Lab 5 Install IoT Core, Hello World, Blinky and BlinkyIoT

By: Kevin Saye

IoT Solution Architect

December, 2017

This lab assumes you have completed Lab 4.

If you have any issues or concerns, please email: virtualbootcamphelp@microsoft.com.

Execution Time: 30 minutes.

Required Hardware:

- Windows 10 PC
- IoT Hardware kit: https://www.adafruit.com/product/3605 or similar hardware.
- Access to a WiFi network (without a captive portal aka web page login)

Required Operating System:

- Windows 10

Other Requirements:

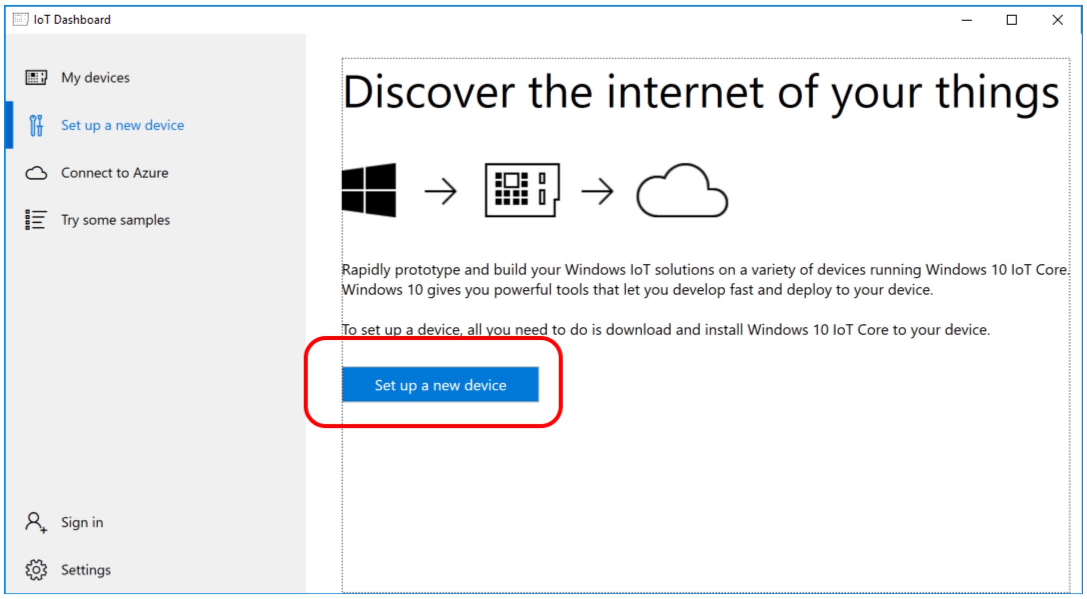
- Azure Subscription

Required Software:

Software	Size	Installation URL
Visual Studio 2017	150 MB	https://aka.ms/vs/15/release/vs_community.exe
IoT Core Dashboard	57 MB	https://go.microsoft.com/fwlink/?LinkID=708576

Step 1. If not already installed, install the IoT Dashboard here: https://go.microsoft.com/fwlink/?LinkID=708576. Launch the IoT Dashboard.

Step 2. Click, "Setup a new device" as shown below:

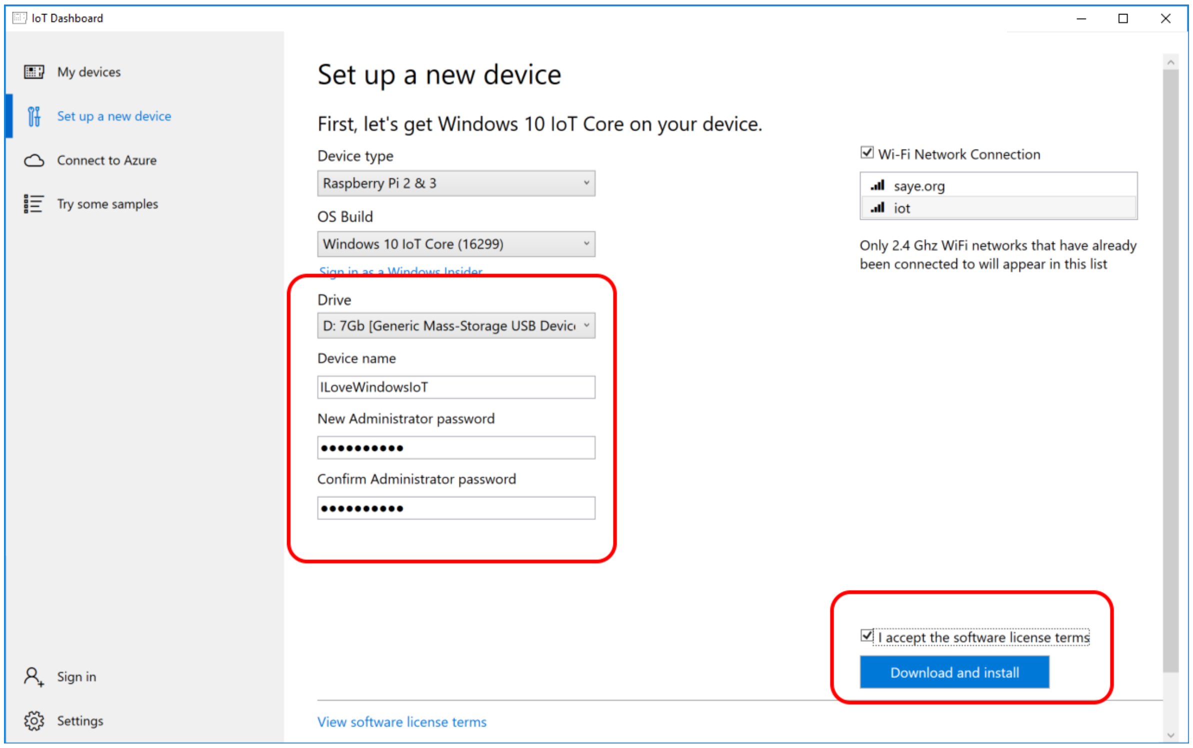


Discover the internet of your things

Rapidly prototype and build your Windows IoT solutions on a variety of devices running Windows 10 IoT Core. Windows 10 gives you powerful tools that let you develop fast and deploy to your device.

To set up a device, all you need to do is download and install Windows 10 IoT Core to your device.

Step 3. Insert your SD Card, name your device, define a password and click "Download and install". if your PC does not have a SD Card reader, the Adafruit package come with a USB Micro SD Card reader. Note this will take a while, as it downloads a 800 MB ISO. Your screen will as for admin approval a few times and you will see the DOS screen performing the required task.



Set up a new device

First, let's get Windows 10 IoT Core on your device.

Device type: Raspberry Pi 2 & 3

OS Build: Windows 10 IoT Core (16299)

Drive: D: 7Gb [Generic Mass-Storage USB Devic

Device name: ILoveWindowsIoT

New Administrator password: ••••••••••

Confirm Administrator password: ••••••••••

☑ Wi-Fi Network Connection

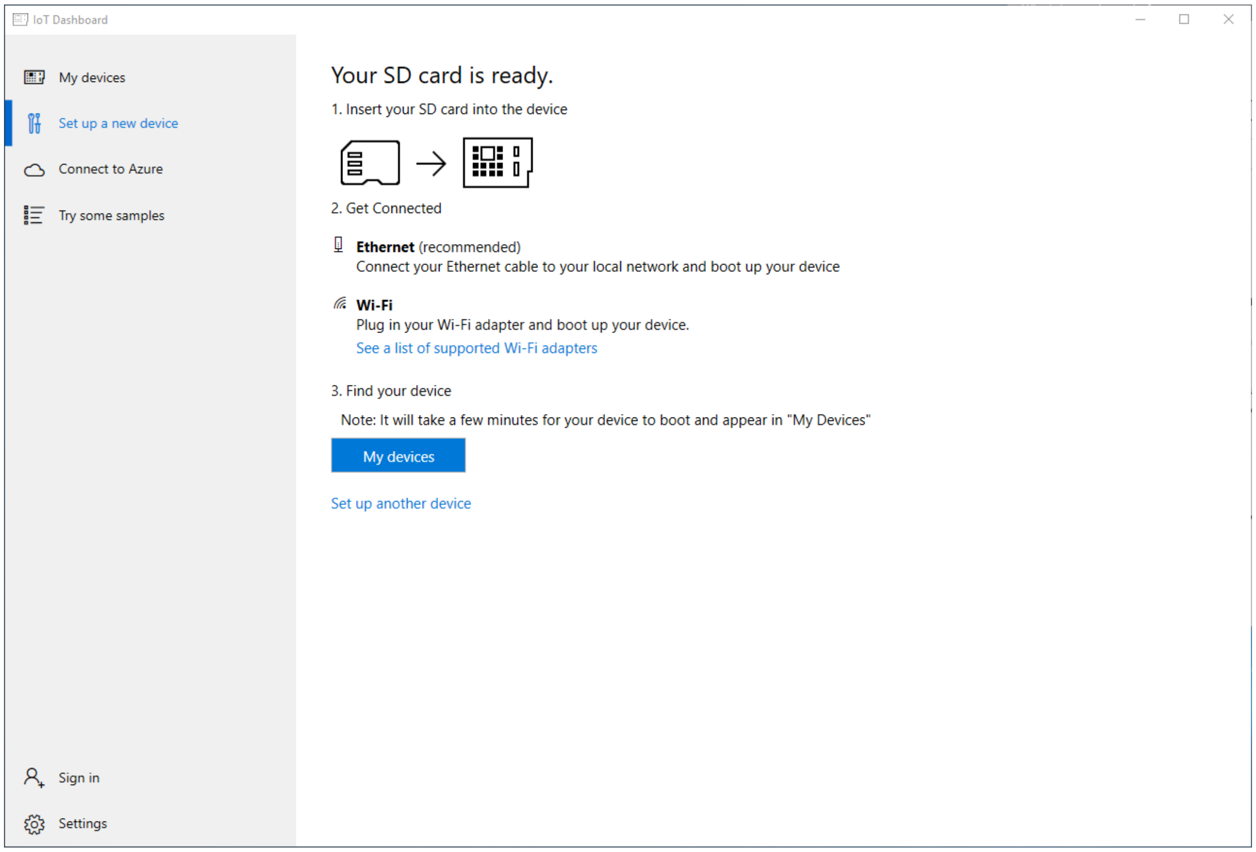
saye.org

iot

Only 2.4 Ghz WiFi networks that have already been connected to will appear in this list

☑ I accept the software license terms

Step 4. Once your SD card has been flashed, you will see the following screen.



Your SD card is ready.

1. Insert your SD card into the device

2. Get Connected

Ethernet (recommended)
Connect your Ethernet cable to your local network and boot up your device

Wi-Fi
Plug in your Wi-Fi adapter and boot up your device.
See a list of supported Wi-Fi adapters

3. Find your device

Note: It will take a few minutes for your device to boot and appear in "My Devices"

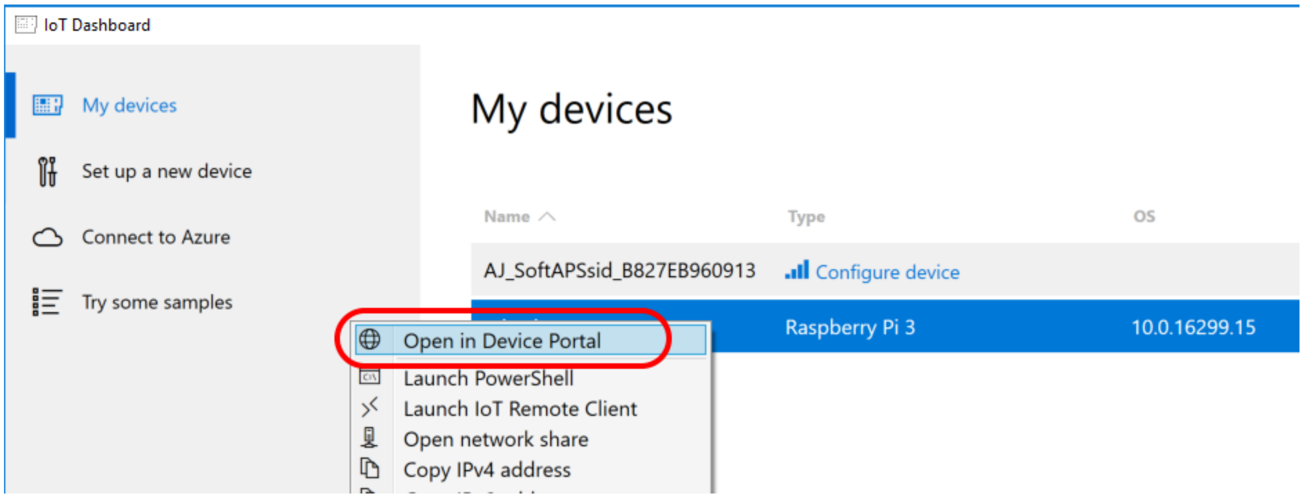
My devices

Set up another device

Step 5. Insert the SD card in Raspberry Pi, Connect a USB Keyboard, a USB mouse and a display to the Raspberry Pi.

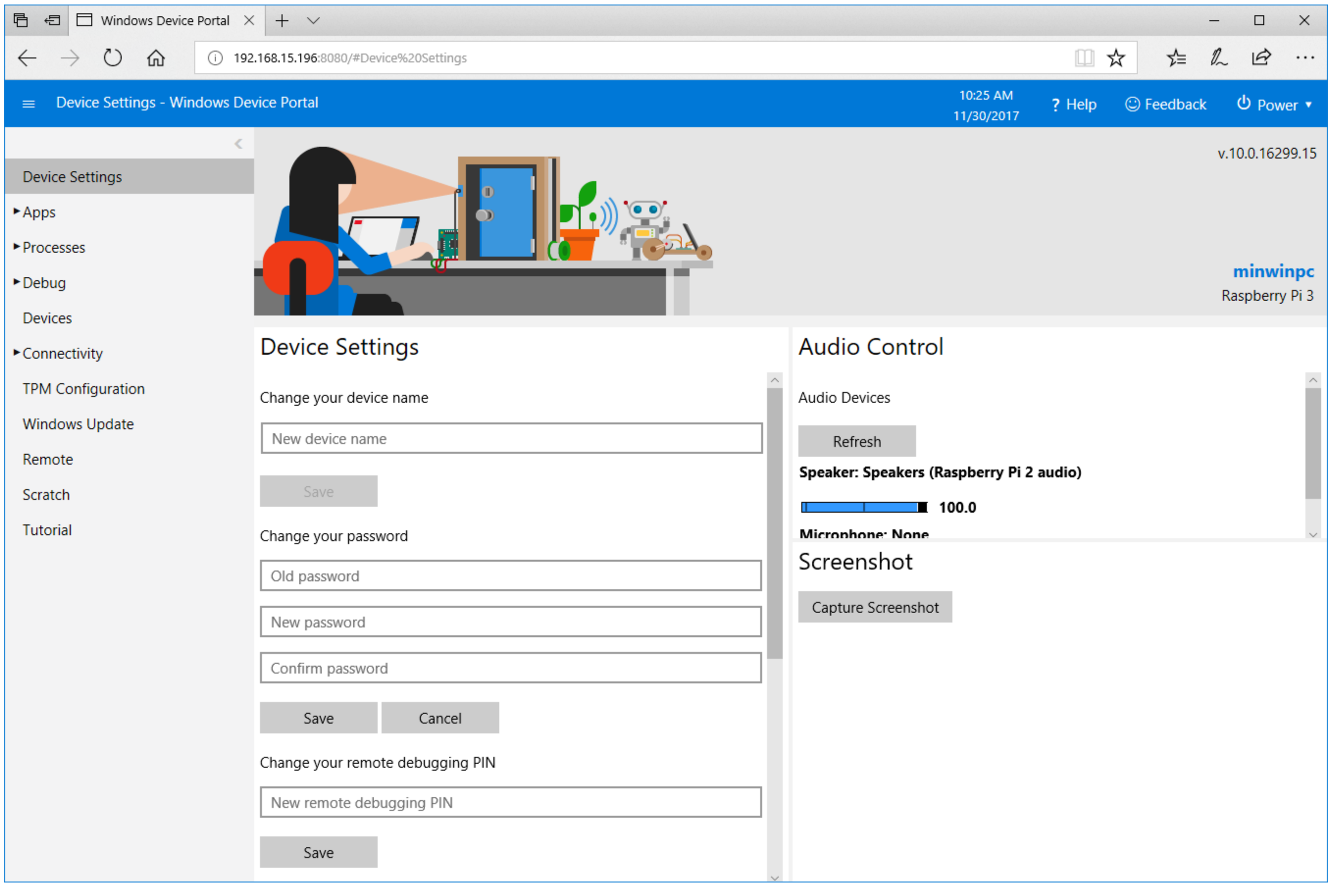
Step 6. Power on the Raspberry Pi. It will boot Windows 10 IoT Core automatically. The first time you are booting, this will take a few minutes.

Step 7. Soon, you should see your PI from the IoT Dashboard, as shown below. (Optional) You can view the properties via the Device Portal. Log in as Administrator with the matching password, as shown below:



My devices

Name	Type	OS
AJ_SoftAPSsid_B827EB960913	Configure device	
	Raspberry Pi 3	10.0.16299.15



Device Settings

Change your device name

New device name

Save

Change your password

Old password

New password

Confirm password

Save Cancel

Change your remote debugging PIN

New remote debugging PIN

Save

Audio Control

Audio Devices

Refresh

Speaker: Speakers (Raspberry Pi 2 audio)

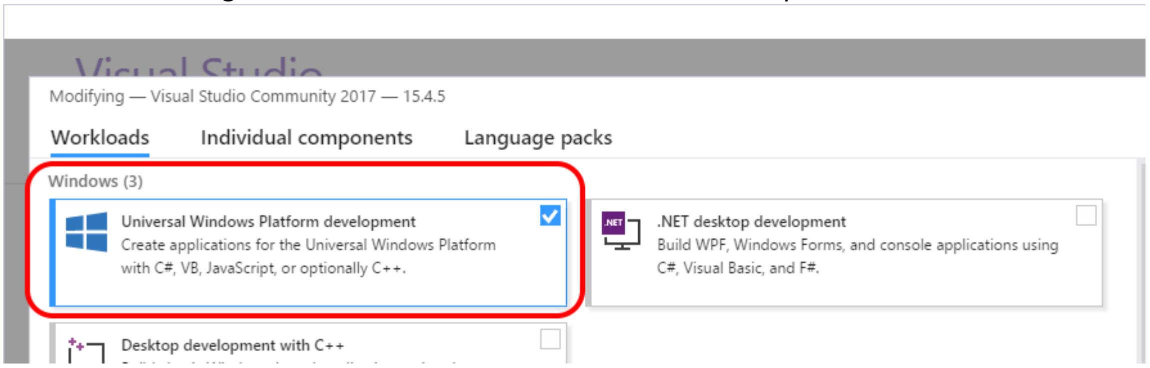
100.0

Microphone: None

Screenshot

Capture Screenshot

Step 8. If not already installed, in stall Visual Studio from: https://aka.ms/vs/15/release/vs_community.exe Take the default settings and add Universal Windows Platform development as shown below:



Visual Studio

Modifying — Visual Studio Community 2017 — 15.4.5

Workloads | Individual components | Language packs

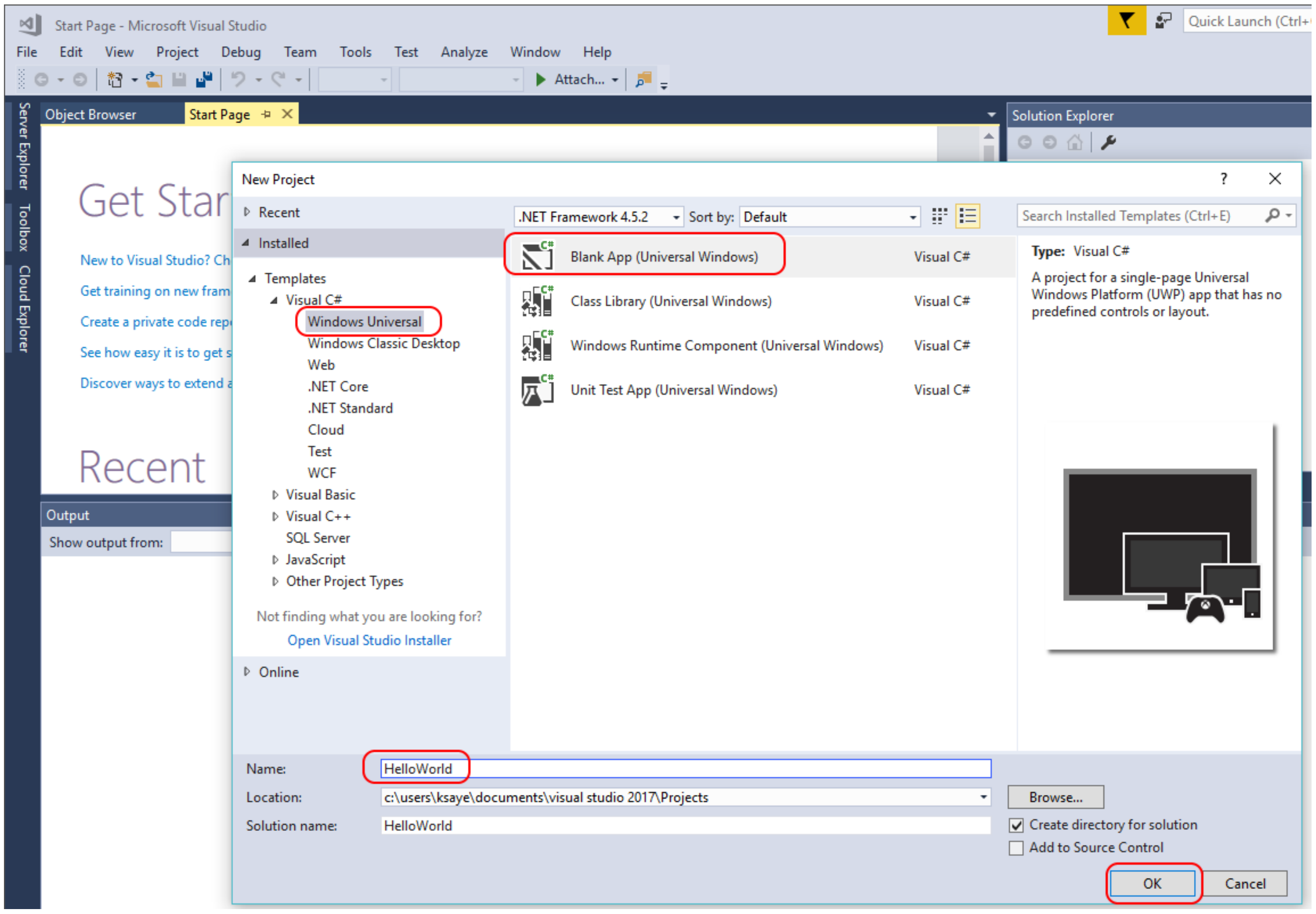
Windows (3)

[✓] Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

[ ] .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

[ ] Desktop development with C++

Step 9. Open Visual Studio 2017, select File → New Project and select Universal App → Blank App, as shown below. Name the app something unique, like HelloWorld.



New Project

- Recent
- Installed
  - Templates
    - Visual C#
      - Windows Universal
      - Windows Classic Desktop
      - Web
      - .NET Core
      - .NET Standard
      - Cloud
      - Test
      - WCF
    - Visual Basic
    - Visual C++
    - SQL Server
    - JavaScript
    - Other Project Types
- Not finding what you are looking for?
- Open Visual Studio Installer
- Online

.NET Framework 4.5.2 Sort by: Default

Search Installed Templates (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#

Type: Visual C#

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

Name: HelloWorld

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

Solution name: HelloWorld

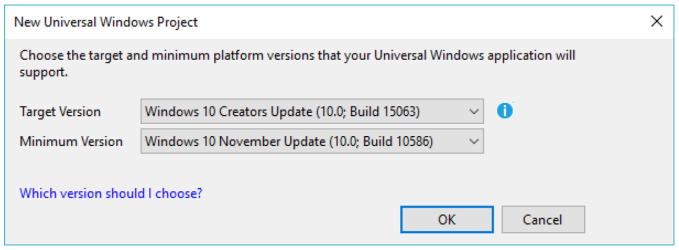
Browse...

[✓] Create directory for solution

[ ] Add to Source Control

OK Cancel

Step 10. Select the build versions shown below:



New Universal Windows Project

Choose the target and minimum platform versions that your Universal Windows application will support.

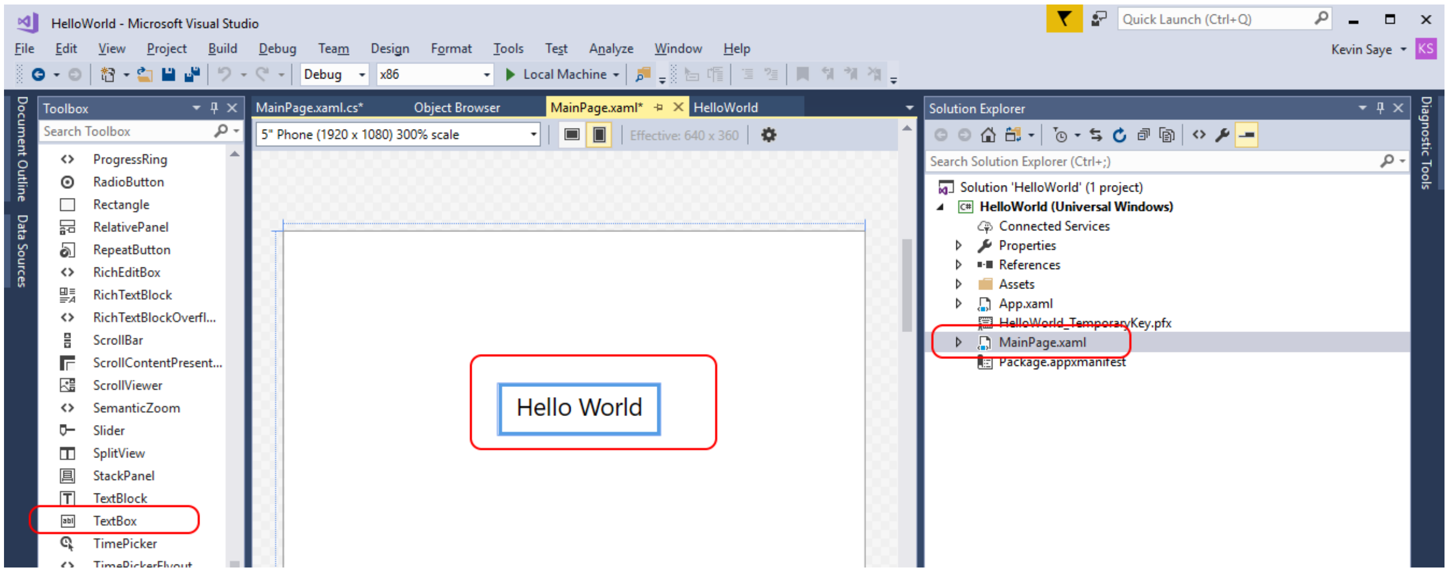
Target Version: Windows 10 Creators Update (10.0; Build 15063)

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

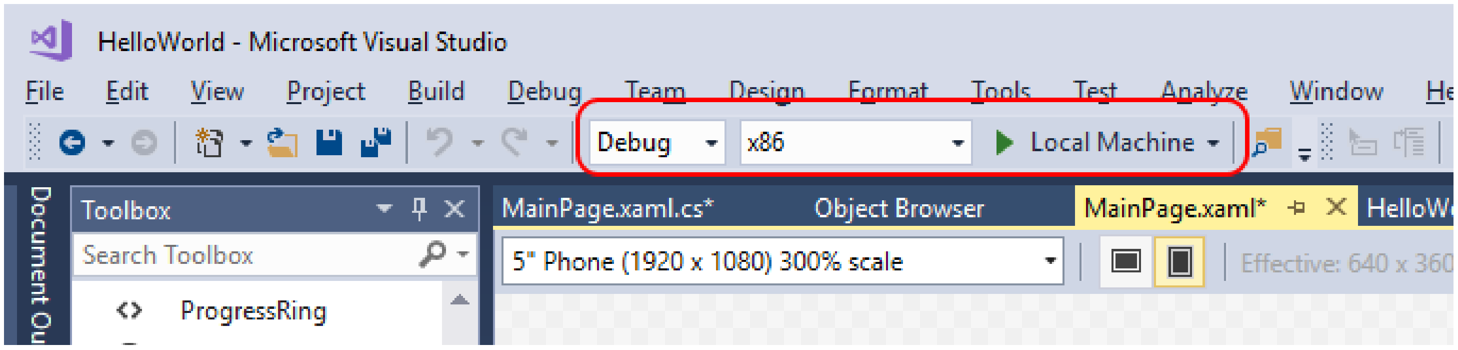
Which version should I choose?

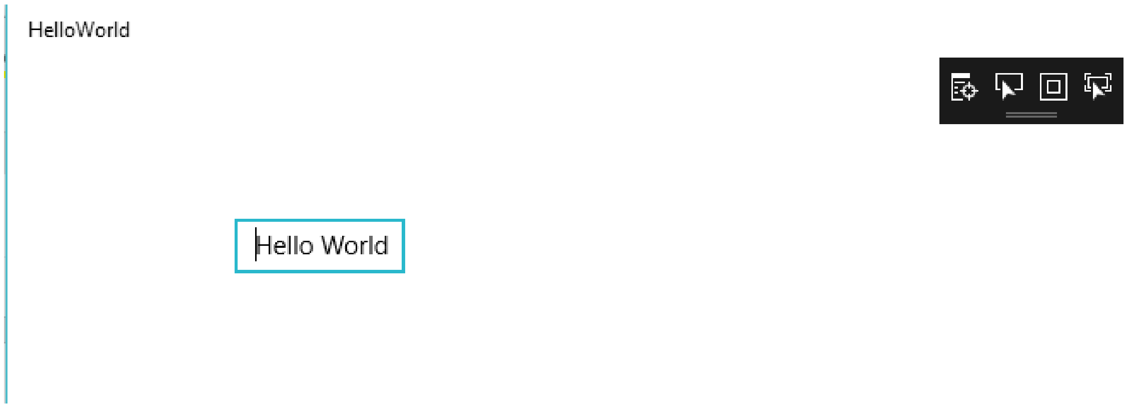
OK Cancel

Step 11. Double click MainPage.xaml on the right, then drag a Text Box to the center page. Change the text to "Hello World" as shown below.

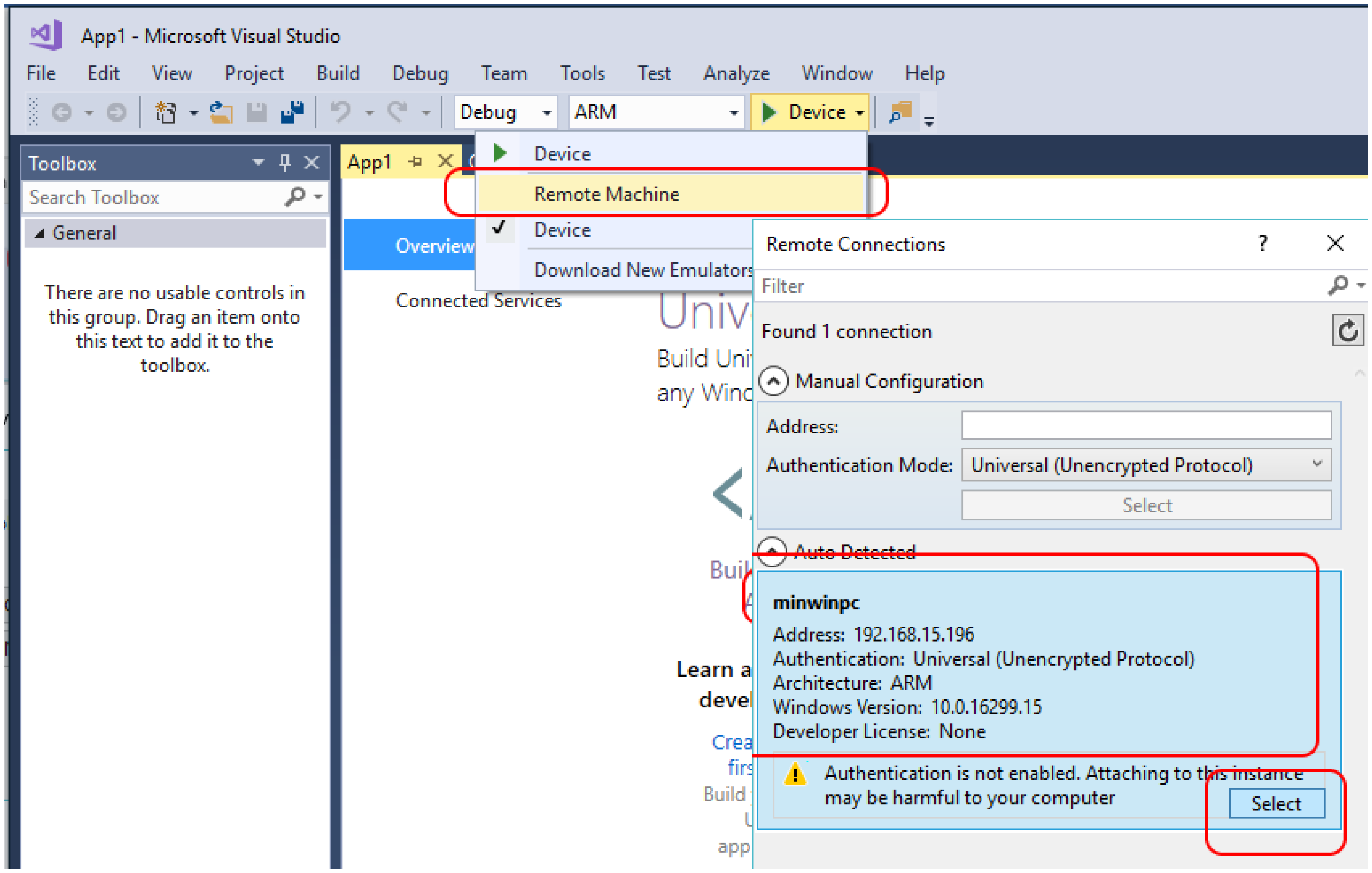


Step 12. With Debug Mode set and x86 platform, click the "Local Machine" debug button and you will see the application start, as shown below:


HelloWorld - Microsoft Visual Studio
File Edit View Project Build Debug Team Design Format Tools Test Analyze Window He
Debug
x86
Local Machine
Toolbox
Search Toolbox
ProgressRing
MainPage.xaml.cs*
Object Browser
MainPage.xaml*
HelloW
5" Phone (1920 x 1080) 300% scale
Effective: 640 x 360



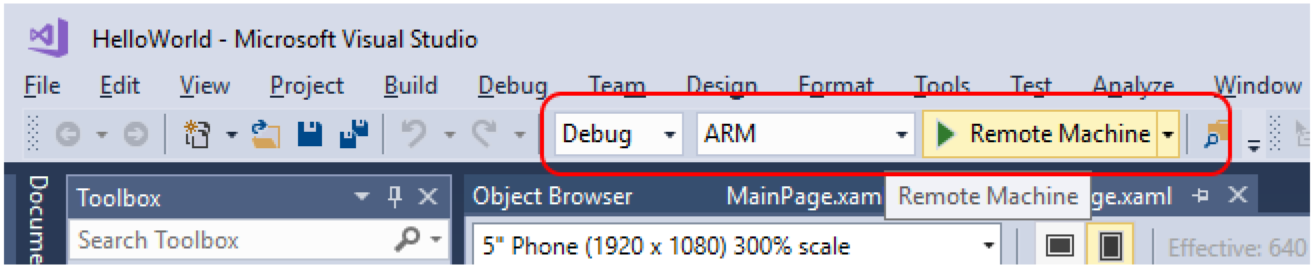
Step 13. With Debug Mode still set change the platform to ARM click Device → Remote Machine. Select the machine auto detected or manually type in the ip address as shown below.


App1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug ARM Device
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
App1
Overview
Connected Services
Device
Remote Machine
Device
Download New Emulators
Remote Connections
Filter
Found 1 connection
Manual Configuration
Address:
Authentication Mode: Universal (Unencrypted Protocol)
Select
Auto Detected

minwinpc
Address: 192.168.15.196
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

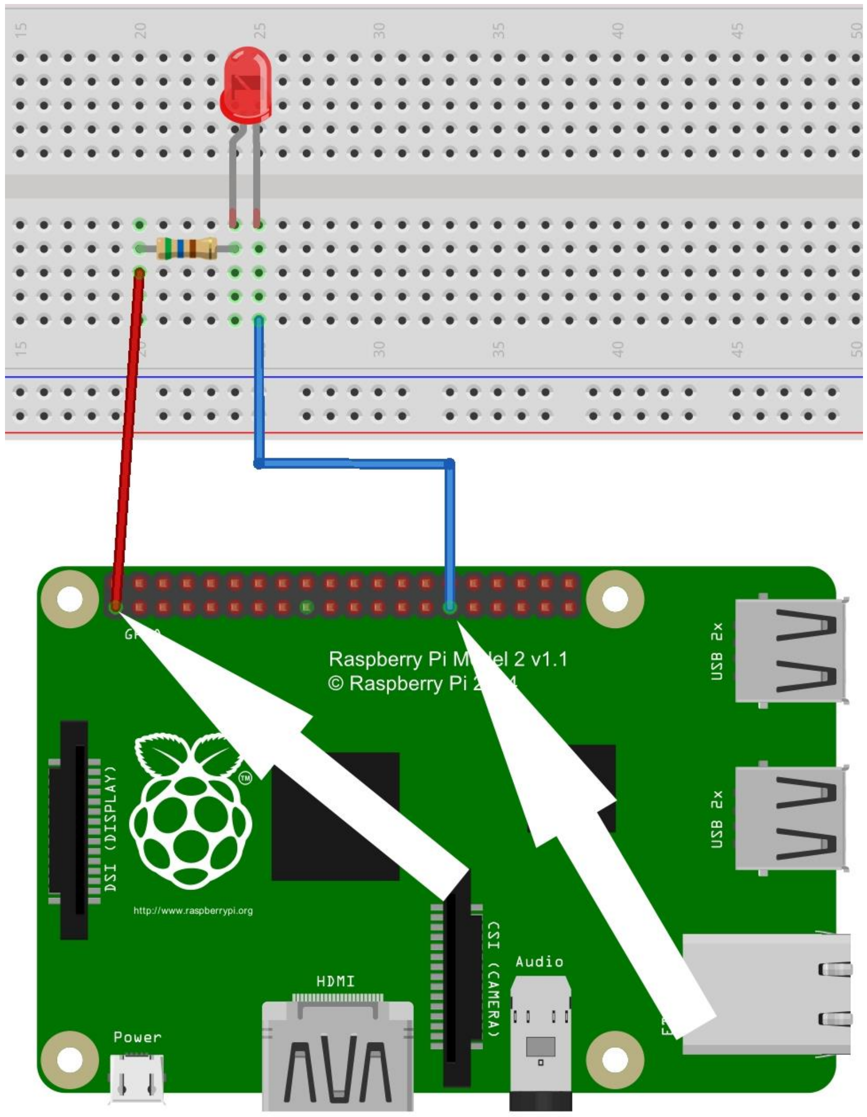
Step 14. With Debug and ARM still selected, click Remote Machine. After it compiles and deploys (installs requirements the first time), your hello world application will launch on the Windows IoT Core device.


HelloWorld - Microsoft Visual Studio
File Edit View Project Build Debug Team Design Format Tools Test Analyze Window
Debug ARM Remote Machine
Toolbox
Search Toolbox
Object Browser MainPage.xam Remote Machine ge.xaml
5" Phone (1920 x 1080) 300% scale
Effective: 640

Step 15. Download the sample code from: https://github.com/ms-iot/samples/archive/develop.zip

Step 16. Copy the contents from: samples-develop.zip ➔ samples-develop\HelloBlinky\CS to your desktop. Open "CS\BlinkyIoT.sln" in Visual Studio.

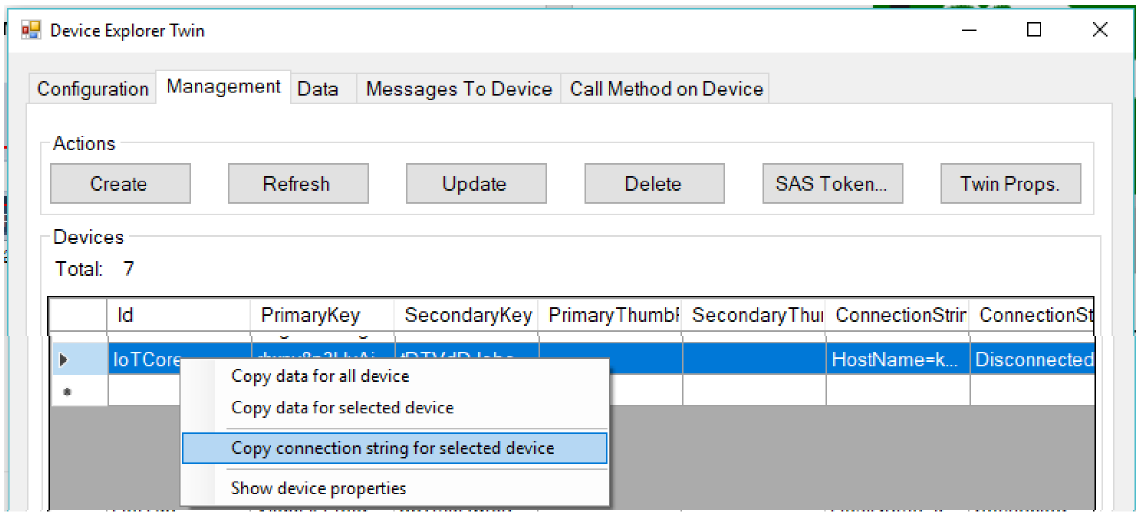
Step 17. Using either the Green, Red or Blue LED with the 560 Ohm resistor, wire the Raspberry Pi as shown below. Note the long lead of the LED must be towards the resistor (else it will not work).



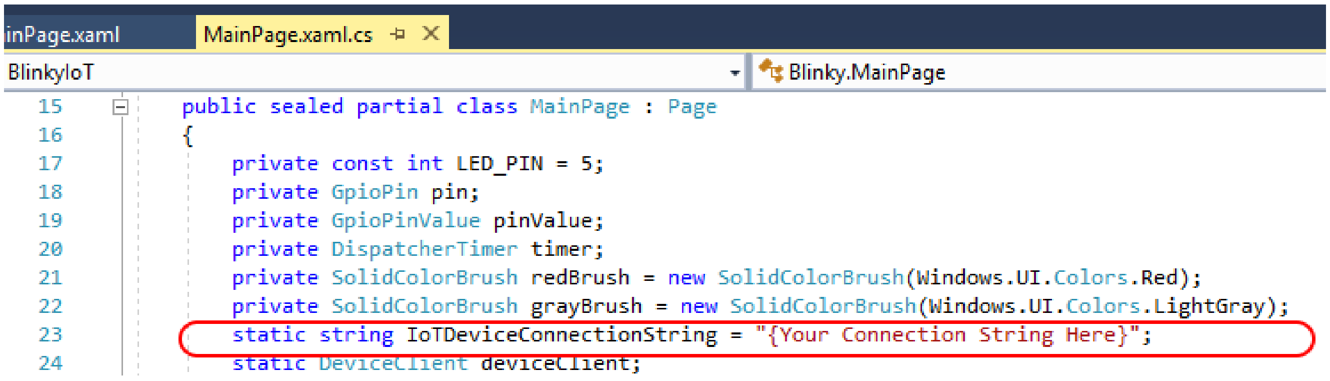
Step 18. With HelloBlinky open in Visual Studio change the platform to ARM click Device → Remote Machine. Select the machine auto detected or manually type in the ip address and Debug/Execute the code. You should see the LED blink every 500 milliseconds. If time allows, review the code to understand how simple it is to interact with the GPIO.

Step 19. Download https://github.com/ksaye/IoTDemonstrations/blob/master/BlinkyIoT/BlinkyIoT.zip?raw=true and unzip the file to your desktop.

Step 20. In IoT Device Explorer, create a new device and copy the connection string, as shown below:

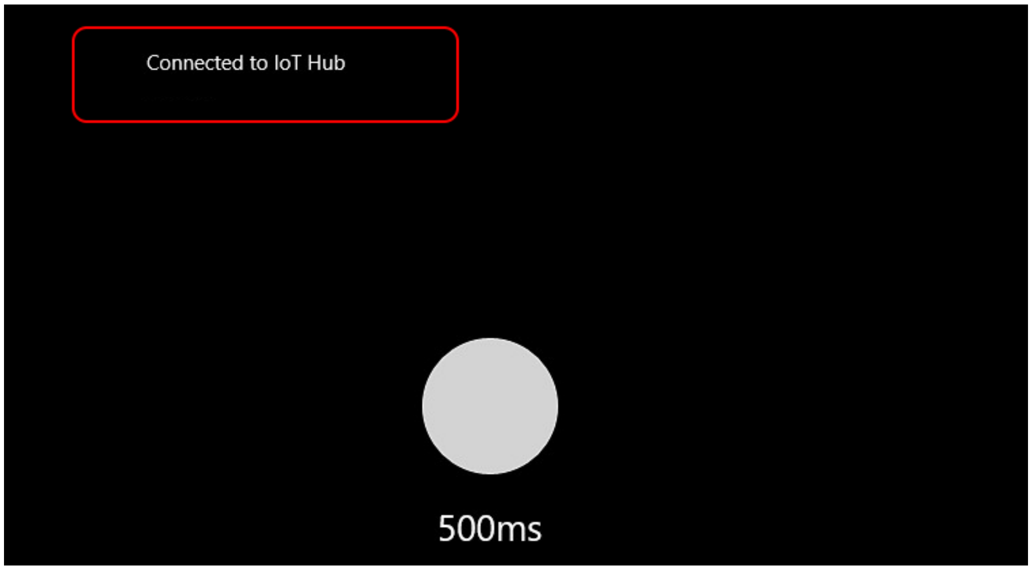


Step 21. Open BlinkyIoT.zip → CS\BlinkyIoT.sln in Visual Studio, past the connection string in the MainPage.xaml.cs on line 26, as shown below:

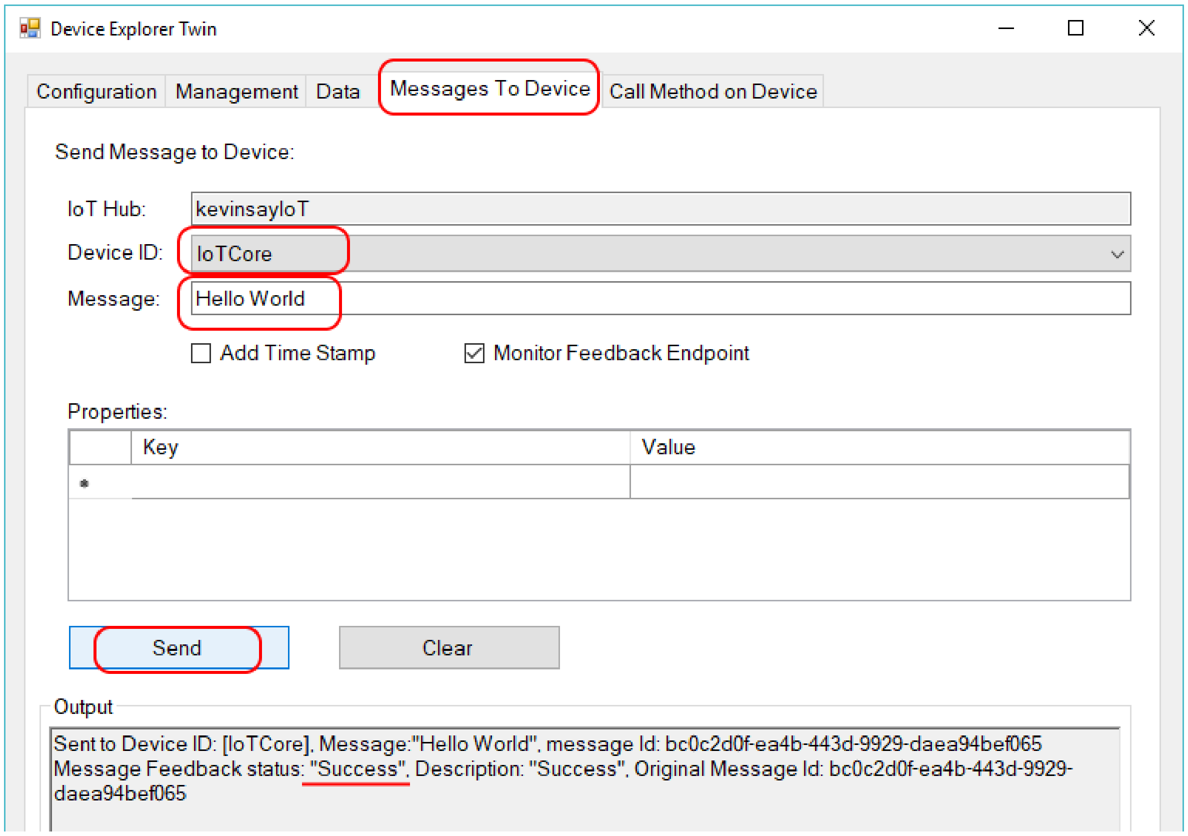


```
public sealed partial class MainPage : Page
{
    private const int LED_PIN = 5;
    private GpioPin pin;
    private GpioPinValue pinValue;
    private DispatcherTimer timer;
    private SolidColorBrush redBrush = new SolidColorBrush(Windows.UI.Colors.Red);
    private SolidColorBrush grayBrush = new SolidColorBrush(Windows.UI.Colors.LightGray);
    static string IoTDeviceConnectionString = "{Your Connection String Here}";
    static DeviceClient deviceClient;
```

Step 22. Leave the platform to ARM click Device → Remote Machine. Select the machine auto detected or manually type in the ip address and Debug/Execute the code. Visual Studio will download the needed NuGet packages. The new status screen should say "Connected to IoT Hub", as shown below:



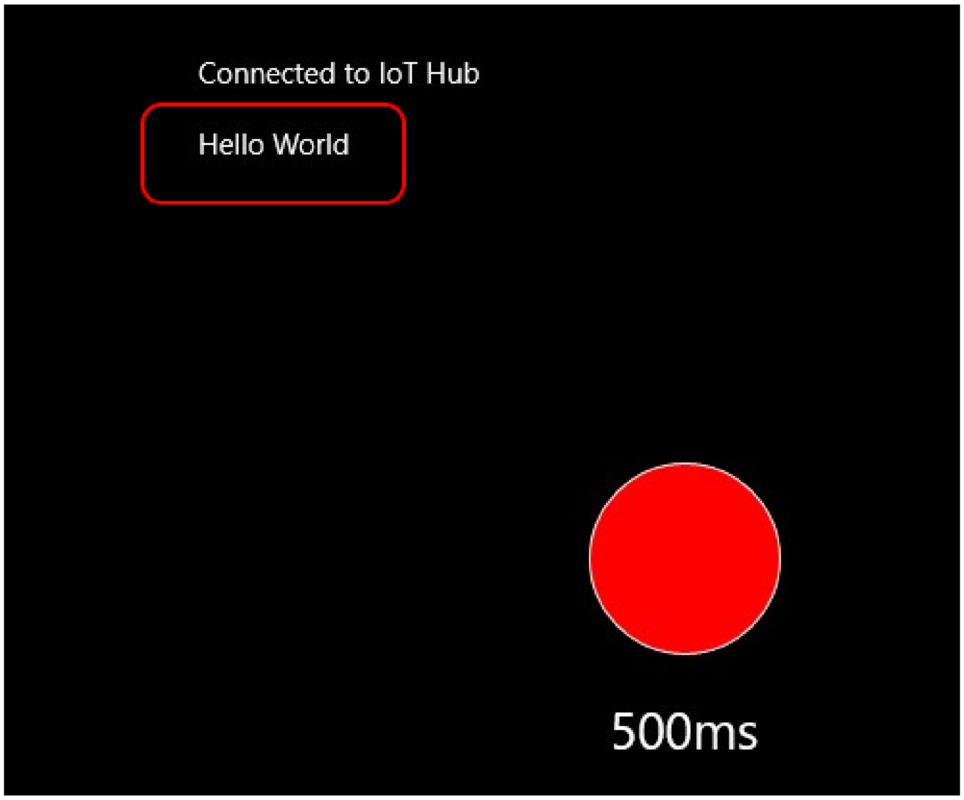
Step 23. Back in Device Explorer, click Message To Device, select the device id, type in Hello World and click send.



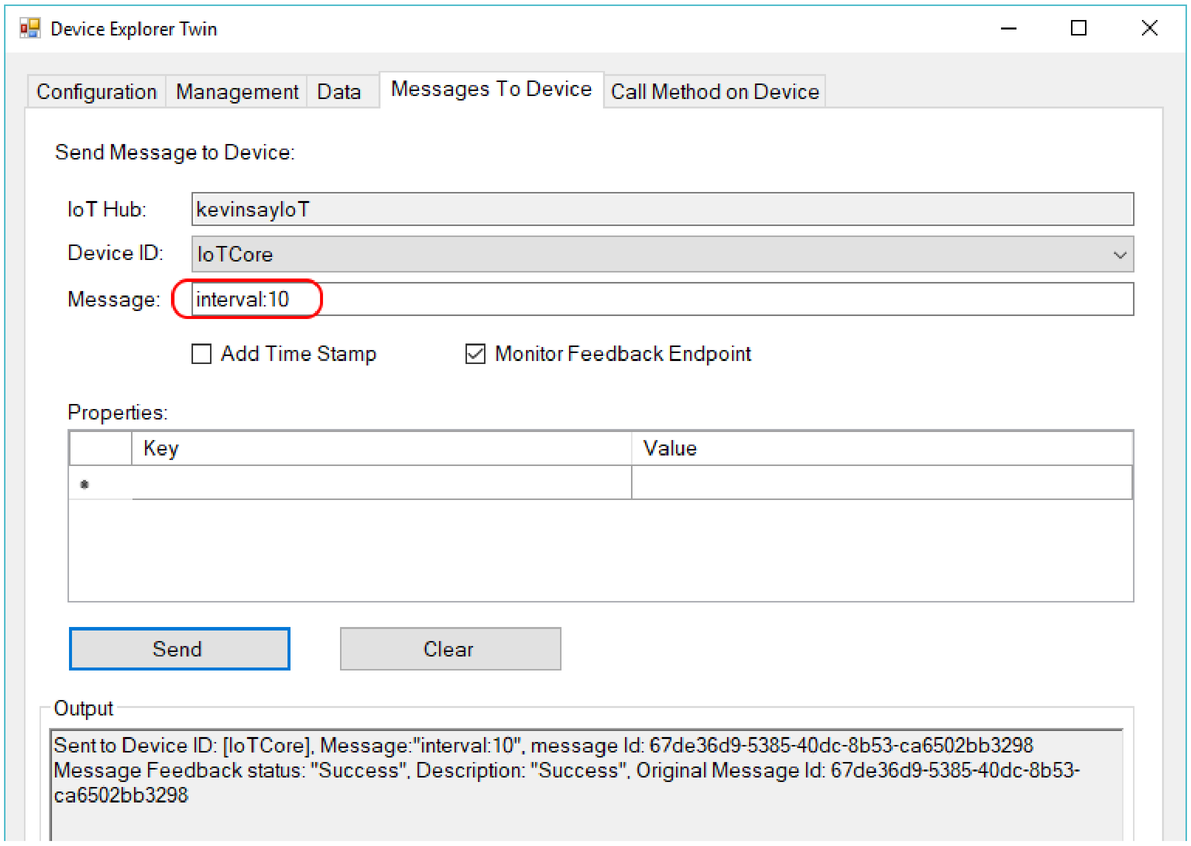
IoT Hub: kevinsayIoT
Device ID: IoTCore
Message: Hello World
☐ Add Time Stamp ☑ Monitor Feedback Endpoint

```
Sent to Device ID: [IoTCore], Message:"Hello World", message Id: bc0c2d0f-ea4b-443d-9929-daea94bef065
Message Feedback status: "Success", Description: "Success", Original Message Id: bc0c2d0f-ea4b-443d-9929-
daea94bef065
```

You should see a screen, similar to below:



Step 24. Back in Device Explorer, click Message To Device, select the device id, type in interval:10 and click send. Note the change in blink frequency on the device and screen.



IoT Hub: kevinsayIoT

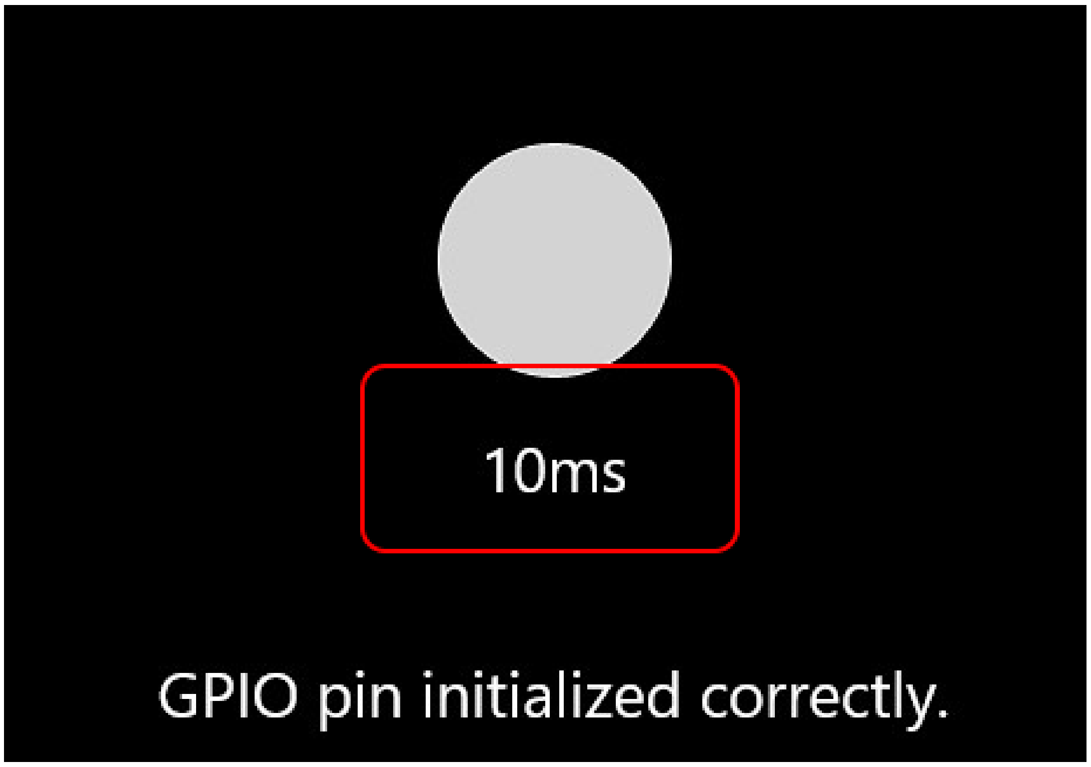
Device ID: IoTCore

Message: interval:10

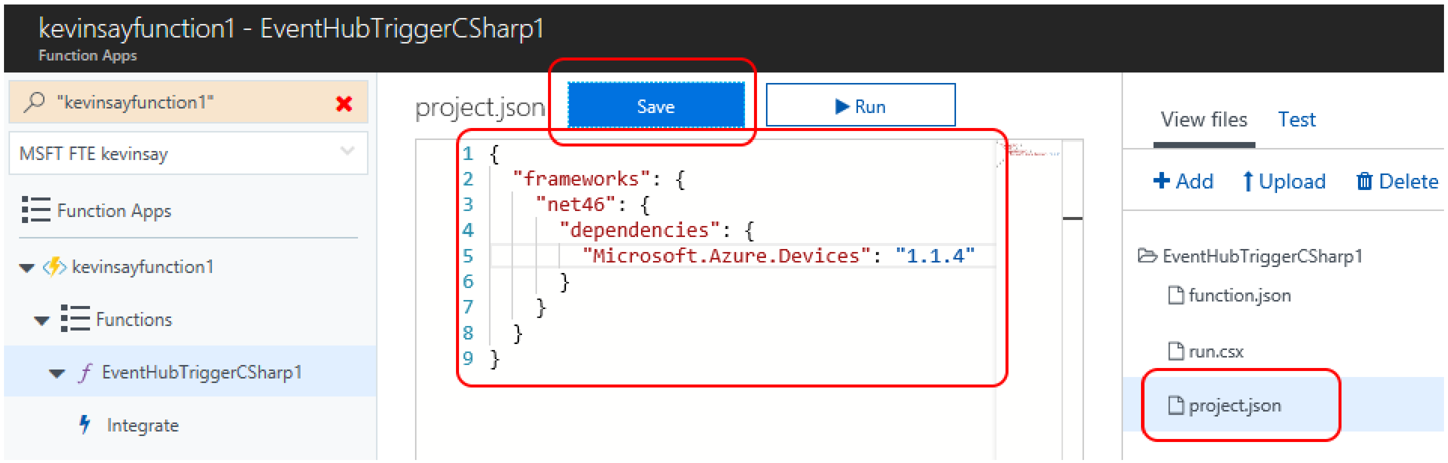
☐ Add Time Stamp ☑ Monitor Feedback Endpoint

```
Sent to Device ID: [IoTCore], Message:"interval:10", message Id: 67de36d9-5385-40dc-8b53-ca6502bb3298
Message Feedback status: "Success", Description: "Success", Original Message Id: 67de36d9-5385-40dc-8b53-
ca6502bb3298
```

You should see a screen, similar to below:

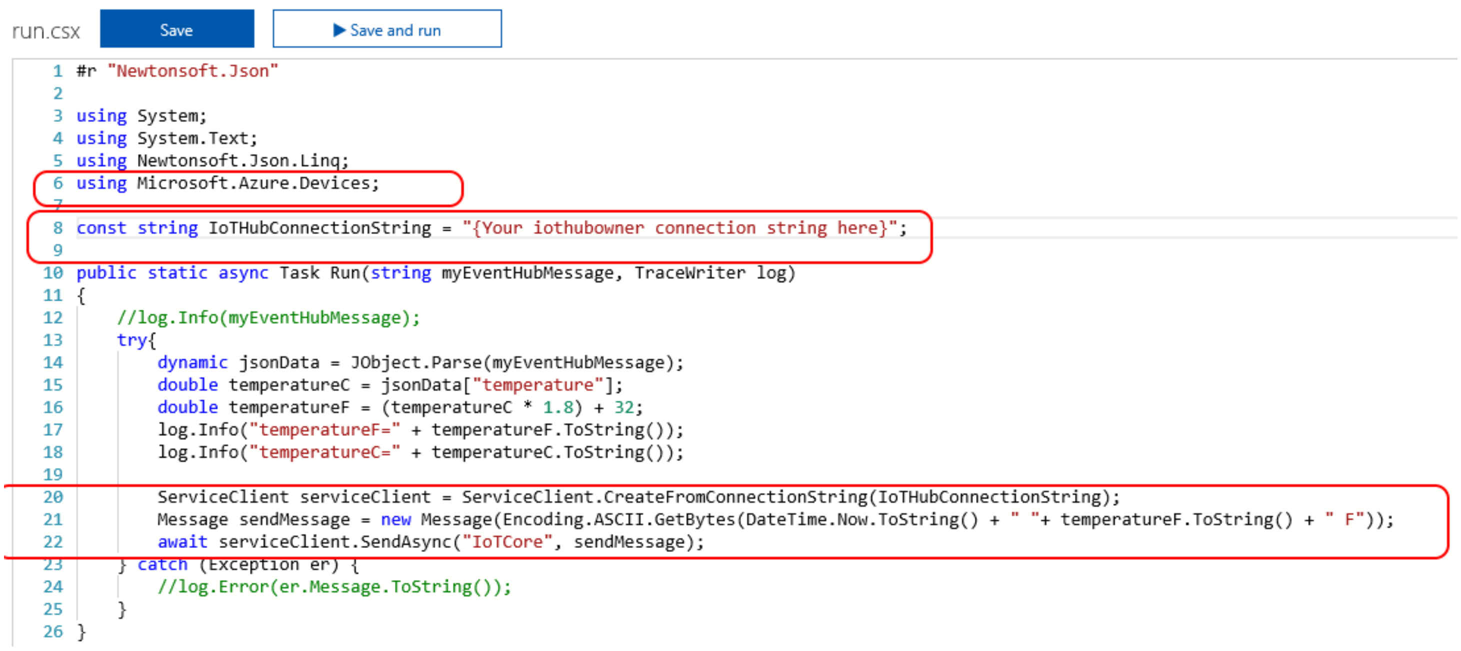


Step 25. (Optional Steps) Stop the Stream Analytics, open your Azure Function and add a file called project.json and add the following content then click save.



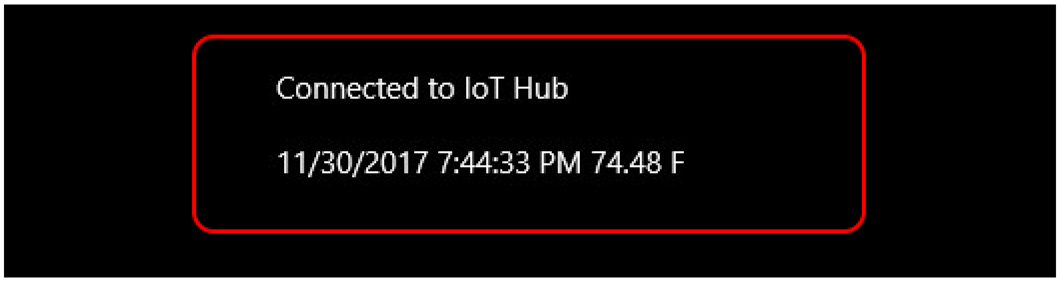
```json
{
  "frameworks": {
    "net46": {
      "dependencies": {
        "Microsoft.Azure.Devices": "1.1.4"
      }
    }
  }
}
```

Step 26. (Optional Steps) In the run.csx file, make the following changes. Correct line 8 and line 21 to reflect your iothubowner connection string and the name of the device id associated with your IoT Core deployment. You can copy and paste from: https://tinyurl.com/IOTVBCFunction



```csharp
1  #r "Newtonsoft.Json"
2
3  using System;
4  using System.Text;
5  using Newtonsoft.Json.Linq;
6  using Microsoft.Azure.Devices;
7
8  const string IoTHubConnectionString = "{Your iothubowner connection string here}";
9
10 public static async Task Run(string myEventHubMessage, TraceWriter log)
11 {
12     //log.Info(myEventHubMessage);
13     try{
14         dynamic jsonData = JObject.Parse(myEventHubMessage);
15         double temperatureC = jsonData["temperature"];
16         double temperatureF = (temperatureC * 1.8) + 32;
17         log.Info("temperatureF=" + temperatureF.ToString());
18         log.Info("temperatureC=" + temperatureC.ToString());
19
20         ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(IoTHubConnectionString);
21         Message sendMessage = new Message(Encoding.ASCII.GetBytes(DateTime.Now.ToString() + " "+ temperatureF.ToString() + " F"));
22         await serviceClient.SendAsync("IoTCore", sendMessage);
23     } catch (Exception er) {
24         //log.Error(er.Message.ToString());
25     }
26 }
```

Now the temperature should show up on your Windows IoT Core deployment screen as shown below. Try to breath on the DHT22 to raise the temperature and watch it change on the screen. Note the speed in which the 2 devices communicate with each other.



```
Connected to IoT Hub

11/30/2017 7:44:33 PM 74.48 F
```

This completes this lab.