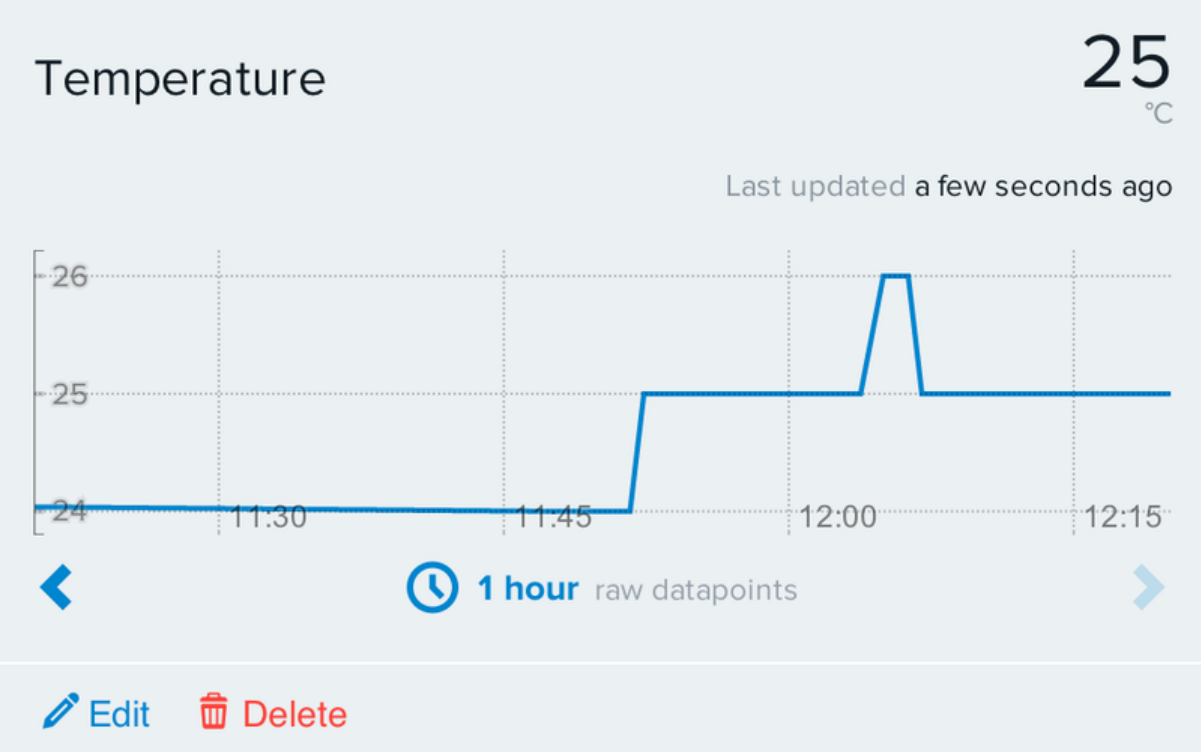
Adafruit CC3000 WiFi and Xively

Created by Marc-Olivier Schwartz



https://learn.adafruit.com/adafruit-cc3000-wifi-and-xively

Last updated on 2024-06-03 01:24:05 PM EDT

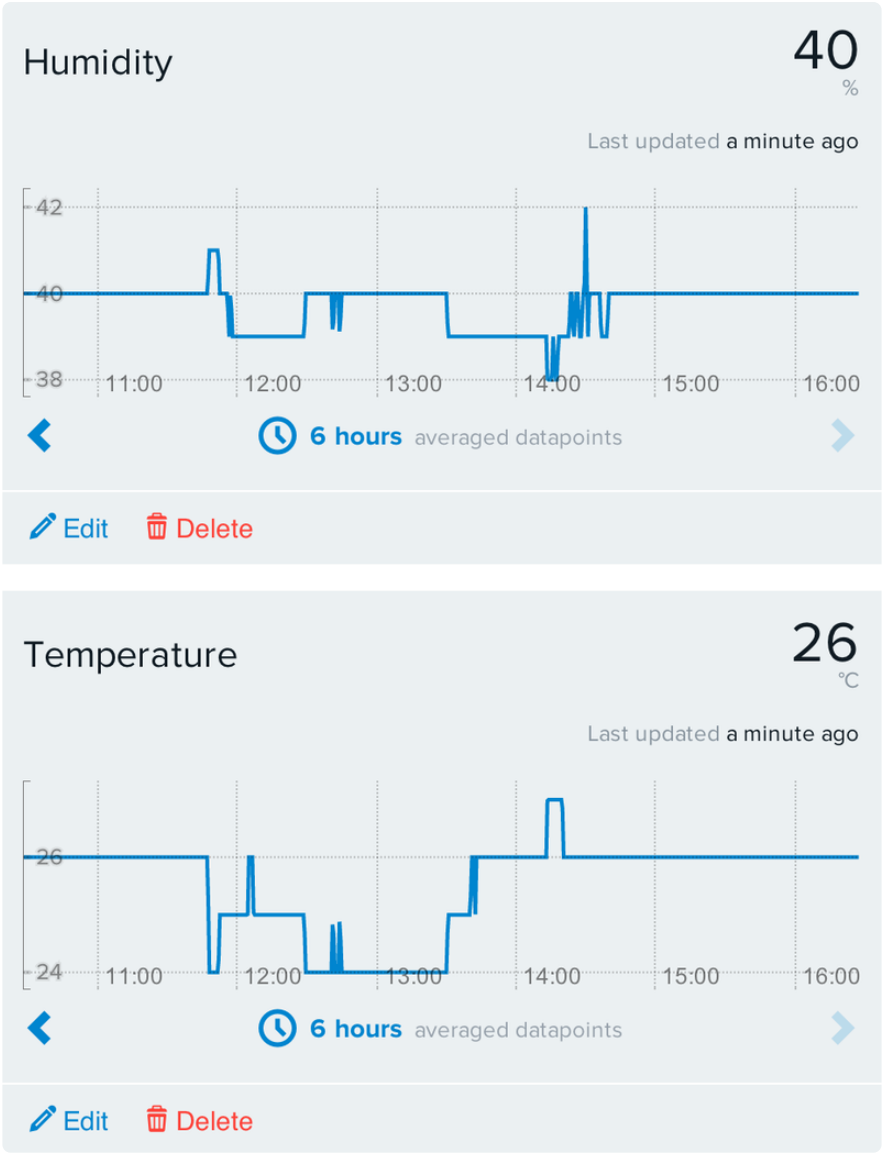
Table of Contents

Introduction

Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

The CC3000 WiFi chip from Texas Instrument is a quite versatile chip that can be used to connect your projects to the web. However, connecting your Arduino project to a web server can be tricky: you need to know how to install & configure a web server, and know a bit about HTML & PHP. Luckily, there are other solutions to make things easier.

In this guide, we are going to see how to connect a temperature & humidity sensor to an online platform for connected objects, Xively. The sensor will be connected to an Arduino Uno board, which will also communicate with the Adafruit CC3000 breakout board for the WiFi connectivity. But instead of communicating with a local server, the CC3000 chip will communicate directly with the Xively server and send the data over there. At the end, you will be able to monitor the data sent by the server directly from your browser, wherever you are in the world, just by logging into the Xively website.

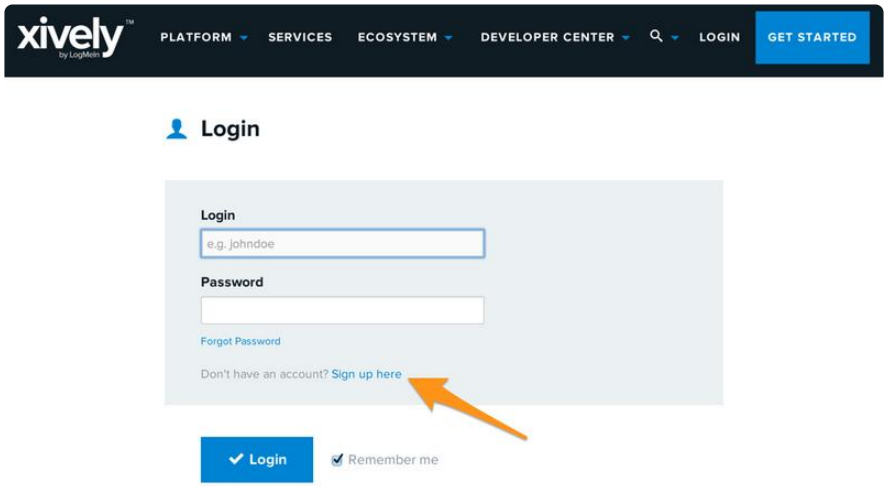


Setting up your Xively account

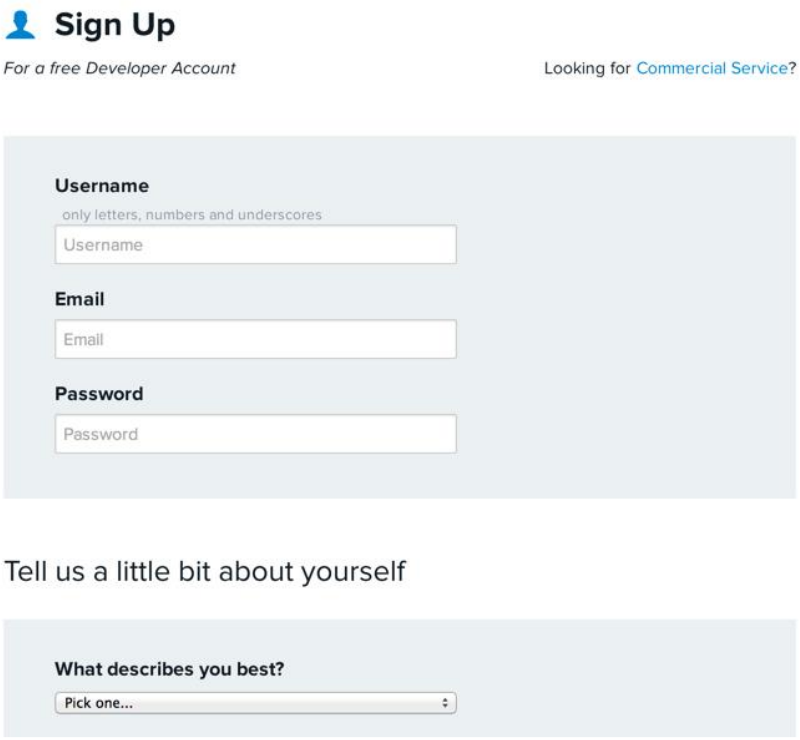
Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

The first step is to create your Xively account, and to configure it for this project. It is all done via their graphical interface, so there is no need to code anything in this part. First, you need to create an account. Just click on "Get started":

And then on "Sign up here":

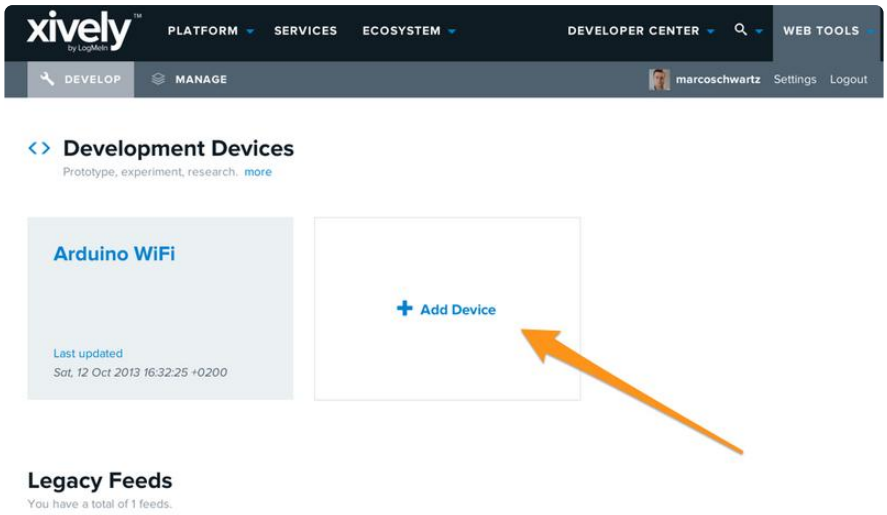


You can now enter your personal informations:

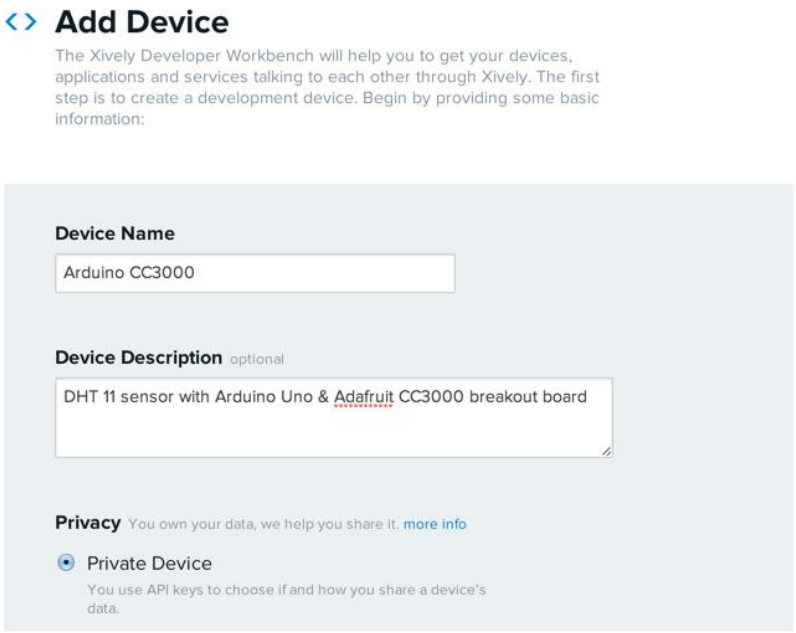


Now that your account is created, you can create a device to track some data. First, click on "Develop":

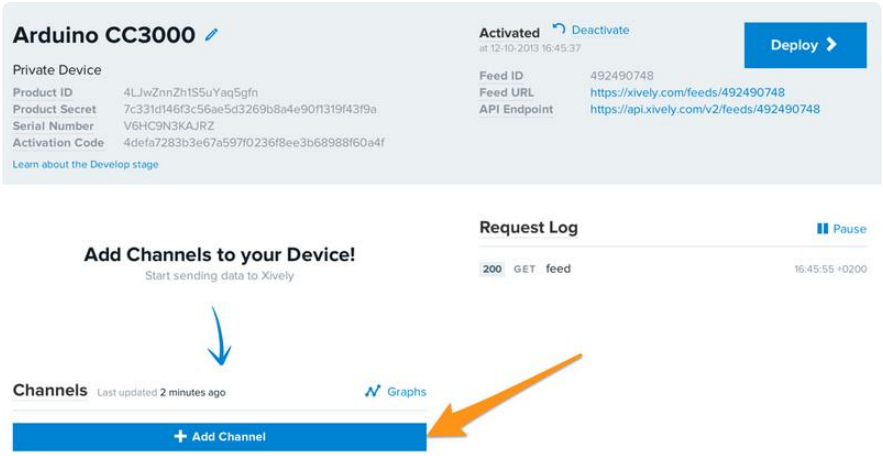
And on "Add Device":



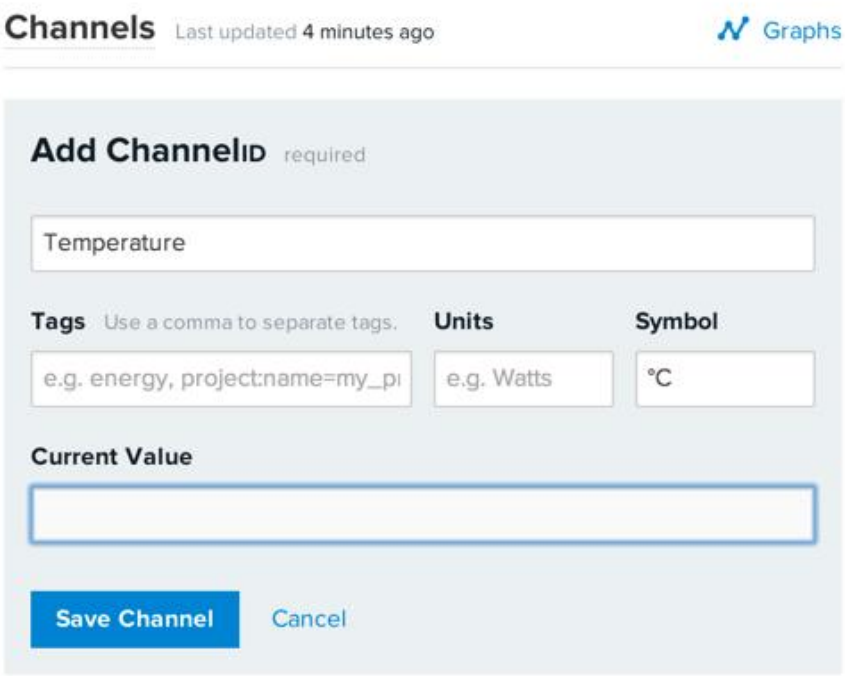
It's now time to enter some information about the device:



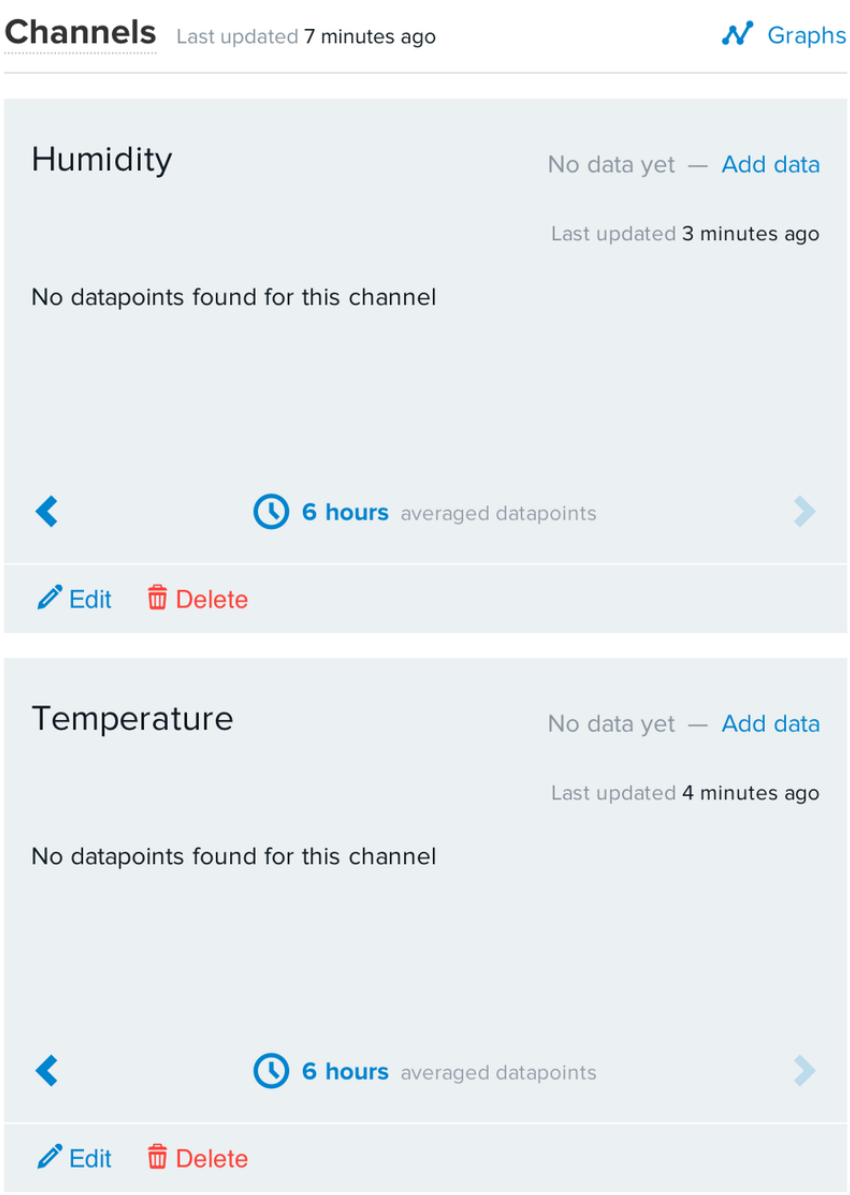
Finally, you should arrive to this page corresponding to your device. What is important on this page is the Feed ID that identify your device, and the API key that identify your account. Please note these two values, you will need them later. Now, we need to create "Channels" to track some data. Click on "Add Channel" to create the first one:



Now enter the details about this first channel for the Temperature. The name of the channels are important, we will use them in the Arduino sketch.



You can now do the same for the Humidity channel. At the end, you will end up with this on your device page:



Your Xively account is now ready to receive some data coming from your Arduino board. Let's now focus on how to connect your Arduino board, the DHT 11 sensor, and the CC3000 chip.

Connections

Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

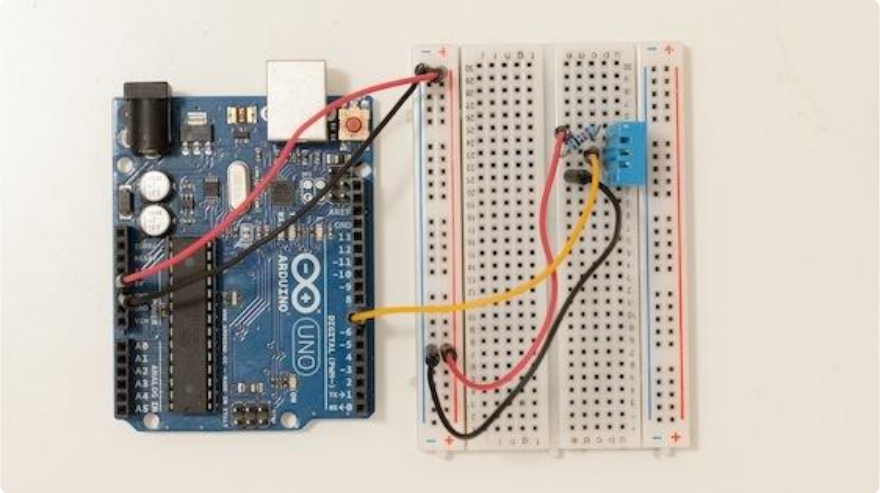
The whole project is based on the Arduino platform, so you will need an Arduino board. I really recommend using the Arduino Uno board for this project, as it is one of the only board that is compatible with the CC3000 library at the time this tutorial was written.

Then, you need the Adafruit CC3000 breakout board (http://adafru.it/1469) to make the WiFi communication, and the DHT11 temperature & humidity sensor (you can also use the DHT22 or the AM2302 sensors which are almost identical to wire up but higher quality). You also need a 10K Ohm resistor to be used with the DHT sensor.

Finally, you need a breadboard and some jumper wires to make the connections between the different parts.

DHT11 sensor

The DHT sensor is quite easy to connect: just plug the pin number 1 to the Arduino's 5V, pin number 4 to GND, and pin number 2 to Arduino pin 7. Finally, put the 10K resistor between the sensor pins number 1 and 2.

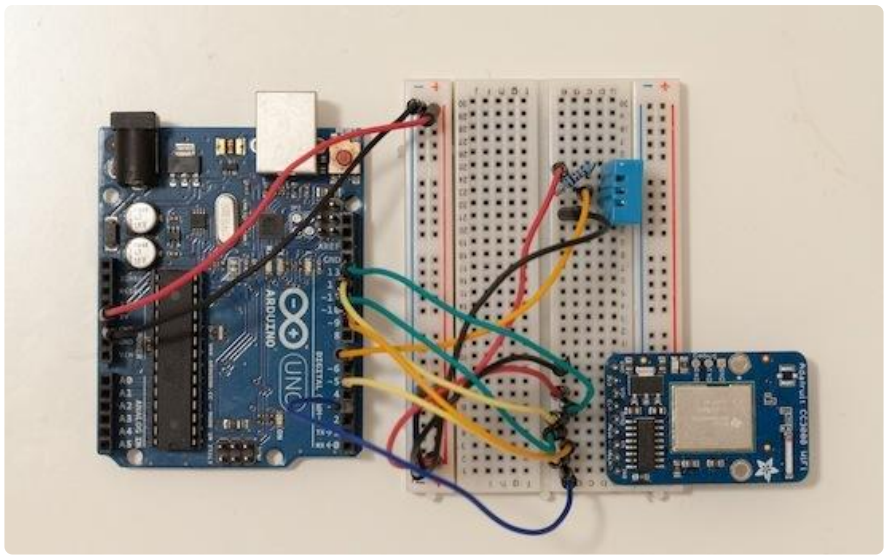


CC3000 Breakout Board

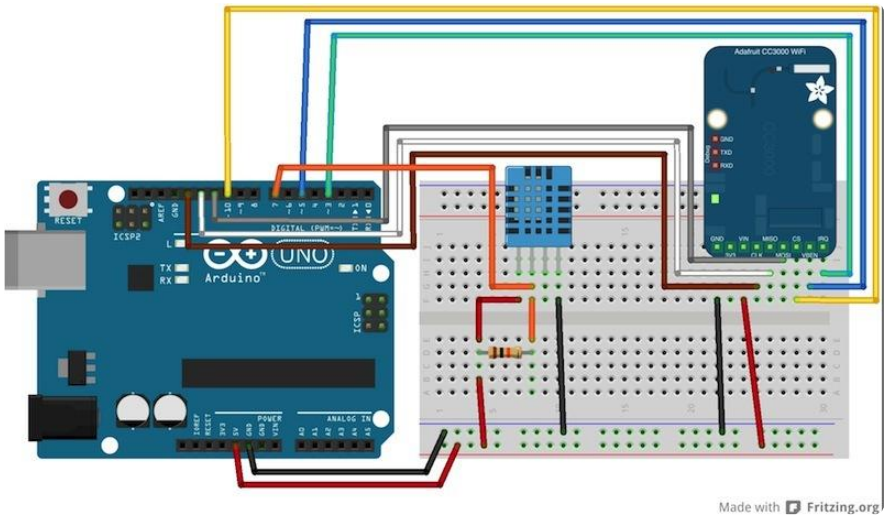
The hardware configuration of the CC3000 breakout board is relatively easy. Connect the IRQ pin of the CC3000 board to pin number 3 of the Arduino board, VBAT to pin 5, and CS to pin 10.

Then, you need to connect the SPI pins of the board to the corresponding pins on the Arduino board: MOSI, MISO, and CLK go to pins 11,12, and 13, respectively.

Finally, you have to take care of the power supply: Vin goes to the Arduino 5V, and GND to GND.



The following picture summarizes the hardware connections:



Arduino sketch

Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

The Arduino sketch for this project consists in initializing the DHT 11 sensor & the CC3000 WiFi chip, and then in running a loop that measures the temperature & humidity, send data to the Xively server, and repeat itself every 10 seconds. It starts by importing the right libraries.

The DHT library is at https://github.com/adafruit/DHT-sensor-library (https://adafru.it/aJX) and CC3000 is at https://github.com/adafruit/Adafruit_CC3000_Library (https://adafru.it/cFn) but be sure to follow the tutorials for both of those products first!

```
#include &lt;Adafruit_CC3000.h&gt;
#include &lt;SPI.h&gt;
#include "DHT.h"
#include &lt;avr/wdt.h&gt;
```

We can then define the correct pins for the CC3000 breakout board:

```
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
```

And the correct pin for the DHT sensor, as well as the sensor type:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

We can then create the CC3000 instance:

```
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);
```

And the DHT sensor instance:

```
DHT dht(DHTPIN, DHTTYPE);
```

The next set of parameters is something your have to modify with your parameters: your WiFi network name, your password, and the type of security.

```
#define WLAN_SSID "yourNetwork"
#define WLAN_PASS "yourPass"
#define WLAN_SECURITY WLAN_SEC_WPA2
```

The next set of parameters concerns Xively. If you remember, I asked you to write down some parameters of your Xively account: your API key and your feedID. Just enter them in these fields, it will be used to make the request to the Xively server.

```
#define WEBSITE  "api.xively.com"
#define API_key  "yourAPIKey"
#define feedID  "yourFeedID"
```

Now, we enter the setup() part of the sketch. As we will connect and disconnect from the Xively server every time we want to send data, the setup() we will only include the initialization of the CC3000 chip, and the connection to the WiFi network:

```
Serial.println(F("\nInitializing..."));
if (!cc3000.begin())
{
  Serial.println(F("Couldn't begin()! Check your wiring?"));
  while(1);
```

```
}

// Connect to WiFi network
Serial.print(F("Connecting to WiFi network ..."));
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
Serial.println(F("done!"));

// Wait for DHCP to complete
Serial.println(F("Request DHCP"));
while (!cc3000.checkDHCP())
{
  delay(100);
}
```

Then, in the loop() part of the sketch, we need to get the IP of the Xively server:

```
uint32_t ip = 0;
Serial.print(F("api.xively.com -&gt; "));
while  (ip  ==  0)  {
  if  (!  cc3000.getHostByName("api.xively.com", &amp;ip))  {
    Serial.println(F("Couldn't resolve!"));
    while(1){}
  }
  delay(500);
}
cc3000.printIPdotsRev(ip);
Serial.println(F(""));
```

We also need to keep in mind that we want the project to run continuously day after day, whatever happens. For example, we want to deploy this project in a remote location and have it run for months without any human intervention. So if the Arduino cannot connect to the Xively server or crashes when sending the data, we don't want it to freeze and do nothing anymore.

This is why we need to use the Arduino watchdog. This will basically reset the Arduino if no reset signal is received after a given delay. Here, we will initialise the watchdog with the maximum delay of 8 seconds:

```
wdt_enable(WDTO_8S);
```

We can now measure the temperature & humidity using the dht instance, and convert these values into integers:

```
float h = dht.readHumidity();
float t = dht.readTemperature();

int temperature = (int) t;
int humidity = (int) h;
```

This is now the time to format the data for the Xively website in the JSON format. It might seem complicated, but the Xively website has many tutorials to format your data correctly. We also need to know the total length (in number of characters) of the data so we can put it in the HTTP request:

```
// JSON data
String data = "";
data = data + "\n" + "{\"version\":\"1.0.0\",\"datastreams\" : [ "
+ "{\"id\" : \"Temperature\",\"current_value\" : \"" + String(temperature) + "\"},"
+ "{\"id\" : \"Humidity\",\"current_value\" : \"" + String(humidity) + "\"}]}";

// Get length
length = data.length();
```

We can now connect to the Xively server with the following piece of code. If it is successful, we print a message on the Serial monitor:

```
Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
if (client.connected()) {
  Serial.println(F("Connected to Xively server."));
```

When the client is connected, we can send the request to the server. This is a typical HTTP PUT request, where we specify the feedID, the API key, and send the data at the end of the request. We first send the headers, and we reset the watchdog after this step:

```
Serial.print(F("Sending headers"));
client.fastrprint(F("PUT /v2/feeds/"));
client.fastrprint(feedID);
client.fastrprintln(F(".json HTTP/1.0"));
Serial.print(F("."));
client.fastrprintln(F("Host: api.xively.com"));
Serial.print(F("."));
client.fastrprint(F("X-ApiKey: "));
client.fastrprintln(API_key);
Serial.print(F("."));
client.fastrprint(F("Content-Length: "));
client.println(length);
Serial.print(F("."));
client.fastrprint(F("Connection: close"));
Serial.println(F(" done."));

// Reset watchdog
wdt_reset();
```

We can now transmit the core of the data itself. We transmit the JSON data in several chunks using a dedicated function called sendData. This function cuts the data into small pieces, send these pieces one by one, and reset the watchdog after each chunk is sent. This way, we are protected in case the transmission doesn't work and makes the Arduino freeze.

The size of a chunk is defined in the buffer_size variable. Depending on your connection speed, you might have to change this variable so the watchdog doesn't reset the sketch every time.

```
Serial.print(F("Sending data ..."));
client.fastrprintln(F(""));
sendData(client,data,buffer_size);
```

```
client.fastrprintln(F(""));
Serial.println(F("done."));
```

After the request is sent, we read the answer from the server to be sure that everything is ok, and we print it on the Serial monitor. This is be useful when trying to debug the project.

```
while (client.connected()) {
 while (client.available()) {
  char c = client.read();
  Serial.print(c);
 }
}
```

After the answer has been received from the server, we close the connection, and disable the watchdog until the next loop():

```
client.close();
Serial.println(F("Closing connection"));

// Reset watchdog &amp;amp; disable
wdt_reset();
wdt_disable();
```

Because the temperature & humidity are slow-changing values, we can read these values & send them over to Xively only a few number of times every hour. For this guide, I used a delay of 10 seconds to make sure that the sketch is working, but you can use any value you want. This part is done by using a dedicated function called wait, which is also protected by the watchdog in case the Arduino sketch crashes:

```
wait(10000)
```

Finally, the complete code can be found in the GitHub repository of this project (https://adafru.it/cPv).

Using Xively

Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

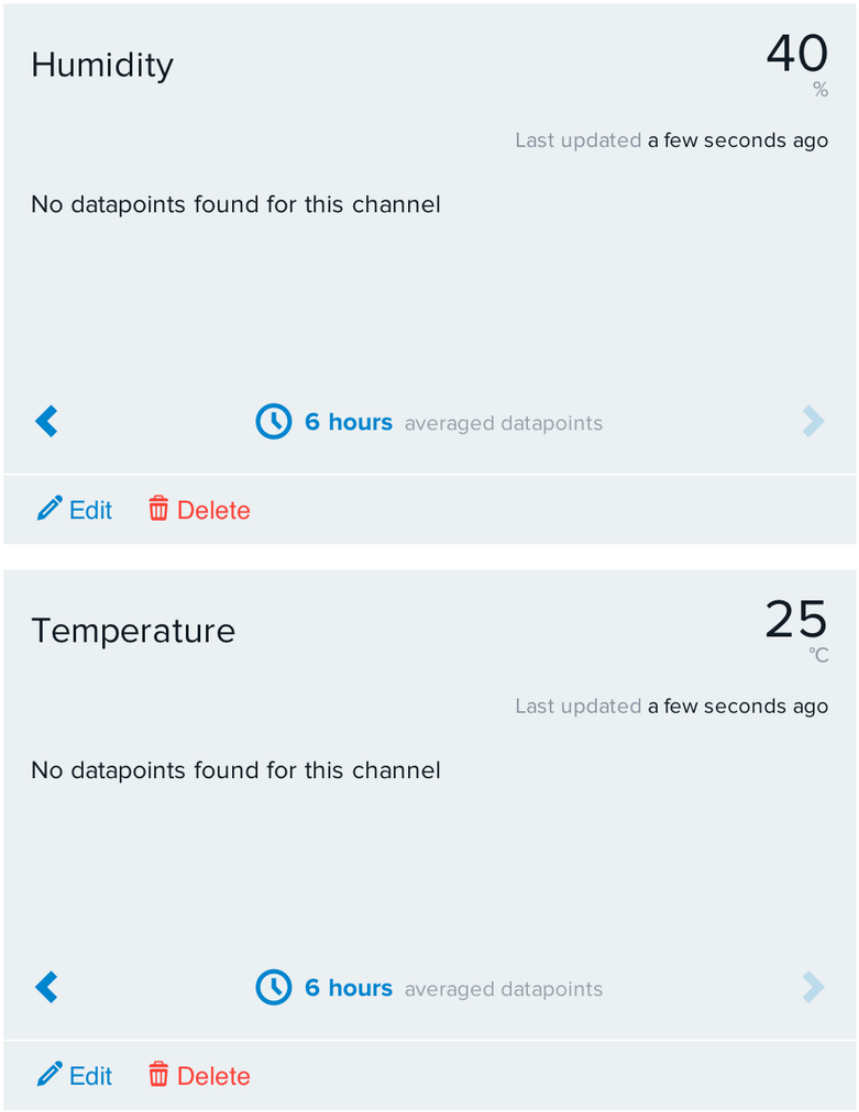
Now that the Arduino sketch is completed, you can upload the sketch to your Arduino board, and test the project. Make sure that the page corresponding to your device on Xively is opened. You can now open your Serial Monitor and you should see the Arduino connecting to your network, preparing the request, and sending it to the Xively website:

```
Initializing WiFi chip...
Connecting to WiFi network ...done!
Request DHCP
Checking WiFi connection ...done.
Pinging Xively server ...done.
Connected to Xively server.
Sending headers.... done.
Sending data...
```

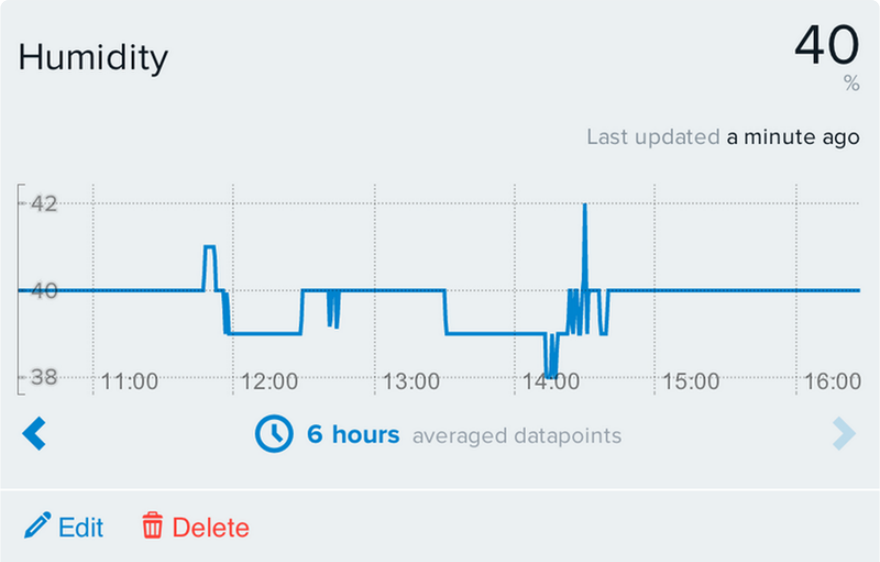
If everything goes well, you should see this response from the Xively server:

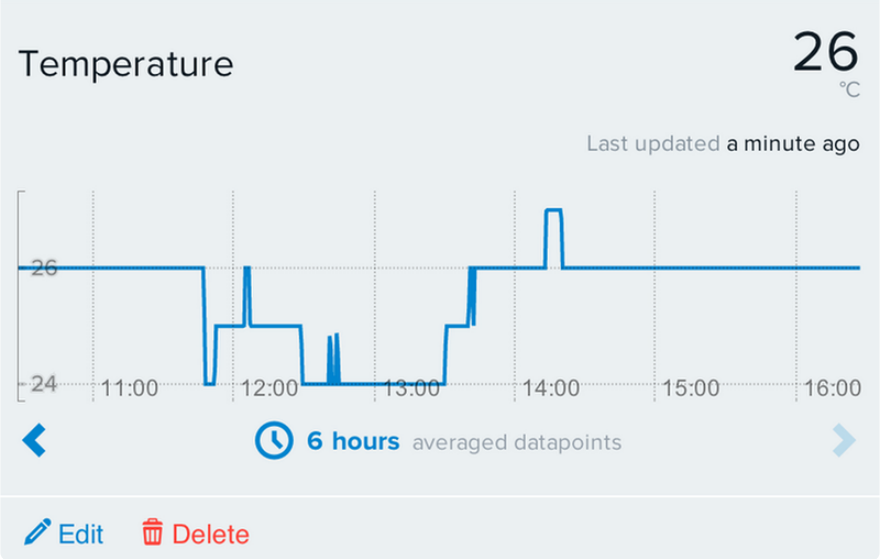
```
HTTP/1.1 200 OK
Date: Mon, 14 Oct 2013 17:44:20 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 0
Connection: close
X-Request-Id: 55c792d07f4a679dfb8a1a09141264d7c98eea1e
Cache-Control: max-age=0
Vary: Accept-Encoding
```

And finally, have a look on your Xively page in your browser. You should see that you successfully sent some information to the Xively server and the temperature & humidity should be displayed on this page:



After a some measurements, you should also have the graphs displayed on the device's page:





Finally, I also made a video showing the different aspects of the project:

Of course, you can use this tutorial to connect other sensors to Xively: motion sensors, contact switches, luminosity sensors, ... the possibilities are endless, so don't hesitate to experiment and share your results!