CHAPTER 2

Generating Data with Devices

The Internet of Things, or IoT, is about devices that generate and transmit data over the Internet. As mentioned in the last chapter, it is estimated that by 2020 there will be over 25 billion "things" (i.e. devices) connected to the Internet; these devices will range from the washer or oven in your house to the watch you wear or the phone in your pocket. The last chapter also covered several scenarios in which the IoT and devices were implemented, including cars, homes, and animals.

This chapter will simply build on that information and show several examples of the types of devices used today, how to use them to generate data, and how to send that data over the Internet and store it for analysis. The devices that this chapter will use will be the Raspberry Pi board from the Raspberry Pi Foundation (`www.raspberrypi.org/`) and the Tessel board from the Tessel Project (`https://tessel.io/`). There are a large number of devices available that provide very similar functionality, including the Adafruit Feather HUZZAH, the Edison from Intel, the DragonBoard from Arrow, and the boards from Beaglebone, but it would be unrealistic to discuss and demo them all here. Visit `https://azure.microsoft.com/en-us/develop/iot/get-started/` for a full list of IoT boards supported by Microsoft.

Adafruit makes a great Azure IoT starter kit complete with the Feather HUZZAH board, a DHT22 sensor, cables, breadboard, a power cable, and other jumpers and switches. It is available via the Adafruit web site: `www.adafruit.com/products/3032`. This chapter won't cover how to use it but I have blogged about how to get it set up and running with Azure. Visit my blog for more information.

The reason I have chosen the Raspberry Pi is because of its ability to run Windows 10 IoT Core and the familiar development environment of Visual Studio, which makes it easy to code and deploy solutions. I have chosen to also discuss the Tessel board not only because it is red (my favorite color) and looks cool, but to also illustrate Microsoft's support for open source technologies. In all fairness, I am including the Tessel in this chapter simply to show Microsoft's support for a wide range of operating systems, languages, tools, and frameworks.

Now, a couple of things by way of disclaimer. First, as discussed in the introduction, this book is about Microsoft's vision of big data and IoT via the IoT and Cortana Intelligence suite of Azure services, thus data generated will be routed as such. However, for the sake of the examples of this chapter, the data generated from these devices will simply be routed to the screen for output. Chapter 3 will show in detail how to create, configure, and connect the device to an Azure IoT Hub for data storage and processing, and Chapter 4 will then hook up the devices to an Azure IoT Hub. This chapter is all about using devices to generate data.

So, with that, let's get started.

Raspberry Pi

A popular IoT device, the Raspberry Pi is a small but proficient device that does everything a normal computer does by plugging in a monitor, keyboard, mouse, and Ethernet cable. You can browse the Internet, play games, or even run desktop applications such as Microsoft Word or Excel. However, this tiny but powerful mini-pc is really targeted to those who want to explore the world of devices and maker projects. A "maker" is a hardware hacker, someone who likes to build things that make the world a better place.

S. Klein, *IoT Solutions in Microsoft's Azure IoT Suite*, DOI 10.1007/978-1-4842-2143-3_2

In Figure 2-1 you can see two Raspberry Pis. The device on the top is the Raspberry Pi 2 Model B, which has been available since February of 2015. The device on the bottom is the Raspberry Pi 3 Model B, which became available in February of 2016.

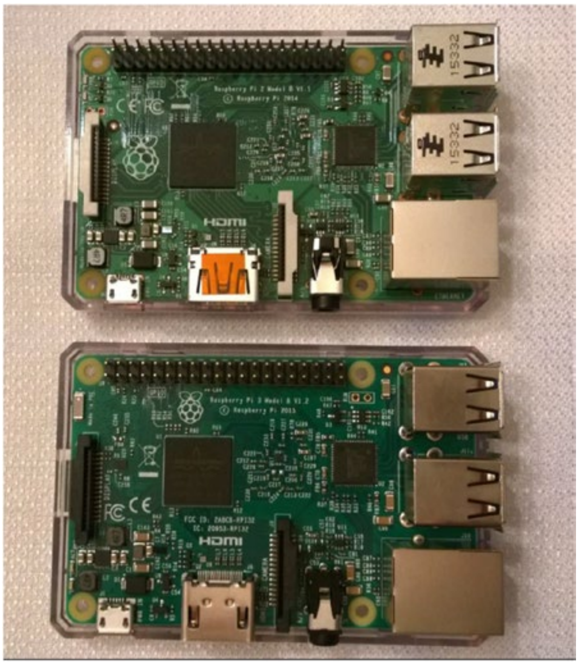


Figure 2-1. *Raspberry Pi 2 (top) and 3 (bottom)*

The Raspberry Pi is about the exact same size as a credit card and both the Pi 2 and Pi 3 vary very little in functionality. Table 2-1 details the main differences between the Pi 2 and the Pi 3.

Table 2-1. *Comparing the Raspberry Pi 2 and 3*

Raspberry Pi 2	Raspberry Pi 3
• 900MHz quad-core ARM Cortex – A7 CPU	• 1.2GHz 64-bit quad-core ARMv8 CPU
• 1GB RAM	• 1GB RAM
	• 802.11n Wireless LAN
	• Bluetooth 4.1
	• Bluetooth Low Energy (BLE)

Both the Pi 2 and the Pi 3 include the following:

- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Camera interface (CSI)
- Display Interface (DSI)
- Micro SD card slot
- Combined 3.5mm audio jack and composite video

Honestly, having the latest and greatest is cool, but if you don't plan on doing any Bluetooth or Wi-Fi, the Pi 2 is perfect for getting started. However, you can't argue $35, so feel free to spring for the Pi 3 if you are so inclined. If you already have a Pi 2 and are looking at the Pi 3, the Pi 3 is identical in form factor so any case you have will work for the Pi 3. Plus, the Pi 3 is completely compatible with the Pi 2.

With the Raspberry Pi in hand, it is time to start setting up and coding. The next section will walk through the setup and configuration of the Raspberry Pi.

Getting Started

Hopefully you noticed on the list of items *not* included was a hard drive. If there is no hard drive, how does it boot up and work? Great question! In order to get the Pi to work, you need a MicroSD card, shown in Figure 2-2.

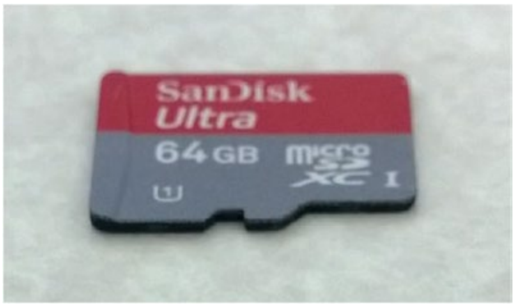


***Figure 2-2.** MicrosSD card*

You can pick up a MicroSD card anywhere. The one in Figure 2-2 is 64GB but you really don't need one that big. An 8GB or 16GB one will do just fine, and you can find them online quite easily. I ordered mine from Amazon:

`www.amazon.com/SanDisk-Ultra-Micro-SDHC-16GB/dp/9966573445?ie=UTF8&psc=1&redirect=true&ref_=oh_aui_detailpage_o01_s00`

Simply slide the card into the MicroSD card slot on the back of the Pi, as shown in Figure 2-3.

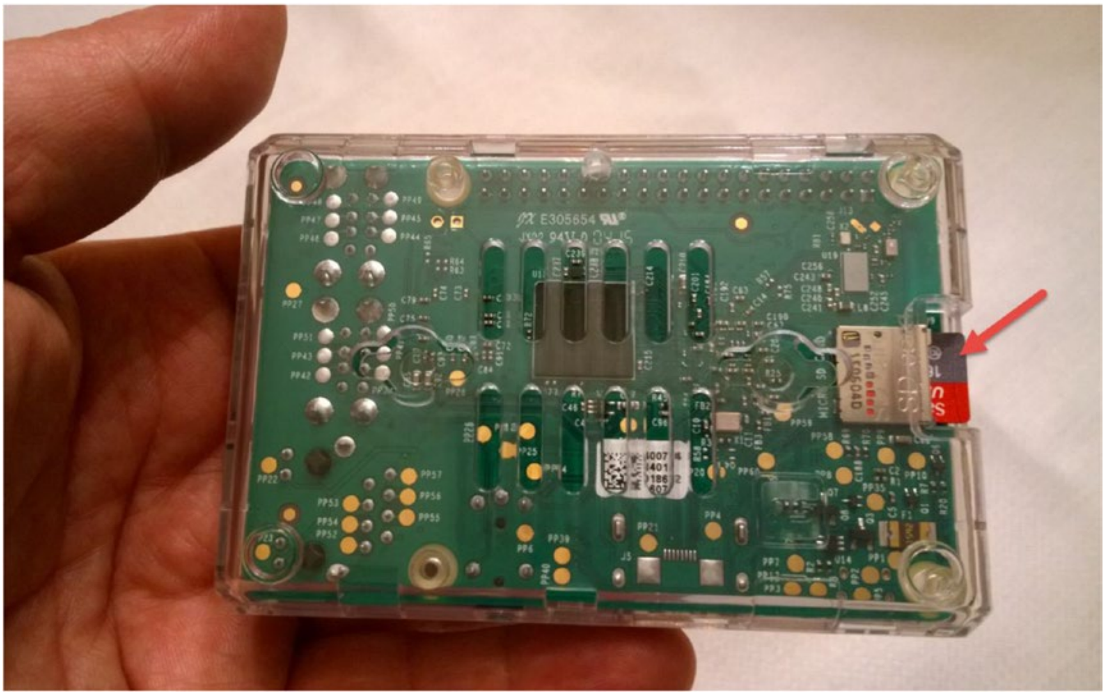


Figure 2-3. *Inserting the MicroSD card into the Raspberry Pi*

However, there is nothing on the MicroSD card, thus to get started with the Pi and for it to be useful, you need to install an operating system on it. For the Raspberry Pi, you'll be putting Windows 10 IoT core on it. You'll need a computer with an SD or MicroSD card slot, so slide your new card into the slot.

The Pis in the figures are shown in a case. Depending on where you order your Pi from, it may not come with a case so please take precautions with anti-static bags or gloves to avoid blowing the electronics on the board if you do not have a case. I ordered these clear cases from Amazon:

`www.amazon.com/gp/product/B00MQLB1N6/ref=oh_aui_detailpage_o05_s00?ie=UTF8&psc=1`

It is best practice to format your SD card before installing any OS on it, and there is an easy tool which does that. The following website has a nifty utility which formats all sorts of memory cards:

`www.sdcard.org/downloads/formatter_4/`

Scroll down towards the bottom of the page and click the blue "Download SD Formatter for Windows" link. On the EULA page, scroll down and click Accept and then the download will begin. Unzip the Setup file and install the formatter. Once installed, run the formatter. It will find your SD card. By default the Quick Format is selected, which is fine. Click Format. Within a matter of seconds your SD card will be formatted.

Installing Windows IoT Core

There are several ways to do this, depending on if you have a Pi 2 or Pi 3. I'll begin with the Pi 2.

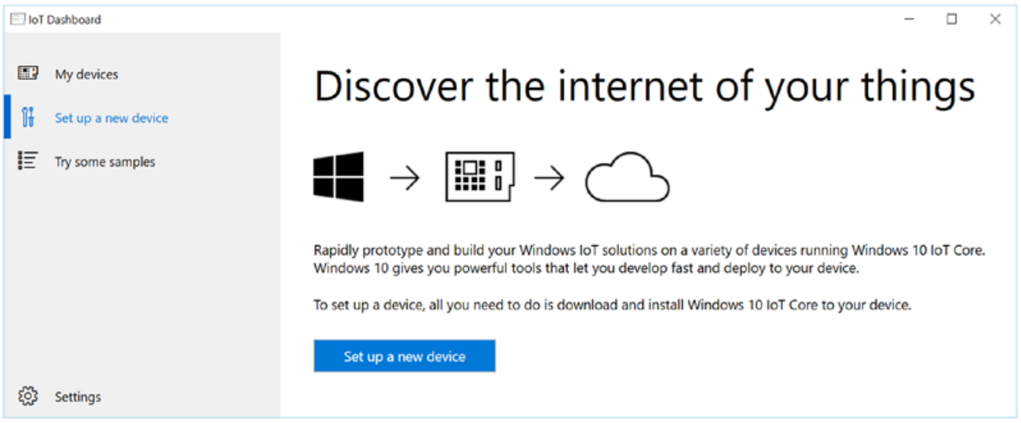
Pi 2

The easiest way to install Windows 10 IoT Core on your Raspberry Pi 2 is by installing the Microsoft IoT Core Dashboard, which you can download from here:

`http://ms-iot.github.io/content/en-US/Downloads.htm`

Once on the Downloads and Tools page, simply click the big blue "Get IoT Core Dashboard" button. It is a small download, so it doesn't take that long. Once downloaded, run the setup program. The install itself is also quick, and once done the IoT Dashboard will automatically launch.

On the left of the IoT Dashboard the "Set up a new device" option should automatically be selected, so click the blue "Set up a new device" button on the main page. See Figure 2-4.



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.

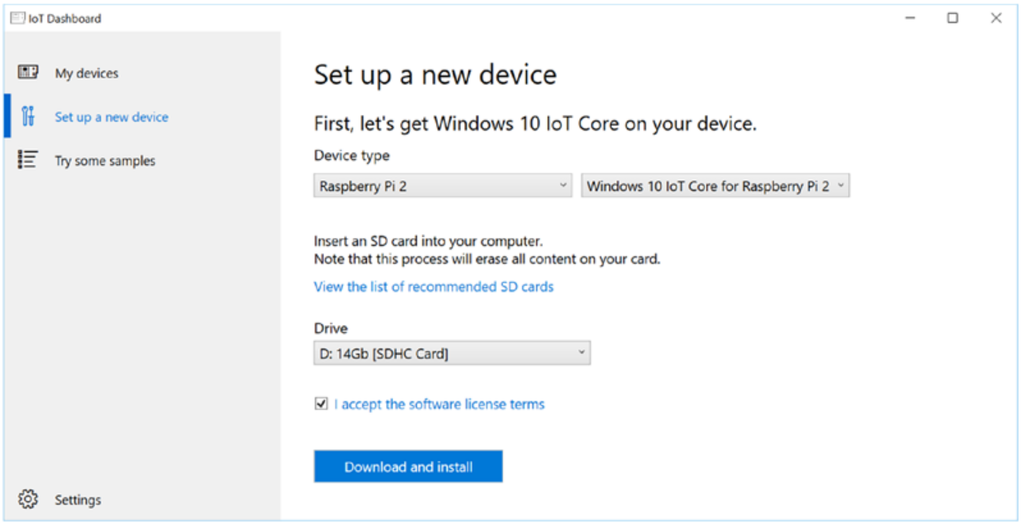
To set up a device, all you need to do is download and install Windows 10 IoT Core to your device.

Figure 2-4. *Microsoft IoT Core Dashboard*

On the "Set up a new device" page, make sure the device type is set to Raspberry Pi 2 and that Windows 10 IoT Core for Raspberry Pi is selected. At the time of this writing, Windows 10 IoT Core for Raspberry Pi 2 is the only option but this may change to include Windows 10 IoT Core for Raspberry Pi 3 in the future.

However, clicking the drop-down for device type shows four options: Raspberry Pi, Minnowboard Max, Qualcomm Dragonboard, and Custom, which shows the breadth of devices that Microsoft and Windows 10 IoT Core supports.

Check the "I accept the software license terms" button and click the "Download and install" button, as shown in Figure 2-5.


IoT Dashboard

My devices
Set up a new device
Try some samples
Settings

Set up a new device

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

Device type

Raspberry Pi 2 | Windows 10 IoT Core for Raspberry Pi 2

Insert an SD card into your computer.
Note that this process will erase all content on your card.

View the list of recommended SD cards

Drive

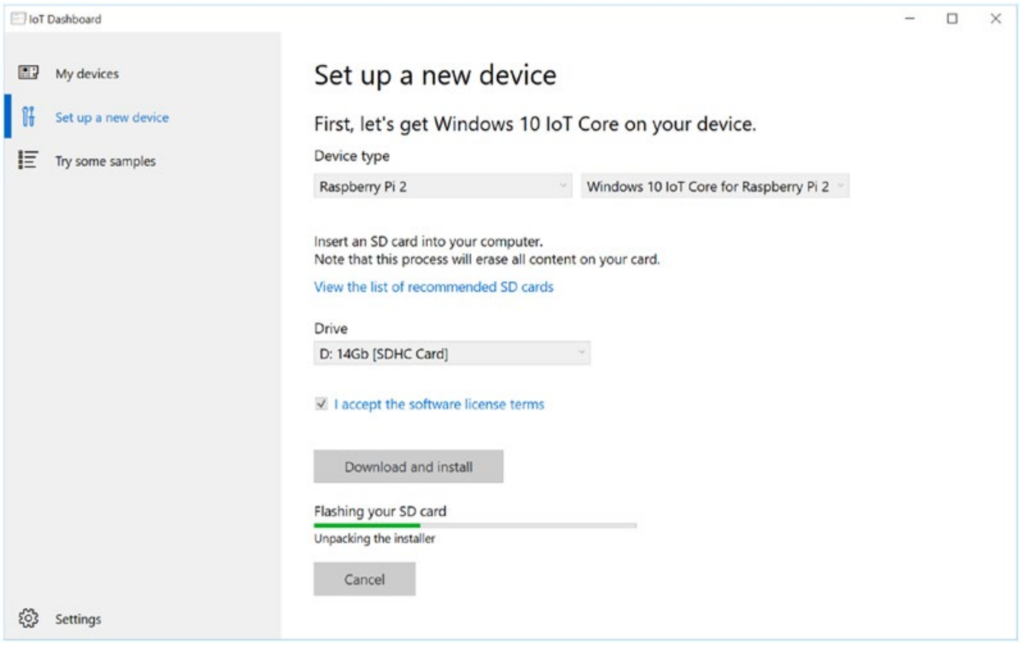
D: 14Gb [SDHC Card]

☑ I accept the software license terms

Download and install

Figure 2-5. *Setting up Windows 10 IoT Core on a Raspberry Pi 2*

You might get a pop-up encouraging you to back up any files on the card before proceeding. Click OK. The Dashboard will flash your device, and then download and install Windows 10 IoT Core to the SD card. See Figure 2-6.


IoT Dashboard

My devices
Set up a new device
Try some samples
Settings

Set up a new device

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

Device type

Raspberry Pi 2 | Windows 10 IoT Core for Raspberry Pi 2

Insert an SD card into your computer.
Note that this process will erase all content on your card.

View the list of recommended SD cards

Drive

D: 14Gb [SDHC Card]

☑ I accept the software license terms

Download and install

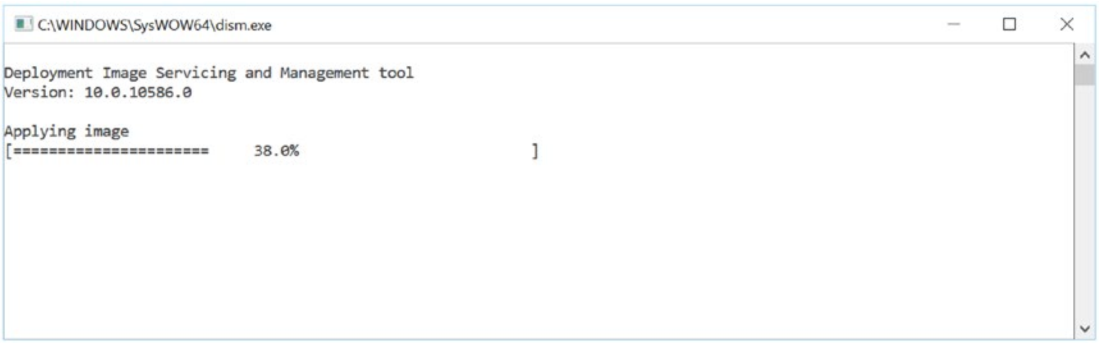
Flashing your SD card

Unpacking the installer

Cancel

Figure 2-6. *Installing Windows IoT Core for Raspberry Pi 2*

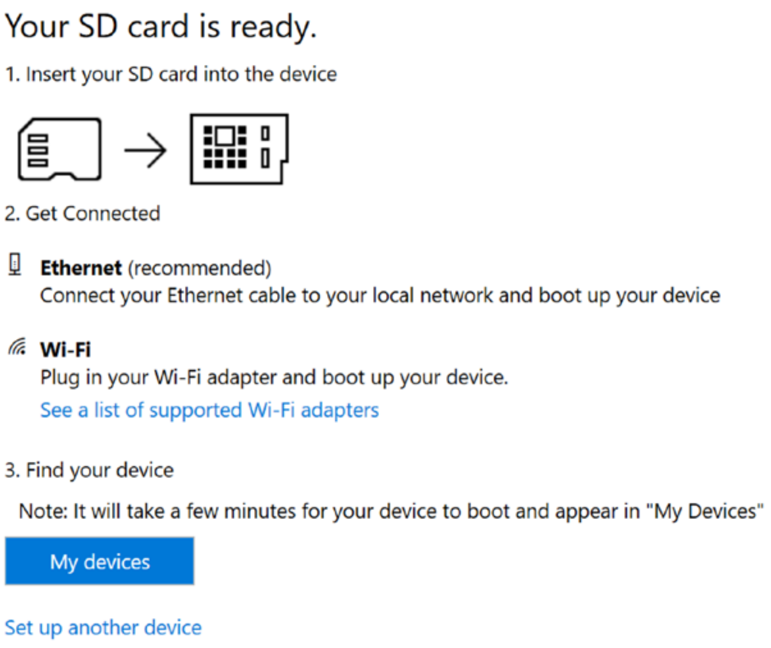
Depending on your Internet connection this might take a few minutes, so be patient (see Figure 2-7).



```
Deployment Image Servicing and Management tool
Version: 10.0.10586.0

Applying image
[======================      38.0%                    ]
```

Figure 2-7. *Installing Windows 10 IoT Core on the MicroSD Card*

When the process is completed, your SD card will be ready, as shown in Figure 2-8. Remove it from your computer and insert it into the MicroSD slot in the Raspberry Pi, as shown in Figure 2-3 above. If you want to set up another device, click the “Set up another device” button; otherwise, close the IoT Dashboard.



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"

Figure 2-8. *Completed process*

At this point you are ready to connect and boot up your Raspberry Pi.
See the next section for the Pi 3 setup.

Pi 3

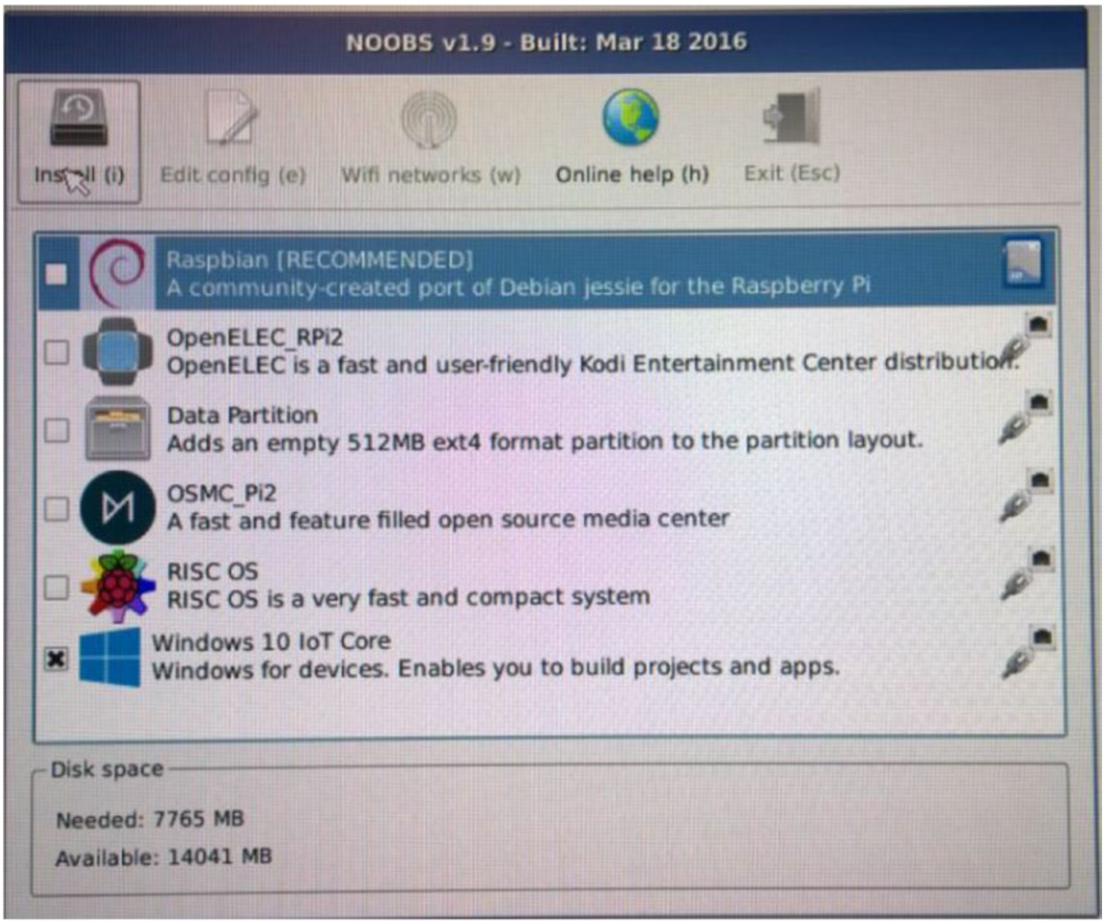
By the time you read this, you will probably be able to install Windows 10 IoT Core via the Dashboard. However, this section will walk you through another option because currently it is the only way to install Windows 10 IoT Core on a Pi 3. Plus, this other option provides a good look at the Raspberry Pi NOOBS (New Out-of-the-Box Software).

The first thing to do is to go to the Raspberry Pi website and download NOOBS:

www.raspberrypi.org/help/noobs-setup/

Click the Download Zip link for the Offline and Network Install. The current version of NOOBS as of this writing is 1.9. Extract the contents of the zip file to a folder, and then drag and drop all the contents onto your MicroSD card. When the copy is finished, safely remove the SD card and insert it into your Raspberry Pi.

Plug in your keyboard, mouse, video, and Ethernet, and then power up your Raspberry Pi by either plugging it in to your PC (via the USB-to-micro-USB cable) or via the power adapter. When the Pi boots up, you will be presented with the screen shown in Figure 2-9. This screen shows you all of the operating systems supported by the Raspberry Pi and which you can install on the SD card. Select the Windows 10 IoT Core option.



NOOBS v1.9 - Built: Mar 18 2016

[ ] Raspbian [RECOMMENDED]
A community-created port of Debian jessie for the Raspberry Pi

[ ] OpenELEC_RPi2
OpenELEC is a fast and user-friendly Kodi Entertainment Center distribution.

[ ] Data Partition
Adds an empty 512MB ext4 format partition to the partition layout.

[ ] OSMC_Pi2
A fast and feature filled open source media center

[ ] RISC OS
RISC OS is a very fast and compact system

[✓] Windows 10 IoT Core
Windows for devices. Enables you to build projects and apps.

Disk space
Needed: 7765 MB
Available: 14041 MB

Figure 2-9. *Running NOOBS to Install Windows 10 IoT Core for Raspberry Pi 3*

Click Install. You will be prompted with a warning which states that the selected operating system will be installed and all existing data on the SD card will be overwritten, at which point NOOBS will begin by preparing the operating system partition for use on the SD card and the process for Windows 10 IoT Core will be started.

If this is the first time you are going through this process, you will be asked to provide your Microsoft account login credentials and join the Microsoft Insider Program. Once logged in, you will see a screen asking you to select the Windows 10 IoT core edition. You might see two editions listed:

- Windows 10 IoT Core
- Windows 10 IoT Core Insider Preview

If installing on Pi 2, select Windows 10 IoT Core. If installing on Pi 3, select Windows 10 IoT Core Insider Preview.

Confirm the device you are installing the OS on (Raspberry Pi) and click Confirm, and then click the Download Now button. You will need to accept the license agreement, so click Yes. The OS will then be downloaded and installed on your SD card. Depending on your Internet connection, this might take several minutes. Once downloaded, you will see a confirmation dialog. Click OK to reboot your device. Once the device reboots, you will see the Windows boot screen followed by a Device Info page.

Now you are ready to start putting the device to work and generate some data.

Generating Data

All the devices, including the Raspberry Pi, can be used in many different environments and scenarios. This example will use a simple temperature and humidity sensor to generate data. For this example, I am using both a DHT11 sensor and a DHT22 sensor. You will also need a breadboard and jumper wires. I ordered a sensor kit from SunFounder that contains a total of 37 modules, including the DHT11 sensor, plus the breadboard and jumper wires. I got mine from here:

www.amazon.com/SunFounder-Sensor-Kit-Raspberry-Extension/dp/B00P66XRNK?ie=UTF8&psc=1&redirect=true&ref_=oh_aui_detailpage_o04_s00

You can also order a standalone DHT22 temperature and humidity sensor, and I ordered mine from PrivateEyePi at http://ha.privateeyepi.com/store/index.php?route=product/product&product_id=84&search=dht22.

With all of that in hand, I thus wired up my Raspberry Pi 3 and sensor as shown in Figure 2-10.

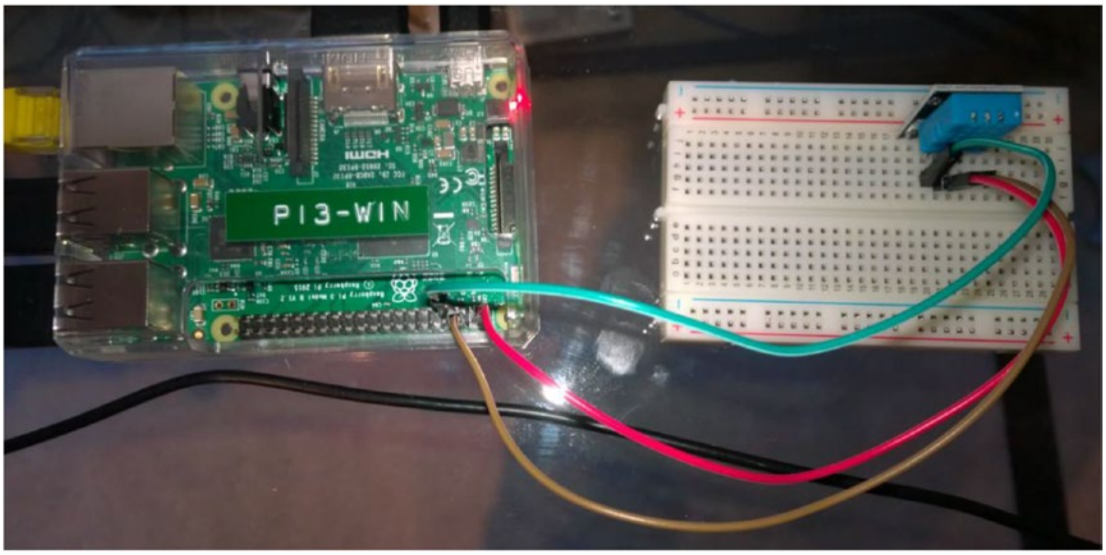


Figure 2-10. *Wiring up the DHT11 sensor to the Raspberry Pi*

To wire it up correctly, I followed the instructions included with the SunFounder kit for the DHT11 sensor as well as the appropriate pin mappings for the Raspberry Pi here:

`https://ms-iot.github.io/content/en-US/win10/samples/PinMappingsRPi2.htm`

Per the instructions for the DHT11 sensor that came with the SunFounder kit, the right pin is ground, the middle pin is power, and the left pin is Signal, or data. Thus, following the pin mappings from the link above and the instructions from SunFounder kit, the brown wire (my ground wire) goes from the right pin on the sensor to pin 6 on the Raspberry Pi. The red wire, my power wire, goes from the middle pin on the sensor to pin 2 (5 volt power) on the Pi, and the green wire, my data wire, goes from the left pin on the sensor to pin 4 (GPIO) on the Pi.

Now it's time to write the code to tell the Pi and the sensor to do their thing. Do the following:

1. Open Visual Studio 2015 and go to **File ➤ New ➤ Project**. From the Templates select **Windows ➤ Universal ➤ Blank app (Universal Windows)**. Select a location to save the project to and click the **Ok** button.

2. Download the source code for the DHT sensor from `https://github.com/porrey/dht/tree/master/source/Windows%2010%20IoT%20Core/DHT%20Solution/Sensors.Dht` and add the `Sensors.DHT` project to your solution.

3. Optionally, the DHT libraries were recently added to Nuget.org as a Nuget package. Thus, you can add the DHT sensor libraries by running command *Install-Package Dht* in the Package Manager Console window.

4. Add a reference to the Sensors.DHT project to the project you just created.

5. Next, add a reference to your project for Windows 10 IoT Extensions for UWP. To do this, right-click the references node in the Solution Explorer and select Add. From the left-hand tree, select **Universal Windows ➤ Extensions** and then select **Windows 10 IoT Extensions for UWP** and click the **Ok** button.

6. Next, using Nuget, add the **Microsoft.Azure.Devices.Client** package to your project.

7. At the top of your `MainPage.xaml.cs` file, add the following imports:

```
using Sensors.Dht;
using Microsoft.Azure.Devices.Client;
using Windows.Devices.Gpio;
using System.Text;
```

8. Add the following constants and variables:

```
private const int DHTPIN = 4;
private IDht dht = null;
private GpioPin dhtPin = null;
private DispatcherTimer sensorTimer = new DispatcherTimer();
```

9. In the `MainPage()` constructor add the following code:

```
dhtPin = GpioController.GetDefault().OpenPin(DHTPIN, GpioSharingMode.Exclusive);
dht = new Dth11(dhtPin, GpioPinDriveMode.Input);
sensorTimer.Interval = TimeSpan.FromSeconds(1);
sensorTimer.Tick += sensorTimer_Tick;
sensorTimer.Start();
```

10. In the class add the following methods:

```
private void sensorTimer_Tick(object sender, object e)
{
             readSensor();
}
private async void readSensor()
{
    DhtReading reading = await dht.GetReadingAsync().AsTask();
    if (reading.IsValid)
    {

        double temp = ConvertTemp.ConvertCelsiusToFahrenheit(reading.Temperature);
        string message = "{\"temperature\" :" + temp.ToString() + ", \"humidity\":" +
        reading.Humidity.ToString() + "}";
        listBox.Items.Add(message.ToString());
    }
}
```

11. Next, add the following class:

```
static class ConvertTemp
{
    public static double ConvertCelsiusToFahrenheit(double c)
    {
        return ((9.0 / 5.0) * c) + 32;
    }
}
```

12. Now open the MainPage and add a ListBox to the form. This is simply to visual see that data is being returned when deployed to the Pi.

13. Next, you need to deploy the solution to your Raspberry Pi. In Visual Studio, from the tool bar, set the solution platform to **ARM** and the target as **Remote Machine,** as shown in Figure 2-11.

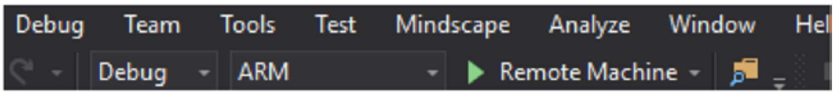
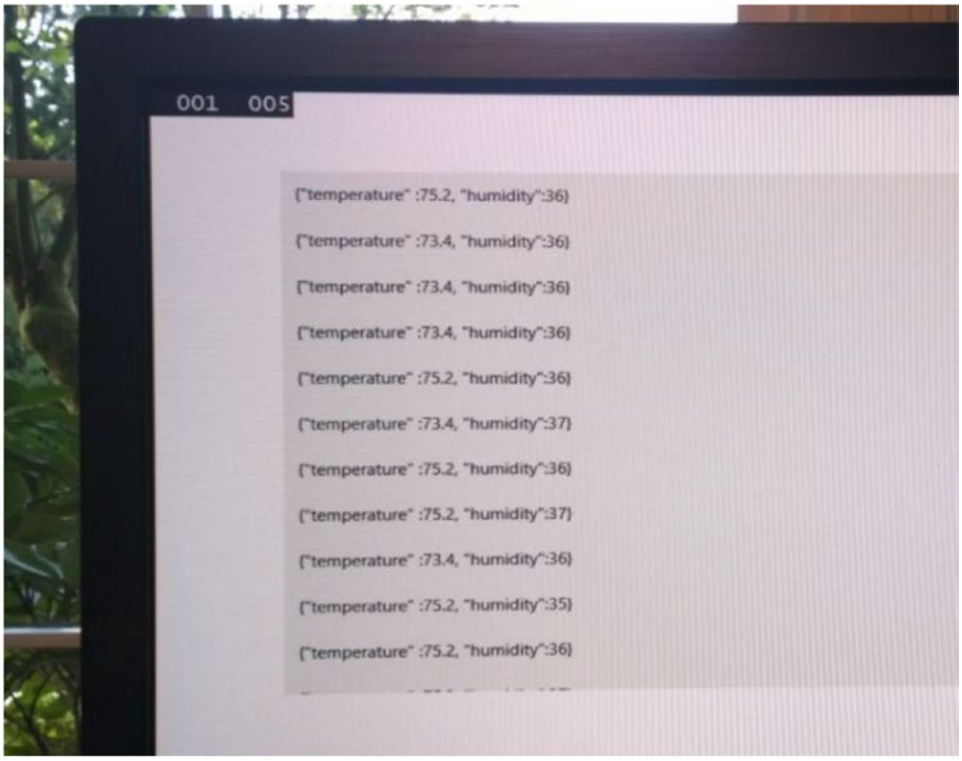


Figure 2-11. *Running the application to publish to the Raspberry Pi*

14. Open the project properties by double-clicking the **Properties** node in Solution Explorer and navigate to the **debug** settings.

15. In the **Remote machine:** textbox you need to enter the IP address/name of your Raspberry Pi. To do this, click the Find button and wait a few moments for Visual Studio to search for your devices. Your Raspberry Pi should show in the Auto Detected section. Click your Raspberry Pi and click the **Select** button. If your device doesn't show, it should be called **minwinpc** so enter this value.

16. Press the **green play button** next Remote Machine or press **F5** to build and deploy the solution to your Raspberry Pi. Your device should now start sending temperature and humidity readings, as shown in Figure 2-12.



```
{"temperature" :75.2, "humidity":36}
{"temperature" :73.4, "humidity":36}
{"temperature" :73.4, "humidity":36}
{"temperature" :73.4, "humidity":36}
{"temperature" :75.2, "humidity":36}
{"temperature" :73.4, "humidity":37}
{"temperature" :75.2, "humidity":36}
{"temperature" :75.2, "humidity":37}
{"temperature" :73.4, "humidity":36}
{"temperature" :75.2, "humidity":35}
{"temperature" :75.2, "humidity":36}
```

Figure 2-12. *Temperature and humidity results*

By default, the DHT11 sensor returns the temperature in Celsius. Thus, the method to convert it to Fahrenheit. If you are ok with Celsius, comment out the two lines directly above the `listbox.Items.Add` call, and then uncomment out the first `string message` line.

If you are using the DHT11 sensor, the code above will also work. All you need to do is change the following line to reference the DHT22 instead of the DHT11:

```
dht = new Dht22(dhtPin, GpioPinDriveMode.Input);
```

I'll also point out the line of code used above:

```
private const int DHTPIN = 4;
```

Notice that the DHT pin is set to 4. As mentioned above, the green wire, my data wire, goes from the left pin on the sensor to pin 4 (GPIO) on the Pi so I am specifying which pin to get the data from.

At this point in the example, installing the Nuget package `Microsoft.Azure.Devices.Client` wasn't necessary but it will come in handy in an upcoming chapter with this sample so let's get it out of the way. This package provides device support for sending messages to an Azure IoT Hub, which will be discussed in the next chapter.

To get the temperature to fluctuate to simulate a malfunction (either the sensor going bad or the actual temperature not remaining consistent), I took a hair dryer and an air canister and rotated blowing air on the sensor to get the temperature to go up and down. Now, truth be told, for this example you didn't need a breadboard. To make things simpler, you could have simply connected the sensor via the wires directly to the Raspberry Pi using the same pin configuration.

For more information on the Raspberry Pi, visit `www.raspberrypi.org/`.

Before moving on to the Tessel, the next section will discuss a device that is branded as the "ultimate topping for your Pi!"

FEZ HAT

The previous section used a breadboard, wires, and a DHT22 sensor to generate data via the Raspberry Pi. This meant there was a need to understand the pin layout on the devices to wire up the sensor correctly. This comes in handy if you need to wire up multiple sensors, but there is an easier way via the FEZ HAT, shown in Figure 2-13.

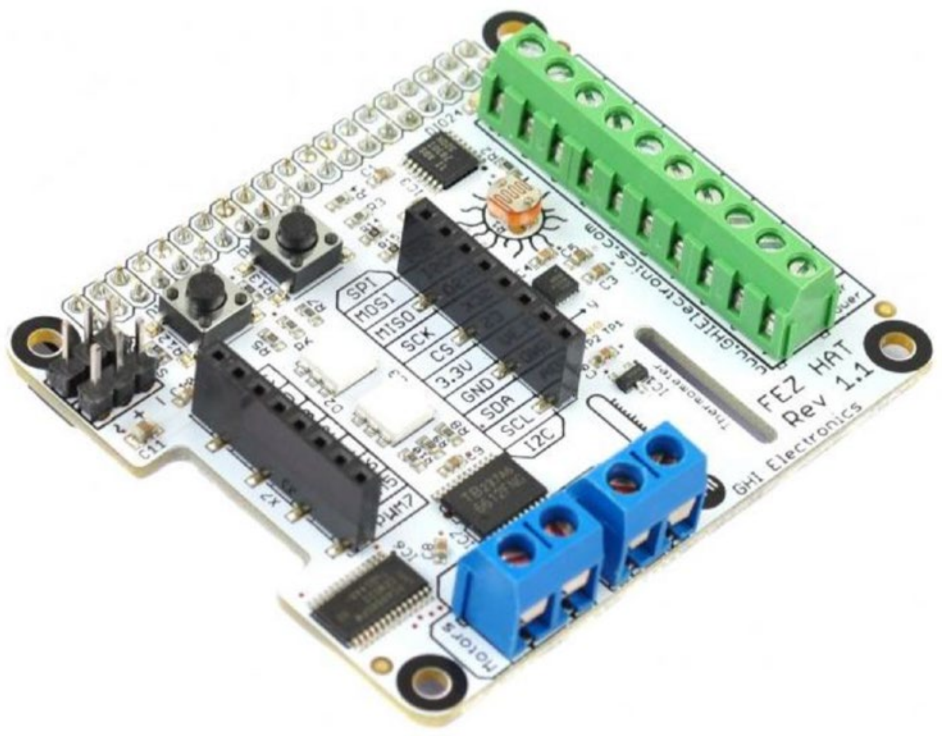


Figure 2-13. *The FEZ HAT*

The FEZ (Fast and Easy) HAT is a simple device that simply sits on top of the Raspberry Pi, as shown in Figure 2-14, and includes a wide variety of features:

- Temperature Sensor
- Light Sensor
- Accelerometer
- Two Server Motor Connections

For a complete list of features and functionality, visit `www.ghielectronics.com/catalog/product/500`.

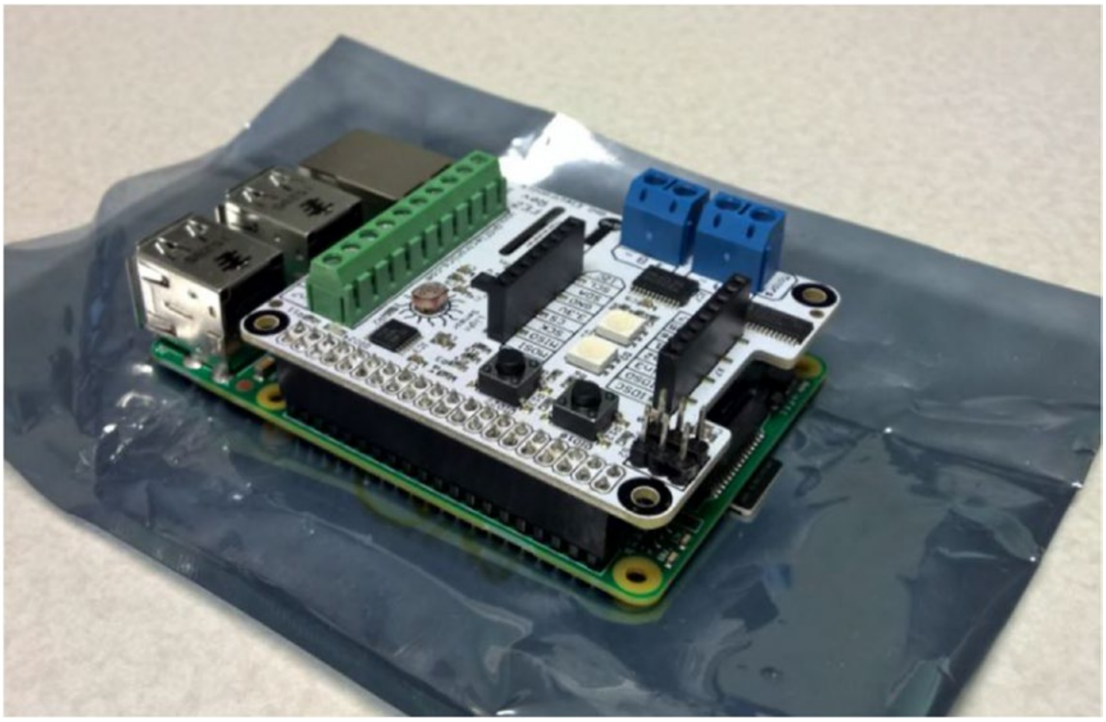


Figure 2-14. *FEZ HAT with a Raspberry Pi*

Using the FEZ HAT is quite simple. The first step is to install the drivers from Nuget. While the code will be very similar, go ahead and create a new blank UWP app and run the following commands in the Package Manager Console to install the appropriate FEZ HAT drivers:

```
Install-Package GHIElectronics.UWP.Shields.FEZHAT
```

Once the package is installed, add the following statement to the MainPage form in the `using` section:

```
using GHIElectronics.UWP.Shields;
using Newtonsoft.Json;
```

Before you start adding code, open up the MainPage and add a ListBox to the form, similar to what you did in the previous section. And, just like you did in the previous example, you will need to add a reference to your project for Windows 10 IoT Extensions for UWP. To do this, right-click the references node in the Solution Explorer and select Add. From the left-hand tree, select **Universal Windows ➤ Extensions** and then select **Windows 10 IoT Extensions for UWP** and click the **Ok** button.

Next, in the code behind the `MainPage`, add the following statements above the `MainPage()` constructor:

```
private FEZHAT hat;
private DispatcherTimer timer;
private bool next;
```

Add the following to the `MainPage()` constructor:

```
this.SetupHat();
```

Next, add the `SetupHat()` method:

```
private async void SetupHat()
{
    this.hat = await FEZHAT.CreateAsync();
    this.timer = new DispatcherTimer();
    this.timer.Interval = TimeSpan.FromSeconds(1);
    this.timer.Tick += this.Timer_Tick;
    this.timer.Start();
}
```

Next, add the `Time_Tick()` method:

```
private void Timer_Tick(object sender, object e)
{
    readSensor();
}
```

Lastly, add the `readSensor()` method:

```
private async void readSensor()
{

    var light = this.hat.GetLightLevel();

    // Temperature Sensor
    var temp = this.hat.GetTemperature();

    double ftemp = ConvertTemp.ConvertCelsiusToFahrenheit(temp);

    System.Diagnostics.Debug.WriteLine("Temperature: {0} °C, Light {1}", ftemp.
ToString("N2"), light.ToString("N2"));

    string message = "{\"temperature\" :" + temp.ToString() + "}";

    //send to listbox
    listBox.Items.Add(message);
}
```

Similarly, add the `ConvertTemp` class so that the temperature returned from the FEZ HAT can be converted to Fahrenheit.

The entire code listing is below for your reference:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using GHIElectronics.UWP.Shields;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=402
352&clcid=0x409

namespace App1
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        private FEZHAT hat;
        private DispatcherTimer timer;
        private bool next;

        public MainPage()
        {
            this.InitializeComponent();
            this.SetupHat();
        }

        private async void SetupHat()
        {
            this.hat = await FEZHAT.CreateAsync();
            this.timer = new DispatcherTimer();
            this.timer.Interval = TimeSpan.FromSeconds(1);
            this.timer.Tick += this.Timer_Tick;
            this.timer.Start();
        }

        private void Timer_Tick(object sender, object e)
        {
            readSensor();
        }

        private async void readSensor()
        {

            var light = this.hat.GetLightLevel();
            // Temperature Sensor
            var temp = this.hat.GetTemperature();

            double ftemp = ConvertTemp.ConvertCelsiusToFahrenheit(temp);

            System.Diagnostics.Debug.WriteLine("Temperature: {0} °C, Light {1}", ftemp.
            ToString("N2"), light.ToString("N2"));

            string message = "{\"temperature\" :" + temp.ToString() + "}";

            //send to listbox
            listBox.Items.Add(message);
        }

    }
    static class ConvertTemp
    {
        public static double ConvertCelsiusToFahrenheit(double c)
        {
            return ((9.0 / 5.0) * c) + 32;
        }
    }
}
```

This code sends the code to the ListBox. This is a good way to make sure your FEZ HAT is working. Once working, comment out the three lines of code below the `//send to Azure` comment and rerun it. The code above also gets the light sensor information but does not display it. Feel free to change the code to display the light data.

To run this application, you will need to follow the same simple steps for adding the IP address or name of the Raspberry Pi to the project. Simply click the Properties node in Solution Explorer and go to the debug settings. Enter the IP address and click Select. You should be good to go. Make sure your output is Remote Machine, as shown in Figure 2-11, and then compile and run the project! You should see the temperature displayed in the ListBox each second.

In Chapter 4, you'll modify these two examples to have the data saved to Azure IoT Hub. But first, let's do this same thing using a Tessel.

Tessel

The Tessel board is an open source, community-driven project driven by the Tessel Project. The aim of the Tessel is the development and learning for IoT devices. The recently released Tessel 2, shown in Figure 2-15, shipped in March of 2016. Compared to the Raspberry Pi, it is a bit smaller and a bit less powerful, but it is still very functional and is a fantastic option for IoT development. In Figure 2-15, the device on the left is the climate module which will be used to gather temperature data, similar to what was done with the Raspberry Pi.

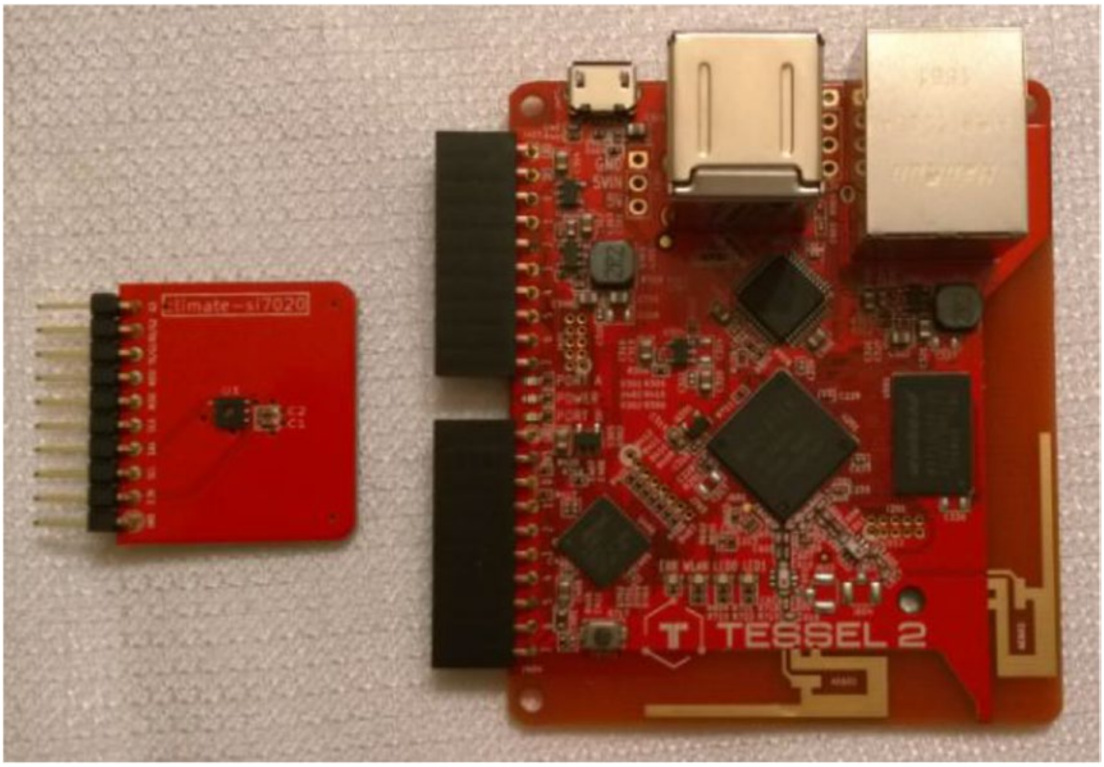


Figure 2-15. *Tessel 2 with a climate sensor*

New to Tessel 2 is improved Wi-Fi, an Ethernet jack, two USB ports, and support for Node.js (whereas Tessel 1 was JavaScript-based). Tessel uses Node.js as its language of choice but should support Python and Rust in the future.

Let's get started. For these examples you can use any code editor. Two popular editors are Visual Studio Code (`https://code.visualstudio.com/`) or Notepad++ (`https://notepad-plus-plus.org/`). I'll be using Visual Studio Code.

Before plugging in the Tessel, download the following:

- Download version 4.4.4 of Node.js from `https://nodejs.org/download/release/v4.4.4/`
- Download the Zadig USB driver install from `http://zadig.akeo.ie/`

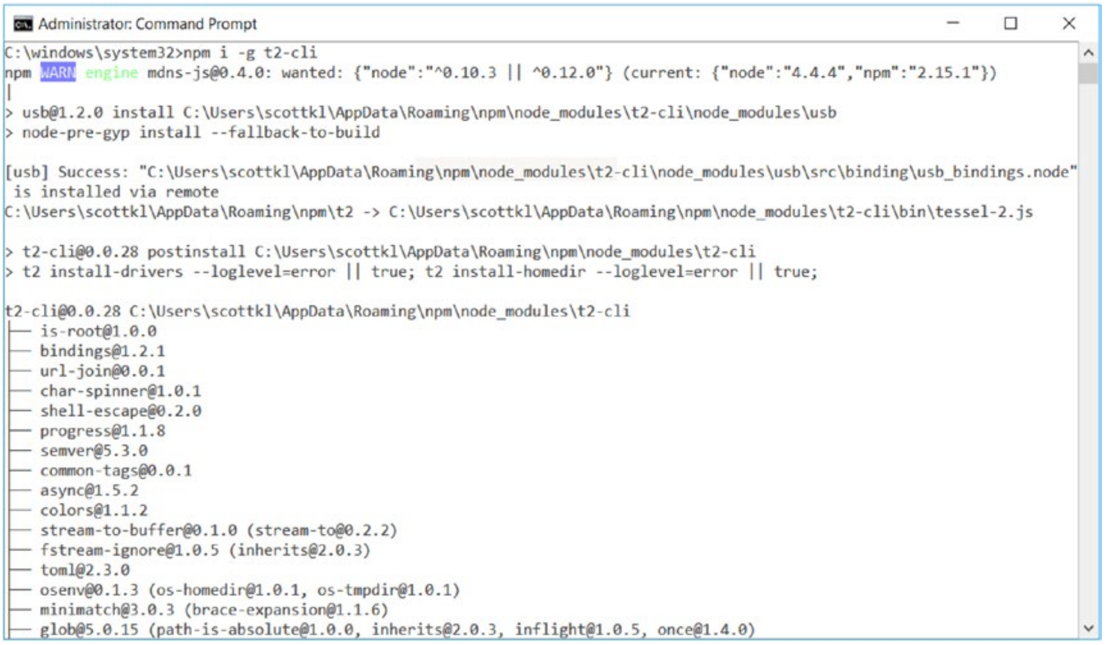
Once these are downloaded, run the Node.js install. You probably don't need to reboot after the installation completes, but it might be a good idea just in case. Next, plug in your Tessel. Your Tessel came with a USB power cable. Go ahead and connect your Tessel to power it on. A blue light will appear on your Tessel.

Next, run the Zadig USB tool, which will install the generic USB drivers such as WinUSB. When the Zadig tool launches, accept all the defaults and simply click the Install button. The installation will only take a few seconds. Once installed, close the Zadig tool.

Open a Command prompt as Administrator and at the command prompt type the following to install the Tessel drivers:

```
npm install i -g t2-cli
```

The output will look similar to Figure 2-16.



```
C:\windows\system32>npm i -g t2-cli
npm WARN engine mdns-js@0.4.0: wanted: {"node":"^0.10.3 || ^0.12.0"} (current: {"node":"4.4.4","npm":"2.15.1"})
|
> usb@1.2.0 install C:\Users\scottkl\AppData\Roaming\npm\node_modules\t2-cli\node_modules\usb
> node-pre-gyp install --fallback-to-build

[usb] Success: "C:\Users\scottkl\AppData\Roaming\npm\node_modules\t2-cli\node_modules\usb\src\binding\usb_bindings.node"
 is installed via remote
C:\Users\scottkl\AppData\Roaming\npm\t2 -> C:\Users\scottkl\AppData\Roaming\npm\node_modules\t2-cli\bin\tessel-2.js

> t2-cli@0.0.28 postinstall C:\Users\scottkl\AppData\Roaming\npm\node_modules\t2-cli
> t2 install-drivers --loglevel=error || true; t2 install-homedir --loglevel=error || true;

t2-cli@0.0.28 C:\Users\scottkl\AppData\Roaming\npm\node_modules\t2-cli
├── is-root@1.0.0
├── bindings@1.2.1
├── url-join@0.0.1
├── char-spinner@1.0.1
├── shell-escape@0.2.0
├── progress@1.1.8
├── semver@5.3.0
├── common-tags@0.0.1
├── async@1.5.2
├── colors@1.1.2
├── stream-to-buffer@0.1.0 (stream-to@0.2.2)
├── fstream-ignore@1.0.5 (inherits@2.0.3)
├── toml@2.3.0
├── osenv@0.1.3 (os-homedir@1.0.1, os-tmpdir@1.0.1)
├── minimatch@3.0.3 (brace-expansion@1.1.6)
├── glob@5.0.15 (path-is-absolute@1.0.0, inherits@2.0.3, inflight@1.0.5, once@1.4.0)
```

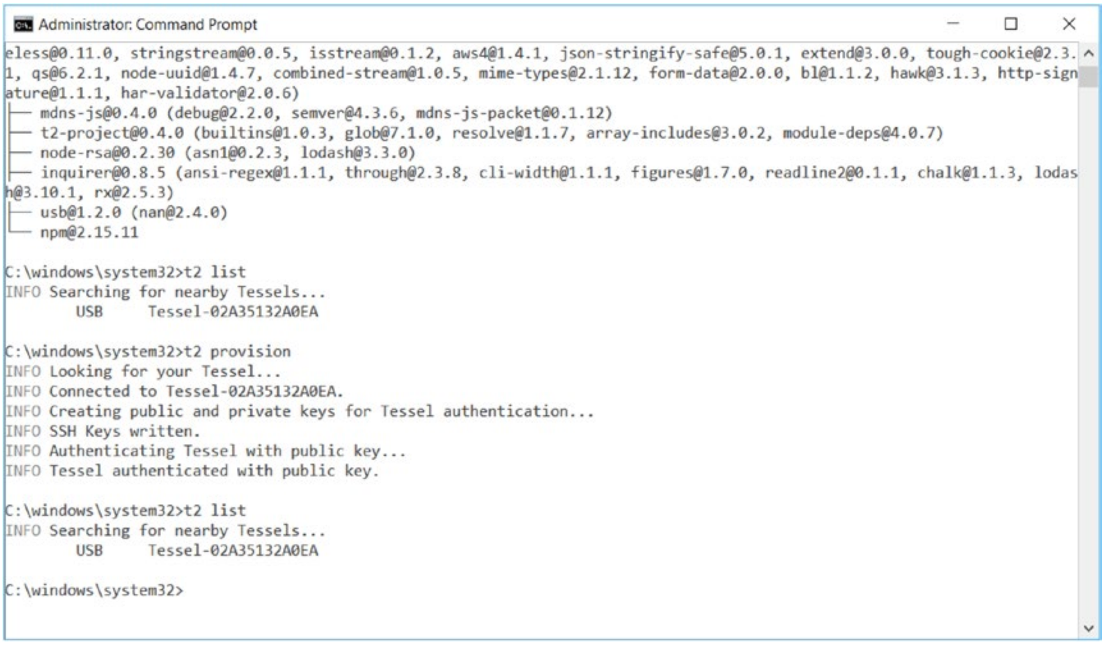
Figure 2-16. *Setting up the Tessel environment*

Once the Tessel drivers are installed, type the following command and press Enter. This command will list your Tessel and any other Tessels available over Wi-Fi and USB.

```
t2 list
```

As shown in Figure 2-17, your Tessel should be listed, specifying that is connected via USB. Next, type the following command and press Enter:

```
t2 provision
```



```
eless@0.11.0, stringstream@0.0.5, isstream@0.1.2, aws4@1.4.1, json-stringify-safe@5.0.1, extend@3.0.0, tough-cookie@2.3.
1, qs@6.2.1, node-uuid@1.4.7, combined-stream@1.0.5, mime-types@2.1.12, form-data@2.0.0, bl@1.1.2, hawk@3.1.3, http-sign
ature@1.1.1, har-validator@2.0.6)
├── mdns-js@0.4.0 (debug@2.2.0, semver@4.3.6, mdns-js-packet@0.1.12)
├── t2-project@0.4.0 (builtins@1.0.3, glob@7.1.0, resolve@1.1.7, array-includes@3.0.2, module-deps@4.0.7)
├── node-rsa@0.2.30 (asn1@0.2.3, lodash@3.3.0)
├── inquirer@0.8.5 (ansi-regex@1.1.1, through@2.3.8, cli-width@1.1.1, figures@1.7.0, readline2@0.1.1, chalk@1.1.3, lodas
h@3.10.1, rx@2.5.3)
├── usb@1.2.0 (nan@2.4.0)
└── npm@2.15.11

C:\windows\system32>t2 list
INFO Searching for nearby Tessels...
        USB     Tessel-02A35132A0EA

C:\windows\system32>t2 provision
INFO Looking for your Tessel...
INFO Connected to Tessel-02A35132A0EA.
INFO Creating public and private keys for Tessel authentication...
INFO SSH Keys written.
INFO Authenticating Tessel with public key...
INFO Tessel authenticated with public key.

C:\windows\system32>t2 list
INFO Searching for nearby Tessels...
        USB     Tessel-02A35132A0EA

C:\windows\system32>
```

Figure 2-17. *Configuring the Tessel 2*

The `provision` command authorizes your computer to access and push code to the connected Tessel 2 via SSH. As shown in Figure 2-17, the provisioning generates public and private keys for Tessel authentication.

If you don't like the name of your Tessel, you can rename your Tessel by typing `t2 rename` and specifying a new name. For example,

```
t2 rename "newname"
```

Executing `t2 update` updates the firmware of the Tessel, and executing `npm install i -g t2-cli` any time will check for, and update, the Tessel drivers.

It's time to do some actual work. Power off the Tessel and then plug in the climate sensor to Port A of the Tessel with the electrical components up. Plug the Tessel back in. Create a directory on your hard drive, and in the command prompt, navigate to that directory. Install the climate sensor drivers by executing the following command:

```
npm install climate-si7010
```

Next, open up Visual Studio Code or your favorite text editor and enter the following code:

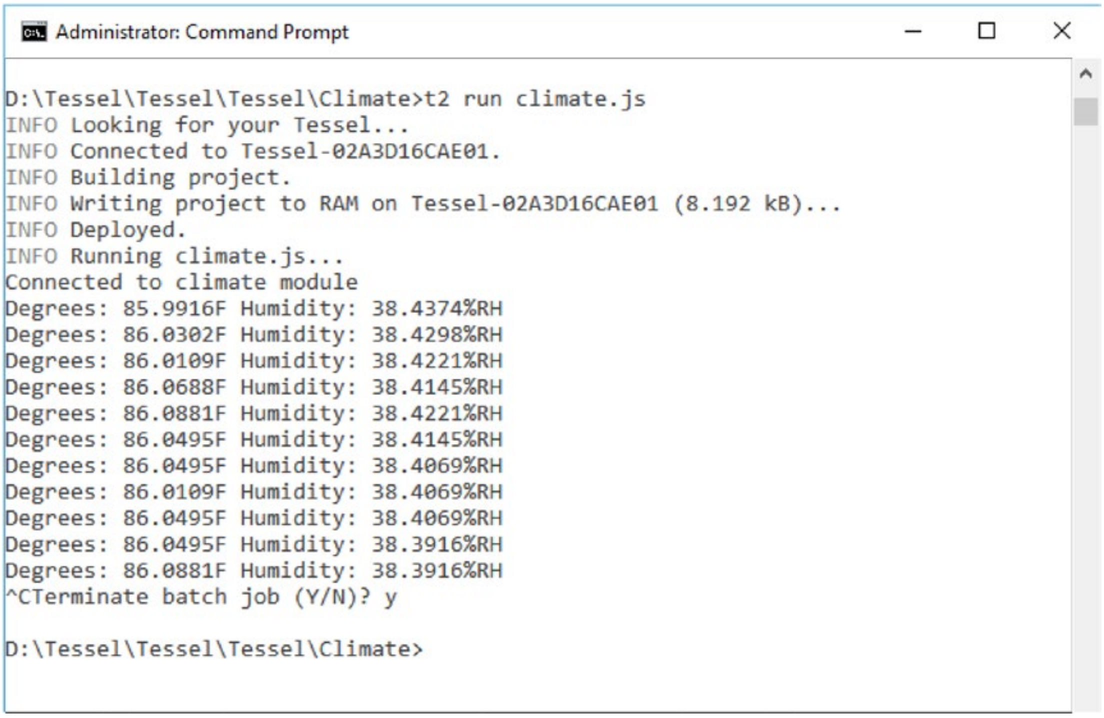
```
var tessel = require('tessel');
var climatelib = require('climate-si7020');

var climate = climatelib.use(tessel.port['A']);

climate.on('ready', function () {
  console.log('Connected to climate module');
  // Loop forever
  setImmediate(function loop () {
    climate.readTemperature('f', function (err, temp) {
      climate.readHumidity(function (err, humid) {
      console.log('Degrees:', temp.toFixed(4) + 'F', 'Humidity:', humid.toFixed(4) + '%RH');
      setTimeout(loop, 300);
      });
    });
  });
});

climate.on('error', function(err) {
  console.log('error connecting module', err);
});
```

The code above loops each second and gets the temperature and humidity reading from the sensor plugged into the Tessel. I have a few things to point out. The first three lines of code specify that the following code needs the Tessel and climate drivers, and that the climate sensor is plugged in to Port A on the Tessel.

Save the file as `climate.js`, and in the same command prompt used earlier to the directory you created above, type the following and press Enter:

```
t2 run climate.js
```

The `t2 run` command finds your connected Tessel, compiles the code (in this case, `climate.js`), and then deploys the code to the Tessel and runs it, as shown in Figure 2-18.



```
D:\Tessel\Tessel\Tessel\Climate>t2 run climate.js
INFO Looking for your Tessel...
INFO Connected to Tessel-02A3D16CAE01.
INFO Building project.
INFO Writing project to RAM on Tessel-02A3D16CAE01 (8.192 kB)...
INFO Deployed.
INFO Running climate.js...
Connected to climate module
Degrees: 85.9916F Humidity: 38.4374%RH
Degrees: 86.0302F Humidity: 38.4298%RH
Degrees: 86.0109F Humidity: 38.4221%RH
Degrees: 86.0688F Humidity: 38.4145%RH
Degrees: 86.0881F Humidity: 38.4221%RH
Degrees: 86.0495F Humidity: 38.4145%RH
Degrees: 86.0495F Humidity: 38.4069%RH
Degrees: 86.0109F Humidity: 38.4069%RH
Degrees: 86.0495F Humidity: 38.4069%RH
Degrees: 86.0495F Humidity: 38.3916%RH
Degrees: 86.0881F Humidity: 38.3916%RH
^CTerminate batch job (Y/N)? y

D:\Tessel\Tessel\Tessel\Climate>
```

Figure 2-18. *Receiving temperature and humidity data from the Tessel 2*

Congratulations on getting your Tessel up and running. Along with the previous two examples, Chapter 4 will also modify this example so that the data is sent to Azure.

For more information on the Tessel, visit `https://tessel.io/`.

Big data and the IoT are about gathering, processing, analyzing, and gaining valuable insights into extremely large quantities of data. Remember that two of the three Vs are volume and velocity. Thus, imagine not one but hundreds or thousands of these boards generating data.

Summary

This chapter focused on two different devices that generate data, the Raspberry Pi and Tessel. First, I walked you through how to install Windows 10 IoT Core on a Pi 2 and a Pi 3, and then I walked you through an example of using the Pi and a temperature sensor to generate data. That was followed by a similar example using a Tessel board to show Microsoft's support for open source technologies.

The next chapter will introduce the Microsoft Azure IoT Hub and its importance in the IoT service spectrum.