Gebze Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü

CSE 396 –Bilgisayar Mühendisliği Projesi
Final Raporu
Proje Grubu 5

PARKING SIMULATOR

Grup Üyeleri

Ozan GEÇKİN

Ömer Faruk BİTİKÇİOĞLU

Hatice ARGUN

Kivanç TÜRKER

Hüsnü AKÇAK

Mustafa Alp

İrfan KARATEKİN

Ece ERER

Berat ÖZDİN

Gonca Ezgi ÇAKIR

Murat BEYAZ

Alper Yaşar

Furkan KALABALIK

Ahmet Yusuf ERDOĞAN

Haziran 28, 2021

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

1. GİRİŞ

1.1. PROJENİN TANIMI VE AMACI

Projenin geliştirilmesindeki temel hedef kullanıcının paralel park üzerine deneyim kazanmasını sağlamaktır. Ülkemizde direksiyon sınavlarında temel kriterlerden biri olan paralel park aşamasında çoğu yeni sürücü adayı sıkıntı yaşamaktadır. Bu projeyle kullanıcıya gerçeğe yakın bir ortam sunularak paralel park konusunda tecrübenin arttırılması ve yaşanan sıkıntının minimuma indirilmesi amaçlanmıştır. Bu doğrultuda projenin donanım tarafında aracın temel fonskiyonlarının gerçeklenebilmesi için direksiyon ve gaz - fren pedallarına yer verilmiştir. Projeyi görsel olarak desteklemek adına simülasyon ortamı geliştirilmiş, çeşitli park alanları tasarlanmıştır. Son aşamadaysa kulllanıcı kullanıımı kolaylaştırmak için harici bellek üzerine program işlenerek tek aşamada simülasyon ortamına erişim sağlanmıştır.

- Projeye ait web sitesine ulaşmak için *tıklayınız.*
- Projeye ait youtube videosuna ulaşmak için *tıklayınız*.

1.2. MODÜLLER VE ÜYE DAĞILIMI

Proje 5 farklı modüle ayrılmıştır, aşağıda her modülde görev alan ekip üyelerini listelenmiştir.

TABLO 1-1 Modüller ve Ekip Üyeleri Dağılımı

Board Bilgisayar haberleşme	Tasarım	Simülasyon	GUI	Container/Driver
Mustafa Alp	Ozan GEÇKİN	Kıvanç TÜRKER	Gonca Ezgi Çakır	Mustafa Alp
Murat BEYAZ	Yusuf Erdoğan	Gonca Ezgi Çakır	Kıvanç TÜRKER	Murat BEYAZ
Furkan Kalabalık	Furkan Kalabalık	Mustafa Alp	Ece Erer	İrfan Karatekin
Berat Özdin	Murat BEYAZ	Ece Erer	İrfan Karatekin	Ömer Faruk Bitikçioğlu
Hüsnü AKÇAK		Alper Yaşar	Alper Yaşar	Berat Özdin
Ozan Geckin		Hatice Argun	Hatice Argun	Hüsnü AKÇAK
		Ömer Faruk Bitikçioğlu	Yusuf Erdoğan	
		İrfan Karatekin		

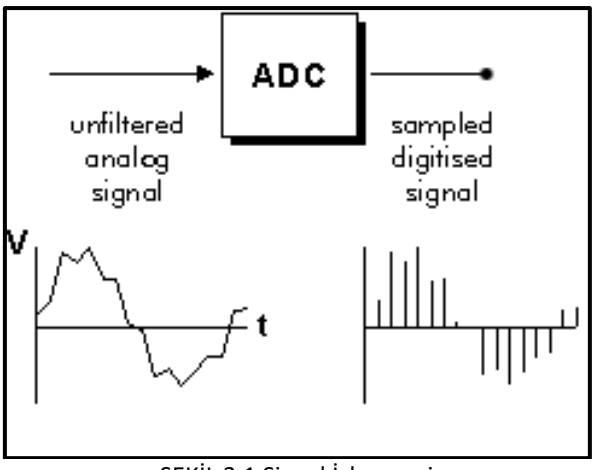
2. GELİŞTİRMELER

2.1. BOARD VE BİLGİSAYAR HABERLEŞMESİ

Projede, donanım kısmı temel olarak üç adet potansiyometre üzerinden okunan voltaj değerlerinin ADC portları aracılığı ile dijital dataya dönüştürülmesi ve bu dönüştürülen değerlerin, dönme açısı veya pedal basınç değerleri olarak yorumlanıp, sistemin çalıştığı bilgisayar üzerine USB aracılığı ile iletilmesinden oluşmaktadır.

Cihaz, işleme gücünü STM32F103C8T6 demo board üzerinden almaktadır. Board üzerinde A7, B0 ve B1 pinler üç adet ADC pini olarak ayarlanmıştır. ADC, açılımı Anolog-Digital Converter olmak üzere analog bir veriyi dijital bir veriye, yani bitlere dönüşütüren bir donanımdır. Bir adet referans analog giriş üzerinden, veriyi digital bir veriye dönüştürürler. Kesin bir, dönüşüm yapamamakla birlikte, iyi sonuçlar elde edilebilmektedir.

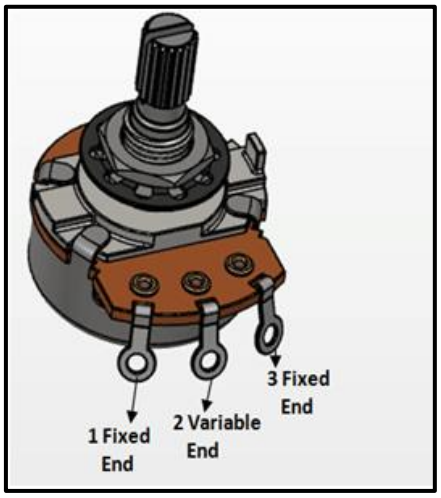
Çok farklı ADC donanımları bulunsa da, projede STM32 üzerindeki ADC girişleri kullanılmıştır. ADC girişleri, 28.5 Cycle/sample yapacak şekilde ayarlanmıştır. ADC üzerinde okuma yapılırken, STM32 bize farklı modlar sunmaktadır. Polling, Interrupt ve DMA gibi modlar bulunmaktadır. Biz burada iki farklı ADC kanalı üzerinden okuma yaptığımızdan dolayı, DMA ile okumayı tercih ettik. DMA ise, açımılı Direct Memory Access olmak üzere, interrupt ile gelen datayı, belirtilen hafıza üzerine işlemciyi meşgul etmeden, bus üzerinden direkt olarak yazan bir sistemdir. Bu sayede, okunan ve dönüştürülen digital data, sürekli bir şekilde belirtilen hafıza bölümüne yazılmış olacak ve her zaman taze bir data ile işlem yapılmış olacaktır. ADC ile okunan veri0-4096 arası değerler sunmaktadır. Bu değerlerin, yön ve büyüklük olarak tayin edilmesi ile sistem işlevsellik kazanmaktadır.



ŞEKİL 2.1 Sinyal İşlenmesi

Potansiyometre, bir ucu referans voltajına bağlı, diğer ucu ise değişken voltaj değerini output olarak veren ayarlanabilir bir dirençtir. Verilen voltaj değeri, ile alınan çıktı STM32 üzerinde dönüştürülerek, değişken değerler elde edilmiştir. Sistemimiz örneğin bir direksiyonu, düşünürsek 0 ile 4096 arasındaki değerleri ele alırsak, 0-2048 arası aslında sol tarafa dönüşü, 2048 ile 4096 arası ise sol tarafa dönüşü belirtmektedir. Sol tarafta 0'a ne kadar yakın olunursa, teker sol yöne o kadar kırılmış, sağ tarafta ise 4096 tarafına ne kadar yakınsa teker sağ tarafa o kadar kırılmış demektir. Aynı şekilde, pedallar

üzerinde 0'dan başlamak üzere, potansiyometre ne kadar oynatılır ise, gaza veya frene o kadar basılmış demektir. Bu basit mantık ile, gerekli offset değerleri sağlanarak hem yön hem de pedallara basınç değerleri tayin edilmiş olup sistem donanımsal olarak, işlevsel hale getirilmiştir.

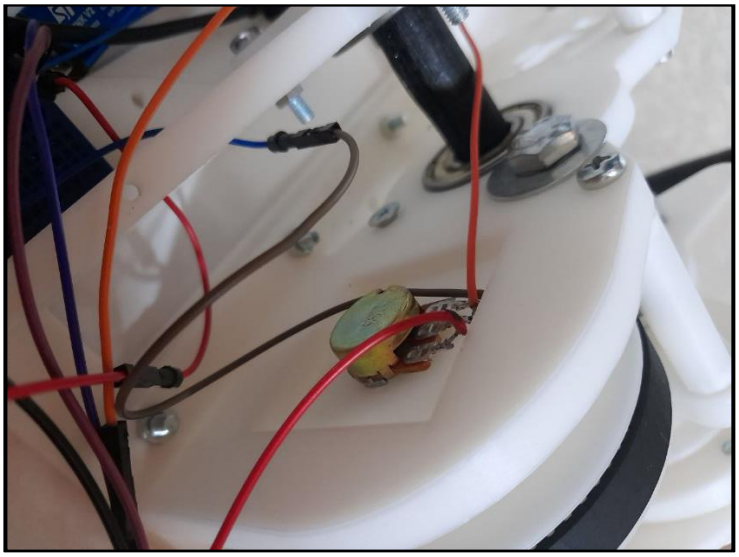


ŞEKİL 2.2 Potansiyometre

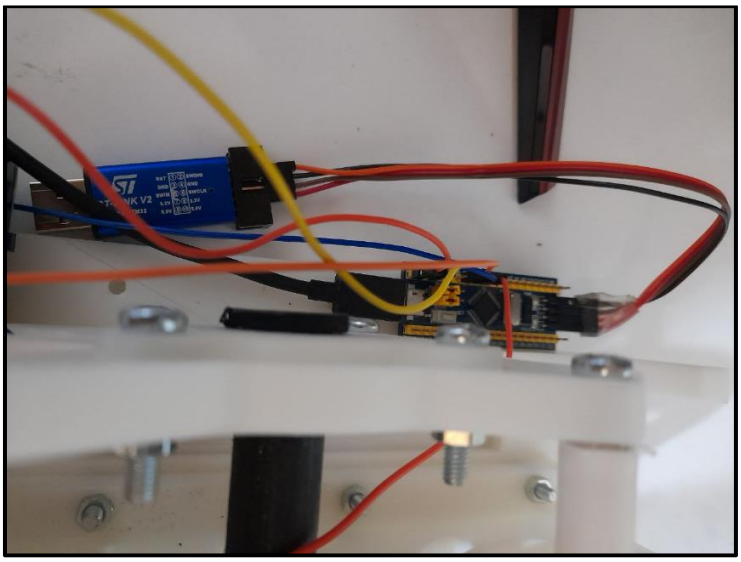
Sistemdeki veriler alınıp, yorumlanabilir hale getirildikten sonra, bilgisayar üzerindeki simülasyonu yönetmek için gerekli yapılar oluşturuldu. Burada oluşturulan yapı, generic bir yapı olup, sadece bu simülasyonu değil, kalibre edildikten sonra, tüm oyunları yönetecek bir yapıdır. Sistemde kullanılan yapı USB protokolünün bir bölümü olan HID (Human Interface Device)'dan oluşmaktadır. USB HID ile bilgisayara dışardan takılan cihazlar, kendini tanımlayan descriptorları aracılığı ile kendini bilgisayara tanıtır ve bilgisayar bu sayede cihazın, kendisini yönetmesini sağlar. Bu raporlar, host cihaza kendisinin yollladığı verileri nasıl yorumlaması gerektiğini söyler ve yönetimi kendisi sağlar. Bu sistemin güzelliği, tüm bilgisayar sistemlerinin, HID driver'larının bulunmasıdır. Bu sayede cihazdan bağımsız bir şekilde kullanım imkânı vardır. Ayarlanan cihazı, herhangi bir mouse veya klavye gibi tak çalıştır şekilde kullanma imkânı sağlar. Bunun için STM32'yi bir HID olarak ayarlamak gerekmekteydi. Ayrıca sistemin ayarlanan cihazı bir joystick olarak algılayıp oyunlar üzerinde kullanım imkânı sağlaması gerekmekteydi. Bunun için, cihaza, simülasyon gereksinimlerine uygun bir şekilde bir HID descriptor oluşturulması gerekmekte.

```
char ReportDescriptor[34] = {
    0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
    0x09, 0x05,                    // USAGE (Game Pad)
    0xa1, 0x01,                    // COLLECTION (Application)
    0xa1, 0x00,                    //   COLLECTION (Physical)
    0x05, 0x01,                    //     USAGE_PAGE (Generic Desktop)
    0x09, 0x30,                    //     USAGE (X)
    0x09, 0x31,                    //     USAGE (Y)
    0x09, 0x32,                    //     USAGE (Z)
    0x09, 0x33,                    //     USAGE (Rx)
    0x09, 0x34,                    //     USAGE (Ry)
    0x09, 0x35,                    //     USAGE (Rz)
    0x15, 0x81,                    //     LOGICAL_MINIMUM (-127)
    0x25, 0x7f,                    //     LOGICAL_MAXIMUM (127)
    0x75, 0x08,                    //     REPORT_SIZE (8)
    0x95, 0x06,                    //     REPORT_COUNT (6)
    0x81, 0x02,                    //     INPUT (Data,Var,Abs)
    0xc0,                          //     END_COLLECTION
    0xc0                           // END_COLLECTION
};
```
ŞEKİL 2.3 Descriptor Sistem

ŞEKİL 2.3' teki descriptor sistem üzerinde kullanılan descriptordur. Burada ilk olarak, desktop bir cihazda kullanacağımızı belirtiyoruz. Daha sonra cihazın bir gamepad olduğunu, tek bir application üzeride fiziksel tuşlar olduğunu söylüyoruz. Daha sonra tekrar Desktop için olduğunu belirtip, kaç farklı axis'te hareket desteklediğimizi belirtiyoruz. Bu axisler için hangi aralıkta değerler yollayacağımızı ve bunları nasıl değerlendirmesi gerektiğini söyleyerek, descriptor'u bitiriyoruz. Burada 6 adet axis değeri destekliyoruz. Her bir axis değeri 8 bit veri yolladığı için REPORT_SIZE 8 olarak ayarlanmıştır. Bu 8 bit –127 ile 127 arası değerleri desteklemektedir. Daha sonra, bu axis değerlerini karşılayan bir struct üzerinden, HID report bilgisayar tarafına gönderildiğinde, sistem tıpkı bir joystick ile yönetiliyormuş gibi, simülasyonu hareket ettirmeye başlamaktadır. Bu sayede raw olarak verilen değerler, tıpkı bir dönüş ve gaz hissiyatı vererek, sistemi daha gerçekçi hale getirmiştir. Digital değerler 0-4096 arası gelmektedir. Bu değerler 0-127 arasına indirgenerek bilgisayar gönderilmiş ve hareket kabiliyeti sağlanmıştır. Sistem generic bir şekilde, takıldığı bilgisayar üzerinde joystick olarak tanımlanmıştır. Proje üzerinde axis değerleri ayarlanarak kalibrasyon yapılmış, çalışır hale getirilmiştir.



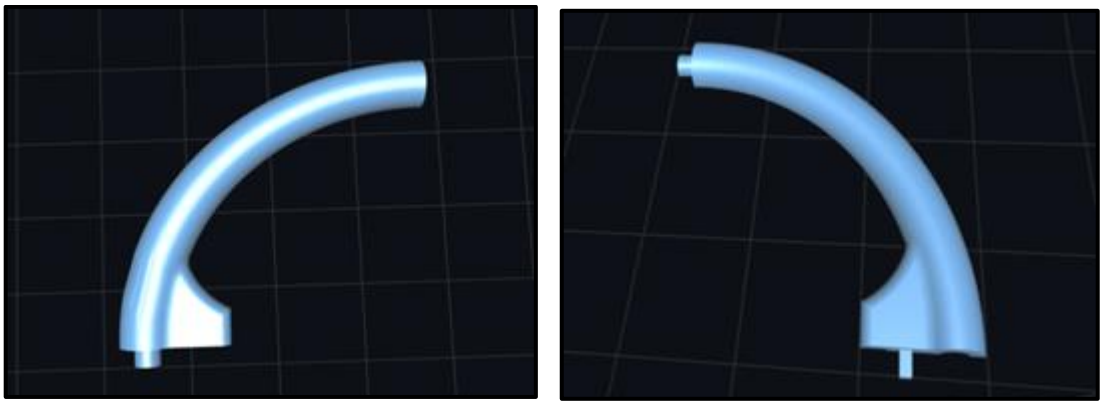
ŞEKİL 2.4 Donanım Bağlantı 1



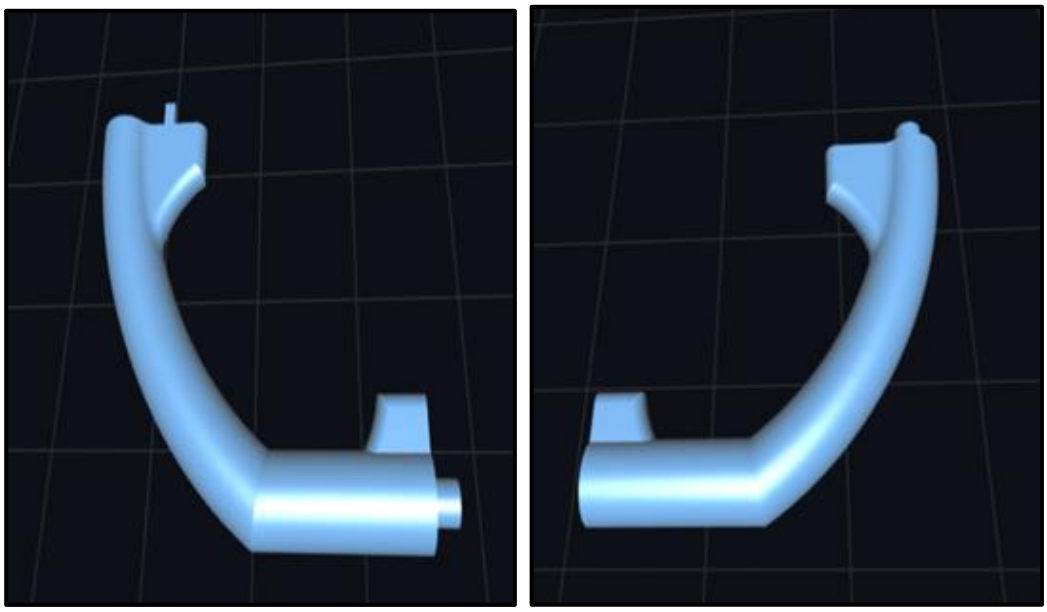
ŞEKİL 2.5 Donanım Bağlantı 2

2.2. TASARIM

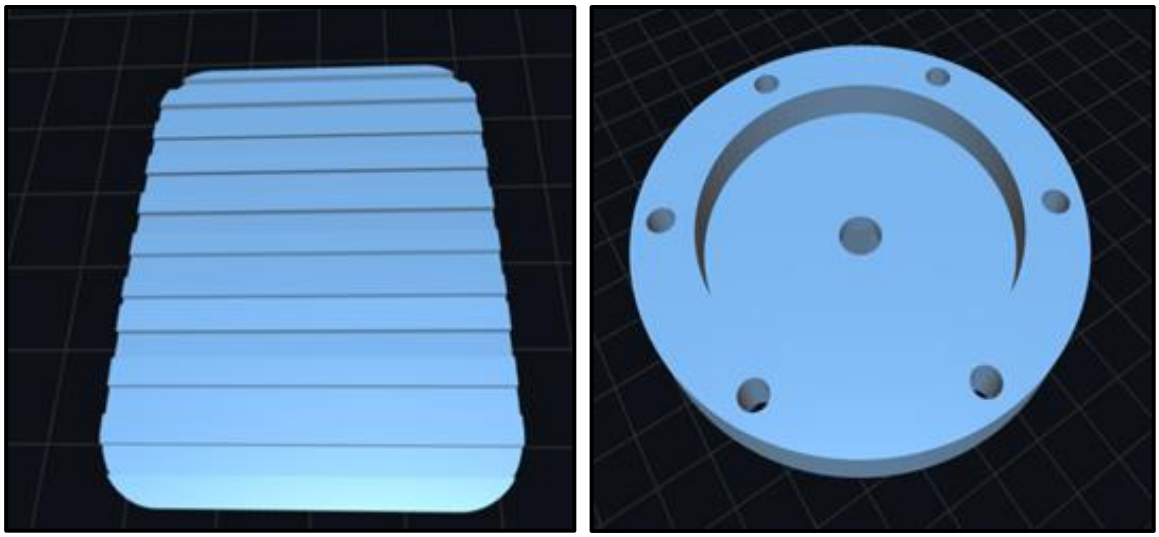
Simülatörün fiziksel tasarımı kısmında direksiyon, pedal, piston ve bağlantı noktaları gibi 36 adet küçük ve büyük parçanın tasarımı yapıldı. Parçaların 3D printerda basımına başlandı. Basımlar tamamlandıktan sonra kurulum ve potansiyometreler ile entegrasyonu yapılacaktır.



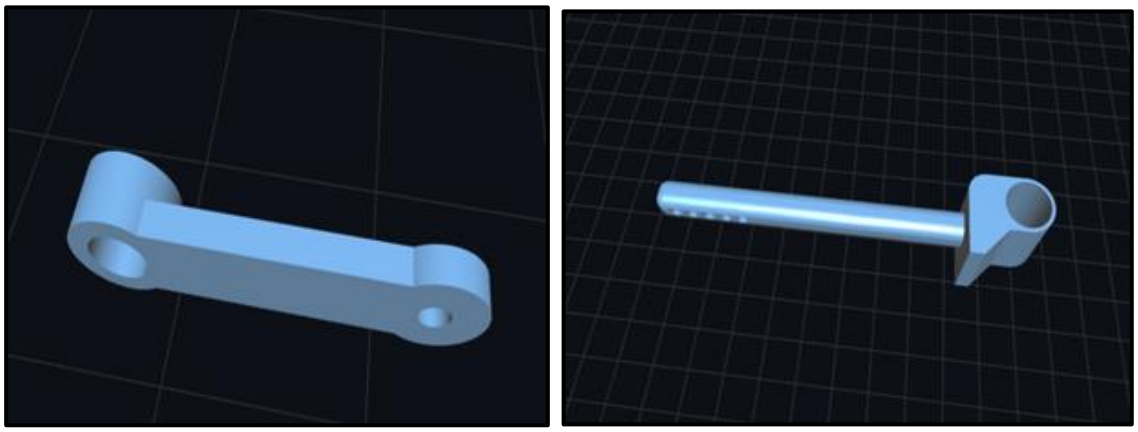
ŞEKİL 2.6 Tasarım – Direksiyon Sol ve Sağ Üst



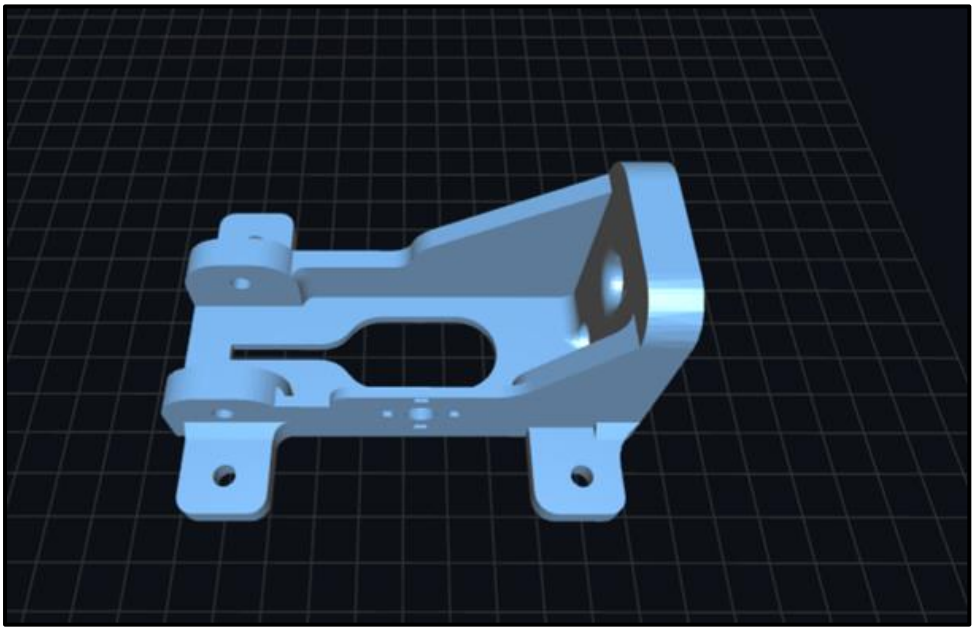
ŞEKİL 2.7 Tasarım – Direksiyon Sol ve Sağ Alt



ŞEKİL 2.8 Tasarım – Pedal ve Şaft



ŞEKİL 2.9 Tasarım Piston



ŞEKİL 2.10 Tasarım – Pedal Platformu

2.3. SİMÜLASYON

Simülasyon ortamı projenin görselliğini desteklemek için gelişitirildi. Geliştirme Unreal Engine 4 oyun motoru ile yapıldı. Geliştirmede harita tasarımı, sürüş kontrolü, araç aynaları ve mini map olmak üzere üç aşama yer almaktadır.

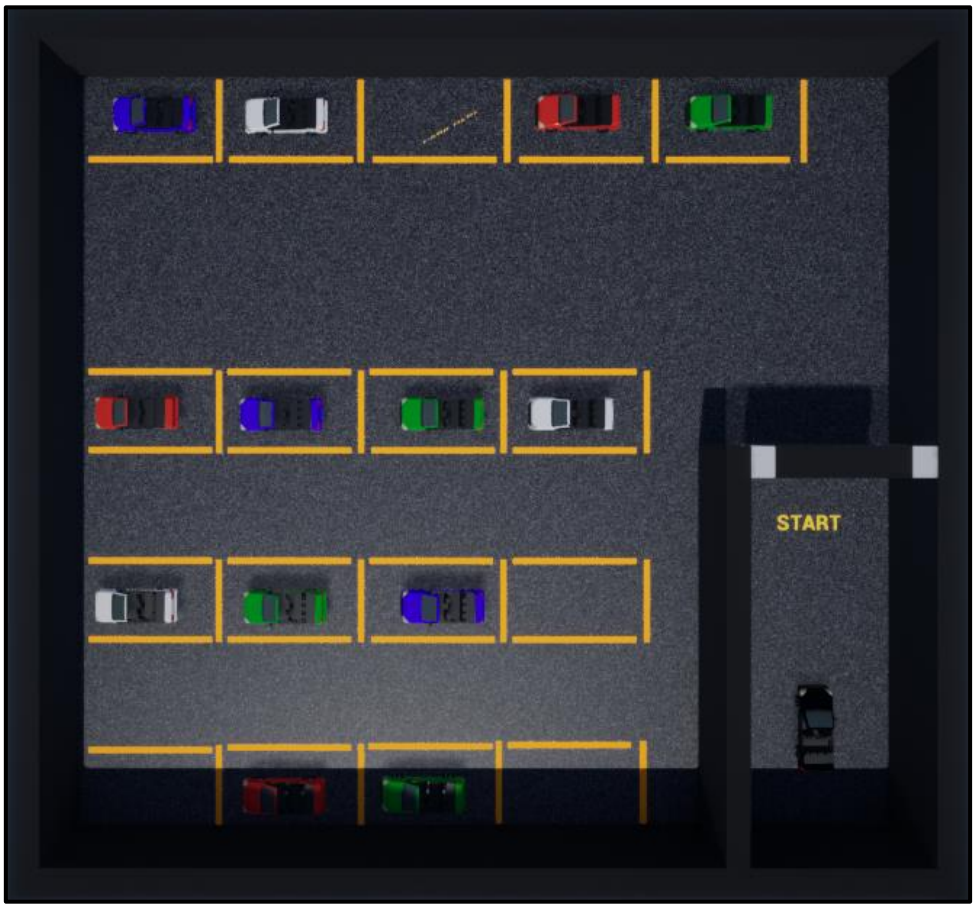
2.3.1. Harita Tasarımı

Bu aşamada otopark alanı ve şehir içi park alanı konseptli iki farklı harita tasarlanmıştır.

Otopark Alanı

Otoparkta giriş kapısı objesine, park çizgilerine ve araç varlıklarına yer verildi. Öncelikle ortama kare bir zemin konumlandırıldı ve asfalt dokusu oluşturuldu. Zeminin etrafı küp (cube) bileşenleri ile çevrelenerek duvar görünümü sağlandı.

Alan(plane) bileşenleri ile park çizgileri oluşturuldu, sarı ile renklendirildi ve istenen konumlara yerleştirildi. Park halindeki araçlar için araç varlığının renk materyali çeşitlendirilerek farklılık sağlandı ve araçlar park çizgileri arasına konumlandırıldı.



ŞEKİL 2.11 Garage Map Tasarımı – Kuş Bakışı

Giriş kapısı objesi 3 adet küp bileşeniyle oluşturuldu ve çevresine CollisionBox eklendi. CollisonBox ile aracın yaklaşması durumunda “Begin Overlap” çıktısı “1 (True)” ise konum verisini y ekseninde arttıran “OpenDoor” fonksiyonu çağırılarak kapının yukarı açılması sağlandı. Aracın kapıdan uzaklaşması durumunda CollisionBox “End Overlap” çıktısı “1 (True)” ise konum verisini y ekseninde azaltan “CloseDoor” fonksiyonu çağırılarak kapının kapanması sağlandı. Kullanıcı simülasyona otopark kapısından başlatıldı.

Aracın park etmesi beklenen konumu işaretlemek amacıyla “PARK HERE” ve “P”; başlangıç içinse “START” içerikli “TextRender” bileşenleri eklendi.

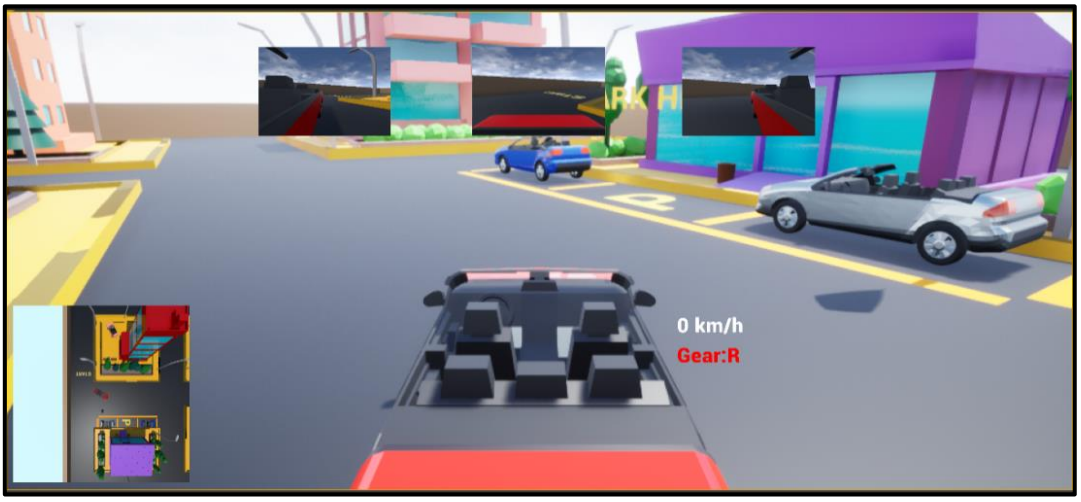
ŞEKİL 2.12 Garage Map - Garaj Kapısı

ŞEKİL 2.13 Garage Map – Park Alanı

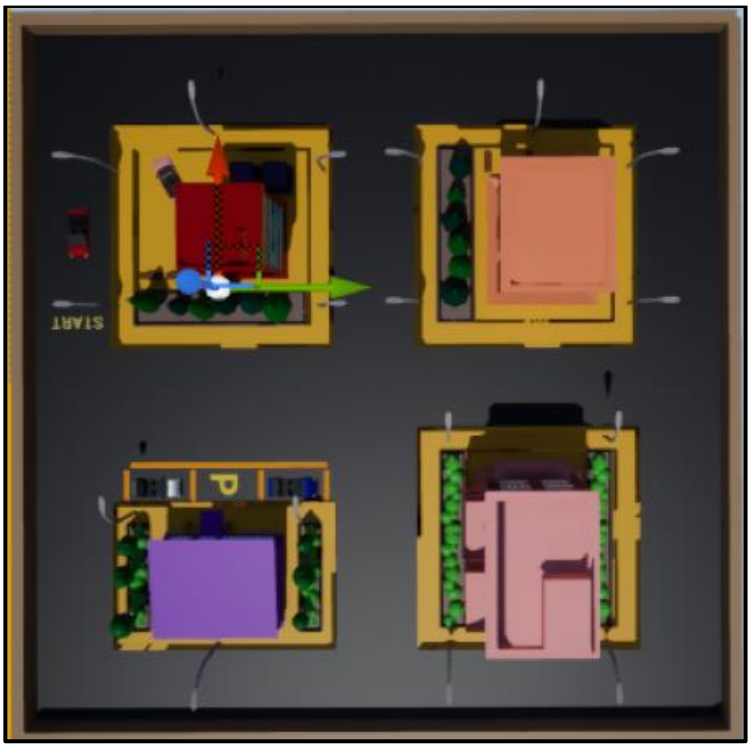
Şehir İçi Park Alanı

Bu haritada kare bir zemin üzerine bina, ağaç, lamba, araç varlıkları(asset) ve park alanı konumlandırılarak şehir görünümü elde edildi. Kare zeminin etrafı küp bileşenleri kullanılarak çevrildi ve bu sayede aracın harita dışına çıkması engellendi. Araç assetleri ise renk materyalleri değiştirilerek çeşitlendirildi ve park çizgileri arasına konumlandırıldı. Araçlara çarpma durumunun kontrolü ve doğru park kontrolü TriggerBox ile sağlandı.

Aracın park etmesi beklenen konumu işaretlemek amacıyla "PARK HERE" ve "P"; başlangıç içinse "START" içerikli "TextRender" bileşenleri eklendi.



ŞEKİL 2.14 City Map – Park Alanı

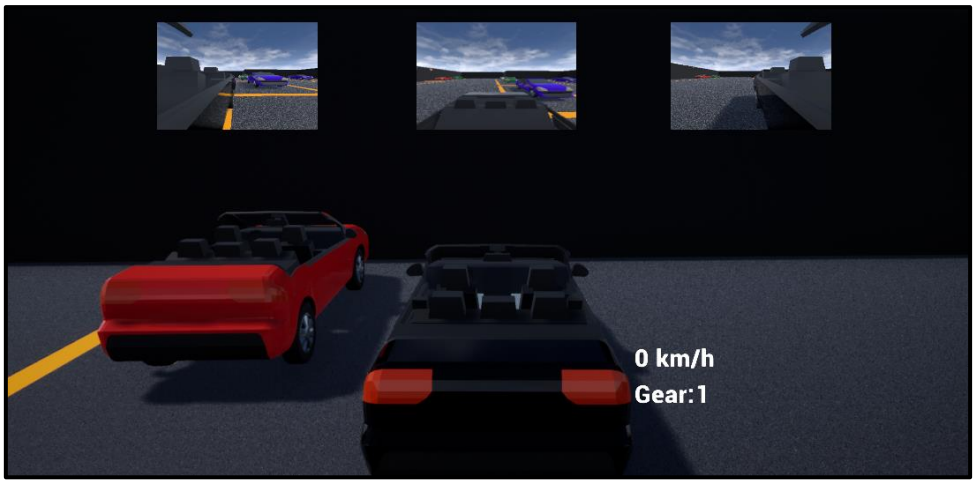


ŞEKİL 2.15 City Map Tasarım - Kuşbakışı

2.3.2. Araç Aynaları, Kamera ve Mini Harita

Araçta araç içi , araç dışı izleyici , arka , sağ ve sol dikiz aynası olmak üzere 5 farklı kamera açısı konumlandırılmıştır.

Unreal Engine 4 ‘de araç içi ve araç dışı izleyici için kamera bileşenleri; arka , sağ ve sol dikiz aynası için ise SceneCapture2D bileşeni oluşturulup araç objesinin istenen kısımlarına konumlandırıldı. Sonrasında simülasyonun araç dışı izleyici kamera görüntüsü ile başlaması ve kullanıcı isteyine göre tek tuşla araç içi kameraya geçebilmesi sağlandı. Arka, sağ ve sol dikiz aynası açıları ise pencerinin üst kısımda menü halinde simülasyon boyunca yer alması sağlandı.



ŞEKİL 2.16 Araç Dikiz Aynaları

ŞEKİL 2.17 Araç İçi Kamera

Aracın haritadaki konumu ve park edeceği alanı göstermek için simülasyonun sol alt tarafına mini harita konumlandırıldı. Spring arm ile aracın konumu değiştikçe mini haritadaki görüntü değişti. Aynı aynalardaki gibi SceneCapture2D ile haritanın görüntüsü simülasyonda gösterildi.

ŞEKİL 2.18 Mini Harita

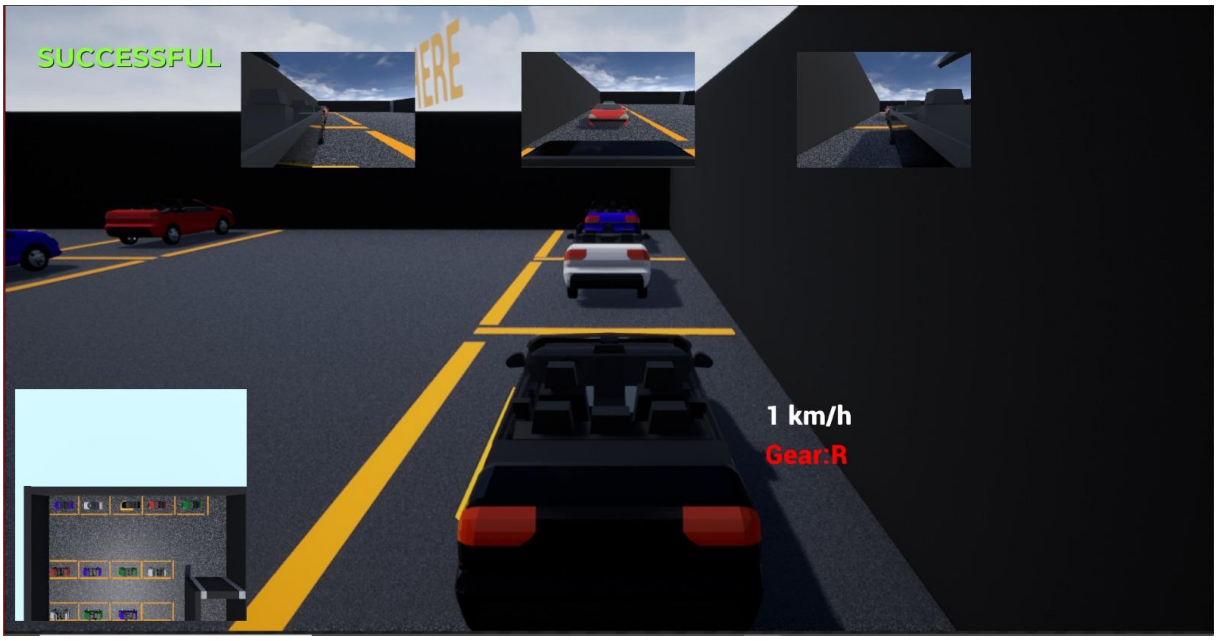
2.3.3. Sürüş Kontrolü

Sürüş için aracın ok tuşlarıyla sağa, sola, ileri ve geri gitmesi sağlandı. Aynı zamanada araç objesine hız (km/h) ve vites (1,2,3,N) göstergesi için TextRender bileşenleri eklendi. Bu göstergelerin araç içi ve araç dışı izleyici kameralarda ayrı konumlarda gösterilmesi sağlandı.

Aracın simülasyon ortamında yer alan diğer araçlara çarpma durumu TriggerBox ile kontrol edildi. Kontrol edilmesi gereken park halindeki araçlara TriggerBox entegre edildi. Herhangi bir araca ait TriggerBox "Begin Overlap" kontrol çıktısı "1 (True)" ise ekranın sol üst köşesinde "CRASH" uyarısı yer aldı.

ŞEKİL 2.19 CRASH Uyarısı

Aracın doğru park etmesi durumu 5 farklı TriggerBox ile kontrol edildi. Araç tekerlek konumlarına denk gelen alanlar için 4 tane TriggerBox yerleştirlidi ve “Begin Overlap” kontrolü alındı. Park alanına giriş kontrolü için de 1 adet TriggerBox park çizgisi dışına yerleştirildi ve “End Overlap” kontrolü alındı. Bu kontrollere ait çıkışlar “and” lojik kapısı ile kotrol edildi. Alınan sonuç “1 (True)” ise ekranın sol üst köşesinde “SUCCESSFUL” çıktısı yer aldı.

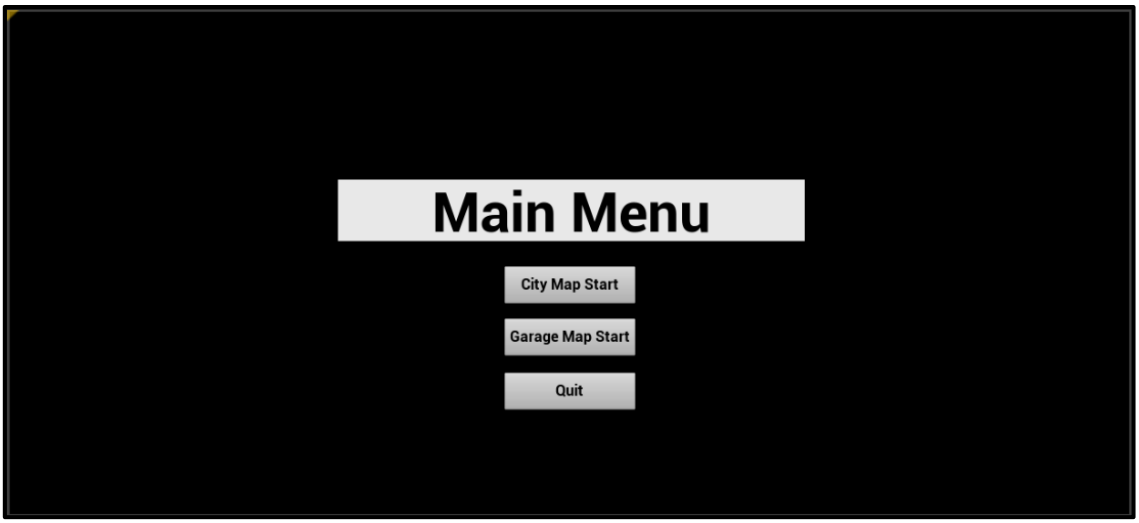


ŞEKİL 2.20 SUCCESSFUL Çıktısı

2.4. GUI

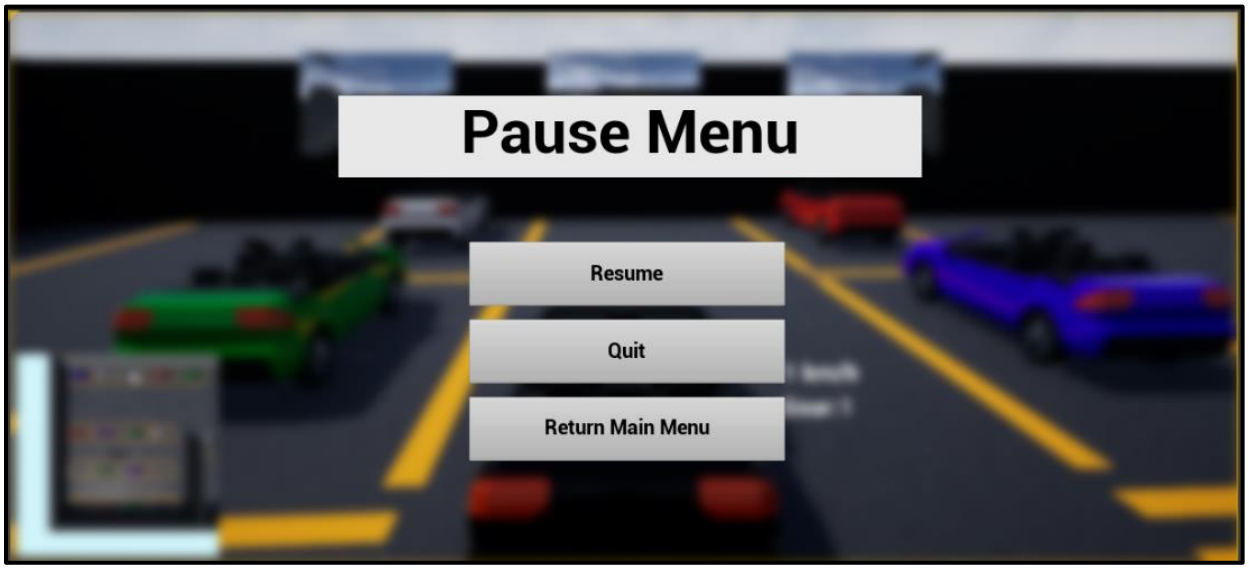
Simülasyonda kullanıcı ile etkileşim için Unreal Engine 4 üzerinde iki adet menünün yer aldığı bir arayüz geliştirldi. “imülasyon başlangıcında yer alan “Main Menu” de harita seçenekleriyle başlatma ve çıkış butonuna yer verildi. City Map (şehir içi park alanı) ve Garage Map (otopark alanı) olmak üzere 2 farklı harita seçeneği için butonlar oluşturuldu. Simülasyondan çıkış içinse farklı bir buton daha eklendi. Bunun için öncellikle simülasyondan ayrı bir level oluşturuldu. Daha sonrasında arayüzü kullanıcıya sunmak için “Main Menu” yapısı widget oluşturularak geliştirildi. Widget içerisinde opsiyonlara yönelik button ve text bileşenleri kullanıldı.

Kullanıcının tercih ettiği harita butonuna göre istenilen harita leveline yönlendirildi ve simülasyon başlatıldı; çıkış seçeneğinde ise program sonlandırıldı.



ŞEKİL 2.21 Main Menu Arayüzü

Simülasyon sırasında kullanıcının simülasyonu durdurup daha sonra devam edebilmesi veya çıkış yapabilmesi için "Pause Menu" yapısı widget oluşturularak geliştirildi. Widget içerisinde opsiyonlara yönelik button ve text bileşenleri kullanıldı. Kullanıcının tercihine göre simülasyone geri dönüş yapıldı, program sonladırıldı, veya "Main Menu" nün bulunduğu levele geçiş sağlandı.



ŞEKİL 2.22 Pause Menu Arayüzü

2.5. CONTAINER - DRIVER

2.5.1. USB Bellek

Bil bilgisayarın çalışma hızını etkileyen en önemli faktörlerden biri hardware dosya okuma/yazma hızıdır. Bu bilgiyi göz önünde bulundurarak simülasyonumuzu ve simülasyonumuzu çalıştıran işletim sistemini içerisinde barındıracak USB bellek için mümkün olduğunca okuma/yazma hızı yüksek bir cihaz tercih etmeye çalıştık. Kullandığımız bellek, USB 3.1 teknolojisi desteği, 200 MB/s okuma, 30 MB/s yazma hızıyla simülasyonumuz için USB portu üzerinden mümkün olan en uygun çalışma ortamını sağlamaktadır.

TABLO 2-1 USB Bellek Bilgileri

Input/ Output Port	USB 3.1
Storage	32 GB
Reading Speed	200 MB/s
Writing Speed	30 MB/s

2.5.2. İşletim Sistemi Seçimi

İşletim sistemi olarak Windows 10'u tercih ettik. Bu tercihimizi etkileyen en önemli faktör, donanımımızdan gelen datayı simülasyonumuzda kullanabilmemize yardımcı olan rawinput isimli unreal engine pluginin sadece Windows tarafından destekleniyor olmasıydı. Rawinput, Microsoft XInput API yardımıyla kullanıcı tarafından geliştirilen cihazlardan input alınmasına yardımcı olan bir Unreal Engine pluginidir.

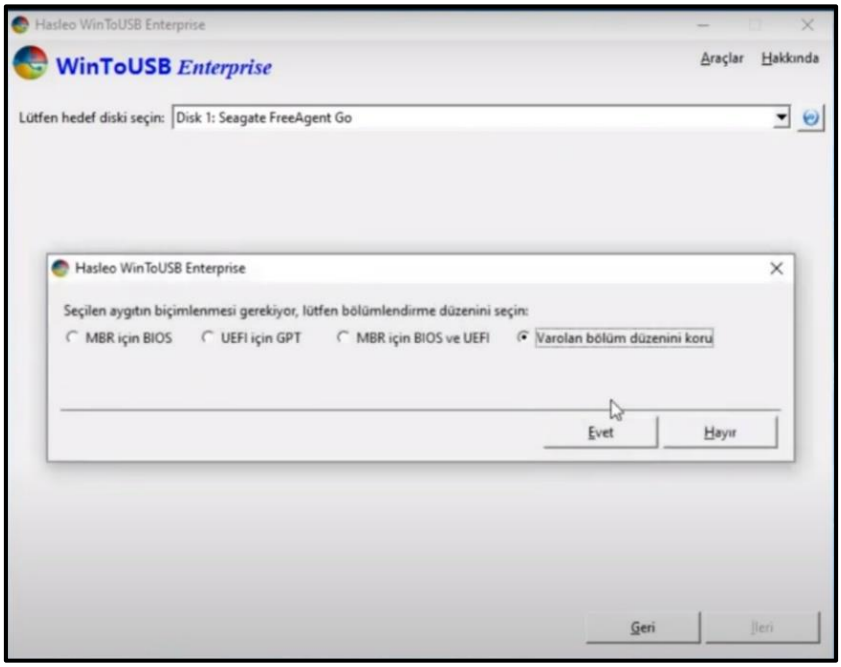
2.5.3. İşletim Sisteminin Hazırlanması

Belleğimiz içerisine simülasyonumuzu ve simülasyon gereksinimlerini kalıcı olarak depolayabileceğimiz bir işletim sisteminin bulunmasına ihtiyaç duyuyorduk. Bu ihtiyacı **WinToUSB** isimli bir program yardımıyla karşıladık.

WinToUSB, Windows işletim sisteminin bir flaş disk veya harici bir sabit sürücü gibi çıkarılabilir bir sürücü üzerinde kurulmasını sağlayan bir araçtır. Bu program ile herhangi bir kullanıcı, orijinal CD'nin ISO görüntüsünü, fiziksel disk kullanmak yerine kurulum kaynağı yapmak için harici sürücüye kopyalayabilir.

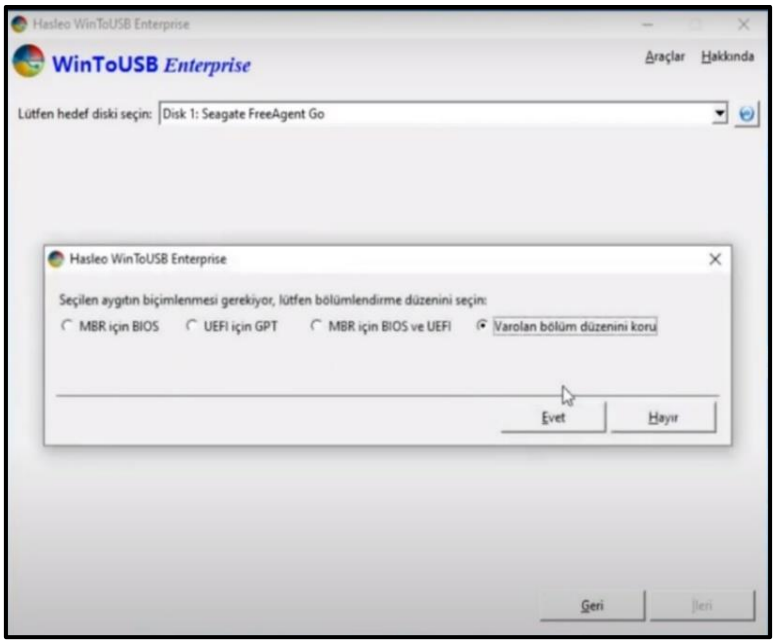
İşletim sistemi kurulumunu aşağıdaki adımları takip ederek gerçekleştirdik;

1. ISO dosyasının seçilmesi



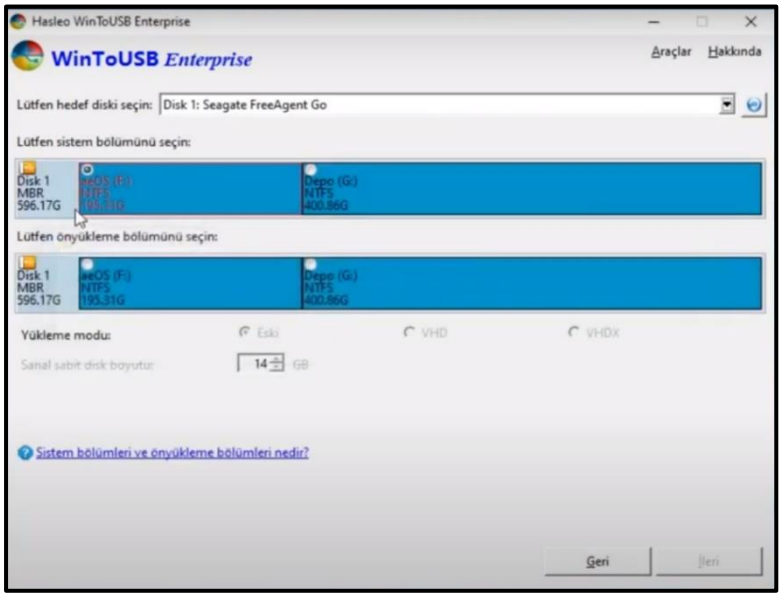
ŞEKİL 2.23 ISO Dosyasının Seçilmesi

2. Booting tercihinin seçilmesi



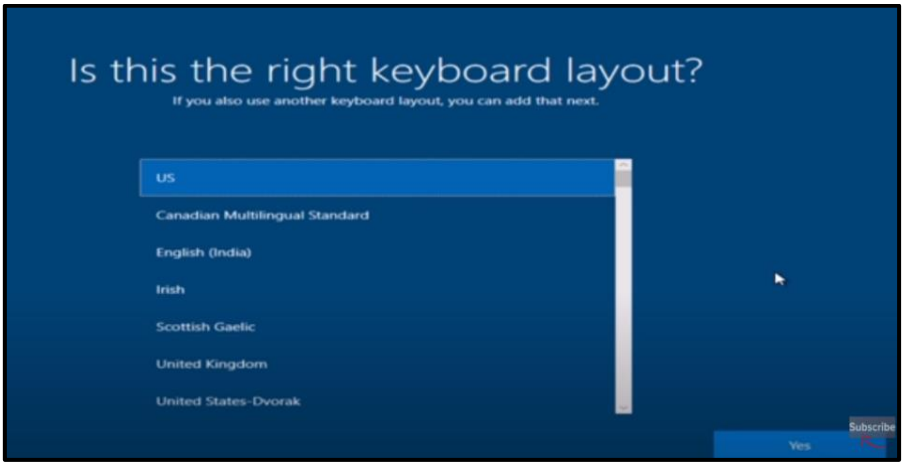
ŞEKİL 2.24 Booting Tercihi

3. Disk bölümlerinin belirlenmesi



ŞEKİL 2.25 Disk bölümlerinin belirlenmesi

4. Operating sistmeme ait dil, keyboard, user… vb. tercihlerin belirlenmesi



ŞEKİL 2.26 Sistem Tercihleri

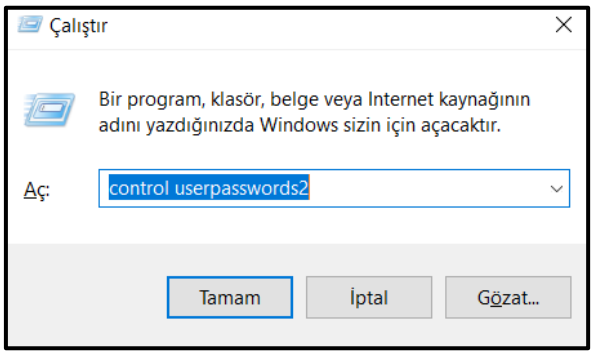
2.5.4. İşletim Sistemi Ortamının Hazırlanması

İşletim sistemi kurulduktan sonra simülasyonun çalıştırılabilmesi için bazı düzenlemeler yapılması gerekmektedir. Bu düzenlemelerin bazıları kullanıcı deneyimini arttırmak (oturum açma şifresinin kaldırılması vb.), bazıları simülasyonun çalışabilmesi için (uygulama bağımlılıkları örnek: directX runtime) yapılması gerekmektedir. Bu düzenlemeler aşağıda açıklamalarıya beraber sıralanmıştır.

İşletim Sistemi Oturum Açma Şifresinin Kaldırılması

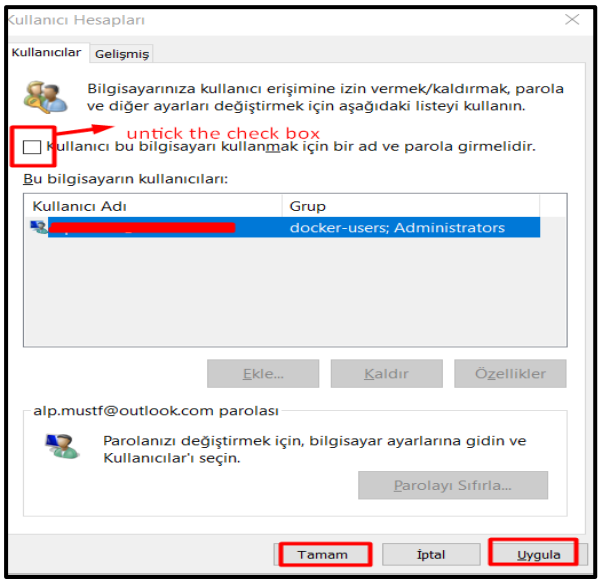
İşletim sistemi başladığı zaman kullanıcı oturum açmak, şifre girmek gibi işlemler yapmayacaktır. Bu sebeple işletim sisteminin bu adımları doğrudan geçip simülasyonun başlatılabilmesi için aşağıdaki adımlar sırasıyla uygulanmıştır.

1. **Windows ❖ + R** tuşlarına basınız.
2. Açılan çalıştır penceresine **control userpasswords2** komutunu yazarak enter tuşuna basınız.



ŞEKİL 2.27 cmd Komutu 1

3. Kullanıcıyı seçin ve checkbox seçimini kaldırarak uygula ve tamam tuşlarına basınız.

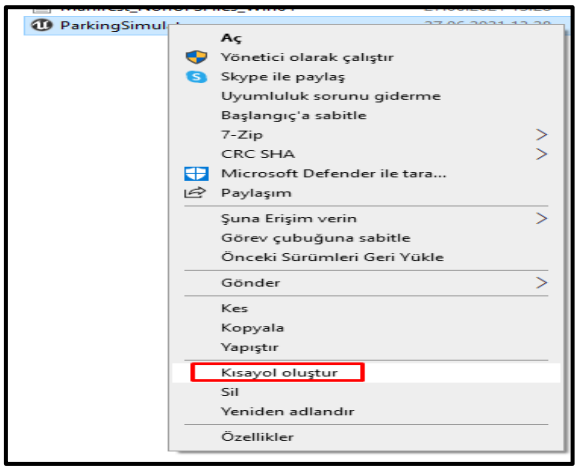


ŞEKİL 2.28 Kullanıcı Seçimi

Simülasyonun Açılış Sonrasında Otomatik Başlatılması

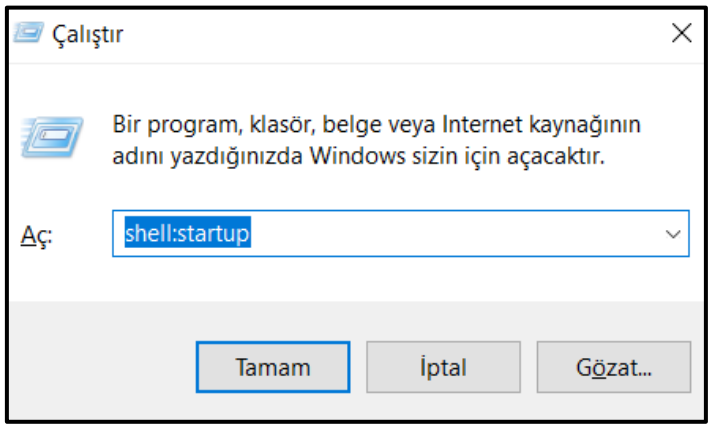
Kullanıcı sistem çalıştırıldıktan sonra simülasyonun başka bir işlem gerekmeksizin başlaması bir proje gereksinimi olarak belirleniştir. Bu gereksinimin sağlanabilmesi için aşağıdaki adımlar sırasıyla uygulanmıştır.

1. Simülasyonun bir kısa yolunu oluşturun.



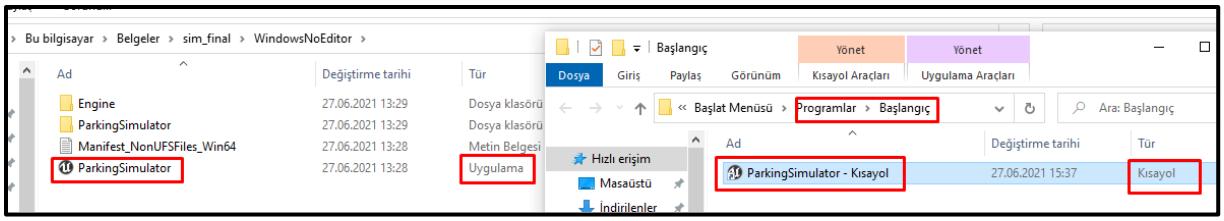
ŞEKİL 2.29 Simülasyon Kısayolu Oluşturma

2. **Windows ❖ + R** tuşlarına basınız.
3. **Çalıştır** penceresine **shell:startup** komutunu yazarak enter tuşuna basınız**.**



ŞEKİL 2.30 cmd Komutu 2

- Daha önce oluşturmuş olduğunuz kısayol dosyasını, açılan **başlangıç** klasörüne kopyalayınız.



ŞEKİL 2.31 Kısayolun Taşınması

Simülasyon Gereksinimleri

Simülasyonun çalışabilmesi için işletim sistemine kurulması gereken program ve kütüphaneler aşağıda listelenmiştir.

1. Windows visual c++ runtime
2. DirectX runtime

3. KULLANILAN ÜRÜNLER VE MAALİYET

Projede kullanılan üürnlerin maliyet tablosu TABLO 3.1' de detaylı şekilde sunulmuştur.

TABLO 3-1 Maaliyet Tablosu

Malzemeler	Gereken Miktar	Birim Maliyeti	Reçete Maliyeti
STM32f103	1	30 ₺	**30 ₺**
10K potansiyometre	3	6 ₺	**18 ₺**
Kablo	10 M	1 ₺	**10 ₺**
Buton	2	1 ₺	**2 ₺**
Direksiyon Simidi	1	150 ₺	**150 ₺**
Plexiglass	2	25 ₺	**50 ₺**
Rulman	1	50 ₺	**50 ₺**
Somun	8	25 ₺	**20 ₺**
USB 8gb	1	30 ₺	**30 ₺**
		Toplam:	**370 ₺**