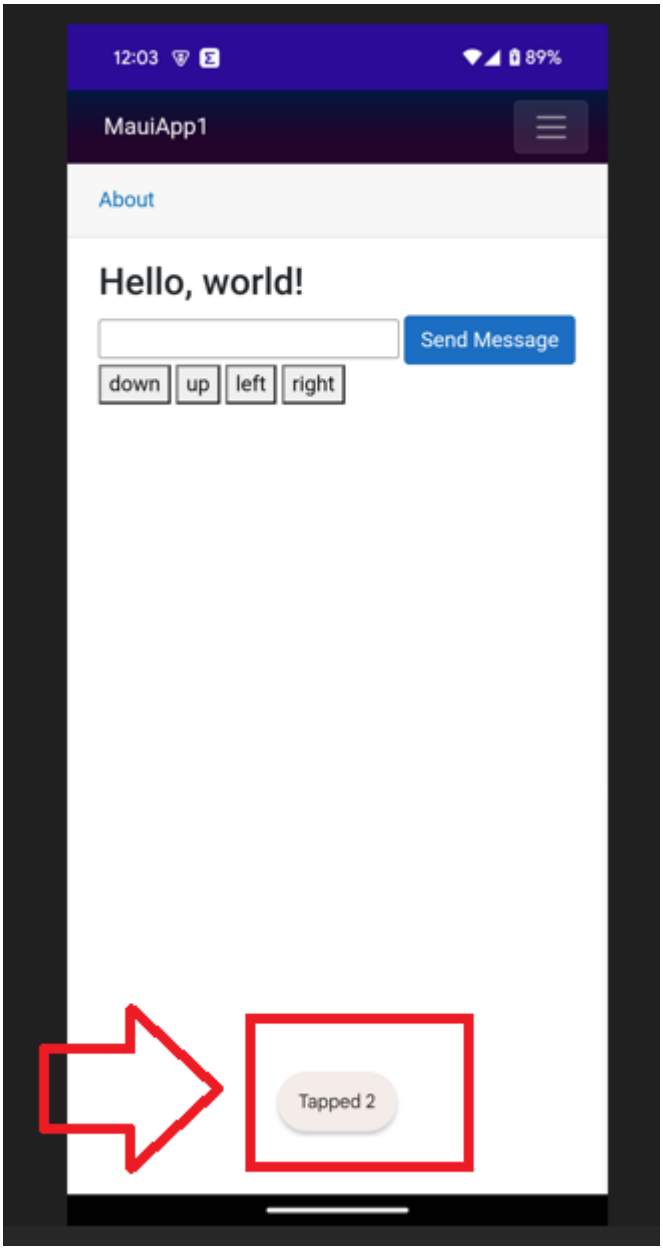
.NET MAUI - Handling taps

Now that we can send text and images, we can handle events.

We'll continue from the last project. But this one is quite straightforward now, so see the base example .NET MAUI Integration: First contact

Android will show a notification of how many times we tapped the glasses:



To "communicate it back to MAUI/.NET" - I will have to expand the SDK, and will write up the usage after this is done.

So lets get to it!

1. Locate you **MainActivity.cs** and let the compiler know we're implementing **Com.Vuzix.Ultralite.IEventListener**

```
public class MainActivity : MauiAppCompatActivity, Com.Vuzix.Ultralite.IEventListener
```

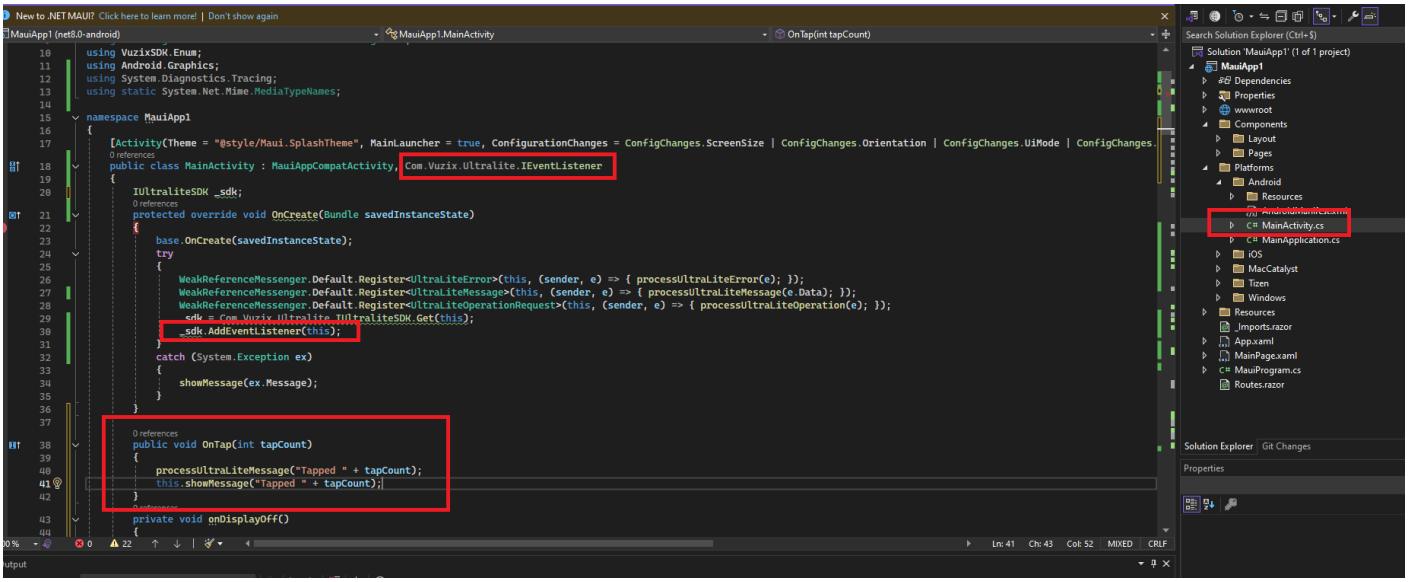
2. In the Oncreate, let the SDK know we're handling the events

```
        IUltraliteSDK _sdk;
    protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            try
            {
                WeakReferenceMessenger.Default.Register<UltraLiteError>(this, (sender, e) => {
processUltraLiteError(e); });
                WeakReferenceMessenger.Default.Register<UltraLiteMessage>(this, (sender, e) => {
processUltraLiteMessage(e.Data); });
                WeakReferenceMessenger.Default.Register<UltraLiteOperationRequest>(this, (sender, e)
=> { processUltraLiteOperation(e); });
                _sdk = Com.Vuzix.Ultralite.IUltraliteSDK.Get(this);
        _sdk.AddEventListener(this); // <- this line we added
    }
    catch (System.Exception ex)
            {
                showMessage(ex.Message);
            }
        }
```

3. We now create the functions of the known events we'll handle (these are the complete list)

```
    public void OnTap(int tapCount)
    {
        processUltraLiteMessage("Tapped " + tapCount);
        this.showMessage("Tapped " + tapCount);
    }
    private void onDisplayOff()
    {
        this.showMessage("onDisplayOff");
    }
    private void onDisplayOn()
    {
        this.showMessage("onDisplayOn");
    }
    private void onDisplayTimeout()
    {
        this.showMessage("onDisplayTimeout");
    }
    private void OnPowerButtonPress(bool turningOn)
    {
        this.showMessage("onPowerButtonPress, turnongOn:" + turningOn);
    }
    private void OnScrolled(bool isScreenEmpty)
    {
        this.showMessage("OnScrolled, isScreenEmpty:" + isScreenEmpty);
    }
```

In your Visual Studio 2022 it will look like this:



Your full **MainActivity** might look among these lines:

```
using Android.App;
using Android.Content.PM;
using Android.OS;
using Android.Widget;
using CommunityToolkit.Mvvm.Messaging;
using VuzixSDK.Class;
using Com.Vuzix.Ultralite;
using Layout = Com.Vuzix.Ultralite.Layout;
using TextAlignment = Com.Vuzix.Ultralite.TextAlignment;
using VuzixSDK.Enum;
using Android.Graphics;
using System.Diagnostics.Tracing;
using static System.Net.Mime.MediaTypeNames;

namespace MauiApp1
{
    [Activity(Theme = "@style/Maui.SplashTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout |
ConfigChanges.SmallestScreenSize | ConfigChanges.Density)]
    public class MainActivity : MauiAppCompatActivity, Com.Vuzix.Ultralite.IEventListener
{
        IUltraliteSDK _sdk;
protected override void OnCreate(Bundle savedInstanceState)
        {
```

```
        base.OnCreate(savedInstanceState);
        try
        {
            WeakReferenceMessenger.Default.Register<UltraLiteError>(this, (sender, e) => {
processUltraLiteError(e); });
            WeakReferenceMessenger.Default.Register<UltraLiteMessage>(this, (sender, e) => {
processUltraLiteMessage(e.Data); });
            WeakReferenceMessenger.Default.Register<UltraLiteOperationRequest>(this, (sender, e) => {
processUltraLiteOperation(e); });
            _sdk = Com.Vuzix.Ultralite.IUltraliteSDK.Get(this);
□□□□_sdk.AddEventListener(this);
□□□}
□□□catch (System.Exception ex)
        {
            showMessage(ex.Message);
        }
    }

□□public void OnTap(int tapCount)
□□{
□□□processUltraLiteMessage("Tapped " + tapCount);
□□□this.showMessage("Tapped " + tapCount);
□□}
□□private void onDisplayOff()
□□{
□□□this.showMessage("onDisplayOff");
□□}
□□private void onDisplayOn()
□□{
□□□this.showMessage("onDisplayOn");
□□}
□□private void onDisplayTimeout()
□□{
□□□this.showMessage("onDisplayTimeout");
□□}
□□private void OnPowerButtonPress(bool turningOn)
□□{
□□□this.showMessage("onPowerButtonPress, turnongOn:" + turningOn);
□□}
□□private void OnScrolled(bool isScreenEmpty)
```

```
□□{
□□□this.showMessage("OnScrolled, isScreenEmpty:" + isScreenEmpty);
□□}
□□public void clearScreen()
□□{
□□□_sdk.Canvas.RemoveText(_lastTextId);
□□□_sdk.Canvas.RemoveAnimation(_lastAnimationId);
□□□_sdk.Canvas.RemoveImage(_lastImageId);
□□}
□□public void showMessage(string message)
        {
            MainThread.BeginInvokeOnMainThread(() =>
            {
                var toast = Toast.MakeText(this, message, ToastLength.Short);
                toast.Show();
            });
        }
        protected void processUltraLiteError(UltraLiteError error)
        {
            if (_sdk.IsConnected)
            {
                if (!_sdk.IsControlledByMe)
                {
                    _sdk.RequestControl();
                }
                if (_sdk.IsControlledByMe)
                {
                    string _title = $"[Error]{(error.Source != null ? " " + error.Source : "")}";
                    string _error = (error.Exception != null ? $"Exception : {error.Exception.Message}" : "Error
occured");
                    _sdk.SendNotification(_title, _error);
                }
            }
        }
        int _lastTextId = -1;
        int _lastImageId = -1;
        int _lastAnimationId = -1;

□□protected void processUltraLiteMessage(String message)
□□{
```

```
□□□if (_sdk.IsConnected)
□□□{
□□□□if (!_sdk.IsControlledByMe)
□□□□{
□□□□□_sdk.RequestControl();
□□□□}
□□□□if (_sdk.IsControlledByMe)
□□□□{
□□□□□_sdk.SetLayout(Layout.Canvas, 0, true);
□□□□□bool _messageSucceeded = false;
□□□□□if (_lastTextId >= 0)
□□□□□{
□□□□□□_messageSucceeded = _sdk.Canvas.UpdateText(_lastTextId, message);
□□□□□}
□□□□□else
□□□□□{
□□□□□□_lastTextId = _sdk.Canvas.CreateText(message, Anchor.Center);
□□□□□□_messageSucceeded = (_lastTextId != -1);
□□□□□}
□□□□□if (!_messageSucceeded)
□□□□□{
□□□□□□showMessage("Text failed");
□□□□□}
□□□□□_sdk.Canvas.Commit();
□□□□□SystemClock.Sleep(1000);
□□□□}
□□□}
□□}
□□protected void processUltraLiteOperation(UltraLiteOperationRequest Request)
      {
         if (_sdk.IsConnected)
         {
            if (!_sdk.IsControlledByMe)
            {
               _sdk.RequestControl();
            }
            if (_sdk.IsControlledByMe)
            {
               _sdk.SetLayout(Layout.Canvas, 0, true);
               if (Request.Operation == eUltraLiteOperation.ShowImage && Request.ImageBitMap != null)
```

```
                {
                    LVGLImage image = loadLVGLImage(Request.ImageBitMap);
                    bool _imageSucceeded = false;
                    if (_lastImageId >= 0)
                    {
□□□□□□□_imageSucceeded = _sdk.Canvas.UpdateImage(_lastImageId, image);
                    }
                    else
                    {
                        _lastAnimationId = _sdk.Canvas.CreateImage(image, Anchor.Center);
□□□□□□□_imageSucceeded = (_lastImageId != -1);
□□□□□□}
                    if(!_imageSucceeded) showMessage("Image failed");
                    _sdk.Canvas.Commit();
                }
                if (Request.Operation == eUltraLiteOperation.ShowAnimation && Request.AnimationBitMap != null)
                {
                    LVGLImage[] image = loadLVGLImage(Request.AnimationBitMap);
                    int _animationDelay = 500;
                    if(_lastAnimationId >= 0)
                    {
                        _sdk.Canvas.RemoveAnimation(_lastAnimationId);
□□□□□□}
□□□□□□_lastAnimationId = _sdk.Canvas.CreateAnimation(image, Anchor.Center, _animationDelay);

□□□□□□if (_lastAnimationId == -1)
                    {
                        showMessage("Animation failed");
                    }
                    _sdk.Canvas.Commit();
                }
            }

        }
        else
        {
            showMessage("SDK is not connected");
        }
    }
    private static Bitmap loadBitmap(byte[] bitmapbytes)
```

```
        {
            BitmapFactory.Options options = new BitmapFactory.Options();

            // https://proandroiddev.com/image-decoding-bitmaps-android-c039790ee07e
            options.InSampleSize = 2;

            //options.InTargetDensity = 640 * 2;
            //options.InTargetDensity = 480 * 8;
            //options.InScaled = true;
            options.InPreferredConfig = Bitmap.Config.Argb8888;
            /* options.InMutable = true;

            options.InSampleSize = 8;
□□□options.OutWidth = 600;
□□□options.OutHeight = 400;
□□□options.InScaled = scaled;*/
            Bitmap bmp = BitmapFactory.DecodeByteArray(bitmapbytes, 0, bitmapbytes.Length, options);

            return bmp;// resize(bmp, 640, 480);
        }
        private static LVGLImage[] loadLVGLImage(List<byte[]> images)
        {
            List<LVGLImage> _images = new List<LVGLImage>();
            foreach (var image in images)
            {
                _images.Add(LVGLImage.FromBitmap(loadBitmap(image), LVGLImage.CfIndexed1Bit));
            }
            return _images.ToArray();
        }
        private static LVGLImage loadLVGLImage(byte[] image)
        {
            //ColorObject[] _colors = { LVGLImage.IColorMapper.White, LVGLImage.IColorMapper.Mid };
            //LVGLImage _img = new LVGLImage(LVGLImage.CfIndexed1Bit, 480, 640, _colors, image);
            LVGLImage _img2 = LVGLImage.FromBitmap(loadBitmap(image), LVGLImage.CfIndexed1Bit);
            return _img2;
        }
    }
}
```

When the code is run on your Android device - and you tap the side of the glasses, it will show a little message.
Vuzix's SDK mentions that 1, 2 and 3 taps are triggered.

Revision #3
Created 5 October 2024 07:54:30 by Tim
Updated 7 October 2024 08:36:28 by Tim