**TEKNOFEST**

**HAVACILIK, UZAY VE TEKNOLOJİ FESTİVALİ**

**ROBOTAKSİ – BİNEK OTONOM ARAÇ YARIŞMASI**

**KRİTİK TASARIM RAPORU**

**(HAZIR ARAÇ KATEGORİSİ)**

**TAKIM ADI: K.İ.T.T.**

**TAKIM ID: 449020**

**İÇERİK**

[1.Takım Organizasyonu 3](#_Toc109228538)

[2.Ön Tasarım Raporu Değerlendirmesi 4](#_Toc109228539)

[3.Hazır Araç Özellikleri ve Analizi 4](#_Toc109228540)

[4.Araç Kontrol Ünitesi 5](#_Toc109228541)

[5.Araç Dinamiği Modelleme/Tanılama ve Kontrolü 6](#_Toc109228542)

[5.1. İleri Beslemeli Açı Kontrolü 6](#_Toc109228543)

[6.Otonom Sürüş Algoritmaları 7](#_Toc109228544)

[6.1. Şerit Takip Algoritması 7](#_Toc109228545)

[6.2. Levha ve Trafik Işıkları Tespit Algoritması 8](#_Toc109228546)

[6.3 Zed 2 Kamera Yardımıyla Mesafe Tespiti 9](#_Toc109228547)

[6.4 Park Algoritması 9](#_Toc109228548)

[7.Güvenlik Önlemleri 10](#_Toc109228549)

[8.Simülasyon 11](#_Toc109228550)

[8.1. Gazebo 11](#_Toc109228551)

[8.2. ROS 12](#_Toc109228552)

[9.Sistem Entegrasyonu 12](#_Toc109228553)

[9.1. CAN-BUS Haberleşme Arayüzü Kurulumu 13](#_Toc109228554)

[10.Test ve Doğrulama 14](#_Toc109228555)

[10.1. Can Bus Testi 15](#_Toc109228556)

[10.2.Trafik Işıkları ve Levha Testi 15](#_Toc109228557)

[10.3. Zed Kamera Testi 15](#_Toc109228558)

[10.4. Şerit Algılama Testi 15](#_Toc109228559)

[10.5. Acil Buton Testi 15](#_Toc109228560)

[11.Referanslar 16](#_Toc109228561)

# Takım Organizasyonu

Kalyon İleri Teknoloji Takımı (K.İ.T.T.) Robotaksi otonom hazır araç kategorisi yarışması kapsamında 2022 yılında kurulmuştur. 1 yüksek lisans, 8 lisans öğrencisi olmak üzere toplam 9 kişiden oluşmaktadır. Otonom araç takımı, Hasan Kalyoncu Üniversitesi Yazılım Mühendisliği Bölümü Araştırma Görevlisi Hatice Meltem danışmanlığında ve takım kaptanı ile 5 Yazılım Mühendisliği, 2 Bilgisayar Mühendisliği ve 1 Elektrik Elektronik Mühendisliği öğrencileriyle beraber çalışmalarına devam etmektedir. Takım üyelerinin geçmiş tecrübeleri ve bilgileri aşağıdaki gibi sıralanmaktadır.

**Akademik Danışmanı**

**Takım Kaptanı**

**Görüntü İşleme- Veri Seti Hazırlama**

**Otonom Sürüş Algoritmaları**

**Simülasyon / Test**

**Beyza ÖZKAN**

**Fahrettin Emin Korkut**

**Coşkun Emre Zevkirlioğlu**

**Bilal ÖZÇELİK**

**Kerem Emir Ercan**

**Enes Arslan**

**Melike Sultan BOZOĞLAN**

**Murat Süleyman Çapan**

Yarışmaya hazırlık sürecinde iki yazılım bir bilgisayar mühendisliği öğrencisi farklı hava koşullarında fotoğraflar çekerek veri seti oluşturup, eğitmişlerdir.

Bir yazılım bir bilgisayar mühendisliği öğrencisi ise süreç boyunca otonom sürüş algoritmaları odaklanmışlardır.

İki yazılım bir elektrik elektronik mühendisliği öğrencisi ise ön tasarım raporu hazırlık sürecinde algoritmaların simülasyon ortamında test edilmesinde çalışmışlardır.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | Proje Takvimi | Ocak | Şubat | Mart | Nisan | Mayıs | Haziran | Temmuz |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Rapor ve Simülasyon |  |  |  |  |  |  |  |
| 2 | Görüntü işleme ve veri seti oluşturma |  |  |  |  |  |  |  |
| 3 | Yazılım test ve iyileştirmeler |  |  |  |  |  |  |  |
| 4 | Test |  |  |  |  |  |  |  |

# Ön Tasarım Raporu Değerlendirmesi

Ön Tasarım Raporundan sonra algoritmalarımızı geliştirme sürecinde yaşanan bazı uyuşmazlıklar ve sorunlar sebebiyle ön tasarım raporunda trafik ışıkları ve levha okunmasında seçenek olarak sunulan open cv yerine nihai olarak yolo v3 e karar verilmiştir. Diğer yandan ön tasarım raporunda engelleri tespit etmek üzere lidar sensörünün kullanması planlanırken, test sürecindeki gelişmelerle zed kamerasıyla yapılmasına karar verilmiştir.

# Hazır Araç Özellikleri ve Analizi

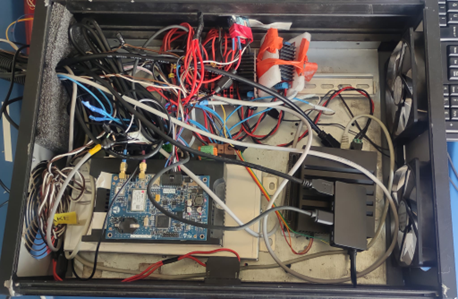
Yarışma süresince kullanaılabilmesi için teknofest tarafından elektrikli araç sağlanmaktadır. Aracın motoru 7,4 KWgücünde olup çalışma gerilimi 48 volttur. Araç 25 Km/h maximum hız ile gitmektedir. Aracın fren sisteminde park freni mekanik servis freni hidroliktir. Araç ölçüleri ise toplam uzunluğu 2825, toplam yükseklik 1950 ve toplam genişlik 1330 santimetredir. Araçda sağ ve sol sinyal ışıkları bulunmaktadır. Bu ışıkların yansıma yüzeyinin çapı 7cm ve ışık miktarı 500 lümendir. Araçta elektronik direksiyon (by-wire) olarak adlandırılan otonom direksiyon sistemi bulunmaktadır ayrıca xavier, canbus, çevirici gibi elektronik kartları içeren araç kontrol ünitesi bulunmaktadır. Araç üzerinde bir adet Velodyne Lidar, bir adet ZED 2 Stereo Kamera bulunmaktadır. Zed kamera mekansal algılamayı iyileştirmek için tasarlanmıştır. Sensörlerinden yazılımına kadar son teknoloji ile donatılmıştır. Kameranın 120 derecelik geniş açılı görüş alanı sunmaktadır, derinlik algılama özelliğine sahiptir, Dahili IMU Barometre ve Manyometre sayesinde gerçek zamanlı eylemsizlik yükseklik ve manyetik alan verilerine ulaşabilme imkanı sunmaktadır, bulut bağlantısı sayesinde uzaktan erişim imkânı vermektedir ve Zed Wrapper paketi ile ROS ortamında kullanılabilmektedir.



Resim 1: Lidar ve kamera

# Araç Kontrol Ünitesi

Otonom araç kontrol ünitesi aracın otonom gidebilmesi için gerekli olan elektronik kartlardan (xavier, canbus, çevirici vb.) oluşmaktadır. Araçta 2 tane acil butonu bulunmaktadır. Araç uzaktan kumanda ile çalışabilmektedir. Araçta tüm alt sistemleri biribirne bağlamak çok zor olacağı için merkezi ağ sistemi kullanılmaktadır. Bu sistem, sistemleri toplar dolaşır ve haberleşmeyi sağlar. “The Controller Area Network” (CAN) ile kablo ağırlığı minimuma indirilmektedir ayrıca iletişimde hata olasılığı en az orandadır. Bu iletişim “python-can” kütüphanesi ile sağlanmaktadır. CanBus mesaj öncelikli bir haberleşme sistemidir. CanBus sistemi tek kablo üzerinden eşit erişimli mesaj yollanmasına izin vermektedir. Araç çalışmaları için geliştirilmiş Nvidia Xavier AGX 32 GB geliştirme kartı bulunmaktadır[1].

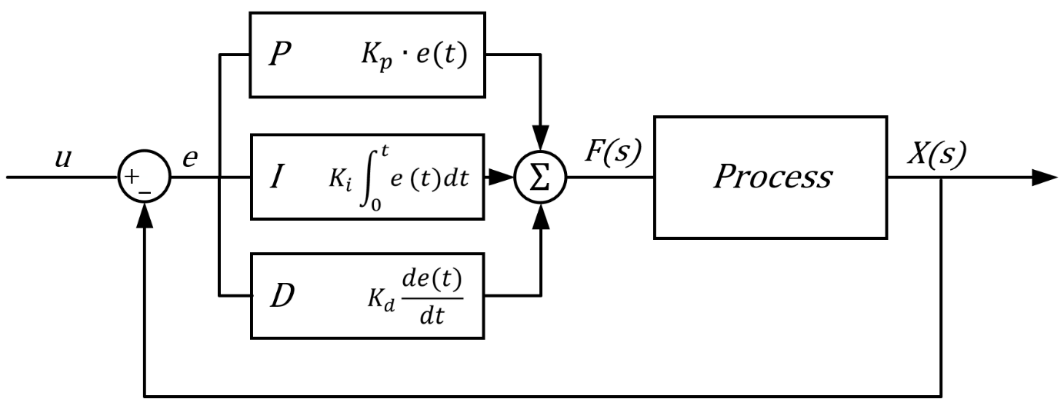


Resim 2: Araç Kontrol Ünitesi

# Araç Dinamiği Modelleme/Tanılama ve Kontrolü

## **5.1. İleri Beslemeli Açı Kontrolü**

Şerit algoritmasından elde edilen verilere göre aracın dönüş açısının kontrolü için PID kontrolü kullanılmaktadır. Endüstriyel kontrol sistemlerinde en çok kullanılan kontrollerden olan PID kontrolü, geri besleme yoluyla hata farkı (sistemin mevcut durumu ile istenen durumu arasındaki fark) na göre çıkışa bir sinyal üreterek ilgili sistemlerin kontrolünü sağlayan bir kontrol yöntemidir. Proportional Integral Derivative yani Oransal İntegral Türevsel açılımı olan PID, P(Oransal), I (İntegral) ve D (Türevsel) olmak üzere 3 tane farklı parametreye sahiptir. Bu parametrelerden P (mevcut hata), I (geçmiş hataların toplamı), D (gelecekteki hataların bir tahminini) ifade eder. PID kontrol yönteminin zaman domainindeki matematiksel ifade ve blok diyagramı aşağıda mevcuttur.



Burada Kp, Kd ve Ki olarak gösterilen katsayılar P, I ve D parametrelerin katsayılarıdır. Bu katsayıları farklı algoritmaları kullanılarak en iyi sonuç elde edilebildiği gibi farklı denemelerin sonucunda da elde edilebilmektedir. Farklı denemelerde Kp = 1.0 / 835.0, Ki = 1.0 / 600.0, Kd = 1.0 / 1000.0, değerleriyle en iyisonuçlar elde edilmiştir.

Burada istenilen değer araç haberleşme arayüzü CAN ile araca gönderilirken aracın mevcut

dönüş açısını ZED 2 kamerada yer alan IMU sensörünün (Inertial Measurement Units -

İnersiyal Ölçme Ünitesi) verilerinden elde etmekteyiz. ZED 2 Kamerasının IMU sensöründen anlık olarak (“Parent” – Araç) açısal hız, açısal ivme ve oryantasyonu gibi bilgiler sağlanmaktadır.

PID Hesaplanması sözde kodu aşağıdaki şekildedir.

errorSum = Error + errorSum; # Baştan şu ana kadarki hataların toplamı alındı

errorDif = errorPre - Error; # Bir önceki ve şimdiki hataların farkı alındı

errorPre = Error; # Bir önceki hata güncellendi

P = Kp\*Error; # Oransal Kontrol Sinyali hesaplandı

I = Ki\*errorSum\*dt; # Integral Kontrol Sinyali hesaplandı

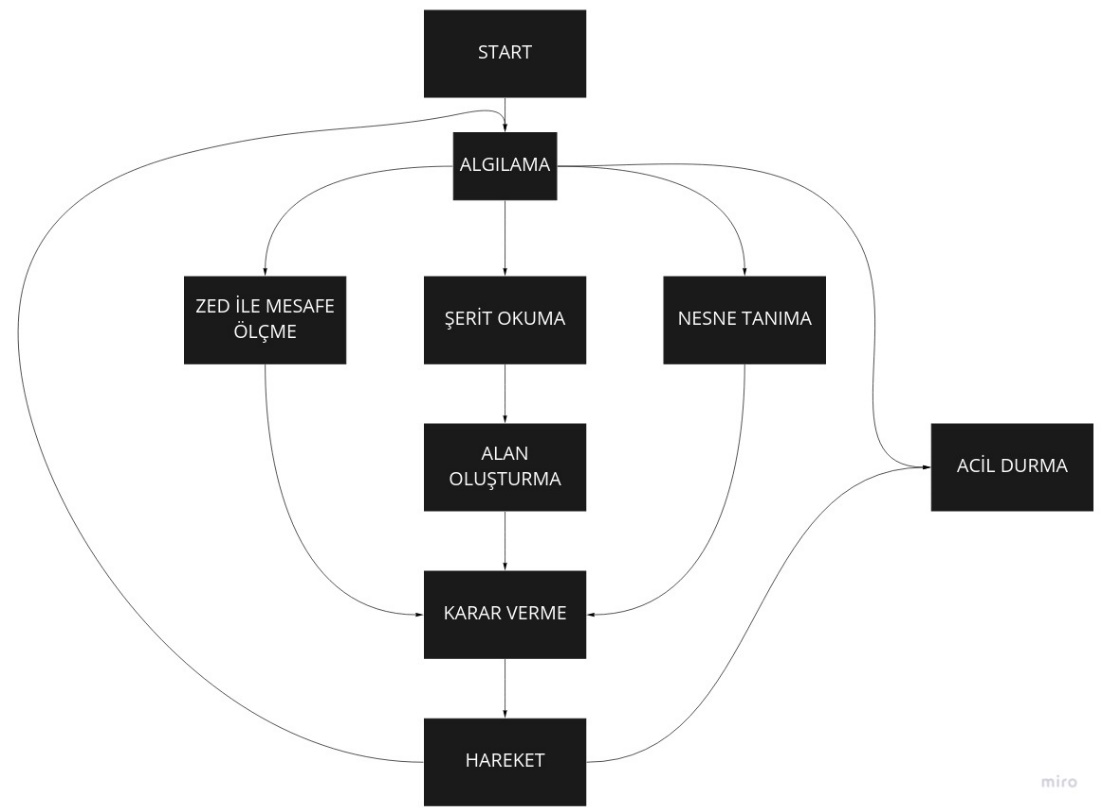
D = Kd\*errorDif/dt; #Türev Kontrol Sinyali hesaplandı

PID = P + I + D; # O,I ve T kontrol sinyallerinin toplamıyla

# PID Kontrol sinyalini elde edildi

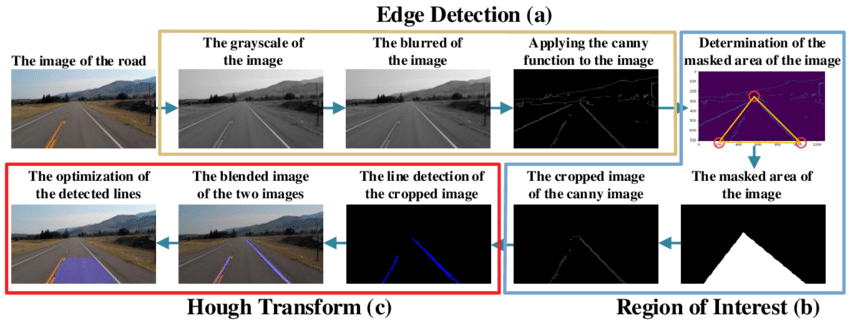
# Otonom Sürüş Algoritmaları

Bu bölümde aracın yarışma platformunu tamamlaması için geliştirelen otonom sürüş algoritmalardan bahsedilmiştir. Algoritmamızın akışı aşağıdaki gibidir**.**



## **6.1. Şerit Takip Algoritması**

Şerit takip[2] Ön Tasarım Raporunda da belirtildiği üzere open cv ile hough dönüşümü kullanılarak yapılmıştır. Hough Dönüşümü, görüntülerdeki doğru ve daireleri tespit etmeyi kolaylaştıran, bilgisayarda görme ve görüntü işleme alanlarında kullanılan algoritmalar bütünüdür[3]. Adımları resim 3 gibidir.



Resim 3 [4]

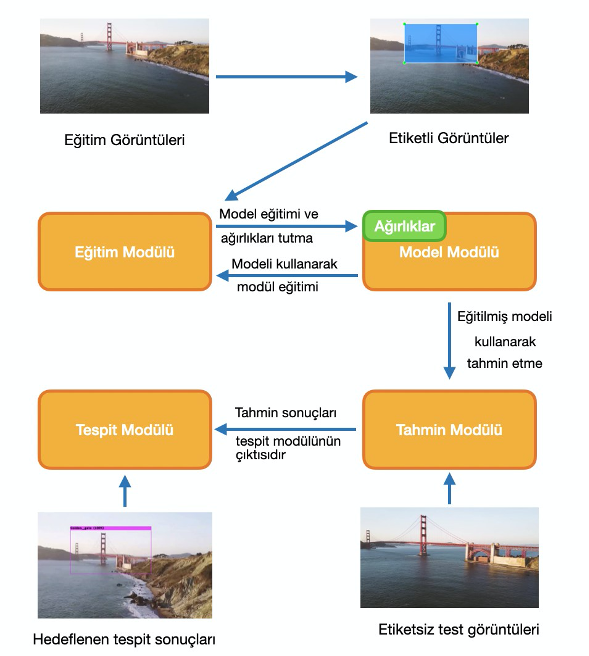
## **6.2. Levha ve Trafik Işıkları Tespit Algoritması**

Görüntü işleme dijital bir görüntü içerisindeki önemli bilgilerin okunması, çıkartılması ve işlenmesi için kullanılan bir yöntemdir. Görüntü içerisinde bulunan bir nesne ya da bir ortam hakkında insan görme sistemine benzer şekilde nitel bilgiler edinilmesi ve kullanılması görüntü işlemenin temel amaçlarındandır. Görüntülerde bulunan nesnelerin tespiti, tanımlanması, sınıflandırılması ve takibi gibi ihtiyaçları karşılayacak birçok yöntem geliştirilmiştir[5]. Biz de bu süreçte Ön tasarım raporunda da belirttiğimiz üzere levha ve trafik ışıklarının tespitinde yolo v3 tercih edilmiştir. Bu süreçte resim .. gibi çeşitli yerlerden, farklı zaman dilimlerinde toplanan levha ve trafik görsellerinden labellmg ile etiketleme yapılıp veri seti oluşturulmuş ve model eğitilmiştir.



Resim 4

Yolo’ nun çalışma akışı ve bizim izlediğimiz yolda resim 4 gibidir.



Resim 5

## **6.3 Zed 2 Kamera Yardımıyla Mesafe Tespiti**

Ön tasarım raporunda engelleri aşma noktasında lidar ile mesafe verisi almayı planlamamıza rağmen test aşamasında bunun zed 2 stereo kamerasıyla sağlanmasına karar verilmiştir. Aracımızda bulunan Zed 2 stereo kameradan aldığımız mesafe bilgisi doğrultusunda aracımız herhangi bir engelle karşılaşması durumunda olası problemleri engellemek için yavaşlayarak yeni rota çizmektedir.

## **6.4 Park Algoritması**

Park algoritması için öncelikle araç park edilebilir tabelasını algılayana kadar open cv ile güvenli sürüş yapmaktadır. Araç ve park levhası arasındaki açının bulunabilmesi için ZED2 özellikleri ile birlikte çeşitli hesaplamalar ve dönüşümler yapılarak kameranın levhaları görme açısı çıkarılmıştır. Elde edilen hesaplamalar sonucunda görüş açısı kullanılarak araca tekerlek açısı verilmiştir. Aynı zamanda araç ve levhalar arası mesafe ZED kameranın uzaklık hesaplama fonksiyonu ile belli bir mesafeden (ort 1m) sonra araç durmaktadır.

# Güvenlik Önlemleri

Can ve mal güvenliğini tehlikeye atmayacak güvenlik önlemleri alınmalıdır. Acil enerji kesme anahtarı, gerektiğinde dışarıdan da kolayca erişilebilecek bir yerde olmalıdır. Bu nedenle hem aracın ön kısmında hemde aracın arka kısmında buton bulunmaktadır. Ön kısımda bulunan buton tüm sistemi durdururken arka taraftaki buton ise elektronik sistemleri durdurur. Kablolar kesinlikle uygun kablo kılıfı içerisinde bulunmaktadır. Kablo demetleri uygun şekilde kelepçeli bulunmaktadır.



Resim 6



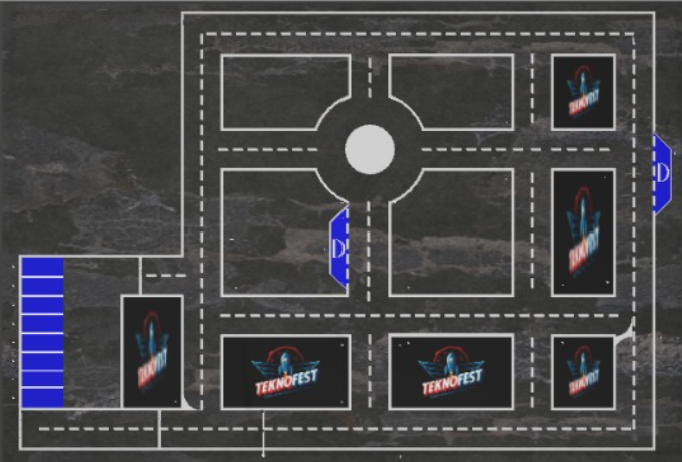
Resim 7

Derinlik kamerasından alınan bilgiler incelendiğinde aracın önünde belirli mesafeden yakın bir nesne tespit edebilmektedir.

Ayrıca aracımızda kullandığımız Canbus haberleşme protokolü sayesinde öncelikli mesajların daha önce iletilmesi ile iletişim çakışmalarını önlemektedir. Bu sayede bir sorun anında o an için önemsiz bilgiler yerine daha fazla önem arz eden bilgileri iletim hattına göndermektedir. Bu da olası hataları büyük oranda azaltmaktadır.

# Simülasyon

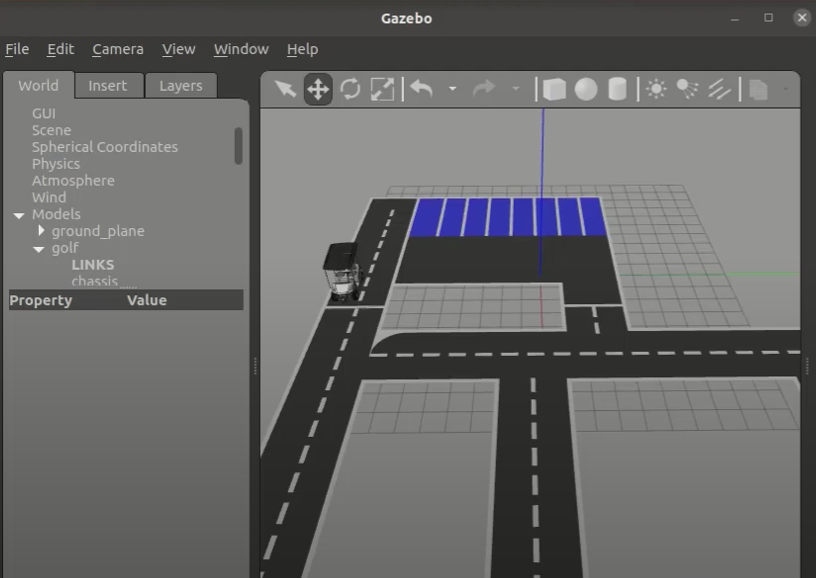
Takımımız işletim sistemi olarak Ubuntu, haritalama için Blender ve AutoCAD kullanmıştır. TEKNOFEST tarafından belirlenen ve şartnamede yazan şekilde parkur ve şeritler ayarlanmıştır. Haritanın araç ile entegrasyonu ve algoritmaları kullanmak için Gazebo ortamında geliştirilmiştir. YOLOv3 (You Only Look Once) algoritmasının nesne tespitini oldukça hızlı bir şekilde ve tek seferde yapabilmektedir bunun içinde simülasyon sırasında trafik levhaları/ışıkları için YOLOv3 nesne tanıma algoritması kullanılmıştıri. OpenCV ile şerit takibi yapılmıştır.



Resim 8

## **8.1. Gazebo**

Gazebo [6], temelinde istemci sunucu ilişkisiyle çalışmaktadır. Sunucu(gzclient) kullanıcının etkileşimini ve simülasyonunun görselleştirilmesi sağlar.



Resim 9

## **8.2. ROS**

Ros (Robot İşletim Sistemi) [7], robot yazılımı için esnektir çok çeşitli robotic platformlarda kompleks robot davranışını yaratma görevini basitleştirmektedir.

# Sistem Entegrasyonu

Otonom aracın hareketi 4 aşamadan oluşur:

1. Çevreyi algılama: Bu aşamada, sensörlerden gelen bilgiler sayesinde GPS yardımı ile konumu belirler. Engel, tabela, ışık ve yol tespit edilerek araç etrafını tanımaktadır.

2. Aracın çevredeki yerinin tespiti: Çevrede algılanan nesneler GPS yardımı ile algılanmaktadır.

3. Yol planlaması: Önceki aşamalardan gelen verilerin kullanılarak aracın rotasının tespit edilmesi sağlanmaktadır.

4. Kontrol: Son aşamada rotanın belirlenmesi ile donanıma komutlar verilmesi ile aracın kontrolu sağlanmaktadır.



Resim 10

metin, elektronik eşyalar içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resim 11

## **9.1. CAN-BUS Haberleşme Arayüzü Kurulumu**

Aracın gazını ve frenini kontrol etmek için gereken 0-10V arasındaki voltaj değerleri, kontrol ünitesi Üzerindeki PWM kanalları ile kontrol edilir. PWM kanallarını kullanma CAN-BUS haberleşme formatında olduğu gibi pinmux değişikliklerinin yapılmalıdır. Öncelikle python üzerinden kurulumu gerçekleştirebilmek için gerekli olan python-can ve busybox kütüphanesi kurulmalıdır. Sonrasında sistemin CAN kontrolörlerinin hangi pinlere karşılık geldiğini yapılandırılır. Bu adımlar ve gerekli tanımlamalar için aşağıdaki adımlar takip edilmelidir.

\* sudo ip link set can1 type can bitrate 500000 \

\* dbitrate 2000000 berr-reporting on fd on

• Şimdisadece CAN arayüzlerinin açılması gerekmektedir:

\* sudo ip link set up can0

\* sudo ip link set up can1

• Yapılan bu can yapılandırmalarının sistem başladığında otomatik yapılması için

enable\_CAN.sh isimli dosya oluşturup içerisine yukarıdaki kodlar eklenerek kök dizine

eklenebilir ve bu sayede sistem her açıldığında otomatik olarak yapılandırma yapılmış

olur.

\* printf '%s\n' '#!/bin/bash' 'exit 0' | sudo tee -a /etc/rc.local

\* sudo chmod +x /etc/rc.local

• Bu işlemler sonunda enable\_CAN.sh dosyası sistem başlangıcında otomatik olarak

çalıştırılır ve can denetleyicilerini yapılandırır.

- img : Giriş görüntüsünün boyutunun verildiği parametredir. Orijinal giriş görüntüsünü sıkıştırarak eğitim sürecini hızlandırmayı amaçlar.

– batch: Binlerce görüntüyü aynı anda sinir ağına iletmek, modelin bir çağda öğrendiği ağırlık sayısının çok artmasına neden olur. Bu nedenle, veri kümesi genellikle birden çok n görüntü grubuna ve toplu iş birimine bölünür.

- epochs: öğrenme algoritmasının tüm eğitim veri kümesi boyunca geçtiği zamanı tanımlayan parametredir.

- data : Veri kümesinin içeriğindeki sınıfların listesini içeren yaml uzantılı dosyanın yolunu belirtir.

- weights: Önceden eğitilmiş bir ağırlık kullanmak eğitim süresini kısaltacaktır. Bu bölüm boş bırakılırsa, model eğitim için otomatik olarak rastgele ağırlıklar başlatır.

Görüntüleri önbelleğe almak, eğitim hızını artırır. Farklı dönemler için eğitim süresi ve veri kümesindeki trafik işaretlerini algılayan eğitim kümeleri çok uzun sürüyor. Bundan önce, verilerin yüklenmesi ve eğitim için gerekli formata dönüştürülmesi ciddi zaman alan bir süreçtir.

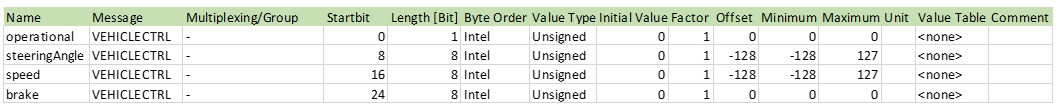
Proje sonunda eğitim ve testlerin net bir şekilde analiz edilebilmesi için çalışmada Tensorboard grafik eklentisi kullanılarak modellerin performansı görselleştirilmiştir. Şekilde belirtilen dönem sayısı ve eğitim sonrası performansları verilmiştir. Sonuç olarak, modeller test için hazır hale getirilir. Modellerin Eğitimlerinin Performans Değerlendirmesi ve Test Sonuçları Google Colab'da eğitimlerini tamamladıktan sonra test edilen modeller TensorBoard ile grafiğe dönüştürüldü. TensorBoard, makine öğrenimi denemeleri için gereken görselleştirmeyi ve araçları sağlayan bir araçtır. Kayıp ve doğruluk gibi metrikleri izlemek ve model grafikleri oluşturmak için yaygın olarak kullanılır.

# Test ve Doğrulama

TEKNOFEST tarafından verilen araç için yapılan algoritmalar öncelikle büyük oranda simülasyon ortamında test edilmiştir. Daha sonrasında Teknofest tarafından verilen imkanlar doğrultusunda hazır araç üzerinde testler yapılmıştır. Yapılan testlerde geliştirilen algoritmaların tepkileri ölçülmüş olup algoritmalardaki hatalar giderilmeye çalışımıştır ve gerekli olan optimizasyon işlemleri yapılmıştır. Bu doğrultuda gerçekleştirilen test ve doğrulama aşamaları şu şekildedir:

## **10.1. Can Bus Testi**

Aracın Can Bus protokolüyle haberleşerek tekerlekleriyle iletişim sağlayıp sağlanmadığı test edilmiş olup hız, tekerlek açısı, fren gibi aracın hareketine etki eden faktörler bu protokol üzerinden test edilmiştir.Aracı kontrol ettirebilmek için aşağıda tabloya göre Can mesajı gönderilmektedir.Baudrate 250 kbps olarak ayarlanmaktadır.



Tablo 1

## **10.2.Trafik Işıkları ve Levha Testi**

Trafik ışıkları ve levha tespit edebilmek için parkurdan ve çeşitli yerlerden alınan görüntülerden veri seti oluşturulmuş, Yolo v3 ile model eğitilmiştir. Yapılan testlerden sonra büyük oranda başarı sağlanmıştır.

## **10.3. Zed Kamera Testi**

Teknofest tarafından sağlanan araç testi aşamasında, araç üzerinde kullanılan kamera sensörlerinden otonom sürüşte kullanılacak olan algoritmalar için Point Cloud verisi, derinlik verisi gibi verilerin alınması test edilmiştir.

## **10.4. Şerit Algılama Testi**

Teknofest tarafından sağlanan test sürecinde şerit takip algoritması open cv ile hough dönüşümü kullanılarak yapılmıştır, ve büyük oranda başarı sağlandığı gözlemlenmiştir.

## **10.5. Acil Buton Testi**

Araç hareket halindeyken kazaya sebebiyet verebilecek bir nesnenin önüne çıkması durumunda devreye girecek olan acil fren güvenlik sistemi TEKNOFEST tarafından sağlanan test sürecinde test edilmiştir.

**Simülasyon video linki**

https://www.youtube.com/watch?v=IXTpUK0F4kI&ab\_channel=COSKUNEMREZEVKIRLIOGLU

# Referanslar

* [1] <https://developer.nvidia.com/>
* [2] <https://www.youtube.com/watch?v=KEYzUP7-kkU&t=82s&ab_channel=Pysource>
* [3] <https://tr.wikipedia.org/wiki/Hough_d%C3%B6n%C3%BC%C5%9F%C3%BCm%C3%BC>
* [4] <https://www.researchgate.net/figure/The-phases-of-the-lane-lines-detection-and-tracking-in-the-road-image-a-Edge_fig1_360841512>
* [5] <https://dergipark.org.tr/tr/download/article-file/384735>
* [6] <http://yapbenzet.kocaeli.edu.tr/> yy
* [7] <https://www.ros.org/> zz