# Internet of Things Maker Den Lab Guide



Windows 10 IoT Core plus FEZ HAT

Internet of Things

Maker Den Lab Guide

Document Version 3.0

|  |  |
| --- | --- |
| Social | **Twitter** #makerden #iot #raspberrypi #windows10 #azure |
| Document Authors | Dave Glover | [dglover@microsoft.com](mailto:dglover@microsoft.com) | @dglover Andrew Coates |[acoat@microsoft.com](mailto:acoat@microsoft.com) |@coatsy Fai Lai |[hoongfai@microsoft.com](mailto:hoongfai@microsoft.com) |@faister |
| Document location | <https://github.com/MakerDen/IoT-Maker-Den-Documentation-and-Resources> |
| Source Code Location | <https://github.com/MakerDen/IoT-Maker-Den-Windows-for-IoT> |
| 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

Welcome to the Internet of Things Maker Den Lab where you will get firsthand experience with hardware prototyping and deploying code to a Raspberry Pi running Windows 10 IoT Core.

## Goal

The goal of the Maker Den is to familiarise you with some of the components and technologies associated with the Internet of Things (IoT). Along the way, you will experience wiring circuits, deploying code, and streaming sensor data to Microsoft Azure.

## 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>.

## Time Required

There are 2 parts to this lab. The first part is device centric and will take less than 15 minutes. Part 2 is more cloud centric and will take approximately an hour. You are more than welcome to stay longer and delve a little deeper.

## Spread the Word

Be sure to spread the word about the Internet of Things 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**](https://www.raspberrypi.org/)  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>. | http://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg |
| [**GHI electronics FEZ HAT**](https://www.ghielectronics.com/catalog/product/500)  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**](https://www.ghielectronics.com/docs/329/fez-hat-developers-guide) | https://www.ghielectronics.com/img/www/products/500-0_large.jpg |

## Experiments

* There are ten Maker Den experiments to get you started with Windows 10 IoT Core and Microsoft Azure IoT Services.
* All the source code can be referenced from the Source Code folder on the Desktop.
* This user guide and an architectural overview of the Maker Den can be found in the Documents folder on the Desktop.
* Be sure to check out the [Windows 10 IoT Core Doc, Tutorials and Samples](http://ms-iot.github.io/content/en-US/win10/StartCoding.htm). There is a link to this page in the Desktop Documents folder.
* For the self-sufficient adventurous types, the Windows 10 IoT Core Node.Js and Python developer tools have been installed. Reference the [Windows 10 IoT Core Doc, Tutorials and Samples](http://ms-iot.github.io/content/en-US/win10/StartCoding.htm) and the [GHI Electronics FEZ HAT](https://www.ghielectronics.com/docs/329/fez-hat-developers-guide) for more information.

### Resetting the Labs

* 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.

Part 1



Meet Windows IoT Core and Visual Studio

EXPERIMENT 1: Connecting and configuring your device

The Raspberry Pi will be connected to the development PC through 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](http://ms-iot.github.io/content/en-US/win10/ConnectToDevice.htm) is enabled on the PC.

1. Identifying the device using IoT Core Dashboard

In this task, you'll connect to your device and explore the web management interface.

Launch the Windows 10 IoT Core Dashboard, go to My devices and click the Open in Device Portal icon of your device name.

You can right mouse click a device for more options including copying the device IP Address, Name, plus start a PowerShell session.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module1-IntroWindows10IoTCore/Images/ex1task1-watcher.png?raw=true)

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](https://tinkertry.com/how-to-change-windows-10-network-type-from-public-to-private).

Alternatively, open a web browser and browse to the default device url [http://minwinpc:8080](http://minwinpc:8080/).

In the credentials dialog, use the default username and password. Username: *Administrator* Password: *p@ssw0rd*

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module1-IntroWindows10IoTCore/Images/ex1task1-device-portal-credentials.png?raw=true)

**Windows Device Portal** should launch and display the web management home screen!

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module1-IntroWindows10IoTCore/Images/ex1task1-device-portal.png?raw=true)

Windows Device Portal

1. Verify Device Configuration

From the **Home** Tab verify the Time Zone, date and time are correct. If the device has the incorrect data or time, then refer to the [troubleshooting](#_TroubleShooting) section in the appendix.

From the **Remote** tab verify that **Windows IoT Remote Server** enabled. If it is not, then enable it.

* **STEP 3:** Test Windows IoT Remote Client connection

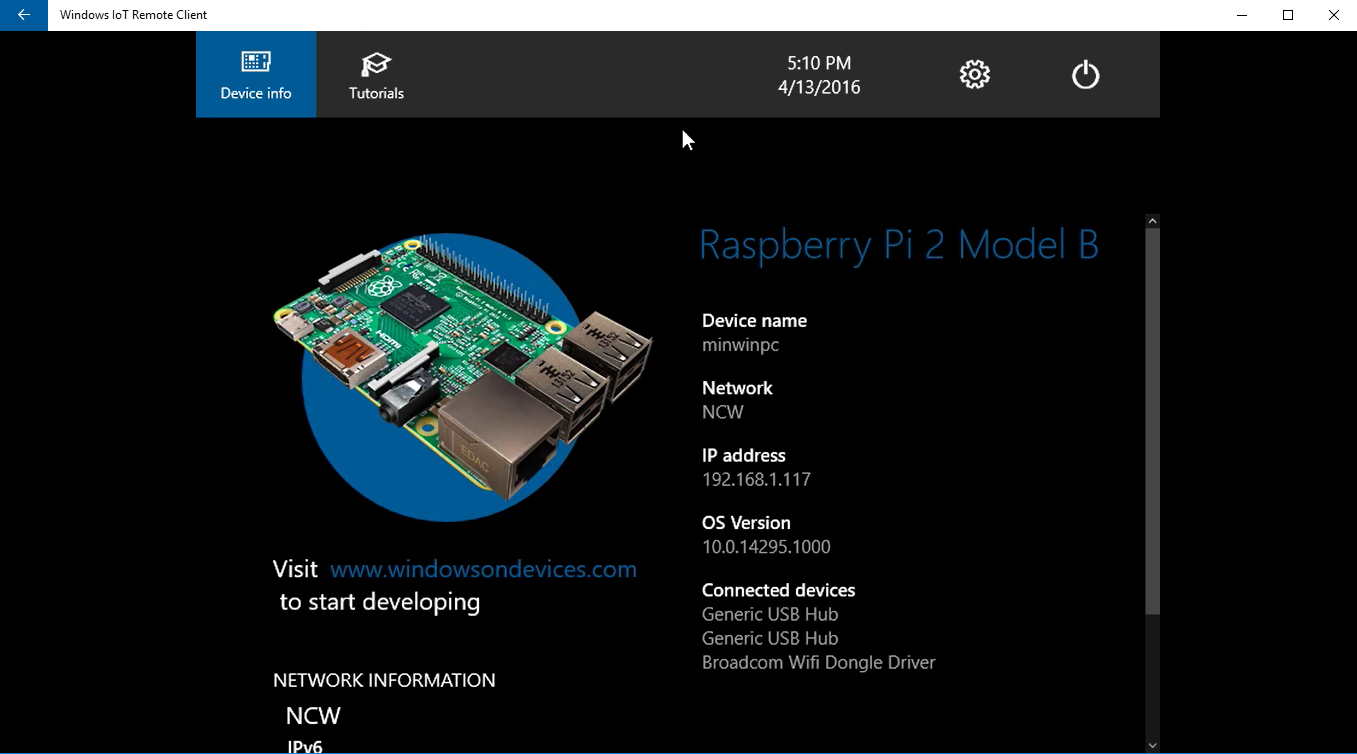
The [Windows IoT remote Client](https://www.microsoft.com/store/apps/9nblggh5mnxz) is available from the Windows Store.

To run, press the Windows key and type “Windows IoT Core Remote Client” and run the app.

It is likely that you will 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.

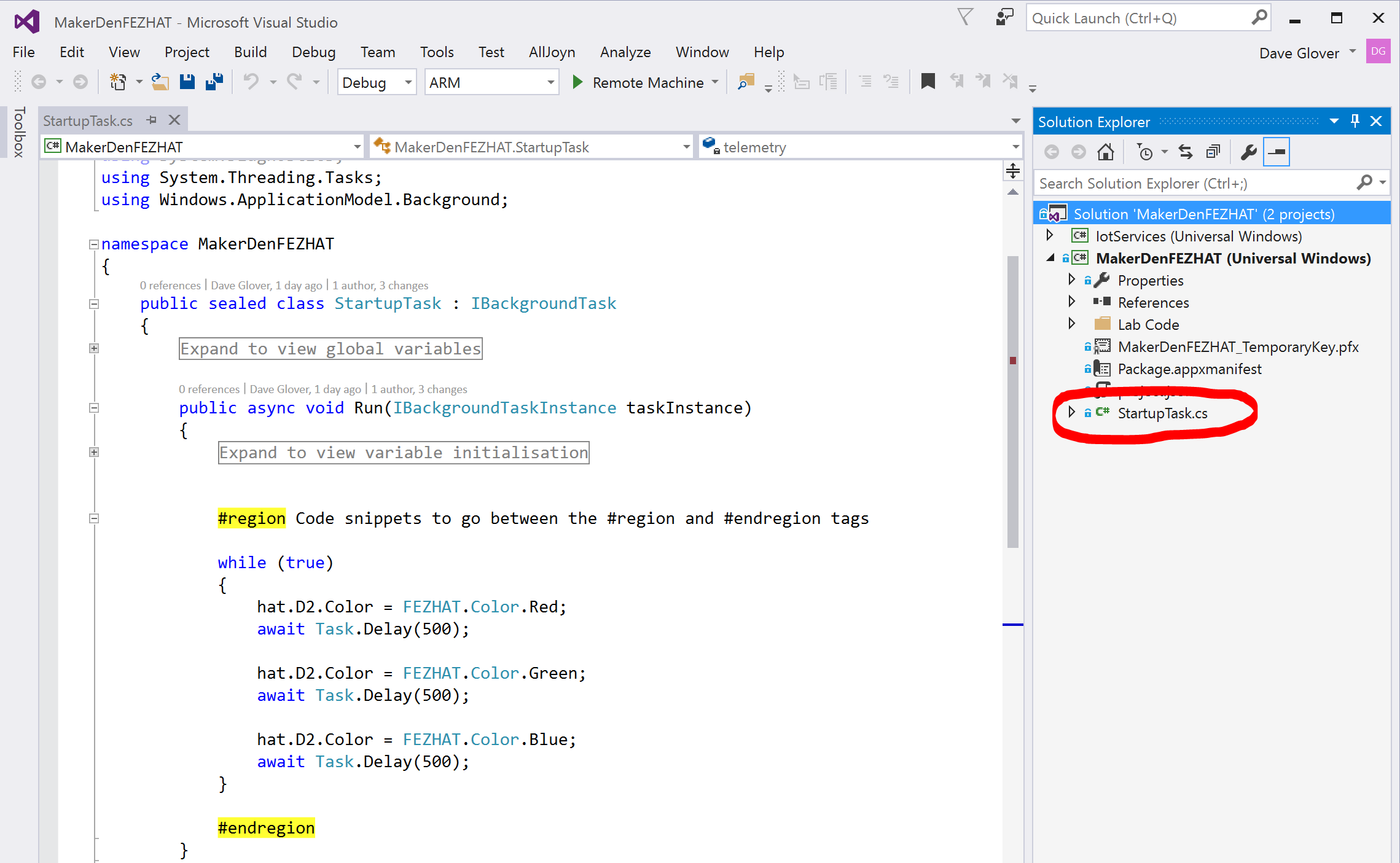
Minimize the remote client application when you have verified that it is working.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module1-IntroWindows10IoTCore/Images/windows-iot-remote-client.png?raw=true)

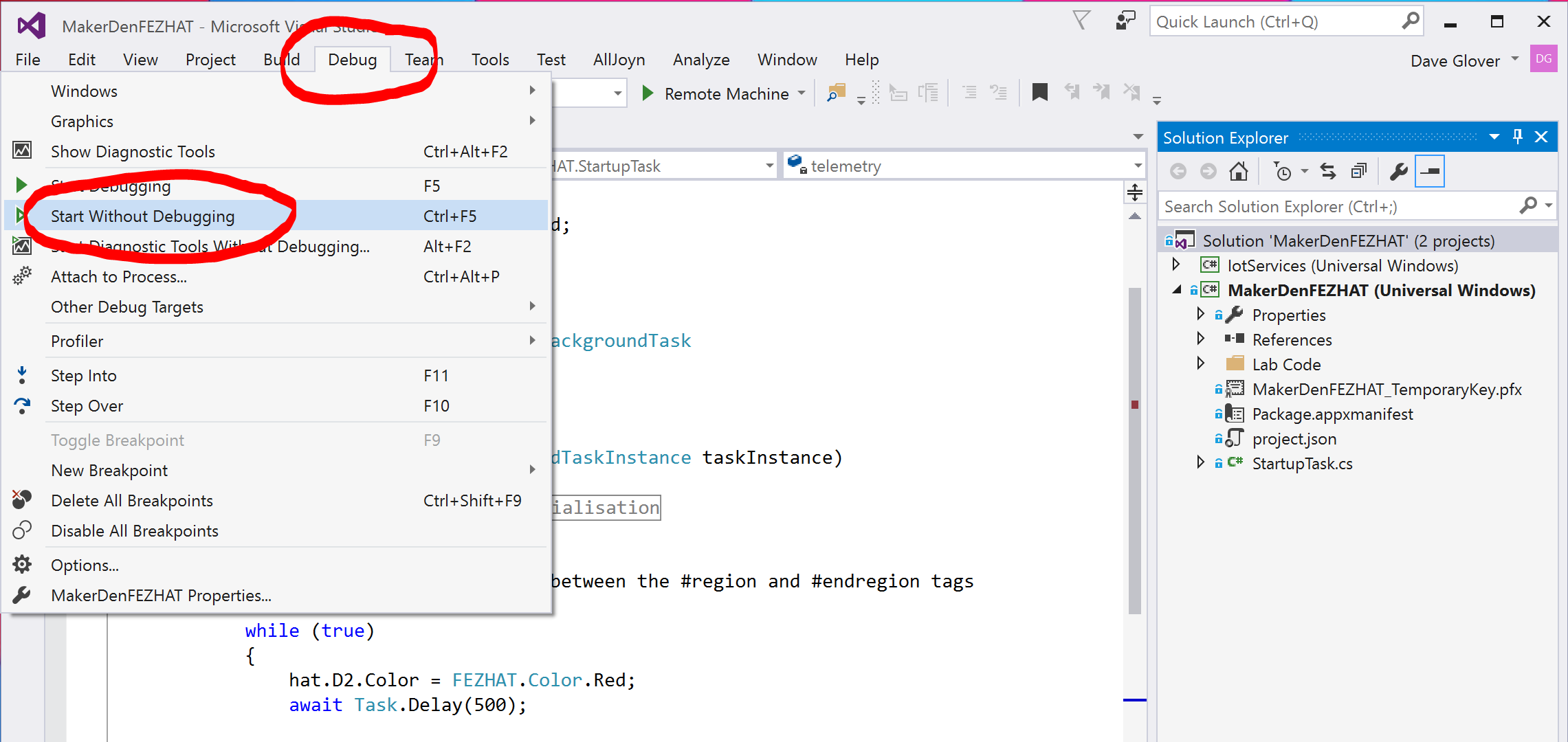
Experiment 2: Hello WORLD

Deploy your first experiment to ensure everything is setup correctly and to check Visual Studio is communicating with your Raspberry Pi.

* **STEP 1**: Expand the **MakerDen** project then double click the **StartupTask.cs** file to open it.



* **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.

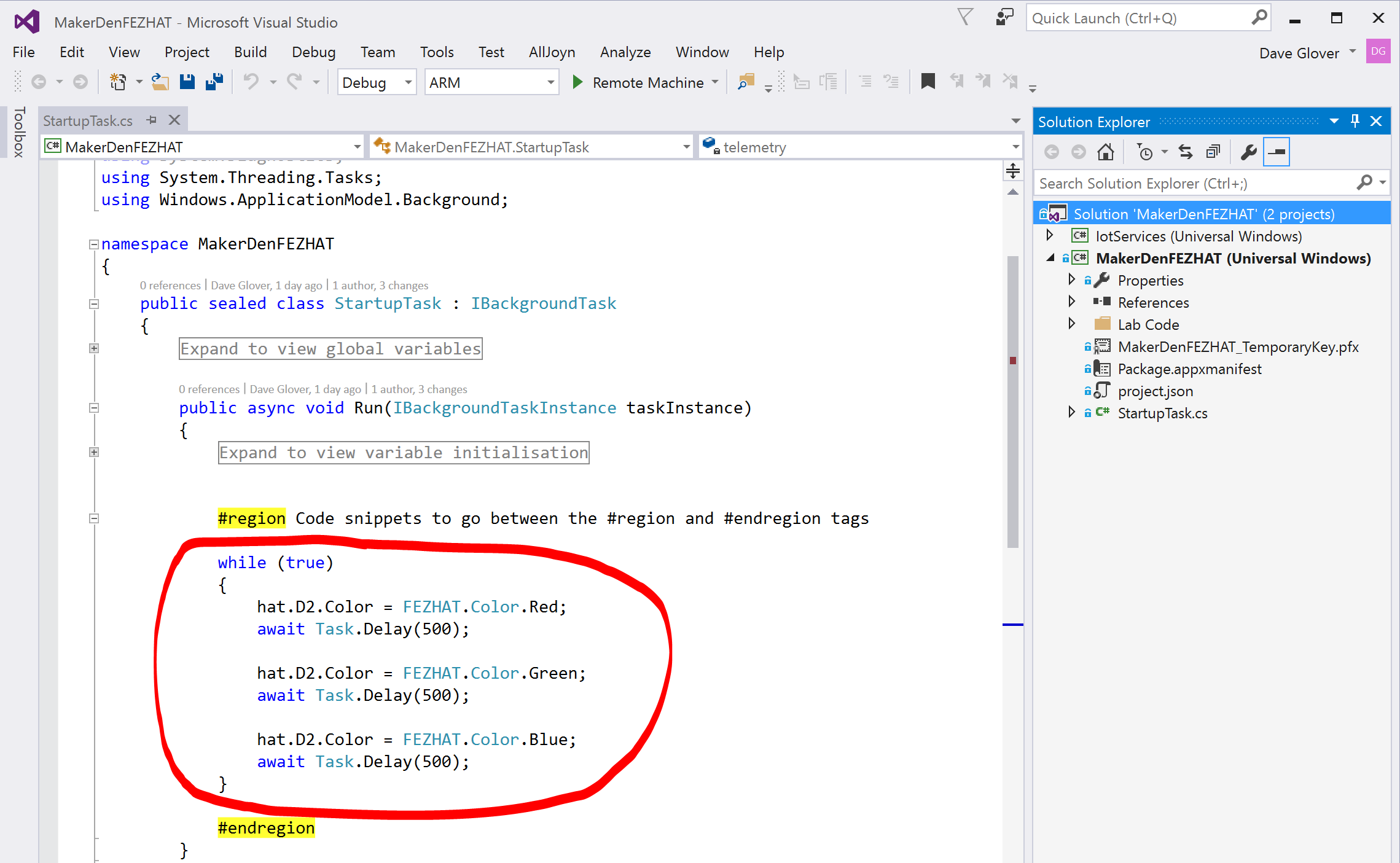


* **STEP 4:** Check the LEDs on the FEZ HAT. You should see an LED alternating between Red, Green and Blue.
* **STEP 5:** Pat yourself on the back, you did it☺

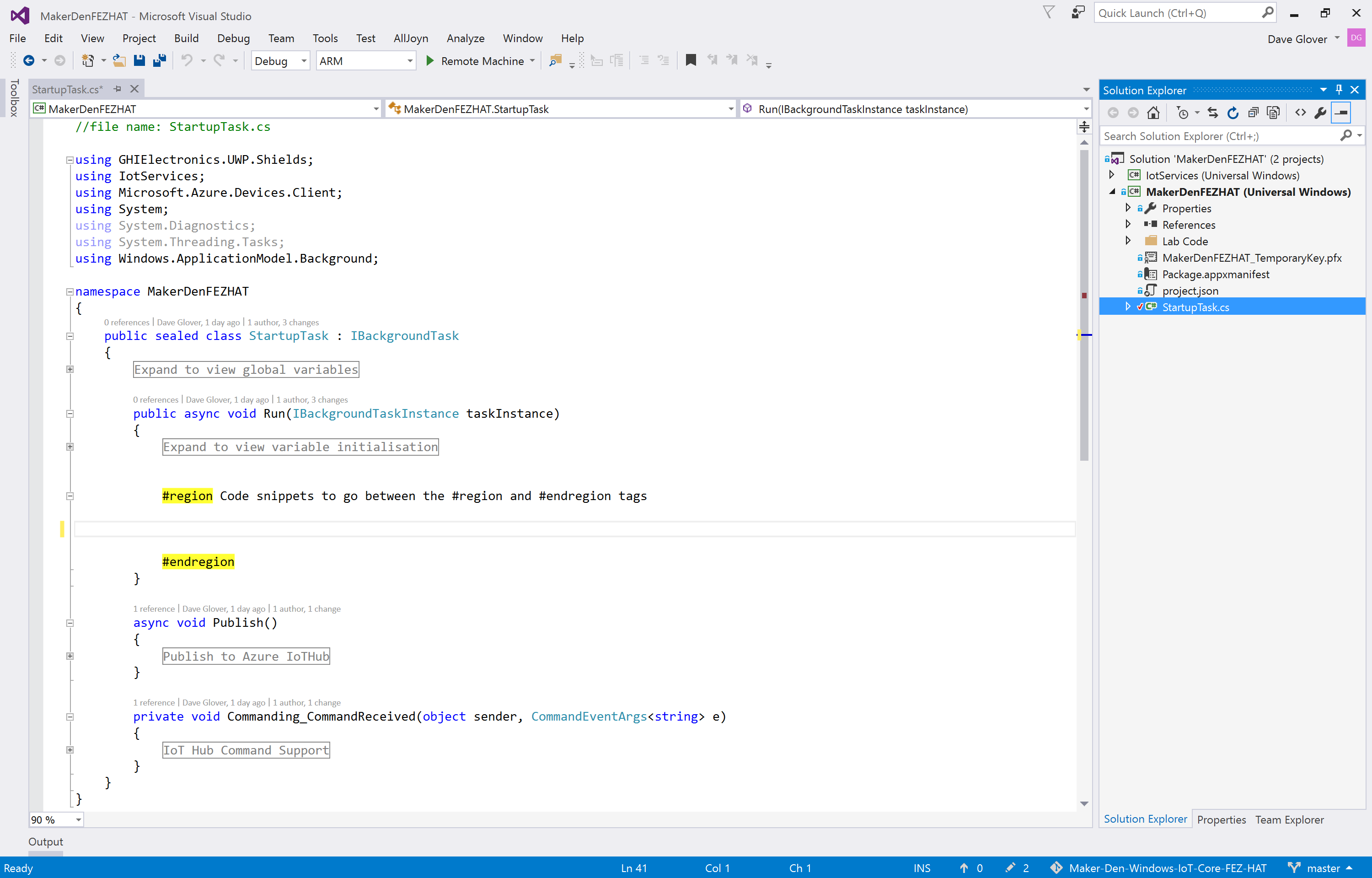
Experiment 3: Sensing the World

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

* **STEP 1:** Review the code in the **StartupTask.cs** file. Look for the **#region** and **#endregion** tags.
* **STEP 2:** Delete the code circled in red **inside** the #region tags.



Your “StartupTask.cs” file should look like the screenshot below after you have deleted the code. If it doesn’t look the same then **Ctrl+Z** to undo the changes you made and try again.



* **STEP 3:** Type the following code between the #region tags **OR** using a code snippet type **lab3** and press Tab twice.

while (true)

{

var level = hat.GetLightLevel() \* 100;

if (hat.GetLightLevel() \* 100 > 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.Threading.Tasks;

using Windows.ApplicationModel.Background;

namespace MakerDenFEZHAT

{

public sealed class StartupTask : IBackgroundTask

{

DeviceClient deviceClient = DeviceClient.CreateFromConnectionString("Connection String”);

Expand to view global variables

public async void Run(IBackgroundTaskInstance taskInstance)

{

Expand to view variable initialisation

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

while (true)

{

var level = hat.GetLightLevel() \* 100;

if (hat.GetLightLevel() \* 100 > LIGHT\_THRESHOLD)

{

hat.D2.Color = FEZHAT.Color.Blue;

}

else

{

hat.D2.Color = FEZHAT.Color.Red;

}

await Task.Delay(500);

}

#endregion

}

async void Publish()

{

#region Publish to Azure IoTHub

#endregion

}

private void Commanding\_CommandReceived(object sender, CommandEventArgs<string> e)

{

#region 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 alternates 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.  
    
  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 that LED changes colour depending on ambient light levels.
* **Step 5**: Press Shift-F5 to stop debugging.

Part 2

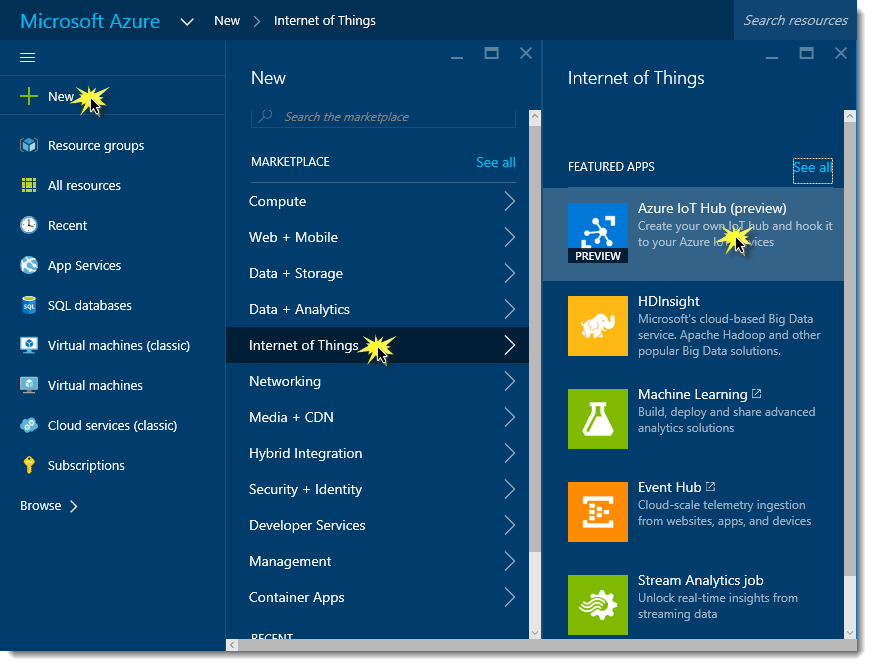


Meet Microsoft Azure, Stream Analytics and Power Bi

EXPERIEMENT 5: Provisioning an Azure IoT Hub and an IoT Device

Creating an IoT Hub

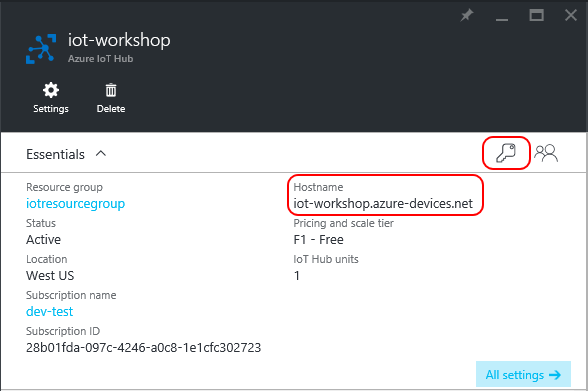
* **STEP 1:** Sign in to the Azure portal at [http://portal.azure.com](http://portal.azure.com/)
* **STEP 2:** Create a new IoT Hub. To do this, click **New** in the Jumpbar, then click **Internet of Things**, then click **Azure IoT Hub**.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/creating-a-new-iot-hub.png?raw=true)

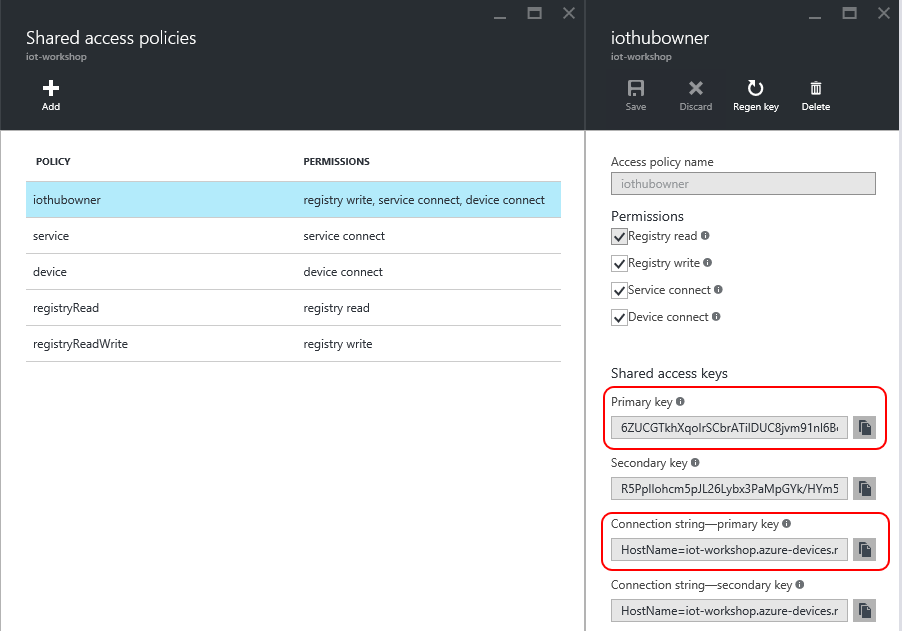
* **STEP 3:** Configure the **IoT hub** with the desired information:
  + Enter a **Name** for the hub e.g. iot-workshop (must be global unique name)
  + Select a **Pricing and scale tier** (F1 Free tier is enough)
  + Create a new resource group, or select and existing one. For more information, see [Using resource groups to manage your Azure resources](https://azure.microsoft.com/en-us/documentation/articles/resource-group-portal/).
  + Select the **Region** closest to where your solution and or devices will be located.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/new-iot-hub-settings.png?raw=true)

* **STEP 4:** It can take a few minutes for the IoT hub to be created. Once it is ready, open the blade of the new IoT hub, take note of the URI and select the key icon at the top to access to the shared access policy settings:

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/iot-hub-shared-access-policies.png?raw=true)

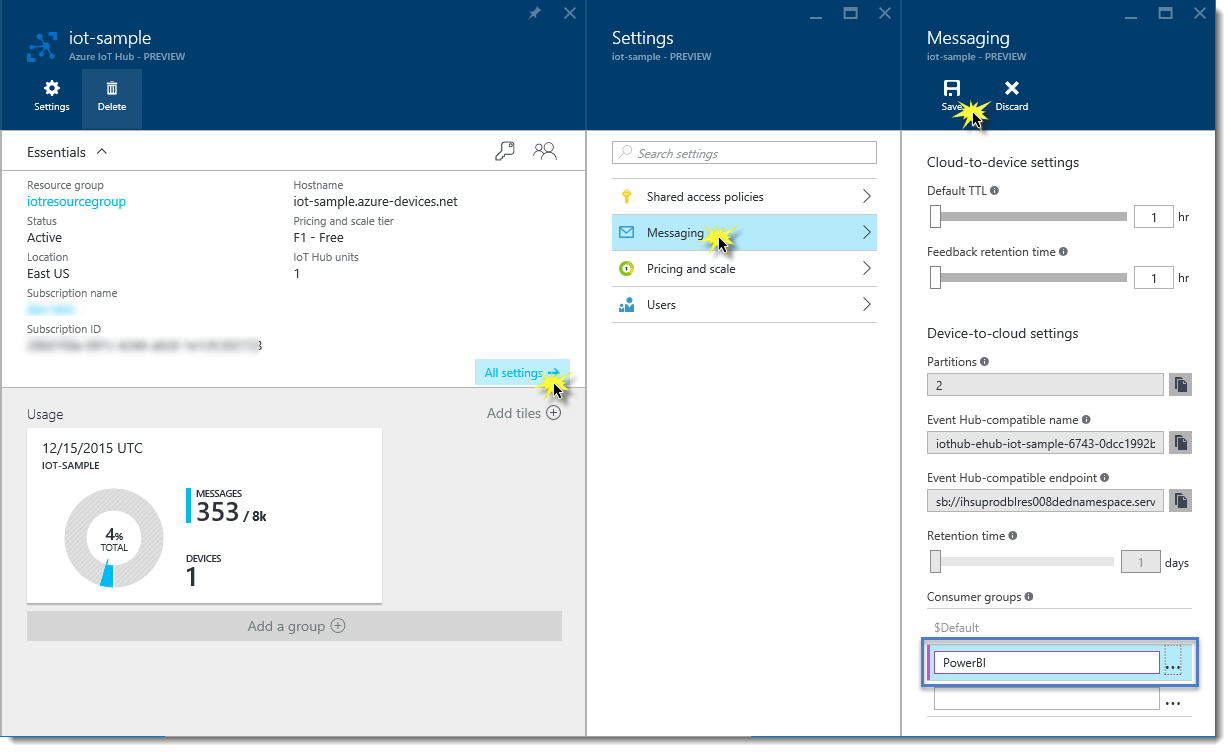
* **STEP 6:** Select the Shared access policy called **iothubowner**, and take note of the **Primary key** and **connection string** in the right blade. You should copy these into a text file for future use.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/get-iot-hub-owner-connection-string.png?raw=true)

Create a Service Bus Consumer Group

For Experiment 7 we next an additional consumer group to allow several applications to read data from the IoT Hub independently. Follow these steps:

* **STEP 1:** From the settings blade, click on **Messaging**
* **STEP 2:** At the bottom of the Messaging blade, type the name of the new Consumer Group "PowerBI"
* **STEP 3:** From the top menu, click on the Save icon

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/create-consumer-group.png?raw=true)

Registering your device

You must register your device in order to be able to send and receive information from the Azure IoT Hub. This is done by registering a [Device Identity](https://azure.microsoft.com/en-us/documentation/articles/iot-hub-devguide/#device-identity-registry) in the IoT Hub.

* **STEP 1:** Press the Windows key and type “Device Explorer[[1]](#footnote-2)” and run the app.
* **STEP 2:** Copy the **IoT Hub Connection String** you previously pasted in to the IoT Connection String field click **Update**.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/configure-device-explorer.png?raw=true)

* **STEP 2:** Go to the **Management** tab and click on the **Create** button. The Create Device popup will be displayed. Fill the **Device ID** field with a new Id for your device (myFirstDevice for example) and click on Create:

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/creating-a-device-identity.png?raw=true)

* **STEP 3:** Once the device identity is created, it will be displayed in the grid. Right click on the identity you just created, select **Copy connection string for selected device** and take note of the value copied to your clipboard, since it will be required to connect your device with the IoT Hub.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/copying-device-connection-information.png?raw=true)

Experiement 6: Streaming telemetry data to Azure IoT hub

* **STEP 1:**

public async void Run(IBackgroundTaskInstance taskInstance)

{

Expand to view variable initialisation

deviceClient = DeviceClient.CreateFromConnectionString("connection string");

#region Code snippets 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

}

* **STEP 2:** Type the following code in the **Publish** method between the **#region Publish to Azure IoTHub** tags **OR** using a code snippet type **lab4** and press Tab twice.

try // Exception handling if problem streaming telemetry to Azure IoT Hub

{

hat.D3.Color = publishColor; // turn on publish indicator LED

var temperature = hat.GetTemperature(); // read temperature from the FEZ HAT

var light = hat.GetLightLevel(); // read light level from the FEZ HAT

var json = telemetry.ToJson(temperature, light, 0, 0); //serialise to JSON

var content = new Message(json);

await deviceClient.SendEventAsync(content); //Send telemetry data to IoT Hub

}

catch { telemetry.Exceptions++; }

finally { hat.D3.TurnOff(); }

* **STEP 3:** Your completed Publish method should look like this

async void Publish()

{

#region Publish to Azure IoTHub

try // Exception handling if problem streaming telemetry to Azure IoT Hub

{

hat.D3.Color = publishColor; // turn on publish indicator LED

var temperature = hat.GetTemperature(); // read temperature from the FEZ HAT

var light = hat.GetLightLevel(); // read light level from the FEZ HAT

var json = telemetry.ToJson(temperature, light, 0, 0); //serialise to JSON

var content = new Message(json);

await deviceClient.SendEventAsync(content); //Send telemetry data to IoT Hub

}

catch { telemetry.Exceptions++; }

finally { hat.D3.TurnOff(); }

#endregion

}

* **STEP 4:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start without Debugging** or from the keyboard press **Ctrl+F5** and wait for the solution to deploy.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/monitoring-messages-sent.png?raw=true)

**Note**: If you navigate back to your IoT Hub blade in the Azure Portal, it may take a couple minutes before the message count is updated to reflect the device activity under **Usage**.

EXPERIMENT 7: Consuming the IoT Hub data

You have seen how to use the Device Explorer to peek the data being sent to the Azure IoT Hub. However, the Azure IoT suite offers many different ways to generate meaningful information from the data gathered by the devices. In the following section you will use Azure Stream Analytics in combination with Microsoft Power BI to consume the data and to generate meaningful reports.

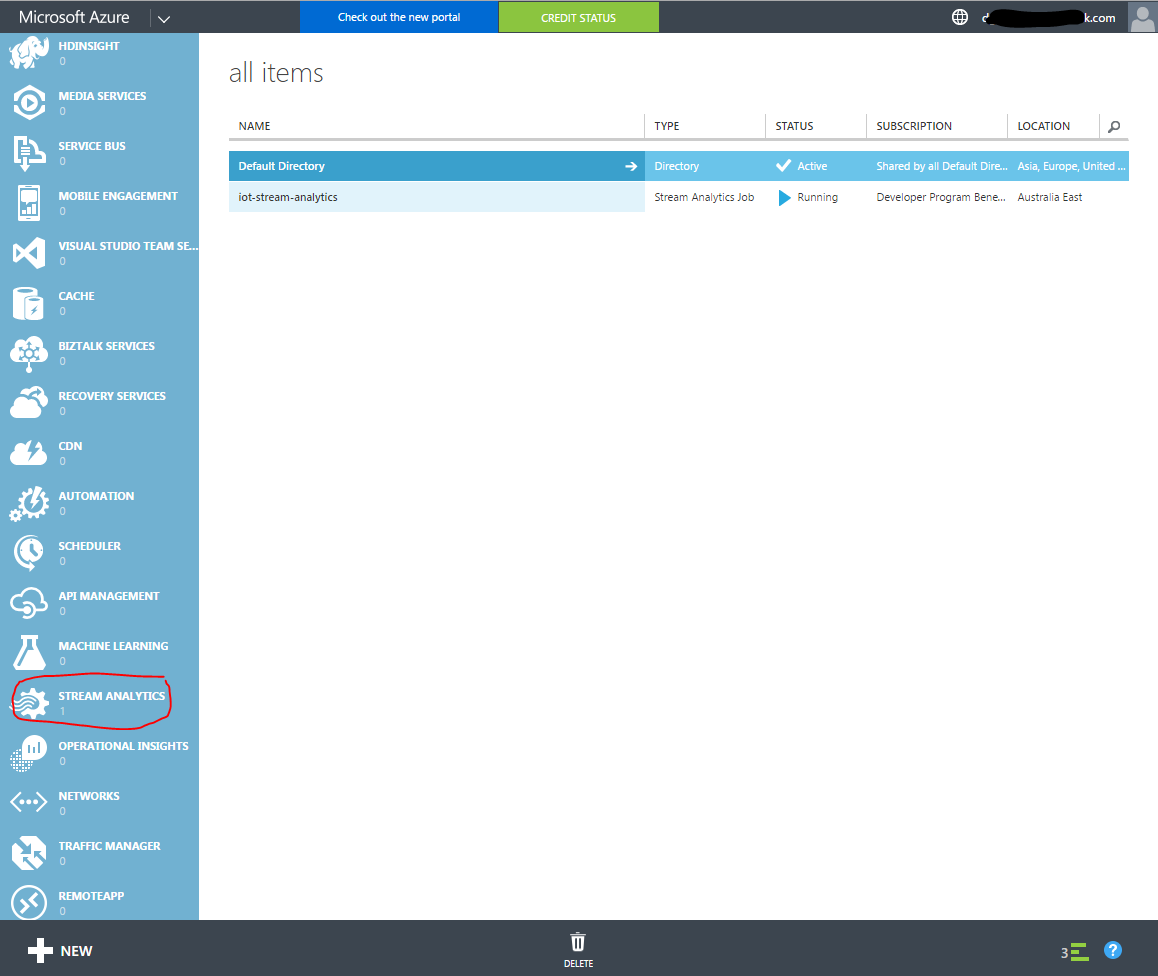
* **STEP 1:** Setting the data source

In order to feed the Power BI reports with the information gathered by the hats and to get that information in near real-time, **Power BI** supports **Azure Stream Analytics** outputs as data source. The following section will show how to configure the Stream Analytics job created in the Setup section to take the input from the IoT Hub and push that summarized information to Power BI.

* **STEP 2:** Stream Analytics Input Setup

Before the information can be delivered to **Power BI**, it must be processed by a **Stream Analytics Job**. To do so, an input for that job must be provided. As the Raspberry devices are sending information to an IoT Hub, it will be set as the input for the job.

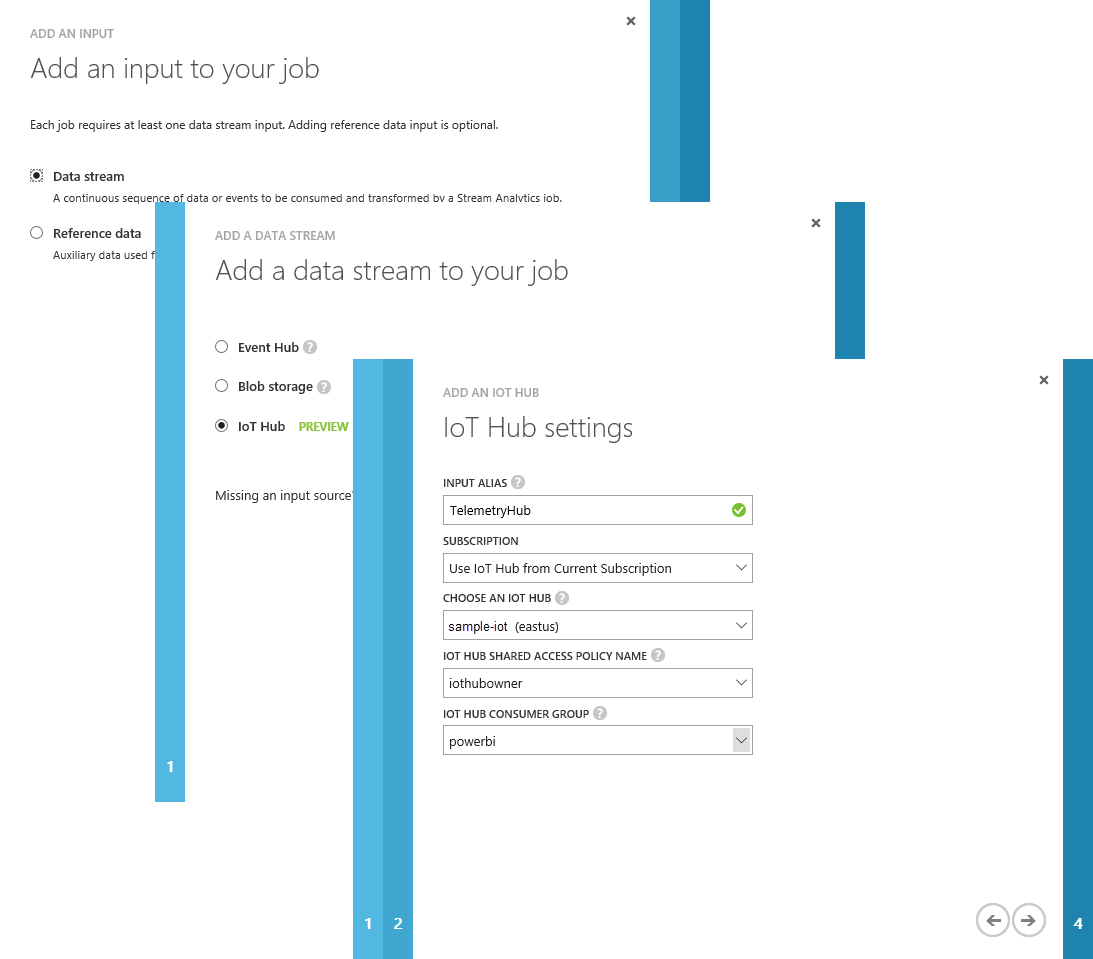
1. Go to the classic [Azure management portal](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Stream%20Analytics%20Input) ([https://manage.windowsazure.com](https://manage.windowsazure.com/)) and select the **Stream Analytics**service.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/azure-create-stream-analytics-job.png)

1. Click "Create a new Stream Analytics job" if this is your first job, or click the plus sign to add an additional job.
2. Complete the form

* JOB NAME: iot-stream-analytics or something similar
* REGION: Suggest selecting the region closest to your IoT Hub
* REGIONAL MONITORING STORAGE ACCOUNT: Likely "Create new storage account
* NEW STORAGE ACCOUNT NAME: Needs to be a globally unique name. Your app or alias name followed by the word storage eg gloveboxstorage

1. Click "CREATE STREAM ANALYTICS JOB
2. Once the Stream Analytics job is created click on it to configure.
3. As you can see, the Start button is disabled since the job is not configured yet. To set the job input click on the **INPUTS** tab and then in the **Add an input** button.
4. In the **Add an input to your job** popup, select the **Data Stream** option and click **Next**. In the following step, select the option **IoT Hub** and click **Next**. Lastly, in the **IoT Hub Settings** screen, provide the following information:
   * **Input Alias:** TelemetryHub
   * **Subscription:** Use IoT Hub from Current Subscription (you can use an Event Hub from another subscription too by selecting the other option)
   * **Choose an IoT Hub:** iot-sample (or the name used during the IoT Hub creation)
   * **IoT Hub Shared Access Policy Name:** iothubowner
   * **IoT Hub Consumer Group:** powerbi

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/stream-analytics-input-configuration.png?raw=true)

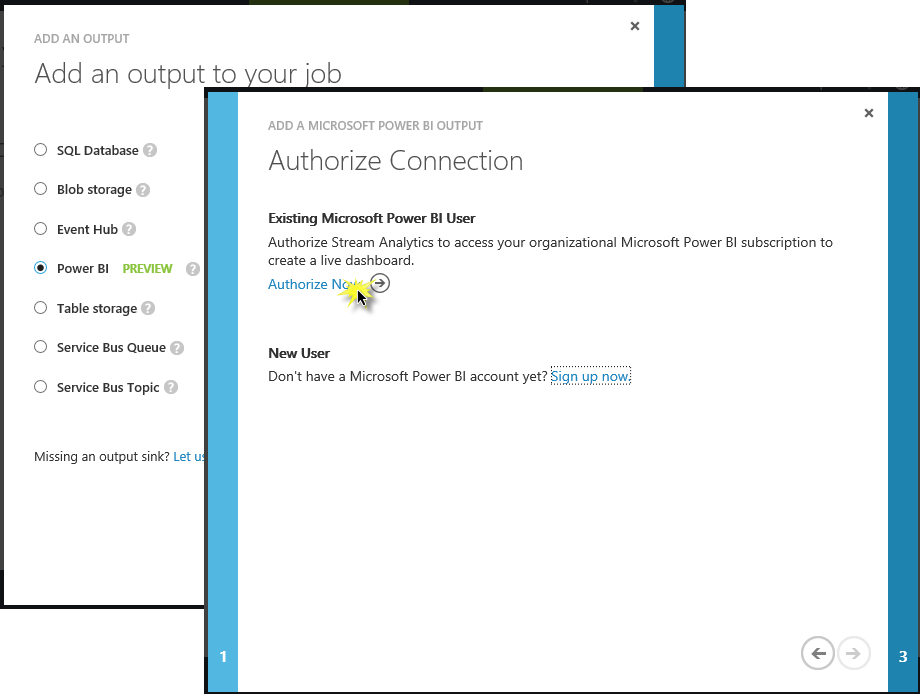
Stream Analytics Input Configuration

1. Click **Next**, and then **Complete** (leave the Serialization settings as they are).

### Task 5 - Stream Analytics Output Setup

The output of the Stream Analytics job will be Power BI.

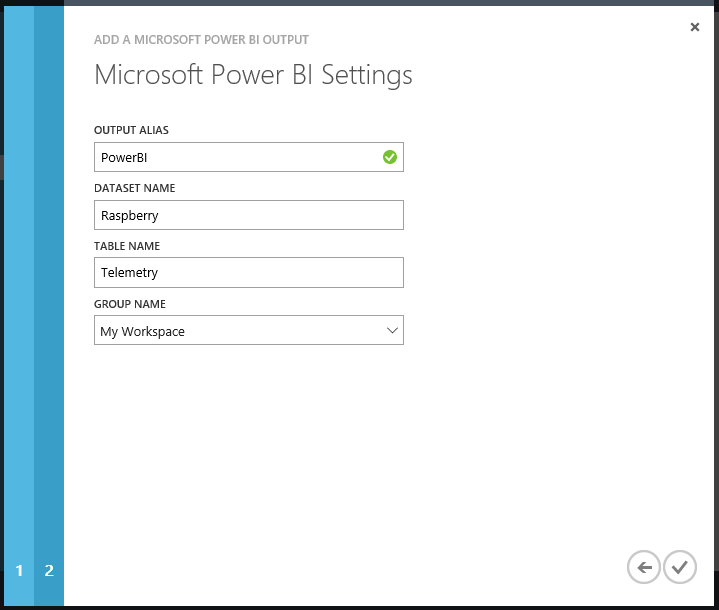
1. To set up the output, go to the Stream Analytics Job's **OUTPUTS** tab, and click the **ADD AN OUTPUT** link.
2. In the **Add an output to your job** popup, select the **POWER BI** option and the click the **Next button**.
3. In the following screen you will setup the credentials of your Power BI account in order to allow the job to connect and send data to it. Click the **Authorize Now** link.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/steam-analytics-output-configuration.png?raw=true)

Stream Analytics Output Configuration

You will be redirected to the Microsoft login page.

1. Enter your Power BI account email and password and click **Continue**. If the authorization is successful, you will be redirected back to the **Microsoft Power BI Settings** screen.
2. In this screen you will enter the following information:
   * **Output Alias**: PowerBI
   * **Dataset Name**: Raspberry
   * **Table Name**: Telemetry
   * **Group Name**: My Workspace

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/power-bi-settings.png?raw=true)

Power BI Settings

1. Click the checkmark button to create the output.

### Task 6 - Stream Analytics Query configuration

Now that the job's inputs and outputs are already configured, the Stream Analytics Job needs to know how to transform the input data into the output data source. To do so, you will create a new Query.

1. Go to the Stream Analytics Job **QUERY** tab and replace the query with the following statement:

SELECT

iothub.connectiondeviceid deviceid,

Geo AS GeoLocation,

Max(DateAdd(Hour, 10, EventEnqueuedUtcTime)) AS TimeCreated, -- AU EST UTC + 10

Avg(Celsius) AS Temperature,

AVG(Humidity) AS Humidity,

AVG(Light) AS Light,

AVG(HPa) AS AirPressure

INTO

[PowerBI]

FROM

[TelemetryHUB] TIMESTAMP BY EventEnqueuedUtcTime

GROUP BY

iothub.connectiondeviceid, Geo,

TumblingWindow(Second, 30)

The query takes the data from the input (using the alias defined when the input was created **TelemetryHUB**) and inserts into the output (**PowerBI**, the alias of the output) after grouping it using 10 seconds chunks.

1. Click on the **SAVE** button and **YES** in the confirmation dialog.

### Task 7 - Starting the Stream Analytics Job

Now that the job is configured, the **START** button is enabled. Click the button to start the job and then select the **JOB START TIME** option in the **START OUTPUT** popup. After clicking **OK** the job will be started.

Once the job starts it creates the Power BI datasource associated with the given subscription.

**NOTE: Starting a Stream Analytics job will start to consume your Azure Credits.**

Experiement 8: Setting up the Power BI dashboard

1. Now that the datasource is created, go back to your Power BI session, and go to **My Workspace** by clicking the **Power BI** link.

After some minutes of the job running you will see that the dataset that you configured as an output for the Job, is now displayed in the Power BI workspace Datasets section.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/power-bi-new-datasource.png?raw=true)

**Note:** The Power BI dataset will only be created if the job is running and if it is receiving data from the IoT Hub input, so check that the Universal App is running and sending data to Azure to ensure that the dataset be created. To check if the Stream Analytics job is receiving and processing data you can check the Azure Management Stream Analytics monitor.

Once the datasource becomes available you can start creating reports. To create a new Report click on the **Raspberry** datasource:

* **STEP 1: Using the Power Bi Designer Create Reports**

The Report designer will be opened showing the list of fields available for the selected datasource and the different visualizations supported by the tool.

|  |  |
| --- | --- |
| Selecting the Line Chart   1. Then you will set a filter to show only the Light sensor data. To do so drag the **measurename** field to the **Filters** section and then select the **Light** value: 2. To create the Average Light by time report, select the following fields:    * avgvalue    * timecreated   As you can see the **avgvalue** field is automatically set to the **Value** field and the **timecreated** is inserted as an axis. Now change the chart type to a **Line Chart**: |  |

* **STEP 2:** Save the Report

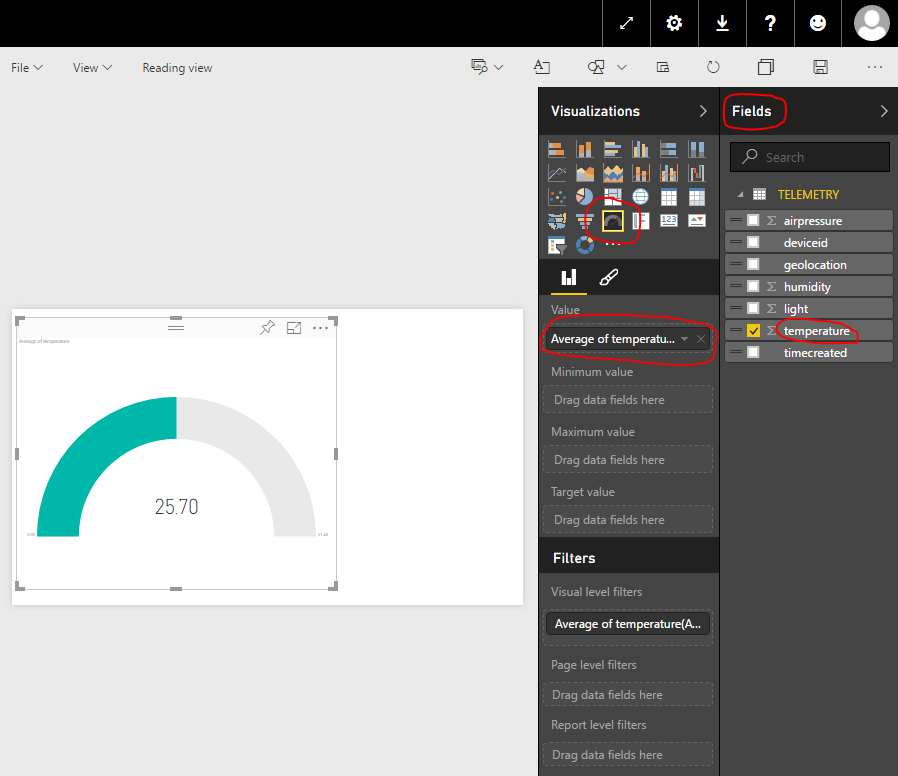
1. Now the report is almost ready. Click the **SAVE** button and set Light by Time as the name for the report.
2. Now you will create a new Dashboard, and pin this report to it. Click the plus sign (+) next to the **Dashboards** section to create a new dashboard. Set Raspberry Telemetry as the **Title** and press Enter. Now, go back to your report and click the pin icon to add the report to the recently created dashboard.

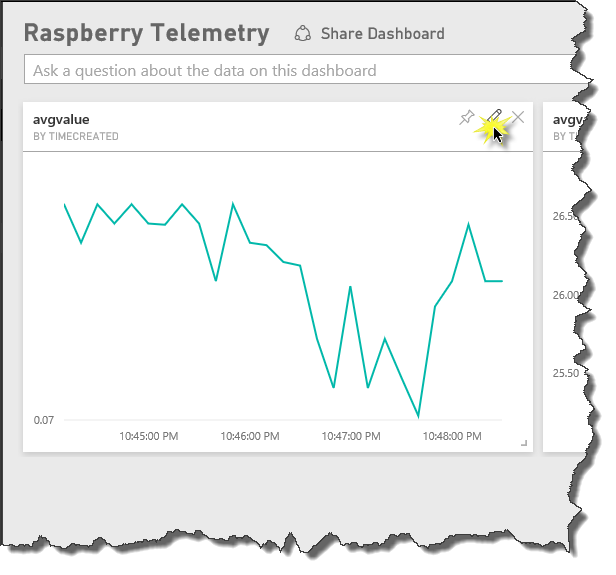
[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/pin-a-report-to-the-dashboard.png?raw=true)

Temperature Power Bi Chart

To create a second chart with the information of the average Temperature follow these steps:

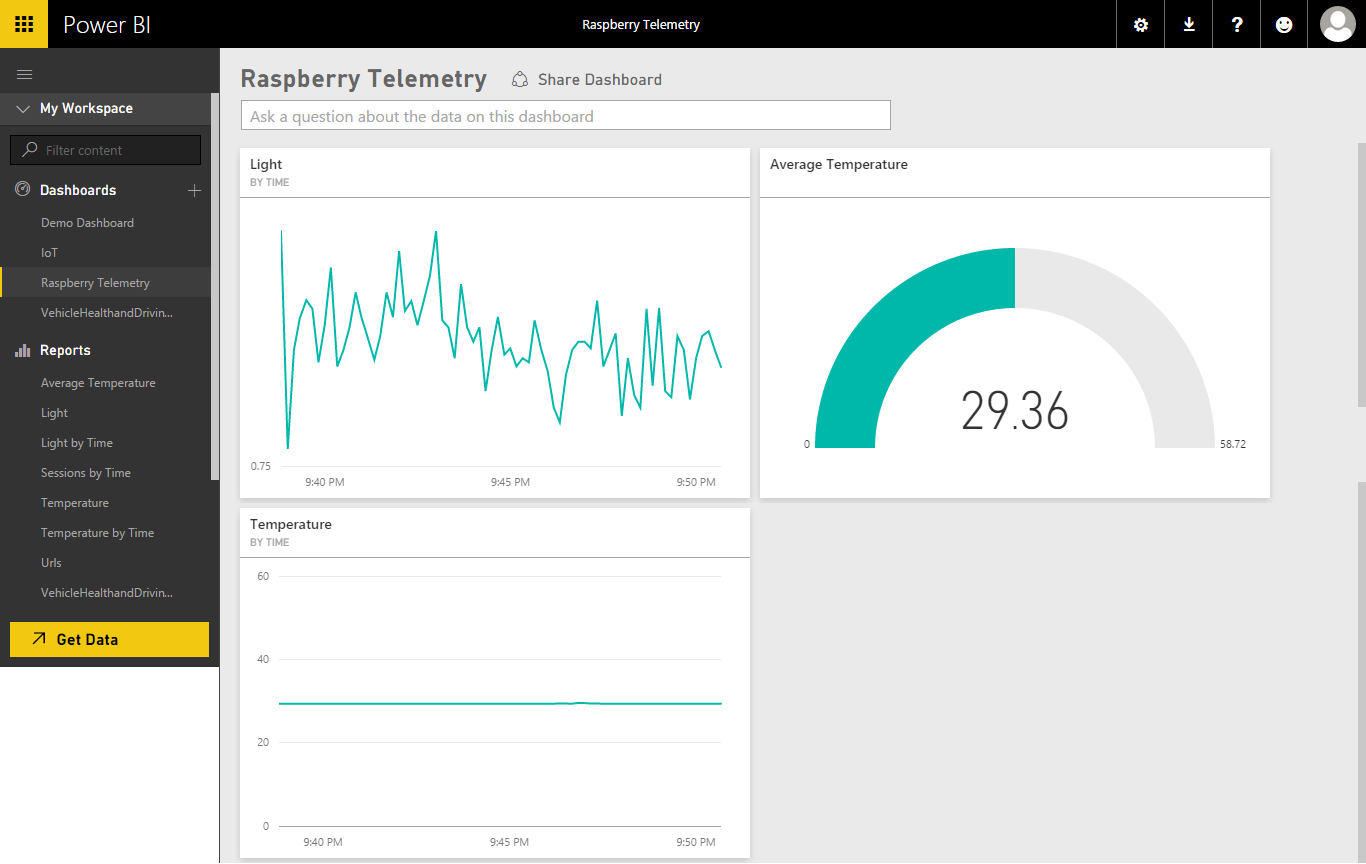
1. Click on the **Raspberry** datasource to create a new report.
2. Drag the **temperature** field to the Value Data Field.
3. Now change the visualization to a **gauge** chart
4. Change the **Value** from **Sum** to **Average**
5. Following the same directions, create a TEMPERATURE report and add it to the dashboard.
6. Lastly, edit the reports name in the dashboard by clicking the pencil icon next to each report.



[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/edit-report-title.png?raw=true)

Editing the Report Title

After renaming both reports you will get a dashboard similar to the one in the following screenshot, which will be automatically refreshed as new data arrives.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/final-power-bi-dashboard.png?raw=true)

Final Power BI Dashboard

1. Try the Power Bi app available in the iOS App store, Google Play and Windows Store.

EXPERIMENT 9: Controlling a device from Azure IoT Hub

Azure IoT Hub is a service that enables reliable and secure bi-directional communications between millions of IoT devices and an application back end. In this section you will see how to send cloud-to-device messages to your device to command it to change the colour of one of the FEZ HAT LEDs, using the Device Explorer app as the back end.

Azure IoT Hub supports a number of protocols including [AMQP](https://en.wikipedia.org/wiki/AMPQ), HTTPS and [MQTT](https://en.wikipedia.org/wiki/MQTT).

* **STEP 1:** Type the following code in the **Commanding\_CommandReceived** method between the **#region IoT Hub Command Support** tags **OR** using a code snippet type **lab8** and press Tab twice.

#region IoT Hub Command Support

char cmd = e.Item.Length > 0 ? e.Item.ToUpper()[0] : ' '; // get command character sent from IoT Hub

switch (cmd)

{

case 'R':

publishColor = FEZHAT.Color.Red;

break;

case 'G':

publishColor = FEZHAT.Color.Green;

break;

case 'B':

publishColor = FEZHAT.Color.Blue;

break;

case 'Y':

publishColor = FEZHAT.Color.Yellow;

break;

case 'M':

publishColor = FEZHAT.Color.Magneta;

break;

default:

System.Diagnostics.Debug.WriteLine("Unrecognized command: {0}", e.Item);

break;

}

hat.D3.Color = publishColor;

#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** and wait for the solution to deploy.
* **STEP 3:** From Device Explorer select the Messages to Device Tab, select your device from the Device ID: dropdown and in the Message field type a colour. Valid colours are Red, Green, Blue or Yellow – or just the first letter of a colour.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/sending-cloud-to-device-message.png?raw=true)

1. After a few seconds the message will be processed by the device and the LED will turn on in the colour you selected. The feedback will also be reflected in the Device Explorer screen after a few seconds.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/cloud-to-device-message-received.png?raw=true)

Congratulations, all done!

Experiment 10: Not so Fast Cowboy

Congratulations, you have successfully completed the Maker Den Experience. You have done some hardware prototyping, deployed a Universal Windows App to a Raspberry Pi 2. You have streamed data to Microsoft Azure, ingested telemetry using Azure IoT Hub and visualised data with the [Azure IoT Suite Remote Monitoring](https://github.com/Azure/azure-iot-remote-monitoring/wiki) solution. Optionally you also streamed data over MQTT and visualised on the IoT Dashboard.

Please complete the following steps before you leave.

* **STEP 1**: Close Visual Studio
* **STEP 2:** **Take 30 seconds to complete the Maker Den Evaluation at** <http://aka.ms/ignite2015makerden> (there is a shortcut on the desktop)

All the documentation and software for the Maker Den is available at <http://www.github.com/makerden>

Appendix:

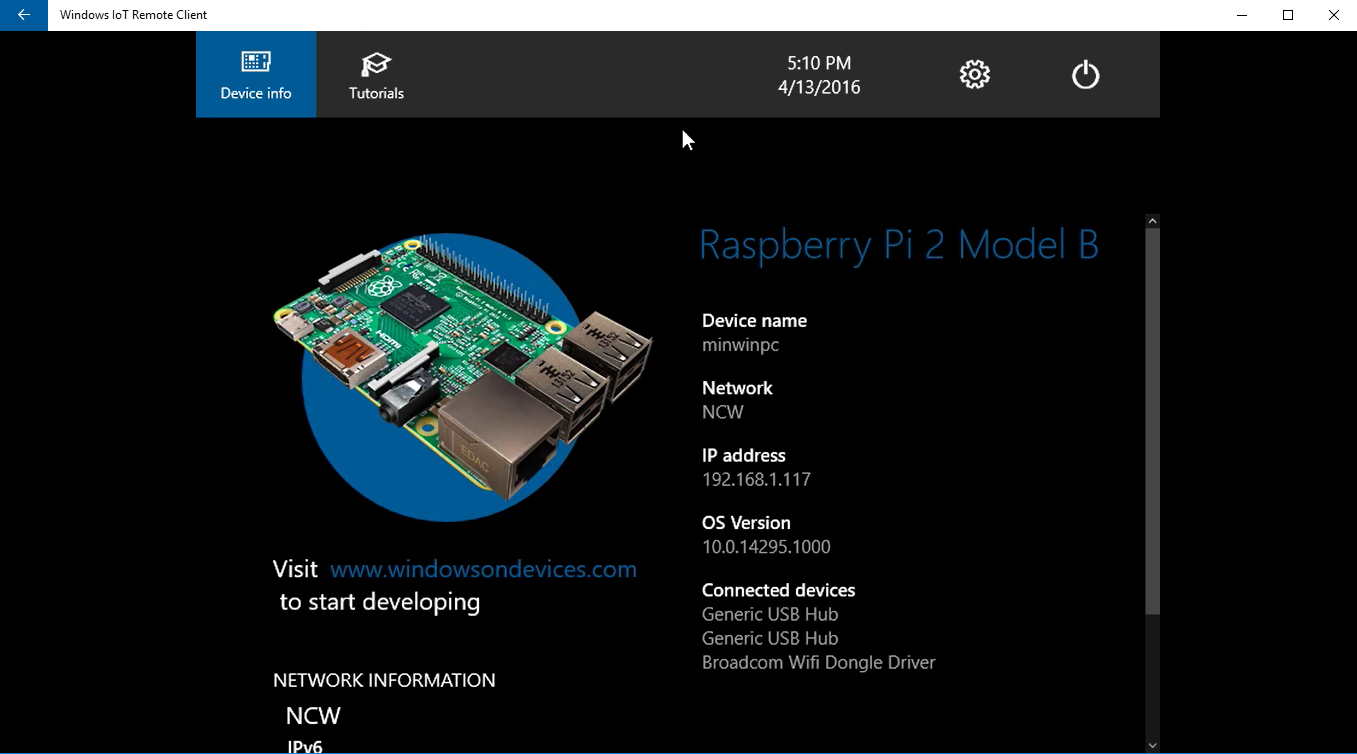
Windows IoT Remote Client connection

Press the Windows key and type “Windows IoT Core Remote Client” and run the app.

It is likely that you will 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.

Minimize the remote client application when you have verified that it is working.

[](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module1-IntroWindows10IoTCore/Images/windows-iot-remote-client.png?raw=true)

## TroubleShooting

w32tm /resync

1. **Note:** The Device Explorer is an Open Source sample. In production you would integrate device provisioning in to your solution. See [Get started with Azure IoT Hub for .NET](https://azure.microsoft.com/en-us/documentation/articles/iot-hub-csharp-csharp-getstarted/#create-a-device-identity). [↑](#footnote-ref-2)