YunServer · Sensor values to HTML page

matlo
12/09/2013

1. Start from the examples

f you look under the Bridge library examples, there's one called “TemperatureWebPanel” that takes a value from an Arduino pin, then sends it to the server using the Bridge library and finally prints it on the HTML page. This is a very basic application but the concept is super powerful and it’s exactly what we are recreating from scratch in this tutorial. We start from this example to understand some core concepts of this system.

All the processes that are sending and receiving data from Arduino to Linino and reverse are carried on by the Bridge Library. This new piece of software comes with the IDE 1.5.4 and makes it very simple to communicate with the Linino part of your Arduino board. Have a look at the Bridge examples or look at the reference for more details

2. TemperatureWebPanel

Let's see how to build something interactive ourselves, starting from the TemperatureWebPanel example. We'll try to do the same but with less data, to grasp the core concepts.

To make this work, we need two elements: our **Arduino sketch** and a **HTML page**.

The HTML page will send requests to the Yun server, who will answer with some value that the page will display. The communication will be like: "Yun, can you tell me the temperature value?" "It's 34 degrees".

Let's start from the HTML page: we are not focusing on the content as much as the functionality. Our page will be empty with just the numeric value of the sensor. If you already know some CSS wizardry you can customize it.

To send requests, we need to write some Javascript. If you're new to this, don't worry, we are going to use a library that simplifies the job. In the TemperatureWebPanel example, they used Zepto, a lighter version of jQuery (but you can use jQuery or any other JS library, as everything will be uploaded on the SD card). We are just imitating the example, to better understand it, so we are going to use Zepto as well.

3. HTML & Zepto

Make a new empty folder, then place a new **index.html** file inside (you can make it with any text editor, personally I use SublimeText2).

Second step is copying **zepto.min.js** in that folder (you can copy it from the "www" folder of the TemperatureWebPanel example, or download it from their website).

Zepto is nothing more than a lightweight version of jQuery. To make it so small, they had to sacrifice browser compatibility. If this doesn't work in your browser, just use standard jQuery. In our sketch, jQuery / Zepto is used for the $("#content").load function, that without any library, in pure JavaScript, it would be longer and more complicated

4. index.html

The content of our **index.html** page:

```
&lt;!DOCTYPE html&gt;
&lt;head&gt;
 &lt;script type=&quot;text/javascript&quot;
src=&quot;zepto.min.js&quot;&gt;&lt;/script&gt;
```

```
    &lt;script
type=&quot;text/javascript&quot;&gt;
        function refresh() {
            $('#content').load('/arduino/temperature');
    }
    &lt;/script&gt;
&lt;/head&gt;
&lt;body onload=&quot;setInterval(refresh,
1000);&quot;&gt;
 &lt;span id=&quot;content&quot;&gt;Waiting
for Arduino...&lt;/span&gt;&lt;/body&gt;
```

5. HTML in details

In the

, after including Zepto, make a new script in which we create a function, refresh(). It simply loads into our 'content' span, the full content of the "/arduino/temperature" URL.

```
function refresh() {
        $('#content').load('/arduino/temperature');
    }
```

The function above can be read as: “load into the element tagged as ‘content’, the content of the ‘/arduino/temperature’ URL.

In the **body** tag, you can call the refresh function every second (1000 milliseconds) with the javascript "setInterval" method.
Inside the body, make an element with the '**content**' id, that is where our message from Arduino will be loaded.

When our script loads the content every second, it makes a **call** to that URL (/arduino/temperature) on the Yún server. The Yún receives that call and answers back with the value, that is loaded as it is, straight into the "content" div

6. Arduino Sketch

Now let's see the Arduino side:

Open up the Arduino IDE and start a new sketch.

Let's start by including the libraries that we use, at the top of our sketch, before the setup(), with that list of **#include.**

Then initialize a YunServer object, named server.

```
YunServer server;
```

Now in the setup function, we start Bridge and server on the localhost (no one from the external network can connect)

```
void setup() {
 Bridge.begin();
 server.listenOnLocalhost();
 server.begin();
}
```

Then in the loop, we create a client object to get the client requests

```
 // Get clients coming from server
 YunClient client = server.accept();
```

Our code will be executed only “on-demand”, that is when the Arduino receives a client request

```
if(client) {
...
}
```

If we receive an answer, the first thing that we want to do is to save the content in a String object and eliminate the whitespaces, with the trim() method.

```
String request = client.readString();
request.trim();
```

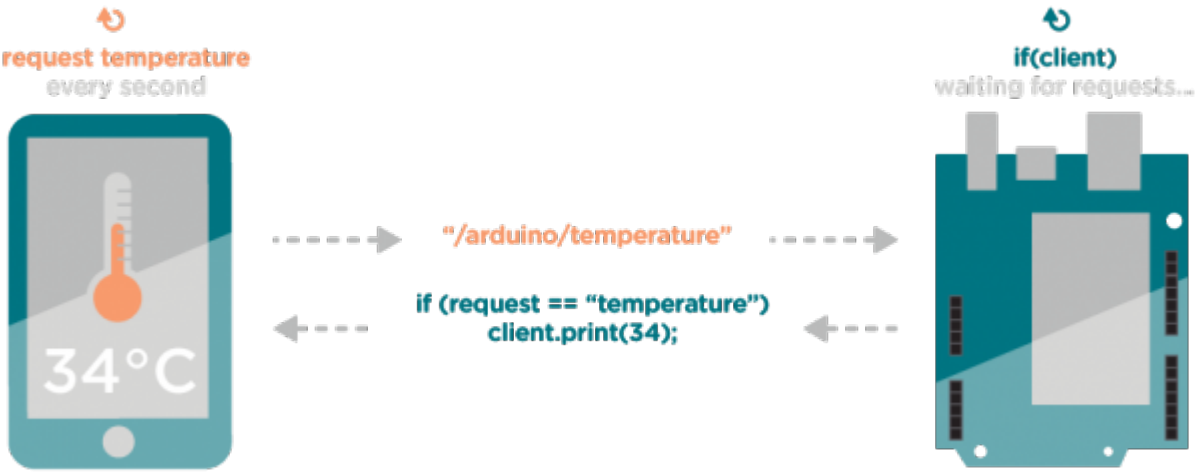
After that we want to operate a check on the String. This is probably unnecessary if we are dealing with a single element, as we can just send the element without any “label”; however if we are dealing with multiple elements, this is the way to do it. We check that the string contains the “temperature” substring.

```
if (request == "temperature") {
    int val = analogRead(A1);
    client.print(val);
  }
```

If the request string is "temperature", read the value from the pin A1, where the temperature sensor is connected, then print it on the client.
Everything that you put in the client.print() brackets, it’s going to be reported in the "content" span of our HTML page.

In this case just the value of the A1 pin, but if you have a look at the TemperatureWebPanel example, it prints also time and number of hits.

Although we are sending a string that is "/arduino/temperature", the "/arduino" part is not considered in the sketch (even though it's necessary in the JS part), so our string is just "temperature/value".



7. Up & Running

The last step of this process is to upload everything on the Yún.

After saving your sketch (you can see the full code after the break), make the **www** subfolder and copy **index.html** and **zepto.min.js** into it. Now select the IP Yún from the **Tools->Port** menu and upload it via WiFi.

To see the webpage just open a browser window and go to this URL:

http://arduino.local/sd/mysketch
("arduino" is your board 's name and "mysketch" your sketch name)

If everything went fine, at the beginning you should see just a zero on the top left of the page, but after some seconds it will start to print values from the A1 analog port.

You can access this page from any device connected to the same network, just copy the link.


```
#include &lt;Bridge.h&gt;
#include &lt;YunServer.h&gt;
#include &lt;YunClient.h&gt;
YunServer server;
void setup() {
  Bridge.begin();
  server.listenOnLocalhost();
  server.begin();
}
void loop() {
  YunClient client = server.accept();
  if (client) {
    String command = client.readString();
    command.trim();
    if (command == &quot;temperature&quot;) {
      int val = analogRead(A1);
      client.print(val);
    }
    client.stop();
  }
  delay(50);}
```