# AZURE IOT DEVICE CONTROL

A hack lab using the Azure IoT Hub

## Summary

This document details step-by-step a way to connect a Raspberry Pi to Azure IoT Hub and control it remotely using another computer with Cortana voice activation.

## Contents

## Prerequisites

A developer computer with Windows 10 for integrating with Cortana.

Visual Studio 2017 Community Edition with Universal Windows App Development Tools installed.

A micro SD card for the Raspberry Pi

Windows IoT Core installed on the micro SD card or Windows IoT Dashboard

A Raspberry Pi 2 or 3 with power supply or micro USB cable connected to your computer.

Two LED’s (Preferably green and red).

Two 270-330 ohm resistor.

Six female-female jumper connectors.

An Azure subscription with permission to add resources.

## Part 1 – Setting up the Raspberry Pi

### Windows 10 IoT Core

If you don't have Windows 10 IoT Core already installed on the SD card for the Raspberry Pi you can install it easily by using the Windows 10 IoT Core Dashboard.

Step by step instructions can be found here: https://docs.microsoft.com/en-us/windows/iot-core/connect-your-device/iotdashboard

### IoT Hub and device registration

An IoT Hub in Azure is required as part of this lab. Step by step instructions on setting up an IoT Hub can be found here: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal

### Raspberry Pi GPIO pins

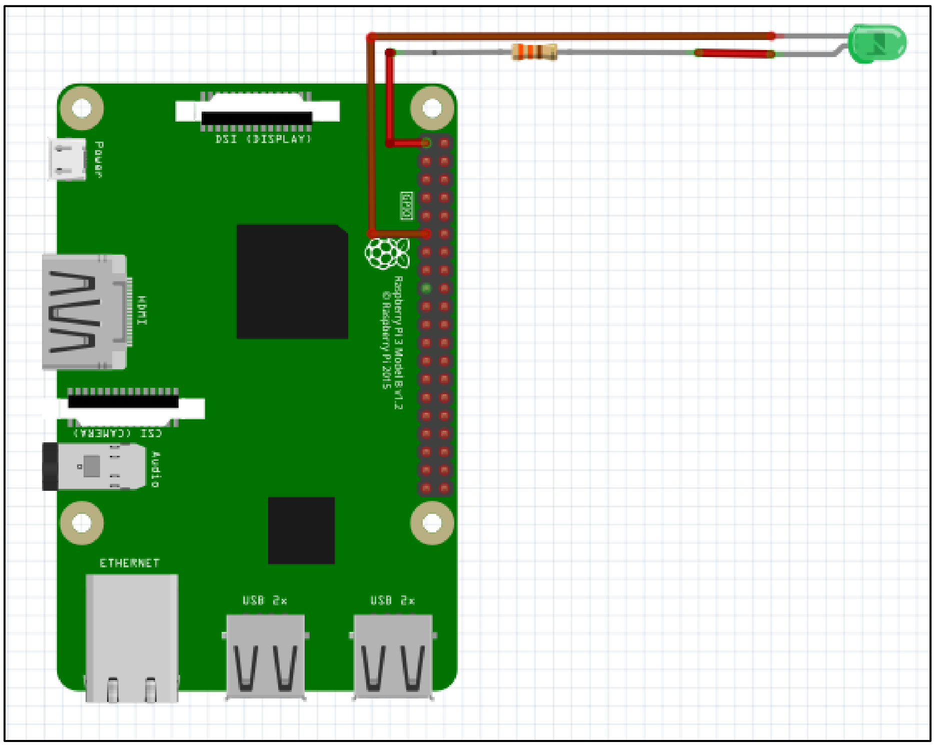
You'll need to understand how the GPIO pins are configured on the Raspberry Pi. Details can be found in the following two links: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi

And here: https://pinout.xyz/pinout/dpi

### Connecting the green LED

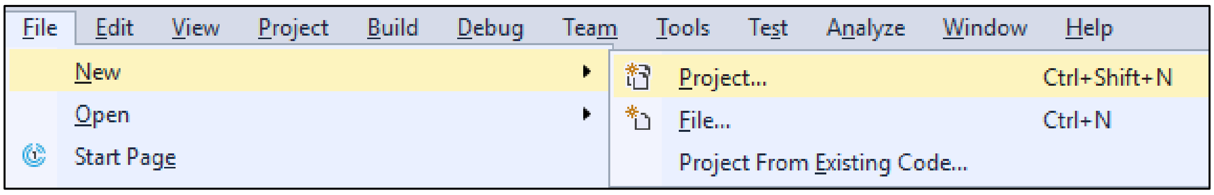
Connect the negative end of the LED to one of the jumper wires, then connect the other end of that jumper wire to GPIO 17 (pin 11). The negative end of the LED is the shorter one.

Connect the positive end to a new jumper wire, then connect the other end of that jumper wire to the resister. Connect a new jumper wire to the other end of the resister, then connect the other end of that jumper wire to a 3v3 power pin (use pin 1).

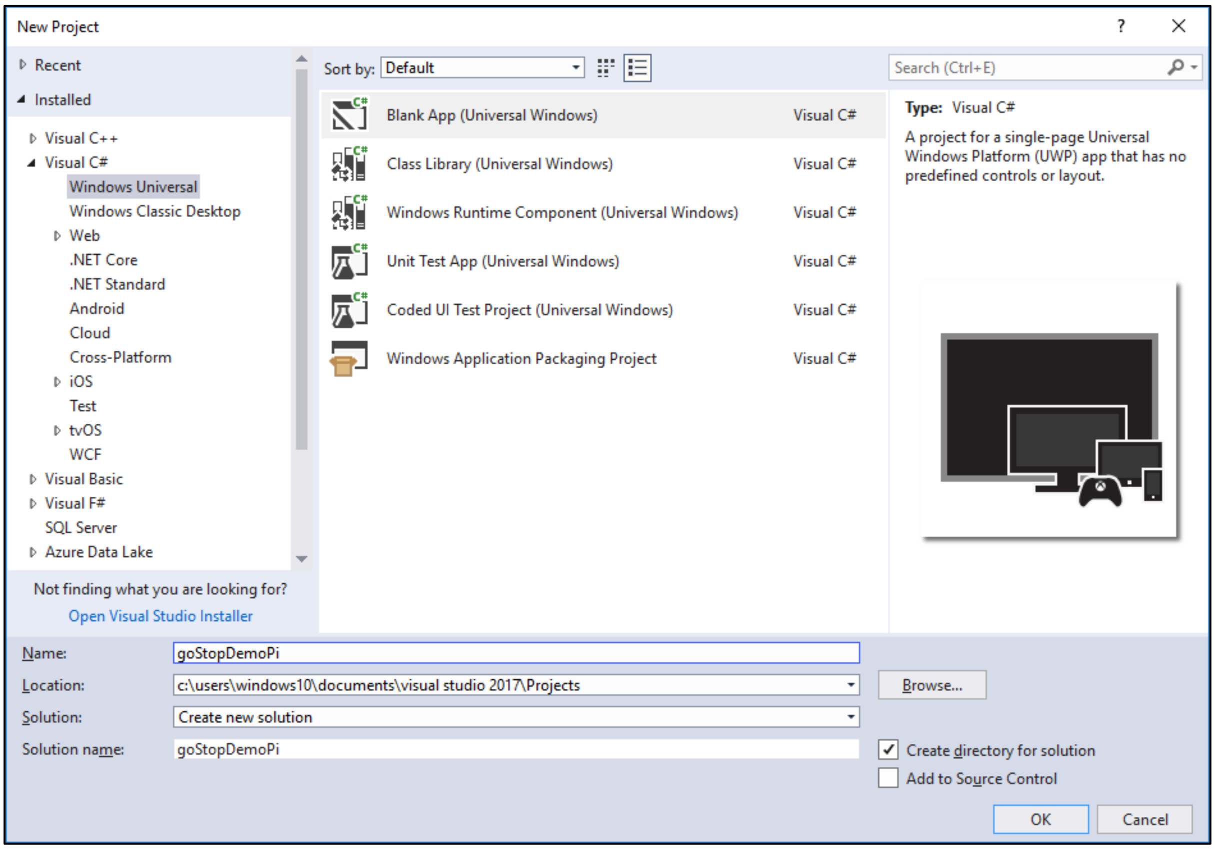


## Part 2 – Turning on an LED using the Raspberry Pi

Create a new project. Open Visual Studio and select **File**, then **New**, then **Project**:



Select **Windows Universal** under Visual C# in the left pane, then select **Blank App (Universal Windows)**. Enter **goStopDemoPi** for the name of the project, then select **OK**:


New Project
Recent
Installed
Visual C++
Visual C#
Windows Universal
Windows Classic Desktop
Web
.NET Core
.NET Standard
Android
Cloud
Cross-Platform
iOS
Test
tvOS
WCF
Visual Basic
Visual F#
SQL Server
Azure Data Lake
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows) Visual C#
Class Library (Universal Windows) Visual C#
Windows Runtime Component (Universal Windows) Visual C#
Unit Test App (Universal Windows) Visual C#
Coded UI Test Project (Universal Windows) Visual C#
Windows Application Packaging Project Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

Name: goStopDemoPi

Location: c:\users\windows10\documents\visual studio 2017\Projects

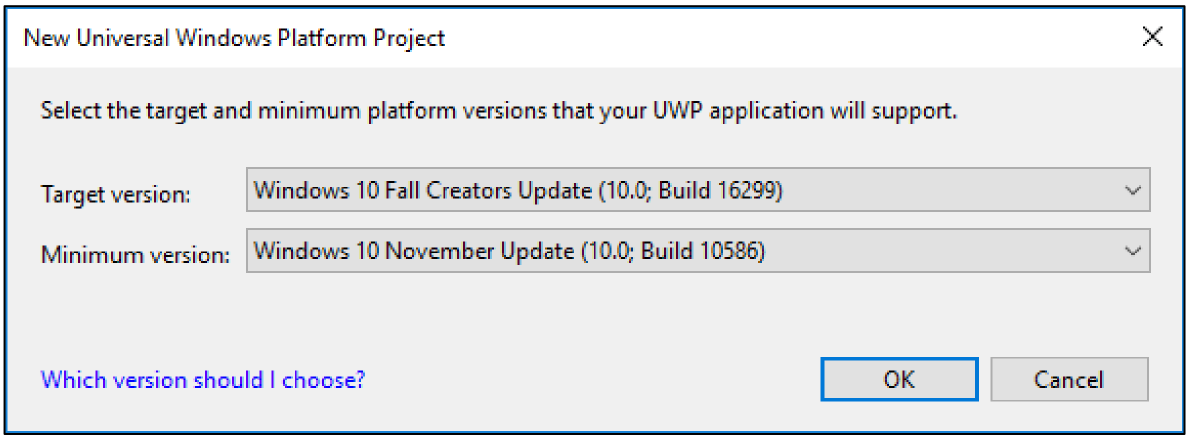
Solution: Create new solution

Solution name: goStopDemoPi

☑ Create directory for solution

☐ Add to Source Control

Leave the default selection for Target and Minimum version selected, then select **OK:**

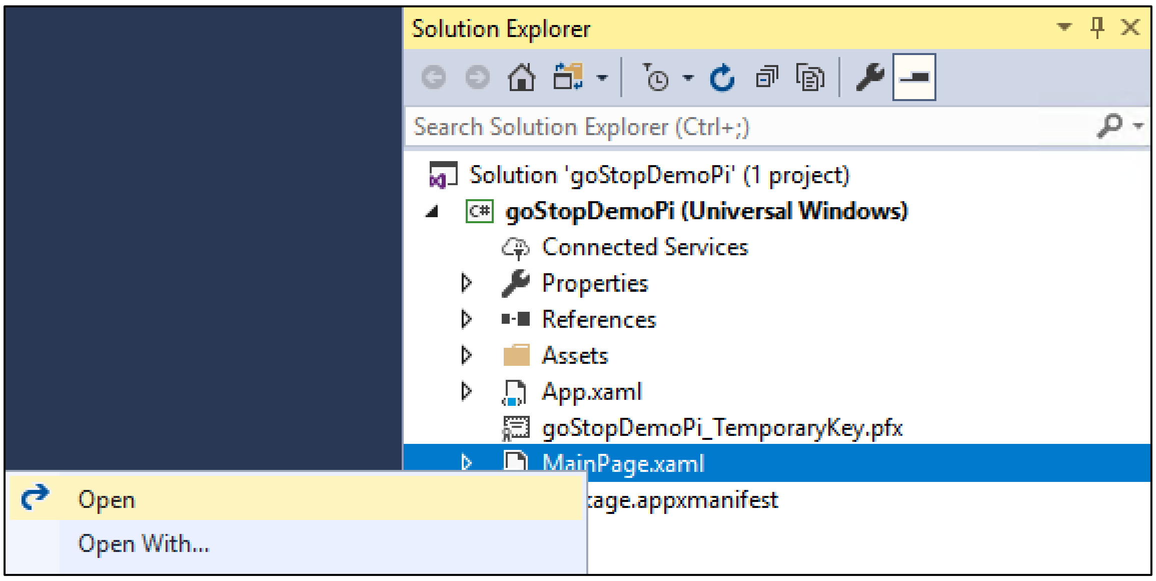

New Universal Windows Platform Project
Which version should I choose?
OK
Cancel

Select the target and minimum platform versions that your UWP application will support.

Target version: Windows 10 Fall Creators Update (10.0; Build 16299)

Minimum version: Windows 10 November Update (10.0; Build 10586)

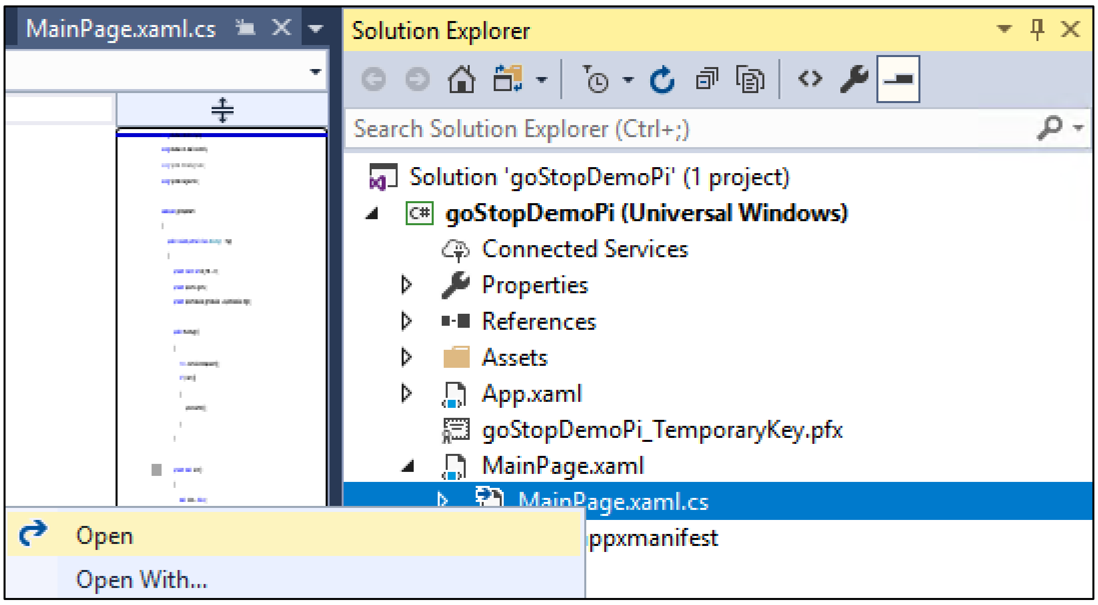
Open **MainPage.xaml**:


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'goStopDemoPi' (1 project)
goStopDemoPi (Universal Windows)
Connected Services
Properties
References
Assets
App.xaml
goStopDemoPi_TemporaryKey.pfx
MainPage.xaml
age.appxmanifest
Open
Open With...

Add the following xaml to the Grid element.
This will show message on screen when if you have a monitor connected to the Raspberry Pi:

```
<StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
       <TextBlock x:Name="message" Text="Initialising GPIO..." Margin="10,50,10,10"
TextAlignment="Center" FontSize="26.667" />
</StackPanel>
```

Open **MainPage.xaml.cs** from the project in Solution Explorer:



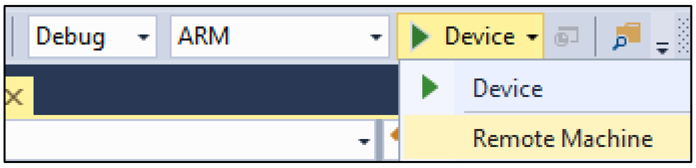
If you selected a name for the project different to the example above, take note of the namespace you have on this page.

Replace the code on the MainPage.xaml.cs with the following code, replacing the 'goStopDemoPi' namespace with the one you're using:

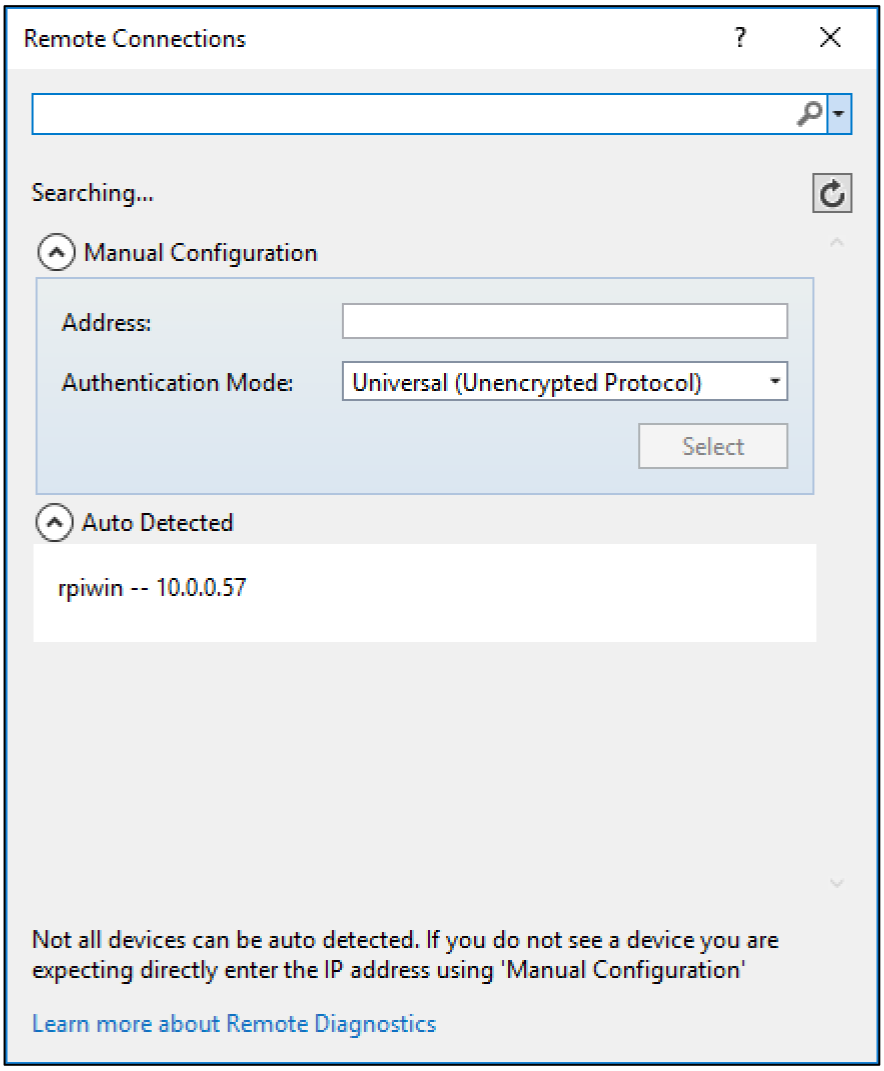
```
using Windows.Devices.Gpio;
using Windows.UI.Xaml.Controls;
using System.Threading.Tasks;
using System.Diagnostics;

namespace goStopDemoPi
{
    public sealed partial class MainPage : Page
    {
        private const int GO_PIN = 17;
        private GpioPin goPin;
        private GpioPinValue goPinValue = GpioPinValue.High;

        public MainPage()
        {
            this.InitializeComponent();
            if (init())
            {
                processGPIO();
            }
        }

        private bool init()
        {
            bool isOk = false;
            var gpio = GpioController.GetDefault();

            if (gpio == null)
            {
                message.Text = "There is no GPIO controller on this device.";
                Debug.WriteLine(message.Text);
            }
            else
            {
                goPin = gpio.OpenPin(GO_PIN);
                goPin.Write(GpioPinValue.High);
                goPin.SetDriveMode(GpioPinDriveMode.Output);
                message.Text = "GPIO controller initialised.";
                Debug.WriteLine(message.Text);
                isOk = true;
            }
            return isOk;
        }


        private void processGPIO()
        {
            goPinValue = GpioPinValue.Low;
            goPin.Write(goPinValue);
            message.Text = $"Go Pin set to '{goPinValue}'.";
            Debug.WriteLine(message.Text);
        }

    }
}
```

This code initialises the GPIO Controller, then if the GPIO Controller has been initialised, sets the GPIO for the green LED to ‘Low’, turning on the green LED.

At this point, you should be able to test what we have so far and check that the LED turns on before moving forward. To debug, select **Debug** from the toolbar, then select **ARM** as the device type, then select **Remote Machine** under the **Device** dropdown:



The first time you will be prompted for details on the device. You should have your device listed under the **Auto Detected** section. If not, type in the hostname or IP address of your RaspberryPi:



Searching...

Manual Configuration

Address:

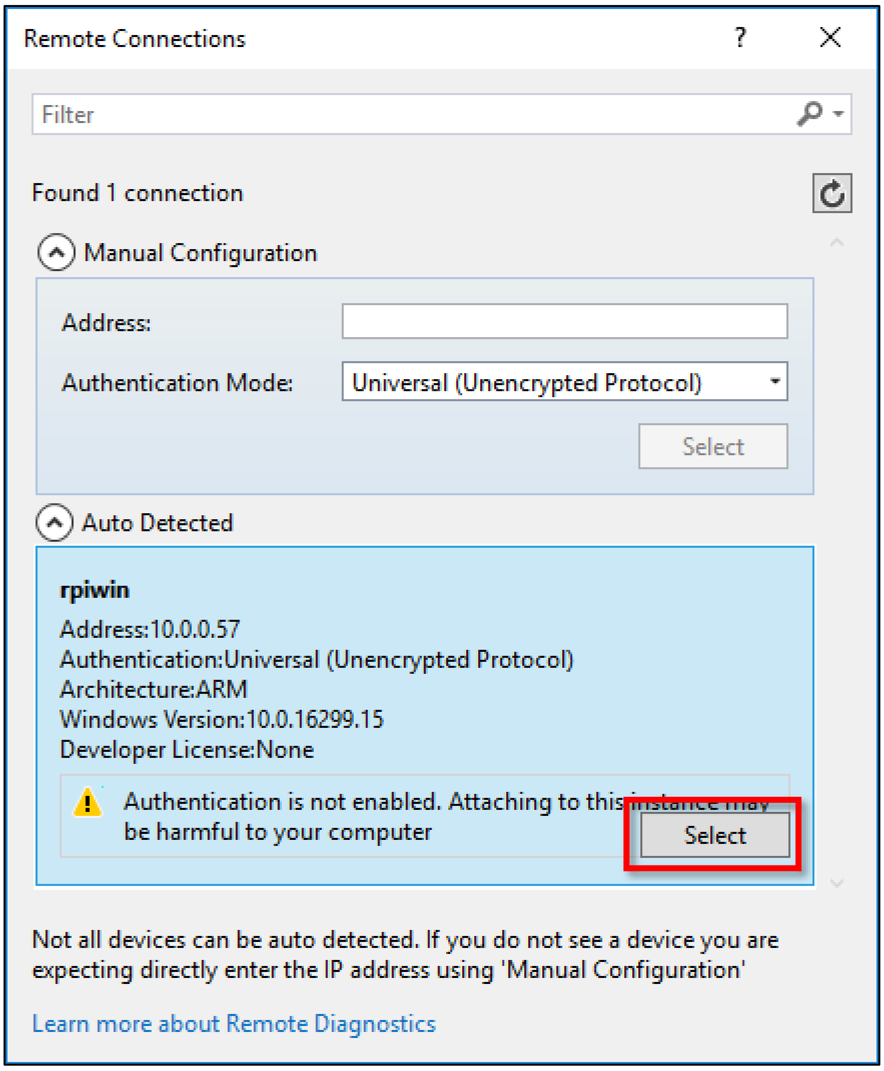
Authentication Mode: Universal (Unencrypted Protocol)

Auto Detected

rpiwin -- 10.0.0.57

Not all devices can be auto detected. If you do not see a device you are expecting directly enter the IP address using 'Manual Configuration'

Select the auto detected device, then select **Select**:



Found 1 connection

Manual Configuration

Address:

Authentication Mode: Universal (Unencrypted Protocol)

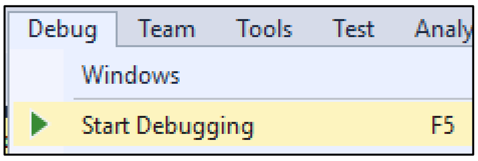
Auto Detected

**rpiwin**
Address:10.0.0.57
Authentication:Universal (Unencrypted Protocol)
Architecture:ARM
Windows Version:10.0.16299.15
Developer License:None

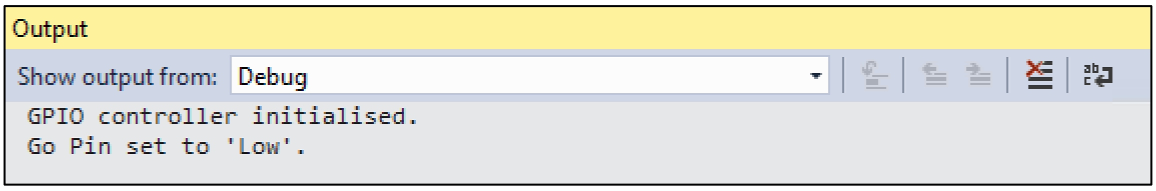
Authentication is not enabled. Attaching to this instance may be harmful to your computer

Not all devices can be auto detected. If you do not see a device you are expecting directly enter the IP address using 'Manual Configuration'

Select **Debug**, then **Start Debugging** from the toolbar, or press **F5** to build, deploy and start debugging the app on your RaspberryPi:



After a few moments your app should be built and deployed to the RaspberryPi and your code should have executed within the debugger. The output window should display a message that the GPIO controller was initialised successfully and the Go Pin was set to **Low** (‘on’):



Show output from: Debug

```
GPIO controller initialised.
Go Pin set to 'Low'.
```

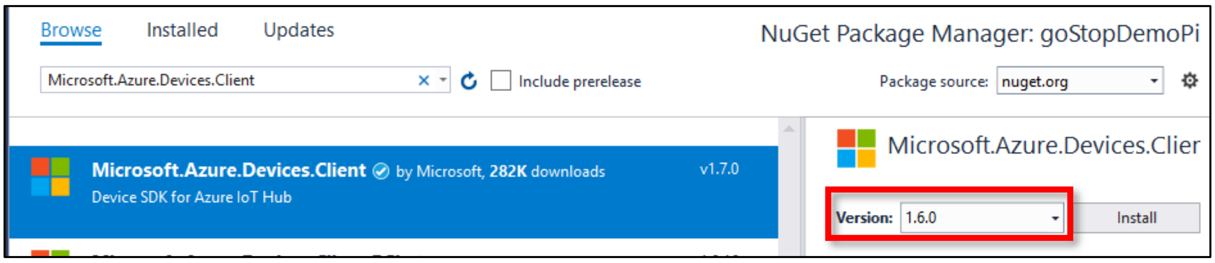
The green LED we connected earlier on the Raspberry Pi should now be turned on.

Stop the debugger. The green LED should now be off.

## Part 3 – Enable the Raspberry Pi to act on request

In the NuGet Package Manager window, select **Browse** and search for **Microsft.Azure.Device.Client**.

Select **Microsft.Azure.Device.Client** from the list. In the right window pane, ensure to select **Version 1.6.0**, then select **Install**:



Microsoft.Azure.Devices.Client

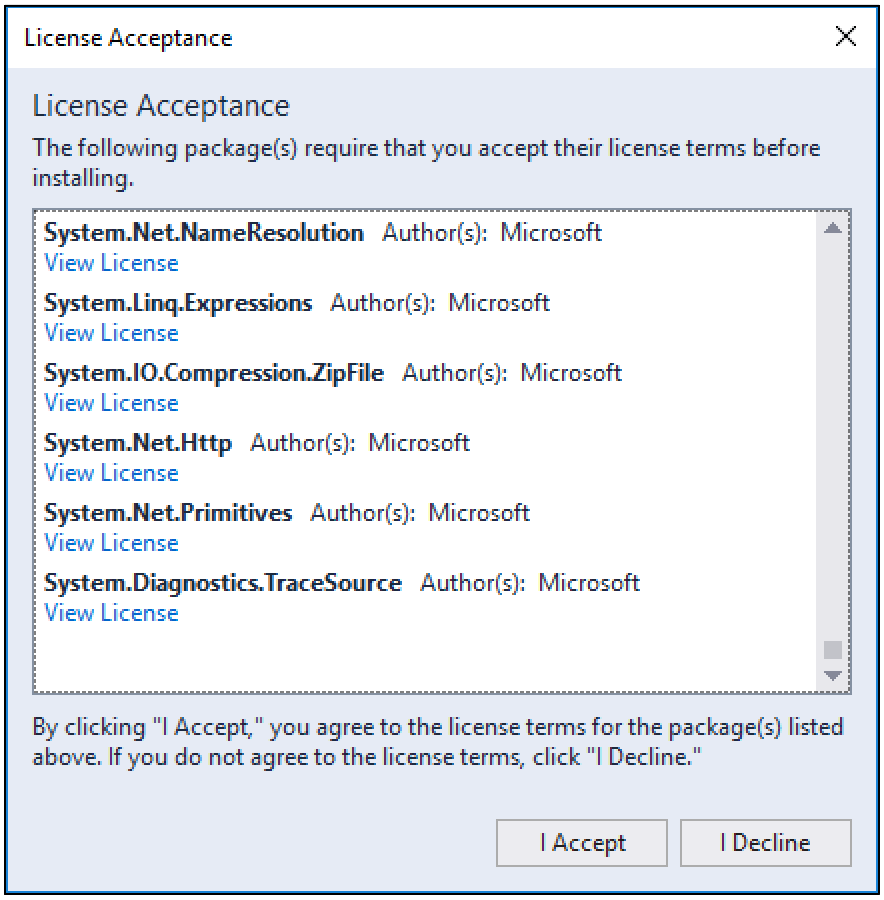
Include prerelease

Package source: nuget.org

Microsoft.Azure.Devices.Client by Microsoft, 282K downloads v1.7.0
Device SDK for Azure IoT Hub

Version: 1.6.0

If prompted, select **I Accept** from the License Acceptance window:



License Acceptance

The following package(s) require that you accept their license terms before installing.

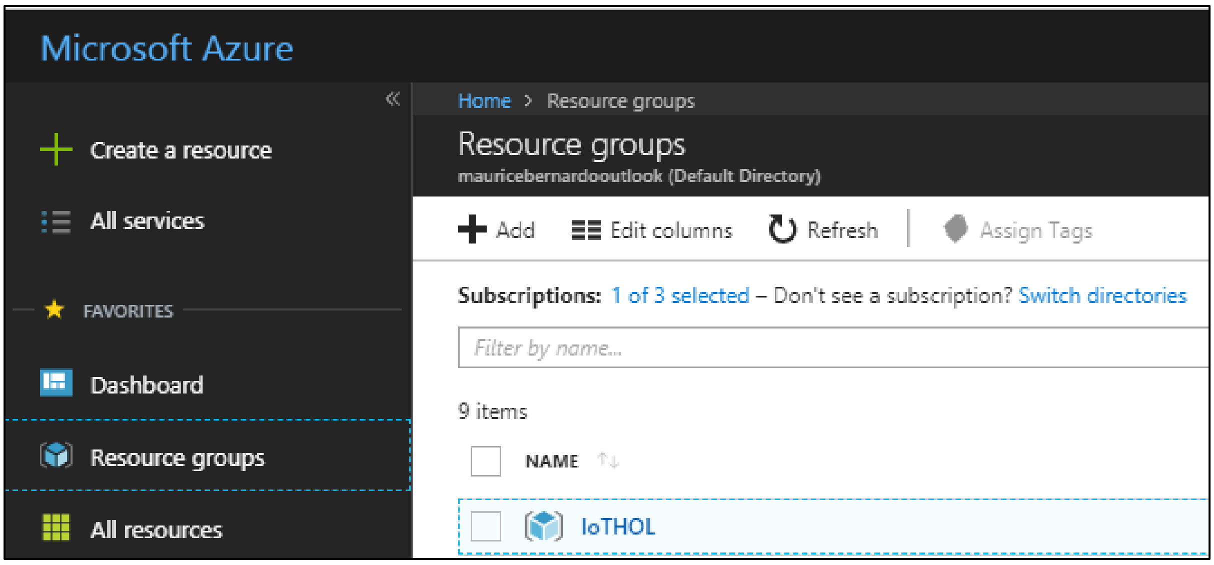
System.Net.NameResolution Author(s): Microsoft
View License
System.Linq.Expressions Author(s): Microsoft
View License
System.IO.Compression.ZipFile Author(s): Microsoft
View License
System.Net.Http Author(s): Microsoft
View License
System.Net.Primitives Author(s): Microsoft
View License
System.Diagnostics.TraceSource Author(s): Microsoft
View License

By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."

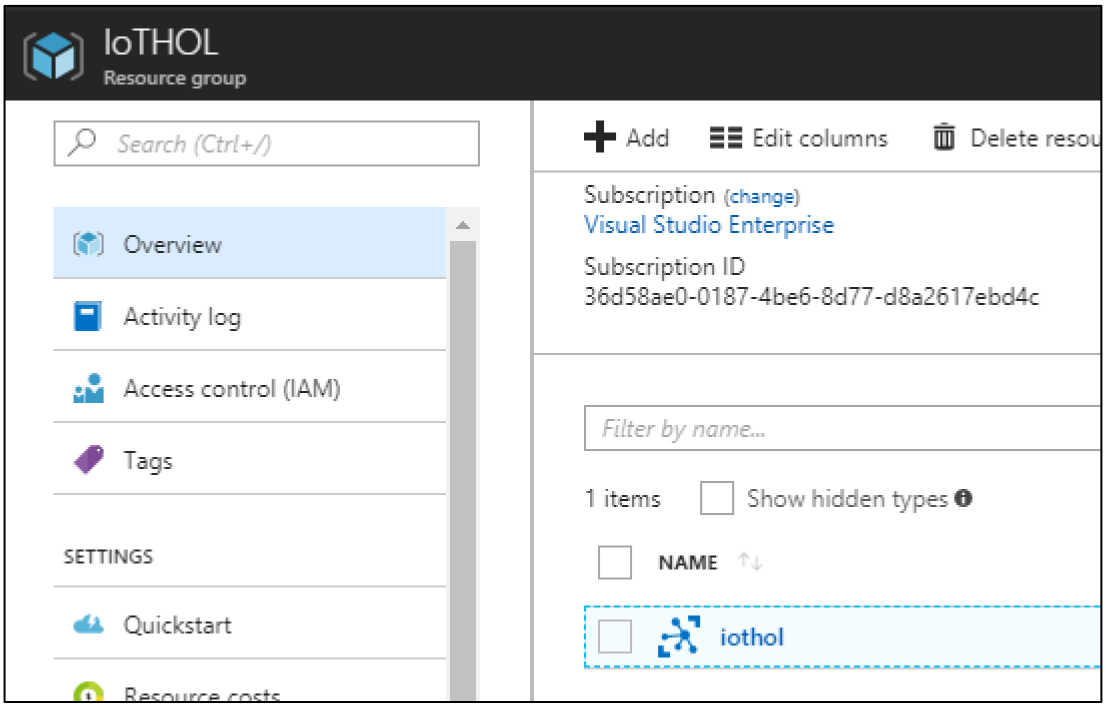
Replace the entire contents of MainPage.xaml.cs with the following code:

```
using Windows.Devices.Gpio;
using Windows.UI.Xaml.Controls;
using System.Threading.Tasks;
using System.Diagnostics;
using Microsoft.Azure.Devices.Client;
using System;
using System.Text;
using Windows.UI.Xaml.Navigation;
using Windows.UI.Core;

namespace goStopDemoPi
{
    public sealed partial class MainPage : Page
    {
        static string DeviceConnectionString = "connection_string_here";
        static DeviceClient Client = null;

        private const int GO_PIN = 17;
        private GpioPin goPin;
        private GpioPinValue goPinValue = GpioPinValue.High;

        public MainPage()
        {
            this.InitializeComponent();
            if (init())
            {
                OpenIoTHubConnection();
            }
        }

        private bool init()
        {
            bool isOk = false;
            var gpio = GpioController.GetDefault();

            if (gpio == null)
            {
                message.Text = "There is no GPIO controller on this device.";
                Debug.WriteLine(message.Text);
            }
            else
            {
                goPin = gpio.OpenPin(GO_PIN);
                goPin.Write(GpioPinValue.High);
                goPin.SetDriveMode(GpioPinDriveMode.Output);
                message.Text = "GPIO controller initialised.";
                Debug.WriteLine(message.Text);
                isOk = true;
            }
            return isOk;
        }

        private void OpenIoTHubConnection()
        {
            try
            {
                message.Text = "Connecting to IoT Hub ...";
                Debug.WriteLine(message.Text);
                Client =
DeviceClient.CreateFromConnectionString(DeviceConnectionString, TransportType.Mqtt);
                message.Text = "IoT Hub connection successful.";
                Debug.WriteLine(message.Text);
                string msg = "Waiting for direct method call.";
                message.Text += $"\n\n{msg}";
                Debug.WriteLine(msg);
                // setup callback method:
                Client.SetMethodHandlerAsync("RaspberryPiGo", RaspberryPiGo, null);
            }
            catch (Exception ex)
            {
                message.Text = $"\n\nError! : {ex.Message}";
                Debug.WriteLine(message.Text);
            }
        }

        async Task<MethodResponse> RaspberryPiGo(MethodRequest methodRequest, object
userContext)
        {
            string result = string.Empty;
            try
            {

                await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
                {
                    message.Text += $"\n\nReceived a request for RaspberryPiGo.
{methodRequest.DataAsJson}";
                    Debug.WriteLine(message.Text);
                    goPinValue = GpioPinValue.Low;
                    goPin.Write(goPinValue);
                    message.Text += $"\nGo Pin set to '{goPinValue}' (turned on).";
                    Debug.WriteLine(message.Text);
                    result = "'Green LED turned on.'";
                });
                return await Task.FromResult(new
MethodResponse(Encoding.UTF8.GetBytes(result), 200));
            }
            catch (Exception ex)
            {
                message.Text = $"\n\nError! : {ex.Message}";
                Debug.WriteLine(message.Text);
                result = "'An error ocurred.'";
                return await Task.FromResult(new
MethodResponse(Encoding.UTF8.GetBytes(result), 200));
            }
        }

        protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
        {
            Client?.SetMethodHandlerAsync("RaspberryPiGo", null, null).Wait();
            Client?.CloseAsync().Wait();
        }
    }
}
```

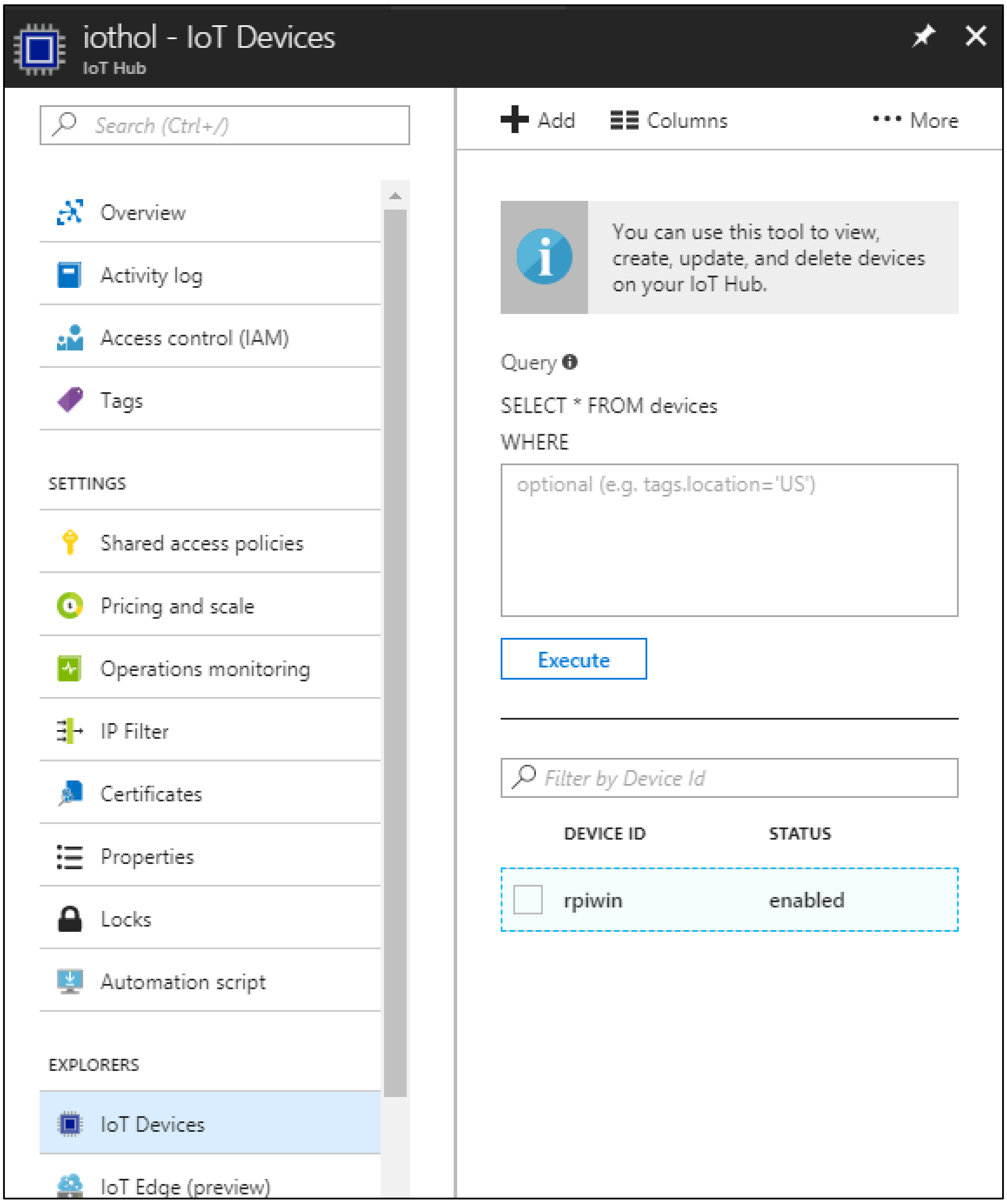
In the code just copied in, update the connection string variable with the connection string from IoT Hub in Azure. To obtain the connection string, navigate to the Azure Portal, select **Resources** from the left menu, then select your resource group (Eg, **IoTHOL**):



On the overview blade, select the IoT Hub you configured previously (Eg **iothol**):



On the IoT Hub blade, under **explorers**, select **IoT Devices,** then select your RaspberryPi Device (Eg **rpiwin**) from the devices list:



iothol - IoT Devices
IoT Hub

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags

SETTINGS
Shared access policies
Pricing and scale
Operations monitoring
IP Filter
Certificates
Properties
Locks
Automation script

EXPLORERS
IoT Devices
IoT Edge (preview)

Add | Columns | More

You can use this tool to view, create, update, and delete devices on your IoT Hub.

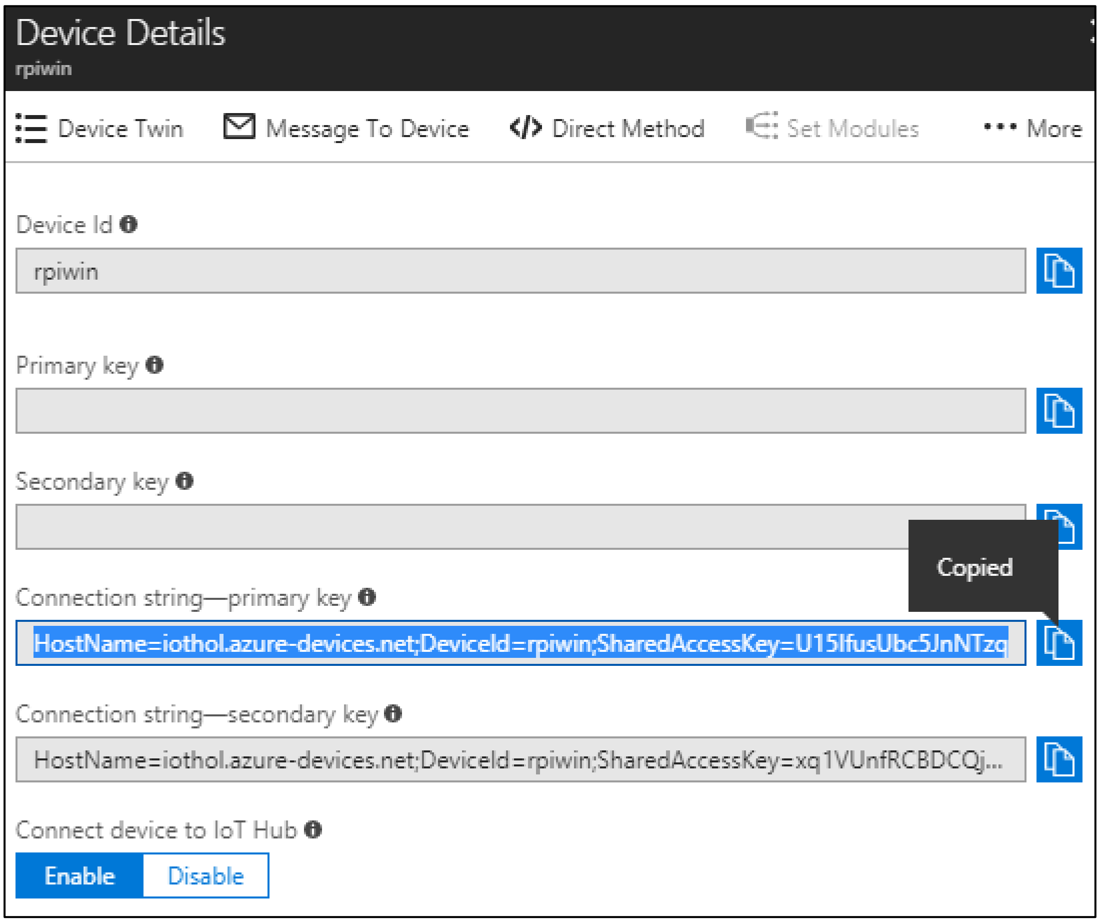
Query
SELECT * FROM devices
WHERE
optional (e.g. tags.location='US')

Execute

Filter by Device Id

| DEVICE ID | STATUS |
|---|---|
| rpiwin | enabled |

The final step is to copy the value in the **Connection string – primary key** field. This is the connection string to add to the **DeviceConnectionString** variable in **MainPage.xaml.cs**:



Device Id: rpiwin

Primary key:

Secondary key:

Connection string—primary key: HostName=iothol.azure-devices.net;DeviceId=rpiwin;SharedAccessKey=U15IfusUbc5JnNTzq

Connection string—secondary key: HostName=iothol.azure-devices.net;DeviceId=rpiwin;SharedAccessKey=xq1VUnfRCBDCQj...

Connect device to IoT Hub

## IoT Hub Connection String Example:

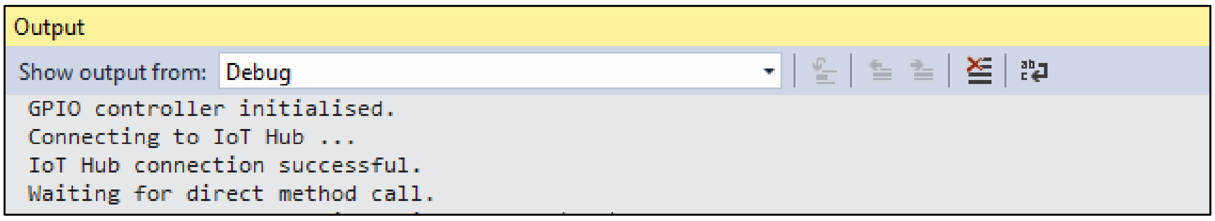
```
static string DeviceConnectionString = "HostName=iothol.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=abcd/efg3t/CFGDH/WSA4xgDE+
LSdhQ2lLxP+S/P9ss=";
```

This updated code connects to the IoT Hub and configures a method that can be used as a Direct Method call from the cloud.

You should now be able to test that your RaspberryPi connects to the IoT Hub and waits for a call to come through.

Select the **Debug** button from the toolbar or press **F5**. After a few moments the app should be built and deployed to the RaspberryPi.

You should see a few debug messages showing the results of the changes in the output window. If you have connected a screeen to your RaspberryPi you will see similar messages displayed on that screen through the IoT Core UWP app (goStopDemoPi):

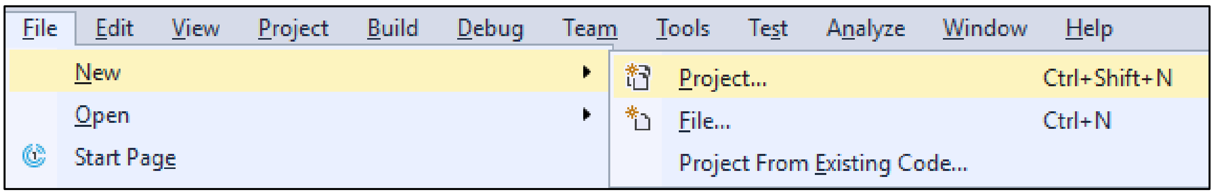


```
GPIO controller initialised.
Connecting to IoT Hub ...
IoT Hub connection successful.
Waiting for direct method call.
```

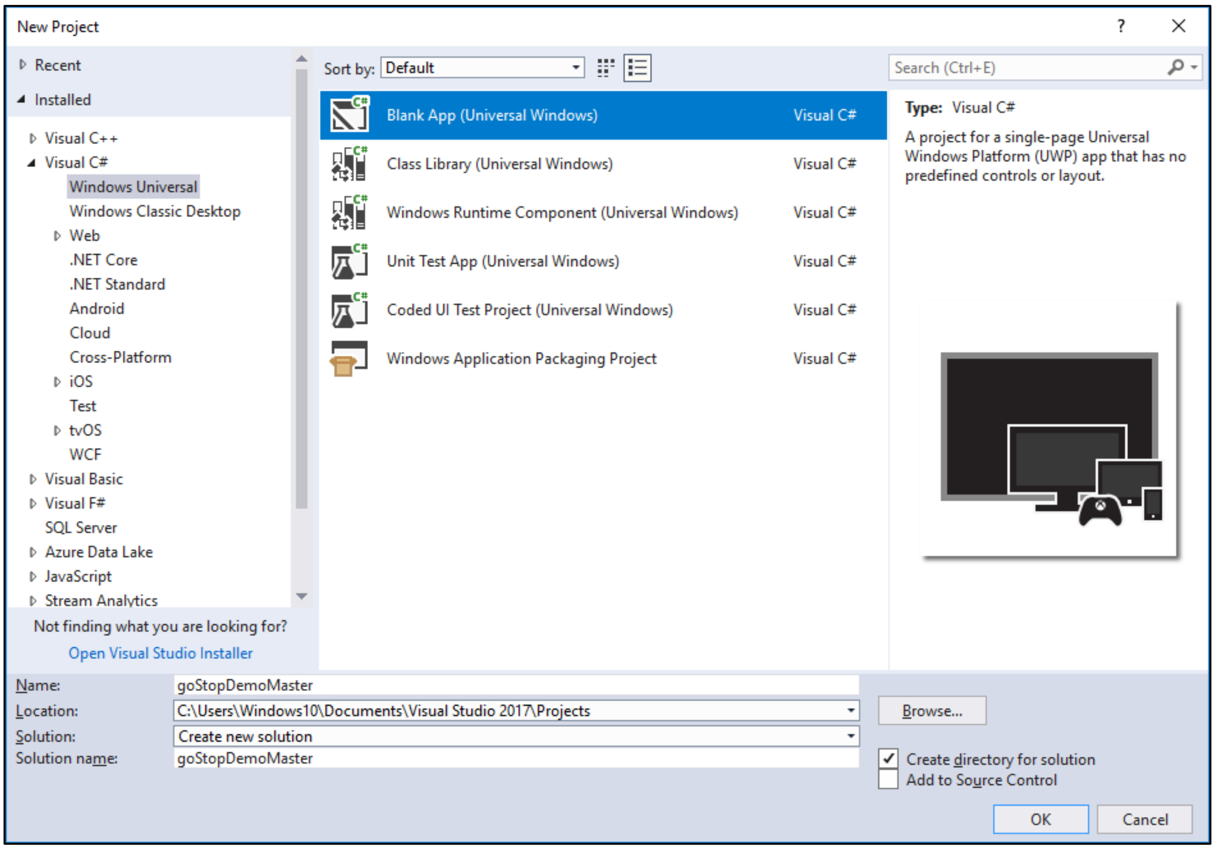
## Part 4 – Remote controlled LED

Create a new project. This will be used to control the LED.

Open Visual Studio. Select **File**, then **New**, then **Project**:



Select **Windows Universal** under Visual C# in the left pane, then select **Blank App (Universal Windows)**. Enter **goStopDemoMaster** for the name of the project, then select **OK**:


New Project
Recent
Installed
Visual C++
Visual C#
Windows Universal
Windows Classic Desktop
Web
.NET Core
.NET Standard
Android
Cloud
Cross-Platform
iOS
Test
tvOS
WCF
Visual Basic
Visual F#
SQL Server
Azure Data Lake
JavaScript
Stream Analytics
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows) Visual C#
Class Library (Universal Windows) Visual C#
Windows Runtime Component (Universal Windows) Visual C#
Unit Test App (Universal Windows) Visual C#
Coded UI Test Project (Universal Windows) Visual C#
Windows Application Packaging Project Visual C#
Browse...
OK
Cancel

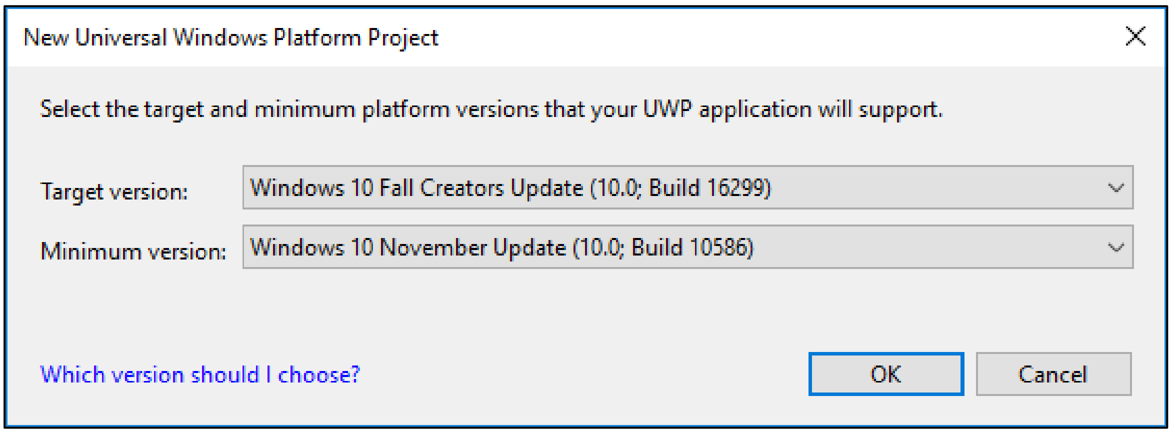
Type: Visual C#

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

| | |
|---|---|
| Name: | goStopDemoMaster |
| Location: | C:\Users\Windows10\Documents\Visual Studio 2017\Projects |
| Solution: | Create new solution |
| Solution name: | goStopDemoMaster |

✓ Create directory for solution
Add to Source Control

Leave the default selection for Target and Minimum version selected, then select **OK:**


New Universal Windows Platform Project
Which version should I choose?
OK
Cancel

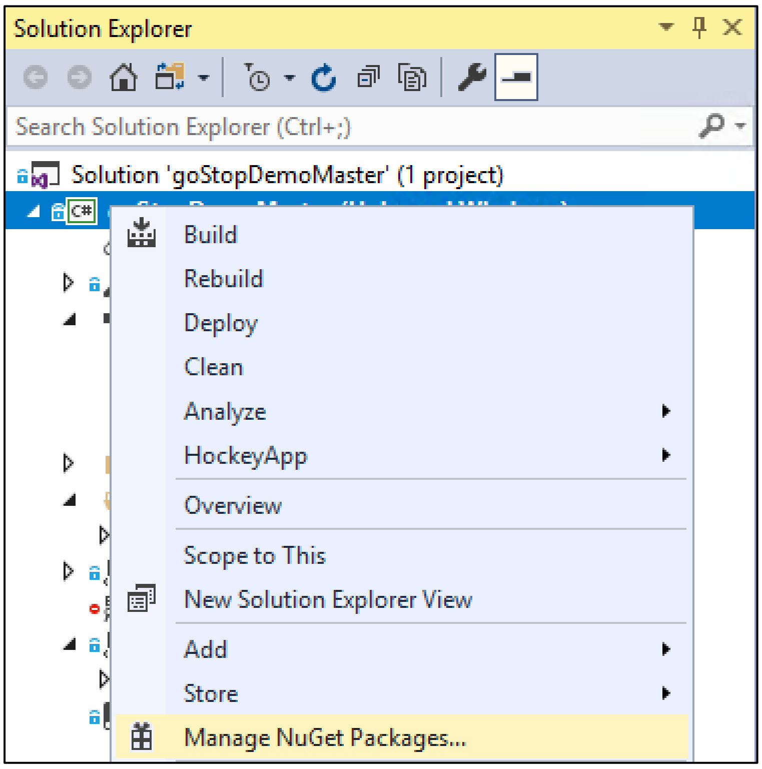
Select the target and minimum platform versions that your UWP application will support.

| | |
|---|---|
| Target version: | Windows 10 Fall Creators Update (10.0; Build 16299) |
| Minimum version: | Windows 10 November Update (10.0; Build 10586) |

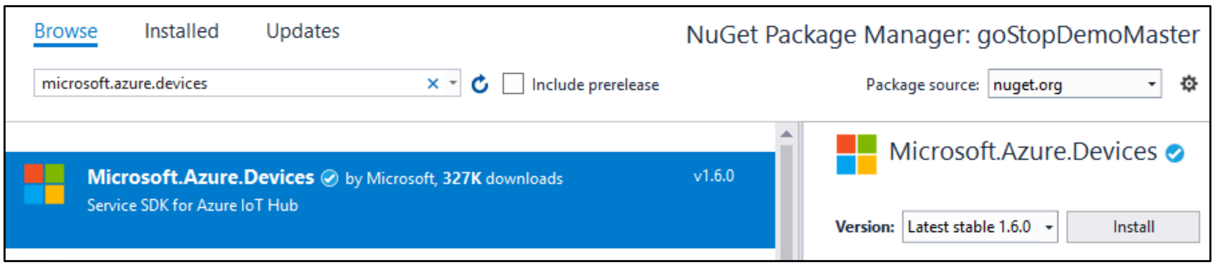
## Install Azure IoT service SDK

This procedure downloads, installs, and adds a reference to the Azure IoT service SDK NuGet package and its dependencies.

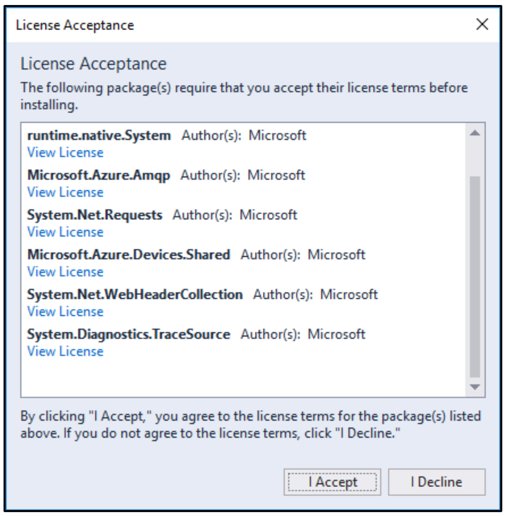
Right-click the project, then select **Manage NuGut Packages…** :



In NuGet Package Manager, search for the **microsoft.azure.devices** package and select it:


Browse
Installed
Updates
NuGet Package Manager: goStopDemoMaster
microsoft.azure.devices
Include prerelease
Package source: nuget.org
Microsoft.Azure.Devices by Microsoft, 327K downloads
Service SDK for Azure IoT Hub
v1.6.0
Microsoft.Azure.Devices
Version: Latest stable 1.6.0
Install

If you're prompted with a License Acceptance, select **I Accept** to continue:


License Acceptance

License Acceptance

The following package(s) require that you accept their license terms before installing.

**runtime.native.System** Author(s): Microsoft
View License

**Microsoft.Azure.Amqp** Author(s): Microsoft
View License

**System.Net.Requests** Author(s): Microsoft
View License

**Microsoft.Azure.Devices.Shared** Author(s): Microsoft
View License

**System.Net.WebHeaderCollection** Author(s): Microsoft
View License

**System.Diagnostics.TraceSource** Author(s): Microsoft
View License

By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."

I Accept | I Decline

After a few moments the packages will be installed.

Close the NuGet Package Manager window.

## Update MainPage.xaml

Open MainPage.xaml from Solution Explorer.

Replace the entire contents with the following code:

```
<Page
    x:Class="goStopDemoMaster.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:goStopDemoMaster"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
Margin="0,0,396,0">
        <Button x:Name="Go" Content="Go!" HorizontalAlignment="Left"
Margin="28,35,0,0" VerticalAlignment="Top" Click="btnGo_Click" Foreground="White"
Background="#FF29AC14" Width="160" Height="102" FontWeight="Bold" FontSize="36"/>
        <TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="10,186,0,0"
TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="806" Width="1082"/>
        <TextBox HorizontalAlignment="Left" Margin="0,148,0,0" Text="Messages:"
VerticalAlignment="Top" BorderBrush="{x:Null}" Width="98"/>
    </Grid>
</Page>
```

## Update MainPage.xaml.cs

Open MainPage.xaml.cs from Solution Explorer.

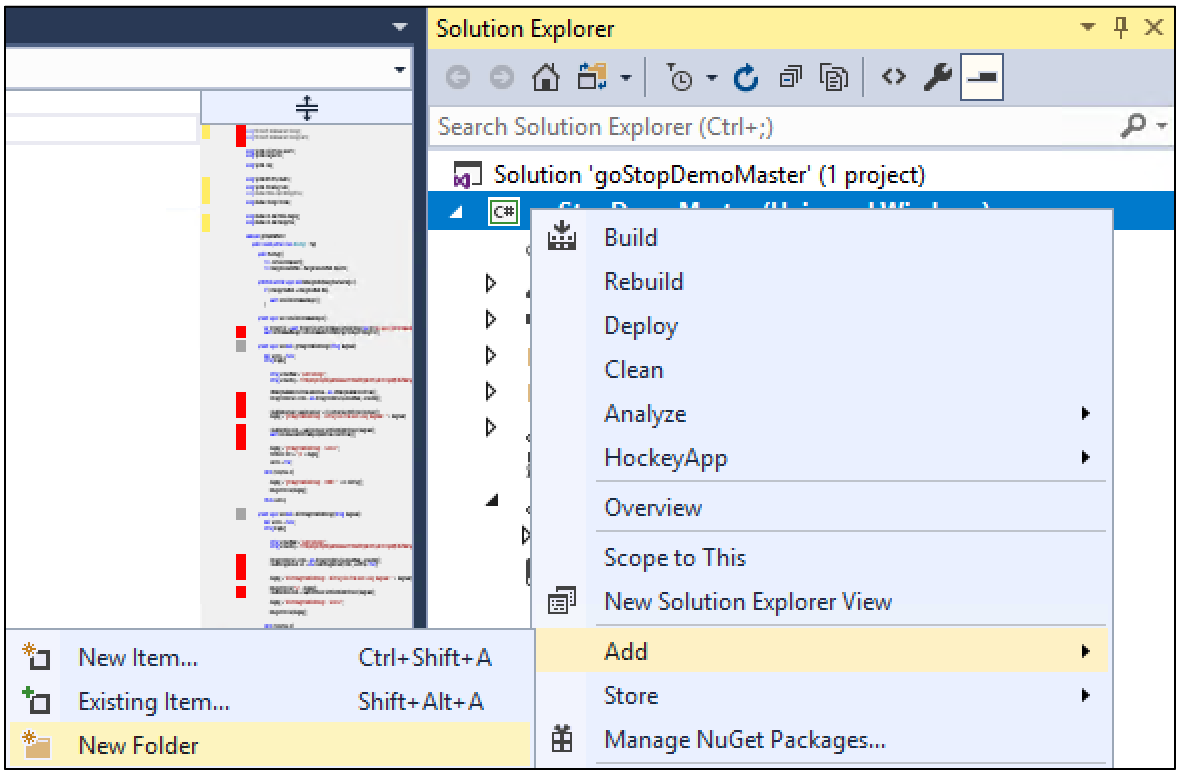
Replace the entire contents with the following code:

```
using goStopDemoMaster.Classes;
using System;
using System.Threading.Tasks;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace goStopDemoMaster
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            this.NavigationCacheMode = NavigationCacheMode.Required;
        }


        private async void btnGo_Click(object sender, RoutedEventArgs e)
        {
            var msgUpdate = new Progress<string>(msg =>
            {
                textBlock.Text += (textBlock.Text == string.Empty) ? msg : "\n\n" +
msg;
```

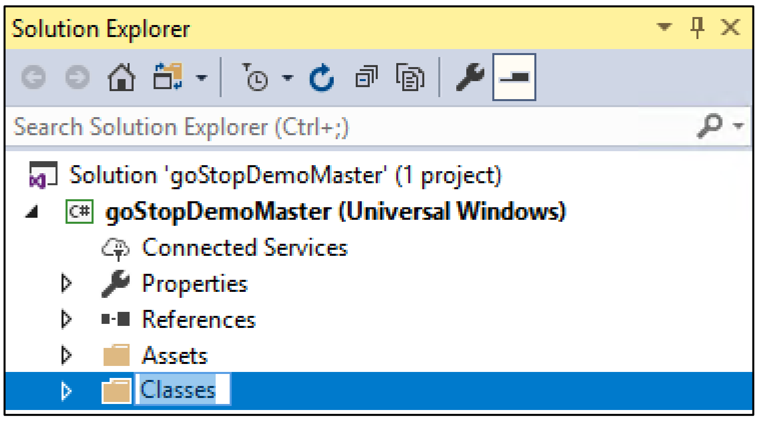
```
                });
                await Task.Run(() => IoTHub.InvokeMethod("RaspberryPiGo", msgUpdate));
            }
        }
    }
```

## Add a new Classes folder

Add a new folder to the project called 'Classes'; Right-click on the project in solution explorer, select **Add** then select **New Folder**:



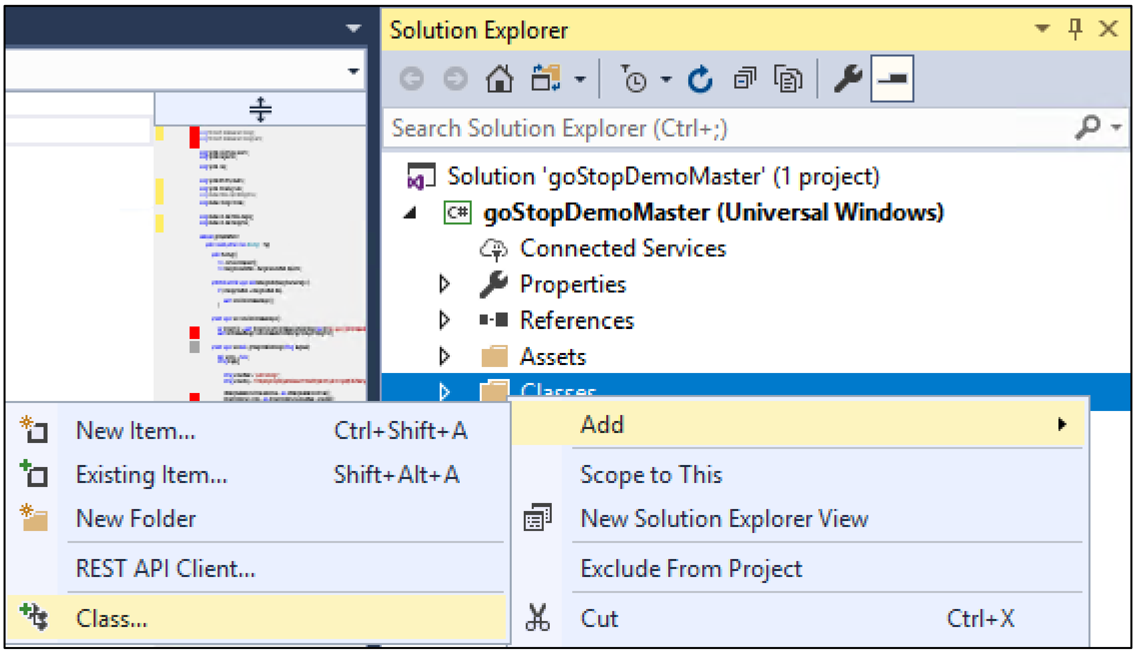
Name the new folder **Classes:**



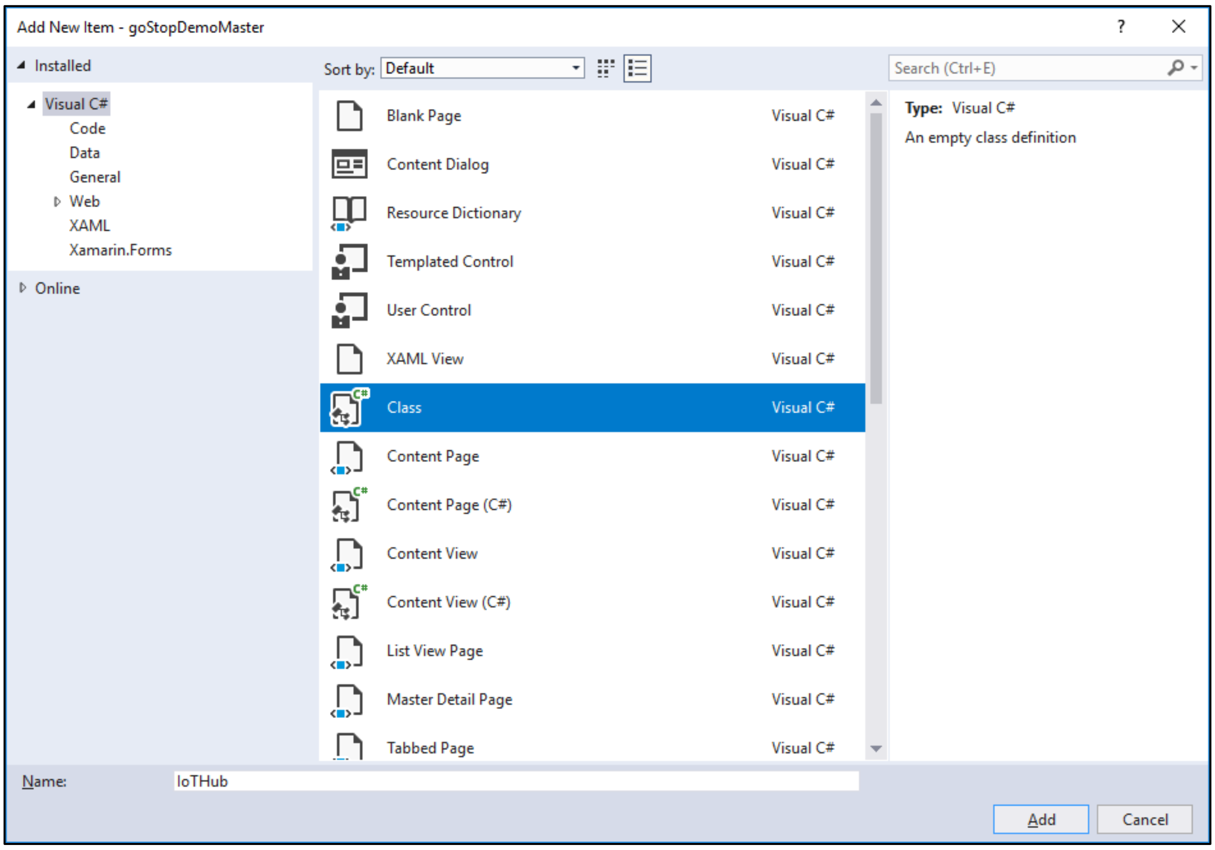
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'goStopDemoMaster' (1 project)
goStopDemoMaster (Universal Windows)
Connected Services
Properties
References
Assets
Classes

## Add a class for IoT Hub methods

Add a new Class to the **Classes** folder called **IoTHub**; Right-click the new **Classes** folder, select **Add**, then select **Class…** :



Select **Visual C#** from the left window pane, select **Class** from the main window pane, enter **IoTHub** for the name, then select **Add**:



Replace the contents of the new **IoTHub** class with the following code.

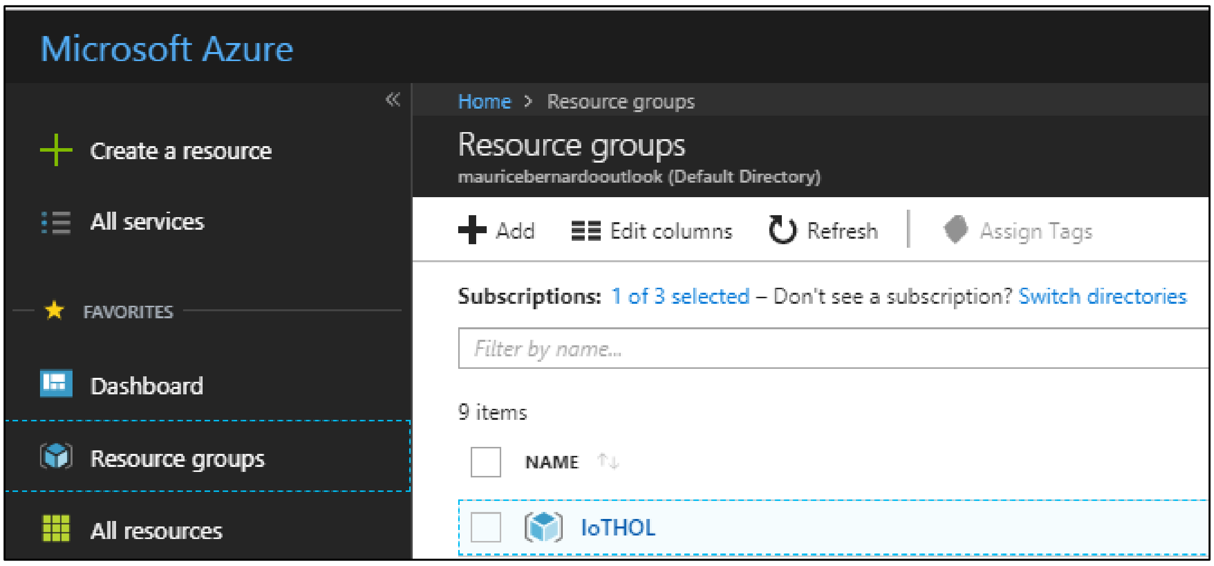
This class will provide methods that will call the methods exposed by the app on the Raspberry Pi.

We will set the correct value for the correctionString and the deviceId in the next steps:

```
using System;
using System.Collections;
using System.Diagnostics;
using System.Threading.Tasks;
using Microsoft.Azure.Devices;

namespace goStopDemoMaster.Classes
{
    class IoTHub
    {
        static ServiceClient serviceClient;
        static string connectionString = "enter_connection_string";
        static string deviceId = "enter_deviceId";
        static string diagmsg = string.Empty;

        public static async Task InvokeMethod(string Method, IProgress<string>
msgUpdate = null)
        {
            string diagmsg = string.Empty;

            try
            {
                serviceClient =
ServiceClient.CreateFromConnectionString(connectionString);

                var methodInvocation = new CloudToDeviceMethod(Method) {
ResponseTimeout = TimeSpan.FromSeconds(30) };
                //methodInvocation.SetPayloadJson("'Add payload data here if
needed'");

                var response = await serviceClient.InvokeDeviceMethodAsync(deviceId,
methodInvocation);

                diagmsg = $"Response status: {response.Status}, payload:";
                Debug.WriteLine(diagmsg);
                msgUpdate?.Report(diagmsg);
                diagmsg = response.GetPayloadAsJson();
                Debug.WriteLine(diagmsg);
                msgUpdate?.Report(diagmsg);
            }
            catch (Exception ex)
            {
                diagmsg = "InvokeMethod - ERROR! " + ex.ToString();
                msgUpdate?.Report(diagmsg);
                Debug.WriteLine(diagmsg);
            }
        }
    }
}
```

## Add the connection string and deviceId to the master project

Update the connection string variable with the connection string from IoT Hub in Azure. To obtain the connection string, navigate to the Azure Portal, select **Resources** from the left menu, then select your resource group (Eg, **IoTHOL**):

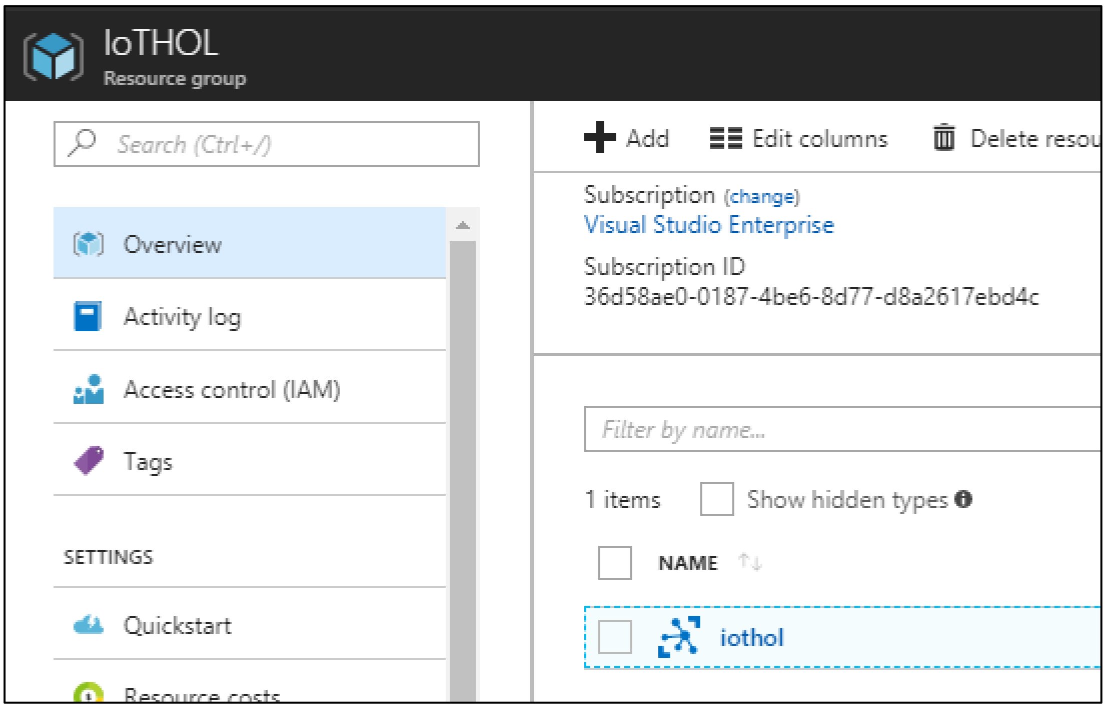


mauricebernardooutlook (Default Directory)

Subscriptions: 1 of 3 selected – Don't see a subscription? Switch directories

9 items

On the overview blade, select the IoT Hub you configured previously (Eg **iothol**):

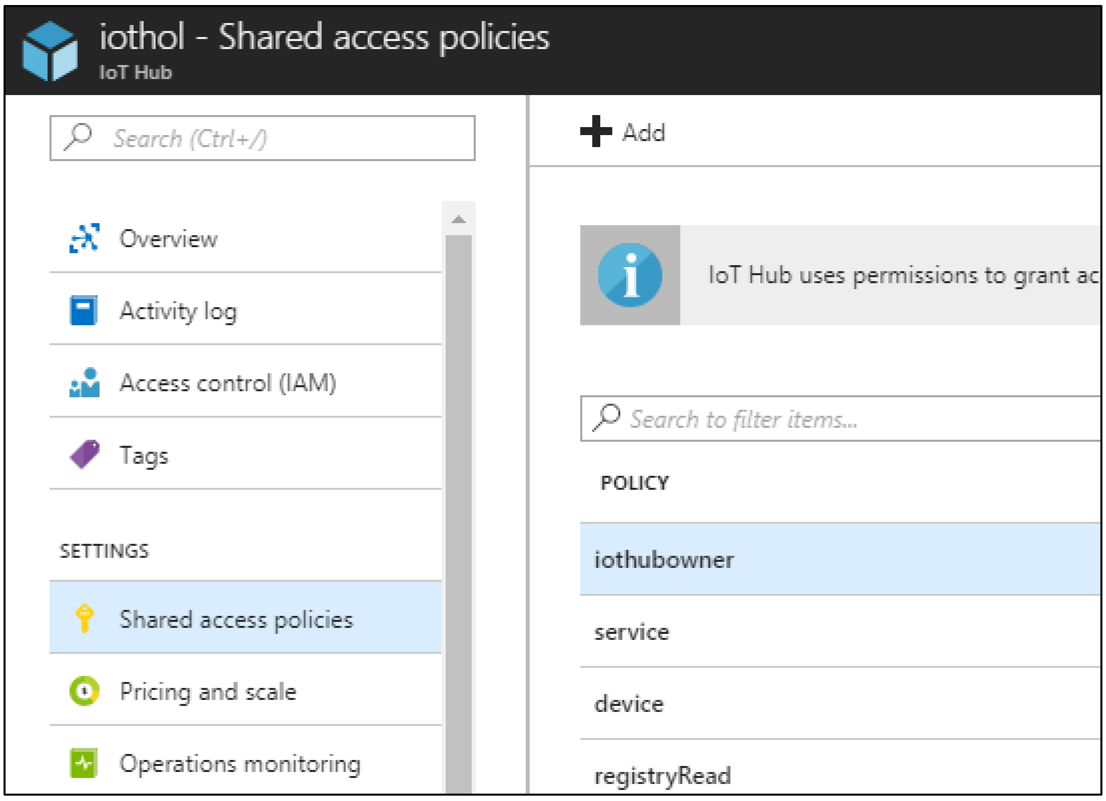


Subscription (change)
Visual Studio Enterprise

Subscription ID
36d58ae0-0187-4be6-8d77-d8a2617ebd4c

1 items

On the IoT Hub Blade, select **Shared access policies**, then select **iothubowner**:

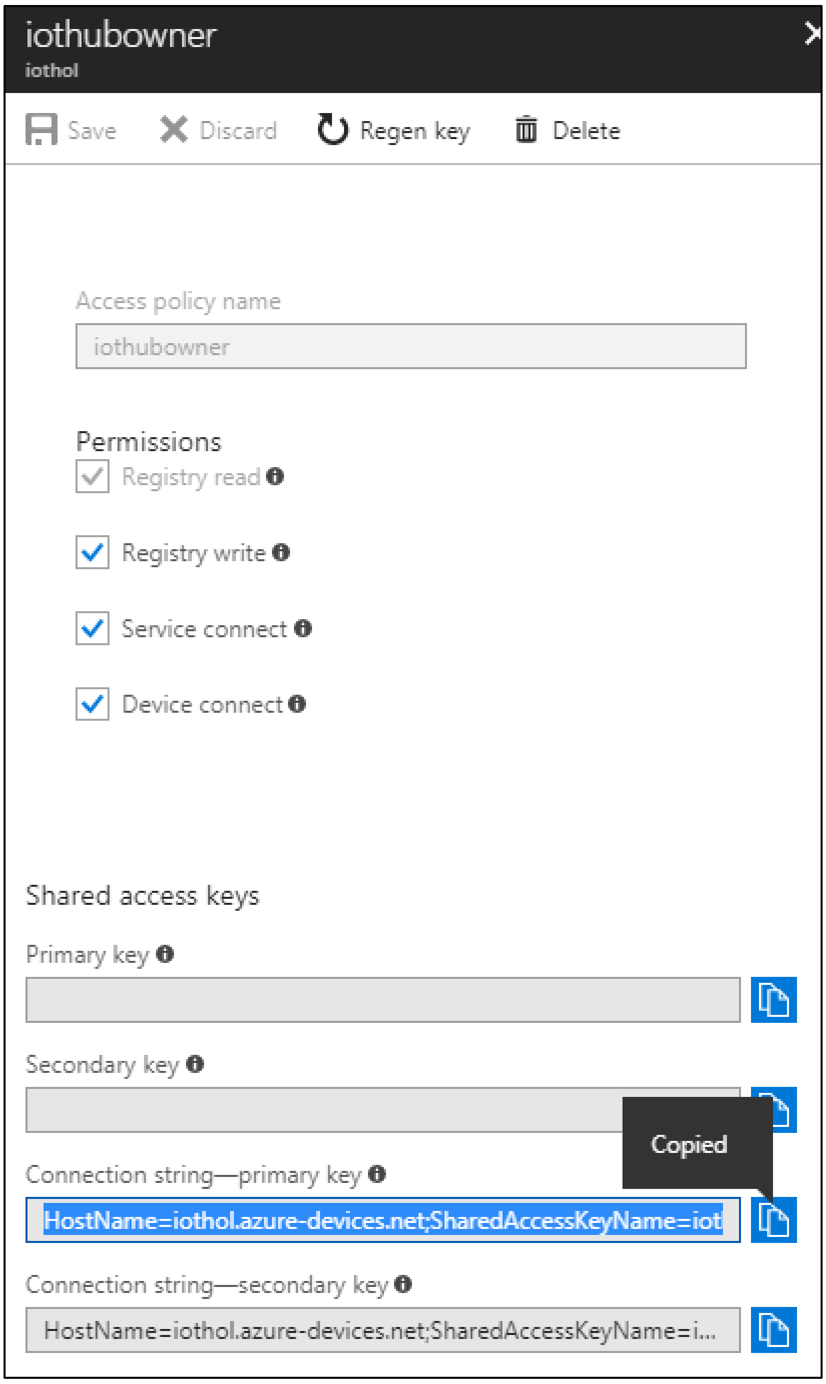


IoT Hub uses permissions to grant ac

Search to filter items...

| POLICY |
|---|
| iothubowner |
| service |
| device |
| registryRead |

On the **iothubowner** blade, copy the **Connection string – primary key** value:


iothubowner
iothol

Save  Discard  Regen key  Delete

Access policy name
iothubowner

Permissions
✓ Registry read
✓ Registry write
✓ Service connect
✓ Device connect

Shared access keys
Primary key

Secondary key

Copied

Connection string—primary key
HostName=iothol.azure-devices.net;SharedAccessKeyName=iot

Connection string—secondary key
HostName=iothol.azure-devices.net;SharedAccessKeyName=i...

Go back to your project in Visual Studio, ensure you have MainPage.xaml.cs open and replace the value for connectionString with the value you just copied from Azure Portal:

```
static string connectionString = "HostName=iothol.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=wert/Doc4s/WSDEQ/PQO5xdWL+
PQslRlgGwP+M/W9ss=";
```

Navigate back to the Azure Portal. On the IoT Hub blade, select **IoT Devices** from the left menu. You should have just the one device registered in prior steps. Copy the device id and update the value of the deviceId variable in your project in visual studio:

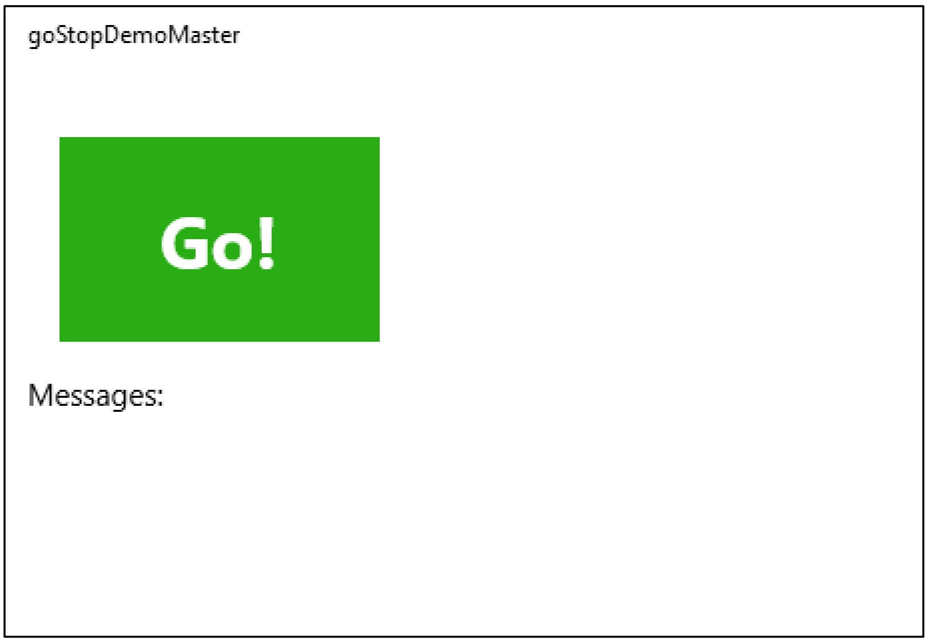
```
static string deviceId = "rpiwin";
```

## Test remote control

You should now be able to test both apps and confirm that remote control works as expected.

Check if the **goStopDemoPi** app is running in Visual Studio in debug mode. If it isn't, select the **debug** button from the toolbar or press **F5** to start debugging as done in previous steps.

Navigate back to the newly created **goStopDemoMaster** project in Visual Studio and enter debug mode for this project also. It should build successfully and present a screen showing a green **Go!** Button:



Select the green **Go!** Button and observe the results in the both the Output window and the page itself. It should show messages indicating a successful call was made to the RaspberryPi.

The **Output Window** for the **goStopDemoPi** project should also show messages indicating a call was received and successfully processed. If you have a screen connected to your RaspberryPi you will also see messages on that screen in the UWP app.

Check the RaspberryPi. The green LED should now be turned on!

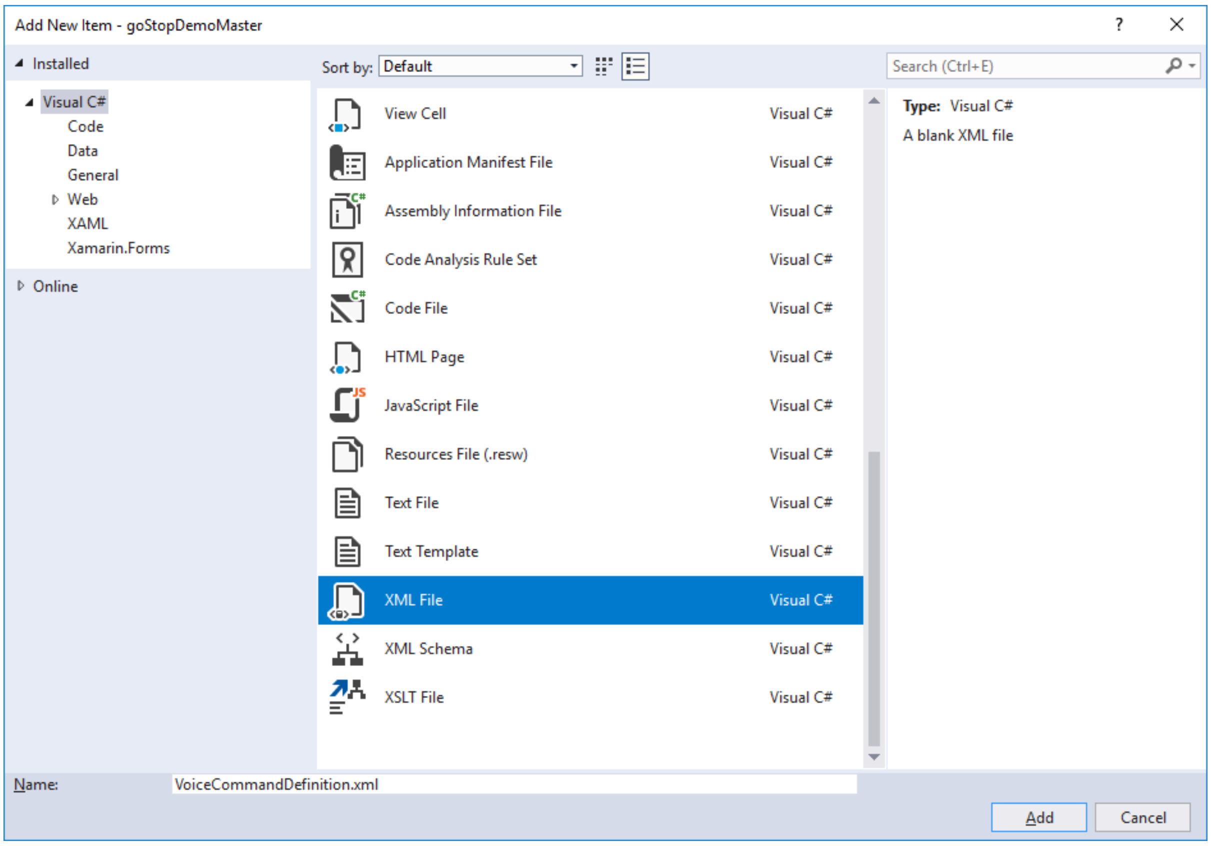
Stop the **goStopDemoMaster** project by selecting the **stop** button ■ from the toolbar or pressing **Shift + F5**.

## Part 5 – Integrate with Cortana

In this section we will add remote control by voice by integrating with Cortana.

### Add Voice Definitions

Select **Visual C#** from the left window pane, select **XML File** from the main window pane, enter **VoiceCommandDefinition.xml** as the name, then select **Add**:



Add New Item - goStopDemoMaster

Installed
- Visual C#
  - Code
  - Data
  - General
  - Web
  - XAML
  - Xamarin.Forms
- Online

Sort by: Default

Search (Ctrl+E)

| | |
|---|---|
| View Cell | Visual C# |
| Application Manifest File | Visual C# |
| Assembly Information File | Visual C# |
| Code Analysis Rule Set | Visual C# |
| Code File | Visual C# |
| HTML Page | Visual C# |
| JavaScript File | Visual C# |
| Resources File (.resw) | Visual C# |
| Text File | Visual C# |
| Text Template | Visual C# |
| XML File | Visual C# |
| XML Schema | Visual C# |
| XSLT File | Visual C# |

**Type:** Visual C#
A blank XML file

Name: VoiceCommandDefinition.xml

Add Cancel

Replace the contents of the newly added file with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.1">
  <CommandSet xml:lang="en-AU">
    <CommandPrefix>Raspberry Pi</CommandPrefix>
    <Example>Go</Example>
    <Command Name="Go">
      <Example>Go</Example>
      <ListenFor>Go</ListenFor>
      <Feedback>Raspberry Pi about to Go ...</Feedback>
      <Navigate/>
    </Command>
  </CommandSet>
  <CommandSet xml:lang="en-GB">
    <CommandPrefix>Raspberry Pi</CommandPrefix>
    <Example>Go</Example>
    <Command Name="Go">
      <Example>Go</Example>
      <ListenFor>Go</ListenFor>
      <Feedback>Raspberry Pi about to Go ...</Feedback>
      <Navigate/>
    </Command>
  </CommandSet>
</VoiceCommands>
```

## Add voice commands to MainPage.xaml.cs

Open MainPage.xaml.cs and replace all its contents with the following code:

```
using goStopDemoMaster.Classes;
using System;
using System.Threading.Tasks;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace goStopDemoMaster
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            this.NavigationCacheMode = NavigationCacheMode.Required;
        }

        protected override async void OnNavigatedTo(NavigationEventArgs e)
        {
            if (e.NavigationMode == NavigationMode.New)
            {
                await InstallVoiceCommandsAsync();
            }
        }

        private async Task InstallVoiceCommandsAsync()
        {
            StorageFile vcdFile = await StorageFile.GetFileFromApplicationUriAsync(new
Uri("ms-appx:///VoiceCommandDefinition.xml"));
            await
Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.InstallCommandDef
initionsFromStorageFileAsync(vcdFile);
        }

        private async void btnGo_Click(object sender, RoutedEventArgs e)
        {
            var msgUpdate = new Progress<string>(msg =>
            {
                textBlock.Text += (textBlock.Text == string.Empty) ? msg : "\n\n" +
msg;
            });
            await Task.Run(() => IoTHub.InvokeMethod("RaspberryPiGo", msgUpdate));
        }
    }
}
```

## Update app.xaml.cs to initialise voice commands

Open app.xaml.cs and replace all its contents with the following code:

```
using System;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;
using Windows.Media.SpeechRecognition;
using System.Threading.Tasks;
using goStopDemoMaster.Classes;

namespace goStopDemoMaster
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application
class.
    /// </summary>
    sealed partial class App : Application
    {
        /// <summary>
        /// Initializes the singleton application object.  This is the first line of
authored code
        /// executed, and as such is the logical equivalent of main() or WinMain().
        /// </summary>
        public App()
        {
            this.InitializeComponent();
            this.Suspending += OnSuspending;
        }

        /// <summary>
        /// Invoked when the application is launched normally by the end user.  Other
entry points
        /// will be used such as when the application is launched to open a specific
file.
        /// </summary>
        /// <param name="e">Details about the launch request and process.</param>
        protected override void OnLaunched(LaunchActivatedEventArgs e)
        {
            Frame rootFrame = Window.Current.Content as Frame;

            // Do not repeat app initialization when the Window already has content,
            // just ensure that the window is active
            if (rootFrame == null)
            {
                // Create a Frame to act as the navigation context and navigate to the
first page
                rootFrame = new Frame();

                rootFrame.NavigationFailed += OnNavigationFailed;

                if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)
                {
                    //Load state from previously suspended application
                }

                // Place the frame in the current Window
                Window.Current.Content = rootFrame;
            }
            if (e.PrelaunchActivated == false)
            {
                if (rootFrame.Content == null)
                {
                    // When the navigation stack isn't restored navigate to the first
page,
                    // configuring the new page by passing required information as a
navigation
                    // parameter
                    rootFrame.Navigate(typeof(MainPage), e.Arguments);
                }
                // Ensure the current window is active
                Window.Current.Activate();
            }
        }

        /// <summary>
        /// Invoked when Navigation to a certain page fails
        /// </summary>
        /// <param name="sender">The Frame which failed navigation</param>
        /// <param name="e">Details about the navigation failure</param>
        void OnNavigationFailed(object sender, NavigationFailedEventArgs e)
        {
            throw new Exception("Failed to load Page " + e.SourcePageType.FullName);
        }

        /// <summary>
        /// Invoked when application execution is being suspended.  Application state
is saved
        /// without knowing whether the application will be terminated or resumed with
the contents
        /// of memory still intact.
        /// </summary>
        /// <param name="sender">The source of the suspend request.</param>
        /// <param name="e">Details about the suspend request.</param>
        private void OnSuspending(object sender, SuspendingEventArgs e)
        {
            var deferral = e.SuspendingOperation.GetDeferral();
            //Save application state and stop any background activity
            deferral.Complete();
        }

        protected override void OnActivated(IActivatedEventArgs args)
        {
            base.OnActivated(args);
            if (args.Kind == ActivationKind.VoiceCommand)
            {
                var commandArgs = args as VoiceCommandActivatedEventArgs;
                if (commandArgs != null)
                {
                    SpeechRecognitionResult speechRecognitionResult =
commandArgs.Result;
                    var voiceCommandName = speechRecognitionResult.RulePath[0];
                    switch (voiceCommandName)
                    {
                        case "Go":
                            Task.Run(() => IoTHub.InvokeMethod("RaspberryPiGo"));
                            break;
                    }
                }
            }
            Window.Current.Activate();
        }
    }
}
```

### Test remote control via Cortana

Ensure the **goStopDemoPi** project is currently running. If not, run it in debug mode as done previously.

Run the **goStopDemoMaster** project as done previously.

Click the microphone button and speak the words 'Raspberry Pi go'. You should receive feedback from Cortana stating that the Raspberry Pi is about to go! Moments later the green LED on the Raspberry Pi will be turned on. Check the Raspberry Pi to confirm.

**Note:** If you experience issues with Cortana not issuing the command to your app the work around is to install the app and run it from there.

## Conclusion

You've built an end to end solution that allows controlling a device remotely and receiving feedback from that device by turning on an LED and responding accordingly. The solution integrates with Cortana to allow controlling the remote device by voice.

## Additional Exercises

Add a red LED to the Raspberry Pi.

Modify the **goStopDemoPi** project to allow remotely turning on the red LED.

Add a red button to the **goStopDemoMaster** project that triggers turning on the red LED.

Add another button to the **goStopDemoMaster** project that turns off both LEDs.

Add voice commands to integrate control of the red LED with Cortana, allowing it to be turned on by voice, also allow turning them off. Use voice commands such as 'Raspberry Pi stop' to turn on the red LED and turn off the green LED and 'Raspberry Pi reset' to turn them both off.