

Step One: Host Your “Things” in the Cloud

For this project, I used Arduino Create IoT Cloud (<https://create.arduino.cc/iot/things/>) to host five esp8266 devices. The Maker Plan lets me host five “Things” and utilize one type of 3rd party board. My recommendation is to create just one Thing to start. Setting up all 5 at once was a great learning experience, but very confusing at times.

CREATE - MAKER PLAN

\$ 6.99/MONTH

FEATURES RECAP

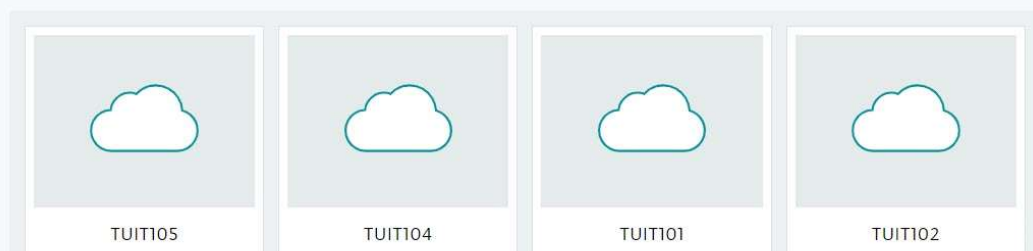
250 Total Sketches	5 for each Arduino board type
200MB Storage	Cloud-enabled Arduino Boards
unlimited Compilation Time	1 Cloud-enabled 3rd Party Boards
5 Things	Custom Library editing
20 Properties	Chrome App Access
15 days Cloud Data Retention	ESP8266 3rd Party Boards support
Cloud API	on Web Editor
3 Cloud-enabled Linux Devices	



YOUR THINGS

Arduino IoT Cloud allows you to connect devices to the internet and to other devices. This tool makes the creation of connected objects quick, simple and secure. [Read more](#) or check out our [step-by-step tutorial](#). You might also look at our [FAQ](#).

ADD NEW THING



New Dashboards

Are you looking for the dashboards you have created with the new dashboard tool?

GO TO DASHBOARDS

Step Two: Add Properties to Your Thing

Two properties have been added in this example: An Int TYPE called “Counter” and a Character String called “TUIT” (change this name to anything you like).

TUIT105

EDIT SKETCH

Last synced 20 hours ago

Properties

Dashboard

Webhooks

Board

ADD PROPERTY

NAME	TYPE	UPDATE	PERMISSION		
Counter	Int	On change	R&W		
TUIT	Character String	On change	RO		

Thing ID:

Step Three: Setup an IoT Board

- 1) Go to <https://create.arduino.cc/devices/> to “be able to attach your board to an interactive widget in the [Arduino Create IoT Cloud](#).”

DEVICE MANAGER

ARDUINO DEVICE MANAGER

Below are the boards and devices you have added to Arduino Create. If you want to set up a new device, go to the Getting Started section of Create.

search device

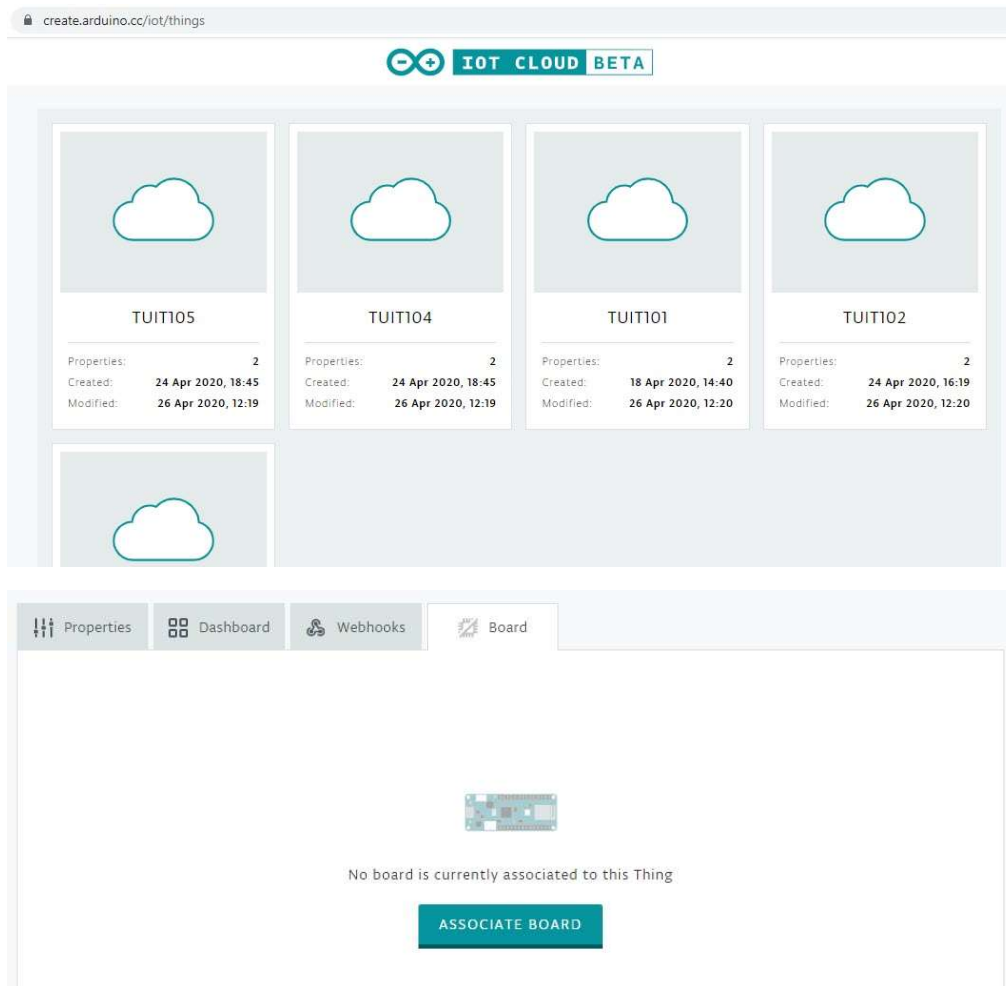
Q

MY ARDUINO BOARDS

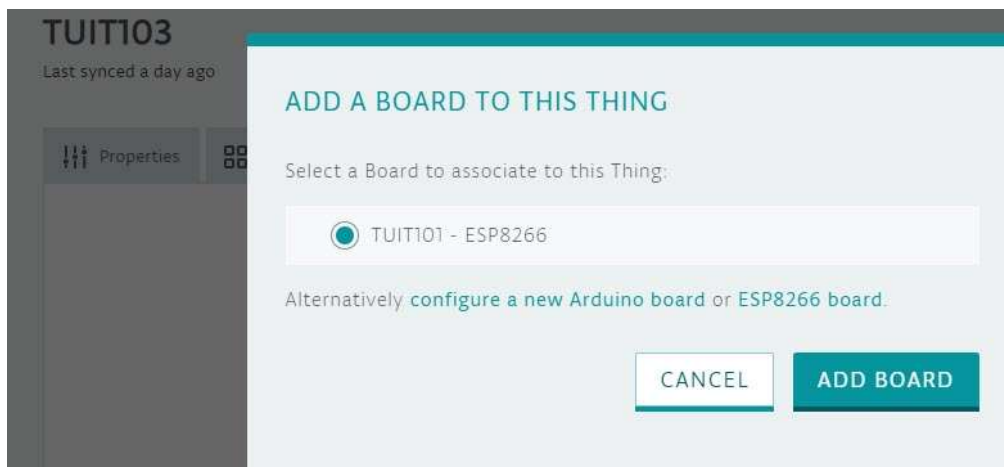
If you've set up a MKR1000, MKR WiFi 1010, MKR GSM 1400, MKR NB 1500, or a Nano 33 IoT you can set up a dashboard in Arduino IoT Cloud.

If you've set up a MKR WAN 1300 or a MKR WAN 1310 board, you can access a dashboard to check information about it (e.g., AppEUI and AppKey), view messages sent from it and send messages to it. You can also attach webhooks that will be triggered when a message is received. An Arduino Pro Gateway LoRa® Connectivity is required to communicate with the MKR WAN 1300.

- 2) Go back to your IoT Dashboard and select your Thing. Click the Board Tab and the Associate Button.

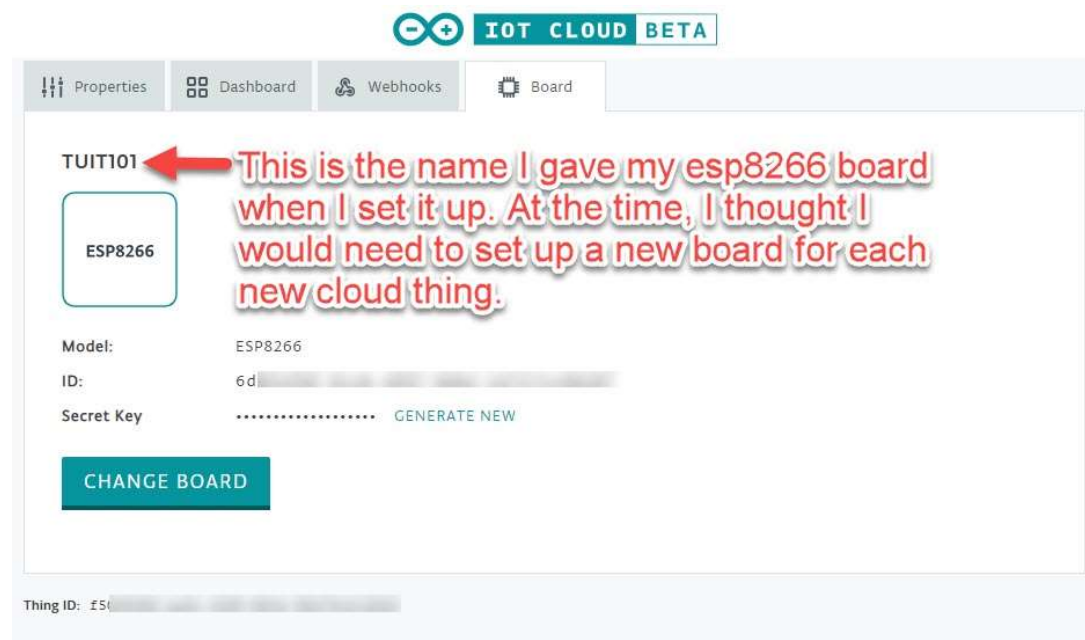
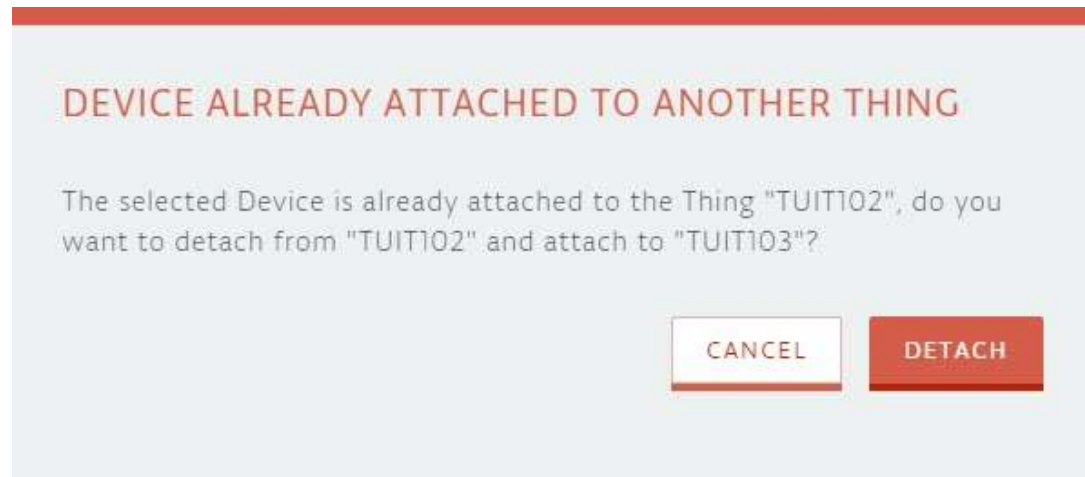


- 3) Hopefully, your Board is now listed here and ready to be connected to your Thing.



Note: If your board is showing up as already attached to another Thing, you can detach it and associate it with the Thing you are currently working on. This is the only way I was able to use the same cloud-enabled Arduino board with multiple Things.

Be careful not to change the Secret Code associated with the board at this time. This will become clearer in Step Five when you tweak your Arduino code.



4) Click Edit Sketch.



Step Four: Tweak and Upload Your Code

This step is a beautiful thing. Arduino Create automatically creates Arduino code for the Thing you have connected to the IoT Cloud (which includes the properties you set up during Step Two). Just a few tweaks, and you are ready to upload your first code.

```
TUIT103.ino  ReadMe.adoc  thingProperties.h  Secret  ▼

/*
Sketch generated by the Arduino IoT Cloud Thing "TUIT103"
https://create.arduino.cc/cloud/things/f50...

Arduino IoT Cloud Properties description
The following variables are automatically generated and updated when changes are made to the Thing properties

String tUIT;
int counter;

Properties which are marked as READ/WRITE in the Cloud Thing will also have functions
which are called when their values are changed from the Dashboard.
These functions are generated with the Thing and added at the end of this sketch.
*/

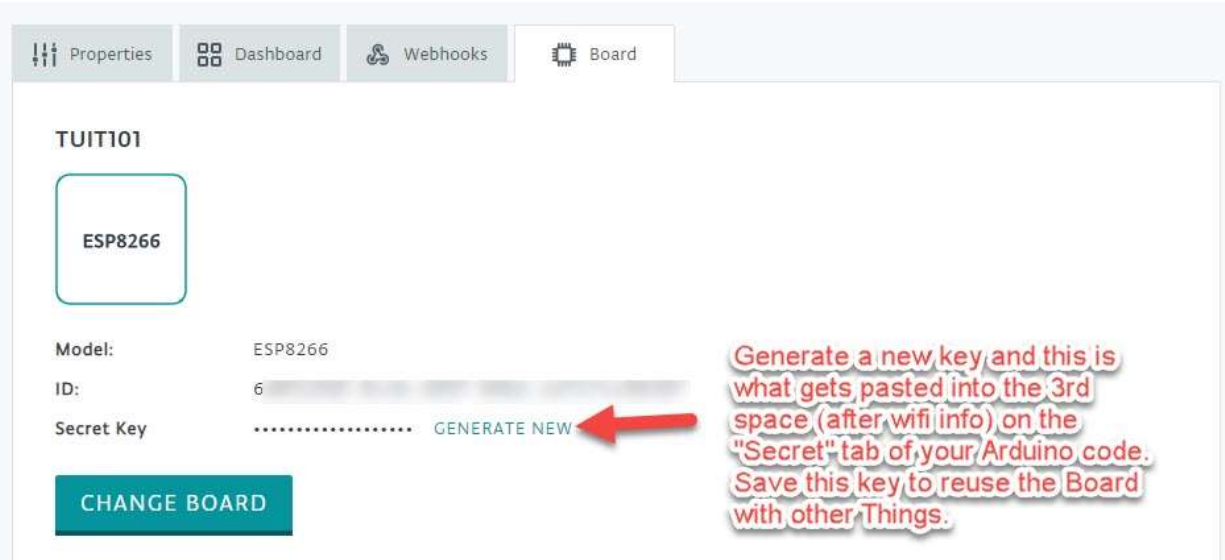
#include "thingProperties.h"
```

- 1) Secret Tab... load it with your wifi info (SSID and password) and the Secret Device Key of your Board. You can have one peek at this. Then keep it stored safely. (If you want to connect more Things to the same type of board, you will need to reuse this key.) If you did not store this key the first time you saw it, change it and store it safely.

```
TUIT103.ino  ReadMe.adoc  thingProperties.h  Secret  ▼
```

> WHAT IS THE SECRET TAB?

SECRET_SSID	<input type="text"/>
SECRET_PASS	<input type="password"/>
SECRET_DEVICE_KEY	<input type="password"/>



- 2) Switch to the thingProperties.h tab. Change String tUIT (or whatever you named your String in Step Two) to String tUIT = "TUIT 103", or whatever you want to call it.

```
TUIT103.ino  ReadMe.adoc  thingProperties.h  Secret  ▼
4
5 const char THING_ID[] = "f5
6 const char DEVICE_LOGIN_NAME[] = "6d
7
8 const char SSID[] = SECRET_SSID; // Network SSID (name)
9 const char PASS[] = SECRET_PASS; // Network password (use for WPA, or use as key for WEP)
10 const char DEVICE_KEY[] = SECRET_DEVICE_KEY; // Secret device password
11
12 void onCounterChange();
13
14 String tUIT;
15 int counter;
16
17 void initProperties(){
18
19   ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
20   ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
21   ArduinoCloud.setThingId(THING_ID);
22   ArduinoCloud.addProperty(tUIT, READ, ON_CHANGE, NULL);
23   ArduinoCloud.addProperty(counter, READWRITE, ON_CHANGE, onCounterChange);
24
25 }
26
27 WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
28
```

```
TUIT103.ino  ReadMe.adoc  thingProperties.h  Secret  ▼
3
4
5 const char THING_ID[]      = "f5( [REDACTED] 3";
6 const char DEVICE_LOGIN_NAME[] = "6d( [REDACTED] 7";
7
8 const char SSID[]          = SECRET_SSID;    // Network SSID (name)
9 const char PASS[]          = SECRET_PASS;    // Network password (use for WPA, or use as key for WEP)
10 const char DEVICE_KEY[]    = SECRET_DEVICE_KEY; // Secret device password
11
12 String TUIT_Name = "TUIT103";
13 String tUIT(TUIT_Name);
14
15 void onCounterChange();
16
17 int counter;
18
19 void initProperties(){
20
21   ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
22   ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
23   ArduinoCloud.setThingId(THING_ID);
24   ArduinoCloud.addProperty(tUIT, READ, ON_CHANGE, NULL);
25   ArduinoCloud.addProperty(counter, READWRITE, ON_CHANGE, onCounterChange);
26
27 }
```

3) Finally, go to the Arduino code tab, where you can add your own code. Here is the code I used after thingProperties.h. (Sample code is attached to this Github Branch.)

```
17 #include "thingProperties.h"
18
19 //String TUIT_Name = "TUIT103"; //copy these lines to thingProperties
20 //String tUIT(TUIT_Name);
21
22 int blue_led = 2; //onboard blue led
23 int redPin = 15;
24 int greenPin = 12;
25 int bluePin = 13;
26 int sensorPin = A0; // select the input pin for ldr
27
28 const int buttonPin = 4; //the number of the pushbutton
29
30
31 // Remote site information
32 const char http_site[] = "l[REDACTED]";
33 const int http_port = 80;
34
35 // Global variables
36 WiFiClient client;
37
```



```

38 void setup() {
39   // Initialize serial and wait for port to open:
40   Serial.begin(9600);
41   // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
42   delay(1500);
43
44   // Defined in thingProperties.h
45   initProperties();
46
47   // Connect to Arduino IoT Cloud
48   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
49
50   /*
51    * The following function allows you to obtain more information
52    * related to the state of network and IoT Cloud connection and errors
53    * the higher number the more granular information you'll get.
54    * The default is 0 (only errors).
55    * Maximum is 1
56    */
57   setDebugMessageLevel(2);
58   ArduinoCloud.printDebugInfo();
59

```

```

60   // Attempt to connect to website
61   if ( !getPage() ) {
62     Serial.println("GET request failed");
63   }
64   //just a little LED action to let me know the wifi connected
65   digitalWrite(greenPin, HIGH); // turn the LED on (HIGH is the voltage level)
66   delay(500); // wait for a second
67   digitalWrite(greenPin, LOW); // turn the LED off by making the voltage LOW
68   delay(500); // wait for a second
69   digitalWrite(greenPin, HIGH); // turn the LED on (HIGH is the voltage level)
70   delay(500); // wait for a second
71   digitalWrite(greenPin, LOW); // turn the LED off by making the voltage LOW
72   delay(500); // wait for a second
73   digitalWrite(greenPin, HIGH); // turn the LED on (HIGH is the voltage level)
74   delay(500); // wait for a second
75   digitalWrite(greenPin, LOW); // turn the LED off by making the voltage LOW
76   delay(500); // wait for a second
77 }
78

```

```

79 void loop() {
80   ArduinoCloud.update();
81   // Your code here
82   pinMode(blue_led, OUTPUT);
83   pinMode(buttonPin, INPUT);
84
85   digitalWrite(blue_led, HIGH); //off
86
87   // read the input pin:
88   int buttonState = digitalRead(buttonPin);
89
90   if (buttonState == LOW) {
91     counter = counter + 1;
92     digitalWrite(blue_led, LOW); //on
93     if (counter > 10) {
94       getPage();
95       counter = 1;
96     }

```



```

97 Serial.println();
98 Serial.print(tUIT);
99 Serial.print(" has ");
100 Serial.print(counter);
101 Serial.print(" points.");
102 Serial.println();
103
104 delay(1000);           // wait for a second
105 }
106 }
107
108 void onCounterChange() {
109
110 }
111
112 // Perform an HTTP GET request to a remote page
113 bool getPage() {
114
115     // Attempt to make a connection to the remote server
116     if ( !client.connect(http_site, http_port) ) {
117         return false;
118     }
119

```

```

119
120     // Make an HTTP GET request
121
122     client.println("GET /tuitapp/.....php HTTP/1.0");
123     client.print("Host: ");
124     client.println(http_site);
125     client.println("Connection: close");
126     client.println();
127
128     return true;
129 }
130
131 void onTUITChange() {
132     // Do something
133 }
134

```

Step Five: Post Twilio PHP Code to Website

Check out <https://www.twilio.com/docs/sms/quickstart/php> to learn how to use PHP to receive and respond to texts sent to your Twilio Phone Number.

Post the PHP code to your website. Insert your Twilio account SID Auth Token and the 'from' Twilio phone number. Customize the phone number you would like to send a message to and the content of your message in the 'body'.

Make note of the url path to the php page, as you will need to update this in your Arduino sketch.

```
<?php
// Require the bundled autoload file - the path may need to change
// based on where you downloaded and unzipped the SDK
require __DIR__ . '/twilio-php-master/Twilio/autoload.php';

// Use the REST API Client to make requests to the Twilio REST API
use Twilio\Rest\Client;

// Your Account SID and Auth Token from twilio.com/console
$sid = 'AC[REDACTED]';
$token = '[REDACTED]';
$client = new Client($sid, $token);

// Use the client to do fun stuff like send text messages!
$client->messages->create(
    // the number you'd like to send the message to
    '+[REDACTED]',
    array(
        // A Twilio phone number you purchased at twilio.com/console
        'from' => '+1[REDACTED]',
        // the body of the text message you'd like to send
        'body' => "Hey Paula! You reached 10 TUIT Points!"
    )
);
?>
```

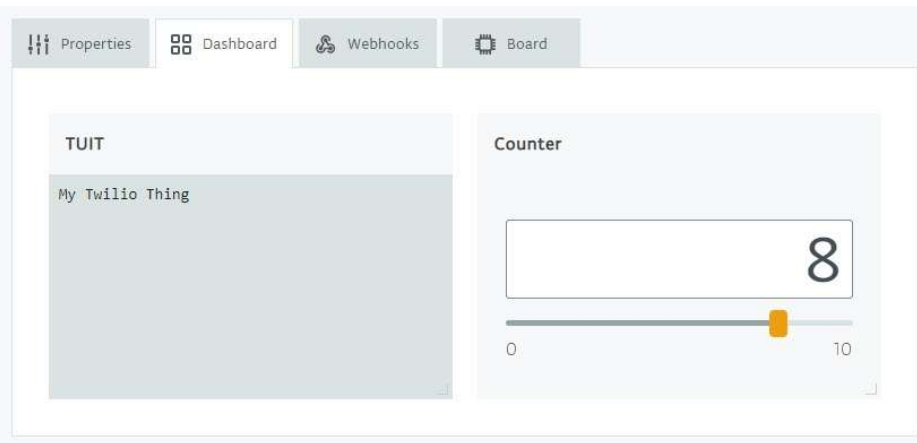
```
31 // Remote site information
32 const char http_site[] = "your_website_url.com"; //you can also hook up to a secure site
33 const int http_port = 80;
34

120 // Make an HTTP GET request
121
122 client.println("GET /yourpath/to/yourtwilio.php HTTP/1.0");
123 client.print("Host: ");
124 client.println(http_site);
125 client.println("Connection: close");
126 client.println();
127
```

Step Six: Upload Your Code

If Arduino Create is set up correctly on your computer, it should be picking up your board and serial port. (If not, you probably need to install the plug-in. You can find instructions at <https://software.intel.com/en-us/arduino-create-for-intel-based-platforms-getting-started-guide-set-up-plugin>.)

Once your code is uploaded, you should be able to press the button on your Thing and have the counter counting up from 1 to 10. You can watch this happen in the Thing Dashboard.



And now, drumroll, please..... when you reach 10, you will get a text message on your phone! So cool.



Next Steps: So Many Possibilities!!!

- Post incoming data to a Google Sheet using a Webhook (I already have this working, but need to post the instructions.)
- Create a web leaderboard out of the data (See www.learncode.events for my start on this project.)
- Send a message to someone (or yourself) when you reach a goal (Yay, I won!)
- Display incoming messages on an L.E.D. matrix or change the colour of an L.E.D. to say, "Hello World!"
- Attach your Thing to a push handle hand sanitizer to count how many times it gets used. (This is going to be my weekend geek out project! Stay tuned!)

Stay tuned for more tutorials from
The L.E.D. Lady!