1. Configure Edison Board

To perform this task we will follow the following steps

1. Connect to the Intel Edison Board from a Laptop.
2. Setup an IDE.
    a. We will use the XDK integrated development environment which provides the necessary libraries to interact and configure Intel Edison Board using Node JS. Download: https://software.intel.com/intel-xdk
    b. Scripts

1.1. Flash Intel Edison board

The first step is to flash the intel Edison board i.e. to install the operating system on the chip. There are multiple OS we can select from, to install on the Board. The default OS is YOCTO linux. The following tutorial from intel gives us the clear step by step installation process.

Windows:	https://software.intel.com/en-us/get-started-edison-windows-step2
Linux:	https://software.intel.com/en-us/get-started-edison-linux-step2
Mac:	https://software.intel.com/en-us/get-started-edison-osx-step2

1.2. Connect to the board using Serial console

Once the OS is installed we can connect to the Edison board using a serial console provided on the base board. To connect from windows use the following steps.

1. Check for the USB serial port in the Device manager.
2. Open Putty
3. Select Serial connection
4. Input USB serial port found from step 1
5. Set speed to 115200
6. Connect

You will be prompted for the username (root) and password. If it is the first-time logging into the system there will be no password and hence type Enter.

For connecting from linux use the following steps

1. Install screen using apt-get or yum install.
2. Use the command `sudo screen /dev/ttyUSB0 115200` The USB number might be different on different systems

For Mac Please use the steps in the below link.
https://software.intel.com/en-us/setting-up-serial-terminal-on-system-with-mac-os-x

1.3. Connect to the Edison board using WiFi

Once we are in the Edison it is just another Linux machine with a shell Interface. Since YOCTO supports bash most of the shell commands are available on the Edison board.

In the shell use the command `configure_edison` to setup root password, device name (not the username) and wifi.
To only connect to a wifi use `configure --wifi`

Now that we have a wifi setup we can connect to the Edison Board using the ssh (use putty on windows). The laptop and the device should be in same network. Use ifconfig to get the IP address.

2. Create an IoT application on Bluemix

Similar to temperature app created, create an application on Bluemix.
At this stage do not go to Node RED editor. We first have to register the Intel Edison as an "Internet of Things" service.

2.1 Create "Internet of Things Platform"

Before registering the device we need to create a "Internet of things platform". This is the platform on which we will register our device. Use the following steps to create the service
a. In the dashboard for the created app use "Add a service or API"
b. Select "Internet of Things Platform"
c. Click create
d. Click restage

2.2 Get the Intel Edison MAC address

We need to get the MAC address to use as the device ID. Since MAC address is globally unique we can safely use this as the device ID. Use the following steps to get the Device ID
a. Connect to Intel Edison via ssh or serial console
b. Give the command "wpa_cli status"
c. Note the mac address and remove the ":"

2.3 Registering Intel Edison as an "Internet of Things service"

Perform the following steps to register the Edison Device
a. From the Dashboard select the Internet of Things Platform created in section 2.1
b. Select Launch Dashboard
c. From the left selection menu click devices > add devices > create device type
d. Enter a name and create the device
e. In the add device window select the device name created and enter Device ID found from section 2.2

f. Note the org ID, Device type, Authentication Method and Authentication Token

3. Send data from Intel Edison to Bluemix via MQTT broker (Publish)

Create a new XDK project on the host laptop and connect to your Intel Edison Device.
Write the following code and edit the ‘config’ JSON variable to match to the values found in the above step. At this step we are connecting to the IBM Bluemix MQTT broker “<orgid>.messaging.internetofthings.ibmcloud.com” and publishing to the topic “iot-2/evt/status/fmt/json”.

```
var mqtt = require('mqtt');

/* JSON variable which holds the required configuration details to \
connect to Bluemix */
var config = {
	"org" : "<org-id>",
	"port" :"1883",
	"type" : "<type>",
	"id" : "<device-id>",
	"auth-method" : "token",
	"password" : "<token-key>",
	"username" : "use-token-auth"
};

/* cloud platform provided by the Bluemix */
var host = config.org + ".messaging.internetofthings.ibmcloud.com";

var clientId = "d:" + config.org + ":" + config.type + ":" + config.id;

/* topic to publish the data from Edison*/
var topic_pub = "iot-2/evt/status/fmt/json";

/* topic to subscribe to receive data */
var topic_sub= "iot-2/cmd/display/fmt/json";


/* connect to the mqtt broker */
var client = mqtt.connect(
{
	host: host,
	port: config.port,
	username: config.username,
	password: config.password,
	clientId: clientId
});
```

```
/* function to publish message (random num between 25 and 55) in JSON format */
function sendMessage(){
        var value = Math.floor((Math.random() * 30) + 25);
        var message = {
        "d" : {
        "temp" : value,
        }
        };
        client.publish(topic_pub, JSON.stringify(message));
}


/* When the client connects to the Bluemix host, this function is called.
It then calls ‘sendMessage’ function to publish the message */
client.on('connect', function() {
        console.log('connected to IBM');
        setInterval(sendMessage, 1000);
});


/* Function to subscribe to the topic to receive message from the Bluemix */
client.on('connect', function(){
        client.subscribe(topic_sub, {
                qos: 1
        }, function(err, granted) {
                if (err) throw err;
                console.log("subscribed");
        });
        setInterval(sendMessage, 1000);
});

client.on('error', function(err) {
        console.error('client error ' + err);
        process.exit(1);
});

/* The function that handles receiving of messages */
client.on('message', function(topic, message, packet) {
        var msg = JSON.parse(message.toString());
        console.log(msg);
});
```

Deploy and run this app on the intel Edison. This will start sending random numbers between 60 and 90 at an interval of 1 second.

More information on subscribe/publish topics is provided in the link below
https://docs.internetofthings.ibmcloud.com/applications/mqtt.html

4. Receive the data from Intel Edison at IoT application on Bluemix (Subscribe)

Now we need to receive this data on our Bluemix app. Use the following steps to receive the data.
For Node-RED to process the real-time data generate, we need to create an API key. To create an API key do the following steps:
a. From IoT Organization dashboard (Previously from where you created the device type) select Access(on the left pane) > API keys > Generate API Keys
b. Note the key and authentication > Finish

Now use the Node-RED subscribe to the device and receive the data sent by Intel Edison
a. Open the app in node RED editor (you should see the default temperature sensor app. We will use this to receive and send data)
b. Configure the "IBM IoT App In" with following configuration
a. Use API keys as authentication
b. For "API Key" create a new key and give the values noted in the previous step (Generate API keys)
c. Make "Device type" as the one created earlier (During 'Registering Intel Edison as an "Internet of Things service"' in section 2.3)
d. Give the Device ID (We are using MAC address of the device as Device ID)
e. Deploy the Application

At this point you should be able to receive data from the Intel Edison and the Debug node should display these in the debug window.