UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Desarrollo de un sistema para la monitorización de micropartículas en el aire usando drones

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Sánchez Sánchez, José

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

CURSO ACADÉMICO: 2023/2024

AGRADECIMIENTOS

"A mi mujer, Michaila, por su apoyo y amor incondicional siempre. No habría llegado hasta aquí sin ella."

“A Carlos Tavares Calafate, por todo su apoyo como mi mentor en este proyecto. Ha sido una experiencia muy gratificante.”

"A todas las personas que me han apoyado para llegar hasta aquí, sabéis quiénes sois."

RESUMEN

El presente Trabajo de Fin de Grado muestra el desarrollo de un sistema dedicado a la monitorización de micropartículas en el aire usando drones. Para ello se ha utilizado una interfaz basada en Arduino, que en conjunto con el sistema propio del drone, puede realizar durante su pilotaje los siguientes procesos:

- Recogida de datos de micropartículas en el aire.
- Obtención de coordenadas GPS.

Al ser ambas medidas tomadas de forma simultánea el sistema provee datos a tiempo real sobre la concentración de micropartículas en el espacio. Estos datos pueden entonces ser enviados a una interfaz para su posterior interpretación mediante un módulo NodeMCU.

Un componente fundamental de este proyecto es el uso del protocolo MAVLink para la comunicación entre la controladora de vuelo Pixhawk utilizada y la placa Arduino. Así como el uso del sensor de partículas Sensirion SPS30.

En este trabajo se muestra un ejemplo de aplicación donde se reflejan los datos obtenidos mediante mapas de calor. Se muestran también otras posibles aplicaciones del proyecto, así como las posibilidades de expansión que este presenta.

Palabras Clave: Micropartículas, Drone, Pixhawk, Arduino, NodeMCU, MAVLink.

RESUM

El present Treball de Fi de Grau mostra el desenvolupament d'un sistema dedicat al monitoratge de micropartícules en l'aire usant drons. Per a això s'ha utilitzat una interfície basada en Arduino, que en conjunt amb el sistema propi del dron, pot realitzar durant el seu pilotatge els següents processos:

- Recollida de dades de micropartícules en l'aire.
- Obtenció de coordenades GPS.

A l'ésser totes dues mesures preses de forma simultanieja el sistema proveeix dades a temps real sobre la concentració de micropartícules en l'espai. Estes dades poden llavors ser enviats a una interfície per a la seua posterior interpretació mitjançant un mòdul NodeMCU.

Un component fonamental d'este projecte és l'ús del protocol MAVLink per a la comunicació entre la controladora de vol Pixhawk utilitzada i la placa Arduino. Així com l'ús del sensor de partícules Sensirion SPS30.

En este treball es mostra un exemple d'aplicació on es reflecteixen les dades obtingudes mitjançant mapes de calor. Es mostren també altres possibles aplicacions del projecte, així com les possibilitats d'expansió que este presenta.

Paraules Clau: Micropartícules, Dron, Pixhawk, Arduino, NodeMCU, MAVLink.

ABSTRACT

This Final Degree Project shows the development of a system dedicated to the monitoring of microparticles in the air using drones. For this purpose, an Arduino-based interface has been used, which in conjunction with the drone's own system, can perform the following processes during its piloting:

- Data collection of microparticles in the air.

- Obtain GPS coordinates.

As both measurements are taken simultaneously, the system provides real-time data on the concentration of microparticles in space. These data can then be sent to an interface for further interpretation using a NodeMCU module.

A key component of this project is the use of the MAVLink protocol for communication between the Pixhawk flight controller used and the Arduino board. As well as the use of the Sensirion SPS30 particle sensor.

This work shows an example of an application where the data obtained is reflected through heat maps. Other possible applications of the project are also shown, as well as the possibilities of expansion that it presents.

Keywords: Microparticles, Drone, Pixhawk, Arduino, NodeMCU, MAVLink.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

En este primer capítulo de la memoria se introduce el trabajo de fin de grado, la motivación tras este y la metodología utilizada. Además, se provee una visión general de los contenidos de la memoria por capítulos.

1.1. ESTRUCTURA DEL DOCUMENTO

La memoria de este trabajo se divide en seis capítulos, siendo este el primero de ellos. En este apartado se describe brevemente el propósito de cada capítulo y la estructura general de la memoria.

En primer lugar, en el capítulo 2 se habla sobre el estado del arte, o en otras palabras el estado actual de las tecnologías que en los capítulos posteriores se utilizarán. En este se hacen referencias a diferentes artículos científicos para apoyar la teoría y presentar el contexto actual y el lugar donde la solución propuesta encaja y las necesidades que cubre.

El capítulo 3 trata sobre el hardware y software utilizados en el proyecto. En este se proveen las instrucciones completas de este montaje, así como una descripción de los diferentes componentes y sus conexiones entre sí. Tras el hardware se describen los diferentes programas utilizados en este proyecto, así como algunas configuraciones necesarias para poder operar con éxito.

El capítulo 4 se enfoca en la solución propuesta. En este se hace un breve recorrido sobre el proceso que se utilizó para llegar a la solución, incluyendo fallos y aprendizajes hasta alcanzar esta. Se muestra cómo funciona el ciclo de la solución y se explican las principales partes del código para dar un entendimiento detallado de estas.

A continuación, en el capítulo 5 se explica el procedimiento seguido para la toma de datos que se usarán para los experimentos y análisis. Se muestran 3 formas diferentes de interpretar los datos de micropartículas, en función del espacio, tiempo y altura. En cada uno de estos se sacan conclusiones sobre los resultados, y se busca su significado basándose en la teoría y en la exploración real del terreno analizado

Finalmente, el capítulo 6 contiene las conclusiones del trabajo, se habla de las limitaciones de este y se proponen algunos trabajos futuros usando como base lo que ya se ha construido en este. Al final de la memoria se incluyen las referencias, dos anexos (códigos y presupuesto) y los índices de tablas y figuras.

1.2. MOTIVACIÓN DEL TRABAJO

Con este trabajo se busca desarrollar un sistema para la monitorización de partículas basada en drones de bajo coste y totalmente abierto para la comunidad científica y universitaria.

Se ha buscado crear un sistema lo más modular posible, de modo que sea posible expandir el sistema y añadir otros componentes y funcionalidades al código, adaptándose al usuario y sus necesidades. El tamaño del sistema, su precio y su compatibilidad lo hacen una opción excelente para la mayoría de los usuarios. Es un prototipo funcional listo para usar en la industria.

Además, se ha enfocado este trabajo a la medición de micropartículas ya que es un área tremendamente importante, especialmente para la salud de las personas. La comunidad científica y tecnológica ha estado dirigiendo más la mirada a este tema en los últimos años y con este trabajo se espera realizar una aportación, al igual que traer consciencia sobre un problema que las personas enfrentan a diario, pero del que la mayoría no son conscientes.

1.3. OBJETIVO DEL TRABAJO

El objetivo de este trabajo es diseñar un sistema para la monitorización de micropartículas en el aire utilizando drones. Para esto se utilizará un sistema basado en Arduino, capaz de conectarse y comunicarse con el dron para recibir datos de coordenadas de este, y de realizar mediciones de partículas mediante un sensor dedicado a ello. El sistema recogerá estos datos y los pondrá juntos para su posterior interpretación.

Se busca crear un sistema autónomo que pueda ser utilizado para realizar diferentes tipos de mediciones de micropartículas y provea un paquete de datos listo para su análisis. Este sistema también podrá servir como base para proyectos más complejos y será accesible para la comunidad en general al usar solo hardware y software de código abierto.

Además, se realizarán diferentes estudios y análisis para comprobar la eficacia del sistema.

1.4. METODOLOGÍA EMPLEADA

Para la realización de este trabajo se ha realizado una implementación real de un prototipo basado en Arduino capaz de cumplir con las funciones que se necesitan, en este caso la obtención de coordenadas mediante la electrónica del dron y la medición de partículas mediante un sensor. Además, se ha añadido un módulo adicional que permite expandir este trabajo en diferentes direcciones.

El primer paso en este trabajo ha sido crear un prototipo funcional y su código correspondiente para obtener los datos que se buscan. La primera etapa fue hacer funcionar los componentes por separado, para posteriormente unirlos todos en un sistema funcional.

Una vez realizado el montaje, y para comprobar su eficacia se realizaron mediciones reales en un área seleccionada para su estudio. Los datos obtenidos con estas fueron posteriormente procesados de diferentes formas. Una vez hecho esto se compararon los datos con la realidad para comprobar si reflejaban esta y se encontró que definitivamente este era el caso. Además, estos aportaron información muy valiosa e interesante, confirmando que el proyecto es extremadamente útil y está perfectamente equipado para cumplir los objetivos propuestos.

Se discuten también las limitaciones actuales, sugerencias de mejora para estas y se proponen trabajos futuros.

CAPÍTULO 2. ESTADO DEL ARTE

En este capítulo se mostrará el estado actual de la tecnología y el uso de drones en la industria, así como el estado de la monitorización de las micropartículas suspendidas en el aire (PM). Además, se pondrá en contexto donde encaja el proyecto realizado en el ámbito industrial y con las demás tecnologías actuales.

2.1. MONITORIZACIÓN DE PARTÍCULAS

En primer lugar, se empezará hablando de la importancia de la monitorización de las micropartículas en el aire debido a su impacto negativo sobre la salud de las personas. En este documento, a partir de ahora se referirá a las partículas con las siglas PM (Particulate Matter), seguido de un número, el cual índica el tamaño (diámetro) en micras de ese tipo de partículas. La figura 2.1 muestra una referencia del tamaño de dos de las partículas que más adelante se analizarán, las PM2.5 y las PM10.0.

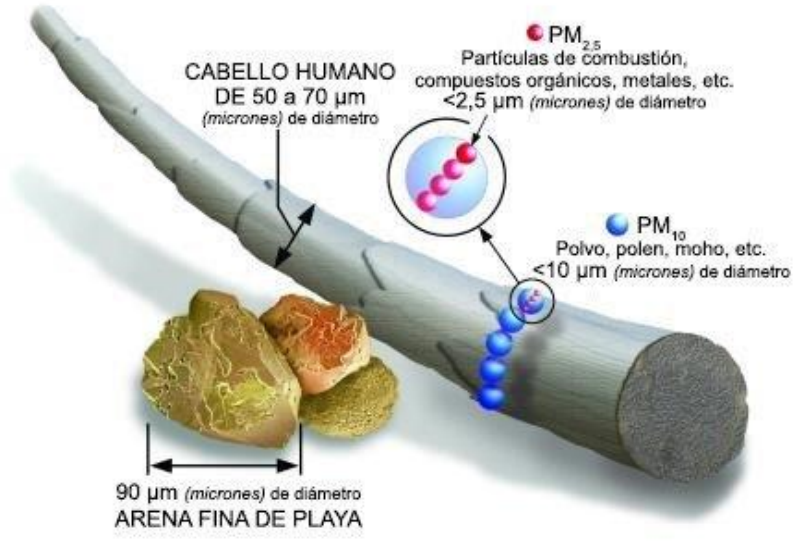

Figura 2.1: Tamaño de partículas

Como se puede observar estas partículas son minúsculas y están compuestas por varios componentes químicos como pueden ser sólidos aerosoles, pequeñas gotas de líquido o sólidos. Su reducido tamaño es lo que les permite viajar directamente hacia los alveolos pulmonares, pudiendo provocar problemas como infartos, embolias, e incluso cáncer.

La correlación entre la concentración de partículas y algunas enfermedades es conocida desde hace mucho tiempo, aunque recientemente se ha puesto mucho más enfoque en el estudio de estas. Los estudios más antiguos muestran las partículas PM10.0 como partículas finas, aunque en estudios recientes este título se lo llevan las partículas PM2.5 (Englert, 2004).

Estudios recientes muestran también posibles asociaciones entre la exposición a las partículas PM2.5 y el desarrollo de la enfermedad de Alzheimer u otras enfermedades neurológicas (Shou et al., 2019). Debido a su tamaño estas partículas, pueden encontrar fácilmente una entrada al cuerpo y causar diversos problemas de salud, llegando incluso a afectar los intestinos o al propio ADN de la persona.

Debido a esto, en las últimas décadas, y coincidiendo con el crecimiento rápido de la industria se ha puesto más atención al problema de la contaminación por micropartículas y sus soluciones posibles. Se pueden encontrar gran variedad de estudios sobre la población y su salud, revelando la gran importancia que una buena calidad de aire tiene para el ser humano, y el detrimento que supone cuando este no es el caso, y se está respirando contaminación de forma constante (Ortiz, 2022).

Incluso con todos estos estudios, todavía hay áreas que se desconocen, y no se puede confirmar el rango especifico de partículas que causan estos problemas (Kappos et al., 2004). Algunas hipótesis barajadas tratan de explicar cómo las diferentes partículas con diferentes orígenes y componentes podrían cargar las sustancias o elementos responsables por causar las enfermedades. Hasta aquí se puede observar la gran importancia de la monitorización de partículas en el aire. A continuación, se hablará de la situación respecto a la industria.

En el ámbito de la industria se pueden encontrar muchos estudios donde el objetivo es identificar las fuentes (o sea, qué elementos) son los mayores contribuyentes a esta contaminación, poniendo especial atención a la concentración de partículas PM2.5 debido a sus efectos nocivos sobre la salud (Wang et al., 2020) (Luo et al., 2018) (Park et al., 2001) (Giones & Brem, 2017).

2.2. USO DE DRONES EN LA INDUSTRIA

En los últimos años los vehículos aéreos no tripulados (UAVs), coloquialmente llamados drones han ido en auge de forma increíblemente rápida, por lo que es sorprendente escuchar que el desarrollo de esta tecnología haya echo emanar un nuevo mercado de nuevas industrias, y que la demanda se haya disparado debido a todo lo que un dron puede proveer al usuario, ya sea a nivel personal o profesional (Nwaogu et al., 2023).

El uso de drones ha encontrado su lugar en prácticamente todos los rincones de la industria ya que permite facilitar muchísimo las diferentes tareas y operaciones y en muchos casos eliminar la necesidad de tener a una persona realizando la actividad (Moreno-Jacobo et al., 2021). Esto es sobre todo útil en situaciones con más riesgo o peligro para los usuarios. Las aplicaciones de drones van desde la monitorización de cultivos, la inspección de cableado (figura 2.2), construcción, arquitectura, toma de fotos y mucho más.

Figura 2.2: Operarios revisando cableado con drone.

En el caso concreto de monitorización de micropartículas, autores como Marinov et al. (2019) proponen el uso vehículos aéreos no tripulados y sensores de calidad del aire económicos para estudiar la calidad del aire a diferentes altitudes, y siendo accesible a una amplia gama de usuarios. En la misma línea, Li et al. (2018) han equipado un UAV con sensores miniaturizados para investigar los patrones de distribución vertical y las fuentes de partículas finas de aerosol (PM2.5) dentro de los 1.000 m inferiores de la troposfera. Realizaron un total de 16 vuelos de vehículos aéreos no tripulados en la región del delta del río Yangtze (YRD), China, entre el verano y el invierno de 2014. Descubrieron que en verano y el otoño las partículas provinieron principalmente de emisiones de fuentes locales de la región, pero que en invierno procedían principalmente de fuentes de transporte de larga distancia desde el norte y noroeste de China debido al impacto del monzón invernal asiático.

Estos y muchos otros trabajos evidencian el gran interés en monitorizar las micropartículas presentes en nuestro entorno, así como las ventajas de usar UAVs para llevar a cabo dicho proceso.

2.3. CONEXTO DEL PROYECTO

En la línea de trabajos anteriores, este proyecto provee una solución y herramienta muy eficaz para la industria. Combinar la monitorización de partículas con drones tiene innumerables ventajas frente a medir desde estaciones fijas ya que se pueden obtener datos a diferentes alturas y acceder a lugares difíciles a los que de normal no se podría. Por ello se va a proceder al diseño de una solución abierta, usando componentes estándar de bajo coste, que permite alcanzar el propósito que se busca.

CAPÍTULO 3: SISTEMA PROPUESTO

En este capítulo se muestran todos los componentes del sistema propuesto, tanto de hardware como de software y su función. Se proveen también instrucciones para la conexión correcta de los componentes, así como la configuración de software necesaria. Finalmente se muestra la arquitectura general del sistema

3.1. HARDWARE EMPLEADO Y CONEXIONES

3.1.1. Arduino Uno Rev3

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar para cualquier usuario. Las placas Arduino son circuitos impresos con un microcontrolador integrado que puede ser programado por el usuario para realizar infinidad de funciones. Para interactuar con el exterior u otros componentes, las placas Arduino cuentan con pines digitales y analógicos que pueden ser programados como entradas o salidas.

Figura 3.1: Arduino Uno Rev3 Original.

En el presente montaje se ha utilizado el modelo Uno Rev3, entre otras placas Arduino, debido a su pequeño tamaño, lo cual lo hace perfecto para ahorrar espacio en nuestro dron, y a su versatilidad. El modelo Uno Rev3, mostrado en la figura 3.1 dispone de capacidad y memoria suficiente para realizar las operaciones necesarias en este proyecto. El Arduino se encarga de recibir y procesar los datos tanto de la controladora de vuelo del dron como del sensor de partículas a bordo, empaquetando ambos y enviándolos para su posterior procesamiento.

3.1.2. Arduino Proto Shield Rev3

Las placas Arduino Shield, también llamadas escudos, son extensiones de hardware que se colocan sobre la placa Arduino base usada en el proyecto con el objetivo de ampliar sus capacidades, o facilitar el montaje y conexión de los diferentes elementos. En este proyecto será utilizada para montar el módulo NodeMCU de cara a posibilitar la alimentación de los diferentes componentes, así como facilitar su fácil conexión. El uso del escudo es opcional, y su configuración se deja en manos del usuario, debido a que debe ser adaptado a las necesidades de cada proyecto o dron utilizado. La figura 3.2 muestra el escudo utilizado en el trabajo.

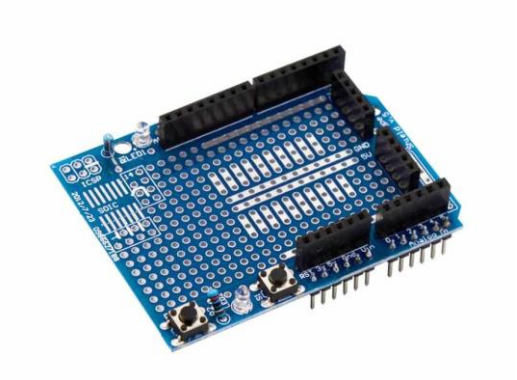

Figura 3.2: Arduino Proto Shield Rev3.

3.1.3. Dron

Para este proyecto se ha utilizado el dron mostrado en la figura 3.3, el cual ha sido proporcionado por el Departamento de Informática de Sistemas y Computadores (DISCA) de la Universidad Politécnica de Valencia (UPV). El diseño o componentes del dron más allá de la controladora de vuelo quedan fuera del alcance de este trabajo, ya que la solución recogida en este documento puede ser utilizada con cualquier drone que utilice una controladora Pixhawk conectada a un GPS.

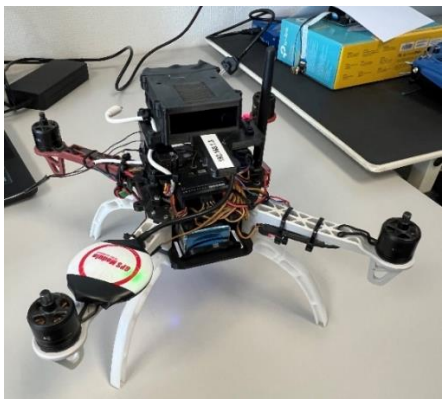

Figura 3.3: Dron utilizado.

3.1.4. Controladora de vuelo Pixhawk

La controladora de vuelo es el componente central de cualquier dron, y actúa como el cerebro de este; mediante su procesador se encarga de que todo el conjunto funcione correctamente. Dispone de diferentes conexiones de entrada o salida donde se conectan los diferentes componentes del dron, como pueden ser el GPS, reguladores de velocidad de los motores, etc.

Pixhawk es un proyecto independiente de hardware libre que provee un autopiloto de buen rendimiento a bajo coste. Esto lo hace una referencia en los sectores académicos, de entretenimiento o incluso industriales, donde es ampliamente utilizado. El autopiloto provee algoritmos de guía, navegación y control para distintos tipos de drones. Las controladoras Pixhawk utilizan el software Dronecode, que controla y conecta la controladora con sensores, telemetría y otros periféricos.

Figura 3.4: Controladora de vuelo Pixhawk.

La controladora Pixhawk, mostrada en la figura 3.4, se conecta a la placa Arduino mediante el puerto de telemetría denominado como “TELEM 2” como se muestra en la Figura 3.5. Para una transmisión de datos estable es importante que ambas placas compartan una conexión a masa común. La comunicación entre la Pixhawk y el Arduino se realizará mediante una conexión serial, utilizando el protocolo de comunicación MAVLink.

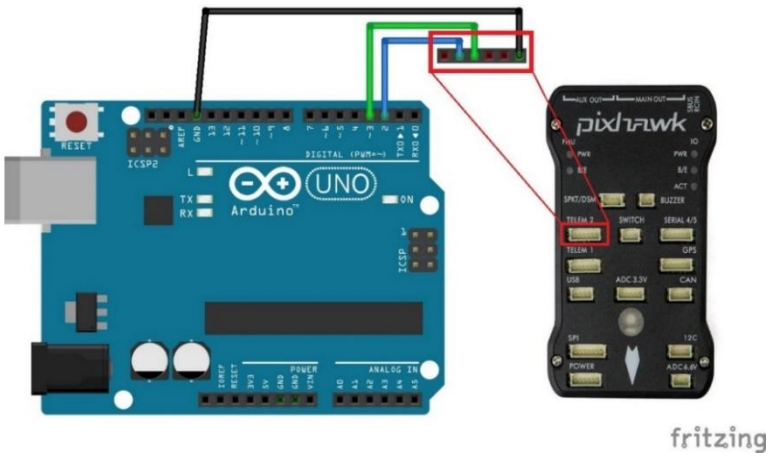

Figura 3.5: Conexión Arduino - Pixhawk.

3.1.5. NodeMCU ESP8266 MOD

NodeMCU es otra plataforma de código abierto de bajo coste. En este proyecto se ha utilizado la placa de desarrollo NodeMCU ESP8266 MOD, mostrada en la figura 3.6 basada en el chip ESP8266. La versión MOD de esta placa la modificó la comunidad, con las mismas características que su versión normal, con la peculiaridad de que es más pequeña y puede encajar en placas de pruebas u otros componentes.

Este módulo es muy popular en proyectos relacionados con el "Internet de las cosas" gracias a su capacidad Wifi. En este proyecto su objetivo es recoger los datos para su posterior procesado, funcionando de forma interdependiente y paralela a la placa Arduino.

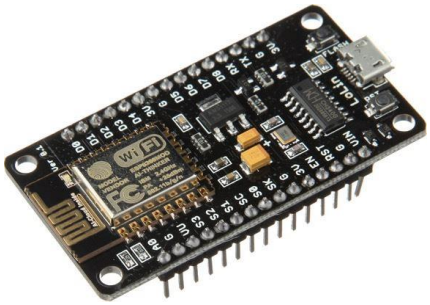

Figura 3.6: Placa de desarrollo NodeMCU.

La conexión entre la placa Arduino y el módulo NodeMCU se realiza mediante una conexión serial, tal y como se muestra en la Figura 3.7. El NodeMCU se alimenta a través de la placa Arduino a 5V. El módulo puede operar correctamente a solo 3.3V de alimentación, pero para asegurar una buena operación, especialmente con programas más complejos siendo ejecutados, usamos 5V.

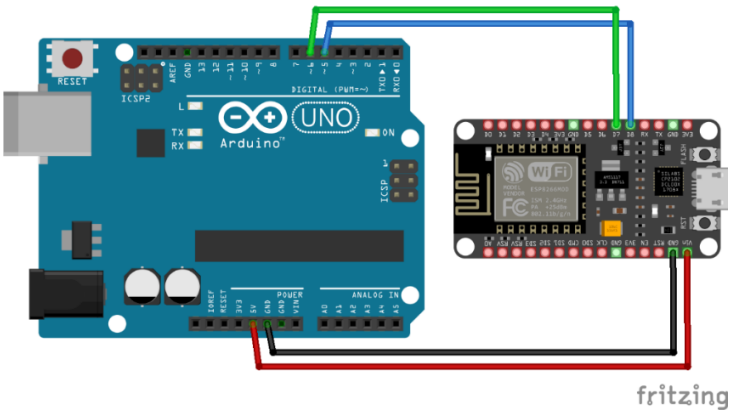

Figura 3.7: Conexión Arduino - NodeMCU.

3.1.6. Sensor de micropartículas Sensirion SPS30

El sensor de medición de micropartículas SPS30, mostrado en la Figura 3.8, de la compañía SENSIRION es otro de los componentes fundamentales de este proyecto. Se trata de un sensor óptico de buena calidad y muy compacto (41mm x 41mm x 12mm), lo cual lo hace perfecto para el uso en este trabajo y su montaje en drones.

Figura 3.8: Sensor Sensirion SPS30.

El sensor provee diferentes mediciones de micropartículas de diferentes tamaños, tanto en concentración en masa como en concentración en número. Estas serán el objetivo de estudio de este proyecto. Algunas de las especificaciones de este sensor, como los rangos de medición, unidades, etc., están recogidas en la tabla 3.1.

Tabla 3.1: Especificaciones del sensor.

Parameter	Conditions	Value	Units
Mass concentration range	-	0 to 1'000	µg/m³
Mass concentration size range	PM1.0	0.3 to 1.0	µm
	PM2.5	0.3 to 2.5	µm
	PM4	0.3 to 4.0	µm
	PM10	0.3 to 10.0	µm
Mass concentration precision[1,2] for PM1 and PM2.5[3]	0 to 100 µg/m³	±10	µg/m³
	100 to 1000 µg/m³	±10	% m.v.
Mass concentration precision[1,2] for PM4, PM10[4]	0 to 100 µg/m³	±25	µg/m³
	100 to 1000 µg/m³	±25	% m.v.
Maximum long-term mass concentration precision limit drift	0 to 100 µg/m³	±1.25	µg/m³ / year
	100 to 1000 µg/m³	±1.25	% m.v. / year
Number concentration range	-	0 to 3'000	#/cm³
Number concentration size range	PM0.5	0.3 to 0.5	µm
	PM1.0	0.3 to 1.0	µm
	PM2.5	0.3 to 2.5	µm
	PM4	0.3 to 4.0	µm
	PM10	0.3 to 10.0	µm
Number concentration precision[1,2] for PM0.5, PM1 and PM2.5[3]	0 to 1000 #/cm³	±100	#/cm³
	1000 to 3000 #/cm³	±10	% m.v.
Number concentration precision[1,2] for PM4, PM10[4]	0 to 1000 #/cm³	±250	#/cm³
	1000 to 3000 #/cm³	±25	% m.v.
Maximum long-term number concentration precision limit drift[2]	0 to 1000 #/cm³	±12.5	#/cm³ / year
	1000 to 3000 #/cm³	±1.25	% m.v. / year
Sampling interval	-	1±0.04	s

El sensor SPS30 ofrece interfaces UART e I2C, dependiendo de cómo sean conectados los pines. En este caso usaremos la interfaz I2C para el envío de datos a la placa Arduino. La numeración de pines y sus funcionalidades pueden ser observadas en la figura 3.9 y en la tabla 3.2. El conector usado por el sensor es un conector hembra ZHR-5. Normalmente este viene incluido con la compra del sensor.

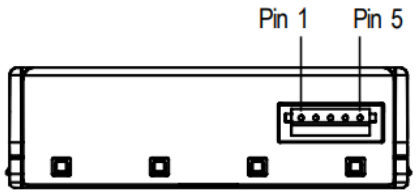

Figura 3.9: Conector SPS30.

Tabla 3.2: Asignación de pines de SPS30.

Pin	Name	Description	Comments
1	VDD	Supply voltage	5V ± 10%
2	RX	UART: Receiving pin for communication	TTL 5V and LVTTL 3.3V compatible
	SDA	I²C: Serial data input / output	
3	TX	UART: Transmitting pin for communication	TTL 5V and LVTTL 3.3V compatible
	SCL	I²C: Serial clock input	
4	SEL	Interface select	Leave floating to select UART
			Pull to GND to select I²C
5	GND	Ground	Housing on GND

La conexión con la placa Arduino se realiza tal y como se muestra en la Figura 3.10. La alimentación es a 5V, y el pin 4 se conecta a GND para seleccionar la interfaz I2C. Los pines SDA y SCL del sensor se conectan respectivamente a los pines A4 y A5 del Arduino (que equivalen a SDA y SDL en este).

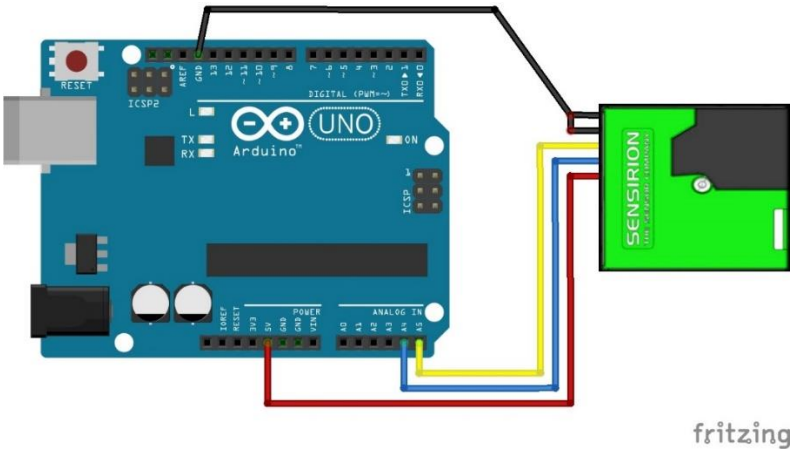

Figura 3.10: Conexión Arduino - SPS30.

3.1.8. BEC

Para alimentar la placa de Arduino directamente desde la batería del drone se utilizará un circuito BEC (circuito de eliminación de batería), mostrado en la figura 3.11 para adaptar la corriente y voltaje a uno que pueda ser utilizable por la electrónica. Se conectará la entrada a los terminales de la batería del drone y la salida a los pines “Vin” y “GND” de la placa Arduino, honrando la polaridad de las conexiones.

Figura 3.11: BEC.

3.2. SOFTWARE EMPLEADO

3.2.1 Arduino IDE

El entorno de desarrollo integrado (IDE) de Arduino, mostrado en la figura 3.12, es la aplicación utilizada para escribir el código y programar de forma muy sencilla la placa Arduino. Este cuenta con un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, así como las herramientas necesarias para cargar el programa en la memoria flash del hardware. El lenguaje de programación utilizado en Arduino IDE es C++.

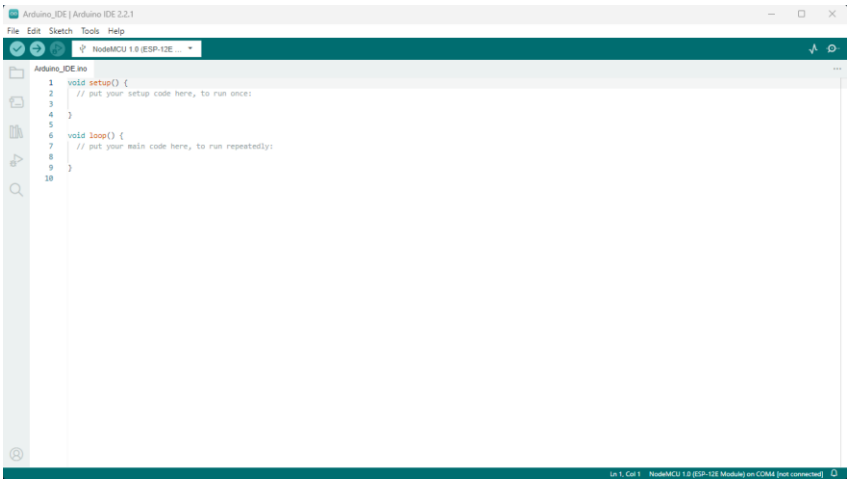

Figura 3.12: Interfaz de Arduino IDE.

3.2.1.1 Librerías de Arduino necesarias

Las librerías en Arduino son colecciones de código que facilitan el uso de determinados elementos, o proporcionan funcionalidades específicas. Las librerías pueden ser instaladas directamente desde el IDE de Arduino o mediante su descarga desde un repertorio de código. Para hacer posible el funcionamiento de este proyecto ha sido necesario el uso de las siguientes librerías:

- Librería “sps30.h”: contiene las funciones que permiten al usuario interactuar con el sensor de partículas SPS30, realizar mediciones, obtener los resultados, entre otros. Puede ser instalada desde el IDE
- Librería “SoftwareSerial.h”: permite establecer mediante software una comunicación serial (o más) utilizando cualquier par de pines digitales en la placa Arduino. Puede ser instalada desde el IDE.
- Librería “mavlink.h”: contiene las colecciones de código necesarias para utilizar de forma sencilla el protocolo de comunicación MAVLink. Esta librería se puede descargar desde un repositorio. Es crucial obtener la librería con la versión correcta de MAVLink (versión 1 o 2) dependiendo de que versión del protocolo se está utilizando. En el caso de este proyecto se utiliza la versión 1.

3.2.2. Mission Planner

Mission Planner es una estación de control terrestre, cuya interfaz se muestra en la figura 3.13. Esta aplicación puede ser usada para configurar y ajustar nuestro vehículo no tripulado, así como para programar otras funciones en este. Mission Planner permite crear y cargar misiones autónomas en nuestro dron, las cuales serán ejecutadas por el piloto automático. Con el hardware apropiado es posible monitorear el estado de la aeronave durante su operación, recibir información detallada del piloto automático o incluso manejar el drone desde tierra.

En este proyecto el uso de Mission Planner se limita a la configuración básica de la controladora de vuelo para su correcta comunicación con la placa de Arduino.

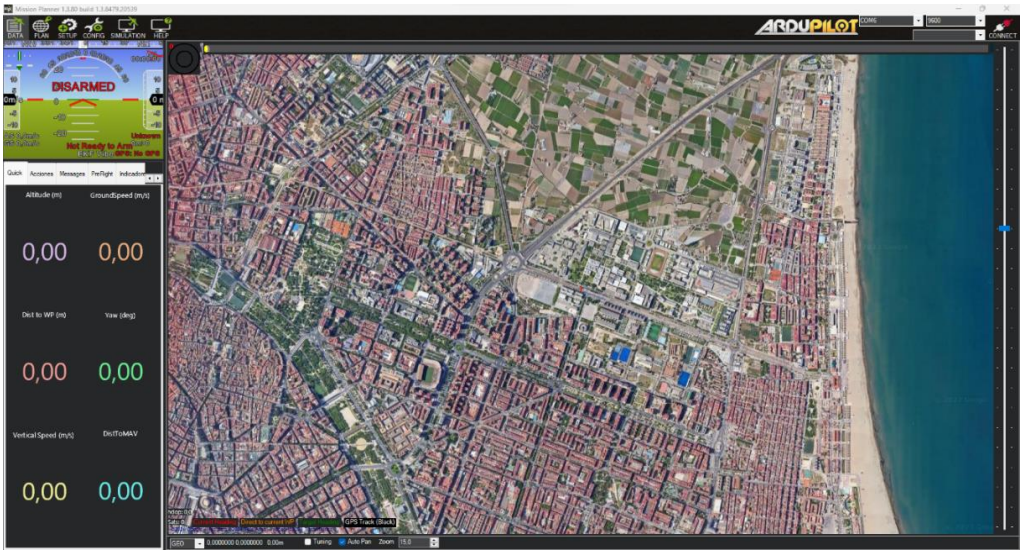

Figura 3.13: Interfaz de Mission Planner.

3.2.2.1. Configuración de Pixhawk mediante Mission Planner

En este apartado se muestran los pasos necesarios para controlar la controladora de vuelo a Mission Planner, y realizar la configuración necesaria para poder utilizar el protocolo MAVLink.

El primer paso es conectar la controladora Pixhawk al ordenador mediante conexión USB; a continuación, y si no se realiza de forma automática, seleccionar el puerto de conexión y la velocidad de transmisión en baudios. Finalmente pulsar conectar. Estos pasos se muestran en la figura 3.14.

Figura 3.14: Conexión a Pixhawk desde Mission Planner.

Una vez realizada la conexión de forma exitosa, el siguiente paso es configurar el protocolo MAVLink. En la pestaña “Config” se selecciona el apartado “Full Parameter Tree”, y dentro de este se deben configurar los valores del canal “Serial 2”. La figura 3.15 muestra estos pasos. Los parámetros que se ajustan en este apartado son la tasa de baudios y la versión de MAVLink utilizada. Para el correcto funcionamiento de la comunicación MAVLink es crucial que la velocidad de transmisión coincida con la usada en el código del proyecto para la comunicación serial MAVLink. Lo mismo sucede con la versión de MAVLink, la cual debe coincidir con la utilizada en nuestro código.

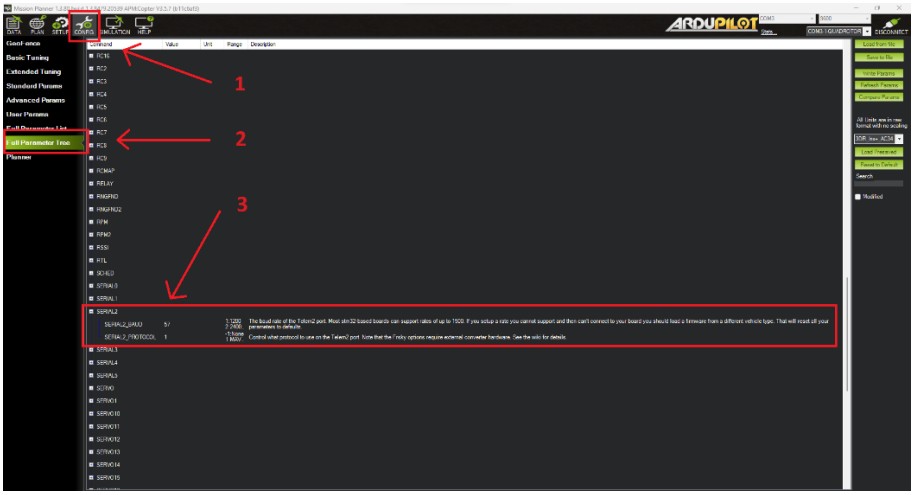

Figura 3.15: Configuración de Pixhawk.

3.2.3. MAVLink

MAVLink (MAVLink) es un protocolo de comunicación binario utilizado para establecer comunicación entre vehículos no tripulados y con sus estaciones de tierra. Una de las ventajas principales que presenta es ser un protocolo muy ligero, lo que resulta especialmente útil en sistemas con poco ancho de banda, o que no pueden manejar demasiados recursos.

El protocolo MAVLink permita hasta 255 sistemas estar conectados a la red, incluyendo vehículos, estaciones terrestres, etc. Además, proporciona métodos para detectar caída de paquetes, errores, comprobación de recepción y muchas otras funciones que lo hacen muy útil en ambientes con alta latencia.

El protocolo es muy eficiente ya que sus paquetes o mensajes tienen solo 8 bytes de sobrecarga (cabecera más checksum) en su versión MAVLink1, incluyendo la señal de inicio y la detección de caída de paquetes. La versión MAVLink 2 cuenta con 16 bytes de sobrecarga. La estructura de estos mensajes se puede observar en la figura 3.16 y en la tabla 3.3.

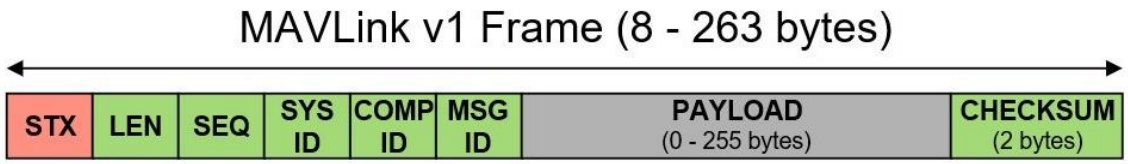

Figura 3.16: Formato de los paquetes de datos en MAVLink 1.

MAVLink dispone de dos formas de comunicación, publicación-suscripción y punto a punto, dependiendo de la información a comunicar. Los mensajes del protocolo se definen dentro de archivos XML. Para diferentes lenguajes de programación se crean bibliotecas especificas a partir de estos archivos.

Tabla 3.3: Paquete de datos MAVLink 1.

Byte	Descripción
STX	Indica el comienzo del paquete.
LEN	Byte de longitud. Indica el tamaño (número de bytes) del paquete.
SEQ	Número de secuencia. Contiene el número de paquete enviado, para detección de errores.
SYS ID	Identificación del sistema, para diferenciarlo de otros en la misma red.
COMP ID	Identificación del componente del sistema con el que se está comunicando.
MSG ID	Identificador del tipo de mensaje.
PAYLOAD	Paquete de datos, donde se encuentra la información útil.
CHECKSUM	Controla y registra los errores que hayan podido producirse en la transmisión.

3.2.4. MATLAB

MATLAB es una plataforma de cálculo numérico y programación utilizada por millones de profesionales. Ofrece un entorno de desarrollo integrado y cuenta con su propio lenguaje de programación, el lenguaje M. En este proyecto se utilizará MATLAB para analizar los datos de coordenadas y micropartículas obtenidos por el drone, representando estos de diferentes formas.

3.3 ARQUITETURA GENERAL Y PROTOTIPO

Una vez se han realizado todas las conexiones es importante conocer cómo funciona el sistema en su totalidad, y como fluye la información a través de este.

En el sistema propuesto la placa de Arduino es el corazón del sistema, y recibe los datos de micropartículas desde el sensor, y los datos de coordenadas (latitud, longitud y altura) desde la controladora de vuelo. Como se observará en el capítulo siguiente, algunos de estos datos no se reciben de forma directa, por lo que necesitan ser procesados para tenerlos en el formato necesario. La controladora de vuelo envía gran cantidad de mensajes mediante la comunicación MAVLink, por lo que es necesario filtrarlos para aislar aquellos que son útiles para el proyecto.

Una vez se han recibido y procesado ambos paquetes de datos, estos son enviados al módulo NodeMCU. En el actual trabajo este es el final del proceso, pero, como se detallará en el capítulo 6, este sistema está listo para ser expandido según necesite el usuario. Una vez los datos se reciben en el NodeMCU, estos pueden ser fácilmente mandados a un servidor web, donde pueden ser interpretados. Otra opción posible sería transmitir desde aquí los datos a una unidad de memoria que recoja todas las medidas, para su posterior interpretación. Este flujo de información, actual del proyecto y sugerido, se puede observar en la Figura 3.17.

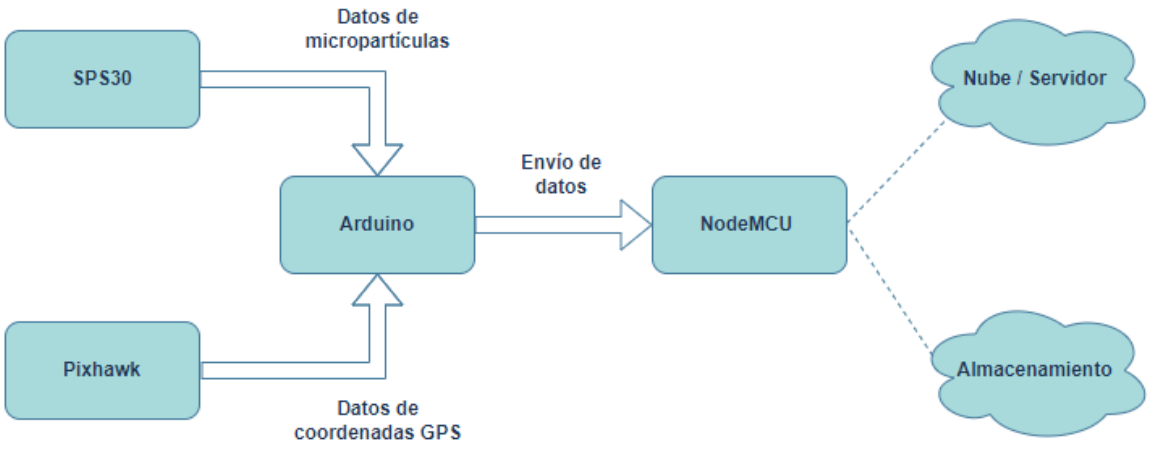

Figura 3.17: Flujo de datos en el sistema completo.

La figura 3.18 muestra todas las conexiones del sistema; se recomienda revisarlas antes de la alimentación del sistema, comprobando que todos los componentes están alimentados con el voltaje apropiado, y que los diferentes componentes estén correctamente conectados. En caso de que el sistema no funcione como debería, es importante revisar especialmente las conexiones encargadas de la comunicación serial entre dispositivos.

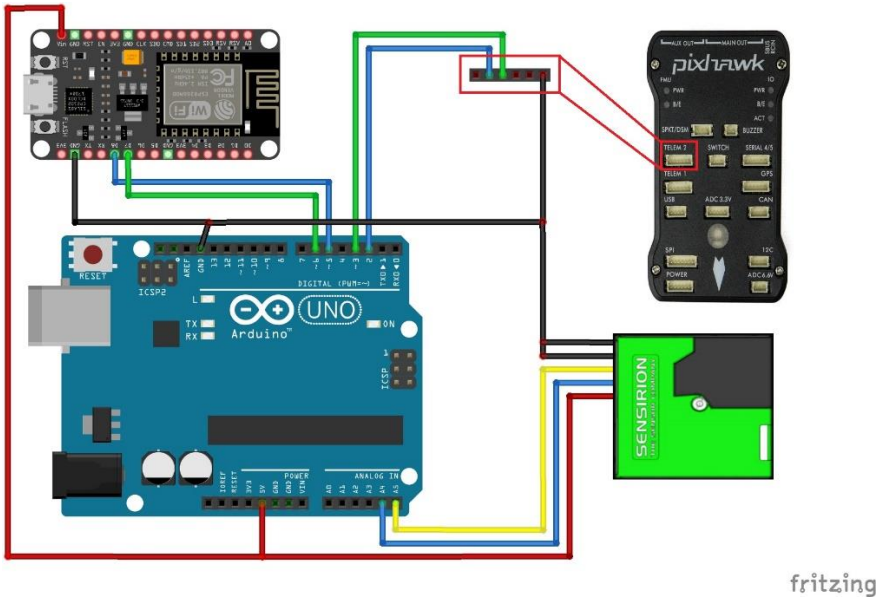

Figura 3.18: Diagrama de conexiones.

El prototipo resultante es bastante compacto y ligero, además de modular. Esto lo hace perfecto para ser incorporado a prácticamente a cualquier drone de forma sencilla. El hecho de que solo 3 conexiones se hagan con el drone en sí permite que el módulo sea fácilmente intercambiable entre varios vehículos. El prototipo usado en este trabajo se muestra en la figura 3.19, así como su montaje en el dron en la figura 3.20.

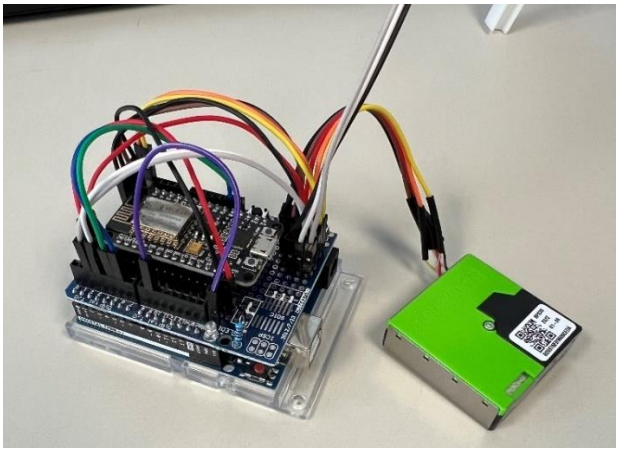

Figura 3.19: Prototipo.

Figura 3.20: Montaje final.

CAPÍTULO 4. DESARROLLO DE LA SOLUCIÓN

En este capítulo se describe la metodología empleada para llegar a la solución final, así como algunos errores cometidos por el camino y su corrección. Además, se muestran las funciones fundamentales del código y se explica el funcionamiento de estas. Los códigos completos se pueden encontrar en el Anexo 1.

4.1. METODOLOGÍA EMPLEADA

Como se ha podido observar en capítulos anteriores, el sistema propuesto debe realizar tres acciones principales: obtener coordenadas GPS, realizar una lectura de micropartículas, y enviar estos datos al módulo NodeMCU. El orden de las mediciones realmente no importa, siempre que, para cada lectura de micropartículas, estén asociadas las coordenadas correspondientes.

En una primera versión del sistema la solución seguía una estructura como la mostrada en la figura 4.1, donde en primer lugar se abren todas las conexiones seriales necesarias para permitir la comunicación entre los diferentes componentes, y a continuación se entra en un bucle de recibir y enviar datos.

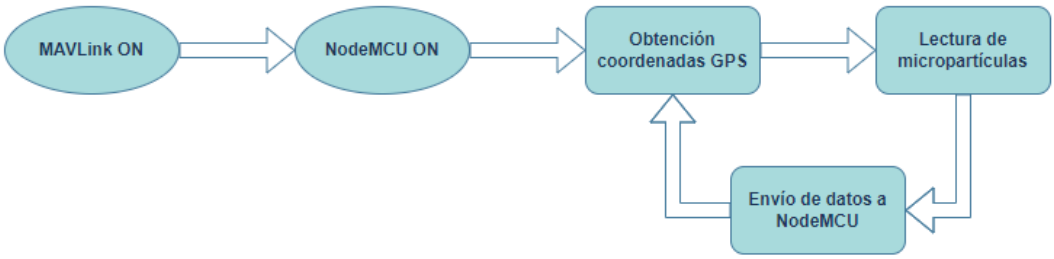

Figura 4.1: Primera versión del sistema.

Esta versión, aunque en teoría correcta, no resultó en la práctica. Tras un estudio del error se encontró que el problema estaba en tener dos comunicaciones seriales abiertas, enviando o recibiendo datos al mismo tiempo, lo cual resultaba en que el proceso se congelaba, no logrando recopilar la información deseada.

Esta es una de las limitaciones que la placa de Arduino utilizada presenta, ya que solo dispone de un canal para conexión serial, y la solución requiere de dos. Debido a esto, para el sistema propuesto se utilizaron dos conexiones "Software Serial", pero esta librería también tiene sus limitaciones, y no permite mandar o recibir datos por diferentes canales al mismo tiempo.

La solución a este problema, y con la condición de mantener el modelo "Uno Rev3" de Arduino por conveniencia, fue mantener una sola conexión serial abierta al mismo tiempo, alternando ambas según se necesitara en ese momento. Es decir, cuando la comunicación MAVLink estuviera en curso, se cerraría la comunicación serial con el NodeMCU, y viceversa. La nueva versión aplicando esta solución se muestra en la Figura 4.2. En los siguientes apartados se explican en detalle los principales componentes de la solución.

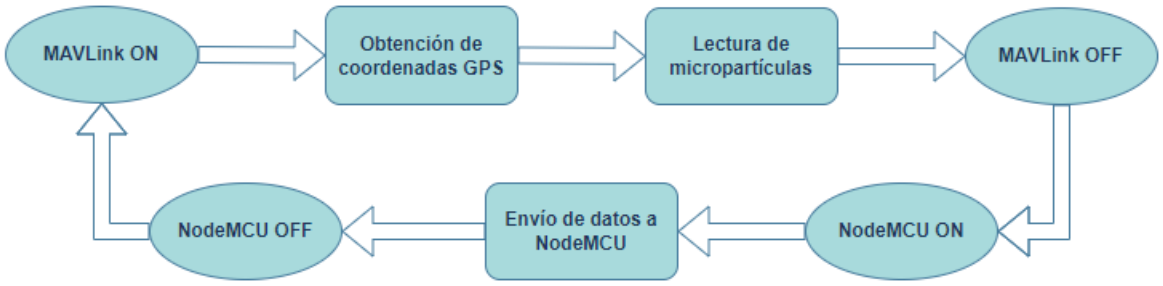

Figura 4.2: Versión final del sistema.

4.2. COMUNICACIÓN MAVLINK PARA OBTENCIÓN DE COORDENADAS GPS

El primer paso en el código de la solución propuesta es la obtención de las coordenadas GPS a través de la comunicación con el protocolo MAVLink entre la controladora de vuelo y la placa de Arduino. Esta consta de diferentes partes, detalladas en este apartado. Para entender la solución y algunos aspectos del código es importante tener un entendimiento de cómo funciona este protocolo.

El protocolo MAVLink no tiene control de flujo, esto quiere decir que se tendrá que mantener un control de la información que se manda, y de aquella que se espera recibir. Es posible que en algunos casos los mensajes se pierdan, y la solicitud de datos se tenga que repetir.

Para esto MAVLink utiliza un envío de latidos (heartbeats). Estos son mensajes que se envían de forma periódica entre los diferentes dispositivos para permitir una comunicación confiable. Estos latidos indican que los dispositivos están operativos y listos para enviar o recibir información, sirven como una señal de vida. Estos mensajes contienen información importante sobre el sistema como el modo de funcionamiento, tipo de vehículo no tripulado, modo de funcionamiento, etc. Con el envío y recepción de latidos se consigue un control eficaz, ya que los diferentes componentes del sistema pueden monitorear el estado y la disponibilidad de los demás, así como detectar errores en la comunicación.

Volviendo al código, se puede observar un bucle principal bastante simple (López, s.f.), ilustrado en la figura 4.3. Se comienza abriendo la comunicación serial MAVLink con el Arduino. A continuación, se envía un latido a la controladora de vuelo, el cual contiene la información importante del sistema, y se empieza a contar el tiempo pasado desde el envío de este. Se verifica si ha pasado cierta cantidad de tiempo desde la última transmisión, y se van contando los latidos que se van enviando. Tras cierto número de latidos, los cuales podemos configurar cambiando la variable “num_hbs” en el código, se realiza una solicitud de datos a la controladora de vuelo mediante la función “Mav_Request_Data”, mostrada en la figura 4.5.

La función de petición de datos, mostrada en la figura 4.6, es crucial, pues solicitamos a la controladora de vuelo las cadenas de datos que nos interesan; incluye los datos de posición global y la velocidad a la que queremos recibirlos. En la página web de MAVLink (REF) se pueden encontrar todas las cadenas que pueden solicitarse.

```
74  //Conexión Mavlink
75      DatosRecibidos = false;
76      Mavlink.begin(57600);
77      delay(100);
78    //Empaquetado del mensaje
79    mavlink_msg_heartbeat_pack(1,0, &msg, type, autopilot_type, system_mode, custom_mode, system_state);
80
81    uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);  //Copiado de mensaje en buffer
82
83    unsigned long currentMillisMAVLink = millis();
84    if (currentMillisMAVLink - previousMillisMAVLink >= next_interval_MAVLink){
85      //Variables de tiempo
86      previousMillisMAVLink = currentMillisMAVLink;
87
88      Mavlink.write(buf, len);
89
90      num_hbs_pasados++;
91      if(num_hbs_pasados >= num_hbs){
92        Mav_Request_Data();             //Petición de datos de Pixhawk
93        num_hbs_pasados = 0;
94      }
95    }
96
97    comm_receive();    //Comprobación de buffer de recepción
98  //!Conexión Mavlink
```

Figura 4.3: Código correspondiente a la comunicación MAVLink desde la placa Arduino.

Una vez se ha solicitado la información necesaria habrá un gran flujo de datos desde la controladora de vuelo, por lo que es necesario filtrar y descodificar aquellos datos necesarios para el proyecto. Este proceso lo realiza la función “comm_receive”, que constantemente lee si hay datos disponibles siendo enviados desde la Pixhawk, filtrando y descodificando estos. Los datos se encuentran en variables de tipo estructura, por lo que es necesario conocer la definición de esta. Esta información puede ser encontrada en la librería “mavlink.h”, debajo del directorio “\common”. En este caso, la estructura que contiene los datos de coordenadas y altura se ilustra en la figura 4.4.

```
typedef struct __mavlink_global_position_int_cov_t {
 uint64_t time_usec; /*< [us] Timestamp (UNIX Epoch time o
 int32_t lat; /*< [degE7] Latitude*/
 int32_t lon; /*< [degE7] Longitude*/
 int32_t alt; /*< [mm] Altitude in meters above MSL*/
 int32_t relative_alt; /*< [mm] Altitude above ground*/
 float vx; /*< [m/s] Ground X Speed (Latitude)*/
 float vy; /*< [m/s] Ground Y Speed (Longitude)*/
 float vz; /*< [m/s] Ground Z Speed (Altitude)*/
 float covariance[36]; /*<  Row-major representation of a
 uint8_t estimator_type; /*<  Class id of the estimator th
} mavlink_global_position_int_cov_t;
```

Figura 4.4: Estructura de datos utilizada.

Como se puede observar, en el código solo se realizará una petición de datos cuando se haya cumplido la condición de tiempo; esto tiene como propósito no sobrecargar el sistema con mensajes de petición de datos. La función de recepción está activa siempre, permitiendo la recepción de datos importantes, incluso cuando no se están solicitando de forma directa. Solo cuando se hayan recibido los datos de coordenadas correctamente se realizará la medición de micropartículas; de esta forma se asegura tener siempre datos completos.

```
182  void Mav_Request_Data(){
183
184    mavlink_message_t msg;
185    uint8_t buf[MAVLINK_MAX_PACKET_LEN];
186
187    const int maxStreams = 1;
188    const uint8_t MAVStreams[maxStreams] = {MAV_DATA_STREAM_POSITION};
189    const uint16_t MAVRates[maxStreams] = {0x05};
190
191    for(int i=0; i<maxStreams; i++){
192
193      mavlink_msg_request_data_stream_pack(2, 200, &msg, 1, 0, MAVStreams[i], MAVRates[i], 1);
194      uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);
195      Mavlink.write(buf, len);
196
197    }
198  delay(1000);
199  }
```

Figura 4.5: Función de petición de datos.

```
201  void comm_receive(){
202
203    mavlink_message_t msg;
204    mavlink_status_t status;
205
206    while(Mavlink.available()>0){
207
208      uint8_t c = Mavlink.read();
209
210      if(mavlink_parse_char(MAVLINK_COMM_0, c, &msg, &status)){      //Obtención de nuevo mensaje
211
212        //Manejo de mensaje
213        switch(msg.msgid){
214
215          case MAVLINK_MSG_ID_HEARTBEAT: // #0: Heartbeat
216          {
217            //Serial.print("HEARTBEAT RECEIVED!\n");     //Activar esta linea para comprobar recepción de Heartbeats
218          }
219          break;
220
221          case MAVLINK_MSG_ID_GLOBAL_POSITION_INT:     // #33: Posición global
222
223            mavlink_global_position_int_t posicion;
224            mavlink_msg_global_position_int_decode(&msg, &posicion);
225
226           latitud = posicion.lat / 10000000.0;   //Los datos de longitud y latitud se reciben con exponente 7
227           longitud = posicion.lon / 10000000.0;
228           altura = posicion.alt / 1000.0;        //La altura se recibe con exponente 3
229
230            Serial.print("Latitud: "); Serial.println(String(latitud,4));
231            Serial.print("Longitud: "); Serial.println(String(longitud,4));
232            Serial.print("Altura: "); Serial.println(String(altura,3));
233
234            DatosRecibidos = true;
235          break;
236
237          default:
238          break;
239        }
240      }
241    }
242  }
```

Figura 4.6: Función de recepción de datos.

4.3. MEDICIÓN DE MICROPARTÍCULAS

La lectura de micropartículas que el programa realiza a continuación es bastante sencilla cuando se usan las librerías y funciones propias que existen para el sensor SPS30 (Winkelmann, s.f.). En primer lugar, se comprueba que haya una comunicación correcta, y que el sensor esté listo para funcionar. Una vez las comprobaciones acaban, y si todo funciona bien, empezarán las mediciones. Los datos de mediciones se reciben en una variable de tipo estructura. El código se muestra en la figura 4.7. Los datos se pueden leer desde el monitor serial del Arduino si se desea.

```
105  //Mediciones SPS30
106      struct sps30_measurement mediciones;
107      char serial[SPS30_MAX_SERIAL_LEN];
108      uint16_t data_ready;
109      int16_t ret;
110
111    do{
112      ret = sps30_read_data_ready(&data_ready);
113      if(ret<0){
114        Serial.print(F("Error leyendo medición: "));
115        Serial.println(ret);
116      } else if (!data_ready)
117        Serial.print(F("No hay nuevas mediciones disponibles\n"));
118        else
119         break;
120         delay(100);
121    }while(1);
122
123    ret = sps30_read_measurement(&mediciones);
124    if(ret<0){
125      Serial.print(F("Error leyendo mediciones\n"));
126    }else {
127      Serial.print("PM  1.0: "); Serial.println(mediciones.mc_1p0);
128      Serial.print("PM  2.5: "); Serial.println(mediciones.mc_2p5);
129      Serial.print("PM  4.0: "); Serial.println(mediciones.mc_4p0);
130      Serial.print("PM 10.0: "); Serial.println(mediciones.mc_10p0);
131
132      Serial.print("NC  0.5: "); Serial.println(mediciones.nc_0p5);
133      Serial.print("NC  1.0: "); Serial.println(mediciones.nc_1p0);
134      Serial.print("NC  2.5: "); Serial.println(mediciones.nc_2p5);
135      Serial.print("NC  4.0: "); Serial.println(mediciones.nc_4p0);
136      Serial.print("NC 10.0: "); Serial.println(mediciones.nc_10p0);
137
138      Serial.print("Typical particle size: "); Serial.println(mediciones.typical_particle_size);
139      Serial.println();
140    }
141  //!Mediciones SPS30
```

Figura 4.7: Código correspondiente al proceso de medición de micropartículas desde Arduino.

4.4. EMPAQUETADO Y ENVÍO DE DATOS A NODEMCU

Una vez obtenidos los datos de coordenadas y de micropartículas, el último paso es enviarlos al módulo NodeMCU. Un paso crucial antes de enviarlos es cerrar la comunicación serial de MAVLink, y abrir la correspondiente a este módulo hasta que los datos hayan sido enviados. Para obtener suficiente precisión en las medidas se ha usado la función “String” de Arduino para mostrar cuatro decimales en datos de coordenadas, tres en altura, y cuatro en datos de partículas. Este proceso se muestra en la figura 4.8.

```
143  //Envio de datos NodeMCU
144   Mavlink.end();
145   delay(100);
146   NodeMCU.begin(115200);
147   delay(100);
148
149   NodeMCU.print(String(latitud,4)); NodeMCU.print(" ");
150   NodeMCU.print(String(longitud,4)); NodeMCU.print(" ");
151   NodeMCU.print(String(altura,3)); NodeMCU.print(" ");
152
153   NodeMCU.print(String(mediciones.mc_1p0,4)); NodeMCU.print(" ");
154   NodeMCU.print(String(mediciones.mc_2p5,4)); NodeMCU.print(" ");
155   NodeMCU.print(String(mediciones.mc_4p0,4)); NodeMCU.print(" ");
156   NodeMCU.print(String(mediciones.mc_10p0,4)); NodeMCU.print(" ");
157
158   NodeMCU.print(String(mediciones.nc_0p5,4)); NodeMCU.print(" ");
159   NodeMCU.print(String(mediciones.nc_1p0,4)); NodeMCU.print(" ");
160   NodeMCU.print(String(mediciones.nc_2p5,4)); NodeMCU.print(" ");
161   NodeMCU.print(String(mediciones.nc_4p0,4)); NodeMCU.print(" ");
162   NodeMCU.print(String(mediciones.nc_10p0,4)); NodeMCU.print(" ");
163
164   NodeMCU.print(String(mediciones.typical_particle_size,4)); NodeMCU.print(" ");
165
166   NodeMCU.print("\n"); //Salto de linea para indicar final de transmisión
167
168   NodeMCU.end();
169   delay(100);
170  //!Envio de datos NodeMCU
```

Figura 4.8: Envío de datos a NodeMCU desde Arduino.

4.5. RECEPCIÓN DE DATOS NODEMCU

Para la recepción de datos en el módulo NodeMCU se utiliza un código muy sencillo, mostrado en la figura 4.9. Este sencillamente espera hasta que haya información disponible, y lee los datos hasta encontrar un salto de línea, lo cual indica el final de la transmisión de un paquete de información. El usuario puede añadir más funciones a este código dependiendo de cómo desee procesar los datos. En el caso de este proyecto se obtendrán los datos a partir del monitor serial y se pasarán a un documento para su interpretación.

```
1   #include <SoftwareSerial.h>
2   SoftwareSerial NodeMCU(D7,D8); //RX(D7) TX(D8)
3
4   void setup() {
5
6    Serial.begin(9600);    //Monitor Serial NodeMCU
7    NodeMCU.begin(115200); //Conexión Serial con Arduino
8   }
9
10  void loop() {
11
12   if(NodeMCU.available()){
13     String mediciones = NodeMCU.readStringUntil('\n');
14     Serial.println( mediciones );
15     //Latitud Longitud Altura PM1.0 PM2.5 PM4.0 PM10.0 NC0.5 NC1.0 NC2.5 NC4.0 NC10.0 Media
16   }
17
18  }
```

Figura 4.9: Código ejecutado en el módulo NodeMCU.

CAPÍTULO 5. RESULTADOS

En este capítulo se muestran los resultados obtenidos tras realizar algunos experimentos con el prototipo propuesto, y relacionados con la medición de micropartículas. Se explican las condiciones y parámetros de estas pruebas, su objetivo, y se analizan de tres formas diferentes los resultados obtenidos en función de diferentes variables.

5.1. PROCEDIMIENTO Y PARÁMETROS DE LAS MEDICIONES

Para poner a prueba el sistema propuesto, en este trabajo se realizan una serie de medidas de micropartículas en el campus de la Universidad Politécnica de Valencia. Debido a que no se posee el permiso para volar sobre este, las medidas se realizarán a pie cargando el dron, y cubriendo suficiente espacio para poder tener datos relevantes.

En los apartados siguientes de este capítulo se analizarán los datos de micropartículas en función del espacio, el tiempo y la altura. Para las dos primeras se realiza un recorrido sobre el campus cubriendo un área rectangular, mostrado en la figura 5.1, con el objetivo de facilitar su posterior interpretación en mapas de calor. Para su análisis en mapas de calor se recogerán suficientes datos para tener referencias sobre la mayor área posible del campus, y se realizará una interpolación espacial con MATLAB para tener una representación de todo el espacio.

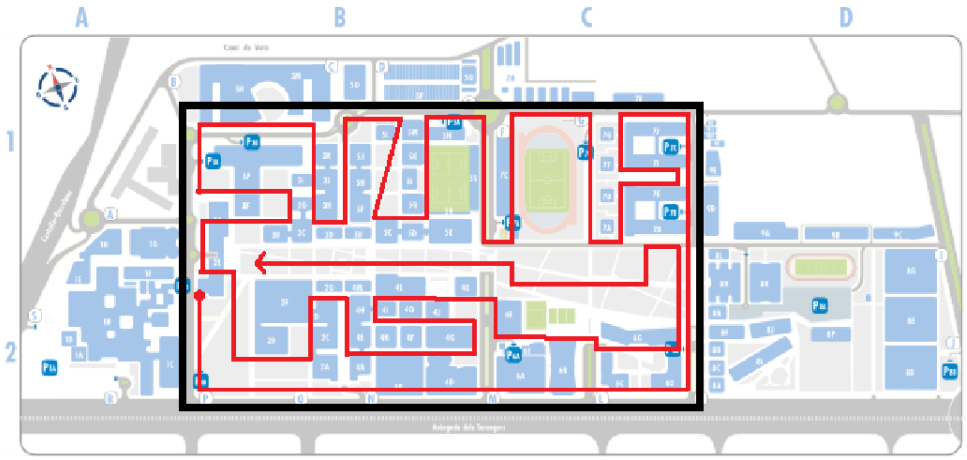

Figura 5.1: Recorrido realizado.

Un detalle importante para realizar la medición respecto al espacio es disponer de suficientes decimales en las mediciones de latitud y longitud ya que, en superficies relativamente pequeñas como estas, la variación entre una posición y la siguiente es muy pequeña. En una primera toma de estas medidas se cometió el error de recoger datos de coordenadas con tan solo dos decimales, y por consecuencia no se puedo realizar un análisis válido. Esto puede ser ajustado fácilmente en el código del programa ejecutado en la placa Arduino.

Para el análisis de concentración de micropartículas respecto al tiempo se utilizarán los mismos datos del experimento previo. Durante el recorrido por el campus para recoger los datos se realizó también una medida del tiempo total que esto conllevó. Se considerará una velocidad uniforme de toma de datos para la interpretación de estas.

Finalmente, para el análisis de partículas respecto a la altura, lo ideal sería poder volar el drone y tomar mediciones en un rango significativo, pero, debido a las limitaciones del proyecto, se utilizará en su lugar un edificio del campus, y se tomarán medidas a diferentes alturas de este, tomando la misma columna de aire. El edificio se encuentra en la escuela de ingeniería industrial, y tanto este como los puntos de medición (marcados con cuadrados rojos) se pueden observar en la figura 5.2 En este proceso se permitirá que las mediciones de micropartículas se estabilicen; para mayor precisión se tomarán varias medidas, y se usará la media ponderada de estas para generar los diferentes gráficos.

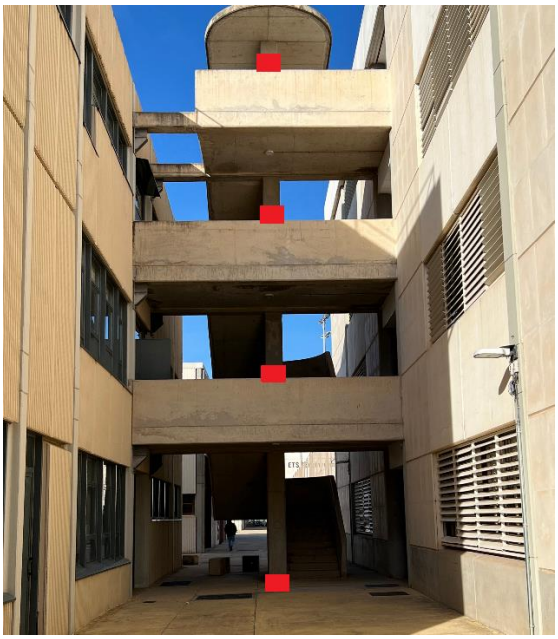

Figura 5.2: Mediciones de alturas.

5.2. ANÁLISIS DE LA CONCENTRACIÓN DE PARTÍCULAS RESPECTO AL TIEMPO

A continuación, se realiza un análisis de la concentración de micropartículas respecto al tiempo. Se han realizado tres gráficas, representando: concentración en masa de los diferentes tipos de partículas (figura 5.3); concentración numérica (figura 5.4); y media de tamaño de partícula respecto al tiempo (figura 5.5).

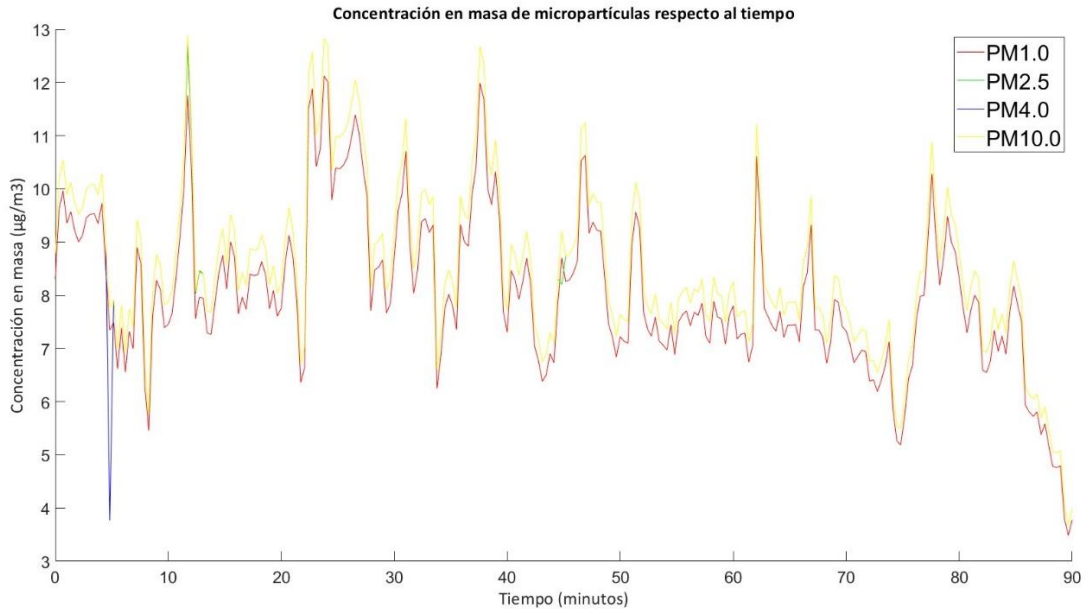

Figura 5.3: Concentración en masa respecto al tiempo.

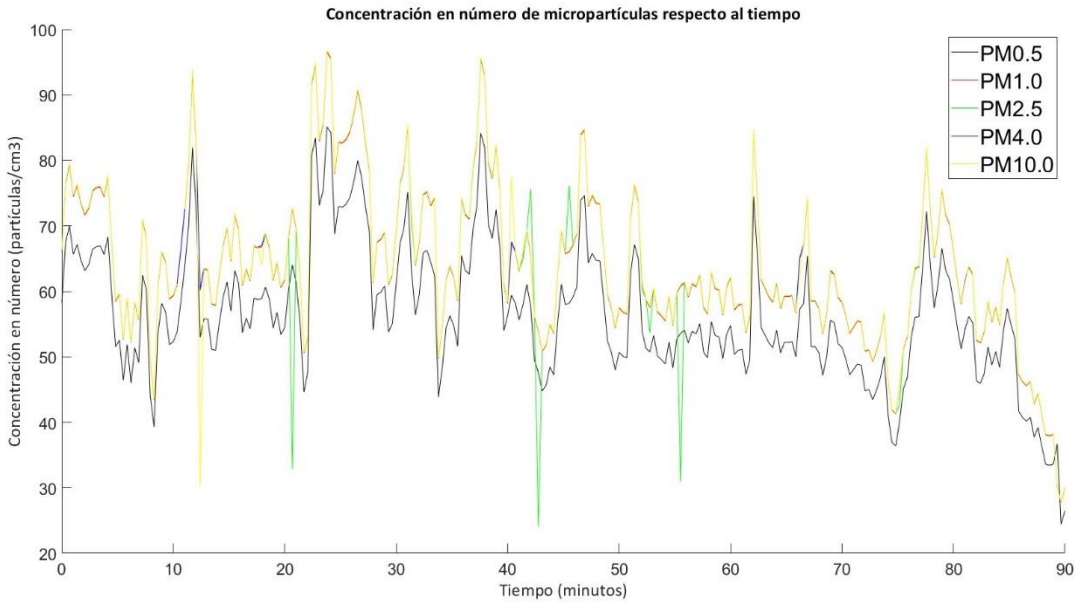

Figura 5.4: Concentración en número respecto al tiempo.

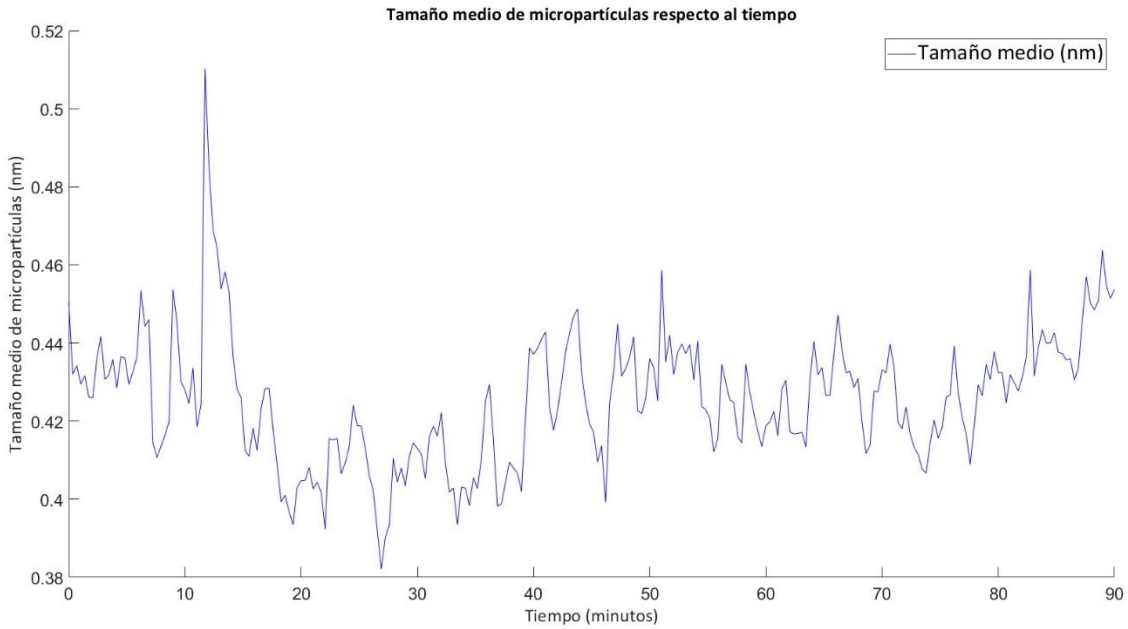

Figura 5.5: Tamaño medio de partícula respecto al tiempo.

Según las gráficas, como se mencionó en el capítulo anterior, hay muy poca diferencia entre las concentraciones de los diferentes tipos de micropartículas, y que siguen la misma tendencia. Al haber realizado este estudio paseando por el campus se puede poner en contexto la exposición que una persona tendría a la contaminación y, como los cambios de concentración de esta pueden ser bastantes repentinos, dependiendo de la zona en la que se encuentre. Estos análisis también pueden mostrar al usuario si existe alguna medida inusual, pudiendo explorar más en profundidad o detectar errores en las mediciones.

5.3. ANÁLISIS DE LA CONCENTRACIÓN DE PARTÍCULAS EN EL ESPACIO

A continuación, se muestran en las figuras 5.6-5.14 los resultados obtenidos a partir de la generación de datos de calor, para analizar la concentración de micropartículas en el espacio. El script de MATLAB facilitado en el Anexo 1 puede ser fácilmente adaptado para conseguir cualquiera de estos. Se muestran en primer lugar la representación de los datos de concentración en masa, seguidos de los datos de concentración en número de los diferentes tamaños de partículas. Finalmente, se muestra un análisis del tamaño medio de partícula respecto al espacio en la figura 5.15, y se analizan los resultados obtenidos.

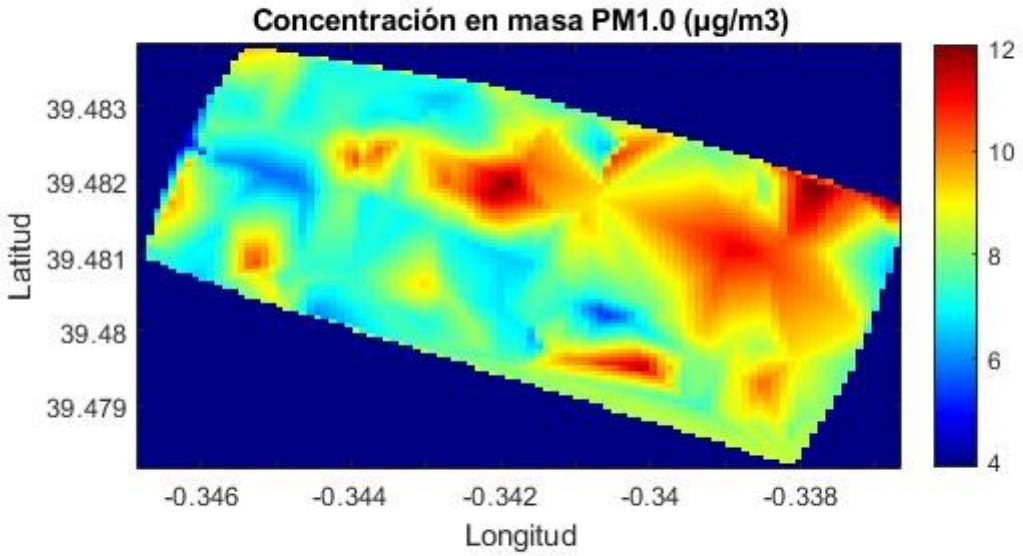

Figura 5.6: Concentración en masa PM1.0.

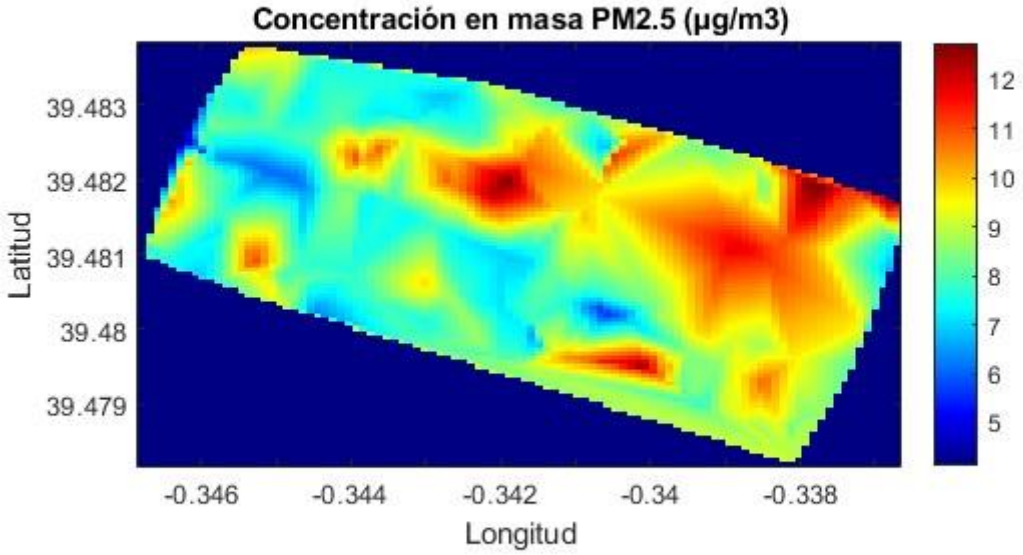

Figura 5.7: Concentración en masa PM2.5.

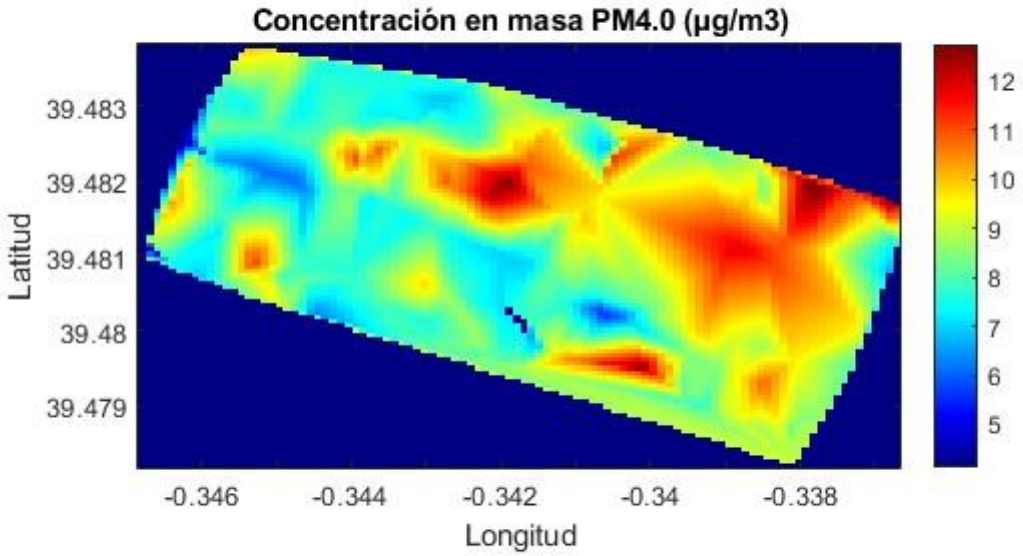

Figura 5.8: Concentración en masa PM4.0.

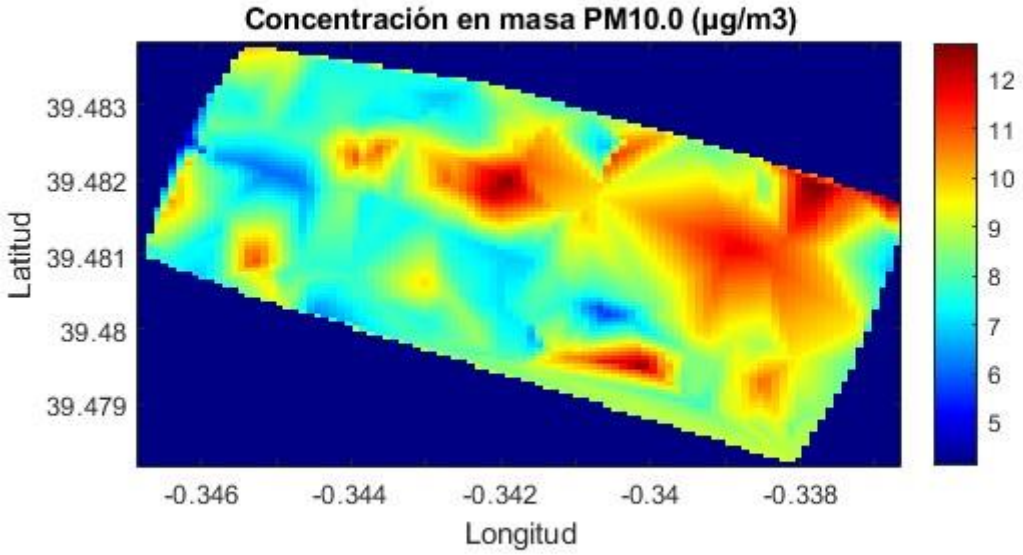

Figura 5.9: Concentración en masa PM10.0.

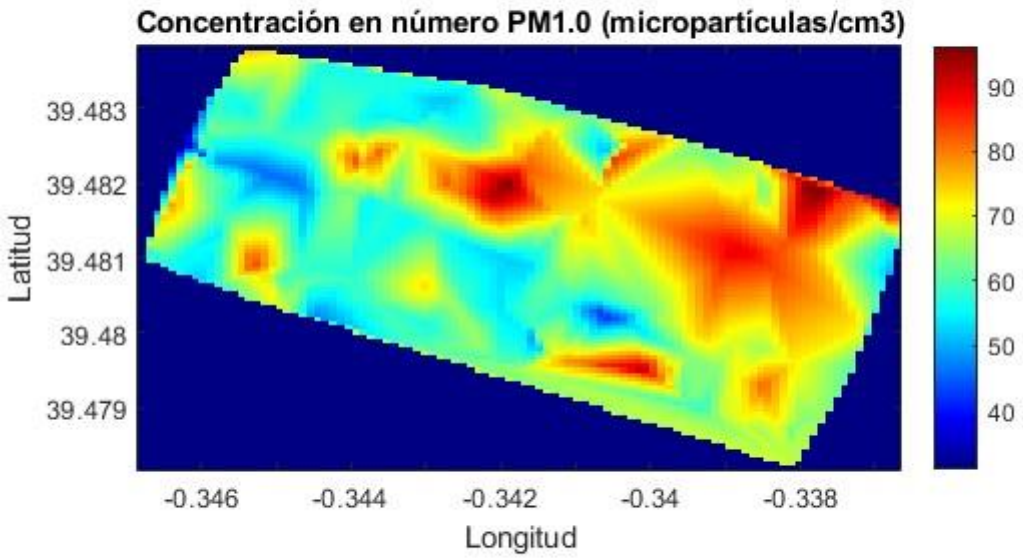

Figura 5.10: Concentración numérica PM1.0.

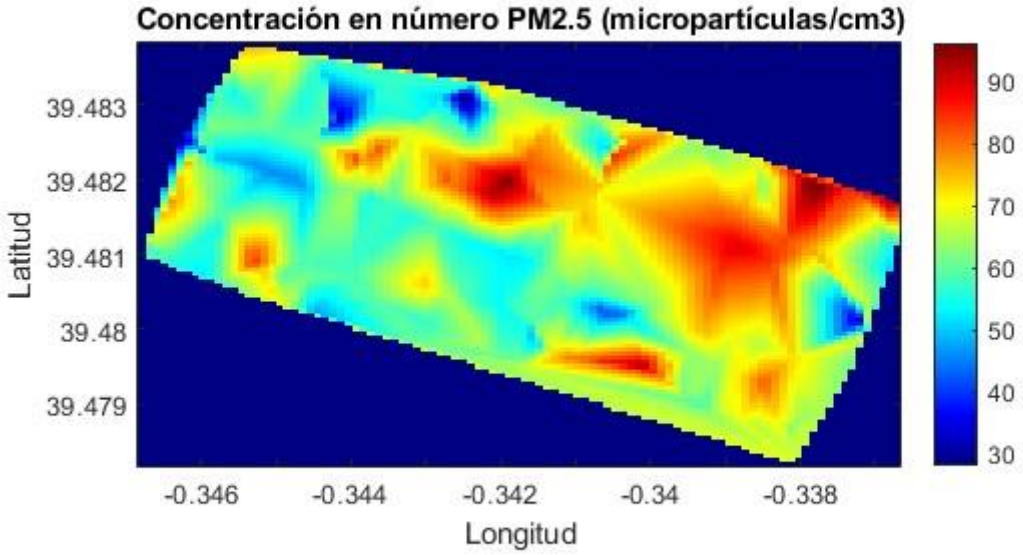

Figura 5.11: Concentración numérica PM2.5.

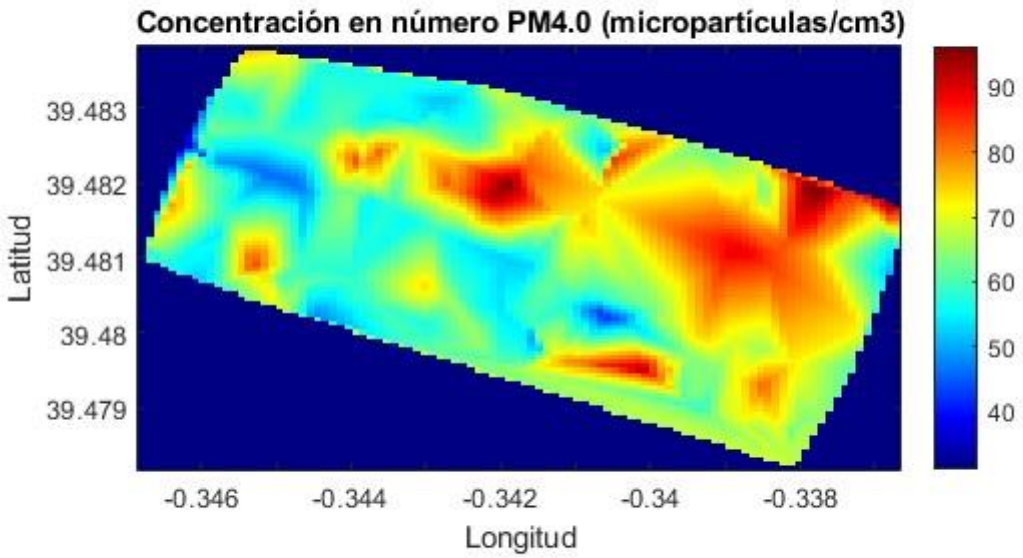

Figura 5.12: Concentración numérica PM4.0.

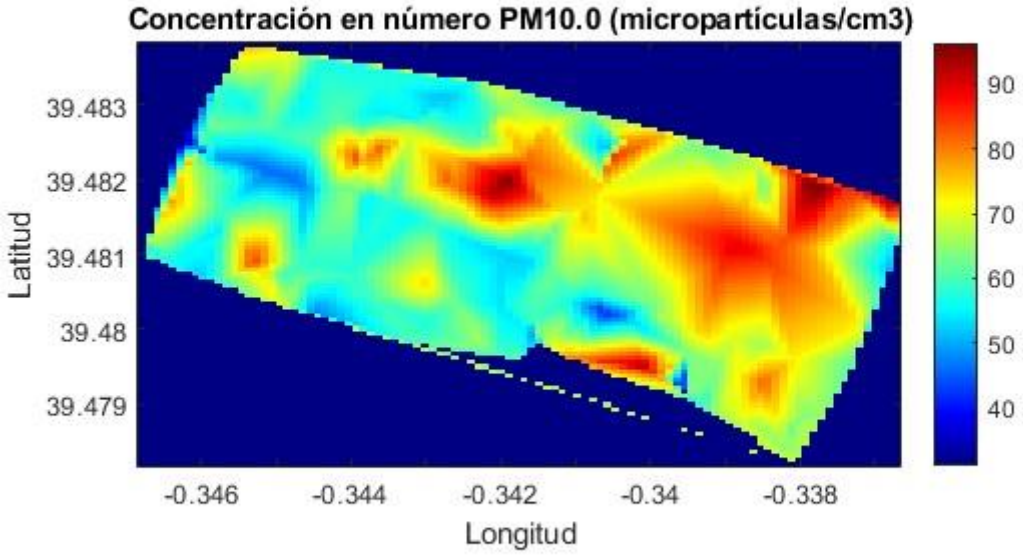

Figura 5.13: Concentración numérica PM10.0.

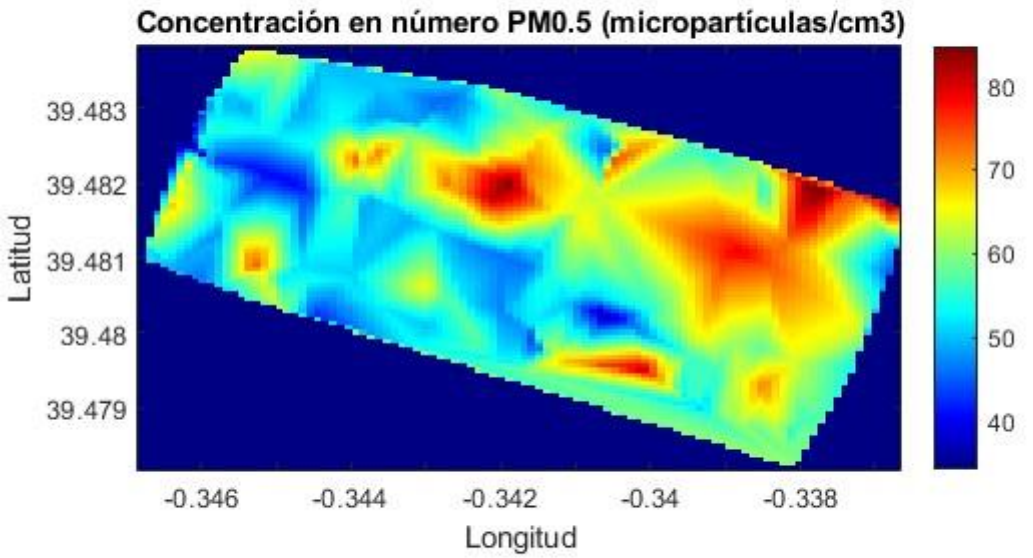

Figura 5.14: Concentración numérica PM0.5.

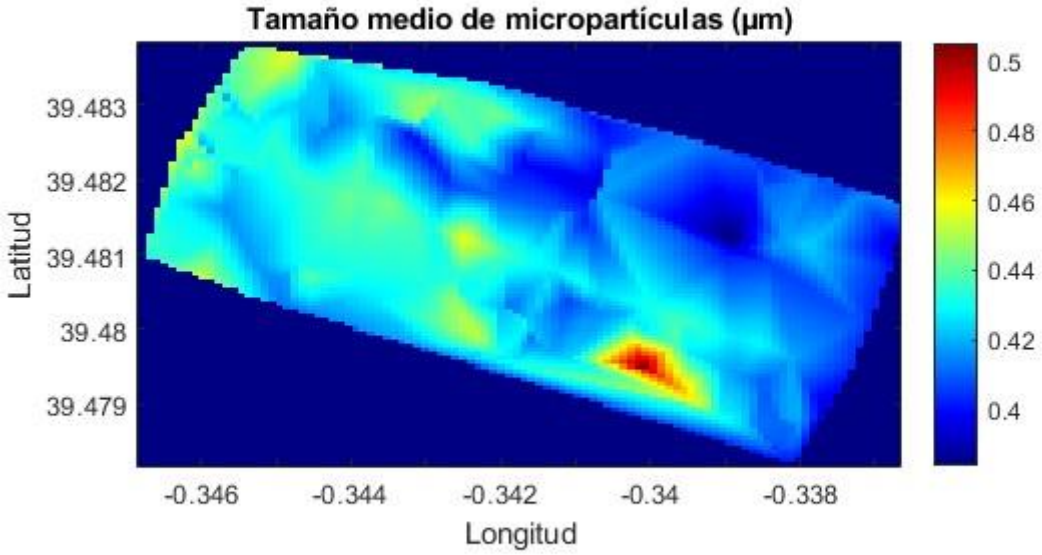

Figura 5.15: Tamaño medio de partículas.

Se analizan a continuación los resultados obtenidos por los mapas de calor. Como se puede observar, las diferencias entre la mayoría de ellos son mínimas, a excepción del análisis de PM0.5 y del tamaño medio de partícula. Para facilitar el entendimiento de estos se ha solapado uno de ellos con el mapa de la universidad, el resultado se muestra en la Figura 5.16, donde se han marcado ciertas áreas de especial interés para ser discutidas.

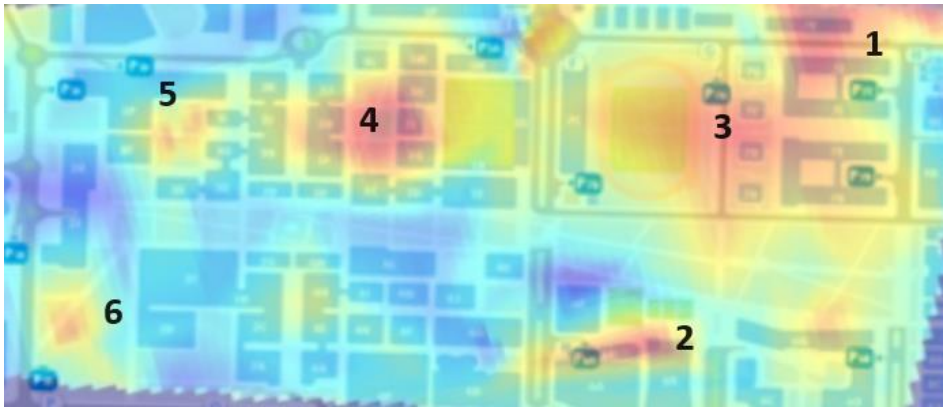

Figura 5.16: Áreas destacadas.

Los resultados obtenidos en comparación con el mapa son bastante interesantes. Se discutirán los puntos de mayor interés, en donde la concentración de micropartículas ha alcanzado los valores más altos.

Empezando por el área 1 se puede observar una gran concentración de partículas proveniente del exterior de la universidad. Observando en el mapa tiene sentido, en primer lugar, debido a la carretera que hay en ese lugar, pero sobre todo ya que esa zona se encuentran algunos invernaderos con bastantes unidades de calefacción o ventilación como se muestra en la figura 5.17. Todas estas unidades que apuntan hacia la universidad encajarían con el gran incremento de micropartículas en esa área.

Figura 5.17: Invernaderos.

A continuación, se analiza el área 2, muy interesante debido a la gran concentración de micropartículas en un área muy concreta. Tras comprobar el mapa y acudir a la zona en cuestión, se encuentra que ese pico en contaminación coincide exactamente una entrada a uno de los aparcamientos de la universidad, mostrado en la figura 5.18, donde hay circulación constante.

Figura 5.18: Entrada del aparcamiento.

Pasando al área 3 se encuentra una concentración menor en comparación con los otros lugares, pero muy extendida. Esta medición corrobora algo muy similar al área 2, ya que coincide perfectamente con una carretera de la universidad, mostrada en la figura 5.19, muy utilizada para acceder a varios aparcamientos en esa zona, algunos de ellos subterráneos pero abiertos al aire libre en algunas zonas. Esta alta concentración de partículas se extiende sobre una superficie considerable, afectando a los edificios y pista de atletismo colindantes.

Figura 5.19: Aparcamientos.

El área 4 muestra un alto nivel de concentración de micropartículas. Es muy curioso observar cómo esta zona se sitúa sobre la escuela de ingeniería industrial. En un principio no se entendían muy bien estos resultados, ya que es una zona ajardinada bastante lejos de cualquier carretera. Sin embargo, al ir al lugar persona un detalle que se observó fue la cantidad de conductos de ventilación provenientes de algunos edificios que apuntan hacia esta área de jardines. Estos se muestran en la figura 5.20. Tiene sentido ya que esta zona cuenta con muchos laboratorios de construcción, químicos, etc. Sería muy interesante realizar un estudio en detalle en este lugar para confirmar esto y tener más información.

Figura 5.20: Chimeneas sobre la escuela de industriales.

Para acabar se hace una breve mención a las áreas 5 y 6. La primera coincide con otra pequeña área entre edificios. La hipótesis es similar al área 4: conductos de ventilación, aires acondicionados, etc. El área 6 fue un punto curioso en este estudio, ya que está en medio de un jardín sin causas particulares a la vista. La hipótesis barajada en este caso es que el sensor haya detectado algunas partículas procedentes de los árboles, ya que esta zona está cubierta por estos. Puede que un pequeño levantamiento de arena al caminar, o por personas circundantes, lo cause, ya que es una zona terregosa. Resultaría muy útil analizar esta área más en profundidad.

Destacan los resultados del análisis de tamaño medio de partícula. Destaca sobre todas las demás el área 2, con una media mayor a toda la demás área del campus. Tras exploración cabe decir que el aire se siente notablemente más cargado en esa área, lo cual tiene sentido por el aparcamiento y por ser un área más cerrada que la 3, por ejemplo.

Como se ha podido observar en este apartado el análisis de micropartículas mediante mapas de calor es probablemente la mejor forma para realizar estos análisis. La lectura de los resultados revela rápidamente zonas de interés para estudios en profundidad y, al disponer de las coordenadas, y pudiendo comparar datos con el mapa real, permite al usuario ganar mucho entendimiento sobre lo que está sucediendo a sus alrededores. Por otro lado, se observa que la diferencia entre las áreas de concentración de los diferentes tamaños de partícula es mínima.

5.4. ANÁLISIS DE LA CONCENTRACIÓN DE PARTÍCULAS RESPECTO A LA ALTURA

Finalmente se procede a realizar el análisis de partículas respecto a la altura. Para este se han generado tres gráficas, la primera respecto a la concentración en masa, la segunda respecto a la concentración en número de los diferentes tamaños de micropartículas, y una tercera mostrando el tamaño medio de partícula respecto a la altura.

En este análisis se puede observar algo muy bastante inesperado e interesante. La hipótesis inicial era que ambas concentraciones disminuirían un poco a medida que fuéramos ascendiendo a una altura más alta; sin embargo, la realidad del experimento fue ligeramente diferente.

Como se muestra en la figura 5.22 la concentración numérica de todos los tipos de partículas disminuye de forma notable con la altura, lo cual se esperaba. Sin embargo, si se observa la concentración en masa respecto a la altura en la Figura 5.21, se puede ver que esta tiene una tendencia ascendente para todos los tipos de partículas excepto las PM1.0. En un principio podría pensarse que estos datos se contradicen, sin embargo, esto no es así.

Las Partículas en Suspensión (PM) se clasifican en función de su tamaño en micrómetros. Por ejemplo, las partículas PM1.0 tienen un diámetro igual o inferior a un micrómetro, mientras que las partículas PM10.0 tienen un diámetro igual o inferior a diez micrómetros. Sin embargo, el peso de las partículas no depende exclusivamente de su tamaño, sino de diversos factores, como su composición. Este fenómeno puede ser observado en las mediciones.

Leyendo los datos, y entendiendo el principio explicado anteriormente, se puede realizar una lectura más clara de los datos. Como se observa en las gráficas, las partículas PM1.0 disminuyen tanto en cantidad como en masa. Sin embargo, se observa que, para los otros tipos de partículas, la concentración en masa asciende a la vez que la concentración desciende. Esto muestra que, con el aumento de la altura, hay menos partículas, pero estas son más pesadas. Con la altura el tamaño medio de partícula también aumenta, como se observa en la Figura 5.23.

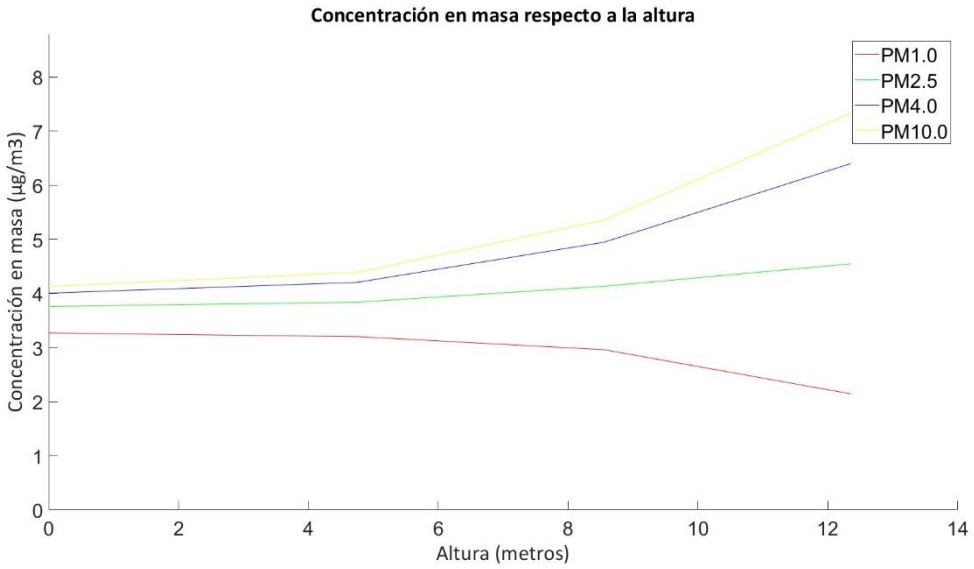

Figura 5.21: Concentración en masa con respecto a la altura.

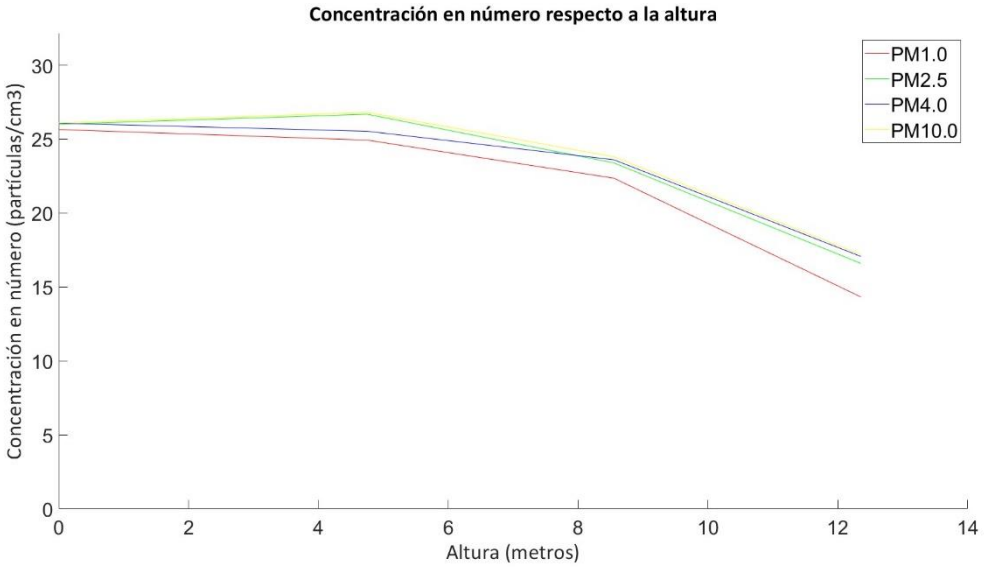

Figura 5.22: Concentración numérica con respecto a la altura.

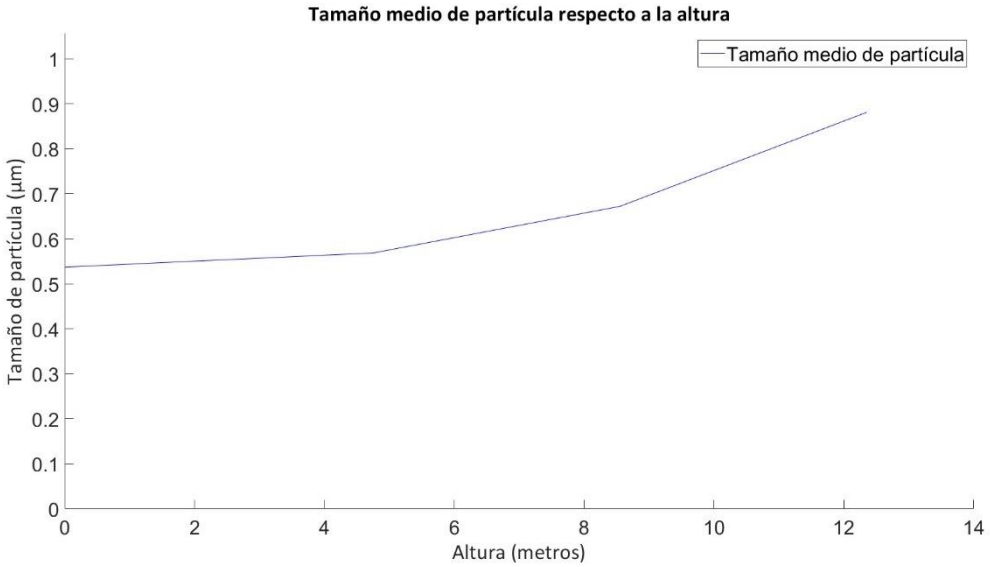

Figura 5.23: Tamaño medio de partículas con respecto a la altura.

Cabe mencionar que este experimento obtendría mejores resultados si tuviera un rango mayor de alturas, y sería muy interesante la realización de este sobre una zona de alta concentración de partículas para ver qué ocurre con estas, y como es su difusión en la columna de aire. Con más datos se podrían tener un análisis mucho confiable de las tendencias, así como confirmar la hipótesis anterior respecto a la masa de las diferentes partículas variando según su composición, y sin ser su tamaño el factor determinante de su peso.

CAPÍTULO 6. CONCLUSIONES Y POSIBLES APLICACIONES

En este último capítulo se sacarán algunas conclusiones sobre el proyecto, se explorarán sus limitaciones, y se propondrán algunos trabajos futuros.

6.1. CONCLUSIONES

Una vez realizadas las pruebas descritas en el capítulo anterior, y conociendo un poco más el contexto actual sobre la monitorización de partículas y el uso de drones, se puede decir que este trabajo ha traído mucha claridad sobre este ámbito, y que su realización ha ayudado a despertar conciencia y nuevos conocimientos. A continuación, se mencionan las conclusiones principales obtenidas.

Se ha ganado conocimiento sobre el ámbito de micropartículas, la importancia de su monitorización, y los efectos que estas pueden tener sobre la salud de las personas. Explorar las tecnologías actuales de monitorización de partículas y el uso de drones ha sido algo muy satisfactorio.

Tras el estudio realizado, es obvio que el uso de drones es una herramienta muy potente y eficaz para la monitorización de partículas en el aire, ofreciendo datos relevantes y una interpretación muy intuitiva de estos. La solución propuesta en este trabajo ha demostrado ser muy eficaz en realizar estos análisis. También abre las puertas a la realización de estudios preliminares detectando anomalías en un área para su posterior estudio en detalle, como se ha visto en el capítulo 5.

El análisis de datos respecto a diferentes parámetros da información única, y arroja mucha luz sobre el comportamiento de las partículas. Los mapas de calor parecen ser la forma más útil de representar estos datos.

En el desarrollo de este proyecto se han ganado también conocimientos sobre programación, sobre protocolos como MAVLink, y sobre diferentes formas de representación de información. Además, la aplicación práctica y el componente experimental de este proyecto han ayudado a asentar el conocimiento sobre muchos temas diferentes.

En conclusión, se considera este trabajo y los resultados obtenidos un éxito, y de interés para la universidad y la comunidad científica en general.

6.2. LIMITACIONES DEL PROYECTO

Antes de operar con el sistema es crucial conocer sus limitaciones y los límites existentes en la operación de este. Las principales limitaciones de este proyecto se explican brevemente a continuación.

En primer lugar, y probablemente la limitación más importante en relación con este proyecto, es la duración de las baterías usadas en los drones. Por lo general, estas proveen unos pocos minutos de vuelo (normalmente menos de 40 minutos), por lo que, para realizar medidas en grandes superficies, o prolongadas en el tiempo, habría que tenerlo en cuenta, ya que incrementaría el número de equipos necesarios para realizar las pruebas. Hay algunas formas de mejorar este aspecto, como por ejemplo usando baterías en paralelo, aumentando la duración (pero también incrementando más el peso del sistema, lo cual de nuevo reduce el tiempo de vuelo).

Por otro lado, se deben entender las limitaciones técnicas de los componentes utilizados por conveniencia. La placa de Arduino Uno tiene una memoria interna limitada y no puede almacenar códigos muy extensos o procesar datos a una velocidad demasiado alta cuando hay mucha información fluyendo a través del sistema. En el caso de este proyecto, el código utilizado en el Arduino ocupa casi toda su memoria. Por otro lado, no dispone de más de un puerto dedicado a la comunicación serial como otros modelos de Arduino; por ello, en capítulos anteriores se explica la necesidad de establecer las conexiones mediante software. Se recuerda que se eligió esta placa por motivos de espacio, peso y consumo.

El sensor SPS30 también presenta ciertas limitaciones. A pesar de ser un sensor con muy buena precisión y calidad para su nivel de precio, no es el más eficiente o preciso en el mercado.

Finalmente, cabe recordar que la operación de drones debe realizarla personal con las licencias y permisos necesarios para esta y obedecer todas las leyes que regulan el uso de vehículos no tripulados. En áreas urbanas, la incorrecta operación puede resultar peligrosa para las personas, y por eso el operario deberá asegurarse de volar con responsabilidad y seguir la regulación aeronáutica definida por la Agencia Estatal de Seguridad Aérea (AESA).

6.3. TRABAJOS FUTUROS

En este apartado se propondrán varias ideas para trabajos futuros usando el trabajo realizado en este proyecto como base. Este sistema es fácilmente expandible ya que todo el software y hardware utilizado es de código abierto.

6.3.1. Muestreo de partículas mediante enjambre de drones

El primer trabajo propuesto es la medición de micropartículas con un enjambre de drones, que implica el uso de múltiples drones que operan en conjunto para realizar una tarea mucho más eficiente de lo posible con una unidad. En la figura 6.1 se muestra un ejemplo de enjambre.

Figura 6.1: Enjambre de drones.

En el caso del proyecto, y siempre que se disponga de los recursos necesarios, esto haría mucho más fácil tomar las medidas, ya que los drones podrían realizar un barrido del área en paralelo, y la trayectoria de cada unidad podría ser tan simple como una línea recta. El trabajar con varias unidades ayudaría a mejorar la limitación de la batería, ya que el espacio ahora se reparte entre los drones, y estos podrían cubrir una superficie mayor sin tener que parar para cambiar la batería.

Además, los drones podrían ser conectados entre sí utilizando el protocolo MAVLink, y estas misiones podrían ser realizadas en piloto automático si se planeara la misión con Mission Planner. Por otro lado, los módulos NodeMCU de los diferentes drones podrían tener comunicación con un servidor único, y subir los datos directamente a la nube o en una unidad de almacenamiento que el dron principal cargue. Como se ve, las posibilidades al disponer de más de un dron permiten muchas mejoras.

6.3.2. Estudio de difusión de partículas con la altura

La segunda propuesta está relacionada con la difusión de partículas con relación a la altura. Como se ha mencionado en capítulos anteriores, esta es una de las motivaciones de este proyecto, ya que los drones nos permiten medir las partículas en suspensión en diferentes alturas, y analizar cómo se comportan.

Sería muy interesante estudiar las áreas industriales como la mostrada en la figura 6.2, o las áreas de más concentración de partículas descubiertas, y medir hasta donde el dron pueda legalmente (120 metros). Esto nos permitiría observar y hacer un muy buen estudio de la calidad del aire.

Figura 6.2: Emisiones en planta industrial.

6.3.3. Monitorización de partículas en tiempo real

Finalmente, se propone crear un sistema para la monitorización en tiempo real de la concentración de partículas. Esto se podría conseguir creando una interfaz en línea servida por el NodeMCU principal; esta interfaz podría usar diferentes lenguajes de programación para funcionar. Su objetivo sería crear los mapas de calor en tiempo real con los datos recibidos, y actualizar la información cuando haya datos nuevos. Para esto el drone, mediante el NodeMCU, mandaría los datos a la nube para su procesado, en lugar de almacenarlos.

Esta interfaz sería también accesible mediante cualquier dispositivo móvil en rango, y sería bastante útil para estudiar puntos en concreto. Como se puede ver, existe un rango de posibilidades muy grande esperando que el usuario adapte el proyecto a sus necesidades específicas.

REFERENCIAS

REFERENCIAS BIBLIOGRÁFICAS

Englert, N. (2004). Fine particles and human health—a review of epidemiological studies. *Toxicology Letters*, *149*(1–3), 235-242. https://doi.org/10.1016/j.toxlet.2003.12.035

Shou, Y., Huang, Y., Zhu, X., Liu, C., Hu, Y., & Wang, H. (2019). A review of the possible associations between ambient PM2.5 exposures and the development of Alzheimer's disease. *Ecotoxicology and Environmental Safety*, *174*, 344-352. https://doi.org/10.1016/j.ecoenv.2019.02.086

Ortiz, J. L. (2022). Incidencia de material particulado (PM10/PM2,5) en enfermedades respiratorias de pobladores en urbanización Nuevo San Lorenzo.

Kappos, A. D., Bruckmann, P., Eikmann, T., Englert, N., Heinrich, U., Höppe, P., Koch, E., Krause, G. H. M., Kreyling, W. G., Rauchfuss, K., Rombout, P., Schulz-Klemp, V., Thiel, W. R., & Wichmann, H.-E. (2004). Health effects of particles in ambient air. *International Journal of Hygiene and Environmental Health*, *207*(4), 399-407. https://doi.org/10.1078/1438-4639-00306

Wang, S., Ji, Y., Zhao, J., Lin, Y., & Lin, Z. (2020). Source apportionment and toxicity assessment of PM2.5-bound PAHs in a typical iron-steel industry city in northeast China by PMF-ILCR. *Science of The Total Environment*, *713*, 136428. https://doi.org/10.1016/j.scitotenv.2019.136428

Luo, Y., Zhou, X., Zhang, J., Xiao, Y., Wang, Z., Zhou, Y., & Wang, W. (2018). PM2.5 pollution in a petrochemical industry city of northern China: Seasonal variation and source apportionment. *Atmospheric Research*, *212*, 285-295. https://doi.org/10.1016/j.atmosres.2018.05.029

Park, S. S., Kim, Y. J., & Fung, K. (2001). Characteristics of PM2.5 carbonaceous aerosol in the Sihwa industrial area, Korea. *Atmospheric Environment*, *35*(4), 657-665. https://doi.org/10.1016/S1352-2310(00)00357-5

Giones, F., & Brem, A. (2017). From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry. *Business Horizons*, *60*(6), 875-884. https://doi.org/10.1016/j.bushor.2017.08.001

Nwaogu, J. M., Yang, Y., Chan, A. P. C., & Chi, H.-l. (2023). Application of drones in the architecture, engineering, and construction (AEC) industry. *Automation in Construction*, *150*, 104827. https://doi.org/10.1016/j.autcon.2023.104827

Moreno-Jacobo, D., Toledo-Nin, G., Ochoa-Zezzatti, A., Torres, V., & Estrada-Otero, F. (2021). Evaluation of Drones for Inspection and Control in Industry 4.0. En A. Ochoa-Zezzatti, D. Oliva, & A. Juan Perez (Eds.), *Technological and Industrial Applications Associated with Intelligent Logistics*. Lecture Notes in Intelligent Transportation and Infrastructure. Springer, Cham. https://doi.org/10.1007/978-3-030-68655-0_29

Marinov, M. B., Topalov, I., Ganev, B., Gieva, E., & Galabov, V. (2019). UAVs Based Particulate Matter Pollution Monitoring. En *2019 IEEE XXVIII International Scientific Conference Electronics (ET)* (pp. 1-4). Sozopol, Bulgaria. doi: 10.1109/ET.2019.8878586

Li, X.-B., Wang, D.-S., Lu, Q.-C., Peng, Z.-R., & Wang, Z.-Y. (2018). Investigating vertical distribution patterns of lower tropospheric PM2.5 using unmanned aerial vehicle measurements. *Atmospheric Environment*, *173*, 62-71. https://doi.org/10.1016/j.atmosenv.2017.11.009

RECURSOS EN LINEA

MAVLink. (s.f.). Recuperado de https://mavlink.io/en/

López, J. P. (s.f.). MAVLink and Arduino step-by-step. Recuperado de https://discuss.ardupilot.org/t/mavlink-and-arduino-step-by-step/25566

Winkelmann, J. (s.f.). Sensirion/arduino-sps. Recuperado de https://github.com/Sensirion/arduino-sps

ANEXO 1. SCRIPTS

En este anexo se encuentran todos los códigos utilizados en el proyecto, empezando por los códigos para la placa de Arduino y el módulo NodeMCU, y mostrando a continuación los scripts de MATLAB utilizados para generar los mapas de calor y las gráficas.

CÓDIGO PARA LA PLACA ARDUINO

```
#include <sps30.h>
#include <SoftwareSerial.h>
#include <mavlink.h>

#define RXpin 2
#define TXpin 3
SoftwareSerial Mavlink(RXpin,TXpin);
SoftwareSerial NodeMCU(5,6);

//MAVLink
unsigned long previousMillisMAVLink = 0;          //Almacena la última
vez que se transmitió y se escucho
unsigned long next_interval_MAVLink = 1000;       //Próximo intervalo
const int num_hbs = 60;                           //Número de látidos a
esperar (60 equivalen a 1 minuto)
int num_hbs_pasados = num_hbs;
int sysid = 1; int compid = 158;
//!MAVLink

//Pixhawk
int type = MAV_TYPE_QUADROTOR;

uint8_t system_type = MAV_TYPE_GENERIC;          //Definición de tipo de
sistema
uint8_t autopilot_type = MAV_AUTOPILOT_INVALID;

uint8_t system_mode = MAV_MODE_PREFLIGHT;        //Arranque
uint32_t custom_mode = 0;
uint8_t system_state = MAV_STATE_STANDBY;

mavlink_message_t msg;                           //Inicializando los
buffers
uint8_t buf[MAVLINK_MAX_PACKET_LEN];
//!Pixhawk

const int Led = 9;
```

```
bool DatosRecibidos;
float latitud, longitud, altura;

void setup(){

  pinMode(Led, OUTPUT);
  Serial.begin(57600);
  delay(1000);

//SPS30
  int16_t ret;
  uint8_t auto_clean_days = 4;
  uint32_t auto_clean;

  sensirion_i2c_init();

  while (sps30_probe() != 0){
    Serial.print(F("Error en el sondeo del sensor SPS30\n"));
    delay(500);
  }

  Serial.print(F("Éxito en el sondeo del sensor SPS30\n"));

  ret=sps30_set_fan_auto_cleaning_interval_days(auto_clean_days);
  if (ret){
    Serial.print(F("Error ajustando el intervalo de limpieza automática:
"));
    Serial.print(ret);
  }

  ret=sps30_start_measurement();
  if (ret < 0){
  Serial.print(F("Error empezando mediciones\n"));
  }

  Serial.print(F("Inicio de mediciones\n"));
  delay(1000);
//!SPS30
}

void loop() {

//Conexión Mavlink
    DatosRecibidos = false;
    Mavlink.begin(57600);
    delay(100);
  //Empaquetado del mensaje
```

```
  mavlink_msg_heartbeat_pack(1,0, &msg, type, autopilot_type,
system_mode, custom_mode, system_state);

  uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);  //Copiado de
mensaje en buffer

  unsigned long currentMillisMAVLink = millis();
  if (currentMillisMAVLink - previousMillisMAVLink >=
next_interval_MAVLink){
    //Variables de tiempo
    previousMillisMAVLink = currentMillisMAVLink;

    Mavlink.write(buf, len);

    num_hbs_pasados++;
    if(num_hbs_pasados >= num_hbs){
      Mav_Request_Data();              //Petición de datos de Pixhawk
      num_hbs_pasados = 0;
    }
  }

  comm_receive();    //Comprobación de buffer de recepción
//!Conexión Mavlink

//Lectura de partículas y envio a NODEMCU

if (DatosRecibidos){
//Comienzo de instrucciones

//Mediciones SPS30
    struct sps30_measurement mediciones;
    char serial[SPS30_MAX_SERIAL_LEN];
    uint16_t data_ready;
    int16_t ret;

  do{
    ret = sps30_read_data_ready(&data_ready);
    if(ret<0){
      Serial.print(F("Error leyendo medición: "));
      Serial.println(ret);
    } else if (!data_ready)
      Serial.print(F("No hay nuevas mediciones disponibles\n"));
      else
       break;
       delay(100);
  }while(1);

  ret = sps30_read_measurement(&mediciones);
```

```
  if(ret<0){
    Serial.print(F("Error leyendo mediciones\n"));
  }else {
    Serial.print("PM  1.0: "); Serial.println(mediciones.mc_1p0);
    Serial.print("PM  2.5: "); Serial.println(mediciones.mc_2p5);
    Serial.print("PM  4.0: "); Serial.println(mediciones.mc_4p0);
    Serial.print("PM 10.0: "); Serial.println(mediciones.mc_10p0);

    Serial.print("NC  0.5: "); Serial.println(mediciones.nc_0p5);
    Serial.print("NC  1.0: "); Serial.println(mediciones.nc_1p0);
    Serial.print("NC  2.5: "); Serial.println(mediciones.nc_2p5);
    Serial.print("NC  4.0: "); Serial.println(mediciones.nc_4p0);
    Serial.print("NC 10.0: "); Serial.println(mediciones.nc_10p0);

    Serial.print("Typical particle size: ");
Serial.println(mediciones.typical_particle_size);
    Serial.println();
  }
 //!Mediciones SPS30

//Envio de datos NodeMCU
 Mavlink.end();
 delay(100);
 NodeMCU.begin(115200);
 delay(100);

 NodeMCU.print(String(latitud,4)); NodeMCU.print(" ");
 NodeMCU.print(String(longitud,4)); NodeMCU.print(" ");
 NodeMCU.print(String(altura,3)); NodeMCU.print(" ");

 NodeMCU.print(String(mediciones.mc_1p0,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.mc_2p5,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.mc_4p0,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.mc_10p0,4)); NodeMCU.print(" ");

 NodeMCU.print(String(mediciones.nc_0p5,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.nc_1p0,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.nc_2p5,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.nc_4p0,4)); NodeMCU.print(" ");
 NodeMCU.print(String(mediciones.nc_10p0,4)); NodeMCU.print(" ");

 NodeMCU.print(String(mediciones.typical_particle_size,4));
NodeMCU.print(" ");

 NodeMCU.print("\n"); //Salto de linea para indicar final de transmisión

 NodeMCU.end();
 delay(100);
```

```
//!Envio de datos NodeMCU
 digitalWrite(Led, HIGH);
 delay(1000);
 digitalWrite(Led, LOW);

}
 delay(100);
//!Lectura de partículas y envio a NODEMCU

//delay(2500);  //Delay opcional para ajustar velocidad de muestreo
}

void Mav_Request_Data(){

  mavlink_message_t msg;
  uint8_t buf[MAVLINK_MAX_PACKET_LEN];

  const int maxStreams = 1;
  const uint8_t MAVStreams[maxStreams] = {MAV_DATA_STREAM_POSITION};
  const uint16_t MAVRates[maxStreams] = {0x05};

  for(int i=0; i<maxStreams; i++){

    mavlink_msg_request_data_stream_pack(2, 200, &msg, 1, 0,
MAVStreams[i], MAVRates[i], 1);
    uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);
    Mavlink.write(buf, len);

  }
delay(1000);
}

void comm_receive(){

  mavlink_message_t msg;
  mavlink_status_t status;

  while(Mavlink.available()>0){

    uint8_t c = Mavlink.read();

    if(mavlink_parse_char(MAVLINK_COMM_0, c, &msg,
&status)){      //Obtención de nuevo mensaje

      //Manejo de mensaje
      switch(msg.msgid){

        case MAVLINK_MSG_ID_HEARTBEAT: // #0: Heartbeat
```

```
        {
          //Serial.print("HEARTBEAT RECEIVED!\n");     //Activar esta
linea para comprobar recepción de Heartbeats
        }
        break;

        case MAVLINK_MSG_ID_GLOBAL_POSITION_INT:     // #33: Posición
global

          mavlink_global_position_int_t posicion;
          mavlink_msg_global_position_int_decode(&msg, &posicion);

         latitud = posicion.lat / 10000000.0;   //Los datos de longitud
y latitud se reciben con exponente 7
         longitud = posicion.lon / 10000000.0;
         altura = posicion.alt / 1000.0;        //La altura se recibe
con exponente 3

          Serial.print("Latitud: ");
Serial.println(String(latitud,4));
          Serial.print("Longitud: ");
Serial.println(String(longitud,4));
          Serial.print("Altura: "); Serial.println(String(altura,3));

          DatosRecibidos = true;
        break;

        default:
        break;
      }
    }
  }
}
```

CÓDIGO PARA EL MÓDULO NODEMCU

```
#include <SoftwareSerial.h>
SoftwareSerial NodeMCU(D7,D8); //RX(D7) TX(D8)

void setup() {

 Serial.begin(9600);    //Monitor Serial NodeMCU
 NodeMCU.begin(115200); //Conexión Serial con Arduino
}

void loop() {

 if(NodeMCU.available()){
   String mediciones = NodeMCU.readStringUntil('\n');
   Serial.println( mediciones );
   //Latitud Longitud Altura PM1.0 PM2.5 PM4.0 PM10.0 NC0.5 NC1.0 NC2.5
NC4.0 NC10.0 Media
 }

}
```

SCRIPT DE MATLAB PARA LA GENERACIÓN DE MAPAS DE CALOR

```
% Lectura de datos, especificar la ruta al archivo
ruta_archivo =
"C:\Users\joses\OneDrive\Documentos\MATLAB\TFG_DEF_DATA.txt";
data = readmatrix(ruta_archivo);
x = data(:, 2); %Longitud
y = data(:, 1); %Latitud
z = data(:, 4); % Introducir número de columna a leer(4-13)

res = 100; %Resolución
xi = linspace(min(x), max(x), res);
yi = linspace(min(y), max(y), res);
[X, Y] = meshgrid(xi, yi);

Z = griddata(x, y, z, X, Y, 'linear'); %Interpolación

% Generación de mapa de calor
imagesc(xi, yi, Z);
colormap jet;
colorbar;
xlabel('Longitud');
ylabel('Latitud');
title('Concentración en masa PM1.0 (µg/m3)');

axis equal; %Ajustes de presentación
axis tight;
set(gca, 'YDir', 'normal'); %Inversión eje Y
```

SCRIPT DE MATLAB PARA LA GENERACIÓN DE GRÁFICAS RESPECTO AL TIEMPO

```
%Lectura de datos, especificar la ruta al archivo
ruta_archivo =
"C:\Users\joses\OneDrive\Documentos\MATLAB\TFG_DEF_DATA.txt";
data = readmatrix(ruta_archivo);
[num_filas, num_columnas] = size(data);


tiempo_maximo = 90;  %Ajustar tiempo del experimento
tiempo = linspace(0, tiempo_maximo, num_filas);  % Eje x equidistante

columnas_a_mostrar = 4:7;
datos_a_mostrar = data(:, columnas_a_mostrar);
colores = {'r', 'g', 'b', 'y'};

% Gráficas
figure; hold on;
for i = 1:numel(columnas_a_mostrar)
    plot(tiempo, datos_a_mostrar(:, i), 'Color', colores{i});
end

xlabel('Tiempo (minutos)', 'FontName', 'Calibri', 'FontSize', 24);
ylabel('Concentración en masa (µg/m3)', 'FontName', 'Calibri',
'FontSize', 24);
title('Concentración en masa de micropartículas respecto al tiempo',
'FontName', 'Calibri', 'FontSize', 24);

% Leyenda
hLegend = legend('PM1.0', 'PM2.5', 'PM4.0', 'PM10.0');
set(hLegend, 'FontSize', 24);

% Ajuste de ejes
set(gca, 'FontSize', 18);

hold off;
```

SCRIPT DE MATLAB PARA LA GENERACIÓN DE GRÁFICAS CON RESPECTO A LA ALTURA

```
%Lectura de datos, especificar la ruta al archivo
ruta_archivo = "C:\Users\joses\OneDrive\Documentos\MATLAB\lecturas medias.txt";
data = readmatrix(ruta_archivo);

% Seleccionar las columnas a leer
datos_a_mostrar = data(:, 9:12);

% Valores fijos de altura
valores_x = [0, 4.75, 8.55, 12.35];

colores = {'r', 'g', 'b', 'y'};

% Gráficas
figure; hold on;
for i = 1:size(datos_a_mostrar, 2)
    plot(valores_x, datos_a_mostrar(:, i), 'Color', colores{i});
end

xlabel('Altura (metros)', 'FontName', 'Calibri', 'FontSize', 24);
ylabel('Concentración en número (partículas/cm3)', 'FontName', 'Calibri',
'FontSize', 24);
title('Concentración en número respecto a la altura', 'FontName', 'Calibri',
'FontSize', 24);

% Leyenda
hLegend = legend('PM1.0', 'PM2.5', 'PM4.0', 'PM10.0');
set(hLegend, 'FontSize', 24);

% Ajuste de ejes
set(gca, 'FontSize', 24);

% Ampliación del eje y
ylim([0, max(datos_a_mostrar(:)) * 1.2]);

hold off;
```

ANEXO 2. PRESUPUESTO

En este anexo se realiza el presupuesto del proyecto realizado. En este se reflejan los diferentes costes del sistema de monitorización de partículas diseñado. En este caso no se considerará el drone en el sistema ya que el objetivo es diseñar un módulo que sea compatible con cualquier drone que utilice una controladora Pixhawk. Se analizarán los recursos necesarios tanto de software como de hardware y se proporcionará un presupuesto final.

RECURSOS DE SOFTWARE

Debido a que se han utilizado solo plataformas de código abierto como Arduino, Pixhawk, Mission Planner, etc. este proyecto no tiene gastos de software. Es decir, el coste es de cero euros.

RECURSOS DE HARDWARE

En la tabla siguiente se detallan los componentes necesarios para crear el módulo, así como sus costes unitarios y el coste total del proyecto.

Tabla A2.1: Presupuesto Hardware.

Componente	Valor unitario (€)
Arduino Uno Rev3	29.04
Arduino Shield Rev3	6.49
NodeMCU ESP 8266 MOD	7.99
Sensor Sensirion SPS30 (con conector)	38
BEC	7.50
Cables Arduino	11.99
Total (€)	101.05

PRESUPUESTO TOTAL

El presupuesto total del proyecto queda en ciento y un euros con cinco céntimos (101.05€). En esto se incluyen todos los gastos necesarios para realizarlo. No se incluyen en este herramientas o materiales que se espera que el usuario ya disponga. Se puede observar que se ha conseguido realizar un sistema de bastante calidad y muy económico.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS