Instituto Politécnico Nacional

Escuela Superior de Ingeniería
Mecánica y Eléctrica
Unidad Profesional "Adolfo López Mateos" Zacatenco

"SISTEMA DE MONITOREO MÓVIL DE CALIDAD DEL AIRE CON RASPBERRY PI Y DISPOSITIVOS MÓVILES"

TESIS

PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTAN:

EUGENIO POPOCA JUAN MANUEL
SANCHEZ SANDOVAL JESUS JAIR
SANTILLAN GERVACIO ALEJANDRO

ASESORES:

DR. JUAN PABLO FRANCISCO POSADAS DURÁN
ING. FELIPE FEDERICO DURÁN

CIUDAD DE MÉXICO, JUNIO 2019

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	***TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL***
DEBERA (N) DESARROLLAR	**C. JUAN MANUEL EUGENIO POPOCA** **C. JESUS JAIR SANCHEZ SANDOVAL** **C. ALEJANDRO SANTILLAN GERVACIO**

"SISTEMA DE MONITOREO MÓVIL DE CALIDAD DEL AIRE CON RASPBERRY PI Y DISPOSITIVOS MÓVILES"

DESARROLLAR UN SISTEMA PARA LA MEDICIÓN DE GASES TÓXICOS EN EL AIRE (PARTICULARMENTE LOS GASES CO, NO_2, O_3 Y SO_2) Y LA ADMINISTRACIÓN DE LAS MEDICIONES MEDIANTE UNA APLICACIÓN PARA TELÉFONOS INTELIGENTES Y UN SITIO WEB.

- **INTRODUCCIÓN**
- **ESTADO DEL ARTE**
- **MARCO TEÓRICO**
- **DESARROLLO**
- **ANÁLISIS DE RESULTADOS Y PRUEBAS**

CIUDAD DE MÉXICO, A 14 DE JUNIO DEL 2019.

A S E S O R E S

DR. JUAN PABLO FRANCISCO POSADAS DURÁN

ING. FEDERICO FELIPE DURAN

M. EN C. RABINDRANATH RESÉNDIZ GUZMÁN
SUBDIRECTOR ACADÉMICO

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA
ESTADOS UNIDOS MEXICANOS
SUBDIRECCION ACADEMICA

Autorización de uso de obra

Instituto Politécnico Nacional

Presente

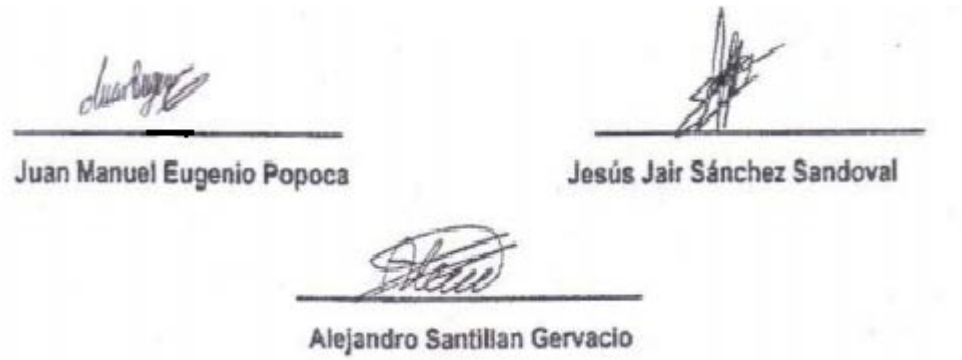
Bajo protesta de decir la verdad los que suscriben **Juan Manuel Eugenio Popoca, Jesus Jair Sanchez Sandoval y Alejandro Santillan Gervacio,** manifestamos ser los autores y titulares de los derechos morales y patrimoniales de la obra titulada **“SISTEMA DE MONITOREO MÒVIL DE CALIDAD DEL AIRE CON RASPBERRY PI Y DISPOSITIVOS MÒVILES”,** en adelante **“La Tesis”** y de la cual se adjunta una copia, en un impreso y un cd por lo que por medio del presente y con fundamento en el artículo 27 fracciòn II, inciso b) de la Ley Federal del Derecho de Autor otorgamos al **INSTITUTO POLITÈCNICO NACIONAL**, en adelante el **EL IPN,** autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales o en cualquier otro medio; **para apoyar futuros trabajos relacionados con el tema de “La Tesis”** por un periodo de **2 años** contando a partir de la fecha dela presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a **EL IPN** de su terminación.

En virtud de lo anterior, **EL IPN** deberá reconocer en todo momento nuestra calidad de autores de **“La Tesis”,**

Adicionalmente, y en nuestra calidad de autores y titulares de los derechos morales y patrimoniales de **“La Tesis”,** manifestamos que la misma es original y que la presente autorización no contraviene ninguna otorgada por los suscritos de **“La Tesis”,** por lo que deslindamos de toda responsabilidad a **EL IPN** en caso de que el contenido de **“La Tesis”** o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumimos las consecuencias legales y económicas de cualquier demanda o reclamación que pueda derivarse del caso.

Ciudad de México, a 11 de octubre de 2019

Atentamente



Juan Manuel Eugenio Popoca

Jesús Jair Sánchez Sandoval

Alejandro Santillan Gervacio

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La humanidad ha creado diversas máquinas y desarrollado distintos tipos de procesos, los cuales en su mayoría requieren de energía para funcionar. La energía puede provenir de diversas fuentes como lo son las celdas solares, generadores eólicos, plantas hidroeléctricas, plantas térmicas, plantas geotérmicas, combustibles nucleares, combustibles fósiles, entre otros. Las máquinas que requieren de combustibles fósiles liberan una gran cantidad de gases y elementos tóxicos que afectan a la salud. La contaminación generada por los combustibles fósiles también provoca, por ejemplo, cambios inesperados en el clima y alteraciones en los ecosistemas.

A nivel mundial se han creado programas que ayuden a controlar la cantidad de gases y elementos tóxicos del ambiente. Una estrategia para el control de gases consiste en el uso de equipos dedicados a la detección y medición de gases tóxicos, ubicados en puntos específicos de un área geográfica con especial interés en zonas muy transitadas y con gran densidad de población para prevenir contingencias.

Por otro lado, con el avance tecnológico que sucede día con día y el desarrollo de lo que se conoce como Ciudades Inteligentes, es importante mantener un control y medición sobre aquello que pueda afectar el entorno. Es por esto que se busca mejorar los sistemas existentes y desarrollar nuevos sistemas que cumplan con el concepto de Ciudades Inteligentes de manera que le permitan a estos entornos volverse sustentables y mejor diseñados, para que disminuyan el impacto ambiental que podrían generar.

En México se cuenta con un sistema de monitoreo de la calidad del aire que cubre la Ciudad de México y parte del Estado de México, sin embargo, las estaciones de monitoreo son fijas y no alcanzan a cubrir por completo el área designada. Además estos equipos son de costos elevados por lo que en este proyecto se presenta un prototipo móvil de costo menor en comparación con las estaciones fijas, que mide la concentración 4 gases tóxicos: Monóxido de Carbono (CO), Ozono (O_3), Dióxido de azufre (SO_2) y Dióxido de Nitrógeno (NO_2). El prototipo cuenta con un banco de baterías, que le

brinda una autonomía de 4,6 horas promedio de operación permitiéndole operar de forma independiente.

1.1. Planteamiento del problema

La contaminación del aire es una problemática mundial que afecta a todos los sectores de la sociedad principalmente a niños, adultos mayores y personas con enfermedades crónicas y respiratorias. En México se creó el Sistema de Monitoreo Ambiental (SIMAT) en la Ciudad de México (CDMX) y se estableció el índice IMECA (determina la calidad del aire) como medidas para disminuir el impacto de la contaminación del aire.

La medición de los contaminantes asociados al índice IMECA se lleva a cabo en estaciones fijas, las cuales se localizan en puntos específicos del Valle de México. Las estaciones del SIMAT realizan mediciones de los contaminantes en distintos horarios y presentan sus resultados en la página Web del SIMAT[1]. Por otro lado, las estaciones fijas no cubren todo el área designada, por lo que las mediciones realizadas no registran los niveles de contaminación reales en todo el Valle de México. Es por esto que se buscan soluciones alternativas por ejemplo: se puede incrementar la cantidad de estaciones fijas de monitoreo, o implementar módulos móviles que lleven acabo las mediciones en diferentes puntos de la región, complementando así los datos recabados por las estaciones fijas.

1.2. Justificación

A pesar de los intentos del Gobierno de la Ciudad de México por monitorear los niveles de contaminación del aire del Valle de México, existen zonas que no se encuentran dentro del rango de las estaciones fijas de monitoreo por lo que no se puede brindar información más precisa sobre sus índices de contaminación. Tal es el caso de la unidad profesional Adolfo López Mateos del IPN la cual no cuenta con ninguna unidad de monitoreo de la calidad del aire lo que representa un riesgo para la comunidad estudiantil especialmente para aquellos que toman parte en las actividades deportivas que ofrece el Instituto, o que realizan sus prácticas de laboratorio en exteriores.

Se propone realizar un sistema compuesto por un arreglo de sensores, los sensores seleccionados cumplen con las normas de la Agencia de Protección Ambiental de Estados Unidos (EPA), para proyectos estudiantiles y pueden servir como referencia para el SIMAT, que midan el nivel de aquellos contaminantes relacionados con el índice IMECA presentes en el aire para poder realizar mediciones de gases tóxicos en los lugares alejados de las estaciones de monitoreo fijas. Mediante una aplicación para teléfono

[1]Véase `www.aire.cdmx.gob.mx`

inteligentes se visualizan los datos obtenidos durante la medición y se envían a una base de datos para su almacenamiento. Finalmente una página Web muestra a la población la información recabada en un mapa de la zona con identificadores de los puntos en donde se realizó la medición y los valores obtenidos de cada sensor.

El desarrollo de este sistema de medición móvil permitirá a los estudiantes de la Unidad Profesional Adolfo López Mateos estar informados acerca de la calidad del aire dentro de la unidad. Con esta información la comunidad podrá tomar las precauciones necesarias.

El sistema podrá llevar un registro acerca de los niveles de gases tóxicos en el aire para poder obtener un promedio de las zonas que presenten una mayor cantidad de contaminantes de la unidad académica y los horarios en que se presentan esto niveles máximos. El sistema podrá ser usado fuera de la unidad académica para obtener mediciones de los niveles de gases tóxicos en el aire en diversas zonas mediante la definición de rutas.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un sistema para la medición de gases tóxicos en el aire (particularmente los gases CO, NO_2, O_3 y SO_2) y la administración de las mediciones mediante una aplicación para teléfonos inteligentes y un sitio Web.

1.3.2. Objetivos particulares

- Investigar sensores para la medición de concentración de gases tóxicos.
- Caracterizar el comportamiento de los sensores de gases tóxicos.
- Implementar un circuito atenuador para la protección del convertidor analógico digital (ADC) frente a voltajes que lo pueden dañar.
- Desarrollar un programa para un sistema de procesamiento de datos en Raspberry Pi 3 modelo B.
- Desarrollar una aplicación para dispositivos móviles Android con la cual controlar el sistema de procesamiento de datos y para almacenar los resultados.
- Implementar un sistema Web mediante la cual, se puedan observar las mediciones realizadas y el lugar exacto de las mismas en un mapa geográfico.

1.4. Alcance del trabajo

El sistema proporciona información sobre 4 diferentes componentes tóxicos del aire, dicha información permite a las autoridades y a la comunidad en general, planear mejor las actividades al aire libre y el tiempo de exposición, para evitar daños a la salud. Aunque posee ciertas limitaciones, por ejemplo, para realizar las mediciones es necesario mantener una distancia menor a 10 metros en un área sin obtaculos entre el equipo con las Raspberry Pi 3 y el teléfono inteligente, ya que se comunican entre sí mediante comunicación Bluetooth.

En el caso del tiempo de uso, este depende de dos condiciones: la primera involucra la batería utilizada para alimentar el equipo de medición y de la batería del teléfono al tener activado tanto el Bluetooth como el GPS. Además es importante recalcar que el tiempo de vida útil de los sensores es de máximo 2 años por lo que es necesario reemplazarlos por unos nuevos cada que se cumpla este tiempo.

Por otro lado, el sistema completo contribuye con la información necesaria para realizar un análisis de la contaminación en el área y de esta forma ser participe en proyectos relacionados con la predicción de la contaminación, del clima, incluso también en proyectos de reducción del daño ambiental. De esta manera, el sistema permite a la comunidad local conocer la calidad del aire en su zona, y tomar las precauciones que crean pertinentes para evitar sufrir algún daño a su salud.

El trabajo se organiza de la siguiente forma: en el capítulo 2 se hablará acerca del estado del arte, en el capítulo 3 se habla sobre las herramientas y conceptos clave necesarios para el proyecto, en el capitulo 4 se describe el método propuesto para el proyecto, en el capitulo 5 se muestran los resultados de los experimentos realizados para evaluar el sistema y finalmente se muestran las conclusiones y el trabajo futuro.

Capítulo 2

Estado del arte

A diario millones de desechos tóxicos son producidos por los carburantes fósiles y el transporte, la industria química, los problemas de basura, los incendios forestales, la concentración de metano entre otros. Dichas sustancias alteran la composición del aire provocando un deterioro en la calidad de vida de sus habitantes y en su ecosistema.

Los sucesos que se viven en la actualidad a causa de la mala calidad del aire son el incremento de enfermedades respiratorias, daños en la piel, el efecto invernadero, lluvia ácida, por mencionar algunos. Estos efectos han obligado al gobierno de la CDMX poner en marcha un sistema de monitoreo del aire, que le permite evaluar en ciertos periodos el comportamiento de la concentración de los contaminantes mezclado en el aire, informar y prevenir a la población de posibles riesgos, así como llevar un historial de dichas mediciones para hacer estadística y predecir otros factores de riesgo.

2.1. Los sistemas de monitoreo atmosférico.

Los gobierno de los países han implementado programas para el monitoreo de la calidad del aire, en México el SIMAT es la organización encargada de realizar las acciones ya mencionadas anteriormente y cuenta con alrededor de 40 sitios (29 de estos sitios están conformados por equipos de monitoreo continuo y 11 son equipos de monitoreo manual) ubicados en diferentes puntos estratégicos de la CDMX y del Estado de México, estos sitios toman mediciones de los contaminantes (Dióxido de Azufre, Monóxido de Carbono, Dióxido de Nitrógeno, Ozono y partículas suspendidas) basados en su normatividad federal, de igual forma en algunas estaciones se encargan de recolectar información metodológica de la superficie así como la radiación solar ultravioleta. Cada estación, como la de la figura 2.1, realiza las lecturas correspondientes de cada contaminante, basándose en un método de referencia definido por la Agencia de

Protección Ambiental de los Estados Unidos (US EPA, por sus siglas en inglés) y con la Norma Oficial Mexicana para cada contaminante, en caso de que esté disponible. Esto para asegurar que el personal y la infraestructura ocupada para realizar esta tarea, cumplan con las normas correspondientes y se tenga la certeza que la información obtenida es precisa y confiable.

Figura 2.1: Estación de monitoreo, ubicada en la Calzada de los Agustinos s/n, Col. Centro, C.P. 55870

Cada elemento tiene un principio de operación determinado por alguna propiedad física o química del contaminante que se está estudiando. La agencia de protección ambiental de los Estados Unidos renueva constantemente la relación de equipos con designación de método de referencia o equivalente. En el área comprendida entre la CDMX y el Estado de México, la información de la calidad del aire se puede visualizar a través de su página Web [2](ver figura 2.2), que contiene información relevante del estado del aire, por diferentes puntos distribuidos por alcaldías de la CDMX así como recomendaciones para la población bajo ciertos niveles de contingencia.

El proyecto SIMAT cuenta con 27 estaciones fijas en el Valle de México dispersas por toda la región, el problema es que muchas de las estaciones se encuentran agrupadas en pequeñas áreas de la zona por lo que crean espacios en los que no se realiza medición alguna.

[2]`www.aire.cdmx.gob.mx`

Figura 2.2: Página Web de la calidad del aire de la CDMX.

Como se puede observar en la figura 2.3 existe una gran cantidad de estaciones de monitoreo, las cuales están identificadas por puntos de color. El color de cada punto corresponde a la notación mostrada al inicio de la figura 2.3, dicha notación esta diseñada conforme al índice IMECA.

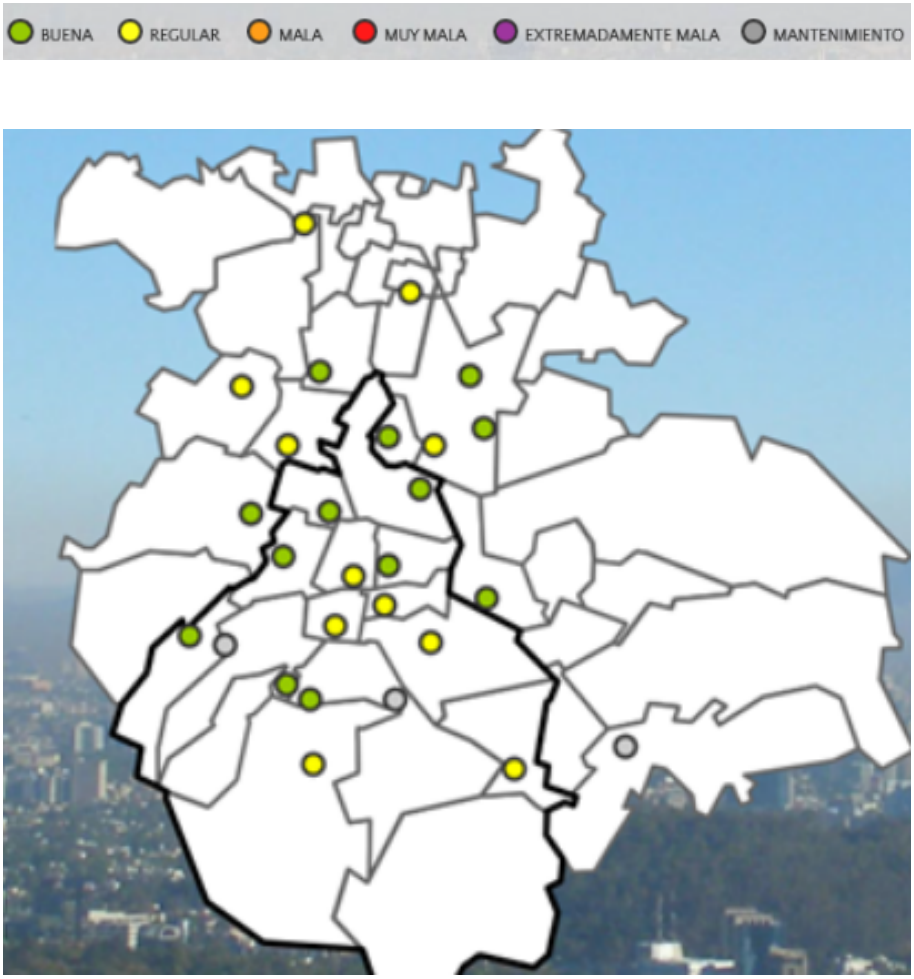


Figura 2.3: Mapa del Valle de México con las ubicación de las estaciones de monitoreo.

Los siguientes son los datos de contacto del programa de calidad de aire de la CDMX en caso de que se requiera conocer más sobre el sistema ya existente en el Valle de México. Vias de contacto:

- Twitter: *@Aire_CDMX*
- IMECAtel: 5278-9931 ext. 1

- e-mail: calidadaire@sedema.cdmx.gob.mx
- App: Aire en la App Store para iOS y Google Play para Android

2.2. Proyectos relacionados

2.2.1. Sistema de medición de calidad del aire con base a las normas IEEE/ISO/IEC:

Es un sistema que hace uso de la tecnología GSM que le permite realizar mediciones en tiempo real y a distancias largas de los gases contaminantes como el CO2, CO, NO2 y SO2. Esto se logra bajo la comunicación máquina a máquina es decir desde un equipo de cómputo y la estación de monitoreo. Las mediciones se logran con sensores electroquímicos en este caso sensores infrarrojos que consumen poca energía y son muy precisos con la ayuda de una interfaz gráfica que permite al usuario interactuar con el sistema de una manera sencilla y eficaz.

Los valores de concentración de gas son trazados en la interfaz y con la ayuda de la calibración de los instrumentos a determinados intervalos de tiempo asegura que la precisión deseada sea confiable, posteriormente los datos son almacenados en una tarjeta SD en forma de texto que más adelante son enviado a una PC configurada como servidor. Para administrar la información se utiliza MySQL.

2.2.2. Proyecto del CIC

En el Centro de Investigación en Computación (CIC) del Instituto Politécnico Nacional (IPN) se realizó una tesis de maestría que tenía como finalidad el desarrollo de un modelo computacional para evaluación de la calidad del aire en el Valle de México utilizando procesos analíticos jerárquicos. Para el desarrollo de esta tesis se utilizaron los datos proporcionados por las estaciones de monitoreo del SIMAT tomando los registros de los contaminantes: $PM_{2,5}$, PM_{10} , O_3 ,NO_2 , SO_2 y CO, una vez que se obtiene esta información se procede a realizar un modelo de razonamiento difuso con el cual se lleva a cabo la evaluación de la calidad del aire para posteriormente crear un modelo de asignación de jerarquías el cual determinará la asignación de pesos a los diferentes contaminantes tomados en cuenta para así establecer un orden de importancia. Habiendo creado estos modelos se procedió a comparar los resultados obtenidos con los principales indicadores de calidad del aire como lo son el IMECA y el USEPA (United States Environmental Protection Agency).

2.2.3. Proyecto CLAIRITY

CLAIRITY es un proyecto desarrollado por el MIT (Massachusetts Institute of Technology), cuyo objetivo es evaluar la exposición a la que se enfrenta la población del MIT en cuanto a contaminación del aire se refiere, esto mediante el diseño y la implementación de una red de sensores de calidad del aire distribuida a través del campus del MIT. La red de sensores del MIT cuenta con un equipo especial diseñado

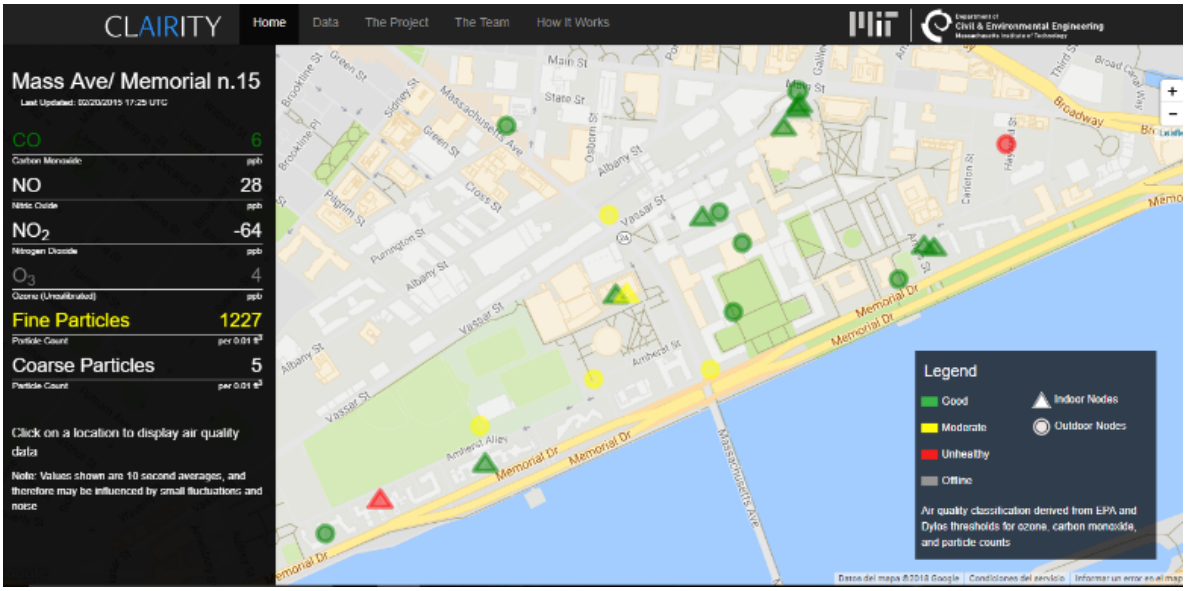


Figura 2.4: Mapa del Memorial n.15 en la página oficial del proyecto CLAIRITY en el cual se puede ver la concentración de diversos contaminantes.

por el equipo de CLAIRITY llamado “node” figura2.5 el cual se compone de un sistema ”Dylos”; el cual es una pequeña computadora que cuenta con un sistema láser que toma medidas sobre la cantidad de partículas que se encuentra en el aire esto absorbiéndolas mediante un ventilador, y además cuenta con una pantalla la cual muestra los resultados obtenidos, pero el dispositivo además fue modificado para conducir el aire a un sensor de temperatura y a un equipo que contiene 4 sensores “Alphasense”. Respecto a los sensores Alphasense, estos son sensores electroquímicos los cuales miden la concentración de los contaminantes que son introducidos al “node”; los contaminantes que pueden ser monitoreados con estos sensores son: el Monóxido de Carbono (CO), el Monóxido de Nitrógeno (NO), el Dióxido de Nitrógeno (NO_2) y el Ozono (O_3). Otro componente importante son las tarjetas de circuito impreso (o PCB en inglés), estas tarjetas sirven no solo para energizar a todos los componentes del equipo, sino que además comunica los datos recolectados de todos los sensores hacia una tarjeta con convertidor analógico-digital, el cual como su nombre indica su trabajo es convertir una entrada analógica en

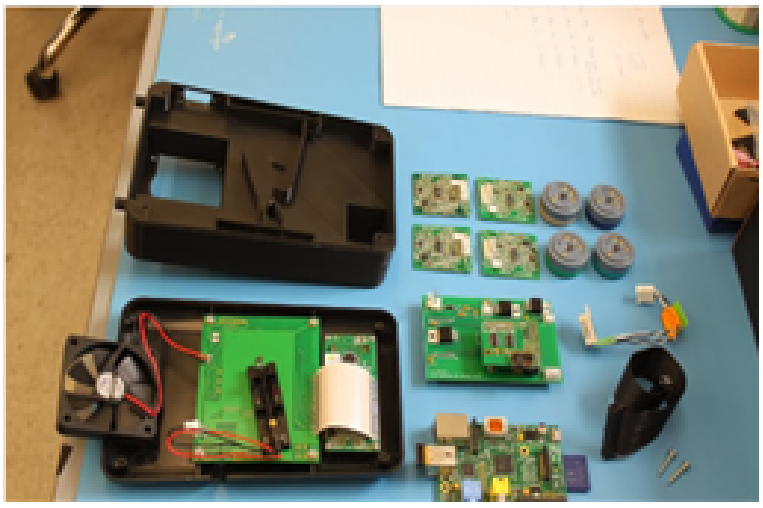


Figura 2.5: Componentes utilizados en los equipos "node"del proyecto CLAIRITY.

una salida digital para que una Raspberry Pi pueda procesar los datos obtenidos. Por último, la Raspberry Pi almacenará la información y la enviará a una base de datos de manera inalámbrica.[3]

2.2.4. Prototipo portátil de monitoreo ambiental desarrollado en Quito, Ecuador.

En este proyecto, de la Facultad de Ingeniería eléctrica y electrónica de Quito, Ecuador, se presenta el diseño de un sistema autónomo para la detección de variables, que para este caso son los niveles de contaminantes, gases como lo son el Dióxido de Azufre (SO_2), Dióxido de Nitrógeno (NO_2), Monóxido de Carbono (CO), Ozono (O_3) y Dióxido de carbono (CO_2), así como también la presión, temperatura y humedad, con el propósito de almacenar los datos y analizar su comportamiento para predicciones futuras. Esta información es enviada a una computadora, gracias a la interfaz que con que ya vienen integrados los sensores. El prototipo cuenta con baterías recargables que le brindan una autonomía de tres a 5 horas de operación continua, logrando satisfacer cinco cámaras de gases que son independientes para la toma de muestras, así como una interfaz hombre máquina que le facilita mostrar los datos censados a través de una LCD (pantalla de cristal líquido).

[3]Véase: http://clairity.mit.edu/site/html/home.html

Figura 2.6: Prototipo final del sistema de Quito

Capítulo 3

Marco teórico

Los altos niveles de contaminación han representado un alto riesgo ambiental para la salud de los seres humanos y la alteración del ecosistema. Se le ha visto como la causante de accidentes cerebrovasculares, cánceres de pulmón y neuropatías crónicas y agudas entre ellas el asma, de acuerdo a la secretaría de la salud se estima que el siete por ciento de la población en México sufre de asma, en donde el 80 % de los adultos que padecen esta enfermedad, manifestaron síntomas desde los primeros cinco años de vida.

En el 2012 se estimó que la contaminación mató alrededor de doce millones de personas lo que la ha convertido en un gran problema de salud medio ambiental mundial según las OMS (Organización Mundial de la Salud). En los países que han desarrollado programas y tecnologías para disminuir los niveles de contaminación han logrado disminuir la tasa de mortalidad causada por la mala calidad del aire, logrando que entre más bajo sean los niveles de contaminación mejor será la salud cardiovascular y respiratoria para la población, tanto a largo como a corto plazo, sin mencionar que el calentamiento global siga en incremento [16].

3.1. Métodos de medición de contaminantes

Para realizar mediciones sobre contaminantes existen métodos de operación para realizarlas, estos son propuestos por la EPA, y se toman como métodos de referencia, pero tratar de implementarlos en estaciones de medición expuestas al ambiente urbano regular supone cierto grado de peligro ya que de no tener cuidado estos pueden ser dañados, así que la EPA también propuso métodos equivalentes para realizar las mediciones. Estos métodos se exponen en la siguiente tabla:

Tabla 3.1: Métodos para la medición de la calidad del aire.

Contaminante	**Principio de operación**	**Descripción del método**
Dióxido de azufre (SO_2)	Fluorescencia UV	Método equivalente: Medición de la fluorescencia emitida por las moléculas de SO2 cuando son excitadas por una fuente de radiación ultravioleta.
Monóxido de carbono (CO)	Absorción en el infrarrojo	Método de referencia: Medición de la absorción de luz infrarroja por parte del Monóxido de carbono en una celda de correlación.
Dióxido de nitrógeno (NO_2)	Quimioluminiscencia	Método de referencia: Medición de la luz emitida durante la reacción entre el NO y el O_3. La separación de las especies nitrogenadas se realiza a través de la medición diferencial de NO y $NO2$ (previa reducción catalítica). El valor de NO_X corresponde a la suma de $NO + NO_2$.
Ozono (O_3)	Fotometría UV	Método equivalente: Absorción de luz ultravioleta en una longitud de onda de $254nm$, la disminución en la intensidad es proporcional a la concentración de ozono de acuerdo a la ley de Beer-Lambert.
Partículas suspendidas PM_{10}, $PM_{2,5}$	Gravimetría	Método equivalente: Determinación de la masa de partículas presente en un flujo de aire. Las partículas son separadas de la corriente y depositadas sobre un filtro colocado en un elemento oscilante, la variación en la frecuencia de oscilación es proporcional a la masa. El tamaño de partícula está determinado por la entrada selectiva y el flujo de muestra.
Partículas suspendidas PM_{10}, $PM_{2,5}$	Atenuación de radiación beta	Método equivalente: Atenuación en la intensidad de la radiación beta por las partículas depositadas sobre un filtro continuo.

Los métodos equivalentes de la tabla anterior a pesar de ser más sencillos que los de referencia aun así se muestran un tanto difíciles de reproducir en masa para equipos que serán puestos al servicio de la comunidad regular. Así que muchas compañías desarrollan sensores de diferentes tipos los cuales reaccionan de diferentes formas a los contaminantes del aire, pudiendo ser utilizados casi por cualquiera que los necesite.

En el año 2006 aparecieron dos normas más que definen los requisitos para la elaboración del índice de la calidad del aire.

Tabla 3.2: Normas para la elaboración de un índice de calidad del aire

Norma	**Publicación**	**Descripción**
NADF-009-AIRE-2006	29 de noviembre de 2006	Establece los requisitos para elaborar el Índice Metropolitano de la Calidad del Aire.
NOM-156-SEMARNAT-2012	16 de julio de 2012	Establece las características de sistemas de monitoreo de la calidad del aire.

En cuanto a los estándares publicados en la Normas Oficiales Mexicanas (NOM), estos especifican un límite de partículas por cada compuesto, dichas normas son estudiadas y actualizadas continuamente basándose en los efectos que hay en la salud de sus habitantes y en la gestión de la calidad del aire. La tabla 3.3 muestra las NOM oficiales que establecen el método de medición de los contaminantes.

3.2. Índice de calidad del aire

El SIMAT cuenta con un índice indicador que muestra a la población el estado de la calidad del aire, que tan contaminado está el aire y que consecuencias podría ocasionar en el estado en que se encuentre. El índice permite visualizar cinco contaminantes criterios: Dióxido de Azufre, Monóxido de Carbono, Dióxido de Nitrógeno, Ozono y partículas suspendidas representados en un intervalo de 0 a 500 donde el valor de 100 es asignado al valor indicado por la NOM para cada compuesto, un valor menor de 100 se considera de bajo riesgo para la población ya que el número de contaminantes en el aire es mínima y un número que sobrepase el 100 es considerado una amenaza para la salud de sus habitantes, entre más alejado esté del 100 mayor serán los efectos en la población.

En la tabla 3.4, publicada en la Gaceta Oficial del Distrito Federal como la Norma Ambiental "*NADF-009-AIRE-2006*", se muestran cada intervalo, su significado y las recomendaciones que se deben seguir de acuerdo al mismo.

Tabla 3.3: Normas que establecen la medición de los contaminantes en México

Contaminante	**NOM**	**Publicación**	**Descripción**
Dióxido de azufre (SO_2)	NOM-038-SEMARNAT-1993	18 de octubre de 1993	Método equivalente: fluorescencia ultravioleta
Monóxido de carbono (CO)	NOM-034-SEMARNAT-1993	18 de octubre de 1993	Método de referencia: absorción en el infrarrojo
Dióxido de nitrógeno (NO_2)	NOM-037-SEMARNAT-1993	18 de octubre de 1993	Método de referencia: quimioluminiscencia en fase gaseosa
Ozono (O_3)	NOM-036-SEMARNAT-1993	18 de octubre de 1993	Método equivalente: Fotometría ultravioleta
Partículas suspendidas totales (PST)	NOM-035-SEMARNAT-1993	18 de octubre de 1993	Muestreo: Alto volumen Análisis: Gravimetría
Partículas menores a 10 micrómetros (PM_{10})	No se cuenta con una NOM de métodos de medición, sin embargo, se considera el método equivalente que recomienda la US EPA.		Gravimetría o atenuación de radiación beta
Partículas menores a 2.5 micrómetros ($PM_{2,5}$)	No se cuenta con una NOM de métodos de medición, sin embargo, se considera el método equivalente que recomienda la US EPA.		Gravimetría o atenuación de radiación beta
Plomo (Pb)	No se cuenta con una NOM de métodos de medición.		

Tabla 3.4: Intervalo, significado y recomendaciones a seguir en cada categoría de contaminación [4].

Categoría	**Intervalo**	**Mensaje**	**Significado**	**Recomendaciones**
BUENA	0-50	Sin riesgo	La calidad del aire es satisfactoria y existe poco o ningún riesgo para la salud.	Se puede realizar cualquier actividad al aire libre.
REGULAR	51-100	Aceptable	La calidad del aire es aceptable, sin embargo, en el caso de algunos contaminantes, personas inusualmente sensibles, pueden presentar síntomas moderados.	Las personas que son extremadamente sensibles a la contaminación deben considerar limitar los esfuerzos prolongados al aire libre.
MALA	101-150	Dañina a la salud de los grupos sensibles	El público en general usualmente no es afectado.	Los niños, adultos mayores, personas que realizan actividad física intensa o con enfermedades respiratorias y cardiovasculares, deben limitar los esfuerzos prolongados al aire libre.
MUY MALA	151-200	Dañina a la salud	Todos pueden experimentar efectos en la salud.	La población en general debe limitar el esfuerzo prolongado al aire libre.
EN EXTREMO MALA	>200	Muy dañina a la salud	Representa una condición de emergencia. Toda la población tiene probabilidades de ser afectada.	La población en general debe suspender los esfuerzos al aire libre.

3.3. Contaminantes

La calidad del aire se ve afectada en su mayoría por los distintos contaminantes producidos por las maquinas del hombre las cuales usualmente utilizan combustibles fósiles para funcionar, aunque otras tantas generan gases tóxicos a partir de los procesos que realizan por ejemplo los hornos para la quema de basura. Es por eso que es necesario conocer los contaminantes principales que afectan la calidad del aire como son:

Óxido de Azufre (SO_2) : Es un gas incoloro que presenta un característico olor asfixiante. Se crea en varios procesos de combustión ya sea de carbón, gas natural, petróleo y diesel, los cuales contienen cantidades de compuestos azufrados. El Dióxido de Azufre es un gas tóxico e irritante el cual afecta las mucosidades y los pulmones lo cual provoca ataques de tos, y aunque en su mayor parte es absorbido por el sistema nasal, el exponerse a altas concentraciones por cortos periodos de tiempo puede causar bronquitis, irritación del tracto respiratorio y congestión de los conductos bronquiales en asmáticos. Cuando se convierte en Lluvia Ácida, produce daños en la vegetación, ya que al filtrarse al suelo incrementa la acidez modificando la composición de este, lo que termina por disolver los nutrientes que utilizan las plantas, dañando sus tejidos y haciéndolas vulnerables a plagas. La *Norma Oficial Mexicana (NOM-022-SSA1-2010)* recomienda concentraciones menores a 110 ppb como promedio máximo de 24 horas y 200 ppb como el segundo máximo de los promedios de 8 horas. Además recomienda un límite de exposición crónica de 25 ppb como promedio anual. [10]. Las siguientes ecuaciones muestran algunas de las reacciones mediante las cuales se obtiene Dióxido de Azufre:

$$S_8 + 8O_2 \rightarrow 8SO_2 : \text{Combustión del Azufre} \tag{3.3.1}$$

$$2H_2S + 3O_2 \rightarrow 2H_2O + 2SO_2 : \quad \text{Combustión del Sulfuro de Hidrógeno} \tag{3.3.2}$$

Monóxido de Carbono (CO): También denominado como Óxido de Carbono, Gas Carbonoso o Anhídrido Carbonoso, es un gas incoloro e inodoro altamente tóxico que puede producir la muerte al respirar altas concentraciones de este. Se produce a partir de la combustión deficiente de sustancias como gasolina, gas, carbón, petróleo, madera, tabaco o queroseno, por lo que las chimeneas calentadores de agua, calefactores, vehículos con motor de combustión, calderas y aparatos domésticos que queman combustible pueden producirlo en grandes cantidades si no funcionan adecuadamente. En altas concentración el Monóxido de Carbono puede ser fatal para los seres vivos, puede inhabilitar el transporte del oxígeno hacia las células y provocar mareos, náuseas, dolores de cabeza, estados de inconciencia e inclusive la muerte. La *Norma Oficial Mexicana (NOM-021-SSA1-1993)*

establece un límite para la concentración en aire ambiente de 11 ppm, para un promedio de 8 horas.[15] Las siguientes ecuaciones muestran las reacciones por las cuales se puede obtener el Monóxido de Carbono:

$$O_2 + 2C \Leftrightarrow 2CO : \text{Combustión con Oxígeno} \tag{3.3.3}$$

$$H_2O + C \Leftrightarrow H_2 + CO : \text{ Gas de agua} \tag{3.3.4}$$

Ozono(O_3): Es un gas incoloro, con un olor muy irritante y muy reactivo. Es una de las formas en las que se encuentra el Oxígeno en el ambiente, y se forma mediante 3 moléculas de Oxígeno (O_2) . Al nivel de la Troposfera se forma una relación entre los óxidos de nitrógeno emitidos durante la combustión de hidrocarburos, la luz solar y los compuestos orgánicos volátiles producidos naturalmente a través de las plantas, frutas, etc. En altas concentraciones funciona como un potente oxidante, el cual produce irritación en los ojos y las vías respiratorias, disminuyendo la función respiratoria. Respecto a las plantas, el daño varía de acuerdo al tiempo de exposición, concentración de Ozono y la sensibilidad al mismo, pero en la mayoría de las actividades agrícolas provoca una disminución importante en el rendimiento de los cultivos. En México la *Norma Oficial Mexicana (NOM-020-SSA1-2014)* recomienda concentraciones menores a 0.095 ppm para el promedio de 1 hora, y menores a 0.070 ppm para el promedio de 8 horas (máximo anual).[9]

Dióxido de Nitrógeno(NO_2): Es un gas perteneciente al grupo de los Óxidos de Nitrógeno, formado a través de dos partículas de Oxígeno y una de Nitrógeno. Todos los Óxidos de Nitrógeno se forman principalmente a partir de la combustión, por lo que los autos en la CDMX son la principal fuente de estos. De todos los Óxidos de Nitrógeno, solo el Dióxido de Nitrógeno tiene efectos negativos sobre la salud. La exposición a altas concentraciones puede ocasionar daño en la membrana celular del tejido pulmonar y a bajas concentraciones puede ocasionar irritación en las vías respiratorias o agravar los síntomas de enfermedades respiratorias como bronquitis y pulmonía. La *Norma Oficial Mexicana (NOM-023-SSA1-1993)* establece un límite para el dióxido de nitrógeno (NO_2) de 210 ppb para el promedio de una hora, el cual no debe excederse más de una vez al año.[11]

Partículas suspendidas($PM_{10}, PM_{2,5}$): Son cualquier tipo de material sólido o líquido que se encuentra en suspensión en el aire ambiente. En la Ciudad de México una fracción importante se forma de reacciones químicas en la atmósfera contaminada. Su tamaño puede variar, las más pequeñas apenas miden unas cuantas millonésimas de milímetro, mientras que las más grandes son del tamaño de granos de arena. Entre las fuentes de emisión de este contaminante están las tolvaneras,

los incendios, las emisiones de camiones y automóviles. Las partículas suspendidas representan el principal problema de salud pública, ya que sus efectos dependen de la concentración, composición química y tamaño. El riesgo es mayor a medida que se reduce el tamaño de la partícula, y el incremento en la concentración está relacionado con enfermedades respiratorias, cardiovasculares y un incremento en el riesgo de mortalidad. En México la *Norma Oficial Mexicana (NOM-025-SSA1-2014)*[12] establece los indicadores para partículas suspendidas:

- Partículas menores a 10 μm: $40\mu g/m^3$, promedio anual; $75\mu g/m^3$, promedio de 24 hora.
- Partículas menores a 2. 5 μm: $12\mu g/m^3$, promedio anual; $45\mu g/m^3$, promedio de 24 horas.

3.4. Herramientas de desarrollo

En esta sección se explica brevemente las herramientas utilizadas para el desarrollo del prototipo, así como algunas de sus características.

3.4.1. Sensores de gases

Los sensores de gases o sistemas de detección de gases son dispositivos básicamente de seguridad ya que permiten proteger y asegurar la zona, ya que sirven para conocer la concentración de diferentes gases, existiendo sensores que se centran en un solo gas hasta aquellos que pueden detectar varios gases a la vez, además de que pueden ser dispositivos portátiles (semi-portátiles) o fijos, los cuales difieren por el tiempo que pueden estar activados, ya que los fijo normalmente deben permanecer activados todo el año las 24hrs. en cambio los portátiles además de poder ser trasladados aun activos, estos pueden estar apagados cierto tiempo para poder ahorrar energía u otras razones. Para elegir que sensores ocupar en un sistema de detección de gases además de las razones anteriormente expuestas y de las razones obvias como el costo, tamaño, etc. Se debe tener en cuenta el uso que se le dará a la información recopilada por el sistema. La EPA (Agencia de Protección Ambiental), clasificó la exactitud que deben poseer los sensores de acuerdo al uso de la información como se expresa en la siguiente tabla:

Tipos de sensores de gas

Existe una amplia variedad de sensores de gas, los cuales se pueden catalogar de acuerdo a la forma en que realizan la medición, por ejemplo:

Tabla 3.5: Intervalo y descripción de las aplicaciones de sistemas de detección de gases. [20]

Aplicación	**Precisión y bias**	**Exactitud**	**Contaminante**	**Descripción**
Educación e información	<50 %	Mayor o igual al 50 %	Varios	El error de medición no es importante sino el saber si existe presencia del contaminante en cuestión.
Caracterización de la zona	<30 %	Mayor o igual al 75 %	Varios	Debe mejorar la exactitud para poder conocer la concentración de gas con un valor más cercano al real.
Monitoreo suplementario	<20 %	Mayor o igual al 80 %	Elementos tóxicos del aire	La exactitud y precisión deben ser mayor, ya que este tipo de sistemas sirve de apoyo a sistemas ya existentes por lo que la información que entreguen debe servir de ayuda al monitoreo.
Exposición personal	< 30 %	Mayor o igual al 80 %	Varios	La precisión y el bias de esta aplicación son los utilizados en literatura científica, ya que permiten conocer cómo, cuándo y dónde fue la exposición.
Monitoreo regulado	<7 % <10 % <15 % <10 %	Mayor o igual al 75 %	O_3 CO, SO_2 NO_2 $PM_{10}, PM_{2,5}$	Se necesita alta precisión para asegurar alta fiabilidad en los datos para cumplir con los requisitos regulatorios.

Sensores electroquímicos Están formados por dos electrodos sumergidos en un medio electrolítico común. El electrolito es aislado de las influencias externas mediante una barrera, que puede ser una membrana permeable al gas. Normalmente son compactos, requieren muy poca energía, detectan la mayoría de los gases tóxicos comunes, muestran gran linealidad y repetibilidad, además de que poseen una larga vida útil (normalmente de uno a tres años). Los tiempos de respuesta, pueden ir de 10 a 60 segundos, la sensibilidad oscila entre 0.02 y 50 ppm según el tipo de gas. Funcionamiento Se aplica

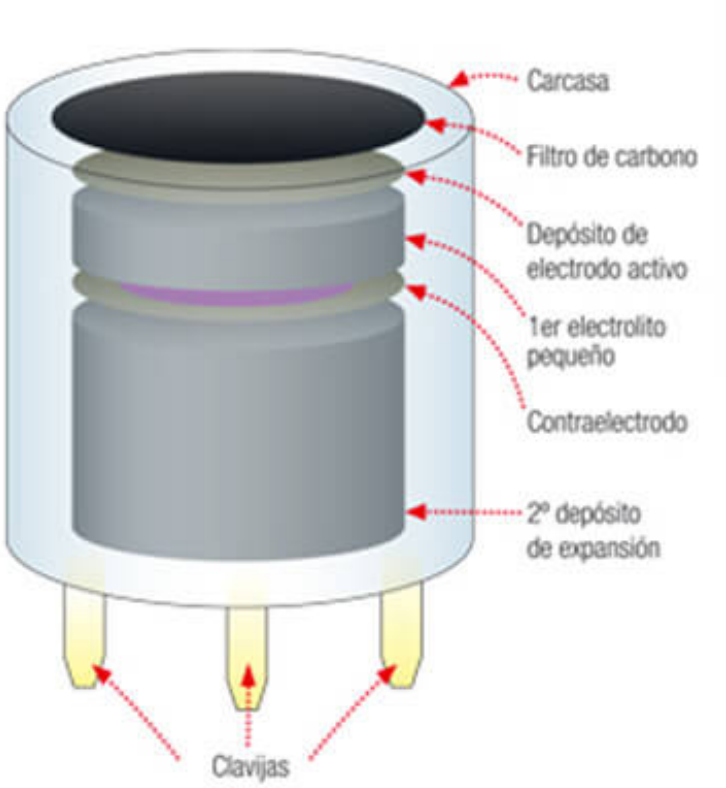


Figura 3.1: Esquema de sensor electroquímico.

una diferencia de potencial a los electrodos del sensor y cuando el gas penetra a través de la membrana del sensor hacia su interior, se produce una reacción Redox la cual genera una corriente eléctrica proporcional a la concentración del gas.

Sensores por semiconductor Funcionan con una película sensible al gas hecha a base de semiconductores, los cuales al hacer contacto con el gas provocan un decremento en la resistencia eléctrica del sensor. Estos sensores son muy eficientes ya que pueden operar en un amplio rango de ambientes húmedos, pero el problema reside en que la

sensibilidad varia con la temperatura.

Sensores de conductividad térmica Consisten en colocar con dos filamentos con propiedades conductoras y térmicas formando así un puente de Wheastone. Se guarda una

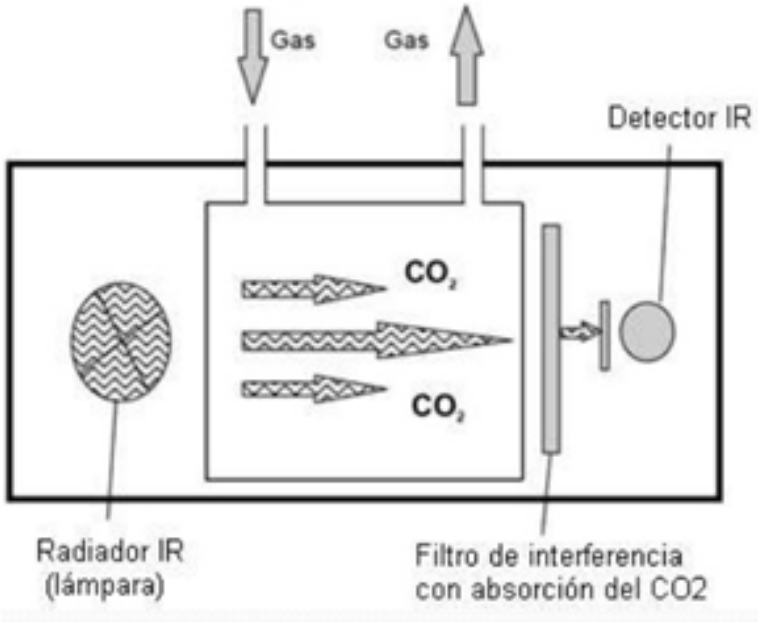


Figura 3.2: Esquema de sensor de conductividad térmica.

cantidad determinada de un gas en las entradas del sensor, luego se almacenará dentro de una célula la cual será sometida a una diferencia de temperatura, entonces el gas a detectar el cual posee una conductividad térmica diferente al gas restante de la célula, hará que se altere la temperatura del filamento haciendo que se desequilibre el puente de Wheastone.

Sensores catalíticos También llamados pellistores, estos sensores se componen de dos bobinas de platino encapsuladas en un material cerámico. Uno de los encapsulados está cubierto de un material catalizador (como el paladio), lo que causa y acelera la oxidación del elemento (detector) mientras que el otro encapsulado no posee ese material para la oxidación (referencia). Este sensor consiste en la oxidación del gas en la superficie del elemento catalítico por medio de calor generado a partir de una corriente eléctrica que circula por la bobina. La corriente al pasar por las bobinas alcanzara una temperatura entre los 450°C y los 550°C, causando la oxidación del gas.

Sensores infrarrojos. Gases cuyas moléculas tienen de dos a mas átomos disimiles que

absorben la radiación infrarroja en largos de ondas específicas. Esta energía absorbida causa que se incremente la temperatura de las moléculas de gas. El cambio de temperatura se mide como una concentración de gas.

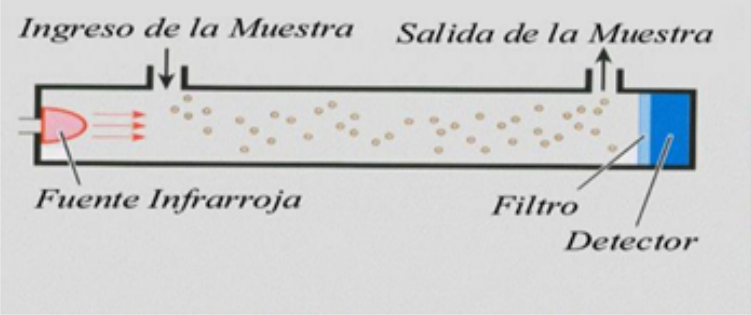


Figura 3.3: Esquema de un sensor Infrarrojo.

3.4.2. Divisor de voltaje

El divisor de voltaje es un circuito que permite convertir una entrada de voltaje grande a un voltaje menor que el de entrada. El divisor de voltaje consta en esencia, de

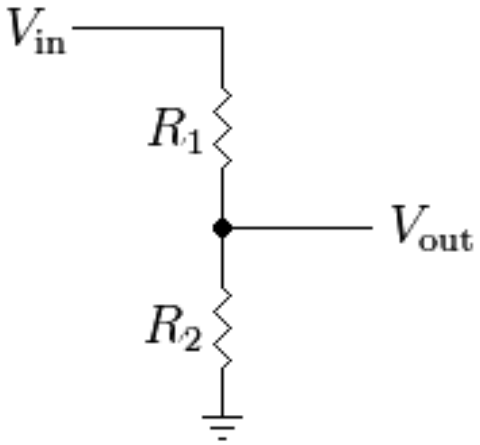


Figura 3.4: Esquema básico de un divisor de voltaje

dos resistencias R_1 y R_2 obteniendo en las terminales de la resistencia R_2, la salida de voltaje, donde se obtiene una fracción de nuestro voltaje de entrada. Para la ecuación de voltaje se supone que se conocen tres de las cuatro variables del diagrama en la figura 3.4, voltaje de entrada V_{in} y ambos valores de resistencia (R_1 y R_2), tomando

en cuenta en estos valores se utiliza la siguiente ecuación para encontrar el voltaje de salida V_{out}.

$$V_{out} = V_{in} * \frac{R_2}{R_1 + R_2} \tag{3.4.1}$$

Se observa que la ecuación establece que el voltaje de salida es directamente proporcional al voltaje de entrada conforme la relación entre R_1 y R_2. Generalidades: Si R_1= R_2entonces el voltaje de salida es la mitad de la entrada, cabe destacar que esto siempre se cumplirá independientemente del valor de las resistencias.

$$R_1 = R_2 : V_{out} = V_{in} * \frac{R}{2R} = \frac{V_{in}}{2} \tag{3.4.2}$$

Si $R_1 < R_2$, entonces el voltaje de salida será muy cercano al de la entrada. Lo que presume que la mayor cantidad de voltaje se encontrara en la resistencia R_2.

$$R_1 < R_2 : V_{out} \approx V_{in} * \frac{R_2}{R_2} = V_{in} \tag{3.4.3}$$

Aplicaciones:
El divisor de voltaje tiene diferentes aplicaciones y es uno de los circuitos más usados, es usado en potenciómetros, en la lectura de sensores resistivos, cambios de nivel, entre otros.

3.4.3. Convertidor Analógico Digital

Un convertidor analógico digital (ADC), es un sistema que permite convertir señales analógicas (puede ser voltaje o corriente) a señales digitales esto se logra a través del proceso del muestreo, cuantización y codificación obtenido a la salida un numero binario. Debido que la Raspberry Pi no cuenta con un ADC integrado, se pueden obtener ADC desarrollados específicamente para Raspberry Pi, como el ADC Pi, ADC-DAC Pi Zero, ADC Pi Plus y ADC Differencial Pi. A continuación, se habla un poco más de estos:

ADC Pi: Es un convertidor analógico digital de ocho canales que puede ser configurado con 11, 13, 15 y 17 bits. Para su funcionamiento hace uso de dos convertidores analógico digital MPC3424 A/D de Microchip que le permite tener bajo ruido en sus entradas analógicas. ADC-DAC Pi Zero: Es un convertidor analógico digital de dos canales de 12 bits y un convertidor, que funciona a través de MCP3202 A/D de Microchip con referencia de voltaje bajo.

ADC Pi Plus: Es un convertidor analógico digital de ocho canales de 17 bits integrado con un MCP3424 integrado con dos MCP3424 A/D de Microchip que le ofrece un bajo ruido en sus entradas.

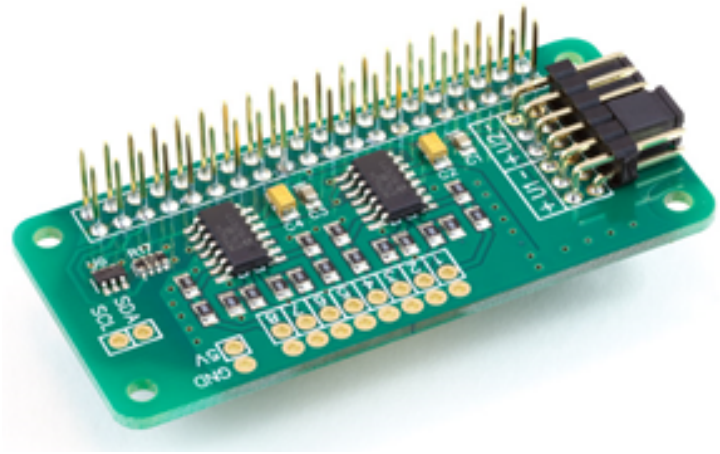


Figura 3.5: ADC Pi

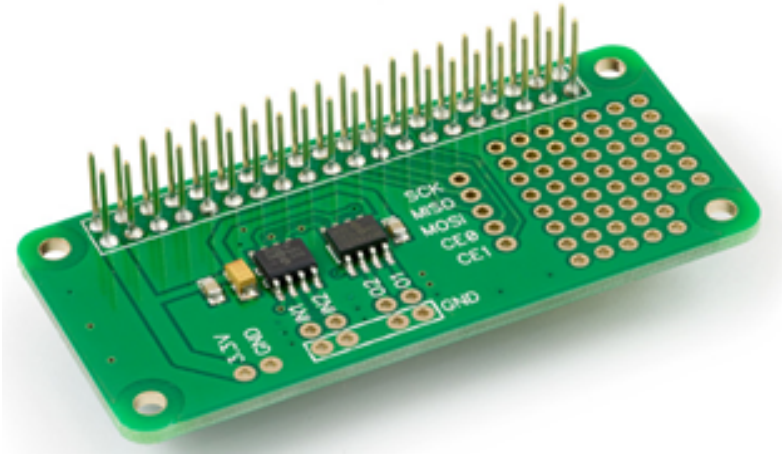


Figura 3.6: ADC-DAC Pi

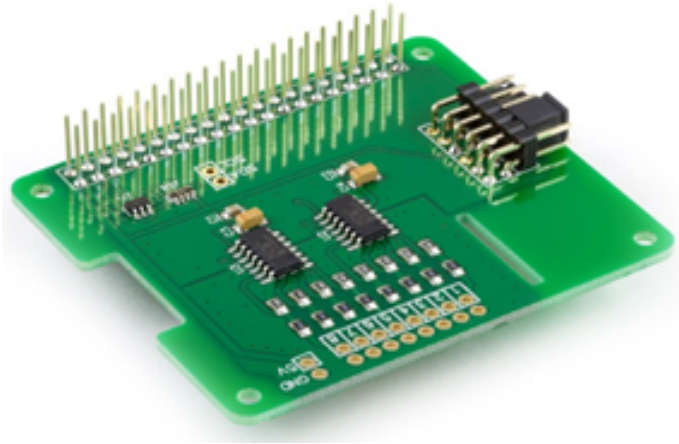


Figura 3.7: ADC Pi Plus

ADC Differential Pi: Es un convertidor analógico digital de ocho canales que permite una configuración de 12,14 16 y 18 bits de resolución, y al igual que los anteriores cuenta con dos conversores MCP3424 A/C de Microchip. En sus entradas soportan un voltaje de entrada de +- 2.048 V, debido a que en su diseño no cuenta con un divisor de voltaje, le permite utilizar un divisor de voltaje externo en caso de tener que trabajar con voltajes altos. Se comunica con la Raspberry Pi a través del protocolo I2C sin dejar atrás que le facilita la comunicación con otros dispositivos con I2C que operen a 5 V sin dañar el I2C de 3.3 Volts. El ADC Differential Pi destaca de entre los demás ADC

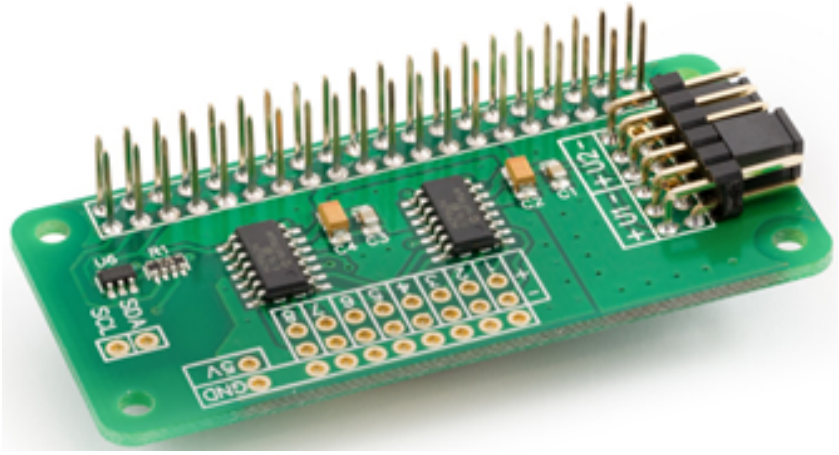


Figura 3.8: ADC Differential Pi

anteriormente expuestos debido a su bajo costo, su mayor resolución y por el rango de voltaje a la entrada que va de entre +2.048V a -2.048V.

3.4.4. Protocolo I2C

El bus I2C o Inter Integrated Circuit (Inter IC), es un bus bidireccional para la comunicación entre dispositivos que tengan integrado este bus. Fue desarrollado por Philips Semiconductors (hoy en día conocido como NXP Semiconductors). Su desarrollo se debe a la necesidad de resolver la gran cantidad de problemas que existen al diseñar circuitos digitales ya que muchos de los dispositivos utilizados, no contaban con las mismas interfaces para comunicarse entre si lo que generaba un costo mayor para las empresas ya que tenían que diseñar sus propias interfaces, así como protocolos de comunicación para sus circuitos [19]. El bus I2C cuenta con muchas características que lo hacen eficiente para la comunicación entre dispositivos, algunas son:

- Solo se requiere de 2 líneas de comunicación: La línea de datos serial (SDA) y la línea de reloj serial (SCL).

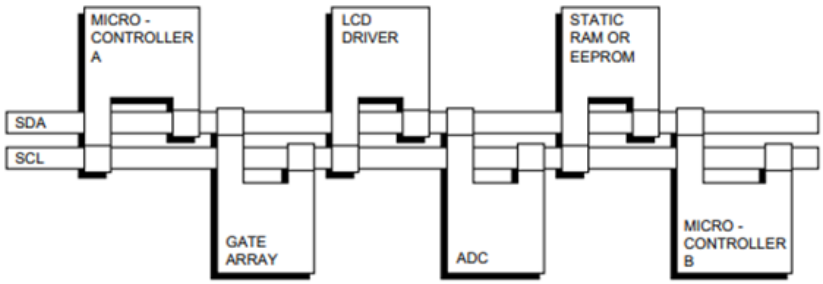


Figura 3.9: Esquema general del bus I2C

- Cada dispositivo conectado mediante el bus I2C puede ser direccionado mediante software, y además cada uno posee una dirección única al conectarse al bus.
- Los dispositivos conectados al bus forman una relación de maestro/esclavo entre sí; los dispositivos maestros tienen la posibilidad de fungir tanto como maestros-transmisores como maestros-receptores.
- Es un bus multimaestro, lo que significa que cuenta con detección de colisiones y una forma de arbitraje lo que previene la corrupción de datos en el caso de que dos o más maestros logren iniciar la transferencia de datos simultáneamente.
- Es serial y orientado a 8 bits.
- Cuenta con diferentes velocidades de transferencia de acuerdo al modo en que esté funcionando, por ejemplo:
  - Hasta 100kbit/s en modo Estándar (Standard-mode).

  - 400kbit/s en modo rápido (Fast-mode).
  - 1Mbit/s en modo rápido plus (Fast-mode Plus).
  - 3.4Mbit/s en modo de alta velocidad (High-speed mode).
  - Puede llegar hasta los 5Mbit/s en modo ultra rápido (Ultra Fast-mode), pero esto ocasiona que la transferencia solo sea unidireccional.
- Cuenta con filtros montados en chip para eliminar picos de voltaje en el bus de datos y así preservar la integridad de los datos.
- La cantidad de circuitos integrados que pueden ser conectados al mismo bus está limitado por la capacitancia máxima del bus.

Los circuitos integrados (IC's) compatibles con el bus I2C permiten que el diseño del sistema progrese más rápido ya que, aunque existan modificaciones, estas se pueden implementar directamente al prototipo sin gastar muchos recursos para modificar el sistema. Lo anterior también permite que el prototipo pueda ser modificado, o mejorado ya que una de las mayores ventajas del bus I2C es que los circuitos conectados mediante este bus pueden ser conectados y desconectados sin ningún procedimiento especial.
Las siguientes son algunas de las ventajas de utilizar IC's compatibles con el bus I2C, que son particularmente atractivas para los diseñadores que trabajan con equipos portátiles que utilizan baterías:

- Bajo consumo de energía.
- Alta resistencia al ruido.
- Amplio rango de temperatura de operación.
- Amplio rango de voltaje de alimentación.

3.4.5. Raspberry Pi 3

La Raspberry Pi 3 es una microcomputadora de bajo costo, se creó en el año del 2006 por la fundación Raspberry Pi en Reino Unido, con el propósito de involucrar a niños y a jóvenes en el aprendizaje de los lenguajes de programación. Cuenta con un procesador ARM Cortex A53 de 4 núcleos de 64 bits, que trabaja a una velocidad de 1.2 Ghz. Esto le brinda un rendimiento 50 % mayor al de las versiones anteriores de Raspberry, ofrece conectividad Wifi, Bluetooth, así como también alámbrica mediante el puerto ethernet. Trabaja con una memoria RAM de 1Gb, y permite conectar cuatro dispositivos a través de sus puertos USB, además tiene un puerto HDMI, y una entrada Micro SD en la cual se debe insertar una memoria del mismo tipo que hará función de

disco duro para el sistema.
Es compatible con diferentes sistemas operativos como: Ubuntu Mate, Q4OS, OSMC, Kali Linux, RISC OS, Recalbox, Raspbian, Windows 10, entre otros. Sin embargo, Raspbian es el sistema operativo oficial de Raspberry Pi y su entorno es basado al entorno PIXEL (P).
Raspberry permite gestionar todo tipo de sensores a través de distintos protocolos de comunicación y puede ser utilizada en aplicaciones como: mini computadora de escritorio, para configurar un servidor Web o FTP, para crear un sistema NAS, para desarrollar un centro de entretenimiento, para fines educativos, aplicaciones en la robótica y domótica, por mencionar alguno.
Raspberry Pi, es compatible con los lenguajes de programación Python y C/C++, más sin en cambio, Python es el lenguaje mas utilizado por parte de la comunidad de Raspberry, por su fácil sintaxis y estructura.

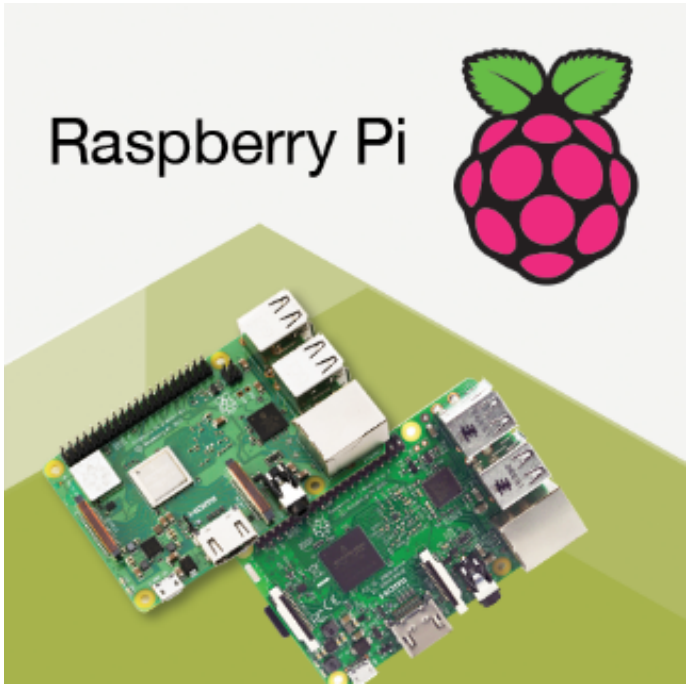


Figura 3.10: Imagen promocional de Raspberry Pi

La Raspberry Pi a través del sistema operativo Raspbian permite gran compatibilidad con lenguajes de programación como JavaScript, Python, C/C++ entre otros. Por otro lado, los desarrolladores han hecho más uso de los lenguajes Python y C debido a su fácil sintaxis y fama en el mundo del desarrollo. C++ es un lenguaje de programación híbrido creado por Bjame Stroustrop en el año de 1979. Gran parte de sus sintaxis es heredada del lenguaje C por lo que es compatible con C. A diferencia

del C, C++ soporta la programación orientada a objetos (Abstracción, Encapsulado, Herencia y Polimorfismo) pero tiene una pequeña limitación, y es que C++ tiene un grado de complejidad, que es necesario que el usuario que haga uso de C++ tenga ya experiencia programando. Python es un lenguaje de programación creado por Guido van Rossum en el año de 1991. Se caracteriza por ser un lenguaje de programación Interpretado, Multiplataforma y Orientado a objetos, a continuación se hablará con más profundidad de cada una de estas características:
Multiplataforma: A diferencia de Visual Basic, que es un lenguaje pensado solo para el desarrollo de aplicaciones en dispositivos y sistemas con Windows, Python permite desarrollar aplicaciones para una gran variedad de dispositivos y sistemas operativos debido que cuenta con interpretes para todos los sistemas que conocemos hoy en día tales como Linux, Windows y Mac Os.
Orientado a objetos: Python permite una programación estructurada y orientada a objetos, esta última conlleva una serie de ventajas y estándares gracias a la herencia, polimorfismo y otras características de la programación orientada a objetos. Otras de las ventajas con las que cuenta Python es que es de libre distribución y debido la gran popularidad que ha obtenido hasta ahora, se puede acceder a diferentes librerías y funciones hechas por la comunidad de Python, sin dejar atrás que soporta una múltiple variedad de bases de datos. Python ha ganado mucha popularidad en el mundo del desarrollo de software, ya que ha sido utilizado para la creación de aplicaciones como:

- Calibre: el mejor gestor de e-books para todos los usuarios.
- GNU MailMan: un programa para manejar listas de correo.
- BitTorrent: programa para compartir ficheros de tipo torrent estándar.
- Odoo (antes OpenERP): un ERP y mucho más para la gestión de empresas, de software libre.

De acuerdo con la IEEE Spectrum, Python ha sido el número uno en los lenguajes más utilizados para el desarrollo Web, sistemas embebidos y aplicaciones para las empresas y es en base a estos Ranking por el cual muchos usuarios y empresas se basan en el para el desarrollo de sus aplicaciones.

3.4.6. Dispositivos móviles

Un dispositivo móvil puede ser definido como un aparato de pequeñas dimensiones basado en una computadora de bolsillo, el cual tiene capacidades de procesamiento, con conexión a una red, con memoria interna limitada y que ha sido diseñado para realizar una función específica pero que es capaz de llevar a cabo una gran variedad de

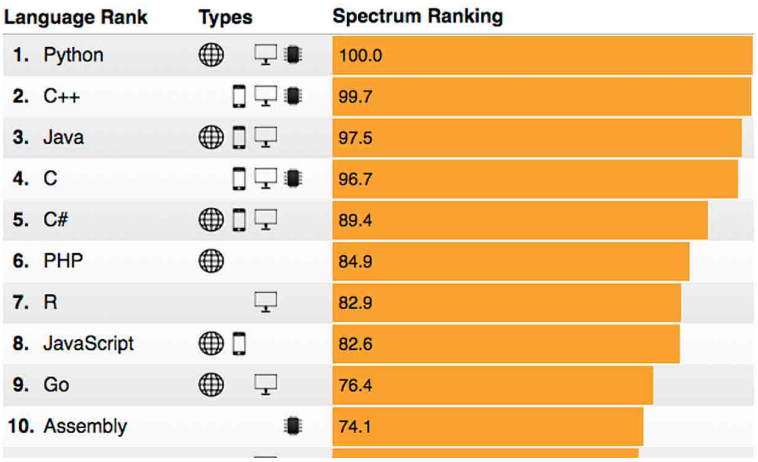


Figura 3.11: Ranking de popularidad 2018 sobre lenguajes de programación por la revista IEEE Spectrum

tareas adicionales. Hoy en día existen muchos tipos de dispositivos móviles entre los que destacan los teléfonos inteligentes, las tabletas, relojes inteligentes y los dispositivos utilizados para realizar aplicaciones del llamado internet de las cosas (IoT). Dentro de estas clasificaciones también pueden dividirse los dispositivos de acuerdo a su capacidad de procesamiento, sus dimensiones y servicios que puede proporcionar tales como cámara, conexión bluetooth, conexión wifi, sensores de movimiento, sensores biométricos, etc.

Teléfonos inteligentes

Un teléfono inteligente (smartphone en inglés) es un tipo de computadora portátil que combina características de una tableta con las de un teléfono celular. El término inteligente hace referencia a la capacidad que tienen estos dispositivos de realizar funciones propias de un computador portátil, llegando incluso a reemplazar a una computadora personal. Una de las características más importantes de un teléfono inteligente es el hecho de que permite la instalación de programas para incrementar la cantidad de tareas que puede realizar.

Tabletas

Una tableta (tablet en inglés) es una computadora portátil de mayores dimensiones que un teléfono inteligente formado por una pantalla táctil que permite interactuar con el dispositivo sin necesidad de un teclado físico ni ratón. El término de tableta puede aplicarse a una variedad de formatos, ya que si bien el formato estándar de una tableta suele ser de 7 a 12 pulgadas sin teclado físico (existiendo la posibilidad de conectar uno

por USB) también se fabrican versiones miniatura (de 7 a 8 pulgadas) y en formato portátil convertible la cual cuenta con un teclado físico que puede deslizarse debajo de la pantalla permitiendo utilizarlo como ordenador portátil o como tableta convencional.

Dispositivos enfocados al internet de las cosas

Dentro de la variedad de dispositivos móviles existentes se encuentran aquellos que tienen como finalidad realizar una tarea relacionada con el llamado internet de las cosas (IoT), estos dispositivos constan de un sistema de procesamiento formado por una microcomputadora, y los sensores y actuadores necesarios para llevar a cabo la tarea para la cual han sido diseñados, estos componentes pueden ser botones, sensores de temperatura, pantallas táctiles, entre muchos otros. Al igual que en las tabletas y los celulares inteligentes en los dispositivos dedicados al IoT también se pueden cargar nuevos programas para mejorar su funcionamiento o agregar nuevas tareas a desarrollar.

3.4.7. Java

Java es un lenguaje de programación de propósito general, orientado a objetos, así como también múltiples flujos de ejecución (multithreading). Java está diseñado para que los desarrolladores solo tengan que escribir el programa una vez y este pueda ser ejecutado en cualquier dispositivo, inclusive con diferente sistema operativo, de manera que este no tenga que ser recompilado. Java está basado principalmente en los lenguajes de programación C y C++ por lo que su sintaxis es muy similar utilizando también punto y coma (;) al final de cada sentencia. Claro está que Java cuenta con sus propias instrucciones, así como también sus propios tipos de variable, lo que le permite manejar de manera distinta los valores tanto de entrada como los de salida. Las ventajas de Java frente a otros lenguajes de programación son:

- Escribir el software en una plataforma y poder ejecutarla en cualquier otra.
- Crear programas que se pueden ejecutar en un explorador de internet y acceder a servicios Web.
- Combinar aplicaciones o servicios que utilizan Java para crear servicios o aplicaciones con un mayor grado de personalización.
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, microcontroladores, sensores, y casi cualquier dispositivo electrónico.

Debido a las ventajas ya explicadas Google decidió hacer de Java su lenguaje oficial de programación para aplicaciones en su sistema operativo Android por lo que las opciones restantes C++ y Kotlin fueron descartadas inmediatamente, C++ principalmente

porque Java permite utilizar mejor los recursos de los teléfonos móviles ya que posee una API (Application Progam Interface) diseñada especialmente para estos dispositivos, mientras que Kotlin al ser un lenguaje con el que se está menos familiarizado, fue que se optó mejor por Java.

3.4.8. Bluetooth

Fue creado por Bluetooth Special Interest Group, Inc. para la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz. Proporciona:

1. Libertad de derechos de autor de manera que está disponible para el público en general.

2. La capacidad de comunicación inalámbrica de corto alcance la cual permite comunicar diversos dispositivos a través de un medio compartido sin la necesidad de cables ni conectores.

3. Soporte para la transmisión de voz y datos haciéndolo una tecnología ideal para la comunicación entre distintos tipos de dispositivos.

4. El uso de una banda de frecuencias disponible en cualquier parte del mundo.

La tecnología Bluetooth se maneja como una red de área personal inalámbrica (WPAN-wireless personal area network), y cuenta con un estándar otorgado por la IEEE siendo este la IEEE 802.15.1.[8] De acuerdo al estándar 802.15.1, el Bluetooth se puede incluir dentro del modelo OSI específicamente las capas 1 y 2, que de otra forma se les conoce como capa Física y capa de Enlace de datos respectivamente. Aunque respecto a la capa de Enlace de datos, la tecnología Bluetooth solo ocupa una de las dos mitades de la capa, específicamente la subcapa MAC (Control de Acceso al Medio).

La trama Bluetooth se conforma de la siguiente manera: Respecto a la figura 3.12, los elementos de la trama se definen como:

- Slot: Es cada paquete enviado por el módulo.

- Código de Acceso: Es un código específico para la sincronización entre dispositivos Bluetooth. Normalmente es de 66 a 72 bits.

- Cabecera: Contiene datos necesarios para el direccionamiento del paquete. Parcialmente se conforma de 18 bits pero se repite 3 veces en el mismo paquete para asegurar el envió, por lo que llega a ser de 54 bits.

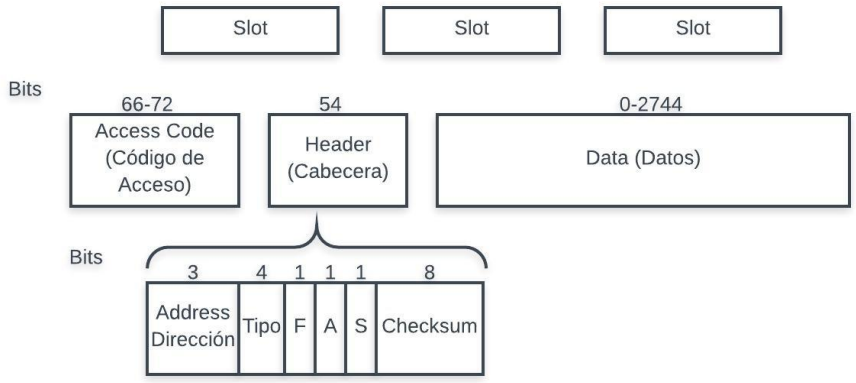


Figura 3.12: Trama Bluetooth.

- D irección(Address): Dirección MAC del dispositivo destino. Esta es única e irrepetible para cada módulo Bluetooth. 3 bits

- T ipo: Es el tipo de trama, y también se utiliza para corrección de errores y longitud del paquete. 4 bits.

- F : Bit de control de flujo. 1 bit

- A : acknowledgment o contestación. Sirve para verificar si llego correctamente el paquete.1 bit

- S : Numero de secuencia del paquete. Incluye el protocolo de parada y espera.1 bit

- C hecksum: Suma de verificación. Sirve para verificar si el paquete ha sido dañado o es el incorrecto. 8 bits

- Datos: Es la información a enviar ocupa de 0 a 2744 bits por paquete

3.4.9. Sistema de Base de datos

Un sistema de base de datos es un sistema computarizado para guardar registros, es decir, es un sistema computarizado cuya finalidad es almacenar información y permitir a los usuarios recuperar y actualizar la información con base en peticiones. La información puede ser de cualquier tipo desde números hasta palabras, inclusive direcciones, siempre y cuando la información de importancia para el individuo u organización a la que se destine el sistema [2].

Figura 3.13: Esquema de un base de datos.

Datos y modelado de datos

Los datos son la información que contienen los Sistemas de Base de datos y pueden ser manipulados de acuerdo a las necesidades del usuario. Estos datos se acomodan dentro de tablas por medio de filas y de acuerdo a un modelo de datos, el cual determina la estructura lógica de una base de datos, el cual además determina el modo de almacenar, organizar y manipular los datos [18].
Entre los modelos lógicos comunes para bases de datos se encuentran:

- Modelo jerárquico: Este modelo tiene como objetivo definir una jerarquía de fichas, de modo que cada ficha contenga a la vez listas de otras fichas y así sucesivamente. Una base de datos jerárquica está compuesta por una secuencia de bases de datos físicas, de manera que cada base de datos física se compone de todas las ocurrencias de un tipo de registro o ficha determinada.

- Modelo relacional: Este modelo busca representar la Base de datos como un conjunto de tablas, estableciendo una correspondencia entre varias tablas a través de los datos utilizando dos conceptos básicos los cuales son: -Tabla: Es un conjunto de registros de un mismo tipo. -Registro: Casilla de datos dentro de la tabla.

- Modelo en red: Este modelo es una combinación entre el modelo jerárquico y el relacional, ya que su estructura es parecida a la del modelo jerárquico aunque más compleja, pero gracias a eso se consigue evitar gran parte de los problemas del modelo jerárquico. Para definir el esquema de una base de datos en red se necesitan los siguientes conceptos:

  -Registro: Son las fichas almacenadas en un fichero.

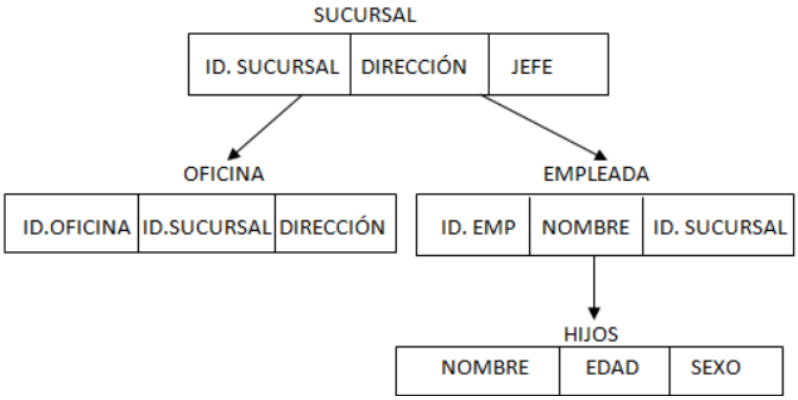


Figura 3.14: Modelo Jerárquico de Base de datos

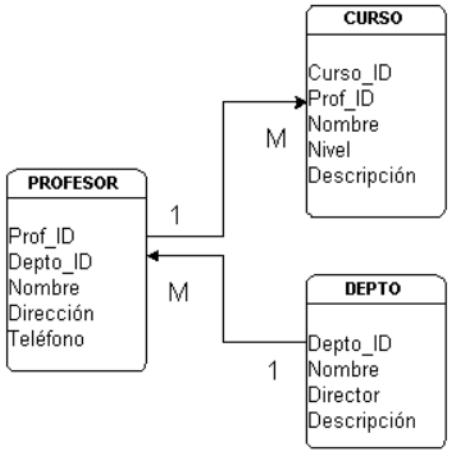


Figura 3.15: Modelo relacional de Base de datos

-Campos o elementos de datos: Son cada uno de los apartados de que se compone una ficha.

-Conjunto: Es el concepto que permite relacionar entre sí tipos de registro distintos.

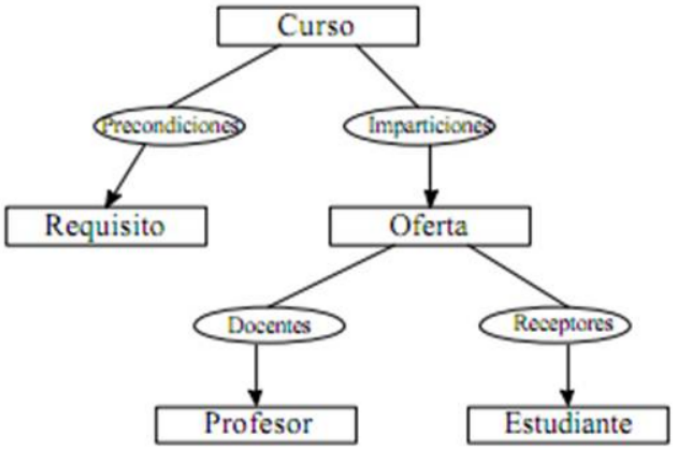


Figura 3.16: Modelo de Base de datos en red.

Sistema de Gestión de Base de datos

"Un Sistema de Gestión de Base de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a los mismos."
Un SGBD permite al administrador almacenar, manipular y consultar los datos pertenecientes a una Base de datos, la cual es un conjunto de tablas entre las que se establecen relaciones mediante los datos almacenados o a almacenar en las mismas. De lo anterior se puede decir que las SGBD se asemejan a las hojas de cálculo con la diferencia de que en las SGBD:

- El programa desde el que se hacen las consultas (el cliente) es independiente del método de almacenamiento y el programa de gestión de datos(servidor).
- El objetivo fundamental es permitir consultas complejas, expresadas en un lenguaje formal.
- El almacenamiento de los datos se hace de forma eficiente y está oculta al usuario, y a diferencia de las hojas de cálculo, no tiene que ver con la estructura en que estén los datos.

- Se permite el acceso de múltiples usuarios a los datos siempre y cuando esten autorizados, ademas de que pueden realizar diversas operaciones de actualización y consulta, evitando problemas de seguridad o de integridad de la información.

SQL

SQL es un lenguaje utilizado para el manejo de Base de datos y se define como Lenguaje de Consulta Estructurada, permite realizar consultas a Bases de datos, conseguir información, insertar nueva información, actualizar y/o borrar información. Inclusive permite crear nuevas Bases de datos, nuevas tablas, procedimientos almacenados, y también puede definir los permisos necesarios para las tablas, los procedimientos y las vistas sobre la Base de datos. SQL es un tipo de lenguaje general para diversos tipos de software que gestionan Bases de datos y aunque dependiendo del software que se utilice varían diferentes estructuras y comandos, en si, se mantienen como constantes los siguientes comandos:

- CREATE: Este comando permite crear bases de datos, tablas, vistas, procedimientos y objetos de datos.

- ALTER: Es comando permite modificar la estructura de una tabla u objetos, agregando o quitando campos, así como modificar el tipo de un campo, etc.

- DROP: Este comando permite eliminar un objeto de una base de datos. Como una tabla, un índice, una función, en sí esto depende del motor de la base de datos ya que estos soportan distintos tipos de objetos.

- TRUNCATE: Este comando es similar a DROP solo que TRUNCATE borra tablas de manera absoluta, lo que a diferencia de DROP en el cual puedes borrar secciones de la tabla.

- SELECT: Este comando permite consultar los datos almacenados en una tabla, ya sea toda la información o solo ciertos campos que obedezcan condiciones puestas por el administrador.

- INSERT: Este comando permite agregar uno o más registros a una tabla en una base de datos relacional.

- UPDATE: Este comando permite modificar los valores de un conjunto de registros.

- DELETE: Este comando borra uno o más registros de una tabla.

SQLite

SQLite es un motor de Base de datos, contenido en una biblioteca escrita con lenguaje C. El código de SQLite es de dominio público por lo que puede ser utilizado por cualquiera, y para cualquier tipo de proyecto sea público o privado. Su autor y desarrollador es D. Richard Hipp, y fue lanzado en agosto del año 2000.[17] La característica principal de SQLite es que no necesita de un servidor para realizar sus operaciones, si no que las realiza dentro del mismo dispositivo en el que se está ejecutando, por lo tanto, escribe y lee directamente en archivos de la memoria. Lo anterior permite el traspaso de los archivos entre sistemas operativos diferentes.

Dentro de las ventajas de SQLite existe el hecho de que el formato de transferencia de datos permite que el destinatario puede incluso realizar consultas al archivo sin tener que realizar cambios al archivo ni conversiones, esto utilizando lenguaje SQL, además de que los archivos son de tamaño reducido evitando en muchos de los casos el uso de un compresor de archivos. Además, se ve reducida la latencia para el acceso a la base de datos, ya que el programa que esté utilizando SQLite realiza llamadas a las funciones de SQLite en vez de realizar una comunicación entre procesos como ocurre con otros motores de base de datos. SQLite se enfoca en la eficiencia, independencia, simplicidad, economía y confiabilidad, que a diferencia de otros motores como MySQL, Oracle o SQL Server que se centran en escalabilidad, centralización, control y concurrencia, permite ser utilizado en dispositivos y aplicaciones que no requieran un uso experto de Base de datos, ni una administración constante sobre la misma por lo que funciona perfectamente para teléfonos móviles, televisores, juegos de video, cámaras, relojes, dispositivos médicos, y en especial con dispositivos relacionados al internet de las cosas. Esto es porque son dispositivos que no se encuentran conectados a la red todo el tiempo, aunque también puede trabajar con dispositivos con conexión casi permanente a la red.

SQLite es muy sencillo para gente que ya posee conocimientos en MySQL ya que utiliza líneas de comandos similares para el análisis de datos, la creación de reportes específicos, inclusive se pueden realizar consultas o configuraciones a la Base de datos más complejos, pero esto requiere el uso de códigos en otros tipos de lenguajes como por ejemplo Python. La versión más actual de SQLite es la versión 3.25.2 la cual ocupa hasta 275 kiB de memoria y puede soportar bases de datos de hasta 2 Terabytes de tamaño.

3.4.10. Lenguajes Web

Para crear una página Web asi como agregar diferentes características se necesita de diferentes lenguajes de programación orientados a páginas Web. para agregar funciones especificas

HTML

El lenguaje de marcado para hipertextos (Hyper Text Markup Language) mejor conocido como HTML, es el principal lenguaje de marcado del internet, originalmente fue pensado como un lenguaje para describir documentos científicos de manera semántica, sin embargo, su diseño general ha permitido que se haya podido adaptar para describir una gran variedad de tipos de documentos. Desde su lanzamiento en 1993 el lenguaje ha ido teniendo una gran variedad de versiones las cuales son reguladas por el World Wide Web Consortium (W3C) siendo la más reciente la versión 5.2 publicada el 14 de diciembre de 2017 [3].

Los documentos en HTML consisten de un árbol de elementos y textos, cada uno de estos elementos esta denotado por una etiqueta de inicio y una de término, a excepción de algunos casos particulares en los cuales estas etiquetas pueden ser omitidas. Un elemento puede estar contenido dentro de otro pero se debe tener cuidado con la correcta apertura y cierre de las etiquetas correspondientes al elemento, es decir, si se desea insertar un elemento dentro de otro las etiquetas de apertura y cierre del elemento interno deben colocarse siempre entre las etiquetas de apertura y cierre del elemento externo, en la figura 3.17 se ilustra la manera correcta en la que se deben colocar las etiquetas de un elemento individual y como se realiza la inserción de un elemento dentro de otro. En la figura anterior se muestra un ejemplo de cómo se utilizan las etiquetas

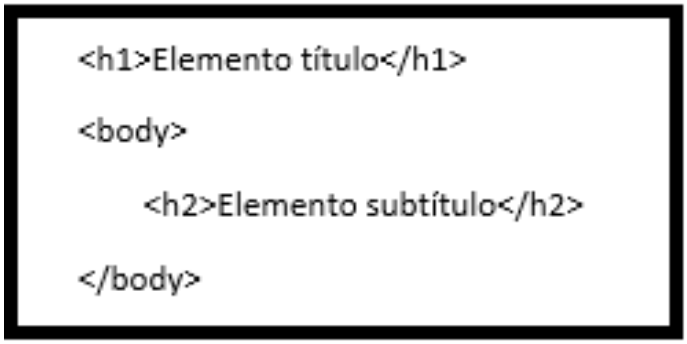


Figura 3.17: Etiquetas en HTML.

en HTML, se tiene una etiqueta de inicio de elemento "$< h1 >$" y su correspondiente etiqueta de cierre "$< /h1 >$", donde "h1" es el nombre de la etiqueta que se está utilizando, en este caso se trata de una etiqueta empleada en HTML para mostrar los títulos de una página Web, los símbolos "<>" son utilizados para diferenciar el nombre de la etiqueta que se desea utilizar del resto del texto presente en la página Web, es decir que el texto que se encuentre entre estos símbolos hará referencia al nombre de la etiqueta y sus propiedades (las cuales se explicaran más adelante), finalmente el carácter "/" indica que se trata de una etiqueta de cierre y el nombre del elemento que

se cierra va del lado derecho de la diagonal. Todo lo que se encuentre dentro de las etiquetas de apertura y cierre del elemento recibe el nombre de contenido del elemento, este contenido puede ser simplemente texto, como en el caso ilustrado en el elemento "h1" o pueden ser otros elementos, los cuales a su vez pueden contener texto u otros elementos, como en el caso del elemento "body" que contiene en si a un elemento "h2" el cual a su vez contiene texto, esta es la manera en que trabaja HTML mediante el uso de etiquetas para crear elementos de contenido. Además de las características ya mencionadas sobre las etiquetas en HTML existen otros atributos que estas etiquetas pueden tener, dichos atributos permiten controlar cual será el funcionamiento de ciertas etiquetas particulares como por ejemplo el caso del elemento de hipervínculo "$< a >$", este elemento permite agregar enlaces a otras páginas Web, para lo cual necesita el URL de la página a la cual se redireccionará, este URL será agregado al elemento mediante su atributo particular "href" tal y como se ilustra en la figura 3.18, de esta manera al seleccionar al elemento hipervínculo se podrá acceder a la página Web a la cual corresponda la dirección especificada. Al igual que el elemento de hipervínculo

Figura 3.18: Atributo "href" en elemento "a" (HTML).

"a" existen muchas otras etiquetas que requieren de ciertos atributos para funcionar correctamente, por lo cual cuando se desea utilizar un elemento HTML es necesario conocer cuáles son los atributos requeridos por dicho elemento en particular.

En HTML5 existe una gran variedad de elementos disponibles, por lo que es posible crear desde sitios Web que tengan simplemente elementos de texto, hasta complejas páginas de internet que permitan reproducir video, visualizar imágenes, llenar formularios de contacto, etc. Y si bien el número de elementos disponibles es bastante elevado todos ellos comparten una misma sintaxis (ver figura 3.19) lo cual facilita el trabajo de diseñar un sitio Web.

CSS

Las hojas de estilo en cascada (Cascading Style Sheets) mejor conocidas como CSS son el lenguaje empleado para definir la manera en que son presentadas las páginas Web, es decir se utilizan para definir el diseño, los colores y las fuentes del sitio Web. CSS también permite adaptar el diseño de la página Web a diferentes tipos de dispositivos, tales como dispositivos con pantalla grandes, medianas o pequeñas para brindar una buena experiencia al usuario que ingresa al sitio Web independientemente del dispositivo desde

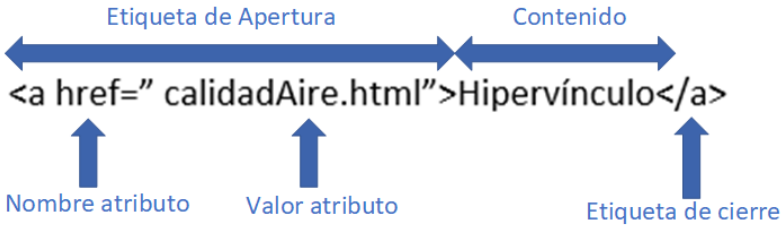


Figura 3.19: Estructura de un elemento en HTML.

el que haya ingresado. Las hojas de estilo en cascada son completamente independientes del diseño en HTML de la página Web, lo que facilita la tarea de dar mantenimiento al sitio Web, permitiendo también el compartir los diseños entre diferentes páginas Actualmente la versión más reciente que existe de CSS es la versión 3 y, al igual que en el caso de HTML la organización encargada de mantener la especificación CSS es el W3C. [5]

Como ya se mencionó, los estilos CSS son independientes del diseño en HTML, sin embargo es necesario vincular de alguna manera los estilos CSS con los elementos HTML del sitio Web, existen dos maneras de hacer esto, la primera de estas es insertar el código CSS en el archivo HTML al cual se desean aplicar los estilos, y la otra manera es vinculando el documento en el cual se encuentran todos los estilos CSS y el archivo HTML mediante el elemento "link", el cual permite a los elementos HTML acceder a los estilos CSS fácilmente, siendo esta segunda opción la más común al desarrollar páginas Web ya que al tener el código en HTML y lo estilos CSS en documentos separados se facilita la detección y corrección de errores asi como la implementación de actualizaciones. Para vincular la hoja de estilos CSS con el archivo HTML basta con colocar el elemento "link" e indicarle que lo que se desea vincular es una hoja de estilos, colocando también el nombre del archivo de la hoja de estilos con terminación ".css", tal y como se ilustra en la figura 3.20. Una vez que se han vinculado los estilos CSS

```
<link rel="stylesheet" href="css/estilos.css">
```

Figura 3.20: Vinculación de documento HTML con la hoja de estilos "estilos.css".

con el archivo en HTML se puede comenzar a modificar los estilos de los elementos de la página Web, para modificar los estilos de estos elementos es necesario utilizar algún identificador que permita indicar a que elemento o elementos serán aplicados los estilos CSS, estos identificadores reciben el nombre de selectores, si bien existe un gran número de selectores disponibles en CSS los más comunes están enlistados en la tabla 3.6. Cada selector tiene una forma particular de seleccionar a un elemento o elementos HTML,

Selector	Ejemplo	Descripción del ejemplo
class	.botón	Selecciona a todos los elementos con la clase "botón".
#id	#nombre	Selecciona al elemento con el id="nombre".
*	*	Selecciona todos los elementos.
Elemento	h1	Selecciona a todos los elementos $< h1 >$.
Elemento, elemento	div, p	Selecciona a todos los elementos ¡div¿y también a los elementos $< p >$
elemento elemento	*ul li*	Selecciona a todos los elementos $< li >$ que se encuentren dentro del elemento $< ul >$.

Tabla 3.6: Selectores más comunes en CSS.

tal y como se ejemplifica en la tabla anterior, por lo que, por ejemplo, si se desea dar el mismo estilo CSS a un grupo de elementos HTML diferentes lo más indicado será utilizar el selector ".class" ya que este permitirá asignarle el mismo estilo a todos los elementos que cuenten con esa clase indiferentemente de que sean diferentes tipos de elemento, mientras que si se busca darle el mismo estilo a un grupo de elementos HTML iguales lo más apropiado será utilizar el selector "element", algo a tener en cuenta es que si bien el selector "#id" permite seleccionar a un elemento en específico de la página Web, no suele ser empleado en la creación de estilos CSS más bien su uso queda reservado para la implementación de código JavaScript.

Los selectores de elemento no requieren ser asignados ya que el selector en si es el nombre del elemento por lo que bastara con nombrar al elemento en el documento CSS tal y como ilustra la figura 3.21, sin embargo, los demás selectores deben ser asignados al elemento al que se le desea aplicar los estilos CSS, esta asignación se realiza en la etiqueta de inicio del elemento como atributo y varía dependiendo de si se trata de un selector de clase o de ID, de igual manera en el archivo CSS la manera en que se declaran los selectores varía dependiendo del selector en sí, en la figura 3.21 se ilustra la manera en que se asignan los selectores a los elementos HTML y en la figura 3.22 como se representan en el archivo CSS. La asignación de estilos CSS a un elemento o grupo de elementos HTML se hace mediante el uso de reglas, las cuales son las declaraciones de los estilos que se desean aplicar. Una regla consta de dos partes un selector (puede ser cualquiera de los ya descritos anteriormente) y una declaración, la cual está conformada

```
<div class="clase-elemento" id="id-elemento">
    <h1>Elemento título</h1>
</div>
```

Figura 3.21: Asignación de selectores en HTML.

```
.clase-elemento{..}

#id-elemento{...}

h1{...}

div h1{...}
```

Figura 3.22: Selectores en CSS

por una propiedad y el valor que se le desea asignar, la figura 3.23 ilustra un ejemplo de una regla CSS que cambia el color de fondo de una página Web. Existe una gran variedad de propiedades disponibles pero todas siguen la misma estructura, por lo que bastara con revisar la documentación correspondiente a la propiedad que se desea modificar para conocer cuales son los valores que se le pueden asignar y de esta manera crear un sitio Web con la apariencia deseada.

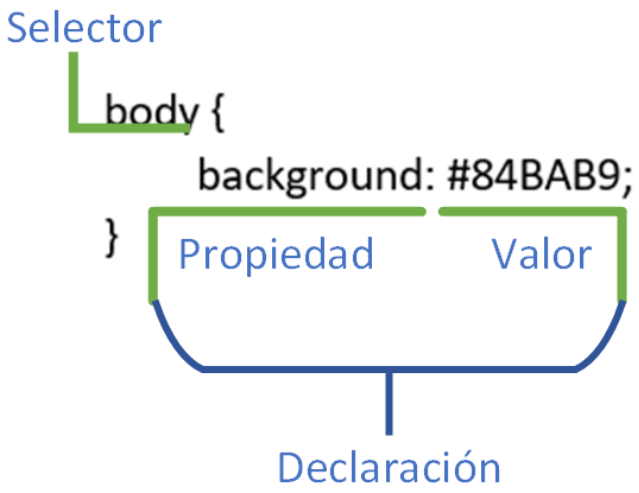


Figura 3.23: Estructura de una regla CSS.

Bootstrap

Bootstrap es una biblioteca empleada en el diseño de páginas y aplicaciones Web, está basada en los lenguajes HTML y CSS y contiene una gran variedad de plantillas de diseño para los diferentes elementos de una página Web, tales como formularios, botones, menús de navegación entre otros, también cuenta con extensiones de JavaScript lo que permite crear sitios Web más dinámicos. Su versión estable más reciente es Bootstrap 4.0.0 la cual fue lanzada el 18 de enero de 2018, la cuál es compatible con HTML5 y CSS3 [1].

Su función es agilizar el proceso de asignación de estilos CSS a los elementos HTML, ya que, como se mencionó anteriormente, cuenta con un conjunto de plantillas que permite establecer estilos CSS predefinidos a elementos HTML mediante selectores de clase, lo cual reduce bastante el tiempo de diseño de una página Web. Para utilizar las plantillas de Bootstrap es necesario descargar de la página oficial de Bootstrap, la versión que se desee utilizar, cada versión incluye un conjunto de archivos de estilo CSS los cuales contienen todas las reglas CSS proporcionadas por Bootstrap y de

igual manera incluyen un conjunto de archivos JavaScript, los cuales contienen los códigos empleados para proporcionar dinamismo al sitio Web, estos archivos deben ser vinculados con el archivo HTML de la página Web mediante el elemento "link", tal y como se hace con los archivos CSS y mediante el uso del elemento "src" para el caso de los archivos de JavaScript, en la figura 3.24 se indican los archivos de Bootstrap que deben ser vinculados. Una vez vinculados estos archivos las plantillas de Bootstrap estarán disponibles. La característica principal de Bootstrap es su sistema de

```
<link rel="stylesheet" type="text/css" media="screen href="css/bootstrap.min.css"/>
<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

Figura 3.24: Vinculación de Bootstrap con un archivo HTML.

cuadricula (Grid en inglés), el cual permite dividir a la página Web en filas y columnas, lo que brinda a la página de una mejor distribución de sus elementos. Este sistema de cuadricula cuenta con un número máximo de 12 columnas, es decir un elemento puede ocupar un espacio pequeño en la página Web (1 columna) u ocupar todo el ancho disponible del sitio Web (12 columnas), esta característica de Bootstrap es de gran ayuda cuando se desea realizar páginas Web responsivas, es decir, sitios Web que cambien su apariencia dependiendo del tamaño de pantalla del dispositivo desde el cual se está consultando, lo cual permite brindar una buena experiencia al usuario tanto si consulta el sitio Web desde una computadora como si lo hace desde un smartphone. Para convertir a un elemento HTML (generalmente el elemento "div") en una fila basta con agregar la clase "row" al elemento, una vez que se cuenta con un elemento fila se le pueden agregar los elementos columna, la manera de darle a un elemento las propiedades de columnas es igual que en el caso de las filas, basta con agregar la clase "col" al elemento. Como ya se mencionó las columnas en Bootstrap pueden tener diferentes tamaños, dependiendo de las necesidades del desarrollador, para establecer el tamaño de una columna basta con agregar a la clase "row" el número de columnas que se desea asignar al elemento separado por un guión tal y como se muestra en la figura 3.25, al no indicar de qué tamaño será la columna se tomara el valor por defecto que es 12, además se debe tener en cuenta que el número máximo de columnas por fila es de 12. También se ha comentado que Bootstrap puede ser utilizado para crear páginas Web responsivas, ya que permite variar el tamaño de sus elementos de acuerdo con el tamaño de la ventana del explorador, para hacer esto Bootstrap cuenta con clases que se encargan de proporcionar ciertos estilos a un elemento dependiendo del tamaño de la ventana del explorador, es decir, por medio de estas clases se le puede asignar un estilo aun elemento cuando se consulta la página desde un smartphone y asignarle otro estilo diferente al mismo elemento cuando se consulta la página desde

```
<div class="row"> <!--Elemento fila-->
   <div class="col"></div> <!--Elemento columna de tamaño 12-->
</div>
<div class="row">
   <div class="col-6"></div> <!--Elemento columna de tamaño 6-->
   <div class="col-3"></div> <!--Elemento columna de tamaño 3-->
   <div class="col-2"></div> <!--Elemento columna de tamaño 2-->
   <div class="col-1"></div> <!--Elemento columna de tamaño 1-->
</div>
```

Figura 3.25: Dos filas con Bootstrap con sus elementos columna.

una computadora de escritorio, las encargadas de realizar este cambio de estilos son las clases "xs" para el caso de pantallas muy pequeñas, "sm" para pantallas pequeñas, "md" para pantallas medianas, "lg" para pantallas grandes y "xl" para extra grandes. Si bien estas clases pueden ser utilizadas de muchas maneras, uno de sus usos más comunes es el de variar el tamaño de columnas de manera que el sitio Web tenga buena presentación independientemente de donde se le consulte, también está la posibilidad de hacer visible un elemento solo para ciertos tamaños de la ventana del explorador, este último caso se ilustra en la figura 3.26 el cual también es frecuentemente utilizado para dar un funcionamiento responsivo a un sitio Web. Si bien el sistema de cuadricula

```
<div class="row">
        <div class="col columna d-block d-sm-none">XS - Extra small</div>
        <div class="col columna d-none d-sm-block d-md-none">SM - Small</div>
        <div class="col columna d-none d-md-block d-lg-none">Md - Medium</div>
        <div class="col columna d-none d-lg-block d-xl-none">Lg - Large</div>
        <div class="col columna d-none d-xl-block">XL - extralarge</div>
</div>
```

Figura 3.26: Columnas con propiedades responsivas.

es la característica fundamental de Bootstrap existen muchas otras posibilidades de estilos que esta biblioteca puede proporcionar, tales como colores, estilos de elementos, animaciones y muchos otros elementos preestablecidos, lo cual permite reducir los tiempos de creación de un sitio Web y obtener diseños estéticos sin tener que crear demasiadas reglas CSS.

JavaScript

JavaScript generalmente abreviado como JS es un lenguaje de programación orientado a objetos, es ampliamente conocido por ser el lenguaje de programación utilizado en la creación de páginas Web, generalmente del lado del cliente, lo cual se refiere al entorno de un sistema cliente-servidor en el cual el usuario realiza sus operaciones siendo este típicamente la computadora del usuario, JS fue desarrollado por Netscape y está basado en el estándar ECMAScript [6]. Fue lanzado en 1995 y su última versión estable fue publicada el 17 de junio de 2016. Comúnmente se suele asociar a JS con el lenguaje de programación Java, sin embargo, ambos lenguajes tienen bastantes diferencias en sintaxis, semántica y usos.

La sintaxis de JS es muy semejante a la de otros lenguajes de programación tales como C y Java, ya que, al igual que en estos lenguajes de programación, en JS no se tienen en cuenta los espacio en blanco y las nuevas líneas, lo que permite codificar programas de manera ordenada para facilitar su lectura, esto mediante tabulaciones, creación de nuevas líneas, etc. Otra característica de JS es que existe distinción entre letras mayúsculas y minúsculas, por lo que una variable llamada "var1" será completamente diferente a una llamada "Var1", por lo que se debe tener cuidado de respetar el nombre asignado a una variable durante el resto del programa. Una diferencia notable entre JS y otros lenguajes como Java y C es el hecho de que al declarar una variable en JS no es necesario definir el tipo de la variable en cuestión, por lo que una misma variable podrá almacenar diferentes tipos de datos durante la ejecución del programa. Algo que también se debe tener en cuenta es el hecho de que JS no es necesario incluir el carácter punto y coma ";" al final de cada sentencia, como en los lenguajes de programación previamente mencionados, sin embargo, a pesar de que no es obligatorio el uso del carácter ";" se suele seguir colocando al final de las sentencias de manera que no cause conflicto para los programadores que suelen programar en aquellos lenguajes donde su uso si es obligatorio.

Si bien con la aparición de las aplicaciones AJAX (Asynchronous Java Script And XML en español JavaScript asíncrono y XML) JavaScript ha obtenido una gran popularidad dentro de los lenguajes de programación Web, presenta una serie de limitaciones que hay que tener en cuenta si se desea trabajar con este lenguaje de programación, entre estas limitaciones se encuentra el hecho de que los scripts de JS no pueden establecer comunicación con recursos que no pertenezcan al mismo dominio del cual se descargó el script. Un script de JS tampoco puede tener acceso a los archivos alojados en la computadora del usuario y, en caso de que la ejecución de un script se prolongue por demasiado tiempo el navegador informará al usuario de que dicho script está demandando demasiados recursos y le permitirá cancelar la ejecución del script. Sin embargo, existe la posibilidad de eliminar estas limitaciones, esto se hace firmando

digitalmente el script y solicitando al usuario que proporcione permiso para realizar acciones que de otra forma no podrían ser ejecutadas.

Al igual que ocurre con los estilos CSS los scripts de JS deben ser vinculados con el archivo principal de la página Web, para hacer esto se hace uso del elemento HTML "script", al cual se le indica que archivo de scripts es el que se desea vincular mediante su atributo "src" tal y como se ilustra en la figura 3.27. Una vez hecho lo anterior el

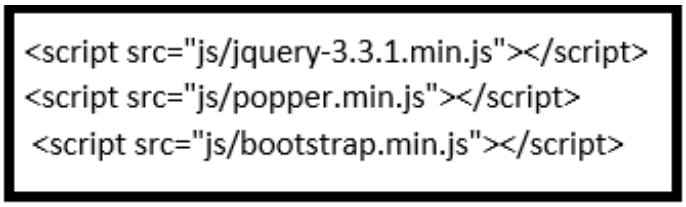


Figura 3.27: Vinculación de scripts de JS con el archivo principal de la página Web.

script de JS podrá interactuar con los elementos HTML de la página Web.

jQuery

jQuery es una biblioteca de JavaScript que permite interactuar con los documentos HTML de una manera más sencilla, de igual manera permite la manipulación del modelo de objetos para la representación de documentos (DOM por sus siglas en ingles), también permite crear sitios Web más dinámicos gracias al desarrollo de animaciones y mediante la técnica AJAX. Esta biblioteca fue desarrollada por John Resign y presentada el 26 de agosto de 2006, su última versión estable fue lanzada el 20 de marzo de 2017. Cuenta con la licencia pública general de GNU, por lo cual puede ser utilizada tanto en proyectos libres como en privados. El uso de esta biblioteca permite simplificar el desarrollo de funciones en JavaScript que, de no ser por esta biblioteca requerirían de una gran cantidad de tiempo y líneas de código para su desarrollo. La característica principal de jQuery es el hecho de que permite realizar cambios en el contenido de una página Web sin que sea necesario actualizar la página. [7]

PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de programación de código abierto utilizado del lado del servidor, lo que quiere decir que, a diferencia de lenguajes ejecutados en el lado del cliente, como es el caso de JavaScript, el código PHP es ejecutado del lado del servidor, de esta manera el cliente recibirá aquello que haya resultado de la ejecución del script pero no tendrá conocimiento acerca del código del script. PHP apareció en 1995 y su última versión estable (la versión 7.2.10)

fue lanzada el 13 de septiembre de 2018. [14] Una característica fundamental de PHP es el hecho de que puede ser incrustado dentro de HTML, es decir que dentro de un archivo HTML se puede incrustar directamente código PHP en lugar de tener que llamar a un archivo externo que realice el procesamiento de los datos, esto se hace mediante el uso de las etiquetas especiales de comienzo y final de PHP "¡?php" y "?¿", tal y como se ilustra en la figura 3.28. Las actividades más comunes para las que es utilizado PHP

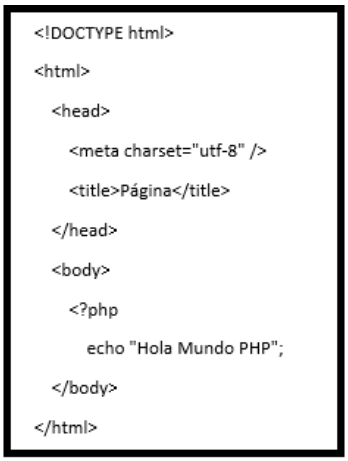


Figura 3.28: Inserción de código PHP en archivo HTML.

son la recopilación de formularios, la generación de páginas con contenidos dinámicos y el envío y recepción de cookies, aunque puede ser utilizado en otras tareas, el uso más común es la vinculación de una página Web con una base de datos, generalmente en MySQL, ya que mediante el uso de bases de datos se puede recopilar información del usuario del sitio Web así como se le puede proporcionar información almacenada en las bases de datos con las cuales está vinculada la página Web, un ejemplo de esto es la creación de cuentas de usuario, las cuales requieren de información del usuario, la cual es recopilada por PHP a través del uso de formularios, y de este modo, un usuario registrado podrá acceder a información que no esté disponible para usuarios sin cuenta y esto también se hace a través de PHP. Para que PHP funcione del lado del servidor es necesario cumplir con tres requisitos, primero se debe contar con un servidor PHP, un servidor Web y un navegador Web. Para poder realizar la ejecución de los scripts de PHP es necesario tener una conexión con el servidor, si se cumple con estos requisitos se podrá visualizar el resultado de un programa PHP a través de un navegador de internet consultando la página a través del servidor.

XAMPP

XAMPP (X: para cualquier sistema operativo, Apache, MySQL, PHP, Perl) es una distribución de Apache gratuita, la cual contenía el gestor de base de datos MySQL hasta su versión 5.6.15 en la cual cambio a este gestor por MariaDB el cual es una bifurcación de MySQL, así mismo cuenta con los intérpretes para los lenguajes de programación PHP y Perl. La última versión estable es la 7.2.9 la cual fue lanzada el 27 de agosto de 2018. Este programa actúa como un servidor Web libre el cual busca ser fácil de utilizar y, aunque en un principio fue concebido solamente como una herramienta que le permitiera a los diseñadores y desarrolladores Web probar el trabajo que habían realizado en sus propios equipos de cómputo sin la necesidad de tener acceso a internet, hoy en día XAMPP es utilizado como servidor de sitios Web reales, ya que proporciona los niveles suficientes de seguridad.

3.4.11. Servidor

Un servidor es un dispositivo informático que provee los datos solicitados por parte de otros equipos de cómputo o en términos de redes es aquel que facilita el acceso a la red y a sus recursos, en la actualidad podemos encontrar diferentes tipos de servidores como los servidores de bases de datos, servidores Web, etc. y cada uno de ellos brinda acceso a la información a todos los clientes que quieran acceder.

Capítulo 4

Desarrollo

Este capítulo presenta el procedimiento realizado para llevar a cabo el prototipo para el sistema de medición de variables ambientales que afectan la calidad del aire. El prototipo esta dividido en 5 etapas las cuales se muestran en la figura 4.1. Los

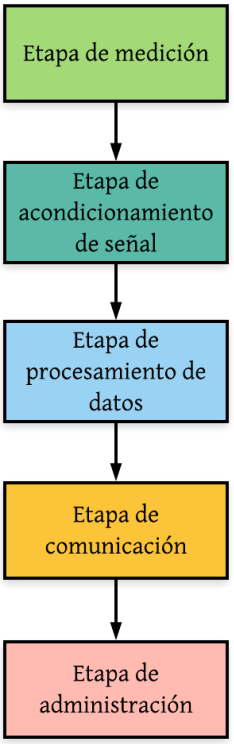


Figura 4.1: Diagrama a bloques del sistema.

siguientes apartados explican detalladamente el procedimiento realizado para llevar a cabo el prototipo.

4.1. Etapa de medición

La primera etapa del prototipo corresponde a la caracterización del comportamiento de los sensores, de la marca Alphasense, esto para identificar cual es el mayor valor de voltaje que puede llegar a entregar, y observar como se comporta, en cuanto a valores de voltaje, al ser energizados. Para obtener los datos anteriormente expresados son necesarios los siguientes materiales:

- 2 Voltímetros de CD (corriente directa).
- Fuente de alimentación de 5 Volts y mínimo 35mA.
- Sensores Alphasense A4 modelo 810-0023-00, con placa AFE (Analogue Front-End) de 12 pines.

El primer paso para caracterizar el comportamiento de los sensores es conectar ambos voltímetros a los sensores, uno de ellos debe conectarse a la terminal del electrodo de trabajo de uno de los sensores definido por el nombre SNX (X es el número del sensor al que corresponde la terminal), mientras que el otro voltímetro se conecta a la terminal del electrodo auxiliar del mismo sensor. El segundo paso es energizar la placa AFE con la fuente de alimentación, y observar cuál es el voltaje inicial que entrega el sensor en cuestión, pero con el paso del tiempo este valor irá disminuyendo, a este tiempo en el que el voltaje del sensor disminuye paulatinamente se le llama tiempo de estabilización.

El tiempo de estabilización termina cuando los valores de voltaje se mantienen constantes, durante las pruebas se observó que en cualquiera de los casos los 4 sensores inicialmente entregan un valor de voltaje igual al de la fuente de alimentación, y además los 4 en aproximadamente una hora, terminan su tiempo de estabilización y pueden ser utilizados con normalidad sin el riesgo de lecturas erróneas debido a los efectos transitorios de la fuente de alimentación. Después se realiza una prueba de cuál es el máximo valor que se puede obtener respectivamente de cada uno de los 4 sensores, en su mayoría estos provienen de la quema de combustibles fósiles así que se utilizó un automóvil para realizar las pruebas. Durante las cuales se observó que en el caso del sensor de CO (Monóxido de Carbono) y SO_2 (Dióxido de Azufre), sus valores de voltaje pueden llegar a alcanzar el valor máximo (el voltaje de la fuente) debido al aumento de las revoluciones del motor del automóvil, mientras que el sensor de O_3 (Ozono) muestra cambios dependiendo de el nivel de calor y de luz solar, aunque cabe mencionar que durante las pruebas realizadas el valor de voltaje obtenido no superaba

los 2 volts. En cuanto al sensor de NO_2 (Dióxido de Nitrógeno), las pruebas mostraron que el máximo valor de voltaje alcanzado es de 2 volts. Tomando en cuenta estos datos, y las características del ADC es que se ha decidido implementar una etapa de acondicionamiento de señal con la cual se busca evitar ocasionar algún daño al ADC ya que este soporta como máximo ±2,048 volts a la entrada.

4.2. Etapa de acondicionamiento de señal

Tomando en consideración que los sensores pueden entregar hasta 5 Volts a la salida, y que el ADC Differential Pi solo soporta como máximo 2.048 Volts a la entrada de sus canales de lectura, es necesario incluir una forma de proteger el ADC. Por esto mismo se desarrolló una etapa de acondicionamiento de señal, conformada por divisores de voltaje que disminuyen el nivel máximo de voltaje de la salida de los sensores a 1.8 volts como máximo, lo cual permite manipular con seguridad el dispositivo sin correr el riesgo de dañar sus componentes. Para el diseño del divisor de voltaje se utilizó la siguiente ecuación:

$$V_{out} = V_{in} * \frac{R_2}{R_1 + R_2} \tag{4.2.1}$$

Respecto a la ecuación anterior, y analizando la figura 4.2, se sabe que V_{in} debe ser igual a 5 volts, V_{out} deberá de ser igual a 1.8 volts, por lo que se propone que R_2 tenga un valor de 1kOhm. De esta manera solo se tiene que calcular R_1. Por lo que la ecuación

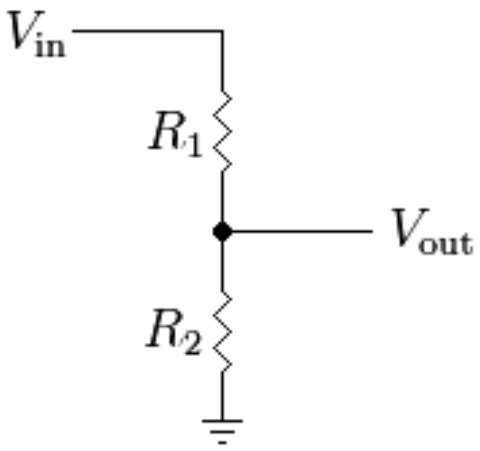


Figura 4.2: Esquema básico de un divisor de voltaje

ya despejada respecto a R_2 queda de la siguiente manera:

$$R_1 = R_2 * (\frac{V_{in} - V_{out}}{V_{out}}) \tag{4.2.2}$$

De modo que al sustituir los valores respectivos se obtiene:

$$R_1 = 1000[Ohm] * (\frac{5[V] - 1{,}8[v]}{1{,}8[V]}) = 1777{,}77[Ohm] \tag{4.2.3}$$

Por lo tanto, el circuito con los valores de resistencia sustituidos queda como en la siguiente figura: Un inconveniente del resultado obtenido es que la resistencia R_1 no

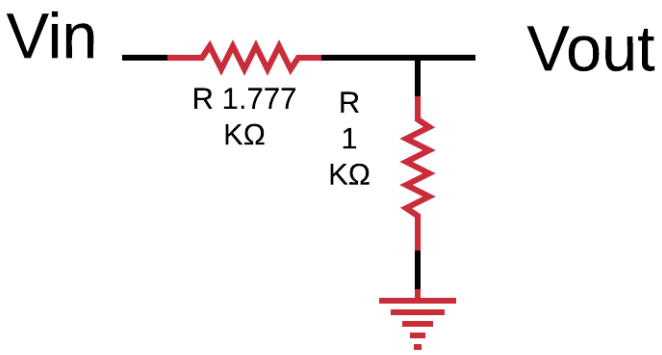


Figura 4.3: Esquema eléctrico del divisor de voltaje del prototipo

es de valor comercial, por lo que es necesario realizar un arreglo de resistencias para obtener ese valor, el cual queda de la siguiente manera: Una vez terminados los cálculos

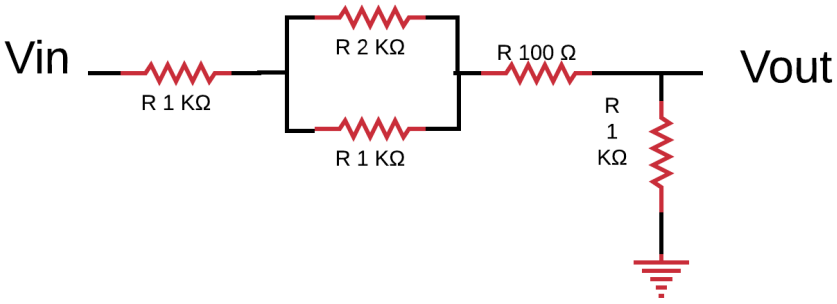


Figura 4.4: Esquema eléctrico final del divisor de voltaje del prototipo.

se hace el diseño de la placa en PCB para implementarlo en el prototipo final. El circuito final montado en PCB queda de la siguiente manera:

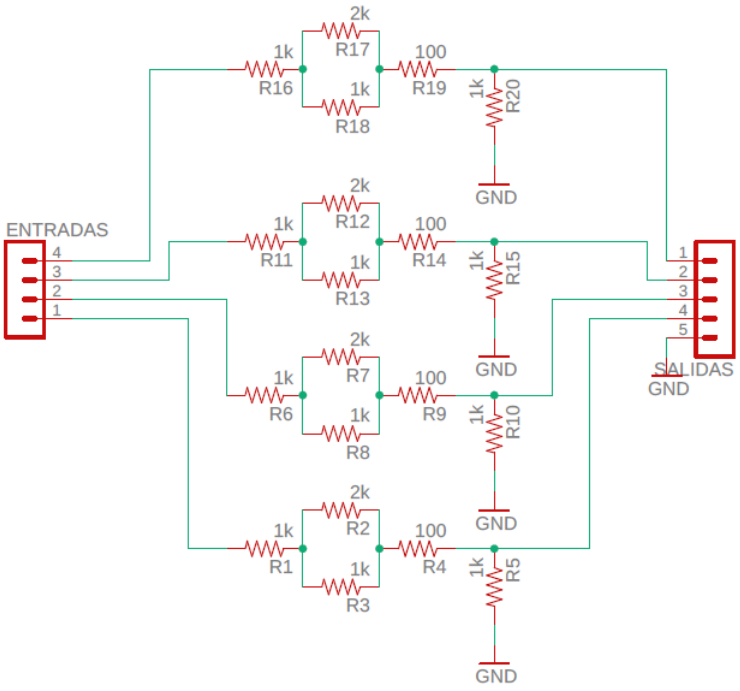


Figura 4.5: Esquema eléctrico de los 4 divisores de voltaje para PCB.

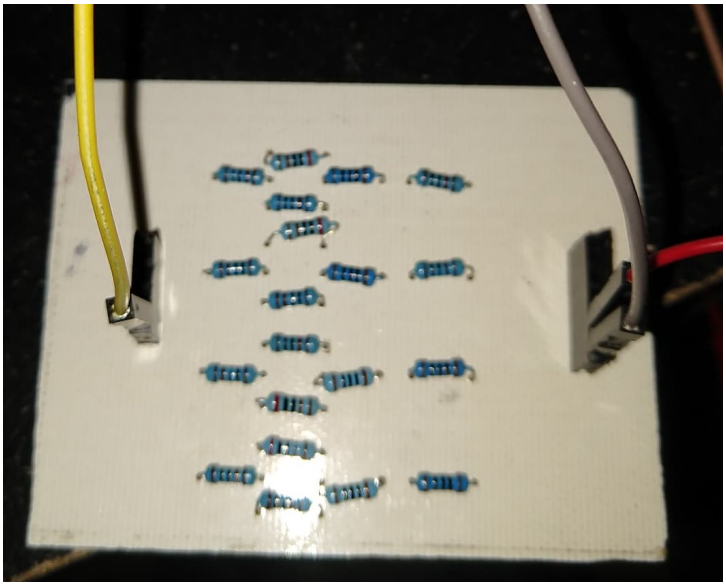


Figura 4.6: PCB con 4 divisores de voltaje.

4.3. Etapa de procesamiento de datos

Esta sección explica el procedimiento realizado para la configuración de la Raspberry Pi, asi como del algoritmo y las fórmulas utilizadas para el procesamiento de los datos entregados por los sensores de componentes tóxicos.

4.3.1. Instalación de Raspbian en Raspberry Pi 3

En este apartado se explica el proceso de instalación del sistema operativo Raspbian en la Raspberry Pi 3 en una serie de pasos.

Materiales necesarios para la instalación y configuración de Raspbian:

- 1 teclado
- 1 mouse
- 1 monitor
- 1 memoria micro SD (De preferencia de 8GB, clase 10)
- 1 adaptador de tarjetas SD a USB
- Win32DiskImage
- Acceso a internet

Paso 1: Acceder a la página oficial (`https://www.raspberrypi.org/downloads/raspbian/`) de Raspberry, específicamente en el apartado de downloads para la descarga del S.O. Raspbian. Como se ve en la figura 4.7, se tienen dos formas de instalación: La primera que es NOOBS, la cuál es una versión completa de Raspbian que puede ser instalada en una tarjeta SD sin la necesidad de una conexión a internet, La segunda opción es NOOBS LITE, que requiere de una conexión a internet por lo que la decisión de elegir uno de los dos archivos ya dependerá del usuario.

Paso 2: Al concluir la descarga de Raspbian se procede a descomprimir el archivo donde se puede observar una serie de librerías que serán almacenadas en la memoria micro SD, se recomienda que la memoria cuente con una capacidad mínima de 8 GB y sea de clase 10. Paso 3: Formatear la Memoria, en la cuál se cargara el sistema operativo Raspbian (se puede utilizar cualquiera de los dos formatos disponibles).
Paso 4: Copiar el contenido resultante de la descompresión de la descarga en la Memoria SD ya formateada.
Paso 5: Insertar la tarjeta SD en el puerto correspondiente de la Raspberry, después

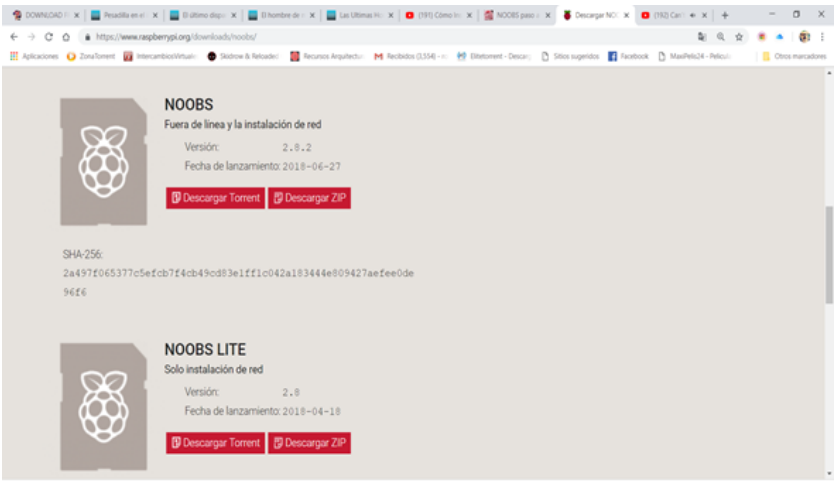


Figura 4.7: Apartado de descargas de la página oficial de Raspberry

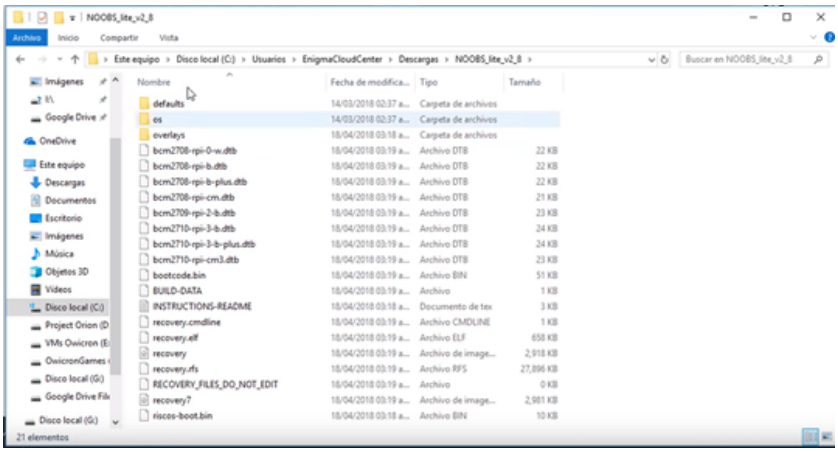


Figura 4.8: Bibliotecas de Raspbian.

conectar todos sus periféricos (HDMI, cable de red, mouse, teclado y cable de alimentación).
Paso 6: Conectar el cable de video al monitor, y el cable de alimentación a la toma de corriente, para encender la Raspberry. Si hasta ahora el procedimiento se ha realizado correctamente se observara en el monitor lo mismo que en la figura 4.9
Paso 7: Acceder a la terminal e ingresar el comando **sudo apt-get update** para

Figura 4.9: Primer inicio del Sistema Raspbian

descargar las listas de paquetes de los repositorios y actualizarlas, para asi obtener información sobre las versiones más recientes de paquetes y sus dependencias. Paso 8: Una vez terminada la búsqueda de paquetes, ingresamos **sudo apt-get upgrade** para la actualización del sistema y cada una de sus aplicaciones instaladas.

4.3.2. Proceso de instalación de los paquetes necesarios para la comunicación Bluetooth

Debido a que el proyecto requiere de una comunicación Bluetooth con un dispositivo Android, aquí se describen los primeros pasos esenciales para la descarga de la librería Bluetooth para Python.

Primero es necesario actualizar el repositorio de datos de la Raspberry con los comandos:

- *sudo apt-get update*
- *sudo apt-get upgrade*

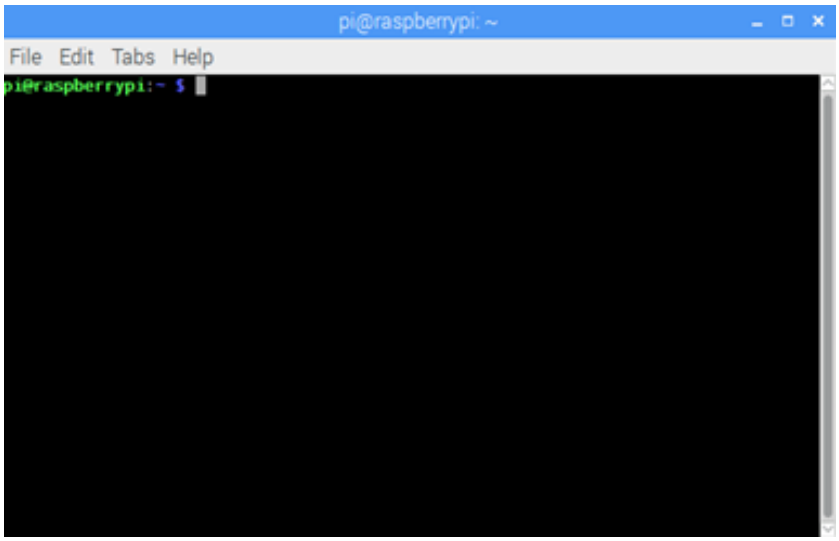


Figura 4.10: Terminal de Raspbian

Después se deben instalar los paquetes que permiten iniciar el Bluetooth de la placa Raspberry, mediante el comando *sudo apt-get install bluetooth blueman bluez*. Donde *bluez* es un archivo de código abierto y una pila oficial de protocolos Bluetooth en Linux, y *blueman* es una interfaz gráfica que permite administrar y controlar el dispositivo Bluetooth. Una vez instalado *blueman* y *blues* se debe reiniciar el dispositivo Raspberry.

Para finalizar se instala la librería Bluetooth para Python, para poder y enviar y recibir datos a través de RFCCOM, esto con el comando *sudo apt-get install python-bluetooth*.

Posteriormente se muestra como establecer una conexión Bluetooth con Android, para ello abrimos la terminal en Raspbian e ingresamos los siguientes comandos:

- sudo bluetoothctl
- $[bluetooth]\#poweron$
- $[bluetooth]\#agenton$
- $[bluetooth]\#discoverableon$
- $[bluetooth]\#pairableon$
- $[bluetooth]\#scanon$

Con el comando **pair** "< *Dirección MAC del celular* >" se ingresa la dirección MAC del dispositivo con el cual se quiera conectar, para emparejar ambos dispositivos, es importante asegurar que el dispositivo móvil tenga el Bluetooth encendido y visible.

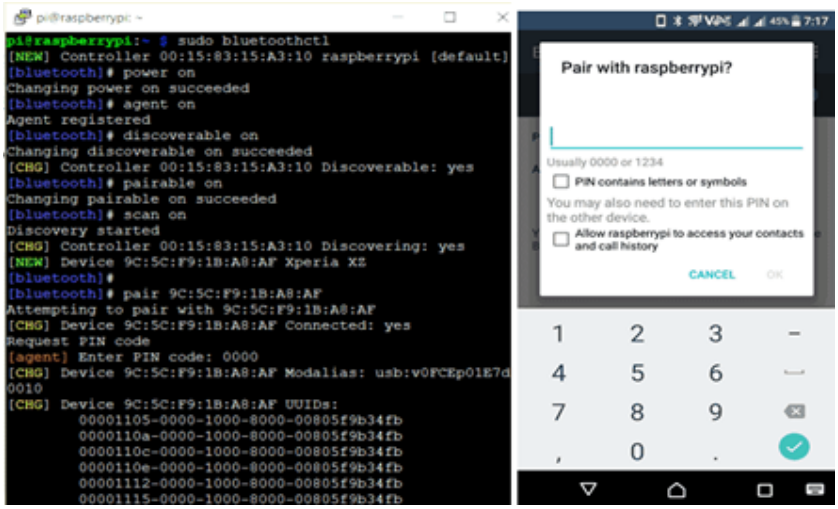


Figura 4.11: Sincronización del dispositivo Android con Raspberry Pi 3

4.3.3. Etapa de Procesamiento de datos

En el siguiente apartado se explican las líneas de código utilizadas para procesar los datos obtenidos por los sensores de gases para que posteriormente sean enviadas a la aplicación en Android.

PyBluez es una biblioteca de Python Bluetooth que cuenta con dos diferentes configuraciones "Cliente o Servidor", el primero actúa como el dispositivo que solicita la conexión y el segundo como el dispositivo que acepta la conexión. usando el protocolo "RFCOMM". Para este caso se usó la configuración servidor debido a que el usuario decide por medio de la aplicación en Android cuando tomar una medición y el driver simplemente obedece a las peticiones del usuario.

Para iniciar la configuración como servidor, se construye un socket para el servicio RFCOMM (ver figura 4.12), el servidor enlaza el script en el host aceptando una sola conexión a la vez (recibiendo 1024 caracteres como máximo) por el puerto 1. Una vez realizada la configuración de la Raspberry como servidor, es necesario implementar la biblioteca ."ADCDifferentialPi"(ver figura 4.13), está se puede obtener de la página Web del fabricante (ABelectronic). Es mediante esta biblioteca que se facilita la lectura de los 8 canales del ADC, utilizando el protocolo I2C el cual ya también viene implementado en la librería. Para este paso es importante agregar un código de error con el cual evitar

```
server_socket=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
server_socket.bind(("",1))
server_socket.listen(1)
client_socket,address = server_socket.accept()
print("Accepted connection from",address)
```

Figura 4.12: Configuración de servidor para la Raspberry Pi

```
from ADCDifferentialPi import ADCDifferentialPi
except ImportError:
    print("Failed to import ADCDifferentialPi from python system path")
    print("Importing from parent folder instead")
    try:
        import sys
        sys.path.append('..')
        from ADCDifferentialPi import ADCDifferentialPi
    except ImportError:
        raise ImportError(
            "Failed to import library from parent folder")
```

Figura 4.13: Código para la importación de la biblioteca ADCDifferentialPi.

que el programa detenga su ejecución de manera que también lo busque en alguna otra parte del sistema.

Para realizar la lectura de los canales del ADC se utiliza la siguiente función: Esta

```
SN1W=adc.read_voltage(1)#channel 1
SN1A=adc.read_voltage(2)#channel 2
```

Figura 4.14: Instrucción en Python para la lectura de voltajes con el ADC.

función solo requiere de el número del canal en el cual se va a realizar la lectura. (Los canales van ordenados del 1 al 8 y de derecha a izquierda como se muestra en la figura 4.15)

Una vez almacenados los valores del ADC, es necesario aplicar la siguiente fórmula para convertir los valores de voltaje obtenidos en valores de ppb.

$$Concentracion = \frac{(OP_1 - Z_1) - (OP_2 - Z_2)}{S} \tag{4.3.1}$$

Donde:

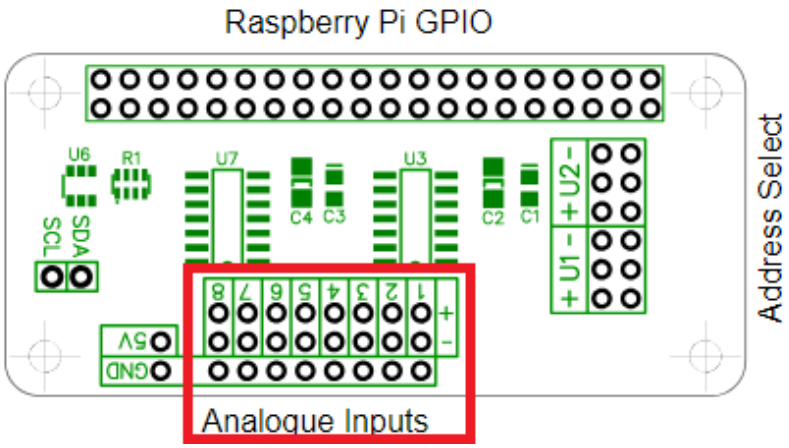


Figura 4.15: Esquema de la distribución de los canales del ADC.

- Los valores OP_1, OP_2 son los voltajes correspondientes al electrodo de trabajo y el electrodo auxiliar de los sensores, y deben corresponder al canal en el que se encuentren conectado. (Estos valores se encuentran en milivolts)
- Las variables Z_1, Z_2 son los valores totales de los electrodos de trabajo y auxiliar, respectivamente, y se encuentran en la hoja de datos de los sensores. (Representan el valor de su respectiva variable ambiental en un ambiente de aire limpio)
- La variable S: es la sensibilidad del sensor (Se encuentra en la hoja de datos del sensor).

Después de aplicar la fórmula de la concentración, el resultado obtenido se divide entre 1000 el resultado para convertir de ppb a ppm, esto porque las ecuaciones del índice IMECA requieren valores en ppm.
Posteriormente se aplican las ecuaciones del indice IMECA (ver tabla 4.1), para clasificar la información en sus cinco apartados (buena, regular, mala, muy mala y extremadamente mala). Cabe mencionar que en algunos gases no siempre se aplica la misma ecuación, esto depende de la cantidad de concentración obtenida en la medición, por lo que se hace uso de condiciones para implementar cada ecuación correspondiente a la cantidad de concentración obtenida.

Finalmente el programa queda en espera de que el usuario necesite realizar una medición para enviar los datos recabados. En el código que se muestra abajo se establecen 4 condiciones, ya que las mediciones son enviadas de manera consecutiva, llamando a cada lectura a través de la función sensores que lleva como argumento el número relacionado a cada gas. El sistema anterior se describe con el siguiente diagrama de flujo.

Tabla 4.1: Ecuaciones del índice IMECA con su correspondiente concentración en ppm.

Intervalo	Concentración (ppm)	Ecuaciones
Ozono (O_3)		
Buena (0-50)	0-0.055	$I[O_3] = C[O_3] * 100/0{,}11$
Regular (51-100)	0.056-0.110	
Mala (101-150)	0.111-0.165	
Muy mala (151-200)	0.166-0.220	
En extremo mala (>200)	>0.220	
Dióxido de Nitrógeno (NO_2)		
Buena (0-50)	0-0.105	$I[NO_2] = C[NO_2] * 50/0{,}105$
Regular (51-100)	0.106-0.210	$I[NO_2] = 1{,}058 + C[NO_2] * 49/0{,}104$
Mala (101-150)	0.211-0.315	$I[NO_2] = 1{,}587 + C[NO_2] * 49/0{,}104$
Muy mala (151-200)	0.316-0.420	$I[NO_2] = 2{,}115 + C[NO_2] * 49/0{,}104$
En extremo mala (>200)	>0.420	$I[NO_2] = C[NO_2] * 201/0{,}421$
Dióxido de Azufre (SO_2)		
Buena (0-50)	0-0.065	$I[O_3] = C[SO_2] * 100/0{,}13$
Regular (51-100)	0.066-0.130	
Mala (101-150)	0.131-0.195	
Muy mala (151-200)	0.196-0.260	
En extremo mala (>200)	>0.260	
Monóxido de Carbono (CO)		
Buena (0-50)	0-5.50	$I[CO] = C[CO] * 50/5{,}50$
Regular (51-100)	5.51-11.00	$I[CO] = 1{,}82 + C[CO] * 49/5{,}494$
Mala (101-150)	11.01-16.50	$I[CO] = 2{,}73 + C[CO] * 49/5{,}49$
Muy mala (151-200)	16.51-22.00	$I[CO] = 3{,}64 + C[CO] * 49/5{,}49$
En extremo mala (>200)	>22.00	$I[CO] = C[CO] * 201/22{,}01$

```
try:
        data = client_socket.recv(1024)
        if(data == b'0'):
            contador = 0
            CO = sensores(contador)
            costr = str(CO)
            client_socket.send(costr)
        elif(data == b'1'):
            contador = 1
            SO2 = sensores(contador)
            so2str = str(SO2)
            client_socket.send(so2str)
        elif(data == b'2'):
            contador = 2
            O3 = sensores(contador)
            o3str = str(O3)
            client_socket.send(o3str)
        else:
            contador = 3
            NO2 = sensores(contador)
            no2str = str (NO2)
            client_socket.send(no2str)
```

Figura 4.16: Segmento condicional para el envio de los datos de cada gas.

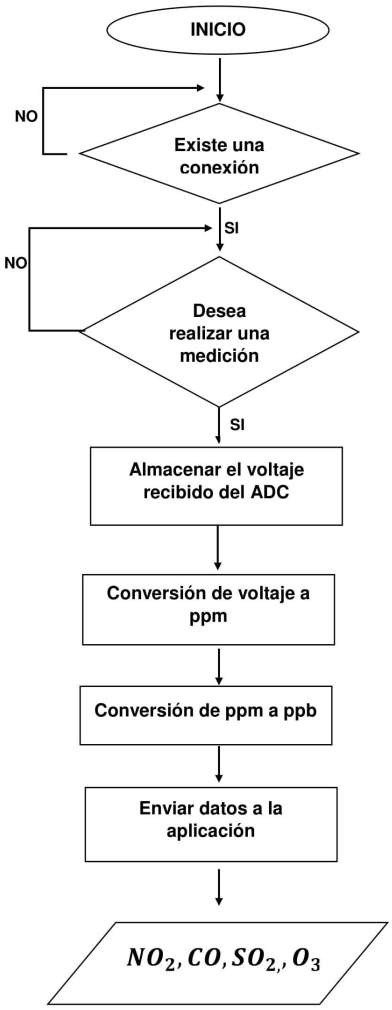


Figura 4.17: Diagrama de flujo del sistema de procesamiento de datos.

4.3.4. Creación de un SCRIPT en Raspberry Pi

Un script es un servicio que permite ejecutar una tarea en segundo plano desde el inicio del sistema operativo, detenerse o reiniciarse en el momento que el usuario lo desee. A continuación, se describe el proceso para la creación de un script en el sistema Raspbian. Para comenzar, se guarda el archivo a ejecutar dentro del script, en la dirección */home/pi/TESIS* (se recomienda crear una carpeta que almacene el archivo que se configurara dentro del script), para este caso se crea una carpeta con el nombre de TESIS. Posteriormente se define el servicio (script), se guarda en */lib/systemd/system/* y se coloca el nombre del servicio (calidadaire) con el comando *sudo nano calidadaire.service*. Finalmente se configura el script indicando la ubicación y nombre del archivo a ejecutar dentro de script en *ExecStart=/usr/bin/python /home/pi/calidadaire.py*, como se muestra en la fig 4.18. Se valida la configuración presionando "ctl + x" en el teclado, y se pro-

```
[Unit]
Description=calidadaire
After=multi-user.target

[Service]
Type=simple
ExecStart=/usr/bin/python /home/pi/calidadaire.py
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Figura 4.18: Comandos utilizados para configuración del script.

cede a la activación del servicio indicando nombre y ruta del script con el comando *sudo chmod 644 /lib/systemd/system/calidadaire.service* seguido del nombre y ubicación del archivo a ejecutar mediante el comando *chmod +x /home/pi/TESIS/calidadaire.py*. Despues se carga y se habilita el script con el comando *sudo systemctl daemon-reload* y *sudo systemctl enable calidadaire.service*, respectivamente y en el mismo orden que se explica aquí. Una vez realizado lo anterior, se inicia el script con *sudo systemctl start calidadaire.service*, en caso de que se desee dejar de ejecutar el script basta con introducir el comando *sudo systemctl stop calidadaire.service*, mencionando el nombre del servicio, que en este caso es llamado "*calidadaire*". Si se desea conocer el estado actual del script, se utiliza el comando *sudo systemctl status calidadaire.service*, donde se mostrara si el script está activado o no, así como el nombre y la ubicación del mismo dentro de la Raspberry, como se muestra en la fig. 4.19.

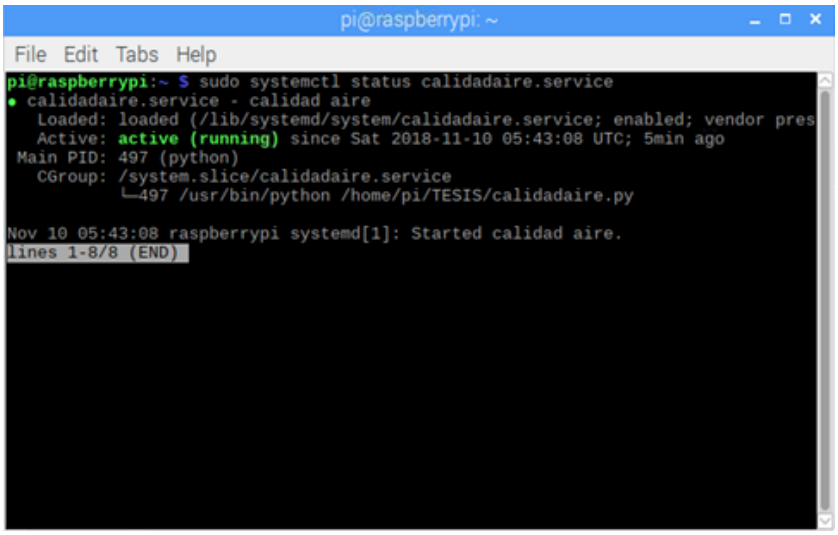


Figura 4.19: Terminal en Raspbian con el estado y caracteristicas del script.

4.4. Etapa de comunicación

Para realizar la medición de la presencia de contaminantes en el aire de manera móvil se desarrolló una aplicación para dispositivos móviles con sistema operativo Android, la cual es la encargada de controlar el proceso de adquisición de datos indicando a la Raspberry Pi el momento en el cual debe realizar una lectura de los datos entregados por los sensores, una vez realizada la medición, el dispositivo Android obtiene las coordenadas del lugar donde se realizó la medición, así como la hora en que se llevó a cabo la misma. Una vez obtenidos todos los valores anteriormente mencionados, el dispositivo los almacena en una base de datos creada con SQLite para su posterior subida al servidor Web.

El desarrollo de la aplicación se dividió en cuatro etapas individuales tal y como se ilustra en la figura 4.20. La etapa correspondiente a la comunicación Bluetooth se encarga de realizar la búsqueda del dispositivo con el cual se desea establecer la conexión, en este caso la Raspberry, una vez identificado el dispositivo la etapa de Bluetooth se encargará de establecer la conexión con él, así como también enviar las instrucciones necesarias a la Raspberry Pi para realizar la medición de los contaminantes y posteriormente recibir los datos obtenidos de la medición.

La etapa correspondiente al sistema de localización tiene la labor de obtener las coordenadas del dispositivo móvil en el momento en que se realiza la medición de los contaminantes utilizando Network provider.

Otra de las etapas con las que cuenta la aplicación es la que se encarga de la gestión de la base de datos en SQLite, esta etapa se encarga de almacenar la información

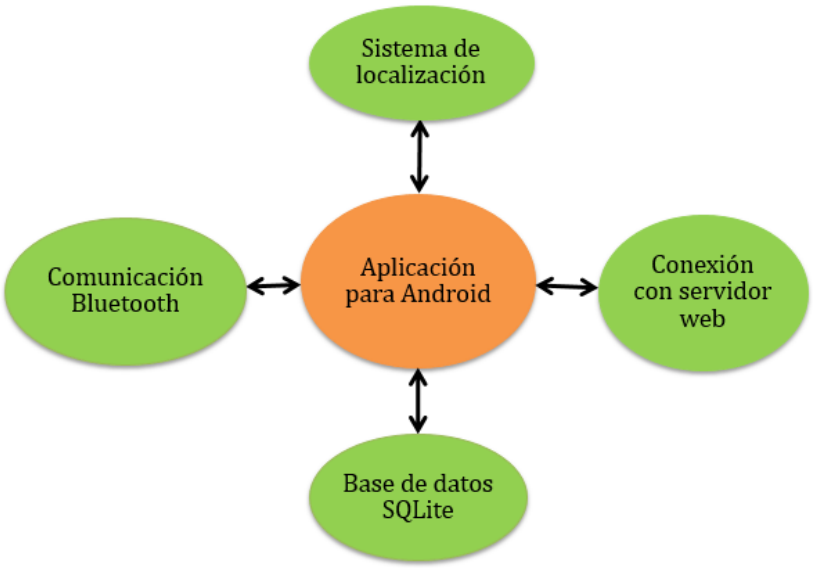


Figura 4.20: Diagrama de las etapas de la aplicación en Android

recibida de la Raspberry Pi por medio de la etapa de Bluetooth y las coordenadas proporcionadas por el módulo del sistema de localización, la información almacenada en la base de datos podrá ser visualizada a través de una interfaz gráfica de usuario y de igual manera podrá ser subida a un servidor Web por medio de la etapa de conexión con el servidor, esta es la manera en que se relacionan cada una de las cuatro etapas de la aplicación.

4.4.1. Etapa de comunicación Bluetooth

Como ya se ha mencionado anteriormente la etapa Bluetooth es la encargada de establecer la comunicación entre el dispositivo Android y la Raspberry Pi, para lograr esta comunicación es necesario realizar una serie de procesos tales como la búsqueda de dispositivos móviles, la gestión de la conexión con dispositivos remotos y la transferencia de información, estos procesos han sido separados en tres clases Java distintas para hacer más comprensible la estructura del programa, estas clases llamadas “principalBt”, “conexionBt” y “comunicacionBt” cuyo diagrama de clases se ilustra a continuación. Antes de realizar la programación de las clases Java descritas anteriormente es necesario conceder algunos permisos a la aplicación que se está desarrollando para que pueda interactuar con el sistema Bluetooth del dispositivo Android, es decir, si no se conceden ciertos permisos relacionados con el Bluetooth a la aplicación esta no podrá interactuar con las bibliotecas propias de Android encargadas del manejo del módulo Bluetooth

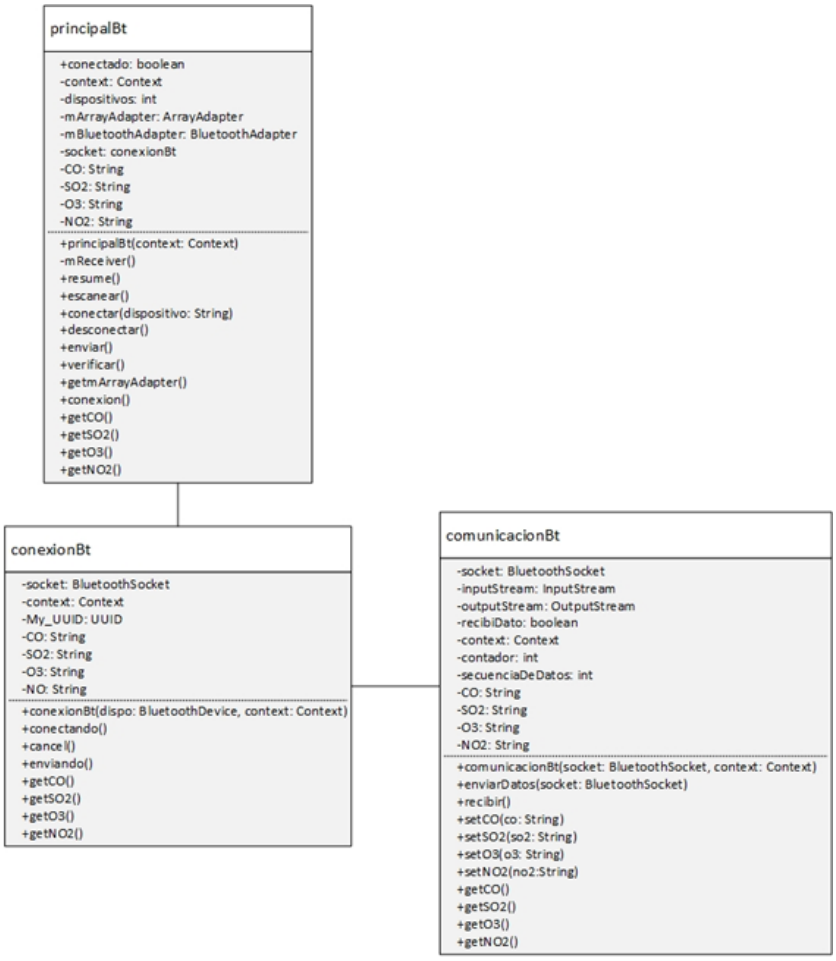


Figura 4.21: Diagrama de clases de la etapa Bluetooth.

del dispositivo, por lo que lo primero que hay que hacer es otorgar los permisos a la aplicación, el primero de ellos recibe el nombre de "Bluetooth", el cual gestiona los aspectos relacionados con la comunicación tales como solicitar y aceptar conexiones y transferir información, el otro permiso que debe concederse es el llamado "BLUETOOTH_ADMIN" el cual permite realizar la búsqueda de dispositivos y manipular las configuraciones del Bluetooth. La declaración de estos permisos se realiza en el documento "*manifest*" de la aplicación (ver figura 4.22), Una vez declarados estos permisos se pueden comenzar a programar las clases Java mencionadas al inicio de esta sección.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="mx.juane1711.airqsensing">

  <uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Figura 4.22: Declaración de los permisos Bluetooth.

Clase principalBt

De las clases mencionadas en la sección anterior. "*principalBt*.[es] la primera clase involucrada en el proceso de establecer una conexión por medio de Bluetooth con otro dispositivo, en este caso en particular la Raspberry Pi 3, y su función principal es la de realizar un escaneo para encontrar los dispositivos Bluetooth cercanos. Este escaneo consiste en enviar una solicitud de información a todos los dispositivos cercanos, sin embargo, solo los aparatos que estén configurados como detectables responderán a la solicitud realizada enviando el nombre del dispositivo y dirección MAC. Para realizar este escaneo se hace uso de algunas clases propias de Android Studio como "Bluetooth Adapter" la cual representa el adaptador Bluetooth del dispositivo local, es decir el equipo en el cual se ejecutará la aplicación, esta clase permite llevar a cabo las tareas fundamentales del módulo Bluetooth, como iniciar la búsqueda de nuevos dispositivos y obtener una lista de los que ya han sido vinculados anteriormente. Otra clase utilizada es "*Bluetooth Device*" la cual permite representar a un dispositivo Bluetooth lejano, al cual a partir de ahora se referirá como *bluetooth device*, esta representación es creada mediante un *bluetooth adapter*, utilizando la dirección MAC del dispositivo con el que se desea establecer la conexión, las principales funciones del *bluetooth device* son establecer la conexión con el dispositivo y obtener información de este como su nombre y dirección. Además de las clases propias del Bluetooth es necesario hacer uso de la clase "*Intent*", básicamente un *Intent* es un objeto de acción el cuál puede ser

utilizado para solicitar una acción de otro componente de la aplicación, como puede ser iniciar una nueva actividad, iniciar un nuevo servicio o mandar un mensaje, en este caso se utilizará un *Intent* para iniciar un nuevo servicio, el cual será empleado para realizar el escaneo de dispositivos, este proceso se explicará más adelante en esta sección. Teniendo en cuenta lo anterior para realizar el escaneo se crea un método llamado "escanear", este método será llamado desde la clase principal de la aplicación, la función de este método será crear un *Intent* que inicie el servicio de búsqueda de dispositivos cercanos a partir del método "ACTION_FOUND" de la clase "*BluetoothDevice*", este método enviará un *Broadcast* cuando detecte un dispositivo remoto, dicho Broadcast será recibido por un objeto de la clase "*BroadcastReceiver*" el cual será el encargado de almacenar el nombre y la dirección MAC de los dispositivos detectados en un *Arrayadapter* para posteriormente enviarlos a la actividad principal y desplegarlos en un elemento *ListView* de manera que el usuario pueda elegir el dispositivo con el cual desea establecer conexión. Como se mencionó anteriormente el responsable de detectar si un nuevo dispositivo ha sido encontrado, será un objeto de la clase "*BroadcastReceiver*.ª través de su método "on receive", el cual será llamado cada vez que el método "ACTION_FOUND" haya obtenido una respuesta de un dispositivo bluetooth, cuando esto suceda se creará un nuevo *bluetooth device* el cual contendrá la información del dispositivo remoto necesaria para poder establecer conexión con el cómo su nombre y dirección MAC. Ya que se tiene está información es necesario, antes de establecer la comunicación Bluetooth, almacenarla de tal manera que sea visible para el usuario y sea el quien decida con que dispositivo desea establecer comunicación (en caso de haber más de un dispositivo disponible) o en caso de que solo se haya obtenido respuesta de un dispositivo verificar que sea con el que sea desea conectarse.

Para almacenar la información obtenida y después mostrarla en pantalla se creó un *Arrayadapter* el cual almacenara el nombre y la dirección del dispositivo encontrado, sin embargo, ya que el método "start Discovery" continuara escaneando en busca de dispositivos hasta que se le indique que se detenga es posible que detecte más de una vez al mismo dispositivo lo que ocasionaría que se tuviera información repetida, para evitar esto es necesario hacer una comparación de los datos del nuevo dispositivos con los ya almacenados en el *Arrayadapter*, de esta forma si no se tiene registrado ese dispositivo se agregara a la lista y en caso contrario el bluetooth device será desechado. El *Arrayadapter* será enviado al main activity por medio del método "get*Arrayadapter*" para posteriormente ser mostrado en un elemento de lista desplegable (spinner).

Ya que el usuario ha seleccionado el dispositivo con el cuál quiere establecer comunicación comienza el proceso de conexión, el cual se lleva a cabo en el método "conectar", la función de este método consiste en obtener la dirección MAC del dispositivo seleccionado en la lista de dispositivos disponibles y crear un nuevo bluetooth device con la dirección MAC obtenida, posteriormente se instancia un objeto de la clase conexionBt

llamado "socket" mandando como parámetro el bluetooth device creado previamente y se detiene el proceso de búsqueda de dispositivos, para ahorrar recursos, finalmente se llama al método "conectando" de la clase conexionBt a través del socket creado, este método será el encargado de establecer la conexión con el dispositivo .En el caso de que se desee terminar la conexión se llamará al método "desconectar" el cual a su vez llamara al método "cancel" de la clase conexionBt a través del mismo socket empleado para realizar la conexión.

Finalmente se cuenta con un método "enviar" el cual verificara que exista conexión con algún dispositivo, en caso de que así sea se llamara al método "enviando" de conexionBt y si fuera el caso de que no hay conexión con ningún dispositivo remoto se le notificara al usuario a través de un Toast.

Clase conexionBt

Esta clase es la encargada de establecer la conexión con un dispositivo Bluetooth remoto, el cual será proporcionado por la clase Bluetooth a través del constructor de clase, de igual manera se encargará de cancelar una conexión ya establecida y a su vez forma parte del proceso de envío de información al dispositivo remoto. Para realizar estas funciones la clase conexionBt hace uso de la clase propia de Android "BluetoothDevice", la cual ya ha sido descrita en la sección anterior, además hará uso de otra clase propia de Android, la llamada "Bluetooth Socket" la cual permitirá crear objetos llamados "socket" los cuales permitirán iniciar, administrar y finalizar una conexión. Antes de poder comenzar con el proceso de conexión con el dispositivo remoto es necesario crear un socket para el dispositivo proporcionado por la clase Bluetooth, esto se hace en el constructor de la clase tal y como se ilustra en la figura 4.23. Lo

```
tmp = device.createRfcommSocketToServiceRecord(My_UUID);
```

Figura 4.23: Creación del socket para el dispositivo remoto.

que se ilustra en la imagen anterior, es la creación de un socket para el dispositivo con el que se desea establecer la comunicación, como se puede ver se le está enviando el parámetro "My_UUID" al método encargado de la creación del socket, este parámetro corresponde a una cadena de texto la cual tiene almacenada el UUID a utilizar, un UUID (identificador único universal) es un número de 16 bytes expresado por 32 dígitos hexadecimales, los cuales están agrupados en un bloque de 8 dígitos, seguido de tres bloques de 4 dígitos y uno de 12, estos bloques están separados por un guion, este identificador es definido por el dispositivo que hace la función de servidor en la

comunicación Bluetooth, sin embargo, cuando se desea establecer conexión con una tarjeta serial (como es el caso con la Raspberry) se puede utilizar el siguiente UUID bien conocido: El identificador ilustrado en la figura 4.24 puede ser utilizado para establecer

```
private static final UUID My_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

Figura 4.24: UUID bien conocido para tarjetas seriales.

la conexión bluetooth con tarjetas con comunicación serial tales como las placas Arduino y Raspberry.

Habiendo creado el socket para el dispositivo deseado es posible realizar el proceso de conexión, para ello se ha creado el método "conectando" el cual será llamado desde la clase principalBt y cuyo funcionamiento es sencillo, basta con llamar al método "connect" de la clase BluetoothSocket mediante el socket creado previamente, sin embargo el proceso de establecer conexión bloqueará la aplicación hasta que haya terminado, por lo que es necesario implementar un receptor de posibles excepciones mediante el uso de una secuencia "Try-Catch" la cual intentará establecer la conexión y avisará al usuario cuando está se haya establecido correctamente, y en caso de que se presente alguna excepción se le notificara al usuario que no ha sido posible establecer la conexión y cerrara el socket evitando de esta manera que la aplicación se trabe. La implementación de la secuencia Try-Catch se realiza tal y como se ilustra en la figura 4.25.

```
try{
   //Conecta el dispositivo a través del socket
   //esto bloqueará la aplicación hasta conectarse
   //satisfactoriamente o que se presente una excepción.
   mmSocket.connect();
   Toast.makeText(context, "Conexión establecida", Toast.LENGTH_SHORT).show();
}catch (IOException e){
   //Incapaz de establecer la conexión
   try {
      Toast.makeText(context, "No se pudo establecer la conexión", Toast.LENGTH_SHORT).show();
      mmSocket.close();
   }catch (IOException closeException){}
   return;
}
```

Figura 4.25: Secuencia Try-Catch utilizada para establecer la conexión con el dispositivo remoto.

El proceso para finalizar la conexión es igualmente sencillo, basta con llamar al método "close" mediante el socket utilizado tal y como se muestra en la figura 4.26. Al

igual que en el caso del método encargado de establecer la comunicación, el método encargado de terminar la comunicación requiere de una secuencia capaz de recibir excepciones en caso de que se presente algún problema al terminar la conexión. Es

```
//Cancelará una conexión en progreso y cerrará el socket.
public void cancel(){
   try {
      mmSocket.close();
      Toast.makeText(context, "Desconectado", Toast.LENGTH_SHORT).show();
   }catch (IOException e){}
}
```

Figura 4.26: Método encargado de eliminar la conexión establecida con un dispositivo remoto.

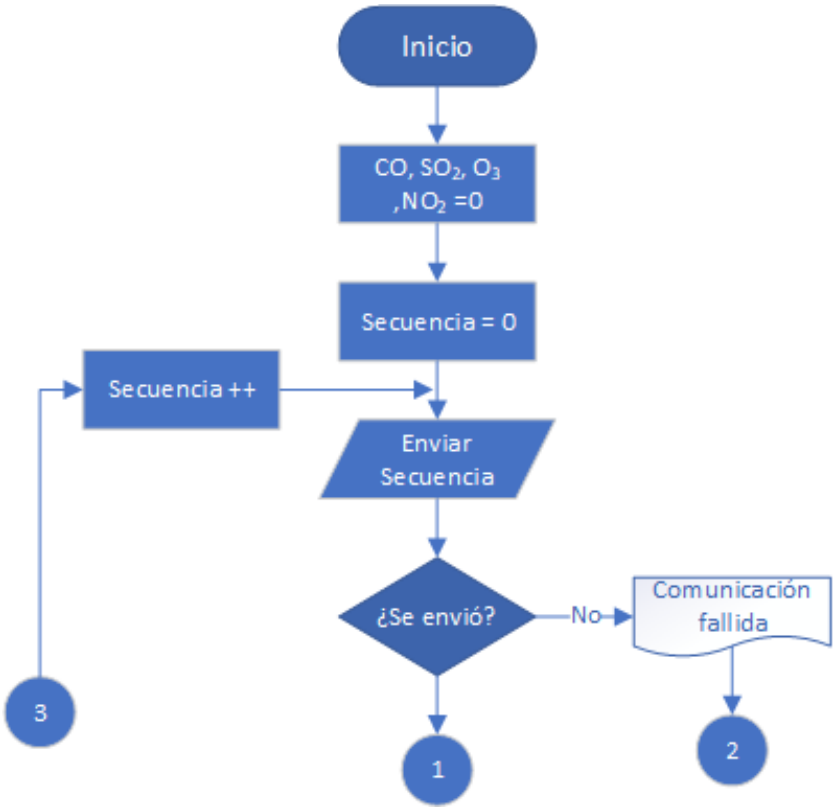
importante informar al usuario acerca del estado de la conexión con el dispositivo remoto, esto se hace mediante un Toast que será el encargado de mostrar en pantalla cuando la conexión se haya realizado exitosamente, cuando no se haya podido establecer la conexión y cuando se haya finalizado la conexión.

Clase comunicacionBt

Como ya se ha mencionado a lo largo de esta sección, el desarrollo de este proyecto requiere realizar una transferencia de información entre el dispositivo móvil (smartphone) y la Raspberry, está información corresponde a los índices IMECA de cada uno de los contaminantes del aire que han sido adquiridos por la Raspberry, es decir se requiere realizar la transferencia de los índices IMECA correspondientes al CO, SO2, O3 y NO2, la clase "*comunicacionBt*.[es] la encargada de realizar este proceso de comunicación Bluetooth, su función es la de enviar y recibir información del dispositivo remoto, para lo cual es necesario haber establecido previamente la conexión con el dispositivo deseado a través de la clase "conexionBt". Esta clase utiliza dos superclases propias de Android para realizar el tráfico de información entre dispositivos, la primera de ellas es la clase abstracta "InputStream", la cual es la superclase de todas aquellas clases que llevan a cabo un flujo de bytes de entrada, y la segunda es la superclase "OutputStream" la cual es una clase abstracta que, complementaria a "InputStream", se encarga de controlar el flujo de bytes de salida, en otras palabras la clase "InputStream" será la encargada de recibir la información enviada desde el dispositivo remoto hacia el dispositivo local mientras que la clase "OutputStream" permitirá enviar información hacia el dispositivo remoto.

Para realizar la transferencia de información entre dispositivos, es necesario establecer la comunicación con el dispositivo remoto deseado, por lo que antes de comenzar la transferencia de información, se verifica que exista un socket con el cual establecer la comunicación. En caso de no existir se le notificará al usuario que no existe conexión con ningún dispositivo remoto, por lo que no es posible realizar la transferencia de información.
En caso contrario se comienza el proceso de transferencia de información, para lo cual existen dos métodos involucrados en el proceso de transferencia de información. El primer método se llama "*enviardatos*" t es el encargado de enviar la indicación a la Raspberry de que debe realizar una medición, y a la par solicitará que se envíen los datos medidos en orden. El segundo método involucrado se llama "*recibir*", este método se encargará de guardar la información recibida de la Raspberry en variables para su posterior almacenamiento en la base de datos. Tal y como se ilustra en el diagrama de la figura 4.27, para realizar la transferencia de información se utiliza un contador que establezca la secuencia de los datos que se están enviando y de los datos que se están recibiendo, de esta manera se garantiza que los datos recibidos de la Raspberry se asignen a su variable correspondiente.

Para realizar el envío del indicador de secuencia por medio de Bluetooth se hace uso del método de la clase OutputStream "write" al cual solo es necesario pasarle como parámetro el dato que se desea enviar al dispositivo remoto, cabe destacar que si bien antes de comenzar el proceso de transferencia de información se ha verificado que exista conexión entre ambos dispositivos puede darse el caso de que se pierda la conexión una vez iniciado el proceso de transferencia de información, para evitar problemas con la aplicación en caso de que se presente este problema se hace uso de una secuencia de recepción de excepciones, la cual en caso de presentarse alguna excepción relacionada con la entrada o salida de datos informará al usuario de que se presentó un problema y suspenderá el proceso de transferencia de datos.
Si no se presenta ningún problema en el proceso de envío de la secuencia de datos hacia la Raspberry lo que procede es poner al dispositivo móvil en espera de recibir información mediante el inputStream, esto se hace mediante el método de la clase InputStream "read" , el cual leerá el contenido de un arreglo de bytes llamado buffer, la información recibida de la Raspberry es almacenada en este buffer de donde es leída por el método "read" y convertida a una cadena de caracteres para ser almacenada, en caso de que no se reciba ninguna información de la Raspberry el buffer quedará vacío lo cual ocasionará una excepción que dará por terminado el proceso de transferencia de datos. Este proceso se realiza un total de cuatro veces (que es el número de contaminantes a medir), una vez terminado de manera satisfactoria se envían los datos recibidos al MainActivity para posteriormente ser almacenados en la etapa correspondiente a la base de datos.


Inicio
CO, SO2, O3 ,NO2 =0
Secuencia = 0
Secuencia ++
Enviar Secuencia
¿Se envió?
No
Comunicación fallida
3
1
2

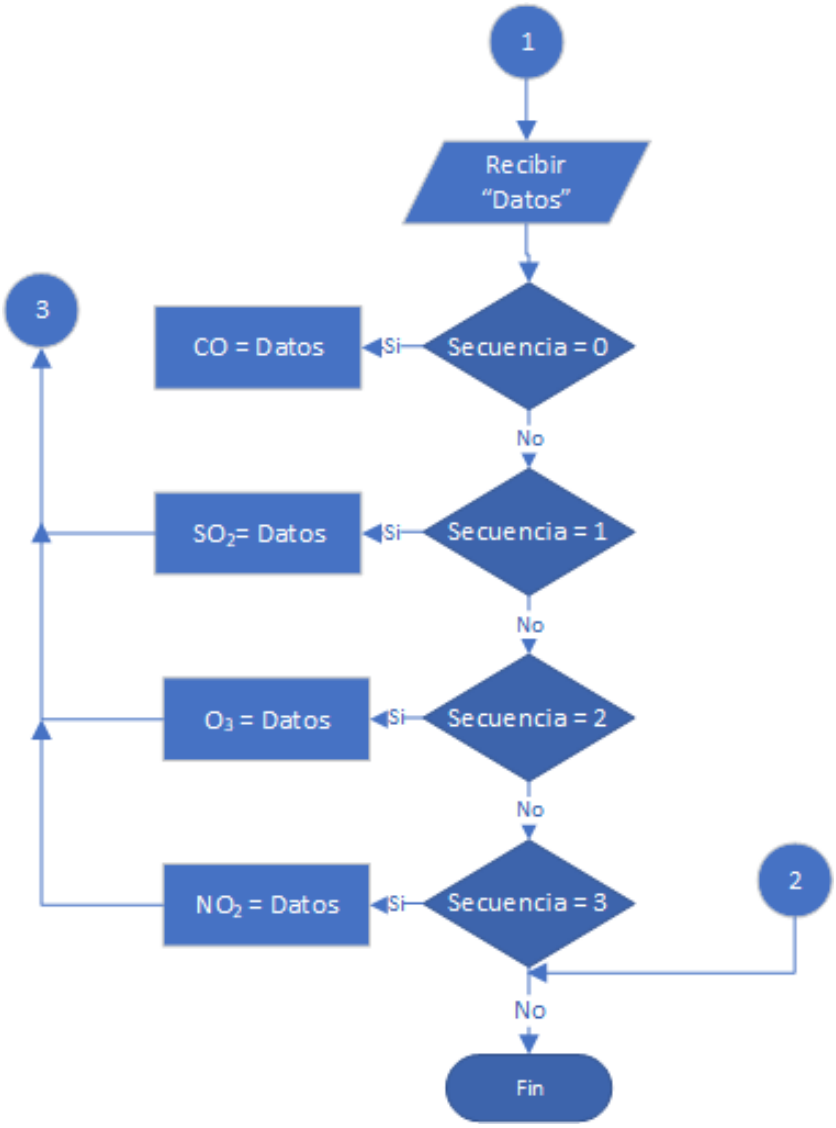


Figura 4.27: Diagrama de flujo de la transferencia de información.

4.4.2. Etapa de localización

Para poder dar un informe más preciso de los contaminantes presentes en el aire en una zona específica es necesario registrar el lugar en que se realizó la medición, esto es posible gracias a que los smartphone hoy en día cuentan con dos métodos de localización: el GPS y el proveedor de localización por red de Android (referido a partir de ahora como ANLP por sus siglas en ingles), cada uno de estos sistemas tiene características particulares que hay que tener en cuenta, en el caso del sistema de localización por GPS se cuenta con una mayor precisión comparado con el ANLP sin embargo, tanto el tiempo de respuesta como el consumo de batería son mayores, además de solo funcionar en exteriores, por otro lado el ANLP puede ser usado tanto en exteriores como en interiores debido a que determina la ubicación mediante la señal de torres de celular y Wi-Fi, sumado a esto tiene un menor consumo de energía y un tiempo de respuesta más corto. Al considerar lo anterior en un principio hubo incertidumbre acerca de que método de localización sería mejor utilizar, el GPS con mayor precisión, pero igualmente mayor consumo de energía y tiempo de respuesta o el ANLP que sacrifica precisión en favor de ahorrar batería, también se encontró que es posible habilitar la aplicación para que utilice cualquiera de los dos métodos, esto se hace declarando los permisos requeridos para ambos métodos y configurando ambos proveedores de localización, sin embargo al hacer pruebas se obtuvo que quien obtenía siempre la ubicación era el ANLP y con resultados bastante satisfactorios (dentro de un rango menor a 5 metros), por lo que se decidió probar utilizando únicamente el proveedor GPS obteniendo como resultado que la aplicación era incapaz de obtener la ubicación actual del dispositivo, por lo que se decidió tomar como proveedor el ANLP.

Declaración de permisos

De manera semejante a como ocurre con la etapa de comunicacion Bluetooth, para obtener la ubicación del smartphone es necesario pedir permiso al usuario para llevar a cabo la localización, según la documentación proporcionada por la comunidad de desarrolladores de Android (Android Developers), el permiso requerido por el ANLP es "ACCESS_COARSE_LOCATION" sin embargo al hacer pruebas no era posible obtener la ubicación, por lo que se probó utilizando el permiso destinado al proveedor GPS "ACCESS_FINE_LOCATION" el cual permite al usuario utilizar tanto GPS como ANLP, con este nuevo permiso los resultados fueron satisfactorios y se logró obtener la posición correctamente. Como ya se mencionó, para solicitar estos permisos es necesario declararlos en el archivo *manifest* junto con los permisos Bluetooth, en la figura 4.28 se ilustra la sección del archivo *manifest* de la aplicación correspondiente a la declaración de permisos. Una vez declarados los permisos la aplicación es capaz de utilizar los sistemas de localización del dispositivo móvil, sin embargo esto solo es

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 4.28: Declaración de permisos en el archivo manifest de la aplicación.

válido en dispositivos con sistemas operativos anteriores a Android 5.0, para el caso de aquellos equipos que cuenten con una versión más actual del sistema operativo es necesario indicar en el archivo manifest de la aplicación que es necesario utilizar las características de hardware correspondientes al proveedor de localización, en este caso del ANLP, esta declaración se realiza tal y como se ilustra en la figura 4.29. Con

```
<uses-feature android:name="android.hardware.location.network"/>
```

Figura 4.29: Declaración de las características de hardware utilizada para el funcionamiento del ANLP.

lo anterior se garantiza que el sistema de localización funcionará en dispositivos que cuenten con un sistema operativo Android 5.0 o mayor y debido a que las pruebas de la aplicación se realizaron en un smartphone con Android 5.0 "lollipop" se decidió incluir esta declaración lo cual, además hará a la aplicación compatible con dispositivos más modernos.

Clase localizacion

Esta clase es la encargada de realizar el proceso de obtención de las coordenadas (longitud y latitud) del dispositivo móvil, para esto hace uso de un conjunto de variables y métodos ilustrados en el siguiente diagrama de clase. Para comenzar el proceso de obtener la ubicación actual del dispositivo es necesario crear un objeto de la clase "*LocationManager*" llamado *locationManager*, el cual permite acceder a los servicios de localización del sistema, entre estos servicios se encuentra el de obtener actualizaciones periódicas de la localización geográfica del dispositivo, este objeto se instancia en el constructor de la clase como se observa en la figura 4.31. Una vez instanciado el objeto "*locationManager*" se implementa un método encargado de verificar que se hayan declarado los permisos en el archivo *manifest* de manera correcta, asimismo se encargará de registrar un "*listener*" con el *locationManager*, el cual se encargará de recibir las actualizaciones de la ubicación del dispositivo, este método queda de la siguiente manera: El *listener* será el encargado de detectar cuando haya un cambio

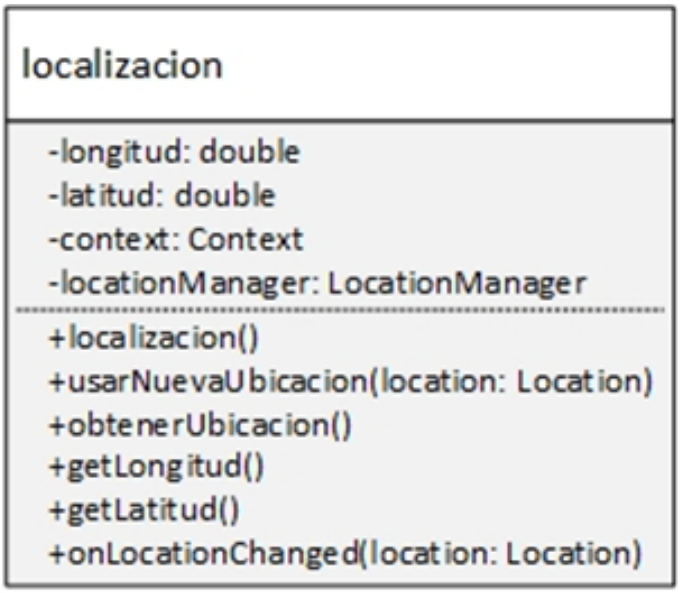


Figura 4.30: Diagrama de la clase "localizacion".

```
//Instanciamiento del objeto locationManager
public localizacion(Context context) {
   this.context = context;
   locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
}
```

Figura 4.31: Instanciamiento del objeto locationManager.

```
public void obtenerUbicacion() {
   //verificar si se tiene permiso para obtener ubicación
   int permissionCheck = ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION);

   //Iniciar Listener en busca de cambios de ubicación
   locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
   locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
}
```

Figura 4.32: Método encargado de registrar el *listener* para el *locationManager*.

relacionado con la ubicación o el proveedor, de hecho al declararlo Android Studio autocompleta la creación del *listener* con varios métodos, sin embargo el único que se utiliza es el método "*onLocationChanged*", este método tal y como su nombre lo indica es llamado cuando se detecta un cambio en la posición del dispositivo y su única función en esta aplicación será la de llamar al método encargado de obtener las coordenadas de la nueva ubicación. Para obtener tanto la latitud como la longitud en la que se encuentra el dispositivo basta con utilizar los métodos propios de la clase "*Location*" que son "*getLatitude*" y "*getLongitude*", estos métodos serán llamados por medio de un objeto de dicha clase llamado *location* y devolverán un valor double a unas variables previamente creadas con la finalidad de almacenar dichos valores para posteriormente enviarlos al main activity a través de sus métodos "get" correspondientes, una vez que se han almacenados los valores de las coordenadas es necesario indicarle al listener que deje de esperar por más actualizaciones, esto debido a que al estar en espera de un cambio de posición se consume demasiada batería además de que una vez obtenida la posición actual ya no es necesario monitorear la ubicación del dispositivo hasta que el usuario desee realizar una nueva medición, finalmente el método para obtener la nueva ubicación quedó de la siguiente manera:

```
public void usarNuevaUbicacion(final Location location) {
    latitud = location.getLatitude();
    longitud = location.getLongitude();
    locationManager.removeUpdates(locationListener);
}
```

Figura 4.33: Método que obtiene la ubicación del dispositivo

4.4.3. Base de datos SQLite

Una vez realizadas las mediciones es necesario almacenar los datos obtenidos para que el usuario sea capaz de visualizar los resultados de las mediciones, además de que al utilizar una base de datos se resguarda la información recabada en la memoria no volátil del dispositivo móvil por lo que en caso de presentarse alguna falla en el sistema no se perderán los datos obtenidos. Para la implementación de la base de datos se ha utilizado el sistema gestor de bases de datos SQLite, el cual es compatible con dispositivos móviles con sistema operativo Android y puede ser fácilmente añadido a una aplicación simplemente agregando un par te bibliotecas. El proceso de gestión de la base de datos se ha dividido en tres clases llamadas "medicionesDb", "medicionesDbContract" y "medicionesDbHelper", cada una de estas clases es fundamental para la correcta

implementación de la base de datos que se encargará de almacenar los valores obtenidos de las mediciones. En la figura 4.34 se muestran los diagramas de clase de cada una de las clases anteriormente mencionadas.

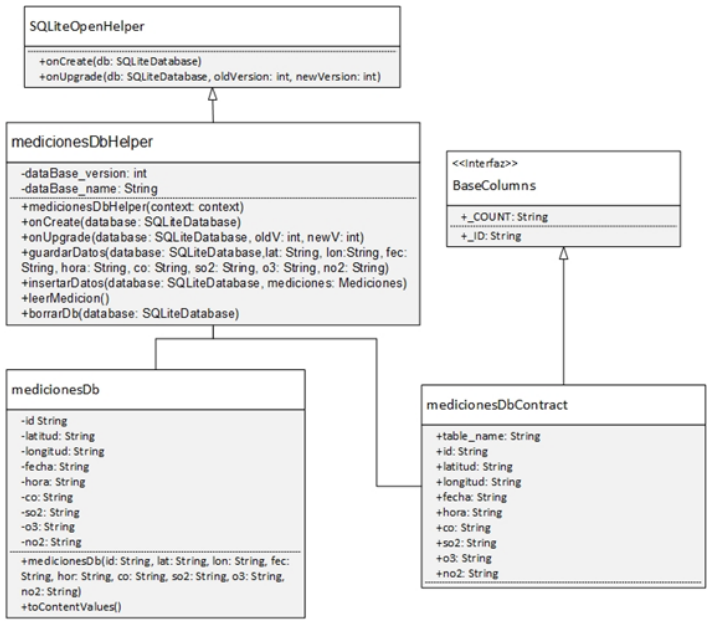


Figura 4.34: Diagrama de clases de la etapa de almacenamiento en la base de datos.

Clase medicionesDbHelper

Esta es la clase principal en la etapa de almacenamiento de la información en la base de datos, es en esta clase donde se define el nombre que tendrá la base de datos y la versión de la misma, así mismo esta clase contiene los métodos encargados de guardar y leer datos de la base de datos. La clase medicionesDbHelper es una clase hija de la clase propia de Android SQLiteOpenHelper, la cual es la clase encargada de la creación y manejo de versiones de las bases de datos, entre sus funciones se encuentran el abrir la base de datos indicada (si es que existe), crear la base de datos en caso de que no exista y realizar las actualizaciones necesarias. Una característica fundamental de la clase SQLiteOpenHelper es el hecho de que permite abrir y actualizar la base de datos hasta que es utilizada por primera vez y no cuando la aplicación es iniciada, esto evita que la aplicación se bloquee hasta que haya abierto la base de datos. Al crear un objeto de la clase medicionesDbHelper se mandará a llamar al constructor de la clase "*SQLiteOpenHelper*"pasándole como parámetros el nombre de la base de datos que se utilizará y la versión de la misma con la cual se desea trabajar. La clase *medicionesDbHelper* hace uso de otros dos métodos de su clase padre, el primero de

ellos es el método "*onCreate*", el cual fue sobrescrito con una consulta SQL la cual se encargará de crear la tabla "mediciones" cuyo diagrama se ilustra en la figura 4.35. De

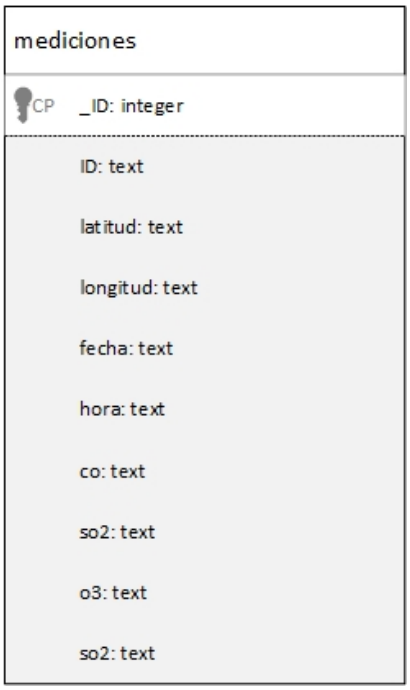


Figura 4.35: Diagrama de la tabla "mediciones".

la figura anterior, se desarrolla la tabla 4.2, que define todas las características de las variables en la base de datos.

El segundo método de la clase padre es el método "*onUpgrade*", este método es el encargado de realizar las modificaciones en la base de datos cada vez que exista una nueva versión disponible, sin embargo, en este caso la versión de la base de datos empleada es la número 1, por lo que no se hace uso de este método, pero puede ser utilizado si se realiza un trabajo a futuro donde sea necesario utilizar una nueva versión de la base de datos.

Los métodos encargados de guardar la información en la base de datos son dos: "guardardatos" el cual es llamado desde la clase principal de la aplicación y recibe la base con la que se está trabajando y los valores que se desean guardar en misma, este método a su vez manda a llamar al segundo método que forma parte del proceso de almacenamiento de la información "insertardatos", pasándole como parámetros la base de datos y un objeto de la clase "medicionesDb", el cual es creado utilizando los valores de las mediciones realizadas. El método "insertardatos" almacena los valores proporcionados por "medicionesDb" mediante el método "insert" de la clase SQLiteDatabase, al cual se le envían como parámetros el nombre de la tabla (definido en

Tabla 4.2: Definiciones de las variables de la tabla "mediciones"

Nombre	Tipo	Null	Llave	Descripción
_ID	Integer	No	Primaria	Identificador de SQLite para cada uno de los registros de la tabla. (No visible para el usuario)
ID	Text	No	No aplica	Identificador para los registros de la tabla. (Visible para el usuario)
Latitud	Text	No	No aplica	Latitud del punto de medición.
Longitud	Text	No	No aplica	Longitud del punto de medición
Fecha	Text	No	No aplica	Fecha en que se realizó la medición
Hora	Text	No	No aplica	Hora en que se realizó la medición.
CO	Text	No	No aplica	Índice IMECA de CO obtenido.
SO_2	Text	No	No aplica	Índice IMECA de SO_2 obtenido.
NO_2	Text	No	No aplica	Índice IMECA de NO_2 obtenido.
O_3	Text	No	No aplica	Índice IMECA de O_3 obtenido.

la clase medicionesDbContract) y los valores de los mediciones a través del método "*toContentValues*" de la clase "medicionesDb".
Para realizar la lectura de la base de datos se utiliza el método "leerMedicion", el cual hace uso del método "getReadableDatabase" perteneciente a la clase padre "SQLiteOpenHelper", con este método se puede obtener los datos almacenados en la base mediante una consulta SQL en la cual se le indica el nombre de la tabla que se quiere leer (mediciones) y los registros que quieren consultarse, en este caso se seleccionan todos los registros de la tabla.

Finalmente se cuenta con un método encargado de borrar la base de datos, este método es bastante útil ya que el sistema completo está pensado para realizar mediciones diarias por lo que si no se borra la información obtenida durante un día la base de datos crecería bastante, además de que al subir la base de datos al servidor Web se tendrían datos duplicados, es por esto que los registros obtenidos durante una jornada deben ser eliminados de la base de datos local una vez que se han almacenado en el servidor Web. El método encargado de borrar los datos almacenados recibe el nombre de "borrarDb", este método recibe como parámetro la base de datos con la cual se está trabajando y lo que hace es ejecutar una consulta SQL "Delete From" a la tabla mediciones, esto eliminará todos los datos almacenados en la base. Se realiza el borrado total de la base por los motivos previamente mencionados, para realizar una eliminación selectiva de los datos refiérase a la sección correspondiente a la página Web.

Clase medicionesDb

La función que tiene esta clase es asignar el valor de las mediciones realizadas a las variables de la clase "medicionesDbContract" correspondientes, para esto hace uso de un método constructor el cual será el encargado de recibir los valores a almacenar. Para realizar la inserción de datos en la base, es necesario hacer uso de la clase propia de Android "ContentValues", la cual es una clase empleada para almacenar un conjunto de valores, esto lo hace definiendo el nombre de la variable que se desea almacenar y el valor de dicha variable, esta característica permite almacenar en un objeto "ContentValues" un conjunto variado de datos diferenciados entre si por su nombre de variable, la cual recibe el nombre de "llave". Para agregar una variable a un objeto "ContentValues" se hace uso del método "put", el cual recibe como parámetros el nombre de la llave y el valor de la variable que se desea agregar. En este caso las llaves son los nombres de cada una de las columnas de la base, las cuales están definidas en la clase "medicionesDbContract" y los valores a almacenar son las mediciones recibidas en el método constructor, el encargado de almacenar estos datos es un objeto de la clase "ContentValues" llamado "values". El proceso anterior se realiza en el método "toContentValues" el cual una vez finalizado el almacenamiento, retornará el objeto "values".

Clase medicionesDbContract

Cuando se desarrollan aplicaciones que hacen uso de columnas como es el caso de una base de datos, se suele recurrir a las llamadas "contract class", las cuales son clases en las que se definen valores constantes que serán utilizados por otras clases para la realización de algún proceso, en este caso en particular la creación de una base de datos. Retomando lo anterior se tiene que la clase "medicionesDbContract" es la clase encargada de definir el nombre de la tabla en la que se almacenarán los datos y los nombres de cada una de las columnas. Antes de definir las constantes que serán empleadas lo primero que se hace es implementar la interfaz propia de Android "BaseColumns", la cual proveerá a la tabla de una columna por default llamada "_ID", esta columna sirve como un identificador para las columnas de la tabla ya que su valor no es repetible y es autoincrementado. Una vez implementada la interfaz se procede a definir el nombre de la tabla y de cada una de las columnas que la compondrán, esto se hace definiendo variables de tipo cadena de carácter las cuales deben ser públicas y estáticas ya que así podrán ser accedidas por las otras clases involucradas en la etapa de almacenamiento en la base de datos sin que estas tengan la necesidad de crear un objeto de la clase "mediconesDbContract".

4.4.4. Conexión con el servidor Web

Una vez que se han realizado todas las mediciones requeridas y se tienen almacenadas en la base de datos local el proceso siguiente es subirla a un servidor Web. Para realizar esta tarea es necesario hacer uso de una biblioteca desarrollada por la comunidad Android llamada "Volley", esta biblioteca permite que la aplicación en el dispositivo móvil sea capaz de conectarse con un servidor Web, sin embargo, el entorno de desarrollo Android Studio no cuenta con esta biblioteca, por lo cual es necesario instalarla de manera manual. Existen varias maneras de añadir la biblioteca "Volley" a Android Studio, pero la más sencilla de ellas es agregando una dependencia al archivo "buid.gradle" de la aplicación, tal y como se ilustra en la figura 4.36. Una vez añadida

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'

    implementation 'com.android.volley:volley:1.1.1'
```

Figura 4.36: Adición de la dependencia para la clase "Volley".

la dependencia correspondiente a la clase "Volley" es necesario sincronizar el proyecto con los archivos del "Gradle", para esto solo es necesario hacer clic en el icono de sincronización en la esquina superior derecha del entorno de desarrollo, tal y como se muestra en la figura 4.37. Una vez realizada la sincronización la biblioteca "Volley"

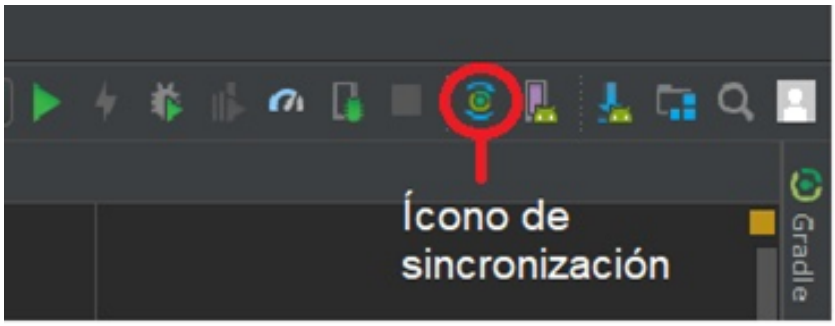


Figura 4.37: Ícono de sincronización del proyecto.

podrá ser utilizada en el proyecto.
Ya que para subir la base de datos al servidor es necesario contar con una conexión a internet es necesario definir los permisos necesarios para que la aplicación pueda acceder

a la red, estos permisos deben ser definidos en el archivo *manifest* de la aplicación, la figura 4.38 ilustra la manera en que estos permisos son declarados. Son dos los permisos

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="mx.tesis.airqsensing">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figura 4.38: Declaración de permisos para el acceso a la red.

requeridos para trabajar con aplicaciones conectada a la red, el primero de ellos es el encargado de permitir la conexión de la aplicación con internet, mientras que el segundo permite a la aplicación conocer si el dispositivo móvil cuenta con conexión a internet. Una vez definido lo anterior se puede comenzar a trabajar con la etapa de conexión con el servidor. Esta etapa ha sido dividida en dos clases llamadas "medicionesVolley" y "medicionesVolleyS", y cuyos diagramas de clase se ilustran en la figura 4.39.

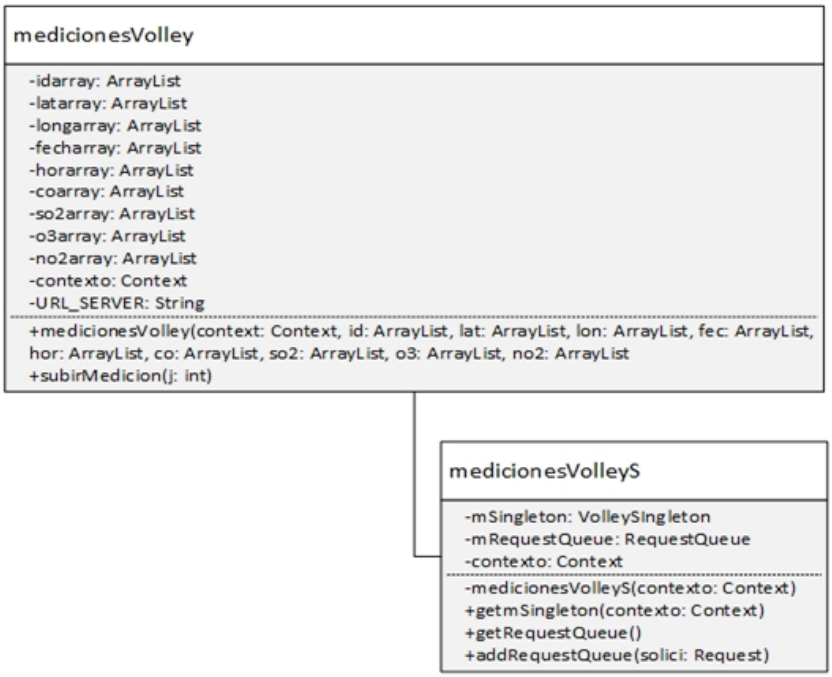


Figura 4.39: Diagrama de clases de la etapa de conexión con el servidor.

Clase medicionesVolley

Esta clase es la encargada de definir las peticiones HTTP que serán añadidas a la cola de solicitudes para posteriormente ser enviadas al servidor Web. Esta clase recibe las mediciones almacenadas en la base de datos por medio de la clase principal de la aplicación, las cuales son recibidos como arreglos que contienen toda la información de las mediciones realizadas, existe un arreglo por cada columna de la base. Además de los valores obtenidos de la base, para poder realizar consultas http es necesario definir el URL del servidor al cual se realizarán las consultas, en este caso la URL es la dirección de un sitio Web PHP que contiene el código requerido para recibir las consultas que serán realizadas por la aplicación.

Una vez que se cuenta con los datos que serán almacenados en el servidor y la URL del mismo se puede comenzar el proceso de creación de las solicitudes HTTP, el encargado de realizar este proceso es el método "subirMedicion", este método es llamado desde la clase principal de la aplicación y recibe como parámetro un número entero correspondiente al número de registro de la base, que se quiere subir al servidor. Dentro de este método se crea un objeto de la clase "StringRequest" perteneciente a la biblioteca "Volley", esta clase permite crear solicitudes HTTP y definirlas como una cadena de caracteres. Al crear un objeto StringRequest es necesario definir algunos parámetros a través de su método constructor, en total son cuatro los parámetros que deben ser definidos, el primero de ellos hace referencia al método que será empleado en la solicitud (GET, POST), en este caso se optó por utilizar el método POST ya que es más seguro que el método GET y por lo tanto más recomendable cuando se está trabajando con bases de datos. El siguiente parámetro que hay que definir es la URL del servidor al cual se enviará la consulta.

El tercer parámetro a definir es un listener que estará a la espera de recibir una respuesta satisfactoria, es decir, cuando se establece una comunicación exitosa entre el servidor Web y la aplicación, este listener es ejecutado. La función de este método es informar al usuario del resultado de la solicitud efectuada, ya que al momento de establecerse la comunicación con el servidor se pueden presentar situaciones inesperadas, como que el dispositivo se quede sin conexión a internet, lo cual ocasionaría que algunos registros no puedan ser subidos al servidor, es por esto que se hace uso de dos banderas, una llamada "completa" y otra "parcial", la primera de ellas comenzará con un estado verdadero el cual solo será modificado en caso de ejecutarse el *listener* de error (explicado más adelante en esta sección), mientras que la bandera "parcial" comenzará con un valor falso y en caso de ejecutarse el *listener* de respuesta el valor de esta bandera pasara a ser verdadero. Una vez que se realizan todas las consultas este *listener* pregunta si el valor de la bandera "completa" es verdadero de ser así le indica al usuario que la base de datos se subió satisfactoriamente, caso contrario le indicara

que solo se subió parcialmente la base de datos y se deberá verificar en el servidor Web cuales datos no fueron subidos correctamente.

El último parámetro que se define en la creación del objeto "StringRequest" es el *listener* de error, el cual es ejecutado cuando una consulta no se realiza satisfactoriamente, es decir cuando algún registro de la base de datos no pudo ser almacenado correctamente en el servidor Web. Este *listener* utiliza las banderas "completa" y "parcial" descritas anteriormente, sin embargo, lo que hará este *listener* es verificar el valor de la bandera "parcial", si esta es verdadera quiere decir que se subió por lo menos un registro por lo que se desplegara en pantalla un mensaje indicando que la base de datos se subió parcialmente al servidor, mientras que si la bandera es falsa significa que ninguna consulta fue exitosa por lo que se le informará al usuario que hubo un error al subir la base de datos.

Una vez creado el objeto de la clase "StringRequest" es necesario proporcionarle los valores de los datos obtenidos de la base de datos, para esto se sobre escribe el método "getParams" que pertenece a la clase "StringReques". Lo que hace este método es proporcionar los valores de la base de datos al objeto "StringRequest" creado, esto mediante un mapa, el cual es una estructura que almacena datos en parejas de claves y valor, es decir, almacena el valor de una variable asociado a una clave indicada, de esta manera, al utilizar un mapa se puede almacenar en una estructura los valores de las mediciones con su respectiva clave que indique a que corresponde cada valor.

Ya que se tiene el objeto "StringRequest" con el mapa correspondiente a los datos que se almacenarán en la base de datos se procede a llamar al método estático "getmSingleton" de la clase "medicionesVolleyS" para crear una cola de solicitudes y se envía la solicitud a esta cola por medio del método "addRequestQueue" de la clase "medicionesVolleyS".

Clase medicionesVolleyS

SI bien la clase "medicionesVolley" permite crear las solicitudes HTTP que serán enviadas al servidor es necesario crear una clase que se encargue de definir una cola de solicitudes para posteriormente enviarlas al servidor. Ya que el número de solicitudes que se realizarán puede ser considerable (dependiendo de la cantidad de mediciones realizadas) lo recomendable es definir una instancia única (singleton en inglés) lo cual permitirá definir una cola de solicitudes única que permanecerá vigente durante todo el tiempo que este activa la aplicación, en vez de tener que crear una nueva cola cada vez que se realice una solicitud al servidor, para definir esta instancia única se hace uso de la clase "VolleySingleton" perteneciente a la biblioteca "Volley".

Partiendo de lo anterior se tiene que la clase "medicionesVolleyS" es la encargada de crear la instancia única para la cola de solicitudes al servidor. El proceso de creación de la instancia comienza cuando el método "getmSingleton" es llamado desde la

clase "medicionesVolley", este método verificará si ya existe algún objeto de la clase "VolleySigleton", de ser así retornará ese objeto a la clase "mdicionesVolley", caso contrario creará un objeto "VollySingleton" pasándole como parámetro el contexto te la aplicación y después lo retornará. Una vez definida la instancia única se crea la cola de solicitudes, para esto se hace uso de la clase "RequestQueue" (propia de la biblioteca "Volley"), lo primero que se hace es verificar si ya existe una cola de solicitudes, de ser así se retorna a la clase "medicionesVolley", si aún no se ha definido una cola de solicitudes se crea un objeto de la clase "RequestQueue" al cual se le proporcionará el contexto de la aplicación como parámetro, de esta manera se habrá creado la cola de solicitudes utilizada para la comunicación con el servidor Web.

Finalmente el método "addRequestQueue" recibe la solicitud a realizar desde la clase "medicionesVolley" y la agrega a la cola de solicitudes mediante el método "add" propio de la clase "RequestQueue", una vez añadida una solicitud a la cola esta esperara su turno para ser ejecutada, el envió de las solicitudes al servidor Web lo realiza la clase "RequestQueue" de manera transparente para el desarrollador, una vez ejecutada una solicitud la respuesta obtenida será interpretada por lo métodos listener definidos en la clase "medicionesVolley", sin importar si la respuesta es satisfactoria o no la cola se solicitudes seguirá ejecutando cada una de las solicitudes hasta que la cola este vacía.

4.4.5. Clase principal airQSensing

Esta es la clase principal de la aplicación, es la encargada de controlar cada una de las etapas descritas anteriormente, además de ser la clase con la cual el usuario interactuará a través de la interfaz gráfica de usuario (GUI por sus siglas en inglés), los atributos y métodos que conforman esta clase se ilustran en la figura 4.40. Si bien la clase "AirQSensing" es la encargada de interactuar con la interfaz gráfica de usuario, esta no es diseñada a partir de esta clase, sino que es creada de manera visual con el modo de diseño del entorno de desarrollo Android Studio, con el cual se pueden agregar los elementos visuales de manera más rápida que se si hiciera por medio de código, una vez creada la interfaz de usuario (ver figura 4.41), los elementos visuales pueden ser relacionados con objetos creados en la clase "AirQSensing" y pueden interactuar con los métodos de la misma.

Cuando se inicia la aplicación el primer método en ser ejecutado es el método "onCreate", en el cual se inicializan los objetos relacionados con la clase "principalBt", las listas "ArrayList", la clase "localización" y el formato de obtención de fecha y hora "SimpleDateFormat", también se establecerá una relación entre el objeto Spinner "listaDispositivos" y el elemento visual Spinner de la interfaz gráfica de usuario, de esta manera se puede tener acceso al contenido de este último. Otra función que desempeña el método "onCreate" es verificar que el Bluetooth del dispositivo móvil se encuentre encendido, de no ser así enviará una notificación en pantalla al usuario para que encienda

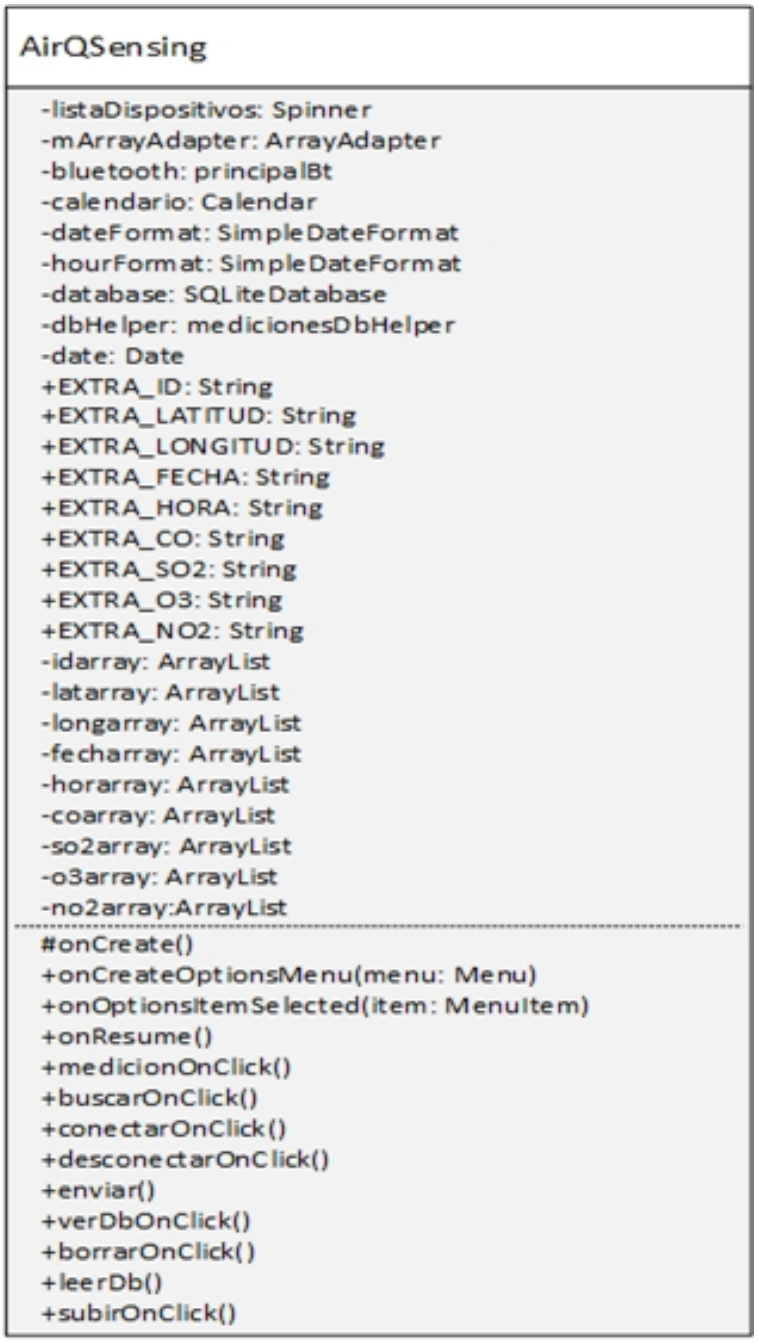


Figura 4.40: Diagrama de la clase "AirQSensing".

Figura 4.41: Interfaz gráfica de usuario asociada a la clase "AirQSensing".

el Bluetooth del dispositivo. Finalmente manda a llamar al método "leerDb" para tener acceso a los registros de las mediciones realizadas si es que se han realizado previamente.

El siguiente método en ser ejecutado es "onCreateOptionsMenu" el cual tiene la función de crear un menú de opciones que se ubica en la esquina superior derecha de la aplicación, en este menú se encuentran las opciones de subir y borrar la base de datos. La selección de una de las opciones del menú generado por "onCreateOptionsMenu" se realiza mediante el método "onOptionsIemSelected", el cual es llamado cuando el usuario selecciona cualquiera de las opciones del menú, si el usuario escoge la opción de subir se hace un llamado al método "subirOnClick", este realiza una lectura de la base de datos, luego crea un objeto de la clase "medicionesVolley" con el cual manda a realizar una consulta HTTP para cada registro de la base de datos. Por otra parte si el usuario selecciona la opción de borrar el método invocado es "borrarOnClick", cuya función es llamar al método "borradDb" de la clase "medicionesDbHelper" eliminando así todos los registros de la base de datos.

El método "medicionOnClick" es el encargado de controlar todo el proceso de medición y es ejecutado una vez que el usuario hace clic sobre el botón de realizar medición de la interfaz de usuario. Lo primero que hace este método es obtener las coordenadas (latitud y longitud) del dispositivo móvil por medio de la clase "localización", una vez que ha obtenido las coordenadas verifica que estas sean válidas, es decir, revisa que las coordenadas obtenidas tengan al menos seis dígitos después del punto decimal, ya

que esto garantiza una mayor precisión en la obtención de la ubicación del dispositivo móvil. Una vez validadas las coordenadas se verifica que siga existiendo comunicación Bluetooth con el dispositivo remoto, de no ser así se detendrá el proceso de medición.

Si no existe ningún problema con las coordenadas ni con la conexión con el dispositivo remoto se procede a obtener la fecha y hora del momento en que se esta realizando la medición, y se inicia el proceso de envío de datos (descrito en la sección de comunicación Bluetooth), de esta manera se obtienen los valores de los gases contaminantes medidos por la Raspberry Pi. Finalmente se hace uso del método "guardardatos" de la clase "medicionesDbHelper" para almacenar los valores de los contaminantes, así como las coordenadas, la fecha y la hora en la base de datos local.

Otro de los métodos con que cuenta la clase "AirQSensing" es el método "buscaronClick", el cual comienza el proceso de escaneo de dispositivos Bluetooth cercanos por medio de la clase "principalBt". El método "conectarOnClick" verifica que exista algún dispositivo Bluetooth en la lista de dispositivos del elemento de lista "spinner", si esta está vacía, se mostrará en pantalla un mensaje que le indique al usuario que no puede establecerse conexión ya que no ha seleccionado ningún dispositivo remoto, una vez que el usuario escoge un dispositivo remoto se inicia el proceso de conexión a través de la clase "principalBt". La función del método "desconectarOnClick" consiste en terminar la conexión con un dispositivo remoto (si es que existe) haciendo uso de la clase "principalBt".

Debido al poco espacio que se tiene en la pantalla de un dispositivo móvil como los teléfonos inteligentes, para realizar la consulta de la base de datos es necesario abrir una nueva actividad (activity) en la aplicación, es decir, se debe crear una nueva interfaz gráfica de usuario con la única finalidad de desplegar en pantalla una tabla con los registros de la base de datos, esta nueva actividad cuenta con una clase principal (semejante a la clase AirQSensing), llamada "Visor" la cual es la encargada de mostrar en pantalla la información proporcionada por la clase "AirQSensing". El intercambio de información entre actividades de una aplicación se realiza mediante un *Intent*, es decir, que para poder enviar el contenido de la base de datos desde la clase "AirQSensing" para que sea mostrado en pantalla por la clase "Visor" es necesario implementar un *Intent*.

El método encargado de realizar la transferencia de información entre las actividades es "verDbOnClick", el cual hace una lectura de la base de datos, si no hay ningún registro almacenado se muestra en pantalla un mensaje indicando al usuario que no hay datos guardados. Si existen datos almacenados en la base de datos se creará un objeto de la clase "*Intent*" indicando que la clase "Visor" es el destino al que se desea enviar información, una vez creado este objeto se envían a la clase "Visor" las listas de arreglo que contienen los datos de los registros de la base de datos, cada uno con un nombre que sirve para que la clase "Visor" conozca a que variable pertenece el valor

que esta recibiendo. El proceso de transferencia de información entre las actividades se ilustra en la figura 4.42.

```
public void verDbOnClick() {
   leerDb();
   if (!latarray.isEmpty()) {
      Intent intent = new Intent(this, Visor.class);


      intent.putStringArrayListExtra(EXTRA_LATITUD, latarray);
      intent.putStringArrayListExtra(EXTRA_LONGITUD, longarray);
      intent.putStringArrayListExtra(EXTRA_FECHA,fecharray);
      intent.putStringArrayListExtra(EXTRA_HORA, horarray);
      intent.putStringArrayListExtra(EXTRA_ID, idarray);
      intent.putStringArrayListExtra(EXTRA_CO, coarray);
      intent.putStringArrayListExtra(EXTRA_SO2, so2array);
      intent.putStringArrayListExtra(EXTRA_O3, o3array);
      intent.putStringArrayListExtra(EXTRA_NO2, no2array);

      //Iniciar actividad visor
      startActivity(intent);
   }else {
      Toast.makeText(this, "No hay datos guardados", Toast.LENGTH_SHORT).show();
   }
}
```

Figura 4.42: Transferencia de información entre actividades.

4.4.6. Clase Visor

Esta clase es la encargada de mostrar en pantalla los registros de la base de datos proporcionada por la clase "AirQSensing", los métodos y atributos empleados en esta clase se ilustran en el siguiente diagrama. Para mostrar la base de datos lo primero que se debe hacer es recibir los arreglos de datos proporcionados por la clase "AirQSensing" esto se hace en el método onCreate, este método es ejecutado una vez que se inicia la actividad "Visor" y una de las funciones que tiene es asociar cada uno de los elementos visuales que conforman la interfaz gráfica de usuario con un objeto de su clase correspondiente, de esta manera se pueden realizar modificaciones en el contenido de estos elementos desde la clase "Visor". Este método también se encarga de crear un objeto de la clase "*Intent*" el cual recibe los datos proporcionados por la clase "AirQSensing", y los almacena de acuerdo al identificador de cada uno de estos valores, tal y como se ilustra en la figura 4.44. Una vez obtenidos los registros de la base de datos se genera una tabla en la cual, debido al poco espacio disponible en pantalla y al hecho de que la base de datos puede tener una gran cantidad de registros, solo se muestra el contenido de una fila de la base de datos a la vez, y para consultar los demás registros de la tabla el usuario debe seleccionar el número de ID correspondiente al registro que quiere consultar. Cada vez que el usuario cambia el número de registro que

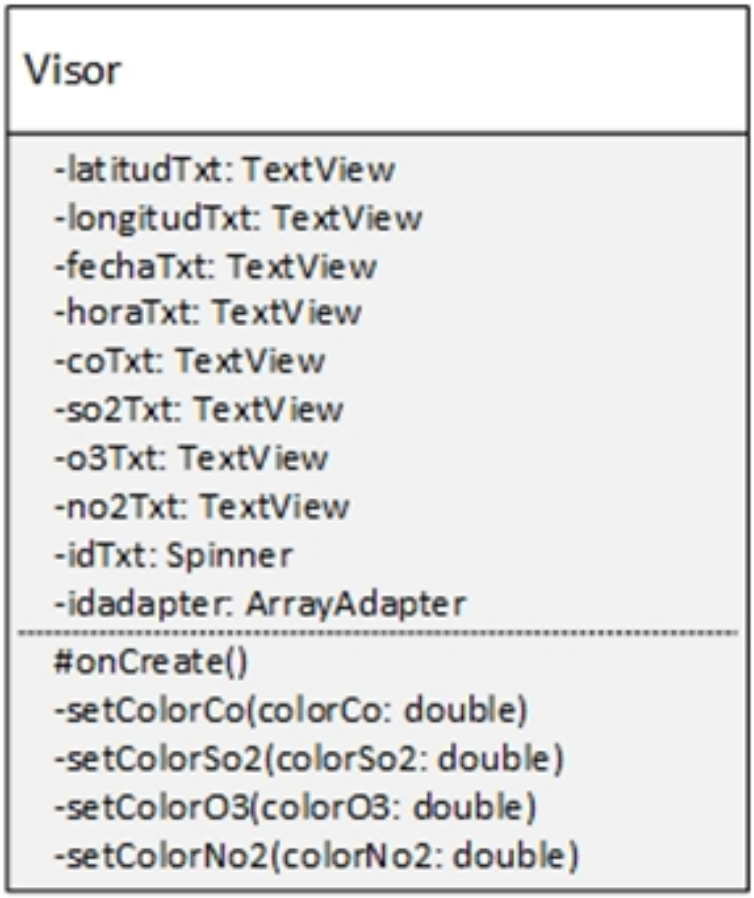


Figura 4.43: Diagrama de la clase "Visor".

```
Intent intent = getIntent();

final ArrayList latarray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_LATITUD);
final ArrayList longarray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_LONGITUD);
final ArrayList fecharray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_FECHA);
final ArrayList horarray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_HORA);
final ArrayList coarray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_CO);
final ArrayList so2array = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_SO2);
final ArrayList o3array = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_O3);
final ArrayList no2array = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_NO2);
ArrayList idarray = intent.getCharSequenceArrayListExtra(AirQSensing.EXTRA_ID);
```

Figura 4.44: Recepción de la base de datos.

desea consultar se manda a llamar al método "onItemSelected" asociado al elemento visual spinner, lo que este método hace es obtener el número de ID seleccionado por el usuario y colocar en cada una de las celdas de la tabla los valores de las variables correspondientes al número de registro seleccionado.

Ya que el índice IMECA establece una relación entre el índice obtenido y un color asociado al nivel de contaminación, cada vez que se visualiza un registro de la base de datos en la tabla generada en la interfaz gráfica de usuario se cambia el color de las celdas de los contaminantes de acuerdo con su índice IMECA, los encargados de realizar este cambio de color son los métodos "setColorCo", "setColorSo2", "setColorO3" y "setColorNo2". El principio de funcionamiento de estos métodos es el mismo, reciben el valor del índice IMECA que les corresponde y con base al nivel de contaminación que este índice representa (ver tabla 3.4) cambiaran el color de la celda por medio del método "setBackgroundColor", de esta manera la consulta de la base de datos tiene un impacto visual mayor. La interfaz de consulta de la base de datos se muestra en la siguiente figura.

Figura 4.45: Interfaz gráfica de usuario para la consulta de la base de datos.

4.5. Etapa de administración

Una vez que se han realizado las mediciones de los gases contaminantes del aire, y los datos obtenidos por los sensores han sido procesados y almacenados en la base de datos del dispositivo móvil, lo que se hace es subir los datos recabados a la base de datos de un servidor web, de manera que esta información pueda ser consultada por el público en general, para esto se hace uso de un sitio Web que muestra los resultados obtenidos de las mediciones así como información del proyecto en general y mantiene al tanto de las mediciones realizadas a los usuario registrados por medio de su correo electrónico. El desarrollo del sitio Web se divide en dos etapas fundamentales, la etapa "front-end", que es la parte del sitio Web con la cual los usuarios interactúan, en ella se muestra la información anteriormente mencionada, y la etapa de "back-end", encargada de acceder al a base de datos donde se almacenan los datos obtenidos de la aplicación móvil.

4.5.1. Front-end

Como ya se mencionó esta es la etapa del sitio Web encargada de interactuar con los usuarios y a través de la cual se muestra la información referente al proyecto y a los resultados obtenidos del mismo. Este segmento de la etapa de administración, está conformada por seis páginas Web, tres encargadas de mostrar información al usuario, dos que cumplen con la función de crear una cuenta de usuario y una página de administración de la base de datos, la cual solo esta disponible para usuarios con cuenta de administrador.

Las tres páginas de contenido tienen en esencia el mismo diseño, ya que cuentan con un menú de navegación para acceder a cada una de las páginas que conforman el sitio Web, en esta barra de menú también se encuentra un botón que despliega un submenú que permite al visitante iniciar sesión en el sitio Web, de esta manera es como los usuarios con cuenta de administrador pueden acceder a la página de administración de la base de datos descrita más adelante, este submenú desplegable también cuenta con un enlace a la página de creación de cuenta de usuario, de esta manera si el visitante no esta registrado puede registrarse en el sitio Web, lo cual le permitirá recibir notificaciones en su correo electrónico cada vez que exista una actualización en la base de datos.

Después del menú de navegación se tiene un área de contenido, en el caso de la página de inicio esta sección muestra una pequeña introducción acerca del proyecto y menciona las etapas en las que este está dividido, en la página "Acerca de" se detalla más a fondo cada una de las partes que conforman el proyecto y se mencionan el objetivo que se busca cumplir, finalmente la página de consulta muestra un mapa indicando los puntos en los cuales se realizaron las mediciones de los gases contaminantes del aire mediante marcadores del color correspondiente al índice IMECA obtenido en cada

punto, En la página de consulta también se cuenta con una tabla que muestra toda la información de las mediciones, es decir se muestran las coordenadas del punto donde se realizó la medición, la fecha y hora en que se realizó y el índice IMECA de cada uno de los gases medidos. La creación del mapa y la tabla se describen más adelante en esta sección. Finalmente se tiene la sección de pie de página en el cual se muestra información referente al IPN y enlaces hacia las redes sociales del instituto.

Página de consulta

Si bien el diseño en general de la página de consulta es igual al de las otras dos páginas de contenido esta tiene la característica de tener un enfoque más dinámico, ya que en esta página se muestran los resultados obtenidos en las mediciones de los gases contaminantes del aire, su contenido cambiará cada vez que exista una modificación en la base de datos, y aunque la creación de la tabla de resultados se lleva a cabo mediante PHP (ver la sección dedicada al back-end), la implementación de colores a la tabla y la creación del mapa se llevan a cabo del lado del cliente con JavaScript.

La finalidad de añadir colores a la tabla es crear un mayor impacto visual en el visitante del sitio al añadir el color correspondiente al índice IMECA de cada uno de los valores medidos. Para esto se hace uso de dos funciones JavaScript, la primera de ellas llamada "selección" es llamada una vez que la página ha sido creada, esto mediante el evento "onload", y su función es crear arreglos que contengan a cada uno de los elementos de la tabla asociados al índice IMECA de los gases. Esta selección de elementos es posible gracias a que cada elemento celda de la tabla tiene una clase correspondiente al gas que esta muestra, por ejemplo, todas las celdas que contienen el índice IMECA de las mediciones del monóxido de carbono están identificadas por la clase "co", de esta manera mediante el método "getElementsByClassName" es posible crear un arreglo que contenga a todos los elementos identificados con la clase indicada, tal y como se ilustra en la figura 4.46. La segunda función que forma parte del proceso para agregar color a la tabla es la función "color", esta recibe como parámetro un arreglo de elementos (los generados con la función "selección") y, mediante la implementación de un bucle obtiene el contenido de cada uno de los elementos del arreglo, es decir, crea una variable llamada "valorIMECA" en la cual almacena el valor del índice IMECA de una celda de la tabla, esta variable ingresa a una estructura de condición en la cual se modifica el atributo "style.background" del elemento celda de acuerdo con el valor de "valorIMECA" cambiando así el color de fondo de la celda, este proceso se repite para cada uno de los elementos del arreglo y se aplica a cada uno de los cuatro arreglos obtenidos con la función "seleccion" (correspondientes a cada gas medido), el proceso completo de la función "color" se ilustra en la figura 4.47. La figura 4.48 muestra el resultado obtenido de aplicar las funciones "selección y "color". Para la creación del mapa donde se muestran los puntos donde se realizaron las mediciones se hace

```
function seleccion() {
        var co = document.getElementsByClassName("co");
        color(co);
        var so2 = document.getElementsByClassName("so2");
        color(so2);
        var o3 = document.getElementsByClassName("o3");
        color(o3);
        var no2 = document.getElementsByClassName("no2");
        color(no2);
        mapeo();
    }
```

Figura 4.46: Función "selección" para la página "Consulta".

```
function color(elementos) {
        for( var i = 0; i < elementos.length; i++) {
          var valorIMECA = elementos[i].innerHTML;
          if( valorIMECA <= 50 ) {
            elementos[i].style.background = "#84FF63";
          }else if( valorIMECA > 50 &&  valorIMECA <= 100) {
            elementos[i].style.background = "#FEFF1B";
          }else if( valorIMECA > 100 &&  valorIMECA <= 150) {
            elementos[i].style.background = "#FFBC2C";
          }else if( valorIMECA > 150 &&  valorIMECA <= 200) {
            elementos[i].style.background = "#FF2929";
          }else {
            elementos[i].style.background = "#8D0AFD";
          }
        }
      }
```

Figura 4.47: Función "color" para la página "Consulta".

ID	Latitud	Longitud	Fecha	Hora	CO	SO4	O3	NO2
81	19.5000799	-99.134103	2018-10-12	13:01:31	-3.95	29.58	6.784	75.54
82	19.4999015	-99.133701	2018-10-12	13:27:14	2.047	34.13	31.65	28.01
83	19.4999015	-99.133701	2018-10-12	13:27:17	2.264	31.06	15.82	24.50
84	19.4999015	-99.133701	2018-10-12	13:27:19	2.140	34.13	90.45	36.76

ID	Latitud	Longitud	Fecha	Hora	CO	SO4	O3	NO2
81	19.5000799	-99.134103	2018-10-12	13:01:31	-3.95	29.58	6.784	75.54
82	19.4999015	-99.133701	2018-10-12	13:27:14	2.047	34.13	31.65	28.01
83	19.4999015	-99.133701	2018-10-12	13:27:17	2.264	31.06	15.82	24.50
84	19.4999015	-99.133701	2018-10-12	13:27:19	2.140	34.13	90.45	36.76

Figura 4.48: Tabla original, abajo: tabla con funciones "selección" y color" aplicadas.

uso de la biblioteca de código abierto "OpenLayers"[13] la cual permite agregar mapas interactivos a sitios Web de forma gratuita. Existen muchas versiones de esta biblioteca, pero la versión empleada en el desarrollo de este sitio Web es la versión 3.3.0, para tener acceso a las funciones con las que esta biblioteca cuenta basta con agregar la dirección del repositorio de estilos CSS y funciones JavaScript de OpenLayers en la cabecera de la página tal y como se ilustra en la figura 4.49. Una vez hecho esto es posible utilizar las

```
<link rel="stylesheet" href="https://openlayers.org/en/v3.3.0/css/ol.css" type="text/css">
    <script src="https://openlayers.org/en/v3.3.0/build/ol.js"></script>
```

Figura 4.49: Vinculación de la página Web con los repositorios de OpenLayers.

funciones con las que cuenta la biblioteca OpenLayers. Lo primero que se hace es crear arreglos que contengan a los elementos de la tabla de mediciones, tal y como se hizo con la función "selección", sin embargo, en este caso también se tomaran las columnas correspondientes al ID de medición y a las coordenadas del punto de medición (longitud y latitud). Para poder trabajar con los datos obtenidos de las mediciones lo que se hace es crear arreglos que contengan únicamente los datos de la tabla de mediciones, es decir se extrae el contenido de las celdas de la tabla, se hace la conversión de la información a datos de tipo flotante y se almacenan en nuevos arreglos, este proceso se ilustra en la figura 4.50. La finalidad de obtener el valor de las coordenadas de cada punto de medición es colocar en el mapa marcadores que indiquen los lugares donde se realizaron las mediciones, respecto al índice IMECA, este es utilizado para cambiar el color de los marcadores de acuerdo con el nivel de contaminación que en cada punto se registró (semejante a la manera en que se agrega color a la tabla de mediciones).

```
for(var i = 0; i < latitudArray.length; i++) {
                ID.push(parseFloat(idArray[i].innerHTML));
                latitud.push(parseFloat(latitudArray[i].innerHTML));
                longitud.push(parseFloat(longitudArray[i].innerHTML));
                co.push(parseFloat(coArray[i].innerHTML));
                so2.push(parseFloat(so2Array[i].innerHTML));
                o3.push(parseFloat(o3Array[i].innerHTML));
                no2.push(parseFloat(no2Array[i].innerHTML));
            }
```

Figura 4.50: Almacenamiento del contenido de la tabla en arreglos.

El primer paso a realizar para la creación de los marcadores es definir el estilo de los mismos, es decir se debe establecer la forma de los marcadores, los cuales pueden ser figuras geométricas o dibujos, así mismo se debe definir el color de estos marcadores, en este caso la figura elegida es un circulo, y debido a que los colores del marcador cambian de acuerdo con el nivel de contaminación medido se definen cinco estilos de marcador, uno para cada uno de los colores utilizados para representar la calidad del aire (verde, amarillo, naranja, rojo y morado).

Para crear los estilos de marcadores se hace uso de la clase “Style” de OpenLayers, con la cual se crea una variable en la que se definen la forma del marcador, el color de relleno y el color de margen del marcador, la declaración de una variable de estilo de marcador se ilustra en la figura 4.51. Una vez definidos los estilos de los marcadores es necesario establecer otros parámetros de los mismos como las coordenadas en las que se ubicará el marcador, para esto se hace uso de la función de OpenLayers “transform”, a la cual se le pasan como parámetros las coordenadas del marcador (latitud y longitud) y se deben especificar los parámetros EPSG (European Petroleum Survey Group), los cuales son un conjunto de datos de sistemas de coordenadas y proyecciones cartográficas utilizados en la definición de datos de posición en sistemas de información geográfica.

Además de las coordenadas de los marcadores se define un atributo “type” con la propiedad “click”, este atributo permite ejecutar una acción cuando se hace clic sobre uno de los marcadores. Adicionalmente se agregan los atributos “id”, utilizado para identificar cada uno de los marcadores, y mediante el atributo “imeca” se le asigna un color al marcador. Cada variable de propiedades es almacenada en un arreglo, este procedimiento se ilustra en la siguiente figura 4.52. Para cambiar el color de los

```
var stylebuena = new ol.style.Style({
                image: new ol.style.Circle({
                  fill: new ol.style.Fill({
                    color: 'rgba(10, 224, 28, 0.7)'
                  }),
                  stroke: new ol.style.Stroke({
                    width: 1,
                    color: 'rgba(10, 224, 28, 1)'
                  }),
                  radius: 7
                }),
              });
```

Figura 4.51: Definición de los estilos para los marcadores.

```
feature = new ol.Feature({
                geometry: new ol.geom.Point(
                  ol.proj.transform([longitud[i],latitud[i]], 'EPSG:4326', 'EPSG:3857')
                ),
                type: 'click',
                id: idmarker,
                imeca: imeca
              });
              features.push(feature);
```

Figura 4.52: Definición de los atributos de los marcadores.

marcadores primero se selecciona el índice IMECA más alto obtenido en la medición de entre los cuatro gases medidos, este valor es introducido en una estructura condicional con la cual se define el estilo que será aplicado al marcador. Para establecer el estilo del marcador se hace uso de la función "setStyle" a la cual se le pasa como parámetro la variable de estilo a la variable de propiedades (ambas creadas previamente).

En la figura 4.53 se ilustra la manera en que se asigna un estilo a un marcador mediante la función "setStyle", en este ejemplo la variable "styleBuena" corresponde a la variable en la que se definen los estilos de forma y color para un marcador que reporta una calidad del aire buena. Finalmente, para terminar el proceso de creación

Figura 4.53: Asignación de estilos a un marcador.

de los marcadores es necesario definir dos variables de la clase "vector" de OpenLayers, el primer vector que se crea es el llamado "vector fuente", el cual contiene el arreglo de variables de propiedades de los marcadores, así como el parámetro EPSG. El segundo vector que se crea es el vector de capa o "vectorLayer", este es empleado para mostrar los marcadores en el mapa, este vector tiene como propiedad "fuente" al vector fuente previamente mencionado. La creación de ambos vectores se muestra en la siguiente figura. Una vez que se ha creado el vector de capa para los marcadores es momento de definir la variable encargada de la creación del mapa, esto se logra haciendo uso de la clase "Map" de OpenLayers. Para crear una variable mapa es necesario establecer algunos parámetros, el primero de ellos es la parte de la página en la que se desea mostrar el mapa, en este caso el contenedor del mapa es un elemento "div" con un id = "map". A continuación, se agrega al mapa la capa de marcadores, esto se realiza con el atributo "layers" de la clase "Map", al cual se le agrega una variable de la clase "OSM" (Open Street Map) y el vector de capa que se ha creado para los marcadores. Finalmente hay que definir dos aspectos referentes a la visualización del mapa en la página Web, el primero de ellos es el punto en el que aparecerá centrado el mapa una vez que se termine de cargar la página, esto se hace con el parámetro "center", al cual se le indican las coordenadas del punto central del mapa, el segundo atributo que hay que definir es el zoom que tendrá el mapa. En este caso el mapa se ha centrado en las coordenadas del edificio cuatro de la ESIME Zacatenco. La definición de la variable "Map" se ilustra en la figura 4.55. Una vez realizado el procedimiento anterior se tiene un mapa en el cual aparecen los marcadores en los puntos de medición, sin embargo, para permitirle al usuario conocer información más detallada acerca de la medición

```
//Vector fuente para los marcadores
        var vectorSource = new ol.source.Vector({
          projection: 'EPSG:4326',
          features: features
        });
        //Vector layer utilizado para agregar los marcadores al mapa
        var vectorLayer = new ol.layer.Vector({
          source: vectorSource,
        });
```

Figura 4.54: Creación de vectores para la creación de los marcadores.

```
var map = new ol.Map({
        target: 'map',
        renderer: 'canvas',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.OSM()
          }),
          vectorLayer
        ],
        view: new ol.View({
          center: ol.proj.transform([-99.134697,19.4994781], 'EPSG:4326', 'EPSG:3857'),
          zoom: 17
        })
      });
```

Figura 4.55: Creación de la variable "map".

realizada como el índice IMECA de los gases medidos y la fecha y hora en la que se realizó la medición, cuando se hace clic sobre un punto marcador este muestra el ID de la medición a la que corresponde. Para proporcionar este dinamismo al mapa se hace uso de una función propia de la clase "Map" de OpenLayers llamado "on('singleclick')", esta función detecta cuando se hace clic sobre el mapa y obtiene los atributos del punto en cuestión, siendo los dos atributos de interés las coordenadas del punto y el atributo "type" del mismo, es con este último que se determina si el visitante hizo clic sobre un marcador, ya que todos los marcadores de puntos de medición cuentan con el atributo "type = click", tal y como se describió previamente, es decir, si el punto seleccionado por el visitante cuenta con este atributo se mostrará en las coordenadas del punto seleccionado un elemento párrafo que indique el ID del marcador seleccionado, de esta manera el visitante puede referirse a la tabla de mediciones y consultar todos los datos correspondientes al marcador seleccionado. El resultado obtenido es el mapa que se muestra en la figura 4.56.

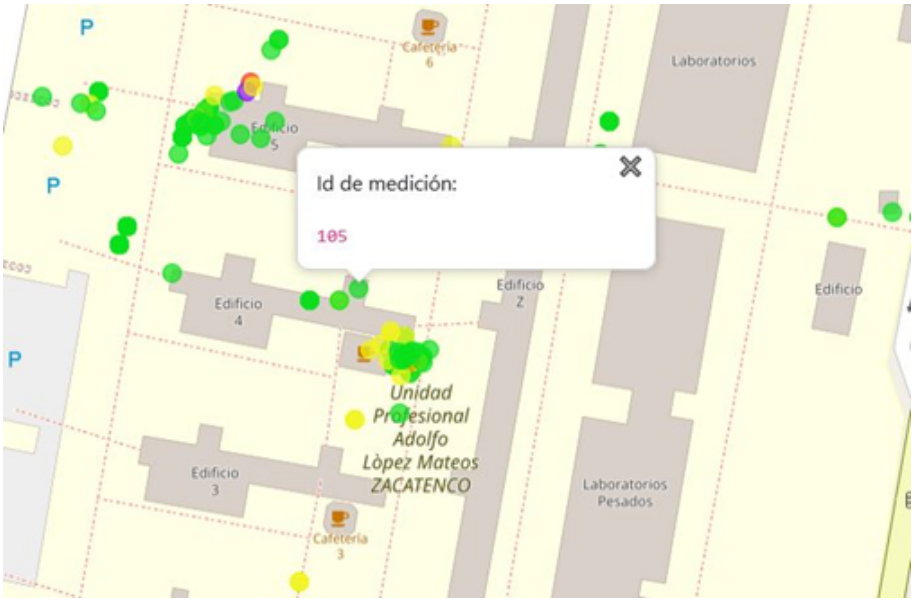


Figura 4.56: Mapa de los puntos de medición.

Páginas de registro y cuenta creada

Como se ha mencionado el sitio Web permite a los usuarios crear cuentas de usuario que les permiten mantenerse al día con las actualizaciones realizadas a la base de datos, y las páginas de registro y de cuenta creada son las encargadas de llevar a cabo el registro de los usuarios. Ambas páginas cuentan con una estructura similar a las páginas de

contenido, ya que están conformadas por una barra de navegación, un pie de página y una sección principal la cual es diferente para ambas páginas. En el caso de la página de registro, la sección principal está conformada por un formulario que el visitante debe llenar para poder crear una cuenta de usuario.

El primer dato que el visitante debe ingresar es el nombre de usuario con el cual iniciará sesión en el sitio Web, este nombre no puede ser igual a ningún otro nombre de usuario existente en la base de datos, la validación de este dato se explica en la etapa de back-end, al ingresar un nombre de usuario que ya esta registrado en la base de datos se mostrará un mensaje indicando al visitante que ya existe una cuenta asociada a ese nombre. El siguiente dato que el usuario debe ingresar es la contraseña que desea establecer, para el campo de ingreso de contraseña, así como para el de confirmación de contraseña se ha utilizado un elemento de entrada tipo "contraseña", el cual mantiene oculto el texto escrito en estos campos de manera que la contraseña ingresada no quede expuesta.

Algo importante a considerar es el hecho de que la contraseña y la confirmación de contraseña deben ser iguales, para asegurar esto se hace una verificación de la información del lado del servidor y en caso de que la información ingresada no sea la misma se notificara al usuario que ambas contraseñas deben coincidir. Por último, el visitante debe ingresar el correo electrónico al cual se enviarán las notificaciones de las actualizaciones de la base de datos, esto mediante un elemento de entrada tipo "correo", el cual se encarga de verificar que el texto ingresado cuente con las características propias de un correo electrónico, como el hecho de que se debe incluir el símbolo "@" y debe existir texto antes y después del mismo, esto se realiza de manera automática liberando al desarrollador de la tarea de validar el correo ingresado.

Al igual que ocurre con el nombre de usuario, el sitio Web no permite crear más de una cuenta de usuario asociada a un correo electrónico, por lo que se realiza una verificación del lado del servidor para detectar si existe una cuenta asociada al correo ingresado, de ser así se notificará al visitante que ese correo no es válido. En la figura 4.57 se muestra el formulario de creación de cuenta de usuario. Una vez que el visitante ingresa todos los datos de manera correcta es redireccionado a la página de cuenta creada (ver figura 4.58), la cual le da la bienvenida al sitio Web, le notifica que a partir de ahora recibirá notificaciones de las mediciones realizadas en su correo electrónico y finalmente se agrega un link hacia la página de inicio del sitio Web.

Página de administrador

Si bien la mayoría de las cuentas de usuario solo son creadas para recibir notificaciones cada vez que existe una actualización de la base de datos, existe un tipo de usuario administrador, este es el encargado de verificar que los datos subidos al servidor desde la aplicación móvil sean correctos y, en caso de detectar una medición errónea el usuario

Figura 4.57: Formulario de registro de usuario.

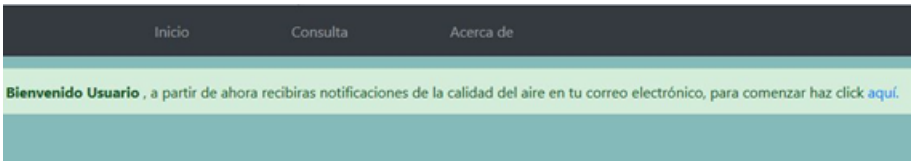


Figura 4.58: Mensaje de bienvenida en la página de cuenta creada.

administrador es capaz de eliminar un registro o un conjunto de registros de la base de datos de manera definitiva. El monitoreo de la base de datos se lleva acabo mediante una tabla de mediciones creada de la misma manera que la tabla de la página de consulta, solo que en este caso se agrega una columna más la cual contiene elementos de entrada tipo "checkbox" los cuales cuentan con la clase "registros" para ser identificados y permiten al administrador seleccionar los registros que desea eliminar. Para ejecutar la tarea de eliminación de registros se implementa un botón el cual, al ser presionado manda a llamar a la función "borrar" programada en JavaScript. Antes de comenzar con el proceso de eliminación de los registros se pregunta al usuario si está seguro de querer eliminar los registros seleccionados, esto se hace mediante el método "confirm" propio de JavaScript, en caso de seleccionar la opción "cancelar" el proceso se interrumpe. Si el usuario selecciona la opción "aceptar" lo que se hace es crear dos arreglos, uno que contenga a todos los elementos checkbox de la tabla y otro que contenga a las celdas asociadas al ID de cada registro.

A continuación, se hace un recorrido de todo el arreglo de elementos checkbox y mediante la propiedad "checked" se verifica cuales elementos han sido seleccionados. Cuando se detecta un elemento checkbox que ha sido seleccionado por el usuario, es decir si su atributo checked es verdadero, se agrega a un arreglo de elementos seleccionados llamado "idElegidos" el contenido del arreglo de ID correspondiente al elemento checkbox seleccionado, tal y como se ilustra en la figura 4.59. De esta manera se obtiene un arreglo con el ID de todos los registros de la base de datos que se desean eliminar. Finalmente, para eliminar los registros seleccionados se envía una petición

```
for (var i=0; i<registros.length; i++) {
                    if(registros[i].checked == true) {
                      idElegidos.push(id[i]);
                    }
                  }
```

Figura 4.59: Adquisición del ID de los elementos checkbox seleccionados.

al servidor por medio de AJAX con los ID de los elementos seleccionados, ya que la etapa de back-end es la encargada de realizar la consulta SQL que elimine los registros seleccionados. Una vez que se tiene la respuesta del servidor de que los registros se han eliminado se recarga el contenido de la página con el método "reload" propio de JavaScript, esto con la finalidad de actualizar la tabla de mediciones. La interfaz de la página de administrador se muestra a continuación.

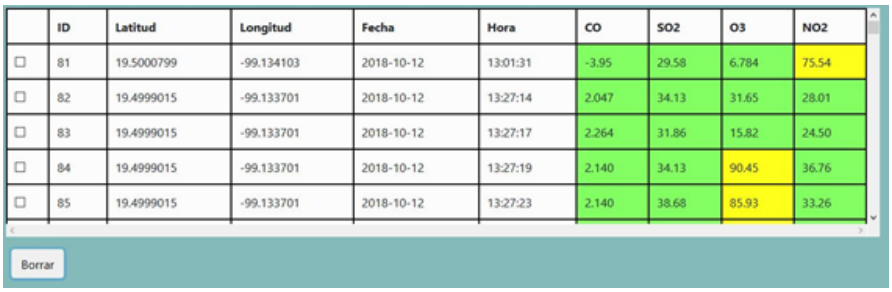


Figura 4.60: Interfaz de la página de administrador.

4.5.2. Back-end

La etapa de back-end hace referencia a todos los procesos que se realizan del lado del servidor a los cuales ni los visitantes ni administradores tienen acceso, es en esta etapa donde se hacen las consultas a la base de datos la cual cuenta con dos tablas, una que se encarga de almacenar la información de los usuarios registrados y otra en la que se guardan los datos de todas las mediciones realizadas. Otros procesos que se llevan a cabo del lado del servidor son la validación de datos tanto para la creación de cuentas de usuario como para el inicio de sesión, el envío de correos electrónicos y el inicio y término de sesión. La etapa de back-end está conformada por seis páginas diferentes las cuales llevan a cabo los procesos mencionados previamente.

Página nueva cuenta

Esta página recibe los datos enviados por la página "registro", verifica que los datos recibidos sean correctos, si es así añade al nuevo usuario a la base de datos y envía un mensaje de bienvenida, de lo contrario notifica a la página registro que los datos son erróneos para que esta a su vez informe al usuario de los errores que se presentaron. Lo primero que hace esta página es verificar que los datos recibidos hayan sido enviados por el método POST, de lo contrario redirecciona al usuario a la página de registro. Si los datos fueron enviados por el método POST se almacenan en variables diferentes y se crea una variable llamada "error", en la cual se almacenarán los mensajes de error en caso de que se produzca un problema en el proceso de creación de la cuenta. Ya que se tienen las variables se procede a la limpieza de los datos ingresados, es decir se eliminan todos aquellos símbolos que no son letras, ya que de no hacer esto se puede dar el caso que se haga inyección de código en la página, lo que podría comprometer la seguridad de la misma. Para llevar a cabo la limpieza de los datos de entrada se hace uso de los filtros de saneamiento de PHP "FILTER_SANITIZE_STRING" y "FILTER_SANITIZE_MAIL", el primero de estos es usado para limpiar cadenas de caracteres simples, tales como

nombres de usuario y contraseñas, el segundo filtro está diseñado específicamente para realizar la limpieza de correo electrónicos.

Una vez realizado el saneamiento de los datos de entrada se verifica que la contraseña y la confirmación de contraseña sean iguales, esto se hace con una estructura de condición simple, en caso de ser diferentes se añade a la variable "error" el mensaje de que las contraseñas no coinciden. El siguiente paso a realizar es aplicar la función hash criptográfica "sha512", esto con la finalidad de cifrar la contraseña ingresada por el usuario y proporcionar mayor seguridad a la cuenta.

A continuación, se intenta establecer la conexión con la base de datos ubicada en el servidor del sitio Web, esto se hace con ayuda del controlador "PDO", el cual intentará conectar con la base de datos señalada, en este caso la base de datos se llama "calidad_aire", haciendo uso del nombre de usuario y contraseña indicados. Si se produce algún error al establecer la comunicación con la base de datos se mostrará en pantalla un mensaje que le indique al usuario que no fue posible acceder a la base de datos y se suspenderá el proceso de creación de la cuenta. En caso de que la conexión con la base de datos haya sido exitosa se realiza una consulta a la tabla "users" (cuya estructura se ilustra en la figura 4.61) preguntando si ya existe algún registro con el nombre de usuario de la cuenta que se esta intentando crear, si ya hay algún registro con ese nombre de usuario se agrega a la variable "error" un mensaje indicando que ese nombre de usuario ya no está disponible. Si la consulta no arroja ningún resultado significa que el nombre de usuario está disponible, así que se repite el proceso anterior, pero esta vez preguntando por el correo electrónico.

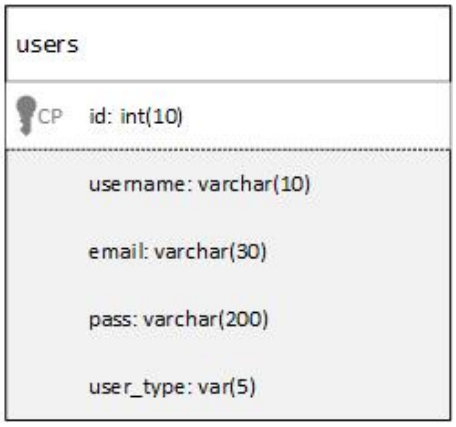


Figura 4.61: Modelo de la tabla "users" de la base de datos "calidad_aire".

De la figura anterior se obtienen la tabla 4.3 de definiciones de la tabla de usuarios de la base de datos.

Tabla 4.3: Tabla de definiciones para la tabla "users" del sitio web

Nombre	Tipo	Null	Llave	Descripción
Id	Int(10)	No	Primaria	Identificador para cada uno de los registros de la tabla.
username	Varchar(20)	No	No aplica	Nombre de usuario de la cuenta.
email	Varchar(30)	No	No aplica	Correo electrónico asociado a la cuenta de usuario.
pass	Varchar(200)	No	No aplica	Contraseña cifrada de la cuenta de usuario.
user_type	Varchar(5)	No	No aplica	Tipo de usuario. (Este campo determina los privilegios que tendrá el usuario sobre la base de datos)

Antes de guardar la información de la cuenta en la base de datos se examina la variable "error", si esta contiene algún mensaje significa que se presentaron errores en el proceso de creación de la cuenta de usuario, de ser así se redirecciona al visitante a la página de registro indicándole cuales son ellos errores que se presentaron. Si no se han presentado errores en la creación de la cuenta de usuario se realiza la consulta SQL para ingresar el nombre de usuario con su respectiva contraseña y correo electrónico a la base de datos. Además de estos datos que han sido ingresados por el usuario, a la cuenta se le asigna un tipo de usuario "user", existen dos tipos de usuarios los usuarios comunes (user) y los usuarios administradores (admin), la asignación de los usuarios administradores la lleva a cabo el administrador del servidor modificando directamente la base de datos. Ya que se ha guardado la información de la cuenta en la base de datos se inicia una sesión PHP con la función "sesion_start()" a la cual se le asigna el nombre de la cuenta creada por medio del parámetro "username" del arreglo de sesión.

Finalmente se envía un correo al correo electrónico de la cuenta creada utilizando la función "mail()", este correo cuenta con un mensaje de bienvenida al sitio Web, una vez enviado el correo se redirecciona al usuario a la página de cuenta creada.

Páginas iniciar sesión y cerrar sesión

Esta página es cargada cada vez que el usuario presiona el botón de iniciar sesión en cualquiera de las páginas de contenido del sitio Web, y su función es verificar que el nombre de usuario y contraseña ingresados por el usuario correspondan a una cuenta registrada en la base de datos y de ser así se inicia sesión en el sitio Web. Lo primero que se hace es guardar la contraseña y el nombre de usuario ingresados por el usuario en variables PHP, además, se crea una variable llamada "errores" que es la encargada de indicar cuando se ha producido un error en el proceso de inicio de sesión. A continuación,

se aplican filtros de saneamiento a las variables de nombre de usuario y contraseña, esto para evitar la inyección de código al sitio Web, una vez saneadas estas variables se aplica la función criptográfica "sha512" a la contraseña.

El siguiente paso es intentar establecer conexión con la base de datos, si ocurre alguna falla durante este proceso se muestra en pantalla el error ocurrido y se detiene el proceso de inicio de sesión. Cuando se establece satisfactoriamente la conexión con la base de datos se ejecuta una consulta SQL en busca de un registro cuyo nombre de usuario y contraseña coincidan con los datos introducidos por el usuario, en caso de que la consulta no proporcione ningún resultado se le agrega a la variable "errores" un mensaje indicando que el nombre de usuario o la contraseña son incorrectos y se redirecciona al usuario a la página desde la cual se intento hacer inicio de sesión mostrando en pantalla el contenido de la variable "errores". Si la consulta detecta que existe un registro que coincide con los datos proporcionados por el usuario se inicia una sesión PHP, y se establecen los parámetros "username" y "user_type" con los valores obtenidos de la consulta. Finalmente se implementa una estructura condicional para identificar a que tipo de usuario corresponde la sesión iniciada, en caso de ser un usuario tipo "user" se hará un redireccionamiento hacia la página de inicio del sitio Web con la única diferencia de que el menú de inicio de sesión será reemplazado por el menú de cerrar sesión y se mostrará el nombre de usuario de la sesión iniciada ver figura 4.62. Por otra parte, si el usuario es de tipo "admin" el redireccionamiento se hará hacia la página de administración de la base de datos y, en la barra de navegación se agregará la opción de la página de administración.

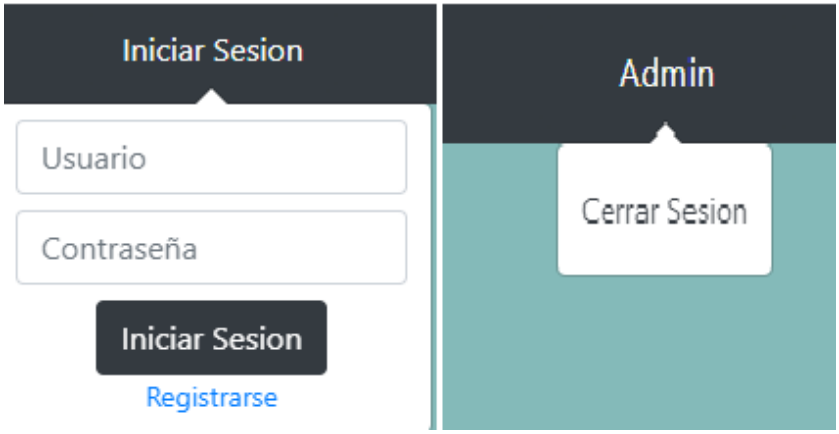


Figura 4.62: Menú desplegable antes y después de iniciar sesión.

Cuando se ha iniciado sesión y se hace clic sobre el botón "cerrar sesión" se realiza un redireccionamiento hacia la página "cerrar sesión" que es la encargada de dar por terminada una sesión de usuario, esto se logra haciendo uso de la función "sesion_destroy()"

propia de PHP, finalmente se redirecciona al usuario hacia la página de inicio del sitio Web.

Página guardar mediciones

Esta página es la encargada de recibir los datos obtenidos de las mediciones de los gases contaminantes del aire enviados desde la aplicación móvil. Para esto se hace uso de un programa en PHP, el cual se encarga de atender las peticiones realizadas por la aplicación móvil, así como de ejecutar la sentencia SQL que almacenen los datos proporcionados por la aplicación móvil en la base de datos del servidor web. En la figura 4.63 se ilustra el diagrama del procedimiento llevado a cabo para almacenar la información en la base de datos del servidor web.

De la figura anterior, se desarrolla la tabla 4.4, que define todas las características de las variables en la base de datos del sitio Web.

Tabla 4.4: Definiciones de las variables de la tabla "mediciones"

Nombre	Tipo	Null	Llave	Descripción
ID	Int(10)	No	No aplica	Identificador para los registros de la tabla.
Latitud	Varchar(10)	No	No aplica	Latitud del punto de medición.
Longitud	Varchar(10)	No	No aplica	Longitud del punto de medición
Fecha	Varchar(10)	No	No aplica	Fecha en que se realizó la medición
Hora	Varchar(8)	No	No aplica	Hora en que se realizó la medición.
CO	Varchar(5)	No	No aplica	Índice IMECA de CO obtenido.
SO_2	Varchar(5)	No	No aplica	Índice IMECA de SO_2 obtenido.
NO_2	Varchar(5)	No	No aplica	Índice IMECA de NO_2 obtenido.
O_3	Varchar(5)	No	No aplica	Índice IMECA de O_3 obtenido.

El proceso de almacenamiento de los datos comienza estableciendo una conexión con la base de datos, en caso de que esta conexión falle se envía un mensaje a la aplicación móvil indicando que no se ha podido realizar el almacenamiento de la información. Si la conexión con la base de datos es satisfactoria se verifica si el método empleado para enviar los datos ha sido el método POST, si este no ha sido el método empleado se cancela el proceso y se notifica a la aplicación de que ha ocurrido un error. Cuando los datos han sido enviados utilizando el método POST se crean variables PHP que almacenen cada uno de los datos recibidos, correspondientes a los índices IMECA de los gases medidos, las coordenadas del lugar donde se hizo la medición y la fecha y hora

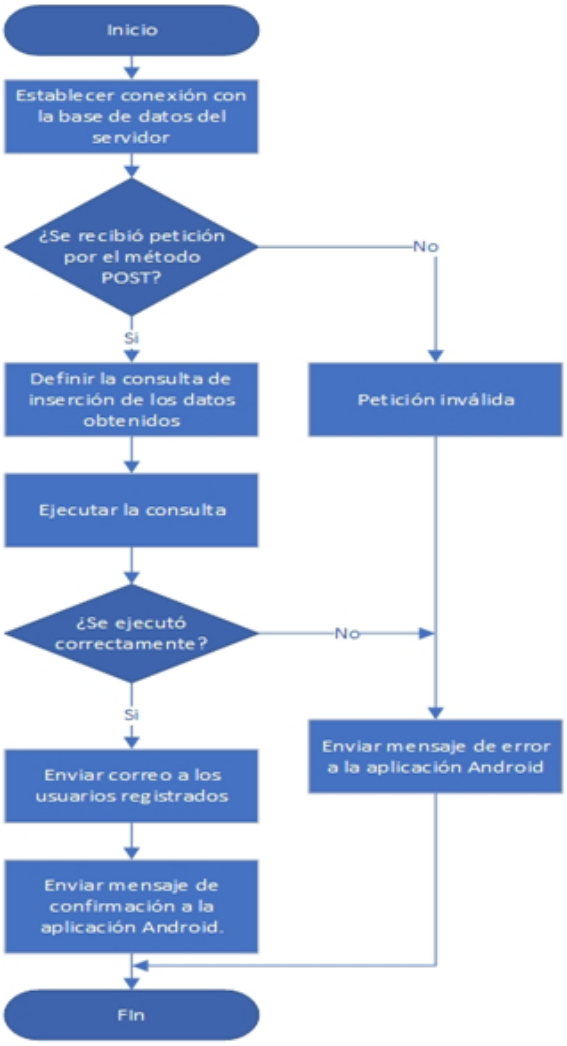


Figura 4.63: Diagrama de flujo del programa PHP de la página "guardar mediciones"

en que esta fue realizada.

A continuación, se realiza una consulta SQL para insertar los valores recibidos por la aplicación en su correspondiente columna de la tabla "mediciones", si se detecta que ha ocurrido algún error al realizar la inserción de los datos se notifica a la aplicación móvil que ocurrió un problema al guardar la información. Si los datos son guardados satisfactoriamente se envía un mensaje de confirmación a la aplicación móvil y se realiza un redireccionamiento hacia la página "correo".

Correo de aviso de actualización

Cada vez que se agregan nuevos registros de mediciones a la base de datos se realiza un redireccionamiento a esta página, ya que es la encargada de enviar un mensaje a todos los usuarios registrados por medio de correo electrónico notificándoles que se han actualizado los registros de las mediciones de los gases contaminantes del aire. Esta tarea comienza estableciendo la conexión con la base de datos, si no es posible establecer la comunicación con la base de datos se cancela el proceso de enviar los correos electrónicos. Una vez establecida la comunicación con la base de datos se realiza una consulta SQL para obtener todos los nombres de usuario y su respectivo correo electrónico de la table "users", como resultado de esta consulta se tiene un arreglo con los parámetros "username" y "email" que contienen todos los nombres de usuario y correos electrónicos registrados en la base de datos. Lo que se hace es recorrer el arreglo y por cada elemento de este se envía un correo electrónico a la dirección contenida en el atributo "email", el mensaje enviado incluye un saludo personalizado para cada usuario y un enlace al sitio Web. El mensaje recibido por el usuario cada vez que se realiza una actualización en la base de datos se muestra en la figura 4.64.

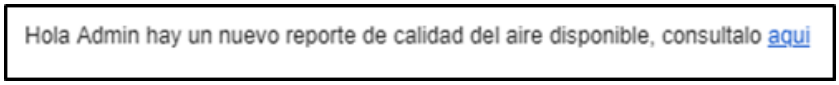


Figura 4.64: Correo de notificación de nuevo reporte disponible.

Página borrar

Esta página es la encargada de eliminar los registros de la tabla "mediciones" seleccionados por el administrador desde la página de administración de la base de datos. El proceso para eliminar los registros seleccionados consiste en establecer comunicación con la base de datos, en caso de que esta comunicación falle se mostrará en pantalla un mensaje indicando el error ocurrido y se detendrá el proceso de eliminación de registros. Si se establece la conexión con la base de datos de manera satisfactoria se almacena

el “id” recibido correspondiente al registro que se desea eliminar en una variable PHP y, a continuación, se ejecuta una consulta SQL “DELETE”, especificando que se desea eliminar el registro con el ID indicado de la tabla “mediciones”, una vez ejecutada la consulta se muestra en pantalla un mensaje indicando que se ha borrado el registro de manera satisfactoria.

Creación de la tabla de mediciones para la página “consulta”

A pesar de que la página de consulta pertenece a la sección de front-end requiere del desarrollo del lado del servidor para poder mostrar en pantalla la tabla de las mediciones efectuadas, ya que esta información se encuentra en la base de datos. Es por lo anterior que para mostrar correctamente el contenido de la página “consulta” se ha implementado un programa en PHP, este programa se encarga de acceder a la base de datos y mediante una consulta SQL se obtiene una variable que contiene todos los datos de las mediciones realizadas, (índices IMECA, coordenadas fecha y hora de las mediciones). Una vez que se tiene este arreglo con toda la información de las mediciones, se implementa un ciclo para recorrer dicho arreglo y, por cada elemento de este se crean elementos celda (td) a los cuales se les asignan los valores de las mediciones guardados en el arreglo antes mencionado, también es por medio de este programa en PHP que se les asigna una clase a cada uno de los elementos celda para que puedan ser identificados por el programa en JavaScript explicado en la sección de front-end. La función PHP encargada de crear la tabla se ilustra en la figura 4.65, donde “$resultado” es el arreglo que contiene todos los datos de las mediciones, este programa solo se encarga de crear la tabla, la aplicación de estilos se realiza en el lado del cliente con JavaScript.

```
<?php
                    while($datos = $resultado->fetch_array()){
                      ?>
                        <tr>
                          <td class="ID"><?=$datos['id']?></td>
                          <td class="latitud"><?=$datos['latitud']?></td>
                          <td class="longitud"><?=$datos['longitud']?></td>
                          <td ><?=$datos['fecha']?></td>
                          <td><?=$datos['hora']?></td>
                          <td class="co"><?=$datos['co']?></td>
                          <td class="so2"><?=$datos['so2']?></td>
                          <td class="o3"><?=$datos['o3']?></td>
                          <td class="no2"><?=$datos['no2']?></td>
                        </tr>
                        <?php
                    }
                    ?>
```

Figura 4.65: Programa PHP encargado de crear la tabla de resultados mostrada en la página "consulta".

Capítulo 5

Análisis de resultados y pruebas

5.1. Comparación con estaciones de monitoreo fijas

En las siguientes tablas se muestra la fecha, hora, ubicación y resultado promedio de cada uno de los gases medidos por él prototipo en la Unidad Zacatenco del IPN. Se presentan en 5 posibles casos donde cada uno hacer referencia a un estado del aire: El color verde hace referencia a un estado bueno, el color amarillo para hacer referencia a un estado regular, el color naranja hace referencia a un estado malo, el color rojo hacer referencia a estado muy malo y el color morado hacer referencia a un estado extremadamente malo. Los estados de cada uno de los gases se clasifican siguiendo las referencias que brinda el índice Metropolitano de la Calidad del Aire (IMECA).

Tabla 5.1: Mediciones realizadas el 12/10//2018 en el horario de 13:00 a 14:00

ID	Fecha	Hora	CO	SO_2	O_3	NO_2
1	2018-10-12	13:00	3.95	29.58	6.78	75.54
2	2018-10-12	13:20	2.15	37.54	58.79	31.80
3	2018-10-12	13:30	1.76	40.96	67.84	40.20
4	2018-10-12	13:40	1.46	36.41	64.07	36.76

Al analizar las tablas 5.1 y 5.2, se observa que los datos recabados por el prototipo, en ocasiones son menores a los que muestra la página de la CDMX, debido que en la medición influyen varios factores. Las estaciones fijas, se ubican en puntos estratégicos (longitud, latitud, altura), en los cuales se considera que es donde hay mas presencia de gases tóxicos, y a diferencia del prototipo, las mediciones se realizaron a una altura promedio de 1.5 metros del suelo, en zonas alejadas de vehículos o de algún otro objeto que se considere, emite contaminantes en grandes proporciones.

Tabla 5.2: Datos obtenidos del sitio oficial de la CDMX respecto a la estación

ID	Fecha	Hora	CO	SO_2	O_3	NO_2
1	2018-10-12	13:00	34	3	12	5
2	2018-10-12	13:20	42	3	12	5

5.2. Análisis de ruta

Una vez que el prototipo está funcionando correctamente, se procede a realizar pruebas en diferentes puntos de la ESIME Zacatenco, esto con la finalidad de recabar datos para su posterior análisis. Para definir los puntos en los cuales se realizaron las mediciones se tomó en cuenta la cantidad de gente que suele circular por esas zonas, así como la cercanía con automóviles, ya que estos representan una fuente emisora de gases contaminantes del aire. Tomando en cuenta lo anterior, los puntos en los que se decidió realizar las mediciones de los gases contaminantes son: la zona de estacionamientos del edificio cinco de la ESIME Zacatenco, el pasillo intermedio que une a los edificios, el pasillo central ubicado entre los edificios de salones y el edificio Z, la planta baja del edificio Z a la altura del edificio cinco de la ESIME, la cafetería número cuatro y la plaza roja.

La metodología empleada para obtener los niveles de contaminación presentes en el aire consistió en realizar un total de diez mediciones en cada uno de los puntos ya mencionados, como resultado de esto se obtuvo el siguiente mapa. A pesar de que las mediciones fueron realizadas en un mismo lugar, en la figura 5.1 se aprecia que los puntos de mediciones se encuentran dispersos, esto se debe a que el sistema de localización del teléfono inteligente no es muy preciso, por lo que al realizar las mediciones los valores de latitud y longitud se veían modificados, aun cuando el teléfono inteligente no se hubiese movido. Esta variación en las coordenadas registradas de los puntos de medición puede considerarse como no significativa, ya que en la mayoría de los casos esta variación no va más allá de un par de metros por lo que no afecta de manera considerable a las mediciones realizadas.

De la figura 5.1 también se tiene que la mayoría de las mediciones realizadas en la zona del estacionamiento y los pasillos reportaron una calidad del aire buena, siendo pocas las mediciones en las que se reportó una calidad de aire regular o mala, lo cual puede atribuirse a ligeras variaciones en los voltajes proporcionados por los sensores. Por otra parte, se tiene que en la zona de la cafetería se tiene una mayor tendencia a registrar una calidad del aire regular, siendo el ozono el gas que presenta mayores valores de índices IMECA. Este aumento del índice IMECA en la zona de la cafetería es un factor que debe tenerse muy en cuenta, ya que de las zonas en las que se realizaron las mediciones

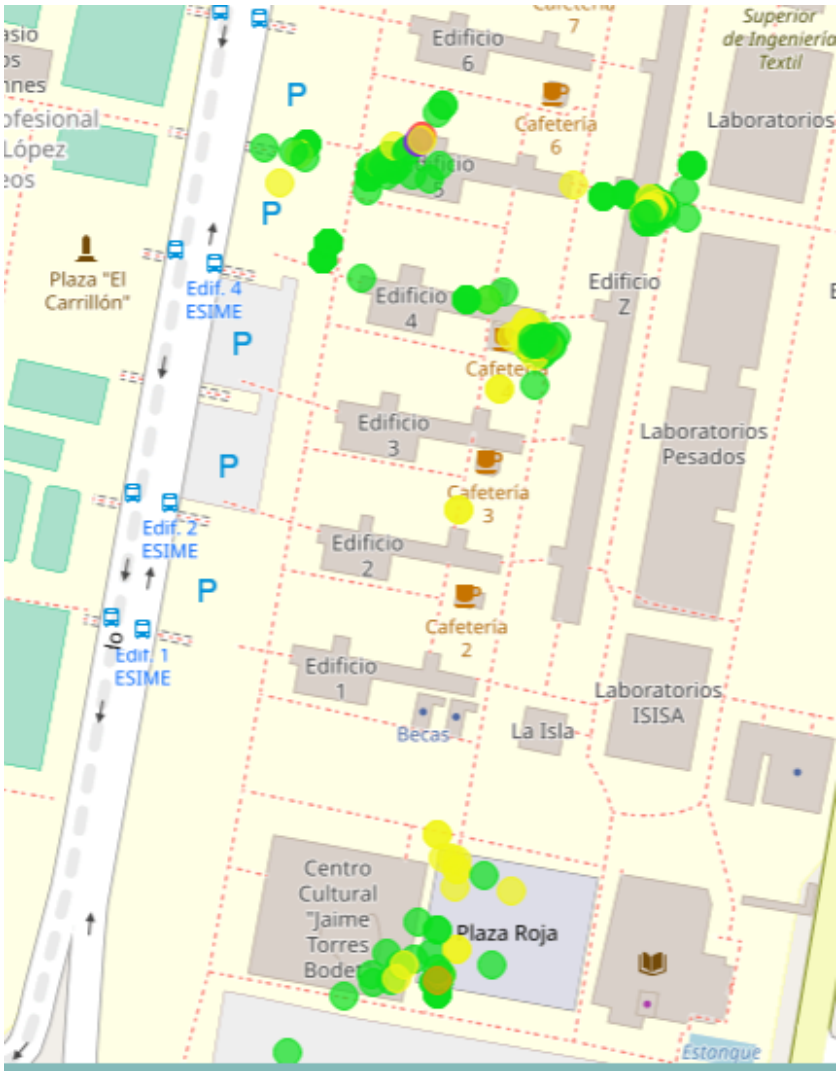


Figura 5.1: Mapa de los puntos de medición.

esta es en la que hay una mayor concentración de personas. Finalmente, en la zona de la plaza roja se tienen registros de altos niveles de ozono en el aire, lo que ocasiona que haya un número considerable de mediciones que reportan una calidad del aire regular, sin embargo la tendencia sigue siendo una calidad del aire buena.

5.3. Duración de la batería

Para determinar el tiempo de duración de la batería es necesario considerar el consumo de corriente total del prototipo y los A/h proporcionados por la batería. El consumo de corriente del circuito se describe en la siguiente tabla:

Tabla 5.3: Tabla de consumos de corriente de los elementos del prototipo.

Elemento	Consumo de corriente
Raspberry Pi 3	2.6 A
Sensores Alphasense A4	3.25 mA
Divisor de voltaje	3.6 mA
ADC Differential Pi	100 mA
Total	2.606 mA

De la tabla anterior hay que tener en cuenta que la corriente considerada para la Raspberry Pi es para el caso en que se encuentre ejecutando únicamente el programa que controla los sensores, tenga activada la conexión Bluetooth y no cuente con dispositivos periféricos conectados (tales como pantalla, teclado o ratón), para cualquier otro caso el consumo de corriente puede variar. Para el caso de los sensores se ha considerado la corriente requerida para cinco sensores, ya que la placa AFE cuenta con un sensor de temperatura, que si bien no es empleado en el desarrollo del proyecto sigue representando un consumo de energía en el prototipo. En cuanto a la corriente demandada por los divisores de voltaje se tiene que esta varía dependiendo del voltaje proporcionado por los sensores, el valor indicado en la tabla corresponde al caso en el que los sensores tengan un voltaje de salida de 5v, ya que es cuando mayor demanda de corriente hay en el circuito. Una vez que se ha determinado el consumo de corriente del prototipo, basta con dividir la capacidad de la batería utilizada, que en este caso es de 12 A/h, entre la corriente demandada para obtener el tiempo de funcionamiento del prototipo, tal y como se muestra en la siguiente ecuación.

$$t = \frac{Capacidad\ de\ la\ bateria\ en\ mA/h}{Consumo\ del\ prototipo\ en\ mA/h} : \text{Ecuación para obtener la duración de la batería.} \tag{5.3.1}$$

$$t = \frac{12000\ mA/h}{2{,}606\ mA/h} = 4{,}6 horas. \tag{5.3.2}$$

5.4. Análisis de costos

El prototipo final involucra diversos dispositivos electrónicos y elementos eléctricos, los cuales son listados a continuación con su respectivo precio:

Tabla 5.4: Costos sobre los materiales utilizados en el prototipo.

Material	Costo
Raspberry Pi 3 modelo B	$ 1172.00
Memoria Kingston 16 GB Class 4 MicroSD Flash Card	$ 150.00
Kit de sensores Alphasense A4 modelo 810-0023-00	$ 12,600.00
ADCDifferentialPi	$ 510.00
Placa fenólica de 10x10cm	$ 10.00
Caja de acrílico	$ 1068.00
Batería de 12000 mAh	$ 549.00
12 Resistencias de 1000 Ohm	$ 1.00 c/u
4 Resistencias de 2000 Ohm	$ 1.00 c/u
4 Resistencias de 100 Ohm	$ 1.00 c/u
Costo total	**$ 16,079.00**

Conclusiones

Se desarrolló un sistema móvil de medición de gases tóxicos (específicamente CO, NO_2, O_3 y SO_2) para informar a la comunidad local sobre los niveles de concentración de estos gases en el aire en una zona mediante un sitio Web en el que se muestren los datos obtenidos. El sistema presentado utiliza 4 sensores de gases tóxicos de la marca Alphasense, que fueron caracterizados mediante pruebas con fuentes de estos gases, para conocer su comportamiento durante diferentes casos. Se implementó la Raspberry Pi 3 modelo B, para convertir al índice IMECA, los datos obtenidos de los sensores de gases tóxicos mediante un ADC.

Este subsistema se controla mediante un teléfono inteligente que además almacena los registros de acuerdo a la hora, fecha y lugar en el que se realizó la medición. Finalmente estos datos se cargan en una base de datos de un sitio Web desde el el cual se muestran los datos en un mapa geográfico, el cual representa las mediciones realizadas mediante indicadores de color, que identifican si existe algún componente tóxico en abundancia, incluso permite borrar registros de la base de datos mediante una cuenta de administrador.

El sistema completo cumple el objetivo de brindar información sobre el estado del aire respecto a la concentración de gases tóxicos, sin embargo este requiere de una conexión a internet para subir las mediciones realizadas a la base de datos del sitio Web, por lo que no entra de la categoria de sistema de monitoreo.

Respecto al funcionamiento del prototipo, este necesita de una batería para funcionar de manera móvil, así como de un teléfono inteligente con sistema operativo Android, comunicación Bluetooth y servicio ubicación. Cabe destacar que el teléfono inteligente debe ser emparejado con la Raspberry Pi 3, utilizando la Raspberry Pi 3 como una computadora de escritorio, ya que es necesario hacerla visible ante los demás dispositivos Bluetooth y después deshacer la visibilidad, esto porque si se deja activada la visibilidad, la energía que demanda la Raspberry Pi es mayor y disminuiría la duración de la batería.

Trabajo futuro

El prototipo presenta ciertas desventajas ante estaciones fijas como lo son la precisión y la exactitud, esto debido a que los sensores utilizados son de gama baja, por lo que uno de los principales objetivos para mejorar el prototipo es implementar sensores con mejores prestaciones, pero que mantengan un prototipo compacto y ligero, de igual manera también es necesario cambiar el ADC por uno con mayor número de canales y que soporten una cantidad mayor de voltaje a la entrada, para evitar cualquier riesgo de sobrecarga y daños al prototipo. En lo referente al mayor número de canales, se debe a que, faltan los sensores de partículas suspendidas $PM_{2,5}$ y PM_{10}, así como también un sensor de temperatura y uno de presión atmosférica, ya que las mediciones de gases y partículas, requieren una corrección debida a estas dos variables.

Una forma de hacer más eficiente es implementando un módulo de GPS de altas prestaciones para mejorar la exactitud del posicionamiento.

Glosario

Activity: Es cualquier cosa que un usuario puede hacer en una aplicación Android, generalmente son presentadas al usuario como ventanas de pantalla completa.

ADC: Dispositivo electrónico que convierte señales analógicas en señales digitales a través de la cuantificación y la codificación.

Android: Es un sistema operativo basado en Linux, diseñado para dispositivos móviles de pantalla táctil como teléfonos inteligentes, tablets, por mencionar algunos.

***Arrayadapter*:** Es un tipo de adaptador empleado en aplicaciones Android, su función es retornar un elemento visual por cada objeto en una colección de objetos de datos. Puede ser utilizado con widgets basados en listas como los elementos "*ListView*" y "Spinner".

Atenuación: Se define como la pérdida de potencia de una señal que transita por un medio de transmisión.

Back-end: También conocido como motor, es un término que hace referencia a la sección de un sitio Web que tiene comunicación con el servidor, esta sección no es visible para los usuarios comunes, siendo el administrador del sitio Web el único con acceso a esta sección.

Bluetooth: Protocolo de comunicación para compartir información de forma inalámbrica mediante una radiofrecuencia de 2.4GHz, en un espacio relativamente pequeño de en promedio unos 10 metros.

Broadcast: Un broadcast o difusión amplia hace referencia a la transmisión de datos que serán recibidos por todos los dispositivos que formen parte de una red.

C/C++: Lenguaje de programación de alto nivel, cuyo sistema ha sido la base para el desarrollo de otros lenguajes de programación.

Carburante: Combustible líquido, formado por hidrocarburos. Se utiliza en los motores de explosión y de combustión interna.

Corriente Eléctrica: Es el flujo de carga eléctrica (electrones) que recorren un circuito, sus unidades de medida son los amperes [A].

Dióxido de azufre (SO_2): Gas conformado por un átomo de azufre y dos de oxígeno, es un compuesto incoloro, irritante con un olor penetrante a partir de los 0.3 ppm.

Dióxido de nitrógeno (NO_2): Gas conformado por un átomo de nitrógeno y uno de oxigeno resultado de las emisiones de los automóviles y que son dañinas para la salud.

Dirección MAC: La dirección MAC (Media Access Control), o dirección física de un dispositivo es un número de 48 bits representado por seis bloques de dos caracteres hexadecimales cada uno, el cual identifica a una tarjeta única de red.

Entorno de desarrollo integrado: Un entorno de desarrollo integrado (IDE por sus siglas en inglés) es un conjunto de herramientas integradas que tienen como finalidad el desarrollo de software.

EPA: Agencia de protección del medio ambiente es la agencia federal de los Estados Unidos que tiene como misión el proteger la salud humana, así como el medio ambiente (agua, aire y suelo).

EPSG (European Petroleum Survey Group): Son un conjunto de datos de sistemas de coordenadas y proyecciones cartográficas utilizados en la definición de datos de posición en sistemas de información geográfica.

Funciones Hash: Son funciones criptográficas que cifran una entrada, comprimiéndola a una salida de menor longitud. Estas funciones son usadas en sistemas de contraseñas debido a su resistencia frente a ataques maliciosos.

Front-end: También conocido como interfaz, es un término que hace referencia a la sección de un sitio Web que es visualizada por el usuario.

GPIO: Sistema de entrada y salida de propósito general que consta de una serie de pines incorporados en la Raspberry pi.

GPU: Unidad de Procesamiento Gráfico que puede aligerar la carga de información que se debe procesar por la unidad central, y esta lo pueda realizar de manera más eficiente.

Gradle: Paquete de herramientas de compilación avanzadas utilizadas en la automatización y administración del proceso de compilación de un programa.

GSM: Sistema de radiotelefonía celular digital.

I2C: Protocolo que permite la comunicación entre dos dispositivos que tengan integrador el mismo bus.

IEEE (Instituto de Ingeniero Eléctricos y Electricistas): Es una asociación técnico-profesional mundial encargada de la estandarización y búsqueda de información de nuevos avances en el área de la electrónica y la electricidad.

IMECA (Índice Metropolitano de la Calidad del Aire): Es una herramienta desarrollada para el análisis y difusión de los niveles de contaminación, de manera rápida y sencilla.

Instancia: Es la particularización de una clase, entidad o prototipo. En lenguajes de programación orientada a objetos se tiene que un objeto es una instancia de una clase, es decir que es miembro de una clase y tiene atributos en lugar de variables.

Intent: Es un objeto de acción el cuál puede ser utilizado para solicitar una acción de otro componente de la aplicación, como puede ser iniciar una nueva actividad, iniciar un nuevo servicio o mandar un mensaje.

Latitud: Es la distancia angular que hay desde un punto de la superficie de la Tierra hasta el paralelo del ecuador, esta distancia se mide en grados, minutos y segundos sobre los meridianos.

LCD: Pantalla de cristal líquido, es una pantalla de pequeñas dimensiones que cuenta con determinado numero de pixeles ya sea de color o monocromáticos insertados frente a una fuente de luz reflectora.

Listener: Es una interfaz encargada de detectar y manejar aquellos eventos que ocurren cuando se produce una acción sobre un elemento del programa.

***ListView*:** Es un grupo de vistas que muestra una lista de elementos desplazables.

Longitud: Es la distancia angular de un punto de la superficie terrestre al meridiano de Greenwich, se mide en grados, minutos y segundos.

Método POST: Es un método perteneciente al protocolo HTTP usado para enviar información desde front-end para que esta sea procesada y actualice o agregue información en el servidor.

Microcontrolador: Circuito programable que cuenta en su interior con 3 bloques principales para su funcionamiento unidad central de procesamiento, memoria, periféricos.

Microprocesador: Circuito integra central encargado de ejecutar instrucciones programadas en lenguaje de bajo nivel.

Monóxido de carbono (*CO*): Gas conformado por un átomo de carbono y uno de oxígeno, es un compuesto toxico, insípido y soluble al agua. Es el resultado de la oxidación incompleta del carbono durante en el proceso de combustión.

Multiplataforma: Software que puede ser ejecutado en cualquier plataforma sin preparación especial.

OMS: Organización mundial de la salud es un organismo encargado de gestionar políticas de prevención, promoción e intervención de la salud a nivel mundial.

ppb (Partes por billón): Es una unidad de medida con la cual se mide el nivel de concentración de una sustancia por cada billón de unidades de un conjunto.

ppm (Partículas por partes por millón): Es una unidad de medida con la cual se mide el nivel de concentración de una sustancia por cada millón de unidades de un conjunto.

Presión: Magnitud física que da como resultado la fuerza ejercida en algún punto de un material.

Protoboard: Tarjeta perforada que cuenta en su interior con un patrón de pistas conductoras de corriente.

Python: Lenguaje de programación interpretado, orientado a objetos, multiplataforma y de propósito general.

Raspberry Pi: Es una microcomputadora con los periféricos necesarios para poder ser operada.

Reacción Redox: También denominado reacción de oxidación-reducción, es un tipo de reacción química en la cual se transfieren electrones produciendo un cambio en el número de oxidación entre los reactivos y los productos.

Resistor: Componente electrónico que se encarga de la oposición al flujo de electrones al moverse por un conductor.

Sensor: Dispositivo electrónico que responde a una secuencia de comportamientos externos.

Sha512: Es un algoritmo de hash seguro de 512 bits.

SIMAT: Sistema de Monitoreo de la Ciudad de México, es la secretaria encargada del monitoreo y vigilancia de la calidad del aire en la Ciudad de México.

Sistema operativo (S.O.): Conjunto de programas informáticos hechos para la ejecución de varias tareas.

Socket: Hace referencia a un concepto abstracto por el cual dos programas pueden intercambiar cualquier flujo de datos de manera fiable y ordenada. Los sockets constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados, está definido por un par de direcciones IP, una local y una remota, un protocolo de transporte y un par de números de puerto (local y remoto).

Spinner: Es un elemento visual usado en aplicaciones Android que muestra un elemento de una lista a la vez y que permite al usuario escoger el elemento de la lista que desea mostrar.

Temperatura: Magnitud física que proporciona la cantidad de calor que tiene un cuerpo, un objeto, entre otros.

Toast: Es un tipo de notificación disponible en aplicaciones Android el cual proporciona una realimentación acerca de una operación realizada, esto a través de un pequeño cuadro de texto.

UUID: El Identificador Único Universal (UUID por sus siglas en inglés), es un número de 16 bytes expresado por 32 dígitos hexadecimales dividido en cinco grupos separados por guiones, es usado para identificar información en sistemas de computación.

VNC: Software libre con estructura cliente-servidor, que permite observar las acciones remotamente a través de un ordenador cliente.

Voltaje: Energía potencial por unidad de carga, medida en Volts.

WPAN (Red de Área Personal Inalámbrica): Es una red de comunicación inalámbrica para distintos dispositivos en un rango de unos pocos metros y para uso personal.

Bibliografía

[1] *Bootsrap Introduction*. URL: https://getbootstrap.com/docs/4.0/getting-started/introduction/. (accessed: 12.10.2018).

[2] C.J. Date. *Introducción a los Sistemas de Bases de Datos*. PEARSON EDUCACIÓN, Addison-Wesley Publishing Company, 2001. ISBN: 968-444-419-2.

[3] *HTML5 w3schools*. URL: https://www.w3schools.com/html/. (accessed: 12.10.2018).

[4] *Índice Metropolitano de la Calidad del Aire (IMECA)*. URL: https://es.wikipedia.org/wiki/%C3%8Dndice_Metropolitano_de_la_Calidad_del_Aire. (accessed: 27.09.2018).

[5] *Introducción a CSS*. URL: https://librosweb.es/libro/css/. (accessed: 12.10.2018).

[6] *Javascript w3schools*. URL: https://www.w3schools.com/js/. (accessed: 12.10.2018).

[7] *JQuery*. URL: https://jquery.com. (accessed: 12.10.2018).

[8] P. McDermott-Wells. «What is Bluetooth?» En: *IEEE Potentials* 23.5 (2004), págs. 33-35. DOI: https://ieeexplore.ieee.org/abstract/document/1368913/authors#authors.

[9] *NORMA Oficial Mexicana NOM-020-SSA1-2014, Salud ambiental*. 2014. URL: http://dof.gob.mx/nota_detalle.php?codigo=5356801&fecha=19/08/2014. (accessed: 18.10.2018).

[10] *NORMA Oficial Mexicana NOM-022-SSA1-2010, Salud ambiental*. 2010. URL: http://dof.gob.mx/nota_detalle.php?codigo=5158348&fecha=08/09/2010. (accessed: 18.10.2018).

[11] *NORMA OFICIAL MEXICANA NOM-023-SSA1-1993, SALUD AMBIENTAL*. 1994. URL: http://www.salud.gob.mx/unidades/cdi/nom/023ssa13.html. (accessed: 18.10.2018).

[12] *NORMA Oficial Mexicana NOM-025-SSA1-2014, Salud ambiental*. 2014. URL: http://www.dof.gob.mx/nota_detalle.php?codigo=5357042&fecha=20/08/2014. (accessed: 18.10.2018).

[13] *Openlayers-Welcome*. URL: https://openlayers.org. (accessed: 12.10.2018).

[14] *PHP Manual*. URL: http://php.net/manual/es/intro-whatis.php. (accessed: 12.10.2018).

[15] *PROYECTO de Norma Oficial Mexicana NOM-021-SSA1-1993, Salud Ambiental*. 1994. URL: http://dof.gob.mx/nota_detalle.php?codigo=4661315&fecha=18/01/1994. (accessed: 18.10.2018).

[16] Organización Mundial de la Salud. *Calidad del aire y salud, Datos y cifras*. 2018. URL: http://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health. (accessed: 27.09.2018).

[17] *SQLite Home Page*. URL: https://www.sqlite.org/index.html. (accessed: 12.10.2018).

[18] *Tipos de Bases de Datos*. URL: http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf. (accessed: 18.10.2018).

[19] *UM10204: I2C-bus specification and user manual*. 2014. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. (accessed: 26.10.2018).

[20] R. Williams y col. *Air Sensor Guidebook*. 2014. URL: https://cfpub.epa.gov/si/si_public_record_report.cfm?Lab=NERL&dirEntryId=277996&simpleSearch=1&searchAll=air+sensor+guidebook. (accessed: 18.10.2018).