TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

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 the robot to cloud services.

1. You will interface a CC3100 to the MSP432 using SPI communications
2. You will configure the combo to connect to the internet via Wi-Fi
3. You will create a place in the cloud to log data
4. Your running robot will log data onto your cloud

Good to Know: There are many possible applications you can implement once your system is connected to the internet. Your robot could receive data or commands from the internet, or your robot can send data to the internet. Combining receiving and transmitting allows you to develop a remote controller.

20.1 Getting Started

20.1.1 Software Starter Projects

Look at these projects:
 CC3100_GetWeather (fetches weather from openweathermap.org)
 Lab20_CC3100 (starter project for this lab)

20.1.2 Student Resources (in datasheets directory-Links)

CC3100.pdf (SimpleLink Wi-Fi Wireless Network Processor*)*
swru371b.pdf CC3100 BoosterPack Hardware
swru368b.pdf CC3100 SimpleLink Wi-Fi Internet-on-a-Chip

20.1.3 Reading Materials

Chapter 20, “Embedded Systems: Introduction to Robotics"

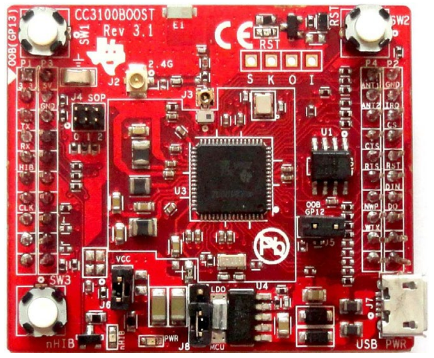


Figure 1. CC3100 Wi-Fi BoosterPack.

20.1.4 Components needed for this lab

All components needed for this lab are included in the TI-RSLK Max kit (TIRSLK-EVM), for this lab you will need to purchase the CC3100BOOST. Batteries will be needed to power your robot.

Quantity	Description	Manufacturer	Mfg P/N
1	TI-RSLK MAX kit	Texas Instruments	TIRSLK-EVM
1	SimpleLink™ Wi-Fi® CC3100 wireless network processor BoosterPack™ plug-in module	TI	CC3100BOOST

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 many 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. **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. In this lab, the MSP432/CC3100 device will run in station mode, and the router will run in AP mode. The CC3100 can also run in **Peer-to-peer mode** (P2P), allowing it to connect directly to other nearby devices, without using an access point.

Lab: Wi-Fi

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. 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 two software libraries available from TI. These libraries were designed for general use. In other words, they were not designed specifically for the TI-RSLK MAX curriculum

driverlib, MSP432 SimpleLink SDK, I/O on the MSP432
cc3100-sdk, implement SimpleLink Wi-Fi using the CC3100

Please notice the difference in software style between TI-RSLK MAX software and the rest of TI software. TI-RSLK MAX is an educational platform. Software within the TI-RSLK MAX curriculum was designed to solve very simple and specific problems. E.g., activate PWM output on P2.6 with a fixed frequency of 100 Hz and variable duty cycle. The goal was to create software solutions so simple, that students could understand everything. On the other hand, commercial software like **driverlib** and **cc3100-sdk** were designed to solve all problems. The goals were functionality, reliability, code reuse, legacy, stability, evolution, abstraction, and portability. When using commercial software, one rarely looks at the .c source files to see how they work (in fact, many commercial products don't give you .c source files, just compiled libraries), but rather, one looks at the .h header files or software documentation to see what the software does and how to use it. As a user of commercial software one is most interested in time to project completion, reliability, and functionality. One is interested in long-term profits, which will require code reuse, evolution, stability, abstraction, and portability. As a student of the art of embedded systems, one is most interested in understanding, which requires exposing how it works and seeing the engineering tradeoffs. Code reuse, stability, evolution, and portability are not usually important a student engineer. With TI-RSLK our goal was to make it so simple you understand it all.

20.3 Experiment set-up

The original installer for TI-RSLK MAX included a copy of the **driverlib** and **cc3100-sdk** libraries into your workspace. These two libraries will allow you to complete Lab 20. Note that the TI documentation references the CC31XXEMUBOOST for updating firmware, but **you will not need CC31XXEMUBOOST to complete the activities in TI-RSLK**. You will use the CC3100 with the preloaded software, as described in **swru371b.pdf** and swru368b.pdf.

Warning: Please ensure the +5V jumper on the MSP432 LaunchPad is disconnected or removed. Not removing this jumper will cause permanent damage to the LaunchPad and the TI-RSLK chassis board.

The SimpleLink Wi-Fi SDK Plugin is designed for development on the CC3100 Network Processor and MSP432 Host MCU. The CC3100 and MSP432 could communicate over SPI or UART, but we will use SPI. The CC3100 requires an external host MCU for the user application.
First, we will set up the hardware. You will implement this lab using the MSP432 LaunchPad and the CC3100 Wi-Fi BoosterPack. You do not need to disconnect your MSP432 from the TI-RSLK MAX robot. However, you cannot run both the **CC2650 and the CC3100 at the same time.** In other words, please remove any BoosterPacks from previous modules.

1. Mount the CC3100 BoosterPack on top of a MSP432 LaunchPad so the pins align so the silkscreens on both boards are facing the same direction. Be careful not to bend any pins.

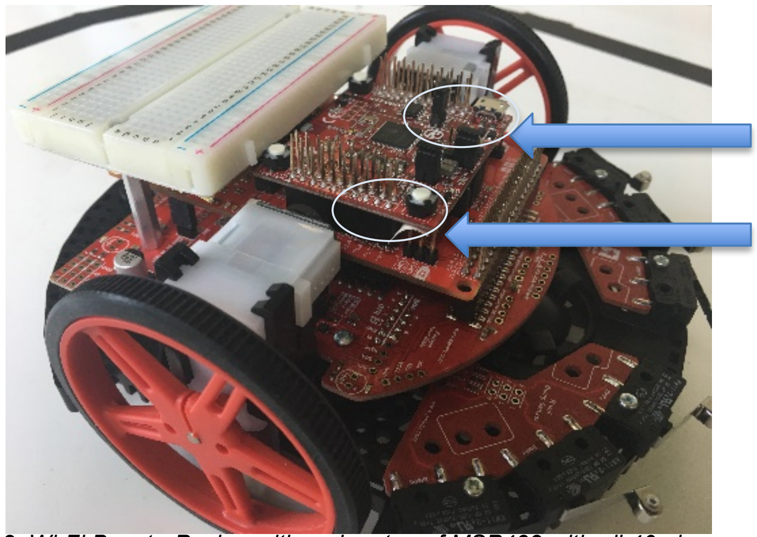


Figure 2. Wi-Fi BoosterPack positioned on top of MSP432 with all 40 pins aligned. Notice we placed insulating tape between the MSP432 LaunchPad and the CC3100 BoosterPack because the three pins on each side are very close.

2. You will power the system in the usual manner. You should not plug in to the USB connector on the BoosterPack, this USB is used for

reprogramming the CC3100, and but not needed in this lab. You just want one USB cable to the MSP432 LaunchPad like the previous labs.

The pins used for CC3100 do not overlap with any of the TI-RSLK MAX functions. There are six pins connecting the MSP432 and the CC3100

P2.5 IRQ input causes edge interrupts, CC3100 -> MSP432
P3.0 SPI_CS (GPIO output, active low), MSP432 -> CC3100
P1.5 SPI_CLK (UCB0, 12 MHz), MSP432 -> CC3100
P1.6 SPI_MOSI (UCB0), MSP432 -> CC3100
P1.7 SPI_MISO (UCB0), CC3100 -> MSP432
P4.1 nHIB (GPIO output to hibernate), MSP432 -> CC3100

Six additional pins are allocated for the CC3100 but not used. Even though these pins do not participate in the MSP432-CC3100 communication you will not be able to use these pins for other TI-RSLK MAX functions because they may be driven by the CC3100, or they may cause action in CC3100 if you drive them:

P5.1 WLAN_LOG_TX UART log data arrives from CC3100 into MSP432,
P2.3 NWP_LOG_TX UART log data arrives into MSP432
P3.3 UART1_RX UART from MSP432 to CC3100 with weak pullup
P3.2 UART1_TX UART from CC3100 into MSP432
P5.6 UART1_CTS from MSP432 to CC3100
P6.6 UART1_RTS from CC3100 to MSP432

Good to Know: For security reasons, you may consider creating a new and separate Google account with its own email and googledocs. This way, your robot need not have access to any actual information you have on Google.

20.4 System Development Plan

20.4.1 Get weather

In order to test the Wi-Fi functionality we will use the MSP432-CC3100 combination to fetch weather from the internet, using the **CC3100_GetWeather** project. Run a terminal program like TExaSdisplay so you can observe the interactions between the MSP432 and CC3100. The first step is to determine the SSID and password for your Wi-Fi access point or hot spot that you will use to provide access to the internet. You will edit the **sl_common.h** file with this access point information. You will find this file in \cc3100-sdk\examples\common. In particular, you will edit the following three lines with specifications of your access point.

```
#define SSID_NAME        "your_ap_name"
#define SEC_TYPE         SL_SEC_TYPE_WPA
#define PASSKEY          "your_password"
```

The system will operate with three possible security types

OPEN (**SL_SEC_TYPE_OPEN**),
WPA/WPA2 (**SL_SEC_TYPE_WPA)**, or
WEP (**SL_SEC_TYPE_WEP**)

The second step is to create a user account on **openweathermap.org**, with which you can fetch weather information from the server.

1) go to **http://openweathermap.org/appid#use**
2) Register on the Sign up page
3) get an API key (APPID)

You will replace the APPID=**1234567890abcdef1234567890abcdef** with your APPID. In particular, you should edit the line that begins with

```
#define REQUEST "GET /data/2.5/weather?q=Austin&APPID=
```

As an option, you can change the city. Refer to the **openweathermap.org** site for other options for this request. Run a terminal program like **TExaSdisplay** and open the appropriate COM port (115200 baud). Build and debug this project. After connecting to the hot spot it will fetch the weather and output the response to the COM port. The following is a typical response. This is the TCP packet from MSP432 to openweathermap.org, where **<X>** will be your APPID

```
GET /data/2.5/weather?q=Austin&APPID=<X>&units=metric
HTTP/1.1
Host:api.openweathermap.org
Accept: */*
```

The following is a typical TCP packet response from the server, where **<X>** will be your APPID.

```
HTTP/1.1 200 OK
Server: openresty
Date: Fri, 21 Jun 2019 17:21:21 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 465
Connection: keep-alive
X-Cache-Key:
/data/2.5/weather?APPID=<X>&q=austin&units=metric
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST
```

```
{"coord":{"lon":-
97.74,"lat":30.27},"weather":[{"id":803,"main":"Clouds","de
scription":"broken
clouds","icon":"04d"}],"base":"stations","main":{"temp":31.
87,"pressure":1012,"humidity":59,"temp_min":30.56,"temp_max
":33.33},"visibility":16093,"wind":{"speed":3.6,"deg":170},
"clouds":{"all":75},"dt":1561137681,"sys":{"type":1,"id":33
44,"message":0.0104,"country":"US","sunrise":1561116583,"su
nset":1561167340},"timezone":-
18000,"id":4671654,"name":"Austin","cod":200}
```

Use this project to verify your MSP432, CC3100, and hot spot are operational.

20.4.2 Send an email with IFTTT

There are many options for create services in the cloud with which your robot could interact. One easy-to-use cloud service called **If This Then That**. IFTTT is a website that lets us set up rules and triggers to automate a process. For example, we will configure the robot so that if we push a button on the LaunchPad, it will send us an email.

1. Sign up for an IFTTT account at **ifttt.com** and associate it with an email address for testing your LaunchPad. Again, for security reasons, feel free to create a new google account and email just for this robot. Create a new applet from the interface. An applet is a logical connection between two web services supported in IFTTT.

2. Click New Applet" and choose a service by clicking "this" highlighted in blue and pointed to by the arrow.

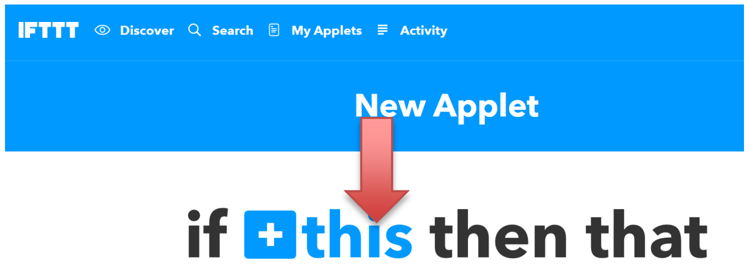


Figure 3. IFTTT new applet this.

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

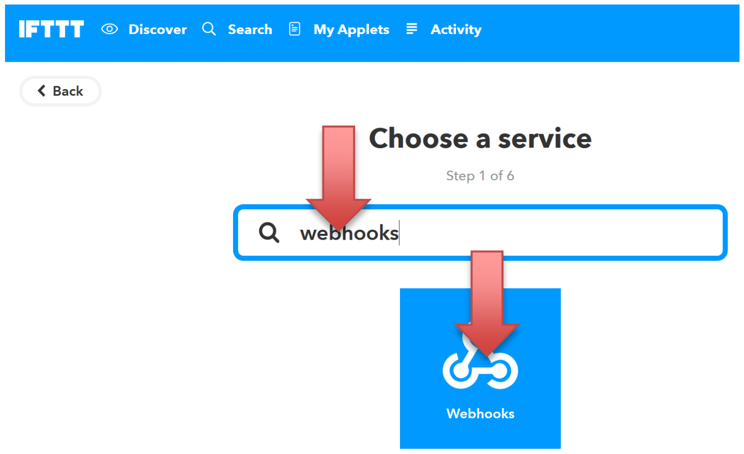
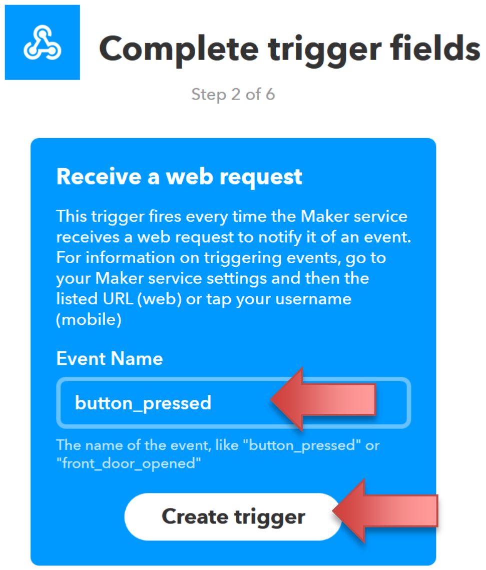


Figure 4. IFTTT choose service.

4. Click "Receive a web request" and call the event name "button_pressed". Click "Create trigger".



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 5. IFTTT complete trigger fields.

5. Specify an action by clicking “that” highlighted in blue.

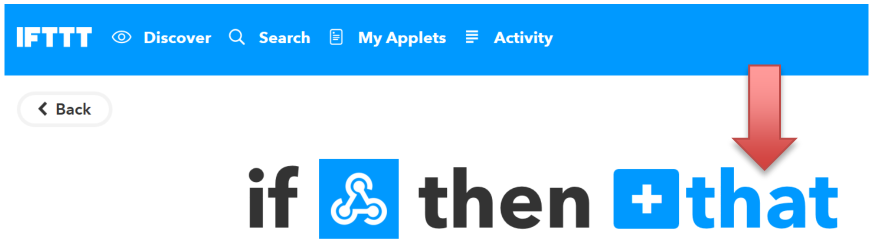


Figure 6. IFTTT new applet that.

6. Choose the action service by typing “email” and select email. You will have to connect your email to IFTTT by receiving a PIN via the email and typing it back into IFTTT.

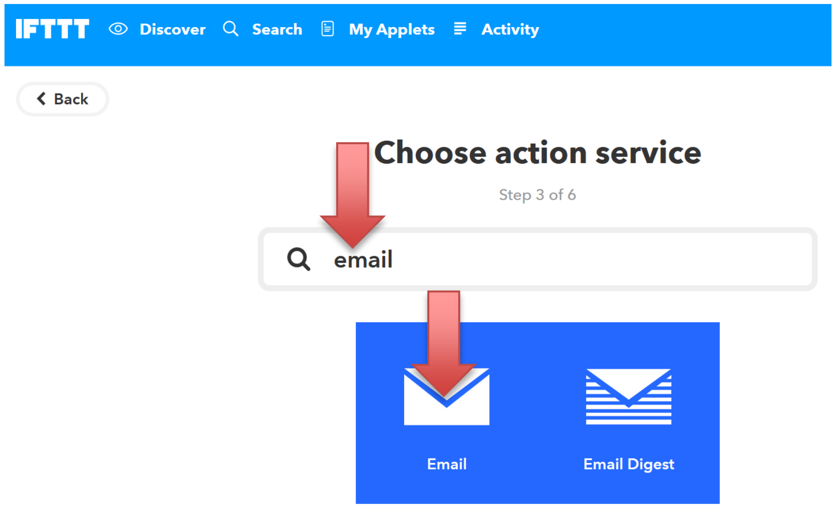
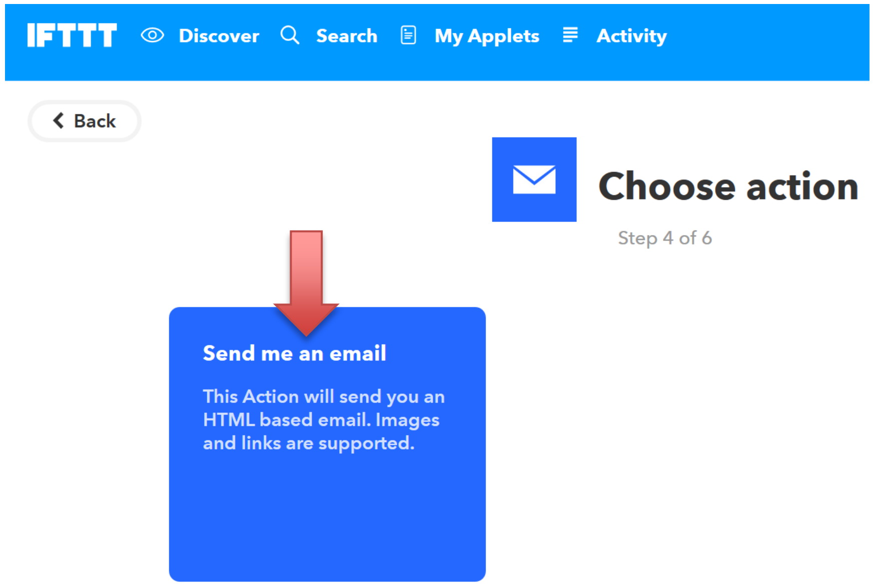


Figure 7. IFTTT choose action service.

7. Choose an action “send me an email”.



Choose action

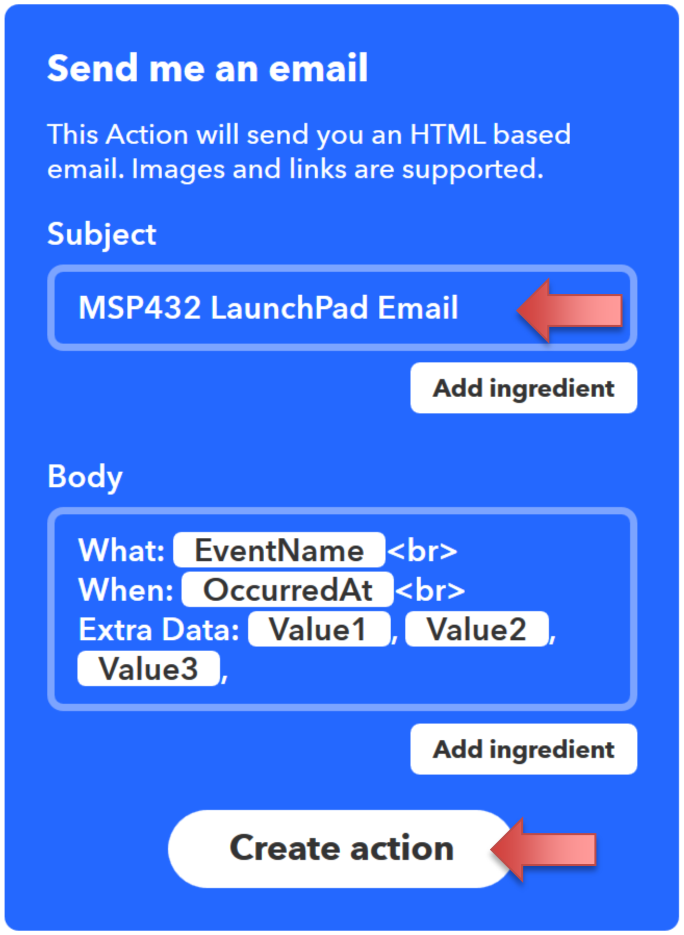
Step 4 of 6

Send me an email

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

Figure 8. IFTTT choose action email.

8. Set your subject to “MSP432 LaunchPad Email” and leave the body with the default values. **Value1 Value2** and **Value3** will be data fields you could use to send information about the robot in the email body.



Send me an email

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

Subject

MSP432 LaunchPad Email

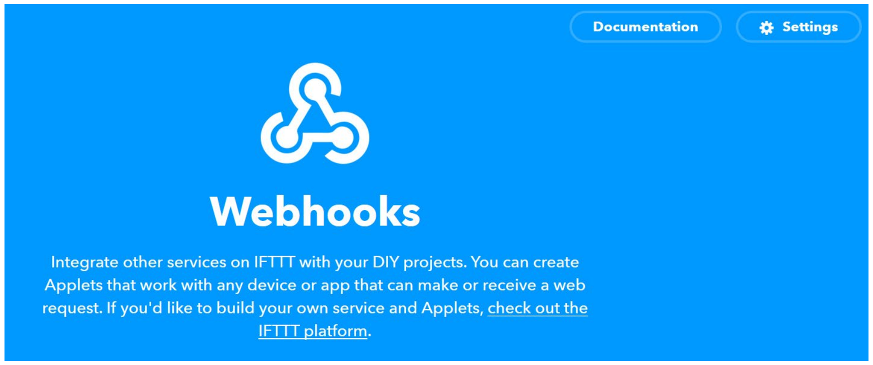
Body

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

Figure 9. IFTTT action setup

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

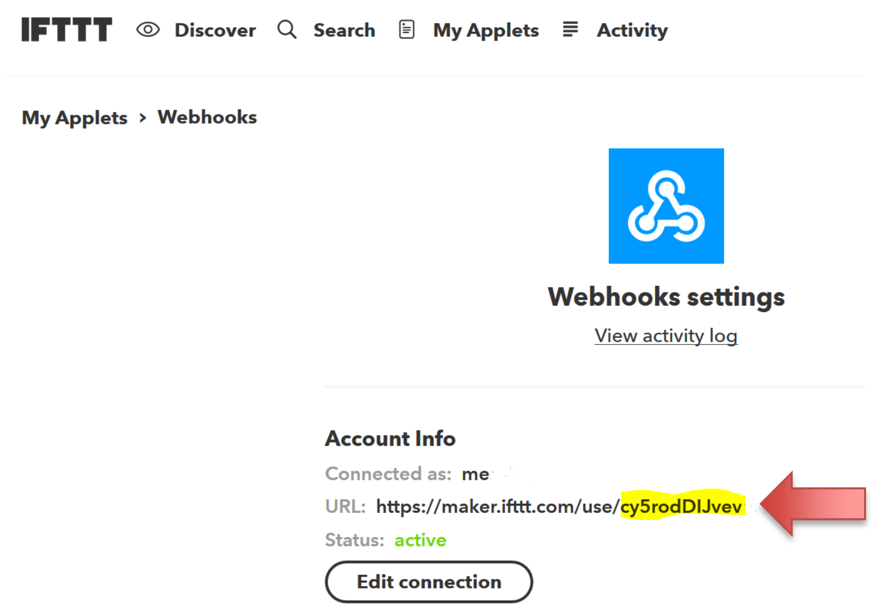
10. Now we need to go into the settings of the Webhooks service. Go into the settings from my applets menu.



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 10. 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}. Save a copy of this key.



Account Info

Connected as: me

URL: https://maker.ifttt.com/use/cy5rodDIJvev

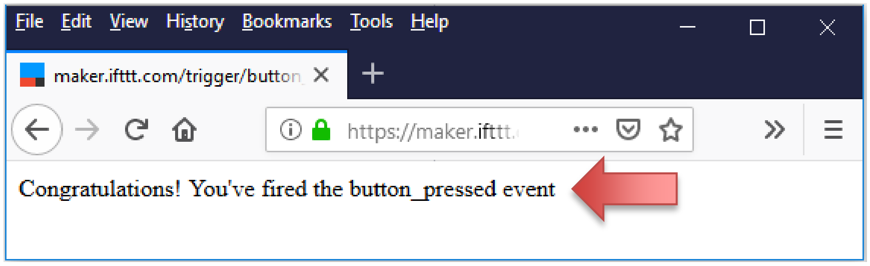
Status: active

Figure 11. IFTTT webhooks account information with key highlighted in yellow.

11. If you go to this URL, it will trigger a Webhooks evebt and send you an email.

https://maker.ifttt.com/trigger/button_pressed/with/key/{key}

where {key} is your key. For the robot to send an email through IFTTT the robot will need send a POST TCP packet to IFTTT. You can try it out in your web browser first to verify the email sends correctly. In other words, if you enter the above URL into a browser and type enter, IFTTT will send you an email and show a response similar to Figure 12.


File Edit View History Bookmarks Tools Help
maker.ifttt.com/trigger/button
https://maker.ifttt.

Congratulations! You've fired the button_pressed event

Figure 12. IFTTT web page response to triggering an event with Webhooks.

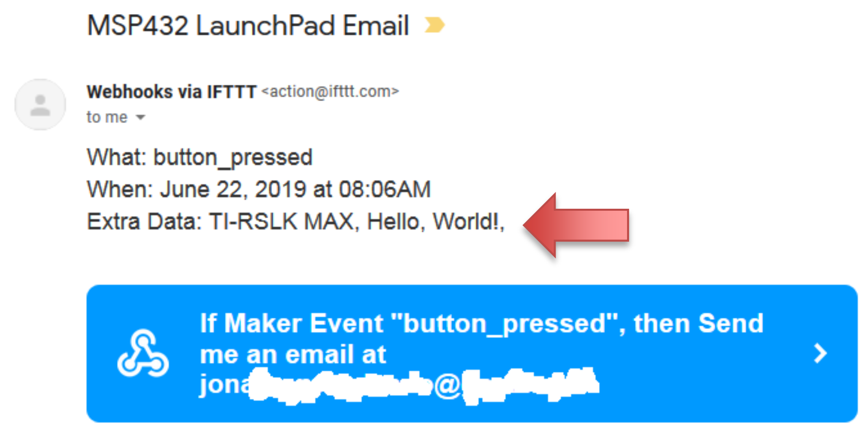
12. Now you can test this feature with CCS code running on your robot. You can easily convert the **CC3100_GetWeather** project to post a message to IFTTT, triggering a Webhooks event. Comment out the three lines for getting weather

```
//#define WELCOME "\nFetching weather from openweathermap.org"
//#define WEBPAGE "api.openweathermap.org"
//#define REQUEST "GET /data/2.5/weather?q=Austin&APPID= 123456
```

and uncomment the corresponding three lines for IFTTT

```
#define WELCOME "\nIFTTT trigger email"
#define WEBPAGE "maker.ifttt.com"
#define REQUEST "POST /trigger/button_pressed/with/key/123
```

You will replace the **1234567890abcdef1234567890abcdef** with your IFTTT key. Run this code to verify it will send an email from your RSLK robot. You will get an email similar to Figure 13.



MSP432 LaunchPad Email

Webhooks via IFTTT <action@ifttt.com>
to me

What: button_pressed
When: June 22, 2019 at 08:06AM
Extra Data: TI-RSLK MAX, Hello, World!,

If Maker Event "button_pressed", then Send me an email at jona[redacted]@[redacted]

Figure 13. Typical email received from IFTTT.

20.4.3 Log sensor data

The overall goal of this lab is to combine the features you have developed in previous labs, so that the robot performs some complex task. For example, the robot could follow a line, solve a maze, or race around a track. The starter code for Lab 20 will operate the robot to move in a square using odometry from the tachometers. Feel free to establish any complex task for your robot. Periodically the robot should log data onto a server. This provides real-time debugging as your robot solves a challenge.

Create another applet on your IFTTT account. Similar to the email example, this applet will be triggered by a Webhooks event. Give it a different name from button_pressed, such as log_data. However, the action event will be to add a row to a spreadsheet on your GoogleDrive.

Within your Lab20 code, you will pass robot data in the three fields **Value1 Value2** and **Value3.** For example, the starter code logs the robot position as x, y, and theta. In particular, you will need to populate the fields with debugging information as appropriate for your task. However, since there are limits to how often you can trigger Webhooks events, you should consider aggregating multiple recordings while running the robot, and then send large log packets to IFTTT. For the starter code, a typical TCP packet looks like this with nine data recordings aggregated into one log packet.

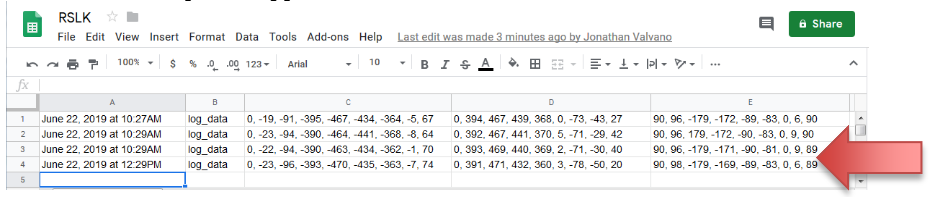
```
POST /trigger/log_data/with/key/<X> HTTP/1.1
Host: maker.ifttt.com
User-Agent: CCS/9.0.1
Connection: close
Content-Type: application/json
Content-Length: 165

{"value1" : "0, -23, -96, -393, -470, -435, -363, -7, 74",
"value2" : "0, 391, 471, 432, 360, 3, -78, -50, 20",
"value3" : "90, 98, -179, -169, -89, -83, 0, 6, 89" }
```

The IFTTT server response should be something like this

```
HTTP/1.1 200 OK
Date: Sat, 22 Jun 2019 17:29:36 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 48
Connection: close
X-Top-SecreTTT: <X>
Server: web_server
Congratulations! You've fired the log_data event
```

And the recording on GoggleSheets is shown as the last row



RSLK
File Edit View Insert Format Data Tools Add-ons Help Last edit was made 3 minutes ago by Jonathan Valvano

	A	B	C	D	E
1	June 22, 2019 at 10:27AM	log_data	0, -19, -91, -395, -467, -434, -364, -5, 67	0, 394, 467, 439, 368, 0, -73, -43, 27	90, 96, -179, -172, -89, -83, 0, 6, 90
2	June 22, 2019 at 10:29AM	log_data	0, -23, -94, -390, -464, -441, -368, -8, 64	0, 392, 467, 441, 370, 5, -71, -29, 42	90, 96, 179, -172, -90, -83, 0, 9, 90
3	June 22, 2019 at 10:29AM	log_data	0, -22, -94, -390, -463, -434, -362, -1, 70	0, 393, 469, 440, 369, 2, -71, -30, 40	90, 96, -179, -171, -90, -81, 0, 9, 89
4	June 22, 2019 at 12:29PM	log_data	0, -23, -96, -393, -470, -435, -363, -7, 74	0, 391, 471, 432, 360, 3, -78, -50, 20	90, 98, -179, -169, -89, -83, 0, 6, 89
5					

20.5 Troubleshooting

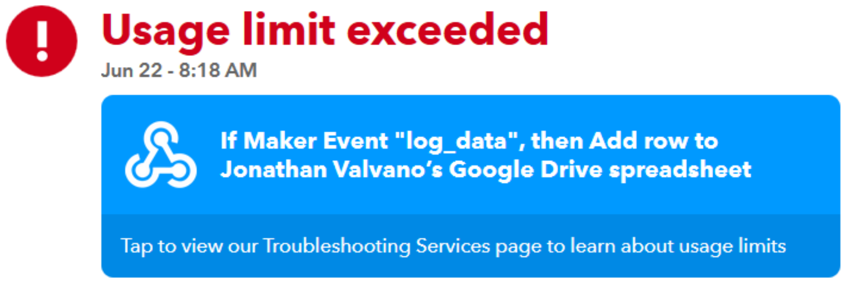
The Wi-Fi can't connect to the router:

- 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. Check the voltages on the robot and the MSP432.

- 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 CC3100 is not able to know what to do with that.

Cloud issues:

- Make sure the router has an internet connection
- Check your activity log on IFTTT to make sure you have not exceeded your usage limit. There is a limit to both the number and rate of Webhooks you can trigger.


Usage limit exceeded
Jun 22 - 8:18 AM
If Maker Event "log_data", then Add row to Jonathan Valvano's Google Drive spreadsheet
Tap to view our Troubleshooting Services page to learn about usage limits

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. The RSLK projects used CC3100 SDK v1.3.0 release. If you plan to use the MSP432/CC3100 combination for projects beyond RSLK, you could download and install the newest versions of the drivers on the TI web site

SimpleLink MSP432P4 SDK http://www.ti.com/tool/SIMPLELINK-MSP432-SDK
SimpleLink CC3100 SDK http://www.ti.com/tool/CC3100SDK

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.

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

ti.com/rslk