Universitat de les Illes Balears

TREBALL FI DE GRAU

DESENVOLUPAMENT D'UN SISTEMA REMOT DE MESURA DE PARTÍCULES EN SUSPENSIÓ A L'AIRE

Patricia Bennàssar Llabrés

Grau de Física

Facultat de Ciències

Any Acadèmic 2021-22

DESENVOLUPAMENT D'UN SISTEMA REMOT DE MESURA DE PARTÍCULES EN SUSPENSIÓ A L'AIRE

Patricia Bennàssar Llabrés

Treball de Fi de Grau

Facultat de Ciències

Universitat de les Illes Balears

Any Acadèmic 2021-22

Paraules clau del treball:

partícules, OPC, sensor, Nodemcu, programació.

Nom Tutor/Tutora del Treball Rodrigo Picos Gayà

Nom Tutor/Tutora (si escau)

Resum

Donada la importància de la salut i del medi ambient, és important tenir en compte els factors que poden alterar-ho i prendre mesures per a millorar aquests aspectes perjudicials. En el cas d'aquest TFG, ens hem centrat en programar diferents sensors amb l'objectiu de poder obtenir dades de les concentracions del partícules que trobem a l'aire i que afecten directament a la nostra vida diària. Si podem facilitar el control d'aquestes partícules i trobar la causa de concentracions altes quan es donen, podrem adaptar i/o crear mesures que les facin disminuir o prevenir i arribar a una millora de la salut de tots i del medi que ens envolta.

Resumen

Dada la importancia de la salud y del medio ambiente, es importante tener en cuenta los factores que pueden alterarlo y tomar medidas para mejorar estos aspectos perjudiciales. En el caso de este TFG, nos hemos centrado en programar diferentes sensores con el objetivo de poder obtener datos de las concentraciones de partículas que encontramos en el aire y que afectan directamente a nuestra vida diaria. Si Podemos facilitar el control de estas partículas I encontrar la causa de concentraciones altas cuando se dan, Podemos adaptar y/o crear medidas que las hagan disminuir o prevenir y llegar a una mejora de la salud de todos y del medio que nos rodea.

Summary

Given the importance of health and the environment, it is important to consider the factors that can alter it and take actions to improve these harmful aspects. In the case of this TFG, we have focused on programming different sensors with the aim of being able to obtain data on the concentrations of particles that we find in the air and that affect our daily lives directly. If we can help to control these particles and find the cause of high concentrations when we measure them, we can adapt and/or create actions that make them decrease or prevent and improve the health of everyone and the environment around us.

Índex

1. INTRODUCCIÓ. 2

1.1. PARTÍCULES A L'AIRE. 2

1.2. DISPERSIÓ DE PARTÍCULES. 3

1.3. NORMATIVA DE SALUT. 4

1.4. COMPONENTS: NODEMCU, OPC I BMP085. 7

2. MUNTATGE I PROGRAMACIÓ. 11

2.1. MUNTATGE EXPERIMENTAL. 11

2.2. PROGRAMACIÓ I IMPLEMENTACIÓ DELS SENSORS. 12

2.3. FUNCIONAMENT I LOCALITZACIÓ. 13

3. ANÀLISI I RESULTATS. 16

4. CONCLUSIONS. 23

5. REFERÈNCIES. 24

ANNEX I: CODI ARDUINO. 26

1. Introducció.

Per a desenvolupar un sistema remot de mesura de partícules a l'aire hem d'entendre la importància de les partícules que trobem a l'aire en les que centrarem el treball i les normatives de salut que hi ha vigents a diferents nivells governamentals (internacional, europeu, espanyol, balear). També analitzarem el funcionament i la utilitat dels components electrònics que emprarem per a implementar el nostre sistema de mesura de partícules PM. Aquest es basarà en una placa tipus Arduino per controlar la resta de perifèrics, com són un sensor de mesura de partícules i un de pressió i temperatura, i enviar les dades via wifi a una web especialitzada en projectes IoT com és Thingspeak per a poder processar-les i analitzar-les posteriorment.

1.1. Partícules a l'aire.

L'atmosfera està constituïda per aigua (vapor d'aigua) i l'atmosfera seca, que està constituïda per gasos, sals i altres substàncies. En el nostre treball estam interessats en mesurar algunes de les partícules en suspensió que causen la contaminació atmosfèrica. Aquestes partícules alteren la composició natural de l'atmosfera, sigui per causes naturals o per l'acció de l'home (causes antropogèniques), i afecten sobretot a la salut, als ecosistemes i al clima, d'aquí la seva importància.

Si ens centrem en classificar les partícules pel seu tamany tenim diferents dominis (moda) relacionats amb el mecanisme de formació i el tamany. Hem de tenir present que, la granulometria i la composició química de les partícules sòl estar relacionada directament amb el focus emissor [1].

Com podem veure a la figura 1.1, separarem les partícules en PM1, PM2.5 i PM10.

- PM1: son les partícules més perilloses ja que, per la seva mida inferior a 1µm, poden entrar fins al flux sanguini i inclouen virus, bactèries, aerosols atmosfèrics, etc.
- PM2.5: tant per qualitat d'aire com per epidemiologia es tenen en compte les partícules amb un diàmetre superior a 2.5 µm. Aquestes partícules poden ser inspirades fins a acumular-se i recobrir els pulmons o donar lloc a problemes a pell i ulls. Solen ser partícules de combustió, composts orgànics, metalls, etc. que son partícules secundàries formades a l'atmosfera.

- PM10: les partícules de major tamany, menors a 10 µm, poden acumular-se i obstruir conductes respiratoris, essent pol·len, pols, etc. Principalment partícules primàries emeses a l'atmosfera.

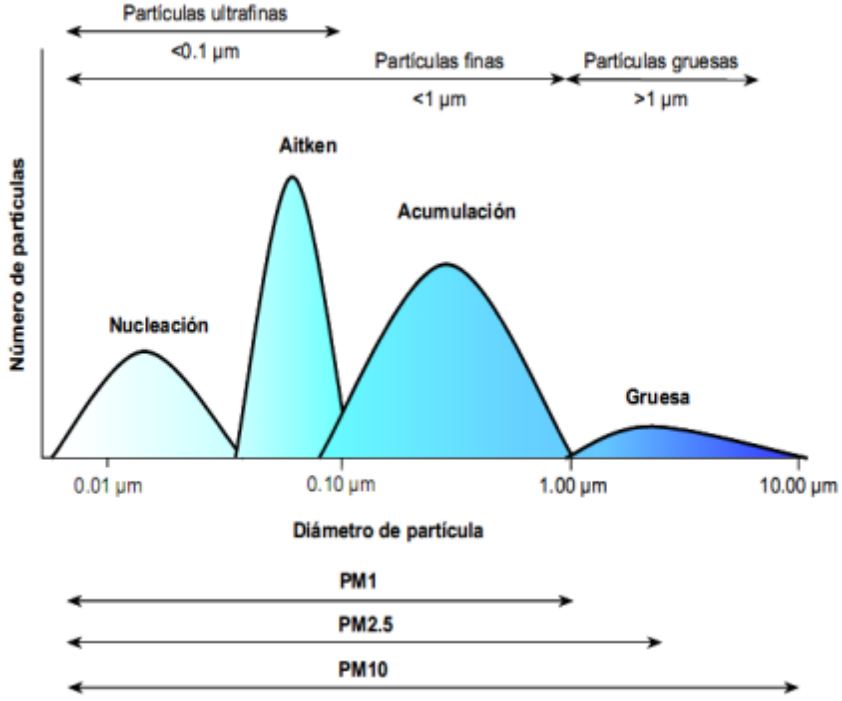

Fig. 1.1. Distribucions modals segons el diàmetre de partícula, segons Querol et al (2001).

1.2. Dispersió de partícules.

Una vegada entès la importància de les partícules de l'aire, hem de saber que hi ha diferents processos de dispersió d'aquestes deguts a factors atmosfèrics com poden ser la capacitat de l'atmosfera per dispersar les partícules, la neteja natural que fan les precipitacions o els núvols en diluir o dipositar els materials, la deposició seca o inclús la reactivitat química de les partícules.

Els mecanismes de dispersió més rellevants son:

- Vent i turbulència: El vent desplaça les masses d'aire segons els gradients de pressió atmosfèrica. Això, depenent del terreny, la direcció i la velocitat del vent fa que les partícules es dissipin més ràpid/lent i més lluny/prop de la font emissora. També afecta en aquest aspecte la turbulència, que transportarà major quantitat de partícules quan la turbulència sigui major.
- Humitat atmosfèrica: Quan trobem grans quantitats de vapor d'aigua a l'atmosfera, la humitat es major i, per tant, les partícules s'adhereixen a les gotes i són transportades. En aquests casos obtenim una acumulació de fum o pols com en el cas de calimes i pols sahrauí en suspensió.

- Inversió tèrmica: En el cas de trobar-se superposades masses d'aire diferents (fronts freds i càlids) o un massa homogènia alterada com és el refredament del sòl a la nit, ens trobem a partir d'una certa alçada que la temperatura augmentarà amb l'alçada i no al revés. Aquesta franja d'inversió tèrmica que es crea impedeix la difusió de les partícules.
- Precipitacions: Quan es donen precipitacions de qualsevol tipus (pluja, neu, calabruix) les partícules es van absorbint a elles. Això fa que les partícules es dissolguin i es dipositin a terra, on poden quedar o ser transportats per altres factors.

1.3. Normativa de salut.

Donada la importància que representa pel medi ambient i la salut el nivell d'acumulació de partícules a l'aire, hi ha establides una sèrie de normatives a diferents nivells governamentals, des de directrius de la OMS fins a normatives europees, espanyoles i balears. Ens centrarem únicament en les que afectin als valors límits de PM.

A nivell internacional, trobem unes directrius mundials de la OMS sobre la qualitat de l'aire del 2005 [2] on, tenint en compte els efectes perjudicials dels materials particulats, es recomanen uns valors límits de certs contaminants atmosfèrics resumits en la següent figura.

Contaminante	Tiempo promedio	Meta intermedia				Nivel de las directrices sobre la calidad del aire
		1	2	3	4	
$MP_{2,5}$, µg/m³	Anual	35	25	15	10	5
	24 horas[a]	75	50	37,5	25	15
MP_{10}, µg/m³	Anual	70	50	30	20	15
	24 horas[a]	150	100	75	50	45
O_3, µg/m³	Temporada alta[b]	100	70	–	–	60
	8 horas[a]	160	120	–	–	100
NO_2, µg/m³	Anual	40	30	20	–	10
	24 horas[a]	120	50	–	–	25
SO_2, µg/m³	24 horas[a]	125	50	–	–	40
CO, mg/m³	24 horas[a]	7	–	–	–	4

[a] Percentil 99 (es decir, 3-4 días de superación por año).
[b] Promedio de las concentraciones máximas diarias de O_3 (medias octohorarias) en los seis meses consecutivos con la concentración media móvil de O_3 más alta.

Fig. 1.2. Nivells recomanats de les directrius de la OMS sobre qualitat de l'aire.

A nivell europeu, el Parlament Europeu i el Consell de la Unió Europea publicaren la directiva 2008/50/CE relativa a la qualitat de l'aire ambient i a aconseguir una atmosfera més neta a Europa [3]. Els seus objectius son establir objectius per evitar, prevenir i reduir els efectes nocius per la salut humana i el medi ambient, establir mètodes i criteris comuns per avaluar la qualitat de l'aire ambient, fer un seguiment de l'evolució de les millores aplicades i fomentar la cooperació entre els països per reduir la contaminació atmosfèrica, entre altres. En les figures 1.3 i 1.4 tenim els valors límits de PM2.5 i PM10 respectivament, el marge de tolerància permès i la data fitxada per aplicar aquests límits. Degut a la importància i la major dificultat de control del PM2.5, podem veure com es van imposar límits diferents en dues fases amb l'objectiu de reduir progressivament els nivells de PM2.5 fins a valors més restrictius.

Período medio	Valor límite	Margen de tolerancia	Fecha en que debe alcanzarse el valor límite
FASE 1			
Año civil	25 µg/m^3	20 % el 11 de junio de 2008, que se reducirá el 1 de enero siguiente y, en lo sucesivo, cada 12 meses, en porcentajes idénticos anuales hasta alcanzar un 0 % el 1 de enero de 2015	1 de enero de 2015
FASE 2 (1)			
Año civil	20 µg/m^3		1 de enero de 2020

(1) Fase 2: valor límite indicativo que será revisado por la Comisión en 2013 a la luz de informaciones suplementarias sobre los efectos sobre la salud y el medio ambiente, la viabilidad técnica y la experiencia obtenida con el valor objetivo en los Estados miembros.

Fig.1.3. Valors límit per a la protecció de la salut humana per a PM2.5.

Período medio	Valor límite	Margen de tolerancia	Fecha en la que debe alcanzarse
PM_{10}			
1 día	50 µg/m^3, que no podrá superarse más de 35 veces por año civil	50 %	(1)
Año civil	40 µg/m^3	20 %	(1)

(1) Ya en vigor desde el 1 de enero de 2005.

Fig. 1.4. Valors límit per a la protección de la salut humana per a PM10.

Posteriorment, la Directiva 2015/1480 [4] canvià mètodes de referència, validació de dades i ubicació dels punts de mostreig per avaluar la qualitat de l'aire, però sense modificar els límits establerts.

Com a punt a tenir en compte, amb els criteris de la Unió Europea, es fa distinció dels PM2.5 i PM10 en general i també es fa una distinció d'altres contaminants segons els PM. En el cas de PM2.5 s'indiquen especiacions com sulfats, clorurs, carboni, magnesi, etc. Per PM10 trobem metalls pesats i hidrocarburs aromàtics entre altres.

En quant a normativa espanyola, trobem la Llei 34/2007 [5], que estableix unes bases per prevenir, vigilar i reduir la contaminació atmosfèrica, com per exemple limitar o controlar activitats potencialment contaminants per l'atmosfera.

En un decret posterior, RD 102/2011 [6], s'estableixen mètodes i normes de com mesurar els contaminants atmosfèrics, entre els quals es troben el PM2.5 i PM10. Així com també uns valors límit, que coincideixen amb els valors i dates imposats per la directiva europea. Una de les mesures establides és l'obligatorietat a les comunitats autònomes i entitats locals de mesurar i avaluar, depenent la classificació del territori en quant a aglomeracions, la qualitat de l'aire segons els principis establerts.

A nivell de Balears no tenim cap modificació respecte a la normativa nacional, per tant el Govern de les Illes Balears es regeix pel RD 102/2011 i du a terme el control de la qualitat de l'aire mitjançant una xarxa d'estacions a punts d'interès o que, per obligatorietat de les seves condicions com a zona aglomerada (Palma Centre, Inca, etc) o amb activitats potencialment contaminadores de l'atmosfera o APCA (ports, aeroports, incineradores, centrals tèrmiques) [7].

Encara que el control de les estacions el faci el Govern, algunes de les estacions estan instal·lades per les empreses APCA segons la normativa. Es revisen i mantenen setmanalment, es fa una revisió quinzenal de les dades, es recalibren els aparells cada tres mesos i anualment es realitza una revisió completa.

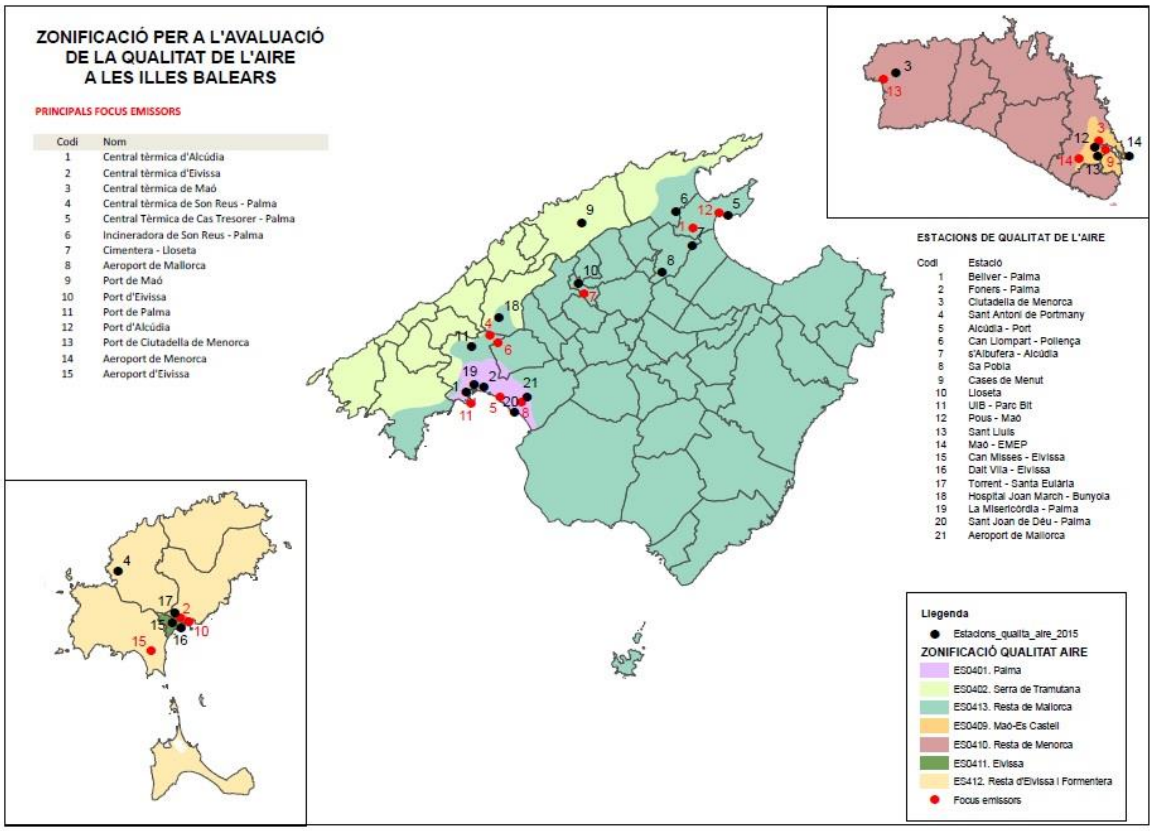

Fig. 1.5. Mapa de les estacions de control de qualitat de l'aire (punts negres) i els principals focus emisors (punts vermells).

1.4. Components: Nodemcu, OPC i BMP085.

Per a dur a terme el nostre sistema de monitorització hem emprat una placa de desenvolupament tipus Arduino, un sensor de mesura de partícules OPC (Optical Particle Counter) i un sensor BMP085 de pressió i temperatura, així com una web especial per a projectes IoT com és Thingspeak.

- Nodemcu ESP-12E: Placa de desenvolupament basada en el chip ESP8266 i amb el mòdul Wifi ESP-12E, que ens permet implementar projectes amb IoT (Internet of Things). És el component que s'encarrega de gestionar els sensors i les dades que s'obtenguin d'ells. En el nostre cas empram la placa de tercera generació, amb versió 1.0 V3.

  És una placa similar a Arduino UNO, és Open Source i per tant, compatible amb l'IDE d'Arduino (Integrated Development Environment). La placa disposa de 17 pins digitals i un d'analògic. Una de les diferències clares amb l'Arduino UNO és que proporciona un

voltatge de sortida de 3.3V, però gràcies al conversor serial CH340G que du inclòs i al pin Vusb reservat, que permet connectar un voltatge d'entrada i sortida igual al voltatge proporcionat pel micro-USB i podem aconseguir voltatges majors. L'altre funció rellevant és gràcies al chip ESP8266, que ens permet emprar Wifi sense haver d'implementar un mòdul addicional extern. Un dels protocols que empra el Nodemcu és l'I2C (Inter-Integrated Circuit), un bus de comunicacions en sèrie, el mestre i l'esclau envien dades pel mateix cable, controlat pel mestre. Amb aquest protocol la placa es pot comunicar amb la resta de components de sistema integrat mitjançant un SDA (línia de dades) i un SCL (línia de rellotge). L'altre protocol d'interès és SPI (Serial Peripheral Interface), que empra quatre línies: SCK (Serial Clock), MOSI (Master Output Slave Input), MISO (Master Input Slave Output) i SS (slave select). És un protocol senzill per controlar dispositius electrònics digitals a velocitats majors que el I2C.

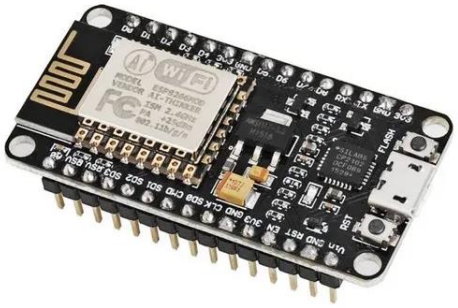

Fig. 1.6. Nodemcu 1.0 V3, Esp-12E

- OPC-N2 (Alphasense): Comptador de partícules òptic de la marca Alphasense [8].

  Els comptadors de partícules òptics extreuen aire a una càmera de dispersió oberta amb un ventilador. La llum emesa per un làser passa a través del flux d'aire i es dispersa quan colisiona amb partícules. A l'interior trobem un mirall el·líptic amb un focus dins la corrent d'aire i l'altre a un fotodíode. La llum dispersada per les partícules arriba al mirall enfocant al fotodíode i produeix un voltatge, que dependrà de la intensitat del raig dispersat. Aquesta intensitat està relacionada amb la concentració de partícules i el tamany d'aquestes.

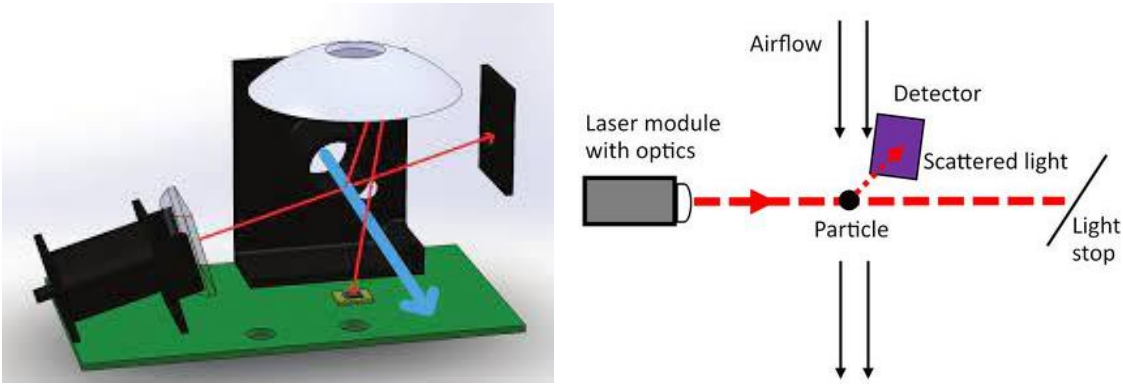

Fig. 1.7. Representació de l'interior de l'OPC i l'esquema del seu funcionament.

En el cas de l'OPC-N2 d'Alphasense, és un sensor de partícules que funciona a un voltatge aproximat de 5V, amb connector micro-USB que permet actualitzacions i que conté un software integrat que permet mesures de PM1, PM2.5 i PM10 i la representació d' histogrames del tamany dels PM a temps real. Es comunica mitjançant protocol SPI, mesura partícules de entre 0.38µm i 17 µm, permet agafar mesures en intervals des de 1s a 30s i fa les mesures en µg/m^3 [9].

Per poder programar-ho amb el Nodemcu s'empren els 6 fils de sortida dels que disposa: SS, MISO, MOSI, SCK, GND i 5V, el que permet integrar-ho en sistemes de més sensors amb facilitat.

Fig. 1.8. OPC-N2 Alphasense

- BMP085: Sensor de pressió atmosfèrica i temperatura d'alta precisió. Empra protocol I2C, per tant el connectem amb els pins SDA i SDL a més de terra i voltatge. Té una resolució de 0.06hPa, funciona amb un voltatge aproximat de 3V a 5V i té un temps de reacció de 7.5ms.

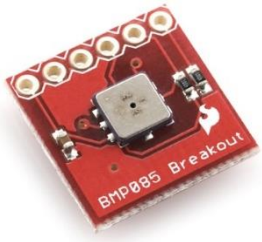

Fig. 1.9. Sensor BMP085

- Thingspeak: És una plataforma Open Source especialitzada en projectes IoT que permet recollir i emmagatzemar dades al núvol basant-se amb una API (Application Programming Interfaces) emprant el protocol HTTP sobre internet [10]. També es presenten moltes aplicacions que permeten visualitzar i analitzar les dades a la mateixa web i permet integrar dispositius Arduino com el que emprem.

  Per emprar-lo és necessari registrar-se i crear un canal amb 8 camps disponibles, que seran les variables que podrem recollir. Una vegada creat el canal, es generen dues claus API úniques, que seran les que permetran llegir les dades del canal (Read API Key) o escriure-les (Write API Key). Finalment, només és necessari carregar un codi de programació per Thingspeak i compatible amb el dispositiu que volem emprar, modifiquem les claus wifi, l'ID Channel i la Write API Key del nostre canal i podem visualitzar en temps real les dades a la pàgina web o a diverses Apps compatibles.

Fig. 1.10. Logotip de la plataforma Thingspeak.

2. Muntatge i programació.

2.1. Muntatge experimental.

El disseny experimental del nostre sistema de mesura és senzill. S'alimenta la Nodemcu amb el micro-USB. El BMP085 es connecta als pins de la Nodemcu de la següent manera:

Nodemcu	BMP085
3V	VCC
G	GND
D2	SDA
D1	SCL

Per connectar l'OPC a la placa ho fem amb els pins següents [11]:

Nodemcu	OPC-N2
VU	5V
G	GND
D8	SS
D7	MOSI
D6	MISO
D5	SCK

Degut a que l'OPC es connecta al pin VU com a pin d'alimentació, tot el sistema es pot alimentar mitjançant un micro-USB a la placa o directament a l'OPC, compartint així el voltatge in i facilitant-nos les connexions, ja que la placa te un voltatge de sortida de 3.3V, mentre que l'OPC necessita 5V d'entrada.

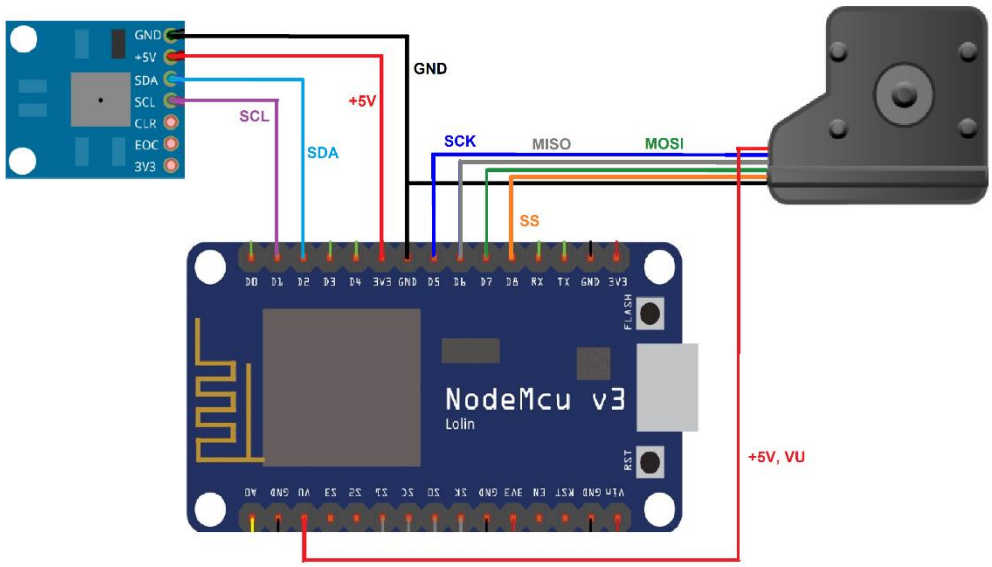

Fig. 2.1. Esquema de muntatge experimental.

Per altra banda, per a poder tenir tot el sistema a l'exterior i que estigui protegit de les condicions climatològiques, es va adaptar una caixa on es fitxà l'OPC amb perns, de manera que el tub d'entrada d'aire quedés a l'exterior per a una millor entrada del flux d'aire a mesurar. També es feren forats a tota la caixa per permetre la ventilació natural de l'aire interior i que les condicions a l'interior del nostre sistema siguin les més semblants possibles a les condicions exteriors.

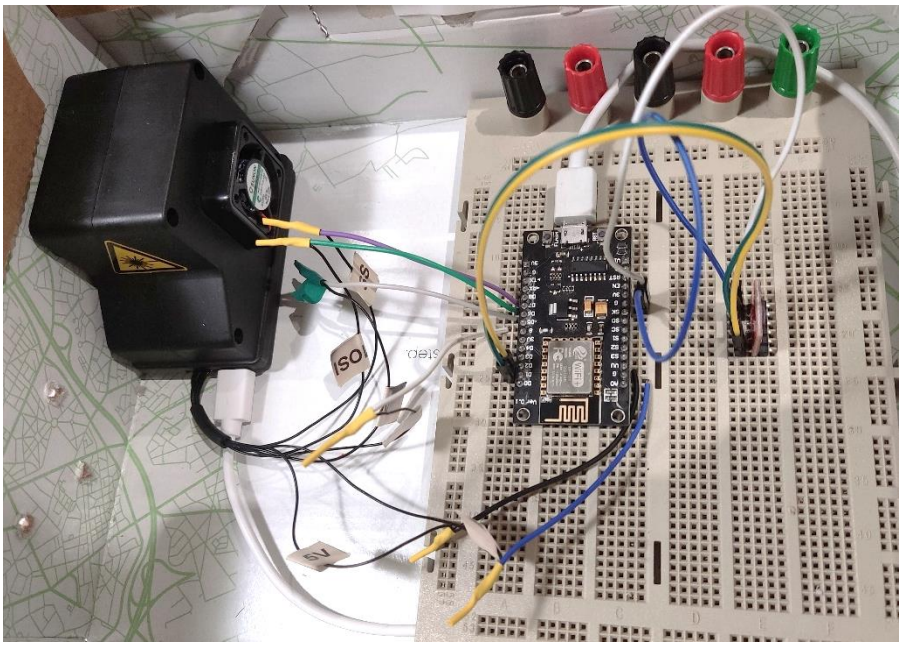

Fig. 2.2. Muntatge experimental del sistema de mesura complet amb la placa, l'OPC i el sensor de pressió.

2.2. Programació i implementació dels sensors.

Per programar el nostre sistema hem emprat l'IDE d'arduino (Integrated Development Environment), ja que és de codi obert i compatible amb la nostra placa. Una vegada connectada la Nodemcu a l'ordinador mitjançant l'USB, s'obre l'IDE i es carrega el codi.

En el nostre cas, primer es va implementar el sensor de pressió i temperatura mitjançant la llibreria Adafruit-BMP085 [12], es va comprovar que funcionava i que obteníem pel monitor valors de pressió i temperatura correctament.

El següent pas va ser carregar el codi de l'OPC obtingut entre dues llibreries diferents ([13], [14]) modificant els paràmetres de pin corresponent a la nostra placa. La llibreria està formada pel programa principal, que encén l'OPC, llegeix l'histograma i escriu al monitor les mesures dels PM, així com el període de mostreig. Una de les variables que

hem de canviar del programa és el període de temps entre mesures, per la qual cosa hem de tenir en compte l'interval de temps al qual està capacitat el sensor en prendre dades. També trobem a la llibreria l'opcn2.h, que defineix totes les variables (l'histograma, volums i densitats de partícules, el coeficient d'escala de guany, la tasa de flux de la mostra, l'interval de mostreig o si el ventilador i el làser han estat en repòs), mentre l'opcn2m.h defineix totes les funcions necessàries per fer funcionar el sensor (funcions per encendre el ventilador i el làser, per llegir les dades de PM, restablir l'histograma amb les dades recollides i llegir-lo, etc).

Finalment, mitjançant la llibreria de Thingspeak (previ registre a la web per poder obrir un canal on pujar les dades via wifi) [15], completem els paràmetres necessaris per connectar el sistema al wifi i els del canal creat a la web de Thingspeak i afegim a aquest codi els emprats anteriorment pel sensor BMP085 i l'OPC, ajustant el temps de mesura a 15s (codi final a l'Annex I).

2.3. Funcionament i localització.

Una vegada que el codi s'ha programat al sistema, cerquem un lloc exterior però cobert on poder agafar dades, connectem l'USB de la placa a la corrent i comprovem que s'agafen les dades correctament a través de la web Thingspeak. Des de la web podem veure una representació gràfica de les dades que es van mesurant en temps real (veure Fig.2.4) i en qualsevol moment podem descarregar un arxiu .csv amb totes les dades obtingudes per a analitzar-les posteriorment (com veurem més endavant, amb Matlab).

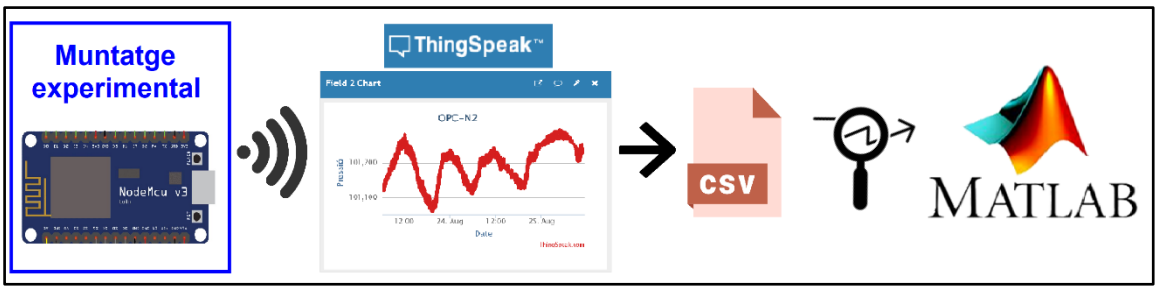

Fig. 2. 3. Esquema de funcionament del sistema remot que emprarem desde muntatge experimental fins anàlisi de dades.

Degut a les condicions dels llocs on es podia col·locar el sistema (aparells de ventilació propers, zones d'activitats que afecten a la mesura de partícules com moviment de tèxtils, llocs massa exposats a les condicions climàtiques, etc), es va decidir prendre mesures en dos llocs diferents. El primer lloc estava situat dalt d'un armari davall una porxada d'uralita i amb diferents aparells d'aire condicionat propers (preveiem un

increment de la temperatura i l'acumulació i moviment no natural de partícules) (veure Fig.2.5). El segon lloc era una finestra a 1m d'alçada, amb plantes properes i amb bastant de moviment humà proper (esperàvem disminució de temperatura però pics de partícules en certs moments d'activitats) (veure Fig.2.6). Per al nostre treball agafàrem dades a la localització 1 des de dia 10 d'Agost de 2022 fins dia 18 d'Agost de 2022. Per la localització 2 tenim dades des de dia 18 d'Agost de 2022 fins dia 24 d'Agost de 2022.

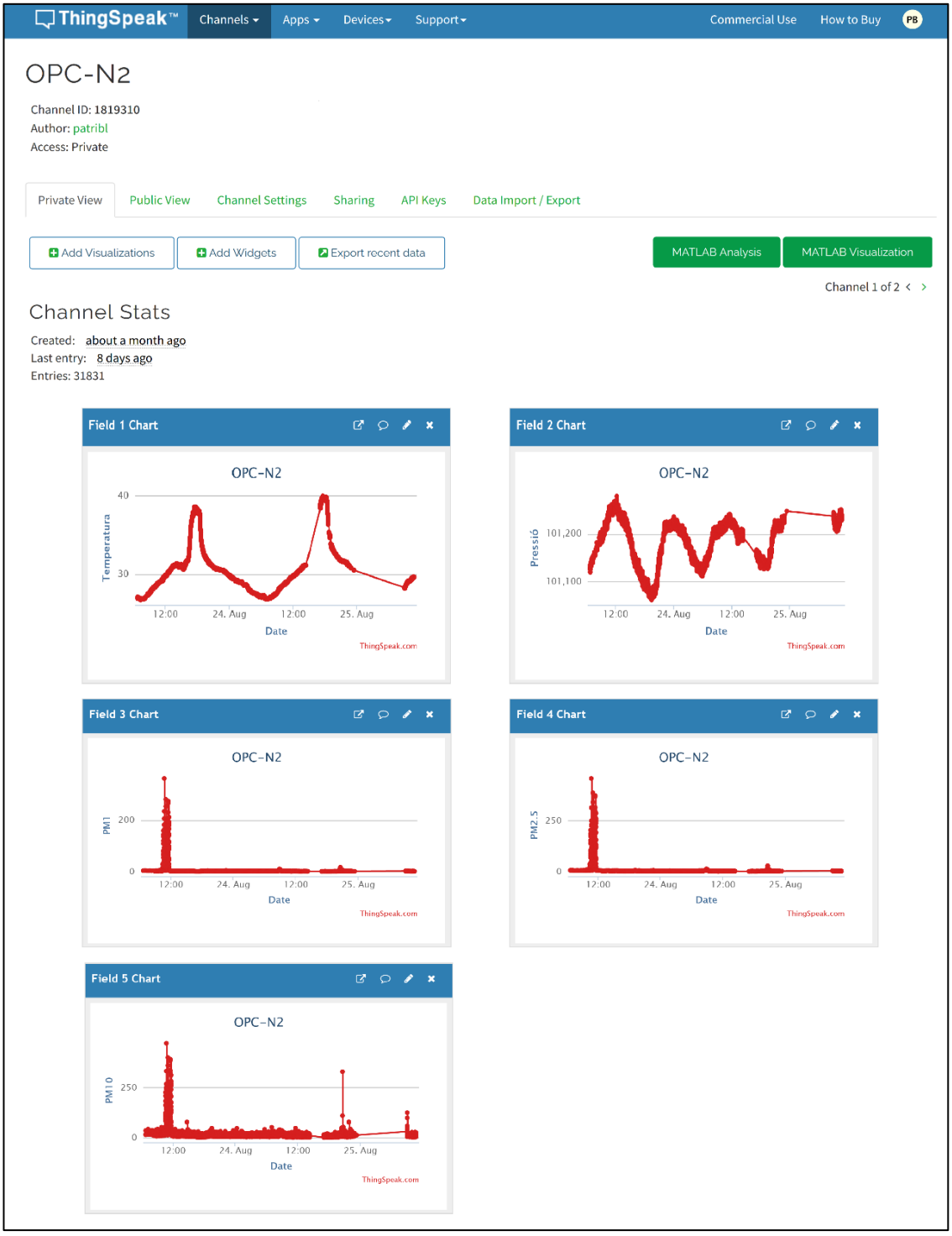

Fig. 2.4. Vista del canal creat a Thingspeak durant la recollida de dades.

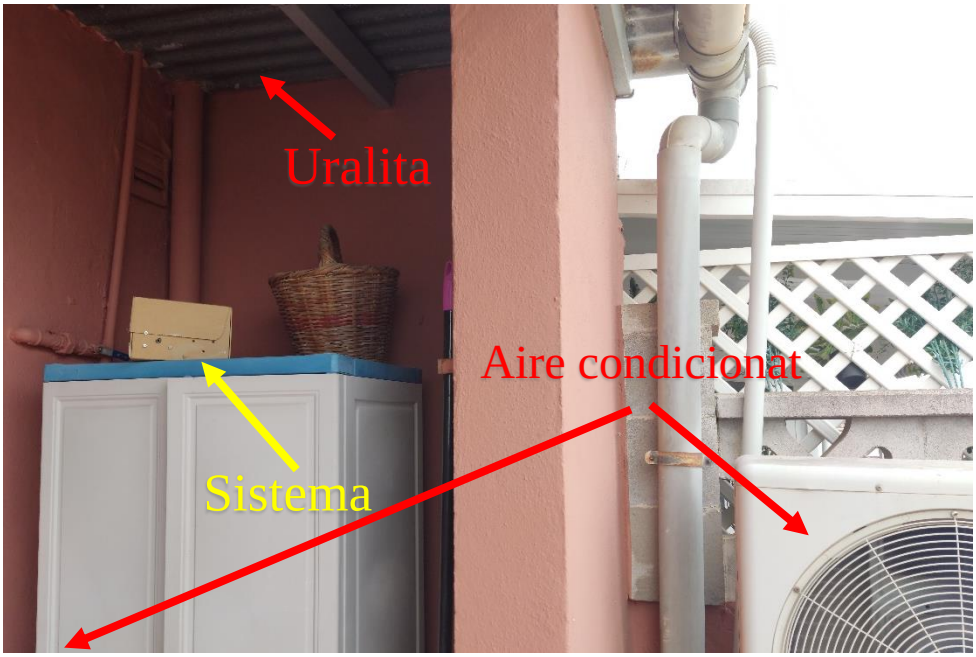

Fig. 2.5. Primera localització del sistema on s'agafen mesures.

Fig. 2.6. Segona localització del sistema on s'agafen mesures.

3. Anàlisi i resultats.

Una vegada han passat els dies que volem agafar les mesures a les dues localitzacions, descarreguem les dades del nostre canal Thingspeak [10] en format .csv i també descarreguem les dades registrades de l'estació de mesura de l'AEMET a l'aeroport de Palma [16], que es troba aproximadament a 2'5km d'on fem les nostres mesures, de manera que aquestes dades ens serviran de referència per comparar amb les nostres. Tot l'anàlisi i representació de dades que realitzem a partir d'ara es farà amb Matlab, que és un programa de computació numèrica que ens permetrà manipular les dades i representar-les de la manera que ens interessi més.

Representem les dades obtingudes als dos llocs de PM1, PM2.5 i PM10. Per a facilitar l'observació de dades, representem les dades dels dos llocs a l'eix horitzontal temporal separant-los amb una línia rosa:

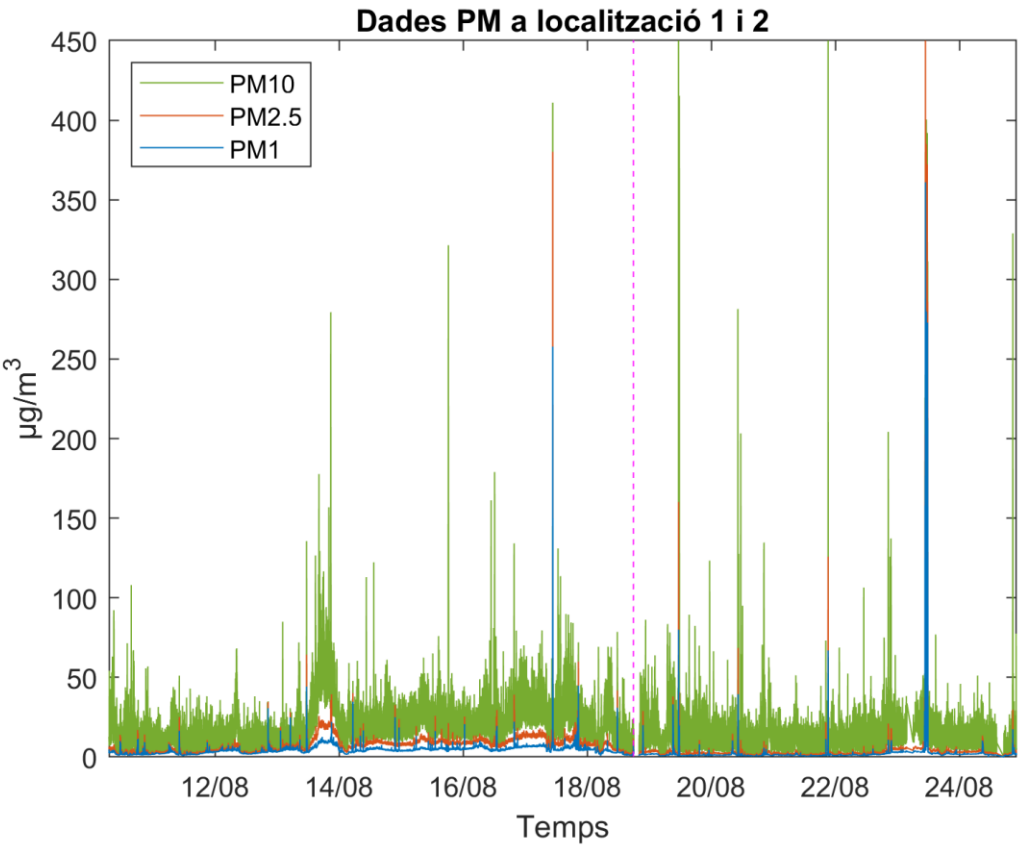

Fig. 3. 1. Representació dels PM mesurats a les dues localitzacions.

Podem veure com els PM1 i PM2.5 tenen valors més baixos i clars, mentre els PM10 presenten molt de renou a les dades i els valors son significativament majors. Els pics majors que observem on tenim altes concentracions dels 3 tamanys de PM, els dies 17, 19, 22 i 23 corresponen a episodis d'activitat antropogènica verificats, com per exemple

moments de rega abundant amb manguera o pulverització d'esprais de diferents productes. En el cas dels pics majors d'aproximadament 80µg/m^3 de PM10, vam observar que es devien a altres activitats antropogèniques, però que afectaven la mesura significativament, com per exemple la posada en marxa dels aparells d'aire condicionat. Durant la presa de dades vam comprovar com, en engegar aquests aparells, hi havia un pic d'activitat PM10 durant uns pocs minuts i descendia.

Ampliem el gràfic a la zona amb concentracions menors de 100µg/m^3 per observar millor els valors dels PM més petits:

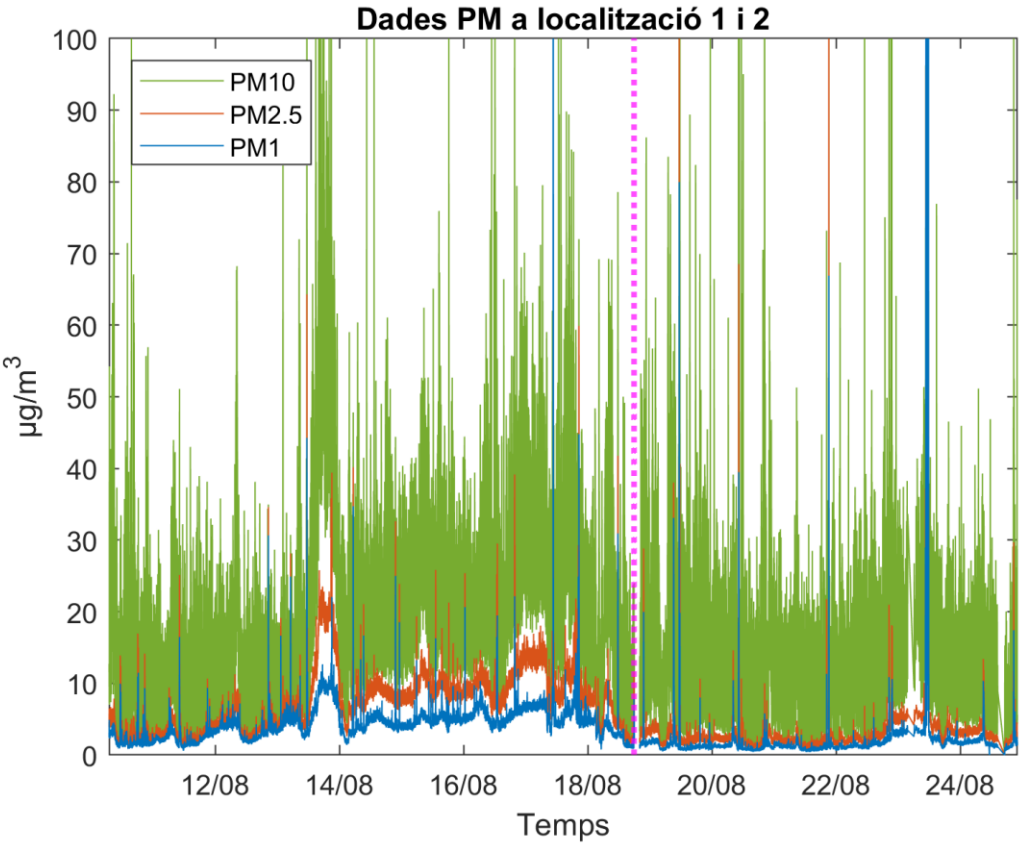

Fig. 3. 2. Representació dels PM mesurats a les dues localitzacions

Amb l'ampliació podem apreciar millor la dinàmica dels PM1 i PM2.5, observant el menor renou en les dades, veiem una similitud en el comportament dels tres tipus de partícules i també pics més petits. Durant la presa de mesures vam comprovar que es deuen a activitats antropogèniques més petites com espolsar roba, obrir l'armari de la localització 1, moure objectes propers o passar molt prop del sensor.

Degut a que tots aquests pics no són naturals, calculem un límit de 4 vegades el valor mig dels PM1 i PM2.5 i 5 vegades el del PM10, a fi d'eliminar els pics comentats anteriorment.

Una vegada tenim eliminats els pics més, calculem el valor mig horari de cada PM a les dues localitzacions i ho representem amb les dades filtrades:

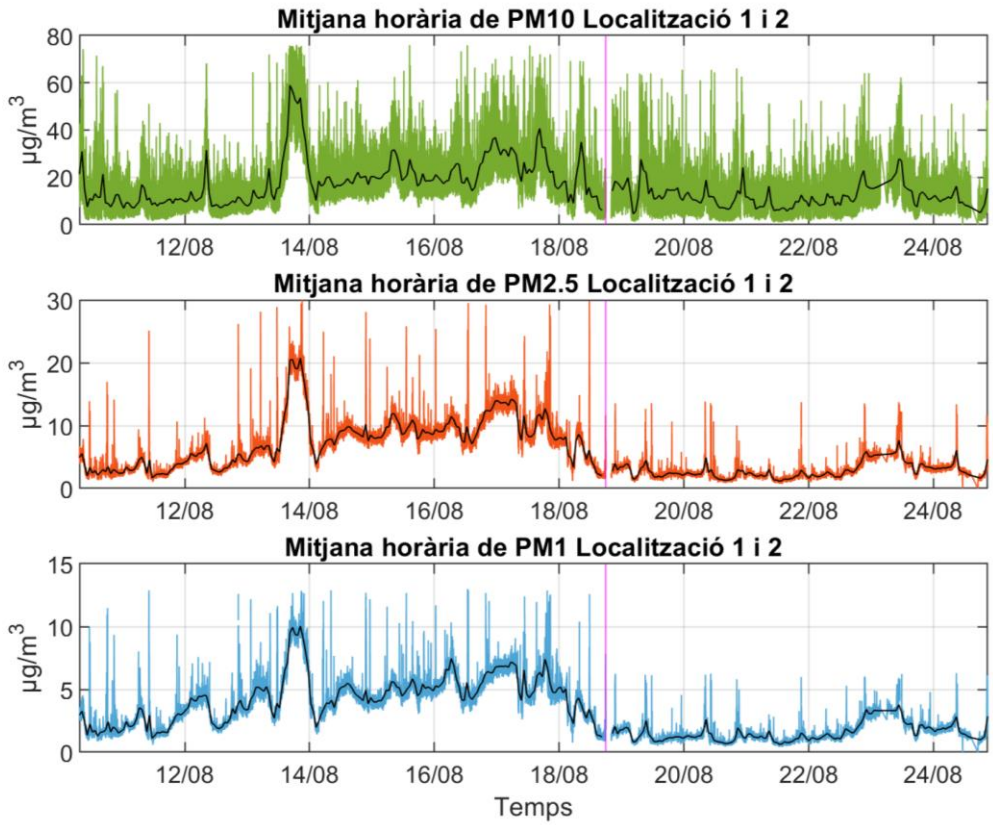

Fig. 3. 3. Representació de les dades de PM a les dues localitzacions filtrades i el valor mig horari.

Si ens fixem en els valors dels PM podem veure qualitativament la dinàmica que tenen durat els dies de mesura. Els primers i els darrers dies de mesura podem veure uns valors aproximats similars, per la qual cosa, podem suposar que el comportament de mesura en les dues localitzacions és similar i no afecta molt significativament a les mesures. Si tenim en compte els valors límits marcats per la directiva europea, per PM2.5 (límit de 25µg/m^3) podem considerar que amb la mitjana no superem aquests valors, encara que s'apropen molt dia 13. Per PM10 (límit 50µg/m^3) es mantenen concentracions inferiors, a excepció del mateix dia 13, en el que sí podem observar com es supera el límit. Més endavant analitzarem aquest cas particular i d'interès en relació a les condicions meteorològiques.

El nostre sistema també va mesurar dades de pressió i temperatura i les comparem amb les dades obtingudes per l'AEMET per comprovar si són correctes.

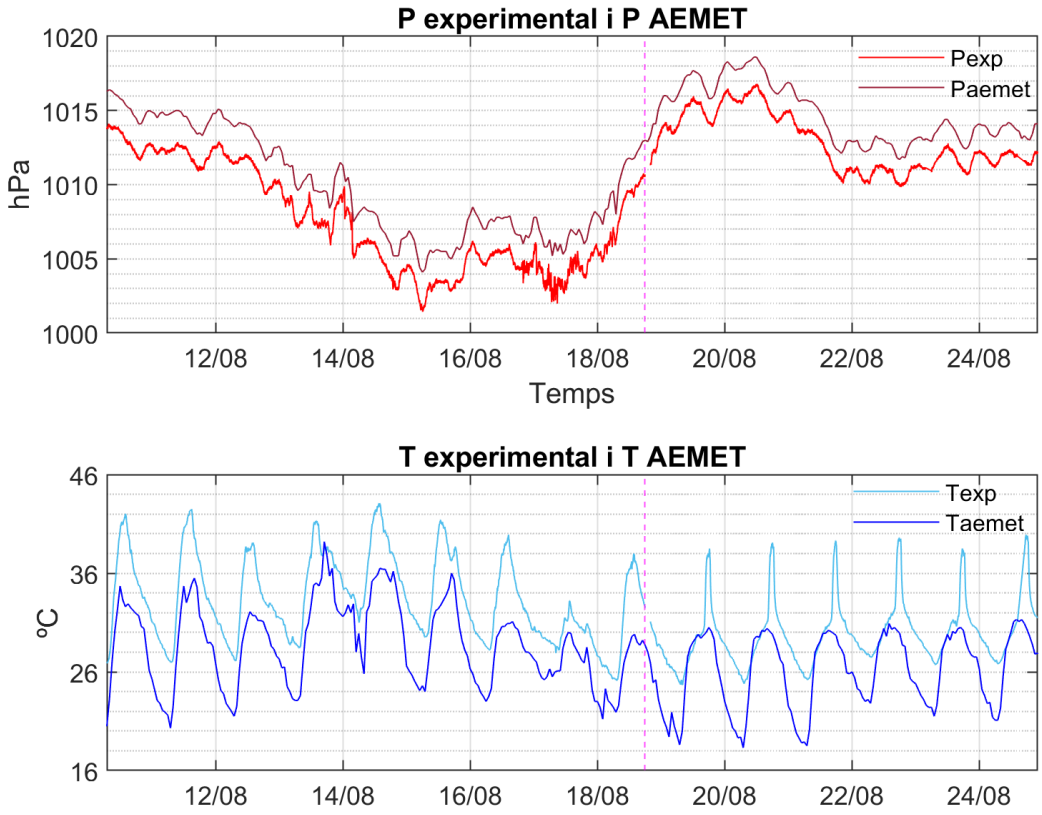

Fig. 3. 4. Dades de temperatura i pressió del nostre sistema a les dues localitzacions i dades de l'aemet.

Podem observar com tenim una correspondència de dades de pressió entre el nostre sensor i l'estació de l'aeroport en els dos llocs de mesura, però amb una diferència de la pressió experimental d'aproximadament 2hPa inferior respecte als valors de l'aemet.

Respecte a la temperatura, com era d'esperar i havíem comentat, a la localització 1 la temperatura és entre 3ºC i 8ºC superior a la mesurada a l'aeroport, amb un comportament similar i amb els pics de temperatura discretament més marcats i amb major amplitud que els de l'aemet. Per altra banda, a la localització 2 tenim un efecte similar en quant a valors més alts en 6-7ºC en els mínims i 7-8ºC en els màxims, per tant l'amplitud és major que als valors de referència. A més, els pics màxims dels nostres valors en la segona localització són un temps posterior als valors màxims de l'aemet. Tot això pot ser degut a que en la franja horària de 15h-18h al nostre sistema li incidia directament el sol i possiblement la ventilació que li vam proporcionar a la caixa no fos suficient per graduar bé la temperatura dins, creant un petit microclima on teníem el sensor.

Una vegada vist el comportament del nostre sensor de pressió i temperatura considerem correcte emprar les dades de l'aemet com a referència per buscar una possible relació entre els valors de PM i la situació meteorològica.

Representem les mitjanes calculades dels tres PM, així com les dades de pressió, temperatura i vent de referència a fi de cercar relació entre ells. Anem a analitzar els valors de PM amb P, T, velocitat del vent i Humitat.

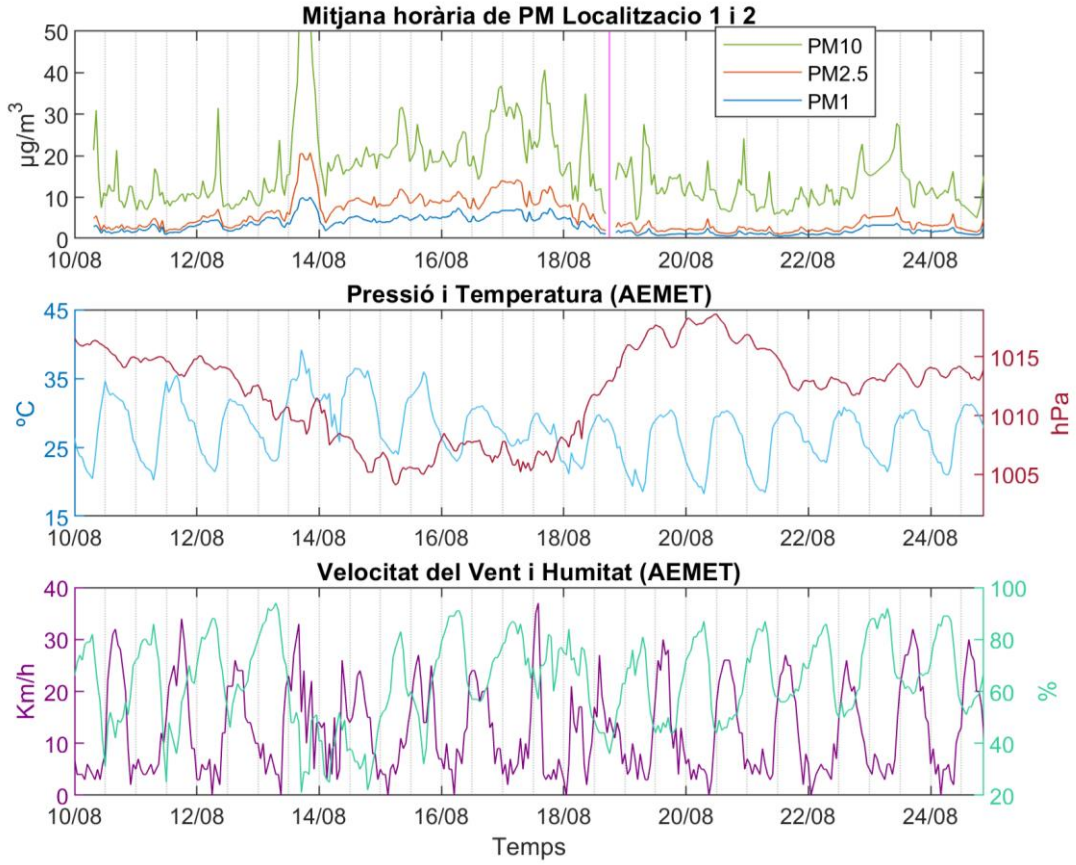

Fig. 3. 5. Representació de mitjanes de PM i dades de l'AEMET de P, T i velocitat del vent.

- Respecte T: No podem observar una relació clara entre PM i T, però en alguns casos puntuals podem veure com correspon un valor major de PM amb l'augment significatiu de la T. A més, tenim un cas particular de dia 13, on observem un gran pic de concentracions dels tres tipus de PM i correspon amb el pic de T d'aquest dia.
- Respecte a Velocitat del vent: Comparació similar a la anterior amb la T. No veiem cap relació significativa, possiblement degut també a la localització del nostre sistema, que es trobava en un racó sense circulació d'aire natural (localització 1) i a una finestra en un pati interior (localització 2), on la circulació de l'aire tampoc és la millor i es pot veure molt afectada. Per tant, no podem veure reflexat en les nostres dades la dispersió de partícules produïda pel vent.
- Respecte a P: En valors generals, podem veure una relació en quant a P i valors de PM. Amb pressions més altes veiem valors baixos de concentracions de PM, però amb la disminució de P (des de dia 13 fins dia 18) observem un augment major de partícules.

- Respecte a H: Podem veure com pràcticament tots els pics d'humitat coincideixen amb un pic de partícules o almenys amb un augment d'aquestes. Això ens du a pensar en un dels processos de dispersió de partícules comentats en l'apartat 1.2., la humitat atmosfèrica.

  També hem de tenir en compte que es van donar dos casos de poca precipitació durant el temps de recollida de mesures, que foren els dies 17 (acumulació de 0.3mm) i dia 22 (acumulació de 1.9mm). Aquestes dades de precipitació no afectaren les mesures de PM ja que són molt petites i no veiem que en el moment de la precipitació hi hagi cap disminució de les concentracions, que és el que caldria esperar com efecte de les precipitacions.

  Per tant, de manera qualitativa, veiem la relació que té la concentració de partícules amb la humitat i també trobem un període de majors concentracions de partícules que correspon amb una disminució de la pressió atmosfèrica.

  Centrant-nos en aquest cas particular, cerquem la situació atmosfèrica d'aquest període de dies amb un descens de P. Algunes de les imatges de satèl·lit [17] d'aquest període que podem veure a la Fig. 3.7., mostren l'arribada d'una vaguada d'aire subtropical portadora de pols sahrauí en suspensió, que també es va comprovar amb un mapa de calima [18] com veiem a la figura 3.6. En aquest mapa confirmem que, l'entrada de la calima a l'atmosfera balear es va produir dia 13, que coincideix amb el cas particular enregistrat d'un fort augment de partícules que, concretament donà lloc a una superació del valor límit de PM10.

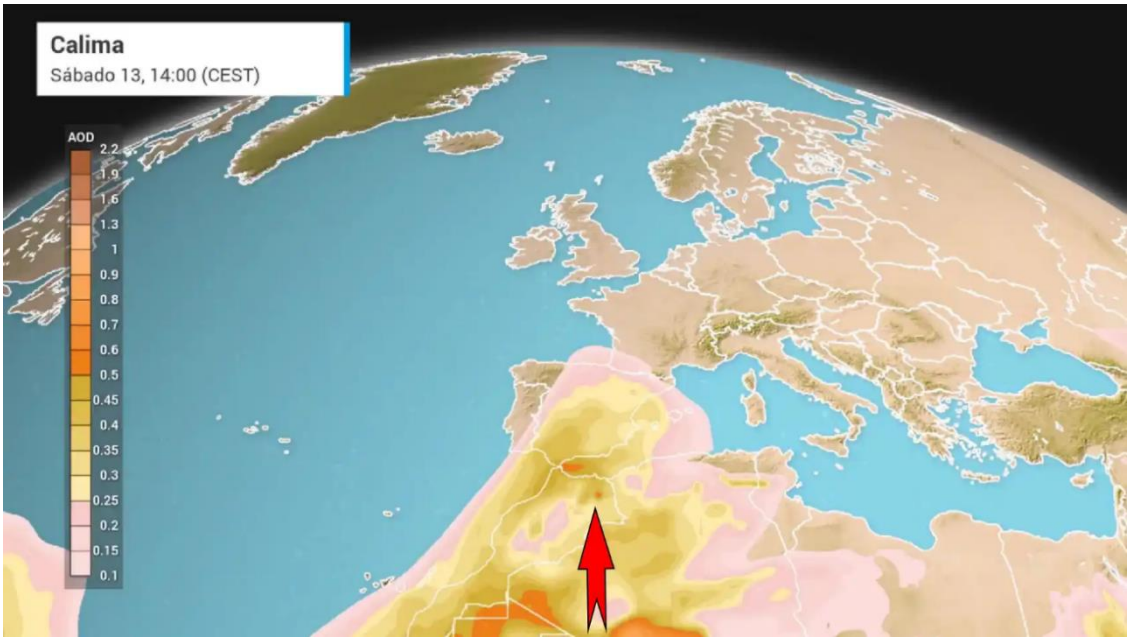

Fig. 3. 6. Mapa de calima de dia 13 d'agost, on es veu l'entrada de pols sahrauí.

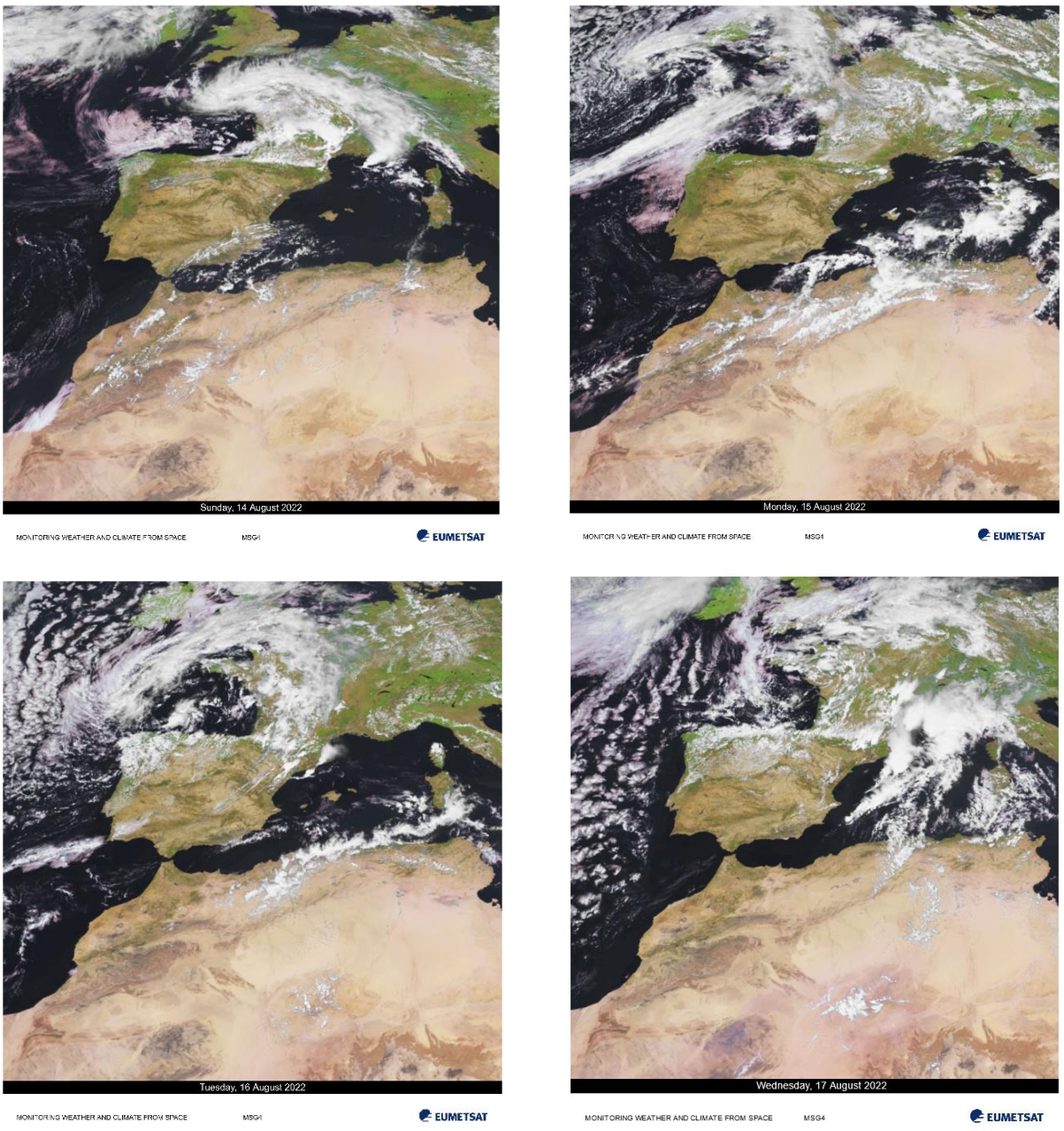

Fig. 3. 7. Imatges satèlit dels dies amb menor P i increment de les concentracions de PM.

4. Conclusions.

Degut a la importància de la qualitat de l'aire per a la salut i el medi ambient, és interesant cercar sistemes de mesura de les concentracions de partícules a l'aire. En el nostre cas ens hem centrat en PM10, PM2.5 i PM1 i hem pogut realitzar les mesures amb un sensor OPC molt pràctic i fiable. Gràcies a la programació d'una placa Nodemcu hem pogut integrar l'OPC i el sensor de pressió i temperatura formant un sistema de monitoreig de variables diferents interessants. A més hem facilitat el recull de dades a través de wifi i amb una aplicació com Thingspeak, que ens ha permès fàcilment controlar les dades a temps real i la descàrrega de totes les mesures en un arxiu de dades pel seu posterior tractament.

Hem après a programar i implementar la placa amb diferents sensors, la importància de les partícules a l'aire, així com la normativa vigent del seu control i alguns dels seus processos de dispersió. Finalment hem analitzat les dades obtingudes a fi de relacionar els valors de les concentracions mesurades amb els factors meteorològics naturals i les activitats antropogèniques.

Com a millores del treball, seria interesant obtenir les dades en un lloc amb unes condicions millors a les que hem treballat, com tenir el sistema de monitoreig a una alçada d'1m i mig o superior, sense parets al voltant per poder veure l'efecte del vent, sense cap aparell que mogui l'aire i afecti a les mesures i, possiblement, també una caixa amb major ventilació per evitar un augment significatiu de la pressió i la temperatura.

5. Referències.

[1] M. Puigcerver y M. D. Carrascal, *El medio atmosférico: meteorología y contaminación*. Barcelona, 2008.

[2] OMS, «Directrices mundiales de la OMS sobre la calidad del aire Resumen ejecutivo», pp. 1-16, 2021.

[3] C. de la U. E. Parlamento Europeo, «Directiva 2008/50/CE del Parlamento Europeo y del Consejo relativa a la calidad del aire ambiente y a una atmósfera más limpia en Europa», *D. Of. la Unión Eur. Ser. L*, n.º 152, 11 de junio, pp. 1-44, 2008.

[4] Comisión Europea, «Directiva 2015/1480/UE, de la Comisión, de 28 de agosto de 2015, por la que se modifican varios anexos de las Directivas 2004/107/CE y 2008/50/CE del Parlamento Europeo y del Consejo en los que se establecen las normas relativas a los métodos de referenci», *Boletín Of. la Unión Eur.*, vol. 226, n.º 6, pp. 4-11, 2015.

[5] España, «Ley 34/2007», pp. 46962-46987, 2007.

[6] Gobierno de España, «Real Decreto 102/2011, de 28 de enero, relativo a la mejora de la calidad del aire.», *Boletín Of. del estado*, vol. 25, pp. 9574-9626, 2011.

[7] «Sección de atmósfera-Introducción». http://www.caib.es/sites/atmosfera/es/introduccion-3183/.

[8] A. Ltd, «Alphasense User Manual OPC-N2 Optical Particle Counter Issue 2», vol. 44, n.º 5, pp. 1-34, 2019.

[9] Alphasense, «OPC-N2 Technical Specifications», vol. 44, n.º 0, 2014.

[10] «IoT Analytics - ThingSpeak Internet of Things». https://thingspeak.com/.

[11] «Alphasense OPC-N2 Interface Cable Assembly», p. 30, 2016.

[12] «GitHub - adafruit/Adafruit-BMP085-Library: A powerful but easy to use BMP085/BMP180 Library». https://github.com/adafruit/Adafruit-BMP085-Library.

[13] «GitHub - dhhagan/opcn2: C++ Library for the Alphasense OPC-N2 particle counter». https://github.com/dhhagan/opcn2.

[14] «GitHub - nmarcelo/opcn2-Arduino: C++ Library for the Alphasense OPC-N2 particle counter-Arduino». https://github.com/nmarcelo/opcn2-Arduino.

[15] «GitHub - mathworks/thingspeak-arduino: ThingSpeak Communication Library for Arduino, ESP8266 and ESP32». https://github.com/mathworks/thingspeak-arduino.

[16] «El Tiempo. Hoy y últimos días: Palma de Mallorca, Aeropuerto - Datos horarios - Tabla - Agencia Estatal de Meteorología - AEMET. Gobierno de España».

[17] «30th Anniversary Viewer — EUMETSAT». https://pics.eumetsat.int/viewer/index.html.

[18] «Mapa de calima - Meteored». https://www.tiempo.com/mapas-meteorologicos/#type=duaod550.

Annex I: Codi Arduino.

A.1: Codi principal

```
#include "Adafruit_BMP085.h"
#include "SPI.h"
#include "opcn2m.h"
#include "ESP8266WiFi.h"
#include "secrets.h"
#include "ThingSpeak.h"

#define CS D8

OPCN2 alpha(CS);
HistogramData hist;
ConfigVars vars;
Adafruit_BMP085 bmp;

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
int keyIndex = 0;
WiFiClient  client;

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
String myStatus = "";

void setup(){
    pinMode(CS, OUTPUT);
    Serial.begin(115200);

    if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085 sensor, check wiring!");
    while (1) {}
    }
    while (!Serial) {
    ;
    }

    WiFi.mode(WIFI_STA);
    ThingSpeak.begin(client);

    Serial.println("Testing OPC-N2 v" + String(alpha.firm_ver.major) + "." +
String(alpha.firm_ver.minor));
    vars = alpha.read_configuration_variables();
    Serial.println("\nConfiguration Variables");
    Serial.print("\tGSC:\t"); Serial.println(vars.gsc);
    Serial.print("\tSFR:\t"); Serial.println(vars.sfr);
    Serial.print("\tLaser DAC:\t"); Serial.println(vars.laser_dac);
    Serial.print("\tFan DAC:\t"); Serial.println(vars.fan_dac);
    Serial.print("\tToF-SFR:\t"); Serial.println(vars.tof_sfr);
    alpha.on();
    delay(1000);
}

void loop(){
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass);
      Serial.print(".");
      delay(30000);
    }
    Serial.println("\nConnected.");
  }
```

```
    hist = alpha.read_histogram();

  ThingSpeak.setField(1, bmp.readTemperature());
  ThingSpeak.setField(2, bmp.readPressure());
  ThingSpeak.setField(3, hist.pm1);
  ThingSpeak.setField(4, hist.pm25);
  ThingSpeak.setField(5, hist.pm10);

  ThingSpeak.setStatus(myStatus);
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }
    delay(15000);
}
```

A.2: opcn2.h

```
#ifndef Opcn2_h
#define Opcn2_h

struct Status {
    int fanON;
    int laserON;
    int fanDAC;
    int laserDAC;
};

struct Firmware {
    int major;
    int minor;
};

struct HistogramData {
    double bin0;
    double bin1;
    double bin2;
    double bin3;
    double bin4;
    double bin5;
    double bin6;
    double bin7;
    double bin8;
    double bin9;
    double bin10;
    double bin11;
    double bin12;
    double bin13;
    double bin14;
    double bin15;
    float bin1MToF;
    float bin3MToF;
    float bin5MToF;
    float bin7MToF;

    float sfr;

    unsigned long temp_pressure;

    float period;

    unsigned int checksum;
```

```
    float pm1;
    float pm25;
    float pm10;
};

struct PMData {
  float pm1;
  float pm25;
  float pm10;
};

struct ConfigVars {
    int bb0;
    int bb1;
    int bb2;
    int bb3;
    int bb4;
    int bb5;
    int bb6;
    int bb7;
    int bb8;
    int bb9;
    int bb10;
    int bb11;
    int bb12;
    int bb13;
    int bb14;

    float bpv0;
    float bpv1;
    float bpv2;
    float bpv3;
    float bpv4;
    float bpv5;
    float bpv6;
    float bpv7;
    float bpv8;
    float bpv9;
    float bpv10;
    float bpv11;
    float bpv12;
    float bpv13;
    float bpv14;
    float bpv15;

    float bpd0;
    float bpd1;
    float bpd2;
    float bpd3;
    float bpd4;
    float bpd5;
    float bpd6;
    float bpd7;
    float bpd8;
    float bpd9;
    float bpd10;
    float bpd11;
    float bpd12;
    float bpd13;
    float bpd14;
    float bpd15;

    float bsvw0;
    float bsvw1;
    float bsvw2;
    float bsvw3;
    float bsvw4;
    float bsvw5;
```

```
    float bsvw6;
    float bsvw7;
    float bsvw8;
    float bsvw9;
    float bsvw10;
    float bsvw11;
    float bsvw12;
    float bsvw13;
    float bsvw14;
    float bsvw15;

    float gsc;
    float sfr;

    unsigned int laser_dac;
    unsigned int fan_dac;
    unsigned int tof_sfr;
};

struct ConfigVars2 {
  int AMSamplingInterval;
  int AMIntervalCount;
  int AMFanOnIdle;
  int AMLaserOnIdle;
  int AMMaxDataArraysInFile;
  int AMOnlySavePMData;
};

class OPCN2
{
private:
    uint8_t _CS;
    int _fv;

    uint16_t _16bit_int(byte MSB, byte LSB);
    bool _compare_arrays(byte array1[], byte array2[], int length);
    float _calculate_float(byte val0, byte val1, byte val2, byte val3);
    uint32_t _32bit_int(byte val0, byte val1, byte val2, byte val3);

public:
    OPCN2(uint8_t chip_select);

    Firmware firm_ver;

    bool ping();
    bool on();
    bool off();
    bool write_config_variables(byte values[]);
    bool write_config_variables2(byte values[]);
    bool write_serial_number_string(byte values[]);
    bool save_config_variables();
    bool enter_bootloader();
    bool set_fan_power(uint8_t value);
```

A.3: opc2m.h

```
#include "SPI.h"
#include "opcn2.h"

OPCN2::OPCN2(uint8_t chip_select)
{
   _CS = chip_select;
 pinMode(_CS, OUTPUT);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
```

```
  SPI.setDataMode(SPI_MODE1);
  SPI.setClockDivider(SPI_CLOCK_DIV32);
  firm_ver.major = 18;
  firm_ver.minor = 2;
}
uint16_t OPCN2::_16bit_int(byte LSB, byte MSB)
{
  return ((MSB << 8) | LSB);
}
bool OPCN2::_compare_arrays(byte array1[], byte array2[], int length)
{
  bool result = true;
  for (int i = 0; i < length; i++){
    if (array1[i] != array2[i]){
      result = false;
    }
  }
  return result;
}
float OPCN2::_calculate_float(byte val0, byte val1, byte val2, byte val3)
{
  union u_tag {
    byte b[4];
    float val;
  } u;
  u.b[0] = val0;
  u.b[1] = val1;
  u.b[2] = val2;
  u.b[3] = val3;
  return u.val;
}
uint32_t OPCN2::_32bit_int(byte val0, byte val1, byte val2, byte val3)
{
  return ((val3 << 24) | (val2 << 16) | (val1 << 8) | val0);
}
bool OPCN2::ping()
{
  byte resp[1];
  byte expected[] = {0xF3};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0xCF);
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 1);
}
bool OPCN2::on()
{
  byte vals[2];
  byte expected[] = {0xF3, 0x03};
  digitalWrite(this->_CS, LOW);
  vals[0] = SPI.transfer(0x03);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  vals[1] = SPI.transfer(0x00);
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(vals, expected, 2);
}
bool OPCN2::off()
{
  byte vals[2];
  byte expected[] = {0xF3, 0x03};
  digitalWrite(this->_CS, LOW);
  vals[0] = SPI.transfer(0x03);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  vals[1] = SPI.transfer(0x01);
  digitalWrite(this->_CS, HIGH);
```

```
  return this->_compare_arrays(vals, expected, 2);
}
String OPCN2::read_information_string()
{
  String result = "";
  String tmp;
  byte vals[61];
  digitalWrite(this->_CS, LOW);
  SPI.transfer(0x3F);
  digitalWrite(this->_CS, HIGH);
  delay(3);
  digitalWrite(this->_CS, LOW);
  for (int i = 0; i < 60; i++){
    vals[i] = SPI.transfer(0x00);
    result += String((char)vals[i]);
    delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  return result;
}
struct Status OPCN2::read_status()
{
  Status data;
  byte vals[4];
    digitalWrite(this->_CS, LOW);
    SPI.transfer(0x13);
    digitalWrite(this->_CS, HIGH);
    delay(10);
    digitalWrite(this->_CS, LOW);
    for (int i = 0; i < 4; i++){
      vals[i] = SPI.transfer(0x13);
      delayMicroseconds(4);
    digitalWrite(this->_CS, HIGH);
    data.fanON    = (unsigned int)vals[0];
    data.laserON  = (unsigned int)vals[1];
    data.fanDAC   = (unsigned int)vals[2];
    data.laserDAC = (unsigned int)vals[3];
  }
  return data;
}
struct Firmware OPCN2::read_firmware_version()
{
  Firmware res;
    digitalWrite(this->_CS, LOW);
    SPI.transfer(0x12);
    digitalWrite(this->_CS, HIGH);
    delay(10);
    digitalWrite(this->_CS, LOW);
    res.major = (unsigned int)SPI.transfer(0x00);
    delayMicroseconds(4);
    res.minor = (unsigned int)SPI.transfer(0x00);
    digitalWrite(this->_CS, HIGH);
    return res;
}
bool OPCN2::write_config_variables(byte values[])
{
  return true;
}
bool OPCN2::write_config_variables2(byte values[])
{
  return true;
}
bool OPCN2::write_serial_number_string(byte values[])
{
  return true;
}
bool OPCN2::save_config_variables()
{
```

```
  byte resp[6];
  byte commands[] = {0x43, 0x3F, 0x3C, 0x3F, 0x3C, 0x43};
  byte expected[] = {0xF3, 0x43, 0x3f, 0x3c, 0x3f, 0x3c};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(commands[0]);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  for (int i = 1; i < (int)sizeof(commands); i++){
    resp[i] = SPI.transfer(commands[i]);
    delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 6);
}
bool OPCN2::enter_bootloader()
{
  byte resp[1];
  byte expected[] = {0xF3};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0x41);
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 1);
}
bool OPCN2::set_fan_power(uint8_t value)
{
  byte resp[3];
  byte expected[] = {0xF3, 0x42, 0x00};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0x42);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  resp[1] = SPI.transfer(0x00);
  delayMicroseconds(4);
  resp[2] = SPI.transfer(value);
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 3);
}
bool OPCN2::set_laser_power(uint8_t value)
{
  byte resp[3];
  byte expected[] = {0xF3, 0x42, 0x01};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0x42);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  resp[1] = SPI.transfer(0x01);
  delayMicroseconds(4);
  resp[2] = SPI.transfer(value);
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 3);
}
bool OPCN2::toggle_fan(bool state)
{
  byte resp[2];
  byte expected[] = {0xF3, 0x03};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0x03);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  if (state == true){
    resp[1] = SPI.transfer(0x04);
  }
  else {
    resp[1] = SPI.transfer(0x05);
```

```
  }
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 2);
}
bool OPCN2::toggle_laser(bool state)
{
  byte resp[2];
  byte expected[] = {0xF3, 0x03};
  digitalWrite(this->_CS, LOW);
  resp[0] = SPI.transfer(0x03);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  if (state == true){
    resp[1] = SPI.transfer(0x02);
  }
  else {
    resp[1] = SPI.transfer(0x03);
  }
  digitalWrite(this->_CS, HIGH);
  return this->_compare_arrays(resp, expected, 2);
}
struct ConfigVars OPCN2::read_configuration_variables()
{
  ConfigVars results;       // empty structure for the data
  byte vals[256];
  digitalWrite(this->_CS, LOW);
  SPI.transfer(0x3c);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  for (int i = 0; i < 256; i++){
    vals[i] = SPI.transfer(0x00);
    delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  results.bb0 = this->_16bit_int(vals[0], vals[1]);
  results.bb1 = this->_16bit_int(vals[2], vals[3]);
  results.bb2 = this->_16bit_int(vals[4], vals[5]);
  results.bb3 = this->_16bit_int(vals[6], vals[7]);
  results.bb4 = this->_16bit_int(vals[8], vals[9]);
  results.bb5 = this->_16bit_int(vals[10], vals[11]);
  results.bb6 = this->_16bit_int(vals[12], vals[13]);
  results.bb7 = this->_16bit_int(vals[14], vals[15]);
  results.bb8 = this->_16bit_int(vals[16], vals[17]);
  results.bb9 = this->_16bit_int(vals[18], vals[19]);
  results.bb10 = this->_16bit_int(vals[20], vals[21]);
  results.bb11 = this->_16bit_int(vals[22], vals[23]);
  results.bb12 = this->_16bit_int(vals[24], vals[25]);
  results.bb13 = this->_16bit_int(vals[26], vals[27]);
  results.bb14 = this->_16bit_int(vals[28], vals[29]);
  results.bpv0 = this->_calculate_float(vals[32], vals[33], vals[34], vals[35]);
  results.bpv1 = this->_calculate_float(vals[36], vals[37], vals[38], vals[39]);
  results.bpv2 = this->_calculate_float(vals[40], vals[41], vals[42], vals[43]);
  results.bpv3 = this->_calculate_float(vals[44], vals[45], vals[46], vals[47]);
  results.bpv4 = this->_calculate_float(vals[48], vals[49], vals[50], vals[51]);
  results.bpv5 = this->_calculate_float(vals[52], vals[53], vals[54], vals[55]);
  results.bpv6 = this->_calculate_float(vals[56], vals[57], vals[58], vals[59]);
  results.bpv7 = this->_calculate_float(vals[60], vals[61], vals[62], vals[63]);
  results.bpv8 = this->_calculate_float(vals[64], vals[65], vals[66], vals[67]);
  results.bpv9 = this->_calculate_float(vals[68], vals[69], vals[70], vals[71]);
  results.bpv10 = this->_calculate_float(vals[72], vals[73], vals[74], vals[75]);
  results.bpv11 = this->_calculate_float(vals[76], vals[77], vals[78], vals[79]);
  results.bpv12 = this->_calculate_float(vals[80], vals[81], vals[82], vals[83]);
  results.bpv13 = this->_calculate_float(vals[84], vals[85], vals[86], vals[87]);
  results.bpv14 = this->_calculate_float(vals[88], vals[89], vals[90], vals[91]);
  results.bpv15 = this->_calculate_float(vals[92], vals[93], vals[94], vals[95]);
  results.bpd0 = this->_calculate_float(vals[96], vals[97], vals[98], vals[99]);
```

```
  results.bpd1 = this->_calculate_float(vals[100], vals[101], vals[102],
vals[103]);
  results.bpd2 = this->_calculate_float(vals[104], vals[105], vals[106],
vals[107]);
  results.bpd3 = this->_calculate_float(vals[108], vals[109], vals[110],
vals[111]);
  results.bpd4 = this->_calculate_float(vals[112], vals[113], vals[114],
vals[115]);
  results.bpd5 = this->_calculate_float(vals[116], vals[117], vals[118],
vals[119]);
  results.bpd6 = this->_calculate_float(vals[120], vals[121], vals[122],
vals[123]);
  results.bpd7 = this->_calculate_float(vals[124], vals[125], vals[126],
vals[127]);
  results.bpd8 = this->_calculate_float(vals[128], vals[129], vals[130],
vals[131]);
  results.bpd9 = this->_calculate_float(vals[132], vals[133], vals[134],
vals[135]);
  results.bpd10 = this->_calculate_float(vals[136], vals[137], vals[138],
vals[139]);
  results.bpd11 = this->_calculate_float(vals[140], vals[141], vals[142],
vals[143]);
  results.bpd12 = this->_calculate_float(vals[144], vals[145], vals[146],
vals[147]);
  results.bpd13 = this->_calculate_float(vals[148], vals[149], vals[150],
vals[151]);
  results.bpd14 = this->_calculate_float(vals[152], vals[153], vals[154],
vals[155]);
  results.bpd15 = this->_calculate_float(vals[156], vals[157], vals[158],
vals[159]);
  results.bsvw0 = this->_calculate_float(vals[160], vals[161], vals[162],
vals[163]);
  results.bsvw1 = this->_calculate_float(vals[164], vals[165], vals[166],
vals[167]);
  results.bsvw2 = this->_calculate_float(vals[168], vals[169], vals[170],
vals[171]);
  results.bsvw3 = this->_calculate_float(vals[172], vals[173], vals[174],
vals[175]);
  results.bsvw4 = this->_calculate_float(vals[176], vals[177], vals[178],
vals[179]);
  results.bsvw5 = this->_calculate_float(vals[180], vals[181], vals[182],
vals[183]);
  results.bsvw6 = this->_calculate_float(vals[184], vals[185], vals[186],
vals[187]);
  results.bsvw7 = this->_calculate_float(vals[188], vals[189], vals[190],
vals[191]);
  results.bsvw8 = this->_calculate_float(vals[192], vals[193], vals[194],
vals[195]);
  results.bsvw9 = this->_calculate_float(vals[196], vals[197], vals[198],
vals[199]);
  results.bsvw10 = this->_calculate_float(vals[200], vals[201], vals[202],
vals[203]);
  results.bsvw11 = this->_calculate_float(vals[204], vals[205], vals[206],
vals[207]);
  results.bsvw12 = this->_calculate_float(vals[208], vals[209], vals[210],
vals[211]);
  results.bsvw13 = this->_calculate_float(vals[212], vals[213], vals[214],
vals[215]);
  results.bsvw14 = this->_calculate_float(vals[216], vals[217], vals[218],
vals[219]);
  results.bsvw15 = this->_calculate_float(vals[220], vals[221], vals[222],
vals[223]);
  results.gsc = this->_calculate_float(vals[224], vals[225], vals[226], vals[227]);
  results.sfr = this->_calculate_float(vals[228], vals[229], vals[230], vals[231]);
  results.laser_dac = (unsigned int)vals[232];
  results.fan_dac = (unsigned int)vals[233];
  results.tof_sfr = (unsigned int)vals[234];
  return results;
```

```
}
struct ConfigVars2 OPCN2::read_configuration_variables2()
{
  ConfigVars2 results;
  byte vals[9];
  digitalWrite(this->_CS, LOW);
  SPI.transfer(0x3D);
  digitalWrite(this->_CS, HIGH);
  delay(10);
  digitalWrite(this->_CS, LOW);
  for (int i = 0; i < 9; i++){
      vals[i] = SPI.transfer(0x00);
      delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  results.AMSamplingInterval    = this->_16bit_int(vals[0], vals[1]);
  results.AMIntervalCount       = this->_16bit_int(vals[2], vals[3]);
  results.AMFanOnIdle           = (unsigned int)vals[4];
  results.AMLaserOnIdle         = (unsigned int)vals[5];
  results.AMMaxDataArraysInFile = this->_16bit_int(vals[6], vals[7]);
  results.AMOnlySavePMData      = (unsigned int)vals[8];
return results;
}
String OPCN2::read_serial_number()
{
  String result = "";
  byte vals[60];
    digitalWrite(this->_CS, LOW);
    SPI.transfer(0x10);
    digitalWrite(this->_CS, HIGH);
    delay(3);
    digitalWrite(this->_CS, LOW);
    for (int i = 0; i < 60; i++){
        vals[i] = SPI.transfer(0x00);
        result += String((char)vals[i]);
        delayMicroseconds(4);
    }
   digitalWrite(this->_CS, HIGH);
   result.trim();
  return result;
}
struct PMData OPCN2::read_pm_data()
{
  PMData data;
  byte vals[12];
  digitalWrite(this->_CS, LOW);
  SPI.transfer(0x32);
  digitalWrite(this->_CS, HIGH);
  delay(12);
  digitalWrite(this->_CS, LOW);
  for (int i = 0; i < 12; i++){
      vals[i] = SPI.transfer(0x00);
      delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  data.pm1  = this->_calculate_float(vals[0], vals[1], vals[2], vals[3]);
  data.pm25 = this->_calculate_float(vals[4], vals[5], vals[6], vals[7]);
  data.pm10 = this->_calculate_float(vals[8], vals[9], vals[10], vals[11]);
  return data;
}
struct HistogramData OPCN2::read_histogram(bool convert_to_conc)
{
  HistogramData data;
  byte vals[62];
  digitalWrite(this->_CS, LOW);
  SPI.transfer(0x30);
  digitalWrite(this->_CS, HIGH);
  delay(12);
```

```
  digitalWrite(this->_CS, LOW);
  for (int i = 0; i < 62; i++){
      vals[i] = SPI.transfer(0x00);
      delayMicroseconds(4);
  }
  digitalWrite(this->_CS, HIGH);
  data.period = this->_calculate_float(vals[44], vals[45], vals[46], vals[47]);
  data.sfr    = this->_calculate_float(vals[36], vals[37], vals[38], vals[39]);
  double conv;
  if ( convert_to_conc != true ) {
      conv = 1.0;
  }
  else {
      conv = data.sfr * data.period;
  }
  data.bin0   = (double)this->_16bit_int(vals[0], vals[1]) / conv;
  data.bin1   = (double)this->_16bit_int(vals[2], vals[3]) / conv;
  data.bin2   = (double)this->_16bit_int(vals[4], vals[5]) / conv;
  data.bin3   = (double)this->_16bit_int(vals[6], vals[7]) / conv;
  data.bin4   = (double)this->_16bit_int(vals[8], vals[9]) / conv;
  data.bin5   = (double)this->_16bit_int(vals[10], vals[11]) / conv;
  data.bin6   = (double)this->_16bit_int(vals[12], vals[13]) / conv;
  data.bin7   = (double)this->_16bit_int(vals[14], vals[15]) / conv;
  data.bin8   = (double)this->_16bit_int(vals[16], vals[17]) / conv;
  data.bin9   = (double)this->_16bit_int(vals[18], vals[19]) / conv;
  data.bin10  = (double)this->_16bit_int(vals[20], vals[21]) / conv;
  data.bin11  = (double)this->_16bit_int(vals[22], vals[23]) / conv;
  data.bin12  = (double)this->_16bit_int(vals[24], vals[25]) / conv;
  data.bin13  = (double)this->_16bit_int(vals[26], vals[27]) / conv;
  data.bin14  = (double)this->_16bit_int(vals[28], vals[29]) / conv;
  data.bin15  = (double)this->_16bit_int(vals[30], vals[31]) / conv;

  data.bin1MToF = int(vals[32]) / 3.0;
  data.bin3MToF = int(vals[33]) / 3.0;
  data.bin5MToF = int(vals[34]) / 3.0;
  data.bin7MToF = int(vals[35]) / 3.0;
  data.temp_pressure = this->_32bit_int(vals[40], vals[41], vals[42], vals[43]);
  data.checksum = this->_16bit_int(vals[48], vals[49]);
  data.pm1 = this->_calculate_float(vals[50], vals[51], vals[52], vals[53]);
  data.pm25 = this->_calculate_float(vals[54], vals[55], vals[56], vals[57]);
  data.pm10 = this->_calculate_float(vals[58], vals[59], vals[60], vals[61]);
  return data;
}
```

A.4: Secrets

```
#define SECRET_SSID "nom de la xarxa wifi"
#define SECRET_PASS "contrasenya de la xarxa wifi"

#define SECRET_CH_ID 1819310
#define SECRET_WRITE_APIKEY "6N8LEXHSB2HCT68U"
```