TI Designs: TIDC-01002
Sub-1 GHz Embedded Sensor to Cloud Industrial Internet of Things (IoT) Gateway Reference Design

Description
The TIDC-01002 design demonstrates how to connect sensors to the cloud over a long-range, Sub-1 GHz wireless network, which may be used in industrial settings, such as building control and asset tracking. This TI Design is powered through a TI SimpleLink™ CC3220 processor and SimpleLink ultra-low-power (ULP), Sub-1 GHz CC1310 and CC1350 devices. The reference design pre-integrates the TI 15.4-Stack part of the SimpleLink CC13x0 software development kit (SDK) and SimpleLink CC3220 SDK, which are part of the TI SimpleLink MCU platform, providing a unified software experience across TI low-power, wired, and wireless MCUs.

Resources
| TIDC-01002 | Design Folder |
| CC1310 | Product Folder |
| CC1350 | Product Folder |
| CC3220 | Product Folder |

Features
- Large Network-to-Cloud Connectivity Enabling Long Range, Up to 1 km (Line of Sight)
- Facilitates Designers’ System Compliance with IEEE 802.15.4e/g by Using TI 15.4-Stack
- Based on TI Tested Hardware Designs Enabling Quick Time to Market With Out-of-the-Box, Ready-to-Use Demonstration Software
- Implementation Based on Portable Operating System Interface (POSIX), Allows for Easy Portability Across TI Internet Connected Microcontrollers (MCUs)
- Supports Star Networks
- ULP Sensor Nodes

Applications
- Building Security Gateway
- Door and Window Sensor Networks
- HVAC Gateway
- Asset Management and Tracking

An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.
1 System Description

The TIDC-01002 provides a reference for creating an industrial Internet of things (IoT) gateway that is capable of connecting a network of wireless sensors to an enterprise cloud provider. In this TI Design, a long-range, low-power wireless network, made up of Sub-1 GHz CC1310 or CC1350 devices (both are supported) that run the TI 15.4-Stack-based application, can be connected to multiple cloud service providers, such as IBM Watson IoT®, AWS IoT, and so on. An online dashboard is provided that allows the user to visualize the real-time sensor data as well as send actuation commands from anywhere in the world using an Internet-connected device with a web browser.

This reference design provides a list of suggested hardware, schematics, and foundational software to quickly begin IoT product development. IBM Watson IoT™ was chosen as the cloud service provider for the demonstration. The design also provides an ability to visualize the data inside a local network without connecting to a cloud service. The software design is architected to be flexible to enable other cloud service providers of choice.

This TI Design enables IoT in numerous applications, such as building security gateways, door and window sensor networks, asset management and tracking, and other IoT-enabled home and industrial automation applications.

The connection between the wireless sensor network and the cloud is made possible by the TI SimpleLink CC3220 device on the CC3220SF LaunchPad™ development platform. On one side, the CC3220 is connected to a Sub-1 GHz device acting as the central node in the wireless network, and on the other side, the device is connected to the cloud service IBM Watson IoT using Wi-Fi®. These two connections allow the CC3220 device to act as a gateway to get the sensor messages from the Sub-1 GHz wireless network to the cloud and to get the actuation requests from the cloud dashboard sent back to the Sub-1 GHz wireless network.

Due to the long-range and low-power capabilities of the Sub-1 GHz sensors, this TI Design may be useful for any application that would benefit from distributed sensing. This reference design provides an example that gives the ability to visualize or actuate tens or hundreds of sensors while only needing one gateway device, the SimpleLink CC3220, to be connected to the Internet.
2 System Overview

2.1 Block Diagram

Figure 1. Block Diagram of IoT Gateway TI Design
2.1.1 Software Block Diagram

The following is a high-level description of each module in the software block diagram:

- **User interface application:** This application presents the network information and device information and provides ability to control network behavior to the end user.

- **IoT cloud application:** This application runs on the cloud server, which communicates with the IoT gateway application. The interface of the cloud service task with the cloud server is described in Section 3.2.2.2.

- **IoT gateway application:** This application runs on the SimpleLink CC3220. The application interfaces on one side with the cloud service task to enable cloud connectivity and on the other side to the collector task to interface with the TI 15.4-Stack based network. The interface between the IoT gateway and the cloud service is described in Section 3.2.2.2.
  - **Cloud service task:** This task provides the cloud service provider specific functionality. Users can take the current interface, which is designed as an extensible framework, and quickly modify the interface to add their own functionality for their end product development.
  - **Gateway:** This application component interfaces with the collector task through a POSIX message queue interface to enable connection with the TI-15.4 Stack network.

- **TI 15.4-Stack collector task:** This task implements the functionality that starts the network, allows new devices to join the network, configures the joining devices on how often to report the sensor data, configures how often to poll for buffered messages in case of non-beacon and frequency-hopping mode of network operation for sleepy network devices, and tracks connected devices to determine if they are active or inactive on the network. This determination is achieved by the collector periodically sending tracking request messages and awaiting corresponding tracking response messages. The collector task also implements components that talk to the gateway module. The communication is

![Software Block Diagram of TI 15.4-Stack Sensor-to-Cloud TI Design](image-url)
implemented through POSIX-based message queues.

- **MAC CoP** application: The MAC coprocessor application runs on the CC1310 or CC1350 LaunchPad, which provides a UART-based interface from TI 15.4-Stack to the IoT gateway application.
- **CC1310 or CC1350 LaunchPad Sensor End Node**: The sensor example application from TI 15.4-Stack and runs on the CC1310 or CC1350 LaunchPad.
- **CC13xx SensorTag**: The CC13xx SensorTag runs the sensor example application ported from the TI 15.4-Stack out-of-box sensor example applications, which enable support of the CC1350 SensorTag platform and integrate support for various sensors on the SensorTag platform.

### 2.2 Highlighted Products

This section highlights key hardware devices and software components used in the TI Design.

#### 2.2.1 SimpleLink™ CC1310 or CC1350

The CC1350 is a member of the CC26xx and CC13xx family of cost-effective, ULP, 2.4-GHz and Sub-1 GHz RF devices. In addition to flexible low-power modes, very-low active RF and MCU current consumption provide excellent battery lifetime and allow long-range operation on small, coin-cell batteries and in energy-harvesting applications.

The CC1350 is the first device in the CC13xx and CC26xx family of cost-effective, ULP wireless MCUs capable of handling both Sub-1 GHz and 2.4-GHz RF frequencies. The CC1350 device combines a flexible, very-low-power RF transceiver with a powerful, 48-MHz, Cortex®-M3 MCU in a platform supporting multiple physical layers and RF standards. A dedicated radio controller (Cortex-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus, ensuring ULP and flexibility to handle both Sub-1 GHz protocols and 2.4-GHz protocols (for example, **Bluetooth**® low energy). This enables the combination of a Sub-1 GHz communication stack that offers the best possible RF range together with a connection to a Bluetooth low energy smartphone that enables a great user experience through a phone application. The Sub-1 GHz-only device in this family is the CC1310.

The CC1350 device is a highly-integrated, true single-chip solution, which incorporates a MCU with a complete RF system and an on-chip DC-DC converter.

#### 2.2.2 SimpleLink™ CC3220

The CC3220x device is part of the SimpleLink MCU platform, which consists of Wi-Fi, low energy, Sub-1 GHz and host MCUs, which all share a common, easy-to-use development environment with a single core SDK and rich tool set. A one-time integration of the SimpleLink platform enables the user to add any combination of the portfolio’s devices into their design, which allows 100% code reuse when the design requirements change. For more information, visit [SimpleLink Solutions](#) overview.

Created for the IoT, the SimpleLink CC3220x device family from Texas Instruments is a single-chip solution that integrates two physically separated, on-chip MCUs. One of the MCUs is an application processor—an ARM® Cortex®-M4 with a user-dedicated 256KB of RAM and an optional 1MB of XIP flash. The other MCU is a network processor in charge of running all Wi-Fi and Internet logical layers. This ROM-based subsystem includes an 802.11b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure internet connections with 256-bit encryption.

The CC3220x wireless MCU family is part of the second generation of TI’s Internet-on-a-chip™ family. This generation introduces new features and capabilities that further simplify the connectivity of things to the Internet. The new capabilities including the following:

- IPv6
- Enhanced Wi-Fi provisioning
- Enhanced power consumption
- Enhanced file system security (supported only by the CC3220S and CC3220SF devices)
- Wi-Fi AP connection with up to four stations
- More concurrently opened BSD sockets; up to 16 BSD sockets, of which six are secure
- HTTPS support
- RESTful API support
• Asymmetric keys crypto library

The CC3220x wireless MCU family supports the following modes: station, AP, and Wi-Fi Direct®. The device also supports both WPA2-Personal and WPA2-Enterprise security modes. This subsystem includes embedded TCP/IP and TLS/SSL stacks, HTTP server, and multiple Internet protocols. The device supports a variety of Wi-Fi provisioning methods including HTTP based on AP mode, SmartConfig™ technology, and WPS2.0.

Figure 3. CC3220 Block Diagram

2.2.3 TI 15.4-Stack

TI 15.4-Stack is an IEEE802.15.4e/g-based software stack part of the SimpleLink CC13x0 SDK supporting a star network topology for Sub-1 GHz applications. TI 15.4-Stack software runs on TI’s SimpleLink Sub-1 GHz CC1310 or CC1350 wireless MCU. TI 15-4 Stack offers several key benefits, such as longer range in FCC band and better protection against in-band interference by implementing frequency hopping. The SDK also offers customers an accelerated time to market by providing a complete end-to-end, node-to-gateway reference design. TI 15.4-Stack is supported on the industry’s lowest-power SimpleLink Sub-1 GHz wireless MCU platform.

This release is available royalty-free to customers using TI’s CC1310 or CC1350 wireless MCU and also runs on TI’s SimpleLink Sub-1 GHz CC1310 or CC1350 wireless MCU LaunchPad development kit. This release is available royalty-free to customers using TI’s CC1310 or CC1350 wireless MCU and also runs on TI’s SimpleLink Sub-1 GHz CC1310 or CC1350 wireless MCU LaunchPad development kit.

Features:
• IEEE 802.15.4e/g standards-based software stack
• Frequency hopping
• Medium access with CSMA/CA
• Built-in acknowledgment and retry
• Network and device management (joining, commissioning, service discovery)
• Security feature through AES 128-bit encryption and integrity check
• Supported on SimpleLink Sub-1 GHz CC1310 wireless MCU
• Star topology: point-to-point, one-to-many, and data concentrator
• Synchronous (beacon) and asynchronous (non-beacon) modes
• Designed for 915-MHz FCC, 863-MHz ETSI, and 433-MHz China bands
• SimpleLink long range mode for all supported frequency bands
• Support for SimpleLink CC1190
• Bluetooth low energy beacon advertisement support
• Sensor-to-web example application
• Easy application development guided through sample applications showcasing the stack configuration and APIs
• Coprocessor mode for adding connectivity to any MCU or MPU with Linux® host middleware and console application

For more details and to get the TI 15.4-Stack software, download the SimpleLink CC13x0 SDK, which includes the TI 15.4-Stack.

2.2.4 SimpleLink™ Wi-Fi CC3220 SDK

The SimpleLink Wi-Fi CC3220 SDK contains drivers for the CC3220 programmable MCU, 30+ sample applications, and related documentation. The SDK also contains the flash programmer, a command line tool for flashing software, configuring network and software parameters (SSID, access point channel, network profile, and so on), system files, and user files (certificates, web pages, and so on). This SDK can be used with TI’s SimpleLink Wi-Fi CC3220 LaunchPad development kits.

Features:
• Internet-on-a-chip sample applications:
  – Email from SimpleLink Wi-Fi
  – Information center: Get time and weather from the internet
  – https server: Host a secure web page on SimpleLink Wi-Fi
  – XMPP: IM chat client
  – Serial interface
• Wi-Fi sample applications:
  – Easy Wi-Fi configuration
  – Station, AP modes
  – TCP/UDP
  – Security—Enterprise and personal, TLS/SSL
  – Power management—Deep sleep, hibernate
• MCU peripheral sample applications:
  – Including parallel camera, I2S audio, ADC, I²C, PWMs, JTAG Flashing, and more
3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

The CC3220SF plus CC13x0 sensor-to-cloud TI Design helps developers create ULP, long-range, star-topology network solutions. The sensor-to-cloud TI Design includes the Gateway example application running on the CC3220SF, MAC CoProcessor (CoP) application running on the CC1310 or CC1350, in addition to sensor node applications. The CC3220SF Gateway example application interfaces over UART with a CC13x0 LaunchPad, which acts as a MAC CoP. The Gateway example application implements a IEEE 802.15.4 full-function device, which performs the functions of a network PAN coordinator (starting a network and permitting devices to join this network) and also provides an interface for monitoring and collecting sensor data from one or more sensor devices.

The Gateway example application provides an IEEE 802.15.4 network to IP bridge and is a great starting point to create IoT applications based on TI 15.4-Stack.

3.1.1 Hardware

- 2× CC1350 or 2× CC1310 LaunchPad development kits
- 1× CC3220SF LaunchPad development kit
- USB cables
- Wi-Fi AccessPoint with internet access

3.1.2 Software

- CC3220-SensorToCloud SW
- CC3220 SDK v1.30.01.03
- SimpleLink CC13x0 SDK v1.30
- UniFlash v4.1.1.1250 or later
- Tera Term or any other equivalent terminal program
- Cloud Foundry CLI
- (Optional) SimpleLink Starter Pro IOS® app or SimpleLink Wi-Fi Starter Pro Android™ app (downloaded from the app store on smartphones or tablets)
3.2 Testing and Results

This section describes the hardware and software used for running the tests and the results obtained.

3.2.1 Test Setup

During the development process of this TI Design, the full hardware and software portions described in earlier sections were used for testing. Multiple CC1310 and CC1350 sensor nodes and a CC3220SF LaunchPad (connected to a CC1310 coprocessor) were used to verify the IoT gateway functionality with the IBM cloud. The test results of this TI Design can be visualized by the IoT dashboard shown in Section 3.2.2.

3.2.1.1 Running the Out-of-Box Demonstration

This section makes use of pre-built binary files; no coding or building is required. This section provides detailed instructions to assist developers set up and understand the principles behind this demonstration.

1. Program the MAC-CoP LaunchPad
2. Program the sensor
3. Program the gateway on the CC3220SF LaunchPad
4. Connect the MAC-CoP LaunchPad with the CC3220SF LaunchPad
5. Open and configure an IBM Bluemix® Account
6. Setup Watson IoT platform service
7. Setup Node.js cloud foundry app
8. Run and use the example

**NOTE:** This guide can be performed using either CC1310 or CC1350 LaunchPad development kits for the MAC CoP. The training material is based on CC1350, so if using CC1310, all references to CC1350 in text and screen shots should be referred to as CC1310.

3.2.1.1.1 Label the LaunchPad™ Development Kits

Label one CC1350 LaunchPad as Sensor and the other as MAC-CoP. These labels will be referred to throughout this lab. It is recommended to use a non-permanent marking for this (for example, sticky notes), as these labels may only be relevant for this specific lab.

3.2.1.1.2 Program the MAC-CoP LaunchPad™

1. It is assumed that all the required software is already installed on a Windows PC. If not, install the required software now.
2. Connect the MAC-CoP LaunchPad to the PC.
3. Open UniFlash.
4. Select LAUNCHXL-CC1350 as shown in Figure 4, and click on the Start button.

![Figure 4. Choose Device: UniFlash](image-url)
5. Make sure the *Program* tab is selected on the right, and click the *Browse* button to select the desired image for the LaunchPad (CC1350 LaunchPad CoProcessor.hex or sensor_cc1350lp_doorlock.hex).

![Figure 5. Browse for Firmware Image](image-url)
6. After selecting the desired image, click on the Load Image button to flash the CC1350 LaunchPad.

![Figure 6. Load Image](image)

7. If loading the image was successful, a success message should show in the console as shown in Figure 7.

![Figure 7. Successful Load](image)

### 3.2.1.3 Program the Sensor LaunchPad™

Follow the same steps in Section 3.2.1.1.2 to program the Sensor LaunchPad using the hex image `sensor_cc1350lp_doorlock.hex` provided in the Sensor folder.

### 3.2.1.4 Programming the CC3220SF LaunchPad™

This section describes two ways of programming the CC3220SF LaunchPad. Section 3.2.1.1.4.1 explains the process of programming the out-of-the-box demonstration by importing a preconfigured Image Creator project to UniFlash. Section 3.2.1.1.4.2 shows how to create an Image Creator project from scratch to program the CC3220SF with a binary generated from Code Composer Studio™.
3.2.1.1.4.1 Programming the Preconfigured Image Creator Project

1. Open UniFlash.

2. On the Choose your Device section select **CC3220SF-LAUNCHXL**, and make sure to select the **Serial** option and not **On-Chip** as shown in Figure 8. Click on the **Start Image Creator** button.

![Figure 8. Choose Device: CC3220SF](image)
3. After starting Image Creator, click on the *Manage Projects* button as shown in *Figure 9.*
4. Click on the **Import Project from ZIP file** button, and select the zip folder 
C:\S2C_Repo_Directory\CC3220_CC13x0Gateway\prebuilt\CC3220SF_LaunchXL\Uniflash_CC3220ImageCreatorProject.zip.

**Figure 10. Import Project**
5. Open the **CC3220SF_154StackGateway** project.

---

**Figure 11. Open Project**
6. Connect the device to the PC through a USB cable, and press the Connect button found on the bottom-right corner. Once the device is connected, select the Generate Image button underneath the Disconnect button, and select Program Image (Create & Program).

![Figure 12. Generate Image](image-url)
3.2.1.1.4.2 Creating an Image Creator Project

The following steps will allow the user to customize the example and use the new, updated files instead of the pre-build ones.

1. Open UniFlash.
2. On the Choose your Device section, select CC3220SF-LAUNCHXL. Make sure the Serial option is selected and not On-Chip as shown in Figure 13. Click on the Start Image Creator button.

![Figure 13. Select CC3220 Device](image-url)
3. After starting Image Creator, click on the New Project button as in Figure 14.

Figure 14. New Project

4. Enter a project name, select CC3220SF in the Device Type drop-down menu, make sure device mode is in Develop, and click on the Create Project button.

Figure 15. Starting a New Project
5. Select Trusted Root-Certificate Catalog in the bottom-left corner and uncheck the Use default Trusted Root-Certificate Catalog box. Include the Source File (certcatalogPlayGround20160911.lst) and Signature Source File (certcatalogPlayGround20160911.lst.signed.bin) found in C:\ti\simplelink_cc32xx_sdk_1_30_01_03\tools\cc32xx_tools\certificate-catalog.

Figure 16. Trusted Root Certificates
6. Select Service Pack in the bottom-left corner and include the service pack bin (sp_3.3.0.0_2.0.0.0_2.2.0.4.bin) found in C:\ti\simplelink_cc32xx_sdk_1_30_01_03\tools\cc32xx_tools\servicepack-cc3x20.

![Figure 17. Service Pack](image-url)
7. Select User Files and include the *dummy-root-ca-cert* and *dummy-root-ca-cert-key* files by clicking on the Add File icon. These files can be found in C:\ti\simplelink_cc32xx_sdk_1_30_01_03\tools\cc32xx_tools\certificate-playground. Create a folder named www by clicking on the New Folder icon. This folder will contain all the web server required files.

**NOTE:** When adding the files do not select any of the options in the pop-up window—just click the Write button.

![Figure 18. User Files](image-url)
8. Include all the files and folders inside the C:\S2C Repo Directory>CC3220_CC13x0Gateway\examples\CC3220SF_LaunchXLcc3220sf_gateway_app/15_4_Stack_Gateway_CC3220SF_LAUNCHXL_tirtos_ccs/www folder into the www folder of the device.

Figure 19. www Folder
9. On the drop-down box in the top-right corner, select Select MCU Image, and press Browse. On the next menu select Private Key Name: and include the dummy-root-ca-cert-key. On the Certification File Name: select dummy-root-ca-cert from the list.
10. Connect the device to the PC through a USB cable, and press the Connect button found on the bottom-right corner. Once the device is connected, select the Generate Image button underneath the Disconnect button. Select Program Image (Create & Program).

![UniFlash Development Mode - Generate Image](image_url)

**Figure 21. Generate Image**
3.2.1.1.5 **Connect the MAC-CoP LaunchPad™ With the CC3220SF LaunchPad™**

1. Remove all jumpers at the center of the MAC-CoP LaunchPad except the Reset jumper highlighted in green. Also remove the VSENSE jumper highlighted in blue in Figure 22.

![Figure 22. CC1350 LaunchPad™](image)
2. Stack both LaunchPad development kits on top of each other as shown in Figure 23.

Figure 23. Stacked CC1350 and CC3220SF LaunchPad™ Development Kits

3. Connect a USB cable only to the CC3220 LaunchPad, and plug it in the PC.
4. Open a serial console (such as, PuTTy or Tera Term), select the COM port associated to the CC3220SF LaunchPad, and open the port.
3.2.1.1.6 **Open and Configure an IBM® Bluemix® Account**

1. It is assumed that all the required software is already installed on the Windows® PC. If not, install the required software now.

2. Go to [Create IBM Bluemix Account](https://www.ibm.com), and register a 30 day trial account with an email address.

3. Confirm the Bluemix account using the link provided by IBM through email.

4. Log in to the Bluemix account, and walk through the steps on the screen.

5. Once finished with the steps, click on the **Catalog** tab located on the top-right corner.

![Figure 24. IBM® Bluemix®](image-url)
6. After selecting the Catalog, click on **Cloud Foundry Apps** under Apps on the left menu.

![Figure 25. IBM® Bluemix® Catalog](image)

7. Choose **SDK for Node.js** from the options provided.

![Figure 26. Create Node.js Application](image)
8. Enter a name for the new Node.js application under the **App name** text box, and press the **Create** button on the bottom-right corner.

![Figure 27. Name and Create Application](image)

9. Once the app has been created, click on the **Catalog** tab located on the top-right corner.

![Figure 28. Go to Catalog](image)
10. After selecting the Catalog, click on **Internet of Things** under **Services** on the left menu.

![Figure 29. IBM® Bluemix® Catalog](image)

11. Choose **Internet of Things Platform** from the options provided.

![Figure 30. IoT Service Options](image)
12. Give the new platform a service name, and click on the Connect to drop-down menu. Select the cloud foundry app name created in the previous steps. Click the Create button on the bottom-right corner.

![Figure 31. Create IoT Platform](image)

3.2.1.1.7 **Set up Watson IoT™ Platform Service**

1. Click on the IBM Bluemix on the top-right corner, or use the drop-down menu and choose Dashboard to open up the dashboard.

![Figure 32. Go to Dashboard](image)
2. The cloud foundry apps and IoT services will be visible on the dashboard. Click on the IoT service.

![Dashboard](image)

Figure 33. Dashboard

3. On the Manage tab, click the Launch button.

![Launch IoT Platform](image)

Figure 34. Launch IoT Platform
4. Select Devices from the navigation bar on the left.

5. Select Add Device on the top-right corner.
6. Select **Create device type**.

![Figure 37. Create Device Type](image)

7. Select **Create gateway type**.

![Figure 38. Create Gateway Type](image)
8. Enter gateway as the Name, and keep clicking Next until gateway creation is complete.

9. Once finished, setup will return to the device creation page. Choose gateway on the Choose Device Type drop-down menu, and press Next.
10. Enter a Device ID, and press Next. Ignore the metadata, and press Next again.

![Image showing Device ID input](image1)

**Figure 41. Device ID**

11. On the Security tab, fill out the token field. Make note of this token, as this will be used for authenticating the device to the cloud. Click Next.

![Image showing Token field](image2)

**Figure 42. Token**
12. Keep clicking Next until a summary of the device credentials and information shows. Take a screenshot of this page, as this will be the last time the Authentication Token is visible.

![Add Device](image)

**Figure 43. Save Authentication Credentials**

### 3.2.1.1.8 Set up Node.js Cloud Foundry App

1. Open UniFlash.
2. Locate `C:\<S2C Repo Directory>\CC3220_CC13x0Gateway\examples`.
3. Open the `ibm_cloud_application` folder, and open the `Manifest.yml` with a text editor. Replace the `name` and `services` fields with the name of the cloud foundry app and Watson IoT platform.

```yaml
applications:
  - path: .
    memory: 256M
    instances: 1
    domain: mybluemix.net
    name: APP_NAME
    host: APP_NAME
    disk_quota: 1024M
    services:
      - SERVICE_NAME
```

**Figure 44. Manifest**
4. Inside the same folder, open the `package.json`, and replace the name field with the name of the cloud foundry app.

![package.json](image)

**Figure 45. Package**

5. Open a command console and navigate to the IBM-Cloud-Dashboard folder (`cd C:\S2C Repo Directory\CC3220_CC13x0Gateway\examples\ibm_cloud_application`).

6. Type in `cf api https://api.ng.bluemix.net`.

7. Log in to the created account: `cf login`.

8. Push the code to the IBM cloud foundry app (`cf push`).
9. Go back to the IBM Bluemix Dashboard by clicking IBM Bluemix on the top left.

**Figure 46. Dashboard**

10. The cloud foundry apps and IoT services created on the previous on the dashboard will be visible. Click on the cloud foundry app.

**Figure 47. Cloud Foundry App**
11. Click on the **Connections** tab to the left.

![Figure 48. Connections](image)

12. Click on the **View credentials** button. This page shows the information required to establish a connection between the cloud front end and the back end server. Screenshot or save the information for later.

![Figure 49. View Credentials](image)
13. Now open the webpage, the link can be found on the dashboard next to the cloud foundry app name.

![Figure 50. IBM® Web Page Link](image)

14. If everything setup correctly, the dashboard will be visible. At this point, open the configuration menu located on the top-left corner.

![Figure 51. Sensor2Cloud Front End](image)
15. A form will pop up. Use the information saved in step 13 to fill out the form. For the Device Type and Device ID, use the information saved in Section 3.2.1.1.3. Save the changes, and close when done.

Figure 52. IBM® IoT Credentials
3.2.1.1.9 Run and Use the Gateway

There are two ways to get the S2C Gateway up and running. The first method is described in Section 3.2.1.1.9.1, which explains how to provision the CC3220SF LaunchPad to a WiFi Network from the Simple Link Starter Pro App. The second method is explained in Section 3.2.1.1.9.2; this method uses the built-in, local provisioning web page.

3.2.1.1.9.1 Using the SimpleLink™ Starter Pro App

This section assumes that either the IOS or the Android app is already installed on the user's mobile phone. If not, install the app now.

1. Launch the SimpleLink Starter Pro App from the phone.
2. If the device is not found automatically by the app, go to Device to configure, and tap on search for your device.

3. Wait for the app to find the device to connect. The name should be something like mysimplelink-XXXX.
4. Select the device to connect, and tap **OK**.

![Select Device to Configure](image)

**Figure 54. Select Device to Configure**
5. Select the desired Wi-Fi network to connect with the CC3220SF LaunchPad. Enter the password, tap **OK**, and then tap **Start Configuration**.

![Select Wi-Fi Router](image)

**Figure 55. Select Wi-Fi Router**

6. After the configuration is done, make sure the phone is connected to the same Wi-Fi network that the CC3220SF is connected.
7. Once connected, select the device from the device list.

Figure 56. Devices

8. This will open a web page hosted by the device with some information about the device network.
15.4 Stack Embedded Sensor to Cloud

Welcome to your board web client!

Start

Figure 57. Local Web Server Start Page

NOTE: If the browser gives a warning about security certificates, ignore it, and continue to the web page.

9. Press the Start button in the middle, which will direct to the dashboard. Click on the top-right button, and input the IBM cloud information.

Figure 58. Local Dashboard
10. A form will pop up, and fill out the form with the IBM Bluemix account information. When complete, click Save Changes and then Close.

![Cloud Credentials Form](image1)

**Figure 59. Cloud Credentials Form**

11. Go to the URL provided on the IBM Bluemix account, which looks something like `APP_NAME.mybluemix.net`.

12. Click the open button on the dashboard to allow sensors to join the network.

![Cloud Dashboard Open Button](image2)

**Figure 60. Cloud Dashboard Open Button**

13. Apply power to the LaunchPad labeled Sensor.

14. Now the sensor should automatically start looking for a network. If paired with the network successfully, the Sensor board can be viewed and controlled from the web browser.
NOTE: If the device is not visible in the web browser, the device is most likely connected to another network. To solve this error, complete a factory reset on the sensor by pressing the reset button while holding the right button (BTN-2), and try again.

Figure 61. Sensor2Cloud Front End

3.2.1.9.2 Using the Local Provisioning Web Page

NOTE: This is an alternate option if using the SimpleLink App is not desired.

1. Make sure the CC3220SF LaunchPad is powered on and that a serial console, like Tera Term, is opened to see the console output of the device.
2. On the PC search for Wi-Fi networks, and connect to the one broadcasted by the LaunchPad, which should look something like *mysimplelink-XXXX*.

![Figure 62. Connect to LaunchPad™](image)
3. Open a browser window, and navigate to `mysimplelink.net`, as shown in Figure 63.

![Figure 63. mysimplelink.net Local Home Page](image)

4. Press the **Start** button in the middle to go to the **Network Configuration** page.

![Figure 64. Network Configuration Page](image)
5. Enter the required information for the desired Wi-Fi access point, and click on the Add button. A pop-up window with instructions like Figure 65 will appear.

![Figure 65. Pop-up Message]

6. Go to Using SimpleLink Starter Pro App at the beginning of Section 3.2.1.1.9, and follow the instructions starting from step 8.
3.2.2 Test Results

3.2.2.1 IoT Dashboard

Figure 66 is an example of the IoT dashboard displayed on the web interface. Note that the current network information is shown. The network chart displays the number of connected devices, and the sensor nodes section shows the device and current sensor information for all the devices in the network.

![Sensor To Cloud Dashboard Results](image)

3.2.2.2 TI IoT Gateway-to-Cloud Service Interface

The purpose of this section is to provide a description of the message types and expected data flows that will be shared between the TI IoT gateway and an IoT cloud server. The interface is designed to be flexible to support multiple cloud vendors. For this purpose, the Sub-1 GHz wireless network and node information will be exchanged between the gateway and the cloud using the long-established JavaScript object notation (JSON) format. Additionally, IPSO alliance smart object definitions will be used to define sensors (and their data) that are connected to each node in the wireless networks.

3.2.2.2.1 Message Types

To fully specify the Sub-1 GHz wireless network information, as well as the Sub-1 GHz sensors and their data, two distinct message types have been defined for the IoT gateway to update the cloud. In order to allow the cloud to send messages back to the TI IoT gateway, two additional message types are defined that allow the cloud to update the wireless network state and send actuation messages to specific devices in the network.

3.2.2.2.1.1 Network Information Message Type (From TI IoT Gateway to the Cloud)

This message type presents information about the wireless network, its current state, and a list of devices that are connected to the network. As described later in this design guide, this will be the first message type sent after the network is initialized. This message type contains all the information necessary to prepare for receiving sensor data from devices. This message type contains the following fields:

- **name**: begins as the short address of the network but allows for the cloud to provide a more specific name
- **channels**: list of channels that the wireless network is operating on
- pan_id: the 16-bit PAN identifier of the network
- short_addr: the 16-bit short address of the PAN-coordinator
- ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
- security_enabled: yes, if security enabled; no, otherwise
- mode: network operation mode (beacon, non-beacon, frequency hopping)
- state: PAN-coordinator state values (waiting, starting, restoring, started, open, closed)
- devices: list of wireless nodes in the network
  - name: begins as the short address of the device but allows cloud to update
  - short_addr: the 16-bit short address of the PAN-coordinator
  - ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
  - topic: the topic that the device will send its sensor data updates to
  - object_list: list of IPSO alliance smart objects (sensors) attached to this device
    - oid: object ID which specifies the sensor type in the IPSO standard
    - iid: list of instance IDs for the current object (can be multiple same type sensors)

3.2.2.2.1.2 Device Information Message Type (From TI IoT Gateway to the Cloud)

This message type provides information about the wireless device as well as the latest data for all of the sensors connected to the device. This message type will be sent when a device reports sensor data or switches between an active or inactive state. The following fields are contained in this message type:
- active: whether or not the wireless node is active
- ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
- rssi: received signal strength indicator of the last message received
- smart_objects: list of the IPSO alliance smart objects connected to this wireless device
  - object ID description: type of sensor (as defined in the IPSO standard); can be multiple types of sensors connected to each device
  - instance ID: the instance ID for the parent object type; can be multiple sensors of the same type
  - resource ID description list: sensor data name value pairs (for example, sensorValue: 32.5, units: Celsius, and so forth); these resources match what is specified for the given object ID in the IPSO standard

3.2.2.2.1.3 Update Network State Message Type (From Cloud to TI IoT Gateway)

In the current implementation of the TI IoT gateway, this message type is intended to be able to open or close the wireless network to new devices joining. The cloud’s front end user interface can allow a user to click a button to open or close the network and then generate this message type and send it to the TI IoT gateway. The gateway will then notify the network on whether it needs to open or close to new device joins. This message type only includes the desired state of the network and should be sent to the same topic that the cloud is receiving the network information messages from. The following field is all that is required:
- state: should be set to either open or closed

3.2.2.2.1.4 Device Actuation Message Type (From Cloud to TI IoT Gateway)

This message type is added to allow the cloud to send actuation messages to specific devices in the wireless network. The current implementation only supports toggling an LED on the wireless device’s board. The device actuation message should be sent to the topic of the device as given in the devices list of the network information message. The following field is the only requirement for this message:
- toggleLED: should be set to true
3.2.2.2.2 Data Flows

3.2.2.2.2.1 Network Information Sent to the Cloud

The following bulleted items are the list of events that can occur on the TI IoT gateway that will cause a network information message type to be sent to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

- **Network Startup**
  This is the initial event in the TI IoT gateway. The TI IoT gateway will aggregate the information about the wireless network as well as the list of connected devices and their sensor types. The TI IoT gateway will then make a connection to the cloud and send the aggregated data contained in the network information message type.

- **Network Information Update**
  This event can occur if any of the information about the wireless network changes. For example, if the network operation mode of the wireless network was changed, the TI IoT gateway would once again aggregate all the information needed (network information and device list) and send the network information message type to the cloud.

- **Network State Change**
  This event occurs if the state of the wireless network changes. For example, if the network state changes from open to closed the TI IoT gateway will send a network information message type to the cloud.

- **Device Joins the Wireless Network**
  When a new device joins the network, after the network is up and running, this event will occur. In this case, the TI IoT gateway will add the new device and its information to the devices list within the network information message type and then send the updated information to the cloud.

- **Expected Cloud Behavior**
  It is expected that the cloud will be prepared for the network startup event and will be able to receive the network information message type (using a wildcard and then filtering or by having prior knowledge about the destination or topic of the message). Once the cloud receives the network information message, the wireless network information (PANID, security, mode, and so on) can be displayed to users and the device list information (topic, object list, and so on) can be used to prepare itself to receive and display device and sensor data.

3.2.2.2.2.2 Device Information Sent to the Cloud

The following bulleted items are the list of events that will cause the TI IoT gateway to send a device information message type to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

- **Device Becomes Inactive**
  This event occurs when the TI IoT gateway detects that one of the devices in the connected devices list has stopped sending sensor data updates. The TI IoT gateway will update the active field and send a Device Information Message Type to the cloud for the inactive device.

- **Device Reports Sensor Data**
  Each time a sensor on a connected device reports sensor data this event occurs. The TI IoT gateway updates the IPSO Alliance Smart Object list in the device for each sensor and then sends a device information message type to the cloud.

- **Expected Cloud Behavior**
  It is expected that the cloud will be listening on each topic given in the connected devices list from the network information message. When one of the two events occur in this section, the TI IoT gateway will send the device information message to the topic (corresponding to the device being update) that the cloud should be listening on or subscribed to. When the device information message arrives at the cloud, the cloud should display the latest device information and sensor data to users.
### 3.2.2.2.2.3 Update Network State Message Sent to the TI IoT Gateway

This message is used to open or close the wireless network to new devices joining. This message should be an option provided to users in the front end user interface that the cloud presents. When the user decides to update the network state, the cloud should send an update network state message type to the TI IoT gateway on the same topic that the network information messages are arriving on.

- **Expected TI IoT Gateway Behavior**
  
The TI IoT gateway will receive the Update Network State message and will generate the correct command (either open or close) to the wireless network. This message should in turn cause a network state change event (from 7.2.1 above) that will send a network information message back to the cloud which can confirm the successful completion of the Update Network State command.

### 3.2.2.2.4 Device Actuation Message Sent to the TI IoT Gateway

This method is used to toggle the LED on the board of the connected devices. This message is meant to be a proof-of-concept on the current device setup and will change for customer use-case specific actuations. A toggle LED button for each device will be provided to users of the cloud’s front end interface. When the toggle LED button is clicked the cloud should send a device actuation message to the TI IoT gateway on the same topic that the device information messages are arriving on.

- **Expected TI IoT Gateway Behavior**
  
The TI IoT gateway will generate a toggle LED command and send it to the device corresponding to the topic that the device actuation message was received on. This will cause the LED to toggle. Because the state of the LED is not captured in the device information message type, there will be no feedback to the cloud that the LED actually toggled.
4 Design Files

This TI Design showcases the connectivity between CC3220SF and CC13x0 devices. The CC3220SF acts as a gateway processor and the CC13x0 as communication node. The CC3220SF LaunchPad is used as a platform for gateway processor, and the CC13x0-based LaunchPad acts as communication node. The recommended schematics for this TI Design uses the schematics of the CC3220SF LaunchPad and CC13x0 LaunchPad and interface the two devices using the UART lines. This IoT gateway reference design only uses one UART port. In addition, bootloader backdoor pins are described in the CC2538/CC26xx Serial Bootloader Interface application report. These pins can be connected to upgrade the firmware on the CC13x0 using the serial ROM bootloader on the CC13x0 devices.

4.1 Schematic

To download the schematics for this reference design, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

4.2 Bill of Materials

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

4.3 PCB Layout Recommendations

For layout prints, Altium project files, Gerber files, and assembly drawings, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

5 Software Files

To download the software files for this reference design, see the link at https://git.ti.com/tidc01002/tidc01002.

6 Related Documentation

1. Texas Instruments, SimpleLink TI 15.4-Stack IEEE 802.15.4e/g Standard Based Star Networking Software Development Kit, Tools Folder
2. Texas Instruments, SimpleLink CC3220 SDK, Tools Folder
3. Texas Instruments, TI 15.4-Stack Wiki, Wiki Page
4. Texas Instruments, TI 15.4-Stack Embedded Developers Guide

6.1 Trademarks

SimpleLink, LaunchPad, Internet-on-a-chip, SmartConfig, Code Composer Studio are trademarks of Texas Instruments Incorporated.
Cortex is a registered trademark of ARM Limited.
ARM is a registered trademark of ARM Ltd.
Bluetooth is a registered trademark of Bluetooth SIG, Incorporated.
IOS is a registered trademark of Cisco.
Android is a trademark of Google Inc..
Watson IoT is a trademark of International Business Machines Corporation.
IBM Watson IoT, Bluemix are registered trademarks of International Business Machines Corporation.
Linux is a registered trademark of Linux Foundation.
Windows is a registered trademark of Microsoft Corporation.
Wi-Fi, Wi-Fi Direct are registered trademarks of Wi-Fi Alliance.
About the Authors

HECTOR RAMOS is a software applications engineer at Texas Instruments, where he has been working at since 2014 and is specialized in low-power wireless protocols. Hector earned his bachelors of science in electrical engineering and computer science from University of California Berkeley.

ANDRES BLANCO is an applications engineer at Texas Instruments, where he is responsible for supporting customers designing low power wireless systems. Andres earned his bachelor of science in computer engineering from University of Puerto Rico in Mayagüez, PR.
IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI’) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT. AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include, without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/samptersm.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated