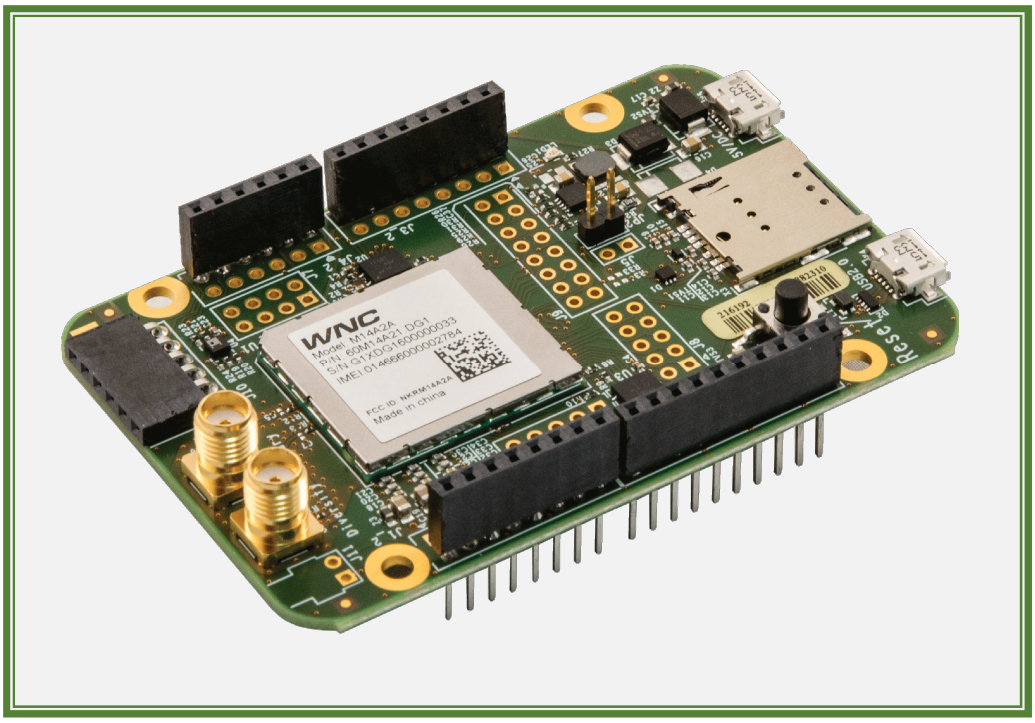


LTE IoT Add-on Kit for Raspberry Pi 3

- Getting Started Guide -

AVNET®

Reach Further™

Version 1.3

June 6, 2017

Table of Contents

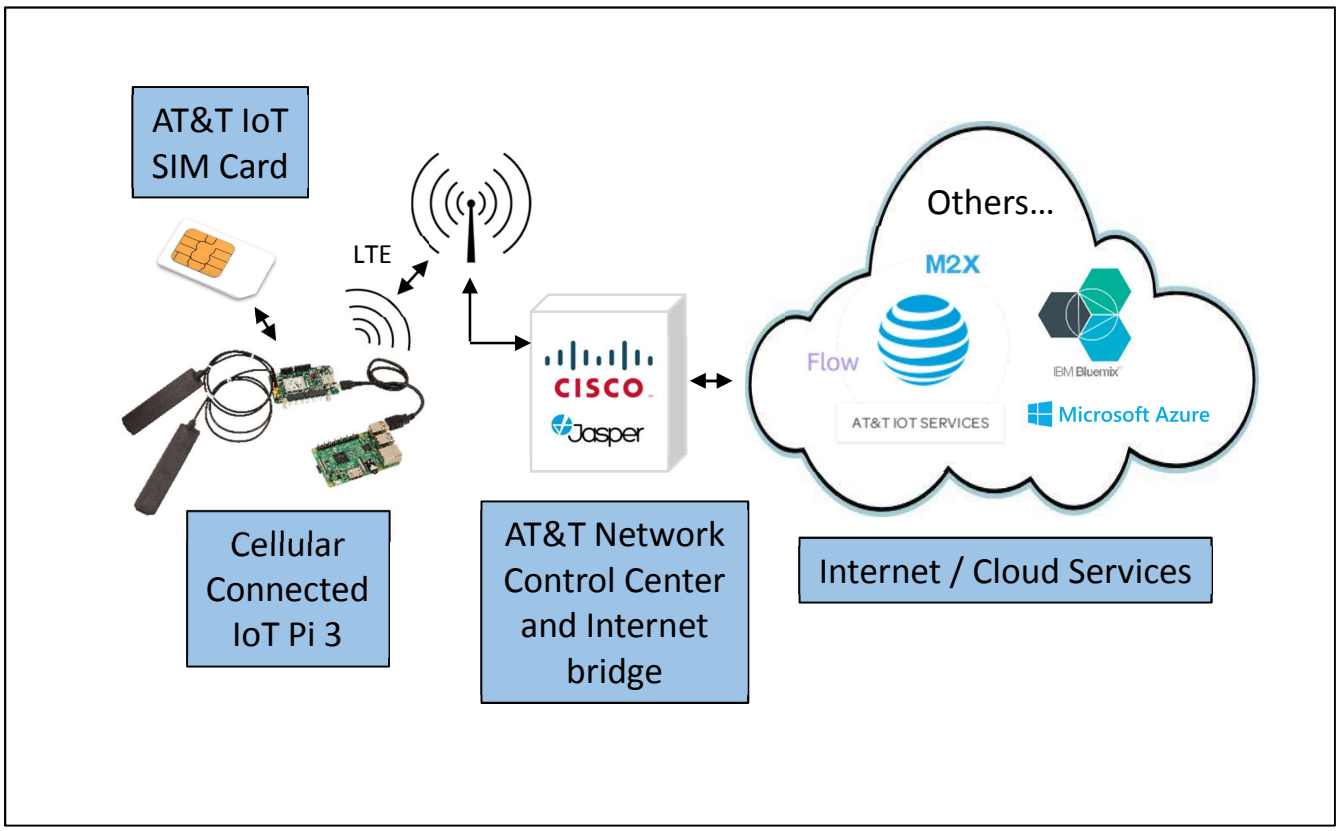
Revision History

Document Version:	Version 1.3		
Document Date:	10 December 2016		
Prior Version History:	Version:	Date:	Comment:
	1.0	2016/12/11	Initial Release
	1.1	2017/1/24	Added how to make route persistent, fixed a few typos, more info on ping.
	1.2	2017/2/24	Added fix for broken persistent route add (see Apendix G)
	1.3	2017/6/6	Changed logos, added multiple Pi Sense Hat demo, corrected typos, added note to upgrade firmware of 14A2A, re-branding
Comments:			

1 Introduction

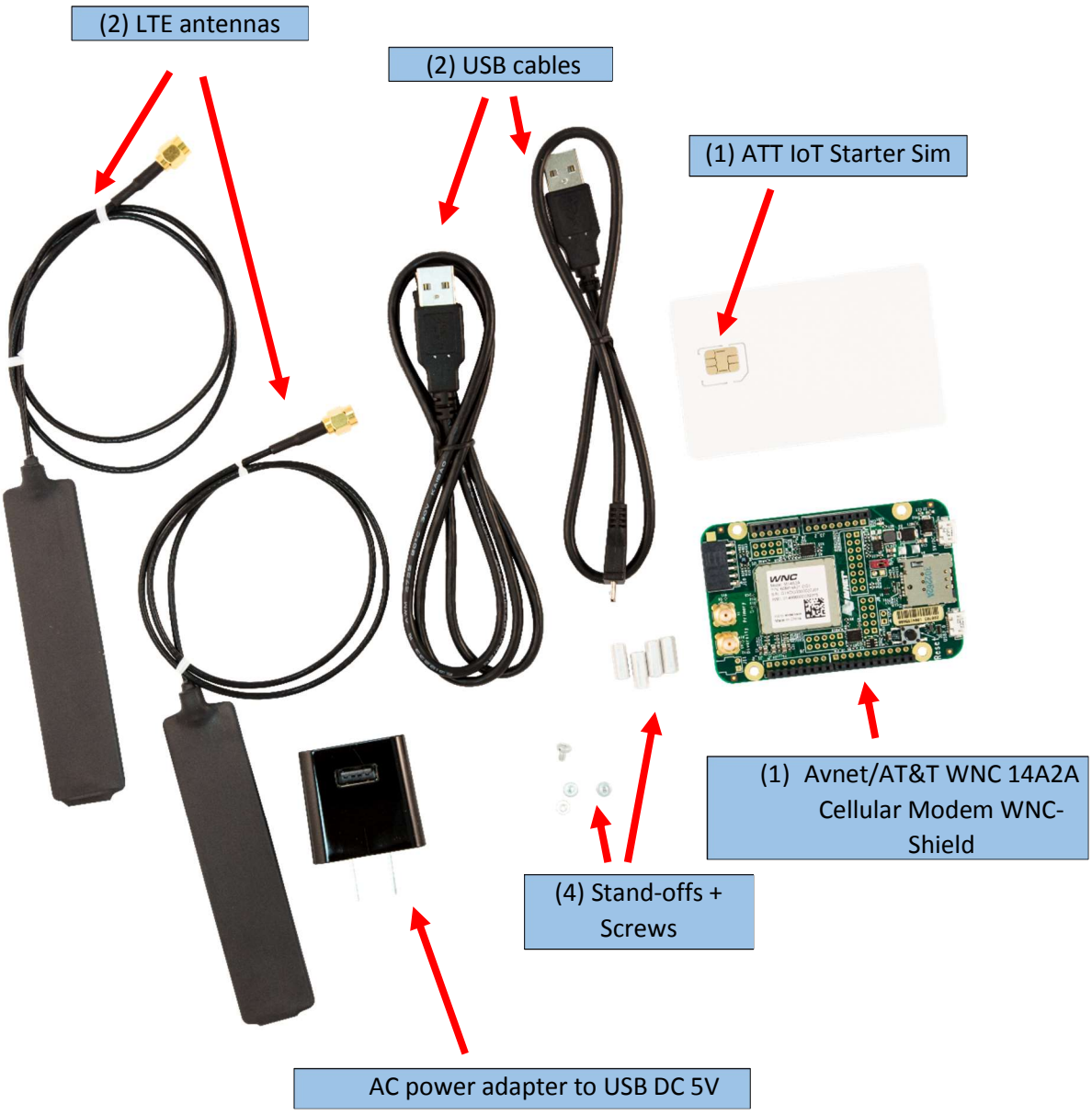
This guide provides an overview of how to setup the Avnet AT&T LTE IoT add-on cellular kit for use with a Raspberry Pi 3. This guide includes steps that allow you to quickly setup and begin using the kit. The example software application instructions will show you how to send data to both AT&T's M2X/Flow and IBM's Bluemix IoT cloud services.

The LTE cellular modem board is compatible with a Raspberry Pi 3 when running the Jessie Raspbian operating system. Other OS's may work but are not guaranteed. Utilizing the LTE modem board a Raspberry Pi 3 can send internet data via AT&T's cellular data link. A system data flow is shown below.



1.1 Kit Contents

The kit contains all of the elements needed to implement a complete cellular data solution for the Raspberry Pi 3. It includes all components shown below:



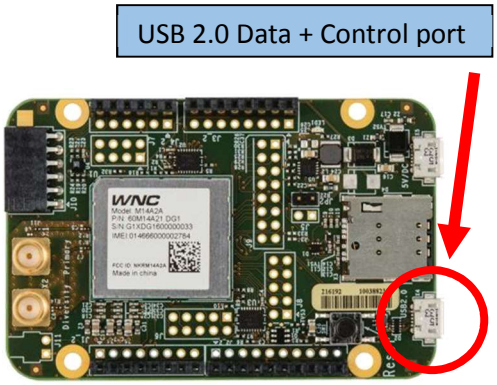
Kit contents:

Note: Raspberry Pi 3 not included

- (1) Avnet/AT&T WNC 14A2A cellular modem board
- (2) LTE Cellular Antennas with SMA RF connectors
- (2) USB micro cables
- (1) AC to USB 5V Adapter
- (4) Stand-offs and screws
- (1) AT&T 300MB IoT SIM Starter micro SIM
- (1) Quick Start Card
- Example designs and tutorials
- Forum support via: http://cloudconnectkits.org/forum/topic/att-cellular-iot-starter-kit

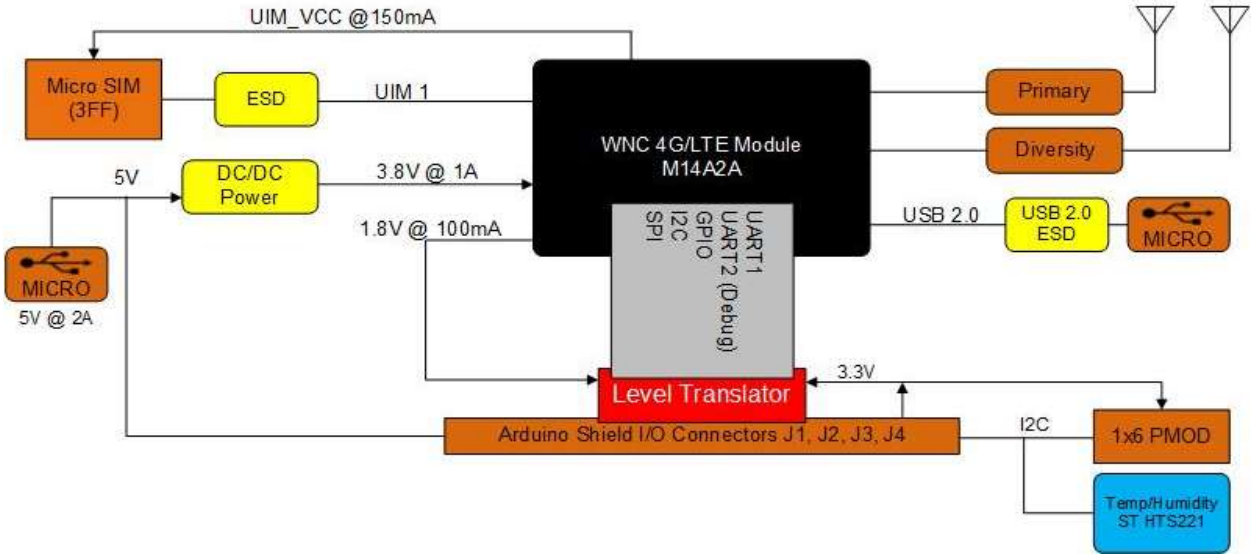
If any parts are missing please contact the distributor that the kit was purchased through or your local Avnet FAE office.

1.2 LTE IoT Cellular Board for Raspberry Pi 3



Avnet in partnership with AT&T has produced a board that utilizes the Wistron NeWeb Corporation (WNC) M14A2A LTE Cat 1 cellular module. LTE Cat 1 is theoretically capable of 10Mbit download and 5Mbit upload speeds. This module also contains all of the internet protocol (IP) stack functionality needed to establish and maintain a data connection to the Internet via an AT&T LTE cellular network. When this board is plugged into a Pi 3 USB port, the Pi's internet data can be routed through it. The WNC cellular module may also be controlled and pass internet data through a USB 2.0 port (indicated to the right, with circle). When the LTE modem board is plugged into a host platform such as the Raspberry Pi 3 running Jessie Raspbian, the Pi with will automatically activate several USB-ACM device drivers. Both a USB Ethernet device and a USB Serial Port will be enabled once the LTE modem board powers up and the Pi enumerates the devices. Pi IP network traffic can directly be routed through the LTE modem's USB Ethernet device. USB serial AT commands may also be used either exclusively or simultaneously to control the modem with AT commands and send ASCII formatted IP data through the 14A2A. The serial port AT commands are referred to by WNC as a Type III interface which supports many 3GPP and proprietary defined AT commands. See **Appendix B** for more information about where to learn more about the format of the AT commands. The LTE modem also supports sending IoT SMS messages to other IoT devices like itself or to AT&T Jasper servers; see **Appendix C** for more information.

The hardware block diagram for the LTE modem board is shown below:



Note: The HTS221 sensor is not available directly to the Raspberry Pi 3 from the USB 2.0 connection. See the schematics for wiring details.

2 Setting up the Raspberry Pi 3

You will need the following to setup your Pi 3:

- Raspberry Pi 3 and power supply (not included)
- Jessie Raspberry with Pixel Operating System (requires an internet connection and host computer to download)
- 16gb micro SD card (not included, recommend 32gb)
- Micro SD card reader/writer to transfer the Raspbian OS onto (not included)
- An available Pi USB port (the Pi 3 has 4 USB ports)
- A connection to the internet via an already established wired or wireless LAN.
- USB Mouse and Keyboard that is compatible with Pi 3 (not included)
- HDMI cable for the Pi 3 video output (not included)
- HDTV or Monitor with HDMI connection (not included)
- Recommended but not required or included: obtain a Pi 3 case to protect it. Many types of cases exist, choose one suitable to your budget
- If you have never setup your Pi 3 you will need an additional host computer to download and read the installation instructions, Windows, MAC or Linux are supported

2.1 How to setup a new Pi 3 that doesn't yet have the Raspbian Operating System (OS) installed

Please see **Appendix F** for instructions on how to setup a new Pi that does not have the Jessie Raspbian OS installed.

2.2 For a Pi 3 already running the Jessie Raspbian OS and already connected to the Internet (follow these steps as a continuation from setting up a new Pi – Appendix F)

1. Login, open a terminal window  and enter the following commands:

> Notes: answer yes/ok for any questions asked during apt-get upgrade, it is normal for some packages to not be updated. During the install you may also see a ':', just press ENTER repeatedly until you are given a press, y to quit, then press y.
>
> The upgrade command can take a while even with a fast internet connection!
>
> Another complication may arise, sometimes the updates will override some of the prior configuration settings, upon reboot a dialog will make clear what it did. At worst for a new Pi setup you might have to re-setup the locales, keyboard and time zone again.

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install minicom lsof

2. You should now reboot the Pi 3 by entering in the terminal command window:

   reboot

3. Wait for the Pi 3 to reboot and restart the desktop.
4. As long as the Raspberry Pi 3 is running Jessie Raspbian, there is no additional software required to use the LTE modem board to send internet IP data over the cellular network. Your Pi 3 should now be ready to use the LTE modem board!

3 Creating Online Accounts

In order to use the Pi 3 LTE modem board, several online accounts are required:

- Cloudconnectkits.org account for kit registration
- AT&T IoT Platform account with access to Jasper for SIM card registration and usage tracking

The following are optional depending upon which example application you want to experiment with:

- AT&T M2X account, used for the AT&T M2X Tutorials, see Appendix A
- If you would like to run your own M2X/Flow programs, a AT&T Flow account is required, see **Appendix A**

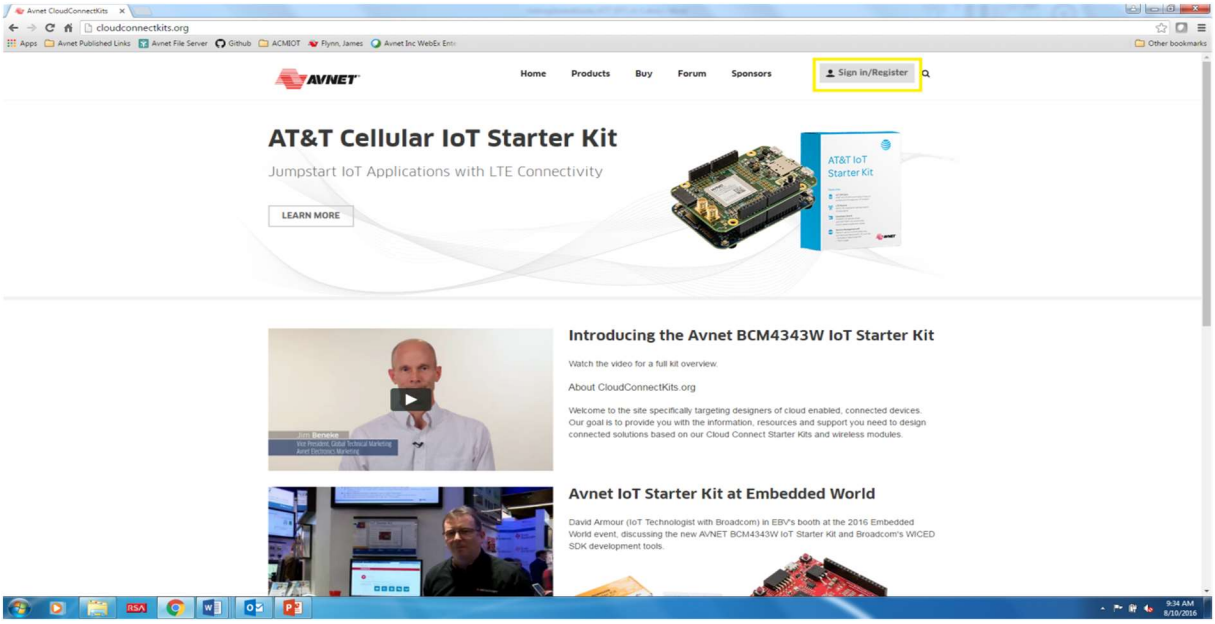
An AT&T Flow/M2X is not needed to experiment with the Virtual Pi Hat Sensor data demo. There is also an example IBM Bluemix application detailed later but it does not require an account.

Note: You may use the Pi 3's included Web browser or create the new accounts from another computer.

3.1 Cloudconnectkits.org Account

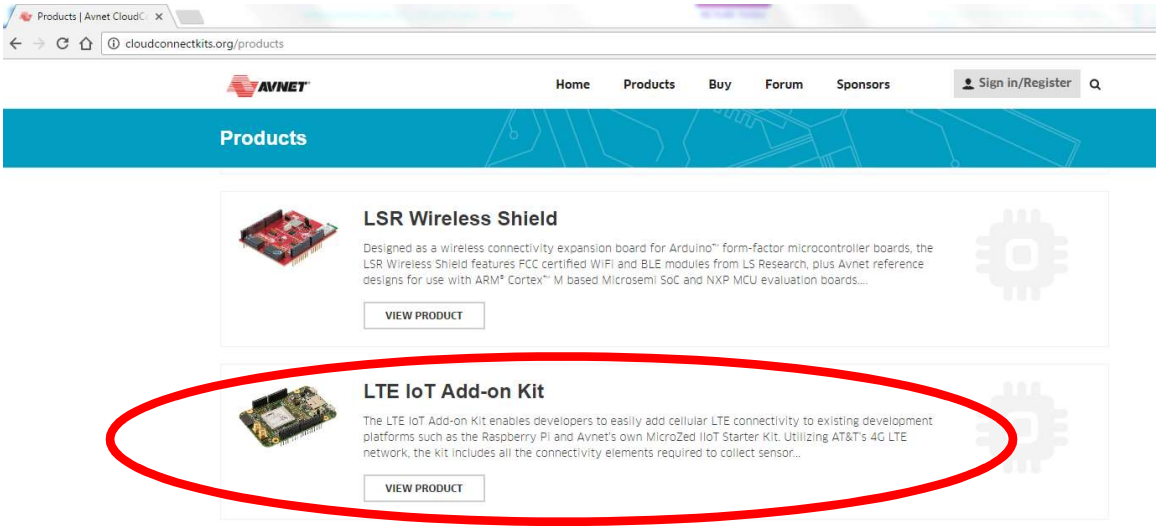
Register your kit at www.cloudconnectkits.org to be notified of firmware and software updates as well as other information.

1. Navigate to www.cloudconnectkits.org
2. Press the Sign In/Register button. A drop-down appears from which you can create an account.



3. When asked for the Serial Kit Number, use the WNC Serial Number (S/N) as shown below.

4. Once registered, you are logged into the site. A confirmation email is sent to the email address you provided. You will now have access to several product support pages as shown.



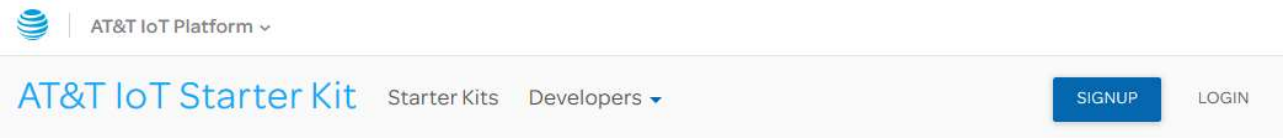
From the LTE IoT Add-on Kit page, you can view the kit's design documentation, bill of materials, and other information.

3.2 Create an AT&T IoT Platform account and Register the SIM Card

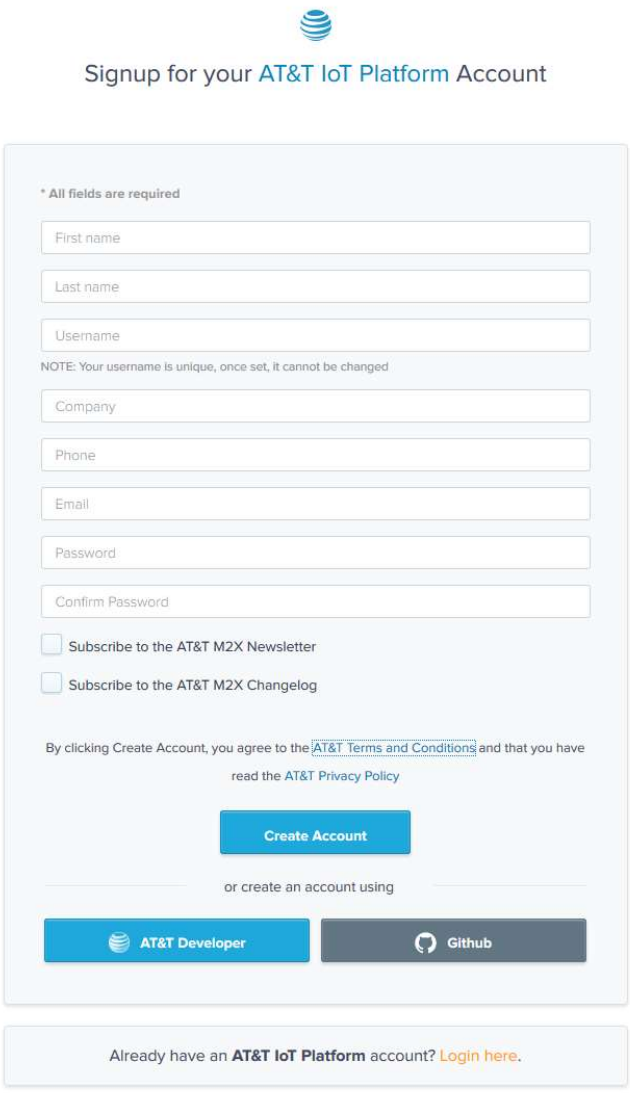
Locate the SIM card that came with the kit, you will also need the ICCID numbers typed on the card. Proceed to:

https://starterkit.att.com

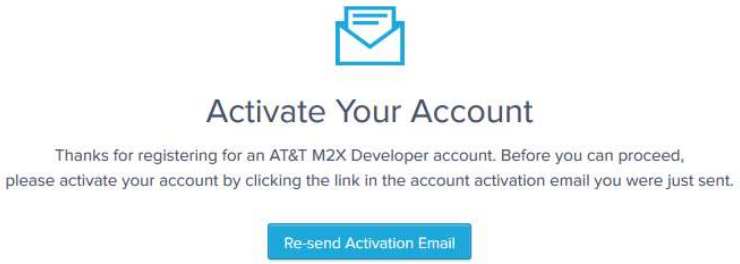
Select **SIGNUP** if this is your 1st SIM card registration for an IoT account or select LOGIN if you already have an existing account.



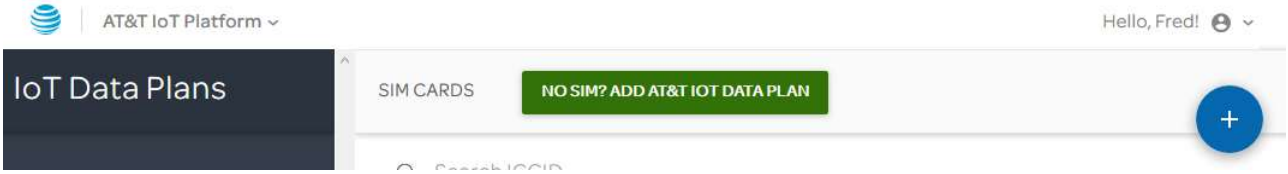
A new **SIGNUP** will require your name, email and some other contact information:



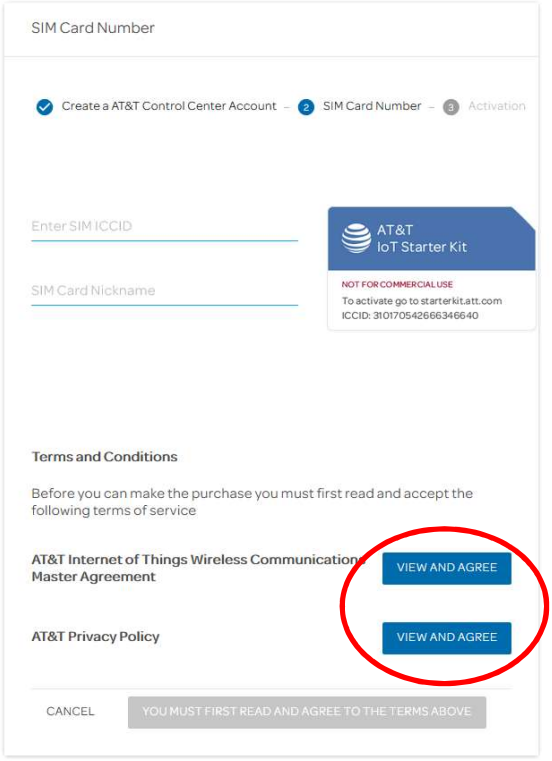
Fill out the form and follow the prompts until your Starter Kit account has been created. Next you should see:



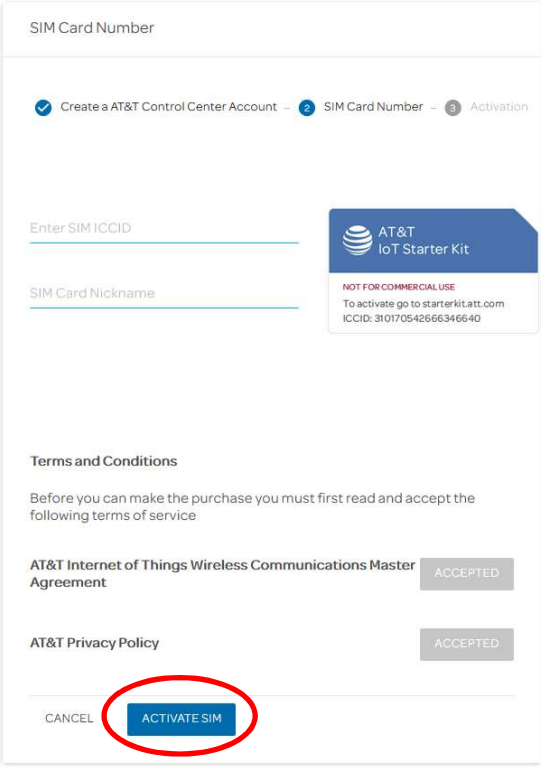
You will need to find the email that was sent, then click the link to verify the new account. Once verified, next the SIM must be registered. Login to your verified existing or new Starter Kit IoT Platform and Jasper account, click the + symbol:



Select "**Activate SIM**" and then proceed to fill out the forms with the ICCID number and agree to the Terms and Conditions. You may also give a custom name to your SIM card. The ICCID number is printed on the card and should be either 19 or 20 digits long. See the example photo below:



After agreeing to both you should see an **Activate SIM button**, click to finish the registration:



Once registered, the IoT Platform account should quickly enable your LTE modem board to access the provided 300 MB of data and 300 SMS messages for up to 6 months of usage. Once either the data or time expires, you may purchase more *pre-paid* data through your AT&T Jasper/IoT Platform Account.

Note: The SIM card included in the kit is an IoT SIM card and will not work in a phone. Also, a standard AT&T SIM card for a phone will not work in the LTE modem board. The IoT SIM card included in the kit will currently only work in the United States on AT&T and partner towers and also wholly owned AT&T towers in Mexico.

Note: If you have a larger scale project in mind you should contact AT&T directly to possibly obtain a different arrangement. Currently the IoT Platform account supports up to 1000 SIM cards in one account.

4 Connecting the LTE modem board to the Pi 3

4.1 Prep work

CAUTION: There are several header pins extending from the LTE modem board, you do not want to short out any of these pins while using the board. Keep static electricity away from the LTE Modem board!

1. Find a suitable location to physically place the LTE modem board, LTE antennas, cables, power-supply and Raspberry Pi 3.
2. Currently a hard case does not exist for the LTE modem board. You may want to consider making your own if you have access to a 3D printer or install the included stand-offs that are tall enough to lift the modem board so that it does not rest on its pins.
3. Locate the items from the kit's contents including: SIM Card, Modem board, USB Power Supply, 2 USB Micro cables and 2 LTE Antennas.

4.2 IMPORTANT: perform firmware upgrade of LTE modem board

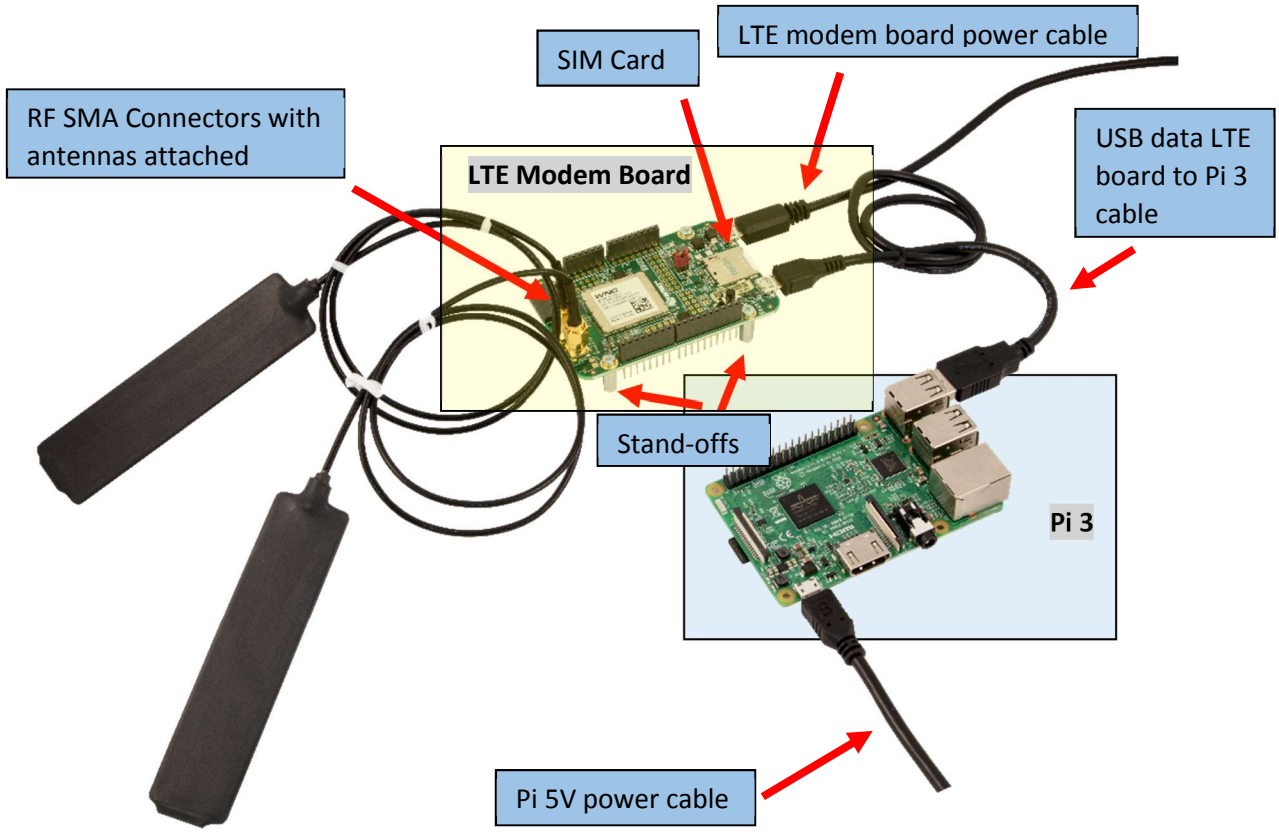
Before using the LTE modem board extensively, it is highly recommended that the M14A2A firmware be upgraded to the latest. The first version of release firmware that comes programmed on the board has a bug that results in the modem randomly going to sleep.

Please follow the instructions here to upgrade the firmware:

https://starterkit.att.com/tutorials/cellular-shield-firmware-upgrade

4.3 Kit assembly of Pi with LTE modem board

First study the following picture that shows the assembled kit and then follow the instructions to assemble it yourself:



Note: please refer to the Figure below for connector label references.

1. If you do not have a way to mount the LTE modem board so that the pins are not touching the resting surface, please attach the included stand-offs (see photo above).
2. Assuming you have registered the SIM card as explained in **section 3.2**, separate the micro SIM from the card and then insert the SIM into the SIM socket on the LTE modem board (**S1** in the Figure). The SIM will only go in one way, it is a push to insert and push to release type of socket.
3. Locate the kit's pair of LTE Antennas and attach both to the RF SMA Connectors on the LTE modem board (**X1** and **X2**). Screw the antenna connectors on, for an optimum RF seal an SMA torque wrench should be used, if one is not available just finger tighten the connectors.
4. Locate and attach 1 of the included USB Micro cables to the USB Data connector (**U2** in the Figure) on the LTE modem board but do not yet connect the other end to the Pi 3.
5. Locate the kit's included AC to 5V USB power supply and 2nd micro USB cable.
6. Plug the 2nd USB Micro cable into the connector marked 5V USB (**U1** in the Figure) and the other end into the kit's USB power supply.
7. Double check all connections, once assembled it should look like the diagram on the previous page.

8. Next plug in the power supply and wait for about 15 seconds.
9. Finally, plug the other end of the micro USB that was connected to the LTE modem's USB data port (**U2**) into any available Pi 3 USB port.

4.4 Setup the Raspbian OS to use the LTE modem board for data

Login into your already setup Pi 3 using the GUI (or SSH login see **Appendix E**). Only a command line is needed for the following steps, the GUI is not necessary but recommended for ease of setup.

1. Open up a Pi terminal window via SSH or the GUI  , then enter the following:

   minicom –b 115200 –D /dev/ttyACM0

2. This should change the terminal window into a minicom window.

3. While in minicom, issue the following command by typing the case-insensitive letters and hitting enter:

   AT

> Note: For any type of serial command entered, a successful response from the cellular modem is OK. If you do not receive any response or an ERROR, double check your syntax and then see the **Appendix D**, Troubleshooting.

4.

IMPORTANT: We need to set the proper AT&T cellular network APN. Enter the following command and you should also see OK as a successful response:

AT%PDNSET=1,m2m.com.attz,IP

5. All done with APN setup, so exit the minicom window type: ctrl-A then 'X', hit enter when Yes is selected.
6. Power down the cellular shield and power it back up… About 15 seconds after turning on power you should now be able to begin using the LTE cellular modem for Pi Internet data.
7. Assuming you only have a stock Pi 3, the LTE modem board should be enumerated as /dev/eth1. If you have additional network adapters, the path may differ. Adjust the following instructions accordingly.
8. From a Pi terminal window, SSH or GUI  , enter:

ifconfig eth1

9. Look at the response, the **inet addr:**, it should not be 10.0.0.x (.x here means x is any number from 0 – 255), if it is, the cell shield is not connected to the cellular network for some reason. Make sure the SIM is registered and properly inserted.

10. If the LTE modem's IP address is good (meaning not 10.0.0.x) from a Pi terminal, enter:

ping –I eth1 8.8.8.8

11. This should return ping times shown from passing ping data through the LTE modem board:

```
pi@raspberrypi:~ $ ping -I wlan0 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 10.0.0.123 wlan0: 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=3.20 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=5.92 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=4.91 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=5.09 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 3.202/4.784/5.928/0.991 ms
pi@raspberrypi:~ $
```

For more information about ping see end of **Appendix F**. To stop ping press <CTRL>-c

12. If ping doesn't work, we're going to have to do some troubleshooting, please see **Appendix D**.

4.5 Using the LTE modem data connection for your Pi data

Note: this section assumes you have a stock network connection setup, meaning no additional network adapters besides the Pi 3's built in Bluetooth, WiFi and hardwire Ethernet. Therefore, when connecting the modem it should create a /dev/eth1.

1. If everything is working, the Pi 3 should still be using your LAN connection for all of its Internet traffic.
2. Only applications or programs that can select alternative connections will be able to use the LTE modem. ping is one example.
3. If you want to have the Pi 3 use the LTE modem for all internet traffic, enter the following command in a Pi Terminal window:

   sudo route add default eth1

   CAUTION: All Pi data will now be running through your metered LTE modem board! Remember you only have 300MB to start with!

4. To undo the default route being set to eth1, enter the following command:

   sudo route delete default

5. The route command as issued will not stay in effect if the Pi is power cycled or rebooted. Please see **Appendix G** for how to make the routing become persistent.

5 Example Applications

This section assumes that the Pi 3 is connected and operational with the LTE modem board.

CAUTION: You should setup your Pi to first use your LAN connection to install and run the Example demos for the 1st time! That is unless you want to use up your limited AT&T data. You probably do not want to have your LTE modem board setup to carry all of the Pi's Internet data. By default it is not, see **section 4.4** to learn about how to use route to enable and disable data traffic through the LTE modem board. After getting the demos to work through your LAN, then you should use the route command to enable the application to use the LTE modem for sending small amounts of data.

Note: you can track your LTE cellular data usage by logging into your AT&T IoT Platform/Jasper account!

5.1 Raspberry Pi Hat Sensor Emulator AT&T Flow Demo

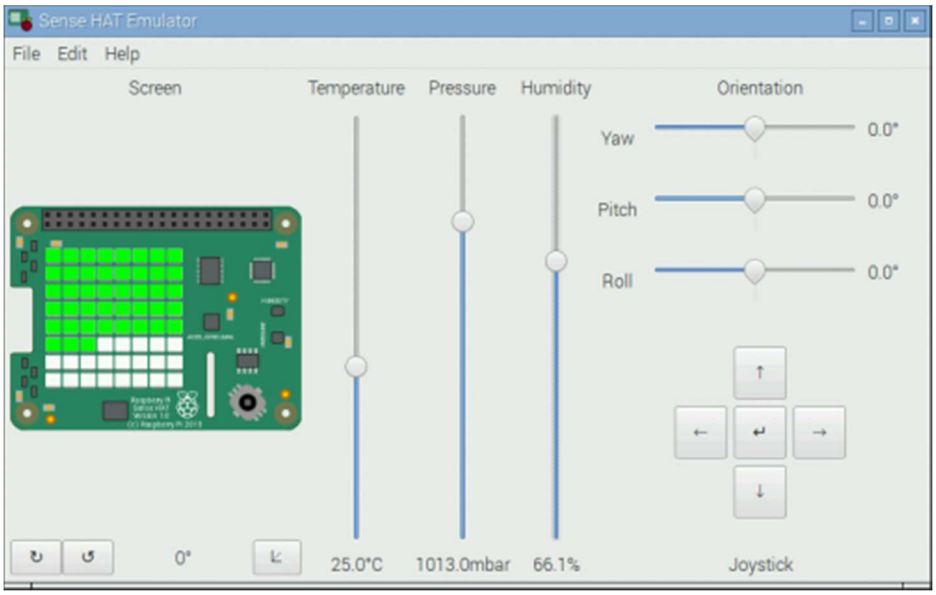
Before attempting to run this demo, you must setup an AT&T M2X and Flow account. See Appendix A for how to do this. When finished come back here and proceed.

5.1.1 Overview

Without additional hardware, the Raspberry Pi 3 does not have any on board sensors. This can easily be worked around by using a program that comes with the Jessie Raspbian Pixel Operating System. For this demo we will utilize the Raspbian GUI Desktop Sense Hat Emulator.

Note: you can find out even more information about the virtual sensor hat here:

https://www.raspberrypi.org/blog/desktop-sense-hat-emulator/



To start the virtual sensor program, click on the Raspberry start button  and then select Programming, then Sense HAT Emulator. Click the latter and a virtual sensor program should open. This program is written in Python and we will be using a Python program to read the virtual IoT sensor data and send that to a program running on the AT&T Flow cloud servers.

The included virtual sensors are: temperature, pressure, humidity and orientation (yaw, pitch and roll) and a joystick. There is also an array of 8x8 simulated RGB LEDs available.

For this demo we will utilize the temperature, humidity and orientation sensors and send the virtual readings to the AT&T Flow servers that are running a program that will compute a color string and send it back to the Pi. From these sensor values, the Flow program will compute a color and send that color value back to the virtual sensor board which we will then set the color of the virtual RGB LEDs to.

Note: The sensor data is delivered to the Flow server by using the http protocol to send a JSON formatted string. The reply back from the flow server is also read by using the http protocol to read back the JSON formatted color string. For more information inspect the attsensorflow.py Python script.

5.1.2 Installing the virtual sensor Python program

Open a Pi Terminal window, you may use the GUI or SSH. When opening a new window you will be in the home directory for your user name. Create a new directory and change directory into it by entering:

```
mkdir iotvirtualpi

cd iotvirtualpi
```

Now use the wget command to pull down the following program source code from Github, enter:

wget https://raw.githubusercontent.com/Avnet/raspberrypi/master/iotvirtualpi/attsensorflow.py

You should now have a new file, attsensorflow.py located under **/home/pi/iotvirtualpi** (or wherever you chose to put it). Leave the terminal window open for the next step.

5.1.3 Run the virtual sensor Pi demo

> Note: to run this demo you do not need to create a Flow or M2X account. However if you would like to modify or develop your own Flow application you must obtain a Flow and M2X account as described in **section 4.2** and **Appendix A**. **Appendix A** also explains how to fork the example program to your own account.

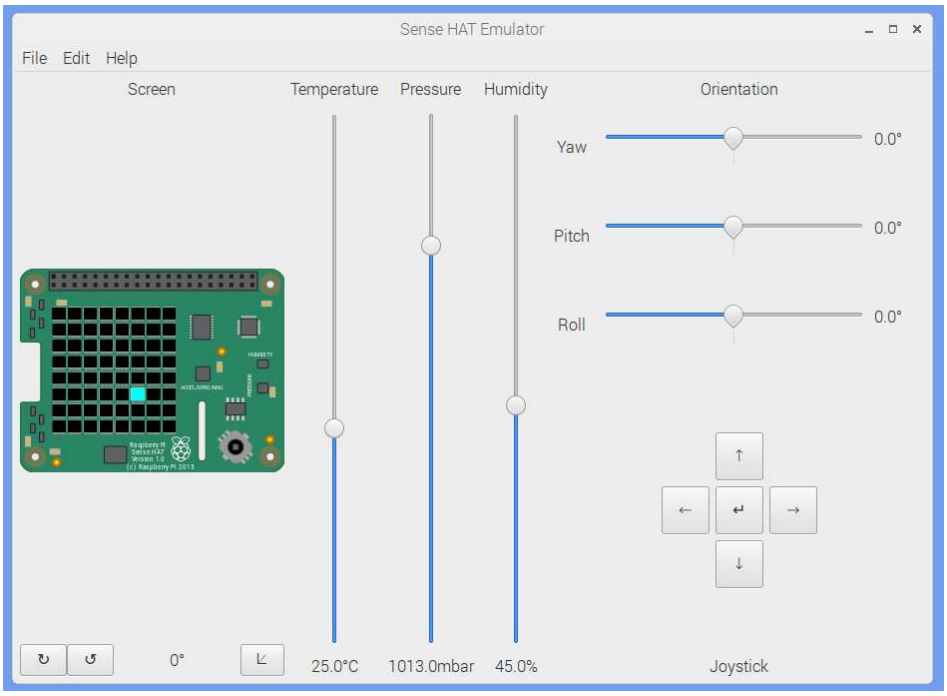
Run the Python script by entering in a Pi terminal:

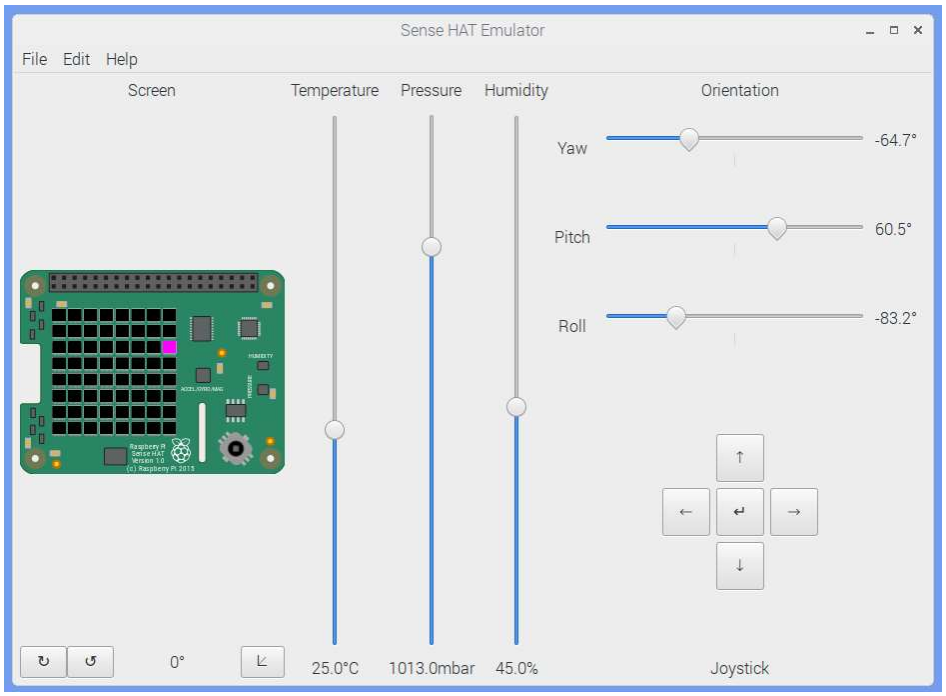
python attsensorflow.py

It should run without any errors. If you do see an error, double check your network connectivity. Occasionally the AT&T Flow server is under maintenance or can become busy, in which case the program will print out some kind of error status. When the Flow server is up and operating correctly, the response status should be "Accepted".

Now go back to the Pi and the Sense HAT Emulator GUI program. Adjust the temperature, humidity or yaw/pitch/roll by moving the sliders around. Moving the sliders will simulate changing values which are being sent to the ATT Flow server. You can see the string sent to the server as the Python program runs. The Flow server will read the sensor data and compute a color string that will be sent back to the Pi. The Python script takes the computed color string and sets the virtual Pi Sense Hat Emulator LEDs to that color. See the following examples showing the LED color in response to the different sensor data.

> Note: if you see a stuck LED that just means the Flow server did not respond for some reason. This will occasionally happen due to the server becoming busy or overloaded. Eventually the moving LED will cycle back and turn it off as long as future replies are received.





Note: it takes a little bit of time for the data to make it to the server and back. The rate at which the lit LED moves across the array tells you how much time it takes to read the sensors, send it to Flow and receive the reply.

5.1.4 Changing the Python program to interact with your private Flow program

Once you have forked the original Flow program to your own account **(see Appendix A, Section 6.3)**, if you would like to have your Pi interact with it rather than the public Flow program, you will have to modify the following parameters in attsensorflow.py:

FLOW_SERVER_URL

FLOW_BASE_URL

5.1.4.1 FLOW_SERVER_URL Modifications

The original attsensorflow.py scrip sets the server URL to the following value:

FLOW_SERVER_URL = "run-west.att.io"

Use a text editor to modify attsensorflow.py to use your own appropriate values. If you do not have a favorite editor, consider using Idle that comes with the Pi. Idle can be accessed from the Raspberry Pi GUI, start icon  , then programming tools.

You will need to change FLOW_SERVER_URL to match the values for your own version of the Flow program. See **Appendix A, section 6.3.2 step 6**. You will want to take the server URL portion that is **not** highlighted in green from there (see step 6) and set the FLOW_SERVER_URL variable to this new value.

5.1.4.2 FLOW_BASE_URL Modifications

The original attsensorflow.py script sets the base URL to the following value:

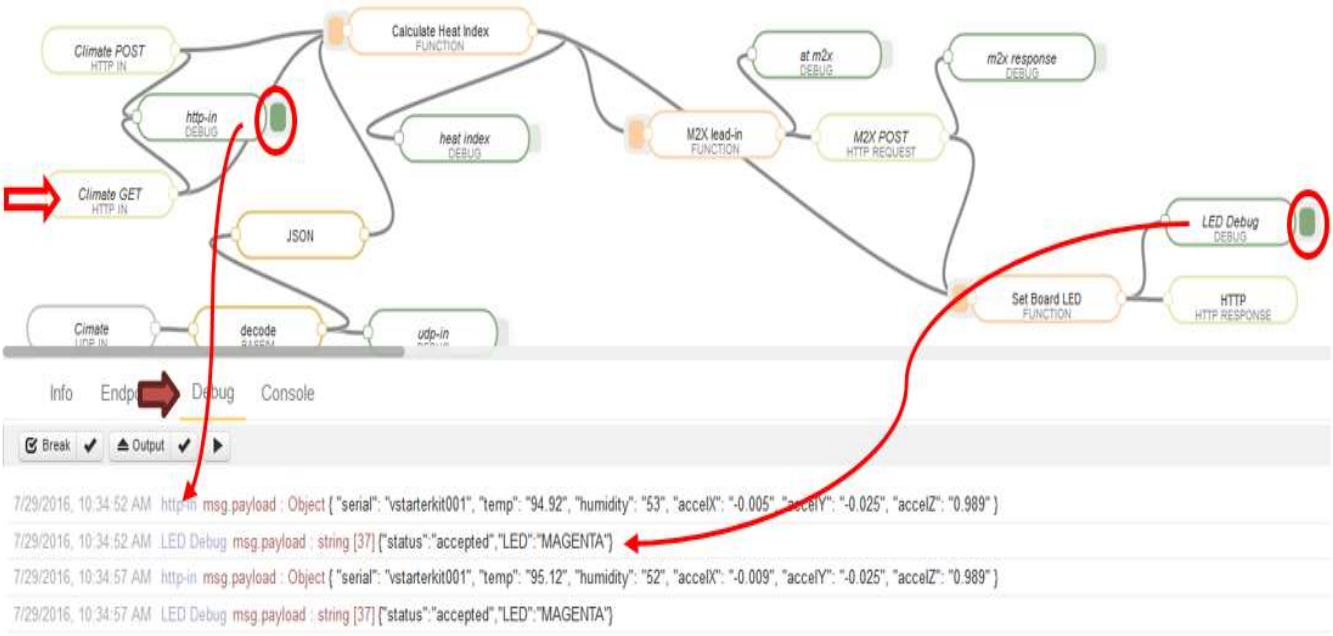
FLOW_BASE_URL = "/1e464b19cdcde/774c88d68202/86694923d5bf28a/in/flow"

Use a text editor to modify attsensorflow.py. You will need to change FLOW_BASE_URL to match the values for your own version of the Flow program. See **Appendix A, section 6.3.2 step 6**. You will want to take the base URL portion that is highlighted in green (see step 6) and set the FLOW_BASE_URL to this new value.

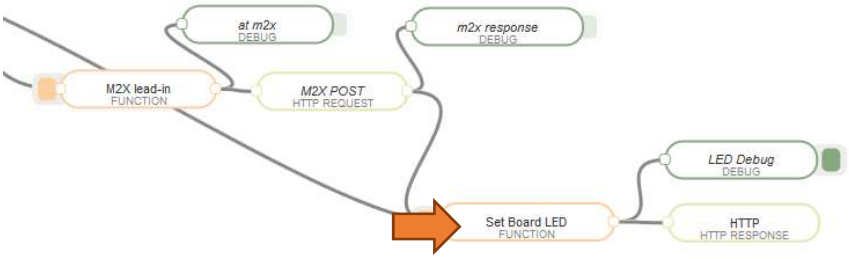
5.1.4.3 Exploring and enhancing Flow on the server side

If you are curious or it is not working, you can see more about what is going on in the server side of things by logging into your AT&T Flow account, connecting it to M2X and forking the baseline project. See **Appendix A** for how to do this.

Flow/M2X Operation: To verify messages from the board are arriving in Flow, go to the Flow project and click the **Debug tab** at the bottom of the canvas. Because the board is doing an HTTP GET, data currently enters the Flow design through the bottom HTTP IN port. Messages with sensor information start appearing. There are debug messages from both the input and output debug ports. You can enable/disable these debug messages by clicking the solid part to the right of the port.



Unsolicited messages cannot currently be sent to the board. A **response** to the HTTP GET message and the subsequent M2X posting, can however be created. On the Flow canvas (Data tab), double-click the **Set Board LED** function. Examine how the decision for setting the LED color is made.



5.1.5 Send virtual Sense Hat Emulator data through the LTE modem board

The next step would be to have this data routed through your LTE modem board instead of your LAN. If you would like to do this, back in the Pi Terminal window, stop the attsensorflow.py program from running by pressing <CTRL>-c. See **section 4.4** for how to route the network data through the LTE board. Restart attsensorflow.py by using the same method that you used before to run the program.

5.2 IBM Bluemix Quickstart Pi Demo

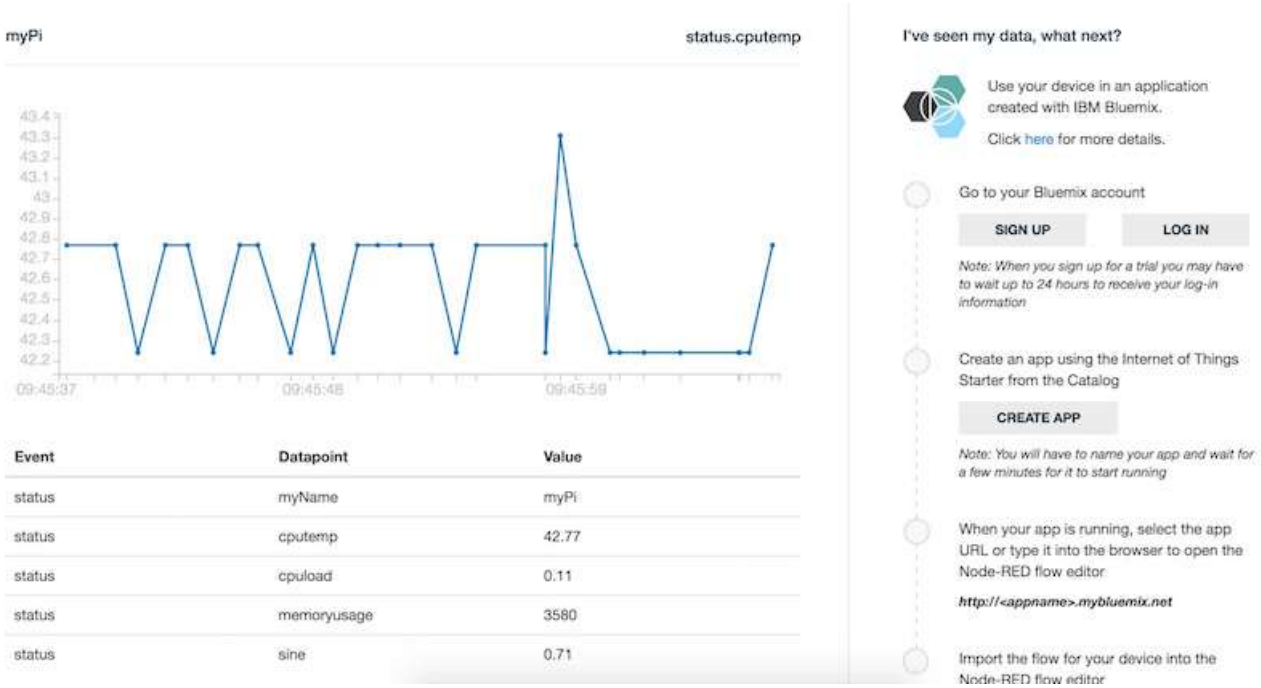
CAUTION: to save your limited AT&T cellular data you will probably want to setup and prove that it works by using your LAN networking connection first. After you have it working you can switch over to use the LTE modem board, see **section 4.4** to see how to use the route command.

The following demo will read the Pi 3's CPU parameters such as loading and temperature and send that data to the IBM Bluemix Cloud Server. The Server will then plot the data as a time series on a graph that is accessible with any Web Browser. An IBM account is not needed to run this demo.

The demo will require installing an IoT IBM Bluemix service on your Raspberry Pi 3.

> Note: before installing the services, be sure to do "sudo apt-get update upgrade" first to ensure the system is up to date.

If successful after following the instructions you should see something similar to the following on your web browser:



Open a Pi Terminal window via the GUI or SSH and follow the instructions provided by IBM found here:

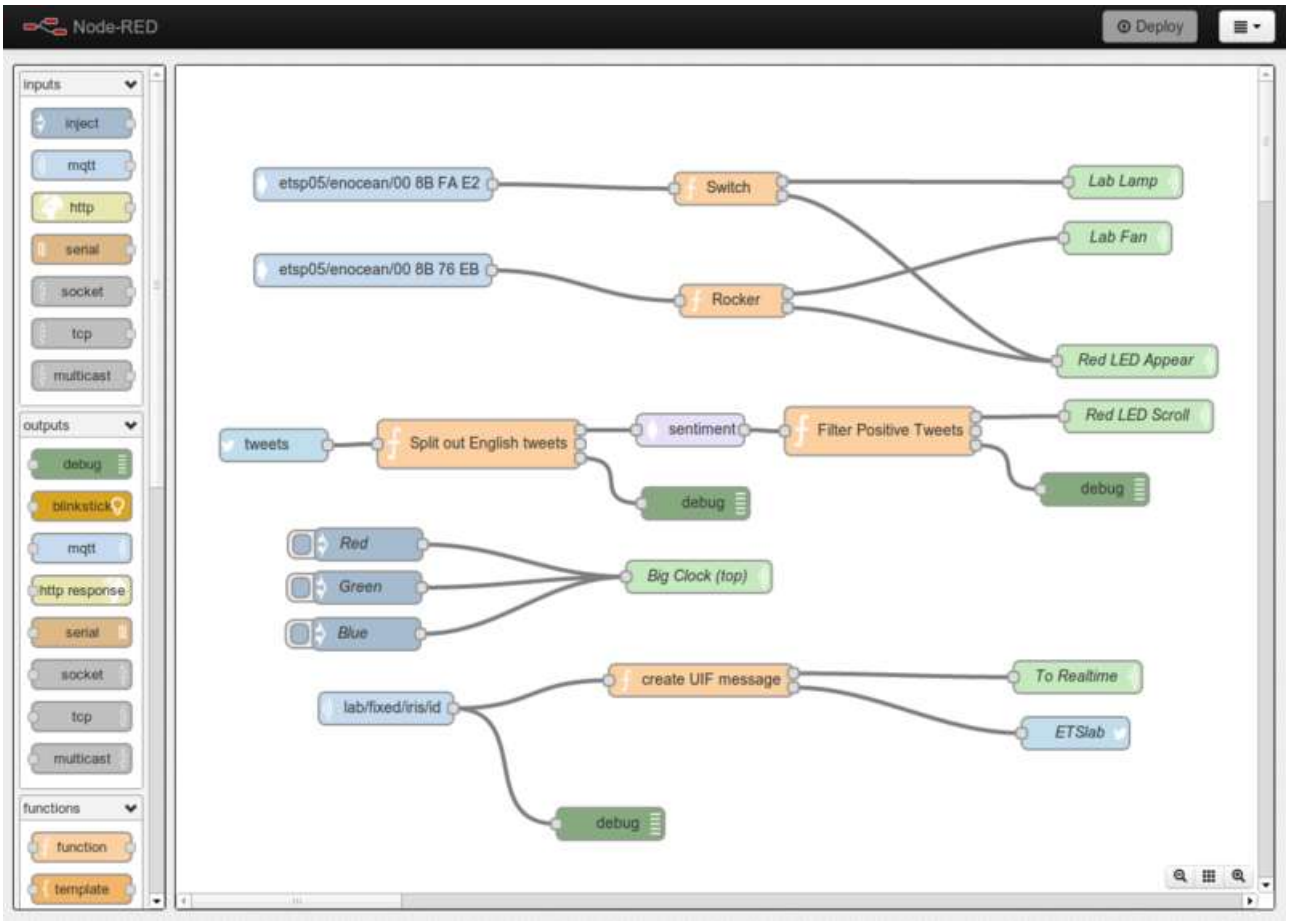
https://github.com/ibm-watson-iot/iot-raspberrypi/blob/master/samples/c/README.md

If you would like to go even further with IBM Bluemix and learn about more advanced applications, see:

https://www.sitepoint.com/connecting-a-raspberry-pi-to-ibm-watson-and-bluemix/

These additional instructions will help you obtain a free starter Bluemix account to connect your Pi 3 to, eventually it also shows how to connect to a Node Red graphical programming tool. Node Red is a graphical programming environment used for writing server side applications that can interact with your Pi and more.

Note: The following diagram is shown as an example of a Node Red program, it is not the actual one used in the tutorials.



5.3 Multiple Pi Sense Hat AT&T Flow Demo

This demo reads temperature, humidity, 3D orientation data, joystick position and LTE cellular signal strength data from up to 8 independent Pi Sense Hat hardware setups. The data is sent through the cellular data shield (or any other internet connection that the Pi has) to an AT&T Flow server. The Flow server program translates the X,Y,Z positional information into the HSL (Hue Saturation Lightness) color representation and then converts the HSL into the RGB color representation. The resulting RGB color is sent back to the Pi which sets the devices row color. Each of up to 8 Pi setups has a unique row assigned to it.

When the program starts, it first attempts to auto detect if a Sense Hat is attached to the Pi. If a real Sense Hat is not found, it will attempt to use the Python emulator that is built into Raspbian Jessie.

> Note: If you do not have a real Sense Hat you must manually start the emulator as outlined in section 5.1.1

When starting the program will also attempt to auto detect the LTE IoT Add-on Shield. If one is not found the program will attempt to use the Pi's default internet connection. If a Shield is found it will initialize it and change the route tables so that the Pi uses the cellular modem for all internet traffic while the program is running. Once the program is terminated the original route table's default internet connection will be restored. When the shield is detected the program will display cellular signal strength bars to aid in proper setup.

Install the program by copying all of the files from Github to any directory on the Pi from:

https://github.com/Avnet/raspberrypi/tree/master/senseflowdemo

Make sure all the files that have the .py ending are set for executable. This can be done by entering the following in a Terminal window: **chmod +x *.py**

Included is an html file that can be used to interact with the Pi. Messages can be sent from a web page as well as various other remote-control features. In order to use the web page, you will have to first establish our own Flow server and obtain it's URL. The Javascript in the html must be modified to use your own Flow URL. A password is utilized for security purposes. The value of the password (default: "password") is set in the Flow program itself in the Configuration node. If you want to set your own password you will have to modify the Flow program. You will have to setup a web server to host the flow.html page as well if you would like to use it.

Before running the program, you will need to setup your own Flow account to have the Pi feed data into. The instructions for how to do this are referenced below in **Lab 3**. The instructions were originally created for a class about the Pi and some assumptions have been made that will no longer be true for the perspective from which you will be starting at. They are outlined at the beginning of the following document. How to use the demo program is outlined in the following document:

http://avnet.me/LTE_IoT_Addon_sense_flow_demo_pdf

Once you have your own Flow account setup and you have modified setupflowpaths.py to point to your own Flow program you can now execute the program by entering the following in the same directory

where you placed and chmod the Python files. Enter the following in that directory from a command terminal window: **./atthatflow.py**

If a cellular shield is detected the signal strength bars will be displayed. If one or more blue bar is displayed your cellular signal strength should be good enough to proceed. Press the joystick momentarily down in the center and next a number 1 should be displayed. Use the up/down/left/right to change this number from 1-8. This number is the going to determine the ID of the Pi setup and the row of color that pertains to that Pi's position is going to be that row number counting down from the top of Sense Hat LED matrix. After selecting that number press the center button of the joystick momentarily once again. The demo should now begin to send data to your own Flow cloud.

About the source modules:

atthatflow.py – This is the main program that reads the sensor data and sends it to Flow.

bars.py – This controls the 14A2A modem on the shield through AT commands via the USB serial port.

setupflowpaths.py – This contains variables that atthatflow.py uses to determine which Flow URL to use.

flow.html – A web-page containing both HTML and Javascript that interacts with the Pi program and the Flow program. It allows for control of the Pi from the web page via the Flow server.

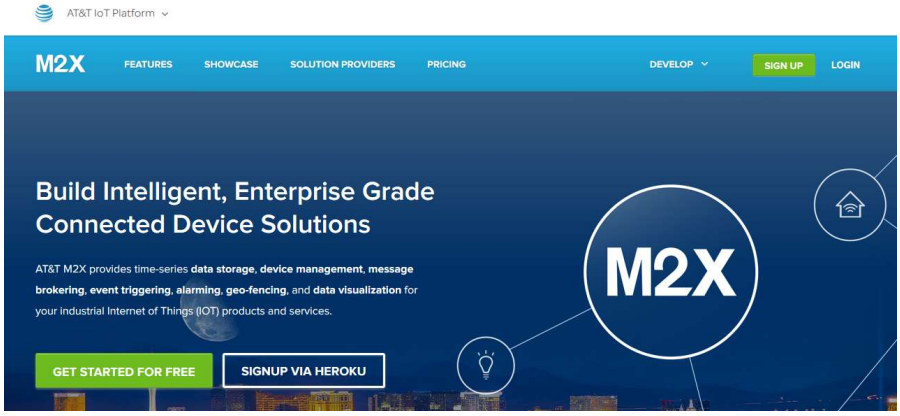
6 Appendix A: AT&T M2X and Flow

6.1 AT&T Flow and M2X Accounts

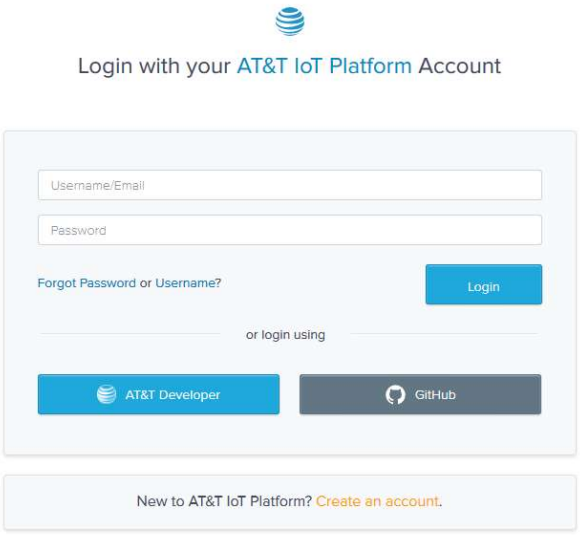
> Note: only if you are intending to use the kit to run your own version of the AT&T M2X and Flow Pi sensor examples will you need to create these accounts.

Assuming you have already registered your SIM card, you should already have an AT&T IoT Platform developer kit. You will want to link that account to the M2X and Flow accounts.

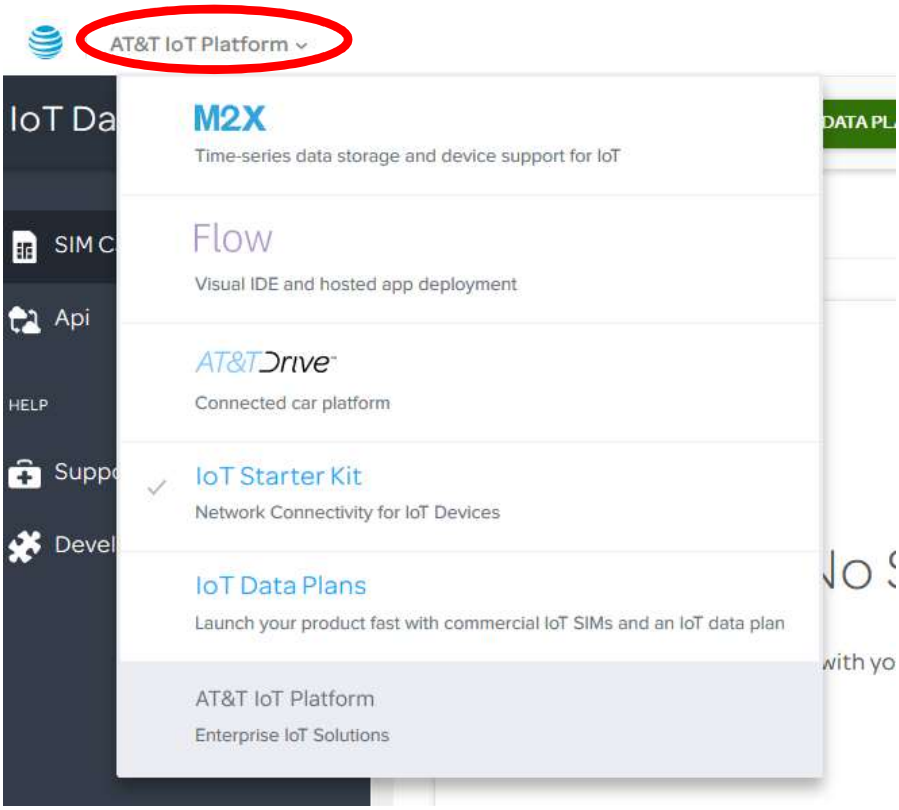
First go to https://m2x.att.com and click the **LOGIN button**:



Which will bring up the following dialog where you will need to enter your previously created IoT Platform account credentials:



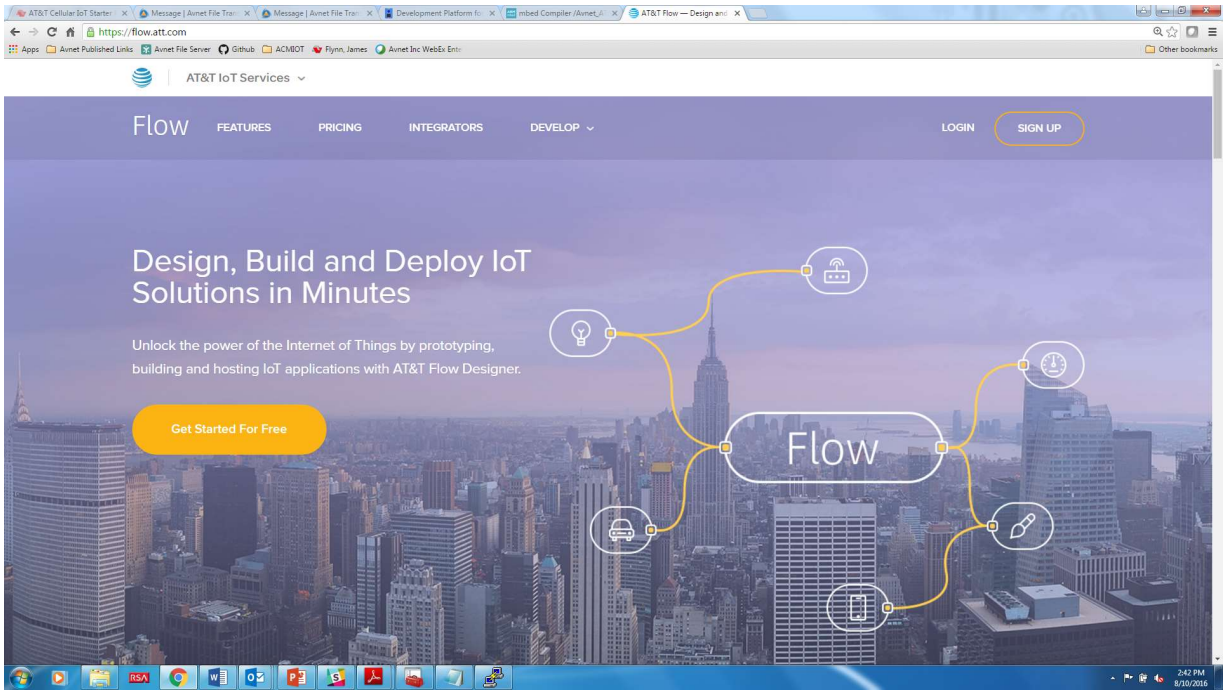
If any additional forms are presented, fill them out and agree to the terms, accepting defaults if any are presented. This will now give you access to all of the AT&T services: M2X, Flow, AT&T Drive, IoT Starter Kit and IoT Data Plans. So if you would like to login to your M2X or Flow account, you may go to their sign-in pages and always use your IoT Platform account credentials. You can also from within the IoT Platform web page select the other services. This can be done by selecting the AT&T IoT Platform pull-down.



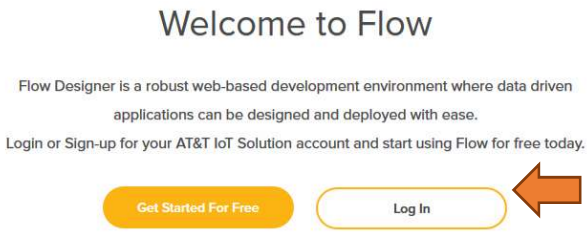
6.2 More about AT&T Flow and setup for the Raspberry Pi 3 sensor demo

To setup the AT&T Flow cloud services to interact with the Raspberry Pi for the sensor demo, either go to Flow directly at https://flow.att.com and use your IoT Platform credentials or go to Flow from the pull-down in your IoT Platform account as explained in Appendix A.

Flow is a GUI-based IoT development tool based on NodeJS. It allows for data input and output along with user defined rules for up to 100,000 data points per month free of charge. Data *flows* through nodes from left to right when viewing a Flow. You can learn even more about Flow by visiting: https://flow.att.com/start



Note: Unlike M2X, Flow times out after a few minutes of inactivity. When this happens, simply click the **Log In** button to log back in. Even when you have a fast connection, Flow is slow to respond.



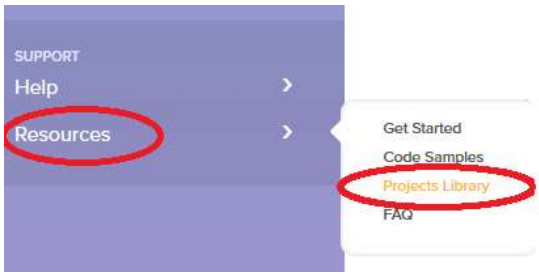
If you want to learn by creating your own Flow design from the ground up, you may want to follow AT&T's instructions that were created for a recent event at: https://att.app.box.com/s/15umqzirkrmgru2yl7utbp8x4jv0bgmg.

6.3 Forking your own Flow program

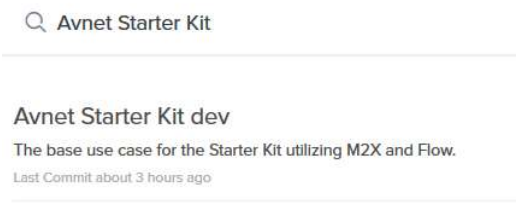
For the Pi 3 LTE Cellular IoT Kit, we start by copying a reference design and modifying it. Each Flow design is unique in terms of its endpoint addresses and account keys, so you need to edit these unique parameters.

To duplicate an existing project, follow these steps:

1. On the lower left side of the Flow IDE, click **Resources > Projects Library**.

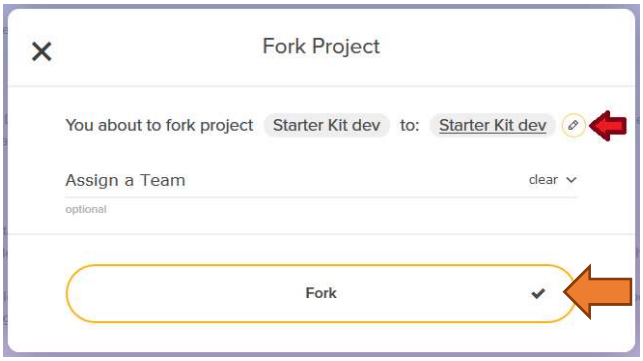


2. In the top middle, under Search Projects, enter **Avnet Starter Kit**.



3. Click **Avnet Starter Kit dev**.
4. Click the **Fork** button in the top right to create a copy of the project for your exclusive use.

5. Hover your curser to the right of the new name and click the pencil to change the name to something unique (e.g., **My Starter Kit Dev**). Then click the **Fork** button.

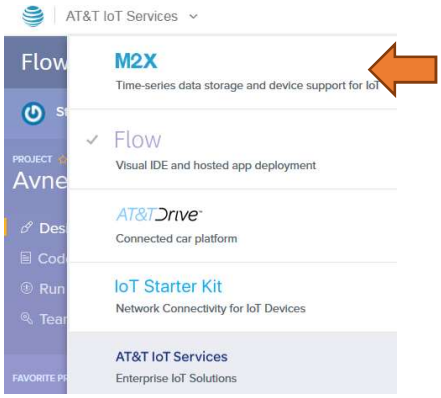


6. Once the project is Forked, click the **Deploy** button it activates and runs your new Flow program.

This compiles and activates your project. This step takes a minute the first time. Subsequent deploy actions will be quicker. Now that the project is built and deployed, **modifications to the sensor demo Python module attsensorflow.py must be made for it to work properly.** These changes are detailed in the sensor demo instructions found in **section 5.1.4** but first you must continue to the end of this Appendix.

6.3.1 More about M2X services

M2X is a time-series data store that is free for up to 10 devices. Navigate to **https://m2x.att.com** or access it from within your AT&T IoT Services Account.

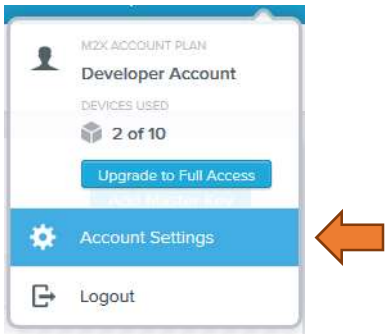


When we connect Flow and M2X projects, we will use Flow to create the necessary devices and streams.

6.3.2 Connect Flow and M2X

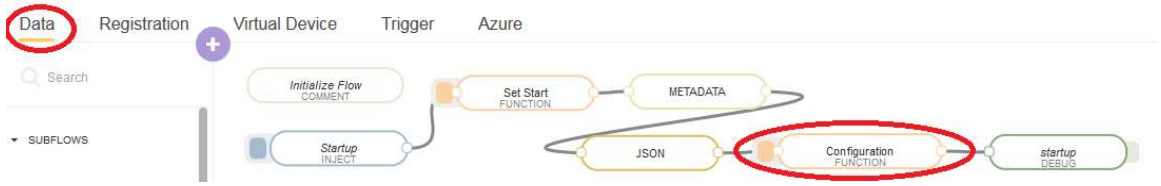
At this point, it is easiest to have two browser windows open (one for M2X and one for Flow) because the M2X master key must be pasted into the Flow Configuration Function.

1. In M2X, click the **Hello** power button on the top right and select **Account Settings**.

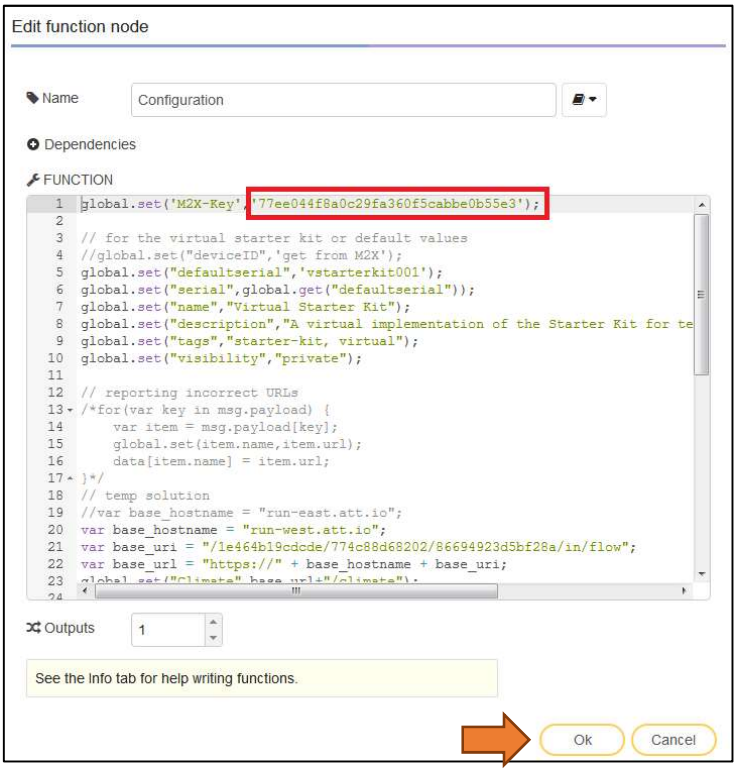


2. Highlight the Master Key and copy it to the clipboard.

3. Return to Flow in your other window/tab and find the Configuration function. Notice the graphical design has different tabs for each logical section. The Configuration function is on the Data tab.



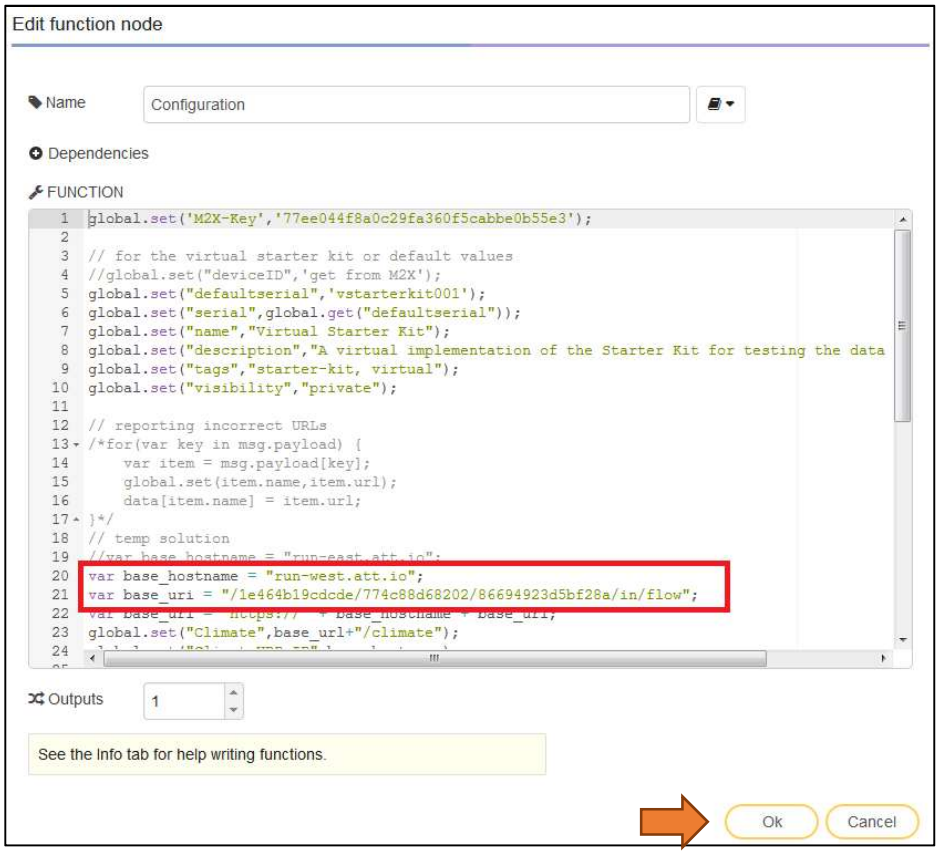
4. Double-click the Configuration function and a window with JavaScript appears. Paste the key you previously copied into the location of the current **M2X-Key** field and click the **OK** button. It may take a while for the script to update.



5. At the bottom of the canvas (where the configuration bubble is located), click the Endpoints tab where the **Base URL** field is located. Click the **Copy** button. The copied information is used to update two fields: **base_hostname** and **base_url**.

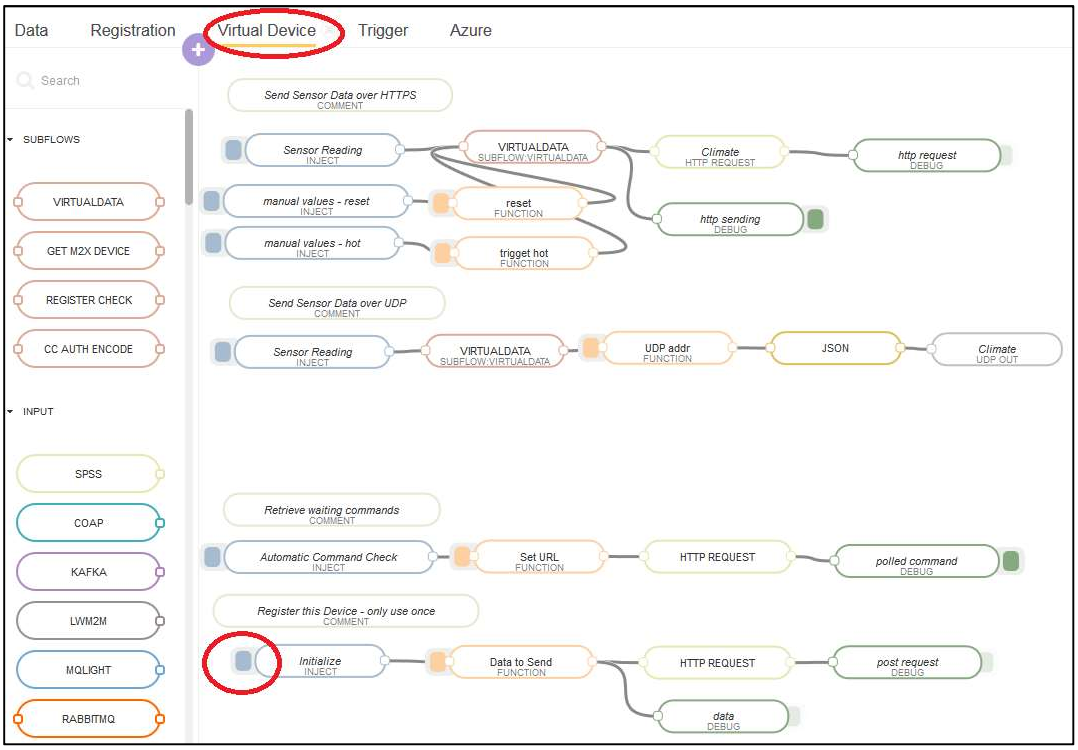
The Base URL is: https://run-west.att.io/1e464b19cdcde/774c88d68202/86694923d5bf28a/in/flow. The first part, run-west.att.io, is the **base_hostname**. The **base_url** is, above, displayed in green.

6. Open the Configuration function and paste the fields where they belong. Click the **OK** button.

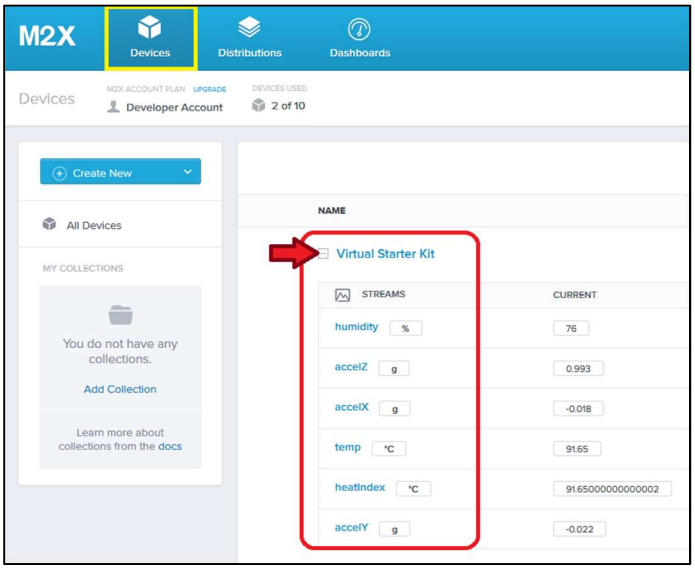


7. With the configuration updated, a blue dot appears on your function graphic. This indicates it has changed but has not been deployed. Click the **Deploy** button to resolve this.

8. Flow automatically creates a Virtual Device in M2X for you. This is a one-time operation that has to be performed. In Flow, go to the Virtual Device tab of the canvas and find the Initialize component in the bottom left. Click once on the solid part to the left of Initialize.

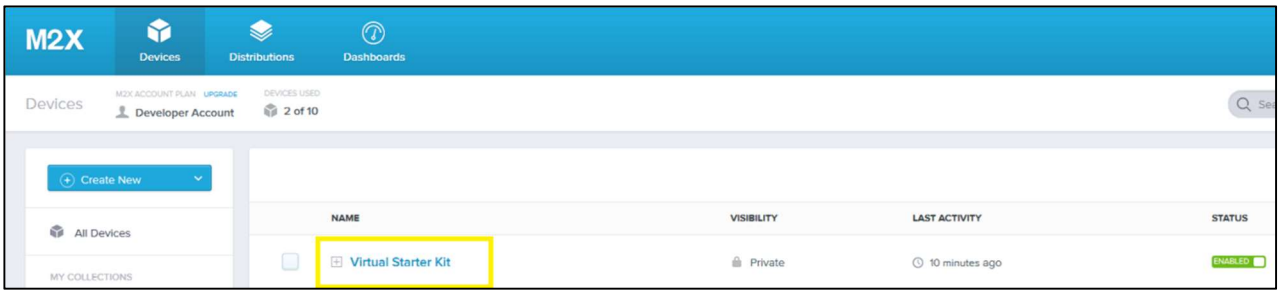


9. You can verify the component has been created in the M2X environment by returning to your M2X window and clicking **Devices**. Your virtual device should appear. If you expand it by clicking the plus sign (+), you will see the different M2X streams that have been created for it.



6.3.3 Intro: Flow Graphing Capabilities

In the M2X screen, click **Devices** and click **Virtual Starter Kit**.



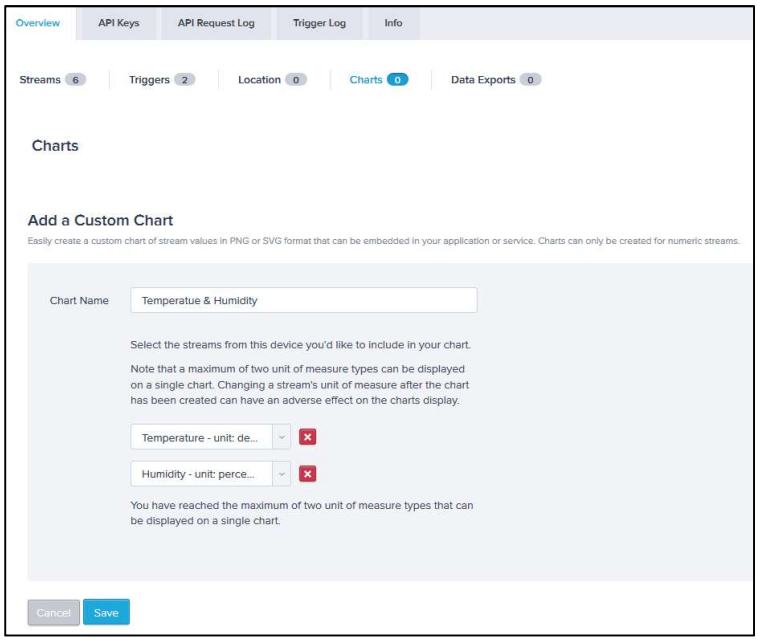
In your device, you will see a number of enabled Streams on the left. Select **Acceleration Z-axis**. Then go and change the position of the board by moving the levers that control the virtual boards orientation.

Note: The graph below was originally generated with the Avnet AT&T Cellular IoT kit which had a true hardware accelerometer. Your time plot will show a stair step reflecting when and where you moved the position sliders.

For more information about the alternative kit please go to: http://cloudconnectkits.org

Note: You can select how many values you want to view from the drop-down box below the graph.

It is also possible to view multiple streams on one chart by clicking **Charts > Add Chart**. Select the streams you want to view and click the **Save** button.

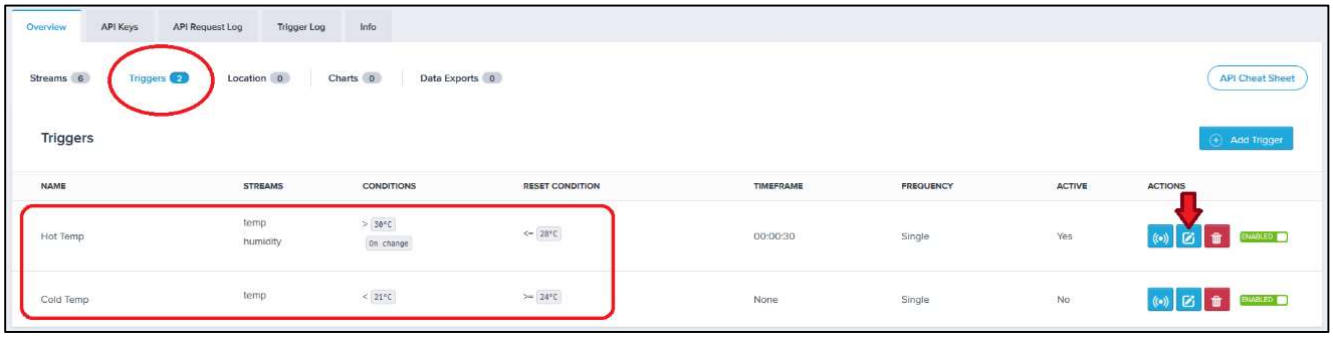


Note: With the free account, you can add a maximum of two streams.

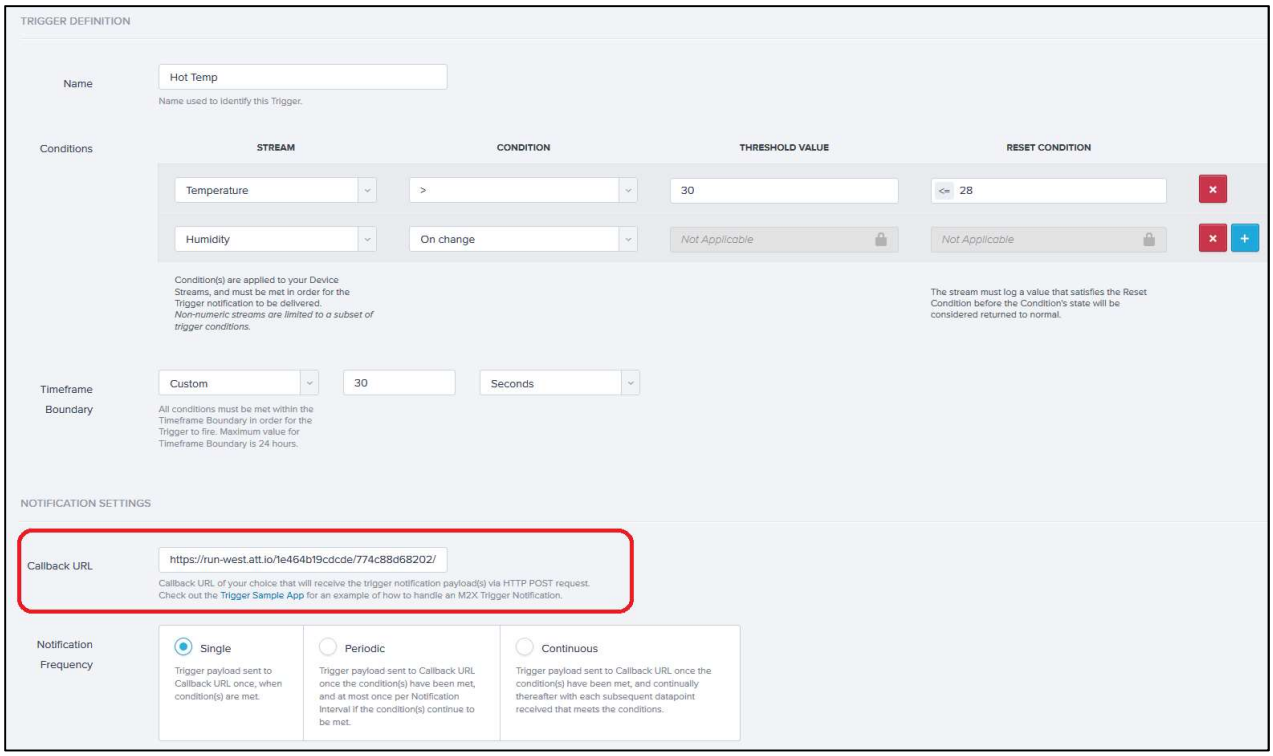
You can create graphs that use streams from different devices. It is also possible to draw a graph of Temperature vs. Humidity by clicking **Dashboards** at the top of the screen and selecting different dashboards using customizable widgets that can be embedded elsewhere using the URL.

6.3.4 Intro: M2X TRIGGERS

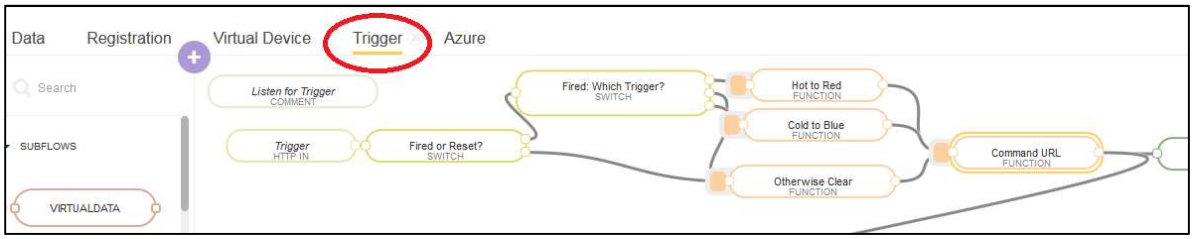
Triggers can be created in M2X devices. In the IoT Kit project, click **Devices > Virtual Starter Kit > Triggers**. Flow automatically creates two triggers for **Hot Temp** and **Cold Temp**. Click the edit pencil on the right.



You can modify the trigger conditions. The Callback URL is the endpoint address for the /trigger input in the Flow design.

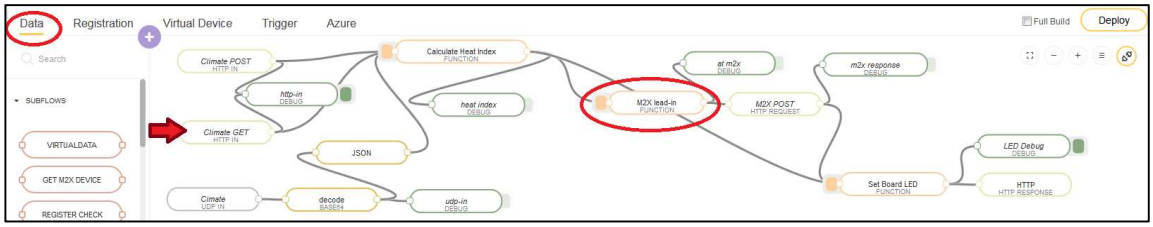


Click the Trigger tab to explore how Flow processes the trigger inputs and uses them for notifications.

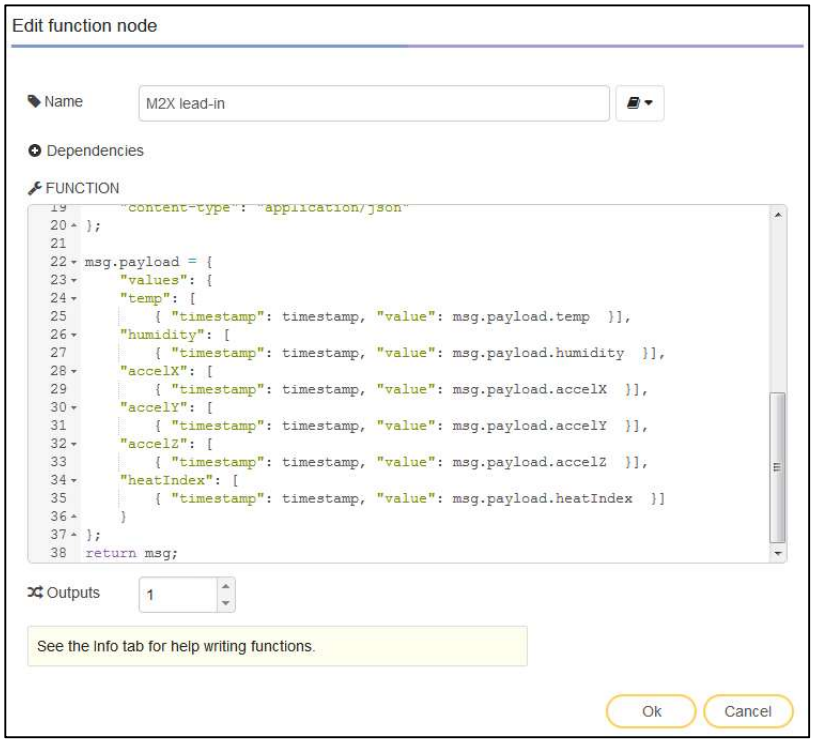


6.3.5 Intro: Data Processing in AT&T Flow

When data is sent from the WNC-Shield sensors to Flow, it enters via a HTTP-IN GET port. From here, it goes through a number of nodes. On the Flow Data canvas, the M2X lead-in function is where data enters. Double-click the bubble to open it.

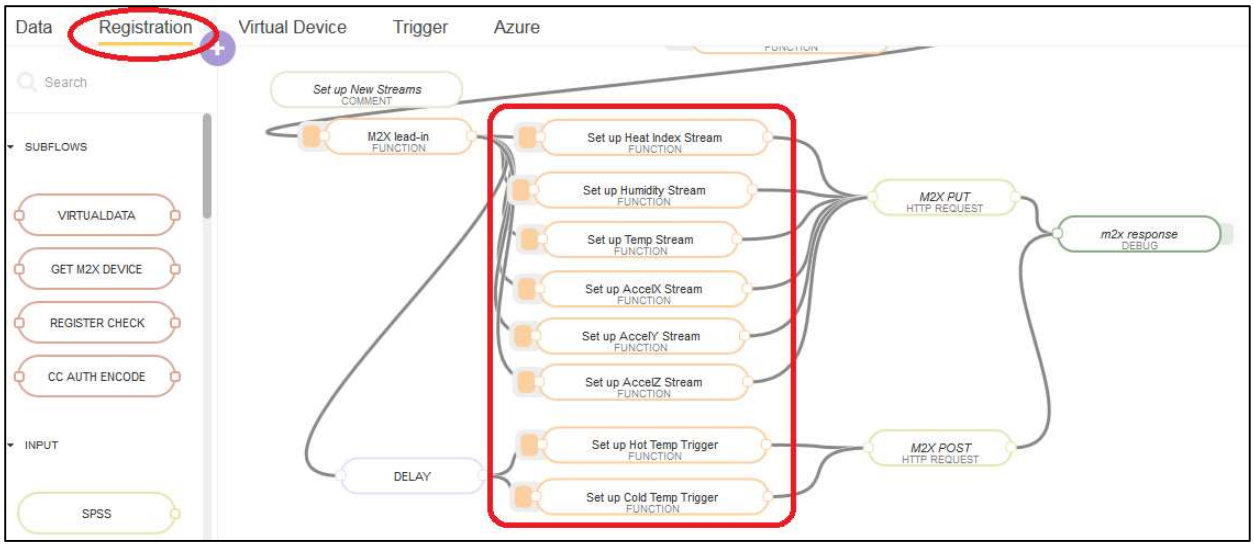


Scroll to the bottom of the JavaScript content. This is the function that composes the POST message for M2X. This function contains the sensor fields that get extracted from the incoming payload.



Note: If attempting to POST to an M2X stream that does not exist, an error is generated.

To determine which streams are present, click the **Registration** part of the Flow canvas as shown below.



This is where the various sensor readings are processed and either sent (PUT) or posted to M2X.

7 Appendix B: AT commands for the WNC 14A2A

When the modem board is connected to the USB port of the Raspberry Pi 3, two separate types of devices will be installed. The first is the Ethernet connection which we've been using to allow the Pi to send data to the internet. In addition, a serial port will also appear. This serial port was used prior to setup the APN setting of the modem with an AT command.

Serial AT commands are in the form of ASCII characters. The modem will not process the command until a return character is entered, then the modem will respond with OK or an error. Once the modem replies it is safe to issue another command. You may use any serial terminal program to interact with the 14A2A using AT commands. In an earlier section this guide showed how to install and use minicom.

The modem supports many types of serial AT commands that allow for querying the status of the cellular signal, the ICCID of the SIM card, connectivity states, SMS messaging, sending internet data and more.

Under the Developer / Resources section, you may find a PDF document containing a complete list of AT commands on the ATT website: http://starterkit.att.com

Avnet's other kit, the AT&T Cellular IoT Starter Kit exclusively controls the 14A2A modem with AT commands. You can find examples of many AT commands used in a C++ class: WncControllerLibrary. The source for this class is located at: https://developer.mbed.org/teams/Avnet/

8 Appendix C: SMS messaging

The WNC 14A2A modem and the included AT&T SIM card provide for basic IoT SMS messaging capabilities. This means you can send and receive SMS messages to other IoT devices on the m2m.com.attz APN as well as to AT&T's Jasper servers.

> Note: You will not be able to directly send and receive SMS messages to the voice phone cellular network. You will be able to do this using either Jasper as a bridge or services like Twilio.

The only way to use the modem for SMS messaging is through serial AT commands. See Appendix B for more information about the serial AT commands, in the AT command document the SMS commands are listed. Avnet's other kit, the AT&T Cellular IoT Starter Kit performs SMS messaging with the 14A2A modem and AT commands. See the C++ class: WncControllerLibrary. The source for this class is located at: https://developer.mbed.org/teams/Avnet/

In order to send a message to a device you need to become familiar with MSISDN and ICCID numbers. A brief description of these as well as Jasper can be found under the section Developers on: http://starterkit.att.com

Here is a brief example of sending an SMS message to another 14A2A modem. First obtain the ICCID number by entering:

AT%CCID

This will return a 19 or 20-digit number which is the ICCID number. The 14A2A can only dial using the MSISDN number. To convert the ICCID number to a MSIDN number:

1) MSISDN numbers start with: 882350
2) If the ICCID is 20-digits long: start at the 10th digit of the ICCID and use the remaining digits except discard the last 11.
3) If the ICCID is 19-digits long: start at the 10th digit of the ICCID and use the remaining digits except discard the last 10.

In C++ code this looks like:

```
bool WncController::convertICCIDtoMSISDN(const string & iccid, string * msisdn)
{
    msisdn->erase();

    if (iccid.size() != 20 && iccid.size() != 19) {
        dbgPuts("Invalid ICCID length!");
        return (false);
    }

    *msisdn = "882350";

    if (iccid.size() == 20)
        *msisdn += iccid.substr(10,iccid.size() - 11);
    else
        *msisdn += iccid.substr(10,iccid.size() - 10);

    return (true);
}
```

Now that you have a MSISDN number you can use it with the SMS AT commands to send a message. First you have to setup some memory slots on the SIM card, this is more for incoming messages but can also be used to save a message to memory and send it from there. This will setup the max allowed 3 memory slots:

AT+CPMS="SM","SM","SM"

Next we can send the test, you will need your MSISDN number and it will take several AT commands, enter the following sequence:

AT+CMGF=1

AT+CMGS=<your 15 MSISDN digits go here>

The modem will now go into a mode where it expects you to type the text that you would like to send in the SMS message. Go ahead and type your message, it has to be shorter than 150 characters and when finished you will need to press **<CTRL>-Z**. This will tell the modem to send the message. If successfully sent, you should see a response of OK.

You can also send messages to and from the AT&T Jasper server which you can access from your SIM account. See http://starterkit.att.com for more information. When sending to Jasper you will use a much shorter phone number, an easy way to get this number is to login to your SIM account and send an SMS message from there to your modem. The received message will contain the sender's number. More information on the necessary AT commands can be found on http://starterkit.att.com in the AT command reference PDF.

9 Appendix D: Troubleshooting

9.1 Cellular Modem Troubles

If you are having trouble with AT commands through the serial port, make sure you are using the correct serial port and settings. The proper settings are: 115200, 8, N, 1. In order to excute these commands you need to use a serial terminal program such as the minicom we installed and used during the initial setup.

> Note: for further information on the AT commands, please refer to the AT&T AT Command Guide found on http://starterkit.att.com

If you cannot get data through, it may be a signal quality or SIM card issue, the following AT commands can be used to debug that.

AT+CSQ

Returns the signal quality as measured by the modem it returns the RSSI and BER.

AT+CPIN?

Returns whether or not the modem can read the SIM card properly. When things are working correctly the response will be: CPIN: READY.

AT+CREG?

Returns whether the modem is connected and registered on the cellular network. If connected, the 2nd argument returned will be either 1 (registered on home network) or 5 (registered but roaming).

9.2 If you are having troubles with the USB Ethernet port

First ensure it is not a signal or SIM card issue before proceeding. Another indication of a signal quality or network trouble is the IP address that the modem obtains. Issue the following command in a Pi terminal window:

ifconfig eth1

If the Pi responds with invalid or nothing it is a USB connection issue, the cable may be bad or the Pi was unable to enumerate the driver. Try disconnecting the USB data cable between the LTE modem board and the Pi and then power cycling the LTE modem board. Wait for about 15 seconds after re-applying power before re-attaching the USB data cable. Issue ifconfig eth1 again.

If the Pi responds that the modem has an ip address of 10.0.0.x then it was unable to connect to the internet through the cellular network. Detach the USB cable from the data connector that goes to the Pi. Power cycle the LTE modem board and wait for about 15 seconds before re-plugging in the USB data cable. Issue ifconfig eth1 again. If that doesn't work, double check the signal quality, SIM card and cellular network registration with the serial AT commands. Try relocating your setup or the antennas to improve the signal quality.

10 Appendix E: SSH Setup for Raspberry Pi 3 Jessie Raspbian

SSH is a means to securely access the terminal capabilities of your Raspberry Pi over a network connection.

To enable the Raspberry Pi SSH server, open a Pi terminal window through the GUI Desktop:  and type and enter:

sudo raspi-config

Use the up/down arrow keys to select **Interfacing Options** then TAB to <Select> and ENTER.

Use the up/down arrow keys to select **P2 SSH** then TAB to <Select> and ENTER.

When asked to enable the SSH server, use TAB to choose <Yes> and ENTER.

A dialog should appear informing you that the SSH server is enabled, just ENTER to exit. Then TAB to <Finish> and ENTER. You should now be returned to the command prompt of the terminal window.

You will now need to know your Pi's IP address, this can be retrieved by typing and entering in the terminal window:

Ifconfig

If using WiFi look for the IP address listed under the wlan0 interface, it will be of the form x.x.x.x where x is a number from 0 to 255.

Once you have the IP address you will setup your SSH client software on a remote computer to use the IP. The default login credentials are:

Default user name: **pi**
Password: **raspberry**

For client software, use Linux or for Window's or Mac, use Putty or for even more functionality on Window's client's use the BitVise SSH Client. Once both client and server are setup you will have to connect them, see the manual for the client software for more information.

11 Appendix F: Setup a Pi with the Raspbian Jessie OS

Important: while it may work with other OS's, we strongly suggest that you choose and install Jessie Raspbian with Pixel. The instructions following are written assuming this version of OS.

Once you have obtained the required components listed above, you will need to download the Raspbian image onto your micro SD card, do this by following the instructions found at:

Please go to: **https://www.raspberrypi.org**

Find and click the **HELP button**, scroll down to Documentation and select **INSTALLATION**, next choose "**Installing Images**" and select and follow the instructions for your host computer's OS.

Note: it is OK to use the NOOBs installer but it generally takes longer since you are going to have to download more than just the Raspbian image. Also the Pi micro SD card will have to both read and write and the speed through the Pi socket is slower than many external card readers. If you wanted to install an alternative OS besides Raspbian for your Pi, NOOBs would be one way to do this. In this case though we don't! However, one advantage to using NOOBs is that during the first step you can more easily setup the country and proper keyboard. The default Raspbian image is set to UK and you will need to reconfigure if that is incorrect for your setup. Instructions for how to do this are included below.

There are lots of great learning tips on the Pi web-site under **HELP**, come back and peruse while you're waiting for the OS image download to complete!

Assuming you did not use NOOBs and have completed installing the Jessie Raspbian image onto your micro SD card:

> Note: if you are going to use SSH or at any time you are prompted for a user and password, login as:
>
> User: **pi** Password: **raspberry**

1. Insert your programmed micro SD card into your Raspberry Pi 3.
2. Attach the HDMI cable to the Pi 3 connector and a monitor or HDTV.
3. Plug in a USB Mouse and Keyboard to the Pi 3.
4. Power up the Pi by attaching an appropriate Pi 3 compatible USB Power Supply.
5. If everything is working correctly, the monitor or HDTV should show a startup terminal and eventually the Raspbian GUI Desktop.
6. By default the Raspbian OS is setup for the UK. Unless you desire this setup, we need to change the local to something more appropriate. The following assumes the US but you should select whatever is appropriate:

    a. From the gui only (do not use SSH here) open a Pi Terminal window 
    b. In the terminal window type and enter: **sudo dpkg-reconfigure locales**
    c. This will open a dialog where you will select alternative locals:
        i. Use the down arrow key to find and then unselect the en_GB.UTF-8 UTF-8 option. Unselect by using the space bar.
        ii. Use the down arrow key to scroll down and locate the US option, use the space bar to select it. When selected, an asterisk will show. Choose: en_US.UTF-8 UTF-8
    d. Use the TAB key to select <Ok> on the dialog and then press ENTER.
    e. Use the down arrow key to select en_US.UTF-8, then TAB to <Ok> and press ENTER to finalize the setup.
    f. Wait for a little while for the Raspberry Pi 3 to process your choice. When complete it should say: en_US.UTF-8… done. Leave the terminal window open for the next step.
    g. Next we need to reboot the Pi. In the open Pi Terminal window type and enter, then wait for the Pi to reboot: **reboot**
7. After the Pi reboots, open back up a Pi Terminal window and now let's change the keyboard layout from the UK to the US:

> **CAUTION:** If you select the wrong keyboard, some keys may not work properly, at worse it may not work at all and you might have to start over and copy the default OS image again to the micro SD card.

    a. From the same terminal window used in step 6, type and enter:

sudo dpkg-reconfigure keyboard-configuration

b. This will put up a dialog that lists all of the choices for a US keyboard model. Use the up/down arrow keys to find your keyboard. If you are unsure, **Generic 101-key PC** is a good choice. Use TAB to select <Ok> and press ENTER to accept.
c. Next a dialog for the keyboard layout will appear, use the down arrow key to find and select **Other**. Use TAB to select <Ok> and ENTER to accept.
d. Another dialog will appear with a list of country keyboard layouts. Use the up/down arrow keys to find: **English (US).** It should be just below the UK. Use TAB to select <Ok> and ENTER to accept.
e. Another list of layout choices will appear. For QWERTY keyboards, which are the norm use the up/down arrow keys to select: **English (US)**, then TAB to select <Ok> and ENTER to accept it.
f. Next setup the AltGr setting, use the up/down arrow keys to select: **The default for the keyboard layout**, use TAB to select <Ok> and then ENTER to accept.
g. Chose **No compose key**, use TAB to select <Ok> and ENTER to accept.
h. Leave the Control+Alt+Backspace set to: <No>, ENTER to accept.
i. Wait for the Raspberry Pi to complete the selections. It is normal to see a few warnings about start and stop.
j. You should now be setup properly to use a US keyboard, leave the terminal window open for the next step.

8. Now setup the proper time zone, by default it is setup for the UK:
   a. From the same terminal window opened in the prior steps, type and enter:
      sudo dpkg-reconfigure tzdata
   b. A geographic dialog will appear with a list of locations:
      i. Use the up/down arrows to select the proper country (US), use TAB to select <Ok> and ENTER to accept.
      ii. A new dialog will appear with a list of cities and regions and time zones, choose the closest to you by using the up/down arrow keys and then use TAB to select <Ok> and press ENTER to accept.
      iii. The dialog should close and the terminal window will show a review of your choices.
      iv. Leave the terminal window open for the next step.
   c. One last time reboot the Pi. In the open Pi Terminal window type and enter, then wait for the Pi to reboot: **reboot**
9. The next step is to connect your Pi 3 to a WiFi network.

> Note: if you are using a hardwire Ethernet connection, you may skip this step. See the Raspbian support page for further instructions on setting up Ethernet.

   a. Use the mouse to left-click the disconnected network symbol 
   b. Look for your WiFi SSID, left-click to select it from the list.

> Note: if the WiFi SSID is not listed or you have a hidden SSID, you will need to go right-click the disconnected network symbol and choose Wireless & Wired Network Settings and configure accordingly.

c. If you have a secure WiFi network you will enter your password in the dialog box "Pre Shared Key". Left-click OK to enter the password.

d. If successful, the network connection icon should now show a WiFi symbol 

e. Next open a terminal window by clicking on  , from the terminal window enter:

> Note: some firewalls will not allow ping responses back into the Pi. So if after entering the command it doesn't respond, press <CTRL>-c to stop. Instead go and open the Pi web browser by using the mouse to left-click on: 

ping –I wlan0 8.8.8.8

f. If everything is working proper you should see the ping command respond similar to:

```
pi@raspberrypi:~ $ ping -I wlan0 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 10.0.0.123 wlan0: 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=3.20 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=5.92 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=4.91 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=5.09 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 3.202/4.784/5.928/0.991 ms
pi@raspberrypi:~ $
```

To stop ping once it is running, press <CTRL>-c. ping is a utility program that is often used to investigate connection and routing issues. ping is synonymous with the ICMP IP protocol. When the **ping** command is invoked with an IP address (or URL), ping repetitively sends a small amount of data to the specified destination's ping server. Once the ping server receives the request, it issues a short reply back to the sender. Along the way timestamps are taken which allows one to also see how fast the connection to that destination is. Lower numbers for time indicate faster communications. If no response is given it indicates simply the sender cannot talk to the ping server at the destination. Not all destinations run ping server services. For more information about the Pi's version of ping, from a command prompt issue the command: **man ping**

12 Appendix G: Making the cellular connection's route persistent over power cycles for IP routing

> CAUTION: After making the following changes, all Pi internet traffic will be running through your metered LTE modem board! Remember you only have 300MB to start with! You may purchase extra data through your AT&T IoT Platform Jasper account.

This assumes a default networking setup with the built-in Pi /dev/eth0, /dev/wlan0 and the added LTE modem board's /dev/eth1 that is created once the USB data cable is connecting the LTE modem board and the Pi.

Open a Pi terminal window  and type and enter the following:

sudo cp /etc/network/interfaces /etc/network/interfaces.orig

This will copy and save the original networking configuration. If you ever want to revert to the original settings copy the backup file with extension .orig back over top of /etc/network/interfaces

Next we are going to edit the /etc/network/inferfaces file. To do this, open the file with either the vi or nano text editors. Assuming nano, from the Pi terminal window type and enter:

sudo nano /etc/network/interfaces

Use the arrow keys to go toward the bottom of the file right below the line "iface eth0 inet manual" add a blank line and add "iface eth1 inet dhcp" and "post-up route add default eth1" and leave another blank line in between the next line. Add these new lines in same alignment and format as the other commands there. When you are done, that section should look something like this.

iface eth0 inet manual

iface eth1 inet dhcp

post-up route add default eth1

> Note: the space indentations for the "post-up " line, 1 or more spaces are necessary.

Once the new lines have been added, press <CTRL>-x to exit the nano editor. Press 'Y' and ENTER to save the file.

If you are not using the Pi's wired Ethernet connection eth0 but rather the wlan0 connection, nothing further needs to be done to have Pi use the cellular mode for IP data. This is because by default eth1 has higher priority than the wlan0 connection.

If you are using the wired Ethernet connection eth0, you will also have to add the following line just under the “iface eth0 inet manual” line:

iface eth0 inet manual

post-up route delete default eth0

Next, restart the Pi for these changes to take effect. Do this by entering the following in the Pi terminal window:

reboot

Wait for the Pi to restart and now your data should be routing through the LTE cellular modem board. The settings we added to the interfaces file give priority to the /dev/eth1 interface above the other Pi interface devices. Other scenarios for priority are possible but beyond the scope of this getting started guide. For more info see “man route” and “man interfaces” as well as searches on the subject of routing for Linux. Be careful with reading information based on older versions of Raspbian/Debian. Prior to Jessie the interfaces file worked somewhat different. Jessie brought into play the dhcp client services and configuration files. There are a lot of tutorials and forums that are referencing older versions of Raspbian/Debian OS.

Attention: It appears that Jessie Raspbian is broken for some versions. This prevents the “post-up route add default eth1” from making eth1 the default route.

To fix or work around the problem. Use **sudo nano /etc/rc.local** to open and edit the file. Consider making a backup of the file (use: **cp /etc/rc.local /etc/rc.local.orig**). Add the following line at the end of the file:

sudo route add default eth1

Add just above exit 0, exit 0 should be the very last line. After the line is added, press <CTRL-X> then Y and enter to save. Next time you reboot eth1 should be setup to use as the default to route IP traffic.

If you also need to delete the default for eth0, you can add a line in /etc.rc.local right below the newly added sudo route add default eth1, add with:

sudo route delete default eth0