How to access and manage the Microphone raw data in WP

Introduction

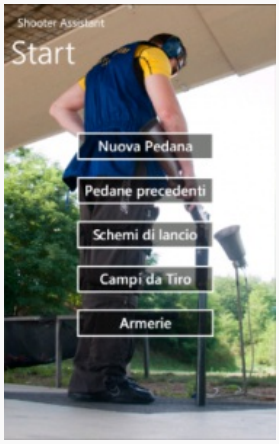
This article shows how to manage **microphone raw data** in Windows Phone, focusing on volume management. The accompanying example project is a (work in progress) real application: **Shooter Assistant** which assists shooters during training for specialities including: Olympic Trap, Skeet and Double Trap.

Shooter Assistant app

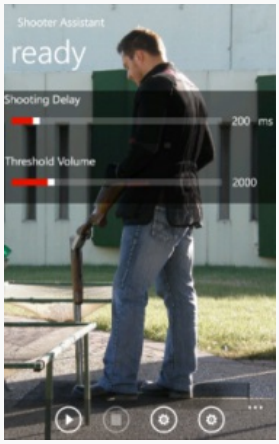
Olympic Trap shooting is one of the three major forms of competitive clay shooting, generally shot with a 12 gauge double barreled shotgun. The shooter waits with the gun to shoulder and charged with two shots. When ready, the shooter shouts "PULL" and a clay target ("plate") is immediately launched from a trap machine mounted to their left or right.

For best results, the shooter must fire between 5 and 6 tenths of a second after shouting "PULL" - the precise time varies because the target can be launched with a 45° variation in angle from either left or right of the shooter and with an output speed of between 100 - 120 km/h. Reaction time is the key of success as results can heavily influenced being **1 tenths** faster or slower.

The Shooter Assistant app improves performance by training the user to consistently fire within the optimal tenth of a second, and by recording their reaction time for later analysis. It does this by recording when PULL is called and playing a beep after 0.5 seconds (the approximate point when the shooter should fire). It also records when the gun is fired and uses **genetic algorithms** to match itself to your reaction time.



Home screen



Record Screen

Launch Schemes Screen

Creating the project

- Install Windows Phone SDK
- Create a new project by selecting the File | New Project menu command.
- The New Project window will be displayed. Expand the Visual C# templates, and then select the Silverlight for Windows Phone templates.
- Select the Windows Phone Application template. Fill in the project name as desired. {Note|Don't call it simply "Microphone"}
- In the Solution Explorer, right-click References and choose Add Reference.
- Choose Microsoft.Xna.Framework from the list of .NET components and click the OK button.
- If you see a dialog that warns about adding a reference to a Silverlight assembly, click the Yes button.

Add the following using statements to the top of your MainPage.xaml.cs file:

```
using System.IO;
using System.Windows.Threading;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
```

In **MainPage.xaml.cs**, declare variables as global members of your MainPage class:

```
public partial class MainPage : PhoneApplicationPage
{

        Microphone microphone = Microphone.Default;
        byte[] buffer;
        MemoryStream stream = new MemoryStream();
        SoundEffect sound;
```


(C) Copyright Nokia 2013. All rights reserved.

```
        // Constructor
        public MainPage()
        {
```

Add the following code to the constructor of your MainPage class after the call to **InitializeComponent**

```
// Contructor
public MainPage()
{
            InitializeComponent();

           DispatcherTimer dt = new DispatcherTimer();
            dt.Interval = TimeSpan.FromMilliseconds(50);
            dt.Tick += delegate { try { FrameworkDispatcher.Update(); } catch { } };
            dt.Start();

            microphone.BufferReady += new EventHandler<EventArgs>(microphone_BufferReady);
}
```

Now add the following method to manage Microphone **raw data**

```
void microphone_BufferReady(object sender, EventArgs e)
{
       microphone.GetData(buffer);
       // If you want to store the audio data in a stream
       stream.Write(buffer, 0, buffer.Length);
}
```

Detecting Volume changes. The RMS Method.

Declare a global variable in your MainPage class to set **threshold**:

```
public partial class MainPage : PhoneApplicationPage
{
        Microphone microphone = Microphone.Default;
        byte[] buffer;
        MemoryStream stream = new MemoryStream();
        SoundEffect sound;
        ....
        private int minimumThreshold = 500;
        ....
```

```
void microphone_BufferReady(object sender, EventArgs e)
{
            // Retrieve audio data
            microphone.GetData(buffer);

            // RMS Method
            double rms = 0;
            ushort byte1 = 0;
            ushort byte2 = 0;
            short value = 0;
            int volume = 0;
            rms = (short)(byte1 | (byte2 << 8));

            for (int i = 0; i < buffer.Length - 1; i += 2)
            {
                byte1 = buffer[i];
                byte2 = buffer[i + 1];

                value = (short)(byte1 | (byte2 << 8));
                rms += Math.Pow(value, 2);
            }
            rms /= (double)(buffer.Length / 2);
            volume = (int)Math.Floor(Math.Sqrt(rms));

            if ((volume > minimumThreshold))
            {
                System.Diagnostics.Debug.WriteLine("Threshold exceeded");
                System.Diagnostics.Debug.WriteLine("buffer.Length" + buffer.Length + " Volume:" + volume);
            }
}
```

Start Recording

```
private void recordButton_Click(object sender, RoutedEventArgs e)
{
            if (microphone.State == MicrophoneState.Stopped)
            {
                microphone.BufferDuration = TimeSpan.FromMilliseconds(1000);
                buffer = new byte[microphone.GetSampleSizeInBytes(microphone.BufferDuration)];
                microphone.Start();
                System.Diagnostics.Debug.WriteLine("Threshold setted to:" + minimumThreshold);
            }
}
```

Stop Recording

```
private void stopButton_Click(object sender, RoutedEventArgs e)
{
        if (microphone.State == MicrophoneState.Started)
        {
            microphone.Stop();
        }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Related Links

- Using QMicrophone - Symbian platform

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.