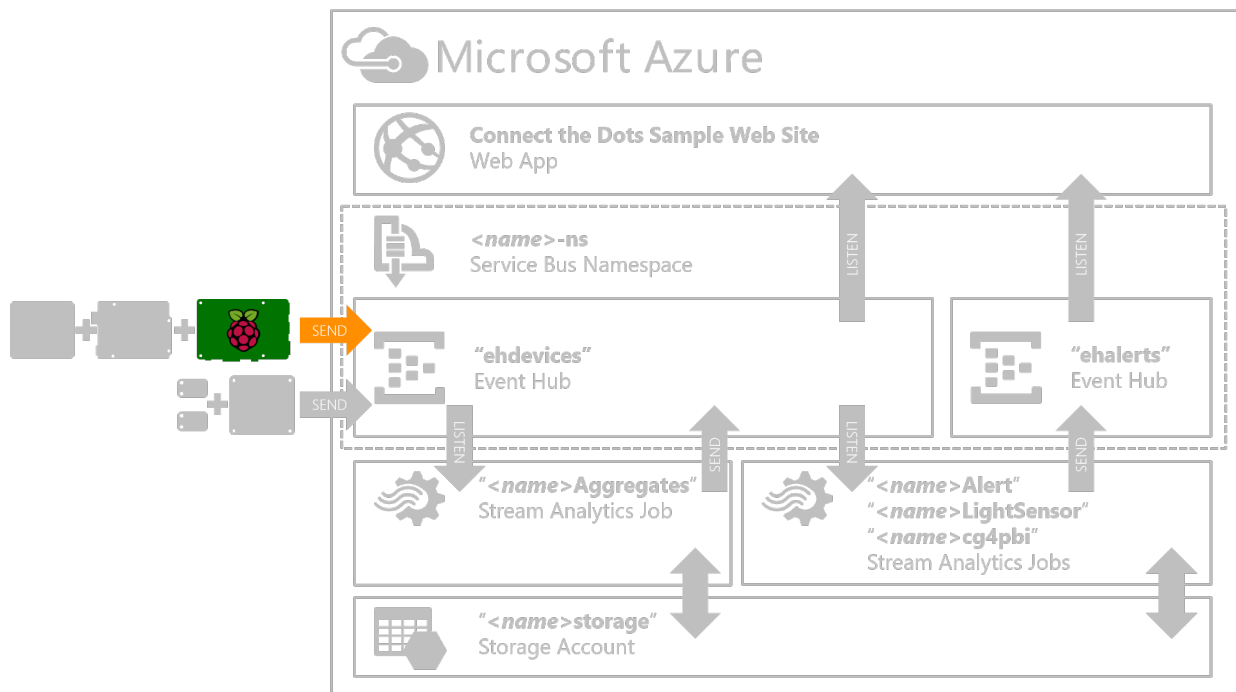


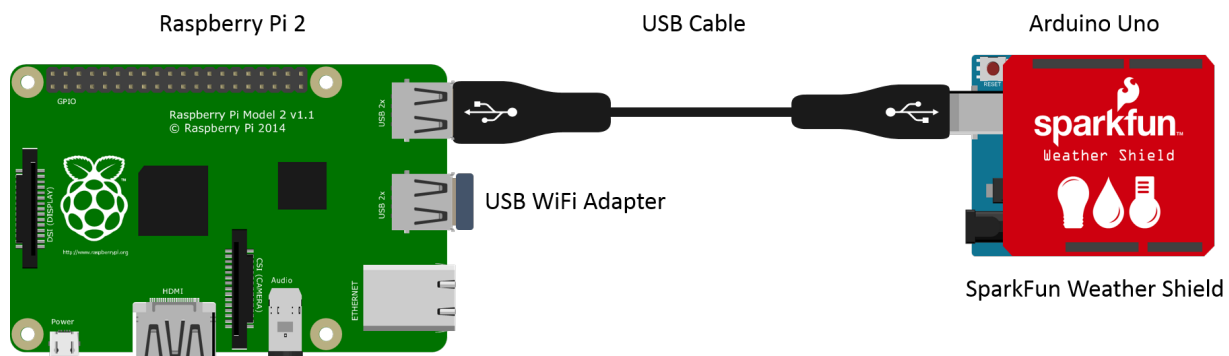
"Raspberry Pi Field Gateway" Hands-On Lab

Overview

In this lab, you will configure your Raspberry Pi 2 as a "Field Gateway".



A "Field Gateway" is a common solution for securing and transmitting data from lower powered microcontrollers, or those without direct network connectivity.



In our scenario, we have the Arduino Uno with the SparkFun Weather Shield that we implemented in a previous lab. The Arduino Uno is an awesome platform, but it does have a few real limitations:

- Limited processing power and memory. This makes it hard to communicate using secure protocols like HTTPS, or AMQPS where more intensive processing is required.
- No built in remote communication capabilities. You need to extend the Arduino with additional components for Ethernet, WiFi, Bluetooth, etc.

Our solution here then is to connect the Arduino to our Raspberry Pi using a USB-to-Serial connection. The Raspberry Pi can then receive the sensor data messages from the Arduino Uno over the serial connection, and then forward them on securely using HTTPS, or AMQPS over Ethernet or WiFi.

Alternative, More Hands-On Walkthrough

This Hands-On Lab is a simplified, and more streamlined version of the original Raspberry Pi gateway setup documentation. In this lab, we assume you are at an event where a pre-configured Raspberry Pi image has already been applied to the SD Card in your Raspberry Pi.

This pre-configured image already has

- Raspian operating system installed (via NOOBS)
- Mono

- WiFi Configuration
- The GatewayService .NET project already deployed

All you really need to do in this lab is:

- Remote into your Raspberry Pi via ssh or remote desktop
- Modify the GatewayService application configuration file with the path and keys to your "ehdevices" event hub
- Plug in your Arduino with SparkFun Weather Shield
- Sit back and watch the data flow!

If you are at an event where the pre-configured image is available you may want to start with this lab, then if you have time and want to get more hands on with the Pi setup, you can wipe out the SD card and start over, following the documentation in the [Original Raspberry Pi Gateway Setup Docs](#)

Prerequisites

To successfully complete this lab, you will need:

- An active Azure Subscription. If needed you can create a [free trial here](#).
- A copy of the ConnectTheDots.io repository. You can get the latest version [here](#).
- An ssh client (like PuTTY on Windows)
- A Raspberry Pi 2 with a USB WiFi adapter
- A copy of the Raspberry Pi image on an SD Card with the Gateway Service code pre-deployed. If you prefer to configure and deploy the GatewayService yourself, you can refer to the [Original Raspberry Pi Gateway Setup Documentation](#)
- Previous completion of the "Azure Prep" Hands-On Lab
- Previous Completion of the "Arduino Uno With SparkFun Weather Shield" Hands-On Lab
- Knowledge of your Raspberry Pi's IP address.

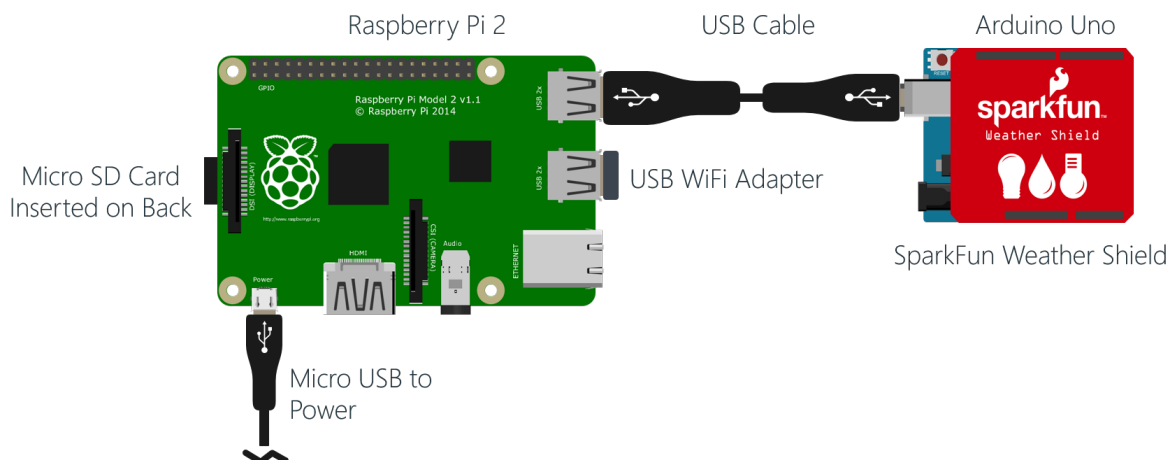
Tasks

1. [Boot your Raspberry Pi off the Pre-Configured Image](#)
2. [Modify the Gateway Config](#)
3. [Connect the Arduino Uno](#)
4. [Task 4](#)
5. [Task 5](#)

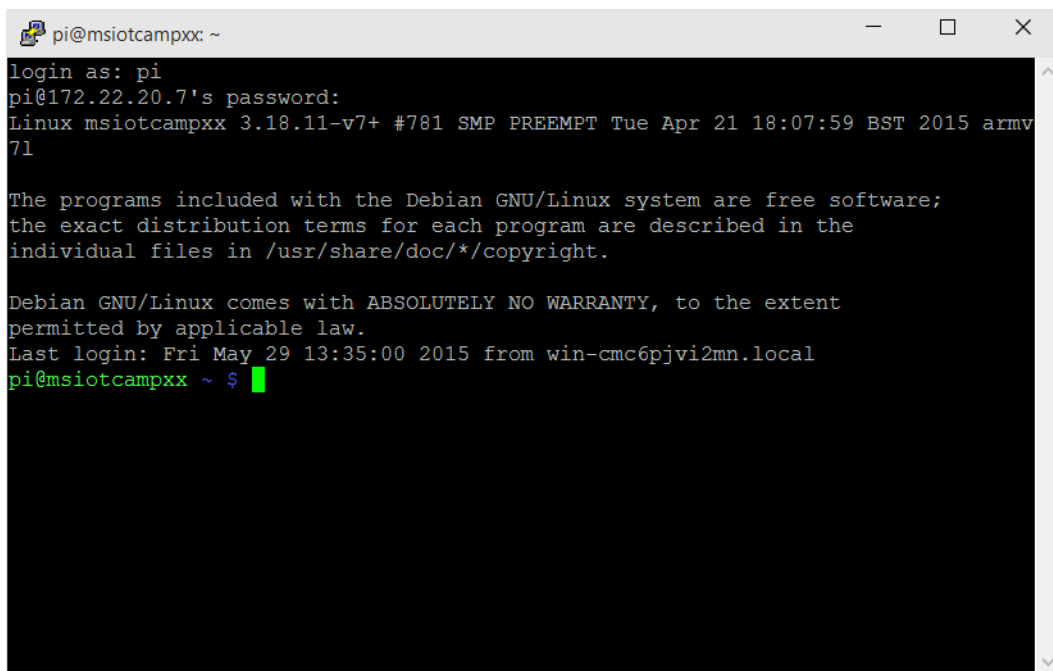
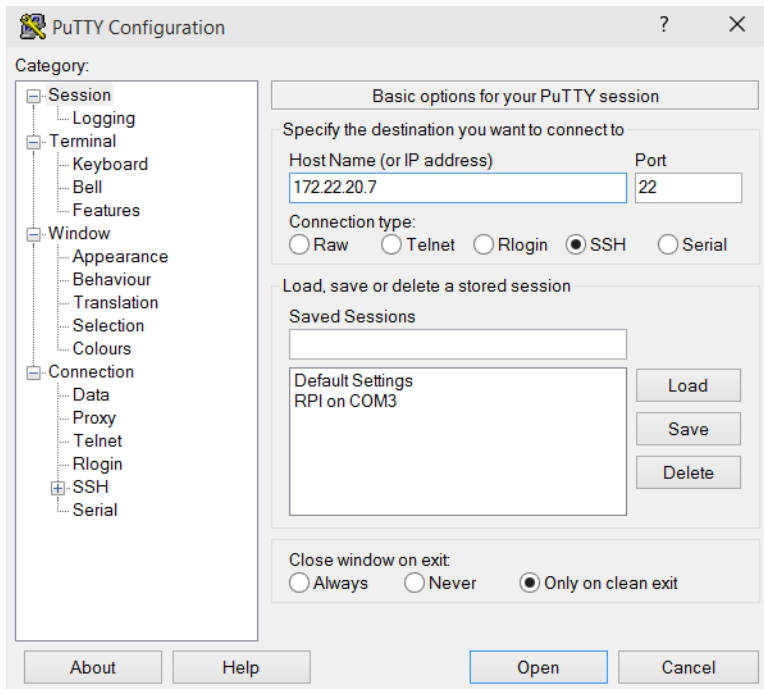
Task 1 - Boot your Raspberry Pi off the Pre-Configured Image

1. Ensure that the SD Card with the pre-configured image is installed in the Raspberry Pi
2. Ensure that the USB WiFi Adapter is connected to a USB port (or if you are using a direct wired ethernet cable, that the ethernet cable is plugged in)
3. Connect your Arduino Uno with the SparkFun Weather Shield attached to a USB port on the Raspberry Pi.
4. Finally connect the power supply to the Raspberry Pi
5. The following image should show you your approximate configuration

Note: If you don't have the Arduino ready yet, that's ok. You can plug it in later.



6. Work with your event staff to determine the Raspberry Pi's IP Address
7. Use an ssh client (You can use [PuTTY](#) on Windows) or a Remote Desktop Client (mstsc.exe on Windows) to connect to the IP Address of your Raspberry Pi and login with the credentials:
 - Login: **pi**
 - Password: **raspberry**



8. The SD Card is configured with the "Raspbian" linux distribution, so the commands that you enter will be linux commands. Start by getting a listing of your home folder by typing `ls` and pressing enter. Notice the "**ctdgtwy**" folder name:

```
ls
```

```
pi@msiotcampxx: ~
login as: pi
pi@172.22.20.7's password:
Linux msiotcampxx 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 29 14:11:26 2015 from win-cmc6pjvi2mn.local
pi@msiotcampxx ~ $ ls
ctdgtwy Desktop python_games
pi@msiotcampxx ~ $
```

- The "ctdgtwy" is the folder that contains the "GatewayService" deployment. The "GatewayService" is actually a .NET application that is being run on the Raspberry Pi using the Mono open source .NET implementation. If you are interesting in seeing that source code, and how it was deployed, refer to the Original Raspberry Pi Gateway Setup Docs. Here, we'll just assume it is deployed correctly.
- Change into the ctdgtwy/staging folder (ctdgtwy is short for "Connect the Dots Gateway"), do another `ls` command and notice the (very long named) "Microsoft.ConnectTheDots.GatewayService.exe.config" (whew!) file.

```
cd ctdgtwy/staging
```

```
pi@msiotcampxx: ~/ctdgtwy/staging
Last login: Fri May 29 14:11:26 2015 from win-cmc6pjvi2mn.local
pi@msiotcampxx ~ $ ls
ctdgtwy Desktop python_games
pi@msiotcampxx ~ $ cd ctdgtwy/staging
pi@msiotcampxx ~/ctdgtwy/staging $ ls
Microsoft.Common.dll
Microsoft.ConnectTheDots.Common.pdb
Microsoft.ConnectTheDots.Gateway.dll
Microsoft.ConnectTheDots.Gateway.pdb
Microsoft.ConnectTheDots.GatewayService.exe
Microsoft.ConnectTheDots.GatewayService.exe.config
Microsoft.ConnectTheDots.GatewayService.pdb
Microsoft.ConnectTheDots.SerialPortAdapter.dll
Microsoft.ConnectTheDots.SocketAdapter.dll
Newtonsoft.Json.dll
Newtonsoft.Json.xml
NLog.config
NLog.dll
NLog.xml
pi@msiotcampxx ~/ctdgtwy/staging $
```

- We need to edit the contents of that file. There are numerous text editors available on linux, and if you have on you prefer, feel free to use it. We will use a simple one called "Nano". Enter the command:

```
nano Microsoft.ConnectTheDots.GatewayService.exe.config
```

```
pi@msiotcampxx: ~/ctdgtwy/staging
pi@msiotcampxx ~/ctdgtwy/staging $ nano Microsoft.ConnectTheDots.GatewayService.exe.config
```

12. Use the arrow keys on your keyboard to move down through the file and locate the section that reads:

```
<AMQPServiceConfig
AMQPAddress="amqp://[key-name]:[key]@[namespace].servicebus.windows.net"
EventHubName="ehdevices"
EventHubMessageSubject="gtsv"
EventHubDeviceId="a94cd58f-4698-4d6a-b9b5-4e3e0f794618"
EventHubDeviceDisplayName="SensorGatewayService"/>
```

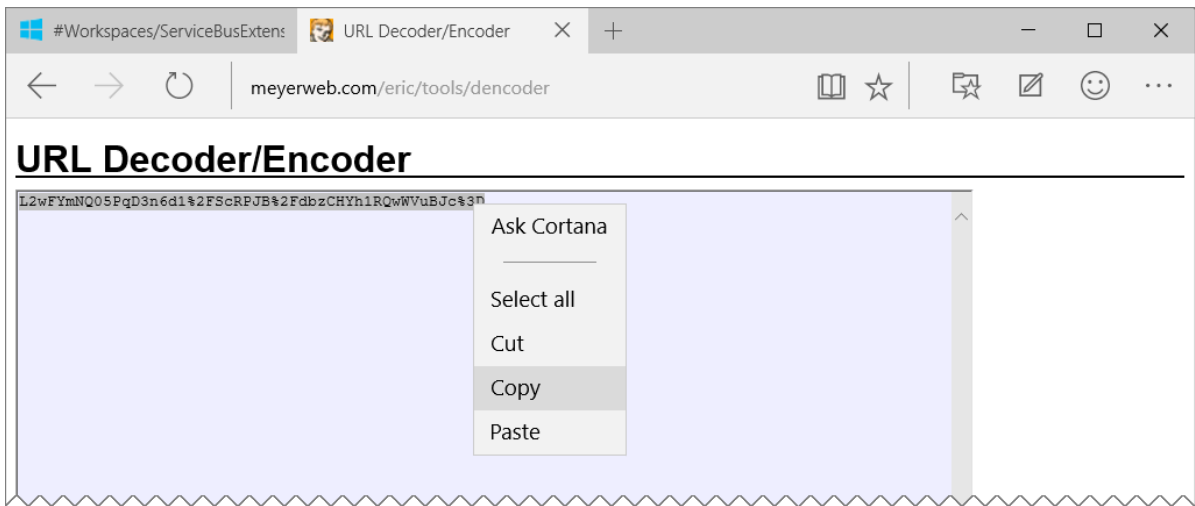
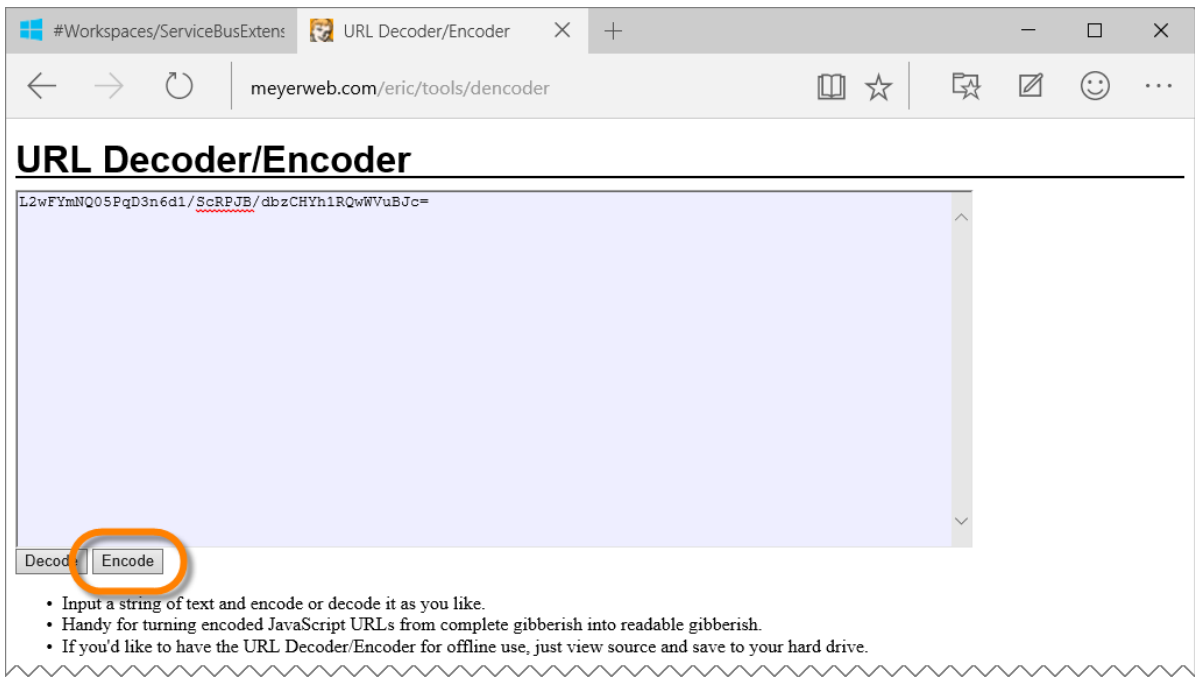
13. Notice the missing [key-name], [key], and [namespace] placeholders. We need to enter those so that the Raspberry Pi can successfully connect to the "ehdevices" event hub we created previously.
14. Leave your ssh window open, and back on your computer open the browser, login to the [Azure Management Portal](https://manage.windowsazure.com) (<https://manage.windowsazure.com>).
15. 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".

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains navigation icons for various services, with 'ehdevices' selected. The main content area is titled 'shared access policies' and displays a table of policies:

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

Below the table is a 'NEW POLICY NAME' input field. Below that is the 'shared access key generator' section, which is highlighted with an orange border. It contains a 'POLICY NAME' dropdown set to 'D1', a 'PRIMARY KEY' field with a copy icon and a 'Regenerate' button, and a 'SECONDARY KEY' field with a copy icon and a 'Regenerate' button.

16. 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.



17. Back in your ssh, and nano, use the arrow keys and your key and keyboard to edit the string. 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. Note that many ssh clients (like PuTTY) will paste whatever is in your clipboard if you right click. So you can delete the place-holder with the keyboard, get the cursor in the right place, then right click to paste the encoded version of the key you copied to the clipboard previously
[namespace]	The service bus namespace you created earlier, "ctdhol-ns" in this case

```
pi@msiotcampxx: ~/ctdgtwy/staging
GNU nano 2.2.6 File: ...ft.ConnectTheDots.GatewayService.exe.config Modified

    type="Microsoft.ConnectTheDots.Common.SensorEndpointConfigSection, Mic$
    requirePermission="true" restartOnExternalChanges="true" allowLocat$
<section name="AMQPServiceConfig"
    type="Microsoft.ConnectTheDots.Common.AMQPServiceConfigSection, Mic$
    requirePermission="true" restartOnExternalChanges="true" allowLocat$
<section name="dataTransformsConfig"
    type="Microsoft.ConnectTheDots.Common.DataTransformsConfigSection, $
    requirePermission="true" restartOnExternalChanges="true" allowLocat$
</configSections>

<AMQPServiceConfig
  AMQPAddress="amqps://D1:L2wFYmNQ05PqD3n6d1%2FSsCRPJB%2FdbzCHYh1RQwWVuBJc%3D$
  EventHubName="ehdevices"
  EventHubMessageSubject="gtstv"
  EventHubDeviceId="a94cd58f-4698-4d6a-b9b5-4e3e0f794618"
  EventHubDeviceDisplayName="SensorGatewayService"/>

<dataIntakes>
</dataIntakes>

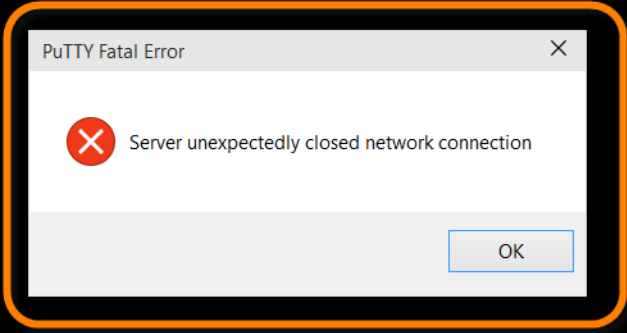
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

18. Finally, to save your changes in Nano, press "Ctrl-X" (Exit), the "Y" to save the changes, and then "ENTER" to confirm the original file name. And as long as you didn't make any typos, you should be good to go.
19. To reboot your Raspberry PI, "DON'T JUST UNPLUG IT!. SHUT IT DOWN NICELY!!!!". in your ssh window, run the following command to shut reboot it. If you are using PuTTY you'll see an error about being disconnected, of course that is to be expected:

```
sudo reboot
```

```
pi@msiotcampxx: ~/ctdgtwy/staging
pi@msiotcampxx ~/ctdgtwy/staging $ sudo reboot

Broadcast message from root@msiotcampxx (pts/0) (Fri May 29 15:06:39 2015):
The system is going down for reboot NOW!
pi@msiotcampxx ~/ctdgtwy/staging $
```

A PuTTY Fatal Error dialog box is overlaid on the terminal window. It has a title bar that says "PuTTY Fatal Error" with a close button. The main area contains a red circle with a white 'X' icon and the text "Server unexpectedly closed network connection". At the bottom right is an "OK" button.

20. When the Raspberry PI starts back up, you should now be able to go to your website and see sensor values coming in!

