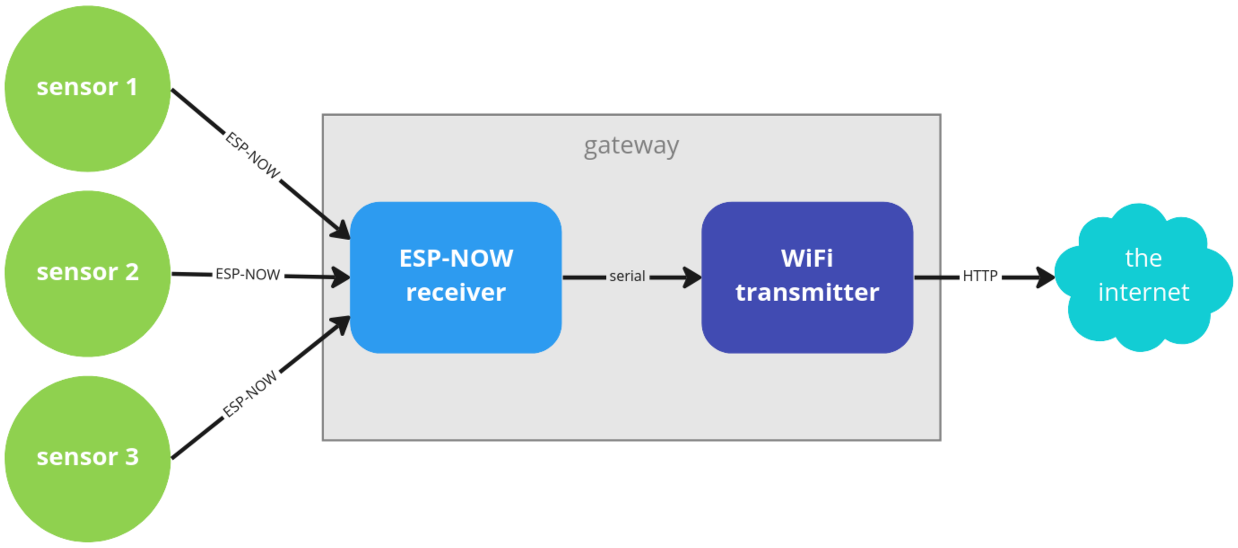
ESP-NOW to WiFi Gateway for ESP8266

salvatorelab.com/2023/02/esp-now-to-wifi-gateway-for-esp8266

Salvatore Archivo del Autor 10 de febrero de 2023

ESP-NOW is a **much more energy efficient** protocol compared to regular WiFi that can make your **battery-powered ESP devices** run for months or even years. The downside is that we usually want to send data to servers that don't speak ESP-NOW. So here is a solution: a gateway that takes ESP-NOW messages and sends them over WiFi, for example as HTTP requests to an API.



ESP-NOW to WiFi – Gateway Diagram

This project was inspired by the ESP-NOW to MQTT gateway video (and code) by MrDIY. But I have adapted it to HTTP and also changed the message structure so it works for all kinds of messages.

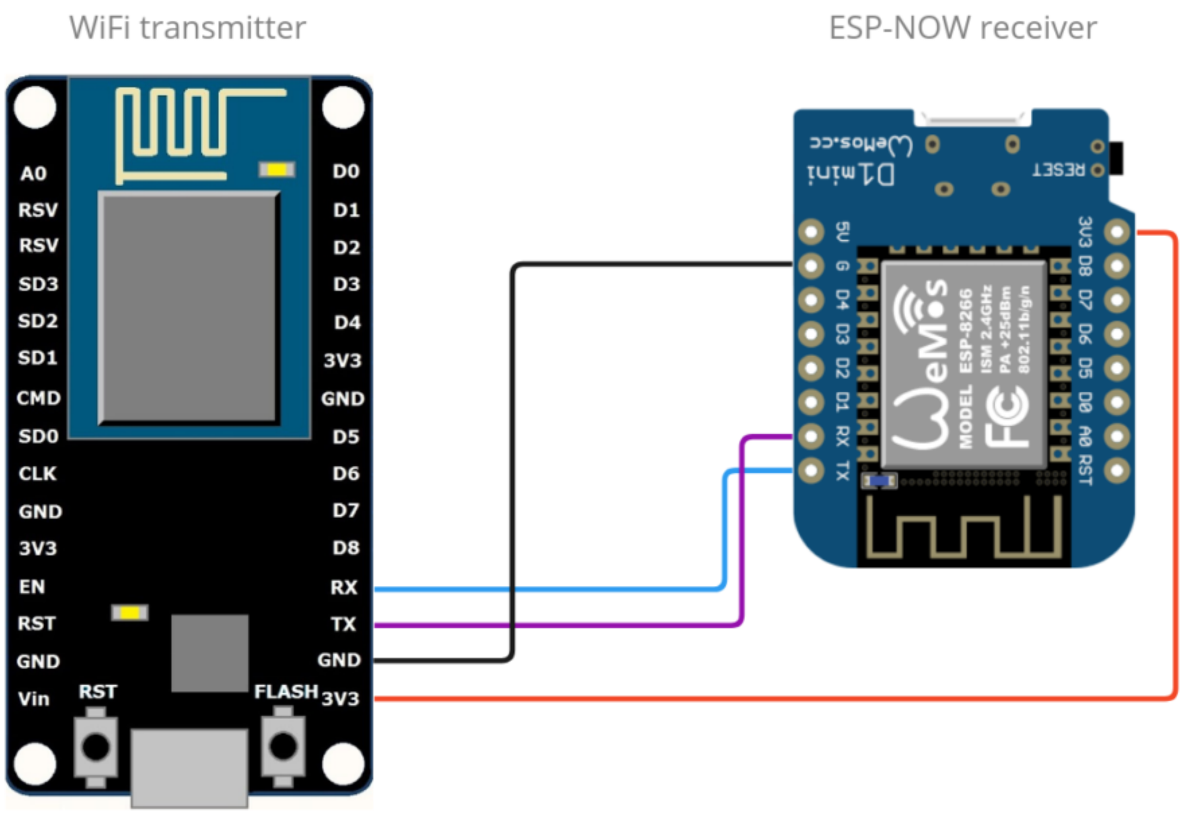
The **gateway has two ESP8266 boards**. You could use just one but it requires setting a fixed WiFi channel and all sensors knowing that channel which isn't ideal. But since ESPs are dirty cheap I think using two is better.

I will be using a Wemos D1 mini pro as the ESP-NOW receiver and a NodeMCU v1 as the WiFi transmitter. You can use whatever you have but **it is very important that the ESP-NOW receiver has an external antenna**. I tried using other Wemos boards that have an internal antenna and the range was ridiculous, only a few centimeters. But with this setup the range is pretty decent, like 10 meters with multiple walls in between, and your sensors don't need external antennas, just the receiver.

Connections

The connection between the boards is pretty straightforward: RX on one board connected to TX and the other and vice versa. And then for power, the NodeMCU will be connected via USB and then the Wemos will get power from the 3.3V pin. Ground to ground and that's it.

Didn't find a Wemos D1 mini pro for the diagram so I used a D1 mini.



ESP-NOW wiring diagram between NodeMCU and Wemos D1 mini

Code

Below you will find the code for the three main components of this project:

1. An ESP-NOW **sender**: could be any sensor, this is just an example so you can test that everything works and then adapt it to your needs.
2. The ESP-NOW **receiver**: should be fine as is, but feel free to enter a different MAC address if you want.
3. The WiFi **transmitter**: you will need to change the WiFi credentials and the API details. And if you want to use MQTT instead of HTTP you should be able to adapt it easily.

ESP-NOW Sender

An example of what a sender would look like, in my case it is a temperature and humidity sensor but I've removed that part for simplicity and instead I just send fixed values every 30 seconds.

I use a somewhat generic message that consists of a couple strings:

- a sensor id: any string to identify the sensor that is sending the message, for example «test»
- a payload: it can be any string, I will be using JSON

I limited the total message size to 200 bytes as that is (apparently) the max ESP-NOW supports on ESP8266. It is 250 bytes for ESP32.

```
#include <ESP8266WiFi.h>
#include <espnow.h>

uint8_t receiverAddress[] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC};
char sensor_id[20] = "test";
const char* payload = "{\"temperature\":%.02f,\"humidity\":%.02f}";

typedef struct sensor_message {
  char id[20];
  char payload[180];
} sensor_message;
sensor_message myMessage;

void setup() {
  Serial.begin(115200);
  Serial.println();

  WiFi.mode(WIFI_STA);

  setupEspNow();
}

void setupEspNow() {
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    delay(1000);
    ESP.restart();
  }

  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
  esp_now_register_send_cb(onDataSent);

  int status = esp_now_add_peer(receiverAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
  if (status == 0) {
    Serial.println("Adding peer succeeded");
  } else {
    Serial.print("Adding peer failed.");
  }
}

void onDataSent(uint8_t* mac_addr, uint8_t sendStatus) {
  if (sendStatus == 0){
    Serial.println("Delivery succeeded");
  } else{
    Serial.print("Delivery failed!");
  }
}

void loop() {
  float temperature = 12.50;
  float humidity = 49.31;
  strcpy(myMessage.id, sensor_id);
  snprintf(myMessage.payload, 180, payload, temperature, humidity);

  int status = esp_now_send(receiverAddress, (uint8_t *) &myMessage,
sizeof(myMessage));
```

```
  if (status == 0) {
    Serial.println("Sending message succeeded");
  } else {
    Serial.print("Sending message failed");
  }

  delay(30000);
}
```

ESP-NOW Receiver

The receiver listens for messages via ESP-NOW and when it gets one it writes the content (the raw bytes) to the serial interface.

```
#include <ESP8266WiFi.h>
#include <espnow.h>

uint8_t fixedAddress[] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC};

void setup() {
  Serial.begin(115200);
  WiFi.disconnect();
  WiFi.mode(WIFI_STA);
  wifi_set_macaddr(STATION_IF, &fixedAddress[0]);
  if (esp_now_init() != 0) {
    delay(1000);
    ESP.restart();
  }
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(onDataRecv);
}

void onDataRecv(uint8_t *macAddress, uint8_t *incomingData, uint8_t length) {
  Serial.write(incomingData, length);
  Serial.write('\n');
}

void loop() {}
```

WiFi Transmitter

This component reads messages (bytes) from the serial port (RX/TX), converts those bytes back into a message structure and then sends a POST request to an API with the sensor id as part of the URL and the JSON payload as the request body.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

#define DEBUG_FLAG 1
#ifdef DEBUG_FLAG
#define debug(x) Serial.print(x)
#define debugln(x) Serial.println(x)
#else
#define debug(x)
#define debugln(x)
#endif

const char* WIFI_SSID = "YOUR_WIFI";
const char* WIFI_PASS = "the password of your wifi";
const char* API = "http://192.168.1.100:7000/api/sensor/";

typedef struct sensor_message {
  char id[20];
  char payload[180];
} sensor_message;
sensor_message myMessage;
uint8_t incomingData[sizeof(struct sensor_message)];
size_t incomingDataLength;

WiFiClient client;
HTTPClient http;
char url[255];

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_AP_STA);
  WiFi.setSleep(WIFI_PS_NONE);
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  while (WiFi.status() != WL_CONNECTED) { debug("."); delay(200); }
  debugln("connected to wifi");
}

void loop() {
  if (Serial.available()) {
    incomingDataLength = Serial.readBytesUntil('\n', incomingData,
sizeof(myMessage));
    if (incomingDataLength == sizeof(myMessage)) {
      handleMessage(incomingData);
    }
  }
  delay(100);
}


void handleMessage(uint8_t* data) {
  debug("New message from sensor: ");

  memcpy(&myMessage, data, sizeof(myMessage));
  debugln(myMessage.id);
  debug("Payload: ");
  debugln(myMessage.payload);

  sendToAPI(&myMessage);
}

void sendToAPI(sensor_message* message) {
  strcpy(url, API);
  strcat(url, message->id);
  http.begin(client, url);
  http.setTimeout(1000);
  http.addHeader("Content-Type", "application/json");
  int httpResponseCode = http.POST(message->payload);
  http.end();
  client.stop();
  debugln("Message sent");
}
```

The good and the bad

WiFi is more convenient, let’s get that out of the way. But it is slow, very slow, and power hungry. Terrible combination when running on batteries. As we saw in a previous article, connecting to WiFi takes seconds while ESP-NOW takes only milliseconds.

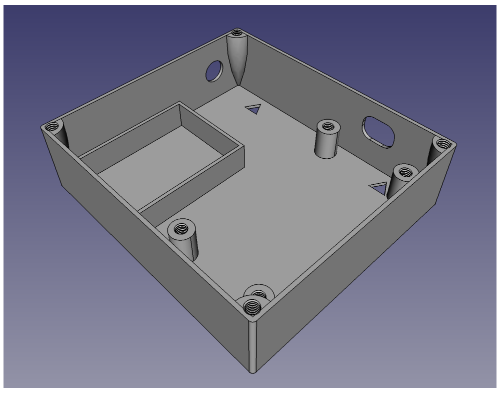
But it is possible to speed up dramatically the connection to WiFi as explained by John Mueller in his blog. Unfortunately it involves configuring your WiFi to use the same channel all the time and giving your sensors the MAC of the router they have to connect to, apart from an IP address, the IP of a DNS server and things like that. Not ideal but hey, if you can live with that it is easier than setting up a gateway!

In my case since I have multiple routers at home running them all in the same channel was not a good idea. I am also not a big fan of having IPs and MACs hard-coded in sensors if I can avoid it. With ESP-NOW I need to know the MAC of the gateway but that one won’t change and is the only hard-coded thing.

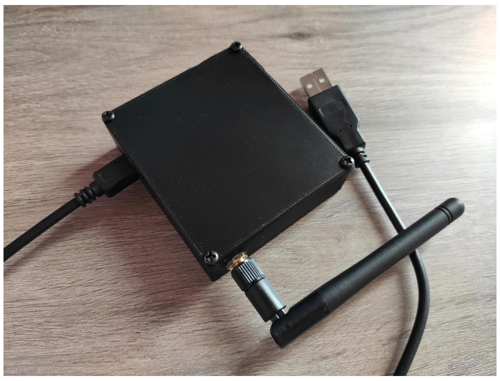
As a result my sensor went from a month on batteries to over a year and a half thanks to this. Massive difference. At least on paper, it has not been running for that long yet hehe.

Lastly for those of you reading until the end here is a few pictures of the result, turned out pretty decent if I do say so myself. Let me know what you think in the comments, or message me on Mastodon.

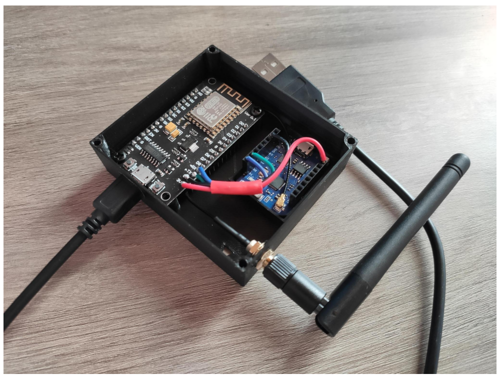
If you think my content is worth it you can Buy me a Ko-fi



Case model in FreeCAD



The final result with the antenna and USB conected



What it looks like inside