Proje Uygulaması : Klavye tuşlarıyla (W , A , S , D) Arduino üzerinden kablosuz RC araba kontrolü

- Proje Uygulamasında Kullanılan Donanımlar
  - Oyuncak Araba Şasesi
  - PC
  - Arduino UNO
  - L298N Motor Sürücü Kartı
  - HC-06 Bluetooth Modülü
  - 3 adet 9V Pil
  - 11 adet Jumper Kablo
- Proje Uygulamasında Kullanılan Yazılımlar
  - Arduino IDE
  - Visual Studio Community 2017

Öncelikle kullanılan donanım ve yazılımları tanıyalım.

Oyuncak Araba Şasesi : Klavye ile kontrol edeceğimiz önde (sağ-sol kontrolü) ve arkada (ileri-geri kontrolü) olmak üzere iki tane motoru olan ve tekerleklerin motorlara bağlı şekilde aracımızın temel hareketleri yapmasını sağlayan iskelet araba yapısı.

PC : Klavyeden araca hareket vereceğiz.

Arduino UNO : PC den gelen verileri Bluetooth aracılığıyla okuyup Motor Sürücü Kartına ileten mikrodenetleyici kart.

L298N Motor Sürücü Kartı : Arduinodan gelen verileri aracımızın motorlarına ileten elektronik entegre kart.

HC-06 Bluetooth Modülü : PC den gelen verileri Arduinodaki pinlere iletmek için kullandığımız , seri haberleşme sağlayan elektronik kart.

9V Pil : Motor Sürücü Kartımıza ve Arduinomuza güç veren kaynak.(3 adet kullanıldı.)

Jumper Kablo : Devremizdeki ilgili bağlantıları yapmamıza yarayan kablolar.

Visual Studio : Bluetooth üzerinden Arduinoya veri göndermek için kullandığımız Microsoft yapımı C# , C++ , Visual Basic , F# vs… dilleri ile yazılım uygulamaları geliştirilen bir yazılım geliştirme platformudur.

Arduino IDE : Arduino mikrodenetleyicisine gömülmek için yazılan komutları yazdığımız bir yazılım geliştirme platformudur.Kodlarımızı yazdıktan sonra Arduinoya gömmek için bir kablo aracılığıyla komutlarımızı Arduinoya upload etmemizi de sağlar.

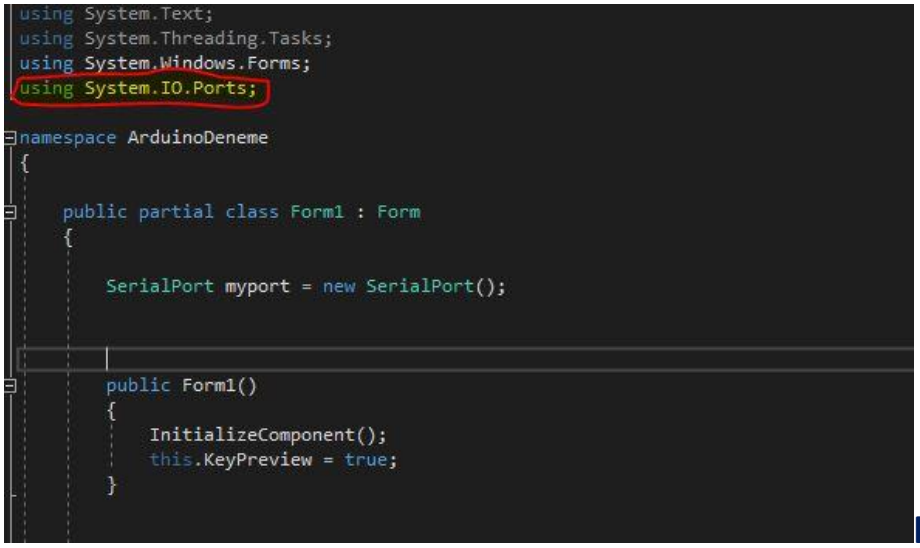
Bu uygulamada yukarıdaki yazılım ve donanımlar kullanılarak bir bilgisayardaki W tuşu ile ileri , S tuşu ile geri giden A tuşu ile sola ve D tuşu ile sağa yön verilen bir RC araba yapılmıştır.

Uygulamanın Yapımı

Uygulama için öncelikle PCmizin ve Arduinomuzun haberleşmesini sağlamak gerekir.PC mizdeki tuşlar vasıtasıyla seri port üzerinden Bluetooth modülümüze iletilen verilerin , oradan da Arduinomuza iletilmesi sağlanmalıdır.

Bunun için hem verici tarafımızda yazacağımız kodların (Visual Studio) hem de alıcı tarafında yazacağımızın kodların (Arduino IDE) yazıldığı ortamımıza haberleşme ile ilgili kütüphaneleri eklememiz gerekecektir.

C# ta kütüphane eklemek için **using** komutu kullanılır.Biz de haberleşme sağlamak için C# tarafında komutlarımıza **using System.IO.Ports;** komutunu ekliyoruz.



Resim 1.1

Not:Yukarıdaki System.IO.Ports; komutunu ekleyeceğimiz yer Formumuzdaki nesnelerin kontrolü için yazdığımız komutların olduğu yerdir.Visual Studio programını açtığımızda sol üst köşede File->New->Project ve gelen ekranda Windows Form App dediğimizde yeni bir proje açılır ve karşımızda bir form nesnesi oluşur.Bu form nesnesinin üzerinde mouse ile sağ tıklayıp Wiew Code dediğimizde ilgili yere gelmiş oluruz.Komutlarımız hep buradadır.

Aynı şekilde alıcı komutlarımızı yazdığımız Arduino IDE sinde de haberleşme için kütüphane eklemeliyiz.

Arduinoda haberleşme için en yukarıya **#include <SoftwareSerial.h>** komutunu ekleriz.

Sonra Visual Studioda Resim 1.1 de görüldüğü gibi eklediğimiz kütüphanenin içerisinde bulunan **SerialPort** sınıfından **myport** isimli bir nesne tanımladık.**SerialPort** sınıfı sayesinde Arduinomuzla seri haberleşme sağlayacağız.

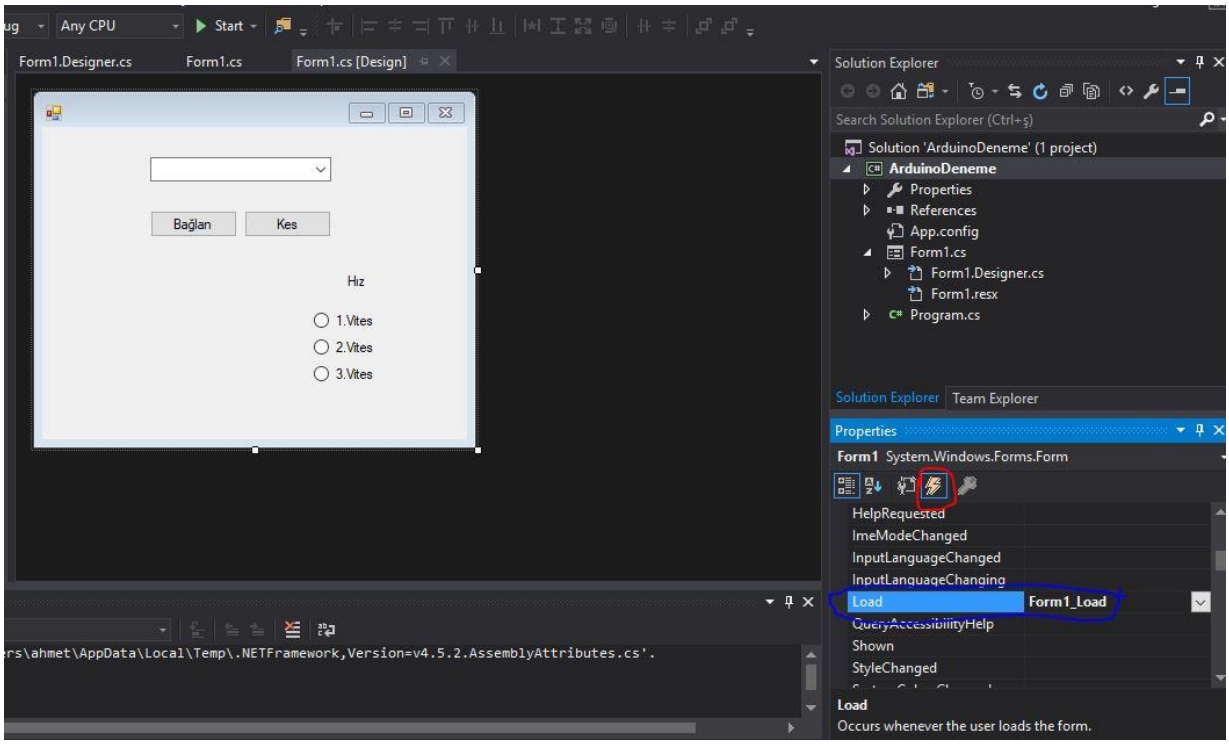
Seri haberleşmeyi bilgisayarımızdaki bluetoothun bağlı olduğu seri porttan yapacağımız için bu portu seçmemiz gerekecek.Bunun için ise Formumuza 1 adet ComboBox ekliyoruz.

Bunun için Visual Studioda sol tarafta bulunan toolboxtan ComboBox nesnesini formumuza süreklememiz yeterli.Bize comboBox1 isminde bir ComboBox nesnesi üretir.(İsmi sağ alt tarafta propertiesteki Name özelliğinden değiştirilebilir.)

ComboBox dizi gibi içerisinde birden fazla veri tutar.Bizde bilgisayarımızdaki seri portların listesini yazdırmak için uygulamamız çalıştığında (yani Form1 isimli formumuz load edildiğinde) o andaki seri portları comboBox1 isimli nesnemize atamak için SerialPort sınıfındaki GetPortNames() metodunu kullanacağız.

Öncelikle load eventi ile Form1 load edildiğinde(yani uygulama çalıştığında olacak komutlar) olacak komutları yazmak için Visual Studioda sağ alt tarafta bulunan properties kısmının yanındaki şimşek işaretli yere tıklayıp (Resim 1.2) eventlerin olduğu liste karşımıza çıkar.

Not : Event herhangi bir olayda programın nasıl çalışacağını söyleyeceğimiz metotlardır.Örneğin mouse ile butona tıklanılırsa herhangi bir resimin boyutunu büyült veya klavyeden ESC tuşuna basılırsa uygulamayı kapat gibi.

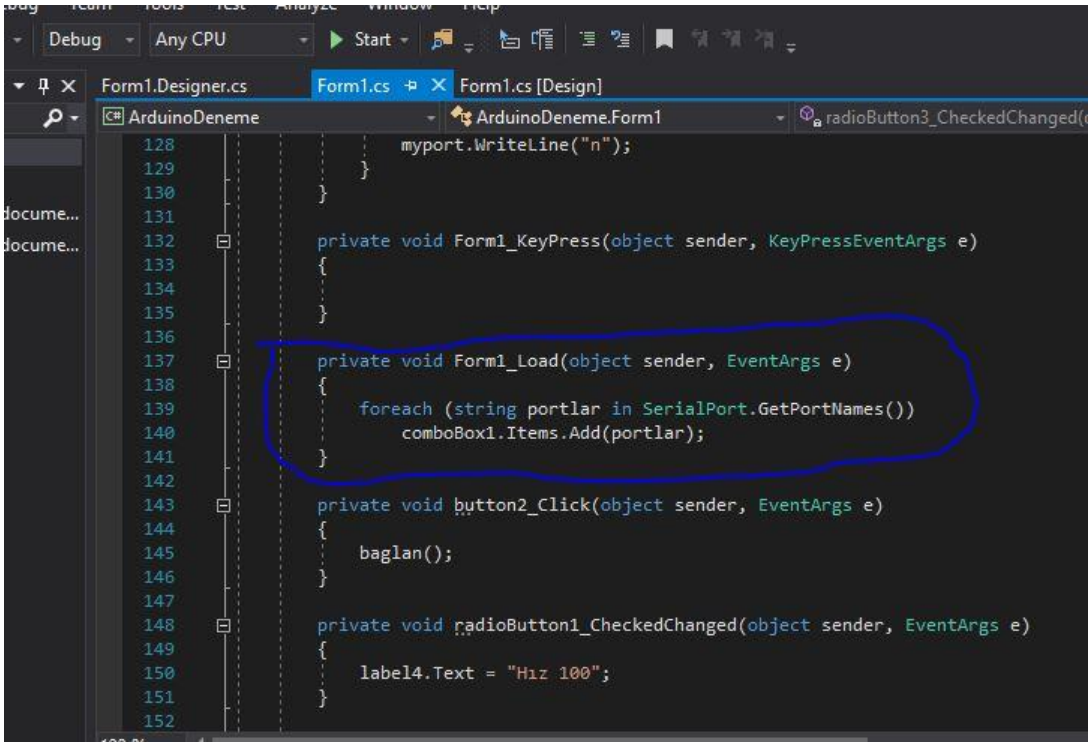


Resim 1.2

Event listemizden Load'a çift tıkladığımızda metodun içerisine SerialPort sınıfındaki GetPortNames metodu sayesinde comboboxımıza portlarımızı atayarak uygulama çalıştığında port listemizi comboboxımızdan elde etmiş oluyoruz.(Resim 1.3)

(Listeden bluetoothumuzun bağlı olduğu portu seçerek bağlantı sağlamış oluruz.)

Not : Eğer HC-06 bluetooth modülüne ilk defa bağlanacaksanız parola isteyecektir.Parola genellikle ya 0000 ya da 1234 olur.



Resim 1.3

Şimdi ise comboboxtan portumuzu seçip bağlantı yapmak için gerekli bağlantı kodlarını yazacağız.Bunun için baglan isimli bir metot yazıp buton ile tıklanıldığında (Resim 1.2 deki bağlan butonu) bu metodu(Resim 1.3 te görülüyor) çağır diyeceğiz.

Bağlantı metodumuzda BaudRate haberleşmemizin hızı , PortName kullanacağımız portumuz ve Open() metodu da haberleşmemizin başlamasını sağlayan metotdur.(Resim 1.4)

Try-Catch yapısı ise bir Aykırı Durum Yönetimidir.

Yani eğer try blokları içerisindeki komutlar çalışmazsa catch blokları içerisindeki komutları çalıştır demiş oluruz.

Bunu yapmamızın sebebi herhangi bir sebepten ötürü bağlantı sağlayamazsak bize bunu söylemesidir.

```
22      public Form1()
23      {
24          InitializeComponent();
25          this.KeyPreview = true;
26      }
27
28
29
30      private void baglan()
31      {
32          try
33          {
34              myport.BaudRate = 9600;
35              myport.PortName = comboBox1.Text;
36              myport.Open();
37              MessageBox.Show(comboBox1.Text + " nolu porta bağlantı sağlandı");
38          }
39
40          catch (Exception)
41          {
42              MessageBox.Show("Port Bağlantısı Yok.");
43          }
44      }
45
```

Resim 1.4

Arduino tarafında ise haberleşmeyi başlatmak için setup fonksiyonumuzun içine Serial.begin(9600) komutunu yazarız.Burada 9600 yukarıdan da tahmin edeceğiniz gibi haberleşme hızımızdır.

```
void setup() {

  pinMode(ileriPin, OUTPUT);
  pinMode(geriPin, OUTPUT);
  pinMode(sagPin, OUTPUT);
  pinMode(solPin, OUTPUT);
  Serial.begin(9600);
}
```

Resim 1.5

Bu işlemlerden sonra haberleşmemiz için gerekli komutlar yazılmış olur.

Portumuza bağlanıp haberleşme sağlandıktan sonra yapılacak iş klavyeden basacağımız tuşlara göre motorlara sinyal iletmek yani motorları hareket ettirmektir.

Şekildeki fritzing devresinde görüldüğü üzere biri sağ-sol yönü diğeri ileri-geri için iki tane motorumuz vardır.

Bir DC motorun çalışması için kutuplarından birine + diğerine – yani bir kutuba HIGH diğer kutuba LOW vermemiz gerekir.

Fritzing devresindeki L298N motor sürücünün sol baştan sayarsak Arduinoya bağlı ilk üç pini ileri-geri motorunun hareketleri için , diğer üç pini ise sağ-sol motorunun hareketleri için veri(sinyal) yollar.En baş ve en sondaki pinler ise sırasıyla ileri-geri ve sağ-sol motorlarına Arduino sayesinde PWM sinyali uygulayarak motorların dönüş hızlarını ayarlamamızı sağlar.(Motor Sürücünün Arduinodaki 5 ve 6. Pinlerine bağlı olan pinler.)

Not : PWM uygulayacağımız pinler Arduinodaki PWM destekli pinlere bağlanmalıdır.Bu pinler 3,5,6,9,10,11 dir.Numaraların hemen altlarında ~ gibi bir işaret vardır.

Fritzing devresinden de görüldüğü gibi Motor Sürücüsüne bağlı pinlerimiz 5 , 6 , 8 , 9 , 10 , 11 dir.Bunlardan 5 ve 6 motorlara PWM sinyali uygulayarak dönüş hızlarını ayarlamamızı sağlayacaktır.

Öncelikle dijital olarak kullanacağımız pinleri setup fonksiyonunda belirttik.(Resim 1.6)

```
#include <SoftwareSerial.h>

int solPin = 9;
int sagPin = 8;
int geriPin = 11;
int ileriPin = 10;
int e1 = 5;
int e2 = 6;

void setup() {

  pinMode(ileriPin, OUTPUT);
  pinMode(geriPin, OUTPUT);
  pinMode(sagPin, OUTPUT);
  pinMode(solPin, OUTPUT);
  Serial.begin(9600);
}
```

Resim 1.6

Şimdi ise motorları hareket ettirmek için yazdığımız fonksiyonlarımızı açıklayalım.

Önceki Visual Studio resimlerindeki formda görüldüğü gibi , formun sağ alt tarafında üç tane radio butonumuz var.Bu radio butonlar yanlarındaki labelText lerde yazdığı üzere seçili olduğunda ilgili hızları ayarlayacaktır.

Devredeki PWM sinyali maksimum 255 minimum 0 değeri alır.255 değeri HIGH a 0 değeri LOW a karşılık gelir.Motor sürücümüze devreden de görüldüğü üzere 18V verdiğimiz için burada 255 18V'u 0 ise 0V'u sağlar.

Bunların ikisi arasındaki değer ise ilgili voltaja oranla voltaj değeri vererek hız kontrolü yapmamıza olanak sağlar.Misal olarak formdaki 100 değeri için (Formdaki 1.vites radio butonu) 18V'un 255'de 100'ü yani (18*100)/255=7.05 V olduğundan 100 değeri seçildiğinde motor 7.05V voltaj değeri alacaktır ve hızı biraz azalacaktır.

Şimdi aracımızı hareket ettirecek fonksiyonlara bakalım.

Aracımız arka motorda radiobutonlara göre 3 farklı hızla ileri yönde harekete , geri yönde harekete , ön motorda da sağ ve sol yönde harekete geçecek şekilde yapılmıştır.

```
void ileri1(){
  analogWrite(e1 , 100);
  digitalWrite(ileriPin, HIGH);
  digitalWrite(geriPin , LOW);

}

void ileri2(){
  analogWrite(e1 , 170);
  digitalWrite(ileriPin, HIGH);
  digitalWrite(geriPin , LOW);

}

void ileri3(){
  analogWrite(e1 , 255);
  digitalWrite(ileriPin, HIGH);
  digitalWrite(geriPin , LOW);

}

void geri(){
  analogWrite(e1 , 210);
    digitalWrite(ileriPin, LOW);
  digitalWrite(geriPin , HIGH);
}
```

```
void sag(){
  analogWrite(e2 , 210);
    digitalWrite(sagPin, HIGH);
  digitalWrite(solPin , LOW);
}

void sol(){
  analogWrite(e2 , 210);
    digitalWrite(sagPin, LOW);
  digitalWrite(solPin , HIGH);
}

void dur(){
    digitalWrite(ileriPin, LOW);
  digitalWrite(geriPin , LOW);
}

void duz(){
    digitalWrite(sagPin, LOW);
  digitalWrite(solPin , LOW);
}
```

İleri1 , İleri2 ve İleri3 fonksiyonlarında 10.pine HIGH 11.pine LOW verilerek arka motorun ileri yönde hareketi sağlanmış olur.Sırasıyla 100 , 170 ve 255 PWM değerleriyle de farklı hızlarda hareket sağlanmış olur.

Geri fonksiyonunda ise 10.pine LOW 11.pine HIGH verilerek arka motorun geri yönde hareketi sağlanmış olur.

Dur fonksiyonu ile 10. ve 11. pinlere yani motorun her iki kutbunada LOW değeri verilerek motorun hareket etmemesi sağlanmış olur.

Sağ fonksiyonunda 8.pine HIGH 9.pine LOW verilerek ön motorun sağ yönde hareketi sağlanmış olur.

Sol fonksiyonunda ise 8.pine LOW 9.pine HIGH verilerek ön motorun sağ yönde hareketi sağlanmış olur.

Son olarak Düz fonksiyonu ile 8. ve 9. pinlere yani motorun her iki kutbunada LOW değeri verilerek ön motorun hareket etmemesi düz bir konuma gelmesi sağlanmış olur.

Not : Motorların hangi kutuplarında hangi hareketi yapacağını deneme yanılma yoluyla uygulamak gerekir.Eğer beklediğinizin tersi bir hareket olursa pinlere uyguladığınız HIGH ve LOW değerlerini değiştirmeniz veya devredeki ilgili pinleri değiştirmeniz yeterli olacaktır.

Arduino tarafında son olarak verici tarafından (PC) gelen verilere göre çalıştıracağımız fonksiyonları yazacağız.

Arduinoda karşı taraftan gelen verileri Serial.read() komutuyla okuruz.(Resim 1.9)

```
void loop() {
char veri=Serial.read();

switch(veri){
  case '1' : ileri1(); break;
  case '2' : ileri2(); break;
  case '3' : ileri3(); break;
  case 'b' : geri(); break;
  case 'r' : sag(); break;
  case 'l' : sol(); break;
  case 's' : dur(); break;
  case 'n' : duz(); break;

}

}
```

Resim 1.9

Burada PC deki uygulamamızdan gelen veriyi okuyup char tipindeki veri isimli değişkene atıyoruz ve gelen veriye göre switch-case yapısıyla fonksiyonları çalıştırıp aracımızın hareketini sağlamış oluyoruz.

Şimdi sıra bu fonksiyonları çalıştıracak verilerin PCmizden iletilmesine geldi.

Öncelikle nasıl çalıştığını bir hatırlayalım.

Klavyemizdeki **W** tuşuna bastığımızda (basılı tuttuğumuzda) aracımızın ileri gitmesini sağlamak istiyoruz.Ancak ileri yönlerdeki hareketlere mahsus olarak hız kontrollerimiz vardı.Bunun için hem W tuşuna bastığımızda hem de radiobutonlardan birini seçtiğimizde ilgili radio butona göre ileri yönde hareket sağlanacak.

Eğer elimizi W tuşundan çekersekte aracımız duracak.

Aynı şekilde **S** tuşuna bastığımızda geri gidecek ve S tuşundan elimizi kaldırdığımızda da duracak.

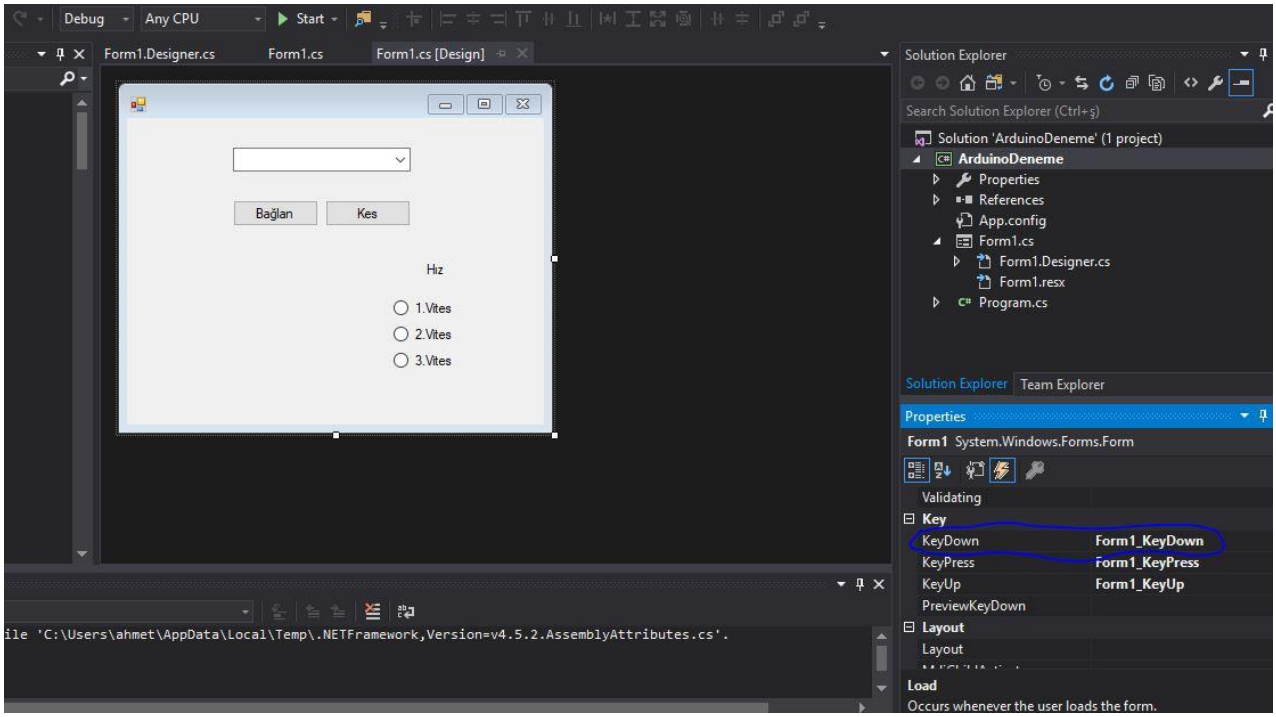
A tuşuna bastığımızda ön tekerlekler sola dönecek , **D** ye bastığımızda ise sağa dönecek.

Bunlardan da elimizi kaldırdığımızda motoru durdurup tekerlerin düz bir konuma gelmesini sağlamış olacağız.

Visual Studio ortamında klavyeden bir tuşa basınca ve basılan tuş kaldırılınca yapılacak komutları yazmak için KeyDown ve KeyUp eventlerini kullanmamız gerekir.

Not : Eğer sadece KeyDown eventini kullanırsak ilgili tuşa bastığımızda elimizi çeksek bile ilgili komut çalışmaya devam edecektir.Bunun için KeyUp eventi ile beraber kullandık.

Öncelikle aracımızın ileri yönde 100 değerinde hız ile gitmesi için W tuşuna basıp ilgili radiobutonu seçmemiz lazımdır.Bunun için Event listesinden KeyDown eventine çift tıklayıp(Resim 2.0) ilgili fonksiyonun içine portumuza ‘1’ verisini yaz diyeceğiz.(Resim 2.1)Çünkü alıcı tarafımızda ‘1’ bilgisi gelirse ileri1 fonksiyonunu çalıştıracak.



Resim 2.0

Not : C# ta seriportumuza veri yazmak için WriteLine metodunu kullanırız.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{

    if (e.KeyCode == Keys.W && radioButton1.Checked==true)
    {
        myport.WriteLine("1");
    }

    if(e.KeyCode==Keys.W && radioButton2.Checked==true)
    {
        myport.WriteLine("2");
    }

    if(e.KeyCode==Keys.W && radioButton3.Checked==true)
    {
        myport.WriteLine("3");
    }


    if (e.KeyCode==Keys.S)
    {
        myport.WriteLine("b");
    }

    if (e.KeyCode == Keys.A)
    {
        myport.WriteLine("l");
    }

    if (e.KeyCode == Keys.D)
    {
        myport.WriteLine("r");
    }
```

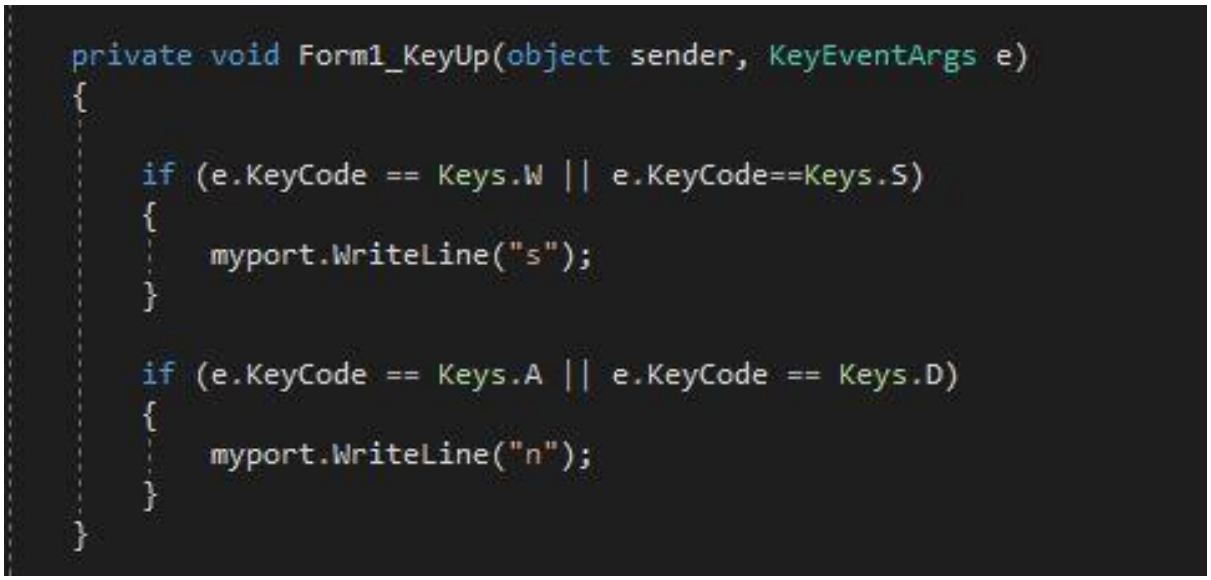
Resim 2.1

Resim 2.1 de görüldüğü gibi if komutu ile eğer parametre olarak gelen tuş **W** ise ve radiobuton1 seçili ise seriportumuza ‘1’ verisini ileterek arduinomuzda ileri1 fonksiyonumuzu çalıştırmış oluyoruz.

Resim 2.1 deki diğer if komutlarıylada ilgili verileri arduinomuza iletip ilgili fonksiyonları çalıştırmış oluyoruz.

Burada KeyDown eventini kullanarak ilgili tuşlara bastığımız sürece aracımıza hareket sağlıyoruz.Ancak aracın durmasını sağlayan dur() ve duz() fonksiyonlarını çalıştırmak için de ilgili tuşlardan elimiz kaldırılınca (KeyUp olunca) KeyUp eventi ile arduinomuza dur() ve duz()

fonksiyonlarını çalıştıracak verileri iletmemiz gerekecektir.Bunun için event listemizden KeyUp eventine çift tıklayıp ilgili komutları yazarız.(Resim 2.2)



```
private void Form1_KeyUp(object sender, KeyEventArgs e)
{

    if (e.KeyCode == Keys.W || e.KeyCode==Keys.S)
    {
        myport.WriteLine("s");
    }

    if (e.KeyCode == Keys.A || e.KeyCode == Keys.D)
    {
        myport.WriteLine("n");
    }
}
```

Resim 2.2