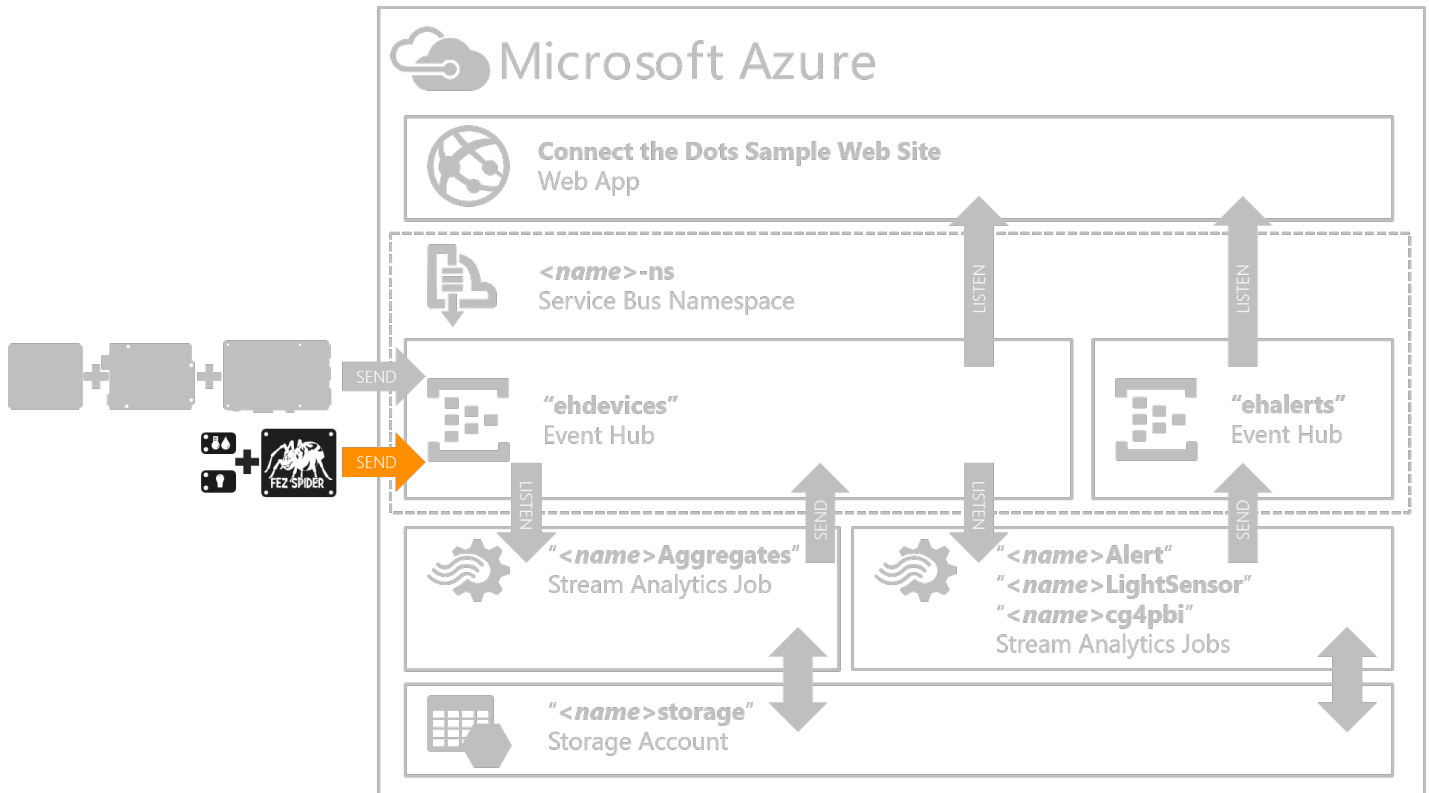


"NETMF Gadgeteer Configuration" Hands-On Lab

Overview

In this lab you will use A Gadgeteer FEZ Spider and the .NET Micro Framework to publish sensor data into the "ehdevices" Azure Event Hub you created previously in the "Azure Prep" Hands-On Lab.



Prerequisites

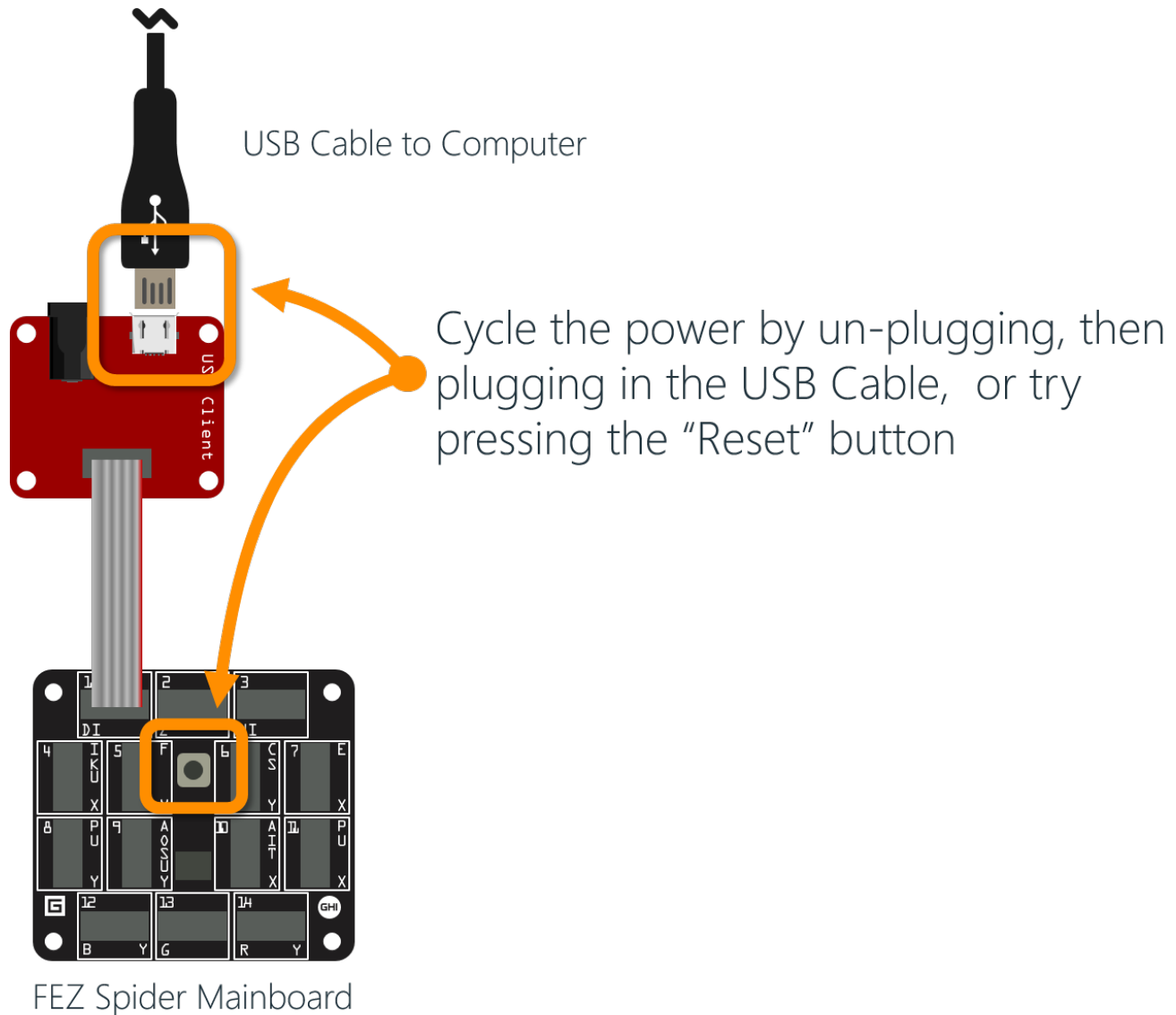
To successfully complete this lab, you will need:

- An active Azure Subscription. If needed you can create a [free trial here](#).
- A Windows computer with Visual Studio 2013 Community Edition or better [download](#). **NOTE: YOU CAN'T USE VISUAL STUDIO 2015 FOR THIS LAB. IT MUST BE VISUAL STUDIO 2013.**
- A copy of the ConnectTheDots.io repository. You can get the latest version [here](#).
- You will need to download and install the software components covered in the "Install Dependencies" task
 - .NET Framework 3.5. This is included as a Windows Feature, but you need to verify that it is "On"
 - .Net Micro Framework Core SDK
 - .Net MF Visual Studio 2013 integration
 - .Net Gadgeteer Core
 - GHI NETMF and Gadgeteer package
- You will need a FEZ Spider and the additional hardware components covered in the "Connect the Components to the FEZ Spider Mainboard" task.
 - Gadgeteer FEZ Spider Mainboard
 - Gadgeteer USB DP Module
 - Gadgeteer Ethernet J11D Module
 - Gadgeteer TempHumid S170 Module
 - Gadgeteer LightSense Module

Troubleshooting Tips

If you have problems with the Gadgeteer, you can refer to the [NETMF](#) and [Gadgeteer Troubleshooting Guide](#)

1. If you have problems deploying to the Gadgeteer, for example after it crashes, or you stop a running debug session, you can try resetting the Gadgeteer using either the Reset button, or by cycling the power:



Tasks

1. [Install Dependencies](#)
 2. [Connect the Components to the FEZ Spider Mainboard](#)
 3. [Modify and Deploy the ConnectTheDotsGadgeteer Project](#)
 4. [Update the SSL Seed on the Gadgeteer](#)
 5. [Verify Your Results](#)
-

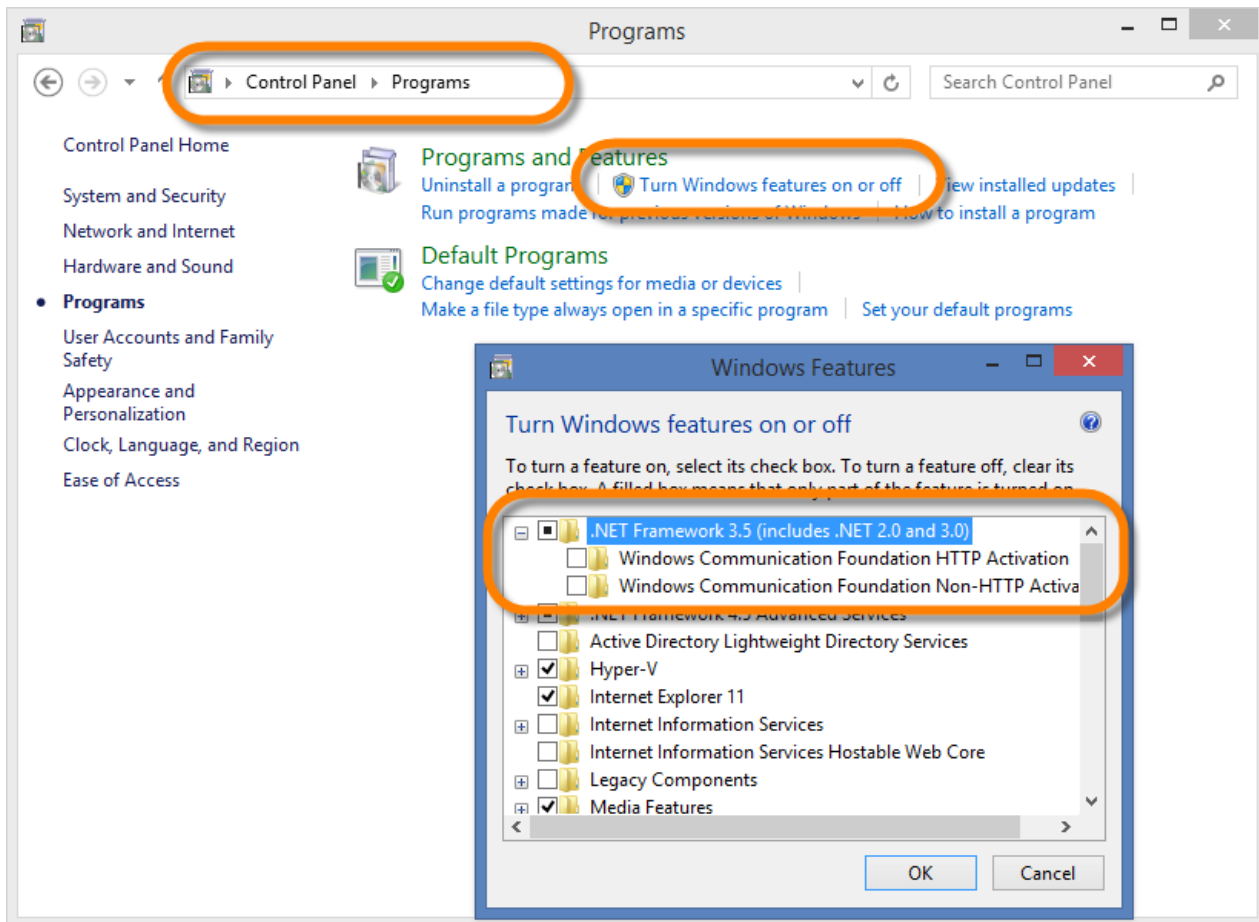
Task 1

To complete this lab you will need Visual Studio 2013 Community Edition or better. At the time this is being written, Visual Studio 2015 is NOT supported so you will need VS 2013. The free Community Edition works fine, but if you already have a VS 2013 Professional or VS 2013 Ultimate,

those work just fine as well.

You need to install the .NET Micro Framework SDK, and a few additional tools to help Visual Studio work with NETMF and the Gadgeteer boards.

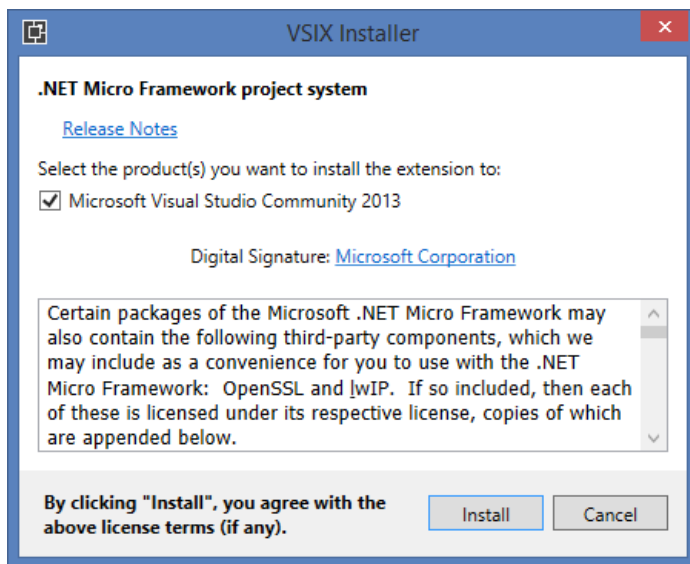
1. Verify that the .NET Framework 3.5 is enabled on your system. To do so, go to your **"Control Panel" | "Programs" | "Turn Windows features on or off"** and ensure that the .NET Framework 3.5 is enabled. You do not need the Windows Communication Foundation Activation features enabled.



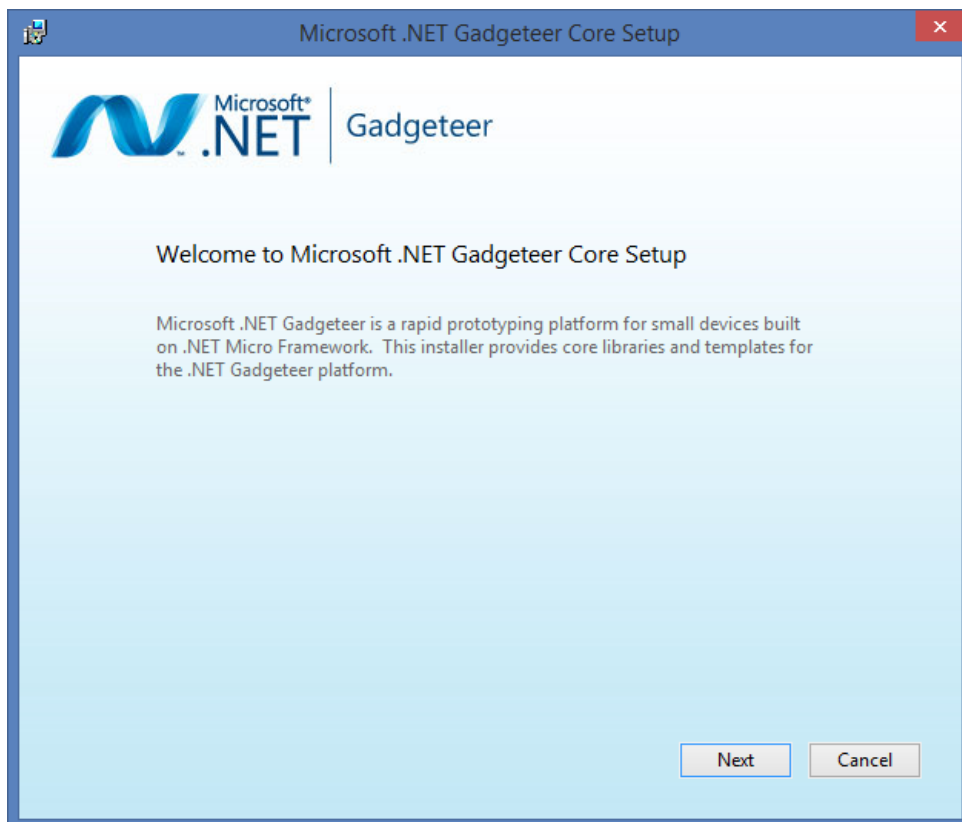
2. Download and install the .Net Micro Framework Core SDK 4.3 (SDK-R2-Beta)



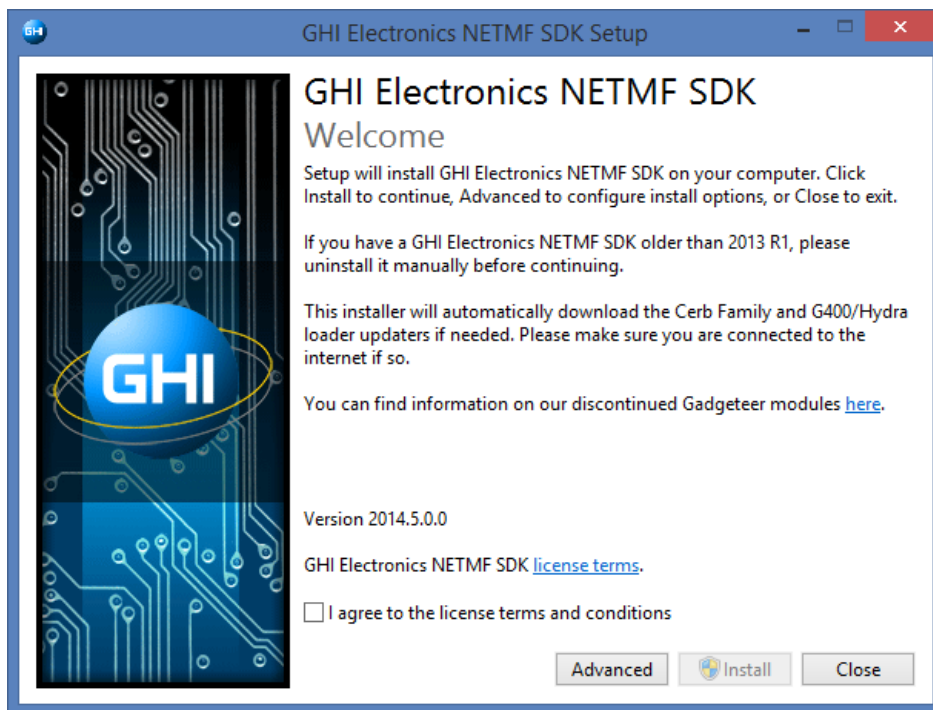
3. Download and install the .Net MF Visual Studio 2013 integration



4. Download and install the .Net Gadgeteer Core



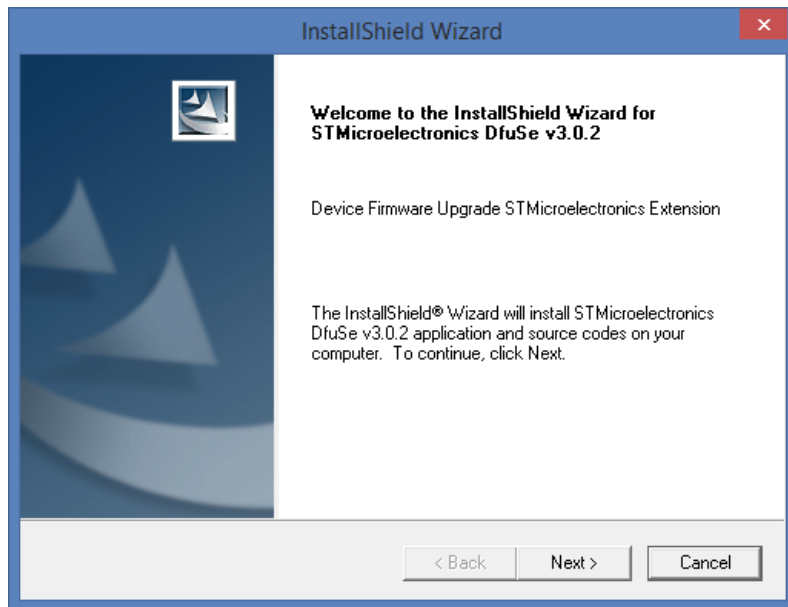
5. Finally, download and install the GHI NETMF and Gadgeteer package This installer actually calls other installers. You need to pay attention as you run through this because you will need to answer the prompts for:



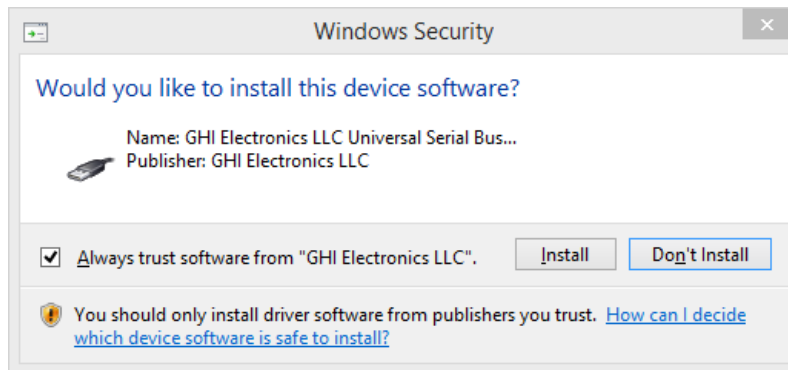
- The nested "Atmel SAM Boot Assistant" ("SAM-BA") installer:



- The ST Microelectronics "Device Firmware Upgrade ST Microelectronics Extension" ("DfuSe") installer:



- GHI USB Driver:

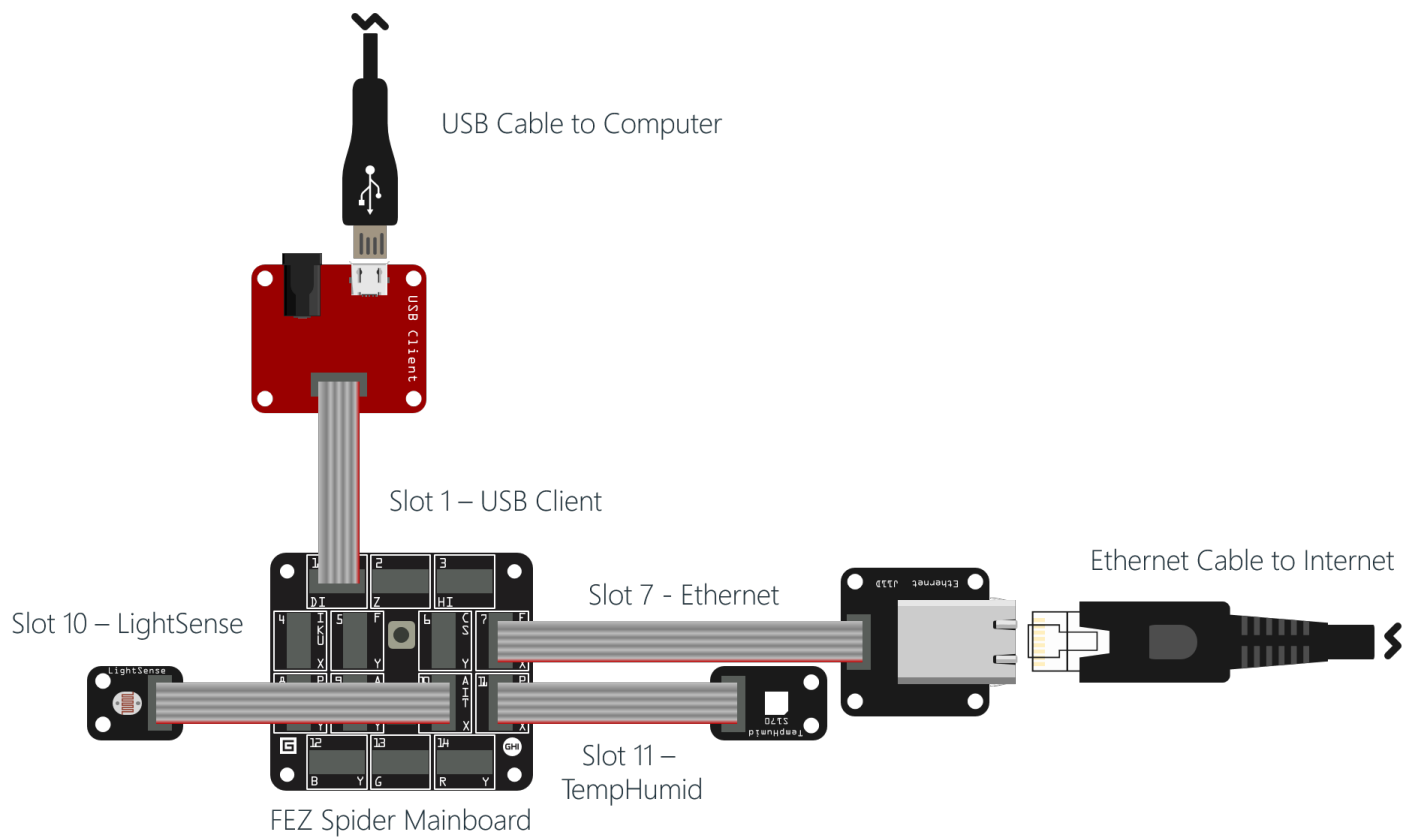


6. Make sure that you complete all of the required installs. If you miss one, you will not be able to complete the lab.

Task 2 - Connect the Components to the FEZ Spider Mainboard

One of the benefits of the Gadgeteer platform is the ease with which you can connect components. In this task, you need to simply connect the components as described below:

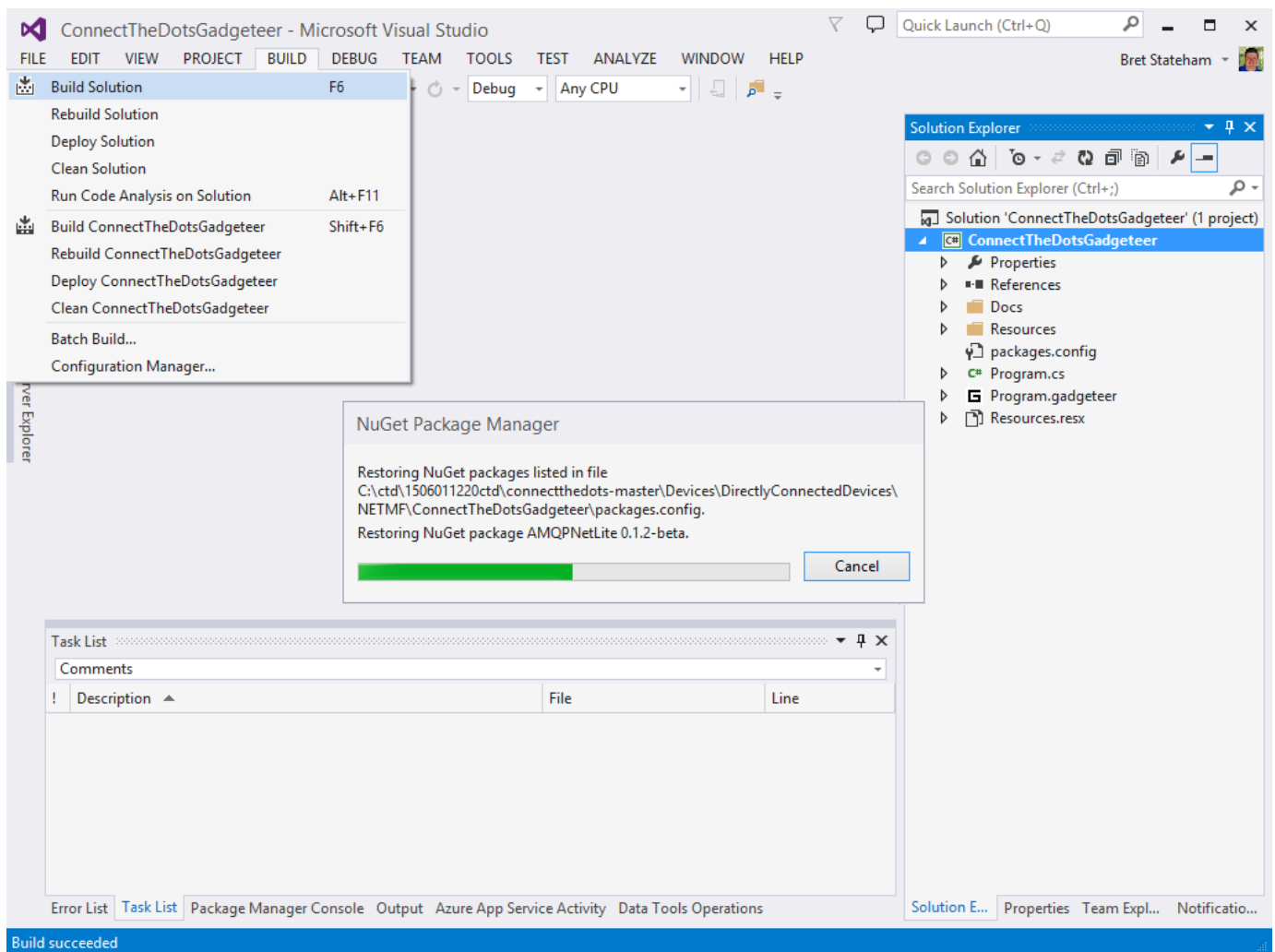
1. Connect the [Gadgeteer LightSense Module](#) to **Slot 10** on the FEZ Spider Mainboard.
2. Connect the [Gadgeteer TempHumid S170 Module](#) to **Slot 11** on the FEZ Spider Mainboard
3. Connect the [Gadgeteer Ethernet J11D Module](#) to **Slot 7** on the FEZ Spider Mainboard
4. Connect the [Gadgeteer USB DP Module](#) to **Slot 1** on the FEZ Spider Mainboard



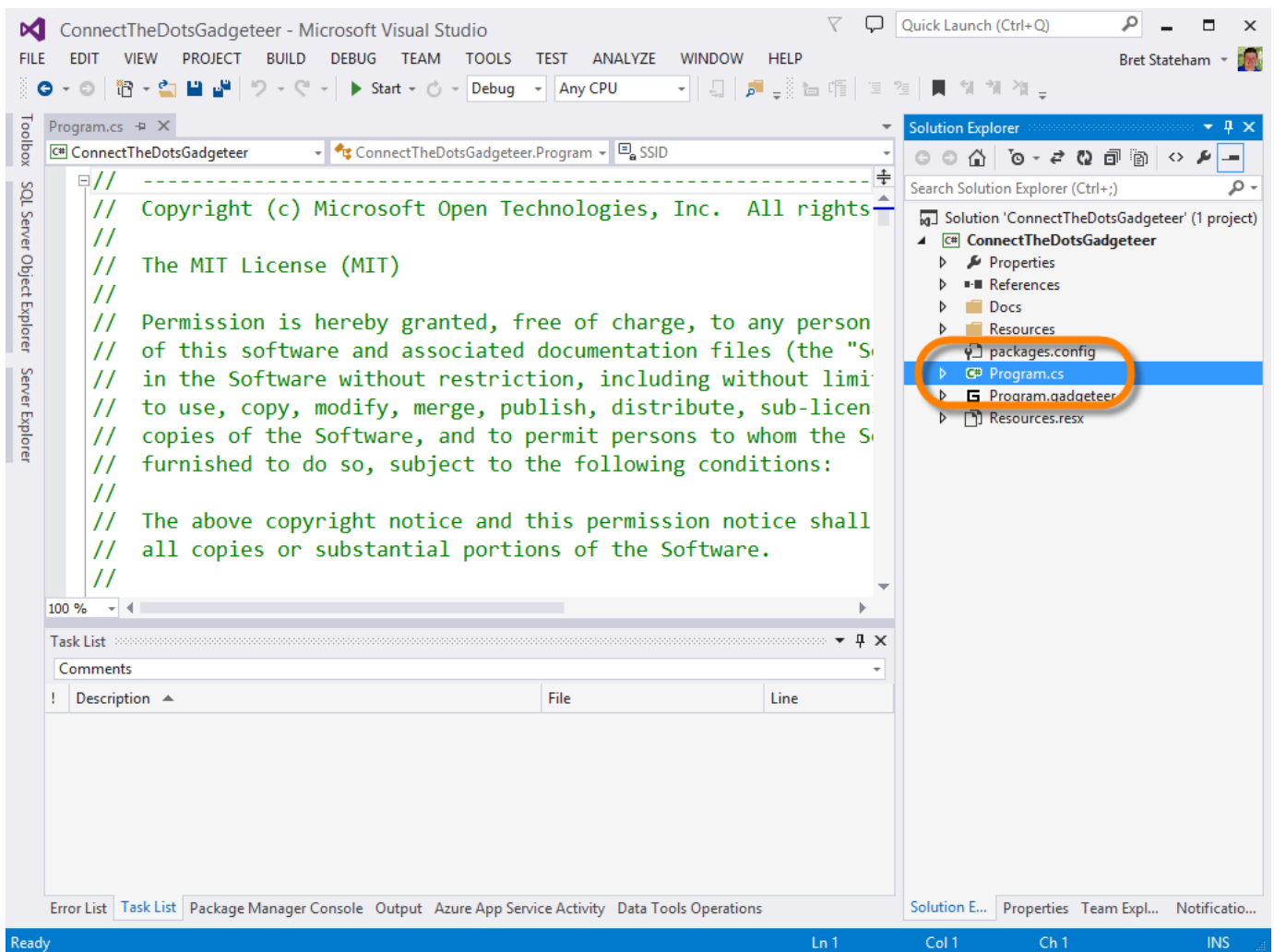
Task 3 - Modify and Deploy the ConnectTheDotsGadgeteer Project

In this task, you will modify the "ConnectTheDotsGadgeteer" solution in Visual Studio to point it towards your. "ehdevices" Event Hub.

1. In Visual Studio 2013, open the `\Devices\DirectlyConnectedDevices\NETMF\ConnectTheDotsGadgeteer.sln` Solution.
2. To download the required NuGet packages, and ensure that the project base is functional, go ahead and do build. From the menu bar select **"BUILD" | "Build Solution"**, and once the build is done, ensure that it reads **"Build succeeded"** in the lower left hand corner.



3. In the "**Solution Explorer**" Window, double click on the "**Program.cs**" file top open it.



4. In the "**Program.cs**" file, locate the lines that configure the client connect to the "**ehdevices**" event hub (should be somewhere around line 54):

```
// Azure Event Hub connection information
private const string AMQPAddress = "amqps://{key-name}:{key}@{namespace-name}.servicebus.windows.net"; // Azure event hub conn

private const string EventHub = "ehdevices"; // Azure event hub name
private const string SensorName = "Gadgeteer"; // Name of the device you want to display
private const string SensorGUID = "{GUID}"; // unique GUID per device. Use GUIDGEN to generate new one

private const string Organization = "MSOpenTech"; // Your organization name
private const string Location = "My Room"; // Location of the device

// Define the frequency at which we want the device to send its sensor data (in milliseconds)
const int SendFrequency = 60000;
private readonly GT.Timer _timer = new GT.Timer(SendFrequency);
```

5. Verify that the `EventHub` variable value matches the name of the Event Hub you created for "**ehdevices**". If you called it something other than "**ehdevices**" make sure to update the variable value to match:

```
private const string EventHub = "ehdevices"; // Azure event hub name
```

6. Leave your Visual Studio window open, open the browser, and login to the [Azure Management Portal](https://manage.windowsazure.com) (<https://manage.windowsazure.com>).
7. Navigate the portal to find your "**ehdevices**" event hub, and on the "**CONFIGURE**" page, and get the "**PRIMARY ACCESS KEY**" for your "**D1**" "**Shared Access Policy**".



Microsoft Azure | CREDIT STATUS | bretdemo@live.com



shared access policies

NAME	PERMISSIONS
D1	Send
D2	Send
D3	Send
D4	Send
WebSite	Manage, Send, Listen
StreamingAnalytics	Manage, Send, Listen
NEW POLICY NAME	

shared access key generator

POLICY NAME: D1

PRIMARY KEY: L2wFYmNQ05PqD3n6d1/ScRPJB/dbzCHYh1RQwWVuBjc=  

SECONDARY KEY: N4LCZwt7C+faVNXFKQoWs28BKC8hNEDwoc8Nbb89BF  

+ NEW

8. Before you can use the key though, we need to URL encode it. Go to <http://meyerweb.com/eric/tools/dencoder/> to use their URL Encoder / Decoder tool. Paste the key you just copied in, then hit the "Encode" button, then copy the encoded to the clipboard.

#Workspaces/ServiceBusExtens URL Decoder/Encoder

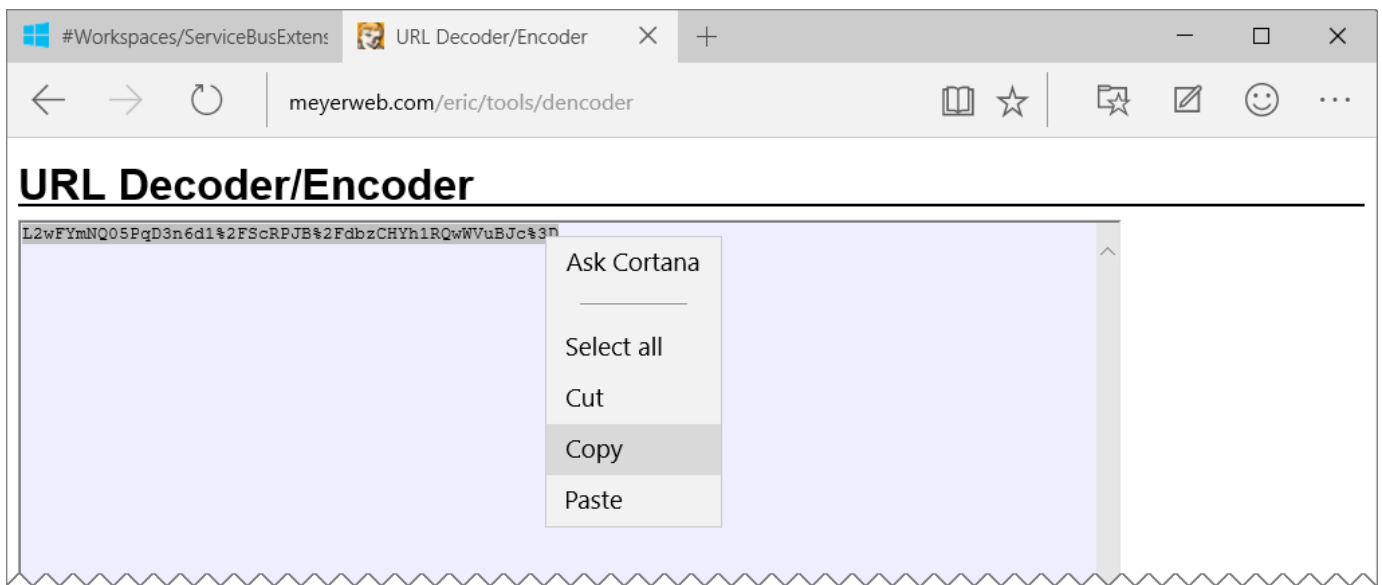
meyerweb.com/eric/tools/dencoder

URL Decoder/Encoder

L2wFYmNQ05PqD3n6d1/ScRPJB/dbzCHYh1RQwWVuBjc=

Decode Encode

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.



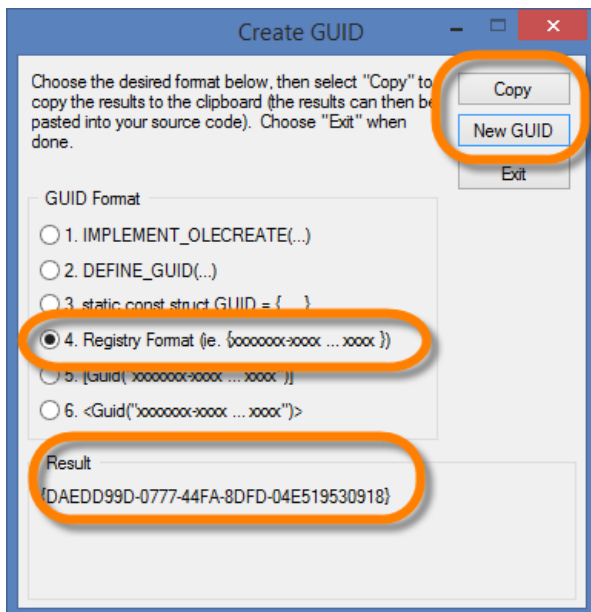
9. Back in Visual Studio, in the **"Program.cs"** file, edit the `AMQPAddress` the string variable. Replace the place holders with the values from your Service Bus Namespace & Event Hub:

Place Holder	Value
{key-name}	"D1" (no quotes)
{key}	The URL encoded version of the key you just copied.
{namespace-name}	The service bus namespace you created earlier, "ctdhol-ns" in this case

10. For example:

```
// Azure Event Hub connection information
private const string AMQPAddress = "amqps://D1:L2wFYmNQ05PgD3n6d1%2FScRPJB%2FdbzCHYh1RQwWVuBJc%3D@ctdhol-ns.servicebus.windows.net";
```

11. For the `SensorGUID` string, you can generate a new GUID value in Visual Studio by , from the menu bar, going to **"TOOLS" | "Create GUID"**. Then use the tool to generate a new GUID, copy it to the clipboard (Try using the "Registry Format"), then paste it into your code and remove the curly braces.



12. For example

```
private const string SensorGUID = "C62F1539-84CF-4F1C-AAF3-286E63A1099C"; // unique GUID per device. Use GUIDGEN to generate n
```

- For the SendFrequency variable, you may want to try a lower frequency, like 1000 (milliseconds). The default 60000 only publishes sensor data once a minute. For example:

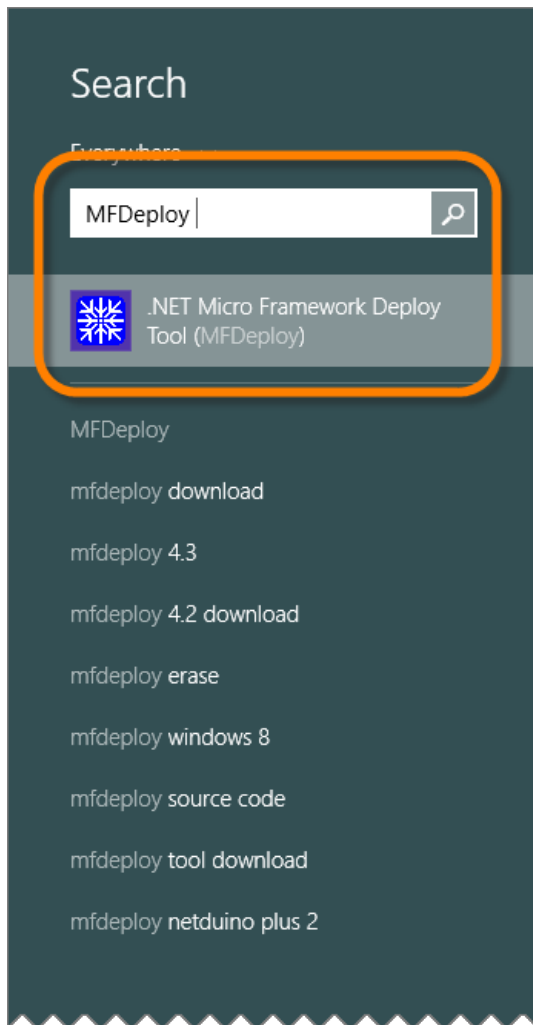
```
// Define the frequency at which we want the device to send its sensor data (in milliseconds)
const int SendFrequency = 1000;
```

- Lastly, update the SensorName, Organization, and Location variable values to whatever you like.

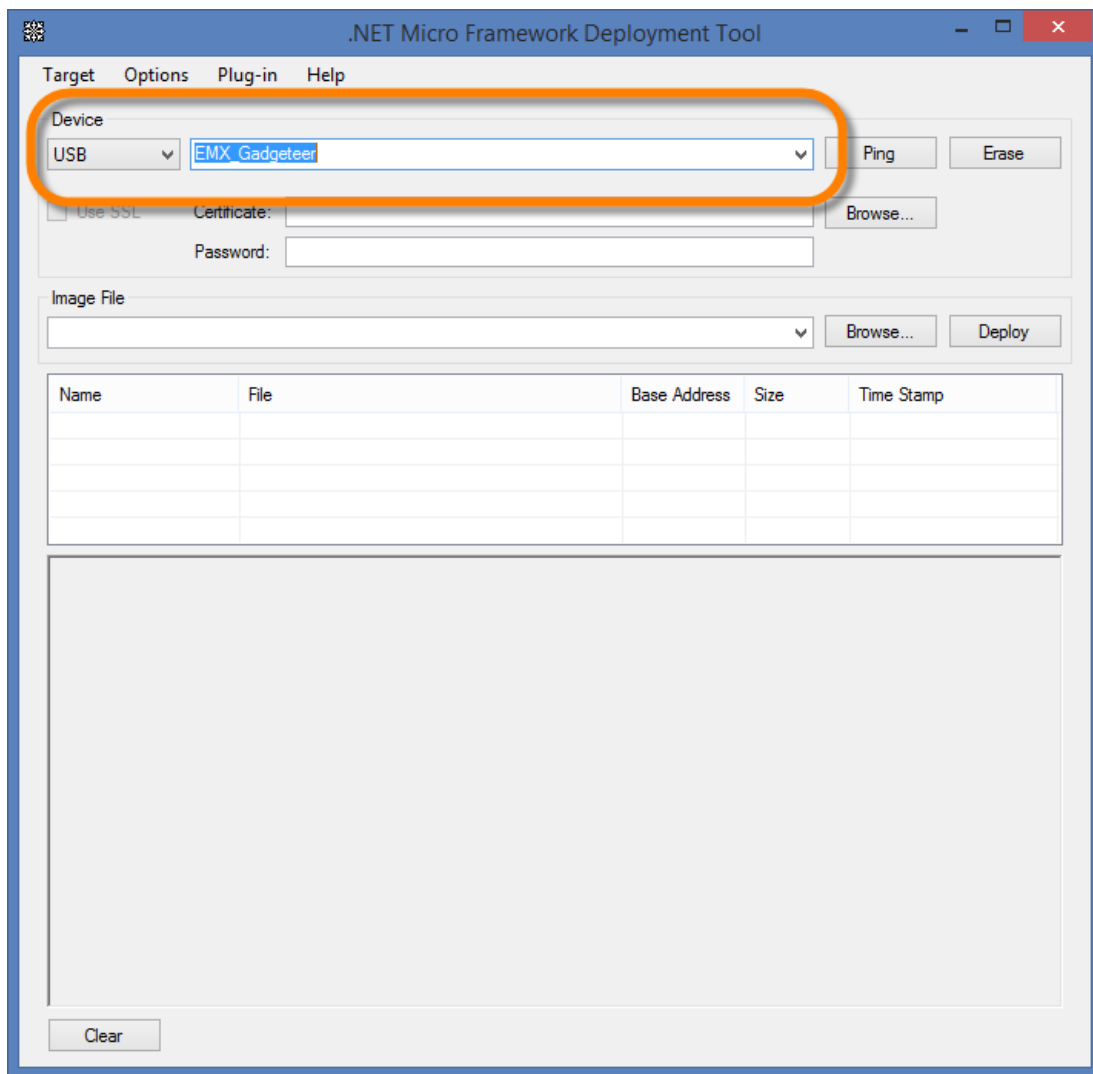
Task 4 - Update the SSL Seed on the Gadgeteer

Your Gadgeteer device needs a current, valid set of keys to participate in the SSL communications required by AMQP.

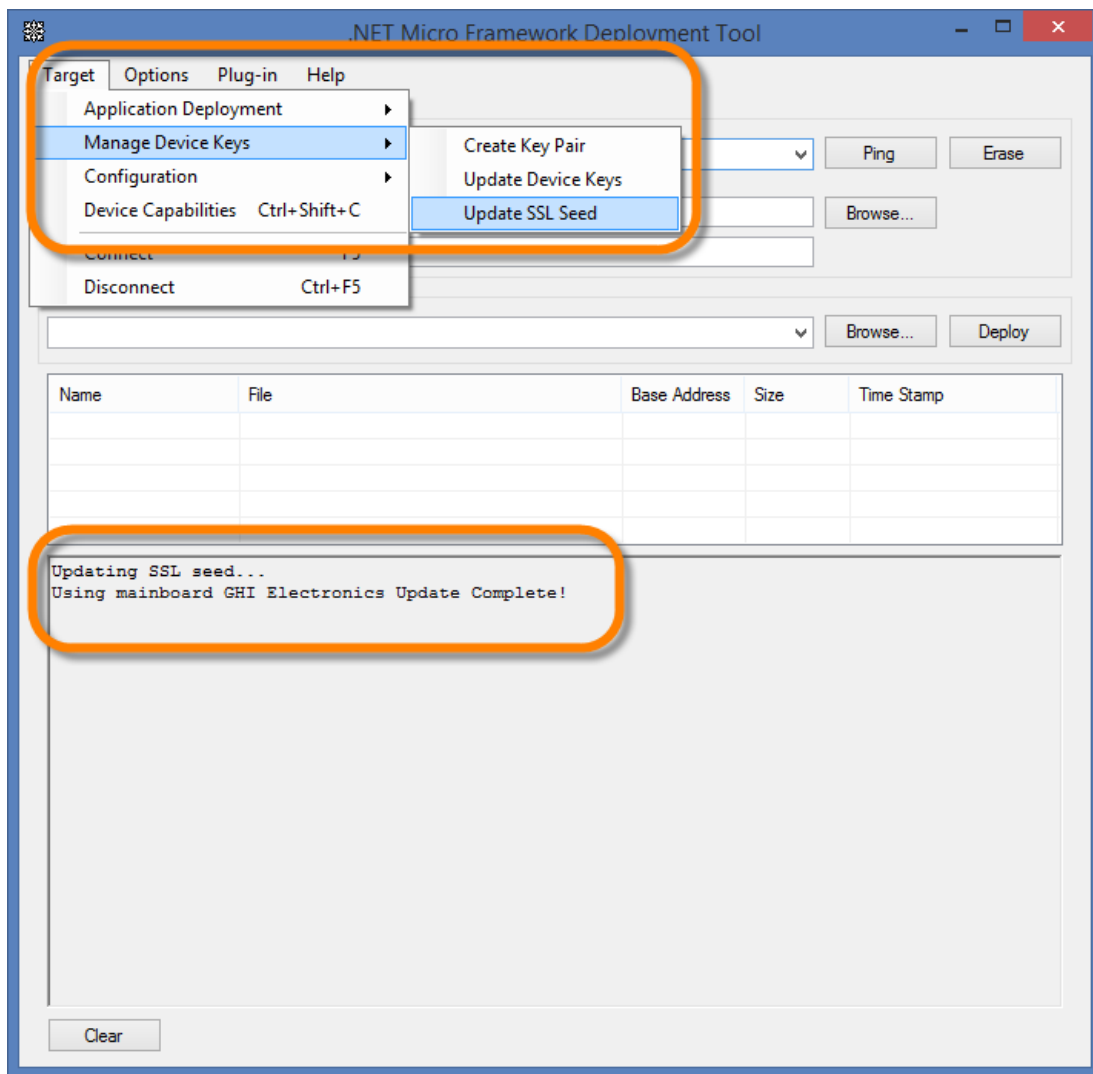
- From the Start Menu's Search function, find the MFDeploy.exe tool and run it:



- In the "Device" dropdown, select USB, then select your Gadgeteer:



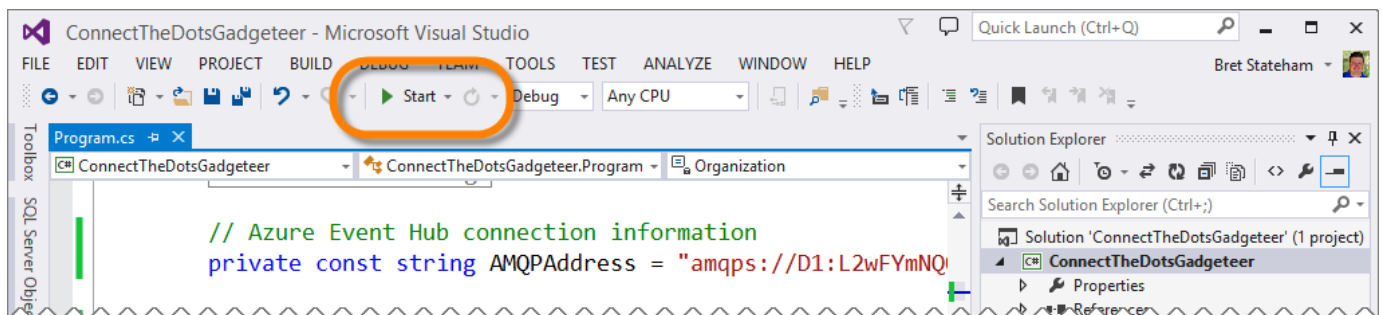
3. From the "MFDeploy" menu bar, select "Target" | "Manage Device Keys" | "Update SSL Seed", then verify that it completes.



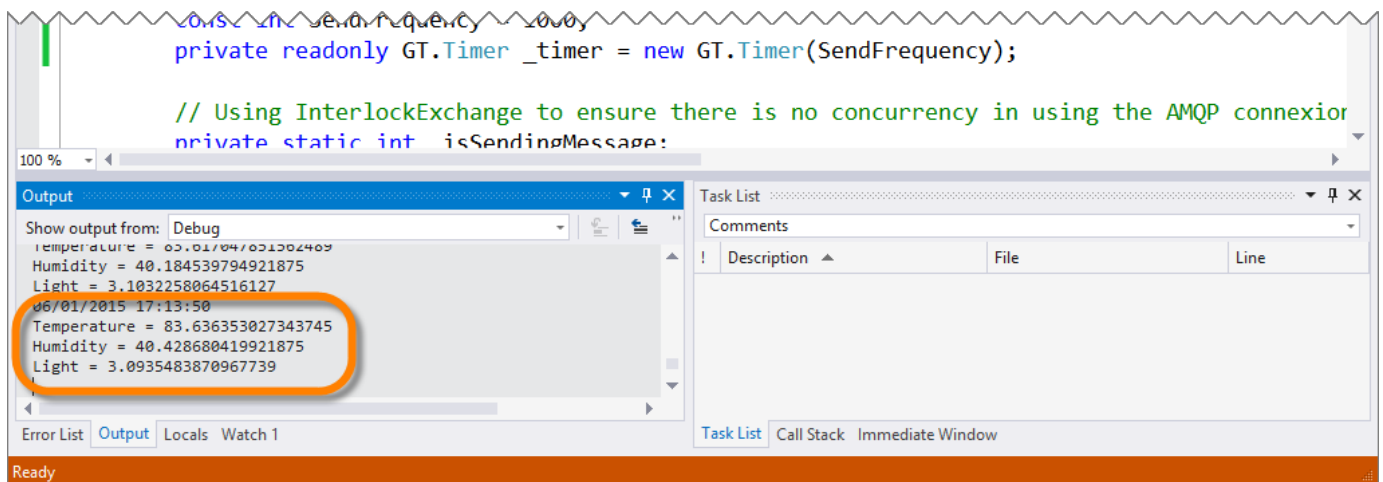
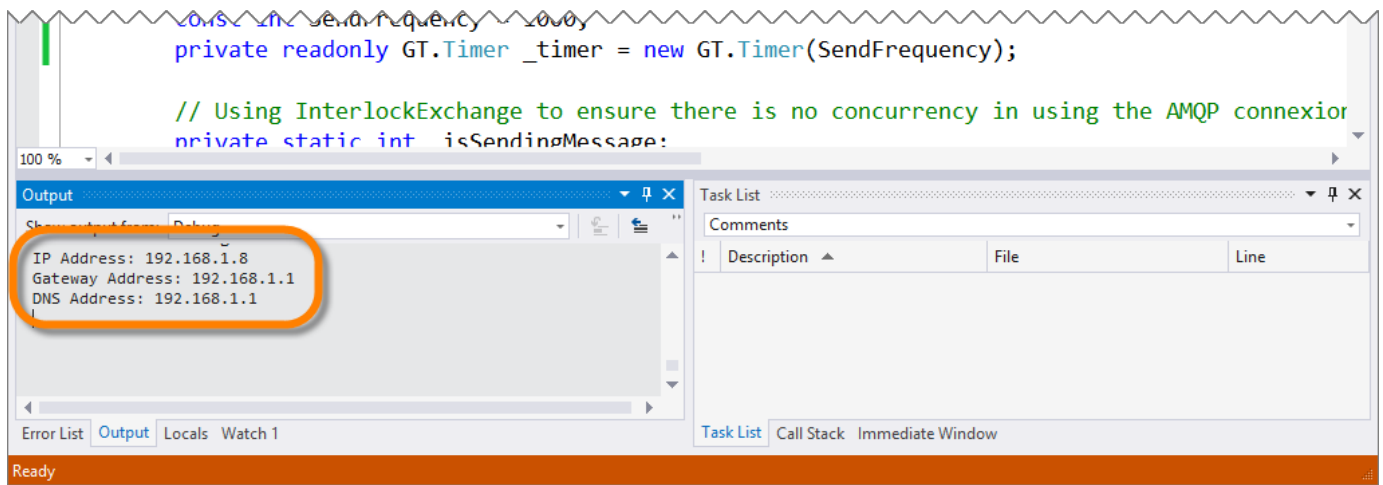
Task 5 - Verify Your Results

You should be ready to go. Let's deploy the code to the Gadgeteer and watch data flow in on the Connect the Dots Sample Website you created earlier.

1. In Visual Studio, hit the **"Start"** button on the toolbar to start a debug session:



2. In the Visual Studio **"Output"** Window, you should be seeing messages about the Gadgeteer's IP Address, as well as periodic messages (the rate of which depends on your `SendFrequency` variable value) appear.



3. If you open the "Connect the Dots Sample Web Site" you deployed in the **Connect the Dots Web Site** Hands-On Lab you should start to see sensor readings coming in.

