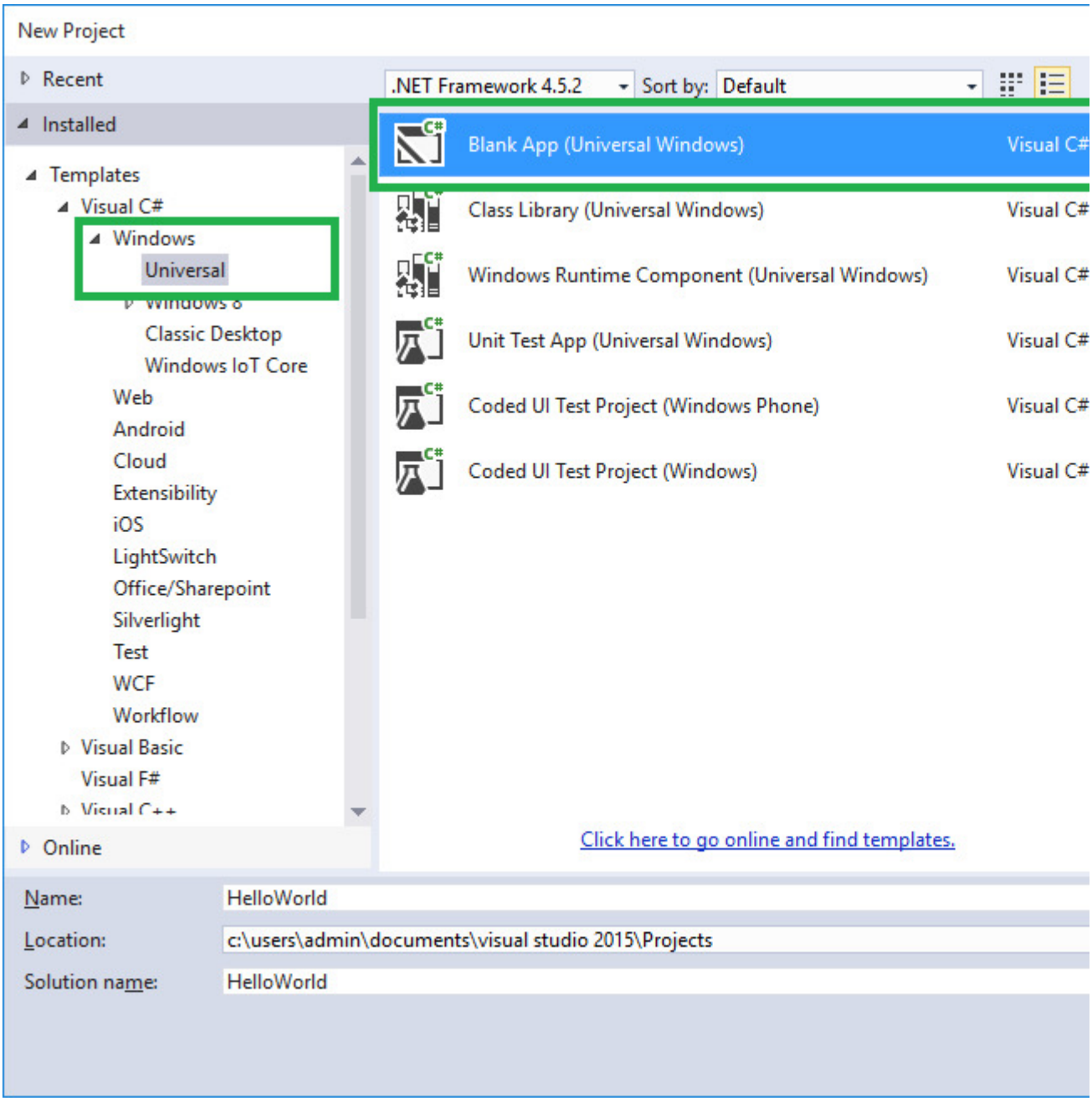
‘Hello, world!’

In this sample, we will create and deploy the proverbial 1st app, “Hello, world!” to any device running Windows 10 IoT Core.

Create a new C# project

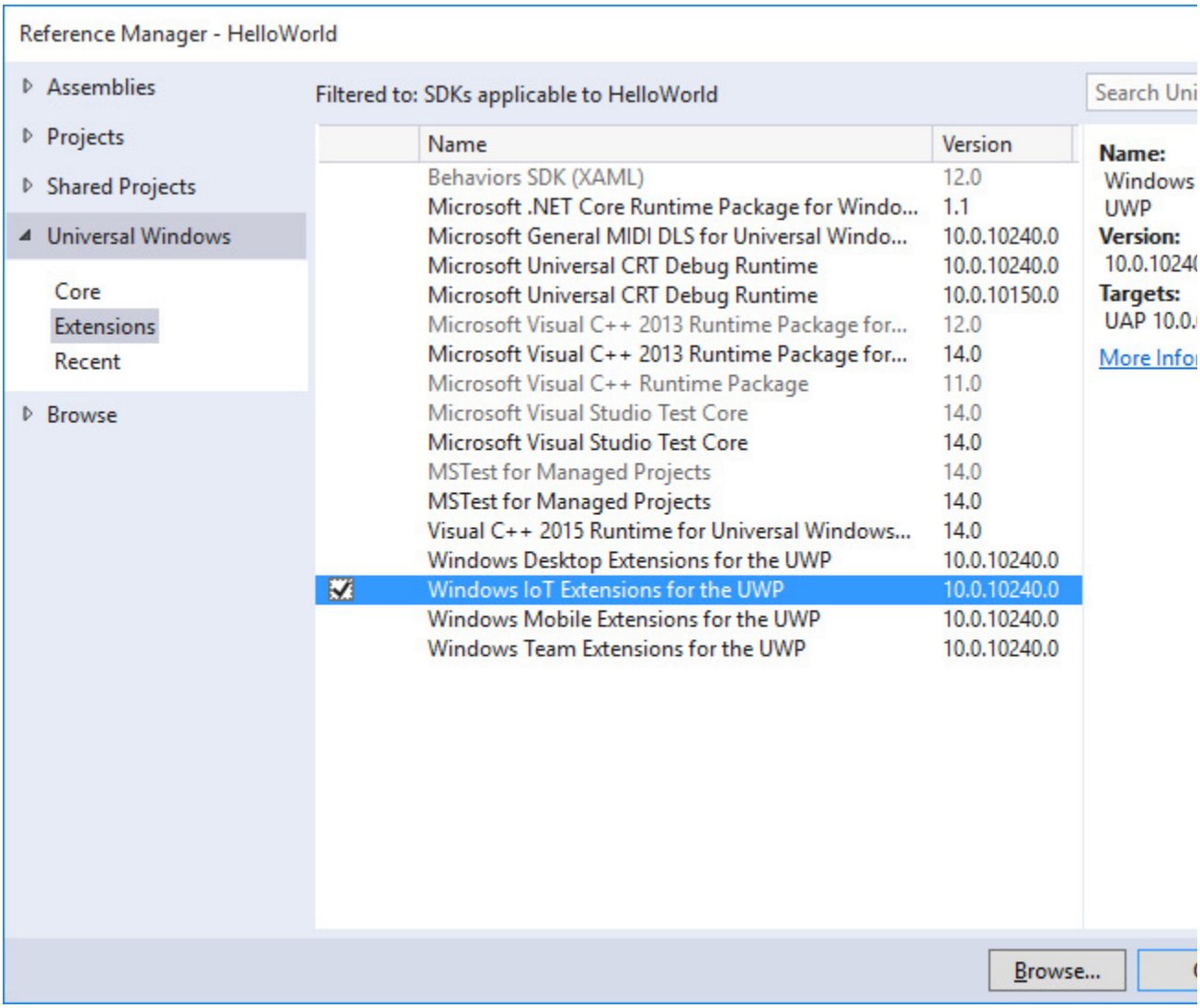
All of the sample code is available to download, but as an exercise, this tutorial will take you through the complete steps to create this app from scratch. You need to make sure you have installed the Windows 10 IoT Core Project Templates from here.
Start Visual Studio 2017.
Create a new project with **(File | New Project...)**.
In the **New Project** dialog, navigate to **Universal** as shown below (in the left pane in the dialog: Templates | Visual C# | Windows | Universal).
Select the template **Blank App (Windows Universal)**.
Remember to give a good name to your first app! In this example, we called the project ‘HelloWorld’.



NOTE
If this is the first project you create, Visual Studio will likely prompt you to enable developer mode for Windows 10.

Add a reference to the Windows IoT extension SDK

Since the IoT extension SDK is not added to projects by default, we'll need to add a reference so that namespaces like **Windows.Devices.Gpio** will be available in the project. To do so, just right-click on the References entry under the project, Select "Add Reference" then navigate the resulting dialog to **Universal Windows->Extensions->Windows IoT Extensions for the UWP**, check the box, and click OK.



Add content to MainPage.xaml

Let's add some content to the MainPage. From Solution Explorer, select the **MainPage.xaml** file. We want to add a TextBox and a Button, to show some interaction. So we will edit the XAML file to add these elements. Locate the **** tag in the XAML section of the designer, and add the following markup.

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
    <TextBox x:Name="HelloMessage" Text="Hello, World!" Margin="10" IsReadOnly=
    <Button x:Name="ClickMe" Content="Click Me!"  Margin="10" HorizontalAlignme
    </StackPanel>
</Grid>
```

Now that we have a TextBox and a Button, we can add some code which will be executed when the Button is pressed. Double click on the Button in the design surface: Visual Studio will add a **Click** property to the Button XAML tag and generate the **ClickMe_Click** method in **MainPage.xaml.cs**. Let's add a simple line of code in this method.
MainPage.xaml:

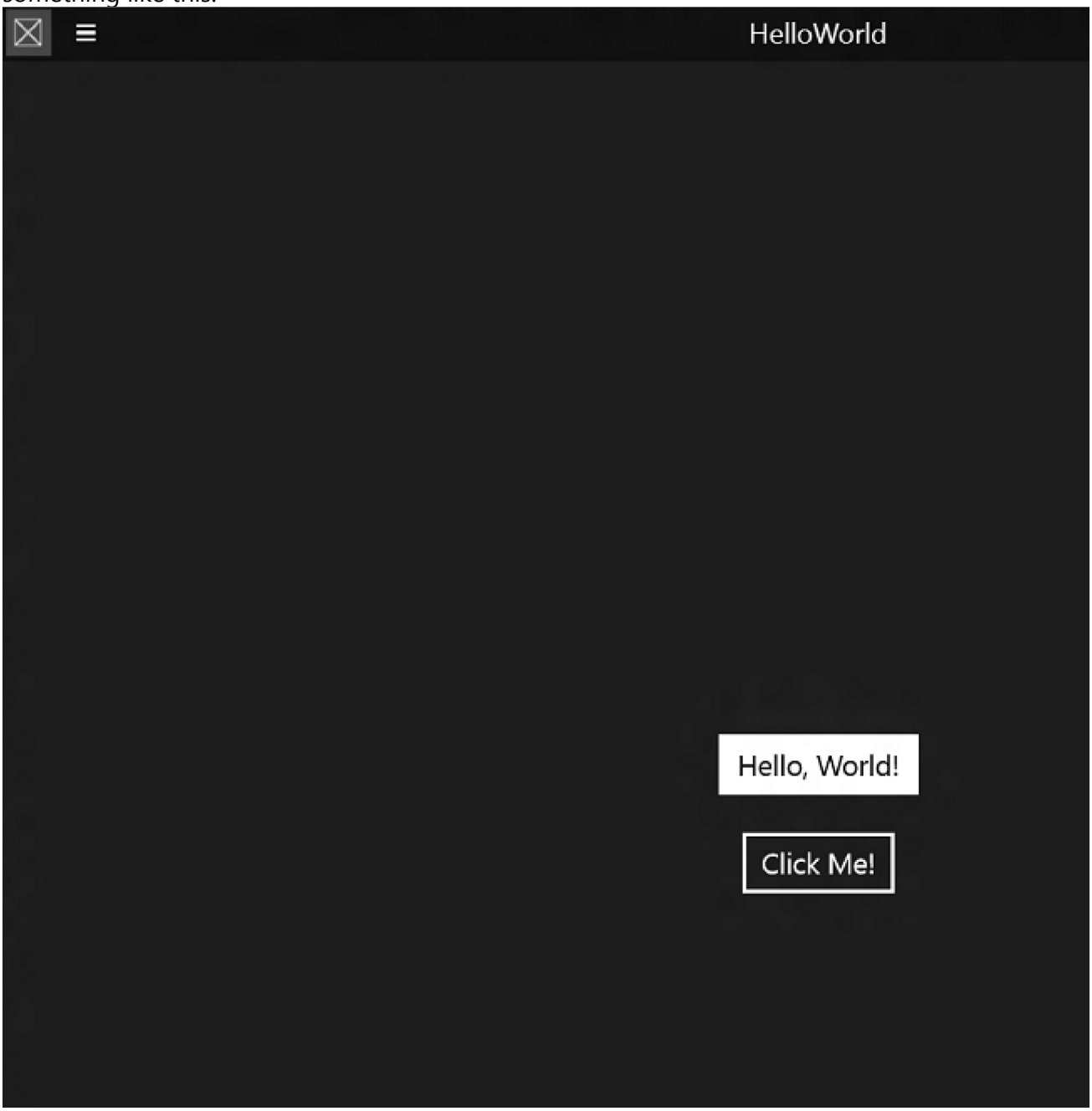
```
<Button x:Name="ClickMe" Content="Click Me!"  Margin="10" HorizontalAlignment="
```

MainPage.xaml.cs:

```
private void ClickMe_Click(object sender, RoutedEventArgs e)
{
    this.HelloMessage.Text = "Hello, Windows 10 IoT Core!";
}
```

Build and test the app locally

Make sure the app builds correctly by invoking the **Build | Build** Solution menu command. Since this is a Universal Windows Platform (UWP) application, you can test the app on your Visual Studio machine as well: Just press F5, and the app will run inside your machine. You should see something like this:



Close the app after you're done validating it.

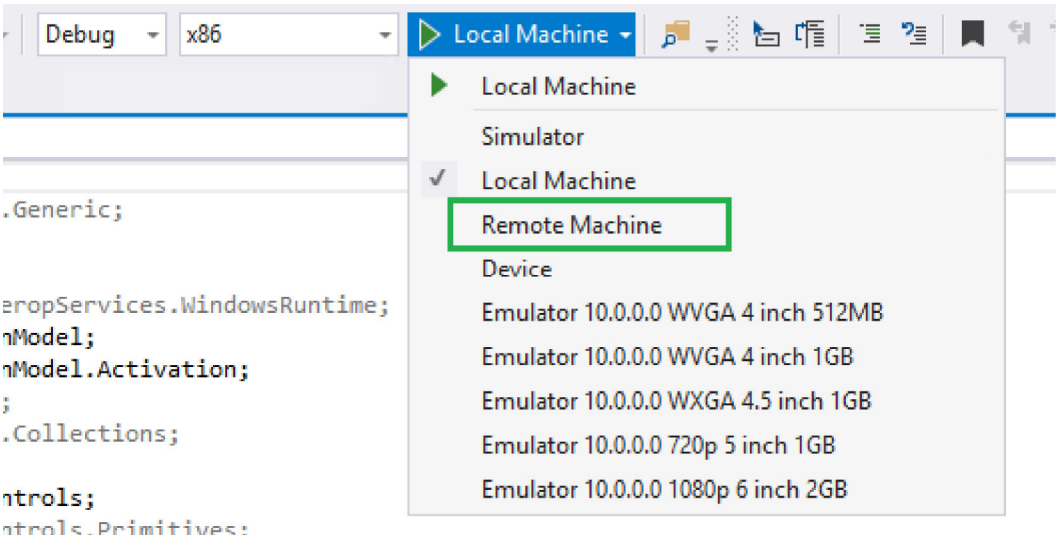
If you would like to learn more about Universal Windows Platform applications, click here.

Deploy the app to your Windows IoT Core device

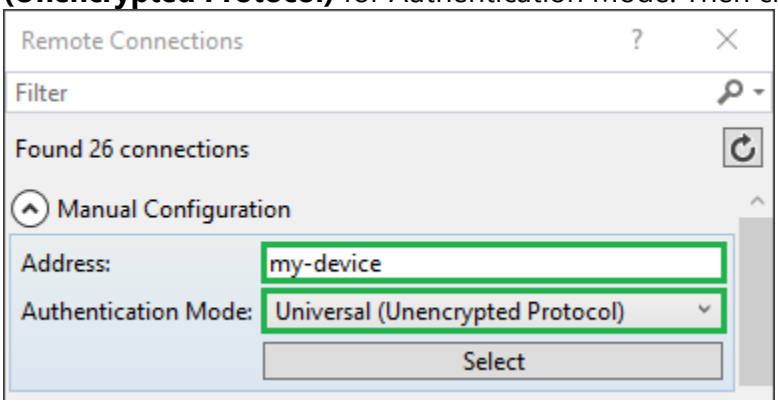
Of course, we want to deploy our first app to our IoT Core device. It's easy. In the PowerShell documentation, you can find instructions to chose a unique name for your IoT Core device. In this sample, we'll use that name (though you can use your IP address as well) in the 'Remote Machine Debugging' settings in Visual Studio.

If you're building for Minnowboard Max, select **x86** in the Visual Studio toolbar architecture dropdown. If you're building for Raspberry Pi 2 or 3 or the DragonBoard, select **ARM**.

Next, in the Visual Studio toolbar, click on the **Local Machine** dropdown and select **Remote Machine**



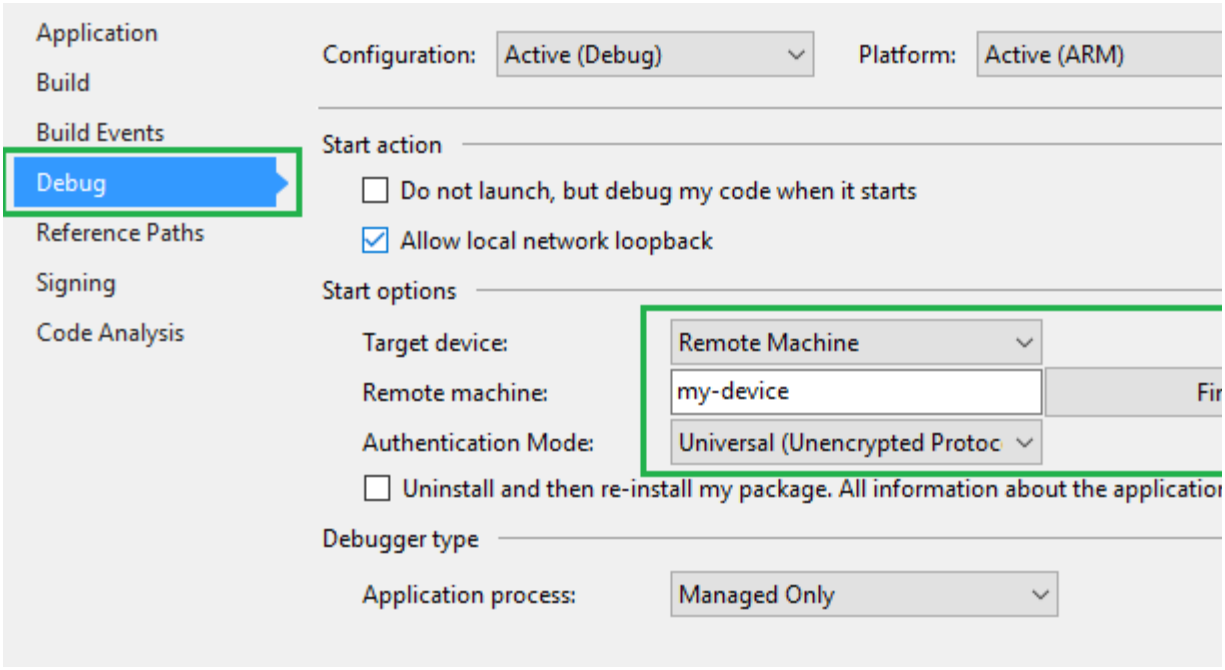
At this point, Visual Studio will present the 'Remote Connections' dialog. Put the IP address or name of your IoT Core device (in this example, we're using 'my-device') and select **Universal (Unencrypted Protocol)** for Authentication Mode. Then click **Select**.



Couple of notes:

You can use the IP address instead of the IoT Core device name.

You can verify and/or modify these values navigating to the project properties (select 'Properties' in the Solution Explorer) and choose the 'Debug' tab on the left:



Now we're ready to deploy to the remote IoT Core device. Simply press F5 (or select **Debug | Start Debugging**) to start debugging our app. You should see the app come up in IoT Core device screen, and you should be able to click on the button.
You can set breakpoints, see variable values, etc. To stop the app, press on the 'Stop Debugging' button (or select Debug | Stop Debugging).
Having successfully deployed and debugged your first UWP application, create a Release version by simply changing the Visual Studio toolbar configuration dropdown from **Debug** to **Release**.
You can now build and deploy your app to your device by selecting **Build | Rebuild Solution and Build | Deploy Solution**.
Congratulations! You just deployed your first UWP application to a device running IoT Core!

Set your app as the Startup App

You can set your app to be the 'Startup App' for your Windows IoT Core device, so that when the device reboot, it will start your app automatically. To do so, you'll need to run a command line utility called iotstartup on the Windows IoT Core device. We will do this using PowerShell.
Start a PowerShell (PS) session with your Windows IoT Core device as described here.
From the PS session, type (for simplicity, we will assume the app's name is HelloWorld, **please substitute your app's actual name**):

```
[192.168.0.243]: PS C:\> iotstartup list HelloWorld
```

and you should see the full name of the UWP application, i.e. something like:

```
Headed    : HelloWorld_n2pe7ts0w7wey!App
```

the utility is confirming that your app is an 'headed' application, and is installed correctly.
Now, it's easy to set this app as the 'Startup App'. Just type the command:

```
[192.168.0.243]: PS C:\> iotstartup add headed HelloWorld
```

The utility will confirm that the new Startup headed app is now your app:

```
AppId changed to HelloWorld_n2pe7ts0w7wey!App
```

Go ahead and restart your Windows IoT Core device. From the PS session, you can issue the shutdown command:

```
[192.168.0.243]: PS C:\> shutdown /r /t 0
```

Once the device has restarted, you'll see your app start automatically.
At this point, you can revert back to using the DefaultApp as your 'Startup App'. Just type the command:

```
[192.168.0.243]: PS C:\> iotstartup add headed IoTCoreDefaultApp
```

The utility will confirm that the new Startup headed app is now IoTCoreDefaultApp:

```
AppId changed to IoTCoreDefaultApp_kwmcxzszfer2y!App
```