Name of Experiment: GSM Parameter –Cell-ID	**Exp No:** MC10

Background: Student should have basic knowledge of C#.

Summary: Location based services are very popular services among the mobile users in today's world. These services are mainly based on the Assisted GPS (A-GPS) technology, But majority of people are restricted to use these services on the fly because of lack of GPS transceiver in their mobile phones. We have tried to overcome this limitation by using a third party API's. These API's required some parameter like Mobile Country Code (mcc), Mobile Network Code (mnc), Local Area Code (lac) in order to fetch the servicing Cell-ID and its geo-coordinates.

Learning Objective: Through this experiment we want to get the Cell-ID and Neighbouring Cell-ID of the intended Mobile Station. We also can exploit the geo-location of the mobile device without GPS.
***Note: -** Here, we have passed some static values of mcc, mnc and lac as parameter to the third party API's. If we are allowed to use the GSM telephony Library of the Windows Phone then we can get the exact geo-location of the mobile device on which this experiment is running.

Target Platforms: This experiment is tested on Windows Phone Emulator and Nokia Lumia 800.

Third Party API's:

```
1. http://www.opencellid.org/cell/get?&mcc=404&mnc=54&cellid=381&lac=43&fmt=txt
2. http://www.opencellid.org/cell/getMeasures?key=YourKey
&mcc=404&mnc=54&lac=43&cellid=381&fmt=xml
3. http://www.opencellid.org/cell/getInArea?&BBOX&
limit=10&mcc=404&mnc=54&lac=43&fmt=xml
```

***Note:** You need to register for the KEY at www.opencellid.org.

Procedure:

Step1. Repeat the steps [1-4] as in experiment MC1 [refer experiment "Hello World"].

Step2. Create string builder for containing some textual information regarding the experiment inside the MainPage constructor. [refer source code section]

Step3. Add new item in the project as Windows Phone Portrait Page, and named it as Signal.xaml.

Step4. Inside the button1_Click handler in the MainPage.xaml.cs, call the navigation service in order to navigate to the new Signal.xaml page that we have created in previous step.

Step5. Now design the UI part for the Signal.xaml.[refer source code section]

Step6. For the tab "getCellID" we have bGet_Click event handler, inside this we need to create a WebClient that will forward the request using third party API's asynchronously. There we have DownloadStringCompleted event to fetch the result.[refer source code section]

Step7. Get the API's by registering to the website www.opencellid.org.

Step8. Add a new class Cell.cs and add the getter and setter method.[refer source code section]

Step9. For the tab "getMeasure" we have bMeasure_Click event handler, inside this we need to create a

Webclient that will forward our request to the opendatabase server through the third party API's asynchronously. We have DownloadStringCompleted event to fetch the result.[refer source code section]

Step10. Add a new class CellMeasure.cs and add the getter and setter method for the same.[refer source code section]

Step11. We also need to parse the result as the response is in xml format. The response should be allocate to the list in order to display on the UI.[refer source code section]

Step12. Now, repeat the same procedure for the tab "getArea". [refer source code section]

Step13. Add a new class Area.cs and add the getter and setter method for the same.

Step14. Save all the changes made to the project by pressing Ctrl+ S or Save icon on the palette.

Step15. Press F5 to debug the experiment in the Windows Phone Emulator.

Step16. By this way, we are able to deploy this experiment on the Windows Phone Emulator.

Source Code	Comments
MainPage.xaml `<!--LayoutRoot is the root grid where all page content is placed--> <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="Exp No MC10" TextAlignment="Right" Style="{StaticResource PhoneTextNormalStyle}"/> <TextBlock x:Name="ApplicationTitle" Text="Mobile Computing" Style="{StaticResource PhoneTextNormalStyle}"/> <TextBlock x:Name="PageTitle" Text="Gsm Parameters" 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"> <Button Content="Go" Click="button1_Click" Height="72" HorizontalAlignment="Left" Margin="138,457,0,0" Name="button1" VerticalAlignment="Top" Width="160" /> <TextBlock Height="350" HorizontalAlignment="Left" Margin="12,36,0,0" Name="textBlock1" Text="TextBlock" VerticalAlignment="Top" Width="438" TextWrapping="Wrap" FontFamily="Segoe WP Bold" FontSize="22" /> </Grid> </Grid>`	← Exp No MC10 (Experiment Title) ←Mobile Computing (Application Computing) ← Gsm Parameters (Page Title)

MainPage.xaml.cs

```
using System.Windows;
using Microsoft.Phone.Controls;
using System;
using System.Text;

namespace GSM_Parameter
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
 string s = "A GSM Cell ID (CID) is a generally unique number
used to identify each Base transceiver station (BTS) or sector of
a BTS within a Location area code (LAC).";
            StringBuilder output = new StringBuilder();
            output.AppendLine(s);
            output.AppendLine(" ");
            output.AppendLine(" ");
            output.AppendLine("Some important Annotations are:");
            output.AppendLine(" ");
            output.AppendLine("mcc: Mobile Country Code");
            output.AppendLine("mnc: Mobile Network Code");
            output.AppendLine("lac: Local Area Code");
            textBlock1.Text = output.ToString();
        }

        private void button1_Click(object sender, RoutedEventArgs
e)
        {
            this.NavigationService.Navigate(new
Uri("/Signal.xaml", UriKind.Relative));
        }
    }
}
```

← Add new Windows Phone Portrait Page Signal.xaml

Signal.xaml

```
<!--LayoutRoot is the root grid where all page content is placed-
->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <!--Pivot Control-->
        <controls:Pivot Title="GSM Parameter">
            <!--Pivot item one-->
            <controls:PivotItem Header="get CellId">
                <StackPanel >
                    <TextBlock Text="Get the position of Cell-Id
in terms of longitude and latitude where the input parameters are
mcc=404,mnc=54,cellid=381 and lac=43." Height="187" Width="380"
TextAlignment="Left" TextWrapping="Wrap" Margin="12,22,0,0"
FontSize="22"  />
   <Button x:Name="bGet" Content="Go"  Click="bGet_Click"
Height="70" Width="150" Margin="12,48,0,0"/>
   <TextBlock Text="Result" FontSize="24" FontWeight="Bold"
Height="50" Width="100" Margin="12,10,306,0"/>
   <TextBlock x:Name="txtRes" Height="200" Width="400"
TextAlignment="Center" TextWrapping="Wrap"/>
                </StackPanel>
```

← get CellId tab

```
            </controls:PivotItem>

            <!--Pivot item two-->
            <controls:PivotItem Header="get Measures">
                <StackPanel>
                    <TextBlock Text="Get the raw measures used to
compute the position of cell(i.e when the data  has been updated
on the server) where the input parameters are mcc=404,
mnc=54,lac=43 and cellid=381." Height="200" Width="430"
TextAlignment="Left" Margin="12,22,0,0" TextWrapping="Wrap"/>
                    <Button x:Name="bMeasure"
Click="bMeasure_Click" Content="Get Measure" Height="70"
Width="250" Margin="12,22,0,0"/>
                    <TextBlock Text="Result" FontSize="24"
FontWeight="Bold" Height="50" Width="100" Margin="12,22,286,0"/>
                    <StackPanel x:Name="stk" Margin="12,0,0,0"
Height="220" ScrollViewer.VerticalScrollBarVisibility="Visible"
Orientation="Horizontal">
  <ListBox x:Name="listbox">
   <ListBox.ItemTemplate>
      <DataTemplate>
        <StackPanel Margin="10">
        <TextBlock Text="{Binding Latitude}"/>
        <TextBlock Text="{Binding Longitude}"/>
        <TextBlock Text="{Binding DateOn}"/>
         <TextBlock Text="{Binding Sample}"/>
              </StackPanel>
              </DataTemplate>
               </ListBox.ItemTemplate>

                 </ListBox>
                  </StackPanel>
                    <!--<TextBlock x:Name="txtMe" Height="200"
Width="400" TextAlignment="Left" TextWrapping="Wrap"
ScrollViewer.VerticalScrollBarVisibility="Visible"/>-->
                </StackPanel>
            </controls:PivotItem>
            <!--Pivot item three-->
            <controls:PivotItem Header="getInArea">
                <StackPanel>
                    <TextBlock Text="Obtain the list of cells in
specified area alonwith their latitude and longitude coordinates
where input parameters are mcc=404, mnc=54,lac=43. "
TextWrapping="Wrap" Height="150" Width="390" Margin="12,23,0,0"/>
                    <Button x:Name="bGetArea"
Click="bGetArea_Click" Content="Get Area" Height="77" Width="200"
Margin="12,0,0,0"/>
                    <TextBlock Text="Result" FontWeight="Bold"
Height="50" Width="100" Margin="12,0,296,0"/>
                    <StackPanel Height="300"
Orientation="Horizontal"
ScrollViewer.VerticalScrollBarVisibility="Visible">
    <ListBox x:Name="myList2">
     <ListBox.ItemTemplate>
     <DataTemplate>
     <StackPanel Margin="10">
    <TextBlock Text="{Binding Latitude}"/>
    <TextBlock Text="{Binding Longitude}"/>
    <TextBlock Text="{Binding CellId}"/>
    <TextBlock Text="{Binding MCC}"/>
    <TextBlock Text="{Binding MNC}"/>
    <TextBlock Text="{Binding LAC}"/>
```

← get Measures tab

← get InArea

```
    <TextBlock Text="{Binding Samples}"/>
     </StackPanel>
      </DataTemplate>
        </ListBox.ItemTemplate>
          </ListBox>
           </StackPanel>
             </StackPanel>
            </controls:PivotItem>
        </controls:Pivot>

    </Grid>
```

Signal.xaml.cs

```
using System;
using System.Linq;
using System.Net;
using System.Windows;
using Microsoft.Phone.Controls;
using System.Diagnostics;
using System.Xml.Linq;
using System.Collections.ObjectModel;

namespace GSM_Parameter
{


    public partial class Signal : PhoneApplicationPage
    {
        public ObservableCollection<Area> Collection { get; set;
}

        public Signal()
        {
            InitializeComponent();


       }
    // This vent is used to get the Cell-ID
location in terms of longitude and latitude
     private void bGet_Click(object sender, RoutedEventArgs e)
 {
    try
     {
       WebClient c = new WebClient();
      c.DownloadStringAsync(new
Uri("http://www.opencellid.org/cell/get?
&mcc=404&mnc=54&cellid=381&lac=43&fmt=txt"));
      c.DownloadStringCompleted += new
 DownloadStringCompletedEventHandler(c_DownloadStringCompleted);
            }
            catch(Exception ex){
                MessageBox.Show(ex.Message);
            }
        }
public void c_DownloadStringCompleted
(object sender,DownloadStringCompletedEventArgs e) {
     String res = e.Result;
     txtRes.Text = res;
     Debug.WriteLine(res);
        }
```

```
 //This event is used to get the raw
measurement at which timestamp the data has been updated at the
server end.
  private void bMeasure_Click(object sender, RoutedEventArgs e)
        {
   try
       {
        WebClient b = new WebClient();
 b.DownloadStringAsync(new
Uri("http://www.opencellid.org/cell/getMeasures?
key=581bf75a811e719d07529a0982349ea4&mcc=404&mnc=54
&lac=43&cellid=381&fmt=xml"));
   b.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(b_DownloadStringCompleted);
     }
     catch (Exception ex){
   MessageBox.Show(ex.Message);
         }
     }
 public void b_DownloadStringCompleted
(object sender, DownloadStringCompletedEventArgs e) {


    if (e.Result == null || e.Error != null)
           {
               MessageBox.Show("Resource is unable to reach");
           }
    else {
   XElement myMeasure = XElement.Parse(e.Result);
   var list = (from query in myMeasure.Descendants("measure")
   select new CellMeasure
  {
    Latitude = (string)query.Attribute("lat").Value,
    Longitude = (string)query.Attribute("lon").Value,
   DateOn = (string)query.Attribute("takenOn").Value,
   Sample = (string)query.Attribute("takenBy").Value
    }).ToList();
   ObservableCollection<CellMeasure> cellValues =
 new ObservableCollection<CellMeasure>(list);
     listbox.ItemsSource = cellValues;

           }

        }
 // This event is related to get the list of cells
 in the specified area along with their geo-Coordinates and Cell-
ID's.
 private void bGetArea_Click(object sender, RoutedEventArgs e)
  {
    try
     {
      WebClient web = new WebClient();
 web.DownloadStringAsync(new
Uri("http://www.opencellid.org/cell/getInArea?
&BBOX&limit=10&mcc=404&mnc=54&lac=43&fmt=xml"));
web.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(web_DownloadStringCompleted);
            }
     catch(Exception ex){
     MessageBox.Show(ex.Message);
            }
```

```
        }
 public void web_DownloadStringCompleted(object sender,
DownloadStringCompletedEventArgs e) {
   string result = e.Result;
    if (e.Error != null)
        {
    MessageBox.Show("Resource is Unable to reach");
      }
    else
      {
   XElement myElement = XElement.Parse(e.Result);
  var myList = (from query in myElement.Descendants("cell")
  select new Area
       {
 Latitude = (string)query.Attribute("lat").Value,
 Longitude = (string)query.Attribute("lon").Value,
 Samples = (string)query.Attribute("nbSamples").Value,
 MCC = (string)query.Attribute("mcc").Value,
 MNC = (string)query.Attribute("mnc").Value,
 LAC = (string)query.Attribute("lac").Value,
CellId = (string)query.Attribute("cellId").Value
        }).ToList();
   ObservableCollection<Area> myArea = new
ObservableCollection<Area>(myList);
   myList2.ItemsSource = myArea;

           }

   Debug.WriteLine(result);
        }

    }
}
```

Cell.cs

```
using System.Collections.ObjectModel;
using System.Xml.Serialization;

namespace GSM_Parameter
{
    [XmlRoot("rsp")]
    public class Cell
    {
        [XmlArray("cell")]
        [XmlArrayItem("measure")]
  public ObservableCollection<CellMeasure>
 Collection { get; set; }
    }
}
```

← getter and setter for cellId

Measure.cs

```
namespace GSM_Parameter
{
    public class CellMeasure
    {
    public string Latitude{ get; set;}
    public string Longitude{get;set;}
    public string DateOn{get;set ;}
```

← getter and setter for Measure

```
    public string Sample {get;set;}
    }
}
```

Area.cs

```
namespace GSM_Parameter
{
    public class Area
    {
        public string Latitude { get; set; }
        public string Longitude { get; set; }
        public string CellId { get; set; }
        public string Samples { get; set; }
        public string MCC { get; set; }
        public string MNC { get; set; }
        public string LAC { get; set; }


    }
}
```

← getter and setter for Area

Screenshots

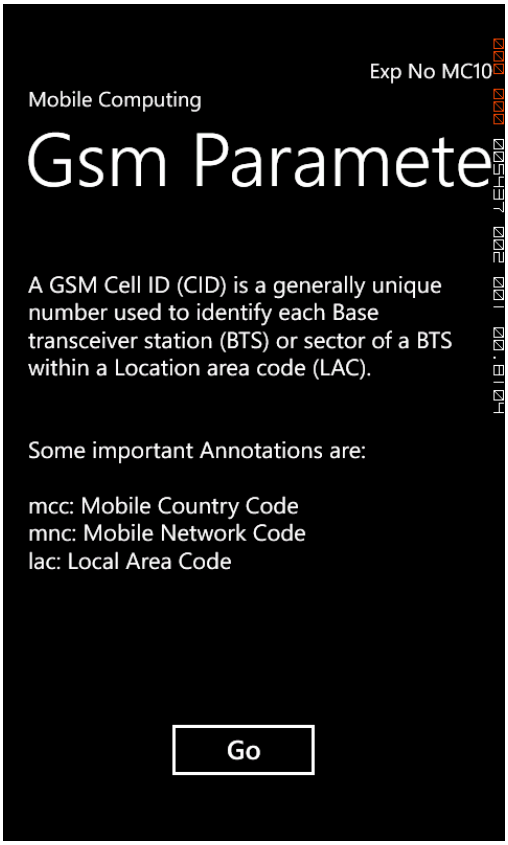

Fig. No 1 Home Screen

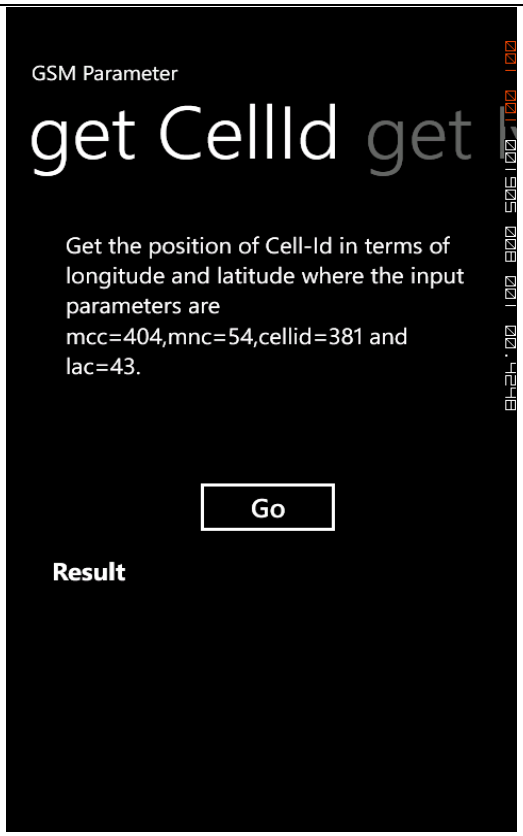

Fig. No 2 getCellId

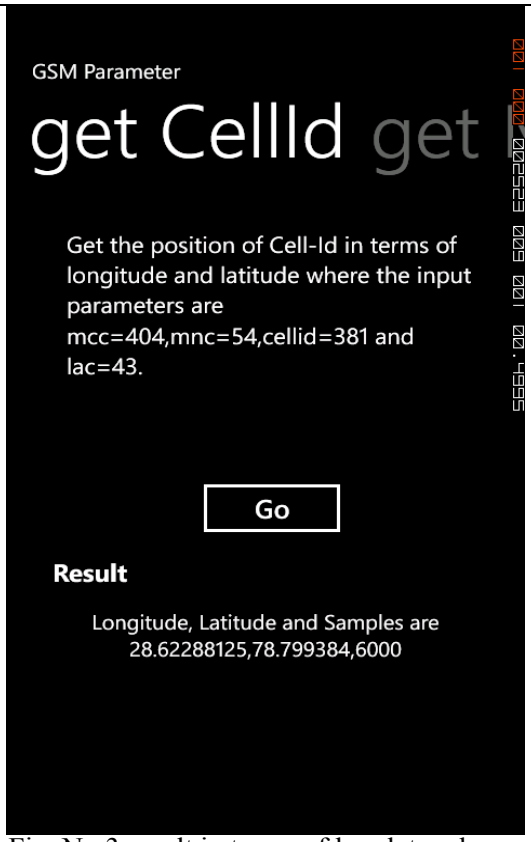

Fig. No 3 result in terms of lon, lat and samples

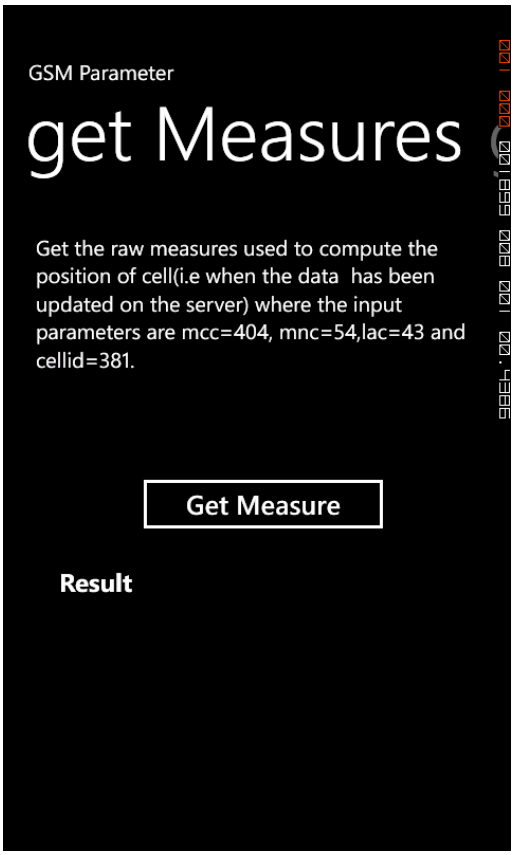

Fig. No 4 get Measure

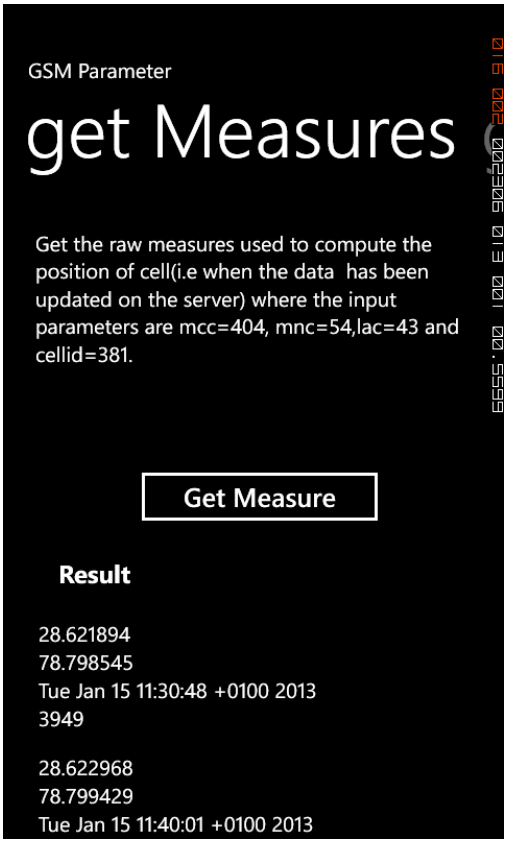

Fig. No 5 result in terms of lon, lat timestamp and cellid

Fig. No 6 getInArea

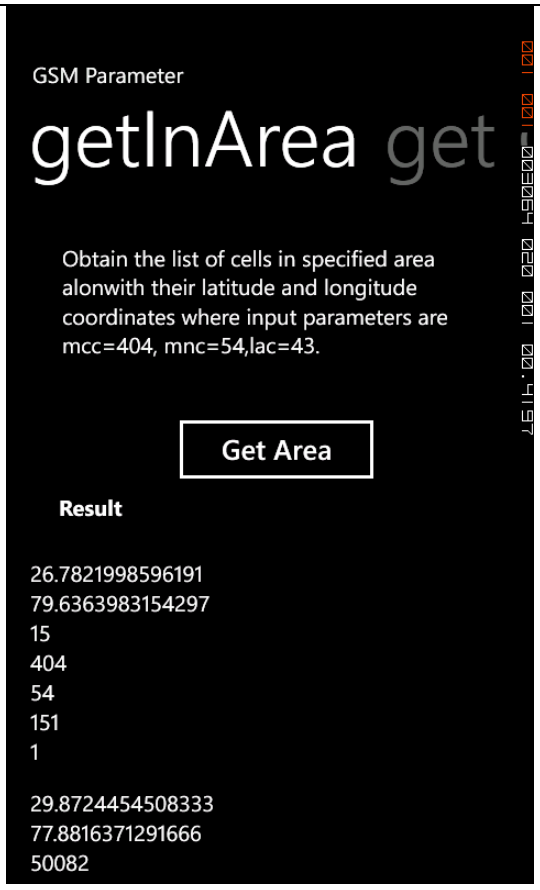

Fig. no 7 result in terms of lon, lat ,cellId, sample, mcc, mnc and lac .

Observation: It is observed that if we are able to get the library for the GSM module, then we are able to extract the information related to SIM module i.e. mcc, mnc, lac as well as the cellId.