


**Universidad Distrital Francisco José de Caldas**
**Facultad de Ingeniería**

# IMPLEMENTACIÓN RED DE SENSORES INALÁMBRICOS DE BAJO COSTO UTILIZANDO PROTOCOLO LORAWAN APLICADA AL MONITOREO DE MATERIAL PARTICULADO, MONÓXIDO DE CARBONO Y DIÓXIDO DE NITRÓGENO COMO INDICADORES DE CALIDAD DE AIRE EN EXTERIORES

**Presentado por:**
Ivette Lorena Nossa Ruiz
Rudy Marcela Arias Castañeda

**Bajo la Dirección de:**
Leonardo Plazas Nossa
Grupo GITEM++

Bogotá, Colombia
Agosto 2020

# Agradecimientos

En primer lugar agradecemos a Dios la oportunidad de haber realizado esta carrera haciendo parte de la gran familia de la Universidad Distrital que nos hizo crecer personal y profesionalmente, agradecemos a nuestras familias que son nuestra motivación para salir adelante y que siempre han estado allí apoyándonos, y a todas las personas que de una u otra manera hicieron parte de este proceso y fueron esa fuerza que nos empujó a seguir y no rendirnos, nuestros caminos siempre serán inciertos pero el universo se encargara de guiarnos, fueron grandes experiencias las que adquirimos durante este proceso de formación en el que nuestros compañeros, profesores y familiares aportaron conocimientos y consejos.

Agradecemos a cada uno de nuestros profesores que nos transmitieron más que teoría su forma de ver las cosas y así nos permitieron crecer integralmente, agradecemos a nuestros compañeros porque gracias a ellos cada semestre fue una aventura y de forma especial agradecemos al profesor Leonardo Plazas quien guió el desarrollo del proyecto y estuvo siempre dispuesto a darnos comentarios constructivos, que llevaron a un buen resultado.

# Resumen

En este proyecto se implementó una red de sensores inalámbricos haciendo uso de la tecnología LORA y el protocolo de comunicaciones LORAWAN, que permite determinar la calidad del aire en un determinado lugar, midiendo las concentraciones de algunos indicadores de calidad del aire expuestos en la norma colombiana como lo son el monóxido de carbono, material particulado y dióxido de nitrógeno. Los contaminantes anteriormente nombrados, representan un gran riesgo en la salud de las personas dependiendo de su concentración en el ambiente, por lo que la red le permitirá al usuario conocer el estado de ambiente y tomar las medidas necesarias para disminuir dichas concentraciones y velar por el bienestar de las personas.

La red está conformada por dos prototipos o nodos finales compuestos por un microcontrolador como eje central, sensores, un módulo de comunicación LORAWAN y un sistema de alimentación que incluye baterías recargables. También se tenía pensado en colocar un sistema de almacenamiento externo, pero durante el desarrollo del proyecto se explicarán las limitaciones que imposibilitaron la implementación del mismo.

El segundo componente de la red es el gateway que a pesar de que tiene limitaciones con respecto al protocolo LORAWAN permitió la implementación correcta de la red, pudiéndose comunicar con los nodos finales. La información que es recibida por el gateway es enviada al siguiente componente de la red, como lo es el servidor de red, en este caso The Things Network (TTN), que permite el registro de gateways para conformar una red pública y gratuita. El servidor de red se encarga de decodificar los datos para finalmente integrarse con un servidor de aplicación, en este caso ThingSpeak, que permite visualizar los datos de forma ordenada y además crear gráficos a partir de la información, para que el usuario final pueda interpretar de manera más sencilla cuál fue la calidad del aire.

Como resultado el usuario final tiene acceso a dos páginas web donde se encuentra la información de cada nodo, además de la generación de alarmas visuales y de correo electrónico que le indican cuándo debe tomar medidas sobre la situación del ambiente.

Para comprobar y evaluar la red se realizaron pruebas con diferentes contaminantes, prueba de cobertura y pruebas para determinar la pérdida de mensajes en la red.

# Índice general

**Lista de Figuras** VI

**Lista de Tablas** X

**1. Planteamiento del problema** 2

**2. Justificación** 4

2.1. Investigativa . . . 4
2.2. Académica . . . 4
2.3. Social . . . 4
2.4. Económica . . . 5
2.5. Ambiental . . . 5
2.6. Personal . . . 6

**3. Objetivos** 7

3.1. Objetivo general . . . 7
3.2. Objetivos específicos . . . 7

**4. Diagrama del proyecto** 8

**5. Marco teórico** 10

5.1. Calidad del aire . . . 10
5.1.0.1. Definición . . . 10
5.1.0.2. Normas internacionales . . . 10
5.1.0.3. Norma calidad del aire en Colombia . . . 11
5.1.1. Indicadores que determinan la calidad del aire . . . 12
5.1.1.1. Material Particulado ($PM$) . . . 12
5.1.1.2. Ozono ($O_3$) . . . 12
5.1.1.3. Dióxido de Nitrógeno ($NO_2$) . . . 13
5.1.1.4. Dióxido de Azufre ($SO_2$) . . . 13
5.1.1.5. Monóxido de Carbono ($CO$) . . . 13
5.1.1.6. Indicador de Calidad del Aire (ICA) . . . 13
5.1.1.7. Cálculo del Índice de Calidad el Aire para Colombia [7],[15] . . . 17
5.1.1.8. Elementos que producen los contaminantes que afectan la calidad del aire . . . 18
5.2. Sensores [23] . . . 20
5.2.1. Sensor de gases MICS6814 . . . 20
5.2.1.1. Características . . . 20

5.2.1.2. Funcionamiento . . . . . 20
5.2.2. Sensor material particulado DSM501A [25],[26] . . . . . 22
5.2.2.1. Características . . . . . 22
5.2.2.2. Funcionamiento . . . . . 23
5.2.3. Sensor temperatura, presión y humedad BME280 . . . . . 25
5.2.3.1. Características . . . . . 25
5.2.3.2. Funcionamiento . . . . . 26
5.3. Microcontrolador STM32F103C8 . . . . . 27
5.4. Elementos alimentación del sistema . . . . . 29
5.4.1. Módulo de carga y descarga de batería BM 3S . . . . . 29
5.4.2. Reductor de voltaje LM2596 [36] . . . . . 31
5.4.3. Módulo indicador de batería: E1022 . . . . . 31
5.5. Módulo de lectura y escritura de tarjeta SD [39] . . . . . 32
5.6. Red de sensores inalámbricos . . . . . 33
5.6.1. Definición . . . . . 33
5.6.2. Características [41] . . . . . 33
5.6.3. Elementos [41] . . . . . 34
5.6.4. Topología de red [41] . . . . . 34
5.7. IoT . . . . . 35
5.7.1. Definición [42] . . . . . 35
5.7.2. Características [43] . . . . . 36
5.8. Tecnología y protocolo de red . . . . . 37
5.8.1. LoRa . . . . . 37
5.8.1.1. Definición . . . . . 37
5.8.1.2. Características . . . . . 37
5.8.1.3. Parámetros de red [45], [46] . . . . . 38
5.8.2. LORAWAN . . . . . 41
5.8.2.1. Definición . . . . . 41
5.8.2.2. Características . . . . . 41
5.8.2.3. Arquitectura de la red [50] . . . . . 41
5.8.2.4. Mensajes de enlace descendente y ascendente [52] . . . . . 42
5.8.2.5. Clases de nodo [40] . . . . . 43
5.8.2.6. Formas de conectarse a una red lorawan y seguridad [40] . . . . . 44
5.8.2.7. Estructura de los paquetes LORAWAN . . . . . 45
5.8.3. Módulo LORABEE [54] . . . . . 45
5.8.4. Gateway LG02 [56] . . . . . 47
5.8.5. The Things Network (TTN) [57] . . . . . 48
5.9. Servidores IoT . . . . . 49
5.9.1. ThingSpeak [59] . . . . . 49

5.9.2. Ubidots [60],[61] . . . . . 50

# 6. Estado del arte 52

6.1. Artículos relacionados con la determinación de la calidad del aire . . . . . 52
6.2. Artículos relacionados con las redes LoRa, LORAWAN y plataforma de desarrollo arduino . . . . . 53
6.3. Artículos relacionados con más de una temática del proyecto a realizar . . . . . 55

# 7. Metodología 58

7.1. Definición de los indicadores a medir . . . . . 58
7.2. Arquitectura e implementación nodo final . . . . . 58
7.2.1. Microcontrolador: STM32F103C8 . . . . . 58
7.2.1.1. Comparación entre STM32F103C8 Y Arduino Uno . . . . . 58
7.2.1.2. Configuración del IDE de Arduino para STM32F1013C8 . . . . . 59
7.2.1.3. Formas de subir un código a STM32F103C8 . . . . . 60
7.2.1.4. Ubicación de los periféricos en el STM32F103C8 . . . . . 68
7.2.2. Sensores . . . . . 69
7.2.2.1. BME280: presión, temperatura y humedad . . . . . 69
7.2.2.2. MICS6814: CO y NO2 . . . . . 71
7.2.2.3. DSM501A: PM2.5 Y PM10 . . . . . 73
7.2.3. Módulo LORABEE . . . . . 73
7.2.4. Sistema de almacenamiento externo: Memoria SD . . . . . 78
7.2.5. Código nodo final . . . . . 80
7.2.6. Alimentación . . . . . 81
7.2.6.1. Análisis de potencia . . . . . 81
7.2.6.2. Sistema de alimentación implementado . . . . . 82
7.2.6.3. Tiempo de carga y descarga de las baterías . . . . . 83
7.2.7. Adecuación del prototipo en una caja . . . . . 84
7.3. Conexión nodo final con el gateway . . . . . 88
7.3.1. Configuración del gateway con servidor de Dragino [53] . . . . . 88
7.3.2. Configuración del servidor de red TTN . . . . . 91
7.3.3. Pruebas de conexión del nodo final con el gateway . . . . . 96
7.3.4. Prueba algoritmo de memoria SD con Ubidots y TTN . . . . . 99
7.4. Visualización de los datos . . . . . 102
7.4.1. Función de decodificación en TTN . . . . . 102
7.4.2. Integración de ThingSpeak con TTN . . . . . 107
7.4.3. Gráficas en ThingSpeak . . . . . 110
7.4.4. Alarmas visuales, alarmas de correo electrónico y otros . . . . . 111
7.5. Prueba con diferentes contaminantes . . . . . 112
7.6. Prueba de cobertura de la red . . . . . 117

7.7. Conteo de tramas . . . . . 121
7.8. Costos de la Red . . . . . 122

**8. Análisis de Resultados** **124**

**9. Conclusiones** **128**

**10. Trabajos futuros** **130**

**11. Anexos** **138**
11.1. Código nodo final con todos los sensores . . . . . 138
11.2. Código nodo final con emulador de MICS6814 . . . . . 146
11.3. Código nodo prueba de memoria SD . . . . . 153
11.4. Código decodificador TTN en lenguaje Javascript . . . . . 161
11.5. Código gráficas de barras del ICA de CO en MATLAB . . . . . 166
11.6. Código para crear alerta correo electrónico ThingSpeak en MATLAB . . . . . 169

## Índice de figuras


4.1. Diagrama explicativo del proyecto . . . . . 8

5.1. Gases que puede detectar el sensor MICS6814 . . . . . 20
5.2. Circuito utilizado para alimentar el sensor MICS6814 . . . . . 21
5.3. Auto ajuste de la resistencia de carga para realizar la medición dentro de la zona de tensión optimizada . . . . . 22
5.4. Dimensiones del sensor DSM501A . . . . . 23
5.5. Diagrama del sensor DSM501A . . . . . 24
5.6. Low ratio VS Particle . . . . . 25
5.7. Diagrama de bloques del sensor BME280 . . . . . 27
5.8. Apariencia física del sensor BME280 . . . . . 27
5.9. Aspecto del microcontrolador STM32F103C8 . . . . . 29
5.10. Conexiones del BMS . . . . . 30
5.11. Módulo LM2596 . . . . . 32
5.12. Indicador de nivel de batería . . . . . 32
5.13. Modulo SD. . . . . 33
5.14. Simbología red WSN . . . . . 34
5.15. Topologías de una red WSN: (a) Estrella (b) Malla (c) Híbrida . . . . . 35
5.16. Ancho de Banda de doble banda lateral manejados en LoRa . . . . . 38
5.17. Escala RSSI para clasificación de la señal recibida . . . . . 40
5.18. Relación entre los parámetros de red . . . . . 40
5.19. Arquitectura LORAWAN . . . . . 42
5.20. Mensajes Uplink y Downlink . . . . . 43
5.21. Nodo Tipo A . . . . . 44
5.22. Nodo Tipo B . . . . . 44
5.23. Nodo Tipo C . . . . . 44
5.24. Estructura paquete LoRaWAN . . . . . 45
5.25. Comparación antena de LORABEE y RFM95W convencional . . . . . 46
5.26. Gateway LG02 . . . . . 47
5.27. Utilidades The Things Network . . . . . 49
5.28. Utilidades ThingSpeak . . . . . 50
5.29. Utilidades Ubidots . . . . . 51

7.1. Arquitectura del nodo . . . . . 60
7.2. Añadir soporte para las placas SAND . . . . . 61
7.3. Añadir el STM32F103C8 al gestor de URLs . . . . . 61
7.4. Añadir el STM32F103C8 al gestor de placas . . . . . 62

7.5. Conexiones del conversor USB-TTL . . . . . 63
7.6. Subir el archivo Blink al microcontrolador . . . . . 63
7.7. Drivers y conexiones del ST-LINK . . . . . 64
7.8. Modo de operación del módulo STM32F103C . . . . . 65
7.9. Flash Loader . . . . . 66
7.10. Seleccionar placa STM32F1 . . . . . 66
7.11. Adicionar URL . . . . . 67
7.12. Conexión directa del STM32 por el puerto USB . . . . . 68
7.13. Distribución de pines . . . . . 69
7.14. Conexión BME280 con STM32F103 . . . . . 70
7.15. Mediciones BME280 . . . . . 70
7.16. Conexiones MICS6814 . . . . . 71
7.17. Gráficas MICS6814 . . . . . 72
7.18. Conexiones DSM501A . . . . . 73
7.19. Mediciones DSM501A . . . . . 74
7.20. PCB Adaptador LORABEE . . . . . 74
7.21. Comparación módulo Lora: izquierda: sin el adaptador; derecha: con el adaptador . . 75
7.22. Pines LORABEE . . . . . 76
7.23. Configuraciones iniciales archivo config.h de la librería LMIC . . . . . 76
7.24. Otras configuraciones archivo config.h de la librería LMIC . . . . . 77
7.25. Configuración pines en el IDE de arduino . . . . . 77
7.26. Memoria FLASH sobrepasada . . . . . 78
7.27. Configuración para usar módulo SD en puerto SPI2 en arduino IDE . . . . . 79
7.28. Diagrama de flujo para implementación del algoritmo con la memoria SD . . . . . 80
7.29. Organización y explicación de las partes del código del nodo final . . . . . 81
7.30. Circuito medición de corriente . . . . . 82
7.31. Resultados análisis de corriente . . . . . 82
7.32. Sistema de alimentación . . . . . 83
7.33. Esquemático baquela principal . . . . . 85
7.34. Dibujo baquela principal . . . . . 85
7.35. Esquemático baquela sensores . . . . . 86
7.36. Dibujo baquela sensores . . . . . 86
7.37. Sistema de potencia en la caja . . . . . 87
7.38. Fotografía baquelas ensambladas. Izquierda: baquela principal y derecha baquela con los sensores . . . . . 87
7.39. Fotografías nodo con y sin baquela de los sensores. . . . . 88
7.40. Acceso al Gateway . . . . . 88
7.41. Cuenta de acceso al Gateway . . . . . 89
7.42. Comprobando el estado de la red . . . . . 90

7.43. Análisis de la conexión a internet . . . . . 90
7.44. Configuración de LoRaWAN en el Gateway . . . . . 91
7.45. Configuración de los dos canales del gateway LG02 . . . . . 92
7.46. Registro de nuevo usuario en el servidor TTN . . . . . 92
7.47. Registro del Gateway en TTN . . . . . 93
7.48. Gateway correctamente registrado y conectado EN TTN . . . . . 94
7.49. Ubicación del Gateway . . . . . 94
7.50. Creación de una aplicación en TTN . . . . . 95
7.51. Registro de un dispositivo en TTN . . . . . 96
7.52. Resumen del dispositivo (identificaciones únicas del dispositivo) en TTN . . . . . 97
7.53. Dirección del dispositivo y claves nodo ABP . . . . . 97
7.54. Tráfico del Gateway cuando se establece conexión con el nodo final . . . . . 98
7.55. Datos en la aplicación TTN . . . . . 98
7.56. Metadatos en TTN . . . . . 99
7.57. Número de mensajes diarios de cada nodo. . . . . 99
7.58. Explicación código arduino memoria . . . . . 100
7.59. Formato datos guardados en archivo txt . . . . . 101
7.60. Primeros pasos para programar Downlink en UBIDOTS . . . . . 101
7.61. Otras configuraciones para programar Downlink en UBIDOTS . . . . . 102
7.62. Mensajes programados en TTN . . . . . 102
7.63. Ingreso a la ventana de decodificación en TTN . . . . . 103
7.64. Función de decodificación para mensajes de texto . . . . . 103
7.65. Codificación y decodificación de mensajes en Bytes . . . . . 104
7.66. Asignación de valores para variables de la ecuación del ICA . . . . . 105
7.67. Hallar mayor ICA . . . . . 106
7.68. Retorno de las variables a analizar . . . . . 106
7.69. Creación de un canal en ThinkSpeak . . . . . 107
7.70. Ubicación del nodo y añadir video en el canal . . . . . 108
7.71. Edición de la zona horaria . . . . . 108
7.72. Seleccionar el tipo de integración en TTN . . . . . 109
7.73. Creación de una integración en TTN . . . . . 109
7.74. Gráficas en ThingSpeak . . . . . 110
7.75. Rangos ICA en ThingSpeak . . . . . 111
7.76. Gauge ICA más reciente . . . . . 111
7.77. Alertas correo electrónico THINGSPEAK . . . . . 112
7.78. Complementos expuestos en el canal . . . . . 112
7.79. Nodo expuesto a humo de cigarrillo . . . . . 113
7.80. Resultados sensores expuestos a humo de cigarrillo . . . . . 114
7.81. Nodo expuesto a reacción entre alcohol y oxígeno . . . . . 115

7.82. Resultados sensores expuestos a reacción entre alcohol y oxígeno . . . . . 115
7.83. Nodo expuesto a humo y partículas de papel quemado . . . . . 115
7.84. Resultados sensores expuestos a humo y partículas de papel quemado . . . . . 116
7.85. Nodo expuesto al humo liberado por el exosto de un carro . . . . . 117
7.86. Resultados sensores expuestos al humo liberado por el exosto de un carro . . . . . 118
7.87. Mapa donde se resume las pruebas de cobertura realizadas . . . . . 119
7.88. Mapa satelital donde se resume las pruebas de cobertura realizadas . . . . . 120
7.89. Precios de nodos y redes en el mercado . . . . . 123

## Índice de tablas


5.1. Máximos permisibles de contaminantes criterio en el aire . . . . . 11
5.2. ICA $CO$ . . . . . 14
5.3. ICA $O_3$ . . . . . 14
5.4. ICA $PM_{10}$ y $PM_{2,5}$ . . . . . 15
5.5. ICA $SO_2$ . . . . . 16
5.6. ICA $NO_2$ . . . . . 17
5.7. Puntos de corte del ICA . . . . . 18
5.8. Rangos del factor de difusión . . . . . 39
5.9. Sobrecarga de codificación cíclica . . . . . 39
5.10. Frecuencias Sub-bandas LoRaWAN en América . . . . . 42

7.1. Comparación Arduino Uno VS STM32F103C8 . . . . . 58
7.2. Pruebas conteo de tramas . . . . . 121
7.3. Costo prototipo nodo final . . . . . 122

# 1. Planteamiento del problema

El crecimiento progresivo de las industrias además de la facilidad actual de adquirir vehículos como carros y motos, han incrementado en gran proporción la contaminación del aire, ya que estos son los principales causantes de combustiones y dióxido de nitrógeno (Gas de color pardo rojizo fuertemente tóxico cuya presencia en el aire de los centros urbanos se debe a la oxidación del nitrógeno atmosférico [1]). En Colombia, el monitoreo y control de la contaminación atmosférica ha tomado día a día mayor relevancia, ya que *"según los resultados del estudio del Instituto Nacional de Salud -INS- sobre la carga de enfermedades ambientales en Colombia, 15.681 personas fallecieron por enfermedades asociadas a la calidad del aire en el país, es decir, el 8 % del total de muertes en Colombia está asociada al impacto ambiental, y de este porcentaje, el mayor número se produce por enfermedades relacionadas con factores de riesgo asociados a la contaminación atmosférica"* [2].

Por esta razón, *"se ha implementado la Red de Monitoreo de la Calidad de Aire en Bogotá, más conocida como RMCAB, que permite recolectar información sobre la concentración de contaminantes de origen antropogénico y natural y el comportamiento de las variables meteorológicas que regulan la distribución de los mismos en la atmósfera bogotana. Los datos recolectados en distintos sitios de la ciudad se reciben en una estación central donde se someten a un proceso de validación final y posterior análisis con el fin de evaluar el cumplimiento de los estándares de calidad de aire en Bogotá dados por la Resolución 610 del 24 de marzo de 2010 expedida por el entonces Ministerio de Ambiente, Vivienda y Desarrollo Territorial (MAVDT). Además, resulta información base para la definición de las políticas de control de la contaminación y de la gestión ambiental"* [3]. Estas estaciones de monitoreo y control de calidad del aire se encuentran localizadas principalmente en las grandes ciudades debido a sus altos costos, ya que utilizan sensores de última tecnología, además de un software privado para la manipulación de datos.

Además de estaciones exteriores, también existen algunas estaciones que monitorean la Calidad del Aire Interior (IAQ) y otros factores, como la temperatura y el ruido, sin embargo, muchos de estos sistemas también tienen un alto costo para tratar lo que es denominado como el Edificio Enfermo. Pero teniendo en cuenta las necesidades presentadas para mejorar la calidad del aire la ingeniería ha buscado reducir costos haciendo uso de software libre en sus sistemas de procesamiento, entre ellos encontramos: el desarrollo de un sistema de red de sensores inalámbricos de bajo costo para aplicaciones de monitoreo de calidad del aire interior, que utiliza Arduino, módulos XBee y sensores de micro gas [4], el monitoreo del ambiente interior basado en condiciones de humedad utilizando una red de sensores de bajo costo [5] y el monitoreo continuo de la calidad ambiental interior utilizando un sistema de adquisición de datos basado en Arduino, donde se conectaron a los sensores para la adquisición de datos, el protocolo de comunicación ZigBee se estableció entre un dispositivo XBee para cada placa Arduino y un receptor XBee conectado a una computadora, además, la caja de he-

rramientas utilizó la plataforma de software basada en agentes de código abierto VOLTTRON para la comunicación y el análisis de datos [6].

Teniendo en cuenta lo anterior se propone implementar un prototipo basado en un microcontrolador con sensores de bajo costo que permitan medir algunas condiciones ambientales como presión y temperatura, e indicadores para monitorear la calidad del aire en exteriores, como el sensor de partículas PM10 y PM2.5, sensor de monóxido de carbono y sensor de dióxido de nitrógeno, además este prototipo estaría en la capacidad de generar alarmas de acuerdo a los rangos de los sensores a implementar basándonos en los niveles de contaminación en el aire especificados por la Resolución 2254 de 2017 del Ministerio de Ambiente, Vivienda y Desarrollo Territorial [7]. Además, se contará con la asesoría del ingeniero Andrés Torres Abello de la universidad Pontificia Javeriana de Bogotá para la implementación de varios de estos prototipos conformen una Red de sensores inalámbricos (WSN) utilizando los protocolos LORAWAN y WIFI. Los prototipos serán probados haciendo uso de distintas concentraciones de los indicadores mencionados. Esta red podría ser usada por sus bajos costos en algunas poblaciones del país con el fin de tener mediciones de los indicadores de calidad del aire en exteriores, ya que muchas poblaciones están teniendo desarrollos importantes que incluyen nuevas sedes de industrias.

De acuerdo a lo anterior, ¿Es posible implementar una red de sensores inalámbricos de bajo costo, utilizando el protocolo LORAWAN para el monitoreo de material particulado, monóxido de carbono y dióxido de nitrógeno, como indicadores de calidad del aire en exteriores exponiendo los sensores a diferentes fuentes de contaminación ubicados cerca del área de influencia de la Facultad de Ingeniería de la Universidad Distrital.?

# 2. Justificación

## 2.1 Investigativa

Para el desarrollo del proyecto como primer paso se hace necesario entender el funcionamiento de los sensores de partículas y de gases que permitirán la medición de los indicadores que determinan la calidad del aire, además de interpretar los datos obtenidos de dichos sensores con ayuda del microcontrolador. Después de tener las medidas de los indicadores, se debe investigar la implementación de redes con la arquitectura LORAWAN utilizando un microcontrolador de bajo costo y además la conexión a Internet necesaria para monitorear los datos. Esta investigación da un paso para la implementación de otras redes de sensores inalámbricos en exteriores, que tengan en cuenta otras variables y por ende puedan resolver otros problemas.

## 2.2 Académica

Para el desarrollo del proyecto se hace necesario relacionar varias temáticas como instrumentación que abarcan el uso de los sensores para la toma de datos de los indicadores de contaminación del aire, la implementación de conceptos de redes y protocolos como LORAWAN con conexión WIFI, que serán el camino para el monitoreo de cada uno de los nodos que como base cuentan con un microcontrolador.

## 2.3 Social

El desarrollo de este proyecto podría ser de ayuda a las organizaciones encargadas de monitorear los niveles de los indicadores de contaminación del aire, pudiendo reaccionar de manera adecuada ante eventos de emergencia ambiental y a su vez evitar que el número de personas que padecen enfermedades asociadas a la calidad del aire aumente en el país, ya que *según los resultados del estudio del Instituto Nacional de Salud -INS, en Colombia, 15.681 personas fallecieron por causas asociadas a la calidad del aire, es decir, el 8 % del total de muertes en Colombia está asociada al impacto ambiental y la contaminación atmosférica* [2], enfermedades cardiorrespiratorias como el asma, bronquitis e incluso ciertos tipos de cáncer.

Las personas con enfermedades previas, los niños menores de cinco años y los adultos entre 50 y 75 años de edad son los más afectados. Las personas pobres y aquellas que viven en situación de vulnerabilidad, así como las mujeres y sus hijos que utilizan estufas tradicionales de biomasa para cocinar y calentarse, también corren mayor riesgo. En efecto, el 36 % de las muertes por cáncer de pulmón, el 35 % de la enfermedad pulmonar obstructiva crónica (COPD), el 34 % de los accidentes cerebrovasculares y el 27 % de las cardiopatías isquémicas son atribuibles a la contaminación atmosférica [8].

Sin embargo, el mayor impacto es sobre la mortalidad infantil, ya que más de la mitad de las muertes de niños menores de 5 años por infecciones agudas de las vías respiratorias inferiores (ALRI) son debidas a partículas inhaladas por la contaminación del aire interior producto del uso de combustibles sólidos.

Los informes generados con las mediciones respectivas de los parámetros de la calidad del aire ayudarán a generar nuevas políticas para reducir la contaminación del aire en muchos sectores del país ya que será muy asequible para pueblos alrededor del país, y la baja contaminación en el aire reduciría gastos por el estado debido a que estas enfermedades afectan a la población más vulnerable.

### 2.4 Económica

Se estima que los costos de adquisición de datos disminuirían debido a que usaremos en el desarrollo del sistema software libre, donde se implementará un sistema de monitoreo para mediciones de la calidad del aire exterior, haciendo uso de una Red de sensores inalámbricos (WSN) de bajo costo, que no se compara con la Red de Monitoreo de la Calidad de Aire de Bogotá (RMCAB), por ejemplo.

*"El Departamento Nacional de Planeación de Colombia (DNP) concluyó en un informe en 2017 que los costos en la salud asociados a la degradación ambiental en Colombia ascienden a $20,7 billones, equivalentes al 2,6% del PIB del año 2015, relacionados con 13.718 muertes y cerca de 98 millones de síntomas y enfermedades. Dentro de estos costos, la contaminación del aire urbano aportó el 75%, con $15,4 billones (1,93% del PIB de 2015) asociados a 10.527 muertes y 67,8 millones de síntomas y enfermedades. En Bogotá, el 10,5% (3.219) del total de las muertes que se presentan en la ciudad, son atribuidas a la contaminación del aire urbano, lo que generó costos estimados en $4,2 billones, equivalentes al 2,5% del PIB de la ciudad"* [9].

### 2.5 Ambiental

El desarrollo de este proyecto permitiría monitorear los indicadores que determinan la calidad del aire y por ende tomar las medidas pertinentes de regulación, como por ejemplo lo que se hace en Bogotá aplicando el pico y placa ambiental, ya que en la capital el principal componente químico de uno de los contaminantes son las partículas de carbono asociadas a emisiones vehiculares de gasolina y diésel, además *"el distrito también está gestionando una licitación para renovar la flota del transporte público, así como las volquetas y los camiones de carga pesada, que sería una medida clave para reducir la contaminación del aire en Bogotá"* [10]. De igual forma, si el sistema se implementará en algunos municipios se podría determinar y hacer control de las principales causas de contaminación en estos lugares, donde *"la erosión del suelo, el deterioro del pavimento, la resuspensión del polvo vial, así como actividades de construcción y canteras también contaminan el aire"* [10]. Además, es de tener en cuenta que en algunos municipios hay más presencia de vehículos antiguos y estos

contaminan el aire hasta 40 veces más que las nuevas tecnologías [11].

### 2.6 Personal

Procesar las mediciones de los parámetros de calidad del aire para generar informes que ayuden a contribuir en el diseño de nuevas políticas ambientales para la reducción de contaminación atmosférica y con ello disminuir la cantidad de personas afectadas por enfermedades causadas por la contaminación del aire y la atmósfera. Ayudar a la población con esta problemática sería de gran orgullo y satisfacción, porque todos los seres vivos merecen una mejor calidad de vida y los conocimientos adquiridos durante el pregrado serán utilizados para beneficio de la sociedad.

# 3. Objetivos

## 3.1 Objetivo general

- Implementar una red de sensores inalámbricos de bajo costo, utilizando el protocolo LORAWAN para el monitoreo de material particulado, monóxido de carbono y dióxido de nitrógeno como indicadores de calidad del aire en exteriores, exponiendo los sensores a diferentes fuentes de contaminación ubicados cerca del área de influencia de la Facultad de Ingeniería de la Universidad Distrital.

## 3.2 Objetivos específicos

- Comprender los indicadores de calidad del aire en centros urbanos.
- Determinar los indicadores de calidad del aire y los factores ambientales a monitorear en el proyecto.
- Determinar y realizar pruebas con los sensores encargados de medir el material particulado (PM10 y PM2.5), el dióxido de nitrógeno, el monóxido de carbono, la temperatura y la presión atmosférica, indicadores que permiten evaluar la calidad del aire en exteriores.
- Comprender y determinar la arquitectura de los nodos que serán los encargados de recibir la información de los sensores, procesarla y enviarla al Gateway de la red.
- Realizar la implementación de cada nodo teniendo en cuenta la arquitectura definida.
- Implementar el sistema de red de sensores con el protocolo de comunicación de LORAWAN.
- Determinar e implementar la plataforma que permitirá visualizar y realizar el análisis a través de los protocolos de internet, generando las respectivas alarmas de acuerdo a los niveles de los indicadores de calidad del aire.

# 4. Diagrama del proyecto

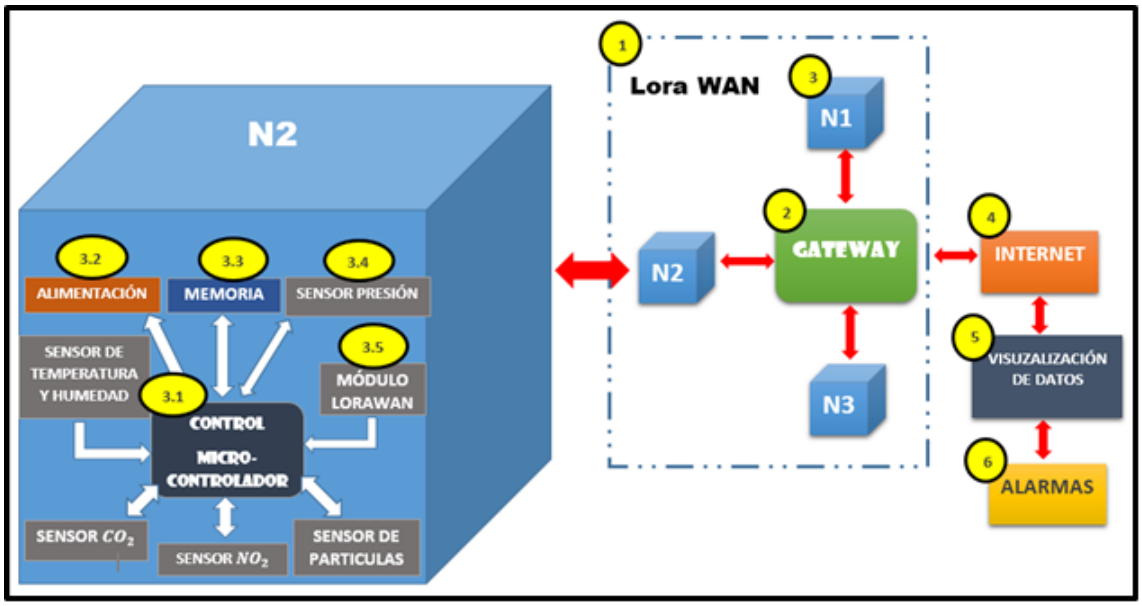



**Figura 4.1: Diagrama explicativo del proyecto**

Fuente: Autores

En la Figura 4.1 se muestra el diagrama que explica el proyecto a realizar. A continuación, se describen cada una de las partes del proyecto teniendo en cuenta la numeración de dicha figura.

- En (1) se representa la red con el protocolo LORAWAN y topología estrella. Esta red cuenta con un Gateway (2) que se encarga de recibir los datos provenientes de los nodos (3) (N1, N2 y N3).
- Los nodos (3) corresponden a cada uno de los prototipos que va a medir los indicadores de la calidad del aire y algunos factores ambientales. Estos prototipos se componen de un microcontrolador (3.1) que se encarga de recibir la información de los sensores y decidir si se envían al Gateway o se almacenan en la Memoria (3.3), dependiendo si hay o no conexión a la red. Una parte importante es la Alimentación (3.2) estará compuesta por una batería dando la posibilidad de mover el prototipo. El nodo tendrá varios sensores (3.4): dos sensores de gases como lo son el Monóxido de Carbono (CO) y el Dióxido de Nitrógeno (NO2); un sensor de material particulado (PM) y 3 sensores de condiciones ambientales como lo son el de temperatura, presión atmosférica y humedad. El último elemento del prototipo es el Módulo LORAWAN (3.5) que es el encargado de transmitir los datos al Gateway (2).
- Después de recibir la información en el Gateway (2) se procede a visualizar los datos (5) en un sistema de información que tenga conexión a Internet (4).

- En el sistema de información se presentarán alarmas (6) de acuerdo, a las concentraciones de los indicadores. Las alarmas dependerán de los niveles de los indicadores de contaminación del aire, que estarán limitados por los rangos de los sensores a usar y basados en la normatividad de la Calidad del Aire en Colombia (Resolución 2254 del 2017).

A partir de la figura 4.1 se decidieron los temas que son parte del marco teórico.

# 5. Marco teórico

## 5.1 Calidad del aire

En esta sección se hablarán de las normas a nivel nacional e internacional existentes sobre los factores que afectan la calidad de aire, además se definirán cada uno de los contaminantes.

#### 5.1.0.1. Definición

El índice de calidad de aire es una cifra que proporciona a las autoridades de una zona y estas cifras reflejan las cantidades de contaminantes presentes en el aire. La contaminación atmosférica es generada por la emisión, acumulación y mezcla de contaminantes, entre muchos otros factores provenientes de fuentes naturales como volcanes, plantas, animales y fuentes antropogénicas como industrias, servicios y vehículos.

Cuando la calidad del aire es buena, tiene pocos contaminantes además se considera que el aire limpio es un requisito básico de la salud y el bienestar humano para que las personas puedan respirarlo indefinidamente sin que su salud se vea afectada. Cuando es mala pueden darse irritaciones oculares, todo tipo de problemas respiratorios e incluso cardíacos. Si la calidad del aire baja o se prevé que pueda bajar de unos límites, que son distintos en cada país, las autoridades pueden tomar medidas como prohibir la circulación de todos los vehículos que emitan contaminación, de algunos de ellos, prohibir la práctica de deporte al aire libre o recomendar el uso de mascarillas [12].

De acuerdo con la forma en que se liberan los contaminantes al aire, las fuentes de emisión se clasifican en: fuentes fijas en las que las emisiones se generan en un lugar determinado e inamovible como por ejemplo las industrias y fuentes móviles en las que, por razón de su uso o propósito, las emisiones se generan durante el desplazamiento de la fuente como vehículos, trenes y aviones.

Los contaminantes que se generan en mayor cantidad son: el MP (Material Particulado), los óxidos de azufre $SO_2$, los óxidos de nitrógeno $NO_2$, el monóxido de carbono $CO$, y el dióxido de carbono $CO_2$, lo anterior como consecuencia del uso masivo de combustibles fósiles, la suspensión de polvos y la explotación minera [12].

#### 5.1.0.2. Normas internacionales

La OMS (Organización Mundial de la Salud) ofrece orientación sobre la manera de reducir los efectos de la contaminación del aire en la salud mediante las guías de calidad de aire. Estas guías se basan en la evaluación de pruebas científicas.

Estas guías tienen por objeto informar a los encargados de la formulación de políticas y proporcionar objetivos apropiados para una amplia variedad de opciones en materia de políticas en relación con la gestión de la calidad del aire en diferentes partes del mundo.

Las Guías de calidad del aire se refiere a cuatro contaminantes comunes del aire: MP, ozono *($O_3$)*, $NO_2$ y $SO_2$. El ámbito de este examen pone de manifiesto la disponibilidad de nuevas pruebas sobre los efectos de los mencionados contaminantes en la salud y su importancia relativa con respecto a los efectos presentes y futuros de la contaminación del aire en la salud en cada una de las regiones de la OMS [13].

#### 5.1.0.3. Norma calidad del aire en Colombia

El Ministerio de Ambiente y Desarrollo Sostenible presentó la nueva norma de Calidad del Aire (Resolución 2254 de 2017), que plantea las acciones que se deben tomar desde el Gobierno Nacional para alcanzar niveles adecuados que permitan proteger la salud de los colombianos.

Desde hace dos años, el jefe de la cartera verde venía trabajando en esta nueva norma, la cual de forma progresiva y con un horizonte de tiempo a 2030, establece niveles máximos permisibles más estrictos de los contaminantes que afectan la salud en los centros urbanos, principalmente las partículas finas $PM_{10}$ y $PM_{2,5}$ [14].

En el año 2010 el Ministerio de Ambiente y Desarrollo Sostenible adopto la política de prevención y control de la contaminación del aire – PPCCA la cual tiene como objetivo "Impulsar la gestión de la calidad del aire en el corto, mediano y largo plazo, con el fin de alcanzar los niveles de calidad del aire adecuados para proteger la salud y el bienestar humano, en el marco del desarrollo sostenible" [14].

En el artículo 2 *"Niveles máximos permisibles de contaminantes criterio"*. En la tabla No.1 se establecen los niveles máximos permisibles a condiciones de referencia para contaminantes criterio que regirán a partir del primero de enero del año 2018 [7].

**Tabla 5.1: Máximos permisibles de contaminantes criterio en el aire**

| Contaminante | Nivel máximo permisible | Tiempo de exposición |
|---|---|---|
| $PM_{10}$ | 50 | Anual |
| | 100 | 24 horas |
| $PM_{2,5}$ | 25 | Anual |
| | 50 | 24 horas |
| $SO_2$ | 50 | 24 horas |
| | 100 | 1 hora |

| Contaminante | Nivel máximo permisible | Tiempo de exposición |
|---|---|---|
| $NO_2$ | 60 | Anual |
| | 200 | 1 hora |
| O3 | 100 | 8 horas |
| $CO$ | 5000 | 8 horas |
| | 35000 | 1 hora |

Fuente: [7]

*"**Parágrafo 1:** A partir del 1 de Julio de 2018, los niveles máximos permisibles de $PM_{10}$ y $PM_{2,5}$ para un tiempo de exposición 24 horas serán de 75 ug/$m^3$ y 37$ug/m^3$ respectivamente."*[7].

### 5.1.1. Indicadores que determinan la calidad del aire

#### 5.1.1.1. Material Particulado (*PM*)

La resolución número 610 del 24 de marzo de 2010 del Ministerio de Ambiente, vivienda y desarrollo territorial en Colombia se define el material particulado como se muestra a continuación:

- *PST* **(Partículas Suspendidas Totales):** Material particulado que incluye tanto a la fracción inhalable como a las mayores de 10 micras, que no se sedimentan en periodos cortos, sino que permanecen suspendidas en el aire debido a su tamaño y densidad.
- $PM_{10}$ **(Material Particulado Menor a 10 Micras):** Material Particulado con un diámetro aerodinámico menor o igual a 10 micrómetros nominales.
- $PM_{2,5}$ **(Material Particulado Menor a 2,5 Micras):** Material Particulado con un diámetro aerodinámico menor o igual a 2,5 micrómetros nominales [1].

En la Tabla 5.1 se muestran los niveles máximos permisibles de este contaminante.

#### 5.1.1.2. Ozono ($O_3$)

La resolución número 610 del 24 de marzo de 2010 del Ministerio de Ambiente, vivienda y desarrollo territorial en Colombia se define el ozono como se muestra a continuación:

*"Gas azul pálido que, en las capas bajas de la atmósfera, se origina como consecuencia de las reacciones entre los óxidos de nitrógeno y los hidrocarburos (gases compuestos de carbono e hidrógeno principalmente) en presencia de la luz solar"* [1].

En la Tabla 5.1 se muestran los niveles máximos permisibles de este contaminante.

#### 5.1.1.3. Dióxido de Nitrógeno ($NO_2$)

La resolución número 610 del 24 de marzo de 2010 del Ministerio de Ambiente, vivienda y desarrollo territorial en Colombia se define el dióxido de nitrógeno como se muestra a continuación:

*"Gas de color pardo rojizo fuertemente tóxico cuya presencia en el aire de los centros urbanos se debe a la oxidación del nitrógeno atmosférico que se utiliza en los procesos de combustión en los vehículos y fábricas."* [1].

En la Tabla 5.1 se muestran los niveles máximos permisibles de este contaminante.

#### 5.1.1.4. Dióxido de Azufre ($SO_2$)

La resolución número 610 del 24 de marzo de 2010 del Ministerio de Ambiente, vivienda y desarrollo territorial en Colombia se define el dióxido de azufre como se muestra a continuación:
*"Gas incoloro, no inflamable que posee un fuerte olor en altas concentraciones."* [1].

En la Tabla 5.1 se muestran los niveles máximos permisibles de este contaminante.

#### 5.1.1.5. Monóxido de Carbono ($CO$)

La resolución número 610 del 24 de marzo de 2010 del Ministerio de Ambiente, vivienda y desarrollo territorial en Colombia se define el dióxido de nitrógeno como se muestra a continuación:

*"Gas inflamable, incoloro e insípido que se produce por la combustión de combustibles fósiles"* [1].

En la Tabla 5.1 se muestran los niveles máximos permisibles de este contaminante.

#### 5.1.1.6. Indicador de Calidad del Aire (ICA)

De acuerdo al protocolo para el monitoreo y seguimiento de la calidad del aire del Instituto de Hidrología, Meteorología y Estudios Ambientales (IDEAM), la definición de *"el Índice de calidad del aire (ICA) permite comparar los niveles de contaminación del aire de las estaciones de monitoreo que conforman un Sistema de Vigilancia de Calidad del Aire (Unidades espaciales de referencia), en un tiempo t, que corresponde al período de exposición previsto en la norma para cada uno de los contaminantes que se está midiendo"*[15], es de tener en cuenta que los contaminantes a los cuales se les determinara el valor del ICA en Colombia son los contaminantes denominados **contaminantes criterio**, que se han nombrado anteriormente.

En una escala numérica que oscila en valores entre 0 y 500 se asigna un color al ICA, para relacionarlo con los efectos que este tiene sobre la salud, a continuación en las tablas 5.2, 5.3, 5.4, 5.5 y 5.6

se nombran los contaminantes de acuerdo a la escala nombrada anteriormente y los efectos que estos representan en la salud de la población.

**Tabla 5.2: ICA *CO***

| **Rangos ICA** | **Clasificación** | **CO, $ug/m^3$, 8 horas** | **Acciones preventivas** |
|---|---|---|---|
| 0-50 | Verde | 0-5094 | Ninguna. |
| 51-100 | Amarillo | 5095-10819 | Ninguna. |
| 101-150 | Anaranjado | 10820-14254 | Personas con enfermedades cardiovasculares como la angina deben reducir la actividad física y las fuentes de CO como el tráfico pesado. |
| 151-200 | Rojo | 14255-17688 | Personas con enfermedades cardiovasculares como la angina, deben reducir moderadamente el esfuerzo y evitar la exposición a fuentes de CO como el tráfico pesado. |
| 201-300 | Púrpura | 17689-34862 | Personas con enfermedades cardiovasculares como la angina, deben reducir el esfuerzo y evitar la exposición a fuentes de CO como el tráfico pesado. |
| 301-500 | Marrón | 34863-57703 | Personas con enfermedades cardiovasculares como la angina, deben reducir el esfuerzo y evitar la exposición a fuentes de CO como el tráfico pesado; todos los demás deben limitar el esfuerzo. |

Fuente: [15]

**Tabla 5.3: ICA $O_3$**

| **Rangos ICA** | **Clasificación** | **$O_3$, $ug/m^3$, 8 horas** | **Acciones preventivas** |
|---|---|---|---|
| 0-50 | Verde | 0-106 | Ninguna. |
| 51-100 | Amarillo | 107-138 | Ninguna. |
| 101-150 | Anaranjado | 139-167 | Niños activos, adultos y personas con enfermedades respiratorias como asma, deben reducir la actividad física al aire libre. |

Continúa en la página siguiente

| Rangos ICA | Clasificación | $O_3$, $ug/m^3$, 8 horas | Acciones preventivas |
|---|---|---|---|
| 151-200 | Rojo | 168-207 | Niños y adultos activos y personas con enfermedades respiratorias, deben reducir la actividad física prolongada al aire libre, especialmente los niños. |
| 201-300 | Púrpura | 208-393 | Niños activos, adultos y personas con enfermedades respiratorias como asma, deben reducir la actividad física al aire libre; especialmente los niños. |
| 301-500 | Marrón | 394 | Todas las personas deben evitar el esfuerzo al aire libre. |

Fuente: [15]

**Tabla 5.4: ICA $PM_{10}$ y $PM_{2,5}$**

| Rangos ICA | Clasificación | $PM_{10}$ y $PM_{2,5}$, $ug/m^3$, 8 horas | Acciones preventivas |
|---|---|---|---|
| 0-50 | Verde | 0-54 | Ninguna. |
| 51-100 | Amarillo | 55-154 | Personas extremadamente sensibles con asma y adultos con enfermedad cardio-cerebrovascular como hipertensión arterial, enfermedad isquémica del miocardio o pulmonar como asma, enfisema y bronquitis crónica deben reducir la actividad física fuerte o prolongada. |
| 101-150 | Anaranjado | 155-254 | Personas con enfermedades cardiacas o respiratorias, mayores de 60 años y niños deben evitar la actividad física fuerte o prolongada. |
| 151-200 | Rojo | 255-354 | Personas con enfermedades cardiovascular o respiratorias, mayores de 60 años y niños deben evitar actividad física fuerte o prolongada. |

Continúa en la página siguiente

| **Rangos ICA** | **Clasificación** | $PM_{10}$ **y** $PM_{2,5}$**,** $ug/m^3$**, 8 horas** | **Acciones preventivas** |
|---|---|---|---|
| 201-300 | Púrpura | 355-424 | Personas con enfermedades cardiacas o respiratorias, mayores de 60 años y niños deben evitar cualquier actividad física en exteriores. Las demás personas deben evitar la actividad fuerte o prolongada. |
| 301-500 | Marrón | 425-604 | Todas las personas deben evitar el esfuerzo al aire libre. |

Fuente: [15]

**Tabla 5.5: ICA** $SO_2$

| **Rangos ICA** | **Clasificación** | $SO_2$**,** $ug/m^3$**, 8 horas** | **Acciones preventivas** |
|---|---|---|---|
| 0-50 | Verde | 0-93 | Ninguna. |
| 51-100 | Amarillo | 94-197 | Ninguna. |
| 101-150 | Anaranjado | 198-486 | Personas con asma deben considerar reducir la actividad al aire libre. |
| 151-200 | Rojo | 487-797 | Los niños, los asmáticos y las personas con enfermedades cardiacas y pulmonares deben reducir el esfuerzo al aire libre . |
| 201-300 | Púrpura | 798-1583 | Niños, asmáticos y personas con enfermedades cardiacas y pulmonares deben evitar el esfuerzo al aire libre; todos los demás deben reducir el esfuerzo al aire libre. |
| 301-500 | Marrón | 1584-2629 | Niños, asmáticos y personas con enfermedades cardiacas o pulmonares no deben estar al aire libre; todos los demás deben evitar el esfuerzo al aire libre. |

Fuente: [15]

**Tabla 5.6: ICA $NO_2$**

| **Rangos ICA** | **Clasificación** | $NO_2$**,** $ug/m^3$**, 8 horas** | **Acciones preventivas** |
|---|---|---|---|
| 0-50 | Verde | 0-100 | Ninguna. |
| 51-100 | Amarillo | 101-189 | Ninguna. |
| 101-150 | Anaranjado | 190-677 | Ninguna. |
| 151-200 | Rojo | 678-1221 | Ninguna. |
| 201-300 | Púrpura | 1222-2349 | Niños y personas con enfermedades respiratorias como asma deben reducir el esfuerzo al aire libre. |
| 301-500 | Marrón | 2350-3853 | Niños y personas con enfermedades respiratorias como asma, deben reducir el esfuerzo moderado o pesado al aire libre. |

Fuente: [15]

#### 5.1.1.7. Cálculo del Índice de Calidad el Aire para Colombia [7],[15]

A partir de la metodología usada por la US-EPA para calcular el AQI, el índice de calidad del aire ICA sera calculado para cada uno de los contaminantes individualmente y se reportara el mayor de los valores que se obtengan del ICA para realizar el calculo, se hace uso de la ecuación (1):

$$I_P = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}(C_P - BP_{Lo} + I_{Lo}) \quad (1)$$

Donde:
$I_P$= Índice para el contaminante p.
$C_P$= Concentración medida para el contamimante p.
$BP_{Hi}$= Punto de corte mayor o igual a $C_P$
$BP_{Lo}$= Punto de corte menor o igual a $C_P$
$I_{Hi}$= Valor del indice de calidad del aire correspondiente al $BP_{Hi}$
$I_{Lo}$= Valor del indice de calidad del aire correspondiente al $BP_{Lo}$

*"Teniendo en cuenta que el ICA tiene una correlación directa con los efectos en la salud, los puntos de corte del ICA son los límites correspondientes a efectos entre la salud y la calidad del aire"*, es por ello que en la tabla 5.7 se representan los puntos de corte.

**Tabla 5.7: Puntos de corte del ICA**

| Índice de Calidad del Aire | | | Puntos de corte del ICA | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **ICA** | **Color** | **Cat.** | **PM10** $ug/m^3$ **24 horas** | **PM2.5** $ug/m^3$ **24 horas** | **CO** $ug/m^3$ **8 horas** | **SO2** $ug/m^3$ **1 hora** | **NO2** $ug/m^3$ **1 hora** | **O3** $ug/m^3$ **8 horas** | **O3** $ug/m^3$ **1 hora** |
| 0-50 | Verde | Buena | 0-54 | 0-12 | 0-5094 | 0-93 | 0-100 | 0-106 | – |
| 51-100 | Amarillo | Aceptable | 55-154 | 13-37 | 5095-10819 | 94-197 | 101-189 | 107-138 | – |
| 101-150 | Naranja | Dañina a la salud de grupos sensibles | 155-254 | 38-55 | 10820-14254 | 196-486 | 190-677 | 139-167 | 254-323 |
| 151-200 | Rojo | Dañina a la salud | 255-354 | 56-150 | 14255-17688 | 487-797 | 678-1221 | 168-207 | 324-401 |
| 201-300 | Muy dañina a la salud | Buena | 355-424 | 151-250 | 17689-34862 | 796-1583 | 1222-2349 | 208-393 | 402-794 |
| 301-500 | Marrón | Peligrosa | 425-604 | 251-500 | 34863-57703 | 1584-2629 | 2350-3853 | 394 | 795-1185 |

Fuente: [7]

#### 5.1.1.8. Elementos que producen los contaminantes que afectan la calidad del aire

Como ya se nombraron anteriormente los contaminantes del aire y las afectaciones que estos tienen sobre la salud de las personas, es vital conocer qué producen estos contaminantes en el ambiente, la información de cómo estos reaccionan con el medio y cuáles son sus principales causantes, para ello se hablará de cada uno a continuación:

- **Ozono ($O_3$):** Este contaminante fotoquímico, se origina principalmente en áreas urbanas por la emisión de los automóviles y la industria, la concentración elevada de ozono reacciona para destruir o alterar otras moléculas, este produce daños respiratorios pulmonares, dolores de pecho, irritación de la garganta y ojos, empeoramiento de enfermedades preexistentes del corazón [16].

  *El ozono tiene efectos nocivos sobre la vegetación, ya que deteriora las hojas de los árboles y plantas, así como reduce el rendimiento de los cultivos y el crecimiento de los bosques debido a*

*que interfiriere en la capacidad de almacenar y producir nutrientes, lo cual, hace más susceptibles las plantas a insectos, a otros contaminantes y a las inclemencias del tiempo. El ozono deteriora materiales de uso común, como el caucho, el nylon, los plásticos, los colorantes y las pinturas, corroe metales y deteriora las llantas de los vehículos* [16].

- **Material particulado ($PM_{10}$ y $PM_{2,5}$):** Los principales componentes son: sulfatos, nitratos, el amoníaco, el cloruro sódico, el carbón, el polvo de minerales, cenizas metálicas y agua. Según su tamaño se clasifican en partículas PM 10, que serían las de mayor tamaño, cuya diámetro aerodinámico teórico sería de 10 *um* (micrones de metro = millonésima parte del metro) y las partículas finas conocidas como PM 2.5 cuyo diámetro sería de 2.5 *um* [17].

  Mientras que la partículas PM 10 quedarían retenidas en las vías respiratorias, produciendo efectos a nivel de sistema respiratorio, las partículas menores, como las PM 2.5, tienen la capacidad de pasar al torrente sanguíneo por lo que pueden, potencialmente, dañar cualquier órgano o sistema [17].

- **Monónoxido de carbono ($CO$):** *"Las fuentes más comunes de producción de monóxido de carbono son los calentadores de paso a gas, los gases de escape de motores a explosión, los incendios, el humo de tabaco y determinados procesos industriales"*. Para ser mas general, todo aparato que genere combustión puede generar monóxido de carbono (CO) si hay problemas de conexión o de tipo técnico [18].

- **Dióxido de azufre $SO_2$:** La principal fuente de emisión de dióxido de azufre es la combustión de productos petrolíferos, la quema de carbón en centrales eléctricas y calefacciones centrales. Existen también algunas fuentes naturales, como es el caso de los volcanes [19]. Este contaminante también se da por la combustión de sustancias que contengan Azufre, quemadores industriales que emplean carbón y gas óleo y vehículos a diesel [20].

- **Dióxido de nitrógeno $NO_2$:** El dióxido de nitrógeno en la naturaleza se produce por los incendios forestales o las erupciones volcánicas. También se produce de forma natural por la descomposición de nitratos orgánicos. La mayor parte de este contaminante tiene su origen en la oxidación del $NO$ que se produce en la combustión de los motores de los vehículos, fundamentalmente los diesel. El $NO$ emitido por los motores, una vez en la atmósfera, se oxida y se convierte en $NO_2$. Es también un potenciador del material particulado, sobre todo de partículas finas PM2,5 que son las más perjudiciales. En su reacción con la luz UV del sol es un precursor de $O_3$ ozono troposférico [21].

  El $NO_2$, es un contaminante que se produce como resultado de las combustiones por oxidación del nitrógeno de la atmósfera, la actividad microbiana en el suelo, la utilización de fertilizantes nitrogenados, la quema de leña, etc. Es uno de los gases generadores de la lluvia ácida, a su vez, puede contribuir a los efectos de invernadero y al agotamiento del ozono [22].

## 5.2 Sensores [23]

### 5.2.1. Sensor de gases MICS6814

El MICS-6814 es un sensor MEMS robusto para detección de contaminación por escapes de automóviles y para olores agrícolas / industriales.

#### 5.2.1.1. Características

Algunas de las características del sensor de gases MIC6814 son:

- Es de tamaño pequeño para los diseños compactos (5 x 7 x 1,55 *mm*).
- El MICS6814 incluye tres chips de sensores con calentadores independientes y capas sensibles. Un chip sensor detecta los gases oxidantes (OX), el otro sensor detecta los gases reductores (RED) y el otro detecta el $NH_3$.
- Tiene una única fuente de alimentación de 5 $V$.

En la Figura 5.1, se observan los gases que el sensor MICS6814 puede detectar.

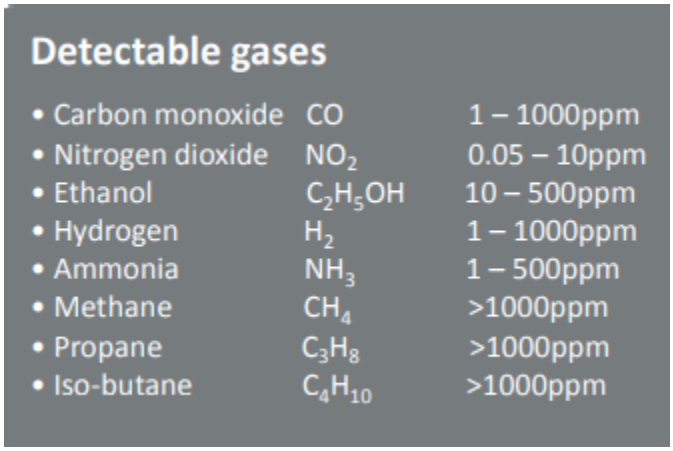



**Figura 5.1: Gases que puede detectar el sensor MICS6814**

Fuente: [23]

#### 5.2.1.2. Funcionamiento

El sensor trae tres chips de sensores con calentadores independientes y capas sensibles. Un chip sensor detecta los gases oxidantes ($OX$), el otro sensor detecta los gases reductores ($RED$) y el otro detecta el $NH_3$. Estos sensores necesitan un voltaje definido ($VH$) para alimentar la resistencia del calentador. La señal de salida ($V_o$) se lee entre la resistencia de la capa sensible $RS$ y una resistencia de carga [24]. El circuito mas simple utilizado para alimentar el sensor de gases es el que se ve en la Figura 5.2.

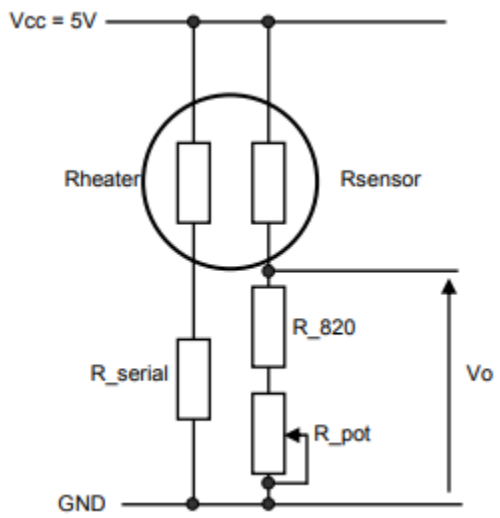



**Figura 5.2: Circuito utilizado para alimentar el sensor MICS6814**

Fuente: [24]

De la Figura 5.2 se tiene una resistencia R_serial usada para construir un divisor de voltaje con la resistencia del calentador (Rheater). Las dos resistencias en serie se alimentan con 5 $V$. La forma más simple de medir los cambios de resistencia del sensor es agregar una resistencia de carga al Rsensor y medir la caída de voltaje en esta resistencia cuando el puente (Rsensor + Rcarga) esta operando entre 5 voltios y GND.

La salida del sensor ($V_o$) se mide entonces con un indicador de salida analógica (voltímetro), un convertidor AD o un microcontrolador, por ejemplo. Para realizar mediciones fiables y estables a largo plazo es necesario limitar la cantidad de corriente que pasa a través de la capa de detección. Por esta razón se propone utilizar un circuito de carga de dos resistencias donde una primera resistencia de carga de 820 *Ohms* (R_820) está en serie con un potenciómetro de 500 *kOhms* (R_pot). R_820 es el valor mínimo para limitar la corriente del sensor.

Para la aplicación final en la que el rango de concentración de gas genera grandes cambios de resistencia, se aconseja utilizar un auto ajuste de la resistencia de carga para realizar la medición dentro de la zona de tensión optimizada.

Esta aplicación de ajuste se describe en la Figura 5.3 con el circuito de medición del ADC. Para adaptar la escala al rango de resistencia, el microcontrolador puede conmutar dos o tres resistencias en serie. El número de resistencias a conmutar varía con la precisión del ADC. Con un ADC de 10 bits, dos resistencias de conmutación pueden ser suficientes para cubrir un rango de resistencia de detección que se extiende, por ejemplo, de 2 *kOhms* a 1 *MOhms*.

Se deben tener en cuenta las siguientes recomendaciones para el uso del sensor:

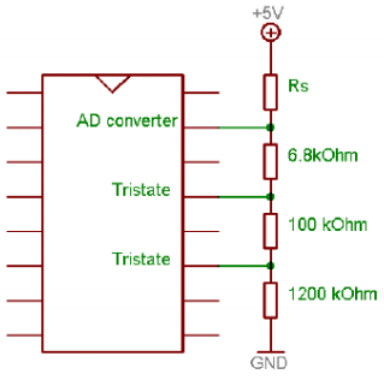



**Figura 5.3: Auto ajuste de la resistencia de carga para realizar la medición dentro de la zona de tensión optimizada**

Fuente: [24]

- Se debe soldar por reflujo en una atmósfera neutra, sin vapores de flujo de soldadura.
- No debe exponerse a altas concentraciones de disolventes orgánicos, vapores de silicona o humo de cigarrillos para evitar el envenenamiento de la capa sensible.
- El voltaje del calentador por encima del máximo especificado destruirá el sensor debido al sobrecalentamiento.
- Este sensor debe ser colocado en un paquete filtrado que lo proteja contra las proyecciones de agua y polvo [24].

### 5.2.2. Sensor material particulado DSM501A [25],[26]

El sensor de polvo DSM501 puede detectar humo de tabaco y polen, polvo doméstico, etc. después de calentarlo automáticamente al dispositivo de entrada de aire, la resistencia ajustable establece el rango del polvo. Usando el mismo principio del contador de partículas como base, para medir la unidad de volumen del número absoluto de partículas.

#### 5.2.2.1. Características

Este módulo está diseñado para detectar la partícula del tamaño mayor que un micrómetro, que generalmente incluye humo de cigarrillo, polvo doméstico, garrapatas, esporas, polen y moho.

Algunas de sus características son:

- Puede detectar humo de tabaco y polen, polvo doméstico, etc.
- Puede medir 1 micrón o más de partículas pequeñas.
- Tamaño pequeño, ligero, fácil de instalar.

- Circuito de entrada de 5 $V$ para procesamiento de señal
- Generador de flujo de aire incorporado, puede atraer la atmósfera externa.
- El mantenimiento es simple, puede mantener las características del sensor durante mucho tiempo[25].

En la Figura 5.4, se observan las dimensiones y aspecto físico del sensor:

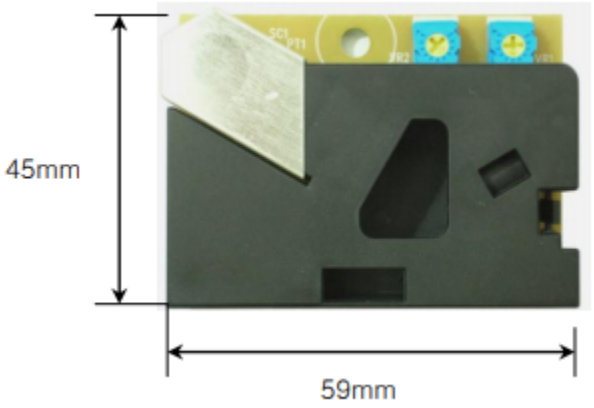



**Figura 5.4: Dimensiones del sensor DSM501A**

Fuente: [26]

#### 5.2.2.2. Funcionamiento

Este sensor detecta el nivel de polvo, partículas y polvo en el aire interior como el humo de cigarrillo, polvo domestico, garrapatas, esporas, polen y moho. La salida PWM del sensor se puede conectar directamente a $u - com$ para controlar la velocidad del ventilador o la ventilación y también para mostrar el nivel de partículas. El tamaño de las partículas detectable está limitado a un mínimo de 0.7 $um$, y la aplicación para la alarma de humo o incendio está prohibida . Se pueden detectar partículas finas de más de un micrón con alta sensibilidad debido a que el calentador interno induce la entrada de aire al módulo.

Este sensor tiene un calentador (resistencia) para generar calor, el calor crea una corriente ascendente de aire que atrae el aire exterior al módulo.Este sensor de polvo DSM501 debe instalarse verticalmente y mantenerse alejado de cualquier corriente artificial de aire por ventiladores, también debe haber ranuras cerca del módulo para que pueda entrar aire.

Se debe tener cuidado con la ubicación del sensor para evitar que partículas (como aceite, etc.) se adhieran a la parte óptica y causen un mal funcionamiento, al igual que la humedad, por ello se deben evitar lugares donde pueda producirse condensación con frecuencia.

La lente está recubierta con polímero antiestático y antipolvo. Pero para un mejor rendimiento, debe ser limpiado dependiendo de la condición. Limpieza cada seis meses para entornos de oficina y Se recomienda entornos industriales cada tres meses. Al limpiar, moje un lado del trapo con agua y frote la lente con ella y luego seque la lente con el otro extremo.

Este sensor cuenta con:

- Un contacto de control y dos contactos de salida.
- Lámpara de diodo emisor de luz (LED).
- Detector.
- Circuito amplificador de señal.
- Circuito de accionamiento de salida 1.
- Circuito de accionamiento de salida 2.
- Calentador inducido flujo de aire.

En la Figura 5.5 se aprecia el diagrama del sensor DSM501A.

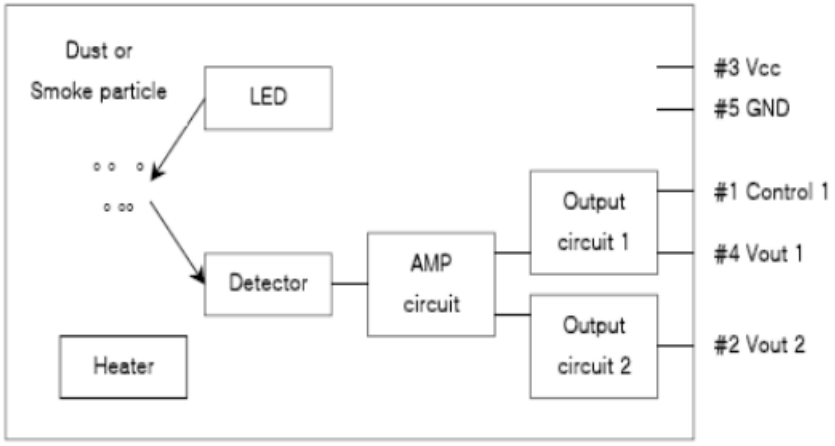



**Figura 5.5: Diagrama del sensor DSM501A**

Fuente: [26]

A continuación la descripción de los pines del sensor DSM501A:

- **Control (Pin # 1):**Este pin se usa para ajustar la sensibilidad cuando se usa Vout1.
- **Vout 2 (Pin # 2):** El Vout 2 es un puerto de salida estándar. Este puerto proporciona salida PWM para la densidad de partículas de más de 1 *um*.
- **Vcc (Pin # 3):** Suministro de energía positiva (DC 5V).

- **Vout 1 (Pin # 4):**Este pin se usa para ajustar el nivel de detección del tamaño mínimo de partícula. La sensibilidad de Vout 1 es más opaca que la de Vout 2 aproximadamente 2.5 veces. Este pin da una salida PWM.
- Agregando una resistencia entre Control (pin 1) y Tierra (pin # 5), el tamaño mínimo de las partículas se pueden ajustar de 1 m a 2.5 *um*.
- El tamaño mínimo estándar (abierto) de las partículas es de 2.5 *um*.
- **Tierra (Pin # 5):** Este pin se usa para tierra.

Su curva característica se aprecia en la Figura 5.6.

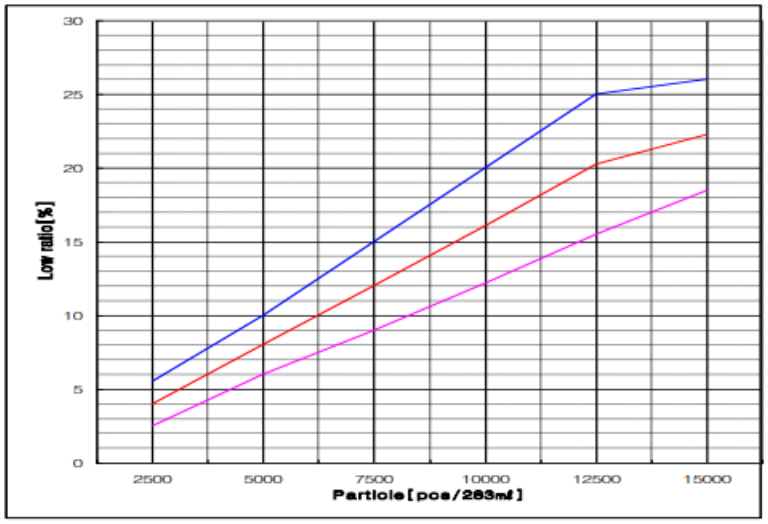



**Figura 5.6: Low ratio VS Particle**

Fuente: [26]

### 5.2.3. Sensor temperatura, presión y humedad BME280

El BME280 es un sensor ambiental con temperatura, presión barométrica y humedad. Este sensor puede ser usado tanto en I2C como con SPI. Este sensor de precisión es la mejor solución de bajo costo para medir la humedad con una precisión de $\pm 3\,\%$ , la presión barométrica con una precisión absoluta de $\pm$ 1 hPa, y la temperatura con una precisión de $\pm$ 1.0 grados centigrados. Debido a que la presión cambia con la altitud, y las medidas de presión son tan buenas, también puede usar como un altímetro con una precisión de $\pm$ 1 metro [27].

#### 5.2.3.1. Características

Algunas de las características principales son:

- Envase 2,5 *mm* x 2,5 *mm* x 0,93 *mm* tapa metálica LGA.
- Interfaz digital I2C (hasta 3,4 $MHz$) y SPI (3 y 4 hilos, hasta 10 $MHz$).
- Voltaje de suministro VDD rango de voltaje de suministro principal: 1,71 $V$ a 3,6 $V$.
- Rango de voltaje de la interfaz VDDIO: 1,2 $V$ a 3,6 $V$.
- Consumo de corriente 1.8 *uA* a 1 Hz de humedad y temperatura:
  - 2,8 *uA* a 1 Hz de presión y temperatura.
  - 3,6 *uA* a 1 Hz de humedad, presión y temperatura.
  - 0,1 *uA* en modo de suspensión.
- Rango de operación -40...+85 grados centígrados, 0...100 % de humedad relativa, 300...1100 *hPa*.
- El sensor de humedad y el sensor de presión pueden ser habilitados o deshabilitados independientemente.
- Tiempo de respuesta 1 *s*.
- Tolerancia de precisión $\pm 3$ %de humedad relativa.
- Histéresis $\pm 1$ % de humedad relativa.

#### 5.2.3.2. Funcionamiento

El BME280 es un sensor digital combinado de humedad, presión y temperatura basado en principios de detección. El módulo del sensor está alojado en un paquete LGA de tapa metálica extremadamente compacto con una huella de sólo 2,5  2,5 $mm^2$ y una altura de 0,93 *mm* [28].

Su salida se utiliza para la compensación de temperatura de los sensores de presión y humedad y puede también se utilizará para la estimación de la temperatura ambiente.

El sensor proporciona tanto interfaces SPI como I2C y puede ser suministrado usando 1,71 a 3,6 $V$ para la alimentación del sensor VDD y 1,2 a 3,6 $V$ para la alimentación de la interfaz VDDIO. Las mediciones pueden ser activadas por el anfitrión o realizadas a intervalos regulares. Cuando el sensor está desactivado, el consumo de corriente cae a 0,1 *uA*.

El BME280 puede funcionar en tres modos de potencia:

- **Modo de reposo:** sin funcionamiento, todos los registros accesibles, potencia mínima, seleccionado después del arranque.

- **Modo normal:** ciclo perpetuo de mediciones y períodos inactivos.
- **Modo forzado:** realizar una medición, almacenar los resultados y volver al modo de reposo.

A continuación en la Figura 5.7 se observa el diagrama de bloques del sensor de temperatura, presión y humedad.En la Figura 5.8 la apariencia física del sensor BME280.

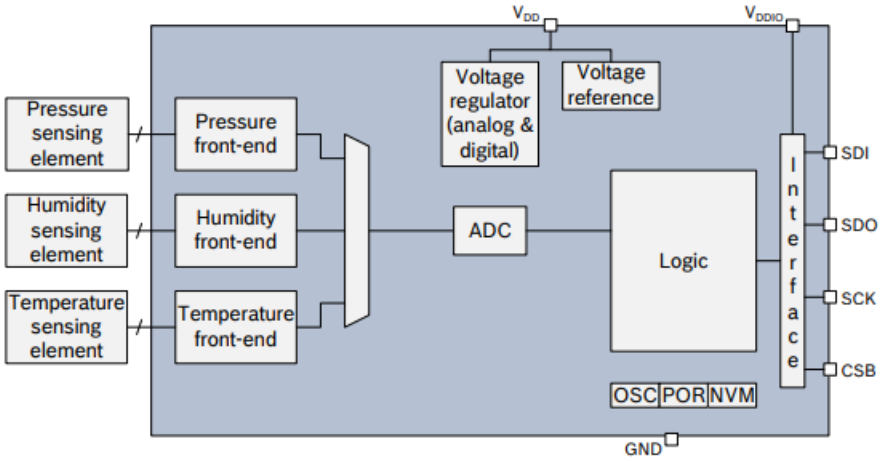



**Figura 5.7: Diagrama de bloques del sensor BME280**

Fuente: [28]

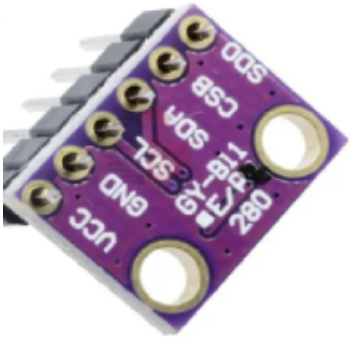



**Figura 5.8: Apariencia física del sensor BME280**

Fuente: [28]

## 5.3 Microcontrolador STM32F103C8

*Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como micro computadora. Se puede decir con toda propiedad que un microcontrolador es una micro computadora completa encapsulada en un circuito integrado* [29].

Estos están diseñados para interpretar y procesar datos e instrucciones en forma binaria. La forma de representar los datos, instrucciones y señales en forma de bits resulta difícil y tediosa para aquellos que no están familiarizados con el sistema de numeración binario, también llamado lenguaje de bajo nivel, debido a que las instrucciones no son propias del lenguaje humano (son ceros y unos). Por esta razón las instrucciones se llevan a cabo en un lenguaje de alto nivel, es decir, un lenguaje que utilice frases o palabras semejantes a las del lenguaje humano. Las sentencias de los lenguajes de alto nivel facilitan enormemente la programación ya que son familiares a nuestra manera de comunicarnos.

Todo programa escrito en un lenguaje de alto nivel debe ser transformado en código maquina.El compilador, traduce y transforma nuestro programa en código maquina, que es lo que realmente puede leer e interpretar el microcontrolador.

Una vez compilado el programa, es momento de transferir nuestro código maquina hacia la memoria interna del microcontrolador, usualmente hacia la ROM. Para esta tarea se utiliza un programador físico, que es una pieza de hardware que tiene el propósito de escribir el programa en la memoria interna del micro [29].

La familia de rendimiento de densidad media STM32F103xx incorpora el núcleo de alto rendimiento ARM Cortex-M3 RISC de 32 bits que funciona a una frecuencia de 72 $MHz$, memorias incrustadas de alta velocidad (memoria Flash de hasta 128 $KB$ y SRAM de hasta 20 $KB$). Todos los dispositivos ofrecen dos ADC de 12 bits, tres temporizadores de 16 bits de uso general y un temporizador PWM, así como interfaces de comunicación estándar y avanzadas: hasta dos I2C y SPI, tres USART, un USB y un CAN.

Los dispositivos funcionan con una fuente de alimentación de 2.0 a 3.6 $V$. Están disponibles tanto en el rango de temperatura de -40 a +85 grados centígrados como en el rango de temperatura ampliado de -40 a +105 grados centígrados.Un conjunto completo de modo de ahorro de energía permite el diseño de aplicaciones de baja potencia.

Estas características hacen que la familia de microcontroladores de línea de rendimiento de densidad media STM32F103xx sea adecuada para una amplia gama de aplicaciones, como accionamientos de motor, control de aplicaciones, equipos médicos y de mano, periféricos de PC y de juegos, plataformas GPS, aplicaciones industriales, PLC, inversores, impresoras, escáneres, sistemas de alarma, videoporteros y HVAC [30].

En la Figura 5.9 se puede observar la el aspecto físico y los pines con los que cuenta este microcontrolador.

Como se hablo anteriormente estos microcontroladores necesitan de un lenguaje de alto nivel para realizar ciertas instrucciones, para hacer uso de este lenguaje necesitamos un programa que ejecute

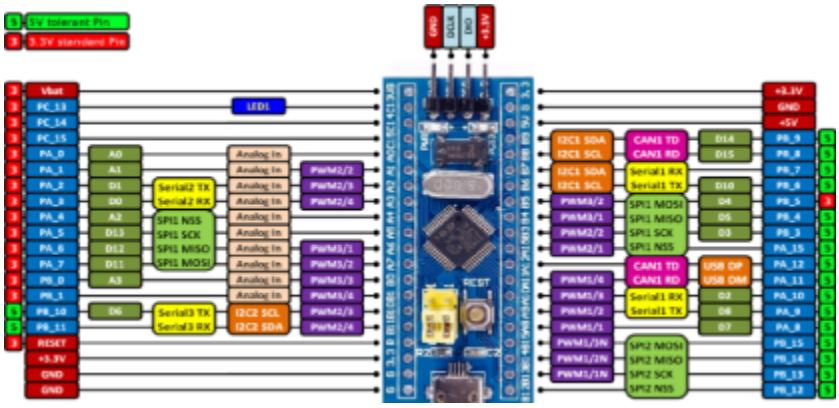


**Figura 5.9: Aspecto del microcontrolador STM32F103C8**

Fuente: [31]

los códigos y los introduzca al microcontrolador. Este programa se llama IDE, que significa Ïntegrated Development Environment"(Entorno de Desarrollo Integrado). Este IDE estará instalado en nuestro PC, es un entorno muy sencillo de usar y en él escribiremos el programa que queramos que el programa ejecute. Una vez escrito, lo cargaremos al microcontrolador y este comenzará a trabajar de forma autónoma [32].

Uno de los IDE compatibles con el microcontrolador STM32F103C8 es Arduino, el cual se puede instalar o subir en el microcontrolador a través de: ST-LINK V2, el conversor USB-TTL y esquema de patilla je [33].

## 5.4 Elementos alimentación del sistema

### 5.4.1. Módulo de carga y descarga de batería BM 3S

Uno de los módulos de Carga y descarga de baterías 18650 con 3 Celdas o 3S de 10 $A$ y 11.1 $V$ se encarga de proteger las baterías de Ion de Litio (Li-ion) o Lipo (Polimero de Litio) de cualquier sobrecarga o descarga mínima que pueda llegar a tener el banco de baterías [34].

El Modulo Cargador de Baterías 18650 para 3 Celdas incluye balanceador de carga y descarga equilibrado, lo que permite alargar la vida útil de las pilas llevando al máximo el rendimiento de las baterías. Esto quiere decir que el módulo BMS protege las baterías de una sobrecarga y una descarga, como ya se sabe la vida útil de nuestras baterías la podemos a largar si evitamos que nuestras baterías se descarguen por completo, es por ello que el BMS evita que se siga el consumo de energía de las baterías si su nivel de batería es muy bajo, además este modulo tiene un balanceador que permite que las baterías tengan todas la misma carga evitando que una batería se cargue más que otra evitando una sobrecarga [35].

El modulo BMS se conecta como se observa en la figura 5.10.

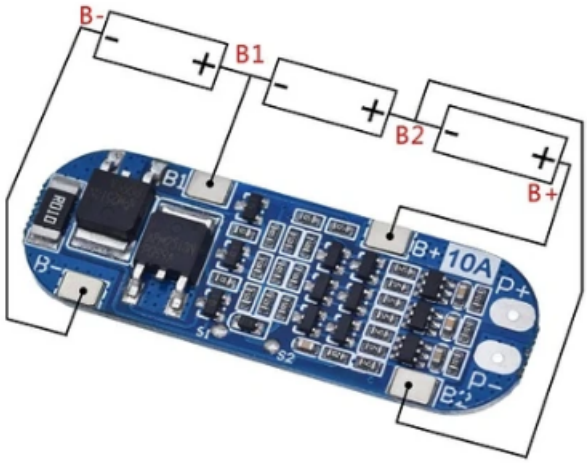



**Figura 5.10: Conexiones del BMS**

Fuente: [34]

Sus principales características son:

- Modelo: HX-3S-FL10*A*
- Modulo para 3 baterías Lipo o Litio 18650
- Voltaje Entrada de Carga: 12.6 $V$ - 13 $V$
- Voltaje de baterías a Cargar: 11.1 $V$ (Serie)
- Voltaje de cada celda: 3.7 $V$
- Corriente Máxima: 10 $A$
- Voltaje de sobre carga: 4.25 $V$ - 4.35 $V \pm 0.05V$ (por celda).
- Voltaje de Descarga: 2.5 $V$ - 3 $V \pm 0.05V$ (por celda).
- Protección contra cortocircuitos.
- Conector P+: Terminal positiva de carga y descarga.
- Conector P-: Terminal negativa de carga y descarga.
- Conector B+: Terminal de conexión al ánodo (Positivo) del banco de baterías.
- Conector B1: Terminal de conexión entre las baterías 1 y 2.
- Conector B2: Terminal de conexión entre las baterías 2 y 3.
- Conector B-: Terminal de conexión al cátodo (negativo) del banco de baterías.
- Dimensiones: 53 *mm* x 18 *mm* x 3.6 *mm*.

### 5.4.2. Reductor de voltaje LM2596 [36]

El LM2596 son circuitos integrados monolíticos que proporcionan todas las funciones activas para un regulador de conmutación de paso, capaz de conducir una carga de 3 $A$ con una excelente regulación de línea y carga. Estos dispositivos están disponibles en voltajes de salida fijos de 3,3 $V$, 5 $V$, 12 $V$, y una versión de salida ajustable [37].

Estos reguladores son fáciles de usar e incluyen una compensación de frecuencia interna y un oscilador de frecuencia fija. La serie LM2596 funciona a una frecuencia de conmutación de 150 $kHz$, lo que permite componentes de filtro de menor tamaño que los que se necesitarían con reguladores de conmutación de frecuencia más bajos.

Este circuito permite tener un voltaje regulado a partir de una fuente de alimentación con un voltaje mayor, por ejemplo si se tiene una fuente de 12 $V$ se puede regular a 5 $V$, 3.3 $V$, 2.2 $V$, etc, para el uso con microcontroladores, Arduino, PICs, Raspberry Pi, fuentes variables, drivers para leds, etc [38].

Sus principales características son:

- salida entre 1,5 y 35 $Vdc$
- Voltaje de entrada: 4.5-40 $V$
- Voltaje de salida: 1.5-35 $V$ (Ajustable)
- Corriente de salida: Maxima 3 $A$
- Dimensiones: 43*20*14 $mm$
- Frecuencia de switching: 150 $KHz$

En la figura 5.11 se ve el aspecto físico del módulo LM2596.

### 5.4.3. Módulo indicador de batería: E1022

Este módulo indicador es para batería de litio, su modo de uso es: conectar el puerto positivo y negativo de la pantalla con el puerto positivo y negativo de la batería, el tubo digital mostrará la cantidad de electricidad de la batería en tiempo real.

Parámetros de cantidad de electricidad, donde N es el numero de baterías:

- Cuando la tensión de la batería es superior a N * 3.3 $V$, se iluminará 1 bloque de electricidad (nota: N representa la cantidad de batería)
- Cuando la tensión de la batería es superior a N * 3.5 $V$, iluminará 2 bloques de electricidad




**Figura 5.11: Módulo LM2596**

Fuente: [38]

- Cuando la tensión de la batería es superior a N * 3.7 $V$, se iluminará 3 bloques de electricidad
- Cuando la tensión de la batería es superior a N * 3.9 $V$, iluminará 4 bloques de electricidad
- Cuando la tensión de la batería es menor que N * 3.3 $V$, la pantalla de 4 bloques estará apagada, lo que representa que la batería es menor que 3.3 $V$, y puede cargar la batería

En la figura 5.12 se observa el aspecto físico del indicador de batería.

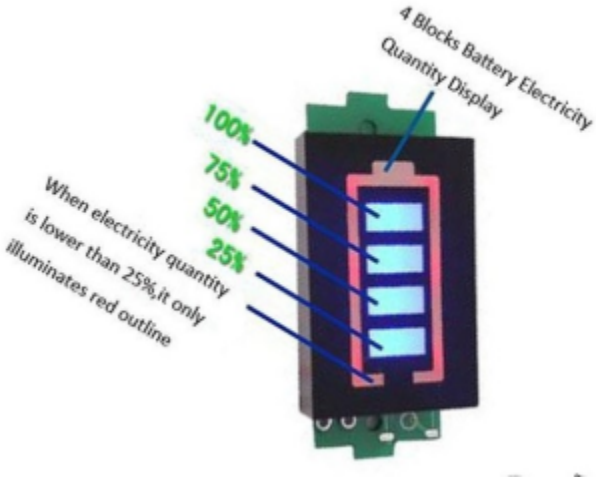



**Figura 5.12: Indicador de nivel de batería**

Fuente: [36]

## 5.5 Módulo de lectura y escritura de tarjeta SD [39]

Este módulo permite facilitar el uso de una tarjeta de memoria SD controlada con Arduino. A través de la programación, se puede leer y escribir en la tarjeta, ademas se puede utilizar como repro-

ductor de archivos de audio como MP3, control de MCU, ARM. La interfaz de comunicaciones es una interfaz estándar de SPI y sus pines de salida son: MOSI, SCK, MISO y CS.

Emplear una tarjeta SD o micro SD en con Arduino tiene la ventaja de proporcionar una memoria casi ilimitada para nuestros proyectos. Además es no volátil (es decir, resiste cuando se elimina la alimentación), y puede ser extraída y conectada a un ordenador con facilidad.

En la Figura 5.13 muestra la distribución de pines del modulo SD.




**Figura 5.13: Modulo SD.**

Fuente: [39]

## 5.6 Red de sensores inalámbricos

### 5.6.1. Definición

En la Red de sensores inalámbricos se define como *"una red de característica auto-configurable integrada por un pequeño número de nodos sensores también llamados motes distribuidos espacialmente y comunicados entre sí para lo que se usan señales de radio, con la finalidad de monitorizar y entender el mundo físico para controlar diversas condiciones en distintos puntos, entre ellas la temperatura, el sonido, la vibración, la presión y movimiento o los contaminantes"* [40].

### 5.6.2. Características [41]

Según la WSN tiene similitud con la red AD HOC que es inalámbrica y descentralizada , por lo cual comparte ciertas características, además de unas propias que se nombran a continuación:

- Aplicación específica.
- Interacción con el ambiente
- Energía (Alimentación no dependiente de la red eléctrica)
- Desplegado AD-HOC (Adaptación a los cambios de la red)

- Tolerancia a las fallas (La falla de un sensor no debe afectar a toda la red)
- Escalabilidad (No importa si se varía el tamaño de la red no debe verse afectada)
- Conectividad alta
- Control de Overhead (*"Debe haber compromiso entre la conservación de la energía, auto configuración, falla por nodo y latencia que puede existir"*)

### 5.6.3. Elementos [41]

Según la red de sensores inalámbricos está conformada por:

- Nodos de sensores.
- Tarjeta de acondicionamiento.
- Puerta de enlace.
- Bases de datos.
- Aplicaciones Cliente/Servidor.

### 5.6.4. Topología de red [41]

La topología de una WSN puede ser en malla, estrella o híbrida (malla y estrella), por lo cual primero se especifica la simbología de los elementos de la red, mostrada en la Figura 5.14.

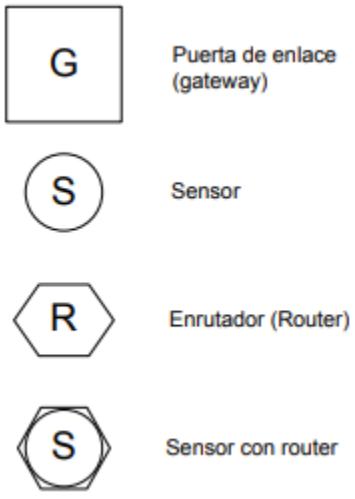



**Figura 5.14: Simbología red WSN**

Fuente: [41]

Los elementos de la Figura 5.14 se especifican a continuación:

- **Gateway:** Punto de unión que recoge los datos de la red.
- **Sensor:** elemento que convierte una señal física en una variable eléctrica.
- **Enrutador:** Direcciona los datos.
- **Sensor con Router:** Envía la información a una dirección específica sin intervención del Gateway

Teniendo en cuenta la simbología se muestran en en la Figura 5.15 las topologías ya mencionadas.

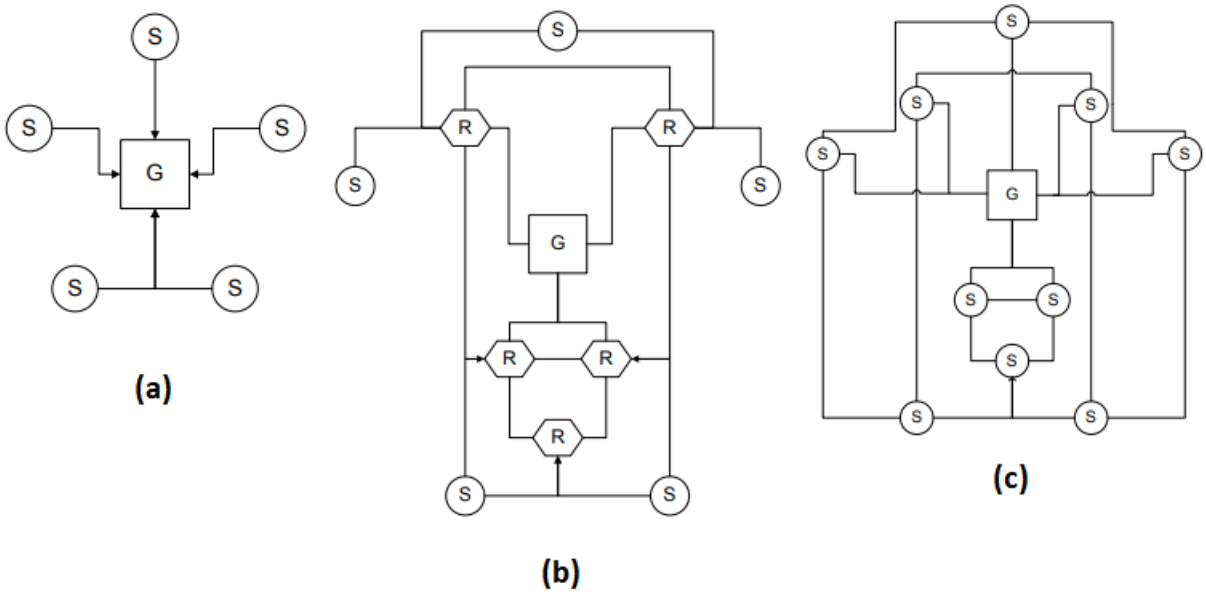



**Figura 5.15: Topologías de una red WSN: (a) Estrella (b) Malla (c) Híbrida**

Fuente: [41]

## 5.7 IoT

### 5.7.1. Definición [42]

Se basa en la interconexión de un producto con cualquier otro de su alrededor. El objetivo es hacer que todos estos dispositivos se comuniquen entre sí y, por consiguiente, sean más inteligentes e independientes.

Es necesario el empleo del protocolo IPv6 y el desarrollo de numerosas tecnologías que actualmente están siendo diseñadas por las principales compañías del sector. Actualmente grandes multinacionales como Cisco y Microsoft desarrollan y ofrecen múltiples aplicaciones como las siguientes:

- Soluciones de transporte inteligentes acelerando los flujos de tráfico, reduciendo el consumo de combustible, teniendo como prioridad salvar vidas.

- Redes eléctricas inteligentes se conectan de manera más eficiente los recursos renovables, mejorando la fiabilidad del sistema.
- Sensores de control en máquinas de diagnóstico, con el fin de predecir problemas.
- Sistemas basados en datos en la infraestructura de çiudades inteligentes", así es más fácil para los municipios para ejecutar la gestión de residuos, la policía y otros programas de manera más eficiente.
- Sistemas que permitan monitorear variables ambientales ("Internet of Things (IoT) — Microsoft," n.d.), ("Internet de las cosas (IoT) - Cisco," n.d.).

### 5.7.2. Características [43]

Una solución de IoT cuenta con una arquitectura que incluye los aspectos físicos (las cosas) y los aspectos virtuales (como los servicios y los protocolos de comunicación), por ello adoptar una arquitectura con múltiples niveles permite concentrarse en mejorar la comprensión acerca de cómo todos los aspectos más importantes de la arquitectura funcionan antes de integrarlos dentro de la aplicación de IoT. Este enfoque modular le ayuda a gestionar la complejidad de las soluciones IoT. Por ejemplo, a continuación se explica un modelo IoT con tres capas.

- **Capa de dispositivos**: incluye sensores físicos y actuadores que están conectados a dispositivos de IoT y los propios dispositivos de IoT. Los sensores y los actuadores normalmente no se consideran por si mismos dispositivos «inteligentes», pero los sensores y los actuadores a menudo se conectan directamente o de forma inalámbrica, a través de tecnologías como Bluetooth LE o ZigBee, a dispositivos de IoT que tienen más capacidades de procesamiento.
- **Capa Edge:** se muestra como el nivel medio capas de una arquitectura de IoT, se relaciona con servicios de analítica y de preprocesado que se ubican en el límite de la red. La analítica Edge ocurre en tiempo real (o casi en tiempo real) al procesar el flujo de datos en el punto en el que los datos se recopilan según llegan desde los sensores. Las tareas básicas de preprocesamiento, como el filtrado y la agregación de datos, se ejecutan en el límite y luego los datos principales preprocesados se transfieren en sentido ascendente hacia los servicios y aplicaciones de la nube para procesarlos y analizarlos aún más.
- **Capa de nube:** Después de preparar los datos, estos se envían en sentido ascendente para procesarlos aún más, almacenarlos y utilizarlos dentro de las aplicaciones de la nube, en la capa de la nube. Las aplicaciones en la nube que realizan el procesado de datos, a menudo se complementan con aplicaciones móviles y con aplicaciones de clientes basadas en la web, que presentan los datos a los usuarios finales que brindan acceso a herramientas para explorar y analizar más a fondo, a través de paneles de instrumentos y de visualizaciones.

## 5.8 Tecnología y protocolo de red

### 5.8.1. LoRa

#### 5.8.1.1. Definición

LORA se deriva de “Long Range” en español Largo Alcance, es una tecnología que pertenece a Semtech y hace referencia a la Capa 1 del modelo OSI, es decir a la capa física, por lo cual a LORA se le conoce como una técnica de modulación y recepción de datos.

La red LORA es inalámbrica ideal para dispositivos de bajo consumo y baja tasa de bits (bitrate), por lo cual es muy utilizada en proyectos de IoT. Puede implementarse con topología estrella o malla, y no tiene funciones de seguridad ya que sólo garantiza la transmisión de datos [44].

#### 5.8.1.2. Características

A continuación, se muestran algunas de las principales características de la Red LORA.

- Utiliza una modulación de radio basada en Chirp Spread Spectrum (CSS), utiliza la banda de radio industriales, científicas y médicas (ISM), es decir por debajo por 1 GHz.
- Generalmente se implementa una topología en estrella donde los nodos envían la información a un Gateway que se encarga de transmitir los datos a un servidor para luego ser procesados de acuerdo a la aplicación requerida. Si se implementa la red en topología malla ya no se habla de una red LPWAN, es decir de baja potencia, debido a que ya que todos los nodos están interconectados deben estar encendidos permanentemente lo que conlleva a alto consumo de energía.
- Puede configurarse la tasa de bit (bitrate), alcanzando un máximo de 37.5 $Kbps$.
- El rango de alcance en se aproximadamente 2 Km en zonas urbanas y de hasta 10 Km en zonas rurales donde se presenta menor interferencia, debido a las pocas construcciones.
- Es una red de bajo costo.
- Existen 3 parámetros que pueden variar en la modulación LORA según el fabricante como los son el ancho de banda (BW), el factor de dispersión (SF) y Code Rate (CR). Con estos tres parámetros se determina la bitrate, la inmunidad a la interferencia y el tiempo en el cual el canal está ocupado.
- La sensibilidad del receptor también se ve afectada por los parámetros BW y SF. La sensibilidad es inversamente proporcional al BW y es proporcional al SF.

#### 5.8.1.3. Parámetros de red [45], [46]

A continuación se describen los parámetros independientes configurables, que según sus combinaciones afectan el alcance o distancia máxima de recepción de datos, la velocidad de los datos, sobrecarga por corrección o detección de errores en la información transmitida y el consumo energético al transmitir y recibir datos [47].

- **Ancho de banda (BW):** Indica el ancho de frecuencia que usa la señal, entre mayor sea permite el uso de una velocidad de datos efectiva más alta, reduciendo así el tiempo de transmisión, sin embargo, la sensibilidad de recepción empeora. Existen restricciones regulatorias en la mayoría de los países sobre el ancho de banda ocupado permitido. El ancho de banda en LoRa se refiere al ancho de banda de doble banda lateral (o ancho de banda total del canal), como se ve en la figura 5.16, donde además se muestran los tres anchos de banda configurables en LoRa como lo son 125 $kHz$, 250 $kHz$ y 500 $kHz$.

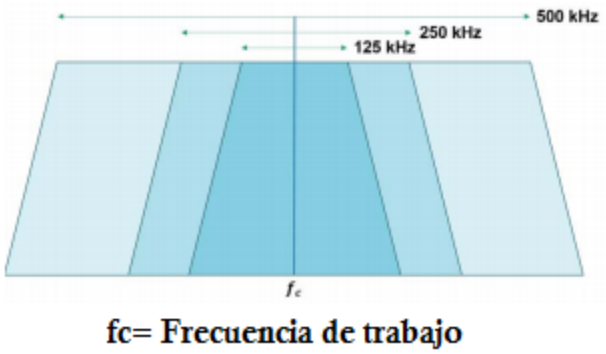



**Figura 5.16: Ancho de Banda de doble banda lateral manejados en LoRa**

Fuente: [45]

- **Factor de difusión (SF):** La modulación LoRa de amplio espectro se realiza representando cada bit de información de carga útil mediante múltiples chips de información (este término se utiliza para distinguir los bits codificados más cortos, de los bits no codificados más largos de la señal de información). La velocidad a la que se envía la información de propagación se conoce como la velocidad de símbolos (Rs), la relación entre la velocidad de símbolos nominal y la velocidad de chip es el factor de difusión y representa el número de símbolos enviados por bit de información.

  El factor de difusión es el encargado de configurar el alcance que puede existir en un enlace entre transmisor y receptor, este factor puede ser un número de 7 a 12, y entre mayor sea este, mayor será la sensibilidad en el receptor lo que aumenta el alcance, pero a la vez necesita de un mayor número de chirridos de subida (upchirp) para asegurar el enlace, lo que aumenta el tamaño del preámbulo y disminuye el tamaño de carga útil a transportar.

  Tenga en cuenta que el factor de propagación, debe conocerse de antemano en ambos lados del enlace de transmisión y recepción, ya que los diferentes factores de propagación son ortogonales

entre sí. Además, hay que tener en cuenta la relación señal a ruido resultante (SNR) requerida en la entrada del receptor. En la tabla 5.8 se muestran los diferentes factores de difusión con la relación de chips por símbolo y la relación señal a ruido (SNR) aceptada por el receptor donde la señal es aún identificable.

**Tabla 5.8: Rangos del factor de difusión**

| SF | (Chips / symbol) | LoRa Demodulador SNR |
|---|---|---|
| 6 | 64 | -5 dB |
| 7 | 128 | -7.5 dB |
| 8 | 256 | -10 dB |
| 9 | 512 | -12.5 dB |
| 10 | 1024 | -15 dB |
| 11 | 2048 | -17.5 dB |
| 12 | 4096 | -20 dB |

Fuente: [46]

- **Tasa de codificación (CR):** con la tecnología LoRa a usar y para dar más robustez del enlace, se emplea una codificación de error cíclico para realizar la detección y corrección de errores hacia adelante. Tal codificación de error incurre en una sobrecarga de transmisión como se muestra en la tabla 5.9.

**Tabla 5.9: Sobrecarga de codificación cíclica**

| CR | Sobrecarga en la transmisión |
|---|---|
| 4/5 | 1.25 |
| 4/6 | 1.5 |
| 4/7 | 1.75 |
| 4/8 | 2 |

Fuente: [46]

Además de los parámetros ya expuestos hay otros que se ven afectados de acuerdo a la elección de los anteriores.

- **Indicador de fuerza de la señal recibida (RSSI):** El el valor de potencial de la señal recibida en dBm, en donde 0 bBm equivale a 1 mW que correspondería a una señal ideal, entra más

pequeño sea el valor menor será la potencia. En la figura 5.17 se muestra cómo se considera la señal si débil o fuerte según el valor de RSSI.




**Figura 5.17: Escala RSSI para clasificación de la señal recibida**

Fuente: [48]

- **Relación señal a ruido (SNR):** Es la diferencia entre la potencia de la señal de recibida y la potencia del ruido que la interfiere, se mide en decibelios (dB). Cuando la SNR es positiva significa que la señal tiene mayor potencia del ruido y si es negativa que la señal se encuentra por debajo del nivel del ruido. LoRa puede trabajar por debajo del límite del ruido pudiendo llegar a una SNR de -20 *dB* sin perder la información de la señal. El rango de operación de LoRa está entre aproximadamente -20 *dB* y 10 *dB*.

- **Tiempo de transmisión:** Es el tiempo que tarda en llegar la información desde el nodo final al Gateway, está fuertemente relacionado con el factor de difusión, que entre mayor sea, mayor será el tiempo en que el mensaje estará en el aire, pero menor será la cantidad de datos que se puedan enviar [48].

En la figura 5.18 se muestra una figura que resume la relación entre algunos de los parámetros nombrados.

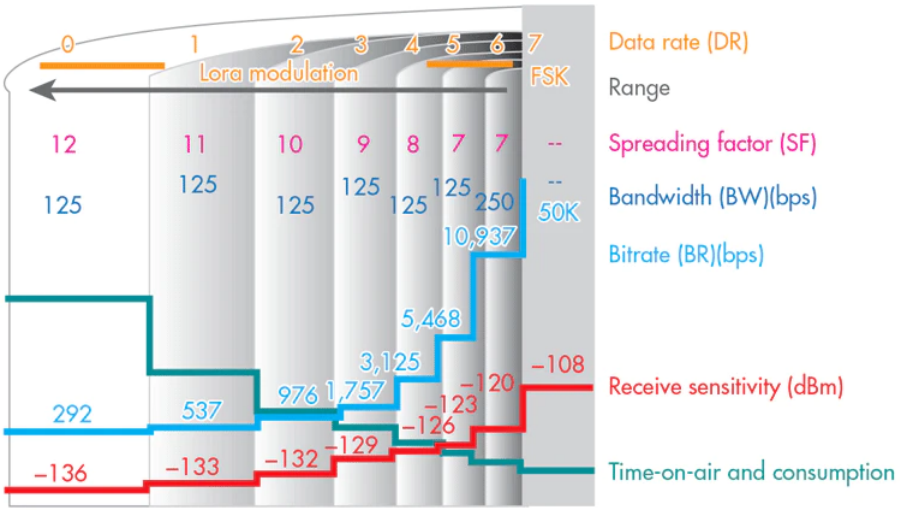



**Figura 5.18: Relación entre los parámetros de red**

Fuente: [49]

### 5.8.2. LORAWAN

#### 5.8.2.1. Definición

LORAWAN es *"un protocolo de red que hace uso de la técnica de modulación LORA para comunicar y administrar dispositivos, y se compone principalmente de dos partes: gateways y nodos, los primeros son los encargados de recibir, procesar y coordinar la recepción de la información a los nodos y los segundos son los dispositivos finales que envían y reciben información hacia el Gateway" [49].*

#### 5.8.2.2. Características

A continuación, se muestran algunas de las principales características de la Red LORAWAN.

- Utiliza topología estrella y puede tener uno o varios Gateway que forman una estrella con sus nodos respectivos.
- Este protocolo si cuenta con aplicación de seguridad que garantiza autenticidad, integridad y confidencialidad de paquetes.
- El protocolo está diseñado para implementar dispositivos alimentados por baterías, ubicados en lugares fijo o que sean móviles.
- El alcance de esta red se encuentra entre 10 a 15 *Km* con línea de vista
- Se utilizan varios factores de dispersión (SF) en diferentes canales de frecuencia de distintos anchos de banda
- Para optimizar la utilización de batería y la capacidad de la red se cuenta con un bilrate variable, además que se puede controlar la potencia entregada a cada uno de los nodos. Lo que se denomina como Adaptative Data Red (ADR).

#### 5.8.2.3. Arquitectura de la red [50]

Una red con el protocolo LORAWAN generalmente tiene una topología estrella, donde el gateway es el centro y los nodos finales las puntas. Además de los elementos mencionados la red está conformada por el servidor de y el servidor de aplicación. Pueden haber múltiples gateways en una red como se muestra en la figura 5.19.

Específicamente el nodo final es quien toma las medidas de las variables del medio y envía dicha información codificada al Gateway, este último recibe la información y la envía al servidor de red que se encarga de decodificar los datos, gestionar la red filtrando los paquetes redundantes recibidos, realizar comprobaciones de seguridad, programar confirmaciones a través del Gateway, realizar adaptación a la velocidad de datos y por último enviar la información correcta al servidor de aplicación, donde se

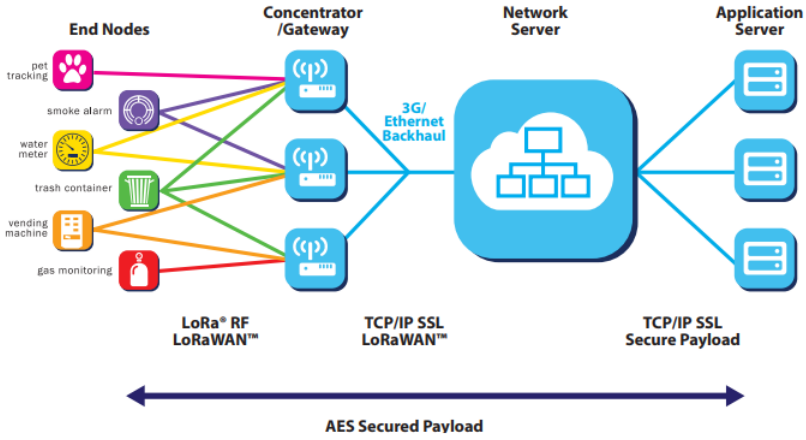



**Figura 5.19: Arquitectura LORAWAN**

Fuente: [51]

interpretan y manipulan los datos para mejor entendimiento del usuario final de la red.

#### 5.8.2.4. Mensajes de enlace descendente y ascendente [52]

Cuando los mensajes se envían desde el nodo final al gateway se conocen como mensajes de enlace ascendente o uplink, mientras que si es al contrario, del gateway al nodo final son mensajes de enlace descendentes o downlink, como se muestra en la Figura 5.20. Hay que tener en cuenta que el gateway puede recibir hasta 8 mensajes de enlace ascendente dependiendo del número de canales que tenga gracias a las sub-bandas de LORAWAN, pero cuando envía un mensaje de enlace descendente no puede recibir nada, por lo cual este último tipo de mensajes están restringidos.

En cada país hay una lista de definiciones de planes de frecuencia donde se define una frecuencia base de trabajo la cual debe tenerse en cuenta para comprar los módulos y dispositivos para la red, en Colombia esta frecuencia es de 915 $MHz$ como en los Estados Unidos. La banda de trabajo es desde los 902 a 928 $MHz$, en la tabla 5.10 se especifican las frecuencias, factor de difusión y ancho de banda para el envío de mensajes en enlace ascendente y descendente, respectivamente.

**Tabla 5.10: Frecuencias Sub-bandas LoRaWAN en América**

| FRECUENCIAS UPLINK | | FRECUENCIAS DOWNLINK | |
|---|---|---|---|
| **Frecuencia** | **SF y BW** | **Frecuencia** | **SF y BW** |
| 903.9 | SF7,BW125 a SF10,BW125 | 923.3 | SF7,BW500 a SF12,BW500 |
| 904.1 | SF7,BW125 a SF10,BW125 | 923.9 | SF7,BW500 a SF12,BW500 |
| 904.3 | SF7,BW125 a SF10,BW125 | 924.5 | SF7,BW500 a SF12,BW500 |

Continúa en la página siguiente

| FRECUENCIAS UPLINK | | FRECUENCIAS DOWNLINK | |
|---|---|---|---|
| **Frecuencia** | **SF y BW** | **Frecuencia** | **SF y BW** |
| 904.5 | SF7,BW125 a SF10,BW125 | 925.1 | SF7,BW500 a SF12,BW500 |
| 904.7 | SF7,BW125 a SF10,BW125 | 925.7 | SF7,BW500 a SF12,BW500 |
| 904.9 | SF7,BW125 a SF10,BW125 | 926.3 | SF7,BW500 a SF12,BW500 |
| 905.1 | SF7,BW125 a SF10,BW125 | 926.9 | SF7,BW500 a SF12,BW500 |
| 905.3 | SF7,BW125 a SF10,BW125 | 927.5 | SF7,BW500 a SF12,BW500 |
| 904.6 | SF7,BW125 a SF10,BW125 | | |

Fuente: [52]

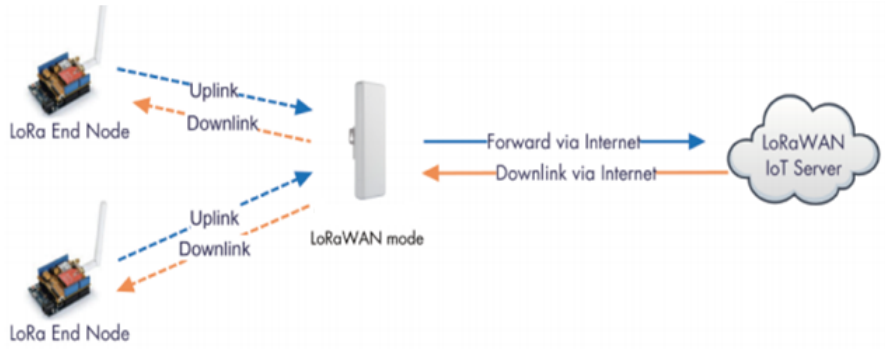



**Figura 5.20: Mensajes Uplink y Downlink**

Fuente: [53]

#### 5.8.2.5. Clases de nodo [40]

LORAWAN cuenta con tres tipos de nodo que se describen enseguida.

- **Nodo Clase A:** Es un nodo bidireccional que consume la menor cantidad de energía, debido que sólo hay comunicación si el nodo envía primero un dato al Gateway, por cada transmisión se abren dos ventanas de recepción por si el gateway envía confirmación u otro tipo de respuesta como se ve en la figura 5.21.
- **Nodo Clase B:** Es un nodo bidireccional con ventanas de tiempos de recepción predeterminados con el Gateway, no necesariamente después de una transmisión a diferencia del nodo A, pueden o no utilizar baterías externas dependiendo del tiempo programado como se ve en la figura 5.22.
- **Nodo Clase C:** Son los nodos de mayor consumo de energía ya que están permanentemente encendidos para recibir información, sólo cierran la recepción cuando hay una transmisión, como se ve en la figura 5.23. Como ventaja tienen menor suma de retardos (latencia) entre el nodo y el Gateway.

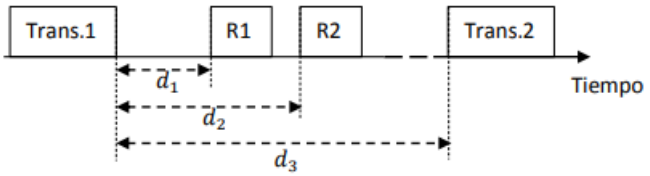



**Figura 5.21: Nodo Tipo A**

Fuente: [40]

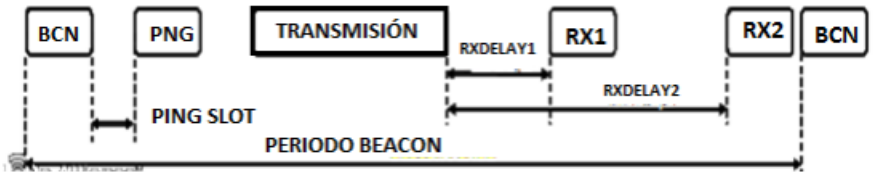



**Figura 5.22: Nodo Tipo B**

Fuente: [40]

#### 5.8.2.6. Formas de conectarse a una red lorawan y seguridad [40]

Existen dos formas de conectarse a una red LORAWAN:

- **Activation By Personalization (ABP):** En esta modalidad se conocen tres parámetros como lo son

  - **DevAdress:** se refiere a la dirección del nodo final, es equivalente a una dirección IP, por lo cual es única para cada nodo. Dirección de 32 bits la cual permite a la red interpretar los datos y usar las claves de encriptación correctamente.
  - **NetworkSessionKey:** es una clave única de encriptación AES (Advanced Encryption Standard) de 128 bits, proporciona integridad de los mensajes y seguridad para la comunicación entre el nodo y el servidor de la red.
  - **ApplicationSessionKey** es otra clave de encroptación AES de 128 pero entre el dispositivo y el servidor de red a través de la aplicación.

En esta forma de conectarse primero el nodo envía los datos al Gateway, este último, valida que los datos correspondan a la sesión, si es así los procesa sino los rechaza.

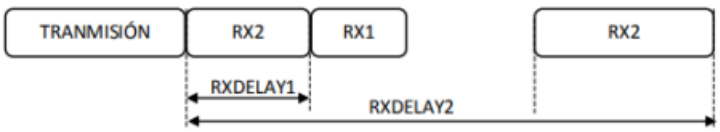



**Figura 5.23: Nodo Tipo C**

Fuente: [40]

- **Over The Air Activation (OTAA):** Es más segura y por ello la preferida, tiene 3 parámetros de configuración:

  - **DevEUI:** Identificador de fábrica único
  - **AppEUI:** Identificador de aplicación que clasifica los dispositivos por aplicación
  - **AppKey:** Clave secreta AES de 128 bits, la conoce el dispositivo y la red.

  Para conectarse con esta modalidad primero el nodo solicita un inicio de sesión (JOIN) a la red con los datos de configuración lo que abre la ventana de recepción, luego el Gateway recibe la solicitud y la envía al servidor, quien verifica que el nodo está encendido y la clave, si es correcto el servidor le contesta al Gateway y ente al nodo que empieza a transmitir, sino es correcta rechaza la solicitud.

#### 5.8.2.7. Estructura de los paquetes LORAWAN

La tecnología utilizada emplea dos tipos de formato de paquete, explícito e implícito. El paquete explícito incluye un encabezado corto que contiene información sobre el número de bytes, la velocidad de codificación y si se utiliza un CRC (Código de Redundancia cíclica) en el paquete para detectar errores, el paquete implícito elimina el encabezado porque no varía en la red, ahorrando bytes. El formato del paquete se muestra en la figura 5.24 [46].

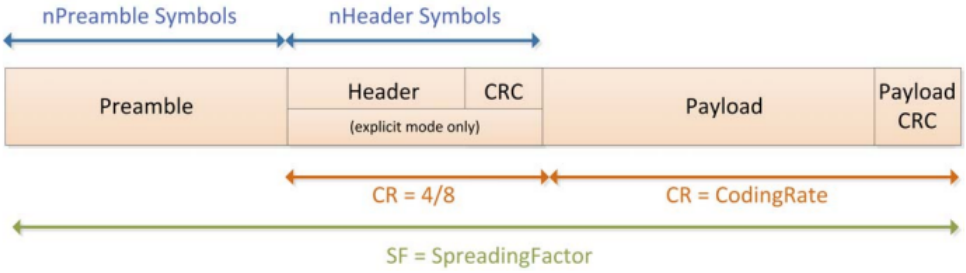



**Figura 5.24: Estructura paquete LoRaWAN**

Fuente: [46]

El preámbulo se usa para sincronizar el receptor con el flujo de datos entrantes, su longitud es de mínimo 4.25 símbolos, pero generalmente es de 12 símbolos; con CR=4/8, Header y CRC es de 4 *bytes*, este tramo sólo se usa si el paquete es explícito corrigiendo errores del preámbulo; el payload o carga útil es máximo de 255 bytes y es el que contiene el mensaje y por último el payload CRC es la parte de la trama que se usa para verificar errores en el mensaje principal.

### 5.8.3. Módulo LORABEE [54]

Para el proyecto se decidió utilizar el módulo LORABEE de Dragino que contiene al transceptor RM95W y trabaja a una frecuencia de 915 $MHz$, utiliza el chip SX1276/SX1278, que permite al usuario enviar datos y proporciona una comunicación de espectro extendido de rango ultra largo y una alta

inmunidad a interferencias mientras minimiza el consumo de corriente. Lo mejor de esta versión es que tiene un conector SMA para la antena, lo que lo hace más robusto, ya que normalmente esta antena es hecha con cable y soldada al circuito, como se compara en la figura 5.25. Hay que tener en cuenta que este módulo es sensible a corrientes estáticas, por lo cual para su manipulación se recomienda el uso de guantes o manilla antiestática.

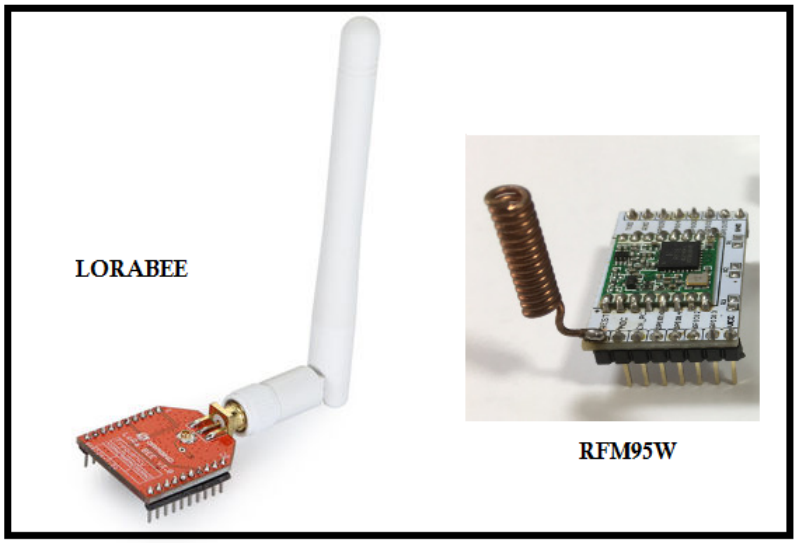



**Figura 5.25: Comparación antena de LORABEE y RFM95W convencional**

Fuente: [54], [55]

Algunas de las características más importantes del módulo LORABEE se nombran a continuación:

- 168 *dB* de presupuesto de enlace máximo.
- +20 *dBm* - 100 *mW* de salida de RF constante vs. +14 *dBm* de alta eficiencia PA.
- Velocidad de bits programable de hasta 300 *kbps*.
- Alta sensibilidad: hasta -148 *dBm*.
- Excelente inmunidad de bloqueo.
- Baja corriente RX de 10.3 *mA*, retención de registro de 200 nA.
- Modulación FSK, GFSK, MSK, GMSK, LoRaTM y OOK.
- Sincronizador de bits incorporado para recuperación de reloj.
- Motor de paquetes de hasta 256 bytes con CRC.

Para conectarse con el microcontrolador el LORABEE utiliza el protocolo de comunicación SPI y en el nodo final tiene como función enviar los datos recogidos por los sensores al gateway.

### 5.8.4. Gateway LG02 [56]

Debido a los altos costos de los Gateway LORAWAN en el mercado se decidió comprar el Gateway de Dragino LG02 que tiene 2 canales de recepción y uno de transmisión, por lo que hay limitaciones al usar LORAWAN ya que no permite saltos de frecuencia sino que trabaja a una frecuencia única de recepción en ambos canales y sólo admite nodos tipo ABP; sin embargo, para lo requerido en la red donde se van a conectar sólo dos nodos sus limitaciones no impiden la realización del proyecto. EL gateway elegido se muestra en la figura 5.26, al igual que algunas de sus características.

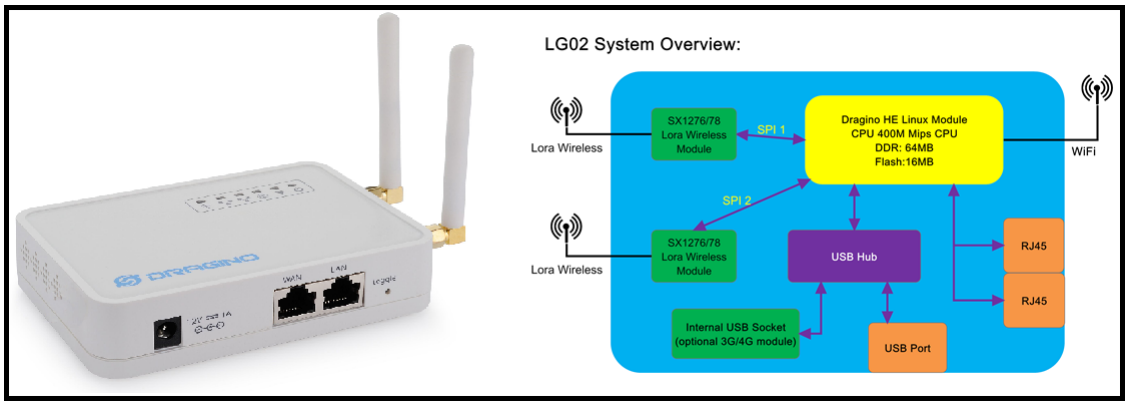



**Figura 5.26: Gateway LG02**

Fuente: [56]

Algunas de las características más importantes del gateway LG02 se nombran a continuación:

- Sistema de código abierto OpenWrt
- Bajo consumo de energía
- Actualización de firmware vía web
- Software actualizable a través de la red
- Protocolo flexible para conectarse a servidores IoT
- Servidor web incorporado
- Administrado por Web GUI, SSH a través de LAN o WiFi
- Conexión a Internet a través de LAN, WiFi, o 3G / 4G a través del módulo LTE opcional
- El diseño a prueba de fallos proporciona un sistema robusto
- 2 x módulos LoRa SX1276 / SX1278
- Transceptor LoRa full-duplex
- Dos canales de recepción y un canal de transmisión.

- Soporte limitado en LoRaWAN / Protocolo de soporte privado LoRa
- Admite hasta 300 nodos
- Banda LoRa disponible a 433/868/915/920 $MHz$
- Alcance máximo en LoRa: 5 a 10 $km$. Área de densidad: mayor a 500 $m$
- Excepto en el modo LoRaWAN, LG02 puede admitir modos de funcionamiento mutuo como: modo repetidor LoRa, modo MQTT, modo Cliente TCP /IP, modo Servidor TCP/IP para adaptarse a diferentes requisitos para la conexión IoT.

### 5.8.5. The Things Network (TTN) [57]

The Things Network es una red descentralizada y colaborativa, que cuenta con miles de puertas de enlace en todo el mundo, utilizada por decenas de miles de desarrolladores y empresas para crear casos de uso que generen valor. Se pueden usar puertas de enlace que ya están instaladas o agregar puertas de enlace adicionales donde se necesite cobertura adicional.

TTN en este proyecto funciona como el servidor de red encargándose de decodificar los datos, realizar comprobaciones de seguridad, programar confirmaciones a través del Gateway y por último enviar la información correcta al servidor de aplicación por medio de integraciones.

En este servidor de red pueden visualizarse los mensajes de enlace ascendente y descendente pero no los almacenarse por eso es necesario el servidor de aplicación, por lo cual TTN se puede integrar con varias plataformas con las que tiene convenio y lo datos pueden ser enviados a cualquier página web por HTTP. En la figura 5.27 se resumen las utilidades de esta plataforma. Además de los datos de los sensores también permite visualizar la información sobre los parámetros de la red.

Para utilizar TTN hay que tener en cuenta que tiene una Política de acceso justo nombrada en [58] que limita los datos que cada dispositivo final puede enviar, al permitir:

- Un promedio de 30 segundos de tiempo de enlace ascendente en el aire, por 24 horas, por dispositivo.
- Como máximo 10 mensajes de enlace descendente por 24 horas, incluidos los ACK para enlaces ascendentes confirmados.

En este tipo de servicio un buen objetivo es mantener el payload o carga útil de la aplicación por debajo de 12 bytes y el intervalo entre mensajes al menos varios minutos. También se debe tener en cuenta que el protocolo LoRaWAN agrega al menos 13 bytes al payload de la aplicación. Por ejemplo:

- Algunas bibliotecas, como LMIC, sólo admite enviar 51 bytes para todas las velocidades de datos.

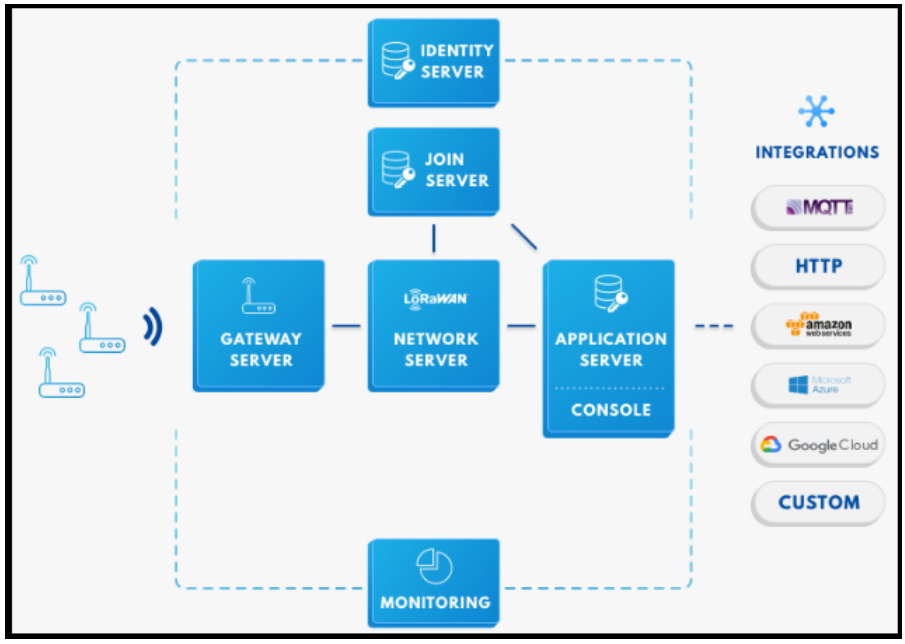



**Figura 5.27: Utilidades The Things Network**

Fuente: [57]

- Algunos proveedores, como el KPN holandés, solo admite enviar 51 bytes para todas las velocidades de datos.
- Algunas regiones definen un tiempo de permanencia máximo, reduciendo el tamaño máximo de carga útil para velocidades de datos lentas (SF alta).

## 5.9 Servidores IoT

### 5.9.1. ThingSpeak [59]

*ThingSpeak es un servicio de plataforma de análisis de IoT que permite agregar, visualizar y analizar flujos de datos en vivo en la nube. ThingSpeak proporciona visualizaciones instantáneas de los datos publicados por sus dispositivos. Con la capacidad de ejecutar código MATLAB en ThingSpeak, puede realizar análisis y procesamiento en línea de los datos a medida que ingresan, a menudo se usa para prototipos y sistemas de prueba de concepto de IoT que requieren análisis.*

Para utilizar ThingSpeak se tienen varios tipos de licencias en las cuales varía la cantidad de canales que se pueden habilitar, cada canal tiene 8 campos que reciben los datos en tiempo real, además de 3 datos adicionales sobre la ubicación de los nodos, como lo son longitud, latitud y altura, sin embargo, si la ubicación es fija se configura directamente en el canal. Depende de los campos que se habiliten, ThingSpeak crea gráficas para cada uno donde se puede ver el dato y la hora a la que se envió, cada canal cuenta con una pagina WEB que puede ser pública y/o privada, si es pública cualquier usuario con el link puede ver el flujo de datos en estas gráficas y descargar los datos más recientes, pero no puede editar.

Al utilizar la licencia gratuita se pueden usar hasta 4 canales, es decir, en una red con 4 nodos, además con esta licencia se tiene un límite de 8200 mensajes por día y 3'000.000 de mensajes por año que es lo que dura habilitada la licencia.

En el proyecto se decidió utilizar este servidor IoT por su facilidad de integración con TTN y por la oportunidad de realizar distintos tipos de gráficos utilizando código de MATLAB, además que puede crear reacciones como alertas por correo electrónico. En la figura 5.28 se muestra las utilidades de ThingSpeak.

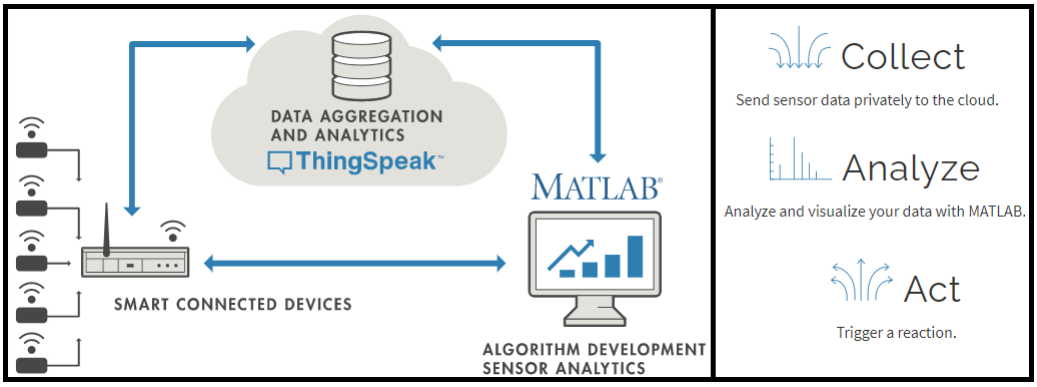



**Figura 5.28: Utilidades ThingSpeak**

Fuente: [59]

### 5.9.2. Ubidots [60],[61]

Ubidots es una plataforma IoT para monitorear, controlar y automatizar de forma remota los procesos para clientes de atención médica en sus inicios, así como nuevas empresas financiadas y en fin, proyectos conectados a Internet en Salud, Energía / Servicios públicos, Manufactura, Transporte y venta minorista. Ubidots faculta a más de 60,000 estudiantes, creadores e investigadores para explorar y desarrollar tecnología y soluciones, ofreciendo a todos los usuarios sus primeros 3 dispositivos gratis, siempre.

Con UbiFunctions se pueden codificar funciones de nube personalizadas que analicen datos existentes utilizando UbiParsers, además de ejecutar las funciones de filtrado de datos, limpieza de datos, agregación de datos y optimización de datos en la nube para extraer información.

Con el motor de eventos de Ubidots puede agregarse lógica empresarial condicional y compleja a su hardware activando la comunicación M2M y alertas por SMS, correo electrónico, Telegram y Slack para mantener a los operadores informados, también tiene la opción de programar mensajes descendentes que serán transmitidos al nodo final, esa fue la utilidad de Ubitdots en este proyecto, ya que

gracias a la plataforma se enviaba una confirmación programada de los enlaces ascendentes.

Esta plataforma también es de fácil integración con TTN por lo cual fue escogida para este proceso. En la figura 5.29 se resumen las utilidades de Ubidots.

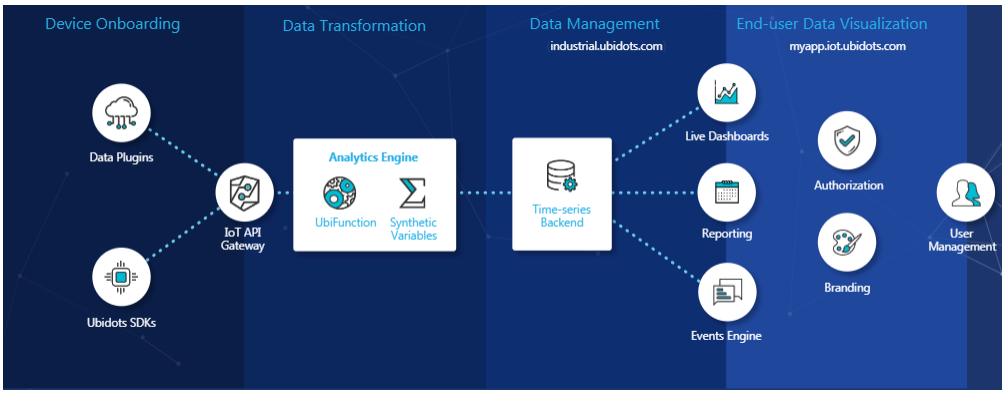



**Figura 5.29: Utilidades Ubidots**

Fuente: [61]

# 6. Estado del arte

## 6.1 Artículos relacionados con la determinación de la calidad del aire

- **Dispositivo electrónico portable para la medición de la contaminación del aire [62]**

Este proyecto resalta la importancia de crear un dispositivo portable de bajo costo que permita medir algunos de los indicadores de la contaminación del aire, ya que como se menciona la mayoría de dispositivos existentes son de muy alto costo y no tienen una gran cobertura. El proyecto constó de 4 fases como se describe a continuación.

**Fase 1:** Identificación de las variables de medición. En esta fase se definieron las variables a medir que fueron la humedad, la temperatura, el material particulado y el Monóxido de Carbono en el ambiente.

**Fase 2:** Seleccionar los sensores y el controlador. En el proyecto no se especifican las referencias de los sensores utilizados, sin embargo, se explican las características de los mismos. Como microcontrolador decidieron utilizar Arduino por su bajo costo.

**Fase 3:** Implementar el dispositivo. Esta fase tuvo tres partes, la primera la conexión de los sensores al microcontrolador, además de la adecuación del circuito en una caja de proyecto, la segunda la programación del microcontrolador que permite medir las variables en cuestión y calibrar los sensores y la tercera parte fue la visualización en una aplicación llamada AirCasting, que recibe los datos en el celular a través de una conexión Bluetooth.

**Fase 4:** Pruebas. En esta fase se realizaron las pruebas del dispositivo comparando su respuesta con instrumentos de medición más especializados.

Lo que se pudo observar es que al principio las mediciones contenían mucho ruido y eran muy lejanas a lo esperado, por lo cual decidieron implementar algunos filtros que mejoraron su respuesta. Lo que se concluyó es que aún es posible mejorar el dispositivo caracterizándolo y ajustándolo para mejorar su precisión. Como trabajo futuro se propone probar el dispositivo en diferentes ambientes y a diferentes distancias.

- **Estudios perceptivos sobre calidad del aire y sonido a través de paseos urbanos [63]**

En este documento se plantea que además de la medición por medio de sensores para determinar la calidad del aire, es también importante tener en cuenta la precepción de los ciudadanos con respecto al tema, esto para tomar medidas que permitan mejorar la calidad del aire y de igual forma mejorar la opinión de las personas sobre lugares donde este factor es desfavorable. También es necesario aclarar que se realiza el mismo estudio analizando el parámetro de la calidad

del sonido.

Para lograr este objetivo se hace una correlación cónica entre los datos subjetivos (opinión de las personas) y los datos objetivos (medición de sensores) en diferentes entornos como parques y calles concurridas. Como resultado se obtuvo que existe una correlación de 0.531 lo que indica que los datos obtenidos tienen una relación fuerte, información que demuestra que para evaluar la calidad del aire de un lugar no siempre es necesaria la utilización de sensores, sino que un método subjetivo como la percepción de las personas también es válido.

Al obtener un resultado positivo se sugiere recolectar datos, hacer simulaciones de diferentes aspectos y verificar la interacción de los parámetros analizados para tomar decisiones que minimicen el impacto de la contaminación del aire y por ende la calidad de vida urbana.

- **Análisis de la calidad del aire basado en imágenes utilizando una red neuronal convolucional profunda [64]**

Se propone crear una red neuronal para analizar imágenes y determinar la calidad del aire, dado que instalar estaciones de calidad del aire para algunas regiones resulta tener costos muy elevados. Es de notar que en la actualidad las personas con cámaras fotográficas como sus celulares ha crecido, debido a esto se hace factible y económico para regiones que no cuentan con presupuesto.

*En este documento, motivado por el notable éxito de las CNN, se propone el primer método que aplica la CNN para el análisis de la calidad del aire, en particular, la estimación de la concentración de PM 2.5 basada en imágenes para cualquier ubicación. Los canales de color RGB de la imagen se ingresan a la red de aprendizaje profundo para estimar la concentración de PM 2.5 directamente, lo que supera las restricciones de tener sensores, datos auxiliares, ubicación fija o funciones de imagen sofisticadas. Además del método propuesto, también creamos un conjunto de datos de 591 imágenes de la ciudad de Beijing con datos de PM 2.5 asociados.*

Este documento pretende hacer uso de las cámaras fotográficas de los ciudadanos para determinar la calidad del aire y con ello minimizar costos.

## 6.2 Artículos relacionados con las redes LoRa, LORAWAN y plataforma de desarrollo arduino

- **Implementación de un gateway low-cost para el protocolo LoRa [49]**

*En este trabajo de grado se propone el desarrollo de un Gateway de bajo costo compatible con la técnica de modulación LORA y los módulos RN2903 desarrollados por MicroChip, de forma que se construyan aplicaciones propias del IoT de forma rápida y eficiente*. Este proyecto enfatiza el hecho que los Gateway comerciales son muy costosos y hechos en el exterior, por lo cual lograr un dispositivo de bajo costo sería ideal para el desarrollo de más proyectos en la Universidad Distrital y en Colombia. Este dispositivo se implementa a partir de los módulos RN2903 y el dispositivo embebido RaspberryPi.

Se realizaron pruebas que permitieron conectar tres nodos en diferentes ubicaciones mediante la modulación LORA, luego subir esta información a internet para su análisis y postprocesamiento, con estas pruebas se evidenció el correcto funcionamiento del Gateway, pudiendo concluir que la modulación LORA no es recomendable para redes que necesiten un alto tránsito de datos, debido a la latencia en el tiempo de transmisión y a la baja tasa de datos que se pueden transmitir, por ello recomiendan aplicar este tipo de redes en agricultura, domótica y SmartCities.

Como trabajos futuros se espera que con esta herramienta pueden implementarse más proyectos de IoT en la Universidad Distrital, además de que si es requerida una red más compleja puedan tomarse características del protocolo LORAWAN para por ejemplo implementar la seguridad en la red.

- **Monitoreo de sensores de IoT de área grande usando un sistema de red de malla inalámbrica LoRa: diseño y evaluación [65]**

  En este proyecto se propuso implementar una red en topología malla con modulación LORA que utilice menor cantidad de Gateways, ya que generalmente se implementan en topología estrella y utilizan muchos de dichos dispositivos para evitar la interferencia en espacios interiores y exteriores.

  La arquitectura propuesta tiene un solo Gateway que recopila los datos secuencialmente de los sensores IoT y también envía los datos a la interfaz del usuario, que luego pueden leerse en la aplicación del usuario para poder leer los datos y analizarlos.

  Como resultado se obtuvo que *La red de malla inalámbrica es una solución para aumentar el rango de comunicación y la relación de entrega de paquetes (PDR) sin la necesidad de instalar GW adicionales, ya que el sistema de redes de malla LORA propuesto logró un promedio de 88.49% de PDR, mientras que la topología de red en estrella utilizada por LORA logró solo el 58.7% en los mismos ajustes.*

Sin embargo, en los resultados se obtuvo un retraso en la transmisión debido a que la información pasa por diferentes nodos, por ello se propone a futuro buscar soluciones para mejorar la transmisión, además de que en el proyecto realizado para el envío de datos los nodos deben recibir una señal de activación lo que genera más retrasos, esto es otro factor a mejorar. También se propone implementar un sistema de seguridad para la red basándose en la solución de seguridad de LORAWAN. Por último, se especifica que también es necesario buscar una mejor solución energética si se requiere una red donde los nodos son alimentados por batería, ya que como los nodos siempre están atentos a una señal de activación emitida por el Gateway se encuentran siempre encendidos, en este proyecto no hubo inconveniente debido a que la alimentación fue suministrada por la Red eléctrica.

- **Diseño de una red de sensores inalámbricos LPWAN mediante módulos LoRa para el monitoreo de la calidad del agua en dos ríos [40]**

  En este documento se muestra el diseño y la implementación de una red de sensores inalámbricos que permiten medir los parámetros de calidad del agua en ríos utilizando componentes de bajo presupuesto. Los módulos LORA se emplean para la comunicación entre nodos, tecnología que surgió con el auge que ha tenido la WSN (redes de sensores inalámbricos) y el IoT.

  Este documento muestra que el proyecto desarrollado tiene su propósito en diseñar e implementar *dos módulos que están construidos para detectar los siguientes parámetros: conductividad eléctrica, pH, temperatura. Por ello la adquisición, diseño y construcción de los circuitos de acondicionamiento está hecho. También se crea un nodo receptor que se conectará a una computadora y mostrará los datos a través de una interfaz gráfica programada en java.*

  Este proyecto implementó una red de sensores inalámbricos haciendo uso de los módulos LORA, que permitieron la comunicación entre nodos.

## 6.3 Artículos relacionados con más de una temática del proyecto a realizar

- **Sistema de monitoreo de calidad de aire de grano fino de bajo costo que usa LORAWAN [66]**

  Este documento implementa un sistema que monitorea la calidad del aire midiendo el material particulado, la temperatura y humedad del ambiente por medio de sensores de bajo costo e implementando una red LORAWAN, teniendo en cuenta que los sistemas actuales no tienen gran cobertura, además que son costosos. El proyecto utiliza una red con topología estrella, propia de LORAWAN y se compone de tres elementos principales como los son los dispositivos finales,

Gateways y servidores Web.

Los dispositivos finales que hacen parte de la capa de percepción están compuestos del módulo SX1272MB2DAS con LORA MAC modificado, que se conectó a la placa NUCLEO-L073RZ y luego se equipó con el sensor Plantower PMS5003T. La capa de red recoge los datos, asigna una ruta y envía la información a un servidor Web a través de internet. El Gateway fue implementado con la tarjeta de desarrollo Raspberry Pi 3 conectada con una placa RAK831 que contiene un chip concentrador SX1301. Por último, se tiene una aplicación que permite monitorear los resultados de los sensores y se actualiza cada 15 minutos.

Con respecto a los dispositivos finales, estos *están alimentados por baterías portátiles de 10200 mAh y están revestidos para bloquear las perturbaciones externas*. Como resultado obtuvieron una red con nodos separados por una distancia de aproximadamente 473 m. Como trabajo futuro se propone implementar más nodos y más Gateway para mejorar la cobertura y fidelidad de las mediciones.

- **Sistema de comunicación en redes de sensores inalámbricos que utilizan Raspberry Pi y arduino para monitorear el gas de la contaminación del aire [67]**

En este artículo se plantea como necesidad medir la calidad del aire en Gesik, Indonesia considerada una ciudad industrial, construyendo un módulo de bajo costo que permita realizar una red de sensores inalámbricos (WSN) para monitorear los contaminantes en el aire. El proyecto cuenta con dos partes, la primera parte es la implementación de la red que combina uno nodo receptor realizado con el embebido Raspberry Pi y dos nodos estáticos realizados con la tarjeta de desarrollo Arduino, tarjeta SD, el módulo Wifi ESP8266, sensor de humedad, sensor de temperatura y sensores de gases (Serie MQ). Y la segunda parte es la visualización de los datos obtenidos en una aplicación para que todas las personas tengan acceso a la información.

También se hizo una evaluación sobre el consumo de energía de la red teniendo en cuenta 3 estados del nodo estático: cuando está inactivo, al recibir una señal del Gateway y al enviar los datos a la parte central, lo que les permitió concluir que al poner el sensor en modo inactivo se logra un ahorro de energía entre el 17 % al 27 % con respecto al segundo estado y de un 39 % al 49 % teniendo en cuenta el tercer estado.

Al final obtuvieron que la mejor respuesta de la red fue a una distancia de 110 $m$ entre el nodo receptor y un nodo estático ya que obtuvieron 0 % en la Tasa de Error de Bit (BER), lo que significa que a esta distancia no hay pérdidas de los datos.

- **Soluciones para SMARTCITIES: propuesta de un sistema de monitoreo de la calidad del aire basado en una red LORAWAN con sensores portátiles [68]**

  Este proyecto fue realizado en la Plata, Argentina y consiste en el desarrollo de una red de sensores portátiles basada en el protocolo LORAWAN, utilizando software y hardware libre, el conjunto de datos es transmitido a una plataforma escalable para IoT en la nube. Realizaron las pruebas utilizando 3 sensores de modelos diferentes (Nova SDS011, Nova SDS021 y SHINYEI PPD42) y con un sensor de material particulado de bajo volumen MiniVol TAS Airmetrics.

  "En esta experiencia piloto se desplegó una red LORAWAN usando la infraestructura de torres de la Universidad Nacional de la Plata (UNLP) y el backbone de fibra óptica que las conecta. Se utilizaron 5 dispositivos marca Kerlink modelo iBTS con antenas omnidireccionales". Lo que les permitió tener una amplia cobertura que cubría tanto la ciudad de la Plata y parte de otras ciudades. Utilizaron nodos desarrollados con el embebido Arduino y 3 modelos de sensores diferentes, donde tuvieron un problema de desconexión de uno de los sensores a pesar de tener un gabinete que resguardó los sensores, por lo cual una de las observaciones es que a futuro será necesario poder determinar cuando el sensor está desconectado. Realizaron 4 campañas de monitoreo de las cuales concluyeron que los sensores utilizados dos sensores (Nova SDS011 y Nova SDS021) tuvieron un comportamiento similar, mientras que el otro (SHINYEI PPD42) si tenía variaciones considerables.

  Al notarse una alta variación entre los resultados obtenidos y los esperados de acuerdo a estudios en el laboratorio se requiere ajustar los equipos en el entorno donde se trabaje. Además, se recomienda realizar un mantenimiento periódico de los equipos que permita determinar el desgaste de los mismos y su ajuste, si es necesario.

  Este sistema está enfocado al desarrollo de ciudades inteligentes (Smart Cities) y se concluye que aún queda mucho más que integrar para lograr este objetivo, que no busca sustituir los sistemas de monitoreo ya existentes sino complementarlos.

# 7. Metodología

## 7.1 Definición de los indicadores a medir

Para la definición de los indicadores de calidad de aire a medir se realizó una reunión con el Doctor Andres Torres, actualmente profesor de la Pontificia Universidad Javeriana y director del grupo de investigación: Ciencia e Ingeniería del Agua y el Ambiente, quien gracias a sus amplios conocimientos en la calidad del aire en la actualidad nos sugirió medir material particulado, monóxido de carbono, dióxido de nitrógeno, presión, temperatura y humedad debido a que en la actualidad estos indicadores son de gran importancia a la hora de determinar la calidad del aire y son los más perjudiciales para la salud. Por tal motivo los indicadores anteriormente mencionados fueron los elegidos para el desarrollo del nodo que permitirá determinar la calidad del aire, además se realizará la toma de muestras cada cinco minutos debido a que las variables a medir no tienen cambios muy significativos en lapsos cortos de tiempo.

## 7.2 Arquitectura e implementación nodo final

En esta sección se explican las partes que componen cada uno de los nodos finales, para ello se hace un recuento de las consideraciones más importantes que se tuvieron en cuenta para la implementación del nodo y su correcto funcionamiento. Cada una de las partes que componen el nodo se muestran en la figura 7.1. Se va a iniciar con el microcontrolador que es su parte central y luego con cada uno de los periféricos, además se incluye la explicación del código final, el sistema de alimentación, el diseño de la PCB y estructura robusta del nodo.

### 7.2.1. Microcontrolador: STM32F103C8

#### 7.2.1.1. Comparación entre STM32F103C8 Y Arduino Uno

Para hacer la elección de microcontrolador se decidió realizar una comparación entre el microcontrolador STM32F103C8 y el Arduino uno, a continuación se mostrarán las características principales de cada uno en la tabla 7.1.

**Tabla 7.1: Comparación Arduino Uno VS STM32F103C8**

| CARACTERÍSTICAS ARDUINO UNO | CARACTERÍSTICAS STM32F103C8 |
|---|---|
| Microcontrolador ATmega328 | Microcontrolador ARM 32-bit Cortex$^{TM}$-M3 CPU Core |

Continúa en la página siguiente

| CARACTERÍSTICAS ARDUINO UNO | CARACTERÍSTICAS STM32F103C8 |
|---|---|
| Voltaje Operativo de 5 *V* y Voltaje de Entrada (Recomendado) de7 – 12 *V* | Voltaje Operativo de 3.3 *V* |
| Pines de Entradas/Salidas Digital 14 | 26 entradas y salidas digitales, la mayoría tolerantes a 5V |
| 6 entradas analógicas con ADC de 10 bits | 10 entradas analógicas con ADC de 12 bits |
| Memoria Flash de 32 KB (ATmega328) de los cuales 0,5 KB es usado por Bootloader | 64 KB de memoria Flash con 20 KB de Bootloader |
| 2 KB de SRAM | 20 KB de SRAM |
| EEPROM: 1 KB (ATmega328) | No tiene EEPROM |
| 1 interfaz I2C | 2 intefaces I2C |
| 1 intefaz SPI | 2 intefaces SPI |
| 1 intefaz USART | 3 intefaces USART |
| Reloj de 16 MHz | Reloj 72 MHz |
| No contiene RTC interno | Contiene RTC interno |
| 2 pines para interrupciones | Interrupciones en todas las I/O |
| Costo:$25.000 | Costo:$15.000 |

Fuente: [69],[70]

El STM32F103C8 es una placa de desarrollo, similar a Arduino. Sin embargo, mientras la mayoría de Arduinos cuenta con un procesador AVR, el STM32F103 dispone de un procesador ARM, lo que quiere decir que tiene unas especificaciones superiores [69].

Lo que hace novedoso y especial al STM32F103C8 es que es la primera placa ARM realmente barata. Otra comparación realizada entre el arduino y el STM32, fue una prueba donde subiéndoles el programa con las librerías de los sensores se verificó que tanta disponibilidad tenían a lo cual el Arduino uno arrojó: 84 % espacio almacenamiento programa. 76 % memoria dinámica y un mensaje de advertencia de **Poca memoria disponible, se pueden producir problemas de estabilidad**; a diferencia del microcontrolador STM32F103C8 quien arrojó: 87 % Espacio almacenamiento programa. 27 % memoria dinámica. Lo anterior reitero de la decisión de elegir el STM32F103C8 como el microcontrolador apto para el desarrollo del nodo.

#### 7.2.1.2. Configuración del IDE de Arduino para STM32F1013C8

Para que el IDE de Arduino funcione con la placa STM32F103C8 es necesario agregarlo al gestor de placas. En primer lugar, se abre el IDE de Arduino y luego se selecciona al gestor de placas (Herramientas/Placa/Gestor de tarjetas). Aquí se añade el soporte para las placas SAND (Zero o similar),

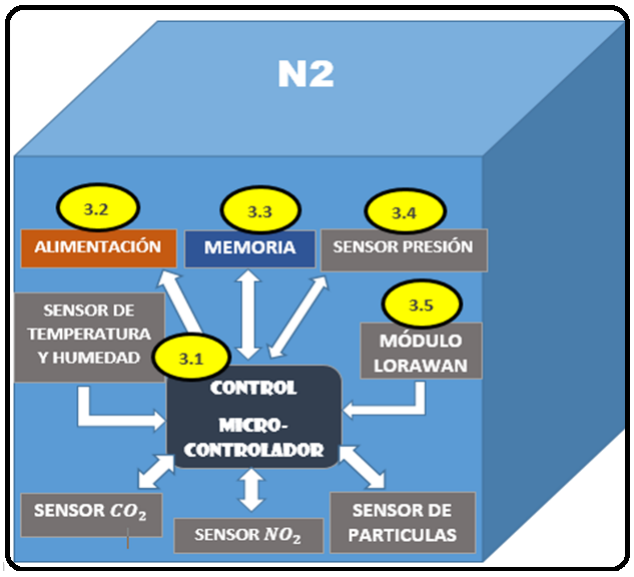



**Figura 7.1: Arquitectura del nodo**

Fuente: Autores

en la Figura 7.2 se puede observar el proceso [33].

A continuación se abre preferencias y en Gestor de URLs adicionales de tarjetas, se hace click en el botón en forma de ventana de la derecha y se pega la dirección `"http://dan.drown.org/stm32duino/package_STM32duino_index.json"` en la ventana que aparece, como se ve en la figura 7.3.

Luego volvemos al gestor de placas y tendremos disponibles las placas STM32. Instalamos las que necesitemos como se ve en la Figura 7.4.

De esta manera ya se puede subir el programa a la tarjeta STM32.

#### 7.2.1.3. Formas de subir un código a STM32F103C8

Para programar el microcontrolador STM32F103C8 con el IDE de Arduino, tenemos tres opciones las cuales se describirán a continuación.

- **Conectar el STM32 con el conversor USB-TTL [33]**

  La primera forma para programar el STM32 es por UART, por lo que se va a necesitar un conversor USB-TTL. La tensión de alimentación del STM32 es de 3.3 $V$, por lo que se tiene que asegurar que el conversor TTL trabaje a 3.3 $V$. Normalmente funcionan tanto a 5 $V$ como a 3.3 $V$ y se puede elegir la tensión de salida con un jumper de configuración, bien en la placa o bien

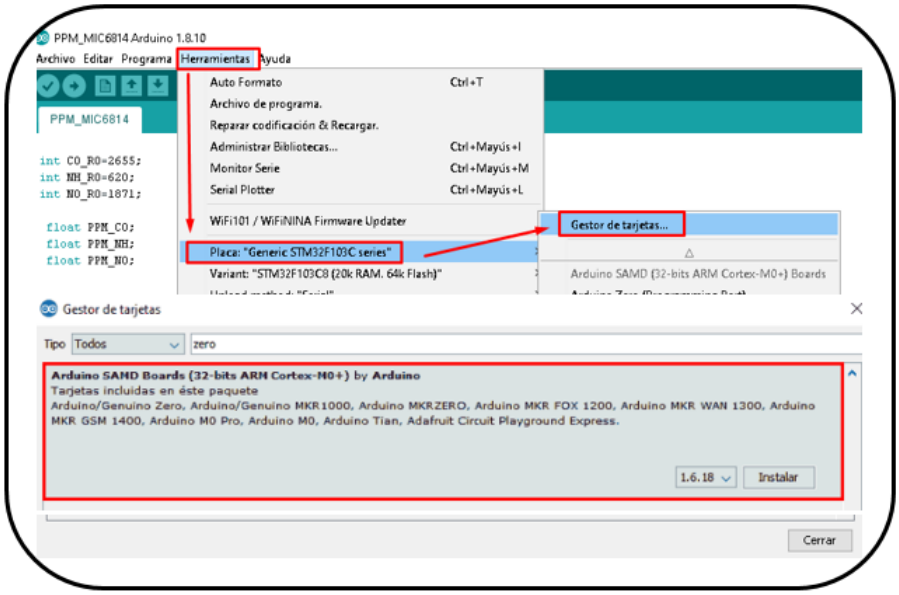



**Figura 7.2: Añadir soporte para las placas SAND**

Fuente: [33]

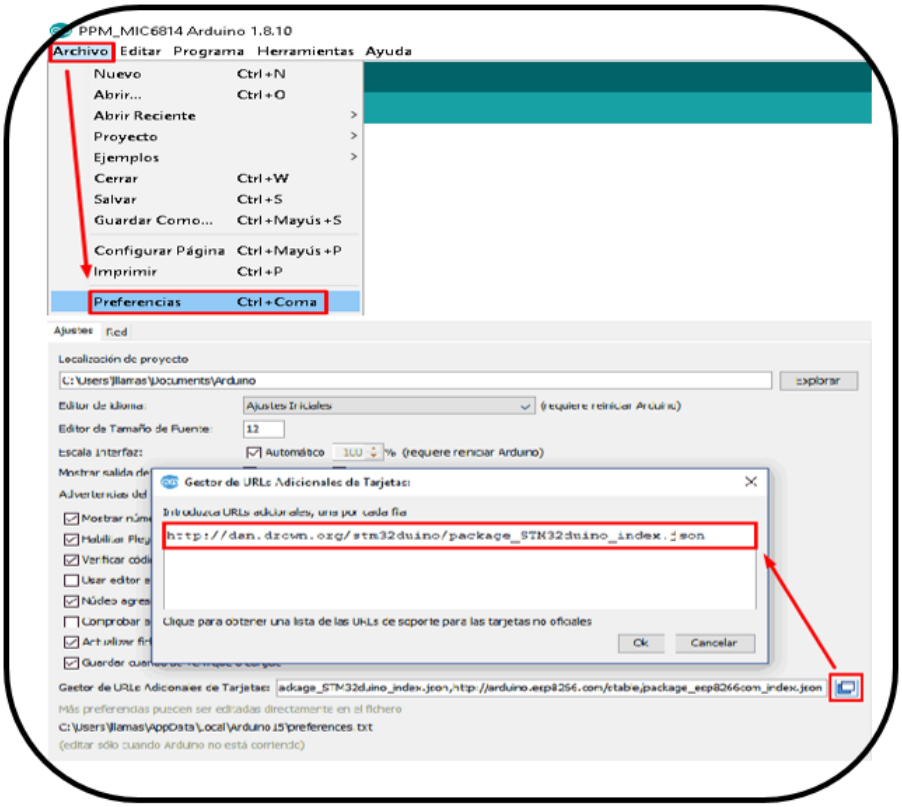



**Figura 7.3: Añadir el STM32F103C8 al gestor de URLs**

Fuente: [33]

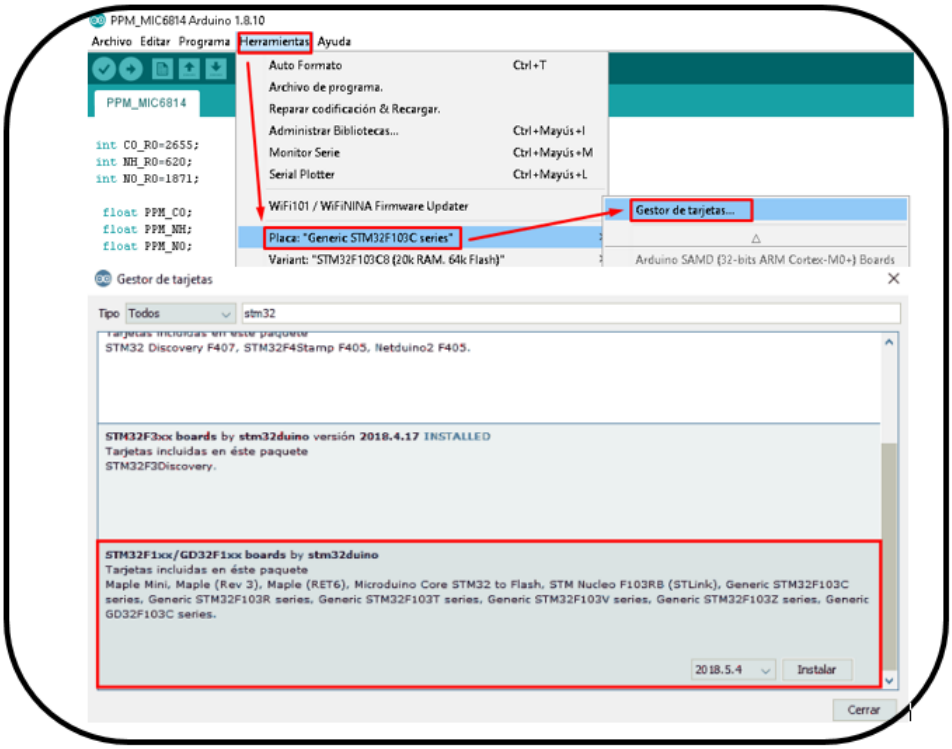



**Figura 7.4: Añadir el STM32F103C8 al gestor de placas**

Fuente: [33]

en los pines de salida.

Para programar el STM32 se deben conectar los pines de comunicación del conversor USB-TTL: TX y RX a los pines de UART1 del STM32, que son el A10 y A9, respectivamente. Por tanto, la conexión es como se ve en la Figura 7.5.

Ahora, se conecta el conversor USB-TTL al ordenador y se elige la placa genérica STM32F103C, con los parámetros de configuración que se muestran en la figura 7.6, para probar que todo funciona correctamente se sube el archivo conocido como Blink, el equivalente al **Hola mundo** en placas de desarrollo, que hace parpadear el LED integrado en la placa. Por tanto, se pega el texto como se ve en la Figura 7.6, se carga en la placa y el LED debería ponerse a parpadear. Eso es así porque el bootloader está configurado para ejecutar el programa inmediatamente después de su carga.

- **Programar STM32 con IDE de Arduino y ST-LINK V2 [33]**

  El ST-LINK V2 es un programador/debugger construido por ST Microelectronics que permite programar y debuggear los procesadores STM32. El ST-Link V2 implementa SWIM (Single Wire Interface Module) y JTAG/SWD (Serial Wire Debugging) para comunicar con los proce-

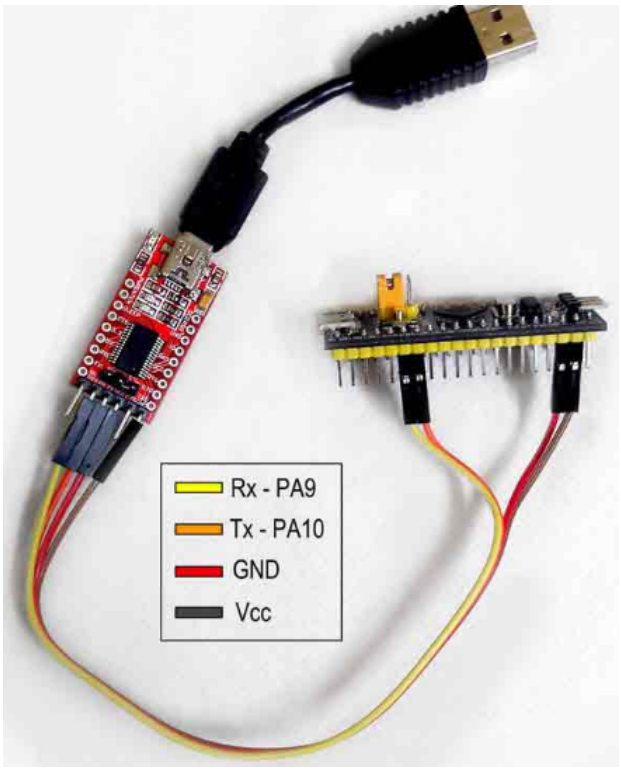



**Figura 7.5: Conexiones del conversor USB-TTL**

Fuente: [33]

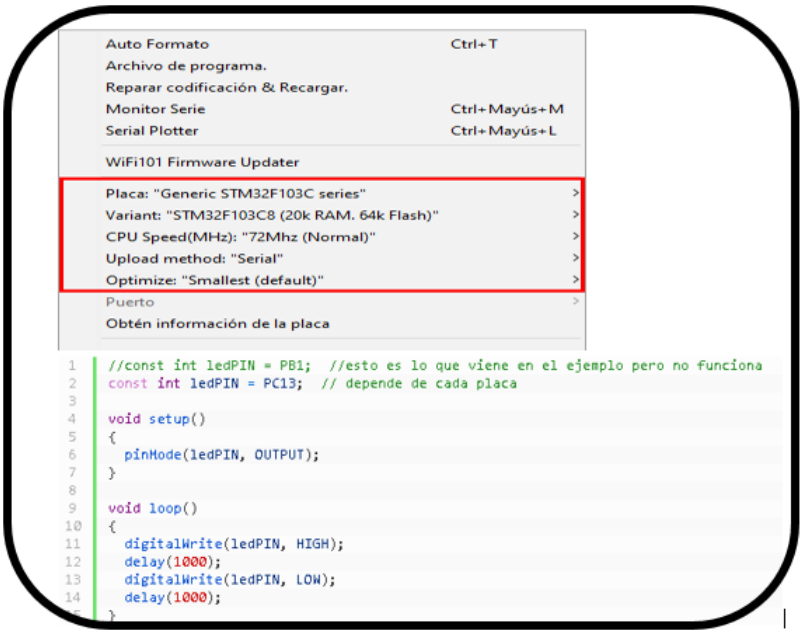



**Figura 7.6: Subir el archivo Blink al microcontrolador**

Fuente: [33]

sadores de la placa de desarrollo.

En primer lugar, se deben instalar los drivers de la página de ST. Los driver son gratuitos, pero hay que registrarse para poder descargarlos. De la carpeta que se descarga, hay que ejecutar **install_drivers.bat** como administrador. Se conecta el dispositivo y hay que asegurarse de que el sistema lo reconoce correctamente como se ve en la Figura 7.7 [33].




**Figura 7.7: Drivers y conexiones del ST-LINK**

Fuente: [33]

Para probar que todo funciona correctamente de nuevo se prueba con el archivo conocido como Blink, el equivalente al **Hola mundo**, en placas de desarrollo, que simplemente hace parpadear el LED integrado en la placa como se ve en la Figura 7.6, sin embargo para que funcione debe colocar en **Upload method** la opción **STLink**. Debe tenerse en cuenta que con el STLINK no se puede utilizar el puerto serial, es una de sus limitaciones.

Se carga en la placa y el LED debería ponerse a parpadear. Eso es así porque el bootloader está configurado para ejecutar el programa inmediatamente después de su carga.

- **Programar STM32 con cable USB Y Bootloader [71]**

  Para programar la placa STM32F103C8 directamente a través del puerto USB, primero se debe flashear el cargador de arranque Maple en el MCU. Para esto, se hace uso del conversor USB-

TTL. Esta placa está conectada al pin Rx y Tx del STM32 como se mostró en la figura 7.5, conectando los pines del conversor USB-TTL al microcontrolador de la siguiente forma:

- RX: PA9
- Tx: PA10
- GND: GND
- Vcc: Vcc (5V)

Se debe asegurar que el pin del puente de arranque 0 en la placa esté configurado en 1 (modo de programación) mientras se carga el Bootloader o cargador de arranque. Una vez que se muestra el cargador de arranque, este pin puede cambiarse nuevamente a la posición inicial (modo operativo), como se ve en la Figura 7.8.

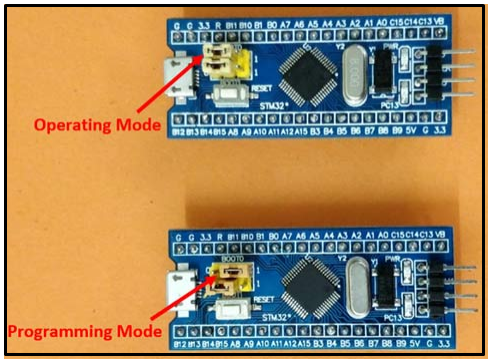



**Figura 7.8: Modo de operación del módulo STM32F103C**

Fuente: [71]

Una vez que se haga la conexión anterior, se conecta el conversor USB-TTL a la computadora y se siguen los siguientes pasos para actualizar el cargador de arranque en el STM32.

- **Paso 1:** Se descargan los archivos binarios del archivo del programa del cargador de arranque (archivo bin), existen muchas versiones de este archivo.
- **Paso 2:** Descargar e instalar el software del cargador STM Flash para actualizar el archivo bin descargado en STM32.
- **Paso 3 :** Una vez descargado, se instala el software, se asegura de que la placa STM32 esté conectada a la computadora a través del conversor USB-TTL, luego se inicia el software. El software detectará automáticamente el puerto COM, si no, use el administrador de dispositivos y se asegurará de seleccionar el número de puerto COM correcto. Se debe dejar el resto de la configuración como se muestra en la figura 7.9.
- **Paso 4:** Se da click en el botón siguiente dos veces y el software nuevamente detectará automáticamente los detalles de la placa y se mostrará como se ve en la Figura 7.10, es de recordar que la placa en uso es la STM32F1 con memoria flash de 64 *KB*.

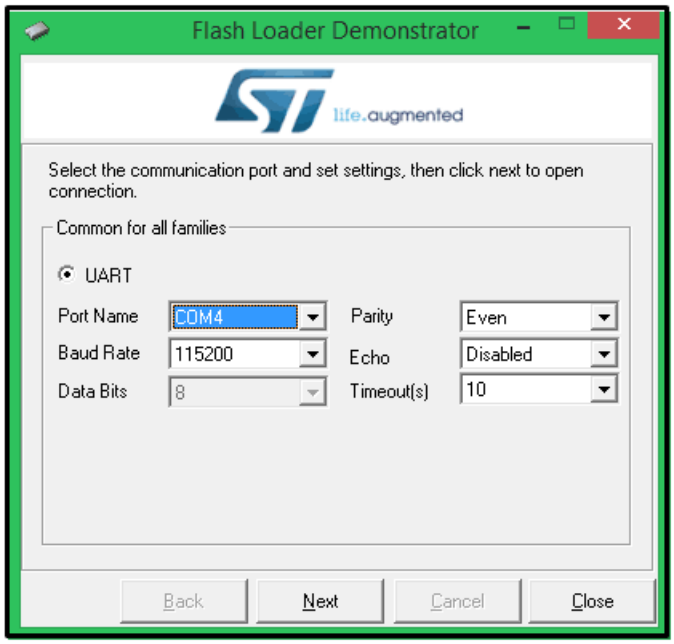



**Figura 7.9: Flash Loader**

Fuente: [71]

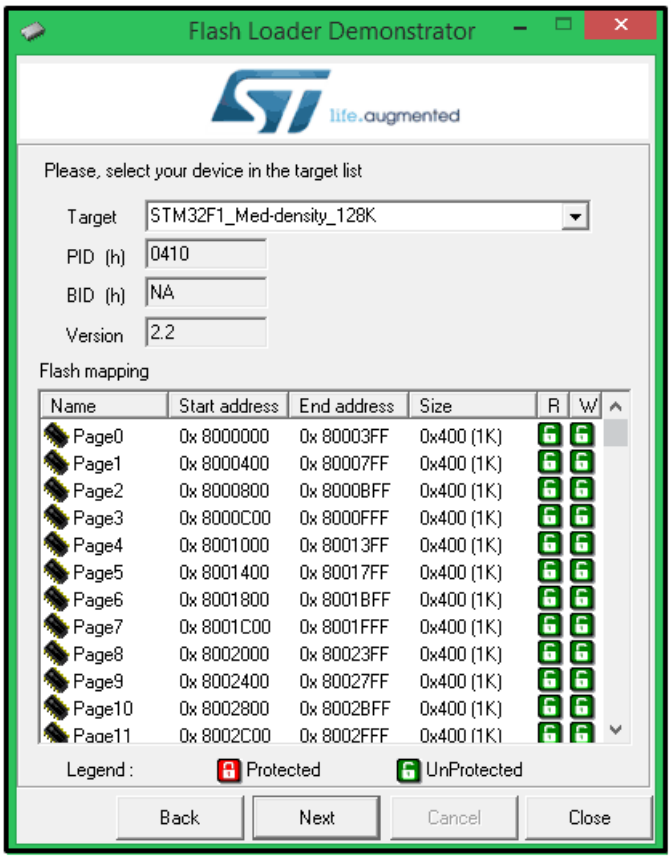



**Figura 7.10: Seleccionar placa STM32F1**

Fuente: [71]

- **Paso 5:** Se selecciona descargar en el dispositivo y se busca la ubicación donde se descargo el archivo bin en el paso 1 y se selecciona. Luego se presiona siguiente.
- **Paso 6:** El software descargará algunos archivos necesarios y luego comenzará el proceso de actualización, al finalizar cerramos esta ventana. Ahora se debe preparar el IDE de Arduino e instalar los controladores antes de poder programar la placa STM32.

Para la preparación del IDE de Arduino e instalación de los controladores se deben seguir los siguientes pasos:

- **Paso 1:** Se descarga los paquetes necesarios para la placa STM32. Esto se puede hacer seleccionando Archivo y luego Preferencias.
- **Paso 2:** Al hacer clic en Preferencias, se abrirá un cuadro de diálogo, en el cuadro de texto URL de Boards Manager adicional se debe pegar el siguiente enlace `http://dan.drown.org/stm32duino/package_STM32duino_index.json` y por último OK, como se muestra en la figura 7.11

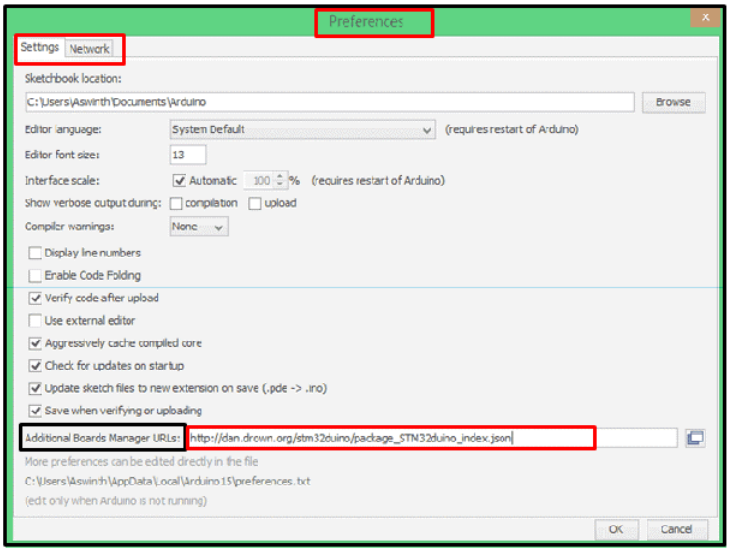



**Figura 7.11: Adicionar URL**

Fuente: [71]

- **Paso 3:** Ahora se ingresa a herramienta /Tableros/Tablero Administrador . Esto abrirá el cuadro de diálogo administrador de placas, se buscará "STM32F1.[ei]nstalará el paquete que aparece.
- **Paso 4:** Después de instalar el paquete, navegue a C: / Archivos de programa (x86) / Arduino / hardware / Arduino_STM32-master / drivers / win donde encontrará install_drivers.bat e install_STM_COM_drivers.bat .
- **Paso 5:** Hacer click en ambos archivos bat e instalar los controladores. Ahora el IDE Arduino está preparado para programar la placa de desarrollo STM32 y los controladores también están instalados [71].

Ahora se debe retirar el conversor USB-TTL y todas las conexiones existentes del microcontrolador STM32. Simplemente se usa el puerto micro-USB en la placa STM32 para conectarlo a la computadora como se muestra en la Figura 7.12. Se debe asegurar que el pin 0 del puente esté posicionado nuevamente en 0 (modo de operación). De aquí en adelante, ya no se necesita alternar el puente para cargar y ejecutar los programas. En el código debe asegurarse que en el menú Herramientas, en **Upload method** debe elegirse la opción **STM32duino bootloader** sino saldrá error, puede que desinstale el bootloader y no reconozca el dispositivo, por lo cual se tiene que repetir el proceso anterior.

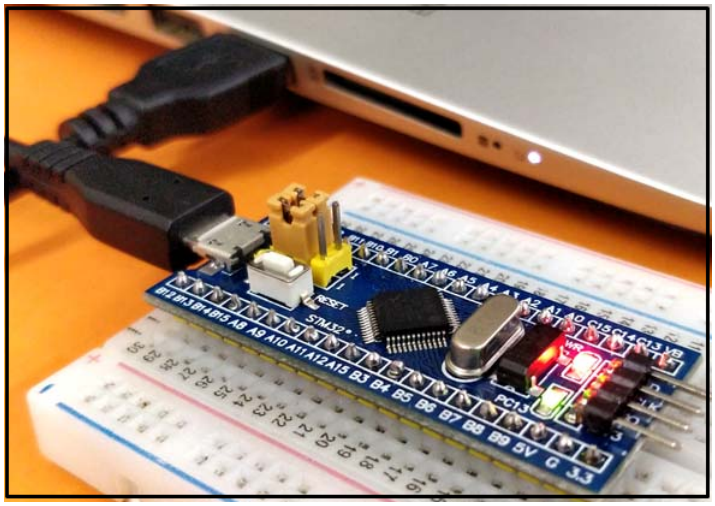


**Figura 7.12: Conexión directa del STM32 por el puerto USB**

Fuente: [71]

#### 7.2.1.4. Ubicación de los periféricos en el STM32F103C8

En esta sección se mostrará como se distribuyeron los pines del microcontrolador STM32F103C8 para ubicar cada uno de los sensores, el modulo LORA BEE e iniciar con la lectura, manejo de datos y envió, como se muestra en la Figura 7.13, los pines del A2 al A7 fueron asignados para el modulo LORA BEE que corresponde al puerto SPI1, los pines B10-B11 para el sensor de partículas, los pines A0-A1 para el sensor de gases, el pin B1 es un led que indica si los sensores están listos, esto debido a que el sensor de gas debe calentarse una hora antes de iniciar a medir, el pin B0 es el led que indica si se están transmitiendo mensajes, los pines B12 al B15 sería destinados para el modulo SD que corresponde al puerto SPI2, los pines B6-B7 para el sensor de presión, temperatura y humedad que corresponden al puerto I2C1.

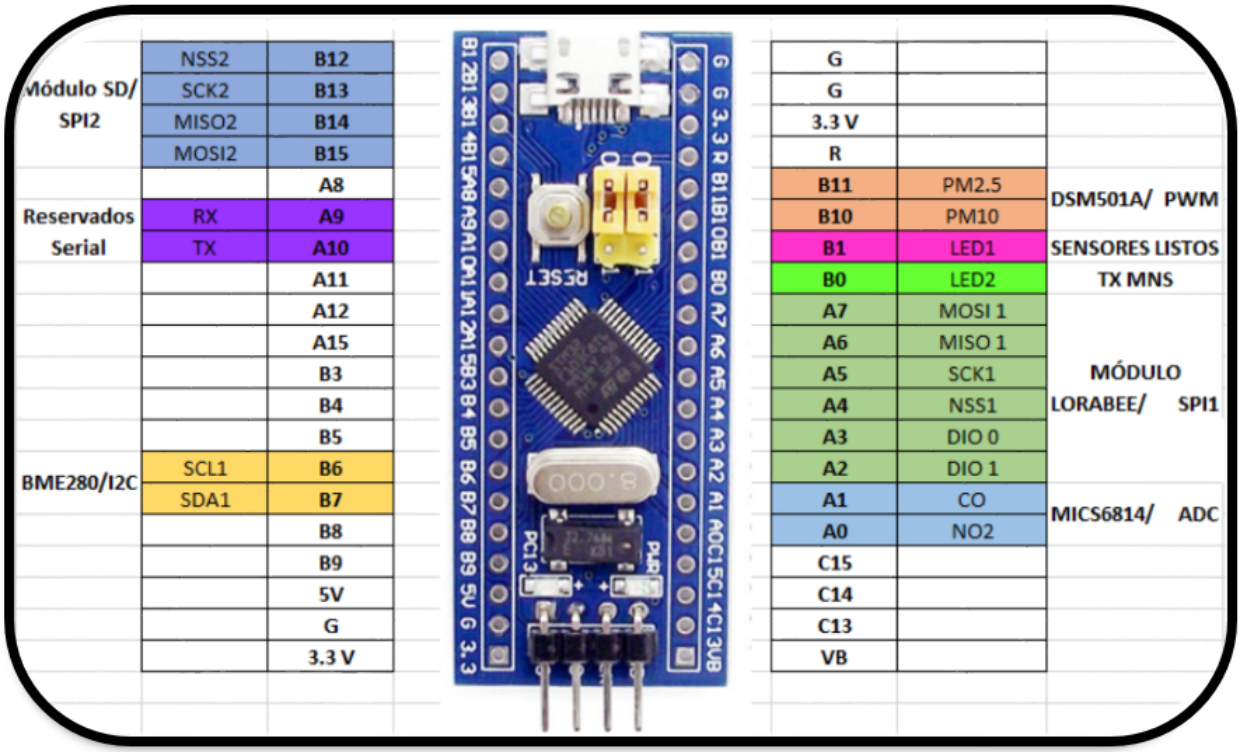



**Figura 7.13: Distribución de pines**

Fuente: Autores

### 7.2.2. Sensores

En esta sección se describe cómo deben conectarse los sensores de acuerdo a las señales de salida de cada uno en el microcontrolador y las librerías usadas, no se especifica el código utilizado en cada sensor porque en los anexos se incluye el código final con comentarios donde se encuentra esta información.

**Nota:** Al aprender a usar el sensor de gases MICS6814, uno de ellos se quemó y por la situación actual no fue posible reemplazarlo, por lo cual uno de los nodos tiene emulado los valores de este sensor.

#### 7.2.2.1. BME280: presión, temperatura y humedad

Este sensor puede comunicarse con el microcontrolador a través del protocolo I2C o el protocolo SPI, se eligió el primer protocolo de comunicación debido a su facilidad de uso con dos pines de conexión en lugar de 4 que se usarían con SPI.

Su conexión es muy sencilla, sólo hay que tener en cuenta cuáles son los pines I2C del microcontrolador, en el caso del STM32 este cuenta con 3 puertos I2C, sin embargo, por la librería implemen-

tada para la comunicación I2C llamada **Wire.h** que usa el puerto I2C número 1 de los microcontroladores se debe usar el pin PB6 que corresponde al pin del reloj **SCL1** y PB7 que corresponde al pin de datos **SDA1** como se muestra en la figura 7.14.

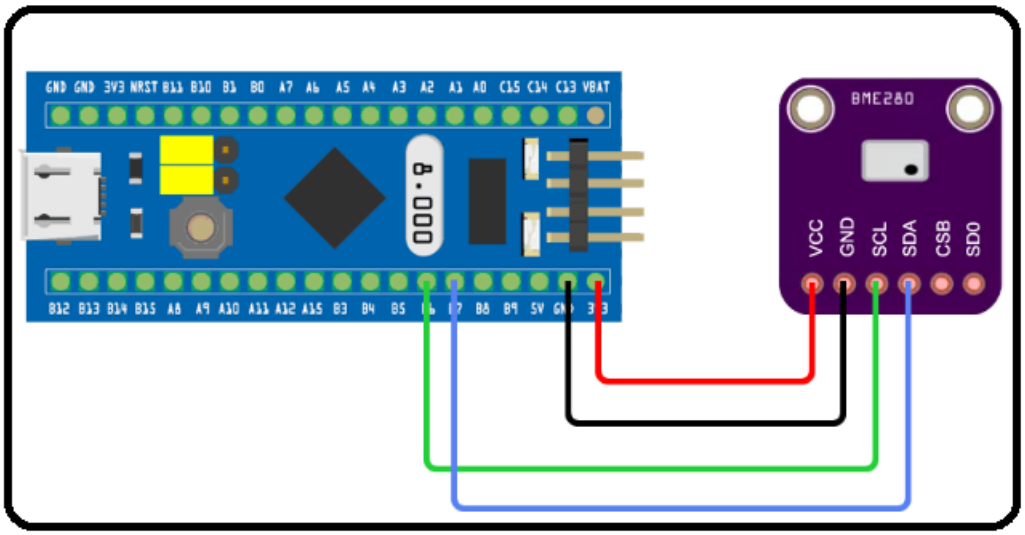



**Figura 7.14: Conexión BME280 con STM32F103**

Fuente: [72]

Además de la librería I2C en [73] se encuentra la librería **BMEI2C.h** utilizada para realizar la lectura de este sensor que debe ser instalada en el IDE de arduino. La librería tiene varias utilidades entre la cuales están : detectar la versión de BM utilizado, verifica que el sensor está conectado y permite elegir la unidad de medida de las variables de temperatura y presión. La salida de las lecturas del sensor en el puerto serial se muestra en la Figura 7.15.

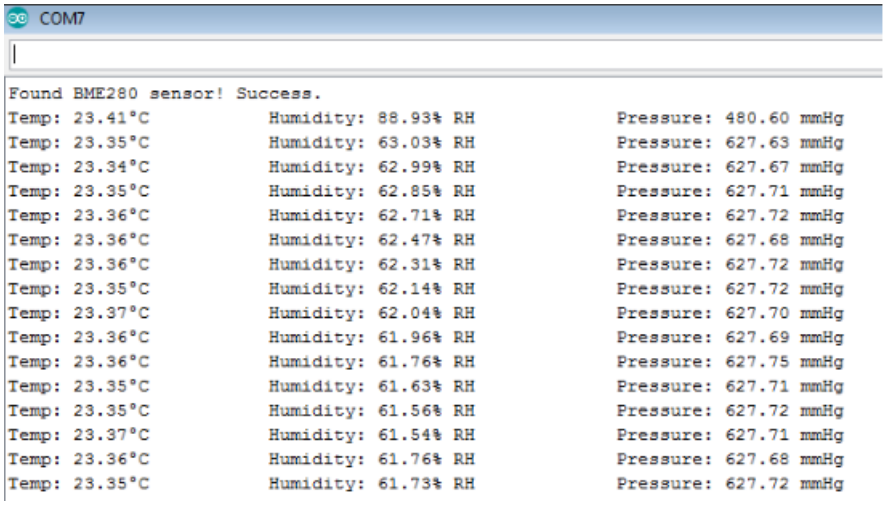



**Figura 7.15: Mediciones BME280**

Fuente: Autores

#### 7.2.2.2. MICS6814: CO y NO2

El sensor MICS6814 tiene varias versiones, entre las cuales se encuentra una con el protocolo I2C, sin embargo no es muy sencilla de adquirir en el mercado por lo cual se usó la versión sin protocolo de comunicación establecido y las lecturas del sensor son hechas por el Conversor Analógico Digital (ADC) del microcontrolador, que en el STM32 es de 12 bits, es decir, 4096 posibles valores.

Este tipo de sensor requiere de un tiempo de calentamiento de 1 hora para que sus lecturas sean correctas, por lo cual la placa lleva algunas resistencias de calentamiento. Como el sensor varía su resistencia en presencia de contaminantes, para poder interpretar sus cambios debe realizarse un divisor resistivo y este voltaje resultante es el valor a leer con el ADC. En la figura 7.16 se muestran la correcta forma de conexión del sensor con sus resistencias y se especifican los pines de lectura en el microcontrolador, cabe aclarar que a pesar de que no se tomen datos del sensor de NH3, debe conectarse la resistencia para que no se deteriore el sensor. El valor de las resistencias fue elegido para tener la mayor resolución con el ADC, teniendo en cuenta los rangos de lectura del sensor.

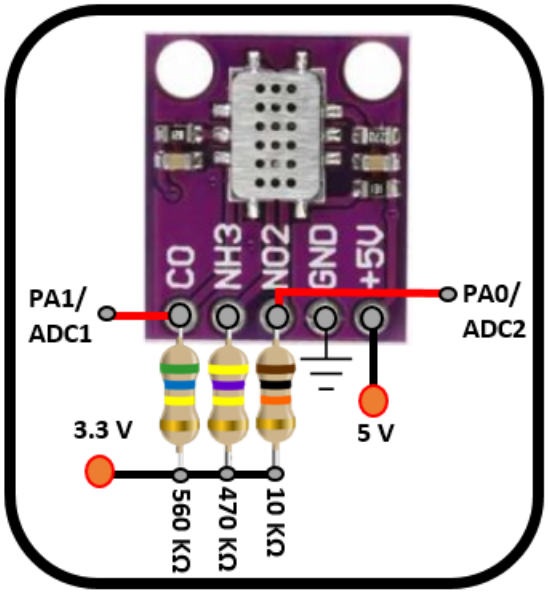



**Figura 7.16: Conexiones MICS6814**

Fuente: Autores

Las concentraciones de $CO$ y $NO_2$ que da el sensor se dan en partes por millón ($ppm$), sin embargo hay que hacer la conversión a $ug/m^3$ que son las unidades que se muestran en la norma, para determinar estos valores a partir de las lecturas del ADC teniendo en cuenta lo encontrado en [74] se debe:

- Primero debe determinarse el valor de la resistencia cuando no hay o es muy baja la contaminación del lugar llamada **R0** , para ello se elige un lugar donde no hayan fuentes de contaminación cercanas, se conecta el sensor, se espera una hora para que se caliente y cuando las medidas del

ADC sean estables o muy cercanas entre ellas, se hace un promedio entre 5 valores, hay que aclarar que este valor no es la resistencia sino un equivalente del voltaje medido por el ADC. Para el caso de $CO$ el valor fue de 1756 y el de $NO_2$ de 1300.

- Teniendo en cuenta esos valores se hace la relación entre el valor actual del sensor llamado **Rs** y el valor de **R0** hallado anteriormente para cada contaminante.
- A partir de las gráficas de la figura 7.17 se determinaron las ecuaciones (2) y (3).

$$CO_ppm = 4{,}385 * (Rs/R0)^{-1{,}177} \quad (2)$$
$$N0_{2p}pm = 0{,}1459 * (R0/Rs)^{1{,}007} \quad (3)$$

- Para hacer la conversión a $ug/m^3$ se utilizan las ecuaciones (4) y (5):

$$CO_{ug/m^3} = 0{,}0409 * CO_{ppm} * PesoM_{CO} * 1000 \quad (4)$$
$$N0_{2ug/m^3} = 0{,}0409 * CO_{ppm} * PesoM_{NO_2} * 1000 \quad (5)$$

Donde $PesoM_{CO}$ es el peso molecular del $CO$ que es de 28.01 $g/mol$ y $PesoM_{NO_2}$ al peso molecular del $NO2$ que equivale a 46.0055 $g/mol$. Esta última conversión se realizará en el decodificador del servidor de red TTN porque los valores en **ppm** son menores por lo cual gastan menos bytes para enviarlos, lo cual se encuentra en la sección **7.4.1**.

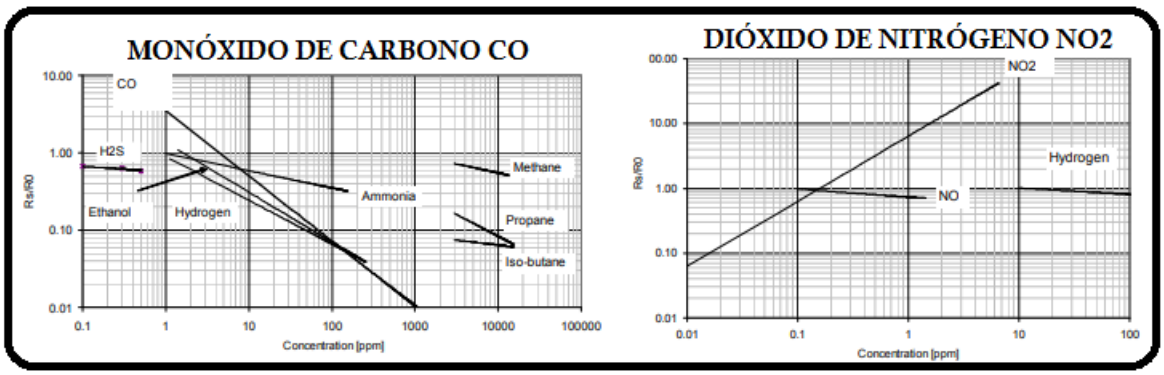



**Figura 7.17: Gráficas MICS6814**

Fuente: [23]

Dentro del código se deben definir los valores de **R0** para cada contaminante a leer y aplicar las ecuaciones para calcular la concentración en **ppm** que se va a enviar en la red.

#### 7.2.2.3. DSM501A: PM2.5 Y PM10

Para determinar los valores de $PM10$ y $PM2{,}5$ se hizo uso de la librería **dsm501-interrupt** que está en [75], esta librería interpreta los PWM entrantes y por medio de ecuaciones calcula la concentración de material particulado en $parts/283mL$, se realiza luego la conversión a $ug/m^3$ unidad que usa la norma. En el STM32 los pines de lectura son PB10 para $PM10$ y PB11 para $PM2{,}5$, en la figura 7.18 se muestra la explicación de los pines del sensor.

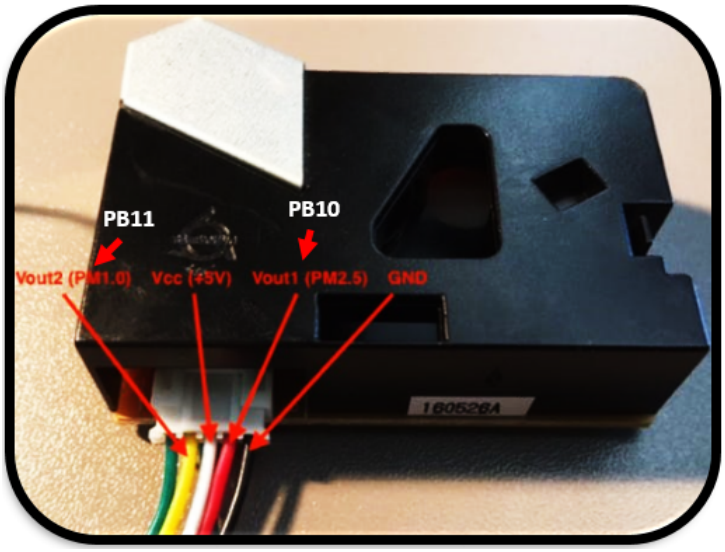



**Figura 7.18: Conexiones DSM501A**

Fuente: Autores

Para hacer la conversión de $parts/283mL$ a $ug/m^3$ simplemente basta con multiplicar por 0,002179 que corresponde al peso molecular de las partículas. En la figura 7.19 se muestra la salida del puerto serial utilizando la librería.

Como el sensor sólo mide con un pin partículas mayores a 1 *um* y con el otro mayores a 2.5 *um*, después de realizar las mediciones se hace la resta entre estos dos valores y de esta manera se determina el material particulado menor a 2.5 *um* que correponde a $PM2{,}5$. Para $PM10$ que es el material particulado menor a 10 *um* simplemente se toma el valor del pin que mide partículas mayores a 2.5 *um*.

### 7.2.3. Módulo LORABEE

Para poder hacer pruebas con el módulo LORABEE en una protoboard se tuvo que realizar un adaptador ya que los pines del módulo tienen una separación de 2 *mm* y la protoboard de 2.54 *mm*, fue necesario porque el módulo debe tener una conexión estable y sus cables no deben ser tan largos porque sino es muy posible que el microcontrolador no lo detecte. En la figura 7.20 se muestra la cara

```
PM1.0 particle count: 11.08 ug/m3 → Partículas superiore a 1 um
PM2.5 particle count: 1.95  ug/m3 → PM10: Partículas superiores a 2.5 um
PM1.0 ~ PM2.5 concentration: 9.13 ug/m3 → PM2.5: Partículas entre 1 y 2.5 um

PM1.0 particle count: 25.23 ug/m3
PM2.5 particle count: 6.41  ug/m3
PM1.0 ~ PM2.5 concentration: 18.82 ug/m3

PM1.0 particle count: 15.15 ug/m3
PM2.5 particle count: 3.72  ug/m3
PM1.0 ~ PM2.5 concentration: 11.43 ug/m3

PM1.0 particle count: 4.12  ug/m3
PM2.5 particle count: 0.00  ug/m3
PM1.0 ~ PM2.5 concentration: 4.12 ug/m3
```

**Figura 7.19: Mediciones DSM501A**

Fuente: Autores

superior e inferior de la baquela adaptadora diseñada.

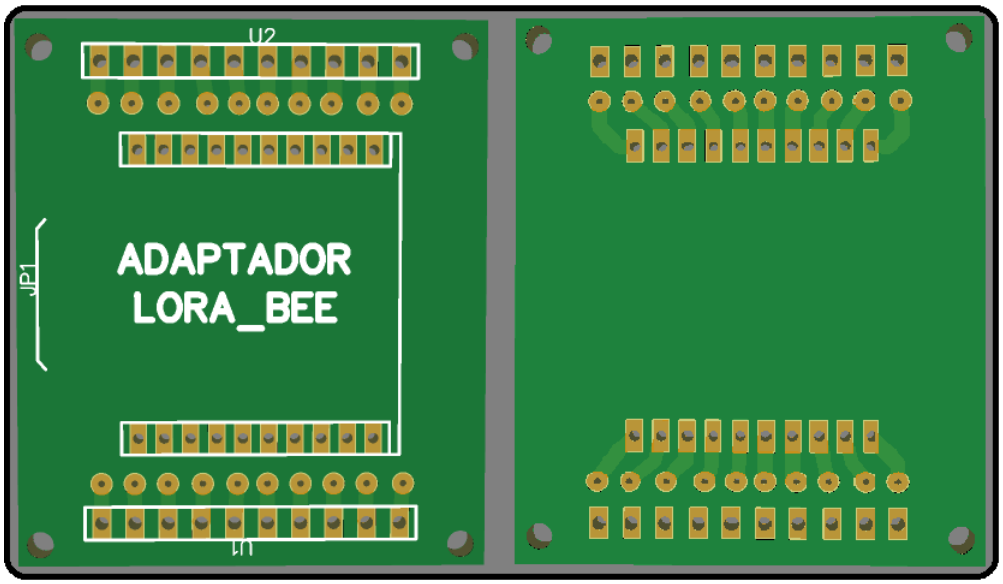



**Figura 7.20: PCB Adaptador LORABEE**

Fuente: Autores

En la figura 7.21 se muestra la comparación entre el módulo LORABEE sin adaptador y con adaptador.

El módulo LORABEE se conecta al microcontrolador por medio del protocolo SPI, además tiene otros pines llamados **DIO** que controlan cuando la antena es transmisora y receptora, que también permiten colocar al nodo en sueño profundo cuando no está trasmitiendo o recibiendo datos. En la

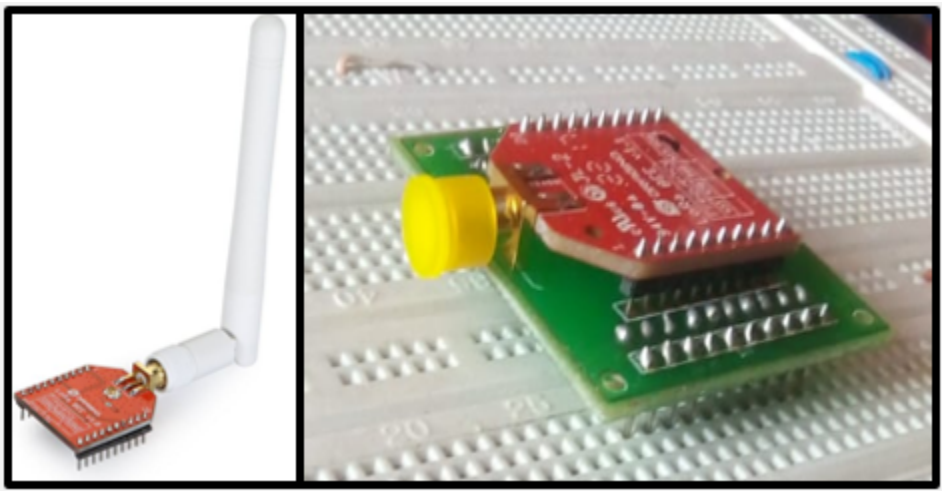


**Figura 7.21: Comparación módulo Lora: izquierda: sin el adaptador; derecha: con el adaptador**
Fuente: Autores

figura 7.22 se especifican los pines que van a ir conectados al puerto SPI1 del STM32 y dos de los pines **DIO** utilizados.

Para utilizar este módulo se hizo uso de una de las versiones de la librería **LMIC** encontrada en [77] desarrollada en su inicio por IBM en lenguaje C, que fue adecuada para utilizar el gateway elegido LG02.

Después de descargar e instalar la librería en arduino se tiene que abrir el archivo config.h, en este archivo se elige la frecuencia de trabajo, en este caso de 915 $MHz$, también la frecuencia de enlace ascendente, la frecuencia de enlace descendente, el factor de difusión y el ancho de banda teniendo en cuenta los valores de la tabla 5.10. Debido a las limitaciones del gateway como sólo se puede trabajar en una sub-banda se configuró el módulo con la sub-banda 1, que trabaja a una frecuencia de enlace ascendente de 903.9 $MHz$ con SF7 y BW de 125 $KHz$, y una frecuencia de enlace descendente de 923.3 $MHz$ con SF12 y ancho de banda de 500 $KHz$. Dichas configuraciones se muestran en la figura 7.23.

Además en el archivo **config.h** se debe especificar si se quiere que el radio sólo transmita o reciba o ambas funciones, para ello en este caso que requiere ambas funciones en **LMIC_DEBUG_LEVEL** debe colocarse **2**, además también se debe elegir el radio utilizado en este caso el transceptor **SX1276** y por último como caso particular para usar esta librería con el STM32 debe comentarse la línea de código que habilita **PRINTF** ya que esto sólo sirve para microcontroladores AVR como el arduino. Todas estas configuraciones adicionales se muestran en la figura 7.24.

Para la realización del código se hizo uso de la plantilla **ttn-abp** que se encuentra en los ejemplos de la librería, dentro de este código deben configurarse el puerto SPI a utilizar al configurar el pin **NSS**

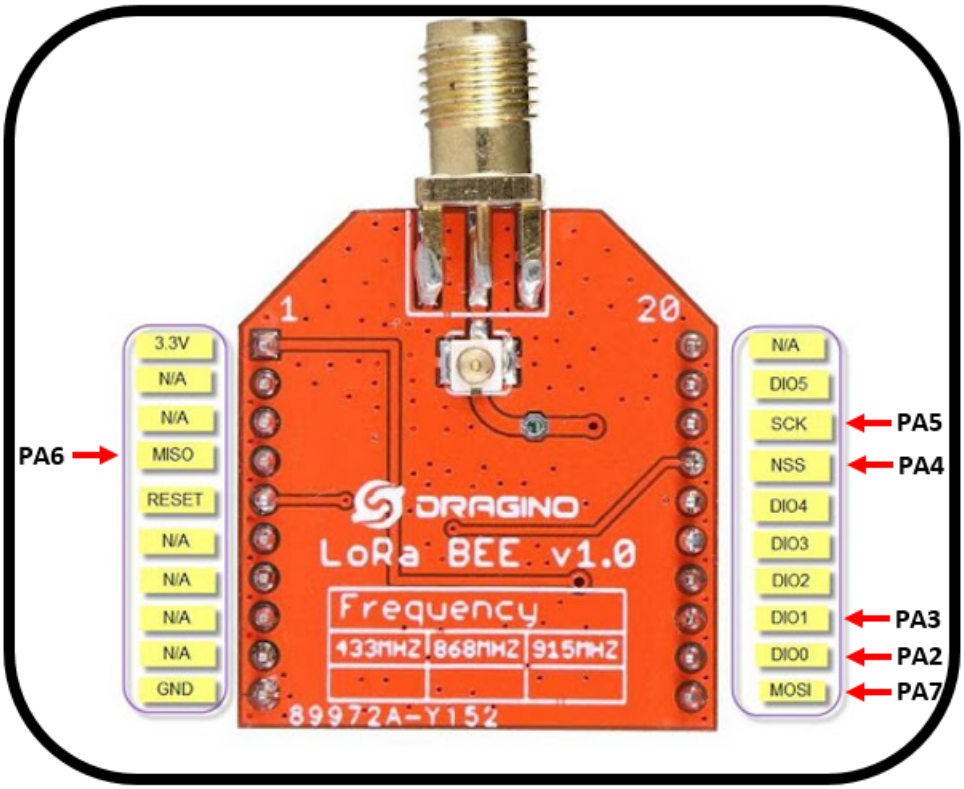



**Figura 7.22: Pines LORABEE**

Fuente: [76]

```
//#define CFG_eu868 1
#define CFG_us915 1        -> Cambio de Frecuencia a 915 MHz
//#define CFG_au921 1
//#define CFG_as923 1
//#define CFG_in866 1

#define LG02_LG01 1

//US915: DR_SF10=0, DR_SF9=1, DR_SF8=2, DR_SF7=3, DR_SF8C=4
//       DR_SF12CR=8, DR_SF11CR=9, DR_SF10CR=10, DR_SF9CR=11,
DR_SF8CR=12, DR_SF7CR
#if defined(CFG_us915) && defined(LG02_LG01)
// CFG_us915 || CFG_as923                  -> Frecuencia ascendente (nodo-gateway
#define LG02_UPFREQ    902320000             )y descendente (gateway-nodo) para la
//902320000                                   comunicación
#define LG02_DNWFREQ   923300000  -> Factor de difusión S7 F ascendente y S12 F
//923300000                                 descendente
#define LG02_RXSF      3          // DR_SF7   For LG01/LG02 Tx
#define LG02_TXSF      8          // DR_SF12CR For LG02/LG02 Rx
```

**Figura 7.23: Configuraciones iniciales archivo config.h de la librería LMIC**

Fuente: Autores

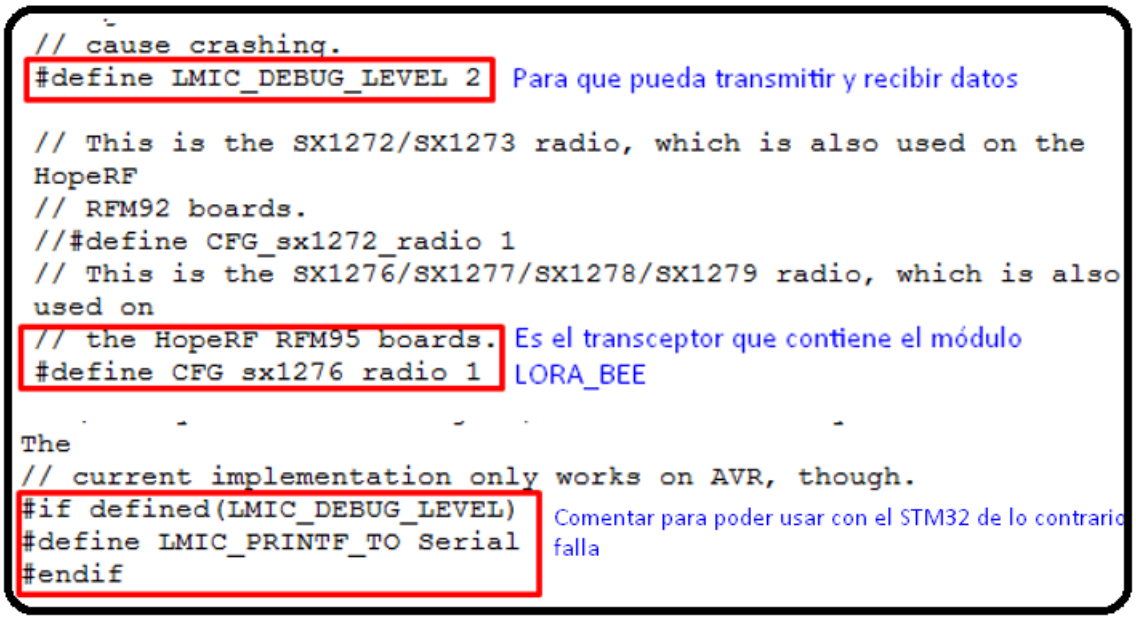



**Figura 7.24: Otras configuraciones archivo config.h de la librería LMIC**

Fuente: Autores

que corresponde en el STM32 al pin **PA4**, además se eligen dos pines para las entradas **DIO**, como se muestra en la figura 7.25 .

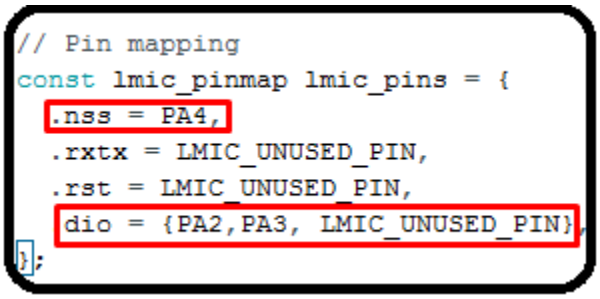



**Figura 7.25: Configuración pines en el IDE de arduino**

Fuente: Autores

Como es un nodo ABP este debe tener una clave de aplicación, clave de red y dirección del dispositivo para poder comunicarse con la red, en la sección **7.3.2** se especifica cómo se adquieren estos parámetros en el servidor de red que deben ser ingresados en el código del nodo. Además en el código de arduino se elige la sub-banda de trabajo en este caso la 1 y se comenta la línea **LMIC.dn2Dr = DR_SF9** que indica que en la segunda ventana de recepción del nodo se utilizaría SF9, esto dañaría la configuración interna ya hecha.

Después de la configuración para que el módulo funcione correctamente debe comprobarse que el arduino lo detecte, de lo contrario se producen errores que generalmente se atribuyen al archivo **radio.c** en la salida del puerto serial al intentar enviar datos. Para evitar estos errores se hace uso

del código que se encuentra en [78] donde sólo debe cambiarse el pin **NSS** del puerto SPI y tener el módulo conectado, no es aconsejable conectar los pines **DIO**. DEspués de cargar el código, en el puerto serial se indica si el módulo está bien conectado, de lo contrario puede que los cables estén demasiado largos o no tengan buena conexión con el microcontrolador, no se aconseja utilizar jumpers.

Después de configurar todo, si la transmisión del dato fue correcta en el puerto serial debe salir **EV_TXCOMPLETE (includes waiting for RX windows)**, esto indica que el nodo transmitió el mensaje y luego abrió dos ventanas de recepción, tal comportamiento, como se especifica en la teoría es de un nodo tipo A.

### 7.2.4. Sistema de almacenamiento externo: Memoria SD

La idea de la memoria en este tipo de red es almacenar los datos que no lleguen al gateway por pérdida de conexión y luego cuando se restablezca la conexión enviarlos de nuevo. Para realizar este proceso es necesario que el gateway envíe un mensaje de confirmación, es decir, un mensaje de enlace descendente. Como está especificado en la sección **5.8.5** por el servidor de red utilizado que es público y gratuito, y debido a que cuando el gateway transmite no puede recibir mensajes, hay una limitación con este tipo de mensajes a 10 diarios, lo que hace que la implementación de este tipo de memoria no sea viable.

Además de esta limitación, al usar las librerías de los sensores, el módulo LORABEE y la memoria SD, se sobrepasa la capacidad de memoria FLASH del microcontrolador como se ve en la figura 7.26. A pesar de que la memoria FLASH del STM32 es el doble de un arduino uno, debido a el espacio que ocupa el bootloader del STM32 y el núcleo instalado para su uso, este usa más espacio de la memoria FLASH que un arduino, por consiguiente para futuras implementaciones se aconseja usar el STM32CubeIDE.

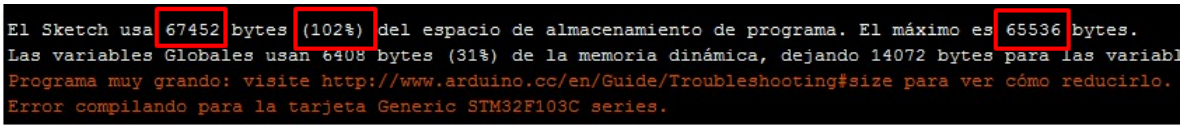



**Figura 7.26: Memoria FLASH sobrepasada**

Fuente: Autores

Sin embargo, a continuación se especifica el algoritmo propuesto que fue utilizado sólo usando el sensor BME280.

Para utilizar el módulo SD en el STM32 se tiene que hacer uso de la librería **SdFat-beta** que se encuentra en [79], con esta librería es posible elegir que la memoria use el puerto SPI2 del STM32, para ello se debe configurar como se muestra en la figura 7.27. Hay que tener en cuenta para que funcione que si se tiene un módulo SD con regulador interno debe conectarse a 5 $V$ y la velocidad del

reloj debe configurarse entre 1 $MHz$ y 16 $MHz$.

```
#include <SPI.h>
//#include <SD.h>
#include "SdFat.h"
static SPIClass mySPI2(2);
#define SD2_CONFIG SdSpiConfig(PB12, DEDICATED_SPI, SD_SCK_MHZ(16), &mySPI2)
SdFat SD;


#define SD_CS_PIN PB12
File myFile;


  if (!SD.begin(SD2_CONFIG)) {
```

**Figura 7.27: Configuración para usar módulo SD en puerto SPI2 en arduino IDE**

Fuente: Autores

En el código, primero hay que tener en cuenta que la librería **LMIC** tiene una función de reenvio, que se activa si en la plantilla **ttn-abp** utilizada, en la línea de código **LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 0)** que se encarga de poner en cola los datos a enviar, si en el último parámetro se coloca un **1** en lugar de un 0, esto hace que se active la opción de recibir un mensaje de confirmación o también llamado ACK por parte del gateway, sino se recibe dicho ACK se reenviará el mensaje las veces que sea necesario hasta restablecer la conexión. Este proceso funciona, sin embargo gasta mucha energía sin necesidad, por lo cual para el algoritmo fue descartado.

En la figura 7.28 se encuentra un diagrama de flujo que especifica el algoritmo implementado, en el cual si el mensaje no recibe confirmación se almacena en la memoria y hasta que haya de nuevo conexión con la red es de nuevo enviado. En el algoritmo hay que tener en cuenta que cuando se hace el envió de datos guardados en la memoria, el tiempo entre de datos enviados es más corto, que el tiempo entre datos cuando la conexión es estable. Para futuras implementaciones sería interesante colocar un reloj con el fin de saber la hora exacta de la toma de la muestra, en el STM32 es más sencillo de implementar ya que cuenta con una RTC incorporado, sólo se necesita de la batería externa.

Para enviar mensajes ACK programados, fue necesario utilizar el servidor de aplicación UBIDOTS, en el cual a través de una variable de control enviada desde nodo final, si esta cambia su valor, se dispara un evento que programa un mensaje de enlace descendente que se enviará cuando se reciba un nuevo mensaje ascendente y se abran las ventanas de recepción por parte del nodo final. Todo esto será explicado en la sección **7.3.4** ya que es necesaria más información para poder explicar este algoritmo.

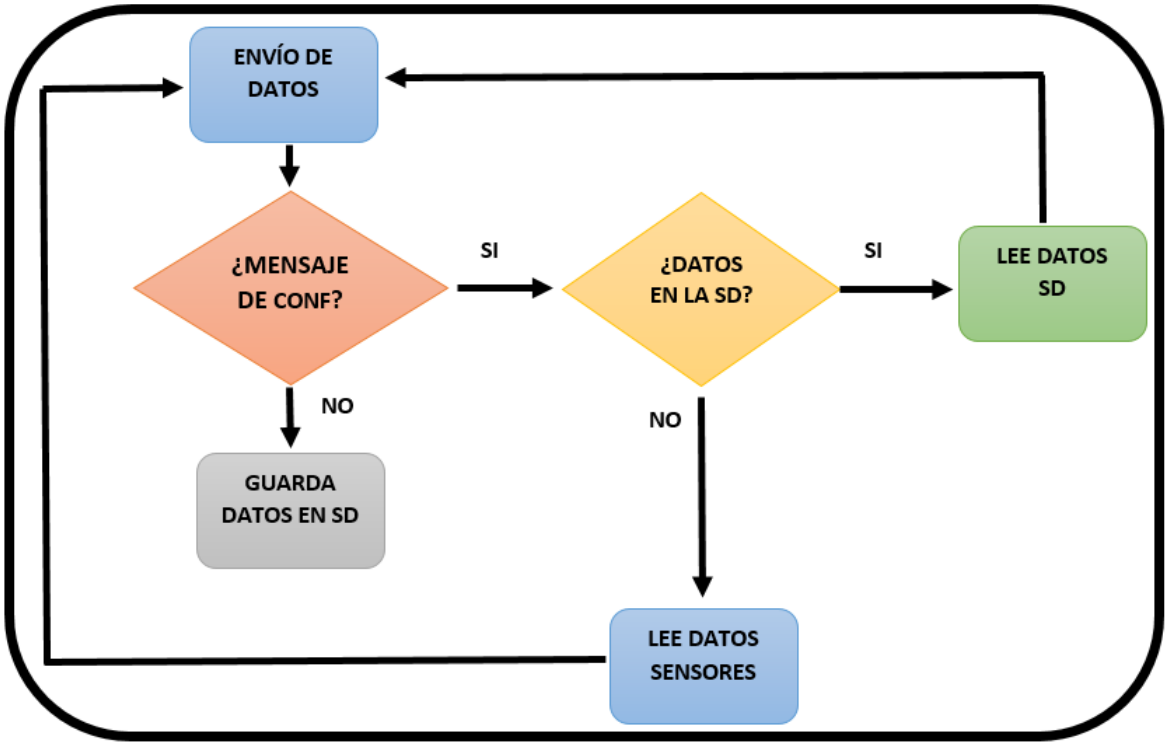



**Figura 7.28: Diagrama de flujo para implementación del algoritmo con la memoria SD**

Fuente: Autores

### 7.2.5. Código nodo final

Después de haber especificado todo lo necesario para la implementación del nodo final, en esta sección se explica el código con el cual estarán programados los nodos, para ellos en la figura se hace un resumen de todas las partes importantes de este código y en el anexo **11.1** se encuentra el código completo implementando todos los sensores, con los comentarios debidos.

Es importante aclarar que para el envío de los datos de los sensores se usó un arreglo de Bytes porque es la forma como se obtiene un payload pequeño, los datos son codificados en la cantidad de Bytes que requieren de acuerdo a su valor máximo, en la sección **7.4.1** se explica cómo estos estos datos son decodificados en el servidor de red.

Como fue especificado antes, el código está basado en el ejemplo ttn-abp de la librería **LMIC**, donde a grandes rasgos se incluyeron las librerías, pines y variables de los sensores implementados.

Cabe aclarar que hay dos códigos finales debido a que uno de los sensores de gas desafortunadamente se dañó y por circunstancias actuales no fue posible conseguir uno, por lo cual en el anexo **11.2** se coloca el código que tiene un emulador del sensor de gas teniendo en cuenta los valores de salida del mismo.

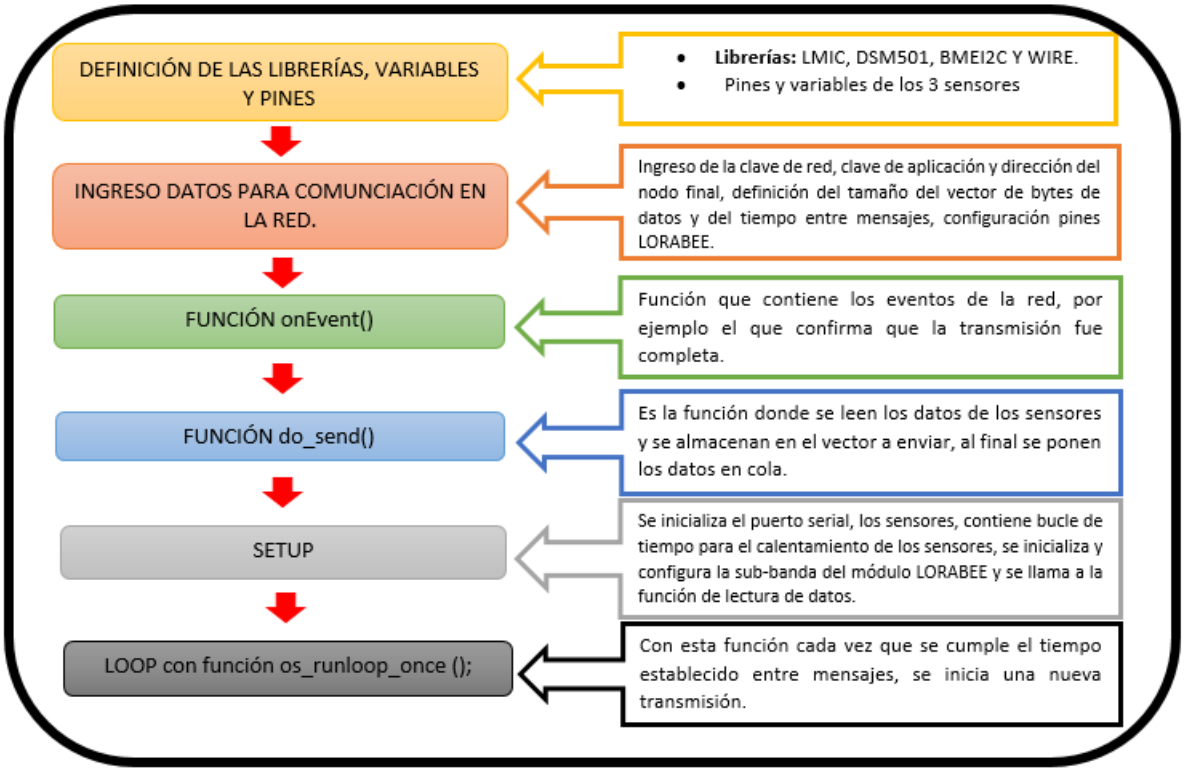



**Figura 7.29: Organización y explicación de las partes del código del nodo final**

Fuente: Autores

## 7.2.6. Alimentación

### 7.2.6.1. Análisis de potencia

Para determinar el consumo de corriente del prototipo teniendo en cuenta la información encontrada en [80], se coloca entre el pin positivo de la fuente de alimentación y el pin del microcontrolador que recibe esta fuente, una resistencia pequeña entre 1 *Ohm* y 10 *Ohm* y con otro sistema de medición, en el caso del proyecto, otro microcontrolador que funciona como osciloscopio se mide el voltaje sobre dicha resistencia, como se muestra en la figura 7.30 y usando la ley de Ohm se calcula la corriente, tenga en cuenta que la fuente de alimentación del nodo y del microcontrolador que funciona como osciloscopio deben ser diferentes para evitar un corto.

Con los datos obtenidos y a partir de la siguiente ecuación se calcula la corriente IRMS o promedio de consumo del dispositivo cuando todo está conectado teniendo en cuenta los momentos cuando el nodo trasmite y no trasmite, usando la ecuación (6).

$$IRMS = \sqrt{\frac{1}{N} \cdot \sum_{n=1}^{N} {I_n}^2} = \sqrt{\frac{{I_1}^2 + {I_2}^2 + \ldots + {I_N}^2}{N}} \text{ con } N = \text{cantidad de datos} \quad (6)$$

Los datos obtenidos se muestran en la figura 7.31, donde al hacer el cálculo con la ecuación (6) se obtuvo que el consumo de corriente promedio es de 184.5 *mA* y tiene un pico de corriente en la

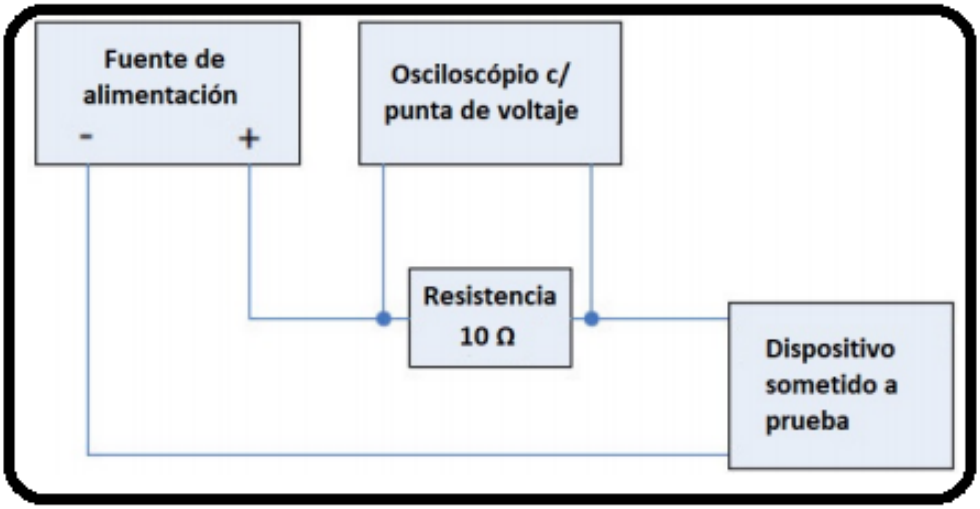



**Figura 7.30: Circuito medición de corriente**

Fuente:[80]

transmisión de 380 *mA*, este consumo se esperaba más alto en sus inicios ya que se pretendía usar el módulo SD que tiene un consumo de 200 *mA*.

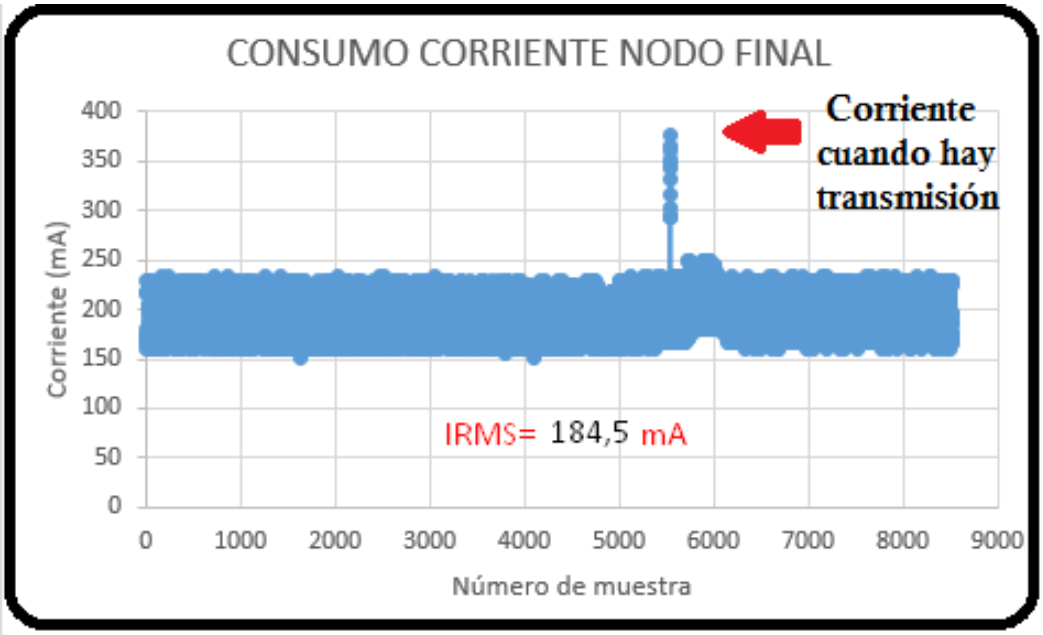



**Figura 7.31: Resultados análisis de corriente**

Fuente: Autores

#### 7.2.6.2. Sistema de alimentación implementado

El sistema de alimentación tiene dos partes fundamentales, la primera la alimentación por 3 baterías recargables de litio 18650 que fueron recicladas de la batería de una computadora antigua y tienen la capacidad de 2200 *mA*; y la segunda opción es la alimentación fija en una toma corriente con

un cargador mayor a 5 $V$ y corriente superior a los 400 $mA$ según el análisis de corriente realizado, en este caso se usó un cargador de 12 $V$/1 $A$ que además carga las baterías. Para controlar la conmutación entre los dos sistemas, además de la carga y descarga de las baterías se hizo uso de un BMS de 3 celdas, que corta la corriente cuando las baterías están cargadas y quita el voltaje al sistema cuando alguna de las celdas ha llegado al voltaje nominal.

Después de estás fuentes de alimentación se encuentra el reductor de voltaje LM2596, que mantiene el voltaje de salida en 5 $V$ si su entrada es superior a este valor, es más estable que los reguladores comunes y como trabaja por conmutación no requiere de disipador cuando el consumo de corriente es bajo.

Además de lo mencionado se cuenta con un medidor de capacidad de batería que indica un estimado de qué tanta carga tiene el nodo. El sistema de alimentación se resume en la figura 7.32.

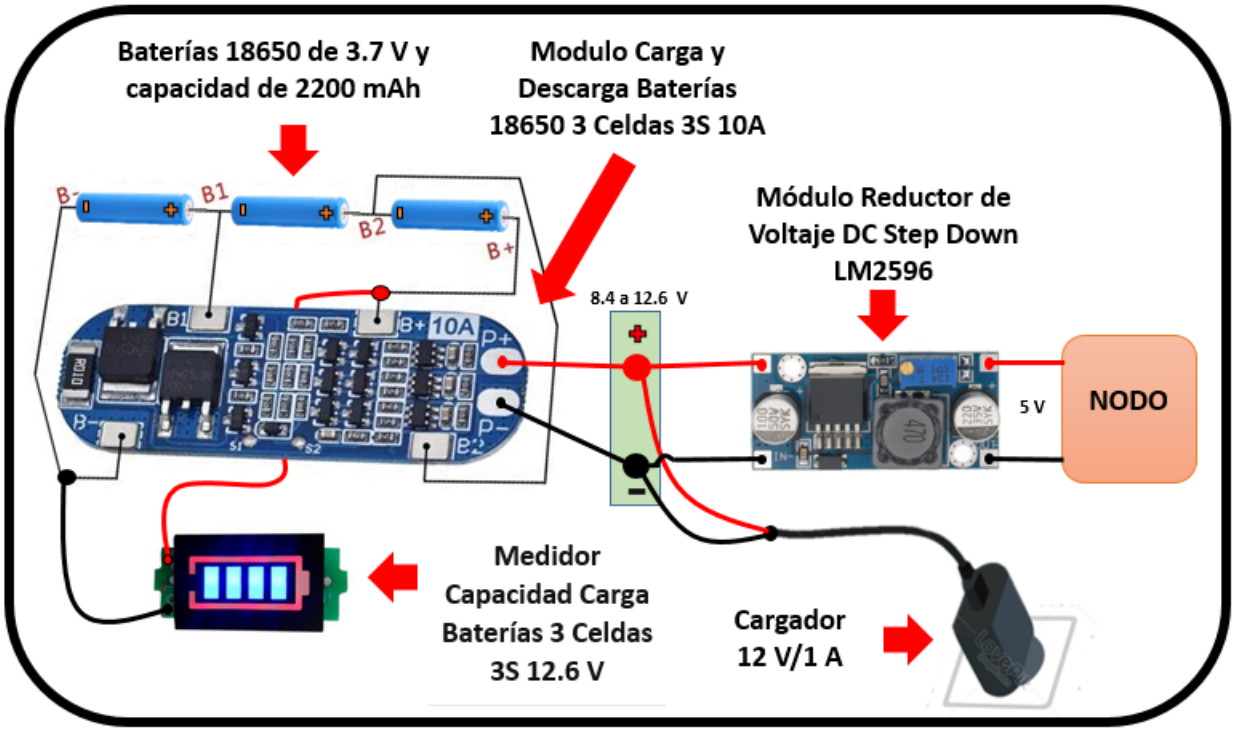



**Figura 7.32: Sistema de alimentación**

Fuente: Autores

### 7.2.6.3. Tiempo de carga y descarga de las baterías

Analizando el patrón de carga y descarga de las baterías se llegó a la conclusión que con las baterías usadas se tiene:

- **Tiempo de carga**: Aproximadamente 5 horas.

- **Tiempo de descarga**: Aproximadamente 14 horas.

Estos resultados pueden mejorarán si se compran baterías nuevas y de mayor capacidad.

### 7.2.7. Adecuación del prototipo en una caja

Debido al tipo de sensores utilizados que deben estar en contacto con el ambiente, la caja de protección va a ser destinada al microcontrolador, módulo LORABEE y el regulador LM2596. Para poder colocar una protección a los sensores que les permitan estar en el ambiente y tener protección contra la lluvia, debe hacerse un estudio más profundo de los vientos, cosa que se propondría como una mejora en proyectos futuros.

Por lo anterior fue necesario el diseño de 2 baquelas, una que contenga lo que va al interior de la caja y otra donde estén ensamblados los sensores. La baquela de los sensores es insertada en la baquela principal por medio de una regleta. Para el diseño de las baquelas se usó el programa **EasyEDA** que trabaja en línea.

El esquemático de las conexiones y el dibujo de la PCB principal se muestran en las figuras 7.33 y 7.34, respectivamente. Estaba baquela tiene un tamaño de 8.7 *cm* x 7.4 *cm*, por su tamaño y debido al espacio que ocupan cables y el resto de componentes del sistema de potencia, se eligió una caja con 10 *cm* de ancho, 10 *cm* de largo y 6 *cm* de alto.

En las figuras 7.35 y 7.36 se muestra el esquemático de conexiones y el dibujo de la PCB de la baquela que lleva a los sensores, respectivamente.

En la parte interior e inferior de la caja se ubicó el resto del sistema de potencia, como lo son las baterías, el BMS, el indicador de batería y el conector hembra para el cargador de 12 V, además de esto se instaló un interruptor que encenderá y apagará el nodo, y también un pulsador para activar el indicador de batería cuando quiera ser consultado el estado de la misma, como se ve en la figura 7.37.

En las figuras 7.38 y 7.39 se encuentran fotografías de cada una de las placas y el ensamblaje del nodo final.

Cabe aclarar que como se ha informado uno de los nodos no tiene el sensor de gases, sin embargo su baquela de sensores fue la misma, para que a futuro este sensor sea reemplazado y puedan realizarse más pruebas con los dos nodos.

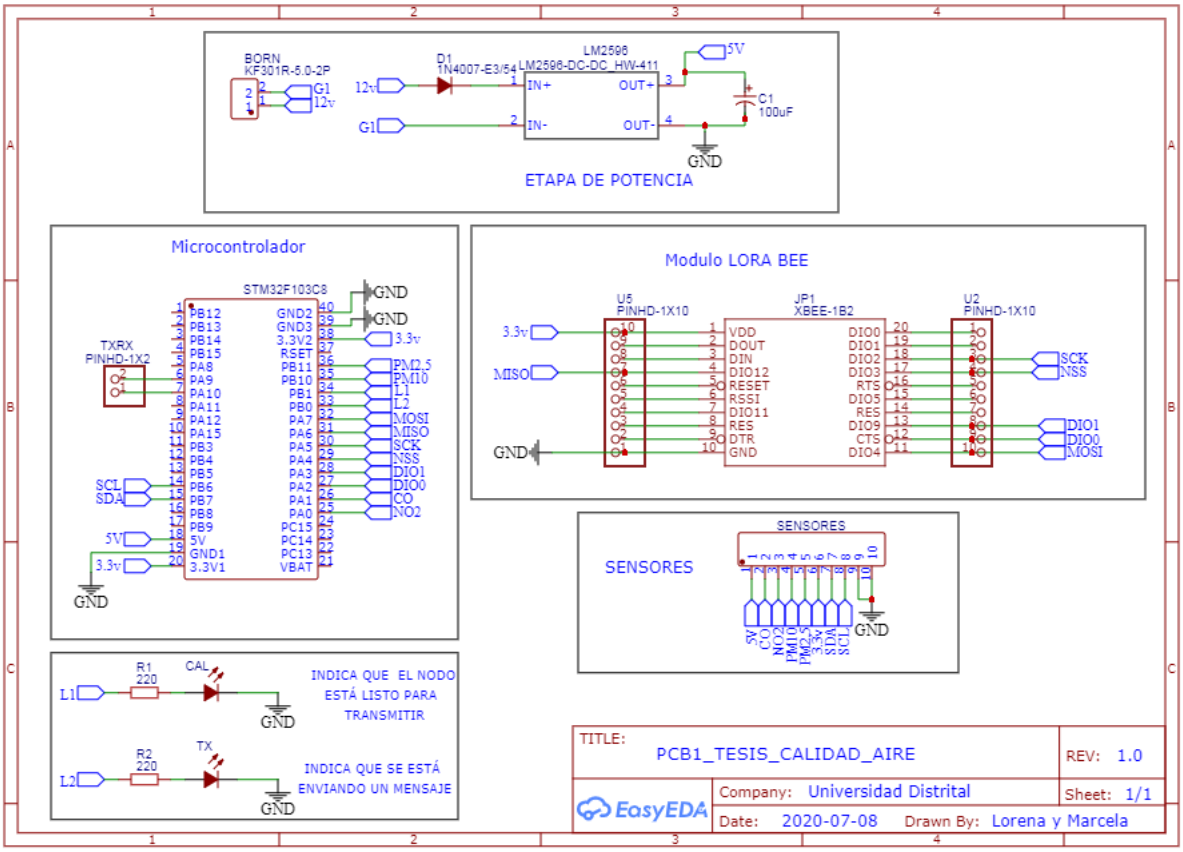



**Figura 7.33: Esquemático baquela principal**

Fuente: Autores

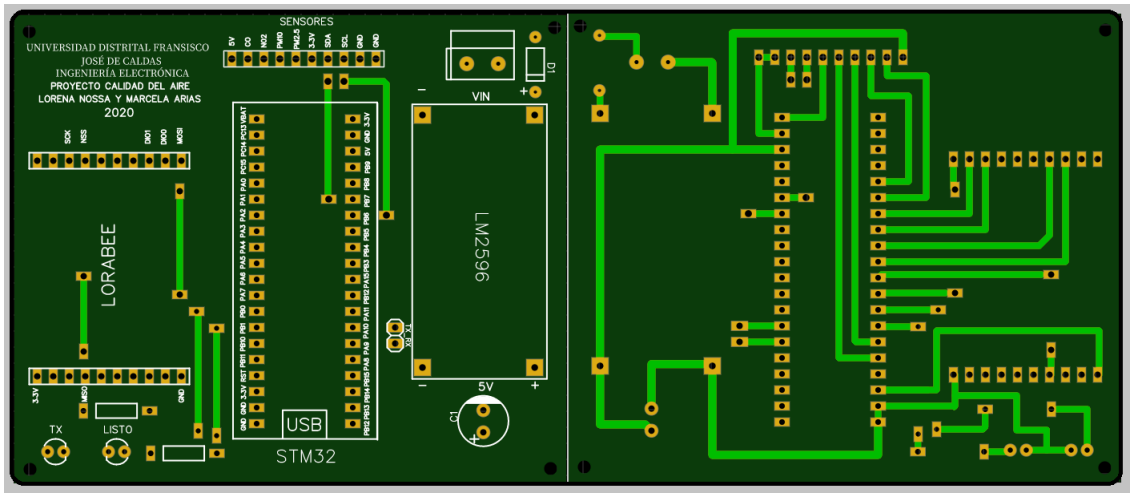



**Figura 7.34: Dibujo baquela principal**

Fuente: Autores

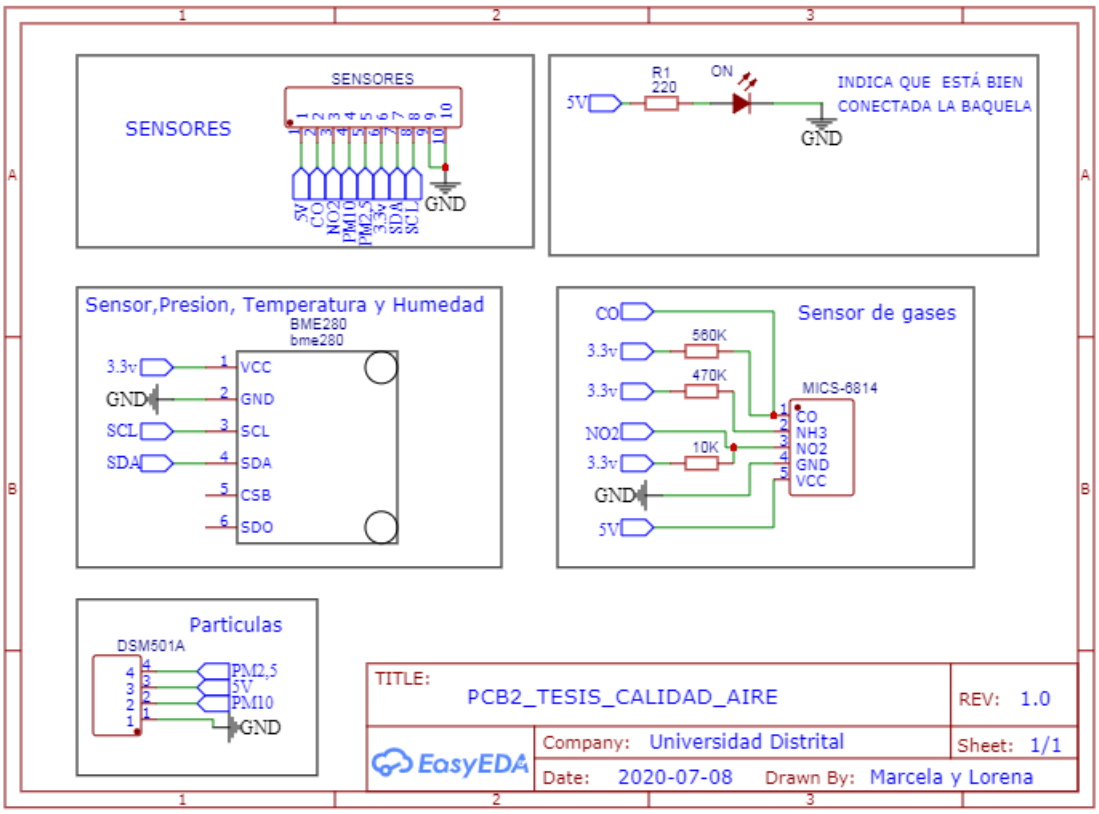



**Figura 7.35: Esquemático baquela sensores**

Fuente: Autores

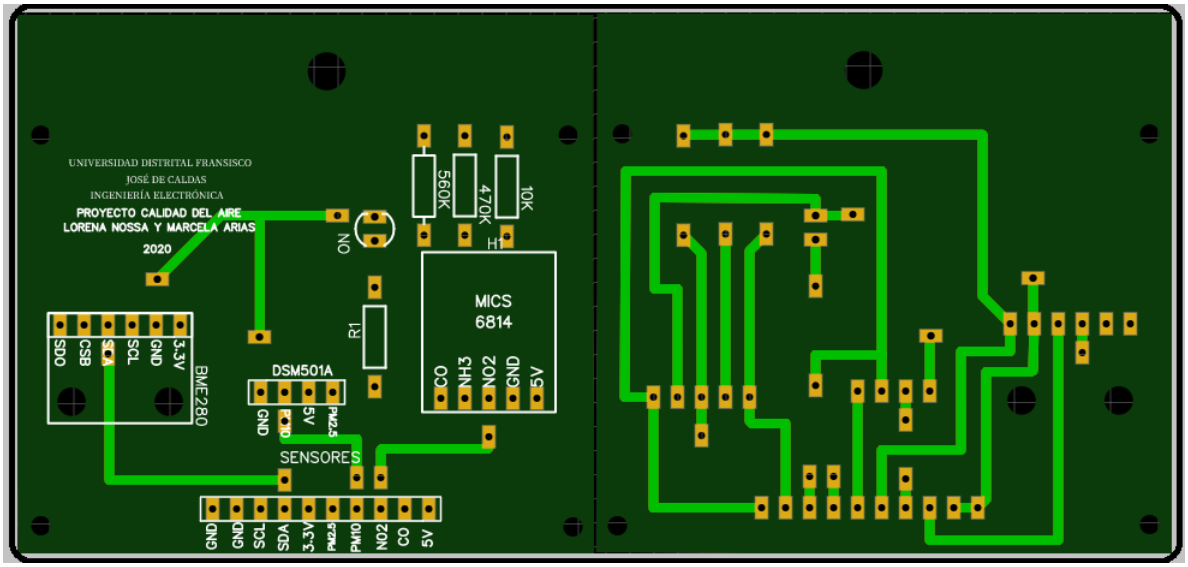



**Figura 7.36: Dibujo baquela sensores**

Fuente: Autores

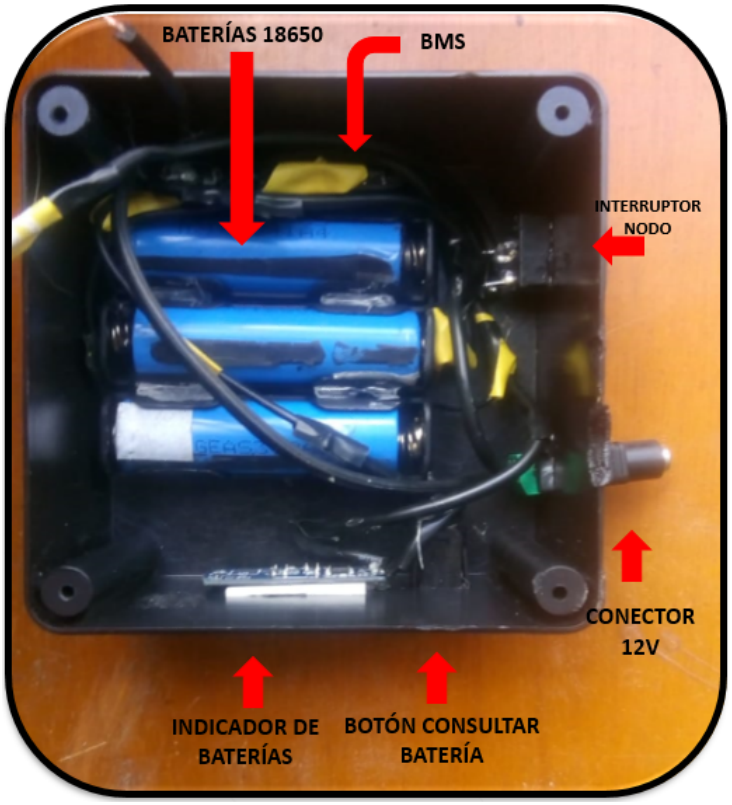



**Figura 7.37: Sistema de potencia en la caja**

Fuente: Autores

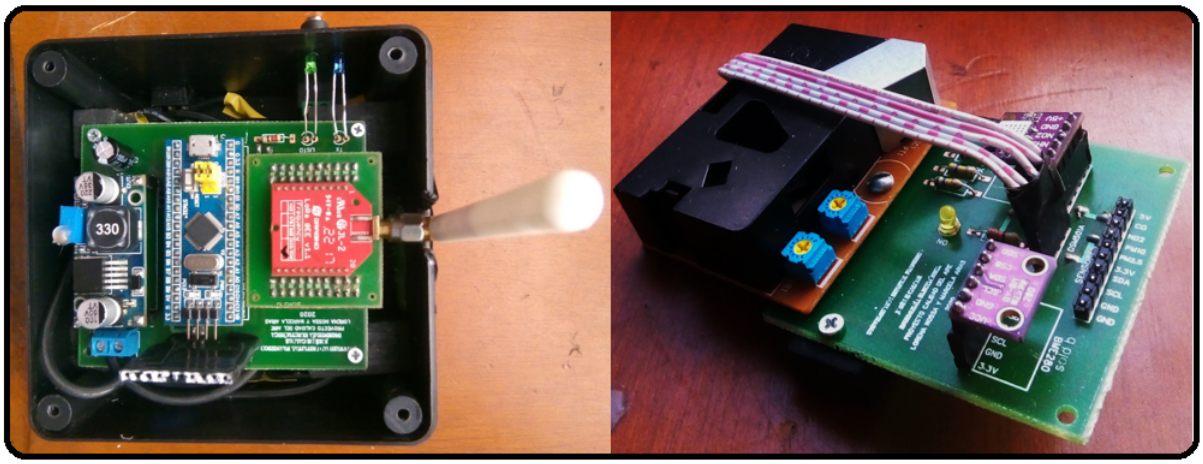


**Figura 7.38: Fotografía baquelas ensambladas. Izquierda: baquela principal y derecha baquela con los sensores**

Fuente: Autores



**Figura 7.39: Fotografías nodo con y sin baquela de los sensores.**

Fuente: Autores

## 7.3 Conexión nodo final con el gateway

### 7.3.1. Configuración del gateway con servidor de Dragino [53]

Para poder configurar el gateway LG02 por medio del servidor de dragino en el computador se puede conectar de varias formas, por defecto está configurado como un AP WiFi, por esta razón en el primer arranque del LG02 se generará automáticamente una red Wifi llamada: **dragino-xxxxxx**, como se ve en la figura 7.40. Después de conectarse a la red Wifi hay que ingresar en el navegador la url: `http://10.130.1.1/cgi-bin/luci/admin` .




**Figura 7.40: Acceso al Gateway**

Fuente: [53]

Allí para ingresar a la interfaz del LG02 como se ve en la Figura 7.41 se introducen los siguientes datos:

- **Nombre de usuario:** root
- **Contraseña:** dragino




**Figura 7.41: Cuenta de acceso al Gateway**

Fuente: [53]

Sin embargo, al conectarse de esta manera el computador pierde el acceso a internet por Wifi, por lo que no es recomendable, lo más conveniente es conectar el gateway al computador por el puerto LAN de ambos dispositivos utilizando un cable de red, esta fue la opción elegida para el proyecto, se comprueba la conexión si se puede ingresar a la interfaz del LG02 de dragino.

También es necesario que el gateway esté conetado a internet, para lo cual existen varias opciones:

- Puerto WAN Modo de Internet
- Modo cliente WiFi
- Modo WiFi AP
- Modo de marcación USB

Para el desarrollo del prototipo se eligió conectar el gateway a internet por el puerto WAN a través de un cable de red entre el gateway y el router de internet, ya que opr defecto el gateway está configurado para usar el puerto WAN como conexión de red y es más estable. Cuando se conecta el puerto WAN de LG02 al enrutador, LG02 obtendrá la IP del enrutador y tendrá acceso a Internet. El estado de la red puede comprobarse como se ve en la Figura 7.42.

Luego se hizo uso de la página de diagnóstico para comprobar y analizar la conexión a Internet, ingresando en la pestaña **Network** y realizando **ping** como se ve en la Figura 7.43.

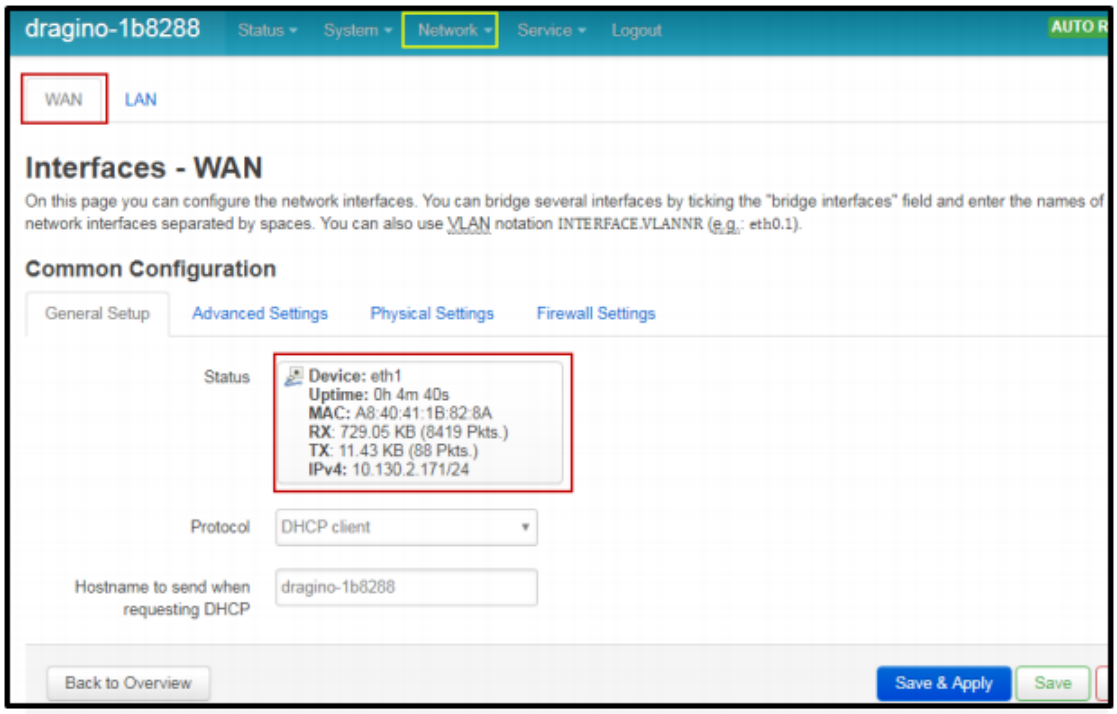



**Figura 7.42: Comprobando el estado de la red**

Fuente: [53]

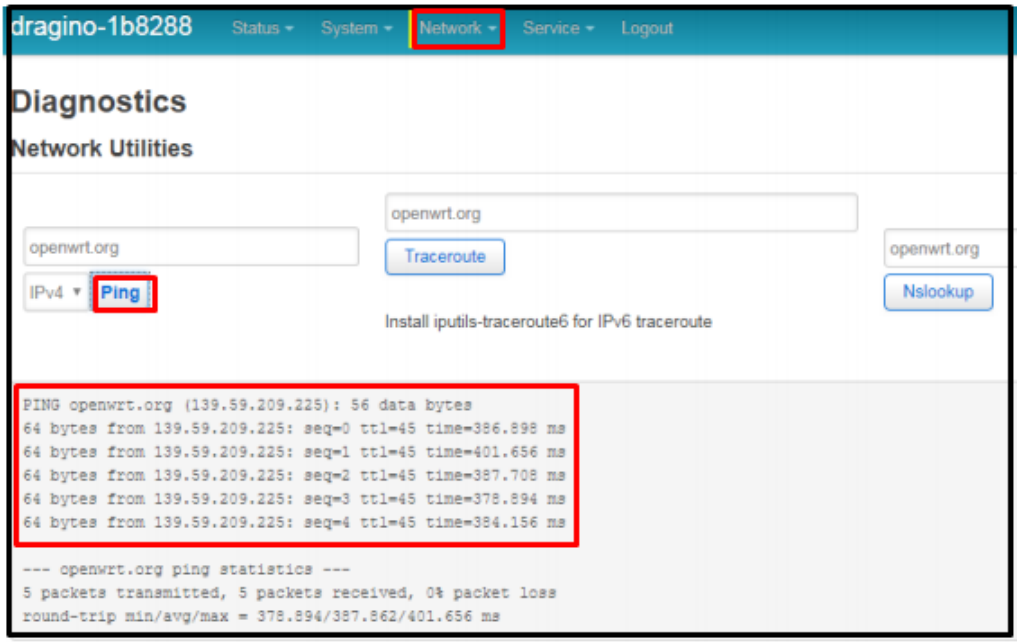



**Figura 7.43: Análisis de la conexión a internet**

Fuente: [53]

Como ya se tiene acceso a la interfaz del gateway y está conectado a internet se procede a configurarlo para hacer uso del protocolo LORAWAN. Primero se ingresa en el menú **servicio** y luego en la opción **LoRaWAN GateWay**, donde se especifica que el gateway trabaja en LORAWAN, se elige el servidor red que en este caso es TTN, la zona de trabajo **us-west** que indica que se trabaja a 915 $MHz$ al igual que en Estados Unidos, el identificador único, el correo del propietario y la ubicación del gateway como se muestra en la figura 7.44. El identificador único del gateway está en la cara de abajo del mismo.

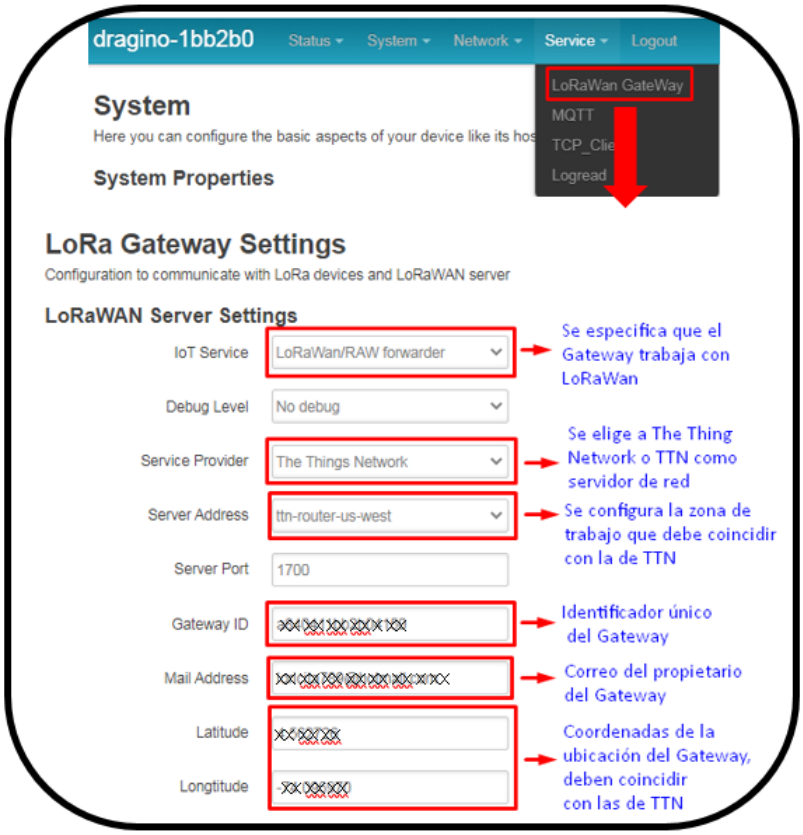



**Figura 7.44: Configuración de LoRaWAN en el Gateway**

Fuente: Autores

Como el gateway trabaja a una única frecuencia, se configuran los dos canales a la frecuencia de enlace ascendente que fue elegida e ingresada en el código del nodo final en la sección **7.2.3**, además de los otros parámetros de red como el factor de difusión, tasa de codificación y ancho de banda. Esto se muestra en la figura 7.45.

### 7.3.2. Configuración del servidor de red TTN

Como primer paso para hacer uso del gateway con el servidor TTN se creó un usuario en esta página usando un correo propio del proyecto como se ve en la Figura 7.46.

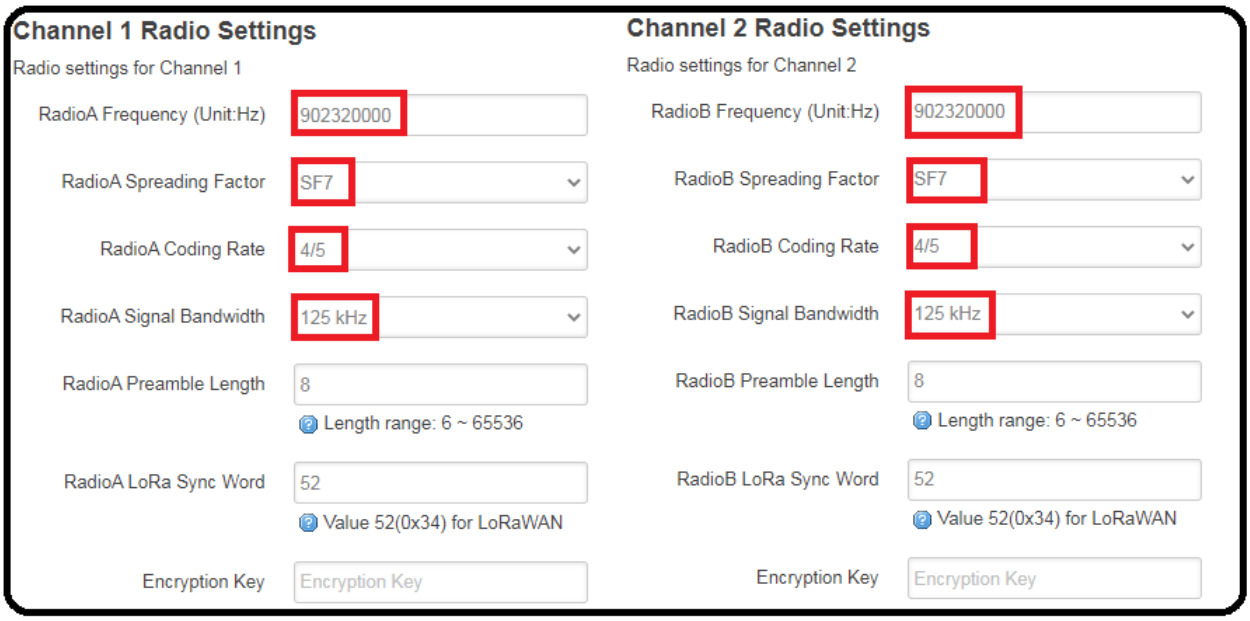



**Figura 7.45: Configuración de los dos canales del gateway LG02**

Fuente: Autores




**Figura 7.46: Registro de nuevo usuario en el servidor TTN**

Fuente: Autores

El segundo paso a seguir es registrar el gateway en la plataforma, para ello ingresamos al menú **Consola**, se elige **Gateways** y por último **Register Gateway**. Como tercer paso en la ventana que se abre se ingresa el identificador único del gateway que se encuentra impreso en el aparato, se activa el reenviador de paquetes que toma las tramas LORAWAN de los nodos y las convierte a UPD/IP para el servidor de red, se escribe el nombre con el que se identificará el gateway en la plataforma y se configura la frecuencia como se observa en la figura 7.47. Además de la anterior configuración en el

mapa que aparece abajo del formulario se colocan los datos de longitud y latitud donde está ubicado el gateway.




**Figura 7.47: Registro del Gateway en TTN**

Fuente: Autores

El cuarto paso es verificar que los parámetros del registro son correctos, para ello después de encender el gateway y verificar que esté conectado a internet, se ingresa en TTN al gateway registrado y debe obtenerse una respuesta como la de la figura 7.48. Además después de configurar la ubicación del gateway y si este se encuentra encendido, al buscar los gateway de Bogotá en TTN puede ser localizado como se ve en la figura 7.49.

El quinto paso consiste en crear una aplicación donde se registrará el nodo ingresando, para ello de nuevo se ingresa en **Consola**, pero se elige esta vez **Aplicaciones** y **agregar aplicación**, aquí deben ingresarse el nombre de la aplicación, descripción de la aplicación y la zona de trabajo que debe coincidir con la configurada en el gateway como se observa en la figura 7.50.

Como sexto paso se procede a registrar el nodo para ello después de ingresar a la aplicación crea-




**Figura 7.48: Gateway correctamente registrado y conectado EN TTN**

Fuente: Autores

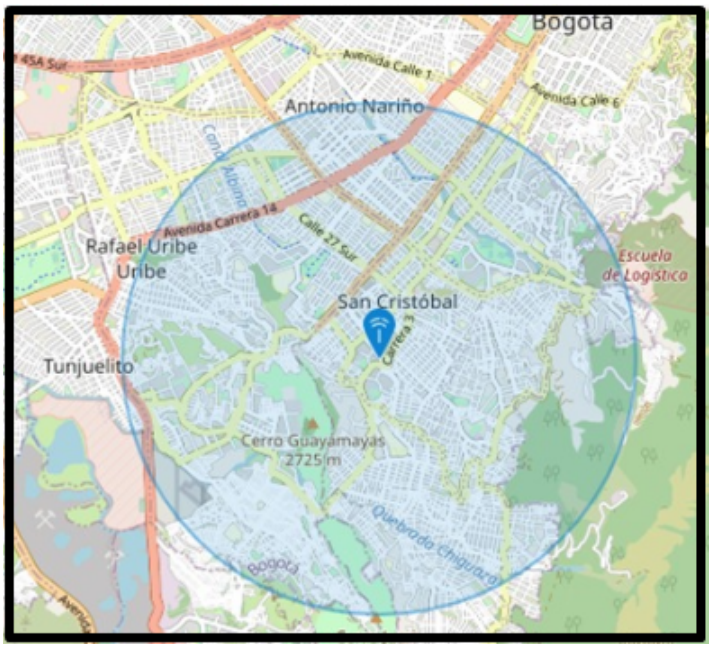



**Figura 7.49: Ubicación del Gateway**

Fuente: Autores

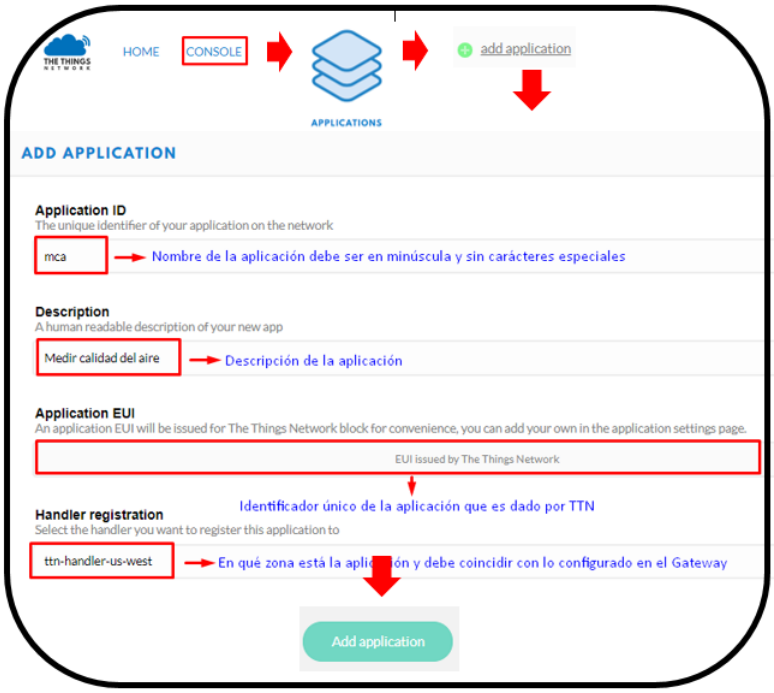



**Figura 7.50: Creación de una aplicación en TTN**

Fuente: Autores

da se ingresa a **Dispositivos** y se elige **Crear nuevo dispositivo**, donde deben llenarse los campos: nombre único del nodo, se presionan las flechas para generar un identificador único para el nodo, la clave para la comunicación entre el nodo y la red, la puede generar TTN o colocarla quien maneja la aplicación y el identificador de la aplicación que fue generado por TTN que asocia el nodo con la aplicación como se ve en la figura 7.51 .

En el séptimo paso se debe configurar el nodo creado en el modo de activación ABP, primero al entrar en el nodo se elige el menú **Configuraciones**, donde se puede ingresar descripción del nodo, se muestra identificación única del nodo y identificación única de la aplicación dadas por TTN, luego se configura el método de activación del nodo como ABP donde al elegir esta opción es mostrada la dirección única del nodo, clave de sesión de red y clave de la aplicación. Por último se desmarca la opción **Frame counter checks** para que se reinicien los contadores de mensajes después de una desconexión, todo esto es mostrado en la figura 7.52.

La dirección única del nodo en inglés **Device Address**, la clave de la sesión de red en inglés **Network Session Key** y la clave de sesión de red en inglés **App Session Key** pueden encontrarse en el nodo en TTN después de configurarlo como ABP, estos datos deben ser ingresados en el código de arduino del nodo final, para poder establecer la conexión con la red. La **Network Session Key** y **App Session Key** en el código deben ser ingresadas en formato **msb**, que se obtiene al dar en las flechas en TTN como se muestra en la figura 7.53.




**Figura 7.51: Registro de un dispositivo en TTN**

Fuente: Autores

### 7.3.3. Pruebas de conexión del nodo final con el gateway

Después de configurar el gateway, el servidor de red y el nodo final, se procede a hacer la prueba de su interconexión, para ello TTN permite visualizar los mensajes recibidos en el tráfico de mensajes del Gateway, obteniendo un resultado como el mostrado en la figura 7.54. Se puede ver que en el tráfico del gateway se muestran los mensajes de enlace ascendentes y descendentes con sus respectivas configuraciones de frecuencia, tasa de codificación, factor de difusión y ancho de banda, además se muestran otros datos como el tiempo del mensaje en el aire y el tamaño de la trama.

En la aplicación de TTN, al entrar a cada nodo se encuentra el menú de datos donde también pueden comprobarse la recepción de los mensajes, el número de mensaje enviado, puerto por donde se envía el mensaje, el mensaje o payload codificado y también el mensaje decodificado que muestra la información de los sensores, en la sección **7.4.2** se explica cómo se decodifican los datos. Lo anterior se ve en la figura 7.55.

Además de estos datos en cada mensaje pueden verse los metadatos, que corresponden a la potencia de la señal o RSSI, relación señal a ruido o SNR, ubicación del gateway y tiempo del aire, entre los más importantes, como pueden verse en la figura 7.56.




**Figura 7.52: Resumen del dispositivo (identificaciones únicas del dispositivo) en TTN**

Fuente: Autores




**Figura 7.53: Dirección del dispositivo y claves nodo ABP**

Fuente: Autores




**Figura 7.54: Tráfico del Gateway cuando se establece conexión con el nodo final**

Fuente: Autores




**Figura 7.55: Datos en la aplicación TTN**

Fuente: Autores

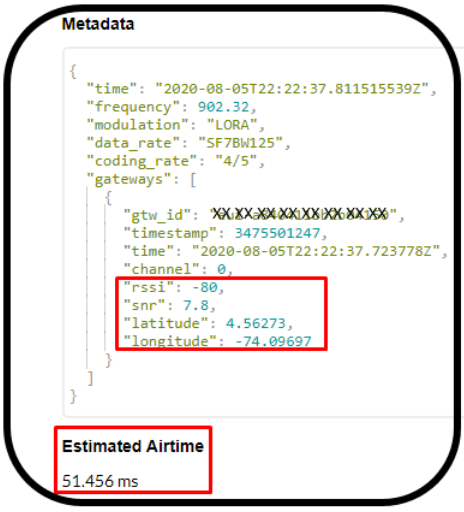



**Figura 7.56: Metadatos en TTN**

Fuente: Autores

Teniendo en cuenta el tiempo en el aire del mensaje de los nodos y las limitaciones de la plataforma TTN en la figura 7.57, se estima la cantidad de mensaje diarios que enviaría cada uno de los nodos.




**Figura 7.57: Número de mensajes diarios de cada nodo.**

Fuente: Autores

### 7.3.4. Prueba algoritmo de memoria SD con Ubidots y TTN

Para realizar esta prueba se usó solamente el sensor de temperatura, presión y humedad, además de una nueva variable que cambiaba entre 0 y 1, con esta variación cada vez que en UBIDOTS se reportaba este cambio se ponía en cola un mensaje de enlace descendente. El código del nodo para

implementar el algoritmo se encuentra en anexo **11.3** y la explicación del algoritmo en la figura 7.58.

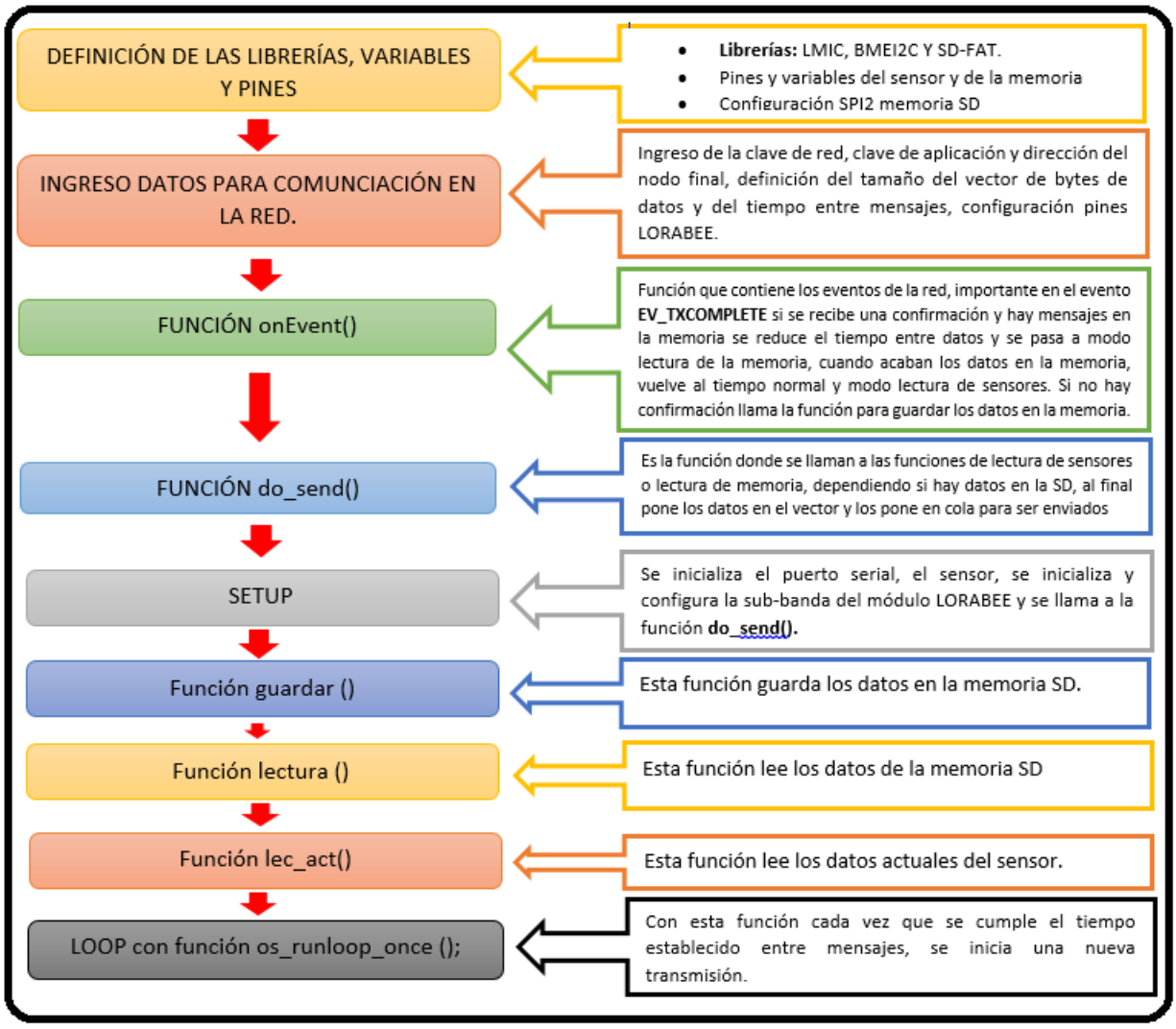



**Figura 7.58: Explicación código arduino memoria**

Fuente: Autores

Para entender mejor cómo funciona la función **guardar()** y **lectura()**, en la figura 7.59 se muestran cómo son almacenados los datos en el archivo **txt**.

Para realizar la conexión entre UBIDOTS y TTN se hace una integración por medio de HTTP siguiendo los pasos que se muestran en [81], cabe aclarar que hay otra forma de integrar las plataformas ya que TTN tiene convenio con UBIDOTS como se muestra en [82], pero con esa configuración no es posible enviar mensajes Downlink. Después de que ambas plataformas estén conectadas, es decir que UBIDOTS reciba los datos de TTN, se procede luego a configurar el evento downlink.

Para enviar mensajes de downlink programados fue necesaria la información encontrada en [83] y

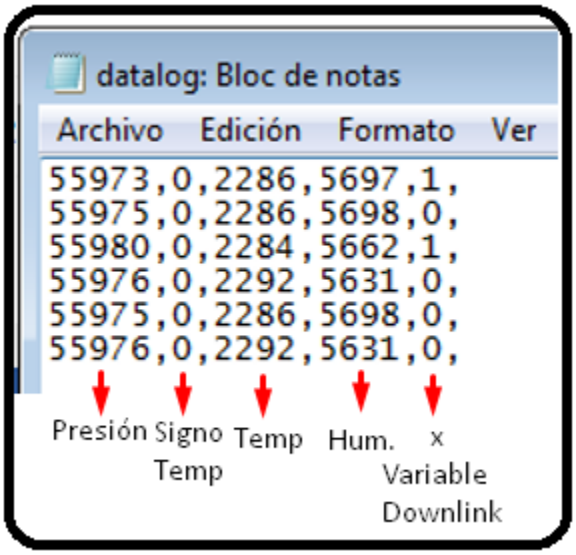



**Figura 7.59: Formato datos guardados en archivo txt**

Fuente: Autores

[84]. Primero se ingresa en el menú datos donde se elige eventos, se da en nuevo vento y en la parte **if triggers** se selecciona la variable de control, que varía entre 0 y 1 y se crea el condicional que dispara el evento cuando esa variable es igual a 1, como se ve en la figura 7.60.

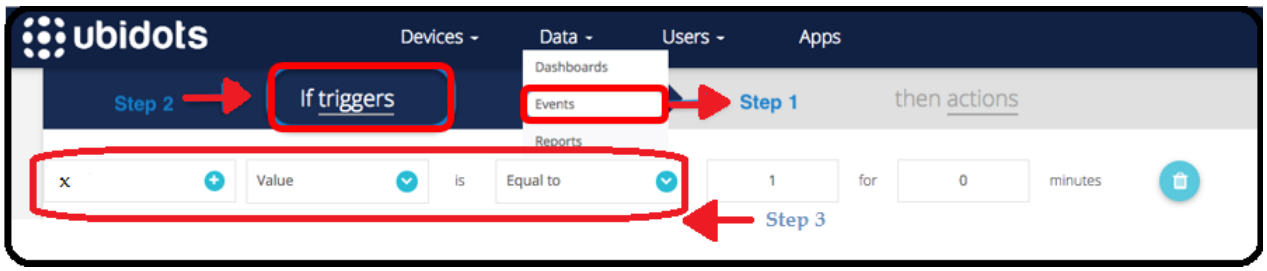



**Figura 7.60: Primeros pasos para programar Downlink en UBIDOTS**

Fuente: [84] y Autores

Luego en la sección **then actions** se selecciona **TTN Downlink**, para el siguiente paso se deben tener a la mano los siguientes datos de la aplicación que está integrada en TTN: **Application_ID**, **Handler** que indica la región de trabajo, la clave de acceso a la aplicación o **ACCESS KEY**, **Device ID** y el puerto para envío de mensajes Downlink que por lo general es el 1. Tanto en la sección **ACTIVE TRIGGER** y **BACK TO NORMAL** se configura con **Payload raw** y se escribe un 1 en este tipo de lenguaje, lo importante es enviar un dato no importa cual, pero entre más pequeño sea mejor. Todos estos datos son ingresados como se muestra en la figura 7.61.

Cuando ya se tiene esta configuración y se inicia el envío de datos en TTN cada vez que se programa un dato Downlink se utiliza la sentencia **scheduled** que significa programado, estos mensajes como se ve en la figura 7.62 se colocan en color gris y cuando se recibe otro mensaje de enlace ascen-

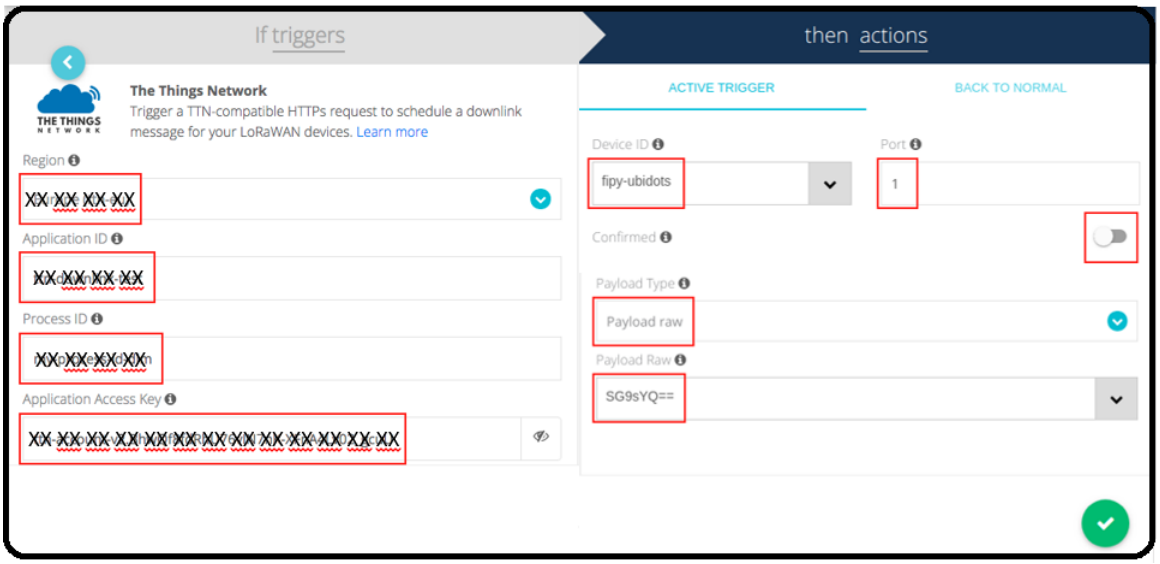



**Figura 7.61: Otras configuraciones para programar Downlink en UBIDOTS**

Fuente: [83] y Autores

dente y se abren las ventanas de recepción del nodo, se envía el mensaje downlink programado como se ve en la misma figura que es de color azul.




**Figura 7.62: Mensajes programados en TTN**

Fuente: Autores

## 7.4 Visualización de los datos

### 7.4.1. Función de decodificación en TTN

El gateway después de recibir los mensajes del nodo final, envía estos datos codificados al servidor de red, en este caso TTN, que debe decodificar el mensaje para obtener los datos medidos por los sensores. Primero se ingresa con el usuario y contraseña creado para hacer uso del servidor TTN donde se encuentra registrado el gateway y cada uno de los nodos, nos dirigimos a la consola y seguidamente a las aplicaciones, elegimos una de las aplicaciones creadas y por último ingresamos a **Payload Formats** y en la ventana **decoder** escribiremos los comandos para decodificar los mensajes como se muestra en la Figura 7.63. En el decodificar se utiliza el lenguaje de programación JavaScript.

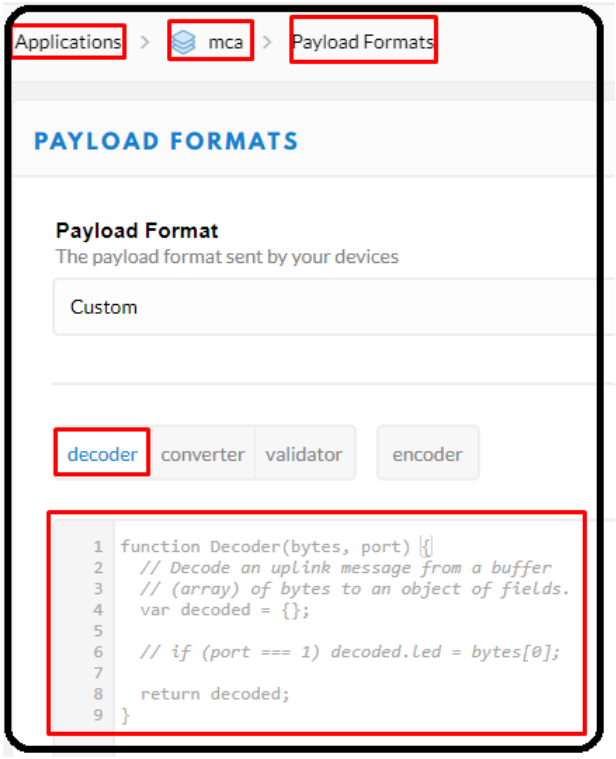



**Figura 7.63: Ingreso a la ventana de decodificación en TTN**

Fuente: Autores

Para la realización de pruebas si el mensaje es una frase se utiliza un decodificador como el de la figura 7.64, se hace referencia a este porque en la plantilla de **ttn-abp** el mensaje que viene por defecto es **hello_world**.




**Figura 7.64: Función de decodificación para mensajes de texto**

Fuente: Autores

Debido al corto ancho de banda de LORAWAN los mensajes deberían ser los más cortos posibles, por lo cual teniendo en cuenta la explicación de TTN acerca del envío de bytes que se encuentra en [85], los valores de los sensores son codificados en la cantidad de bytes que requieran, de acuerdo a su rango. Para la realización de pruebas con bytes, hay que tener en cuenta que al usar el código propuesto deben enviarse por lo menos dos bytes para que la red funcione adecuadamente.

En el código del nodo final, por ejemplo, como los datos no pueden enviarse en formato decimal, son multiplicados por 100 para tener una resolución de 2 decimales antes de ser enviados, después sí se evalúa la cantidad de bytes que necesitan para ser enviados. Si un dato requiere de 2 bytes, se codifica en arduino utilizando las funciones **HighByte** y **LowByte** que hace los corrimientos respectivos, si tiene más de 2 bytes se especifican los corrimientos. La función de decodificación en TTN recibe los bytes e inicia hacer la suma de los corrimientos contrarios para formar el valor, luego este valor se divide en 100 y se obtiene el resultado medido por el sensor. En la figura 7.65 se muestra la codificación y decodificación, de datos de 2 y 3 bytes, respectivamente.

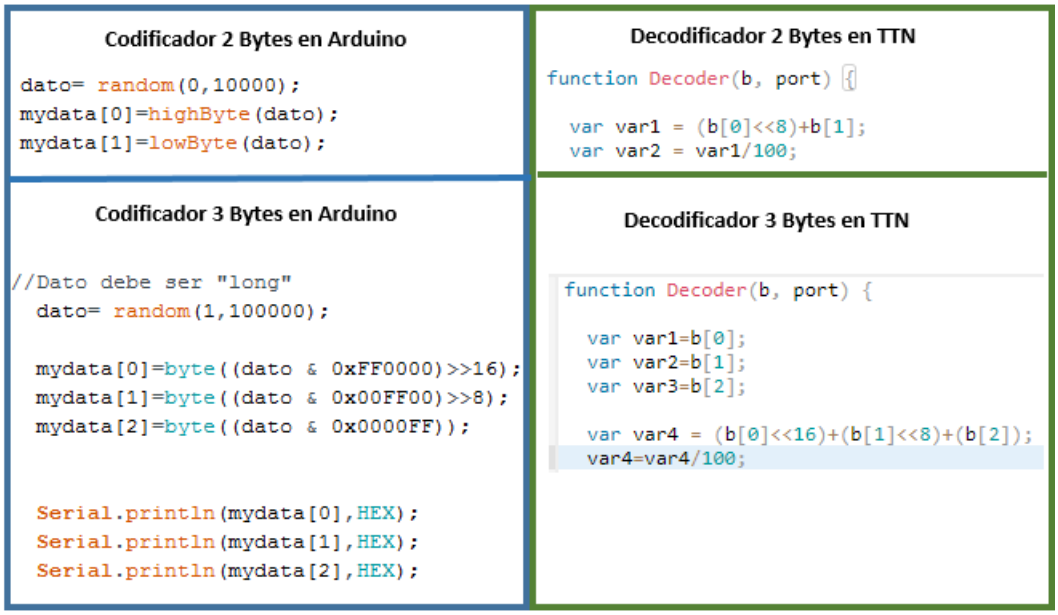



**Figura 7.65: Codificación y decodificación de mensajes en Bytes**

Fuente: Autores

En total se reciben 15 Bytes que contienen en orden la información de presión, temperatura, humedad, concentración de PM10, concentración de PM2.5, concentración de CO y concentración de NO2. Las concentraciones de CO y NO2 fueron enviadas en *ppm* porque requerían menos bytes que en $ug/m^3$ que es la unidad que se encuentra en la norma, por lo cual además de la decodificación, en la función de decodificación se realiza la conversión de unidades usando las ecuaciones (4) y (5) de la sección **7.2.2.2**.

Como una de las variables a enviar es el Indicador de Calidad del Aire, debe calcularse este indicador para cada contaminante y el mayor es el que indica el ICA general según la normativa, por lo cual en la función de decodificación también se incluye este proceso, basándose en la ecuación (1) de la sección **5.1.1.7**, para ello se crean las variables para cada contaminante que deben ser inicializadas como se muestra en la figura 7.66, para que en caso de obtener de no obtener un valor dentro de los rangos la función no se indetermine. Los valores tendrán una asignación de acuerdo al rango en el que se encuentre la concentración del contaminante, basados en la tabla 5.7, a continuación se explica qué

significa cada uno.

- capm1: Valor del ICA
- iapm1: ICA alto según rango según rango del contaminante
- ibpm1: ICA bajo según rango del contaminante
- pcapm1: concentración alta según rango del contaminante
- pcbpm1: concentración baja según rango del contaminante

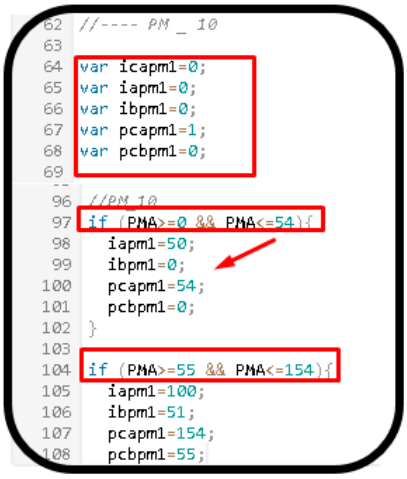



**Figura 7.66: Asignación de valores para variables de la ecuación del ICA**

Fuente: Autores

Después de calcular los ICA de cada contaminante como se especificó anteriormente, se crea una variable que almacene el valor del mayor ICA entre los contaminantes para ello se hace uso de la función **Math.max** que compara los valores y finalmente arroja el mayor valor y lo almacena el la variable llamada **mayorICA** como se puede apreciar en la figura 7.67.

Finalmente al servidor de aplicación serán enviados los valores que se midieron de cada contaminante y el valor de mayor de los tres ICA, como se observa en la figura 7.68, se declararon los campos **Field** con las variables que contienen los valores de los contaminantes. El código de la función de decodificación en su totalidad puede encontrarse en el anexo **11.4**.

Hay que tener en cuenta que después de cualquier cambio en la función de decodificación debe darse en **save payload functions** porque sino se pierde el trabajo realizado, y además con esto se verifica que no hayan errores. En donde dice Payload pueden ingresarse la cantidad de bytes que requiere la función para simular una entrada de datos codificados y confirmar que funcione correctamente.

```
92  //_--- Valor del mayor ICA
93
94  var mayorICA;
258 //--- ICA
259 //PM_10
260 icapm1 = (((iapm1-ibpm1)/(pcapm1-pcbpm1))*(PMA-pcbpm1))+ibpm1;
261 // PM_2.5
262 icapm = (((iapm-ibpm)/(pcapm-pcbpm))*(PMB-pcbpm))+ibpm;
263 // --- C
264 icac = (((iac-ibc)/(pcac-pcbc))*(C-pcbc))+ibc;
265 //___NO2
266 ican = (((ian-ibn)/(pcan-pcbn))*(N-pcbn))+ibn;
268 // _Mayor valor de los tres ICA
269
270 mayorICA= Math.max(icapm1,icapm,icac,ican);
271 mayorICA=mayorICA.toFixed(2);
```

**Figura 7.67: Hallar mayor ICA**

Fuente: Autores

```
273   return {
274     field1: pre,    → Presión
275     field2: temp,   → Temperatura
276     field3: hum,    → Humedad
277     field4: PMA,    → Material Particulado 10
278     field5: PMB,    → Material particulado 2.5
279     field6: C,      → Dioxido de carbono
280     field7: N,      → Monóxido de nitrogeno
281     field8: mayorICA, → Mayor valor de los tres ICA
282
283   };
284 }
```

**Figura 7.68: Retorno de las variables a analizar**

Fuente: Autores

### 7.4.2. Integración de ThingSpeak con TTN

Para crear la integración con ThingSpeak se debe crear una cuenta en MathWorks o si se cuenta con una se inicia sesión en ThingSpeak. La cuenta tendrá una licencia gratuita de un año que cuenta con la oportunidad de crear 4 canales con 8 campos de datos cada uno y 800 mensajes de alerta de correo electrónico.

Para hacer la integración primero se debe crear un canal, para ello en la parte superior se selecciona el menú **Channels**, luego **My Channels** y **New Channel**, se procede en configuraciones a colocar el nombre del canal, descripción del canal, los campos que se habilitarán para graficar los datos con un máximo de 8 campos, en el caso del proyecto se habilitaron todos, como se ve en la Figura 7.69. Además de los campos puede añadirse la localización del nodo y un video como se ve en la figura 7.70.




**Figura 7.69: Creación de un canal en ThinkSpeak**

Fuente: Autores

También es de gran importancia editar la zona horaria para que los datos se visualicen con la hora correcta, para ello ingresamos a **Myprofile**, luego en **Time Zone** se selecciona **Edit** como se ve en la Figura 7.71, en este caso se eligió Bogotá.

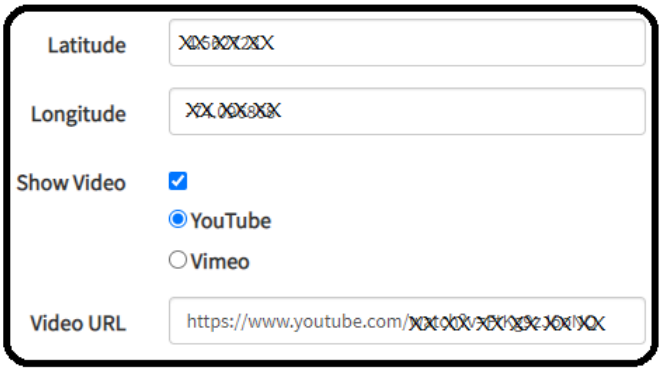



**Figura 7.70: Ubicación del nodo y añadir video en el canal**

Fuente: Autores

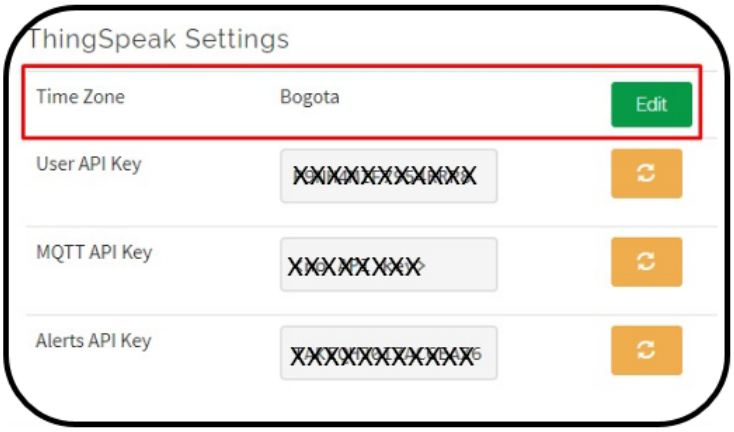



**Figura 7.71: Edición de la zona horaria**

Fuente: Autores

Para hacer la integración con TTN del canal se deben tener a la mano los siguientes datos:

- **Channel ID**, que aparece en la parte superior al entrar en cualquier menú del canal.
- **Write API Key**, que se puede encontrar en la pestaña **API Keys** del canal.

Ahora se procede a crear la integración en TTN para ello se inicia sesión en TTN, luego se ingresa a la consola, a la aplicación registrada, luego a **integraciones**, **agregar una integración** y se busca ThingSpeak como se ve en la Figura 7.72. Luego de seleccionar ThingSpeak se nos abrirá una ventana como se ve en la Figura 7.73 en la cual se ingresan un nombre cualquiera para la integración y los datos del canal anteriormente especificados, como se ve en la Figura 7.73.

De esta manera todos los datos decodificados en TTN serán enviados a ThingSpeak, en la siguiente sección se explica el tipo de gráficos que muestran estos datos.

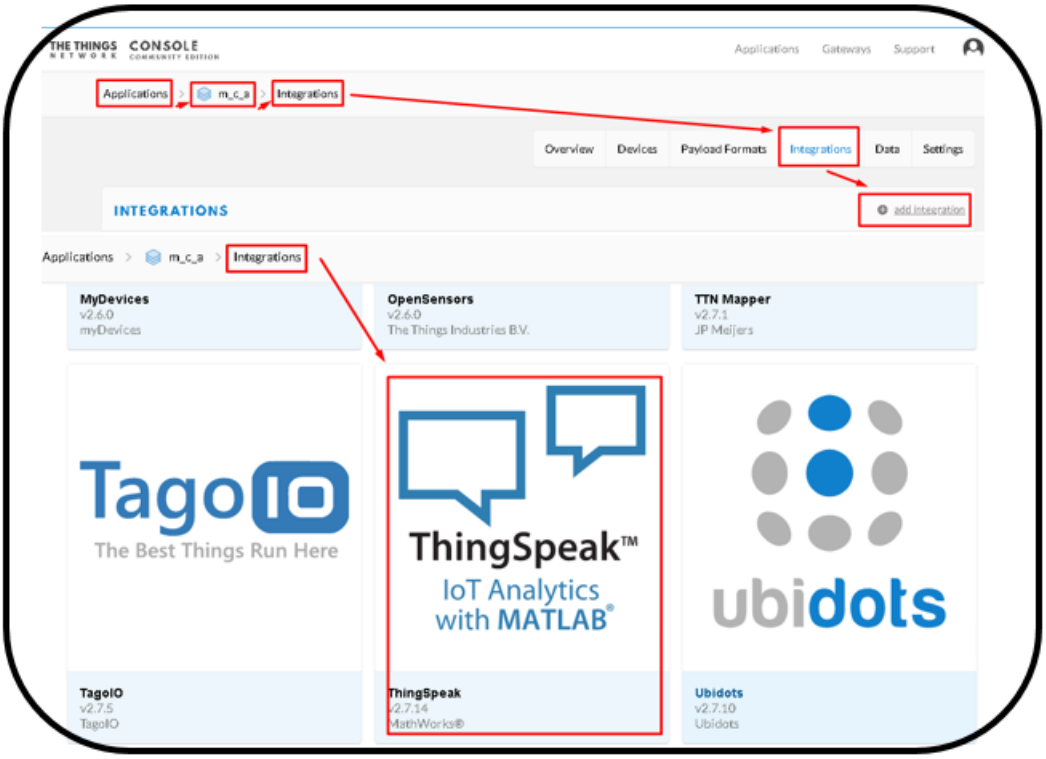



**Figura 7.72: Seleccionar el tipo de integración en TTN**

Fuente: Autores

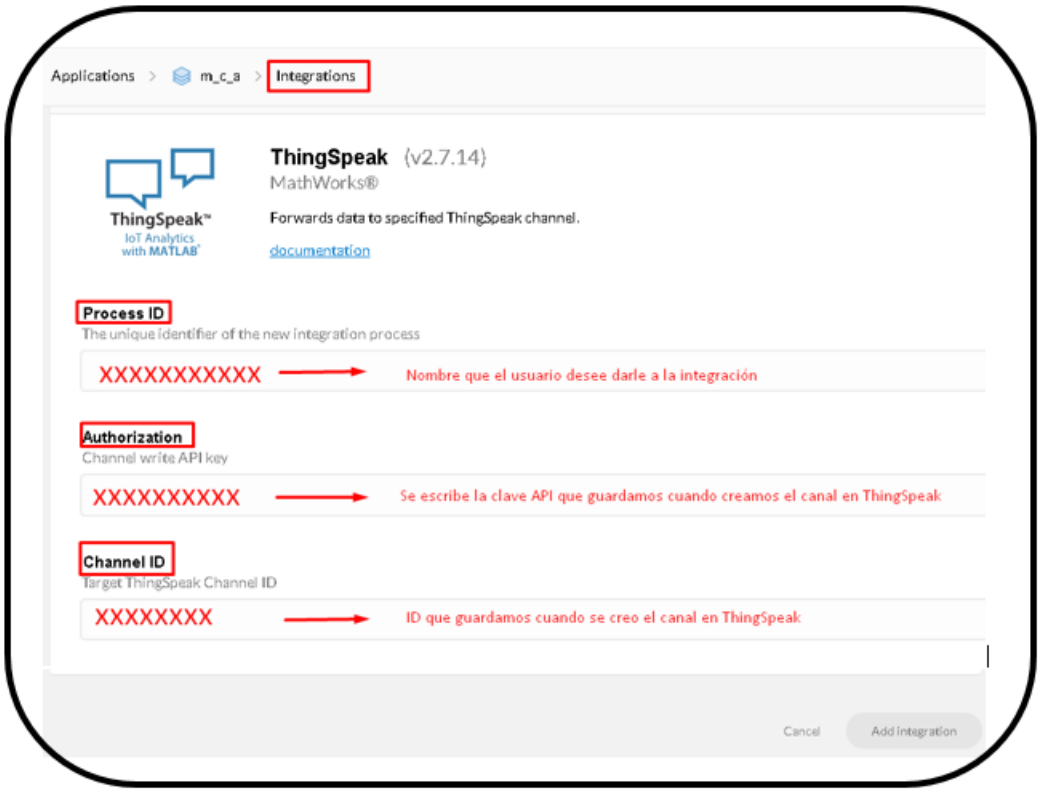



**Figura 7.73: Creación de una integración en TTN**

Fuente: Autores

### 7.4.3. Gráficas en ThingSpeak

ThingSpeak tiene dos tipos de gráficas, las primeras son gráficas dinámicas que crea automáticamente para cada campo habilitado, en ellas llegan los datos en tiempo real, donde el eje **x** corresponde a la hora que el dato fue enviado y el eje **y** el valor del dato recibido, la gráfica varía sus ejes con respecto a los datos recibidos, además se puede configurar si los datos quieren verse como puntos, barras o columnas, la cantidad de datos de cada gráfica, el color, nombre de los ejes y títulos. En este tipo de gráficas es donde se pueden visualizar los datos decodificados enviados por TTN.

El segundo tipo de gráficas las realiza el usuario de acuerdo a su necesidad y utilizando código en Matlab con la herramienta MATLAB Visualization, la desventaja de este tipo de gráficas es que no se actualiza en tiempo real, para ver su última actualización debe recargarse la página.

En el caso del proyecto los datos que reciben los campos corresponden a las concentraciones de los contaminantes, variables ambientales y el ICA más reciente, estos datos son mostrados con el primer tipo de gráficas. Sin embargo, para mayor análisis y entendimiento de la información se decidió realizar una gráfica con el ICA de cada contaminante para ello basados en la ecuación del ICA y con ayuda de la tabla 5.7, se realizó un gráfico de barras que dependiendo el valor del ICA varia su color, el código en Matlab del ICA calculado para el contaminante CO se encuentra en el anexo 11.5, para los demás contaminantes varían los rangos de las concentraciones.

Los dos tipos de gráficas se pueden apreciar en la en la figura 7.74.

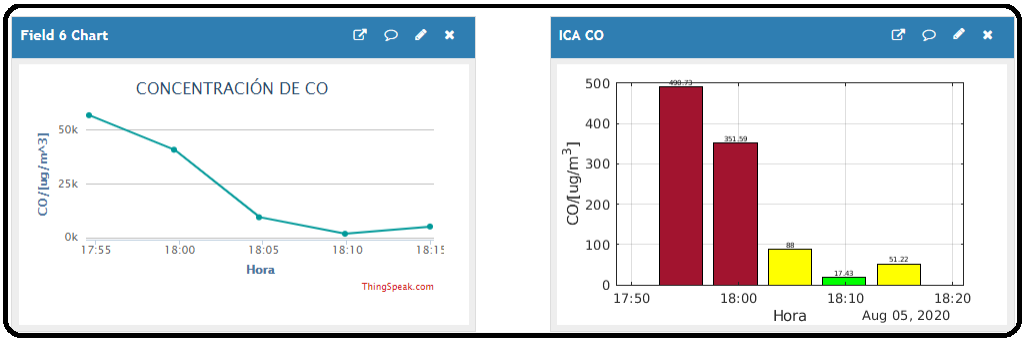



**Figura 7.74: Gráficas en ThingSpeak**

Fuente: Autores

Además de las gráficas mencionadas con la herramienta MATLAB Visualization se realizó una gráfica que muestra los rangos del ICA con sus respectivos colores, para que el usuario pueda interpretar mejor los datos. Esta gráfica se muestra en la figura 7.75 fue realizada haciendo uso de la herramienta **patch** de Matlab.

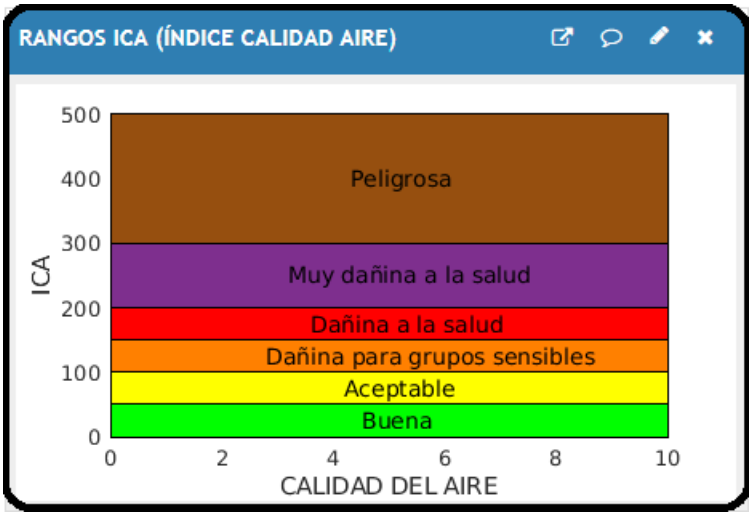



**Figura 7.75: Rangos ICA en ThingSpeak**

Fuente: Autores

### 7.4.4. Alarmas visuales, alarmas de correo electrónico y otros

Además de las gráficas ya nombradas para crear una ambiente visual más dinámico se hizo uso de otra tipo de visualización en ThingSpeak como lo es el Widget llamado Gauge, este muestra el ICA más reciente como se ve en la figura 7.76, muestra hasta 250 y no hasta 500, para que sean mayormente apreciados los rangos de verde, amarillo y naranja donde se encuentran los valores más comunes del índice de calidad del aire.

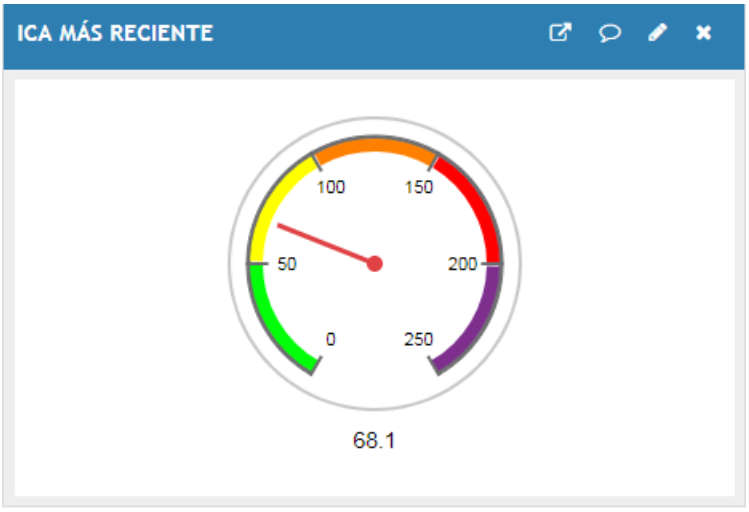



**Figura 7.76: Gauge ICA más reciente**

Fuente: Autores

Además de las alarmas visuales, con ayuda de otra herramienta de ThingSpeak llamada **MATLAB Analysis** y con base en [86], se hizo un código que lee el valor más reciente del ICA y si es mayor de 100 se envía al correo una alerta naranja y si es mayor a 150 alerta roja. Con otra herramienta

denominada **React** se programa la frecuencia del evento, en este caso, cada vez que llegue un dato, y se puede configurar si la reacción se da sólo la primera vez que se cumple la condición o cada vez que se cumpla la condición. Se limita un poco esto porque sólo puede ponerse un evento que suceda cada vez que se cumpla la condición. El código desarrollado se muestra en el anexo **11.6**. El contenido del mensaje de las alertas se muestra en la figura 7.77.

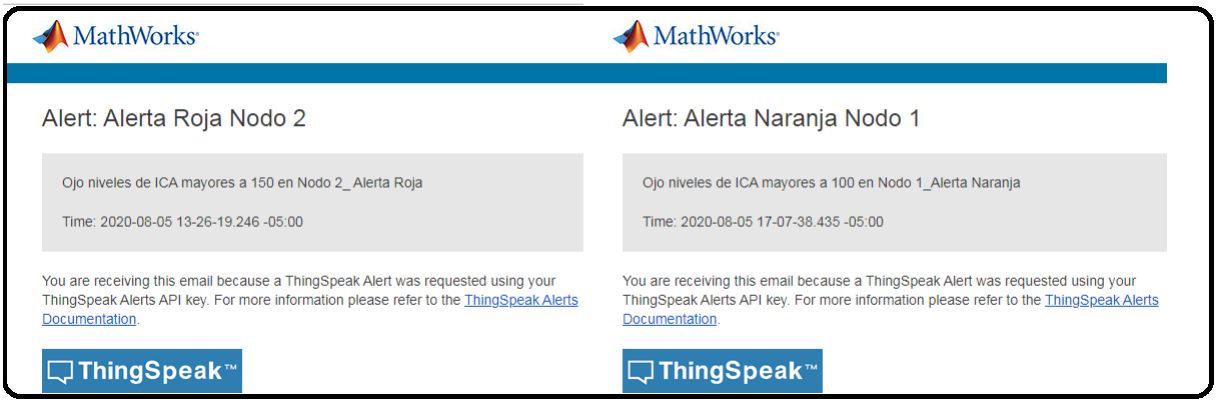



**Figura 7.77: Alertas correo electrónico THINGSPEAK**

Fuente: Autores

Como complementos en la página también se añadió la ubicación del nodo que puede ser configurada manualmente en cada canal, un video del Ministerio del Medio Ambiente donde explica lo que se debe saber sobre la calidad del aire de una forma dinámica, y también se colocó el escudo de la Universidad Distrital que simboliza el sentido de pertenencia a la institución. Todo esto se ve en la figura 7.78.




**Figura 7.78: Complementos expuestos en el canal**

Fuente: Autores

Para cada nodo final se creó un canal público, para ingresar de click en:

- **NODO 1:** `https://thingspeak.com/channels/1067984`
- **NODO 2:** `https://thingspeak.com/channels/1092918`

## 7.5 Prueba con diferentes contaminantes

En esta sección se mostrarán las pruebas realizadas con diferentes contaminantes. Es importante aclarar que las pruebas no se pudieron realizar cerca de la Universidad Distrital como se encontraba

planteado en el objetivo general, debido a las medidas restrictivas tomadas por el gobierno frente al Covid-19, razón por la cual se realizaron cerca al barrio 20 de Julio, que fue donde se ubió el gateway. En primer lugar se hizo la prueba usando un cigarrillo, las imágenes que evidencian la prueba se encuentran en las figuras 7.79 y 7.80, se muestran los resultados de todos los sensores porque en todos hubo un cambio significativo.

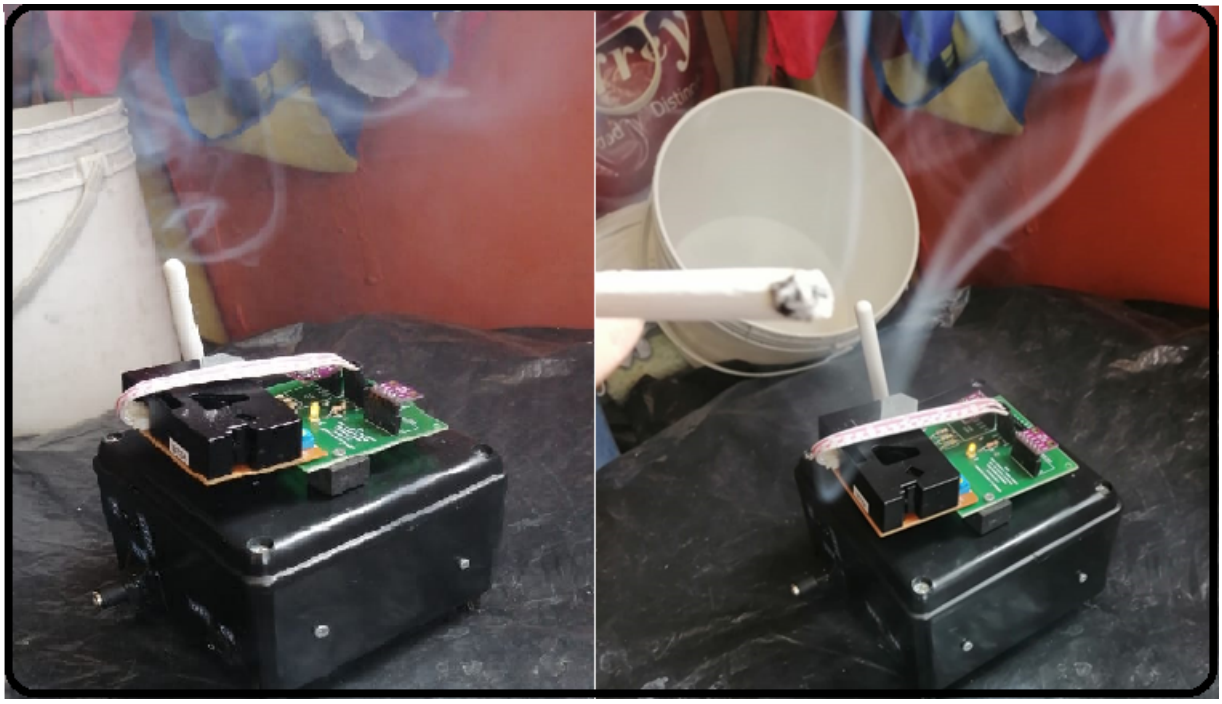


**Figura 7.79: Nodo expuesto a humo de cigarrillo**

Fuente: Autores

En segundo lugar se hizo la prueba usando un trozo de algodón con alcohol,, ya que este líquido al entrar en contacto con el oxígeno desprende CO, las imágenes que evidencian la prueba se encuentran en las figuras 7.81 y 7.82, se muestra sólo el resultado del sensor de CO porque en el resto de sensores no hubo un cambio significativo.

Para realizar la siguiente prueba se quemó un trozo de papel que desprende humo y cenizas, las imágenes que evidencian la prueba se encuentran en las figuras 7.84 y 7.83, se muestran los resultados de todos los sensores porque en todos hubo un cambio significativo.

La última prueba realizada fue colocando el prototipo al lado del exosto de una carro recién encendido, las imágenes que evidencian la prueba se encuentran en las figuras 7.85 y 7.86, se muestran los resultados de todos los sensores porque en todos hubo un cambio significativo.

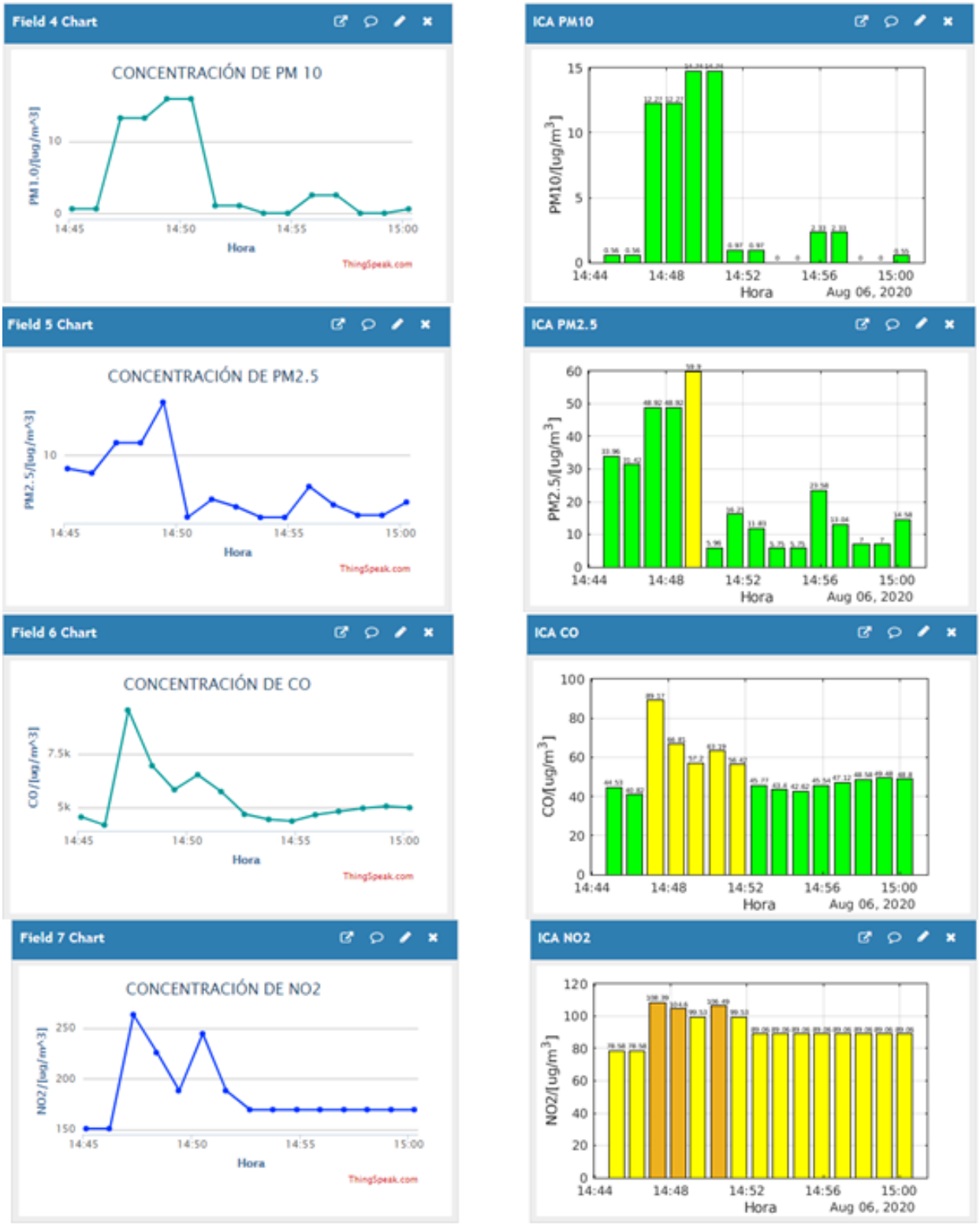



**Figura 7.80: Resultados sensores expuestos a humo de cigarrillo**

Fuente: Autores



**Figura 7.81: Nodo expuesto a reacción entre alcohol y oxígeno**

Fuente: Autores

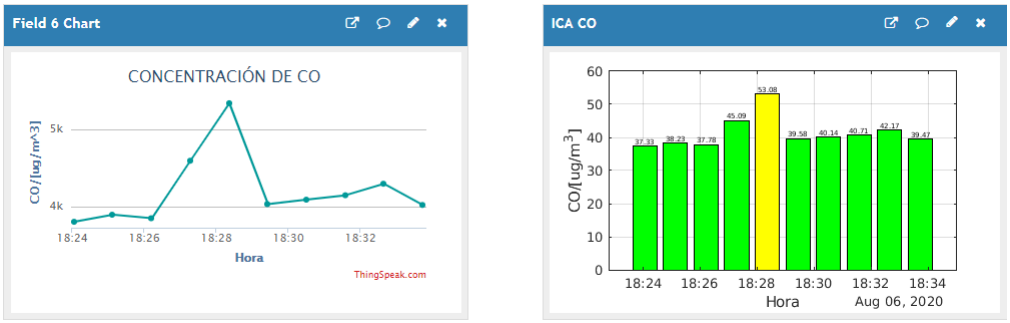



**Figura 7.82: Resultados sensores expuestos a reacción entre alcohol y oxígeno**

Fuente: Autores

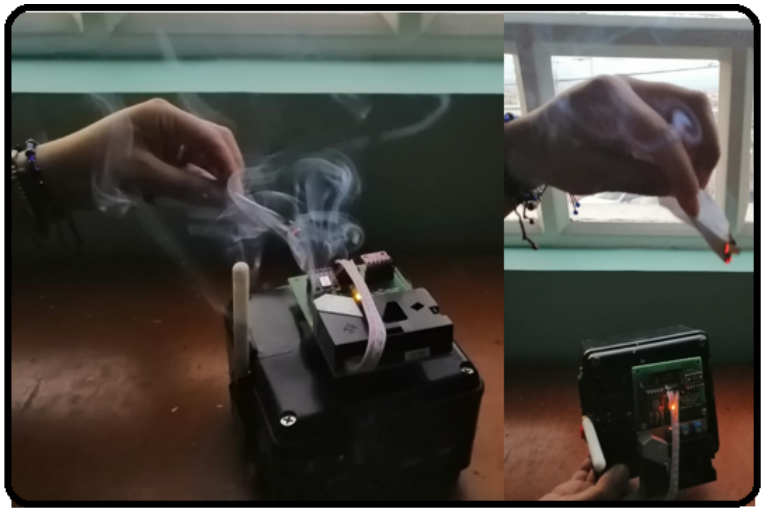


**Figura 7.83: Nodo expuesto a humo y partículas de papel quemado**

Fuente: Autores

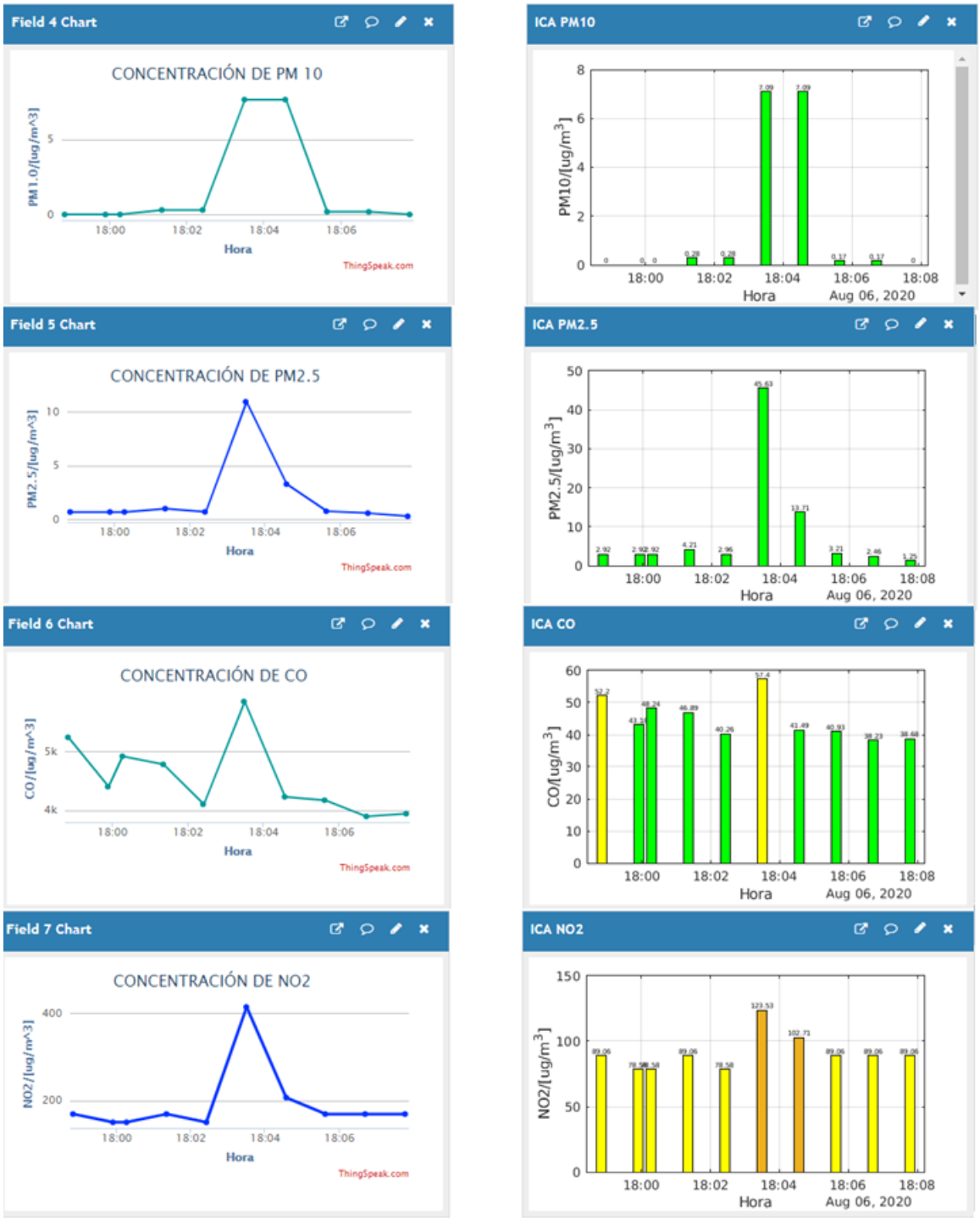



**Figura 7.84: Resultados sensores expuestos a humo y partículas de papel quemado**

Fuente: Autores



**Figura 7.85: Nodo expuesto al humo liberado por el exosto de un carro**

Fuente: Autores

## 7.6 Prueba de cobertura de la red

Para realizar las pruebas de cobertura se procedió a conectar el nodo a las baterías para poder movilizarse sin inconvenientes. Las pruebas fueron realizadas en los alrededores de la casa donde se encuentra el gateway debido a las limitaciones de movilidad actuales por el Covid-19, cerca al barrio 20 de Julio. Se realizó la prueba en 16 puntos diferentes, donde variaba la distancia y la cantidad de obstáculos. Con esta prueba se trató de estimar qué tanto afecta la distancia y los obstáculos a la hora de recibir la señal, para ello se tomaron los datos de la potencia de la señal y se evaluó en estos puntos si la señal era recibida por el gateway.

Para comparar mejor las distancias se realizó el mapa de la figura 7.87 y para comparar mejor los obstáculos se realizó un mapa saltelital como se muestra en la figura 7.88. En cada mapa se incluyó una tabla con las distancias a las que se tomaron datos y la RSSI o potencia de la señal, puede verse en los puntos H e I no se registra el factor RSSI porque no llegaron los datos al gateway, además hay que teer en cuenta que la realción señal a ruido o SNR para todos los puntos fue de 7.8 debido a que no se cambio el factor de difusión.

También cabe resaltar que estas pruebas fueron hechas a diferentes alturas debido a las variaciones en el terreno por lo que las interferencias por construcciones en algunos puntos no afectaron tanto como en otros don habían mas construcciones entre el el lugar donde se encontraba el gateway y el lugar donde se estaban tomando los datos.

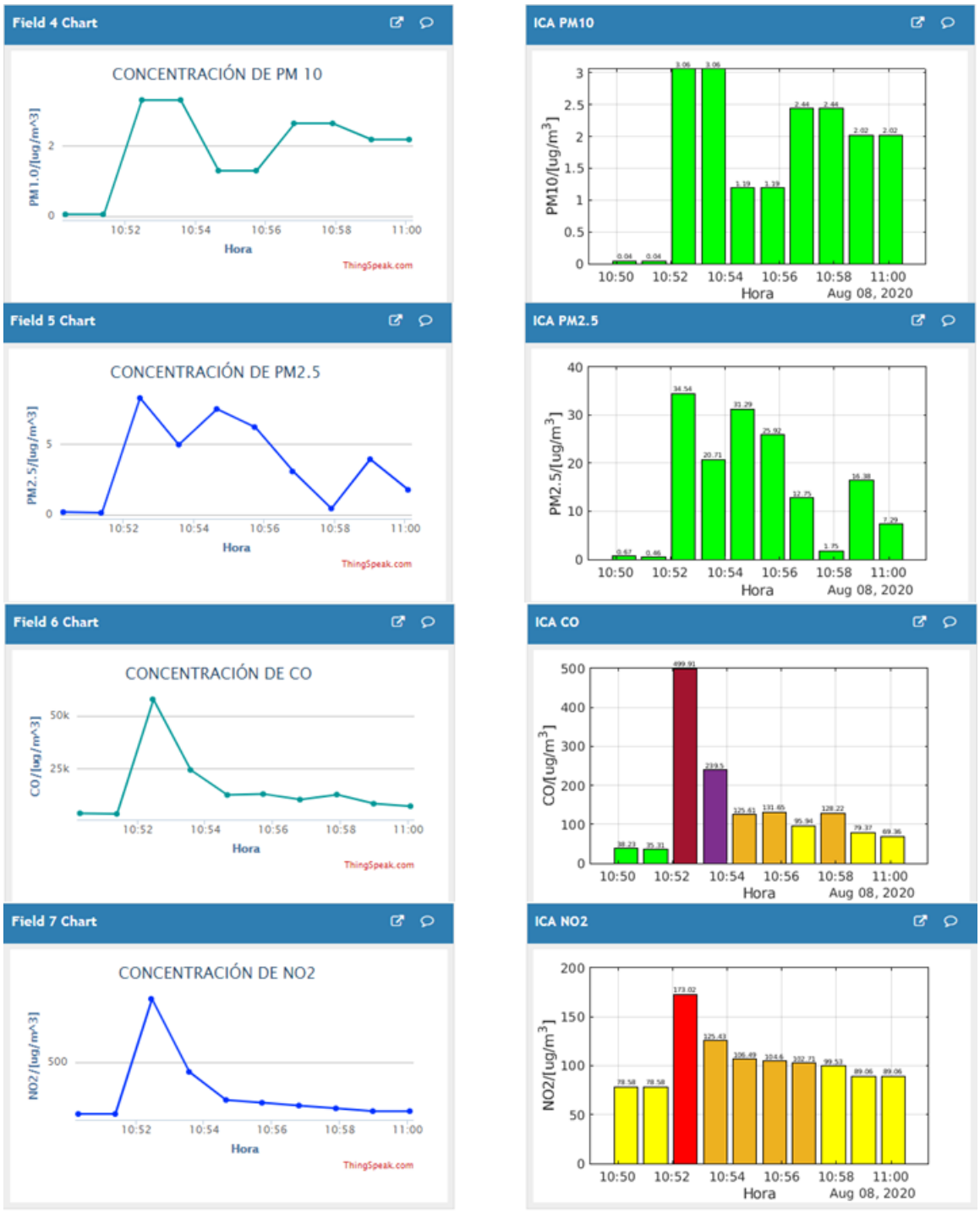
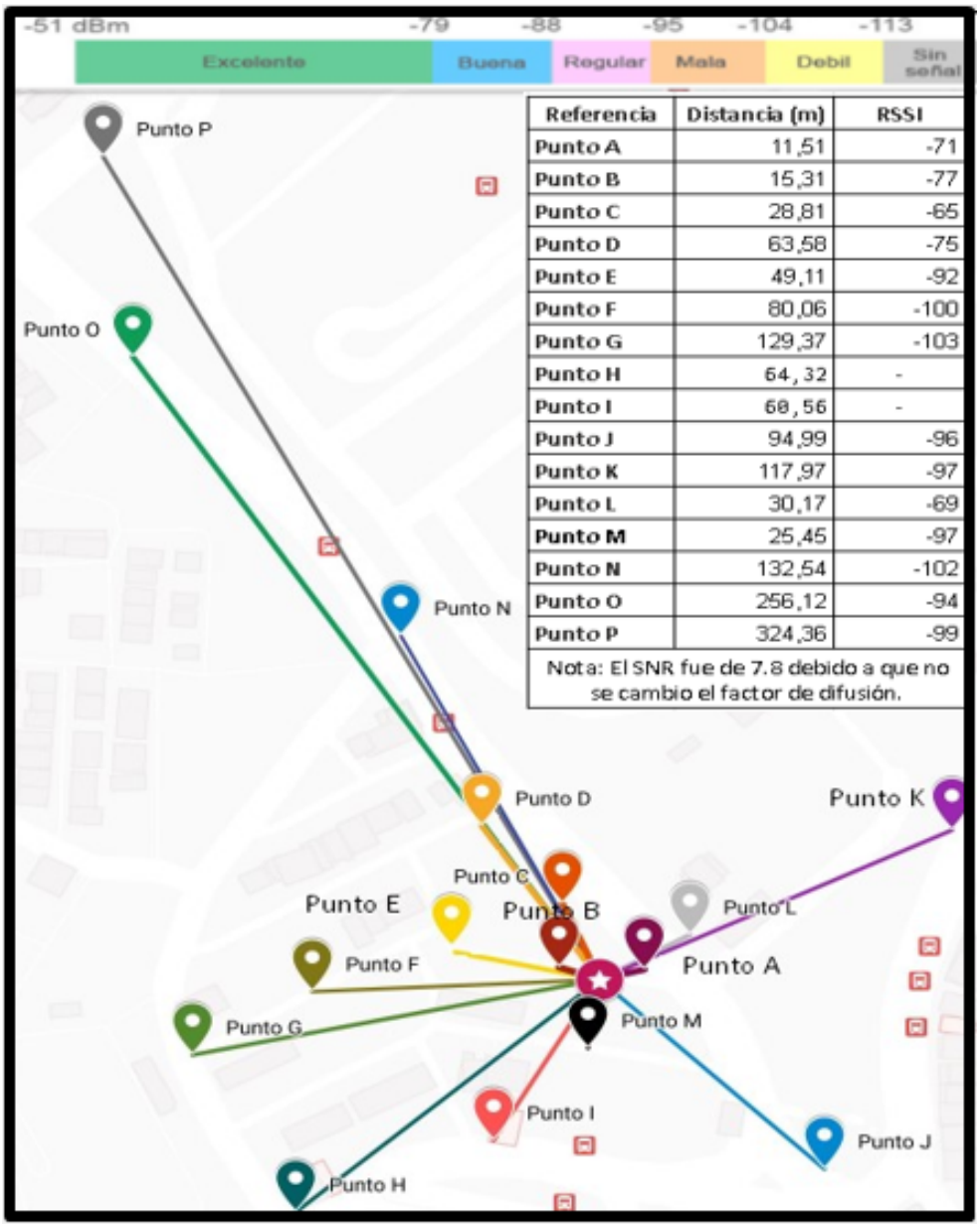



**Figura 7.86: Resultados sensores expuestos al humo liberado por el exosto de un carro**

Fuente: Autores

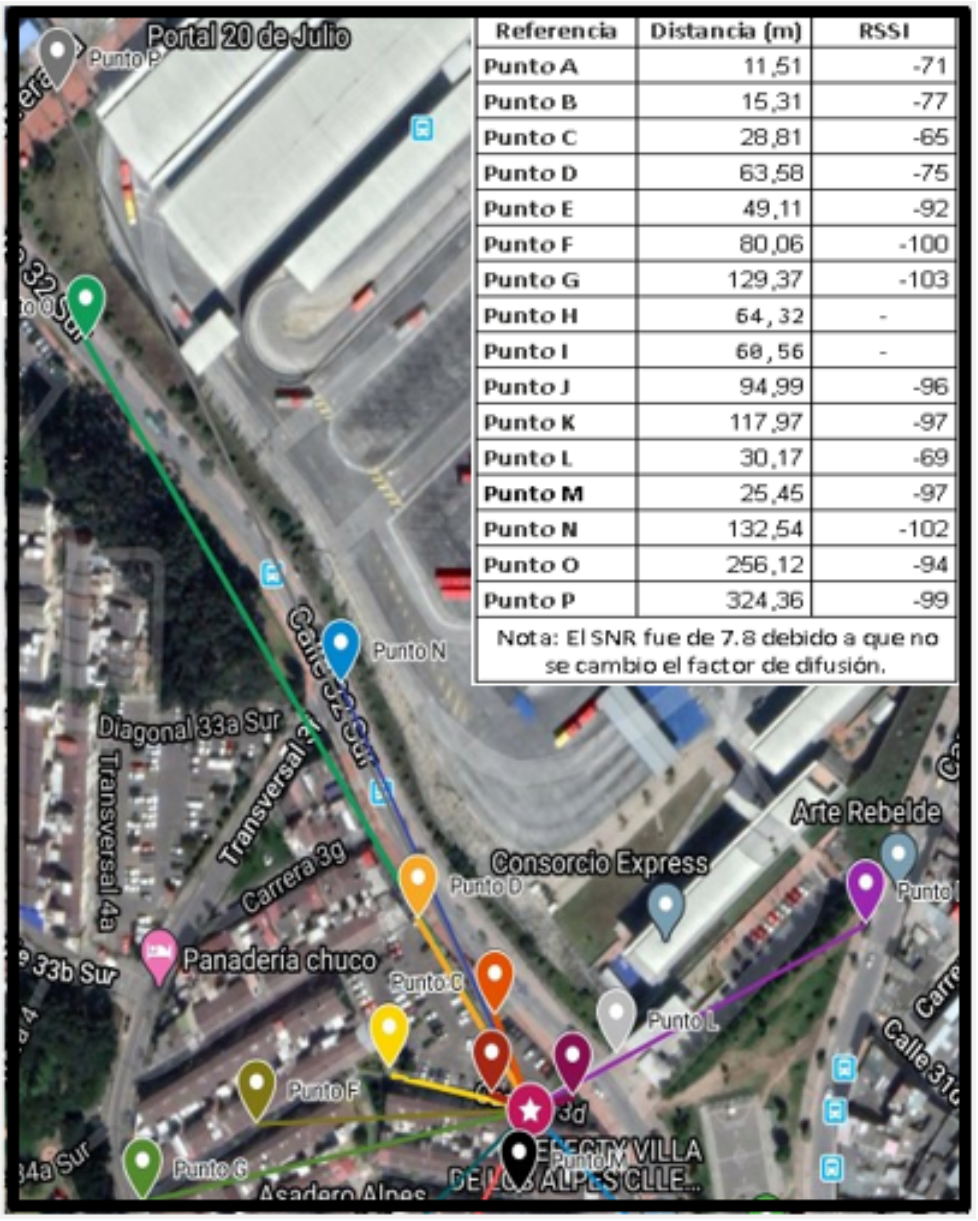



| Referencia | Distancia (m) | RSSI |
|---|---|---|
| Punto A | 11,51 | -71 |
| Punto B | 15,31 | -77 |
| Punto C | 28,81 | -65 |
| Punto D | 63,58 | -75 |
| Punto E | 49,11 | -92 |
| Punto F | 80,06 | -100 |
| Punto G | 129,37 | -103 |
| Punto H | 64,32 | - |
| Punto I | 68,56 | - |
| Punto J | 94,99 | -96 |
| Punto K | 117,97 | -97 |
| Punto L | 30,17 | -69 |
| Punto M | 25,45 | -97 |
| Punto N | 132,54 | -102 |
| Punto O | 256,12 | -94 |
| Punto P | 324,36 | -99 |

Nota: El SNR fue de 7.8 debido a que no se cambio el factor de difusión.

**Figura 7.87: Mapa donde se resume las pruebas de cobertura realizadas**

Fuente: Autores




| Referencia | Distancia (m) | RSSI |
|---|---|---|
| Punto A | 11,51 | -71 |
| Punto B | 15,31 | -77 |
| Punto C | 28,81 | -65 |
| Punto D | 63,58 | -75 |
| Punto E | 49,11 | -92 |
| Punto F | 80,06 | -100 |
| Punto G | 129,37 | -103 |
| Punto H | 64,32 | - |
| Punto I | 68,56 | - |
| Punto J | 94,99 | -96 |
| Punto K | 117,97 | -97 |
| Punto L | 30,17 | -69 |
| Punto M | 25,45 | -97 |
| Punto N | 132,54 | -102 |
| Punto O | 256,12 | -94 |
| Punto P | 324,36 | -99 |

**Figura 7.88: Mapa satelital donde se resume las pruebas de cobertura realizadas**

Fuente: Autores

## 7.7 Conteo de tramas

Parar realizar un conteo de tramas se puede hacer uso de TTN donde se inicializan los contadores en cero y este registra cuántos fueron los últimos mensajes enviados, si se detectó pérdida de mensajes con ayuda de las gráficas dinámicas de ThingSpeak se pueden analizar los datos y ver el momento en que hubo esta perdida, también es posible analizar lo datos descargándolos directamente desde ThingSpeak en un archivo CSV que se abre con Excel.

Teniendo en cuenta todas estas posibilidades se realizaron 3 pruebas para analizar la cantidad de datos que se pierden durante la transmisión. La primera y tercera prueba se realizó en el mismo lugar que corresponde al piso más alto de la casa donde está ubicado el gateway a 6 *m* más alto del gateway y a un distancia de 7.2 *m* aproximadamente, fue un ambiente con varios obstáculos.

En la segunda prueba el nodo se ubicó en una casa vecina que se encuentra a 62.5 *m* de distancia, pero la altura fue aproximadamente igual a la del gateway, las casas están separadas por un parqueadero, por lo cual el gateway tenía línea de vista con el nodo.

Teniendo en cuenta que el gateway se encuentra en el segundo piso de la casa, se hizo una cuarta prueba con el nodo en el tercer piso en la parte más alta con aproximadamente 5 *m* de distancia y obstáculos, esta prueba se hizo hasta que las baterías se descargaran, que corresponde a un lapso de 14 horas.

Los resultados de las pruebas se muestran en la tabla 7.2.

**Tabla 7.2: Pruebas conteo de tramas**

| Prueba | Distancia al gateway | Frec. datos | Número de horas | Datos esperados | Datos obtenidos | Efectividad |
|---|---|---|---|---|---|---|
| 1 | 7.2 *m* con obstáculos | 1 dato cada min | 4 horas | 240 | 223 | 93 % |
| 2 | 63.5 *m* línea de vista | 1 dato cada min | 6 horas | 360 | 334 | 93 % |
| 3 | 7.2 *m* con obstáculos | 1 dato cada 5 min | 4 horas | 48 | 48 | 100 % |
| 4 | 5 *m* con obstáculos | 1 dato cada 5 min | 14 horas | 168 | 168 | 100 % |

Fuente: Autores

## 7.8 Costos de la Red

En la tabla 7.3 se muestran los gastos de un nodo teniendo en cuenta los ítems generales para determinar el costo final del mismo, es de aclara que en el ítem de armazón se incluyen los precios de la caja de almacenamiento, las baquelas y demás elementos que permiten armar el nodo final. En total se realizaron dos nodos.

Los costos de este nodo permiten realizar una comparación con elementos de producción que se encuentran en el mercado los cuales tienen un costo que puede variar entre un millón de pesos y quince millones de pesos si el nodo no pertenece a una red, y un costo entre ochocientos treintay nueve mil y veinticuatro millones si el nodo está conectado a una red, como se muestra en la figura 7.89, pero cabe aclarar que estos aparatos no miden todos los contaminantes que se miden con el nodo que se realizó en el desarrollo del proyecto, estos en promedio llegan a medir entre uno o dos de los indicadores del aire y algunos factores ambientales.

Para completar los costos de la red debemos nombrar el gateway LG02 de Dragino que permitió la comunicación con el nodo final y el servidor de red que tiene un costo de trecientos once mil ($311.000), por lo que haciendo el ponderado de los dos nodos y el gateway que conforman la red, tiene un costo final de novecientos sesenta mil seiscientos pesos ($960.600).

Como se usó software libre tanto para el código del microcontrolador y servidor de red, y una versión gratuita con un año de uso para el servidor de aplicación, en este caso estos costos no se tienen en cuenta en el desarrollo final, sin embargo, si se quieren más de cuatro nodos en la red debe considerarse pagar una licencia en el servidor de aplicación para uso comercial, gubernamental y generadores de ingresos, tiene un costo de $ 2’571.960.

**Tabla 7.3: Costo prototipo nodo final**

| **Ítem** | **Precio** |
|---|---|
| Microcontrolador + Programador | $ 25.000 |
| Sensores | $ 102.300 |
| Módulo LORABEE | $ 69.500 |
| Sistema de alimentación | $ 78.000 |
| Armazón | $ 50.000 |
| **Total** | $ 324.800 |

Fuente: Autores




**Figura 7.89: Precios de nodos y redes en el mercado**

Fuente: Autores

# 8. Análisis de Resultados

Debido al desarrollo de la industria y campo automotor se han identificado ciertos contaminantes que afectan la salud de las personas, por lo cual su estudio en los últimos años ha aumentado, a tal grado de implementar estaciones de monitoreo en las ciudades para controlar el flujo de carros, por ejemplo y otras medidas que permitan reducir este tipo de contaminación. Sin embargo, estas estaciones son fijas y costosas, lo que limita la implementación de más estaciones que permitan dar mediciones más exactas y de esta manera controles más específicos. A partir de lo anterior se ha visto la necesidad de buscar alternativas más económicas que permitan monitorear la calidad del aire y que tengan mejor cobertura, por lo cual nace este proyecto que busca iniciar el camino de la construcción de una red de calidad de aire robusta, dando los primero pasos en la construcción de dos prototipos que con un gateway, conformarán una red utilizando el protocolo de comunicación LORAWAN.

Con ayuda del Doctor Andres Torres director del grupo de investigación Ciencia e Ingeniería del Agua y el Ambiente de la Pontificia Universidad Javeriana, que cuenta con más experiencia en este tipo de proyectos, en primera instancia se decidió medir cuatro de los indicadores de calidad del aire expuestos en la resolución 2254 de 2017 del Ministerio de Ambiente y Desarrollo, y 3 factores ambientales.

En el desarrollo del prototipo se logró la integración correcta de los sensores, el sistema de comunicación a través de un eje central como lo es un microcontrolador y el sistema de alimentación que permite al nodo ser portátil o en su defecto, dependiente de la toma de corriente. También quiso integrarse un sistema de almacenamiento externo que permitiera guardar los datos que no pudieran ser enviados a la red por desconexión de la misma, sin embargo, a lo largo del desarrollo se encontraron dos limitantes que no permitieron este objetivo. En primer lugar, al utilizar herramientas públicas no pagas, como en este caso del servidor de red TTN, que limita los mensajes del gateway al nodo a 10 diarios, no se hace viable implementar mensajes de confirmación necesarios para la realización del algoritmo de almacenamiento externo. En segundo lugar al no utilizar el IDE propio del microcontrolador, por la instalación del núcleo del dispositivo en el IDE usado y el cargador de arranque, al tratar de implementar el código completo se encontró que el espacio en la memoria FLASH era insuficiente.

A pesar de esto en el desarrollo del proyecto se dejó planteado un algoritmo que podrá ser implementado si se hace el cambio del servidor de red que tenga menos limitaciones y se utiliza el IDE el STM32 para evitar gastos innecesarios en la memoria FLASH.

El sistema de alimentación implementado permitió la realización de todas la pruebas necesarias de forma más sencilla al ser portátil, además con respecto a las baterías usadas, a pesar de no ser nuevas dieron buenos resultados teniendo un tiempo de duración de 14 horas aproximadamente y un tiempo

de recarga de 5 horas.

Con lo anterior, respecto a la arquitectura del nodo planteada inicialmente en el proyecto, todo fue acoplado menos el sistema de almacenamiento externo. Cabe aclarar que en uno de los prototipos no fue posible colocar uno de los sensores, ya que durante el desarrollo de las pruebas uno de los sensores de gases se dañó y por la situación actual de la pandemia no fue posible adquirir otro, pero para la implementación del nodo se emuló el sensor por medio de una función random que varía de acuerdo al rango del sensor.

Después de un estudio juicioso de los parámetros de una red LORAWAN y debido a las limitaciones con las que cuenta el gateway, sólo es posible configurar el envío y recepción de datos por medio de una frecuencia única para mensajes ascendentes y descendentes, respectivamente, lo que no pasa en una red LORAWAN normal donde hay saltos de frecuencia para evitar que se pierdan datos, sin embargo, como el gateway cuenta con dos canales de recepción y sólo se cuenta con dos nodos, fue posible implementar de forma correcta la red teniendo una conexión estable. Hay que tener en cuenta que como sólo puede configurarse una frecuencia de igual manera ocurre con los demás parámetros como el factor de difusión, ancho de banda y tasa de codificación.

Debido a que LORAWAN tiene un ancho de banda pequeño la cantidad de datos que pueden enviarse es bastante limitada, por lo cual lo más aconsejable es enviar los datos codificados por medio de bytes. Por esta razón y debido a las configuraciones elegidas en los parámetros de red, el tiempo de cada mensaje es muy corto y al escoger una tasa de 1 dato cada 5 minutos por el tipo de variables medidas, con respecto a las limitaciones del servidor de red de 30 segundos de mensajes ascendente en un día, la red no está ni cerca de este límite, lo que no resulta ser una preocupación a la hora de implementar la red por mucho tiempo.

La comunicación del gateway, con el servidor de red TTN y el servidor de aplicación ThingSpeak fue muy sencilla de configurar, lo que facilitó el análisis y visualización correcta de los datos, ya que de no haber tenido este tipo de herramientas se hubiera tenido que crear desde ceros una servidor que permitiera analizar los datos recibidos por el gateway.

ThingSpeak a pesar de las limitaciones de la licencia gratuita, permitió el uso de varias gráficas y un ambiente visual agradable para que el usuario final de la red pueda interpretar de forma asertiva los datos obtenidos.

Después de ensamblada la red se procede a hacer pruebas con diferentes contaminantes que permitan comprobar el correcto funcionamiento de la red y la visualización de los datos. En primer lugar, en la figura 7.80 se muestran los resultados obtenidos al exponer el nodo al humo del cigarrillo, como puede verse este humo aumenta la concentración de todos los contaminantes medidos, lo que convierte

en una de las fuentes de contaminación que más efectos perjudiciales tiene sobren la salud. En especial se aumenta la concentración de PM2.5 y CO, por lo cual se recomienda que las personas con angina que es una enfermedad que afecta las arterias coronarias, asma, hipertensión o bronquitis crónica, no tengan un alto contacto con esta fuente de contaminación.

En la figura 7.81 se pueden ver los resultados obtenidos al exponer el nodo al contacto del alcohol con el oxígeno que liberan monóxido de carbono, por lo cual sólo se mostró esta gráfica donde hubo un cambio significativo. Este tipo de contaminante en altas cantidades puede ser mortal.

En la figura 7.83 se muestran los resultados obtenidos al exponer al nodo a la quema de papel donde se liberan contaminantes gaseosos y material particulado; como puede verse esta fuente de contaminación umenta la concentración de todos los contaminantes medidos, por lo cual también es un riesgo para la salud y una muestra de que la quema de materia puede dañar de forma significativa el ambiente. Esta fuente de contaminación en especial libera NO2 y material particulado, por lo cual no se recomienda el contacto frecuente y de grandes cantidades, con las personas que padezcan asma, hipertensión arterial, isquemia en el miocardio y bronquitis crónica.

Por último en la figura 7.86 están los resultados obtenidos al exponer el nodo al humo desprendido del exosto de un carro, donde se puede notar que todos los contaminantes tienen un crecimiento, pero especialmente este humo aumenta la cantidad de CO y NO2 en el ambiente, de ahí viene la importancia de los controles que se hacen a los automotores y la necesidad de que se encuentren en buen estado, debido a la gran cantidad de carros que hay por ejemplo en la ciudad de Bogotá. Este tipo de contaminación en gran medida afecta a las personas con angina y asma.

Durante la prueba de diferentes contaminantes fueron recibidas las alertas cuando el nodo superó un ICA de 100, lo que indica alerta naranja, esto le permitiría al usuario final estar atento para tomar decisiones en momentos precisos y si es desconocida determinar la fuente de contaminación ya que se indica la hora exacta del suceso.

En cuanto a las pruebas realizadas de cobertura, es importante resaltar que no fueron demasiadas por los problemas de movilidad que se presentan actualmente por el Covid- 19, pero fueron suficientes para realizar un análisis de factores que causaron interferencia en la transmisión de datos. En la Figura 7.88 se puede ver que en el punto H, con una distancia de 64 *m* no se recibieron datos en el gateway debido a la cantidad de casas y construcciones que generaron una barrera de interferencia, comparando con el punto O que está a una una distancia de 324 *m*, donde sí hubo recepción de datos, a pesar de estar a una distancia 5 veces mayor, sin embargo, como puede observarse en la misma figura, entre el gateway y ese punto hay menos construcciones que reducen la interferencia en la comunicación.

Con respecto a la tabla de referencias mostrada en la figura 7.87 puede notarse que como era de

esperarse a mayor distancia menor es la RSSI o potencia de la señal y también se reduce por la cantidad de obstáculos entre el gateway y el nodo. Cabe aclarar que no se incluyó un análisis de la relación señal a ruido ya que al no variar el factor de difusión, esta relación siempre obtuvo el mismo valor en todos los puntos.

Al analizar la cantidad de tramas que se pierden en una transmisión, teniendo en cuenta los resultados de la tabla 7.2 se puede decir:

- Al analizar los datos recibidos en la consola de TTN hay un error de generalmente 3 segundos entre cada envío del mensaje, se produce un error acumulativo, por lo cual al tener una tasa de 1 dato por minuto, en vez de enviarse 60 mensajes en una hora como es lo esperado se están enviado aproximadamente 56 mensajes.
- Teniendo en cuenta el ítem anterior, en la prueba 1 se esperarían en realidad 224 mensajes, lo que indica que se perdió 1 en la transmisión, cosa que fue comprobada al analizar los datos recibidos en ThingSpeak.
- Al hacer la prueba con una tasa de 1 mensaje cada 5 minutos, el error no es tan notorio por lo cual no ocurre este fenómeno, lo que pasa es que al comparar el primer mensaje enviado y el último recibido en una hora se van a diferenciar en la cantidad de segundos transcurridos, porque a pesar que la prueba 3 y 4 tienen tiempos de duración bastante distantes se obtuvieron los mismos resultados.
- A pesar de que en la prueba 2 el nodo estaba 8 veces más lejos, el resultado con la prueba 1 fue muy similar, lo que indica que tiene más influencia la tasa de envío de datos que la distancia.

En esta última parte como se muestra en la tabla 7.3 y en la figura 7.89 se pueden comparar los precios entre el nodo final desarrollado con algunos nodos del mercado que miden menos variables que el nodo desarrollado, sin embargo, su costo dependiendo de las variables que mide es mucho más alto. Si el nodo tiene conexión a una plataforma para visualizar los datos su costo se eleva como puede verse en la figura 7.89.

# 9. Conclusiones

- Debido a la gran cantidad de contaminación que se encuentra en las grandes ciudades por las industrias y sector automotor, y con el auge de las nuevas tecnologías IoT, se crea un prototipo portátil de bajo costo que permite la medición algunos de estos contaminantes para calcular el índice de calidad el aire, pudiendo ser visualizado desde cualquier dispositivo que tenga acceso a internet. Estos prototipos tienen un tamaño pequeño y sus costos comparados con las estaciones que actualmente se tienen en Colombia son muy bajos, además de dar la posibilidad de determinar la calidad del aire de manera más puntual ya que se podrán ubicar con mayor facilidad en diversas zonas.

- Es vital tener en cuenta todas las precauciones y recomendaciones que indica el fabricante, porque en el desarrollo del prototipo uno de los sensores que se adquirió sufrió daños y debido a las condiciones actuales y al alto costo reemplazarlo no fue posible. Además debe procurarse al instalar la red hacer los mantenimientos respectivos para evitar fluctuaciones en las mediciones y prolongar la vida útil de los sensores.

- El gateway LG02 tiene varias limitaciones que no permiten el desarrollo de una red LORAWAN convencional con saltos de frecuencias, al sólo poder configurarse en una única frecuencia.

- Debido a las limitaciones del servidor de red en cuanto a los mensajes de enlace descendente y a que el uso del microcontrolador STM32 con el IDE de arduino gasta más memoria FLASH, no fue posible implementar el sistema de almacenamiento externo que permitiría reenviar los mensajes que por desconexión no fueran enviados a la red.

- Con el algoritmo propuesto para la implementación del sistema de almacenamiento externo, se busca reducir el gasto de energía innecesaria, ya que con un sistema normal el mensaje se reenvía hasta que vuelva a haber conexión, mientras que con este algoritmo sólo cuando hay conexión se reenvían los mensajes.

- Actualmente en el mercado se pueden encontrar muchos dispositivos que ayudan a implementar un sistema de alimentación sin muchos tropiezos, como por ejemplo el BMS que protege y prolonga la vida de las baterías y el módulo LM2596 reductor de voltaje que no necesita un disipador ya que funciona por conmutación, además que son de bajo costo.

- Con respecto al sistema de alimentación implementado se obtuvo un buen resultado, ya que a pesar de que las baterías eran usadas su duración fue de 14 horas aproximadamente y el tiempo de carga de 5 horas que no es muy largo.

- Por el poco ancho de banda que tiene LORAWAN se debe buscar la manera de enviar la menor cantidad de datos posibles, para ello se recomienda la codificación de los datos en bytes. Cabe

aclarar que para el correcto funcionamiento de esta red deben enviarse un mínimo de 2 bytes de datos,esto para las pruebas iniciales de envío de datos.

- En la actualidad existen servidores de red libres como TTN y servidores de aplicación con un año de prueba como ThingSpeak, que permiten el desarrollo de diferentes redes para la implementación de proyectos IoT de investigación, académicos o libres.
- TTN es un excelente servidor de red que permite visualizar los datos, visualizar metadatos, decodificar los mensajes y hacer diversas integraciones con servidores de aplicación.
- Con la tasa de datos de 1 mensaje cada 5 minutos, no se sobrepasa ni se está cerca de la limitación del servidor de red, en cuanto a la cantidad de datos diarios que puede enviar el nodo al gateway.
- Thingspeak permite un ambiente de visualización práctico para que el usuario pueda interpretar de la mejor manera datos. Esta herramienta permite una implementación ágil, debido a que no se debe iniciar desde cero el entorno de visualización
- En el servidor de aplicación el usuario final puede visualizar los factores ambientales, contaminantes medidos por los nodos y niveles del ICA, permitiéndole a este tomar las medidas pertinentes para mejorar la calidad el aire y así proteger su salud y la de los demás, también con ayuda de las alarmas de correo electrónico el usuario no necesita estar revisando la página sino cuando haya un evento adverso será notificado.
- Las pruebas con diferentes contaminantes permitieron comprobar el correcto funcionamiento de la red LORAWAN implementada, es decir, la interconexión entre el nodo final, el gateway, el servidor de red y el servidor de aplicación.
- En las pruebas de cobertura realizadas se observó que las construcciones tuvieron mayor afectación en la recepción de datos que la distancia, por lo que las construcciones representan una gran barrera afectando la potencia de la señal. La relación señal a ruido siempre fue la misma para todas las pruebas debido a que el factor de difusión no cambio.
- Al analizar los datos que se pierden en un tiempo de transmisión determinado, se concluye que entre menor sea el intervalo de tiempo entre mensajes, se acumulan más errores por la imprecisión del tiempo, lo que hace que se obtengan menos mensajes de los esperados, sin embargo, con este problema la efectividad fue alta con un 93 %. Además también se concluye que en la pérdida de datos hay más influencia del el tiempo entre mensaje que de la distancia y los obstáculos.
- El prototipo desarrollado en el proyecto tiene un costo muy bajo comparado con los aparatos que actualmente se encuentran en el mercado, además del bajo costo este permite medir mas indicadores de los que los dispositivos encontrados en el mercado realizan, lo que hace que sea un prototipo altamente competitivo en el mercado.

# 10. Trabajos futuros

Se espera que para trabajos futuros se cambie el IDE de arduino por el STM32CubeIDE propio del microcontrolador, este tiene una página en la que se pueden encontrar las librerías necesarias para realizar desarrollos, esto permitirá liberar espacio en la memoria FLASH y dar el primer paso para poder implementar la memoria de almacenamiento externo.

Otro trabajo futuro es el cambio de servidor de red o pagar para tener más beneficios que eliminen las limitaciones de la red en cuanto a mensajes de confirmación, dando el otro paso para poder implementar el almacenamiento externo. Para tener mejor control de los mensajes si se pudiera implementar el almacenamiento externo, se recomienda utilizar el reloj interno del STM32F103 por su facilidad de uso, que dará la hora exacta de la transmisión de los datos, pudiendo ordenarlos cronológicamente.

Para implementar una red más robusta se sugiere cambiar el gateway por uno multicanal que permita hacer saltos de frecuencia utilizando el protocolo LORAWAN, lograr la implementación de más de dos nodos y evitar las perdidas de datos. Con la mejora anterior, también sería posible realizar variaciones de los parámetros de la red que permitan mejorar la cobertura, por ejemplo cambiando el factor de difusión dependiendo de la distancia. Sin embargo, antes de realizar estas pruebas es prudente determinar la verdadera cobertura de la red, haciendo envíos de mensajes a mayores distancias que las realizadas por la situación actual.

El prototipo desarrollado cuenta con una caja que no brinda ninguna protección contra lluvias, polvo, fuertes vientos, entre otros factores que pueden alterar el funcionamiento de los sensores y evitar que se prolongue la vida útil de estos, por esta razón también se propone hacer un estudio sobre la protección más adecuada para este tipo de nodos, ya que debe realizarse un análisis de vientos y los sensores deben ubicarse de tal forma que puedan medir verdaderamente las variables para las que fueron diseñados.

Teniendo en cuenta los buenos resultados obtenidos con la implementación de las baterías usadas en cuanto a su duración, se propone realizar el análisis con baterías nuevas y de mayor capacidad con la que podría lograrse que la batería del nodo dure 24 horas o más. Además para que el nodo tenga mayor independencia, también se aconseja que el cargador sea reemplazado por un panel solar y se cuenten con dos juegos de baterías, que mientras unas alimentan el nodo, las otras se recarguen.

## Bibliografía


[1] VIVIENDA Y DESARROLLO TERRITORIAL MINISTERIO DE AMBIENTE. "Resolución Número 610 del 24 de Marzo de 2010, Establece la Norma de Calidad del Aire o Nivel de Inmisión." En: (24 de Marzo de 2010).

[2] MEDELLÍN CÓMO VAMOS. "Estudio en Colombia revela que la contaminación del aire afecta la salud humana." En: (11 de Octubre de 2018). URL: https://www.medellincomovamos.org/estudio-en-colombia-revela-que-la-contaminacion-del-aire-afecta-la-salud-humana.

[3] Secretaría Distrital de Ambiente. "Red de Monitoreo de Calidad del Aire de Bogotá - RMCAB." En: (2013). URL: http://ambientebogota.gov.co/red-de-calidad-del-aire.

[4] S. Abraham y X. Li. "A cost-effective wireless sensor network system for indoor air qualitymonitoring applications". En: *Procedia Comput. Sci.* vol: 34 (2014), pag: 165–171. URL: https://www.sciencedirect.com/science/article/pii/S1877050914009454.

[5] F. Osebor y A. Cheshmehzangi. "Indoor environment monitoring based on humidity conditions using a low-cost sensor network a low-cost sensor". En: *Energy Procedia* vol: 145 (2018), pag: 464–471. URL: https://www.sciencedirect.com/science/article/pii/S1876610218301024.

[6] G. V. McMorrow M. Karami y L. Wang. "Continuous monitoring of indoor environmental quality using an Arduino-based data acquisition system". En: vol: 19 (2018), pag: 412–419. URL: https://www.sciencedirect.com/science/article/pii/S2352710218301025.

[7] VIVIENDA Y DESARROLLO TERRITORIAL MINISTERIO DE AMBIENTE. "Resolución 2254 del Ministerio de Medio Ambiente y Desarrollo Sostenible del 2017". En: (2017). URL: https://www.minambiente.gov.co/images/normativa/app/resoluciones/96-res\%202254\%20de\%202017.pdf.

[8] "Contaminación del Aire Ambiental". En: *Organización Panamericana de la Salud* (2016). URL: https://www.paho.org/hq/index.php?option=com_content&view=article&id=12918:ambient-airpollution&Itemid=72243&lang=es.

[9] "La nube de smog sobre Bogotá genera costos por $2 billones." En: *Revista Dinero* (2019). URL: https://www.dinero.com/pais/articulo/cuanto-le-cuesta-la-contaminacion-del-aire-a-los-bogotanos/268113.

[10] "¿Qué respiran y de dónde proviene la contaminación en Bogotá?" En: *Revista Semana* (2018). URL: https://sostenibilidad.semana.com/medio-ambiente/articulo/calidad-del-aire-bogota-contaminacion-querespiran-y-de-donde-proviene-la-contaminacion-en-bogota/41330.

[11] "Vehículos viejos contaminan el aire hasta 40 veces más." En: *El Tiempo* (2018). URL: https://www.eltiempo.com/colombia/medellin/vehiculos-viejos-contaminan-el-aire-hasta-40-veces-mas272342.

[12] SIAC. *Calidad del aire*. URL: http://www.siac.gov.co/calidadaire.

[13] OMS. "Guías de calidad del aire de la OMS relativas al material particulado, el ozono, el dióxido de nitrógeno y el dióxido de azufre". En: (2005).

[14] "Nueva normativa para la calidad del aire en Colombia." En: *El Tiempo* (2017). URL: https://www.eltiempo.com/vida/medio-ambiente/quedo-lista-la-nueva-normativa-para-medir-la-calidad-del-aire-en-colombia-157122.

[15] " Manual de Operación de Sistemas de Vigilancia de Calidad de Aire del Protocolo para el Monitoreo y Seguimiento de la Calidad del Aire MAVDT (MINISTERIO DE AMBIENTE, VIVIENDA Y DESARROLLO TERRITORIAL)." En: *IDEAM* (2010). URL: http://www.ideam.gov.co/documents/51310/527391/Protocolo+para+el+Monitoreo+y+seguimiento+de+la+calidad+del+aire.pdf/6b2f53c8-6a8d-4f3d-b210-011a45f3ee88.

[16] "OZONO TROPOSFÉRICO." En: *IDEAM* (2010). URL: http://www.meteoaeronautica.gov.co/ozono-troposferico.

[17] "Material Particulado." En: *Instituto para la salud geoambiental* (2013). URL: https://www.saludgeoambiental.org/material-particulado.

[18] "El monóxido de carbono sí mata." En: *El Tiempo* (2007). URL: https://www.eltiempo.com/archivo/documento/MAM-2716382.

[19] "El dióxido de Azufre SO2." En: *Instituto para la salud geoambiental* (2013). URL: https://www.saludgeoambiental.org/dioxido-azufre-so2.

[20] "Concentración promedio anual de dióxido de azufre en el aire ($SO_2$)." En: *IDEAM* (2017).

[21] "Dióxido de Nitrogeno NO2." En: *Instituto para la salud geoambienta* (2013). URL: https://www.saludgeoambiental.org/dioxido-nitrogeno-no2.

[22] "Concentración promedio anual de dióxido de nitrógeno en el aire ($NO_2$)." En: *IDEAM* (2017).

[23] "The MiCS-6814 is a compact MOS sensor with three fully independent sensing elements on one package." En: *Sensortech* (2013-). URL: https://www.sgxsensortech.com/content/uploads/2015/02/1143_Datasheet-MiCS-6814-rev-8.pdf..

[24] "MiCS Application Note how to set-up load resistance for MiCS sensors measurements." En: *SGX sensortech* (2013-).

[25] "Sensor De Polvo DSM501A PM2.5 Medición De Aire De 5V Para Arduino - NB060." En: *RESEARCH* (2013-). URL: https://rees52.com/arduino-sensors/1107-dsm501a-dust-sensor-dsm-50-1a-nb060.

[26] "DUST SENSOR MODULE: DSM501A SPECIFICATIONS". En: *ISWEEK* (2013-). URL: https://www.isweek.com/Uploads/20150313/55028f2949640.pdf.

[27] "Adafruit BME280 Humidity + Barometric Pressure + Temperature, Sensor Breakout." En: *Adafruit* (2014). URL: https://www.marutsu.co.jp/contents/shop/marutsu/ds/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout.pdf.

[28] "BME280." En: *BOSCH* (2014). URL: https://www.embeddedadventures.com/datasheets/BME280.pdf.

[29] "¿Qué es un microcontrolador?." En: *ELECTRONICA ESTUDIO* (2019). URL: https://www.electronicaestudio.com/que-es-un-microcontrolador/.

[30] "STM32F103." En: *ARM* (2015). URL: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

[31] "Getting Started With Stm32f103c8 With Mbed+STLink V2." En: *ARM* (2018). URL: https://www.instructables.com/id/Getting-Started-With-Stm32f103c8-With-MbedSTLink-V/.

[32] "Tutorial Arduino: IDE Arduino." En: *OpenWebiiniars* (2019). URL: https://openwebinars.net/blog/tutorial-arduino-ide-arduino/).

[33] "STM32." En: *Luis Malla (ingeniería, informática y diseño)* (2019). URL: https://www.luisllamas.es/tag/stm32/.

[34] "Modulo Carga y Descarga Baterias 18650 3 Celdas 3S 10A." En: *Ferretrónica* (2010). URL: https://ferretronica.com/products/modulo-carga-y-descarga-baterias-18650-3-celdas-3s-10a.

[35] "Como funciona un BMS para baterias de litio?." En: *Javier Camacho* (2010). URL: https://www.youtube.com/watch?v=uyK3vAQFzos&feature=youtu.be.

[36] "Medidor Capacidad Carga Baterias 3 Celdas 3S 12.6V Azul." En: *Yorobotics* (2013). URL: https://yorobotics.co/inicio/974-medidor-capacidad-carga-baterias-3-celdas-3s-126v-azul.html.

[37] "LM2596 SIMPLE SWITCHER Power Converter 150-kHz 3-A Step-Down Voltage Regulator." En: *Texas Instruments* (2013). URL: https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1596507040167&ref_url=https\%253A\%252F\%252Fwww.google.com\%252F.

[38] "Módulo LM2596 Convertidor de Voltaje DC-DC Buck 1.25V-35V." En: *Electroniclab* (2013). URL: https://electronilab.co/tienda/modulo-lm2596-convertidor-de-voltaje-dc-dc-buck-1-25v-35v/.

[39] "Módulo de lectura y escritura de tarjeta SD." En: *AG Electrónica* (2018). URL: http://www.agspecinfo.com/pdfs/O/OKY3001.PDF.

[40] C. Y. Ordoñez Burbano. “IMPLEMENTACIÓN DE UNA RED DE SENSORES INALÁMBRICOS LPWAN MEDIANTE MÓDULOS LoRa PARA EL MONITOREO DE LA CALIDAD DEL AGUA EN 2 RÍOS”. En: *UNIVERSIDAD POLITÉCNICA DE MADRID* (2017). URL: http://repository.udistrital.edu.co/bitstream/11349/6433/1/BurbanoOrdo\%C3\%B1ezCristianYamith2017.pdf.

[41] I. N. G. Javier M. Valero I. N. G. Javier I. N. G. Manuel R. Garcia y M. Valero. “Guía para la implementación de redes de sensores inalámbricos (WSN).” En: (2010). URL: https://repository.upb.edu.co/bitstream/handle/20.500.11912/877/digital_19215.pdf?sequence=1..

[42] JORDAN C. TRIANA U. y RONALD E. RODRIGUEZ L. “Prototipo de solución IoT con tecnología LoRa en monitoreo de Cultivos Agrícolas .” En: *Universidad Distrital Francisco José de Caldas* (2017). URL: http://repository.udistrital.edu.co/bitstream/11349/13388/1/TrianaUsecheJordanCamilo2018.pdf.

[43] “Arquitectura IoT.” En: *IBM* (2017-). URL: https://www.ibm.com/developerworks/ssa/library/iot-lp201-iot-architectures/index.html.

[44] R. Mayobre. “Telemetría de contenedores de residuos.” En: *Universidad ORT Uruguay, Facultad de Ingeniería* (2018). URL: https://dspace.ort.edu.uy/handle/20.500.11968/3878.

[45] Laurent Clavier Umber Noreen Ahcene Bounceur. “A Study of LoRa Low Power and Wide Area Network Technology”. En: *3rd International Conference on Advanced Technologies, IEEE* (2017). URL: https://ieeexplore.ieee.org/document/8075570,.

[46] Semtech. “SX1276/77/78/79 DATASHEET”. En: *MOUSER* (2016). URL: https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf.

[47] JOSÉ DANIEL RODRIGUEZ MUNCA. “DISPOSITIVO LoRa DE COMUNICACIÓN A LARGO ALCANCE Y BAJO CONSUMO ENERGÉTICO PARA APLICACIONES DEL ÁMBITO DEL DESARROLLO”. En: *UNIVERSIDAD POLITÉCNICA DE MADRID* (2016). URL: http://oa.upm.es/44890/1/TFM_JOSE_DANIEL_RODRIGUEZ_MUNCA.pdf.

[48] L. Dopazo Gonzáles. “Test y desplieque de decnología de comunicaciones LoRa para aplicaciones de Internet of Things ”. En: *UNIVERSIDAD POLITÉCNICA DE MADRID* (2019). URL: http://oa.upm.es/54465/1/TFG_LAURA_DOPAZO_GONZALEZ.pdf.

[49] A. Campos M. Chacon. “Implementación de un Gateway Low-cost para el protocolo LoRa”. En: *Universidad Distrital Francisco José de Caldas* (2018). URL: http://repository.udistrital.edu.co/bitstream/11349/13877/1/CamposRamirezYeissonAlejandro2018.pdf.

[50] Michael A. Moya Q. “Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos”. En: *UNIVERSIDAD POLITÉCNICA DE VALENCIA* (2018). URL: https://riunet.upv.es/handle/10251/109791.

[51] LoRa Alliance. “LoRaWAN. What is it?” En: (2015). URL: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.

[52] The Things Network. *Planes de Frecuencia*. URL: https://www.thethingsnetwork.org/docs/lorawan/frequency-plans.html.

[53] Dragino. “LG02/OLG02 MANUAL”. En: (2018). URL: http://www.dragino.com/downloads/downloads/LoRa_Gateway/LG02-OLG02/LG02_LoRa_Gateway_User_Manual_v1.5.1.pdf.

[54] Dragino. *Módulo LoRa Bee con tecnología LoRa*. URL: https://www.dragino.com/products/lora/item/109-lora-bee.html.

[55] Mobilefish. “Build LoRa node using Arduino Uno and HopeRF RFM95 LoRa transceiver module”. En: (2017). URL: https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_build_lora_node_rfm95_arduino_uno.html.

[56] Dragino. *LG02 Dual Channels LoRa IoT Gateway*. URL: https://www.dragino.com/products/lora-lorawan-gateway/item/135-lg02.html.

[57] The Things Network. *Building a global open LoRaWAN™ network.* URL: https://www.thethingsnetwork.org/.

[58] The Things Network. “Limitaciones: velocidad de datos, tamaño de paquete, enlace ascendente de 30 segundos y enlace descendente de 10 mensajes por día Política de acceso justo [pautas]”. En: (2016). URL: https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy-guidelines/1300.

[59] ThingSpeak. *Aprenda más sobre ThingSpeak*. URL: https://thingspeak.com/pages/learn_more.

[60] Ubidots. *Acerca de Ubidots*. URL: https://ubidots.com/about/.

[61] Ubidots. *IoT y herramientas en la nube para construir su negocio*. URL: https://ubidots.com/platform/.

[62] Galvis B. y Pachón J. Lancheros-Cuesta D. “Dispositivo electrónico portable para la medición de la contaminación del aire. Ingenio Magno”. En: *Ingenio Magno* (2017). URL: https://dialnet.unirioja.es/descarga/articulo/6881751.pdf.

[63] C. Schneider C. Pfaffenbach M. S. Engel B. Paas y J. Fels. “Perceptual studies on air quality and sound through urban walks”. En: *Science Direct, Cities* vol: 83 (Diciembre 2017), pag: 173–185. URL: https://www.sciencedirect.com/science/article/pii/S0264275117314208.

[64] T. Cao J. Lin A. Chakma B. Vizena y J. Zhang. “Image-based air quality analysis using deep convolutional neural network”. En: *IEEE International Conference on Image Processing (ICIP), Beijing* (2017), pag: 3949-3952. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8297023&isnumber=8296222.

[65] S. Member H. Lee y K. Ke. “Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation”. En: *IEEE Trans. Instrum. Meas.* vol: 67 (2018), pag: 2177–2187. URL: http://ieeexplore.ieee.org/document/8326735/.

[66] J. Park y C. Kim. “Low Cost Fine-Grained Air Quality Monitoring System Using LoRa WAN”. En: *Int. Conf. Inf. Netw.* vol: 2019 (2019), pag: 439–441. URL: https://ieeexplore.ieee.org/document/8718193.

[67] Eliyani H. A. Winarno A. Muklish Misbah R. P. Astutik y D. Andesta. “Communication System on Wireless Sensor Networks using Raspberry Pi and Arduino for Monitoring Gas of Air Pollution”. En: *Int. Semin. Intell. Technol. Its Appl.* vol: 2019 (2018), pag: 27–32. URL: https://ieeexplore.ieee.org/document/8711371.

[68] A. Candia y L. A. Marrone. “Soluciones para SmartCities: propuesta de un sistema de monitoreo de la calidad del aire basado en una red LoRaWAN con sensores portátiles”. En: *Int. Semin. Intell. Technol. Its Appl.* vol: 2019 (2018), pag: 31–36. URL: http://sedici.unlp.edu.ar/bitstream/handle/10915/73116/Documento_completo.pdf?sequence=1&isAllowed=y.

[69] Luis Llamas. “STM32F103, EL COMPETIDOR ARM DE ARDUINO DE BAJO COSTE”. En: (2015). URL: https://www.luisllamas.es/stm32f103-el-competidor-arm-de-arduino-de-bajo-coste/.

[70] blog PLUSELECTRIC. “Arduino Uno: Especificaciones y características”. En: (2014). URL: https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/.

[71] Aswinth Raj. “Programación de la placa STM32F103C8 usando el puerto USB”. En: (2018). URL: https://circuitdigest.com/microcontroller-projects/programming-stm32f103c8-board-using-usb-port.

[72] Hobby Projects. “BME280, Arduino y BluePill”. En: (2019).

[73] GitHub. “Librería BME280 de finitspace”. En: (2019). URL: https://github.com/finitespace/BME280.

[74] GitHub. “Multichannel-gas de paulvha”. En: (2017). URL: https://github.com/paulvha/multichannel-gas.

[75] GitHub. “DSM501-interrupt de Sovichea”. En: (2019). URL: https://github.com/Sovichea/dsm501-interrupt.

[76] Dragino Wiki. “LORABEE”. En: (2017). URL: https://wiki.dragino.com/index.php?title=Lora_BEE.

[77] GitHub. *Librería arduino-lmic-master de codeload*. URL: https://codeload.github.com/dragino/arduino-lmic/zip/master.

[78] GitHub. “SX12XX-LoRa de StuartsProjects”. En: (2019). URL: https://github.com/LoRaTracker/SX12XX-LoRa/tree/master/examples/SX127x_examples/Basics/2_Register_Test.

[79] GitHub. “Librería SdFat-beta de greiman”. En: (2014). URL: https://github.com/greiman/SdFat-beta.

[80] Ing. Rodrigo Alejandro Tirapegui. “Sistema de monitoreo de presión y temperatura para neumáticos (TPMS)”. En: *UNIVERSIDAD DE BUENOS AIRES FACULTAD DE INGENIERÍA, CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS* (2018).

[81] Maria Carlina Hernandez. *Integre sus datos TTN con Ubidots - Configuración manual de Ubi-Function*. URL: https://help.ubidots.com/en/articles/1866089-integrate-your-ttn-data-with-ubidots-manual-ubifunction-setup.

[82] David Sepúlveda. *Integre sus datos TTN con Ubidots - Configuración simple*. URL: https://help.ubidots.com/en/articles/2362758-integrate-your-ttn-data-with-ubidots-simple-setup.

[83] David Sepúlveda. *Eventos: administre mensajes de enlace descendente con TTN y Ubidots*. URL: https://help.ubidots.com/en/articles/2687214-events-manage-downlink-messages-with-ttn-and-ubidots.

[84] David Sepúlveda. *Eventos: creación de eventos condicionales y alertas*. URL: https://help.ubidots.com/en/articles/1445537-events-creating-conditional-events-and-alerts.

[85] The Things Network. *Working with Bytes*. URL: https://www.thethingsnetwork.org/docs/devices/bytes.html.

[86] MATLAB. *Analyze Channel Data to Send Email Notification*. URL: https://www.mathworks.com/help/thingspeak//analyze-channel-data-to-send-email.html.

# 11. Anexos

## 11.1 Código nodo final con todos los sensores

```
//LIBRERIAS Y VARIABLES DE SENSORES Y MODULO LORABEE
        //LIBRER A LORABEE
        #include <lmic.h>
        #include <hal/hal.h>
        #include <SPI.h>

        //LIBRER A, PINES y VARIABLES DSM501A
        #include "DSM501_1.h"
        #define DSM501_PM1_0     PB11
        #define DSM501_PM2_5     PB10
        #define SAMPLE_TIME      60 // seconds
        DSM501 dsm501;
        float PMA;
        float PMB;
        int PMA1;
        int PMB1;

        //LIBRER A Y VARIABLES BME280
        #include <BME280I2C.h>
        #include <Wire.h>
        BME280I2C bme;
        float temp(NAN), hum(NAN), pres(NAN);
        signed int temp1;
        int hum1, pres1;
        int signo = 0;

        //VARIABLES  Y PINES MICS6814
        #define CO_RO 1756
        #define NO_RO 1300
        float PPM_CO;
        float PPM_NO;
        int PPM_CO1;
        int PPM_NO1;
        #define pinCO PA1
        #define pinNO PA0

//INDICADORES LUMINOSOS
        #define LEDLI PB1
        #define LEDTX PB0
        #include "SdFat.h"

//CLAVES DE RED, APLICACION Y DIRECCION DEL NODO
```

```
43      // LoRaWAN NwkSKey, network session key
44      // This is the default Semtech key, which
45      //is used by the early prototype TTN
46      // network.
47      static const PROGMEM u1_t NWKSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX
    , 0xXX
48      , 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX};
49
50      // LoRaWAN AppSKey, application session key
51      // This is the default Semtech key, which
52      //is used by the early prototype TTN
53      // network.
54      static const u1_t PROGMEM APPSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX
    , 0xXX
55      ,0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };
56
57      // LoRaWAN end-device address (DevAddr)
58
59      static const u4_t DEVADDR = 0xXXXXXXXX ; // <-- Change this address for
    every node!
60
61      // These callbacks are only used in over-the-air activation, so they are
62      // left empty here (we cannot leave them out completely unless
63      // DISABLE_JOIN is set in config.h, otherwise the linker will complain).
64      void os_getArtEui (u1_t* buf) { }
65      void os_getDevEui (u1_t* buf) { }
66      void os_getDevKey (u1_t* buf) { }
67
68 //CANTIDAD DE BYTES
69      byte mydata[16];
70
71
72    static osjob_t sendjob;
73
74 // Schedule TX every this many seconds (might become longer due to duty
75 // cycle limitations).
76 //CONFIGURACIN TIEMPO DE ENV O ENTRE MENSAJES
77    const unsigned TX_INTERVAL = 60;
78
79  //CONFIGURACIN DE PINES SPI-STM32
80  // Pin mapping
81  const lmic_pinmap lmic_pins = {
82    .nss = PA4,
83    .rxtx = LMIC_UNUSED_PIN,
84    .rst = LMIC_UNUSED_PIN,
85    .dio = {PA2,PA3, LMIC_UNUSED_PIN},
86  };
87
88 //FUNCI N DE EVENTOS DE LA RED
```

```
89      void onEvent (ev_t ev) {
90        Serial.print(os_getTime());
91        Serial.print(": ");
92        switch (ev) {
93          case EV_SCAN_TIMEOUT:
94            Serial.println("EV_SCAN_TIMEOUT");
95            break;
96          case EV_BEACON_FOUND:
97            Serial.println("EV_BEACON_FOUND");
98            break;
99          case EV_BEACON_MISSED:
100           Serial.println("EV_BEACON_MISSED");
101           break;
102         case EV_BEACON_TRACKED:
103           Serial.println("EV_BEACON_TRACKED");
104           break;
105         case EV_JOINING:
106           Serial.println("EV_JOINING");
107           break;
108         case EV_JOINED:
109           Serial.println("EV_JOINED");
110           break;
111         case EV_RFU1:
112           Serial.println("EV_RFU1");
113           break;
114         case EV_JOIN_FAILED:
115           Serial.println("EV_JOIN_FAILED");
116           break;
117         case EV_REJOIN_FAILED:
118           Serial.println("EV_REJOIN_FAILED");
119           break;
120         case EV_TXCOMPLETE:
121           Serial.println("EV_TXCOMPLETE (includes waiting for RX windows)");
122            digitalWrite(LEDTX , LOW);     // poner el Pin en LOW
123            delay(1000);
124           if (LMIC.txrxFlags & TXRX_ACK)
125             Serial.println("Received ack");
126
127           if (LMIC.dataLen) {
128             Serial.println("Received ");
129             Serial.println(LMIC.dataLen);
130             Serial.println(" bytes of payload");
131           }
132           // Schedule next transmission
133           os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL),
    do_send);
134           break;
135         case EV_LOST_TSYNC:
136           Serial.println("EV_LOST_TSYNC");
```

```
137             break;
138           case EV_RESET:
139             Serial.println("EV_RESET");
140             break;
141           case EV_RXCOMPLETE:
142             // data received in ping slot
143             Serial.println("EV_RXCOMPLETE");
144             break;
145           case EV_LINK_DEAD:
146             Serial.println("EV_LINK_DEAD");
147             break;
148           case EV_LINK_ALIVE:
149             Serial.println("EV_LINK_ALIVE");
150             break;
151           default:
152             Serial.println("Unknown event");
153             break;
154         }
155       }
156
157
158 //FUNCI N DONDE SE OBTIENEN LOS DATOS DE LOS SENSORES Y
159 //SE PONEN EN COLA PARA SER ENVIADOS
160         void do_send(osjob_t* j) {
161
162           digitalWrite(LEDTX , HIGH);    // poner el Pin en HIGH
163           delay(1000);
164           //-------------------------------------------------------//
165           //-----------------------Sensor BME280-----------------//
166
167           BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
168           BME280::PresUnit presUnit(BME280::PresUnit_Pa);
169
170           bme.read(pres, temp, hum, tempUnit, presUnit);
171           pres = pres / 133.32;
172           pres1 = pres * 100;
173           temp1 = temp * 100;
174           hum1 = hum * 100;
175           if (temp1 < 0) {
176             signo = 1;
177           } else {
178             signo = 0;
179           }
180
181          // Serial.println(pres1);
182          // Serial.println(temp1);
183          // Serial.println(hum1);
184
185           mydata[0] = highByte(pres1);
```

```
186           mydata[1] = lowByte(pres1);
187           mydata[2] = signo;
188           mydata[3] = highByte(abs(temp1));
189           mydata[4] = lowByte(abs(temp1));
190           mydata[5] = highByte(hum1);
191           mydata[6] = lowByte(hum1);
192
193           //-------------------------------------------//
194           //----------------Sensor DSM501A---------//
195            if (dsm501.update()){
196          PMA = dsm501.getParticleCount(0) * 0.002179;
197          PMB = dsm501.getParticleCount(1) * 0.002179;}
198
199        if (PMA>PMB){
200              PMA = PMA-PMB;
201              }
202            if(PMA>500.00){
203            PMA=500.00; }
204
205           if(PMB>604.00){
206            PMB=604.00;}
207
208           PMA1 = PMA * 100; //PM2.5
209           PMB1 = PMB * 100; //PM10
210
211
212           mydata[7]= highByte(PMB1);
213           mydata[8]= lowByte(PMB1);
214           mydata[9]= highByte(PMA1);
215           mydata[10]= lowByte(PMA1);
216
217           //-------------------------------------------//
218           //----------------Sensor MICS6814---------//
219
220
221        Serial.println(analogRead(pinNO));
222        Serial.println(analogRead(pinCO));
223           PPM_CO = pow(float(analogRead(pinCO)) / float(CO_RO), -1.179) * 4.385;
224           PPM_NO = pow(float(NO_RO)/ float(analogRead(pinNO)), 1.007) * 0.1459;
225
226
227            if (PPM_CO > 50.36) {
228
229              PPM_CO = 50.36;
230           }
231
232           if (PPM_CO < 1.00) {
233
234             PPM_CO = 1.00;
```

```
235           }
236
237           if (PPM_NO > 2.05) {
238
239             PPM_NO = 2.05;
240           }
241           if (PPM_NO < 0.05) {
242
243             PPM_NO = 0.05;
244           }
245
246           Serial.println(PPM_CO);
247           Serial.println(PPM_NO);
248
249           PPM_CO1 = PPM_CO*100;
250           PPM_NO1 = PPM_NO*100;
251           mydata[11]= highByte(PPM_CO1);
252           mydata[12]= lowByte(PPM_CO1);
253           mydata[13]= highByte(PPM_NO1);
254           mydata[14]= lowByte(PPM_NO1);
255
256           //-------------------------------------------------//
257           //-------------------------------------------------//
258
259           // Check if there is not a current TX/RX job running
260           if (LMIC.opmode & OP_TXRXPEND) {
261             Serial.println("OP_TXRXPEND, not sending");
262           } else {
263             // Prepare upstream data transmission at the next possible time.
264             LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 0);
265             Serial.println("Packet queued");
266
267           }
268                               // esperar un segundo
269
270
271           // Next TX is scheduled after TX_COMPLETE event.
272         }
273
274 //SETUP DIVERSAS CONFIGURACIONES
275       void setup() {
276         Serial.begin(115200);
277         Serial.println("Starting");
278         pinMode(LEDLI,OUTPUT);
279         pinMode(LEDTX,OUTPUT);
280
281         //-----------------------------------------//
282         //-------------------Sensor BME280-----//
283         Wire.begin();
```

```
284
285         while (!bme.begin())
286         {
287           Serial.println("Could not find BME280 sensor!");
288           delay(1000);
289         }
290
291         // bme.chipID(); // Deprecated. See chipModel().
292         switch (bme.chipModel())
293         {
294           case BME280::ChipModel_BME280:
295             Serial.println("Found BME280 sensor! Success.");
296             break;
297           case BME280::ChipModel_BMP280:
298             Serial.println("Found BMP280 sensor! No Humidity available.");
299             break;
300           default:
301             Serial.println("Found UNKNOWN sensor! Error!");
302         }
303
304         //-------------------------------------------------------------------//
305         //----------------------Sensor DSM501A-------------------------------//
306
307         dsm501.begin(DSM501_PM1_0, DSM501_PM2_5, SAMPLE_TIME); // collect sensor
    data every SAMPLE_TIME
308
309         // Wait 60s for DSM501 to warm up
310         Serial.println("Wait 60s for DSM501 to warm up");
311         for (int i = 1; i <= 60; i++)
312         {
313           delay(1000); // 1s
314           Serial.print(i);
315           Serial.println(" s (wait 60s for DSM501 to warm up)");
316         }
317
318         //-------------------------------------------------//
319         digitalWrite(LEDLI , HIGH);    // poner el Pin en HIGH
320         delay(1000);
321       #ifdef VCC_ENABLE
322         // For Pinoccio Scout boards
323         pinMode(VCC_ENABLE, OUTPUT);
324         digitalWrite(VCC_ENABLE, HIGH);
325         delay(1000);
326       #endif
327
328         // LMIC init
329         os_init();
330         // Reset the MAC state. Session and pending data transfers will be
    discarded.
```

```
331         LMIC_reset();
332
333         // Set static session parameters. Instead of dynamically establishing a
    session
334         // by joining the network, precomputed session parameters are be provided.
335       #ifdef PROGMEM
336         // On AVR, these values are stored in flash and only copied to RAM
337         // once. Copy them to a temporary buffer here, LMIC_setSession will
338         // copy them into a buffer of its own again.
339         uint8_t appskey[sizeof(APPSKEY)];
340         uint8_t nwkskey[sizeof(NWKSKEY)];
341         memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
342         memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
343         LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
344       #else
345         // If not running an AVR with PROGMEM, just use the arrays directly
346         LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
347       #endif
348
349       #if defined(CFG_eu868)
350
351         LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
352         LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI
    );      // g-band
353         LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
354         LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
355         LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
356         LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
357         LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
358         LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI
    );      // g-band
359         LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),  BAND_MILLI
    );      // g2-band
360
361       #elif defined(CFG_us915)
362         // NA-US channels 0-71 are configured automatically
363         // but only one group of 8 should (a subband) should be active
364         // TTN recommends the second sub band, 1 in a zero based count.
365         // https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-
    global_conf.json
366         LMIC_selectSubBand(1);
367       #endif
368
```

```
369         // Disable link check validation
370         LMIC_setLinkCheckMode(0);
371 
372         // TTN uses SF9 for its RX2 window.
373         //LMIC.dn2Dr = DR_SF9;
374 
375         // Set data rate and transmit power for uplink (note: txpow seems to be
    ignored by the library)
376         LMIC_setDrTxpow(DR_SF7, 14);
377 
378         // Start job
379         do_send(&sendjob);
380       }
381 
382 //CICLO PRINCIPAL
383     void loop() {
384       os_runloop_once();
385     }
```

## 11.2 Código nodo final con emulador de MICS6814

```
1 
2 //LIBRERIAS Y VARIABLES DE SENSORES Y MODULO LORABEE
3         //LIBRER A LORABEE
4         #include <lmic.h>
5         #include <hal/hal.h>
6         #include <SPI.h>
7 
8         //LIBRER A BME280
9         #include <BME280I2C.h>
10        #include <Wire.h>
11        BME280I2C bme;
12 
13        //LIBRER A Y PINES DSM501A
14        #include "DSM501_1.h"
15        #define DSM501_PM1_0     PB11
16        #define DSM501_PM2_5     PB10
17        #define SAMPLE_TIME      30 // seconds
18        DSM501 dsm501;
19 
20        //VARIABLES BME280
21        float temp(NAN), hum(NAN), pres(NAN);
22        signed int temp1;
23        int hum1, pres1;
24        int signo = 0;
25 
26        //VARIABLES DSM501A
27        float PMA;
28        float PMB;
```

```
29          int PMA1;
30          int PMB1;
31
32          //VARIABLES MICS6814
33          int PPM_CO1;
34          int PPM_NO1;
35
36          \\INDICADORES LUMINOSOS
37          #define LEDLI PB1
38          #define LEDTX PB0
39
40  //CLAVES DE RED, APLICACION Y DIRECCION DEL NODO
41          // LoRaWAN NwkSKey, network session key
42          // This is the default Semtech key, which is used by the early prototype
        TTN
43          // network.
44
45          static const PROGMEM u1_t NWKSKEY[16] = {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0
        xXX,
46          0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };
47
48          // LoRaWAN AppSKey, application session key
49          // This is the default Semtech key, which is used by the early prototype
        TTN
50          // network.
51
52          static const u1_t PROGMEM APPSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0
        xXX,
53          0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };
54
55          // LoRaWAN end-device address (DevAddr)
56          //nuevo correo
57          static const u4_t DEVADDR = 0x260212C8 ; // <-- Change this address for
        every node!
58
59          // These callbacks are only used in over-the-air activation, so they are
60          // left empty here (we cannot leave them out completely unless
61          // DISABLE_JOIN is set in config.h, otherwise the linker will complain).
62          void os_getArtEui (u1_t* buf) { }
63          void os_getDevEui (u1_t* buf) { }
64          void os_getDevKey (u1_t* buf) { }
65
66  //CANTIDAD DE BYTES
67          byte mydata[16];
68          static osjob_t sendjob;
69
70  //TIEMPO ENTRE MENSAJES
71          // Schedule TX every this many seconds (might become longer due to duty
72          // cycle limitations).
```

```
73          const unsigned TX_INTERVAL = 60;
74
75  //PINES LORABEE
76          // Pin mapping
77          const lmic_pinmap lmic_pins = {
78            //       .nss = 10,
79            //       .rxtx = LMIC_UNUSED_PIN,
80            //       .rst = LMIC_UNUSED_PIN,
81            //       .dio = {9,8, LMIC_UNUSED_PIN},
82            .nss = PA4,
83            .rxtx = LMIC_UNUSED_PIN,
84            .rst = LMIC_UNUSED_PIN,
85            .dio = {PA2, PA3, LMIC_UNUSED_PIN},
86          };
87
88  //FUNCI N DE EVENTOS DE LA RED
89          void onEvent (ev_t ev) {
90            Serial.print(os_getTime());
91            Serial.print(": ");
92            switch (ev) {
93              case EV_SCAN_TIMEOUT:
94                Serial.println("EV_SCAN_TIMEOUT");
95                break;
96              case EV_BEACON_FOUND:
97                Serial.println("EV_BEACON_FOUND");
98                break;
99              case EV_BEACON_MISSED:
100               Serial.println("EV_BEACON_MISSED");
101               break;
102             case EV_BEACON_TRACKED:
103               Serial.println("EV_BEACON_TRACKED");
104               break;
105             case EV_JOINING:
106               Serial.println("EV_JOINING");
107               break;
108             case EV_JOINED:
109               Serial.println("EV_JOINED");
110               break;
111             case EV_RFU1:
112               Serial.println("EV_RFU1");
113               break;
114             case EV_JOIN_FAILED:
115               Serial.println("EV_JOIN_FAILED");
116               break;
117             case EV_REJOIN_FAILED:
118               Serial.println("EV_REJOIN_FAILED");
119               break;
120             case EV_TXCOMPLETE:
121               Serial.println("EV_TXCOMPLETE (includes waiting for RX windows)");
```

```
122               digitalWrite(LEDTX , LOW);     // poner el Pin en LOW
123               delay(1000);
124               if (LMIC.txrxFlags & TXRX_ACK)
125                 Serial.println("Received ack");
126
127               if (LMIC.dataLen) {
128                 Serial.println("Received ");
129                 Serial.println(LMIC.dataLen);
130                 Serial.println(" bytes of payload");
131               }
132               // Schedule next transmission
133               os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL
    ), do_send);
134               break;
135             case EV_LOST_TSYNC:
136               Serial.println("EV_LOST_TSYNC");
137               break;
138             case EV_RESET:
139               Serial.println("EV_RESET");
140               break;
141             case EV_RXCOMPLETE:
142               // data received in ping slot
143               Serial.println("EV_RXCOMPLETE");
144               break;
145             case EV_LINK_DEAD:
146               Serial.println("EV_LINK_DEAD");
147               break;
148             case EV_LINK_ALIVE:
149               Serial.println("EV_LINK_ALIVE");
150               break;
151             default:
152               Serial.println("Unknown event");
153               break;
154           }
155         }
156
157 //FUNCI N DE LECTURA DE SENSORES
158 void do_send(osjob_t* j) {
159
160           digitalWrite(LEDTX , HIGH);    // poner el Pin en HIGH
161           delay(1000);
162           //-----------------------------//
163           //-------Sensor BME280--------//
164
165           BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
166           BME280::PresUnit presUnit(BME280::PresUnit_Pa);
167
168           bme.read(pres, temp, hum, tempUnit, presUnit);
169           pres = pres / 133.32;
```

```
170           pres1 = pres * 100;
171           temp1 = temp * 100;
172           hum1 = hum * 100;
173           if (temp1 < 0) {
174             signo = 1;
175           } else {
176             signo = 0;
177           }
178
179           mydata[0] = highByte(pres1);
180           mydata[1] = lowByte(pres1);
181           mydata[2] = signo;
182           mydata[3] = highByte(abs(temp1));
183           mydata[4] = lowByte(abs(temp1));
184           mydata[5] = highByte(hum1);
185           mydata[6] = lowByte(hum1);
186
187           //---------------------//
188           //---------Sensor DSM501A--------//
189           if (dsm501.update()) {
190             PMA = dsm501.getParticleCount(0) * 0.002179;
191             PMB = dsm501.getParticleCount(1) * 0.002179;
192           }
193
194           if (PMA>PMB){
195               PMA = PMA-PMB;
196               }
197             if(PMA>500.00){
198             PMA=500.00; }
199
200           if(PMB>604.00){
201            PMB=604.00;}
202
203           PMA1 = PMA * 100; //PM2.5
204           PMB1 = PMB * 100; //PM10
205
206           mydata[7] = highByte(PMB1);
207           mydata[8] = lowByte(PMB1);
208           mydata[9] = highByte(PMA1);
209           mydata[10] = lowByte(PMA1);
210
211           //------------------------//
212           //--------Sensor MICS6814-------//
213           PPM_CO1 = random(100, 5037);
214           PPM_NO1 = random(5, 205);
215           mydata[11] = byte((PPM_CO1 & 0xFF0000) >> 16);
216           mydata[12] = byte((PPM_CO1 & 0x00FF00) >> 8);
217           mydata[13] = byte(PPM_CO1 & 0x0000FF);
218           mydata[14] = highByte(PPM_NO1);
```

```
219           mydata[15] = lowByte(PPM_NO1);
220
221
222           //--------------------//
223
224           // Check if there is not a current TX/RX job running
225           if (LMIC.opmode & OP_TXRXPEND) {
226             Serial.println("OP_TXRXPEND, not sending");
227           } else {
228             // Prepare upstream data transmission at the next possible time.
229             LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 0);
230             Serial.println("Packet queued"); }
231
232           // Next TX is scheduled after TX_COMPLETE event.
233         }
234
235 //SETUP
236         void setup() {
237           Serial.begin(115200);
238           Serial.println("Starting");
239           pinMode(LEDLI, OUTPUT);
240           pinMode(LEDTX, OUTPUT);
241           //--------------------//
242           //--------Sensor BME280----------//
243           Wire.begin();
244
245           while (!bme.begin())
246           {
247             Serial.println("Could not find BME280 sensor!");
248             delay(1000);
249           }
250
251           // bme.chipID(); // Deprecated. See chipModel().
252           switch (bme.chipModel())
253           {
254             case BME280::ChipModel_BME280:
255               Serial.println("Found BME280 sensor! Success.");
256               break;
257             case BME280::ChipModel_BMP280:
258               Serial.println("Found BMP280 sensor! No Humidity available.");
259               break;
260             default:
261               Serial.println("Found UNKNOWN sensor! Error!");
262           }
263
264           //----------------------//
265           //----------Sensor DSM501A-------//
266
267           dsm501.begin(DSM501_PM1_0, DSM501_PM2_5, SAMPLE_TIME); // collect sensor
```

```
    data every SAMPLE_TIME
268
269         // Wait 60s for DSM501 to warm up
270         Serial.println("Wait 60s for DSM501 to warm up");
271         for (int i = 1; i <= 60; i++)
272         {
273           delay(1000); // 1s
274           Serial.print(i);
275           Serial.println(" s (wait 60s for DSM501 to warm up)");
276         }
277
278         //-----------------------------//
279         digitalWrite(LEDLI , HIGH);    // poner el Pin en HIGH
280         delay(1000);
281       #ifdef VCC_ENABLE
282         // For Pinoccio Scout boards
283         pinMode(VCC_ENABLE, OUTPUT);
284         digitalWrite(VCC_ENABLE, HIGH);
285         delay(1000);
286       #endif
287
288         // LMIC init
289         os_init();
290         // Reset the MAC state. Session and pending data transfers will be
    discarded.
291         LMIC_reset();
292
293         // Set static session parameters. Instead of dynamically establishing a
    session
294         // by joining the network, precomputed session parameters are be
    provided.
295       #ifdef PROGMEM
296         // On AVR, these values are stored in flash and only copied to RAM
297         // once. Copy them to a temporary buffer here, LMIC_setSession will
298         // copy them into a buffer of its own again.
299         uint8_t appskey[sizeof(APPSKEY)];
300         uint8_t nwkskey[sizeof(NWKSKEY)];
301         memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
302         memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
303         LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
304       #else
305         // If not running an AVR with PROGMEM, just use the arrays directly
306         LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
307       #endif
308
309       #if defined(CFG_eu868)
310
311         LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
```

```
312           LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
    BAND_CENTI);       // g-band
313           LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
314           LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
315           LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
316           LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
317           LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
318           LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
319           LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
    BAND_MILLI);       // g2-band
320
321         #elif defined(CFG_us915)
322           // NA-US channels 0-71 are configured automatically
323           // but only one group of 8 should (a subband) should be active
324           // TTN recommends the second sub band, 1 in a zero based count.
325           // https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-
    global_conf.json
326           LMIC_selectSubBand(1);
327         #endif
328
329           // Disable link check validation
330           LMIC_setLinkCheckMode(0);
331
332           // TTN uses SF9 for its RX2 window.
333           //LMIC.dn2Dr = DR_SF9;
334
335           // Set data rate and transmit power for uplink (note: txpow seems to be
    ignored by the library)
336           LMIC_setDrTxpow(DR_SF7, 14);
337
338           // Start job
339           do_send(&sendjob);
340         }
341
342 //LOOP PRINCIPAL
343         void loop() {
344           os_runloop_once();
345         }
```

### 11.3 Código nodo prueba de memoria SD

```
1
2 //LIBRER AS Y VARIABLES DEL SENSOR Y MEMORIA SD
```

```
3         #include <BME280I2C.h>
4         #include <Wire.h>
5         BME280I2C bme;
6
7         #include <lmic.h>
8         #include <hal/hal.h>
9         #include <SPI.h>
10        float temp(NAN),hum(NAN), pres(NAN);
11        signed int temp1;
12        int hum1,pres1;
13        int signo=0;
14
15        #include "SdFat.h"
16        static SPIClass mySPI2(2);
17        #define SD2_CONFIG SdSpiConfig(PB12, DEDICATED_SPI, SD_SCK_MHZ(16), &
      mySPI2)
18        SdFat SD;
19
20        File myFile;
21        int i = 0;
22        int Ultimaposicion = 0;
23        boolean vcon=true;
24        int cont=0;
25
26
27  //CLAVES Y DIRECCI N M DULO LORABEE
28        static const PROGMEM u1_t NWKSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0
      xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };
29        static const u1_t PROGMEM APPSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0
      xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };
30        static const u4_t DEVADDR = 0xXXXXXXXX ; // <— Change this address for
      every node!
31
32        void os_getArtEui (u1_t* buf) { }
33        void os_getDevEui (u1_t* buf) { }
34        void os_getDevKey (u1_t* buf) { }
35
36  //VECTOR DE DATOS
37        byte mydata[9];
38        int x=0;
39        static osjob_t sendjob;
40
41  //TIEMPO ENTRE MENSAJES
42        // Schedule TX every this many seconds (might become longer due to duty
43        // cycle limitations).
44        int TX_INTERVAL = 180;
45
46  //PINES LORABEE
47        // Pin mapping
```

```
48          const lmic_pinmap lmic_pins = {
49              .nss = PA4,
50              .rxtx = LMIC_UNUSED_PIN,
51              .rst = LMIC_UNUSED_PIN,
52              .dio = {PA0,PA1, LMIC_UNUSED_PIN},
53          };
54
55  //FUNCI N EVENTOS DE RED
56          void onEvent (ev_t ev) {
57              Serial.print(os_getTime());
58              Serial.print(": ");
59              switch(ev) {
60                  case EV_SCAN_TIMEOUT:
61                      Serial.println(F("EV_SCAN_TIMEOUT"));
62                      break;
63                  case EV_BEACON_FOUND:
64                      Serial.println(F("EV_BEACON_FOUND"));
65                      break;
66                  case EV_BEACON_MISSED:
67                      Serial.println(F("EV_BEACON_MISSED"));
68                      break;
69                  case EV_BEACON_TRACKED:
70                      Serial.println(F("EV_BEACON_TRACKED"));
71                      break;
72                  case EV_JOINING:
73                      Serial.println(F("EV_JOINING"));
74                      break;
75                  case EV_JOINED:
76                      Serial.println(F("EV_JOINED"));
77                      break;
78                  case EV_RFU1:
79                      Serial.println(F("EV_RFU1"));
80                      break;
81                  case EV_JOIN_FAILED:
82                      Serial.println(F("EV_JOIN_FAILED"));
83                      break;
84                  case EV_REJOIN_FAILED:
85                      Serial.println(F("EV_REJOIN_FAILED"));
86                      break;
87                  case EV_TXCOMPLETE:
88                      Serial.println(("EV_TXCOMPLETE (includes waiting for RX
    windows)"));
89                      if (LMIC.txrxFlags & TXRX_ACK)
90                        Serial.println(F("Received ack"));
91
92                      if (LMIC.dataLen) {
93                        Serial.println("Received ");
94                        Serial.println(LMIC.dataLen);
95                        Serial.println(" bytes of payload");
```

```
96                        if (cont==0){
97                          vcon=true;
98                          if(SD.exists("datalog.txt"))
99                          {SD.remove("datalog.txt"); }
100                         TX_INTERVAL=180;
101                         }else{vcon=false;cont=cont-1; TX_INTERVAL=30;}
102                       }else{
103                        Serial.println("no hay");
104                        guardar(); cont=cont+1;
105                        vcon=true;
106                        TX_INTERVAL=180;}
107                       // Schedule next transmission
108                       os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(
    TX_INTERVAL), do_send);
109                       break;
110                 case EV_LOST_TSYNC:
111                       Serial.println(F("EV_LOST_TSYNC"));
112                       break;
113                 case EV_RESET:
114                       Serial.println(F("EV_RESET"));
115                       break;
116                 case EV_RXCOMPLETE:
117                       // data received in ping slot
118                       Serial.println(F("EV_RXCOMPLETE"));
119                       break;
120                 case EV_LINK_DEAD:
121                       Serial.println(F("EV_LINK_DEAD"));
122                       break;
123                 case EV_LINK_ALIVE:
124                       Serial.println(F("EV_LINK_ALIVE"));
125                       break;
126                  default:
127                       Serial.println(F("Unknown event"));
128                       break;
129             }
130         }
131
132 //FUNCI N PONE EN COLA DATOS
133
134         void do_send(osjob_t* j){
135
136           if(vcon){
137             lec_act();}
138           else{
139             lectura();}
140
141            mydata[0]=highByte(pres1);
142            mydata[1]=lowByte(pres1);
143            mydata[2]=signo;
```

```
144             mydata[3]=highByte(temp1);
145             mydata[4]=lowByte(temp1);
146             mydata[5]=highByte(hum1);
147             mydata[6]=lowByte(hum1);
148 
149           if (x==0){
150             mydata[7]=byte(1);x=1;
151             }else{mydata[7]=byte(0);
152             x=0;
153             }
154 
155             // Check if there is not a current TX/RX job running
156             if (LMIC.opmode & OP_TXRXPEND) {
157                 Serial.println(F("OP_TXRXPEND, not sending"));
158             } else {
159                 // Prepare upstream data transmission at the next possible time.
160                 LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
161                 Serial.println(F("Packet queued"));
162 
163             }
164 
165 
166             // Next TX is scheduled after TX_COMPLETE event.
167         }
168 
169 //SETUP
170 
171         void setup() {
172             Serial.begin(115200);
173             Serial.println(F("Starting"));
174 
175             //Sensor BME280
176             Wire.begin();
177 
178           while(!bme.begin())
179           {
180             Serial.println("Could not find BME280 sensor!");
181             delay(1000);
182           }
183 
184           Serial.print("Initializing SD card...");
185 
186           if (!SD.begin(SD2_CONFIG)) {
187             Serial.println("initialization failed!");
188             return;
189           }
190           Serial.println("initialization done.");
191 
192           // bme.chipID(); // Deprecated. See chipModel().
```

```
193         switch(bme.chipModel())
194         {
195            case BME280::ChipModel_BME280:
196              Serial.println("Found BME280 sensor! Success.");
197              break;
198            case BME280::ChipModel_BMP280:
199              Serial.println("Found BMP280 sensor! No Humidity available.");
200              break;
201            default:
202              Serial.println("Found UNKNOWN sensor! Error!");
203         }
204
205
206           #ifdef VCC_ENABLE
207           // For Pinoccio Scout boards
208           pinMode(VCC_ENABLE, OUTPUT);
209           digitalWrite(VCC_ENABLE, HIGH);
210           delay(1000);
211           #endif
212
213           // LMIC init
214           os_init();
215           // Reset the MAC state. Session and pending data transfers will be
    discarded.
216           LMIC_reset();
217
218           // Set static session parameters. Instead of dynamically establishing
    a session
219           // by joining the network, precomputed session parameters are be
    provided.
220           #ifdef PROGMEM
221           // On AVR, these values are stored in flash and only copied to RAM
222           // once. Copy them to a temporary buffer here, LMIC_setSession will
223           // copy them into a buffer of its own again.
224           uint8_t appskey[sizeof(APPSKEY)];
225           uint8_t nwkskey[sizeof(NWKSKEY)];
226           memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
227           memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
228           LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
229           #else
230           // If not running an AVR with PROGMEM, just use the arrays directly
231           LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
232           #endif
233
234           #if defined(CFG_eu868)
235
236           LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);       // g-band
237           LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
```

```
    BAND_CENTI);        // g-band
238             LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
239             LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
240             LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
241             LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
242             LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
243             LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
    BAND_CENTI);        // g-band
244             LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
    BAND_MILLI);        // g2-band
245
246             #elif defined(CFG_us915)
247             // NA-US channels 0-71 are configured automatically
248             // but only one group of 8 should (a subband) should be active
249             // TTN recommends the second sub band, 1 in a zero based count.
250             // https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-
    global_conf.json
251             LMIC_selectSubBand(1);
252             #endif
253
254             // Disable link check validation
255             LMIC_setLinkCheckMode(0);
256
257             // TTN uses SF9 for its RX2 window.
258             //LMIC.dn2Dr = DR_SF9;
259
260             // Set data rate and transmit power for uplink (note: txpow seems to
    be ignored by the library)
261             LMIC_setDrTxpow(DR_SF7,14);
262
263             // Start job
264             do_send(&sendjob);
265         }
266
267 //FUNCI N QUE GUARDA LOS DATOS EN LA SD
268         void guardar() {
269
270           myFile = SD.open("datalog.txt", FILE_WRITE);//abrimos  el archivo
271           if (myFile) {
272             myFile.print(pres1);
273             myFile.print(",");
274             myFile.print(signo);
275             myFile.print(",");
276             myFile.print(temp1);
```

```
277             myFile.print(",");
278             myFile.print(hum1);
279             myFile.println(",");
280             myFile.close(); }}
281
282 //FUNCI N QUE LEE LOS DATOS DE LA SD
283         void lectura() {
284           int c1 = 0;
285           myFile = SD.open("datalog.txt");//abrimos  el archivo
286           int totalBytes = myFile.size();
287           String cadena = "";
288
289           if (myFile) {
290             if (Ultimaposicion >= totalBytes)   Ultimaposicion = 0;
291             myFile.seek(Ultimaposicion);
292
293             //--Leemos una l nea de la hoja de texto----------
294             while (myFile.available()) {
295
296               char caracter = myFile.read();
297               cadena = cadena + caracter;
298               Ultimaposicion = myFile.position();
299               if (caracter == 10) //ASCII de nueva de l nea
300               {
301                 break;
302               }
303             }
304             //-----------------------------------------------------
305             myFile.close(); //cerramos el archivo
306
307           }
308
309           String temp = "";
310           for (i = 0; i < (cadena.length() - 2); i++)
311           {
312
313             if (cadena[i] != ',') {
314
315               temp = temp + cadena[i];
316
317             } else {
318
319               switch (c1) {
320                 case 0:
321                   pres1 = temp.toInt();
322                   break;
323                 case 1:
324                   signo = temp.toInt();
325                   break;
```

```
                case 2:
                  temp1 = temp.toInt();
                  break;
                case 3:
                  hum1 = temp.toInt();
                  break;}
              temp = "";
              c1 = c1 + 1;
            }}}

//FUNCI N QUE LEE LOS DATOS DEL SENSOR
        void lec_act(){
          BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
           BME280::PresUnit presUnit(BME280::PresUnit_Pa);

           bme.read(pres, temp, hum, tempUnit, presUnit);
            pres=(pres/133.32);
            pres1=pres*100;
            temp1=temp*100;
            hum1=hum*100;
            if (temp<0){signo=1;}else{signo=0;}
          }

//FUNCI N BUCLE PRINCIPAL
        void loop() {
            os_runloop_once();
        }
```

## 11.4 Código decodificador TTN en lenguaje Javascript

```
function Decoder(b, port) {

  //BME 280
  var pre= (b[0]<<8)+b[1];
  pre=pre/100;

  var temp= (b[3]<<8)+b[4];
  temp=temp/100;
  if (b[2]==1){temp=temp*-1;}

  var hum= (b[5]<<8)+b[6];
  hum=hum/100;
  //PM10
  var PMA= (b[7]<<8)+b[8];
 PMA = PMA/100;
  //PM2.5
 var PMB= (b[9]<<8)+b[10];
 PMB = PMB/100;

```

```
  var C= (b[11]<<8)+b[12];
 C = ((C/100)*0.0409*28.01)*1000;
 C=C.toFixed(2);

 var N= (b[13]<<8)+b[14];
  N = ((N/100)*0.0409*46.0055)*1000;
  N=N.toFixed(2);
//———
//——— PM _ 10

var icapm1=0;
var iapm1=0;
var ibpm1=0;
var pcapm1=1;
var pcbpm1=0;

//——— PM _ 2.5

var icapm=0;
var iapm=0;
var ibpm=0;
var pcapm=1;
var pcbpm=0;

//__CO

var icac=0;
var iac=0;
var ibc=0;
var pcac=1;
var pcbc=0;

//———NO2
var ican=0;
var ian=0;
var ibn=0;
var pcan=1;
var pcbn=0;
//_—— Valor del mayor ICA

var mayorICA;

//PM_10
if (PMA>=0 && PMA<=54){
  iapm1=50;
  ibpm1=0;
  pcapm1=54;
  pcbpm1=0;
}
```

```
69
70 if (PMA>=55 && PMA<=154){
71   iapm1=100;
72   ibpm1=51;
73   pcapm1=154;
74   pcbpm1=55;
75 }
76
77 if (PMA>=155 && PMA<=254){
78   iapm1=150;
79   ibpm1=101;
80   pcapm1=254;
81   pcbpm1=155;
82 }
83 if (PMA>=255 && PMA<=354){
84   iapm1=200;
85   ibpm1=151;
86   pcapm1=354;
87   pcbpm1=255;
88 }
89 if (PMA>=355 && PMA<=424){
90   iapm1=300;
91   ibpm1=201;
92   pcapm1=424;
93   pcbpm1=355;
94 }
95 if (PMA>=425 && PMA<=604){
96   iapm1=500;
97   ibpm1=301;
98   pcapm1=604;
99   pcbpm1=425;
100 }
101
102
103 //PM2_5
104 if (PMB>=0 && PMB<=12){
105   iapm=50;
106   ibpm=0;
107   pcapm=12;
108   pcbpm=0;
109 }
110
111 if (PMB>=13 && PMB<=37){
112   iapm=100;
113   ibpm=51;
114   pcapm=37;
115   pcbpm=13;
116 }
117
```

```
118 if (PMB>=38 && PMB<=55){
119   iapm=150;
120   ibpm=101;
121   pcapm=55;
122   pcbpm=38;
123 }
124 if (PMB>=56 && PMB<=150){
125   iapm=200;
126   ibpm=151;
127   pcapm=150;
128   pcbpm=56;
129 }
130 if (PMB>=151 && PMB<=250){
131   iapm=300;
132   ibpm=201;
133   pcapm=250;
134   pcbpm=151;
135 }
136 if (PMB>=251 && PMB<=500){
137   iapm=500;
138   ibpm=301;
139   pcapm=500;
140   pcbpm=251;
141 }
142
143 //------ CO
144 if (C>=0 && C<=5094){
145   iac=50;
146   ibc=0;
147   pcac=5094;
148   pcbc=0;
149 }
150
151 if (C>=5095 && C<=10819){
152   iac=100;
153   ibc=51;
154   pcac=10819;
155   pcbc=5095;
156 }
157
158 if (C>=10820 && C<=14254){
159   iac=150;
160   ibc=101;
161   pcac=14254;
162   pcbc=10820;
163 }
164 if (C>=14255 && C<=17688){
165   iac=200;
166   ibc=151;
```

```
    pcac=17688;
    pcbc=14255;
}
if (C>=17689 && C<=34862){
    iac=300;
    ibc=201;
    pcac=34682;
    pcbc=17689;
}
if (C>=34863 && C<=57703){
    iac=500;
    ibc=301;
    pcac=57703;
    pcbc=34863;
}
//---NO2

if (N>=0 && N<=100){
    ian=50;
    ibn=0;
    pcan=100;
    pcbn=0;
}

if (N>=101 && N<=189){
    ian=100;
    ibn=51;
    pcan=189;
    pcbn=101;
}

if (N>=190 && N<=677){
    ian=150;
    ibn=101;
    pcan=677;
    pcbn=190;
}
if (N>=678 && N<=1221){
    ian=200;
    ibn=151;
    pcan=1221;
    pcbn=678;
}
if (N>=1222 && N<=2349){
    ian=300;
    ibn=201;
    pcan=2349;
    pcbn=1222;
}
```

```
if (N>=2350 && N<=3853){
  ian=500;
  ibn=301;
  pcan=3853;
  pcbn=2350;
}


//--- ICA
//PM_10
icapm1 = (((iapm1-ibpm1)/(pcapm1-pcbpm1))*(PMA-pcbpm1))+ibpm1;
// PM_2.5
icapm = (((iapm-ibpm)/(pcapm-pcbpm))*(PMB-pcbpm))+ibpm;
// --- C
icac = (((iac-ibc)/(pcac-pcbc))*(C-pcbc))+ibc;
//___NO2
ican = (((ian-ibn)/(pcan-pcbn))*(N-pcbn))+ibn;

// _Mayor valor de los tres ICA

mayorICA= Math.max(icapm1,icapm,icac,ican);
mayorICA=mayorICA.toFixed(2);

  return {
    field1: pre,
    field2: temp,
    field3: hum,
    field4: PMA,
    field5: PMB,
    field6: C,
    field7: N,
    field8: mayorICA,

  };
}
```

## 11.5 Código gráficas de barras del ICA de CO en MATLAB

```

% Template MATLAB code for visualizing data from a channel as a 2D line
% plot using PLOT function.

% Prior to running this MATLAB code template, assign the channel variables.
% Set 'readChannelID' to the channel ID of the channel to read from.
% Also, assign the read field ID to 'fieldID1'.

%TODO - Replace the [] with channel ID to read data from:
readChannelID = XXXXXXXX;
%TODO - Replace the [] with the Field ID to read data from:
```

```
fieldID1 = 6;

% Channel Read API Key
% If your channel is private, then enter the read API
% Key between the '' below:
readAPIKey = 'XXXXXXXXXXXXXX';

%%Read Data %%

[data, time] = thingSpeakRead(readChannelID, 'Field', fieldID1, 'NumPoints', 5, '
    ReadKey', readAPIKey);
%%time=time-hours(5);
for i=1:length(data)

    if (data(i) >= 0) && (data(i) <= 5094)

    iapm=50;
    ibpm=0;
    pcapm=5094;
    pcbpm=0;
    a=0;
    end

     if (data(i) >= 5095) && (data(i) <= 10819)
    iapm=100;
    ibpm=51;
    pcapm=10819;
    pcbpm=5095;
    a=1;
     end

     if (data(i) >= 10820) && (data(i) <= 14254)

    iapm=150;
    ibpm=101;
    pcapm=14254;
    pcbpm=10820;
    a=2;
     end

     if (data(i) >= 14255) && (data(i) <= 17688)
    iapm=200;
    ibpm=151;
    pcapm=17688;
    pcbpm=14255;
    a=3;
     end

      if (data(i) >= 17689) && (data(i) <= 34862)
```

```
60
61      iapm=300;
62      ibpm=201;
63      pcapm=34862;
64      pcbpm=17689;
65      a=4;
66       end
67
68       if (data(i) >= 34863) && (data(i) <= 57703)
69      iapm=500;
70      ibpm=301;
71      pcapm=57703;
72      pcbpm=34863;
73      a=5;
74       end
75
76
77       %%co(i)= ((data(i)-ibpm)*((pcapm-pcbpm)/(iapm-ibpm)))+pcbpm;
78       %%co(i)=round(co(i),2);
79       icac(i) = (((iapm-ibpm)/(pcapm-pcbpm))*(data(i)-pcbpm))+ibpm;
80       icac(i)=round(icac(i),2);
81
82
83
84
85 end
86
87 b=bar(time,icac,'FaceColor','flat')
88 refreshdata
89
90    for i=1:length(icac)
91
92        if (icac(i) >= 0) && (icac(i) <= 50.99)
93         b.CData (i, :) = [0 1 0];
94
95        end
96
97       if (icac(i) >= 51) && (icac(i) <= 100.99)
98          b.CData (i, :) = [1 1 0];
99       end
100
101      if (icac(i) >= 101) && (icac(i) <= 150.99)
102    b.CData (i, :) = [0.9290 0.6940 0.1250];
103      end
104
105      if (icac(i) >= 151) && (icac(i) <= 200.99)
106     b.CData (i, :) = [1 0 0];
107      end
108
```

```
      if (icac(i) >= 201) && (icac(i) <= 300.99)
     b.CData (i, :) = [0.4940 0.1840 0.5560];
     end

     if (icac(i) >= 301) && (icac(i) <= 500)
        b.CData (i, :) = [0.6350 0.0780 0.1840];
     end


   end
    hold on
    grid on
   xtips1 = b.XEndPoints;
   ytips1 = b.YEndPoints;
   labels1 = string(b.YData);
    text(xtips1,ytips1,labels1,'HorizontalAlignment','center','VerticalAlignment',
    'bottom','FontSize',5)


%%Visualize Data %%

%%plot(time, pm2_5,'-p','MarkerFaceColor','red','MarkerSize',10);
%plot(time, co,'-*','MarkerSize',10);
xlabel('Hora');ylabel('CO/[ug/m^3]');

grid on
```

### 11.6 Código para crear alerta correo electrónico ThingSpeak en MATLAB

```
% Read the soil moisture channel data from the past two weeks.
% Send an email and tell the user to add water if the value
% is in the lowest 10 %.

% Store the channel ID for the moisture sensor channel.
channelID = XXXXXXX;
fieldID1=8;

% Provide the ThingSpeak alerts API key.  All alerts API keys start with TAK.
alertApiKey = 'XXXXXXXXXXXXXXXXX';
readAPIKey='XXXXXXXXXXXXXX';

% Set the address for the HTTTP call
alertUrl="https://api.thingspeak.com/alerts/send";

% webwrite uses weboptions to add required headers.  Alerts needs a ThingSpeak-
    Alerts-API-Key header.
options = weboptions("HeaderFields", ["ThingSpeak-Alerts-API-Key", alertApiKey ]);


```

```
20
21 %Read the recent data.
22 [data, time] = thingSpeakRead(channelID, 'Field', fieldID1, 'NumPoints', 1, '
       ReadKey', readAPIKey);
23 %Check to make sure the data was read correctly from the channel.
24 if isempty(data)
25     alertBody = 'No hay datos de ICA';
26 else
27     %Calculate a 10% threshold value based on recent data.
28     %Set the outgoing message
29     if (data>100)&&(data<=150)
30         alertBody = 'Ojo niveles de ICA mayores a 100 en Nodo 1_Alerta Naranja';
31         %Set the email subject.
32     alertSubject = sprintf("Alerta Naranja Nodo 1");
33     end
34     if (data>150)
35         alertBody = 'Ojo niveles de ICA mayores a 150 en Nodo 1_ Alerta Roja';
36         %Set the email subject.
37     alertSubject = sprintf("Alerta Roja Nodo 1");
38     end
39 end
40
41  %Catch errors so the MATLAB code does not disable a TimeControl if it fails
42 try
43     webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);
44 catch someException
45     fprintf("Failed to send alert: %s\n", someException.message);
46 end
```