UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE CIENCIA

Departamento de Física

Desarrollo de nodos ambientales para medir exposición personal.

Rodrigo Alejandro Donoso Villegas

Tesis para optar al grado de Magister en Ciencia con Mención en Física.

Profesor Guía:

Yaroslav Ispolatov

Santiago – Chile

2019

RESUMEN

Se ha diseñado y construido un nodo sensor portable para medir concentraciones de MP, NO_2 y O_3 y estimar la exposición personal en algunos microambientes urbanos (hogar, oficina, bus, metro y laboratorio de física). Este nodo utiliza sensores de bajo costo, que representan una innovadora opción a la medición con equipos certificados, pero su salida necesita ser ajustada. Los datos se han utilizado para integrar la exposición personal y encontrar puntos críticos donde las concentraciones son altas, generando estrategias y recomendaciones para reducir la exposición personal.

Palabras claves: Sensores de bajo costo, nodo sensor, contaminación atmosférica, exposición personal, material partículado, dióxido de nitrógeno, ozono.

TABLA DE CONTENIDOS

Introducción 1

Exposición personal 5

Difusión de contaminantes 7

1 Sensores de bajo costo 9

1.1 Metodología de medición 9

1.1.1 Sensores de MP 9

1.1.2 Sensores de gas 12

1.2 Fuentes de error 13

1.2.1 Material Particulado 14

1.3 Sensores seleccionados 17

2 Diseño y Construcción 19

2.1 Construcción 20

2.1.1 Versión completa (versión 2) 20

2.1.2 Versión Reducida (versión 1) 23

2.2 Nube digital 23

3 Ajuste con Referencia 27

3.1 Primer Ajuste 28

3.2 Segundo Ajuste 30

3.2.1 Material Particulado 31

3.2.2 Sensores electroquímicos 34

4 Evaluación de la exposición personal 40

4.1 Material Particulado 42

4.2 Dióxido de Nitrógeno y Ozono 43

4.3 Resultados 45

5 Conclusiones 47

6 Aplicaciones 50

Referencias bibliográficas 53

Anexos 59
Diagramas esquemáticos 60
1 Nodo Sensor 1 60
2 Nodo Sensor 2 62
Programación 64
Especificaciones técnicas de los sensores seleccionados 95
3 Sensor de MP: Sensirion SPS030 95
4 Sensor de MP: Novasensor SDS011 97
5 Sensor de NO_2: Alphasense A43F 98
6 Sensor de O_3 + NO_2: Alphasense OX 99

ÍNDICE DE FIGURAS

1 Emisión de contaminantes criterio. 2
2 Capacidad de ingreso de las partículas de MP al cuerpo humano, según su diámetro aerodinámico. 3
3 Principales fuentes de MP en la Región Metropolitana. 4
4 Fuentes de NO_2 y O_3 y sus impactos en la salud. 5
5 Microambientes, exposición y dosis 6
6 Componentes del sensor Honeywell HPMA115S0-XXX que utiliza el principio de dispersión óptica infrarroja. 10
7 Frente de ondas plano aproximándose a una esfera 11
8 Celdas de los sensores electroquímicos utilizados 12
9 Plantower PMS 5003, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22] 15
10 Novasensor SDS011, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22] 16
11 Sensirion SPS030, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22] 17
12 Componentes del diseño del nodo sensor 20
13 Diseño y conexiones del nodo con soporte de GPS, transmisión de datos mediante conectividad celular y batería. 22
14 Diseño y conexiones del nodo con WiFi 23
15 Interfase gráfica para visualización de datos. 24
16 Nodo Sensor con WiFi, caja IP55, 85x85x50 mm 25
17 Nodo Sensor v2. Caja IP65 200x100x70 mm. 26
18 Instalación de nodos sensores en estación de referencia. Izquierda: La Florida (julio de 2019). Derecha: Las Condes (marzo de 2020) 27
19 Primer ajuste, principales datos medidos. 28

20 Comparación entre el MP medido y la estación de referencia. . . 29
21 Correlación entre los valores de MP medidos por el nodo sensor y la estación de referencia . . . 29
22 Datos medidos en los sensores de gas. . . . 30
23 Datos de MP_{10} medidos. . . . 31
24 Datos de $MP_{2,5}$ medidos. . . . 31
25 Comparación entre el $MP_{2,5}$ medido por la estación de referencia y los sensores nodos . . . 32
26 Comparación entre el MP_{10} medido por la estación de referencia y los sensores nodos . . . 32
27 matriz de correlación de PM_{10} . . . 33
28 Matriz de correlación de $PM_{2,5}$ . . . 34
29 Ajuste multilinear de NO_2 con la estación de referencia . . . 35
30 Correlación entre los datos ajustados de NO_2 y la estación de referencia . . . 37
31 Ajuste multilinear de O_3 con la estación de referencia . . . 38
32 Dispersión lineal del ajuste de O_3 con la estación de referencia 38
33 Nodo sensor en el interior de bolso respirable. . . . 40
34 Tiempo promedio en cada microambiente. . . . 42
35 MP_{10} medido en todos los microambientes visitados. . . . 42
36 $MP_{2,5}$ medido en todos los microambientes visitados. . . . 43
37 NO_2 . . . 44
38 O_3 . . . 44
39 variación de NO_2 y O_3 el día 05 de marzo de 2020. . . . 45
40 Proyecto experimental de sensores en movimiento. . . . 51
41 Nodo sensor instalado en helipuerto de la Torre Entel . . . 52
42 Instalación en camión Scania . . . 52
43 Conexiones del Nodo sensor V1. . . . 61
44 Conexiones del Nodo sensor V2. . . . 62

ÍNDICE DE TABLAS

1	Características de los sensores seleccionados	18
2	Versiones de nodos diseñadas	20
3	Error y tiempo de respuesta de los sensores utilizados	21
4	Parámetros de ajuste para NO_2	36
5	Raíz del error cuadrático medio (RMSE), coeficiente de correlación, Error cuadrático medio (MSE), Error absoluto medio (MAE) del ajuste de NO_2.	36
6	Parámetros de ajuste para O_3	39
7	Raíz del error cuadrático medio (RMSE), coeficiente de correlación, Error cuadrático medio (MSE), Error absoluto medio (MAE) del ajuste de O_3.	39
8	Exposición calculada promedio en cada microambiente	45
9	Componentes Nodo sensor v2.	63
10	Programación del Nodo sensor	64

INTRODUCCIÓN

La contaminación del aire es una de las mayores causas de muerte y enfermedades en el mundo. Sus efectos en la salud van desde el incremento en admisiones y emergencias hospitalarias al incremento del riesgo de sufrir muerte prematura[47].

La organización mundial de la salud estima que globalmente, la contaminación del aire es responsable de:

- 29 % de todas las muertes y enfermedades asociadas al cáncer de pulmón.
- 17 % de todas las muertes y enfermedades asociadas a las infecciones respiratorias agudas.
- 24 % de todas las muertes por ataque cerebral.
- 25 % de todas las muertes por enfermedad coronaria.
- 43 % de todas las muertes por enfermedad pulmonar obstructiva crónica.

Investigaciones recientes, han mostrado que la exposición a altos niveles de contaminación reduce significativamente el rendimiento cognitivo[2][36][49].

Es conocido que respirar aire contaminado incrementa el riesgo de sufrir síndrome de dificultad respiratoria aguda (*Acute Respiratory Distress Syndrome (ARDS)*)[31], que es extremadamente mortal y una de las causas de muerte relacionadas con Covid-19[17]. Múltiples estudios correlacionan los altos niveles de muerte debido a esta pandemia con altos niveles de polución del aire[11][25][28][34][48].

Los contaminantes con mayor evidencia de riesgos de salud incluyen material particulado (MP), ozono (O_3) y dióxido de nitrógeno (NO_2) entre otros[47]. Como se muestra en la figura 1, una de las principales fuentes de estos contaminantes es la industria y la generación eléctrica.

La exposición a material particulado (MP) ha sido asociada a una variedad de riesgos de salud, el MP está clasificado como carcinógeno humano por la Agencia Inter-

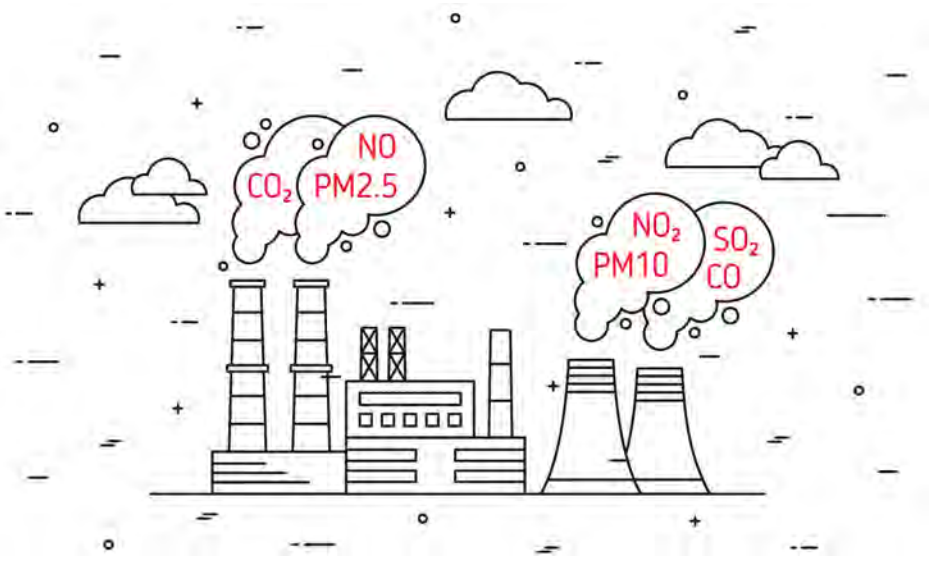


Figura 1: Emisión de contaminantes criterio.

Fuente de la imagen: AleksOrel/Shutterstock.com.

nacional para la Investigación del Cáncer [15], éste es capaz de penetrar profundamente en los pulmones, irritar y comprometer los alvéolos y entrar al flujo sanguíneo causando impactos respiratorios, cardiovasculares y cerebrovasculares [47]. Se debe notar que MP_{10} son las partículas cuyo diametro aerodinámico es menor a $10\mu m$, mientras que $MP_{2,5}$ son las partículas cuyo diámetro aerodinámico es menor a $2,5\mu m$, esto significa que el $MP_{2,5}$ es un subconjunto del MP_{10}. En la figura 2 se puede observar una comparación de la capacidad de ingreso al cuerpo humano según diámetro aerodinámico.

Las mayores fuentes de MP en una zona urbana, son producidas por el transporte, la industria incluyendo la generación eléctrica a partir de combustibles fósiles, la combustión dentro del hogar y algunas fuentes naturales como las sales marinas y la corteza terrestre [18]. En la Región Metropolitana, las principales fuentes son las observadas en la figura 3.

El O_3 en su mayoría, no se emite directamente, sino que se forma en la atmósfera como resultado de complejas reacciones fotoquímicas entre contaminantes precursores como óxidos de nitrógeno y compuestos orgánicos volátiles[35], es fuertemente oxidante, sus altas concentraciones en el ambiente y variabilidad se asocian con la urbanización e industrialización[29]. La exposición a altos niveles de O_3 reduce la capacidad pulmonar,

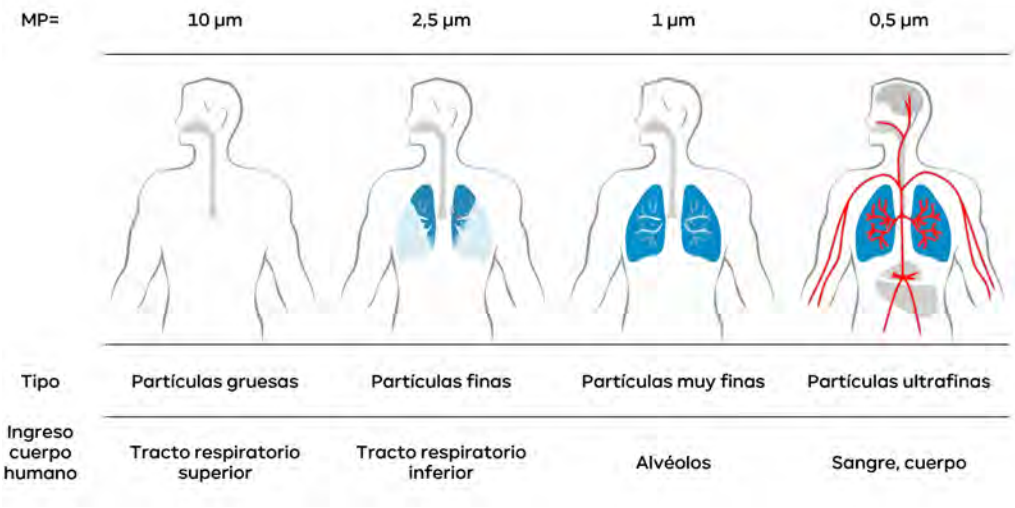


Figura 2: Capacidad de ingreso de las partículas de MP al cuerpo humano, según su diámetro aerodinámico.

Adaptada de: https://www.encyclopedie-environnement.org/app/uploads/2019/04/Air-pollution_fig2-lung-penetration-particles-1.jpg (Última visita: julio de 2020).

produce daño genético e incrementa los síntomas y el desarrollo de asma[8][46], ver figura 4.

El NO_2 es emitido por la quema de combustibles fósiles como gas, gasolina y petróleo [42] y el uso de fertilizantes. Los efectos de la exposición a altas concentraciones incluyen irritación de los ojos, garganta y pulmones. Su exposición prolongada puede producir asma y bronquiolitis obliterante [46]. En ambientes húmedos y de altas concentraciones, puede producir ácido nítrico.

El monitoreo adecuado de la contaminación atmosférica es necesario para evaluar la aplicación de normas de reducción y estrategias de reducción. Actualmente este monitoreo se realiza a través de una red de estaciones de calidad del aire a veces suplementado por modelación [10][32]. La normativa nacional e internacional requiere obtener resultados con bajas incertezas y alta confiabilidad lo que incrementa los costos, resultando en un bajo número de estaciones, alejadas entre sí y enfocadas a medir la contaminación de fondo[30]. Los mapas generados por estas estaciones a menudo poseen baja resolución espacial, lo que no es adecuado para identificar máximos estacionales y no considera la variabilidad espacio temporal de la contaminación en escala urbana[20][26].

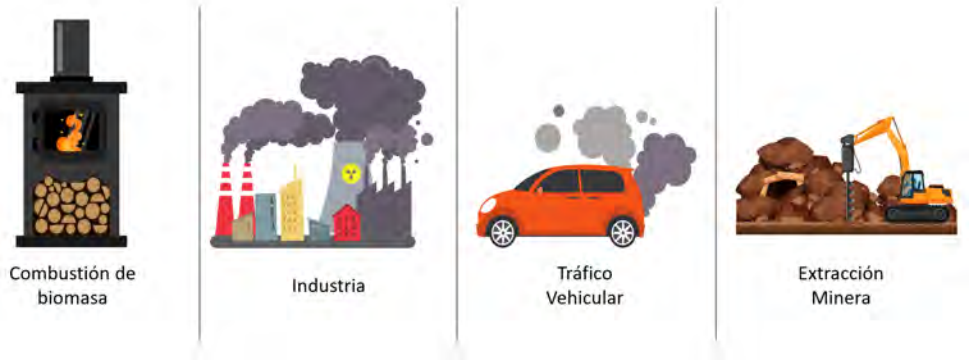


Figura 3: Principales fuentes de MP en la Región Metropolitana.
Fuente de la imagen: Roi and Roi/Miki022/BlueRingMedia/Shutterstock.com.

Actualmente se encuentran disponibles sensores de bajo costo que potencialmente pueden suplir estas deficiencias. Aunque estos sensores no poseen la misma resolución, sensibilidad, estabilidad, exactitud y precisión que las redes de monitoreo tradicionales [10], ellos pueden representar un suplemento en los programas de monitoreo y reemplazar parcialmente algunas estaciones. Empleando sensores de bajo costo se puede mejorar la resolución espacial y temporal de los mapas de contaminación[10][13][33], mejorar el conocimiento de la dinámica, identificar máximos estacionales y validar o reconsiderar la ubicación de algunas estaciones de monitoreo. Además, estos sensores pueden ser utilizados como una herramienta de investigación para medir la exposición personal[26], como sistemas de alerta de eventos de emergencia[21], y para medir y controlar las emisiones de las fuentes directas.

Estos sensores individuales necesitan ser integrados en un sistema de medición (nodo sensor). Este nodo sensor posee un microcontrolador que se encarga de la gestión de datos, los sensores y otros componentes opcionales, como medios de almacenamiento, visualización, transmisión de datos, reloj, entre otros, además de toda la electrónica requerida. El precio de los sensores individuales varía entre 15.000 y 400.000 pesos chilenos (CLP). El precio de un nodo sensor comercial puede alcanzar los 4.000.000 CLP.[10]

En este trabajo se propone la construcción y evaluación de un nodo sensor compuesto de sensores de bajo costo de MP, O_3 y NO_2 con la finalidad de estudiar la utilidad de estos sensores en la evaluación de la exposición personal y en el estudio de algunas variables y situaciones relevantes.

Figura 4: Fuentes de NO_2 y O_3 y sus impactos en la salud.
Adaptada de https://www.anmfvic.asn.au/~/media/files/anmf/submissions/submission-nepcaaq-july2019.pdf (Última visita: julio de 2020).

Exposición personal

El contacto de un agente externo con un receptor (por ejemplo, organismo humano) se denomina exposición[41]. La evaluación de la exposición es el proceso de caracterizar, estimar, medir y modelar la magnitud, frecuencia y duración del contacto con este agente[12].

El agente puede estar contenido en un producto (por ejemplo, en un alimento) o ser transportado a través de algún medio, como aire, agua o suelo. La superficie o punto de contacto es la región del receptor donde el agente se encuentra presente, pudiendo ser el exterior visible de la persona (por ejemplo, piel, boca, ojos o fosas nasales) o donde ocurre el proceso de absorción (por ejemplo, los pulmones o el tracto intestinal). La ruta de exposición es la forma de entrada del agente al receptor (inhalación o ingestión entre otros). La duración de la exposición es el tiempo de contacto entre el agente y el receptor medido de forma continua. La dosis absorbida es la medida directa o indirecta de la cantidad de agente que ingresa al receptor.

Un receptor puede exponerse en mayor o menor medida a un agente dependiendo de su patrón de actividad. La exposición humana puede ser expresada para un individuo, un grupo de individuos que comparten alguna característica común (por ejemplo, que practican

un determinado deporte o que residen en un determinado lugar) o para poblaciones enteras.

La ruta más importante de exposición humana a contaminantes atmosféricos es la inhalación. Generalmente, se supone que la exposición por inhalación de gases, aerosoles y partículas, es igual a la dosis absorbida, debido a que estas sustancias pueden alcanzar los alvéolos pulmonares y subsecuentemente el torrente sanguíneo.

Los patrones de actividad de un individuo consideran tanto la exposición en ambientes exteriores (al aire libre) como en ambientes interiores (hogar, transporte, oficina, etc.). Una actividad realizada en un determinado ambiente se conoce como microambiente (ver figura 5).

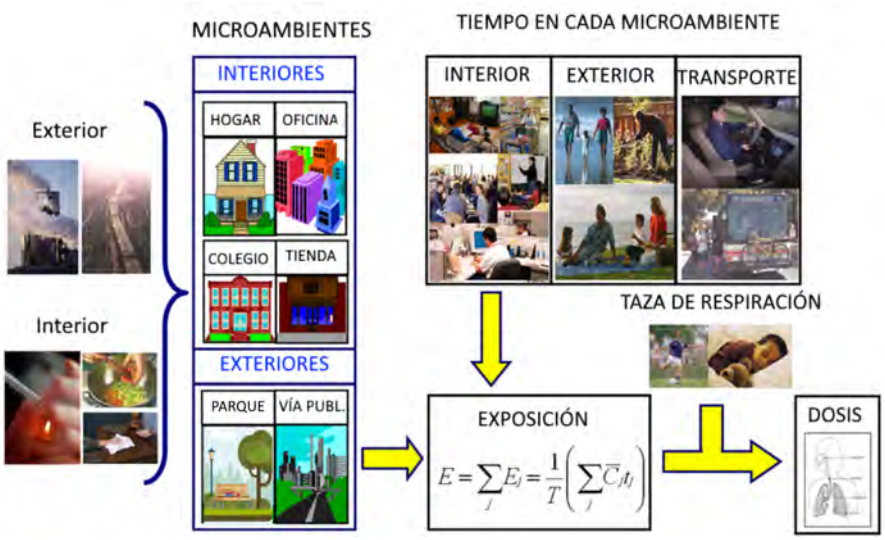


Figura 5: Microambientes, exposición y dosis

Referencia de la imagen: Halûk Özkaynak US EPA, Office of Research and Development National Exposure Research Laboratory, RTP, NC. Presented at the CMAS Special Symposium on Air Quality October 13, 2010.

Para calcular la dosis por inhalación a un contaminante atmosférico en un área geográfica que incluye personas de distintos grupos de edades, se puede utilizar la ecuación 1.[41][43]

$$D_j = \sum_{u=1}^{n} \int IR_j(u)\, P(u)\, dt \qquad (1)$$

donde, D_j es la dosis del grupo j en μg; u es el tipo de microambiente; $P(u)$ es la concentración de contaminante en el microambiente u en unidades de $\mu g/m^3$; $IR_j(u)$ es

la tasa de respiración del grupo j en el microambiente u en $\left(m^3/h\right)$ y t es el tiempo de exposición (h). Si no se considerá la tasa de respiración, se puede calcular la exposición en un periodo de tiempo como:

$$E_j^* = \sum_{u=1}^{n} \int \quad P(u)\, dt \tag{2}$$

en unidades de concentración por tiempo ($\mu g\ h/m^3$). La exposición promedio en un rango de tiempo se puede calcular como:

$$E_j = \frac{1}{T} \sum_{u=1}^{n} \int \quad P(u)\, dt \tag{3}$$

en unidades de concentración $\mu g/m^3$[41], T es el tiempo total de duración de la exposición. Esta ecuación es útil para comparar los niveles de exposición en un periodo de tiempo.

Con esta información es posible calcular la exposición personal en un rango de tiempo y la contribución de cada microambiente relevante para una persona o población. Considerando las tasas de respiración, es posible calcular la dosis inhalada.

Difusión de contaminantes

En un principio, los objetivos de este trabajo consideraban el desarrollo de modelos simples de reacción-difusión para determinar la dispersión de contaminantes producidos por el tránsito vehicular en geometrías simples. Comparar estos resultados con mediciones directas y determinar el impacto de la distancia entre la fuente y el receptor. Se ha realizado una investigación en la literatura científica, encontrando que la difusión es un mecanismo de transporte completamente irrelevante en esta escala y es necesario considerar modelos complejos para describir la distribución de contaminantes como los descritos por [3][4].

El transporte por difusión de contaminantes no es relevante en la escala dada por el tamaño promedio de las calles e incluso de las veredas, la mayor parte de los estudios sobre el transporte en cañón urbano (calles entre edificios) se concentran en el transporte por advección y convección[45][44]. Estos términos, a menudo intercambiables son los términos predominantes. La advección es el transporte hidrodinámico generado por flujos externos, como los vientos generados en la atmósfera superior, mientras que la convección es el transporte hidrodinámico generado por flujos internos como los generados por gradientes térmicos.

Objetivos

Objetivo General

Construcción de un nodo sensor portátil de bajo costo para medir gradientes y niveles de contaminación de MP , NO_2 y O_3 en microambientes urbanos. Utilizar este nodo sensor para estimar la exposición personal y generar recomendaciones para evitar exposiciones altas.

Objetivos Específicos

- Diseño y construcción de un nodo sensor portátil de bajo costo para medir MP, NO_2 y O_3. Ajustar su medición con las mediciones de la estación certificada de calidad del aire. Evaluar su rendimiento, considerando linealidad, exactitud, precisión, tiempo de respuesta, límite de detección, rango de funcionamiento, impacto de los cambios en temperatura, humedad e interferencias.
- Evaluar la exposición personal a MP, NO_2 y O_3 considerando un conjunto de micro-ambientes urbanos representativos. Encontrar hotspot donde las exposiciones son particularmente altas. Realizar una integración de la exposición personal en estos microambientes.
- Generar estrategias y recomendaciones para evitar exposiciones altas y proteger la salud de las personas que residen en una ciudad contaminada.

Hipótesis

Los nodos detectores funcionan como dosímetros individuales, para medir fuentes, mejorar los mapas de contaminación y pronósticos de calidad del aire. Estos nodos permiten medir gradientes espaciales y temporales de contaminación en tiempo real, estimar la contribución de distintos microambientes a la exposición personal, generar mapas de contaminación, y en conjunto con las tasas de respiración, conocer la dosis absorbida por contaminante.

CAPÍTULO 1:

SENSORES DE BAJO COSTO

Los sensores seleccionados para medir MP utilizan el principio de dispersión óptica infrarroja. Los sensores para medir O_3 y NO_2 son sensores electroquímicos que utilizan el principio de reacciones de oxido-reducción. La selección de estos sensores se ha realizado a través de una investigación en las bases de datos científicas: Scopus y Google Scholar. Algunas referencias se pueden encontrar en [6][9][10][22][20][26].

1.1 Metodología de medición

1.1.1 Sensores de MP

Estos sensores utilizan el principio de dispersión óptica infrarroja, determinando las concentraciones de MP a través de la medición de la intensidad de la luz dispersada por estas partículas. Se han extendido 2 tipos de sensores, tipo nefelómetro y tipo contador de partículas (*Optical Paticle Number OPC*). En un nefelómetro, la concentración de partículas es determinada por la intensidad de la luz dispersada, medido por un fotodetector. En un sensor tipo OPC, la luz dispersada por una sola partícula genera un pulso en el fotodetector. El total de pulsos y su intensidad es proporcional al número de partículas y a su diámetro aerodinámico respectivamente.

Se considera que los sensores tipo OPC son capaces de medir solamente partículas de tamaño mayor a $0,3\mu m$, debido a que el pulso generado por partículas pequeñas está en el rango del error del instrumento. En los sensores de tipo nefelómetro, si el número de partículas pequeñas es alto, éstas si pueden generar una respuesta que puede

ser detectada [23]. En la figura 6 se muestra el diagrama de un sensor tipo OPC Honeywell HPMA115S0-XXX.

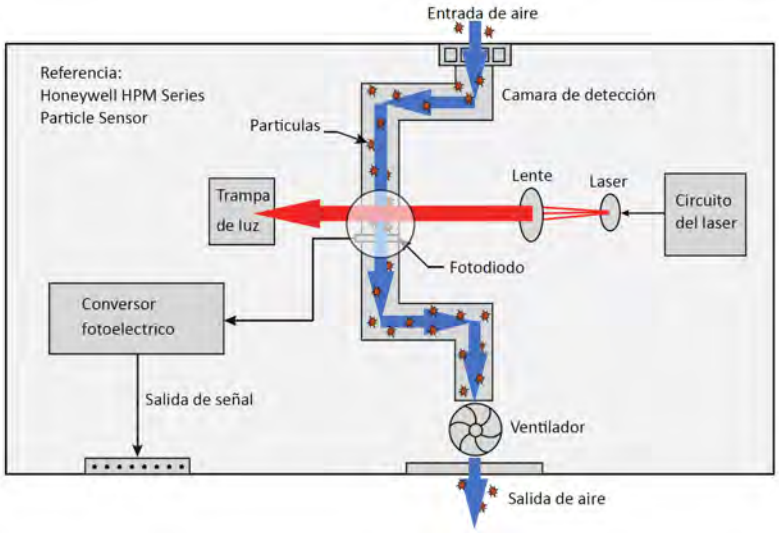


Figura 6: Componentes del sensor Honeywell HPMA115S0-XXX que utiliza el principio de dispersión óptica infrarroja.

Adaptada de `https://sensing.honeywell.com/sensors/particulate-matter-sensors/HPM-series` (Última visita: noviembre de 2019).

Los sensores de MP utilizan el principio de dispersión óptica infrarroja, donde una fuente de luz monocromática y compuesta de ondas planas incide sobre las partículas (ver figura 7). Se considera que las partículas son esféricas e isotrópicas. Además, se considera la dispersión de la luz como proveniente de la fuente principal y no de otra partícula (solamente la primera dispersión).

Desarrollando las ecuaciones de Maxwell en coordenadas esféricas para un frente de ondas plano de luz monocromática que incide sobre una esfera isotrópica, se encuentra que la intensidad de dispersión $I(\theta)$ está dada por:

$$I(\theta) = I_0 \frac{\lambda^2}{8\pi^2 r^2} \left(|S_1(\theta)|^2 + |S_2(\theta)|^2 \right) \tag{4}$$

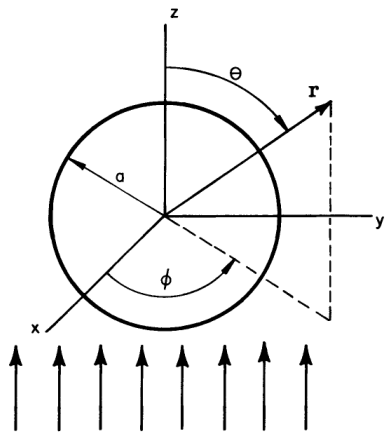


Figura 7: Frente de ondas plano aproximándose a una esfera

con:

$$S_1(\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{2(n+1)} \left(a_n \pi_n(\cos\theta) + b_n \tau_n(\cos\theta)\right) \tag{5}$$

$$S_2(\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{2(n+1)} \left(a_n \tau_n(\cos\theta) + b_n \pi_n(\cos\theta)\right) \tag{6}$$

donde $\pi_n(\cos\theta)$ y $\tau_n(\cos\theta)$ son los polinomios asociados de Legendre y su derivada. a_n y b_n son los coeficientes de la matriz de scattering.

Cuando el tamaño de la esfera se vuelve muy pequeño comparado con la longitud de onda incidente ($a \ll \lambda$), solo el primer termino de los coeficientes de scattering contribuyen. Además $b_n = 0$

$$a_1 \approx -i\frac{2}{3}\left(\frac{2\pi n_2 a}{\lambda}\right)^3 \frac{(n_1/n_2)^2 - 1}{(n_1/n_2)^2 + 2} \tag{7}$$

$\tau_1(\cos\theta) = 1$ y $\pi_1(\cos\theta) = \cos\theta$, entonces:

$$S_1(\theta) = -i\frac{(n_1/n_2)^2 - 1}{(n_1/n_2)^2 + 2}\left(\frac{2\pi n_2 a}{\lambda}\right)^3 \cos\theta \tag{8}$$

$$S_2(\theta) = -i\frac{(n_1/n_2)^2 - 1}{(n_1/n_2)^2 + 2}\left(\frac{2\pi n_2 a}{\lambda}\right)^3 \tag{9}$$

$$I(\theta) = I_0\left(\frac{1+\cos^2\theta}{2r^2}\right)\left(\frac{2\pi}{\lambda}\right)^4 \left(\frac{(n_1/n_2)^2 - 1}{(n_1/n_2)^2 + 2}\right)^2 (a)^6 \tag{10}$$

1.1.2 Sensores de gas

Los sensores utilizados, son sensores electroquímicos que utilizan la metodología de oxido-reducción para su funcionamiento, estos sensores reaccionan con el gas objetivo generando una señal eléctrica proporcional a la concentración del gas. El sensor posee 4 electrodos separados por filtros hidrofílicos y en contacto con una delgada capa de líquido electrolito (ver figura 8). Cuando el gas entra en contacto con el sensor, se difunde a través de una membrana hidrofóbica, luego de difundir, el gas alcanza la superficie de detección, los electrodos oxidan o reducen y balancean la corriente generada. Estas reacciones son catalizadas por los compuestos presentes en el electrodo y están desarrolladas precisamente para el gas objetivo[5][24].

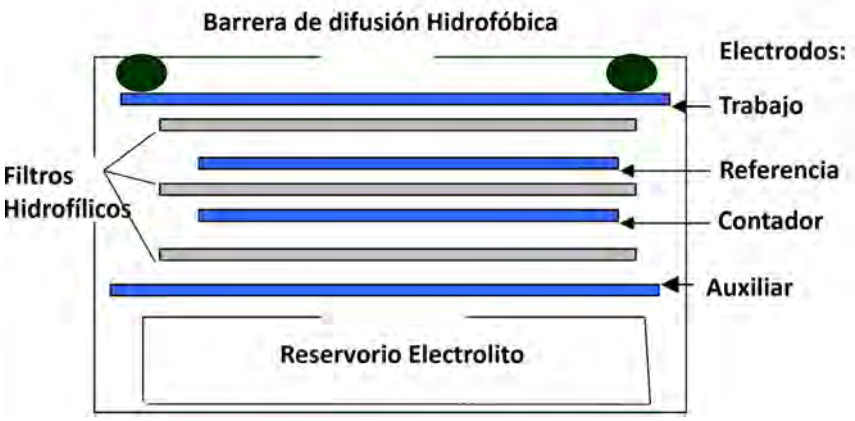


Figura 8: Celdas de los sensores electroquímicos utilizados

Referencia: Dr John Saffell, Electrochemical Gas Cells in Air Quality Networks- Solving the Problems, dic. 2013.

Las funciones de los electrodos de estos sensores son:

- Electrodo de trabajo: es donde ocurre el proceso de oxidación o reducción, creado una corriente que es proporcional a la concentración del gas, esta corriente es suministrada por el electrodo contador.
- Electrodo de Referencia: Es utilizado por el circuito potenciostático para mantener fijo el potencial del electrodo de trabajo. Su función es mantener este electrodo en el rango de funcionamiento con sensibilidad constante.

  El electrodo de trabajo debe ser mantenido al mismo potencial que el electrodo de referencia, o con un offset dependiendo del sensor.

- Electrodo Contador: balancea la reacción del electrodo de trabajo, reduciendo algunos compuestos químicos (normalmente oxigeno) si el electrodo de trabajo se encuentra oxidando u oxida si el electrodo de trabajo está reduciendo. El voltaje del electrodo contador puede variar dependiendo de las condiciones de medición y de la concentración del gas objetivo. Su magnitud no es importante, siempre y cuando el circuito potenciómetro pueda suministrar suficiente voltaje y corriente para mantener el electrodo de trabajo al mismo potencial que el electrodo de referencia.
- Electrodo Auxiliar: su potencial se ajusta para balancear la reacción que ocurre en el electrodo de trabajo. Su medición permite conocer el potencial del electrodo de trabajo sin medir directamente en él y así no comprometer la estabilidad del electrodo de referencia.

La reducción de NO_2:

$$\text{Electrodo de Trabajo: } NO_2 + 2H^+ + 2e^- \rightleftharpoons NO + H_2O$$

$$\text{Electrodo Contador: } H_2O \rightleftharpoons \frac{1}{2}O_2 + 2H^+ + 2e^-$$

$$\text{Reacción: } NO_2 \rightarrow NO + \frac{1}{2}O_2$$

La reducción de O_3:

$$\text{Electrodo de Trabajo: } O_3 + 2H^+ + 2e^- \rightleftharpoons O_2 + H_2O$$

$$\text{Electrodo Contador: } H_2O \rightleftharpoons \frac{1}{2}O_2 + 2H^+ + 2e^-$$

$$\text{Reacción: } O_3 \rightarrow O_2 + \frac{1}{2}O_2$$

En estas reacciones, los protones (H^+) difunden mediante el electrolito entre electrodos, mientras la señal de los electrones pasa por circuito externo.

Otros gases pueden interferir en la respuesta, ya que estos sensores son sensibles a más de un gas (ver anexo 2). Algunos estudios científicos, calibran su respuesta con modelos de regresión lineal multivariable[1].

1.2 Fuentes de error

Múltiples publicaciones, estudian el rendimiento de los sensores de bajo costo en diversas situaciones. La revisión de estos estudios, indica que los datos medidos por los

sensores de MP se ajustan a los datos validados con coeficiente de correlación generalmente mayor a 0,6[19], mientras que los datos medidos por los sensores electroquímicos necesitan un modelo de ajuste que considere variables interferentes[16][27][37].

Para ajustar los datos de un sensor, se realiza calibración en terreno (condiciones reales) y en laboratorio (condiciones controladas). En el laboratorio se realiza calibración con parámetros ambientales y concentraciones controladas, estos parámetros ambientales incluyen temperatura, humedad relativa y tasa de ventilación.[23]

En esta sección se realiza una revisión de los resultados de algunos estudios que han testeado los sensores de bajo costo en terreno y laboratorio.

1.2.1 Material Particulado

Numerosos estudios enfatizan la utilidad de los sensores que funcionan con principios ópticos, junto a ello también se advierte que el riesgo de mal uso es alto, considerando que algunas variables ambientales como la humedad relativa puede interferir en los resultados. Las partículas que se encuentran en el ambiente poseen distintas distribuciones de tamaño que puede variar significativamente dependiendo de su fuente, ubicación y las condiciones ambientales. Estas partículas poseen distintas propiedades físicas, como su forma y su índice de refracción, lo que incide en la detección de los sensores. Diversos estudios han mostrado que para alcanzar altos niveles de precisión y exactitud es necesario realizar calibraciones in-situ.

Además, los tamaños detectables de partículas varían de sensor a sensor y pueden ser significativamente diferentes a los que se encuentran especificados en su documentación. Para hacer este análisis es necesario generar partículas de diferentes tamaños e índices de refracción y dispersarlas uniformemente, luego comparar la respuesta de los sensores con la respuesta de los instrumentos de referencia.

Los resultados a continuación, fueron obtenidos del estudio: *"Laboratory evaluation of particle size-selectivity of optical low-cost particulate matter sensors"*[22] que compara la respuesta de distintos modelos de sensores de bajo costo, cuando se exponen a estas partículas con la respuesta de un instrumento calibrado: Aerodynamic Particle Sizer 3321, TSI Inc., USA. Esta investigación fue hecha en el laboratorio usando un sistema novel de generación de aerosol capaz de producir constantemente partículas de 20 diferentes

diámetros aerodinamico mono-dispersadas. Se han considerado los siguientes modelos de sensores: Plantower PMS5003, Novasensor SDS011, Sensirion SPS030, entre otros.

Plantower PMS 5003

Como se aprecia en las figura 9, este sensor detecta MP_{10} y $MP_{2,5}$, la máxima eficiencia de la detección para ambos canales es de $80\,\%$ y se encuentra en torno a las partículas de diámetro aerodinámico $0,7\mu m$, por lo que no resulta efectivo para medir MP_{10}.

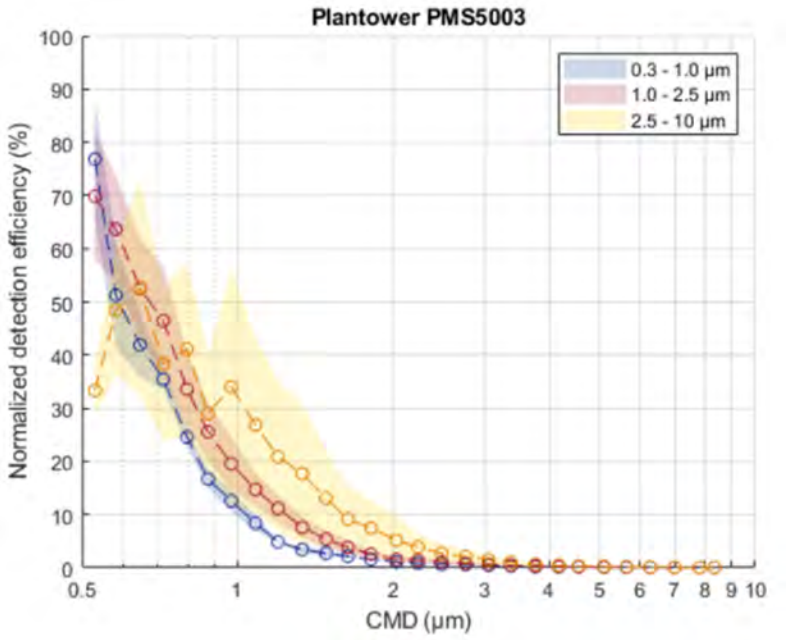


Figura 9: Plantower PMS 5003, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22]

Novasensor SDS011

Como se aprecia en las figura 10, este sensor detecta MP_{10} y $MP_{2,5}$, la máxima eficiencia de la detección se encuentra entre $80\,\%$ y $90\,\%$, la máxima eficiencia para $MP_{2,5}$ se encuentra para partículas de diámetro aerodinámico menor a $0,8\mu m$ y para MP_{10} entre $0,7$ y $1,7\mu m$. Por lo tanto tampoco es adecuado para medir MP_{10}.

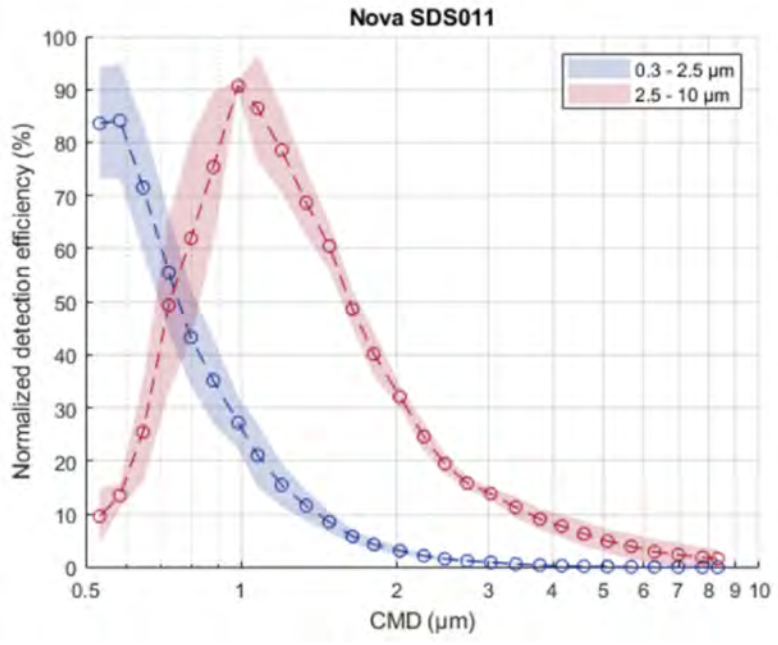


Figura 10: Novasensor SDS011, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22]

Sensirion SPS030

Como se aprecia en las figura 11, este sensor posee 4 canales de detección, $0,3$ a $1,0\mu m$, $1,0$ a $2,5\mu m$, $2,5$ a $4,0\mu m$ y $4,0$ a $10\mu m$. La eficiencia de detección para todos los canales se encuentra en el $90\,\%$ y centradas en torno a las partículas menores a $0,9\mu m$ para $MP_{2,5}$ y entre $0,7$ a $1,3\mu m$ para los demás canales. Por lo que este sensor resulta efectivo para medir $MP_{2,5}$ pero no para la fracción gruesa.

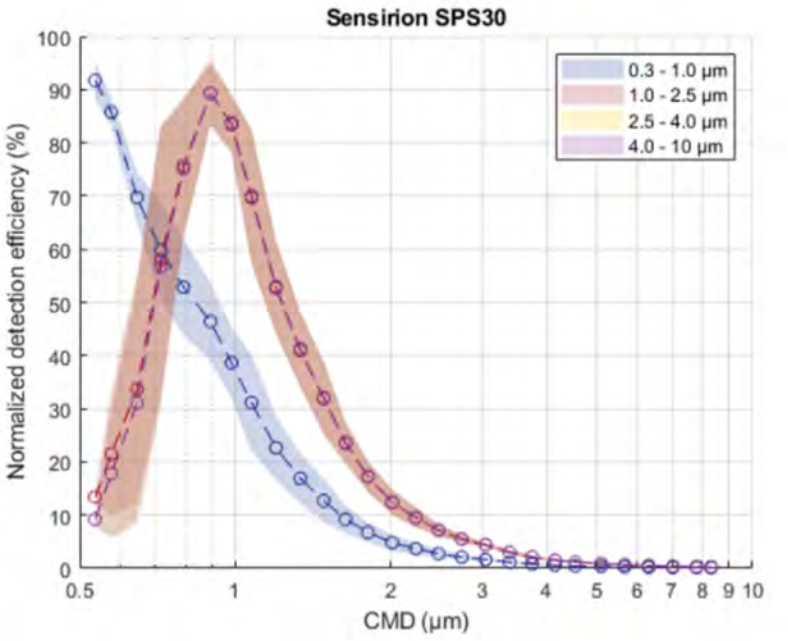


Figura 11: Sensirion SPS030, diámetro aerodinámico (CMD) (μm) versus eficiencia de medición normalizada (%)[22]

1.3 Sensores seleccionados

Para medir MP se han seleccionado dos modelos de sensores: Sensirion SPS030 y Novasensor SDS011. Estos sensores utilizan la metodología OPC y nefelómetro, respectivamente.

Para medir gases, se han seleccionado los sensores Alphasense NO2-A43F y OX-A431 que miden NO_2 y $NO_2 + O_3$, respectivamente. El modelo seleccionado integra dos sensores en un circuito análogo (*Analog Front End*) el que estabiliza el potencial en las celdas electroquímicas y permite medir su voltaje.

Los sensores seleccionados y sus características se mencionan en la tabla 1.

Tabla 1: Características de los sensores seleccionados

Componente	Modelo	Error	Tiempo de respuesta
MP_{10},	Sensirion SPS030	$\pm 10 \mu g/m^3$ $(0 - 100 \mu g/m^3)$	$1s$
		$\pm 10\,\%$ $(100 - 1000 \mu g/m^3)$	
$MP_{2,5}$	Novasensor SDS011	$\pm 10 \mu g/m^3$ $(0 - 100 \mu g/m^3)$	$0,5s$
		$\pm 10\,\%$ $(100 - 1000 \mu g/m^3)$	
O_3 + NO_2	Alphasense OX-A431	$\pm 0,5ppb$ $(0 - 20ppb)$	$t_{90} < 45s$ $(0 - 1ppb$ de $O_3)$
NO_2	Alphasense NO2-A43F	$\pm 0,5ppb$ $(0 - 20ppb)$	$t_{90} < 45s$ $(0 - 1ppb$ de $NO_2)$

CAPÍTULO 2:

DISEÑO Y CONSTRUCCIÓN

Las principales variables de interés son: MP_{10}, $MP_{2,5}$, NO_2 y O_3.

El nodo sensor debe ser portable, con capacidad de alimentación por baterías, de bajo costo y compatible con los sensores seleccionados. La medición debe ser en tiempo real (medición instantánea, tiempo de respuesta del orden de segundos).

Para estimar la exposición personal, es necesario conocer el microambiente en el que se está midiendo y la fecha y hora de la medición. Para este fin, el nodo sensor se ha equipado con un GPS y un reloj RTC, cuya función es medir los parámetros espaciales y temporales de la exposición personal y encontrar las altas concentraciones.

Considerando que las metodologías utilizadas para medir $MP_{2,5}$ (dispersión óptica infrarroja) y gases NO_2 y O_3 (sensores electroquímicos, método de reacciones Redox) son susceptibles a los cambios en las variables meteorológicas, el nodo sensor también se ha equipado con un sensor Bosch BME280 que permite medir en tiempo real la temperatura, presión atmosférica y humedad relativa.

Para guardar los datos de forma local se ha incluido una memoria microSD. Para su transmisión se ha utilizado WiFi y red de celular 2G.

La metodología de diseño se puede observar en la figura 12.

Para reducir el uso de recursos y realizar mediciones en paralelo, se ha optado por la construcción de dos modelos de nodo sensor; una versión completa para medir en forma estacionaria y en movimiento y una versión reducida para medir solamente $MP_{2,5}$ en forma estacionaria. Sus características y tiempos de respuesta se muestran en la tabla 2 y 3 respectivamente.

Figura 12: Componentes del diseño del nodo sensor

Tabla 2: Versiones de nodos diseñadas

Nodo	Tipo	Parámetro	Sensor	Especificación (Error)
Versión 1	Material Particulado	MP_{10}	Nova sensor SDS011	± 10 µg/m³ (0 - 100 µg/m³)
		$MP_{2.5}$		± 10 % (100 - 1000 µg/m³)
	Meteorología	Temperatura	Bosch BME280, Modulo de Adafruit	± 1° C (de 0 - 65 ° C)
		Presión		± 3% (20 - 80% de RH)
		Humedad		± 1,0 hPa (300 - 1100 hPa)
Versión 2	Material Particulado	MP_{10}	Sensirion SPS030	± 10 µg/m³ (0 - 100 µg/m³)
		$MP_{2.5}$		± 10 % (100 - 1000 µg/m³)
	Meteorología	Temperatura	Bosch BME280, Modulo de Adafruit	± 1° C (de 0 - 65 ° C)
		Presión		± 3% (20 - 80% de RH)
		Humedad		± 1,0 hPa (300 - 1100 hPa)
	Gases	NO_2	Alphasense OX-NO2	± 0,5 ppb (0 - 20 ppb)
		O_3		± 0,5 ppb (0 - 20 ppb)

2.1 Construcción

2.1.1 Versión completa (versión 2)

Los sensores de $MP_{2,5}$ seleccionados utilizan el puerto de comunicaciones UART, mientras que los sensores de NO_2 y O_3 requieren un conversor análogo digital para su funcionamiento.

El diseño del nodo sensor utiliza un modem 2G (GSM) modelo Adafruit FONA - Mini Cellular GSM Breakout, este modem incorpora todas las funcionalidades de un teléfono celular y es compatible con las bandas 850/900/1800/1900MHz disponibles en todo el mundo. Su comunicación es mediante puerto UART y su programación por comandos AT.

El modelo de GPS es el Adafruit Ultimate GPS FeatherWing, el que utiliza un chipset MTK3339, que puede rastrear hasta 22 satélites en 66 canales. Este GPS es altamente sensible (-165dB), incorpora una antena cerámica, su resolución máxima es 10 actualizaciones por segundo, y consume 20mA. Además, incorpora un reloj RTC que se

Tabla 3: Error y tiempo de respuesta de los sensores utilizados

Componente	Modelo	Error	Tiempo de respuesta
MP sensor	Sensirion SPS030	± 10 µg/m³ (0 - 100 µg/m³) ± 10 % (100 - 1000 µg/m³)	1 s
	Novasensor SDS011	± 10 µg/m³ (0 - 100 µg/m³) ± 10 % (100 - 1000 µg/m³)	0,5 s
Sensor O_3 + NO_2	Alphasense OX-A431	± 0,5 ppb (0 - 20 ppb)	t_{90} < 45 s (0 - 1 ppb de O_3)
Sensor NO_2		± 0,5 ppb (0 - 20 ppb)	t_{90} < 45 s (0 - 1 ppb de NO_2)
Ubicación (GPS)	MTK3339	2,5 m	10 Hz
Time (reloj)	PCF8523	6 s/yr	32 kHz
Temperatura	Bosch BME280	± 1° C (de 0 - 65 ° C)	13,51 Hz
Humidad		± 3% (20 - 80% de RH)	
Presión		± 1,0 hPa (300 - 1100 hPa)	

Referencia: Documentación respectiva (datasheet)

sincroniza con la hora satelital y una batería CR1220 que mantiene la hora cuando no se encuentra sincronizado.

La pantalla incorporada es el modelo Adafruit FeatherWing OLED - 128x32 que incorpora el driver SSD1306. Esta pantalla es monocromática (blanca) y puede mostrar hasta 4 líneas de 24 caracteres de información con tasa de refresco de 1 Hz, consume 10mA y se conecta mediante protocolo I2C.

El módulo de batería es compatible con el factor de forma 18650. Este módulo actúa como fuente de poder UPS, activándose automáticamente cuando se desconecta la fuente de alimentación. Además, posee controlador de carga y circuitos de protección.

Se ha diseñado una plataforma modular que consiste en una placa de desarrollo que incorpora un microcontrolador, 1 GPS, 1 pantalla, 1 modem 2G (GSM) y 1 conversor análogo digital de 16 bits, utilizando una tarjeta de desarrollo Adafruit Feather M0 Adalogger, que integra un microcontrolador Atmel ATSAMD21G18 a 48MHz, 3,3V, 256KB de memoria FLASH (ROM) y 32KB de RAM, un puerto de carga de 100mAh compatible con una batería de litio polímero (LiPo), USB nativo para programación, LEDS y puerto microSD. Se ha elegido esta placa de desarrollo debido a que incorpora múltiples puertos serial (UART), I2C y SPI programables. Además de una pantalla, una memoria microSD y un módulo celular SIM800H como se muestra en la Figura 13.

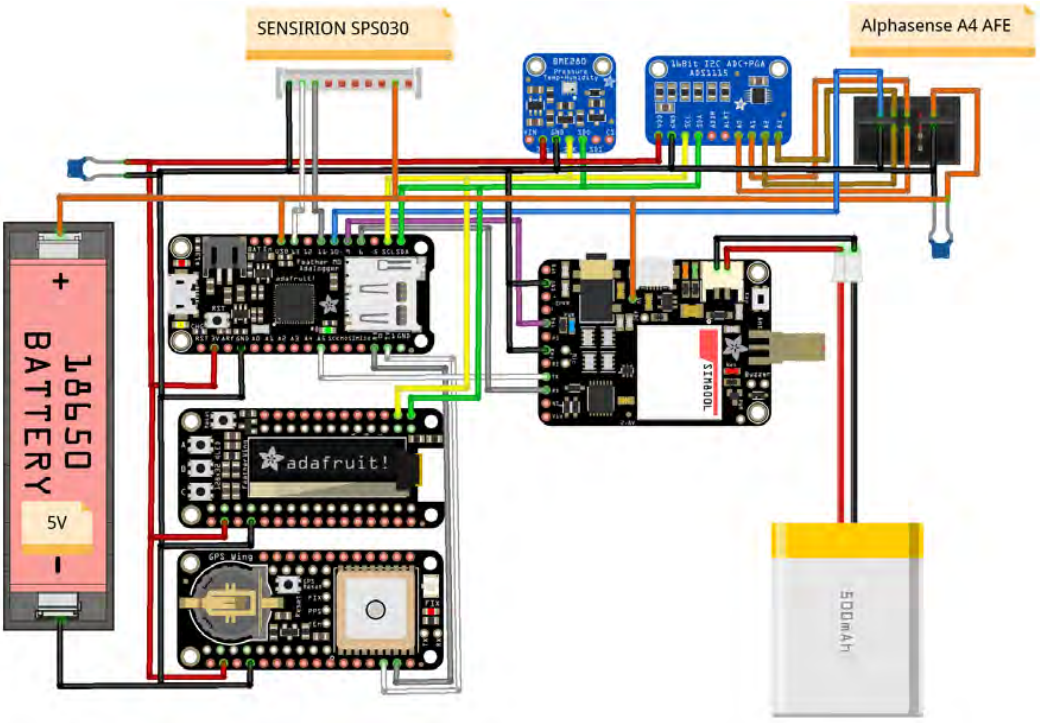


Figura 13: Diseño y conexiones del nodo con soporte de GPS, transmisión de datos mediante conectividad celular y batería.

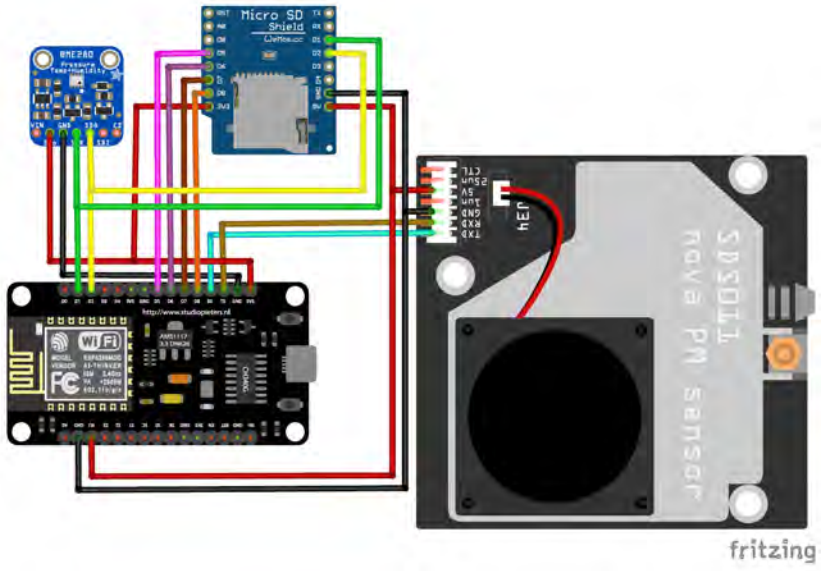


Figura 14: Diseño y conexiones del nodo con WiFi

2.1.2 Versión Reducida (versión 1)

Además, se ha construido una versión más simple que utiliza una tarjeta de desarrollo NodeMCU que integra un microcontrolador ESP8266 con capacidad de transmisión de datos vía wifi, además se ha utilizado un módulo microSD con reloj y el sensor de meteorología BME280 como se muestra en la Figura 14. Este diseño no incorpora los sensores de gas ni batería.

2.2 Nube digital

Para la descarga y visualización de datos en primera instancia se ha configurado un servidor FTP que recibe los datos y los almacena. Luego se configuró el envió de datos mediante MQTT, que es un protocolo más seguro y eficiente, estos datos se envían en tiempo real a la nube de Adafruit IO, que permite su visualización y almacenamiento. En esta nube se pueden configurar diversas interfases graficas (dashboard) que permiten observar las tendencias en tiempo real en una página web de acceso público (ver figura 15).

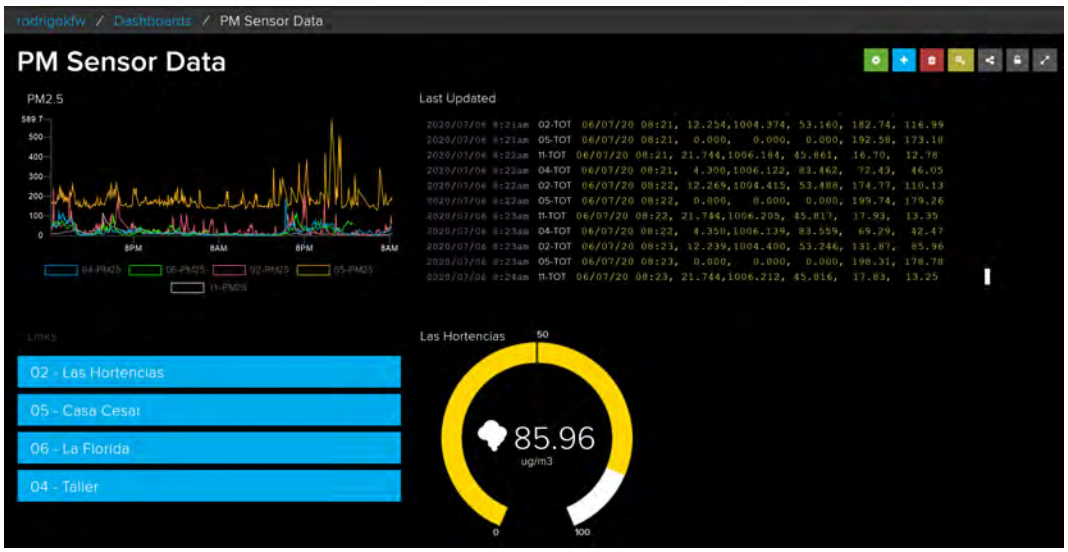


Figura 15: Interfase gráfica para visualización de datos.

Los nodos sensores armados y sus componentes instalados se muestran en las figuras 16 y 17.

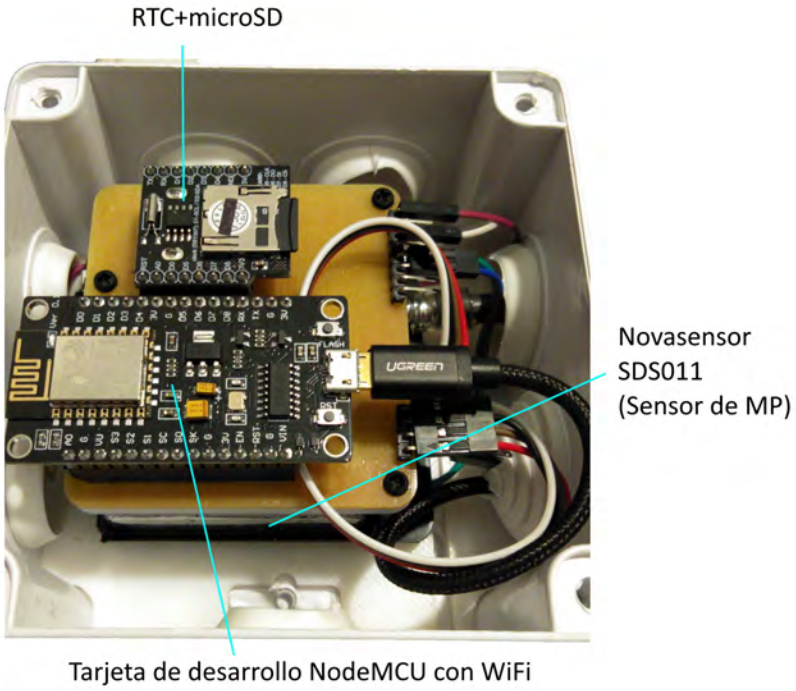


Figura 16: Nodo Sensor con WiFi, caja IP55, 85x85x50 mm

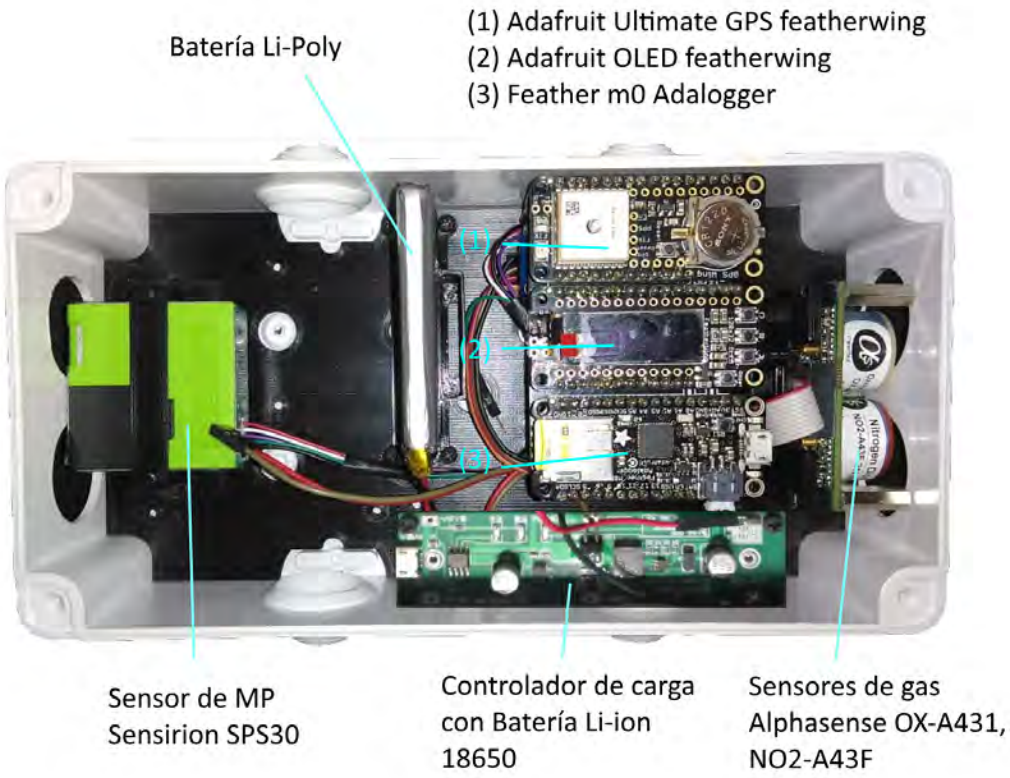


Figura 17: Nodo Sensor v2. Caja IP65 200x100x70 mm.

CAPÍTULO 3:

AJUSTE CON REFERENCIA

Figura 18: Instalación de nodos sensores en estación de referencia. Izquierda: La Florida (julio de 2019). Derecha: Las Condes (marzo de 2020)

Se realizaron 2 campañas de ajuste utilizando las estaciones del Sistema Nacional de Calidad de Aire (SINCA) del Ministerio del Medio Ambiente en la estación La Florida y Las Condes (ver figura 18).

- Primer ajuste: Estación La Florida entre el 03 y el 24 de julio de 2019.
- Segundo ajuste: Estación Las Condes entre el 11 y el 13 de marzo de 2020.

Los nodos sensores hacen 20 mediciones cada 1 minuto, luego guardan el promedio y su desviación standard. Las estaciones del SINCA entregan sus datos cada 1

hora, para hacer este ajuste, se promediaron los datos medidos cada hora por los nodos sensores.

Debido a las restricciones de movimiento impuestas por el covid-19, el segundo ajuste sólo se ha realizado durante 3 días. Desafortunadamente, los datos de la estación de referencia para MP_{10} y $MP_{2,5}$ no fueron validados por la autoridad ambiental, debido a que los equipos de referencia presentaron fallas en aproximadamente la mitad de tiempo. Para este ajuste, sólo se consideraron los datos validados.

3.1 Primer Ajuste

En el primer ajuste, se midió con la versión completa del nodo sensor. Este nodo estaba equipado con los sensores de gases, pero el ruido generado por el módulo 2G, afecto la electrónica asociada a estos sensores, provocando desestabilización en la salida análoga. Posterior a esta medición, se desconecto el modulo 2G del nodo sensor. Los resultados se muestran en la figura 19. La comparación entre la medición y la estación de referencia se muestra en la figura 20.

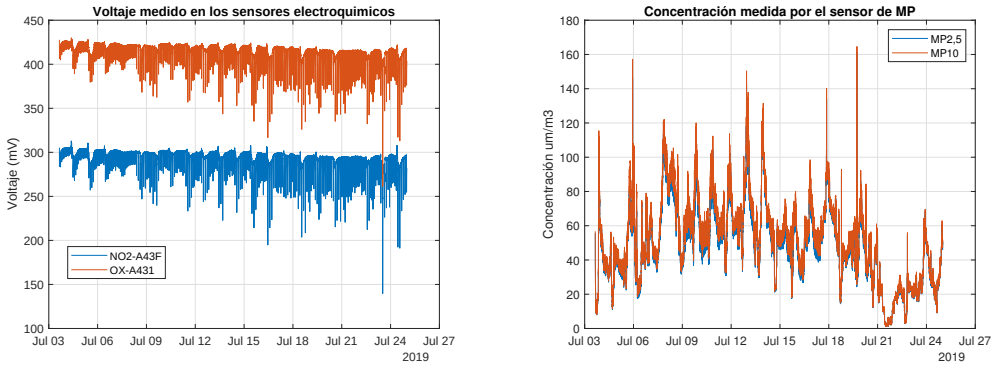


Figura 19: Primer ajuste, principales datos medidos.

En la figura 21 se muestra un gráfico de dispersión entre los valores de MP medido por el nodo sensor y la estación de referencia, diferenciando los casos cuando la Humedad relativa (H) es menor a $70\,\%$ y mayor o igual a $70\,\%$. No se aprecian diferencias significativas en la linealidad entre las variables cuando se compara los resultados de MP

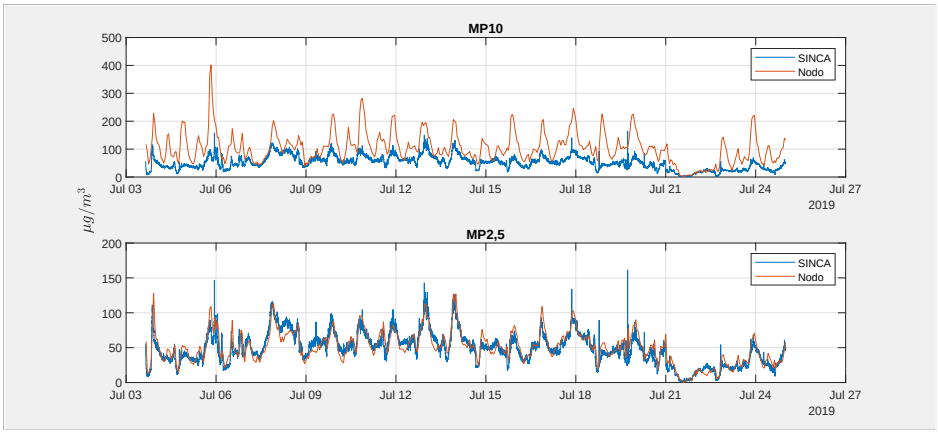


Figura 20: Comparación entre el MP medido y la estación de referencia.

considerando estos rangos de H. El coeficiente de correlación cuando H $< 70\,\%$ es 0,43 para MP_{10} y 0,85 para $MP_{2,5}$.

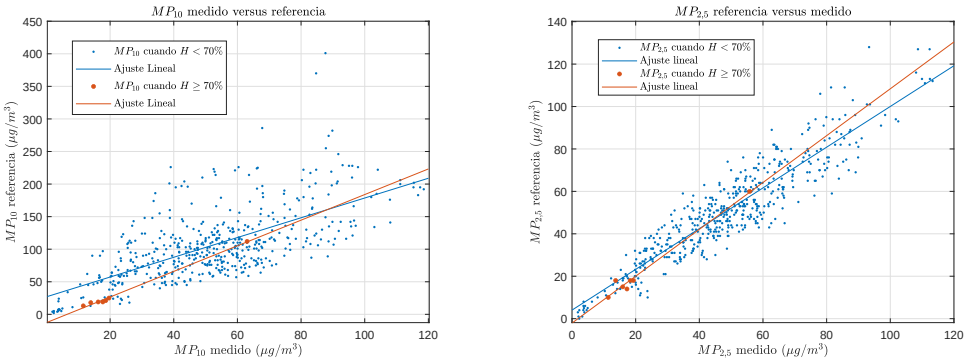


Figura 21: Correlación entre los valores de MP medidos por el nodo sensor y la estación de referencia

Considerando estos resultados, las ecuaciones de ajuste son las siguientes:

$$REF_{MP_{10}} = 1,540 MP_{10} + 25,31 \quad \text{RMSE} = 8,421 \quad r^2 = 0,4315 \tag{11}$$

$$REF_{MP_{2,5}} = 0,9628 MP_{2,5} + 3,861 \quad \text{RMSE} = 17,61 \quad r^2 = 0,8496 \tag{12}$$

Como se aprecia en la figura 21, los datos de MP_{10} y $MP_{2,5}$ muestran una correlación positiva. Los datos de $MP_{2,5}$ se ajustan con pendiente cercana a 1 e intercepto de 3,861

que indica un buen ajuste lineal entre las variables. Mientras que el ajuste de MP_{10} muestra una pendiente de 1,5 indicando subestimación y además un intercepto que indica que este sensor no se encuentre calibrado para medir en este rango.

3.2 Segundo Ajuste

Este ajuste se realizó utilizando 6 nodos sensores, 2 de ellos equipados con sensores de MP Sensirion SPS030 y sensores de gas Alphasense, y 4 equipados solamente con sensores de MP Novasensor SDS011. Los resultados se muestran en las figuras 22, 23 y 24. Los id 07, 08, 10 y 22 corresponden a sensores Novasensor SDS011 (nodo sensor v1), mientras que los id 98 y 99 corresponden a sensores Sensirion SPS30 (nodo sensor v2).

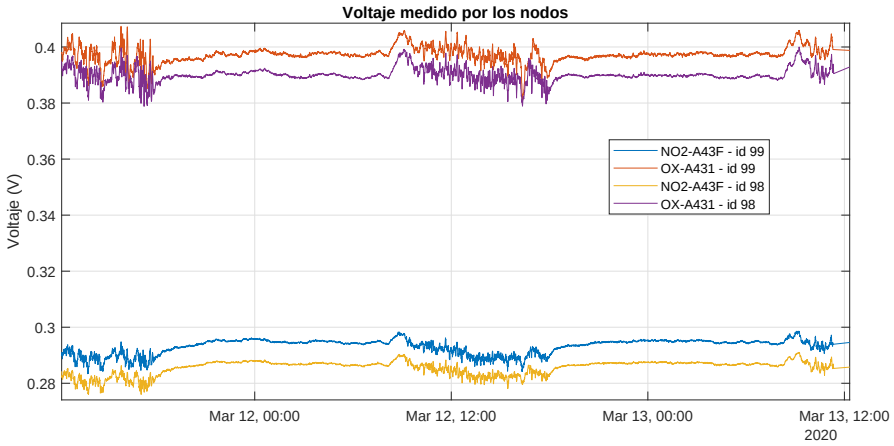


Figura 22: Datos medidos en los sensores de gas.

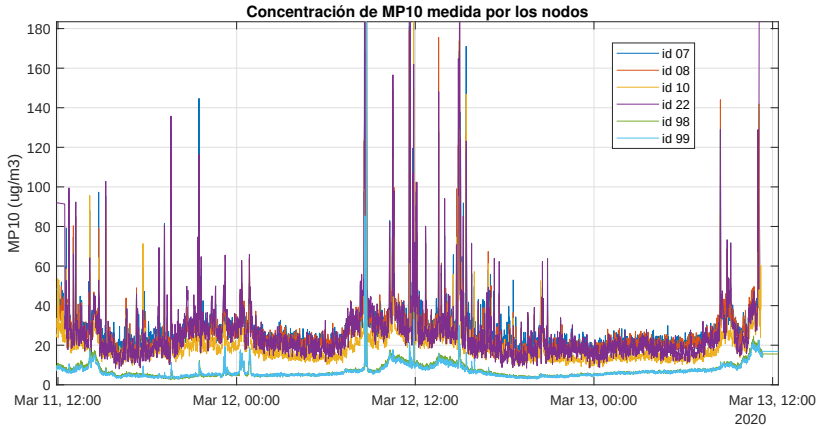


Figura 23: Datos de MP_{10} medidos.

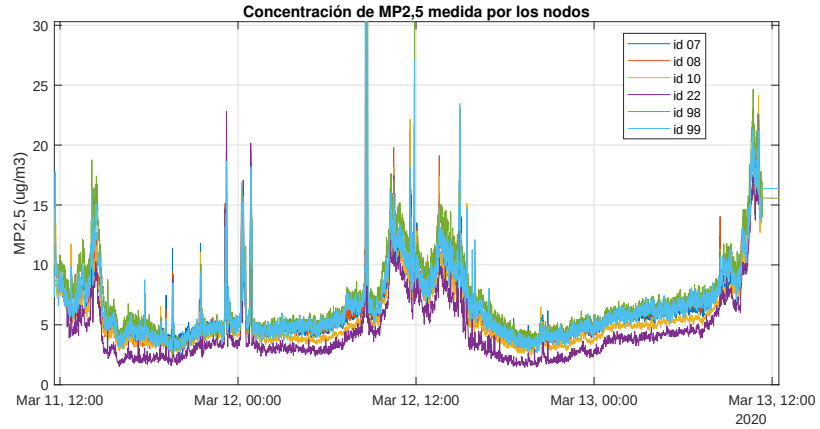


Figura 24: Datos de $MP_{2,5}$ medidos.

3.2.1 Material Particulado

Como se aprecia en las figuras 25 y 26, la estación de referencia muestra datos no validados aproximadamente la mitad del tiempo de medición, sólo se han considerado los datos validados para hacer el ajuste. Los nodos sensores poseen la misma tendencia que es diferente a la medida por la estación de referencia. No obstante los coeficientes de correlación (ver figuras 27 y 28) entre los nodos sensores y la estación de referencia se

encuentran entre 0,61 y 0,82. La correlación de $MP_{2,5}$ es mayor a la de MP_{10}. La diagonal de la matriz de correlación indica la distribución de la variable.

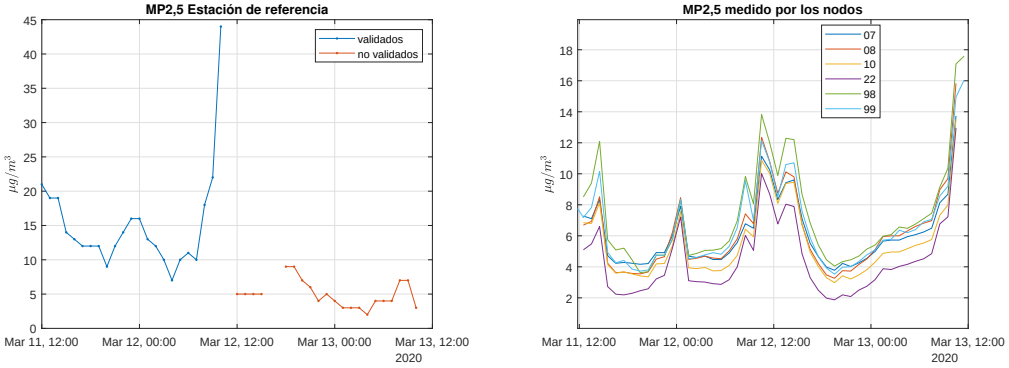


Figura 25: Comparación entre el $MP_{2,5}$ medido por la estación de referencia y los sensores nodos

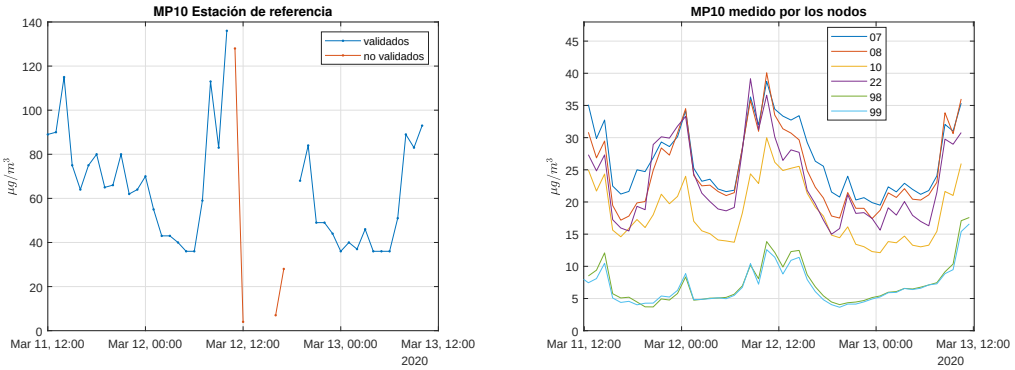


Figura 26: Comparación entre el MP_{10} medido por la estación de referencia y los sensores nodos

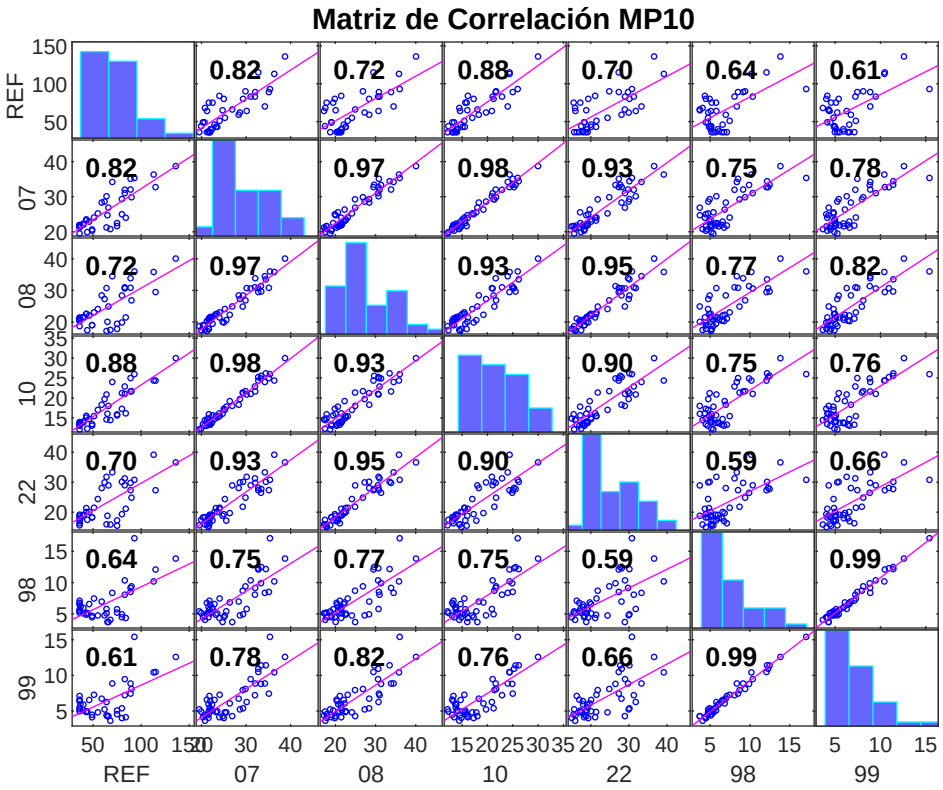


Figura 27: matriz de correlación de PM_{10}

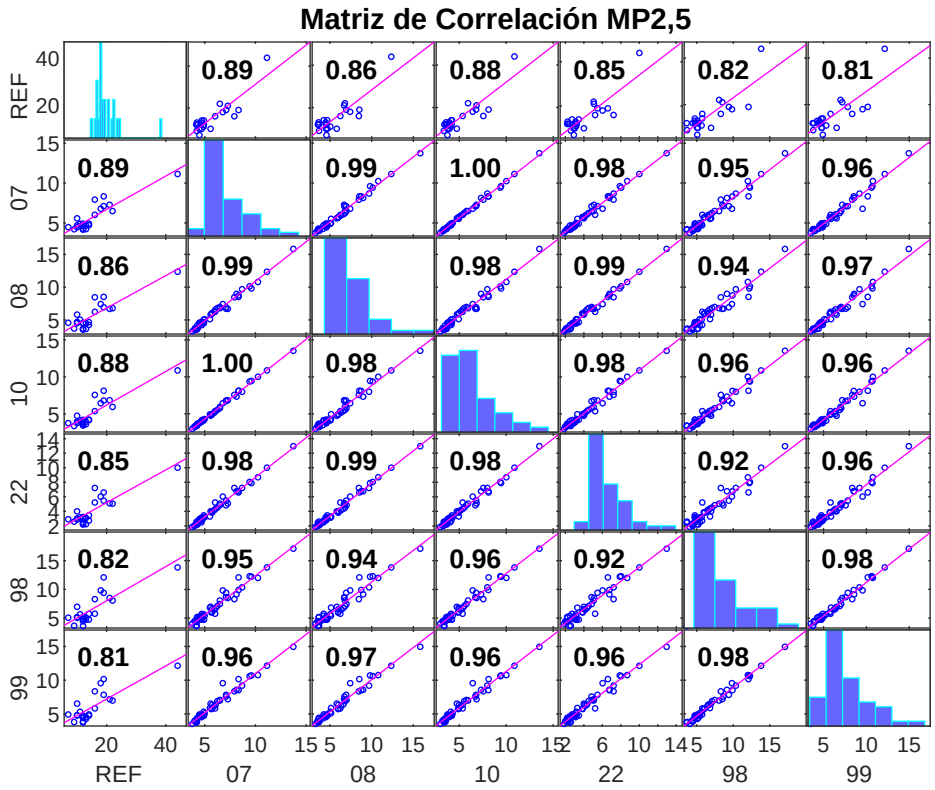


Figura 28: Matriz de correlación de $PM_{2,5}$

3.2.2 Sensores electroquímicos

La lectura de los sensores electroquímicos es sensible a diversos compuestos y factores ambientales como el ozono, la temperatura, y la humedad relativa [37][27][38]. Para estimar las concentraciones de NO_2 y O_3 usando la lectura de estos sensores, se ha utilizado la herramienta de MATLAB *Regression Learner*, considerando las variables temperatura (T), presión atmosférica (P), Humedad relativa (H), voltaje en el electrodo principal (V) y voltaje en el electrodo auxiliar (V_a). Se han entrenado los modelos: regresión lineal, aprendizaje basado en árboles de decisión (*Regression tree*), máquinas de vectores de soporte (*Support vector machine*), proceso de Gauss (*Gaussian process Regression model*) y arboles de predicción (*Ensemble of trees*), encontrando que el modelo que mejor

se ajusta a las concentraciones de NO_2 y O_3 es el ajuste lineal multivariable, lo que esta de acuerdo con los resultados encontrados en la literatura científica [27][7]. Este modelo es una generalización del ajuste lineal simple al caso de más de una variable independiente y restringido a una variable dependiente.

Como se ha indicado anteriormente, por razones de fuerza mayor el periodo de calibración ha sido muy corto y puede no cubrir un rango representativo de datos.

Dióxido de Nitrógeno

Se ha utilizado un ajuste lineal multivariable con términos de interacción. Los parámetros de ajuste se han designado como β_i donde i indica el coeficiente. V es el voltaje medido en el electrodo principal del sensor de NO_2. V_a es el voltaje medido en el electrodo auxiliar. T, R es la temperatura y humedad relativa medido por el sensor de meteorología.

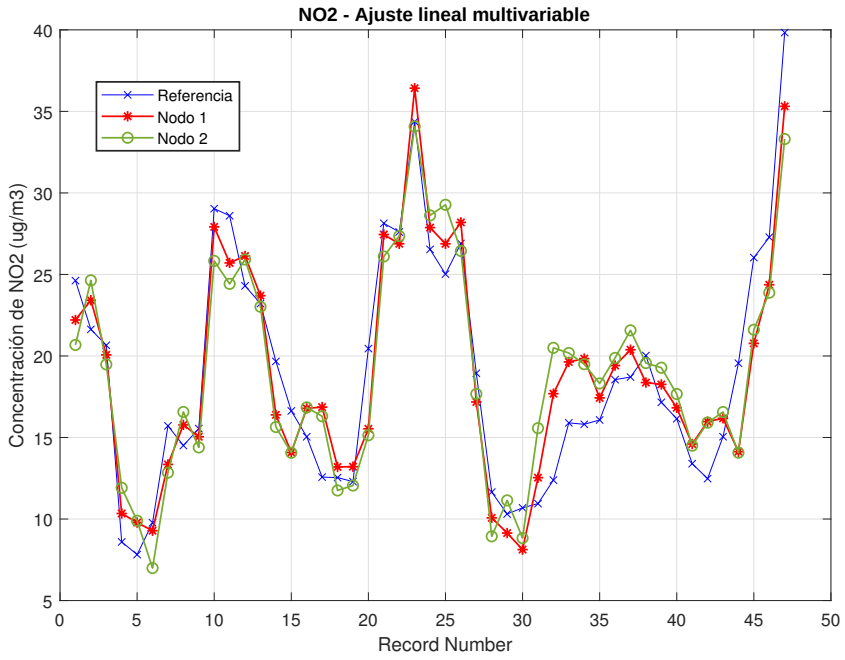


Figura 29: Ajuste multilinear de NO_2 con la estación de referencia

$$NO_2 = \beta_0 + \beta_1 V + \beta_2 T + \beta_3 H + \beta_4 V_a + \beta_5 (VT) + \beta_6 (VH) + \beta_7 (VV_a) + \beta_8 (TH) + \beta_9 (TV) + \beta_{10} (HV_a) \quad (13)$$

Los parámetros de ajuste se muestran en la tabla 4

Tabla 4: Parámetros de ajuste para NO_2

	Nodo Sensor 1				Nodo Sensor 2			
	Estimate	**SE**	**tStat**	**pValue**	**Estimate**	**SE**	**tStat**	**pValue**
β_0	4.56E+04	1.52E+04	2.99E+00	5.00E-03	1.73E+04	1.56E+04	1.11E+00	2.75E-01
β_1	-1.32E+05	5.33E+04	-2.47E+00	1.83E-02	-1.25E+05	4.98E+04	-2.52E+00	1.64E-02
β_2	2.02E+02	4.93E+01	4.11E+00	2.20E-04	2.40E+02	6.99E+01	3.44E+00	1.50E-03
β_3	1.12E+02	2.30E+01	4.86E+00	2.33E-05	1.06E+02	3.12E+01	3.39E+00	1.71E-03
β_4	-2.24E+05	5.27E+04	-4.24E+00	1.47E-04	-3.94E+04	5.82E+04	-6.77E-01	5.03E-01
β_5	-1.38E+03	3.00E+02	-4.59E+00	5.15E-05	5.17E+02	1.89E+02	2.73E+00	9.69E-03
β_6	-5.53E+02	1.33E+02	-4.16E+00	1.89E-04	1.63E+02	8.39E+01	1.94E+00	6.00E-02
β_7	6.80E+05	1.78E+05	3.82E+00	5.12E-04	3.66E+05	1.76E+05	2.08E+00	4.48E-02
β_8	-5.14E-02	1.16E-02	-4.43E+00	8.44E-05	-5.60E-02	1.58E-02	-3.55E+00	1.09E-03
β_9	7.15E+02	2.00E+02	3.58E+00	1.01E-03	-1.36E+03	4.05E+02	-3.37E+00	1.82E-03
β_{10}	1.80E+02	8.34E+01	2.16E+00	3.78E-02	-5.35E+02	1.77E+02	-3.02E+00	4.65E-03

Tabla 5: Raíz del error cuadrático medio (RMSE), coeficiente de correlación, Error cuadrático medio (MSE), Error absoluto medio (MAE) del ajuste de NO_2.

Nodo Sensor	RMSE	R^2	MSE	MAE
1	3,4486	0,77	11,893	2,9194
2	4.3773	0,63	19,161	3,4752

Ozono

Se ha utilizado un ajuste lineal multivariable con términos de interacción. Los parámetros de ajuste se han designado como β_i donde i indica el coeficiente. V es el voltaje medido, calculado como la diferencia entre el voltaje entregado por el electrodo principal del sensor de O_3 + NO_2 y el sensor de NO_2. V_a es la diferencia entre voltaje

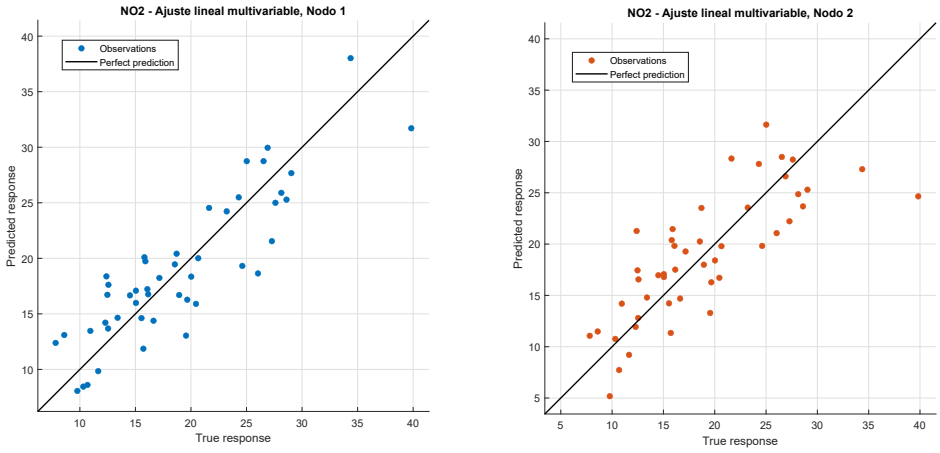


Figura 30: Correlación entre los datos ajustados de NO_2 y la estación de referencia

entregado por el electrodo auxiliar del sensor O_3 + NO_2 y el sensor de NO_2. T, R es la temperatura y humedad relativa medido por el sensor de meteorología. Para este ajuste, además se ha considerado el voltaje medido por el sensor de NO_2 designado por V_b.

$$\begin{aligned} O_3 =& \beta_0 + \beta_1 V + \beta_2 T + \beta_3 H + \beta_4 V_a + \beta_5 V_b + \beta_6 (VT) + \beta_7 (VH) \\ &+ \beta_8 (VV_a) + \beta_9 (VV_b) + \beta_{10} (TH) + \beta_{11} (TV_a) + \beta_{12} (TV_b) \\ &+ \beta_{13} (HV_a) + \beta_{14} (HV_b) + \beta_{15} (V_a V_b) \end{aligned} \tag{14}$$

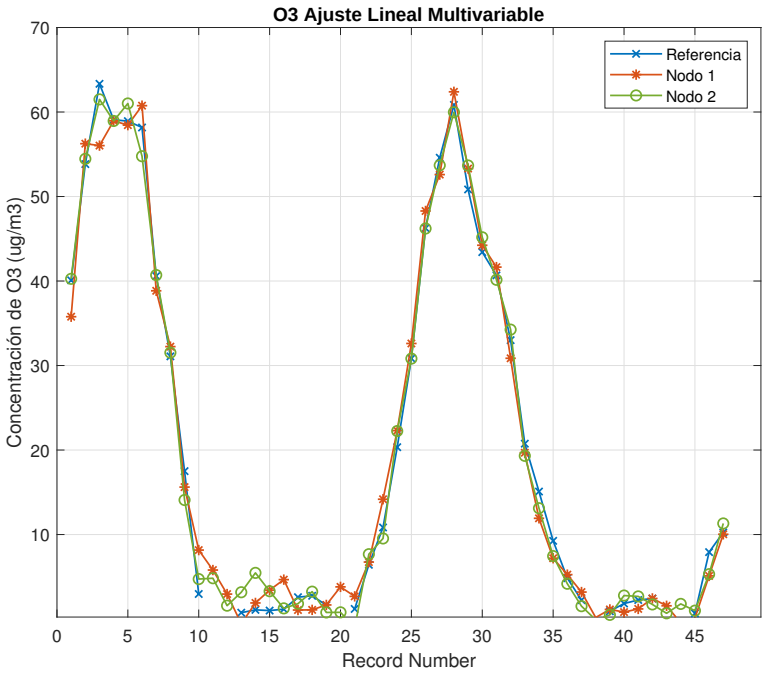


Figura 31: Ajuste multilinear de O_3 con la estación de referencia

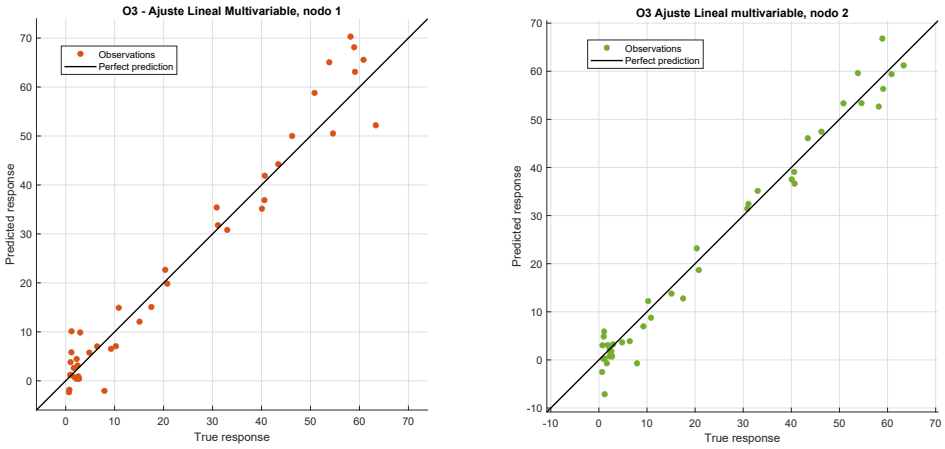


Figura 32: Dispersión lineal del ajuste de O_3 con la estación de referencia

Tabla 6: Parámetros de ajuste para O_3

	Nodo Sensor 1				Nodo Sensor 2			
	Estimate	**SE**	**tStat**	**pValue**	**Estimate**	**SE**	**tStat**	**pValue**
β_0	1.00E+04	2.93E+04	3.42E-01	7.35E-01	6.57E+03	6.16E+03	1.07E+00	2.97E-01
β_1	-4.54E+05	3.32E+05	-1.37E+00	1.84E-01	-7.69E+04	2.34E+04	-3.29E+00	2.98E-03
β_2	5.52E+02	2.68E+02	2.06E+00	5.01E-02	2.56E+02	6.47E+01	3.95E+00	5.59E-04
β_3	1.63E+02	9.79E+01	1.67E+00	1.08E-01	6.46E+01	2.65E+01	2.44E+00	2.21E-02
β_4	9.70E+04	3.77E+04	2.57E+00	1.65E-02	-9.99E+04	5.68E+04	-1.76E+00	9.10E-02
β_5	-2.23E+04	9.21E+04	-2.42E-01	8.11E-01	-1.27E+04	1.78E+04	-7.13E-01	4.82E-01
β_6	1.00E+03	3.64E+02	2.75E+00	1.09E-02	-2.73E+02	6.40E+01	-4.26E+00	2.51E-04
β_7	2.03E+02	1.33E+02	1.53E+00	1.38E-01	-9.39E+01	3.47E+01	-2.71E+00	1.20E-02
β_8	3.40E+05	2.21E+05	1.54E+00	1.36E-01	5.20E+05	1.25E+05	4.17E+00	3.22E-04
β_9	1.28E+06	1.03E+06	1.24E+00	2.27E-01	1.24E+05	4.61E+04	2.70E+00	1.24E-02
β_{10}	-2.56E-03	3.33E-02	-7.68E-02	9.39E-01	2.20E-02	2.42E-02	9.11E-01	3.71E-01
β_{11}	-9.23E+02	2.31E+02	-3.99E+00	5.03E-04	-5.21E+02	2.59E+02	-2.01E+00	5.56E-02
β_{12}	-1.83E+03	8.31E+02	-2.20E+00	3.74E-02	-5.81E+02	1.43E+02	-4.05E+00	4.32E-04
β_{13}	-2.50E+02	8.68E+01	-2.88E+00	7.95E-03	-7.29E+01	1.15E+02	-6.36E-01	5.30E-01
β_{14}	-5.18E+02	2.95E+02	-1.76E+00	9.15E-02	-1.59E+02	5.46E+01	-2.92E+00	7.39E-03
β_{15}	-3.27E+05	1.16E+05	-2.82E+00	9.26E-03	1.97E+05	1.70E+05	1.16E+00	2.57E-01

Tabla 7: Raíz del error cuadrático medio (RMSE), coeficiente de correlación, Error cuadrático medio (MSE), Error absoluto medio (MAE) del ajuste de O_3.

Nodo Sensor	**RMSE**	R^2	**MSE**	**MAE**
1	3,3462	0,98	11,1970	2,6385
2	5,0087	0,95	25,087	3,8191

CAPÍTULO 4:

EVALUACIÓN DE LA EXPOSICIÓN PERSONAL

Para evaluar la exposición personal se utilizó la versión completa del sensor, que se transportó en un bolso respirable y se mantuvo encendido midiendo los microambientes donde el usuario permaneció entre el 14 de febrero y el 10 de marzo de 2020. Su autonomía permitió medir en la vía pública, medios de transporte y en los microambientes sin suministro eléctrico. Se documentó fecha y hora de cada uno de los microambientes visitados.

Figura 33: Nodo sensor en el interior de bolso respirable.

Durante el periodo de medición los microambientes documentados fueron:

- Hogar: Microambiente interior, donde el usuario reside la mayor parte del tiempo. Este microambiente considera las concentraciones medidas dentro de una casa de un piso que se encuentra a 20 metros de una avenida residencial sin congestión vehicular, rodeada de casas y con un área verde con árboles de 20 metros entre la avenida y la entrada del hogar. No se encuentra cercana a otras fuentes de contaminación. Dos de los integrantes del hogar son fumadores de frecuencia 10 veces por día aproximado. Estos integrantes siempre fuman afuera del hogar.
- Bus: Microambiente móvil, considera los tiempos de viaje en bus realizado por el usuario, que son aproximadamente 20 minutos por día. Los buses empleados pertenecen a la red de transporte de Santiago, norma EURO VI, sin aire acondicionado y sin aislación exterior (ventanas abiertas). El bus suele ir a mediana capacidad.
- Metro: Microambiente móvil que considera el transporte en tren urbano de Santiago, línea 1, línea 5 y línea 4 durante 3 horas por día aproximado. En los tiempos de medición, el tren se mantuvo a un 80 % de su capacidad de pasajeros.
- Oficina: Microambiente fijo, que considera una oficina con aire acondicionado emplazada en el sexto piso de un edificio del año 2016, el espacio medio de separación entre las personas es de $8m^2$ aproximadamente.
- Laboratorio: Microambiente fijo que considera un laboratorio equipado con máquinas para cortar metales, estación de soldadura, maquinas CNC e impresora 3D y computadores, medianamente aislado del aire exterior.

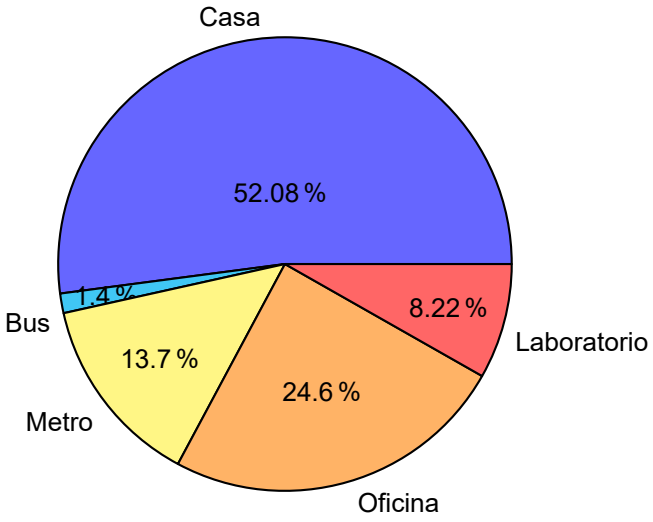


Figura 34: Tiempo promedio en cada microambiente.

4.1 Material Particulado

Utilizando los resultados del ajuste de MP_{10} y $MP_{2,5}$ (Ecuación 12), se ajustaron todos los datos medidos, la medición total incluyendo todos los microambientes y el periodo de ajuste con la referencia se muestra en las figuras 35 y 36.

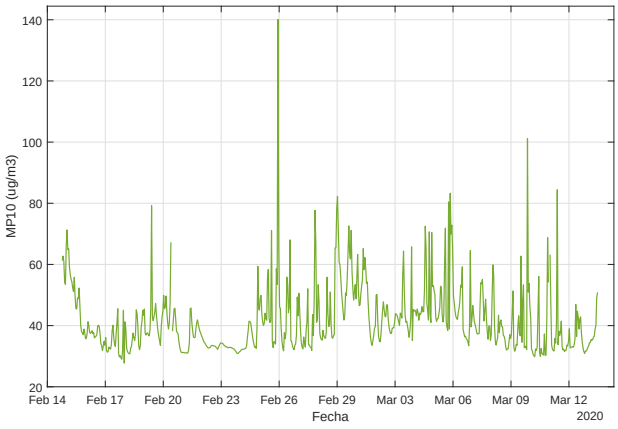


Figura 35: MP_{10} medido en todos los microambientes visitados.

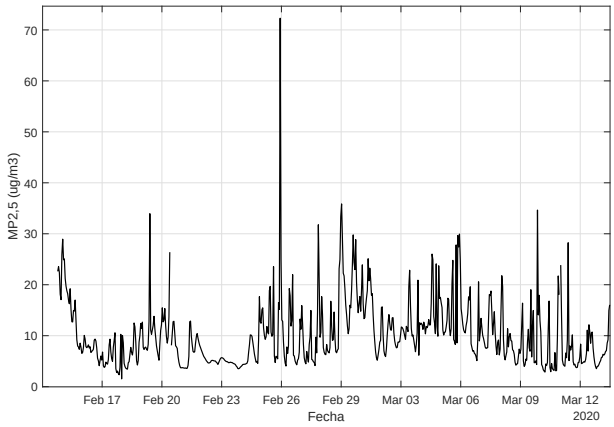


Figura 36: $MP_{2,5}$ medido en todos los microambientes visitados.

La alta concentración de $140\mu g/m^3$ de MP_{10} y $70\mu g/m^3$ de $MP_{2,5}$ medida el día 25 de febrero a las 04:30 horas fue provocada por un incendio cercano al microambiente Hogar.

4.2 Dióxido de Nitrógeno y Ozono

Utilizando los resultados del ajuste lineal multivariable, se ajustaron todos los datos medidos, la medición total incluyendo todos los microambientes y el periodo de ajuste con la referencia se muestra en las figuras 37 y 38.

Se observan valores de concentración negativos, con mayor frecuencia en el ajuste de O_3 a pesar que su correlación con la estación de referencia es $r^2 = 0,77$, estos valores se pueden deber a que se ha considerado 16 variables para ajustar 48 datos, lo que puede provocar que las variables de ajuste y su rango no sean representativos de la medición, puede existir variables no consideradas, variables que no influyen o que el ajuste realizado no sea aplicable a todo el rango de medición. Otras causas de error pueden ser inestabilidad de los sensores, posiblemente provocada por el movimiento, cambios bruscos en las condiciones meteorológicas. Los valores negativos no han sido considerados en el cálculo de la exposición.

Además se observa que entre el 26 y el 28 de febrero, se registran valores muy altos de O_3 y valores muy bajos de NO_2, esto puede ser debido a que las variables de ajuste, están sobre o bajo los niveles en los que se realizó el ajuste lineal multivariable.

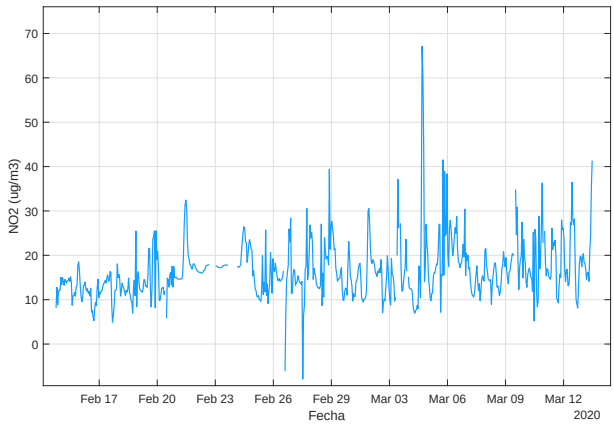


Figura 37: NO_2

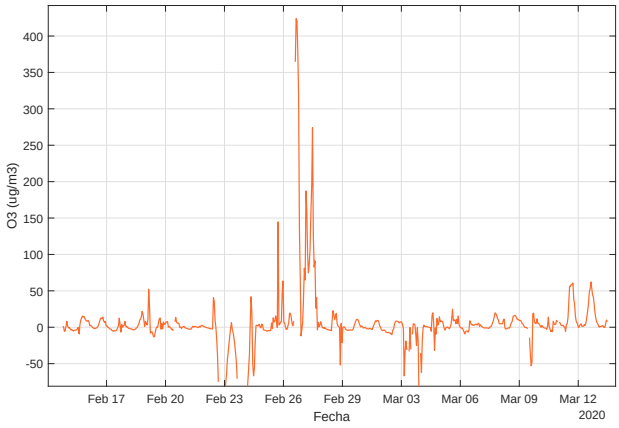


Figura 38: O_3

Además se ha graficado en la figura 39 uno de los días de medición (marzo 05) en el que se puede apreciar el máximo de O_3 que se produce con el máximo de radiación

y luego como éste se reduce y aumenta las concentraciones de NO_2, lo que es provocado por la reacción de los óxidos de nitrógeno con el ozono y la radiación.

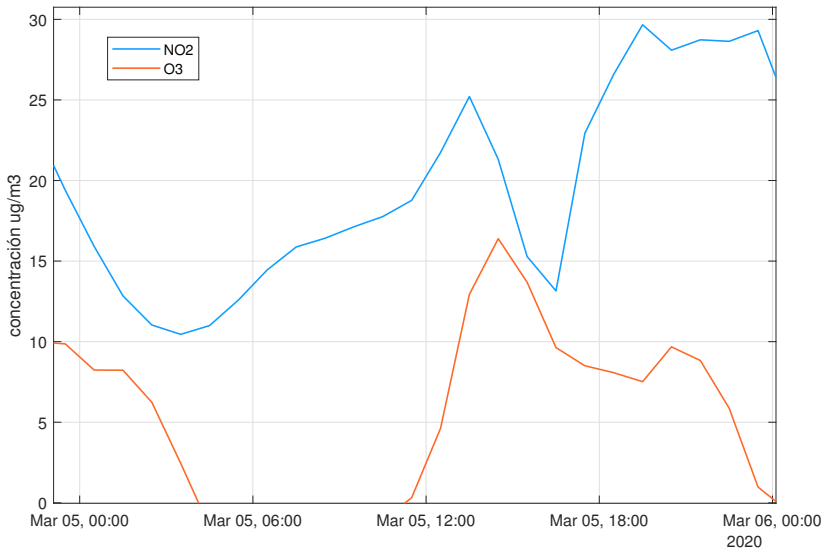


Figura 39: variación de NO_2 y O_3 el día 05 de marzo de 2020.

4.3 Resultados

Los resultados de la exposición personal calculada en cada uno de los microambientes se muestran en la tabla 8, donde DEV es la desviación standard.

Tabla 8: Exposición calculada promedio en cada microambiente

Microambiente	NO2	DEV	O3	DEV	MP_{10}	DEV	$MP_{2,5}$	DEV
Hogar	16.22	5.59	4.46	3.50	43.71	32.58	11.90	4.70
Bus	32.93	5.94	9.00	3.76	60.84	28.84	21.22	2.29
Metro	20.15	4.01	11.78	5.98	47.89	37.75	13.82	6.85
Oficina	21.21	13.64	6.07	4.60	42.73	35.87	10.80	5.67
Laboratorio	17.79	7.84	12.40	5.54	44.05	40.93	11.11	7.97

Considerando una persona que permanece en la oficina 9 horas de lunes a viernes (09 a 18 horas), se desplaza en metro durante 1 hora al día y permanece 14 horas en el hogar, su exposición promedio a los contaminantes medidos son:

- NO_2 = 18,25 $\mu g/m^3h$
- O_3 = 5,37 $\mu g/m^3h$
- PM_{10} = 43,52 $\mu g/m^3h$
- $PM_{2,5}$ = 11,57 $\mu g/m^3h$

CAPÍTULO 5:

CONCLUSIONES

En el presente trabajo se ha logrado construir un nodo sensor portátil de bajo costo para medir las concentraciones en el aire de $MP_{2,5}$, MP_{10}, NO_2 y O_3. Este sensor se ha utilizado para medir la exposición personal en 5 microambientes, 3 de los cuales son microambientes fijos (hogar, oficina y laboratorio) y 2 microambientes móviles (bus, metro).

Los sensores de MP utilizados en este trabajo, funcionan bajo el principio de dispersión óptica infrarroja, la respuesta de estos sensores depende del fabricante, modelo y calibración del sensor, distintos modelos muestran distintas eficiencias a distintos diámetros aerodinámico de partícula. Los sensores utilizados Sensirion SPS30 y Novasensor SDS011 poseen un máximo de eficiencia en torno a los $1\mu m$, lo que resulta adecuado para medir $MP_{2,5}$, pero no para medir MP_{10}, el sensor SPS30 posee una eficiencia de detección menor a 10 % para detectar partículas mayores a $2\mu m$, mientras que el sensor SDS011 posee una eficiencia menor a 20 % para este mismo rango, por lo tanto las mediciones de MP_{10}, corresponden principalmente a $MP_{2,5}$, lo que se evidencia en los resultados obtenidos. Debido a la metodología de funcionamiento, el tamaño mínimo medido en ambientes con bajo número de partículas es $0,3\mu m$, partículas más pequeñas generan respuestas del orden del ruido eléctrico del sensor. Las fuentes de interferencia de los sensores de MP son la temperatura y humedad relativa.

Los sensores de gas funcionan bajo el principio de oxido reducción, su eficiencia en la medición de los gases es muy alta en condiciones de laboratorio, donde solo se mide un gas, pero en condiciones reales de medición, se encuentran afectados por la interferencia de otros gases (ya que son sensibles a un conjunto de compuestos) y las condiciones de temperatura y humedad relativa. Estos sensores poseen un electrodo de referencia que

se encarga de estabilizar al electrodo principal, el potencial de este electrodo se mide a través del electrodo auxiliar y su valor también cambia con las concentraciones del gas objetivo. Se ha realizado un modelo de ajuste lineal multivariable considerando estas variables (temperatura, humedad relativa y voltaje en el electrodo auxiliar), los datos se ajustan a la estación de referencia y las variables de ajuste influyen significativamente en el resultado. Al ajustar el tiempo de medición en los microambientes, se observan valores negativos, más recurrentemente en O_3, a pesar que los datos obtenidos usando el modelo se ajustan a la estación de referencia ($r^2 = 0,77$). Los valores negativos indican error del ajuste, producido fundamentalmente por el reducido periodo de calibración (10 al 13 de marzo de 2020), que puede no haber cubierto un rango representativo de datos. También se puede producir error al medir en movimiento. Es necesario realizar más pruebas en la estación de referencia y un tiempo de ajuste más largo para mejorar el modelo. Sólo se han considerado los resultados positivos y que se encuentran en el rango de funcionamiento de estos sensores.

Se ha evaluado la exposición personal de una persona considerando 14 horas en el hogar, 9 horas en oficina y 1 hora en transporte (metro). La exposición más alta es NO_2 (18,25 $\mu g/m^3h$) luego MP_{10} (43,52 $\mu g/m^3h$), $MP_{2,5}$ (11,57 $\mu g/m^3h$) y O_3 (5,37 $\mu g/m^3h$). El microambiente con mayor concentración de contaminantes es el microambiente bus que muestra las mayores concentraciones excepto de O_3, los buses de Transantiago, a pesar de su tecnología reducida en emisiones (EURO VI) pueden estar afectados por las emisiones del tráfico urbano, los que se consideran hotspot según estos resultados. El microambiente con menor concentración es hogar, seguido de oficina.

Debido a los errores en el ajuste de los sensores electroquímicos, la eficiencia de detección de los sensores de MP y el reducido periodo de calibración, las evaluaciones para todos los ambientes y contaminantes deben ser vistos en forma cualitativa mas que cuantitativa.

La exposición personal a estos contaminantes puede ser reducida en el microambiente con mayor permanencia (hogar), aumentando la aislación de los hogares y permitiendo su ventilación en caso de existir mayores concentraciones dentro del hogar, para lo cual es necesario medir. Se recomienda el uso de metro antes de bus y evitar las calles congestionadas.

Los resultados obtenidos no se contradicen con otros estudios de exposición personal en ambientes urbanos (ver [21], [14], [40], [39]), no obstante es necesario au-

mentar el tiempo de ajuste en la estación de referencia, especialmente en el caso de los sensores de gas y explorar más profundamente el impacto de las variables de interferencia como el movimiento, temperatura y humedad en los resultados. También es necesario estudiar en más profundidad el impacto de las características del microambiente en la exposición personal como el impacto de la ventilación de los hogares y las actividades más perjudiciales.

Con el desarrollo y mejora de estas técnicas de medición de bajo costo, junto a las nuevas tecnologías de conectividad de bajo costo, es posible desarrollar una mega red urbana de sensores conectados para medir calidad del aire y otros parámetros, estas redes son la fundación de las ciudades inteligentes (*smart cities*) y son necesarios para asegurar el bienestar de sus habitantes.

Además, el nodo sensor desarrollado en este trabajo puede ser utilizado para medir las concentraciones en espacios cerrados, representando un indicador de aislación y ventilación en estos espacios. Además, contribuirá a que las personas tomen conciencia de su exposición a concentraciones altas y a generar estrategias de reducción. Reduciendo la exposición se puede reducir el número de admisiones hospitalarias y de muertes prematuras, lo que produce un ahorro en salud pública y un aumento del bienestar de las personas. En salud ocupacional puede representar una herramienta para conocer las actividades y microambientes con mayor exposición y así ayudar a generar medidas de abatimiento, como el uso protecciones (mascarillas).

CAPÍTULO 6:

APLICACIONES

Estos nodos sensores resultan un importante suplemento a la medición de la contaminación del aire que permiten extender las redes de medición, configurando lo que se conoce como *Hyperlocal real-time data* que permitirán medir en muy alta resolución espacial y temporal las concentraciones de contaminantes en el aire. Además, permiten medir la concentración dentro del hogar y en lugares cerrados. En este contexto, el 28 de noviembre de 2019 fue presentada la ”Red Experimental de medición de calidad de aire”, con el apoyo de diversas empresas tecnológicas del sector. Esta red experimental está compuesta por nodos sensores similares a los mostrados en esta tesis en su versión reducida, estos miden material particulado y parámetros meteorológicos (T, P, y HR). Los datos son transmitidos a través de la red Nb-IoT y Cat-M1 de Entel, utilizando las antenas de Ericsson.

Estos nodos permitirán medir $MP_{2,5}$ en diversos puntos fijos y móviles. Se planea dispersarlos en ciudades contaminadas de tal forma que permitirán medir hotspot e investigar e informar con gran resolución las concentraciones.

Para medir en movimiento se utiliza un GPS que detecta cuando el vehículo se encuentra detenido y activa los sensores de MP, luego envía las concentraciones y la ubicación geográfica a la nube digital, donde se implementan modelos de interpolación espacial con el fin de generar un mapa de calidad de aire que se actualice en tiempo real y que pueda ser utilizado por la ciudadanía para tomar mejores decisiones.

NOTICIAS | AGENDA DESTACADOS

Digital

Lanzan red experimental de medición de calidad del aire a través de tecnología IoT

por Agenda País | 23 diciembre, 2019

En el marco del seminario Internet de las Cosas - IoT para enfrentar los desafíos de la sociedad 2019, organizado por la Embajada de Suecia en Chile, Centro Mario Molina junto con Entel Ocean y Ericsson lanzaron la "Red experimental de medición de calidad del aire", que tiene como objetivo medir la calidad del aire a través de sensores fijos y móviles.

Videos

China ordena cierre del consulado de EE.UU. en Chengdu

Corte Suprema condena amenazas que recibió juez que no dejó con prisión preventiva a Martín Pradenas

Reportan caída de granizos y nieve en el sector oriente de Santiago

Oposición califica decisión de Allamand como un "acto de realismo" y emplaza a Piñera a promulgar el retiro de fondos para que la gente empiece a retirar su 10%

1. **Air quality monitoring with 4G and 5G**
To push the boundaries of collaboration yet further, we agreed to build a consortium with Triple Helix across government, academia and industry.

In November 2019, together with Centro Mario Molina, ENTEL, we launched the first experimental air quality network using 4,5G technologies. Normally an area is monitored using a fixed sensor, but in Chile for the first time, one sensor in movement can monitor thousands of kilometers along a path.

This challenge was a success thanks to the trucks provided by Scania and Transporte Ricardo Concha as well as buses provided by Volvo and Subus. This is a perfect example on how partnerships across sectors and expertise areas can enable us to accelerate the reduction of greenhouse gas emissions. This, of course, is a critical demand of the exponential roadmap to halve emissions by 2030.

Figura 40: Proyecto experimental de sensores en movimiento.

Fuente:

https://www.ericsson.com/en/blog/2019/12/cop25-chile-5g-smart-society

Figura 41: Nodo sensor instalado en helipuerto de la Torre Entel

Figura 42: Instalación en camión Scania

REFERENCIAS BIBLIOGRÁFICAS

[1] Afshar-Mohajer, Nima y col. «Evaluation of low-cost electro-chemical sensors for environmental monitoring of ozone, nitrogen dioxide, and carbon monoxide». En: *Journal of Occupational and Environmental Hygiene* 15.2 (oct. de 2017), págs. 87-98. DOI: 10.1080/15459624.2017.1388918.

[2] *Air pollution causes "hugereduction in intelligence, study reveals*. The Guardian. 2018. URL: https://www.theguardian.com/environment/2018/aug/27/air-pollution-causes-huge-reduction-in-intelligence-study-reveals (visitado 15-11-2018).

[3] ALOYAN, A. E., ARUTYUNYAN, V. O. y MARCHUK, G. I. «Dynamics of mesoscale boundary atmospheric layer and impurity spreading with the photochemical transformation allowed for». En: *Russian Journal of Numerical Analysis and Mathematical Modelling* 10.2 (1995). DOI: 10.1515/rnam.1995.10.2.93.

[4] Aloyan, A.E. y col. «Transport of coagulating aerosol in the atmosphere». En: *Journal of Aerosol Science* 28.1 (ene. de 1997), págs. 67-85. DOI: 10.1016/s0021-8502(96)00043-2.

[5] Alphasense. *Alphasense How the Electrochemical Gases Works*. 2013. URL: http://www.alphasense.com/WEB1213/wp-content/uploads/2013/07/AAN_104.pdf.

[6] Badura, Marek y col. «Evaluation of Low-Cost Sensors for Ambient PM2.5 Monitoring». En: *Journal of Sensors* 2018 (oct. de 2018), págs. 1-16. DOI: 10.1155/2018/5096540.

[7] Bigi, Alessandro y col. «Performance of NO, NO₂ low cost sensors and three calibration approaches within a real world application». En: *Atmospheric Measurement Techniques* 11.6 (jun. de 2018), págs. 3717-3735. DOI: 10.5194/amt-11-3717-2018.

[8] Bornholdt, Jette y col. «Inhalation of ozone induces DNA strand breaks and inflammation in mice». En: *Mutation Research/Genetic Toxicology and Environmental Mutagenesis* 520.1-2 (sep. de 2002), págs. 63-72. DOI: 10.1016/s1383-5718(02)00176-6.

[9] Borrego, C. y col. «Assessment of air quality microsensors versus reference methods: The EuNetAir joint exercise». En: *Atmospheric Environment* 147 (2016), págs. 246-263. DOI: 10.1016/j.atmosenv.2016.09.050.

[10] Castell, Nuria y col. «Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?» En: *Environment International* 99 (2017), págs. 293-302. DOI: 10.1016/j.envint.2016.12.007.

[11] Frontera, Antonio y col. «Regional air pollution persistence links to COVID-19 infection zoning». En: *Journal of Infection* (abr. de 2020). DOI: 10.1016/j.jinf.2020.03.045.

[12] Gerba, Charles P. «Risk Assesment and Environmental Regulations». En: *Environmental Monitoring and Characterization*. Elsevier, 2004, págs. 377-392. DOI: 10.1016/b978-012064477-3/50022-9.

[13] Hasenfratz, David y col. «Deriving high-resolution urban air pollution maps using mobile sensor nodes». En: *Pervasive and Mobile Computing* 16 (2015), págs. 268-285. DOI: 10.1016/j.pmcj.2014.11.008.

[14] Hwang, Yunhyung y Lee, Kiyoung. «Contribution of microenvironments to personal exposures to PM10 and PM2.5 in summer and winter». En: *Atmospheric Environment* 175 (2018), págs. 192-198. DOI: 10.1016/j.atmosenv.2017.12.009.

[15] IARC. *PRESS RELEASE N 221*. Ed. por International Agency for Research on Cancer. 15 de oct. de 2018. URL: https://www.iarc.fr/en/media-centre/iarcnews/pdf/pr221_E.pdf.

[16] Jensen, Christian Kjær. «Assessing the applicability of low-costelectrochemical gas sensors for urban airquality monitoring». Technical University ofDenmark, Department of Environmental Engineering, 2016.

[17] Jeschke, Klaus Nielsen y col. «Guideline for the management of COVID-19 patients during hospital admission in a non-intensive care setting». En: *European Clinical Respiratory Journal* 7.1 (ene. de 2020), pág. 1761677. DOI: 10.1080/20018525.2020.1761677.

[18] Karagulian, Federico y col. «Contributions to cities' ambient particulate matter (PM): A systematic review of local source contributions at global level». En: *Atmospheric Environment* 120 (2015), págs. 475-483. DOI: 10.1016/j.atmosenv.2015.08.087.

[19] Karagulian, Federico y col. «Review of the Performance of Low-Cost Sensors for Air Quality Monitoring». En: *Atmosphere* 10.9 (ago. de 2019), pág. 506. DOI: 10.3390/atmos10090506.

[20] Kumar, Prashant y col. «The rise of low-cost sensing for managing air pollution in cities». En: *Environment International* 75 (2015), págs. 199-205. DOI: 10.1016/j.envint.2014.11.019.

[21] Kumar, Prashant y col. «A review of factors impacting exposure to PM 2.5 , ultrafine particles and black carbon in Asian transport microenvironments». En: *Atmospheric Environment* 187 (2018), págs. 301-316. DOI: 10.1016/j.atmosenv.2018.05.046.

[22] Kuula, Joel y col. «Laboratory evaluation of particle size-selectivity of optical low-cost particulate matter sensors». En: (dic. de 2020). DOI: 10.5194/amt-2019-422.

[23] Li, Jiayu. «Recent advances in low-cost particulate matter sensor: calibration and application». En: *Engineering and Applied Science Theses and Dissertations* (2019).

[24] Malky, Saeed Faisal. «Air Quality Monitoring Wireless Sensor Network with Electrochemical Sensing Elements». Tesis doct. Florida Institute of Technology, 2019.

[25] Martelletti, Luigi y Martelletti, Paolo. «Air Pollution and the Novel Covid-19 Disease: a Putative Disease Risk Factor». En: *SN Comprehensive Clinical Medicine* 2.4 (abr. de 2020), págs. 383-387. DOI: 10.1007/s42399-020-00274-4.

[26] Mead, M.I. y col. «The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks». En: *Atmospheric Environment* 70 (2013), págs. 186-203. DOI: 10.1016/j.atmosenv.2012.11.060.

[27] Mijling, Bas y col. «Field calibration of electrochemical NO2 sensors in a citizen science context». En: *Atmospheric Measurement Techniques* 11.3 (mar. de 2018), págs. 1297-1312. DOI: 10.5194/amt-11-1297-2018.

[28] Ogen, Yaron. «Assessing nitrogen dioxide (NO2) levels as a contributing factor to coronavirus (COVID-19) fatality». En: *Science of The Total Environment* 726 (jul. de 2020), pág. 138605. DOI: 10.1016/j.scitotenv.2020.138605.

[29] Ran, Liang y col. «Ozone photochemical production in urban Shanghai, China: Analysis based on ground level observations». En: *Journal of Geophysical Research* 114.D15 (ago. de 2009). DOI: 10.1029/2008jd010752.

[30] *Reglamento de estaciones de estaciones de medicion de contaminantes atmosfericos. Chile*. Subsecretaria de Salud Publica. URL: https://www.leychile.cl/Navegar?idNorma=281728.

[31] Reilly, John P. y col. «Low to Moderate Air Pollutant Exposure and Acute Respiratory Distress Syndrome after Severe Trauma». En: *American Journal of Respiratory and Critical Care Medicine* 199.1 (ene. de 2019), págs. 62-70. DOI: 10.1164/rccm.201803-0435oc.

[32] Reis, Stefan y col. «Integrating modelling and smart sensors for environmental and human health». En: *Environmental Modelling & Software* 74 (2015), págs. 238-246. DOI: 10.1016/j.envsoft.2015.06.003.

[33] Schneider, Philipp y col. «Mapping urban air quality in near real-time using observations from low-cost sensors and model information». En: *Environment International* 106 (sep. de 2017), págs. 234-247. DOI: 10.1016/j.envint.2017.05.005.

[34] Setti, Leonardo y col. «SARS-Cov-2 RNA Found on Particulate Matter of Bergamo in Northern Italy: First Preliminary Evidence». En: (abr. de 2020). DOI: 10.1101/2020.04.15.20065995.

[35] Sharma, Sumit, Sharma, Prateek y Khare, Mukesh. «Photo-chemical transport modelling of tropospheric ozone: A review». En: *Atmospheric Environment* 159 (2017), págs. 34-54. DOI: 10.1016/j.atmosenv.2017.03.047.

[36] Shehab, M. A. y Pope, F. D. «Effects of short-term exposure to particulate matter air pollution on cognitive performance». En: *Scientific Reports* 9.1 (jun. de 2019). DOI: 10.1038/s41598-019-44561-0.

[37] Spinelle, L. y col. *Evaluation of low-cost sensors for air pollution monitoring*. 2017. ISBN: 978-92-79-68830-0. DOI: 10.2760/548327.

[38] Spinelle, Laurent, Gerboles, Michel y Aleixandre, Manuel. «Performance Evaluation of Amperometric Sensors for the Monitoring of O 3 and NO 2 in Ambient Air at ppb Level». En: *Procedia Engineering* 120 (2015), págs. 480-483. DOI: 10.1016/j.proeng.2015.08.676.

[39] Suárez, Liliana y col. «Personal exposure to particulate matter in commuters using different transport modes (bus, bicycle, car and subway) in an assigned route in downtown Santiago, Chile». En: *Environ. Sci.: Processes Impacts* 16.6 (2014), págs. 1309-1317. DOI: 10.1039/c3em00648d.

[40] Tan, Sok Huang, Roth, Matthias y Velasco, Erik. «Particle exposure and inhaled dose during commuting in Singapore». En: *Atmospheric Environment* 170 (2017), págs. 245-258. DOI: 10.1016/j.atmosenv.2017.09.056.

[41] USEPA. *Guidelines for Exposure Assessment*. U.S. Environmental Protection Agency. 12 de ago. de 2018. URL: https://www.epa.gov/sites/production/files/2014-11/documents/guidelines_exp_assessment.pdf.

[42] USEPA. *Nitrogen Dioxide (NO2) Pollution*. U.S. Environmental Protection Agency. 2 de nov. de 2018. URL: https://www.epa.gov/no2-pollution/basic-information-about-no2#What%20is%20NO2.

[43] Vallero, Daniel. *Fundamentals of Air Pollution*. Elsevier, 2014. DOI: 10.1016/c2012-0-01172-6.

[44] Vardoulakis, Sotiris y col. «Modelling air quality in street canyons: a review». En: *Atmospheric Environment* 37.2 (ene. de 2003), págs. 155-182. DOI: 10.1016/s1352-2310(02)00857-9.

[45] W. Pearce, C.J. Baker. «Wind-tunnel investigation of the effect of vehicle motion on dispersion in urban canyons». En: *Journal of Wind Engineering and Industrial Aerodynamics* 69-71 (jul. de 1997), págs. 915-926. DOI: 10.1016/s0167-6105(97)00217-1.

[46] Wayne, Davis T. y Joshua, Fu S. *Air Quality*. Taylor y Francis Inc, 15 de ago. de 2014. 542 págs. ISBN: 1466584440. URL: https://www.ebook.de/de/product/22736316/thad_ball_state_university_muncie_indiana_usa_godish_wayne_t_university_of_tennessee_knoxville_usa_davis_joshua_s_university_of_knoxville_tennessee_usa_fu_air_quality.html.

[47] WHO. *Ambient air pollution: Health impacts*. Ed. por World Health Organization. 30 de sep. de 2018. URL: http://www.who.int/airpollution/ambient/health-impacts/en.

[48] Wu, Xiao y col. «Exposure to air pollution and COVID-19 mortality in the United States: A nationwide cross-sectional study». En: (abr. de 2020). DOI: 10.1101/2020.04.05.20054502.

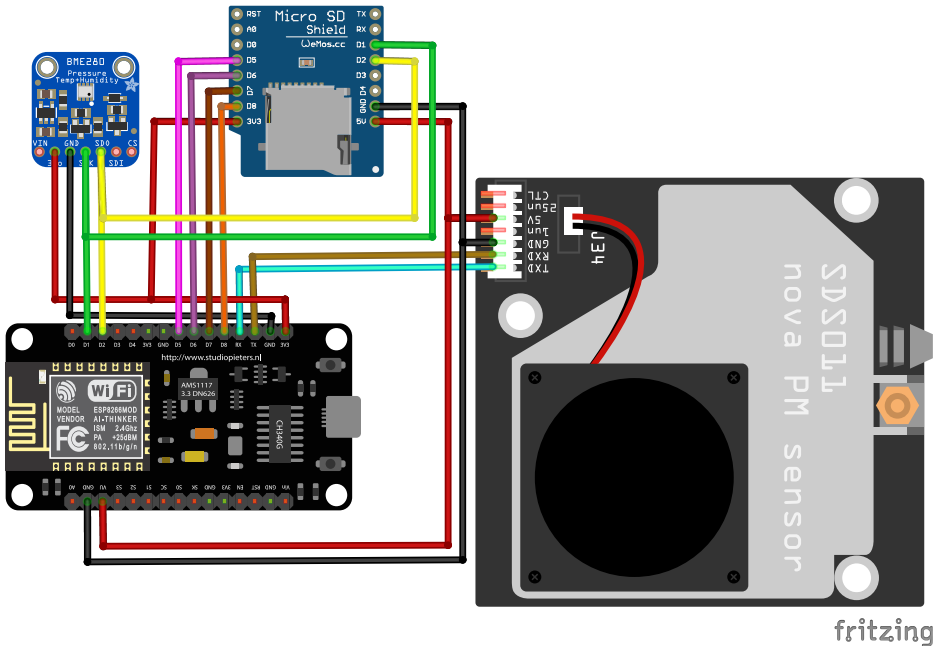
[49] Zhang, Xin, Chen, Xi y Zhang, Xiaobo. «The impact of exposure to air pollution on cognitive performance». En: *Proceedings of the National Academy of Sciences* 115.37 (2018), págs. 9193-9197. DOI: 10.1073/pnas.1809474115.

Anexos

DIAGRAMAS ESQUEMÁTICOS

1 Nodo Sensor 1

Las conexiones entre componentes y los protocolos de comunicación se muestran en las figuras 43 y 44.



Dispositivo	Función	Conexión
ESP8266 NodeMCU	Gestión de datos WiFi	-
Adafruit BME280	Sensor de T, P y HR	I2C (D1 y D2)
Wemos microSD + RTC (real time clock)	Guardado de datos	SPI (D5, D6, D7, D8)
	Medidor de tiempo	I2C (D1 y D2)
Nova Sensor SDS011	Medidor de MP	UART (TX RX)

Figura 43: Conexiones del Nodo sensor V1.

2 Nodo Sensor 2

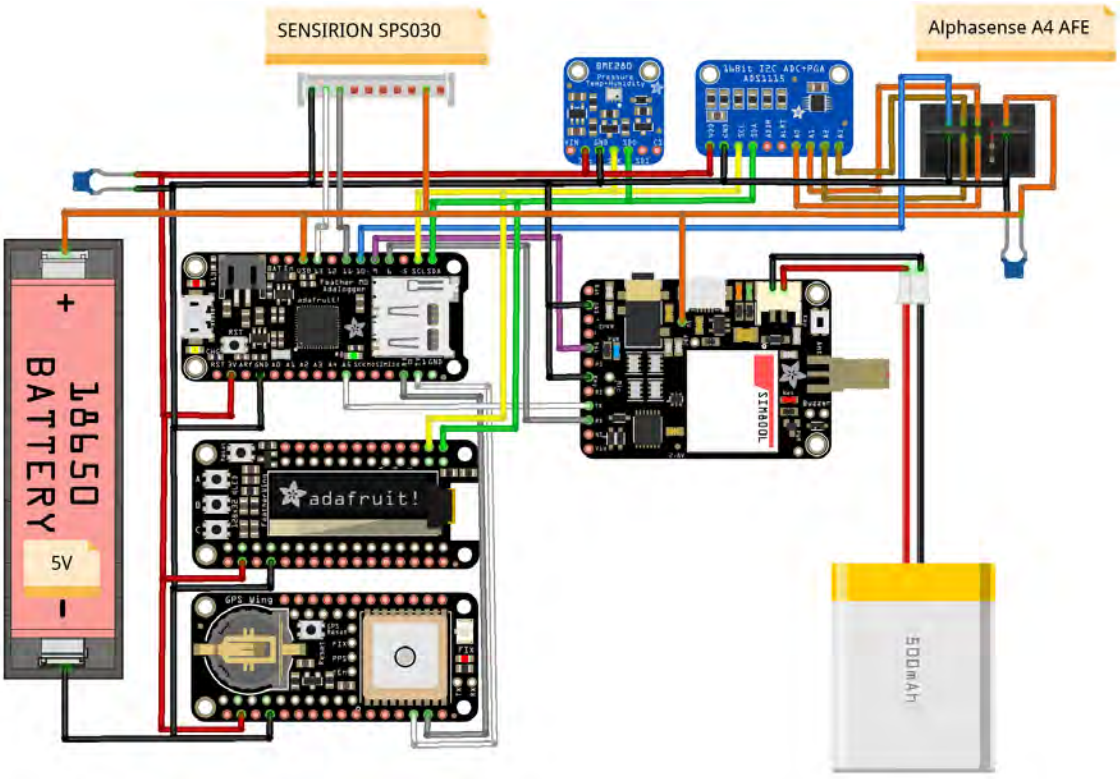


Figura 44: Conexiones del Nodo sensor V2.

Tabla 9: Componentes Nodo sensor v2.

Dispositivo	Función	Conexión
Adafruit m0 Adalogger	Gestión de datos microSD	-
Adafruit BME280	Sensor de T, P y HR	I2C (SCL y SDA)
Adafruit RTC	Medidor de tiempo	I2C (SDL y SDA)
Adafruit Screen	Pantalla monocromática	I2C (SCL y SDA)
Adafruit ADS1115	ADC 16 bits	I2C (SCL y SDA)
Sensirion SPS030	Medidor de MP	UART2 (11 y 13)
Adafruit GPS	GPS	UART1 (TX RX)
Adafruit Fona	Conectividad celular	UART3 (6 y 9)
Alphasense	Sensores de gas NO2 y O3	ADS1115

PROGRAMACIÓN

Se ha escrito un programa compatible con los 2 modelos de nodo sensor desarrollado, escrito en lenguaje Arduino (C y C++) utilizando la plataforma de programación Platform-io y la interfase Visual Studio Code. El esquema de programación se observa en la tabla 2.

Configuración	microSD	Leer config.: ID, taza de muestreo, conexión y credenciales
	Sensores	Iniciar sensores (Meteorología, MP, ADS, GPS)
	Internet	Conectar a internet, verificar servidores remotos
	Tiempo	Iniciar medición de tiempo, sincronizar el reloj con NTP
Ciclo	Tiempo	Esperar el tiempo de muestreo.
	Sensores	Medir parámetros 20 veces, calcular promedio y desviación standard
	Gestión	Construir línea de datos: hh:mm:ss datos
	microSD	Guardar datos en archivo ÏD-YYMMDD.csv”
	Internet	Transmisión de datos
	Tiempo	Guardar el tiempo

Tabla 10: Programación del Nodo sensor

```
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GPS.h>

#if defined(ESP8266)
#include <ESP8266WiFi.h>
#include <AdafruitIO_WiFi.h>
#endif

#if defined(ESP32)
#include <WiFi.h>
#endif

#if defined(ESP8266) || defined(ESP32)
#include <WiFiUdp.h>
#include <SD.h>
#include <SoftwareSerial.h>
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)
#include <avr/dtostrf.h>
#include <strings.h>
#include <Adafruit_FONA.h>
#include <wiring_private.h>
#include <SdFat.h>
#include <Adafruit_ADS1015.h>
#endif

#include <RTClib.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Adafruit_BME680.h>
#include <sps30.h>
#include <sensirion_uart.h>
#include <SdsDustSensor.h>
#include <AvgStd.h>

// Select PM Sensor and Serial Output
#define SPS030_ 0
#define SDS011_ 1
#define BME280_ 1
#define BME680_ 0
#define ADFONA_ 0
#define GPS_attach 0
#define Serial_Out 0
#define ADS1115 0
#define MQTT_ 1
#define microSD_ 1
#define Sync_Mode 0
int tw_ = 0;           //Time wait Sync mode
bool debug_Ind = true; // Serial Debug?

#if (MQTT_)
//MQTT Client Congiguration
#define IO_USERNAME ""
#define IO_KEY ""
#endif

// BME Configuration
#if (BME280_)
bool BME_280;            // BME280 Indicator
bool gen_BME280 = false; // Adafruit Genuine?
Adafruit_BME280 bme;     // I2C
#endif

#if (BME680_)
```

```
Adafruit_BME680 bme_680; // I2C
#endif

/*-----------------------------------------------------
   ESP8266 & ESP32 Configuration
-----------------------------------------------------*/
#if defined(ESP8266)
RTC_DS1307 rtc;
#define SD_PIN D8
#define SDS_RX D3
#define SDS_TX D4
HardwareSerial &P_SPS30 = Serial;
int LED_PORT = 2;
//SdFat SD;
#if (SDS011_)
long BR_ = 9600L;
SdsDustSensor sds(Serial);
#else
long BR_ = 115200L;
#endif
#endif

#if defined(ESP32)
RTC_PCF8523 rtc;
#define SD_PIN A9
#define SDS_RX A0
#define SDS_TX A1
#define PM_RX A2
#define PM_TX A3
#if (SPS030_)
HardwareSerial &P_SPS30 = Serial1;
#endif
int LED_PORT = 13;
#if (GPS_attach) && (SDS011_)
SoftwareSerial SerialYY(PM_RX, PM_TX);
SdsDustSensor sds(SerialYY);
#elif (SDS011_)
SdsDustSensor sds(Serial1);
#endif
long BR_ = 115200L;
#endif

#if (GPS_attach)
#define GPSSerial Serial1
Adafruit_GPS GPS(&GPSSerial);
#define GPSECHO false
uint32_t timer = millis();
#endif

#if defined(ESP8266) || defined(ESP32)
WiFiUDP udp;
#if (Serial_Out)
SoftwareSerial SerialX(SDS_RX, SDS_TX);
#endif
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)
/*-----------------------------------------------------
   FEATHER M0 Configuration
-----------------------------------------------------*/
AvgStd mySamples;
RTC_DS3231 rtc;
#if (ADS1115)
Adafruit_ADS1115 ads;
#endif
//int PM_PORT = 1;
#define SD_PIN 4
```

```
SdFat SD;
#define CSV_DELIM ’,’
/*-----------------------------------------------------
   Serial1 pin and pad definitions (in Arduino files Variant.h & modified Variant.cpp)
*/
#define PIN_SERIAL1_RX (0ul)             // Pin description number for PIO_SERCOM on D0
#define PIN_SERIAL1_TX (1ul)             // Pin description number for PIO_SERCOM on D1
#define PAD_SERIAL1_TX (UART_TX_PAD_2)   // SERCOM pad 2
#define PAD_SERIAL1_RX (SERCOM_RX_PAD_3) // SERCOM pad 3
/*-----------------------------------------------------
   Serial2 pin and pad definitions (in Arduino files Variant.h & Variant.cpp)
*/
#define PIN_SERIAL2_RX (37ul)            // Pin description number for PIO_SERCOM on D13
#define PIN_SERIAL2_TX (35ul)            // Pin description number for PIO_SERCOM on D11
#define PAD_SERIAL2_TX (UART_TX_PAD_0)   // SERCOM pad 0
#define PAD_SERIAL2_RX (SERCOM_RX_PAD_1) // SERCOM pad 1
/*-----------------------------------------------------
   Serial3 pin and pad definitions (in Arduino files Variant.h & modified Variant.cpp)
*/
#define PIN_SERIAL3_RX (47ul)            // Pin description number for PIO_SERCOM on A5
#define PIN_SERIAL3_TX (46ul)            // Pin description number for PIO_SERCOM on D6
#define PAD_SERIAL3_TX (UART_TX_PAD_2)   // SERCOM pad 0
#define PAD_SERIAL3_RX (SERCOM_RX_PAD_0) // SERCOM pad 1
/*-----------------------------------------------------
   Serials config class
*/
Uart Serial1(&sercom0, PIN_SERIAL1_RX, PIN_SERIAL1_TX, PAD_SERIAL1_RX, PAD_SERIAL1_TX);
Uart Serial2(&sercom1, PIN_SERIAL2_RX, PIN_SERIAL2_TX, PAD_SERIAL2_RX, PAD_SERIAL2_TX);
Uart Serial3(&sercom5, PIN_SERIAL3_RX, PIN_SERIAL3_TX, PAD_SERIAL3_RX, PAD_SERIAL3_TX);
void SERCOM0_Handler()
{
  Serial1.IrqHandler();
}
void SERCOM1_Handler()
{
  Serial2.IrqHandler();
}
void SERCOM5_Handler()
{
  Serial3.IrqHandler();
}
#if (SPS030_)
HardwareSerial &P_SPS30 = Serial2;
#endif
#if (SDS011_)
SdsDustSensor sds(Serial2);
#endif
long BR_ = 115200L;
/*-----------------------------------------------------
   FONA800H Configuration
-----------------------------------------------------*/
#define FONA_RST 9
#define VBATPIN A7
HardwareSerial *fonaSerial = &Serial3;
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
/*-----------------------------------------------------
   FONA800H GSM & FTP Server Configuration
-----------------------------------------------------*/
const char *apn_name = ”m2m.entel.cl”;
const char *apn_username = ””;
const char *apn_password = ””;
#endif

// Library definitions
File config_File;

#define SEALEVELPRESSURE_HPA (1013.25)
```

```
// Init Variables
float configg[8];
int Error_Ind = 0;
int configg2_temp;
char espcode[2];
int pulse_time;
uint64_t start99;
uint64_t start98;
int time_wait;
int FirstStart;
int len;
int otherDay;
int feed_aux;
short ret;
char latitud_[10];
char longitud_[10];
double the_latitud_;
double the_longitud_;

// Measure Variables
char *VM2 = (char *)”DD/MM/YY HH:MM, Temp, Press, HR, MP10, MP2.5”;
char *VM = (char *)”HH,MM,SS,Lat,Lon,Temp,Temp_d,Press,Press_d,HumR,Hum_d,PM10,
PM10_d,PM2.5, PM2.5_d, ADC_0, ADC_0_d, ADC_1, ADC_1_d, ADC_2, ADC_2_d, ADC_3, ADC_3_d”;
char *VM_ln = (char *)”HH,MM,SS,Lat,Lon,Temp,Temp_d,Press,Press_d,HumR,Hum_d,
PM10,PM10_d,PM2.5, PM2.5_d, ADC_0, ADC_0_d, ADC_1, ADC_1_d, ADC_2, ADC_2_d, ADC_3, ADC_3_d\n”;

int Ind_medicion;
char medicion[176];
char medicion_ln[177];
char medicion_S[120];
char JSONMessage[370];
char PM25FEED_[10];
char TOTALFEED_[10];
float temp_raw;
float pres_raw;
float humm_raw;
int pmcount;
float pm10_raw;
float pm25_raw;
struct sps30_measurement measurement;
char temp[10];
char pres[10];
char hume[10];
char temp_d[10];
char pres_d[10];
char hume_d[10];
float PM_10;
float PM_25;
char PM25_[10];
char PM25_d[10];
char PM10_[10];
char PM10_d[10];
char ADC_0[10];
char ADC_1[10];
char ADC_2[10];
char ADC_3[10];
char ADC_0_DEV[10];
char ADC_1_DEV[10];
char ADC_2_DEV[10];
char ADC_3_DEV[10];
char filename[25];
char filename2[25];
char filename_old[25];
char filename33[50];
File myFile;
File myFile2;
```

```
uint8_t SS_;

// Configuration Variables
char ssid[50];               // WiFi ID
char password[50];           // WiFi Password
char xhost[50];              // FTP Host
char xusername[50];          // FTP Username
char xpassword[50];          // FTP Password
char xfolder[50];            // FTP Folder
const char *xmode = "APPE";  // FTP Mode
const char *xmode2 = "STOR"; // FTP Mode
short FTPresult;             // outcome of FTP upload
short doFTP(char *, char *, char *, char *, char * = (char *)"");
unsigned int localPort = 2390;                // local port to listen for UDP packets
const char *ntpServerName = "time.google.com"; // NTP Server Name
const int NTP_PACKET_SIZE = 48;               // NTP Packet Size
byte packetBuffer[NTP_PACKET_SIZE];           // NTP Buffer

#if defined(ARDUINO_SAMD_FEATHER_M0)
/*
 * Read a file one field at a time.
 * file - File to read.
 * str - Character array for the field.
 * size - Size of str array.
 * delim - csv delimiter.
 * return - negative value for failure.
 *          delimiter, '\n' or zero(EOF) for success.
 */
int csvReadText(File *file, char *str, size_t size, char delim)
{
  char ch;
  int rtn;
  size_t n = 0;
  while (true)
  {
    // check for EOF
    if (!file->available())
    {
      rtn = 0;
      break;
    }
    if (file->read(&ch, 1) != 1)
    {
      // read error
      rtn = -1;
      break;
    }
    // Delete CR.
    if (ch == '\r')
    {
      continue;
    }
    if (ch == delim || ch == '\n')
    {
      rtn = ch;
      break;
    }
    if ((n + 1) >= size)
    {
      // string too long
      rtn = -2;
      n--;
      break;
    }
    str[n++] = ch;
  }
  str[n] = '\0';
```

```
  return rtn;
}
//---------------------------------------------------------------------------
int csvReadDouble(File *file, double *num, char delim)
{
  char buf[20];
  char *ptr;
  int rtn = csvReadText(file, buf, sizeof(buf), delim);
  if (rtn < 0)
    return rtn;
  *num = strtod(buf, &ptr);
  if (buf == ptr)
    return -3;
  while (isspace(*ptr))
    ptr++;
  return *ptr == 0 ? rtn : -4;
}
//---------------------------------------------------------------------------
int csvReadFloat(File *file, float *num, char delim)
{
  double tmp;
  int rtn = csvReadDouble(file, &tmp, delim);
  if (rtn < 0)
    return rtn;
  // could test for too large.
  *num = tmp;
  return rtn;
}
/*-------------------------------------------------------
   https://gist.github.com/jrleeman/3b7c10712112e49d8607
-------------------------------------------------------*/
int calculatedayOfYear(int dday, int mmonth, int yyear)
{

  // Given a day, month, and year (4 digit), returns
  // the day of year. Errors return 999.

  int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

  // Verify we got a 4-digit year
  if (yyear < 1000)
  {
    return 999;
  }

  // Check if it is a leap year, this is confusing business
  // See: https://support.microsoft.com/en-us/kb/214019
  if (yyear % 4 == 0)
  {
    if (yyear % 100 != 0)
    {
      daysInMonth[1] = 29;
    }
    else
    {
      if (yyear % 400 == 0)
      {
        daysInMonth[1] = 29;
      }
    }
  }

  // Make sure we are on a valid day of the month
  if (dday < 1)
  {
    return 999;
  }
```

```
  else if (dday > daysInMonth[mmonth - 1])
  {
    return 999;
  }

  int doy = 0;
  for (int i = 0; i < mmonth - 1; i++)
  {
    doy += daysInMonth[i];
  }

  doy += dday;
  return doy;
}
#endif

/*-------------------------------------------------------
   ESP8266 & ESP32 FUNCTION DEFINITION
-------------------------------------------------------*/
#if defined(ESP8266) || defined(ESP32)
IPAddress timeServerIP; // NTP IP
// Function NTP Sync
void sendNTPpacket(IPAddress &address)
{
  if (debug_Ind)
  {
    Serial.println("sending NTP packet...");
  }

  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011; // LI, Version, Mode
  packetBuffer[1] = 0;          // Stratum, or type of clock
  packetBuffer[2] = 6;          // Polling Interval
  packetBuffer[3] = 0xEC;       // Peer Clock Precision
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  udp.beginPacket(address, 123);
  udp.write(packetBuffer, NTP_PACKET_SIZE);
  udp.endPacket();
}

#if defined(ARDUINO_SAMD_FEATHER_M0)
// Function dateTime for FAT FileSystem
void dateTime(uint16_t *date, uint16_t *time)
{
  DateTime now = rtc.now();
  // return date using FAT_DATE macro to format fields
  *date = FAT_DATE(now.year(), now.month(), now.day());
  // return time using FAT_TIME macro to format fields
  *time = FAT_TIME(now.hour(), now.minute(), now.second());
}
#endif

/*-------------------------------------------------------
 * FUNCTION - eRcv
 * Reads the response from an FTP server and stores the
 * output in a buffer.Extracts the server return code from
 * the buffer.
 *
 * Parameters passed:
 *   aclient - a wifi client connected to FTP server and
 *   delivering the server response
 *   outBuf - a buffer to store the server response on
 *   size - size of the buffer in bytes
 *
```

```
 * Return codes:
 *    These are the first three chars in the buffer and are
 *    defined in
 *    https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes
 *
 * Dependencies:
 *   Libraries - <ESP8266WiFi.h> wifi library
 *   Functions - none
 ----------------------------------------------------------*/
short eRcv(WiFiClient aclient, char outBuf[], int size)
{
  unsigned long start = millis();
  byte thisByte;
  int index;
  String respStr = "";
  while (!aclient.available() && millis() - start < 60000)
    delay(1);
  index = 0;
  while (aclient.available() && millis() - start < 60000)
  {
    thisByte = aclient.read();
    if (debug_Ind)
    {
      Serial.write(thisByte);
    }
    if (index < (size - 2))
    { //less 2 to leave room for null at end
      outBuf[index] = thisByte;
      index++;
    }
  } //note if return from server is > size it is truncated.
  if (millis() - start >= 60000)
  {
    return 910;
  }
  outBuf[index] = 0; //putting a null because later strtok requires a null-delimited string
  //The first three bytes of outBuf contain the FTP server return code - convert to int.
  for (index = 0; index < 3; index++)
  {
    respStr += (char)outBuf[index];
  }
  return respStr.toInt();
} // end function eRcv

/*----------------------------------------------------------
 * FUNCTION - doFTP
 * Connects to a FTP server and transfers a file. Connection
 * is established using the FTP commands/responses defined at
 * https://en.wikipedia.org/wiki/List_of_FTP_commands or in
 * more detail at http://www.nsftools.com/tips/RawFTP.htm
 *
 * Parameters passed:
 *   host - the IP address of the FTP server
 *   uname - username for the account on the server
 *   pwd - user password
 *   filename - the file to be transferred
 *   folder (optional) - folder on the server where
 *   the file will be copied. This is may be omitted if
 *   the file is to be copied into the default folder on
 *   the server.
 *
 * Return codes:
 *    226 - a successful transfer
 *    400+ - any return code greater than 400 indicates
 *    an error. These codes are defined at
 *    https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes
 *    Exceptions to this are:
```

```
 *    900 - failed to open file on SD
 *    910 - failed to connect to server
 *
 * Dependencies:
 *   Libraries - <ESP8266WiFi.h> wifi library
 *               <SD.h> SD library
 *   Functions - eRcv
 ------------------------------------------------------------*/
short doFTP(char *host, char *uname, char *pwd, char *fileName, char *folder)
{

  WiFiClient ftpclient;
  WiFiClient ftpdclient;

  const byte Bufsize = 128;
  char outBuf[Bufsize];
  short FTPretcode = 0;
  const byte port = 21; //21 is the standard connection port

  File ftx = SD.open(fileName); //file to be transmitted
  if (!ftx)
  {
    if (debug_Ind)
    {
      Serial.println(F("file open failed"));
    }
    return 900;
  }
  if (ftpclient.connect(host, port))
  {
    if (debug_Ind)
    {
      Serial.println(F("Connected to FTP server"));
    }
  }
  else
  {
    ftx.close();
    if (debug_Ind)
    {
      Serial.println(F("Failed to connect to FTP server"));
    }
    return 910;
  }
  FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
  if (FTPretcode >= 400)
    return FTPretcode;

  /* User - Authentication username
   * Send this command to begin the login process. username should be a
   * valid username on the system, or "anonymous" to initiate an anonymous login.
   */
  ftpclient.print("USER ");
  ftpclient.println(uname);
  FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
  if (FTPretcode >= 400)
    return FTPretcode;

  /* PASS - Authentication password
   * After sending the USER command, send this command to complete
   * the login process. (Note, however, that an ACCT command may have to be
   * used on some systems, not needed with synology diskstation)
   */
  ftpclient.print("PASS ");
  ftpclient.println(pwd);
  FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
  if (FTPretcode >= 400)
```

```
  return FTPretcode;

// //CWD - Change the working folder on the FTP server
// if (strcmp(folder, "") != 0)
// {
//   ftpclient.print("CWD ");
//   ftpclient.println(folder);
//   FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
//   if (FTPretcode >= 400)
//   {
//     return FTPretcode;
//   }
// }

/* SYST - Returns a word identifying the system, the word "Type:",
 * and the default transfer type (as would be set by the
 * TYPE command). For example: UNIX Type: L8 - this is what
 * the diskstation returns
 */
ftpclient.println("SYST");
FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
if (FTPretcode >= 400)
  return FTPretcode;

/* TYPE - sets the transfer mode
 * A - ASCII text
 * E - EBCDIC text
 * I - image (binary data)
 * L - local format
 * for A & E, second char is:
 * N - Non-print (not destined for printing). This is the default if
 * second-type-character is omitted
 * Telnet format control (<CR>, <FF>, etc.)
 * C - ASA Carriage Control
 */
ftpclient.println("Type A");
FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
if (FTPretcode >= 400)
  return FTPretcode;

/* PASV - Enter passive mode
 * Tells the server to enter "passive mode". In passive mode, the server
 * will wait for the client to establish a connection with it rather than
 * attempting to connect to a client-specified port. The server will
 * respond with the address of the port it is listening on, with a message like:
 * 227 Entering Passive Mode (a1,a2,a3,a4,p1,p2), e.g. from diskstation
 * Entering Passive Mode (192,168,0,5,217,101)
 */
ftpclient.println("PASV");
FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
if (FTPretcode >= 400)
  return FTPretcode;
/* This is parsing the return from the server
 * where a1.a2.a3.a4 is the IP address and p1*256+p2 is the port number.
 */
char *tStr = strtok(outBuf, "(,"); //chop the output buffer into tokens based on the delimiters
int array_pasv[6];
for (int i = 0; i < 6; i++)
{                             //there are 6 elements in the address to decode
  tStr = strtok(NULL, "(,");  //1st time in loop 1st token, 2nd time 2nd token, etc.
  array_pasv[i] = atoi(tStr); //convert to int, why atoi - because it ignores any non-numeric chars
                              //after the number
  if (tStr == NULL)
  {
    if (debug_Ind)
    {
      Serial.println(F("Bad PASV Answer"));
```

```
    }
  }
}
//extract data port number
unsigned int hiPort, loPort;
hiPort = array_pasv[4] << 8;  //bit shift left by 8
loPort = array_pasv[5] & 255; //bitwise AND
if (debug_Ind)
{
  Serial.print(F("Data port: "));
}
hiPort = hiPort | loPort; //bitwise OR
if (debug_Ind)
{
  Serial.println(hiPort);
}
//first instance of dftp

if (ftpdclient.connect(host, hiPort))
{
  if (debug_Ind)
  {
    Serial.println(F("Data port connected"));
  }
}
else
{
  if (debug_Ind)
  {
    Serial.println(F("Data connection failed"));
  }
  ftpclient.stop();
  ftx.close();
}

/* STOR - Begin transmission of a file to the remote site. Must be preceded
 * by either a PORT command or a PASV command so the server knows where
 * to accept data from
 */
ftpclient.print("STOR ");
if (strcmp(folder, "") != 0)
{
  sprintf(filename33, "%s%s", xfolder, fileName);
  ftpclient.println(filename33);
}
else
{
  ftpclient.println(fileName);
}
FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
if (FTPretcode >= 400)
{
  ftpdclient.stop();
  return FTPretcode;
}
if (debug_Ind)
{
  Serial.println(F("Writing..."));
}

byte clientBuf[64];
int clientCount = 0;

while (ftx.available())
{
  clientBuf[clientCount] = ftx.read();
  clientCount++;
```

```
    if (clientCount > 63)
    {
      ftpdclient.write((const uint8_t *)clientBuf, 64);
      clientCount = 0;
    }
  }
  if (clientCount > 0)
    ftpdclient.write((const uint8_t *)clientBuf, clientCount);
  ftpdclient.stop();
  if (debug_Ind)
  {
    Serial.println(F("Data disconnected"));
  }
  FTPretcode = eRcv(ftpclient, outBuf, Bufsize);
  if (FTPretcode >= 400)
  {
    return FTPretcode;
  }

  //End the connection
  ftpclient.println("QUIT");
  ftpclient.stop();
  if (debug_Ind)
  {
    Serial.println(F("Disconnected from FTP server"));
  }

  ftx.close();
  if (debug_Ind)
  {
    Serial.println(F("File closed"));
  }
  return FTPretcode;
} // end function doFTP
#endif

#if (SPS030_)
/*-----------------------------------------------------------
 * Sensirion SPS30 Functions
 * Copyright (c) 2018, Sensirion AG
 * All rights reserved.
 * https://github.com/Sensirion/embedded-uart-sps
 -----------------------------------------------------------*/
int16_t sensirion_uart_select_port(uint8_t port)
{
  return 0;
}
int16_t sensirion_uart_open()
{
  P_SPS30.begin(115200);
  delay(10);
  return 0;
}
int16_t sensirion_uart_close()
{
  P_SPS30.end();
  return 0;
}
int16_t sensirion_uart_tx(uint16_t data_len, const uint8_t *data)
{
  return P_SPS30.write(data, data_len);
}
int16_t sensirion_uart_rx(uint16_t max_data_len, uint8_t *data)
{
  int16_t i = 0;
  while (P_SPS30.available() > 0 && i < max_data_len)
  {
```

```
    data[i] = (uint8_t)P_SPS30.read();
    i++;
  }
  return i;
}
void sensirion_sleep_usec(uint32_t useconds)
{
  delay((useconds / 1000) + 1);
}
#endif

#if (MQTT_)
void handleMessage(AdafruitIO_Data *data)
{

  // since we sent an int, we can use toInt()
  // to get the int value from the received IO
  // data.
  int received_value = data->toInt();

  // the lat() lon() and ele() methods
  // will allow you to get the double values
  // for the location data sent by IO
  double received_lat = data->lat();
  double received_lon = data->lon();
  double received_ele = data->ele();

  // print out the received values
  Serial.println(”----- received -----”);
  Serial.print(”value: ”);
  Serial.println(received_value);
  Serial.print(”lat: ”);
  Serial.println(received_lat, 6);
  Serial.print(”lon: ”);
  Serial.println(received_lon, 6);
  Serial.print(”ele: ”);
  Serial.println(received_ele, 2);
}
#endif

// /**
//  * Print SAMD chip serial number.
//  *
//  * http://atmel.force.com/support/articles/en_US/FAQ/Reading-unique-serial-number-on-SAM-D20-SAM-
D21-SAM-R21-devices
//  * https://gist.github.com/mgk/c9ec87436d2d679e5d08
//  */
// void printChipId()
// {
//   volatile uint32_t val1, val2, val3, val4;
//   volatile uint32_t *ptr1 = (volatile uint32_t *)0x0080A00C;
//   val1 = *ptr1;
//   volatile uint32_t *ptr = (volatile uint32_t *)0x0080A040;
//   val2 = *ptr;
//   ptr++;
//   val3 = *ptr;
//   ptr++;
//   val4 = *ptr;

//   Serial.print(”chip id: 0x”);
//   char buf[33];
//   sprintf(buf, ”%8x%8x%8x%8x”, val1, val2, val3, val4);
//   Serial.println(buf);
// }

void setup()
{
```

```
  delay(1000);
  // put your setup code here, to run once:

#if defined(ESP32)
#if (GPS_attach) && (SDS011_)
  SerialYY.begin(9600);
#endif
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)
#if (SDS011_)
  sds.begin();
  Serial.println(sds.queryFirmwareVersion().toString());
  Serial.println(sds.setQueryReportingMode().toString());
#endif
#if (SPS030_)
  Serial2.begin(9600);
#endif
#endif

  if (debug_Ind)
  {
    Serial.begin(BR_);
  }

  // Read Config File
  if (debug_Ind)
  {
    Serial.println(" ");
    Serial.print("Initializing SD card...");
  }
  if (!SD.begin(SD_PIN))
  {
    if (debug_Ind)
    {
      Serial.println("initialization failed!");
    }
    Error_Ind = 1;
    return;
  }
  if (debug_Ind)
  {
    Serial.println("initialization done.");
  }

  //printChipId();
  //delay(3000);

#if (microSD_)
  String buff;
  // open the file for reading:
  config_File = SD.open("/config_node.ini");

  if (!config_File)
  {
    if (debug_Ind)
    {
      Serial.println("The text file cannot be opened");
    }
    Error_Ind = 1;
    return;
  }
  int ii = 0;
  while (config_File.available())
  {
    buff = config_File.readStringUntil('=');  //buffer
    buff = config_File.readStringUntil('\n'); //buffer
```

```
    configg[ii] = buff.toFloat();
    ii = ii + 1;
  }
  config_File.close();

#if defined(ESP8266) || defined(ESP32)
  // Read Config WiFi
  if (int(configg[2]) >= 1)
  {
    File configFile2;
    // open the file for reading:
    configFile2 = SD.open("/config_wifi.ini");
    if (!configFile2)
    {
      if (debug_Ind)
      {
        Serial.println("The text file (WiFi) cannot be opened");
      }
      configg[2] = 0;
      Error_Ind = 1;
      return;
    }
    while (configFile2.available())
    {
      buff = configFile2.readStringUntil('"');       //buffer
      buff = configFile2.readStringUntil('"');       //buffer
      buff.toCharArray(ssid, buff.length() + 1);     //ssid
      buff = configFile2.readStringUntil('"');       //buffer
      buff = configFile2.readStringUntil('"');       //buffer
      buff.toCharArray(password, buff.length() + 1); //password
    }
    configFile2.close();
  }
#endif

  // Read Config FTP
  if (int(configg[2]) == 1)
  {
    File configFile3;
    // open the file for reading:
    configFile3 = SD.open("/config_ftp.ini");
    if (!configFile3)
    {
      if (debug_Ind)
      {
        Serial.print("The text file (FTP) cannot be opened");
      }
      configg[2] = 0;
      Error_Ind = 1;
      return;
    }
    while (configFile3.available())
    {
      buff = configFile3.readStringUntil('"');        //buffer
      buff = configFile3.readStringUntil('"');        //buffer
      buff.toCharArray(xhost, buff.length() + 1);     //host
      buff = configFile3.readStringUntil('"');        //buffer
      buff = configFile3.readStringUntil('"');        //buffer
      buff.toCharArray(xusername, buff.length() + 1); //username
      buff = configFile3.readStringUntil('"');        //buffer
      buff = configFile3.readStringUntil('"');        //buffer
      buff.toCharArray(xpassword, buff.length() + 1); //password
      sprintf(xfolder, "/NodeMP-%02d", int(configg[0]));
    }
    configFile3.close();
  }
```

```
#if defined(ARDUINO_SAMD_FEATHER_M0)
  Serial2.begin(9600);
  Serial3.begin(9600);
#endif

  if (GPS_attach)
  {
    Serial.println("GPS");
  }
  else
  {
    File configFile4;
    configFile4 = SD.open("/config_location.ini");

    if (!configFile4)
    {
      if (debug_Ind)
      {
        Serial.print("The text file (location) cannot be opened");
      }
      Error_Ind = 1;
      return;
    }
    while (configFile4.available())
    {
      buff = configFile4.readStringUntil('"');        //buffer
      buff = configFile4.readStringUntil('"');        //buffer
      buff.toCharArray(latitud_, buff.length() + 1);  //latitud
      buff = configFile4.readStringUntil('"');        //buffer
      buff = configFile4.readStringUntil('"');        //buffer
      buff.toCharArray(longitud_, buff.length() + 1); //longitud
    }
    configFile4.close();
    //the_latitud_ = atof(latitud_);
    //the_longitud_ = atof(longitud_);
  }

  // LED Indicador de funcionamiento
  pulse_time = 1000;

#else
  configg[0] = 99;                  // ID
  configg[1] = 60;                  // Sample
  configg[2] = 2;                   // Connection
  configg[3] = -4;                  // UTC Zone
  strcpy(ssid, "HUAWEI-B310-74F3"); // Copy in the first string
  strcpy(password, "B898D2B9FAL");  // Copy in the first string
  //strcpy(ssid, "kuki2");  // Copy in the first string
  //strcpy(password, "6l4a-19il-wgrd-");  // Copy in the first string
  strcpy(latitud_, "0");  // Copy in the first string
  strcpy(longitud_, "0"); // Copy in the first string
#endif

#if !(MQTT_)
#if defined(ESP8266) || defined(ESP32)
  // We start by connecting to a WiFi network
  if (int(configg[2]) == 1)
  {
    if (debug_Ind)
    {
      Serial.print("Connecting to ");
      Serial.println(ssid);
    }
    WiFi.begin(ssid, password);
    delay(10000);
    WiFi.setAutoReconnect(true);
    WiFi.setAutoConnect(true);
```

```
    if (WiFi.status() == WL_CONNECTED)
    {
      configg2_temp = 0;
      if (debug_Ind)
      {
        Serial.println("WiFi connected!");
        //timeClient.begin();
        Serial.println("Starting UDP");
      }
      udp.begin(localPort);
      WiFi.hostByName(ntpServerName, timeServerIP);
      sendNTPpacket(timeServerIP);
      delay(1000);
      pulse_time = 500;
    }
    else
    {
      configg2_temp = 1;
      if (debug_Ind)
      {
        Serial.println("WiFi NO connected!");
      }
    }
  }
#endif
#else

  if (int(configg[2]) == 2)
  {
    WiFi.mode(WIFI_STA); // set to station mode
    AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, ssid, password);
    // connect to io.adafruit.com
    Serial.println("Connecting to Adafruit IO");
    io.connect();
    delay(10000);

    if (io.status() >= AIO_CONNECTED)
    {
      // we are connected
      Serial.println(io.statusText());

      udp.begin(localPort);
      WiFi.hostByName(ntpServerName, timeServerIP);
      sendNTPpacket(timeServerIP);
      delay(1000);
    }
    else
    {
      //configg[2] = 0;
      Serial.println(io.statusText());
    }
  }

  // FEED NAMES
  sprintf(PM25FEED_, "%02d-PM25", (int)configg[0]);
  sprintf(TOTALFEED_, "%02d-TOT", (int)configg[0]);

#endif

  // Time Sync
  if (!rtc.begin())
  {
    if (debug_Ind)
    {
      Serial.println("Couldn't find RTC");
    }
```

```
    //Error_Ind = 1;
    //return;
  }

#if defined(ESP8266)
  if (!rtc.isrunning())
  {
    if (debug_Ind)
    {
      Serial.println(”RTC is NOT running!”);
      rtc.adjust(DateTime(__DATE__, __TIME__));
    }
  }
#endif

#if defined(ESP32)
  if (!rtc.initialized())
  {
    if (debug_Ind)
    {
      Serial.println(”RTC is NOT running!”);
    }
  }
#endif

#if defined(ESP8266) || defined(ESP32)
  if (int(configg[2]) >= 1 && configg2_temp == 0)
  {
    int cb = udp.parsePacket();
    if (!cb)
    {
      //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
      delay(1);
    }
    else
    {
      udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
      unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
      unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
      unsigned long secsSince1900 = highWord << 16 | lowWord;
      const unsigned long seventyYears = 2208988800UL;
      unsigned long epoch = secsSince1900 - seventyYears;
      //epoch = epoch + (configg[3] * 60 * 60); // UTC - 3 ( 3 * 60 * 60 ) // JST
      epoch = epoch + (uint(configg[3]) * 60 * 60); // UTC - 3 ( 3 * 60 * 60 ) // JST
      rtc.adjust(epoch);
      if (debug_Ind)
      {
        Serial.print(”UNX”);
        Serial.println(epoch);
      }
    }
  }
#endif
  delay(1);

#if defined(ARDUINO_SAMD_FEATHER_M0)

  // FONA BEGIN
  if (int(configg[2]) == 1)
  {
    pinMode(FONA_RST, OUTPUT);
    digitalWrite(FONA_RST, HIGH); // Default state

    fonaSerial->begin(4800);
    if (!fona.begin(*fonaSerial))
    {
      Serial.println(F(”Couldn’t find FONA”));
```

```
      configg[2] = 0;
    }
    fona.enableGPRS(false);
  }

  if (int(configg[2]) == 1)
  {
    // Turn on data connection

    fona.setGPRSNetworkSettings(F(apn_name), F(apn_username), F(apn_password));
    while (!fona.enableGPRS(true))
    {
      Serial.println(F("Failed to enable data, retrying..."));
      delay(2000);
    }
    Serial.println(F("Enabled data!"));
    delay(10000);

    // NTP RTC SYNC
    if (!fona.enableNTPTimeSync(true, F("pool.ntp.org")))
    {
      Serial.printf("Failed to enable\r\n");
    }

    char buffer[23];

    fona.getTime(buffer, 23); // make sure replybuffer is at least 23 bytes!
    Serial.printf(buffer);

    int id1, id2, id3;
    int id4, id5, id6;
    sscanf(buffer, "\"%d/%d/%d,%d:%d:%d\"", &id1, &id2, &id3, &id4, &id5, &id6);
    id1 = 19 + 2000 - 1900;

    // int EPOCC = tm_sec + tm_min * 60 + tm_hour * 3600 + tm_yday * 86400 +
    //             (tm_year - 70) * 31536000 + ((tm_year - 69) / 4) * 86400 -
    //             ((tm_year - 1) / 100) * 86400 + ((tm_year + 299) / 400) * 86400

    int id7 = 0;
    id7 = calculatedayOfYear(id3, id2, id1 + 2000);

    int EPOCC = id6 + id5 * 60 + id4 * 3600 + id7 * 86400 + (id1 - 70) * 31536000 +
                ((id1 - 69) / 4) * 86400 - ((id1 - 1) / 100) * 86400 + ((id1 + 299) / 400) * 86400;

    Serial.print("UNX");
    Serial.println(EPOCC);
    EPOCC = EPOCC - 10800 - 86400; // UTC - 3 ( 3 * 60 * 60 ) // JST
    rtc.adjust(EPOCC);
  }

#endif

  // BME280 begin
  bool status;
  if (gen_BME280)
  {
    status = bme.begin();
  }
  else
  {
    status = bme.begin(0x76);
  }
  delay(1000);
  if (debug_Ind)
  {
    Serial.println("BME Begin");
    BME_280 = true;
```

```
  }
  if (!status)
  {
    if (debug_Ind)
    {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      BME_280 = false;
    }
    //Error_Ind = 1;
    //return;
  }

// SDS011 begin
#if (SDS011_)
  if (debug_Ind)
  {
    Serial.println("SDS011 Begin");
  }
  delay(100);
//sds.begin();
#endif

// SPS030 begin
#if (SPS030_)

  if (debug_Ind)
  {
    Serial.println("SPS030 Begin");
  }
  delay(100);

  sensirion_uart_select_port(1);
  sensirion_uart_open();
  delay(5000);
  start99 = millis();
  while (sps30_probe() != 0)
  {
    if (debug_Ind)
    {
      Serial.write("probe failed\n");
    }
    delay(1000);
    if (millis() - start99 > 10000)
    {
      Error_Ind = 1;
      return;
    }
  }
  ret = sps30_set_fan_auto_cleaning_interval_days(1);
  if (ret)
  {
    if (debug_Ind)
    {
      Serial.print("error setting the auto-clean interval: ");
      Serial.println(ret);
    }
  }

  if (sps30_start_measurement() != 0)
  {
    if (debug_Ind)
    {
      Serial.println("error starting measurement");
    }
    Error_Ind = 1;
    return;
  }
```

```
  delay(10000);
#endif

#if (GPS_attach)

  //while (!Serial);  // uncomment to have the sketch wait until Serial is ready

  // connect at 115200 so we can read the GPS fast enough and echo without dropping chars
  // also spit it out
  //Serial.begin(115200);
  Serial.println("Adafruit GPS library basic test!");

  // 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
  GPS.begin(9600);
  // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitude
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  // uncomment this line to turn on only the "minimum recommended" data
  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  // For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
  // the parser doesn't care about other sentences at this time
  // Set the update rate
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
  // For the parsing code to work nicely and have time to sort thru the data, and
  // print it out we don't suggest using anything higher than 1 Hz

  // Request updates on antenna status, comment out to keep quiet
  GPS.sendCommand(PGCMD_ANTENNA);

  delay(1000);

  // Ask for firmware version
  GPSSerial.println(PMTK_Q_RELEASE);
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)
#if (ADS1115)
  //ads.setGain(GAIN_TWOTHIRDS); // +/- 6.144V  1 bit = 0.1875mV (default)
  // ads.setGain(GAIN_ONE);       // +/- 4.096V  1 bit = 0.125mV
  // ads.setGain(GAIN_TWO);       // +/- 2.048V  1 bit = 0.0625mV
  ads.setGain(GAIN_FOUR); // +/- 1.024V  1 bit = 0.03125mV
  // ads.setGain(GAIN_EIGHT);     // +/- 0.512V  1 bit = 0.015625mV
  // ads.setGain(GAIN_SIXTEEN);   // +/- 0.256V  1 bit = 0.0078125mV
  ads.begin();
#endif
#endif

#if (Serial_Out)
  // SerialX
  SerialX.begin(9600);
#endif

  // End Setup
  if (debug_Ind)
  {
    Serial.println("End Setup");
  }
  FirstStart = 1;
  FTPresult = 0;
}

#if (MQTT_)
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, ssid, password);
AdafruitIO_Feed *PM25Feed;
AdafruitIO_Feed *DATAFeed;
#endif

/*
```

```
  VOID LOOP
 */
void loop()
{
  // put your main code here, to run repeatedly:

  Serial.print("Waiting...");

  //wait time
  time_wait = configg[1] * 1000 - (millis() - start99);
  while (time_wait >= tw_)
  {

#if (GPS_attach)

    // read data from the GPS in the 'main loop'
    char c = GPS.read();
    // if you want to debug, this is a good time to do it!
    if (GPSECHO)
      if (c)
        Serial.print(c);
    // if a sentence is received, we can check the checksum, parse it...
    if (GPS.newNMEAreceived())
    {
      // a tricky thing here is if we print the NMEA sentence, or data
      // we end up not listening and catching other sentences!
      // so be very wary if using OUTPUT_ALLDATA and trying to print out data
      //Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false
      if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to false
        return;                       // we can fail to parse a sentence in which case we should just wait for another
    }
#endif

    delay(500);
    time_wait = configg[1] * 1000 - (millis() - start99);
    Serial.print(".");
  }

  // if (tw_ > 0)
  // {
  //   Serial.print(" ");
  //   SS_ = 1;
  //   while (SS_ != 0)
  //   {
  //     delay(500);
  //     now = rtc.now();
  //     SS_ = now.second();
  //     Serial.print(".");
  //   }
  // }
  // else
  // {
  DateTime now_ = rtc.now();
  // }
  Serial.println(" OK!");

  // Execution Time
  start99 = millis();

  if (debug_Ind)
  {
    Serial.print("Error indicator: ");
    Serial.println(Error_Ind);
  }

  while (Error_Ind == 1)
  {
```

```
#if defined(ESP8266)
    pinMode(LED_PORT, OUTPUT);
    digitalWrite(LED_PORT, LOW); // Turn the LED on by making the voltage LOW
    delay(30000);
    ESP.deepSleep(ESP.deepSleepMax());
#endif

#if defined(ESP32)
    pinMode(LED_PORT, OUTPUT);
    digitalWrite(LED_PORT, HIGH); // Turn the LED on by making the voltage LOW
    delay(30000);
    esp_deep_sleep_start();
#endif
  }

  // Realizando medicion con timeout

  Ind_medicion = 0;
  while (Ind_medicion == 0)
  {

    dtostrf(0, 7, 3, temp);
    dtostrf(0, 7, 3, temp_d);
    dtostrf(0, 8, 3, pres);
    dtostrf(0, 7, 3, pres_d);
    dtostrf(0, 7, 3, hume);
    dtostrf(0, 7, 3, hume_d);
    dtostrf(0, 7, 2, PM10_);
    dtostrf(0, 7, 2, PM10_d);
    dtostrf(0, 7, 2, PM25_);
    dtostrf(0, 7, 2, PM25_d);
    dtostrf(0, 7, 4, ADC_0);
    dtostrf(0, 7, 4, ADC_0_DEV);
    dtostrf(0, 7, 4, ADC_1);
    dtostrf(0, 7, 4, ADC_1_DEV);
    dtostrf(0, 7, 4, ADC_2);
    dtostrf(0, 7, 4, ADC_2_DEV);
    dtostrf(0, 7, 4, ADC_3);
    dtostrf(0, 7, 4, ADC_3_DEV);

    if (BME_280)
    {
      // Meteo Data
      AvgStd temp_r, pres_r, hume_r;
      for (int jj = 0; jj < 100; jj++)
      {
        temp_r.checkAndAddReading(bme.readTemperature());
        pres_r.checkAndAddReading(bme.readPressure() / 100.0F);
        hume_r.checkAndAddReading(bme.readHumidity());
        delay(10);
      }

      if (temp_r.getN() > 0)
      {
        dtostrf(temp_r.getMean(), 7, 3, temp);
        dtostrf(temp_r.getStd(), 7, 3, temp_d);
        dtostrf(pres_r.getMean(), 8, 3, pres);
        dtostrf(pres_r.getStd(), 8, 3, pres_d);
        dtostrf(hume_r.getMean(), 7, 3, hume);
        dtostrf(hume_r.getStd(), 7, 3, hume_d);
      }
    }

// SDS011
#if (SDS011_)
```

```
    AvgStd PM10RAW, PM25RAW;
    for (int jj = 0; jj < 20; jj++)
    {
      PmResult pm = sds.queryPm();
      if (pm.isOk())
      {
        PM10RAW.checkAndAddReading(pm.pm10);
        PM25RAW.checkAndAddReading(pm.pm25);
        delay(100);
      }
    }

    if (PM10RAW.getN() > 0)
    {
      dtostrf(PM10RAW.getMean(), 7, 2, PM10_);
      dtostrf(PM10RAW.getStd(), 7, 2, PM10_d);
      dtostrf(PM25RAW.getMean(), 7, 2, PM25_);
      dtostrf(PM25RAW.getStd(), 7, 2, PM25_d);
    }

#endif

//SPS030
#if (SPS030_)

    AvgStd PM10RAW, PM25RAW;
    for (int jj = 0; jj < 10; jj++)
    {
      ret = sps30_read_measurement(&measurement);
      delay(1000);
      if (ret < 0)
      {
        if (debug_Ind)
        {
          Serial.println("error reading measurement");
        }
      }
      else
      {
        PM10RAW.checkAndAddReading(measurement.mc_10p0);
        PM25RAW.checkAndAddReading(measurement.mc_2p5);
      }
    }
    dtostrf(PM10RAW.getMean(), 7, 2, PM10_);
    dtostrf(PM10RAW.getStd(), 7, 2, PM10_d);
    dtostrf(PM25RAW.getMean(), 7, 2, PM25_);
    dtostrf(PM25RAW.getStd(), 7, 2, PM25_d);

    if (PM10RAW.getN() < 5)
    {
      Error_Ind = 1;
    }

#endif

// GPS DATA
#if (GPS_attach)

    if (GPS.fix)
    {
      dtostrf(GPS.latitudeDegrees, 6, 4, latitud_);
      dtostrf(GPS.longitudeDegrees, 6, 4, longitud_);
    }
    else
    {
      dtostrf(0, 6, 4, latitud_);
      dtostrf(0, 6, 4, longitud_);
```

```
    }
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)
// ADS1115 Data
#if (ADS1115)

    int16_t adc0, adc1, adc2, adc3;
    AvgStd V0, V1, V2, V3;
    ;

    for (int j = 0; j < 100; j++)
    {
      adc0 = ads.readADC_SingleEnded(0);
      delay(10);
      adc1 = ads.readADC_SingleEnded(1);
      delay(10);
      adc2 = ads.readADC_SingleEnded(2);
      delay(10);
      adc3 = ads.readADC_SingleEnded(3);
      delay(10);
      V0.checkAndAddReading(adc0 * 0.03125 / 1000);
      V1.checkAndAddReading(adc1 * 0.03125 / 1000);
      V2.checkAndAddReading(adc2 * 0.03125 / 1000);
      V3.checkAndAddReading(adc3 * 0.03125 / 1000);
    }

    Serial.print("V0 avg: " + String(V0.getMean(), 4));
    Serial.println(" +- " + String(V0.getStd(), 4));
    Serial.print("V1 avg: " + String(V1.getMean(), 4));
    Serial.println(" +- " + String(V1.getStd(), 4));
    Serial.print("V2 avg: " + String(V2.getMean(), 4));
    Serial.println(" +- " + String(V2.getStd(), 4));
    Serial.print("V3 avg: " + String(V3.getMean(), 4));
    Serial.println(" +- " + String(V3.getStd(), 4));

    dtostrf(V0.getMean(), 7, 4, ADC_0);
    dtostrf(V0.getStd(), 7, 4, ADC_0_DEV);
    dtostrf(V1.getMean(), 7, 4, ADC_1);
    dtostrf(V1.getStd(), 7, 4, ADC_1_DEV);
    dtostrf(V2.getMean(), 7, 4, ADC_2);
    dtostrf(V2.getStd(), 7, 4, ADC_2_DEV);
    dtostrf(V3.getMean(), 7, 4, ADC_3);
    dtostrf(V3.getStd(), 7, 4, ADC_3_DEV);

#endif
#endif

    Ind_medicion = 1;
  }

  if (debug_Ind)
  {
    Serial.println(" ");
  }
  sprintf(medicion, "%2d,%2d,%2d,%8s,%8s,%7s,%7s,%8s,%8s,%7s,%7s,%7s,%7s,
   %7s,%7s,%7s,%7s,%7s,%7s,%7s,%7s,%7s,%7s",
          now_.hour(), now_.minute(), now_.second(), latitud_, longitud_, temp,
          temp_d, pres, pres_d, hume, hume_d, PM10_, PM10_d, PM25_, PM25_d, ADC_0,
          ADC_0_DEV, ADC_1, ADC_1_DEV, ADC_2, ADC_2_DEV, ADC_3, ADC_3_DEV);
 PM10_d, PM25_, PM25_d, ADC_0, ADC_0_DEV, ADC_1, ADC_1_DEV, ADC_2, ADC_2_DEV, ADC_3, ADC_3_DEV);
  sprintf(medicion_S, "%02d/%02d/%02d %02d:%02d,%7s,%8s,%7s,%7s,%7s",
          now_.day(), now_.month(), now_.year() % 100, now_.hour(), now_.minute(), temp, pres, hume, PM10_, PM25_);

  if (debug_Ind)
  {
    Serial.println(VM2);
```

```
    Serial.println(medicion_S);
  }

#if (MQTT_)
  if (FirstStart == 1)
  {
    feed_aux = 1;
  }
  Serial.print("Uploading...");
  io.run();
  Serial.print(". ");
  if (io.status() >= AIO_CONNECTED)
  {
    if (feed_aux == 1)
    {
      PM25Feed = io.feed(PM25FEED_);
      DATAFeed = io.feed(TOTALFEED_);
      feed_aux = 0;
    }
    PM25Feed->save(PM25_);
    DATAFeed->save(medicion_S);
    Serial.println("OK!");
  }
  else
  {
    Serial.println("Connection Error");
  }

#endif

#if (Serial_Out)
  // Data to SerialX
  SerialX.write(medicion_S);
  Serial.println(medicion_S);
  delay(100);
#endif

  sprintf(filename, "/G%02d-%04d%02d%02d.csv", int(configg[0]), now_.year(), now_.month(), now_.day());
  sprintf(filename2, "/G%02d-%04d%02d%02d.bkp", int(configg[0]), now_.year(), now_.month(), now_.day());
  if (debug_Ind)
  {
    Serial.println(filename);
  }
  if (now_.year() == 2040)
  {
    if (debug_Ind)
    {
      Serial.println("RTC failed!");
    }
    // Error_Ind = 1;
    // return;
  }
  // Savedata to SD
  if (!SD.begin(SD_PIN))
  {
    if (debug_Ind)
    {
      Serial.println("SD failed!");
    }
    // Error_Ind = 1;
    // return;
  }
  otherDay = 0;
  if (!SD.exists(filename))
  {
    otherDay = 1;
  }
```

```
  // #if defined(ARDUINO_SAMD_FEATHER_M0)
  //   SdFile::dateTimeCallback(dateTime);
  // #endif

#if defined(ESP8266) || defined(ARDUINO_SAMD_FEATHER_M0)
  myFile = SD.open(filename, FILE_WRITE);
#endif

#if defined(ESP32)
  myFile = SD.open(filename, FILE_APPEND);
#endif

  if (otherDay == 1)
  {
    myFile.println(VM2);
  }
  //myFile.println(medicion);
  myFile.println(medicion_S);
  myFile.close();

#if defined(ESP8266) || defined(ARDUINO_SAMD_FEATHER_M0)
  myFile2 = SD.open(filename2, FILE_WRITE);
#endif

#if defined(ESP32)
  myFile2 = SD.open(filename2, FILE_APPEND);
#endif

  if (otherDay == 1)
  {
    myFile2.println(VM);
  }
  //myFile.println(medicion);
  myFile2.println(medicion);
  myFile2.close();

#if defined(ESP8266) || defined(ESP32)

#if !(MQTT_)
  // WiFi reconnect (3 times)
  if (int(configg[2]) == 1)
  {
    if (otherDay == 1 && FirstStart == 0 && configg2_temp >= 1)
    {
      configg2_temp = 30;
    }
    if (configg2_temp == 30 || configg2_temp == 60 || configg2_temp == 90)
    {
      if (debug_Ind)
      {
        Serial.print("Connecting to ");
        Serial.println(ssid);
      }
      WiFi.begin(ssid, password);
      delay(10000);
      WiFi.setAutoReconnect(true);
      WiFi.setAutoConnect(true);
      if (WiFi.status() == WL_CONNECTED)
      {
        configg2_temp = 0;
        if (debug_Ind)
        {
          Serial.println("WiFi connected!");
        }
        delay(1000);
      }
```

```
      else
      {
        if (debug_Ind)
        {
          Serial.println("WiFi NO connected!");
        }
        configg2_temp = configg2_temp + 1;
      }
    }
  }

  // Attempt FTP upload
  if (int(configg[2]) == 1 && configg2_temp == 0)
  {
    // if otherDay == 1 upload 1 day after
    if (otherDay == 1 && FirstStart == 0)
    {
      strcpy(filename, filename_old);
    }
    FTPresult = doFTP(xhost, xusername, xpassword, filename, xfolder);
    if (debug_Ind)
    {
      Serial.println("A return code of 226 is success");
      Serial.print("Return code = ");
      Serial.println(FTPresult);
    }
  }

  // WiFi Reconnect Parameter
  if (int(configg[2]) == 1 && configg2_temp >= 1)
  {
    configg2_temp = configg2_temp + 1;
  }
  if (int(configg[2]) == 1 && FTPresult == 910 && configg2_temp == 0)
  {
    configg2_temp = 1;
  }
#endif

  // Reset ESP
  if (otherDay == 1 && FirstStart != 1)
  {
    if (debug_Ind)
    {
      Serial.println("Reset..");
    }
    ESP.restart();
    ;
  }

  // LED Indicador de funcionamiento
  if (FirstStart == 1 && Error_Ind == 0)
  {
#if !(MQTT_)
    if (FTPresult == 226)
    {
      pulse_time = 50;
    }
#else
    if (io.status() >= AIO_CONNECTED)
    {
      pulse_time = 50;
    }
    else
    {
      pulse_time = 1000;
    }
```

```
#endif
    start98 = millis();
    pinMode(LED_PORT, OUTPUT);
    while (millis() - start98 <= 4000)
    {
      delay(pulse_time);           // Wait
      digitalWrite(LED_PORT, LOW); // Turn the LED on by making the voltage LOW
      delay(pulse_time);
      digitalWrite(LED_PORT, HIGH); // Turn the LED off by making the voltage HIGH
    }
    FirstStart = 0;
  }
  if (LED_PORT == 13)
  {
    digitalWrite(LED_PORT, LOW); // Turn the LED off on ESP32
  }

  // Save old date
  //strcpy(filename_old, filename);
#endif

#if defined(ARDUINO_SAMD_FEATHER_M0)

  if (int(configg[2]) == 1)
  {
    //Attemp FTP Upload
    Serial.println(F("Connecting to FTP server..."));
    if (!fona.FTP_Connect(xhost, 21, xusername, xpassword, xmode))
    {
      Serial.println(F("Failed to connect to FTP server!"));
    }
    else
    {
      if (otherDay == 1)
      {
        if (!fona.FTP_PUT(filename, xfolder, VM_ln, strlen(VM_ln)))
        { // File name, file path, content, content length
          Serial.println(F("Failed to upload!"));
        }
      }

      if (!fona.FTP_PUT(filename, xfolder, medicion_ln, strlen(medicion_ln)))
      { // File name, file path, content, content length
        Serial.println(F("Failed to upload!"));
      }
    }

    if (!fona.FTP_Quit())
    {
      Serial.println(F("Failed to close FTP connection!"));
    }

    delay(2000);

    Serial.println(F("Connecting to FTP server..."));
    if (!fona.FTP_Connect(xhost, 21, xusername, xpassword, xmode2))
    {
      Serial.println(F("Failed to connect to FTP server!"));
    }

    if (!fona.FTP_PUT("/now.json", xfolder, JSONMessage, strlen(JSONMessage)))
    { // File name, file path, content, content length
      Serial.println(F("Failed to upload!"));
    }

    if (!fona.FTP_Quit())
```

```
    {
      Serial.println(F("Failed to close FTP connection!"));
    }

    // Reset ESP
    if (otherDay == 1 && FirstStart != 1)
    {
      if (debug_Ind)
      {
        Serial.println("Reset..");
      }
      NVIC_SystemReset(); // processor software reset
      ;
    }
  }

  //Reset m0
  if (otherDay == 1 && FirstStart != 1)
  {
    if (debug_Ind)
    {
      Serial.println("Reset..");
    }
    NVIC_SystemReset(); // processor software reset
    ;
  }

#endif
  //}
}
```

ESPECIFICACIONES TÉCNICAS DE LOS SENSORES SELECCIONADOS

3 Sensor de MP: Sensirion SPS030

Parameter	Conditions		Value	Units
Mass concentration range	-		0 to 1'000	µg/m³
Mass concentration size range	PM1.0		0.3 to 1.0	µm
	PM2.5		0.3 to 2.5	µm
	PM4		0.3 to 4.0	µm
	PM10		0.3 to 10.0	µm
Mass concentration precision[1,2] for PM1 and PM2.5[3]	0 to 100 µg/m³		±10	µg/m³
	100 to 1000 µg/m³		±10	% m.v.
Mass concentration precision[1,2] for PM4, PM10[4]	0 to 100 µg/m³		±25	µg/m³
	100 to 1000 µg/m³		±25	% m.v.
Maximum long-term mass concentration precision limit drift	0 to 100 µg/m³		±1.25	µg/m³ / year
	100 to 1000 µg/m³		±1.25	% m.v. / year
Number concentration range	-		0 to 3'000	#/cm³
Number concentration size range	PM0.5		0.3 to 0.5	µm
	PM1.0		0.3 to 1.0	µm
	PM2.5		0.3 to 2.5	µm
	PM4		0.3 to 4.0	µm
	PM10		0.3 to 10.0	µm
Number concentration precision[1,2] for PM0.5, PM1 and PM2.5[3]	0 to 1000 #/cm³		±100	#/cm³
	1000 to 3000 #/cm³		±10	% m.v.
Number concentration precision[1,2] for PM4, PM10[4]	0 to 1000 #/cm³		±250	#/cm³
	1000 to 3000 #/cm³		±25	% m.v.
Maximum long-term number concentration precision limit drift[2]	0 to 1000 #/cm³		±12.5	#/cm³ / year
	1000 to 3000 #/cm³		±1.25	% m.v. / year
Sampling interval	-		1±0.04	s
Typical start-up time[5]	number concentration	200 – 3000 #/cm³	8	s
		100 – 200 #/cm³	16	s
		50 – 100 #/cm³	30	s
Sensor output characteristics	PM2.5 mass concentration		Calibrated to TSI DustTrak™ DRX 8533 Ambient Mode	
	PM2.5 number concentration		Calibrated to TSI OPS 3330	
Lifetime[6]	24 h/day operation		> 10	years
Acoustic emission level	0.2 m	max.	25	dB(A)
Long term acoustic emission level drift	0.2 m	max.	+0.5	dB(A) / year
Additional T-dependent mass and number concentration precision limit drift[2]	temperature difference to 25°C	typ.	±0.5	% m.v. / °C
Weight	-		26.3 ±0.3	g

[1] Also referred to as "between-parts variation" or "device-to-device variation".

[2] For further details, please refer to the document "Sensirion Particulate Matter Sensor Specification Statement".

[3] Verification Aerosol for PM2.5 is a 3% atomized KCl solution. Deviation to reference instrument is verified in end-tests for every sensor after calibration.

[4] PM4 and PM10 output values are calculated based on distribution profile of all measured particles.

[5] Time after starting Measurement-Mode, until a stable measurement is obtained.

[6] Lifetime is based on mean-time-to-failure (MTTF) calculation. Lifetime might vary depending on different operating conditions.

4 Sensor de MP: Novasensor SDS011

No	Item	Parameter	Note
1	Measurement parameters	PM2.5,PM10	
2	Range	0.0-999.9 μg /m^3	
3	Rated voltage	5V	
4	Rated current	70mA±10mA	
5	Sleep current	<4 mA	Lase&Fan sleep
6	Temperature range	Storage environment：-20 ~ +60℃	
		Work environment：-10 ~ +50℃	
7	Humidity range	Storage environment：Max 90%	
		Work environment：Max 70%	
8	Air pressure	86KPa~110KPa	
9	Corresponding time	1s	
10	Serial data output frequency	1Hz	
11	Minimum resolution of particle	0.3 μm	
12	Counting yield	70%@0.3μm 98%@0.5μm	
13	Relative error	Maximum of ± 15% and ±10μg/m^3	25℃, 50%RH
14	Product size	71x70x23mm	
15	Certification	CE/FCC/RoHS	

5 Sensor de NO_2: Alphasense A43F

PERFORMANCE						
	Sensitivity	nA/ppm at 2ppm NO_2				-175 to -450
	Response time	t_{90} (s) from zero to 2ppm NO_2				< 60
	Zero current	nA in zero air at 20°C				-50 to +70
	Noise*	±2 standard deviations (ppb equivalent)				15
	Range	ppm NO_2 limit of performance warranty				20
	Linearity	ppm error at full scale, linear at zero and 20ppm NO_2				< ±0.5
	Overgas limit	maximum ppm for stable response to gas pulse				50
	*** Tested with Alphasense AFE low noise circuit**					
LIFETIME	Zero drift	ppb equivalent change/year in lab air				0 to 20
	Sensitivity drift	% change/year in lab air, monthly test				< -20 to -40
	Operating life	months until 50% original signal (24 month warranted)				> 24
ENVIRONMENTAL						
	Sensitivity @ -20°C	(% output @ -20°C/output @ 20°C) @ 2ppm NO_2				40 to 80
	Sensitivity @ 40°C	(% output @ 50°C/output @ 20°C) @ 2ppm NO_2				95 to 115
	Zero @ -20°C	nA				0 to +25
	Zero @ 40°C	nA				20 to 60
CROSS SENSITIVITY	O_3	Filter capacity (ppm hrs)	@	2ppm	O_3	> 500
	H_2S	sensitivity % measured gas	@	5ppm	H_2S	< -80
	NO	sensitivity % measured gas	@	5ppm	NO	< 5
	Cl_2	sensitivity % measured gas	@	5ppm	Cl_2	< 75
	SO_2	sensitivity % measured gas	@	5ppm	SO_2	< -5
	CO	sensitivity % measured gas	@	5ppm	CO	< -5
	C_2H_4	sensitivity % measured gas	@	100ppm	C_2H_4	< 1
	NH_3	sensitivity % measured gas	@	20ppm	NH_3	< 0.2
	H_2	sensitivity % measured gas	@	100ppm	H_2	< 0.1
	CO_2	sensitivity % measured gas	@	5% Vol	CO_2	0.1
	Halothane	sensitivity % measured gas	@	100ppm	Halothane	nd
KEY SPECIFICATIONS						
	Temperature range	°C				-30 to 40
	Pressure range	kPa				80 to 120
	Humidity range	% rh continuous				15 to 85
	Storage period	months @ 3 to 20°C (stored in sealed pot)				6
	Load resistor	Ω (AFE circuit recommended)				33 to 100
	Weight	g				< 6

6 Sensor de O_3 + NO_2: Alphasense OX

Specification O_3 Sensing

PERFORMANCE						
	Sensitivity	nA/ppm at 1ppm O_3				-200 to -650
	Response time	t_{90} (s) from zero to 1ppm O_3				< 45
	Zero current	nA in zero air at 20°C				-50 to 70
	Noise*	±2 standard deviations (ppb equivalent)				15
	Range	ppm O_3 limit of performance warranty				20
	Linearity	ppm error at full scale, linear at zero and 20ppm O_3				< ±0.5
	Overgas limit	maximum ppm for stable response to gas pulse				50
	*** Tested with Alphasense AFE low noise circuit**					
LIFETIME	Zero drift	ppb equivalent change/year in lab air				0 to 20
	Sensitivity drift	% change/year in lab air, monthly test				< -20 to -40
	Operating life	months until 50% original signal (24 month warranted)				> 24
ENVIRONMENTAL						
	Sensitivity @ -20°C	(% output @ -20°C/output @ 20°C) @ 2ppm O_3				60 to 80
	Sensitivity @ 40°C	(% output @ 40°C/output @ 20°C) @ 2ppm O_3				80 to 105
	Zero @ -20°C	nA				0 to 25
	Zero @ 40°C	nA				20 to 90
CROSS SENSITIVITY	H_2S	sensitivity % measured gas	@	5ppm	H_2S	< 100
	NO	sensitivity % measured gas	@	5ppm	NO	< 5
	Cl_2	sensitivity % measured gas	@	5ppm	Cl_2	< 85
	SO_2	sensitivity % measured gas	@	5ppm	SO_2	< -6
	CO	sensitivity % measured gas	@	5ppm	CO	< 0.1
	C_2H_4	sensitivity % measured gas	@	100ppm	C_2H_4	< 0.1
	NH_3	sensitivity % measured gas	@	20ppm	NH_3	< 0.1
	H_2	sensitivity % measured gas	@	100ppm	H_2	< 0.1
	CO_2	sensitivity % measured gas	@	5% Vol	CO_2	0.1
	Halothane	sensitivity % measured gas	@	100ppm	Halothane	< 0.1
KEY SPECIFICATIONS						
	Temperature range	°C				-30 to 40
	Pressure range	kPa				80 to 120
	Humidity range	% rh continuous				15 to 85
	Storage period	months @ 3 to 20°C (stored in sealed pot)				6
	Load resistor	Ω (AFE circuit recommended)				33 to 100
	Weight	g				< 6

Specification NO_2 Sensing

PERFORMANCE						
	Sensitivity to NO_2	nA/ppm at 2ppm NO_2				-200 to -550
	Response time	t_{90} (s) from zero to 1ppm NO_2				< 45
	Zero current	nA in zero air at 20°C				-50 to 70
	Noise*	±2 standard deviations (ppb equivalent)				15
	Range	ppm NO_2 limit of performance warranty				20
	Linearity	ppm error at full scale, linear at zero and 20ppm NO_2				< ±0.5
	Overgas limit	maximum ppm for stable response to gas pulse				50
	* **Tested with Alphasense AFE low noise circuit**					
LIFETIME	Zero drift	ppb equivalent change/year in lab air				0 to 20
	Sensitivity drift	% change/year in lab air, monthly test				< -20 to -40
	Operating life	months until 50% original signal (24 month warranted)				> 24
ENVIRONMENTAL						
	Sensitivity @ -20°C	(% output @ -20°C/output @ 20°C) @ 2ppm NO_2				50 to 80
	Sensitivity @ 40°C	(% output @ 50°C/output @ 20°C) @ 2ppm NO_2				115 to 130
	Zero @ -20°C	nA				0 to 25
	Zero @ 40°C	nA				20 to 50
CROSS SENSITIVITY	H_2S	sensitivity % measured gas	@	5ppm	H_2S	< 100
	NO	sensitivity % measured gas	@	5ppm	NO	< 5
	Cl_2	sensitivity % measured gas	@	5ppm	Cl_2	< 85
	SO_2	sensitivity % measured gas	@	5ppm	SO_2	< -6
	CO	sensitivity % measured gas	@	5ppm	CO	< 0.1
	C_2H_4	sensitivity % measured gas	@	100ppm	C_2H_4	< 0.1
	NH_3	sensitivity % measured gas	@	20ppm	NH_3	< 0.1
	H_2	sensitivity % measured gas	@	100ppm	H_2	< 0.1
	CO_2	sensitivity % measured gas	@	5% Vol	CO_2	0.1
	Halothane	sensitivity % measured gas	@	100ppm	Halothane	< 0.1
KEY SPECIFICATIONS						
	Temperature range	°C				-30 to 40
	Pressure range	kPa				80 to 120
	Humidity range	% rh continuous				15 to 85