

ALERTAS AMBIENTALES: MANUAL TÉCNICO Y DE ENSAMBLAJE SENSOR DE RUIDO

¿Por qué nace el proyecto?

Este proyecto se enmarca dentro del concepto de las ciencias ciudadanas, una herramienta donde las comunidades son parte fundamental de cualquier investigación y participan activamente en el proceso de recolección y análisis de datos sobre problemáticas que afectan su calidad de vida, haciendo uso de sensores de bajo costo y accesibles.

Estos dispositivos están al servicio de comunidades para abordar ciertas preocupaciones como la contaminación por ruido y sus integrantes pueden hacer uso de estos recursos informativos, trabajarlos, compartirlos y apropiarse de ellos en la defensa de sus territorios. En este manual se describen los elementos y pasos necesarios ensamblar un sensor de ruido ambiental, que captura datos en decibelios (dB), acompañados de posición (latitud y longitud), fecha y hora, para ser almacenados dentro de una tarjeta microSD o ser enviados al servidor de mapas.

Elementos que componen el sensor de ruido

Los elementos necesarios para ensamblar el sensor de ruido serán descritos a continuación:

- Microcontrolador

Una placa ESP32 (Figura 1), la cuál es un microcontrolador de 32 bits mucho más robusto y económico en procesamiento, módulos hardware y resolución ADC al contar con un conversor de 12 bits en vez de uno de 8 bits. Se opta por esta opción debido a la cantidad de módulos que irán ensamblados, capacidad de procesamiento y precisión de lecturas frente al Arduino MEGA y Arduino UNO. Se recomienda no hacer uso de las ESP 32 blancas.

Para su configuración, es necesario instalar un firmware en concreto para poder hacer uso de la tarjeta desde el IDE de Arduino, que será explicado más adelante.

- Sensor de posición

El sensor GPS propuesto es el Neo 6M (Figura 2) que funciona por comunicación serial UART. Tiene conexión directa satelital, motivo por el cual la lectura es bastante precisa y eficiente cuando inicia el proceso de lectura.



Figura 1. [ESP32](#)



Figura 2. Sensor [GPS Neo 6M](#)

- Sensor de sonido

Un sensor KY-037 con LM393 (Figura 3), el cual captura el nivel de sonido ambiental directamente. Cabe mencionar que para este sensor es necesario realizar un proceso de calibración para que mida correctamente girando con ayuda de un destornillador, el potenciómetro que lleva incorporado.

De acuerdo con el nivel de sensibilidad de ruido que se desea capturar: estará encendida una segunda luz verde para capturar datos con alta sensibilidad, intermitente será lo óptimo y apagado será poco sensible para capturar ruido.



Figura 3. Sensor [KY-037 LM393](#)

- Reloj de tiempo real (RTC)

La captura de datos relacionados a la fecha y hora se hace a través de un módulo RTC DS1307 (figura 4) el cuál funciona por medio de la interfaz de comunicación I2C. Dicho módulo funciona perfectamente y es amigable para los usuarios; es importante que en las conexiones de I2C (SDA y SCL) se añadan resistencias de pull-up (10KOhm recomendado) para regular las diferencias de tensión.

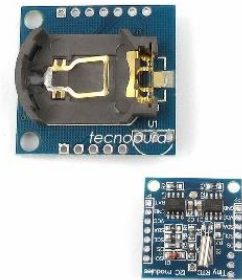


Figura 4. [Módulo RTC DS1307](#)

- Lector de tarjeta SD

Para la tarjeta SD se utiliza un módulo de lectura/escritura de tarjeta SD, el cuál maneja protocolo de comunicación SPI. Hay que tener cuidado con este módulo ya que se vende el de tarjeta SD y microSD, los de tarjeta SD tienden a fallar por no estar bien adaptados a esos tamaños, cosa que no pasa con los microSD.



Figura 5. [Módulo microSD](#)

- Condensadores

Condensador Electrolítico de 470uF x 16V, dispositivo capaz de almacenar energía en forma de campo eléctrico. Está formado por dos armaduras metálicas paralelas (generalmente de aluminio) separadas por un material dieléctrico. Usado en el dispositivo con el fin de regular picos de energía que puedan existir.¹



Figura 6. [Condensadores](#)

- Resistencias

Pequeña resistencia de película de carbón con un valor de 10K ohm y una tolerancia de $\pm 5\%$, capaz de disipar potencias de hasta (1/8w). usadas para regular paso de corriente hacia los leds guía.



Figura 7. [Resistencias](#)

¹ <https://www3.gobiernodecanarias.org/medusa/ecoblog/jgutcor/los-condensadores/>

- LED

Un diodo LED es un dispositivo que permite el paso de corriente en un solo sentido y que al ser polarizado emite un haz de luz.² Usado con diferentes colores de luz para conocer el estado del circuito.

Verde: Transmitiendo y almacenando los datos de manera optima.

Amarillo: No transmision de datos desde el gps a la tarjeta MicroSD

Rojo: Funcionamiento incorrecto de la tarjeta Micro SD, no esta almacenando datos.



Figura 8. [LED](#)

- Power Bank

PowerBank 2600mAh dispositivo portátil para cargar dispositivos electrónicos mientras se está en movimiento. ofrece una fuente de energía adicional para teléfonos móviles, tabletas u otros dispositivos electrónicos (figura 9)



Figura 9. [Power Bank](#)

- Conector USB (Macho)

Conector que permite vincular diferentes elementos a través del Universal Serial Bus (USB), usado en el montaje para alimentar la tarjeta ESP32 de informacion y la corriente desde la power bank (figura 10).



Figura 10. [CABLES USB](#)

- Interruptor Encendido/Apagado

Mini Switch Interruptor Balancin 15mm x 10mm x 12mm para Chasis de 2 Pines y 2 Posiciones altamente empleado en Circuitos Electricos y Electronicos, usado en el prototipo en dar paso a la energia desde la power bank hacia el circuito (figura 11).



Figura 11. [Interruptor Encendido/Apagado](#)

- Bornera de conexión con tornillo

Una clema (también conocido como bornera o regleta) es un tipo de conector eléctrico en el que un cable se aprisiona contra una pieza metálica mediante el uso de un tornillo.³ Se implementa dentro del circuito para conexión cableada (figura 14).



Figura 14. [Bornera de conexión con tornillo](#)

² <https://www.barcelonaed.com/blog/informacion-led/caracteristicas-y-ventajas-de-los-diodos-led/>

³ <https://outletelectricidad.com/es/646-regletas-de-conexion>

- Cables de conexión para protoboard

Cables Dupont para Protoboard Macho/Macho, Hembra/Hembra, Macho/Hembra son cables jumper de 15cm de largo. Estos se usan para conectarse desde cualquier header hembra o macho en cualquier placa, usados dentro de el prototipo para la conexión de los módulos entre si y la protoboard.

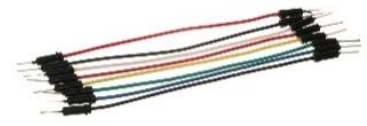


Figura 15. [Cables de conexión para protoboard](#)

- PCB

Una PCB o Placa con circuito impreso, que funciona a manera de soporte físico y de conexiones para ensamblar los componentes del dispositivo de ruido. Esta placa reemplaza la clásica protoboard con orificios que usa de pruebas y contiene las conexiones necesarias y esta contra marcada con logo de Colnodo.



Figura 16. PCB

- SD y adaptador

Un adaptador para tarjeta micro SD en el cual se almacenaran los archivos que se capturen con el sensor de ruido, esta tarjeta microSD tiene una capacidad de 8 gb.



Figura 17. [Adaptador Tarjeta Micro SD](#)

- Cargador y cable de conexión

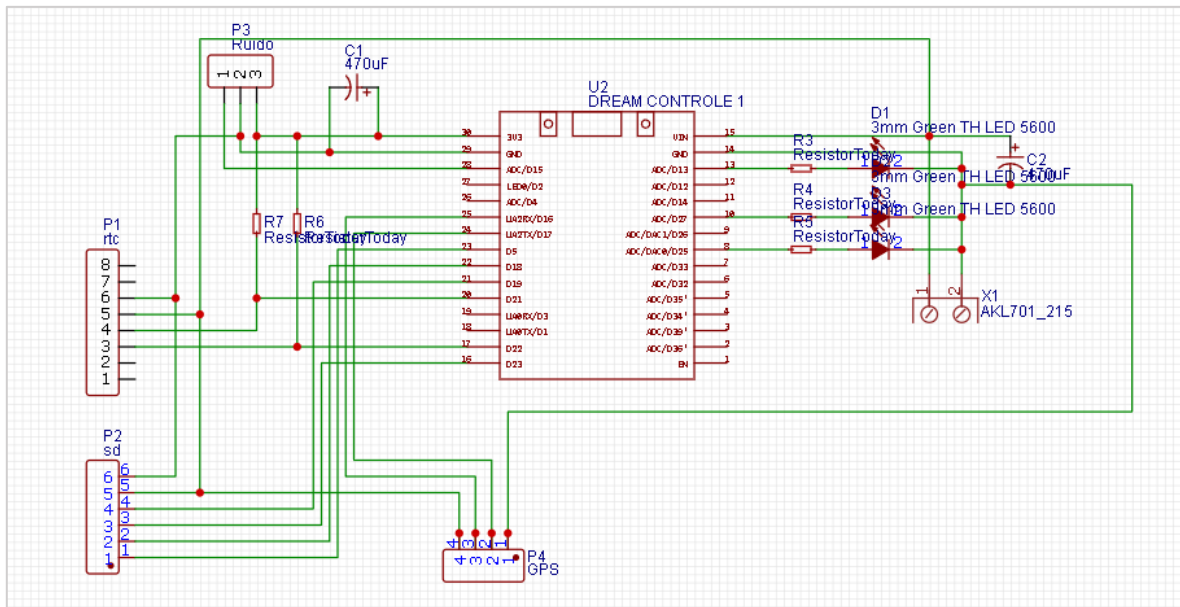
El cargador funciona como un dispositivo que permite entregar corriente continua de bajo voltaje a otros dispositivos como es el caso de la Power Bank y recargarla; para transferencia de código también. La entrada de este es tipo 2.



Figura 18. Cargador tipo 2

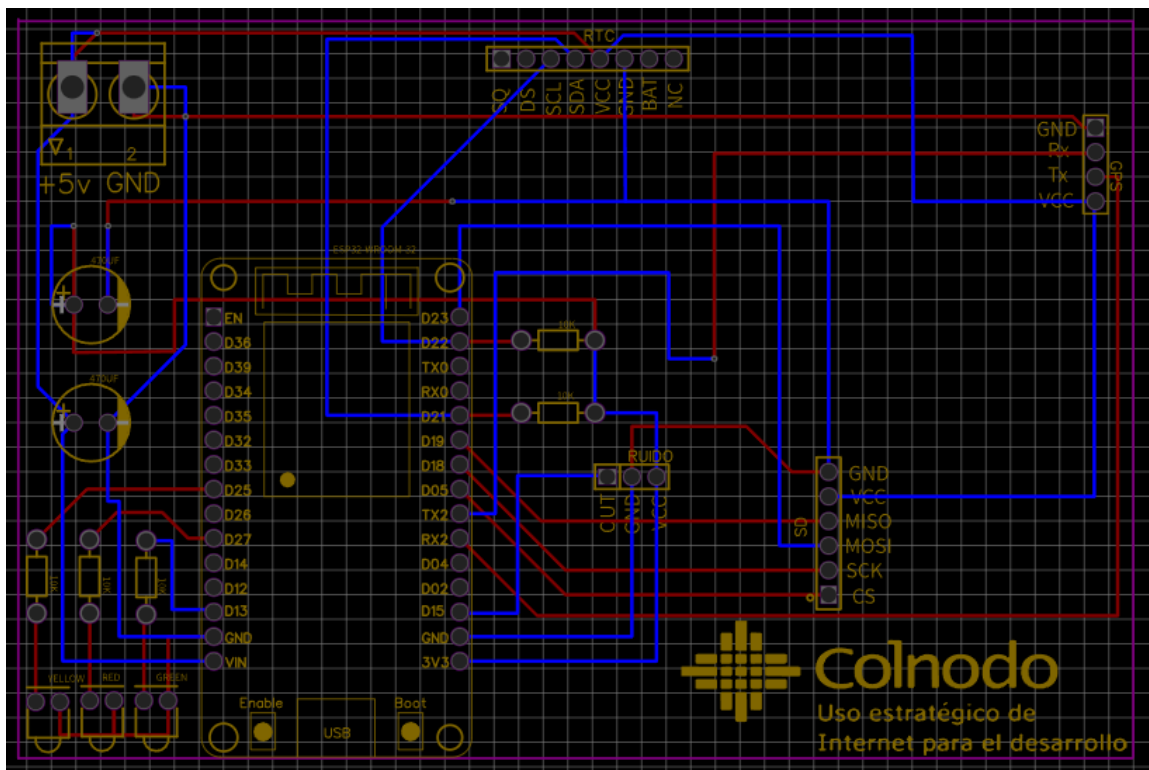
Esquemático de conexiones

A partir de pruebas realizadas en el circuito de pruebas, se propone el siguiente esquemático de conexiones para alimentar cuatro módulos respectivos y componentes pasivos como los diodos LED, condensadores, bornera y resistencias.



Diseño PCB (Placa con circuito impreso)

El diseño ha sido realizado en el software libre **EasyEDA**⁴, en cual se pueden crear y simular circuitos para posteriormente diseñar la placa de circuito impreso y generar el archivo para su impresión.

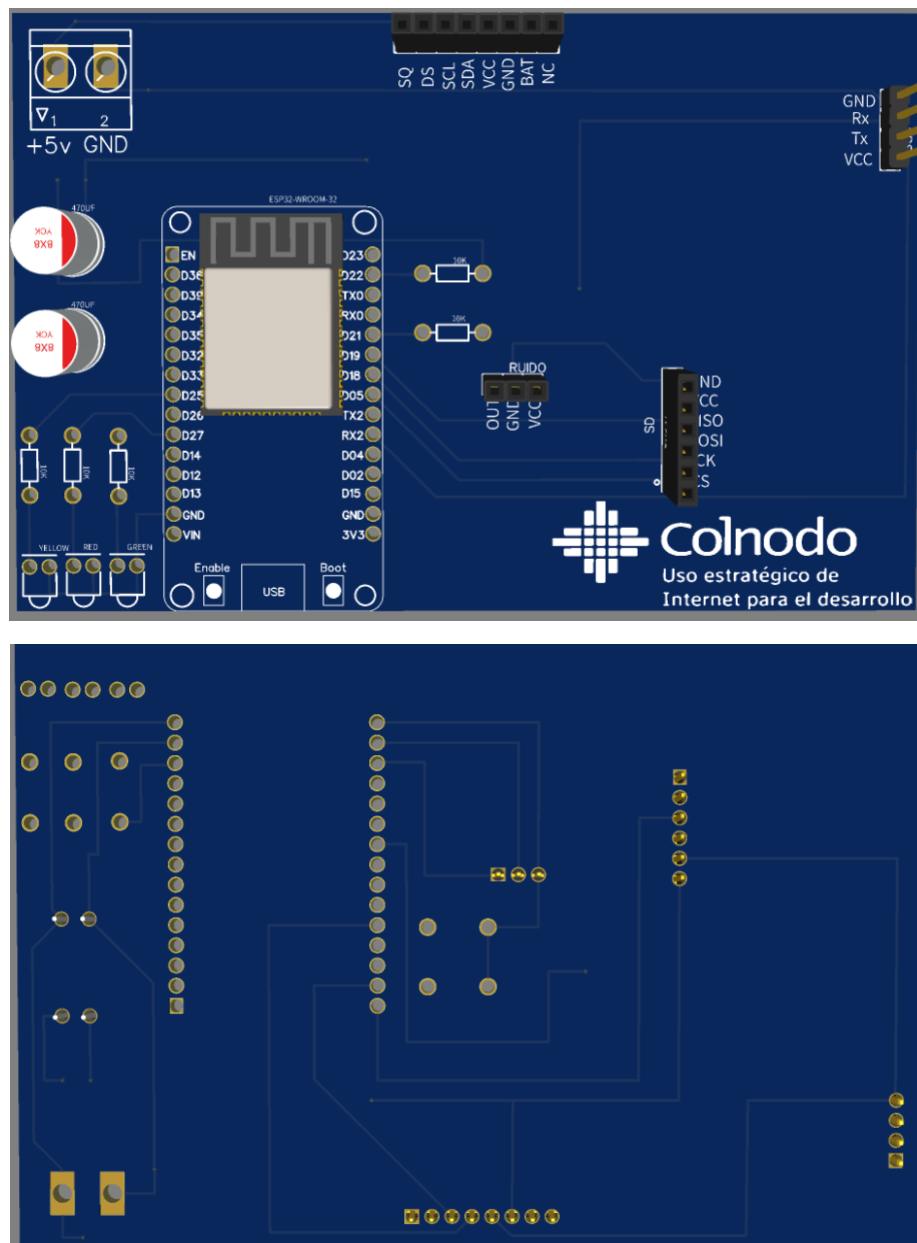


⁴ Documentación para consultar: <https://docs.easyeda.com/en/FAQ/Editor/index.html#Video-Tutorials>

Enlace de descarga: <https://easyeda.com/page/download>

Esquemático en 3D

El diseño de esta PCB es de capa doble con conexiones y puntos de contacto en ambas caras, con la finalidad de que cada uno de los módulos encaje correctamente y contenga las conexiones correctas entre estos. A continuación, la vista por ambas caras:



La fabricación de la PCB a través de un proveedor local, tiene un costo aproximado de \$40.000 COP por unidad, con unas dimensiones de 11,6 cm x 8 cm, lo cual la hace de bajo costo y practica para desplazamientos.

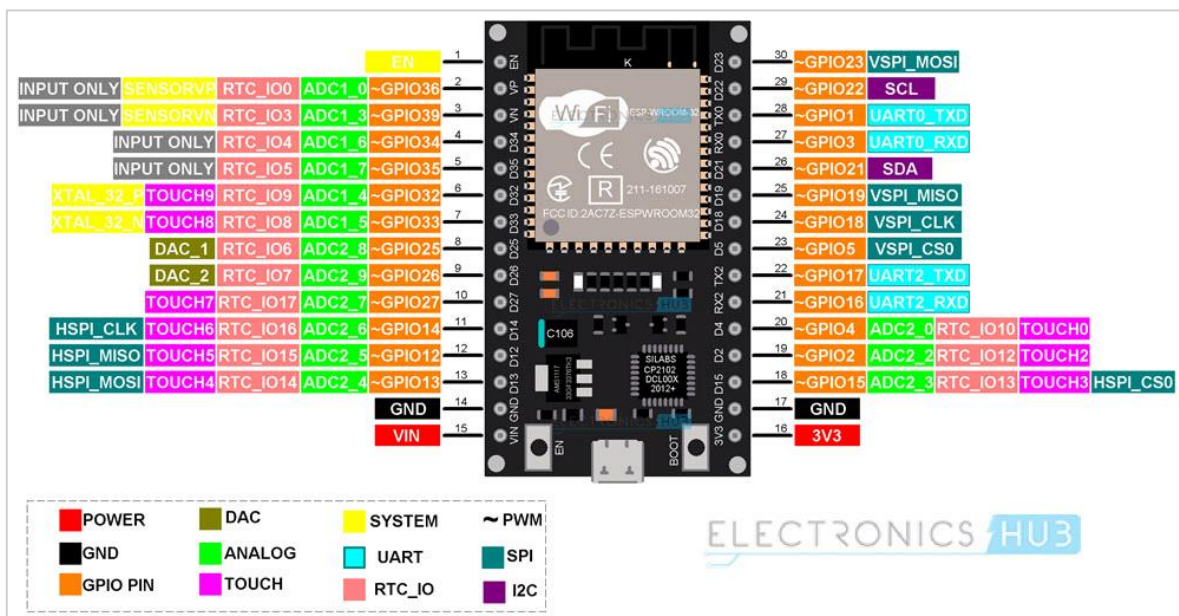
Archivo de la PCB: <https://nube.colnodo.apc.org/index.php/s/bz4iIj8XjZr58bD>

Ensamblaje y conexiones

En ensamblaje del sensor contempla los elementos descritos anteriormente y algunas herramientas adicionales como un kit de soldadura básico, destornilladores y pinzas de corte.

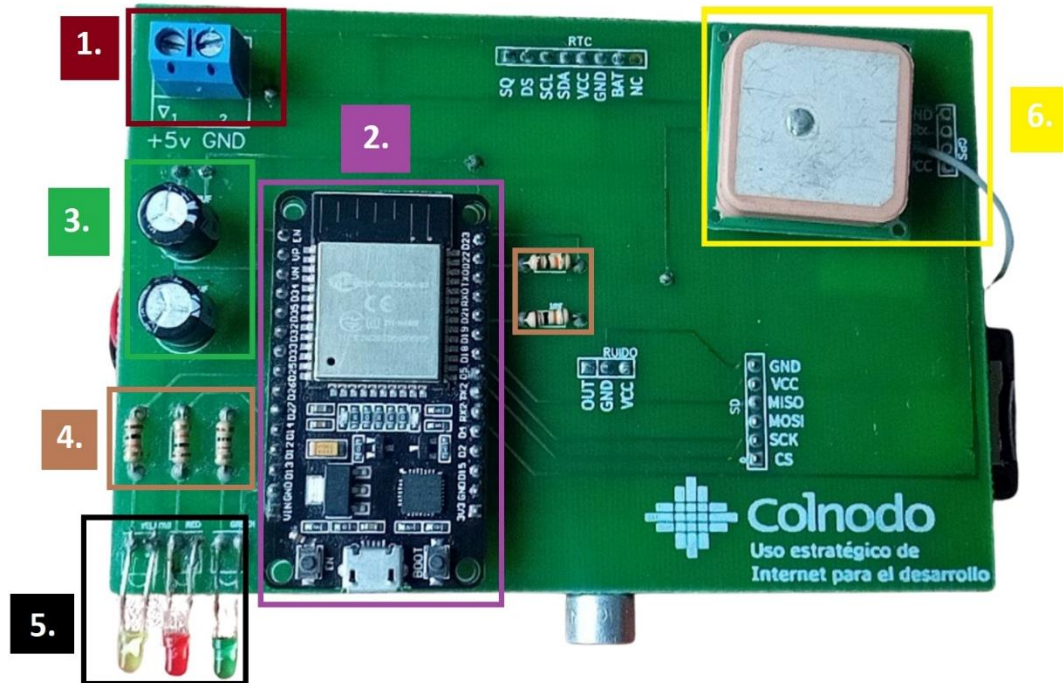
Tener en cuenta que la Bornera necesita soldarse, los módulos y otros componentes del sensor se realizan a través de conexiones entre los pines haciendo uso de cables de conexión, regletas de pines y soldadura de doble capa en la PCB.

Imagen de referencia ESP 32 con sus respectivos pines y funciones asociadas:

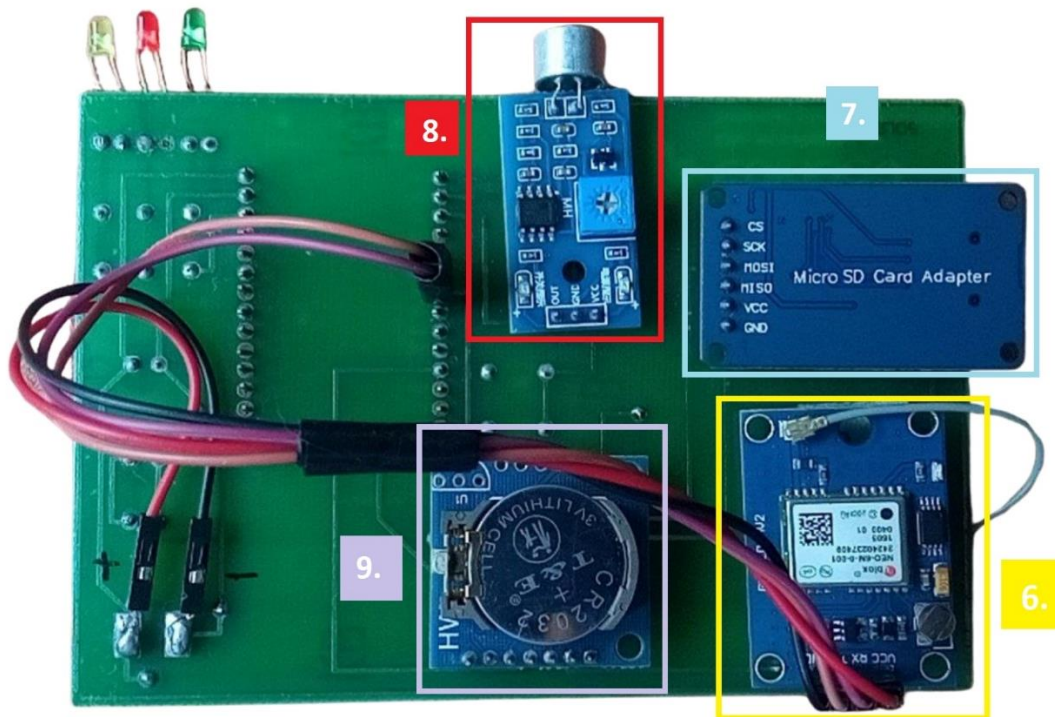


Vista frontal ensamblada:

1. Bornera: ensamblada aplicando soldadura en ambas caras de la PCB.
2. Microcontrolador ESP 32: regleta y soldadura doble capa (pines arriba y abajo)
3. Condensadores soldadura en la cara de inferior.
4. Resistencias 10K: soldadura de doble capa
5. Diodos LED: soldadura de doble capa
6. Módulo GPS: Regleta y soldadura de doble capa



Vista posterior ensamblada:



6. Módulo GPS: Regleta y soldadura de doble capa y cables de conexión.

7. Lector de tarjeta SD: regleta y soldadura de doble capa

8. Sensor de ruido Regleta y soldadura de doble capa

9. Módulo RTC (reloj en tiempo real): Regleta y soldadura de doble capa

Arduino IDE y Código de programación

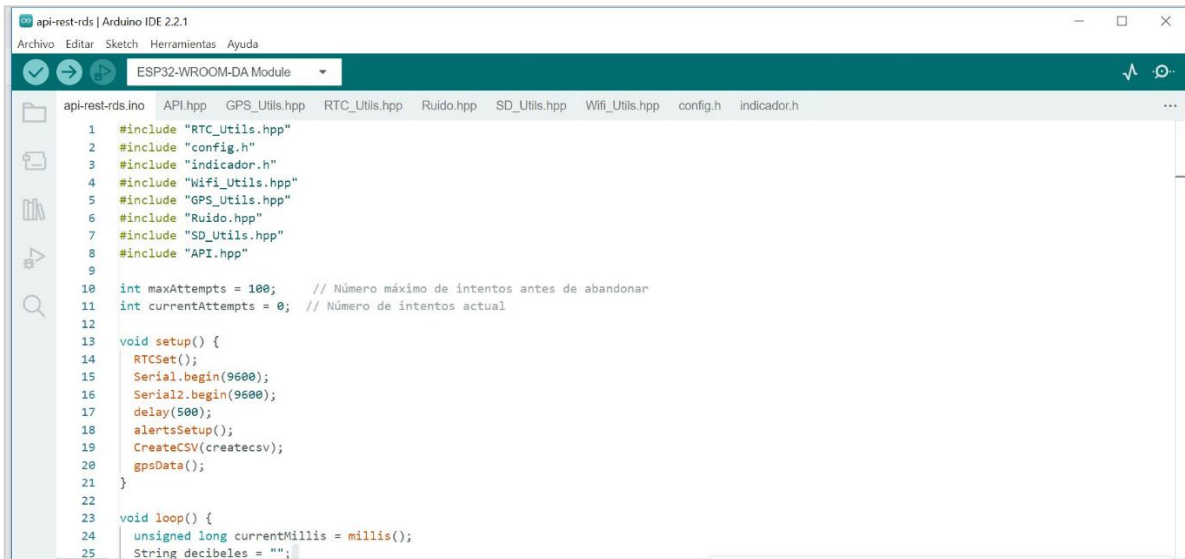
El código de programación del dispositivo para capturar los niveles de ruido ha sido construido y trabajado en software de **Arduino IDE**⁵, un software libre que funciona como un entorno de programación en lenguaje C#. Enlace de descarga: <https://www.arduino.cc/en/software>

Una vez que el Arduino IDE esté instalado, es necesario continuar la instalación de las **librerías**⁶ y controladores USB esenciales para que el computador pueda comunicarse correctamente con la placa ESP 32 a través del puerto USB, como indica la documentación recomendada en el pie de página. El código de programación se encuentra disponible para ser descargado en el siguiente enlace: <https://github.com/SoporteColnodo/Sensor-de-ruido-api>

Al momento de ser ejecutado, encontrará que consta de nueve pestañas dispuestas de la siguiente manera y manejado en una estructura de C# de void setup y void loop:

- **api-rest-rds.ino:**

Pestaña principal donde se inicia la ejecución del sensor y los módulos (objetos y variables globales para el sistema), desde acá se llaman librerías y funciones que están en otras pestañas para que se ejecuten, también la impresión de mensajes de estado en monitor serial



```
1 #include "RTC_Utils.hpp"
2 #include "config.h"
3 #include "indicador.h"
4 #include "Wifi_Utils.hpp"
5 #include "GPS_Utils.hpp"
6 #include "Ruido.hpp"
7 #include "SD_Utils.hpp"
8 #include "API.hpp"
9
10 int maxAttempts = 100; // Número máximo de intentos antes de abandonar
11 int currentAttempts = 0; // Número de intentos actual
12
13 void setup() {
14     RTCSet();
15     Serial.begin(9600);
16     Serial2.begin(9600);
17     delay(500);
18     alertsSetup();
19     CreateCSV(createcsv);
20     gpsData();
21 }
22
23 void loop() {
24     unsigned long currentMillis = millis();
25     String decibels = "";
```

⁵ Documentación adicional y de consulta:

Programar ESP32 con Arduino IDE: <https://programarfacil.com/esp8266/programar-esp32-ide-arduino/>

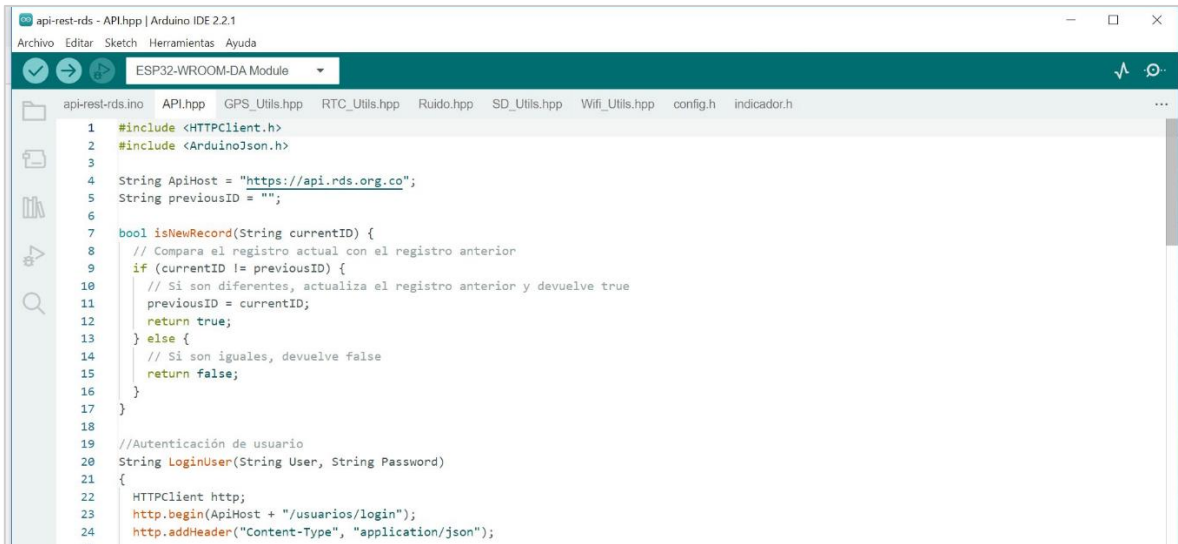
Actualizar drivers, conexiones USB para que reconozca la ESP32: <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>

ESP 32 Arduino: <https://estudiomiranda.github.io/ESP32/guia/#configuracion>

⁶ Enlace de descarga a las librerías: <https://nube.colnodo.apc.org/index.php/s/HjqYNo4RBpBLTRK>

- **API.hpp:**

En esta pestaña se establece la estructura de conexión a la API, parámetros de conexión, así como mensajes de estado.



```
api-rest-rds - API.hpp | Arduino IDE 2.2.1
Archivo  Editar  Sketch  Herramientas  Ayuda
ESP32-WROOM-DA Module

api-rest-rds.ino  API.hpp  GPS_Utils.hpp  RTC_Utils.hpp  Ruido.hpp  SD_Utils.hpp  Wifi_Utils.hpp  config.h  indicador.h

1  #include <HTTPClient.h>
2  #include <ArduinoJson.h>
3
4  String ApiHost = "https://api.rds.org.co";
5  String previousID = "";
6
7  bool isNewRecord(String currentID) {
8      // Compara el registro actual con el registro anterior
9      if (currentID != previousID) {
10         // Si son diferentes, actualiza el registro anterior y devuelve true
11         previousID = currentID;
12         return true;
13     } else {
14         // Si son iguales, devuelve false
15         return false;
16     }
17 }
18
19 //Autenticación de usuario
20 String LoginUser(String User, String Password)
21 {
22     HTTPClient http;
23     http.begin(ApiHost + "/usuarios/login");
24     http.addHeader("Content-Type", "application/json");
```

- **GPS_Utils.hpp:**

Desde aquí se establecen los parámetros del módulo GPS, la conexión, librerías necesarias, mensajes de estado.



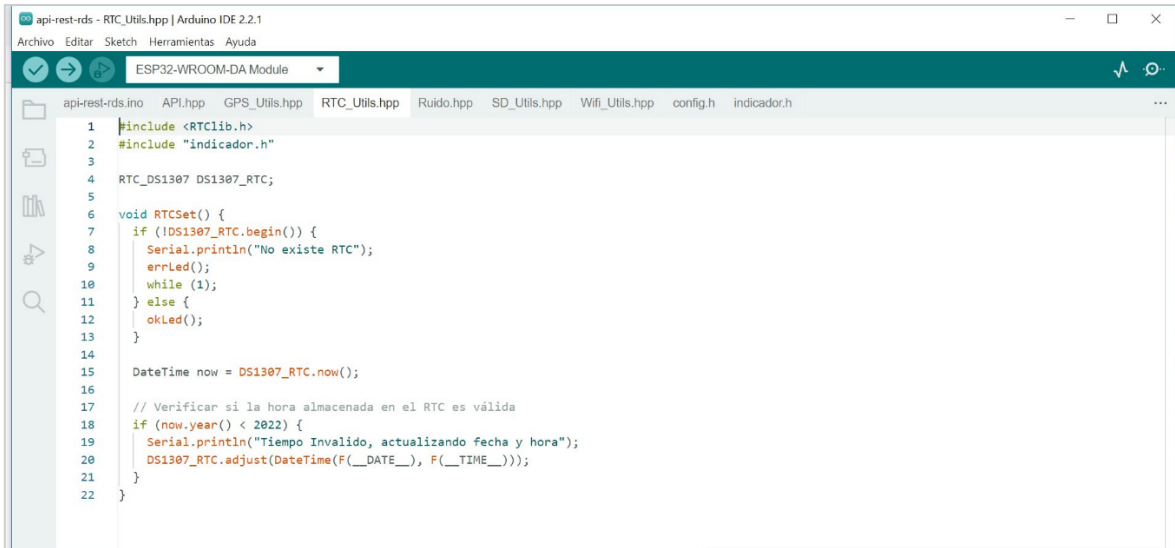
```
api-rest-rds - GPS_Utils.hpp | Arduino IDE 2.2.1
Archivo  Editar  Sketch  Herramientas  Ayuda
ESP32-WROOM-DA Module

api-rest-rds.ino  API.hpp  GPS_Utils.hpp  RTC_Utils.hpp  Ruido.hpp  SD_Utils.hpp  Wifi_Utils.hpp  config.h  indicador.h

1  #include <TinyGPSPlus.h>
2  #include "indicador.h"
3
4  TinyGPSPlus gps;
5
6  bool gpsData() {
7      unsigned long startTime = millis();
8
9      while (millis() - startTime < 10000) { // Esperar hasta 10 segundos para obtener una fijación válida
10         while (Serial2.available() > 0) {
11             if (gps.encode(Serial2.read())) {
12                 if (gps.location.isValid()) {
13                     Serial.println("Conexión a GPS correcta");
14                     //okLed();
15                     return true;
16                 }
17             }
18         }
19         delay(1000); // Pequeño retardo para evitar lecturas excesivas
20     }
21
22     Serial.println("No se pudo obtener una fijación GPS válida");
23     //enrGPS();
24     return false;
25 }
```

- **RTC_Utils.hpp:**

Se establecen los parámetros del módulo RTC, la conexión, librerías necesarias, mensajes de estado y configuración de formato para la hora y fecha del módulo.



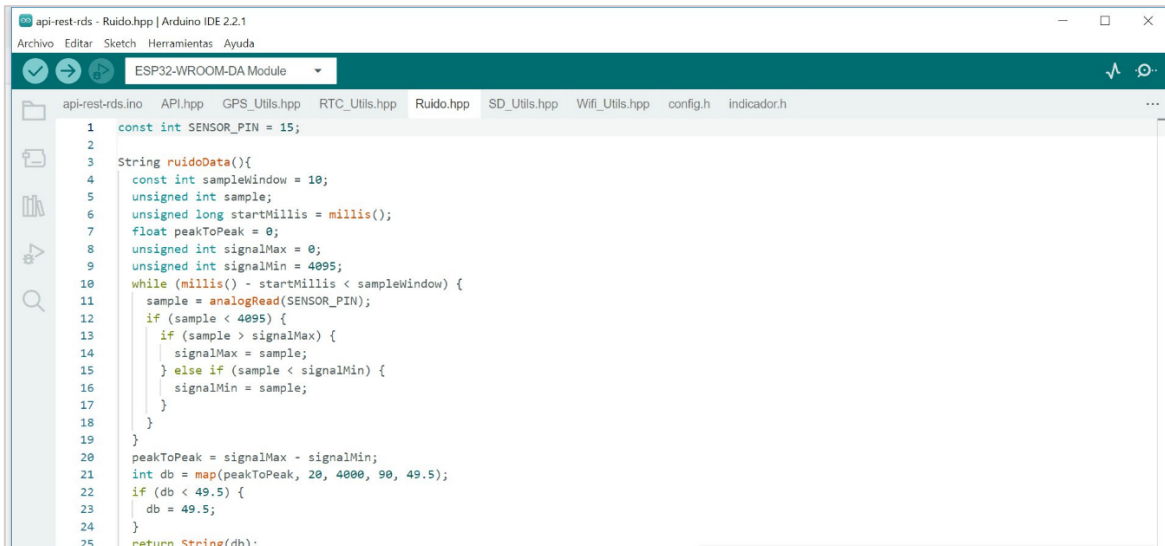
```

1 #include <RTClib.h>
2 #include "indicador.h"
3
4 RTC_DS1307 DS1307_RTC;
5
6 void RTCSet() {
7   if (!DS1307_RTC.begin()) {
8     Serial.println("No existe RTC");
9     errLed();
10    while (1);
11  } else {
12    okLed();
13  }
14
15  DateTime now = DS1307_RTC.now();
16
17  // Verificar si la hora almacenada en el RTC es válida
18  if (now.year() < 2022) {
19    Serial.println("Tiempo Invalido, actualizando fecha y hora");
20    DS1307_RTC.adjust(DateTime(F(__DATE__), F(__TIME__)));
21  }
22 }

```

- **Ruido.hpp:**

Se establecen funciones, ecuaciones y cálculos para capturas datos del nivel de ruido desde el sensor y convertirlo a decibelios (dB), apoyado de librerías y mensajes de estado.



```

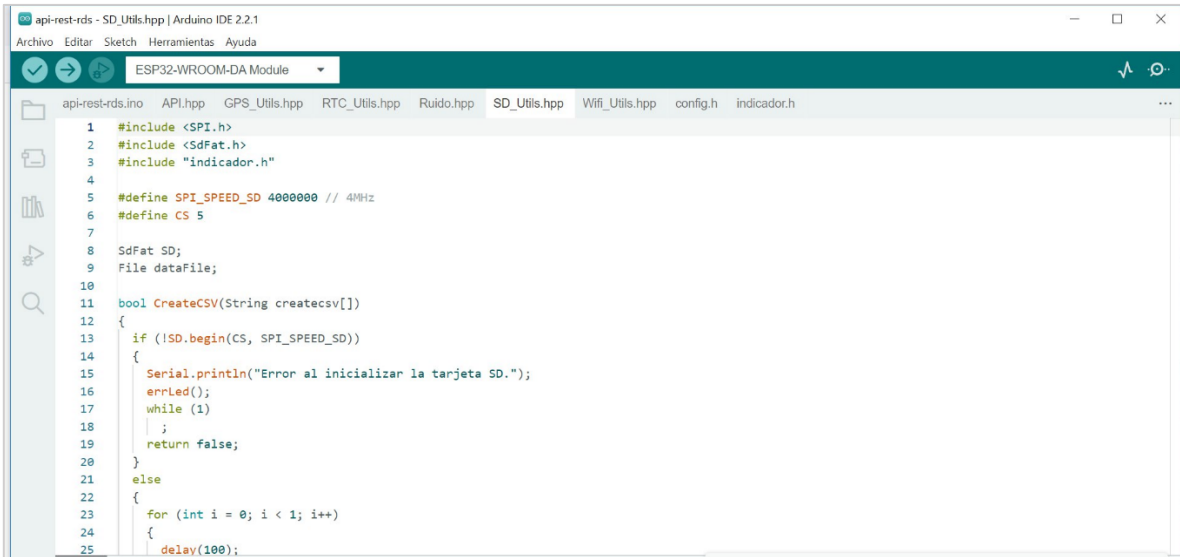
1 const int SENSOR_PIN = 15;
2
3 String ruidoData(){
4   const int sampleWindow = 10;
5   unsigned int sample;
6   unsigned long startMillis = millis();
7   float peakToPeak = 0;
8   unsigned int signalMax = 0;
9   unsigned int signalMin = 4095;
10  while (millis() - startMillis < sampleWindow) {
11    sample = analogRead(SENSOR_PIN);
12    if (sample < 4095) {
13      if (sample > signalMax) {
14        signalMax = sample;
15      } else if (sample < signalMin) {
16        signalMin = sample;
17      }
18    }
19  }
20  peakToPeak = signalMax - signalMin;
21  int db = map(peakToPeak, 20, 4000, 90, 49.5);
22  if (db < 49.5) {
23    db = 49.5;
24  }
25  return String(db);

```

Para capturar el valor de nivel de ruido en decibelios, se hace un escalamiento con la función “map”, teniendo en cuenta niveles de referencia dados en la [página guía](#) y adaptándolo a la capacidad de la ESP32. Los niveles más bajos de ADC corresponden a menores niveles de dB y los más altos a mayores niveles; estos límites están ajustados entre 0 dB y 120 dB de acuerdo con la normativa vigente.]

- **SD_Utils.hpp:**

Se establecen parámetros para el módulo SD, desde el cual se creará el archivo .CSV y el almacenamiento de los datos provenientes de los distintos módulos, junto a mensajes de estado y los respectivos campos que deben crearse dentro del archivo.



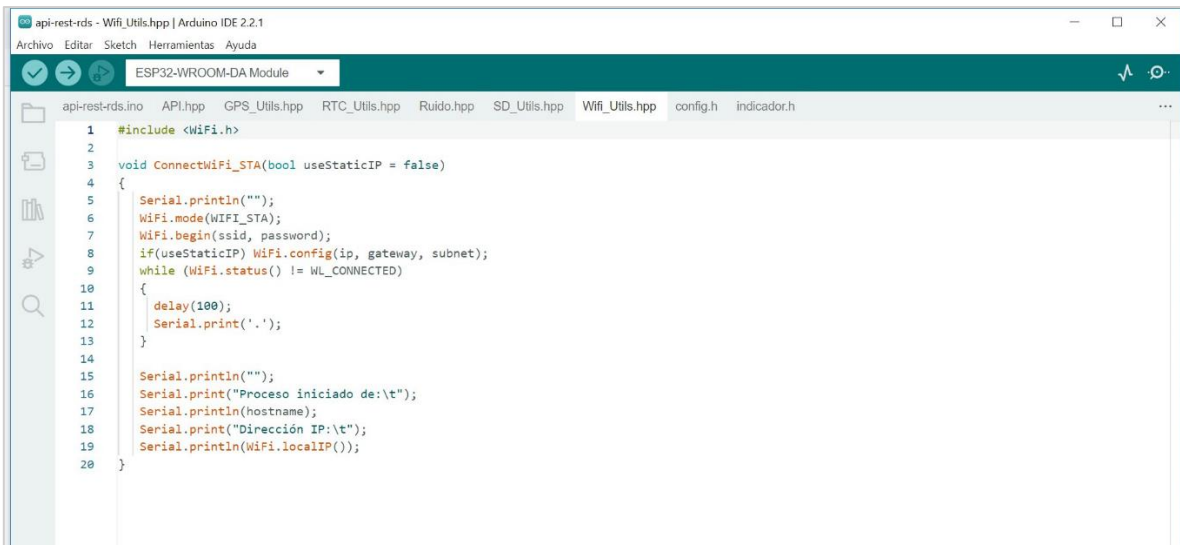
```

1 #include <SPI.h>
2 #include <SdFat.h>
3 #include "indicador.h"
4
5 #define SPI_SPEED_SD 4000000 // 4MHz
6 #define CS 5
7
8 SdFat SD;
9 File dataFile;
10
11 bool CreateCSV(String createcsv[])
12 {
13     if (!SD.begin(CS, SPI_SPEED_SD))
14     {
15         Serial.println("Error al inicializar la tarjeta SD.");
16         errLed();
17         while (1)
18         {
19             ;
20             return false;
21         }
22     }
23     else
24     {
25         for (int i = 0; i < 1; i++)
26         {
27             delay(100);
28         }
29     }
30 }

```

- **Wifi_Utils.hpp:**

Se establecen parámetros de conexión por parte de la tarjeta ESP 32 a una red wifi, con el objetivo de realizar el envío a través de la API y mensajes de estado.




```

1 #include <WiFi.h>
2
3 void ConnectWifi_STA(bool useStaticIP = false)
4 {
5     Serial.println("");
6     WiFi.mode(WIFI_STA);
7     WiFi.begin(ssid, password);
8     if (useStaticIP) WiFi.config(ip, gateway, subnet);
9     while (WiFi.status() != WL_CONNECTED)
10     {
11         delay(100);
12         Serial.print('.');
13     }
14
15     Serial.println("");
16     Serial.print("Proceso iniciado de:\t");
17     Serial.println(hostname);
18     Serial.print("Dirección IP:\t");
19     Serial.println(WiFi.localIP());
20 }

```

- **config.h:**

Se establecen parámetros para asignar una red wifi a la cual se conectará, ingresando un usuario y contraseña y un nombre del sensor, usuario y contraseña para conectar a la API. Desde esta pestaña también se establece el nombre del archivo o archivos a crear para guardar los datos, la frecuencia de tiempo para la captura, almacenamiento y envío de datos tanto a la SD como a la API.



```

1 const char* ssid = "Nombre red "; // Nombre de la red WiFi
2 const char* password = " contraseña "; // Contraseña de la red WiFi
3 const char* hostname = "ESP32_BETA"; // nombre del dispositivo
4
5 //extern String createcsv[] = {"general.csv", "newpromediooctubre.csv"}; //archivos que se crean
6 extern String createcsv[] = {"general.csv"}; //archivos que se crean
7 extern String User = " Usuario asignado en la API "; //nombre del dispositivo en AA
8 extern String Password = " contraseña "; //contraseña del dispositivo en AA
9
10 //Si los dispositivos no se sincronizan o no responden este tiempo puede cambiar es decir el delay no es constante debido
11 // a que funciona a partir de la sincronización de dispositivos externos
12 extern unsigned long previousMillisAPI = 0; //tiempo inicial de envío del API
13 extern unsigned long previousMillisSD = 0; //tiempo inicial de envío a SD
14 extern const long intervalAPI = 30000; // tiempo de ciclo envío al API en milisegundos
15 extern const long intervalSD = 1000; // tiempo de ciclo envío a SD en milisegundos
16
17 IPAddress ip(192, 168, 43, 12); //IP del dispositivo
18 IPAddress gateway(192, 168, 43, 1); //IP-gateway del dispositivo
19 IPAddress subnet(255, 255, 255, 0); //IP-subnet del dispositivo

```

Es necesario reemplazar los datos de la red Wifi (Nombre de la red y contraseña con sus respectivos caracteres) y los datos del sensor previamente registrados en la API de Colnodo.

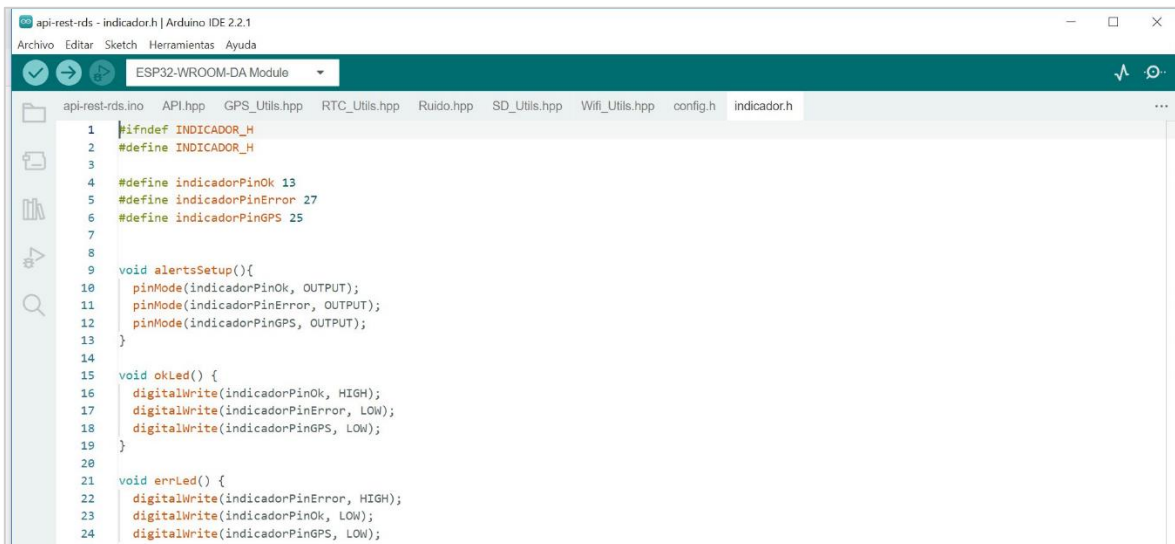
El tiempo de diferencia entre cada captura es establecido con la siguiente relación:

$$TP = TN * 248$$

Siendo TN el tiempo necesario para que mida y TP el tiempo que debo indicar en cada uno de esos retardos (delay(TP);). Por ejemplo, si se desea el registro de un dato cada 5 segundos, el tiempo a poner en esos delays es de 1240.

- **indicador.h:**

Finalmente, en esta pestaña se establecen parámetros para los LED, de manera que se activarán de acuerdo con la función asignada, logrando hacer seguimiento del estado del sensor a modo de semáforo (se encenderán respectivamente cuando uno de los módulos este fallando o capturando datos de manera correcta).



```

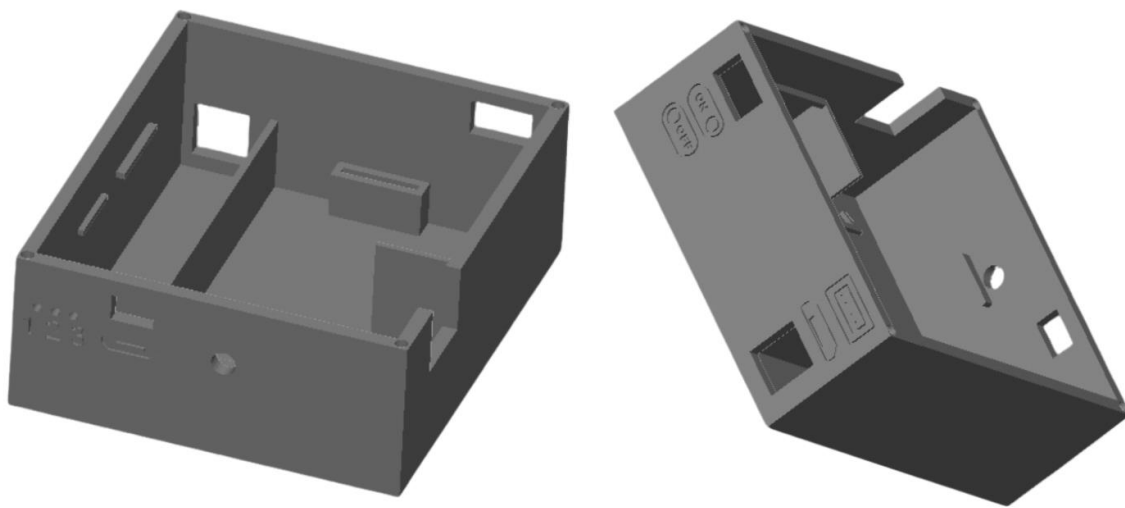
1 #ifndef INDICADOR_H
2 #define INDICADOR_H
3
4 #define indicadorPinOk 13
5 #define indicadorPinError 27
6 #define indicadorPinGPS 25
7
8
9 void alertsSetup(){
10   pinMode(indicadorPinOk, OUTPUT);
11   pinMode(indicadorPinError, OUTPUT);
12   pinMode(indicadorPinGPS, OUTPUT);
13 }
14
15 void okLed() {
16   digitalWrite(indicadorPinOk, HIGH);
17   digitalWrite(indicadorPinError, LOW);
18   digitalWrite(indicadorPinGPS, LOW);
19 }
20
21 void errLed() {
22   digitalWrite(indicadorPinError, HIGH);
23   digitalWrite(indicadorPinOk, LOW);
24   digitalWrite(indicadorPinGPS, LOW);
25 }

```


Caja contenedora

La caja contenedora actúa como un compartimiento protector para el dispositivo y los diferentes componentes que están ensamblados a las placas de circuito impreso (PCB), al tiempo de ser adecuadas para su transporte y uso en campo. La tecnología de impresión en 3D permite que, a partir de un diseño previamente realizado en el software de AutoCAD, se cree este prototipo en forma de caja cuadrada, compuesta de varios compartimientos que permiten visibilizar los LEDs, conectar a una fuente de alimentación eléctrica el dispositivo y dejar expuesto el módulo del sensor de ruido.

Vista del diseño:



Caja contenedora impresa y en uso:



La caja posee unas dimensiones de 12.9 cm x 15 cm x 5.5 cm y la tapa de 12.9 cm x 15 cm x 5.5 cm, en su exterior se encuentra el interruptor de encendido/apagado para su funcionamiento y esta identificada con logos correspondientes a la organización.



Archivo del diseño: <https://nube.colnodo.apc.org/index.php/s/bz4ij8XjZr58bD>

Captura de datos /salida de datos

La captura de datos puede ser comprobada de tres formas generales:

Micro SD:

La captura de datos se realiza mediante el módulo de Micro SD en el cual se crea un archivo en formato CSV de almacenamiento denominado **data.csv**, de tal forma que sea un archivo de bajo peso para su almacenamiento o envío. Dentro del archivo los datos se listan separados por coma (,).

 data.csv		9/02/2024 12:45 p. m.	Archivo de valores separados por comas de Microsoft Excel	13 KB
--	---	-----------------------	---	-------

- Medida ruido: Almacena la medida en decibeles, datos tomados del sensor de ruido
- Unidad de medida: dB (Decibelios), asignado desde el código.
- Lat: Datos de latitud capturados a partir del módulo GPS.
- Lng: Datos de longitud capturados a partir del módulo GPS.
- Fecha: fecha en la que el dato se almacena Año/Mes/Día, datos obtenidos del módulo RTC
- Hora: Hora establecida para ser almacenada Horas/Minutos/Segundos, datos tomados del módulo RTC.

	A	B	C	D	E
1	Medida de ruido, unidad de la medida de ruido , Lat, Lng , Fecha , Hora				
2	49,dB,4.628259,-74.069386,2024/1/7,20:49:35				
3	49,dB,4.628259,-74.069386,2024/1/7,20:49:40				
4	49,dB,4.628259,-74.069386,2024/1/7,20:49:45				

Los datos almacenados pueden ser trabajados y graficados a través de diferentes programas.

Sistema de Información geográfico de la RDS:

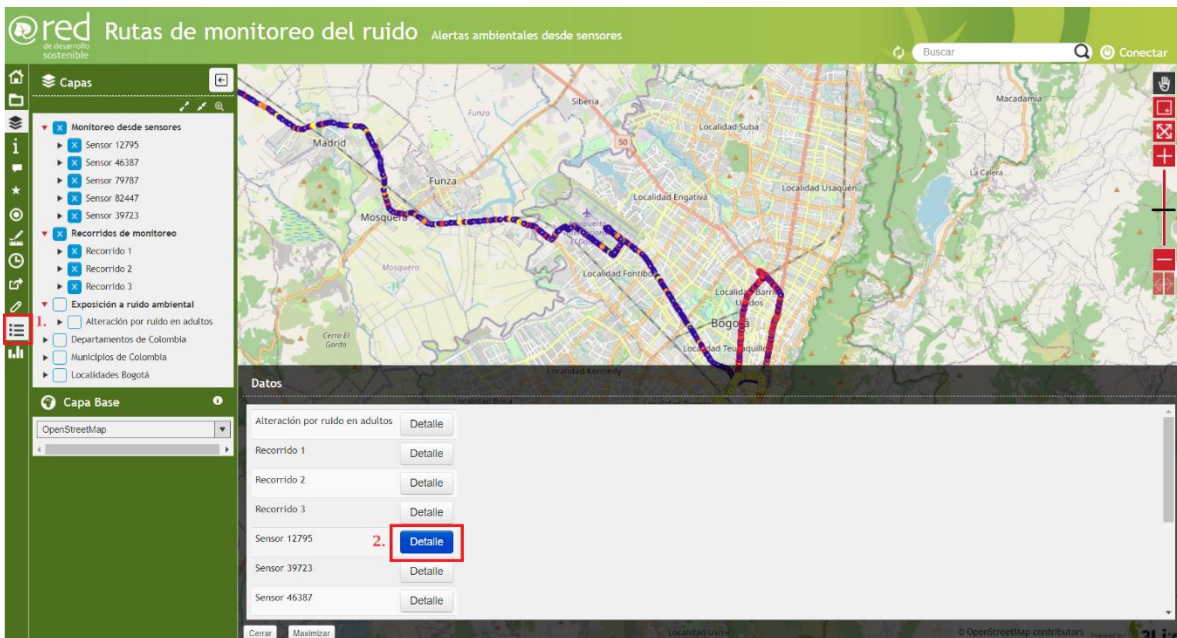
Los datos capturados a través del dispositivo son enviados a través de la API y almacenados en un archivo tipo GeoJSON. Este permite representar cada dato capturado en un elemento geográfico sobre un mapa junto con sus atributos o características como la hora, fecha, medida, etc. A partir de sus coordenadas de latitud y longitud.

El archivo ha sido configurado dentro de QGIS, para representar cada captura dentro del proyecto "Rutas de monitoreo de ruido", disponible en:

<https://rds.org.co/es/alertas-ambientales/alertas-dispositivo-ruido>

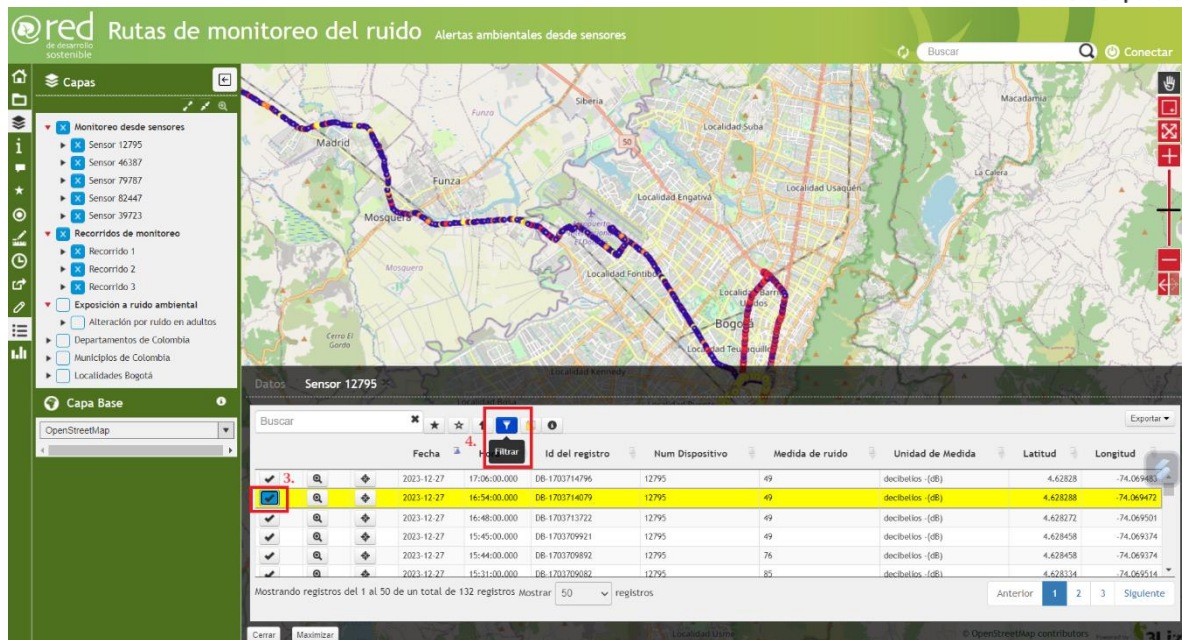


Cada captura esta asociada a un número de dispositivo asignado para su identificación y pueden consultarse o descargarse a través de diversas herramientas que ofrece la interfaz, una de ellas **Datos**, que está localizada en el panel lateral izquierdo con el siguiente icono:



Desde ahí, se identifica el dispositivo y da clic en **Detalle**.

Los datos capturados se podrán seleccionar y filtrar respectivamente para su consulta individual:



Ventana de visualización a la API:

Una ventana donde se visualizan a manera de listado cada una de las capturas realizadas desde los diferentes dispositivos de ruido en tiempo real, con sus respectivos atributos.

SENSORES DE RUIDO						
Sensor que reporta	Categoría	DECIBEL	LATITUD	LONGITUD	FECHAPUB	HORA
esp32ruidocolnodo_003@colnodo.apc.org	Ruido	49	4.628330	-74.069434	2024/02/22	16:27
esp32ruidocolnodo_003@colnodo.apc.org	Ruido	49	4.628330	-74.069434	2024/02/22	16:26
esp32ruidocolnodo_003@colnodo.apc.org	Ruido	49	4.628380	-74.069495	2024/02/22	16:21
esp32ruidocolnodo_003@colnodo.apc.org	Ruido	49	4.628380	-74.069495	2024/02/22	16:08

Enlace de visualización: <https://api.rds.org.co/reportes?categoria=sensor-ruido>

Recomendaciones

La función concreta de los tres LEDS es dar indicaciones de cómo están funcionando los módulos antes de iniciar un recorrido de medición para asegurar que esté funcionando correctamente el código, la captura de datos y envío:

- LED verde intermitente: el código está funcionando correctamente y se está enviado un dato en cada cambio de estado del LED.

- LED rojo: está fallando la comunicación con la tarjeta SD, revisar si funciona esta o que no esté ajustada la conexión.
- LED amarillo: hay comunicación con el GPS, pero no hay ningún envío o almacenamiento de datos, no ha establecido conexión con un satélite.
- LED rojo y amarillo: si están encendidos los dos indica que el módulo GPS está fallando y es necesario revisar conexiones o alimentación de energía.
- LEDS apagados: en caso de que todos estén apagados y el verde no esté intermitente, quiere decir que falló la comunicación con el RTC, es necesario revisar conexiones.

Para la conexión a una red Wifi dentro del código, es indispensable conocer el nombre y contraseña de la red a la cual se desea conectar; se recomiendan redes no superiores a 2.4G.

El dispositivo deberá mantenerse alejado de líquidos o superficies húmedas, ya que, al tener sus componentes expuestos, podría resultar en daño de algunos de los módulos o daño completo del dispositivo; no exponer a ubicaciones que causen caída o golpes.

En caso de que el dispositivo mantenga una ubicación fija, se recomienda hacer verificación periódica de funcionamiento y carga de corriente de la batería.

Mantener el dispositivo dentro de su caja.