Uzay Savaşcısı

1st Aydın Can Altun
180202117

2nd Barış Arslan
180202112

***Abstract*—Uzay Savaşcısı, mikrodenetliyiciler kullanılarak geliştirilmiş bir oyundur. Oyunun amacı uzay gemisinin kendisine doğru gelen engelleri yoketmesi veya aşmasıdır.**

***Index Terms*—Arduino Mega, Push Button, SSD1306 OLED, 7-led segment display, ldr, led, potantimeter**

I. Giriş

Uzay Savaşcısı, kolay ve zor oyun modlarından oluşan, Oyun başladığında 8x16'lık bir harita oluşturan ve bu haritaya belirli aralıklarla meteor, uzay çöpü ve bonus puan objeleri ekleyen ve bu objeleri koşullar sağlandığı zaman birer adım ilerletilmesini sağlayan ve kullanıcıdan gelen giriş değerleri ile oyuncunun yönettiği uzay gemisinin hareket ettiği bir oyundur.

II. Yöntem

A. Devre Tasarımı

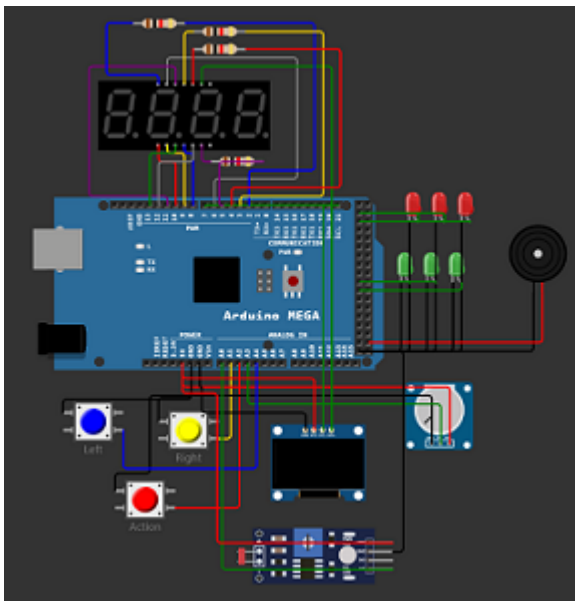

Fig. 1. Devre Tasarımı.

Devre tasarlanırken, bir adet Arduino Mega, bir adet SSD1306 OLED, üç adet push buton, altı adet led, bir adet buzzer, bir adet 7-led display segment, bir adet potansiyometre, bir adet LDR ışık sensörü kullanılmıştır.

Ledlerin üç tanesi oyuncunun silah haklarını göstermek için, kalan üç tanesi can haklarını göstermek için kullanılmıştır.

Push buttonların iki tanesi oyun ilk başladığında menüde oyun zorluğunu seçilmesi için kullanılmıştır, kalan tek button ise menüde aksiyon almak ve oyun başladığında atış yapmak için kullanılmıştır.

7-led display segment, oyuncunun skorunu göstermek için kullanılmıştır.

Potansiyometre, oyun başladıktan sonra oyuncunun yönettiği gemiyi hareket ettirmesi için kullanılmıştır.

Buzzer oyuncuya bir obje çarptığı zaman uyarı vermesi için kullanılmıştır.

LDR ışık sensörü, değer değişikliği sonrası oyun haritasının kontrastını değiştirmek için kullanılmıştır.

SSD1306 OLED ekran ise yukarıda elemanların işlevlerini gerçekleştirdikten sonra kullanıcıya sonucunun gösterilmesi için kullanılmıştır.

B. Menü Tasarımı

Cihaz ilk çalıştırıldığında OLED ekran üzerinde menü çizdirilir ve kolay zorluğu seçilmiş olarak gelir. Kullanıcı mavi push button ile sola, sarı push button ile sağa gidebilir. Kırmızı push button ile de seçilen zorlukta oyun başlar.

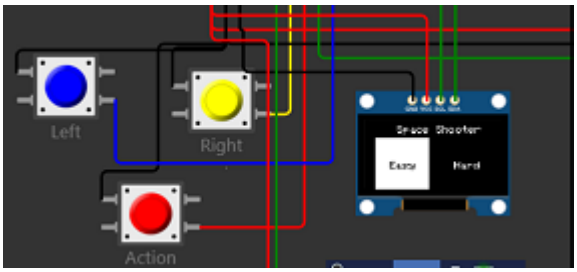

Fig. 2. Kolay zorluk seçilmiş menü görüntüsü.

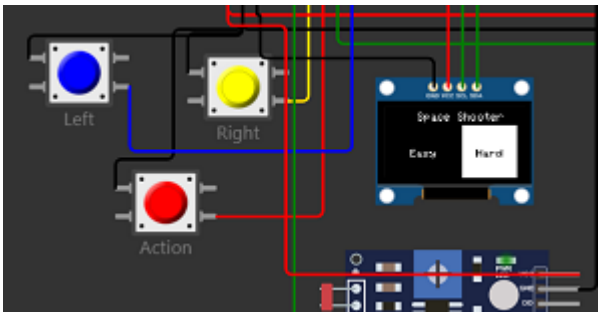

Fig. 3. Kolay zorluk seçilmiş menü görüntüsü.

C. Oyun Haritası Matrisi ve Oyun Haritasının Çizdirilmesi

Oyun haritası, 8x16'lık integer matrisi üzerinden oluşturulur. Matrisin üzerindeki değerler bir objenin ne olduğunu temsil eder.

Matris üzerindeki bir geminin baz değeri 6'dır ve 6'nın üzerindeki değerler geminin canını temsil eder. Örneğin matris üzerinde 7 değeri var ise bu 1 canı kalmış bir gemiyi temsil eder. Matris üzerindeki bir meteorun baz değeri 2'dir ve 2 ile 4 arasındaki değerler meteoru temsil eder. Harita matrisine yeni eklenmiş bir meteorun matris üzerindeki değeri 4'dür çünkü meteor atış ile haritadan silinir. Matris üzerindeki bir uzay çöpünün matris üzerindeki değeri 1'dir ve ilk atış ile yokedilmiş olur. Bonus puan objesinin ise matris üzerindeki değeri 5'dir ekstra bir canı yoktur ve üzerine ateş edildiği anda matris üzerinden silinir.

```
0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 9
0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0
0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0
```

Fig. 4. Harita matrisini.

OLED ekran genişliği 128 piksel ve uzunluğu 64 piksel olacak şekilde ayarlanmıştır. Bu sebepten ötürü harita 8 piksel x 8 piksel karelere bölünmüştür. Oyun haritasında, oyuncunun yönettiği gemi haritanın diğer ucuna bakan bir üçgen, bir meteor yuvarlak, uzay çöpü içi dolu bir kare ve bonus puan içinde yıldız olan bir kare ile temsil edilmiştir.

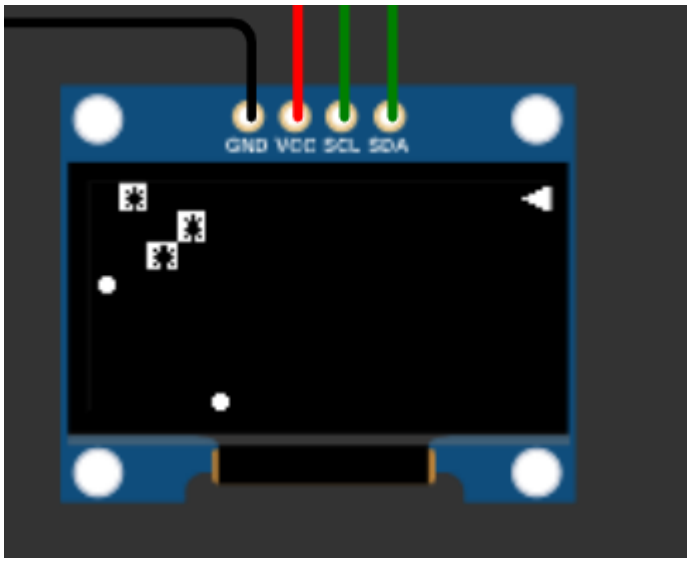

Fig. 5. Harita matrisinin oled üzerinde çizdirilmiş görüntüsü.

D. Objeleri İlerletme, Can Kazanma ve Can Kaybetme

Objeler oyun zorluğuna göre en geç saniye başına bir adım ileriye gitmektedir. Obje bir sonraki adımda oyuncunun yönettiği gemiye çarpacağı durumda obje bonus ise geminin canı artrılıyor eğer çarpacak obje bir engel ise oyuncu gemisinin canı bir azaltılır.

for i **from** 0 **to** 7 **do**
  for j **from** 15 **down to** 0 **do**
    if spaceMap[i][j][i][j] $>$ 0 **and** spaceMap[i][j] $<$ 6 **then**
      $newPosition \leftarrow j + 1$
      if newPosition $>$ 16 **then**
        spaceMap[i][j] $\leftarrow$ 0
      else
        if i $=$ playerPosition **and** newPosition $=$ 15 **then**
          if spaceMap[i][j] $=$ 5 **then**
            spaceMap[$playerPosition$][15] $\leftarrow$ spaceMap[$playerPosition$][15] + 1
            spaceMap[i][j] $\leftarrow$ 0
          else
            if isTakeDamage**is not true then**
              isTakeDamage $\leftarrow$ **true**
              damageTakenTime $\leftarrow$ **millis**
              TONE BUZZER
              newHealth $\leftarrow$ spaceMap[$playerPosition$][15] $-$ 1
              if newHealth6 **then**
                spaceMap[$playerPosition$][15] $\leftarrow$ newHealth
              else
                spaceMap[$playerPosition$][15] $\leftarrow$ 6
              end if
              spaceMap[i][j] $\leftarrow$ 0
            else
              spaceMap[i][j] $\leftarrow$ 0
            end if
          end if
        else
          obstacleValue $\leftarrow$ spaceMap[i][j]
          spaceMap[i][j] $\leftarrow$ 0
          spaceMap[i][$newPosition$] $\leftarrow$ obstacleValue
        end if
      end if
    end for
  end for

Alg. 1. Objeleri bir adım ilerletme algoritması

E. Oyun Akışı

Menü üzerinden oyun zorluğu başladıktan sonra öncelikle oyunun başlangıcı için varsayılan değerler(Silah hakkı 3, Toplam Can 3, Blok ilerletme süresi 1000 ms, Hasar aldıktan sonra dokunulmazlık süresi 3000ms) değişkenlere atanır. Oyuncunun yöneteceği gemi matris'in ilk satırın, sonuncu sütununa yerleştirilir ve başlangıç tarihi saniyede dikkate alınarak milisaniye cinsinde tutulur. Oyuncunun

hareketleri potansiyometrenin değeri değiştiği durumlarda; eğer potansiyometrenin değeri 0-512 arasında ise bir adım yukarıya çıkar; eğer potansiyometrenin değeri 513-1023 arasında ise gemi bir adım aşağıya iner. Oyuncu her adım attıktan sonra bir skor puanı kazanır bu skor puanı 7-led segment display üzerinde gösterilir.

Eğer oyun başladığından beri 1 saniye geçtiyse tüm bloklar bir adım ileriye ilerletilir ve ardından başlangıç sütünundaki rastgele bir blok üzerine rastgele bir obje(meteor, uzay çöpü, bonus) oluşturulur.

Eğer oyuncu aksiyon butonuna basar ise oyuncu bir mermi atışı yapar ve mermi ilk değdiği objenin canını azaltır(Meteor için 2 vuruş sonra yok olur. uzay çöpü ve bonus ilk atış sonrası yok olur).

Eğer matris üzerinde geminin temsiliyet değeri 6'ise oyuncunun hiç bir canı kalmamıştır ve oyun sonlanır ve Oyun Bitiş ekranı oyuncuya gösterilir.

Oyun bitiş ekranında oyuncunun kazandığı toplam skor ekran üzerinde gösterililir ve oyuncunun bir butona basması sonucu tekrar menüye yeni bir oyun başlatması için yönlendirilir.

III. Deneysel Sonuçlar

```
if remainingGun > 3 then
  for i from 14 down to 0 do
    target ← spaceMap[playerPosition][i]
    if target > 0 then
      if target = 5 then
        spaceMap[playerPosition][i] ← 0
        BREAK LOOP
      else if target = 1 then
        spaceMap[playerPosition][i] ← 0
        BREAK LOOP target > 2 and target < 6
        newHealth ← target − 1
        if newHealth = 2 then
          spaceMap[playerPosition][i] ← 0
        else
          spaceMap[playerPosition][i] ← newHealth
        end if
        BREAK LOOP
      end if
    end if
  end for
  remainingGun ← remainingGun − 1
end if
```

Alg. 2. Ateş etme algoritması

```
isFull ← false
for i from 0 to 8 do
  if spaceMap[i][0] = 0 then
    isFull ← false
    BREAK LOOP
  else
    isFull ← true
  end if
end for
if isFullis not true then
  randomObstaclePosition ← random(0, 8)
  while true do
    if spaceMap[randomObstaclePosition][0] = 0
    then
      BREAK LOOP
    else
      randomObstaclePosition ← random(0, 8)
    end if
  end while
  randomObstacleType ← random(0, 3)
  if randomObstacleType = 0 then
    spaceMap[randomObstaclePosition][0] ← 1
    randomObstacleType = 1
    spaceMap[randomObstaclePosition][0] ← 4
  else if randomObstacleType = 2 then
    spaceMap[randomObstaclePosition][0] ← 5
  end if
end if
```

Alg. 3. Rastgele obje oluşturma algoritması

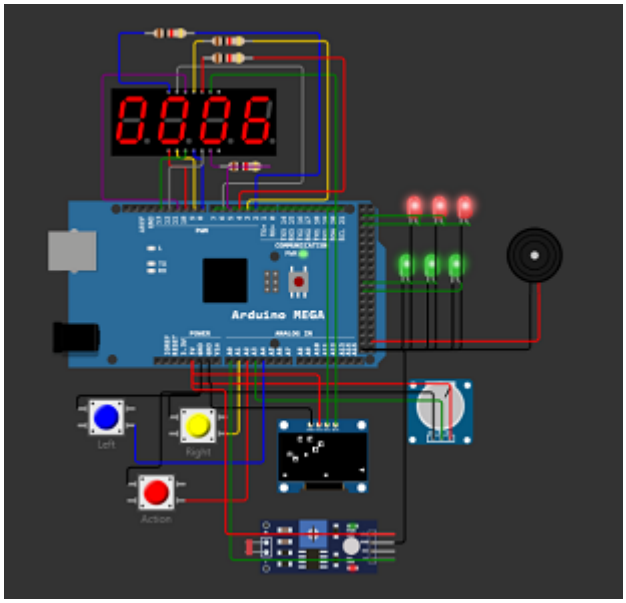

Fig. 6. Oyun Görüntüsü

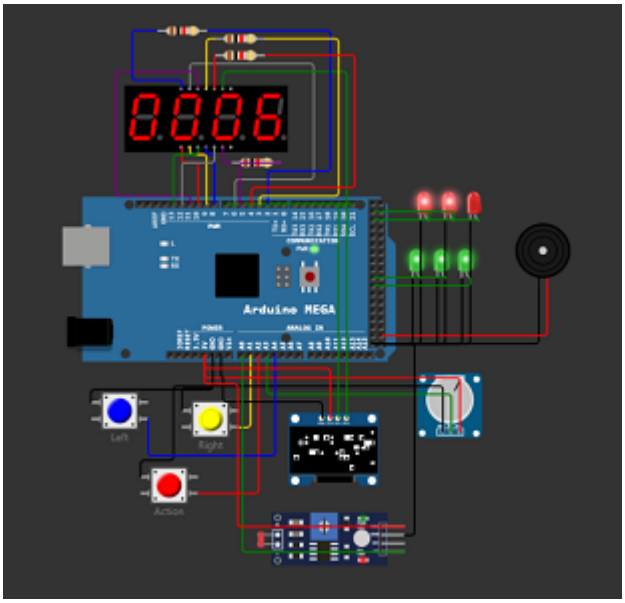

Fig. 7. Mermi hakkını harcamış oyun görüntüsü.

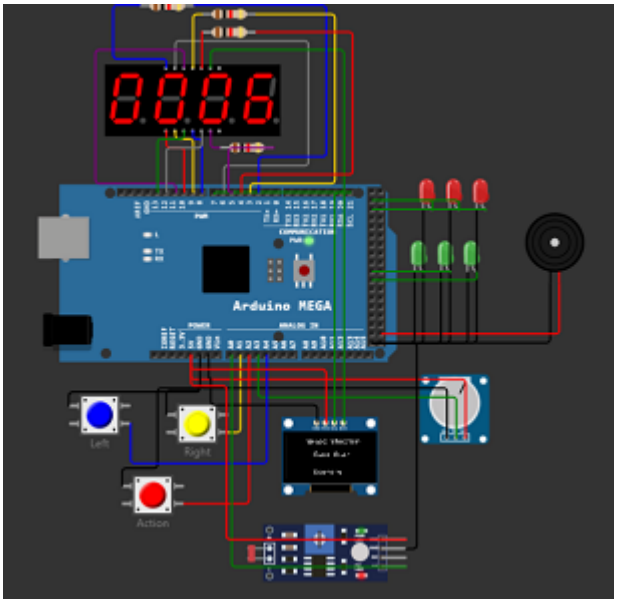

Fig. 9. Game Over oyun görüntüsü.

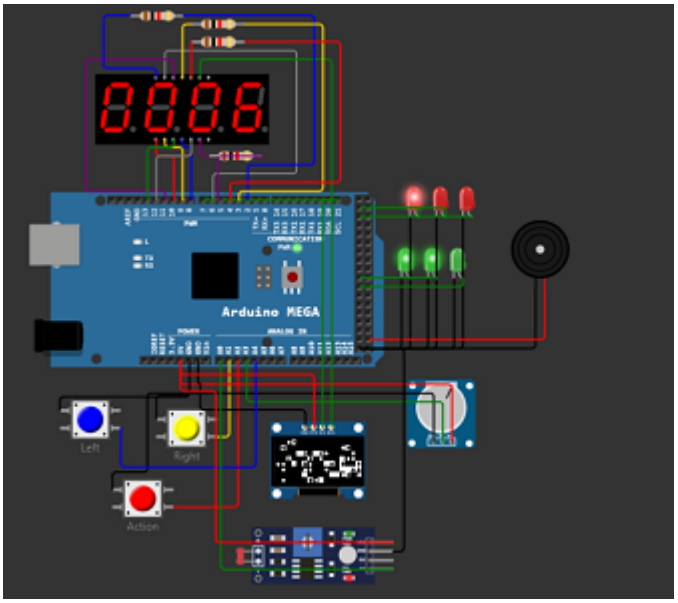

Fig. 8. Mermi haklarını ve can hakkını kaybetmiş oyun görüntüsü.

IV. Sonuç

Mikrodenetleyici ve sensörleri kullanılarak oled ekrana aksiyon alabileceğiniz bir menü ve oyun tasarlanmıştır. Geliştirme aşamasında mikrodenetleyicinin belleği doldurulmadan geliştirme yapılması sağlanmıştır. İki farklı oyun modu eklenmiştir. Kolay oyun modunda her saniyede objeler bir adım ilerletilir ve rastgele bir obje oluşturulur. Zor oyun modunda oyun her 10 saniyede bir yüzde 20 hızlandırılır.

References

[1] $https://docs.wokwi.com/parts/wokwi-arduino-mega.$
[2] $https://docs.wokwi.com/parts/wokwi-7segment.$
[3] $https://github.com/adafruit/Adafruit_SD1306.$
[4] $https://docs.wokwi.com/parts/wokwi-buzzer.$
[5] $https://docs.wokwi.com/parts/wokwi-potentiometer.$
[6] $https://docs.wokwi.com/parts/board-ssd1306.$
[7] $https://docs.wokwi.com/parts/wokwi-photoresistor-sensor.$
[8] $https://docs.wokwi.com/parts/wokwi-pushbutton.$
[9] $https://randomnerdtutorials.com/guide-for-oled-display-wit$
[10] $https://github.com/adafruit/Adafruit_SD1306.$