Project Assignment 2. Turning a BeagleBone Black Wireless (BBW) into an IoT sensor node

General Instructions:

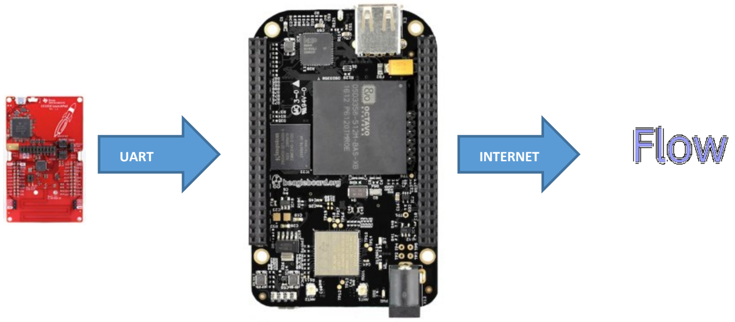
- Submission Deadline: April 18th, 2018. Please submit electronically to (rja150230@utdallas.edu) and consider that you may be required for a demonstration.
- Team members: A maximum of two students per project.
- Collection of Equipment: Stop by TA office ECSN 3.520 Wednesdays and Fridays from 2:30pm to 7:00pm.

Problem Statement

The Internet of Things (IoT) concept involves connecting things to the Internet in order to make our lives easier, more efficient, and friendly with the environment. For this purpose, a BeagleBone Black Wireless (BBW) will be use it as a sensor node collecting data. In addition, the BeagleBone will send the collected data to the Internet using its built-in Wi-Fi radio connectivity. The data being sent over the Internet will be received at the AT&T Flow IoT Designer Platform which is a NodeRed-based graphical environment to design and deploy IoT applications quickly.

Components

- One (1) BeagleBone Black Wireless with Linux installed
- Internet connection over the WiFi built-in adapter of the Beaglebone Black
- AT&T Flow Design user account (it is free of charge at http://flow.att.com)



Procedure

The sensor data being collected by the BBW will be emulated by reading one of its serial ports, that is, everything received over the selected serial port on the BBW will represent the sensor data (You should use an external device to send data to the BBW, for example a CC1310). These data will be sent over the WiFi connectivity of the BBW to the AT&T Flow Design platform and will be displayed there using a debug node to see all the incoming messages. To achieve this, you may follow two approaches as described below:

1. You may implement this functionality by using a Node-red server installed in your BBW or,

2. Writing a Bonescript application for your BBW.

Basic instructions to implement this project will be provided for both approaches as follows. Therefore, you must add the missing pieces to complete the project.

Using a NodeRed Server at the BBW

You will need first to install the Node-Red server in your Linux running on the BBW. By following the steps described below, you should be able to install and start the Node-RED server in your BBW.

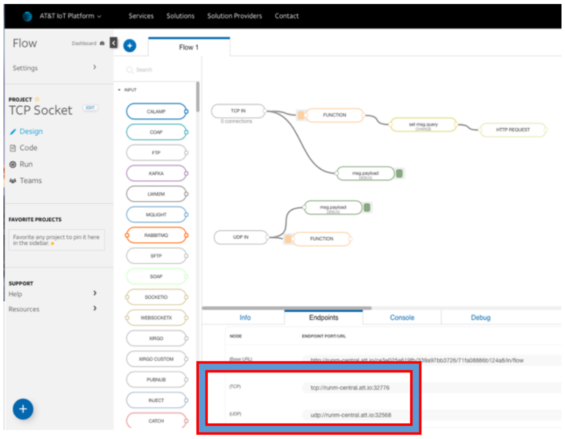
1. Make sure that your Internet connection is up and running in your BBW through your WiFi adapter.
2. Open the root prompt and execute the following command:

```
bone# npm install –unsafe-perm –g node-red@0.12.1
bone# node-red
```

3. Add BeagleBone specific nodes:

```
bone# cd ~/.node-red
bone# npm install node-red-node-beaglebone
```

4. To access the Node-RED server, you need to get the IP address assigned to the BBW WiFi adapter by the WiFi Internet Connection (through your Access Point). Then, open a browser and type http://xx.xx.xx.xx:1880 to access the server (the x marks represent your local IP address).
5. For this experiment you will need to drag the following nodes to the canvas:
   - serial: use it to read a serial port of the BBW
   - function: use it to manipulate the data coming from the serial port to create the message payload to be send over a TCP/UDP socket.
   - tcp out/udp out: use it to implement a TCP/UDP client in order to send the data through a socket which matches the socket configuration at the AT&T Flow Designer Server.
   - debug: use it to debug the implementation by looking the output coming from the nodes.
6. Wire the above nodes properly in order to provide the functionality being asked.
7. At the AT&T Flow Server, you have to implement a TCP/UDP server using a tcp in node and then process and display the incoming messages using the function and debug nodes, respectively. Take note of the endpoints assigned to the TCP/UDP nodes in order to communicate through them, that is, their IP address and the Port Number as illustrated in the figure below.



NODE	ENDPOINT PORT/URL
(Base URL)	http://runm-central.att.io/ce3e025a619fb/339a97bb3726/71fa08886b124a8/in/flow
(TCP)	tcp://runm-central.att.io:32776
(UDP)	udp://runm-central.att.io:32568

8. At the Debug window the data being received must be displayed.

For more documentation about Node-RED please visit http://nodered.org
For more documentation about the BBW please visit http://beagleboard.org

Using a BoneScript

BoneScript is a node.js library which provides a set of Arduino-like functions written in JavaScript to simplify the development of physical computing tasks under embedded Linux, specifically designed to support the BeagleBone board.

The first step to use this set of functions is to check the installed BoneScript version of your BeagleBone to verify if you have the latest version. To check this and to verify the correct path, execute the following command:

```
node -pe "require('bonescript').getPlatform().bonescript"
```

To write a Bonescript application you can use Cloud9 IDE in your browser to start writing your script. Once you are ready for coding, add the bonescript library as part of code using 'require' directive. Functions are then referenced through the object provided back from require. This is equivalent to an #include statement in C.

Here's an example:

```
var b = require('bonescript');

b.pinMode('P8_12', b.INPUT);
b.pinMode('P8_13', b.OUTPUT);

setInterval(copyInputToOutput, 100);

function copyInputToOutput() {
    b.digitalRead('P8_12', writeToOutput);
    function writeToOutput(x) {
        b.digitalWrite('P8_13', x.value);
    }
}
```

The 'P8_12' and 'P8_13' are pin names on the board and the above example would copy the input value at P8_12 to the output P8_13 every 100 ms.

Please visit http://beagleboard.org/Support/bone101/#cloud9 to learn more about how to start writing a bonescript in Cloud9 IDE.

In order to address the requirements for this project, you will need to read the serial port, to establish an internet connection, and to send the data to a TCP/UDP server. For sample codes and instructions about how to use the serial port, please visit https://github.com/node-serialport/node-serialport . At this point, it is recommendable to set up the serial port in “flowing mode” in order to receive data continuously.

For more information about Node.js, please visit: http://nodejs.org
For more documentation about Bonescript, please visit: http://beagleboard.org/bonescript

Analysis

1. Record in the table below, the results of your implementation. 20 samples are enough to complete the table.

Message **Transmitted**	**Size** **(bytes)**	**Message** **Received**	**Size** **(bytes)**

2. What is the size of the payload received at the Flow server and compare this value with the payload size at the transmitter (BBW)? Are they the same?