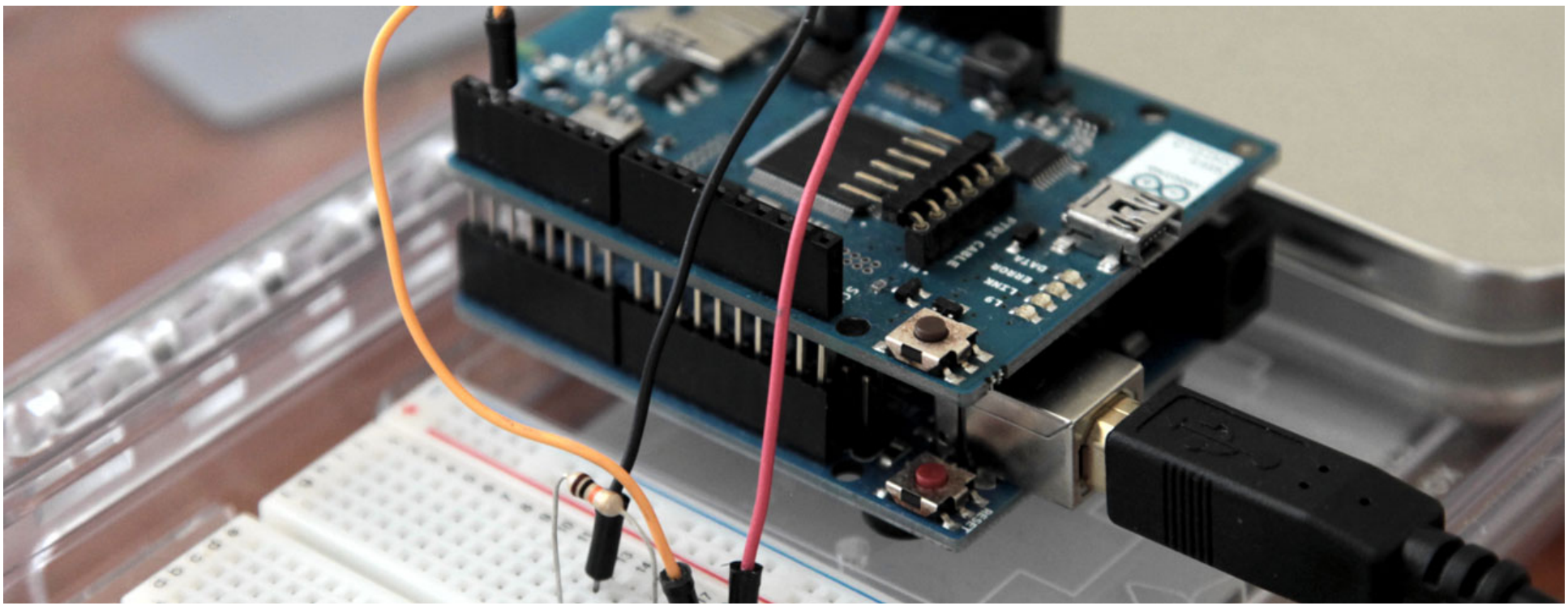


Monitor your home remotely using the Arduino WiFi Shield

In this article, I will show you how to monitor some data in your home using precisely this Arduino WiFi shield.

Along with the Arduino Uno board, the final system will form an autonomous solution to monitor one or several sensors in your home. You could for example be interested in monitoring some contact sensors that are installed on your doors, and you want to know if the door has been opened or not. To emulate one of these sensors, I will simply use a small push button in this tutorial, but the principle is exactly the same.

Hardware requirements

The first thing you will need is an Arduino board. As usual, I used the Arduino UNO board for this tutorial, but any other official Arduino board should be fine as well. The second most important element in this tutorial is the Arduino WiFi shield. You will also need a small push button to emulate the sensor you want to measure, and a 10K ohms resistor.

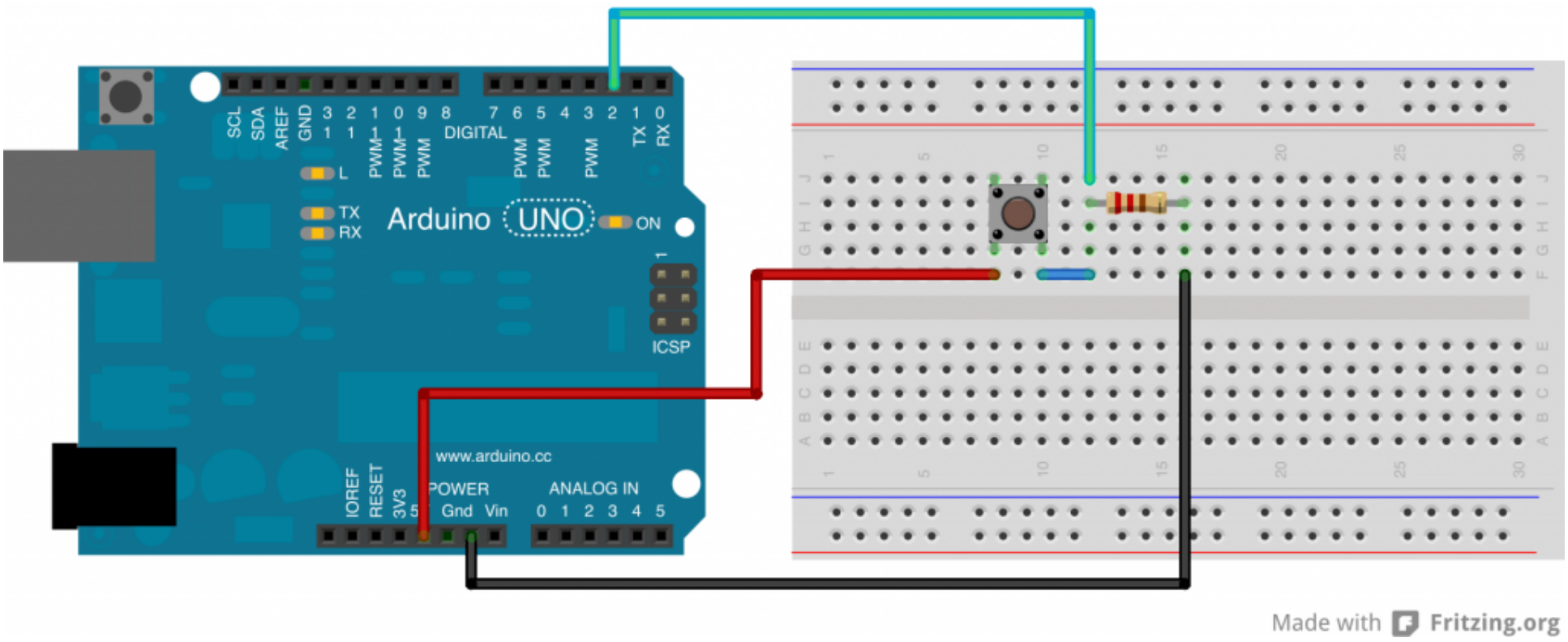
Because we are using the WiFi shield, you will also need to have a wireless network running in your home, with WEP or WPA2 personal authentification. Don't worry, I will show you how to do it in both cases.

Software Requirements

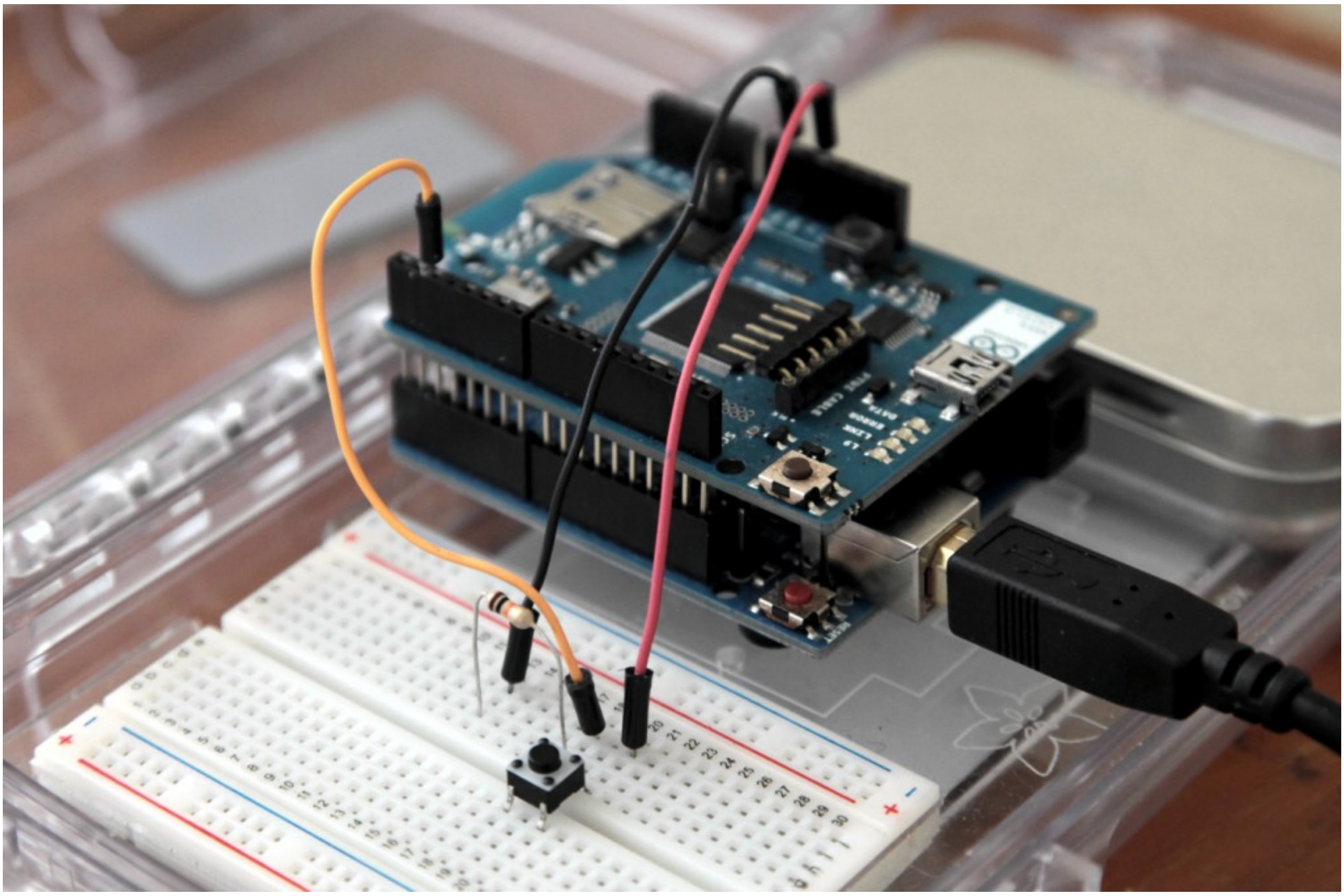
Appart from the latest version of the Arduino IDE, you don't need much software for this tutorial.

Hardware Configuration

The hardware configuration is pretty simple. First, you have to plug the WiFi shield into your Arduino board. Then, you have to connect the resistor and the push button to the Arduino shield using a pull-down configuration: this means that if the button is not pressed, the input will be "pulled" to the ground. At the end, the button will be connected on pin number 2. Why number 2 ? This is because this is an interrupt pin, we will why we need that later. This is the schematic of this configuration:



Without this, the input could actually have a random state and we would trigger a “door has been opened” action even if nothing actually happened. The result should look like this:



Testing Individual Parts

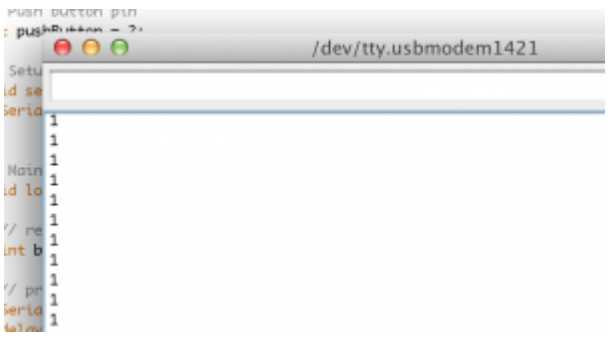
Let’s now really dive into this project and test every part separately. First, let’s see if your Arduino board can connect to your local wireless network. For this, it is really easy, you can just use the official Arduino sketch for WPA networks which you can find here. If you have a network with WEP protection, you can find the corresponding tutorial here. Just modify your wireless network name, set the right password, and then upload the sketch to the shield with the Arduino IDE. Open the serial monitor, and you should see this:

See this IP address ? This is the address of your shield. For now, just write it down, we'll use it later. By the way, the "link" LED should turn green when the shield is connected. Sometimes it takes a while, so be patient ☺

Finally, let's test the push button. We want to detect the state of the button, so we will use the function digitalRead(). Here is the sketch to test the button:

```
// Push button pin
int pushButton = 2;

// Setup
void setup() {
 Serial.begin(9600);
}

// Main loop
void loop() {

 // read the pin number 2
 int buttonState = digitalRead(pushButton);

 // print out the state of the button
 Serial.println(buttonState);
 delay(1);
}
```

Now upload the sketch, and open the serial monitor. You should see a lot of zeros, which is logical because the pin number 2 is connected to the ground. Then, just push the button, and if you see ones like on the following screenshot, it is all good.



Putting it All Together

It's now time to combine everything we've learned so far, and read out remotely the value of of this push button. Actually, we will do something slightly different: we will check if the button changed it's state (which mean for example that the front door of your flat has been opened), and put a message on a server if this happened. And the server of course, it will be the Arduino board with the WiFi shield.

First of all, we need to check if the button changed state. We could do as we did before, and use the digitalRead() function. But there is a problem with that function: what we will do is check the status of the button, which is fast, and then if something connects to the server running on our Arduino board, it will answer to this connection and reply with the state of the button. Sounds good to you ? Well … imagine now if the door is actually opened and closed again during the time the board is answering to the incoming connection: the board will completely miss it, and a thief for example is entering your home without you knowing it !

That's why I spoke about using pin 2 before: it is actually one of the *interrupt* pin of the Arduino board. What this means is that on this pin, you can attach an interrupt with the attachInterrupt() function. This will call a special function if at any time the state of this pin change, in our case going from low to high:

```
attachInterrupt(0, setDoorStatus, RISING);
```

Here, the function setDoorStatus will set a variable to indicate that the door has been opened. That was for the sensor part, now we have to actually make our board answer with the status of the sensor if something (in this case, your computer) access the board. After connecting the shield to the wireless network, the next step is to see if there is some incoming connection. This is done by doing:

```
WiFiClient client = server.available();
```

If this is the case, we start building a very simple HTML page by calling the println() function of client:

```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connnection: close");
client.println();
client.println("");
client.println("");
client.println("");
```

Then, we check the status of the variable door_opened, and print some information accordingly:

```
if (door_status == false){
  client.print("Everything is ok");
}
else {
  client.print("Alert ! The door has been opened");
}
```

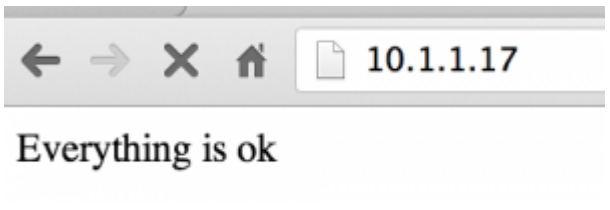
Finally, we finish the HTML page with:

```
client.println("
");
client.println("");
```

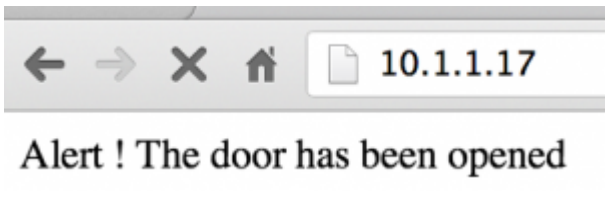
And then we stop the connection:

```
client.stop();
Serial.println("client disonnected");
```

These are the important parts of the code, as usual you can find the full code in our GitHub repository. Now, just upload the final code into the board, and wait until the “link” LED turns green on the board. Now, remember when I told you to write down the shield’s IP address ? This is where you will need it. Just open your favorite browser, and type down this IP into the URL field. This is what you should see:



Now, press the button, and return to the page. The message should have changed to this:



Which means our systems works ! You can now monitor what’s going on in your home directly from your computer. Of course, there would be some more adjustments to do so you can monitor it from anywhere in the world (right now it is limited to your own network), but you get the idea.