



FABLIMP

created by Eduardo Chamorro Martin, Santi Fuentemilla updated by Miriam Choi

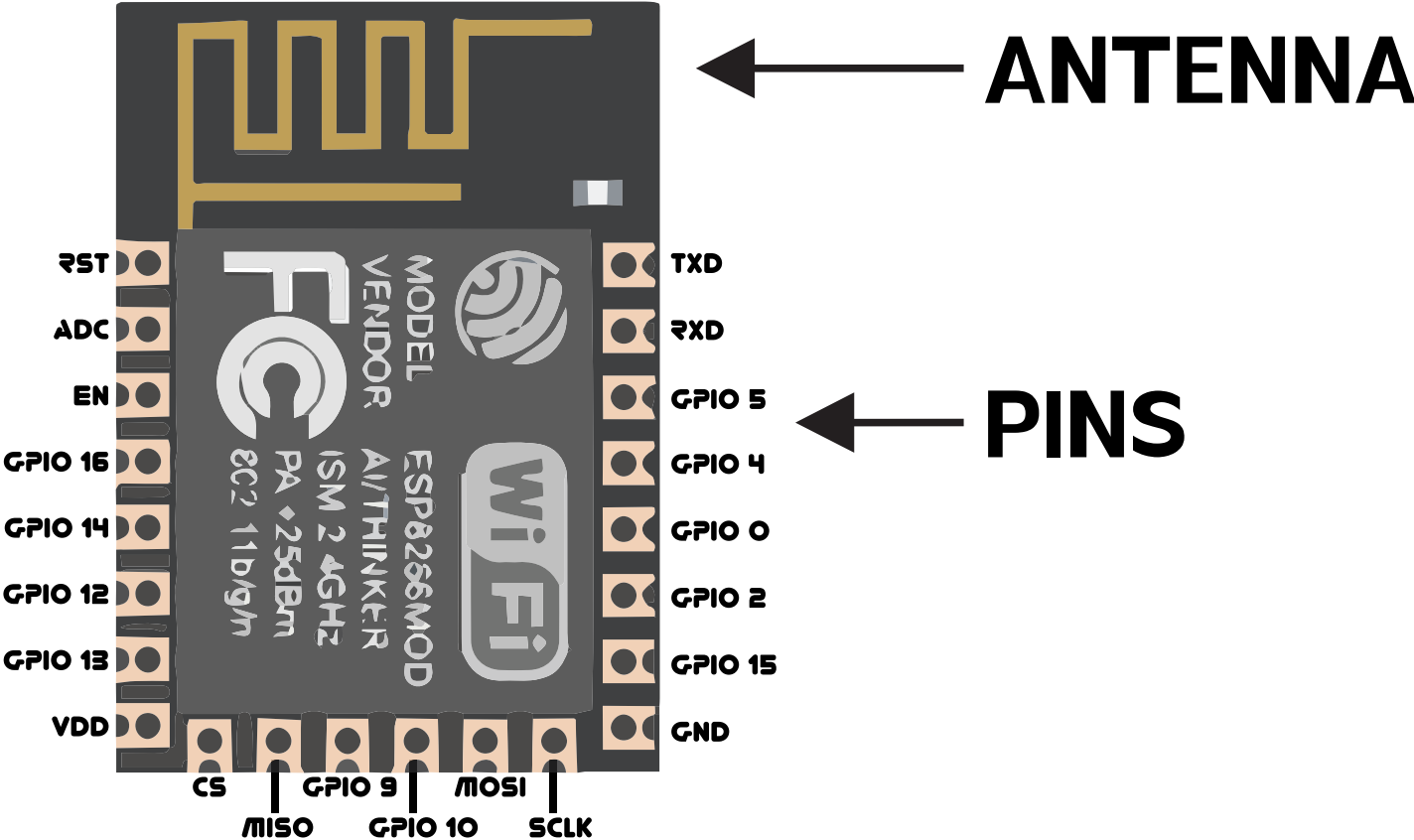
FABLIMP

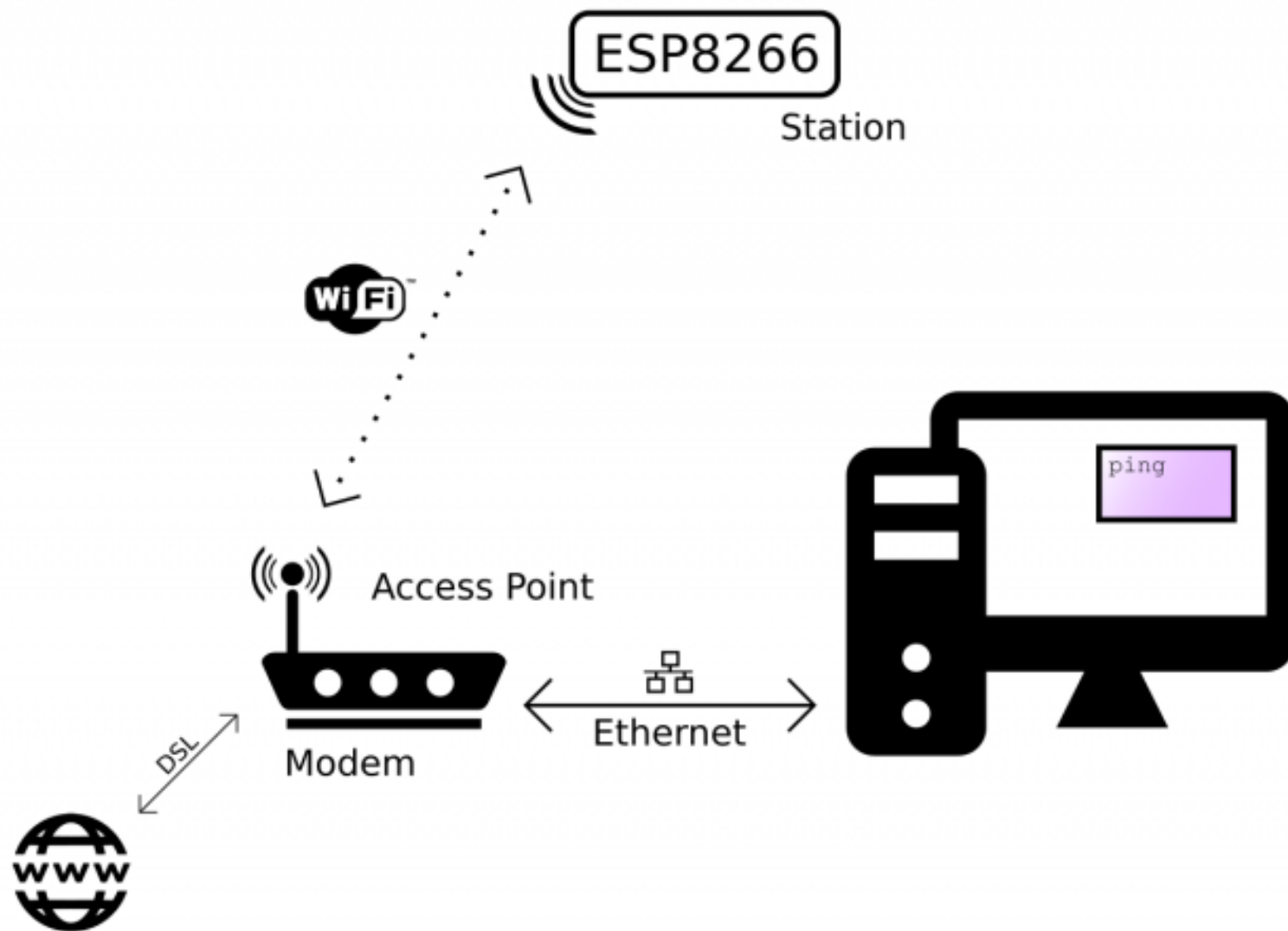
1. USING ESP 8266

2. HOW TO SOLDER

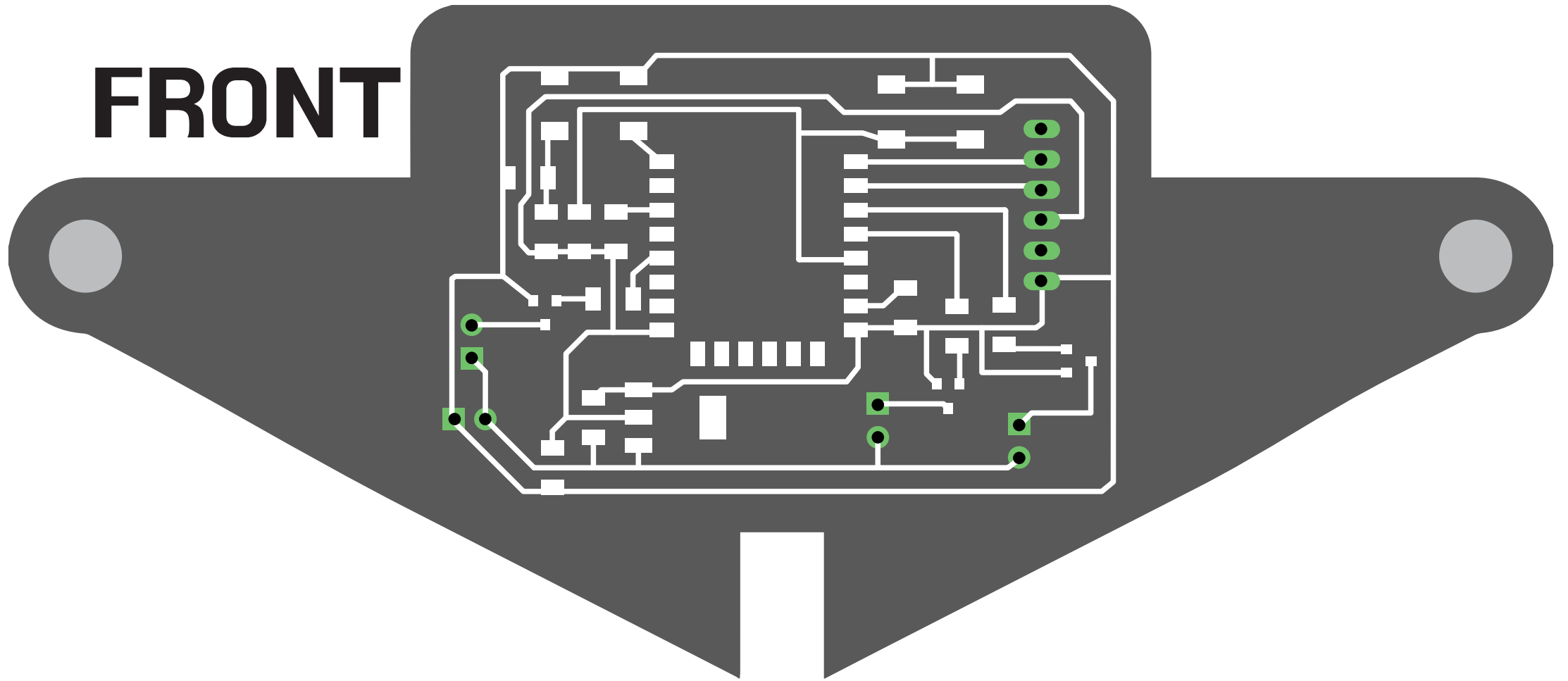
3. UPLOADING THE CODE

ESP 8266 12-e

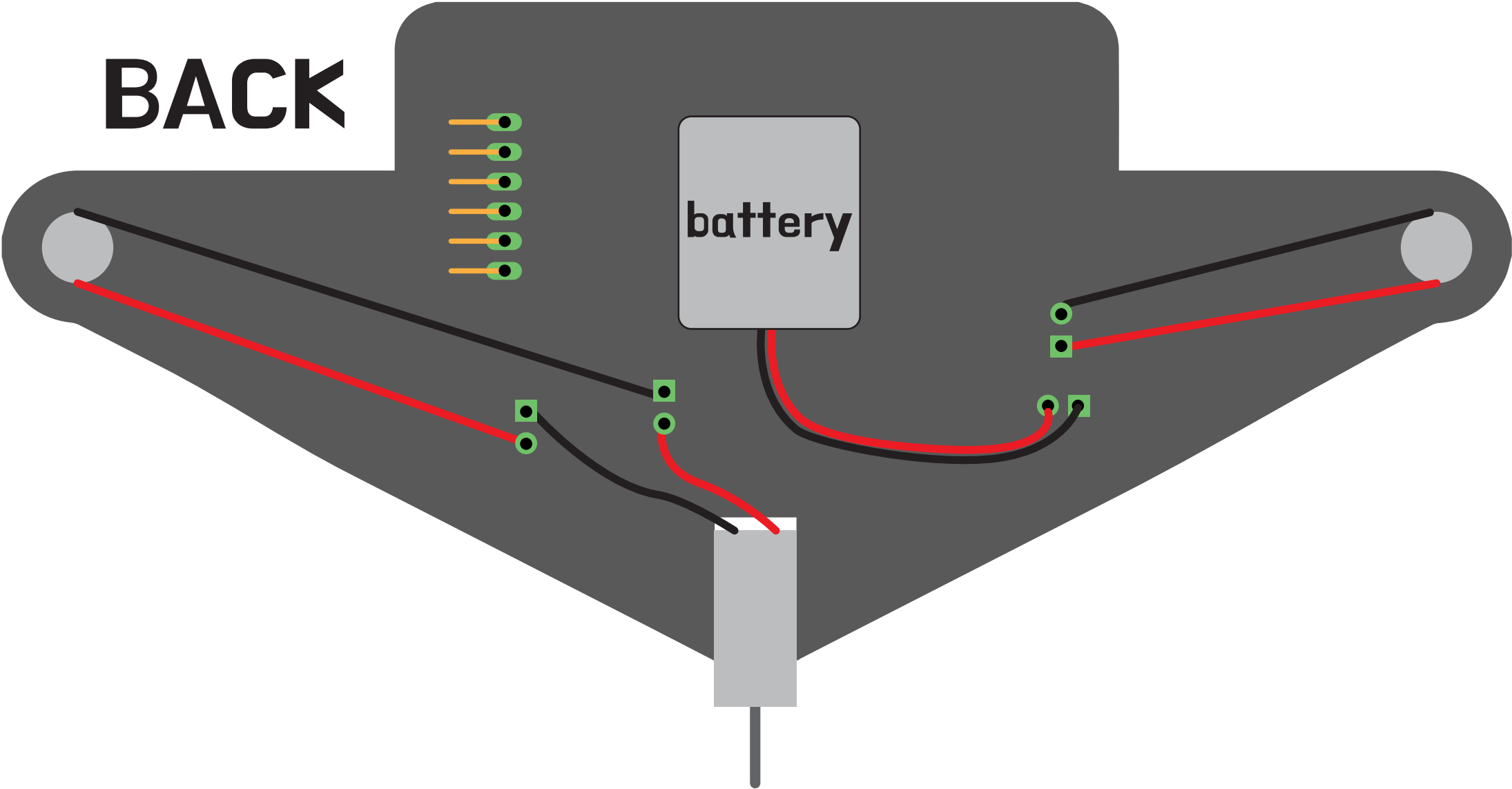




FRONT



BACK



COMPONENTS

3 X 100 OHM
4 X 10K OHM

2 X 4.7 uf
1 X 10 uf

3 X N Channel 30v 1.7 Amp (MOSFET)

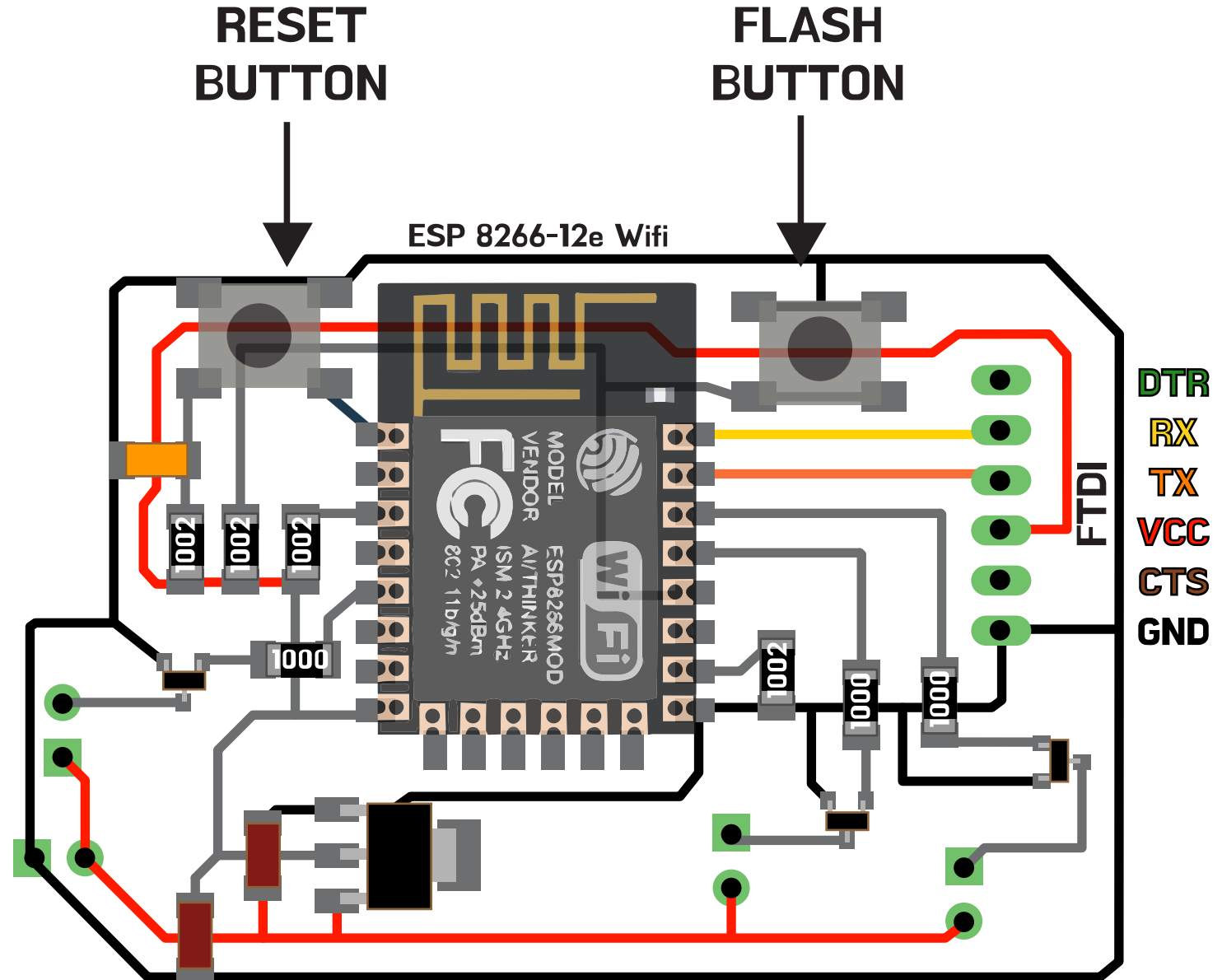
2 X Switch Button

1 X IC Regulator 3.3 v

1 X ESP 8266- 12e

1 X 3.7v battery 220~300 mAh size

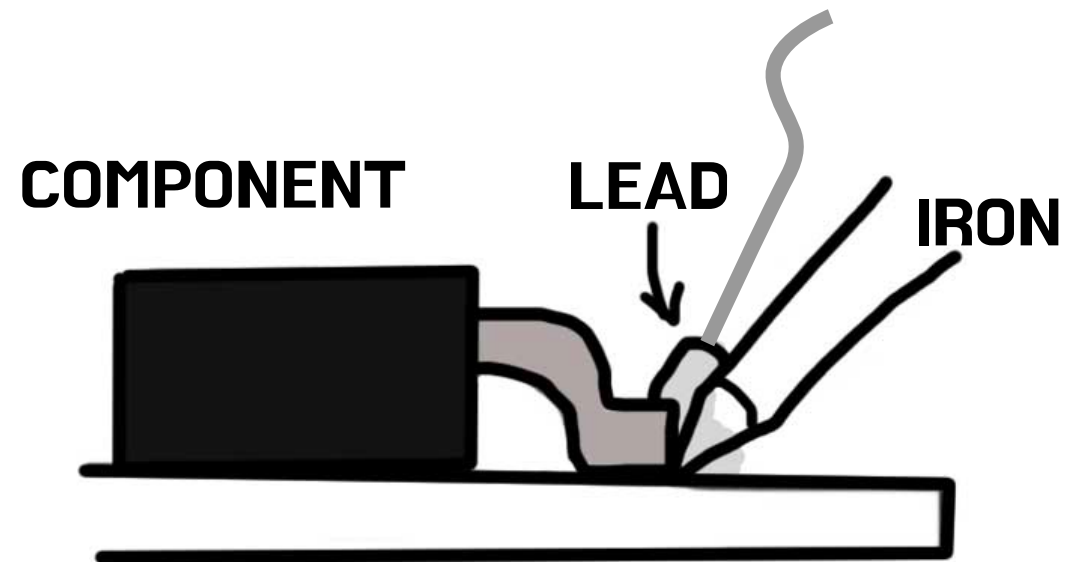
3 X tiny DC motors (and fan wings)



HOW to SOLDER!

*** make sure that the soldering iron is clean and hot**

- 1. Put the soldering iron on the copper.**
- 2. Count to 5.**
- 3. Put the lead wire to the iron.**



SOLDERING THROUGH-HOLE



**NO
Connection**



**NOT ENOUGH
Solder**



**TOO MUCH
Solder**



Burnt



Good job

Coding



WIFI

(Connects to the wifi)

```
// -----  
// Input your WiFi SSID and Password here  
// -----  
const char* ssid = "MyFabLab";  
const char* password = "012345678";
```

Good: Long range. quick response.

Bad: Code needs to be changed with change of wifi. Relies on internet availability.

Hotspot

(Making the Blimp have its own hotspot)

```
// -----  
// Set your WiFi SSID and Password here  
// -----  
const char* ssid = "MyBlimpName";  
const char* password = "012345678";
```

Good: Only need to upload the code once, does not rely on internet availability.

Bad: Less range, slower response.

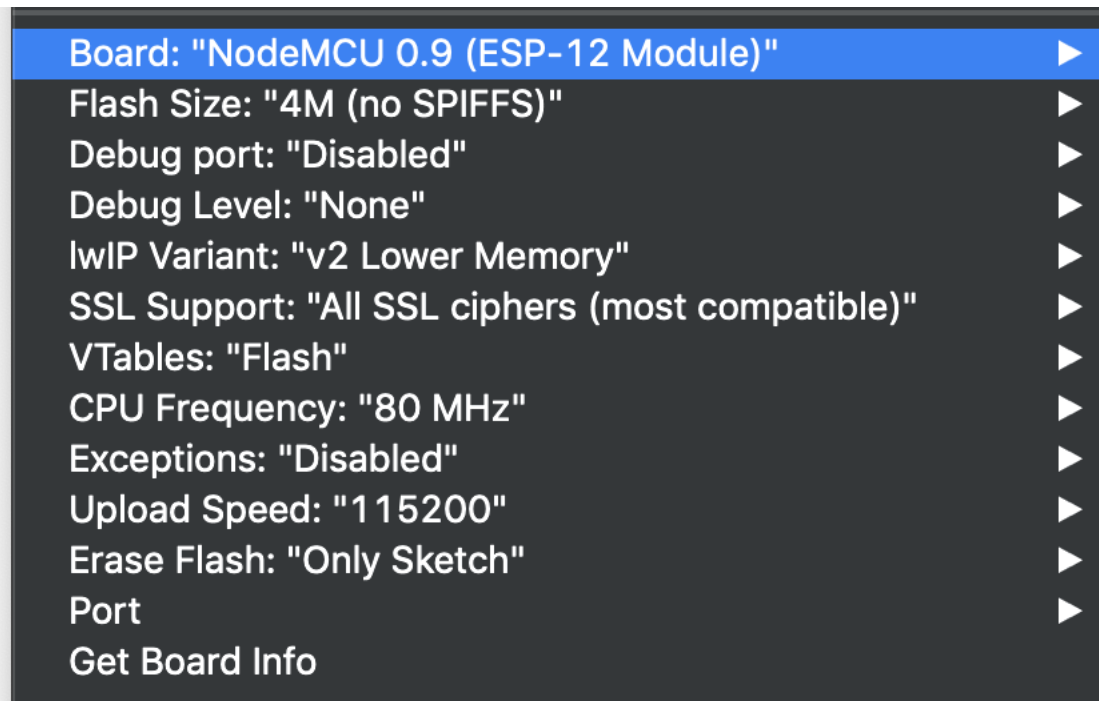
Uploading the code

Download the Arudino Library for ESP 8266 from:

<https://github.com/esp8266/Arduino>

Uploading the code

Load the **Library** for **ESP 8266** and set the board to **NodeMCU**.
Connect the **FTDI Cable** and set the **Port** to your **FTDI**.



After compiling the sketch, click “Upload”
Push both **FLASH** and **RESET BUTTONS** down.

```
Uploading...
Warning: Board arduino:avr:fabduino doesn't define a 'build.board' preference. Auto-set to: AVR_FABDUINO
Sketch uses 320440 bytes (30%) of program storage space. Maximum is 1044464 bytes.
Global variables use 33440 bytes (40%) of dynamic memory, leaving 48480 bytes for local variables. Maximum
esptool.py v2.6
2.6
esptool.py v2.6
Serial port /dev/cu.usbserial-FTARZEVE
Connecting....._____
n, Disabled, All SSL ciphers (most compatible), 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.usbserial-FTARZEVE
```

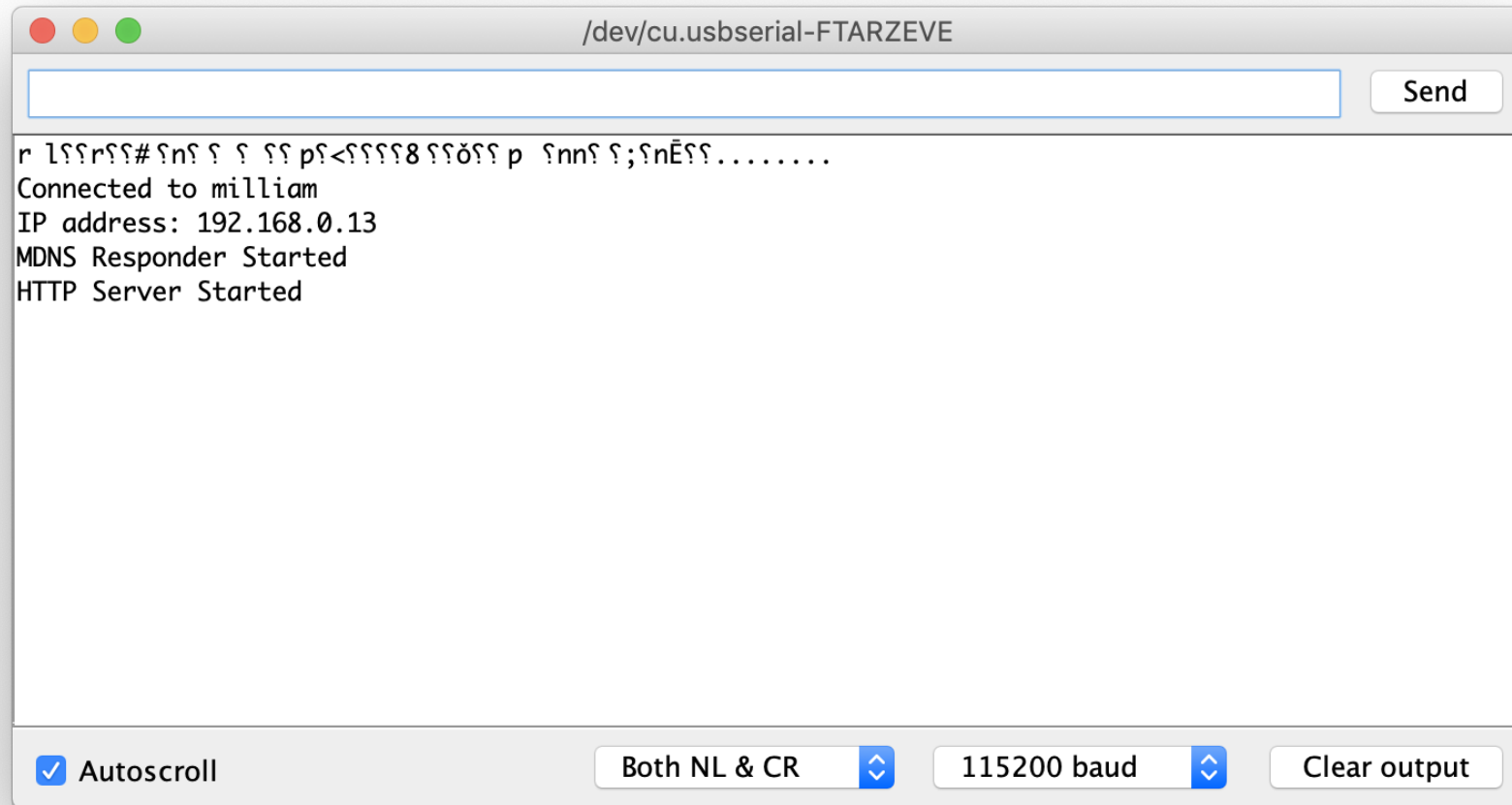
When it says **“Connecting.....”**, let go of the **RESET BUTTON**

```
Done uploading.  
Writing at 0x0000c000... (28 %)  
Writing at 0x00010000... (35 %)  
Writing at 0x00014000... (42 %)  
Writing at 0x00018000... (50 %)  
Writing at 0x0001c000... (57 %)  
Writing at 0x00020000... (64 %)  
Writing at 0x00024000... (71 %)  
Writing at 0x00028000... (78 %)  
Writing at 0x0002c000... (85 %)  
Writing at 0x00030000... (92 %)  
Writing at 0x00034000... (100 %)  
Wrote 324592 bytes (228540 compressed) at 0x00000000 in 20.3 seconds (effective 128.2 kbit/s)  
Hash of data verified.  
  
Leaving...  
Hard resetting via RTS pin...  
  
n, Disabled, All SSL ciphers (most compatible), 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.usbs
```

You may let go of both buttons once it is done uploading.

****After uploading, the motor might still be activated. just press the reset button once.**

**Open up your Serial monitor (keep FTDI in) and let it load.
get the IP address for the controller website and open it on your device.**



The screenshot shows a macOS Serial Monitor window titled "/dev/cu.usbserial-FTARZEVE". The window contains a text area with the following output:

```
r 1??r??#?n? ? ? ?? p?<????8??ö?? p ?nn? ?;?nË??.....  
Connected to milliam  
IP address: 192.168.0.13  
MDNS Responder Started  
HTTP Server Started
```

At the top of the window is a text input field and a "Send" button. At the bottom, there are control buttons: "Autoscroll" (checked), "Both NL & CR" (dropdown), "115200 baud" (dropdown), and "Clear output".

****Make sure that your device is connected to the same wifi as your blimp**



ENJOY!!!