How to Use SONOFF with Node-RED

Random Nerd Tutorials – Rui Santos

How to Use SONOFF with Node-RED

In this guide you're going to learn how to use the SONOFF device with your home automation system. The SONOFF is a device that you put in series with your power lines allowing you to turn any device on and off.

First, you're going to install the default app that comes with the SONOFF device. Later, you'll learn how to integrate the SONOFF with Node-RED.

SONOFF Overview

In the figure below, you can see the SONOFF WiFi smart switch. **You can get a SONOFF for approximately $5.**

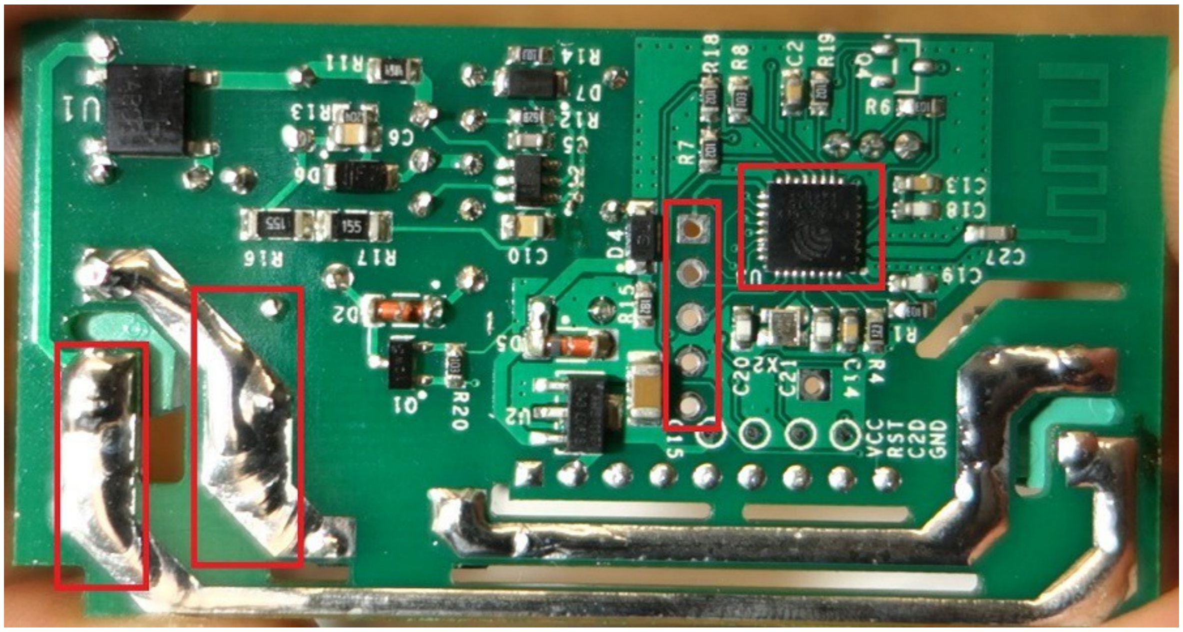
It's very simple, it has an input in one side and an output on the other side.

Then, you can simply send commands via WiFi to turn it on and off. That's pretty much how it works.

Opening the SONOFF

Let's look inside the SONOFF device. These are the main sections:

- There are the **two powerlines** and they are isolated from the rest of the circuit
- The **active line** goes to the **relay** (that's on the other side of the PCB)
- The **ESP8266**, which is the processor that provides WiFi and receives the control commands
- The SONOFF is meant to be hacked and you can see clearly that those **5 connections** were left out, so that you can solder some pins and upload a custom firmware

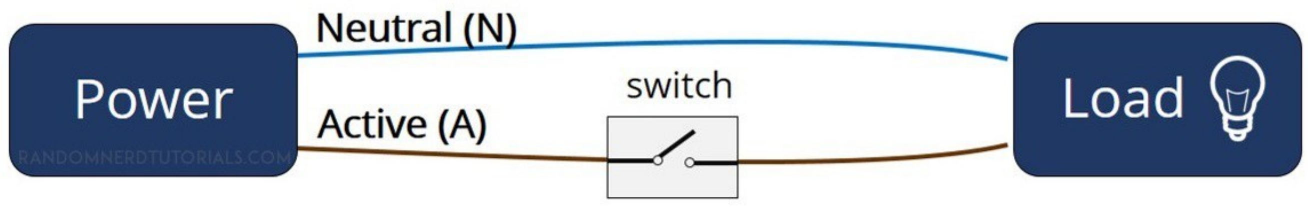


In the first part of this project you're going to use the standard firmware that comes with the SONOFF. Later, I'm going to show how to flash a custom firmware into the SONOFF device, so that you can integrate it with Node-RED.

SONOFF Example

Let's see how the SONOFF would fit in a normal circuit. Basically, you cut the wire that goes to the device, and you put the SONOFF in the middle, so that you can control any device that is connected on the other end.

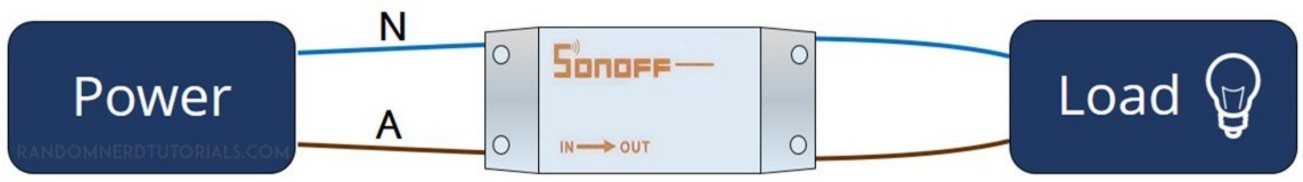
Normally, what you have is a power source that has an active and neutral line that goes to a load, your load can be lamp for example. In the middle, you usually have a switch.



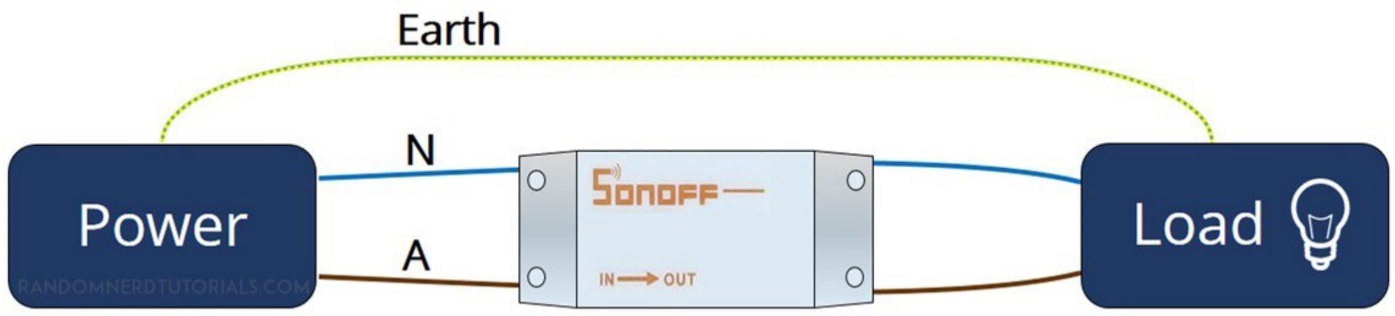
With the SONOFF, you cut that connection…

Power
RANDOMNERDTUTORIALS.COM
N
A
Load

And you place the SONOFF in the middle. The SONOFF acts as a switch that is controlled via WiFi.



Note: if you have an earth line, it has to go outside the SONOFF. In my case, I don't have earth in my home.



Safety Warning

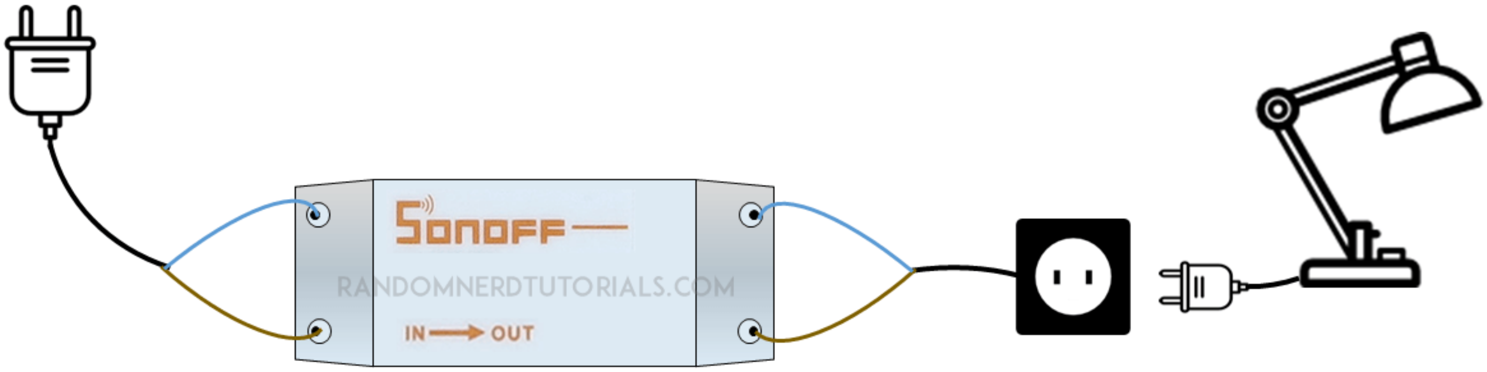
Before proceeding with this project, I want to let you know that you're dealing with mains voltage. Please read the safety warning below carefully.

SAFETY WARNING!

When you are making projects that are connected to mains voltage, you really need to know what you are doing, otherwise you may shock yourself. This is a serious topic and I want you to be safe. If you are not 100% sure what you are doing, do yourself a favor and don't touch anything. Ask someone who knows!

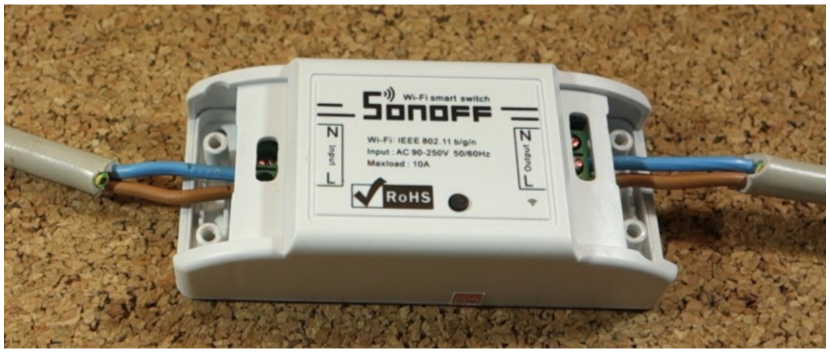
SONOFF Usage

Let's hook up the SONOFF. On the left side, you connect the active and neutral accordingly to the pinout. Active and neutral come out on the right.



On the left side, you have the input that connects to the outlet. The right side is the part that goes to your lamp/load.

Use your screwdriver to tighten the screws and have secure wire connection:



Place the two plastic protections and screw them.

After carefully checking all the connections, plug the male socket to the outlet.

On the other end, connect the female socket to the lamp.

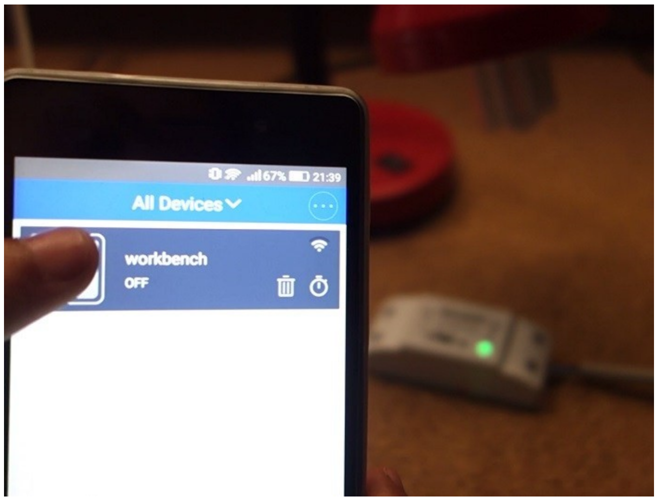
Installing the App

Now you have everything in place to install the app to control the light with your smartphone, follow these next instructions:

- Search for the app **eWeLink** (on the Play Store or App Store) and install it
- Open the app and create an account
- Power up the SONOFF device and connect the appliance that you want to control (in my case, it's a desktop lamp)
- Press and hold the SONOFF button for 5 seconds, so the green LED starts blinking

- Go to the app and press the next button
- Enter your network credentials and choose a name for your device
- Add it to your dashboard.

Refresh the dashboard and you should see your device. Press the on button to turn it on. If you press off, the lamps goes off.



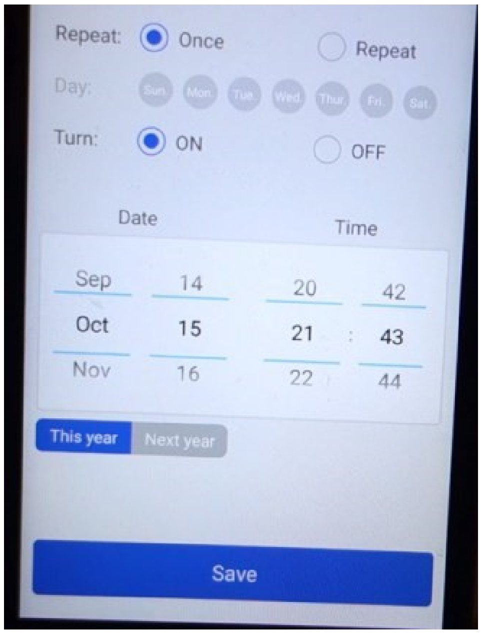
Watch the video to see a live demo of the Sonoff device:

https://youtu.be/mX97u_pQYnU.

Keep in mind that with this app you can control any device on and off from anywhere in the world, because it's controlled through the eWeLink cloud servers.

The app also comes with a nice set of features, click the timer button. You can add a timer that can be activated on a certain date and time.

I've tested this feature and it has been working flawlessly.


Repeat: [✓ Once] Repeat
Day: Sun Mon Tue Wed Thur Fri Sat
Turn: [✓ ON] OFF
Date: Oct 15 — Time: 21 : 43
This year | Next year
Save

Integrate SONOFF with Node-RED

To use the SONOFF with Node-RED, you have to flash custom firmware in the SONOFF device, so that you can control it through the Node-RED Dashboard with the MQTT protocol.

You should have followed the main course "Build a Home Automation System for $100" and you should have your Raspberry Pi running the Node-RED software.

So, let's get started.

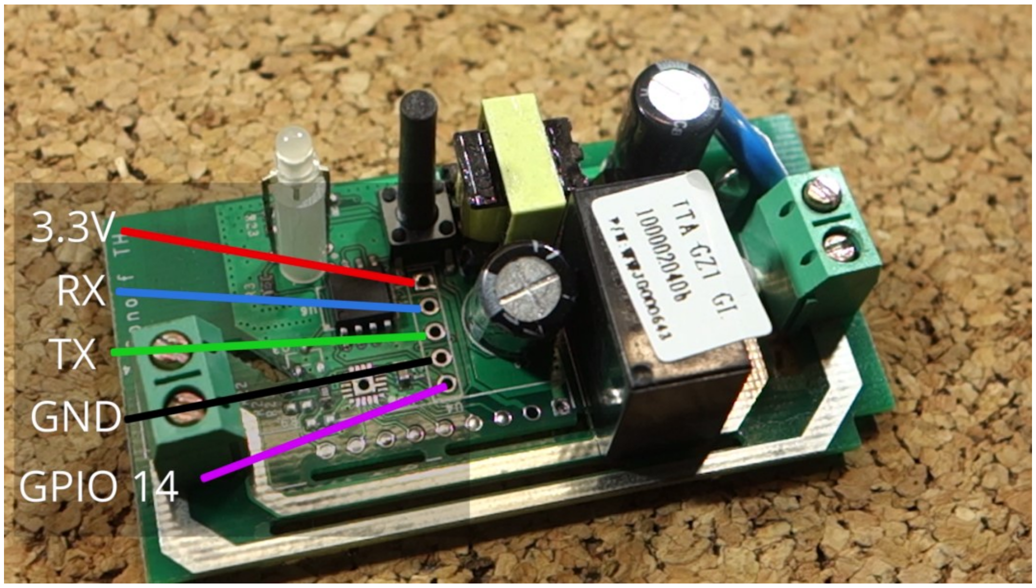
Safety Warning

Make sure you disconnect your SONOFF from mains voltage while flashing a custom firmware. Don't touch any wires that are connected to mains voltage.

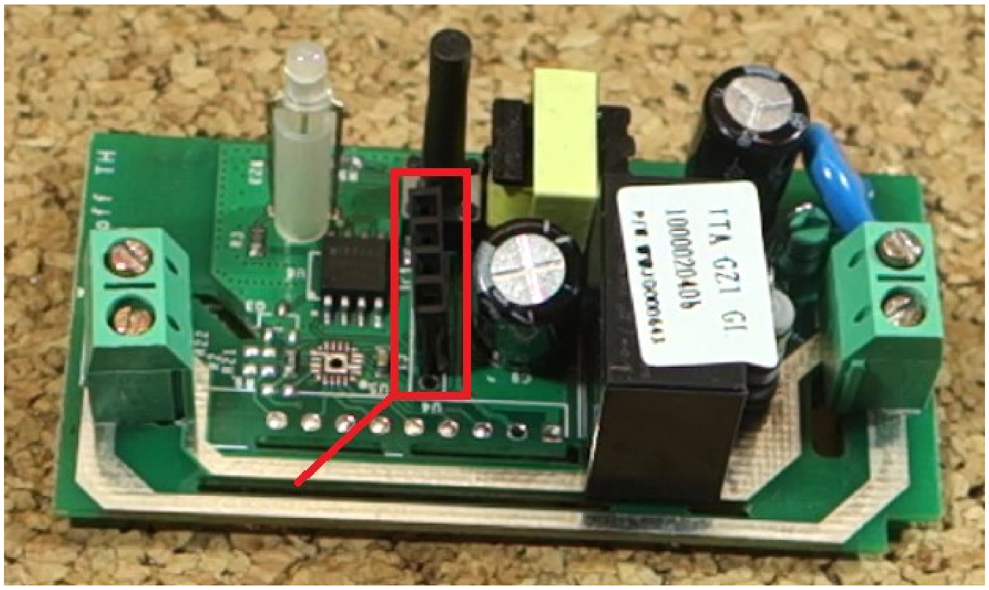
SONOFF Pinout

Open the SONOFF enclosure. As I mentioned earlier, the SONOFF is meant to be hacked, and you can see clearly that s connections were left out, so that you can solder some pins and upload a custom firmware.

That's the pinout that you need to worry about.



I solder 4 header pins, so that I can easily connect and disconnect wire cables to my SONOFF device.



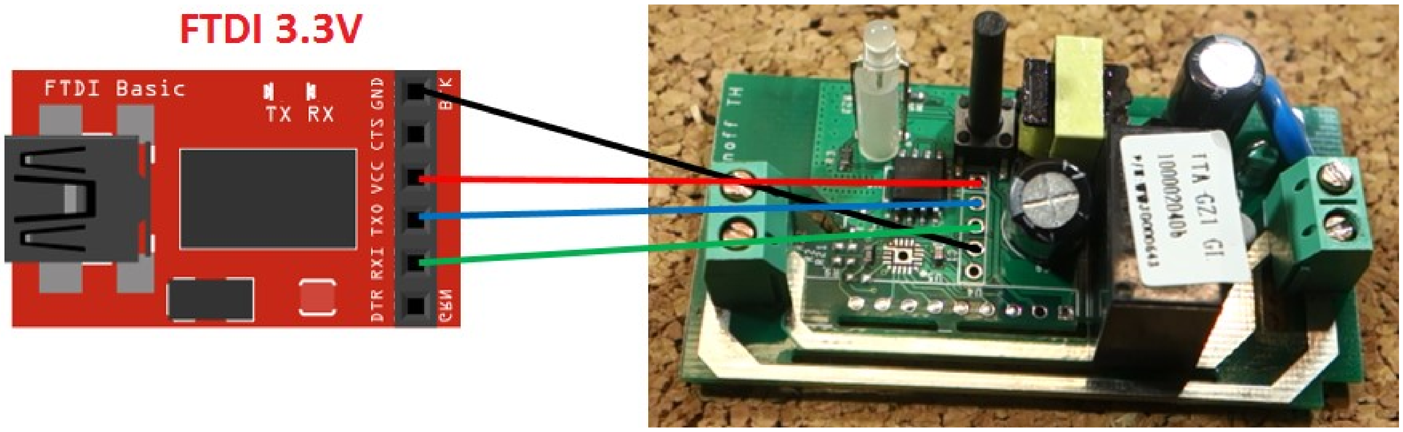
You need an FTDI module to upload a new firmware to your SONOFF.

Note: uploading a custom firmware is irreversible and you'll no longer be able to use the app eWeLink.

Connecting the FTDI to Your SONOFF

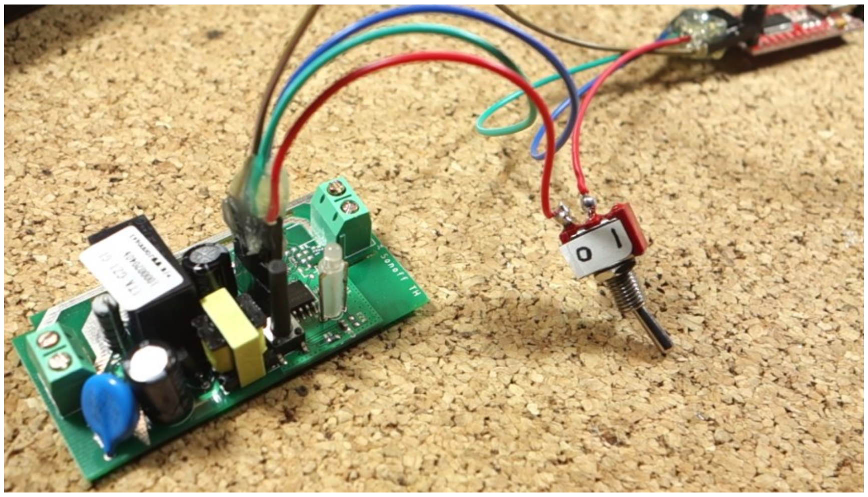
You should follow these next schematics to connect your FTDI programmer to your SONOFF device.

- 3.3V –> 3.3V
- TX -> RX
- RX -> TX
- GND -> GND



I've added a toggle switch in the power line (3.3V), so that I can easily turn the SONOFF on and off to flash a new firmware without having to unplug the FTDI module.

Finally, connect the SONOFF to the FTDI, and connect them via USB to your computer to upload the new firmware.

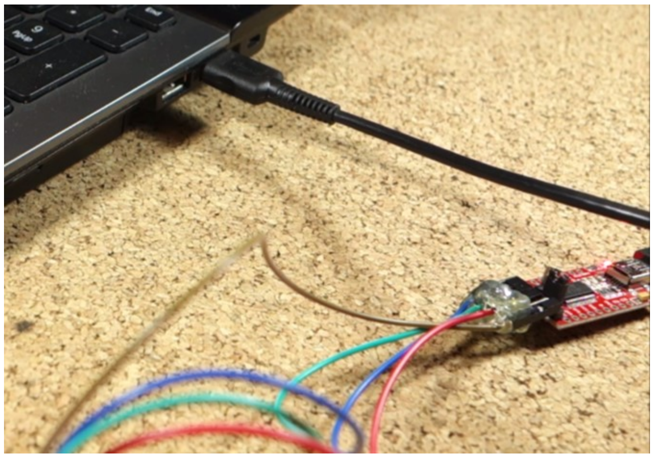


If you look closely to the previous figure, I used hot glue to glue the ends of the wires together. This prevents you to make wrong connections between the FTDI and the SONOFF in the future.

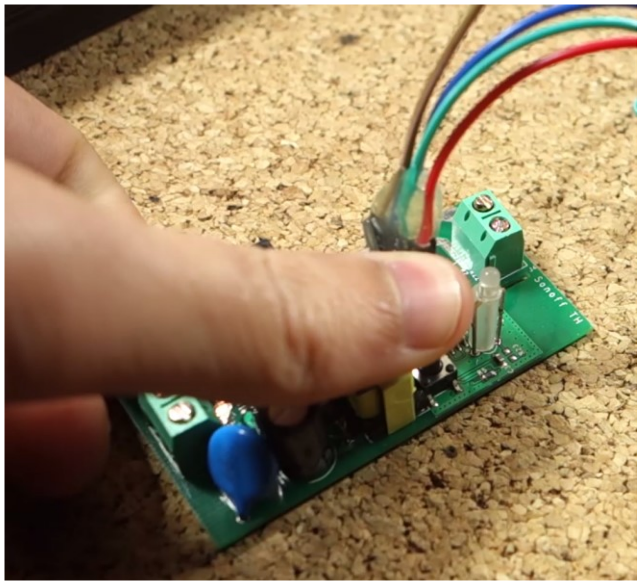
Boot your SONOFF in Flashing Mode

To flash a new firmware to your SONOFF, you have to boot your SONOFF in flashing mode. Follow this 4 step process:

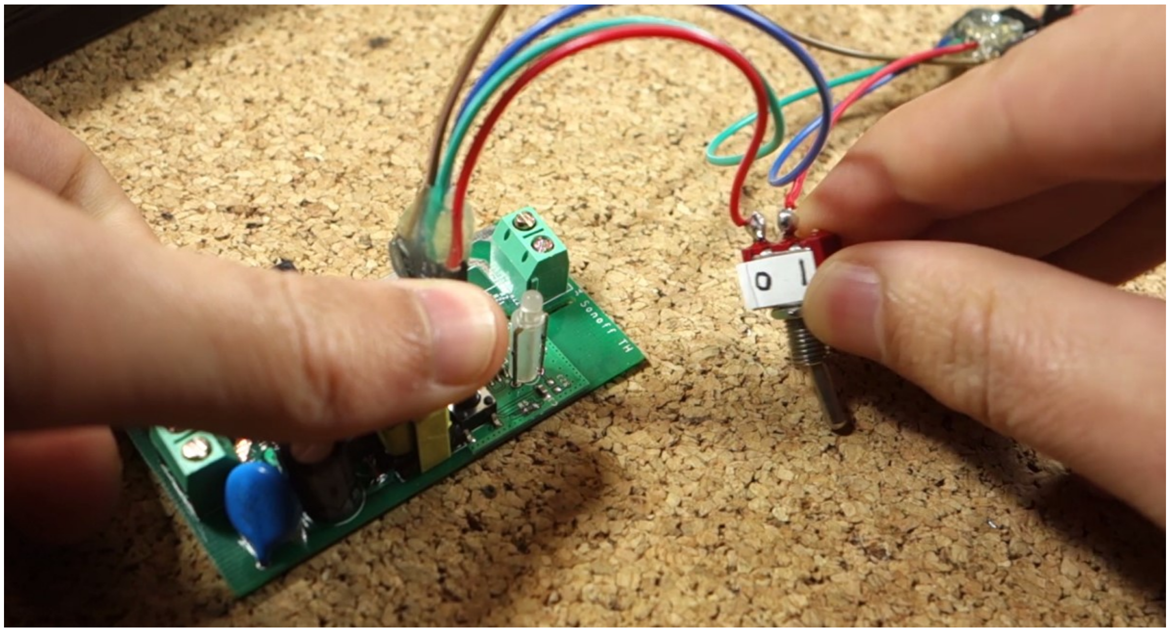
1) Connect your 3.3V FTDI programmer to your computer



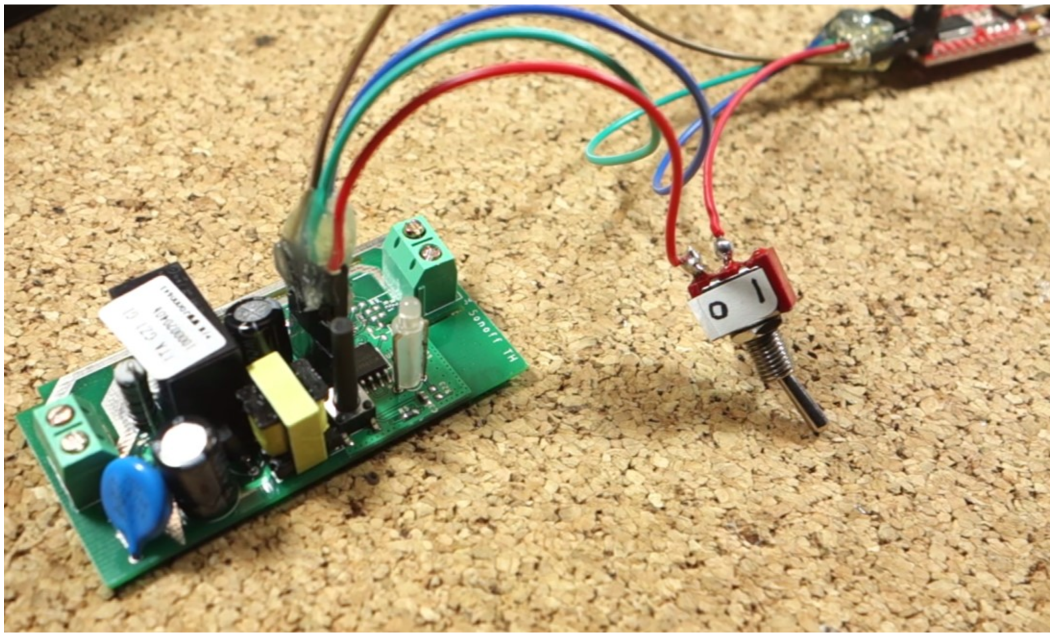
2) Hold down the SONOFF button



3) Toggle the switch to apply power to the SONOFF circuit



4) Then, you can release the SONOFF button



Now, your SONOFF should be in flashing mode.

Opening the Arduino IDE

Having completed or followed the main course "Build a Home Automation System for $100", you should have the ESP8266 add-on installed in the Arduino IDE. If you don't have, follow this tutorial on How to Install the ESP8266 Board in Arduino IDE.

Uploading the Sketch

Finally, open your Arduino IDE. You can upload the full sketch to your SONOFF (replace with your SSID, password and RPi IP address):

DOWNLOAD SOURCE CODE

https://github.com/RuiSantosdotme/Home-Automation-Course/blob/master/code/v2/sonoff_with_node_red.ino

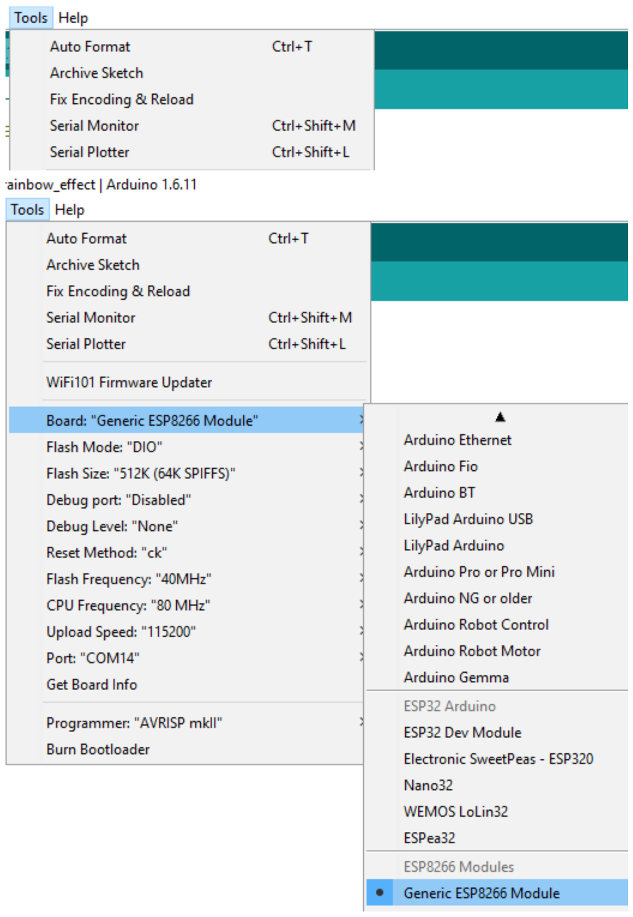
```
/*****

 All the resources for this project:
 https://rntlab.com/

*****/


// Loading the ESP8266WiFi library and the PubSubClient library
#include <ESP8266WiFi.h>
#include <PubSubClient.h>


// Change the credentials below, so your ESP8266 connects to your router
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";


// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "YOUR_RPi_IP_Address";


// Initializes the espClient
WiFiClient espClient;
PubSubClient client(espClient);
```

```
// GPIOs of your ESP8266 on your SONOFF
int gpio13Led = 13;
int gpio12Relay = 12;


// Don't change the function below. This functions connects your ESP8266 to your router
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}


// This functions is executed when some device publishes a message to a topic that your
ESP8266 is subscribed to
// Change the function below to add logic to your program, so when a device publishes a
message to a topic that
// your ESP8266 is subscribed you can actually do something
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();


  // Feel free to add more if statements to control more GPIOs with MQTT
  // If a message is received on the topic home/office/sonoff1, you check if the message is
either 1 or 0. Turns the ESP GPIO according to the message
  if(topic=="home/office/sonoff1"){
      Serial.print("Changing Sonoff to ");
      if(messageTemp == "on"){
        digitalWrite(gpio13Led, LOW);
        digitalWrite(gpio12Relay, HIGH);
        Serial.print("On");
      }
      else if(messageTemp == "off"){
        digitalWrite(gpio13Led, HIGH);
        digitalWrite(gpio12Relay, LOW);
        Serial.print("Off");
      }
  }
  Serial.println();
}


// This functions reconnects your ESP8266 to your MQTT broker
// Change the function below if you want to subscribe to more topics with your ESP8266
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    /*
     YOU MIGHT NEED TO CHANGE THIS LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE
CONNECTIONS
     To change the ESP device ID, you will have to give a new name to the ESP8266.
     Here's how it looks:
       if (client.connect("ESP8266Client")) {
     You can do it like this:
       if (client.connect("ESP1_Office")) {
     Then, for the other ESP:
       if (client.connect("ESP2_Garage")) {
      That should solve your MQTT multiple connections problem
    */
    if (client.connect("sonoff1")) {
      Serial.println("connected");
      // Subscribe or resubscribe to a topic
      // You can subscribe to more topics (to control more Sonoffs)
      client.subscribe("home/office/sonoff1");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


// The setup function sets your ESP GPIOs to Outputs, starts the serial communication at a
baud rate of 115200
// Sets your mqtt broker and sets the callback function
// The callback function is what receives messages and actually controls the LEDs
void setup() {
  // preparing GPIOs
  pinMode(gpio13Led, OUTPUT);
  digitalWrite(gpio13Led, HIGH);

  pinMode(gpio12Relay, OUTPUT);
  digitalWrite(gpio12Relay, HIGH);

  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}


// For this project, you don't need to change anything in the loop function.
// Basically it ensures that you ESP is connected to your broker
void loop() {


  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("sonoff1");
}
```

Preparing your Arduino IDE

Having your SONOFF device still in flashing mode.

1. Select your FTDI port number under the **Tools** > **Port** > **COM14** (in my case)
2. Choose your ESP8266 board from **Tools** > **Board** > **Generic ESP8266 Module**
3. Press the Upload button



Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Port: "COM14"
Programmer: "AVRISP mkII"

Wait a few seconds while the code is uploading. You should see a message saying "Done Uploading".

Troubleshooting

If you try to upload the sketch and it prompts the following error message:

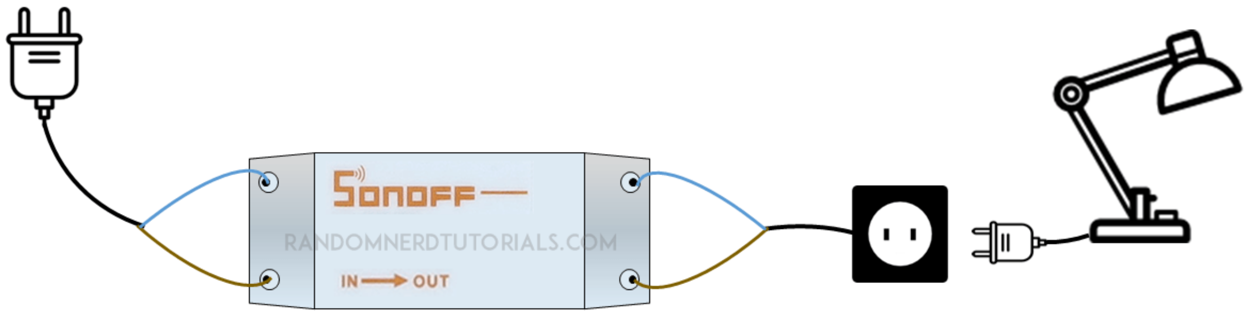
```
warning: espcomm_sync failed
error: espcomm_open failed
```

Your SONOFF is not in flashing mode and you have to repeat the process described in section "Boot your SONOFF in flashing mode" described earlier in this guide.

Final Circuit

After uploading the code, re-assemble your SONOFF. Be very careful with the mains voltage connections.

It's the exact same procedure as shown earlier in this guide:

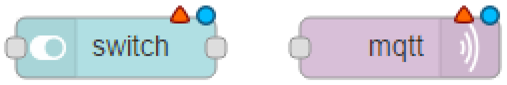


How it should look:

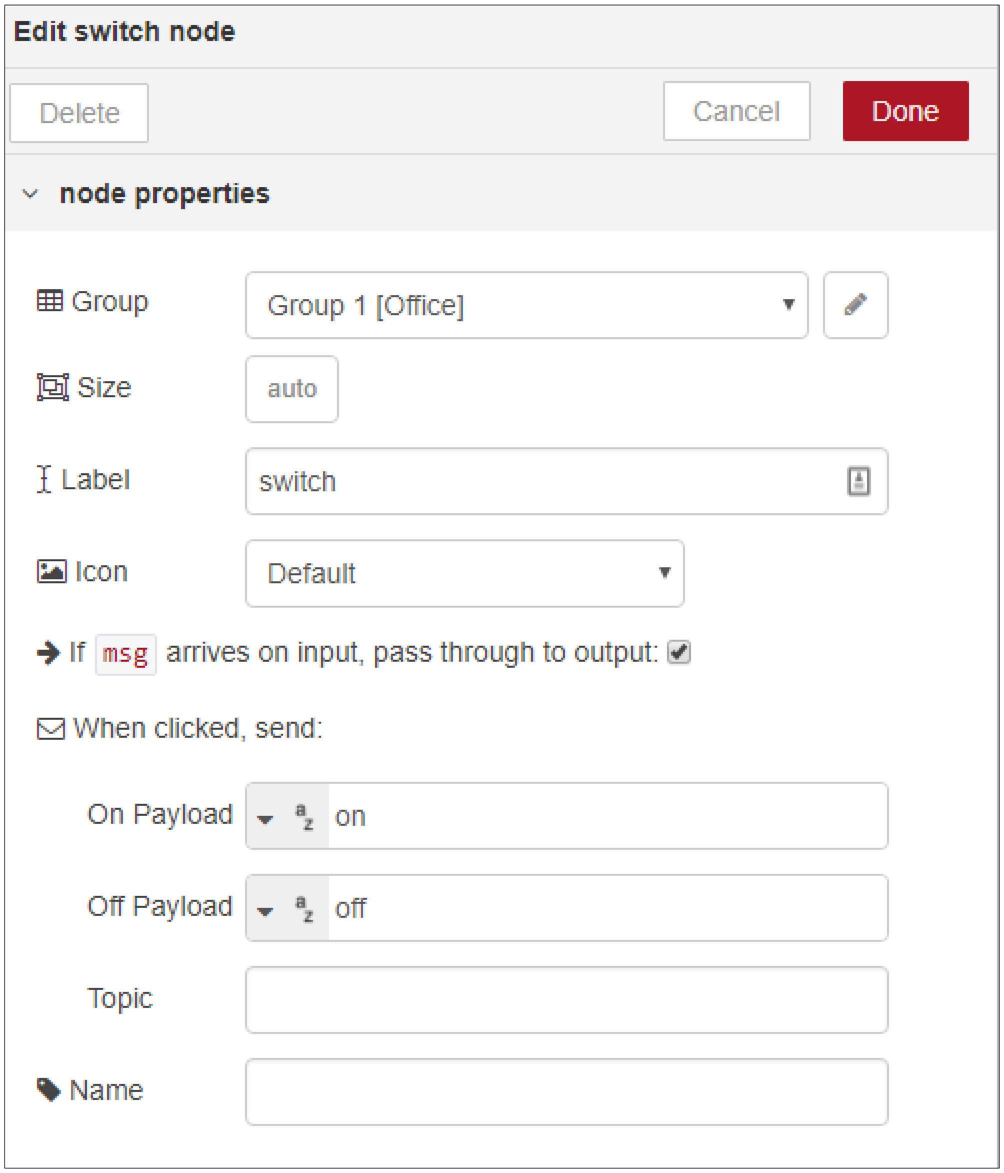
Creating the Flow

In this flow, you're going to use a switch node that when pressed triggers an MQTT out node to turn the SONOFF on and off.

Follow these next 5 steps to create your flow:



1 – Drag 2 Nodes



Group: Group 1 [Office]

Size: auto

Label: switch

Icon: Default

If msg arrives on input, pass through to output: ✓

When clicked, send:

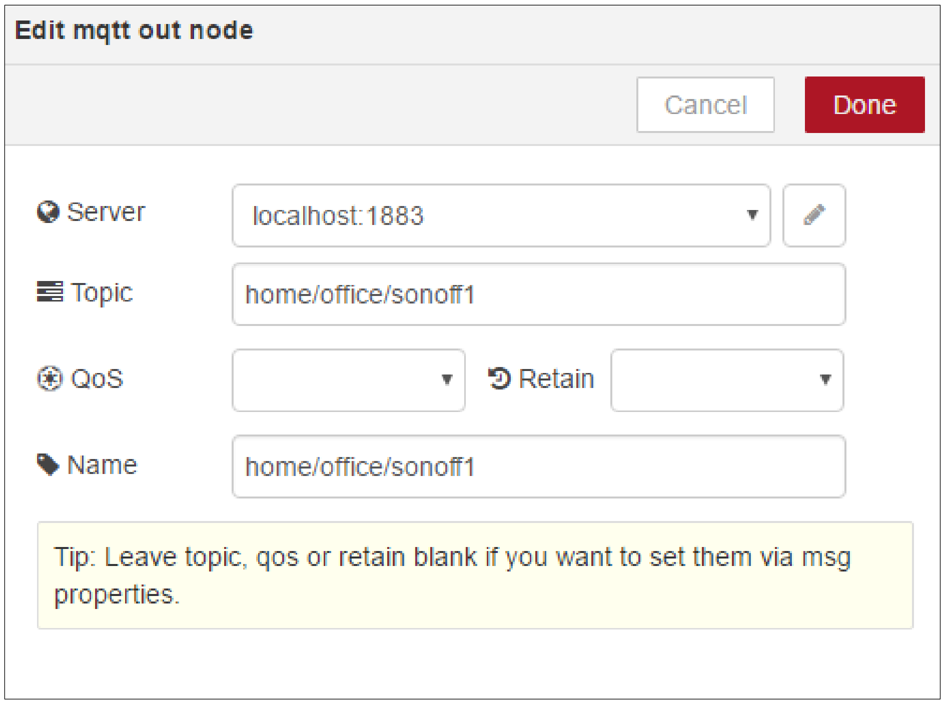
On Payload: on

Off Payload: off

Topic:

Name:

2 – Switch node


Edit mqtt out node

Cancel Done

Server: localhost:1883

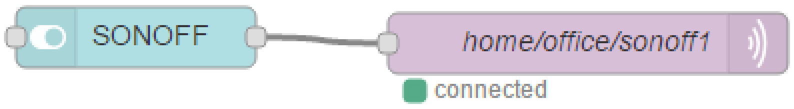
Topic: home/office/sonoff1

QoS: Retain:

Name: home/office/sonoff1

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

3 – MQTT out sonoff1

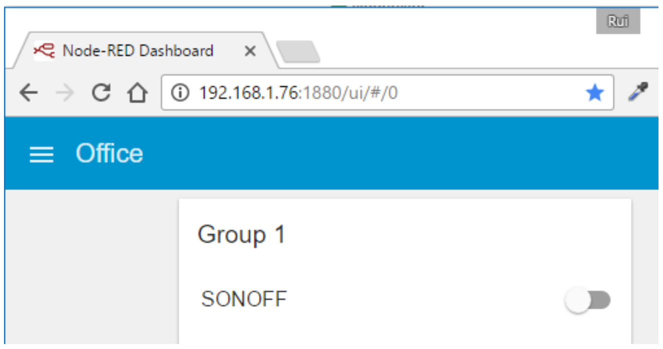


4 – Connecting Nodes

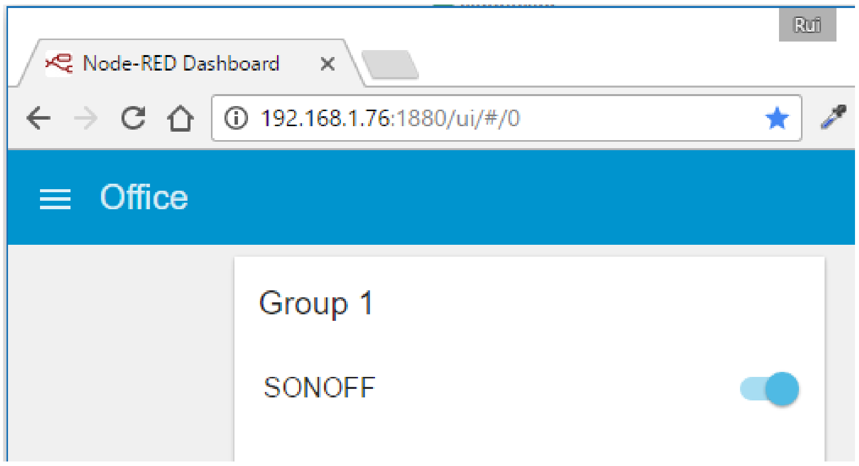
5 – Deploy your application

Testing Your Flow

When you go to the Node-RED Dashboard tab, here's what you should see:



If you press the on Switch:



The Lamp or any device that is connected to your SONOFF should turn on: