarım. Tüm bu süreçte gösterdikleri katkılar ve kesintisiz destekleri için hepsine teşekkür ederim.

Onur Ali KORKMAZ

26 Haziran 2024

İÇİNDEKİLER

ÖNSÖZ

v

ŞEKİLLER TABLOSU

ix

1 GİRİŞ	1
1.1 Projenin İçeriği	1
1.2 Öğrenciler İçin Projenin Önemi	1
1.3 Benzer Projeler	2
1.3.1 Projenin github linki	3
2 OTONOM RC ARABANIN TASARIMI VE MATERİYALLERİ	4
2.1 Raspberry Pi 4 B+ 8GB	5
2.2 Arduino Nano	6
2.3 Raspberry Pi Kamera Modül 3	7
2.4 BTS7960B 40 Amper Motor Sürücü Kartı	7
2.4.1 11.1 V 3S Lipo Batarya-Pil 2800 mAh 35C	8
2.5 Ultrasonik Mesafe Sensörü HC-SR04	10
2.6 DC Motor	10
2.7 Servo Motor	11
2.8 RC Araba Şase Yapısı	12
2.9 Devre Şeması	13
3 GÖRÜNTÜ İŞLEME İLE RC ARAÇ KONTROLÜ	15
3.1 HSV(Hue-Ton, Saturation-Doygunluk, Value-Değer)	16
3.2 Canny Algoritması	17
3.3 İlgi Alanı Bölgesi (Region of Interest)	17
3.4 Hough Dönüşümü	19
3.5 Eğim ve Kesişim ile Sağ ve Sol Çizgileri Bulma	21
3.6 Direksiyon Açısunın Hesaplanması	22
4 YAPAY ZEKA MODELİ İLE NESNE ALGILAMA	25
4.1 TensorFlow	25

4.1.1	Tensörler	25
4.1.2	Graf Yapısı	26
4.1.3	TensorFlow Nasıl Çalışır?	26
4.2	TensorFlow Lite Nedir?	27
4.3	Eğitimmiş Modelin Raspberry Pi ile Uyumu	29
5	OTONOM RC ARACIN GERÇEKLEŞTİRİMİ	30
5.1	Arduino Nano Entegrasyonu	37
5.2	Seri Haberleşme Protokollerı	37
5.2.1	USB haberleşme protokolü	38
5.3	Arduino Nano DC ve Servo Motor Kontrolü	38
6	SONUÇLAR VE ÖNERİLER	42
KAYNAKLAR		44
ÖZGEÇMİŞ		46

ŞEKİLLER TABLOSU

Şekil 2.1 Devrenin Blok Diagramı	4
Şekil 2.2 Raspberry Pi 4 B+	5
Şekil 2.3 Arduino Nano	6
Şekil 2.4 Raspberry Pi Kamera Modül 3	7
Şekil 2.5 BTS7960B 40 Amper Motor Sürücü Kartı	8
Şekil 2.6 11.1 V 3S Lipo Batarya-Pil 2800 mAh 35C	9
Şekil 2.7 Ultrasonik Mesafe Sensörü HC-SR04	10
Şekil 2.8 540 12V 10000Rpm Redüktörsüz DC Motor	11
Şekil 2.9 Servo Motor İç Yapısı ve Kullanılan Servo Motor	11
Şekil 2.10 RC Araba Şase ve Parçaları	12
Şekil 2.11 RC Araba SolidWorks Çizimi	13
Şekil 2.12 Otonom RC Araba Devre Şeması	14
Şekil 3.1 Görüntü İşleme Blok Diyagramı	15
Şekil 3.2 Maskelenmiş Görüntünün Yapısı	16
Şekil 3.3 Canny Algoritması ile Çizgilerin Bulunması	17
Şekil 3.4 ROI ile alanın belirlenmesi	18
Şekil 3.5 Şekil 3.5: Hough Dönüşümü ile Çizgi Tespiti: (a) Orijinal noktalar, (b) Noktalardan geçen olası çizgiler, (c) Parametre uzayında noktaların kesişimi.	19
Şekil 3.6 Çizgi Tespiti	20
Şekil 3.7 HoughLine Transformu ve Çizgilerin Görüntülenmesi	21
Şekil 3.8 Sağ ve Sol Çizgilerin Belirlenmesi	22
Şekil 3.9 Yön Çizgisi	23
Şekil 3.10 Yön Çizgisine Göre Açı Üretmek	23
Şekil 4.1 Üç Eksenli Tensör	25
Şekil 4.2 Graf Yapısı	26
Şekil 4.3 TensorFlow Çalışma Prensibi	27
Şekil 4.4 Dönüşürme İşleminin Diagramı	28
Şekil 4.5 TensorFlow Lite İşleme Adımları	28

Şekil 4.6 TFLite Modelinin Raspberry Pi Üzerinde Testi	29
Şekil 5.1 RC Araba Montaj	30
Şekil 5.2 Montaj Bitmesi	30
Şekil 5.3 Fircasız Motor	31
Şekil 5.4 Araç Problemleri	32
Şekil 5.5 RC Araç V1	33
Şekil 5.6 Kamera ve Montajını Geliştirme	34
Şekil 5.7 Kamera v3 ve Görüntüdeki Kaymalar	35
Şekil 5.8 RC Araç Ön ve Arka Yüzü	36
Şekil 5.9 TFLite Model ve Görüntü İşleme	36
Şekil 5.10 Arduino Nano Akış Diyagramı	37
Şekil 5.11 USB Bağlantı Uçları	38
Şekil 5.12 Arduino Tanım Kodları	39
Şekil 5.13 Arduino Setup Kodları	39
Şekil 5.14 Motor Kontrol Fonksiyonu	40
Şekil 5.15 Mesafe Ölçüm Fonksiyonu	40
Şekil 5.16 Gelen Veriyi Değişkende Saklayan Fonksiyon	40
Şekil 5.17 Servo Motor Kontrol Fonksiyonu	41
Şekil 5.18 Loop Fonksiyonu	41
Şekil 6.1 Düzeltmesi gereken hatalar	43

1 GİRİŞ

Otonom araç teknolojileri, günümüzün en heyecan verici ve hızla gelişen alanlarından biridir. Bu çalışmada, görüntü işleme ve ultrasonik sensörlerin entegrasyonu ile şerit takip ve engel algılama yeteneklerine sahip bir otonom RC araba geliştirilmiştir.

1.1 Projenin İçeriği

Bu projede, görüntü işleme ve ultrasonik sensör teknolojilerini kullanarak şerit takip ve engel algılama yeteneklerine sahip bir otonom RC araba geliştirilmiştir. Görüntü işleme teknikleri ile yol üzerindeki şeritler tespit edilmekte ve matematiksel formüller kullanılarak şeridin orta konumu hesaplanmaktadır. Bu bilgi, aracın motorlarına tekerlek açısı olarak iletilmekte ve araç şerit içinde tutulmaktadır. Aynı zamanda, aracın önüne yerleştirilen ultrasonik sensörler, 30 cm ve altındaki mesafelerde engelleri algılayarak aracı durdurmaktadır. Bu sistemlerin entegrasyonu sayesinde, geliştirilen otonom araç hem şerit takibi yapabilmekte hem de önünde engel geldiğinde durarak güvenliğini sağlamaktadır. Proje kapsamında, sistemin performansı test edilmiş ve sonuçlar değerlendirilerek olası iyileştirme önerileri sunulmuştur.

1.2 Öğrenciler İçin Projenin Önemi

Bu proje, öğrenciler için çok yönlü bir öğrenme deneyimi sunar. Görüntü işleme ve sensör verilerinin işlenmesi gibi temel otonom araç teknolojilerini uygulamalı olarak öğrenme fırsatı sağlar. Ayrıca, öğrencilere teorik bilgilerini pratikte test etme imkanı sunmaktadır.

1.3 Benzer Projeler

Aşağıda otonom RC arabaya ait benzer projeler gösterilmiştir.

- **Real-Time Self-Driving Car Navigation Using Deep Neural Network:** Bu çalışmada, Raspberry Pi üzerinde derin sinir ağları kullanarak tek kameralı bir otonom araç prototipi geliştirilmiştir. Proje, derin öğrenme modellerinin araç kontrolünde kullanılması üzerine odaklanmıştır. Eğitim verisi olarak, RC araç üzerinde bir ön kamera ve manuel sürüş ile toplanan yol görüntüleri ve zaman senkronize edilmiş direksiyon açıları kullanılmıştır. Eğitim işlemi bir masaüstü bilgisayarda gerçekleştirilmiş ve model daha sonra Raspberry Pi üzerinde çalıştırılmıştır. Test sonuçları, modelin şerit takip görevinde etkili ve sağlam olduğunu göstermiştir. Araç, 5-6 km/h hızla çeşitli sürüş koşullarında, şerit işaretleri olup olmamasına bakılmaksızın başarılı bir şekilde kendini yönlendirebilmiştir. Bu araştırma, düşük maliyetli gömülü sistemler kullanarak gerçek zamanlı otonom araç kontrolü için umut verici sonuçlar sunmaktadır [1].
- **BUILDING A SELF-DRIVING RC CAR:** Bu tezde, otonom bir RC araba modelinin sıfırdan nasıl inşa edileceği detaylı bir şekilde ele alınmaktadır. Tezde, otonom araç modelinin önemli bileşenleri, gerekli donanım platformunun inşası ve entegrasyonu ile donanım platformundan alınan veriyi kullanarak aracı kontrol eden uçtan uca makine öğrenme boru hattı kapsamlı bir şekilde açıklanmaktadır. RC arabanın elektronik ve mekanik montajı, Nvidia Jetson Nano gibi gömülü sistemler kullanılarak nasıl gerçekleştirileceği anlatılmaktadır. Ayrıca, DonkeyCar platformu kullanılarak yüksek seviye sinir ağları modellerinin oluşturulması ve eğitilmesi, kamera kalibrasyonu, şerit çizgilerinin çıkarılması ve otonom sürüş için gerekli yapay zeka teknikleri detaylandırılmaktadır. Tez, otonom araç teknolojilerinin gelecekteki potansiyel etkileri ve bu teknolojinin araştırma ve öğrencilere sunulması gerektiğini vurgulayarak, derin öğrenme ve yapay zeka yöntemlerinin hızla ilerlemesi sayesinde bu tür projelerin pahalı laboratuvarlar ve uzun yıllar süren araştırmalara ihtiyaç duymadan gerçekleştirilebileceğini göstermektedir [2].

- **A Self-Driving Car Implementation using Computer Vision for Detection and Navigation:** Bu projede, lidar gibi pahalı sensörler yerine daha uygun maliyetli bir alternatif olan kameralar kullanılarak otonom bir araç prototipi geliştirilmiştir. Kamera tabanlı bu prototip, sanal bir şehir ortamında güvenli, hızlı, verimli ve konforlu bir şekilde gezinmeyi amaçlamaktadır. Araç, şeritleri, trafik araçlarını, engelleri ve trafik sinyallerini tespit etmek için YOLOv3 algoritmasını kullanır ve derinlik hesaplamaları için stereo görüş konseptinden faydalananır. Sistemin yolu algılama, koordinat dönüşümü, yol planlaması ve kontrol gibi modülleri, her bir kamera karesi için ayrıntılı bir işlem hattı ile çalışır. Deneysel sonuçlar, kamera tabanlı otonom araçların uygulanabilir olduğunu ve bu projenin gelecekteki gerçek dünya uygulamaları için bir temel oluşturabileceğini göstermektedir [3].

1.3.1 Projenin github linki

Proje kaynak kod olarak aşağıdaki github linkinde sunulmuştur.

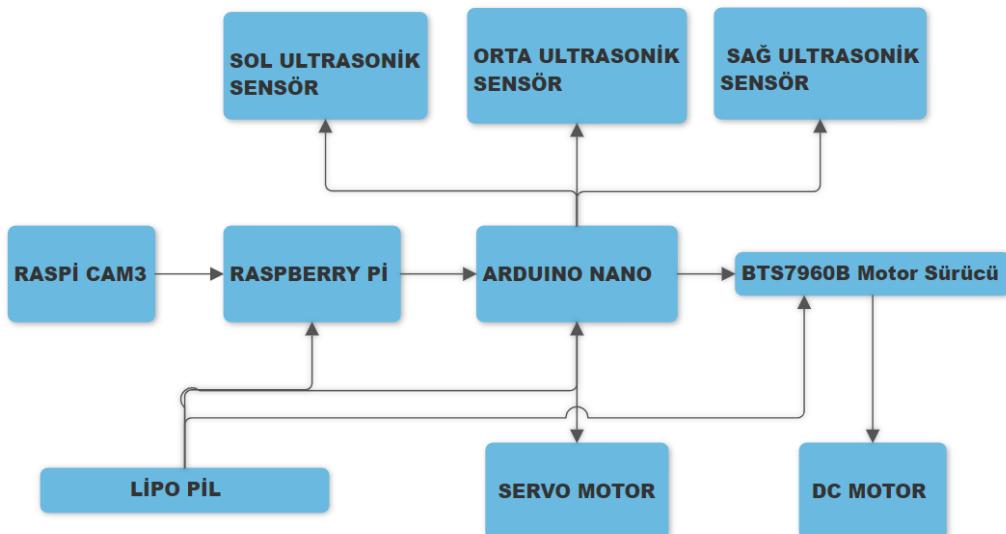
Github Linki

2 OTONOM RC ARABANIN TASARIMI VE MATER-YALLERİ

Raspberry Pi kamera modülünden gelen görüntü, Raspberry Pi'ye iletilir. Burada görüntü işleme aşamalarından geçerek şerit çizgisi tespit edilir. Tespit edilen şerit çizgisinin konumuna göre açı hesaplanır ve bu açı değeri Arduino Nano'ya iletilir. Arduino Nano, gelen açı değerine göre servo motoru maksimum 140 derece ve minimum 40 derece arasında döndürür.

Ayrıca, Arduino Nano üzerinde bulunan üç ultrasonik sensör aracılığıyla mesafe ölçümü yapılır. Eğer sensörler 30 cm ve altında bir engel algılaysa, Arduino Nano DC motoru durdurur. Engelin algılanmadığı durumlarda ise DC motor sabit bir hızda hareket eder.

Bu sistem, Raspberry Pi ve Arduino Nano'nun birlikte çalışarak otonom bir RC araba kontrolünü nasıl gerçekleştirdiğini gösterir. Şekil 2.1'deki diyagramda, kamera modülü, raspberry pi, arduino nano, servo motor, DC motor ve ultrasonik sensörlerin birbirleriyle olan etkileşimleri detaylandırılmıştır.



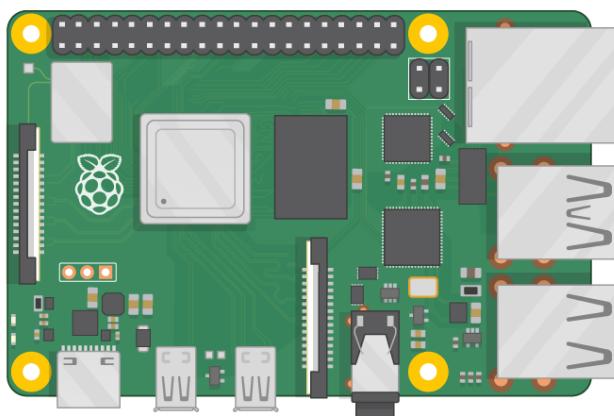
Şekil 2.1: Devrenin Blok Diagramı

2.1 Raspberry Pi 4 B+ 8GB

Raspberry Pi, tek bir karttan oluşan mini bir bilgisayardır. Öğrenci, amatör ve hobi olarak programlamaya veya bilgisayar bilimlerine ilgi duyan kişiler için geliştirilmiştir [4].

Raspberry Pi, Şekil 2.2'de gösterilen temel bilgisayar bileşenlerine sahip kompakt bir cihazdır. İşlemci, RAM, USB portları, HDMI çıkışı, Ethernet girişi ve microSD kart yuvası gibi bileşenleri içerir. Diğer bilgisayarlardan en büyük farkı, küçük ve taşınabilir yapısıdır. Bu özellikleri sayesinde Raspberry Pi, çeşitli projelerde kolay ve esnek bir şekilde kullanılabilir. Genellikle Raspberry Pi OS adı verilen ve Linux tabanlı olan kendi işletim sistemini kullanır. Python, C/C++ gibi farklı programlama dilleri ile yazılım geliştirme imkanı sunar [5].

Raspberry Pi, bu projede kameradan gelen görüntüleri işleyerek şerit algılama ve görüntü sınıflandırma işlemlerini gerçekleştirir. Şerit algılama işlemi sırasında yoldaki çizgileri tespit eder ve bu çizgilerin x eksenine göre açlarını hesaplar. Ayrıca, TensorFlow Lite ile eğitilen model sayesinde gelen görüntüleri sınıflandırır ve algıladığı nesnelerin bilgilerini belirler. Bu veriler, seri haberleşme yoluyla Arduino Nano'ya iletilir. Böylece, Raspberry Pi hem şerit algılama hem de nesne tanıma işlemlerini yaparak Arduino Nano'ya açı ve nesne bilgilerini gönderir, sistemin doğru ve güvenli bir şekilde çalışmasını sağlar.



Şekil 2.2: Raspberry Pi 4 B+

2.2 Arduino Nano

Ardunio; Bir giriş çıkış kartı ve işleme/yazma dilinin bir uygulamasını içeren geliştirme ortamında programlanabilen bir karttır. Ardunio, açık kaynak kodlu yazılım ve donanıma sahip bir mikrodenetleyici platformudur. Ardunio geliştirme kartı üzerindeki mikroişlemci arduino programlama dili ile programlanır ve bu program arduino geliştirme ortamı (IDE) yardımı ile karta yüklenir. Şekil 2.3'ün a ve b yapısında Ardunio Nano'nun ön ve arka yüzü bulunmaktadır [6].

Arduno Nano, bu projede kritik bir rol oynamaktadır. Raspberry Pi'den gelen verileri işleyerek servo motorun doğru açıda dönmesini ve DC motorun hızını kontrol ederken, ultrasonik sensörlerden gelen verileri değerlendirerek engellere çarpmamayı sağlar. Ayrıca, Raspberry Pi'nin gönderdiği nesne algılama bilgilerini işleyerek, trafik işaretlerine ve diğer nesnelere uygun tepkiler verir. Bu çok yönlü görevleri sayesinde, sistemin güvenli ve etkin bir şekilde çalışmasını mümkün kılar. Arduino Nano, proje içerisinde merkezi bir kontrol birimi olarak, tüm bileşenlerin uyum içinde çalışmasını sağlar ve otonom sistemin başarısını destekler.



(a) Arduino Nano Ön Yüz

(b) Arduino Nano Arka Yüz

Şekil 2.3: Arduino Nano

2.3 Raspberry Pi Kamera Modül 3

Raspberry Pi Kamera Modülü 3, yeni bir 12 Megapiksel Sony IMX708 görüntü sensörü, güçlü otomatik odaklılama, gelişmiş düşük ışık hassasiyeti ve HDR (Yüksek Dinamik Aralıktır) desteği sunan Resmi Raspberry Pi kamera modülleri serisinin en son üyesidir. Bu özellikler sayesinde, düşük ışık koşullarında bile yüksek kaliteli görüntüler yakalayabilir ve geniş bir dinamik aralık sunarak daha canlı ve ayrıntılı fotoğraflar çekebilir. Otomatik odaklılama özelliği, çeşitli mesafelerde net ve keskin görüntüler elde etmeyi kolaylaştırır. Kamera Modülü 3, Raspberry Pi projelerinde video kaydı, fotoğraf çekimi ve görüntü işleme uygulamaları için mükemmel bir çözümüdür [7]. Şekil 2.4'de Raspberry Pi Kamera Modülü 3 gösterilmiştir.



Şekil 2.4: Raspberry Pi Kamera Modül 3

2.4 BTS7960B 40 Amper Motor Sürücü Kartı

BTS7960B 40 Amper Motor Sürücü Kartı, sumo ve arazi robotlarında kullanılan, yüksek güçlü ve yüksek akım kapasitesine sahip bir sürücü kartıdır. 28V'a kadar olan besleme gerilimlerinde çalışabilen bu kart, 40 ampere kadar akım çeken motorları kontrol edebilir. Arduino ve diğer mikrodenetleyici platformları ile uyumlu olması, BTS7960B'nin kullanımını oldukça kolay ve pratik hale getirir. Devre kartı üzerindeki pasif soğutucu, yeterli soğutmayı sağladığı için ek bir soğutucuya ihtiyaç duyulmaz [8]. Şekil 2.5'te BTS7960B 40 Amper Motor Sürücü Kartı gösterilmiştir.

BTS7960B Motor ve Besleme Bağlantıları:

- **B+** : Besleme Gerilimi (28 V'a kadar)
- **B-** : Besleme Toprağı
- **M+** : Motor + Ucu
- **M-** : Motor - Ucu

BTS7960B Kontrol Pinleri:

- **VCC:** +5 VDC
- **GND:** Toprak Bağlantısı
- **IS1, IS2:** Akım Test Pinleri
- **EN1, EN2:** PWM Girişleri
- **IN1, IN2:** Motor Yön Kontrol Pinleri



Şekil 2.5: BTS7960B 40 Amper Motor Sürücü Kartı

2.4.1 11.1 V 3S Lipo Batarya-Pil 2800 mAh 35C

LiPo piller (Lityum Polimer), polimer elektrolit kullanan ve tekrar şarj edilebilen, lityum iyon tabanlı bir pil teknolojisidir. LiPo piller, günümüzde pek çok tüketici elektroniki cihazında yaygın olarak kullanılan bir pil türüdür. Bu piller, sıvı elektrolit yerine polimer elektrolit kullanmasıyla diğer lityum iyon pillerden farklılık gösterir. LiPo piller, kullanılmadıkları zamanlarda enerji kaybı çok düşük olması ve yüksek güç sağlama gibi avantajlara sahiptir. Bu nedenle, enerji yoğun uygulamalarda ve taşınabilir elektronik cihazlarda tercih edilen bir güç kaynağıdır [9].

LiPo piller (Lityum Polimer), kuru elektrolit polimerleri kullanarak çalışan bir pil teknolojisidir. Bu pillerin yapısında, elektrotlar arasında doğrudan teması önlemek için ince bir plastik film üzerine kaplanmış polimer tabakalar bulunur. Bu tabakalar, birbirinin üzere yığılmış şekilde yer alır ve pilin iç yapısını oluşturur. LiPo pillerin özel bir özelliği ise, elektrot parçacıklarının değil, yalnızca iyonların bir taraftan diğerine geçmesine izin veren mikro yapıda gözenekli bir ayırcının kullanılmasıdır. Bu ayırcı, iyon iletimini sağlarken elektrotlar arasında doğrudan teması engeller. Bu sayede, LiPo piller daha güvenli, daha verimli ve daha yüksek enerji yoğunluğuna sahip olurlar. Şekil 2.6'da lipo pilin kısımlarına değinilmiştir.

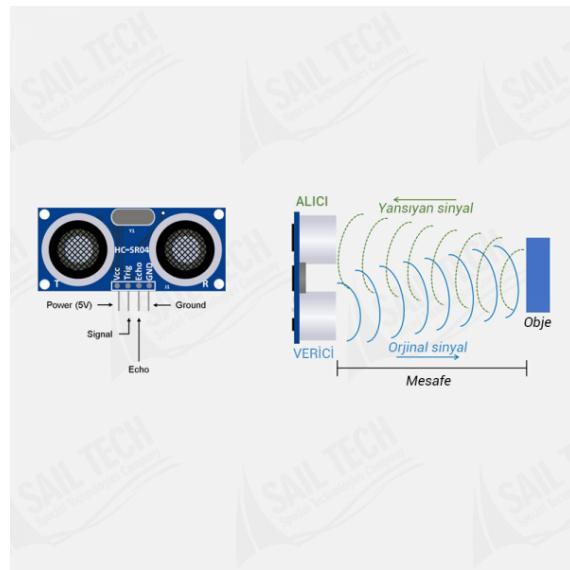
- **Gerilim/Hücre sayısı:** Lipo piller hücrelerden meydana gelir her bir hücrenin boş hali 3V dolu hali ise 4.2V olmalıdır. Hücreler seri(S) ve paralel(P) bağlı olabilirler. S değeri arttıkça pilin voltajı, P değeri arttıkça pilin kapasitesi artar.
- **Kapasitesi:** Pilin kapasitesi ne kadar güç tutabileceğini belirler. Ölçme birimi miliamper saatdir(mAh). Kapasite ne kadar büyük olursa çalışma süresi o kadar artar.
- **Deşarj oranı (C):** Pilin deşarj olma kabiliyetlerini belirleyen bir kat sayıdır. Lipo pillerin hızlı deşark olabilme kabiliyetleri deşarj katsayısı(C) x Kapasite(mAh) formülü kullanarak hesaplanır [10].



Şekil 2.6: 11.1 V 3S Lipo Batarya-Pil 2800 mAh 35C

2.5 Ultrasonik Mesafe Sensörü HC-SR04

HC-SR04 Ultrasonik Sensör, sonar iletişimini kullanarak karşısındaki nesneye olan mesafesini hesaplayan bir cihazdır. Sonar sistemi, ses dalgalarını kullanarak cismin uzaklığını belirlemektedir. Sensörün basit bir çalışma prensibi vardır. **TRIG** pininden sinyal geldiğinde bir ses dalgası sensör tarafından üretilir ve bu ses dalgası bir cisme çarpıp döndüğünde **ECHO** pini aktif hale gelir. Sesin havada yayılma hızı bilindiği için **TRIG** pinine verilen sinyalden sonra **ECHO** pininin aktif olduğu zamana kadar ki geçen süreyi hesaplayarak aradaki mesafe kolayca hesaplanmaktadır [11]. Şekil 2.7’de ultrasonik mesafe sensörü gösterilmiştir.



Şekil 2.7: Ultrasonik Mesafe Sensörü HC-SR04

2.6 DC Motor

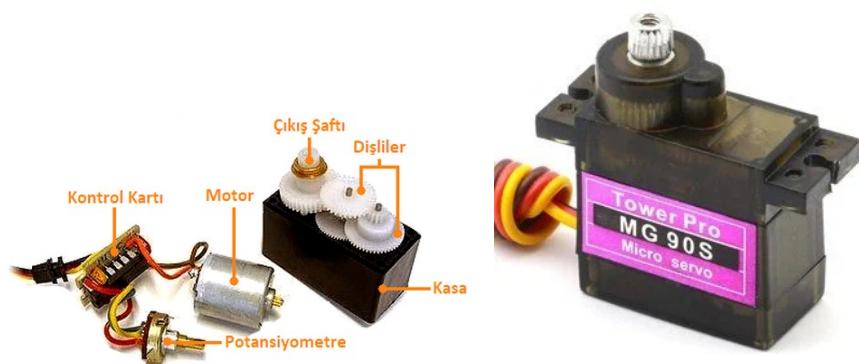
DC motor, doğru akım elektrik enerjisini mekanik enerjiye dönüştüren bir cihazdır. Motorun içindeki sargılara elektrik akımı verildiğinde, motorun içindeki sabit mıknatıslarla zıt yönde oluşan manyetik kuvvetin etkisiyle hareket eder. Bu akımın yönü, sürekli olarak sabit mıknatısa ters manyetik alan oluşturacak şekilde değiştirilmelidir. Bu değişim, fırçalı motorlarda motorun sargılarına temas eden fırçalar aracılığıyla, fırçasız motorlarda ise elektronik hız kontrol devresi ile sağlanır [12]. Şekil 2.8’de DC motor gösterilmiştir.



Şekil 2.8: 540 12V 10000Rpm Redüktörsüz DC Motor

2.7 Servo Motor

Servo motorlarda motorun hareketini sağlayan DC motor bulunmaktadır. Bu motor bir dişli mekanizmasına bağlıdır. Ayrıca iki bileşen dışında potansiyometre ve motor sürücü devresi bulunmaktadır. potansiyometre, motor milinin dönüş miktarını ölçmektedir. DC motor hareket ettikçe potansiyometre de döner ve kontrol devresi istenilen motor pozisyonu ile gerçekleştirilen motor pozisyonunu karşılaştırır. Böylece motor sürüsü gerçekleşir. Servo motorlarının genellikle çalışma aralıkları 0-180 derecedir. ve gerilimleri ise 4.8V-6V aralığındadır [13]. Şekil 2.9'un a ve b yapısında servo motor gösterilmiştir.



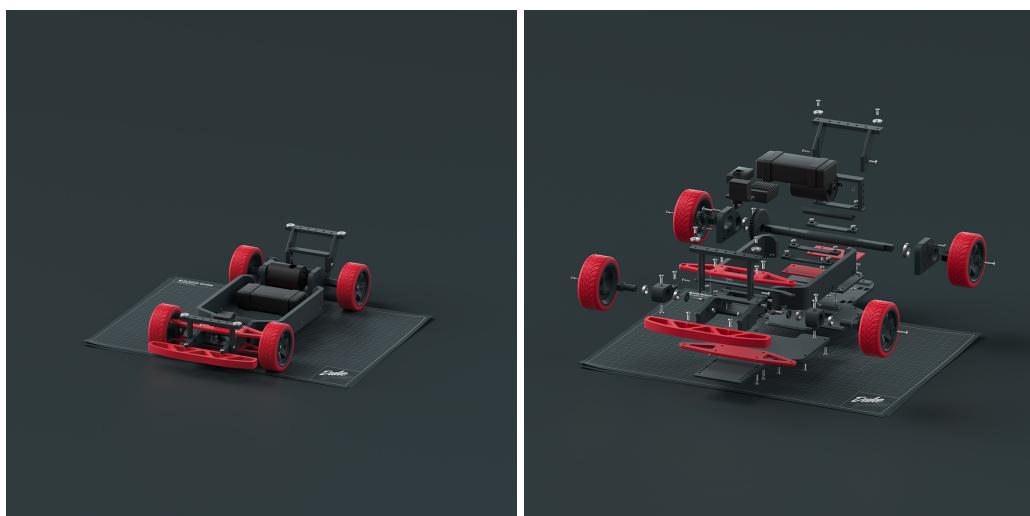
(a) Servo Motor İç Yapısı

(b) Kullanılan Servo Motor

Şekil 2.9: Servo Motor İç Yapısı ve Kullanılan Servo Motor

2.8 RC Araba Şase Yapısı

Şekil 2.10'daki a ve b yapısında, otonom RC arabanın gövdesi ve şasi yapısı detaylı bir şekilde yer almaktadır. Şasi, aracın temel yapısını oluşturan ve diğer tüm bileşenlerin monte edildiği sağlam bir çerçevedir. Hafif ve dayanıklı malzeme olan PLA+ ile üretilmiştir. Kırmızı renkli tekerlekler ve süspansiyon sistemi, arabanın denge ve yol tutuşunu artırırken, elektronik bileşenler için ayrılmış bölmeler düzenli bir montaj ve kablolama imkanı sunmaktadır. Bu tasarım, aynı zamanda çeşitli sensör ve kontrol sistemlerinin entegrasyonunu kolaylaştıracak şekilde tasarlanmıştır. Böylece araca ek parçalarda gelişmiştir.

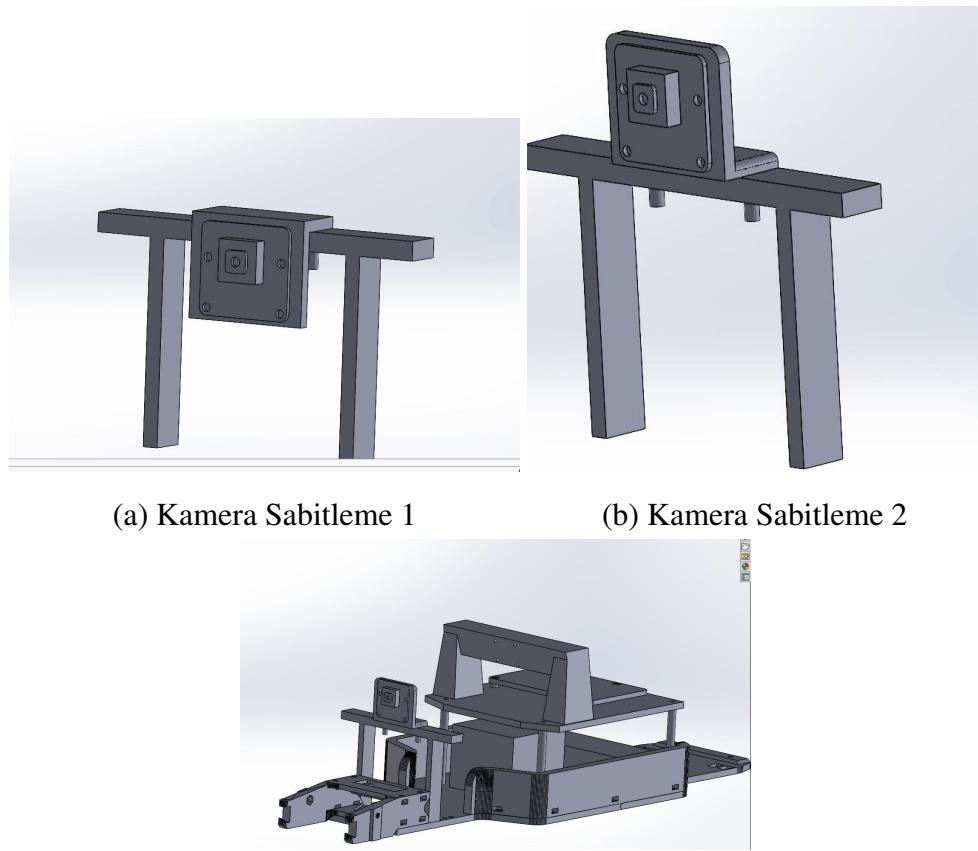


(a) RC Araba Şasesi

(b) RC Araba Şasesi Parçaları

Şekil 2.10: RC Araba Şase ve Parçaları

Şekil 2.11'deki c yapısında aracın genel çizimi gösterilmiştir. Kontrol kartının montajı için şasenin üstüne bir ek parça eklenmiştir. Şasenin en alt kısmında pil yer alırken, pilin üzerinde Raspberry Pi, motor sürücü kartı ve Şekil 2.11'deki b yapısında bulunan kamera bulunmaktadır. Ayrıca, öndeki servo motorun üstüne Arduino Nano yerleştirilmiştir. Son olarak aracın tampon bölgesinde de 3 adet Ultrasonik sensör bulunmaktadır.



Sekil 2.11: RC Araba SolidWorks Cizimi

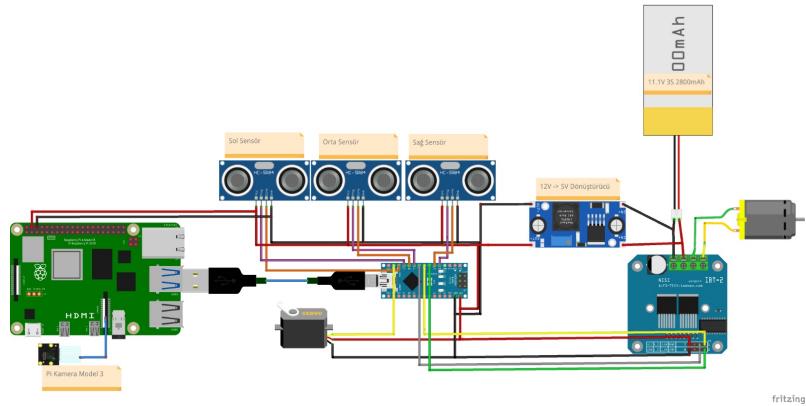
2.9 Devre Şeması

Raspberry Pi ve Arduino Nano kullanarak bir görüntü işleme ve nesne algılama sistemi geliştirilmiştir. Sistem, kameradan gelen görüntüleri işleyerek yoldaki çizgileri tespit eder ve bu çizgilerin x eksenine göre açlarını hesaplar. Hesaplanan açı değerleri, seri haberleşme yoluyla Arduino Nano'ya gönderilir.

Arduino Nano, servo motor, DC motor ve ultrasonik sensörlerden gelen verileri yönetir. Raspberry Pi'den gelen açı değerlerine göre servo motoru döndürür ve DC motoru belirli bir hızda çalıştırır. Ayrıca, ultrasonik sensör 30 cm veya daha yakın bir mesafede bir nesne algılandığında, DC motoru durdurur. Nesne ortadan kalktığında ise sistem normal çalışma düzenine geri döner.

Sistemin bir diğer önemli bileşeni ise TensorFlow Lite kullanılarak eğitilen modeldir. Bu model, kameradan gelen görüntülerini sınıflandırarak belirli durumları tespit eder. Örneğin, model dur işaretti, kırmızı ışık, sarı ışık ve yeşil ışık gibi trafik işaretlerini algılayabilir. Algılanan bu bilgiler, string veri olarak Arduino Nano'ya gönderilir. Arduino Nano, gelen veriye göre motor komutlarını düzenler; dur işaretti veya kırmızı ışık algılandığında araç durur, sarı ışık veya yeşil ışık algılandığında ise hareketine devam eder.

Bu entegrasyon sayesinde, Raspberry Pi'nin güçlü görüntü işleme yetenekleri ile Arduino Nano'nun çevresel sensör verilerini yönetme kapasitesi bir araya getirilmiş, böylece otonom araç uygulamalarında kullanılabilcek etkili bir sistem oluşturulmuştur. Sistem, yoldaki çizgilerin algılanmasından trafik işaretlerinin tanınmasına kadar geniş bir yelpazede işlevsellik sunarak, gerçek dünya uygulamaları için güçlü bir temel sağlar. Şekil 2.12'de devre şeması gösterilmiştir.



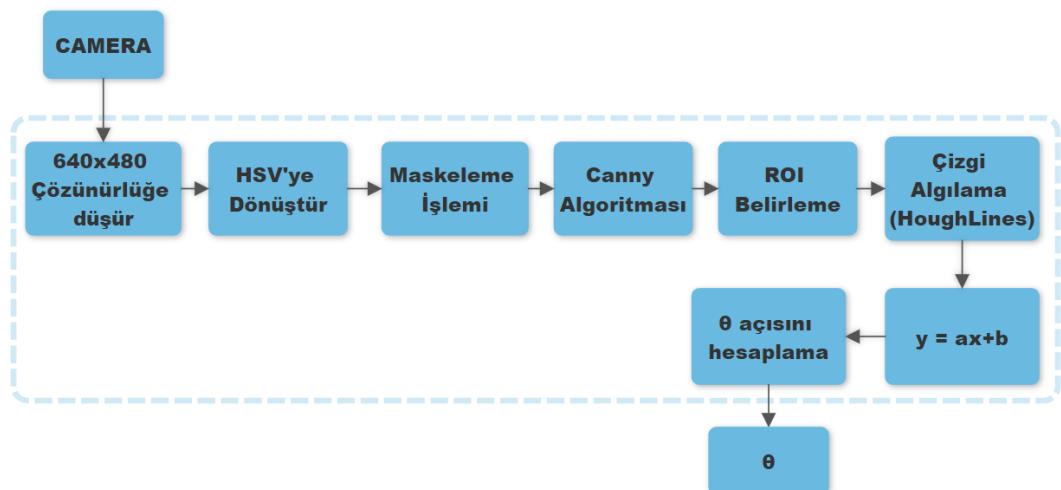
Şekil 2.12: Otonom RC Araba Devre Şeması

3 GÖRÜNTÜ İŞLEME İLE RC ARAÇ KONTROLÜ

Kameradan gelen 1920x1080 çözünürlüğündeki görüntü önce 640x480 çözünürlüğüne düşürülür. Bu işlem, görüntü işleme sürecini hızlandırmak ve performansı optimize etmek için yapılır. Görüntü daha sonra HSV (Hue, Saturation, Value) renk uzayına dönüştürülür. HSV renk uzayı, renklerin doygunluk ve parlaklık oranlarına göre işlenmesini sağlar. Bu aşamada, belirli doygunluk ve parlaklık oranlarının üzerindeki renkler beyaz, geri kalanlar ise siyah olarak maskelenir. Bu işlem sonucunda görüntü, sadece siyah ve beyaz renklerden oluşur.

Daha sonra Canny algoritması kullanılarak kenar algılama işlemi gerçekleştirilir. İlk olarak, görüntü Gauss filtresinden geçirilerek beyaz kısımlar daha belirgin hale getirilir. Belirginleşen beyaz kısımların kenarlarının bulunması daha kolaydır. Kenarlar algılandıktan sonra, ilgi alanı (ROI - Region of Interest) belirlenir. Sistem bu ilgi alanı bölgesini tanıma ve bu bölgeyi belirlemek için AND kapısı kullanılır.

HoughLines algoritması ile ilgi alanı bölgesindeki çizgiler tespit edilir. Bu algoritma, tespit edilen çizgilerden ortalama düz bir çizgi elde eder. İki çizgi arasındaki doğruluk formülü ve trigonometrik hesaplamalar kullanılarak, görüntünün kendisi ile açıları ele alınarak bir açı (θ) elde edilir. Şekil 3.1'de görüntü işleme blok diyagramı gösterilmiştir.



Şekil 3.1: Görüntü İşleme Blok Diyagramı

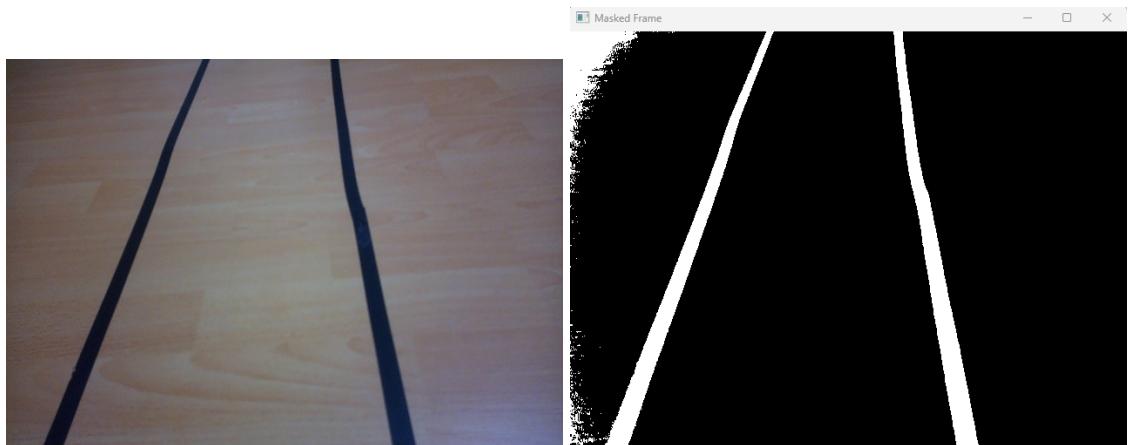
3.1 HSV(Hue-Ton, Saturation-Doygunluk, Value-Değer)

Renk uzayları, görüntüde bulunan renk kanallarını temsil etmenin bir yoludur. Bir kaç farklı renk alanı vardır. Bunlar: RGB(Kırmızı, Yeşil, Mavi), CMYK(Cyan, Magenta, Sarı, Siyah), HSV(Ton, Doygunluk, Değer) vb.'dir. OpenCV'nin varsayılan renk uzayı RGB'dir [14].

HSV renk alanı, ton ve doygunluk bilgilerini parlaklık veya değer bilgilerinden ayıran renklerin bir temsiliidir. Bu, rengi ve parlaklığını bağımsız olarak değiştirmek istediğimiz senaryolarda kullanılabilir [15].

- **Ton (H):** Bir görüntüdeki baskın rengi temsil eder. HSV renk uzayında renk tonu değeri 0 ila 360 arasında değişmektedir ve tüm renk spektrumunu kapsar [15].
- **Doygunluk (S):** Bir rengin canlılığını ifade eder. 0 doygunluk gri tonlamalı bir görüntü ile sonuçlanırken, 1 değeri en saf biçimini temsil eder [15].
- **Değer (V):** Bir rengin parlaklığını temsil eder. 0 değeri siyahı temsil ederken, 1 değeri rengin tam parlaklıktan olmasını sağlar [15].

Bir görüntü, verilen HSV renk aralığında maskeleme işlemiyle işlenir. Bu işlem, belirlenen renk aralığındaki pikselleri beyaz (255), diğer pikselleri ise siyah (0) yapar. Şekil 3.2'nin a yapısında orijinal görüntü, b yapısında maskelenmiş görüntü gösterilmiştir.



(a) Orijinal Görüntü

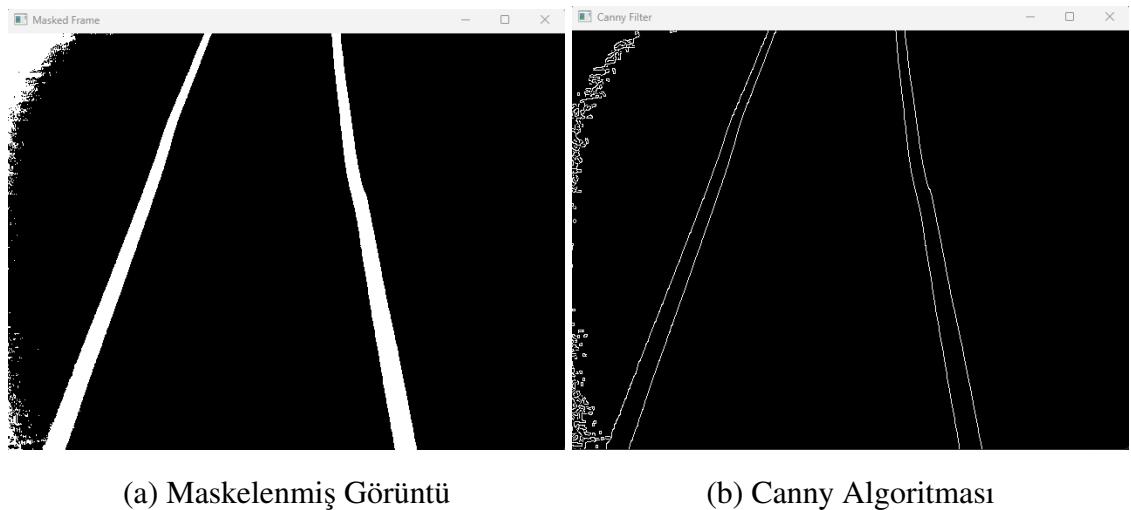
(b) Maskelenmiş Görüntü

Şekil 3.2: Maskelenmiş Görüntünün Yapısı

3.2 Canny Algoritması

Canny kenar belirleme algoritması, görüntüde keskin olarak belirlenmiş kenarları bulmak için John F. Canny tarafından geliştirilmiş ve aşamaları olan bir algoritmadır. Kenar bulmada son derece etkin olarak kullanılan bir algoritmadır. Görüntünün gürültülerini azaltması için Gaussianfiltresi kullanılır. Gaussianfiltresi, görüntüleri bulanıklaştırmak/-yumuşatmak ve görüntü üzerindeki gürültüyü arındırmak için kullanılır [16].

Bu fonksiyon, bir görüntüyü alır ve Canny kenar algılama algoritmasını uygular. İlk olarak, görüntü gri tonlamaya dönüştürülür (eğer renkli ise). Ardından, gri tonlamalı görüntü Gaussian bulanıklaştırma filtresi ile bulanıklaştırılır. Son olarak, bulanıklaştırılmış görüntü üzerinde Canny kenar algılama algoritması uygulanır ve elde edilen kenar algılama sonucu isteğe bağlı olarak ekranda gösterilir. Fonksiyon, kenar algılama sonucunu döndürür. Şekil 3.3’ün a yapısında maskelenmiş görüntü, b yapısında ise canny algoritmasından elde edilen görüntü gösterilmiştir.



(a) Maskelenmiş Görüntü

(b) Canny Algoritması

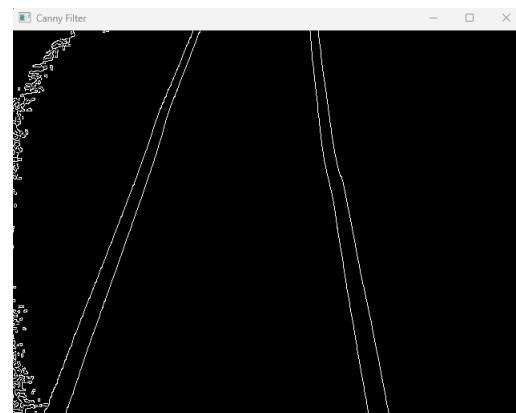
Şekil 3.3: Canny Algoritması ile Çizgilerin Bulunması

3.3 İlgi Alanı Bölgesi (Region of Interest)

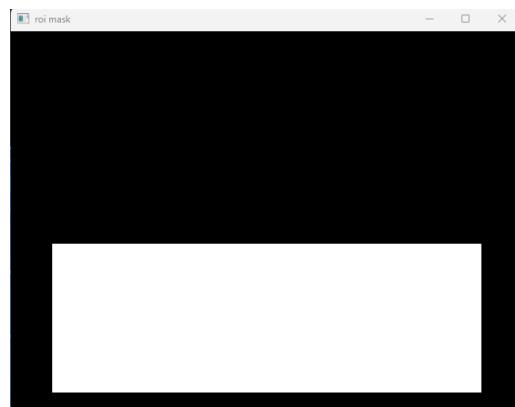
İlgili Alanı Bölgesi, bir görüntünün üzerinde odaklanılacak ve analiz edilecek belirli bir kısmını tanımlayan dikdörtgen, dairesel veya serbest şekilli bir bölgedir. Görüntünün tamamına odaklanmak yerine belirlenen bölgede işlem yapılmasını sağlar.

Bu fonksiyon, verilen bir görüntüde belirli bir bölgeyi (ROI) maskeleme ve yalnızca bu bölgeyi içeren bir görüntü oluşturma işlevini gerçekleştirir. Fonksiyonun adımları şu şekildedir:

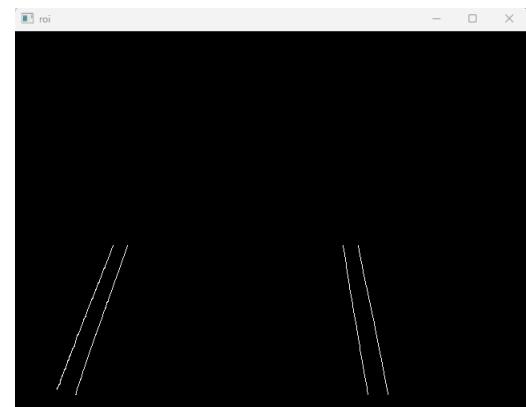
- Görüntünün boyutları alınır.
- Görüntü ile aynı boyutlarda tamamen siyah bir maske oluşturulur.
- Maskede belirli koordinatlarla beyaz bir dikdörtgen çizilir.
- Orijinal görüntü ile maske bitwise AND işlemiyle birleştirilerek sadece belirlenen bölge (ROI) korunur.
- İsteğe bağlı olarak maske ve ROI görüntülenir.
- ROI görüntüsü döndürülür.



(a) Canny algoritması



(b) Roi için belirlenen maske



(c) Sonuç

Şekil 3.4: ROI ile alanın belirlenmesi

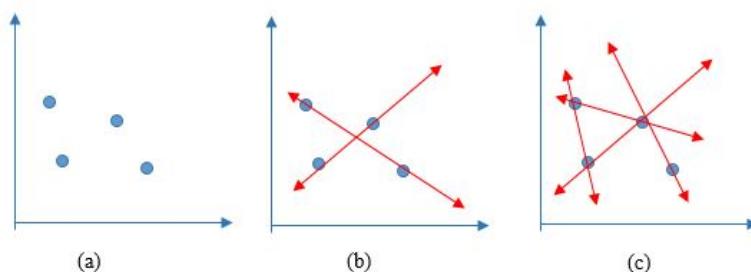
3.4 Hough Dönüşümü

Hough Dönüşümü, bir görüntüdeki çizgiler ve daireler gibi şekilleri tespit eden bir bilgisayarla görme tekniğidir. Bu dönüşüm, bu şekilleri parametre uzayında matematiksel temsillere dönüştürerek, kırık veya gizlenmiş olsalar bile tespit etmeyi kolaylaştırır. Bu yöntem, görüntü analizi, desen tanıma ve nesne tespiti için değerlidir [17].

Hough Dönüşüm algoritması, görüntü analizi, bilgisayarla görme ve dijital görüntü işleme alanlarında bir özellik çıkarma yöntemidir. Belirli bir form sınıfı içindeki nesnelerin kötü örneklerini tanımlamak için bir oylama mekanizması kullanır. Bu oylama mekanizması, parametre uzayında gerçekleştirilir. İlk olarak, HT algoritması, birikim alanında yerel maksimumlar olarak nesne adayları üretir [17].

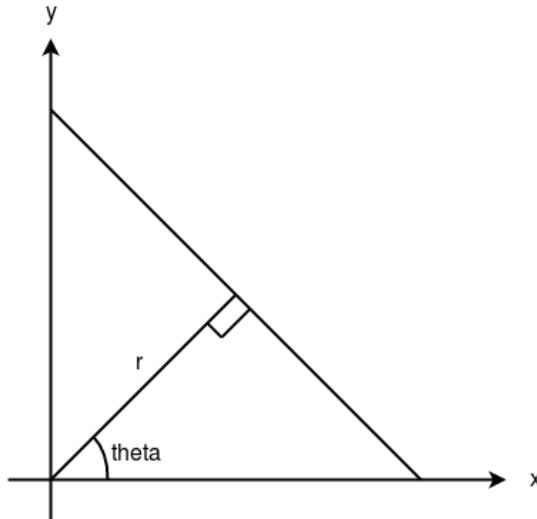
Görüntü verisindeki veya kenar dedektöründeki kusurlar ve ideal çizgi/daire/elips ile kenar dedektörü tarafından elde edilen gürültülü kenar noktaları arasındaki uzaysal farklılıklar nedeniyle gerekli egrilerde eksik noktalar veya pikseller olabilir. Sonuç olarak, çıkarılan kenar özelliklerini uygun bir çizgi, daire veya elips koleksiyonu halinde gruplandırmak çoğu zaman zordur [17].

Hough yaklaşımı, çözüm sınıflarının sayısının sağlanmasına gerek olmadığı durumlarda (potansiyel olarak gürültülü) bir özelliğin/özelliklerin genel bir tanımını hesaplamak için etkilidir. Örneğin, hat tanımlamaya yönelik Hough yaklaşımı, her girdi ölçümünün küresel olarak tutarlı bir çözüme (örneğin, o görüntü noktasına yol açan fiziksel çizgi) katkısını yansıttığı varsayımini harekete geçirir [17].



Şekil 3.5: Şekil 3.5: Hough Dönüşümü ile Çizgi Tespiti: (a) Orijinal noktalar, (b) Noktalardan geçen olası çizgiler, (c) Parametre uzayında noktaların kesimi.

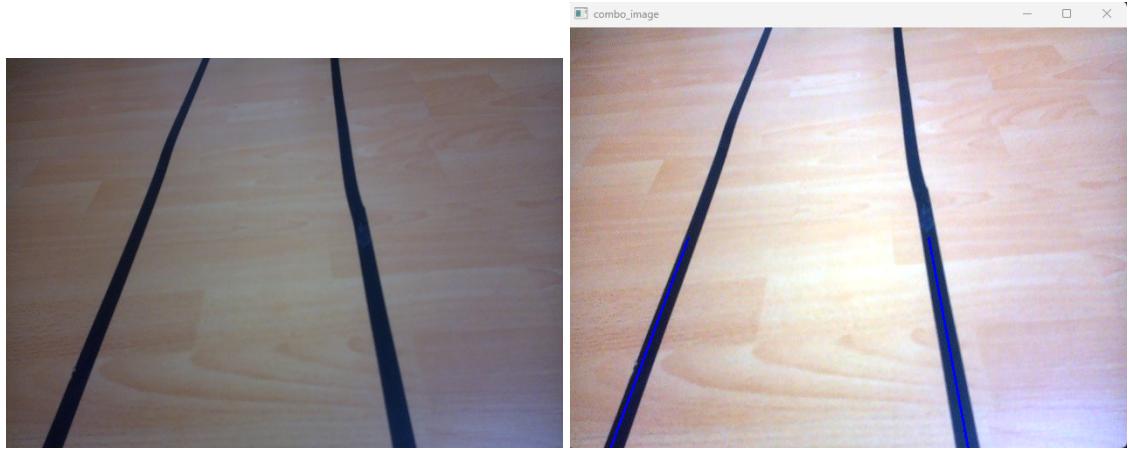
Bir çizgi analitik olarak çeşitli şekillerde tanımlanabilir. Doğru denklemlerinden biri parametrik veya normal kavramını kullanır: $x\cos\theta + y\sin\theta = r$. Burada r , Şekil 3.6'da verildiği gibi, orijinden bu çizgiye kadar olan bir normalin uzunluğu ve θ ise yönelimdir [17].



Şekil 3.6: Çizgi Tespiti

Görüntüdeki bilinen değişkenler (yani xi, yi) parametrik çizgi denklemindeki sabitlerdir, oysa r ve aradığımız bilinmeyen değişkenlerdir. Kartezyen görüntü uzayındaki noktalar, eğer her biri tarafından belirtilen potansiyel (r, θ) değerlerini çizersek, polar Hough parametre uzayındaki eğrilere (yani sinüzoidlere) karşılık gelir. Düz çizgiler için Hough Dönüşümü algoritması bu noktadan eğriye dönüşümdür. Kartezyen görüntü uzayındaki eşdoğrusal noktalar, Hough parametre uzayında incelemişinde belirgin hale gelir çünkü tek bir (r, θ) noktasında örtüsten eğriler sağlarlar [17].

A ve b dairenin merkez koordinatlarıdır ve r ise yarıçaptır. Algoritmanın hesaplama karmaşıklığı artırıyor çünkü artık parametre uzayında üç koordinatımız ve bir 3 boyutlu toplayıcımız var. (Genel olarak parametre sayısı polinom olarak hesaplamayı ve akümülatör dizisinin boyutunu artırır.) Sonuç olarak, burada açıklanan temel Hough yaklaşımı yalnızca düz çizgiler için geçerlidir [17].



(a) Orijinal Görüntü

(b) HoughLine ile Çizginin Tespiti

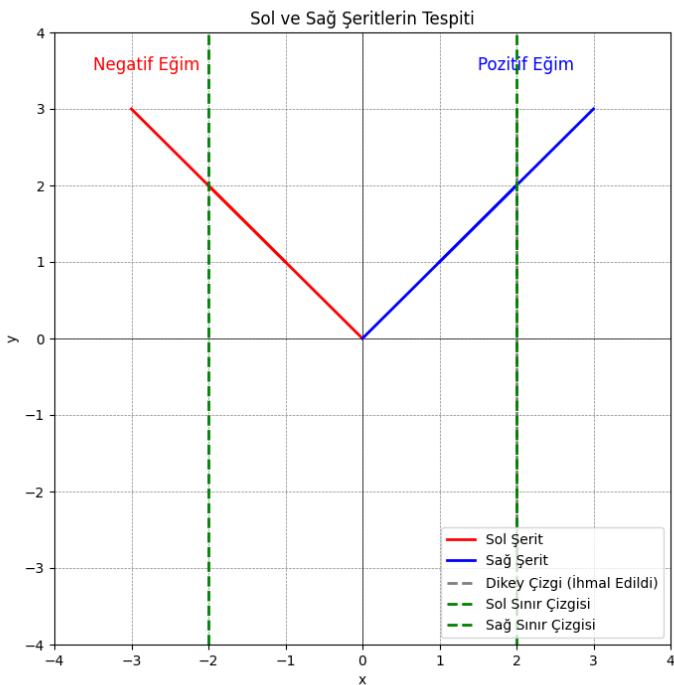
Şekil 3.7: HoughLine Transformu ve Çizgilerin Görüntülenmesi

3.5 Eğim ve Kesişim ile Sağ ve Sol Çizgileri Bulma

Bir doğrunun denklemi $y = mx + c$ olarak verilir. Burada m doğrunun eğimi, c ise doğrunun y -eksenini kestiği noktadır. Hough dönüşümü kullanılarak tespit edilen doğru parçalarının eğimlerinin ve kesişim noktalarının ortalaması hesaplanması gereklidir. Görüntüde sol şeritteki çizgiler negatif bir eğime sahiptir. Sol şerit çizgisinde $x_1 < x_2$ ve $y_2 < y_1$ olur buda $(y_2 - y_1)/(x_2 - x_1)$ eğimini verir. Bu eğim negatiftir. Bu nedenle, negatif eğime sahip olan tüm çizgiler sol, pozitif eğime sahip olan tüm çizgiler ise sağ çizgi olarak kabul edilir.

Dikey çizgiler $x_1 == x_2$ durumunda, eğim sonsuz olacaktır. Bu durumda hata almak için tüm dikey çizgiler göz ardı edilir.

Bu tespitin doğruluğunu arttırmak için, her kare iki bölgeye (sağ ve sol) iki sınır çizgisi ile ayrılır. Sağ sınır çizgisinin sağındaki tüm genişlik noktaları (x -eksen noktaları) sağ şerit hesaplamasına dahil edilir. Sol sınır çizgisinin solundaki tüm genişlik noktaları ise sol şerit hesaplamasına dahil edilir. Şekil 3.8 de örnek bir çizgi belirlenmesinin grafiği gösterilmiştir.

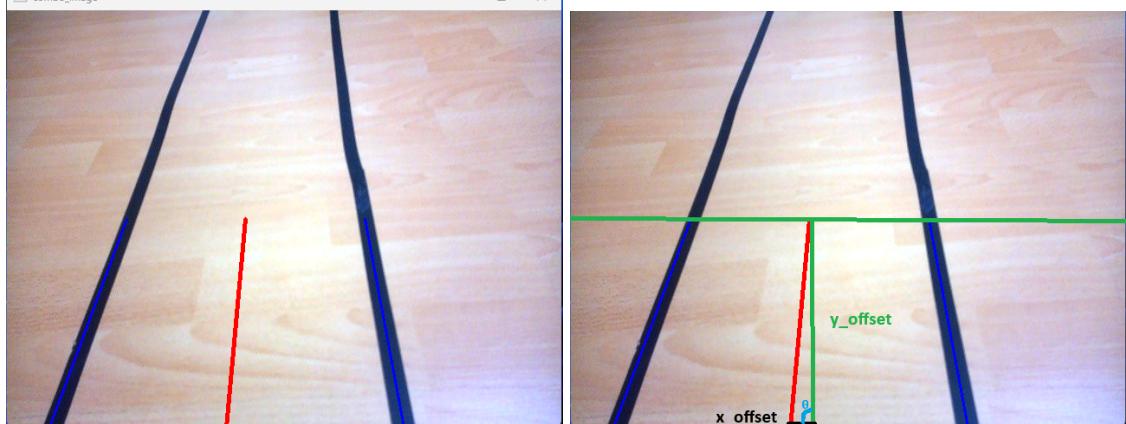


Şekil 3.8: Sağ ve Sol Çizgilerin Belirlenmesi

3.6 Direksiyon Açısının Hesaplanması

Sağ ve sol çizgiler belirlendikten sonra sağ ve sol çizginin tam ortasında olacak şekilde bir çizgi daha çizilir. Bu çizgi sağ ve sol çizginin hareketlerini taklit eder. Böylece çizgiler ne tarafa dönüyorsa orta çizgi de o tarafa döner. Orta çizgiye yön çizgisi adı verilir. Görüntünün x ekseninin tam ortasında duran bu görüntü açı üretmektedir. Ürettiği açı dik ise 90 derecedir. Açı sola doğru azalırken sağ doğru artar. Buda eğer 90'nın üstünde ise arabanın sağ 90'nın altında ise aracın sola döneneceği anlamına gelmektedir. Şekil 3.9'da yön çizgisi gösterilmiştir.

combo_image

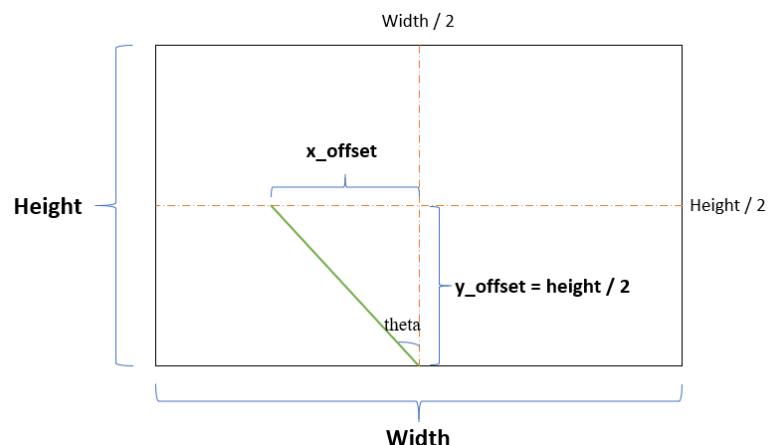


(a) Yön Çizgisi

(b) Açı Hesaplaması

Şekil 3.9: Yön Çizgisi

Orta çizginin x eksenine olan düzleme göre eğim ve kesişimi hesaplanır. Eğer iki çizgi algılanırsa orta çizgiden açı hesabı yapılır. Eğer tek çizgi algılanırsa algıladığı çizgiye göre açı üretecektir.



Şekil 3.10: Yön Çizgisine Göre Açı Üretmek

Şekil 3.10'a göre eğer çift çizgi algılanırsa:

- İki çizgi için:

$$\begin{aligned} \text{center_x} &= \text{width}/2 \\ \text{y_offset} &= \text{height}/2 \end{aligned} \quad (3.1)$$

$$\text{average_point} = \frac{l1x2 + l2x2}{2} \quad (3.2)$$

$$\text{x_offset} = \text{average_point} - (\text{center_x}) \quad (3.3)$$

- Tek çizgi için:

$$\text{slope} = x_2 - x_1 \quad (3.4)$$

$$\text{x_offset} = \text{slope} \quad (3.5)$$

Çizgiler algılandıktan ve x_offset değeri bulunduktan sonra direksiyon için açı hesaplamasına geçilir.

$$\text{angle_to_middle_vertical_rad} = \arctan\left(\frac{\text{x_offset}}{\text{y_offset}}\right) \quad (3.6)$$

$$\text{angle_to_middle_vertical_deg} = \text{angle_to_middle_vertical_rad} \times \frac{180.0}{\pi} \quad (3.7)$$

$$\text{steering} = \text{angle_to_middle_vertical_deg} + 90 \quad (3.8)$$

4 YAPAY ZEKA MODELİ İLE NESNE ALGILAMA

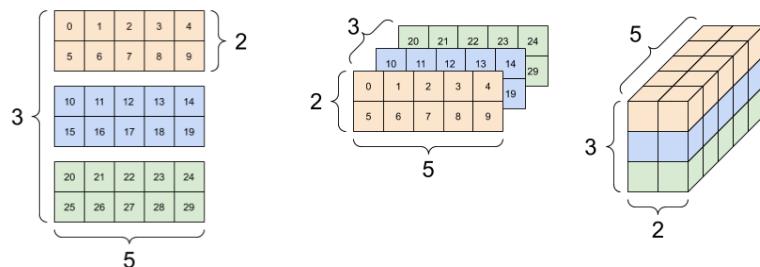
TensorFlow Lite kullanılarak eğitilen yapay zeka modeli ile nesne algılama sistemi geliştirilmiştir. Kameradan gelen görüntüler, Raspberry Pi tarafından işlenir ve TensorFlow Lite modeli kullanılarak sınıflandırılır. Model, dur işaretti, kırmızı ışık, sarı ışık ve yeşil ışık gibi trafik işaretlerini algılayabilir. Algılama sonucunda elde edilen nesne bilgileri, string veri formatında Arduino Nano'ya gönderilir.

4.1 TensorFlow

TensorFlow, Google tarafından geliştirilen açık kaynaklı bir derin öğrenme kütüphanesidir. Makine öğrenimi ve derin öğrenme modelleri oluşturmak, eğitmek ve dağıtmak için güçlü ve esnek bir araç seti sunar. 2015 yılında piyasaya sürülen TensorFlow, günümüzde yapay zeka (AI) ve makine öğrenimi (ML) projelerinde yaygın olarak kullanılmaktadır [18].

4.1.1 Tensörler

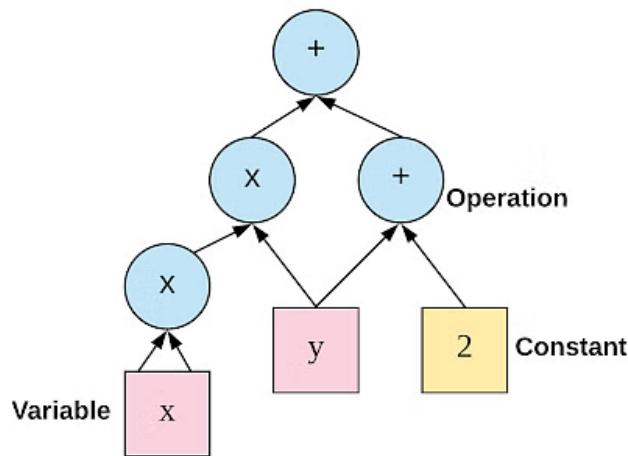
TensorFlow'un temel bileşeni olan tensörler, tüm veri türlerini temsil eden çok boyutlu matris veya vektörlerdir. Tensörler, belirli bir şekle ve veri türüne sahip değerler içerir ve tüm TensorFlow hesaplamalarının temelini oluşturur. Tensörler, hesaplama sonucunda veya giriş verisi olarak kullanılabilir. Tüm işlemler, bir dizi ardışık hesaplamadan oluşan grafiklerde gerçekleşir. Grafikler, işlem düğümleri ve bunların birbirleriyle bağlantılarını içerir. Her düğümün kenarı, tensörler aracılığıyla veri taşıyan bağlantılardır. Bu şekilde, TensorFlow veri işleme ve hesaplama süreçlerini yönetir [19].



Şekil 4.1: Üç Eksenli Tensör

4.1.2 Graf Yapısı

TensorFlow, hesaplamaları bir graf yapısı olarak tanımlar. Bu graf, düğümler (nodes) ve kenarlar (edges) içerir. Düğümler, matematiksel işlemleri temsil ederken, kenarlar tensorları taşıır. Bu yaklaşım, karmaşık hesaplamaların düzenlenmesini ve optimize edilmesini sağlar [19].



Şekil 4.2: Graf Yapısı

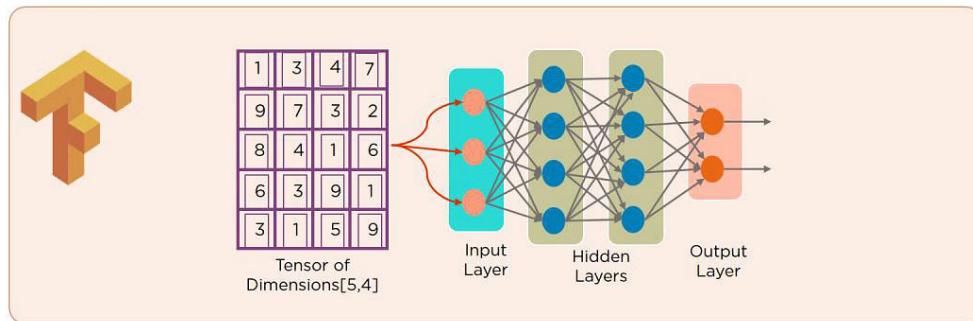
TensorFlow'un Avantajları Nelerdir?

- Açık kaynak kütüphanesine sahiptir.
- Çalıştırması kolaydır.
- Hızlı hata ayıklama yapar.
- Ölçeklenebilir.
- Çok yönlüdür.

4.1.3 TensorFlow Nasıl Çalışır?

TensorFlow, Python, C ve C++ API'leri içeren bir modül seti kullanarak hesaplamaların oluşturulmasını ve yürütülmesini sağlar. Hesaplamalar, etkileşim durumunu takip eden veri akışı grafikleri ile gerçekleştirilir. TensorFlow, düğüm adı verilen veri katmanlarıyla giderek daha karmaşık verileri ortaya çıkarır; örneğin, yuvarlak bir şekli tanıtmaktan bir kedinin gözünü tanıtmaya kadar. TensorFlow, hesaplama grafikleri oluşturarak bunları he-

men yürütülebilir, kaydedilebilir veya farklı platformlarda çalıştırılabilir hale getirir. Bu grafikler, kod dağıtımını gerektirmeden üretim ortamlarına da aktarılabilir ve herhangi bir platformda kolayca optimize edilebilir [20].



Şekil 4.3: TensorFlow Çalışma Prensibi

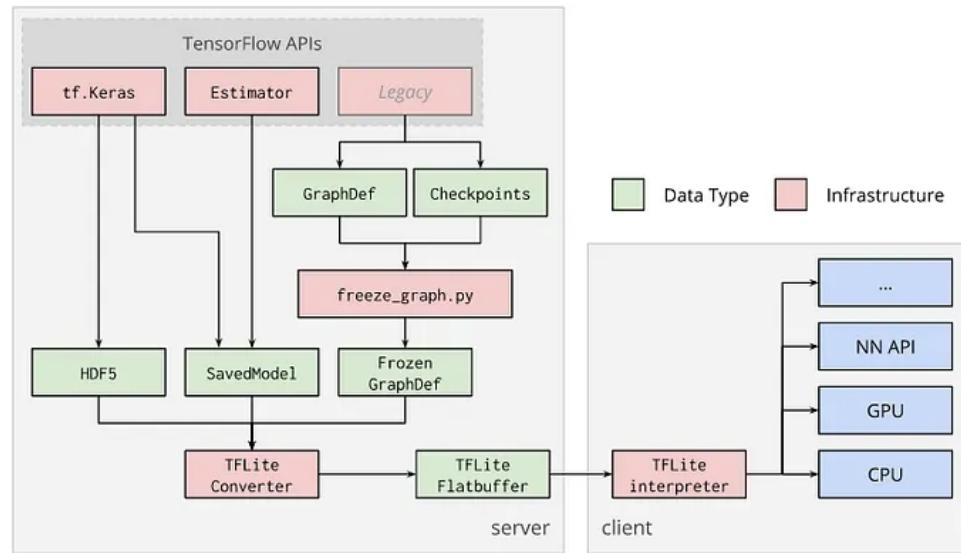
4.2 TensorFlow Lite Nedir?

TensorFlow-Lite, TensorFlow modellerinin mobil cihazlarda, gömülü sistemlerde ve IoT cihazlarda daha verimli çalıştırılması için geliştiriciler tarafından oluşturulmuş bir araç setidir. Bu araç seti, cihaz üzerinde makine öğrenmesi çalıştmak için optimize edilmiş ve küçük boyutlu kernel işlemlerini destekler, böylece cihaz ayarlarına ve uygulamalarına uyum sağlar [21].

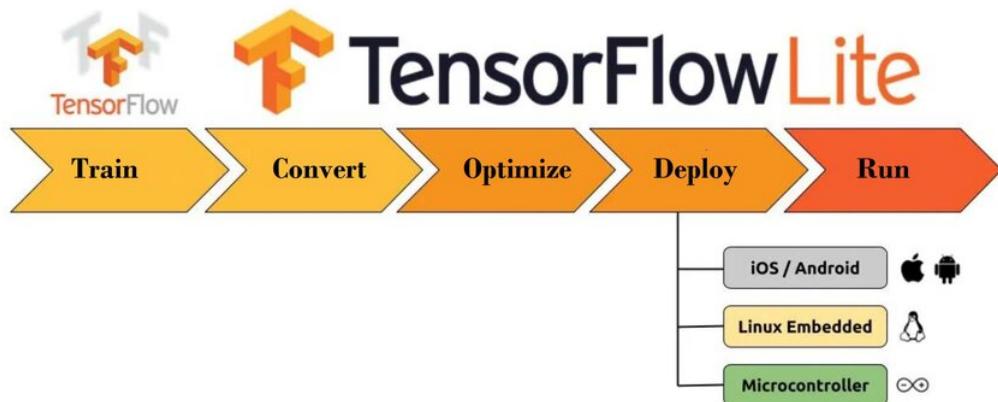
İki ana birimden oluşur:

- **TensorFlow-Lite Yorumlayıcı:** Optimize edilmiş modelleri farklı donanımlarda çalıştmak için sade ve hızlı olarak tasarlanmıştır. TensorFlow-Lite modeli üretmek için tf-lite yorumlayıcısı kullanılır ve bu yorumlayıcı, minimum yük, başlatma ve uygulama gecikmelerini sağlamak için statik bir grafik ve özel bir bellek ayıracı kullanır. Ancak, tf-lite yorumlayıcısı şu anda sınırlı sayıda TensorFlow operatörünü desteklemektedir, bu da bazı modellerin tf-lite ile çalışabilmesi için ek işlemler gerektirebileceği anlamına gelir [21].
- **TensorFlow-Lite Dönüştürücü:** TensorFlow modellerinin tf-lite tarafından kullanılabilmesi için dönüştürülmesini sağlar, böylece ikili dosya boyutu küçülür ve performans optimizasyonları yapılır. Dönüştürücü, TensorFlow modellerini optimize

edilmiş FlatBuffer formatına dönüştürerek ".tflite" uzantılı modeller oluşturur ve bunlar tf-lite tarafından kullanılabilir. Dönüştürmenin desteklediği formatlar; SavedModels, Frozen Graph Def ve tf.Session ile alınan herhangi bir modeldir (yalnızca Python-API ile kullanılabilir) [21].



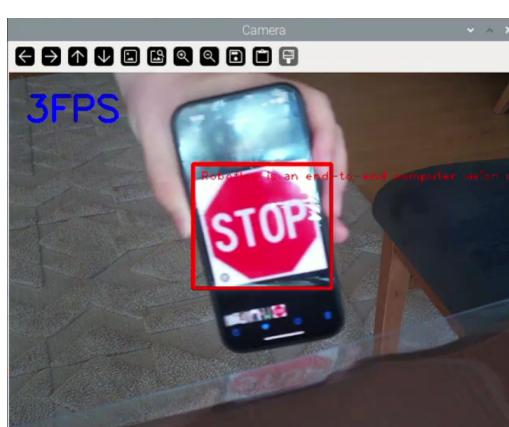
Şekil 4.4: Dönüşüm İşleminin Diagramı



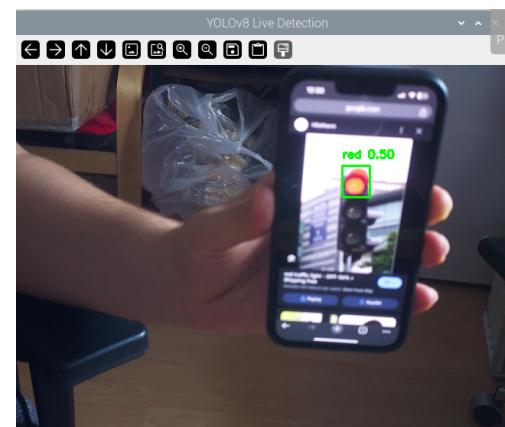
Şekil 4.5: TensorFlow Lite İşleme Adımları

4.3 Eğitilmiş Modelin Raspberry Pi ile Uyumu

Raspberry pi de denenmek için veriseti oluşturulmuştur. Oluşturulan veriseti pascal voc veri setidir. Bu veri seti tensorflow lite ile eğitilmiş ve sign.tflite modeli oluşturulmuştur. Ardından bu model raspberry pi de denenmek için raspberry pi'ye aktarılmıştır ve sonuçlar gayet olumludur. 3-5 FPS ile veri okuma yapılmaktadır. Şekil 4.6'nın a kısmında stop, b kısmında kırmızı ışık, c kısmında turuncu ışık ve d kısmında ise yeşil ışığı algıladığı gösterilmiştir.



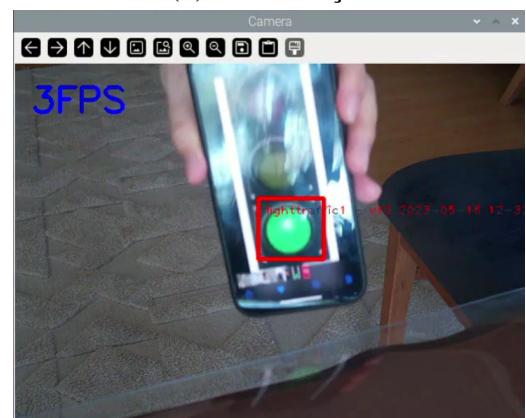
(a) Dur Tabelası



(b) Kırmızı Işık



(c) Turuncu Işık

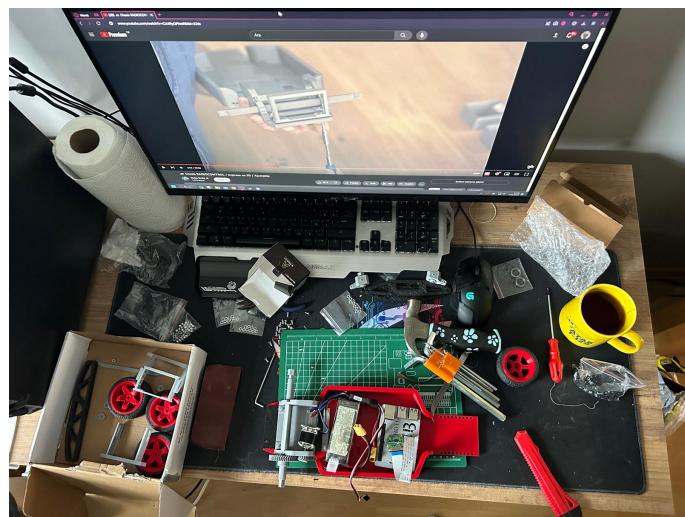


(d) Yeşil Işık

Şekil 4.6: TFLite Modelinin Raspberry Pi Üzerinde Testi

5 OTONOM RC ARACIN GERÇEKLEŞTİRİMİ

Şekil 5.1 de 3D yazıcı ile aracın bütün parçaları basılmış ve montajlama kısmına geçilmiştir.



Şekil 5.1: RC Araba Montaj

Şekil 5.2 de montajlanan parçaların a kısmında üstten b kısmında ise yandan gösterimi mevcuttur. Aracın boş hali gösterilmiştir.



(a) (b)

Şekil 5.2: Montaj Bitmesi

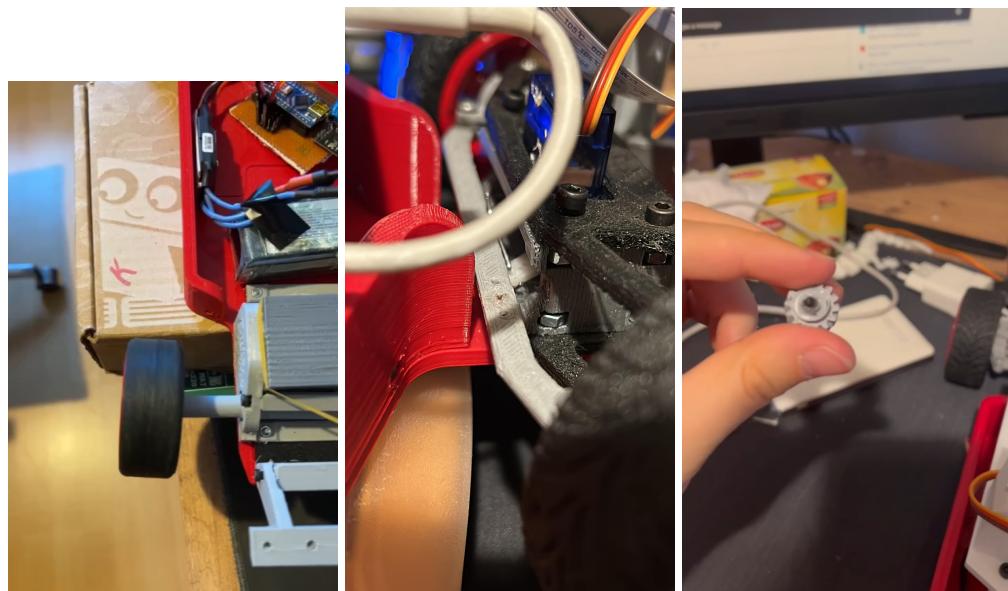
Şekil 5.3 ün a kısmında basit bir kablosuz haberleşme ile RC Araç gerçekleştirilmiştir. RC Aracın motor bölümündeki uyumsuzluktan dolayı küp şeklinde yeni bir parça tasarlanmış ve firçasız motor bu parçaya montajlanmıştır. Ancak firçasız motorun uyumsuz olmasından dolayı araç zorlandığında dışliyi içten oymakta ve motorun mili boşça çıkmaktadır. Bu durum b kısmında gösterilmiştir.



(a) Fircasiz motor ile de- (b) Fircasiz motor prob-
neme lemi

Şekil 5.3: Fircasiz Motor

Şekil 5.4’ün tüm kısımları başlıca montaj hataları mevcuttur. Örneğin a kısmında tekerlein yanık dönmesi, b kısmında ise mini servonun dönüş esnasında boşça çıkması ve c kısmında ise değiştirilen firçasız motor yerine dc motorun uyumsuzluğundan kaynaklı iç çap büyümesi gösterilmiştir.



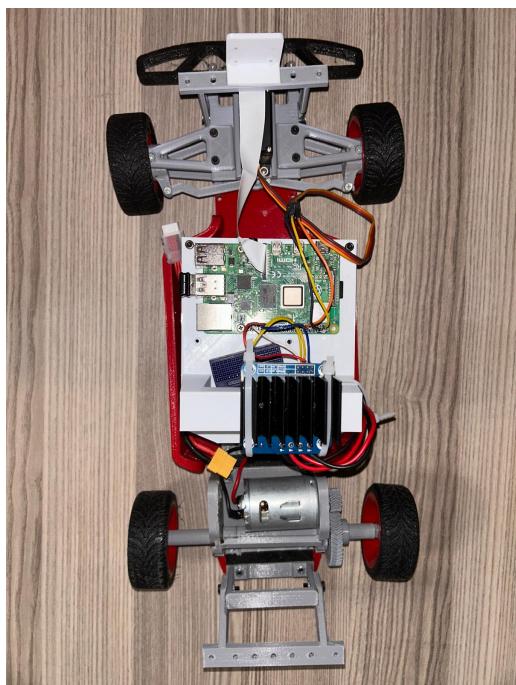
(a) Yamuk Tekerlek (b) Tel Tokanın Yerinden Çıkması (c) Motorun Dişliyi Yemesi

Şekil 5.4: Araç Problemleri

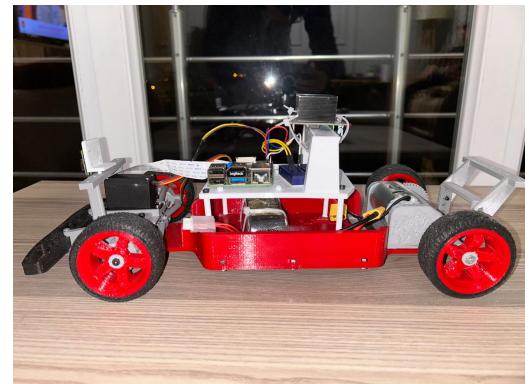
Şekil 5.5'de Mekanik problemlerin çoğu giderilmiş ve donanımlarda montajlanarak ilk versiyonu test edilmek üzere tamamlanmıştır. a kısmında servo motorun değiştiği, b kısmında ise yuvaya tam uyan dc bir motorun montajlandığı görülmektedir. c kısmı aracın yanidan görünümüdür.



(a) RC Araç v1 ön yüz



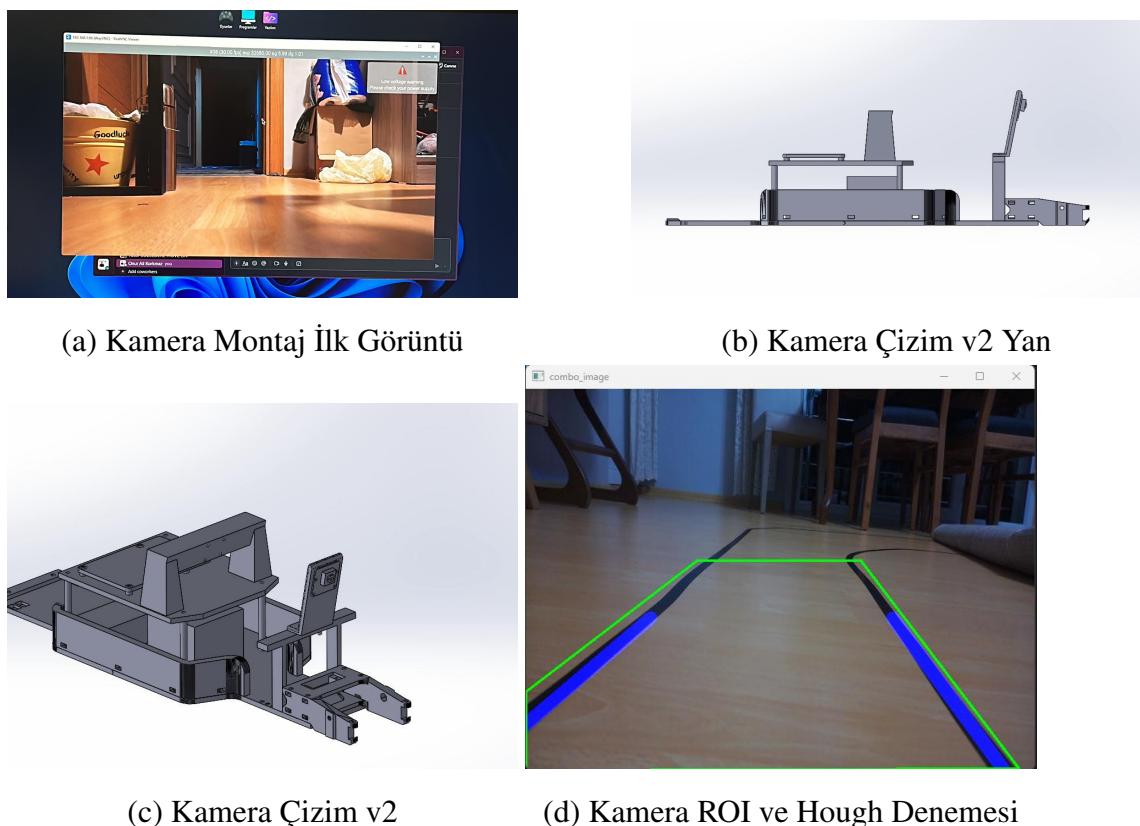
(b) RC Araç v1 üst yüz



(c) RC Araç v1 yan yüz

Şekil 5.5: RC Araç V1

Şekil 5.6'da kameranın aracın en önünde ve yere çok yakın olması a kısmında gösterilmiştir. Bu durum aracın sapmasına ve çizgileri tam olarak algılayamamasına sebep olmuştur. Kameranın montajlanacağı yer uzatılarak eğim verilmiştir. Bu çizim b ve c kısımlarında gösterilmiştir. Daha sonra kamera montajı yapıldıktan sonra d kısmında ise bu kamera açısı ile çizgilerin daha iyi şekilde algılandığı görülmektedir. Fakat burada da çizginin x'e sıfır olan bir eksenden çıkmamasından dolayı açı hesaplamalarında problemler meydana gelmektedir.

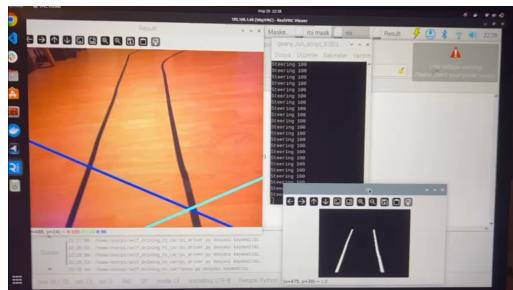


Şekil 5.6: Kamera ve Montajını Geliştirme

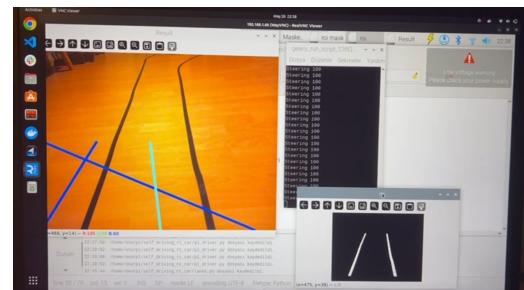
Şekil 5.7' de kamera montajı değiştirilmiştir. Bu sefer açı ve yüksekliği ayarlanacak şekilde olan bir aparat düşünülmüştür. Bu aparat 3D yazıcıdan çıkarılıp araç için hazır hale getirilmiştir. Aparat a kısmında gösterilmiştir. b ve c kısmında ise aparat araca takıldıktan sonraki görüntüler gözükmemektedir. Görüntü açısı ve yüksekliği iyi olmasına karşı hough transformunda sapıtmalar mevcuttur. Bunlar ise daha sonra düzeltilmiştir.



(a) Kamera Çizimi v3



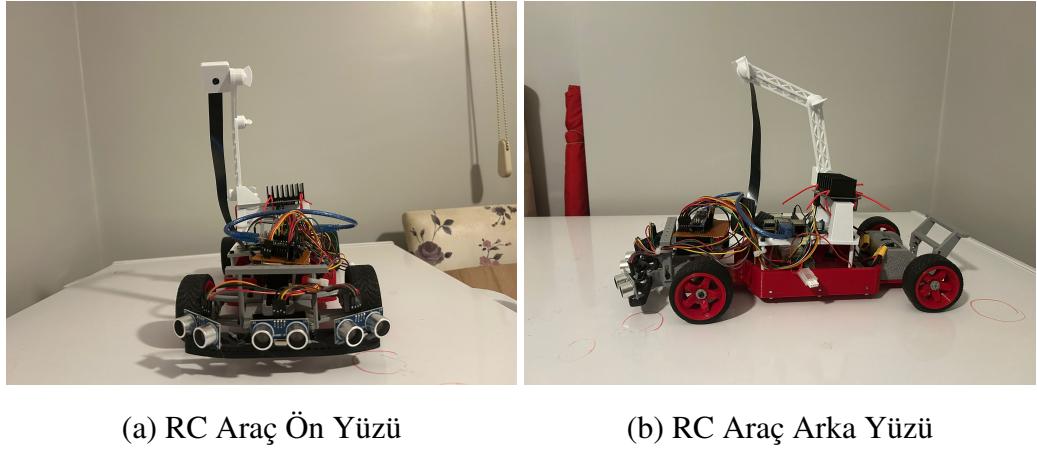
(b) Hough Kayması



(c) Hough Kayması 2

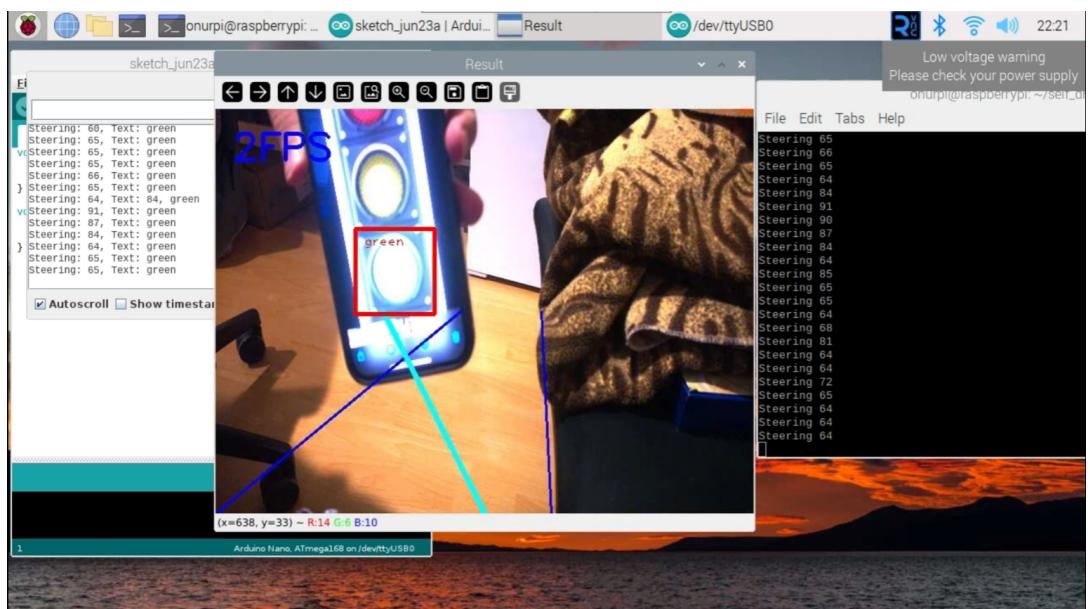
Şekil 5.7: Kamera v3 ve Görüntüdeki Kaymalar

Şekil 5.8'de aracın son sürümü gözükmektedir. İlk versiyondan farklı olarak sisteme arduino da eklenmiştir. Sensörler ve motorların hepsi arduino tarafından yönetilmektedir. a ve b kısımlarında arduino gözükmektedir.



Şekil 5.8: RC Araç Ön ve Arka Yüzü

Şekil 5.9'da eğitilmiş olan tflite modeli ve görüntü işleme birlikte çalışmaktadır. Ölçülen açı değeri ve algılanan nesne arduinoya iletmektedir. Böylece arduino servo motora açı değerini iletirken dc motor'a da yeşil olduğu için ilerle komutu iletmektedir.



Şekil 5.9: TFLite Model ve Görüntü İşleme

5.1 Arduino Nano Entegrasyonu

İşlem, veri alımı ile başlar ve bu veri, Raspberry Pi'den USB seri haberleşme protokolü ile Arduino Nano'ya iletilir. Ardından, ultrasonik sensörler kullanılarak çevredeki engellerin mesafesi ölçülür. Eğer herhangi bir engel 30 cm'den daha yakınsa, motor durdurulur. Aksi takdirde, araç sabit bir hızda ilerlemeye devam eder. Her iki durumda da, alınan açı değeri servo motoruna iletilir ve bu değer servo motor tarafından uygulanır. Bu süreç, sürekli olarak tekrarlanarak aracın çevresine uyum sağlaması ve doğru yönlendirilmesi sağlanır. Şekil 5.10'daki akış diyagramı, bu adımları görsel olarak temsil ederek sürecin anlaşılmasını kolaylaştırmaktadır.



5.2 Seri Haberleşme Protokollerı

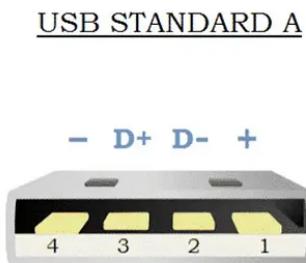
Sistemler arası protokoller, iki iletişim cihazı arasında iletişim kurar. Haberleşme veri yolu sistemi aracılığıyla iletişim kurar. Sistemler arası protokoller 3'e ayrılır bunlar:

- USB Haberleşme Protokollerı

- UART Haberleşme Protokolleri
- USART Haberleşme Protokolleri

5.2.1 USB haberleşme protokolü

Universal Serial Bus (USB) iki kablolu seri haberleşme protokülüdür. Tak ve çalıştır ile çalışır. USB Protokolü D+ ve D- veri sinyali yoluyla ana bilgisayardan çevre bilgisayara ya da mikrodenetleyiciye bilgi gönderir ve alır. Bu veri hatları dışında VCC ve GND hatlarında bulunmaktadır. Şekil 5.11'da pin çıkışları gösterilmiştir [22].



Şekil 5.11: USB Bağlantı Uçları

Raspberry pi tarafından üretilen açı değeri USB seri haberleşme protokolü ile Arduino nano'ya gönderilir.

5.3 Arduino Nano DC ve Servo Motor Kontrolü

Arduino Nano, ilk olarak seri haberleşme ile gelen veriyi okur. Ardından ultrasonik sensörlerden gelen verileri işler. Eğer ultrasonik sensörlerden alınan mesafe verisi 30 cm'nin altındaysa, ana motorun gücünü keser. 30 cm'nin üzerindeyse, ana motora sabit bir hız verir. Gelen açı değerini 40 ila 140 arasında haritalandırarak, bu açı değerini servo motora gönderir.

```
#include <Arduino.h>
#include <HCSR04.h>
#include <Servo.h>

#define IN1 4
#define IN2 5
#define ENA 6
#define servoPin 11

UltraSonicDistanceSensor distanceSensorRight(2, 3); // trig, echo
UltraSonicDistanceSensor distanceSensorMid(7, 8); // trig, echo
UltraSonicDistanceSensor distanceSensorLeft(9,10); // trig, echo

Servo frontServo;

int distanceRight = 0;
int distanceMid = 0;
int distanceLeft = 0;
int steeringInt = 90;
int mapSteeringInt = 0;
int throttleInt = 0;
```

Şekil 5.12: Arduino Tanım Kodları

Şekil 5.12'de arduino kütüphaneler, değişkenler tanımlanır, servo motor nesnesi tanımlanır ve ultrasonik mesafe sensörü için trig, echo pinleri tanımlanmaktadır.

```
void setup () {
    Serial.begin(115200);
    Serial.setTimeout(10);

    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    frontServo.attach(servoPin);
}
```

Şekil 5.13: Arduino Setup Kodları

Şekil 5.13'de seri haberleşme başlatılır, motor sürücü pinleri çıkış olarak ayarlanır. Servo motor atanır.

```

void motor_control(int throotle){
    if (throotle > 0){
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(ENA, throotle);
    }
    else if(throotle < 0){
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
        analogWrite(ENA, -throotle);
    }
    else{
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        analogWrite(ENA, 0);
    }
}

```

Şekil 5.14: Motor Kontrol Fonksiyonu

Şekil 5.14'te motorun yönü kontrol edilir. Eğer throotle değeri pozitif ise motor ileri, negatif ise motor geri hareket eder. throotle değeri 0 ise motor durur.

```

void mesafe_data(int distanceLeft, int distanceMid, int distanceRight){
    if ((distanceLeft >= 0 && distanceLeft <= 30) || (distanceMid >= 0 && distanceMid <= 30) || (distanceRight >= 0 && distanceRight <= 30)){
        //Serial.println("MOTOR DURUR");
        motor_control(0);
    }
    else if(distanceLeft == -1 && distanceMid == -1 && distanceRight == -1){
        motor_control(0);
    }
    else{
        //Serial.println("motor calisiyor.");
        motor_control(85);
    }
}

```

Şekil 5.15: Mesafe Ölçüm Fonksiyonu

Şekil 5.15'te ultrasonik sensörden gelen mesafe kontrol edilir. Mesafe 30cm'in altında ise motor durur. 30cm'in üstündeyse motor sabit bir hızda çalışır.

```

void read_data(){
    if(Serial.available() > 0){
        String steering = Serial.readStringUntil('\n');
        Serial.println("Steering: " + steering);
        steeringInt = steering.toInt();
    }
}

```

Şekil 5.16: Gelen Veriyi Değişkende Saklayan Fonksiyon

Şekil 5.16'de usb port ulaşılabilir olduğu sürece haberleşmeden gelen veri okunur ve

steeringInt değişkenine atanır.

```
void servo_control(){
    steeringInt = constrain(steeringInt, 40, 140);
    mapSteeringInt = map(steeringInt, 0, 180, 900, 2100);
    //Serial.println("Servo angle: " + String(steeringInt));
    //Serial.println("Map Servo angle: " + String(mapSteeringInt));
    frontServo.writeMicroseconds(mapSteeringInt);
}
```

Şekil 5.17: Servo Motor Kontrol Fonksiyonu

Şekil 5.17'da gelen açı değeri 40 ile 140 arasında sınırlanır. Servo motorun mikro-saniye değeri haritalandırılır ve servo motor bu değere göre ayarlanır.

```
void loop () {
    read_data();

    distanceLeft = distanceSensorLeft.measureDistanceCm();
    distanceMid = distanceSensorMid.measureDistanceCm();
    distanceRight = distanceSensorRight.measureDistanceCm();

    // Serial.println("Mesafe Sol: " + String(distanceLeft));
    // Serial.println("Mesafe Orta: " + String(distanceMid));
    // Serial.println("Mesafe Sağ: " + String(distanceRight));

    measure_data(distanceLeft, distanceMid, distanceRight);
    delay(500);
    servo_control();
}
```

Şekil 5.18: Loop Fonksiyonu

Şekil 5.18'de sürekli olarak seri veriyi okur, ultrasonik sensörden mesafe verilerini alır ve motor kontrolü ile servo kontrolünü gerçekleştirir. 500 ms gecikme ile döngü tekrarlanır.

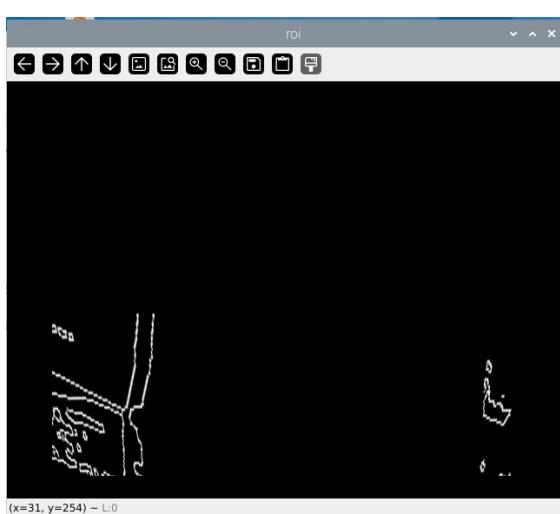
6 SONUÇLAR VE ÖNERİLER

Sonuç olarak, Raspberry Pi ile otonom RC araç ile test denemeleri gerçekleştirilmiş ve sonuçlar her ne kadar mükemmel olmasa da, ideal bir stabiliteye ulaşılmıştır. Uygun ortam ve koşullarda araç, sorunsuz bir şekilde çizгиyi takip edebilecek kabiliyettedir. Ancak, projede karşılaşılan bazı sorunlar ve hatalar da göz önünde bulundurulmalıdır.

Özellikle, kameranın algıladığı görüntülerde parkenin parlaması sonucu oluşan bozulmalar, çizginin yanlış tespit edilmesine ve bu nedenle üretilen açının hatalı bir şekilde Arduino'ya gönderilmesine yol açmaktadır. Bu durum, aracın yanlış yönlendirilmesine ve düzgün bir dümenleme yapamamasına neden olmaktadır. Kameradan gelen görüntülerin parlaması, araç stabilitesini olumsuz etkileyen önemli bir faktördür.

Ayrıca, USB ile seri haberleşmede verilerin çok hızlı gelmesi ve bazı noktalarda veri kaybının yaşanması da başka bir sorundur. Bu durum, aracın bazen sabit bir 90 derece açısı üretirken aniden tam sağa kırmasına neden olmaktadır. Sürüş sırasında bu tür ani sağa dönmeler, aracın şeritten çıkışmasına ve istenmeyen yönlendirmelere yol açmaktadır.

Bir diğer etken ise, Raspberry Pi'nin bu tür projeler için yetersiz kalmasıdır. Yapay zeka entegrasyonu oldukça sıkıntılı olmakta ve çalışma kapasitesini düşürmektedir. Bu da tüm sistemin stabil çalışmasını engellemektedir. Raspberry Pi'nin sınırlı işlem gücü, özellikle gerçek zamanlı görüntü işleme ve yapay zeka algoritmalarının entegrasyonunda performans düşüklüğüne neden olmaktadır.



(a) Ekran Parlaması

101
101
100
100
100
101101
101
101
100
101
101
101
101
100

(b) Veri Kaçırma

Şekil 6.1: Düzeltmesi gereken hatalar

Proje kapsamında Raspberry Pi yerine Nvidia Jetson Nano kullanılarak geliştirme sağlanabilir. Böylece gerçek zamanlı görüntü işleme ve yapay zeka entegrasyonu daha etkin bir şekilde gerçekleştirilebilir. Ayrıca, ROS2 (Robot İşletim Sistemi) dahil edilerek daha gelişmiş bir otonom araç elde edilebilir.

KAYNAKLAR

- [1] <https://ieeexplore.ieee.org/document/8595590> [Ziyaret Tarihi: 2 Haziran 2024]
- [2] https://drive.google.com/file/d/19VUFGd5JM4TB_EQI_OWBcu3aYIvYy6GA/view [Ziyaret Tarihi: 2 Haziran 2024]
- [3] <https://ieeexplore.ieee.org/abstract/document/9065627> [Ziyaret Tarihi: 2 Haziran 2024]
- [4] <https://akademi.roboLinkMarket.com/raspberry-pi-nedir-raspberry-pi-ozellikleri-nelerdir/> [Ziyaret Tarihi 04 Haziran 2024]
- [5] <https://daha.net/blog/raspberry-pi-nedir-ne-icin-kullanilir/> [Ziyaret Tarihi 04 Haziran 2024]
- [6] https://www.robotiksistem.com/arduino_nano_ozellikleri.html [Ziyaret Tarihi 04 Haziran 2024]
- [7] <https://market.samm.com/raspberry-pi-kamera-3#:~:text=Raspberry%20Pi%20Kamera%20Mod%C3%BCl%C3%BC%203,mod%C3%BCllerinin%20en%20son%C3%BCyesidir!> [Ziyaret Tarihi 06 Haziran 2024]
- [8] <https://www.robotistan.com/bts7960b-20-amper-motor-surucu-karti> [Ziyaret Tarihi 06 Haziran 2024]
- [9] <https://www.elektrikport.com/universite/lipo-piller-ve-ozellikleri/21430#ad-image-0> [Ziyaret Tarihi 06 Haziran 2024]
- [10] <https://diyonet.net/lipo-pil/> [Ziyaret Tarihi 06 Haziran 2024]
- [11] <https://www.sailteknoloji.com/blog/hc-sr04-ultrasonik-mesafe-olcum-sensoru-ozelliikleri-nedir-nasil-calisir-b35.html> [Ziyaret Tarihi 06 Haziran 2024]
- [12] <https://maker.robotistan.com/dc-motor-cesitleri-nelerdir/> [Ziyaret Tarihi 06 Haziran 2024]

2024]

- [13] <https://diyot.net/sg90-servo-motor-ozellikleri/> [Ziyaret Tarihi 14 Haziran 2024]
- [14] <https://www.geeksforgeeks.org/color-spaces-in-opencv-python/> [Ziyaret Tarihi 20 Haziran 2024]
- [15] <https://medium.com/@alakhsharmacs/enhancing-image-light-intensity-in-opencv-using-hsv-408fdd85d6d2> [Ziyaret Tarihi 20 Haziran 2024]
- [16] <https://gist.github.com/mervetafrali/69f55d66d168b61cce84fbb91157e43b> [Ziyaret Tarihi 14 Haziran 2024]
- [17] <https://www.analyticsvidhya.com/blog/2022/06/a-complete-guide-on-hough-transform/#:~:text=Hough%20Transform%20is%20a%20computer%20vision%20technique%20to,for%20image%20analysis%2C%20pattern%20recognition%2C%20and%20object%20detection.> [Ziyaret Tarihi 19 Haziran 2024]
- [18] <https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow> [Ziyaret Tarihi 23 Haziran 2024]
- [19] <https://medium.com/@karakus.haciveli/tensorflow-2ye-giri%C5%9F-470ba3ed97a3> [Ziyaret Tarihi 23 Haziran 2024]
- [20] <https://bulutistan.com/blog/tensorflow-nedir/> [Ziyaret Tarihi 23 Haziran 2024]
- [21] <https://puffyy.medium.com/tensorflow-lite-ecae0b7a452a> [Ziyaret Tarihi 17 Haziran 2024]
- [22] <https://cenntceylnn.medium.com/elektronik-haberle%C5%9Fme-protokoller-i-nedir-d11a6d3a5957> [Ziyaret Tarihi 17 Haziran 2024]

ÖZGEÇMİŞ

KİŞİSEL BELGELER

Adı Soyadı : Onur Ali Korkmaz

Uyruğu : T.C.

Doğum Yeri ve Tarihi: Pendik - 29/04/1998

Adres : Aydınılı mah. Perçem sokak no:30 daire:8 kat:3 İstanbul/Tuzla

Telefon : (545) 421 22 44

E-mail : onur.korkmaz@hotmail.com

EĞİTİM DURUMU

Lisans Öğrenimi : BŞEÜ Bilgisayar Mühendisliği Bölümü

Bitirme Yılı : 2024

Lise : Tuzla Mesleki ve Teknik Anadolu Lisesi

İŞ DENEYİMLERİ

Yıl : 2023-2024

Kurum : MOVE ON

Stajlar : MOVE ON

İLGİ ALANLARI:

RC araçlarının tümü

Robotik Kodlama

YABANCI DİLLER:

İngilizce