

ALERTAS AMBIENTALES: MANUAL TÉCNICO Y DE ENSAMBLAJE SENSOR PM 10 Y PM 2.5

¿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 material particulado 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 material particulado, que captura datos de PM 10 y PM 2.5, acompañados de coordenadas (latitud y longitud), identificación del sensor, fecha y hora, para ser almacenados dentro de una tarjeta microSD o ser enviados al servidor de mapas.

Elementos que componen el sensor de material particulado

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.



Figura 1. [ESP32](#)

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

- **Sensor de material particulado PM 10 y PM 2.5**

Un sensor SDS011 Nova de PM (Figura 3), el cual captura material particulado de diámetro de 10 micrómetros y 2.5 micrómetros a partir de tecnología láser. En su interior contiene un ventilador, que de mayor tamaño al de otros sensores del mercado, lo cual implica que será mejor la calidad de los datos capturados

Es necesario adaptarle una manguera de diámetro 0.5 mm para la captura de aire.



Figura 3. Sensor [SDS011 Nova PM](#)

- **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.

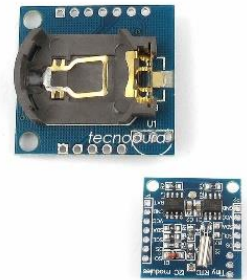


Figura 4. [Módulo RTC DS1307](#)

- **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](#)

- **Power Bank**

PowerBank 5000 mAh 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 6).



Figura 6. [Power Bank 500 mAh](#)

- **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 7).



Figura 7. [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 8).



Figura 8. [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.¹ Se implementa dentro del circuito para conexión cableada (figura 9).



Figura 9. [Bornera de conexión con tornillo](#)

- **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 10. [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 11. 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 12).



Figura 12. [Adaptador TarjetaMicro SD](#)

¹ <https://outletelectricidad.com/es/646-regletas-de-conexion>

- **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 del cargador es tipo 2.



Figura 13. [Cargador tipo 2](#)

- **Pantalla Display LCD**

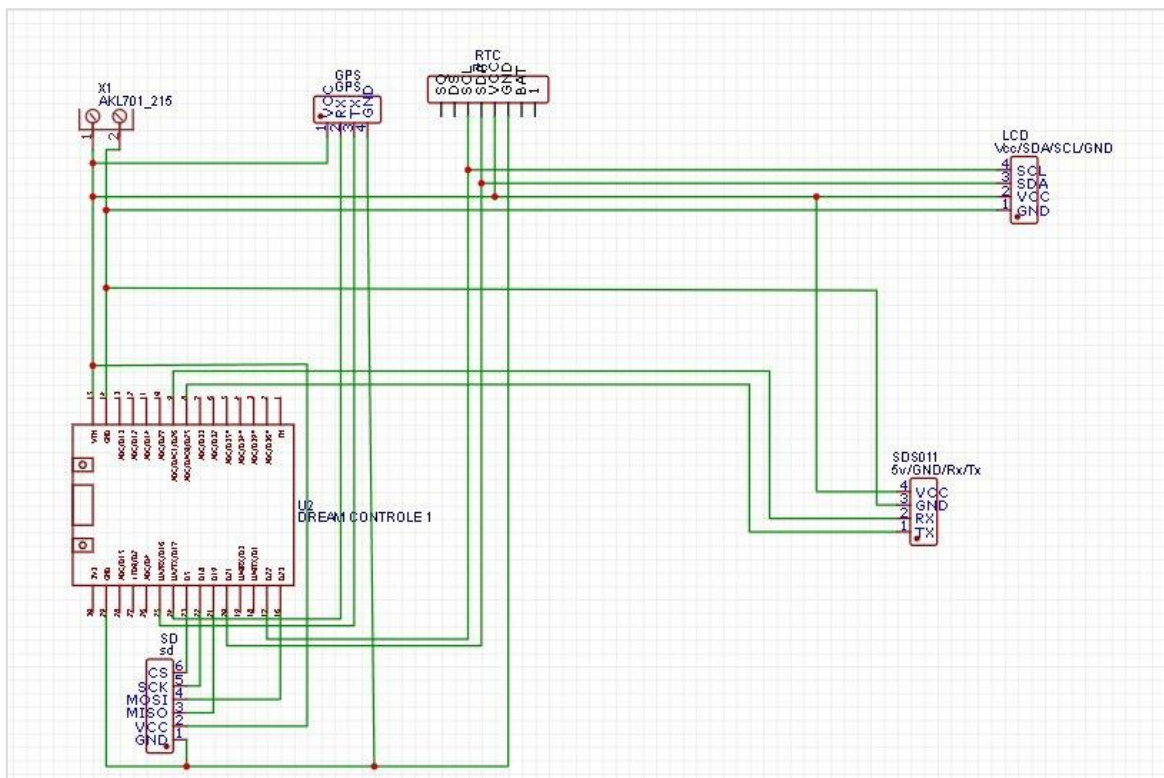
Una LCD o pantalla de cristal líquido, de referencia **4x20**, es un componente que facilita la visualización de información o datos de una forma gráfica, haciendo uso de caracteres y símbolos configurados desde el código de programación en Arduino IDE (figura 14).



Figura 13. [Pantalla LCD 4X20](#)

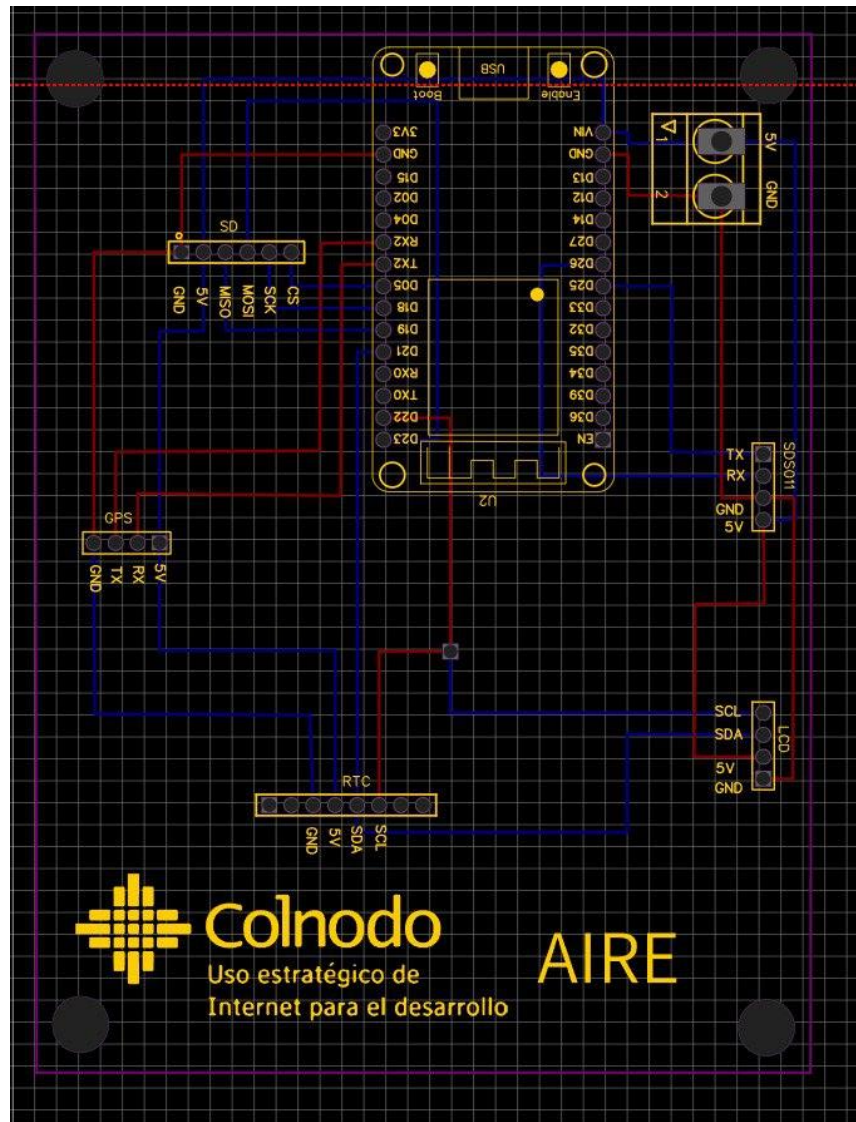
Esquemático de conexiones

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**², en cual se pueden crear y simular circuitos para posteriormente diseñar la placa de circuito impreso y generar el archivo para su impresión.

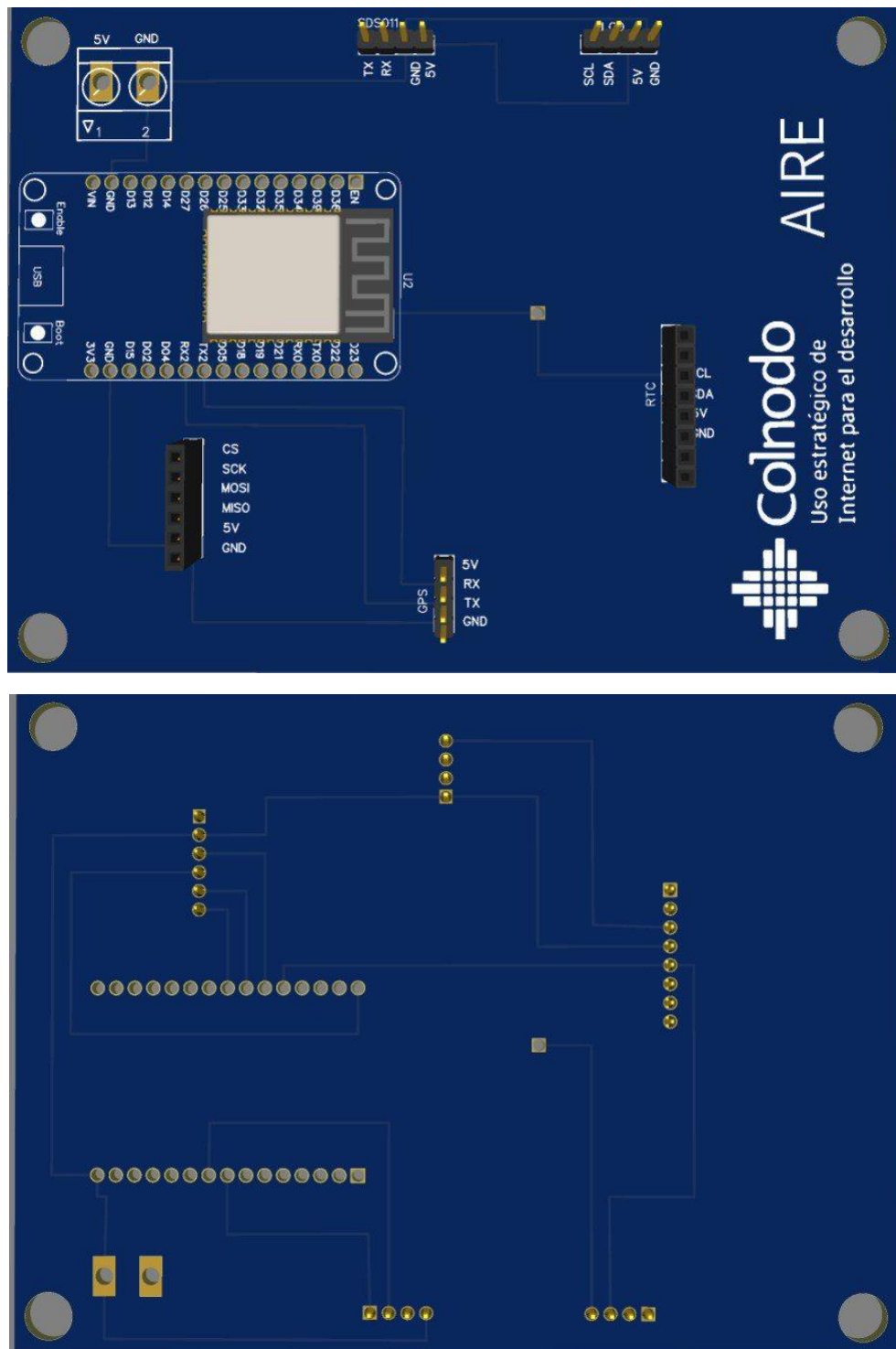


Enlace de descarga: <https://easyeda.com/page/download>

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:

² Documentación para consultar: <https://docs.easyeda.com/en/FAQ/Editor/index.html#Video-Tutorials>



La fabricación de la PCB a través de un proveedor local, tiene un costo aproximado de \$45.000 COP por unidad, con unas dimensiones de 11,8 cm x 8,7 cm, lo cual la hace de bajo costo y practica para desplazamientos.

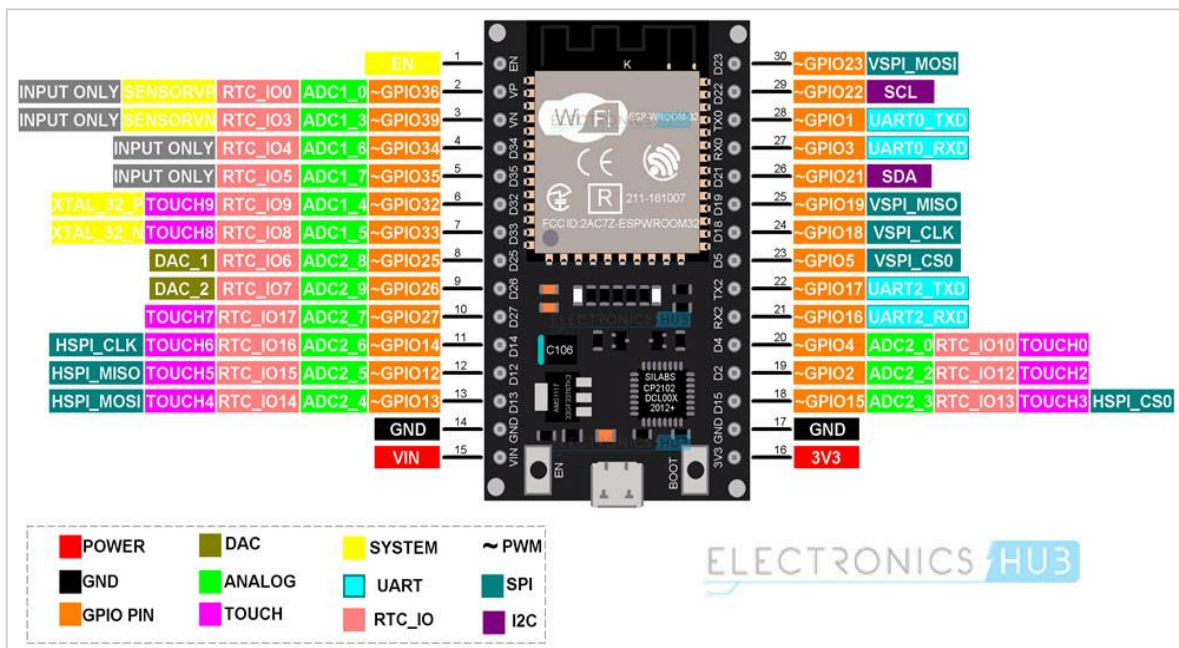
Archivo de la PCB: <https://nube.colnodo.apc.org/index.php/s/MrLtsyfC2da3JpN>

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.

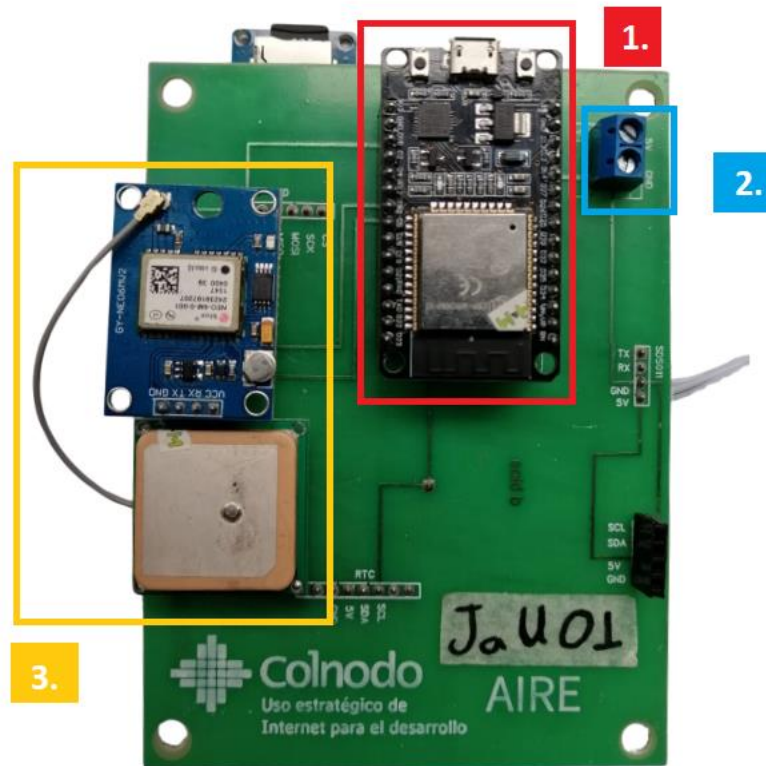
Tener en cuenta que la Bornera necesita soldarse, los módulos y otros componentes del sensor se realizan a través de conexiones entre los pines haciendo uso de cables de conexión, regletas de pines y soldadura de doble capa en la PCB.

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

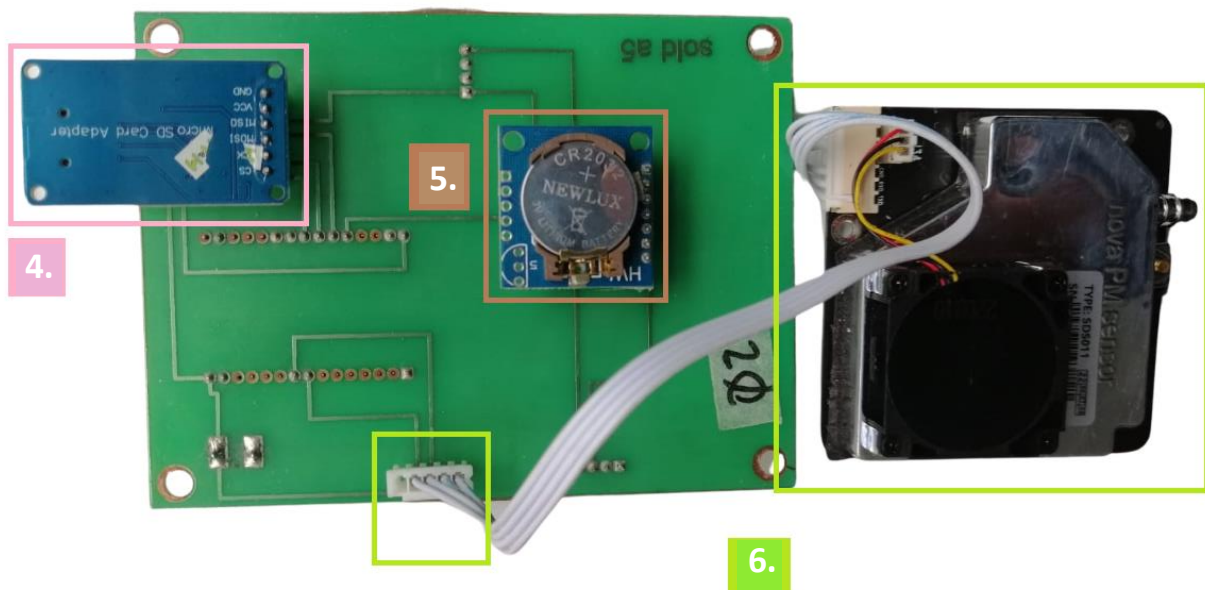


Vista frontal ensamblada:

1. Microcontrolador ESP 32: regleta y soldadura doble capa (pines arriba y abajo)
2. Bornera: ensamblada aplicando soldadura en ambas caras de la PCB.
3. Módulo GPS: Regleta y soldadura de doble capa



Vista posterior ensamblada:



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

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

6. Sensor de material particulado: Regleta y soldadura de doble capa.

Arduino IDE y Código de programación

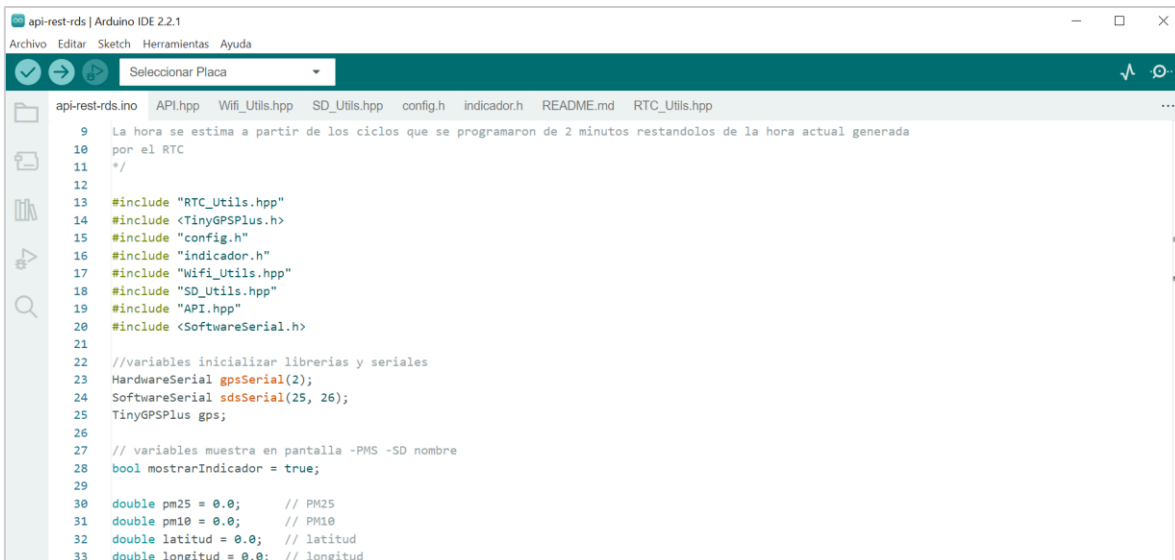
El código de programación del dispositivo para capturar los niveles de ruido ha sido construido y trabajado en software de **Arduino IDE**³, un software libre que funciona como un entorno de programación en lenguaje C#. Enlace de descarga: <https://www.arduino.cc/en/software>

Una vez que el Arduino IDE esté instalado, es necesario continuar la instalación de las **librerías**⁴ 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-PM-SDS011>

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, los parámetros del módulo GPS y también la impresión de mensajes de estado en monitor serial y en la pantalla LCD.



```
api-rest-rds.ino | API.hpp | Wifi_Utils.hpp | SD_Utils.hpp | config.h | indicador.h | README.md | RTC_Utils.hpp
9   La hora se estima a partir de los ciclos que se programaron de 2 minutos restandolos de la hora actual generada
10  por el RTC
11  */
12
13  #include "RTC_Utils.hpp"
14  #include <TinyGPSPlus.h>
15  #include "config.h"
16  #include "indicador.h"
17  #include "Wifi_Utils.hpp"
18  #include "SD_Utils.hpp"
19  #include "API.hpp"
20  #include <SoftwareSerial.h>
21
22  //variables inicializar librerias y seriales
23  HardwareSerial gpsSerial(2);
24  SoftwareSerial sdsSerial(25, 26);
25  TinyGPSPlus gps;
26
27  // variables muestra en pantalla -PMS -SD nombre
28  bool mostrarIndicador = true;
29
30  double pm25 = 0.0;    // PM25
31  double pm10 = 0.0;    // PM10
32  double latitud = 0.0; // latitud
33  double longitud = 0.0; // longitud
```

³ Documentación adicional y de consulta:

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

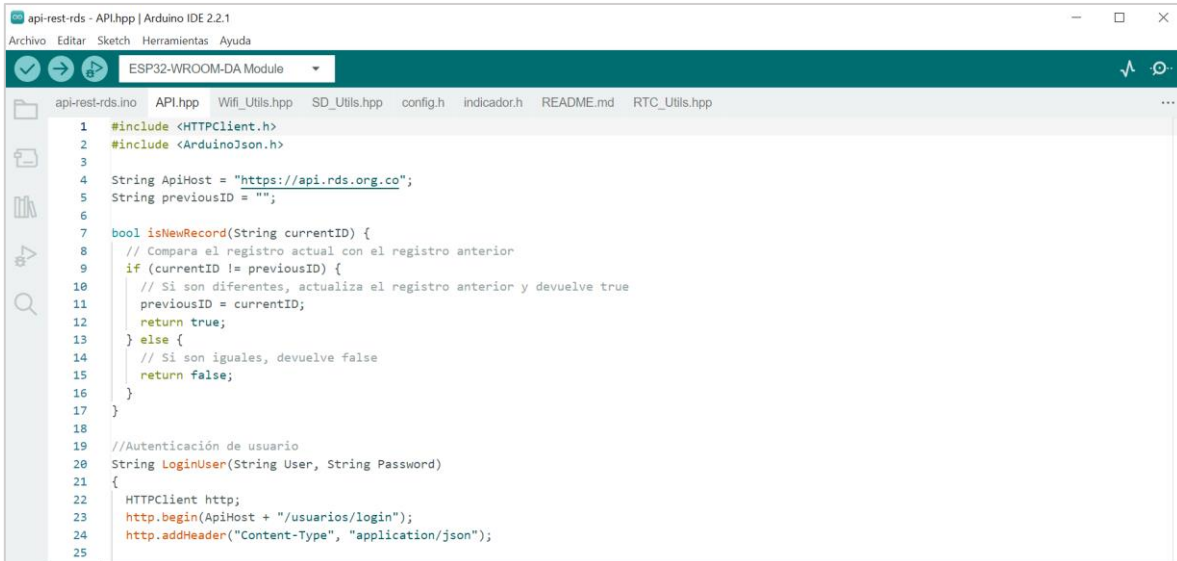
Actualizar drivers, conexiones USB para que reconozca la ESP32: <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>

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

⁴ 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.



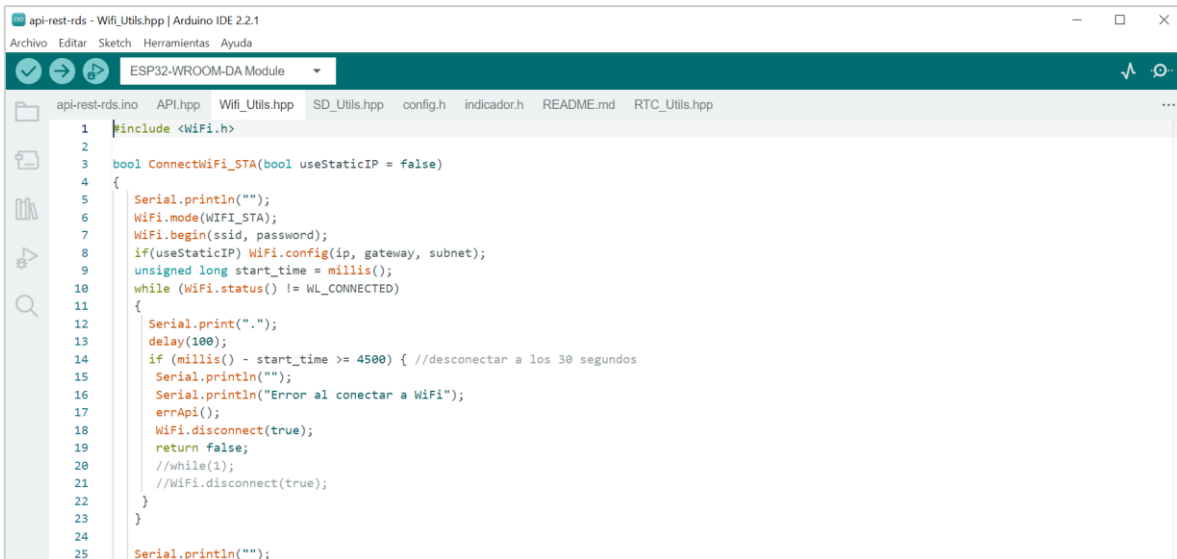
```
api-rest-rds - API.hpp | Arduino IDE 2.2.1
Archivo  Editar  Sketch  Herramientas  Ayuda
ESP32-WROOM-DA Module

api-rest-rds.ino  API.hpp  Wifi_Utils.hpp  SD_Utils.hpp  config.h  indicador.h  README.md  RTC_Utils.hpp

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");
25 }
```

- **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.



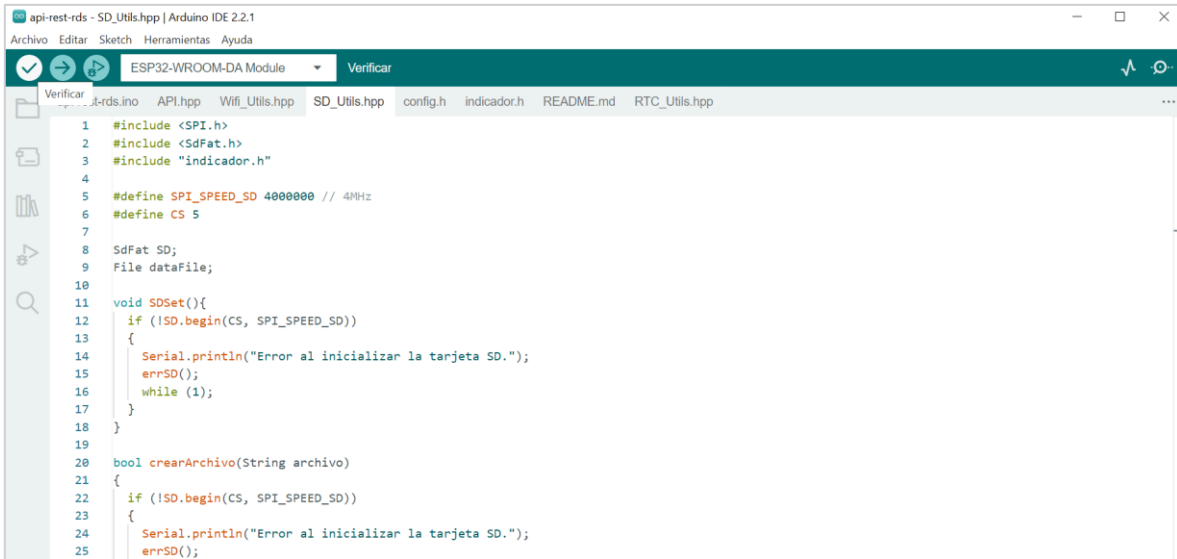
```
api-rest-rds - Wifi_Utils.hpp | Arduino IDE 2.2.1
Archivo  Editar  Sketch  Herramientas  Ayuda
ESP32-WROOM-DA Module

api-rest-rds.ino  API.hpp  Wifi_Utils.hpp  SD_Utils.hpp  config.h  indicador.h  README.md  RTC_Utils.hpp

1  #include <WiFi.h>
2
3  bool 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      unsigned long start_time = millis();
10     while (WiFi.status() != WL_CONNECTED)
11     {
12         Serial.print(".");
13         delay(100);
14         if (millis() - start_time >= 4500) { //desconectar a los 30 segundos
15             Serial.println("");
16             Serial.println("Error al conectar a WiFi");
17             errApi();
18             WiFi.disconnect(true);
19             return false;
20             //while(1);
21             //WiFi.disconnect(true);
22         }
23     }
24
25     Serial.println("");
26 }
```

- **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.



```

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

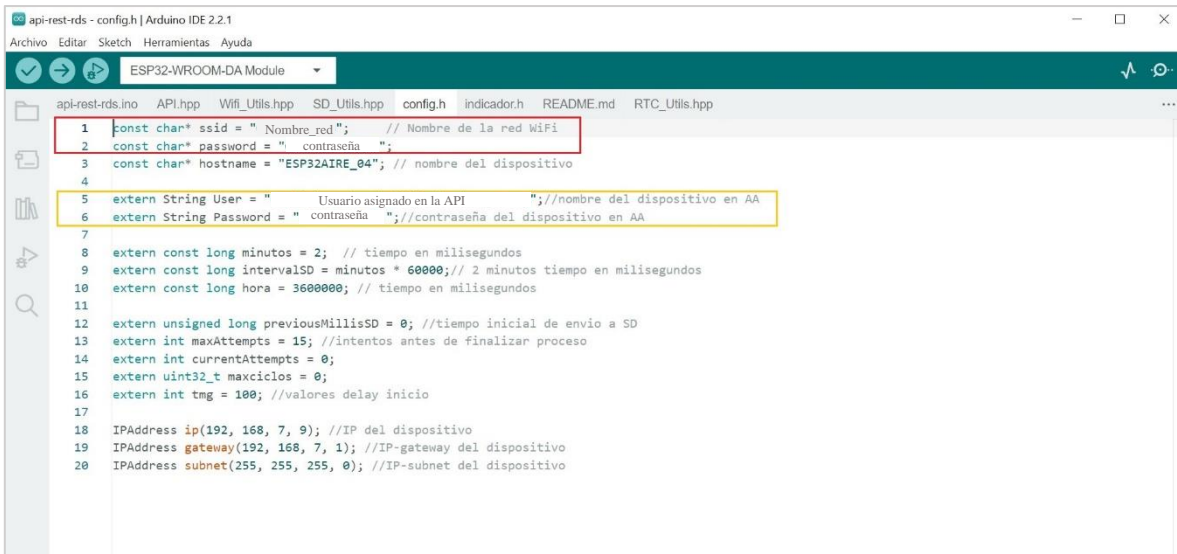
1 #include <SPI.h>
2 #include <SdFat.h>
3 #include "indicador.h"
4
5 #define SPI_SPEED_SD 4000000 // 4MHz
6 #define CS 5
7
8 SdFat SD;
9 File dataFile;
10
11 void SDSet(){
12     if (!SD.begin(CS, SPI_SPEED_SD))
13     {
14         Serial.println("Error al inicializar la tarjeta SD.");
15         errSD();
16         while (1);
17     }
18 }
19
20 bool crearArchivo(String archivo)
21 {
22     if (!SD.begin(CS, SPI_SPEED_SD))
23     {
24         Serial.println("Error al inicializar la tarjeta SD.");
25         errSD();

```

Acá también se establece el tipo de campo y separadores decimales.

- **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.



```

api-rest-rds - config.h | Arduino IDE 2.2.1
Archivo Editar Sketch Herramientas Ayuda
ESP32-WROOM-DA Module

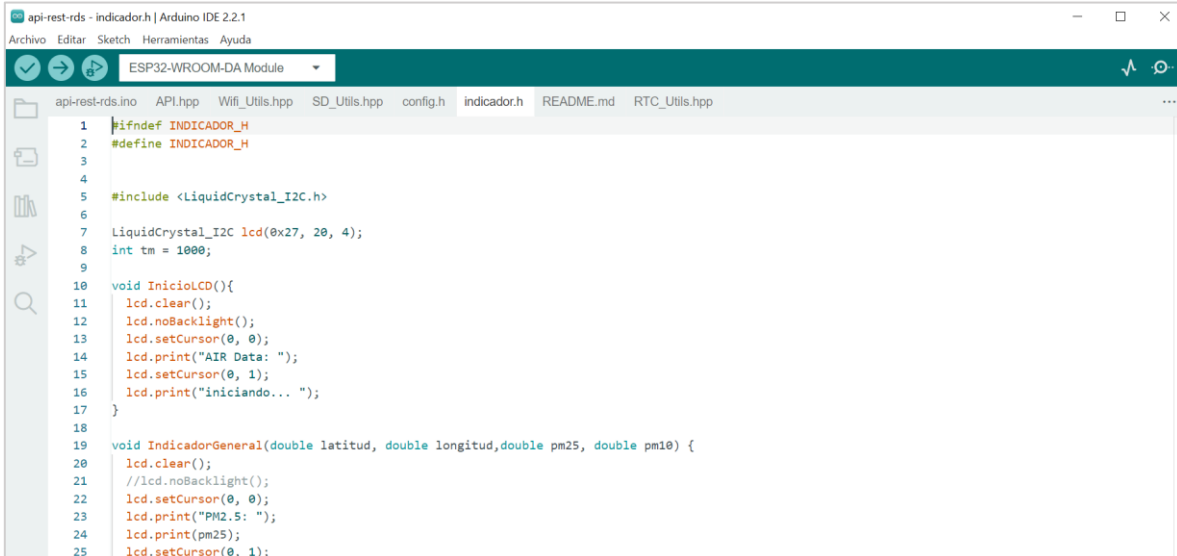
1 const char* ssid = "Nombre_red"; // Nombre de la red WiFi
2 const char* password = "contraseña ";
3 const char* hostname = "ESP32AIRE_04"; // nombre del dispositivo
4
5 extern String User = " Usuario asignado en la API "; //nombre del dispositivo en AA
6 extern String Password = " contraseña "; //contraseña del dispositivo en AA
7
8 extern const long minutos = 2; // tiempo en milisegundos
9 extern const long intervalSD = minutos * 60000; // 2 minutos tiempo en milisegundos
10 extern const long hora = 3600000; // tiempo en milisegundos
11
12 extern unsigned long previousMillisSD = 0; //tiempo inicial de envío a SD
13 extern int maxAttempts = 15; //intentos antes de finalizar proceso
14 extern int currentAttempts = 0;
15 extern uint32_t maxciclos = 0;
16 extern int tmg = 100; //valores delay inicio
17
18 IPAddress ip(192, 168, 7, 9); //IP del dispositivo
19 IPAddress gateway(192, 168, 7, 1); //IP-gateway del dispositivo
20 IPAddress subnet(255, 255, 255, 0); //IP-subnet del dispositivo

```

Importante: 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.

- **Indicador.h:**

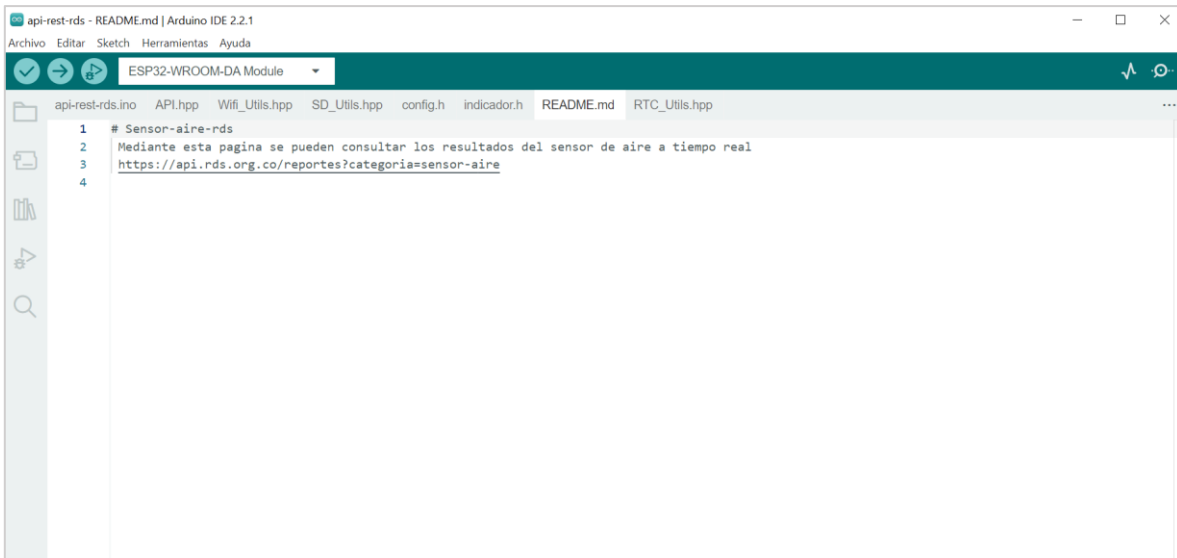
En esta pestaña se establecen parámetros para los mensajes que se reflejarán en la pantalla LCD, de manera que se muestren datos de PM 2.5, PM 10, latitud y longitud. Si el mensaje cambia indicará lo que está sucediendo con el sensor (envío a la API, guardar en CSV o si falla la SD).



```
1 #ifndef INDICADOR_H
2 #define INDICADOR_H
3
4 #include <LiquidCrystal_I2C.h>
5
6 LiquidCrystal_I2C lcd(0x27, 20, 4);
7 int tm = 1000;
8
9
10 void InicioLCD(){
11     lcd.clear();
12     lcd.noBacklight();
13     lcd.setCursor(0, 0);
14     lcd.print("AIR Data: ");
15     lcd.setCursor(0, 1);
16     lcd.print("iniciando... ");
17 }
18
19 void IndicadorGeneral(double latitud, double longitud, double pm25, double pm10) {
20     lcd.clear();
21     //lcd.noBacklight();
22     lcd.setCursor(0, 0);
23     lcd.print("PM2.5: ");
24     lcd.print(pm25);
25     lcd.setCursor(0, 1);
```

- **README.md:**

Acá se encontrará el enlace de la ventana de visualización para visualizar el envío de datos a la API.



```
1 # Sensor-aire-rds
2 Mediante esta pagina se pueden consultar los resultados del sensor de aire a tiempo real
3 https://api.rds.org.co/reportes?categoria=sensor-aire
4
```

- **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.


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

api-rest-rds.ino  API.hpp  Wifi_Utils.hpp  SD_Utils.hpp  config.h  indicador.h  README.md  RTC_Utils.hpp

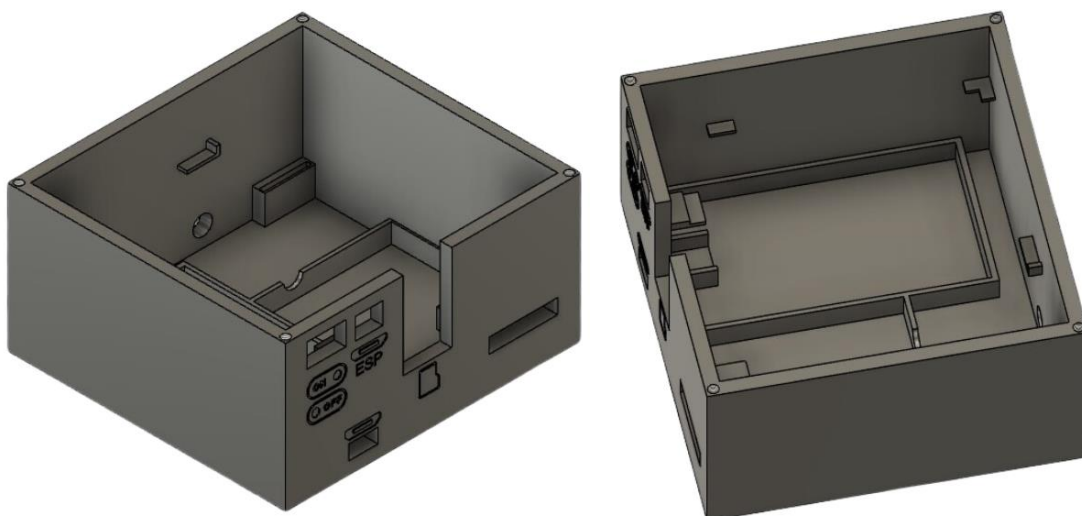
1  #include <RTClib.h>
2  #include "indicador.h"
3
4  RTC_DS1307 DS1307_RTC;
5
6  void RTCSet() {
7      if (!DS1307_RTC.begin()) {
8          Serial.println("No existe RTC");
9          errRTC();
10         while (1);
11     }
12
13     DateTime now = DS1307_RTC.now();
14
15     // Verificar si la hora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

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 para ubicar los elementos del dispositivo y permitir comunicación externa del sensor para la captura de material particulado, 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 15.2 cm x 14 cm x 7 cm y la tapa de 15.2 cm x 14 cm x 1 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/MrLtsyFC2da3JpN>

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 con la fecha de captura, 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 (,).

 1-2-2024-0	5/02/2024 2:38 p. m.	Archivo de valores separados por comas de Microsoft Excel	2 KB
--	----------------------	---	------

- PM25: Almacena los valores de material particulado de diámetro de 2.5 micrómetros, capturados por el sensor de partículas.
- PM10: Almacena los valores de material particulado de diámetro de 10 micrómetros, capturados por el sensor de partículas.
- Latitud: Datos de latitud capturados a partir del módulo GPS.
- Longitud: 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	PM25,PM10,Latitud,Longitud,Fecha,Hora				
2	6.90,11.10,4.628459,-74.069447,2024/2/1,12:1:29				
3	7.10,13.00,4.628433,-74.069455,2024/2/1,12:3:40				
4	8.00,16.40,4.628421,-74.069459,2024/2/1,12:5:41				

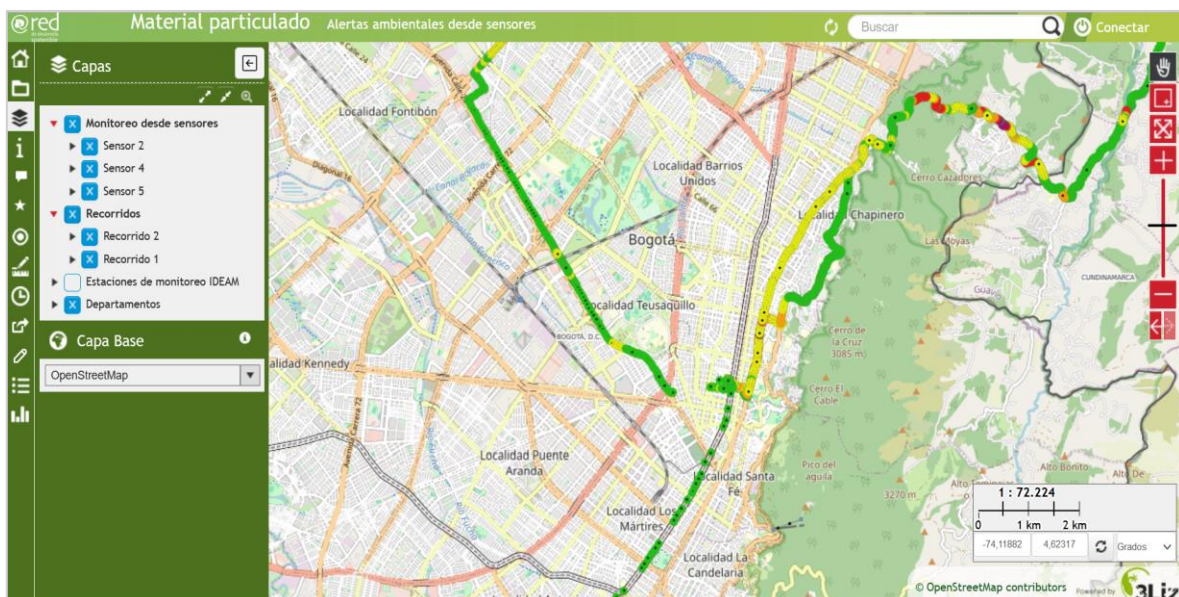
Los datos almacenados pueden ser trabajados y graficados a través de diferentes programas; los nombres de los campos son configurados sin caracteres especiales como puntos, comas, tildes, etc. Para evitar conflictos con el servidor.

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 “Material particulado”, disponible en:

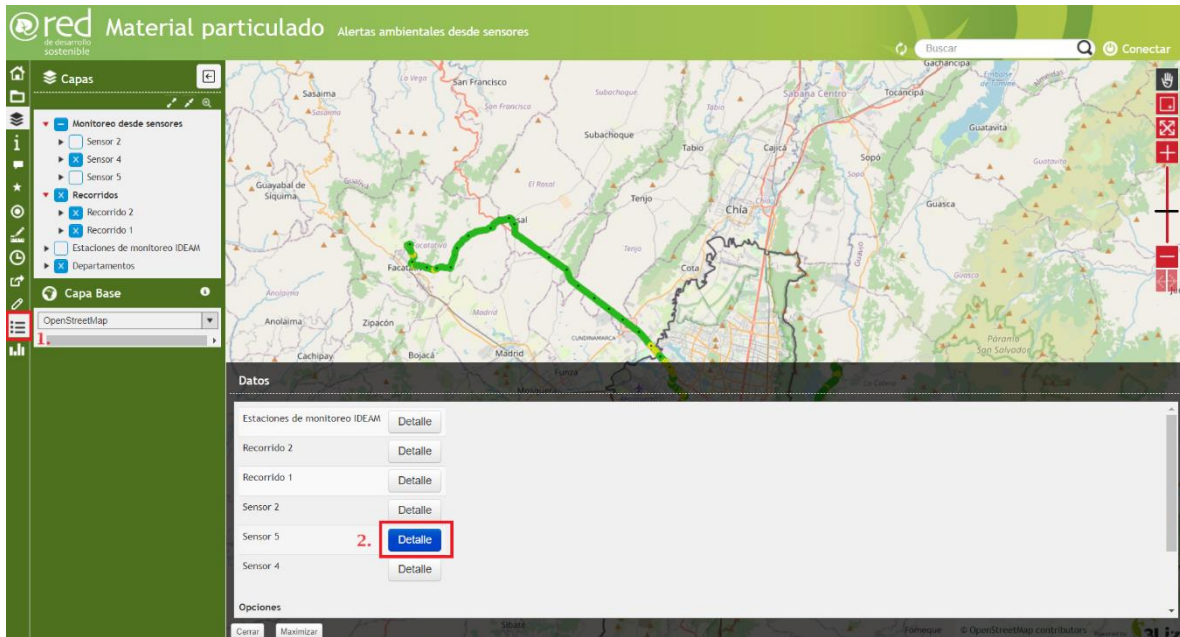
<https://rds.org.co/es/alertas-ambientales/alertas-dispositivo-aire>



Cada captura está 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:

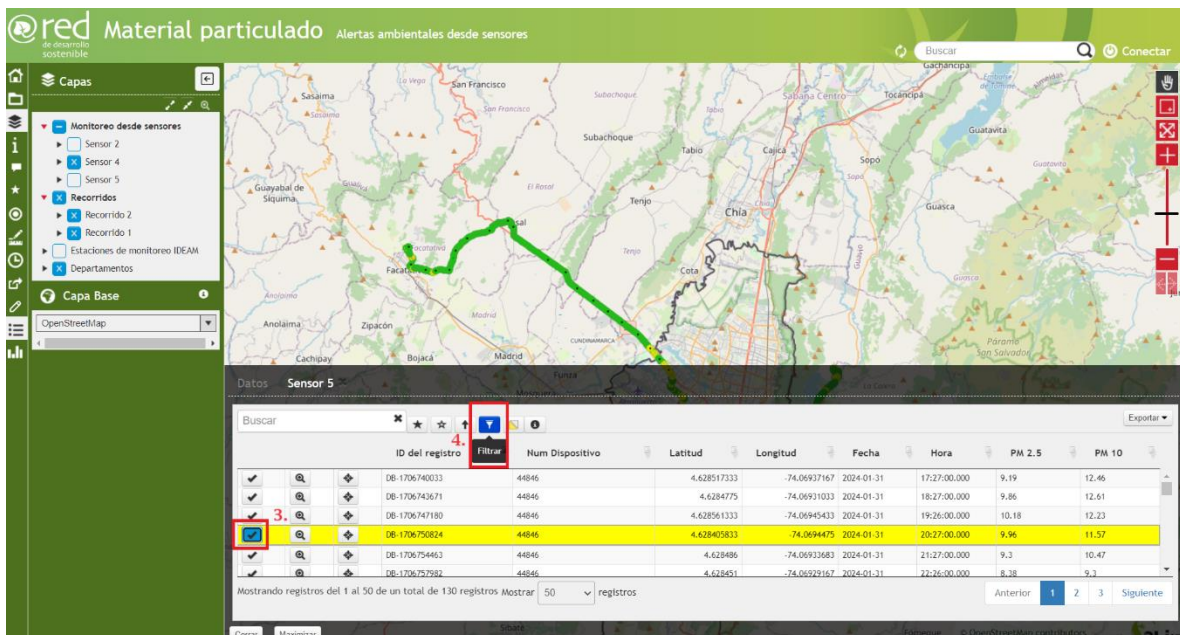


Herramienta Datos



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 AIRE							
Sensor que reporta	Categoría	PM25	PM100	LATITUD	LONGITUD	FECHAPUB	HORA
esp32airecolnodo_004@colnodo.apc.org	Aire	2.98	4.34	4.628226333	-74.06980733	2024/02/22	17:21
esp32airecolnodo_002@colnodo.apc.org	Aire	2.6	6.77	4.628361333	-74.06935	2024/02/22	16:55
esp32airecolnodo_005@colnodo.apc.org	Aire	2.59	4.37	4.628409167	-74.06956367	2024/02/22	16:27
esp32airecolnodo_004@colnodo.apc.org	Aire	764.65	1049.79	4.628335833	-74.06953917	2024/02/22	16:20
esp32airecolnodo_005@colnodo.apc.org	Aire	2.93	4.96	4.628331167	-74.069438	2024/02/22	15:26
esp32airecolnodo_004@colnodo.apc.org	Aire	3.33	5.13	4.628204333	-74.0695055	2024/02/22	15:22

Enlace de visualización: api.rds.org.co/reportes?categoria=sensor-aire

Recomendaciones

Mantener alejada la manguera del sensor de material particulado de superficies calientes que puedan derretir; evitar introducir elementos que puedan obstaculizar el paso del aire.

Evitar golpes o rayones sobre la pantalla LCD que puedan afectar su funcionamiento o visibilidad de los datos en esta.

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.