UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UNA PLATAFORMA INTEGRADA PARA LA MONITORIZACIÓN DE GASES VOLCÁNICOS: MEJORAS EN LA ESTACIÓN MÓVIL PARA LA DETECCIÓN DE DIÓXIDO DE CARBONO EN SUELO VOLCÁNICO

PRESENTADO POR:

COMAYAGUA MARTINEZ, JEFRY ORLANDO

SOLIS MARROQUIN, DAVID ADRIEL

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA ABRIL DE 2024

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. JUAN ROSA QUINTANILLA

SECRETARIO GENERAL:

LIC. PEDRO ROSALÍO ESCOBAR CASTANEDA

FACULTAD DE INGENIERÍA Y ARQUITECURA

DECANO:

ING. LUIS SALVADOR BARRERA MANCÍA

SECRETARIO:

ARQ. RAÚL ALEXANDER FABIÁN ORELLANA

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR INTERINO:

ING. WERNER DAVID MELÉNDEZ VALLE

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título

DESARROLLO DE UNA PLATAFORMA INTEGRADA PARA LA MONITORIZACIÓN DE GASES VOLCÁNICOS: MEJORAS EN LA ESTACIÓN MÓVIL PARA LA DETECCIÓN DE DIÓXIDO DE CARBONO EN SUELO VOLCÁNICO

Presentado por:

COMAYAGUA MARTINEZ, JEFRY ORLANDO

SOLIS MARROQUIN, DAVID ADRIEL

Trabajo de Graduación Aprobado por:

Docente Asesor:

DR. CARLOS OSMIN POCASANGRE JIMÉNEZ

SAN SALVADOR, ABRIL DE 2024

Trabajo de Graduación Aprobado por:

Docente Asesor:

DR. CARLOS OSMIN POCASANGRE JIMÉNEZ

NOTA Y DEFENSA FINAL

En esta fecha, lunes 18 de marzo de 2024, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 2:00 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Werner David Meléndez Valle
   Director Interino

   Firma

2. MSc. José Wilber Calderón Urrutia
   Secretario

   Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- DR. CARLOS OSMIN POCASANGRE JIMENEZ
  (Docente Asesor)

  Firma

- Dr. CARLOS EUGENIO MARTINEZ CRUZ

  Firma

- MSC. SALVADOR DE JESUS GERMAN

  Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

DESARROLLO DE UNA PLATAFORMA INTEGRADA PARA LA MONITORIZACIÓN DE GASES VOLCÁNICOS: MEJORAS EN LA ESTACIÓN MÓVIL PARA LA DETECCIÓN DE DIÓXIDO DE CARBONO EN SUELO VOLCÁNICO

A cargo de los Bachilleres:

- COMAYAGUA MARTÍNEZ JEFRY ORLANDO
- SOLIS MARROQUIN DAVID ADRIEL

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 8.8
(Ocho punto Ocho)

Agradecimientos.

Quiero expresar mi agradecimiento primeramente a Dios quien me ha concedido culminar este proceso académico con éxito, guiando mi esfuerzo y acompañando a mis compañeros de equipo en todas las actividades.

A mi familia, a mis padres Rosina Marroquín de Solis y David Solis Peña por su incondicional apoyo en todo momento aconsejándome, dándome ánimos y aliento. A mis hermanos, Diego Hamil Solis Marroquin por su compañía grata y amena conmigo, a Daniel Vladimir Solis Marroquin estudiante y futuro Ingeniero Informático quien me orientó en áreas específicas en este proyecto donde requería nuevos conocimientos de programación.

Quiero reconocer la dirección y orientación brindada por nuestro docente asesor, Dr. Carlos Osmín Pocasangre, así como la confianza depositada en nosotros por el Lic. Benancio Henríquez Miranda, quien nos facilitó el acceso al equipo y nos brindó conocimientos teóricos fundamentales para el desarrollo del proyecto sobre mediciones con la estación móvil para la detección de CO_2 en suelo volcánico. Agradezco también al Msc. Otoniel Flores por su valiosa orientación en el área de IoT. A todos ellos les estoy agradecido por su consideración, apoyo y paciencia, elementos clave para alcanzar un prototipo funcional mejorado.

Mi reconocimiento se extiende a mis compañeros de la Facultad Multidisciplinaria de Occidente con quienes construí buenas amistades así como mi grupo de amigos de la carrera de Ingeniería Eléctrica con los que estudiaba en especial a Rafael Armando Rivas y Christian Ivan Rivera, a todos ellos les deseo éxitos en la carrera profesional.

Mencionar a todos los docentes de la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería y Arquitectura y la Facultad Multidisciplinaria de Occidente, quienes me formaron académicamente en las diferentes áreas de la carrera profesional.

Particularmente agradecer a mi compañero de trabajo de graduación, a Jefry Orlando Comayagua con quien trabajamos arduamente para culminar este proyecto probando nuevas funcionalidades para la mejora de la estación móvil.

David Adriel Solis Marroquín.

Mis agradecimientos van dirigidos en primer lugar a Jehova Dios por darme fuerzas, sabiduría y perseverancia para completar este logro académico.

A mi familia, en especial a mi madre Silvia Aracely Martínez De Comayagua y a mi hermano y colega Kevin Josias Comayagua Martinez, por nunca dejarme solo e impulsarme a seguir en los momentos más difíciles. A mi padre Amilcar Comayagua por impulsarme con su ejemplo a tomar esta carrera.

Agradezco a los docentes involucrados en el desarrollo del trabajo de investigación, Dr. Carlos Osmin Pocasangre quien fue nuestro director de tesis y guía principal, el Lic. Benancio Henriquez Miranda por depositar su confianza en nosotros para trabajar en las mejoras de la estación móvil para detección de CO_2 e Msc. Ing. Otoniel Flores que nos brindó su apoyo con el conocimiento técnico necesario para poder realizar las modificaciones. Siempre estuvieron dispuestos a apoyarnos y depositaron su confianza en nuestro equipo de trabajo para modificar el prototipo funcional que se había construido previamente.

No olvidaré nunca a todos mis compañeros con los cuales compartí experiencias desde el principio de la carrera, mención especial para mi mejor amigo David Peña, Rene Sandoval y Daniel Monzón con quienes las risas no faltaron. Ellos también fueron fundamentales para superar todos aquellos retos académicos que se presentaron en el camino, deseando siempre que la vida los llené de éxitos en cada uno de sus proyectos.

Gracias a la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador por desarrollarme como profesional y brindarme el conocimiento necesario para desarrollar la investigación.

No olvidar agradecer a mi compañero David Adriel Solis Marroquin quien siempre estuvo dispuesto a apoyarme durante todo el trabajo de graduación a pesar de la distancia y las limitaciones que se presentaban, sin su apoyo no hubiera sido posible culminar este trabajo.

Jefry Orlando Comayagua Martinez

Contenido

Ilustraciones

Tablas

Fragmentos de Código

Programa del microcontrolador

Glosario

A

ADC
Convertidor de Analógico a Digital, dispositivo que convierte señales analógicas en digitales.....45

Android
Sistema operativo basado en Linux para dispositivos móviles.28, 29, 32, 36, 38, 78, 82, 86, 97, 115, 159, 168, 192

API
Interfaz de Programación de Aplicaciones, conjunto de reglas que permiten a los programas comunicarse entre sí..... 117, 120, 121, 123, 124, 150, 191

apk
Extensión de archivo para aplicaciones Android. 32, 37

APK
Paquete de Aplicación Android, el formato de archivo utilizado para distribuir e instalar software...38, 135

App
programa diseñado para realizar funciones específicas. 191

Arduino
plataforma de hardware para prototipos electrónicos. 31, 32, 33, 34, 44, 45, 50, 61, 73, 187, 190, 194

ASCII
Código Estándar Americano para el Intercambio de Información, codificación de caracteres 124

ATMEGA
Serie de microcontroladores de Atmel. 14, 44, 45

B

base64
Codificación Base64, método para codificar datos para transferirlos en medios que no son seguros 124

bits
Unidad básica de información en computación...45, 46

Bluetooth
Estándar para intercambio de datos inalámbrico.27, 31, 34, 36, 38, 39, 40, 43, 44, 45, 46, 61, 64, 65, 67, 79, 81, 82, 85, 87, 89, 90, 91, 96, 98, 99, 100, 106, 107, 108, 109, 110, 112, 160, 187, 188, 191, 192

Buck
Convertidor DC-DC que reduce voltaje. 50

C

CR2032
Batería de botón de litio. 49

CSS
Hojas de Estilo en Cascada, para definir la presentación de documentos HTML y XML 127

D

DC
Corriente continua, flujo eléctrico unidireccional. ...33, 34, 40, 50

df
Dataframe de Pandas 125, 131, 133, 134, 135, 141, 150

E

EEPROM
Memoria solo de lectura programable y borrable eléctricamente, un tipo de memoria no volátil.....33, 45, 46

ESP32

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Capítulo 1

1. Introducción General.

La estación móvil para la medición del flujo de CO_2 en suelo volcánico mediante la técnica de cámara de acumulación se posiciona como una herramienta esencial en el ámbito de la vigilancia volcánica. En este apartado, se sumergirá en una visión general acerca de la investigación del dióxido de carbono del suelo en entornos volcánicos, explorando la metodología de la cámara de acumulación y las bases teóricas que respaldan el cálculo del flujo de CO_2. Además de este enfoque, se presenta una visión detallada de la estación móvil, exhibiendo su funcionamiento, diseño original y los elementos que la componen. Estos componentes operan en sincronía con la aplicación "Medidor de CO_2", cuyo desarrollo y colaboración (Arriaza Carrillo & Aguilar Vega, 2022) encaminan a posicionar este proyecto como un destacado ejemplo de la Internet de las Cosas (IoT). El enfoque se centra en una mejora constante y una mayor automatización de los equipos de medición, con el propósito de optimizar las mediciones y enriquecer la experiencia del investigador y de los usuarios.

1.1 Investigación del Dióxido de Carbono del suelo volcánico.

La investigación del CO_2 en el suelo volcánico es un tema de gran importancia para comprender la actividad volcánica y evaluar los riesgos asociados siendo uno de los gases que se emiten durante la actividad volcánica, tanto por las fumarolas como por el suelo. El flujo de CO_2 volcánico es una medida de la cantidad de gas que se libera por unidad de superficie y de tiempo y monitorearlo puede ser útil para evaluar el estado del sistema hidrotermal y la amenaza volcánica.

El monitoreo volcánico es una actividad esencial en la prevención y mitigación de riesgos naturales asociados a la actividad volcánica. Los volcanes son estructuras geológicas que pueden experimentar erupciones explosivas o efusivas, lo que puede tener consecuencias devastadoras para las comunidades circundantes y el medio ambiente. Por lo tanto, el monitoreo constante y la vigilancia de la actividad

volcánica son cruciales para garantizar la seguridad pública y la toma de decisiones informadas. Esta actividad se basa en la recopilación, análisis y seguimiento de datos relacionados con la actividad volcánica, que incluyen:

1. Sismología: La detección de movimientos sísmicos es un indicador importante de la actividad volcánica, ya que puede señalar la ascensión de magma y la posible erupción. Sismómetros y geófonos se utilizan para registrar estos eventos.
2. Deformación del terreno: Cambios en la forma y elevación de un volcán pueden ser señales de que el magma se está acumulando debajo de la superficie. Esto se monitorea utilizando GPS, inclinómetros y estaciones totales.
3. Gases volcánicos: La emisión de gases como dióxido de azufre y dióxido de carbono es un indicador importante de la actividad del volcán. Estos gases se miden con analizadores de gases y estaciones de monitoreo de calidad del aire.
4. Temperatura: Las temperaturas inusuales en la superficie de un volcán pueden ser un signo de actividad. Las cámaras termográficas se utilizan para monitorear estos cambios.
5. Observación visual y cámaras: Los vulcanólogos realizan observaciones visuales directas de los volcanes y también utilizan cámaras para observar la actividad en tiempo real.
6. Modelado y análisis de datos: El procesamiento y análisis de datos geofísicos y geoquímicos son fundamentales para predecir posibles erupciones y emitir alertas tempranas.

En este caso, se centra en el monitoreo y análisis de los gases volcánicos, en específico la concentración de CO_2, esto ayuda a comprender mejor la actividad volcánica y puede proporcionar alertas tempranas en caso de riesgos para las comunidades circundantes. Si bien los volcanes emiten CO_2 durante sus erupciones, estas emisiones representan una pequeña fracción de las emisiones globales de dióxido de carbono. Sin embargo, pueden tener impactos significativos en áreas cercanas a los volcanes, tanto en términos de riesgos para la salud como en la toma de medidas de precaución. La monitorización y el estudio de las emisiones de CO_2 de los volcanes son esenciales para comprender y gestionar los riesgos asociados con la actividad volcánica.

El Salvador se ubica en el conocido cinturón de fuego del Pacifico, un 90% de su territorio se encuentra conformado por materiales volcánicos, con base a las investigaciones geológicas y a los reconocimientos de campo recientes en la Cordillera Volcánica se establecieron criterios específicos para la clasificación de volcanes activos, se determinó que en El Salvador existen 23 volcanes individuales y

se identificaron cinco campos volcánicos con antecedentes sísmicos que agrupan estructuras volcánicas y lagos cratéricos. En el caso del volcán de Santa Ana según estudios recientes presenta actividad continua (emisión de gases y excepcionalmente cenizas) (Phys.org, 2023).

1.1.1 Medición de flujo de CO_2 mediante el método de la cámara de acumulación.

Los volcanes emiten dióxido de carbono de dos maneras: durante las erupciones y a través del magma subterráneo. El dióxido de carbono del magma subterráneo es liberado a través de grietas, rocas y suelos porosos, además del agua que alimenta lagos volcánicos y manantiales termales. Las estimaciones de las emisiones de dióxido de carbono globales producidas por volcanes deben tomar en cuenta tanto las fuentes en erupción como aquellas que no lo están. Los volcanes son fuentes emisoras de una cantidad de distintos gases, los cuales son expulsados a través de su chimenea y del suelo del terreno. Monitorear las expulsiones de los gases permite encontrar sistemas de fracturas en el suelo (fallas geológicas), y mantener vigilancia de la cámara magmática.

Uno de los métodos que se utilizan para medir el flujo de CO_2 volcánico es el método de la cámara de acumulación. Este método consiste en colocar una campana de sobre el suelo y medir la concentración de CO_2 (Frondini, 2004) que se acumula dentro de ella con un espectrómetro portátil. La campana debe estar sellada lo más posible al suelo para evitar fugas de gas. El espectrómetro debe estar calibrado con una muestra de aire ambiente. El flujo se calcula a partir del cambio de concentración en función del tiempo. (Sierra, 2022)

El método con lleva algunas ventajas y desventajas. Entre las ventajas se encuentran su bajo costo, su facilidad de uso, su portabilidad y su rapidez, ha demostrado ser confiable para medir el flujo y varios estudios han validado este método encontrando una buena correlación entre sus mediciones (Finizola, Sortino, Lénat, Valenza, & Boschi, 2003) (Revil, y otros, 2008).Entre las desventajas se encuentran su sensibilidad a las condiciones ambientales, como la temperatura, la humedad, el viento y la presión atmosférica, su dependencia del tamaño y la forma de la campana, su interferencia con otros gases presentes en el suelo, como el O_2 (oxígeno) y el N_2 (nitrógeno) y H_2O (vapor de agua) , y su posible alteración del equilibrio del gas en el suelo. En la Ilustración 1se muestra un esquema sencillo de este método que se ha utilizado desde (Arriaza Carrillo & Aguilar Vega, 2022). Los cuales crearon un sistema eficiente para la medición del flujo de CO_2 utilizando un sensor electroóptico LI-830 controlado por un CPU desarrollado que a su vez se controla desde una aplicación móvil por Bluetooth.

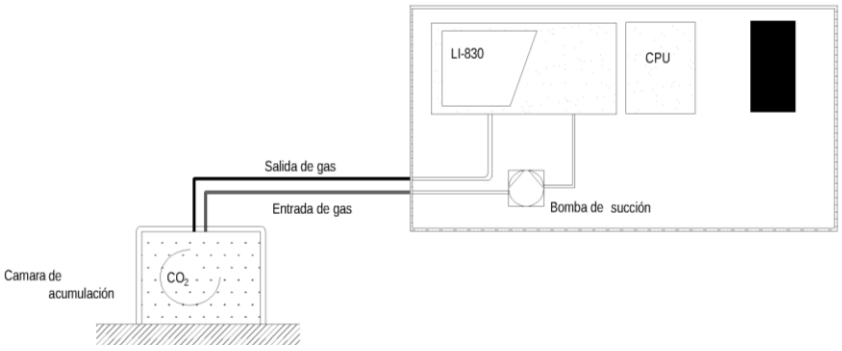


Ilustración 1 Esquema de medición de flujo de CO_2 con la estación móvil

Se observa que en la cámara se conectan dos mangueras las cuales una de ellas succiona el gas gracias a una bomba de succión, este gas se lleva al sensor electroóptico LI-830 y los resultados se leen en el CPU. El procedimiento de medición es sencillo con la estación móvil, puesto que solo se necesita colocar la cámara de acumulación en el suelo y mediante una aplicación Android llamada "Medidor de CO_2" se controla el CPU y la bomba que succiona el gas acumulado y lo hace pasar al sensor de CO_2 por medio de las mangueras.

La teoría del método (Frondini, 2004), indica que la razón de cambio con respecto al tiempo, la derivada o la pendiente de la gráfica de concentración del gas, se utiliza para estimar el flujo, la Ilustración 2 muestra un resultado típico de una medición utilizando la estación móvil.

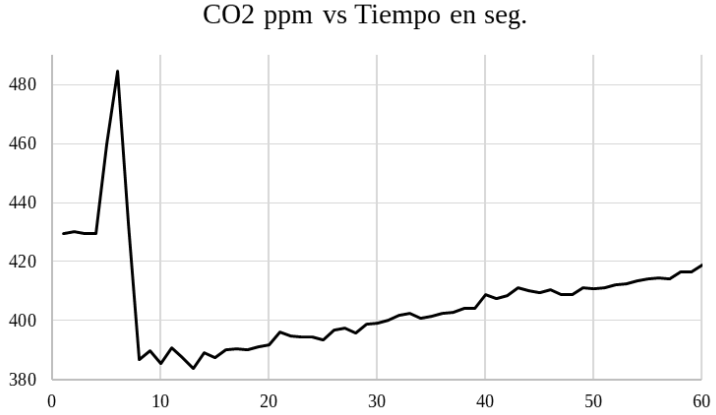


Ilustración 2 Gráfico de concentración de CO_2 vs tiempo en segundos

Se observa un comportamiento líneal de la concentración de CO_2 que va incrementado con una pendiente positiva, entonces una manera de estimar la cantidad de flujo es realizando una regresión lineal a partir del punto donde la gráfica empieza a tener ese comportamiento. Se observa que del segundo 0

hasta el segundo 7 aún no se había colocado la cámara de acumulación en el suelo y del segundo 7 al 60 se colocó y se mantuvo haciendo lecturas en el sensor, ésta es la parte de los datos que interesa para aplicar la regresión lineal y calcular la pendiente y su coeficiente de correlación lineal que tiene que ser lo más cercano a 1, la Ilustración 3 muestra el resultado de este procedimiento.

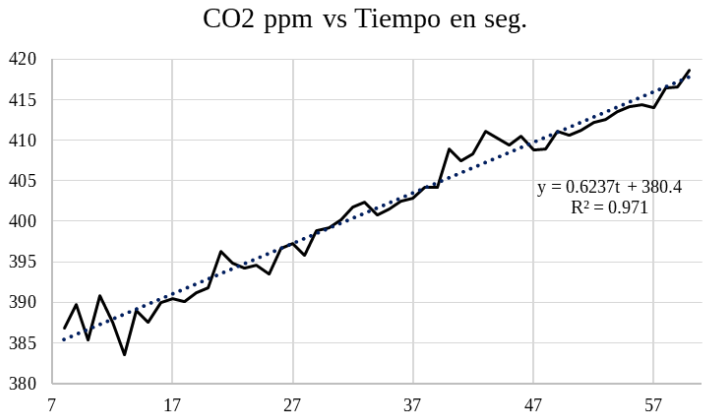


Ilustración 3 Regresión lineal a la medición de concentración con tendencia lineal

Como resultado para esta medición se obtiene que el flujo de CO_2 es 0.6237 ppm/s siendo una estimación aproximada.

Para obtener una estimación precisa del flujo de CO_2 volcánico, se debe realizar un muestreo representativo en la zona de interés y aplicar técnicas geoestadísticas para elaborar un mapa de la emisión de CO_2 . La mejora del equipo de la estación móvil significa una automatización mayor, control efectivo acerca de los datos en las lecturas teniendo un registro ordenado y organizado para la investigación y sirviendo como prototipo para otros equipos de medición u otras variables de interés que se puedan incorporar a este.

1.2 Descripción inicial de la estación móvil.

La estación móvil para la medición de CO_2 su diseño y utilización se describe detalladamente en (Arriaza Carrillo & Aguilar Vega, 2022), realizando un breve resumen, el equipo consta de 2 elementos, la aplicación Android y la maleta que contiene el CPU, bomba de succión y el sensor LI-830.

1.2.1 Descripción del Hardware.

1.2.1.1 Elementos de la estación móvil.

1.2.1.1.1 Sensor de dióxido de carbono LI-830 :

Sensor de alto rendimiento, es un analizador de gas que mide la concentración de dióxido de carbono en el aire. Utiliza la tecnología de infrarrojos no dispersivos (NDIR) (Biosciences, 2023). NDIR funciona midiendo la cantidad de luz infrarroja absorbida por el dióxido de carbono en una muestra de aire.

El sensor utiliza una fuente de luz infrarroja, que se emite a través de una cámara de referencia y una cámara de medición, la cámara de referencia contiene aire limpio sin CO_2, mientras que la cámara de medición contiene la muestra de gas que se está analizando. La luz infrarroja se transmite a través de ambas cámaras (Testo, 2023) y se detecta en el otro extremo por un detector de luz, la cantidad de luz que se detecta en la cámara de referencia es comparada con la cantidad de luz que se detecta en la cámara de medición, y la diferencia en la cantidad de luz absorbida por el CO_2 en la muestra de gas se utiliza para determinar la concentración de éste en el aire, en la Ilustración 4 (Scientific, 2021), se puede visualizar el funcionamiento del sensor.

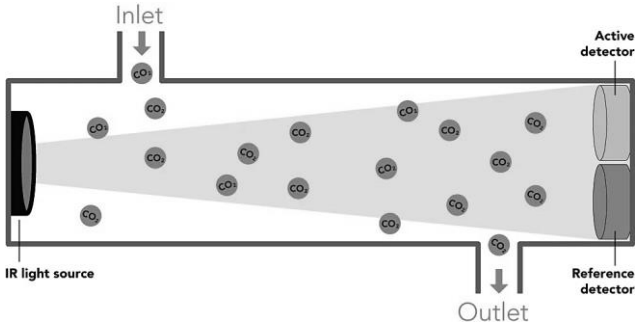


Ilustración 4 Diagrama de un sensor NDIR de CO_2. Fuente:“How Does An NDIR CO_2 Sensor Work?,” Atlas Scientific.

El LI-830 utiliza un sensor NDIR para medir la concentración de CO_2. El sensor NDIR del LI-830 tiene una fuente de luz infrarroja que emite luz infrarroja a una longitud de onda de 4.26 µm. Esta longitud de onda es absorbida por el CO_2, pero no por otros gases comunes en el aire, como el oxígeno (O_2) y el nitrógeno (N_2) (Scientific, 2021).

Para las necesidades específicas del proyecto se ha personalizado la programación del sensor LI-COR 830 enfocándolo exclusivamente en la medición de CO_2 por ello se han deshabilitado otras variables de medición (Arriaza Carrillo & Aguilar Vega, 2022) como: la tasa de flujo de gas, temperatura, la presión de la celda, la tensión en la celda, la absorción de CO_2, el punto de rocío del agua, los datos en bruto, entre otros.

1.2.1.1.2 Maleta.

La Pelican Vault V200, es una maleta de transporte resistente y duradera diseñada para proteger equipos sensibles. Esta maleta está hecha de polímero estructural de alto impacto y fabricado con una junta tórica que proporciona una protección hermética al agua y al polvo. En su interior contiene espuma de alta densidad que se puede adaptar a la forma del equipo a transportar. La maleta además cuenta con dos cierres de pestillo de fácil apertura, un sistema de bisagra resistente y asas de transporte ergonómicas para facilitar su manipulación (Arriaza Carrillo & Aguilar Vega, 2022).

Sus dimensiones son 39.1 x 33.2 x 15.6 cm al exterior y 35.6 x 25.4 x 14 cm al interior.

1.2.1.1.3 Cámara de Acumulación.

Es una cámara de aluminio sin fuga que mediante dos mangueras transportan las muestras de aire con ayuda de una bomba 12V al sensor LI-830. El objetivo de la cámara es evitar la fuga de gas y se capte sólo la emanación de éste desde el suelo (Arriaza Carrillo & Aguilar Vega, 2022).

1.2.1.1.4 CPU.

- Módulo Bluetooth HC 06:

El HC 06 tiene un alcance de hasta 10 metros, dependiendo de las condiciones del entorno, y funciona con una tensión de alimentación de 3.3 V a 5 V, y una transmisión de hasta 2.1 Mbps y una tasa de baudios ajustable. Es un módulo de bajo costo, una de las principales ventajas de este módulo es su fácil integración con Arduino, actuando como esclavo (Mechatronics, Naylamp Mechatronics, 2023).

- Sensor de Temperatura y Humedad DTH22:

Es un sensor de temperatura y humedad relativa de alta precisión. El DHT22 se comunica a través de un solo cable de datos, utilizando el protocolo de comunicación de un solo bus, lo que lo hace fácil

de usar con microcontroladores y dispositivos electrónicos. El sensor DHT22 tiene una resolución de 0.1°C para la medición de temperatura y 0.1% para la medición de humedad relativa. La medición se realiza en la estación cada 10 segundos para dar prioridad a la información obtenida de la medición del LI COR 830. (Mechatronics, Tutorial: Sensor de temperatura y humedad DHT11 y DHT22, 2016)

- Módulo GPS Neo 6M:

Es un módulo GPS que utiliza el sistema de navegación por sistema para proporcionar información de posición precisa y confiable. Utiliza una antena cerámica para la recepción de señales GPS, y puede recibir señales de hasta 22 satélites simultáneamente. Tiene un consumo de energía bajo, lo que lo hace adecuado para proyectos de bajo consumo energético, y funciona con una tensión de alimentación de 3,3V o 5V (Lab, 2023).

- Módulo convertidor RS232 a TTL Max232:

La conexión entre el microcontrolador de Arduino y el sensor LI COR 830 no se puede hacer de manera directa debido a que el protocolo RS232 utiliza variaciones de voltaje que van desde -13V hasta 13V, mientras que el microcontrolador trabaja a un nivel de voltaje TTL de 3.3V/5V. Para solucionar este problema se emplea un convertidor de niveles lógicos MAX232, que permite convertir las señales de transmisión serial RS232 a niveles TTL y viceversa. El MAX232 es un circuito integrado que integra la función de generación de tensiones positivas y negativas a partir de una única fuente de alimentación (Hub, 2023).

- Relé 5V 1 Canal:

El Relé 5V 1 Canal es un dispositivo electromecánico que se utiliza para controlar el encendido y apagado de cargas eléctricas a través de una señal digital de entrada. Controla la bomba del equipo de medición mediante una señal enviada desde la apk Android desarrollada. Su función es controlar la eficiencia energética del equipo ya que la bomba al encender consume mucha energía.

Para utilizar el Relé 5V 1 Canal con Arduino, es necesario conectar la señal digital de salida del microcontrolador al pin de entrada del relé y proporcionar la alimentación de 5V al dispositivo. De esta manera, al enviar una señal digital de 5V al relé, se activará la bobina y se cerrará el circuito eléctrico, permitiendo el encendido de la carga eléctrica. Por el contrario, al enviar una señal digital de 0V, se desactiva la bobina y se abre el circuito eléctrico, apagando la carga eléctrica.

- Convertidor DC-DC LM2596:

Es un regulador de voltaje DC-DC que proporciona una solución eficiente y económica para convertir una fuente de alimentación de tensión alta a tensión baja, capaz de proporcionar una salida de voltaje regulada y estable a partir de una entrada de voltaje variable, específicamente convierte 12Vdc a 5Vdc.

El diagrama del LM2596 incluye una fuente de alimentación de entrada, un filtro de entrada, un regulador de voltaje LM2596, un inductor de conmutación, un diodo de conmutación, un condensador de salida y un filtro de salida. La fuente de alimentación de entrada proporciona una tensión de entrada no regulada al circuito, que se filtra mediante el filtro de entrada para eliminar cualquier ruido o interferencia.

- Microcontrolador ATMEGA328P:

La placa de Arduino utiliza el ATmega328P como microcontrolador principal, tiene una velocidad de reloj de hasta 20 MHz, 32 KB de memoria flash para almacenamiento de programas, 2 KB de memoria SRAM para almacenamiento de datos, y 1 KB de memoria EEPROM para almacenamiento no volátil. Aquí reside todo el procesamiento de los datos medidos por los sensores y el adecuado funcionamiento de los demás módulos siguiendo las instrucciones programadas actualmente (Microchip, 2023).

1.2.1.1.5 Bomba De Vacío A 12V.

La única función que posee la bomba es succionar el gas a través de una manguera y llevarlo al sensor LI-830. Ésta es controlada desde el ATMEGA328P y se energiza por medio del Relé de 5V 1 Canal. La Bomba funciona con 12V así que el Relé actúa como “interruptor” para que ésta no esté energizada durante todo el funcionamiento de la estación móvil y consuma toda la batería, ya que al ser un motor el porcentaje de energía que consume es mucho mayor que al de los demás componentes.

1.2.1.2 Conexionado inicial del CPU

El conexionado se proporciona gracias a (Arriaza Carrillo & Aguilar Vega, 2022), Tabla. 5, se centrará en las conexiones del microcontrolador ATMega328P a los demás módulos.

Tabla 1

Conexiones del Atmega328P a los Módulos y Sensores (Arriaza Carrillo & Aguilar Vega, 2022)

PIN	Conexión a: Módulo	PIN
D7	Relé	IN
D10	Convertidor RS232 a TTL	TXD
D11	Convertidor RS232 a TTL	RXD
D3	GPS NEO6M	RXD
D4	GPS NEO6M	TXD
D2	Sensor Temp/Humedad	DATA
VIN	Convertidor DC/DC	OUT+
GND	Convertidor DC/DC	OUT-
RX0	Bluetooth	TXD
TX1	Bluetooth	RXD
GND	Todos los módulos	GND
5V	Bluetooth + Relé + SensorT/H	VCC
3V3	GPS + Convertidor RS232 a TTL	VCC

Estas son las conexiones iniciales en la cual se encuentra el CPU, puesto que son varios módulos en el código de Arduino es necesario comunicarse con ellos a través de SoftwareSerial. En la Ilustración 5 de (Arriaza Carrillo & Aguilar Vega, 2022) se puede observar un diagrama hecho en Fritzing acerca del conexionado del CPU (se ha omitido el conexionado del relé a la bomba de 12V y a la batería y del sensor LI-830 al convertidor Max232).

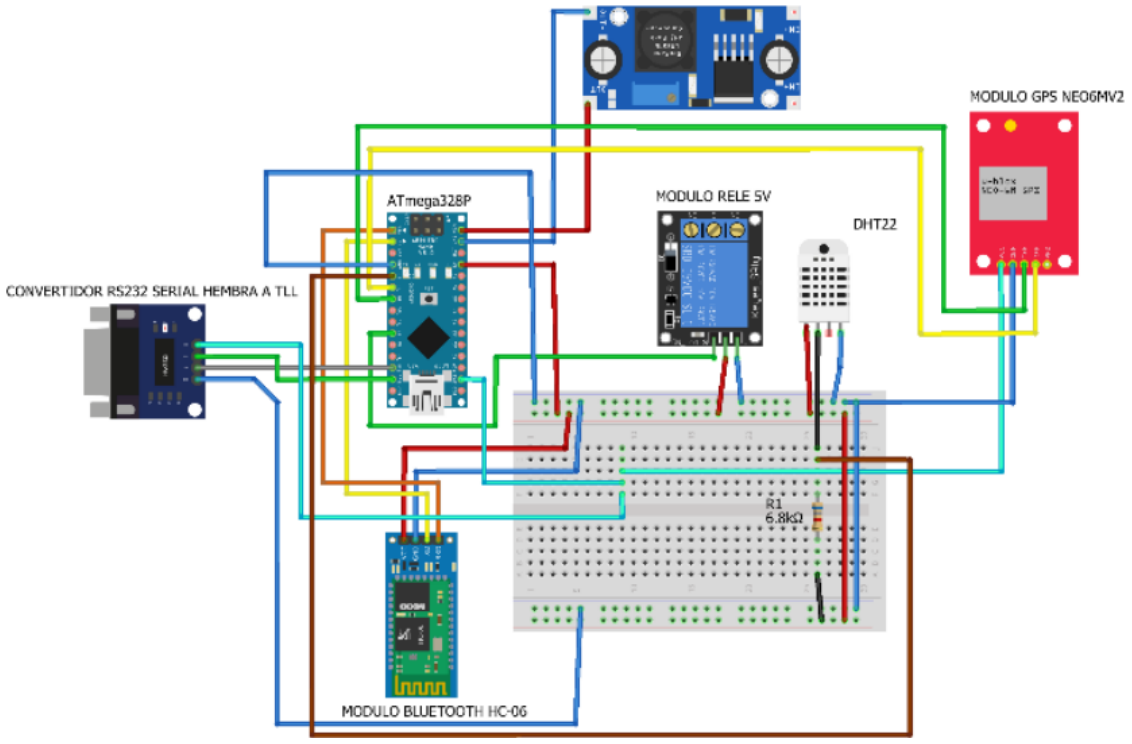


Ilustración 5 Diagrama de CPU (Arriaza Carrillo & Aguilar Vega, 2022)

Debido a que se trata de una estación móvil se hizo un estudio para el dimensionamiento de la batería que resulto en utilizar una batería de plomo-ácido de 7Ah (Arriaza Carrillo & Aguilar Vega, 2022) recargable, este cálculo se hizo sumando la potencia necesaria para cada módulo o sensor resultando en 8W y estimando un tiempo de medición de 6 horas.[1][2]

1.2.2 Descripción del Software.

El programa del microcontrolador anexado de (Arriaza Carrillo & Aguilar Vega, 2022), en resumen está diseñado para realizar dos tareas principales, la de la adquisición de datos por parte de los módulos

[1] Es necesario corroborar el estado de la batería antes de usar el equipo dado que si ésta descargada y marca por debajo de 12V ninguno de los componentes (sensor LI-830, bomba de vacío) que necesitan esta tensión funcionará.

[2] El cálculo de la batería asume un trabajo continuo por parte de la bomba de vacío.

y sensores, y la comunicación por Bluetooth que envía estos datos a la aplicación Android para que sean procesado por ésta.

En la fase de adquisición de datos, el programa establece la comunicación serial con los módulos Max232 y Neo 6M mediante la librería SoftwareSerial, conectados a los pines digitales tal como se indica en la Tabla 1 Conexiones del Atmega328P a los Módulos y Sensores del esquema de conexión con el microcontrolador. Esto se realiza para cumplir con el requisito de un flujo de datos constante. En cuanto a la toma de lecturas, se prioriza la frecuencia de captura de la concentración de CO_2, llevándose a cabo aproximadamente cada 1 segundo.

Por otro lado, las mediciones de temperatura y humedad, obtenidas a través del sensor DHT22, se realizan cada 10 segundos. En el caso de las lecturas de geolocalización, estas se llevan a cabo únicamente cuando son solicitadas desde la aplicación móvil.

En la tarea de comunicación a través de Bluetooth con la aplicación "Medidor de CO_2", el módulo HC-06 cumple la función de establecer dicha comunicación, encargándose tanto de recibir como de enviar datos. Para el envío de información, se elabora un String denominado "StrDataResp", el cual se estructura utilizando comas (",") de la siguiente manera:

```
CO2,BombaON,Latitud,Longitud,Satelites,Precisión,Temperatura,Humedad#
```

Para entender cómo se realiza el envío de este String por medio de Bluetooth se vinculó el CPU con otra app llamada "Serial Bluetooth Terminal" la cual es una consola/terminal (Morich, 2023) y con ella se visualiza claramente como un "Monitor Serial Bluetooth". Entonces al ejecutarse el programa del CPU se obtiene lo siguiente:

```
4.3e2,0,0,0,0,0,0,0#
4.0e2,0,0,0,0,0,0,0#
4.2e2,0,0,0,0,0,0,0#
4.6e2,0,0,0,0,0,0,0#
4.7e2,0,0,0,0,0,0,0#
4.7e2,0,0,0,0,0,0,0#
```

```
4.8e2,0,0,0,0,0,0,0#

4.9e2,0,0,0,0,0,0,0#

0,0,0,0,0,0,28.00,71.50#

0,0,0,0,0,0,28.00,71.50#
```

Cuando la lectura de geolocalización es requerida por ejemplo el String arroja lo siguiente:

```
   0,0,LAT =13.846584,LON =-89.2627426,SAT =6,PRE =126,0,0#

   0,0,LAT =13.846584,LON =-89.2627426,SAT =6,PRE =126,0,0#
```

La aplicación Medidor de CO_2, se encarga de interpretar este String separado por comas y separar cada dato correspondiente. Ahora, ésta también le envía comandos al CPU indicándole que realice diferentes acciones, los comandos que envía son caracteres que son interpretados por el programa del microcontrolador, los cuales son:

- F#: Detiene la lectura de temperatura y humedad por parte del DHT22.
- G#: Enciende la bomba de vacío.
- H#: Apaga la bomba de vació.
- P#: Solicita la lectura de geolocalización.

El encendido y apagado de la bomba no es más que el control del pin D7 del relé (Tabla 1) ya que éste la energiza o des energiza. Cada vez que la bomba se enciende es porque se generará el gráfico de concentración de CO_2 vs tiempo de la Ilustración 2 con el método de la cámara de acumulación en la aplicación Medidor de CO_2.

1.2.3 Descripción general de la apk “Medidor de CO_2”.

Ilustración 6 Logo de la apk

La aplicación móvil (APK) es una pieza fundamental en la operatividad coordinada del CPU y todos los componentes de la estación móvil. Es responsable de supervisar y controlar las acciones de la estación, en especial durante el proceso de toma de mediciones, como se ha detallado en secciones anteriores. Su funcionamiento es fácil y accesible, siguiendo simplemente los pasos descritos en (Arriaza Carrillo & Aguilar Vega, 2022). Desarrollada en Android Studio utilizando Java como lenguaje de programación, la aplicación garantiza una comunicación fluida a través de Bluetooth, recibiendo datos en tiempo real del CPU. Estos datos son utilizados para generar gráficos ilustrativos, como se muestra en la Ilustración 2 y la Ilustración 7:

Ilustración 7 Gráfica en tiempo real de concentración de CO_2 vs tiempo de la aplicación Medidor de CO_2

La gráfica es generada gracias a “hellochart library”, la manera de hacerlo es redibujando el vector que almacena las lecturas de concentración de CO_2 recibidas, de esta manera el gráfico crece a medida que el vector se hace más grande.

Asimismo, la aplicación cuenta con una funcionalidad para llevar a cabo análisis de regresiones lineales. En la Ilustración 7, se presenta otro conjunto de mediciones similares al mostrado en la Ilustración 2. Se observa una tendencia lineal que se establece aproximadamente a los 5 o 10 segundos de inicio de la medición, extendiéndose hasta los 50 segundos. Este patrón indica la presencia de la cámara de acumulación en el suelo, con la bomba activada para extraer el gas de dicha cámara. Dentro de las capacidades de la aplicación, se incluye un método para calcular regresiones lineales entre dos límites específicos, lo cual se aplica cuando la gráfica muestra un comportamiento análogo.

Ilustración 8 Cálculo de regresión lineal entre dos límites de la gráfica concentración de CO_2 vs tiempo

Para hacer esto se involucran varios método del programa de Java desde la creación de un CSV con los datos de la gráfica, la obtención de los límites, hasta la búsqueda de estos para realizar la regresión lineal y calcular el coeficiente de correlación R^2.

Al utilizar la aplicación, es fundamental otorgar los permisos de almacenamiento y Bluetooth. En caso contrario, la aplicación se cerrará debido a una "SecurityException". Este aspecto crítico se abordará en este proyecto, junto con otros detalles y problemas a nivel de código. Se implementará una funcionalidad en la cual se enviará por Bluetooth un nuevo comando en forma de carácter al CPU, se incluirá un método para reiniciar la medición y se evitará el proceso de reconexión por Bluetooth al CPU. Además, se llevarán a cabo mejoras significativas en la interfaz a través de modificaciones en el XML del programa. Todos estos aspectos serán detallados minuciosamente en el Capítulo 3.

Capítulo 2.

2 Rediseño del CPU.

A continuación, se abordarán los aspectos del CPU, incorporando mejoras que abarcan tanto un rediseño a nivel de hardware como de software. Inicialmente, el CPU descrito en el Capítulo 1 operaba exclusivamente en conjunto con la aplicación móvil, un enfoque que se mantendrá ya que representa el método óptimo para controlar el equipo a través de la conexión Bluetooth con la aplicación móvil. El rediseño del hardware implica cambios significativos, entre ellos la sustitución del microcontrolador, la incorporación de dos módulos adicionales (el módulo GSM SIM800L y un módulo RTC), así como el reemplazo del módulo convertidor DC-DC, anteriormente un LM2596, por un módulo convertidor DC-DC con tecnología QC3.0. Estas modificaciones se justificarán en este capítulo. Por otro lado, a nivel de software, el programa para el nuevo microcontrolador será enriquecido con una tarea adicional que añadirá complejidad al código. Esta tarea consiste en establecer comunicación GPRS a través del módulo GSM y enviar las lecturas de los sensores y módulos directamente a una hoja de cálculo en Google Spreadsheet.

2.1 Diseño del Hardware.

Al iniciar el proceso de rediseño, resulta crucial examinar las funcionalidades del microcontrolador y evaluar las nuevas necesidades que han surgido, especialmente en lo concerniente a la comunicación Bluetooth y la conexión GPRS. En este contexto, se determina que no hay un dispositivo más idóneo para abordar estas tareas que el TTGO-TCall SIM800L. Esta placa de desarrollo, que incorpora el ESP32 y ya integra el chip SIM800L, se presenta como la opción más apropiada para afrontar estos requerimientos.

2.1.1 Placa de desarrollo TTGO-TCall SIM800L

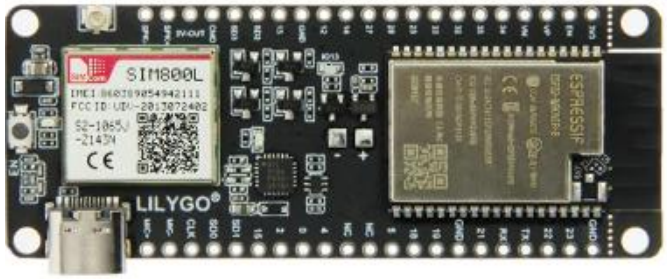


Ilustración 9 Módulo LILYGO-T-CALL. (Xinyuan-LilyGO, 2023)

La placa ha sido desarrollada por LILYGO, y para su implementación se requiere un proceso de aprendizaje continuo y consulta constante, aspectos que se encuentran disponibles en (Xinyuan-LilyGO, 2023), donde se proporciona un soporte constante para su utilización. Puesto que al ser parecido a una placa NodeMCU con el ESP32, el uso de los pines puede cambiar, sobre todo si se está utilizando el chip SIM800L.

Con respecto al patillaje o los pines en la Ilustración 10 se puede observar de mejor manera.

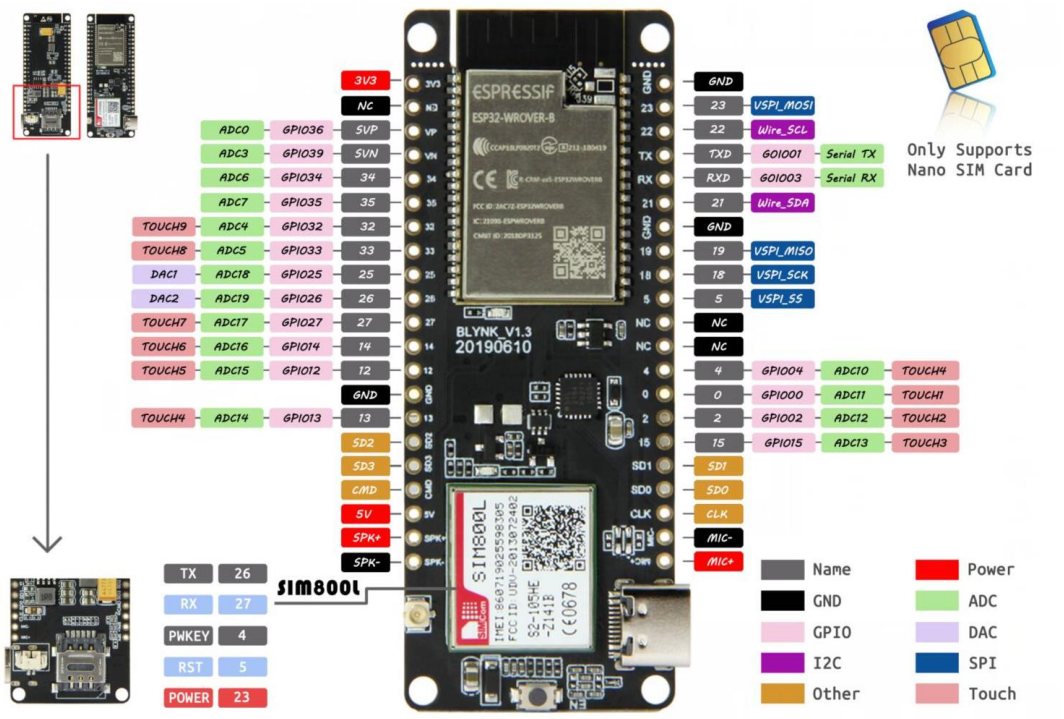


Ilustración 10 Pines de la TTGO T-Call SIM800L (Xinyuan-LilyGO, 2023)

El TTGO T-Call SIM800L presenta un patillaje similar al de una NodeMCU o incluso una Wemos D1 R32 (ESPduino). La diferencia principal radica en que el TTGO T-Call SIM800L dispone de pines que son exclusivos para el chip SIM800L. Esta característica puede representar tanto una ventaja como una desventaja: la ventaja reside en que al estar el chip GSM ya integrado, no hay preocupaciones sobre cómo suministrar energía una vez que se inicializan los pines que lo controlan (26, 27, 4, 5, 23). No obstante, la desventaja es evidente cuando se intenta integrar módulos que necesiten estos pines, como es el caso de los módulos para tarjetas SD. Hasta el momento, no se ha encontrado una solución clara en la documentación (Xinyuan-LilyGO, 2023) ni en la comunidad para usar simultáneamente estos módulos en la tarjeta de desarrollo. Por otro lado sufre de la ausencia de los pines seriales 16 y 17 con los cuales se pueden establecer comunicación mediante HardwareSerial, por lo que el programa seguirá utilizando SoftwareSerial para simular puertos seriales y abrir la comunicación con los diferentes módulos.

En cuanto al consumo de energía según la documentación (Xinyuan-LilyGO, 2023), tomando en cuenta su voltaje de funcionamiento de 5V para el puerto USB, la corriente máxima aproximada que necesita la placa de desarrollo con el SIM800L activo es de 70mA, lo cual se traducen en 0.35W de

potencia necesaria para funcionar. Éste dato se utilizará para sondear las horas de trabajo con la que funcionará la batería con el nuevo CPU.

2.1.2 ESP32 Wrover-B.

El ESP32 es una denominación de microcontroladores de alto rendimiento que ofrece una amplia gama de características y funcionalidades siendo de bajo coste y consumo de energía ultra bajo, con tecnología WiFi y Bluetooth de modo dual integrada, lo que permite la conexión a redes inalámbricas y la comunicación con otros dispositivos.

El ESP32 cuenta con doble núcleo de procesamiento, lo que permite ejecutar múltiples tareas de manera simultánea y mejorar el rendimiento general del sistema (Systems, Espressif , 2023) (System, 2021) (Systems, Espressif Systems, 2023).

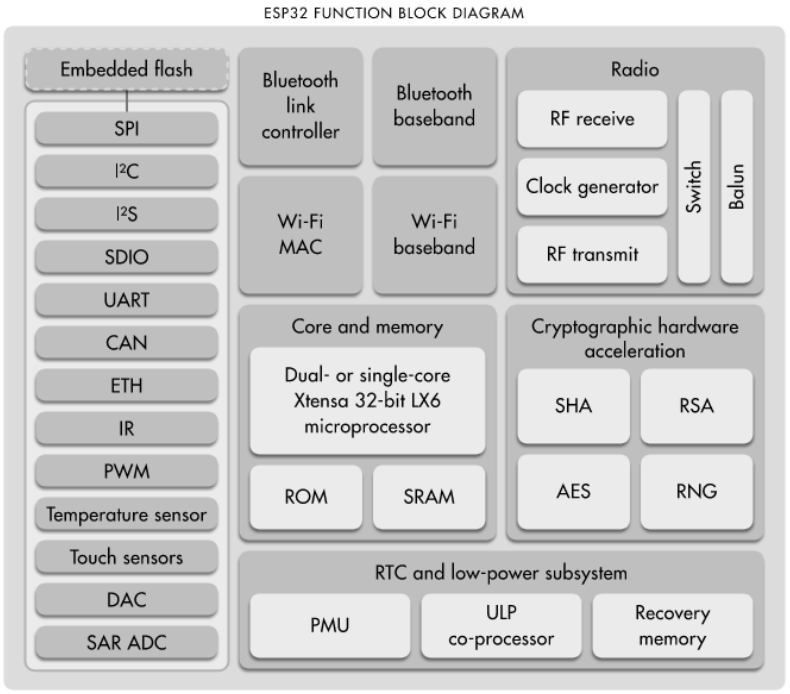


Ilustración 11 Diagrama de funciones del ESP32 (Atif, Muralidharan, Ko, & Yoo, 2020)

1. Embedded Flash: Es la memoria flash integrada en el ESP32 utilizada para almacenar el programa de firmware, datos y otros recursos necesarios para la operación del dispositivo.

2. Bluetooth Link Controller y Bluetooth Baseband: Estos bloques se encargan de la comunicación Bluetooth. El controlador de enlace (Link Controller) y el baseband (procesador de banda base) gestionan la conexión y el control de los enlaces Bluetooth, así como las tareas relacionadas con la transmisión y recepción de datos a través de la tecnología Bluetooth.
3. WiFi MAC y WiFi Baseband: Estos bloques son responsables de la funcionalidad Wi-Fi del ESP32. El MAC controla el acceso a medios y la transmisión de datos a través de la interfaz Wi-Fi, mientras que el Baseband se encarga del procesamiento de las señales Wi-Fi.
4. Core y Memory: Estos bloques representan el procesador principal y la memoria RAM del ESP32. El Core es el núcleo del microcontrolador, donde se ejecuta el programa y se realizan las operaciones del sistema. La memoria se utiliza para almacenar datos en tiempo de ejecución y variables.
5. RTC (Real-Time Clock) y Low Power Subsystem: El RTC es un reloj en tiempo real que permite realizar un seguimiento del tiempo incluso cuando el microcontrolador está en modo de bajo consumo o apagado. El subsistema de bajo consumo se encarga de administrar el consumo de energía del ESP32 y permitir modos de operación de bajo consumo.
6. Radio: El bloque de radio está relacionado con las funciones de comunicación inalámbrica, incluyendo tanto Bluetooth como Wi-Fi. Se encarga de la transmisión y recepción de datos utilizando las tecnologías de radio correspondientes.
7. Cryptographic Hardware Acceleration: Este bloque es responsable de acelerar el rendimiento de las operaciones criptográficas, como el cifrado y descifrado de datos, mediante hardware dedicado.

Tabla 2

Características del ESP32 frente el ATMEGA (Atmel, 2018) (System, 2021) (Bruce, 2021)

Característica	**ATmega (Arduino)**	**ESP32**
Arquitectura	AVR	Xtensa LX6
Velocidad de reloj	Hasta 16 MHz (algunos modelos hasta 20 MHz)	Hasta 240 MHz
Núcleos	Mono núcleo	Dual núcleo

Memoria flash	Hasta 256 KB (algunos modelos hasta 512 KB)	Hasta 4 MB
SRAM	Hasta 8 KB (algunos modelos hasta 16 KB)	Hasta 520 KB
EEPROM	Hasta 4 KB (algunos modelos hasta 32 KB)	No disponible
Conexión Wi-Fi	No	Sí, integrado
Bluetooth	No	Sí, integrado
GPIO	Variable (depende del modelo)	Hasta 34 pines
ADC	Hasta 10 bits	Hasta 12 bits
Interfaces de comunicación	UART, SPI, I2C	UART, SPI, I2C, I2S, CAN, Ethernet
Pines analógicos	Variable (depende del modelo)	Hasta 18 pines
Soporte de programación	Mediante el IDE de Arduino	Mediante el IDE de Arduino o ESP-IDF
Comunidad y soporte	Amplia comunidad y documentación disponible	Amplia comunidad y documentación disponible

El ESP32 es ampliamente utilizado y goza de popularidad en proyectos de IoT debido a las características y ventajas que ofrece, especialmente en términos de conectividad y potencia de procesamiento. Estas características sobresalen en comparación con el ATMEGA (Bruce, 2021), el cual es más común en prototipos de robótica.

Ahora en cuanto a la especificaciones del ESP32 WROVER-B ya que éste es el microcontrolador integrado en la TTGO-T Call SIM800L se tiene lo siguiente (Systems, Espressif , 2023) (Ver Anexo).

El ESP32-WROVER-B es un módulo que se destaca por su gran capacidad de almacenamiento de datos, ya que cuenta con una memoria RAM pseudoestática SPI (PSRAM) adicional de 8MB. Esta memoria PSRAM es especialmente útil para aplicaciones que requieren más espacio para datos, como por ejemplo, búferes para almacenamiento de imágenes, streaming de audio, entre otros. Este aspecto es muy importante debido el microcontrolador ejecutará un programa donde se le estará creando un buffer

temporal para el almacenamiento de datos en especial los datos de la gráfica concentración de CO_2 vs tiempo.

La RAM pseudoestática opera manteniendo los datos mientras está alimentada, funcionando de manera similar a un extenso archivo con múltiples compartimentos, donde cada uno puede almacenar y guardar información. Estos compartimientos pueden abrirse y cerrarse en cualquier momento para acceder y visualizar su contenido, permitiendo así la lectura y escritura de datos de manera ágil y dinámica.

En la Tabla 3 se observa la comparativa completa de las características del ESP32-WROVER-B frente a ESP32-WROOM y el ATMEGA328P.

Tabla 3

ESP32 Wrover-B vs Wroom vs ATMega328P (Systems, Espressif , 2023) (Components101, 2021)

Características	**ESP32-WROVER-B**	**ESP32-WROOM**	**ATmega328P**
CPU	Tensilica Xtensa LX6 (doble núcleo, 80-240 MHz)	Tensilica Xtensa LX6 (doble núcleo, 80-240 MHz)	AVR de 8 bits (hasta 20 MHz)
Memoria Flash	4MB SPI Flash externa	4MB SPI Flash externa	32kB Flash a bordo
Memoria RAM	520 KiB SRAM + 8MB PSRAM adicional	520 KiB SRAM	2kB SRAM
EEPROM	No especificado	No especificado	1kB
GPIOs	23 líneas de E/S de propósito general	23 líneas de E/S de propósito general	Hasta 23
Conectividad	Wi-Fi de 2.4GHz y Bluetooth de doble modo (Clásico y Baja Energía)	Wi-Fi de 2.4GHz y Bluetooth de doble modo (Clásico y Baja Energía)	No tiene

Como se observa entre los microcontroladores ESP32 no hay mucha diferencia en cuanto en tema de conectividad pero si en cuanto a memoria RAM y pues el ATMega328P no compite con ellos en ninguno de estos aspectos. Donde si la comparativa es similar es el precio[3], dependiendo del proveedor, aun así se justifica el cambio que se hará en este nuevo diseño debido a las necesidades planteadas.

2.1.3 GSM SIM800L.

La mejora más significativa del equipo radica en la funcionalidad de conectividad GPRS. Aunque el microcontrolador ESP32 ya tenga integrada la conectividad WiFi, es evidente que en el campo de medición, es totalmente improbable contar con estas señales disponibles. Por lo tanto, se requiere un módulo capaz de establecer conexión a Internet mediante un chip SIM de teléfono móvil.

Ilustración 12 Chip SIM800L.

Durante el desarrollo de este proyecto, se llevaron a cabo pruebas con distintos módulos GSM, incluyendo variantes como el SIM900 o el SIM868. Tras un exhaustivo análisis, se determinó que el módulo SIM800L proporcionaba los resultados más satisfactorios. A pesar de la diversidad de modelos disponibles en el mercado, la elección del SIM800L se fundamenta en la integración directa en la placa de desarrollo, lo cual simplifica su implementación con tan solo requerir la programación específica para su correcto funcionamiento.

El módulo SIM800L, fabricado por SIMCom, no se limita solo a establecer conexiones a Internet a través de GPRS, TCP o IP. Además, ofrece una amplia gama de funcionalidades adicionales: permite

[3] Precio solamente del microcontrolador sin incluir la placa de desarrollo.

realizar y recibir llamadas, enviar y recibir mensajes de texto, e incluso cuenta con la capacidad de sintonizar señales de radio FM.

Tabla 4

Especificaciones del SIM800L (SIMCom, Components101 , 2023)

Características	SIM800L
Banda	Cuádruple banda 850/900/1800/1900MHz
GPRS	Soporta GPRS clase 10 y GSM 07.07, 07.05 y 03.40
Interfaz serial	UART
Voltaje de operación	3.7V a 4.4V
Protocolos de internet	Soporta los protocolos HTTP, FTP, SMTP y POP3
Tamaño	15.8 x 17.8 x 2.4 mm
Peso	1.35g

El aspecto de interés entre las especificaciones es el protocolo HTTP, el cual permitirá enviar los datos de las mediciones a la hoja de cálculo de Google Spreadsheet. A nivel de código el SIM800L se controla mediante los comandos AT (SIMCom, SIM800 Series AT Command Manual, 2014). Una característica de la placa para indicar si la SIM800L tiene conexión GPRS es el parpadeo rápido del LED.

2.1.4 RTC DS1307.

Los RTC (Real Time Clock) o relojes en tiempo real son una solución ideal para integrar mediciones temporales precisas en proyectos. Estos dispositivos se destacan por su bajo consumo de energía, lo que les permite ser alimentados por baterías y mantener la sincronización temporal incluso en ausencia de energía externa. A diferencia de los contadores internos presentes en los microcontroladores, los RTC ofrecen una precisión superior en la medición del tiempo y la fecha.

La integración de este módulo resulta esencial, ya que en ciertos escenarios, puede ser impracticable enviar información de manera inmediata a la base de datos. En estos casos, almacenar los datos para su posterior transmisión desde el microcontrolador es necesario. El RTC se encarga de proporcionar la fecha

y hora exactas en el momento en que se realiza la medición, lo que resulta crucial para registrar y sincronizar los eventos de forma precisa.

El RTC se encarga de registrar el momento en que se llevan a cabo las mediciones. Este registro temporal es fundamental para cada dato de concentración de CO_2 que se refleja en las gráficas de la Ilustración 2 e Ilustración 7. El objetivo principal radica en la creación de un búfer temporal que almacene este conjunto de datos, por lo que resulta crucial contar con la marca de tiempo correspondiente a cada medición para su correcta catalogación.

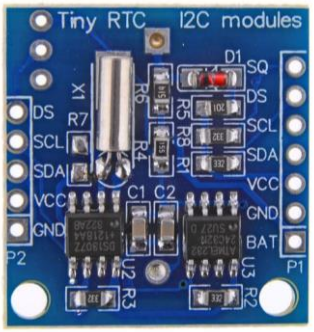


Ilustración 13 Módulo RTC DS1307 (Digest, 2018)

Para establecer la conexión entre el módulo RTC DS1307 y la placa TTGO T-Call SIM800L, se requiere la conexión de los pines de alimentación (VCC y GND) del DS1307 con los pines correspondientes en la placa de desarrollo. Luego, es necesario conectar el pin SDA (datos) del DS1307 al pin SDA de la placa TTGO T-Call, y el pin SCL (reloj) del DS1307 al pin SCL de la placa TTGO T-Call. Estos pines, SDA y SCL, se utilizan para la comunicación I2C entre el DS1307 y la placa TTGO T-Call. Por último, se debe considerar que el DS1307 requiere una batería de respaldo para mantener la hora cuando no está siendo alimentado, normalmente utiliza una batería CR2032. Luego se mostrará un esquemático nuevo para el CPU y una tabla para las nuevas conexiones.

2.1.5 Módulo convertidor DC-DC MH-KC24

Ilustración 14 Módulo convertidor MH-KC24 (MH-KC24, 2023)

El módulo MH-KC24 al que la mayoría de los proveedores suelen referirse como el módulo de convertidor QC3.0 Buck. Se utiliza principalmente para fabricar cargadores QC3.0, portátiles, pero en este caso alimentará la placa de desarrollo TTGO T-Call SIM800L ya que tiene un rango de voltaje de entrada de 6V a 32V, (que es donde se conectaría la batería de 12 V) y puede proporcionar una salida de hasta 24W (5V/3.4A, 9V/2.5A, 12V/2A, etc.), normalmente la salida estándar se ajusta para 5V la cual es la tensión que se necesita para la operación.

Tiene protección contra sobrevoltaje y subvoltaje de entrada, protección contra sobrecorriente de entrada y salida, protección contra cortocircuito y protección contra sobrecalentamiento, una función de compensación de voltaje del cable de salida y una eficiencia de conversión del 90%-97%. El dispositivo utiliza el microcircuito de conversión IP6505 (Anexo) desarrollado por Yingjixin Technology.

La introducción de este nuevo convertidor suplantará al LM2596. Un inconveniente en el suministro de energía a la placa de desarrollo mediante el LM2596 se relaciona con la utilización de un conector JST, lo que requiere la pulsación del botón de reinicio de la placa, tal como se detalla en la documentación (Xinyuan-LilyGO, 2023). Esto resulta poco práctico, ya que, en la configuración original con Arduino, el equipo se energizaba simplemente con el botón de encendido de anclaje (Arriaza Carrillo & Aguilar Vega, 2022). Por lo tanto, el MH-KC24 abastece la placa mediante un cable USB-C a 5V, siguiendo una metodología similar a la carga de un programa en una computadora o una batería PowerBank.

2.1.6 Sensores y módulos originales.

En el capítulo 1 se describió detalladamente sobre los componentes originales del CPU, entre ellos se conservan el módulo convertidor RS232 a TTL, el módulo de geolocalización Neo 6M y el sensor de temperatura y humedad DHT22, ya que son los encargados hacer las lecturas correspondientes de la estación móvil, por otro lado el Relé de 5V 1 canal también se mantendrá, ya que es el responsable de la activación de la bomba.

Resumiendo, el rediseño del CPU a nivel de hardware no consta de demasiados cambios al agregar o cambiar componentes puesto que es necesario tener en cuenta el consumo de energía de éstos.

2.1.7 Conexionado del CPU.

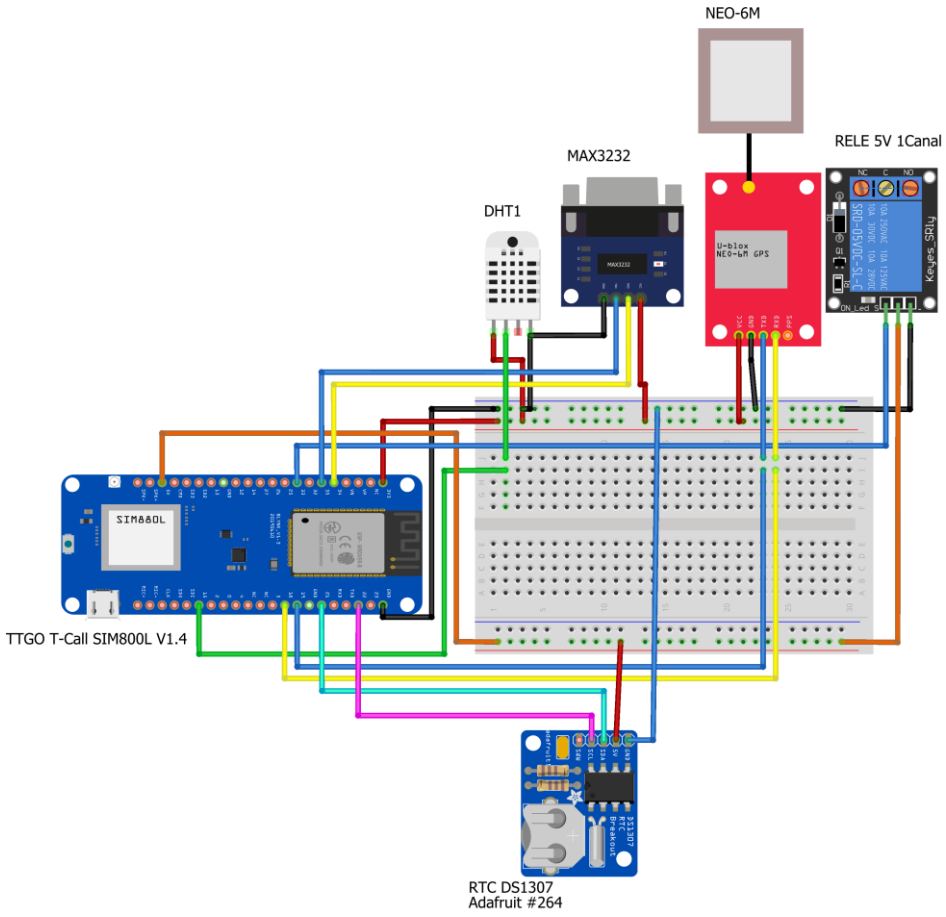


Ilustración 15 Diagrama en fritzing del CPU

En la Ilustración 15 se presenta un esquema realizado en Fritzing para el diseño del CPU, similar al mostrado en la Ilustración 5. Se aprecia la inclusión de los componentes previamente mencionados, mientras que se ha excluido el módulo HC-06 y el MH-KC24 (dado que este último suministra energía al TTGO T-Call SIM800L a través de USB-C). Siguiendo la metodología descrita en (Arriaza Carrillo & Aguilar Vega, 2022), se presentarán tablas detalladas para facilitar la comprensión ordenada de las conexiones tanto desde la placa de desarrollo como de los demás módulos.

Tabla 5

Conexiones de la TTGO T-Call SIM800L a los demás componentes

PIN GPIO TTGO T-Call	**Conexión a:**	**PIN módulo**
33	Relé	IN
35	Convertidor RS232	TXD
34	Convertidor RS232	RXD
18	GPS NEO6M	RXD
19	GPS NEO6M	TXD
15	Sensor Temp/Humedad	DATA
GND	Todos los módulos	GND
3V3	GPS NEO 6 M	VCC
3V3	Convertidor RS232	VCC
3V3	Sensor Temp/Humedad	VCC
5V	Relé	VCC
5V	RTC DS1307	VCC
22	RTC DS1307	SCL
21	RTC DS1307	SDA

Tabla 6

Conexiones del GPS Neo 6M

PIN	**Conexión a:**	**PIN**
VCC	TTGO T-Call	3V3
GND	TTGO T-Call	GND
TXD	TTGO T-Call	19
RXD	TTGO T-Call	18

Tabla 7

Conexiones del Convertidor RS232-TTL

PIN	**Conexión a:**	**PIN**
VCC	TTGO T-Call	3V3
GND	TTGO T-Call	GND
TXD	TTGO T-Call	35
RXD	TTGO T-Call	34

Tabla 8

Conexiones del RTC DS1307

PIN	**Conexión a:**	**PIN**
VCC	TTGO T-Call	3V3
GND	TTGO T-Call	GND
SDA	TTGO T-Call	21
SCL	TTGO T-Call	22

Tabla 9

Conexiones del Relé 5V

PIN	Conexión a:	PIN
VCC	TTGO T-Call	5V
GND	TTGO T-Call	GND
IN	TTGO T-Call	33

Tabla 10

Conexiones del DHT22

PIN	Conexión a:	PIN
VCC	TTGO T-Call	5V
GND	TTGO T-Call	GND
DATA	TTGO T-Call	15

Tabla 11

Conexiones del MH-KC24

PIN	Conexión a:	PIN
IN +/-	Batería 12V	+/-
USB	TTGO T-Call	USB-C

Con respecto al dimensionamiento de la batería se hará una revisión sobre el consumo de energía del nuevo CPU.

2.1.8 Circuito impreso.

El modelo inicial del circuito fue desarrollado utilizando una plataforma de prototipado conocida como placa perforada o placa de pruebas. Esta elección se basó en la necesidad de una solución ágil y versátil para la fase inicial de diseño. Se asume que la elección de la placa perforada para el modelo inicial se basó en su capacidad para facilitar el prototipado rápido. Los componentes pueden ser fácilmente insertados, movidos o cambiados, sin embargo, se identificaron oportunidades de mejora las cuales se detallan a continuación:

- Conexiones inestables.

  Tomando en consideración que las conexiones en la placa perforada se realizaron por medio de cables o jumpers, existe mayor probabilidad de un falso contacto si uno de los cables sufre un movimiento brusco.
- Limitaciones de frecuencia y velocidad.

  Debido a la naturaleza no uniforme de las conexiones y a la falta de apantallamiento, las placas perforadas pueden presentar limitaciones en términos de frecuencia y velocidad de señal. En aplicaciones de alta frecuencia, las capacidades de la placa pueden no ser suficientes.
- Dificultad para el diseño de circuitos complejos

  En el diseño inicial la placa perforada únicamente servía para la interconexión de componentes, todos los módulos estaban pegados al contenedor del CPU.
- Riesgo de corto circuitos

  La proximidad entre los orificios de la placa perforada aumento el riesgo de cortocircuito especialmente cuando se utilizan cables y componentes con pines largos.
- Durabilidad y resistencia

  La manipulación constante del circuito genera desgaste en las soldaduras de los circuitos afectando así la durabilidad y conexión de los componentes.
- Estética y profesionalismo.

  En comparación con los circuitos impresos, los circuitos en placas perforadas pueden parecer menos ordenados y profesionales.

El modelo inicial del circuito se muestra a continuación:

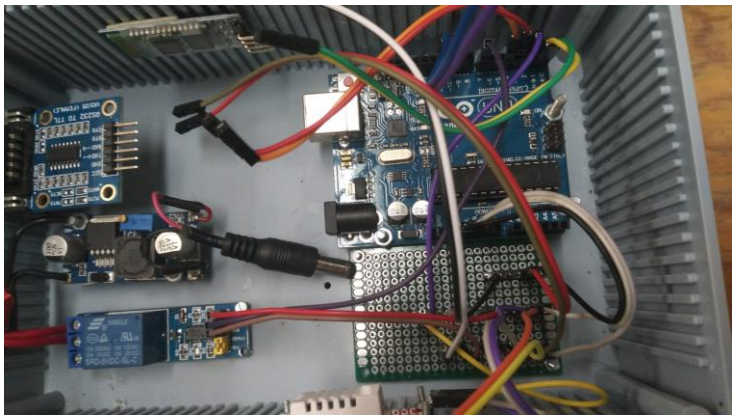


Ilustración 16 Circuito Inicial

Tomando en consideración lo descrito anteriormente se llevó a cabo el montaje de un circuito en una PCB (Placa de Circuito Impreso) lo cual es una elección común y estratégica en el desarrollo de dispositivos electrónicos por varios motivos. Aquí se describen algunos de los motivos clave que se consideraron para su desarrollo:

- Fiabilidad y Durabilidad:
  Las PCBs están diseñadas para proporcionar conexiones eléctricas confiables y duraderas. Las pistas conductoras y las conexiones soldadas garantizan una baja resistencia eléctrica y una menor probabilidad de falla a largo plazo.
- Reducción de Errores:
  El diseño de la PCB se basa en software de diseño asistido por ordenador (KiCad), lo que reduce significativamente la posibilidad de errores en comparación con el montaje en placas perforadas u otras formas de prototipado manual.
- Mejora del Rendimiento Eléctrico:
  Las PCBs están diseñadas para minimizar la inductancia y la capacitancia parasitarias, lo que mejora el rendimiento eléctrico del circuito
- Facilita el Diseño de Circuito Complejo:
  La PCB permite la implementación de circuitos complejos con múltiples capas y componentes. Esto es fundamental para dispositivos electrónicos avanzados y sistemas integrados.
- Aspecto Profesional:
  El montaje en PCB proporciona un aspecto profesional y ordenado. Lo cual es ideal para presentar un prototipo terminado.

El software utilizado para el desarrollo de la PCB fue KiCad, esto se debe a que es un programa con licencia de código abierto y esto permite su uso gratuito, es multiplataforma y cuenta con una amplia biblioteca de componentes los cual es esencial al momento del desarrollo del circuito ya que las medidas de cada uno los modelos deben ser exactos.

Ilustración 17 Logo de KiCad

La descarga del software se realiza directamente desde su sitio oficial, únicamente se debe escoger el sistema operativo en el cual se utilizará y proceder a instalar.

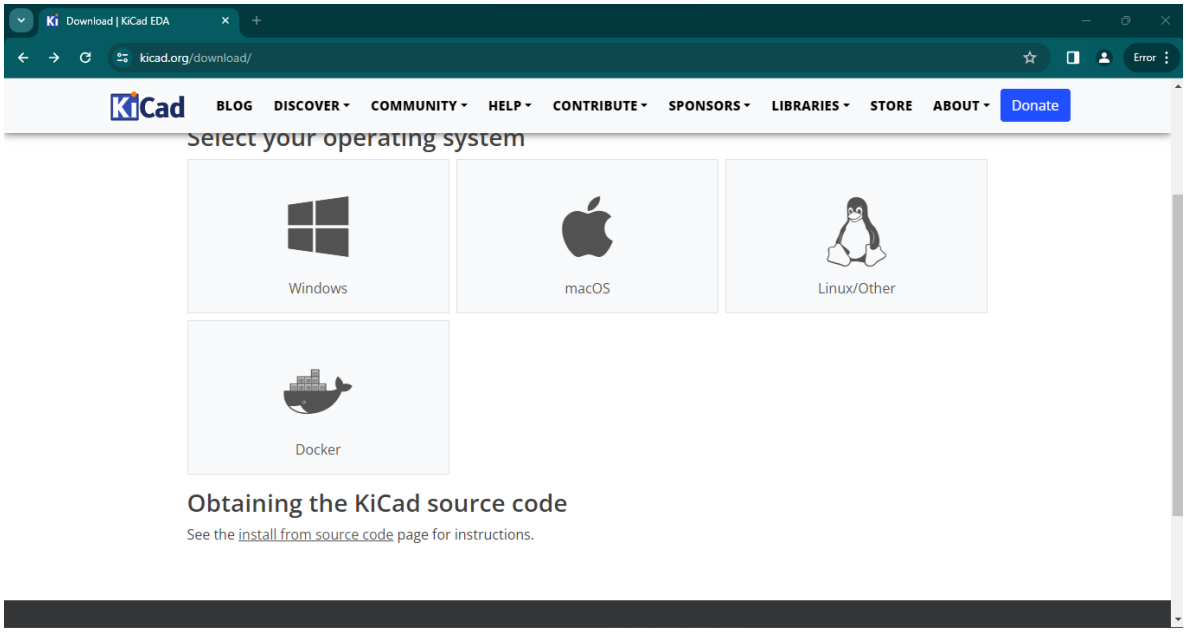


Ilustración 18Interfaz de KiCad

Para mayor detalle al respecto del uso del programa, en el sitio oficial podemos encontrar un manual de usuario detallado sobre como iniciar un nuevo proyecto esquemático y posteriormente llevarlo a una PCB

Dentro de las ventajas de utilizar este programa se menciona el amplio catálogo de bibliotecas con los diseños de diferentes modelos a utilizar en la construcción de la placa impresa, siendo el más importante el microcontrolador TTGO T-Call SIM800L por la cantidad de pines a utilizar y el espacio que utilizaría dentro del circuito, el modelo del microcontrolador se encontraba disponible en el amplio catálogo de "Footprints" como son llamadas los modelos dentro del software, entorno a él se construyen los demás componentes especificando la cantidad de pines a utilizar.

El esquemático realizado para el montaje de la placa es el siguiente:

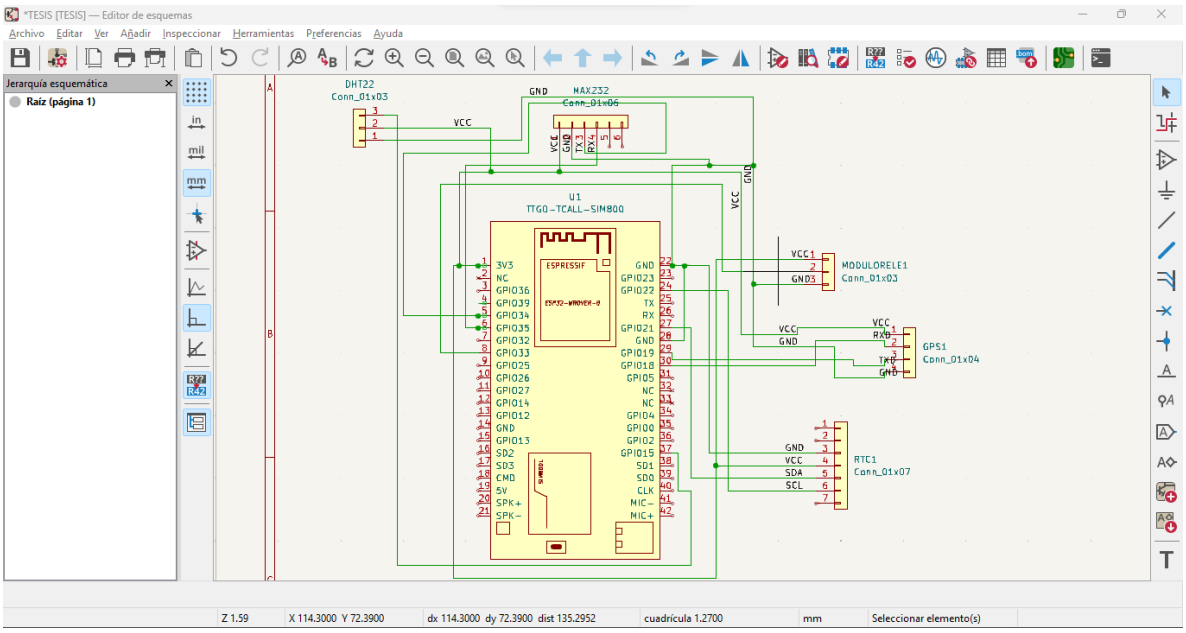


Ilustración 19 Esquemático de los componentes

Una vez generado el diagrama esquemático se procede con la conexión de los componentes en el diseño de la placa impresa, el software no lo realiza de forma automática, únicamente presenta los pines que deben ir conectados y es el usuario quien debe de posicionar de forma adecuada los componentes para aprovechar al máximo los espacios y garantizar la interconexión de todos dentro del circuito.

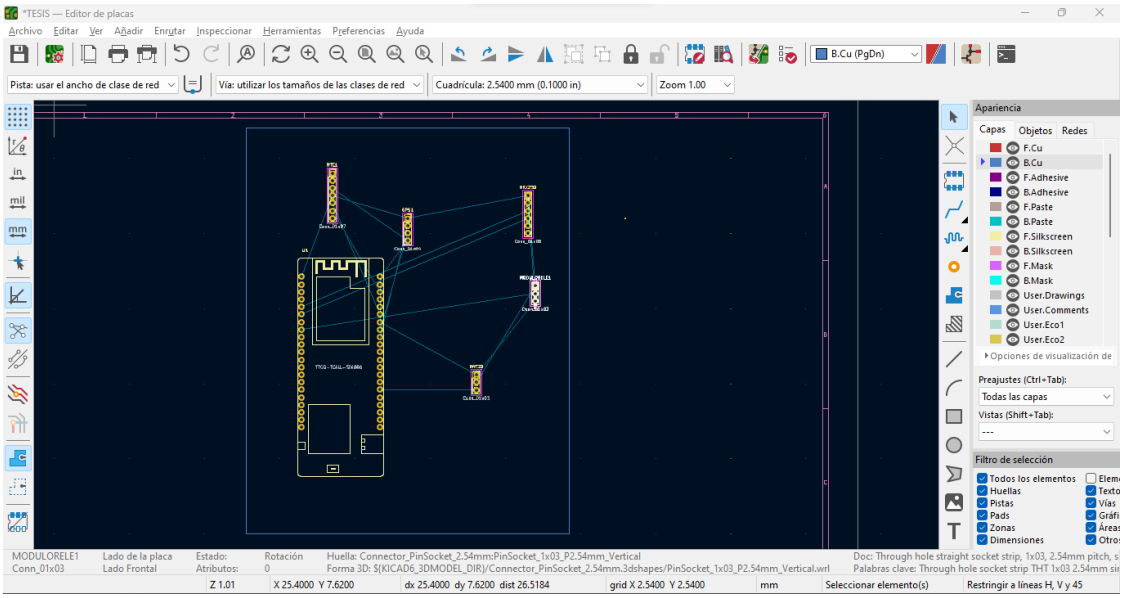


Ilustración 20 Interconexión de los componentes

Una vez los componentes son posicionados y conectados correctamente, para facilitar la fabricación de la PCB, se añaden espacios cubiertos entre el circuito, de esta manera el químico utilizado para el revelado de la placa (Cloruro férrico) debe de consumir menos cobre acelerando así el proceso.

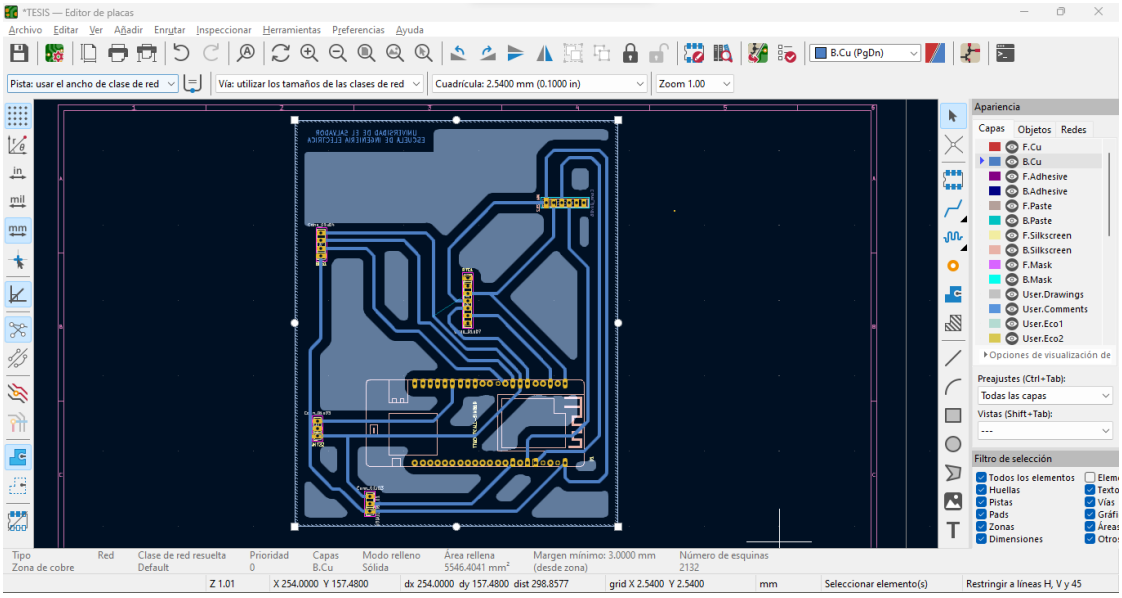


Ilustración 21 Diseño final del circuito impreso

Con la finalidad de mantener la alta calidad del circuito se utilizó una cortadora laser marca TROTECLASER modelo Speedy 360. Para la construcción se hizo uso de una placa combinada de fibra de vidrio a un lado y cobre por el otro, el lado de cobre fue cubierto con pintura negra para que la maquina cortadora retire la pintura de los lugares donde no se necesita el cobre.

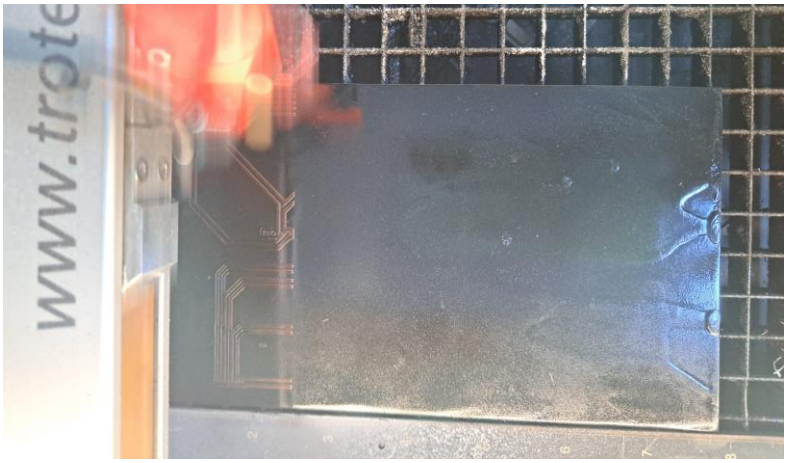


Ilustración 22 Descubrimiento de pistas del circuito

Una vez retirada la pintura de los lugares donde no se necesita cobre con ayuda del grabado laser, la placa es sumergida en cloruro férrico para proceder con el revelado de las pistas del circuito.

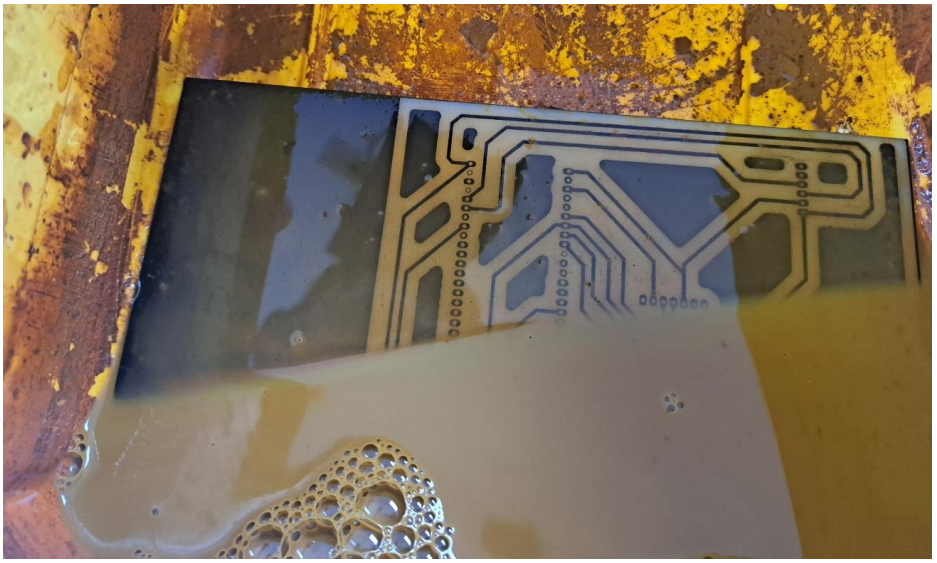


Ilustración 23 Placa sumergida en Cloruro Férrico

Una vez se retira todo el cobre que no se utilizará, se procedió a cortar la parte de la placa que no se utilizará y con ayuda de un taladro eléctrico adaptado a una prensa se perfora con precisión cada uno de los agujeros de los componentes en la placa.

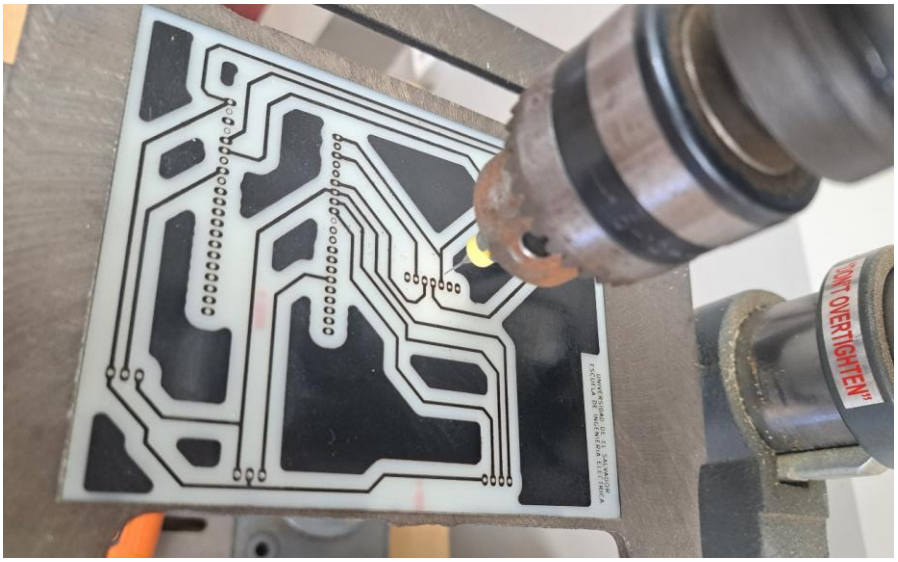


Ilustración 24 Corte en la ubicación de los pines.

Finalmente se procede soldar cada uno de los componentes en la placa impresa, teniendo como resultado final un circuito impreso de alta calidad con un diseño profesional que garantiza durabilidad y conexiones más estables entre todos los módulos utilizados para el funcionamiento del prototipo.

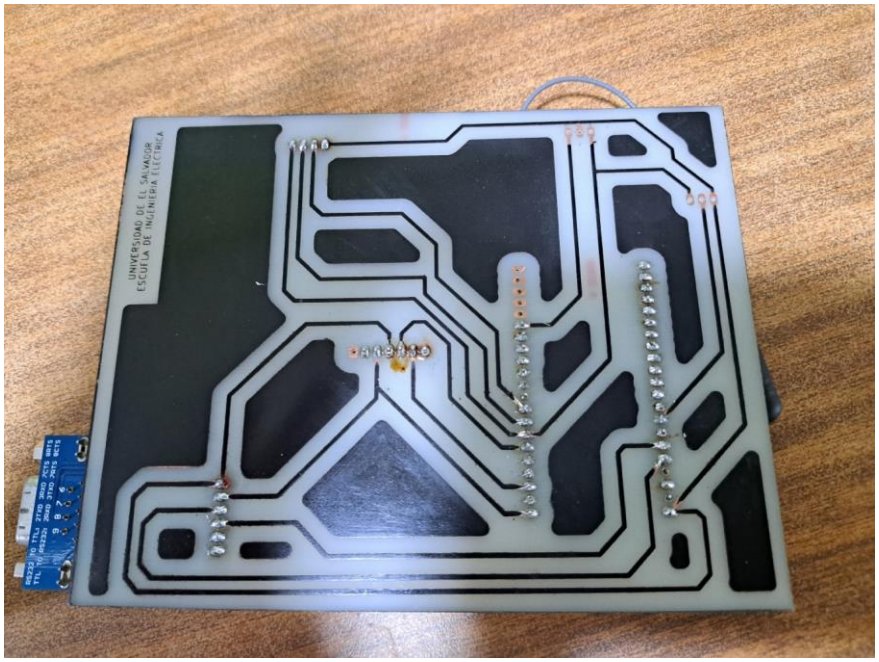


Ilustración 25 Diseño finalizado de circuito impreso

2.2 Diseño del Software.

En la lógica de programación para coordinar múltiples tareas, se empleó el lenguaje C++ junto con el entorno de desarrollo Arduino IDE, reconocido por su practicidad y compatibilidad con el ESP32. No obstante, en algunas situaciones específicas, se requirió el uso del entorno VS Code para acceder a más opciones de depuración. Para la programación del ESP32, se procedió a instalar el soporte correspondiente para este dispositivo en el entorno de desarrollo de Arduino IDE, lo que permitió la adición de la placa correspondiente y la carga del código. La placa utilizada para este propósito fue el ESP32 Dev Module (Tutorials, Random Nerd Tutorials, 2016).

El programa original descrito en el capítulo 1 constaba de 2 tareas, la de la comunicación Bluetooth y la lectura de datos de sensores y módulos. Ahora se le añadirá la tarea extra, la de almacenamiento temporal de datos para su posterior envío de la Google Spreadsheet y se añadirá la función enviar por

Bluetooth los procesos del CPU a la aplicación móvil. En esta sección se explicará detalladamente los fragmentos de código que realizan estas tareas.

2.2.1 Lectura de datos.

En esta parte el código se leen datos de los sensores de temperatura/humedad y CO_2 del DHT2 y LI-COR 830 respectivamente, además se obtiene la geolocalización a través del módulo Neo 6M, y la hora correspondiente que por practicidad se utilizó el módulo RTC.

Se utilizan librerías DHT.h, Wire.h, RTClib.h para la gestión del sensor DHT22 y el módulo RTC. Por otro lado, para la geolocalización se modificó el uso de la librería TinyGPS.h a TinyGPS++.h, ya que se trata de una versión actualizada y mejorada siendo más compacta y resistente con más funcionalidad como la capacidad de manejar múltiples tipos de sentencias NMEA y proporcionar más información sobre la ubicación, el tiempo, altitud, etc.

Tanto el módulo Neo 6M y el módulo convertidor MAX232 utilizan comunicación serial por lo tanto es necesario definir los pines necesarios para este propósito. Esto se hace mediante la librería SoftwareSerial.h que replica la funcionalidad de pines seriales.

2.2.1.1 Lectura de temperatura y humedad.

Anteriormente la lectura de temperatura y humedad se bloqueaba mediante un comando enviado desde la aplicación móvil, ahora se mantendrá siempre activa cada 10 segundos ya que no interfiere con las lecturas de concentración de CO_2.

```
void Temp_Hmd() {
  humedad = Obj_DHT.readHumidity();
  temperatura = Obj_DHT.readTemperature();
  if (isnan(humedad) || isnan(temperatura)) {
    Serial.println("Existe un error en la lectura del sensor DHT22!");
    humedad = 0; temperatura = 0;
  }
}
```

Fragmento 1 Función para leer la temperatura y humedad

La función es la misma que se presenta en el código original (Arriaza Carrillo & Aguilar Vega, 2022).

2.2.1.2 Lectura de geolocalización.

```
void Geolocalizacion() {
  bool datosNuevos = false;
  unsigned long tiempoInicio = millis();
  const unsigned long tiempoLimite = 1000;

  while (millis() - tiempoInicio < tiempoLimite) {
    while (ss.available()) {
      char c = ss.read();
      if (gps.encode(c)) {
        datosNuevos = true;
        break;
      }
    }
  }
  if (datosNuevos && gps.location.isValid()) {
    latitud = gps.location.lat();
    longitud = gps.location.lng();
    satelites = gps.satellites.value();
    precision = gps.hdop.value();
    gpsFlag= true;
  }else{
    gpsFlag= false;
  }
}
```

Fragmento 2 Función para obtener la geolocalización.

Dado a que dentro del código se evita la el uso de delays[4], que detiene la ejecución del programa se declaran algunas variables como datosNuevos, tiempoInicio, tiempoLimite para garantizar que hay datos nuevos del módulo GPS y el tiempo de adquisición de estos será cada segundo, ejecutándose en un bucle cuando se den estas condiciones.

Dentro del bucle, se utiliza otro bucle while para leer datos del objeto ss, que es una instancia de SoftwareSerial. Esto se hace para recibir datos del módulo GPS a través de una comunicación serial.

- Se lee un carácter (char c) de la comunicación serial (ss) y se verifica si el objeto gps (que es un objeto de la biblioteca TinyGPS++) puede interpretar y decodificar este carácter como datos de GPS utilizando gps.encode(c) .
- Si se puede decodificar con éxito un carácter como datos GPS, se establece la bandera datosNuevos en true y se rompe el bucle interno con break.

[4] Es importante evitar el uso de delay() en el código que utiliza el módulo GPS NEO-6M porque puede interferir con la recepción de datos del GPS.

- Esto permite capturar datos del módulo GPS durante el período especificado (tiempoLimite) o hasta que se reciban datos válidos del GPS.

Después de capturar los datos del módulo GPS (o si no se capturan datos nuevos), se procede a leer la temperatura y humedad almacenándose en variables globales tipo float.

La función concluye almacenando en variables locales los datos de geolocalización solo si se han recibido nuevos datos del GPS.

2.2.1.3 Lectura de concentración de CO_2.

Para la lectura de concentración de CO_2, se mantiene el nombre de la función sin embargo ésta si sufre algunos cambios.

```
void Licor() {
  if (SerialMax232.available()) {
    do{
      licorData = SerialMax232.readStringUntil('\n');
      co2Value = licorData.toFloat();
    }while(co2Value>0 && co2Value<100);
  }
    now = rtc.now();
    respuesta = String(co2Value)+","+"0"+","
                                    +"LATITUD =" + String(latitud, 6)
                                    +",LONGITUD =" + String(longitud, 6)
                                    + ",SATELITES =" + String(satelites)
                                    + ",PRECISION =" + String(precision)
                                    + "," + String(temperatura) + "," +
String(humedad) + "#\n";
    if (bombaActivada && envioInicio) {
      almacenarDatos(now);
    }
  imprimirRespuesta(respuesta);
}
```

Fragmento 3 Función para obtener la concentración de CO_2.

Se condiciona esta función si hay comunicación serial con el módulo MAX232 que es el encargado de recibir los datos del LI-COR 830, dado a que la frecuencia en que se reciben los datos es de 1 segundo, es necesario validarlos antes de proceder a guardarlos en las variables globales evitando valores por debajo de los 200 ppm que son catalogados como errores en la comunicación serial.

Esta función toma el tiempo actual guardando la hora y además genera el String principal conformado por las variables globales que posteriormente se enviaran por Bluetooth invocando la función imprimirRespuesta() . Esto se explicará en los siguientes incisos de esta sección. El String respuesta se

genera en esta función porque el valor de concentración de CO_2 es el que se actualiza con mayor frecuencia (1 segundo).

2.2.2 Comunicación Bluetooth.

El código original se basaba en un módulo HC-06 como esclavo y el ATMega328P actuando como maestro, como se describe en (Arriaza Carrillo & Aguilar Vega, 2022). Se destacaba el uso de los pines seriales para la transmisión de datos por Bluetooth. Sin embargo, al realizar modificaciones en el programa resultaba un problema puesto que cualquier línea de depuración impresa en el monitor serial se enviaba e interrumpía el flujo de los datos, entonces se descartó dicho módulo, aprovechando una de las ventajas principales del ESP32, la capacidad de comunicación vía Bluetooth. Para lograr esto, se empleó la librería BluetoothSerial.h, la cual facilita el control de acciones del relé y la recepción en tiempo real de lecturas provenientes de los sensores y módulos en la aplicación.

La comunicación por Bluetooth se divide en dos funciones principales: la transmisión de datos e información de los procesos hacia la aplicación, y la recepción de comandos por parte de la aplicación móvil en forma de caracteres para controlar el encendido o apagado de la bomba.

2.2.2.1 Envío de datos e información de procesos por Bluetooth.

Las lecturas de los sensores y módulos se envían en un String separado por comas (“,”) declarado en la función Licor() llamado respuesta, el cual se utiliza en la función llamada imprimirRespuesta que lo recibe como parámetro:

```
void imprimirRespuesta(String datos) {
  Serial.print(datos);
  SerialBT.print(datos);
}
```

Fragmento 4 Función para enviar datos por Bluetooth

De esta manera lo que recibe la aplicación por ejemplo es lo siguiente:

```
4.3e2,0,LAT =13.846584,LON =-89.2627426 ,SAT =6,PRE =126,28.00,71.50#
```

Fragmento 5 Mensaje enviado por Bluetooth

Luego la aplicación se encarga de descomponer cada lectura y mostrarlas en tiempo real.[5] Se observa que al final siempre se imprime un “#”, esto sirve como una especie de control de paro en los mensajes recibidos y enviados por parte de la aplicación y por parte del microcontrolador.

Ahora el String enviado es diferente al mostrado en el capítulo 1, ahora todas las lecturas son enviadas al mismo tiempo y se actualizan de acuerdo al programa.

Con respecto al envío de los procesos del CPU a la aplicación, funciona de la misma manera.

```
void procesos() {
  String mensaje =
String(conexionGPRS)+","+String(gpsFlag)+","+Almacenamiento+ ","+ envioEstado
+"%\n";
  SerialBT.print(mensaje);
  Serial.print(mensaje);
}
```

Fragmento 6 Función para enviar información de los procesos del CPU a la aplicación.

El String que se envía esta vez es llamado mensaje y se conforma de las banderas conexionGPRS, que indica si hay conectividad por parte del módulo SIM800L para poder hacer envíos de datos, la bandera gpsFlag que indica si el módulo Neo 6M ha encontrado datos nuevos de geolocalización. El String de Almacenamiento que indica la cantidad de datos que se han guardado temporalmente para ser enviados, y el String envioEstado que proporciona información acerca de la preparación del paquete de datos almacenados al momento de ser enviados y cuando estos han sido recibidos por la hoja de cálculo. Además se observa ahora el control de paro lectura que se le da a la aplicación para que procese mensaje es “%”.

Esta información se verá en un nuevo cuadro diseñado en la aplicación simulando una pantalla del CPU.

2.2.2.2 Activación de la Bomba.

El control de las acciones del relé se lleva a cabo a través de caracteres enviados por la aplicación hacia el microcontrolador. En el Fragmento 6 se presentan las funciones encargadas de recibir los mensajes y verificar los caracteres recibidos.

[5] Las lecturas de geolocalización dependiendo de la ubicación pueden tardar ya que el módulo necesita encontrar al menos 4 satélites para recibir los parámetros.

```
void recibirDatosBT() {
  if (SerialBT.available()) {
    String mensajeBT = SerialBT.readStringUntil('\n');
    procesarComando(mensajeBT);
  }
}
void procesarComando(String mensaje) {
  char c = mensaje.charAt(0);
  String numeroStr = mensaje.substring(2);
  if (numeroStr.length() > 0) {
    numeroRecibido = numeroStr.toInt();
    Serial.print("Número recibido: ");
    Serial.println(numeroRecibido);
  }
  switch (c) {
    case 'F':
      bombaActivada = true;
      break;
    case 'G':
      activarBomba();
      break;
    case 'H':
      desactivarBomba();
      break;
    case 'E':
      vaciarEstructuraFlag = true;
      break;
    case 'M':
      reinciarModem();
      break;
    case 'R':
      mostrarYVaciarDatosAlmacenados();
      break;
    default:
      break;
  }
}
```

Fragmento 7 Funciones que procesan los comandos recibidos desde la aplicación móvil.

La función recibirDatosBT() verifica si hay datos disponibles en el puerto serie Bluetooth y, si es así, lee los datos hasta que se encuentra un carácter de nueva línea. Luego, llama a la función procesarComando() con el mensaje recibido como argumento. Esta función toma el primer carácter del mensaje recibido y lo almacena en la variable c y dependiendo del caracter se toman diferentes acciones,

como el caso de “F”[6] que detiene la obtención de geolocalización (esto es porque se está a punto de hacer una medición y no se quiere que se actualicen las coordenadas y existan diferentes para un mismo punto de medición), el comando “G” activa la bomba y el comando “H” la desactiva. El comando “E” pone en verdadero vaciarEstructuraFlag, que al hacerlo en el siguiente iteración del loop principal del programa entrará a una condicional para enviar los datos almacenados y vaciarlos. El comando “M” reiniciará el SIM800L si al caso se pierde la conexión GPRS en la zona. Por último el comando “R” llama a la función mostrarYVaciarDatosAlmacenados(), que solamente muestra los datos en el monitor serial y los vacía posteriormente.

Hay que tener presente algo importante en el comando “F” y es que desde la aplicación viene acompañada de un número de esta forma: “F,10” por ejemplo. El número 10 hace referencia al número del punto de medición y ese también se guarda para su posterior envío.

A comparación del programa original el control algunas de estas acciones se hacían en el mismo loop principal, entonces por organización se decidió apartarlo con una función e invocarla posteriormente debido a que actualmente el programa realiza más actividades.

```
void activarBomba() {
  delay(300);
  bombaActivada = true;
  envioInicio= true;
  digitalWrite(PIN_RELE, LOW);
  delay(2000);
}

void desactivarBomba() {
  delay(300);
  bombaActivada = false;
  envioInicio= false;
  digitalWrite(PIN_RELE, HIGH);
}
```

Fragmento 8 Funciones para encender y apagar la bomba.

```
void reinciarModem() {
    if (conexionGPRS) {
    } else {
        Serial.println("Desconectado");
        int contadorIntentos = 0;
```

[6] Anteriormente los comandos enviados desde la aplicación venían acompañados de un “#” en este nuevo código no se ve la necesidad de utilizar ese carácter.

```
        while (!conexionGPRS && contadorIntentos < 5) {
            contadorIntentos++;
            digitalWrite(SIM800L_POWER, HIGH);
            delay(1000);
            SerialAT.println("AT+CFUN=1,1");
            delay(3000);
            configurarSim800L();
        }
        if (conexionGPRS) {
            Serial.println("Reconexión exitosa"); } else {
            Serial.println("No se pudo reconectar después de 5 intentos");
        }
    }
}
```

Fragmento 9 Función para reiniciar el SIM800L

La función reinciarModem() es una función que intenta restablecer la conexión GPRS en caso de que se haya perdido (luego se verá más detallado como se gestiona la conectividad del módulo SIM800L). Se verifica si la conexión GPRS está activa, si no es así, imprime contrario, comienza un bucle while que intenta reconectar hasta cinco veces o hasta que se restablezca la conexión. En cada iteración del bucle, envía un comando AT para reiniciar el módulo SIM800L y espera tres segundos antes de configurar el módulo SIM800L.

2.2.3 Guardado temporal y envío de datos.

La novedad en funcionalidad del programa consiste en la conexión GPRS, para ello es necesario el módulo SIM800L, en este caso como se trata de la TTGO T-Call SIM800L dicho módulo está integrado, sin embargo, siempre es necesario establecer comunicación con él, (si se está utilizando un módulo GSM individual, es necesario establecer comunicación serial través de interfaz UART y una alimentación externa).

En la documentación (Xinyuan-LilyGO, 2023) de la placa TTGO T-Call SIM800L V1.4, se menciona que es necesario inicializar los pines del módulo SIM800L. Esto se hace para establecer la comunicación entre el chip ESP32 y el módulo SIM800L a través de una interfaz serie.

Tabla 12

Pines dedicados para el SIM800L

Nombre	(v1.3)Pins	(v1.4)Pins
MODEM TX	26	26

MODEM RX	27	27
MODEM PWRKEY	4	4
MODEM RST	5	5
MODEM POWER	23	23

En la placa TTGO T-Call SIM800L V1.4, estos pines ya están conectados internamente, por lo que no es necesario realizar ninguna conexión adicional, simplemente se inicializan en el void setup, el programa establece comunicación serial entre el ESP32 y el módulo utilizando HardwareSerial.h.

2.2.3.1 Conexión GPRS.

Se llama a la función configurarSim800L en el setup, que envía una serie de comandos AT al módulo SIM800L para configurarlo:

```
void configurarSim800L() {
  String comandosAT[] = {
    "AT","AT+CREG?","AT+CSQ","AT+COPS?","AT+CGATT?","AT+CIPSHUT","AT+CIPSTATU
","AT+CIPMUX=0","AT+CSTT=\"internet.ideasclaro\"","AT+CIICR","AT+CIFSR","AT+S
APBR=3,1,\"Contype\",\"GPRS\"","AT+SAPBR=3,1,\"APN\",\"internet.ideasclaro\""
, "AT+SAPBR=1,1", "AT+SAPBR=2,1"
  };
  for (String comando : comandosAT) {
    comandoATConfig(comando);
    delay(700);
  }
}
void comandoATConfig(const String& comando) {
  SerialAT.println(comando);
  delay(200);
  String resp = SerialAT.readString();
  if (resp.indexOf("OK") != -1) {
    conexionGPRS = true;
  } else {
    conexionGPRS = false;
  }
}
```

Fragmento 10 Funciones para inicializar la conectividad del módulo SIM800L

Los comandos AT (SIMCom, SIM800 Series AT Command Manual, 2014), son instrucciones específicas que permiten configurar y controlar el módulo SIM800L para que pueda establecer conexión GPRS y comunicarse con la red móvil, para ello estos comandos se almacenan en un arreglo y se envían uno por uno utilizando la función comandoATConfig cada 700 milisegundos. comandoATConfig toma

cada comando y lo envía al módulo a través del puerto serie cada 200 milisegundos leyendo la respuesta si contiene un “OK” y modificando la variable global conexionGPRS.

Es posible utilizar librerías para utilizar el módulo GSM/GPRS, sin embargo, actualmente suelen ocupar mayor memoria y esta debe de estar bien mantenida para el hardware, en cambio trabajar con los comandos AT directamente, aunque requiere mayor complejidad, se tiene control total y menor sobrecarga de recursos. Los comandos AT son esenciales y cada uno realiza una tarea especifica:

Tabla 13

Comandos AT utilizados para iniciar la conectividad del SIM800L

AT	**Comando básico para verificar la comunicación con el SIM800L**
AT+CREG?	Consulta el estado del registro de red. Proporciona información sobre si el módulo SIM800L está registrado en la red móvil.
AT+CSQ	Consulta la calidad de la señal RSSI en dBm, es decir proporciona la intensidad de la señal de la red móvil.
AT+COPS?	Consulta el operador de red al que está conectado el módulo SIM800L.
AT+CGATT?	Consulta el estado del adjunto de GPRS. Indica si el módulo está adjunto a la red GPRS.
AT+CIPSHUT	Cierra todas las conexiones GPRS previamente abiertas y deshabilita el contexto PDP (Packet Data Protocol).
AT+CIPSTATUS	Consulta el estado de la conexión GPRS. Puede proporcionar información sobre si la conexión está activa, en espera o desconectada.
AT+CIPMUX=0	Deshabilita la multiplexación de canales en el módulo SIM800L.
AT+CSTT="internet.ideasclaro"	Configura el punto de acceso (APN) de la red móvil.

AT+CIICR	Inicia la conexión GPRS. Después de configurar el APN y enviar este comando, el módulo intentará conectarse a la red GPRS.
AT+CIFSR	Este comando se utiliza para obtener la dirección IP asignada al módulo SIM800L después de establecer una conexión GPRS.
AT+SAPBR=3,1, "Contype","GPRS"	Configura el tipo de conexión como GPRS ("Contype" se establece en "GPRS") para la conexión por paquetes de datos.
AT+SAPBR=3,1,"APN", "internet.ideasclaro"	Configura el APN nuevamente, asegurándose de que el módulo esté preparado para la conexión GPRS.
AT+SAPBR=1,1	Activa la conexión GPRS, permitiendo que el módulo se conecte a la red GPRS con la configuración proporcionada.
AT+SAPBR=2,1	Consulta el estado de la conexión GPRS y proporciona información sobre la dirección IP asignada.

Si la respuesta de cada uno de los comandos es satisfactoria, se establece conexión GPRS y está todo preparado para enviar las lecturas de los sensores.

2.2.3.2 Guardado y envío de paquetes de datos a la hoja de cálculo.

Los datos se almacenan en una estructura array para enviarlos posteriormente. Esto permite conservar su integridad, ya que un envío en tiempo real a la hoja de cálculo provocaría la pérdida de algunas lecturas de CO_2, temperatura, humedad, geolocalización y hora por el retraso o la velocidad de la conexión. La frecuencia de las lecturas que se registran en la aplicación es de un segundo, lo cual es demasiado rápido para garantizar un envío confiable. En las pruebas realizadas, la velocidad máxima de registro en la hoja de cálculo fue de una frecuencia de tres segundos con una conexión GPRS estable.

Cuando la medición termina los datos almacenados son llamados para enviarse con una frecuencia adecuada. Los datos almacenados corresponden cuando la gráfica de concentración de CO_2 vs tiempo se está generando en la aplicación, esto quiere decir que las lecturas de geolocalización están detenidas,

permitiendo un almacenamiento fijo de éstas, además la frecuencia de la toma de las lecturas se va registrando en la hora dada por el RTC que también se almacena.

En la función Licor se realiza una condicional para llamar a la función almacenarDatos que recibe como parámetro la hora y que también se registra en la función Licor con la variable global now, cuando la bandera bombaActivada y envioInicio son verdaderas.

```
struct Datos {
  float co2Value, temperatura, humedad, latitud, longitud;
  int precision, satelites;
  DateTime hora;
};
Datos datosAlmacenados[300];
int indiceAlmacenamiento = 0,

void almacenarDatos(DateTime hora) {
  if (indiceAlmacenamiento < 300) {
    Datos datosActuales;
    datosActuales.co2Value = co2Value;
    datosActuales.temperatura = temperatura;
    datosActuales.humedad = humedad;
    datosActuales.latitud = latitud,6;
    datosActuales.longitud = longitud,6;
    datosActuales.precision = precision;
    datosActuales.satelites = satelites;
    datosActuales.hora = hora;
    datosAlmacenados[indiceAlmacenamiento] = datosActuales;
    indiceAlmacenamiento++;
    Almacenamiento=String(indiceAlmacenamiento)+" datos";
  }
}
```

Fragmento 11 Función de almacenamiento temporal de datos.

La función almacenarDatos, solo se ejecuta cuando el índice de almacenamiento es menor a 300, es decir 5 minutos de lectura en la generación del gráfico de CO_2 vs tiempo lo cual es suficiente. El utilizar un almacenamiento temporal significa una supervisión de la memoria RAM del ESP32, en la compilación del programa en el IDE de Arduino se tiene una descripción útil de cómo está utilizando la memoria el microcontrolador.

```
   El Sketch usa 1076981 bytes (82%) del espacio de almacenamiento de programa. El máximo es 1310720 bytes.
   Las variables Globales usan 48084 bytes (14%) de la memoria dinámica, dejando 279596 bytes para las variables locales. El máximo es 327680 bytes..
```

Fragmento 12 Información en la ejecución del programa en el IDE Arduino

En la memoria Flash que es donde se almacena el programa que se carga en el microcontrolador se están utilizando 1,076,981 bytes de los 1,310,720 bytes disponibles, lo que representa el 82% del espacio de programa (Systems, Espressif Systems, 2021).

Por otro lado en la memoria RAM que es la que interesa porque es utilizada por las variables globales y locales del programa. Actualmente, éstas utilizan 48,084 bytes de los 327,680 bytes disponibles, lo que representa el 14% de la memoria dinámica. Entonces haciendo un cálculo aproximado del tamaño de la estructura array creada suponiendo que es llenada con los 300 datos, se tiene que tener en cuenta el tipo de datos utilizados por ejemplo las variables co2Value, temperatura, humedad, latitud, longitud, son tipo float, generalmente ocupan 4 bytes en la mayoría de plataformas, las variables precision y satélites al ser tipo int ocupan 2 bytes, por otro lado la variable DateTime hora, depende de cómo se defina esta estructura. Si asumimos que es una estructura con tres campos (hora, minuto y segundo), y cada uno de estos campos es un tipo int, entonces ocuparía $3 \cdot 2 = 6$ bytes.

$$4+4+4+4+4+2+2+6=32 \text{ bytes}$$
$$\text{Tamaño de la estructura llena } = 32 \text{ bytes } 300$$
$$\text{Tamaño de la estructura llena } = 9600 \text{ bytes}$$

En el mejor de los casos este sería el tamaño consumido por la estructura con un índice de 300 datos, sin embargo se ha supervisado con índices mayores, ya que al exceder los límites de la memoria disponible el programa se vuelve inestable y requiere reinicios constantes inesperados. Afortunadamente el ESP32 contiene 520Kb de memoria RAM (System, 2021). El tamaño del índice corresponde a un tiempo de medición de 5 minutos, lo cual se puede dar en una condición anómala en un punto.

En cuanto al envío del paquete de datos es necesario formatear el array en un JSON para realizar la solicitud HTTP POST, en el Fragmento 12 se observa la función de enviar.

```
void enviar() {
  if(conexionGPRS){
    Serial.println(String(indiceAlmacenamiento)+" a enviar");
    String datos = "[";
    for (int i = 0; i < indiceAlmacenamiento; i++) {
      envioEstado = "Preparando "+String(i+1)+ " datos";
      procesos();
      delay(100);
      String formattedTime = String(datosAlmacenados[i].hora.hour()) + ":"
                           + String(datosAlmacenados[i].hora.minute()) + ":"
                           + String(datosAlmacenados[i].hora.second());

      datos += String("{")
```

```
            + "\"value1\":\"" + formattedTime + "\","
            + "\"value2\":\"" + String(datosAlmacenados[i].temperatura) + "\","
            + "\"value3\":\"" + String(datosAlmacenados[i].humedad) + "\","
            + "\"value4\":\"" + String(datosAlmacenados[i].co2Value) + "\","
            + "\"value5\":\"" + String(numeroRecibido) + "\","
            + "\"value6\":\"" + String(datosAlmacenados[i].latitud,6) + "\","
            + "\"value7\":\"" + String(datosAlmacenados[i].longitud,6) + "\""
            + (i < indiceAlmacenamiento - 1 ? "}," : "}");
    }
    datos += "]";
    comandoATEnvio("AT+HTTPINIT");
    comandoATEnvio("AT+HTTPSSL=1");
    comandoATEnvio("AT+HTTPPARA=\"CID\",1");
    comandoATEnvio("AT+HTTPPARA=\"URL\",\"https://script.google.com/macros/s/AKfycbwA6
U5-pYdu7zXawaP_H1XOoobycMBJkc1OMcqGO5eM6s2mRyp6oUqjHCGHGb2Cc2xrWA/exec\"");
    comandoATEnvio("AT+HTTPPARA=\"CONTENT\",\"application/json\"");
    comandoATEnvio("AT+HTTPDATA=" + String(datos.length()) + ",10000");
    delay(1000);
    comandoATEnvio(datos);
    delay(1000);
    comandoATEnvio("AT+HTTPACTION=1");
    delay(5000);
    String response;
    while (SerialAT.available()) {
      response += (char)SerialAT.read();
    }
    Serial.println(response);
    comandoATEnvio("AT+HTTPTERM");

  }else{
    Serial.println("Sin conexion Datos no enviados");
  }
  envioEstado = "Inactivo";
  procesos();
}
```

Fragmento 13 Función para enviar paquete de datos a la hoja de cálculo

Primero, la función verifica si existe una conexión GPRS. Si es así, la función procede a enviar los datos. De lo contrario, la función pasa de largo y no procede a enviar nada.

Si existe una conexión GPRS, la función crea una cadena de texto que contiene los datos que se enviarán al Appscript de Google Sheets. La cadena de texto está en formato JSON y contiene los siguientes campos:

- value1: La hora en que se recopilaron los datos.
- value2: La temperatura.
- value3: La humedad.
- value4: La concentración de CO_2.
- value5: El número del punto de medición recibido.

- value6: La latitud.
- value7: La longitud.

Una vez que se ha creado la cadena de texto, la función envía una serie de comandos AT al módulo SIM800L para configurar la conexión HTTP y enviar los datos al script de Google Sheets. Los comandos AT que se utilizan son los siguientes (SIMCom, SIM800 Series AT Command Manual, 2014):

- AT+HTTPINIT: Inicializa el servicio HTTP del módulo SIM800L.
- AT+HTTPSSL=1: Habilita las conexiones SSL/TLS.
- AT+HTTPPARA="CID",1: Configura el identificador de conexión a 1.
- AT+HTTPPARA="URL",<URL del script de Google Sheets>: Configura la URL de destino como la implementación del Ascript de Google Sheets.
- AT+HTTPPARA="CONTENT","application/json": Configura el tipo de contenido de la solicitud a JSON.
- AT+HTTPDATA=<longitud de los datos>,10000: Configura la longitud de los datos a enviar.
- <datos>: Los datos a enviar al script de Google Sheets.
- AT+HTTPACTION=1: Inicia la solicitud HTTP POST.

Después de enviar la solicitud HTTP, la función espera a que el módulo SIM800L responda llamando también a la función y una vez que se recibe la respuesta, la función la imprime en el monitor serial y finalmente, se envía el comando AT+HTTPTERM para cerrar la conexión HTTP.

Al principio, esta función operaba mediante solicitudes GET al Appscript de Google Sheets para enviar paquetes de datos, realizando una solicitud para cada fila de la estructura almacenada. Sin embargo, este enfoque resultaba poco práctico, aunque los datos se recibían uno por uno en la hoja de cálculo, existía la posibilidad de que se perdiera más de uno por el tema de la conectividad. Además, la espera para completar el envío de todo el paquete, especialmente cuando era extenso, resultaba excesiva.

La solución a esta problemática fue enviar el array almacenado de una sola vez mediante una solicitud POST. A pesar de su eficiencia, este método requería realizar ajustes en el Appscript de Google Sheets.

Por otro lado para la gestión de la conectividad GPRS la bandera conexionGPRS recorre diferentes partes del código para establecerse como verdadera o falsa, pasando por la función comandoATConfig y la función verificarConexionGPRS del Fragmento 13 del código.

```
void verificarConexionGPRS(){
  Serial.println("Verificando conexion");
  SerialAT.println("AT+CGATT?");
  delay(1000);
  String resp = SerialAT.readString();
  if (resp.indexOf("+CGATT: 1") != -1) {
    conexionGPRS = true;
    Serial.println("Conectado");
  }else{
    conexionGPRS = false;
  }
}
```

Fragmento 14 Función de gestión de la conectividad GPRS

La función verificarConexionGPRS() verifica si el módulo SIM800L está conectado a una red GPRS. Para ello, envía el comando AT+CGATT? al módulo SIM800L y busca la cadena "+CGATT: 1" en la respuesta. Si encuentra esta cadena, la función establece la variable conexionGPRS en true, lo que indica que hay una conexión GPRS. De lo contrario, la función establece la variable conexionGPRS en false, lo que indica que no hay una conexión GPRS.

Esta función se llama desde el loop principal cada minuto siempre y cuando la bomba no este activa, si algún momento la conexión a internet se llegara a perder para ello la aplicación podrá enviar el comando “M” y reiniciará el modem con la función del Fragmento 8.

Capítulo 3.

3 Actualización de la aplicación móvil: “Medidor de CO_2”

En (Arriaza Carrillo & Aguilar Vega, 2022), se ofrece una visión general del entorno de programación en Android Studio, junto con demostraciones generales de los métodos más relevantes empleados en la aplicación. El programa está escrito en Java, mientras que la vista y los componentes gráficos se definen en XML.

Este capítulo se propone brindar explicaciones y descripciones sobre el funcionamiento de la aplicación, ofreciendo una guía para los usuarios sobre el uso de la primera versión. Además, se detallarán las correcciones y mejoras implementadas a nivel de código. Funcionamiento de la aplicación “Medidor de CO_2” V1.

Ilustración 26 Interfaz inicial al abrir la primera versión de la aplicación "Medidor de CO_2"

En una medición para utilizar la aplicación se deben seguir los siguientes pasos.

Antes de abrir la app se deben otorgar permisos de almacenamiento y Bluetooth la aplicación “Medidor de CO_2”, y vincular el CPU por Bluetooth.

1. Al abrir la aplicación el usuario debe presionar el botón buscar para encontrar todos los dispositivos que han sido vinculados por Bluetooth con el teléfono móvil.
2. A continuación se debe abrir un “Spinner” donde se desplegará una lista de todos los dispositivos encontrados y se debe seleccionar CO2_Meter_01 el cual es el nombre del CPU.
3. Luego el usuario debe presionar el botón “Conectar” y si ocurre una conexión exitosa por Bluetooth con el CPU recibirá un mensaje de “CONEXIÓN EXITOSA”.

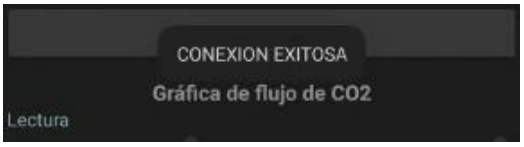


Ilustración 27 Mensaje de conexión exitosa con el CPU.

Ilustración 28 Pantalla de la aplicación ya conectada al CPU.

4. Cuando el dispositivo esté conectado automáticamente en donde dice “Medición de CO_2” se empezará a mostrar el valor de concentración de CO_2 en ppm, en los gráficos de temperatura y humedad[7] como se muestra en la Ilustración 17.

   Para obtener la geolocalización actual en la medición probablemente tenga que esperar unos minutos a que el módulo GPS encuentre satélites, de los contrario obtendrá 0 en todas las lecturas.
5. Cuando ya se tenga la ubicación del punto a medir en el suelo volcánico se debe colocar el número de punto. En el campo de medición primero se traza una línea para realizar las mediciones, en [5 Fig.88] se observa el trazo de esta línea en el Volcán de Santa Ana con 100 puntos, y de ellos se escogieron 20 puntos para ser analizados. Entonces en la aplicación el usuario debe ingresar el número de punto en orden de acuerdo a la distribución que se ha tomado en la línea.
6. Luego el usuario debe desplazarse hacia la segunda pantalla que contiene el gráfico y presionar el botón “Start” lo que hará que la bomba se encienda, debe sostener la cámara de acumulación un momento antes para desplazar el gas acumulado en las mangueras y el sensor de CO_2 y tomar un poco unas muestras de concentración del ambiente.
7. Después del paso 8 se debe colocar la cámara de acumulación en el suelo lo más herméticamente posible y observar cómo se desarrolla la gráfica de concentración de CO_2 vs tiempo.

[7] El gráfico de temperatura y humedad se actualizará cada 10 segundos.

Ilustración 29 Proceso de la generación de la gráfica CO_2 vs tiempo.

8. Ya con la gráfica generada y con un comportamiento aparentemente lineal desde el momento que se puso la cámara de acumulación en el suelo se debe detener el gráfico presionando el botón “Stop” esto apagará la bomba inmediatamente y se dejará de absorber aire en las mangueras de la cámara.
9. Para encontrar el flujo se deben ingresar dos límites de la gráfica y la aplicación calculará una regresión lineal con un coeficiente de correlación R^2, tal cual se explicó en el capítulo 1.
10. Finalmente para realizar una nueva medición se debe presionar el botón desconectar y repetir el proceso.

Como se observa el proceso de medición es sencillo y no requiere más que seguir los pasos anteriormente descritos para utilizar la aplicación. Sin embargo uno de los problemas que posee en esta primera versión es que muchas veces el usuario quiere empezar a utilizarla sin antes darle permisos para almacenamiento y Bluetooth. Esto hace que la aplicación caiga en un “SecurityException” y se cierra de

golpe. Una SecurityException es un error de Android que se produce cuando una aplicación intenta realizar una acción que requiere un permiso que no tiene. En el caso de los permisos de Bluetooth, una SecurityException se produce cuando una aplicación intenta realizar una acción como conectar a un dispositivo Bluetooth, escanear dispositivos Bluetooth o anunciarse a otros dispositivos Bluetooth.

Por otro lado al final en el paso 12 se requiere desconectar el CPU por Bluetooth, aplicación se cierra porque finaliza la actividad. Esto se tiene que hacer para cada medición que se quiera tomar y resulta ser un poco tedioso para el usuario volver a repetir el paso 3,4 y 5, en lugar del botón "Desconectar" se incluirá un botón llamado "Reiniciar" para tomar una nuevas mediciones indefinidamente, no finalizar la actividad ni desconectar el Bluetooth, esto hará una experiencia más dinámica y más rápida a la hora de medir.

3.1 Mejoras para la versión 2.0 de "Medidor de CO_2".

3.1.1 Modificaciones a la interfaz gráfica.

Como se mostró la sección anterior, la interfaz de la aplicación está escrita en XML y éste se genera con cada componente agregado. En las Ilustraciones del inciso anterior se puede observar cómo es la interfaz. La aplicación hace uso de dos componentes gráficos, la librería Hellochart para generar el gráfico de CO_2 vs tiempo, en tiempo real, y los gráficos de pastel tipo "gauge" para la temperatura y humedad que usan de la librería "pawelkleczkowski – custom gauge".

En la siguiente Ilustración 30 se muestra un rediseño en la distribución de los componentes gráficos, nuevos botones y un nuevo panel que muestra información del CPU.

El rediseño debe seguir siendo limpio, ordenado y fácil de utilizar sin cambiar las acciones básicas que realiza la aplicación en conjunto con la estación móvil. Es importante que de acuerdo a las nuevas funcionalidades del CPU, como enviar datos, el control para que este realice esa tarea debe ser controlada desde la aplicación. Obsérvese en la Ilustración 29 que se ha agregado un nuevo botón llamado "Enviar" el cual enviará un comando por Bluetooth al CPU para que éste entre a la función del programa que se encarga de esa tarea, como se explicó en el capítulo 2. Este botón únicamente funcionará o tendrá efecto si en el panel de "Procesos del CPU" se observa que la conexión GPRS está activa.

Buscar dispositivos vinculados guardados

Spinner de la lista de los dispositivos guardados

Conectar con el CPU

Lectura de CO_2

Actualizar GPS

Lectura de temperatura y humedad

Panel de información sobre la conectividad GPRS del CPU,GPS, Almacenamiento y envío de datos

Botón para reconectar manualmente el módulo SIM800l a la red móvil

Ingreso de punto de medición actual.

Gráfico de concentración de CO_2 vs tiempo

Botón para enviar los datos del gráfico a la hoja de cálculo

Sección para el cálculo del flujo

Botón para reiniciar la interfaz y empezar una nueva medición

Ilustración 30 Detalles de la nueva interfaz de la aplicación.

Observe el panel de información del CPU en la Ilustración 31.

Ilustración 31 Panel de información del CPU.

Este panel es como una pantalla del CPU que muestra información relevante acerca de las acciones que está realizando. La pantalla muestra 4 estados.

- GPRS: Cuando la SIM800L está conectada a la red móvil y hay señal GPRS entonces en el panel se mostrará “Conectado” de lo contrario se mostrará “Desconectado”.
- GPS: El módulo Neo 6M muchas veces necesita un tiempo determinado para obtener lecturas de geolocalización, en ese tiempo cuando no ha obtenido lecturas en el panel se mostrará “No Activo” y si se presiona el botón de “Actualizar Geolocalizador” se obtendrá 0 en latitud, longitud, precisión y número de satélites. Si por el contrario el GPS si ha obtenido lecturas, entonces el panel dirá “Activo” y al presionar el botón “Actualizar Geolocalizador” se obtendrán esos nuevos datos. El módulo se actualiza cada 10 segundos, pero si se enciende la bomba detendrá de actualizarse un momento.
- Almacenamiento: Esto hace referencia a la cantidad de datos que se han registrado luego de generar el grafico de CO_2 vs tiempo en la aplicación. Esos datos son los que hay guardados en el búfer temporal[8].Cuando los datos son enviados completamente el búfer se vacía y se mostrarán “0 datos”
- Envío: Este estado refleja varias cosas, cuando se presiona el botón “Enviar” se muestran el número de datos que se van preparando y luego mostrar “Enviando”, esto dura apenas unos segundos ya que el envío de datos se realiza de manera rápida.

[8] Siempre se guardará un paquete de datos aunque no se tenga señal GPRS, sin embargo luego de reiniciar la medición se vaciará para guardar un nuevo paquete y así sucesivamente.

Este panel puede mostrarle al usuario información acerca de cómo se está utilizando la aplicación en conjunto con la maleta de la estación móvil. En la siguiente sección se mostrará la lógica de programación que contiene este panel.

Por otro lado, se encuentra el botón “Reiniciar Modem”, cuya función es enviar un comando por Bluetooth al CPU con el propósito de reiniciar el SIM800L hasta que este pueda restablecer su conexión con la red móvil. Este procedimiento reviste gran importancia, ya que en ocasiones puede ocurrir la pérdida de la cobertura. Dado que el CPU se encuentra encerrado, resulta difícil verificar el estado de la conexión, a pesar de que el SIM800L disponga de diferentes modos de parpadeo para su indicador LED de conexión. Esto se vuelve particularmente desafiante en ambientes con una intensa luminosidad, como ocurre en el campo de medición. Por lo tanto, resulta esencial monitorear el panel para realizar o no el reinicio del modem y asegurar la transmisión efectiva de datos.

3.1.2 Modificaciones a nivel lógico.

3.1.2.1 Conexión Bluetooth y manejo de permisos.

La primera corrección realizada consiste en darle una advertencia al usuario que la aplicación necesita permisos de Bluetooth y almacenamiento y funcionar. Esto se traduce en el siguiente fragmento.

```
private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws
IOException {
    try {
        int permissionCheck = ContextCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH);
        if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
            return device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        } else {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.BLUETOOTH}, REQUEST_BLUETOOTH_PERMISSION);
            return null;
        }
    } catch (SecurityException e) {
        Log.e(TAG, "Error al crear el socket Bluetooth", e);
        return null;
    }
}
```

Fragmento 15 Creación de advertencia de uso de permisos a la aplicación.

Para implementar la mejora, el código utiliza un bloque try-catch. Dentro del bloque try, él se verifica si el permiso está disponible utilizando el método ContextCompat.checkSelfPermission(). Si el permiso está disponible, el código crea el socket Bluetooth utilizando el método

device.createRfcommSocketToServiceRecord(). Si el permiso no está disponible, el código solicita el permiso al usuario utilizando el método ActivityCompat.requestPermissions().

Esto se ve reflejado de la siguiente manera en la Ilustración 32 al abrir la aplicación.

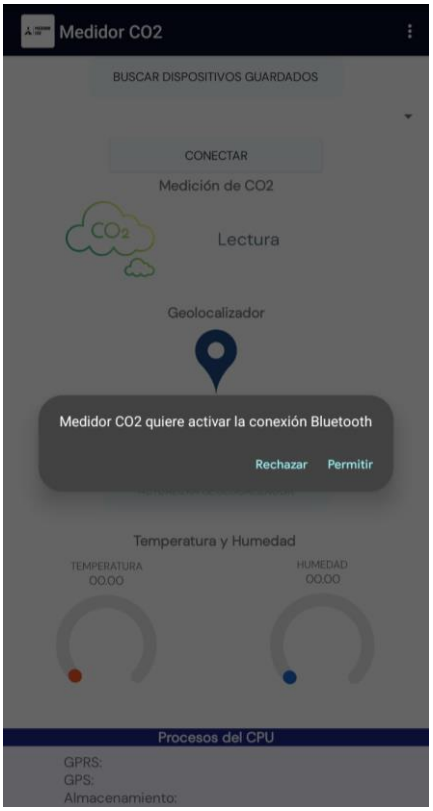


Ilustración 32 Solicitud de permisos para la aplicación.

La exigencia de preguntar al usuario o solicitar permisos al abrir la aplicación en las nuevas versiones de Android es una medida de seguridad que busca proteger la privacidad y la seguridad de los usuarios.

En las versiones anteriores de Android, las aplicaciones podían acceder a los datos y recursos del dispositivo sin el permiso del usuario. Esto podía suponer un riesgo para la privacidad del usuario, ya que las aplicaciones podían recopilar datos personales sin su conocimiento o consentimiento (Orellana, 2021).

Con la nueva exigencia, las aplicaciones deben solicitar el permiso del usuario antes de poder acceder a los datos o recursos del dispositivo. Esto permite al usuario controlar qué aplicaciones tienen acceso a sus datos y recursos, y tomar decisiones informadas sobre la privacidad y la seguridad de sus dispositivos.

Para que la Ilustración 32 se lleve a cabo se utiliza el siguiente Fragmento del programa en Java.

```
public void ConectarDispBT() {
            try {
                int permissionCheck = ContextCompat.checkSelfPermission(this,
        Manifest.permission.BLUETOOTH);
                if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
                    BluetoothDevice device = mBtAdapter.getRemoteDevice(address);
                    try {
                        btSocket = createBluetoothSocket(device);
                    } catch (IOException e) {
                        Toast.makeText(getBaseContext(), "La creacción del Socket
        fallo", Toast.LENGTH_LONG).show();
                    }
                    try {
                        btSocket.connect();
                        Toast.makeText(getBaseContext(), "CONEXION EXITOSA",
        Toast.LENGTH_SHORT).show();
                        isConnected = true;
                    } catch (IOException e) {
                        try {
                            btSocket.close();
                        } catch (IOException e2) {
                        }
                    }
                    MyConexionBT = new ConnectedThread(btSocket);
                    MyConexionBT.start();
                } else {
                    ActivityCompat.requestPermissions(this, new
        String[]{Manifest.permission.BLUETOOTH}, REQUEST_BLUETOOTH_PERMISSION);
                }
            } catch (SecurityException e) {
                Log.e(TAG, "Error al conectar con el dispositivo Bluetooth", e);
            }
        }
```

Fragmento 16 Método Conectar Dispositivo Bluetooth validado

Verifica si el permiso Bluetooth está disponible antes de intentar conectar con el dispositivo Bluetooth. Si el permiso no está disponible, el código solicita el permiso al usuario. Esto es importante porque la aplicación necesita el permiso Bluetooth para poder comunicarse con otros dispositivos.

Establece la variable isConnected en true después de conectar con el dispositivo Bluetooth. Esto permite a la aplicación saber si está conectada a un dispositivo Bluetooth y realizar las acciones adecuadas en consecuencia.

Maneja la excepción SecurityException que puede ocurrir al conectar con el dispositivo Bluetooth. Esta excepción puede ocurrir si el usuario no ha otorgado el permiso Bluetooth a la aplicación.

Luego para el método de los “Dispositivos Vinculados” en (Arriaza Carrillo & Aguilar Vega, 2022) se presenta una operación para obtener dispositivos Bluetooth vinculados, sin un control explícito sobre

la disponibilidad de permisos o el manejo de posibles excepciones. Las correciones introducen mejoras notables en términos de seguridad y control de flujo.

```
public void DispositivosVinculados() {
    try {
        int permissionCheck = ContextCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH);
        if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
            VerificarEstadoBT();
            mBtAdapter = BluetoothAdapter.getDefaultAdapter();
            Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

            if (pairedDevices.size() > 0) {
                mAddressDevices.clear();
                mNameDevices.clear();

                for (BluetoothDevice device : pairedDevices) {
                    mAddressDevices.add(device.getAddress());
                    mNameDevices.add(device.getName());
                }
                DisEncontrados.setAdapter(mNameDevices);
            } else {
                String noDevices = "Ningun dispositivo pudo ser
emparejado".toString();
                mAddressDevices.add(noDevices);
                mNameDevices.add(noDevices);
            }
        } else {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.BLUETOOTH}, REQUEST_BLUETOOTH_PERMISSION);
        }
    } catch (SecurityException e) {
        Log.e(TAG, "Error al obtener los dispositivos vinculados", e);
    }
}
```

Fragmento 17 Método para Dispositivos Vinculados

La principal mejora radica en la inclusión de un procedimiento para verificar la disponibilidad de permisos mediante el uso de ContextCompat.checkSelfPermission. En caso de que el permiso no esté otorgado, se solicita al usuario la autorización necesaria a través de ActivityCompat.requestPermissions. Este enfoque proactivo asegura que la aplicación funcione dentro de los límites de los permisos otorgados por el usuario, lo que garantiza una experiencia segura y una operación ininterrumpida.

Además, se incluye un bloque try-catch para manejar posibles excepciones, como la SecurityException. La captura de esta excepción es esencial para proporcionar un manejo adecuado de errores durante la ejecución del programa. La inclusión de este bloque de código permite registrar y manejar errores de seguridad de manera adecuada, ofreciendo al usuario final una experiencia más

robusta y confiable. Las mejoras explicadas también se extienden a los demás métodos y funciones que controlan la conexión Bluetooth como VerificarEstadoBT.

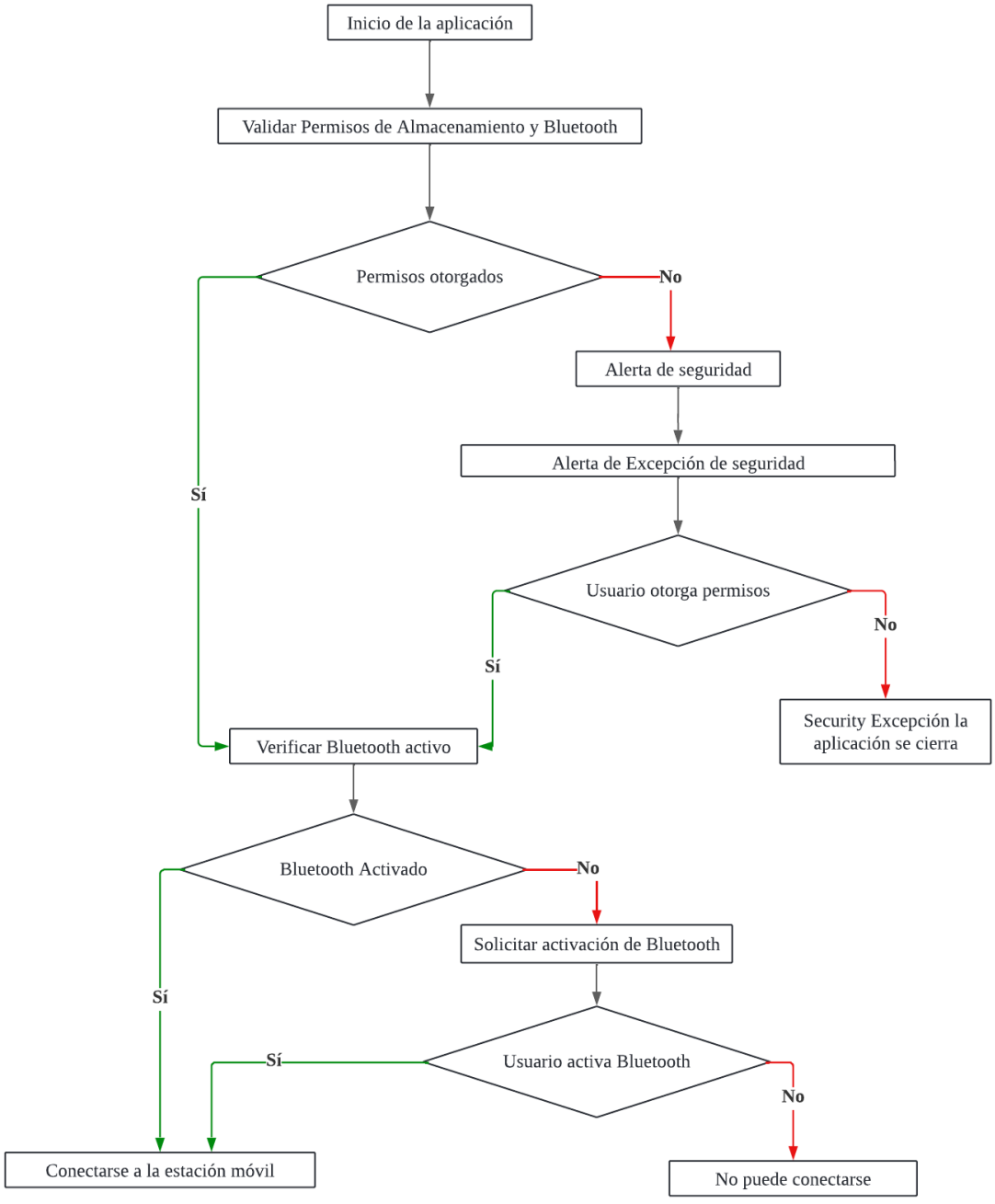


Ilustración 33 Autorización de permisos

3.1.2.2 Manejo de adquisición de datos por Bluetooth.

Para manejar los datos enviados por el microcontrolador en la aplicación el programa (Arriaza Carrillo & Aguilar Vega, 2022) integra un método para manejar mensajes por Bluetooth que se encapsula dentro de un objeto Handler, se encarga de revisar cada parte del mensaje utilizando "msg.what == handlerState"para entender qué tipo de mensaje se ha recibido. Una vez que identifica el mensaje que le interesa (en este caso, aquellos que tienen un indicador especial), empieza a "leer" el contenido del mensaje para encontrar información valiosa (como temperaturas, humedad, coordenadas, etc.). En este proceso de análisis, el código busca un carácter delimitador ("#") (Ver Fragmento 5) para identificar el final de un conjunto de datos. Una vez encontrado este delimitador, el método extrae la porción de datos entre el inicio del mensaje y el delimitador. Posteriormente, se descompone esta información en segmentos más pequeños utilizando la función split(","), lo que permite asignar cada sección a una variable específica según su posición en el mensaje original.

El código (Arriaza Carrillo & Aguilar Vega, 2022) maneja múltiples escenarios de datos entrantes, identificando y asignando valores a variables como la concentración de CO_2, la humedad, la temperatura, la latitud, la longitud y otros parámetros, todos representados en formato de cadenas de texto. Adicionalmente, se efectúa la conversión de ciertos datos a tipos numéricos, particularmente para la temperatura, la humedad y la concentración de CO_2.

Se lleva a cabo un control de flujo para la representación y actualización de los datos en la interfaz de usuario, mostrando la información de temperatura, humedad y concentración de CO2 en campos de texto (TextViews) y los medidores visuales customgauge.

Todo esto se realiza inmediatamente se ha establecido la conexión Bluetooth a través del botón "Conectar" en el paso 5 de los incisos anteriores de este capítulo, sin embargo, cuando el usuario presiona el botón "Start" se abre de nuevo otra comunicación y se cierra la anteriormente descrita ya que esta nueva comunicación lo maneja el botón "Start", esto quiere decir que los botones de "Actualizar gps" y los costomgauge de temperatura y humedad ya no se van actualizar, únicamente se van a tomar en cuenta los valores de CO_2 para formar la gráfica. . Los datos se dividen y asignan a variables correspondientes, en este caso, se extrae la concentración de CO_2 (StrCO2) y una respuesta de control (StrRespIn).

La aplicación toma diferentes acciones en función de la respuesta recibida. Si la respuesta no es "0", actualiza la interfaz de usuario con la información proporcionada. En caso contrario, asume que los datos recibidos son lecturas del sensor de CO_2 y procede a su interpretación.

En este método asociado al botón “Start", el código invoca otra función encargada de crear un archivo CSV para almacenar los datos utilizados en la generación de la gráfica. Originalmente, este archivo se limita a contener datos de CO_2; sin embargo, las mejoras planificadas incluirán la adición de otras variables a este archivo. Asimismo, se eliminará el corte de comunicación que ocurre en la primera sección, optando por un flujo de comunicación único y constante, tanto para la generación de la gráfica como para la visualización de las lecturas en los diversos campos.

```
bluetoothIn = new Handler() {
    @SuppressLint({"HandlerLeak", "SetTextI18n"})
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            recDataString.append(readMessage);

            int endOfLineIndexSharp = recDataString.indexOf("#");
            int endOfLineIndexPercent = recDataString.indexOf("%");

            if (endOfLineIndexSharp > 0 || endOfLineIndexPercent > 0) {
                Log.d("Bluetooth", "mensaje: " + readMessage);

                if (endOfLineIndexSharp > 0) {

                    String dataInPrint = recDataString.substring(0,
endOfLineIndexSharp);

                    try {
                        String[] parts = dataInPrint.split(",");
                        String StrCO2 = parts[0];
                        String StrRespIn = parts[1];
                        StrLat = parts[2]; StrLon = parts[3];
                        StrSat = parts[4]; StrPre = parts[5];
                        StrTemp = parts[6];StrHumd = parts[7];

                        if (!StrTemp.equals("0")) {
                            double DataHumd = Double.parseDouble(StrHumd);
                            double DataTemp = Double.parseDouble(StrTemp);

                            IdTxvTmp.setText(StrTemp + " °C");
                            IdTxvHmd.setText(StrHumd + " %");

                            int DataIntTemp = (int) DataTemp;
                            int DataIntHumd = (int) DataHumd;
                            IdMedidorTemp.setValue(DataIntTemp);
                            IdMedidorHumd.setValue(DataIntHumd);
                        }
                            DataCO2 = Double.parseDouble(StrCO2);

                        Log.d("DEBUG", "CO2: " + DataCO2);
                        IdTxvCo2.setText(DataCO2 + " ppm");
                        if (Graph_CSV) {
                            IdBtnModem.setEnabled(false);
                            String[] sessionTimes = sessionTime.toString().split(" ");
```

```
                            filename = "Datos Medicion " + medicionCount
                                    + " Punto " + Point + "-" + sessionTimes[0]
                                    + "-" + sessionTimes[1] + "-" + sessionTimes[2]
                                    + "-" + sessionTimes[5] + ".csv";

                            IdTxvCo2_G.setText(("CO2: " + DataCO2 + " ppm"));

                            Date measureTime = Calendar.getInstance().getTime();
                            Locale locale = Locale.getDefault();

                            SimpleDateFormat dateFormat = new
SimpleDateFormat("M/dd/yyyy", locale);
                            String formattedDate = dateFormat.format(measureTime);
                            SimpleDateFormat timeFormat = new
SimpleDateFormat("HH:mm:ss", locale);
                            String formattedTime = timeFormat.format(measureTime);
                            String measureLine = formattedDate + "," + formattedTime +
"," + parts[6] + "," + parts[7] + "," + String.format("%.2f", DataCO2) + "," +
pointText +","+ StrLat.replaceAll("[^\\d.]", "") + "," + StrLon.replaceAll("[^\\d.]",
"") + "\n";

                            createFile(measureLine);
                            String currentTime2 = new SimpleDateFormat("HH:mm:ss",
Locale.getDefault()).format(new Date());
                            ListStrHora.add(currentTime2);
                            ListValCO2.add((float) DataCO2);
                            List ListValCO2_2 = new ArrayList();
                            Line line = new
Line(ListValCO2_2).setColor(Color.parseColor("#2729b0")).setCubic(true);

                            for (int i = 0; i < ListStrHora.size(); i++) {
                                ListValCO2_2.add(new PointValue(i,
ListValCO2.get(i)));
                            }

                            List lines = new ArrayList();
                            lines.add(line);
                            LineChartData data = new LineChartData();
                            data.setLines(lines);
                            Axis axis = new Axis();
                            axis.setTextSize(10);
                            axis.setTextColor(Color.parseColor("#03A9F4"));
                            data.setAxisXBottom(axis);
                            data.setAxisXTop(axis);
                            axis.setHasLines(true);
                            Axis yAxis = new Axis();
                            yAxis.setTextColor(Color.parseColor("#03A9F4"));
                            yAxis.setTextSize(10);
                            data.setAxisYLeft(yAxis);
                            yAxis.setHasLines(true);
                            lineChartView.setLineChartData(data);
                            Viewport viewport = new
Viewport(lineChartView.getMaximumViewport());
                            viewport.bottom = 0;
                            viewport.top = 20000;
                            viewport.left = 0;
                            viewport.right = 5000;
```

```
                            lineChartView.setCurrentViewport(viewport);
                            lineChartView.setLineChartData(data);
                            lineChartView.startDataAnimation();
                            lineChartView.startDataAnimation(1000);
                            Log.d("Depuración", "ListStrHora: " +
ListStrHora.toString());
                            Log.d("Depuración", "ListValCO2: " +
ListValCO2.toString());
                        }else{
                            IdBtnModem.setEnabled(true);
                        }
                    } catch (Exception e) {
                    }
                    recDataString.delete(0, endOfLineIndexSharp + 1);
                }
                if (endOfLineIndexPercent > 0 && endOfLineIndexPercent <
recDataString.length()) {
                    String dataInPrint = recDataString.substring(0,
endOfLineIndexPercent);
                    try {
                        String[] parts = dataInPrint.split(",");
                        GPRS = parts[0].trim();
                        String GPS = parts[1];
                        final String Almacen = parts[2];
                        final String Envio = parts[3];

                        if (GPRS.equals("1")) {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    IdTxCPU1.setText("GPRS: Conectado");
                                }
                            });
                        } else {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    IdTxCPU1.setText("GPRS: Desconectado");
                                }
                            });
                        }
                        if (GPS.equals("1")) {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    IdTxCPU2.setText("GPS: Activo");
                                }
                            });
                        } else {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    IdTxCPU2.setText("GPS: No Activo");
                                }
                            });
                        }
                        runOnUiThread(new Runnable() {
```

```
                            @Override
                            public void run() {
                                IdTxCPU4.setText("Envio: "+Envio);
                                IdTxCPU3.setText("Almacenamiento: "+Almacen);
                            }
                        });

                    } catch (Exception e) {
                    }
                    recDataString.delete(0, endOfLineIndexPercent + 1);
                }
            }
        }
    }
};
```

Fragmento 18 Método para manejar los mensajes bluetooth del microcontrolador.

En este código se manejan todos los mensajes recibidos por el microcontrolador, de tal manera que se mantiene un flujo constante, esto significa que los campos de lecturas de la primera sección de la aplicación y medidores de temperatura y humedad siempre se actualizarán aun cuando la gráfica se esté generando, recordar que los mensajes son cadenas de texto separados por coma (“,”) y terminados en (#) o (%) y gracias a ello se pueden procesar cada mensaje e introducirlo en su campo correspondiente.

Como es un flujo constante de mensajes, se necesita que la gráfica únicamente se genere cuando se presiona el botón “Start” y no todo el tiempo desde que se realiza la conexión bluetooth, por ello se utiliza la bandera “Graph_CSV” que controla una condicional y un buen bloque de código para generar el gráfico en tiempo real y guardar los datos en un archivo CSV de manera que en este se va agregando cada línea de la siguiente manera:

Tabla 14
Guardado de datos durante la medición.

Fecha	Hora	Temperatura	Humedad	CO2	Punto	Latitud	Longitud
9/27/2023	11:04:25	30.90	66.10	507.84	1.00	13.969272	-89.574394
9/27/2023	11:04:28	30.90	66.10	507.50	1.00	13.969272	-89.574394
9/27/2023	11:04:28	30.90	66.10	508.14	1.00	13.969272	-89.574394
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Obsérvese en el código que se ha tenido que formatear la hora y la fecha de la manera que muestra el cuadro. La idea de esto es tener un respaldo en todas las mediciones si en un momento no puede enviarse el paquete de datos a la hoja de cálculo en la nube. Cada medición tendrá un archivo CSV, no importa que se trate de una medición diferente para un mismo punto, la razón de esto es porque el método

que realiza el cálculo de correlación y regresión lineal entre dos intervalos, selecciona los datos del archivo actualmente creado.

Con respecto al gráfico en tiempo real que se construye por HelloChart, primero se declaran arrays llamados "ListStrHora" y "ListValCO2", supóngase que llegan 4 datos de concentración de CO_2 (300,400,350 y 500 ppm). Inicialmente los valores se van actualizando de acuerdo a la variable DataCO2 y dichos valores se van agregando a ListValCO2 y ListStrHora guarda los tiempos en que se registraron las lecturas.

Luego se crea un nuevo array para valores de CO_2 llamado ListValCO2_2, esta lista se utilizará para guardar los puntos del gráfico de líneas. En el bucle se hace un recorrido de la lista ListValCO2 que va conteniendo las concentraciones, para convertirlas en puntos que se agregarán a la lista de ListValCO2_2 generando coordenadas, como en el ejemplo esto se vería: (0,300), (1,400) y (2,350), (3,500). El proceso descrito sugiere una animación o un efecto visual en el que, a medida que se agregan más puntos de datos a las listas, el gráfico no se actualiza para incluir solo los nuevos puntos, sino que muestra todos los puntos históricos y los nuevos que se van añadiendo. Esta acumulación de datos en el gráfico refleja un historial o una evolución visual de los valores de CO_2 a lo largo del tiempo, creando una sensación de trayectoria o progresión en el gráfico.

Se insiste que el gráfico no se actualiza dinámicamente con la adición de nuevos datos a la lista ListValCO2. Cada vez que se ejecuta el bucle for que recorre la lista para crear los puntos del gráfico, lo que sucede es que se vuelve a generar el gráfico completo, considerando todos los datos presentes en la lista en ese momento. Es decir, no hay una actualización continua del gráfico con solo los nuevos puntos conforme se añaden al array, sino que cada vez que se actualiza el gráfico, se reconstituye completamente incluyendo todos los datos existentes en la lista en ese momento. Este procedimiento es un manera creativa de desarrollar o aparentar un gráfico dinámico en tiempo real.

A este método se le agrega otro bloque de código que opera dentro del manejador de mensajes y se encarga de procesar una sección específica de datos delimitada por el carácter "%"(Ver Fragmento 7). Tras verificar la presencia y ubicación de dicho delimitador en la cadena de datos recibidos, extrae, analiza y asigna valores a variables representativas de la conectividad GPRS, el estado del GPS, almacenamiento y envío de datos. Estos valores son utilizados para actualizar dinámicamente la interfaz de usuario a través de TextViews del panel de información de procesos del CPU, mostrando mensajes informativos sobre el estado de la conexión y la gestión de datos. Posteriormente, elimina la porción de

datos procesados de la cadena para dejar espacio a nuevas entradas, asegurando así un procesamiento continuo y actualizaciones en tiempo real en la interfaz de la aplicación móvil.

Concluyendo para esta sección ahora el botón “Start” ya no abre otro manejador de mensajes por Bluetooth solo determina el momento para crear el gráfico y guardar los datos en un CSV.

3.1.2.3 Manejo del guardado de archivos.

En relación a la sección de cálculo del flujo, el enfoque detallado se encuentra explicado en cada sección del código original en (Arriaza Carrillo & Aguilar Vega, 2022). La técnica para calcular la regresión lineal y la correlación lineal implica la lectura del archivo CSV, generado de forma inmediata después de completar la gráfica. Esta metodología se mantiene sin cambios, lo que involucra la gestión de un archivo por cada medición realizada. Por ejemplo, si se efectúan tres mediciones en un punto específico, se generarán tres archivos distintos. Esta modificación se realizó debido a que la creación de los archivos se determina por el nombre, que incorpora la fecha, el número de punto y el número de medición (este último se añade automáticamente al nombre si el usuario no cambia la entrada del punto al reiniciar). Antes de abordar el cálculo del flujo, es esencial comprender qué funciones y métodos son requeridos en el manejo del almacenamiento para obtener resultados correctos.

Revisando brevemente los métodos involucrados para esta tarea se tiene lo siguiente:

1. Creación de archivos:

Método createFile, este método de (Arriaza Carrillo & Aguilar Vega, 2022), se encarga de crear un archivo y escribir en él. Específicamente, verifica si el almacenamiento externo está disponible y accesible para escritura antes de escribir en el archivo. En la versión 2 de este método se escribe como la

a. Tabla 14. Se agregan más encabezados tal como la tabla.

Método createFileDos se encarga de escribir información relacionada con mediciones de flujo de CO_2 en un archivo específico, este método se llama luego de haber calculado el flujo. Este método no tiene cambios en la segunda versión. En la

b. Tabla 14 se observa los encabezados del archivo generado para una determinada fecha con el flujo calculado en cada punto para realizar un perfil de la línea que se ha ido midiendo.

2. Disponibilidad de almacenamiento:
    a. Método isExternalStorageWritable: Tiene como objetivo verificar si el almacenamiento externo está en condiciones de escritura. Para ello, se consulta el estado del almacenamiento externo y se determina si está montado y disponible para guardar datos. Esta comprobación es esencial para asegurar que la aplicación pueda almacenar información de manera efectiva en el dispositivo.
    b. Método isExternalStorageReadable: Al igual que el método anterior, este se enfoca en verificar la disponibilidad del almacenamiento externo, pero en este caso, se comprueba si el almacenamiento está listo para lectura. Esto es importante cuando se necesita acceder a datos almacenados previamente.
3. Obtención de directorios específicos:
    a. Método getDocumentStorageDir se encarga de proporcionar el directorio específico donde se almacenarán documentos relacionados con las mediciones de CO_2. Utiliza las directrices del sistema Android para obtener el directorio de documentos público y luego crea un subdirectorio llamado “CO2 measures” dentro de este. Este método resalta la organización y la gestión efectiva de los datos, lo que es crucial para garantizar que los resultados de la investigación estén bien organizados y sean fácilmente accesibles.
    b. Método getDocumentStorageDirDos similar al método anterior, proporciona el directorio para almacenar reportes relacionados con el CO_2. Utiliza el mismo enfoque de obtener el directorio de documentos públicos y crear un subdirectorio llamado “CO_2 Reportes” donde estarán los datos de flujo para diferentes puntos en una fecha determinada. Aquí se subraya la importancia de mantener separados los datos de las mediciones y los reportes, lo que contribuye a una gestión más efectiva de los datos y resultados de la investigación.

En siguiente ilustración se observan las carpetas creadas por la aplicación en donde se guardan los archivos de concentración de CO_2, temperatura, humedad y geolocalización, y los archivos de flujo de CO_2.

Ilustración 34 Carpetas contenedoras de los archivos de medición

Obsérvese que la ruta dirige a las carpetas en el almacenamiento interno del teléfono y en "Documentos". En la modificación realizada se observa sobre todo en el método que controla el botón "Guardar" en donde se le solicita al usuario antes el punto de medición actual. El método del botón se observa en el siguiente Fragmento de código.

```
public void etiqueta(View view) {
    if (isConnected) {

        switch (view.getId()) {
            case R.id.Id_BtnPointMeasure:
                MyConexionBT.write("F");
                etiqueta.setText("Punto de medición: " + idTxtPoint.getText());
                String textoAEnviar = ","+ idTxtPoint.getText().toString();

                MyConexionBT.write(textoAEnviar);

                Log.d("DEBUG", "Texto enviado por Bluetooth: " + textoAEnviar);

                SharedPreferences preferencias = getSharedPreferences("puntos",
Context.MODE_PRIVATE);
                SharedPreferences.Editor Obj_editor = preferencias.edit();
                Obj_editor.putString("id", idTxtPoint.getText().toString());
                Obj_editor.commit();

                Id_BtnPointMeasure.setEnabled(false);
                IdBtnGps.setEnabled(false);

                pointText = idTxtPoint.getText().toString();
                Log.d("Numero", "Punto: " + pointText);
                if (pointText.equals(lastPoint)) {
                    medicionCount++;
                    Log.d("Medicion", "numero: " + medicionCount);

                } else {
                    medicionCount = 1;
                    Log.d("Medicion", "numero: " + medicionCount);
                    lastPoint = pointText;
```

```
                    Log.d("Ultimo Punto", "Punto: " + lastPoint);
                }
                try {
                    Thread.sleep(1000);
                    buttonStartThread.setEnabled(true);
                } catch (Exception errorTiempo) {
                    errorTiempo.printStackTrace();
                }
                break;
        }
        Point = idTxtPoint.getText().toString();
        filename = "Datos Medicion " + medicionCount + " Punto " + Point + "-" +
sessionTimes[0] + "-" + sessionTimes[1] + "-" + sessionTimes[2] + "-" +
sessionTimes[5] + ".csv";

        idTxtPoint.setEnabled(false);
    } else {
        Toast.makeText(getBaseContext(), "No se ha establecido una conexión
Bluetooth", Toast.LENGTH_SHORT).show();
    }
}
```

Fragmento 19 Método para el control de punto de medición

En contraste con la versión 1, el método “etiqueta” en la versión 2 del programa se verifica si hay una conexión Bluetooth establecida antes de enviar datos (esto se verá en una sección más adelante con el control de los botones). Utiliza una cadena de formato "F" seguida del punto de medición que ingresa el usuario y la envía por Bluetooth. Además, maneja un contador de mediciones (“medicionCount”), que se incrementa si el punto actual es igual al último punto registrado, esto se hace debido a que se necesitan crear archivos individuales para cada medición y así poder hacer la regresión lineal para el cálculo del flujo, el método utiliza la información del punto y la fecha para construir el nombre del archivo CSV.

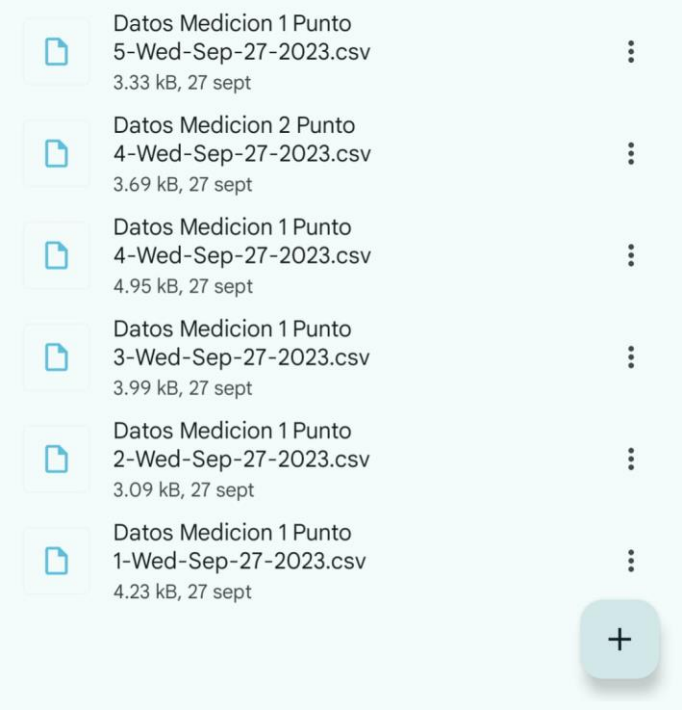


Ilustración 35 Archivos de prueba guardados con el nuevo método

Ambos métodos comparten la funcionalidad de deshabilitar ciertos elementos de la interfaz de usuario y construir el nombre del archivo CSV basándose en el punto y la fecha. Sin embargo, el Método nuevo presenta ventajas al verificar la conexión Bluetooth antes de proceder y al gestionar un contador de mediciones, lo que podría ser crucial en un contexto donde la integridad de los datos y la conexión Bluetooth son prioridades.

3.1.2.4 Cálculo de flujo de CO_2.

Para el cálculo del flujo de CO_2 volcánico, se solicitan al usuario dos límites que corresponden a un valor inferior y superior dentro del intervalo donde se observe un comportamiento aparentemente lineal en la gráfica. La aplicación realiza esta acción al consultar la carpeta denominada “measures” de la Ilustración 34, donde se han registrado los datos de concentración de la gráfica. En la nueva versión, la distribución de las columnas de los datos de medición en este archivo CSV se presenta en la

Tabla 14. Además, el nombre del archivo analizado recién creado sigue la siguiente distribución:

```
Datos <Número de medición> < Número de Punto> <Fecha>
```

Como el csv “Measures” se ha modificado la lectura o la extracción de los datos debe hacerse en otra columna en el método “getFileMeasure” .

```
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
public ArrayList<Float> getFileMeasure() {
    ArrayList<Float> yValues = new ArrayList<>();
    if (isExternalStorageReadable()) {
        File file = new File(getDocumentStorageDir(), filename);
        try {
            BufferedReader br = new BufferedReader(new
FileReader(file.getAbsoluteFile()));
            String line = br.readLine();
            Integer counter = 0;
            while (null != line) {
                line = br.readLine();
                if (line != null) {

                    String[] fields = line.split(SEPARATOR);
                    Float yPoints = Float.parseFloat(fields[4]);
                    yValues.add(yPoints);
                }
            }
            br.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return yValues;
}
```

Fragmento 20 Método para la lectura del archivo de medicion de concentración de CO_2

El método getFileMeasure desempeña un papel fundamental en el sistema de monitoreo de CO_2, diseñado para la lectura de datos almacenados en archivos CSV. Este procedimiento se lleva a cabo en varias etapas.

1. En primer lugar, se inicializa un ArrayList<Float> denominado yValues, destinado a contener los valores de las mediciones recuperadas. Posteriormente, se verifica la accesibilidad del almacenamiento externo para lectura mediante la función isExternalStorageReadable.
2. El siguiente paso consiste en obtener la referencia al archivo CSV, lo cual se logra mediante el método getDocumentStorageDir, encargado de identificar el directorio de documentos, al que se añade el nombre del archivo (“filename”). Acto seguido, se utiliza un BufferedReader

para leer el contenido del archivo línea por línea, omitiendo la primera línea que corresponde al encabezado.

3. Durante el proceso de lectura, cada línea se descompone en campos utilizando un separador específico. El quinto campo, que almacena el valor de la medición de CO_2, se extrae y convierte a tipo Float, aquí es donde se ha hecho el cambio ya que la columna de los datos ha cambiado de posición. Este valor se incorpora al ArrayList previamente creado.
4. El cierre adecuado del BufferedReader es crucial para evitar posibles problemas de recursos. Además, se implementa un manejo de excepciones para gestionar posibles errores, como la falta del archivo o dificultades durante la lectura.
5. En última instancia, el método retorna el ArrayList yValues que alberga los valores de las mediciones de CO_2 extraídos del archivo CSV. Este proceso constituye un componente esencial ya que el array retornado se le aplicará el cálculo de flujo.

El método está diseñado para buscar un archivo CSV específico que contiene los datos de las medidas. El nombre de este archivo se construye utilizando la variable filename, que es inicializada en otras partes del código. En este contexto, filename se establece cuando se realiza una medición o se guarda un conjunto de datos.

La construcción del nombre del archivo incluye información relevante, como el punto de medición, la fecha y la hora. Por ejemplo, en el código proporcionado, se utiliza la variable filename de la siguiente manera (Ver Tabla 14):

```
   filename = "Datos Medicion " + medicionCount + " Punto " + Point + "-" + sessionTimes[0] + "-" + sessionTimes[1] + "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";
```

Esto significa que cada vez que se realiza una medición generando el gráfico, se genera un nombre de archivo único que incorpora la información sobre la medición, el punto, la fecha y la hora. Por lo tanto, el método getFileMeasure sabe exactamente qué archivo CSV buscar, ya que utiliza el mismo nombre de archivo que se utilizó para guardar los datos.

Debido a esto, resulta necesario segmentar los archivos por número de medición, en caso de que el usuario, al reiniciar, opte por llevar a cabo una nueva medición en el mismo punto. El problema que surgiría sin esta implementación sería que, al reiniciar una nueva medición para el mismo punto, los datos de la nueva medición se agregarían al archivo existente. Al intentar calcular el flujo, se

considerarían tanto los datos de la medición anterior como los nuevos, generando confusión al estar almacenados en el mismo archivo. En la Ilustración 36 siguiente se puede apreciar claramente este problema.

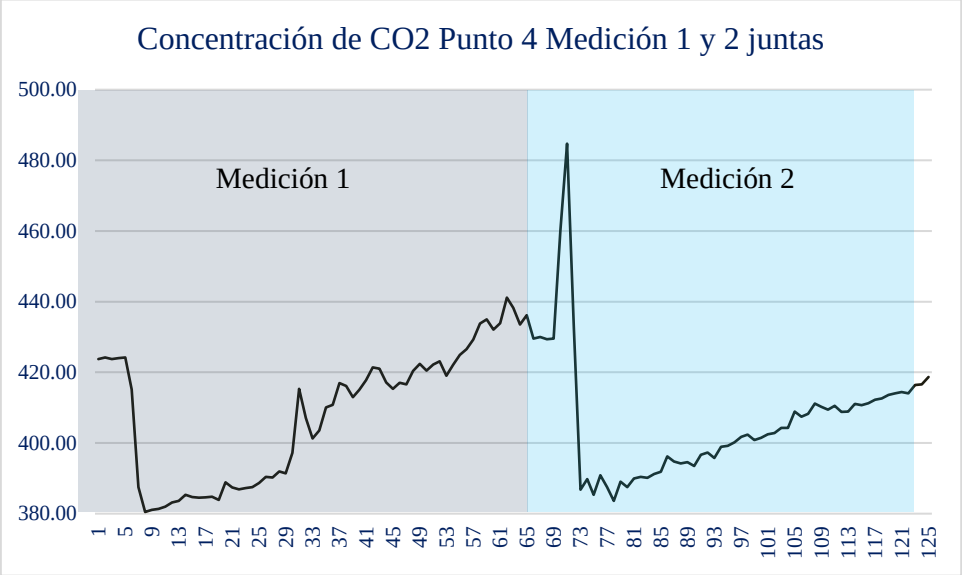


Ilustración 36 Ejemplo: Gráfico generado del archivo del punto 4 sin separar las mediciones

Obsérvese que la gráfica de la Ilustración 36 tiene dos intervalos donde la tendencia se vuelve lineal. Aquí es donde ocurría la confusión de los datos si no se separaban los archivos por número de medición.

Esta confusión surge del método trimYData que tiene como propósito recortar un subconjunto de datos de mediciones almacenados del CSV.

```
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
public ArrayList<Float> trimYData(Integer lima, Integer limb) {
    ArrayList<Float> trimedY = new ArrayList<>();
    ArrayList<Float> yValues = new ArrayList<>(getFileMeasure());
    for (int i = lima; i <= limb; i++) {
        trimedY.add(yValues.get(i));
    }
    return trimedY;
}
```

Fragmento 21 Método para extraer el intervalo con tendencia lineal

Utiliza dos parámetros, “lima” y “limb”, que son proporcionados por el usuario para especificar los límites inferior y superior del rango de datos a recortar. En primer lugar, se inicializa un ArrayList para contener los valores recortados y otro para almacenar todos los datos obtenidos del archivo. Luego, mediante un bucle, la función selecciona y agrega los valores del rango especificado al ArrayList de

valores recortados. Finalmente, retorna este ArrayList, proporcionando así una función para extraer un subconjunto específico de datos de mediciones según los límites definidos. En otras palabras, se obtiene un subconjunto determinado por posiciones, y dichas posiciones se obtienen desde “lima” a “limb”. Este enfoque permite extraer y trabajar con un segmento específico de las mediciones según los límites definidos.

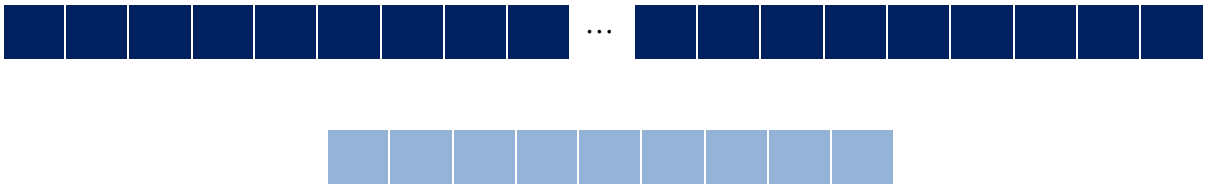


Ilustración 37 Ejecución del método trimYData

En la Ilustración 37 se ejemplifica en forma de cuadros como los datos del ArrayList retornado por el método getFileMeasure, es decir es el conjunto de datos de concentración de CO_2 del archivo recién generado luego de haber creado la gráfica. El método trimYData los toma y busca las posiciones que ha ingresado el usuario y crea un nuevo array trimedY representado por los cuadros de abajo, estos datos son los que contienen la tendencia lineal y van a ser analizados.

Entonces volviendo al problema al no separar los archivos por número de medición el conjunto de datos sería el de la Ilustración 36 y por ejemplo, si el usuario reinicia la medición para el mismo punto (4 en este caso) la gráfica de la aplicación se reinicia (nueva implementación se explicará en la siguiente sección) por lo tanto el eje x de tiempo también, observe que en la gráfica de la Ilustración 36 el segundo intervalo con tendencia lineal se comprende entre los datos 73 a 125 sin embargo, el usuario vería la siguiente gráfico en la aplicación.

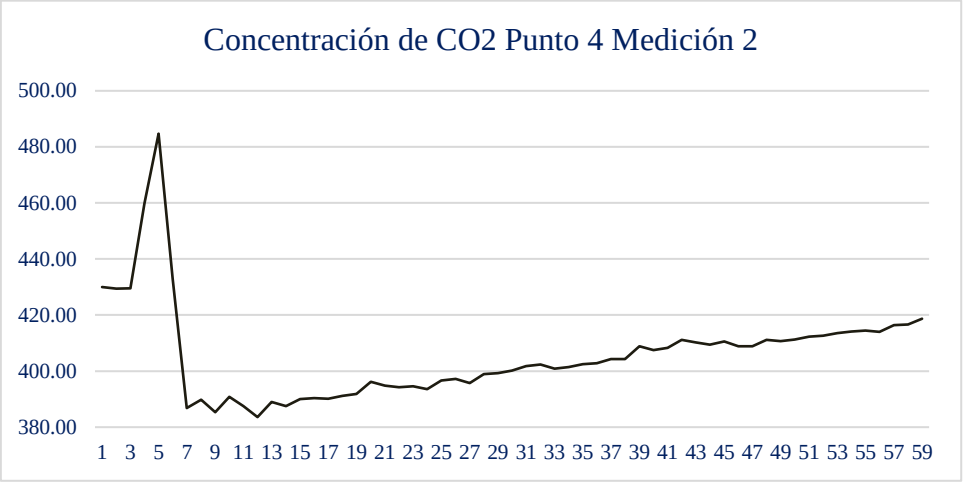


Ilustración 38 Ejemplo: Medición 2 para el punto 4

Entonces el usuario elegiría el intervalo 7 al 59, por lo que el programa lo que hará es buscar el siguiente intervalo pero de todo el archivo el cual vendría a corresponder de la medición 1:

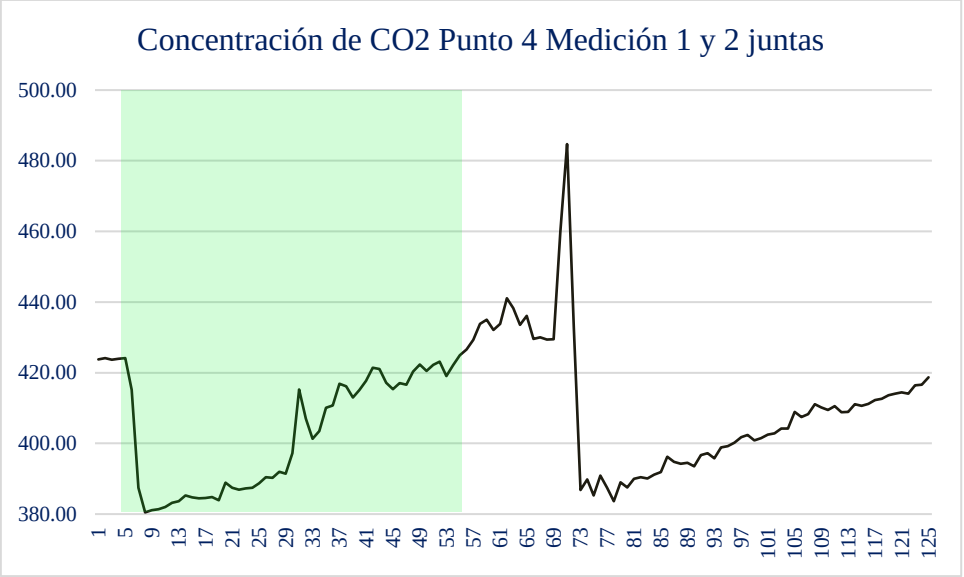


Ilustración 39 Ejemplo: Error en intervalo seleccionado

Este intervalo seleccionado según la Ilustración 39 no pertenece a la medición 2 de la Ilustración 26, por ello se vio la necesidad de separar los archivos como en la Ilustración 35.

Gracias a ese nuevo manejo de archivos ahora si se obtienen los datos exactos y por lo tanto un correcto cálculo del flujo el cual se explica con los siguientes métodos (Arriaza Carrillo & Aguilar Vega, 2022) los cuales no contienen actualizaciones en esta segunda versión del programa.

1. El método getLims recupera los límites de un rango introducidos por el usuario en dos campos de texto en la interfaz de la aplicación. Convierte estos valores a enteros y los almacena en un vector llamado limites. Si el usuario no ingresa valores numéricos, el método maneja la excepción y devuelve el vector con ambos límites establecidos en 0. Este método es esencial para obtener los extremos del rango de datos que se utilizarán en análisis subsiguientes dentro de la aplicación.
2. El método calcularRegresion se activa al hacer clic en un botón “Calcular” y realiza un análisis de regresión lineal. Primero, obtiene los límites para la selección de datos y verifica si son distintos. Luego, extrae segmentos específicos de datos de mediciones mediante llamadas a los métodos de manipulación de datos trimXData y trimYData. A continuación, calcula la pendiente y la correlación con esos datos y muestra los resultados en la interfaz de usuario. Además, genera un informe detallado, incluyendo la fecha, el punto de medición, la medida de flujo y el coeficiente de correlación (R^2), este es el archivo de datos de flujo en la carpeta “CO_2 Reportes”. Este proceso permite analizar la relación lineal entre variables y proporciona información detallada sobre la calidad del ajuste del modelo lineal a los datos seleccionados. Este método es invocado a los siguientes métodos que si realizan el cálculo con el segmento de la Ilustración 36. El nombre de este archivo incluye igualmente el número de medición en el punto.
3. El método calcPendiente calcula la pendiente de la regresión lineal entre dos conjuntos de datos, uno representado por la lista de enteros x y otro por la lista de flotantes y. Utiliza una serie de variables auxiliares, como term1, term2, term3, y term4, para realizar los cálculos intermedios. Contiene un bucle for que itera a través de los elementos de las listas, actualizando estas variables en cada iteración. Finalmente, calcula la pendiente utilizando las fórmulas de la regresión lineal y redondea el resultado a tres decimales antes de devolverlo.
4. El método calcCorrelacion calcula el coeficiente de correlación (r) entre dos conjuntos de datos, representados por las listas de enteros x y flotantes y. El método básicamente calcula el coeficiente de correlación utilizando las fórmulas correspondientes, proporcionando

información sobre la fuerza y dirección de la relación lineal entre los conjuntos de datos. Las impresiones en consola sirven para depurar y entender los cálculos intermedios.

En resumen, el cálculo del flujo está estrechamente vinculado con la gestión del almacenamiento de archivos. Un manejo incorrecto puede producir resultados erróneos por parte de los métodos de cálculo de flujo, al generar confusiones en la interpretación de la posición de los datos.

3.1.2.5 Envío de comandos de los botones por Bluetooth.

En secciones anteriores, se han proporcionado explicaciones introductorias de algunos métodos que controlan ciertos botones vinculados con acciones previamente detalladas, tales como el botón “Conectar”, “Calcular”, “Start”, “Buscar” y “Guardar”. En esta sección, se ofrecerá una descripción más detallada de la función de cada botón en la interfaz del usuario.

Además de ejecutar acciones dentro de la aplicación, los siguientes botones cumplen la función adicional de enviar comandos al microcontrolador mediante conexión Bluetooth, una característica ya implementada en el código original. En la siguiente tabla, se presenta un cuadro comparativo con la descripción de cada botón que contienen esta especialidad.

Tabla 15

Comparación de las funciones de los botones en ambas versiones del programa.

BOTÓN	**COMANDO ENVIADO POR BLUETOOTH**		**FUNCIÓN EN LA VERSIÓN 2 DE LA APLICACIÓN**
	V1	**V2**	
Actualizar GPS	P#	No envía	Muestra en los campos de texto la posición actual dada por el gps Neo 6M.
Reconectar Modem	-	M	Reinicia el SIM800L si se ha perdido la conexión GPRS.
Guardar	F#	F,<Número de punto ingresado>	Detiene la actualización de lecturas del GPS y envía el número de punto de medición al microcontrolador para que lo almacene temporalmente.

Start	G#	G	Enciende la Bomba de vacío para que circule gas al sensor LI 830. Le indica al programa que está listo para generar el gráfico de concentración de CO_2.
Stop	H#	H	Apaga la Bomba y le indica al programa que detenga la generación del gráfico de concentración de CO_2.
Enviar	-	E	Botón nuevo implementado que le indica al microcontrolador que envíe a la hoja de cálculo los datos que ha almacenado en el búfer temporal y reiniciar dicho búfer.
Reiniciar	-	R	Botón nuevo implementado que reinicia la interfaz y el gráfico para realizar una nueva medición, además le envía un comando al microcontrolador a que reinicie el búfer temporal que almacena los datos.
Desconectar	-	R	Este botón envía un comando al microcontrolador a que reinicie el búfer temporal que almacena los datos, antes de cerrar la aplicación y cerrar la conexión Bluetooth.

Los botones nuevos (Guardar, Enviar y Reiniciar) desempeñan funciones clave en la versión 2 de la aplicación, abordando la gestión de datos, la comunicación con el microcontrolador y la preparación para

nuevas mediciones. La introducción de estas funciones mejora la eficiencia y la usabilidad de la aplicación, permitiendo un mejor control y mayor versatilidad del sistema de medición de CO_2 volcánico.

Haciendo énfasis en el botón “Reiniciar” (IdBtnReiniciar) en la aplicación tiene la función de restablecer y limpiar los datos relacionados con la medición de CO_2 y el gráfico asociado. Cuando se presiona este botón, primero verifica si hay una conexión Bluetooth activa. Si la conexión está establecida, envía un comando Bluetooth específico (“R”) para indicar al microcontrolador que limpie el búfer de almacenamiento temporal de datos. Luego, el botón realiza varias operaciones de limpieza, como borrar las listas que almacenan datos temporales y crea un nuevo gráfico vacío. Además, restablece los ejes del gráfico y habilita el botón de “Actualizar GPS” el campo para ingresar el número de punto de medición con el botón “Guardar”. También muestra información relevante, como el punto de medición anterior, recuperado de preferencias compartidas. En general, el botón de reinicio asegura que el sistema esté listo para una nueva sesión de medición y presenta la interfaz de usuario en un estado inicial coherente.

```
IdBtnReiniciar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isConnected) {
            MyConexionBT.write("R");
        ListValCO2.clear();
        ListStrHora.clear();
        List ListValCO2_2 = new ArrayList();
        ListValCO2_2.clear();

        Line line = new Line(ListValCO2_2)
                .setColor(Color.parseColor("#2729b0"))
                .setCubic(true);

        List<Line> lines = new ArrayList<>();
        lines.add(line);

        LineChartData data = new LineChartData();
        data.setLines(lines);

        // Restablecer los ejes del gráfico
        Viewport viewport = new Viewport(lineChartView.getMaximumViewport());
        viewport.bottom = 0;
        viewport.top = 20000;
        viewport.left = 0;
        viewport.right = 5000;
        lineChartView.setCurrentViewport(viewport);
        lineChartView.setLineChartData(data);
        lineChartView.invalidate();

        // Restablecer el estado de los botones
        IdBtnGps.setEnabled(true);
```

```
        Id_BtnPointMeasure.setEnabled(true);
        idTxtPoint.setEnabled(true);
        SharedPreferences preferences = getSharedPreferences("puntos",
Context.MODE_PRIVATE);
        String previousPoint = preferences.getString("id", "");
        puntoAnterior.setText("Punto de Medicion anterior:"+previousPoint);
        } else {
            Toast.makeText(getBaseContext(), "No se ha establecido una conexión
Bluetooth", Toast.LENGTH_SHORT).show();
        }
    }
})
```

Fragmento 22 Método del botón Reiniciar.

Además se ha modificado el bloque de código sobre el método OnCreate que actualiza el campo de texto de los puntos de medición previos. Este bloque de código utiliza preferencias compartidas para recuperar el punto de medición almacenado previamente en la aplicación. Extrae el valor “id” del conjunto de preferencias llamado “puntos” y lo asigna a la variable “previousPoint”. Luego, visualiza este punto de medición anterior en un TextView denominado “puntoAnterior” en la interfaz de usuario. En esencia, proporciona información sobre el punto de medición utilizado en la sesión anterior de la aplicación.

```
SharedPreferences preferences = getSharedPreferences("puntos", Context.MODE_PRIVATE);
String previousPoint = preferences.getString("id", "");

TextView puntoAnterior = findViewById(R.id.puntoAnterior);
puntoAnterior.setText("Punto de Medición anterior: " + previousPoint);
```

Fragmento 23 Recuperación y Presentación del Punto de medición anterior desde Preferencias Compartidas

Este bloque es necesario porque le permite al usuario tener un orden sobre qué punto está por medir.

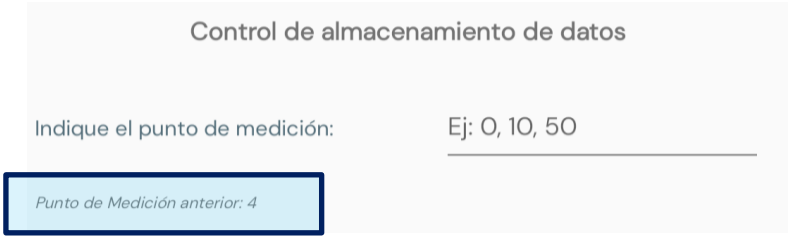


Ilustración 40 Presentación del número de punto anterior

Por otro lado ahora haciendo énfasis en botón “Guardar” se observa en la

Tabla 15 que envía por Bluetooth de la forma:

```
F,<Número de punto de medición ingresado>
```

Este comando funciona en conjunto con el Fragmento 7 del programa del microcontrolador, que separa el “F” que ejecuta una acción y el número que se guarda en una variable para luego ser utilizado. El método que controla este botón es el del Fragmento 19.

En cuanto al botón “Start” ahora en vez de abrir una nueva comunicación Bluetooth ahora solo maneja una bandera llamada “Graph_CSV” el cual indica cuando generar el gráfico y cuando detenerlo. Por lo tanto “Start” establece esta bandera como “verdadera” y “Stop” la establece como “falsa” esto se puede ver en el Fragmento 18 a la hora de generar el gráfico. Ambos botones como lo dice la

Tabla 15 controlan la bomba a través del microcontrolador.

En la versión 1 del código los comandos se enviaban seguidos de un “#” ahora en esta nueva versión esto no es necesario ya que se ha modificado la lógica con la que el microcontrolador recibe estos comandos (Fragmento 6,Fragmento 7).

3.2 Comparación de la aplicación de la versión 1 y 2.

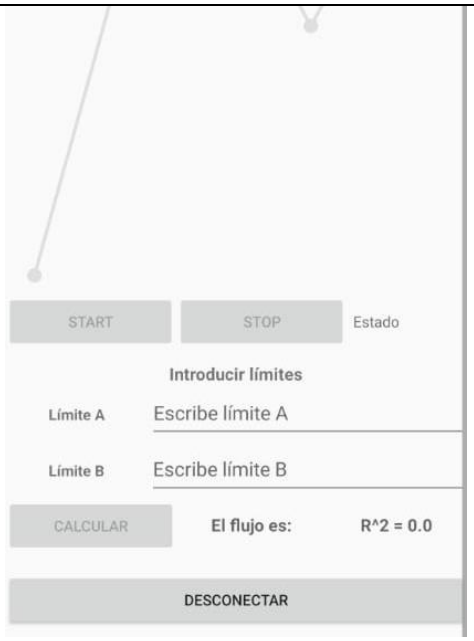
En la siguiente Tabla se muestra una comparación resumen del funcionamiento y características de la versión 1 y 2 de la aplicación.

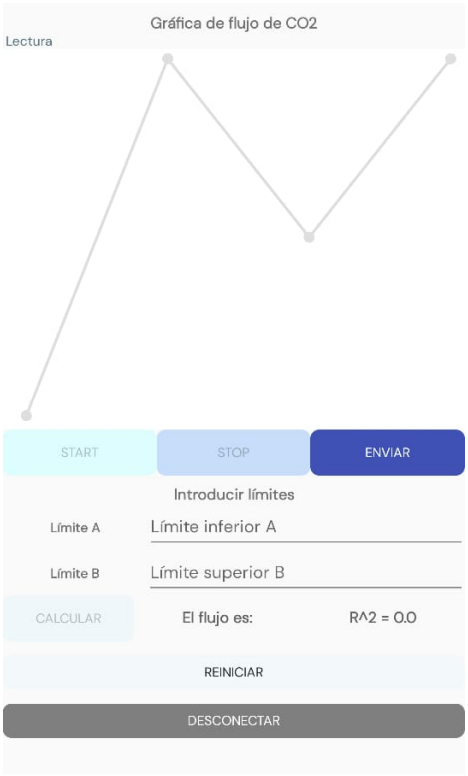
	VERSIÓN 1	VERSIÓN 2
Interfaz gráfica.		


Medidor CO2
BUSCAR
CONECTAR
Medición de CO2
Lectura
Geolocalizador
Temperatura y Humedad de la CPU
00.00
TEMPERATURA
ACTUALIZAR GEOLOCALIZADOR
Lectura
Lectura
Lectura
Lectura
00.00
HUMEDAD
Control de almacenamiento de datos
Indique el punto de medición:
Ej: 0, 10, 50
Punto anterior:
GUARDAR
Gráfica de flujo de CO2
Lectura


Medidor CO2
BUSCAR DISPOSITIVOS GUARDADOS
CONECTAR
Medición de CO2
Lectura
Geolocalizador
Lectura
Lectura
Lectura
Lectura
ACTUALIZAR GEOLOCALIZADOR
Temperatura y Humedad
TEMPERATURA
00.00
HUMEDAD
00.00


Procesos del CPU
GPRS:
GPS:
Almacenamiento:
Envio:
RECONECTAR MODEM
Control de almacenamiento de datos
Indique el punto de medición:
Ej: 0, 10, 50
Punto de Medición anterior: 4
GUARDAR





Manejo de permisos de la aplicación.	No se manejan excepciones si el usuario no otorga permisos de Bluetooth y almacenamiento a la aplicación.	Se manejan excepciones para que el usuario otorgue permisos de Bluetooth y almacenamiento a la aplicación mostrándole un mensaje.
Recepción de datos por Bluetooth.	Abre dos comunicaciones diferentes por Bluetooth, para la primera sección de muestra de datos y otra para la generación del gráfico de CO_2.	Abre una sola comunicación por Bluetooth y el flujo de datos es constante. La generación del gráfico se controla mediante una bandera y no cierra la adquisición de datos para la primer sección.
Gestión de almacenamiento de archivos	Guarda un archivo CSV de concentración de CO_2 por número de punto de medición al crear el gráfico, el cual muestra, fecha, hora y	Guarda un archivo CSV de concentración de CO_2 por número de medición en el punto al crear el gráfico, el cual muestra la fecha, hora,

	concentración en ppm respectivamente en su encabezado. Guarda un archivo CSV cada vez que se calcula el flujo de CO_2.	temperatura, humedad, concentración, número de punto y geolocalización. Guarda un archivo CSV cada vez que se calcula el flujo de CO_2, pero indicando el número de medición en el nombre.
Botones	La aplicación contiene los botones: ■ Buscar dispositivos ■ Conectar ■ Actualizar GPS ■ Guardar ■ Start ■ Stop ■ Calcular ■ Desconectar	La aplicación contiene los botones: ■ Buscar dispositivos ■ Conectar ■ Actualizar GPS ■ Reconectar Modem ■ Guardar ■ Start ■ Stop ■ Enviar ■ Calcular ■ Reiniciar ■ Desconectar

Tabla 16 Comparativa general de la versión 2 vs versión 2 de la aplicación.

La Tabla 17 resume las características principales de las versiones de la aplicación, el objetivo de esta mejora es una integración más eficiente con el equipo de medición y con las nuevas funcionalidades que tiene este y añadir más robustez proporcionando más validaciones y su utilización pueda ser más estable.

El siguiente diagrama muestra las actividades realizadas por la app en orden.

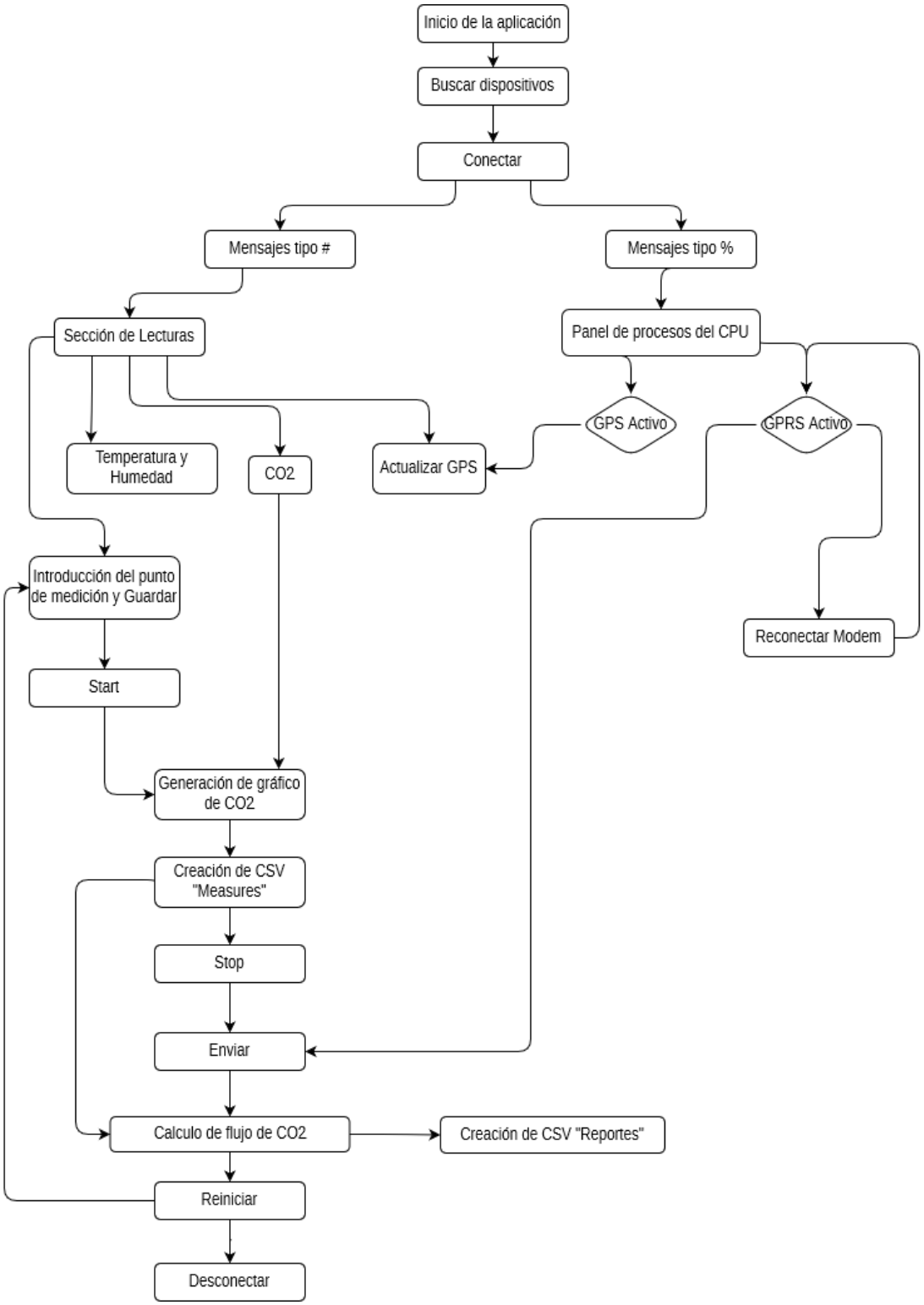


Ilustración 41 Flujo de las actividades y uso de la aplicación

Capítulo 4

4 Transmisión y visualización de datos.

Hasta este punto se ha venido realizando mejoras al sistema de medición que ya ha sido creado en (Arriaza Carrillo & Aguilar Vega, 2022) exceptuando en lo que respecta menciones al envío de datos tanto en los métodos y funciones del programa de la aplicación Android y el microcontrolador ESP32 que hacen referencia o realizan esta tarea de enviar datos a la hoja de cálculo de Google.

En este capítulo se desarrollará la metodología empleada para la transmisión de los datos, la presentación y manipulación de éstos para tener control del registro de estos de acuerdo a fechas y posición en el que fueron medidos. Para ello se divide en dos secciones principales, la transmisión de datos a Google Sheets y la visualización de éstos con Dash Plotly.

4.1 Transmisión de datos a Google Sheets.

En el capítulo 2 en el rediseño del software del programa del microcontrolador ESP32 se detalló la nueva tarea asignada encargada del envío de datos a la hoja de cálculo. El Fragmento 13 de la función enviar es la encargada de dicha tarea donde formatea los datos en un JSON y hace una solicitud HTTP POST al Appscript para que la este pueda procesarlos e introducirlos a la hoja de cálculo.

En la Ilustración 42 se observa un diagrama acerca de este proceso de envío, el diagrama describe el flujo de comunicación para la transmisión de datos desde un microcontrolador ESP32 (Cliente) a la implementación Appscript de Google Sheets (Servidor) a través de una solicitud HTTP POST. Inicia con el ESP32 activando la conexión GPRS mediante el módulo SIM800L. Una vez que la conexión GPRS está establecida, el ESP32 envía una solicitud HTTP POST, que contiene datos en formato JSON, al AppScript de Google Sheets, al recibir la solicitud, procesa los datos y los almacena en la hoja de cálculo de Google Sheets. Finalmente, el AppScript puede responder al ESP32 con un mensaje indicando el éxito o el error de la operación. Este flujo destaca la interacción entre el hardware del ESP32, el

módulo SIM800L, y el software del AppScript, mostrando cómo se transmiten y registran los datos de manera efectiva en la nube.

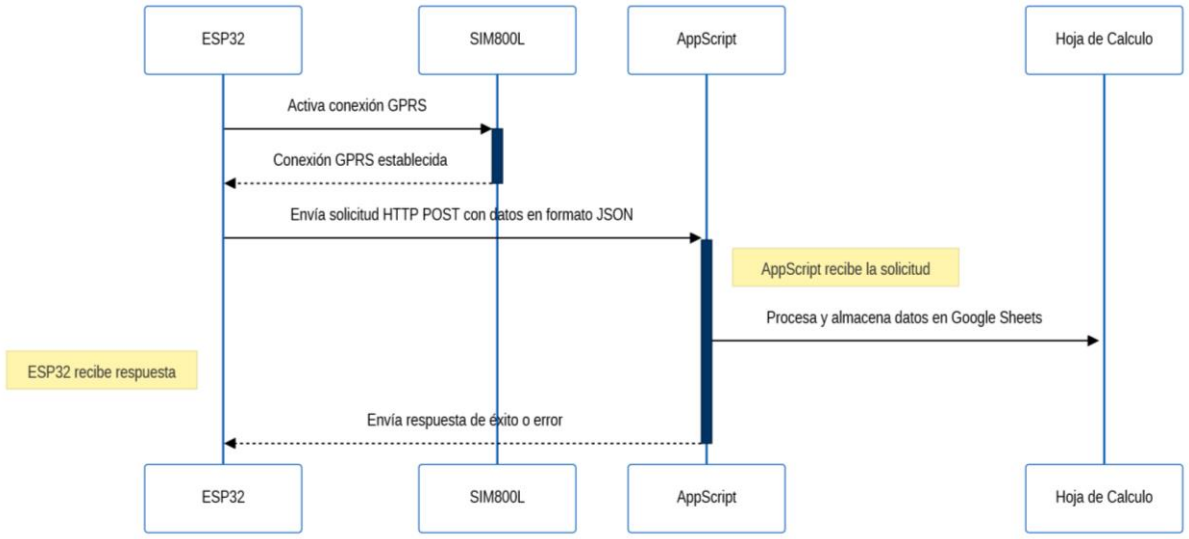


Ilustración 42 Proceso de Solicitud HTTP POST desde ESP32 a Google Sheets mediante AppScript

```
var SHEET_NAME = 'Hoja 1';
var SCRIPT_PROP = PropertiesService.getScriptProperties();
function doPost(e) {
  var lock = LockService.getScriptLock();
  lock.tryLock(20000);
  try {
    var doc = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = doc.getSheetByName(SHEET_NAME);
    var jsonString = e.postData.getDataAsString();
    var jsonData = JSON.parse(jsonString);
    var currentDate = Utilities.formatDate(new Date(), 'America/El_Salvador',
'MM/dd/yy');
    for (var i = 0; i < jsonData.length; i++) {
      var row = [];
      row.push(currentDate);
      row.push(jsonData[i].value1);
      row.push(jsonData[i].value2);
      row.push(jsonData[i].value3);
      row.push(jsonData[i].value4);
      row.push(jsonData[i].value5);
      row.push(jsonData[i].value6);
      row.push(jsonData[i].value7);
      sheet.appendRow(row);
    }
    return ContentService
          .createTextOutput(JSON.stringify({ 'result': 'success', 'row': row }))
          .setMimeType(ContentService.MimeType.JSON);
  } catch(e) {
    return ContentService
          .createTextOutput(JSON.stringify({ 'result': 'error', 'error': e }))
```

```
          .setMimeType(ContentService.MimeType.JSON);
  } finally {
    lock.releaseLock();
  }
}
```

Fragmento 24 Implementación AppScript

El Fragmento 24 es la implementación AppScript diseñado para manejar solicitudes HTTP POST, procesar datos recibidos en formato JSON, y almacenar estos datos en una hoja de cálculo de Google Sheets. La función principal del script se activa cuando recibe una solicitud POST. Inicialmente, implementa un mecanismo para prevenir la modificación concurrente de la hoja de cálculo. Luego, accede a la hoja de cálculo y a la hoja específica donde se almacenarán los datos.

El script procesa los datos recibidos, extrayendo y parseando la información en formato JSON. Para cada conjunto de datos recibidos, crea una nueva fila que incluye la fecha actual y los datos específicos de la solicitud. Estas filas se añaden sucesivamente a la hoja de cálculo, la manera en cómo se almacenan concuerda con la

Tabla 14 del archivo CSV que se almacena en la memoria interna del teléfono.

Finalmente, el script responde a la solicitud POST con un mensaje en formato JSON, indicando si la operación de almacenamiento de datos fue exitosa o si ocurrió un error. Este mensaje puede incluir detalles adicionales como la última fila de datos almacenada o información sobre el error ocurrido. El script también implementa un mecanismo para liberar recursos después de procesar la solicitud, asegurando que el sistema esté listo para futuras solicitudes. El AppScript (Ramírez, 2022) actúa como un intermediario eficiente entre las solicitudes de datos entrantes y su almacenamiento organizado en Google Sheets.

Existen varios métodos para almacenar datos en hojas de cálculo utilizando solicitudes GET (Tutorials, Random Nerd Tutorials, 2018) y Google Sheets API (Merati, 2023) o utilizar IFTTT (Pro, 2022) (Merati, 2023). Como se indicó en el capítulo 2, al principio de este proyecto se realizaban múltiples solicitudes HTTP GET para cada dato guardado en el ESP32 de tal manera que el Fragmento 24 contenía un script totalmente diferente, debido a los posibles problemas de conectividad por parte del SIM800L se modificó a hacer un solo envío de un paquete de datos de tal manera que ahora el script incorpora casi de golpe dicho paquete.

La manera en cómo se comporta este script puede compararse a un recepcionista de restaurante que recibe un grupo de personas (datos en formato JSON) , toma la lista de nombres y los registra

cuidadosamente (como el script procesa el JSON para extraer los datos individuales), a continuación el recepcionista asigna a cada grupo una mesa en orden (grupo de celdas en la hoja de cálculo). Además si son varios grupos de personas el recepcionista gestiona eficientemente su registro para evitar errores o duplicaciones (esto es similar al mecanismo de bloqueo del script, que asegura que los datos se procesen de manera ordenada y sin conflictos).

En este proyecto, el proceso de "parseo" implica analizar y transformar datos en formato JSON (JavaScript Object Notation) en una estructura que el programa pueda utilizar de manera efectiva. JSON es un formato de intercambio de datos ligero, diseñado para ser fácilmente comprensible por humanos, mientras que también es simple de analizar y generar por máquinas. En el contexto de la estación móvil para la medición del flujo de CO_2, se empleó JSON para organizar los datos recolectados, facilitando así su análisis y almacenamiento. Este proceso es vital para manejar eficientemente la gran cantidad de datos generados y para garantizar su correcta interpretación y uso en investigaciones. La analogía de un recepcionista que recibe una lista en un idioma o formato desconocido es útil aquí: el recepcionista necesita "traducir" esa lista a un formato que pueda entender y usar, organizando a cada persona y grupo en su respectiva mesa. Del mismo modo, el parseo de JSON transforma los datos en una forma que nuestro sistema puede procesar y utilizar efectivamente.

Para comprobar el funcionamiento del script fue necesario utilizar herramientas a parte de las pruebas realizadas por el ESP32[9] como por ejemplo Postman.

Ilustración 43 Herramienta Postman

Postman es una herramienta integral para el desarrollo y prueba de APIs que permite a los usuarios enviar solicitudes, recibir respuestas, organizar pruebas en colecciones, y automatizar pruebas para validar el comportamiento de las APIs. Su interfaz de usuario amigable, junto con la capacidad de integración y extensión, hace de Postman una herramienta esencial en el arsenal de desarrolladores

[9] La solicitud HTTP POST en una implementación del AppScript de una hoja de calculo usando correo institucional no tuvo efecto. Probablemente se requieran más permisos para ejecutarse.

modernos, especialmente útil en el desarrollo, prueba y mantenimiento de APIs en diversos entornos de trabajo.

Esta herramienta fue utilizada en este proyecto únicamente para simular solicitudes HTTP POST del ESP32 hacia la implementación AppScript en el cual se utilizaba un JSON ejemplo para comprobar la respuesta del script y si el funcionamiento de éste era correcto.

La imagen de la Ilustración 44 muestra la interfaz de Postman después de haber realizado una solicitud HTTP POST. En la solicitud, se envía el JSON de prueba a la URL del AppScript de Google Sheets, que se ejecuta con éxito, ya que la respuesta del servidor es un código 200 OK, lo cual indica que la solicitud ha sido procesada correctamente.

La respuesta del servidor contiene un objeto JSON que indica un “result”: “success”, junto con un “row” que parece ser un array con una sola entrada mostrando la fecha. Esto sugiere que el AppScript recibió el JSON, procesó la información y respondió con el resultado de la operación, que en este caso fue exitosa. La fecha en la respuesta es la última fila de datos procesada por el AppScript y agregada a la hoja de cálculo de Google Sheets. La interfaz de Postman muestra la respuesta en un formato legible bajo la pestaña “Pretty”, lo que facilita la interpretación de los datos devueltos por el AppScript.

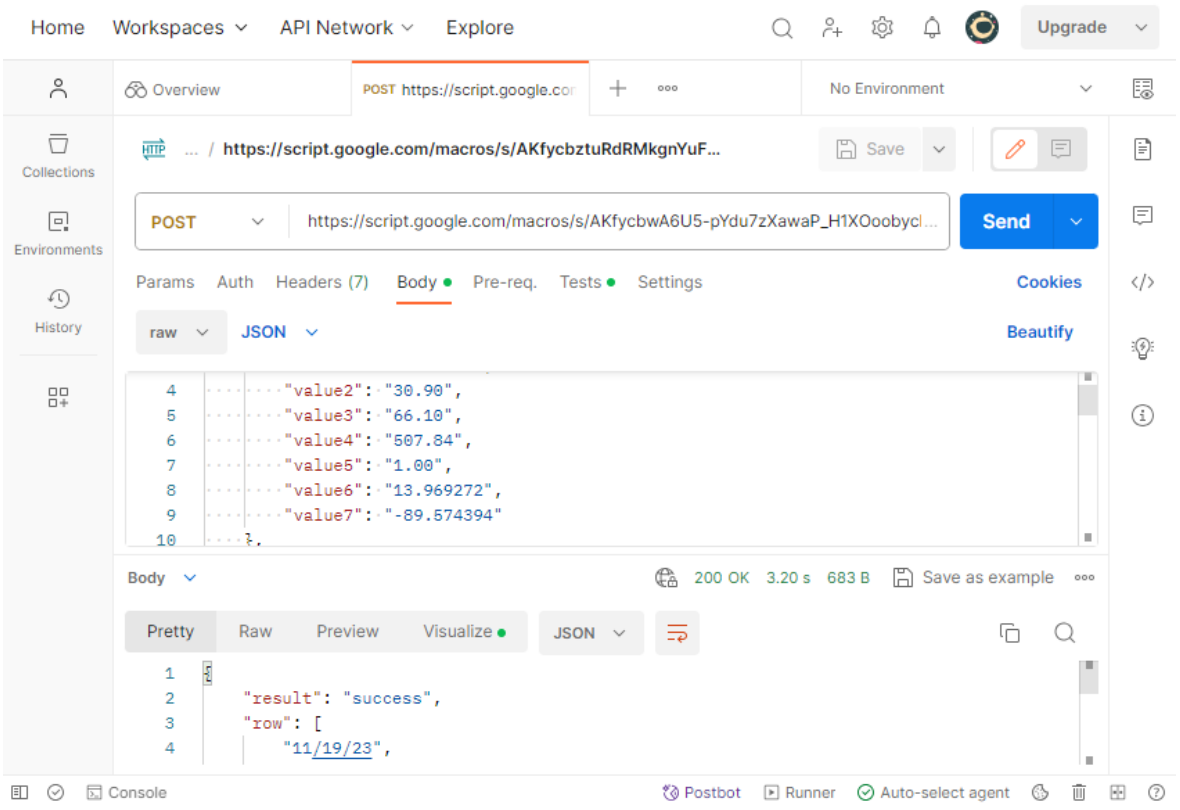
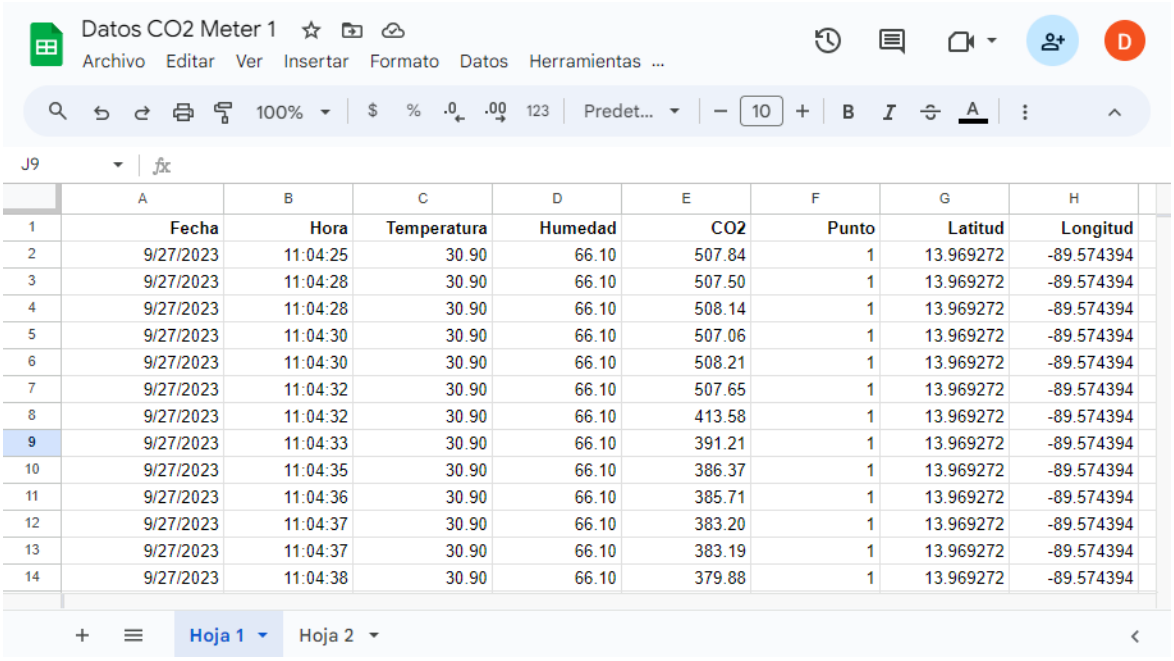


Ilustración 44 Solicitud HTTP POST desde Postman

Luego de realizar esta prueba se está completamente seguro que la ejecución del AppScript es correcta y ante cualquier falla en el proceso envío radicará en el funcionamiento del CPU ya sea por el programa del microcontrolador o la conectividad en el SIM800L.



	A	B	C	D	E	F	G	H
1	Fecha	Hora	Temperatura	Humedad	CO2	Punto	Latitud	Longitud
2	9/27/2023	11:04:25	30.90	66.10	507.84	1	13.969272	-89.574394
3	9/27/2023	11:04:28	30.90	66.10	507.50	1	13.969272	-89.574394
4	9/27/2023	11:04:28	30.90	66.10	508.14	1	13.969272	-89.574394
5	9/27/2023	11:04:30	30.90	66.10	507.06	1	13.969272	-89.574394
6	9/27/2023	11:04:30	30.90	66.10	508.21	1	13.969272	-89.574394
7	9/27/2023	11:04:32	30.90	66.10	507.65	1	13.969272	-89.574394
8	9/27/2023	11:04:32	30.90	66.10	413.58	1	13.969272	-89.574394
9	9/27/2023	11:04:33	30.90	66.10	391.21	1	13.969272	-89.574394
10	9/27/2023	11:04:35	30.90	66.10	386.37	1	13.969272	-89.574394
11	9/27/2023	11:04:36	30.90	66.10	385.71	1	13.969272	-89.574394
12	9/27/2023	11:04:37	30.90	66.10	383.20	1	13.969272	-89.574394
13	9/27/2023	11:04:37	30.90	66.10	383.19	1	13.969272	-89.574394
14	9/27/2023	11:04:38	30.90	66.10	379.88	1	13.969272	-89.574394

Ilustración 45 Registro de datos en la hoja de cálculo

4.2 Desarrollo de Dashboard para presentación de datos.

Para la presentación de los resultados de las mediciones se desarrolló una plataforma en el cual el usuario puede observar de manera interactiva todos los datos históricos que se han ido registrando en la hoja de cálculo y realizar regresiones lineales para obtener un perfil de flujo de CO_2 en la línea de medición, el lenguaje principal que se ha utilizado para este desarrollo es Python ya que es uno de los exponentes para análisis de datos.

4.2.1 Obtención de datos.

Para la obtención de datos de la hoja hacia un programa en Python es necesario hacer conexión con una Google Sheets API (TESEL, 2022) (Google, s.f.) y así poder leer y escribir en ella.

Para ello se necesita crear un proyecto en Google Cloud Plataform:

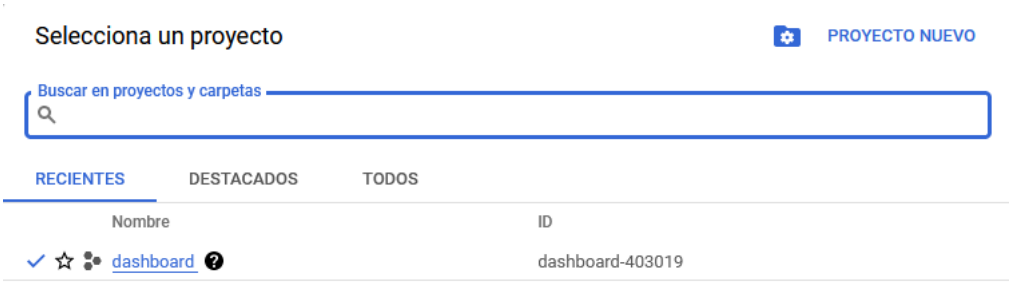


Ilustración 46 Proyecto nuevo en Google Cloud

Luego de crear el proyecto se procede a habilitar la API de Google Sheets en la biblioteca.

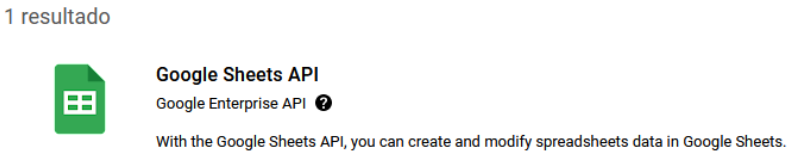


Ilustración 47 Google Sheets API

Para el proyecto escrito en Python es necesario crear una service account y credenciales para sí mismo necesariamente por razones de autenticación, seguridad, automatización de procesos y cumplimiento de las políticas de Google. La service account actúa como una entidad virtual para el programa, permitiéndole autenticarse con los servicios de Google, además el programa puede ejecutarse y realizar operaciones automáticamente sin la necesidad de la autenticación manual, las credenciales permiten definir y restringir qué operaciones puede realizar un script de Google Sheets para mantener la integridad y seguridad de los datos. Todo ello además de que Google exige estas medidas para cumplir con sus políticas de seguridad y uso de API.

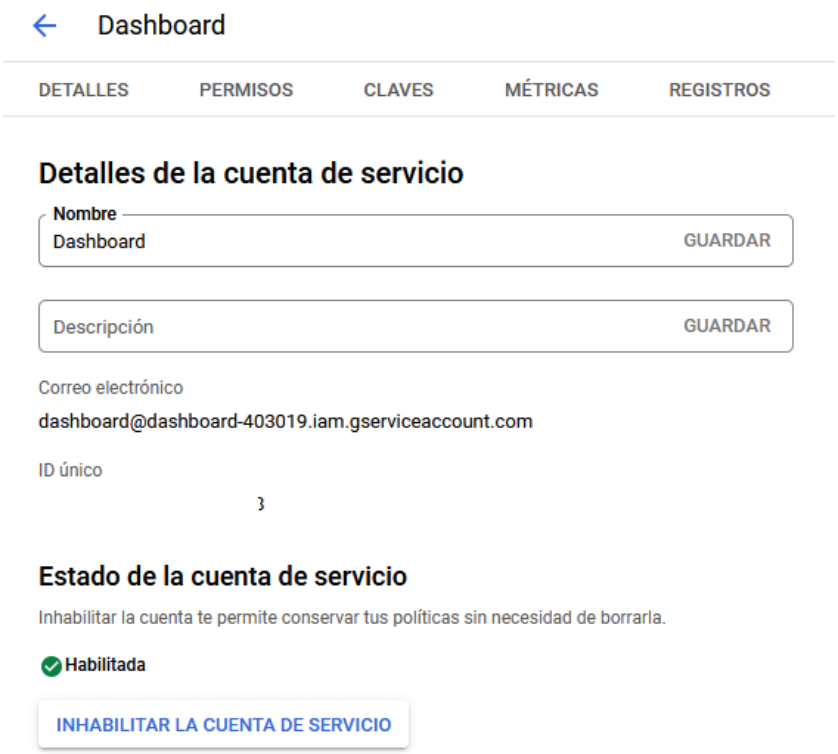


Ilustración 48 Cuenta de servicio creada

El siguiente paso es crear keys para el proyecto de Python, crear claves es necesario principalmente por razones de seguridad y acceso controlado. Las claves sirven como un mecanismo de autenticación para asegurarse de que solo las aplicaciones autorizadas puedan acceder y modificar las hojas de cálculo. Al usar claves, se puede garantizar que el acceso a los datos esté restringido y regulado, evitando así el uso no autorizado. Además, permite una automatización segura y eficiente de tareas relacionadas con el manejo de datos en Google Sheets. Al crear la clave Google Cloud entrega un key en formato JSON que se utilizará en el programa.

Finalmente es necesario compartirle la hoja de cálculo a la service account como editor.

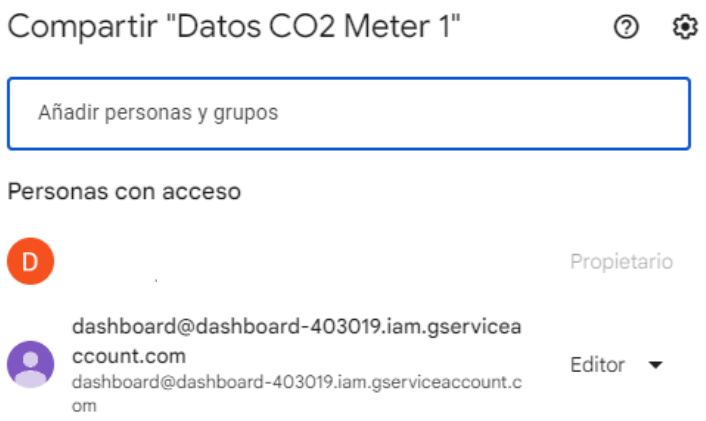


Ilustración 49 Compartir de la Hoja de Cálculo a la Cuenta de servicio creada.

Ahora en el programa escrito en Python se construye la parte encargada de recibir los datos después de hacer todo el procedimiento anterior y así poder establecer conexión con la hoja de cálculo mediante la API.

```
#Importacion de librerias para la conexion con la API de Google Sheets
import os
import base64
import json

# Librerias para el consumo de la Google Sheets
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from google.oauth2 import service_account

#-------------------------------------------------------------------------------
""" SCOPES = ['https://www.googleapis.com/auth/spreadsheets']
KEY = 'key.json'
SPREADSHEETS_ID = '1AZIGMqBmzAqVZzGbZDQdKUK_R9xVk7iuPrZ-0es5Bi4'

creds = None
creds = service_account.Credentials.from_service_account_file(KEY, scopes=SCOPES)

service = build('sheets', 'v4', credentials=creds)
sheets = service.spreadsheets() """

# Constantes
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']
SPREADSHEETS_ID = '1AZIGMqBmzAqVZzGbZDQdKUK_R9xVk7iuPrZ-0es5Bi4'

# Decodificar las credenciales codificadas en base64 de la variable de entorno
creds_json = base64.b64decode(os.environ['GOOGLE_SHEETS_CREDS_BASE64']).decode('utf-
8')
creds_dict = json.loads(creds_json)
```

```
# Carga las credenciales desde el diccionario
creds = service_account.Credentials.from_service_account_info(creds_dict,
scopes=SCOPES)

# Construye el servicio utilizando las credenciales
service = build('sheets', 'v4', credentials=creds)
sheets = service.spreadsheets()
```

Fragmento 25 Script para obtener los datos de la hoja de cálculo

El Fragmento anterior se encarga de establecer la conexión con la API para poder leer y escribir datos en la hoja de cálculo (posteriormente una función utiliza la escritura en la hoja de cálculo). Y se divide en 3 partes principales.

1. **Importación de Librerías:**
   - La librería os se utiliza para interactuar con las funcionalidades del sistema operativo, especialmente para acceder a variables de entorno que contienen información sensible.
   - La librería base64 es empleada para codificar y decodificar datos según el esquema de codificación Base64, que permite representar datos binarios en cadenas de texto ASCII.
   - json se utiliza para manejar y parsear datos estructurados en formato JSON.
2. **Configuración y Autenticación para el Acceso a Google Sheets:**
   - Se definen constantes para la autenticación: SCOPES especifica el nivel de acceso requerido por la aplicación, en este caso, acceso a las hojas de cálculo de Google. SPREADSHEETS_ID es el identificador único de la hoja de cálculo con la que se interactuará.
   - Las credenciales para acceder a la API de Google Sheets se almacenan de manera segura utilizando variables de entorno. Se emplea la codificación en Base64 (GOOGLE_SHEETS_CREDS_BASE64) para proteger estas credenciales. La cadena codificada se decodifica y convierte en un objeto JSON, que se utiliza para generar un diccionario de credenciales.
3. **Establecimiento de la Conexión con el Servicio de Google Sheets:**
   - Las credenciales obtenidas se utilizan para configurar y autorizar el acceso al servicio de Google Sheets. Esto se realiza mediante la función "service_account.Credentials.from_service_account_info", proporcionándole el diccionario de credenciales y el alcance del acceso definido en SCOPES.
   - Con las credenciales autenticadas, se construye el servicio sheets utilizando la función build de la librería "googleapiclient.discovery". Este servicio facilita la interacción con la API de Google Sheets, permitiendo leer y manipular datos en la hoja de cálculo especificada.

Finalmente se utiliza el siguiente bloque de código para recolectar en forma de dataframe los datos que están en la hoja de cálculo. El programa lee dos hojas por separado.

```
#Obtencion de Datos de la Hoja 1-----------------------------------------------------
result = sheets.values().get(spreadsheetId=SPREADSHEETS_ID, range='Hoja
1!A:H').execute()
values = result.get('values', [])
df = pd.DataFrame(values[1:], columns=values[0])
df['Fecha y Hora'] = pd.to_datetime(df['Fecha'] + ' ' + df['Hora'])
df[['Temperatura', 'Humedad', 'CO2','Punto', 'Latitud', 'Longitud']] =
df[['Temperatura', 'Humedad', 'CO2','Punto', 'Latitud', 'Longitud']].astype(float)
dias_unicos = df['Fecha'].unique()
data_count_df = len(df)

#Obtencion de Datos de la Hoja 2-----------------------------------------------------
result2 = sheets.values().get(spreadsheetId=SPREADSHEETS_ID, range='Hoja
2!A:E').execute()
values2 = result2.get('values', [])
start_month = None
end_month = None
if values2:
    expected_columns = ['Fecha Mes', 'Pendientes', 'Punto Medicion', 'Latitud',
'Longitud']
# Verificar si la primera fila de values2 coincide con los nombres de columna
esperados
    if values2[0] == expected_columns:
        df2 = pd.DataFrame(values2[1:], columns=values2[0])
        df2['Fecha Mes'] = pd.to_datetime(df2['Fecha Mes'])
    else:
        df2 = pd.DataFrame(columns=expected_columns)
else:
    # Manejar el caso en que no se encuentren datos en la hoja 2
    df2 = pd.DataFrame(columns=['Fecha Mes', 'Pendientes', 'Punto Medicion',
'Latitud', 'Longitud'])
data_count_df2 = len(df2)
fechas_unicas = df2['Fecha Mes'].dt.strftime('%m/%d/%Y').drop_duplicates().values
```

Fragmento 26 Dataframes utilizados en el programa para el filtrado y la creación de gráficos

Obsérvese que se declaran tomando referencia los títulos de las columnas de la hoja de cálculo que contienen dichos datos (véase Ilustración 45). [10]Luego se realizan diferentes validaciones en caso que por un insólito error se desordenen los datos en la hoja.

[10] Es preferible que el usuario no manipule el formato de los datos en las hojas de cálculo, dado que el programa está diseñado para capturarlos en una manera específica.

4.2.2 Introducción a Dash Plotly.

Ilustración 50 Logo de Plotly (Plotly, 2023)

Plotly es una biblioteca de código abierto desarrollado por la empresa Plotly (Plotly, 2023) centrada en la visualización de datos para la inteligencia empresarial. La biblioteca es reconocida por su capacidad de crear más de 40 tipos de gráficos interactivos de alta calidad y generar paneles complejos con características interactivas con actualizaciones en tiempo real, está disponible tanto para Python como para R y JavaScript.

A comparación de Matplotlib, librería reconocida por su capacidad de ofrecer un control muy detallado sobre casi todos los aspectos de un gráfica, Plotly añade interactividad, facilidad de uso en la web ya que está diseñado con este enfoque al interactuar con Dash, personalización y brinda exportación de los gráficos en diferentes formatos de imagen.

Los usos comunes de Plotly popularmente se utiliza en ciencia de datos y análisis de datos, informes científicos académicos, e ingeniería y análisis financiero. En este proyecto se utilizará para crear distintos gráficos lineales tomando los datos de la hoja de cálculo Ilustración 45 manejados con Pandas como Dataframes[11] (DataScientest, 2022).

Ilustración 51 Logo de Dash (Trung, 2022)

[11] Estructura bidimensional y tabular de Python, similar a una hoja de cálculo. Es uns Serie Pandas indexada por valor.

Dash es un marco de trabajo de código abierto desarrollado por Plotly para la creación de páginas web analíticas utilizando Python y visualizaciones interactivas. Se basa en tecnologías web estándar (HTML, CSS, JavaScript).

Dash utiliza una sintaxis simple para crear componentes de interfaz de usuario. Estos componentes se generan en Python, y Dash los convierte en HTML y JavaScript (Plotly, 2023). Además contiene la función de "Callbacks" que actualiza de manera interactiva una parte de la aplicación web, generalmente en respuesta a una acción del usuario (Plotly, Tecnologies, s.f), en el programa desarrollado para la visualización normalmente se utiliza esta característica a la actualización de los gráficos de acuerdo a filtros, botones o entradas de texto por parte del usuario.

4.2.3 Estructura de la aplicación web.

El "app layout" (disposición de la aplicación) se refiere a la estructura y diseño de la interfaz de usuario de una aplicación web. Es fundamentalmente el esqueleto o el marco sobre el cual se construye la presentación visual y la interactividad.

4.2.3.1 Layout de la aplicación.

4.2.3.1.1 Definición del Layout.

El layout de una aplicación Dash se define como un árbol de componentes, que incluye elementos HTML y componentes específicos de Dash. Se utiliza para estructurar cómo se muestra el contenido en la página web, como textos, gráficos, botones, menús desplegables, etc.

4.2.3.1.2 Componentes HTML y Dash.

Dash proporciona clases para todos los elementos HTML estándar (como html.Div, html.Button, html.H1), así como componentes complejos y especializados de Dash (como dcc.Graph, dcc.Dropdown).

4.2.3.1.3 Estilo y Apariencia:

Se pueden aplicar estilos CSS a los componentes del layout para personalizar su apariencia. Esto incluye ajustes de colores, tamaños, márgenes, posicionamiento y más. El estilo se puede aplicar directamente a través del argumento "style" en cada componente o utilizando hojas de estilo CSS

externas en este caso la aplicación utiliza una llamada a una carpeta “assets” que contiene los estilos en CSS.

4.2.3.1.4 Interactividad y Actualizaciones Dinámicas:

Aunque el layout define la estructura inicial de la aplicación, los componentes dentro del layout pueden actualizarse dinámicamente mediante callbacks. Por ejemplo, el contenido del dcc.Graph (gráfico de CO2) puede cambiar en respuesta a la interacción del usuario con el filtro (dcc.Dropdown).

En el siguiente fragmento se muestra la estructura del app.layout de la aplicación web.

```
app.layout = html.Div(children=[
    html.Link(
        rel='stylesheet',
        href='https://fonts.googleapis.com/css2?..."'),
    html.Div(id='header', children=[
        html.H1('Mediciones de CO2', id='title'),
    ]),
    html.Div(id='content', style={'display': 'flex'}, children=[
        html.Div(id='main-content', children=[
            html.P('Descripción...'),

            html.Div(id='indicacion', children=[
                html.Div(id='description', children=[
                    html.P('Instrucción 1'),
                    html.P('Instrucción 2'),
                ]),
                html.Div(id='filter-container', children=[
                    dcc.Dropdown(id='filtro-dia', options=[], value=None),
                    dcc.Dropdown(id='filtro-punto', options=[], value=None),
                ]),
            ]),
            html.Div(id='counts', children=[
                html.P(id='hoja1'),
                html.P(id='hoja2'),
            ]),
            html.P('Texto sobre gráficos...'),

            dcc.Graph(id='mapa-ubicacion-actual'),

            html.Div(id='graph-row', children=[
                dcc.Graph(id='co2-hora-graph'),
                dcc.Graph(id='temperatura-humedad-graph'),
            ]),
            html.Div(id='regression-controls', children=[
                html.P('Texto sobre regresión lineal...'),
                html.Div(id='regression-inputs', children=[
                    dcc.Input(id='input-inicio-hora', type='text'),
                    dcc.Input(id='input-fin-hora', type='text'),
                    html.Button('Aplicar Regresión Lineal', id='btn-regresion'),
                ]),
```

```
            ]),
            html.Div(id='table-and-graph-container', children=[
                html.Div(id='table-container', children=[
                    dash_table.DataTable(id='tabla-co2', columns=[]),
                ]),
                html.Div(id='graph-container', children=[
                    dcc.Graph(id='regresion-lineal-graph'),
                ]),
            ]),
            html.Div(id='monthly-selector', children=[
                html.P('Texto sobre selección mensual...'),
                dcc.Store(id='df2-store'),
                dcc.Dropdown(id='fecha-selector', options=[], multi=True),
                dcc.Graph(id='grafico-co2-vs-distancia'),
            ]),
            html.Div(id='update-button-container', children=[
                html.Button('Actualizar datos', id='btn-actualizar-2'),
            ]),
        ]),
    ]),
])
```

Fragmento 27 Estructura general de la aplicación web.

Traduciendo el Fragmento 26 de manera gráfica este tendría que verse de la siguiente manera:

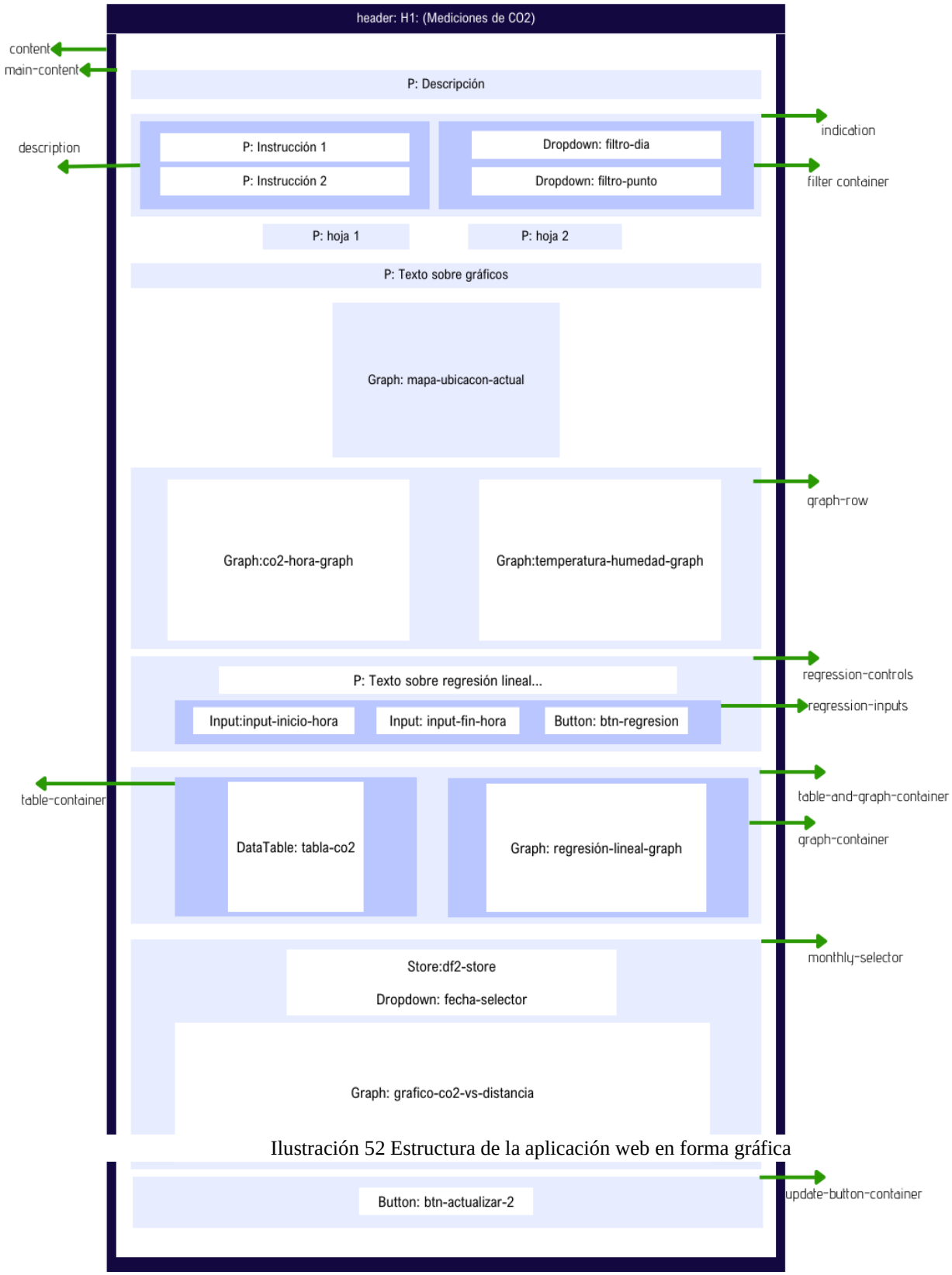


Ilustración 52 Estructura de la aplicación web en forma gráfica

La Ilustración 52 es una buena referencia hacia una estructura HTML, esto es gracias al componente HTML incorporado en Dash. Este tipo de representación visual es una forma excelente de comprender la arquitectura de la interfaz de usuario de una aplicación web desarrollada con Dash. Cada bloque en el diagrama representa un componente de la interfaz de usuario, como un contenedor (html.Div), un elemento de texto (html.P), un gráfico (dcc.Graph), un campo de entrada (dcc.Input), o un botón (html.Button) y elementos especiales (dcc.Dropdown) que son filtros desplegables.

Los bloques están organizados jerárquicamente, lo que refleja la estructura anidada del layout. Los componentes que se encuentran a nivel superior, como el encabezado y el contenedor principal, encapsulan a los componentes secundarios, lo que indica la relación de contenedor a contenido entre los elementos.

4.2.3.2 Gráficos con Plotly.

En el programa se definen diferentes funciones las cuales generalmente necesitan como argumento un dataframe filtrado, el cual es dado de acuerdo al usuario cuando selecciona una fecha de los datos que desea ver, y el punto de medición en el que se han tomado dichos datos en esa fecha. Por ejemplo la siguiente función (Fragmento 27) .

```
def create_co2_hora_graph(df_filtrado,nombre_punto):
    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=df_filtrado['Fecha y Hora'],
        y=df_filtrado['CO2'],
        name=f'CO2 - Punto: {nombre_punto}',line=dict(color='black', width=1.5))
        )
    fig.update_layout(
        title=f'CO2 vs Hora - Punto: {nombre_punto}',
        font=dict(family='Manrope', size=10,color='black'),plot_bgcolor='white',
        xaxis=dict(title='Fecha y Hora', showgrid=True),
        yaxis=dict(title='CO2', showgrid=True),
        margin=dict(l=10, r=10, t=50, b=25),)
    fig.update_xaxes(gridcolor='lightgrey', linewidth=1)
    fig.update_yaxes(gridcolor='lightgrey', linewidth=1)
    return fig
```

Fragmento 28 Función de gráfico lineal de concentración de CO_2 en un punto.

Esta función recrea un gráfico de todos los valores de concentración de CO_2 vs tiempo en punto de medición, el gráfico puede mostrar varias mediciones y el usuario debe identificar el patrón lineal en cada una de ellas, una forma fácil de identificar esto es observar la hora en que aparecen los datos en el eje x. La función muestra la configuración visual al usuario y crea un nuevo objeto de figura (go.Figure) de Plotly que servirá como contenedor para el gráfico.

Note que básicamente la función se subdivide en dos secciones para crear el ploteo, la adición de la serie de datos y la actualización de ejes. De igual manera que el Fragmento 28, para crear el gráfico histórico de temperatura y humedad se utiliza la misma técnica, la diferencia es que se toman dos Dataframes (temperatura y humedad) del dataframe filtrado.

Resumiendo, todas las funciones que definen gráficos para crearlos tienen una estructura parecida, la diferencia radica en los Dataframes que son utilizados. Que se actualizan dinámicamente de acuerdo a los filtros coloca el usuario, esto se verá más detalladamente en la siguiente subsección (4.2.4 Interactividad y Callbacks).

4.2.4 Interactividad y Callbacks

Los Callbacks son funciones que automáticamente actualizan partes de la aplicación en respuesta a entradas del usuario, como clics de botones o selecciones en un menú desplegable.

En la aplicación web se realizan 5 funciones Callbacks principales, cada uno utiliza el decorador[12] "@app.callback" para vincular las entradas del usuario con las actualizaciones de los componentes de la interfaz de usuario. Los "Input" y "State" son los desencadenantes y los valores que se pasan a las funciones del callback, mientras que los "Output" son los componentes que se actualizan como resultado de esta. La lógica dentro de cada función determina exactamente cómo se deben actualizar los componentes en respuesta a las acciones del usuario.

4.2.4.1 Botón actualizar datos y filtro de fecha.

```
@app.callback(
    Output('filtro-dia', 'options'),
    Output('filtro-dia', 'value'),
    Input('btn-actualizar-2', 'n_clicks')
)
```

Fragmento 29 Callbacks para filtro día

Este callback responde a los clics en un botón de actualización para modificar las opciones disponibles en un dropdown de filtro de día. La función que interactúa con este callback es para actualizar los Dataframes de la hoja 1 (actualizar_datos), es decir cuando se han agregado más datos a la hoja de cálculo. La función realiza el mismo proceso del Fragmento 26.

[12] En Python es una función especial que se utiliza para modificar el comportamiento de otra función.

Ilustración 53 Esquema del click al botón actualizar

4.2.4.2 Callback para Actualizar Gráficos Basados en Filtros Seleccionados.

```
@app.callback(
    [Output('hoja1', 'children'),
     Output('temperatura-humedad-graph', 'figure'),
     Output('mapa-ubicacion-actual', 'figure'),
     Output('co2-hora-graph', 'figure')],
    [Input('filtro-dia', 'value'),
     Input('filtro-punto', 'value')]
)
```

Fragmento 30 Callback para actualizar gráficos

Actualiza los contenidos del texto y tres gráficos en función de la fecha y el punto de medición seleccionados por el usuario. La función que acompaña a este callbacks es la siguiente:

```
def update_graphs(dia_seleccionado, punto_seleccionado):
    if dia_seleccionado:
        df_filtrado = df[df['Fecha'] == dia_seleccionado]
        texto_marcadores = [f'Punto: {punto}' for punto in df_filtrado['Punto']]
-------------------------------------------------------------------------
# Configuración del contenedor para el mapa…
        mapa = go.Figure(……)
-------------------------------------------------------------------------
        if punto_seleccionado:
            df_filtrado = df_filtrado[df_filtrado['Punto'] == punto_seleccionado]

        nombre_punto = punto_seleccionado if punto_seleccionado else 'Todos los
Puntos'
        temperatura_humedad_graph = create_temperatura_humedad_graph(df_filtrado,
nombre_punto)
        co2_hora_graph = create_co2_hora_graph(df_filtrado, nombre_punto)

    else:
        temperatura_humedad_graph = go.Figure()
        mapa = go.Figure()
        co2_hora_graph = go.Figure()
        nombre_punto = 'Todos los Puntos'

    return f"Lecturas de Sensores\n {data_count_df}",temperatura_humedad_graph,
mapa,co2_hora_graph
```

Fragmento 31 Función de actualización de gráficos

La función update_graphs se ejecutará automáticamente cuando los valores de los Inputs cambien, es decir, cuando el usuario seleccione una nueva fecha o un nuevo punto de medición, observece como recibe estos valores como argumentos.

Primero verifica si se ha seleccionado una fecha, si es así, filtra el DataFrame df para obtener solo los datos correspondientes a esa fecha, y si es relevante, también al punto de medición seleccionado. Finalmente se devuelven cuatro valores correspondientes a los cuatro Outputs. Estos son los nuevos valores o figuras que se mostrarán en los componentes especificados en la interfaz de usuario de la aplicación Dash.

4.2.4.3 Actualizar Opciones del Filtro de Punto.

```
@app.callback(
    Output('filtro-punto', 'options'),
    Input('filtro-dia', 'value')
def update_punto_options(dia_seleccionado):
    if dia_seleccionado:
        puntos_disponibles = df[df['Fecha'] == dia_seleccionado]['Punto'].unique()
        opciones = [{'label': punto, 'value': punto} for punto in puntos_disponibles]
        return opciones
    else:
        return []
```

Fragmento 32 Callback y función de actualización de filtro de punto de medición.

Este callback es responsable de actualizar dinámicamente las opciones disponibles en un menú desplegable (Dropdown) para el filtro de puntos de medición en la aplicación Dash.

Primero, se verifica si se ha seleccionado una fecha (dia_seleccionado). Si no hay una fecha seleccionada, la función podría retornar una lista vacía. Si hay una fecha seleccionada en el Dropdown "filtro-dia", se utiliza para filtrar el DataFrame global df para obtener solo los datos correspondientes a esa fecha y extraen los puntos de medición únicos disponibles para esa fecha.

El filtrado se realiza mediante indexación booleana[13] (Pandas development team, 2023). Usa la serie de booleanos para seleccionar solo aquellas filas del Dataframe df donde la condición sea verdadera es decir, donde la fecha en la columna 'Fecha' coincide con dia_seleccionado.

[13] Este método se realiza en todos los casos en los que se necesite filtrar el df global y df_filtrado.

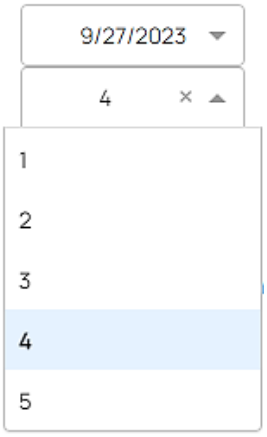


Ilustración 54 Filtro de punto de medición de acuerdo al filtro de fecha

En esta Ilustración se ve el resultado de este procedimiento, se ha selecciona la fecha 9/27/2023 y en la cual se han encontrado 5 puntos de medición.

4.2.4.4 Callback para aplicar regresión lineal y actualizar componentes relacionados.

```
@app.callback(
    Output('hoja2', 'children'),
    Output('regresion-lineal-graph', 'figure'),
    Output('tabla-co2', 'data'),
    Output('fecha-selector', 'options'),
    Input('btn-regresion', 'n_clicks'),
    State('filtro-dia', 'value'),
    State('input-inicio-hora', 'value'),
    State('input-fin-hora', 'value'),
    State('filtro-punto', 'value')
)
```

Fragmento 33 Callback para aplicar regresión lineal

Este decorador Callback se asocia a la función de regresión lineal la cual recibe como argumentos el click del botón “Aplicar Regresión Lineal” el día seleccionado del filtro fecha, el punto seleccionado del filtro de punto de medición, input en el usuario debe ingresar un intervalo donde la gráfica de CO_2 vs tiempo tenga un comportamiento lineal tal como se hace en la APK[14].

La función se encarga de filtrar el df global, para tomar los correspondientes, representarlos en una tabla y aplicar una regresión lineal mostrando en un gráfico la línea con su ecuación de tendencia del intervalo ingresado (esta función se verá más detalladamente la subsección 4.2.5 Análisis de Datos.).

[14] En este caso la gráfica se mostrará como la Ilustración 38 ya que el gráfico puede contener varias mediciones.

En el siguiente esquema se puede apreciar de mejor manera el funcionamiento descrito.

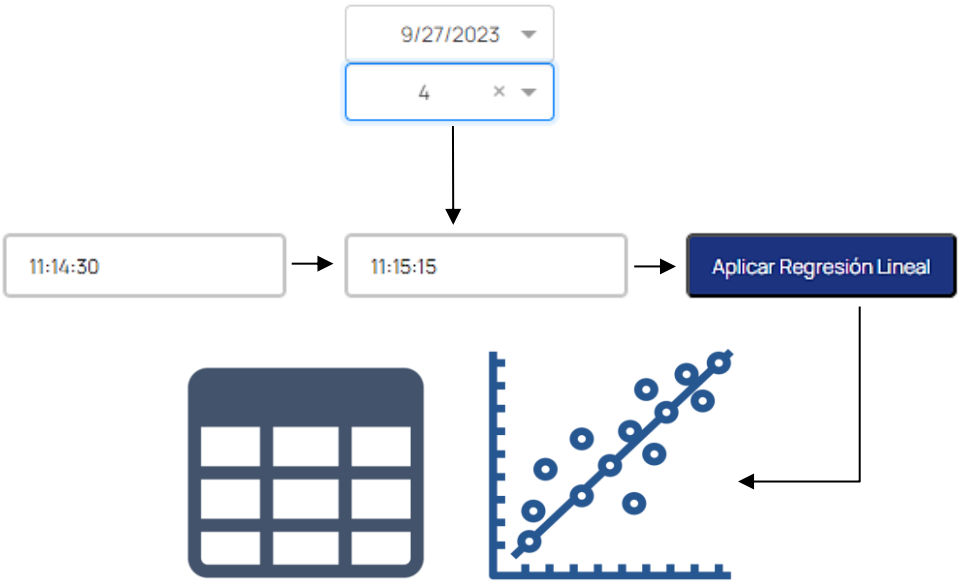


Ilustración 55 Esquema de función callback para regresión lineal.

4.2.4.5 Actualización de gráfico de flujo de CO_2 vs distancia.

```
@app.callback(
    Output('grafico-co2-vs-distancia', 'figure'),
    Input('fecha-selector', 'value')
)
```

Fragmento 34 Decorador callback para actualizar el gráfico de flujo de CO_2 vs distancia.

El resultado de este decorador es que, los usuarios pueden interactuar con el componente "fecha-selector" (este Dropdown, lee las fechas en que se han registrado valores de flujo de CO_2 en la hoja 2), y el gráfico "grafico-co2-vs-distancia" reflejará las selecciones de fecha con una representación visual actualizada de los datos correspondientes a estas.

El procesamiento que realiza la función asociada al callback toma el df2 global de leido de la hoja 2 y selecciona las filas que contengan la fecha seleccionada en la columna "Fecha mes" filtrando y preparando un nuevo Dataframe para ser utilizado en la función que creará el gráfico de flujo de CO_2 vs distancia.

El siguiente esquema ilustra este procedimiento.

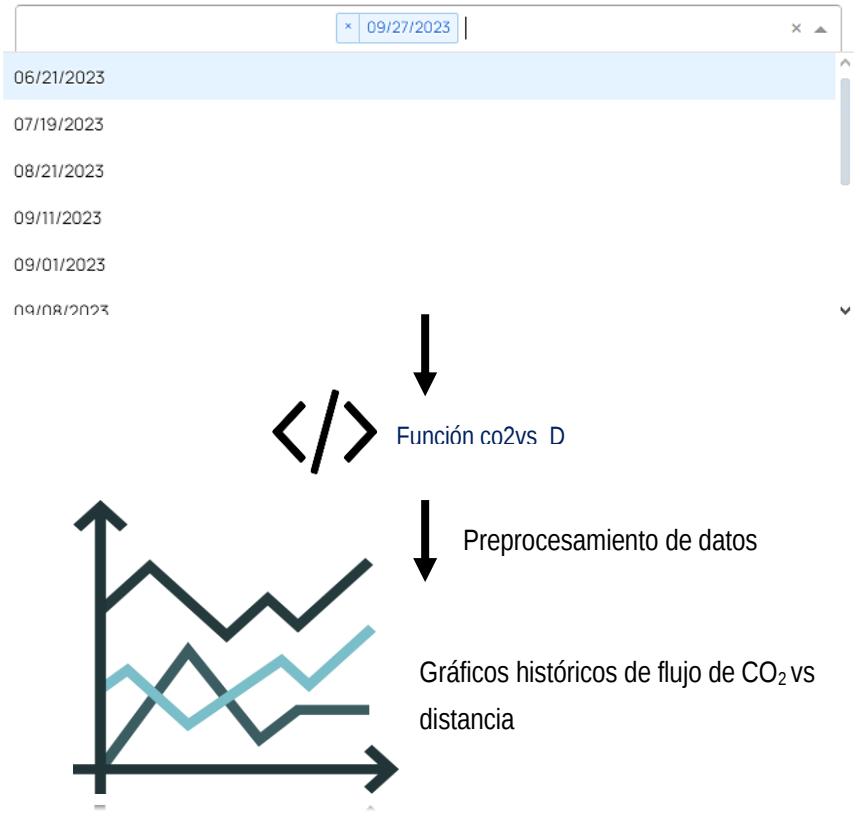


Ilustración 56 Esquema del Callback para el gráfico de flujo de CO_2 vs distancia

El resumen de los callbacks se puede ver en el siguiente diagrama que describe la interactividad de la aplicación web.

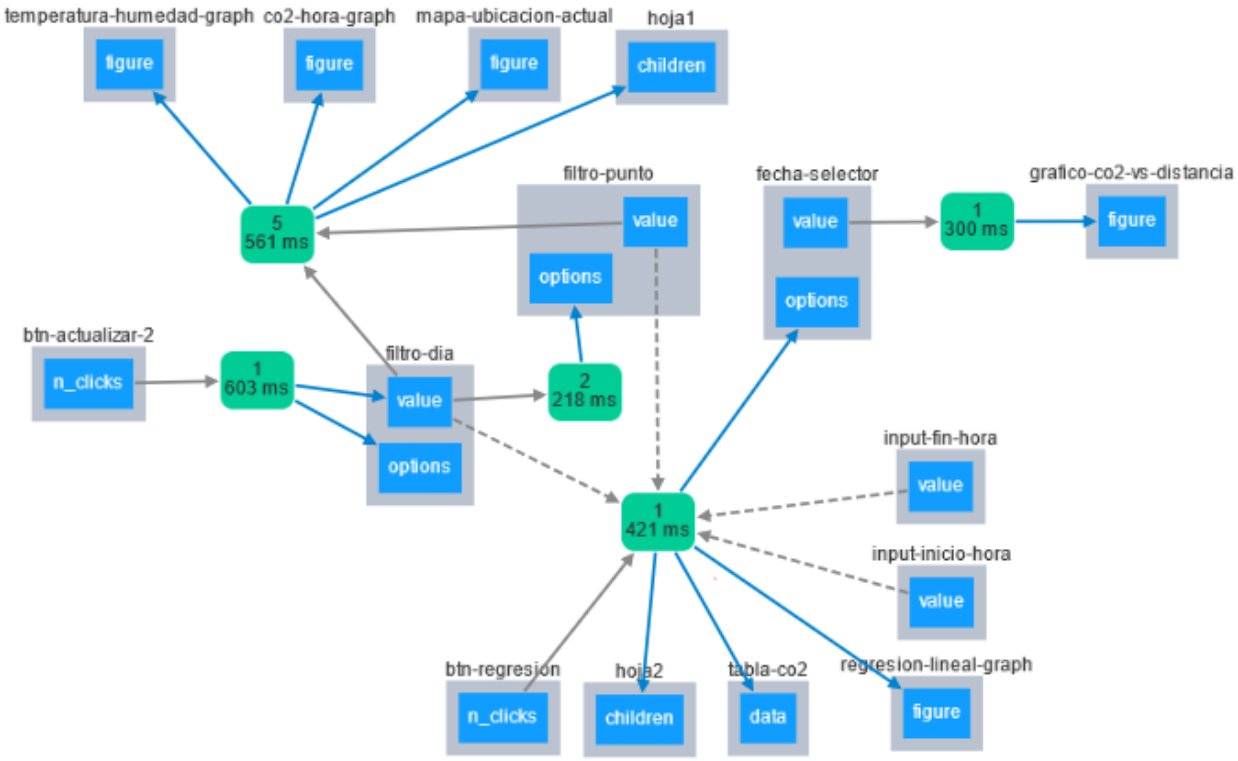


Ilustración 57 Diagrama de Callbacks de la aplicación web

El diagrama detalla la relación entre los componentes interactivos de la interfaz de usuario (UI) y las visualizaciones de datos. Cada nodo representa un componente de la UI o un objeto de datos, y las aristas representan las dependencias de callback entre estos nodos.

Entradas (Inputs): Los nodos en donde salen aristas[15] grises representan los puntos de entrada que desencadenan los callbacks. Por ejemplo, btn-actualizar-2 indica un botón en la UI cuyo número de clics (n_clicks) actúa como entrada para un callback. Otros nodos, como filtro-dia y fecha-selector,

[15] La aristas punteadas representan dependencias indirectas que no se activan directamente por un evento de usuario, sino por el resultado de otros callbacks y las solidas son dependencias directas que desencadenan la ejecución de los callbacks y la actualización de los componentes de salida de manera inmediata tras un evento de usuario.

demuestran cómo los valores seleccionados por el usuario en controles interactivos, como selectores de fecha o filtros, son pasados al servidor para su procesamiento.

Salidas (Outputs): Los nodos de salida representan los componentes que se actualizan como resultado de un callback. Estos incluyen “figure” para la actualización de gráficos, “children” para la actualización del contenido de un componente y “data” para la actualización de los datos presentados en una tabla, todos ellos reciben aristas azules.

Dependencias y Tiempos de Respuesta: Las aristas que conectan los nodos azules y verdes indican las dependencias de los callbacks. Los números sobre estas aristas pueden indicar la secuencia o la cantidad de activaciones de los callbacks, y los tiempos (por ejemplo, 561 ms) sugieren la latencia o el tiempo de procesamiento asociado con cada callback.

4.2.5 Análisis de Datos.

En el análisis y procesamiento de los datos hay dos puntos fuertes en el código que corresponden a la selección de datos para la regresión lineal utilizando la librería sklearn, y la preparación de datos para el gráfico de flujo de CO_2 vs distancia.

4.2.5.1 Regresión lineal para el cálculo de flujo de CO_2.

El fragmento que controla ésta tarea depende de los intervalos ingresados por el usuario, en el eje del gráfico de concentración de CO_2 se muestra la hora en que los datos fueron registrados

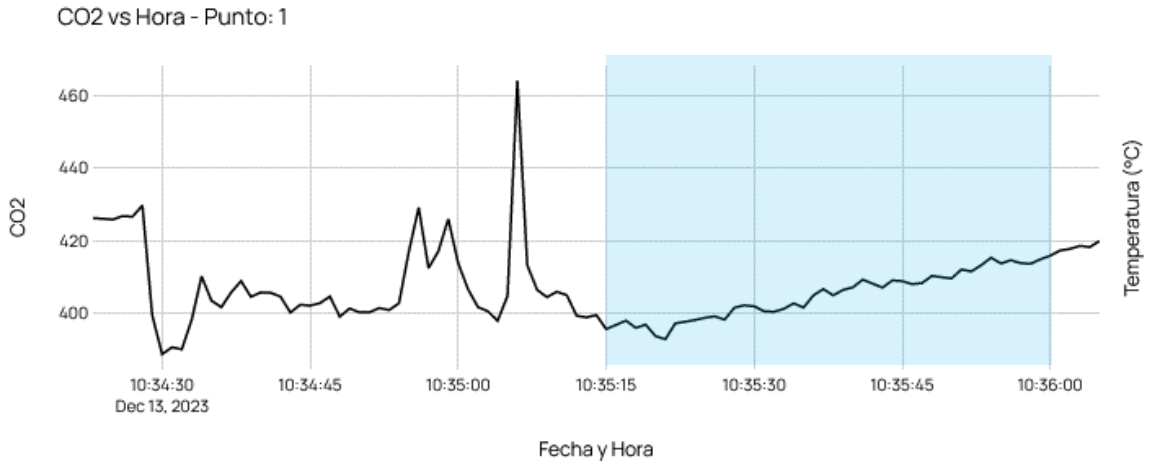
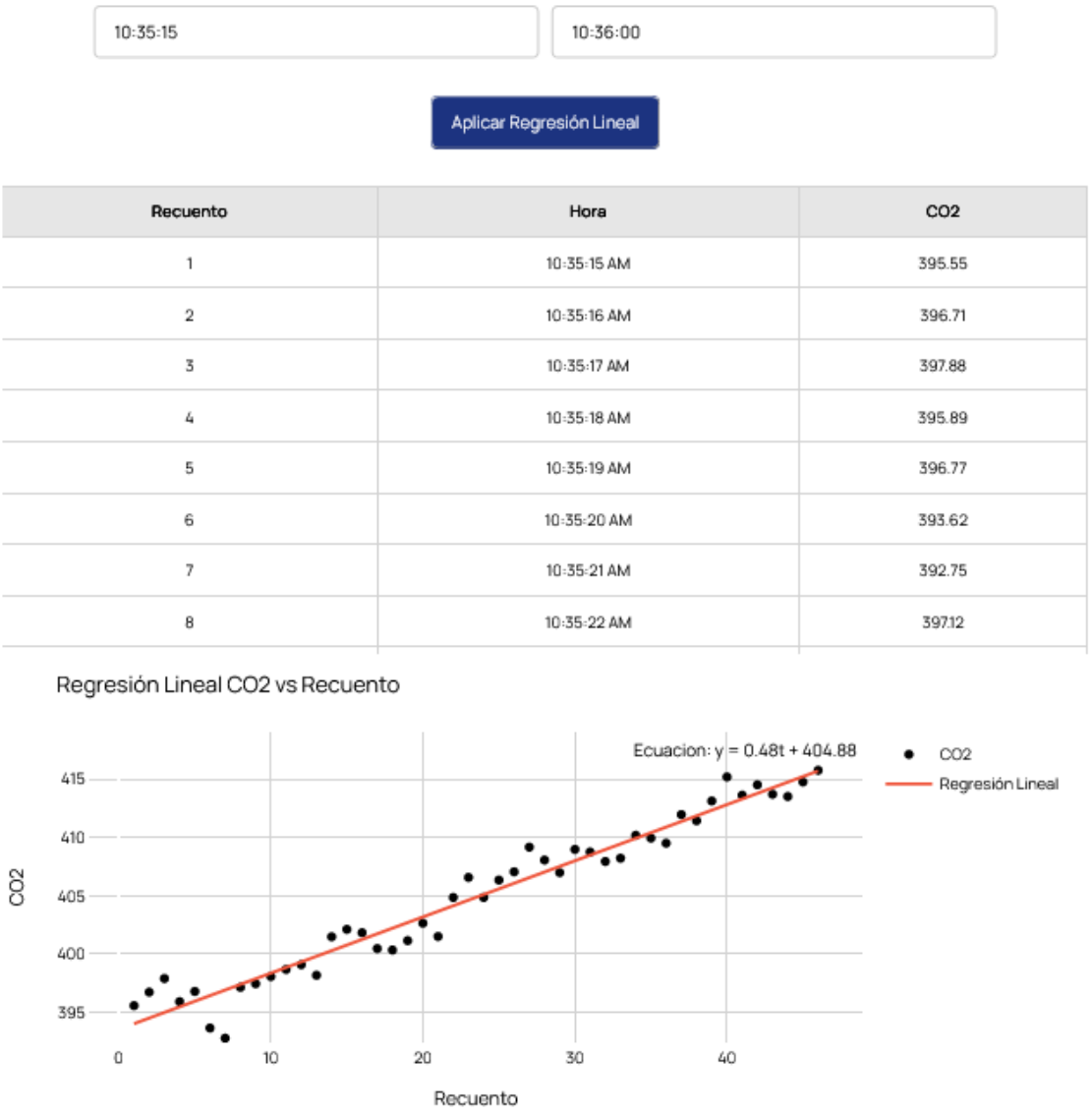


Ilustración 58 Generación del gráfico de concentración de CO_2

A este gráfica se le aplicará la regresión lineal desde la hora 10:35:15 hasta las 10:36:00 aproximadamente, como se ha explicado el usuario ingresa los intervalos como se muestra en la Ilustración 55, e inmediatamente se creará un gráfico con dicha regresión y una tabla de los datos de ese intervalo de hora.

El resultado se observa en la Ilustración 49.



Recuento	Hora	CO2
1	10:35:15 AM	395.55
2	10:35:16 AM	396.71
3	10:35:17 AM	397.88
4	10:35:18 AM	395.89
5	10:35:19 AM	396.77
6	10:35:20 AM	393.62
7	10:35:21 AM	392.75
8	10:35:22 AM	397.12

Ilustración 59 Resultado de la aplicación de regresión lineal para el gráfico de la Ilustración 48

De acá se puede extraer el valor de la pendiente de la ecuación generada la cual dicta en el ejemplo, de tal manera que el flujo ronda un valor de 0.48 ppm/s en la zona medida.

A parte de mostrar al usuario la línea de tendencia automáticamente se registra el valor de la pendiente, la fecha, el punto de medición y las coordenadas en la “hoja 2”.

Fecha Mes	Pendientes	Punto Medicion	Latitud	Longitud
12/13/2023	0.4832088	1	13.8434300	-89.5724180
+ ≡ Hoja 1 ▾	Hoja 2 ▾			

Tabla 17 Guardado de datos en la ejecución de la aplicación de regresión lineal.

Este guardado se utilizará más tarde en el programa de acuerdo al Fragmento 26.

```
def aplicar_regresion_lineal(n_clicks, dia_seleccionado, inicio_hora, fin_hora,
punto_seleccionado):
    global df2, data_count_df2, fechas_unicas
# Inicialización de la figura de Plotly para la regresión lineal----------------------
-------------------------------------------------------------------------------
    fig_regresion_lineal = go.Figure()
# Inicialización de DataFrame para los datos del intervalo de tiempo seleccionado-----
-------------------------------------------------------------------------------
    intervalo_co2_hora = pd.DataFrame()
    pendiente = 0
    punto_etiqueta = None
    latitud = None
    longitud = None
    options = []

    if not n_clicks or not inicio_hora or not fin_hora:
        return "Registros de Flujo CO2: 0", fig_regresion_lineal, [], options
    try:
        if n_clicks and inicio_hora and fin_hora:
            df_filtrado = df[df['Fecha'] == dia_seleccionado]
            inicio_tiempo = datetime.strptime(inicio_hora, '%H:%M:%S').time()
            fin_tiempo = datetime.strptime(fin_hora, '%H:%M:%S').time()

            if punto_seleccionado is not None:
                df_filtrado = df_filtrado[df_filtrado['Punto'] == punto_seleccionado]
                punto_etiqueta = df_filtrado['Punto'].iloc[0]
                latitud = df_filtrado['Latitud'].iloc[0]
                longitud = df_filtrado['Longitud'].iloc[0]

            intervalo_co2_hora = df_filtrado[(df_filtrado['Fecha y Hora'].dt.time >=
inicio_tiempo) & (df_filtrado['Fecha y Hora'].dt.time <= fin_tiempo)][['Hora', 'CO2']]
            intervalo_co2_hora['Recuento'] = range(1, len(intervalo_co2_hora) + 1)

            if len(intervalo_co2_hora) > 1:
                X = intervalo_co2_hora[['Recuento']]
                y = intervalo_co2_hora['CO2']
                regresion = LinearRegression()
                regresion.fit(X, y)
```

```
                y_pred = regresion.predict(X)
                pendiente = regresion.coef_[0]
                intervalo_co2_hora['y_pred'] = y_pred
                fig_regresion_lineal =
create_regresion_lineal_graph(intervalo_co2_hora, pendiente)
#Bloque para enviar datos-----------------------------------------------------------
-------------------------------------------------------------------------------
                if punto_seleccionado is not None:
                    fecha_seleccionada = datetime.strptime(dia_seleccionado,
'%m/%d/%Y').strftime('%Y-%m-%d')
                    values = [[fecha_seleccionada, pendiente, punto_etiqueta, latitud,
longitud]]
                    results = sheets.values().append(spreadsheetId=SPREADSHEETS_ID,
range='Hoja 2!A1',
                                                     valueInputOption='USER_ENTERED',
                                                     body={'values': values}).execute()

                result2 = sheets.values().get(spreadsheetId=SPREADSHEETS_ID,
range='Hoja 2!A:E').execute()
                values2 = result2.get('values', [])
#-----------------------------------------------------------------------------------
-------------------------------------------------------------------------------
                if values2:
                    expected_columns = ['Fecha Mes', 'Pendientes', 'Punto Medicion',
'Latitud', 'Longitud']
                    if values2[0] == expected_columns:
                        df2 = pd.DataFrame(values2[1:], columns=values2[0])
                        df2['Fecha Mes'] = pd.to_datetime(df2['Fecha Mes'])
                    else:
                        df2 = pd.DataFrame(columns=expected_columns)
                else:
                    df2 = pd.DataFrame(columns=['Fecha Mes', 'Pendientes', 'Punto
Medicion', 'Latitud', 'Longitud'])

        else:
            raise ValueError("Por favor, ingrese un formato de hora válido
(HH:MM:SS)")
    except ValueError as e:
        raise dash.exceptions.PreventUpdate("Error de validación: " + str(e))

    fechas_unicas = df2['Fecha Mes'].dt.strftime('%m/%d/%Y').drop_duplicates().values
    options=[{'label': fecha, 'value': fecha} for fecha in fechas_unicas]
    data_count_df2 = len(df2)

    return "Registros de Flujo CO2:"+f" {data_count_df2}",fig_regresion_lineal,
intervalo_co2_hora.to_dict('records'),options
```

Fragmento 35 Función encargada de la regresión lineal, la creación de tabla y gráfico y envío de datos a la Hoja 2.

La función denominada aplicar_regresion_lineal está diseñada para ser invocada en respuesta a la interacción del usuario, específicamente al activar un control de la interfaz de usuario. A continuación se detalla la lógica y estructura de la función:

4.2.5.1.1 Parámetros de Entrada:

La función acepta varios parámetros que reflejan los criterios de filtrado y selección del usuario:

- n_clicks: la cantidad de veces que se ha interactuado con un botón específico, sirviendo como un indicador de activación para la función.
- dia_seleccionado: la fecha elegida por el usuario para el análisis.
- inicio_hora y fin_hora: los límites del intervalo de tiempo dentro del cual se realizará el análisis de la regresión.
- punto_seleccionado: el punto geográfico específico de interés.

4.2.5.1.2 Inicialización y Verificación de Condiciones**:**

Se comienza con la inicialización de una figura de Plotly vacía y un DataFrame vacío, seguido de una verificación para asegurarse de que se han proporcionado los parámetros necesarios para proceder con el análisis.

4.2.5.1.3 Filtrado y Preparación de Datos:

Utilizando los parámetros de entrada, la función filtra el conjunto de datos principal para obtener las observaciones relevantes para el día seleccionado y el intervalo de tiempo. Además, si se ha especificado un punto, se realiza un filtrado adicional para concentrarse en ese punto geográfico.

4.2.5.1.4 Regresión Lineal:

Con los datos filtrados, se aplica una regresión lineal utilizando la clase LinearRegression del paquete “scikit-learn”, si hay suficientes datos para realizar el cálculo. Se calcula la pendiente de la relación lineal (Scikit-learn developers. (n.d.), 2023) y se añade a los datos junto con la línea de tendencia predicha.

La regresión simple utilizada intenta modelar la relación entre la variable dependiente (concentración de CO_2) y la variable independiente (tiempo), mediante el ajuste de una ecuación lineal a los datos observados. La matemática detrás de la regresión lineal, ya sea que sea realice con scikit-learn o con métodos estadísticos más tradicionales, es fundamentalmente la misma.

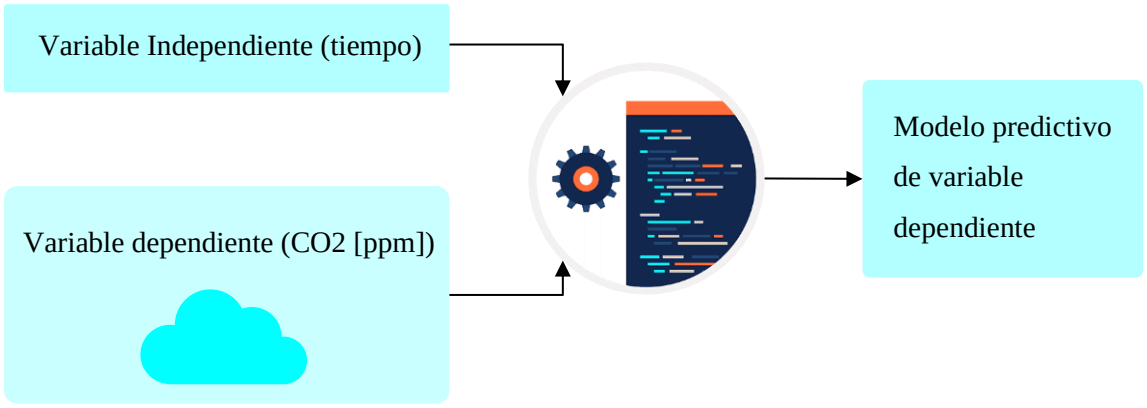


Ilustración 60 Esquema del concepto de regresión lineal

La forma de la ecuación para la regresión lineal simple, donde se tiene una sola variable dependiente, es:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

y : Variable dependiente
x : Variable independiente
β_0 : Intercepto de la linea con el eje y (constante)
β_1 : Pendiente de la línea de tendencia, representa el cambio y para un cambio unitario en x
ε : Error representando la diferencia entre las observaciones y el modelo.

Ilustración 61 Ecuación del modelo de la regresión lineal

El objetivo de la regresión lineal es encontrar los valores de β_0 y β_1 que minimizan la suma de los cuadrados de los errores, que es la suma de las diferencias cuadráticas entre los valores observados y los valores predichos por el modelo. Esto se hace comúnmente utilizando el método de los mínimos cuadrados ordinarios.

La implementación de LinearRegression en scikit-learn utiliza el método de los mínimos cuadrados ordinarios para estimar β_0 y β_1 de manera matricial, este método es en efecto, una forma de implementar el método de los mínimos cuadrados, pero se realiza de manera más eficiente y generalizable (Scikit-learn developers. (n.d.), 2023), especialmente para casos de regresión múltiple (Kutner, Nachtsheim, Neter, & Li, 2004).

4.2.5.1.4.1 Método de inversión de matrices en regresión lineal.

Este método también es conocido como la solución de ecuaciones normales, involucra las siguientes operaciones:

Primero se construye una matriz de diseño $\mathbf{X}$ que incluye una columna de unos para el término de intercepción y las columnas restantes para las variables dependientes. En este caso se trabaja de forma unidimensional teniendo solos dos columnas: una de unos (para β_0) y otra con los valores de tiempo (para β_1).

Los coeficientes se calculan resolviendo las ecuaciones normales:

$$\mathbf{X}^T\mathbf{X}\ \beta = \mathbf{X}^T\mathbf{Y}$$

$$\beta = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

β: Vector de coeficientes β_0 y β_1

$\mathbf{Y}$: Vector de la variable dependiente.

Ilustración 62 Coeficientes de regresión (Kutner, Nachtsheim, Neter, & Li, 2004)

Este método es directo y matemáticamente elegante, pero puede ser computacionalmente costoso y problemático con matrices que son casi singulares[16]. En tales casos, puede surgir problemas de precisión numérica, lo que hace que otros métodos como la descomposición en valores singulares (SVD), sean preferibles para la regresión lineal. Aunque en este caso, debido a que se trata de una regresión lineal simple y unidimensional en la variable independiente es muy poco probable que surjan matrices singulares en el proceso (excepto cuando todas las observaciones de la variable independiente sean idénticas). En contextos más complejos con múltiples variables independientes el riesgo de singularidad aumenta.

4.2.5.1.5 Integración con Google Sheets**:**

La función también incluye una integración con Google Sheets, donde los resultados de la regresión lineal, junto con detalles geográficos y temporales, se envían a una hoja 2 de cálculo para su almacenamiento y análisis posterior. Esto ocurre inmediatamente que el usuario pulse el botón de aplicar

[16] Matriz cuadrada que no tiene inversa cuando su determinante es cero

regresión lineal. Obsérvese que el código contiene una sección similar a la que se mostró en el Fragmento 25. Este bloque también actualiza el filtro para obtener el gráfico de concentración de CO_2 vs distancia.

Ilustración 63 Esquema de envío de datos a la Hoja 2 al calcular la regresión lineal

4.2.5.1.6 Creación de la Visualización:

Se invoca la función auxiliar, denominada create_regresion_lineal_graph, para construir la visualización gráfica de la regresión lineal basada en los resultados obtenidos. Esta función es igual a las demás funciones en donde se crean gráficos basados en Plotly.

4.2.5.1.7 Manejo de Excepciones:

La función incluye un manejo de excepciones robusto para garantizar que cualquier error en la entrada o en el proceso de análisis sea manejado adecuadamente, previniendo así la actualización de la interfaz de usuario con datos erróneos, como por ejemplo evitar que el usuario coloque mal la hora en los campos de los intervalos para evaluar la regresión lineal.

4.2.5.1.8 Retorno de Resultados:

Finalmente, la función devuelve un conjunto de resultados que incluyen una cadena de texto con el conteo de registros de flujo de CO_2, la figura de regresión lineal de Plotly actualizada, los datos de CO_2 en el intervalo seleccionado en un formato adecuado para su uso en Dash, y las opciones actualizadas para los componentes de selección de la interfaz de usuario.

4.2.5.2 Creación de gráfico de flujo de CO_2 vs distancia según fecha registrada.

Este gráfico será útil para comparar los niveles de flujo en una línea base en diferentes épocas del año en diferentes campañas de medición de tal manera que se puede observar una evolución. Para crear este grafico se toman todas las pendientes de acuerdo a una fecha seleccionada y se promedian para obtener un punto. Es decir si hay varios registros de pendientes para un mismo punto de medición entonces se promedian todos ellos para mostrar un único punto en el gráfico.

La función encargada de ésta tarea es llamada “créate_flux_vs_distance_graph” y se centra en la representación gráfica de la relación entre puntos de medición y las pendientes promedio calculadas a partir del conjunto de datos.

Inicialmente, la función configura un gráfico en blanco de Plotly. Luego, procesa una serie de fechas seleccionadas, convirtiéndolas de formatos de cadena a objetos datetime.date. Esta conversión facilita la filtración de datos en el DataFrame de acuerdo con las fechas especificadas.

Posteriormente, la función itera sobre cada fecha seleccionada. Y se filtra el DataFrame para obtener sólo los datos correspondientes a esa fecha. Después, se realiza un segundo bucle que recorre todos los puntos de medición disponibles. En este bucle, se calcula el promedio de las pendientes para cada punto de medición. Si no hay datos disponibles para un punto específico, se asigna un valor predeterminado.

Una vez obtenidos los promedios de pendientes para cada punto y fecha, se añaden al gráfico como una serie de puntos y líneas (trazas), diferenciadas por fecha. Cada traza en el gráfico representa la variación de las pendientes promedio a lo largo de los distintos puntos de medición para una fecha dada.

```
def create_flux_vs_distance_graph(df2_cleaned, selected_dates):
    fig = go.Figure()
    print(selected_dates)

    # Obtén todos los puntos únicos en el dataframe
    all_puntos = sorted(df2_cleaned['Punto Medicion'].unique())

    for selected_date in selected_dates:
        # Convertir la fecha seleccionada a datetime.date
        selected_date = datetime.strptime(selected_date, '%m/%d/%Y').date()
        data = df2_cleaned[df2_cleaned['Fecha Mes'].dt.date == selected_date]
        pendientes_promedio = []
        for punto in all_puntos:
            punto_data = data[data['Punto Medicion'] == punto]

            if not punto_data.empty:
                # Si hay datos para el punto actual, calcula el promedio de pendientes
                pendientes_promedio.append(punto_data['Pendientes'].apply(lambda x:
np.mean([float(val.strip()) for val in x.split()])).mean())
            else:
                # Si no hay datos para el punto actual, añade un cero
                pendientes_promedio.append(0)

        fig.add_trace(go.Scatter(x=all_puntos, y=pendientes_promedio,
mode='markers+lines', name=f'Fecha {selected_date}'))

    fig.update_layout(
        title='CO2 vs Distancia',
        font=dict(family='Manrope', size=10, color='black'),
        xaxis=dict(title='Punto Medicion', showgrid=True, tickvals=all_puntos,
dtick=1),
```

```
        yaxis=dict(title='Pendientes Promedio', showgrid=True, dtick=1),
        plot_bgcolor='white',
        margin=dict(l=10, r=10, t=60, b=30),
    )
    fig.update_xaxes(gridcolor='lightgrey', linewidth=1)
    fig.update_yaxes(gridcolor='lightgrey', linewidth=1)
    return fig
```

Fragmento 36 Creación de gráfico de flujo de CO_2 vs distancia

Los argumentos de ésta función requiere de datos filtrados y limpiados, como eliminar filas con valores nulos, eliminar filas con valores vacíos y una conversion de datos a tipo “int” en cuanto al número de punto de medición.

El resultado de esta función es el siguiente gráfico, con valores de prueba en diferentes fechas, se observa que en el filtro al seleccionar diferentes fechas se puede comparar los niveles de flujo en los mismos puntos de medición.[17]

Este gráfico se construye gracias a la actualización de los datos de la hoja 2 en el Fragmento 35 al momento de realizar la regresión lineal

[17] Se advierte que los puntos a comparar deben observarse primero en el mapa para comprobar que se trata de la misma línea base.

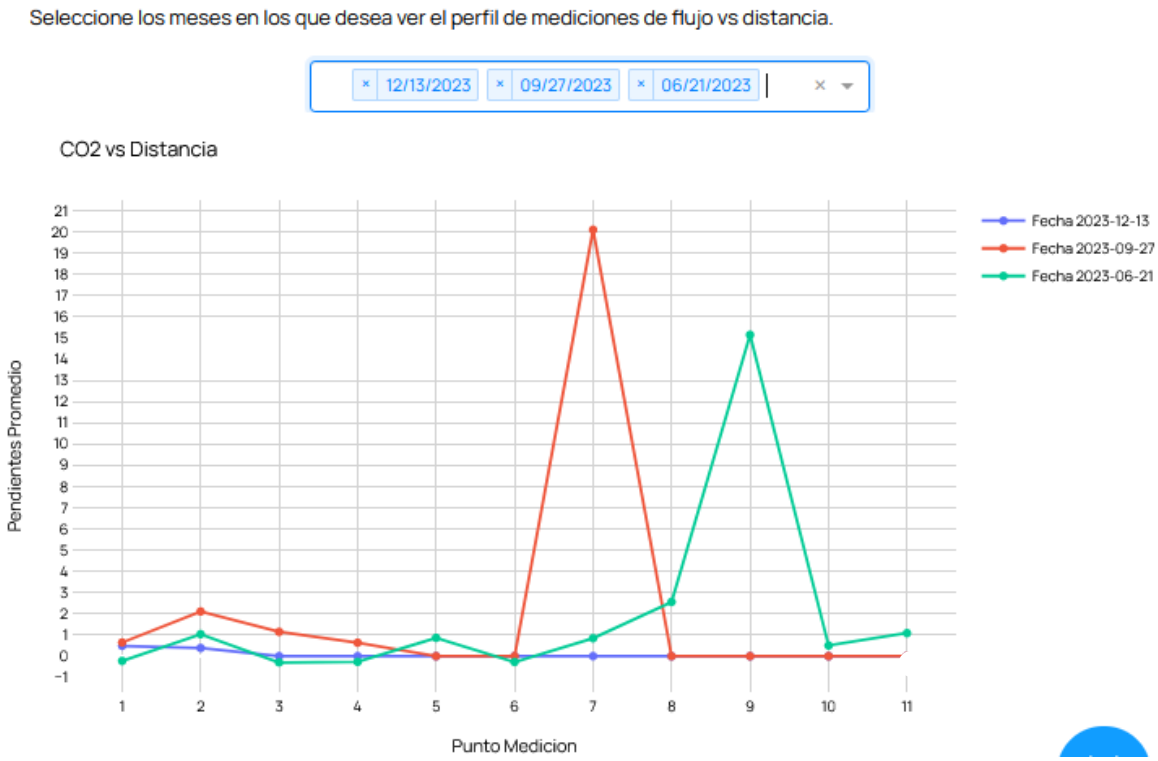


Ilustración 64 Gráfico de CO_2 vs distancia con valores de prueba.

4.2.6 Adición de mapa.

Para la integración de un mapa, destacando la interacción del usuario para visualizar los datos geoespaciales guardados en la hojas de cálculo. La implementación comienza configurando un token acceso a Mapbox, una herramienta de visualización de mapas. Laa función update_graph (Fragmento 31), se separa un bloque para esa tarea, definida como un callback en Dash, se activa mediante la selección de fechas y puntos por parte del usuario.

Mapbox es una plataforma de mapeo y localización que ofrece diversas herramientas y servicios para crear mapas personalizados y soluciones de localización. Proporciona APIs y SDKs que permiten a los desarrolladores integrar mapas interactivos y funciones de localización en aplicaciones web y móviles.

Cuando se selecciona una fecha, se filtra el conjunto de datos para esa fecha específica, y se crea un mapa interactivo con marcadores que representan mediciones en distintas coordenadas geográficas. Este

mapa se genera utilizando Scattermapbox de Plotly, donde cada marcador indica la ubicación de una medición con detalles adicionales.

```
#API para el mapa-----------------------------------------------------------

px.set_mapbox_access_token('pk.eyJ1IjoiZGF2aWRhZHJpZWwiLCJhIjoiY2xtbXoyZWh5MHB4ejJxcXN
sdW5sc3BneSJ9.hg7yqKdO_lRO0UfCrdR???')

def update_graphs(dia_seleccionado, punto_seleccionado):
# ----------------------------------------------------------------------------
#----------Integración del mapa-----------------------------------------
        mapa = go.Figure(
            data=go.Scattermapbox(
                lat = df_filtrado['Latitud'],
                lon = df_filtrado['Longitud'],
                mode = 'markers',
                marker = go.scattermapbox.Marker(
                    size = 10,
                    color = 'blue'
                ),
                text=texto_marcadores,
                textposition="top right",
            ),
            layout = go.Layout(
                title_text = 'Coordenadas de mediciones',
                font=dict(family='Manrope', size=12),
                autosize=True,
                hovermode='closest',
                showlegend=False,
                mapbox=dict(
                    accesstoken='pk.eyJ1IjoiZGF2aWRhZHJpZWwiLCJhIjoiY2xtbXoyZWh5MHB4ej
JxcXNsdW5sc3BneSJ9.hg7yqKdO_lRO0UfCrdRBDA',
                    bearing=0,
                    center=dict(
                        lat=13.5939,
                        lon=-89.3335
                    ),
                    pitch=0,
                    zoom=8,
                    style='outdoors'
                ),
                margin=dict(l=20, r=20, t=50, b=25),
            )
        )
  # ----------------------------------------------------------------------------
    return f"Lecturas de Sensores\n {data_count_df}",temperatura_humedad_graph,
mapa,co2_hora_graph
```

Fragmento 37 Integración del mapa con Mapbox

Observar que los DataFrames que son dados para la creación del mapa son los datos filtrados de la columna de longitud y latitud guardados en la hoja 1. La visualización del mapa es importante para el control de las ubicaciones de medición registradas.

El resultado de la visulaización del mapa puede observarse en la siguiente imagen:

Ilustración 65 Interacción con mapa creado por Mapbox

4.2.7 Despliegue de la aplicación web.

Dash está construido en Flask, Plotly y React.js. Flask es un microframework de Python para desarrollar aplicaciones web. La relación de Dash y Flask es que Dash utiliza a Flask como su servidor web. Esto significa que cuando se crea una aplicación con Dash, internamente está utilizando Flask para manejar las solicitudes HTTP. Flask proporciona la funcionalidad del servidor web, mientras que Dash se enfoca en la interfaz de usuario y la visualización de datos.

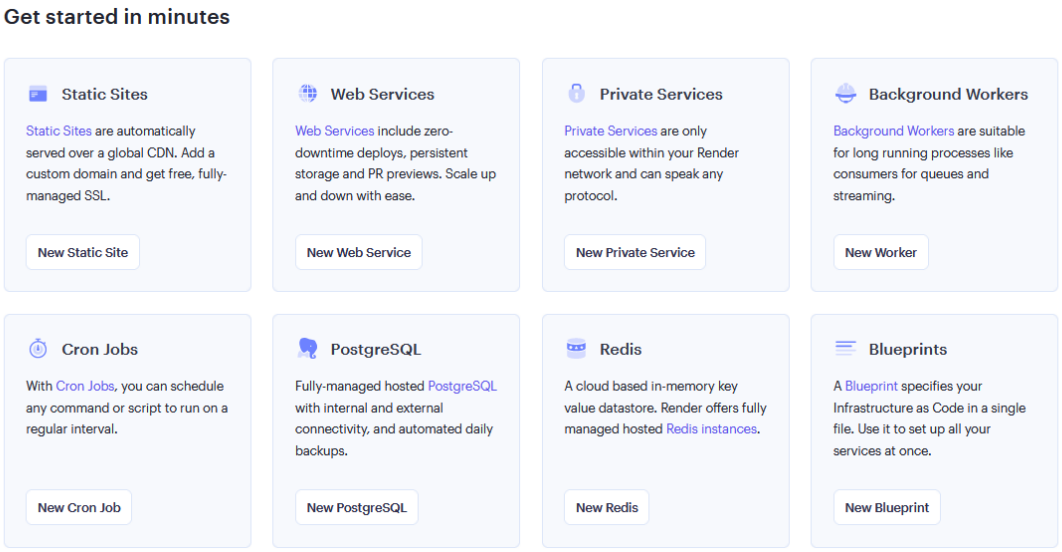
Primero es indispensable crear un repositorio del proyecto ya que debe estar alojado de manera remota además de que se necesita un control de versiones[18] de tal manera que se eligió GitHub para ello. Además es necesario llevar el proyecto a una versión de Ralease[19].

Luego para realizar el despliegue es necesario alojar el proyecto en un servidor remoto que sea capaz de mantenerlo en funcionamiento continuamente, además de encontrar la mejor configuración posible, por eso se eligió un "Paas" para la tarea como "Render"

Ilustración 66 Logo de Render

Los pasos a seguir para el despliegue final son los siguientes:

1. **Iniciar sesión en Render.** La plataforma ofrece varias opciones pero en este caso debido a que el proyecto está en el repositorio de GitHub. Se inicia sesión mediante la cuenta de GitHub.
2. **Elegir el servicio a solicitar.** En este caso se trata de un servicio web, por la interactividad y dinamismo de los datos manejados.



[18] Registro de cada actualización del proyecto.

[19] Se refiere a una versión final y funcional de proyecto.

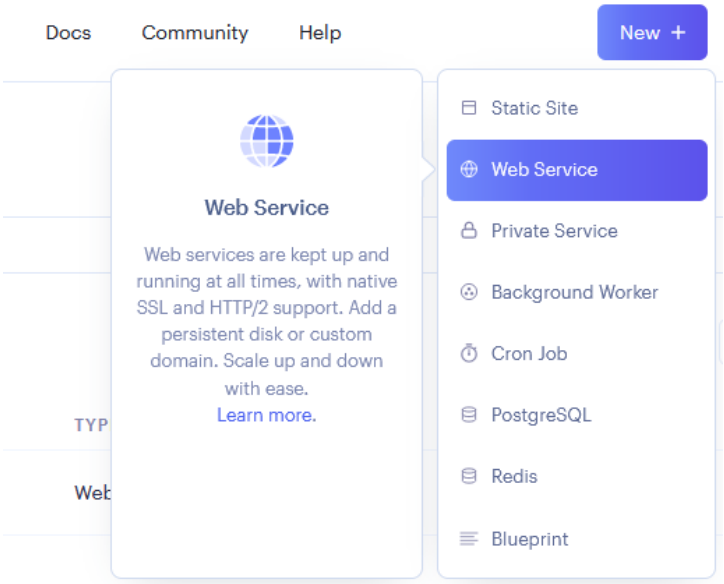
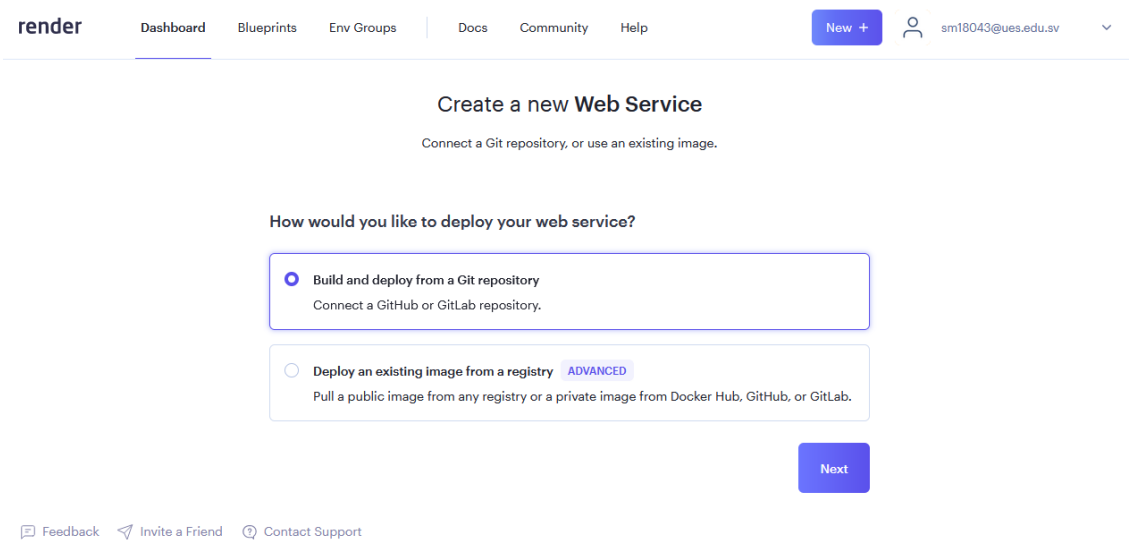


Ilustración 67 Opción "Web Service" para servicio en render.

3. **Elegir el proyecto del repositorio.** Render pregunta donde está alojado el proyecto al cual se le va realizar el despliegue.



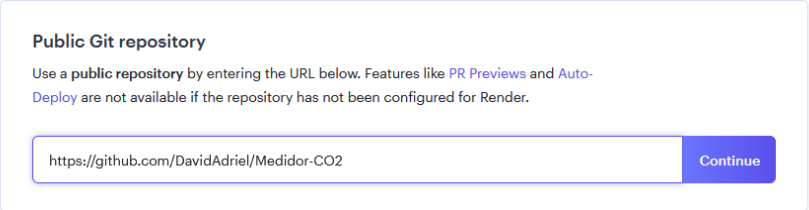


Ilustración 68 Selección del proyecto en el repositorio de GitHub

4. **Configuración para el servicio.** En esta sección se solicita el nombre de la página, la localización del servidor. En "Run time" se selecciona el entorno de ejecución del servicio web en este caso es Python 3. "Build Command" se ejecuta en el directorio raíz del repositorio cada vez que se publica una nueva versión del código o cuando se despliega manualmente. **S**tart Command" es el comando que inicia el servicio web en sí. Debe ejecutarse en el directorio raíz de la aplicación y es responsable de iniciar el proceso de la aplicación.

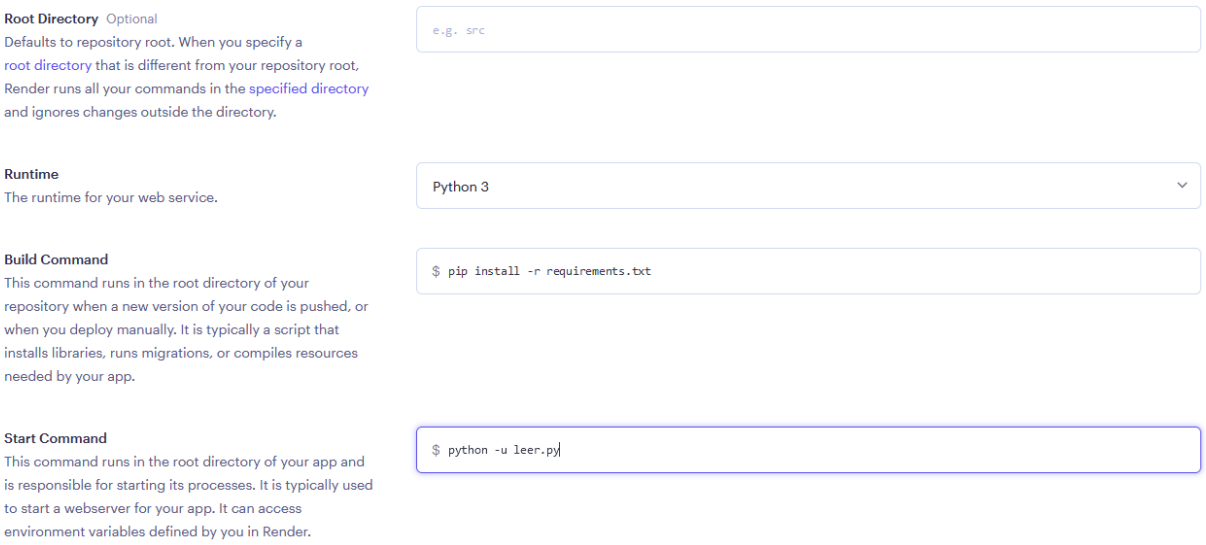


Ilustración 69 Configuración del despliegue

5. **Almacenamiento de variables de entorno.** Render ofrece configuración avanzada para almacenar variables de entorno de manera segura, lo que permite que el código acceda a ellas en tiempo de ejecución sin exponer información sensible en el repositorio del código o en el control de versiones. En este caso se ha añadido la key JSON codificada en base64 para Google Sheets. En el Fragmento 25 se observa cómo se manejan las credenciales. Google Cloud Plataform utiliza archivos JSON como una forma de distribuir credenciales de cuentas de servicio, que incluyen un ID de cliente, una clave privada y otros datos de configuración. Estos archivos a menudo se codifican en base64 cuando se necesitan incorporar en variables de entorno o configuraciones de despliegue.

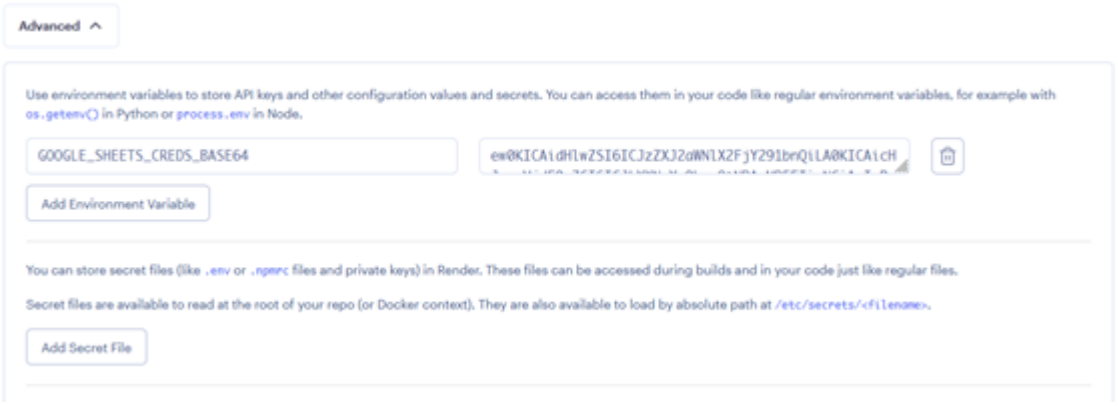


Ilustración 70 Adición de variable de entorno en configuración avanzada en Render

Finalmente luego de haber seguido los pasos anteriores el despliegue se completa mostrando la siguiente pantalla:

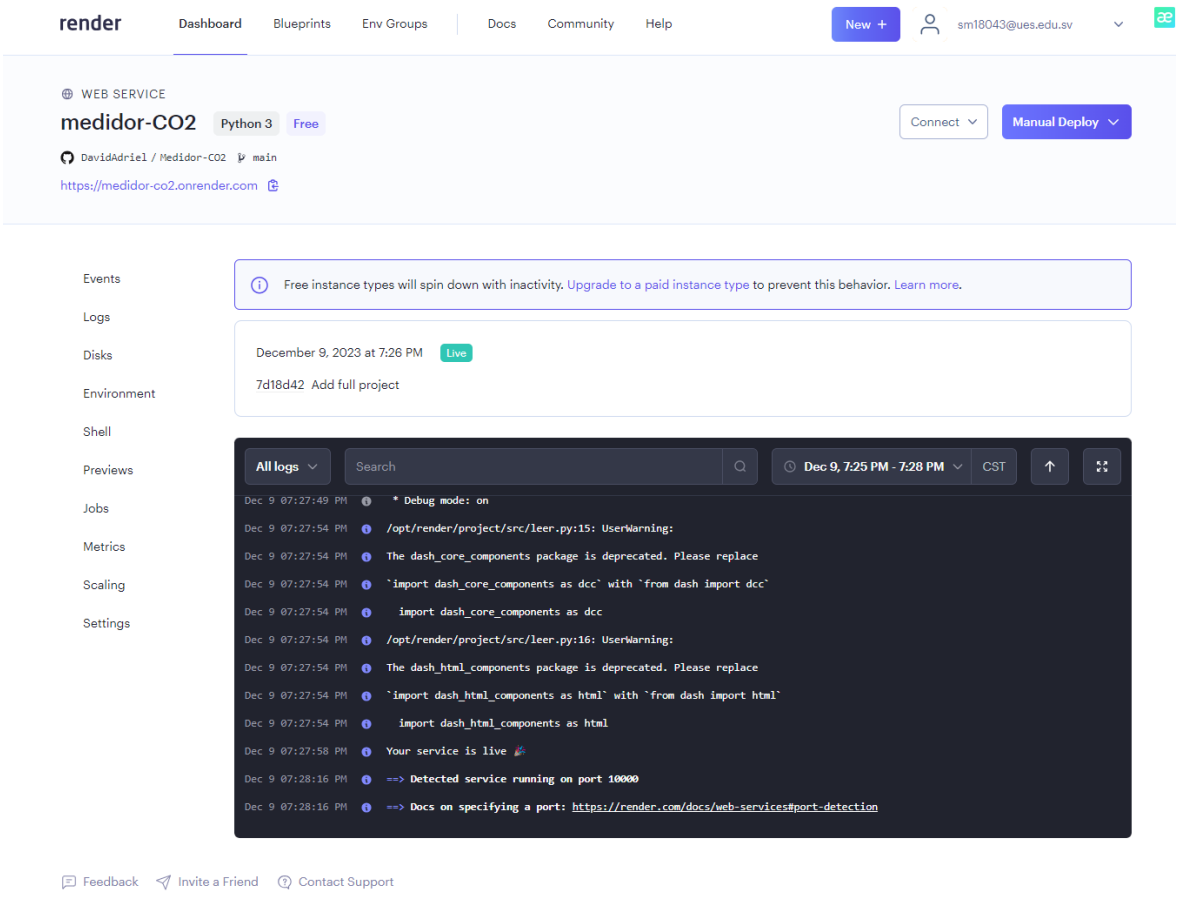


Ilustración 71 Sección de registros para el servicio web

Esta información brinda una visión general del estado del servicio en ese momento y ofrece pistas sobre los pasos a seguir para resolver problemas y optimizar el despliegue del servicio web.

4.2.8 Interfaz final.

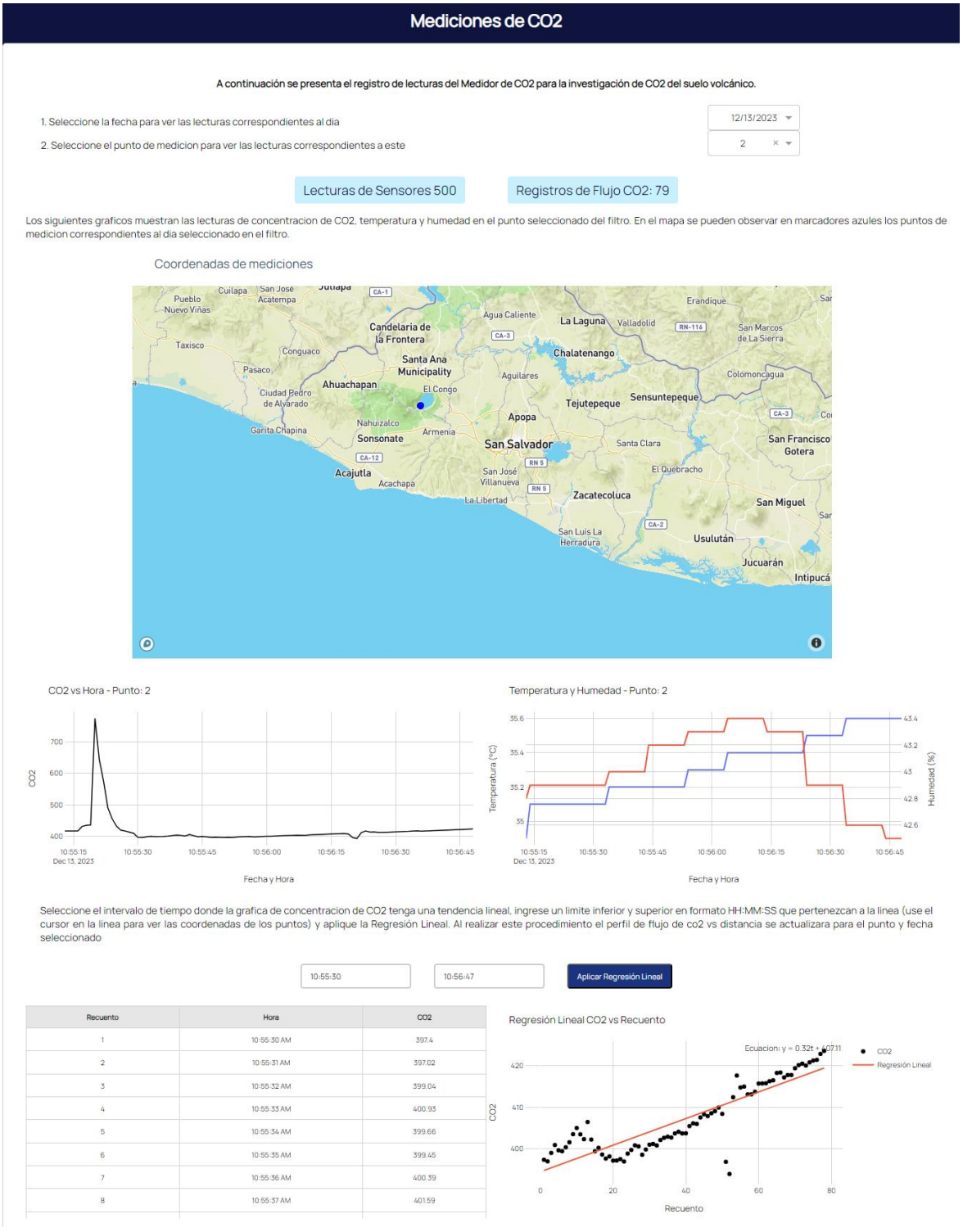


Ilustración 72 Interfaz de usuario de la aplicación web

Resumiendo la interfaz se compone en:

1. **Mapa Interactivo**: Presenta las coordenadas de las mediciones de CO_2 en un mapa, utilizando la plataforma de mapas Mapbox. Los marcadores en el mapa indican la ubicación de los puntos de medición.
2. **Gráficos de Datos**: Hay varios gráficos que muestran diferentes aspectos de los datos recopilados:
   - Gráfico de línea que muestra las lecturas de CO_2 en un punto específico.
   - Gráfico combinado de líneas que representa la temperatura y la humedad en un punto específico a lo largo del tiempo.
   - Gráfico de dispersión (scatter plot) que realiza una regresión lineal de los niveles de CO_2.
   - Un gráfico de flujo de CO_2 vs distancias de acuerdo a la fecha de medición y coordenadas.
3. **Tabla de Datos**: Una tabla que resume los resultados numéricos del intervalo seleccionado del gráfico de concentración de CO_2 vs tiempo.
4. **Controles Interactivos**: Incluye botones o controles deslizantes que permiten al usuario filtrar los datos mostrados en los gráficos y en el mapa, ajustando parámetros como la fecha y el punto de medición.
5. **Diseño y Estética**: La interfaz utiliza un esquema de colores sobrio y proporciona una visualización clara de los datos, lo que facilita la interpretación y el análisis de las mediciones de CO_2.

Capítulo 5

5 Guía de utilización del ecosistema de investigación.

Anteriormente se ha explicado brevemente acerca de la utilización del equipo de medición, la aplicación Android y la Dashboard o aplicación web. Estos tres elementos trabajan en conjunto para formar un ecosistema de medición e investigación para el CO_2 del suelo volcánico. En este capítulo se abarcará el proceso de medición a seguir por el usuario.

5.1 Medición en campo con la app Android “Medidor de CO_2 V 2.0”.

1. Encender el equipo

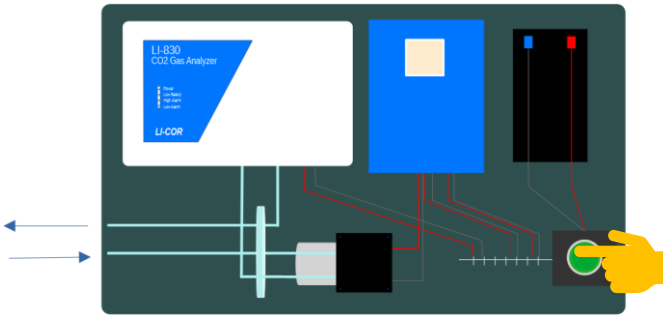


Ilustración 73 Esquema interno de la estación móvil

2. Una vez encendido el equipo se debe abrir la aplicación móvil “Medidor de CO_2”.

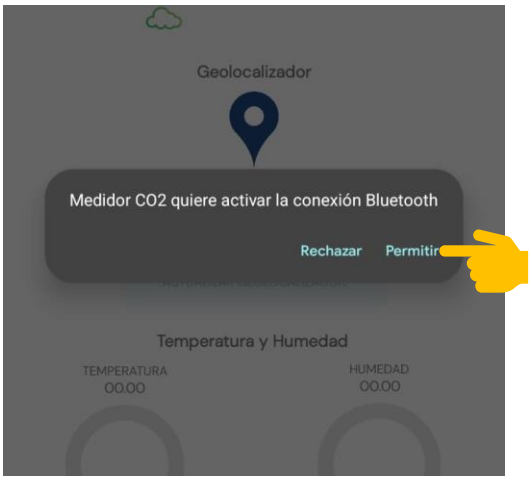


Ilustración 74 Petición de la app para encender el Bluetooth

3. Luego de dar permisos de bluetooth y almacenamiento, se debe vincular el CPU con la aplicación, en dispositivos guardados y presionar el filtro para mostrar el listado.

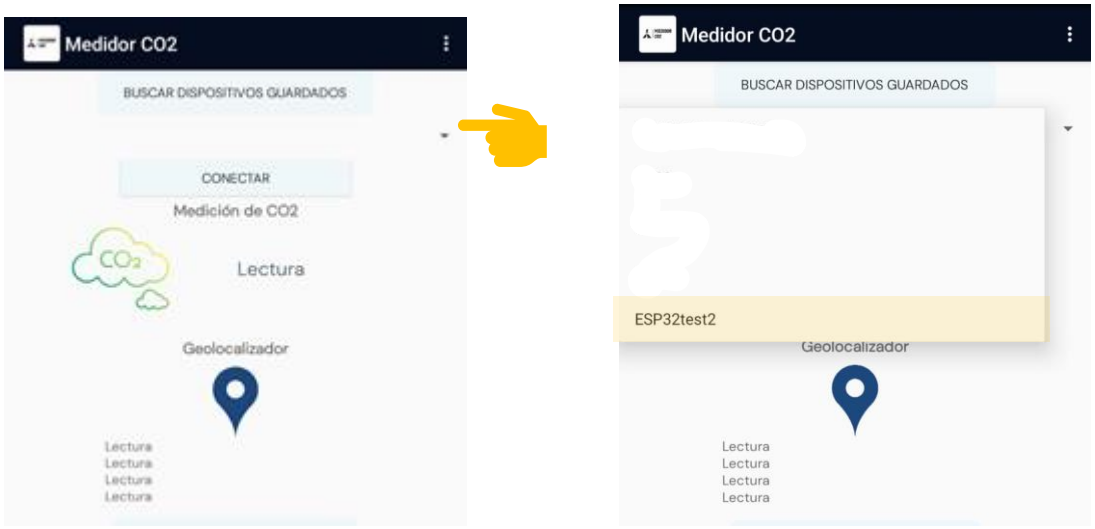


Ilustración 75 Spinner para selección de dispositivos

4. Luego de vincular por Bluetooth al CPU con la aplicación ya se muestran los resultados de medición de Temperatura y Humedad y las lecturas del GPS cuando esté activo. Para verificar que esté activo observar el cuadro donde aparezca “GPS: Activo”.
5. Verificar el estado de la conexión GPRS en el cuadro donde aparezca “GPRS:Conectado”, si no lo está intentar reconectar en el botón “Reconectar”.

Ilustración 76 Panel de información del CPU

6. Ingresar el número de punto de medición que se está analizando.

Ilustración 77 Inserción del punto de medición

7. Para empezar a medir CO_2 presione el botón Start, la bomba de la maleta se encenderá y empezara a generarse la gráfica en la aplicación.

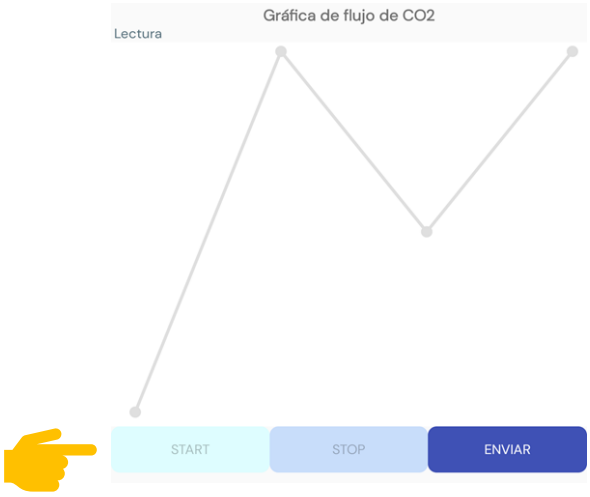
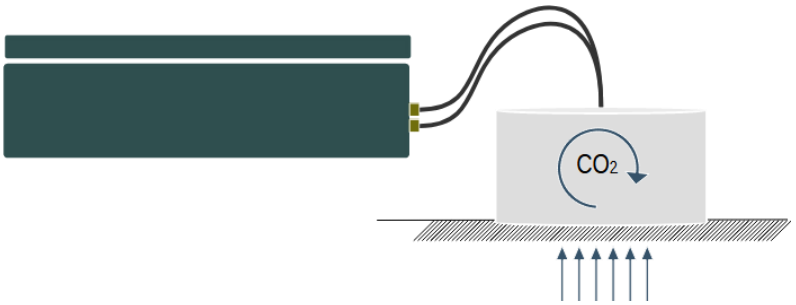


Ilustración 78 Pulsar el botón START para empezar a medir

Debe dejar fluir un momento el gas a través del sensor y luego colocar la cámara de acumulación en el suelo.



8. Cuando la gráfica en la app muestre un patrón lineal considerable, presionar el botón Stop.

Ilustración 79 Pulsar el botón STOP para detener la bomba y detener el gráfico

9. Para enviar los datos a la hoja de cálculo presione el botón “Enviar”. En el panel de información del CPU se mostrarán preparando la cantidad de datos a enviar.

Ilustración 80 Pulsar el botón STOP para detener la bomba y detener el gráfico

10. Luego se procederá al cálculo preliminar de la regresión lineal, seleccionando dos intervalos de la gráfica.

    Presionar el botón calcular y mostrará el cálculo del coeficiente de correlación lineal R^2.

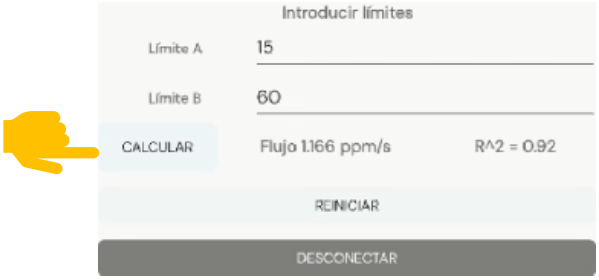


Ilustración 81 Cálculo de flujo de CO_2 en la aplicación

11. Presionar el botón Reiniciar una nueva medición en el mismo punto u otro. Esto reiniciará la gráfica y otros campos.
12. Luego de terminar de medir en todos los puntos. Presionar el botón Desconectar.

Ilustración 82 Reiniciar medición o desconectar la aplicación

5.2 Observación de resultados después de la medición en campo.

1. Ingresar al enlace: medidor-co2.onrender.com[20](), se observará la siguiente ventana.

Ilustración 83 Selección de filtros de fecha y punto de medición

Debe seleccionar el filtro indicado, para elegir las fechas donde se tienen registros de medición, y elegir el número de punto de medición analizar. De acuerdo a la fecha se mostraran donde se encuentran esos puntos en el mapa:

[20] Debe esperar a que cargue la aplicación web debido a que se trata de un servicio de host gratuito puede tardar un poco.

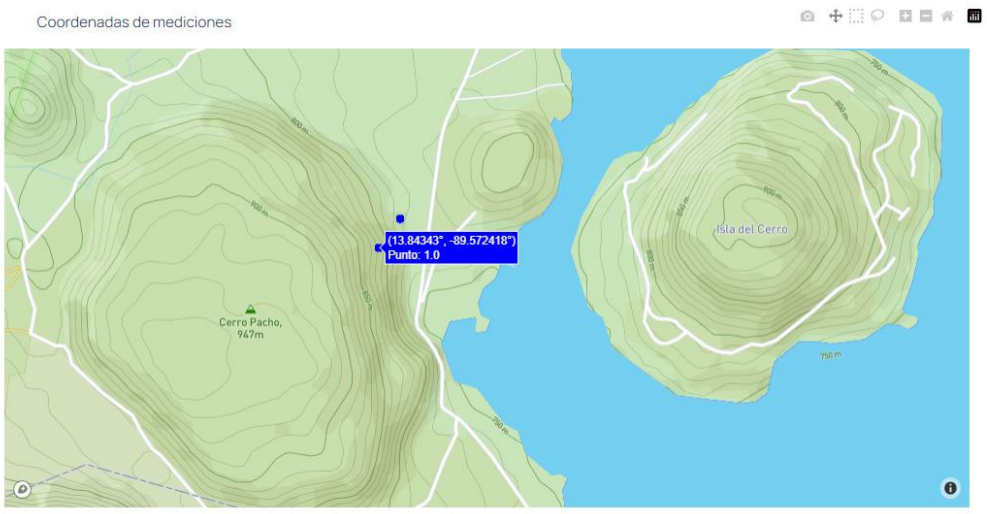


Ilustración 84 Mapa de puntos registrados

Al seleccionar el punto en el filtro se establecen todos los gráficos correspondientes a ese punto.

2. Observar las gráficas correspondientes al punto a analizar.

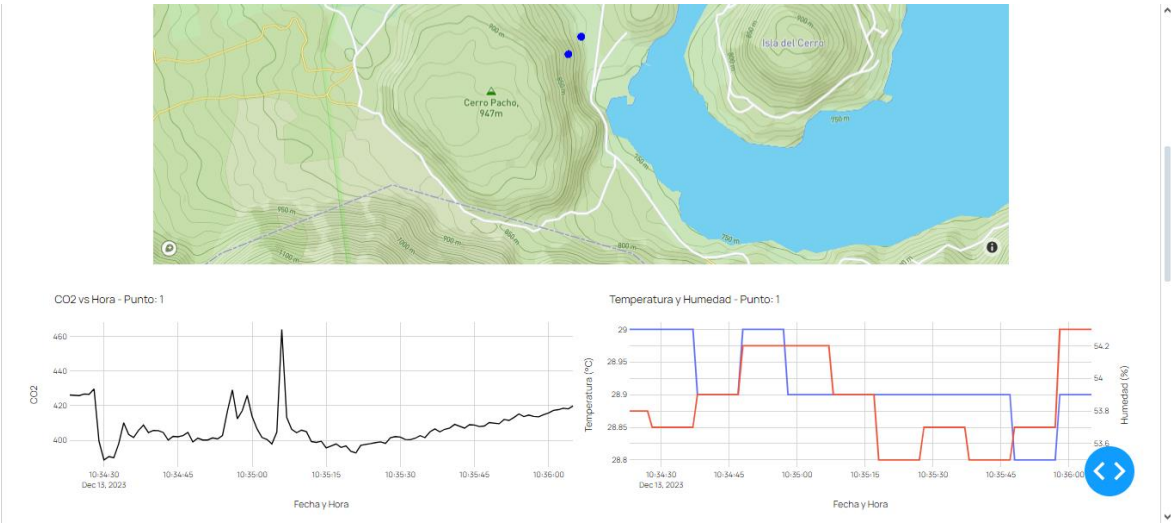


Ilustración 85 Gráficos de CO_2 vs tiempo y Temperatura y Humedad vs tiempo

3. Calcular la regresión lineal y flujo de CO_2 en el punto de medición, seleccionando un intervalo del gráfico de CO_2 vs Hora:

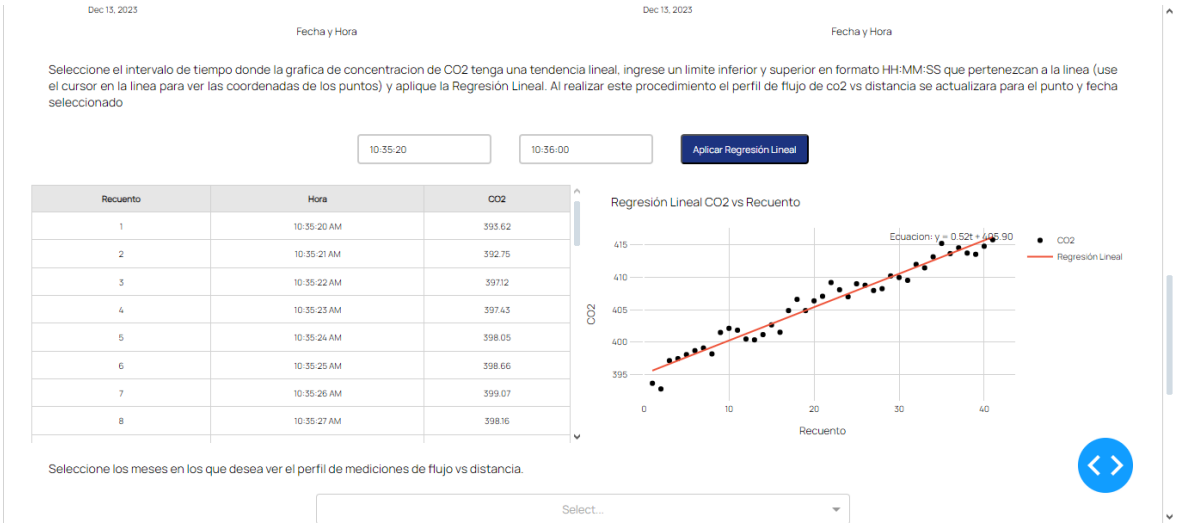


Ilustración 86 Cálculo de flujo de CO_2

Esta sección aparecerá vacía hasta que se presione el botón “Aplicar Regresión Lineal”. Al lado hay una tabla de los datos seleccionados y a la derecha un gráfico de la tendencia lineal con su ecuación.

4. Observar el perfil de Flujo de CO_2 vs distancia en diferentes fechas seleccionado el filtro.

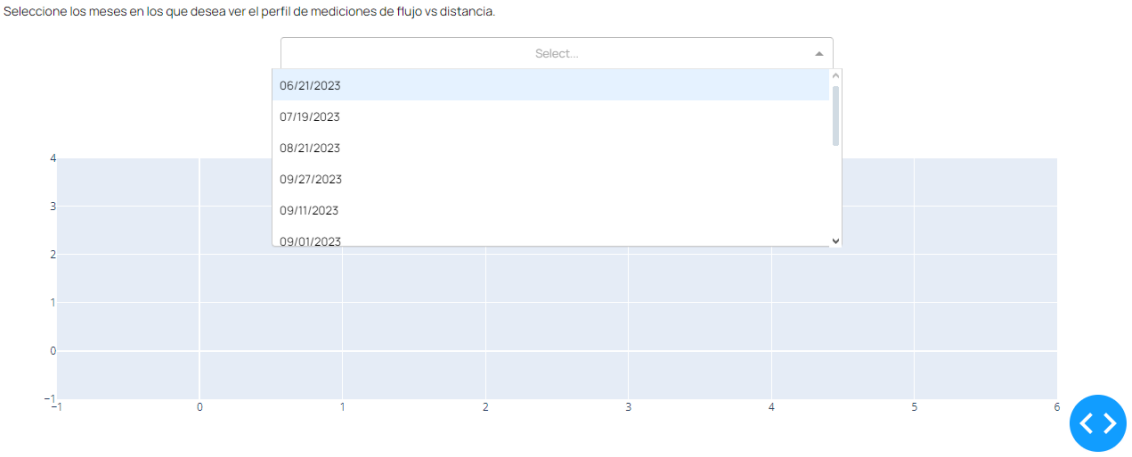


Ilustración 87 Gráfico comparativo de flujo de CO_2 vs punto de medición

La única desventaja de este proceso es que es necesario realizarlo manualmente para cada punto de medición para obtener una comparación de perfiles en diferentes fechas. Sin embargo el procedimiento es sencillo, y brinda respaldo a las mediciones realizadas en campo.

5.3 Análisis manual.

Por otro lado si el usuario quiere realizar un análisis más complejo de las mediciones puede obtener fácilmente la tabla de datos de la Hoja 1 de la Google Sheets utilizando el enlace Datos CO2 meter 1, dichos datos aparecen como la Ilustración 45.

NOTA: SE ADVIERTE QUE EN LAS COLUMNAS EL USUARIO DEBE ABSTENERSE A MODIFICAR EL TIPO DE DATOS DEBIDO A QUE SE PUEDE CREAR UN CONFLICTO EN LA LECTURA POR PARTE DE LA DASHBOARD O EL INGRESO DE NUEVOS DATOS DESDE EL CPU DE LA ESTACIÓN MÓVIL.

Capítulo 6

6 Resultados de mediciones.

Para comprobar el desempeño del equipo se realizaron diferentes pruebas a lo largo del proyecto, de tal manera de probar cada nueva funcionalidad implementada iniciando primero con reproducir el sistema inicial cambiando el microcontrolador, luego incluyendo la conectividad GPRS para enviar datos en tiempo real y luego por paquetes de datos. Por último se corroboró el funcionamiento de la Dashboard para la visualización de datos y crear un registro histórico de las mediciones.

6.1 Comprobación de cobertura en Volcán de Santa Ana.

Una de las primeras pruebas de campo realizadas fue examinar la cobertura de la señal en la zona del volcán de Santa Ana. Para ello se utilizó una aplicación Android (Network Cell Info) de terceros que evalúan el nivel de cobertura ofrecen diferentes compañías de celular y mapear un recorrido.

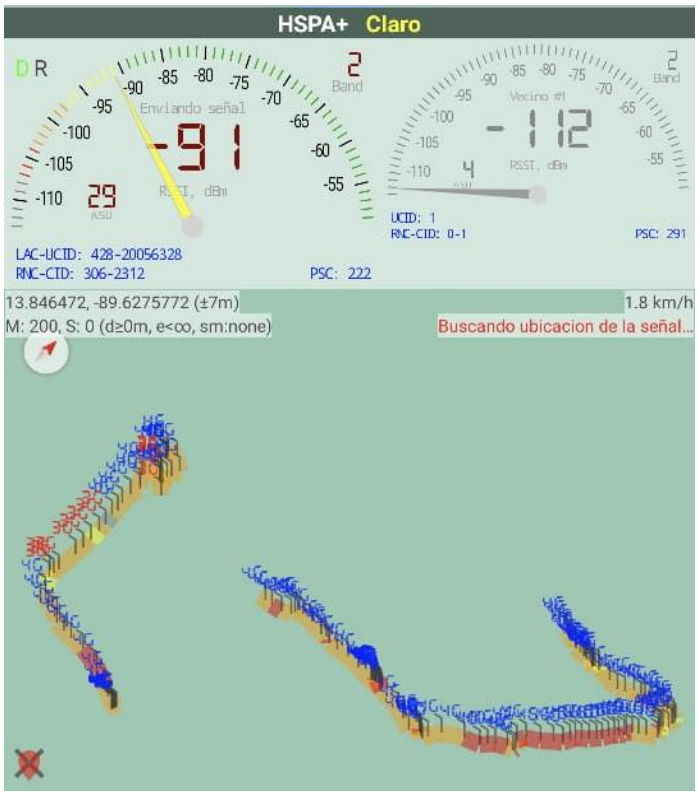


Ilustración 88 Mapa de cobertura de señal en el recorrido hacia la cima del volcán

Este tramo corresponde al nivel de cobertura Claro del recorrido hasta la cima, se observa sobre todo señal 4G y 3G, aunque hay zonas donde no se obtiene cobertura en áreas de vegetación abundante. La aplicación muestra tramos en rojo naranjas y amarillos, lo cual indica que, aunque exista cobertura apenas existe un nivel de -105 dBm.

Sin embargo en la cima y el área de medición de CO_2 se encontró lo siguiente:

Ilustración 89 Cobertura de señal en la línea de medición (Arriaza Carrillo & Aguilar Vega, 2022) en el volcán de Santa Ana

Ambas imágenes muestran el corrido de medición que es el área de interés. En una primera prueba que corresponde a la imagen de la derecha se observa cobertura 2G y 3G. La imagen izquierda es una segunda muestra y se observa 4G y que la cobertura en esa zona es aceptable para las diferentes bandas, aunque esto no quiere decir que es totalmente estable, ya que a lo largo de la toma de datos la cobertura desaparecía por momentos.

Por otro lado, también se evaluó la conexión de los módulos SIM900 y SIM800L para enviar datos en tiempo real, y si bien el resultado fue exitoso, se dijo en capítulos anteriores que no era conveniente enviar datos así, si no por paquetes.

Ilustración 90 Medición en campo en la línea en el volcán de Santa Ana

6.2 Implementación y medición en Cerro Pacho Coatepeque.

Una zona de interés para los geofísicos es en el Cerro Pacho en Coatepeque:

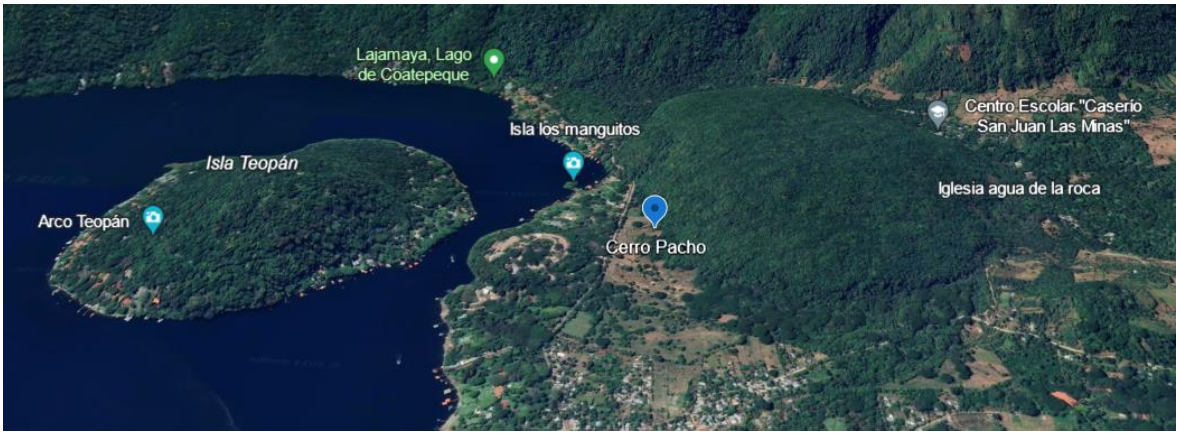


Ilustración 91 Punto de medición en Cerro Pacho, Coatepeque. (Earth, 2024)

Para el inicio de mediciones se sigue el procedimiento explicado en el Capítulo 5 con la estación móvil, en este caso antes de abordar la zona se debe encender el equipo para que el módulo GPS localice los satélites, puesto que hay zonas en que se tarda en mostrar la geolocalización.

Ilustración 92 Fotografías de la implementación del equipo.

6.2.1 Resultados.

6.2.1.1 Resultados en la aplicación móvil.

Por parte de la aplicación móvil se tienen los siguientes resultados para el cálculo del flujo para el día 20 de febrero del 2024:

	System Date	FluxCO_2	R^2
1	TueFeb20	0.255	0.97
2	TueFeb20	0.267	0.979
3	TueFeb20	0.305	0.974
3	TueFeb20	0.231	0.111
3	TueFeb20	0.28	0.947
4	TueFeb20	1.046	0.985
4	TueFeb20	1.057	0.986
5	TueFeb20	91.963	0.875

5	TueFeb20	45.508	0.93
6	TueFeb20	2.945	0.981
7	TueFeb20	0.118	0.887
7	TueFeb20	0.052	0.109
8	TueFeb20	0.292	0.986
9	TueFeb20	-0.282	0.195
9	TueFeb20	-0.338	0.208
9	TueFeb20	0.08	0.341
10	TueFeb20	0.252	0.889
11	TueFeb20	0.068	0.862
11	TueFeb20	0.067	0.805
12	TueFeb20	0.199	0.98
12	TueFeb20	0.228	0.968
13	TueFeb20	0.058	0.863
14	TueFeb20	0.621	0.99
14	TueFeb20	0.641	0.987
15	TueFeb20	0.055	0.813
15	TueFeb20	0.062	0.88
20	TueFeb20	2.227	0.994
21	TueFeb20	0.272	0.979
21	TueFeb20	0.2	0.623

Tabla 18 Resultados del reporte del cálculo del flujo de CO_2 en los diferentes puntos

Se observa que el valor del flujo se dispara en el punto 5 en dos mediciones, identificando una zona anómala con un flujo de CO_2 45 a 90 ppm/s, ahora si se presenta de forma gráfica estos resultados se tiene lo siguiente:

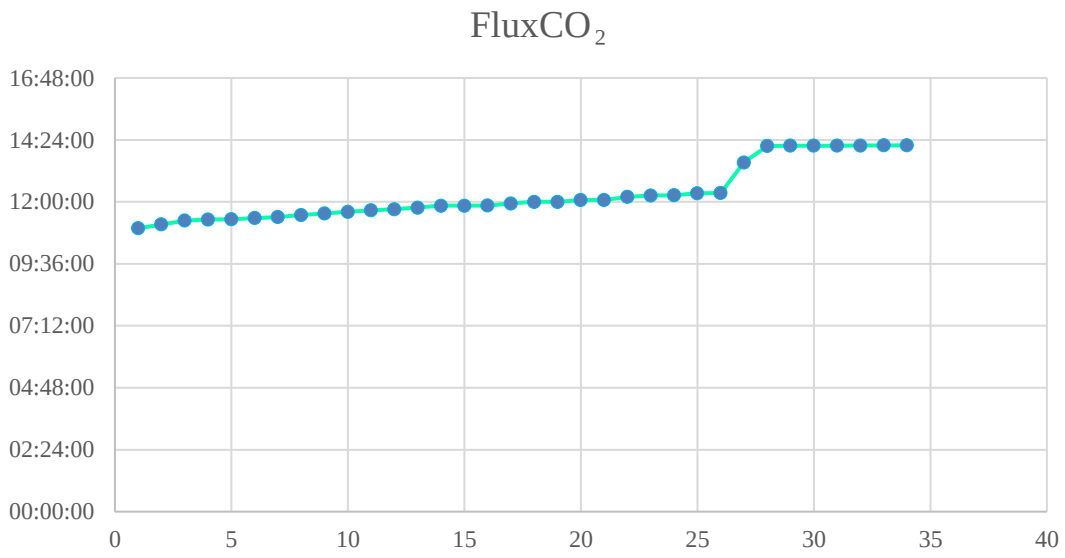


Ilustración 93 Gráfico de los resultados de Flujo de CO_2 en los diferentes puntos

6.2.1.2 Resultados registrados en la hoja de cálculo y Dashboard.

En cada medición se ha registrado un paquete de datos en la Hoja 1 de la manera siguiente:

Fecha	Hora	Temperatura	Humedad	CO2	Punto	Latitud	Longitud
2/20/2024	11:05:00	32.4	43	403.01	1	13.8442790	-89.5718700
2/20/2024	11:05:01	32.4	43.1	408.29	1	13.8442790	-89.5718700
2/20/2024	11:05:02	32.4	43.1	405.35	1	13.8442790	-89.5718700
2/20/2024	11:05:03	32.4	43.1	405.34	1	13.8442790	-89.5718700
2/20/2024	11:05:04	32.4	43.1	431.04	1	13.8442790	-89.5718700
2/20/2024	11:05:05	32.4	43.1	444.53	1	13.8442790	-89.5718700
2/20/2024	11:05:06	32.4	43.1	413.09	1	13.8442790	-89.5718700
2/20/2024	11:05:07	32.4	43.1	399.83	1	13.8442790	-89.5718700
2/20/2024	11:05:08	32.4	43.1	397.65	1	13.8442790	-89.5718700
2/20/2024	11:05:09	32.4	43.1	405.66	1	13.8442790	-89.5718700
2/20/2024	11:05:10	32.4	43.1	401.23	1	13.8442790	-89.5718700
2/20/2024	11:05:11	32.4	43.2	404.59	1	13.8442790	-89.5718700
2/20/2024	11:05:12	32.4	43.2	413.51	1	13.8442790	-89.5718700
2/20/2024	11:05:13	32.4	43.2	408.2	1	13.8442790	-89.5718700
2/20/2024	11:05:14	32.4	43.2	403.3	1	13.8442790	-89.5718700
2/20/2024	11:05:15	32.4	43.2	401.87	1	13.8442790	-89.5718700
2/20/2024	11:05:16	32.4	43.2	403.3	1	13.8442790	-89.5718700
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.

Tabla 19 Tabla registrada en la Hoja 1

En la Dashboard el resultado según el registro de la hoja de cálculo es el siguiente:

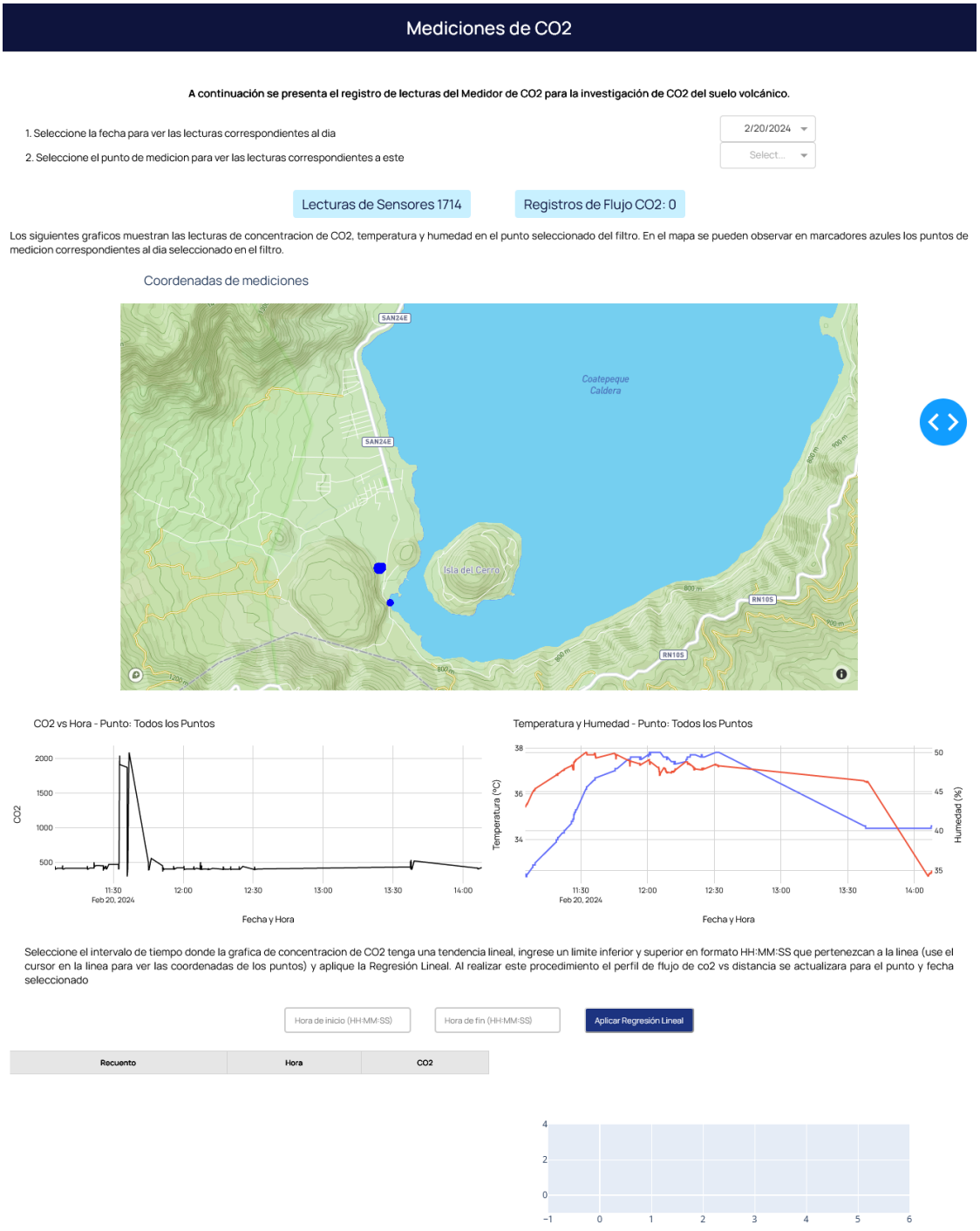


Ilustración 94 Resultado de la lectura de la Hoja 1 en la dashboard versión de escritorio

Mediciones de CO2

A continuación se presenta el registro de lecturas del Medidor de CO2 para la investigación de CO2 del suelo volcánico.

1. Seleccione la fecha para ver las lecturas correspondientes al dia — 2/20/2024

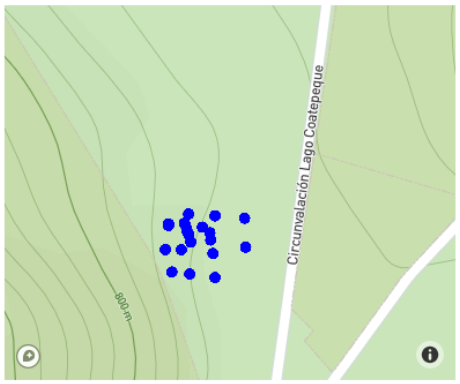
2. Seleccione el punto de medicion para ver las lecturas correspondientes a este — Select...

Lecturas de Sensores 1714

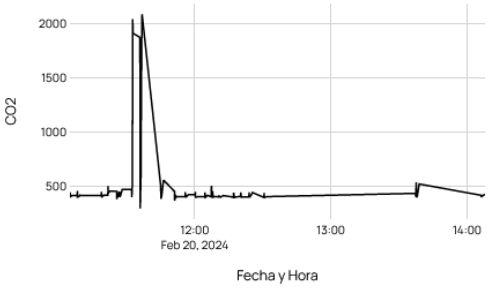
Registros de Flujo CO2: 0

Los siguientes graficos muestran las lecturas de conc[illegible]ion de CO2, temperatura y humedad en el punto seleccion[illegible]ltro. En el mapa se pueden observar en marcadores azul[illegible]ntos de medicion correspondientes al dia seleccionado en el filtro.

Coordenadas de mediciones



CO2 vs Hora - Punto: Todos los Puntos



Temperatura y Humedad - Punto: Todos los Puntos

Ilustración 95 Resultado de la lectura de la Hoja 1 en la dashboard versión teléfono móvil.

Estas dos capturas de pantalla muestran el comportamiento de la dashboard leyendo los datos registrados en el Hoja 1, siguiendo el procedimiento del Capítulo 5.2 para el cálculo del flujo de CO_2 en la dashboard se genera el siguiente gráfico:

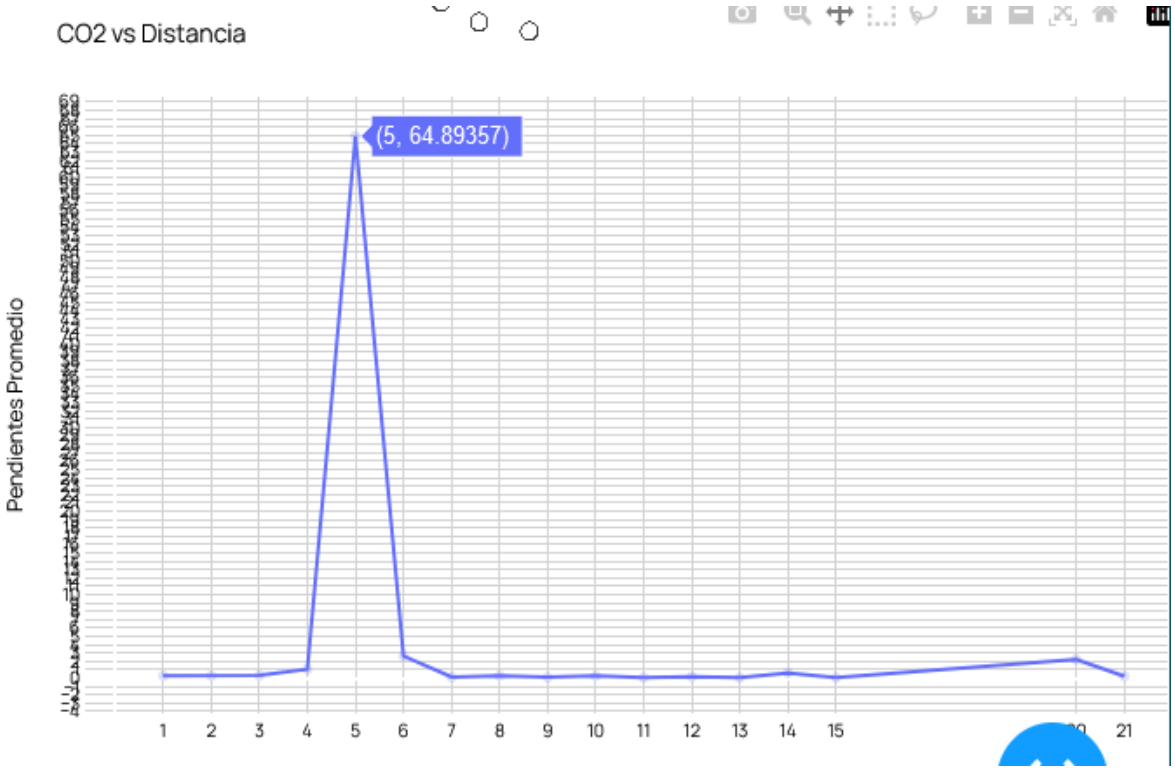


Ilustración 96 Gráfico de flujo de CO_2 vs punto de medición

Este gráfico está dado gracias a la Hoja 2 que contiene los siguientes resultados:

Fecha Mes	Pendientes	Punto Medicion	Latitud	Longitud
2/20/2024	0.2781383	1	13.8442790	-89.5718700
2/20/2024	0.2736824	2	13.8442900	-89.5718690
2/20/2024	0.3115695	3	13.8442960	-89.5717540
2/20/2024	0.3081124	3	13.8442960	-89.5717540
2/20/2024	1.0350460	4	13.8442580	-89.5717390
2/20/2024	1.0683818	4	13.8442580	-89.5717390
2/20/2024	87.5597193	5	13.8442170	-89.5717240
2/20/2024	42.2274281	5	13.8442170	-89.5717240
2/20/2024	2.6224302	6	13.8441080	-89.5717770

2/20/2024	0.1152326	7	13.8439380	-89.5717160
2/20/2024	0.2944354	8	13.8439520	-89.5718460
2/20/2024	0.0871760	9	13.8439130	-89.5715330
2/20/2024	0.2570070	10	13.8441270	-89.5713120
2/20/2024	0.0646053	11	13.8443310	-89.5713200
2/20/2024	0.1574825	12	13.8443470	-89.5715330
2/20/2024	0.0546180	13	13.8441770	-89.5715640
2/20/2024	0.0561455	13	13.8441770	-89.5715640
2/20/2024	0.6128780	14	13.8442680	-89.5716250
2/20/2024	0.0533779	15	13.8440820	-89.5715480
2/20/2024	2.2224190	20	13.8405320	-89.5704960
2/20/2024	0.2208243	21	13.8441100	-89.5718920

Tabla 20 Registro de flujo de CO_2 en la Hoja 2

La comparación entre la Tabla 20 y Tabla 18 no varía mucho, la diferencia radica en los intervalos tomados para realizar el cálculo de la regresión lineal, unos han sido hechos por la app en campo y otra en la revisión de los datos en la Dashboard.

Aquí se muestra de forma gráfica la comparación de resultados tanto en escala normal.

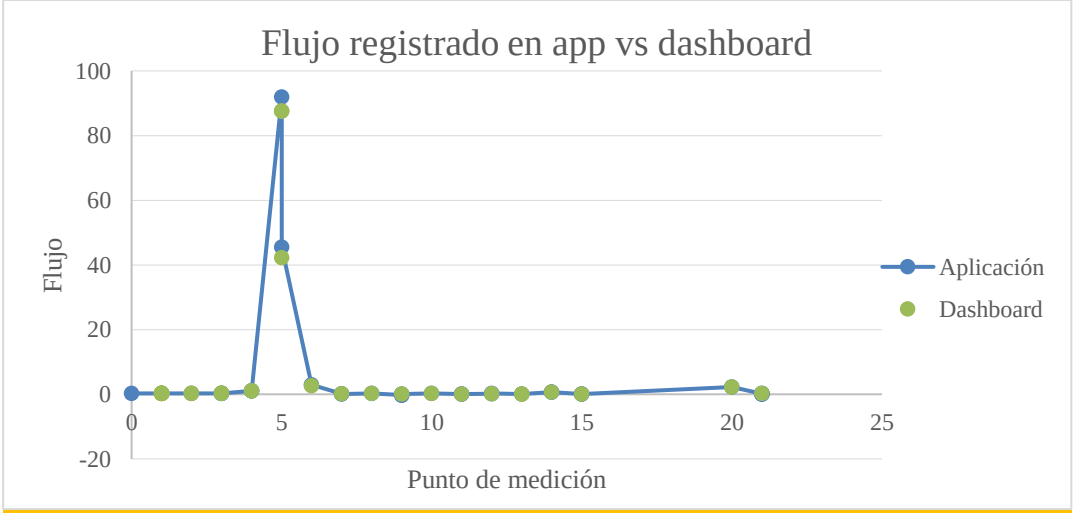


Ilustración 97 Comparativa de la Tabla 20 y Tabla 18

6.2.1.3 Ejemplo de Cálculo de Regresión Lineal en la Dashboard.

En Tabla 19 para el punto 1 se tiene la siguiente gráfica:

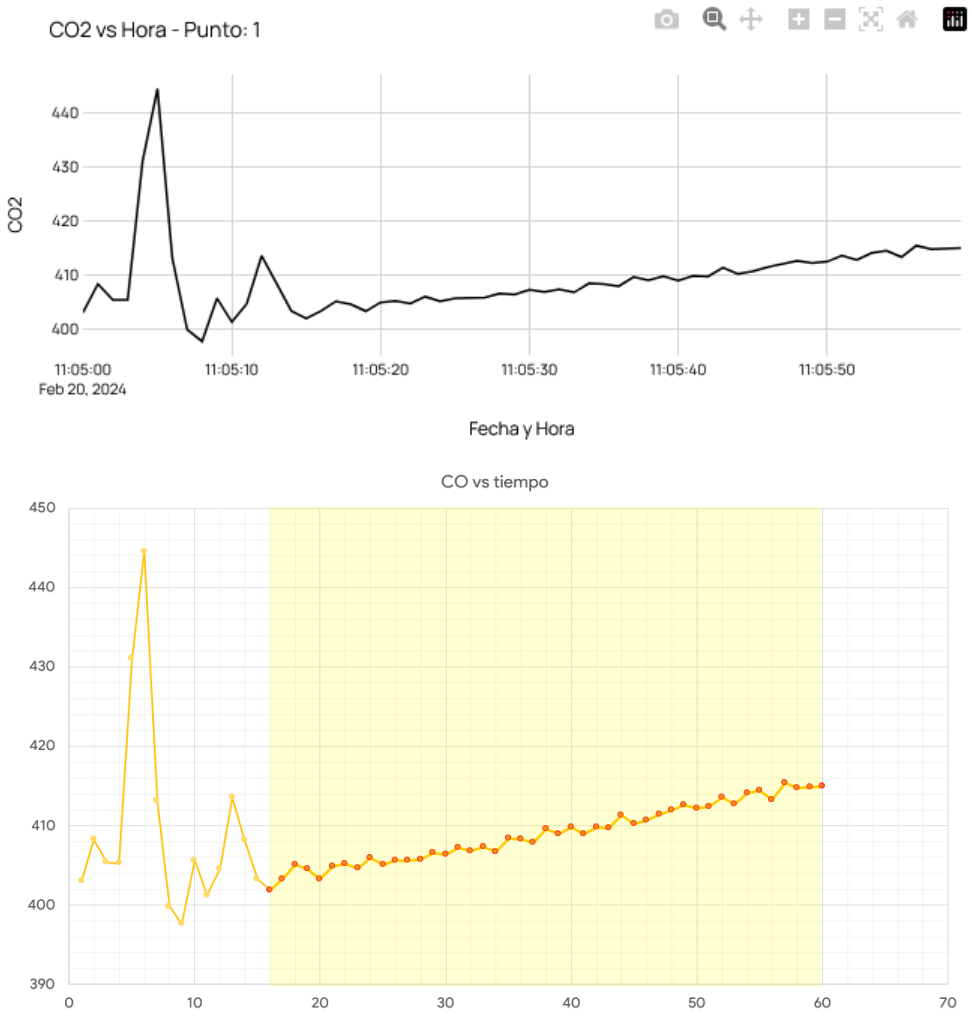


Ilustración 98 Resultados del punto 1.

Se observa el patrón lineal y se toman los datos en ese intervalo, obteniéndose las siguientes matrices:

$$\mathbf{Y} = \begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \\ \vdots & \vdots \end{bmatrix}$$

La fórmula indica lo siguiente para obtener los estimadores o coeficientes de regresión lineal:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \begin{bmatrix} 0.092 & -0.003 \\ -0.003 & 0 \end{bmatrix} \qquad \mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 18\,403.76 \\ 425\,397.55 \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\beta = \begin{bmatrix} 0.092 & -0.003 \\ -0.003 & 0 \end{bmatrix}\begin{bmatrix} 18\,403.76 \\ 425\,397.55 \end{bmatrix} = \begin{bmatrix} 402.575 \\ 0.278 \end{bmatrix}$$

Por lo tanto, el modelo para ese tramo es:

$$y = \beta_0 + \beta_1 x$$
$$y = 402.575 + 0.278x$$

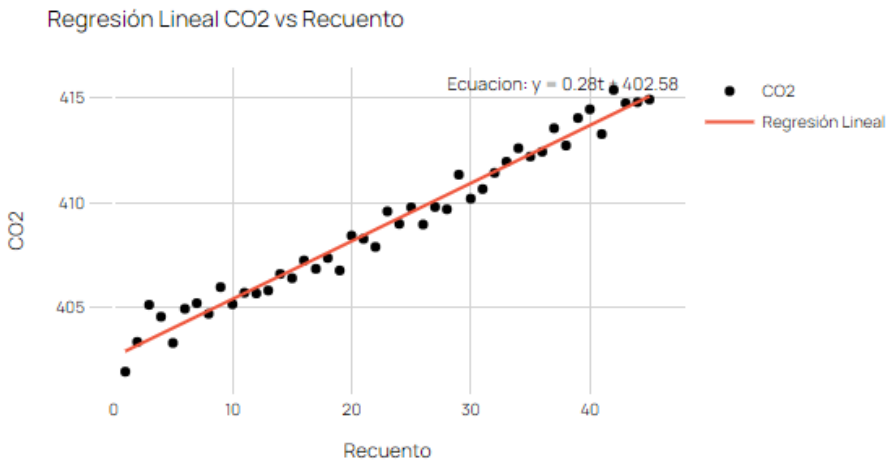


Ilustración 99 Comprobación de la regresión lineal en la Dashboard

Con ello se comprueba el desempeño matemático de las bibliotecas utilizadas para obtener el modelo lineal o predicción lineal y con respecto al R^2 éste también utiliza el método matricial (Kutner, Nachtsheim, Neter, & Li, 2004):

$$R^2 = 1 - \frac{SSE}{SSTO}$$

$$SSE = (\mathbf{Y} - \hat{\mathbf{Y}})^T (\mathbf{Y} - \hat{\mathbf{Y}})$$

$$SSE = \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} - \left[\beta_0 + \beta_1 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ \vdots \end{bmatrix} \right] \right)^T \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} - \left[\beta_0 + \beta_1 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ \vdots \end{bmatrix} \right] \right)$$

$$SSE = \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} - \left[402.575 + 0.278 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ \vdots \end{bmatrix} \right] \right)^T \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} + \left[402.575 + 0.278 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ \vdots \end{bmatrix} \right] \right) = 16.487$$

$$\bar{\mathbf{Y}} = \frac{1}{n} \sum \mathbf{Y} = 408.972$$

$$SSTO = (\mathbf{Y} - \bar{\mathbf{Y}})^T (\mathbf{Y} - \bar{\mathbf{Y}})$$

$$SSTO = \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} - 408.972 \right)^T \left(\begin{bmatrix} 401.87 \\ 403.3 \\ 405.08 \\ 404.51 \\ 403.25 \\ 404.89 \\ 405.16 \\ 404.65 \\ 405.93 \\ \vdots \end{bmatrix} - 408.972 \right) = 603.6562$$

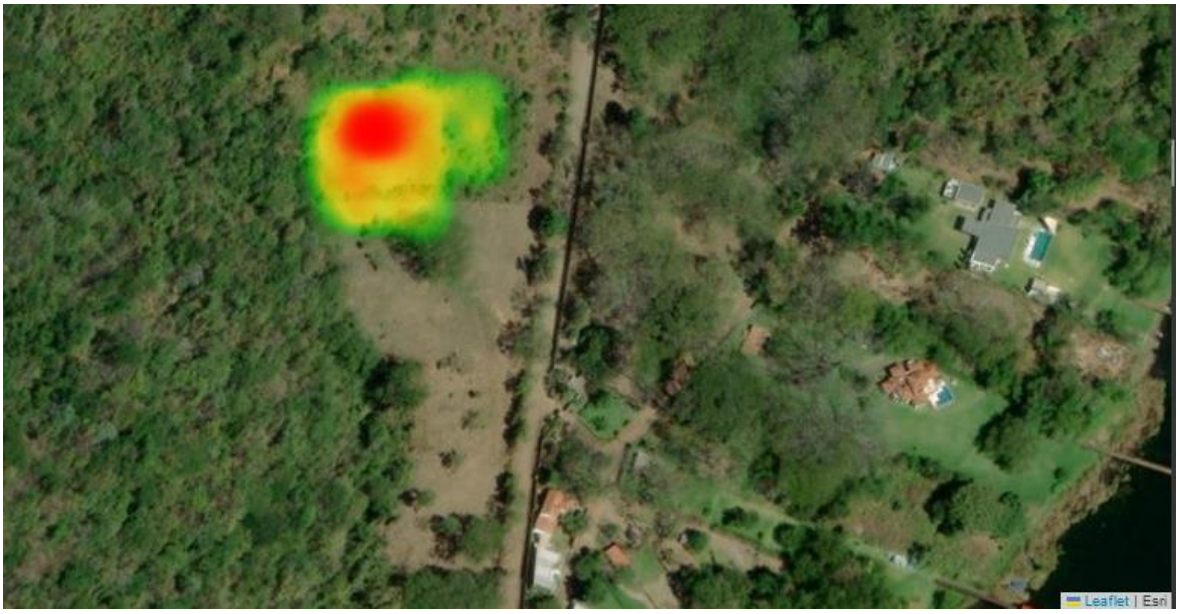
Finalmente se obtiene un R^2 de:

$$R^2 = 1 - \frac{16.487}{603.6562} = 0.97268$$

Lo que significa que el comportamiento o la estimación del modelo es aceptable.

6.2.2 Análisis manual.

Además, la tabla de la Hoja 2 muestra las coordenadas del punto y por lo que se pueden hacer estudios más complejos como mapas de calor como los siguientes (Pérez, 2023):



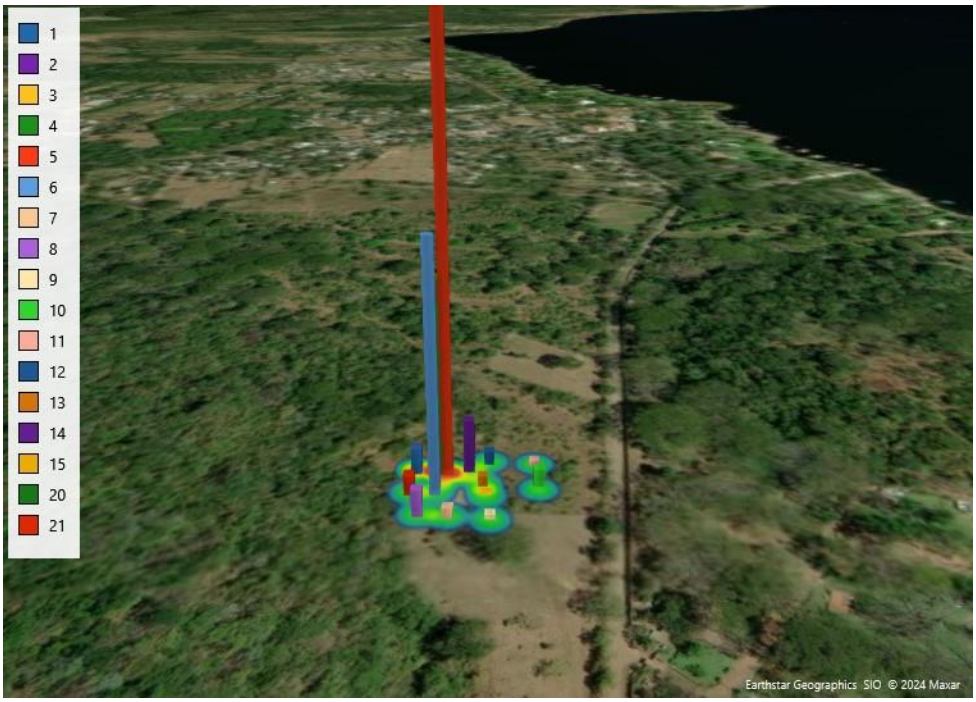


Ilustración 100 Mapas de calor del flujo en las coordenadas de los puntos medidos

Estos mapas han sido generados en Python y en Excel respectivamente, representan la comparativa de flujo de CO_2 en los puntos medidos. Y estos pueden ser comparados con los valores de Self-potential y Temperatura. Los cuales de igual manera se pueden visualizar en un mapa de calor:

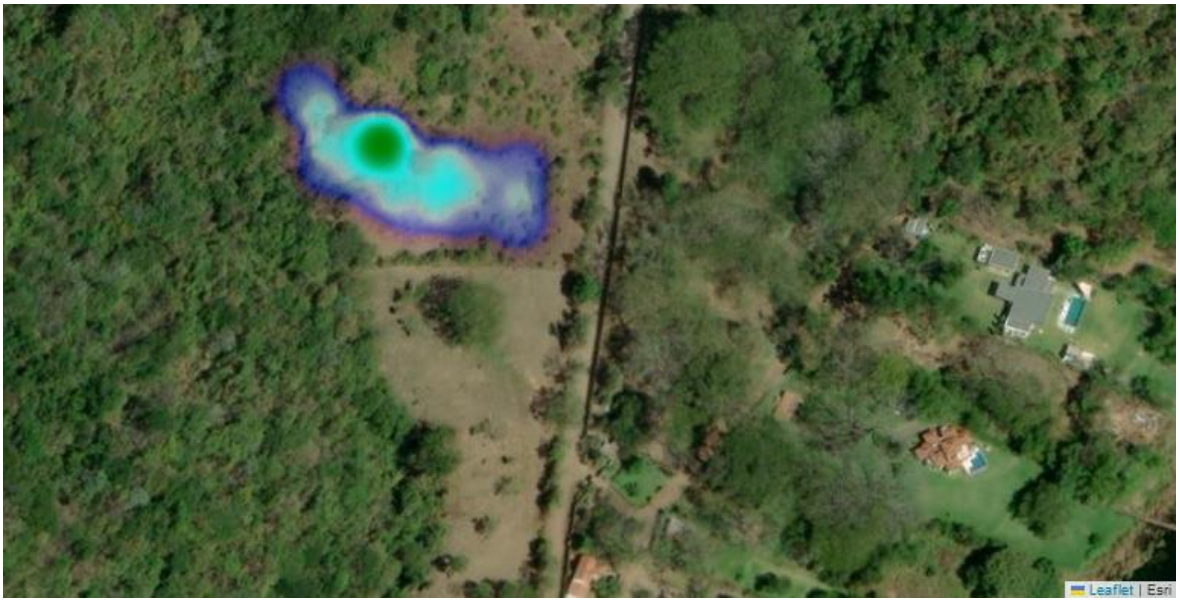


Ilustración 101 Mapa de calor de medición de Potencial Espontáneo

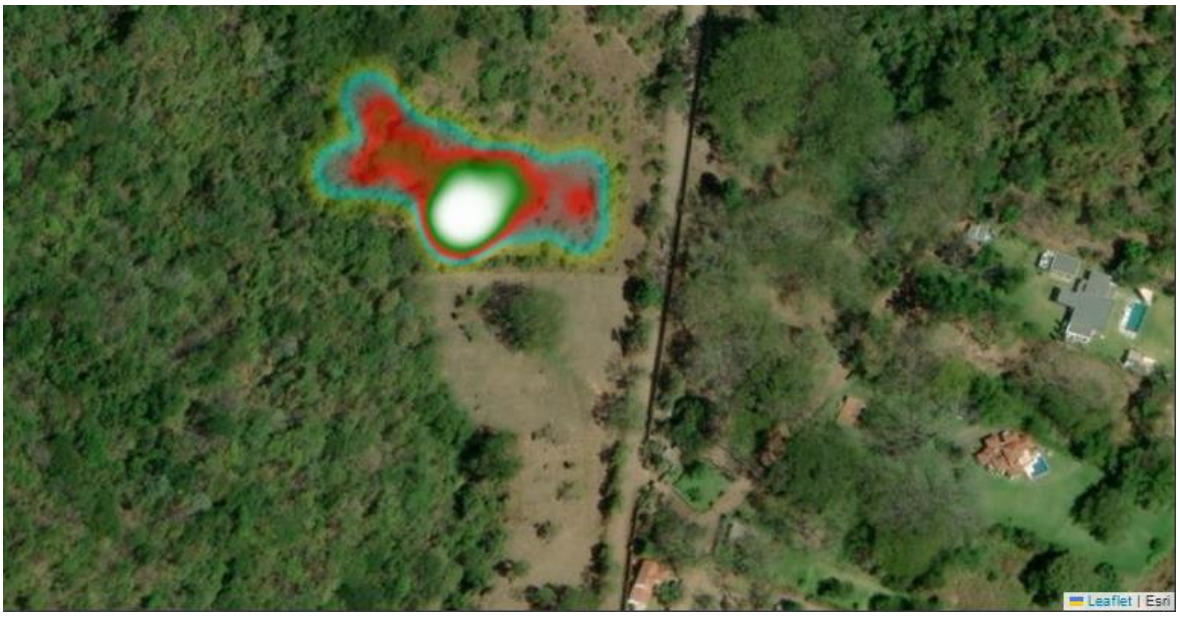


Ilustración 102 Mapa de calor de medición de Temperatura

Se observa que en la zona anómala hay un incremento del valor de temperatura, así como el de flujo de CO_2. Si se gráfica en forma de barras el comportamiento de los datos registrados, de resistencia, SP y temperatura se tiene lo siguiente:

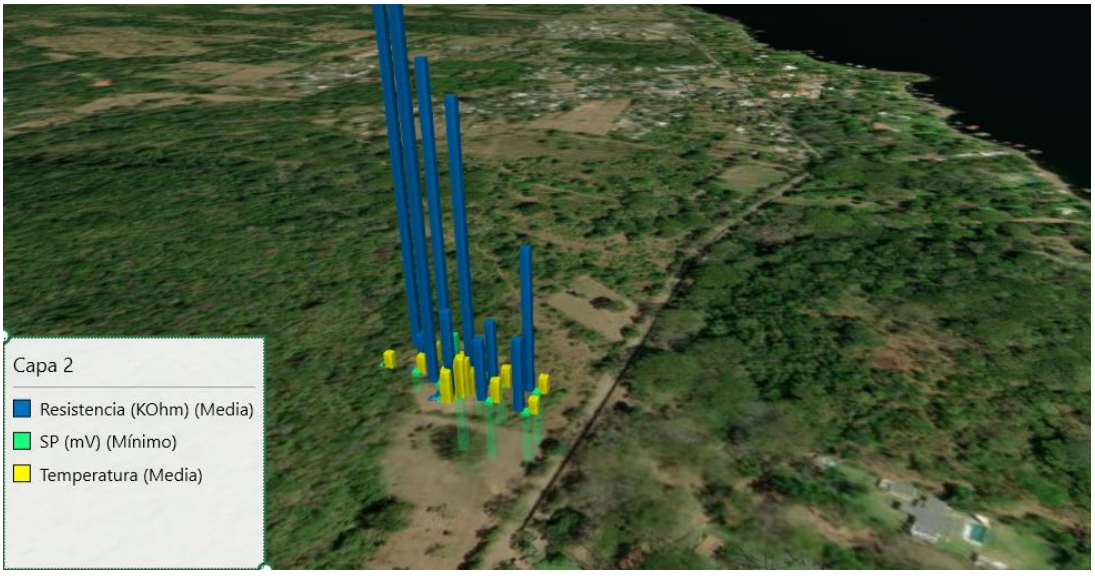


Ilustración 103 Barras superpuestas para el valor de SP, Temperatura y Resistencia

Y superponiendo todos los mapas de calor:

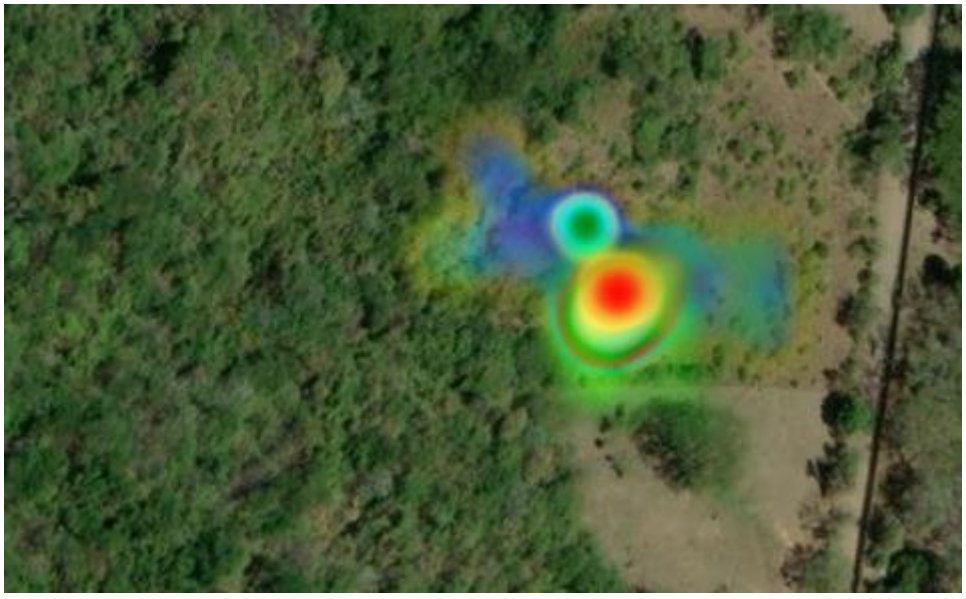


Ilustración 104 Superposición de los mapas de calor, CO_2, Temperatura y SP

El mapa verde, amarillo y rojo corresponde al flujo de CO_2; el mapa azul y verde el SP; y el mapa rojo, blanco y verde, Temperatura. De igual manera como se muestra en la Ilustración 90 de (Arriaza Carrillo & Aguilar Vega, 2022) hay una relación similar de ese gráfico con estos mapas de calor en el Cerro Pacho.

Capítulo 7

7 Conclusiones y recomendaciones.

7.1 Conclusiones.

- La selección del ESP32 como microcontrolador para el CPU representa un avance significativo debido a su mayor potencia de procesamiento, mayor capacidad de memoria y eficiencia energética. Este microcontrolador no solo facilita la programación mediante Arduino, una base inicial crucial para el proyecto, sino que también soporta una variedad de lenguajes de programación, ampliando las posibilidades de desarrollo futuro y la integración con otros sistemas IoT.
- Con el ESP32 por su Bluetooth integrado se ahorra la utilización del módulo HC-06 que dificultaba agregar líneas de depuración al programa 1, de esta manera ahora se tiene un mejor control sobre los mensajes enviados a la aplicación.
- La elección de la placa TTGO T-Call SIM800L se destaca como una decisión estratégica debido a su módulo GSM integrado, lo que elimina la necesidad de componentes adicionales para la conectividad celular. Este diseño enfocado en proyectos IoT remotos permite una recopilación y transmisión de datos más eficiente en entornos sin acceso a Wi-Fi, optimizando el sistema para aplicaciones de campo.
- La implementación de un sistema de almacenamiento temporal y envío de datos en paquetes responde a los desafíos presentados por la rápida generación de datos de CO_2 y la inestabilidad de la conexión GPRS en áreas remotas. Este enfoque mejora significativamente la confiabilidad de la transmisión de datos, asegurando que la información se envíe de manera completa y eficiente.
- La decisión de utilizar peticiones HTTP POST hacia la implementación de AppScript para el envío de datos a Google Sheets aborda eficazmente las limitaciones de las peticiones HTTP GET, que requerían envíos individuales para cada fila de datos, resultando en procesos lentos

y la pérdida potencial de datos. Las peticiones POST permiten una transmisión más rápida y confiable de grandes volúmenes de datos.

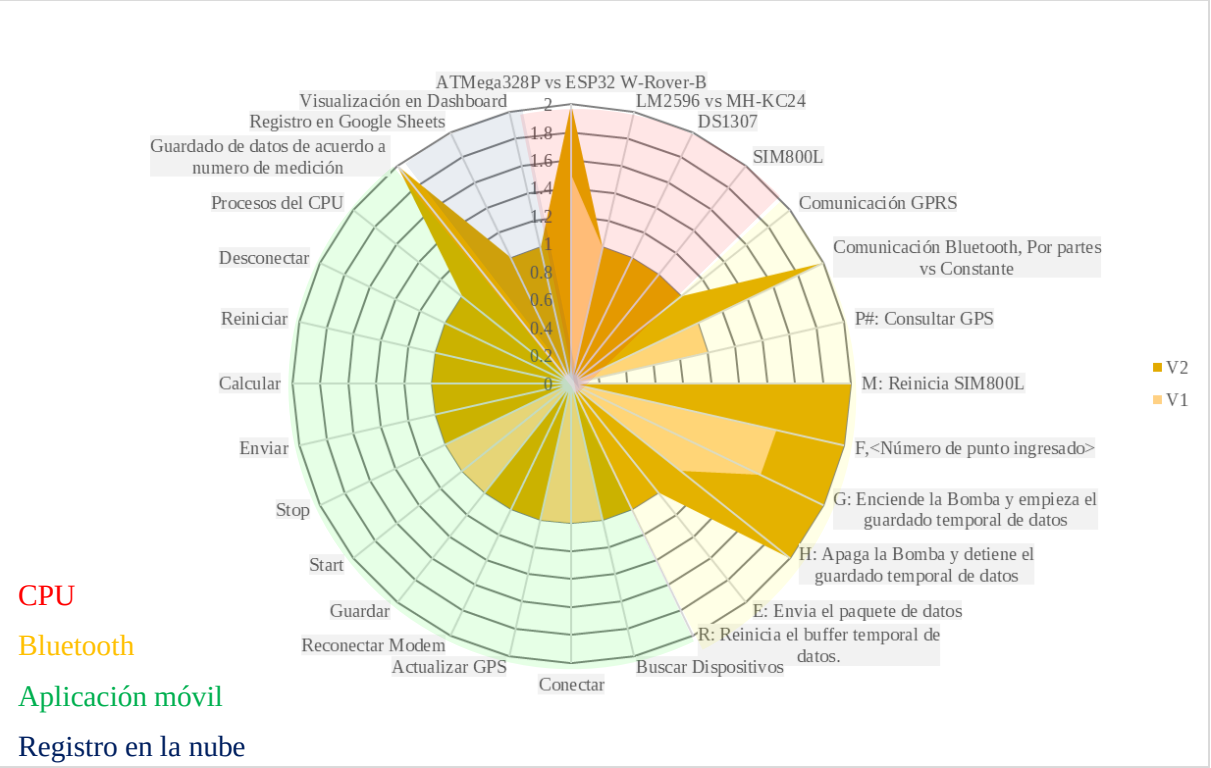
- La optimización de la comunicación Bluetooth en la aplicación móvil mejora la interacción con el CPU, permitiendo pausar y reanudar la recepción de datos para nuevas mediciones sin la necesidad de cerrar la aplicación. Esta mejora contribuye a una mayor eficiencia en el proceso de medición y una mejor experiencia de usuario.
- El desarrollo de una Dashboard personalizada para el análisis y visualización de datos permite realizar cálculos complejos y presentar información de manera intuitiva. Esta herramienta se convierte en un componente esencial para la interpretación de los datos recopilados, ofreciendo a los investigadores y usuarios una plataforma potente para el análisis de las emisiones de CO_2.
- La selección de Render como plataforma para el despliegue de la aplicación web facilita la obtención de un dominio gratuito y ofrece un entorno robusto para la operación de la Dashboard. Esta decisión recalca la importancia de una infraestructura de soporte accesible y confiable para la difusión de los resultados del proyecto.
- La comparación de los cálculos de flujo de CO_2 realizados automáticamente por la aplicación y la Dashboard con aquellos hechos manualmente confirma la precisión y fiabilidad de las herramientas desarrolladas. Este nivel de validación es fundamental para asegurar la confianza en los resultados obtenidos, destacando la exactitud del sistema en la medición del flujo de CO_2.

7.2 Recomendaciones.

- Se recalca que para el uso de la estación móvil, se recomienda encender el equipo un tiempo prudencial antes de realizar la primera medición alrededor de 10 minutos o 20 minutos para que el módulo GPS sintonice los satélites y obtener geolocalización.
- Se sugiere utilizar la secuencia de uso en la Guía de utilización del ecosistema de investigación. Tanto para la aplicación móvil y la aplicación web.
- El tipo de dato registrado en la hoja de cálculo no debe ser cambiado para evitar conflictos en la escritura de datos por parte del CPU, y la lectura de la dashboard.
- Para la secuencia y mantenimiento de la estación móvil, se busca automatizar lo más posible la interacción manual. Automatizar la dashboard para el cálculo del flujo sin la necesidad de seleccionar los intervalos con comportamiento lineal.

- Reforzar la robustez del CPU y los componentes de la estación móvil, añadiendo más variables y módulos.
- Actualizar la aplicación móvil, mostrando las coordenadas actuales. Y mejorar la intuitividad de ésta hacia los usuarios.

7.3 Resultado comparativo de ambas versiones de la Estación Móvil.



En el gráfico radial se observa las funcionalidades de las diferentes áreas de la estación móvil, en amarillo claro se aprecia la versión 1 y en amarillo oscuro la versión 2 , de tal manera que se secciona el gráfico radial con lo relacionado al CPU (rojo), Bluetooth (amarillo), Aplicación móvil (Verde) y el registro en la nube (Azul). En el área del CPU se compara el microcontrolador y los módulos agregados, en el área de la comunicación se agregan funcionalidades e indicaciones al CPU, a comparación de la V1, ahora se añaden más tareas y se obtiene un desempeño mayor. En cuanto a la aplicación móvil, se muestran los botones agregados y el cambio del método de guardado de los archivos en el almacenamiento interno del teléfono. Por último solo la versión 2 tiene registro en la nube. Todo este gráfico radial comparativo concluye las mejoras principales agragadas en la versión 2 con respecto a la versión 1.

Referencias

Arriaza Carrillo, C. A., & Aguilar Vega, E. O. (2022). *Seguimiento y mapeo de línea base de emisión de CO2 en suelo del volcán Santa Ana.* El Salvador: Trabajo de Graduación, Escuela de Ingeniería Eléctrica.

Atif, M., Muralidharan, S., Ko, H., & Yoo, B. (2020). ESP32 functional block diagram. En M. Atif, S. Muralidharan, H. Ko, & B. Yoo, *Wi-ESP: una herramienta para el sensado Wi-Fi sin dispositivos basado en CSI (DFWS) Vol. 7*, (pág. 5). Journal of Computational Design and Engineering.

Atmel. (2018). *Atmel* . Obtenido de 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash.

Biosciences, L.-C. (2023). *LI-850 CO2/H2O Analyzer Specifications.*

Bruce, J. (2021). *MakeUseOf.* Obtenido de What You Should Know About the ATmega328P (Arduino): https://www.makeuseof.com/what-you-should-know-about-atmega328p-arduino/

Components101. (2021). *Components101* . Obtenido de ATmega328P Pinout, Features & Datasheet: https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet

DataScientest. (25 de Mayo de 2022). *¿Qué es un DataFrame?* Obtenido de DataScientest: https://datascientest.com/es/que-es-un-dataframe

Developers, K. (2016). *Getting Started in KiCad.* Obtenido de KiCad Documentation.: https://docs.kicad.org/4.0/es/getting_started_in_kicad/getting_started_in_kicad.pdf

Digest, C. (2018). *Circuit Digest.* Obtenido de Interfacing DS1307 RTC Module with Arduino: https://circuitdigest.com/microcontroller-projects/interfacing-ds1307-rtc-module-with-arduino

Earth, G. (2024). *Vista de Cerro Pacho, Coatepeque, Santa Ana, El Salvador*. Obtenido de Coordenadas (13,8442393;-89,5719923): https://earth.google.com/web/search/13.841529475702302,+-89.57892972709202/@13.84221085,-89.57895065,931.12313893a,3100.89041119d,35y,0h,0t,0r/data=CmoaQBI6GQw9YvTcritAIbcLKC8NZVbAKiYxMy44NDE1Mjk0NzU3MDIzMDIsIC04OS41Nzg5Mjk3MjcwOTIwMhgCIAEiJgokCRRtUrx1ujN

Finizola, A., Sortino, F., Lénat, J. F., Valenza, M., & Boschi, E. (2003). *Soil CO2 flux measurements in volcanic and geothermal areas. Journal of Volcanology and Geothermal Research.*

Frondini, F. C. (2004). *Soil CO2 flux measurements in volcanic and geothermal areas. Applied Geochemistry, .*

Google. (s.f.). *Python Quickstart | Google Sheets API*. Obtenido de Google Workspace Developers: https://developers.google.com/sheets/api/quickstart/python?hl=es-419

Hub, D. (2023). *Datasheet Hub*. Obtenido de MAX232 RS232 to TTL Converter Module: https://datasheethub.com/max232-rs232-to-ttl-converter-module/

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). *Applied Linear Statistical Models*. McGraw-Hill/Irwin.

Lab, M. (2023). *Microcontrollers Lab*. Obtenido de NEO-6M GPS Module with ESP8266 NodeMCU to Track Location on Google Maps: https://microcontrollerslab.com/neo-6m-gps-module-esp8266-nodemcu-track-location-google-maps/

Mechatronics, N. (2016). *Tutorial: Sensor de temperatura y humedad DHT11 y DHT22*. Obtenido de Naylamp Mechatronics Blog: https://naylampmechatronics.com/blog/40_tutorial-sensor-de-temperatura-y-humedad-dht11-y-dht22.html

Mechatronics, N. (2023). *Naylamp Mechatronics*. Obtenido de Tutorial Básico de Uso del Módulo Bluetooth HC-06 y HC-05: https://naylampmechatronics.com/blog/12_tutorial-basico-de-uso-del-modulo-bluetooth-hc-06-y-hc-05.html

Merati, M. (2023). *Electropeak*. Obtenido de Sending Data from ESP32 or ESP8266 to Google Sheets [2 Methods]: https://electropeak.com/learn/sending-data-from-esp32-or-esp8266-to-google-sheets/

MH-KC24. (2023). Obtenido de MH-KC24 : https://www.javanelec.com/CustomAjax/GetAppDocument/106d2abd-4f45-4a48-91ef-f92f9a52c1a8?type=1&inlineName=True

Microchip. (2023). *Microchip*. Obtenido de ATMEGA328P: https://www.microchip.com/en-us/product/ATMEGA328P

Morich, K. (2023). *GitHub*. Obtenido de Simple Bluetooth Terminal: https://github.com/kai-morich/SimpleBluetoothTerminal

Orellana, R. (20 de Abril de 2021). *digitaltrends*. Obtenido de La evolución de Android: los cambios del sistema operativo de Google: https://es.digitaltrends.com/android/evolucion-de-android/

Pandas development team. (2023). *Indexing and selecting data*. Obtenido de pandas documentation: https://pandas.pydata.org/docs/user_guide/indexing.html

Pérez, J. (14 de noviembre de 2023). *Mapas de calor espaciales en Python utilizando Leaflet.js a través del módulo Folium.* Obtenido de Supply Chain Data Analytics.: https://www.supplychaindataanalytics.com/es/mapas-de-calor-espaciales-en-python-utilizando-leaflet-js-a-traves-del-modulo-folium/

Phys.org. (2023). Obtenido de El Salvador Volcanoes and Volcanics: https://volcano.si.edu/volcanolist_countries.cfm?country=El%20Salvador

Plotly, T. (2023). *Plotly Open Source Graphing Library for Python*. Obtenido de Plotly: https://plotly.com/python/

Plotly, Tecnologies. (s.f). *Advanced Callbacks*. Obtenido de Dash Plotly: https://dash.plotly.com/advanced-callbacks

Pro, I. D. (22 de Abril de 2022). *IoT Design Pro*. Obtenido de ESP32 Data Logging to Google Sheets with Google Scripts: https://iotdesignpro.com/articles/esp32-data-logging-to-google-sheets-with-google-scripts

Ramírez, I. (18 de Marzo de 2022). *xatakandroid*. Obtenido de Así cambian los permisos en Android 13: notificaciones, dispositivos cercanos, música: https://www.xatakandroid.com/sistema-operativo/asi-cambian-permisos-android-13-notificaciones-dispositivos-cercanos-musica

Revil, A., Finizola, A., Piscitelli, S., Ricci, T., Angeletti, T., & Barde-Cabusson, S. (2008). *Soil CO2 emissions and heat output during the 2006 eruption of Tungurahua volcano.* Ecuador: Journal of Geophysical Research: Solid Earth.

Scientific, A. (2021). *Atlas Scientific Blog* . Obtenido de How Does an NDIR CO2 Sensor Work?: https://atlas-scientific.com/blog/how-does-an-ndir-co2-sensor-work/

Scikit-learn developers. (n.d.). (2023). *LinearRegression.* Obtenido de scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Sierra, A. H. (2022). *Cuantificación de CO2 difuso en el Complejo Volcánico Chiles Cerro Negro*. Obtenido de https://www.igepn.edu.ec/servicios/noticias/2036-complejo-volcanico-chiles-cerro-negro-informe-de-monitoreo-de-fuentes-termales-septiembre-de-2022

SIMCom. (2014). Obtenido de SIM800 Series AT Command Manual: https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf

SIMCom. (2023). *Components101* . Obtenido de SIM800L Hardware Design V100": [SIM800L Datasheet]

System, E. (2021). *Espressif System.* Obtenido de ESP32-WROOM-32 Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Systems, E. (2021). *Espressif Systems.* Obtenido de ESP32-WROVER-KIT V3 Getting Started Guide: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-wrover-kit-v3.html

Systems, E. (2021). *Espressif Systems.* Obtenido de ESP32-WROOM-32E and ESP32-WROOM-32UE Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

Systems, E. (2023). *Espressif* . Obtenido de ESP32-WROVER-B & ESP32-WROVER-IB Datasheet v2.0: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-b_datasheet_en.pdf

Systems, E. (2023). *Espressif Systems.* Obtenido de ESP32 Series Datasheet v4.3: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

TESEL. (25 de Octubre de 2022). *Cómo usar Python para leer y escribir datos a una hoja de cálculo de Google Sheets*. Obtenido de tesel: https://tesel.mx/como-usar-python-para-leer-y-escribir-datos-a-una-hoja-de-calculo-de-google-sheets-8879/

Testo, A. (2023). *Absorción infrarroja (Proceso IR)"* . Obtenido de Testo: https://www.academiatesto.com.ar/cms/absorcion-infrarroja-proceso-ir

Trung, N. N. (25 de Enero de 2022). *Visualización de datos con Python Dash*. Obtenido de Briswell: https://www.briswell-vn.com/en/news/truc-quan-hoa-du-lieu-bang-python-dash/

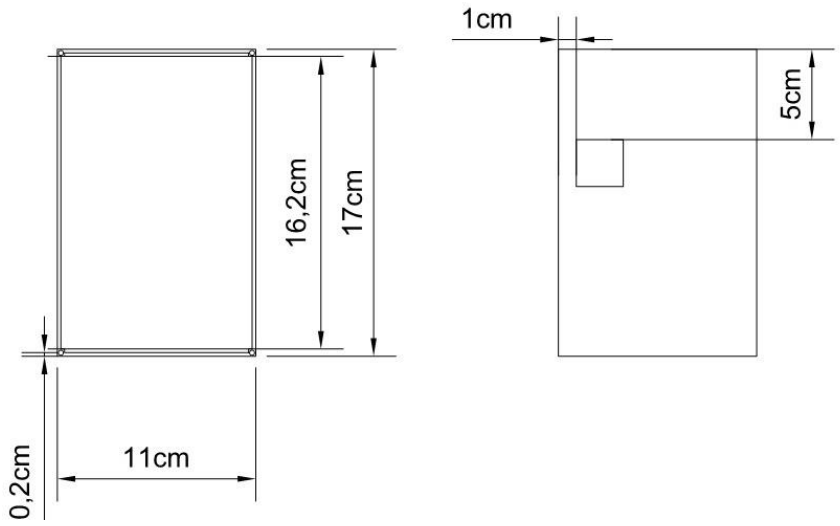
Tutorials, R. N. (2016). *Random Nerd Tutorials*. Obtenido de Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux): https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Tutorials, R. N. (2018). *Random Nerd Tutorials*. Obtenido de ESP32 ESP8266 Publish Sensor Readings to Google Sheets: https://randomnerdtutorials.com/esp32-esp8266-publish-sensor-readings-to-google-sheets/

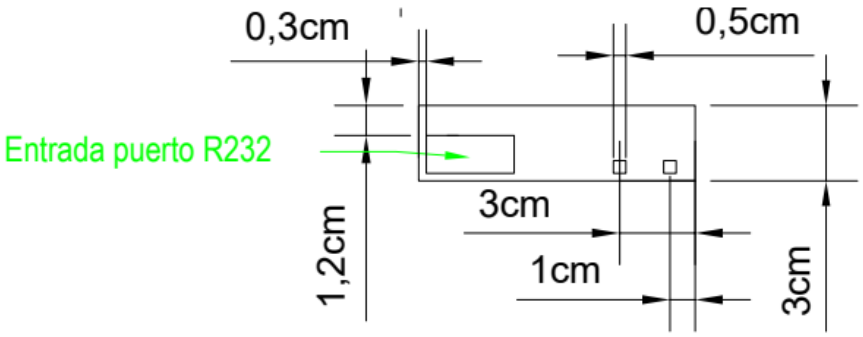
Xinyuan-LilyGO. (2023). *Github*. Obtenido de LilyGo-T-Call-SIM800: https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800

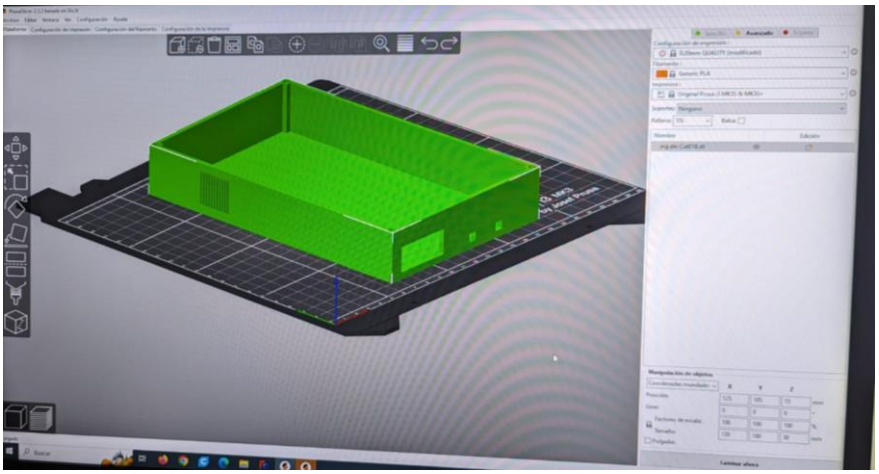
Anexos

A. Contenedor.



En las figuras se muestra el diseño de la caja que contiene el circuito impreso, realizado en FreeCad, para impresión 3D.





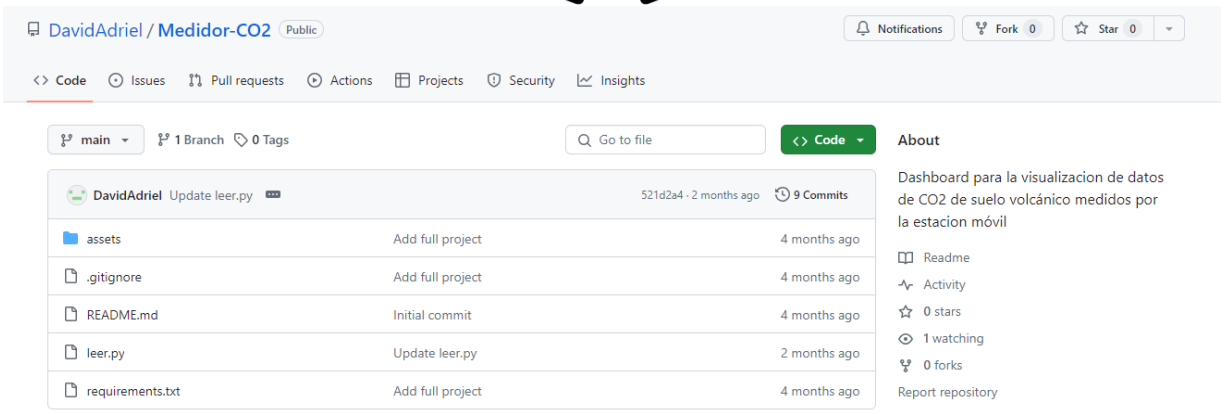
Información del laminado de la pieza tapadera y caja:

Información del laminado	
Filamento Usado (g) (incluyendo la bobina)	36,28 (237,28)
Filamento Usado (m)	12,16
Filamento Usado (mm^3)	29255,51
Coste	1,32
Tiempo estimado de impresión: - modo normal	1h58m

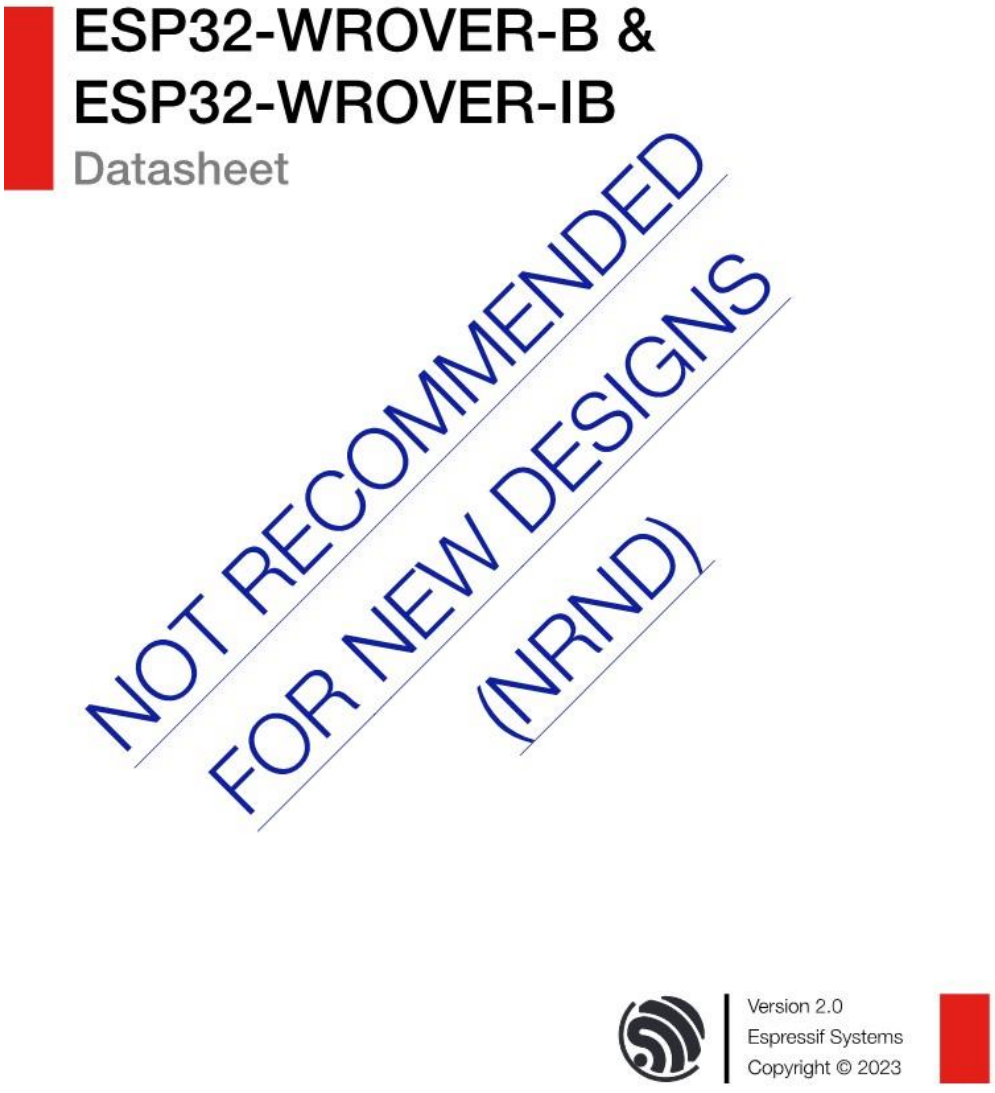
Información del laminado	
Filamento Usado (g) (incluyendo la bobina)	80,90 (281,90)
Filamento Usado (m)	27,12
Filamento Usado (mm^3)	65240,12
Coste	2,94
Tiempo estimado de impresión: - modo normal	6h13m

B. Programas.

Los programas están alojados en el repositorio de GitHub en: https://github.com/DavidAdriel/Medidor-CO2.



C. Datasheets



and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 3 provides the specifications of ESP32-WROVER-B and ESP32-WROVER-IB.

Table 3: ESP32-WROVER-B & ESP32-WROVER-IB Specifications

Categories	Items	Specifications
Certification	RF certification	See certificates for ESP32-WROVER-B and ESP32-WROVER-IB
	Bluetooth certification	BQB
	Green certification	RoHS, REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Center frequency range of operating channel	2412 ~ 2484 MHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and Bluetooth LE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, TWAI® (compatible with ISO 11898-1, i.e. CAN Specification 2.0)
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	See Table 1 and 2
	Integrated PSRAM	See Table 1 and 2
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Minimum current delivered by power supply	500 mA
	Recommended operating ambient temperature range	–40 °C ~ 85 °C
	Package size	18 mm × 31.4 mm × 3.3 mm
	Moisture sensitivity level (MSL)	Level 3

2 Pin Definitions

2.1 Pin Layout

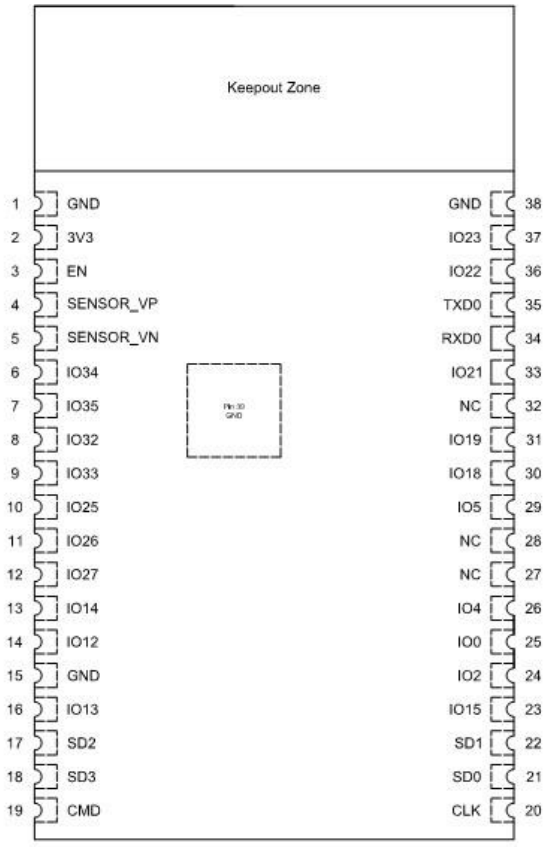


Figure 1: Pin Layout (Top View)

2.2 Pin Description

ESP32-WROVER-B and ESP32-WROVER-IB each has 38 pins. See pin definitions in Table 4.

Table 4: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.

Name	No.	Type	Function
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
NC1	27	-	-
NC2	28	-	-
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE

Name	No.	Type	Function
GND	38	P	Ground

Notice:

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the SPI flash integrated on the module and are not recommended for other uses.

2.3 Strapping Pins

ESP32 has five strapping pins, which can be seen in Chapter 6 Schematics:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

Each strapping pin is connected to its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or use the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset release, the strapping pins work as normal-function pins.

Refer to Table 5 for a detailed boot-mode configuration by strapping pins.

Table 5: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0

3 Functional Description

This chapter describes the modules and functions integrated in ESP32-WROVER-B and ESP32-WROVER-IB.

3.1 CPU and Internal Memory

ESP32-D0WD contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

3.2 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- The external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. Up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

ESP32-WROVER-B and ESP32-WROVER-IB integrate a 4 MB SPI flash and an 8 MB PSRAM for more memory space.

3.3 Crystal Oscillators

The module uses a 40-MHz crystal oscillator.

5 Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in the table below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 7: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	105	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix *IO_MUX* in *ESP32 Datasheet* for IO's power domain.

5.2 Recommended Operating Conditions

Table 8: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating ambient temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 9: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Symbol	Parameter	Min	Typ	Max	Unit
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)	-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor	-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor	-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to shut down the chip	-	-	0.6	V

Notes:

1. Please see Appendix *IO_MUX* in *ESP32 Datasheet* for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. Pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.

5.4 Wi-Fi Radio

Table 10: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Center frequency range of operating channel [note1]	-	2412	-	2484	MHz
Output impedance [note2]	-	-	*	-	Ω
TX power [note3]	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–89	-	dBm
	11g, 6 Mbps	-	–92	-	dBm
	11g, 54 Mbps	-	–74	-	dBm
	11n, HT20, MCS0	-	–91	-	dBm
	11n, HT20, MCS7	-	–71	-	dBm
	11n, HT40, MCS0	-	–89	-	dBm
	11n, HT40, MCS7	-	–69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

1. Device should operate in the center frequency range of operating channel allocated by regional regulatory authorities. Target center frequency range of operating channel is configurable by software.
2. For the modules that use external antennas, the output impedance is 50 Ω. For other modules without external antennas, users do not need to concern about the output impedance.
3. Target TX power is configurable based on device or certification requirements.

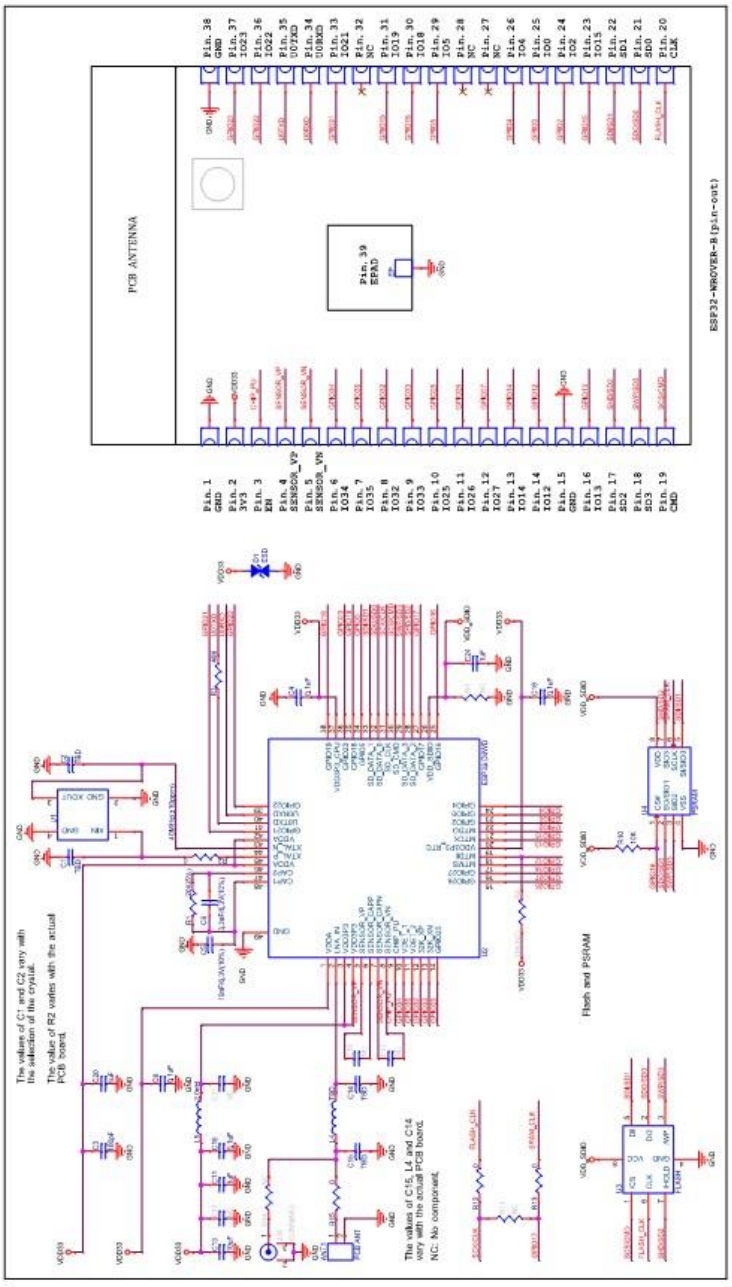


Figure 3: Schematics of ESP32-WROVER-B

7 Peripheral Schematics

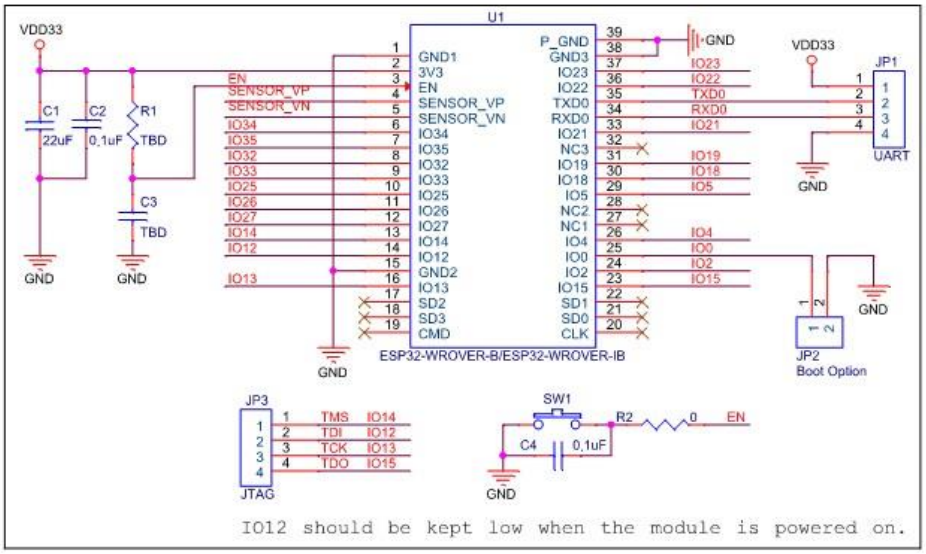


Figure 5: Peripheral Schematics

Note:

- Soldering Pad 39 to the Ground of the base board is not necessary for a satisfactory thermal performance. If users do want to solder it, they need to ensure that the correct quantity of soldering paste is applied.
- To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 1 μF. However, specific parameters should be adjusted based on the power-up timing of the module and the power-up and reset sequence timing of the chip. For ESP32's power-up and reset sequence timing diagram, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

8 Physical Dimensions

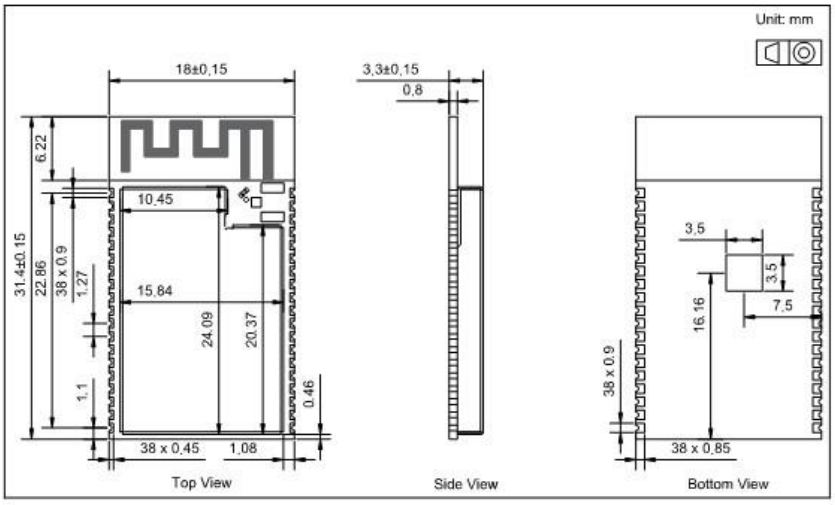


Figure 6: ESP32-WROVER-B Physical Dimensions

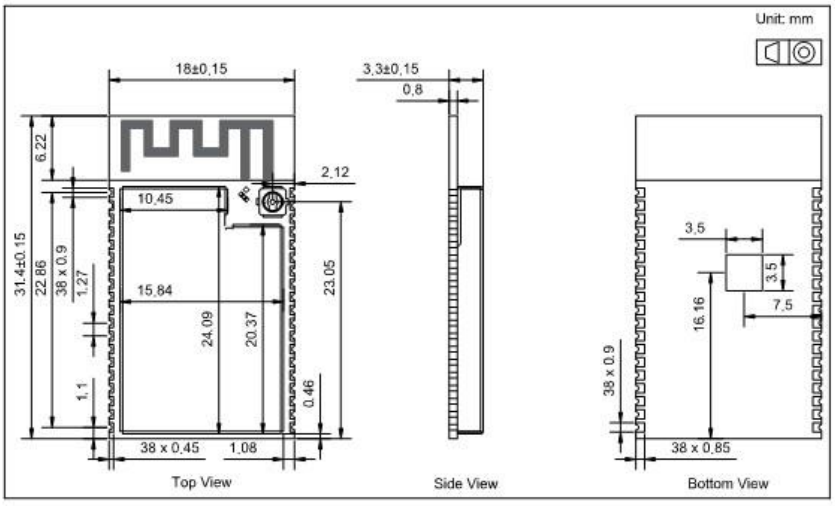


Figure 7: ESP32-WROVER-IB Physical Dimensions

Note:
For information about tape, reel, and product marking, please refer to *Espressif Module Package Information*.

maxim integrated™

DS1307
64 x 8, Serial, I²C Real-Time Clock

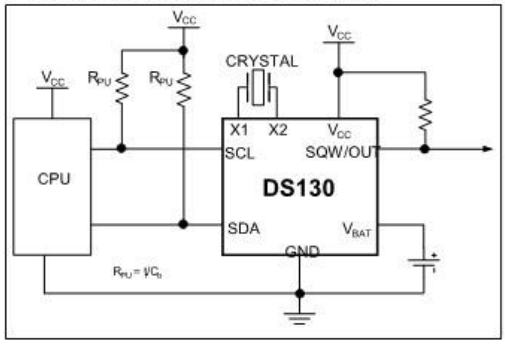
GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

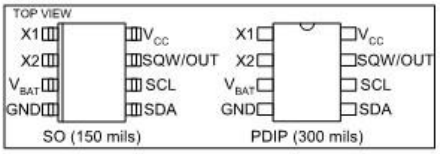
FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Compensation Valid Up to 2100
- 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratories (UL) Recognized

TYPICAL OPERATING CIRCUIT



PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.

REV: 100208

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground ..... -0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial ..... 0°C to +70°C
Industrial ..... -40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads) ..... +260°C for 10 seconds
Soldering Temperature (surface mount) ..... Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

Handson Technology

User Guide

1 Channel 5V Optical Isolated Relay Module

This is a LOW Level 5V 1-channel relay interface board, needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement and also prevent ground loop when interface to microcontroller.

SKU: MDU1091

Brief Data:

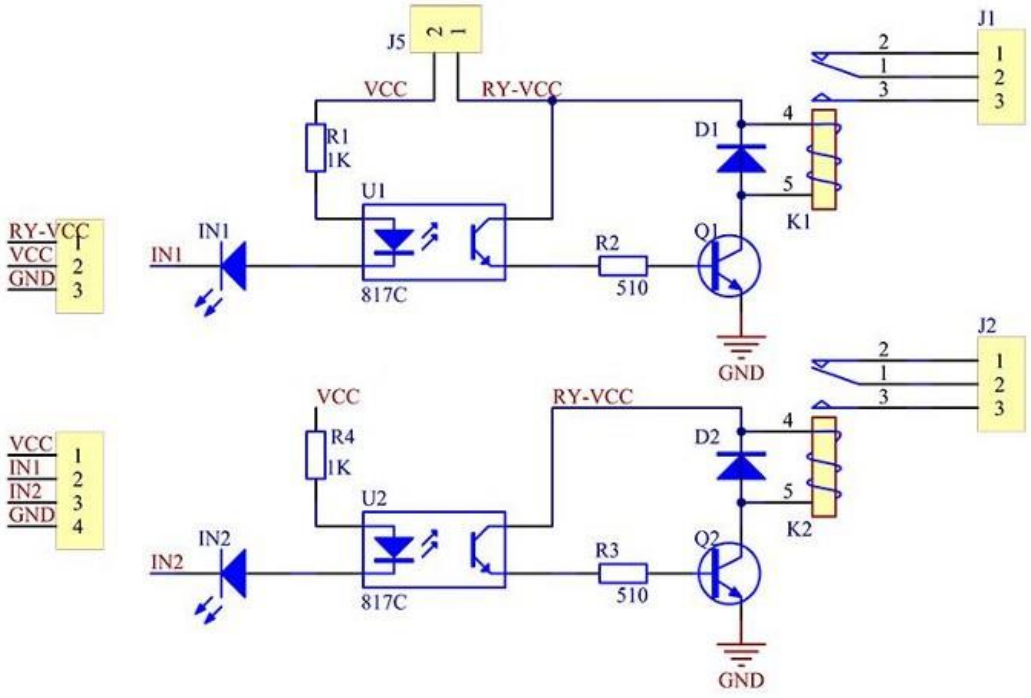
- Operating Voltage: 5Vdc.
- Relay Maximum output: DC 30V/10A, AC 250V/10A.
- 1 Channel Relay Module with Opto-coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
- Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
- Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
- Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.

Schematic:

VCC and RY-VCC are also the power supply of the relay module. When you need to drive a large power load, you can take the jumper cap off and connect an extra power to RY-VCC to supply the relay; connect VCC to 5V of the MCU board to supply input signals.

NOTES: If you want complete optical isolation, connect "Vcc" to Arduino +5 volts but do NOT connect Arduino Ground. Remove the Vcc to JD-Vcc jumper. Connect a separate +5 supply to "JD-Vcc" and board Gnd. This will supply power to the transistor drivers and relay coils.

If relay isolation is enough for your application, connect Arduino +5 and Gnd, and leave Vcc to JD-Vcc jumper in place.

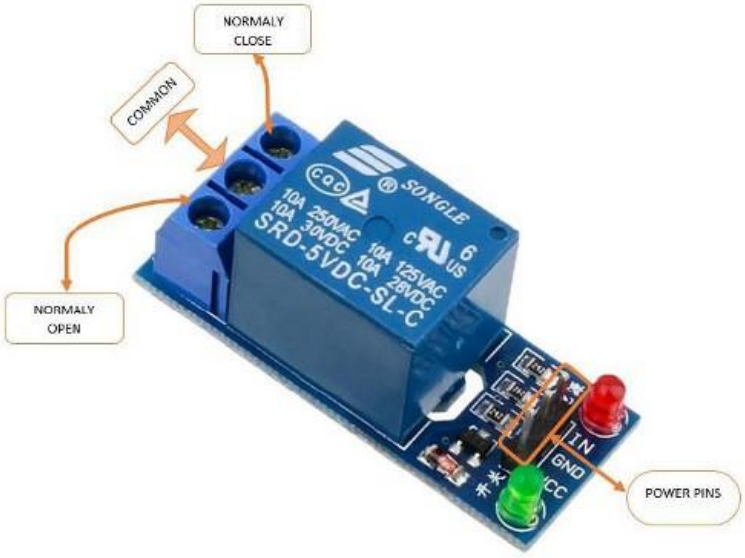


It is sometimes possible to use this relay boards with 3.3V signals, if the JD-VCC (Relay Power) is provided from a +5V supply and the VCC to JD-VCC jumper is removed. That 5V relay supply could be totally isolated from the 3.3V device, or have a common ground if opto-isolation is not needed. If used with isolated 3.3V signals, VCC (To the input of the opto-isolator, next to the IN pins) should be connected to the 3.3V device's +3.3V supply.

NOTE: Some Raspberry-Pi users have found that some relays are reliable and others do not actuate sometimes. It may be necessary to change the value of R1 from 1000 ohms to something like 220 ohms, or supply +5V to the VCC connection.

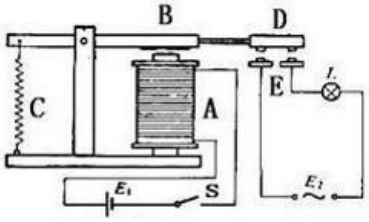
NOTE: The digital inputs from Arduino are Active LOW: The relay actuates and LED lights when the input pin is LOW, and turns off on HIGH.

Module Layout:



Operating Principle:

See the picture below: A is an electromagnet, B armature, C spring, D moving contact, and E fixed contacts. There are two fixed contacts, a normally closed one and a normally open one. When the coil is not energized, the normally open contact is the one that is off, while the normally closed one is the other that is on.



Supply voltage to the coil and some currents will pass through the coil thus generating the electromagnetic effect. So the armature overcomes the tension of the spring and is attracted to the core, thus closing the moving contact of the armature and the normally open (NO) contact or you may say releasing the former and the normally closed (NC) contact. After the coil is de-energized, the electromagnetic force disappears and the armature moves back to the original position, releasing the moving contact and normally closed contact. The closing and releasing of the contacts results in power on and off of the circuit.

Input:

VCC : Connected to positive supply voltage (supply power according to relay voltage)

GND : Connected to supply ground.

IN1: Signal triggering terminal 1 of relay module

Output:

Each module of the relay has one NC (normally close), one NO (normally open) and one COM (Common) terminal. So there are 2 NC, 2 NO and 2 COM of the channel relay in total. NC stands for the normal close port contact and the state without power. NO stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

Testing Setup:

When a low level is supplied to signal terminal of the 2-channel relay, the LED at the output terminal will light up. Otherwise, it will turn off. If a periodic high and low level is supplied to the signal terminal, you can see the LED will cycle between on and off.

For Arduino：

Step 1：

Connect the signal terminal IN1、IN2 of 2-channel relay to digital pin 4 & 5 of the Arduino Uno or ATMega2560 board, and connect an LED at the output terminal.

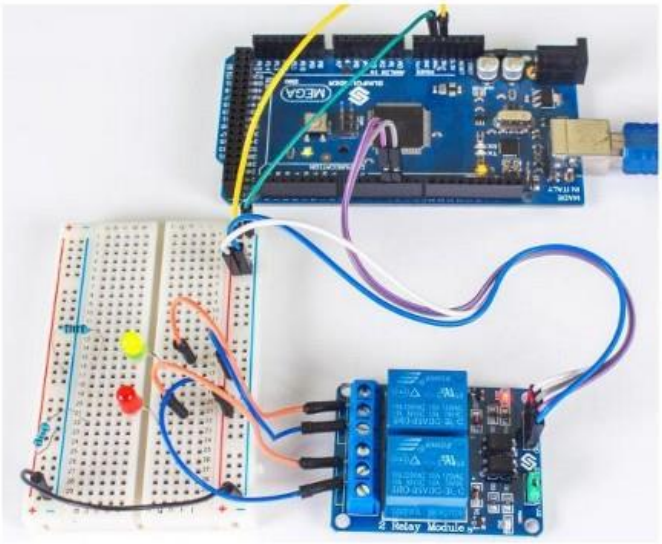
IN1> 4

IN2> 5

Step 2:

Upload the sketch "text_code" to the Arduino Uno or ATMega2560 board. Then you can see the LED cycle between on and off.

The actual figure is shown below:



For raspberry Pi:

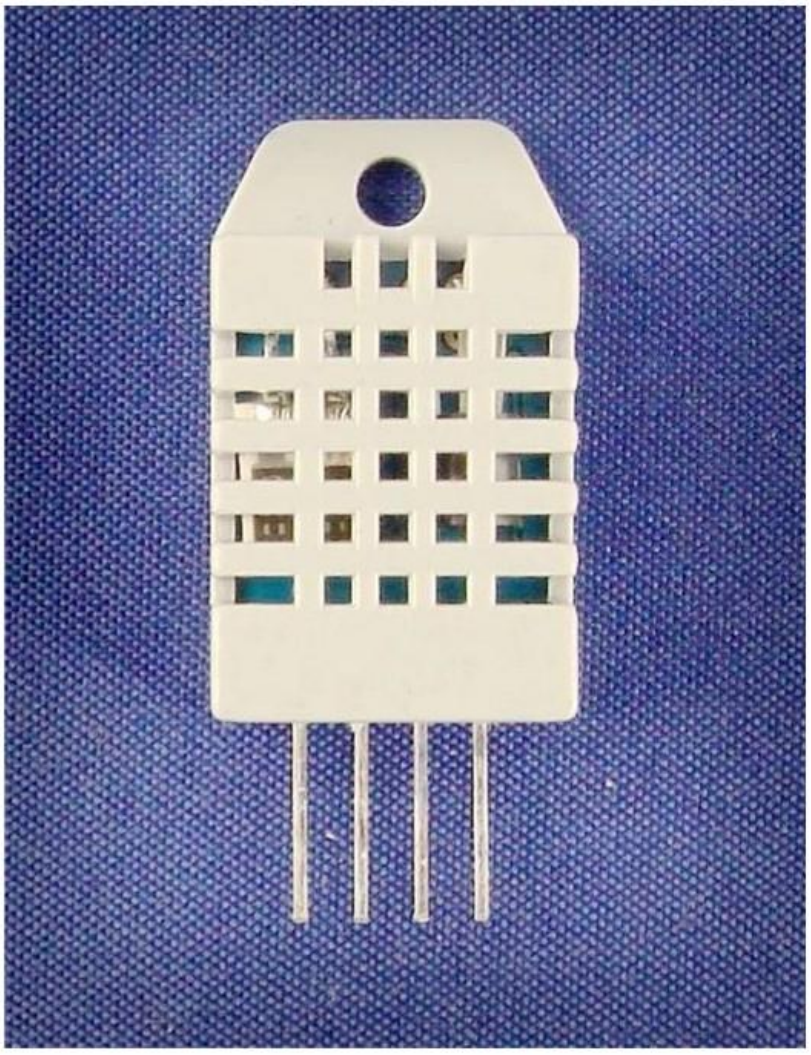
Step1:

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

Digital-output relative humidity & temperature sensor/module

DHT22 (DHT22 also named as AM2302)



Capacitive-type humidity and temperature module/sensor

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

1. Feature & Application:

* Full range temperature compensated * Relative humidity and temperature measurement
* Calibrated digital signal *Outstanding long-term stability *Extra components not needed
* Long transmission distance * Low power consumption *4 pins packaged and fully interchangeable

2. Description:

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

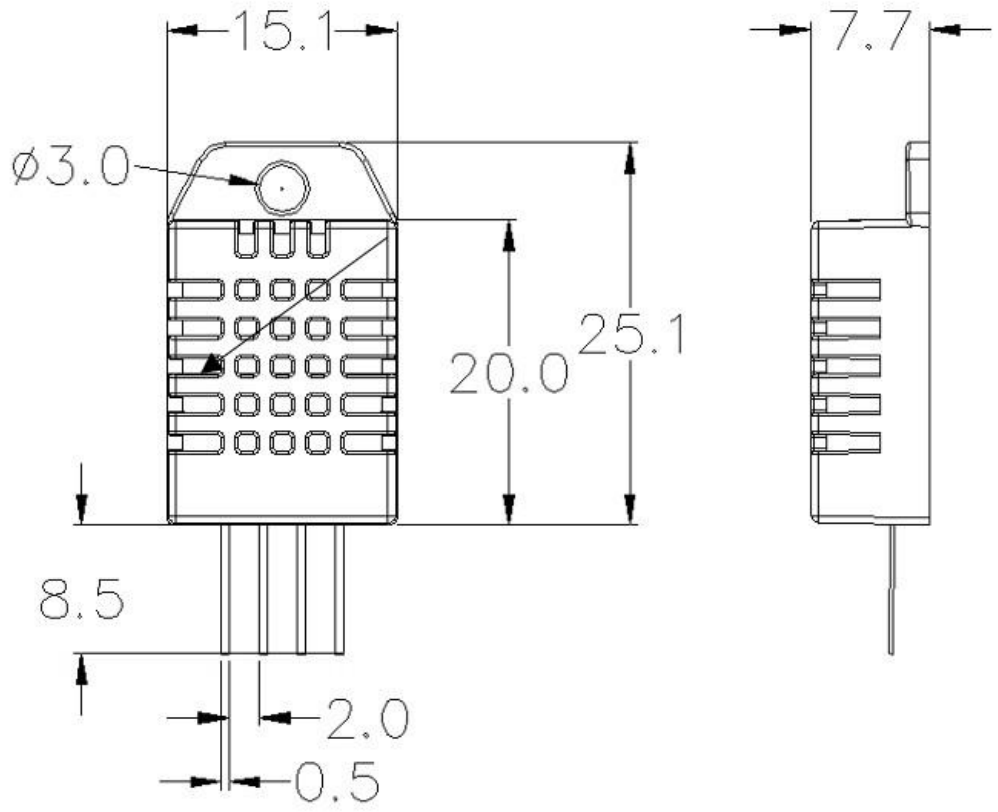
Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

4. Dimensions: (unit----mm)

1) Small size dimensions: (unit----mm)

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

19-0175; Rev 5; 10/03

MAXIM

±15kV ESD-Protected, +5V RS-232 Transceivers

General Description

The MAX202E–MAX213E, MAX232E/MAX241E line drivers/receivers are designed for RS-232 and V.28 communications in harsh environments. Each transmitter output and receiver input is protected against ±15kV electrostatic discharge (ESD) shocks, without latchup. The various combinations of features are outlined in the *Selector Guide*. The drivers and receivers for all ten devices meet all EIA/TIA-232E and CCITT V.28 specifications at data rates up to 120kbps, when loaded in accordance with the EIA/TIA-232E specification.

The MAX211E/MAX213E/MAX241E are available in 28-pin SO packages, as well as a 28-pin SSOP that uses 60% less board space. The MAX202E/MAX232E come in 16-pin TSSOP, narrow SO, wide SO, and DIP packages. The MAX203E comes in a 20-pin DIP/SO package, and needs no external charge-pump capacitors. The MAX205E comes in a 24-pin wide DIP package, and also eliminates external charge-pump capacitors. The MAX206E/MAX207E/MAX208E come in 24-pin SO, SSOP, and narrow DIP packages. The MAX232E/MAX241E operate with four 1µF capacitors, while the MAX202E/MAX206E/MAX207E/MAX208E/MAX211E/MAX213E operate with four 0.1µF capacitors, further reducing cost and board space.

Applications

Notebook, Subnotebook, and Palmtop Computers
Battery-Powered Equipment
Hand-Held Equipment

Next-Generation Device Features

- **For Low-Voltage Applications**
  MAX3222E/MAX3232E/MAX3237E/MAX3241E/MAX3246E: ±15kV ESD-Protected Down to 10nA, +3.0V to +5.5V, Up to 1Mbps, True RS-232 Transceivers (MAX3246E Available in a UCSP™ Package)
- **For Low-Power Applications**
  MAX3221/MAX3223/MAX3243: 1µA Supply Current, True +3V to +5.5V RS-232 Transceivers with Auto-Shutdown™
- **For Space-Constrained Applications**
  MAX3233E/MAX3235E: ±15kV ESD-Protected, 1µA, 250kbps, +3.0V/+5.5V, Dual RS-232 Transceivers with Internal Capacitors
- **For Low-Voltage or Data Cable Applications**
  MAX3380E/MAX3381E: +2.35V to +5.5V, 1µA, 2Tx/2Rx RS-232 Transceivers with ±15kV ESD-Protected I/O and Logic Pins

Ordering Information, Pin Configurations, and Typical Operating Circuits appear at end of data sheet.

Selector Guide

PART	NO. OF RS-232 DRIVERS	NO. OF RS-232 RECEIVERS	RECEIVERS ACTIVE IN SHUTDOWN	NO. OF EXTERNAL CAPACITORS (µF)	LOW-POWER SHUTDOWN	TTL TRI-STATE
MAX202E	2	2	0	4 (0.1)	No	No
MAX203E	2	2	0	None	No	No
MAX205E	5	5	0	None	Yes	Yes
MAX206E	4	3	0	4 (0.1)	Yes	Yes
MAX207E	5	3	0	4 (0.1)	No	No
MAX208E	4	4	0	4 (0.1)	No	No
MAX211E	4	5	0	4 (0.1)	Yes	Yes
MAX213E	4	5	2	4 (0.1)	Yes	Yes
MAX232E	2	2	0	4 (1)	No	No
MAX241E	4	5	0	4 (1)	Yes	Yes

MH-KC24

MH-KC24 QC3.0 QC2.0 USB DC-DC Buck Converter Charging Step Down Module 6-32V 9V 12V 24V to Fast Quick Charger Circuit Board 5V
USB charging module with support for Qualcomm QC3.0 QC2.0 technology (Step Down)

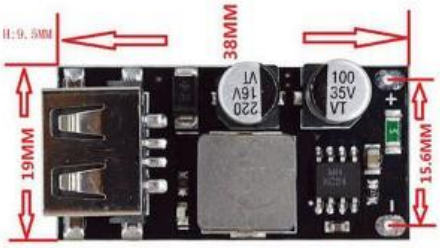
Description

Input voltage range: 6v-32v
Output Voltage: Standard 5V, Automatically adjust between 3-12V after fast charge tripping
Output power: up to 24 w (5 v / 3.4A, 9 v / 2.5A, 12 v / 2A, etc.)MH-KC24 USB charging module with support for Qualcomm QC3.0 QC2.0 technology (Step Down)
Output cable voltage compensation function
Conversion efficiency: 90% -97%
Support for multiple fast loading protocols
Input Overvoltage and undervoltage protection
Input Overcurrent protection
Overcurrent output, short circuit protection
The machine over temperature protection
Size:19*38mm

Fast Charge Support Agreement:

Support DCP protocol, BC1.2, Apple, Samsung protocol
Huawei Support Fast Charge Protocol FCP / SCP
Huawei AFC Fast Charge Support Protocol
Spreadtrum Fast Charge SFCP Support Protocol
Qualcomm QC2.0 and 3.0 Support
MTK PE1.1 / PE2.0 support

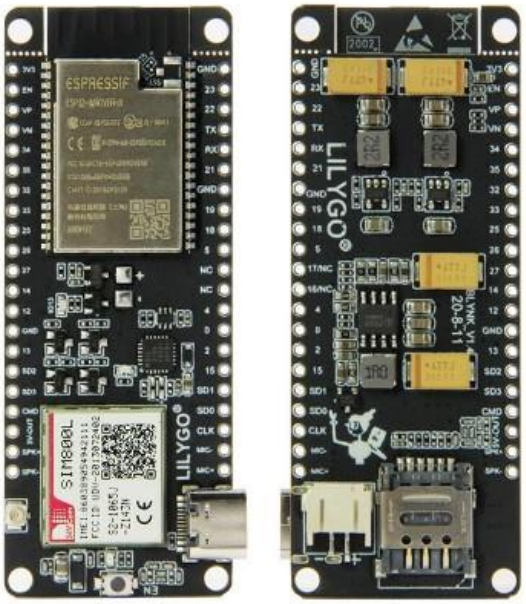

12V 24V to QC3.0
IN+
IN-
Support agreement:
QC2.0, QC3.0,


38MM
19MM
15.6MM


Supported agreement :
DCP
FCP/ SCP
AFC
SCP
QC2.0/ QC3.0
MTK PE1.1/2.0
IN-
IN+
DC 6~32V

TTGO T-Call V1.4 ESP32 Wireless Module

SIM Antenna SIM Card SIM800L Module



Product Description

Hardware

Specifications

Chipset	ESPRESSIF-ESP32 240MHz Xtensa® single-/dual-core 32-bit LX6 microprocessor
FLASH	QSPI flash 4MB / PSRAM 8MB
SRAM	520 kB SRAM
Button	Reset
USB to TTL	CP2104
Modular interface	UART、SPI、SDIO、I2C、PWM、I2S、ADC
On-board clock	40MHz crystal oscillator
Working voltage	2.7V-3.6V
Working current	About 70mA

Sleep current	About 1.1mA
SIM card	Only supports Nano SIM card
Working temperature range	-40℃ ~ +85℃
Size&Weight	78.83mm*28.92mm*8.06mm(11.77g)
Power Supply Specifications	
Power Supply	USB 5V/1A
Charging current	500mA
Battery	3.7V lithium battery
JST Connector	2Pin 2.0mm
USB	Type-C

Wi-Fi

Standard	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC(esp32 chip)
Protocol	802.11 b/g/n(802.11n，speed up to150Mbps)A-MPDU and A-MSDU polymerization，support 0.4μS Protection interval
Frequency range	2.4GHz~2.5GHz(2400M~2483.5M)
Transmit Power	22dBm
Communication distance	300m

Bluetooth

Protocol	meet bluetooth v4.2BR/EDR and BLE standard
Radio frequency	with -97dBm sensitivity NZIF receiver Class-1,Class-2&Class-3 emitter AFH
Audio frequency	CVSD&SBC audio frequency

Software specification

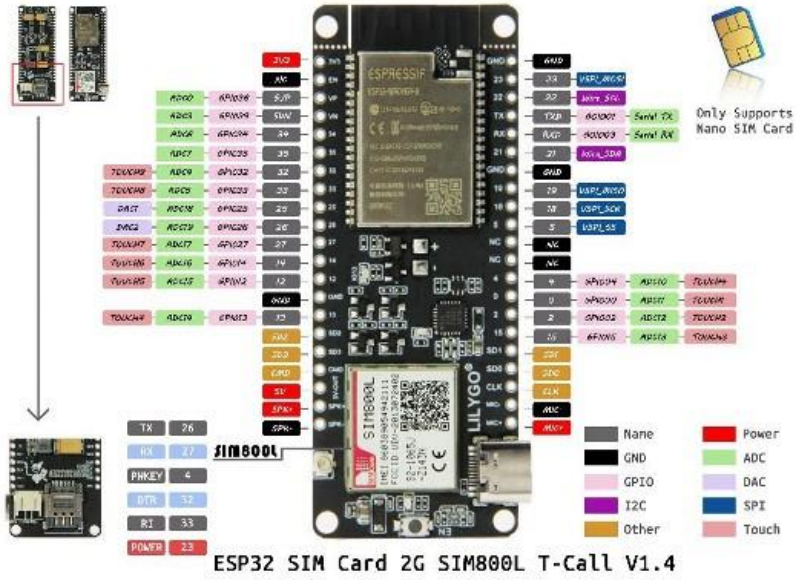
Wi-Fi Mode	Station/SoftAP/SoftAP+Station/P2P
Security mechanism	WPA/WPA2/WPA2-Enterprise/WPS
Encryption Type	AES/RSA/ECC/SHA
Firmware upgrade	UART download/OTA（Through network/host to download and write firmware）
Software Development	Support cloud server development /SDK for user firmware development
Networking protocol	IPv4、IPv6、SSL、TCP/UDP/HTTP/FTP/MQTT

User Configuration	AT + Instruction set, cloud server, android/iOSapp
OS	FreeRTOS

More Information:
https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800

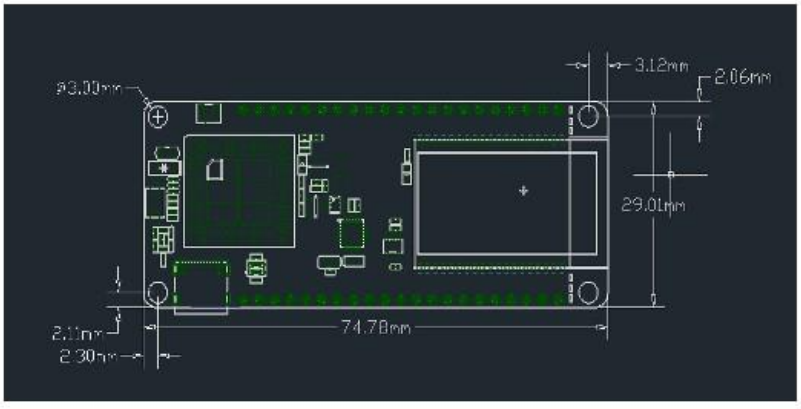
T-Call V1.4 Pin Diagram:

Connect the power supply, click to start, after start, click to reset, click twice to shut down.

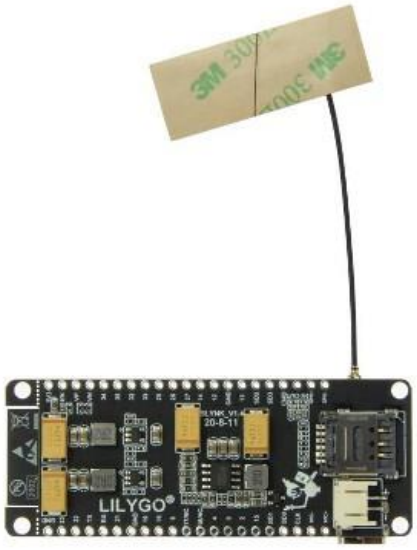


ESP32 SIM Card 2G SIM800L T-Call V1.4

T-Call V1.3 Size Chart



Product Detail



GPS

locate, communicate, accelerate

NEO-6

u-blox 6 GPS Modules

Data Sheet

Abstract

Technical data sheet describing the cost effective, high-performance u-blox 6 based NEO-6 series of GPS modules, that brings the high performance of the u-blox 6 positioning engine to the miniature NEO form factor.
These receivers combine a high level of integration capability with flexible connectivity options in a small package. This makes them perfectly suited for mass-market end products with strict size and cost requirements.

16.0 x 12.2 x 2.4mm

www.u-blox.com

1 Functional description

1.1 Overview

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints.

The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

1.2 Product features

Model	Type				Supply		Interfaces				Features						
	Standalone GPS	Capture & Process	Timing & Raw Data	Dead Reckoning	1.75 - 2.0 V	2.7 - 3.6 V	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash) FW update	TCXO (KickStart)	RTC Crystal	Antenna supply and supervisor	Configuration pins	Timepulse	External interrupt / Wakeup
NEO-6G	•				•		•	•	•	•		•	•		3	1	•
NEO-6Q	•					•	•	•	•	•		•	•		3	1	•
NEO-6M	•					•	•	•	•	•			•		3	1	•

Table 1: Features of the NEO-6 Series

All NEO-6 modules are based on AEC-Q100 qualified GPS chips. See Chapter 5.1 for further information.

1.3 GPS performance

Parameter	Specification		
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS, GAGAN		
Time-To-First-Fix[1]		NEO-6G/Q	NEO-6M
	Cold Start (Autonomous)	29 s	32s
	Warm Start (Autonomous)	29 s	32s
	Hot Start (Autonomous)	<1 s	<1 s
	Aided Starts[2]	<1 s	<3 s
Sensitivity[3]		NEO-6G/Q	NEO-6M
	Tracking & Navigation	-160 dBm	-160 dBm
	Reacquisition	-160 dBm	-160 dBm
	Cold Start (Autonomous)	-147 dBm	-146 dBm
Maximum Navigation update rate		5Hz	
Horizontal position accuracy[4]	Autonomous	2.5 m	
	SBAS	2.0 m	
Configurable Timepulse frequency range		0.1 Hz to 1 kHz	
Velocity accuracy		0.1m/s	
Heading accuracy		0.5 degrees	
Operational Limits	Dynamics	≤ 4 g	
	Altitude[5]	50,000 m	
	Velocity[5]	500 m/s	

Table 2: NEO-6 GPS performance

[1] All satellites at -130 dBm
[2] Dependent on aiding data connection speed and latency
[3] Demonstrated with a good active antenna
[4] Under good GPS signal conditions
[5] Assuming Airborne <4g platform

1.4 Block diagram

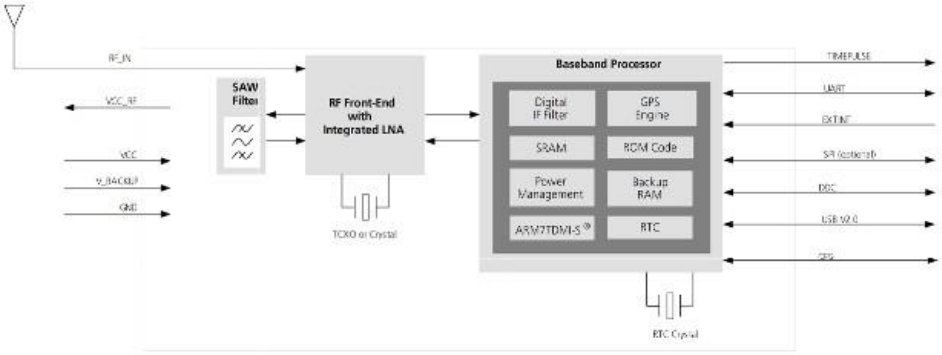


Figure 1: Block diagram (For available options refer to the product features table in section 1.2).

1.5 Assisted GPS (A-GPS)

Supply of aiding information like ephemeris, almanac, rough last position and time and satellite status and an optional time synchronization signal will reduce time to first fix significantly and improve the acquisition sensitivity. All NEO-6 modules support the u-blox AssistNow Online and AssistNow Offline A-GPS services[6] and are OMA SUPL compliant.

1.6 SuperSense Indoor GPS

All NEO-6 modules come with SuperSense, providing improved acquisition/reacquisition and tracking sensitivity. SuperSense enables high performance tracking and navigation even in difficult signal environments such as urban canyons or indoor locations.

1.7 KickStart / Oscillators

An available feature is KickStart. This functionality uses a TCXO to accelerate weak signal acquisition, enabling faster start and reacquisition times. KickStart is available with the NEO-6Q and NEO-6G.

1.8 Protocols and interfaces

Protocol	Type
NMEA	Input/output, ASCII, 0183, 2.3 (compatible to 3.0)
UBX	Input/output, binary, u-blox proprietary

Table 3: Available protocols

Both protocols are available on UART, USB, and DDC. For specification of the various protocols see the *u-blox 6 Receiver Description including Protocol Specification* [2].

NEO-6 modules support a number of peripheral interfaces for serial communication. The embedded firmware uses these interfaces according to their respective protocol specifications. For specific applications, the firmware also supports the connection of external memories.

[6] Requires external memory.

2 Pin Definition

2.1 Pin assignment

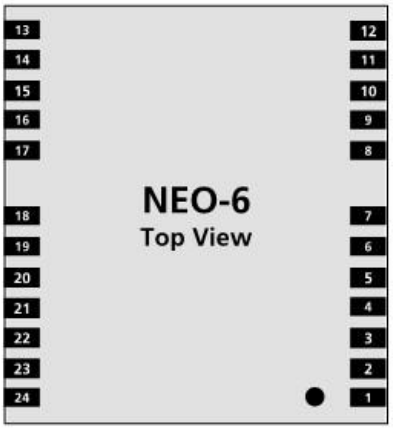


Figure 2 Pin Assignment

No	Module	Name	I/O	Description
1	All	Reserved	I	Reserved
2	All	SS_N	I	SPI Slave Select
3	All	TIMEPULSE	O	Time pulse (1PPS)
4	All	EXTINT0	I	External Interrupt Pin
5	All	USB_DM	I/O	USB Data
6	All	USB_DP	I/O	USB Data
7	All	VDDUSB	I	USB Supply
8	All	Reserved		See Hardware Integration Manual Pin 8 and 9 must be connected together.
9	All	VCC_RF	O	Output Voltage RF section Pin 8 and 9 must be connected together.
10	All	GND	I	Ground
11	All	RF_IN	I	GPS signal input
12	All	GND	I	Ground
13	All	GND	I	Ground
14	All	MOSI/CFG_COM0	O/I	SPI MOSI / Configuration Pin. Leave open if not used.
15	All	MISO/CFG_COM1	I	SPI MISO / Configuration Pin. Leave open if not used.
16	All	CFG_GPS0/SCK	I	Power Mode Configuration Pin / SPI Clock. Leave open if not used.
17	All	Reserved	I	Reserved
18	All	SDA2	I/O	DDC Data
19	All	SCL2	I/O	DDC Clock
20	All	TxD1	O	Serial Port 1
21	All	RxD1	I	Serial Port 1

3 Electrical specifications

3.1 Absolute maximum ratings

Parameter	Symbol	Module	Condition	Min	Max	Units
Power supply voltage	VCC	NEO-6G		-0.5	2.0	V
		NEO-6Q/ NEO-6M		-0.5	3.6	V
Backup battery voltage	V_BCKP	All		-0.5	3.6	V
USB supply voltage	VDDUSB	All		-0.5	3.6	V
Input pin voltage	Vin	All		-0.5	3.6	V
	Vin_usb	All		-0.5	VDDUSB	V
DC current trough any digital I/O pin (except supplies)	Ipin				10	mA
VCC_RF output current	ICC_RF	All			100	mA
Input power at RF_IN	Prfin	All	source impedance = 50 Ω, continuous wave		-5	dBm
Storage temperature	Tstg	All		-40	85	°C

Table 9: Absolute maximum ratings

GPS receivers are Electrostatic Sensitive Devices (ESD) and require special precautions when handling. For more information see chapter 6.2.6.

Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. The product is not protected against overvoltage or reversed voltages. If necessary, voltage spikes exceeding the power supply voltage specification, given in table above, must be limited to values within the specified boundaries by using appropriate protection diodes. For more information see the *LEA-6/NEO-6 Hardware Integration Manual* [1].

3.2 Operating conditions

All specifications are at an ambient temperature of 25°C.

Parameter	Symbol	Module	Min	Typ	Max	Units	Condition
Power supply voltage	VCC	NEO-6G	1.75	1.8	1.95	V	
		NEO-6Q, NEO-6M	2.7	3.0	3.6	V	
Supply voltage USB	VDDUSB	All	3.0	3.3	3.6	V	
Backup battery voltage	V_BCKP	All	1.4		3.6	V	
Backup battery current	I_BCKP	All		22		µA	V_BCKP = 1.8 V, VCC = 0V
Input pin voltage range	Vin	All	0		VCC	V	
Digital IO Pin Low level input voltage	Vil	All	0		0.2*VCC	V	
Digital IO Pin High level input voltage	Vih	All	0.7*VCC		VCC	V	
Digital IO Pin Low level output voltage	Vol	All			0.4	V	Iol=4mA
Digital IO Pin High level output voltage	Voh	All	VCC -0.4			V	Ioh=4mA
USB_DM, USB_DP	VinU	All	Compatible with USB with 22 Ohms series resistance				
VCC_RF voltage	VCC_RF	All		VCC-0.1		V	
VCC_RF output current	ICC_RF	All			50	mA	
Antenna gain	Gant	All			50	dB	
Receiver Chain Noise Figure	NFtot	All		3.0		dB	
Operating temperature	Topr	All	-40		85	°C	

Table 10: Operating conditions

Operation beyond the specified operating conditions can affect device reliability.