adafruit learning system

Wireless Gardening with Arduino + CC3000 WiFi Modules

Created by Marc-Olivier Schwartz

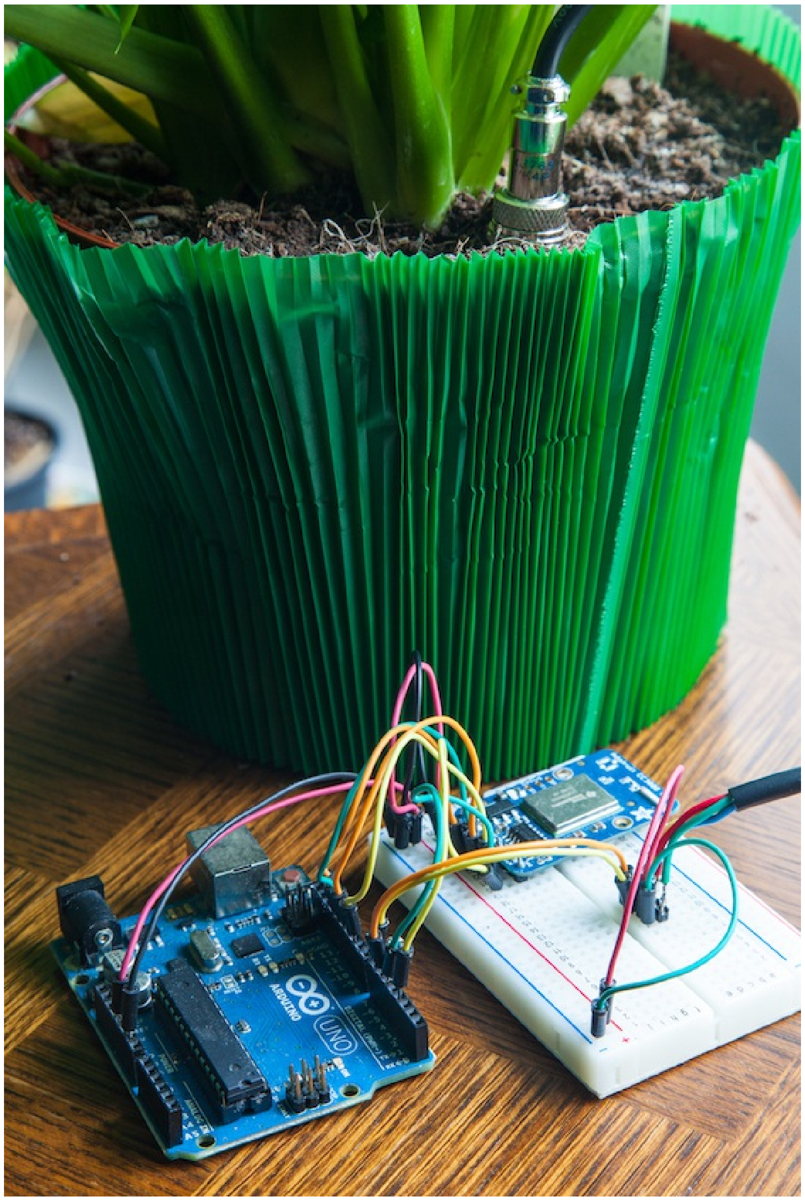
Last updated on 2014-01-10 03:00:47 PM EST

Guide Contents

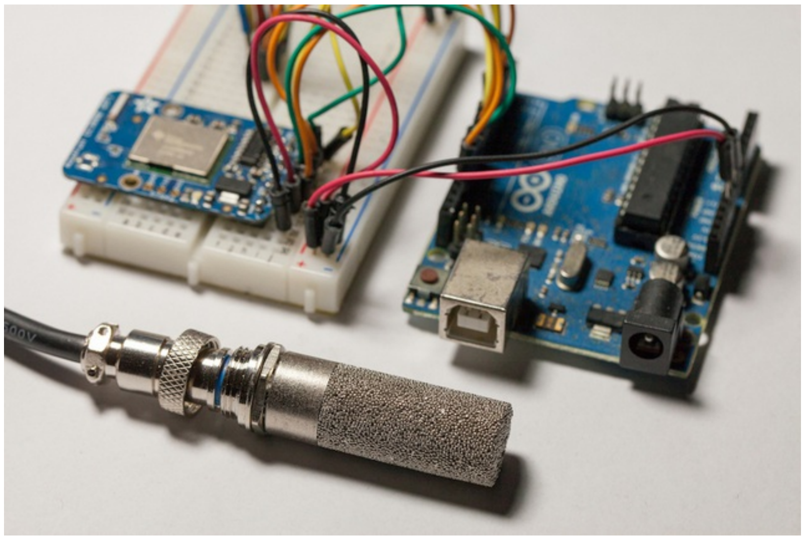
Introduction

In this guide, we are going to give a modern touch to gardening and connect some informations about your garden to the Internet. We are going to use a soil moisture & temperature sensor connected to an Arduino and a WiFi chip to automatically send measurements from your garden to the cloud.

We'll use a service call Carriots (http://adafru.it/d5e) to handle the data and display it nicely on a webpage. Then, an email or SMS alert can be send to you automatically if the moisture falls below a given threshold. The picture below represents the system when fully assembled and with the sensor buried into the soil next to a plant:

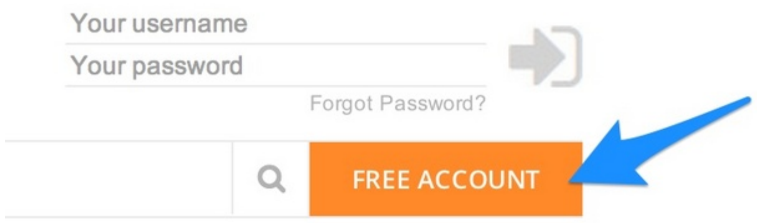


Don't worry, you don't actually need to have a garden or even plants (although we think plants are great to have) to use the content of this article: what you are going to learn can be used for any remote measurement projects. Let's dive into the project!

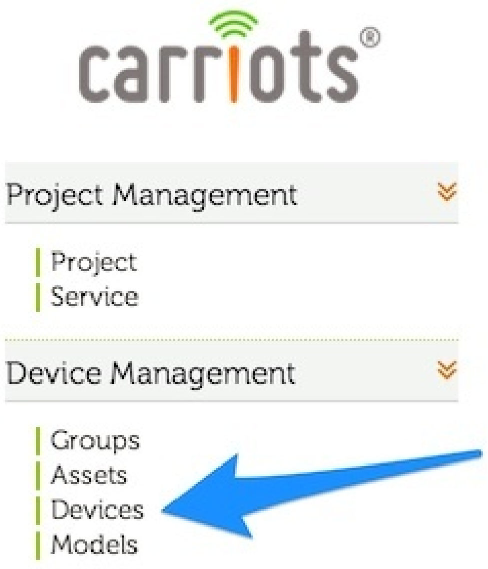


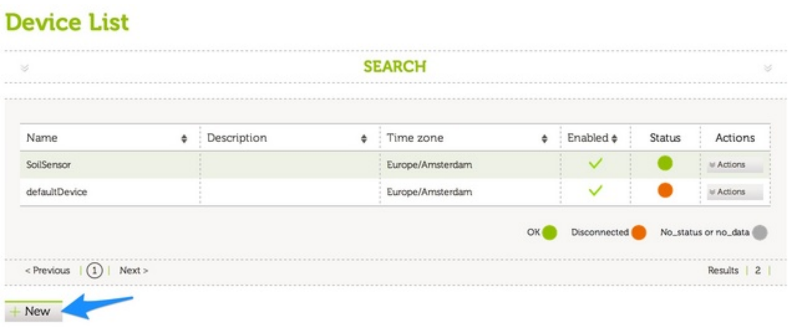
Setting up your Carriots account

To upload data to Carriots (http://adafru.it/d4l), you first need to create an account:



When the account is created, you will have to create a device, which is the entity that will receive the data from our hardware project:





Device List

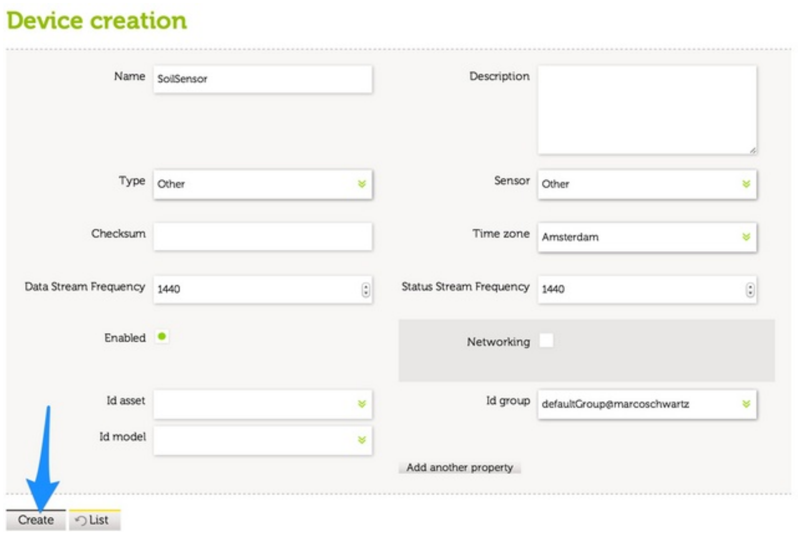
SEARCH

Name	Description	Time zone	Enabled	Status	Actions
SoilSensor		Europe/Amsterdam	✓	●	Actions
defaultDevice		Europe/Amsterdam	✓	●	Actions

OK ● Disconnected ● No_status or no_data ●

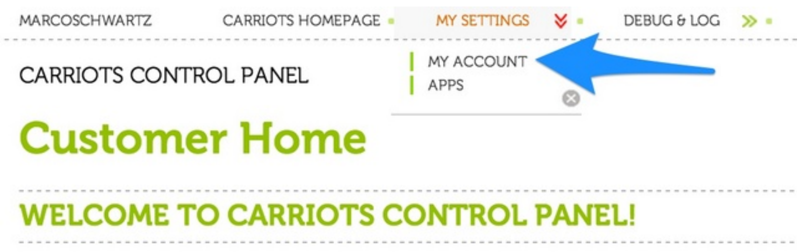
< Previous | 1 | Next >   Results | 2 |

+ New


Device creation

Name	SoilSensor	Description	
Type	Other	Sensor	Other
Checksum		Time zone	Amsterdam
Data Stream Frequency	1440	Status Stream Frequency	1440
Enabled	●	Networking	
Id asset		Id group	defaultGroup@marcoschwartz
Id model			

Finally, you will need to know your API key for the rest of the tutorial. You can find the key under the "MY ACCOUNT" menu:



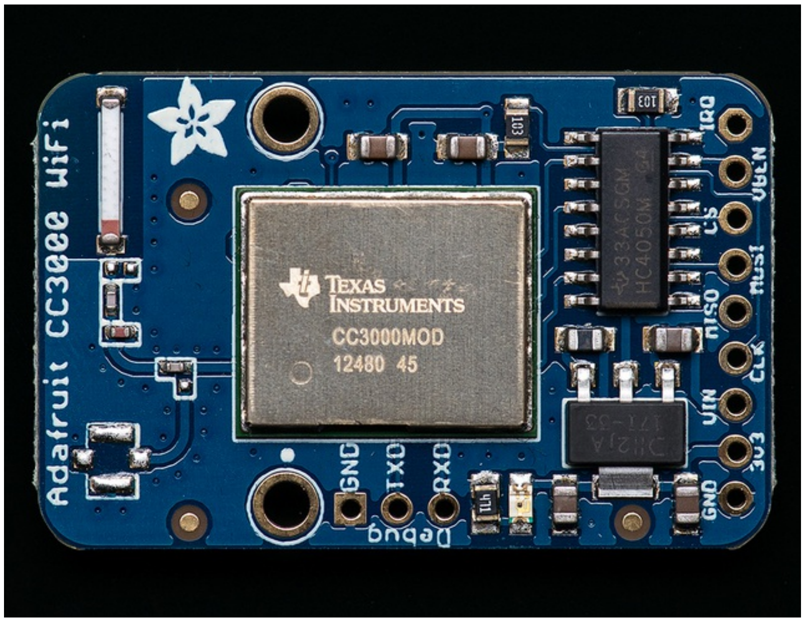
When all these steps are done, congratulations, you are ready to send data to your newly created device! Be sure to keep the device name and your API key around, you'll need them for the next part of the guide.

Connections

The whole project is based on the Arduino platform, so of course you will need an Arduino board. I will use an Arduino Uno board for this project. Then, you need a board with the CC3000 chip. What I recommend is using the Adafruit CC3000 breakout board, which is the only one I tested that worked without problem.

To measure the temperature & humidity in the soil, you will need an appropriate sensor. For this guide, I used the Soil Temperature/Moisture sensor from Adafruit, which is based on the SHT10 sensor from Sensirion. It's quite easy to use with Arduino as there is a dedicated library, so it will be easy to interface with our project.

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

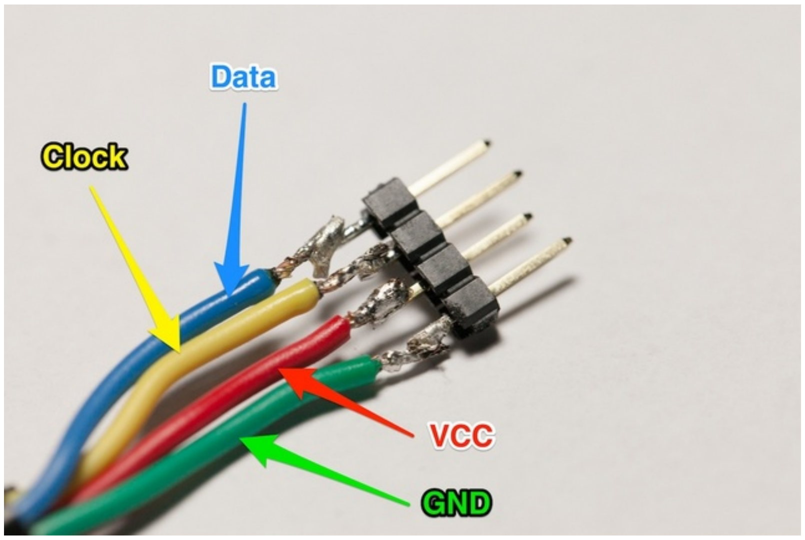


First, you need to connect the CC3000 chip to the Arduino Uno board. To get a step-by-step guide on how to connect and use the CC3000 breakout board, I recommend to visit this detailed guide (http://adafru.it/d4J).

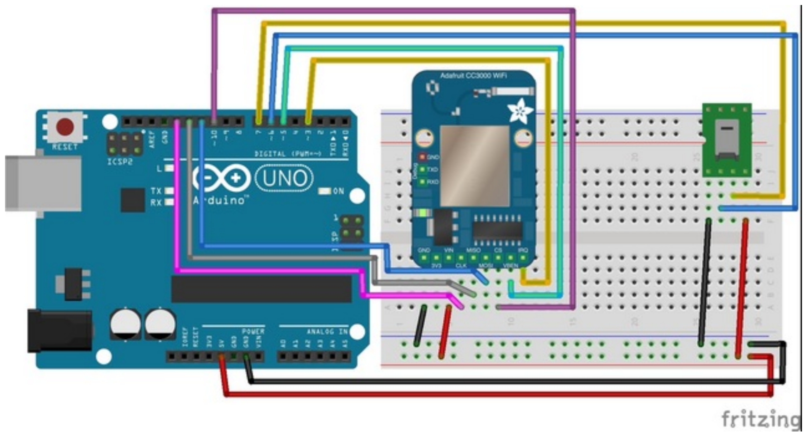
Basically, you need to connect the IRQ pin of the CC3000 board to pin number 3 of the Arduino board, VBAT to pin 5, and CS to pin 10. Then, you need to connect the SPI pins to the Arduino board: MOSI, MISO, and CLK go to pins 11,12, and 13, respectively. Finally, take care of the power supply: Vin goes to the Arduino 5V, and GND to GND.

Make sure to run the Adafruit CC3000 test example sketches before you continue - WiFi is easier to debug before you add all the sensors and extras!

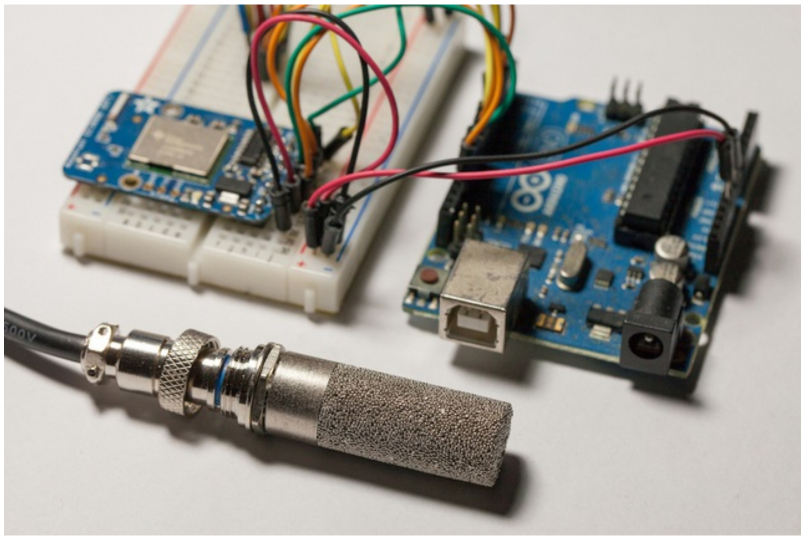
Connecting the soil sensor is relatively easy. It comes with wires not connected to any pins, so I simply soldered a 4-pins header to the 4 wires of the sensor. Simply connect the GND pin to the GND of the Arduino, VCC to 5V, Clock to Arduino pin 7, and Data to Arduino pin 6. The following picture explains the roles of the different wires coming from the sensor:



The following picture summarises all the required connections for the hardware part of this project:



When the project is fully assembled, this is how it should look like:



Arduino sketch

For this project, you simply need the Arduino IDE, the CC3000 library (http://adafru.it/cFn), and the Sensirion library (http://adafru.it/d50). Make sure that the libraries are correctly placed in your /libraries folder inside your main Arduino folder.

The code for this project can be found on the GitHub repository for the project (http://adafru.it/d51).

Now everything is set so that you can send data over to the Carriots cloud service. We have to write the sketch that will be uploaded to the Arduino Uno board. Because the Carriots service uses a REST API, the sketch will actually be quite similar to what was done in the Arduino & Xively project (http://adafru.it/d52): establishing a connection to the Carriots server, measuring some data from the sensor, and then formatting & sending the data as a JSON object.

I won't detail everything here, you can have a look at the Arduino & Xively project (http://adafru.it/d52) for more details, and of course you can find all the code on our GitHub repository for this project (http://adafru.it/d51). The first difference with other cloud services is that you need to set your API key in the sketch, but also the device you want to send data to:

```
#define WEBSITE  "api.carriots.com"
#define API_KEY "yourAPIkey"
#define DEVICE  "yourDeviceName@yourUserName"
```

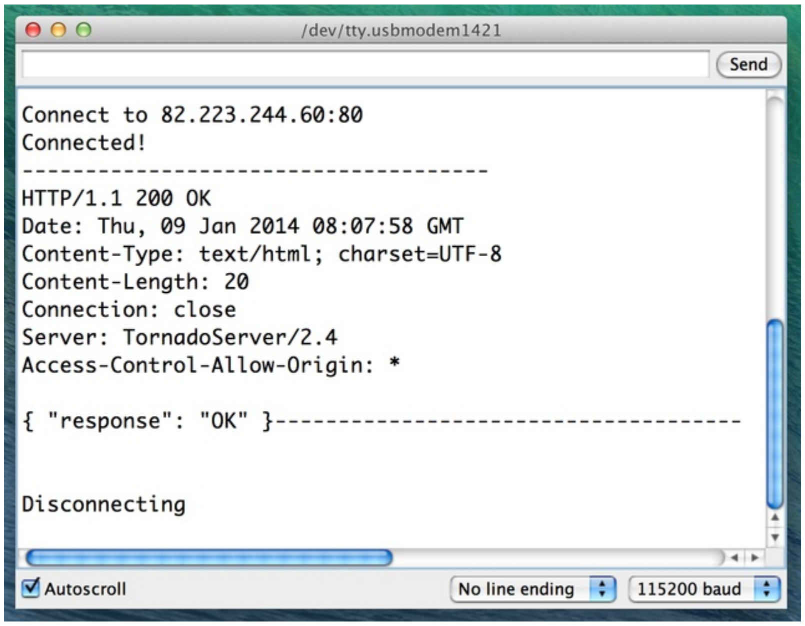
In the loop() part of the sketch, you need to format the data measured by the Arduino according the Carriots guidelines (http://adafru.it/d53). For example, data sent by our sketch should have the following form:

```
{
   "protocol":"v2",
   "at":"now",
   "device":"yourDevice@yourUserName",
   "data":{
      "Temperature":"21.05",
      "Humidity":"58.50"
   },
   "checksum":""
}
```

Then, at the end of each loop, this data is sent along with an HTTP header. We then read the answer back from the server, and you can check it out on the serial monitor. If the server accepts the data, you should see an "OK" message printed out. Finally, the connection is closed and we wait for 10 seconds before the next measurement.

You can get the whole Arduino sketch from the GitHub repository (http://adafru.it/d51), modify the required parts like your WiFi SSID & password and your Carriots account data, and finally upload the code to your Arduino board. You can now open the serial monitor and check if the

Arduino is sending data to the Carriots server:

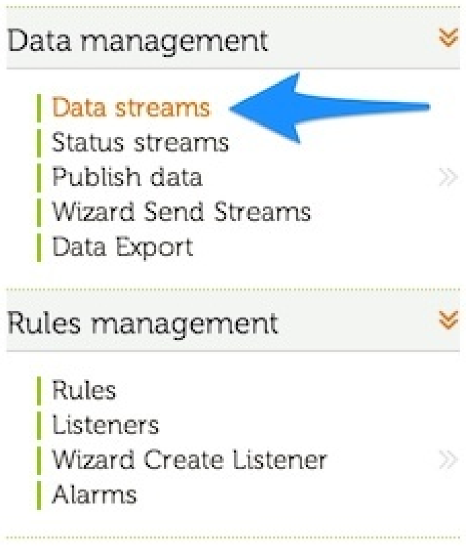


```
Connect to 82.223.244.60:80
Connected!
-------------------------------------
HTTP/1.1 200 OK
Date: Thu, 09 Jan 2014 08:07:58 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 20
Connection: close
Server: TornadoServer/2.4
Access-Control-Allow-Origin: *

{ "response": "OK" }-------------------------------------


Disconnecting
```

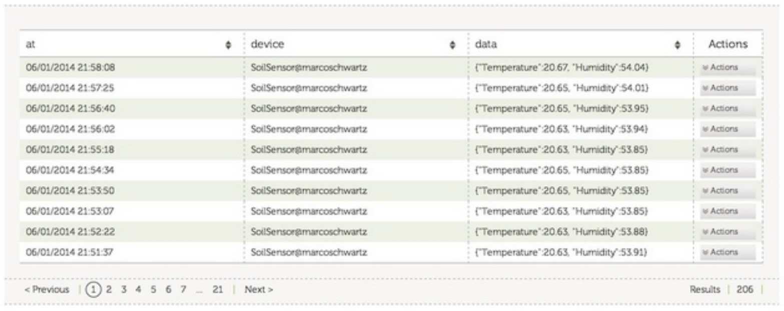
Autoscroll ✓ | No line ending | 115200 baud

Sending data to Carriots

To check if the Carriots service received some data, you can simply go in your Carriots account, under "Data streams":



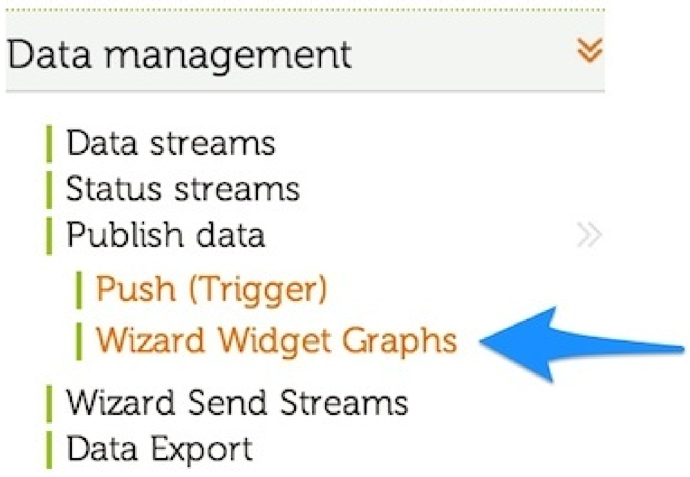
You should see some data recorded for your device:



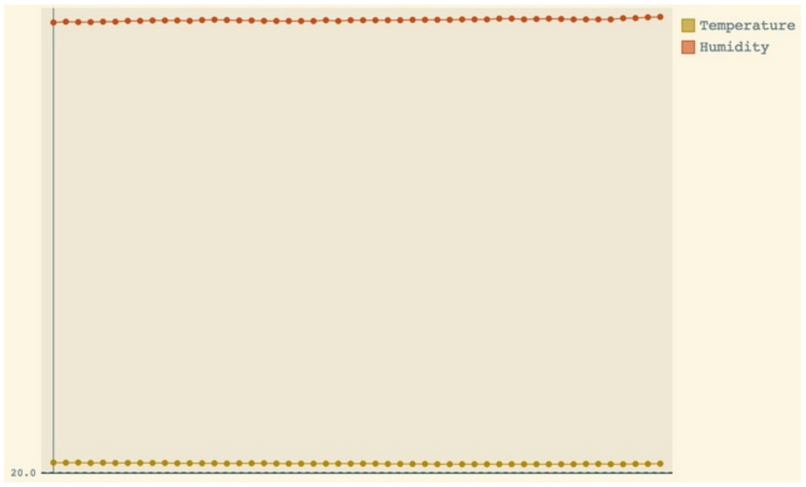
at	device	data	Actions
06/01/2014 21:58:08	SoilSensor@marcoschwartz	{"Temperature":20.67, "Humidity":54.04}	Actions
06/01/2014 21:57:25	SoilSensor@marcoschwartz	{"Temperature":20.65, "Humidity":54.01}	Actions
06/01/2014 21:56:40	SoilSensor@marcoschwartz	{"Temperature":20.65, "Humidity":53.95}	Actions
06/01/2014 21:56:02	SoilSensor@marcoschwartz	{"Temperature":20.63, "Humidity":53.94}	Actions
06/01/2014 21:55:18	SoilSensor@marcoschwartz	{"Temperature":20.63, "Humidity":53.85}	Actions
06/01/2014 21:54:34	SoilSensor@marcoschwartz	{"Temperature":20.65, "Humidity":53.85}	Actions
06/01/2014 21:53:50	SoilSensor@marcoschwartz	{"Temperature":20.65, "Humidity":53.85}	Actions
06/01/2014 21:53:07	SoilSensor@marcoschwartz	{"Temperature":20.63, "Humidity":53.85}	Actions
06/01/2014 21:52:22	SoilSensor@marcoschwartz	{"Temperature":20.63, "Humidity":53.88}	Actions
06/01/2014 21:51:37	SoilSensor@marcoschwartz	{"Temperature":20.63, "Humidity":53.91}	Actions

< Previous | 1 2 3 4 5 6 7 ... 21 | Next > Results | 206

Pretty cool, right ? But that's not all: Carriots can actually help you to plot this data & embed it into your own website if you want. For that, simply go to the "Wizard Widget Graphs" link on the left menu:

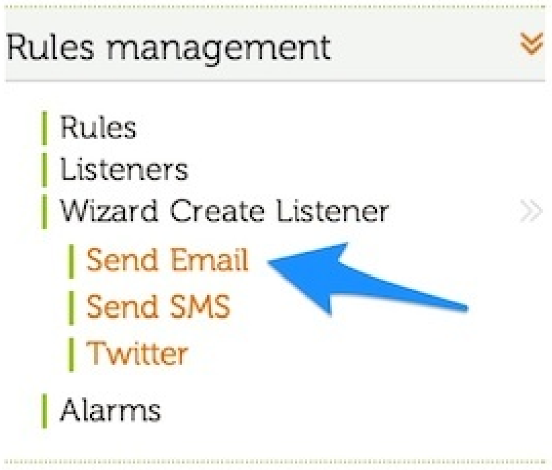


From there, you have a nice step-by-step menu to choose the type of graph you want, the data you want to plot, and finally you will get access to the code that you can embed in a webpage to automatically display the graph. I also included an example of such a webpage in the GitHub repository of the project (http://adafru.it/d51). This is an example with the data I recorded:

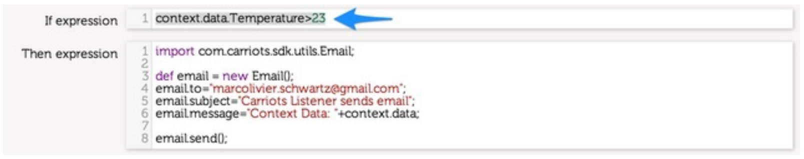


Creating an email alert

To finish this project, we'll create an automatic alert for one of the variable. For example, we want to know if the humidity falls under a given threshold, indicating that the plant needs to be watered. In Carriots, this means creating a "Listener" that will automatically send you an email when an event happens, in this case if the data crosses a given threshold. You can create a listener with the Carriots wizard:



You will be prompted to enter your email, and then to configure the listener. To test my listener, I simply entered the condition that an email will be sent if the temperature reaches 23 degrees Celsius:

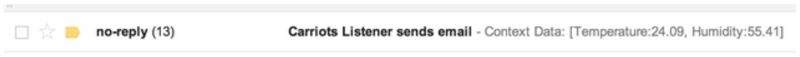

If expression

```
context.data.Temperature>23
```

Then expression

```
import com.carriots.sdk.utils.Email;

def email = new Email();
email.to="marcolivier.schwartz@gmail.com";
email.subject="Carriots Listener sends email";
email.message="Context Data: "+context.data;

email.send();
```

I then applied some heat on the sensor, and I immediately received an email when a data point above 23 degrees was recorded:


no-reply (13) **Carriots Listener sends email** - Context Data: [Temperature:24.09, Humidity:55.41]

You can also add listeners on other variables, and send an SMS to your phone instead of an email. Of course, you can find the whole code for this project on the GitHub repository of the project (http://adafru.it/d51).

That's all for this article, keep in mind that you can use what you've learned in this guide for a lot of other fields than gardening! You can connect multiple sensors inside & outside of your home to the Carriots service, and create more complex conditions to send specific alerts based on the data you are recording. Have fun with the project, experiment with this cloud service, and share your creations!