Name of Experiment: Compass or Magnetometer	**Exp No:** SSN2

Background: Student should have a basic knowledge of C#.

Summary: Compass or magnetometer sensor used to determine the angle by which the device is rotated relative to the Earth's magnetic north pole. It senses orientation relative to the Earth's magnetic field using the Hall Effect. This sensor is mainly use for auto rotation of digital map depending upon your physical orientation.

Learning Objective: Using this experiment student will get to know about the inbuilt Compass sensor in mobile and its capability to use in their own application as per the requirement. This experiment helps him/her to understand the proper working of the Compass and he/she is able to get the real time values generated by Compass.

Target Platforms: This experiment is tested on Nokia Lumia 800.
*Note:- Windows Phone Emulator does not support the Compass simulation.

Procedure:

Step 1. Repeat steps [1-4] as in experiment no MC1 [Refer Hello World].

Step 2. Design the UI part by drag and drop the control elements from toolbox on the design window in MainPage.xaml. [Refer Source Code section]

Step 3. Open the Main Page.xaml.cs, do some coding part.

Step 4. Add some reference by making right click on the project name in the Solution Explorer, as Microsoft.Xna.Framework, Microsoft.Devices.Sensors.

Step 5. Now, add some namespaces in the current project by writing using Microsoft.Devices.Sensors, Microsoft.Xna.Framework, using System, using System.Windows.Threading and using Microsoft.Phone.Controls.

Step 6. In the start button click event handler, start the compass by calling myCompass.Start() and add CurrentValueChanged event for myCompass object.

Step 7. Inside the MainPage constructor, we need to check whether the Compass is supported by the device or not? If yes, then set the time interval for which the updated values from the Compass will be displayed on the UI screen.

Step 8. Inside the CurrentValueChanged event, we get the compass reading through sensor reading argument **"e"** and assign these values to the variables created before the Main Page constructor.[Refer Source Code section]

Step 9. Call the Dispatcher.Tick event handler in order to bind the compass current reading with the UI control elements. [Refer Source Code section]

Step 10. Now, in similar fashion, stop the Compass by calling myCompass.Stop() inside the Stop button event handler.

Step 11. Reset the UI control elements inside the Reset button event handler.

Step 12. Save all changes made to the experiment by pressing Ctrl + S.

Step 13. Press F5, in order to debug the experiment on the Windows Phone Emulator.

***Note :- Compass is not supported by Windows Phone Emulator.**

Source Code	Comments
MainPage.xaml	

`<Grid x:Name="LayoutRoot" Background="Transparent">`
`<Grid.RowDefinitions>`
`<RowDefinition Height="Auto"/>`
`<RowDefinition Height="*"/>`
`</Grid.RowDefinitions>`
 	
`<!--TitlePanel contains the name of the application and page title-->`
`<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">`
`<TextBlock x:Name="ExperimentTitle" Text="ExpNo:SSN2" Style="{StaticResource PhoneTextNormalStyle}"/>`
`<TextBlock x:Name="ApplicationTitle" Text="Sensors" Style="{StaticResource PhoneTextNormalStyle}"/>`
`<TextBlock x:Name="PageTitle" Text="magnetometer" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>`
`</StackPanel>`
 	
`<!--ContentPanel - place additional content here-->`
`<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">`
`<StackPanel Orientation="Vertical">`
`<StackPanel Orientation="Horizontal">`
`<TextBlock Text="status:"/>`
`<TextBlock x:Name="txtStatus" Text=" " FontWeight="Bold" Foreground="Bisque" TextAlignment="Center"/>`
	← ExpNo:SSN2(Experiment Title)
	←Sensors (Application Title)
	←Magnetometer(Page Title)
	
	← Design the UI as per your convenience.

```
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="Compass
Orienatation:"/>
                    <TextBlock x:Name="txtOrientation"
Text=" " Foreground="Aquamarine"></TextBlock>
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="Magnetic
Heading:"/>
                    <TextBlock x:Name="txtMagnetic"
Text="" Foreground="Blue"></TextBlock>
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="Accuracy Level:"/>
                    <TextBlock x:Name="txtAccuracy"
Text=" " Foreground="Chocolate"></TextBlock>
                </StackPanel>
                <Grid Margin="20" Height="220"
Name="mygrid">
                    <Line x:Name="magneticLine"
X1="220" Y1="100" X2="220" Y2="0" Stroke="Yellow"
StrokeThickness="4" ></Line>
                    <Line x:Name="trueLine" X1="220"
Y1="100" X2="220" Y2="0" Stroke="Red"
StrokeThickness="4"></Line>
                </Grid>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="bStart"
Content="Start" BorderBrush="Chocolate"  Width="200"
Click="bStart_Click"/>
                    <Button x:Name="bStop"
Content="Stop" BorderBrush="Chocolate"
Click="bStop_Click" Width="200" />
                </StackPanel>
                <StackPanel Orientation="Horizontal"
HorizontalAlignment="Center">
                    <Button x:Name="bReset"
Content="Reset" BorderBrush="Chocolate"
Click="bReset_Click"/>
                </StackPanel>
            </StackPanel>

        </Grid>
    </Grid>
```

MainPage.xaml.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Controls;
using Microsoft.Xna.Framework;
using Microsoft.Devices.Sensors;
using System.Windows.Threading;

namespace Sensors_Compass
{
    public partial class Main2 : PhoneApplicationPage
    {
        Compass myCompass;
        DispatcherTimer timer;
        double magneticHeading;
```

← Add Reference for Microsoft.Xna.Framwork;
←Add Reference for Microsoft.Devices.Sensors;

```
        double accuracyHeading;
        double trueHeading;
        bool isDataValid=false;
        Vector3 rawMagnetometerReading;
        public Main2()
        {
            InitializeComponent();
    if (!Compass.IsSupported
  {
    txtStatus.Text = "Compass is not supported by the
Device";
    }
     else {
     timer = new DispatcherTimer();
     timer.Interval = TimeSpan.FromMilliseconds(60);
     timer.Tick+=new EventHandler(timer_Tick);

            }
        }

  private void timer_Tick(object sender, EventArgs e) {
   if (isDataValid) {
   txtStatus.Text = "receiving data from compass";
      }
  txtOrientation.Text = "";
  txtMagnetic.Text =magneticHeading.ToString("0.0");
  txtAccuracy.Text = accuracyHeading.ToString("0.0"); ;
  double centerX = mygrid.ActualWidth / 2.0;
  double centerY = mygrid.ActualHeight / 2.0;
  magneticLine.X2 = centerX - centerY *
Math.Sin(MathHelper.ToRadians((float)magneticHeading));
   magneticLine.Y2 = centerX - centerY *
Math.Sin(MathHelper.ToRadians((float)magneticHeading));
   trueLine.X2 = centerX - centerY *
Math.Sin(MathHelper.ToRadians((float)trueHeading));
   trueLine.Y2 = centerX - centerY *
Math.Sin(MathHelper.ToRadians((float)trueHeading));

        }
 private void bStart_Click(object sender,
RoutedEventArgs e)
   {
 if (myCompass == null) {
 myCompass = new Compass();
 myCompass.TimeBetweenUpdates =
 TimeSpan.FromMilliseconds(30);
 myCompass.CurrentValueChanged+=new
EventHandler<SensorReadingEventArgs
<CompassReading>>(myCompass_CurrentValueChanged);

                }
 try {
  txtStatus.Text="Starting Compass";
  myCompass.Start();
  timer.Start();}
  catch(InvalidOperationException){
  txtStatus.Text = "Unable to start compass.";
   }


        }
```

←Check whether Compass is supported by the device or not.

←Add time interval for updating the values from magnetometer.

←Event handler, for mapping the values coming from magnetometer.

← Supplying the values to the UI controls.

←Start button event handler

← Start the Compass Sensor

```
 private void myCompass_CurrentValueChanged(object
sender, SensorReadingEventArgs<CompassReading> e) {
magneticHeading = e.SensorReading.MagneticHeading;
accuracyHeading = e.SensorReading.HeadingAccuracy;
trueHeading = e.SensorReading.TrueHeading;
rawMagnetometerReading =
e.SensorReading.MagnetometerReading;
              }
 private void bStop_Click(object sender,
RoutedEventArgs e)
        {
if (myCompass != null && myCompass.IsDataValid)
     {
        myCompass.Stop();
        timer.Stop();
       txtStatus.Text = "Compass Stopped";
            }
  }

 private void bReset_Click(object sender,
RoutedEventArgs e)
 {
   txtAccuracy.Text = string.Empty;
   txtMagnetic.Text = string.Empty;
   txtStatus.Text = string.Empty;
magneticLine.X1 = 220; magneticLine.X2 = 220;
magneticLine.Y1 = 100; magneticLine.Y2 = 0;
trueLine.X1 = 220;
 trueLine.X2 = 220;
 trueLine.Y1 = 100;
trueLine.Y2 = 0;

        }
    }
}
```

←Fetching the Values, from the Compass

← Stop button event handler

←Stop the Compass

←Reset button event handler

← Set the Initial position and Values of the UI controls.

Screenshots:

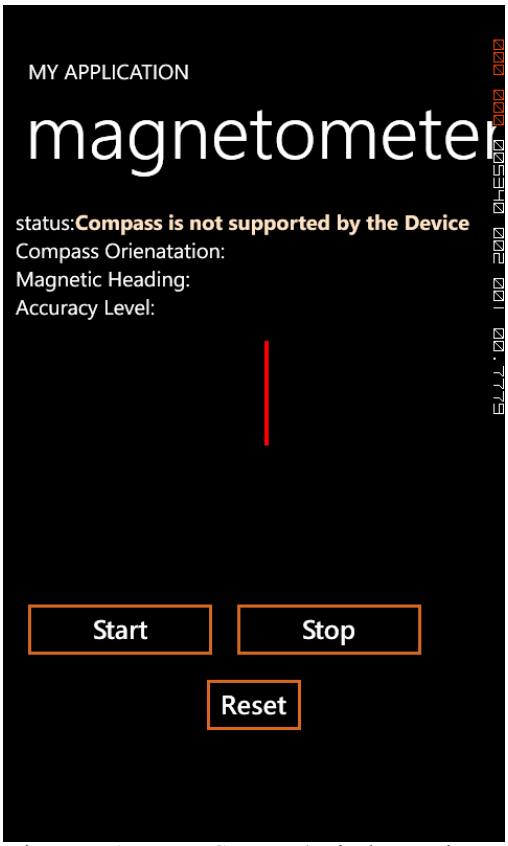

Fig. No. 1 Home Screen (Windows Phone Emulator)

Observations:

It is observed by the developer that Compass or Magnetometer, can be used in variety of application**.** This experiment shows the way, how smartly we can introduce the Compass services in other areas.