UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

SEGUIMIENTO Y MAPEO DE LÍNEA BASE DE EMISIÓN DE CO2 EN SUELO DEL VOLCÁN SANTA ANA

PRESENTADO POR:

ARRIAZA CARRILLO, CARLOS ALBERTO

AGUILAR VEGA, EVER OMAR

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA

SEPTIEMBRE DE 2022

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL:

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

FACULTAD DE INGENIERÍA Y ARQUITECURA

DECANO:

PhD. EDGAR ARMANDO PEÑA FIGUEROA

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR:

ING. ARMANDO MARTÍNEZ CALDERÓN

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

SEGUIMIENTO Y MAPEO DE LÍNEA BASE DE EMISIÓN DE CO2 EN SUELO DEL VOLCÁN SANTA ANA

Presentado por:

ARRIAZA CARRILLO, CARLOS ALBERTO

AGUILAR VEGA, EVER OMAR

Trabajo de Graduación Aprobado por:

Docente Asesor:

DR. CARLOS OSMIN POCASANGRE JIMÉNEZ

SAN SALVADOR, AGOSTO DE 2022

Trabajo de Graduación Aprobado por:

Docente Asesor:

DR. CARLOS OSMIN POCASANGRE JIMÉNEZ

NOTA Y DEFENSA FINAL

En esta fecha, viernes 15 de julio de 2022, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 3:00 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director

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 JIMÉNEZ
  (Docente Asesor)

Firma

- ING. JOSE ROBERTO RAMOS

Firma

- MSC. JOSE WILBER CALDERON URRUTIA

Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

SEGUIMIENTO Y MAPEO DE LÍNEA BASE DE EMISIÓN DE CO2 EN SUELO DEL VOLCÁN SANTA ANA, EL SALVADOR

A cargo de los Bachilleres:

- AGUILAR VEGA EVER OMAR

- ARRIAZA CARRILLO CARLOS ALBERTO

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: _9.6__
(nueve puntos seis)

Agradecimientos

La presente tesis fue desarrollada debido al enlace benéfico entre las ramas de geofísica e ingeniería eléctrica de la Universidad de El Salvador; y con ella se sientan las primeras bases para la construcción de más equipos de medición orientados a la vulcanología en un futuro cercano. Es por ello que quiero expresar mi gratitud con el trabajo previo publicado.

En primer lugar, quiero agradecer al Lic. Benancio Henríquez Miranda quien fue el encargado de prestarnos los sensores de Dióxido de Carbono y darnos el reto de crear una comunicación bluetooth entre los sensores y un teléfono móvil, sin él, la construcción de estos equipos de medición no hubiese sido posible ya que también nos apoyó económicamente a lo largo del desarrollo de esta tesis. Por último, agradecer por los conocimientos que nos brindó en cuanto a la metodología de medición de CO2 en el suelo volcánico y como interpretar los datos de flujo de CO2.

Agradecer también a mi Docente asesor, Dr. Carlos Osmín Pocasangre Jiménez, quien a lo largo de la construcción de los equipos nos orientó por el camino correcto. Agradecer a él por el apoyo en las visitas de campo al volcán de Santa Ana y proporcionarnos un espacio con las condiciones adecuadas para la construcción de los equipos. Además de ser nuestro apoyo y guía en el proceso de graduación y motivarnos a publicar artículos referentes a nuestra tesis en congresos y revistas, gracias a ello pudimos colocar el nombre de la Universidad de El Salvador y de la Escuela de Ingeniería Eléctrica en el XIV Congreso Geológico de América Central & VII Congreso Geológico Nacional, Costa Rica 2022.

Así, mismo agradecer a mi familia; en especial a mi madre Iliana del Carmen Carrillo por todo el amor y consejos recibidos, y siempre apoyarme en cada una de las decisiones que tome a

lo largo de mi carrera, por estar siempre ahí para mí y que no me hubiese hecho falta nada; al igual que a mi padre Ricardo Alexander Arriaza Ayala por apoyarme en todo momento, en las diferentes circunstancias que la vida me iba poniendo siempre estuvo él para apoyarme y motivarme a seguir adelante. A mi hermano Ricardo Alexander Arriaza Carrillo por estar prácticamente en todo momento a mi lado, por escucharme y desearme lo mejor en mi carrera.

Agradecer a mi compañero de tesis, Ever Omar Aguilar Vega, por su invaluable aporte en el desarrollo de este trabajo de tesis, todas las horas de trabajo dedicadas y por ser una gran fuente de apoyo no solo durante este trabajo si no durante toda la carrera.

Mi gratitud también a los docentes de la Escuela de Ingeniería Eléctrica, sus conocimientos tan valiosos me han servido para el desarrollo de la tesis, pero lo más importante, me servirán a lo largo de mi vida profesional.

Agradecer a mis compañeros de carrera, Luis Gerardo Pérez Herrera, Lizardo Ariel Arias Ramírez y Diego Esaú Hernández López, quienes también participaron anteriormente en la construcción del primer prototipo de equipo de medición de CO2 como parte de un trabajo de horas sociales.

Finalmente agradecer y mencionar nuevamente a Diego, Lizardo y Omar por el equipo de estudio que formamos al inicio de la carrera, fueron 5 largos años, pero sin ellos, sin ese equipo que formamos hubiese sido muy complicado afrontar la carrera, agradecerles a ellos por los buenos momentos estudiando hasta altas horas de la noche, por sus consejos y apoyo en todo momento.

A todos, muchas gracias.

Carlos Alberto Arriaza Carrillo

Mis agradecimientos van dirigidos a todos aquellos que mostraron su apoyo y ayuda en cada etapa de esta investigación y también a lo larga de mi formación profesional.

Agradezco al Lic. Benancio Henríquez Miranda, quien fue mi docente de la materia Métodos Experimentales en el primer ciclo de la carrera y nos introdujo al monitoreo volcánico y puso su confianza en nosotros para llevar a cabo este diseño y mostró firmemente su apoyo en la implementación y ejecución de dicho proyecto.

También agradezco al Dr. Carlos Osmín Pocasangre Jiménez, mi docente asesor, quien ha estado presente en cada etapa del desarrollo de la investigación, apoyando con proporcionarnos las herramientas y los laboratorios con las condiciones adecuadas y necesarias en este proyecto. Pendiente de los procesos de la investigación y la motivación para participar del XIV Congreso Geológico de América Central & VII Congreso Geológico Nacional, Costa Rica 2022; gracias a ello, logramos publicar un artículo y exponer el tema de investigación frente a diferentes entidades internacionales.

Gracias a la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador, por abrirnos las puertas para realizar esta investigación y brindarnos un espacio que nos permitió un crecimiento de mis conocimientos y la oportunidad de crear alianzas con otra rama, la vulcanología, y así poder aplicar la ingeniería dentro de esta ciencia encargada del monitoreo volcánico.

Extiendo también mi agradecimiento a mis papás, Ever Omar Aguilar Portillo e Imelda Lizeth Vega de Aguilar, quienes me han dado mucho cariño y motivación para salir adelante con mi carrera profesional y no me han negado cualquier inconveniente económico que este represente para cumplir la realización de este.

Agradezco a mis hermanos, Lizeth Aziel Aguilar Vega y Robin Stanley Aguilar Vega, quienes me concedieron su apoyo y paciencia en los momentos que los requise, siendo parte esencial en los momentos que nos tocó compartir en el hogar.

Extiendo mis agradecimientos a mi compañero en este trabajo de graduación, Carlos Alberto Arriaza Carrillo, por su arduo esfuerzo y constante intervención en la elaboración de este proyecto. Por compartir su confianza y motivación en cada uno de los años que compartimos mientras nos formábamos profesionalmente.

A mis compañeros de carrera Lizardo Ariel Arias Ramírez y Diego Esaú Hernández López, con quienes hemos formado un fuerte equipo de estudio a lo largo de la carrera y también un equipo de apoyo y amistad en nuestra instancia lejos de nuestros hogares. También incluyo a mis compañeros y amigos Steve Vega, Ariel Segovia y Kevin Hernández, por su amistad a lo largo de la carrera.

Expreso con mucho aprecio mis agradecimientos a todas aquellas personas que han formado parte en cada uno de los diferentes puntos para llevar a cabo este presente trabajo de graduación.

"*El principio de la sabiduría es el temor a Jehová*" Proverbios 1:7

Ever Omar Aguilar Vega

CONTENIDO

INDICE DE TABLAS

ÍNDICE DE IMÁGENES

ABSTRACT (RESUMEN)

El monitoreo de volcanes ha resultado ser una de las tareas más difíciles a los que se han enfrentado los vulcanólogos desde tiempos pasados, la difícil tarea de recolectar información de estos es debido a las condiciones del terreno. En la actualidad estos problemas han decrecido, con el nacimiento de nuevas tecnologías de sensores, los profesionales que estudian los volcanes han tenido un gigantesco éxito en la recolecta de datos para el monitoreo de muchas de las variables que rigen el comportamiento de un volcán, desde los gases emergentes del cráter y tierra hasta los micro sismos registrados en el suelo. Introduciéndonos al tema de los sensores, podemos dirigir nuestra atención hacia los sensores de gases, en especial, el sensor de dióxido de carbono (CO2), este sensor de CO2 permite a los profesionales monitorear la concentración de CO2 de un volcán así como el flujo de CO2 que el volcán está emitiendo, si obtenemos un flujo creciente, esto es un fuerte indicativo de que el volcán tiene una alta probabilidad de erupción, porque el magma en el subsuelo está causando que el gas se escape por las grietas terrenales del volcán, aquí es donde entra el LI-COR LI-830, este dispositivo permite el monitoreo constante de CO2 en la zona volcánica pero con el detalle que debe estar conectado a una computadora para poder obtener datos; lo que se realizó en este proyecto fue una forma de facilitar a los estudiosos de los volcanes la tarea de recolectar los datos de CO2: Visualizar los datos de CO2 por medio de una aplicación para Android así como guardar estos datos en un archivo con extensión “csv”, de esta forma los profesionales en el área tendrán acceso a los datos de CO2 en tiempo real, así como obtener el flujo de CO2 en el volcán, de la misma manera se muestran los resultados obtenidos así como la aplicación desarrollada para esto. El proyecto se realizó usando el microcontrolador ATMega328P y el entorno IDE Android Studio para la creación de la aplicación móvil Android.

CAPÍTULO I

INTRODUCCIÓN A MONITOREO VOLCÁNICO

1.1 MONITOREO VOLCÁNICO

El Salvador se ubica en el conocido cinturón de fuego, ha poseído una fuerte actividad volcánica desde muchos años atrás, entre las más recientes podemos mencionar la erupción del volcán de Santa Ana en el 2005, en la zona occidental y la erupción del volcán de San Miguel en el 2013, en la zona de Oriente, dos de los volcanes más activos que posee El Salvador. Parte de la actividad volcánica presentada en el país, se menciona la emisión de gases y excepcionalmente de cenizas. Los volcanes de activos a la fecha son el volcán de Santa Ana, el volcán de San Miguel y el volcán de Izalco.

¿Por qué monitorear la actividad volcánica? Los volcanes son parte del ecosistema del país, los cuales poseen dentro suyo una cámara magmática. La cámara magmática es un almacén de roca fundida (magma) de gran tamaño. La cámara almacena también una gran presión que con el tiempo puede fracturar la roca que lo envuelve. Si el magma encuentra una salida hacia la superficie terrestre, se da una erupción volcánica. Detectar las cámaras magmáticas son muy difíciles por su profundidad dentro de la tierra, la mayoría de las conocidas, comúnmente se encuentran entre 1 y 10 kilómetros de profundidad, cercano a la superficie.

Los gases de origen magmático ascienden a la superficie por medio de fracturas y/o conductos del volcán, manifestándose en la superficie como fumarolas.

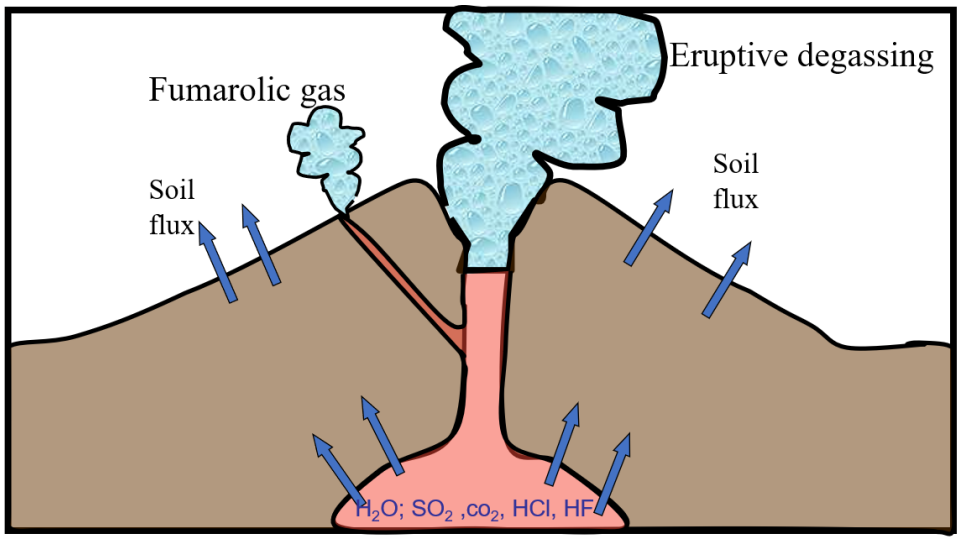
Los volcanes son fuentes emisoras de una cantidad de distintos gases, los cuales son expulsados a través de su chimenea y del suelo del terreno. Monitorear las expulsiones de los

gases nos permite encontrar sistemas de fracturas en el suelo (fallas geológicas), y mantener vigilancia de la cámara magmática. (Observatorio Vulcanológico INGEMMET, 2022)

Los gases que emiten los volcanes son: Vapor de agua (H20), Dióxido de Carbono (C02) Dióxido de Azufre (S02) Sulfuro de Hidrógeno (H2S), Hidrógeno (H2), Monóxido de carbono (CO), Cloruro de Hidrógeno (HCl), Fluoruro de Hidrógeno (HF), y Helio (He). Siendo mayoritarios el Vapor de agua (H2O), el Dióxido de carbono (CO2) y el Dióxido de azufre (SO2). En este caso, el gas que se ha tomado como objeto de medición es el Dióxido de carbono (CO2), este gas es el punto de partida en todo el análisis llevado a cabo por este proyecto.

Ilustración 1

Emisiones de gases de volcanes



La vigilancia permanente de las concentraciones en la composición química y los parámetros fisicoquímicos (temperatura, pH, conductividad eléctrica, potencial espontáneo SP) de las fuentes de aguas termales y fumarolas de zonas próximas a los volcanes, podrían indicar un incremento de la actividad volcánica y podrían ser precursores de una erupción volcánica. (Observatorio Vulcanológico INGEMMET, 2022)

1.2 SENSOR DE CO2 LI COR 830

a) Características

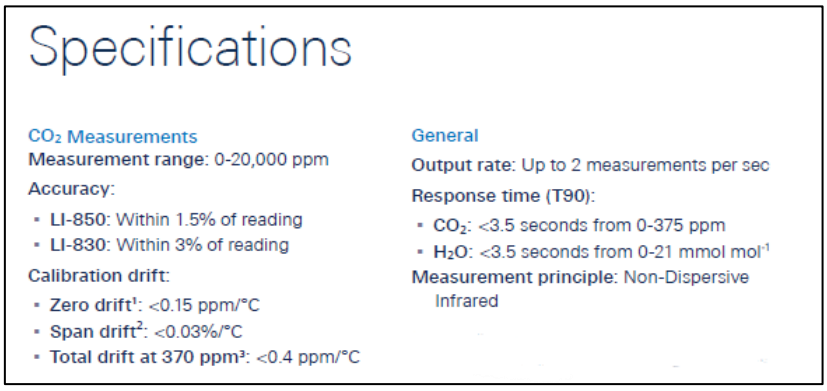
Ilustración 2

Sensor de CO2 LI COR 830

El sensor con el cual se desarrolla todo el presente proyecto, es el Sensor LI COR 830, este es un sensor CO2 de alto rendimiento. Sus características más destacadas son: su rango de medición de 0 partículas por millón (ppm) a 20,000 partículas por millón; cuenta con una precisión del 3%; su rango de voltaje de alimentación va de 12-30Vdc. El uso del sensor LI-COR LI-830 para la medición del gas CO2, nos brinda una lectura estable, precisa y fiable. El principio de medición de este sensor, es el uso de un sensor infrarrojo no disperso (NDIR) para la lectura del gas.

Ilustración 3

Especificaciones principales del Sensor LI COR 830 (LI-COR, 2021)



Specifications

CO_2 Measurements

Measurement range: 0-20,000 ppm

Accuracy:

- **LI-850**: Within 1.5% of reading
- **LI-830**: Within 3% of reading

Calibration drift:

- **Zero drift**[1]: <0.15 ppm/°C
- **Span drift**[2]: <0.03%/°C
- **Total drift at 370 ppm**[3]: <0.4 ppm/°C

General

Output rate: Up to 2 measurements per sec

Response time (T90):

- **CO_2**: <3.5 seconds from 0-375 ppm
- **H_2O**: <3.5 seconds from 0-21 mmol mol^{-1}

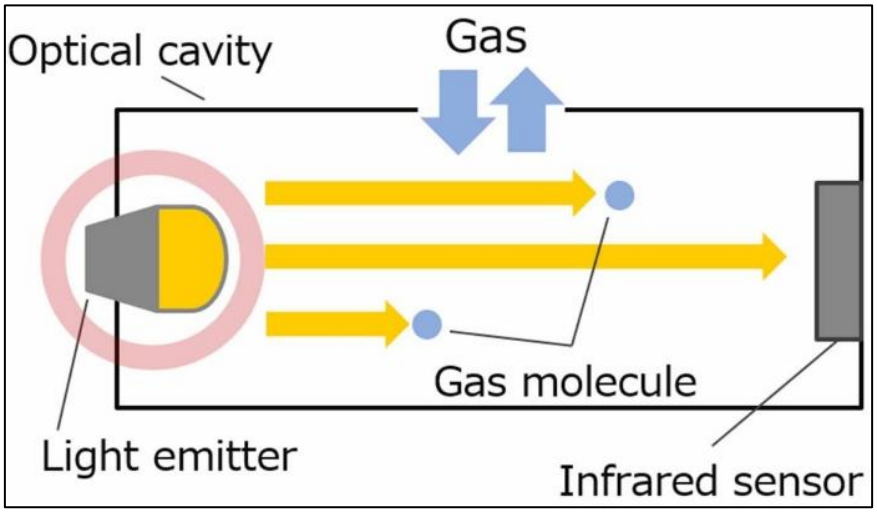
Measurement principle: Non-Dispersive Infrared

b) Sensor NDIR:

El principio de medición utilizado por este sensor, es NDIR, Sensor infrarrojo no dispersor (*Non-Dispersive Infrared).*

Ilustración 4

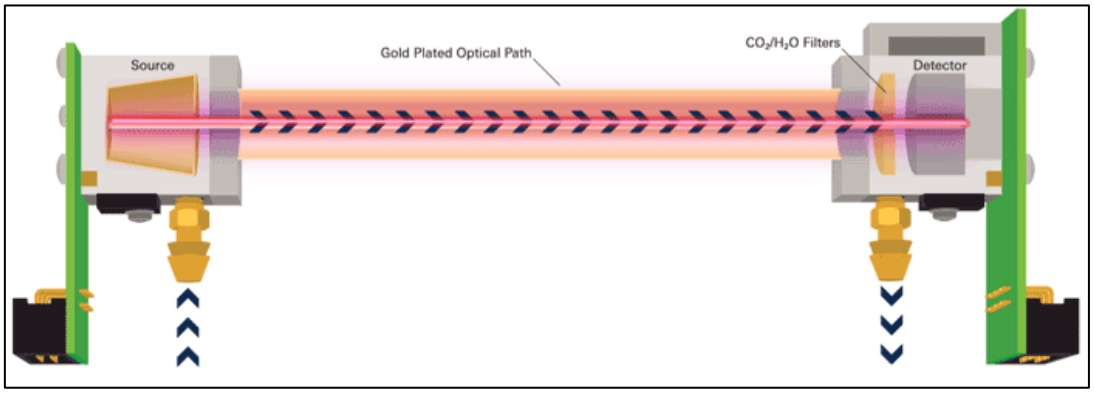
Funcionamiento de NDIR



La concentración de gas se mide electroópticamente por la absorción de una determinada longitud de onda en el infrarrojo (IR).

Ilustración 5

NDIR interno del LI COR 830 (LI-COR, s.f.)



El principio cuenta con distintos componentes, se emite a través de un tubo óptico o de muestra, desde la fuente una señal o luz infrarroja, también ahí se encuentra la entrada del gas en análisis con la ayuda de una bomba; luego, llega a un filtro detector de la señal infrarroja. La

concentración de gas se mide electroópticamente por la absorción de una determinada longitud de onda en el infrarrojo. Si no encuentra ningún componente de CO2, el filtro no absorbería la luz modificada de esa longitud de onda y no afectaría la cantidad de luz que llega al detector.

c) Calibración

Como parte del mantenimiento requerido de este sensor es poco, el proceso de calibración se describe de la siguiente manera:

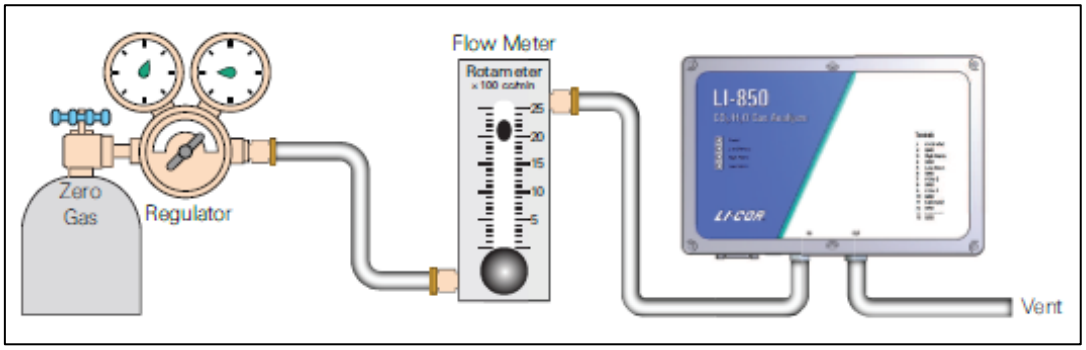
> Si el instrumento no está midiendo como se esperaba, si ha desarmado el banco óptico por algún motivo o simplemente ya ha pasado algún tiempo desde su última calibración, debe verificar los ajustes de cero e intervalo y configurarlos si es necesario. El fabricante pone a disposición un software para calibrar el sensor (LI-COR, 2021).

Es importante hacer primero el ajuste de cero de CO2. Se necesita un tanque de aire seco que no contenga CO2.

Conectar el tanque de gas cero o el depurador a la entrada de aire. Instalando un tubo de 10 a 20 cm de longitud a la salida de aire.

Ilustración 6

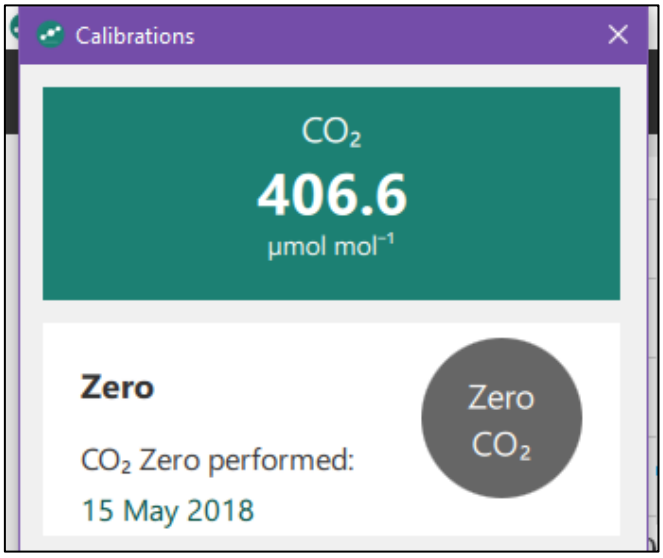
Esquema de conexión para calibración con gas cero CO2 (LI-COR, 2021)



En la aplicación de pc proporcionado por el fabricante, tenemos que ingresar al apartado de calibración. Cuando la lectura de la concentración se estabilice, se hace clic en el Botón Zero CO2.

Ilustración 7

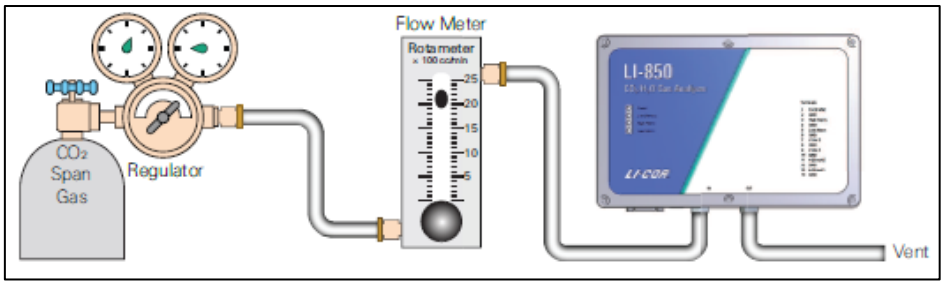
Botón ZeroCO2 dentro de software de calibración (LI-COR, 2021)



El siguiente paso es medir un gas con una concentración de CO2 conocida. Se recomienda uno cercano a los 400 ppm de CO2.

Ilustración 8

Esquema de conexión para calibración con gas conocido de CO2 (LI-COR, 2021)

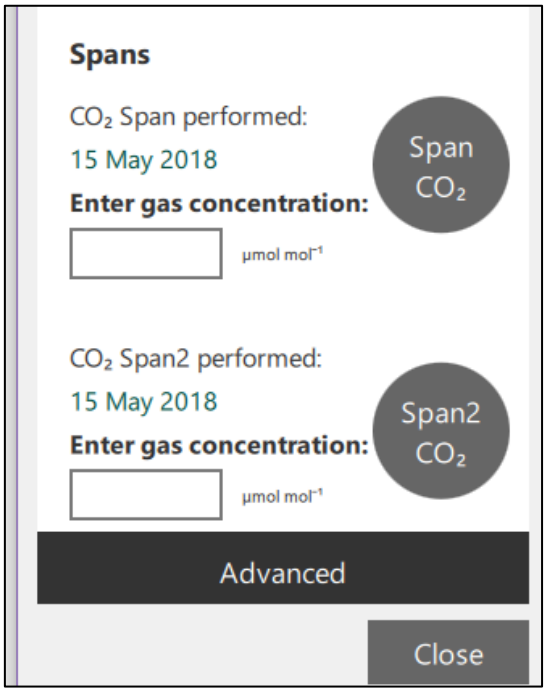


Después en el software introduzca el patrón del gas conocido cuando la lectura de concentración del gas se estabilice. Se puede mejorar la precisión del sensor tomando de base

una segunda sustancia para el span o el intervalo del sensor. Puede ser mayor o menor al primero, solamente se debe de seleccionar el segundo botón.

Ilustración 9

Introducción de valores de concentración de CO2 de los gases para calibración (LI-COR, 2021)



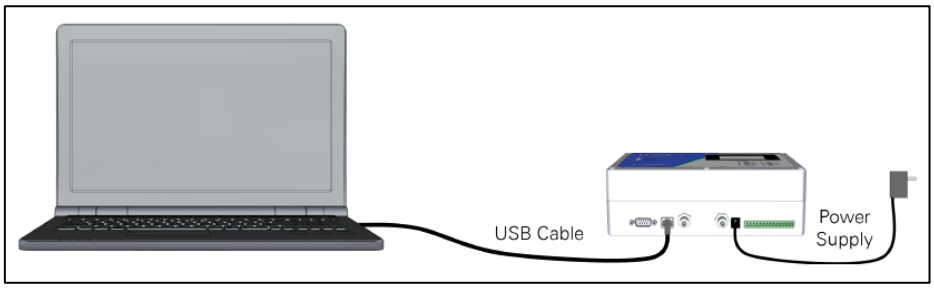
d) Programación

Para nuestro proyecto fue necesario configurar el LI-830 de manera de apagar las demás variables que vienen por defecto en el sensor, para ello solo tenemos que ingresar un código corto en XML para configurarlo y dejar solamente la variable de CO2.

Para configurar el LI-830 solo tenemos que conectarlo a través del cable USB a un ordenador (LI-COR, 2018).

Ilustración 10

Conexión entre LI-COR 830 y PC (LI-COR, 2018)



Iniciamos un programa de terminal, nosotros usamos *TeraTerm* y seleccionamos el puerto llamado COM#: USB Serial Port (COM#) y se verá un flujo de datos en pantalla de la terminal (LI-COR, 2018).

Ilustración 11

TeraTerm utilizado para programar LI-COR (LI-COR, 2018)

Copiamos el siguiente código en cual hemos configurado el sensor para que muestre datos de CO2 cada 1 segundo en pantalla, y a la vez hemos puesto en estado de false el resto de variables. Esto lo hacemos por practicidad al momento de manejar las tramas de datos en el microcontrolador ya que también tenemos data de temperatura y humedad por parte del DHT-22 y datos de geolocalización del GPS NEO6M.

Ilustración 12

Código para programar LI-COR 830

```
<li830>
        <cfg>
                <outrate>1</outrate>
        </cfg>
        <rs232>
                <co2>true</co2>
                <flowrate>false</flowrate>
                <h2o>false</h2o>
                <celltemp>false</celltemp>
                <cellpres>false</cellpres>
                <ivolt>false</ivolt>
                <co2abs>false</co2abs>
                <h2oabs>false</h2oabs>
                <h2odewpoint>false</h2odewpoint>
                <raw>false</raw>
                <echo>false</echo>
                <strip>true</strip>
        </rs232>
</li830>
```

Una vez copiado el código en el portapapeles en *TeraTerm* damos clic en Editar, Pegar<CR> y veremos que el código ya aparece damos ok y de esta forma el LI-830 queda ya configurado para leer solamente el dato de CO2 cada 1 segundo.

e) Observaciones

Se puede destacar 3 aspectos importantes a tomar en cuenta para el diseño de este proyecto:

- Software para medición. La empresa LI COR cuenta a disposición con un software para realizar la toma de mediciones de concentración de CO2.
- Uso en estación fija. Su diseño es para una estación fija, conectado a la red eléctrica y conectado a una PC.

- Sin Display. Este modelo del sensor no cuenta con una pantalla digital para observar su medición.

Ilustración 13

Diseño de funcionamiento propuesto por el fabricante (LI-COR, 2021)

El diseño original de este sensor, es para una comunicación a través de USB o RS232, hacia un software para los sistemas operativos Windows o Mac. La implementación del problema, es la necesidad de que crear una comunicación hacia un dispositivo móvil Android, por medio de una conexión inalámbrica bluetooth. Es por ello la importancia de diseñar un sistema de medición, dicho equipo, ha de ser diseñado para cubrir todas las necesidades para la comunicación móvil. Además de cubrir el sistema de comunicación, existe la necesidad de contar con una alimentación a través de una batería, con una autonomía adecuada para su uso en las salidas de campo para recolectar los datos sin interrupción y con la capacidad de alimentar al equipo las horas necesarias del día de forma eficiente.

1.3 FLUJO DE CO2

Es la velocidad de transferencia con el que se emite el gas CO2. El dióxido de carbono es un gas no tóxico y no inflamable. También es inodoro e incoloro, resulta imposible detectar las fugas, con lo cual es necesario utilizar los sensores adecuados para medir su concentración.

A continuación, presentamos una tabla que muestra los niveles de concentración de CO2:

Tabla 1

Niveles de concentración de CO2

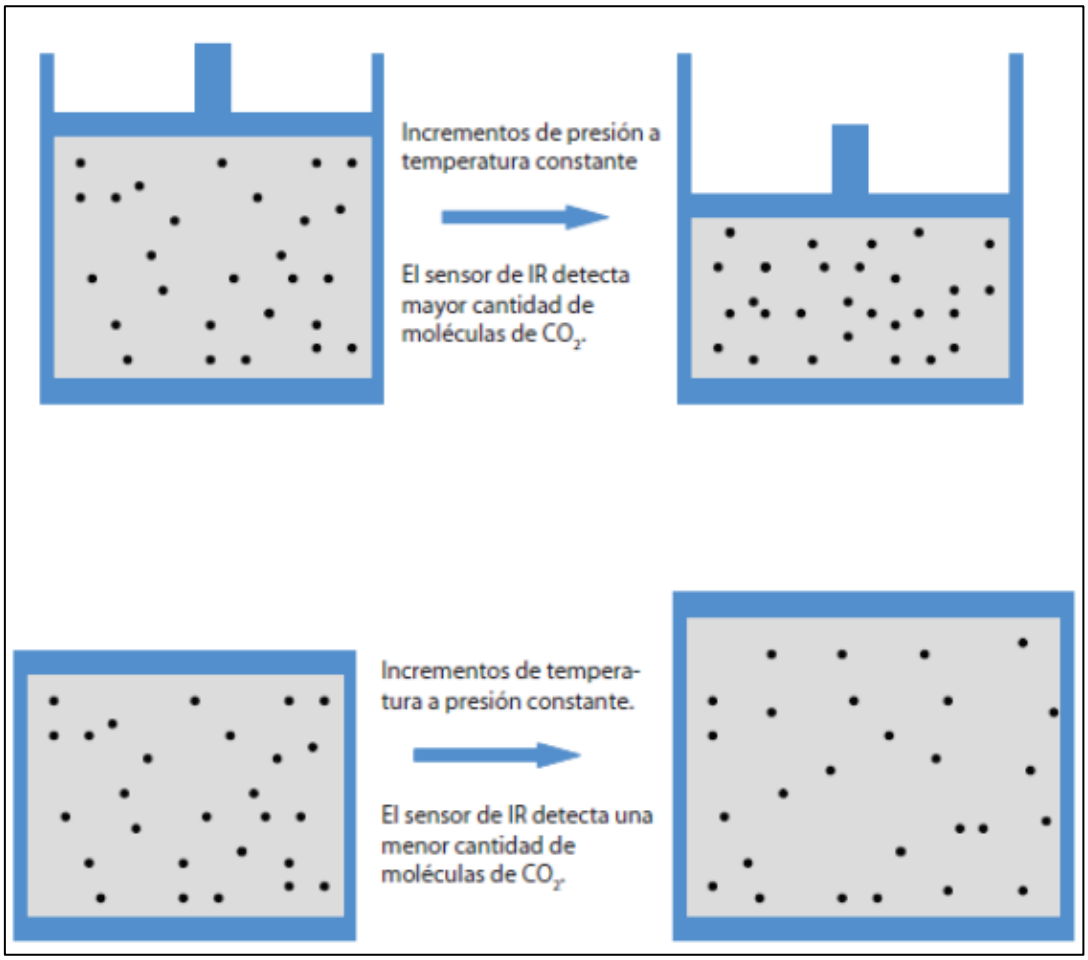
Efecto de los Distintos Niveles de CO2	
Concentración	**Efecto**
350 - 450 ppm	Concentración atmosférica típica
600 – 800 ppm	Calidad del aire interno aceptable
1,000 ppm	Calidad del aire interno tolerable
5,000 ppm	Límite promedio de exposición en un periodo de 8 horas
6,000 – 30,000 ppm	Preocupación, solo exposición breve

A) Temperatura Y Presión Sobre La Medición De CO2.

Cuando cambia la presión o la temperatura, la densidad molecular del gas se modifica. El resultado se observa en la concentración del gas.

Ilustración 14

Efecto de presión y temperatura en la concentración (AKRIBIS, 2021)



Cuando se incrementa la presión, el sensor puede detectar una mayor cantidad de moléculas de CO2. Cuando la temperatura aumenta, el sensor detecta una menor cantidad de moléculas de CO2. Es por eso que el sensor mide también la temperatura y realiza la compensación. Con una diferente presión, debe de ser recalibrado, es este estudio es a presión atmosférica, por lo que no necesitamos una nueva calibración.

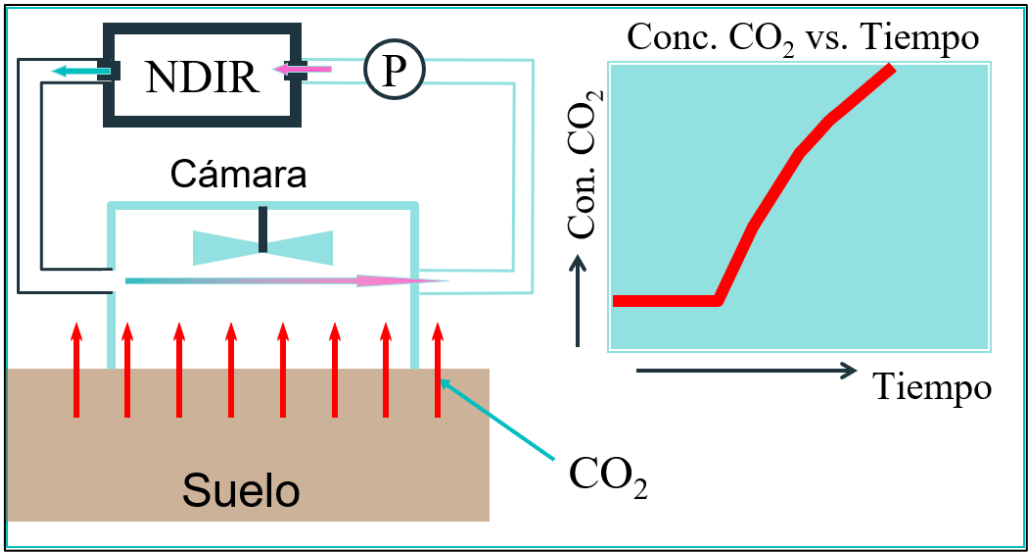
B) ¿Cómo Se Mide El Flujo De CO2 En El Suelo?

El método para trabajar este proyecto es con la recopilación de varios datos de concentración de CO2. Haciendo uso de una cámara de acumulación, se provoca que la concentración dentro de

esta cámara, aumente y acumule cada vez más partículas de CO2 dentro de la cámara. Al graficar la concentración acumulada respecto al tiempo, se visualizará la tendencia de incremento. Aplicando regresión a estos datos, se extrae el valor con el que se comporta la emisión de CO2. En otras palabras, el *flujo de CO2* del suelo.

Ilustración 15

Esquema de medición de flujo de CO2



En la ilustración 15, se percibe como es el esquema del sistema de medición de flujo de CO2. Dentro la cámara, se acumula el gas que se emite del suelo. Con la ayuda de la bomba, el gas atraviesa el NDIR y hace la lectura de CONCENTRACIÓN, ese dato se almacena y se gráfica. Así sucesivamente con cada dato medido. Con el modelo matemático de la regresión se permite predecir el comportamiento de la concentración en el tiempo, así es como obtenemos el FLUJO de CO2 emitido por el suelo.

CAPÍTULO II

DISEÑO DE HARDWARE, APARAMENTA Y MALETA

La parte física de este equipo está detallado dentro de este capítulo. Es de resaltar varios aspectos a los cuales se deben identificar y diseñar de forma en que se considere el funcionamiento y la manera en que se utilizará el equipo de medición. Las consideraciones ya han sido detalladas en el capítulo I, ahora es el capítulo donde se cubren los elementos materiales que se han instalado en el sistema para la construcción del equipo de medición de flujo de CO2.

2.1 DIMENSIONAMIENTO DE BATERÍA

La importancia de una fuente de alimentación que no esté sujeta a una conexión a la red eléctrica, conlleva a la utilización de una batería recargable. Por la disponibilidad de mercado, su voltaje común de operación y un menor efecto en la contaminación por su manera de reciclaje, la batería seleccionada es una batería de plomo-ácido sellada.

El voltaje de operación del sistema es 12Vdc. Esto se debe a que la bomba, el ventilador y el sensor LI COR 830 pueden trabajar con este nivel de voltaje. En el caso del CPU del equipo, utilizamos un convertidor DC/DC para bajar el nivel de tensión a un nivel adecuado para el microcontrolador y todos los módulos incluidos dentro del CPU.

La siguiente tabla, enlista el consumo y la tensión de operación de cada uno de los elementos eléctricos del sistema, basados en la hoja de datos de cada elemento (*Ver anexos)*:

Tabla 2

Consumo eléctrico de cada elemento del equipo.

Corriente consumida de cada equipo		
Equipo	Corriente (A)	Voltaje (V)
LICOR 830	0.33	12
Bomba de Gas	0.5	12
Ventilador	1.30E-01	12
CPU	-	-
ATmega328P	1.90E-02	5
HC-06	4.0E-02	5
NEO 6M	6.7E-02	5
RS232 a TTL	1.0E-02	5
Relé 5V 1Canal	7.0E-02	5
DHT22	1.50E-03	5

Para los elementos que no trabajan a 12V, aplicamos su equivalente de consumo de corriente en 12V para encontrar la corriente total necesaria a alimentar del sistema completo. Esta descrito en la siguiente tabla:

Tabla 3

Consumo eléctrico de elementos a 5vdc

Corriente en 12Vdc CPU				
I1	V1	a	I2	V2
1.9E-02	5	0.417	7.92E-03	12
4.0E-02	5	0.417	1.67E-02	12
6.7E-02	5	0.417	2.79E-02	12
1.0E-02	5	0.417	4.17E-03	12
7.0E-02	5	0.417	2.92E-02	12
1.50E-03	5	0.417	6.25E-04	12

Con todas las corrientes a una misma tensión se obtiene una corriente total de 796mA.

El tiempo estimado para una salida de campo, donde se cubra tranquilamente la recopilación de datos sin tener problemas de quedarse sin carga en la batería, se ha estimado de **6 HORAS** de duración.

Tabla 4

Valores para dimensionamiento de batería.

Valores para dimensionamiento de batería		
I_Total	7.96E-01	A
P_Total	9.5575	W
Horas trabajo	6	h
E_Total	57.345	Wh

El porcentaje de descarga de una batería de plomo ácido sellada es de **60%** (Cambio Energético, 2022). Con (1) se obtiene el tamaño de la batería necesaria para cubrir 6 horas de este equipo de medición.

$$Bat_{Size} = \frac{E_{Total}}{Prof_{Descarga} * 12V} \qquad (1)$$

$$Bat_{Size} = 7.964583 \text{ Ah}$$

Con estos resultados determinados por (1), nos brinda un valor que nos permite seleccionar el uso de una batería de **7Ah** como la fuente de alimentación principal del equipo.

Ilustración 16

Batería para equipo de medición (Steren, s.f.)

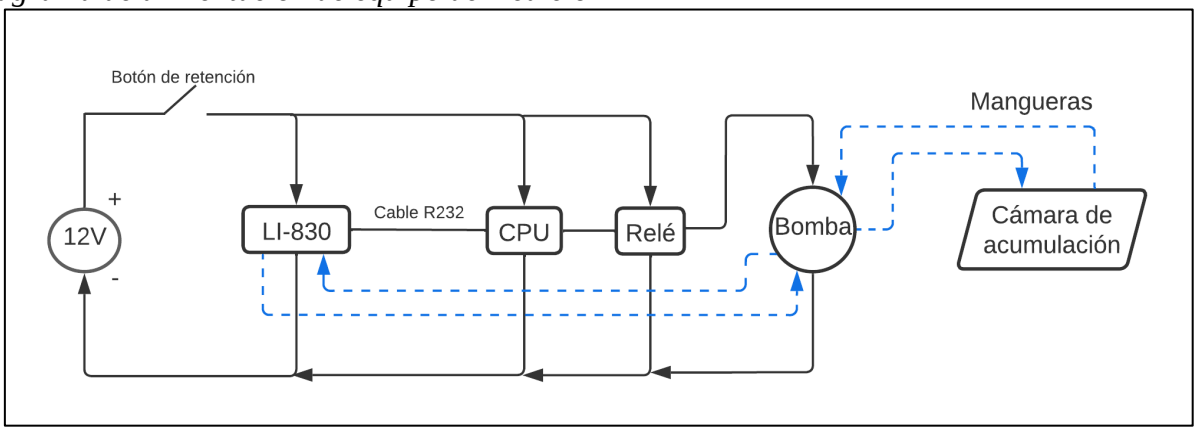
La ilustración 16, muestra la batería que cumple con las características del diseño. Se incorpora un cargador inteligente de batería plomo-ácido, con el fin de tener facilidad carga para toda persona que emplee su uso.

a) Diagrama de Alimentación de Equipo de Medición

En la ilustración 17, se muestra el diagrama de alimentación para el sistema con el que dispone el equipo de medición, ciertos detalles mostrados en el diagrama serán tocados con mayor profundidad en los siguientes capítulos.

Ilustración 17

Diagrama de alimentación de equipo de medición



La batería alimenta a los elementos más grandes del equipo de forma directa, solo la bomba es controlada por medio de un relé controlado por la aplicación Android, esta consideración es debido a que la bomba es el elemento de mayor consumo, y su encendido es solamente en el momento exacto en el que se desee tomar la captura de datos de concentración de CO2.

2.2 MALETA

Otra consideración para la construcción de este equipo de medición, es que cumpla con una facilidad de transporte. Para esto, se ha tomado a bien instalar todo este sistema en una maleta para su movilidad. Para el transporte efectivo del equipo de medición en construcción, se considera que la maleta de transporte, cuente con el espacio adecuado que permita instalar los diferentes elementos que conforman el equipo. Además de considerar sus dimensiones, cabe destacar que la maleta debe ser resistente a golpes y cuente con protecciones internas del equipo. Esta maleta también ha sido adaptada para un transporte cómodo, con la instalación de correas, de tal forma que pueda ser transportado el equipo sobre los hombros dando un estilo de mochila y sobre este, se acomoda la cámara de acumulación. Todo esto considerando siempre que su transporte sea efectivo a cualquier punto de medición que se desee hacer una toma de datos.

Ilustración 18

Maleta vault by pelican v200 para transporte de equipo (PELICAN, 2022)

Esta maleta es de la marca *Pelican* el modelo *Vault V200*. La descripción detallada acerca de esta maleta es la siguiente:

Dimensiones

INTERIOR: 35.6 x 25.4 x 14 cm

EXTERIOR: 39.1 x 33.2 x 15.6 cm

Peso: 2.2 kg

Materiales:

Cuerpo Material: Polietileno

Temperatura: -10°C a 60°C

2.3 CÁMARA DE ACUMULACIÓN

Como parte del método utilizado para la medición del flujo de CO_2 en el suelo, es indispensable contar con una cámara de acumulación del gas emitido del suelo. Esta cámara está hecha de aluminio. No cuenta con ninguna fuga que de salida al gas.

Ilustración 19

Cámara de acumulación enterrada

Cuenta con dos uniones para mangueras. Estas mangueras son las encargadas de transportar el gas con la ayuda de la bomba al sensor de CO2 LI COR 830. Una manguera para la entrada de gas y otra para la salida de este.

Ilustración 20

Cámara de acumulación y equipos

La ilustración 20 muestra físicamente como va instalada junto con la maleta del equipo. Para la recopilación y medición, se entierra una parte de esta cámara con la finalidad de que la fuga de gas sea lo más mínimo posible y acumule todo el gas que emana del suelo.

2.4 ACCESORIOS

Como parte de la documentación y una bitácora completa, incluimos elementos importantes en el equipo que no forman parte del consumo de energía, pero cumplen con una función indispensable dentro del sistema. Su función puede ser descrita brevemente, pero no por ello su función pasa a un segundo plano y no puede pasarse por alto.

Botón de anclaje

La función de este botón de anclaje es permitir y bloquear el paso de corriente al sistema completo. Es el botón de control de general de encendido y apagado de todo el equipo de medición.

Ilustración 21

Botón de anclaje (STEREN, 2022)

Es un botón con iluminación y conectado en su contacto normalmente abierto (NO), para que permita la circulación de la corriente cuando se presione, y se mantiene en esa misma posición hasta que vuelva a presionarse de nuevo. Sus características de fábrica son:

Iluminación color azul o verde

Contacto de aleación de plata

Fabricado con acero inoxidable

Vida útil: 50 000 operaciones

Voltaje de operación: 250 Vca max.

Corriente: 5 A

Voltaje para iluminación: 12 a 24 Vcc

Su ubicación es dentro de la maleta y de fácil acceso para un encendido y apago del equipo. Es el elemento de control y único que debe ser operado dentro de la maleta, todo lo demás está diseñado para que sea controlado desde la aplicación Android.

Cableado y fusible

El sistema de protección del equipo consta de un fusible de 3A, tipo americano de vidrio. Esto porque el convertidor DC/DC soporta hasta 3A corriente máxima.

Ilustración 22

Porta fusible tipo americano

Mientras que el calibre conductor utilizado para alimentar el sistema y conectados a una bornera se distribuye a todo el equipo. El calibre es AWG 18, con capacidad para conducir 5.6 amperios.

2.5 COMPONENTES DE CPU: MÓDULOS

Los módulos que componen la unidad central de procesamientos, CPU, del equipo de medición, son presentados dentro de este capítulo. Se puede encontrar un mayor detalle de estos componentes en los anexos, donde se encuentran las hojas de datos de ellos.

A manera de introducción de esta sección, primero hacemos mención de un poco de teoría:

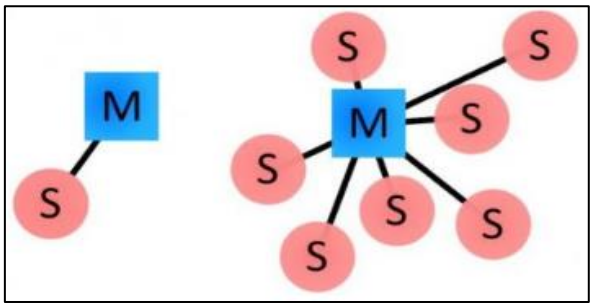
Bluetooth. Como parte de la comunicación inalámbrica necesaria para lograr aprovechar la tecnología que se tiene a disposición, es importante entender cómo funciona esta, BLUETOOH es la tecnología inalámbrica seleccionada a ser utilizada por este sistema de medición. Esto debido a que todos los dispositivos Android cuentan con esta característica y así aprovechamos un medio de comunicación cercana inalámbrico.

Antes de comenzar surge la pregunta ¿Qué es Bluetooth?:

> Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) y trabaja en la banda ISM de los 2.4GHz. Los dispositivos Bluetooth pueden actuar como Masters o como Slaves. La diferencia es que un Bluetooth Slave solo puede conectarse a un master y a nadie más, en cambio un master Bluetooth, puede conectarse a varios Slaves, recibir y solicitar información de todos ellos, arbitrando las transferencias de información (Hasta un máximo de 7 Slaves) (Loachamin, s.f.).

Ilustración 23

Slaves (Esclavos) y Masters (Maestros) en conexiones Bluetooth (Loachamin, s.f.)



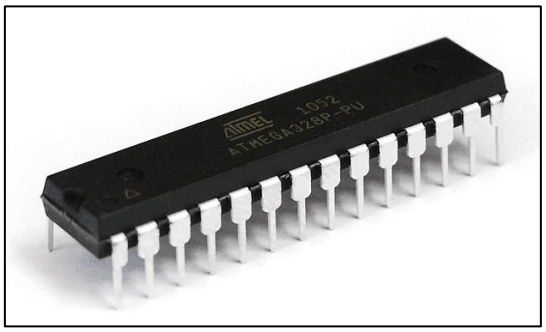
ATMega328P

Este es el microcontrolador seleccionado para ser el centro del CPU. Es un chip microcontrolador de 8-bits con 32kBytes de memoria Flash creado por Atmel.

> Es una computadora de muy baja potencia, pero dedicada a aplicaciones muy específicas. Estos microcontroladores pueden hacer diversas tareas, llevar cuentas, hacer cálculos simples y no tan simples, pueden comunicarse con el usuario, pueden censar, pueden actuar y pueden comunicarse con otros microcontroladores. La plataforma de Arduino u otras plataformas similares nos acercan de una manera fácil y sencilla la programación y el desarrollo de prototipos (Sánchez, 2022).

Ilustración 24

Microcontrolador ATMega328P



La selección de este microcontrolador se definió por su disponibilidad en el mercando. También por su familiaridad y su versatilidad en la programación. Cuenta con la capacidad de

ser programado por medio del IDE de Arduino y otros IDE similares y acordes. Es un chip comúnmente utilizado en distintos sistemas donde se requiere un microcontrolador simple, de bajo consumo y bajo costo.

CPU: Es la parte del microcontrolador encargada de hacer las operaciones matemáticas o cálculos. Sera el encargado de cumplir todas las instrucciones que el usuario cargue en el microcontrolador.

Reloj Base: El reloj es el que marca el pulso o ritmo. “La frecuencia de reloj nos puede dar una idea aproximada o proporcional de la velocidad del microcontrolador” (Sánchez, 2022).

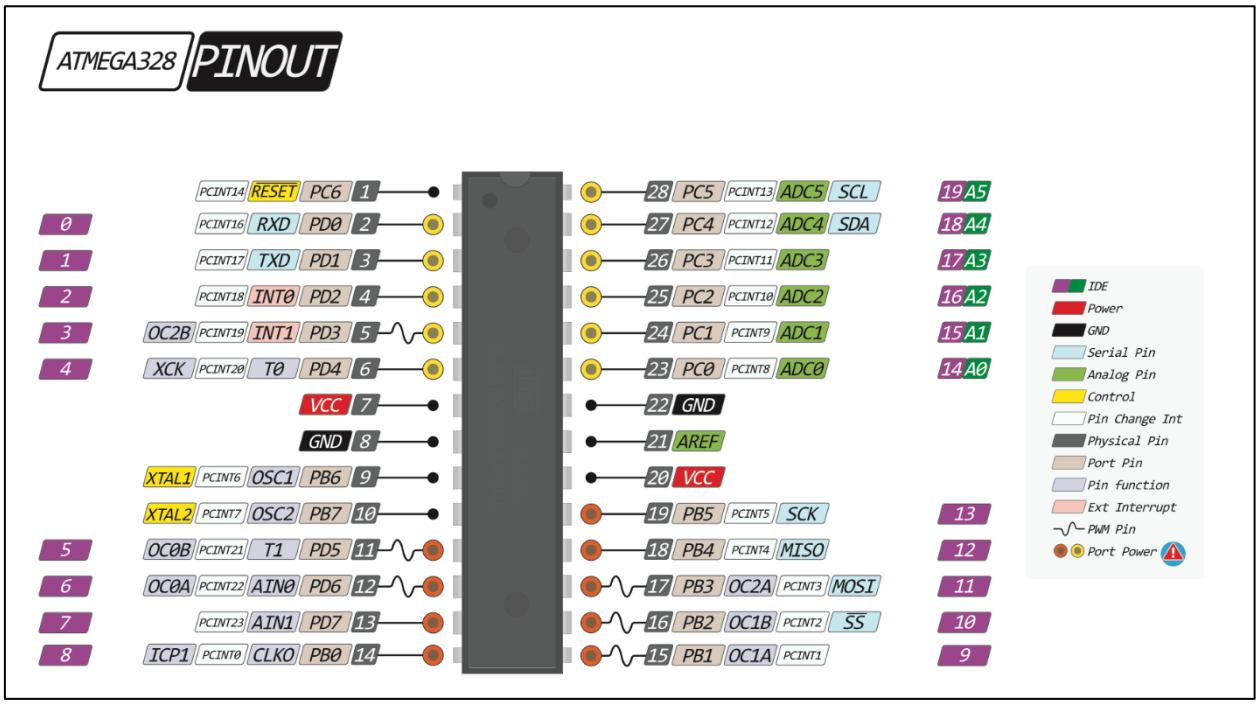
EEPROM: La memoria EEPROM tiene la principal función de almacenar las instrucciones que deberá cumplir el microcontrolador. Estas instrucciones serán las que el usuario generara desde la plataforma de desarrollo (IDE). Su principal característica de la EEPROM es que no se borra (No volátil) cuando el usuario le quita la energía al dispositivo.

RAM: Esta memoria es volátil, se borra cada vez que el usuario le quita la energía al dispositivo. Es una memoria muy rápida. Para que se utiliza: “si por ejemplo hay un proceso que se repetirá 10 veces, entonces tendremos que llevar una cuenta interna para ir contando 1, 2, 3, … 10, y esta cuenta no se necesita guardar” (Sánchez, 2022). En la memoria RAM se almacenarán todas las variables de uso cotidiano de la aplicación que carguemos.

Puertos I/O: Los puertos son los pines que tiene el microcontrolador. Estos son el medio de comunicación que tiene el microcontrolador para sacar o para recibir información. Esta información puede ser digital, es decir un 1 o un 0 (estado alto o estado bajo), el pin tendrá valores de 0 o 5 V, o puede ser información analógica, estos puertos pueden reproducir una tensión analógica como 2.7 V, 3.2 V y así valores entre el 0 y 5V.

Ilustración 25

Pines de chip ATMega328P (Naylamp Mechatronics, s.f.)



Las características más destacadas de este microcontrolador son:

32 kbytes de Memoria Flash Programable

2 kbytes de SRAM

Frecuencia máxima de operación 20MHz

CPU 8-bits AVR

1 kbytes de EEPROM

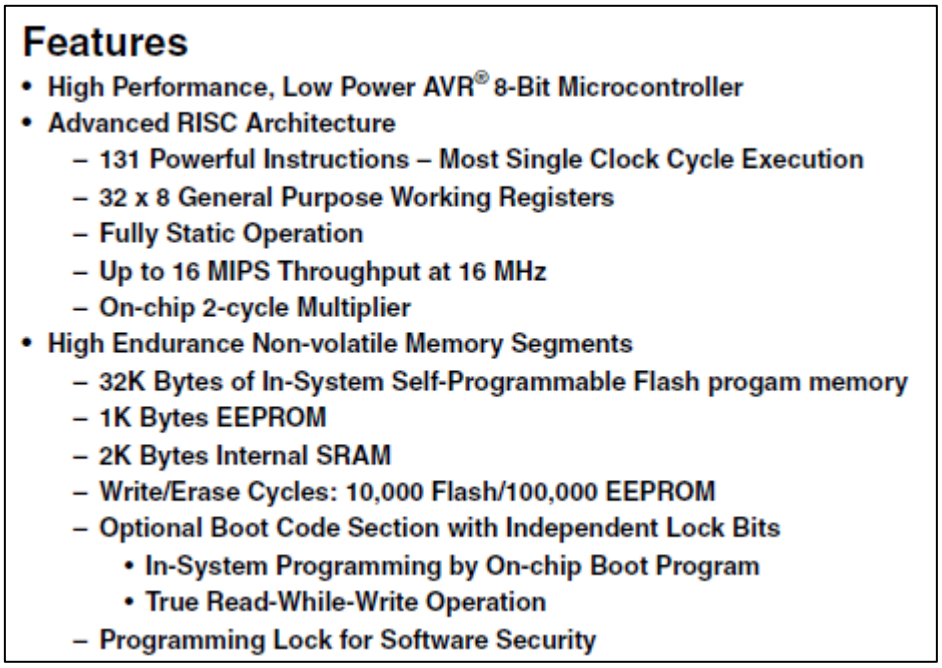
Voltaje de entrada: 2.7V-5.5V

Rango de Temperatura de operación: -40°C a +125°C

También incluimos un resumen de lo más importante incluido en la hoja de datos del ATMega328P.

Ilustración 26

Hoja de datos de ATMega328P



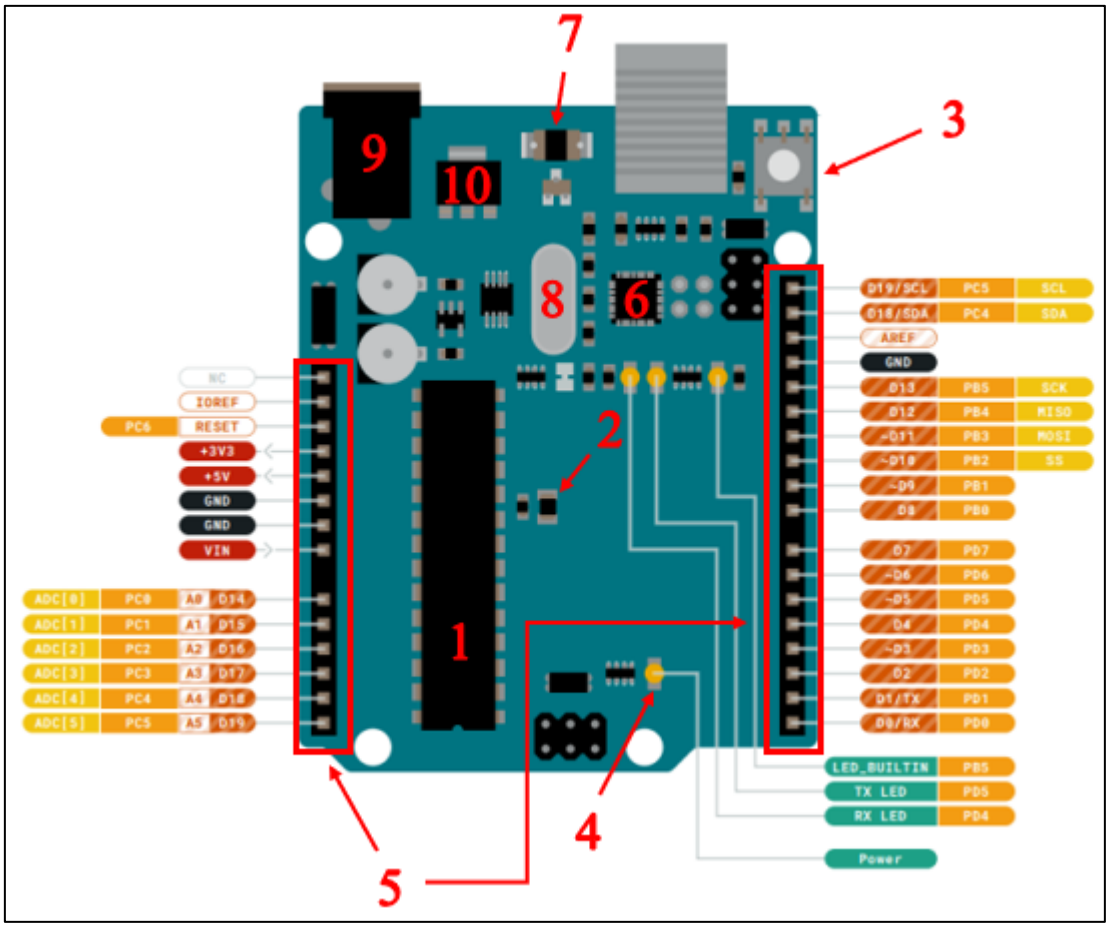
Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32K Bytes of In-System Self-Programmable Flash progam memory
  - 1K Bytes EEPROM
  - 2K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security

Placa Microcontrolador

Ilustración 27

Microcontrolador ATMega328P montado sobre placa Arduino UNO (ARDUINO, 2022)



1. **Microcontrolador ATmega328P.**
2. **Cristal de 16MHz**, esta frecuencia es la que tomara el reloj base del microcontrolador como referencia.
3. **Pulsador**, reinicia el funcionamiento el microcontrolador.
4. **Led** de encendido de la placa.
5. **Headers**, están conectados eléctricamente a los pines del microcontrolador. La forma en la que están dispuestos es para permitirnos conectar desde la parte de

arriba de manera apilada distintas placas que le agregan funcionalidades a la placa de Arduino.

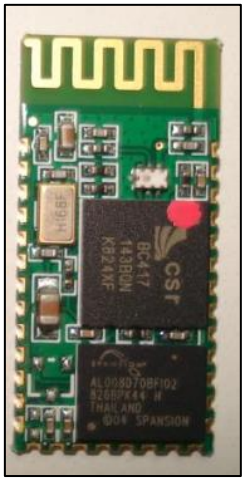
6. **Interface USB-Serial TTL**, Este integrado hace de traductor entre la norma USB y una serial TTL. Debido a esto podemos grabar el programa al microcontrolador desde una PC a través de un cable USB.
7. **Fusible de montaje superficial**, Nos protege de que en caso por error ocurra un cortocircuito en la placa y evitaría dañar el puerto USB de nuestra PC.
8. **Cristal de 16MHz**, da la frecuencia de reloj al integrado USB-Serial TTL.
9. **Power jack 2.1x5.5mm**, nos permite alimentar la placa en el caso que no usemos el conector USB. Aquí puede ingresar niveles de tensión entre 5V a 17V.
10. **Regulador de tensión del puerto Jack**, reduce la tensión de entrada a 5V.

HC 06

Este es el módulo Bluetooth. Es el encargado de vincular el CPU con el dispositivo Android a través de conexión bluetooth.

Ilustración 28

Módulo HC06 (Bluetooth) (Guangzhou Tecnology Co.Ltd)



Cuenta con los pines Rx y Tx. Los cuales son los encargados de la transmisión de datos, Transmisión y Recepción. Son por medio de estos pines por los que transmite la información el microcontrolador ATMega328P. Las características que más cabe resaltar son (Guangzhou Tecnology Co.Ltd):

Voltaje de entrada: 3.3V

Rango de Temperatura de operación: -25°C a +75°C

Sensibilidad: -80dBm

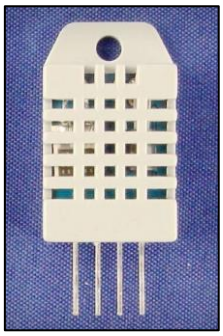
Este módulo solo puede trabajar como esclavo, el dispositivo Android es el maestro al cuál se conecta el HC06.

DHT22

El siguiente módulo es el sensor de temperatura y humedad DHT22. Este módulo es encargado de monitorear el rendimiento de la CPU. Mide el valor de temperatura y humedad del ambiente dentro del CPU y lo transmite al microcontrolador, que se mantiene actualizando su dato cada 10 segundos. Estos datos son presentados dentro de la aplicación Android en dos gráficas. La prioridad es la información obtenida por el LI COR 830, la cual es enviada cada 1 segundo, es por ello que el dato de rendimiento, temperatura y humedad, es cada 10 segundos.

Ilustración 29

Módulo de sensor y temperatura DHT22 (Aosong Electronics Co.,Ltd)



Dentro de las características más relevantes de este módulo, cabe destacar las siguientes especificaciones (Aosong Electronics Co.,Ltd):

Voltaje de entrada: 3.3V-6V

Rango de Temperatura de operación: -40°C a +80°C

Rango de Humedad de operación: 0 a 100%

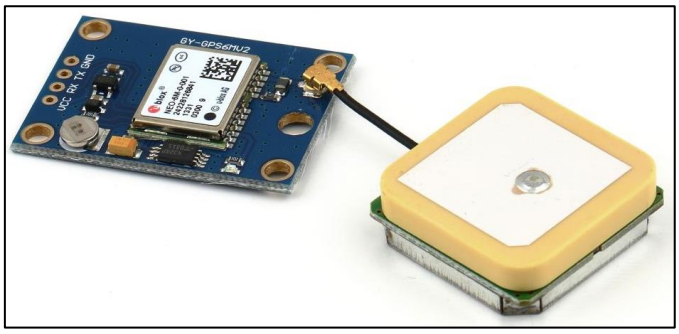
Cuenta con un coeficiente para calibración almacenado en su memoria. Con esto, tiene la capacidad de calibrarse automáticamente por medio de este coeficiente.

Neo 6M

El módulo NEO 6M, es un módulo de geolocalización. El cual “viene con un módulo de serie U-Blox NEO 6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica” (Naylamp Mechatronics, s.f.).

Ilustración 30

Módulo GPS NEO 6M



Puede alcanzar una precisión de hasta 3 metros (Llamas, Localización gps con arduino y los modulos gps neo-6, 2016). Su función es la de brindar el dato de la ubicación geográfica en la que se encuentra específicamente el punto de medición del flujo de CO2.

Con este módulo incluido, se obtienen las coordenadas del sitio muestreado, para posteriormente hacer un mapa del sitio. Las características más relevantes de este módulo son (Naylamp Mechatronics, s.f.):

Voltaje de entrada: 3.3V

Ganancia Max. De antena 50dB

Sensibilidad -161dBm

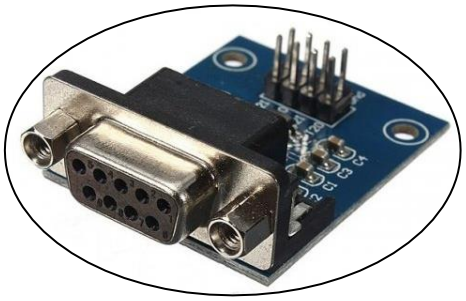
Altura Máxima: 5,000 metro sobre el nivel del mar

MAX 232

La conexión entre el microcontrolador y el sensor de gas de CO2 LI-830 se hace a través de una comunicación serial a través de un puerto serie RS232 del sensor, pero esta conexión no se hace de manera directa, debido a que el protocolo de comunicación RS232 emplea variaciones de voltaje de entre -13V a 13V y el microcontrolador suele trabajar a un nivel de voltaje TTL(Transistor-transistor logic) de 3.3V/5V. TTL nos quiere decir que la comunicación se

realiza mediante variaciones en la señal entre 0V y Vcc (Llamas, Comunicación de arduino con puerto serie, 2014). Para llevar a cabo esta comunicación se empleó un convertidor de RS232 a TTL.

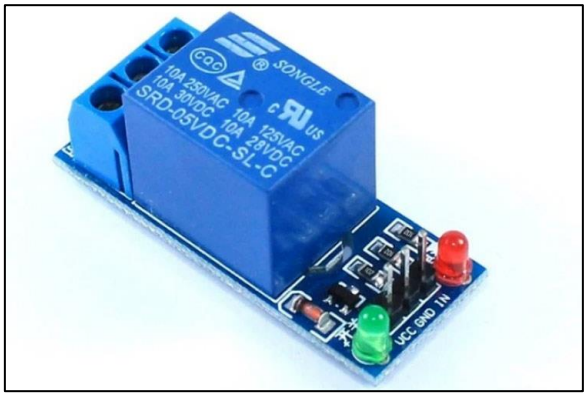
Ilustración 31
Módulo Convertidor RS232 a TTL MAX232 (Maxim Integrated Products, 2003)



Este es el convertidor de RS232 a TTL, MAX232. Su conexión es por medio de un cable Db9, del puerto de salida RS232 encontrado en el LI COR 830 a este pequeño módulo convertidor. Su voltaje de entrada es de 5Vdc. (Maxim Integrated Products, 2003)

Relé 5V 1 Canal

Ilustración 32
Módulo Relé 5V 1 Canal



Este módulo se encarga de abrir y cerrar el paso de la alimentación hacia la bomba del equipo de medición. Es controlado por medio de una señal enviada desde la aplicación Android. Su instalación dentro del equipo, es el uso eficiente de la bomba, el cuál es el elemento que más

consume energía de la batería. Así su consumo es solamente cuando lo indique el operador del equipo a través de la app móvil.

Sus características más importantes son (Future Electronic Corporation):

Voltaje de entrada: 5V

Voltaje máximo de contacto de relé: 250Vac o 30Vdc

Corriente Max de relé: 10A

LM2596

El voltaje del sistema para este equipo es de 12Vdc, con un CPU que sus componentes trabajan con 5Vdc. Ante este inconveniente, la entrada de 12Vdc tiene que ser ajustada a un nivel menor que sea adecuado para alimentar el CPU. Una característica del ATMega238P es que posee pines que permiten alimentar otros módulos, estos pines Vcc 5Vdc, Vcc 3.3Vdc y GND. Es necesario entonces ajustar solamente los 12Vdc para alimentar el microcontrolador ATMega238P, pues este se encarga de alimentar los demás componentes dentro del CPU.

Para este proceso está la implementación del módulo LM2596, el cual es un convertidor de DC/DC.

Ilustración 33

Módulo convertidor LM2596

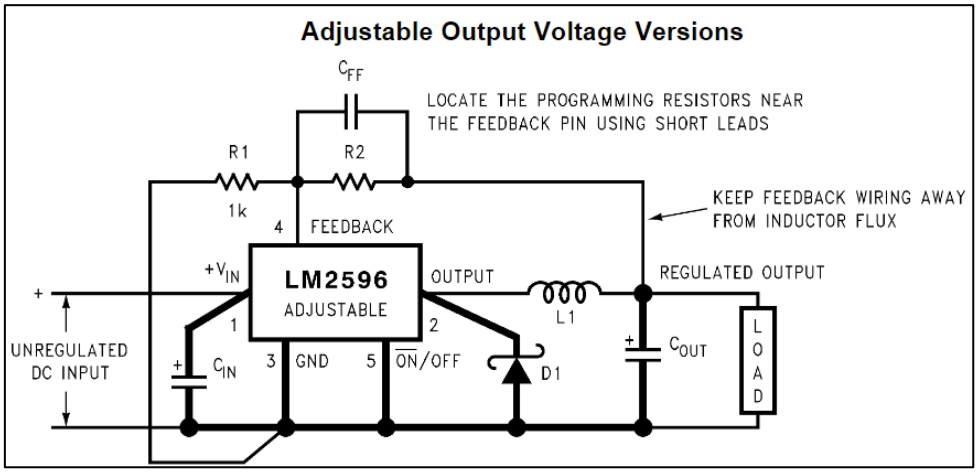
Sus características consideradas para su son (Texas Instruments , 2013):

Salida ajustable a 1.2V-37V

Máx. Corriente de salida 3A

Ilustración 34

Diagrama de módulo convertidor DC/DC (Texas Instruments , 2013)



El diagrama de este módulo explica cómo se ajusta el valor de voltaje en los contactos de salida del módulo. El fabricante coloca R1=1kΩ y con las ecuaciones:

Ilustración 35

Ecuaciones para ajustar V_out del módulo (Texas Instruments , 2013)

$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right)$$

where $V_{REF} = 1.23V$

$$R_2 = R_1\left(\frac{V_{OUT}}{V_{REF}} - 1\right)$$

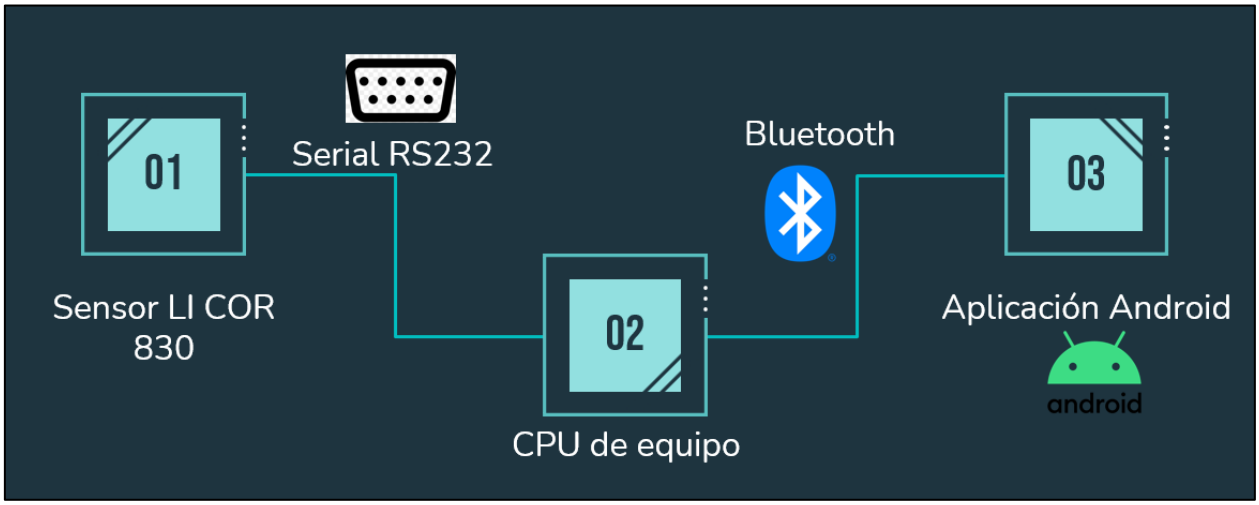
Estas ecuaciones se encuentran en la hoja de datos del módulo. Lo que nos permite ajustar R2 para obtener V_out adecuado para poner a trabajar el microcontrolador. R2 es un potenciómetro, lo que beneficia por la facilidad de ajuste.

2.6 METODOLOGÍA DE COMUNICACIÓN

La manera simplificada de exponer como está comunicado este sistema de medición lo detalla con facilidad el siguiente diagrama:

Ilustración 36

Diagrama de metodología de comunicación



Se observa que el primer elemento encargado de comunicarse o de enviar datos es el Sensor LI COR 830. Este se comunica con el CPU del equipo por medio de un cable con el estándar de conexión RS232.

El CPU se encarga de leer, procesar y enviar los datos proporcionados por el sensor de CO2, LI COR 830, y luego los envía a la aplicación Android. Este último canal de comunicación para la transmisión de datos es por medio de bluetooth.

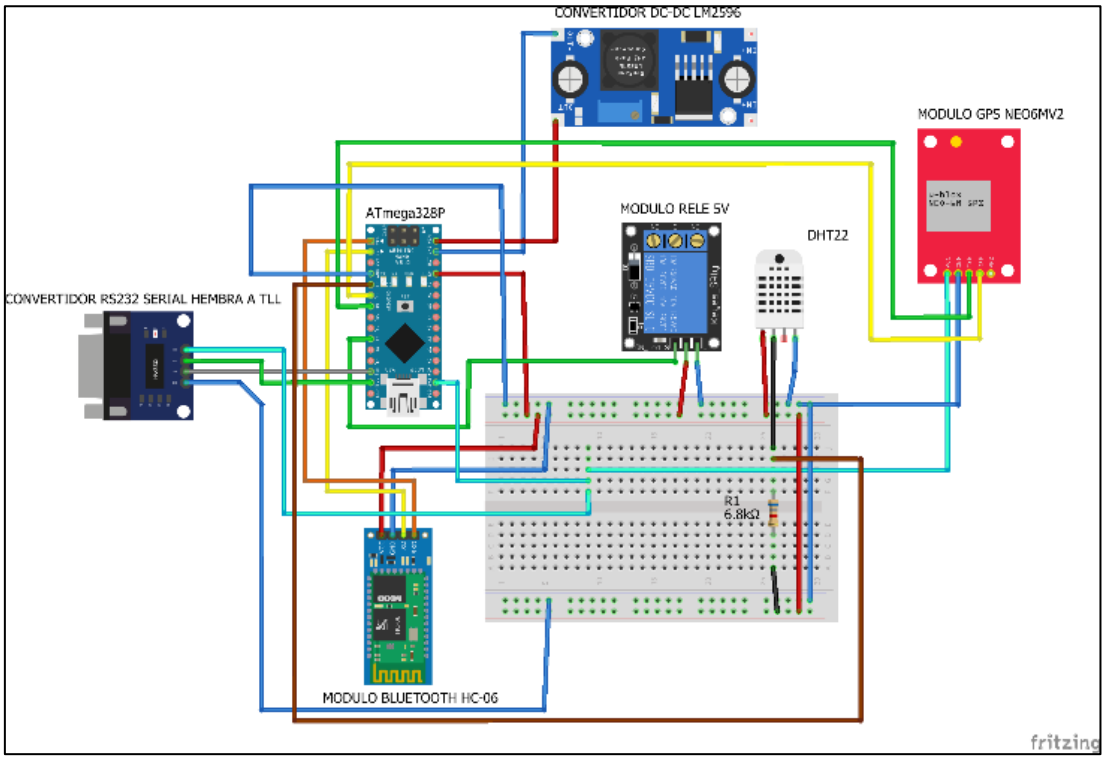
Finalmente, la aplicación Android, muestra los datos, los almacena y procesa para obtener los resultados finales de medición. Es desde la aplicación Android que se observa, controla y opera el sistema de medición de CO2.

2.7 DIAGRAMA DE CPU

Después de introducir cada uno de los elementos que componen el CPU del equipo de medición, es necesario introducirse al tema de cómo se conectan internamente. Para una visualización estéticamente mejor y más agradable a la vista, se utilizó la herramienta Fritzing. Con este programa se incluyeron todos los módulos dentro del CPU y la manera en la que están conectados.

Ilustración 37

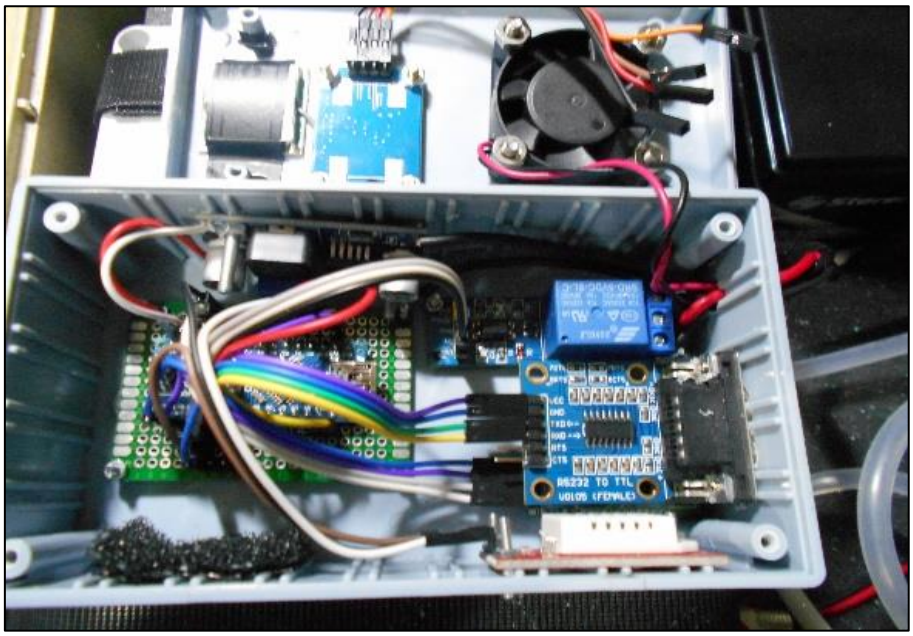
Diagrama de CPU



Todo este cerebro del equipo, se encuentra dentro de un gabinete de plástico, cada uno atornillado a una base e interconectado por medio de buses.

Ilustración 38

Fotografía de CPU dentro de gabinete



La imagen nos ayuda a visualizar como está instalado dentro de un gabinete de plástico el CPU del equipo. Observando la imagen, se aclara la importancia de representar el diagrama haciendo uso del programa Fritzing.

Las siguientes tablas, son una descripción detallada de las conexiones existentes por cada módulo, para una mejor comprensión de las interconexiones dentro del CPU.

Tabla 5

Conexiones de ATMega328P

PIN	CONECTADO 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

Tabla 6

Conexiones Bluetooth HC06

PIN	CONECTADO A:	MÓDULO	PIN
Vcc		ATMega328P	5V
GND		ATMega328P	GND
TXD		ATMega328P	RX0
RXD		ATMega328P	TX1

Tabla 7

Conexiones Convertidor RS232 a TTL

PIN	CONECTADO A:	MÓDULO	PIN
Vcc		ATMega328P	3V3
GND		ATMega328P	GND
RXD		ATMega328P	D11
TXD		ATMega328P	D10

Tabla 8

Conexiones módulo GPS NEO6M

PIN	CONECTADO A:	MÓDULO	PIN
Vcc		ATMega328P	3V3
GND		ATMega328P	GND
TXD		ATMega328P	D4
RXD		ATMega328P	D3

Tabla 9

Conexiones módulo Relé

PIN	CONECTADO A:	MÓDULO	PIN
Vcc		ATMega328P	5V
GND		ATMega328P	GND
IN		ATMega328P	D7

Tabla 10

Conexiones Sensor Temperatura y Humedad

PIN	CONECTADO A:	MÓDULO	PIN
Vcc		ATMega328P	5V
GND		ATMega328P	GND
DATA		ATMega328P	D2

Tabla 11

Conexiones Convertidor DC/DC

PIN	CONECTADO A:	MÓDULO	PIN
Out+		ATMega328P	VIN
Out-		ATMega328P	GND

CAPÍTULO III

DISEÑO DE SOFTWARE

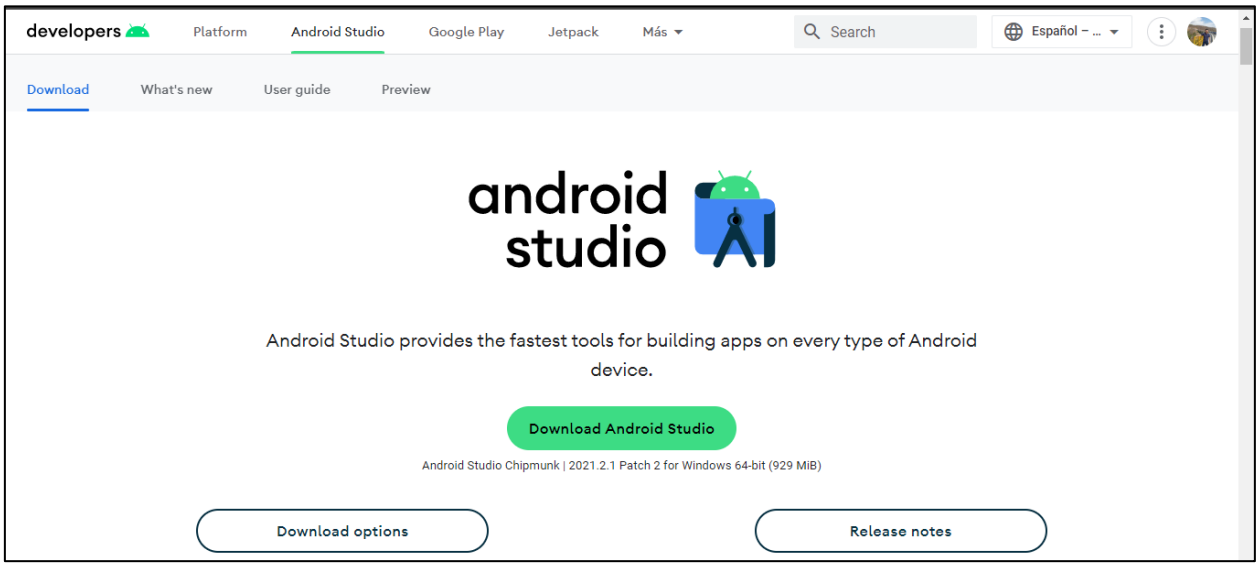
INTRODUCCIÓN A ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA. Android Studio ofrece funciones que aumentan la productividad al momento en que se desarrollan apps para Android, tales funciones son las siguientes: Un sistema de compilación flexible basado en Gradle, un emulador rápido y cargado de funciones, un entorno unificado en donde se puede desarrollar para todos los dispositivos Android, Integración con GitHub y plantillas de código para compilar funciones de apps comunes y también importar código de muestra, compatibilidad con C++ y NDK, etc.

Para descargar Android Studio solo ingresamos al sitio oficial de Android Studio y escogemos la versión del instalador de acuerdo a nuestro sistema operativo.

Ilustración 39

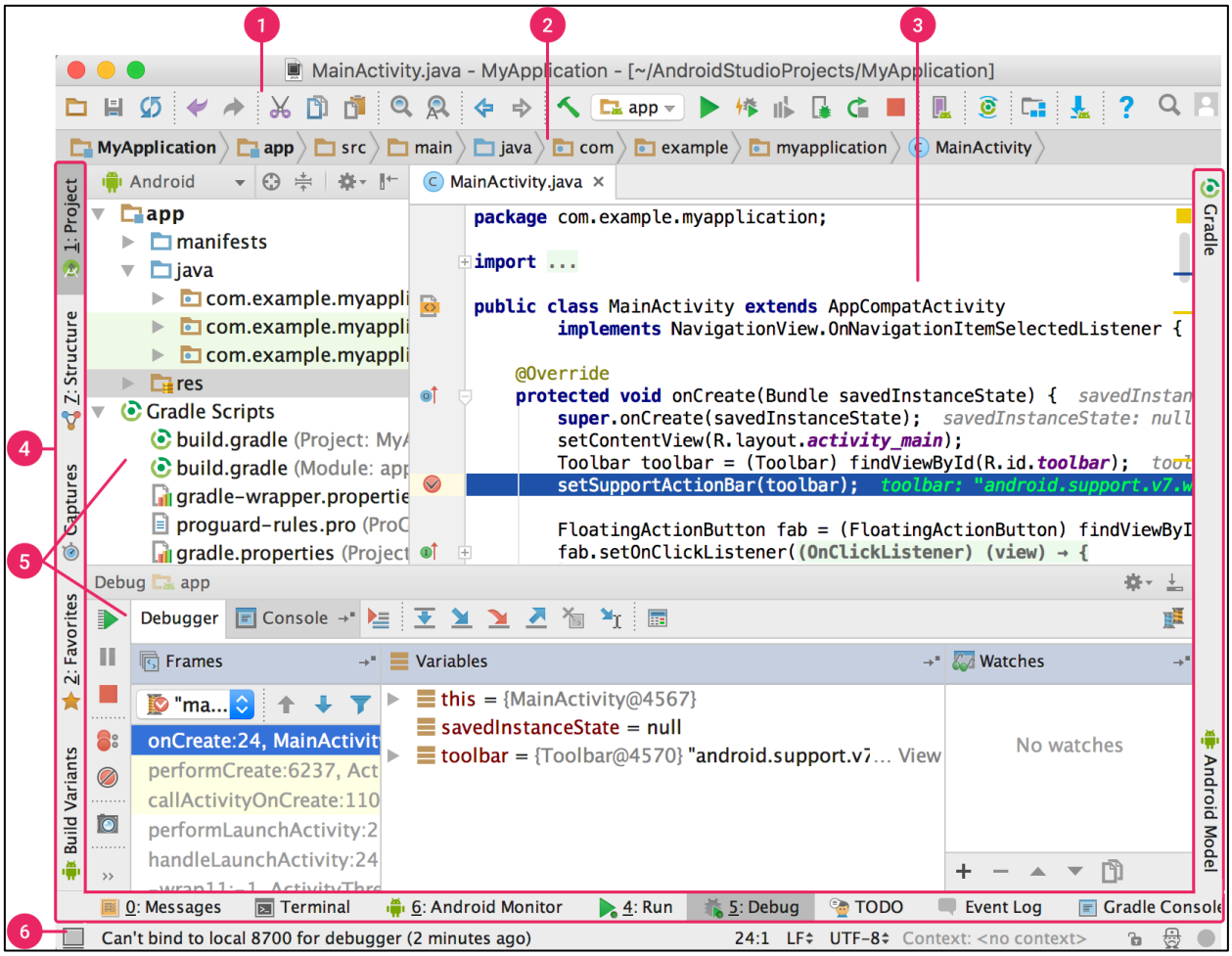
Vista de página web para descargar Android Studio



La interfaz de usuario de Android Studio consta de diferentes áreas lógicas las cuales se detalla en la siguiente imagen:

Ilustración 40

Interfaz de Android Studio (Developers, s.f.)



1. La **barra de herramientas** permite realizar una gran variedad de acciones, tales como ejecutar la app e iniciar las herramientas de Android.
2. La **barra de navegación** nos ayuda a explorar el proyecto y abrir archivos para editar. Y esta barra proporciona una vista más compacta de la estructura visible en la ventana Project.

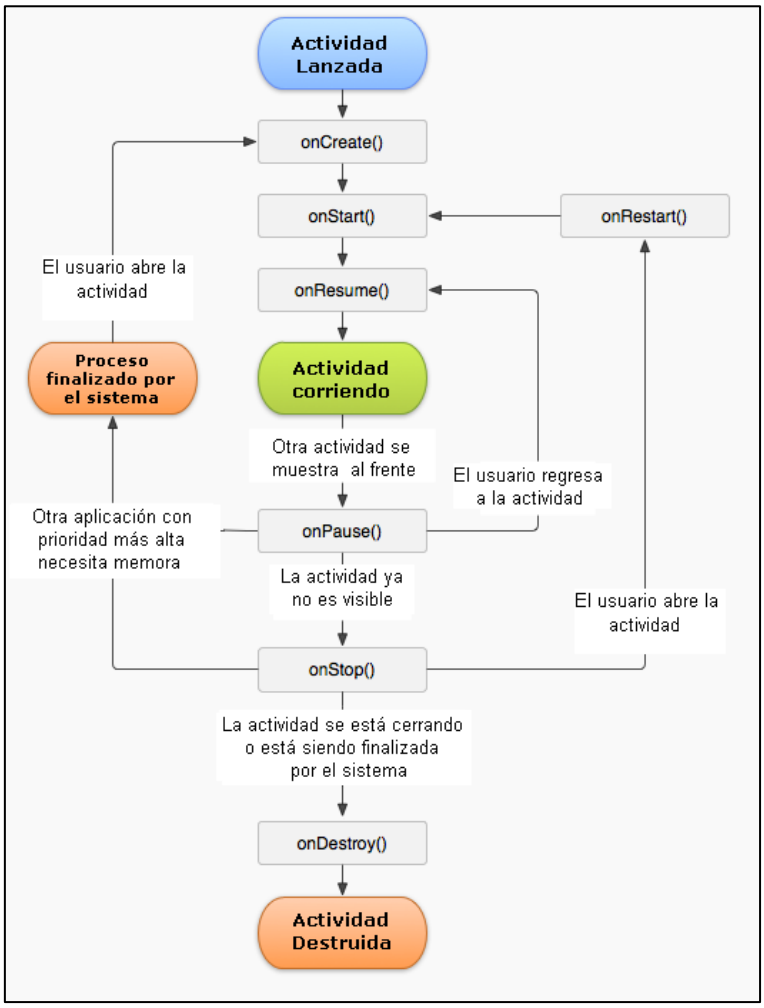
3. La **ventana del editor** es el área en la que se escribe y modifica el código. Esta ventana puede cambiar cuando estamos diseñando la app visualmente.
4. La **barra de la ventana de herramientas** contiene los botones de herramientas individuales.
5. Las **ventanas de herramientas** brindan acceso a tareas específicas, como la administración de proyectos, la búsqueda, el control de versiones, entre otras.
6. En la **barra de estado** se muestra el estado del proyecto y el IDE, además de que en ella se muestran advertencias o mensajes.

Ciclo de vida de una actividad

Una actividad proporciona la ventana en la que la aplicación dibuja su interfaz gráfica. El ciclo de vida de un actividad o actividad hace referencia a los distintos estados por los que va pasar una actividad mientras es utilizada por el usuario desde el momento que se inicia hasta que se cierra (Cursa, s.f.). Para poder comprender estos distintos estados nos apoyaremos del siguiente diagrama de flujo:

Ilustración 41

Diagrama del ciclo de vida de una actividad



Suponiendo que abrimos la app móvil en ese instante lo primero que se ejecutará será el método *OnCreate*, lo que permite este método es crear la actividad, es de gran importancia ya que si no se coloca en el código la actividad no va a funcionar. Una vez creado la actividad se ejecutará el método *onStart* este método va iniciar la actividad que previamente se creó y posteriormente se ejecutara el método *onResume*, el cual permite visualizar la actividad que previamente se creó y se inició, sin este método el usuario no podrá ver la actividad que queremos mostrar. Hasta este punto la actividad ya se está ejecutando y justamente aquí empiezan los distintos caminos que se pueden seguir.

En primer lugar, suponiendo que el usuario minimiza el actividad o aplicación, en ese instante se ejecutara el método *onPause,* este método lo que hace es pausar momentáneamente la actividad haciendo que pase de primer plano a segundo plano, después de esto se ejecutara el método *onStop,* este lo que hará será ocultar la aplicación o en este caso la actividad. En este momento existen dos caminos el primero es que el usuario cierre la aplicación, para lo cual se ejecutara el método *onDestroy* el cual cierra la actividad con el que se está trabajando. Por el otro lado está la parte donde el usuario minimiza la aplicación para ir a revisar otra aplicación y nuevamente cuando regresa a esta aplicación que estaba utilizando lo que se sucederá es que el método *onStop* mandara un aviso al método *onStart* diciéndole que vuelva a iniciar la actividad y así nuevamente se está ejecutando la aplicación.

Implementación de graficas en Android Studio.

Al momento de crear nuestro proyecto en Android Studio aparecen dos ventanas una en la que podremos diseñar la interfaz gráfica de nuestra aplicación y otra en la que podremos crear el código de las funciones que se ejecutaran en nuestra aplicación. En este apartado detallamos el actividad_main.xml el cual nos permite poder diseñar el aspecto de nuestra app.

Como primer paso otorgarnos los permisos a la aplicación, como lo son los permisos Bluetooth y de almacenamiento interno del teléfono. Esto lo hacemos desde el *Manifest* en la ventana de herramientas.

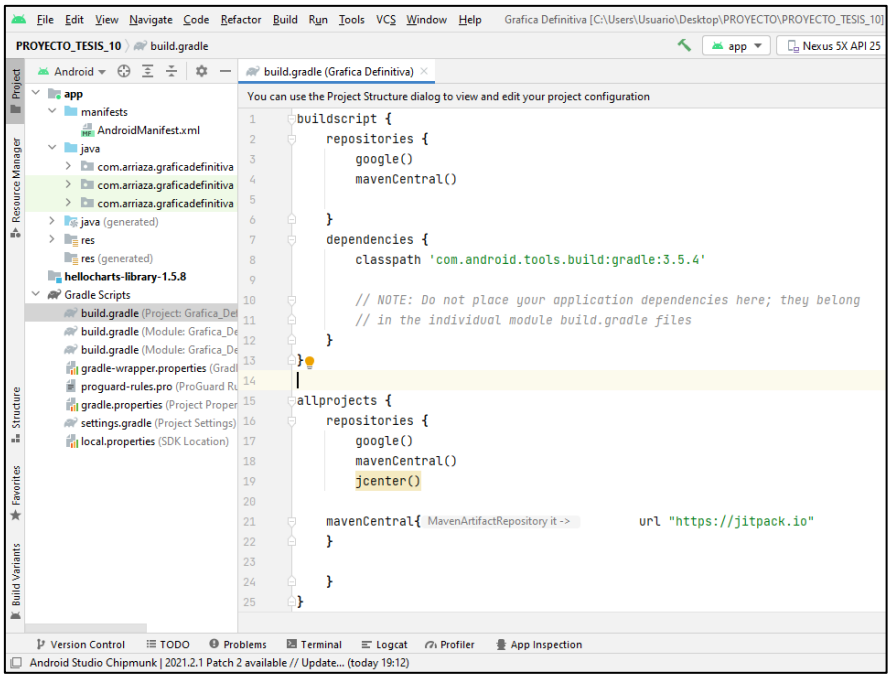
Ilustración 42

Código: permisos para aplicación móvil

Luego en la sección de Gradle en el build.gradle Project digitamos lo siguiente para poder obtener compatibilidad con las librerías de las gráficas de GitHub que se usaron en el proyecto.

Ilustración 43

Código: Compatibilidad de librerías



En el build.gradle Module digitamos lo siguiente:

Ilustración 44

Código: build.gradle 1

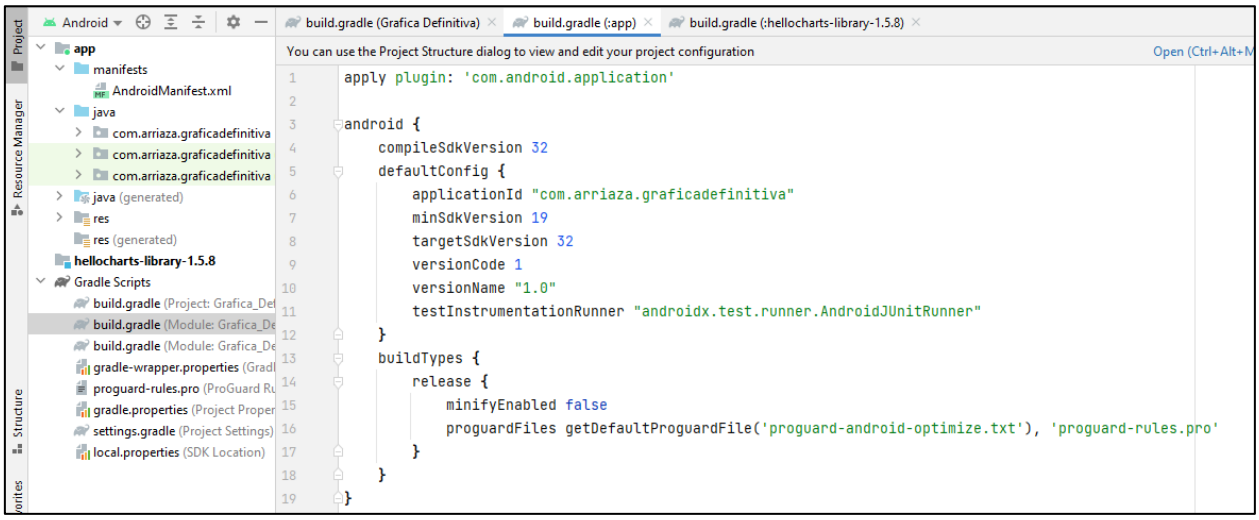
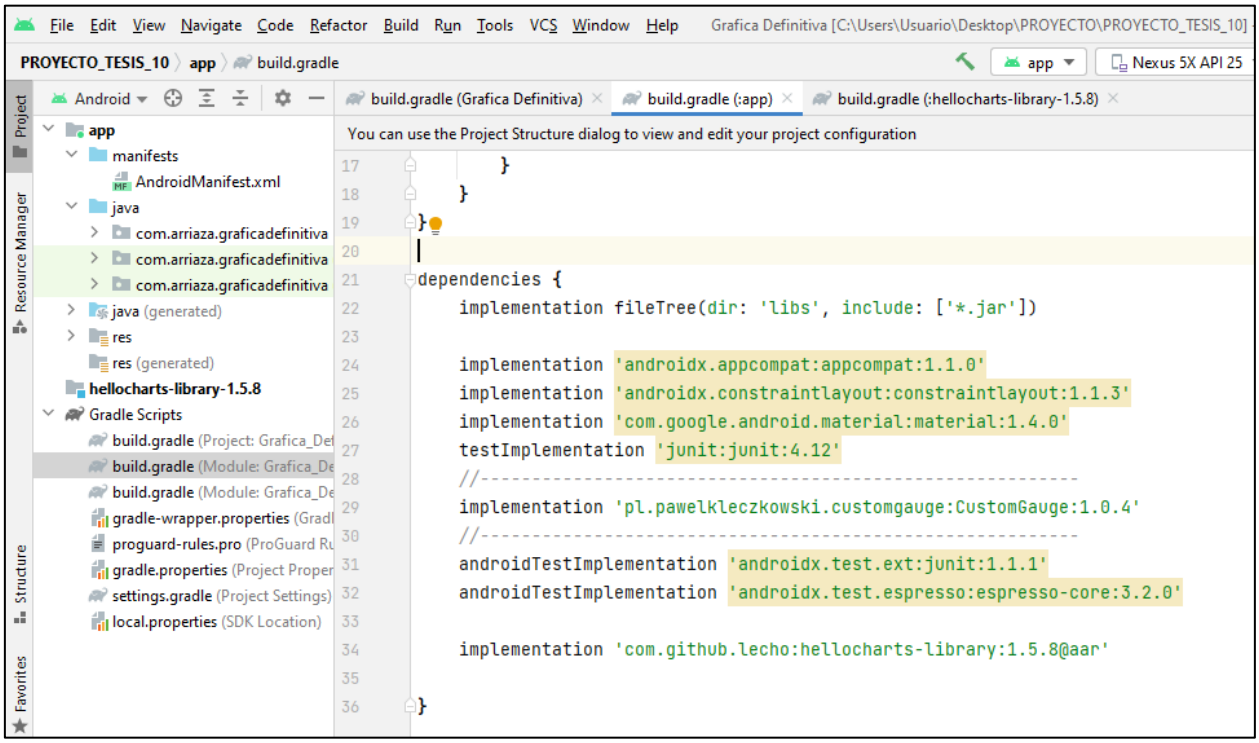


Ilustración 45

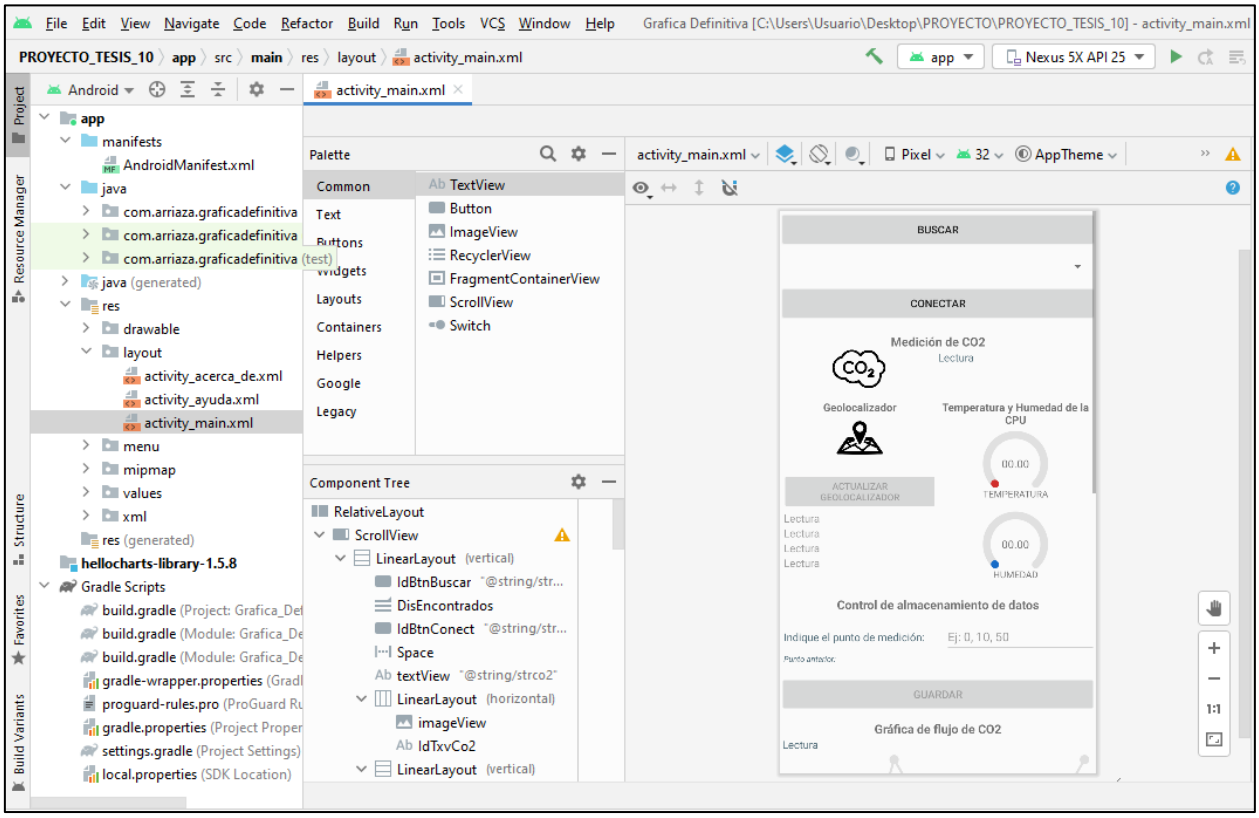
Código: build.gradle 2



En la carpeta *res* encontramos distintas carpetas en las que podemos ir diseñando el aspecto de la app. Específicamente en la carpeta layout. En el layout podemos ir escogiendo diferentes componentes desde la paleta, aquí encontramos botones, texto, imágenes a insertar, etc.

Ilustración 46

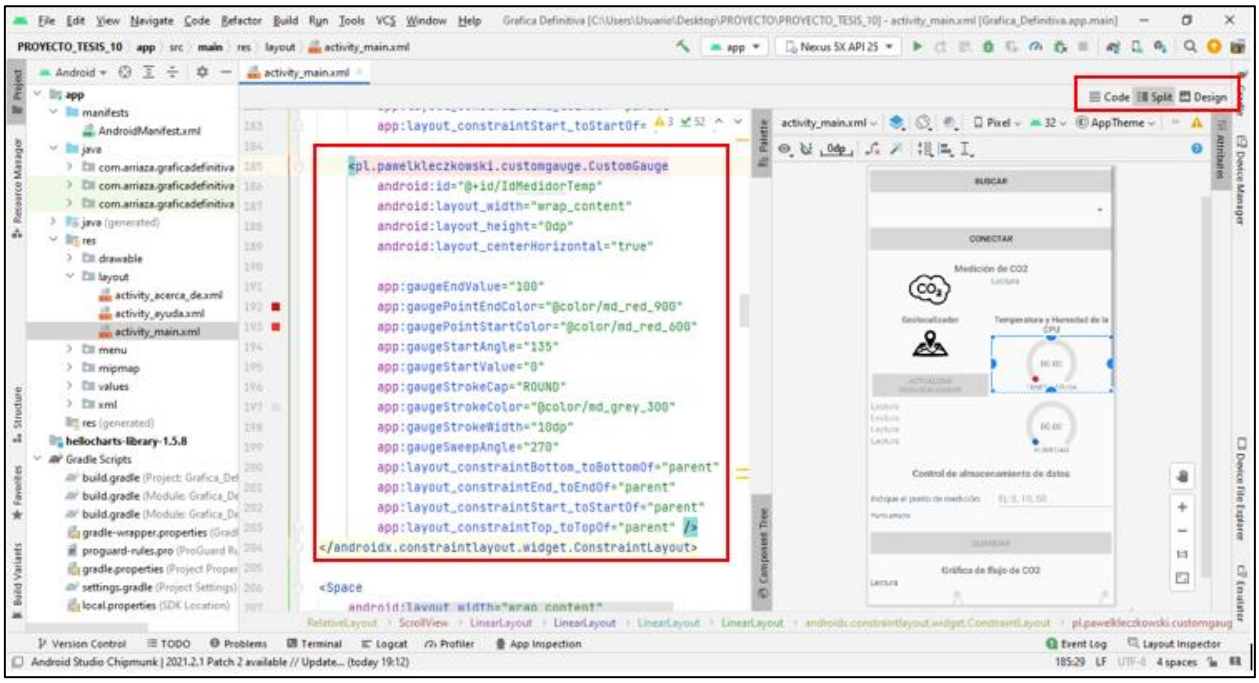
Código: Layout del código para aplicación móvil



Cuando escogemos la vista Split podemos ver código XML que se genera en automático al ir colocando componentes en el layout de la aplicación, justo en esa sección insertamos el siguiente código para poder usar las gráficas customgauge que representaran la temperatura y humedad de la CPU.

Ilustración 47

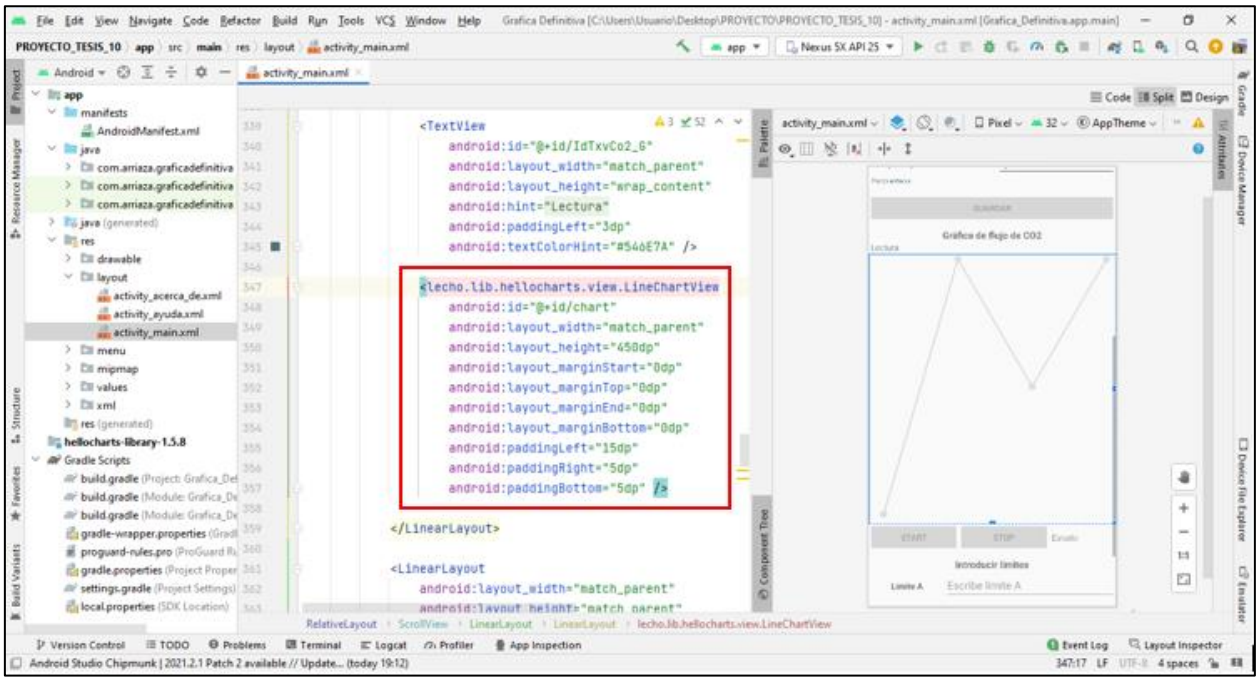
Código: Selección de Split para cambiar visualización



Lo mismo para la gráfica que se usó para mostrar los datos de concentración de CO2 en el tiempo. En este caso usamos la librería hellocharts.

Ilustración 48

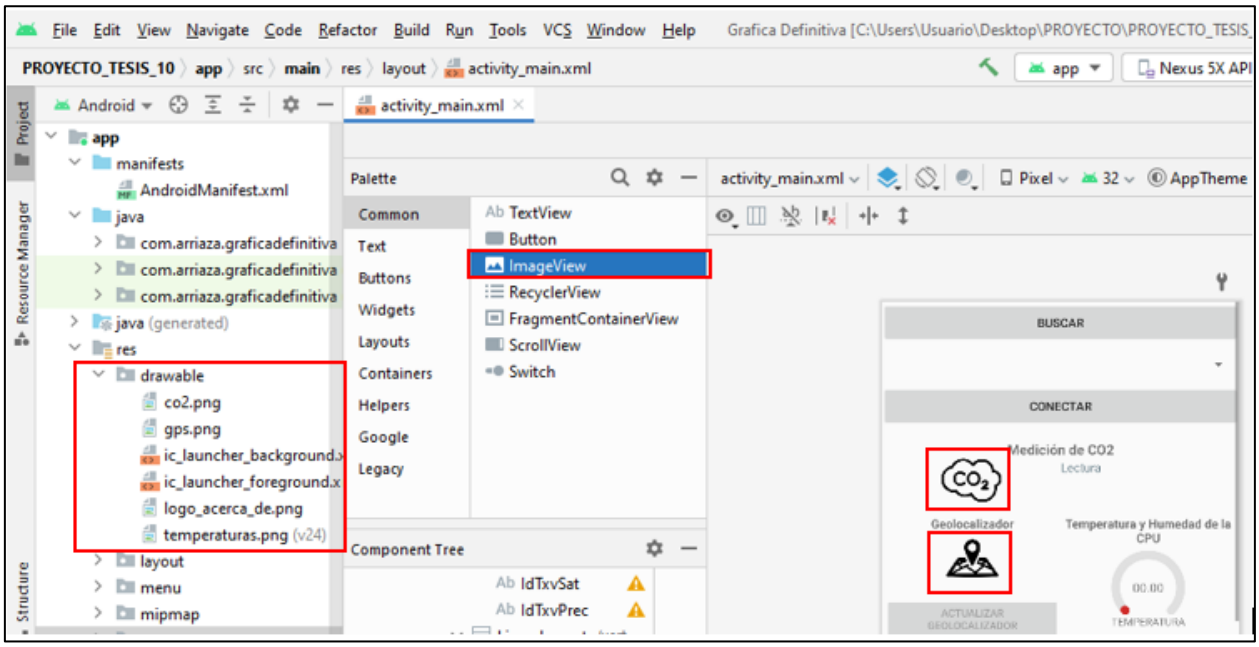
Código: Librería hellocharts



Para las imágenes es importante que sean de formato png y que les asignemos un nombre significativo todo en minúsculas, estas imágenes las insertamos en la carpeta drawable desde Android Studio. Y posteriormente para insértalas en el layout nos auxiliamos del componente ImageView el cual al insertarlo nos desplegara una ventana en la cual escogemos una de las imágenes que deseemos.

Ilustración 49

Insertar imágenes a través de un ImageView



En la carpeta de Java encontramos todo el código de la aplicación el cual nos permitirá vincular los objetos del layout con la parte de la lógica de programación, como lo es crear funciones que se ejecutaran al tocar cierto botón en el entorno de la aplicación por mencionar un ejemplo.

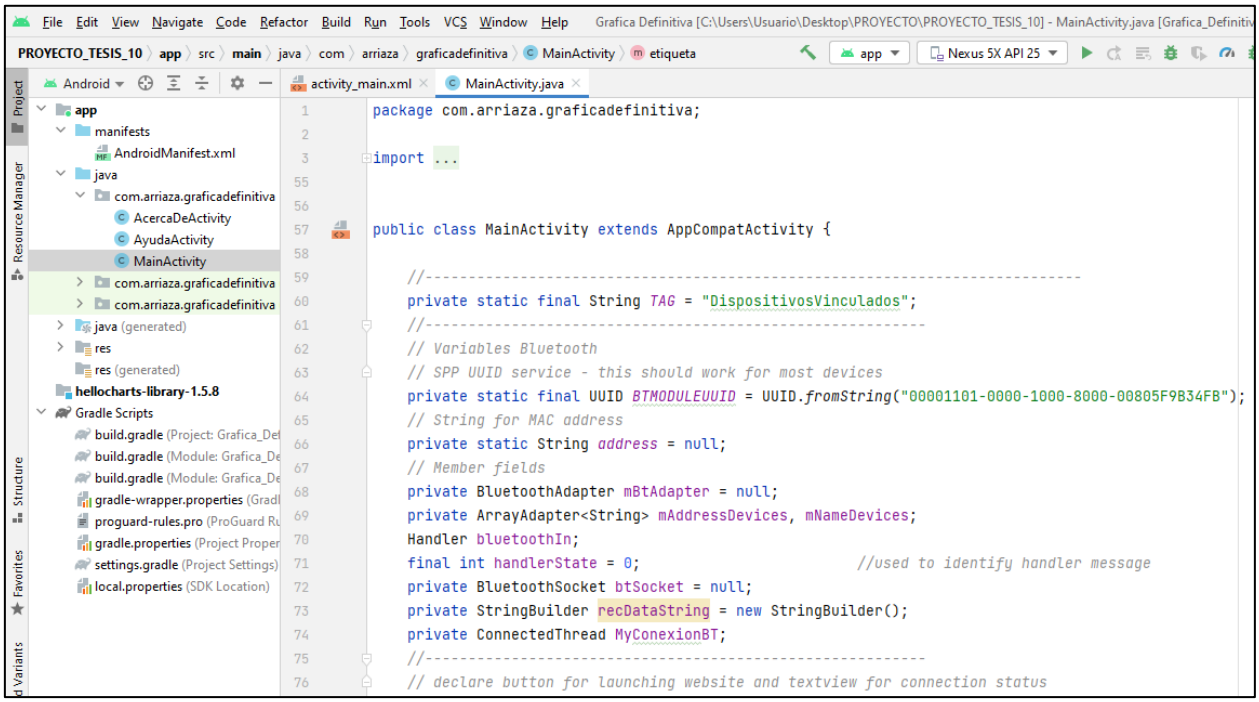
Lógica de programación de la app desarrollada.

Al momento de crear nuestro proyecto se crearon dos ventanas como mencionamos anteriormente, la que detallaremos en esta sección será la del MainActividad.java en esta sección es donde escribiremos el código en java de nuestra aplicación. Cabe mencionar que en la sección de anexos se encontrara todo el código de la aplicación, pero para mayor información acerca del proyecto de Android Studio se recomienda contactar con el docente asesor de la tesis o directamente con los autores.

Para la comunicación Bluetooth entre la aplicación Android y el microcontrolador se detallan a continuación las siguientes líneas de código necesarias para crear esta comunicación

Ilustración 50

MainActivity variables para la comunicación por Bluetooth



Algunas variables globales utilizadas en el proyecto se detallan a continuación y a la vez se identifican los tipos de componentes utilizados en el layout.

Ilustración 51

Variables globales del MainActivity

```
// declare button for launching website and textview for connection status
LineChartView lineChartView;
Button IdBtnBuscar, IdBtnConect, IdBtnDesconectar, IdBtnGps, Id_BtnPointMeasure, IdBtnCalcular;
Spinner DisEncontrados;
TextView IdTxvCo2, IdTxvMensaje, IdTxvCo2_G, IdTxvLat, IdTxvLon, IdTxvSat, IdTxvPrec, IdTxvHmd, IdTxvTmp;
CustomGauge IdMedidorTemp, IdMedidorHumd;

EditText idTxtPoint;
TextView etiqueta,puntoAnterior;

double DataCO2;
private Button buttonStartThread, IdBtnStop;

ArrayList<String> ListStrHora;
ArrayList<Float> ListValCO2;

String medidaFlujo;
float R2;
String StrLat, StrLon, StrSat, StrPre, StrTemp, StrHumd;

Date sessionTime = Calendar.getInstance().getTime();
String[] sessionTimes = sessionTime.toString().split( regex: " ");
String Point;
```

A continuación, se muestra la parte más clave e importante del código, específicamente en el evento *bluetoothIn*, que es en donde llega la trama de datos desde el microcontrolador.

En la línea *int endOfLineIndex = recDataString.indexOf("#")*; cuando llega un carácter con un numeral entonces sabemos perfectamente que ese es el último carácter y se procede a recuperar la trama de datos. Este carácter lo hemos añadido a propósito a las tramas de datos que el microcontrolador está enviando. Luego en la línea de código *String dataInPrint = recDataString.substring(0, endOfLineIndex)*; en el *STRING dataInPrint* tenemos todos los datos que posteriormente se tienen que procesar a manera de desglosarlos como se puede ver a partir del *try* en el cual en el vector *String[] parts* estamos desglosando los datos en partes, reconociendo los tipos de datos a partir de comas.

Ilustración 52

Primer evento BluetoothIn para la recepción de tramas de datos

```
//-------------------------------------------------
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            recDataString.append(readMessage);
            int endOfLineIndex = recDataString.indexOf("#");

            if (endOfLineIndex > 0) {
                String dataInPrint = recDataString.substring(0, endOfLineIndex);

                //467.53,0,0,0,0,0,0,0#
                //0,BombaOn,0,0,0,0,0,0#
                //0,0,LAT,LON,PREC,SAT,0,0,0#
                //0,0,0,0,0,0,temp,hmd#

                try {
                    String[] parts = dataInPrint.split( regex: ",");
                    String StrCO2 = parts[0];
                    String StrRespIn = parts[1];
                    StrLat = parts[2];
                    StrLon = parts[3];
                    StrSat = parts[4];
```

Posteriormente se procede a mostrar los datos en los TextView y en las gráficas customgauge de la actividad. Mas adelante encontraremos otro evento *bluetoothIn* en cual solo estamos recibiendo datos de CO2 para mostrarlos gráficamente.

Ilustración 53

Procedimiento para mostrar la data en TextView.

```
String[] parts = dataInPrint.split( regex: ",");
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);

} else {
    DataCO2 = Double.parseDouble(StrCO2);
    IdTxvCo2.setText(String.valueOf(DataCO2 + " ppm"));

}
```

La manera de enviarle ordenes al microcontrolador tales como actualizar la data de geolocalización, encender o apagar la bomba de gas del equipo, detener la trama de datos de temperatura y humedad se hace a través de la siguiente estructura: *MyConexionBT.write("P#")*; donde la letra P representa que el microcontrolador tiene que obtener la data de geolocalización

del módulo NEO 6M y enviarla a través del módulo Bluetooth HC-06. Para el encendido de la bomba será otra letra y así con las demás acciones que queremos que realice el microcontrolador.

Ilustración 54

Ej. de evento setOnClickListener y se envía un comando al microcontrolador

```
IdBtnGps.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        MyConexionBT.write( input: "P#");
        IdTxvLat.setText("");
        IdTxvLat.setText(StrLat);
        IdTxvLon.setText("");
        IdTxvLon.setText(StrLon);
        IdTxvSat.setText("");
        IdTxvSat.setText(StrSat);
        IdTxvPrec.setText("");
        IdTxvPrec.setText(StrPre);
    }
});
```

Almacenamiento de los datos en la memoria interna del celular

Junto con las variables globales que creamos al inicio del código declararemos lo que sería el nombre de los archivos CSV que generaremos con cada muestra de flujo de CO2 que se va tomando.

Ilustración 55

Construcción del nombre de los archivos CSV.

```
Date sessionTime = Calendar.getInstance().getTime();
String[] sessionTimes = sessionTime.toString().split( regex: " ");
String Point;
String filename = "Datos Punto " + Point + "-" + sessionTimes[0] + "-" + sessionTimes[1] + "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";
String filenameDos = "Reporte " + sessionTimes[0] + "-" + sessionTimes[1] + "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";

@Override
protected void onCreate(Bundle savedInstanceState) {
```

Luego dentro del segundo evento *bluetoothIn* al momento que el usuario presiona el botón *Start* en la aplicación se encenderá la bomba del equipo, se mandara una señal que hará que el microcontrolador ya no envié los datos de temperatura y humedad y que solamente se quede enviando los datos de CO2 cada segundo.

Ilustración 56

Segundo evento BluetoothIn

```
buttonStartThread.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        IdTxvMensaje.setText("");
        IdTxvMensaje.setText("Bomba ON");
        MyConexionBT.write( input: "G#");
        IdBtnStop.setEnabled(true);
        buttonStartThread.setEnabled(false);
        bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) {
                    String readMessage = (String) msg.obj;
                    recDataString.append(readMessage);
                    int endOfLineIndex = recDataString.indexOf("#");

                    if (endOfLineIndex > 0) {
                        String dataInPrint = recDataString.substring(0, endOfLineIndex);
                        //467.53,0,0,0,0,0,0,0#
                        //0,BombaOn,0,0,0,0,0,0#
                        //0,0,LAT,LON,PREC,SAT,0,0,0#
                        //0,0,0,0,0,0,temp,hmd#
```

Mas adelante en esta misma sección se procederá a armar el contenido de los datos de CO2 en el archivo, tomando también la fecha y la hora exacta en que ingreso cada valor de CO2.

Ilustración 57

Guardado de los datos de CO2.

```
} else {
    DataCO2 = Double.parseDouble(StrCO2);

    IdTxvCo2_G.setText(String.valueOf("CO2: " + DataCO2 + " ppm"));
    int DataIntCO2 = (int) DataCO2;

    Date measureTime = Calendar.getInstance().getTime();
    String[] currentTime = measureTime.toString().split( regex: " ");
    String measureLine = currentTime[0] + currentTime[1] + currentTime[2] + "," + currentTime[3] + "," + parts[0] + "\n";
    createFile(measureLine);

    String currentTime2 = new SimpleDateFormat( pattern: "HH:mm:ss", Locale.getDefault()).format(new Date());
    ListStrHora.add(currentTime2);
    ListValCO2.add((float) DataCO2);
```

Mas adelante en el método *createFile* creamos como su nombre lo indica el archivo de los datos de CO2 y también de los cálculos de Flujo de CO2 y su respectivo valor de coeficiente de determinación R^2.

Ilustración 58

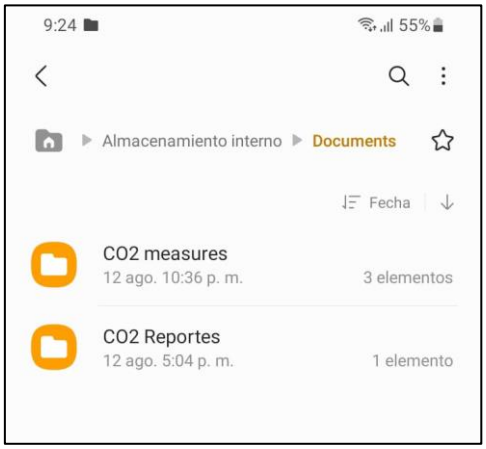
Método para crear el archivo CSV.

```
public void createFile(String measures) {
    if (isExternalStorageWritable()) {
        try {
            //String header = "System_Date_(Y-M-D),System_Time_(h:m:s),CO₂_(µmol_mol⁻¹),Ce
            String header = "System_Date_(Y-M-D),System_Time_(h:m:s),CO₂_(µmol_mol⁻¹)\n";
            File file = new File(getDocumentStorageDir(), filename);
            if (!file.exists()) {
                file.createNewFile();
                FileWriter fw = new FileWriter(file.getAbsoluteFile());
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write(header);
                bw.write(measures);
                bw.close();
            } else if (file.exists()) {
                FileWriter fw = new FileWriter(file.getAbsoluteFile(), append: true);
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write(measures);
                bw.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
            Context contexto = getApplicationContext();
            CharSequence erroraso = "Error escribiendo archivo";
            int duration = Toast.LENGTH_SHORT;
            Toast toast = Toast.makeText(contexto, erroraso, duration);
        }
```

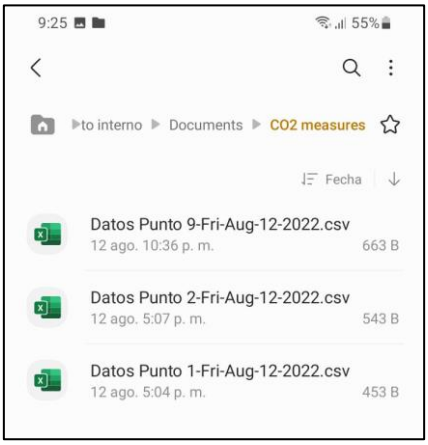
Si hacemos una corrida de la aplicación Android en el dispositivo y posteriormente nos dirigimos a la carpeta de Documentos en el mismo encontraremos dos carpetas una donde se almacenarán los reportes y otro donde se almacenarán por punto toda la data de CO2.

Ilustración 59

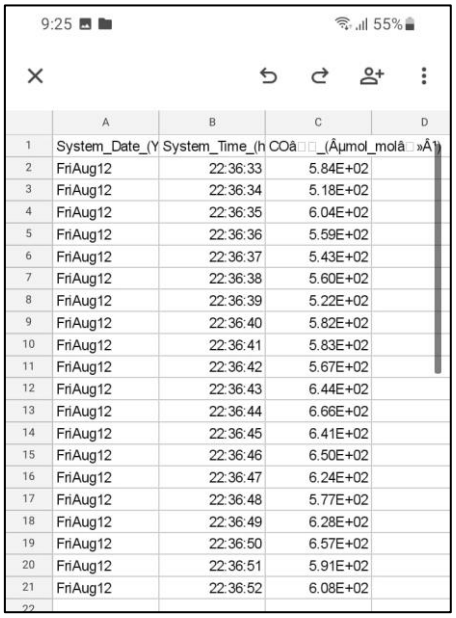
A) Muestra las carpetas, B) Muestra los CSV y C) Contenido de un CSV



A)



B)



	A	B	C	D
1	System_Date_(Y	System_Time_(h	COâ_(Âµmol_molâ»Â¹)	
2	FriAug12	22:36:33	5.84E+02	
3	FriAug12	22:36:34	5.18E+02	
4	FriAug12	22:36:35	6.04E+02	
5	FriAug12	22:36:36	5.59E+02	
6	FriAug12	22:36:37	5.43E+02	
7	FriAug12	22:36:38	5.60E+02	
8	FriAug12	22:36:39	5.22E+02	
9	FriAug12	22:36:40	5.82E+02	
10	FriAug12	22:36:41	5.83E+02	
11	FriAug12	22:36:42	5.67E+02	
12	FriAug12	22:36:43	6.44E+02	
13	FriAug12	22:36:44	6.66E+02	
14	FriAug12	22:36:45	6.41E+02	
15	FriAug12	22:36:46	6.50E+02	
16	FriAug12	22:36:47	6.24E+02	
17	FriAug12	22:36:48	5.77E+02	
18	FriAug12	22:36:49	6.28E+02	
19	FriAug12	22:36:50	6.57E+02	
20	FriAug12	22:36:51	5.91E+02	
21	FriAug12	22:36:52	6.08E+02	

C)

Método de regresión lineal y correlación

Como ya se mencionó anteriormente a los datos de concentración de CO2 que se van acumulando gracias a la cámara de acumulación valga la redundancia, y que se muestran en una gráfica en la aplicación Android se les aplica regresión lineal esto es debido a que los vulcanólogos y geofísico necesitan obtener el valor de la pendiente la gráfica de CO2 que sería el flujo en ppm/s también es necesario obtener el valor del coeficiente de determinación R^2 el cual tiende a estimar de forma optimista el ajuste de la regresión lineal (IBM, s.f.).

Hemos creado el método *calcularRegresion* en el cual mandamos a llamar dos funciones a las cuales les pasamos como parámetros dos ArrayList que contiene la data de los puntos y de CO2 uno en x y otro en y respectivamente.

Ilustración 60

Método para calcular la regresión lineal

```
//CALCULO DE LA REGRESION LINEAL Y FLUJO
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
public void calcularRegresion(View view) {
    Integer[] limites = getLims(); //Obtenemos los limites del metodo "getLims"
    if (limites[0] == limites[1]) {
        return;
    } else {
        //Aqui pasamos la data de los ArrayList "trimXData" y "trimYData" a los ArrayList "nx" y "ny"
        ArrayList<Integer> nx = new ArrayList<>(trimXData(limites[0], limites[1]));
        ArrayList<Float> ny = new ArrayList<>(trimYData(limites[0], limites[1]));

        Float pendiente = new Float(calcPendiente(nx, ny));//Nos ayudams del método "calcPendiente" y le pasamos los par
        final TextView pendientlayer = (TextView) findViewById(R.id.pendiente);
        pendientlayer.setText("Flujo " + pendiente.toString() + " ppm/s");

        Float correlacion = new Float(calcCorrelacion(nx, ny));//Nos ayudams del método "calcCorrelacion" y le pasamos l
        float correlacionR2 = (float) Math.pow(correlacion,2);
        float roundCorrelacionR2 = (float) (Math.round(correlacionR2 * 1000.0) / 1000.0);
        final TextView correlacionlayer = (TextView) findViewById(R.id.correlacion);
        correlacionlayer.setText("R^2 = " + roundCorrelacionR2);
```

La primera función que mandamos a llamar es la de la regresión lineal y la cual nos va a retornar el valor de la pendiente de la gráfica. La segunda función que llamamos es la de la del

coeficiente de correlación R el cual posteriormente elevamos al cuadrado para de esta manera obtener el coeficiente de determinación.

Para comenzar explicaremos como obtener lo datos de CO2 los cuales están almacenados en la memoria interna del teléfono para ellos nos auxiliamos del método *getFileMeasure()* el cual obtendrá los datos de CO2 previamente almacenados y los guardara en un ArrayList llamado *yValues* el cual será lo que nos va a retornar al invocar el método posteriormente en otras secciones del código.

Ilustración 61

Método para obtener el archivo CSV

```
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[2]);
                    yValues.add(yPoints);
                }
            }
            br.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return yValues;
}
```

Luego en el método *trimYData* almacenaremos los datos de CO2 y los retornaremos en un ArrayList *trimedY* para ello del método "*GetFileMeasure*" obtenemos la data y la guardamos en el ArrayList "*yValues*". Rellenamos el ArrayList "*trimedY*" con los datos del ArrayList "*yValues*" desde el límite A al límite B.

Ilustración 62

Método para rellenar un ArrayList con la data del archivo CSV

```
public ArrayList<Float> trimYData(Integer lima, Integer limb) {
    ArrayList<Float> trimedY = new ArrayList<>();
    ArrayList<Float> yValues = new ArrayList<>(getFileMeasure());//
    //Rellenamos el ArrayList "trimedY" con los datos del ArrayList
    for (int i = lima; i <= limb; i++) {
        trimedY.add(yValues.get(i));
    }
    return trimedY;
}
```

Luego los datos en x los vamos contabilizando y almacenando en un ArrayList llamado *trimedX*.

Ilustración 63

Método para rellenar un ArrayList para los valores en X

```
public ArrayList<Integer> trimXData(Integer lima, Integer limb) {
    ArrayList<Integer> trimedX = new ArrayList<>();
    //Aqui rellenamos el ArrayList "trimedX" con un contador ayudandonos de un for
    for (int i = lima; i <= limb; i++) {
        trimedX.add(i);
    }
    return trimedX;
}
```

Creamos el método *getLims*() el cual será el encargado de capturar los valores numéricos que el usuario ingresará en los EditText correspondientes y los retornará en forma de vector.

Ilustración 64

Método para obtener los límites de la gráfica

```
private Integer[] getLims() {
    final EditText lima = (EditText) findViewById(R.id.limita);
    final EditText limb = (EditText) findViewById(R.id.limitb);
    Integer limites[] = {0, 0}; //Creamos vector limites
    try {
        limites[0] = Integer.parseInt(lima.getText().toString()); //Guardamos en posicion 0 el valor de Limite A
        limites[1] = Integer.parseInt(limb.getText().toString());
        return limites;
    } catch (NumberFormatException e) {
        return limites;
    }
}
```

Regresando al método *calcularRegresion* nos damos cuenta que la primera línea de código manda a llamar a la función getLims() para darnos los valores que el usuario a ingresado. Posteriormente hacemos las validaciones correspondientes y entramos a un *else* en cual se crearán 2 ArrayList nuevos a los cuales les pasaremos como parámetros las dos ArrayList creados previamente y estarán delimitados por los límites que el usuario a digitado.

Ilustración 65

Creación de los ArrayList

```
public void calcularRegresion(View view) {
    Integer[] limites = getLims(); //Obtenemos los limites del metodo "getLims"
    if (limites[0] == limites[1]) {
        return;
    } else {
        //Aqui pasamos la data de los ArrayList "trimXData" y "trimYData" a los ArrayList
        ArrayList<Integer> nx = new ArrayList<>(trimXData(limites[0], limites[1]));
        ArrayList<Float> ny = new ArrayList<>(trimYData(limites[0], limites[1]));

        Float pendiente = new Float(calcPendiente(nx, ny));//Nos ayudams del método "calcP
        final TextView pendientlayer = (TextView) findViewById(R.id.pendiente);
        pendientlayer.setText("Flujo " + pendiente.toString() + " ppm/s");

        Float correlacion = new Float(calcCorrelacion(nx, ny));//Nos ayudams del método "c
        float correlacionR2 = (float) Math.pow(correlacion,2);
        float roundCorrelacionR2 = (float) (Math.round(correlacionR2 * 1000.0) / 1000.0);
        final TextView correlacionlayer = (TextView) findViewById(R.id.correlacion);
        correlacionlayer.setText("R^2 = " + roundCorrelacionR2);
```

Luego podemos ver del método *calcularRegresion* que mandamos a llamar a la función de *calcPendiente* y a la función de *calcCorrelacion* respectivamente y les pasamos los *ArrayList* de la función *calcularRegresion.*

El método de la pendiente se puede observar en la siguiente imagen:

Ilustración 66

Método para calcular la pendiente

```
public float calcPendiente(ArrayList<Integer> x, ArrayList<Float> y) {
    float corrector = 1f;
    float pendiente = 0;
    float term1 = 0;
    float tem1 = 0;
    float term2 = 0;
    float sumx = 0;
    float sumy = 0;
    float term3 = 0;
    float tem3 = 0;
    float term4 = 0;

    //System.out.println(x);
    for (int i = 0; i < x.size(); i++) {
        tem1 = x.get(i) * corrector * y.get(i) + tem1;
        sumx = x.get(i) * corrector + sumx;
        sumy = y.get(i) + sumy;
        tem3 = (float) Math.pow(x.get(i) * corrector, 2) + tem3;
    }
    term1 = x.size() * tem1;
    term2 = sumx * sumy;
    term3 = x.size() * tem3;
    term4 = (float) Math.pow(sumx, 2);
    pendiente = (term1 - term2) / (term3 - term4);

    float roundPendiente = (float) (Math.round(pendiente * 1000.0) / 1000.0);
    return roundPendiente;
```

El de la correlación es el siguiente, basándonos en las fórmulas matemáticas para el cálculo de la correlación, por ejemplo, ver (Superprof, s.f.).

Ilustración 67

Método para calcular la correlación

```
public float calcCorrelacion(ArrayList<Integer> x, ArrayList<Float> y) {
    float contadorX = 0;
    float contadorY = 0;
    float contadorX2 = 0;
    float contadorY2 = 0;
    float contadorXY = 0;
    int almacenX, almacenX2;
    float almacenY, almacenY2;
    float xy;
    //Aqui viene la correlacion
    for (int i = 0; i < x.size(); i++) {
        almacenX = x.get(i);
        contadorX = contadorX + almacenX;
        almacenX2 = (int) Math.pow(almacenX, 2);
        contadorX2 = contadorX2 + almacenX2;
    }
    float xMed = (contadorX / x.size());
    float xMed2 = (float) ((contadorX2 / x.size())-Math.pow(xMed, 2));
    float raizX2 = (float) Math.sqrt(xMed2);
    System.out.println("xMed = " + xMed);System.out.println("xMed2 = " + xMed2);

    for (int i = 0; i < x.size(); i++) {
        almacenY = y.get(i);
        contadorY = contadorY + almacenY;
        almacenY2 = (float) Math.pow(almacenY, 2);
        contadorY2 = contadorY2 + almacenY2;
    }
```

Primero se crean algunas variables locales para posteriormente en un *for* crear los valores en X hasta el tamaño del ArrayList. Se elevan al cuadrado con la función *Math.pow* y se almacenan. Luego fuera del *for* se aplican las fórmulas y lo mismo para los valores en Y. Los valores se

imprimen en consola para posterior corroboración. Por último, multiplicamos los valores en X por los valores en Y con la ayuda de un *for*, se almacenan en una variable llamada *contadorXY* y fuera del *for* se aplican las fórmulas necesarias y se imprimen en consola.

Ilustración 68

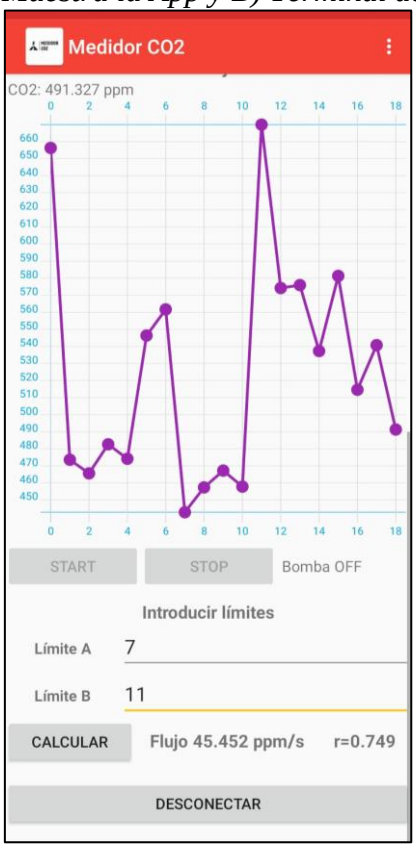
Continuación del método de la correlación

```
    for (int i = 0; i < x.size(); i++) {
        xy = x.get(i) * y.get(i);
        contadorXY = contadorXY + xy;
    }
    float xyMed_F = (contadorXY / x.size()) - xMed * yMed;
    float correlacion = xyMed_F / (raizX2 * raizY2);
    System.out.println("xyMed_F = " + xyMed_F);System.out.println("r = " + correlacion);

    return correlacion;
}
```

A manera de comprobación mostramos una corrida del código:

Ilustración 69

A) Muestra la App y B) Terminal de Android Studio




```
D/skia: anh.lt2 GrGpu::createTexture2
I/System.out: xMed = 9.0
I/System.out: xMed2 = 2.0
I/System.out: raizX2 = 1.4142135
I/System.out: yMed = 499.0712
I/System.out: yMed2 = 7357.268
I/System.out: raizY2 = 85.77452
I/System.out: xyMed_F = 90.90332
I/System.out: r = 0.7493875
I/System.out: TotalXY = 22912.719
D/skia: anh.lt2 GrGpu::createTexture2
D/SmartClipDataCropperImpl: doExtractSmartCl
```

A) B)

Y en Excel haciendo el proceso de la correlación:

Ilustración 70

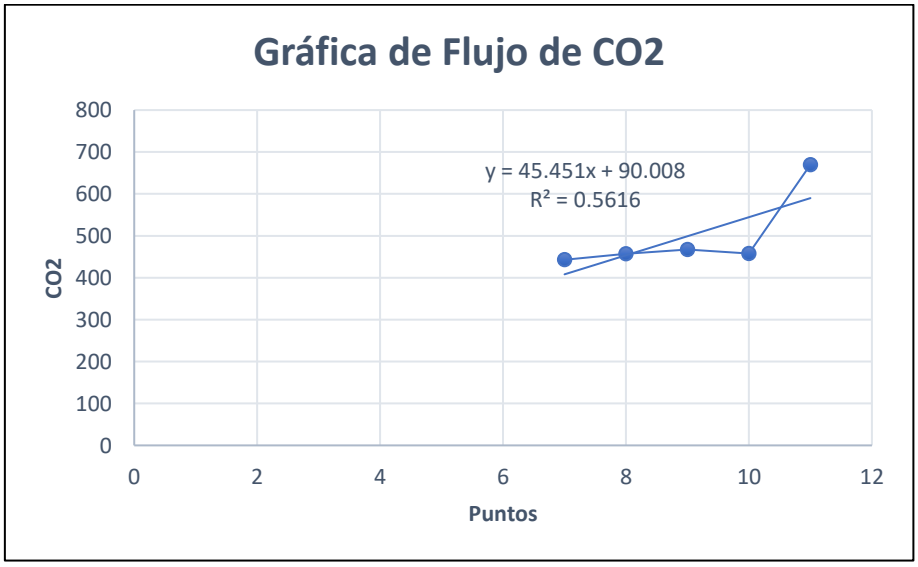
Resultados en Excel de la prueba de la aplicación

	X	Y	X2	Y2	XY
1	7	442.881	49	196143.58	3100.167
2	8	457.424	64	209236.716	3659.392
3	9	467.26	81	218331.908	4205.34
4	10	457.881	100	209655.01	4578.81
5	11	669.91	121	448779.408	7369.01
TOTAL	45	2495.356	415	1282146.62	22912.719
	xMed	yMed	xMed2	yMed2	
	9	499.0712	2	7357.26169	
Raiz			1.41421356	85.7744816	
xtMed_F					90.903
R					0.74938521
R^2					0.56157819

Vemos que R coincide con los valores obtenido por la aplicación Android. Luego graficando estos datos en Excel y aplicando regresión lineal a la gráfica y mostrando el valor del coeficiente de determinación R^2 obtenemos de igual manera los mismos resultados:

Ilustración 71

Gráfica de la prueba del funcionamiento de la aplicación



LÓGICA DE PROGRAMACIÓN DEL MICROCONTROLADOR

Para empezar, es importante que según como vimos el conexionado de los componentes a la placa del microcontrolador, al momento de subir un programa a esta placa debemos desconectar el módulo bluetooth de los pines RX y TX, una vez subido el programa volvemos a conectar el módulo bluetooth a los respectivos pines de la placa.

El IDE que se utilizó para cargar el programa al microcontrolador fue el IDE de Arduino, pero vale la pena mencionar que hay otros IDEs con características similares que pueden cargar programas al microcontrolador, como lo es el IDE Visual Studio Code con el plug-in de PlatformIO.

Para la recepción de datos utilizamos el evento software serial ya que estamos trabajando tanto a nivel del microcontrolador como a nivel de Android Studio para el envío y recepción de tramas de datos.

Ilustración 72

Evento serial

```
void serialEvent() {
  while (Serial.available())
  {
    VarChar = (char)Serial.read();
    StrDatIn += VarChar;
    if (VarChar == '#') {
      StringCompleta = true;
    }
  }
}
```

Cada vez que llega un carácter dentro del evento serial lo que hacemos es leer el carácter y lo cargamos en la variable *VarChar* luego esa variable se pone dentro del *String trDatIn* y cuando

llega el carácter de numeral entonces sabemos que ha llegado el ultimo carácter y posteriormente hacemos que la variable tipo *boolean StringCompleta* sea igual a true. Posteriormente cuando esa variable sea igual a true se habilita la siguiente condición:

Ilustración 73
Parte del Void loop

```
void loop() {
  if (StringCompleta)
  {
    delay(5);

    CharPrimer = StrDatIn.charAt(0);

    StrDatIn.replace(" ", "");
    StrDatIn.replace("F", "");
    StrDatIn.replace("G", "");
    StrDatIn.replace("H", "");
    StrDatIn.replace("#", "");

    if (CharPrimer == 'G') {
      delay(500);
      if (bandera == false) {
        bandera = true;
        digitalWrite(rele, LOW);
        StrDataResp = "0,Bomba ON,0,0,0,0,0,0#\n";
        //co2,rele,lat,lon,sat,pre,temp,humd#}
      } else {
        digitalWrite(rele, LOW);
        StrDataResp = "0,Bomba ON,0,0,0,0,0,0#\n";
        //co2,rele,lat,lon,sat,pre,temp,humd#}
```

Ingresamos al *void loop()* y esperamos un tiempo de 5ms lo cual es un tiempo de prevención porque ya que se están recibiendo datos de CO2 cada segundo. Posteriormente en *CharPrimer* lo que hacemos es recuperar el carácter y procedemos a borrar los primeros caracteres y luego dependiendo que tipo de carácter es procedemos a encender o apagar la bomba, dejar de leer la temperatura y humedad o actualizar los datos del geo localizador y enviamos también una

respuesta como Bomba On. Luego al final tenemos las siguientes líneas de código que lo que hacen es asegurar que todos los datos que se están enviando realmente se han enviado y esperamos 1 segundo y se procede a inicializar nuevamente las variables y estamos a la espera de que ingrese una nueva trama de datos y cuando *StringCompleta* vuelva a ser true procede nuevamente en el *void loop*().

Ilustración 74

Inicialización de las variables

```
    Serial.print(StrDataResp);
    while (Serial.available() > 0 ) {
      Serial.read();
    } Serial.flush();

    delay(1000);

    StrDataResp = "";
    VarChar = ' ';
    StringCompleta = false;
    StrDatIn = "";
```

Como se mencionó anteriormente el microcontrolador está leyendo el valor de temperatura y humedad cada 10 segundos y los está enviando a la aplicación Android esa parte del código se muestra a continuación:

Ilustración 75

Llamado de la función leer temperatura y humedad

```
if (bandera == false) {
  if (millis() > TiempoAhora + periodo) {
    TiempoAhora = millis();
    for (int i = 0; i <= 1; i++) {
      LeerHumd_Temp();
      delay(1000);
    }
  }
}
```

Donde hemos credo una bandera tipo *boolean* que nos permite poder dejar de leer la data del sensor de temperatura y humedad al momento de iniciar una toma de muestras en campo. La función *millis* lo que nos permite es poder entrar al *if* cada 10 segundos ya que periodo es igual a 10000 y es una variable que inicializamos al principio del código. Luego la función *LeerHumd_Temp*() fue una función que creamos para poder leer la data del sensor DHT-22, dicha función se muestra a continuación:

Ilustración 76

Función para leer la temperatura y humedad del DHT-22

```
void LeerHumd_Temp (void)
{
  hmd = Obj_DHT.readHumidity(); //Lee la humedad
  tmp = Obj_DHT.readTemperature();//Lee la temperatura en grados centigrados (Valor por defecto)

  // verifica si alguna lectura ha fallado
  if (isnan(hmd) || isnan(tmp)) {
    Serial.println("Existe un error en la lectura del sensor DHT22!");
    hmd = 0; tmp = 0;
  }
  Serial.print("0,0,0,0,0,0,"); Serial.print(tmp, 2); Serial.print(","); Serial.print(hmd, 2); Serial.print("#\n");
  while (Serial.available() > 0 ) {
    Serial.read();
  } Serial.flush();
}
```

Para leer la data del sensor de CO2 el LICOR LI-830 estamos usando la función software serial, para ello hemos inicializada los pines que emularemos en la placa del microcontrolador usando *SoftwareSerial.*

Ilustración 77

Librerías e inicialización de software serial

```
#include <SoftwareSerial.h>
#define DEBUG(a) Serial.println(a);
#include <TinyGPS.h>
#include "DHT.h"

SoftwareSerial ss(4, 3);//4Rx, 3Tx
SoftwareSerial licor(10, 11); //10 RX, 11 TX.
```

Y la función que hemos creado para el sensor de CO2 es la siguiente:

Ilustración 78

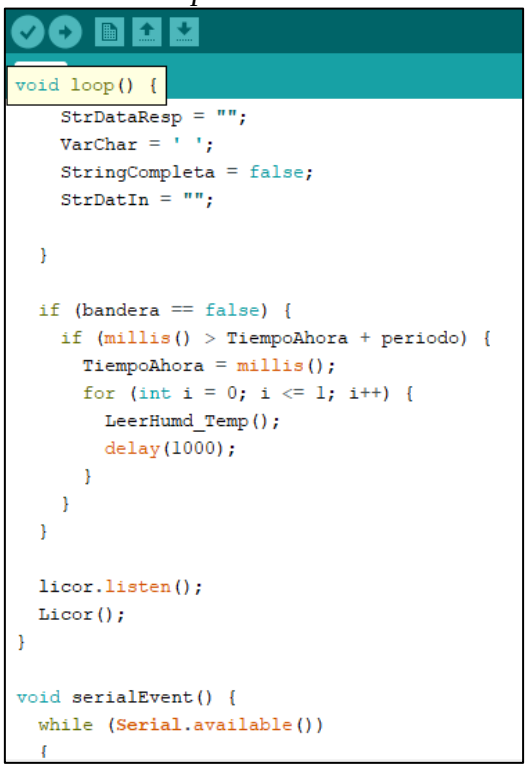
Función para leer la data del LI-830

```
void Licor(void) {
  if (licor.available())
  {
    String data  = licor.readStringUntil('\n') + ",0,0,0,0,0,0,0#";
    DEBUG(data);
  }
}
```

La función LICOR se está ejecutando cada 1 segundo debido a que hemos configurado el LI-830 para que solo envié CO2 cada 1 segundo. Y la manera de mandar a llamar esta función dentro del *void loop()* es dejándola primeramente a fuera de la condicional de *StringCompleta* ya que no estamos esperando ningún carácter para que lea CO2 si no que queremos que siempre se lea CO2 y se envié de manera instantánea a la aplicación Android.

Ilustración 79

Final del void loop



```
void loop() {
    StrDataResp = "";
    VarChar = ' ';
    StringCompleta = false;
    StrDatIn = "";

  }

  if (bandera == false) {
    if (millis() > TiempoAhora + periodo) {
      TiempoAhora = millis();
      for (int i = 0; i <= 1; i++) {
        LeerHumd_Temp();
        delay(1000);
      }
    }
  }

  licor.listen();
  Licor();
}

void serialEvent() {
  while (Serial.available())
  {
```

CAPÍTULO IV

IMPLEMENTACIÓN Y ANÁLISIS DE RESULTADOS

Después de todo el diseño, los planteamientos, construcciones y pruebas, el siguiente paso dentro de este proyecto es poner en marcha su funcionamiento. Para esto es necesario hacer salidas de campo, de tal forma que se establezca un arranque de todas las mediciones por hacerse con los equipos de medición de CO2.

La ventaja de nuestra región, es la variedad de estructuras volcánicas que se tiene a disposición para realizar mediciones de CO2. En El Salvador se cuentan con 23 volcanes individuales y 5 campos volcánicos. El lugar definido por el proyecto es el majestuoso volcán de Santa Ana.

4.1 PROCEDIMIENTO DE MEDICIÓN

a) Ruta de perfil de medición.

Para recolectar datos de monitoreo de CO2 con el equipo construido, se establece la creación de un perfil de medición de CO2. Este perfil cuenta con la siguiente ruta de medición en el rin del volcán de Santa Ana:

Ilustración 80

Ubicación de ruta establecida para perfil de medición.

Dicha ruta, son unos puntos marcados anteriormente por las personas encargadas del monitoreo volcánico de la Universidad de El Salvador.

b) Procedimiento de medición de CO2.

Para el cálculo del flujo de CO2 que emite el suelo del volcán, se debe de tomar varios datos de concentración de CO2 dentro de una cámara de acumulación, en donde se pueda observar la tendencia que se da por el flujo que expulsa el suelo. Se obtiene cierta cantidad de datos de tal forma que permita seleccionarse los puntos para calcular su tendencia por medio de una regresión. Esta regresión calculada, nos da el valor del flujo medido en ese punto.

. La metodología para la toma de datos consistió en tomar muestras de flujo de emisión de CO2 en determinados puntos marcados anteriormente por los vulcanólogos de la Universidad de

El Salvador, dichos puntos están distanciados a 1m, pero se decidió tomar muestras cada 5 puntos, es decir, cada 5m con el fin de obtener 20 muestras y observar como el flujo aumentaba a medida que nos acercábamos a una de las zonas anómalas en el área de estudio.

Ilustración 81

Captura de concentración del aire del ambiente.

Se inicia con la toma de muestras de concentración de CO2 en el aire del ambiente, en dirección externa del cráter, para poder testear la respuesta del equipo y tener un punto de partida, esto se hizo como primer paso en cada punto de estudio en cuestión. Es paso nos permite examinar el aire del ambiente y observar que el equipo se encuentra calibrado y que en el aire no hay una concentración de CO2 fuera de lo normal.

Se obtiene dentro de la aplicación Android, los datos del GPS para ubicar las coordenadas donde se realiza la medición del flujo de CO2. También se observan los datos de rendimiento del equipo, su temperatura y humedad del CPU para tener un control del estado del equipo.

Ilustración 82

Recolección y observaciones previas del proceso de medición de flujo de CO2.

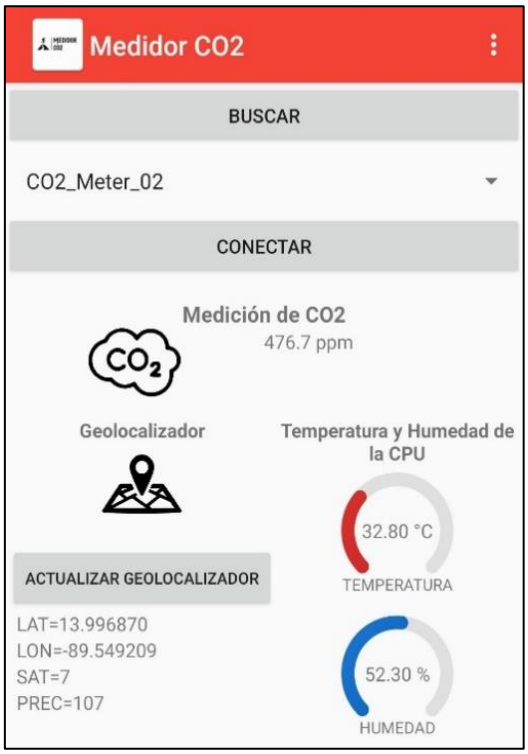


Ilustración 83

Fotografía del proceso de medición de flujo de CO2

El siguiente paso, es proceder a tomar muestras de concentración de CO2 del suelo del volcán.

Ilustración 84

Medición de flujo de emisión de CO2 en el suelo en uno de los puntos del rin del volcán de Santa Ana.

Esto se realiza introduciendo la cámara de acumulación dentro de la superficie del suelo volcánico. Se posiciona la cámara y se deja enterrada aproximadamente por un minuto y medio, lo que nos permite obtener suficientes datos de concentración de CO2, entre 80 y 100 muestras de concentración, para visualizar su tendencia y así obtener la pendiente de la gráfica que nos determinar el flujo de emisión de CO2 en dichos puntos.

Ilustración 85

Fotografía de cámara de acumulación enterrada en el momento de realizar la medición de flujo de CO2

Este proceso se repite todas las veces en cada uno de los puntos definidos para medir y establecidos por la ruta para crear el perfil de medición de CO2.

c) Salidas de campo.

Las fechas en las que se realizaron las mediciones del flujo de CO2 del volcán de Santa Ana fueron los días 29/marzo/2022, 6/abril/2022 y el 13/julio/2022.

Ilustración 86

Fotografía de salida de campo

Las salidas de campo fueron acompañadas por el asesor de este proyecto, el Ingeniero Carlos Pocasangre; también por el vulcanólogo asesor también de este proyecto, el Licenciado Benancio Mirando; la compañía de compañeros de la carrera, Diego Hernández y Lizardo Arias; y un grupo de estudiantes de licenciatura en Geofísica de la Facultad Multidisciplinaria de Occidente.

Ilustración 87

Fotografía de salida de campo, incluye a personas de apoyo.

Estos últimos, con el fin de comparar y sustentar los datos de medición del flujo de CO2, realizaron mediciones de SP (potencial espontáneo, o en inglés "*self-potencial*") y mediciones de temperatura del suelo. Esto reforzó aún más la creación del perfil de medición.

4.2 RESULTADOS

Luego de los días cansados de subir el volcán de Santa Ana con el equipo de medición. Se procede al trabajo de oficina. Consiste en ordenar, interpretar y tabular los datos que se han recopilado de los días de salidas de campo.

Tablas resumen de resultados obtenidos

Se construyeron 2 tablas resumen de las mediciones de flujo de CO2, una del día 29 de marzo y la otra del día 13 de julio. Estas tablas nos muestran lo siguiente:

a) Tabla del 29 de marzo

Tabla 12

Resumen de datos obtenidos en el volcán de Santa Ana el día martes 29/marzo/22

Punto	Latitud	Longitud	CO2-Flujo (ppm/s)
0	13.846617	-89.627437	0.3
1	13.846599	-89.627416	0.3
5	13.846631	-89.627395	0.7
10	13.846669	-89.627349	1.6
15	13.846695	-89.62732	3.9
20	13.84673	-89.627283	4.3
25	13.846767	-89.627241	7.3
30	13.846798	-89.62723	9.1
35	13.846819	-89.627202	8.2
40	13.846863	-89.627145	5.1
45	13.846886	-89.62712	57.2
50	13.846918	-89.627087	10.1
55	13.846944	-89.627064	9.8
60	13.846992	-89.627024	6.9
65	13.847021	-89.626979	4.1
70	13.847033	-89.62693	2.3
75	13.847042	-89.626901	1.3

80	13.847037	-89.626852	0.7
85	13.847063	-89.626796	1.0
90	13.847061	-89.626746	0.9
95	13.847064	-89.626711	1.3
100	13.847075	-89.626663	0.4

Las primeras muestras de flujo de emisión de CO2 rondaban entre 0.20 y 1.70 ppm/s a 15m del punto inicial, como se observa en la figura 18, pero en el punto 15 podíamos ver un considerable aumento en el flujo y el cual a medida que íbamos avanzando en distancia el flujo iba aumentando, ya que nos acercábamos a la zona anómala.

Se observa dentro de la tabla 5 de resumen que en el punto 45 se ubica la zona anómala debido al alto valor de flujo de emisión de CO2 obtenido de aproximadamente 57.2 ppm/s. También se observa la tendencia de incremento en el flujo a medida que nos acercábamos a la zona anómala y el decremento de este a medida continuamos tomando datos después de esta zona, dando una tendencia en forma de campana.

b) Tabla del 13 de julio.

Tabla 13

Resumen de datos obtenidos en el volcán de Santa Ana el día 13/julio/2022

Punto	Latitud	Longitud	CO2-Flujo(ppm/s)
0	13.846584	-89.2627426	0.3
5	13.84663	-89.627403	0.1
10	13.8466	-89.6274	1.0
15	13.846685	-89.627326	2.2
20	13.846626	-89.627281	1.2
25	13.84677	-89.627258	29.0
30	13.846794	-89.627235	33.0
35	13.846837	-89.627182	13.5
40	13.846865	-89.627151	2.4
45	13.846869	-89.62712	65.9
50	13.846909	-89.62709	35.4
55	13.846942	-89.627052	7.6
60	13.84697	-89.627029	7.4
65	13.846996	-89.626983	15.5
70	13.847007	-89.626945	2.8
75	13.847015	-89.626899	6.6
80	13.847036	-89.626861	6.0
85	13.847043	-89.626815	3.9
90	13.847093	-89.62677	2.2
95	13.847057	-89.626716	1.5
100	13.847066	-89.626686	1.3
105	13.847068	-89.62664	0.3

Los resultados mostrados por la tabla 6, nos permite observar y comparar que al igual que en el día 29 de marzo, la zona anómala se encuentra en el punto 45. Este día de medición se logró obtener el valor de 65.9 ppm/s.

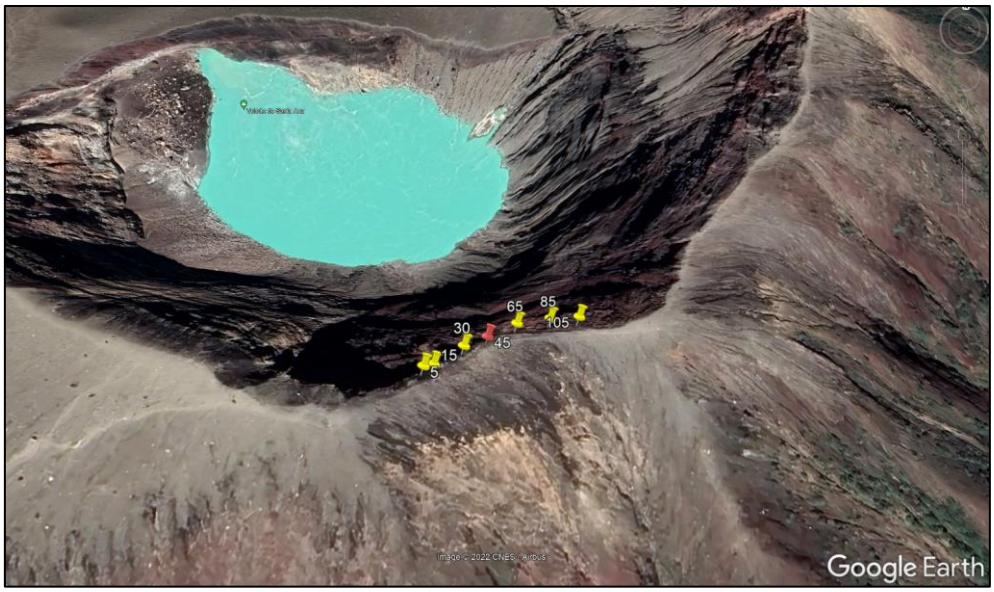
Cabe destacar lo variado de los datos de medición del flujo de CO2 a los alrededores de la zona anómala. Este detalle nos permite resaltar las condiciones meteorológicas del día 13 de julio, destaca los fuertes vientos que se percibieron. Esto pudo haber ocasionado que la expulsión de CO2 haya sido con tendencias repentinas que ocasionaron un incremento del flujo de CO2 por parte del suelo del volcán, esta experiencia nos proporciona la oportunidad de ver como las condiciones meteorológicas intervienen en el comportamiento del volcán, al menos en su superficie.

Otros resultados

Los datos de posicionamiento obtenidos por el módulo GPS integrado al equipo de medición pueden visualizarse:

Ilustración 88

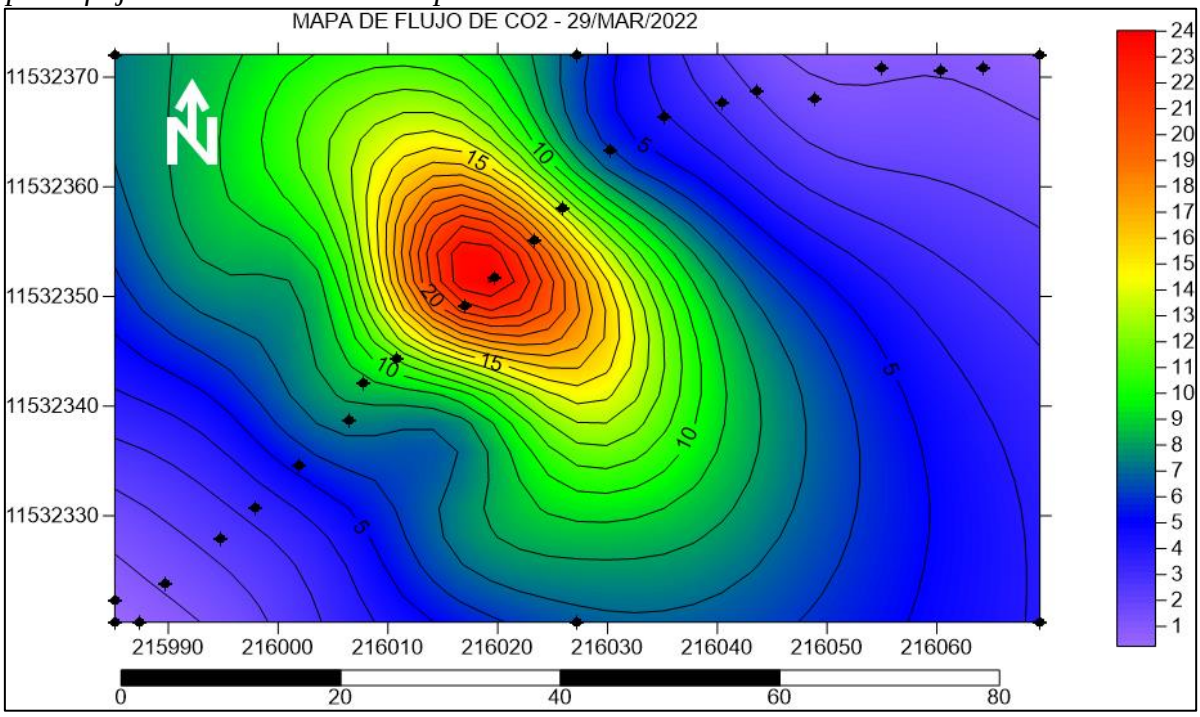
Ubicación de datos del GPS



Se observa la fiabilidad del equipo también en la captura del posicionamiento de los puntos analizados. Cada punto obtenido se encuentra dentro de la ruta establecida para la creación del perfil de medición de flujo de CO2 en el volcán de Santa Ana.

Ilustración 89

Mapa de flujo de CO2. Unidades de posicionamiento en UTM.



Otro resultado extra que se puede se puede representar los datos de medición, es por medio de un mapa de calor. Solamente hay que recalcar, que para la construcción de un mapa de calor es necesario tomar datos con los puntos ubicados en forma de mallado (grid) o una superficie cerrada. En nuestro caso obtuvimos como lo muestra la imagen anterior.

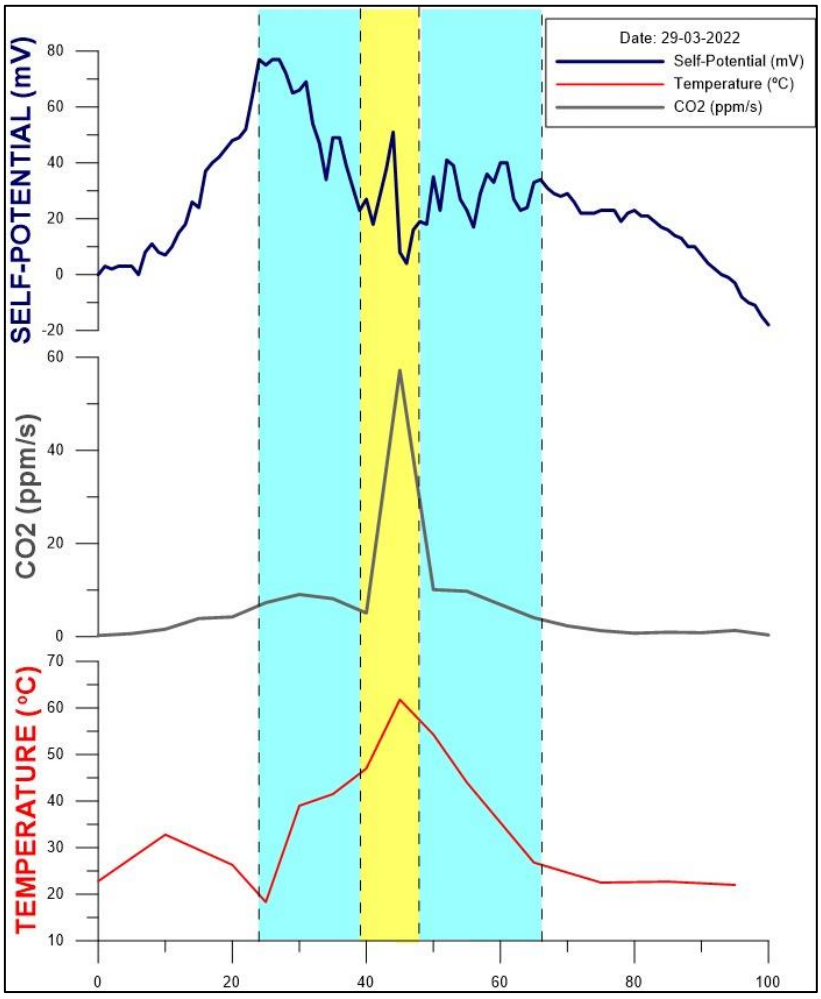
Análisis de perfil.

Un análisis comparativo de los datos de flujo de CO2 medidos por el equipo construido en este proyecto, se complementa con los datos tomados de medición de otros métodos de monitoreo volcánico.

Los métodos con los que se comparan son el método de Potencial Espontáneo y la Temperatura del suelo. Con lo cual podemos observar:

Ilustración 90

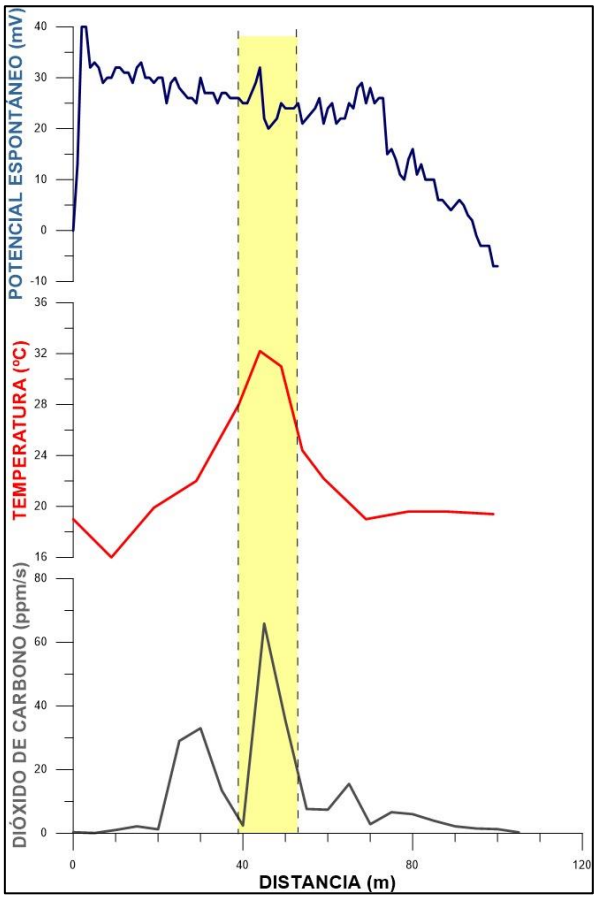
Gráfica comparativa de mediciones. 29/marzo/2022



De igual manera graficamos para el siguiente día:

Ilustración 91

Gráfica comparativa de mediciones. 13/julio/2022



Con esto observamos que, la fractura geológica pasa por el volcán, coincidiendo en el punto 45 de los datos medidos. Los niveles medidos muestran un comportamiento bastante similar, a pesar de tener meses de diferencia. Así se establece un parámetro base de monitoreo, del cual se encargan de ahora en adelante los vulcanólogos de la Universidad de El Salvador en continuar monitoreando y comparando el comportamiento del volcán de Santa Ana.

Este proceso de medición puede llevarse a cabo en cada uno de los diferentes volcanes que se desee tener en monitoreo.

CAPÍTULO V

CONCLUSIONES

•Se ha creado un enlace benéfico entre el área de vulcanología junto con el área de ingeniería, para trabajar con equipos de monitoreo volcánico.

•Con la implementación de este equipo de medición se sientan las bases para la construcción de más equipos de medición similares al construido en este proyecto, tomando como principal objetivo la mejora continua de estos.

•Con la finalización de este proyecto se logró adaptar el mecanismo de funcionamiento de un sensor de CO2 para su uso portátil e inalámbrico a través del diseño de una red inalámbrica para transferencia de datos entre el dispositivo sensor y una aplicación móvil para el almacenamiento y recopilación en tiempo real y el funcionamiento de todo el equipo por medio del uso de batería recargable

•La configuración del Sensor LI-830 se llevó a cabo a través de una conexión USB hacia una computadora y con una terminal, este se configura en XML (Lenguaje de Marcado Extensible). Se configuro para transmitir datos cada 1 segundo, debido a esto, el microcontrolador lee datos de concentración de CO2 cada 1 segundo, por tanto, se vio la importancia de usar tiempos de espera, para las distintas tareas que el microcontrolador tiene que realizar: actualizar la temperatura y humedad de la CPU, encender y apagar la bomba de gas, etc. De no programar estos retardos, el microcontrolador puede pasar por alto alguna orden enviada desde la aplicación Android.

•Vemos de las tablas resumen obtenidas por medio de la aplicación, que el punto 45 es donde se ubica la zona anómala. Debido al alto valor de flujo de emisión de CO2 obtenido de aproximadamente 57.2 y 65.9 ppm/s. Se observa la tendencia de incremento en el flujo a medida nos acercamos a la zona anómala y el decremento después de esta zona.

•Con el microcontrolador ATmega328P se logró un funcionamiento más rápido y eficiente del equipo que, usando un microordenador, tal es el caso de la Raspberry Pi con la que se contaba anteriormente, ya que el microordenador es más lento al momento de encenderse debido a que tiene que cargar un sistema operativo, en cambio el microcontrolador realiza la tarea que se le ha sido programada al momento de encenderse y realiza una única tarea concreta.

•La construcción de un equipo de medición implementando este diseño, permite un ahorro económico aproximadamente del 60%. En comparación con las compañías extranjeras encontradas en el mercado.

REFERENCIAS

AKRIBIS. (2021). *AKRIBIS Sinergizando Conocimiento y Tecnología*. Obtenido de Importancia de la medición de CO2: https://www.akribis.info/web/importancia-de-la-medicion-de-co2/

Aosong Electronics Co.,Ltd. (s.f.). *Digital-output relative humidity & temperature sensor/module.* Aosong: Product Data Sheet.

ARDUINO. (2022). ARDUINO UNO R3.

Cambio Energético. (16 de Febrero de 2022). *Comparación de baterías de plomo-ácido y batería de litio para energía solar*. Obtenido de https://www.cambioenergetico.com/blog/comparacion-baterias-de-plomo-acido-y-bateria-de-litio-para-energia-solar/

Cursa. (s.f.). *La Geekipedia De Ernesto*. Obtenido de https://cursa.app/es/profesor/la-geekipedia-de-ernesto

Developers. (s.f.). *Configura tu compilación*. Obtenido de https://developer.android.com/studio/build?hl=es-419

Developers. (s.f.). *Introducción a Android Studio*. Obtenido de https://developer.android.com/studio/intro?hl=es-419

Future Electronic Corporation. (s.f.). *FEC Relay Modules.* Product Data Sheet.

Guangzhou Tecnology Co.Ltd. (s.f.). *Guangzhou HC Information Tecnology Co.Ltd.* Guangzhou: Product Data Sheet.

IBM. (s.f.). *R cuadrado ajustado*. Obtenido de https://www.ibm.com/docs/es/cognos-analytics/11.2.0?topic=terms-adjusted-r-squared

LI-COR. (2018). *The LI-830 and LI-850 Integrator's Guide.* Lincoln, Nebraska.

LI-COR. (2021). *LI-830 and LI-850 Gas Analyzers For Continuous Monitoring Applications.* Lincoln, Nebraska.

LI-COR. (2021). *Using the LI-830 and LI-850 Gas Analyzers.* Lincoln, Nebraska.

LI-COR. (s.f.). *New LI-830 CO2 and LI-850 CO2/H2O Analyzers*. Obtenido de https://www.licor.com/env/products/gas_analysis/LI-830_LI-850/

Llamas, L. (16 de abril de 2014). *Comunicación de arduino con puerto serie*. Recuperado el 2022, de https://www.luisllamas.es/arduino-puerto-serie/

Llamas, L. (27 de Septiembre de 2016). *Localización gps con arduino y los modulos gps neo-6*. Recuperado el 2022, de https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/

Loachamin, L. (s.f.). *Innova Domotics*. Recuperado el Mayo de 2022, de https://cursos.innovadomotics.com/

Maxim Integrated Products. (2003). *±15kV ESD-Protected, +5V RS-232 Transceivers.* Product Data Sheet.

Naylamp Mechatronics. (s.f.). *ATMEGA328P DIP*. Obtenido de https://naylampmechatronics.com/microcontroladores/111-atmega328p-dip.html

Naylamp Mechatronics. (s.f.). *Tutorial módulo gps con arduino*. Obtenido de https://naylampmechatronics.com/blog/18_tutorial-modulo-gps-con-arduino.html

Observatorio Vulcanológico INGEMMET. (2022). *Sector Energías y Minas*. Obtenido de Monitoreo Volcánico: http://ovi.ingemmet.gob.pe/?page_id=26#:~:text=Consiste%20en%20registrar%20y%20conocer,interior%20de%20la%20estructura%20volc%C3%A1nica.

PELICAN. (2022). *PELICAN*. Obtenido de Cases//Vault/V200: https://www.pelican.com/la/es/product/cases/vault/V200

Sánchez, P. L. (2022). *Udemy*. Obtenido de Master en Arduino 2022 ¡IoT! Internet oF Things y mucho más!: https://www.udemy.com/course/master-en-arduino/

STEREN. (2022). *Tienda Steren*. Obtenido de https://www.steren.com.sv/switch-de-push-con-boton-redondo-metalico-e-iluminacion-normalmente-abierto-o-normalmente-cerrado.html

Steren. (s.f.). *Batería sellada de ácido-plomo, 12 Vcc 7 Ah*. Obtenido de https://www.steren.com.sv/bateria-sellada-de-acido-plomo-12-vcc-7-ah.html

Superprof. (s.f.). *Ejercicios de correlación y regresión*. Obtenido de https://www.superprof.es/apuntes/escolar/matematicas/estadistica/disbidimension/ejercicios-de-correlacion-y-regresion.html

Texas Instruments . (2013). *LM2596 SIMPLE SWITCHER® Power Converter 150 kHz.* Dallas, Texas: Product Data Sheet.

GLOSARIO

Arduino: es una placa que tiene todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador. Es decir, es una placa impresa con los componentes necesarios para que funcione el microcontrolador y su comunicación con un ordenador a través de la comunicación serial.

Gradle: Es un paquete de herramientas de compilación avanzadas, para automatizar y administrar el proceso de compilación (Developers, s.f.).

Cámara magmática: es la zona donde se almacena el magma (roca fundida) proveniente del manto, el cual posteriormente es expulsado a la superficie en forma de erupción volcánica. La cámara magmática se comunica con el cráter del volcán a través de un conducto conocido como chimenea.

Comunicación inalámbrica: es aquella en la que la comunicación (emisor/receptor) no se encuentra unida por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio. En este sentido, los dispositivos físicos solo están presentes en los emisores y receptores de la señal, entre los cuales se encuentran antenas, computadoras portátiles, PDAs, teléfonos móviles, etcétera.

Concentración de gas: la cantidad de gas que se queda en la atmósfera después de las complejas interacciones que tienen lugar entre la atmósfera, la biosfera, y los océanos. Existen varias formas de expresar concentraciones. En gases las más utilizadas son %v/v (porcentaje en volumen), mg/L, g/m3, µg/mL y µmol/mol.

Dióxido de carbono (CO2): es un compuesto de carbono y oxígeno que existe como gas incoloro en condiciones de temperatura y presión estándar. Está íntimamente relacionado con el efecto invernadero.

Emisión de gas: Se entiende por emisión la cantidad de gas que se libera a la atmósfera

Estación fija: estructura destinada a permanecer fijo o inmóvil con la finalidad operativa de llevar un enlace de los datos capturados en un mismo lugar.

Estructura volcánica: un volcán es una abertura en la corteza terrestre a través de la cual roca fundida, gases y escombros escapan a la superficie.

Flujo de gas: describe cualquier efecto que parece pasar o viajar a través de una superficie o sustancia, esta sustancia que fluye va descrita a la de un gas.

Fractura geológica: se entiende por fractura cualquier superficie de discontinuidad producida por la rotura de una masa rocosa (deformación frágil). Hay dos tipos principales de fracturas: las fallas y las diaclasas.

IDE (Entorno de Desarrollo Integrado): es un sistema de software para el diseño de aplicaciones que combina herramientas comunes para desarrolladores en una sola interfaz de usuario gráfica (GUI).

Monitoreo: es un proceso permanente que consiste en revisar el cumplimiento de las actividades programadas y si con esas actividades estamos alcanzando las metas propuestas.

Monitoreo volcánico: Consiste en registrar y conocer la dinámica del volcán a partir de los diferentes tipos de sismos asociados al fracturamiento de rocas (sismo volcando – tectónicos),

ascenso y acumulación de magma y gases (sismo tipo largo periodo, tremor, explosión) que ocurren en el interior de la estructura volcánica

Regresión: en estadística, el análisis de la regresión es un proceso estadístico para estimar las relaciones entre variables.

Salida de campo: método tradicional de la geología apoyados con equipos de ensayo y almacenaje de información adaptados a cada trabajo concreto utilizados por técnicos especializados. Se hace la toma de muestras y realización de ensayos.

Sensor: son herramientas que detectan y responden a algún tipo de información del entorno físico. es un dispositivo que detecta el cambio en el entorno y responde a alguna salida en el otro sistema. Un sensor convierte un fenómeno físico en un voltaje analógico medible (o, a veces, una señal digital) convertido en una pantalla legible para humanos o transmitida para lectura o procesamiento adicional.

Sistema de falla: en el campo de la geología, se denomina falla a una fractura, generalmente plana, en el terreno a lo largo de la cual se han deslizado los dos bloques el uno respecto al otro.

Transmisión de datos: es el proceso de transmisión de un flujo continuo de datos (también conocidos como flujos) que generalmente se introducen en el software de procesamiento de flujos para obtener información valiosa. Un flujo de datos consta de una serie de elementos de datos ordenados en el tiempo.

Vulcanología: es la rama de la geología que estudia el vulcanismo y todas sus manifestaciones, como volcanes, géiseres, fumarolas, erupciones volcánicas, magmas, lavas, tefras, etc... Los vulcanólogos visitan frecuentemente los volcanes terrestres, en especial los que

están activos, para observar las erupciones y recoger restos volcánicos como la tefra (ceniza o piedra pómez), rocas y muestras de lava.

ANEXOS

A) CÓDIGOS

Código Android Studio MainActivity.java

```
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;

import lecho.lib.hellocharts.model.Axis;
import lecho.lib.hellocharts.model.Line;
import lecho.lib.hellocharts.model.LineChartData;
import lecho.lib.hellocharts.model.PointValue;
import lecho.lib.hellocharts.model.Viewport;
import lecho.lib.hellocharts.view.LineChartView;
import pl.pawelkleczkowski.customgauge.CustomGauge;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Environment;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
```

```
import java.util.Locale;
import java.util.Set;
import java.util.UUID;


public class MainActivity extends AppCompatActivity {

    //------------------------------------------------------------------------
-----
    private static final String TAG = "DispositivosVinculados";
    //------------------------------------------------------------
    // Variables Bluetooth
    // SPP UUID service - this should work for most devices
    private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-
1000-8000-00805F9B34FB");
    // String for MAC address
    private static String address = null;
    // Member fields
    private BluetoothAdapter mBtAdapter = null;
    private ArrayAdapter<String> mAddressDevices, mNameDevices;
    Handler bluetoothIn;
    final int handlerState = 0;                         //used to identify
handler message
    private BluetoothSocket btSocket = null;
    private StringBuilder recDataString = new StringBuilder();
    private ConnectedThread MyConexionBT;
    //------------------------------------------------------------
    // declare button for launching website and textview for connection
status
    LineChartView lineChartView;
    Button IdBtnBuscar, IdBtnConect, IdBtnDesconectar, IdBtnGps,
Id_BtnPointMeasure, IdBtnCalcular;
    Spinner DisEncontrados;
    TextView IdTxvCo2, IdTxvMensaje, IdTxvCo2_G, IdTxvLat, IdTxvLon,
IdTxvSat, IdTxvPrec, IdTxvHmd, IdTxvTmp;
    CustomGauge IdMedidorTemp, IdMedidorHumd;

    EditText idTxtPoint;
    TextView etiqueta,puntoAnterior;

    double DataCO2;
    private Button buttonStartThread, IdBtnStop;

    ArrayList<String> ListStrHora;
    ArrayList<Float> ListValCO2;

    String medidaFlujo;
    float R2;
    String StrLat, StrLon, StrSat, StrPre, StrTemp, StrHumd;

    Date sessionTime = Calendar.getInstance().getTime();
    String[] sessionTimes = sessionTime.toString().split(" ");
    String Point;
    String filename = "Datos Punto " + Point + "-" + sessionTimes[0] + "-" +
sessionTimes[1] + "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";
    String filenameDos = "Reporte " + sessionTimes[0] + "-" + sessionTimes[1]
+ "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Poner el icono en el ActionBar
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setIcon(R.mipmap.ic_launcher);

        //-------------------------------------------------------
        //-------------------------------------------------------
        mAddressDevices = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);
        mNameDevices = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);

        DisEncontrados = (Spinner) findViewById(R.id.DisEncontrados);

        lineChartView = findViewById(R.id.chart);
        IdMedidorTemp = (CustomGauge) findViewById(R.id.IdMedidorTemp);
        IdMedidorHumd = (CustomGauge) findViewById(R.id.IdMedidorHumd);

        IdTxvCo2 = (TextView) findViewById(R.id.IdTxvCo2);
        IdTxvMensaje = (TextView) findViewById(R.id.IdTxvMensaje);
        IdTxvCo2_G = (TextView) findViewById(R.id.IdTxvCo2_G);
        IdTxvLat = (TextView) findViewById(R.id.IdTxvLat);
        IdTxvLon = (TextView) findViewById(R.id.IdTxvLon);
        IdTxvSat = (TextView) findViewById(R.id.IdTxvSat);
        IdTxvPrec = (TextView) findViewById(R.id.IdTxvPrec);
        puntoAnterior=(TextView)findViewById(R.id.puntoAnterior);

        IdTxvHmd = (TextView) findViewById(R.id.IdTxvHmd);
        IdTxvTmp = (TextView) findViewById(R.id.IdTxvTmp);

        IdBtnBuscar = (Button) findViewById(R.id.IdBtnBuscar);
        IdBtnConect = (Button) findViewById(R.id.IdBtnConect);
        IdBtnDesconectar = (Button) findViewById(R.id.IdBtnDesconectar);
        IdBtnGps = (Button) findViewById(R.id.IdBtnGps);
        Id_BtnPointMeasure = (Button) findViewById(R.id.Id_BtnPointMeasure);
        IdBtnCalcular = (Button) findViewById(R.id.IdBtnCalcular);

        buttonStartThread = findViewById(R.id.button_start_thread);
        IdBtnStop = (Button) findViewById(R.id.IdBtnStop);

        idTxtPoint = (EditText) findViewById(R.id.idTxtPoint);
        etiqueta = findViewById(R.id.etiqueta);

        ListStrHora = new ArrayList();
        ListValCO2 = new ArrayList();

        try {
            ListStrHora.clear();
            ListValCO2.clear();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        //-------------------------------------------------
        bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) {
                    String readMessage = (String) msg.obj;
                    recDataString.append(readMessage);
                    int endOfLineIndex = recDataString.indexOf("#");

                    if (endOfLineIndex > 0) {
                        String dataInPrint = recDataString.substring(0,
endOfLineIndex);

                        //467.53,0,0,0,0,0,0,0#
                        //0,BombaOn,0,0,0,0,0,0#
                        //0,0,LAT,LON,PREC,SAT,0,0,0#
                        //0,0,0,0,0,0,temp,hmd#

                        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);

                            } else {
                                DataCO2 = Double.parseDouble(StrCO2);
                                IdTxvCo2.setText(String.valueOf(DataCO2 + "
ppm"));

                            }

                        } catch (Exception e) {

                        }
                        recDataString.delete(0, recDataString.length());
                    }
                }
            }
```

```
        };

        mBtAdapter = BluetoothAdapter.getDefaultAdapter();
        //-------------------------------------------------
        IdBtnBuscar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DispositivosVinculados();
            }
        });

        IdBtnConect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int IntValSpin = DisEncontrados.getSelectedItemPosition();
                address = mAddressDevices.getItem(IntValSpin);
                ConectarDispBT();
                //Toast.makeText(getBaseContext(),
String.valueOf(IntValSpin)+" - "+address , Toast.LENGTH_SHORT).show();
                IdBtnGps.setEnabled(true);
                Id_BtnPointMeasure.setEnabled(true);
            }
        });

        IdBtnGps.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MyConexionBT.write("P#");
                IdTxvLat.setText("");
                IdTxvLat.setText(StrLat);
                IdTxvLon.setText("");
                IdTxvLon.setText(StrLon);
                IdTxvSat.setText("");
                IdTxvSat.setText(StrSat);
                IdTxvPrec.setText("");
                IdTxvPrec.setText(StrPre);
            }
        });

        buttonStartThread.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                IdTxvMensaje.setText("");
                IdTxvMensaje.setText("Bomba ON");
                MyConexionBT.write("G#");
                IdBtnStop.setEnabled(true);
                buttonStartThread.setEnabled(false);
                bluetoothIn = new Handler() {
                    public void handleMessage(android.os.Message msg) {
                        if (msg.what == handlerState) {
                            String readMessage = (String) msg.obj;
                            recDataString.append(readMessage);
                            int endOfLineIndex = recDataString.indexOf("#");

                            if (endOfLineIndex > 0) {
                                String dataInPrint =
recDataString.substring(0, endOfLineIndex);
```

```
                                    //467.53,0,0,0,0,0,0,0#
                                    //0,BombaOn,0,0,0,0,0,0#
                                    //0,0,LAT,LON,PREC,SAT,0,0,0#
                                    //0,0,0,0,0,0,temp,hmd#
                                    try {
                                        String[] parts = dataInPrint.split(",");
                                        String StrCO2 = parts[0];
                                        String StrRespIn = parts[1];

                                        if (!StrRespIn.equals("0")) {
                                            IdTxvMensaje.setText(" ");
                                            IdTxvMensaje.setText(StrRespIn);

                                        } else {
                                            DataCO2 = Double.parseDouble(StrCO2);


IdTxvCo2_G.setText(String.valueOf("CO2: " + DataCO2 + " ppm"));
                                            int DataIntCO2 = (int) DataCO2;

                                            Date measureTime =
Calendar.getInstance().getTime();
                                            String[] currentTime =
measureTime.toString().split(" ");
                                            String measureLine = currentTime[0] +
currentTime[1] + currentTime[2] + "," + currentTime[3] + "," + parts[0] +
"\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("#9C27B0"));

                                            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.setMaximumViewport(viewport);

lineChartView.setCurrentViewport(viewport);

//lineChartView.setMinimumHeight(2000);
                                    }
                                } catch (Exception e) {

                                }
                                recDataString.delete(0,
recDataString.length());
                            }
                        }
                    }
                };

            }
        });

        IdBtnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                IdTxvMensaje.setText("");
                IdTxvMensaje.setText("Bomba OFF");
                MyConexionBT.write("H#");
                IdBtnCalcular.setEnabled(true);
                IdBtnStop.setEnabled(false);
                if (btSocket != null) {
                    try {
                        btSocket.close();
                    } catch (IOException e) {
                        Toast.makeText(getBaseContext(), "Error",
Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });
```

```
        IdBtnDesconectar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                if (btSocket != null) {
                    try {
                        btSocket.close();
                    } catch (IOException e) {
                        Toast.makeText(getBaseContext(), "Error",
Toast.LENGTH_SHORT).show();
                    }
                }
                finish();
                //System.exit(0);
            }
        });
        //-----------------------------------------------------------
        //-----------------------------------------------------------
        //"puntos" sera el nombre del archivo que almacenara cualquier número
que se escriba en el EditText
        SharedPreferences preferences = getSharedPreferences("puntos",
Context.MODE_PRIVATE);
        idTxtPoint.setText(preferences.getString("id", ""));//Las comillas
vacias sera el sitio donde se guardara lo que escribamos y "id" es la
referencia de lo que guardaremos
        puntoAnterior.setText("Punto anterior: "+idTxtPoint.getText());
        idTxtPoint.setText("");
    }

    //PARA GUARDAR ARCHIVOS EN RUTAS ABSOLUTAS
    public void etiqueta(View view) {

        //INTRODUCCIÓN DE PUNTO DE MEDICIÓN
        switch (view.getId()) {
            case R.id.Id_BtnPointMeasure:
                MyConexionBT.write("F#");
                etiqueta.setText("Punto de medición: " +
idTxtPoint.getText());

                SharedPreferences preferencias =
getSharedPreferences("puntos",Context.MODE_PRIVATE);
                SharedPreferences.Editor Obj_editor =
preferencias.edit();//Vamos a editar el archivo sharedpreferencias que creaos
previamente
                //Con ayuda del Editor creamos el objeto editor para editar
el preferencias
                Obj_editor.putString("id", idTxtPoint.getText().toString());
                Obj_editor.commit();//Es apara confirmar que lo que cabamos
de editar si se va a guardar

                Id_BtnPointMeasure.setEnabled(false);
                IdBtnGps.setEnabled(false);
                try {
                    Thread.sleep(1000);
                    buttonStartThread.setEnabled(true);
                } catch (Exception errorTiempo) {
                    errorTiempo.printStackTrace();
```

```
                }
                break;
        }
        Point = idTxtPoint.getText().toString();
        filename = "Datos Punto " + Point + "-" + sessionTimes[0] + "-" +
sessionTimes[1] + "-" + sessionTimes[2] + "-" + sessionTimes[5] + ".csv";
        idTxtPoint.setEnabled(false);
    }

    public boolean isExternalStorageWritable() {
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            return true;
        }
        return false;
    }

    public boolean isExternalStorageReadable() {
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state) ||
                Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
            return true;
        }
        return false;
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public File getDocumentStorageDir() {
        // Get the directory for the user's public pictures directory.
        File file = new File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_DOCUMENTS), "CO2 measures");
        if (!file.mkdirs()) {
        }
        return file;
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public File getDocumentStorageDirDos() {
        // Get the directory for the user's public pictures directory.
        File fileDos = new
File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_DOCUMENTS), "CO2 Reportes");
        if (!fileDos.mkdirs()) {
        }
        return fileDos;
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public void createFile(String measures) {
        if (isExternalStorageWritable()) {
            try {
                //String header = "System_Date_(Y-M-
D),System_Time_(h:m:s),CO2_(µmol_mol⁻¹),Cell_Temperature_(°C),Cell_Pressure_(k
Pa),CO2_Absorption,Input_Voltage_(V),Flow_Rate_(L_min⁻¹)\n";
                String header = "System_Date_(Y-M-
D),System_Time_(h:m:s),CO2_(µmol_mol⁻¹)\n";
```

```
                File file = new File(getDocumentStorageDir(), filename);
                if (!file.exists()) {
                    file.createNewFile();
                    FileWriter fw = new FileWriter(file.getAbsoluteFile());
                    BufferedWriter bw = new BufferedWriter(fw);
                    bw.write(header);
                    bw.write(measures);
                    bw.close();
                } else if (file.exists()) {
                    FileWriter fw = new FileWriter(file.getAbsoluteFile(),
true);
                    BufferedWriter bw = new BufferedWriter(fw);
                    bw.write(measures);
                    bw.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
                Context contexto = getApplicationContext();
                CharSequence erroraso = "Error escribiendo archivo";
                int duration = Toast.LENGTH_SHORT;
                Toast toast = Toast.makeText(contexto, erroraso, duration);
            }
        }
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public void createFileDos(String flux) {
        if (isExternalStorageWritable()) {
            try {
                //String header = "System_Date_(Y-M-
D),System_Time_(h:m:s),CO2_(µmol_mol⁻¹),Cell_Temperature_(°C),Cell_Pressure_(k
Pa),CO2_Absorption,Input_Voltage_(V),Flow_Rate_(L_min⁻¹)\n";
                String header = "Point,System_Date_(Y-M-
D),System_Time_(h:m:s),FluxCO2(µmol_mol⁻¹)/s,R^2\n";
                File fileDos = new File(getDocumentStorageDirDos(),
filenameDos);
                if (!fileDos.exists()) {
                    fileDos.createNewFile();
                    FileWriter fwDos = new
FileWriter(fileDos.getAbsoluteFile());
                    BufferedWriter bwDos = new BufferedWriter(fwDos);
                    bwDos.write(header);
                    bwDos.write(flux);
                    bwDos.close();
                } else if (fileDos.exists()) {
                    FileWriter fwDos = new
FileWriter(fileDos.getAbsoluteFile(), true);
                    BufferedWriter bwDos = new BufferedWriter(fwDos);
                    bwDos.write(flux);
                    bwDos.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
                Context contexto = getApplicationContext();
                CharSequence erroraso = "Error escribiendo archivo";
                int duration = Toast.LENGTH_SHORT;
```

```
                Toast toast = Toast.makeText(contexto, erroraso, duration);
            }
        }
    }

    //CALCULO DE LA REGRESION LINEAL Y FLUJO
    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public void calcularRegresion(View view) {
        Integer[] limites = getLims(); //Obtenemos los limites del metodo
"getLims"
        if (limites[0] == limites[1]) {
            return;
        } else {
            //Aqui pasamos la data de los ArrayList "trimXData" y "trimYData"
a los ArrayList "nx" y "ny"
            ArrayList<Integer> nx = new ArrayList<>(trimXData(limites[0],
limites[1]));
            ArrayList<Float> ny = new ArrayList<>(trimYData(limites[0],
limites[1]));

            Float pendiente = new Float(calcPendiente(nx, ny));//Nos ayudams
del método "calcPendiente" y le pasamos los parametros "nx" y "ny"
            final TextView pendientlayer = (TextView)
findViewById(R.id.pendiente);
            pendientlayer.setText("Flujo " + pendiente.toString() + "
ppm/s");

            Float correlacion = new Float(calcCorrelacion(nx, ny));//Nos
ayudams del método "calcCorrelacion" y le pasamos los parametros "nx" y "ny"
            float correlacionR2 = (float) Math.pow(correlacion,2);
            float roundCorrelacionR2 = (float) (Math.round(correlacionR2 *
1000.0) / 1000.0);
            final TextView correlacionlayer = (TextView)
findViewById(R.id.correlacion);
            correlacionlayer.setText("R^2 = " + roundCorrelacionR2);

            //Aqui creamos el contenido del reporte----------------------
----------
            medidaFlujo = pendiente.toString();
            R2=roundCorrelacionR2;
            Date measureTimeDos = Calendar.getInstance().getTime();
            String[] currentTimeDos = measureTimeDos.toString().split(" ");
            String fluxLineDos = Point + "," + currentTimeDos[0] +
currentTimeDos[1] + currentTimeDos[2] + "," + currentTimeDos[3] + "," +
medidaFlujo + "," + R2 + "\n";
            createFileDos(fluxLineDos);
            //----------------------------------------------------------
-----------
        }
    }

    private Integer[] getLims() {
        final EditText lima = (EditText) findViewById(R.id.limita);
        final EditText limb = (EditText) findViewById(R.id.limitb);
        Integer limites[] = {0, 0}; //Creamos vector limites
        try {
            limites[0] = Integer.parseInt(lima.getText().toString());
```

```
//Guardamos en posicion 0 el valor de Limite A
            limites[1] = Integer.parseInt(limb.getText().toString());
            return limites;
        } catch (NumberFormatException e) {
            return limites;
        }
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public float calcPendiente(ArrayList<Integer> x, ArrayList<Float> y) {
        float corrector = 1f;
        float pendiente = 0;
        float term1 = 0;
        float tem1 = 0;
        float term2 = 0;
        float sumx = 0;
        float sumy = 0;
        float term3 = 0;
        float tem3 = 0;
        float term4 = 0;

        //System.out.println(x);
        for (int i = 0; i < x.size(); i++) {
            tem1 = x.get(i) * corrector * y.get(i) + tem1;
            sumx = x.get(i) * corrector + sumx;
            sumy = y.get(i) + sumy;
            tem3 = (float) Math.pow(x.get(i) * corrector, 2) + tem3;
        }
        term1 = x.size() * tem1;
        term2 = sumx * sumy;
        term3 = x.size() * tem3;
        term4 = (float) Math.pow(sumx, 2);
        pendiente = (term1 - term2) / (term3 - term4);

        float roundPendiente = (float) (Math.round(pendiente * 1000.0) /
1000.0);
        return roundPendiente;
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    public float calcCorrelacion(ArrayList<Integer> x, ArrayList<Float> y) {
        float contadorX = 0;
        float contadorY = 0;
        float contadorX2 = 0;
        float contadorY2 = 0;
        float contadorXY = 0;
        int almacenX, almacenX2;
        float almacenY, almacenY2;
        float xy;
        //Aqui viene la correlacion
        for (int i = 0; i < x.size(); i++) {
            almacenX = x.get(i);
            contadorX = contadorX + almacenX;
            almacenX2 = (int) Math.pow(almacenX, 2);
            contadorX2 = contadorX2 + almacenX2;
        }
        float xMed = (contadorX / x.size());
```

```
        float xMed2 = (float) ((contadorX2 / x.size())-Math.pow(xMed, 2));
        float raizX2 = (float) Math.sqrt(xMed2);
        System.out.println("xMed = " + xMed);System.out.println("xMed2 = " +
xMed2);System.out.println("raizX2 = " + raizX2);

        for (int i = 0; i < x.size(); i++) {
            almacenY = y.get(i);
            contadorY = contadorY + almacenY;
            almacenY2 = (float) Math.pow(almacenY, 2);
            contadorY2 = contadorY2 + almacenY2;
        }
        float yMed = (contadorY / y.size());
        float yMed2 = (float) ((contadorY2 / x.size())-Math.pow(yMed, 2));
        float raizY2 = (float) Math.sqrt(yMed2);
        System.out.println("yMed = " + yMed);System.out.println("yMed2 = " +
yMed2);System.out.println("raizY2 = " + raizY2);

        for (int i = 0; i < x.size(); i++) {
            xy = x.get(i) * y.get(i);
            contadorXY = contadorXY + xy;
        }
        float xyMed_F = (contadorXY / x.size()) - xMed * yMed;
        float correlacion = xyMed_F / (raizX2 * raizY2);
        System.out.println("xyMed_F = " + xyMed_F);System.out.println("r = "
+ correlacion); System.out.println("TotalXY = " + contadorXY);

        return correlacion;
    }

    @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());//Del
método "GetFileMeasure" obtenemos la data y la guardamos en el ArrayList
"yValues"
        //Rellenamos el ArrayList "trimedY" con los datos del ArrayList
"yValues" desde el limite A al limte B
        for (int i = lima; i <= limb; i++) {
            trimedY.add(yValues.get(i));
        }
        return trimedY;
    }

    public ArrayList<Integer> trimXData(Integer lima, Integer limb) {
        ArrayList<Integer> trimedX = new ArrayList<>();
        //Aqui rellenamos el ArrayList "trimedX" con un contador ayudandonos
de un for
        for (int i = lima; i <= limb; i++) {
            trimedX.add(i);
        }
        return trimedX;
    }

    public static final String SEPARATOR = ",";

    @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[2]);
                        yValues.add(yPoints);
                    }
                }
                br.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return yValues;
    }

    //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    private BluetoothSocket createBluetoothSocket(BluetoothDevice device)
throws IOException {
        return device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        //creates secure outgoing connecetion with BT device using UUID
    }

    public void ConectarDispBT() {
        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();
        } catch (IOException e) {
            try {
                btSocket.close();
            } catch (IOException e2) {

            }
        }
        MyConexionBT = new ConnectedThread(btSocket);
        MyConexionBT.start();
    }
```

```
    public void DispositivosVinculados() {
        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());
                //.......... EN ESTE PUNTO GUARDO LOS NOMBRE A MOSTRARSE EN
EL COMBO BOX
                mNameDevices.add(device.getName());
                //.......... AUNQUE TAMBIEN DE REQUERIR PUEDO GUARDAR
TAMBIEN LA DIRECCION YA QUE ESO NO AFECTA
                //mNameDevices.add(device.getName() + "\n" +
device.getAddress());
            }
            //ACTUALIZO LOS DISPOSITIVOS
            DisEncontrados.setAdapter(mNameDevices);

        } else {
            String noDevices = "Ningun dispositivo pudo ser
emparejado".toString();
            mAddressDevices.add(noDevices);
            mNameDevices.add(noDevices);
        }
    }

    private void VerificarEstadoBT() {
        mBtAdapter = BluetoothAdapter.getDefaultAdapter(); // CHECK THIS OUT
THAT IT WORKS!!!
        if (mBtAdapter == null) {
            Toast.makeText(getBaseContext(), "El dispositivo no soporta
Bluetooth", Toast.LENGTH_SHORT).show();
        } else {
            if (mBtAdapter.isEnabled()) {
                Log.d(TAG, "...Bluetooth Activado...");
            } else {
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }

    private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket) {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            try {
```

```
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
            }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[256];
            int bytes;

            while (true) {
                try {
                    bytes = mmInStream.read(buffer);
                    String readMessage = new String(buffer, 0, bytes);
                    bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();
                } catch (IOException e) {
                    break;
                }
            }
        }

        public void write(String input) {
            byte[] msgBuffer = input.getBytes();
            try {
                mmOutStream.write(msgBuffer);
            } catch (IOException e) {
                Toast.makeText(getBaseContext(), "La Conexión fallo",
Toast.LENGTH_LONG).show();
                finish();
            }
        }

        public void WriteByte(byte input) {
            byte msgBuffer = input;
            try {
                mmOutStream.write(msgBuffer);
            } catch (IOException e) {
                Toast.makeText(getBaseContext(), "La Conexión fallo",
Toast.LENGTH_LONG).show();
                finish();
            }
        }
    }
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    //MÉTODO PARA MOSTRAR EL MENÚ
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.overflow, menu);
        return true;
    }

    //MÉTODO PARA ASIGNAR LAS FUNCIONES CORRESPONDIENTES A LAS OPCIONES DEL
```

```
MENU
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.item1) {
            Intent acercade = new Intent(this, AcercaDeActivity.class);
            startActivity(acercade);
        } else if (id == R.id.item2) {
            Intent ayuda = new Intent(this, AyudaActivity.class);
            startActivity(ayuda);
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Código Android Studio activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >

            <Button
                android:id="@+id/IdBtnBuscar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/strBuscar" />

            <Spinner
                android:id="@+id/DisEncontrados"
                android:layout_width="match_parent"
                android:layout_height="48dp"
                android:entries="@array/dis_vin" />

            <Button
                android:id="@+id/IdBtnConect"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/strConect" />

            <Space
                android:layout_width="match_parent"
```

```
        android:layout_height="15dp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/strco2"
        android:textAlignment="center"
        android:textSize="16sp"
        android:textStyle="bold" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:contentDescription="@string/strconten"
            app:srcCompat="@drawable/co2" />

        <TextView
            android:id="@+id/IdTxvCo2"
            android:layout_width="50dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="@string/strlectura"
            android:textColorHint="#546E7A" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <Space
            android:layout_width="match_parent"
            android:layout_height="15dp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <LinearLayout
                android:layout_width="205dp"
                android:layout_height="match_parent"
                android:orientation="vertical"
                android:paddingLeft="3dp">

                <TextView
                    android:id="@+id/textView3"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
```

```
            android:text="@string/strgps"
            android:textAlignment="center"
            android:textStyle="bold" />

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:layout_gravity="center"
            app:srcCompat="@drawable/gps" />

        <Space
            android:layout_width="match_parent"
            android:layout_height="15dp" />

        <Button
            android:id="@+id/IdBtnGps"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/str_actgps"
            android:textSize="12sp"
            android:enabled="false"/>

        <TextView
            android:id="@+id/IdTxvLat"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/strlectura"
            android:paddingLeft="2dp" />

        <TextView
            android:id="@+id/IdTxvLon"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/strlectura"
            android:paddingLeft="2dp" />

        <TextView
            android:id="@+id/IdTxvSat"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/strlectura"
            android:paddingLeft="2dp" />

        <TextView
            android:id="@+id/IdTxvPrec"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/strlectura"
            android:paddingLeft="2dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingRight="3dp">
```

```
                        <TextView
                            android:id="@+id/textView14"
                            android:layout_width="match_parent"
                            android:layout_height="wrap_content"
                            android:text="@string/strTyH"
                            android:textAlignment="center"
                            android:textStyle="bold" />

                        <androidx.constraintlayout.widget.ConstraintLayout
                            android:layout_width="match_parent"
                            android:layout_height="95dp">

                            <TextView
                                android:id="@+id/IdTxvTmp"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/strVal"

app:layout_constraintBottom_toBottomOf="parent"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
                                app:layout_constraintTop_toTopOf="parent" />

                            <TextView
                                android:id="@+id/textView15"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/strTemp"
                                android:textSize="12sp"

app:layout_constraintBottom_toBottomOf="@+id/IdMedidorTemp"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
/>

                            <pl.pawelkleczkowski.customgauge.CustomGauge
                                android:id="@+id/IdMedidorTemp"
                                android:layout_width="wrap_content"
                                android:layout_height="0dp"
                                android:layout_centerHorizontal="true"

                                app:gaugeEndValue="100"
                                app:gaugePointEndColor="@color/md_red_900"
                                app:gaugePointStartColor="@color/md_red_600"
                                app:gaugeStartAngle="135"
                                app:gaugeStartValue="0"
                                app:gaugeStrokeCap="ROUND"
                                app:gaugeStrokeColor="@color/md_grey_300"
                                app:gaugeStrokeWidth="10dp"
                                app:gaugeSweepAngle="270"

app:layout_constraintBottom_toBottomOf="parent"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
                                app:layout_constraintTop_toTopOf="parent" />
                        </androidx.constraintlayout.widget.ConstraintLayout>
```

```
                        <Space
                            android:layout_width="wrap_content"
                            android:layout_height="10dp" />

                        <androidx.constraintlayout.widget.ConstraintLayout
                            android:layout_width="match_parent"
                            android:layout_height="95dp">

                            <TextView
                                android:id="@+id/IdTxvHmd"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/strVal"

app:layout_constraintBottom_toBottomOf="parent"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
                                app:layout_constraintTop_toTopOf="parent" />

                            <TextView
                                android:id="@+id/textView17"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/strHumd"
                                android:textSize="12sp"

app:layout_constraintBottom_toBottomOf="@+id/IdMedidorHumd"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
/>

                            <pl.pawelkleczkowski.customgauge.CustomGauge
                                android:id="@+id/IdMedidorHumd"
                                android:layout_width="wrap_content"
                                android:layout_height="0dp"
                                android:layout_centerHorizontal="true"

                                app:gaugeEndValue="100"
                                app:gaugePointEndColor="@color/md_blue_900"
                                app:gaugePointStartColor="@color/md_blue_600"
                                app:gaugeStartAngle="135"
                                app:gaugeStartValue="0"
                                app:gaugeStrokeCap="ROUND"
                                app:gaugeStrokeColor="@color/md_grey_300"
                                app:gaugeStrokeWidth="10dp"
                                app:gaugeSweepAngle="270"

app:layout_constraintBottom_toBottomOf="parent"
                                app:layout_constraintEnd_toEndOf="parent"
                                app:layout_constraintStart_toStartOf="parent"
                                app:layout_constraintTop_toTopOf="parent" />
                        </androidx.constraintlayout.widget.ConstraintLayout>

                    </LinearLayout>
                </LinearLayout>
```

```
<Space
    android:layout_width="match_parent"
    android:layout_height="20dp" />

<TextView
    android:id="@+id/textView11"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/strTituloGuard"
    android:textAlignment="center"
    android:textSize="16sp"
    android:textStyle="bold" />

<Space
    android:layout_width="match_parent"
    android:layout_height="10dp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/etiqueta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="@string/strIndicacion"
        android:paddingLeft="5dp"
        android:textAlignment="viewStart"
        android:textColorHint="#546E7A" />

    <EditText
        android:id="@+id/idTxtPoint"
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:hint="@string/strIntroTitulo"
        android:inputType="number"
        android:textColorHint="?android:attr/textColorHint"
        android:textSize="16sp" />

</LinearLayout>

<TextView
    android:id="@+id/puntoAnterior"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/strPuntoAnterior"
    android:paddingLeft="5dp"
    android:textAlignment="textStart"
    android:textColorHint="#546E7A"
    android:textSize="10.5sp"
    android:textStyle="italic" />

<Space
```

```
            android:layout_width="match_parent"
            android:layout_height="15dp" />

        <Button
            android:id="@+id/Id_BtnPointMeasure"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:enabled="false"
            android:onClick="etiqueta"
            android:text="@string/strGuardar" />

        <Space
            android:layout_width="match_parent"
            android:layout_height="10dp" />

        <TextView
            android:id="@+id/textView8"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/strGrafico"
            android:textAlignment="center"
            android:textSize="16sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/IdTxvCo2_G"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/strlectura"
            android:paddingLeft="3dp"
            android:textColorHint="#546E7A" />

        <lecho.lib.hellocharts.view.LineChartView
            android:id="@+id/chart"
            android:layout_width="match_parent"
            android:layout_height="450dp"
            android:layout_marginStart="0dp"
            android:layout_marginTop="0dp"
            android:layout_marginEnd="0dp"
            android:layout_marginBottom="0dp"
            android:paddingLeft="15dp"
            android:paddingRight="5dp"
            android:paddingBottom="5dp" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button_start_thread"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="startThread"
```

```
            android:text="@string/strBombaOn"
            android:enabled="false"/>

        <Button
            android:id="@+id/IdBtnStop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="stopThread"
            android:text="@string/strBombaOff"
            android:enabled="false"/>

        <TextView
            android:id="@+id/IdTxvMensaje"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="@string/strEstado"
            android:paddingLeft="5dp"
            android:textColorHint="?android:attr/textColorHint" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <Space
            android:layout_width="match_parent"
            android:layout_height="10dp" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/str_tituloRegre"
            android:textAlignment="center"
            android:textSize="16sp"
            android:textStyle="bold" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/str_LimiteA"
            android:textAlignment="center"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/limita"
```

```
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:layout_weight="1"
        android:ems="10"
        android:hint="@string/str_IntroA"
        android:inputType="number" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/str_LimiteB"
        android:textAlignment="center"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/limitb"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:layout_weight="1"
        android:ems="10"
        android:hint="@string/str_IntroB"
        android:inputType="number" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <Button
        android:id="@+id/IdBtnCalcular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:enabled="false"
        android:onClick="calcularRegresion"
        android:text="@string/strCalcular" />

    <TextView
        android:id="@+id/pendiente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="@string/str_Flujo"
        android:textAlignment="center"
        android:textColorHint="?android:attr/textColorHint"
        android:textSize="16sp"
        android:textStyle="bold" />
```

```
                <TextView
                    android:id="@+id/correlacion"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:hint="@string/str_correlacion"
                    android:textAlignment="center"
                    android:textSize="16sp"
                    android:textStyle="bold" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">

                <Space
                    android:layout_width="match_parent"
                    android:layout_height="15dp" />

                <Button
                    android:id="@+id/IdBtnDesconectar"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:text="@string/strdesc" />

                <Space
                    android:layout_width="match_parent"
                    android:layout_height="15dp" />
            </LinearLayout>

        </LinearLayout>
    </ScrollView>

</RelativeLayout>
```

Código del microcontrolador

```
#include <SoftwareSerial.h>
#define DEBUG(a) Serial.println(a);
#include <TinyGPS.h>
#include "DHT.h"

SoftwareSerial ss(4, 3);//4Rx, 3Tx
SoftwareSerial licor(10, 11); //10 RX, 11 TX.
TinyGPS gps;

const byte  Obj_DHT22 = 2;   //El pin2 lee el valor del sensor DHT22
#define    DHTPIN     Obj_DHT22     // Indica el pin que se utilizara para la
lectura
#define   DHTTYPE           DHT22
```

```
DHT Obj_DHT(DHTPIN, DHTTYPE);

int periodo = 10000;
unsigned long TiempoAhora = 0;

byte rele = 7;
float hmd = 0; //Guarda el valor de la humedad
float tmp = 0; //Guarda el valor de la Temperatura

String StrDataResp = "";
boolean StringCompleta = false;
boolean bandera = false;
char VarChar, CharPrimer = ' ';
String StrDatIn = "";

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  IniciarPines();
  Obj_DHT.begin();
  while (!Serial) {
    ;
  }
  licor.begin(9600);
  ss.begin(9600);
  Serial.println("Iniciando Control ...");
}

void loop() {
  if (StringCompleta)
  {
    delay(5);

    CharPrimer = StrDatIn.charAt(0);

    StrDatIn.replace(" ", "");
    StrDatIn.replace("F", "");
    StrDatIn.replace("G", "");
    StrDatIn.replace("H", "");
    StrDatIn.replace("#", "");

    if (CharPrimer == 'G') {
      delay(500);
      if (bandera == false) {
        bandera = true;
```

```
      digitalWrite(rele, LOW);
      StrDataResp = "0,Bomba ON,0,0,0,0,0,0#\n";
      //co2,rele,lat,lon,sat,pre,temp,humd#}
    } else {
      digitalWrite(rele, LOW);
      StrDataResp = "0,Bomba ON,0,0,0,0,0,0#\n";
      //co2,rele,lat,lon,sat,pre,temp,humd#}
    }
  }

  if (CharPrimer == 'H') {
    bandera = false;
    digitalWrite(rele, HIGH);
    StrDataResp = "0,Bomba OFF,0,0,0,0,0,0#\n";
  }
  delay(5);

  if (CharPrimer == 'F') {
    bandera = true; //Para dejar de tomar la temperatura y humedad
    Serial.println("Temperatura & Humedad OFF");
    delay(1000);
  }


  if (CharPrimer == 'P')
  {
    ss.listen();
    geolocalizador();
    delay(250);
  }


  Serial.print(StrDataResp);
  while (Serial.available() > 0 ) {
    Serial.read();
  } Serial.flush();

  delay(1000);

  StrDataResp = "";
  VarChar = ' ';
  StringCompleta = false;
  StrDatIn = "";

}
```

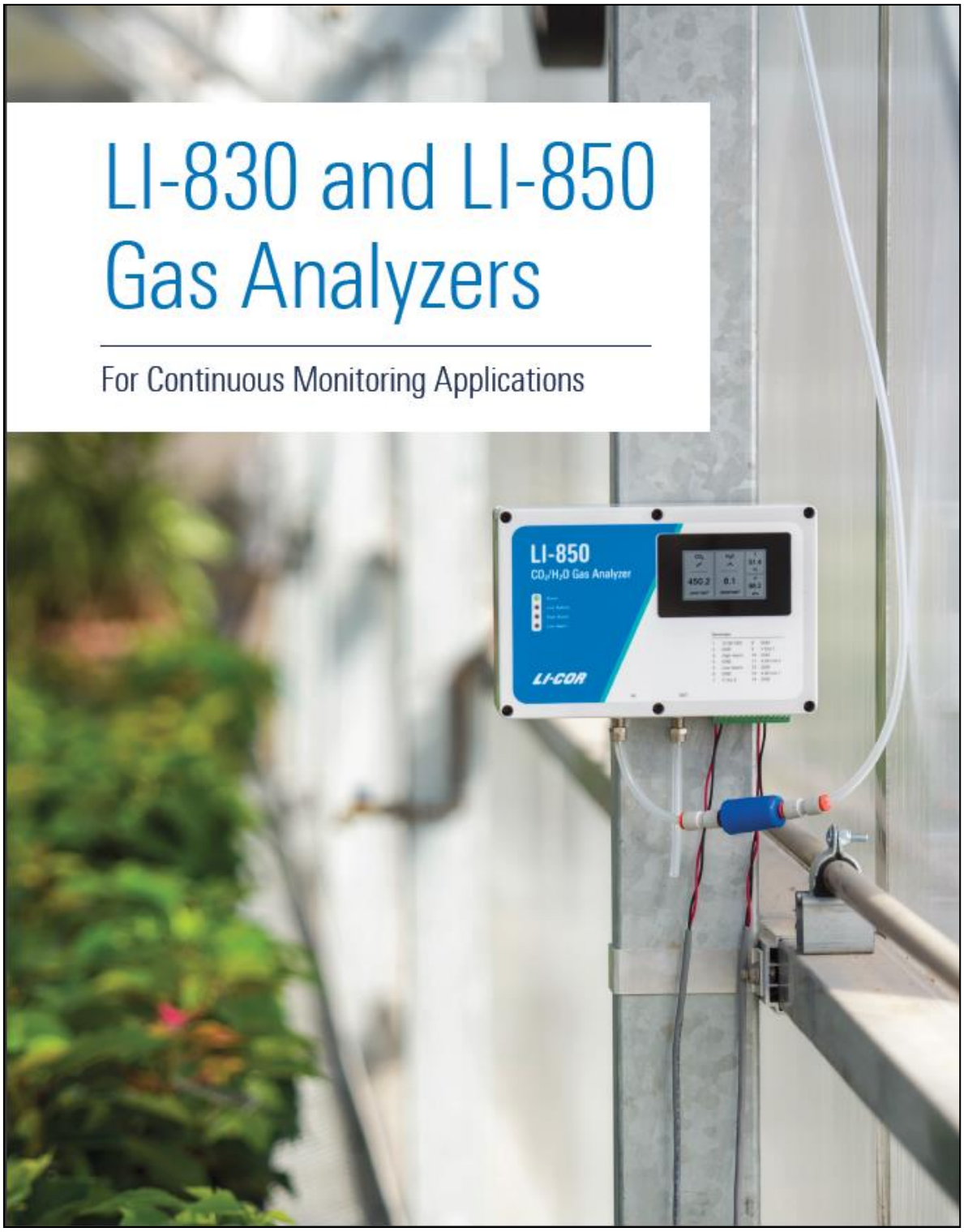
```
  if (bandera == false) {
    if (millis() > TiempoAhora + periodo) {
      TiempoAhora = millis();
      for (int i = 0; i <= 1; i++) {
        LeerHumd_Temp();
        delay(1000);
      }
    }
  }

  licor.listen();
  Licor();
}

void serialEvent() {
  while (Serial.available())
  {
    VarChar = (char)Serial.read();
    StrDatIn += VarChar;
    if (VarChar == '#') {
      StringCompleta = true;
    }
  }
}

void IniciarPines (void)
{
  pinMode(rele, OUTPUT);

  digitalWrite(rele, HIGH);
}

void Licor(void) {
  if (licor.available())
  {
    String data  = licor.readStringUntil('\n') + ",0,0,0,0,0,0,0#";
    DEBUG(data);
  }
}


void geolocalizador(void) {
  bool newData = false;
  unsigned long chars;
```

```
  unsigned short sentences, failed;

  // Intentar recibir secuencia durante un segundo
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      if (gps.encode(c)) // Nueva secuencia recibida
        newData = true;
    }
  }

  if (newData)
  {
    float flat, flon;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    Serial.print("0,0,");
    Serial.print(" LAT=");
    Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
    Serial.print(",");
    Serial.print(" LON=");
    Serial.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
    Serial.print(",");
    Serial.print(" SAT=");
    Serial.print(gps.satellites() == TinyGPS::GPS_INVALID_SATELLITES ? 0 :
gps.satellites());
    Serial.print(",");
    Serial.print(" PREC=");
    Serial.print(gps.hdop() == TinyGPS::GPS_INVALID_HDOP ? 0 : gps.hdop());
    Serial.print(",0,0#\n");
  }
  gps.stats(&chars, &sentences, &failed);
  /*Serial.print(" CHARS=");
    Serial.print(chars);
    Serial.print(" SENTENCES=");
    Serial.print(sentences);
    Serial.print(" CSUM ERR=");
    Serial.println(failed);*/
}

void LeerHumd_Temp (void)
{
  hmd = Obj_DHT.readHumidity(); //Lee la humedad
```

```
  tmp = Obj_DHT.readTemperature();//Lee la temperatura en grados centigrados
(Valor por defecto)

  // verifica si alguna lectura ha fallado
  if (isnan(hmd) || isnan(tmp)) {
    Serial.println("Existe un error en la lectura del sensor DHT22!");
    hmd = 0; tmp = 0;
  }
  Serial.print("0,0,0,0,0,0,"); Serial.print(tmp, 2); Serial.print(",");
Serial.print(hmd, 2); Serial.print("#\n");
  while (Serial.available() > 0 ) {
    Serial.read();
  } Serial.flush();
}
```

B) HOJAS DE DATOS

- *Sensor LI COR 830*



Easy Operation

The LI-830 CO_2 and LI-850 CO_2/H_2O Analyzers are high-performance monitoring solutions that give accurate, stable readings. Easy-to-use software and minimal maintenance provide hassle-free measurements for a wide range of applications and system integration options.

Just plug it in and you're ready to go

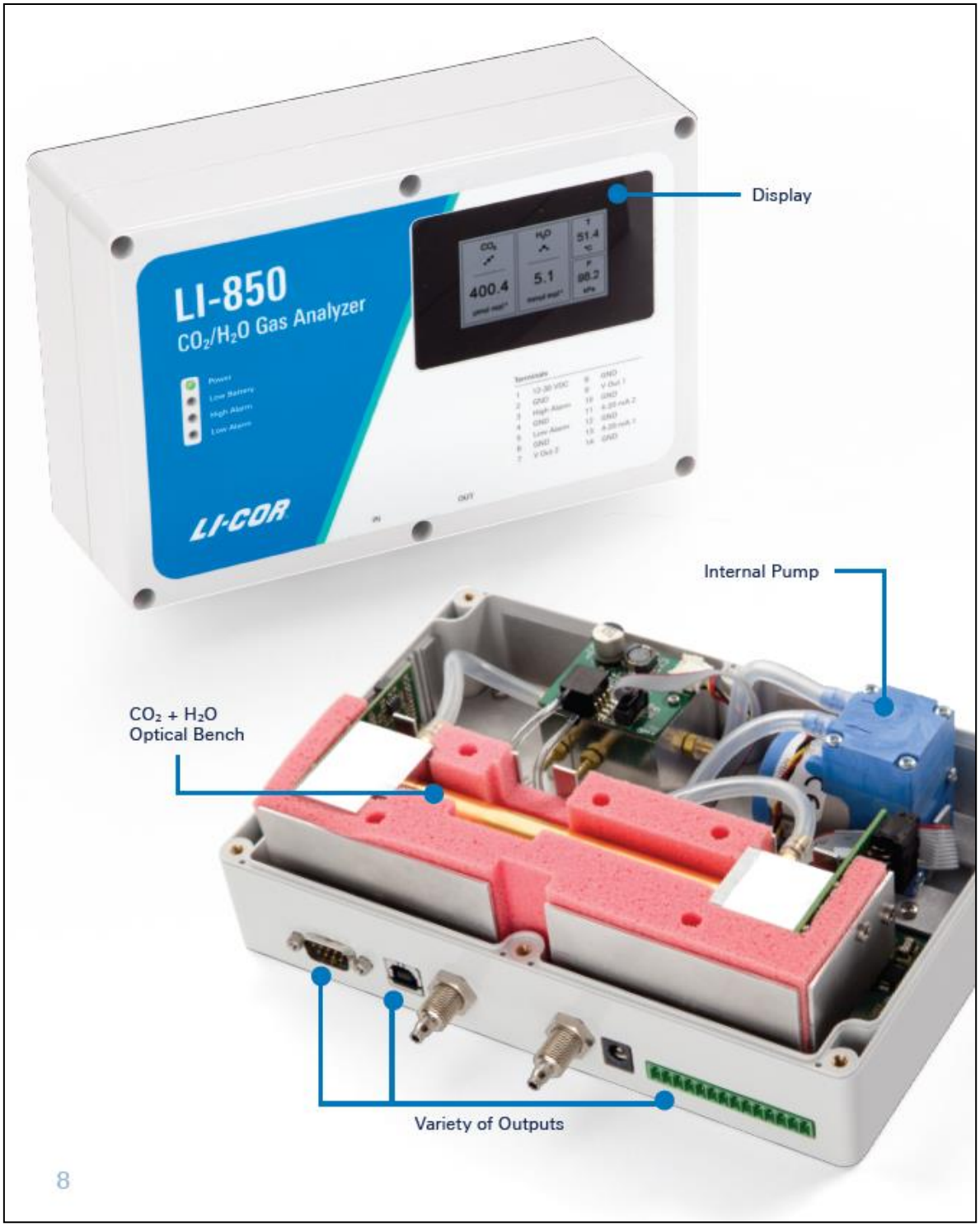
With the new optional pump and display, just power the instrument on, and measurements appear immediately. View real-time measurements, configure graphs, or set up logging options with easy-to-use software.

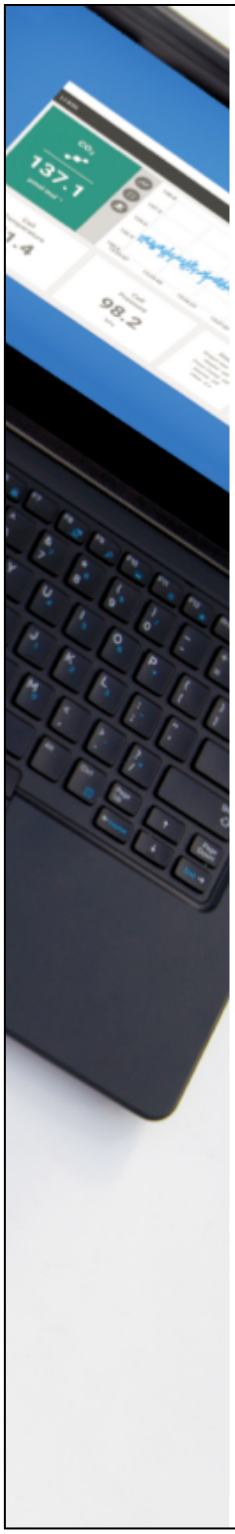
Keep it simple with minimal maintenance

Ensure continuous operation and minimize downtime with a user-cleanable optical bench, and no need for factory recalibration.

Easily analyze measurements

Windows® and Mac® interface software display real-time concentrations and graphs. Easily set up operational parameters and logging options, and view analyzer diagnostics.


LI-850
CO2/H2O Gas Analyzer
LI-COR
Display
Internal Pump
CO2 + H2O
Optical Bench
Variety of Outputs
8



Specifications

CO_2 Measurements

Measurement range: 0-20,000 ppm

Accuracy:

- **LI-850:** Within 1.5% of reading
- **LI-830:** Within 3% of reading

Calibration drift:

- **Zero drift[1]:** <0.15 ppm/°C
- **Span drift[2]:** <0.03%/°C
- **Total drift at 370 ppm[3]:** <0.4 ppm/°C

RMS noise at 370 ppm with 1 sec signal filtering: <1 ppm

Sensitivity to water vapor (LI-850 only): <0.1 ppm CO_2 / mmol mol^{-1} H_2O

Lower limit of detection: 1.5 ppm

H_2O Measurements (LI-850 only)

Measurement range: 0-60 mmol mol^{-1}

Accuracy: Better than 1.5% of reading

Calibration drift:

- **Drift at 0 mmol mol^{-1}:** <0.003 mmol mol^{-1}/°C
- **Span drift at 10 mmol mol^{-1}:** <0.03% mmol mol^{-1}/°C
- **Total drift at 10 mmol mol^{-1}:** <0.009 mmol mol^{-1}/°C

RMS noise at 10 mmol mol with 1 sec signal filtering: <0.01 mmol mol^{-1}

Sensitivity to CO_2: <0.0001 mmol mol^{-1} H_2O / ppm CO_2

Pump (optional)

Operating temperature range: 5 to 45 °C

Storage temperature range: -20 to 60 °C

Operating humidity range: 0 to 80% RH

Nominal flow rate: 0.75 liters $minute^{-1}$

Power consumption: 1 W (nominally)

Expected life span: 8,000 hrs in standard conditions with a normal load

Display (optional)

Dimensions: 6.7 cm corner-to-corner

Resolution: 400 x 200 px; monochrome

Power consumption: <200 µW

Displayed variables: CO_2 reading, H_2O reading (LI-850 only), optical bench temperature, and pressure.

Specifications subject to change without notice

General

Output rate: Up to 2 measurements per sec

Response time (T90):

- **CO_2:** <3.5 seconds from 0-375 ppm
- **H_2O:** <3.5 seconds from 0-21 mmol mol^{-1}

Measurement principle: Non-Dispersive Infrared

Traceability:

- **CO_2:** Traceable gases to WMO standards from 0-3,000 ppm; traceable gases to EPA protocol gases from 3,000-20,000 ppm
- **H_2O (LI-850 only):** NIST traceable LI-610 Portable Dew Point Generator

Pressure compensation range: 50-110 kPa

Maximum gas flow rate: 1 liter min^{-1}

Output signals: Two analog voltage (0-2.5 V or 0-5 V) and two current (4-20 mA)

Digital outputs: TTL (0-5 V) or Open Collector

DAC resolution: 16-bits across user specified range

Power requirements:

- **Input voltage:** 12-30 VDC
- **After warmup (without pump):** 0.33A @ 12 VDC (4.0 W) average
- **After warmup (with pump):** 0.42A @ 12 VDC (5.0 W) average
- **During warmup:** 1.2 A @ 12 VDC (14 W) maximum

Operating temperature range: -20 to 45 °C

Relative humidity range: 0-95% RH, Non-condensing

Dimensions: 22.23 cm W x 15.25 cm D x 7.62 cm H

Weight:

- **No pump, no display:** 1.0 kg
- **No pump, with display:** 1.02 kg
- **With pump, no display:** 1.3 kg
- **With pump, with display:** 1.32 kg

Internal optical cell volume: 14.5 mL

[1]Zero drift is the change with temperature at 0 concentration.

[2]Span drift is the residual error after re-zeroing following a temperature change.

[3]Total drift is the change with temperature without re-zeroing or re-spanning.

- *ATMega328P*

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32K Bytes of In-System Self-Programmable Flash progam memory
  - 1K Bytes EEPROM
  - 2K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltage:
  - 2.7V - 5.5V for ATmega328P
- Temperature Range:
  - Automotive Temperature Range: -40°C to +125°C
- Speed Grade:
  - 0 - 8 MHz @ 2.7 - 5.5V (Automotive Temp. Range: -40°C to +125°C)
  - 0 - 16 MHz @ 4.5 - 5.5V (Automotive Temp. Range: -40°C to +125°C)
- Low Power Consumption
  - Active Mode: 1.5mA @3V - 4MHz
  - Power-down Mode: 1 µA @3V

ATMEL®

8-bit AVR® Microcontroller with 32K Bytes In-System Programmable Flash

ATmega328P

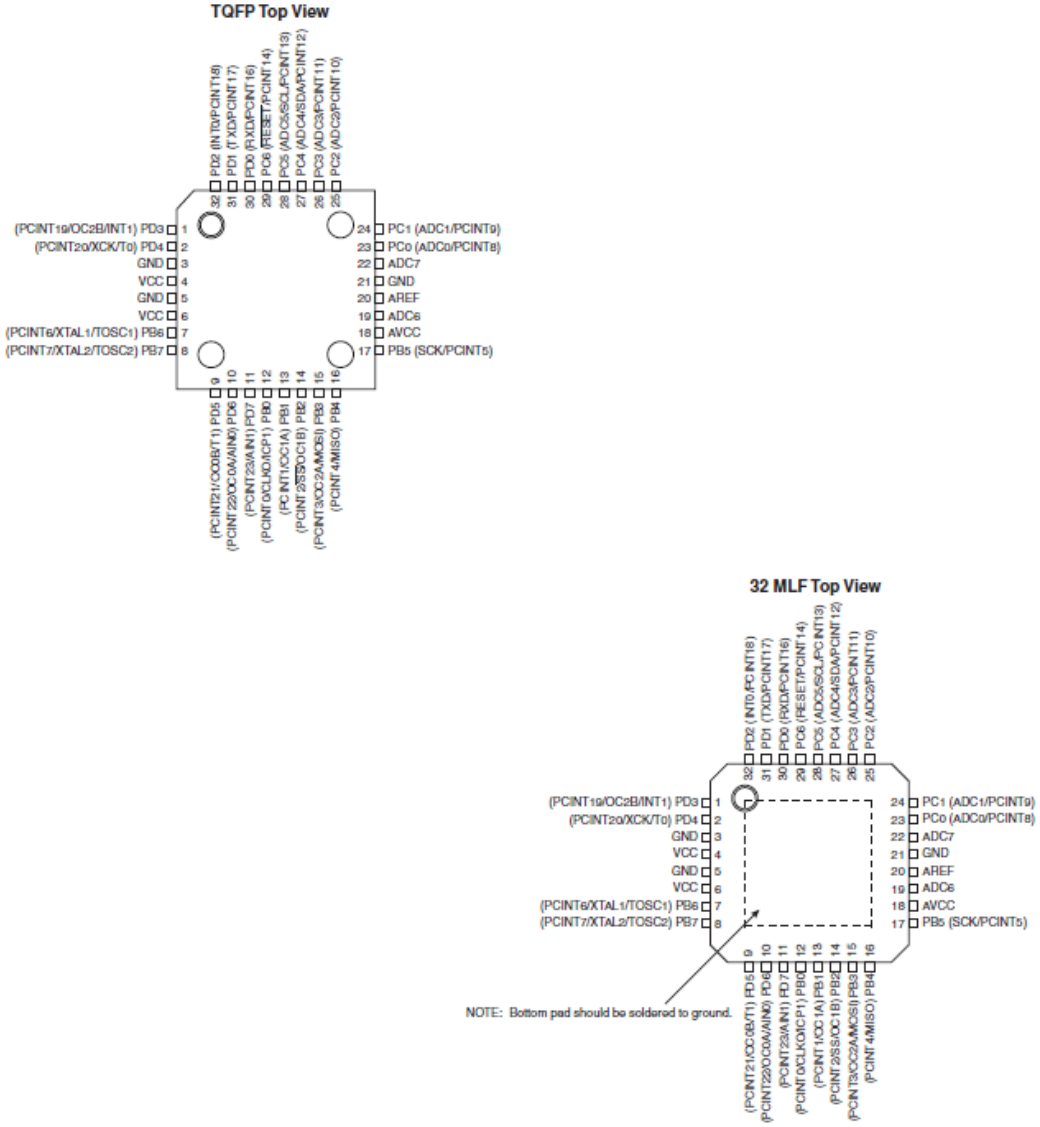
Automotive

Preliminary

7810A–AVR–11/09

1. Pin Configurations

Figure 1-1. Pinout



1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 74 and "System Clock and Clock Options" on page 25.

1.1.4 Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

1.1.5 PC6/$\overline{\text{RESET}}$

If the RSTDISBL Fuse is programmed, PC6 is used as an input pin. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-4 on page 308. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in "Alternate Functions of Port C" on page 77.

1.1.6 Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

The various special features of Port D are elaborated in "Alternate Functions of Port D" on page 80.

1.1.7 AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC6..4 use digital supply voltage, V_{CC}.

1.1.8 AREF

AREF is the analog reference pin for the A/D Converter.

1.1.9 ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

1.2 Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of actual ATmega328P AVR microcontrollers manufactured on the typical process technology. Automotive Min and Max values are based on characterization of actual ATmega328P AVR microcontrollers manufactured on the whole process excursion (corner run).

1.3 Automotive Quality Grade

The ATmega328P have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (Temperature and Voltage). The quality and reliability of the ATmega328P have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the products are available in only one temperature grade.

Table 1-1. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
-40°C ; +125°C	Z	Full AutomotiveTemperature Range

28. Electrical Characteristics

All DC/AC characteristics contained in this datasheet are based on characterization of ATmega328P AVR microcontroller manufactured in an automotive process technology.

28.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA
Injection Current at V_{CC} = 0V	±5.0mA[1]
Injection Current at V_{CC} = 5V	±1.0mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note: 1. Maximum current per port = ±30mA

28.2 DC Characteristics.

T_A = -40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.3V_{CC}$[1]	V
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 2.7V - 5.5V	$0.6V_{CC}$[2]		V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.1V_{CC}$[1]	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 2.7V - 5.5V	$0.7V_{CC}$[2]		V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.1V_{CC}$[1]	V
V_{IH2}	Input High Voltage, $\overline{\text{RESET}}$ pin	V_{CC} = 2.7V - 5.5V	$0.9V_{CC}$[2]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[3]	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 5 mA, V_{CC} = 3V			0.8 0.5	V
V_{OH}	Output High Voltage[4]	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.1 2.3			V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor		20		50	kΩ
V_{ACIO}	Analog Comparator Input Offset Voltage	$0.4V<V_{in}<V_{Vcc-0.5}$ (absolute value)		10	40	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = $V_{CC}/2$	-50		50	nA

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega328P:
   1] The sum of all I_{OL}, for ports C0 - C5, should not exceed 100 mA.
   2] The sum of all I_{OL}, for ports B0 - B5, D5 - D7, XTAL1, XTAL2 should not exceed 100 mA.
   3] The sum of all I_{OL}, for ports D0 - D4, should not exceed 100 mA.
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega328P:
   1] The sum of all I_{OH}, for ports C0 - C5, D0- D4, should not exceed 150 mA.
   2] The sum of all I_{OH}, for ports B0 - B5, D5 - D7, XTAL1, XTAL2 should not exceed 150 mA.
   If II_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

28.3 DC Characteristics

T_A = -40°C to 125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.[2]	Max.	Units
I_{CC}	Power Supply Current[1]	Active 4 MHz, V_{CC} = 3V		1.5	2.4	mA
		Active 8 MHz, V_{CC} = 5V		5.2	10	mA
		Active 16 MHz, V_{CC} = 5V		9.2	14	mA
		Idle 4 MHz, V_{CC} = 3V		0.25	0.6	mA
		Idle 8 MHz, V_{CC} = 5V		1.0	1.6	mA
		Idle 16 MHz, V_{CC} = 5V		1.9	2.8	mA
	Power-down mode[3]	WDT enabled, V_{CC} = 3V			44	µA
		WDT enabled, V_{CC} = 5V			66	µA
		WDT disabled, V_{CC} = 3V			40	µA
		WDT disabled, V_{CC} = 5V			60	µA

Notes:
1. Values with "Minimizing Power Consumption" enabled (0xFF).
2. Typical values at 25°C.
3. The current consumption values include input leakage current.

- *HC06 Bluetooth*

2. Feature

- Wireless transceiver
  - Sensitivity (Bit error rate) can reach -80dBm.
  - The change range of output's power: -4 - +6dBm.
- Function description (perfect Bluetooth solution)
  - Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
  - Has a build-in 2.4GHz antenna; user needn't test antenna.
  - Has the external 8Mbit FLASH
  - Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30~40mA. The current in communication is 8mA.
  - Standard HCI Port (UART or USB)
  - USB Protocol: Full Speed USB1.1, Compliant With 2.0
  - This module can be used in the SMD.
  - It's made through RoHS process.
  - The board PIN is half hole size.
  - Has a 2.4GHz digital wireless transceiver.
  - Bases at CSR BC04 Bluetooth technology.
  - Has the function of adaptive frequency hopping.
  - Small (27mm×13mm×2mm)
  - Peripherals circuit is simple.
  - It's at the Bluetooth class 2 power level.
  - Storage temperature range: -40 ℃ - 85℃， work temperature range: -25 ℃ - +75℃
  - Any wave inter Interference: 2.4MHz， the power of emitting: 3 dBm.
  - Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.
- Low power consumption
- Has high-performance wireless transceiver system
- Low Cost

- Application fields:
  - Bluetooth Car Handsfree Device
  - Bluetooth GPS
  - Bluetooth PCMCIA , USB Dongle
  - Bluetooth Data Transfer
- Software
  - CSR

3. PINs description

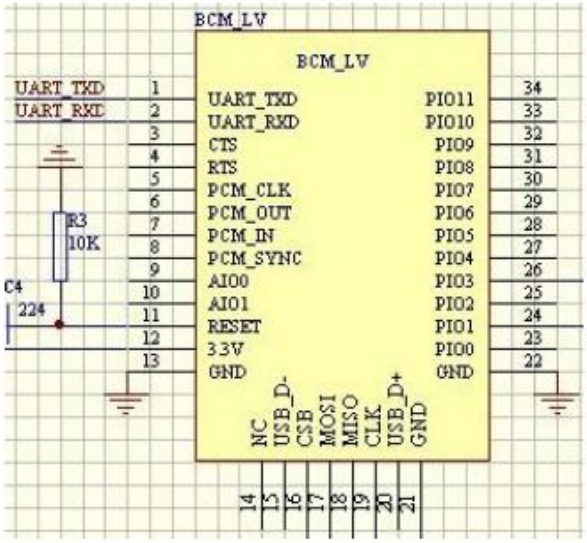


Figure 3 PIN configuration

The PINs at this block diagram is as same as the physical one.

PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
1V8	14	VDD	Integrated 1.8V (+) supply with On-chip linear regulator output within 1.7-1.9V	
VCC	12	3.3V		
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	

- *NEO 6M GPS*

GPS

NEO-6

u-blox 6 GPS Modules

Data Sheet

locate, communicate, accelerate

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 2.4 mm

1.3 GPS performance

Parameter	Specification			
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS			
Time-To-First-Fix[1]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Cold Start[2]	26 s	27 s	32 s
	Warm Start[2]	26 s	27 s	32 s
	Hot Start[2]	1 s	1 s	1 s
	Aided Starts[3]	1 s	<3 s	<3 s
Sensitivity[4]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Tracking & Navigation	-162 dBm	-161 dBm	-160 dBm
	Reacquisition[5]	-160 dBm	-160 dBm	-160 dBm
	Cold Start (without aiding)	-148 dBm	-147 dBm	-146 dBm
	Hot Start	-157 dBm	-156 dBm	-155 dBm
Maximum Navigation update rate		NEO-6G/Q/M/T	NEO-6P/V	
		5Hz	1 Hz	
Horizontal position accuracy[6]	GPS	2.5 m		
	SBAS	2.0 m		
	SBAS + PPP[7]	< 1 m (2D, R50)[8]		
	SBAS + PPP[7]	< 2 m (3D, R50)[8]		
Configurable Timepulse frequency range		NEO-6G/Q/M/P/V	NEO-6T	
		0.25 Hz to 1 kHz	0.25 Hz to 10 MHz	
Accuracy for Timepulse signal	RMS	30 ns		
	99%	<60 ns		
	Granularity	21 ns		
	Compensated[9]	15 ns		
Velocity accuracy[6]		0.1m/s		
Heading accuracy[6]		0.5 degrees		
Operational Limits	Dynamics	≤ 4 g		
	Altitude[10]	50,000 m		
	Velocity[10]	500 m/s		

Table 2: NEO-6 GPS performance

3 Electrical specifications

3.1 Absolute maximum ratings

Parameter	Symbol	Module	Min	Max	Units	Condition
Power supply voltage	VCC	NEO-6G	-0.5	2.0	V	
		NEO-6Q, 6M, 6P, 6V, 6T	-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	NEO-6Q, 6M, 6G, 6V, 6T		15	dBm	source impedance = 50Ω, continuous wave
		NEO-6P		-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 6.4.

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/MAX-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/M NEO-6P/V/T	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.

3.3 Indicative power requirements

Table 11 lists examples of the total system supply current for a possible application.

Parameter	Symbol	Module	Min	Typ	Max	Units	Condition
Max. supply current [15]	Iccp	All			67	mA	VCC = 3.6 V[16] / 1.95 V[17]
Average supply current[18]	Icc Acquisition	All		47[19]		mA	VCC = 3.0 V[16] / 1.8 V[17]
	Icc Tracking (Max Performance mode)	NEO-6G/Q/T		40[20]		mA	
		NEO-6M/P/V		39[20]		mA	
	Icc Tracking (Eco mode)	NEO-6G/Q/T		38[20]		mA	
		NEO-6M/P/V		37[20]		mA	
	Icc Tracking (Power Save mode / 1 Hz)	NEO-6G/Q		12[20]		mA	
		NEO-6M		11[20]		mA	

Table 11: Indicative power requirements

☞ Values in Table 11 are provided for customer information only as an example of typical power requirements. Values are characterized on samples, actual power requirements can vary depending on FW version used, external circuitry, number of SVs tracked, signal strength, type of start as well as time, duration and conditions of test.

---

[15] Use this figure to dimension maximum current capability of power supply. Measurement of this parameter with 1 Hz bandwidth.
[16] NEO-6Q, NEO-6M, NEO-6P, NEO-6V, NEO-6T
[17] NEO-6G
[18] Use this figure to determine required battery capacity.
[19] >8 SVs in view, CNo >40 dBHz, current average of 30 sec after cold start.
[20] With strong signals, all orbits available. For Cold Starts typical 12 min after first fix. For Hot Starts typical 15 s after first fix.

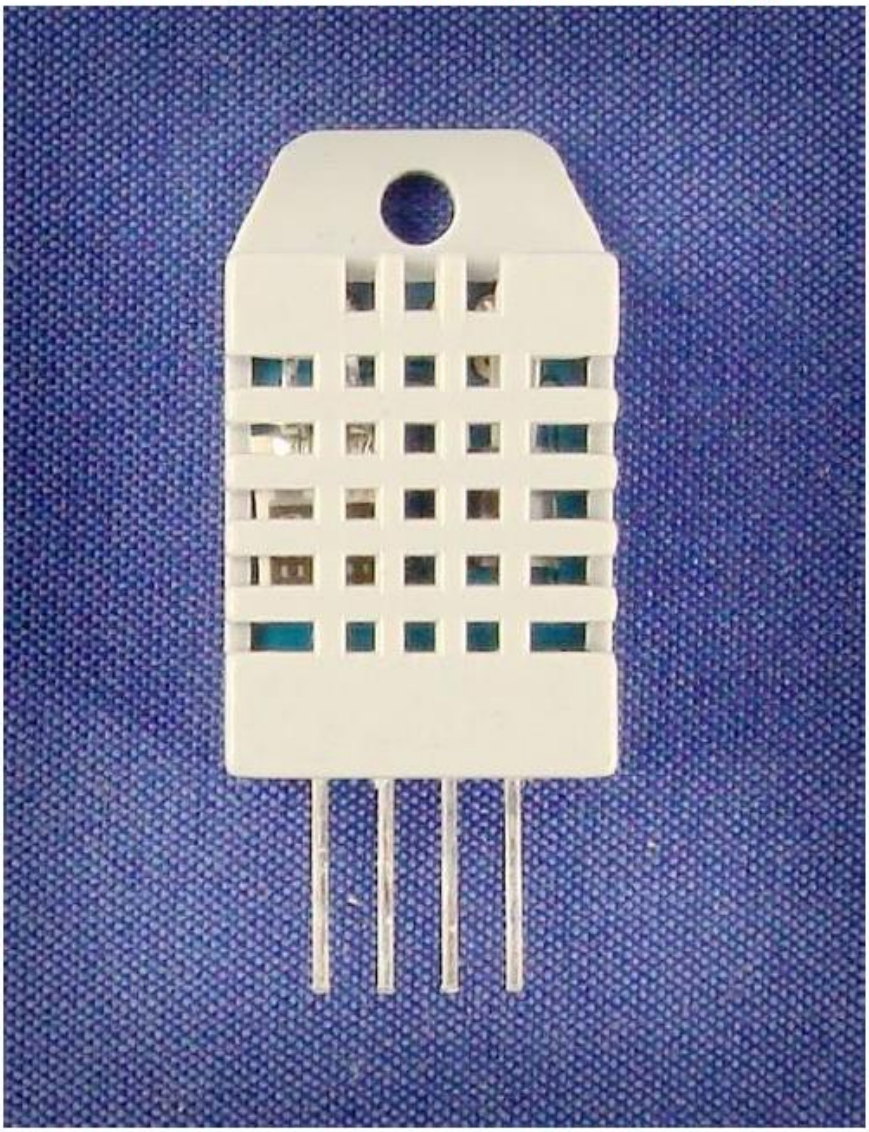
- *DHT22 Sensor de temperatura y humedad*

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

Digital-output relative humidity & temperature sensor/module

DHT22 (DHT22 also named as AM2302)



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.

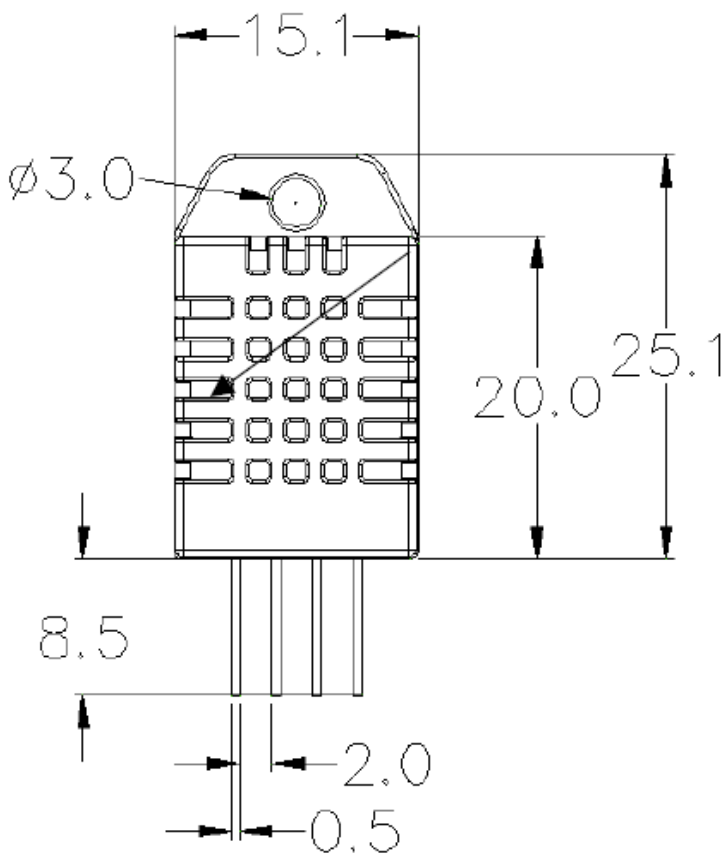
Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

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



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

7. Electrical Characteristics:

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

*Collecting period should be : >2 second.

- *LM2596 Convertidor DC-DC*

TEXAS INSTRUMENTS

LM2596

www.ti.com

SNVS124C – NOVEMBER 1999 – REVISED APRIL 2013

LM2596 SIMPLE SWITCHER® Power Converter 150 kHz 3A Step-Down Voltage Regulator

Check for Samples: LM2596

FEATURES

- 3.3V, 5V, 12V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range, 1.2V to 37V ±4% Max Over Line and Load Conditions
- Available in TO-220 and TO-263 Packages
- Ensured 3A Output Load Current
- Input Voltage Range Up to 40V
- Requires Only 4 External Components
- Excellent Line and Load Regulation Specifications
- 150 kHz Fixed Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, I_Q Typically 80 µA
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current Limit Protection

APPLICATIONS

- Simple High-Efficiency Step-Down (Buck) Regulator
- On-Card Switching Regulators
- Positive to Negative Converter

DESCRIPTION

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation , and a fixed-frequency oscillator.

The LM2596 series operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Available in a standard 5-lead TO-220 package with several different lead bend options, and a 5-lead TO-263 surface mount package.

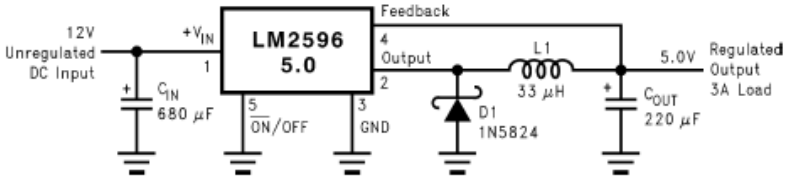
A standard series of inductors are available from several different manufacturers optimized for use with the LM2596 series. This feature greatly simplifies the design of switch-mode power supplies.

Other features include a ensured ±4% tolerance on output voltage under specified input voltage and output load conditions, and ±15% on the oscillator frequency. External shutdown is included, featuring typically 80 µA standby current. Self protection features include a two stage frequency reducing current limit for the output switch and an over temperature shutdown for complete protection under fault conditions. (1)

(1) † Patent Number 5,382,918.

Typical Application

(Fixed Output Voltage Versions)



Connection Diagrams

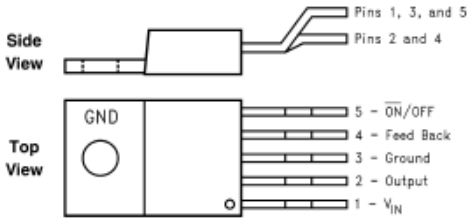


Figure 1. 5-Lead Bent and Staggered Leads, Through Hole TO-220 (T) Package
See Package Number NDH0005D

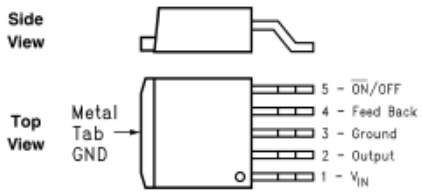


Figure 2. 5-Lead DDPAK/TO-263 (S) Package
See Package Number KTT0005B

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

Absolute Maximum Ratings (1)(2)

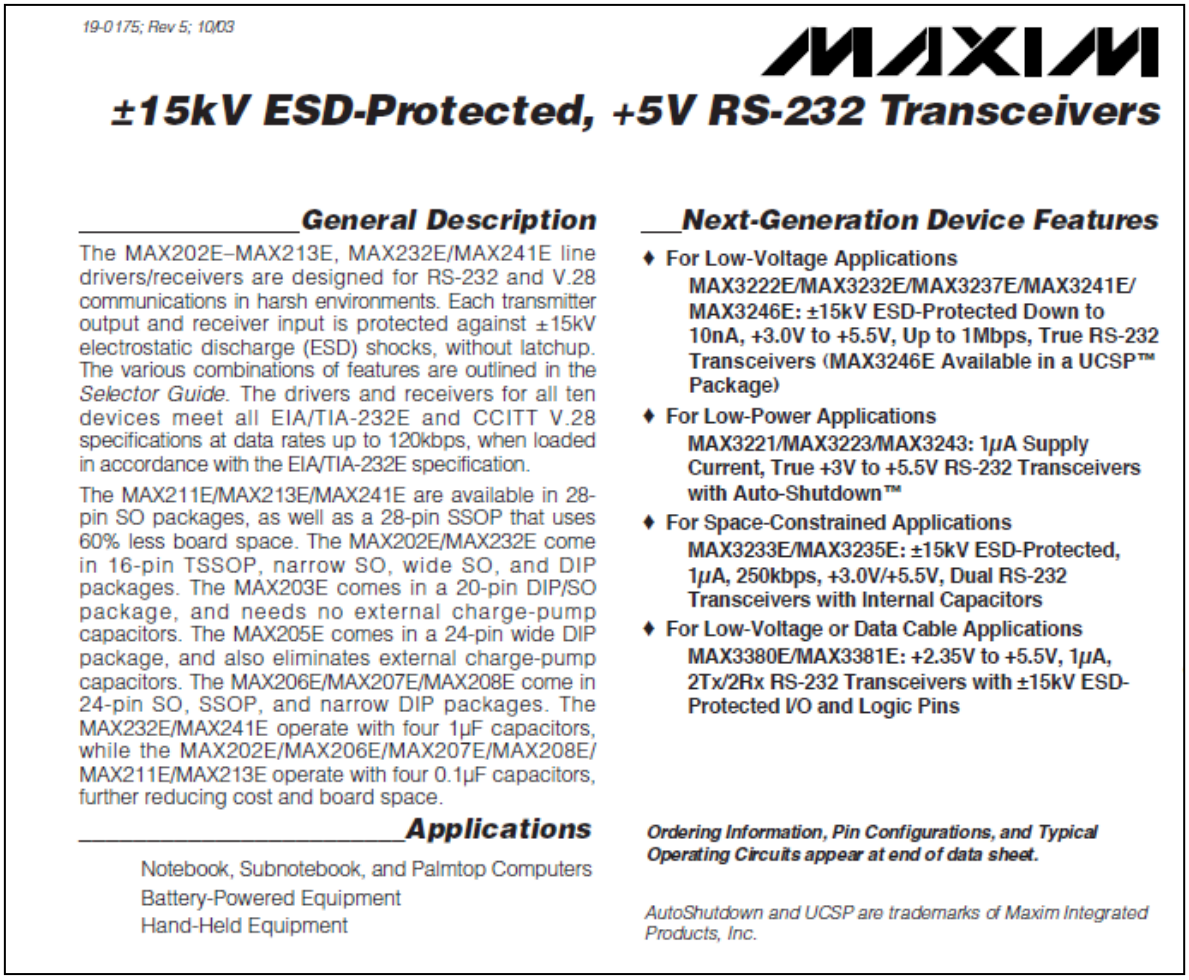
Maximum Supply Voltage	45V
$\overline{ON}$ /OFF Pin Input Voltage	−0.3 ≤ V ≤ +25V
Feedback Pin Voltage	−0.3 ≤ V ≤+25V
Output Voltage to Ground (Steady State)	−1V
Power Dissipation	Internally limited
Storage Temperature Range	−65°C to +150°C
ESD Susceptibility	
Human Body Model (3)	2 kV
Lead Temperature	
DDPAK/TO-263 Package	
Vapor Phase (60 sec.)	+215°C
Infrared (10 sec.)	+245°C
TO-220 Package (Soldering, 10 sec.)	+260°C
Maximum Junction Temperature	+150°C

(1) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is intended to be functional, but do not ensure specific performance limits. For ensured specifications and test conditions, see the Electrical Characteristics.
(2) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(3) The human body model is a 100 pF capacitor discharged through a 1.5k resistor into each pin.

Operating Conditions

Temperature Range	−40°C ≤ T_J ≤ +125°C
Supply Voltage	4.5V to 40V

- *MAX232 Convertidor RS232 a TTL*



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.

AutoShutdown and UCSP are trademarks of Maxim Integrated Products, Inc.

ABSOLUTE MAXIMUM RATINGS

V_{CC} ..... -0.3V to +6V
V+ ..... (V_{CC} - 0.3V) to +14V
V- ..... -14V to +0.3V
Input Voltages
T_IN ..... -0.3V to (V+ + 0.3V)
R_IN ..... ±30V
Output Voltages
T_OUT ..... (V- - 0.3V) to (V+ + 0.3V)
R_OUT ..... -0.3V to (V_{CC} + 0.3V)
Short-Circuit Duration, T_OUT ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) ... 842mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ... 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) ... 762mW
16-Pin TSSOP (derate 9.4mW/°C above +70°C) ... 755mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ... 889mW
20-Pin SO (derate 10.00mW/°C above +70°C) ... 800mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) ... 1.07W
24-Pin Wide Plastic DIP
(derate 14.29mW/°C above +70°C) ... 1.14W
24-Pin SO (derate 11.76mW/°C above +70°C) ... 941mW
24-Pin SSOP (derate 8.00mW/°C above +70°C) ... 640mW
28-Pin SO (derate 12.50mW/°C above +70°C) ... 1W
28-Pin SSOP (derate 9.52mW/°C above +70°C) ... 762mW
Operating Temperature Ranges
MAX2_ _EC_ _ ..... 0°C to +70°C
MAX2_ _EE_ _ ..... -40°C to +85°C
Storage Temperature Range ..... -65°C to +165°C
Lead Temperature (soldering, 10sec) ..... +300°C

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 absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V_{CC} = +5V ±10% for MAX202E/206E/208E/211E/213E/232E/241E; V_{CC} = +5V ±5% for MAX203E/205E/207E; C1–C4 = 0.1µF for MAX202E/206E/207E/208E/211E/213E; C1–C4 = 1µF for MAX232E/241E; T_A = T_{MIN} to T_{MAX}; unless otherwise noted. Typical values are at T_A = +25°C.)

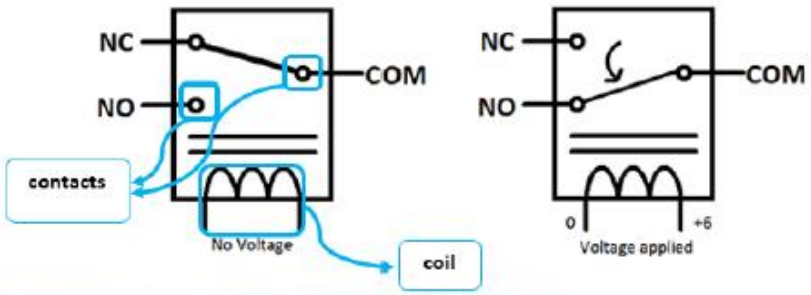
PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
DC CHARACTERISTICS							
V_{CC} Supply Current	I_{CC}	No load, T_A = +25°C	MAX202E/203E		8	15	mA
			MAX205E–208E		11	20	
			MAX211E/213E		14	20	
			MAX232E		5	10	
			MAX241E		7	15	
Shutdown Supply Current		T_A = +25°C, Figure 1	MAX205E/206E		1	10	µA
			MAX211E/241E		1	10	
			MAX213E		15	50	
LOGIC							
Input Pull-Up Current		T_IN = 0V (MAX205E–208E/211E/213E/241E)			15	200	µA
Input Leakage Current		T_IN = 0V to V_{CC} (MAX202E/203E/232E)				±10	µA
Input Threshold Low	V_{IL}	T_IN; EN, $\overline{SHDN}$ (MAX213E) or $\overline{EN}$, SHDN (MAX205E–208E/211E/241E)				0.8	V
Input Threshold High	V_{IH}	T_IN		2.0			V
		EN, $\overline{SHDN}$ (MAX213E) or $\overline{EN}$, SHDN (MAX205E–208E/211E/241E)		2.4			
Output Voltage Low	V_{OL}	R_OUT; I_{OUT} = 3.2mA (MAX202E/203E/232E) or I_{OUT} = 1.6mA (MAX205E/208E/211E/213E/241E)				0.4	V
Output Voltage High	V_{OH}	R_OUT; I_{OUT} = -1.0mA		3.5	V_{CC} - 0.4		V
Output Leakage Current		$\overline{EN}$ = V_{CC}, EN = 0V, 0V ≤ R_{OUT} ≤ V_{CC}, MAX205E–208E/211E/213E/241E outputs disabled			±0.05	±10	µA

- *Relé 5V 1 Canal*

RELAY MODULES

RELAY WORKING IDEA

Relays consist of three pins normaly open pin , normaly closed pin, common pin and coil. When coil powerd on magntic field is generated the contacts connected to each other.

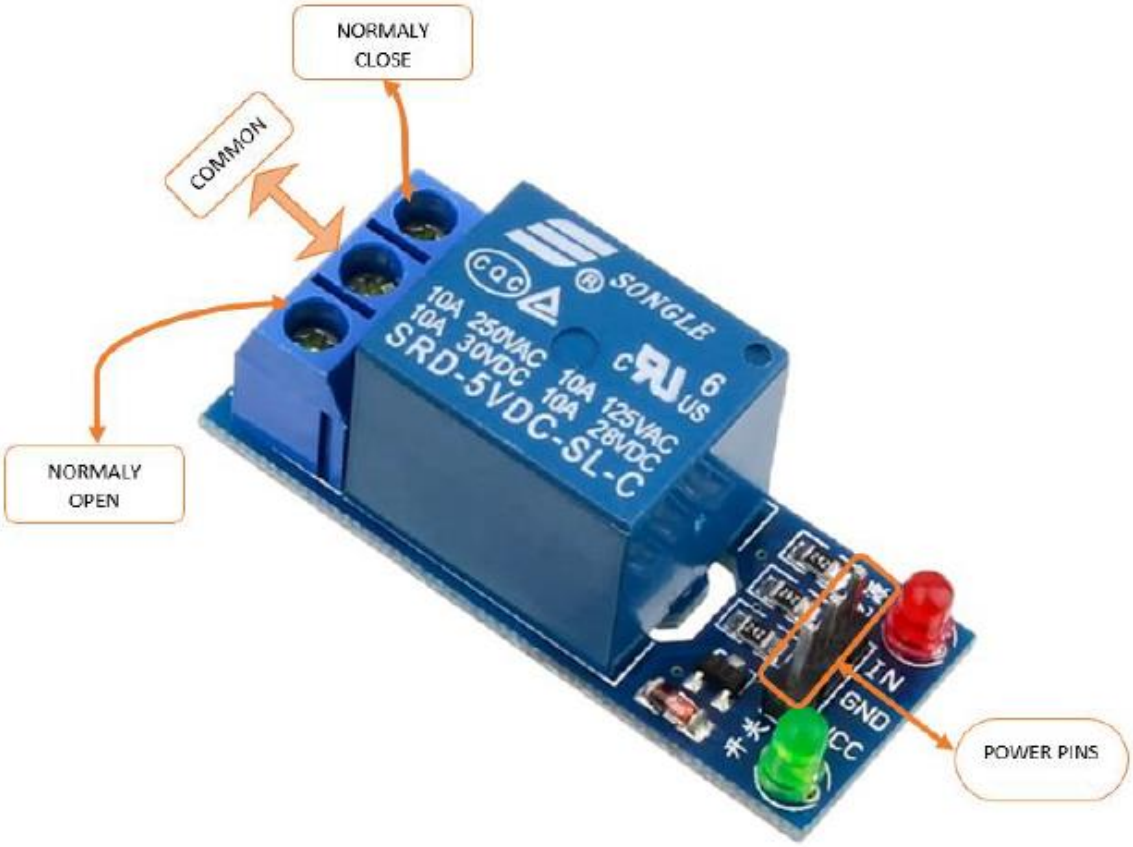


Relay modules 1-channel features

- Contact current 10A and 250V AC or 30V DC.
- Each channel has indication LED.
- Coil voltage 12V per channel.
- Kit operating voltage 5-12 V
- Input signal 3-5 V for each channel.
- Three pins for normally open and closed for each channel.

How to connect relay module with Arduino

As shown in relay working idea it depends on magnetic field generated from the coil so there is power isolation between the coil and the switching pins so coils can be easily powered from Arduino by connecting VCC and GND bins from Arduino kit to the relay module kit after that we choose Arduino output pins depending on the number of relays needed in project designed and set these pins to output and make it out high (5 V) to control the coil that allow controlling of switching process.



NOTE : whatever was the relay channels number the pinconfiguration is the same for every channel except the power pins (VCC and GND) are for the board itself. The input signal (IN) pin for every relay.