

INTERNET OF THINGS DEN LAB GUIDE



Windows 10 IoT Core

Internet of Things Den with FEZ HAT Lab Guide

Document Version 3.1 Lite

This Lab assumes: -

- 1) An Azure account and an IoT Hub have been provisioned. See [Windows IoT Core Lab Setup.docx](#) for more information
- 2) The Lab Supplement data has been provided. See the appendix for required data.

Social	Twitter #iotden #iot #raspberrypi #windows10 #azure
Document Authors	Dave Glover dglover@microsoft.com @dglover Andrew Coates acoat@microsoft.com @coatsy Fai Lai hoongfai@microsoft.com @faister
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 IoT 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 deploying code, streaming sensor data to Microsoft Azure, aggregating data with Stream Analytics and reporting with Microsoft Power Bi.

GETTING STARTED

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

TIME REQUIRED

There are two sections to this lab. The first section is device centric and will take less than 15 minutes. Section 2 and beyond are 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 IoT Den on Twitter. Use hash tags #iotden #iot #raspberrypi #windows10 #azure

LAB HARDWARE

The following components are used for the IoT 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>.



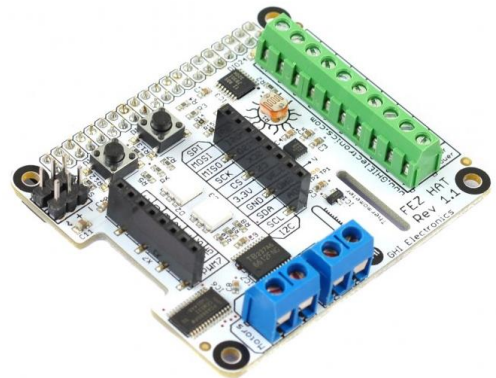
GHI electronics FEZ HAT

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

✕ There are ten IoT Den experiments to get you started with Windows 10 IoT Core and Microsoft Azure IoT Services.

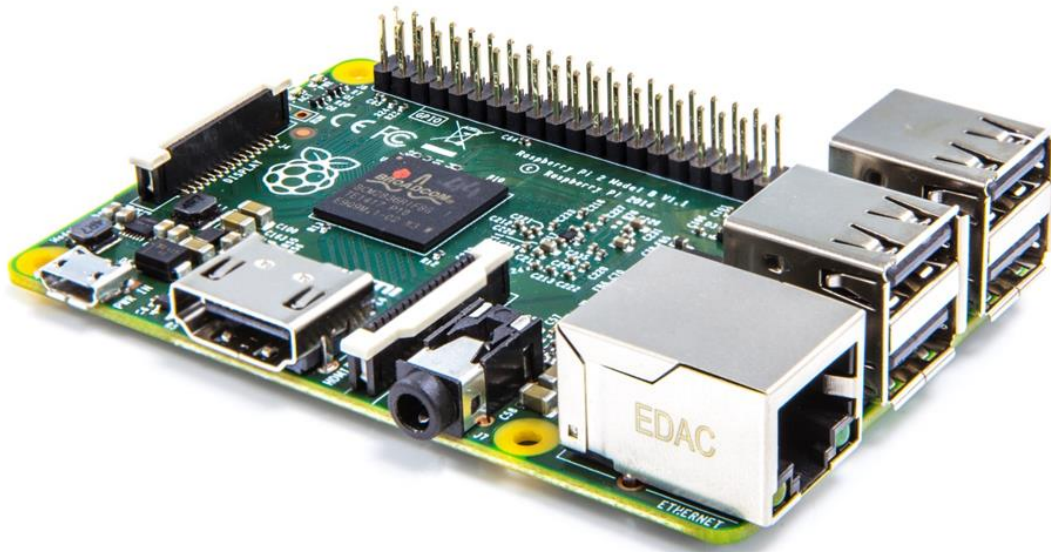
✕ All the source code can be referenced from the Source Code folder on the Desktop.

✕ This user guide 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

Deploying your first app

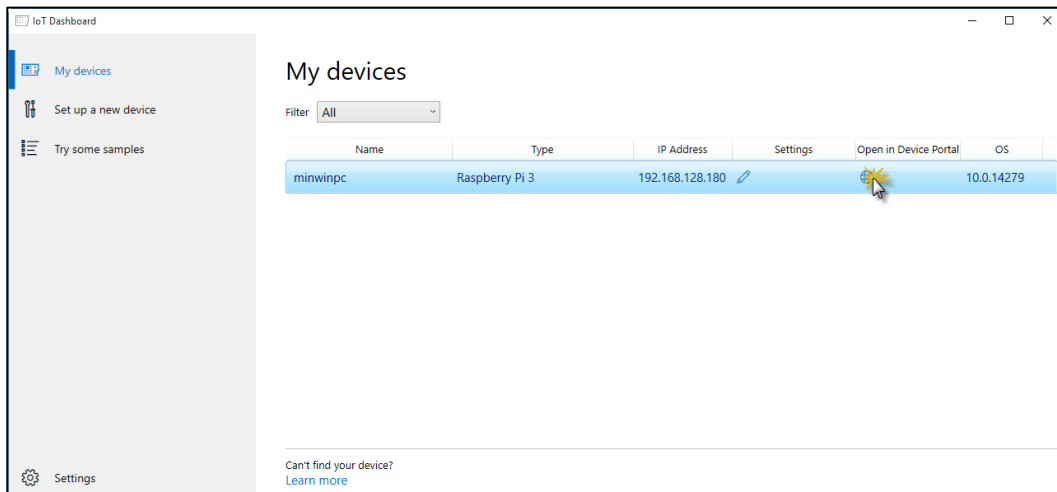
Sensing ambient light levels

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 devices² and click the Open in Device Portal icon for your chosen device.



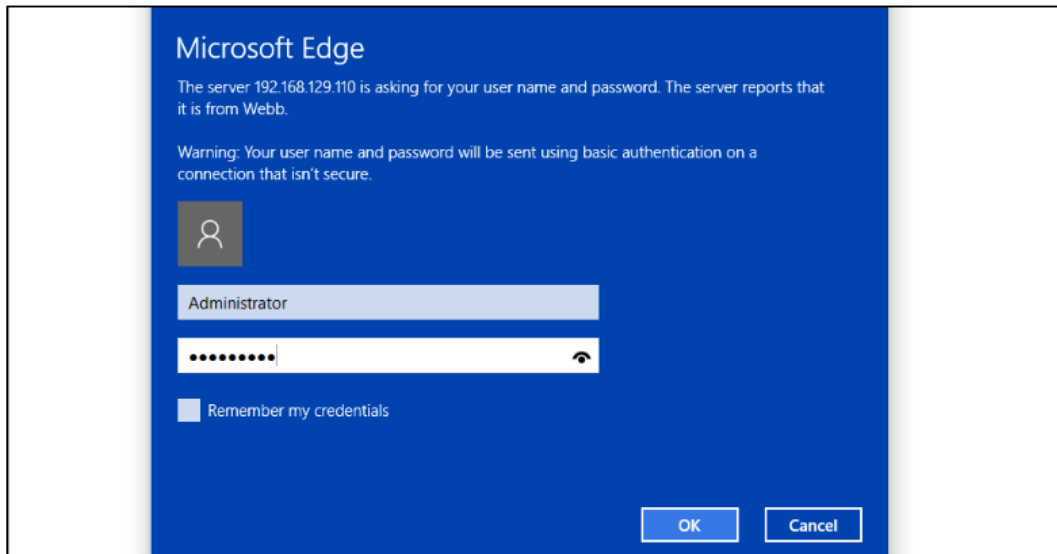
If your device does not show up in the list it is almost certainly because the network connection between your PC and the Raspberry Pi is public and Device Discovery is not enabled. See [How to change Windows 10 network location from Public to Private](#).

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

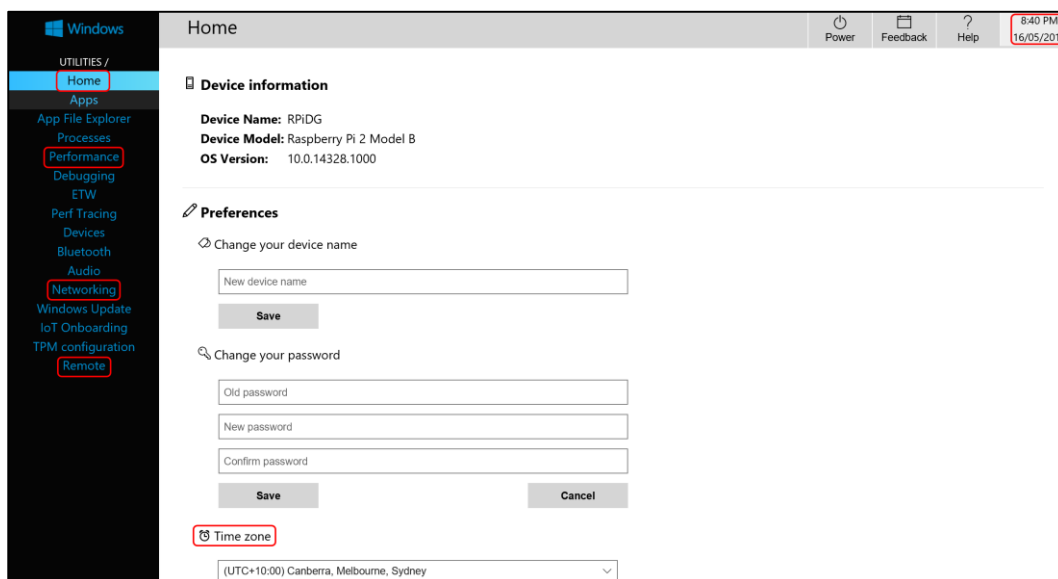
¹ You can download the Windows 10 IoT Core Dashboard from <https://developer.microsoft.com/en-us/windows/iot/getstarted>

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 data 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 Windows IoT Remote Client connection.

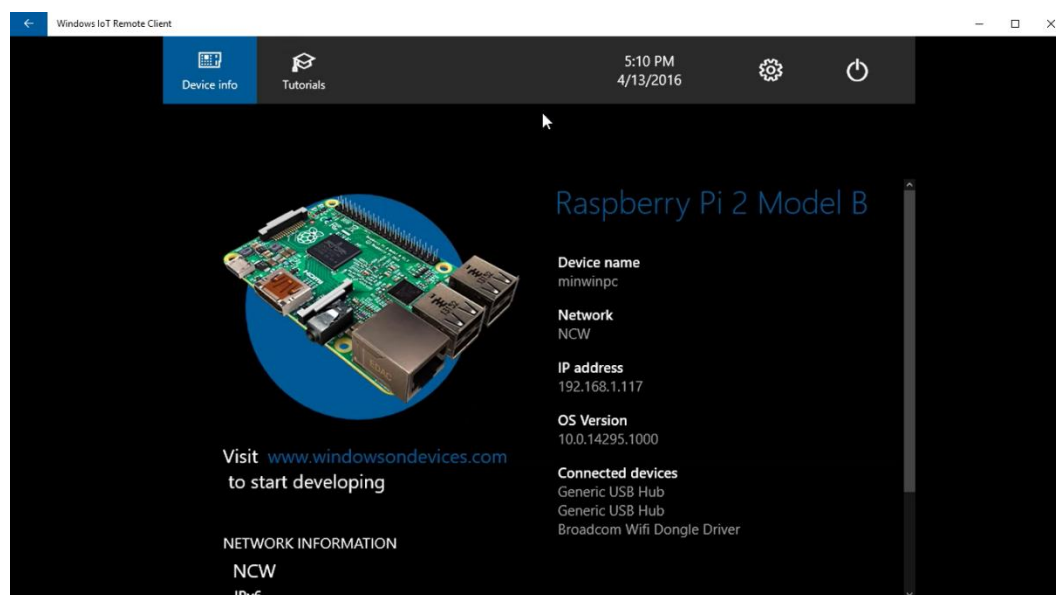
Press the Windows key and type “Windows IoT Core Remote Client”⁴ and run.

✉ **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.

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



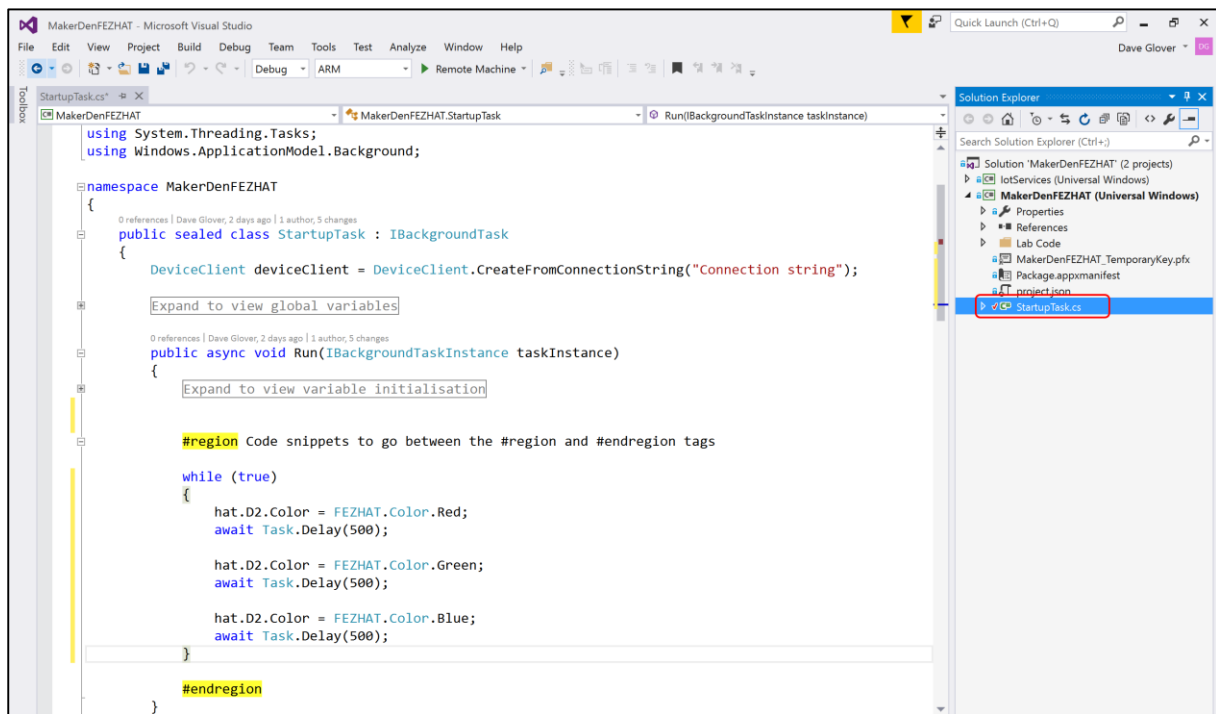
³ The Windows IoT Remote Server does take additional CPU cycles on the Raspberry Pi so depending on what you are doing you may want to disable the Windows Remote Server from the Windows Device Portal.

⁴ The [Windows IoT remote Client](#) is available from the Windows Store.

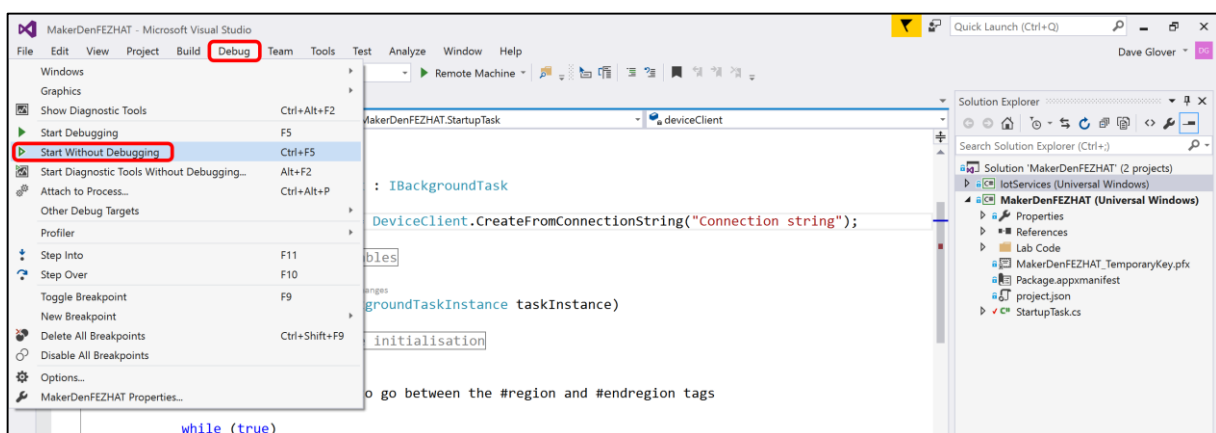
EXPERIMENT 2: HELLO WORLD

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

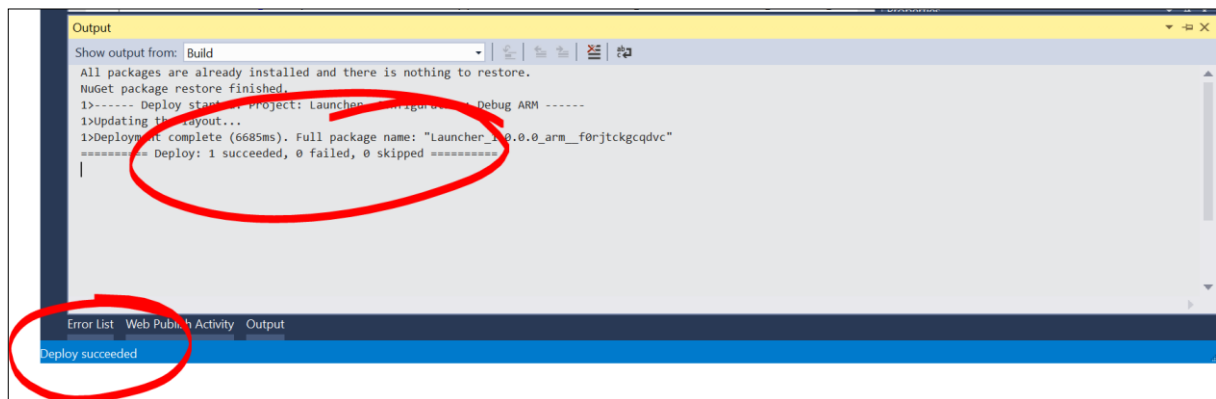
☒ **STEP 1:** Expand the **MakerDen** project then double click the **StartupTask.cs** file to open it.



☒ **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.

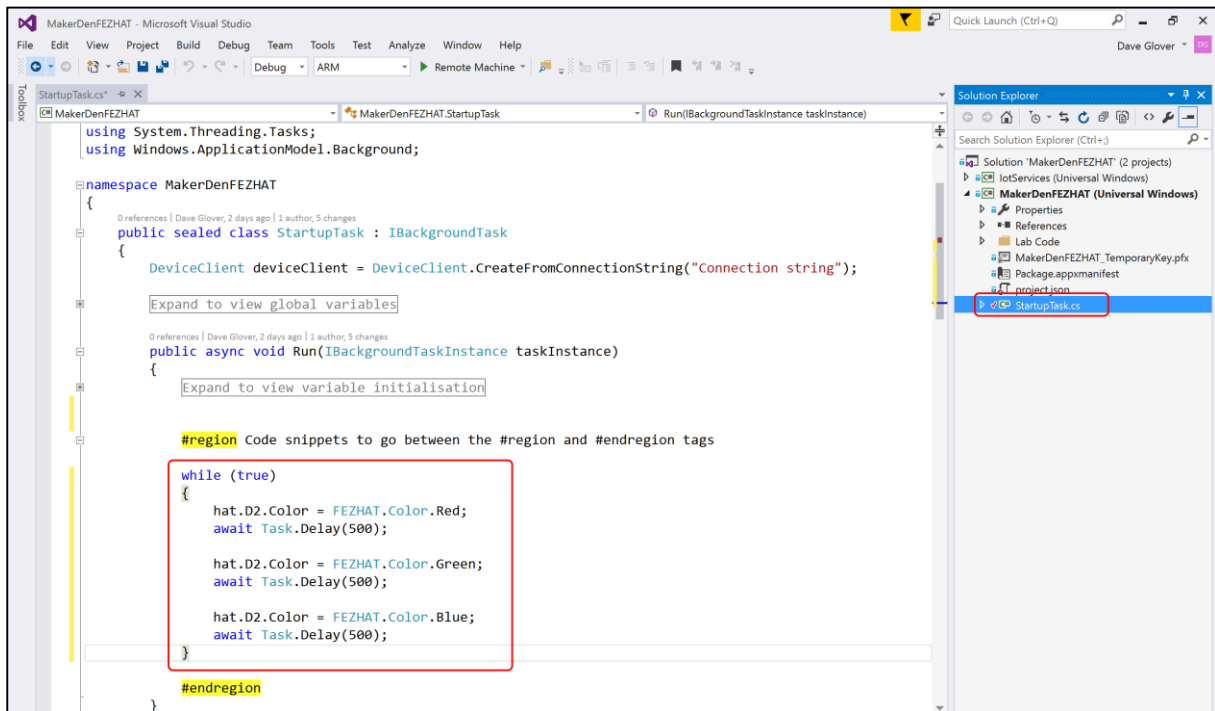
☒ **STEP 5:** Pat yourself on the back, you did it 😊

EXPERIMENT 3: SENSING THE WORLD

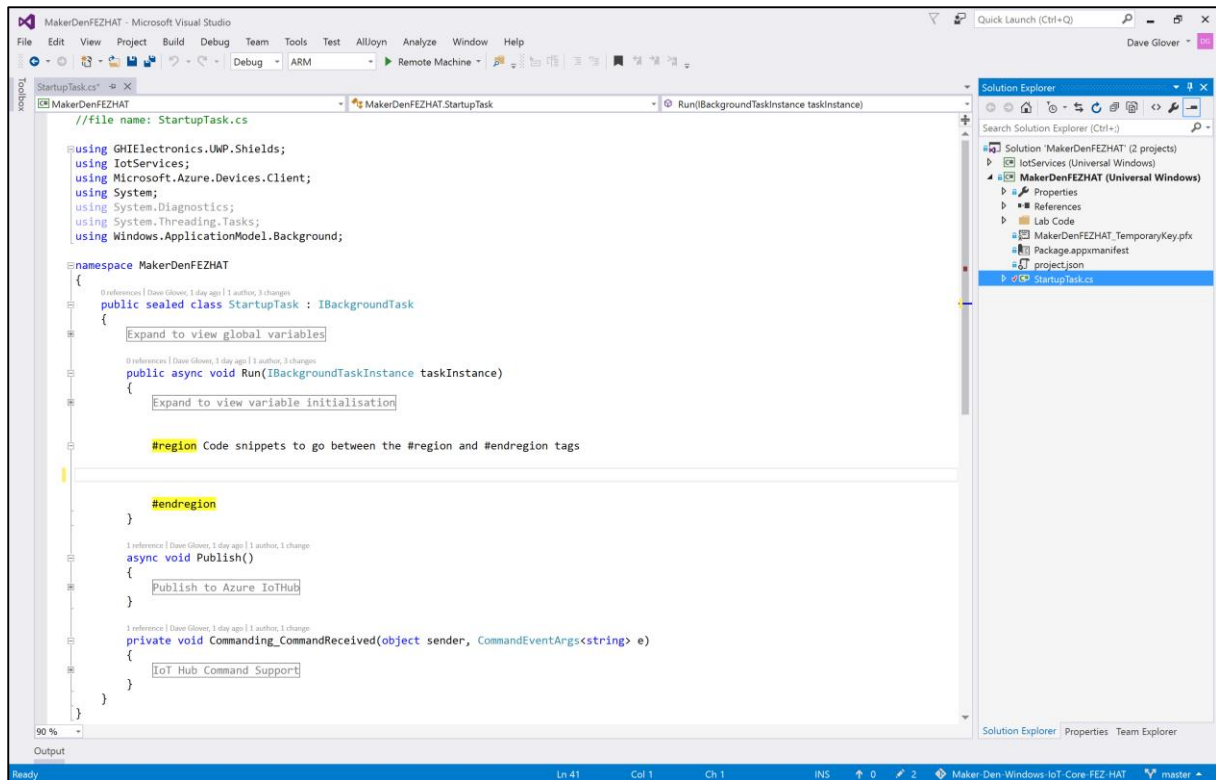
This lab reads the ambient light levels from the light sensor.

✕ **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 **lab1** and press Tab twice.

```
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 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 for labs 1 & 2 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 Publish to Azure IoT Hub
                #endregion
            }

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

```

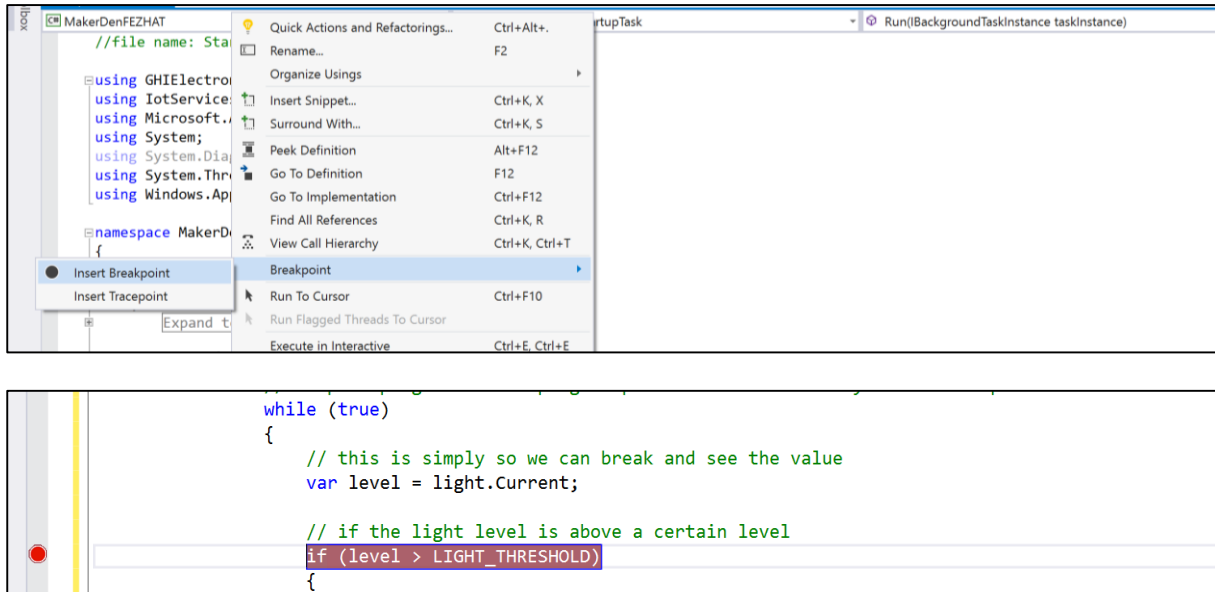
☒ **STEP 5:** Deploy the solution to the Raspberry Pi. From the **Debug** menu select **Start Without Debugging** or from the keyboard press **Ctrl+F5**.

☒ **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 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 changes colour depending on ambient light levels.

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

Section 2



Microsoft Azure Cloud Development

Registering a device with IoT Hub

Stream Analytics

Power BI

EXPERIMENT 5: REGISTERING YOUR DEVICE WITH AZURE IOT HUB

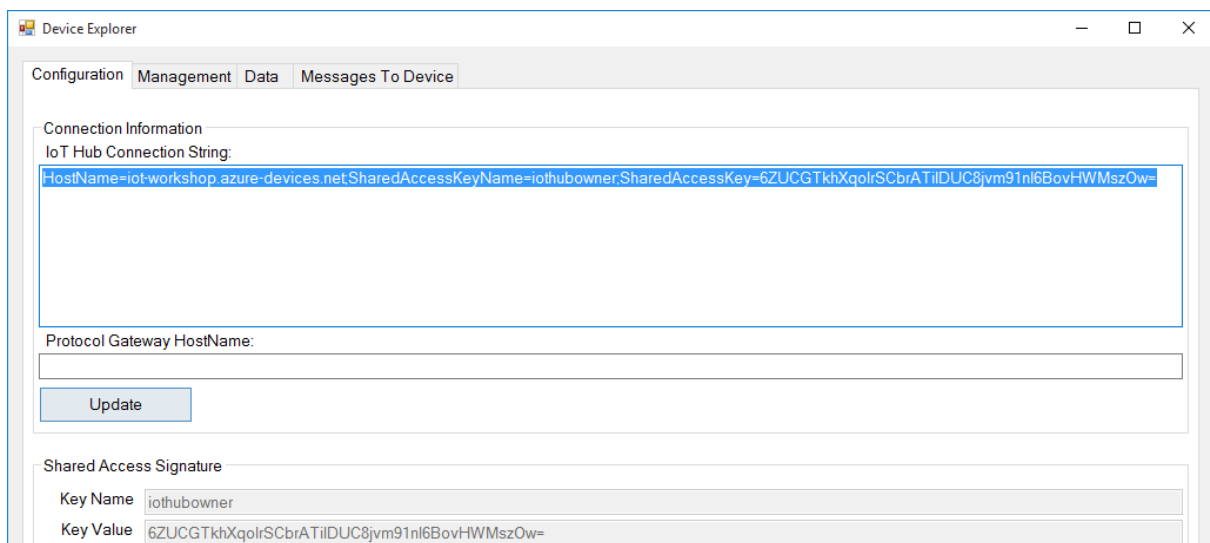
This experiment assumes that you have either been provided with or created your own Azure account and IoT Hub Service. See “[Windows IoT Core Lab Setup.docx](#)” in the Lab GitHub repository.

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](#) in the IoT Hub.

✖ **STEP 1:** Press the Windows key and type “Device Explorer”⁵ and run the app.

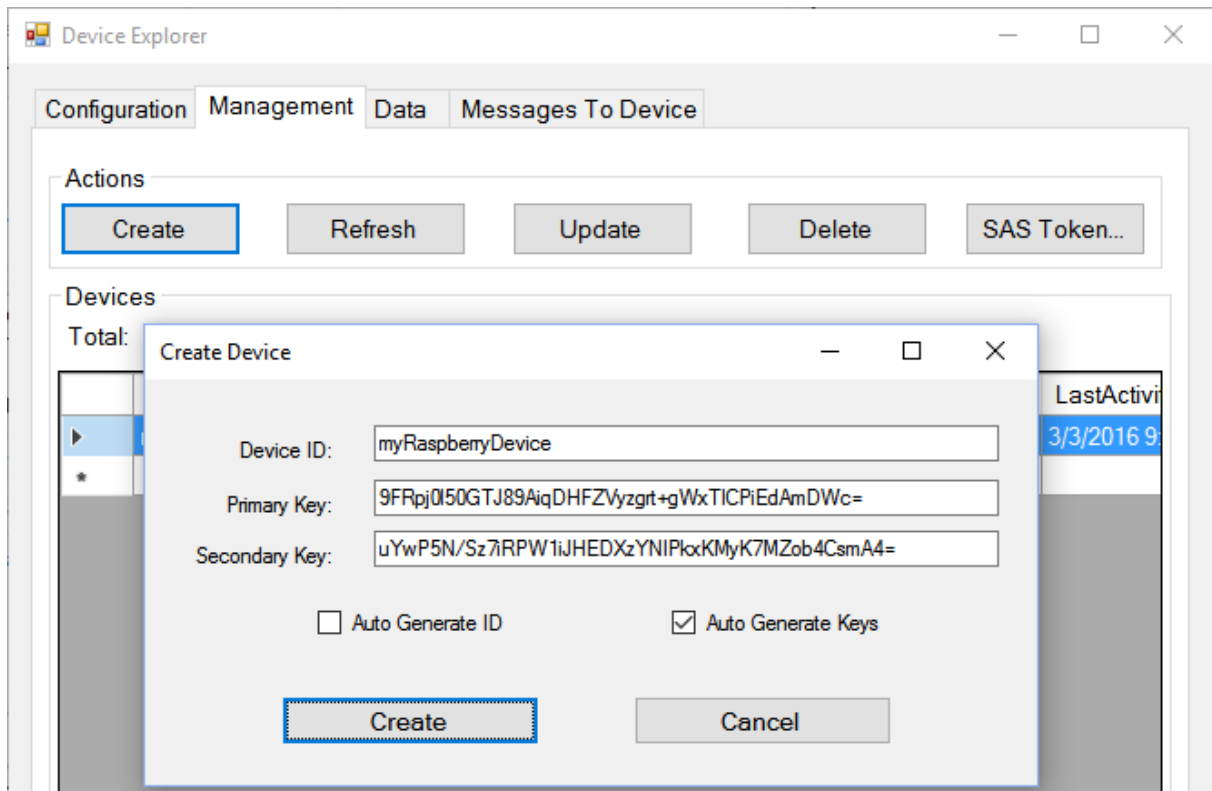
If “Device Explorer” is not installed, then install it from <https://github.com/Azure/azure-iot-sdks/releases> (Scroll down for SetupDeviceExplorer.msi).

✖ **STEP 2:** Paste the **IoT Hub Connection String** provided to you in the Lab Supplement in to the IoT Connection String field and click **Update**.



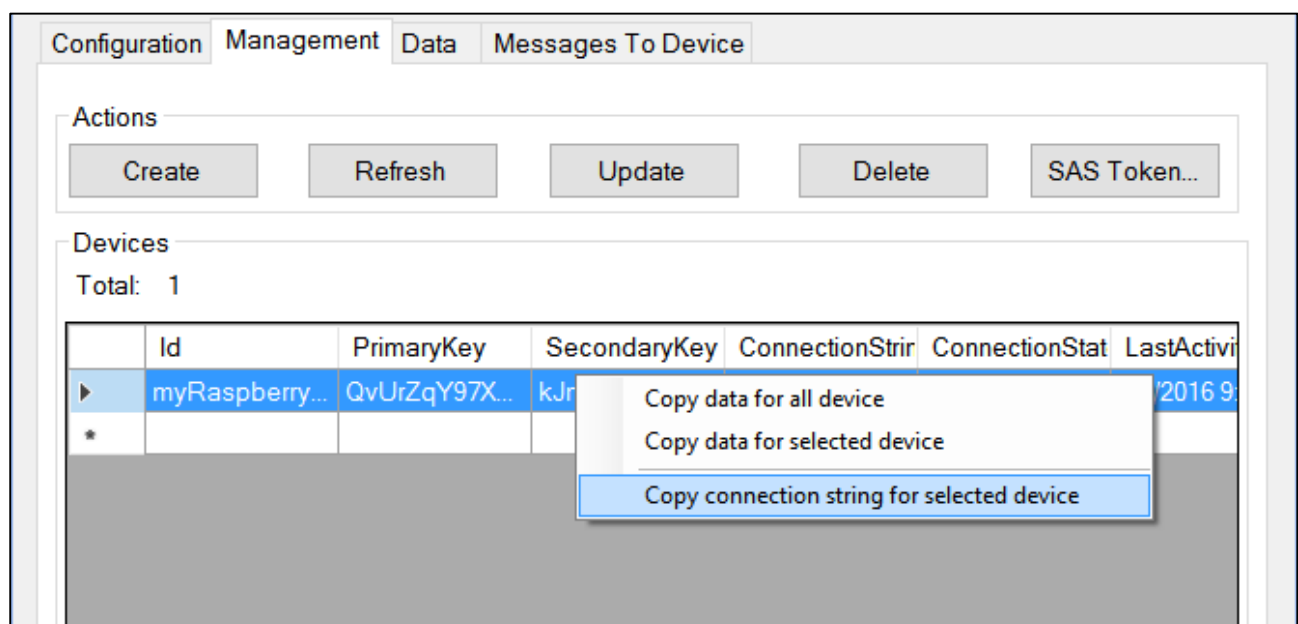
✖ **STEP 3:** Go to the **Management** tab and click on the **Create** button. The Create Device popup will be displayed. Fill the **Device ID** field with a new id for your device. For example, MyRPi01, then click on Create.

⁵The 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](#).



✳️ **STEP 4:** 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**, the connection string will be copied to the clipboard.

This unique connection string allows a device to authenticate and communicate securely with Azure IoT Hub.



Section 3



Azure Stream Analytics

Gain real-time insights from devices, sensors, infrastructure, and applications

EXPERIMENT 6: STREAMING TELEMETRY DATA TO AZURE IOT HUB

- ☒ **STEP 1:** Return to Visual Studio
- ☒ **STEP 2:** Ensure debugging has stopped (Shift-F5) otherwise you will not be able to make any edits to the StartupTask.cs file.
- ☒ **STEP 3:** Paste the Connection String in to the highlighted area in the StartupTask.cs file.

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

        #region Expand to view global variables

        public async void Run(IBackgroundTaskInstance taskInstance)
```

- ☒ **STEP 4:** Type the following code in the **Publish** method between the **#region Lab 6 - Publish to Azure IoT Hub** tags **OR** using a code snippet type **lab6** and press Tab twice.

The Publish method is responsible for reading sensors and streaming data to Azure IoT Hub. The publish method is called by the Telemetry class every 10 seconds.

```
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() * 100; // 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 5:** Your completed Publish method should look like this.

```
async void Publish()
{
    #region Lab 6 - Publish to Azure IoT Hub

    try // Exception handling if problem streaming telemetry to Azure IoT Hub
    {
        hat.D3.Color = publishColor; // turn on publish indicator LED

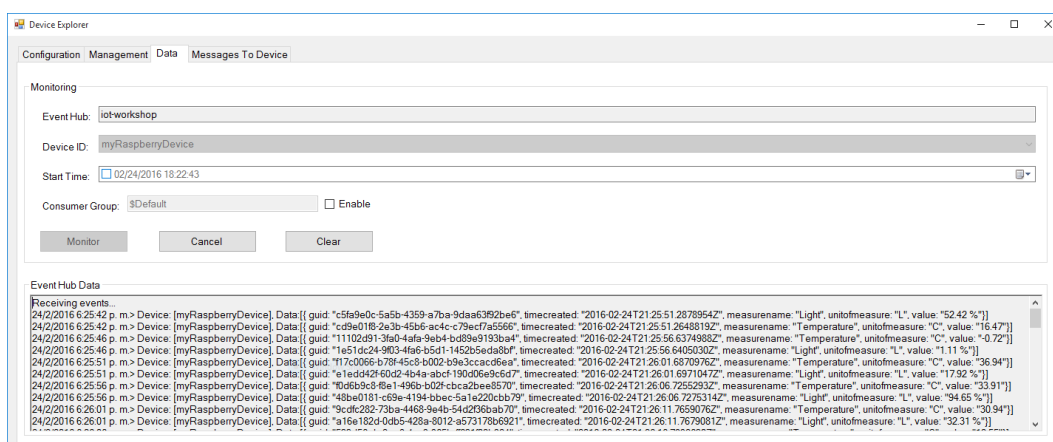
        var temperature = hat.GetTemperature(); // read temperature from the FEZ HAT
        var light = hat.GetLightLevel() * 100; // read light level from the FEZ HAT
        var json = telemetry.ToJson(temperature, light, 0, 0); //serialise to JSON

        var content = new Message(json);
        await deviceClient.SendEventAsync(content); //Send telemetry data to IoT Hub
    }
    catch { telemetry.Exceptions++; }
    finally { hat.D3.TurnOff(); }

    #endregion
}
```

✉ **STEP 6:** 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 7:** Press the Windows key and type “Device Explorer” and run. Navigate to the Data tab and select your device from the dropdown and click on Monitor.



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: INSIGHT AND IOT HUB DATA

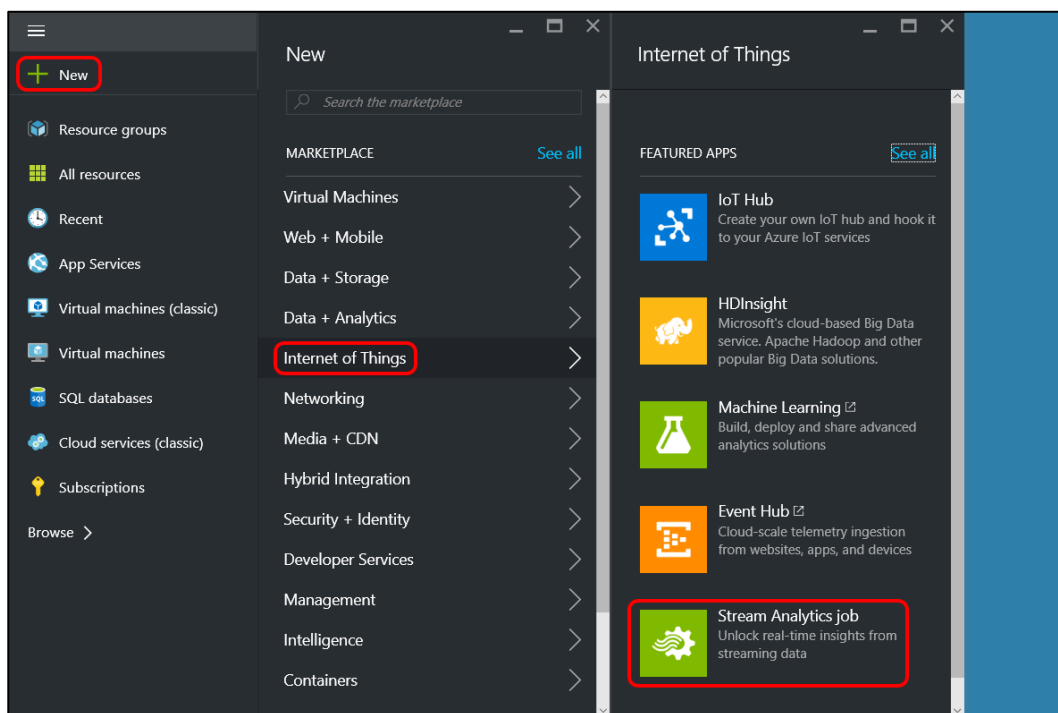
You have used the Device Explorer to view data streamed to the Azure IoT Hub. However, there are many ways to gain insight from the data including Stream Analytics, Power Bi, [Azure IoT Suite Remote Monitoring](#), and of course your own custom solution.

In the following section you will use Azure Stream Analytics in combination with Microsoft Power BI to consume IoT data and to generate meaningful reports.

CREATE A STREAM ANALYTICS JOB

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

✕ **STEP 1:** From the Azure Management Portal, click New -> Internet of Things -> Stream Analytics job.



✖ **STEP 2:** Define the Stream Analytics job.

Complete the form

Job Name
Provided in the Lab Supplement

Resource Group
Use existing IoTCamp

Location
Suggest selecting the region closest to your IoT Hub

New Stream Analytics Job

* Job name
See Lab Supplement

* Subscription
Visual Studio Ultimate with MSDN

* Resource group ⓘ
☐ Create new ☒ Use existing

IoTCamp

* Location
Australia East

☐ Pin to dashboard

Create Automation options

✖ **STEP 3:** Open the newly created Stream Analytics job configuration blade.

Click **All resources** -> **Your newly created Stream Analytics Job**.

+ New

Resource groups

All resources

Recent

App Services

Virtual machines (classic)

Virtual machines

SQL databases

Cloud services (classic)

All resources

Default Directory

+ Add ≡ Columns ↺ Refresh

Subscriptions: Visual Studio Ultimate with MSDN – Don't see a subscription? [Switch directory](#)

Filter items...

NAME	TYPE
gloveboxAE	Cloud service (classic)
gloveboxAE	Virtual machine (clas...
iot-sa-nn	Stream Analytics job
MakerDen	Service bus namesp...

- ✳️ **STEP 4:** Familiarise yourself with the Stream Analytics blade. Note the Inputs, Query and Output zones.

As you can see, the Start button is disabled since the job is not configured yet.

The screenshot shows the Azure Stream Analytics job configuration page for a job named 'iot-sa-nn'. The page has a dark header with the job name and a toolbar with 'Settings', 'Start', 'Stop', and 'Delete' buttons. The 'Start' button is disabled. Below the header is a blue bar with an information icon and the word 'Created'. The main content area is divided into two sections: 'Essentials' and 'Job Topology'. The 'Essentials' section displays job details: Resource group (IoTCamp), Status (Created), Location (Australia East), Subscription name (Visual Studio Ultimate with MSDN), and Subscription ID (8df98239-76ae-47c9-b464-16b680ad654a). It also includes a 'Send feedback' link (UserVoice), creation time (Monday, July 4, 2016 8:45:08 PM), and last output (-). The 'Job Topology' section shows three panels: 'Inputs', 'Query', and 'Outputs'. Each panel has a red border and a '0' with a refresh icon, indicating no results. The 'Query' panel shows a blue double arrow icon. The 'Monitoring' section at the bottom shows a list of metrics: 'InputEvents, OutputEvents and one more metric past hour'.

iot-sa-nn
Stream Analytics job

Settings Start Stop Delete

Created

Essentials

Resource group
IoTCamp

Status
Created

Location
Australia East

Subscription name
Visual Studio Ultimate with MSDN

Subscription ID
8df98239-76ae-47c9-b464-16b680ad654a

Send feedback
UserVoice

Created
Monday, July 4, 2016 8:45:08 PM

Started
-

Last output
-

All settings →

Job Topology

Add tiles (+)

Inputs

0

No results.

Query

Outputs

0

No results.

Monitoring

Add tiles (+)

InputEvents, OutputEvents and one more metric past hour

STREAM ANALYTICS - CONFIGURE NEW INPUT

The New Input blade defines the source of data for the Stream Analytics job.


✉ **STEP 1:** From the Stream Analytics Configuration blade select **Inputs** and define the input parameters.

Settings for New Input Blade	New input
<u>Input alias</u> TelemetryHub	<div><div>* Input alias ⓘ</div><div>TelemetryHub</div><div>!</div></div>
<u>Source</u> IoT hub	<div><div>* Source Type ⓘ</div><div>Data stream</div><div>▼</div></div>
<u>Subscription</u> Provide IoT hub settings manually	<div><div>* Source ⓘ</div><div>IoT hub</div><div>▼</div></div>
<u>IoT Hub</u> Provided in the Lab Supplement	<div><div>* Subscription</div><div>Provide IoT hub settings manually</div><div>▼</div></div>
<u>Shared access policy name</u> iothubowner	<div><div>* IoT hub ⓘ</div><div>See Lab Supplement</div><div>!</div></div>
<u>Shared access policy key</u> Provided in the Lab Supplement	<div><div>* Endpoint ⓘ</div><div>Messaging</div><div>▼</div></div>
<u>Consumer group</u> Provided in the Lab Supplement	<div><div>* Shared access policy name ⓘ</div><div>See Lab Supplement</div><div>!</div></div>
<u>Click Create</u> It will take a moment or two to create the Input stream.	<div><div>* Shared access policy key ⓘ</div><div></div><div>!</div></div>
	<div><div>Consumer group ⓘ</div><div>See Lab Supplement</div><div>!</div></div>
	<div><div>* Event serialization format ⓘ</div><div>JSON</div><div>▼</div></div>
	<div><div>Encoding ⓘ</div><div>UTF-8</div><div>▼</div></div>
	<div><div>Create</div></div>

STREAM ANALYTICS – CONFIGURE NEW OUTPUT

The New Output blade defines the output destination for the Stream Analytics job.

✉ **STEP 1:** From the Stream Analytics Configuration blade select **Outputs** and define the input parameters.

Settings for New Output Blade	New output
<u>Output alias</u> PowerBI	<div><div>* Output alias</div><div>PowerBI</div></div>
<u>Sink</u> Select the POWER BI	<div><div>* Sink ⓘ</div><div>Power BI</div></div>
<u>Authorize</u> The Power Bi credentials are provided in the Lab Supplement. Click Authorise and you will be redirected to the Microsoft login page to authorise output to your Power BI account	<div>Authorize Connection You'll need to authorize with Power BI to configure your output settings.</div> <div><div>Authorize</div></div> <div>Don't have a Microsoft Power BI account yet? Sign Up</div>
	<div><div></div><div><p>Note: You are granting this output permanent access to your Power BI dashboard. Should you need to revoke this access in the future you can do one of the following:</p><ol style="list-style-type: none">1. Change the user account password.2. Delete this output.3. Delete this job.</div></div> <div><div>Create</div></div>

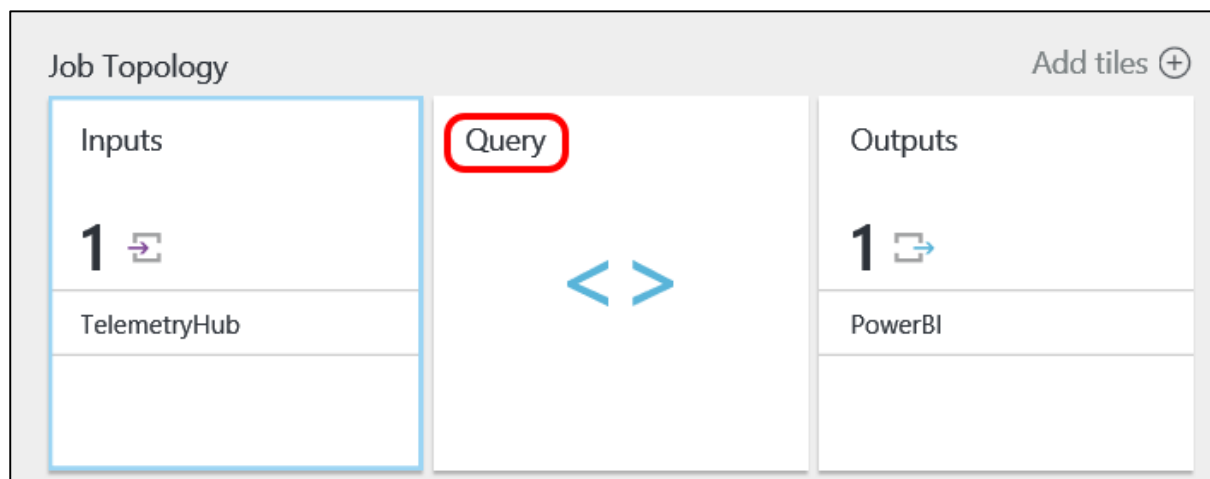
✕ **STEP 2:** Once Power BI is authorised continue setting the remaining configuration fields.

<p><u>Dataset Name</u> Provided in the Lab Supplement</p> <p><u>Table Name</u> Telemetry</p> <p><u>Click Create</u> It will take a moment or two to create the Output stream.</p>	<div>New output</div> <div><p>* Output alias PowerBI</p><p>* Sink ⓘ Power BI</p><p>Group Workspace My Workspace</p><p>* Dataset Name See Lab Supplement</p><p>⚠ If the dataset or table already exists in your Microsoft Power BI subscription, it will be overwritten.</p><p>* Table Name See Lab Supplement</p><p>Currently authorized as Dave Glover (DPE AUSTRALIA) (dglover@microsoft.com)</p><p>Create</p></div>
--	---

STREAM ANALYTICS - QUERY CONFIGURATION

Now that the job's inputs and outputs are 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.

✖ **STEP 1:** From the Stream Analytics Configuration blade click **Query**.

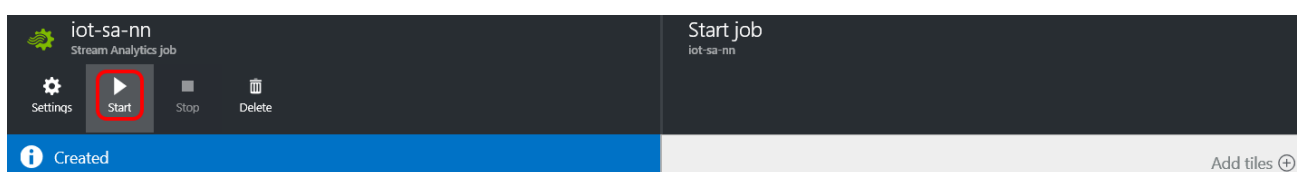


✖ **STEP 2:** Replace the default query with the following Stream Analytics Query.

```
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 in to 30 seconds chunks.

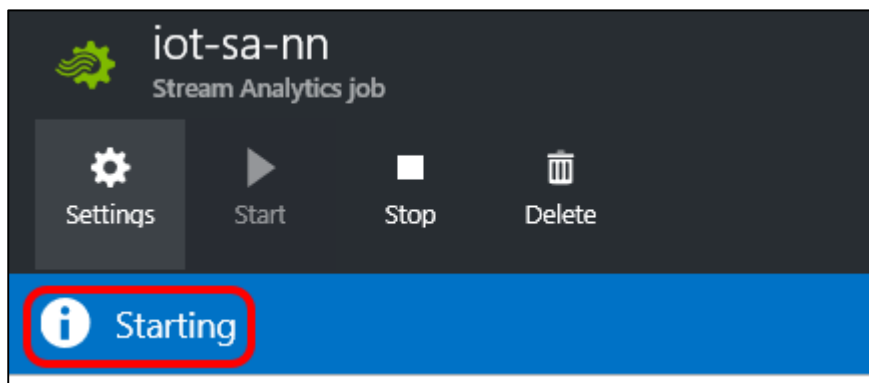
✖ **STEP 3:** Click the **SAVE** button.



STARTING THE STREAM ANALYTICS JOB

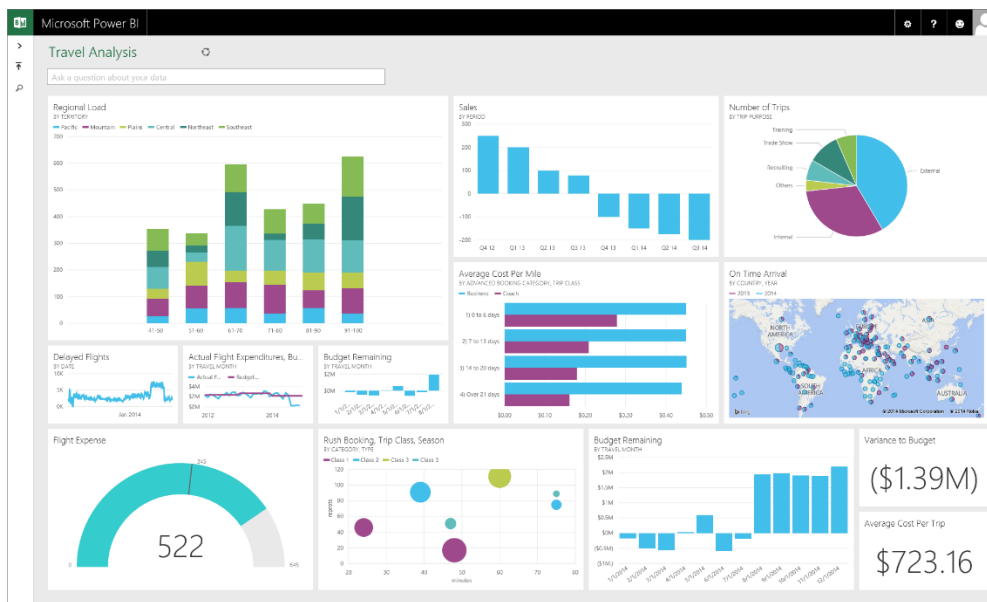
Now that the job is configured it needs to be started.

☒ **STEP 1:** From the Stream Analytics blade click **Start** -> **Now** -> **Start**. Allow for 30 to 60 seconds for the job to enter “**Running**” mode.



Once the job starts and it is receiving data from your IoT device it will create the Power BI datasource associated with the given subscription.

Section 4



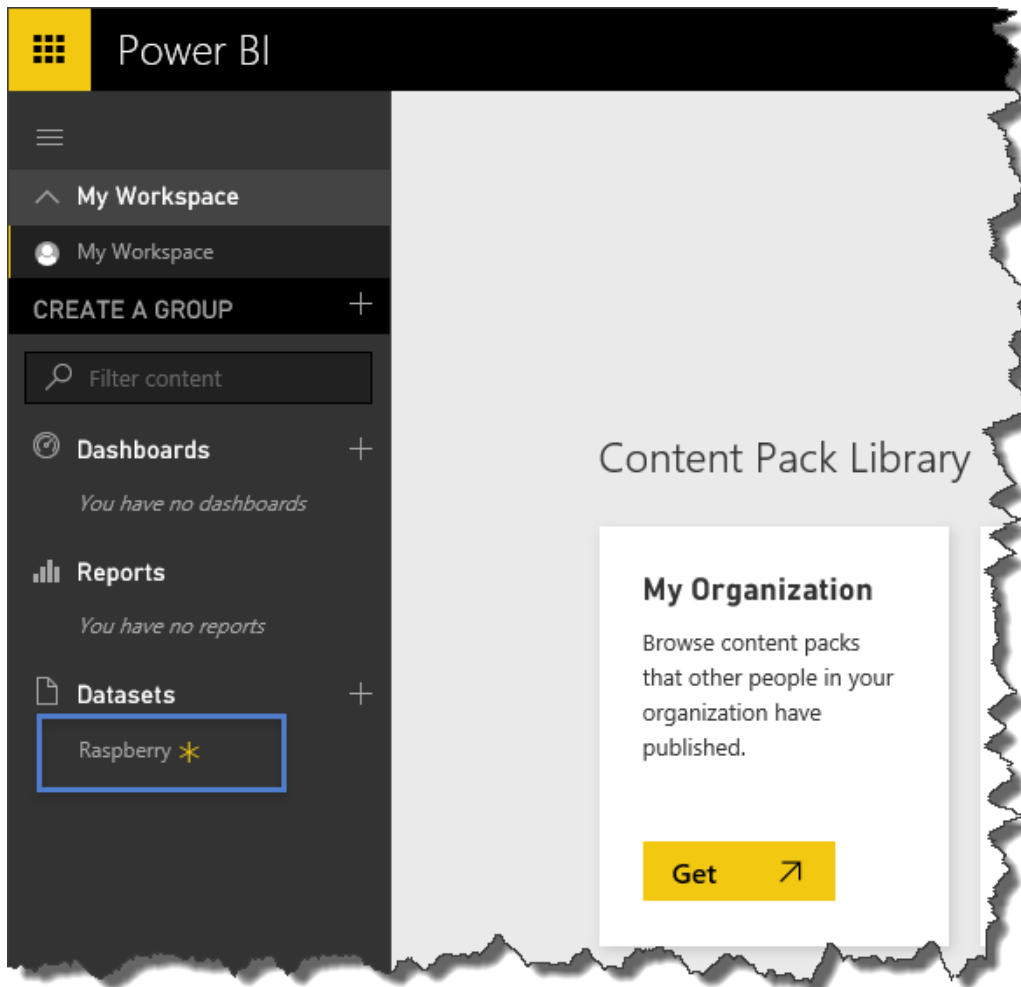
Microsoft Power BI

Transform your data with rich visuals for you to collect and organize

EXPERIMENT 8: SETTING UP THE POWER BI DASHBOARD

✉ **STEP 1:** Navigate to Power Bi (www.powerbi.com) and authenticate. Click the Hamburger to expand the navigation pane.

The Steam Analytics⁶ job needs to run for a few minutes before it appears in the navigation pane.



⁶The Power BI dataset will only be created if the job is running and if it is receiving data from the IoT Hub input. If there is no Raspberry dataset then check the Universal App is running on the Raspberry Pi and it is streaming data to Azure. To verify the Stream Analytics job is receiving and processing data you can check the Azure Management Stream Analytics monitor.

✳️ **STEP 2:** Click on the **datasource name that you created** and start defining the report.

DEFINING A POWER BI REPORT

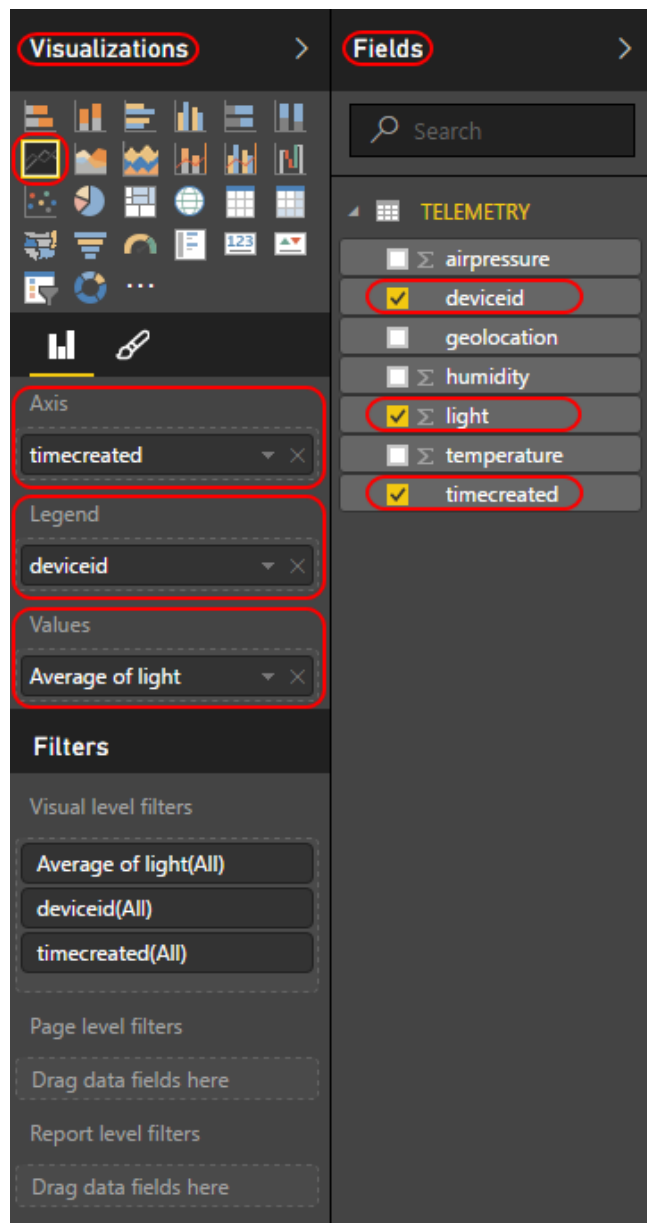
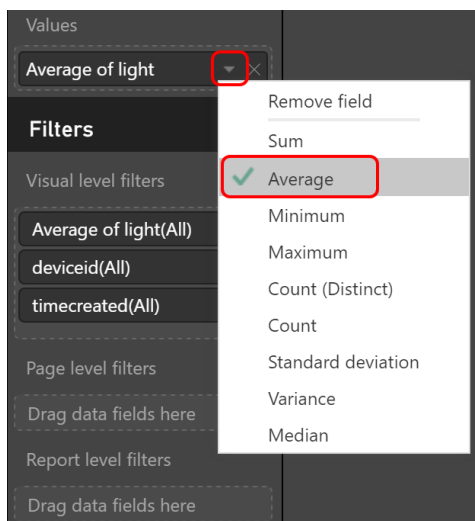
The Report designer will be opened showing the list of fields available for the selected datasource and the different visualizations supported by the tool.

✳️ **STEP 1:** Select Line Chart from the Visualizations

✳️ **STEP 2:** Drag and drop the following fields from the Fields section

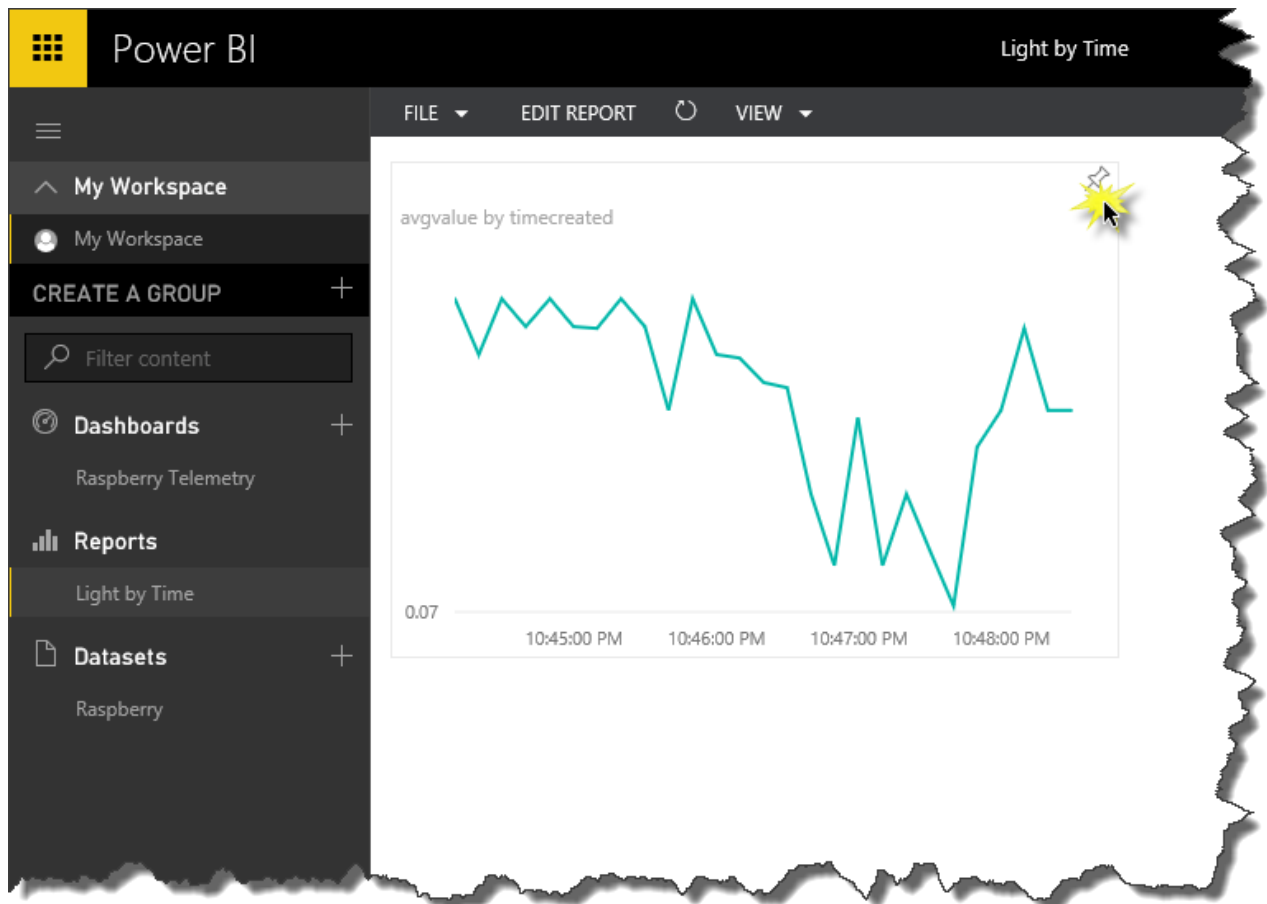
- 1) deviceid -> Legend
- 2) light -> Values
- 3) timecreated -> Axis

✳️ **STEP 3:** Select **Average of light** from the Values dropdown menu.



✳️ **STEP 4:** Click the **SAVE** button and set LIGHT BY TIME as the name for the report.

- ☒ **STEP 5:** Now create a new Dashboard, and pin this report to it. Click the plus sign (+) next to the **Dashboards** section to create a new dashboard and name it Raspberry Telemetry.
- ☒ **STEP 6:** Now, go back to your report and click the pin icon to add the report to the newly created dashboard.



- ☒ **STEP 7:** Experiment with other chart types and remember you have access to both Light and Temperature data in the Power BI reporting tool.

Section 5



Azure IoT Hub

Bi-directional Communications and
Command and Control

EXPERIMENT 9: CONTROLLING A DEVICE FROM AZURE IOT HUB

Azure IoT Hub is a service that enables reliable and secure bi-directional communications⁷ between millions of IoT devices and an application back end.

In this experiment we will send cloud-to-device messages to your device to command it to change the colour of one of the FEZ HAT LEDs. For the experiment Device Explorer will serve as the back end.

✉ **STEP 1:** Type the following code in the **Command_Processing** method between the **#region Lab 7 - IoT Hub Command Support** tags **OR** using a code snippet type **lab7** and press Tab twice.

⁷ Azure IoT Hub supports a number of protocols including [AMQP](#), HTTPS and [MQTT](#).

```

#region Lab 7 - IoT Hub Command Support

while (true)
{
    try
    {
        Message receivedMessage = await deviceClient.ReceiveAsync();
        if (receivedMessage == null)
        {
            await Task.Delay(2000);
            continue;
        }

        await deviceClient.CompleteAsync(receivedMessage);

        string command = Encoding.ASCII.GetString(receivedMessage.GetBytes()).ToUpper();
        if (string.IsNullOrEmpty(command)) { continue; }

        switch (command[0])
        {
            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.Magenta;
                break;
            default:
                Debug.WriteLine("Unrecognized command: {0}", command);
                break;
        }

        hat.D3.Color = publishColor;
    }
    catch { } // just keep going
}

#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.

✕ **STEP 4:** Enable **Monitor Feedback Endpoint** and click Send.

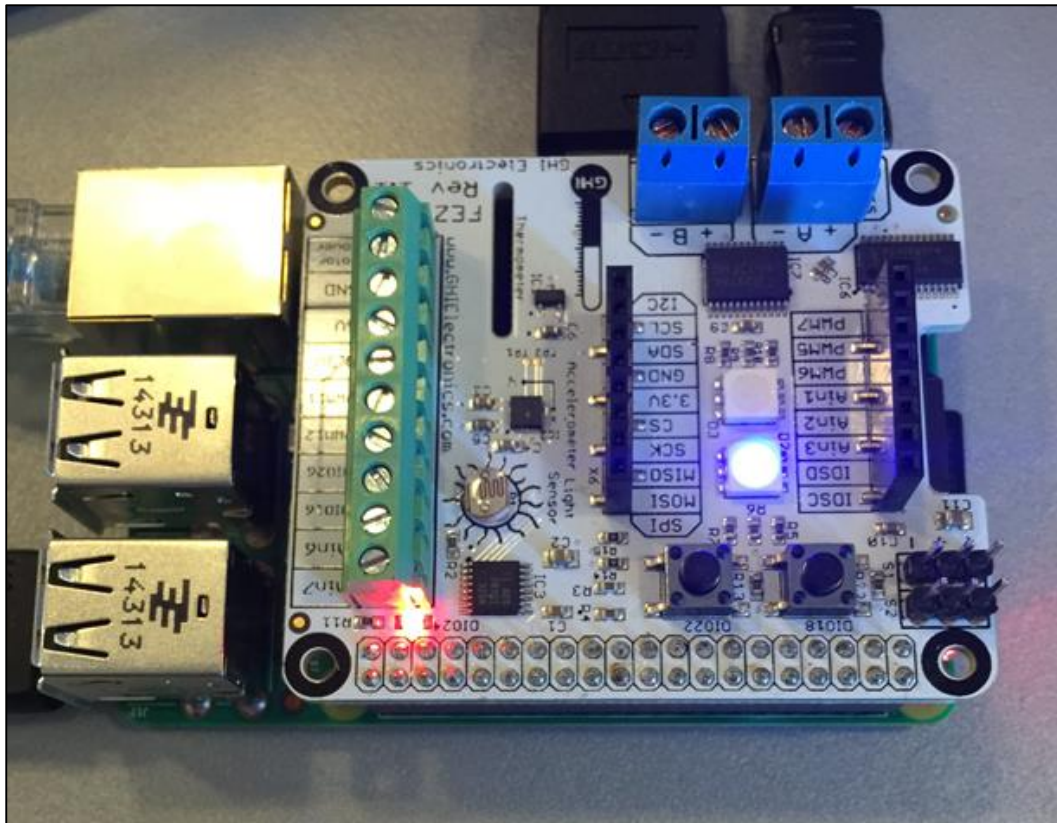
The screenshot shows the 'Device Explorer' application window with the 'Messages To Device' tab selected. The 'Send Message to Device:' section contains the following fields and controls:

- IoT Hub:** A text field containing 'iot-workshop'.
- Device ID:** A dropdown menu showing 'myRaspberryDevice'.
- Message:** A text field containing 'BLUE'.
- Options:** Two checkboxes are present: 'Add Time Stamp' (unchecked) and 'Monitor Feedback Endpoint' (checked).
- Buttons:** 'Send' and 'Clear' buttons are located below the options.

The 'Output' section at the bottom displays the following log messages:

```
Started monitoring feedback for device myRaspberryDevice.  
Sent to Device ID: [myRaspberryDevice], Message:"BLUE", message Id: 376b71c0-f6d7-4c62-885c-b6b7971bef51  
Message Feedback status: "Success", Description: "Success", Original Message Id: 376b71c0-f6d7-4c62-885c-b6b7971bef51
```

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.



Congratulations, you have finished!

EVALUATION

Congratulations, you have successfully completed the IoT Den Experience. You have deployed a Universal Windows App to a Raspberry Pi. You have streamed data to Microsoft Azure, ingested telemetry using Azure IoT Hub and visualised data with the Power Bi.

Please complete the following steps before you leave.

☒ **STEP 1:** Close Visual Studio.

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

APPENDIX

TROUBLESHOOTING

FORCING A TIME RESYNC

1. From “Windows 10 IoT Core Dashboard”, right mouse click your device and Connect using PowerShell
2. Authenticate
3. At the command prompt, type “w32tm /resync” and press the Enter key to execute.
4. Type Date and press the Enter key to verify date and time correct.

LAST BOOT DATE AND TIME

From PowerShell

```
wmic os get lastbootuptime
```

USEFUL NETWORK COMMANDS

From PowerShell

- netsh wlan show profile
- netsh wlan add profile *Wi-Fi-ProfileName.xml*
- netsh wlan export profile key=clear
- netsh wlan delete profile *ProfileName*
- netsh wlan connect name= *ProfileName*
- netsh wlan show interfaces
- netsh wlan delete profile *ProfileName*
- netsh wlan add profile Wi-Fi- *ProfileName.xml*
- netsh wlan connect name= *ProfileName*
- netsh interface ipv4 set dns "Wi-Fi" static 192.168.1.1
- netsh interface ipv4 set address "Wi-Fi" static 192.168.1.107 255.255.255.0 192.168.1.1

LAB SUPPLEMENT DATA

Azure Stream Analytics

- Azure authentication Email address and password
- Stream Analytics Job Name
- IoT Hub Consumer Group
- Shared access policy key
- Shared access policy key
- IOT HUB POLICY KEY
- Consumer group

Power BI

- Power Bi Email
- Power Bi Password
- Dataset Name
- IoT Hub Connection String