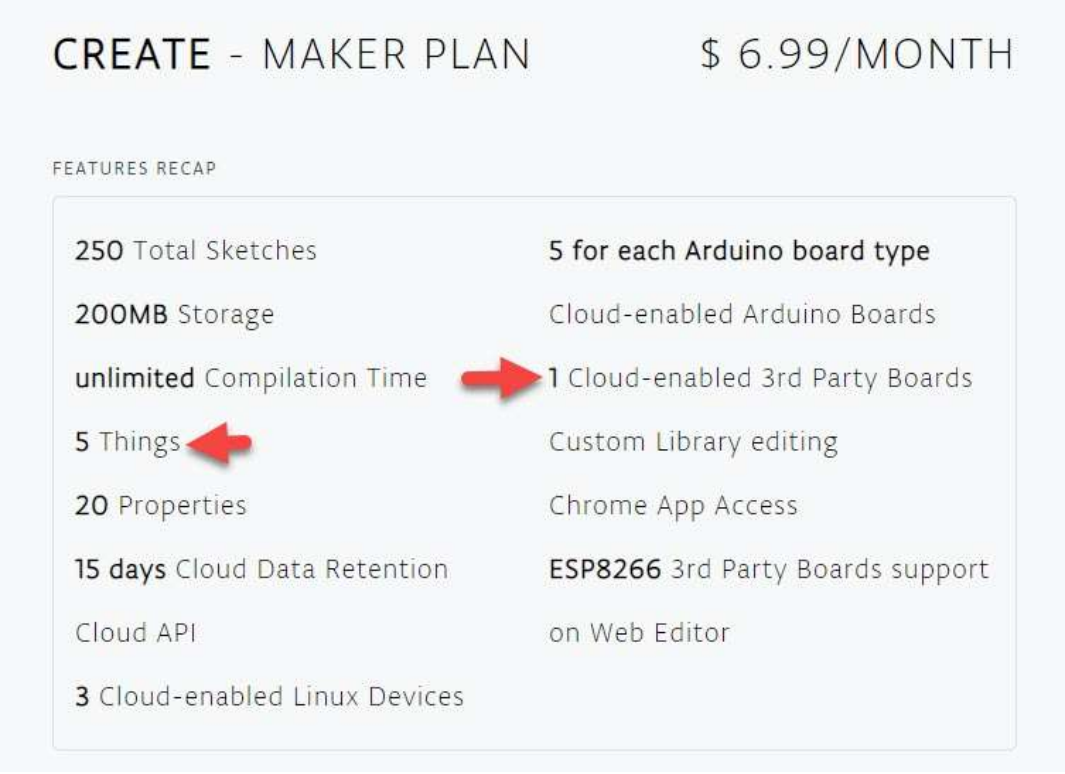
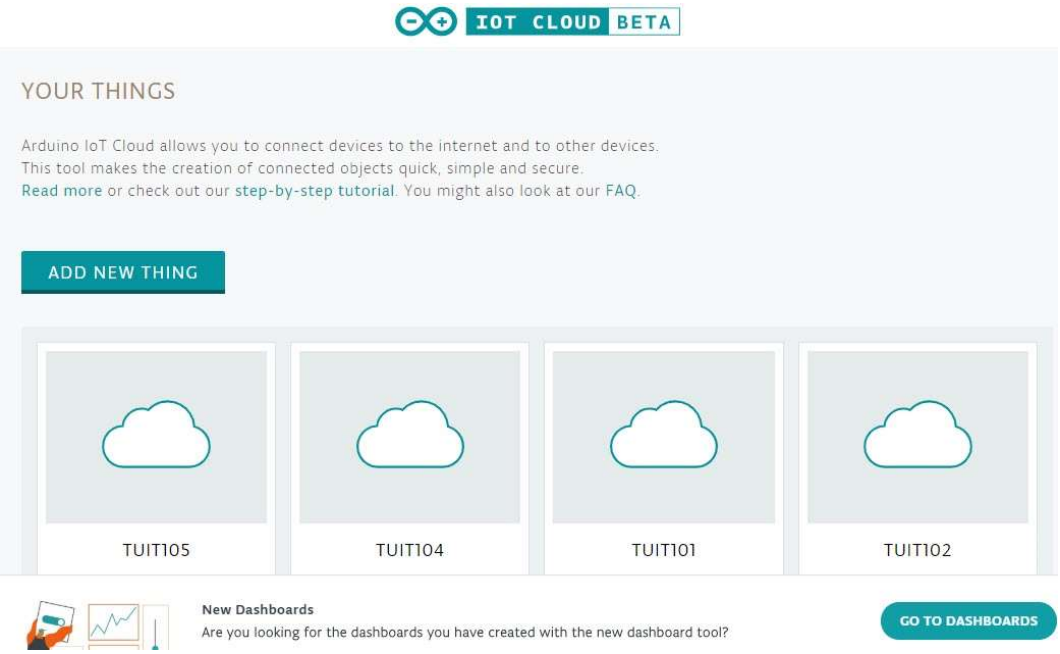
Step One: Host Your “Things” in the Cloud

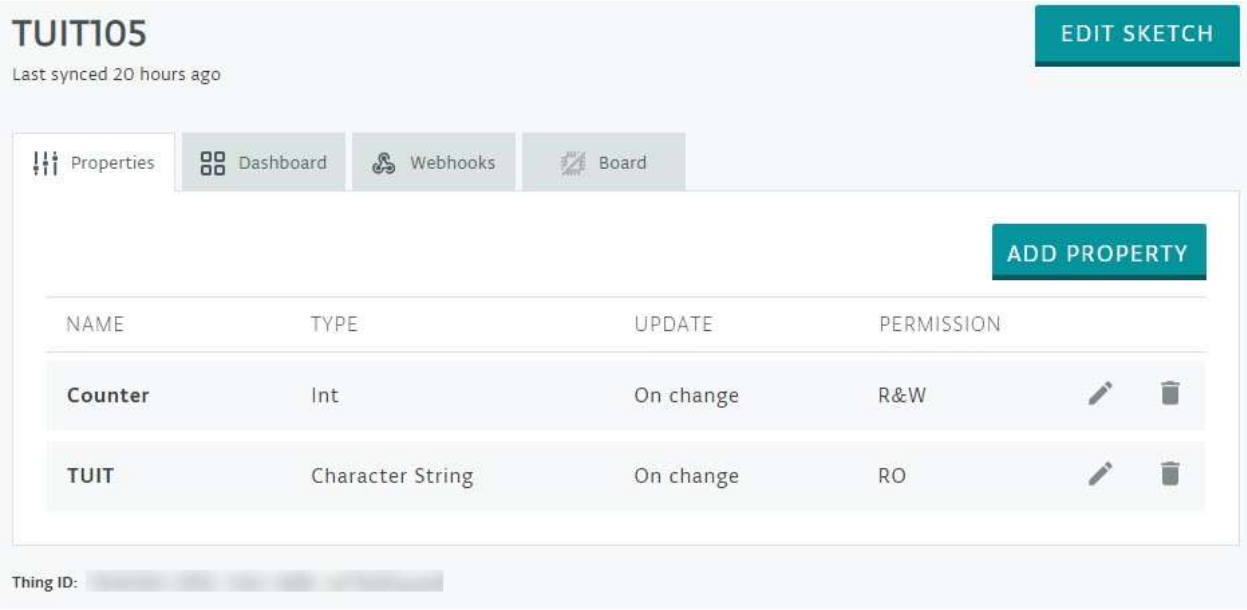
For this project, I used Arduino Create IoT Cloud (https://create.arduino.cc/iot/things/) to host five esp8266 devices. The Maker Plan lets me host five “Things” and utilize one type of 3rd party board. My recommendation is to create just one Thing to start. Setting up all 5 at once was a great learning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





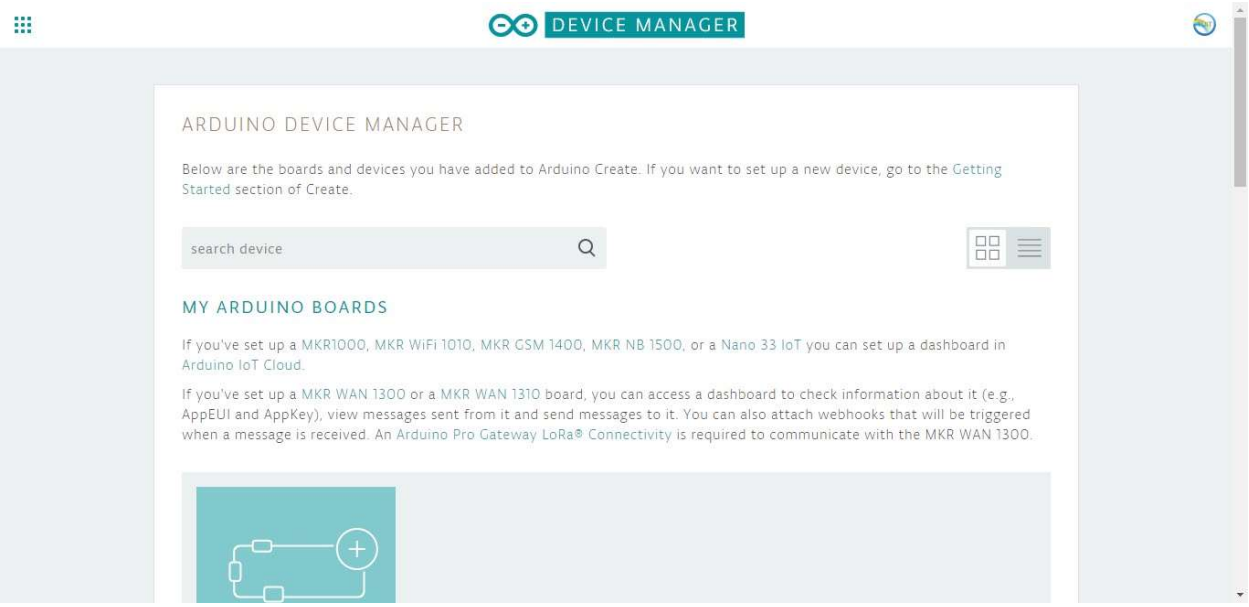
Step Two: Add Properties to Your Thing

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).

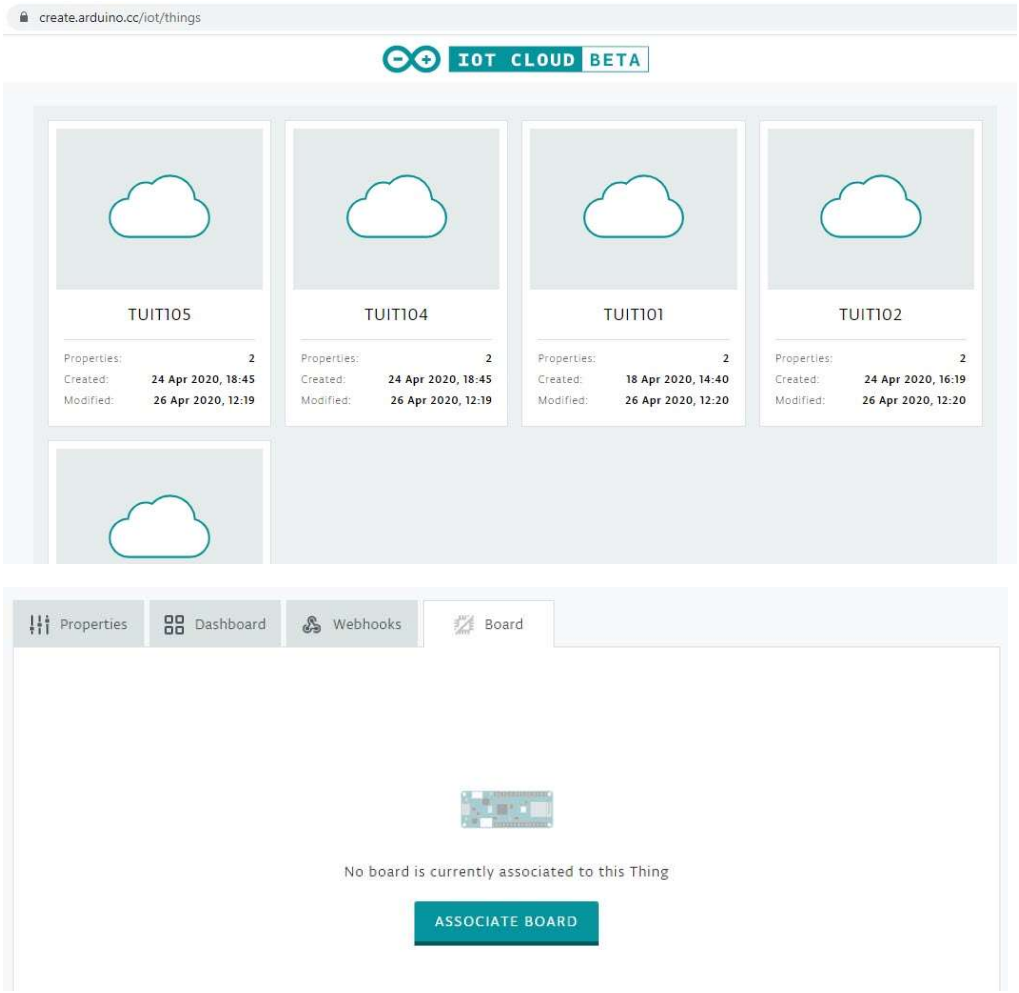


Step Three: Setup an IoT Board

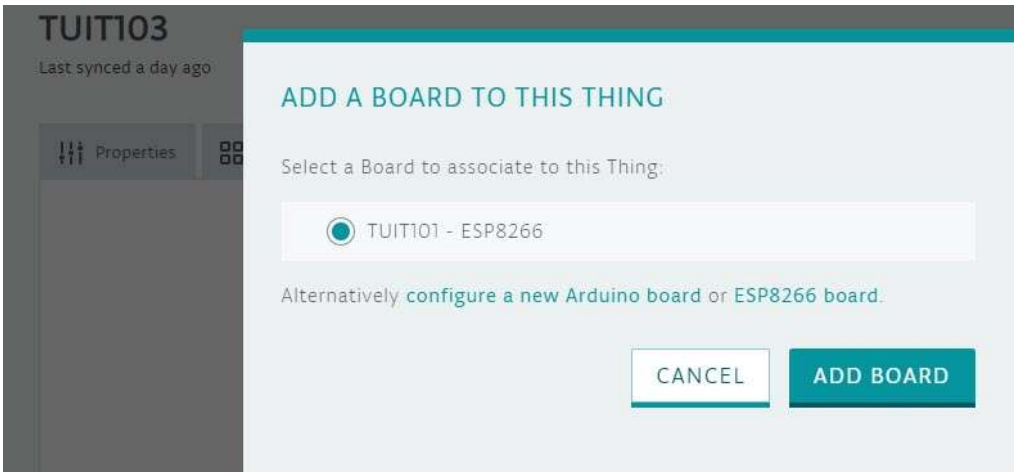
1) Go to https://create.arduino.cc/devices/ to “be able to attach your board to an interactive widget in the Arduino Create IoT Cloud.”



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.



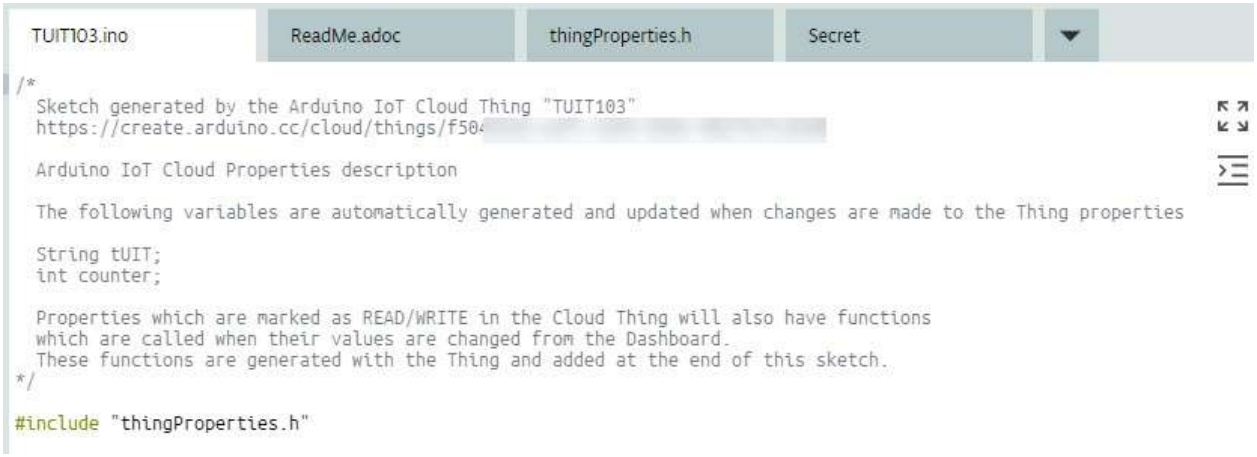
Note: If your board is showing up as already attached to another Thing, you can detach it and associate it with the Thing you are currently working on. This is the only way I was able use the same cloud-enabled Arduino board with multiple Things.

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.

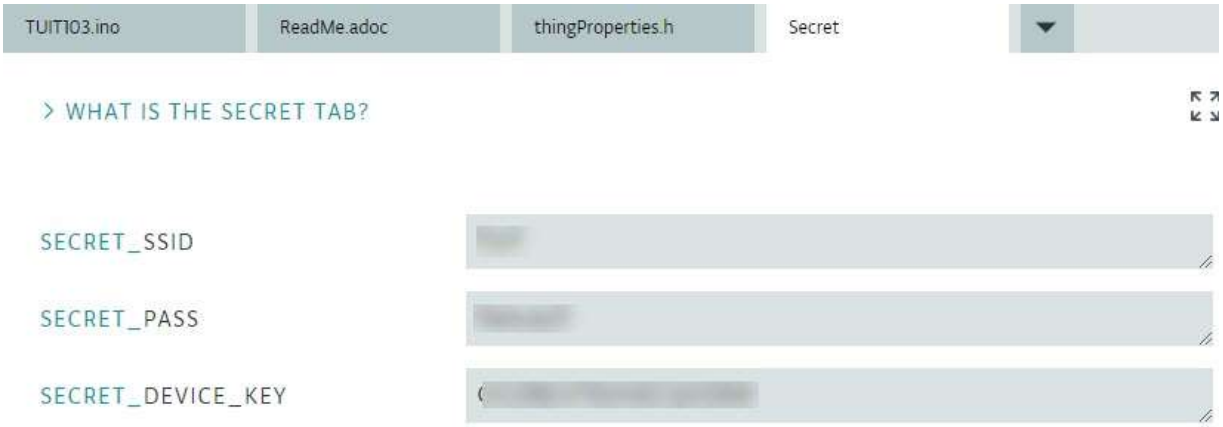
4) Click Edit Sketch.

Step Four: Tweak and Upload Your Code

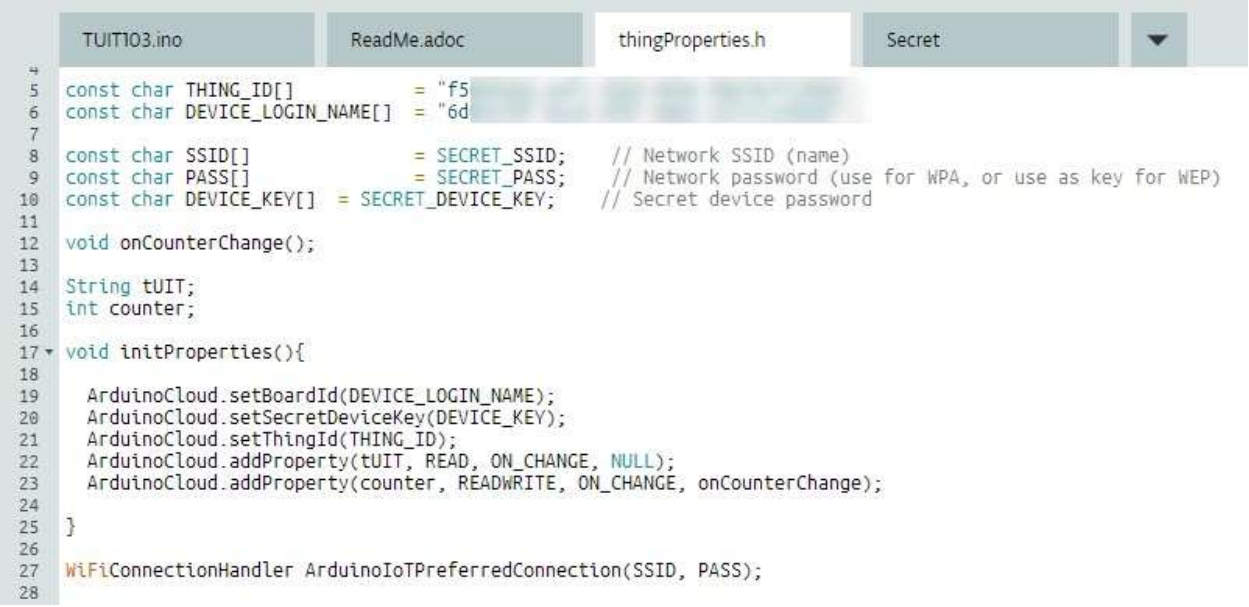
This step is a beautiful thing. Arduino Create automatically creates Arduino code for the Thing you have connected to the IoT Cloud (which includes the properties you set up during Step Two). Just a few tweaks, and you are ready to upload your first code.



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.



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

```
const char THING_ID[]           = "f5[redacted]3";
const char DEVICE_LOGIN_NAME[]  = "6d[redacted]7";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_PASS;    // Network password (use for WPA, or use as key for WEP)
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

String TUIT_Name = "TUIT103";
String tUIT(TUIT_Name);

void onCounterChange();

int counter;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(tUIT, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(counter, READWRITE, ON_CHANGE, onCounterChange);

}
```

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.)

```
#include "thingProperties.h"

//String TUIT_Name = "TUIT103"; //copy these lines to thingProperties
//String tUIT(TUIT_Name);

int blue_led = 2; //onboard blue led
int redPin = 15;
int greenPin = 12;
int bluePin = 13;
int sensorPin = A0; // select the input pin for ldr

const int buttonPin = 4; //the number of the pushbutton


// Remote site information
const char http_site[] = "l[redacted]";
const int http_port = 80;

// Global variables
WiFiClient client;

```

```
void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 1
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

```

```
  // Attempt to connect to website
  if ( !getPage() ) {
    Serial.println("GET request failed");
  }
  //just a little LED action to let me know the wifi connected
  digitalWrite(greenPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                     // wait for a second
  digitalWrite(greenPin, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                     // wait for a second
    digitalWrite(greenPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                     // wait for a second
  digitalWrite(greenPin, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                     // wait for a second
    digitalWrite(greenPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                     // wait for a second
  digitalWrite(greenPin, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                     // wait for a second
}

```

```
void loop() {
  ArduinoCloud.update();
  // Your code here
  pinMode(blue_led, OUTPUT);
  pinMode(buttonPin, INPUT);

  digitalWrite(blue_led, HIGH); //off

  // read the input pin:
  int buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
  counter = counter + 1;
  digitalWrite(blue_led, LOW); //on
  if (counter > 10)  {
    getPage();
    counter = 1;
  }
```

```
  Serial.println();
  Serial.print(tUIT);
  Serial.print(" has ");
  Serial.print(counter);
  Serial.print(" points.");
  Serial.println();

  delay(1000);               // wait for a second
  }
}

void onCounterChange() {

}

// Perform an HTTP GET request to a remote page
bool getPage() {

  // Attempt to make a connection to the remote server
  if ( !client.connect(http_site, http_port) ) {
    return false;
  }
```

```

  // Make an HTTP GET request

  client.println("GET /tuitapp/[redacted].php HTTP/1.0");
  client.print("Host: ");
  client.println(http_site);
  client.println("Connection: close");
  client.println();

  return true;
}

void onTUITChange() {
  // Do something
}

```

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
<?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                                ';
$token = '                                ';
$client = new Client($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
    '+1          ',
    array(
        // A Twilio phone number you purchased at twilio.com/console
        'from' => '+1          ',
        // the body of the text message you'd like to send
        'body' => "Hey Paula! You reached 10 TUIT Points!"
    )
);
?>
```

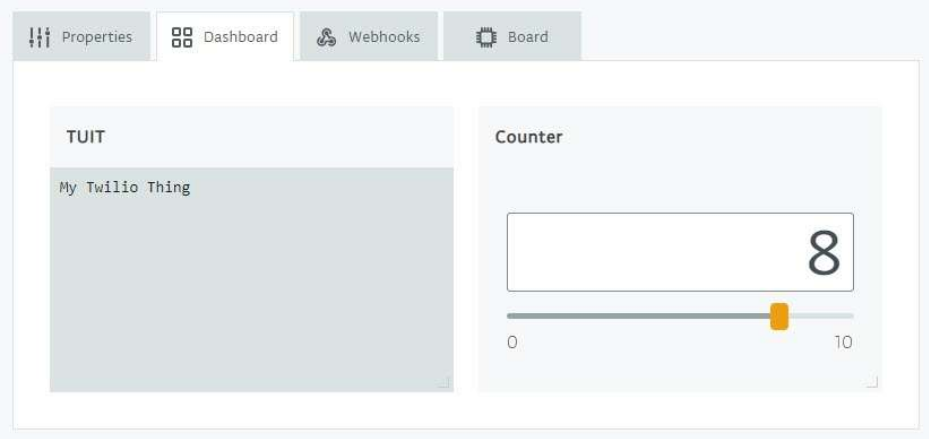
```
31  // Remote site information
32  const char http_site[] = "your_website_url.com"; //you can also hook up to a secure site
33  const int http_port = 80;
34

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!