Windows 10 IoT Core

Internet of Things Den Extended Device with FEZ HAT Lab Guide

Document Version 3.1 NDC

Social	**Twitter** #makerden #iot #raspberrypi #windows10 #azure
Document Authors	Dave Glover	dglover@microsoft.com	@dglover Andrew Coates |acoat@microsoft.com |@coatsy
Document location	https://github.com/MakerDen/Maker-Den-Documentation-and-Resources-FezHat
Source Code Location	https://github.com/MakerDen/Maker-Den-Windows-IoT-Core-FEZ-HAT
Disclaimer	All care has been taken to ensure the accuracy of this document. No liability accepted.
Copyright	You are free to reuse and modify this document and associated software.

INTRODUCTION

The goal of the Maker Den is to familiarise you with some of the components and technologies associated with the Internet of Things (IoT).

GETTING STARTED

If you are setting up your own Maker Den then all source code and documentation is available at https://github.com/MakerDen/Maker-Den-Documentation-and-Resources-FezHat.

SPREAD THE WORD

Be sure to spread the word about the Maker Den on Twitter. Use hash tags #makerden #iot #raspberrypi #windows10 #azure

LAB HARDWARE

The following components are used for the Maker Den.

|	
Raspberry Pi 2 These labs are built on the Raspberry Pi running Windows 10 IoT Core. You can find out more about Windows 10 IoT Core at http://dev.windows.com/iot.	
GHI electronics FEZ HAT The FEZ HAT Key Features: • On-Board Analog Input and PWM chips. • Two DC Motor Drivers, suitable for building small robots. • Terminal Blocks for wiring in DC motors without the need for soldering. • Two Servo Motor Connections. • Two Multi Color LEDs, connected to PWM for thousands of colors. • Single Red LED. • Temperature Sensor. • Accelerometer. • Light Sensor. • Two user buttons. • Terminal block with 2x Analog, 2x Digital I/O, 2x PWM and power. • Female headers with SPI, I2C, 3x Analog, 3x PWM. • Dedicated power input for driving the servo motors and DC motors. • No Soldering required, completely assembled and tested. **Developer Guide** **https://www.ghielectronics.com/docs/329/fez-hat-developers-guide**	
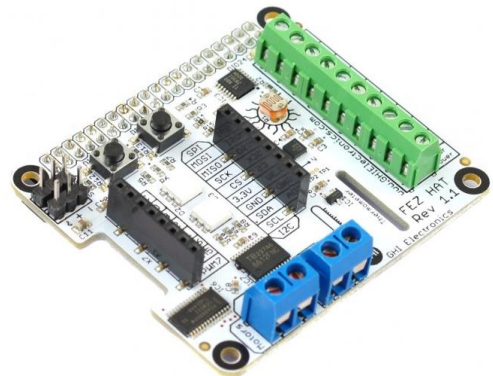
RESET THE LAB

⊠ **STEP 1: Ensure Visual Studio is closed.**

⊠ **STEP 2: Double click the ResetLabs.bat file on your desktop. This will copy the source code from a GitHub repository and launch Visual Studio with the solution opened.**
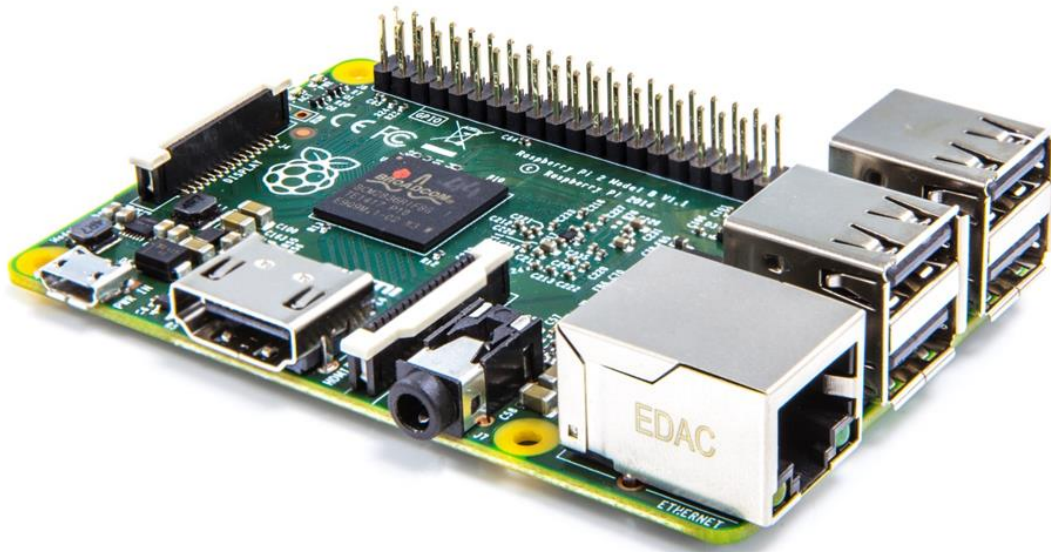
EXPERIMENTS

⊠ All the source code can be referenced from the Source Code folder on the Desktop.

⊠ This user guide can be found in the Documents folder on the Desktop.

⊠ Be sure to check out the Windows 10 IoT Core Doc, Tutorials and Samples. There is a link to this page in the Desktop Documents folder.

⊠ For the self-sufficient adventurous types, you can reference the Windows 10 IoT Core Doc, Tutorials and Samples and the GHI Electronics FEZ HAT developer resources for more information.

Section 1

**Windows IoT Core development with Visual Studio**

Connecting your device

EXPERIMENT 1: CONNECTING AND CONFIGURING YOUR DEVICE

The Raspberry Pi should be connected to the development PC via a wired Ethernet connection. This connection is used both for deployment and debugging as well as passing through internet requests from the Raspberry Pi when Internet Connection Sharing is enabled on the PC.

⊠ **STEP 1:** Press the Windows key and type "Windows 10 IoT Core Dashboard"[1] and run.

⊠ **STEP 2:** Go to My devices[2] and right mouse click your device and select **Open in Device Portal**.
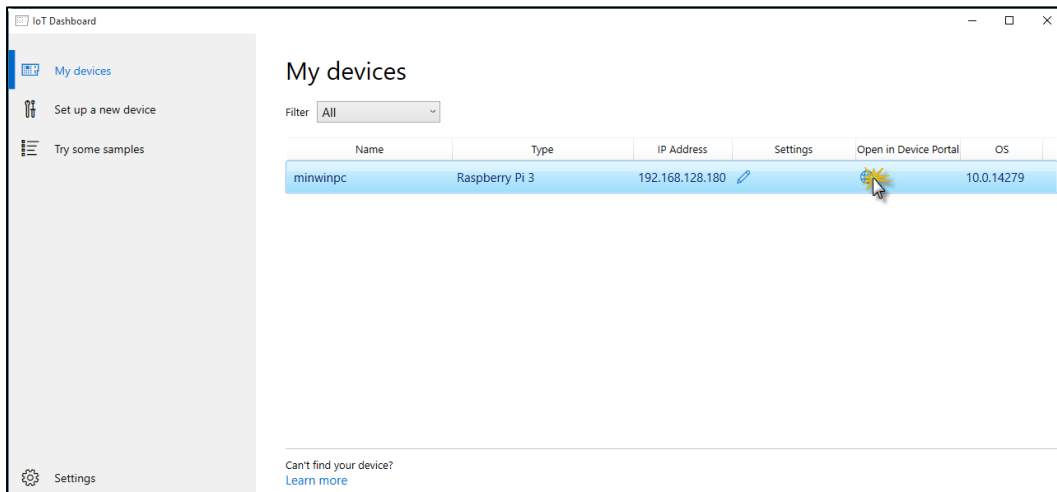
If your device does not show up in the list it is almost certainly because the network connection between your PC and the Raspberry Pi is public and Device Discovery is not enabled. See How to change Windows 10 network location from Public to Private.

Alternatively, navigate to the default device url http://minwinpc:8080.

---

[1] You can download the Windows 10 IoT Core Dashboard from https://developer.microsoft.com/en-us/windows/iot/getstarted

[2] You can right mouse click a device for more options including copying the device IP Address, Name, and to start a PowerShell session.

⊠ STEP 3: Authenticate. The default credentials are Username: *Administrator* and Password: *p@ssw0rd*
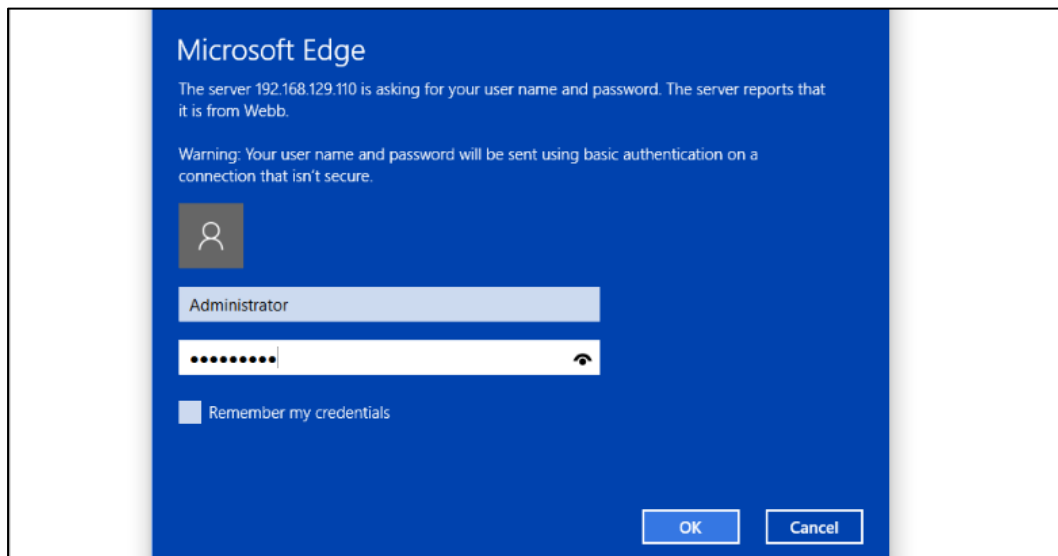
Windows Device Portal will launch and display the web management home screen!
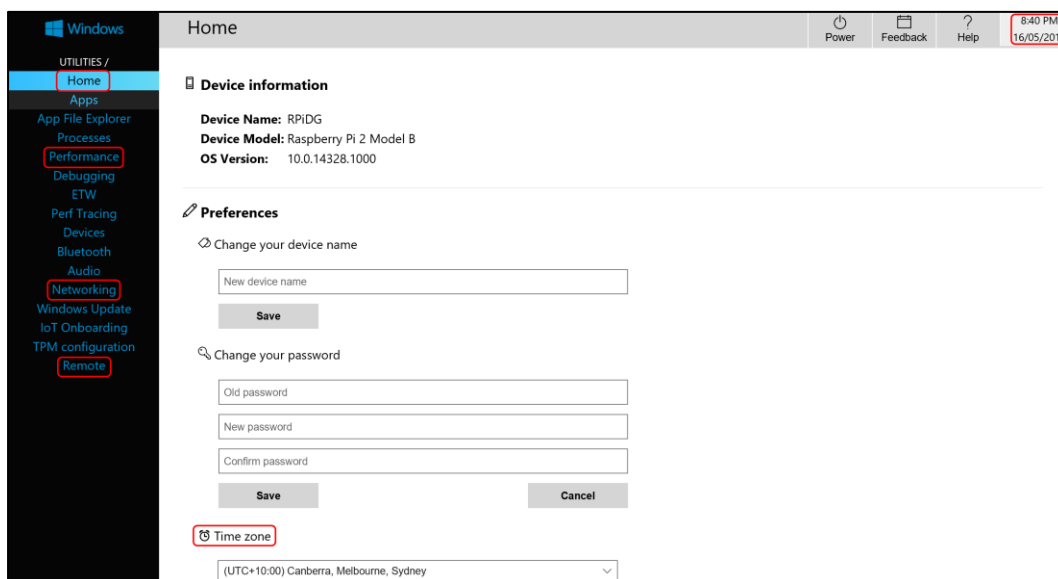
⊠ STEP 4: Verify Device Configuration

- From the **Home** Tab verify the time zone, date and time are correct. If the device has the incorrect date or time, then refer to the troubleshooting section in the appendix.

- From the **Remote** tab verify that **Windows IoT Remote Server**[3] is enabled. If it is not, then enable it.
- Take a moment to explore the other tabs in the Windows Device Portal.

⊠ **STEP 5:** Test the Windows IoT Remote Client connection.

Press the Windows key and type "**Windows IoT Remote Client**"[4] and run.

⊠ **STEP 6:** Select your device from the dropdown list.

Depending on the network setup you may need to enter the IP address of your Raspberry Pi. Get the address of the device from the **Windows 10 IoT Core Dashboard**.

This will take a moment to connect. When it does you will see the video output of the Raspberry Pi remoted to your desktop.
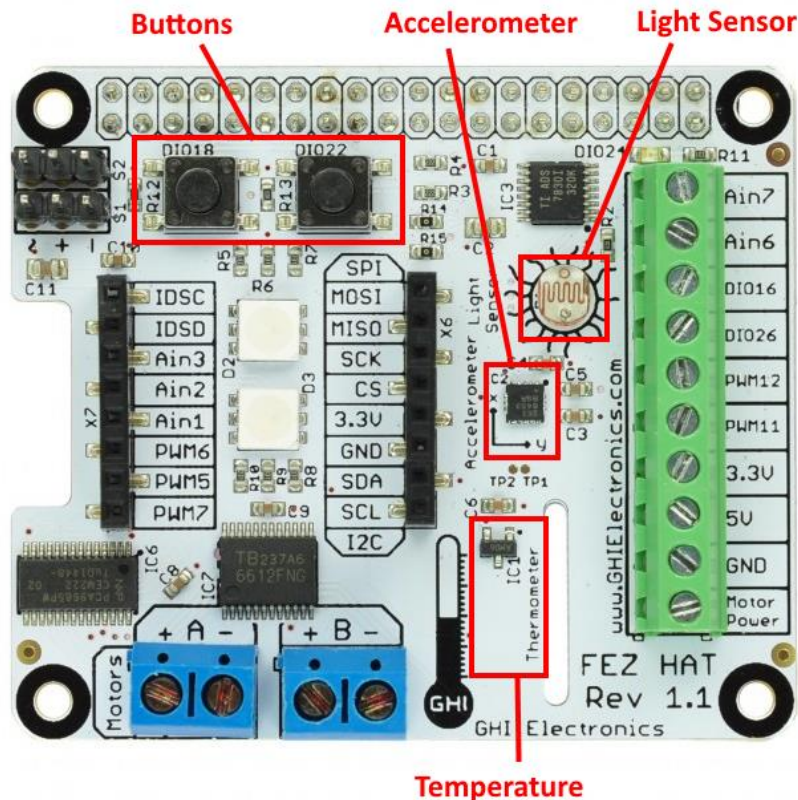
Minimise the remote client application when you have verified that it is working.

---
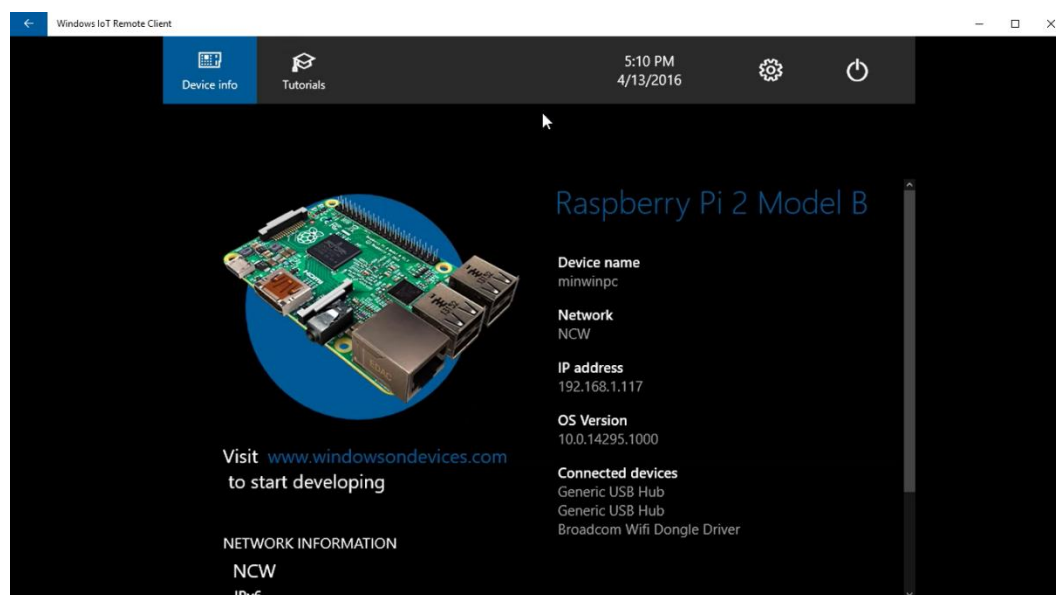
[3] The Windows IoT Remote Server does take additional CPU cycles on the Raspberry Pi so depending on what you are doing you may want to disable the Windows Remote Server from the Windows Device Portal.

[4] The Windows IoT Remote Client is available from the Windows Store.

Section 2

Windows IoT Core development with Visual Studio

Deploying your first "**Headless**" app

Sensing ambient light levels

Debugging

EXPERIMENT 2: HELLO WORLD

There are two styles of applications you can run on Windows IoT Core – Headed and Headless.

This section focuses on "Headless" apps, these apps don't have a user interface, the app runs in the background.

⊠ **STEP 1**: Expand the **MakerDenFezHAT** project then double click the **StartupTask.cs** file to open it. You may need to expand the code regions by clicking the highlighted **+** symbol on the left hand side.
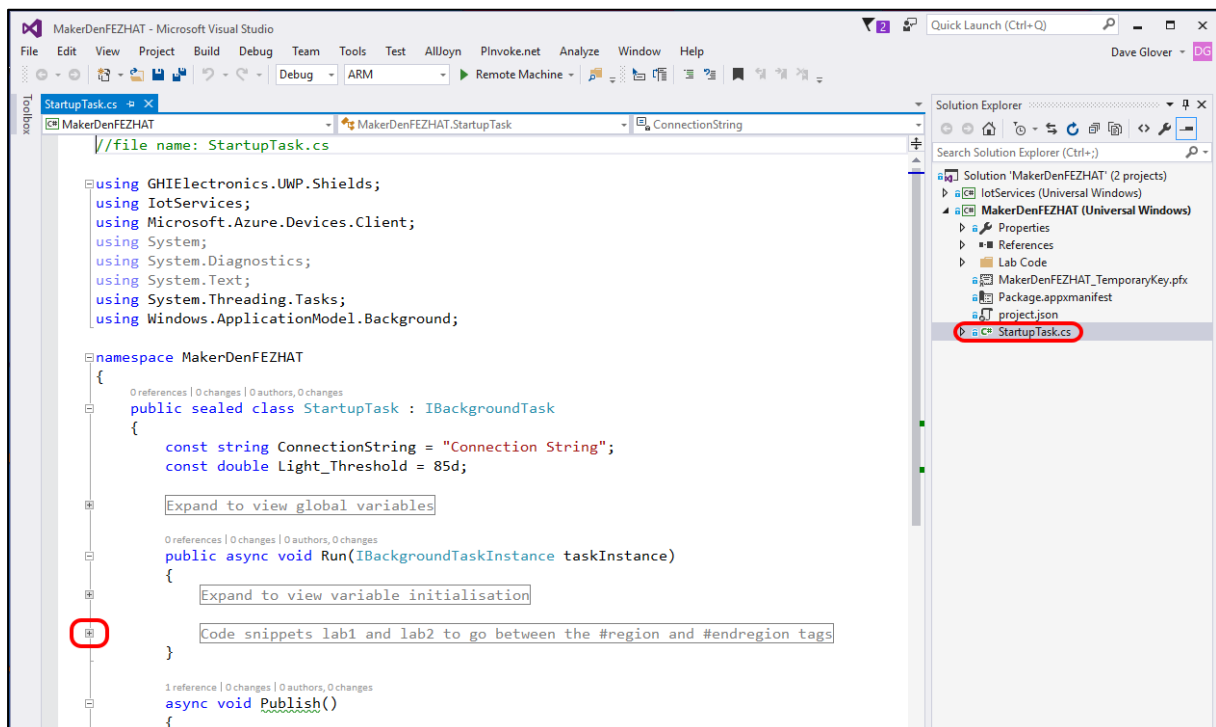
⊠ **STEP 2:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**
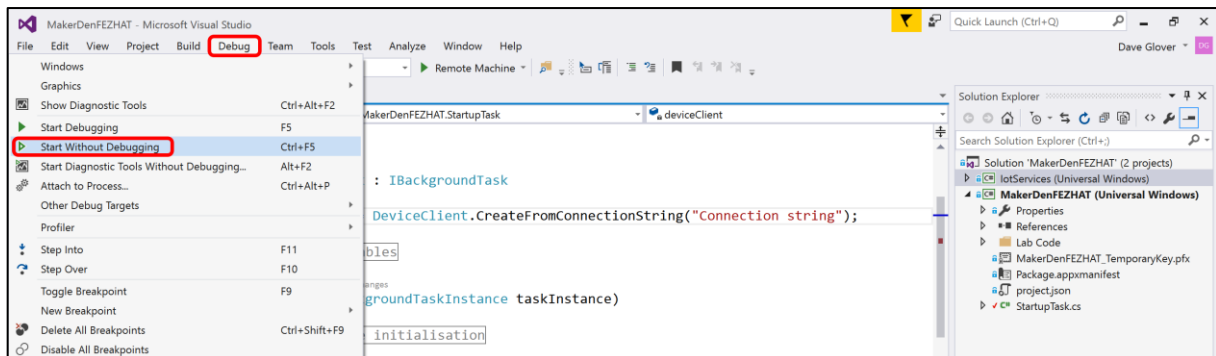
⊠ **STEP 3:** Check that Visual Studio has successfully compiled and deployed the code by looking at the output window and the status bar. It will take approximately 30 to 60 seconds to deploy.
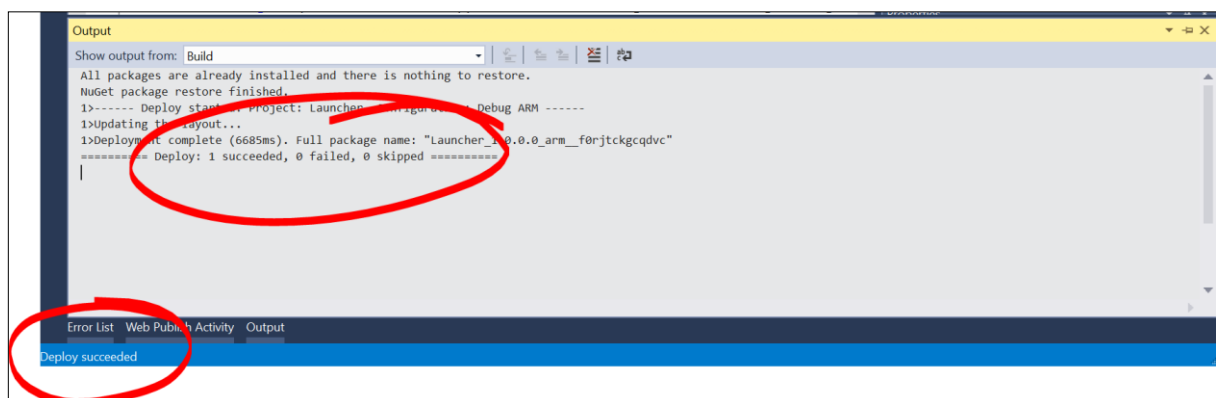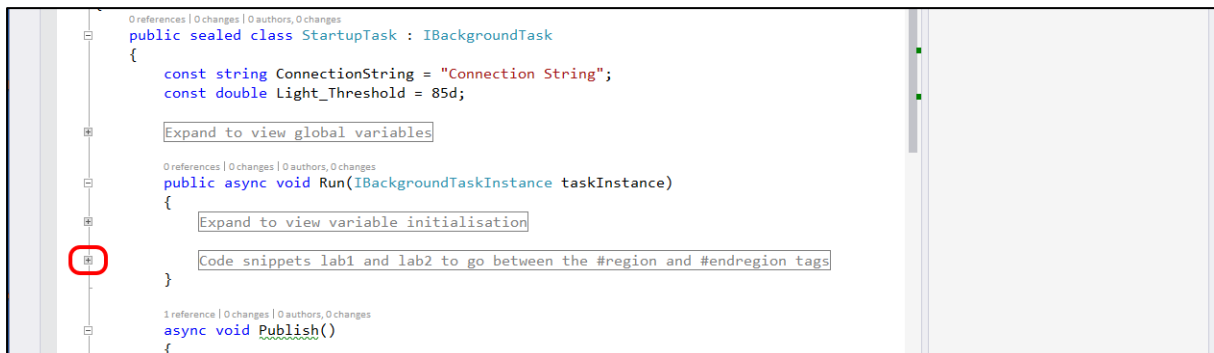

⊠ **STEP 4:** Check the LEDs on the FEZ HAT. You should see an LED alternating between Red, Green and Blue.

⊠ **STEP 5:** Pat yourself on the back, you did it☺

EXPERIMENT 3: SENSING THE WORLD

This lab reads the ambient light levels from the light sensor.

⊠ **STEP 1:** Review the code in the **StartupTask.cs** file. Look for the **#region** and **#endregion** tags. You may need to expand the code regions by clicking the highlighted **+** symbol on the left hand side.

⊠ **STEP 2:** Delete the code circled in red **inside** the #region tags.

⊠ **STEP 3:** Ensure the cursor is between the **#region** tags, then type the following code.

```csharp
while (true)
{
    var level = hat.GetLightLevel() * 100;

    if (level > Light_Threshold)
    {
        hat.D2.Color = FEZHAT.Color.Blue;
    }
    else
    {
        hat.D2.Color = FEZHAT.Color.Red;
    }

    await Task.Delay(500);
}
```

⊠ **STEP 4:** Your "StartupTask.cs" file should like look like the following. If not, **Ctrl+Z** and try again.

```
//file name: StartupTask.cs

using GHIElectronics.UWP.Shields;
using IotServices;
using Microsoft.Azure.Devices.Client;
using System;
using System.Diagnostics;
using System.Text;
using System.Threading.Tasks;
using Windows.ApplicationModel.Background;

namespace MakerDenFEZHAT
{
    public sealed class StartupTask : IBackgroundTask
    {
        const string ConnectionString = "Connection String";
        const double Light_Threshold = 85d;

        Expand to view global variables

        public async void Run(IBackgroundTaskInstance taskInstance)
        {
            Expand to view variable initialisation

            #region Code snippets lab2 and lab3 to go between the #region and #endregion tags

            while (true)
            {
                var level = hat.GetLightLevel() * 100;

                if (level > Light_Threshold)
                {
                    hat.D2.Color = FEZHAT.Color.Blue;
                }
                else
                {
                    hat.D2.Color = FEZHAT.Color.Red;
                }

                await Task.Delay(500);
            }

            #endregion
        }

        async void Publish()
        {
            #region Snippet lab6 - Publish to Azure IoT Hub

            #endregion
        }

        private void Commanding_CommandReceived(object sender, CommandEventArgs<string> e)
        {
            #region Snippet lab9 - IoT Hub Command Support

            #endregion
        }
    }
}
```
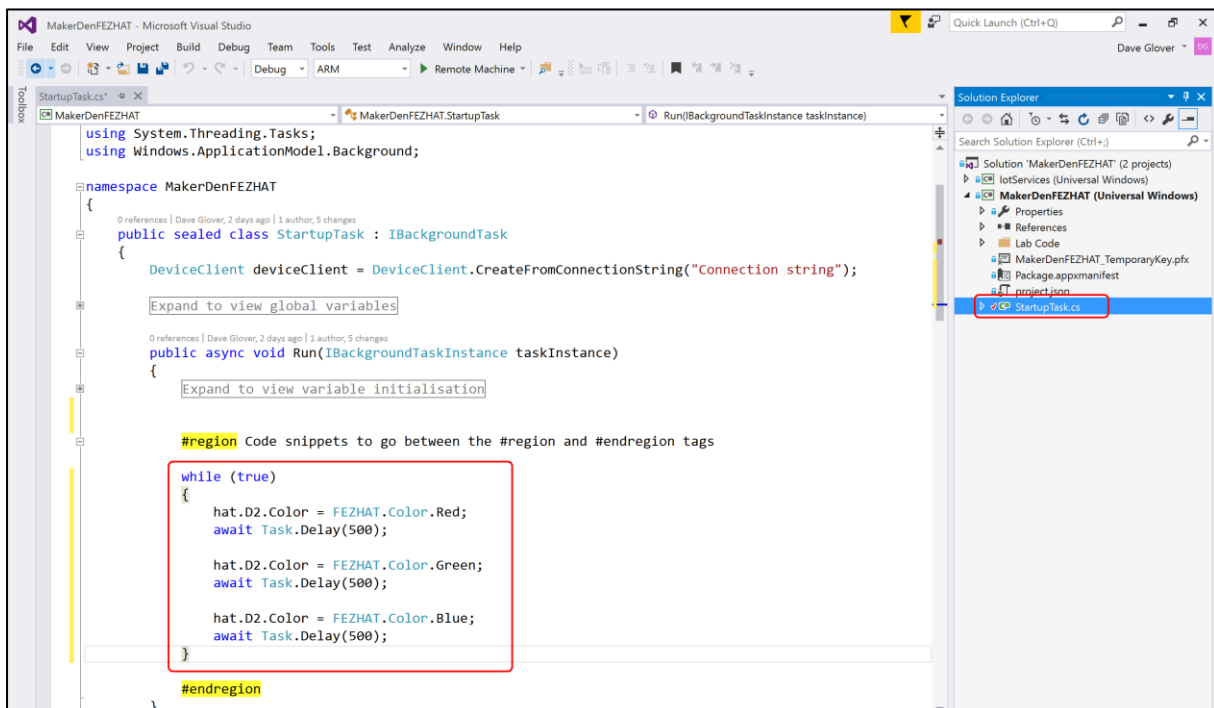
⊠ **STEP 5:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**

⊠ **STEP 6:** Hover your hand over the light sensor and observe the LED alternate between blue and red depending on the ambient light levels.

EXPERIMENT 4: REMOTE DEBUGGING

⊠ **STEP 1:** Next, set a break point to see how easy it is to debug directly on the device. This is a unique capability provided by Visual Studio and Windows IoT Core.

- Right-click on the line that **reads `if (level > Light_Threshold)`**
- Choose Breakpoint, then Insert Breakpoint.
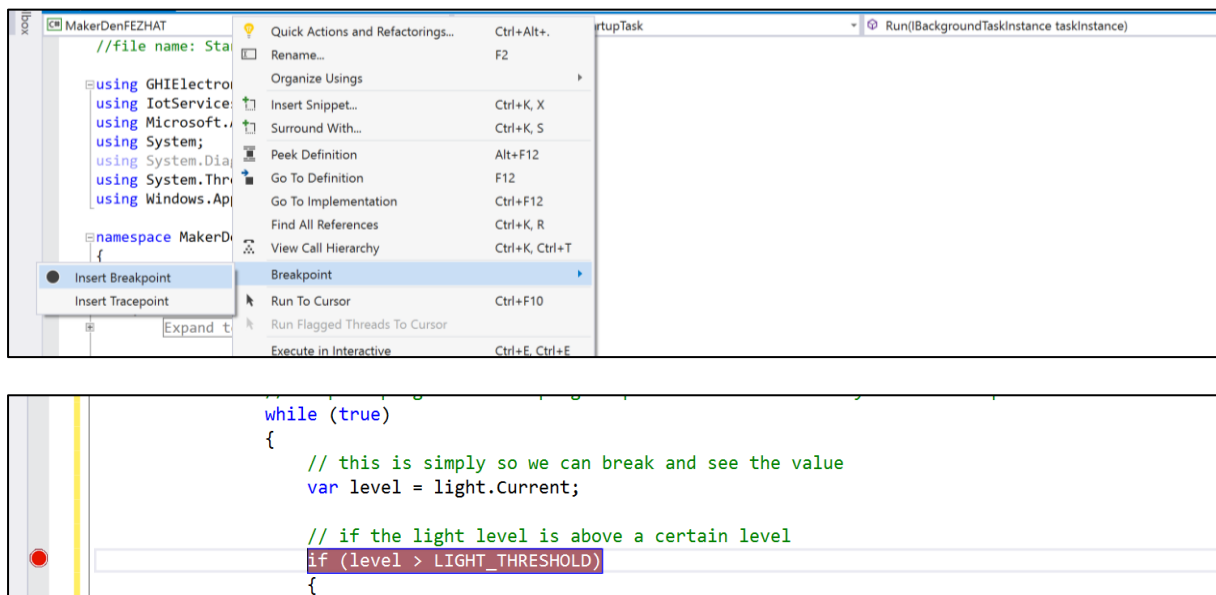
⊠ **STEP 2:** From the **Debug** menu select **Start Debugging** or on the keyboard press **F5** and wait for the solution to deploy and for Visual Studio to hit the breakpoint.

⊠ **STEP 3:** Hover the cursor over the variable "level" and Visual Studio will display its current value.

⊠ **STEP 4:** While holding your hand over the light sensor, press F5 a couple of times to continue and observe the LED change colour depending on ambient light levels.

⊠ **Step 5**: Press Shift-F5 to stop debugging.

Section 3
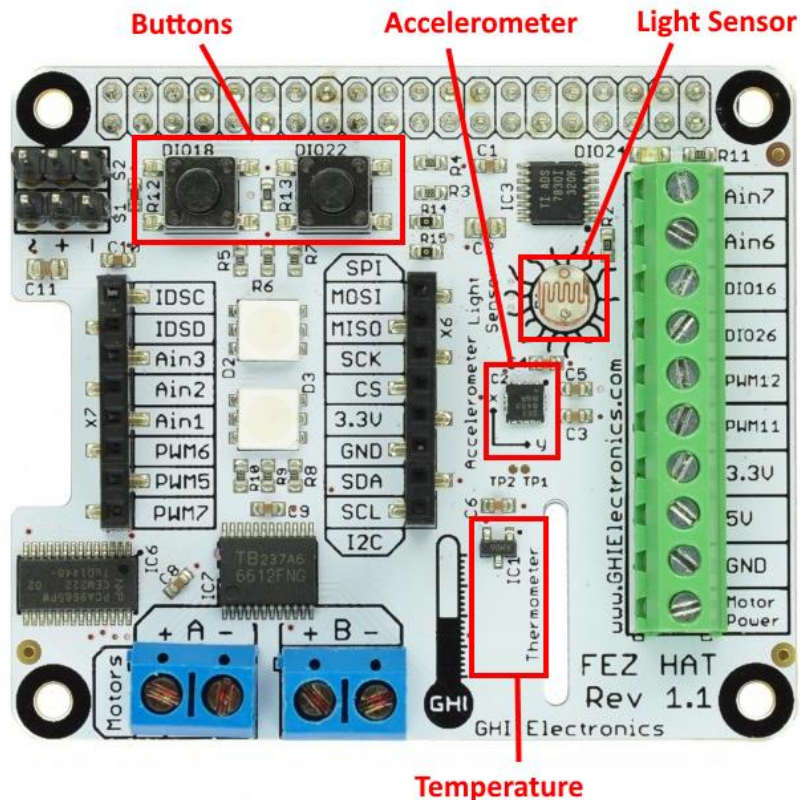
Windows IoT Core development with Visual Studio

Deploying your first "**Headed**" app

Interacting with Sensors

EXPERIMENT 5 EXPLORING APPS AND SENSORS

There are two styles of applications you can run on Windows IoT Core – Headed and Headless.

This section focuses on "Headed" apps. These are apps with a user interface that can display output on a screen (or via the Windows IoT Remote Client) and can use a mouse and keyboard. The app runs in the foreground and only one "Headed" app can be active at any one time.

⊠ **STEP 1:** Set the **MakerDenExtension** as the Startup Project. Right mouse click **MakerDenExtension** in the Solution Explorer and select "**Set as Startup Project**"
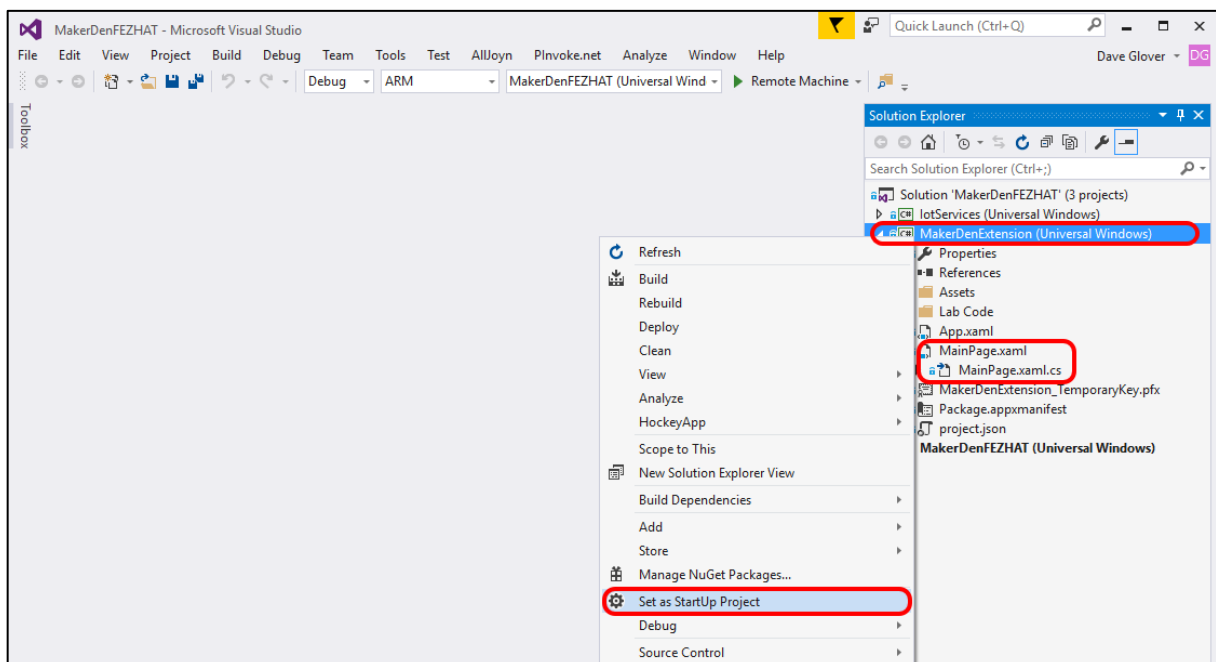
⊠**STEP 2:** Double click on **MainPage.xaml** to open the file. It will take a moment to load.

⊠**STEP 3:** Review the XAML markup that describes the User Interface, the MainPage.xaml page should look like the following.
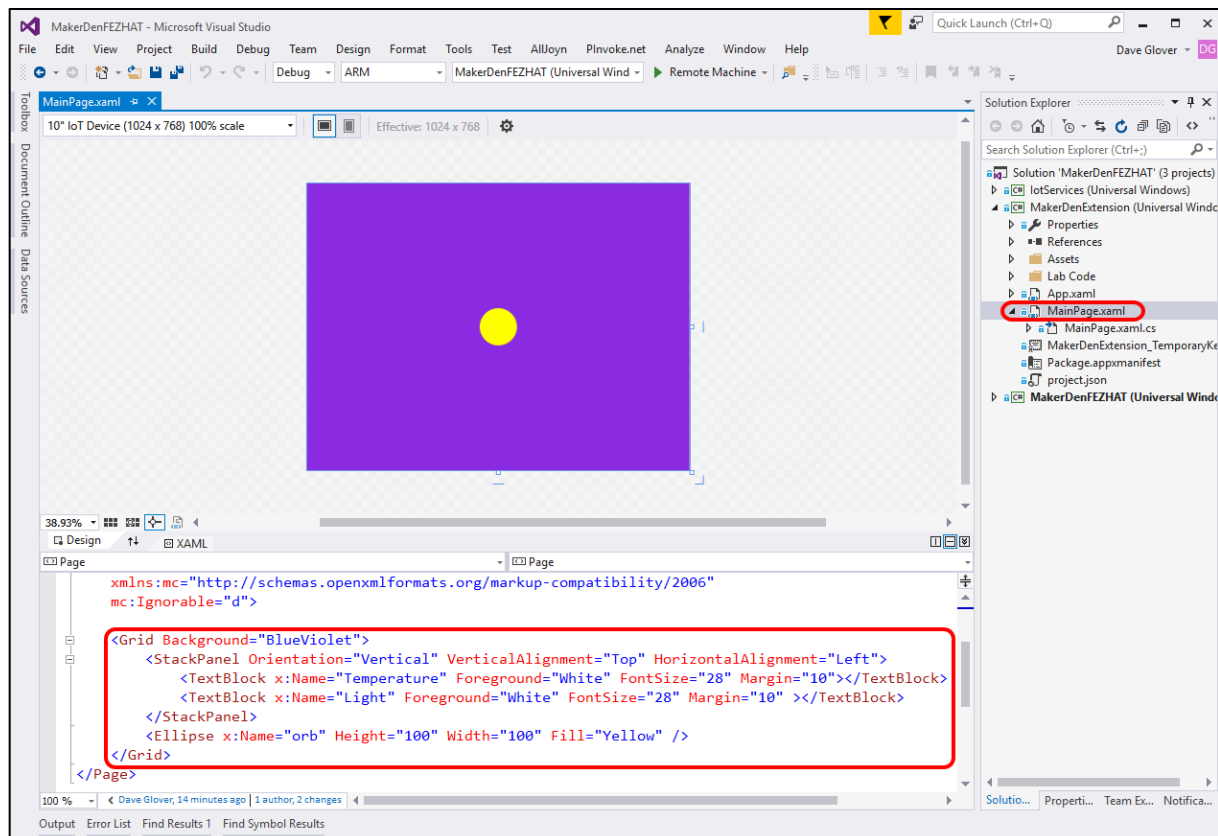
⊠ **STEP 4:** Double click on **MainPage.xaml.cs** to open the file. Locate the **Setup** method in the MainPage.xaml.cs file. Between the **#region Lab4b** tags type **lab4b** and press Tab twice **OR** type the following code.

```csharp
private async void Setup()
{
    #region Lab4b Code to go between the #region and #endregion tags

    myTransformGroup.Children.Add(myTranslate);
    orb.RenderTransform = myTransformGroup;

    this.hat = await FEZHAT.CreateAsync();

    timer = new DispatcherTimer();
    this.timer.Interval = TimeSpan.FromMilliseconds(100);
    this.timer.Tick += this.UpdateOrb;
    this.timer.Start();

    #endregion
}
```

⊠ **STEP 5:** Locate the **UpdateOrb** method in the MainPage.xaml.cs file. Ensure the cursor is between the **#region Lab4c and Lab4d Code** tags, then using a code snippet type **lab4c** and press Tab twice **OR** type the following code.

```csharp
Temperature.Text = string.Format("The temperature is {0:N2}", hat.GetTemperature());
Light.Text = string.Format("The light level is {0:N4}", hat.GetLightLevel());

if (hat.IsDIO18Pressed())
{
    computedColour = Colors.DeepPink;
}
if (hat.IsDIO22Pressed())
{
    computedColour = Colors.Lime;
}

orb.Fill = new SolidColorBrush(computedColour);

orb.UpdateLayout(); // update the orb with the new colour and position
```

⊠ **STEP 6:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**

⊠ **STEP 7: Maximise** the Windows IoT Remote Client from the Windows Taskbar.

You should have minimised the Windows IoT remote Client after you verified that it was working in the first section. But in case you closed it, then press the Windows key and type "Windows IoT Remote Client" and run. Then select your device from the dropdown list.

When your application has started on the Raspberry Pi it should look like the following image.

⊠**STEP 8:** For a bit of fun, modify the **UpdateOrb** method and type the highlighted line as below. This will change the brightness of the orb on the screen based on the ambient light levels.

```
if (hat.IsDIO18Pressed())
{
    computedColour = Colors.DeepPink;
}
if (hat.IsDIO22Pressed())
{
    computedColour = Colors.Lime;
}
computedColour.A = (byte)(255 * hat.GetLightLevel());  // change the orb brightness

orb.Fill = new SolidColorBrush(computedColour);

orb.UpdateLayout(); // update the orb with the new colour and position
```
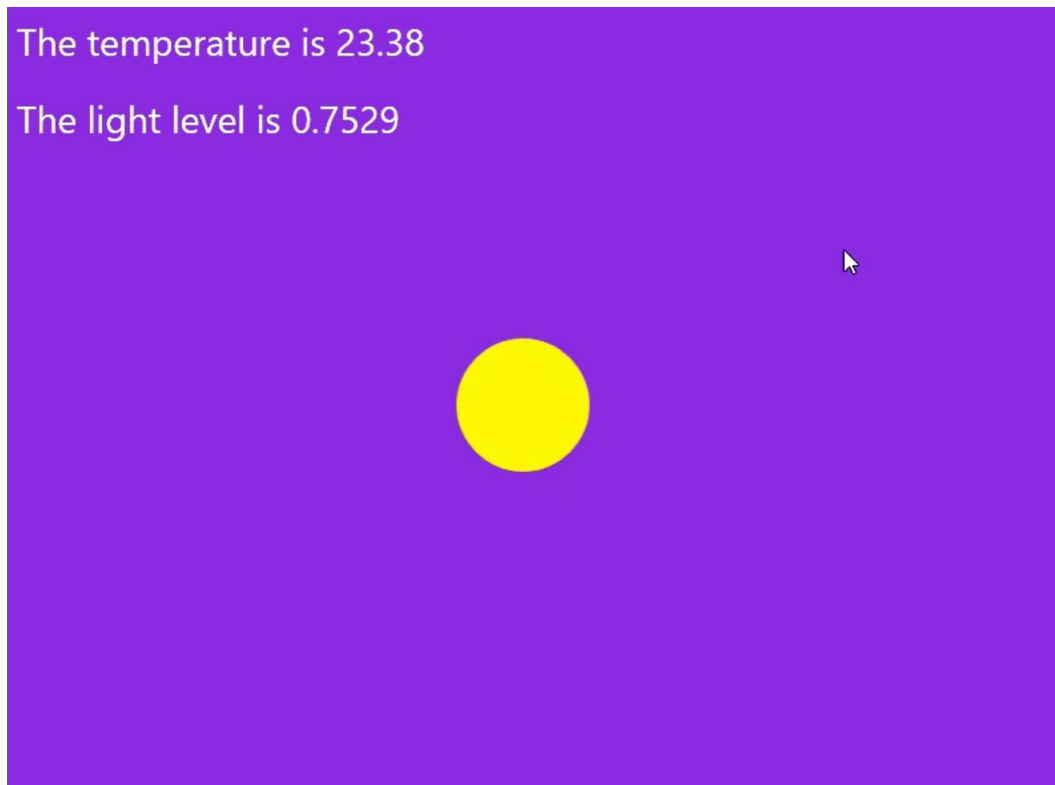
⊠**STEP 9:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**

⊠**STEP 10:** Maximise the Windows IoT Remote Client and observe that when you place your hand over the Raspberry Pi the brightness of the org will changed based on the ambient light levels.

EXPERIMENT 6: WHICH WAY IS UP WITH THE FEZ HAT ACCELEROMETER

⊠ **STEP 1:** Modify the **UpdateOrb** method so the colour and position of the orb change based on data from the accelerometer built into the Fez HAT.

Remove all the code between the **#Region Lab4c and Lab4d** Code tags. Then using a code snippet type **lab4d** and press Tab twice.

```csharp
private void UpdateOrb(object sender, object e)
{
    #region Lab4c and Lab4d Code to go between the #region and #endregion tags

    Temperature.Text = string.Format("The temperature is {0:N2}", hat.GetTemperature());
    Light.Text = string.Format("The light level is {0:N4}", hat.GetLightLevel());

    hat.GetAcceleration(out x, out y, out z);

    computedColour = ComputeColour(x, y, z);

    orb.Fill = new SolidColorBrush(computedColour);

    ComputeOrbPosition(x, y);

    orb.UpdateLayout(); // update the orb with the new colour and position

    #endregion
}
```

⊠ **STEP 2:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**

⊠ **STEP 3:** Maximise the Windows IoT Remote Client. Gently pick up the Raspberry Pi and tilt it backwards and forwards, left and right and observer the colour and position of the orb changes.

EXPERIMENT 7: CONTROLLING A DMX RGB SHOW LIGHT

There are a number of DMX Show lights at the front of the workshop. In this experiment we are going to send a message to the DMX controller to change the colour of your allocated light to match the colour of the orb and change the LEDs on the Fez HAT.

You will need the following information from the workshop coordinator

1. The MQTT broker address: _____
2. Your allocated Fixture Id: _____

⊠ **STEP 1:** In the **MainPage.xaml.cs** file, update the MQTT Broker Address and the Fixture Id with the valued you have been given.

```csharp
public sealed partial class MainPage : Page
{
    Expand to view global variables

    const string MqttBrokerAddress = "MQTT Broker Address";
    const uint FixtureId = 1;

    1 reference | Dave Glover, 17 hours ago | 1 author, 1 change
    public MainPage()
    {
        this.InitializeComponent();

        Setup();
    }
```

⊠**STEP 2:** Modify the **UpdateOrb** method and add the two highlighted lines to the end of the code in this method.

```csharp
private void UpdateOrb(object sender, object e)
{
    #region Lab4c and lab4d Code to go between the #region and #endregion tags

    Temperature.Text = string.Format("The temperature is {0:N2}", hat.GetTemperature());
    Light.Text = string.Format("The light level is {0:N4}", hat.GetLightLevel());

    hat.GetAcceleration(out x, out y, out z);

    computedColour = ComputeColour(x, y, z);

    orb.Fill = new SolidColorBrush(computedColour);

    ComputeOrbPosition(x, y);

    orb.UpdateLayout(); // update the orb with the new colour and position

    hat.D2.Color = hat.D3.Color = FezColour(computedColour); // set both Fez Hat RGB LEDs

    dmxLight.Update(computedColour); // set colour on DMX RGB Light

    #endregion
}
```

⊠**STEP 3:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5.**

⊠**STEP 4:** Tilt the Raspberry Pi backwards and forwards, left to right. Observer the colour and position of the Orb changes, the LEDs on the Fez HAT change and the colour of your DMX RGB Light will also change.

Woohoo you have finished

Congratulations