Design Guide: TIDM-1020

SimpleLink™ CC3220 Wireless MCU-Based Thermostat Reference Design

Description

Creating a low-power, connected, microcontroller (MCU) based smart thermostat that can securely link a variety of sensors to the cloud, which enables remote monitoring and control, is the goal of most smart-thermostat designers. The TIDM-1020 provides a software reference for the implementation of such a smart thermostat, using the CC3220 device as the primary MCU, with Wi-Fi® connectivity. This application note provides the details of adding Bluetooth Low Energy (BLE) connectivity for provisioning to TIDM-1020. This reference design is for thermostat end-equipment developers, engineers, and system evaluators. The design provides reference code to demonstrate integration of the CC3220SF device into a variety of analog and digital sensors, cloud-connectivity services, human machine interfaces (HMIs), passive infrared (PIR) sensors, and external relay controls. This reference design is also intended to showcase: low-power connection to the Internet and cloud, remote control and programming, remote monitoring of the data, and secure Over The Air (OTA) update of the device and application firmware.

Resources

TIDM-1020	Design Folder
CC3220SF-LAUNCHXL	Product Folder
LAUNCHXL-CC2640R2	Product Folder
BOOSTXL-SENSORS	Product Folder
BOOSTXL-K350QVG-S1	Product Folder
SEEED STUDIO GROVE	Product Folder

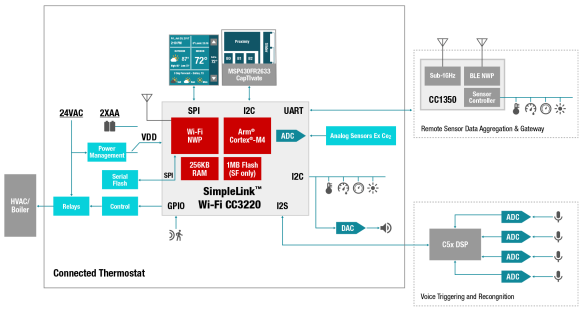
Ask our TI E2E™ support experts

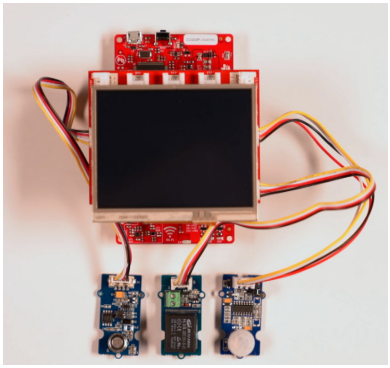
Features

- Demonstrates Battery Life of ~6 Months Using 2 AA Batteries
- HMI Through Resistive Touchscreen
- Cloud Connectivity for Remote Control and Sensor Data Aggregation and Monitoring
- Demonstrate the Security Features
  - Secure OTA Update
  - Secure Server
  - Internal HTTPS Server
  - Secure Sockets (SSL/TLS)
- Provisioning (BLE based, Access Point (AP) Mode and SmartConfig™)
- Low-Power Capabilities
- Configurable Sensor Update Rate

Applications

- Thermostat
- HVAC System Controller
- Boiler System
- Weather Station
- Wireless Environmental Sensor
- Air Quality and Gas Detection





1 System Description

The TIDM-1020 provides a detailed reference implementation for a smart thermostat using the SimpleLink™ MCU-based CC3220 family of devices. The CC3220SF device combines the functionality of a MCU, with a Wi-Fi network processor, to enable best-in-class, low-power, Internet of things (IoT) devices, with enhanced security capabilities, 256KB RAM, and 1MB flash. The design shows how a developer can use the CC3220SF device as the main MCU, to accomplish smart-thermostat functionality, with connectivity to the Internet through Wi-Fi. The design also includes provisioning using BLE to simplify setting up the end equipment.

Internet and cloud connectivity are the primary needs for a smart thermostat. The SimpleLink CC3220 Wireless MCU offers a complete package; with it, the user can create an end node (or end point), which is connected to the cloud and the Internet. The design demonstrates connection to the cloud capability, and the SimpleLink MCU software development kit (SDK) has support for a variety of cloud-service providers, such as IBM Watson®, Amazon Web Services®, Microsoft Azure®, Apple HomeKit®, and more. The same reference, with a list of Internet protocol support (HTTP, HTTPS, MQTT, DHCP, SSL/TLS, SMTP, and SNTP), can enable multiple cloud vendors and Internet connectivity.

Security is another key feature highlighted in this design. The design provides a reference implementation of the OTA-firmware update feature over a secure connection and a secure data transfer to the cloud. This implementation serves as a reference for using the APIs needed to build security in end equipment. The design also demonstrates the following key security feature

- Secure boot of the application code
- AP provisioning using WPA2 authentication and HTTPS
- Secure socket connections to the cloud for data transfer and OTA update
- OTA update using multiple security enablers:
  - File Integrity Check
  - Signature verification
  - Failsafe files
  - Bundle protection

Many of the features demonstrated in the TIDM-1020 application are based on the networking application libraries available in the SimpleLink SDK, including SNTP, MQTT, and HTTP. These protocols and the features they are used to implement in the software are applicable to many other building automation systems that integrate Wi-Fi connectivity.

For this design, the TIDM-1020 uses existing EVMs, the LaunchPad™ (LP) and BoosterPack™, which are available online for purchase. This selection was done with the developer in mind, to allow focus on system and application-software development without requiring custom hardware.

The TIDM-1020 also uses the common SimpleLink SDK platform for all firmware development. This common SDK code can easily be used with other SimpleLink connectivity platforms as well as the MSP432™ platforms. The entire design, including the firmware and hardware components, can be reconfigured or modified to suit specific system requirements and allow for scalability, should additional connectivity needs arise in the future.

A detailed outline of the smart-thermostat features as well as references for implementing the thermostat using the CC3220 family of devices, are provided in the Designing Thermostats With CC3220 SimpleLink™ Single-Chip Wi-Fi® MCU System-on-Chip Application Report. The scope of this TI reference design demonstrates the implementation of the general key features required for a basic smart thermostat. Other features can be added with little to no overhead.

Figure 1-1 shows the overall system-level implementation of the TIDM-1020.

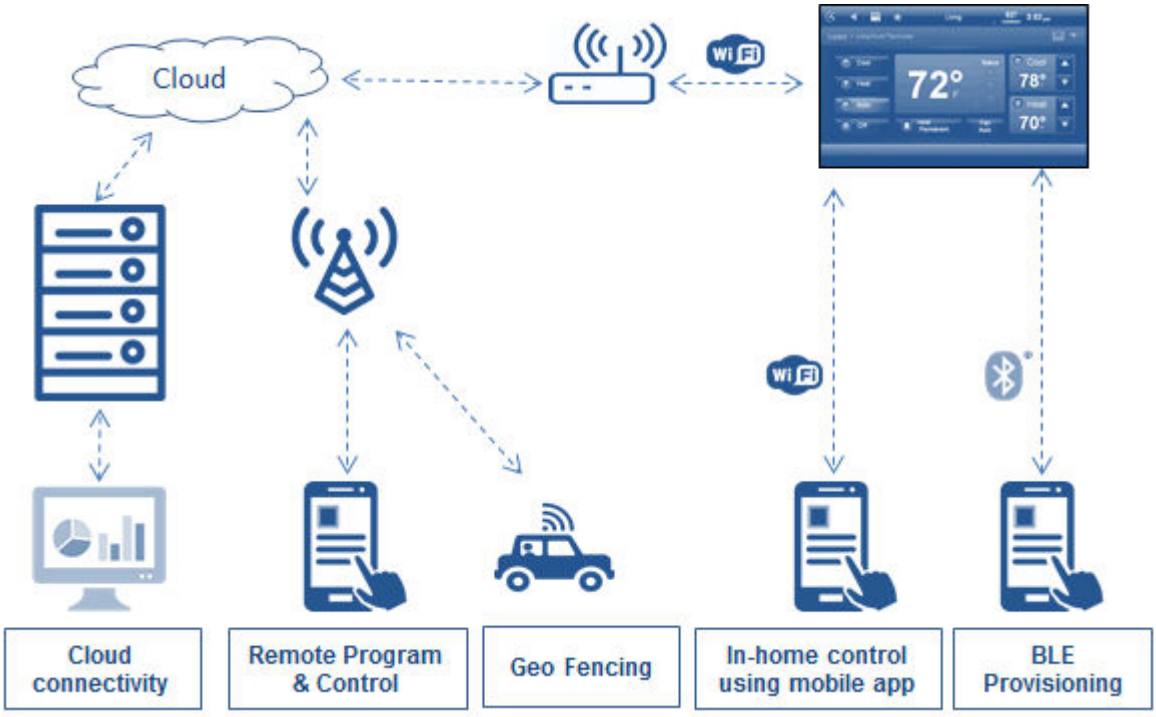


Figure 1-1. Smart-Thermostat System-Level Block Diagram with BLE Provisioning

1.1 Key System Specifications

Table 1-1 lists the key system specification and features of the TIDM-1020.

Table 1-1. Key System Specifications

Feature	Specification	Additional Details
Sensor integration	Temperature, humidity, air quality, atmospheric pressure	The TIDM-1020 demonstrates the integration of both analog and digital sensors. In this design, an air quality sensor is interfaced as an analog sensor.
Cloud connectivity	IBM cloud services	Interface to the cloud server over MQTT.
Provisioning	BLE, AP mode and SmartConfig	Uses SimpleLink Wi-Fi Starter Pro mobile application for Apple iOS or Android™ devices.
HMI	HMI using a Kentec display and resistive touchscreen	—
Serial interface requirement	I^2C – all digital sensors SPI – interface to HMI ADC – analog sensors GPIOs – relay and LED control	This design can be used as a reference for the peripherals listed.
PIR sensor	Proximity-based display turn-on feature	—
Memory use – code	260 KB	Integrated development environment (IDE): Code Composer Studio™ (CCS) RTOS: TI-RTOS Compiler: TI v18.1.1.LTS

2 System Overview

2.1 Block Diagram

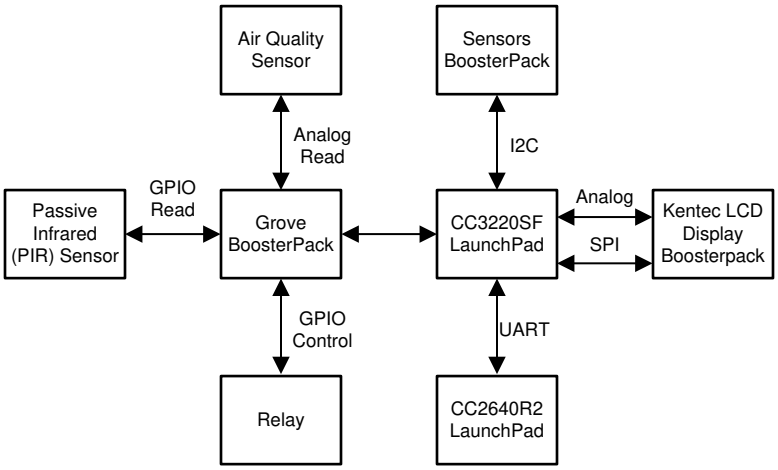


Figure 2-1. TIDM-1020 Block Diagram with BLE Provisioning

2.2 Highlighted Products

The two TI products highlighted in this reference design are the CC3220SF SimpleLink Wi-Fi and IoT, Single-Chip Wireless MCU Solution and the CC2640R2F SimpleLink BLE Wireless MCU.

2.2.1 SimpleLink™ Wireless MCU CC3220SF

The CC3220 device is a part of the SimpleLink MCU platform, which consists of Wi-Fi, Bluetooth® 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. Start your IoT design with a Wi-Fi CERTIFIED®, single-chip, wireless MCU, system-on-chip (SoC), with built-in Wi-Fi connectivity. Created for the IoT market, the SimpleLink Wi-Fi CC3220 device family from TI is a single-chip solution, integrating two physically separated, on-chip MCUs.

- Application processor – Arm® Cortex®-M4 MCU, with a user-dedicated 256KB of RAM and 1MB of Execute in Place (XIP) flash
- Wi-Fi CERTIFIED network-processor MCU, to run 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 and secure Internet connections with 256-bit encryption.

The CC3220 wireless MCU family is a part of the second generation of the Internet-on-a chip™ family of solutions from TI. This generation introduces new features and capabilities that further simplify the connectivity of data to the internet. The new capabilities including the following:

- IPv6
- Enhanced Wi-Fi provisioning
- Enhanced power consumption
- Enhanced file-system security (supported by only 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 WPA2 personal and enterprise security, WPA2 + PMF, and WPA3. This subsystem includes embedded TCP/IP and TLS/SSL stacks, an 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.

The power-management subsystem includes integrated DC-DC converters that support a wide range of supply voltages. This subsystem enables low-power consumption modes for extended battery life, such as low-power deep sleep, hibernate with RTC (consuming only 4.5 µA), and shutdown mode (consuming only 1 µA). The devices also have the Network Learning Algorithm from TI, which dynamically learns the behavior of the AP, to ensure best-in-class low power, as tested with over 200 different APs.

The device includes a wide variety of peripherals, including a fast parallel-camera interface, I2S, SD, UART, SPI, I^2C, and a 4-channel 12-bit ADC.

The SimpleLink CC3220x family of devices comes in three device variants: CC3220R, CC3220S, and CC3220SF.

The CC3220R and CC3220S devices include 256KB of application-dedicated embedded RAM for code and data, ROM with an external serial-flash bootloader, and peripheral drivers.

The CC3220SF device includes an application-dedicated 1MB of XIP flash and 256KB of RAM for code and data, ROM with an external serial flash bootloader, and peripheral drivers. The CC3220S and CC3220SF device options have additional security features, such as encrypted and authenticated file systems, user IP encryption and authentication, secured boot (authentication and integrity validation of the application image at flash and boot time), and more.

The CC3220 family of devices is a complete platform solution that includes software, sample applications, tools, user and programming guides, reference designs, SimpleLink Academy online training, and the TI E2E online community. The device family is also a part of the SimpleLink MCU portfolio and supports the ecosystem of the SimpleLink developer.

2.2.1.1 Functional Block Diagram

Figure 2-2 shows the CC3220x device hardware overview and embedded-software overview.

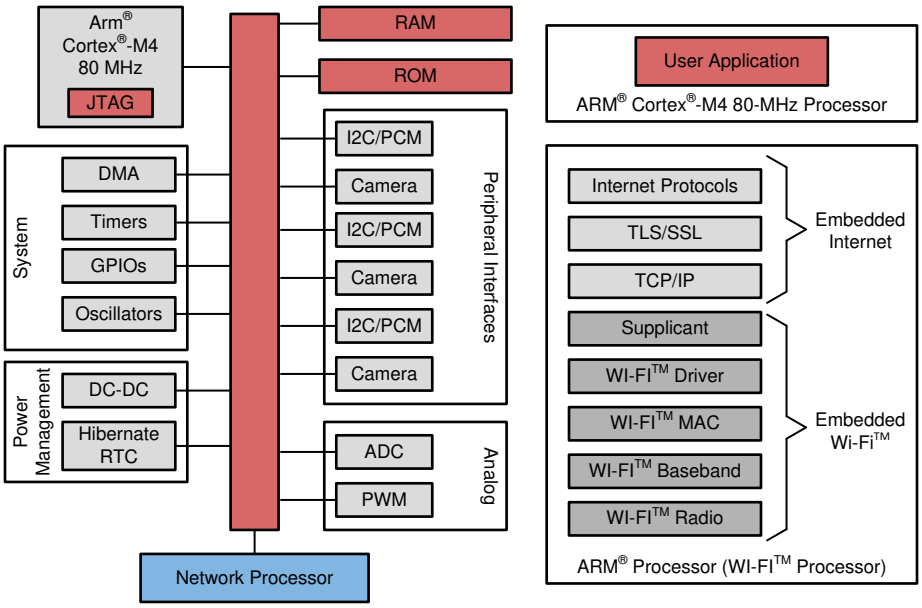


Figure 2-2. CC3220x Device Block Diagram

2.2.2 SimpleLink™ Bluetooth® Low Energy Wireless MCU CC2640R2F

The CC2640R2F device is a wireless MCU targeting Bluetooth 4.2 and Bluetooth 5 low-energy applications. The device is a member of the SimpleLink ultra-low power CC26xx family of cost-effective, 2.4-GHz RF devices. Ultra-low active RF and MCU current and low-power mode current consumption provide excellent battery lifetime and allow for operation on small, coin-cell batteries and in energy harvesting applications. The SimpleLink BLE CC2640R2F device contains a 32-bit Arm® Cortex®-M3 core, which runs at 48 MHz, as the main processor and a rich peripheral feature set that includes a unique ultra-low power sensor controller. This sensor controller is ideal for interfacing external sensors and for collecting analog and digital data autonomously, while the rest of the system is in sleep mode. Thus, the CC2640R2F device is great for a wide range of applications where long battery lifetime, small form factor, and ease of use are important. The power and clock management and radio systems of the CC2640R2F wireless MCU require specific configuration and handling by the software to operate correctly, which has been implemented in the TI-RTOS. TI recommends using this software framework for all application development on the device. The complete TI-RTOS and device drivers are offered in the source code, free of charge from www.ti.com. Bluetooth low energy controller and host libraries are embedded in the ROM and run partly on an Arm® Cortex®-M0 processor. This architecture improves overall system performance and power consumption and frees up significant amounts of flash memory for the application. The Bluetooth stack is available, free of charge from www.ti.com.

2.3 System Design Theory

The TIDM-1020 implements a smart thermostat, with the SimpleLink CC3220SF device as both the host and the Wi-Fi connectivity device. The reference design is a GUI-based, cloud-connected thermostat. The key highlights of the design are follow:

- Local control using the touchscreen
- Remote control through cloud connectivity using a web-based application
- Periodic sensor data upload to the cloud
- Use of analog and digital sensors
- Secure OTA update
- Provisioning (BLE Provisioning, AP mode, and SmartConfig)
- Low-power capabilities
- Configurable sensor-update rate
- GPIOs to activate peripherals and protocol interface of the HVAC system

Figure 2-3 shows the reference block diagram of the smart-thermostat end equipment.

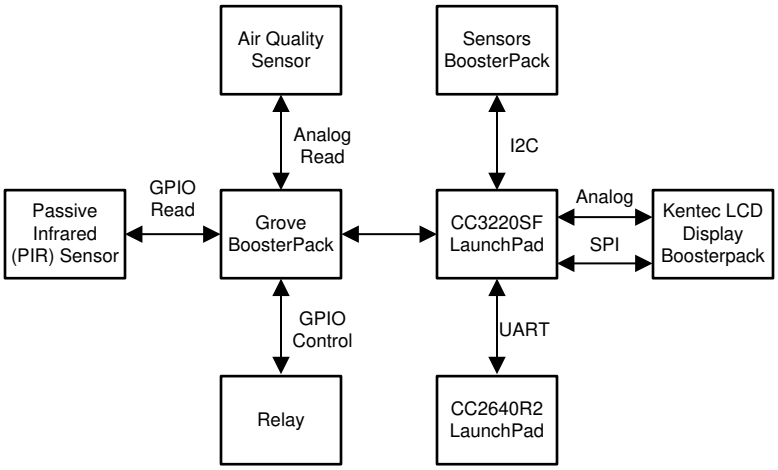


Figure 2-3. TIDM-1020 Block Diagram

The following sections contain details of the implementation of these features and the hardware. The following sections also contain the system-level requirement details of each feature. The hardware and software requirements are detailed in separate sections.

2.3.1 Local Control and GUI Design

The thermostat design uses a GUI to accept user input and to output data and feedback. The display is a BOOSTXL-K350QVG-S1 Kentec 3.5" liquid crystal display (LCD), (Kentec LCD BoosterPack) with a resistive-touch interface. The display interfaces with the CC3220SF LaunchPad through SPI. The touch interface is enabled by GPIO inputs and provides analog outputs.

Figure 2-4 shows the main screen of the demonstration platform. This screen is shown after network connection and calibration, and by default times out after one minute without triggers from the PIR sensor. The target temperature is adjustable using the touchscreen buttons above and below the target temperature. The fan icon and forecast details are selectable through the touchscreen interface.

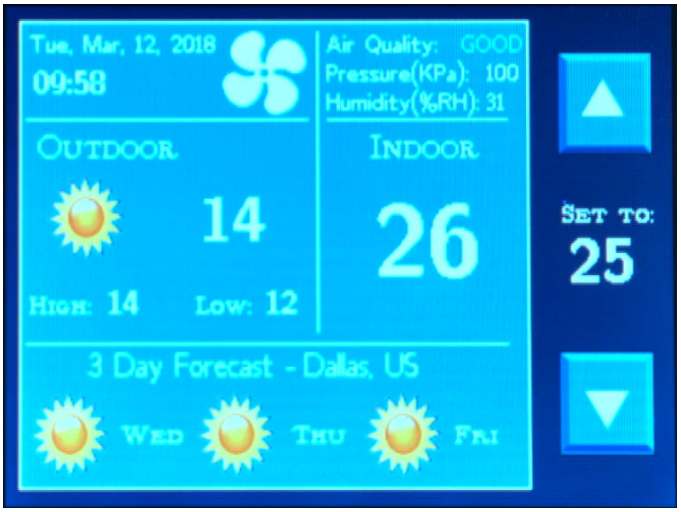


Figure 2-4. Smart-Thermostat HMI Default Window

The flow of activities in this demo is controlled based on the GUI. The background color of the screen indicates the HVAC-system mode of operation (heating or cooling). The HVAC-system mode is based on the user-temperature settings, current-temperature readout, and user input action. Figure 2-5 shows the user actions for this design.

The display software is developed using the grlib graphics library. Customers can use this library to interface with the display and develop a custom GUI.

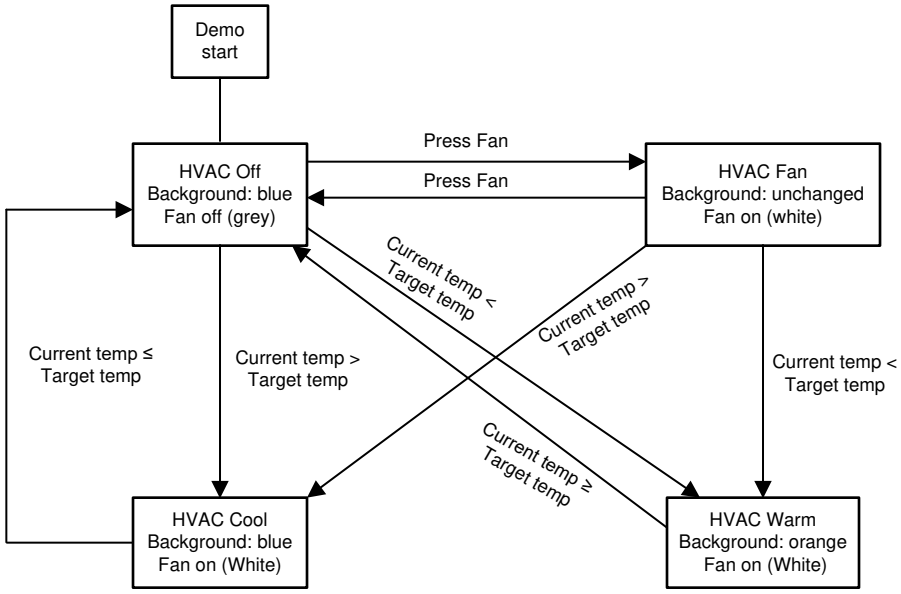


Figure 2-5. State Diagram of HMI Event Flow

2.3.1.1 Touch-Position Detection and Touchscreen Calibration

The resistive touchscreen is composed of two conductive layers separated by a gap and an insulated layer below. The two top conductive plates are coated with conductive material that provides uniform resistance across the layer. When a voltage is applied to the layer, it produces a uniform linear gradient of voltage. When the screen is touched, the pressure from the touch forces the two conductive layers to come in contact. By applying different voltage to the two conducing planes, the point of touch can be determined. Figure 2-6 shows the mechanism of detecting X and Y positions on the touchscreen.

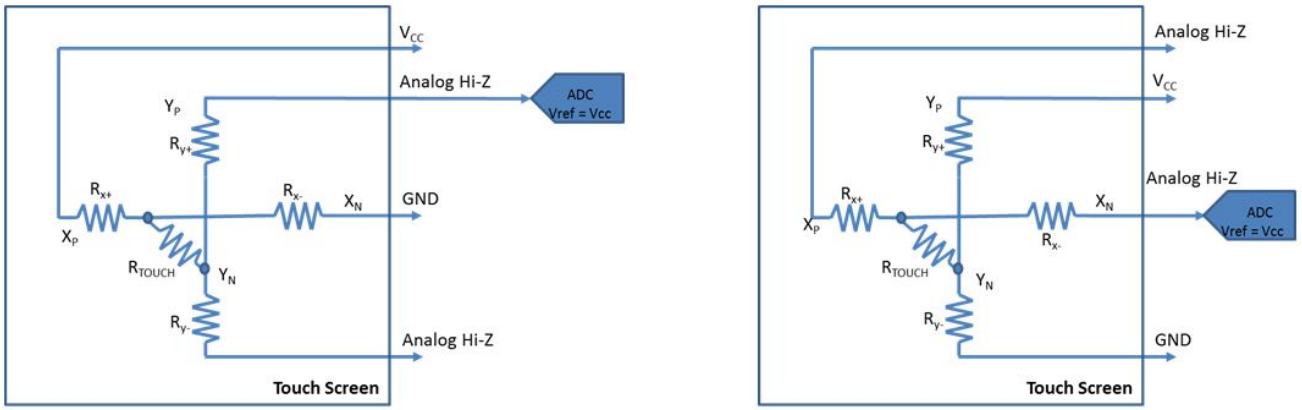


Figure 2-6. Configuration to Read XY Position of Touch Sensor

The TIDM-1020 uses the GPIOs and 2 channels of the ADC to read the X and Y position of the touchscreen. Multiplexing of the pins for the measurement is dynamically done. The resistors on the board must be modified to match the supply voltage.

The touchscreen must be calibrated to check if the orientation screen is correct. This can be done using a three-point touch calibration screen at demo start up.

2.3.2 Provisioning Device for Wi-Fi® Connectivity

The provisioning process provides the SimpleLink Wi-Fi device with the information (network name (SSID), password, and security type) needed to connect to a wireless network. The process usually occurs only once, letting end users connect their devices to their local networks for the first time. Providing this information may become challenging, because all IoT devices are not equipped with conventional-input peripherals such as a keyboard or touchscreen.

The TIDM-1020 offers the following provision modes:

- AP provisioning – a provisioning method in which the device creates a wireless network of its own, allowing a PC or smartphone to connect to it directly, and provides its initial configuration.
- SmartConfig technology provisioning – a proprietary provisioning method from TI that uses a smartphone or tablet to broadcast the network credentials to an unprovisioned device.
- BLE Provisioning - Wi-Fi provisioning over BLE is another method of provisioning a Wi-Fi based system to a new Wi-Fi network. The main benefit of provisioning a system over BLE is that it can provide a more consistent and streamlined user experience when provisioning from a smartphone or tablet.

Figure 2-7 shows the provisioning process within the system. The details of the provisioning implementation are covered in Section 3.2.1.

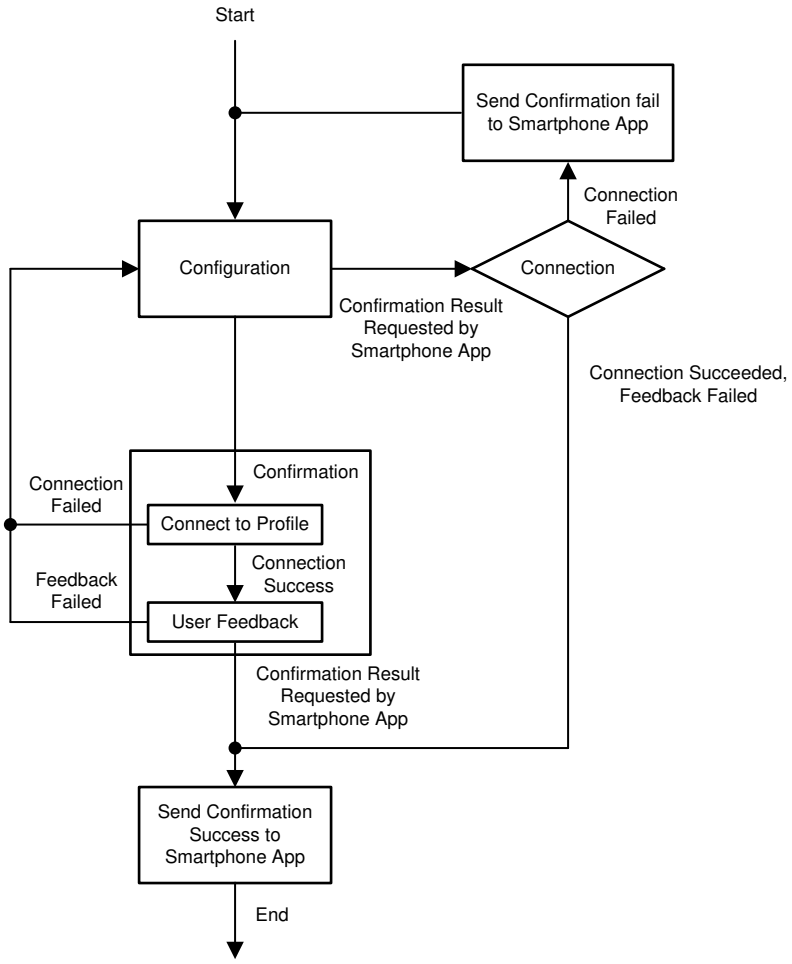


Figure 2-7. Provisioning-Process Flow Chart

2.3.2.1 AP Provisioning and SmartConfig™

AP provisioning is the most common provisioning method. During AP provisioning, an unprovisioned Wi-Fi device temporarily operates as an AP. This method allows a device, such as a smartphone, tablet, or PC, to connect to the AP over Wi-Fi and transmit the information for the desired network connection directly to the device. The CC3220 device receives the network credentials through a Restful API based on its internal HTTP server.

It is important to secure the connection between the device sending the network credentials and the CC3220 device during AP provisioning. To protect the network credentials and ensure a user does not unknowingly transmit them to an unwanted device, the TIDM-1020 uses WPA2 authentication when a station connects to the CC3220 AP. The TIDM 1020 is also configured to direct incoming connections to a secured HTTP port and establish a TLS session over which the credentials are passed.

SmartConfig is a proprietary provisioning method from TI that uses a smartphone or tablet to broadcast network credentials to a TI Wi-Fi device. The unprovisioned device can scan for SmartConfig broadcasts while operating in station mode or AP mode.

The CC3220SF device integrates the code needed to run AP and SmartConfig provisioning in the network processor firmware. This saves the user from having to dedicate a large portion of the application memory for implementing provisioning. Instead, the application must only handle the following:

- Sending a command to the network processor to start the provisioning process
- Receiving asynchronous events during provisioning
- Stopping provisioning when complete

To manage the provisioning process within an application, it is important to understand the internal provisioning flow used by the network processor. When provisioning is first started by the application, the CC3220 device enters the configuration state where it waits to receive the parameters of the network. After receiving the parameters, the device then attempts to connect to the network and provides feedback to the user upon success. A diagram of the provisioning process flow is in the CC3120, CC3220 SimpleLink™ Wi-Fi Internet-on-a chip™ Solution Device Provisioning Application Report.

In this example code, the AP provisioning and SmartConfig process is handled by a dedicated provisioning task. The task remains idle until the provisioning state machine is started by the network connection task. When the provisioning state machine is started, the provisioning task puts the network processor in the provisioning state and handles asynchronous events until the system successfully connects to the network. Whenever the application is restarted, the system always attempts to connect to a network based on a stored profile and skip the provisioning process if possible.

Figure 2-8 shows a diagram of the provisioning state machine implemented by the TIDM-1020 software.

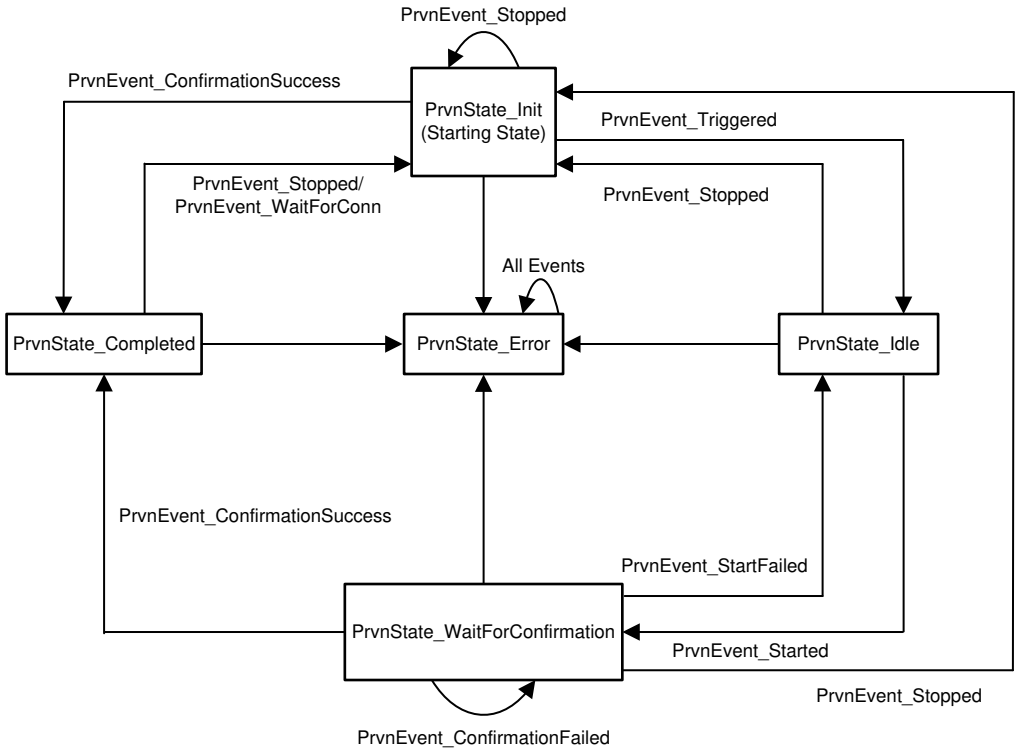


Figure 2-8. Provisioning Task State Machine

2.3.2.2 Wi-Fi Provisioning Over BLE

Wi-Fi provisioning over BLE is another method of provisioning a Wi-Fi based system to a new Wi-Fi network. The main benefit of provisioning a system over BLE is that it can provide a more consistent and streamlined user experience when provisioning from a smartphone or tablet. On some mobile operating systems, AP provisioning requires a user to do the following:

1. Open a settings application to disconnect the mobile device from the Wi-Fi network it is currently using.
2. Connect to the AP created by the device being provisioned.
3. Enter the credentials for the original Wi-Fi network.
4. Switch back to the original Wi-Fi network to confirm that provisioning succeeded.

When provisioning over BLE, the process can be completed without the user ever having to change the Wi-Fi network to which the mobile device is connected. Therefore, the entire provisioning process can be carried out within a single mobile application. The process for provisioning the system over BLE is:

1. BLE advertising starts.
2. A peer connects to a BLE device and authenticates with a passkey.
3. The user writes the SSID, Security Key, and Device Name to corresponding BLE characteristics.
4. The user writes 0x01 to the Provisioning Start characteristic, to signal that a new profile must be tested.
5. The CC3220SF device disconnects the BLE, reads the values written to the characteristics, starts the Wi-Fi network processor, and attempts to connect to the profile.
6. The CC3220SF device waits to obtain an IP address and then disconnects the Wi-Fi and stops the Wi-Fi network processor.
7. The BLE advertising restarts.
8. The mobile device reconnects to the system over BLE and reads the Provisioning Status characteristic (success or fail).
9. If the process succeeds, the CC3220SF device adds the Wi-Fi network information as a stored profile, exits the provisioning state, and then connects to the Wi-Fi network.

Even though the overall process implemented by the system may take more steps, the user experienced is simplified, because the user only needs to take an action for Steps 2, 3, and 4.

The TIDM-1020 software application implements provisioning over BLE using a modified version of the BLE provisioning example from the SimpleLink SDK Bluetooth Plugin. In the TIDM-1020 software, a BLE thread is dedicated to handling communication with the CC2640R2F device, while the system is being provisioned. Because there is no hardware coexistence mechanism for Wi-Fi and BLE in the system, the software application turns off the Wi-Fi network processor while the BLE interface is active and stops the BLE activity while the Wi-Fi interface is active. To keep the Wi-Fi and BLE activity separate, all of the Wi-Fi activity (such as testing received credentials) is handled inside the network interface task instead of the BLE provisioning task.

Figure 2-9 shows the state machine implemented by the task that is dedicated to provisioning over BLE.

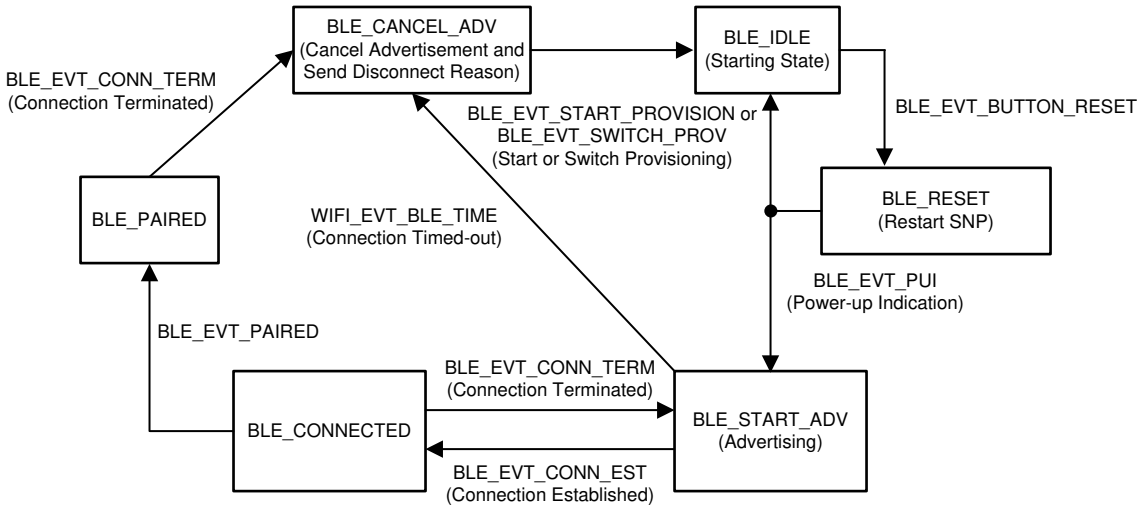


Figure 2-9. BLE Provisioning State Machine

2.3.3 Cloud Connectivity

This thermostat design connects to the IBM Watson IoT platform as a MQTT broker. Sensor data is periodically updated to the cloud, and the cloud application can update the target temperature. The SimpleLink CC3220 device supports a suite of protocols to interface with the cloud.

2.3.4 Sensor Interface

The design demonstrates the use of both the analog- and digital-sensor interface. The digital interface is used to read ambient temperature, pressure, and humidity, measured by a Bosch BME280 sensor mounted on the BOOSTXL-SENSORS BoosterPack.

The air quality is measured through an analog sensor, Winsen MP503, that can be connected to the Grove BoosterPack. The CC3220x ADC runs at a sampling rate of 62.5 KSPS. By default in this design, the ADC is enabled and read every two seconds. The last four readings are averaged to get the actual value for the air-quality measurement. End-equipment developers can add an appropriate signal-conditioning algorithm for the analog sensors.

For motion detection and proximity sensing, a PIR sensor is used. The TIDM-1020 uses a TITAN Micro Electronics TM2291, infrared-sensor signal processor. The reference board can be directly interfaced with the Grove BoosterPack. The detecting angle is up to 120°. The detection toggles the GPIO, which illuminates the screen. Currently, this trigger is used for waking up the display.

2.3.5 Over-the-Air (OTA) Update

The Over-the-Air (OTA) update is the wireless delivery of new software updates or configurations to embedded devices (see Figure 2-10). Using the concept of an identity resolving key (IRK) within the IoT, OTA is an efficient way of distributing firmware updates or upgrades.

The OTA library supports the following cloud content-delivery network (CDN) vendors:

- Dropbox™
- GitHub
- Custom

The OTA library implements a simple HTTP client (TCP) to connect to the CDN server. This client can be configured by the host application as follows:

- Nonsecured (connect to CDN-running HTTP server)
- Secured (connect to CDN-running HTTPS server):
  - Server authentication (required by default)
  - Domain name verifications (required by default)
  - No server authentication

The software-upgrade application and user files must be put in an archive .tar file. By default, all files are nonsecured and fail-safe. The vendor can change the attributes of a file (such as secured, signature, certificate file name, and so forth) by adding an entry to a command file (ota.cmd), which is in JSON format and included in the .tar archive. The implementation details are provided in the CC3x20 SimpleLink™ Wi-Fi® and Internet-of-Things Over the Air Update Application Report.

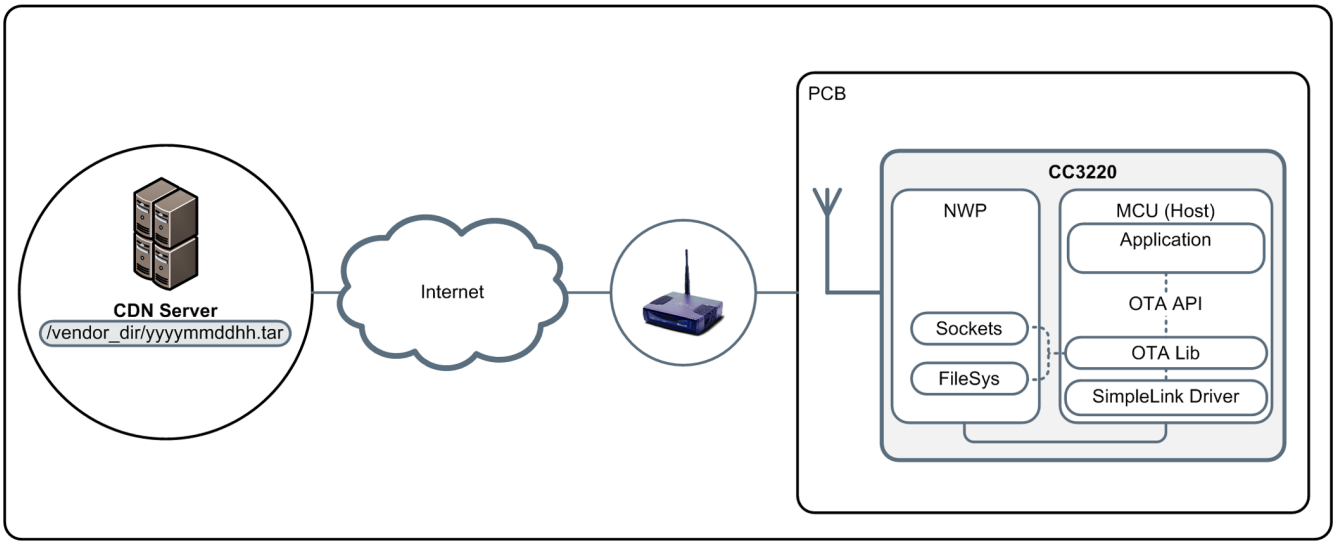


Figure 2-10. OTA System-Block Diagram

2.3.6

2.3.7 Security

Security is an important consideration for all connected IoT devices. The SimpleLink CC3220 device is designed to provide security at several exposure points – Internet-network level security, local-network level security, and protection against physical access and tampering. To provide additional security, the CC3220 device has two separate execution environments for application and network connectivity. The TIDM-1020 demonstrates these security features, as applied to thermostat EE.

For secure cloud connectivity and data transfer from the sensors, the TIDM-1020 uses secure socket implementation (SSL/TLS), with the MQTT implementation. Secure OTA is also implemented as part of the design, to serve as a reference implementation and example.

At the local-network connectivity level, the Wi-Fi Alliance has regulated security and compliance tests as part of its standard. The CC3220 device is a Wi-Fi CERTIFIED device that complies with the security requirements.

The TIDM-1020 demonstrates the secure flash-file system implementation for the following:

- IP protection
- File-system security by means of encryption
- File-system integrity
- Cloning protection

The details of the implementation are provided in Section 3.2.1.

2.3.8 Power Management Options

To remotely control and configure the thermostat, the device must be connected to the Internet. Battery-powered thermostats also demand that power consumption is kept to a minimum. To maintain a continuous connection to the server, the TIDM-1020 uses the always connected mode of operation. This aspect, combined with the proprietary-network learning algorithm from TI, helps to reduce the power consumption by learning the AP sleep interval.

The design also uses a PIR sensor to determine when the display screen must be backlit and sensing for resistive touch input. The overall power requirements and the use-case assumptions are presented in Section 3.3.2.

2.3.9 SimpleLink™ CC3220SF to CC2640R2F Interface

The CC2640R2F device is used in the TIDM-1020 to enable the system to be provisioned to a Wi-Fi network over a Bluetooth low energy connection. In the design, the CC3220SF device acts as the main system controller (simple application processor or SAP) and the CC2640R2F device acts as a BLE network processor (simple network processor or SNP). Using the SAP and SNP solution simplifies the design, because it abstracts away the BLE protocol and allows the BLE functionality to be implemented by sending predefined commands to the CC2640R2F device over the Unified Network Processor Interface (NPI) from TI.

The NPI supports either UART or SPI as the serial communication protocol, but only UART is implemented for this TI Design. In addition to using the standard TX and RX data lines of UART, the NPI also uses three additional signals between the processors, for the host to reset the SNP and to indicate when each device is ready to send or receive data. The three additional signals are called RESET, Master Ready (MRDY), and Slave Ready (SRDY).

For more information on the SAP and SNP solution, see the SimpleAP+SNP wiki page. For more information on the NPI, see the Unified Network Processor Interface wiki page.

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware

The TIDM-1020 uses the available EVM, LaunchPad, and BoosterPack for easy access to the hardware platform. The BoosterPack modules can be stacked on top of each other, to form a complete system.

3.1.1 Hardware

The following sections provide a brief overview of the boards used, as follows.

- SimpleLink™ Wi-Fi® CC3220SF Wireless Microcontroller LaunchPad™ Development Kit
- SimpleLink™ Bluetooth® low energy CC2640R2F wireless MCU LaunchPad™ development kit
- Sensors BoosterPack Plug-In Module (onboard Bosch BME280)
- Kentec QVGA Display BoosterPack
- Grove Base BoosterPack
- Grove - PIR Motion Sensor
- Grove - Air Quality Sensor v1.3
- Grove - SPDT Relay

3.1.1.1 SimpleLink™ Wi-Fi® CC3220SF Wireless MCU LaunchPad™ Development Kit

The SimpleLink Wi-Fi CC3220SF LaunchPad development kit (CC3220SF-LAUNCHXL) highlights the CC3220SF device, a single-chip wireless MCU with 1MB Flash, 256KB of RAM, and enhanced security features. The CC3220SF-LAUNCHXL features onboard emulation and sensors for a full out-of-the-box experience. This board can be directly connected to a PC, for use with development tools such as the Code Composer Studio™ cloud-integrated development environment. Visit dev.ti.com and get started with a SimpleLink Wi-Fi CC3220SF LaunchPad kit today. The CC3220SF device brings IoT-networking security to a new level, empowering developers to easily connect any application to the cloud, with multiple communication protocols. Expedite application development for fast time-to-market with the SimpleLink™ Wi-Fi® CC3220 Software Development Kit (SDK). For additional information, visit the SimpleLink™ Wi-Fi® CC3220SF Wireless Microcontroller LaunchPad™ Development Kit.

The CC3220SF LaunchPad development kit features the following:

- Supports various IDEs: Code Composer Studio™, IAR Embedded Workbench® for Arm® Cortex®-M4
- Stand-alone development platform featuring sensors, LEDs, and push-buttons
- Onboard chip antenna with option for U.FL-based testing
- 2 × 20-pin stackable connectors (BoosterPack headers), to connect to LaunchPad kits and other BoosterPack modules from TI
- Back-channel universal asynchronous receiver/transmitter (UART) through USB to PC
- XDS110-based JTAG emulation with serial port for flash programming

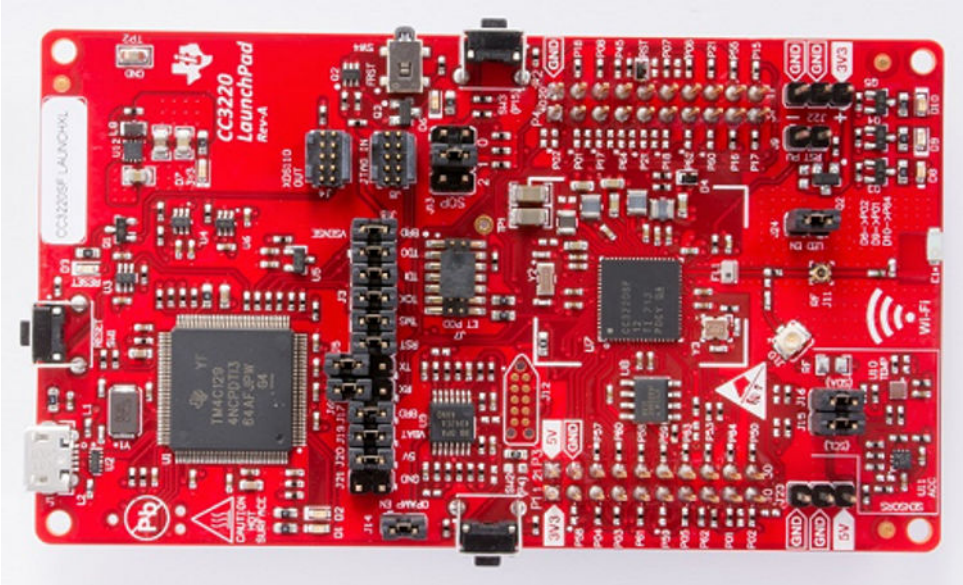


Figure 3-1. SimpleLink™ Wi-Fi® Wireless MCU LaunchPad™ Development Kit

3.1.1.2 SimpleLink™ Bluetooth® Low Energy CC2640R2F Wireless MCU LaunchPad™ Development Kit

The CC2640R2F device has the first fully qualified Bluetooth 5 protocol stack for single-mode Bluetooth low energy applications supporting high speed mode. CC2640R2F is part of the SimpleLink microcontroller (MCU) platform which consists of Wi-Fi, Bluetooth low energy, Sub-1 GHz and host MCUs, which all share a common, easy-to-use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink platform enables you to add any combination of the portfolio's devices into your design, allowing 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/tool/launchxl-cc2640r2.

The CC2640R2 LaunchPad kit brings easy Bluetooth low energy (BLE) connection to the LaunchPad ecosystem with the SimpleLink ultra-low power CC26xx family of devices. Compared to the CC2650 LaunchPad this LaunchPad has:

- More free flash memory for the user application in the CC2640R2F wireless MCU
- Out of the box support for Bluetooth 5 high speed mode and Bluetooth 4.2 specification
- Demo Bluetooth 5 coded physical layers (PHY) for long range mode testing (Bluetooth 5 long range mode stack support will be enabled with future software updates)

The CC2640R2F wireless MCU contains a 32-bit Arm® Cortex®-M3 processor that runs at 48 MHz as the main microcontroller and a rich peripheral feature set that includes a unique ultra-low power sensor controller. This sensor controller was created for interfacing external sensors and for collecting analog and digital data autonomously while the rest of the system is in sleep mode.

The CC2640R2 LaunchPad kit is optimized for BLE applications with more flash memory available. It supports programming and debugging from Code Composer Studio™ and IAR Embedded Workbench® integrated development environments (IDEs).

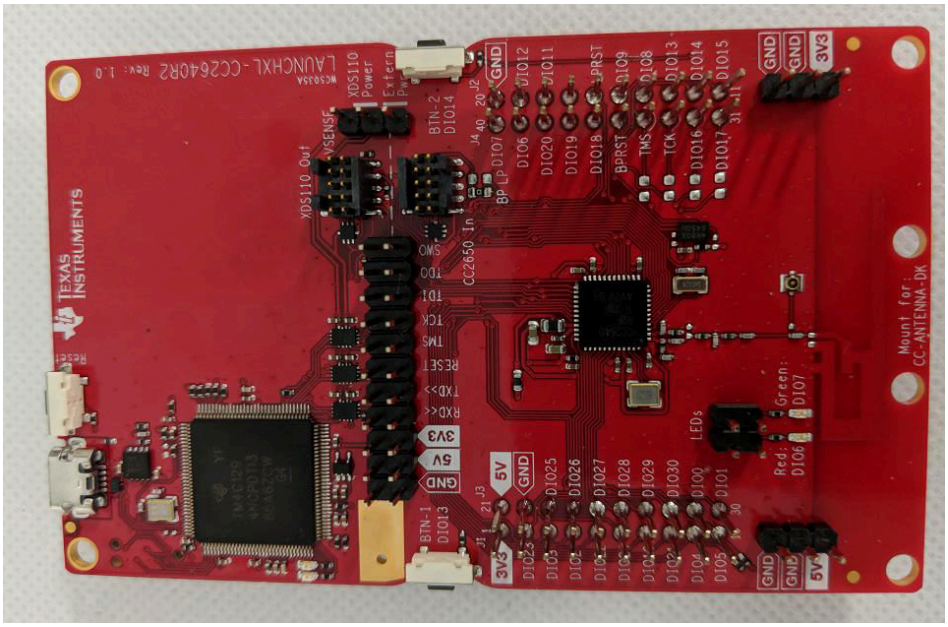


Figure 3-2. CC2640R2F LaunchPad™ Configuration

All jumpers must be removed from the CC2640R2F LaunchPad before connecting it to the CC3220SF LaunchPad to run the demo. However, the steps for programming the LAUNCHXL-CC2640R2 described in Section 3.2 must be completed before removing all jumpers and assembling the hardware.

The default pin mapping of the SNP images for the CC2640R2F, provided in the BLE plugin, lead to pin conflicts with the CC3220SF LaunchPad and Kentec QVGA Display BoosterPack. To allow all EVMs to be connected together with minimal hardware modifications, you must rebuild the SNP application (simple_np) from the SimpleLink CC2640R2F SDK, with an updated pin mapping before programming the board. The process for

updating the pin mapping of the CC2640R2F is described in Section 3.1.2, but it results in the following connections (Table 3-1) between the CC3220SF and the CC2640R2F devices.

Table 3-1. CC3220SF and CC2640R2F Device Pin Mapping

LaunchPad Header Pin	CC3220SF Device Pin	CC2640R2F Device Pin
P1.4	Pin 3 (UART0 TX)	DIO2 (UART RX)
P1.3	Pin 4 (UART0 RX)	DIO3 (UART TX)
P2.14	Pin 6 (MRDY)	DIO8 (MRDY)
P2.18	Pin 8 (SRDY)	DIO11 (SRDY)
P4.35	Pin 18 (RESET)	BPRST (RESET)

3.1.1.3 Sensors BoosterPack™ – BOOSTXL-SENSORS

The BOOSTXL-SENSORS BoosterPack plug-in module provides an easy way to add digital sensors to a LaunchPad development kit design (see Figure 3-3). MCU LaunchPad developers can use this BoosterPack module to start developing sensor applications using the onboard gyroscope, accelerometer, magnetometer, pressure, ambient temperature, humidity, ambient light, and infrared-temperature sensors. The BOOSTXL-SENSORS BoosterPack features the following:

- TI OPT3001 ambient-light sensor
- Bosch BMI160 Inertial Measurement Unit (IMU) sensor – accelerometer and gyroscope
- Bosch BMM150 Magnetometer
- Bosch BME280 Environmental sensor – pressure, ambient temperature, and humidity
- Compatible with TI LaunchPad kits

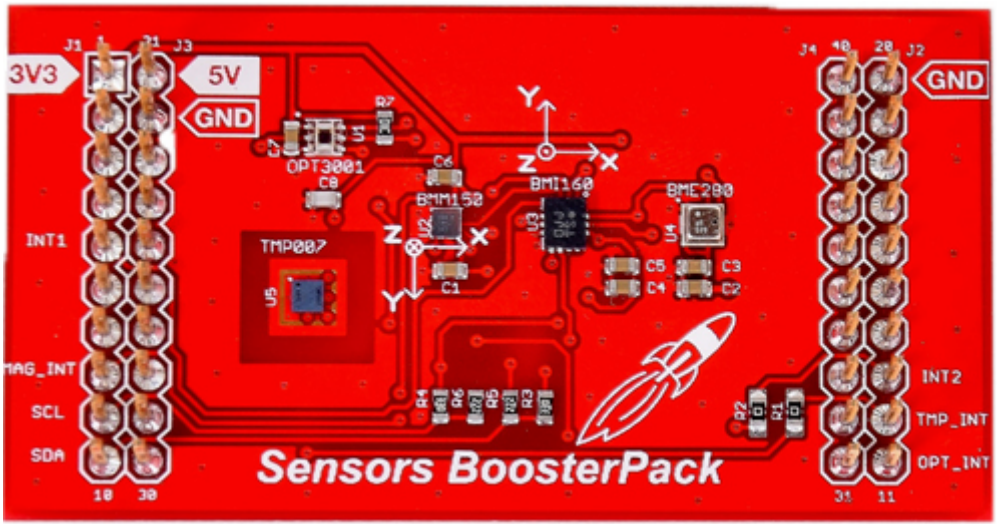


Figure 3-3. BOOSTXL-SENSORS Sensors BoosterPack™ Plug-In Module

3.1.1.4 Kentec QVGA Display BoosterPack™ - BOOSTXL-K350QVG-S1

The BOOSTXL-K350QVG-S1 Kentec QVGA display BoosterPack is an easy-to-use plug-in module for adding a touchscreen color display to a LaunchPad design (see Figure 3-4). MCU LaunchPad developers can use this BoosterPack to start developing applications using the 320 × 240-pixel, SPI-controlled, TFT QVGA display, with the resistive touchscreen. The Kentec QVGA display BoosterPack features the following:

- Kentec TFT LCD (P/N: K350QVG-V2-F)
  - 3.5 inch QVGA (320 × 240 resolution)
  - SPI
  - 4-wire resistive touchscreen
  - White LED backlight
- LED backlight driver circuit
- Complies with the BoosterPack standard for use with 20-pin and 40-pin LaunchPad kits

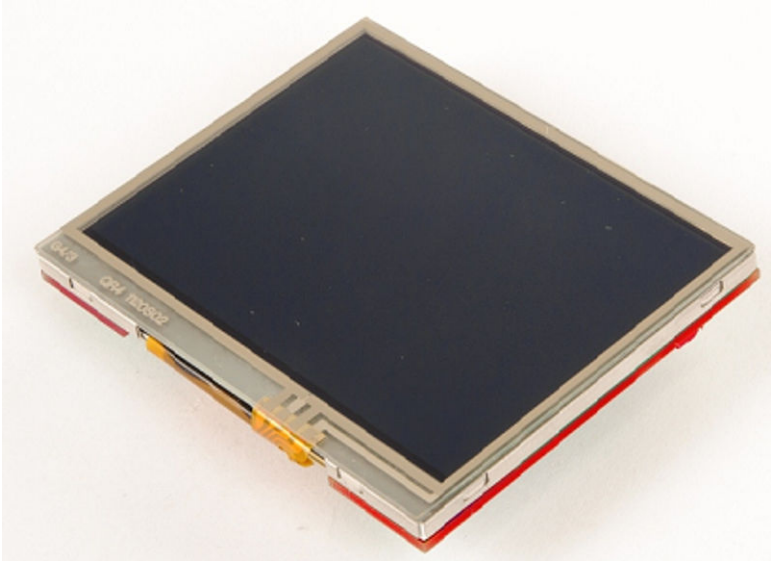


Figure 3-4. Kentec QVGA Display BoosterPack™

3.1.1.5 Grove Base BoosterPack™ for TI LaunchPad™

The Grove Base BoosterPack serves as an interface between the TI LaunchPad and the Seeed Studio Grove Family (see Figure 3-5). Users have the flexibility to build their demos among a range of sensors, actuators, displays, lights, motors, and so forth. The Grove Base BoosterPack is an expansion module that plugs in on top of any TI LaunchPad, and enables the LaunchPad to interact with the expansive Grove family of modules. This setup enables developers to create a complete system of inputs and outputs by simply plugging in multiple Grove sensors and actuators to the LaunchPad.

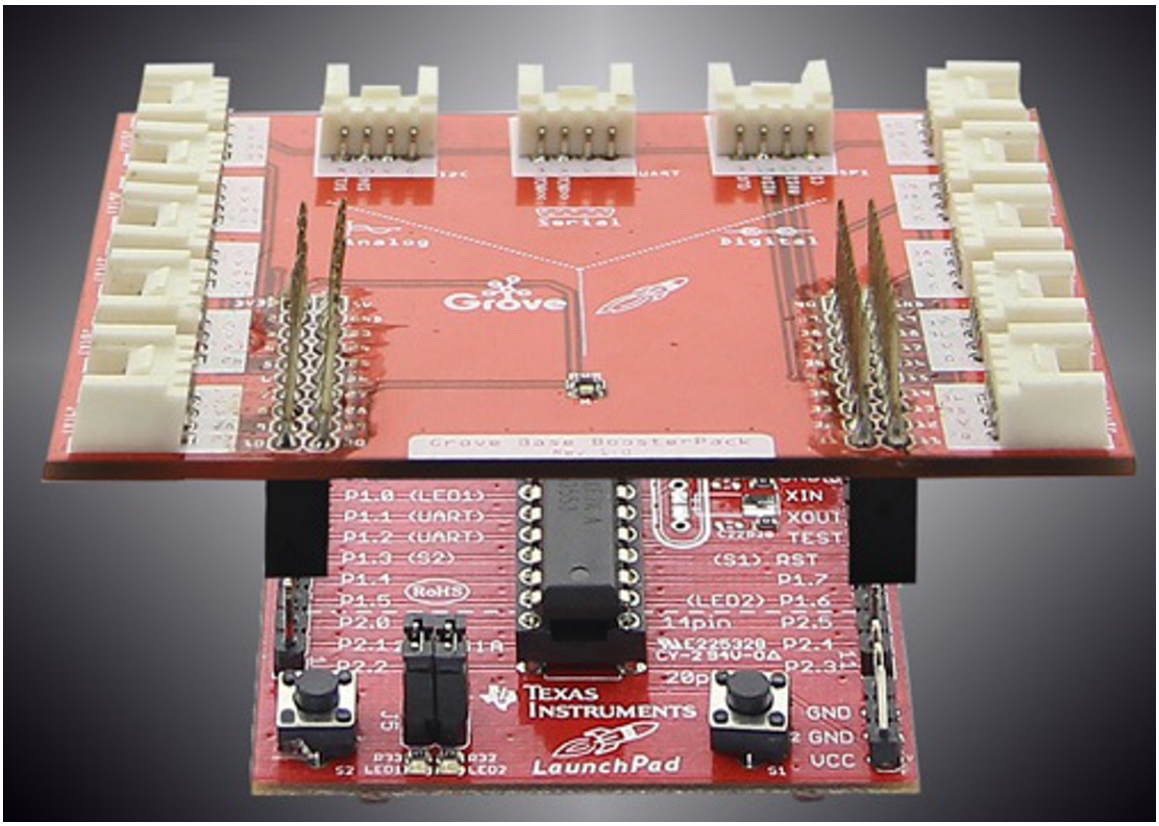


Figure 3-5. Grove Base BoosterPack™ for TI LaunchPad™

3.1.1.6 Grove – PIR Motion Sensor

This component is a simple-to-use PIR motion sensor with a Grove-compatible interface. Connect the sensor to a Grove BoosterPack (see Figure 3-6). When someone moves in its detecting range, the sensor outputs HIGH on its SIG pin. The detecting range and response speed can be adjusted by 2 potentiometers soldered on its circuit board; the response speed is from 0.3 s to 25 s, with a maximum 6 meters of detecting range. The PIR sensor features the following:

- Grove-compatible interface
- Voltage range: 3 V to 5 V
- 2.0 cm × 4.0 cm twig module
- Detecting angle: 120°
- Detecting distance: maximum 6 m (3 m by default)
- Adjustable detecting distance and holding time

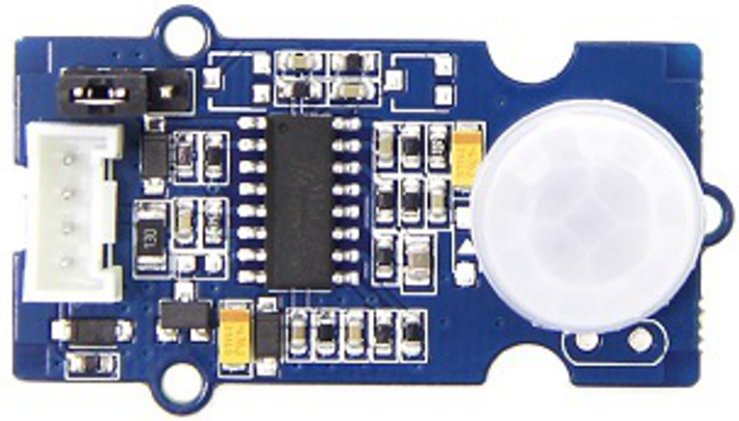


Figure 3-6. Grove – PIR Motion Sensor

3.1.1.7 Grove – Air Quality Sensor (v1.3)

This sensor is designed for indoor air-quality testing (see Figure 3-7). The main gases detected are carbon monoxide, alcohol, acetone, thinner, formaldehyde, and other slightly toxic gases. The sensor is compatible with a 5-V or 3.3-V power supply. The air quality sensor features the following:

- Low-power consumption
- High sensitivity
- Small outline
- Responsive to wide scope of target gases
- Cost efficient
- Durable
- Compatible with 5 V and 3.3 V

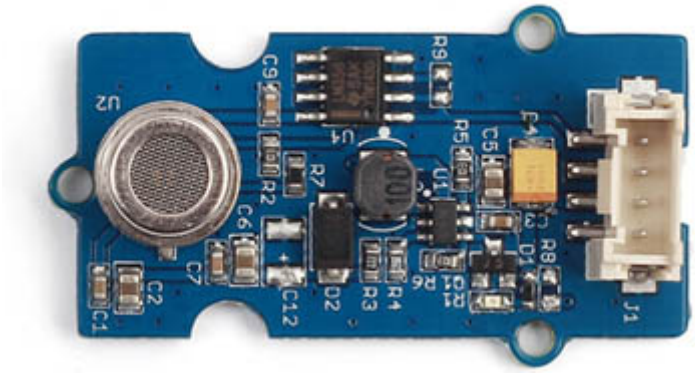


Figure 3-7. Grove – Air Quality Sensor v1.3

3.1.1.8 Grove – Relay (v1.2)

The Grove – Relay is a digital, normally open switch that controls a relay capable of switching higher voltages and currents than the LaunchPad can normally handle (see Figure 3-8). When set to HIGH, the LED lights and the relay closes, allowing current to flow. The peak voltage capability is 250 V at 10 A. In this reference design, the v1.2 board is used, because it supports control voltages from 3.3 VDC to 5 VDC.

Exercise caution when working with main voltages – if in doubt, contact a professional, such as a licensed electrician for help.

Figure 3-8. Grove – Relay

3.1.2 Assembling Reference Design Hardware Stack

The hardware stack of the access panel is assembled by attaching the BoosterPack modules to the SimpleLink CC3220SF LaunchPad. The CC3220SF LaunchPad can attach to the BoosterPack modules using either the headers on the top of the board or the header sockets on the bottom of the board.

TI recommends stacking the boards in this order, board 1 is at the top.

1. BOOSTXL-K350QVG-S1 BoosterPack – for display and touch
2. Grove BoosterPack modules – connected to relay, PIR, and air quality sensors:
   a. GROVE-RELAY
   b. GROVE-PIR Motion Sensor
      - Check that the jumper on the PIR is set to non-retrigger (N_Retrig) mode. (See the diagram on the bottom of the sensor PCB.)
   c. GROVE-Air Quality Sensor v1.3
3. LAUNCHXL-CC2640R2 - SimpleLink Bluetooth low energy CC2640R2F Wireless Microcontroller LaunchPad Development Kit
4. BOOSTXL-SENSORS
5. CC3220SF-LAUNCHXL – SimpleLink Wi-Fi CC3220SF Wireless Microcontroller LaunchPad Development Kit

Due to limitations of pins accessible from the LaunchPad headers, this demo uses 2-wire, JTAG SWD mode for debugging and changes the default UART RX and TX pins. This process requires moving a number of headers, as follows:

- SOP header position (J13)
  1. Flashing: 100
  2. Debugging in SWD mode: 001
  3. Running flashed binary: 001 or 000
- UART RX and TX header positions (J6 and J5)
  1. Flashing: UART RX and TX in top position
  2. Other: UART RX (J6) in bottom position, UART TX (J5) in top position

To see the UART terminal output, connect a jumper wire from the top pin of J6 (XDS110 UART RX) to the header J2 pin 45.

For more information on the LaunchPad headers, see the CC3220 SimpleLink™ Wi-Fi® LaunchPad™ Development Kit Hardware User's Guide.

3.1.3 TIDM-1020 Pin Configuration

This reference design uses a LaunchPad kit and multiple BoosterPack modules with interlocking pin configurations, based on the ports available for use through the footprint of the LaunchPad kit. Because BoosterPack modules may have conflicting pin configurations, Table 3-2 lists the factory pin configuration across the entire system.

Figure 3-9 shows the pin mapping.

Table 3-2. Pin MUX Configuration for TIDM-1020 (1)

LP Header Pin	Demo Function	CC3220SF LP	BOOSTXL-SENSORS	BOOSTXL-K350QVG-S1 (With ECO)	Grove Base BoosterPack	LAUNCHXL-CC2640R2
1	3.3-V supply	3.3-V supply	3.3-V supply	3.3-V supply	3.3-V supply	3.3-V supply
2	—	ANALOG_IN	NC	NC	NC	Not used
3	UART0_RX (PIN 4)	UART_RX	NC	NC		UART_TX (DIO 3)
4	UART0_TX (PIN 3)	UART_TX	NC	NC		UART_RX (DIO 2)
5	—	GPIO	INT1	NC	NC	Not used
6	—	ADC_CH2	NC	NC	NC	Not used
7	LCD_SCL (PIN 5)	SPI_CLK	NC	LCD_SCL	SPI_CLK	Not used
8	LCD_SDC (PIN 62)	GPIO	MAG_INT	LCD_SDC	NC	Not used
9	I2C_SCL (PIN 1)	I2C_SCL	I2C_SCL	NC		Not used
10	I2C_SDA (PIN 2)	I2C_SDA	I2C_SDA	NC		Not used
11	LCD TOUCH YN (PIN 15)	GPIO	OPT_INT	TOUCH_YN	NC	Not used
12	UART1_TX DBG (PIN 55)	MISO	TMP_INT	NC	SPI_CS	Not used
13	—	MOSI	INT2		NC	Not used
14	BLE MRDY (PIN 6)	MISO	NC	NC	SPI_MISO	MRDY (DIO 8)
15	LCD_MOSI (PIN 7)	MOSI	NC	LCD_SDI	SPI_MISI	Not used
16	CC3220_RESET	RESET	NC	NC	NC	Not used (LPRST)
17	UART1_RX DBG (PIN 45)	GPIO	NC	NC	NC	Not used
18	BLE SRDY (PIN 8)	SPI_CS	NC	NC	NC	SRDY (DIO 11)
19	—	GPIO	NC	NC	NC	Not used
20	GND	GND	GND	GND	GND	GND
21	5-V supply	5-V supply	NC	5-V supply	5-V supply	5-V supply
22	GND	GND	GND	GND	GND	GND
23	LCD TOUCH YP (PIN 57)	ANALOG_IN	NC	TOUCH_YP	ANALOG CAPABALE PIN	Not used
24	LCD TOUCH XP (PIN 60)	ANALOG_IN	NC	TOUCH_XP	ANALOG CAPABALE PIN	Not used
25	AIR_QUALITY (PIN 58)	ANALOG_IN	NC	NC	ANALOG CAPABALE PIN	Not used
26	PIR (PIN 59)	ANALOG_IN	NC	NC	ANALOG CAPABALE PIN	Not used
27	LCD_SCS (PIN 63)	I2S_WS	NC	LCD_SCS	ANALOG CAPABALE PIN	Not used
28	LCD_TOUCH_XN (PIN 53)	I2S_SCLK	NC	TOUCH_XN	ANALOG CAPABALE PIN	Not used
29	—	I2S_SDout	NC	NC	NC	Not used
30	—	I2S_SDin	NC	NC	NC	Not used
31	—	GPIO	NC		NC	Not used
32	LCD_RESET (PIN 16)	GPIO	NC	LCD_RESET	NC	Not used
33	—	GPIO	NC	NC	NC	Not used
34	—	GPIO	NC	NC	NC	Not used
35	BLE BPRST	CCAP/GPIO	NC	NC	DIGITAL/PWM pin	BPRST
36	RELAY (PIN 21)	CCAP	NC	NC	DIGITAL/PWM pin	Not used
37	LCD_PWM (PIN 64)	PWM	NC	PWM	DIGITAL/PWM pin	Not used

Table 3-2. Pin MUX Configuration for TIDM-1020 (1) (continued)

LP Header Pin	Demo Function	CC3220SF LP	BOOSTXL-SENSORS	BOOSTXL-K350QVG-S1 (With ECO)	Grove Base BoosterPack	LAUNCHXL-CC2640R2
38	—	PWM	NC	NC	DIGITAL/PWM pin	Not used
39	—	PWM	NC	NC	DIGITAL/PWM pin	Not used
40	—	PWM	NC	NC	DIGITAL/PWM pin	Not used

(1) Cells shaded in:

- green are inputs to a LaunchPad or a BoosterPack
- yellow are outputs from a LaunchPad or a BoosterPack
- blue are left as floating pins
- red are not connected (NC) to the LaunchPad footprint
- white (not shaded) are open-drain or are unused in this TI Design

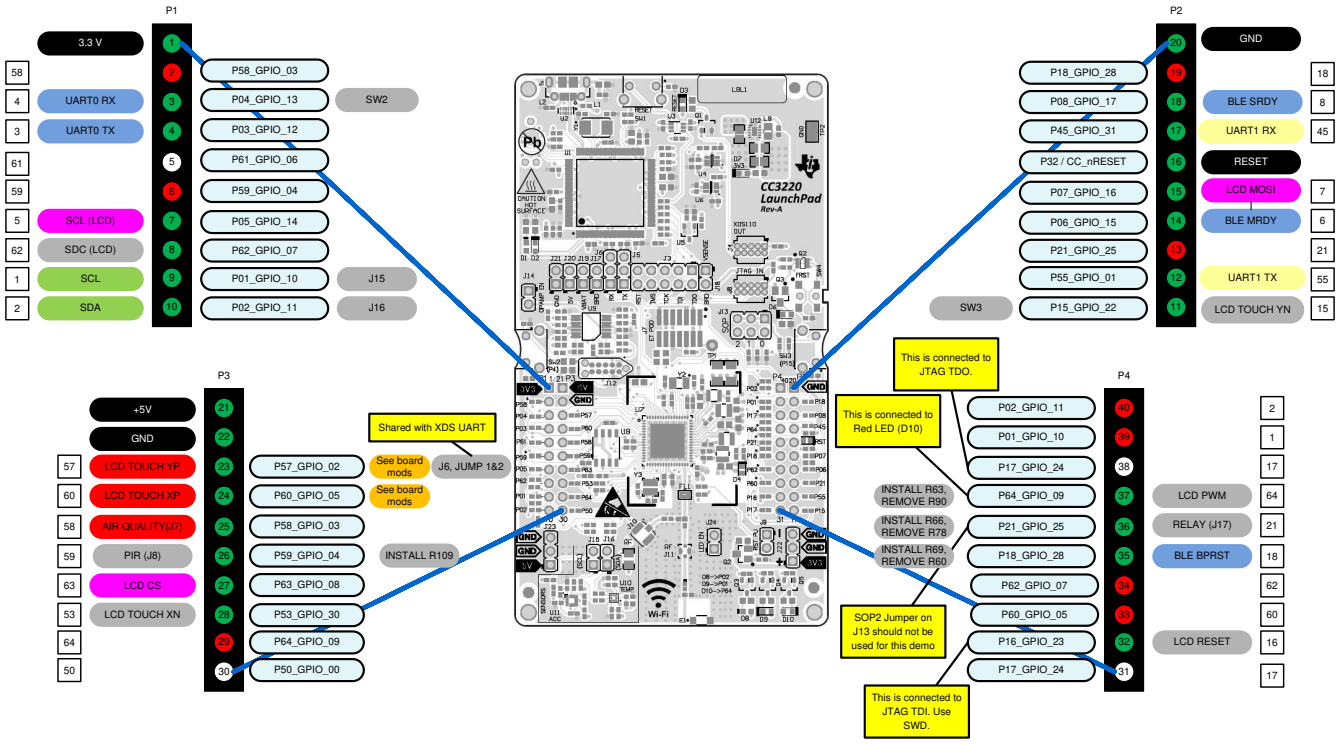


Figure 3-9. TIDM-1020 BLE Provisioning Pin Map

3.1.4 Modifications Required for LaunchPad™ and BoosterPack™

The following sections contain the list of modifications done for the LaunchPad and BoosterPack. These sections also details the need for the listed change. The changes recommended here are applicable for only the TIDM-1020.

3.1.4.1 CC3220SF SimpleLink™ Wi-Fi® LaunchPad™ Modifications for ECOs

The changes are provided as follows (see Figure 3-10):

- Remove R102, R103, R104, R105, R60, R78, R90 (coded red in the diagram)
- Add 0-Ω resistor – R61, R63, R66, R69, R109 (coded green in the diagram)
- Replace with 1K resistor – R100, R101 (coded blue in the diagram)
- Replace with 1.2K resistor – C53, C54 (coded yellow in the diagram)

Figure 3-10 shows the positions of these components, with color coding.

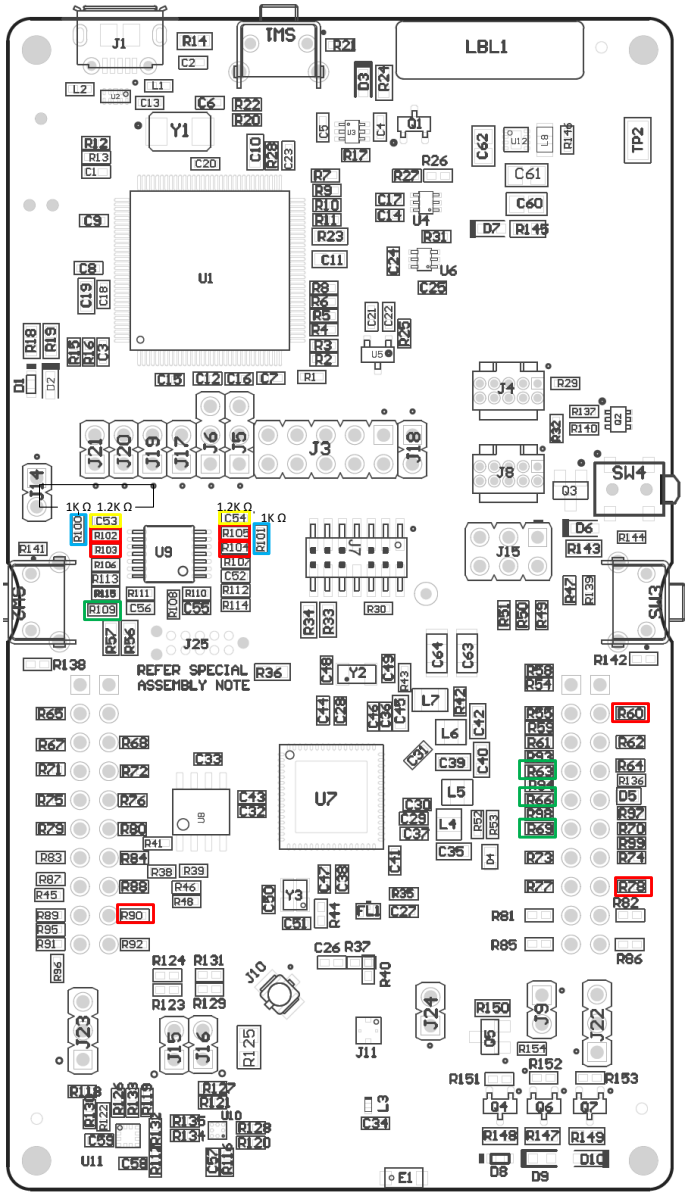


Figure 3-10. Modifications for CC3220SF LaunchPad™

3.1.4.1.1 LaunchPad™ Modification Changes for Resistive Touch

The resistor and capacitor replacement (R102, R103, R104, R105, C53, C54, R100, and R101) are done to create a voltage-divisor network and bypass the buffer. This setup allows the ADC pins on the CC3220SF device to be reconfigured as either output GPIO or input ADC. This configuration is required for reading the resistive-touch position output. The resistor-divisor network also divides the resistive-touch display-input voltage, to match the ADC rating on the CC3220SF device. The CC3220 ADC input can vary from 0 V to 1.4 V, and the output of the resistive-touch display can go from 0 V to 3.3 V.

Figure 3-11 shows the original circuit for this part and the board.

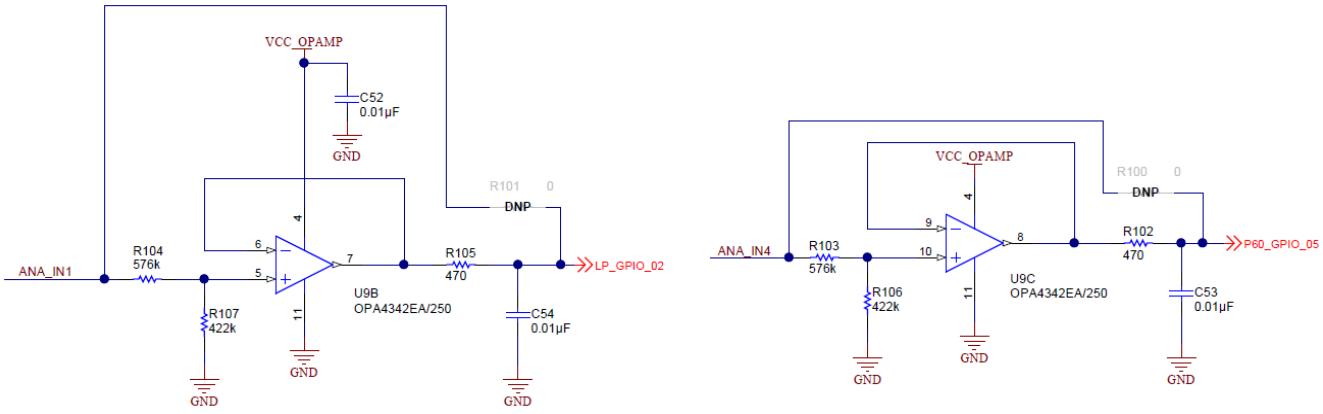


Figure 3-11. Original Circuit for ADC Buffer

After the modifications, Figure 3-12 shows the resultant circuit.

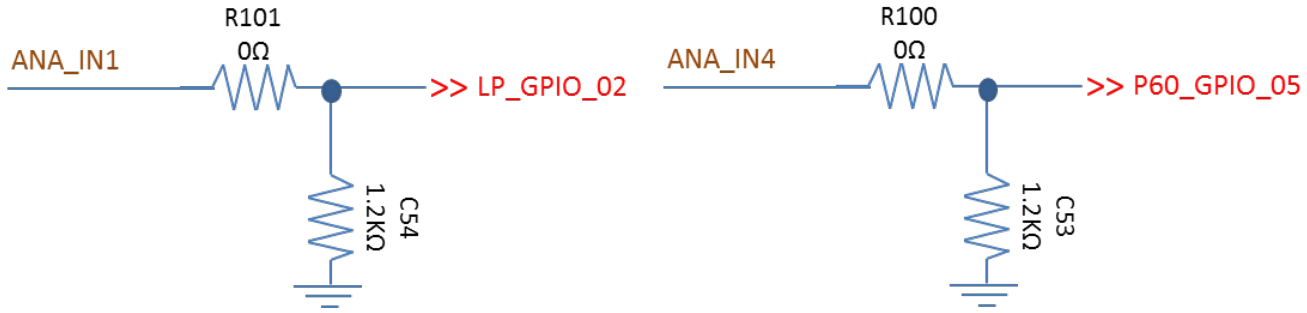


Figure 3-12. Modified Circuit After Modifications

R61, R63, R66, and R69 are 0-Ω resistors for additional GPIOs required for the design.

3.1.4.2 Kentec QVGA Display BoosterPack™ - BOOSTXL-K350QVG-S1 Modifications

Make the following changes for the Kentec QVGA display BoosterPack.

1. Remove R6, R11, and R17.
2. Move the blue wire from header J3 pin 27 to pin 15 of CN1 (the right pad, where R6 connected previously) to change the LCD chip select (SCS).
3. Move the blue wire from header J3 pin 28 to header J4 pin 31 (the left pad, where R17 connected previously, or the right pad of C6) to change the TOUCH XN.
4. Move the blue wire from header J4 pin 37 to pin 4 of U1 (the top pad, where R11 connected previously) to change the LED PWM.

Figure 3-13 shows the changes in the modified schematic.

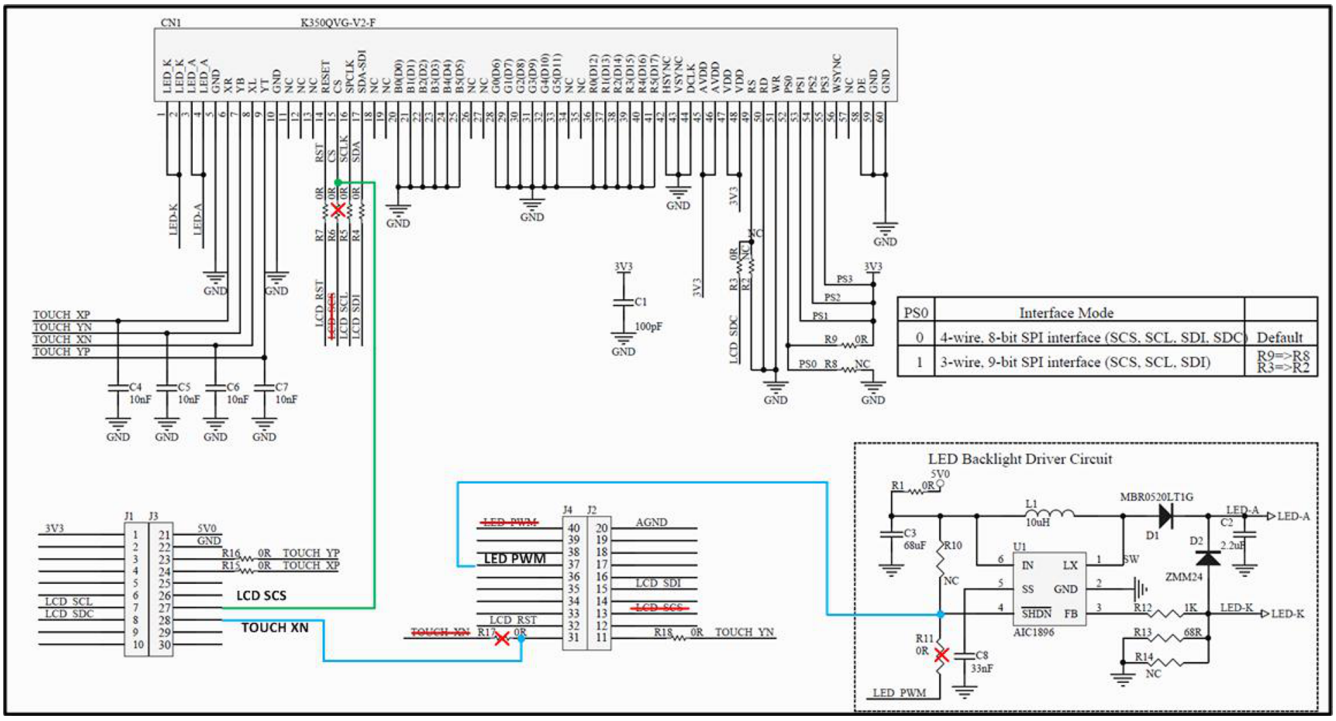


Figure 3-13. Modifications Needed for Kentec QVGA Display BoosterPack™

3.2 Getting Started Firmware

3.2.1 Required Software

- TIDM-1020 Software
- Code Composer Studio (CCS) Integrated Development Environment (IDE) (Arm® Compiler TI v18.1.1.LTS)
- SimpleLink™ Wi-Fi® CC3220 Software Development Kit (SDK) v2.10.00.04
- Bluetooth Plugin for SimpleLink™ MCU SDK v1.40.00.42 (required for BLE provisioning)
- Sensor and Actuator Plug-ins for SimpleLink™ MCU SDKs v1.20.00.02
- SimpleLink™ CC2640R2 SDK - Bluetooth® low energy (required for BLE provisioning)
- CCS UniFlash 4.3.1.1835 or newer
- SimpleLink SDK Explorer, mobile application for Apple iOS or Android devices (required for BLE provisioning)
- **or** SimpleLink™ Wi-Fi® Starter Pro, mobile application for Apple iOS or Android devices

To import the software project into CCS, the required SDK and Plugin packages must be installed.

3.2.2 Opening and Configuring IBM® Cloud™ Account

3.2.2.1 Adding Service

1. Create an IBM Cloud® account, and follow the instructions to confirm the account. The limited free-trial account can be used.
2. When logged in, click the Catalog tab on the top-right corner, as shown in Figure 3-14.

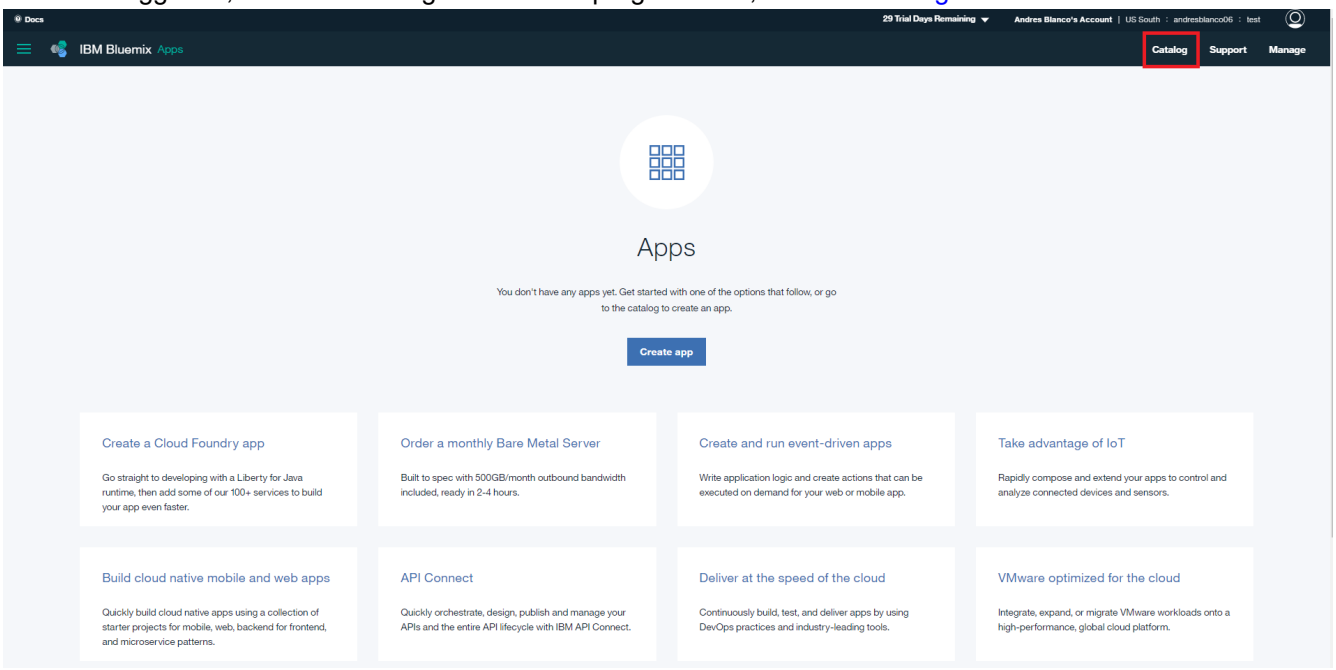


Figure 3-14. Catalog Tab

3. Click on Internet of Things under Services on the left menu, as shown in Figure 3-15.

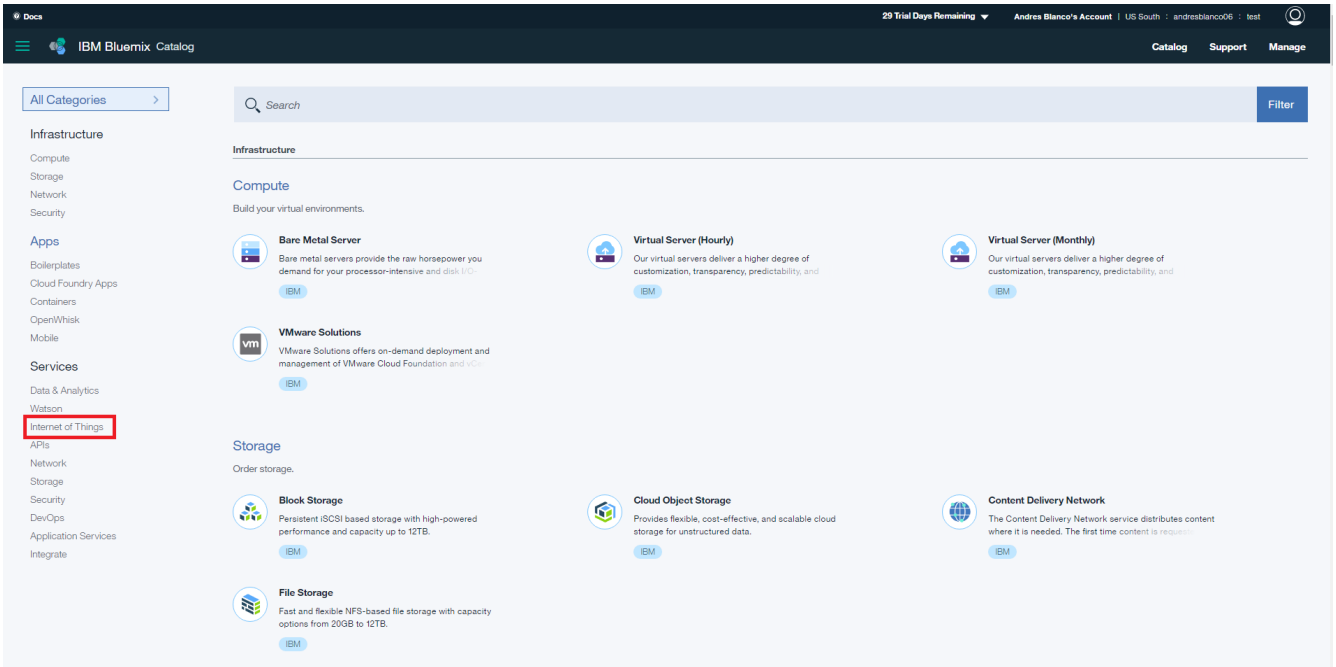


Figure 3-15. Choose Internet of Things

4. Choose Internet of Things Platform from the options provided, as shown in Figure 3-16.

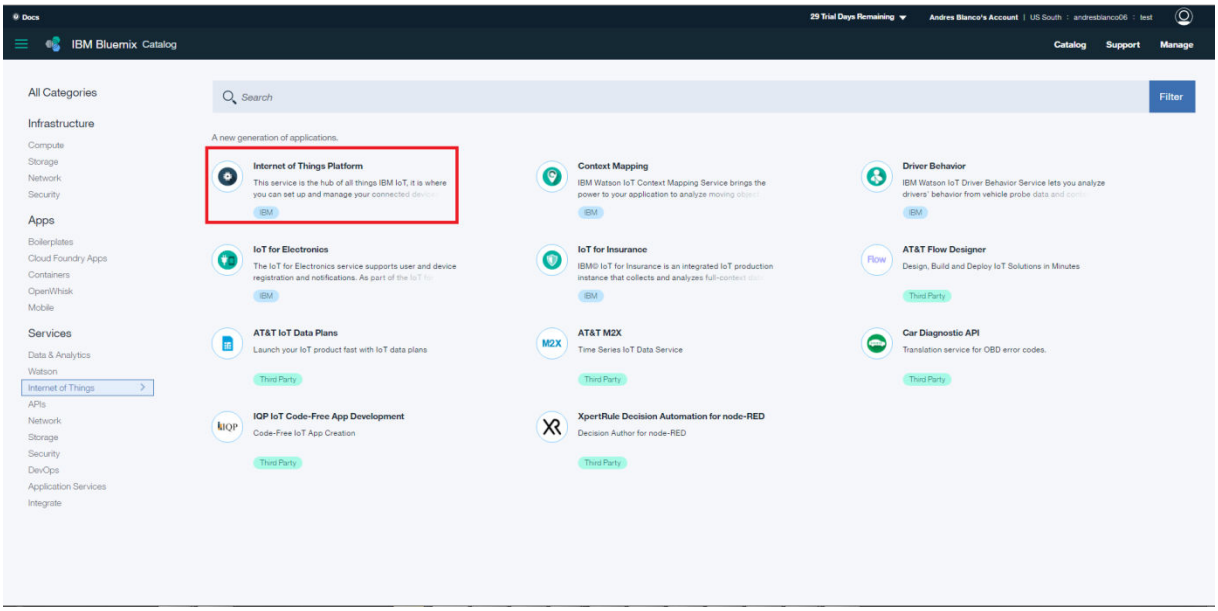


Figure 3-16. Internet of Things Platform

5. Give the new platform a service name, and click the Create button on the bottom-right corner, as shown in Figure 3-17.

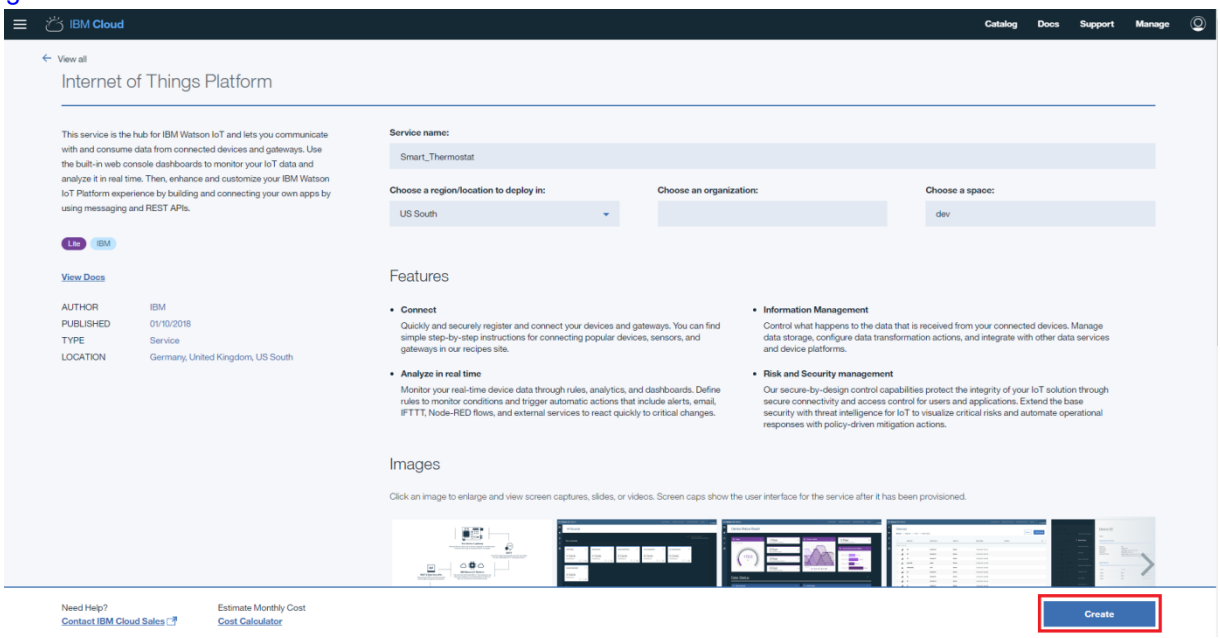


Figure 3-17. Click Create Button

6. Click the Launch button, as shown in Figure 3-18.

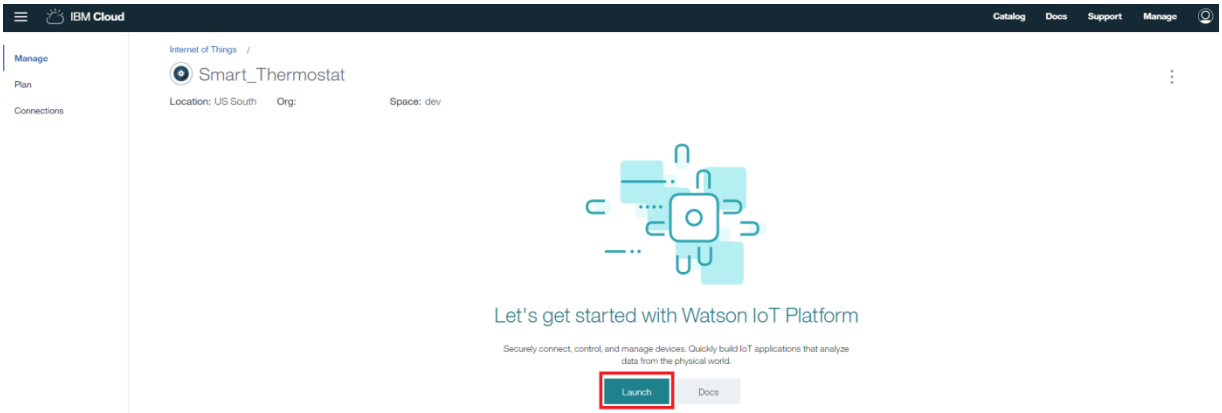


Figure 3-18. Launch Watson IoT Platform

3.2.2.2 Adding Device

1. Select Devices from the navigation bar on the left, as shown in Figure 3-19.

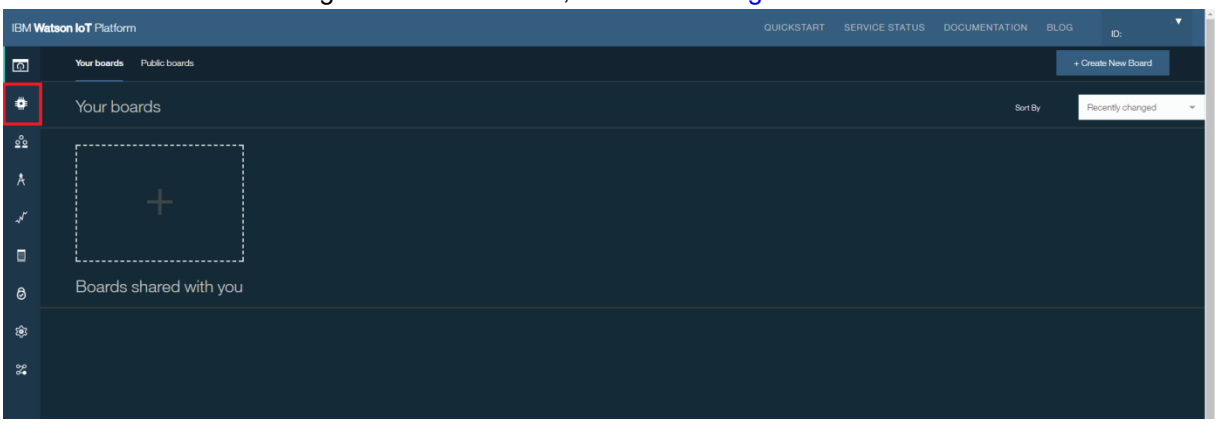


Figure 3-19. Select Devices

2. Select Device Types from the top menu, and click Add Device Type on the right, as shown in Figure 3-20.

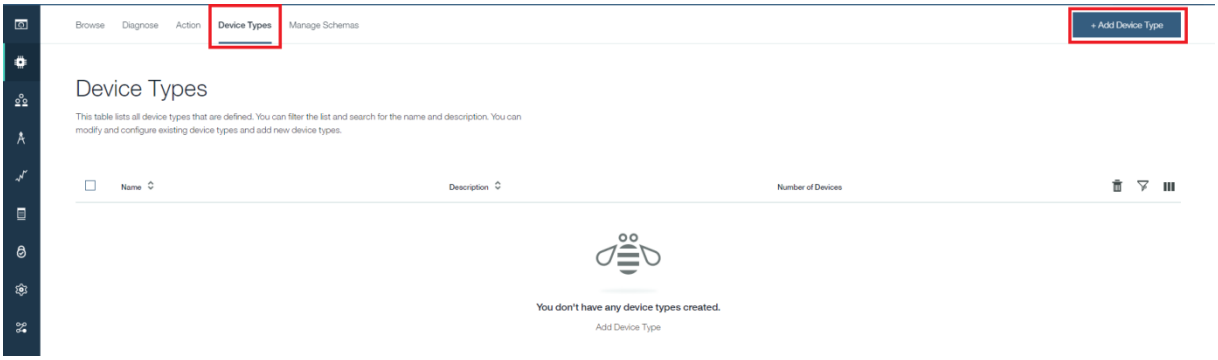


Figure 3-20. Add Device Type

3. Select the Gateway type and enter CC3220 as the device type name, as shown in Figure 3-21. Then, click Next and then click Done.

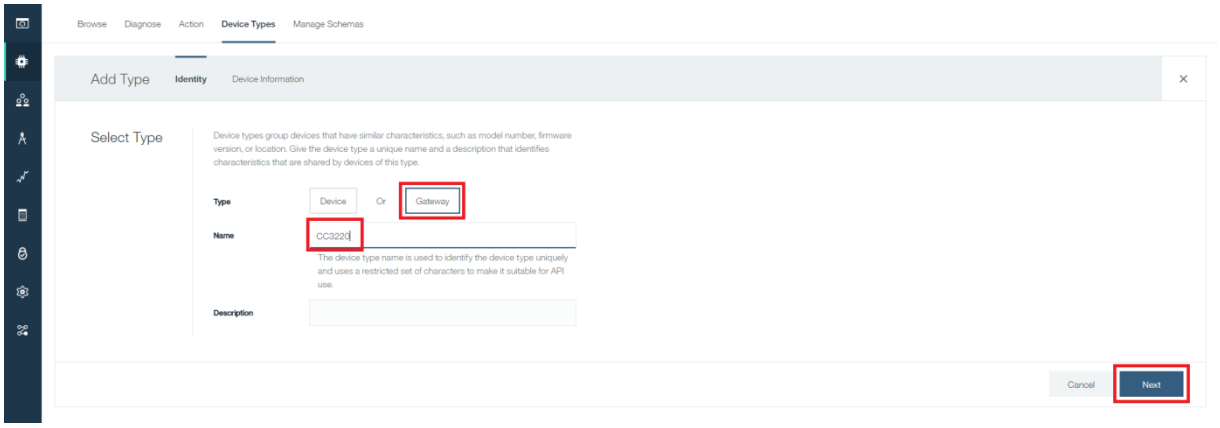


Figure 3-21. Select Gateway Type

4. Return to the device creation page by selecting Browse in the top menu. Click Add Device, as shown in Figure 3-22.

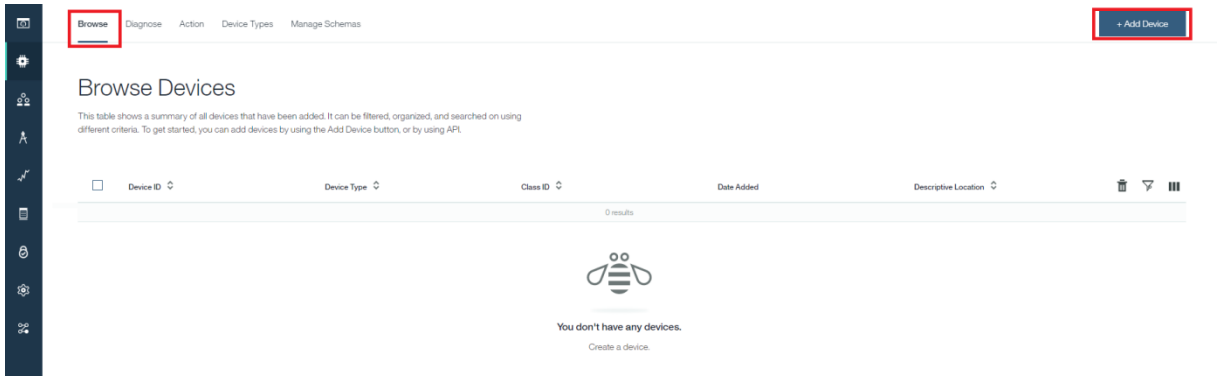


Figure 3-22. Add Device

5. Select the device type, enter Thermostat as the Device ID, and press the Next button, as shown in Figure 3-23. Ignore the device information and press the Next button again.

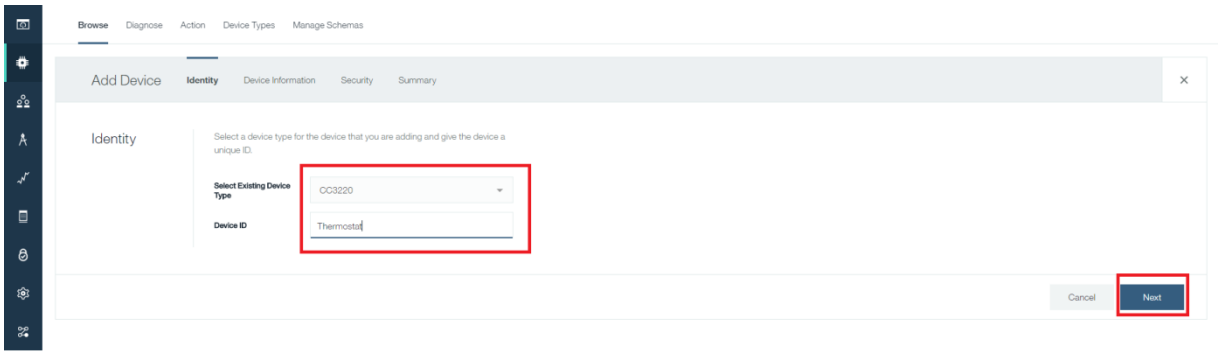


Figure 3-23. Enter Device ID

6. On the Security tab, create a token. Make note of this token field, because it is used for authenticating the device to the cloud. Click Next, and then Done.
7. A summary of the device credentials and information displays. Take a screenshot of this page and save it, because this is the last time the Authentication Token is visible.

3.2.2.3 Generating API Key for Cloud Application

1. Navigate to the Apps tab from the navigation bar on the left, and select Generate API Key.
2. Record the API Key and Authentication Token before continuing.
3. Verify the Standard Application API Role is selected, then click the Generate button.

3.2.3 Setting Up Cloud Application

1. Navigate to the TI GUI Composer Cloud Tool at dev.ti.com/gc, you will need a TI.com login.
2. Select Import Existing Project, and browse for the *WiFi_Thermostat_GUI.zip* included with the design software. Do not unzip this file before importing.
3. Navigate to Project → Properties, and click Next to get to the Target communication setting page (see Figure 3-24).
4. In the my_ibm_iot model, add your IBM Cloud orgID, API key, and token generated earlier. The orgID is in the six letters after *a-* in the API key.
5. Add the Thermostat device ID created earlier with the Watson IoT Platform.

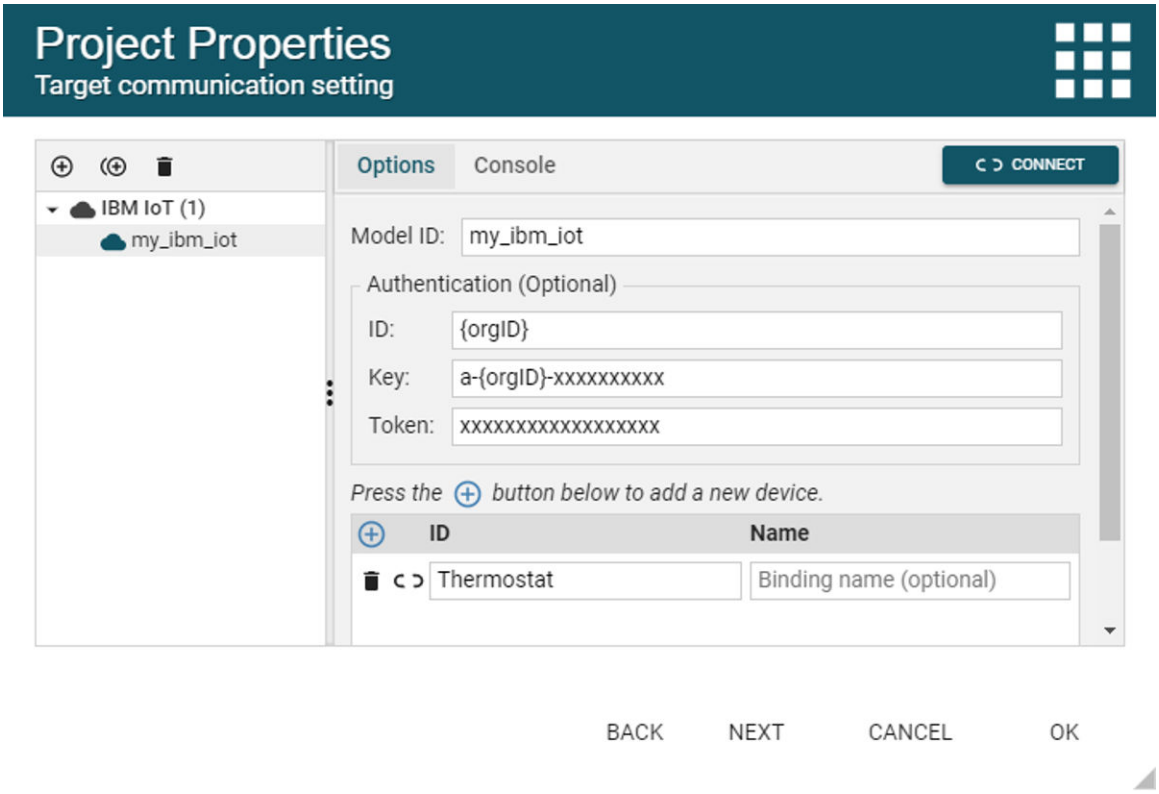


Figure 3-24. GUI Composer Project Properties

6. Press OK to return to the application screen.
7. Select Save in the top menu. Now users can press the Play button in the menu to run the application, publish the application to their TI Cloud Tool Gallery (accessible by URL in a browser), or export it as a stand-alone app.

3.2.4 Configuring Wi-Fi® Thermostat Project

1. Open CCS and select Project → Import CCS Projects.
2. Browse to the *wifi_thermostat* folder in the TI design software and select the project (if this returns an error, ensure the SAIL plugin is installed).
3. In *wifi_thermostat_app.h*, change *DEMO_CITY* and *DEMO_TIME_ZONE* to the desired city and time zone for weather forecast and clock display. The city must be a string containing "City, Country" (do not include a U.S. state in this string). For the full list of supported time-zone definitions in the CC3220 SDK utilities library, see *source/ti/net/utils/clock_sync.h*.
4. In *mqtt_client_task.c*, update *SERVER_ADDRESS* and *ClientID* with your orgID:

```
/* Defining Broker IP address and port Number                                    */#define
SERVER_ADDRESS          "{orgID}.messaging.internetofthings.ibmcloud.com"//#define
SERVER_IP_ADDRESS       ""#define PORT_NUMBER             1883
#define SECURED_PORT_NUMBER     8883
#define LOOPBACK_PORT           1882
char        "ClientId = "g:{orgID}:CC3220:Thermostat";const char  "Username = "use-token-
auth";const char  "Password = "";
```

5. Set the Password chosen when adding the device on the IBM Watson IoT Platform.
6. To create a secure MQTT connection, obtain the root CA certificate for the Watson IoT Platform. The root CA certificate is the last certificate in the full certificate chain messaging.pem, provided by IBM Cloud Docs. Copy only *-----BEGIN CERTIFICATE-----* to *-----END CERTIFICATE-----*, paste it into an empty text file, and save it as messaging_root_ca.pem in PEM format (if this file name is changed, *Mqtt_Client_secure_files* in *mqtt_client_task.c* must be updated).
7. Add this PEM file to a new UniFlash ImageCreator project. For detailed instructions on adding a user file to the SimpleLink Wi-Fi's external serial flash, see the UniFlash ImageCreator Basics SimpleLink Academy lab.
8. The design by default is configured for BLE Provisioning. Users can recompile the application to use AP Provisioning or SmartConfig by removing the BLE module. This can be done by changing the *BLE_PROVISIONING* define in *wifi_thermostat_app.h* to 0. If AP Provisioning is used, the dummy-root-ca-cert-key (<SDK install location>/simplelink_cc32xx_sdk_2_10_00_04/tools/cc32xx_tools/certificate-playground/dummy-root-ca-cert-key) must also be added as a user file to the serial flash.
9. To demonstrate OTA functionality, import and configure the OTA library from the CC3220 SDK. For detailed instructions on preparing and loading an OTA image, see the Wi-Fi OTA SimpleLink Academy lab.
10. Rebuild the Wi-Fi thermostat project. Run a debug session in Code Composer Studio or load the MCU image to the serial flash with UniFlash ImageCreator. For detailed instructions on programming an image to the SimpleLink Wi-Fi's external serial flash, see the UniFlash ImageCreator Basics SimpleLink Academy lab.

3.2.5 Build simple_np Application and Flash CC2640R2F

Regardless of which method is used to evaluate the CC3220SF device application, the CC2640R2F device must first be programmed with the SNP (simple_np) application. The following steps describe how to modify and build the simple_np project.

1. Import the simple_np project from CC2640R2 SDK into a CCS workspace (at <SDK Install Location>/ simplelink_cc2640r2_sdk_1_50_00_58/examples/rtos/CC2640R2_LAUNCHXL/ blestack/simple_np).
2. Right-click on the simple_np_cc2640r2lp_app project that appears in the project explorer, and select the Properties button.
3. Open the Predefined Symbols view in the Project Properties window (under Build → Arm Compiler → Predefined Symbols).
4. Ensure the following symbols are defined:
   - *NPI_USE_UART*
   - *POWER_SAVING*
5. Open the board_cc2640r2lp.h file used by the project (at <SDK Install Location> / simplelink_cc2640r2_sdk_1_50_00_58/examples/rtos/CC2640R2_LAUNCHXL/blestack/simple_np/src/app/ board_cc2640r2lp).
6. Change the *Board_MRDY* and *Board_SRDY* definitions to:

```
144 /* UART Board */
145 #define CC2640R2_LAUNCHXL_UART_RX     IOID_2      /* RXD */
146 #define CC2640R2_LAUNCHXL_UART_TX     IOID_3      /* TXD */
147 #define CC2640R2_LAUNCHXL_UART_CTS    IOID_19     /* CTS */
148 #define CC2640R2_LAUNCHXL_UART_RTS    IOID_18     /* RTS */
149
150 /* MRDY/SRDY Board */
151 #define Board_MRDY                    IOID_8      /* MRDY */
152 #define Board_SRDY                    IOID_11     /* SRDY */
```

7. Right-click on the simple_np_cc2640r2lp_app project in the workspace, and select the Rebuild Project button.

Rebuilding the application generates a file called simple_np_cc2640r2lp_app.hex in the FlashROM_StackLibrary folder of the project. Use the Flash Programmer 2 tool to program the LAUNCHXL-CC2640R2 with the simple_np_cc2640r2lp_app.hex file.

3.2.6 Running Wi-Fi® Thermostat Demo

1. The first screen on start-up is the calibration screen. Carefully tap the boxes onscreen to calibrate the resistive-touch functionality.
2. When calibrated, the application stays on the calibration data page until the CC3220 device is connected. If the device cannot connect to a known profile before the timeout, the device enables BLE Provisioning. For detailed instructions on how to use the SimpleLink SDK Explorer mobile app to provision the CC3220 device, see the SimpleLink SDK Explorer section of the BLE Wi-Fi Provisioning README.
   - If the application has been configured for AP Provisioning and SmartConfig, this is enabled instead of BLE provisioning. For detailed instructions on how to use the Wi-Fi Starter Pro mobile app to provision the CC3220 device, see the Wi-Fi Provisioning SimpleLink Academy lab.
3. After successfully provisioning, the main thermostat screen appears. Use a browser to navigate to the GUI Composer Cloud Tool at dev.ti.com/gc (you will need a TI.com login), and move the set temperature slider on the GUI to trigger the cloud connectivity.

3.2.7 Implementation

3.2.7.1 Program Flow

This TI design application is composed of a number of software modules that are run as individual threads. This multithreading approach allows each component to be easily removed or reused for other applications. Details of each module are listed in Table 3-3.

Table 3-3. Wi-Fi Thermostat Software Modules

Module	Source Files	Description	Priority
Wi-Fi Thermostat Application	wifi_thermostat_app.h/.c	Starts Wi-Fi thermostat application by initializing system and spawning threads. Exits after initialization is complete.	1
Control	control_task.h/.c	Handles system-level requests such as system resets	11
Wi-Fi Network Connection Management	network_if.h/.c	Connects CC3220 device to an AP based on profiles, or handles events to start provisioning	10
Wi-Fi Provisioning	provisioning_task.h/.c	Starts or stops AP provisioning and SmartConfig process and executes state machine to handle provisioning events	9
BLE Provisioning	ble_provisioning.h/.c	Starts or stops Bluetooth low energy provisioning process and executes state machine to handle provisioning events	8
Wi-Fi OTA	cloud_ota.h/.c	Runs OTA state machine	6
MQTT Client and MQTT Client Receive	mqtt_client_task.h/.c and mqtt_client_cbs.h/.c	Runs MQTT client, receives incoming MQTT messages, publishes MQTT messages to signal device state, and triggers events based on MQTT messages	4/5
Resistive Touch	touch_task.h/.c	Initializes and polls the resistive touch screen for user input	3
Sensors	demo_thermostat.h/.c	Configures and reads sensor data	2
Thermostat Demo	demo_thermostat.h/.c	Maintains thermostat state based on received sensor data, MQTT messages, and touch notifications, and triggers the HVAC system and user responses via the display	1

The application begins *main_tirtos.c* which spawns a *mainThread* that initializes the MCU peripheral drivers, the Wi-Fi networking layer, RTOS objects, and spawns each of the module threads before exiting (*wifi_thermostat_app.h/.c*). Each of these threads pends on each thread's initialization before beginning their main execution.

3.2.7.2 Provisioning Device

The AP provisioning and SmartConfig process is handled by a dedicated provisioning task (*provisioning_task.h/.c*). The task remains idle until the provisioning state machine is started by the network connection task. When the provisioning state machine is started, the provisioning task puts the network processor in the provisioning state and handles asynchronous events until the system successfully connects to the network. Whenever the application is restarted, the system always attempts to connect to a network based on a stored profile and skip the provisioning process if possible. For additional details on the implementation of SmartConfig and AP Provisioning, see the Wi-Fi Provisioning SimpleLink Academy lab.

BLE provisioning is implemented with a modified version of the BLE provisioning example from the SimpleLink SDK Bluetooth Plugin (*ble_provisioning.h/.c*). In the thermostat software, a BLE thread is dedicated to handling communication with the CC2640R2F device while the system is being provisioned. Since there is no hardware coexistence mechanism for Wi-Fi and BLE in the system, the software application turns off the Wi-Fi network processor while the BLE interface is active and stops the BLE activity while the Wi-Fi interface is active. To keep the Wi-Fi and BLE activity separate, all of the Wi-Fi activity (such as testing received credentials) is handled inside the network interface task (*network_if.h/.c*) instead of the BLE provisioning task.

3.2.7.3 Secure OTA

The OTA process in the thermostat application is implemented in a software task that executes its own state machine in *cloud_ota.h/.c*. When the OTA task starts, it prevents the user from triggering an OTA update until the system has successfully connected to a local network and acquired an IP address. When connected, the task enters an idle state where it remains until an OTA update is triggered by the user. The user can trigger an OTA update by using MQTT to publish a message to the CC3220 device from the GUI.

When started, the OTA task enters the OTA Run State, where the task downloads the components of the software update and writes the update to the file system until the update is complete. After all the components are downloaded and have been stored on the external serial flash, the OTA task sends a reset request to the control task, to trigger the system to reboot and run with the updated software.

The OTA library used in this example implements a simple HyperText Transfer Protocol (HTTP) client, which supports the Dropbox and GitHub API and can be used to download the software update. The source for the simple HTTP client implementation is in the *OtaHttpClient.h/.c* files of the OTA library. For further details on the SDK example or the OTA library, see the Wi-Fi OTA SimpleLink Academy lab.

3.2.7.4 Secure Cloud Connectivity

This design leverages the MQTT client library from the SimpleLink Wi-Fi CC3220 SDK to communicate with the cloud. When building an application based on the MQTT library, two software threads are responsible for implementing the MQTT client functionality: an overall MQTT Thread and an MQTT Client Thread. The overall MQTT Thread configures the parameters of the MQTT connection, establishes the connection with the MQTT broker, spawns the MQTT Client Thread, and then handles all messages received through the MQTT client callbacks.

The MQTT Client Thread runs a task that is implemented in the MQTT library and is dedicated to receiving incoming messages from the MQTT connection, then passing the messages to the MQTT client callback. In this software, the MQTT client task is implemented as a state machine in *mqtt_client_task.h/.c* that waits for the system to be connected to a local network and then attempts to connect securely to the IBM Cloud Watson IoT platform with TLS and token authentication. When the system is connected to the local network and the broker, the MQTT task runs the main loop to handle received messages from the client callback function (*mqtt_client_cbs.h/.c*).

3.2.7.5 Sensor Interface and Measurements

The thermostat demo uses the BME280 on the Sensors BoosterPack and a PIR and Air Quality Sensor attached to the Grove BoosterPack. If the PIR sensor detects motion, it triggers a GPIO interrupt. If no motion is detected by the PIR before the screen timeout, the LCD backlight and touchscreen polling are turned off. The backlight and touchscreen can be turned back on by the PIR GPIO interrupt.

The BME280 senses humidity, pressure, and ambient temperature and communicates through I^2C. The Air Quality sensor measures harmful gas levels, which are read by the ADC and averaged. These sensor values are updated by the *readSensorsThread*, which pends on the *sensorReadSemaphore* that is posted by the timer callback, *sensorTimeoutHandler*. This timer is configured by *SENSOR_TIMER_MS* in *demo_thermostat.c* and set to two seconds by default. When the MCU is in LPDS, the MCU wakes up to read the sensors in the interval configured by *SENSOR_SLEEP_MS*, set to thirty minutes by default.

3.2.7.6 Power Management

TI Drivers includes a power policy manager that puts the MCU subsystem of the CC3220SF device into low-power modes when the system is idle. In the thermostat design, the MCU enters LPDS any time the idle task is executed and no clocked peripherals are being used. While the PWM supporting the LCD backlight is on, the MCU can only enter sleep mode when idle. The *screenTimeoutHandler* checks if the PIR GPIO interrupt was triggered during the last screen-timer cycle. If not triggered, it turns off the PWM and touchscreen functionality, and allows the RTOS scheduler to enter LPDS when idle. The power policy is configured to wake from LPDS by network activity and the PIR.

The network processor subsystem of the CC3220SF device uses the Normal power policy to handle the amount of time the Wi-Fi NWP spends in a low-power mode between AP beacons and broadcasts, which can significantly reduce the average power consumption of the system. The network task sets the power policy to Normal unless the maximum sleep interval allowed is set longer than 100 ms (generally the standard time between beacons). This is defined by *LSI_DURATION_IN_MS* in *wifi_thermostat_app.h*. The default in this design is 100 milliseconds, which instructs the NWP to not skip any AP beacons. The actual amount of time the Wi-Fi NWP spends in LPDS depends on the NWP's prediction of future activities.

3.3 Average Current Measurement for Thermostat Use Cases for TIDM-1020

The following section details the current measurement made on the thermostat design. The use-case assumptions and details of battery life calculation are also provided. The current measurements include the following periodic events:

- Average current measurement when device is in sleep mode
- Average current measured for processing the Wi-Fi beacons
- Average current measured in active mode, when triggered by the user (wake-up through motion detection)
- Average current to process cloud-based user request to read and set temperature

3.3.1 Test Setup

The average current for the system was measured with the IMETER BoosterPack, compatible with the CC3200 LaunchPad. For more information on the IMETER_BOOST, see the product page.

To measure current on the CC3220SF LaunchPad, remove the VBAT header (J19) and connect V_{IN+} and V_{IN-} (see Figure 3-25).

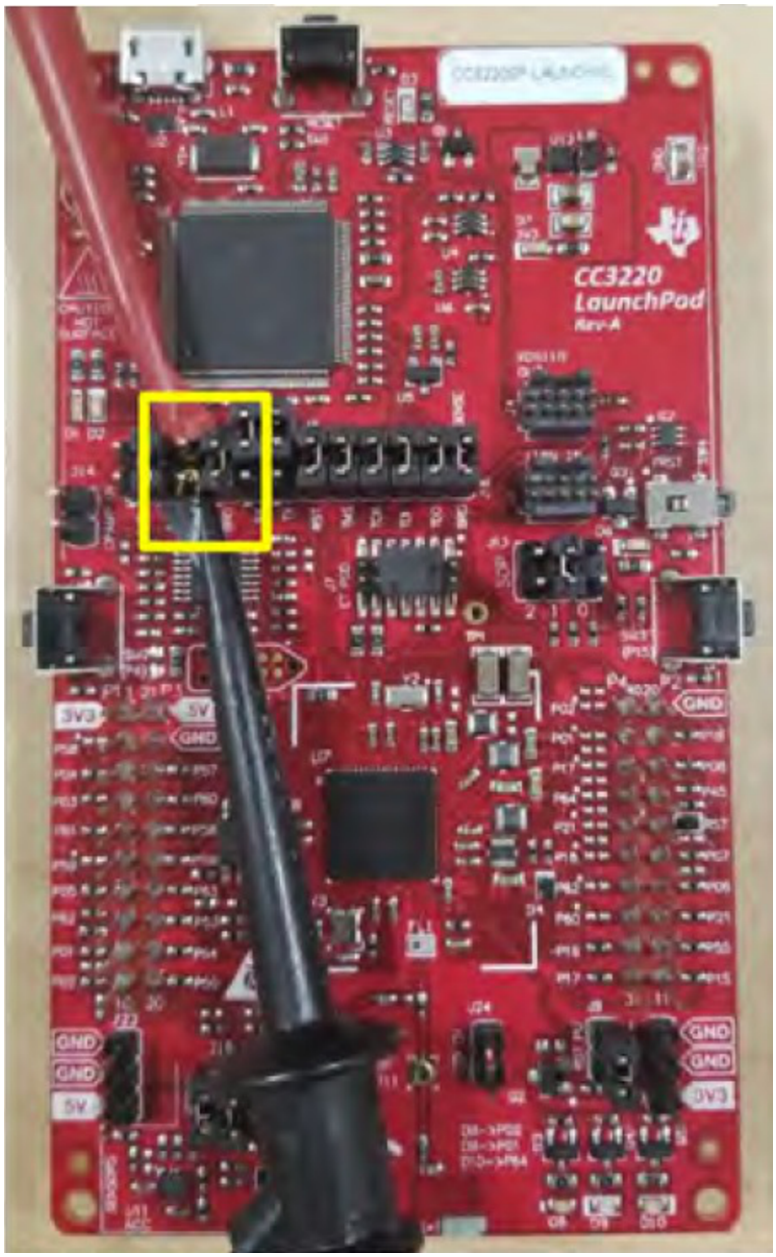


Figure 3-25. Probe Connection for Power Measurement

3.3.2 Test Results

Table 3-4 lists the number of events assumed for the total average-current measurement for one day.

Table 3-4. Number of Thermostat Events Per Day

Parameter	Description	Value	Units
DTIM Interval	Interval at which the AP sends the beacon	100	ms
N_b	Number of DTIM intervals per day	864000	—
N_s	Number of events per day – sensor reading per day	48	—
N_a	Number of events per day – active per day	4	—
N_c	Number of events per day – wakeup through cloud per day	4	—

Table 3-5 lists the time taken to process each event.

Table 3-5. Event Processing Times

Parameter	Description	Value	Units
T_b	Time interval for processing the beacon event	4.2	ms
T_s	Time interval for processing the sensor read event	12.8	ms
T_a	Time interval for processing the HMI interaction, sensor readout, and communicating changed settings to the cloud	25	s
T_c	Time interval for processing the wakeup through the cloud, user activity on the phone (setting limits), and I_s	400	ms

Table 3-6 lists the average current measured for each event.

Table 3-6. Average Current for Each Event

Parameter	Description	Value	Units
I_{Sleep}	Average current during LPDS mode	180	µA
I_b	Average current for processing a beacon interval	13.92	mA
I_s	Average current consumed to read temperature, pressure, humidity, and air quality sensors	33.1	mA
I_a	Average current for I_s, HMI user interaction (setting limits), and communicating changed settings to cloud	26	mA
I_c	Average current for wakeup through the cloud, user activity on phone (setting limits), and I_s	45.2	mA

The average current is computed as Equation 1.

$$I_{avg} = I_{Sleep} + \left(\frac{(I_b \times T_b \times N_b) + (I_s \times T_S \times N_S) + (I_a \times T_a \times N_a) + (I_c \times T_c \times N_c)}{86400\ s / day} \right) \quad (1)$$

$$I_{avg} = 180\ \mu A + \left(\left(13.92\ mA \times 4.2 \times 10^{-3} s \times \frac{86400\ s}{0.1\ s} \right) + (33.1\ mA \times 12.8 \times 10^{-3} s \times 48) + (26\ mA \times 25\ s \times 4) + (45.2\ mA \times 0.4\ s \times 4) \times \left(\frac{1}{86400\ s / day} \right) \right)$$

$$I_{avg} = 900\ \mu A$$

The battery life is estimated using Equation 2. The designs assumes a battery capacity of 3600 mAh.

$$\text{Battery Life} = \left(\frac{\text{Battery Capacity in mAh}}{I_{avg}} \right) \times \left(\frac{1}{720\ hr / month} \right)$$

$$\text{Battery Life} = \left(\frac{3600\ mAh}{0.9\ mA} \right) \times \left(\frac{1}{720\ hr / month} \right) \approx 5.5\ months \quad (2)$$

4 Design Files

The TIDM-1020 is based on existing LaunchPad kits and BoosterPack modules. To download the schematics, bill of materials, PCB layout recommendations, Gerber files, assembly drawings, and software files, see the design files.

- SimpleLink™ Wi-Fi® CC3220SF Wireless Microcontroller LaunchPad™ Development Kit
- SimpleLink™ Bluetooth®v low energy CC2640R2F Wireless MCU LaunchPad™ Development Kit
- Sensors BoosterPack Plug-In Module
- Kentec QVGA Display BoosterPack
- Grove Base BoosterPack for Texas Instruments LaunchPad™ Development Kit
- Grove – PIR Motion Sensor
- Grove – Air Quality Sensor v1.3
- Grove – SPDT Relay

5 Related Documentation

5.1 Product Pages

- SimpleLink™ Wi-Fi® Main Page
- SimpleLink™ CC3220 Main Page
- SimpleLink™ CC2640R2 Main Page
- TI E2E Support Community
- SimpleLink™ MCU Platform

5.2 Application Notes

- Designing Thermostats With CC3220 SimpleLink™ Single-Chip Wi-Fi® MCU System-on-Chip
- CC3120, CC3220 SimpleLink™ Wi-Fi® Internet-on-a chip™ Solution Device Provisioning
- SimpleLink™ CC3120, CC3220 Wi-Fi® Internet-on-a chip™ Solution Built-In Security Features
- SimpleLink™ CC3120, CC3220 Wi-Fi® Internet-on-a chip™ Networking Subsystem Power Management
- CC3x20 SimpleLink™ Wi-Fi® and Internet-of-Things Over-the-Air Update

5.3 Software

- SimpleLink™ CC3220 SDK
- SimpleLink™ CC2640R2 SDK
- TI Resource Explorer
- SimpleLink™ Academy: Provisioning
- SimpleLink™ Academy: Cloud OTA
- SimpleLink™ Academy: MQTT Client Server
- SDK Code Example: Sensor Interface

5.4 Blogs

- What Are You Sensing? Pros and Cons of Four Temperature Sensor Types
- Strengthening Wi-Fi Security at the Hardware Level

5.5 Videos

Thermostat Video

5.6 Trademarks

TI E2E™, SmartConfig™, SimpleLink™, LaunchPad™, BoosterPack™, MSP432™, Code Composer Studio™, and Internet-on-a chip™ are trademarks of Texas Instruments.
Android™ is a trademark of Google, LLC.
Dropbox™ is a trademark of Dropbox, Inc.
Wi-Fi®, Wi-Fi CERTIFIED®, and Wi-Fi Direct® are registered trademarks of Wi-Fi Alliance.
Watson® and Cloud® are registered trademarks of IBM.
Amazon Web Services® is a registered trademark of Amazon.
Azure® is a registered trademark of Microsoft Corporation.
HomeKit® is a registered trademark of Apple.
Bluetooth® is a registered trademark of Bluetooth SIG.
Arm® and Cortex® are registered trademarks of Arm Limited.
IAR Embedded Workbench® is a registered trademark of IAR Systems AB.
All other trademarks are the property of their respective owners.

6 Terminology

OTA	Over the Air update: update device firmware over a wireless connection
IDE	Integrated development environment: software tool for firmware development
HMI	Human machine interface
TCP	Transmission control protocol
CDN	Content delivery network
AP	Access point
PIR	Passive infrared
SPI	Serial peripheral interface
I^2C	Inter-integrated circuit
ADC	Analog-to-digital converter

7 About the Authors

SUDHARSHAN NAGARATHNAM is a Systems Application Engineer at Texas Instruments, where he is responsible for developing a deep knowledge of end equipment, and driving the reference design strategy for SimpleLink Wi-Fi MCU-based end equipment. Sudharshan's current focus is on end equipment in industrial and building automation sectors. Sudharshan brings to this role his extensive experience in embedded systems and algorithm development for low power cellular, consumer, and industrial devices. Sudharshan is a member of the Group Technical Staff (MGTS) at Texas Instruments.

SARAH PELOSI is an Applications Engineer on the SimpleLink Wi-Fi team at TI, where she supports customers and the SimpleLink Academy training platform. Sarah earned her bachelor of science in computer engineering from Bucknell University in Lewisburg, Pennsylvania.

DAVID LARA is an Applications Engineer on the MSP432 Customer Applications Team at TI, and has been in the Embedded Processing business unit since 2012. David earned his masters of science in electrical engineering from New Mexico State University in Las Cruces, NM.

8 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (August 2018) to Revision A (September 2020)	Page
• Added WPA2 + PMF and WPA3 in Section 2.2.1, *SimpleLink™ Wireless MCU CC3220SF*	5

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265