Sending an SMS with Temboo

Created by Vaughn Shinall

https://learn.adafruit.com/sending-an-sms-with-temboo

Last updated on 2023-08-29 02:39:24 PM EDT
Table of Contents

Overview 3
Get Set Up 3
Generate Your Sketch 7
Upload and Run 9
Push to Send 11
Wiring Your Circuit 11
Adding Code 12
Overview

The Temboo service has moved away from Arduino support. The service access in this older guide will no longer work.

This guide will show you the basic framework for programming your Arduino to interact with APIs using the Temboo platform. We'll focus on sending an SMS with Twilio here, but the same basic steps will work for any of the different processes in the Temboo library.

Get Set Up

To build this example project, you'll be using an internet-connected Arduino board that will send a text message. We've chosen to use:
An Arduino Uno R3

An Adafruit CC3000 WiFi Shield
A USB A - B cable

A push button
A 10k Ω resistor

A half-size breadboard
Some wires

You'll also need a Temboo account and a Twilio account. To set up your Temboo account (if you don't already have one) you can create one for free [here](#). You can create a free Twilio account [here](#).

Finally, make sure that you have the latest version of Temboo’s Arduino library installed; if you don't, you can download it [here](#).

---

**Generate Your Sketch**

The Temboo service has moved away from Arduino support. The service access in this older guide will no longer work.

Now that you’re all set up, log in to Temboo and go to the [Twilio > SMS Messages > SendSMS](#) Choreo in the Temboo Library. Turn on IoT Mode and make sure that you've added details about the shield that your Arduino board is using to connect to the internet. You can do this by setting IoT Mode to ON in the upper right-hand corner of the window and then selecting Arduino and Adafruit CC3000 WiFi from the dropdown menus that appear, as shown below:
Next, fill out the Input fields using the information from your Twilio account, the text of the SMS that you want to send, and the phone number to which you would like to send it. You'll find your Twilio Account SID and Auth Token by going to Twilio and checking your Dashboard, as shown below; note that you need to use the credentials found on the Dashboard, not the test credentials from the Twilio Dev Tools panel:

When using a free Twilio account, you'll need to verify the phone numbers to which messages are being sent by going to Twilio and following the instructions under the Numbers > Verified Caller IDs tab, as shown below.

Once you've filled out all of the input fields on the Temboo SendSMS Choreo page, test the Choreo from your browser by clicking Run at the bottom of the window.
If you elected to send the SMS to your own phone number, you should receive it (and if you didn't, whomever you chose will instead). You'll also see a JSON that is returned by Twilio appear under Output, indicating that, among other things, your test executed successfully.

The Temboo service has moved away from Arduino support. The service access in this older guide will no longer work.

Now that you've tested the Choreo successfully, you can scroll down to find the Code section of the page. When you're in IoT Mode and you run a Choreo, Temboo automatically generates code that can be used to make the same API call from an Arduino sketch. Copy the code, and paste it into a new sketch in the Arduino IDE.
In order to run this sketch on your Arduino, it needs to be configured with an appropriate TembooAccount.h header file that contains your Temboo account information and internet shield setup information. To create the header file, make a new tab in the Arduino IDE, and name it TembooAccount.h.

On the Twilio SendSMS Choreo page beneath the sketch code that you previously copied and pasted into the Arduino IDE, you'll find another block of generated code containing #define statements and details about your internet shield. This is your header file. Copy the contents of the header into the TembooAccount.h tab in your Arduino IDE.

With both files in place, you are ready to start sending SMS from your Arduino. Save and upload the sketch, open the serial monitor, and watch the texts roll in!
Push to Send

Of course, a device that just sends off messages with no prompting may not be of much use, so let's add a button to the Uno that will trigger an SMS when pushed.

Wiring Your Circuit

The circuit for the push button is fairly straightforward—connect one of the button's legs to the 5 volt power supply, and the other to one of the digital pins on your WiFi shield. In our diagram, we've chosen pin 8. Then, connect that same leg that is wired to pin 8 to ground through a pull down resistor.
When the button in this circuit is pressed, pin 8 will be connected to power, and will read HIGH. When the button is left open, pin 8 is connected to ground, and will read LOW.

---

**Adding Code**

The Temboo service has moved away from Arduino support. The service access in this older guide will no longer work.

You've already generated all of the code you need to send an SMS through Twilio using your Arduino, so you only need to add a couple of extra lines to your sketch to trigger those messages with your push button. First, initialize the pin that you will be reading (recall that we chose to use pin 8) by adding the following lines of code to what you already have in `void setup()`:

```c
// Initialize pin
pinMode(8, INPUT);
```

Your setup should now look something like this:

```c
void setup() {
    Serial.begin(9600);
    // For debugging, wait until the serial console is connected.
    delay(4000);
    while(!Serial);
    status_t wifiStatus = STATUS_DISCONNECTED;
    while (wifiStatus != STATUS_CONNECTED) {
        Serial.print("WiFi:");
        if (cc3k.begin()) {
            if (cc3k.connectToAP(WIFI_SSID, WPA_PASSWORD, WLAN_SEC_WPA2)) {
                wifiStatus = cc3k.getStatus();
            }
        }
        if (wifiStatus == STATUS_CONNECTED) {
            Serial.println("OK");
        } else {
            Serial.println("FAIL");
        }
        delay(5000);
    }
    // Initialize pin
    pinMode(8, INPUT);
    Serial.println("Setup complete.
");
}
```

Then, add the following two lines of code to `void loop()` to set the SendSMS Choreo to run only when the button is pressed:
You should nest the code that you already have in `void loop()` within this new conditional, so that it should now look something like this:

```c
void loop() {
  int sensorValue = digitalRead(8);
  if (sensorValue == HIGH) {
    if (numRuns <= maxRuns) {
      Serial.println("Running SendSMS - Run "+ String(numRuns++));

      TembooChoreo SendSMSChoreo(client);
      // Invoke the Temboo client
      SendSMSChoreo.begin();

      // Set Temboo account credentials
      SendSMSChoreo.setAccountName(TEMBOO_ACCOUNT);
      SendSMSChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
      SendSMSChoreo.setAppKey(TEMBOO_APP_KEY);

      // Set Choreo inputs
      String AuthTokenValue = "PLACEHOLDER";
      SendSMSChoreo.addInput("AuthToken", AuthTokenValue);
      String BodyValue = "PLACEHOLDER";
      SendSMSChoreo.addInput("Body", BodyValue);
      String ToValue = "PLACEHOLDER";
      SendSMSChoreo.addInput("To", ToValue);
      String AccountSIDValue = "PLACEHOLDER";
      SendSMSChoreo.addInput("AccountSID", AccountSIDValue);
      String FromValue = "PLACEHOLDER";
      SendSMSChoreo.addInput("From", FromValue);

      // Identify the Choreo to run
      SendSMSChoreo.setChoreo("/Library/Twilio/SMSMessages/SendSMS");

      // Run the Choreo; when results are available, print them to serial
      SendSMSChoreo.run();

      while(SendSMSChoreo.available()) {
        char c = SendSMSChoreo.read();
        Serial.print(c);
      }
      SendSMSChoreo.close();
    }
  }
  Serial.println("Waiting...");
  delay(30000); // wait 30 seconds between SendSMS calls
}
```

Don't forget to add a curly bracket at the end of `void loop()` to close out the additional conditional! Note that in the code example above, the Choreo inputs have been replaced with placeholders. The code that you generate on the Temboo website will automatically have these input values filled in, so there's no need to worry about replacing them manually.
With that, your sketch is once again ready to go—you're free to tinker with it as you wish (for example, you might choose to comment out the 30 second delay at the end if you want to send messages more rapidly), but no further alterations are required. Save it, upload it to your Arduino, open the serial monitor, and push the button to dispatch your SMS!