TI-RSLK

Texas Instruments Robotics System Learning Kit

Module 20

Lab: Wi-Fi

Lab: Wi-Fi

20.0 Objectives

The purpose of this lab is to interface a Wi-Fi radio to the microcontroller and connect to cloud services.

1. You will connect a Wi-Fi radio to the microcontroller.
2. You will set up TI-RTOS
3. You will use the synchronous serial protocol to communicate.
4. You will connect the system to cloud services.

Good to Know: There are many possible applications you can implement once your system is connected to the internet. You will be able to provide remote access to the robot or give it data from external sources or sensors that can potentially feed into its control logic.

20.1 Getting Started

20.1.1 Software Starter Projects

Look at these projects:
network_terminal_MSP_EXP432P401R_tirtos_ccs (from TI download)
Lab20_WiFi (starter project for this lab)

20.1.2 Student Resources (in datasheets directory-Links)

CC3120.pdf (SimpleLink™ Wi-Fi Wireless Network Processor*)*

20.1.3 Reading Materials

Volume 2 Sections 11.3 and 11.4
Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller", and
Volume 3 Chapters 3, 4 and 5
Embedded Systems: Real-Time Operating Systems for ARM Cortex-M Microcontrollers

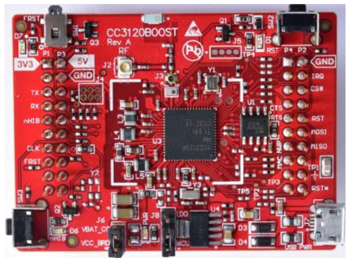


Figure 1. CC3120 Wi-Fi BoosterPack.

20.1.4 Components needed for this lab

Quantity	Description	Manufacturer	Mfg P/N
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	CC3120 Wi-Fi BoosterPack	TI	CC3120BOOST

20.1.5 Lab equipment needed

Wi-Fi router (with internet connection) or cellular hotspot

20.2 System Design Requirements

The overall goal of this lab is to interface a Wi-Fi radio to the microcontroller and use it to connect to the local Wi-Fi router and then interact with a cloud service.

A Wi-Fi device can operate in two main modes. **Station mode** allows it connect to a local access point (AP). For example, your smart home device connects to your house Wi-Fi router or your cell phone connects to the airport Wi-Fi network. The device is in station mode and the router is in AP mode. **AP mode** lets the device act as an access point with a broadcast SSID that other devices can connect to. This is most commonly used by your local router but can also be used by a device if we need to do some first time setup or if we only need local WAN information and no internet access is required. An AP has a limited number of connections it can service at any given time. High end routers can handle hundreds of connections. The CC3120 can handle four simultaneous connections in AP mode.

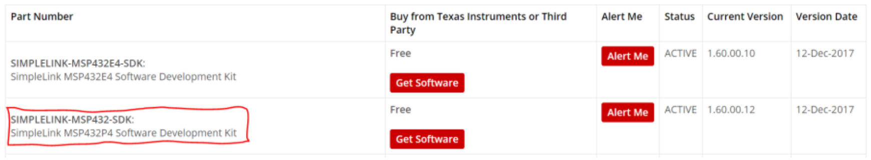
Using SimpleLink Wi-Fi is a more complex operation than what we have done in previous labs. To use SimpleLink Wi-Fi to its full ability, we will make use of a Real-Time Operating System (RTOS) specifically TI-RTOS. TI-RTOS is a free to use RTOS available from TI and optimized on TI processors. This module will not go too deeply into RTOS concepts, but we will be using TI-RTOS to enable our Wi-Fi communication and give you some exposure to how using an RTOS in a system is initialized.

20.3 Experiment set-up

In this first section we are going to setup the SimpleLink SDKs in our CCS environment. You will need to download and install two SDKs for use in your workspace.

TI provides a SimpleLink SDK (software development kit) to enable development with the MSP432 and provide many additional options for the software development of the microcontroller.

First, download the SimpleLink MSP432P4 SDK from http://www.ti.com/tool/SIMPLELINK-MSP432-SDK and download version 1.60.00.12 or later. Be sure to download the SDK for the MSP432P4 and not the E4. Run the downloaded application to install the plugin. Alternatively, you can also download it from the Resource Explorer front page (double check the version number). To access Resource Explorer go to the top menu View → Resource Explorer. You may need to restart CCS to complete the installation.

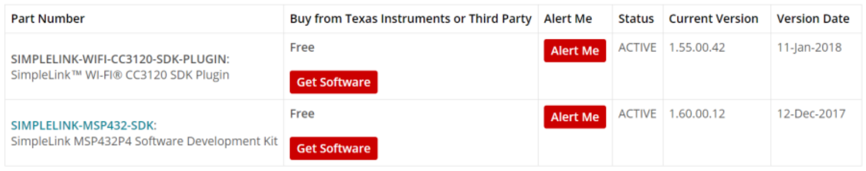


Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date
SIMPLELINK-MSP432E4-SDK: SimpleLink MSP432E4 Software Development Kit	Free Get Software	Alert Me	ACTIVE	1.60.00.10	12-Dec-2017
SIMPLELINK-MSP432-SDK: SimpleLink MSP432P4 Software Development Kit	Free Get Software	Alert Me	ACTIVE	1.60.00.12	12-Dec-2017

SimpleLink plugins are intended to extend functionality of each individual platform SDK to include specialized use-cases such as adding wireless functionality.

While all of the plugins have the same basic structure and look-and-feel of an SDK, they are not meant as standalone applications and rely heavily on components from the platform SDK. The SimpleLink Wi-Fi SDK Plugin, for example, relies heavily on the TI-Drivers and RTOS kernel components from the MSP432P4 SDK.

Second, download the SimpleLink Wi-Fi SDK Plugin from http://www.ti.com/tool/simplelink-wifi-cc3120-sdk-plugin and download version 1.55.00.42 or later. Be sure to download the SIMPLELINK-WIFI-CC3120-SDK-PLUGIN. Run the downloaded application to install the plugin. Alternatively, you can also download it from Resource Explorer front page (double check the version number). To access Resource Explorer go to the top menu View → Resource Explorer. You may need to restart CCS to complete the installation.



Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date
SIMPLELINK-WIFI-CC3120-SDK-PLUGIN: SimpleLink™ WI-FI® CC3120 SDK Plugin	Free Get Software	Alert Me	ACTIVE	1.55.00.42	11-Jan-2018
SIMPLELINK-MSP432-SDK: SimpleLink MSP432P4 Software Development Kit	Free Get Software	Alert Me	ACTIVE	1.60.00.12	12-Dec-2017

Third, run those downloaded executable files to start the installer. Use the default location C:\ti as the destination folder. When complete both SDKs should reside in the C:\ti\simplelink_msp432p4_sdk_1_60_00_12 and C:\ti\simplelink_sdk_wifi_plugin_1_55_00_42 file paths or similar. Restart your CCS session and those new SDKs should be detected by the IDE.

In this next section, we will do a TI-RTOS specific requirement of importing the kernel project to our workspace. The kernel is the main code base required to run the RTOS or any type of modern operating system. This project contains the TI-RTOS kernel and is needed for our Wi-Fi example. The kernel build project comes in a variety of flavors such as release and debug, tirtos and nortos, ccs and gcc compilers. For this setup you must choose the release version of the **tirtos kernel** using the **ccs compiler**. Start CCS and execute File → Import → CCS Projects and go into the file path for the MSP432P4 SDK **C:\ti\simplelink_msp432p4_sdk_1_60_00_12\kernel\tirtos\builds\MSP_EXP432P401R\release\ccs**
Click finish to import. With the TI-RTOS kernel properly imported we are ready to utilize the TI-RTOS based examples in our CCS workspace. We will make use of this in section 20.4.3.

Lab: Wi-Fi

The SimpleLink Wi-Fi SDK Plugin is designed for development on the CC3120 Network Processor and MSP432 Host MCU. The CC3120 and MSP432 communicate over the SPI or UART host interface. We will use SPI. The CC3120 requires an external host MCU for the user application.

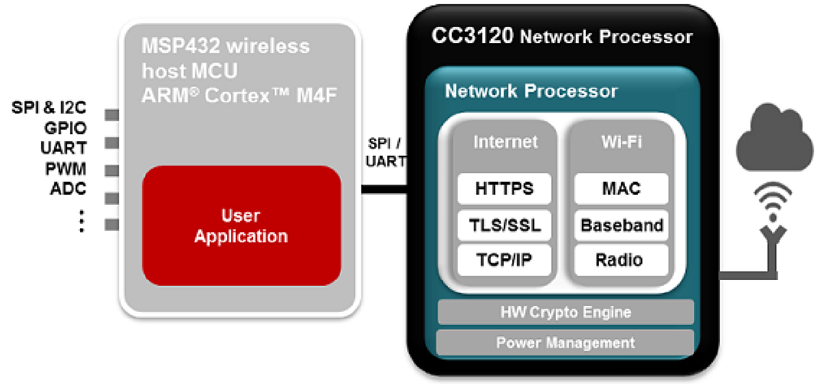


Figure 2. Block Diagram of MSP432 and CC3120 interface.

We will set up the hardware. Disconnect your MSP432 from the robot and Remove any BoosterPacks from previous modules.

1. Mount the CC3120 BoosterPack on top of a MSP432 LaunchPad so the pins align as shown. Be careful not to bend any pins.

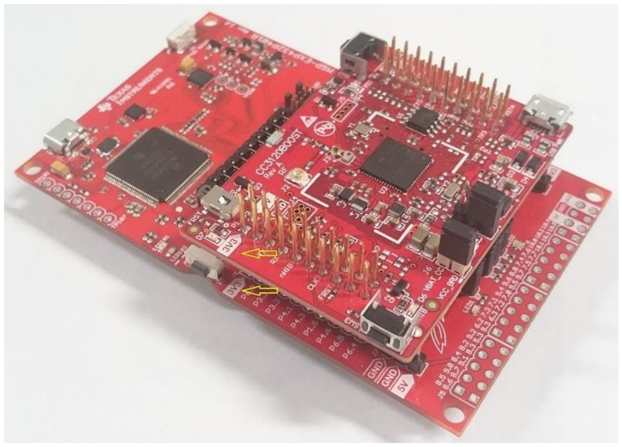


Figure 3. Wi-Fi BoosterPack positioned on top of MSP432 with markers aligned.

2. Plug a micro-USB connector to the MSP432 board. This is used as a power source and the programming interface for the user application. There is no need to plug in to the USB connector on the BoosterPack, this is used for external power source but not needed in the context of this application. You just want one USB cable to the MSP432 LaunchPad like the previous labs.

You will implement this lab using the MSP432 LaunchPad and the CC3120 Wi-Fi BoosterPack. Note that the TI documentation references the CC31XXEMUBOOST for updating firmware, but you will not need that to complete the activities in TI-RSLK.

Lab: Wi-Fi

Tables 4 through 7 show the pins used by the MSP432-CC3120 interface. When extending the system beyond the activities in this lab, you can use any of the pins marked unused for additional interfaces.

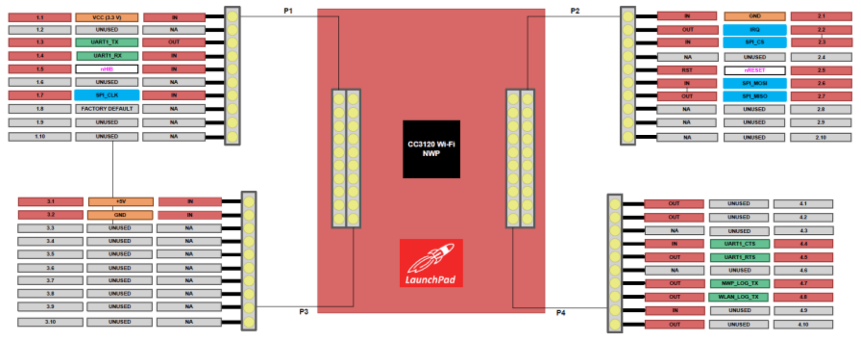


Figure 4. Pin Connections diagram

Pin Number	Signal Name	Direction	Pin Number	Signal Name	Direction
P1.1	VCC (3.3 V)	IN	P2.1	GND	IN
P1.2	UNUSED	NA	P2.2	IRQ	OUT
P1.3	UART1_TX	OUT	P2.3	SPI_CS	IN
P1.4	UART1_RX	IN	P2.4	UNUSED	NA
P1.5	nHIB	IN	P2.5	nRESET	IN
P1.6	UNUSED	NA	P2.6	SPI_MOSI	IN

Table 5. J1 Pin Connections

Pin Number	Signal Name	Direction	Pin Number	Signal Name	Direction
P1.7	SPI_CLK	IN	P2.7	SPI_MISO	OUT
P1.8	FACTORY DEFAULT	NA	P2.8	UNUSED	NA
P1.9	UNUSED	NA	P2.9	UNUSED	NA
P1.10	UNUSED	NA	P2.10	UNUSED	NA

Table 6. J2 Pin Connections

Pin Number	Signal Name	Direction	Pin Number	Signal Name	Direction
P3.1	+5 V	IN	P4.1	UNUSED	OUT
P3.2	GND	IN	P4.2	UNUSED	OUT
P3.3	UNUSED	NA	P4.3	UNUSED	NA
P3.4	UNUSED	NA	P4.4	UART1_CTS	IN
P3.5	UNUSED	NA	P4.5	UART1_RTS	OUT
P3.6	UNUSED	NA	P4.6	UNUSED	NA
P3.7	UNUSED	NA	P4.7	NWP_LOG_TX	OUT
P3.8	UNUSED	NA	P4.8	WLAN_LOG_TX	OUT
P3.9	UNUSED	NA	P4.9	UNUSED	IN
P3.10	UNUSED	NA	P4.10	UNUSED	OUT

Table 7. J3 and J4 Pin Connections

20.4 System Development Plan

20.4.1 Loading the Network Terminal Program

Primary: Import from SimpleLink CC3120 SDK Plugin

1. File → Import → CCS Project and navigate to C:\ti\simplelink_sdk_wifi_plugin_1_55_00_42\examples\rtos\MSP_EXP432P401R\demos\network_terminal
2. Click OK
3. Build the project by selecting *Build Project* from the Project menu or right-clicking the name of the project. It may take a couple minutes to build.
4. **Using CCS Debugger:** Start a Debug session by clicking the green bug in the top menu. Your MSP432 LaunchPad will need to be plugged in.
5. Open a UART terminal (or two) on your device's COM port. We want to use the XDS110 Class Application/User UART port with the following parameters:

   UART Configuration
   Baud rate: 115200
   Data: 8 bit
   Parity: None
   Stop: 1 bit
   Flow control: None

   To open in CCS go to View > Other… > Terminal > Terminal. Open a new Terminal with the above configuration

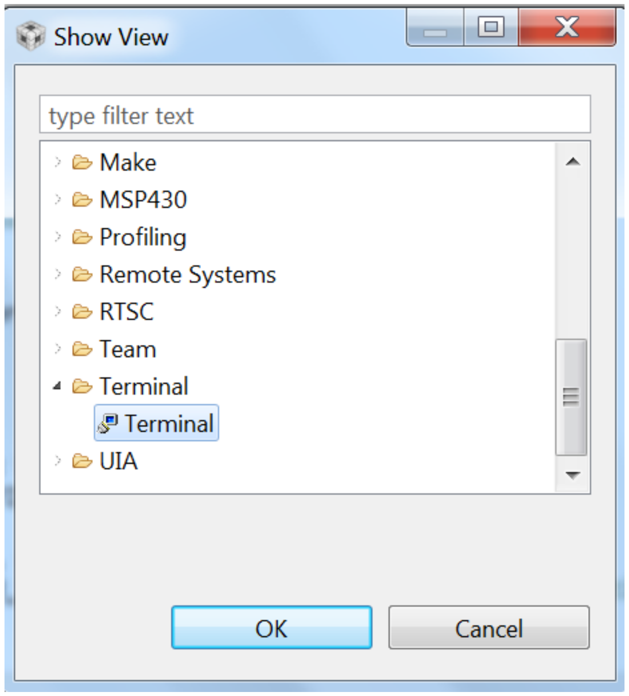
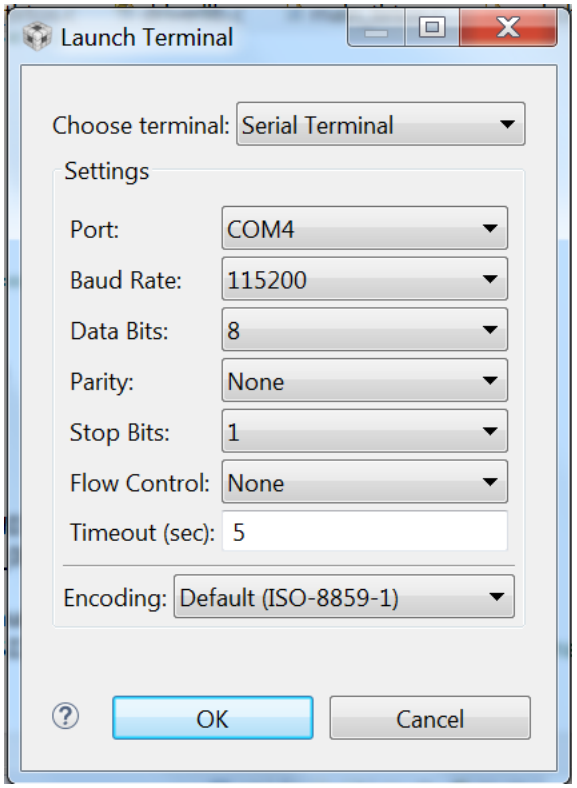


Figure 8. Show view options

Connection type: Serial Port
Click New connection. In the Launch Terminal pop up select Serial Terminal and the COM port. Confirm the configuration is the correct Baud and click ok.



Choose terminal: Serial Terminal
Settings
Port: COM4
Baud Rate: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None
Timeout (sec): 5
Encoding: Default (ISO-8859-1)

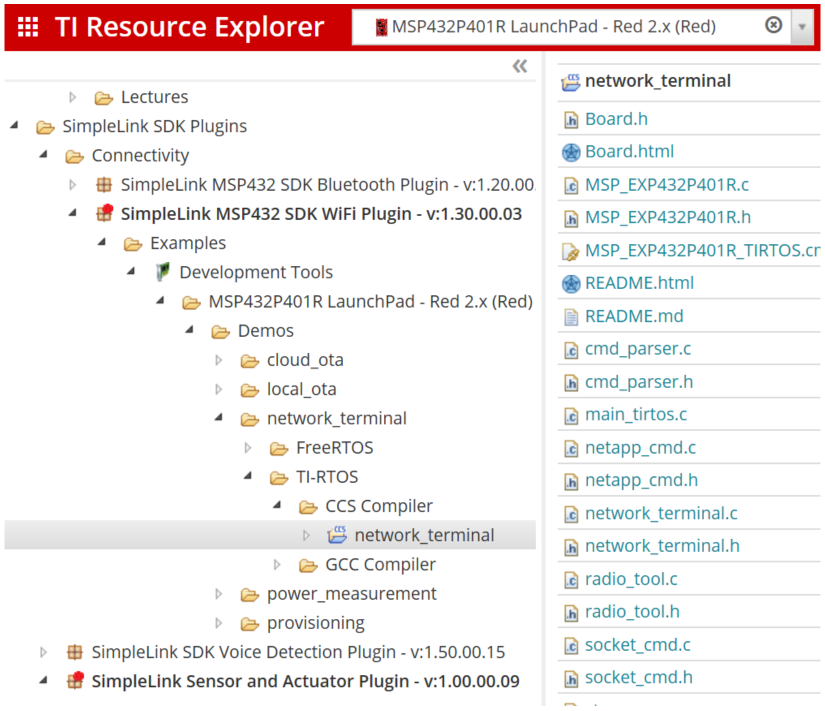
Figure 9. Serial Terminal options

6. Using the CCS debugger, click the green arrow in the top menu to start executing your code. You can now access the Wi-Fi commands by typing them in the terminal. We will explain in 20.4.2 the available commands and then use these to ping TI's website.

Lab: Wi-Fi

Alternative procedure to load example: Using Resource Explorer

1. In CCS, open the TI Resource Explorer (View → Resource Explorer)
2. Type in MSP432 in top search bar and select the MSP432P401R LaunchPad to filter the results to our LaunchPad board.
3. Find the SimpleLink SDK Plugins folder. Expand the connectivity folder and Wi-Fi plugin folder. Expand the folders as shown to select the network_terminal example (Examples → Development Tools → MSP432P401R LaunchPad → Demos → network_terminal → TI-RTOS → CCS Compiler → network_terminal), then click the Import to IDE icon at the top-right to download the code to the IDE and install any dependencies.



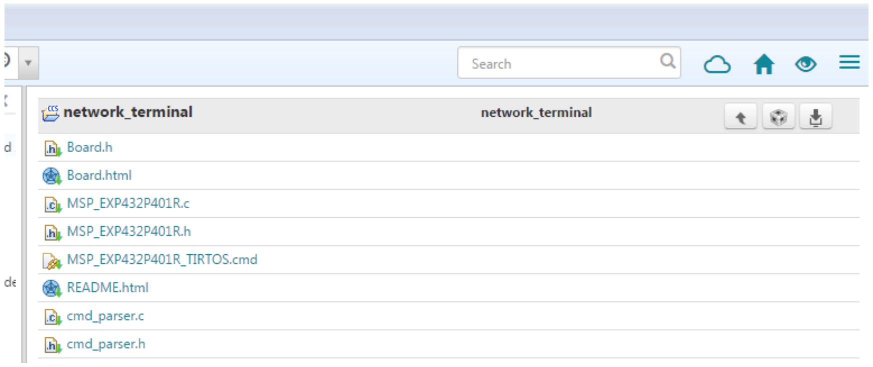
TI Resource Explorer | MSP432P401R LaunchPad - Red 2.x (Red)

```
  ▹ Lectures
◢ SimpleLink SDK Plugins
  ◢ Connectivity
    ▹ SimpleLink MSP432 SDK Bluetooth Plugin - v:1.20.00
    ◢ SimpleLink MSP432 SDK WiFi Plugin - v:1.30.00.03
      ◢ Examples
        ◢ Development Tools
          ◢ MSP432P401R LaunchPad - Red 2.x (Red)
            ◢ Demos
              ▹ cloud_ota
              ▹ local_ota
              ◢ network_terminal
                ▹ FreeRTOS
                ◢ TI-RTOS
                  ◢ CCS Compiler
                    ▹ network_terminal
                  ▹ GCC Compiler
              ▹ power_measurement
              ▹ provisioning
  ▹ SimpleLink SDK Voice Detection Plugin - v:1.50.00.15
  ◢ SimpleLink Sensor and Actuator Plugin - v:1.00.00.09
```

```
network_terminal
Board.h
Board.html
MSP_EXP432P401R.c
MSP_EXP432P401R.h
MSP_EXP432P401R_TIRTOS.cr
README.html
README.md
cmd_parser.c
cmd_parser.h
main_tirtos.c
netapp_cmd.c
netapp_cmd.h
network_terminal.c
network_terminal.h
radio_tool.c
radio_tool.h
socket_cmd.c
socket_cmd.h
```

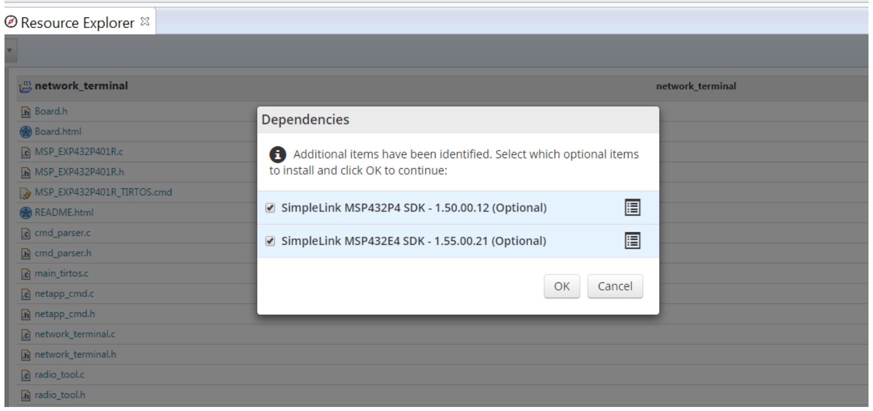
Figure 10. Resource Explorer

- Be sure you select your desired project "flavor" (TI-RTOS, Free-RTOS, CCS, GCC, etc.).
- TI-RTOS + CCS Compiler is recommended. We will be using the TI-RTOS CCS example for this lab: network_terminal_MSP_EXP432P401R_tirtos_ccs



```
network_terminal                              network_terminal
Board.h
Board.html
MSP_EXP432P401R.c
MSP_EXP432P401R.h
MSP_EXP432P401R_TIRTOS.cmd
README.html
cmd_parser.c
cmd_parser.h
```

Figure 11. network_terminal CCS project files



Resource Explorer

Dependencies

Additional items have been identified. Select which optional items to install and click OK to continue:

- ☑ SimpleLink MSP432P4 SDK - 1.50.00.12 (Optional)
- ☑ SimpleLink MSP432E4 SDK - 1.55.00.21 (Optional)

OK | Cancel

Figure 12. Install dependencies dialog box.

4. Restart CCS to install dependencies
5. Click the import to IDE button (CCS Cube icon)
6. Build the project by selecting *Build Project* from the Project menu or right-clicking the name of the project. It may take a couple minutes to build.
7. **Using CCS Debugger:** Start a Debug session by clicking the green bug in the top menu. Your MSP432 LaunchPad will need to be plugged in.
8. Open a UART terminal (or two) on your device's COM port. We want to use the XSD110 Class Application/User UART port with the following parameters:

   UART Configuration
   Baud rate: 115200
   Data: 8 bit
   Parity: None
   Stop: 1 bit
   Flow control: None

   To open in CCS go to View > Other… > Terminal > Terminal. Open a new Terminal with the above configuration

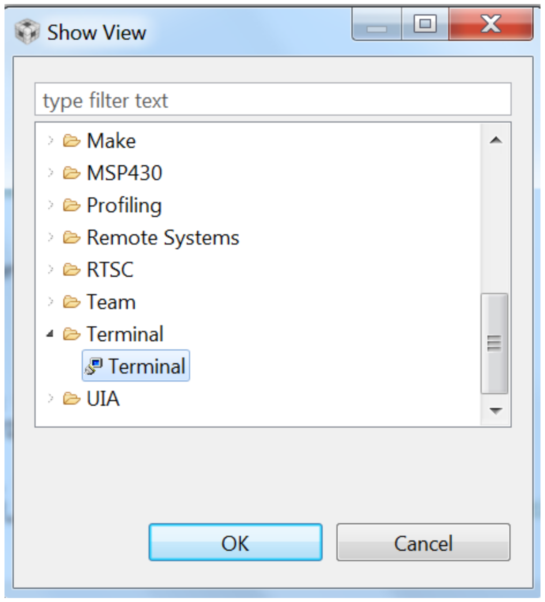
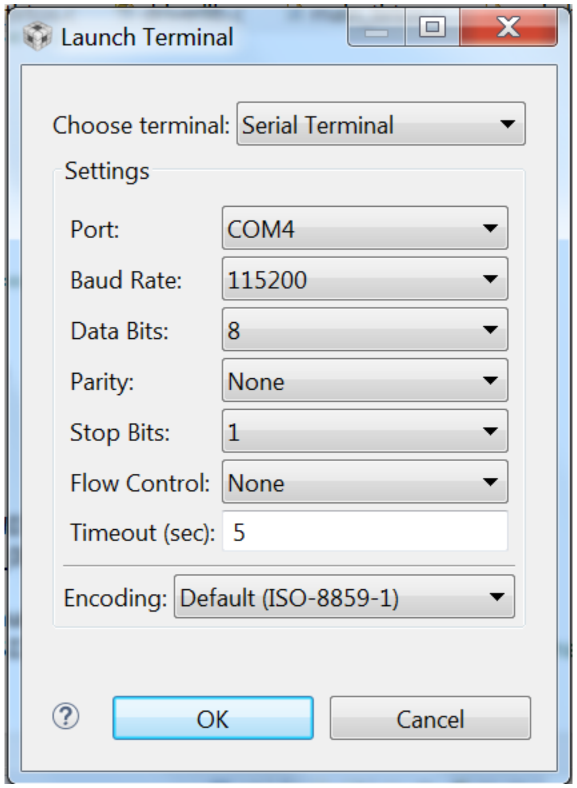


Figure 13. Show view options
Connection type: Serial Port

Click New connection. In the Launch Terminal pop up select Serial Terminal and the COM port. Confirm the configuration is the correct Baud and click ok.



Choose terminal: Serial Terminal
Port: COM4
Baud Rate: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None
Timeout (sec): 5
Encoding: Default (ISO-8859-1)

Figure 14. Serial Terminal options

9. Using the CCS debugger, click the green arrow in the top menu to start executing your code. You can now access the Wi-Fi commands by typing them in the terminal. We will explain the available commands and then use these to ping TI's website.

20.4.2 Using the Network Terminal to connect

You can type help into the terminal at any time to see a complete list of available commands.

WLAN commands

- **scan**: Retrieves scan results from network processor's (NWP) scan cache
- **setpolicy**: Defines the device's scan behavior and starts background scans
- **wlanconnect**: Connects device to an AP
- **wlan_ap_start**: Configures the device to operate in AP mode
- **createfilter**: Creates an RX filter. RX filters are a set of rules and actions imposed on each packet received from the air
- **enablefilter**: Enables all defined filters
- **disablefilter**: Disables all defined filters
- **deletefilter**: Deletes all defined filters
- **enablewowlan**: Defines a pattern-based filter, then sends host MCU to Low Power Deep Sleep (LPDS). Once the pattern filter triggers, the NWP would wake the host MCU from LPDS using host IRQ as a wake up source.
- **p2pstart**: Sets the NWP in discoverable Peer to Peer mode and connects to another visible P2P device

Socket commands

- **send**: Demonstrates opening a TCP or UDP socket, sending data in packets, and closing socket
- **recv**: Demonstrates opening a listening socket, receiving data in packets, and closing socket

NetApp commands

1. **ping**: Pings specific host name or IP, and prints statistics to terminal
2. **mdnsadvertise**: Advertises a service over mDNS
3. **mdnsquery**: Runs mDNS query for services over local LAN

Transceiver commands

- **radiotool**: Starts the Radio Tool. This allows users to run several radio-related tests for RX and TX operations

Learn more about the commands

You can learn more about any command and see an example of usage by typing [command] -help into the terminal.

To ping a website, perform the following steps in the terminal.

1) As a station, connect to an AP with internet access (a hotspot or router, for example).
2) Ping a popular website, such as google.com or ti.com.
3) Send 5 Echo-request packets with a 2 second delay interval.

```
scan -n 20
wlanconnect -s "cc3120-demo" -t WPA/WPA2 -p
"password"
ping -h www.ti.com -c 5 -i 2
```

Use your own AP's SSID and password in the wlanconnect command

You should get a response back from the website. Try entering a different website with different echo and delay interval. If you are not sure what to enter for a command you can type the command with the help flag such as "ping -help" or "wlanconnect -help"

20.4.3 Get weather and time

Now we will need to import an example to our workspace. Download the CCS Project from http://www.ti.com/lit/zip/slac767

Extract the zip file and import the project Lab20_WiFi into your workspace. The project will not build without setting up the MSP432 SDK and importing the kernel project as was described in section 20.3. If you did follow the setup procedure, then the project should build without errors but may contain warnings.

We will now look at the Lab20_WiFi project which will give a demo of connecting to a web server, openweathermap.org and nist.gov, to receive the weather and time. This program is broken down into several files with the **get_time_and_weather.h** and **get_time_and_weather.c** doing the heavy lifting. **main_tirtos.c** is a standard file used to start the RTOS kernel. **network_if.h** and **network_if.c** main job is to set up the MSP432 as a station that will connect to the local Wi-Fi router.

Open up **network_if.h** and change line 62 **#define SSID_NAME** to your router SSID name and line 66 **#define SECURITY_KEY** to your password. The most common router security will be WPA2 which is the default. If you have a password than you should leave the **SEC_TYPE_AP_MODE** on line 64 as default. If you have an open router with no password, you can change line 64

#define SEC_TYPE_AP_MODE to **SL_WLAN_SEC_TYPE_OPEN** and leave the line 64 **#define SECURITY_KEY** as a blank string.

Open **get_time_and_weather.c** and go to line 104. We will change **flashDemoConfigParams** to your Wi-Fi SSID name and password in the first and second arguments. If you have open Wi-Fi with no password again you can change the third argument to **SL_WLAN_SEC_TYPE_OPEN**, but if not you can leave as default. The fifth argument is a city. You can leave as default to get data from Dallas, TX or you can change to another city using this string. It is recommended to try to get the default Dallas data first and then change the city later.

Everything else you can leave as default and go ahead and compile and upload your program to the LaunchPad. Again open up your Terminal inside of CCS to see the UART data of the connection process, plus the current weather and time data. If you are satisfied with the output, try a different city and reflash the program to the LaunchPad. Examine the code for structure and find places where changes could be made to connect to other web servers and output data.

20.4.4 Send an email with IFTTT

Now we will interact with some easy cloud services. Let’s have our robot send an email. We will use a popular aggregator service called **If This Then That**, which is a website that lets us set up rules and triggers to automate a process. For example if the weather forecast rain, send us a notification so we can prepare our umbrella.

1. Sign up for an IFTTT account at **ifttt.com** and associate it with an email address for testing your LaunchPad. Create a new applet from the interface. An applet is a logical connection between two web services supported in IFTTT.

2. Start to choose a service by clicking “this” highlighted in blue.

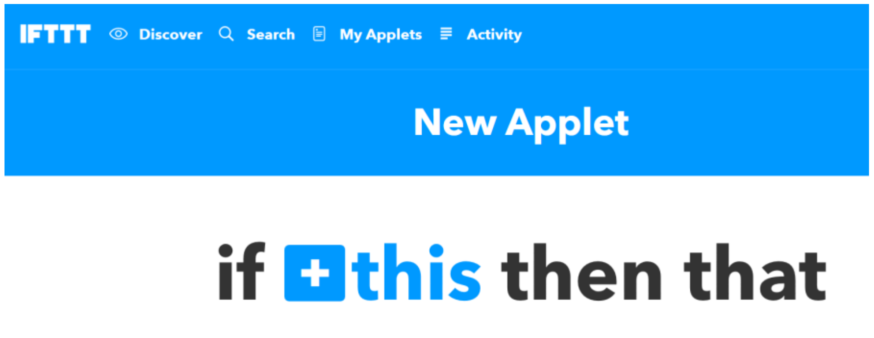


Figure 15. IFTTT new applet this

3. Choose the trigger service by typing “webhooks” and select webhooks which is also called the maker service. Enable the webhooks service if prompted.

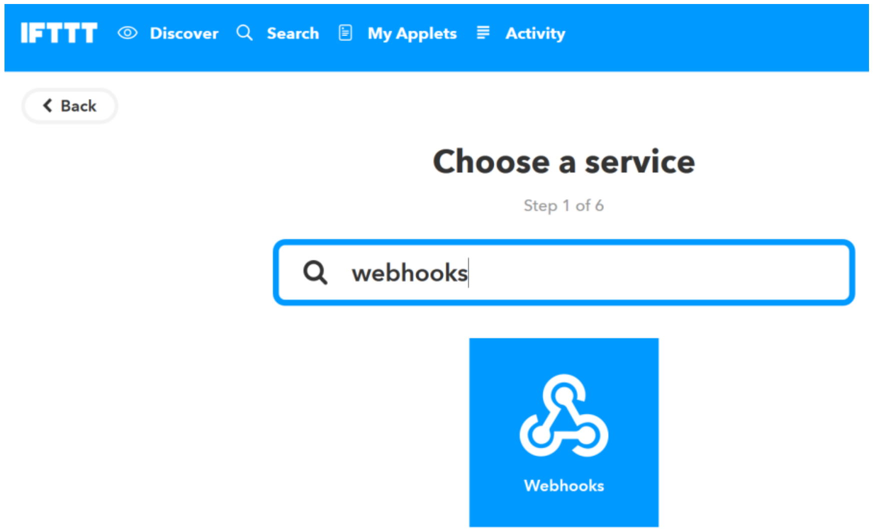
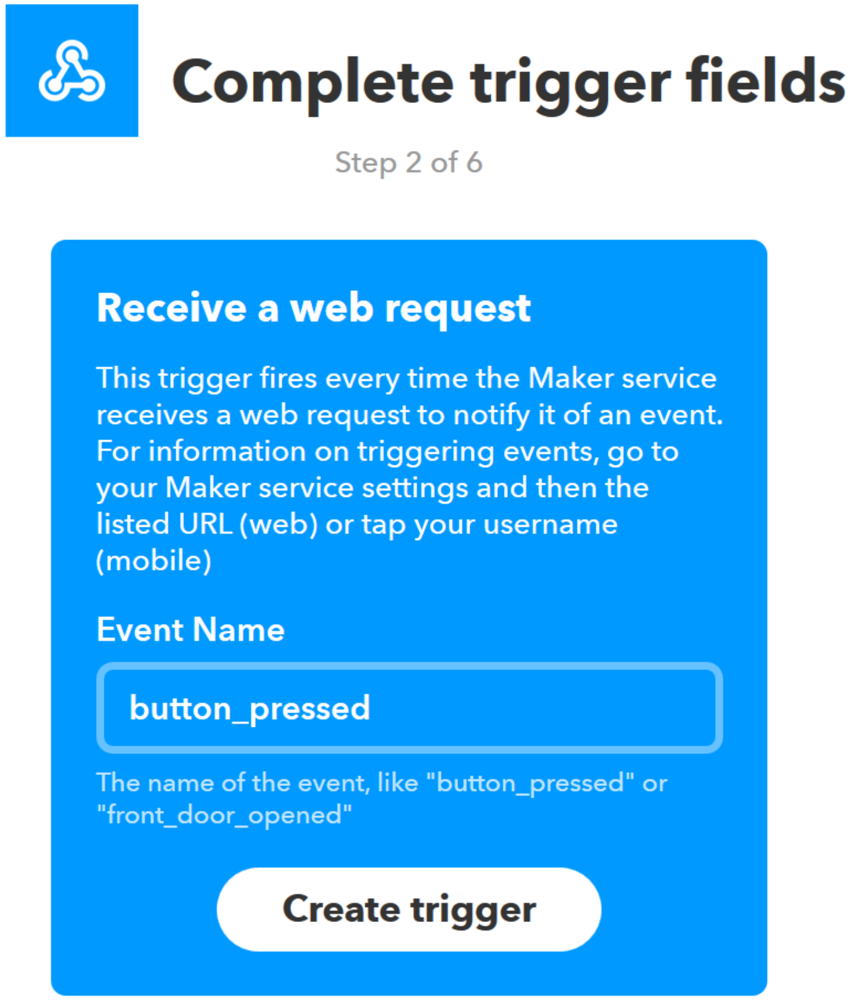


Figure 16. IFTTT choose service

4. Call the event name “button_pressed” or any you have been.



Complete trigger fields

Step 2 of 6

Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Event Name

button_pressed

The name of the event, like "button_pressed" or "front_door_opened"

Figure 17. IFTTT complete trigger fields

5. Start to choose an action by clicking “that” highlighted in blue

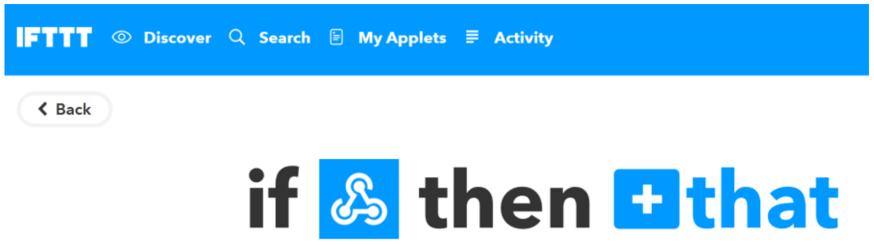


Figure 18. IFTTT new applet that

6. Choose the action service by typing “email” and select email.

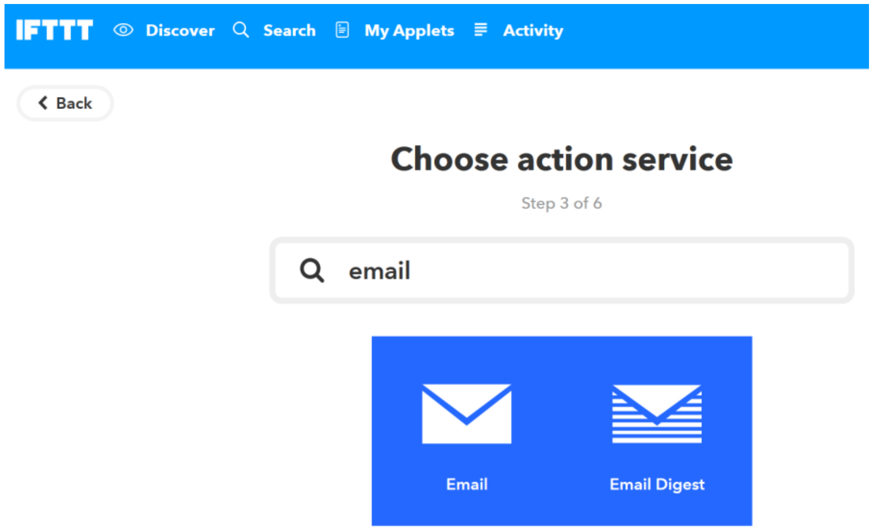
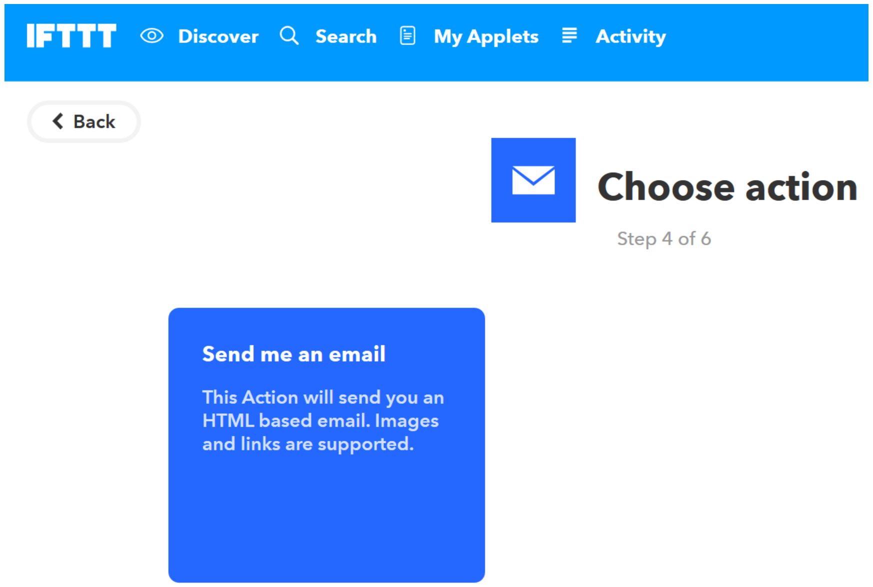


Figure 19. IFTTT choose action service

7. Choose an action “send me an email”

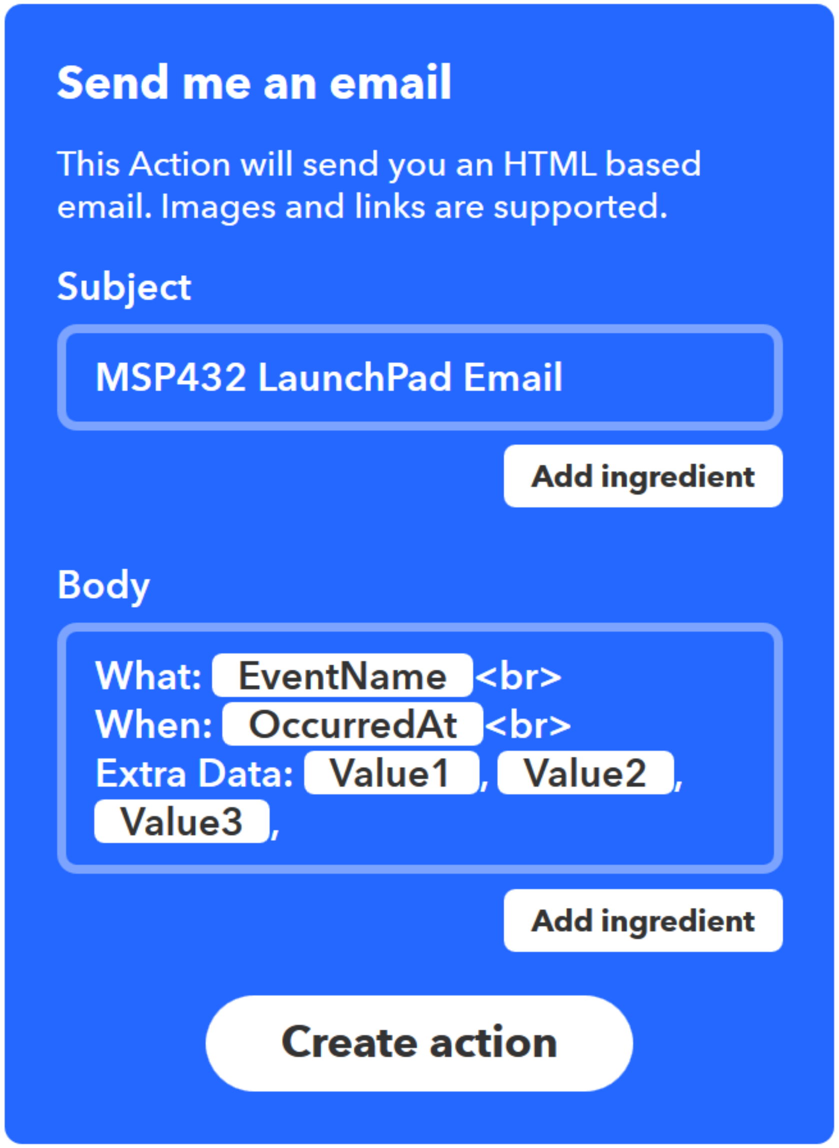


Send me an email

This Action will send you an HTML based email. Images and links are supported.

Figure 20. IFTTT choose action email

8. Set your subject to “MSP432 LaunchPad Email” and leave the body with the default values.


Send me an email

This Action will send you an HTML based email. Images and links are supported.

Subject

MSP432 LaunchPad Email

Body

What: EventName
When: OccurredAt
Extra Data: Value1 , Value2 , Value3 ,

Figure 21. IFTTT action setup

9. Click “Create action” and then your applet is complete.

10. Now we need to go into the settings of the webhooks service. Go into the settings from my applets menu.

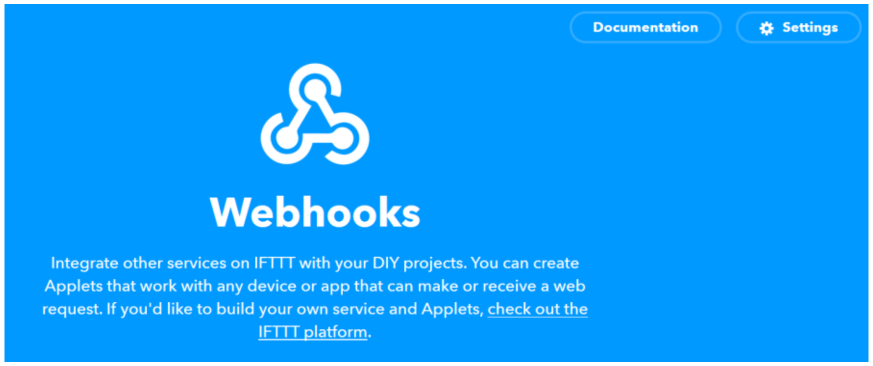
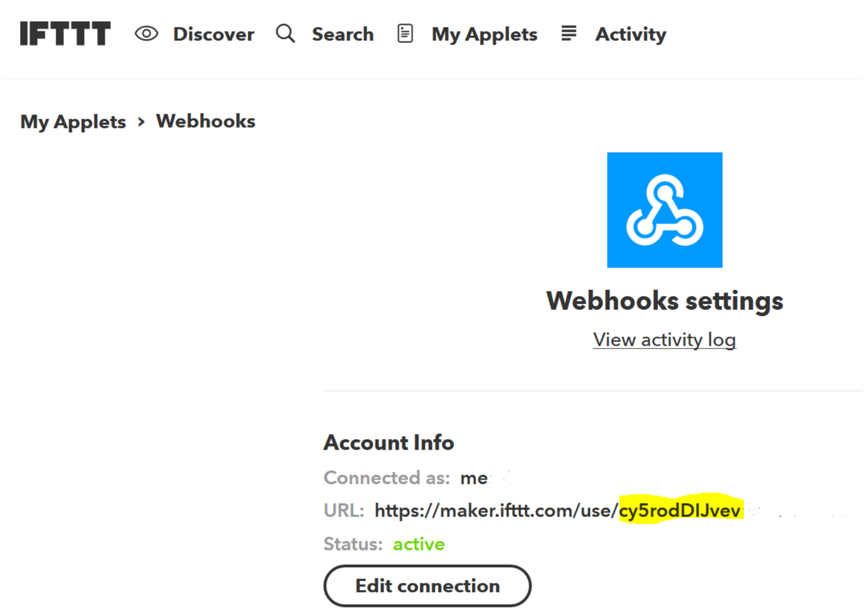

Documentation
Settings
Webhooks
Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.

Figure 22. IFTTT Webhooks settings

Here you will see the URL you need to navigate to with your unique IFTTT key after https://maker.ifttt.com/user/{key}



Account Info
Connected as: me
URL: https://maker.ifttt.com/use/cy5rodDIJvev
Status: active

Figure 23. IFTTT webhooks account information

11. Go to that URL and it will give you instructions on how to use the service.

To send an email through IFTTT we will need to have our LaunchPad ping https://maker.ifttt.com/trigger/{event}/with/key/{key} where event is “button_pressed” and key is your IFTTT key.
https://maker.ifttt.com/trigger/button_pressed/with/key/{key}

12. You can try it out in your web browser first to verify the email sends correctly.

13. Now we will need to set up our CCS code.
Make a copy of **Lab20_WiFi** in your workspace and name it **Lab20_IFTTT**. Look into the **get_time_and_weather.c** and change your server and GET request variables to reflect the IFTTT url. Once you’ve made changes save and upload the code and utilize your debugging skills to get the project converted to the new web service. You are looking in this first stage to send the single query to IFTTT on power up just like the previous example.

The final challenge is to tie the email to a button press. Whenever a button is pressed, fire an interrupt event and trigger the IFTTT email.

20.5 Troubleshooting

The Wi-Fi can't connect to the router:

- Check all the connections between LaunchPad and the BoosterPack and make sure the pins are lined up.
- Make sure the LaunchPad is connected via USB and powered on
- Make sure the Wi-Fi BoosterPack power LED is on when connected to LaunchPad.
- Verify the SSID and password of the router you are connecting to in the code
- Make sure the router does not have a splash screen for logging in. The CC3120 is not able to know what to do with that.

Cloud issues:

- Make sure the router has an internet connection

20.6 Things to think about

In this section, we list thought questions to consider after completing this lab. These questions are meant to test your understanding of the concepts in this lab.

- What does it mean that this interface is serial? Why is serial important?
- What does it mean that this interface is synchronous? Why is synchronous important?

20.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. You could extend the system or propose something completely different. For example,

- Connect your robot to the Dweet.io service and then use the visualization tool freeboard.io to display the data coming from Dweets
- Connect your robot to the Temboo service (www.temboo.com)
- Connect your robot to the Blynk service to enable Wi-Fi mobile app (www.blynk.cc)
- Connect your robot to move based on changes in the stock market
- Set your robot as an AP to transmit diagnostics and sensor data to a connected web client

20.8 Which modules are next?

Modules 1-20 have introduced the basics of the microcontroller and advanced functionality to add to the robot. You should have most of the ground work to complete the robot challenge. Additional supplemental modules are available for study on other techniques and concepts.

Module 21) robot challenges

20.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module:

- Understand basic procedure of Wi-Fi connections
- Utilize the SimpleLink SDK to get started quickly with Wi-Fi development

g mode in CCS