

Accelerometer (ivme ölçer); cihazınızın X, Y, Z koordinatlarındaki bilgisini ve hareketin o anki hızını alır. Cihaza hangi yönden ivme verirsiniz, o kısmın hızı ölçülür. Örneğin; cihazınız dikey pozisyondayken X koordinatından yukarı doğru ivme kazandırırız, X koordinatı negatif değer alacaktır ki, cihazın üzerinde bir top olduğunu hayal edersek, top hangi koordinata ivme kazandırılıyorsa, aksi yönde hareket edecektir.

AppBar içerisinde bulunan On/Off düğmesiyle Accelerometer özelliğini açıp kapatılabilir. On/Off düğmesinin bağlı olduğu AppBarIconButton_Click event handler'ı içerisinde bu işlemin nasıl gerçekleştirildiğini görebilirsiniz.

On/Off düğmesini kullanarak Accelerometer'ı aktifleştirdiğimizde, ivmede değişiklik olma anında tetiklenecek **accelerometer_CurrentValueChanged** aksiyonunu da hazırlamış oluyoruz.

Kullanıcıya geri bildirimde bulunacak kısım ise **timer_Tick** event'idir. Bu event sayfanın çalışma anında (Constructor içinde tanımlaması yapılmıştır) Timer nesnesi ile devreye alınır. Belli periyotlarda (tanımlandığı yerde 30 milisaniye olarak belirtilmiştir.) Accelerometer'dan aldığı bilgiyi ekrana yazdırır.

Bir örnekle bunu gösterelim. Yeni bir Windows Phone projesi açın ve aşağıda verilen kodları uygun yerlere yerleştirin.

Uygulamada bir **Timer** (Zamanlayıcı) kontrolü ile, sürekli olarak **Compass** (pusula) sensöründen gelen yön bilgisi alınır.

C# örneği

```
using System;
```

```
using Microsoft.Phone.Controls;
```

```
using Microsoft.Devices.Sensors;
```

```
using System.Windows.Threading;
```

```
namespace sdkRawSensorDataCS
```

```
{
```

```
    public partial class AccelerometerPage : PhoneApplicationPage
```

```
    {
```

```
        Accelerometer accelerometer;
```

```
        DispatcherTimer timer;
```

```
        Microsoft.Xna.Framework.Vector3 acceleration;
```

```
        bool isValid;
```

```
        public AccelerometerPage()
```

```
        {
```

```
            InitializeComponent();
```

```
            if (!Accelerometer.IsSupported)
```

```

{
    // Öncelikle cihazımız accelerometer destekliyor mu bunu kontrol ediyoruz

    statusTextBlock.Text = "desteklenmiyor";
    ApplicationBar.IsVisible = false;
}
else
{
    // destekliyorsa bir timer oluşturuyoruz.

    timer = new DispatcherTimer();
    timer.Interval = TimeSpan.FromMilliseconds(30);
    timer.Tick += new EventHandler(timer_Tick);
}
}

private void ApplicationBarIconButton_Click(object sender, EventArgs e)
{
    if (accelerometer != null && accelerometer.IsDataValid)
    {
        // Durdurmak için bastığımız düğme eğer accelerometer doluysa ve üzerindeki veri onaylanırsa
        accelerometer'ı durduruyor.

        accelerometer.Stop();
        timer.Stop();
        statusTextBlock.Text = "accelerometer durduruldu.";
    }
    else
    {
        if (accelerometer == null)
        {
            // accelerometer boş ise önce instance alıyoruz
            accelerometer = new Accelerometer();

            // 20 milisaniyede update olacağını söylüyoruz
            accelerometer.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);

```

// biz istediğimiz değeri atasak da accelerometer bunu desteklemiyor olabilir, bu durumda accelerometer en yakın süreyi üzerine alacaktır. Biz de gerçek zaman aralığını bir textblock'a yazdırıyoruz

```
timeBetweenUpdatesTextBlock.Text = accelerometer.TimeBetweenUpdates.TotalMilliseconds  
+ " ms";
```

//değer değiştikçe çalışacak event'i belirtiyoruz.

```
accelerometer.CurrentValueChanged +=  
newEventHandler<SensorReadingEventArgs<AccelerometerReading>>(accelerometer_CurrentValueC  
hanged);
```

```
}
```

```
try
```

```
{
```

//accelerometer ı başlatıyoruz ve aynı zamanda timer'ı da çalıştırıyoruz.

```
statusTextBlock.Text = "Accelerometer başlatıldı";
```

```
accelerometer.Start();
```

```
timer.Start();
```

```
}
```

```
catch (InvalidOperationException ex)
```

```
{
```

```
statusTextBlock.Text = "hata oluştu, ne yazık ki accelerometer başlayamadı. Hata:" +  
ex.Message;
```

```
}
```

```
}
```

```
}
```

```
void accelerometer_CurrentValueChanged(object sender,  
SensorReadingEventArgs<AccelerometerReading> e)
```

```
{
```

//sensör datalarının doğruluğu onaylanıyor. Normal koşullarda burada bir sıkıntı olmasa da yine de okuma anında bir sıkıntı olursa isDataValid false dönmektedir.

```
isDataValid = accelerometer.IsDataValid;
```

//acceleration üzerinde çizim yapacağımız vektör nesnesi

```
acceleration = e.SensorReading.Acceleration;
```

```
}
```

```

void timer_Tick(object sender, EventArgs e)
{
    if (isDataValid)
    {
        statusTextBlock.Text = "accelerometer çalışıyor.";

        // textblocklara accelerator den gelen bilgi yazılıyor
        xTextBlock.Text = "X: " + acceleration.X.ToString("0.00");
        yTextBlock.Text = "Y: " + acceleration.Y.ToString("0.00");
        zTextBlock.Text = "Z: " + acceleration.Z.ToString("0.00");

        // Line nesnesi ile gelen uzaklığı ekrana bastırıyoruz.
        xLine.X2 = xLine.X1 + acceleration.X * 100;
        yLine.Y2 = yLine.Y1 - acceleration.Y * 100;
        zLine.X2 = zLine.X1 - acceleration.Z * 50;
        zLine.Y2 = zLine.Y1 + acceleration.Z * 50;
    }
}
}

```

XAML örneği

```
<phone:PhoneApplicationPage
```

```

x:Class="sdkRawSensorDataCS.AccelerometerPage"

```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
```

xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

FontFamily="{StaticResource PhoneFontFamilyNormal}"

FontSize="{StaticResource PhoneFontSizeNormal}"

Foreground="{StaticResource PhoneForegroundBrush}"

SupportedOrientations="Portrait" Orientation="Portrait"

mc:Ignorable="d" d:DesignHeight="696" d:DesignWidth="480"

shell:SystemTray.IsVisible="True">

<!--İçeride kullanılacak bütün kontrol ve tasarım öğeleri bu gridin içerisinde barındırılacak -->

<Grid x:Name="LayoutRoot" Background="Transparent">

<!-- Grid tanımlaması yaparken düzenli bir yapı kurabilmek için satır tanımlamaları yapılabiliyor. -->

<Grid.RowDefinitions>

<!-- Auto seçildiğinde bu satırın içerisinde bulunan diğer öğelere göre yüksekliğini otomatik olarak ayarlıyor -->

<RowDefinition Height="Auto"/>

<!-- * sembolü ile Grid içerisinde bulunan diğer satırlara bakılarak bu satırın sayfanın geri kalanına yayılması sağlanıyor -->

<RowDefinition Height="*/"/>

</Grid.RowDefinitions>

<!-- StackPanel ile içerisinde eklenen öğelerin dikey veya yatay olarak dizilimi sağlanabiliyor. -->

<!-- Grid.Row=0 diyerek LayoutRoot Grid'inin ilk satırına bu StackPanel'in yerleştirilmesi sağlanıyor -->

<!-- Margin özelliği ile StackPanel'in içinde bulunduğu Grid satırındaki çerçeveden kendisinin uzaklaştırılması sağlanıyor. Bu sayede gridin başladığı yere yapışık olmak yerine ara biraz mesafe bırakılarak kullanıcı deneyimi tasarımına uygun hale getiriliyor -->

<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">

<!-- TextBlock metinsel değerler girilebilen bir kontroldür -->

<!-- Style içerisinde yazılan StaticResource ataması ile farklı bir yerde tanımlanan tasarım özellikleri çağırılıp bu TextBlock'a uygulanmaktadır. StaticResource tasarım atamalarında kullanılan genel yöntemlerden birisidir. HTML ve CSS bilginize var ise, bunu CSS olarak değerlendirebilirsiniz. -->

<TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"/>

<TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>

</StackPanel>

<!-- İç içe grid veya stackpanel kullanılabilir. LayoutRoot gridinin ilk satırına daha önce stackpanel eklemiştiğimiz. Şimdi ikinci satırına içerikleri ekleyeceğimiz yeni bir grid atıyoruz. -->

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<!-- StackPanele daha önce söylediğimiz gibi yatay ve dikey hizalama yaptırabiliyoruz. [Yapmamız](#) gereken sadece Orientation'ı Vertical (dikey) veya Horizontal (yatay) olarak belirlemek -->

```
<StackPanel Orientation="Vertical">
```

```
<StackPanel Orientation="Horizontal">
```

```
<TextBlock>status:</TextBlock>
```

<!-- Accelerometer durduruldu, başlatıldı, hata alındı vb. bilgilerin yazdırılacağı alan -->

```
<TextBlock Name="statusTextBlock"></TextBlock>
```

```
</StackPanel>
```

```
<StackPanel Orientation="Horizontal">
```

```
<TextBlock>time between updates:</TextBlock>
```

```
<TextBlock Name="timeBetweenUpdatesTextBlock"></TextBlock>
```

```
</StackPanel>
```

```
<TextBlock Text="acceleration:"></TextBlock>
```

```
<Grid>
```

```
<TextBlock Height="30" HorizontalAlignment="Left" Name="xTextBlock" Text="X: 1.0"
VerticalAlignment="Top" Foreground="Red" FontSize="28" FontWeight="Bold"/>
```

```
<TextBlock Height="30" HorizontalAlignment="Center" Name="yTextBlock" Text="Y: 1.0"
VerticalAlignment="Top" Foreground="Green" FontSize="28" FontWeight="Bold"/>
```

```
<TextBlock Height="30" HorizontalAlignment="Right" Name="zTextBlock" Text="Z: 1.0"
VerticalAlignment="Top" Foreground="Blue" FontSize="28" FontWeight="Bold"/>
```

```
</Grid>
```

```
<Grid Height="300">
```

<!-- çizgi çizmek için kullandığımız kontrol. Çizginin kalınlığını ve rengini Stroke ve StrokeThickness özellikleriyle tanımlayabiliyoruz. -->

```
<Line x:Name="xLine" X1="240" Y1="150" X2="340" Y2="150" Stroke="Red"
StrokeThickness="4"></Line>
```

```
<Line x:Name="yLine" X1="240" Y1="150" X2="240" Y2="50" Stroke="Green"
StrokeThickness="4"></Line>
```

```
<Line x:Name="zLine" X1="240" Y1="150" X2="190" Y2="200" Stroke="Blue"
StrokeThickness="4"></Line>
```

```
</Grid>
```

```
</StackPanel>
```

```
</Grid>
```

```
</Grid>
```

<!-- ApplicationBar ile içeride bulunan öğeler ile ilgili tekil veya çoğul işlemler yapabileceğimiz ayar tuşları barındırabilir veya uygulamanın kendisiyle ilgili genel ayarlara yönlendirmeler yapabiliriz. -->

```
<phone:PhoneApplicationPage.ApplicationBar>
```



```
<shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
```

```
<!-- Accelerometer özelliğini açıp kapatmak için arka taraftaki C# kodlarını çalıştıracak Button. -->
```

```
<shell:ApplicationBarIconButton IconUri="/ApplicationIcon.png" Text="on/off"  
Click="ApplicationBarIconButton_Click"/>
```

```
</shell:ApplicationBar>
```

```
</phone:PhoneApplicationPage.ApplicationBar>
```

```
</phone:PhoneApplicationPage>
```

