README

The following sections will explain the necessary components, system environments, as well as a general "how to" if you were to replicate this project.

1 System Environment and Hardware Requirements

The project operates using IBM Watson, Node-Red, CLI Docker, TI CC3220SF, TI CC1350, TI CC28505 and TI Debug Devpack. The TI CC3220SF and TI CC1350 were stacked on each other to provide WiFi capability and the TI CC28505 and TI Debug Devpack were stacked on each other to provide sensor data capabilities, with both stacks communicating with each other.

2 SensorTag Tutorial

This section will describe how to use the TICC28505 SensorTag and TI Debug Devpack. To start, make sure they are properly stacked before continuing on to the following steps.

2.1 Connect to TI SensorTag App

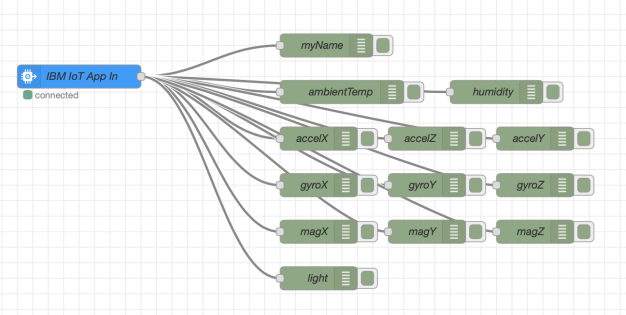
This app can be found in the iOS appstore under the name SensorTag, or the Android app store. Download the app and proceed with the next step.

2.2 Format SensorTag to IBM Watson

The SensorTag must be formatted to work with IBM Watson

2.3 Create a Node-Red Application

Node-Red will allow us to configure all of the different nodes for data coming out of the device. It will eventually look like the following image.



2.4 Create IBM Application

This application will be used in conjunction with the Node-Red application. Once you have created this application, connect it to the Node-Red application.

2.5 Connect SensorTag to IBM QuickStart

This allows us to connect the sensor to IBM Watson and visualize the data.

2.6 Connect IBM QuickStart to IBM Application

We must connect these IBM services together to be able to allow the data to move to each IBM service. You should now be able to recieve the sensor data into your application.

2.7 View our Node-Red application

To see our Node-Red application, go to the following site.

https://sensortag2bj.mybluemix.net

Then enter the following username and passcode.

Username: cmpe189Group4

Password: cmpe189group4

3 TI CC3220SF and TI CC1350 Tutorial

This section will describe how to use the TI CC3220SF and the TI CC1350. In this particular application, we used IBM Watson. These devices also have the functionality to work with AWS, but modifications may be necessary if you are to run these devices on AWS.

3.1 Load Image onto CC1350

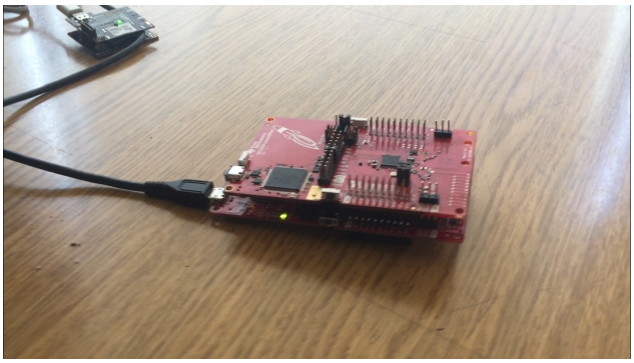
To begin, the proper image must be loaded into the CC1350 Launchpad by using Uniflash. Do this by choosing a new device, browse the firmware images and then load the appropriate firmware.

3.2 Load Image onto CC3220SF

Connect your CC3220SF to the computer and find the appropriate device. We must import the CC3220 gateway folder. Open CCS (Code Composer Studio), install the SDK and then this will allow you to import the gateway folder.

3.3 Stacking CC1350 and CC3220SF

Remove all jumpers EXCEPT the reset jumper. It is also important to set the BAUD rate to 115200. The following image represents what your stacked device should look like.



3.4 IBM Cloud

At this point, the two TI devices have been appropriately stacked, so we can connect the devices to the IBM cloud. Because this is an IoT device, we will choose the IoT option within IBM and then create Watson platform. Create a gateway for this device to IBM watson. Now connect your device to the wifi using the simplelink app.

4 Device Usage

Now that everything has been set up, the device can be used. Within IBM Watson, you will be able to view the data streaming from the temperature sensor (i.e. smoke detector). The Sensor will communicate its data to the stacked CC1350 and CC3220SF boards to relay its information via the gateway we created to the IoT service. When the sensor is detected to have reached a certain level, the appropriate alert will be sent to your mobile phone.

5 Roadmap

This device has many more future possibilities that can become much more complex than the features we have added to our device. One such feature could be to automatically contact an emergency service incase your house were to go up in flames. These precious extra minutes could mean the difference between saving some of your houses possessions versus losing the entire thing.

Another feature would be to monitor the power supply voltage and relay and create a rule to no-

tify your device when the battery is running out, as well as telling you which detector is going out (assuming that there are multiple smoke detectors. Having the ability to label each one would be a necessity). This would prevent the annoying search for the bad smoke detector that beeps once every 5 minutes. Instead, your phone would tell you which one needs a new battery.

6 Project Status

95% of this project has been completed with the exception of being able to use the IBM docker. In the end, we were still able to get our data to IBM Watson and create a live graph of the data, but we did not go through the route in which we had originally intended.