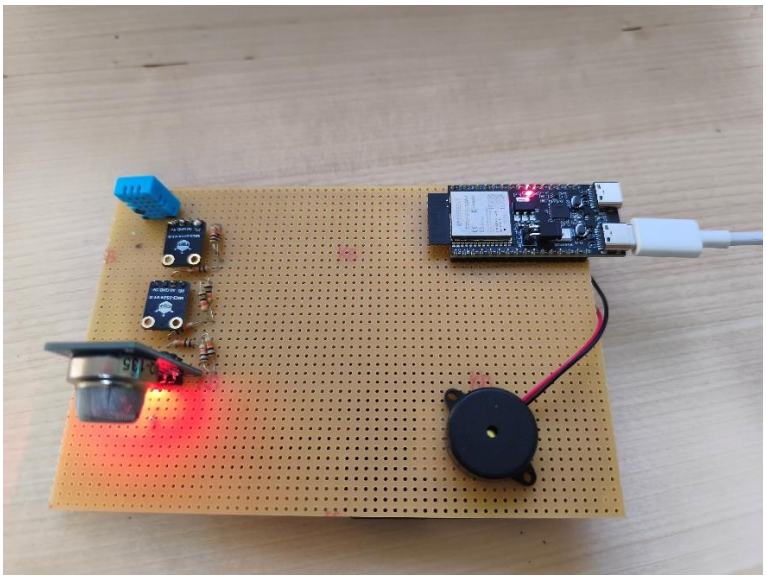
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σύστημα παρακολούθησης της ποιότητας του αέρα σε βιομηχανικές εγκαταστάσεις »



Του φοιτητή
Νούλη Χαράλαμπου
Αρ. Μητρώου: 52210Μ

Επιβλέπων
Χατζόπουλος Αργύριος

Φεβρουάριος 2026

Τίτλος Δ.Ε. «Σύστημα παρακολούθησης της ποιότητας του αέρα σε βιομηχανικές εγκαταστάσεις»
Κωδικός Δ.Ε. 25260
Ονοματεπώνυμο φοιτητή Νούλης Χαράλαμπος
Ονοματεπώνυμο εισηγητή Χατζόπουλος Αργύριος
Ημερομηνία ανάληψης Δ.Ε. 29-04-2025
Ημερομηνία περάτωσης Δ.Ε. 11-02-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στη σύντροφό μου, Μάγκυ»

Πρόλογος

Η ποιότητα του αέρα αποτελεί έναν από τους σημαντικότερους παράγοντες που επηρεάζουν άμεσα την υγεία των εργαζομένων, την ασφάλεια των εγκαταστάσεων και τη συνολική περιβαλλοντική επίδοση των βιομηχανικών μονάδων. Στους σύγχρονους βιομηχανικούς χώρους, η παρουσία αέριων ρύπων και τοξικών ενώσεων είναι συχνά αναπόφευκτη λόγω των διαδικασιών παραγωγής, της καύσης καυσίμων, της χρήσης χημικών ουσιών και των εκπομπών μηχανημάτων.

Η συνεχής και αξιόπιστη παρακολούθηση της ποιότητας του αέρα σε τέτοια περιβάλλοντα κρίνεται απολύτως αναγκαία, τόσο για την πρόληψη ατυχημάτων όσο και για τη συμμόρφωση με τους ισχύοντες περιβαλλοντικούς κανονισμούς. Οι παραδοσιακές μέθοδοι ελέγχου βασίζονται συχνά σε περιοδικές μετρήσεις, οι οποίες δεν μπορούν να αποτυπώσουν σε πραγματικό χρόνο επικίνδυνες μεταβολές των επιπέδων ρύπων.

Η ραγδαία εξέλιξη των τεχνολογιών αισθητηρίων, των μικροελεγκτών και του Διαδικτύου των Πραγμάτων (IoT) προσφέρει νέες δυνατότητες για την ανάπτυξη έξυπνων συστημάτων παρακολούθησης, τα οποία δύνανται να λειτουργούν σε πραγματικό χρόνο, να καταγράφουν και να μεταδίδουν δεδομένα συνεχώς και να παρέχουν άμεση ειδοποίηση σε περιπτώσεις υπέρβασης ορίων ασφαλείας.

Με βάση την προσωπική μου εμπειρία στους εργασιακούς χώρους της βιομηχανίας και του περιβάλλοντος, αποφάσισα να υλοποιήσω ένα τέτοιο σύστημα και να αναδείξω την σημασία του. Έτσι, η προσπάθεια μου επικεντρώθηκε στη δημιουργία ενός συστήματος παρακολούθησης της ποιότητας του αέρα σε βιομηχανικές εγκαταστάσεις, αξιοποιώντας αισθητήρες ανίχνευσης αερίων και μικροελεγκτή ESP32, με στόχο την απομακρυσμένη συλλογή, επεξεργασία και απεικόνιση των δεδομένων μέσω διαδικτυακής πλατφόρμας.

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τη σχεδίαση και υλοποίηση ενός συστήματος παρακολούθησης της ποιότητας του αέρα σε βιομηχανικές εγκαταστάσεις, με σκοπό την έγκαιρη ανίχνευση επιβλαβών αερίων και τη βελτίωση των συνθηκών ασφαλείας στους χώρους εργασίας. Το σύστημα βασίζεται στη χρήση μικροελεγκτή ESP32 και αισθητηρίων, όπως ο MQ-135, MiCS-2714 και MiCS-5524, οι οποίοι επιτρέπουν τη μέτρηση συγκεντρώσεων καυσίμων αερίων, πτητικών οργανικών ενώσεων και άλλων ρύπων σε πραγματικό χρόνο.

Τα δεδομένα που συλλέγονται μεταδίδονται ασύρματα μέσω διαδικτύου και αποθηκεύονται στην πλατφόρμα ThingSpeak, όπου πραγματοποιείται η οπτικοποίηση και η ανάλυσή τους μέσω γραφημάτων. Η μελέτη επικεντρώνεται τόσο στη θεωρητική τεκμηρίωση των παραμέτρων ποιότητας αέρα όσο και στην πρακτική κατασκευή και αξιολόγηση της απόδοσης του συστήματος.

Τα αποτελέσματα δείχνουν ότι η προτεινόμενη λύση μπορεί να αποτελέσει ένα αξιόπιστο, ευέλικτο και χαμηλού κόστους εργαλείο παρακολούθησης, συμβάλλοντας στην πρόληψη επικίνδυνων καταστάσεων και στη βελτίωση της περιβαλλοντικής διαχείρισης των βιομηχανικών χώρων.

«Air Quality Monitoring System in Industrial Facilities»

«Charalampos Noulis»

Abstract

This thesis focuses on the design and implementation of an air quality monitoring system for industrial facilities, aiming at the early detection of harmful gases and the improvement of workplace safety conditions. The system is based on ESP32 microcontroller and various gas sensors such as MQ-135, MiCS-2714 and MiCS-5524, which enable real-time measurement of combustible gases, volatile organic compounds (VOCs) and other atmospheric pollutants.

The collected data are transmitted wirelessly via the Internet and stored on the ThingSpeak platform, where they are visualized and analyzed through graphical representations. The study emphasizes both the theoretical investigation of air quality parameters and the practical development and performance evaluation of the proposed system.

The results demonstrate the proposed solution constitutes a reliable, flexible, and cost-effective monitoring tool, contributing to the prevention of hazardous situations and the enhancement of environmental management in industrial environments.

Ευχαριστίες

Θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου κύριο Αργύριο Χατζόπουλο για την υποστήριξη, την κατανόηση και τη βοήθεια που μου πρόσφερε κατά την διάρκεια της συγγραφής καθώς και τις γνώσεις που μου μετέδωσε κατά την διάρκεια των σπουδών μου στο ΠΜΣ Εφαρμοσμένα Ηλεκτρονικά Συστήματα.

Ακόμη, ευχαριστώ θερμά τον πρώην συμφοιτητή μου και αγαπημένο φίλο, Θεόδωρο Μανούση, για την τεχνική βοήθεια, την ανταλλαγή ιδεών και την αμέριστη συμπαράσταση.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Γενικά

Η ραγδαία ανάπτυξη της βιομηχανίας τις τελευταίες δεκαετίες έχει συμβάλει καθοριστικά στην οικονομική και τεχνολογική πρόοδο των σύγχρονων κοινωνιών. Παράλληλα όμως έχει επιφέρει σημαντικές περιβαλλοντικές επιπτώσεις, με κυριότερη την επιβάρυνση της ποιότητας του αέρα, ιδιαίτερα σε περιοχές όπου λειτουργούν βαριές βιομηχανικές εγκαταστάσεις. Η ατμοσφαιρική ρύπανση αποτελεί ένα από τα σημαντικότερα περιβαλλοντικά προβλήματα παγκοσμίως, καθώς επηρεάζει άμεσα την δημόσια υγεία, το οικοσύστημα και την ποιότητα ζωής των κατοίκων.

Στις βιομηχανικές εγκαταστάσεις, η εκπομπή αερίων ρύπων και αιωρούμενων σωματιδίων είναι αποτέλεσμα διαφόρων παραγωγικών διεργασιών, όπως η καύση καυσίμων, η χημική επεξεργασία, η μεταλλουργία, και η λειτουργία μηχανολογικού εξοπλισμού. Οι ρύποι αυτοί όπως το μονοξείδιο του άνθρακα (CO), το διοξείδιο του άνθρακα (CO_2), τα οξείδια του αζώτου (NOx), το διοξείδιο του θείου και τα αιωρούμενα σωματίδια PM2.5 και PM10, δύνανται να προκαλέσουν σοβαρές επιπτώσεις στην ανθρώπινη υγεία, αλλά και μακροπρόθεσμες βλάβες στο φυσικό περιβάλλον.

Η ανάγκη για συνεχή και αξιόπιστη παρακολούθηση της ποιότητας του αέρα στους βιομηχανικούς χώρους καθίσταται επομένως επιτακτική, τόσο για την προστασία των εργαζομένων όσο και για τη συμμόρφωση με τα εθνικά και ευρωπαϊκά περιβαλλοντικά πρότυπα. Η αξιοποίηση σύγχρονων τεχνολογιών, όπως τα ενσωματωμένα ηλεκτρονικά συστήματα και οι πλατφόρμες Διαδικτύου των πραγμάτων (Internet of Things-IoT), επιτρέπει την ανάπτυξη έξυπνων συστημάτων παρακολούθησης που προσφέρουν συνεχή συλλογή, ανάλυση και απομακρυσμένη απεικόνιση περιβαλλοντικών δεδομένων σε πραγματικό χρόνο.

Εικόνα 1.1:Ατμόσφαιρα βιομηχανικής περιοχής

1.2 Σκοπός και στόχοι της εργασίας

Σκοπός της εργασίας είναι η ανάπτυξη ενός λειτουργικού και αξιόπιστου συστήματος μέτρησης και παρακολούθησης της ποιότητας του αέρα σε βιομηχανικό περιβάλλον, το οποίο θα μπορεί να καταγράφει σε πραγματικό χρόνο τις συγκεντρώσεις βασικών ατμοσφαιρικών ρύπων και να παρέχει την αντίστοιχη πληροφόρηση στους χρήστες μέσω ψηφιακής πλατφόρμας.

Οι επιμέρους στόχοι της εργασίας συνοψίζονται ως εξής:

- Η μελέτη των βιομηχανικών παραγόντων που επηρεάζουν την ποιότητα του αέρα.
- Η επιλογή κατάλληλων αισθητήρων για τη μέτρηση συγκεκριμένων αερίων ρύπων και αιωρούμενων σωματιδίων.
- Η σχεδίαση και υλοποίηση του ηλεκτρονικού κυκλώματος μέτρησης.
- Η ανάπτυξη συστήματος συλλογής και επεξεργασίας δεδομένων μέσω μικροελεγκτή.
- Η υλοποίηση μηχανισμού αποθήκευσης και απεικόνισης των δεδομένων μέσω διαδικτυακής πλατφόρμας.
- Η αξιολόγηση της απόδοσης του συστήματος.

Κεφάλαιο 2ο: Θεωρητικό υπόβαθρο ποιότητας αέρα

2.1 Έννοια και ορισμός της ποιότητας του αέρα

Η ποιότητα του αέρα αναφέρεται στο σύνολο των φυσικών, χημικών και βιολογικών χαρακτηριστικών της ατμόσφαιρας, τα οποία καθορίζουν τον βαθμό καθαρότητας ή ρύπανσης της και, κατ' επέκταση, την καταλληλότητά της για ανθρώπινη αναπνοή και περιβαλλοντική ισορροπία. Η ποιότητα του αέρα αξιολογείται κυρίως μέσω της συγκέντρωσης συγκεκριμένων ατμοσφαιρικών ρύπων, οι οποίοι όταν υπερβαίνουν προκαθορισμένα όρια δύνανται να επιφέρουν αρνητικές επιπτώσεις στην υγεία και το οικοσύστημα. [16]

Σύμφωνα με διεθνείς οργανισμούς, όπως ο Παγκόσμιος Οργανισμός Υγείας (ΠΟΥ) και η Ευρωπαϊκή ένωση, η ποιότητα του αέρα χαρακτηρίζεται ως καλή, μέτρια ή επιβαρυμένη βάσει του δείκτη ποιότητας αέρα (Air Quality Index – AQI), ο οποίος αποτελεί σύνθετο δείκτη που λαμβάνει υπόψη τις συγκεντρώσεις των σημαντικότερων ρύπων.

2.2 Ατμοσφαιρική ρύπανση - Επιπτώσεις

Ως ατμοσφαιρική ρύπανση ορίζεται η παρουσία στην ατμόσφαιρα ουσιών ή μορφών ενέργειας σε συγκεντρώσεις που προκαλούν βλαβερές επιπτώσεις στον άνθρωπο, στους ζώντες οργανισμούς και στο φυσικό περιβάλλον. Οι ρύποι αυτοί μπορεί να προέρχονται τόσο από φυσικές πηγές, όπως ηφαιστειακές εκρήξεις και πυρκαγιές, όσο και από ανθρωπογενείς δραστηριότητες, με κυριότερη τη βιομηχανική παραγωγή.

Στο πλαίσιο των βιομηχανικών εγκαταστάσεων, η ατμοσφαιρική ρύπανση συνδέεται άμεσα με τη λειτουργία καυστήρων, λεβήτων, συστημάτων παραγωγής ενέργειας, χημικών διεργασιών και τη διακίνηση πρώτων υλών, με αποτέλεσμα τη συνεχή έκλυση αερίων ρύπων και αιωρούμενων σωματιδίων.

Επιπτώσεις:

Η έκθεση σε αυξημένες συγκεντρώσεις ατμοσφαιρικών ρύπων μπορεί να προκαλέσει πληθώρα προβλημάτων υγείας, όπως αναπνευστικές δυσλειτουργίες, άσθμα, χρόνιες αποφρακτικές πνευμονοπάθειες και καρδιαγγειακές παθήσεις. Επιπλέον, η ατμοσφαιρική ρύπανση επηρεάζει αρνητικά τη βιοποικιλότητα, την ποιότητα των υδάτων και τη δομή των εδαφών.

Σε επίπεδο βιομηχανικών εγκαταστάσεων, η επιβάρυνση της ποιότητας του αέρα **αυξάνει τον κίνδυνο ατυχημάτων, μειώνει την παραγωγικότητα των εργαζομένων** και ενδέχεται να οδηγήσει σε νομικές κυρώσεις λόγω μη συμμόρφωσης με τα προβλεπόμενα περιβαλλοντικά όρια. [17]

2.3 Κατηγορίες ατμοσφαιρικών ρύπων

2.3.1 Πρωτογενείς ρύποι

Πρωτογενείς ονομάζονται οι ρύποι που εκπέμπονται απευθείας στην ατμόσφαιρα από φυσικές πηγές (ηφαιστειακές εκρήξεις, πυρκαγιές) ή ανθρωπογενείς πηγές (βιομηχανία, μεταφορές, παραγωγή ενέργειας, οικιακή θέρμανση) χωρίς να προηγηθεί χημικός μετασχηματισμός αυτών στο περιβάλλον. Αποτελούν την αρχική μορφή ρύπανσης και συχνά λειτουργούν ως πρόδρομες ουσίες για τον σχηματισμό δευτερογενών ρύπων.

Οι πιο συχνά παρατηρούμενοι ρύποι στα βιομηχανικά περιβάλλοντα ανήκουν σε αυτή την κατηγορία και είναι οι εξής:

- Μονοξείδιο του άνθρακα (CO)

  Άχρωμο και άοσμο αέριο, ιδιαίτερα τοξικό, το οποίο δεσμεύει την αιμοσφαιρίνη, μειώνοντας τη μεταφορά οξυγόνου στον οργανισμό.

  Πηγές εκπομπής: Καυσαέρια οχημάτων (ατελής καύση βενζίνης και πετρελαίου), σόμπες και λέβητες, βιομηχανική καύση.

- Διοξείδιο του θείου (SO_2)

  Παράγεται κυρίως από την καύση θειούχων καυσίμων, ερεθίζει το αναπνευστικό σύστημα και προκαλεί περιβαλλοντική υποβάθμιση.

  Πηγές εκπομπής: Σταθμοί παραγωγής ηλεκτρικής ενέργειας με λιγνίτη ή πετρέλαιο, Διυλιστήρια, μεταλλουργικές βιομηχανίες.

- Οξείδια του αζώτου (NOx)

  Συμβάλλουν στον σχηματισμό φωτοχημικού νέφους και όξινης βροχής.

  Πηγές εκπομπής: Οχήματα με κινητήρες εσωτερικής καύσης, αεροπλάνα, θερμικοί σταθμοί, βιομηχανικοί καυστήρες

- Πτητικές οργανικές ενώσεις (VOCs)

  Οργανικές χημικές ουσίες που εξατμίζονται εύκολα.

  Πηγές εκπομπής: Βαφές και διαλύτες, καύσιμα, χημικές βιομηχανίες, καπνός τσιγάρου

- Αιωρούμενα σωματίδια (PM10, PM2.5)

  Μικροσκοπικά στερεά ή υγρά σωματίδια, εισχωρούν στους πνεύμονες και προκαλούν σοβαρές επιπτώσεις στην υγεία.

  Πηγές εκπομπής: Καύση ξύλου και άνθρακα, φθορά ελαστικών και φρένων, οικοδομικές εργασίες, βιομηχανική σκόνη.

2.3.2 Δευτερογενείς ρύποι

Δευτερογενείς ρύποι είναι εκείνοι που δεν εκπέμπονται απευθείας στην ατμόσφαιρα, αλλά σχηματίζονται εντός αυτής μέσω ή φωτοχημικών αντιδράσεων μεταξύ πρωτογενών ρύπων και φυσικών συστατικών της ατμόσφαιρας, όπως το οξυγόνο, η υγρασία και η ηλιακή ακτινοβολία.

Αποτελούν προϊόντα μετασχηματισμού των πρωτογενών ρύπων και **συχνά είναι πιο επικίνδυνοι** και σύνθετοι, καθώς έχουν μεγαλύτερη χημική δραστικότητα. Χαρακτηριστικό παράδειγμα αποτελεί το όζον (O_3) το οποίο δημιουργείται από αντιδράσεις μεταξύ NOx και VOCs.

2.4 Σημασία του Ελέγχου της Ποιότητας του Αέρα σε Βιομηχανικές Εγκαταστάσεις

Ο **συστηματικός έλεγχος** της ποιότητας του αέρα σε βιομηχανικές εγκαταστάσεις είναι **καθοριστικός** για την **προστασία της υγείας** των εργαζομένων, την ασφαλή λειτουργία των διεργασιών και τη συμμόρφωση με τα περιβαλλοντικά πρότυπα. Οι βιομηχανικές διαδικασίες μπορούν να απελευθερώνουν αέρια και σωματιδιακούς ρύπους, όπως VOCs, οξείδια του αζώτου, CO, CO_2 και αιωρούμενα σωματίδια, τα οποία σε υψηλές συγκεντρώσεις επηρεάζουν άμεσα το αναπνευστικό σύστημα και συμβάλλουν σε μακροχρόνιες επιπτώσεις στην υγεία.

Η παρουσία ρύπων δεν επιβαρύνει μόνο τον άνθρωπο αλλά και τον εξοπλισμό, προκαλώντας φθορές και δυσλειτουργίες, ιδιαίτερα σε περιβάλλοντα με εύφλεκτα αέρια ή ευαίσθητες ηλεκτρονικές συσκευές. Η συνεχής παρακολούθηση επιτρέπει τον άμεσο εντοπισμό επικίνδυνων συγκεντρώσεων και τη λήψη διορθωτικών μέτρων, βελτιώνοντας τόσο την ασφάλεια όσο και την αποδοτικότητα των παραγωγικών διαδικασιών.

Παράλληλα, οι βιομηχανίες οφείλουν να συμμορφώνονται με κανονισμούς για την ποιότητα της ατμόσφαιρας και τις οριακές τιμές επαγγελματικής έκθεσης. Η καταγραφή και ανάλυση δεδομένων αέρα αποτελεί βασικό εργαλείο για την τήρηση των προτύπων, την αποφυγή παραβιάσεων και την περιβαλλοντική υπευθυνότητα. Συνεπώς, ο έλεγχος του αέρα συνιστά θεμελιώδη παράμετρο για ασφαλή, αποδοτική και περιβαλλοντική και βιώσιμη βιομηχανική λειτουργία.

Κεφάλαιο 3ο: Αισθητήρες μέτρησης ποιότητας αέρα

3.1 Εισαγωγή στους αισθητήρες ποιότητας αέρα

Οι αισθητήρες ποιότητας αέρα είναι ηλεκτρονικές διατάξεις που ανιχνεύουν και ποσοτικοποιούν συγκεκριμένους αέριους και σωματιδιακούς ρύπους, επιτρέποντας την αξιολόγηση της χημικής σύστασης και της καθαρότητας του ατμοσφαιρικού περιβάλλοντος. Λειτουργούν αξιοποιώντας φυσικοχημικές αρχές, όπως μεταβολές της ηλεκτρικής αντίστασης, ηλεκτροχημικές αντιδράσεις ή οπτικές και φασματοσκοπικές μετρήσεις, προκειμένου να εντοπίσουν και να μετρήσουν ουσίες όπως μονοξείδιο του άνθρακα (CO), διοξείδιο του άνθρακα (CO_2), πτητικές οργανικές ενώσεις (VOCs), οξείδια του αζώτου (NOx), όζον (O_3) και αιωρούμενα σωματίδια.

Η χρήση τους είναι απαραίτητη σε βιομηχανικούς χώρους και συστήματα περιβαλλοντικής παρακολούθησης, καθώς παρέχουν αξιόπιστα δεδομένα σε πραγματικό χρόνο, συμβάλλοντας στην προστασία της υγείας, στην ασφαλή λειτουργία των διεργασιών και στη συμμόρφωση με τα πρότυπα ποιότητας αέρα.

3.2 Κατηγορίες αισθητήρων αερίων ρύπων

Οι αισθητήρες αερίων ρύπων διακρίνονται ανάλογα με την τεχνολογία λειτουργίας τους στις ακόλουθες κύριες κατηγορίες:

3.2.1 Ηλεκτροχημικοί αισθητήρες

Οι ηλεκτροχημικοί αισθητήρες βασίζονται σε χημικές αντιδράσεις μεταξύ του ρύπου και ενός ηλεκτρολύτη. Η αντίδραση αυτή παράγει ηλεκτρικό ρεύμα ανάλογο της συγκέντρωσης του ρύπου. Χαρακτηρίζονται από υψηλή ακρίβεια και χαμηλή κατανάλωση ενέργειας. Στα αρνητικά τους χαρακτηριστικά ανήκουν η περιορισμένη διάρκεια ζωής (1-3 χρόνια) και το γεγονός ότι επηρεάζονται από θερμοκρασία και υγρασία. [13]

Εικόνα 3.1:Ηλεκτροχημικός αισθητήρας ME3-C2H6S2

Εικόνα 3.2: Ηλεκτροχημικός αισθητήρας CO-B4

3.2.2 Αισθητήρες ημιαγωγών (MOS)

Οι αισθητήρες ημιαγωγών λειτουργούν μέσω της μεταβολής της ηλεκτρικής αντίστασης ενός υλικού (συνήθως οξειδίου μετάλλου) όταν έρχεται σε επαφή με συγκεκριμένα αέρια. Είναι οικονομικοί και ανθεκτικοί, ωστόσο και αυτοί επηρεάζονται από τη θερμοκρασία και την υγρασία. [14]

Εικόνα 3.3:Ημιαγωγός αισθητήρας MQ-135

Εικόνα 3.4: Ημιαγωγός αισθητήρας MiCS-2714

3.2.3 Αισθητήρες υπερύθρων (NDIR)

Αυτός ο τύπος αισθητήρων βασίζεται στην απορρόφηση υπέρυθρης ακτινοβολίας από συγκεκριμένα αέρια. Το επίπεδο απορρόφησης συσχετίζεται με τη συγκέντρωση του αερίου. Οι NDIR αισθητήρες προσφέρουν υψηλή ακρίβεια και σταθερότητα, χωρίς να επηρεάζονται σημαντικά από άλλους ρύπους. [15]

Εικόνα 3.5: Αισθητήρας υπερύθρων MH Z19-C

Εικόνα 3.6: Senseair S8

3.2.4 Οπτικοί αισθητήρες αιωρούμενων σωματιδίων PM

Οι αισθητήρες αυτοί βασίζονται στη σκέδαση φωτός για την ανίχνευση των αιρούμενων σωματιδίων. Χρησιμοποιούν μια πηγή φωτός (συνήθως LED ή laser) και έναν φωτοαισθητήρα που μετρά την ένταση του φωτός που διασκορπίζεται όταν τα σωματίδια διέρχονται από τον θάλαμο μέτρησης. Οι αισθητήρες αυτοί προσφέρουν καλή ακρίβεια, και εύκολη ενσωμάτωση σε μικροελεγκτές όπως ο ESP32. Ωστόσο, επηρεάζονται από την υγρασία και απαιτούν περιοδικό καθαρισμό για βέλτιστη λειτουργία. [10]

Εικόνα 3.7:Οπτικός αισθητήρας DFROBOT

Εικόνα 3.8: Οπτικός αισθητήρας Plantower PMS5003

3.3 Χαρακτηριστικά Αισθητήρων

Οι αισθητήρες που χρησιμοποιούνται σε ένα σύστημα μέτρησης διέπονται από τα εξής χαρακτηριστικά:

- *Συνάρτηση μεταφοράς*

Κάθε αισθητήρας παράγει ένα σήμα εξόδου, το οποίο στην ουσία αντιπροσωπεύει την πραγματική τιμή του ερεθίσματος. Η συνάρτηση μεταφοράς καθορίζει την εξάρτηση του ηλεκτρικού σήματος που παράγεται από τον αισθητήρα και του ερεθίσματος που αποτελεί την είσοδο του αισθητήρα. Η σχέση που περιγράφει τα παραπάνω είναι η εξής:

$$\mathbf{S = f(s)}$$

Όπου **S**: το ηλεκτρικό σήμα που παράγεται από τον αισθητήρα και **s**: το ερέθισμα που δέχεται ο αισθητήρας. [1]

- *Περιοχή τιμών εισόδου*

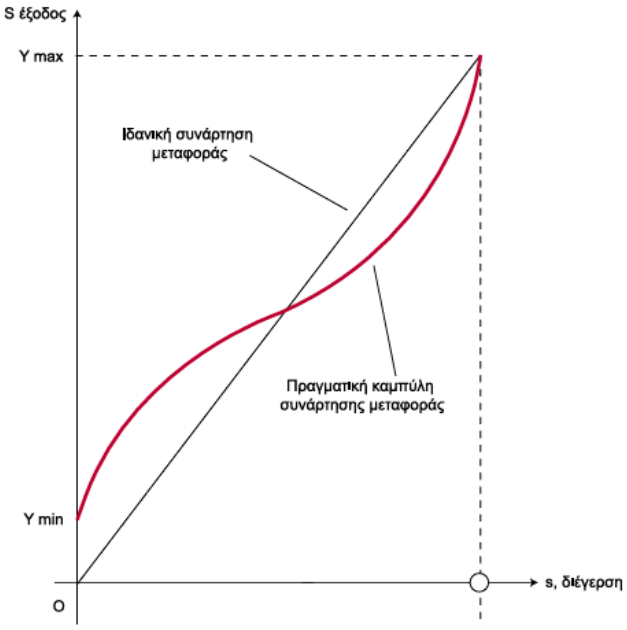
Αναφέρεται στο εύρος των τιμών του ερεθίσματος που μπορεί να δεχτεί ο αισθητήρας. [1]

- *Εύρος πλήρους κλίμακας εισόδου (IFS, Input Full Scale)*

Αποτελεί την αλγεβρική διαφορά της μέγιστης (x_{max}) από την ελάχιστη (x_{min}) τιμή του ερεθίσματος. [1]

- *Εύρος πλήρους κλίμακας εξόδου (FSO, Full Scale Output)*

Η αλγεβρική διαφορά μεταξύ του μέγιστου (y_{max}) και του ελάχιστου (y_{min}) σήματος εξόδου σε όλο το πεδίο εισόδου το ερεθίσματος **s** του αισθητήρα. [1]



Εικόνα 3.9: Εύρος πλήρους κλίμακας εξόδου σε καμπύλη συνάρτηση μεταφοράς αισθητήρα

- *Ακρίβεια*

Ακρίβεια ενός αισθητήρα ορίζεται η διαφορά που παρουσιάζει το σήμα εξόδου σε σχέση με την πραγματική τιμή. Καθώς όμως είναι αδύνατο να γνωρίζουμε την πραγματική τιμή του μεγέθους, η ακρίβεια αποδίδεται σαφέστερα μέσω σχετικού σφάλματος. Η σχέση που περιγράφει την ακρίβεια ενός αισθητήρα μέσω σφάλματος είναι η παρακάτω:

$$\mathbf{e_{res}} = \frac{(\boldsymbol{\mu\varepsilon\tau\rho o\acute{u}\mu\varepsilon\nu\eta\ \tau\iota\mu\acute{\eta}} - \boldsymbol{\pi\rho\alpha\gamma\mu\alpha\tau\iota\kappa\acute{\eta}\ \tau\iota\mu\acute{\eta}})}{\boldsymbol{\pi\rho\alpha\gamma\mu\alpha\tau\iota\kappa\acute{\eta}\ \tau\iota\mu\acute{\eta}}} \qquad \mathbf{(1)}$$

Η συνάρτηση μεταφοράς μπορεί να είναι γραμμική, λογαριθμική, εκθετική ή πολυωνυμική. [1]

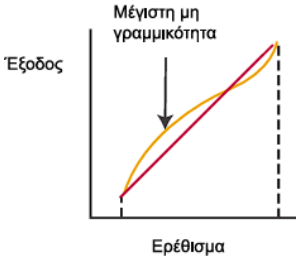
- *Υστέρηση*

Υστέρηση είναι το φαινόμενο όπου ο αισθητήρας δεν δίνει την ίδια τιμή εξόδου για την ίδια τιμή εισόδου όταν η είσοδος αυξάνεται σε σχέση με όταν μειώνεται. [1]

- *Μη γραμμικότητα*

Κατά γενικό κανόνα η τιμή εξόδου των αισθητήρων εμφανίζει απόκλιση από την γραμμικότητά τους. Όταν συμβαίνει αυτό, τότε λέμε πως υπάρχει σφάλμα μη γραμμικότητας. Για την γραμμικοποίηση της εξόδου μπορούν να εφαρμοστούν μαθηματικές διορθώσεις όπως η μέθοδος ελαχίστων τετραγώνων (Least Squares (LS)) (Εικόνα 3.6.1.). [1]

Εικόνα 3.10: Χάραξη της ευθείας με LS



Εικόνα 3.11: Χάραξη της ευθείας σύνδεσης των άκρων της μη γραμμικής καμπύλης

Εικόνα 3.12: Χάραξη της ευθείας μεταξύ των μεγίστων της μη γραμμικής καμπύλης

- *Διακριτική ικανότητα*

Πρόκειται για τη μικρότερη μεταβολή του μετρούμενου μεγέθους που μπορεί να ανιχνεύσει και να αποδώσει ο αισθητήρας στην έξοδό του. [1]

- *Επαναληψιμότητα*

Η δυνατότητα του αισθητήρα να δίνει τις ίδιες τιμές όταν μετρά την ίδια ποσότητα σε ίδιες συνθήκες. Η επαναληψιμότητα εκφράζεται από το σφάλμα επαναληψιμότητας, δηλαδή το σφάλμα που παρουσιάζει ένας αισθητήρας όταν για το ίδιο ερέθισμα δεν δίνει το ίδιο σήμα εξόδου και που δίνεται από τον τύπο:

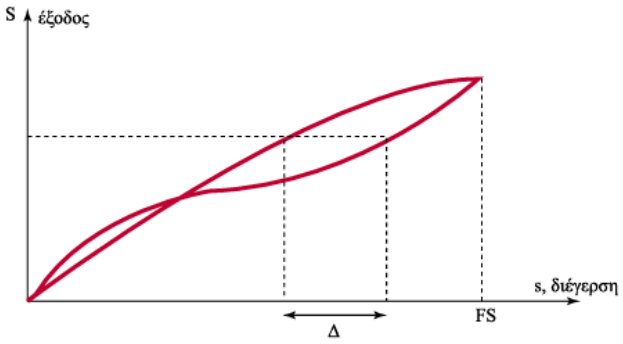
$$\boldsymbol{\delta_r = \frac{\Delta}{FS} * 100\%} \qquad \textbf{(2)}$$

Όπου:

δ_r: Είναι το **σφάλμα επαναληψιμότητας** (*repeatability error*) εκφρασμένο σε **ποσοστό (%)** του πλήρους εύρους μέτρησης.

Δ: Είναι η μέγιστη διαφορά μεταξύ επαναλαμβανόμενων μετρήσεων όταν μετράμε το ίδιο πραγματικό μέγεθος υπό τις ίδιες συνθήκες.

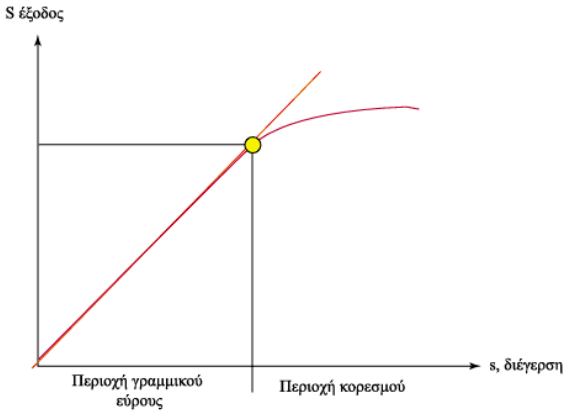
FS (*Full Scale*): Το πλήρες εύρος μέτρησης του αισθητήρα, δηλαδή το διάστημα από τη μικρότερη έως τη μεγαλύτερη τιμή που μπορεί να μετρήσει. [1]



Εικόνα 3.13: Καμπύλη σφάλματος επαναληψιμότητας

- *Συντελεστής κορεσμού*

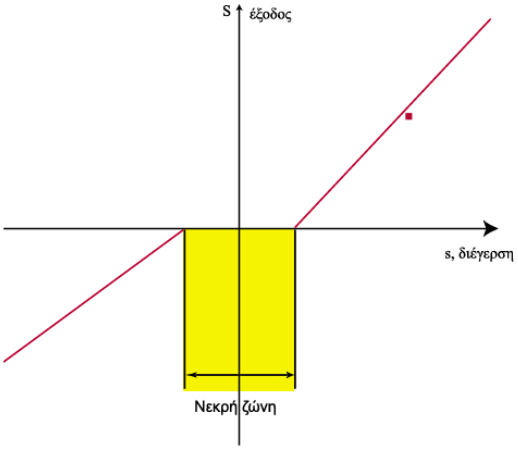
Ο συντελεστής κορεσμού είναι ένα χαρακτηριστικό που περιγράφει το πώς συμπεριφέρεται ο αισθητήρας όταν το ερέθισμα υπερβαίνει το εύρος στο οποίο μπορεί να αποκρίνεται γραμμικά. Ουσιαστικά εκφράζει τον βαθμό στον οποίο ο αισθητήρας σταματά να αυξάνει την έξοδό του, ακόμη κι αν το πραγματικό μέγεθος που μετρά συνεχίζει να αυξάνεται. [1]



Εικόνα 3.14: Σημείο εμφάνισης κορεσμού

- *Νεκρή ζώνη*

Η νεκρή ζώνη (dead zone ή dead band) ενός αισθητήρα είναι το εύρος τιμών του ερεθίσματος στο οποίο η έξοδος του αισθητήρα δεν παρουσιάζει καμία μεταβολή, ακόμη κι αν το πραγματικό μέγεθος αλλάζει. (Εικόνα 3.9)



Εικόνα 3.15: Καμπύλη με εμφάνιση νεκρής ζώνης

- *Σύνθετη αντίσταση*

Η σύνθετη αντίσταση εξόδου είναι η ισοδύναμη ηλεκτρική αντίσταση και αντίδραση που παρουσιάζει ο αισθητήρας στην έξοδό του προς το κύκλωμα μέτρησης. [1]

- *Διέγερση*

Η διέγερση είναι το ηλεκτρικό σήμα τροφοδοσίας (τάση ή ρεύμα) που απαιτείται για τη λειτουργία ενός αισθητήρα. [1]

- *Αξιοπιστία*

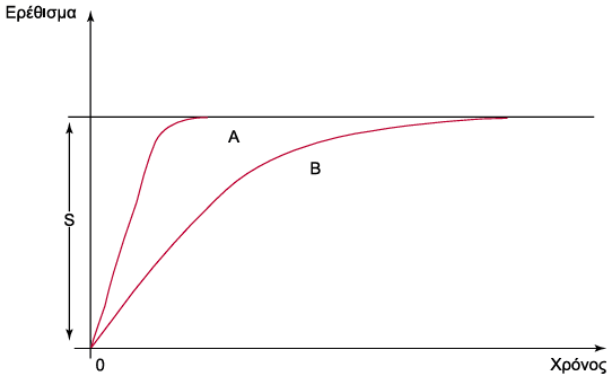
Η αξιοπιστία αναφέρεται στην ικανότητα του αισθητήρα να λειτουργεί σωστά και σταθερά για μεγάλο χρονικό διάστημα, υπό καθορισμένες συνθήκες περιβάλλοντος και χρήσης. [1]

- *Ελάχιστο σήμα κατωφλίου*

Το ελάχιστο σήμα κατωφλίου είναι η μικρότερη τιμή ερεθίσματος που μπορεί να ανιχνεύσει ο αισθητήρας και να τη διαχωρίσει από τον θόρυβο του συστήματος. Κάτω από αυτό το όριο, ο αισθητήρας δεν παρουσιάζει αξιόπιστη ή μετρήσιμη απόκριση. [1]

- *Χρόνος απόκρισης*

Ο χρόνος απόκρισης είναι το χρονικό διάστημα που απαιτείται ώστε ο αισθητήρας να φτάσει σε συγκεκριμένο ποσοστό της τελικής του τιμής μετά από μια ξαφνική μεταβολή του ερεθίσματος. [1]



Εικόνα 3.16: Καμπύλες χρόνου απόκρισης για δύο διαφορετικούς αισθητήρες

- *Θόρυβος*

Ο θόρυβος είναι οι τυχαίες, ανεπιθύμητες μεταβολές της ηλεκτρικής εξόδου του αισθητήρα, οι οποίες δεν σχετίζονται με το πραγματικό ερέθισμα. Ο θόρυβος μπορεί να προέλθει από ηλεκτρονικά στοιχεία, θερμικά φαινόμενα, παρεμβολές ή το περιβάλλον και μπορεί να μειώσει την ευαισθησία, την ακρίβεια και την κατωφλιακή ικανότητα ανίχνευσης. [1]

- *Ολίσθηση*

Η ολίσθηση είναι η αργή και σταδιακή μεταβολή της εξόδου ενός αισθητήρα με την πάροδο του χρόνου, χωρίς αντίστοιχη μεταβολή του ερεθίσματος. Οι παράγοντες που οδηγούν στην ολίσθηση συνήθως είναι περιβαλλοντικοί (θερμοκρασία, πίεση, υγρασία) καθώς επιδρούν στα επιμέρους τμήματα του αισθητήρα. [1]

- *Χρόνος προθέρμανσης*

Ο χρόνος προθέρμανσης είναι το χρονικό διάστημα που απαιτείται μετά την ενεργοποίηση του αισθητήρα ώστε να επιτευχθεί σταθερή και αξιόπιστη έξοδος. [1]

3.4 Επίλογος

Συνοψίζοντας, στο παρόν κεφάλαιο παρουσιάστηκαν οι βασικές κατηγορίες αισθητήρων που χρησιμοποιούνται για την παρακολούθηση της ποιότητας του αέρα, καθώς και τα κύρια λειτουργικά και τεχνικά χαρακτηριστικά που καθορίζουν την απόδοσή τους. Η κατανόηση των ιδιοτήτων αυτών, όπως η ευαισθησία, η γραμμικότητα, ο χρόνος απόκρισης και η αξιοπιστία, είναι απαραίτητη για την ορθή επιλογή και ενσωμάτωσή τους σε ένα ολοκληρωμένο σύστημα μέτρησης. Τα στοιχεία αυτά αποτελούν το θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται ο σχεδιασμός του προτεινόμενου συστήματος παρακολούθησης, εξασφαλίζοντας ότι οι αισθητήρες που θα χρησιμοποιηθούν ανταποκρίνονται στις συνθήκες που απαιτούν οι βιομηχανικές εφαρμογές.

Κεφάλαιο 4ο: Αισθητήρες του συστήματος και ESP32

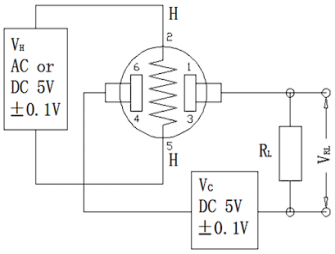
4.1 Επιλογή αισθητήρων

Η επιλογή των συγκεκριμένων αισθητήρων βασίστηκε στον συνδυασμό αξιοπιστίας, κόστους, ευαισθησίας και καταλληλότητας για ενσωμάτωση σε σύστημα συνεχούς παρακολούθησης ποιότητας αέρα σε βιομηχανικό περιβάλλον. Οι αισθητήρες **MQ-135**, **MiCS-2714** και **MiCS-5524**, ως αισθητήρες μεταλλικού οξειδίου (MOS), προσφέρουν υψηλή ευαισθησία σε κρίσιμους ρύπους όπως εύφλεκτα αέρια, NO_2, CO και VOCs, ενώ χαρακτηρίζονται από μικρό κόστος, υψηλή συμβατότητα και εύκολη ενσωμάτωση με τον μικροελεγκτή και δυνατότητα λειτουργίας σε ευρύ φάσμα θερμοκρασιών, στοιχεία που τους καθιστούν ιδανικούς για εφαρμογές IoT. Επιπλέον, έχουν γρήγορο χρόνο απόκρισης και ανθεκτικότητα σε συνθήκες βιομηχανικού περιβάλλοντος, όπου οι συγκεντρώσεις αερίων μεταβάλλονται δυναμικά. Για τη μέτρηση αιωρούμενων σωματιδίων χρησιμοποιήθηκε ο **DFRobot PM2.5 optical sensor**, ο οποίος βασίζεται στην τεχνολογία laser scattering και παρέχει υψηλή ακρίβεια και σταθερότητα για PM2.5 και PM10, επιτρέποντας αξιόπιστη εκτίμηση των σωματιδιακών φορτίων σε πραγματικό χρόνο. Ο συνδυασμός MOS και οπτικών αισθητήρων προσφέρει μια ολοκληρωμένη εικόνα της ποιότητας του αέρα, καλύπτοντας τόσο αέρια ρυπογόνα συστατικά όσο και αιωρούμενα σωματίδια, ενώ παράλληλα διατηρεί χαμηλό κόστος και υψηλή ενεργειακή απόδοση, στοιχεία απαραίτητα για το σχεδιασμό ενός αποδοτικού, φορητού και οικονομικά βιώσιμου συστήματος παρακολούθησης.

4.2 MQ-135 – Αισθητήρας Ανίχνευσης Αερίων (MOS)

Ο αισθητήρας **MQ-135** είναι αισθητήρας αερίων τύπου **Metal Oxide Semiconductor (MOS)**, σχεδιασμένος για την ανίχνευση αερίων που σχετίζονται με την υποβάθμιση της ποιότητας του αέρα. Επιπλέον, ο αισθητήρας παρουσιάζει υψηλή ευαισθησία σε αέρια όπως η αμμωνία, τα θειούχα αέρια και οι ατμοί της σειράς του βενζολίου, ενώ είναι επίσης ικανός να ανιχνεύει καπνό και άλλα τοξικά αέρια. Λόγω της δυνατότητάς του να ανιχνεύει ένα ευρύ φάσμα ρυπογόνων ουσιών και του χαμηλού κόστους του, χρησιμοποιείται ευρέως σε εφαρμογές παρακολούθησης της ποιότητας του αέρα.

Εικόνα 4.1: Αισθητήρας MQ-135



Εικόνα 4.2: Το εσωτερικό κύκλωμα του MQ-135

Το υλικό ανίχνευσης του αισθητήρα αερίων MQ-135 είναι το διοξείδιο του κασσιτέρου (SnO_2), το οποίο παρουσιάζει χαμηλή ηλεκτρική αγωγιμότητα σε συνθήκες καθαρού αέρα. Όταν στο περιβάλλον ανιχνεύονται ρυπογόνα ή τοξικά αέρια, η αγωγιμότητα του αισθητήρα αυξάνεται αναλογικά με τη συγκέντρωσή τους. [2]

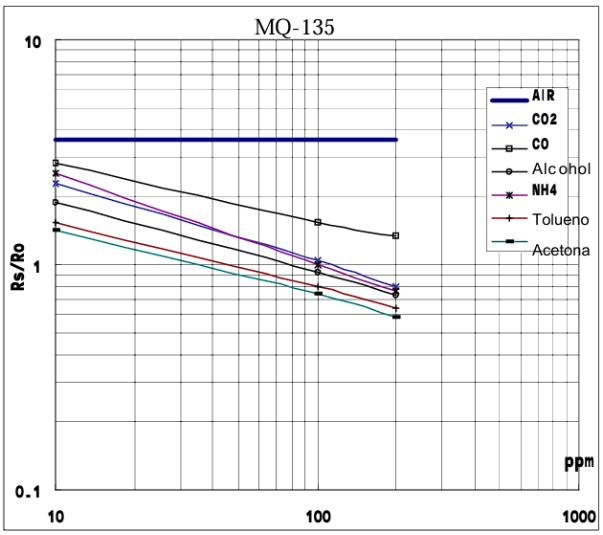
<u>Σημείωση(1)</u>: Παρά το γεγονός ότι ο MQ-135 μπορεί να παρουσιάσει απόκριση σε αέρια όπως αμμωνία (NH_3), διοξείδιο του αζώτου (NO_2), μονοξείδιο του άνθρακα (CO), διοξείδιο του άνθρακα (CO_2), καθώς και σε πτητικές οργανικές ενώσεις (VOCs), η έξοδός του εκφράζει **συνολική επιβάρυνση**

του αέρα και **όχι καθαρή συγκέντρωση ενός μόνο ρύπου.** Με άλλα λόγια ο MQ-135 είναι **αισθητήρας γενικής ποιότητας** αέρα και **όχι ενός συγκεκριμένου ρύπου**.

Σημείωση(2):

Ο αισθητήρας MQ-135 απαιτεί θέρμανση πριν από τη σωστή μέτρηση, λόγω της παρουσίας ενός ενσωματωμένου στοιχείου θέρμανσης που πρέπει να φτάσει σε σταθερή θερμοκρασία λειτουργίας. Για την αρχική χρήση συνιστάται **συνεχής λειτουργία για 24–48 ώρες ώστε να ολοκληρωθεί η διαδικασία προθέρμανσης** και να σταθεροποιηθούν οι μετρήσεις. [2].

Το διάγραμμα της παρακάτω εικόνας (Εικόνα 4.2) παρουσιάζει τις χαρακτηριστικές καμπύλες ευαισθησίας του αισθητήρα MQ-135 για διάφορα αέρια, εκφρασμένες ως λόγος R_s/R_0 σε συνάρτηση με τη συγκέντρωση σε ppm. Παρατηρείται ότι η αύξηση της συγκέντρωσης των αερίων οδηγεί σε μείωση της αντίστασης του αισθητήρα, γεγονός που αντανακλά της αύξηση της αγωγιμότητας του. Οι διαφορετικές καμπύλες για κάθε αέριο καταδεικνύουν τη μη εκλεκτική φύση του αισθητήρα, ο οποίος ανταποκρίνεται σε πλήθος ρυπογόνων ουσιών και χρησιμοποιείται κυρίως ως δείκτης συνολικής ποιότητας αέρα και όχι ως αισθητήρας συγκεκριμένου αερίου.



Εικόνα 4.3: Χαρακτηριστικές καμπύλες ευαισθησίας του αισθητήρα MQ-135 για διάφορα αέρια

Σημείωση:
Rs = εσωτερική αντίσταση αισθητήρα στη συγκεκριμένη συγκέντρωση αερίου.
R_0 = αντίσταση αισθητήρα **σε 100 ppm NH_3 στον καθαρό αέρα** (σημείο αναφοράς). Να σημειωθεί ότι η NH_3 δεν χρησιμοποιείται γιατί μετράμε αμμωνία, αλλά γιατί δίνει σταθερό σημείο αναφοράς για την κανονικοποίηση της απόκρισης του αισθητήρα.

4.2.1 Σχέση τάσης εξόδου – ποιότητας αέρα

Στο παρόν σύστημα, **η έξοδος του αισθητήρα λαμβάνεται υπό μορφή αναλογικής τάσης**, η οποία προκύπτει από τη μεταβολή της ηλεκτρικής αντίστασης του αισθητήρα παρουσία ρυπογόνων αερίων. Η **τάση αυτή δεν αντιστοιχεί άμεσα σε απόλυτη συγκέντρωση συγκεκριμένου αερίου**, αλλά χρησιμοποιείται ως δείκτης συνολικής ποιότητας αέρα. Συγκεκριμένα, αύξηση της τάσης εξόδου υποδηλώνει αυξημένη επιβάρυνση του αέρα, ενώ χαμηλότερες τιμές αντιστοιχούν σε καλύτερη ποιότητα.

Προκειμένου να αξιολογηθεί η ποιότητα του αέρα, υπολογίζεται ο λόγος Rs/R_0. Επιπλέον, η εσωτερική αντίσταση του αισθητήρα R_s μεταβάλλεται ανάλογα με τη συγκέντρωση του αερίου. Εάν η συγκέντρωση αερίου είναι υψηλή, η αντίσταση μειώνεται και εάν η συγκέντρωση αερίου είναι χαμηλή, η αντίσταση αυξάνεται. [3] Από τον λόγο αυτό και σύμφωνα με το Datasheet του MQ-135, προκύπτει ότι όσο μικρότερο το Rs/R_0 → τόσο μεγαλύτερη επιβάρυνση. [4]

Η διαδικασία με την οποία εξάγεται η ποιότητα του αέρα έχει ως εξής:

Λήψη τάσης εξόδου του MQ-135 → Υπολογισμός της R_s → Υπολογισμός του λόγου Rs/R_0 → Εκτίμηση ποιότητας αέρα

Υπολογισμός R_s:

$$\mathbf{Rs} = \boldsymbol{R_L}\frac{\boldsymbol{V_{cc}-V_{out}}}{\boldsymbol{V_{out}}} \qquad \textbf{(3)}$$

Αφού υπολογιστεί ο λόγος **Rs/R_0** εκτιμάται η ποιότητα του αέρα σύμφωνα με τον παρακάτω πίνακα:

Πίνακας 1: Επίπεδα ποιότητας αέρα (MQ-135)

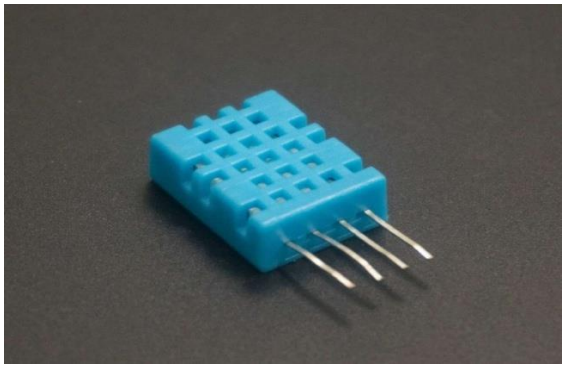
Rs/R_0	**Ποιότητα αέρα**
> 1.5	Πολύ καλή ποιότητα
1.0 – 1.5	Καλή
0.7 – 1.0	Μέτρια
0.4 – 0.7	Κακή
< 0.4	Πολύ κακή

<u>Συνοψίζοντας:</u>

Η εκτίμηση της ποιότητας του αέρα στον αισθητήρα MQ-135 βασίζεται στον λόγο Rs/R_0, όπου όπου Rs είναι η στιγμιαία αντίσταση του αισθητήρα και R_0 η αντίσταση αναφοράς που ορίζεται από τον κατασκευαστή. Η χρήση του λόγου αυτού επιτρέπει την κανονικοποίηση της απόκρισης του αισθητήρα και τη μείωση των επιδράσεων κατασκευαστικών ανοχών και περιβαλλοντικών παραμέτρων. Από τις χαρακτηριστικές καμπύλες ευαισθησίας (Εικόνα 4.2) προκύπτει ότι υψηλές τιμές του Rs/R_0 αντιστοιχούν σε συνθήκες χαμηλής επιβάρυνσης, ενώ η μείωσή του υποδηλώνει αυξανόμενη παρουσία ρυπογόνων αερίων. Με βάση τη σχετική αυτή μεταβολή, ορίστηκαν ημιποσοτικά επίπεδα ποιότητας αέρα, τα οποία χρησιμοποιούνται για τη συγκριτική αξιολόγηση της κατάστασης του περιβάλλοντος.

4.3 DHT11 – Αισθητήρας Θερμοκρασία και Σχετικής Υγρασίας

Ο αισθητήρας **DHT11** είναι ένας χαμηλού κόστους, ψηφιακός αισθητήρας **θερμοκρασίας** και **σχετικής υγρασίας**, σχεδιασμένος για εφαρμογές απλής περιβαλλοντικής παρακολούθησης. Ενσωματώνει στοιχείο μέτρησης υγρασίας και στοιχείο μέτρησης θερμοκρασίας, ενώ διαθέτει εσωτερική βαθμονόμηση και μετατρέπει τις μετρήσεις σε ψηφιακή έξοδο, μειώνοντας την ανάγκη για αναλογική επεξεργασία σήματος από τον μικροελεγκτή. Η επικοινωνία πραγματοποιείται μέσω single-wire σειριακής διεπαφής, γεγονός που απλοποιεί τη διασύνδεση με συστήματα όπως Arduino/ESP. Σύμφωνα με τα τεχνικά χαρακτηριστικά του, ο DHT11 λειτουργεί με τροφοδοσία περίπου 3.3-5 V, παρέχει μετρήσεις θερμοκρασίας στο εύρος 0-50 °C (τυπική ακρίβεια ±2 °C) και υγρασίας στο εύρος 20-90% RH (τυπική ακρίβεια περίπου ±5 %RH), με ρυθμό ανανέωσης 1 Hz (μια μέτρηση ανά δευτερόλεπτο. [5]



Εικόνα 4.4: DHT11

Ο **DHT11** επιλέχθηκε για την παρούσα εργασία επειδή ταιριάζει καλά σε ένα σύστημα παρακολούθησης περιβάλλοντος (ιδίως εσωτερικού χώρου) όπου προτεραιότητα είναι η απλότητα υλοποίησης , το χαμηλό κόστος και η επαρκής (όχι εργαστηριακή ακρίβεια). Παρακάτω αναφέρονται ορισμένα χαρακτηριστικά του αισθητήρα που οδήγησαν στην επιλογή του:

1. Επαρκές εύρος μετρήσεων για το σενάριο χρήσης

   Το εύρος 0–50 °C και 20–90 %RH καλύπτει τις συνθήκες που συναντώνται συνήθως σε εσωτερικούς χώρους.

2. Απλή διασύνδεση με μικροελεγκτή (λιγότερα καλώδια/λιγότερα σφάλματα)

   Η single-wire ψηφιακή επικοινωνία σημαίνει ότι χρειάζεται ουσιαστικά ένα pin δεδομένων (πέρα από τροφοδοσία/γείωση), άρα μειώνονται η πολυπλοκότητα καλωδίωσης και τα πιθανά προβλήματα θορύβου σε σχέση με αναλογικούς αισθητήρες.

3. Συμβατότητα τάσης με ESP32 και χαμηλή κατανάλωση

   Ο DHT11 λειτουργεί τυπικά σε **3–5 V**, άρα είναι πρακτικός τόσο για πλατφόρμες ESP32 χωρίς επιπλέον μετατροπές επιπέδων στις περισσότερες υλοποιήσεις.

4. Ρυθμός δειγματοληψίας κατάλληλος για περιβαλλοντικές μεταβολές

   Ο περιορισμός σε 1 Hz δεν είναι μειονέκτημα σε εφαρμογές όπου η θερμοκρασία/υγρασία μεταβάλλονται σχετικά αργά· αντίθετα, βοηθά σε σταθερές αναγνώσεις και απλή επεξεργασία.

5. Πρακτικότητα και διαθεσιμότητα βιβλιοθηκών

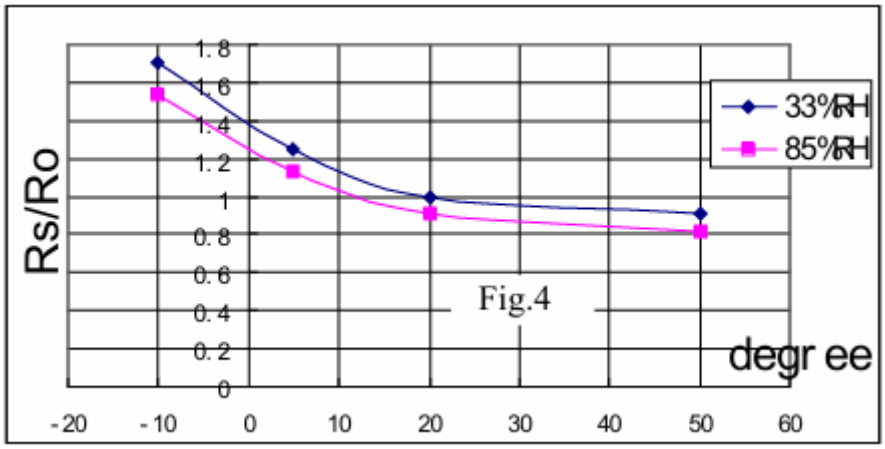
Υπάρχει ευρεία υποστήριξη/τεκμηρίωση από κοινότητες και εκπαιδευτικά υλικά, κάτι που μειώνει τον χρόνο ανάπτυξης και τον κίνδυνο σφαλμάτων ενσωμάτωσης.

4.3.1 Ο ρόλος του DHT11 στο σύστημα παρακολούθησης ποιότητας αέρα.

Στο παρόν σύστημα παρακολούθησης ποιότητας αέρα σε εσωτερικό βιομηχανικό χώρο, ο DHT11 δεν χρησιμοποιείται τόσο για τον ακριβή προσδιορισμό της θερμοκρασία και υγρασίας όσο ως υποστηρικτικός αισθητήρας για την εύρυθμη λειτουργία του συνολικού συστήματος.

4.3.2 DHT11 και MQ-135

Ο υποστηρικτικός ρόλος του DHT11 εμφανίζεται σε όλους τους MOS αισθητήρες του παρόντος συστήματος και ιδιαίτερα στον MQ-135 (λόγω παλαιότητας, μεγέθους και λιγότερο ελεγχόμενης θερμικής και επιφανειακής συμπεριφοράς) [4], [6], [7]. Όπως αναφέρθηκε σε προηγούμενη ενότητα, ο αισθητήρας MQ-135 είναι αισθητήρας μεταλλικού οξειδίου, του οποίου η έξοδος βασίζεται στη μεταβολή της ηλεκτρικής αντίστασης, η οποία επηρεάζεται όχι μόνο από τη συγκέντρωση αερίων αλλά και από τις περιβαλλοντικές συνθήκες. Η θερμοκρασία και η σχετική υγρασία μπορούν να προκαλέσουν μετατόπιση της γραμμής βάσης των μετρήσεων, οδηγώντας σε ψευδείς ενδείξεις επιδείνωσης της ποιότητας του αέρα. Για τον λόγο αυτό, οι μετρήσεις του DHT11 αξιοποιούνται ώστε να λαμβάνονται υπόψη οι συνθήκες περιβάλλοντος και να βελτιώνεται η αξιοπιστία της συνολικής εκτίμησης. Τέλος, μειώνεται η πιθανότητα ψευδών συναγερμών και βελτιώνεται η ερμηνεία των δεδομένων ποιότητας αέρα σε πραγματικές συνθήκες λειτουργίας.



Εικόνα 4.5: Τυπική εξάρτηση του λόγου Rs/R0 από την θερμοκρασία και την υγρασία

Από το γράφημα της εικόνας 4.3 προκύπτουν τα εξής:

Α) Επιδραση υγρασίας:
Για την ίδια θερμοκρασία (π.χ. 20 °C)

- χαμηλή RH (33%) → μεγαλύτερο Rs/R0
- υψηλή RH (85%) → μικρότερο Rs/R0

Συμπέρασμα: Η αυξημένη υγρασία μειώνει την αντίσταση του αισθητήρα. Όμως, όπως αναφέρθηκε και στην ενότητα **4.2.1**, μειωμένη αντίσταση του αισθητήρα σημαίνει αυηξημένη συγκέντρωση αερίου. Επομένως ο αισθητήρας "νομίζει" ότι υπάρχει περισσότερο αέριο, ενώ στην πραγματικότητα άλλαξε μόνο η RH.

Β) Επίδραση θερμοκρασίας

Για σταθερό επίπεδο υγρασίας (π.χ. 33%) και αυξανόμενη θερμοκρασία, παρατηρείται μη γραμμική μεταβολή του λόγου Rs/R0.

Συμπέρασμα: Η θερμοκρασία επηρεάζει τη χημική ισορροπία της επιφάνειας SnO_2 του αισθητήρα.

Γ) Ιδανική περιοχή λειτουργίας-ελάχιστη αβεβαιότητα

Στην περιοχή θερμοκρασιών 20-30 °C και για μέτρια RH (33%), παρατηρείται γραμμική λειτουργία του αισθητήρα. Αυτή είναι η **ιδανική περιοχή λειτουργίας** όπου ο αισθητήρας είναι πιο **σταθερός και προβλέψιμος.**

Δ) Περιοχή αυξημένης αβεβαιότητας

Τέλος, ορίζεται μία περιοχή αυξημένης αβεβαιότητας των μετρήσεων. Στην περιοχή αυτή οι μετρήσεις δεν απορρίπτονται αλλά εμφανίζονται ορισμένες προειδοποιήσεις για την αξιοπιστία τους ώστε να αποφευχθούν λανθασμένες εκτιμήσεις ποιότητας αέρα.

Η περιοχή αυξημένης αβεβαιότητας ορίζεται προσεγγιστικά σύμφωνα με το datasheet του αισθητήρα και φαίνεται στον παρακάτω πίνακα:

Πίνακας 2: Περιοχή αυξημένης αβεβαιότητας

Παράμετρος	**Τιμές**
Θερμοκρασία	> 45 °C
Σχετική υγρασία	> 80 %RH

4.4 MiCS-2714

Ο MiCS-2714 είναι ένας αισθητήρας **οξειδωτικών** αερίων τεχνολογίας MOS (Metal Oxide Semiconductor), σχεδιασμένος για την ανίχνευση πολύ χαμηλών συγκεντρώσεων (~0.05–10 ppm) διοξειδίου του αζώτου (NO_2), καθώς και υδρογόνου (H_2) (~1–1000 ppm). Η αρχή λειτουργίας του βασίζεται στη μεταβολή της ηλεκτρικής αντίστασης του ευαίσθητου υλικού (συνήθως SnO_2) όταν αυτό εκτίθεται σε συγκεκριμένα αέρια, ενώ η απαιτούμενη θερμοκρασία λειτουργίας επιτυγχάνεται μέσω ενσωματωμένου μικροθερμαντήρα. Ο αισθητήρας χαρακτηρίζεται από μικρό μέγεθος, χαμηλή κατανάλωση ισχύος και γρήγορο χρόνο απόκρισης, στοιχεία που τον καθιστούν κατάλληλο για συστήματα συνεχούς παρακολούθησης ποιότητας αέρα. Η αναλογική εξοδός του επιτρέπει την άμεση διασύνδεση με μικροελεγκτές, όπως ο ESP32, και τη συλλογή δεδομένων σε πραγματικό χρόνο, ακόμη και σε απαιτητικά περιβάλλοντα όπως οι βιομηχανικοί χώροι. [6]

Εικόνα 4.6: Αισθητήρας αερίων MiCS-2714

Βασικά τεχνικά χαρακτηριστικά:

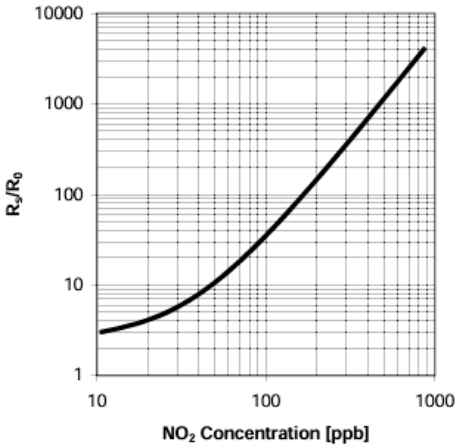
- **Τροφοδοσία**: περίπου 5 V DC.
- **Εύρος θερμοκρασίας λειτουργίας**: από ~–30 °C έως +85 °C.
- **Υγρασία λειτουργίας**: 5 – 95 %RH χωρίς συμπύκνωση.
- Συχνά χρειάζεται **προθέρμανση μερικών λεπτών** πριν τη μέτρηση για σταθερά αποτελέσματα.

4.4.1 Ο ρόλος του MiCS-2714 στο σύστημα παρακολούθησης ποιότητας αέρα

Η επιλογή του συγκεκριμένου αισθητήρα έγινε με στόχο την αξιόπιστη ανίχνευση ρύπων που εμφανίζονται συχνά σε βιομηχανικά περιβάλλοντα και σχετίζονται άμεσα με κινδύνους για την υγεία και την ασφάλεια του προσωπικού. Το διοξείδιο του αζώτου (NO_2), που μπορεί να παραχθεί από καύσεις, μηχανές ή βιομηχανικές διεργασίες, είναι τοξικό ακόμη και σε χαμηλές συγκεντρώσεις· συνεπώς η έγκαιρη ανίχνευσή του συμβάλλει στην πρόληψη αναπνευστικών προβλημάτων και την αποφυγή επικίνδυνων συνθηκών εργασίας. Σε συνδυασμό με τον **MQ-135**, τον **MiCS-5524**, τον **DHT11** και τον **DFRobot PM2.5 optical sensor**, ο MiCS-2714 ενισχύει την πληρότητα της παρακολούθησης, προσφέροντας εξειδικευμένη ευαισθησία σε οξειδωτικά αέρια.

4.4.2 Επεξεργασία σήματος – ερμηνεία μετρήσεων

Η απόκριση του αισθητήρα χαρακτηρίζεται από μη γραμμική συμπεριφορά [6] (Εικόνα 4.5) γεγονός που καθιστά δύσκολη την άμεση και αξιόπιστη μετατροπή της εξόδου του σε απόλυτες τιμές συγκέντρωσης (ppm). Για τον λόγο αυτό, στο παρόν σύστημα η πληροφορία του αισθητήρα αξιοποιείται κυρίως σε επίπεδο σχετικών μεταβολών και κατωφλίων, επιτρέποντας την έγκαιρη ανίχνευση επικίνδυνων καταστάσεων, ιδιαίτερα σε βιομηχανικά περιβάλλοντα όπου η πρόληψη έχει μεγαλύτερη σημασία από την ακριβή ποσοτικοποίηση.



Εικόνα 4.7: RS/R0 σε συνάρτηση με τη συγκέντρωση NO2 (40% RH και 25°C)

Στο παραπάνω γράφημα φαίνεται η εξάρτηση του λόγου RS/R0 από την συγκέντρωση NO2. Όπου **Rs**= η **στιγμιαία αντίσταση** του αισθητήρα όταν "βλέπει" NO_2 και **R0** = η **αντίσταση αναφοράς σε καθαρό αέρα** (baseline) υπό συγκεκριμένες συνθήκες/μετά από προθέρμανση. Σύμφωνα με το datasheet του αισθητήρα **η Rs αυξάνει παρουσία NO_2**, άρα **όσο ανεβαίνει το NO_2, συνήθως ανεβαίνει και το Rs/R0**. [6]

Η ερμηνεία των μετρήσεων του MiCS-2714 ακολουθεί το μοτίβο του MQ-135:

Λήψη τάσης εξόδου του MQ-135 → Υπολογισμός της R_s → Υπολογισμός του λόγου R_s/R_0 → Εκτίμηση ποιότητας αέρα

<u>Σημείωση:</u>

Η εκτίμηση της ποιότητας και ο διαχωρισμός των επιπέδων ρύπανσης του αέρα βασίζεται στο datasheet του αισθητήρα και με γνώμονα τα διεθνή αποδεκτά όρια έκθεσης εργαζομένων σε ρύπους (**ACGIH TLV-TWA 0.2 ppm**, **NIOSH REL (ST) 1 ppm**, **OSHA PEL ceiling 5 ppm**, **NIOSH IDLH 20 ppm)**. [8]

Η αντίσταση Rs του αισθητήρα υπολογίζεται από τη μετρούμενη τάση εξόδου Vout, θεωρώντας κύκλωμα διαιρέτη τάσης με αντίσταση φορτίου RL, σύμφωνα με τη σχέση:

$$\mathbf{Rs} = R_L \frac{V_{cc} - V_{out}}{V_{out}} \qquad \mathbf{(4)}$$

Αφού υπολογιστεί ο λόγος **R_s/R_0** εκτιμάται η ποιότητα του αέρα σύμφωνα με τον παρακάτω πίνακα:

Πίνακας 3: Επίπεδα ποιότητας αέρα (MiCS-2714)

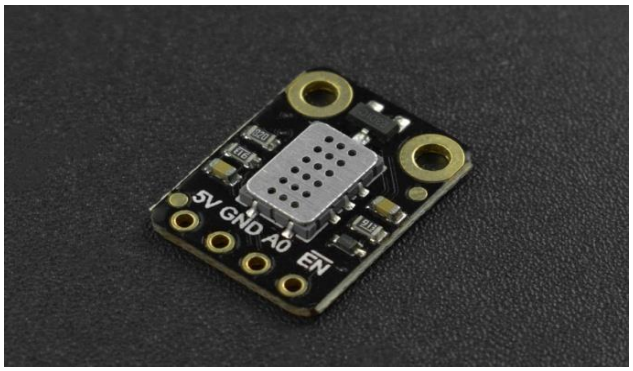
Rs/R0 (ενδεικτικά)	**Συγκέντρωση NO_2 (ppb)**	**Επίπεδο ποιότητας αέρα**
< 0.5	< 50 ppb	Πολύ καλή
0.5 – 1.0	50 – 100 ppb	Καλή
1.0 – 2.0	100 – 200 ppb	Μέτρια
2.0 – 5.0	200 – 500 ppb	Κακή
5.0-10.0	500-1000 ppb	Πολύ κακή / Κρίσιμη

Συνοψίζοντας:

Με βάση τις χαρακτηριστικές καμπύλες του datasheet του αισθητήρα MiCS-2714, ο λόγος Rs/R0 χρησιμοποιήθηκε ως βασικό μέγεθος εκτίμησης της συγκέντρωσης NO_2. Στον Πίνακα 3 παρουσιάζονται ενδεικτικά επίπεδα ποιότητας αέρα σε μονάδες ppb, όπως προκύπτουν από την τυπική απόκριση του αισθητήρα. Η προσέγγιση αυτή επιτρέπει τον καθορισμό κατωφλίων συναγερμού και την έγκαιρη ανίχνευση επικίνδυνων καταστάσεων, χωρίς την απαίτηση πλήρους εργαστηριακής βαθμονόμησης.

4.5 MiCS-5524

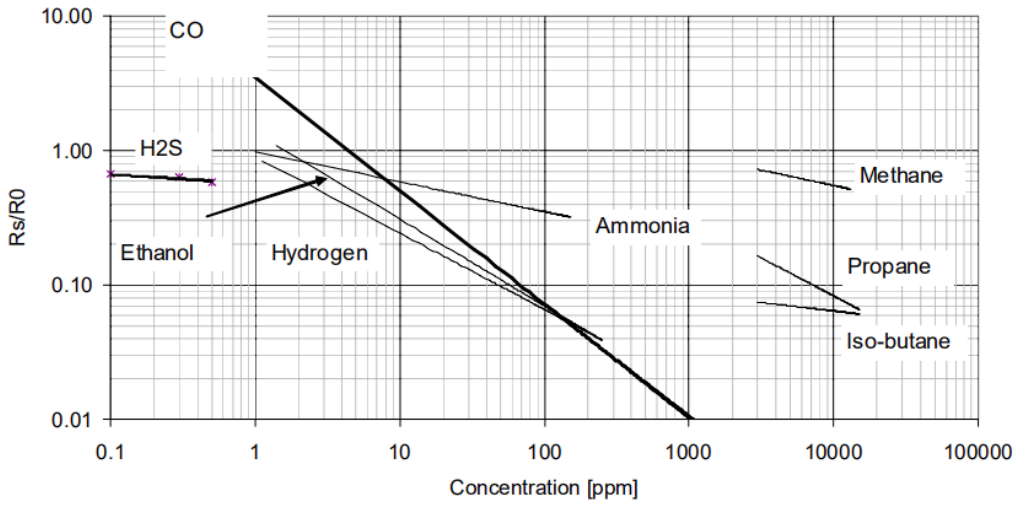
Ο MiCS-5524 είναι αισθητήρας **αναγωγικών** αερίων τεχνολογίας MOS (Metal Oxide Semiconductor), σχεδιασμένος κυρίως για την ανίχνευση αναγωγικών αερίων όπως το μονοξείδιο του άνθρακα (CO), το υδρογόνο (H_2), το μεθάνιο (CH_4) και πτητικές οργανικές ενώσεις (VOCs (προπάνιο, αιθανόλη, βουτάνιο κλπ)). Η αρχή λειτουργίας του βασίζεται στη μεταβολή της ηλεκτρικής αντίστασης ενός ημιαγώγιμου μεταλλικού οξειδίου, όταν αυτό εκτίθεται σε αναγωγικά αέρια υπό αυξημένη θερμοκρασία λειτουργίας, η οποία επιτυγχάνεται μέσω ενσωματωμένου μικροθερμαντήρα. Ο αισθητήρας χαρακτηρίζεται από μικρό μέγεθος, χαμηλή κατανάλωση ισχύος και γρήγορη απόκριση, γεγονός που τον καθιστά κατάλληλο για συστήματα συνεχούς παρακολούθησης ποιότητας αέρα. Η αναλογική του έξοδος επιτρέπει την εύκολη διασύνδεσή του με τον ESP32, και την αξιοποίηση του σε εφαρμογές βιομηχανικής ασφάλειας όπου απαιτείται έγκαιρη ανίχνευση επικίνδυνων αερίων που σχετίζονται με καύσεις, διαρροές ή βιομηχανικές διεργασίες.



Εικόνα 4.8: Αισθητήρας αερίων MiCS-5524

4.5.1 Ερμηνεία μετρήσεων-Αξιολόγηση της ποιότητας του αέρα.

Η αξιολόγηση της ποιότητας του αέρα γίνεται και στην περίπτωση του MiCS-5524 με τον υπολογισμό του λόγου **Rs/R0**. Όπου **Rs**: η αντίσταση του αισθητήρα τη στιγμή της μέτρησης (όταν υπάρχει αέριο) και **R0**: η αντίσταση "αναφοράς" σε αέρα (baseline). Στο datasheet του αισθητήρα δίνονται τυπικά όρια για το R0 σε αέρα. Παρακάτω φαίνεται το λογαριθμικό γράφημα του αισθητήρα για τα διάφορα αέρια μέτρησης.



Εικόνα 4.9: Διάγραμμα Rs/R0 και συγκέντρωσης [ppm] (MiCS-5524, 50% RH, 25°C)

Όπως φαίνεται στο παραπάνω γράφημα, ο MiCS-5524 παρουσιάζει μη γραμμική απόκριση ως προς τη συγκέντρωση των ανιχνευόμενων αερίων, χαρακτηριστικό που είναι σύνηθες σε αισθητήρες αερίων τεχνολογίας MOS. Συγκεκριμένα η μεταβολή της ηλεκτρικής αντίστασης του ευαίσθητου υλικού **δεν** είναι ανάλογη της μεταβολής της συγκέντρωσης του ρύπου, αλλά ακολουθεί λογαριθμική ή εκθετική σχέση. Η μη γραμμική αυτή συμπεριφορά καθιστά δύσκολη την άμεση και αξιόπιστη μετατροπή της αναλογικής εξόδου του αισθητήρα σε απόλυτες τιμές συγκέντρωσης (ppm/ppb) χωρίς διαδικασία βαθμονόμησης. Για τον λόγο αυτό, στο παρόν σύστημα ο αισθητήρας αξιοποιείται κυρίως ως δείκτης μεταβολών και επιπέδων ρύπανσης μέσω του λόγου Rs/R0.

Επιπλέον, ο MiCS-5524 είναι αισθητήρας **αναγωγικών** αερίων (CO, VOCs/ethanol κ.λπ.) σε αντίθεση με τον MiCS-2714 που είναι αισθητήρας **οξειδωτικών** αερίων (κυρίως NO_2). Το γεγονός αυτό έχει ως αποτέλεσμα την **μείωση του λόγου Rs/R0 καθώς αυξάνει η συγκέντρωση των ρύπων**. Αυτός είναι και ο λόγος που στα γραφήματα των δύο αισθητήρων παρατηρούνται **αντίθετες κλίσεις**.

Τέλος, ο αισθητήρας MiCS-5524 παρουσιάζει **διασταυρούμενη ευαισθησία** σε πληθώρα αναγωγικών αερίων και πτητικών οργανικών ενώσεων, με αποτέλεσμα η αναλογική έξοδός του να αντιπροσωπεύει τη συνδυασμένη επίδραση των παρόντων ρύπων. Κατά συνέπεια, ο αισθητήρας δεν είναι σε θέση να ταυτοποιήσει μεμονωμένα το είδος του αερίου, αλλά χρησιμοποιείται ως δείκτης αυξημένης συγκέντρωσης αναγωγικών αερίων. Η διάκριση μεταξύ διαφορετικών ρύπων καθίσταται εφικτή μόνο μέσω συνδυαστικής αξιολόγησης με άλλους αισθητήρες του συστήματος ή με βάση τη γνώση του περιβάλλοντος εφαρμογής.

Επειδή το μονοξείδιο του άνθρακα (CO) και η Αιθανόλη (C_2H_5OH) είναι οι κύριοι ρύποι (από αυτούς που ανιχνεύει ο αισθητήρας) που εντοπίζονται στα βιομηχανικά περιβάλλοντα, δημιουργήθηκαν δύο αντίστοιχοι πίνακες για την αξιολόγηση της ποιότητας του αέρα. Έτσι, ανάλογα με τον τύπο των ρύπων που αναμένονται στον εκάστοτε βιομηχανικό χώρο, γίνεται και η αντίστοιχη αναδρομή στον κατάλληλο πίνακα. Οι τιμές Rs/R0 που φαίνονται παρακάτω είναι **ενδεικτικά thresholds** που προκύπτουν από ανάγνωση/προσεγγιστική παρεμβολή της τυπικής καμπύλης του datasheet. [9]

Πίνακας 4: Ενδεικτική αντιστοίχιση Rs/R0 σε συγκέντρωση CO (τυπική καμπύλη datasheet)

Rs/R0 (ενδεικτικά)	**Συγκέντρωση CO (ppm)**	**Επίπεδο CO**
~3.0 → ~0.8	1-5	Πολύ χαμηλό
~0.8 → ~0.45	5-10	Χαμηλό–Μέτριο
~0.45 → ~0.12	10-50	Αυξημένο
~0.12 → ~0.038	50-200	Υψηλό
~0.038 → ~0.01	200-1000	Πολύ υψηλό

Πίνακας 5: Ενδεικτική αντιστοίχιση Rs/R0 σε συγκέντρωση Αιθανόλης (τυπική καμπύλη datasheet)

Rs/R0 (ενδεικτικά)	**Συγκέντρωση Αιθανόλης (ppm)**	**Επίπεδο Αιθανόλης**
~0.35 → ~0.26	10-20	Πολύ χαμηλό
~0.26 → ~0.17	20-50	Χαμηλό–Μέτριο
~0.17 → ~0.12	50-100	Αυξημένο
~0.12 → ~0.09	100-200	Υψηλό
~0.09 → ~0.06	200-500	Πολύ υψηλό

Σημείωση: Οι πίνακες χρησιμοποιούνται ως σενάρια/thresholds για ενεργοποίηση συναγερμών όταν το μετρούμενο μίγμα ρύπων είναι κυρίως CO (καύσεις) ή Αιθανόλη και VOC-like (διαλύτες). **Δεν αποτελούν μέθοδο ταυτοποίησης αερίου ούτε πιστοποιημένη ποσοτικοποίηση**.

Η αντίσταση Rs του αισθητήρα υπολογίζεται από τη μετρούμενη τάση εξόδου Vout, θεωρώντας κύκλωμα διαιρέτη τάσης με αντίσταση φορτίου RL, σύμφωνα με τη σχέση:

$$\mathbf{Rs} = \boldsymbol{R_L}\frac{\boldsymbol{V_{cc}}-\boldsymbol{V_{out}}}{\boldsymbol{V_{out}}} \qquad \textbf{(5)}$$

Συνοψίζοντας:

Ο αισθητήρας MiCS-5524 ενσωματώθηκε στο σύστημα ως MOS αισθητήρας **αναγωγικών αερίων**, κατάλληλος για την ανίχνευση αυξημένων επιπέδων ρύπανσης που σχετίζονται με **CO** και **πτητικές οργανικές ενώσεις (VOCs)**, όπως ατμοί αλκοολών/διαλυτών. Η έξοδός του μεταβάλλεται **μη γραμμικά** και παρουσιάζει **διασταυρούμενη ευαισθησία,** δηλαδή η ίδια ένδειξη μπορεί να προκληθεί από διαφορετικά αέρια · για τον λόγο αυτό, αντί για «ταυτοποίηση» αερίου χρησιμοποιείται ο λόγος **Rs/R0** ως δείκτης μεταβολής σε σχέση με μια τιμή αναφοράς (R0) που βαθμονομείται σε συνθήκες βάσης. Στην εργασία η ερμηνεία των μετρήσεων γίνεται με δύο πίνακες/σεναριακές αντιστοιχίσεις (CO και Αιθανόλης/VOC-like) σύμφωνες με τις χαρακτηριστικές καμπύλες του datasheet, ενώ για σκοπούς ασφάλειας υιοθετείται **συντηρητική αξιολόγηση** (worst-case), ώστε το σύστημα να ενεργοποιεί έγκαιρα προειδοποιήσεις σε βιομηχανικό περιβάλλον ακόμη και όταν η ακριβής σύσταση του μίγματος ρύπων δεν είναι γνωστή.

4.6 DFRobot PM2.5 optical sensor

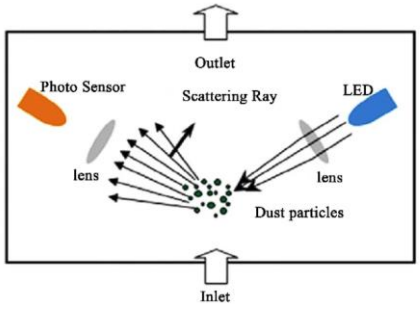
Ο **DFRobot PM2.5 Optical Sensor**, της εταιρείας **DFRobot**, είναι ένας οπτικός αισθητήρας μέτρησης αιωρούμενων σωματιδίων (Particulate Matter), ο οποίος χρησιμοποιείται για την ανίχνευση και ποσοτικοποίηση των συγκεντρώσεων **PM1.0, PM2.5 και PM10** στον αέρα. Τα σωματίδια αυτά αποτελούν βασικό δείκτη της ποιότητας του αέρα, καθώς σχετίζονται άμεσα με επιπτώσεις στην ανθρώπινη υγεία, ιδιαίτερα στο αναπνευστικό σύστημα.

Η επιλογή του συγκεκριμένου αισθητήρα έγινε λόγω της υψηλής αξιοπιστίας των μετρήσεών του και της ψηφιακής διεπαφής επικοινωνίας I2C, η οποία μειώνει τον θόρυβο και απλοποιεί την ενσωμάτωσή του σε μικροελεγκτικά συστήματα. Επιπλέον, δεν απαιτεί μεγάλο χρόνο προθέρμανσης και επηρεάζεται σε μικρό βαθμό από μεταβολές θερμοκρασίας και υγρασίας. Τα χαρακτηριστικά αυτά, σε συνδυασμό με το προσιτό του κόστος τον καθιστούν κατάλληλο για εφαρμογές συνεχούς παρακολούθησης της ποιότητας του αέρα.

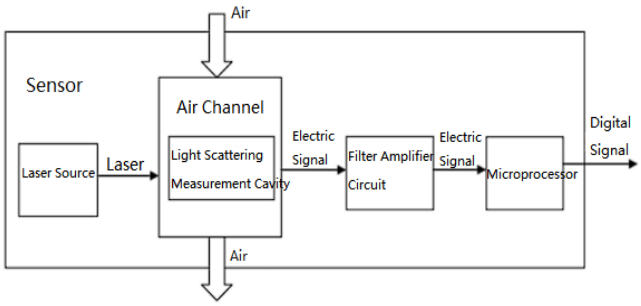
Εικόνα 4.10: DFRobot PM2.5 Optical Sensor

4.6.1 Αρχή λειτουργίας

Ο αισθητήρας βασίζεται στην αρχή της οπτικής σκέδασης λέιζερ για την ανίχνευση αιωρούμενων σωματιδίων στον αέρα. Κατά τη λειτουργία του, ένα λέιζερ φωτίζει τα σωματίδια που διέρχονται από τον θάλαμο μέτρησης και το φως που σκεδάζεται ανιχνεύεται από φωτοευαίσθητο στοιχείο. Από την ένταση και τα χαρακτηριστικά της σκέδασης υπολογίζεται το μέγεθος και η συγκέντρωση των σωματιδίων, επιτρέποντας την εξαγωγή τιμών PM1.0 , PM2.5 και PM10 σε μονάδες μg/m³. [10]



Εικόνα 4.11: Αρχή λειτουργίας οπτικού αισθητήρα



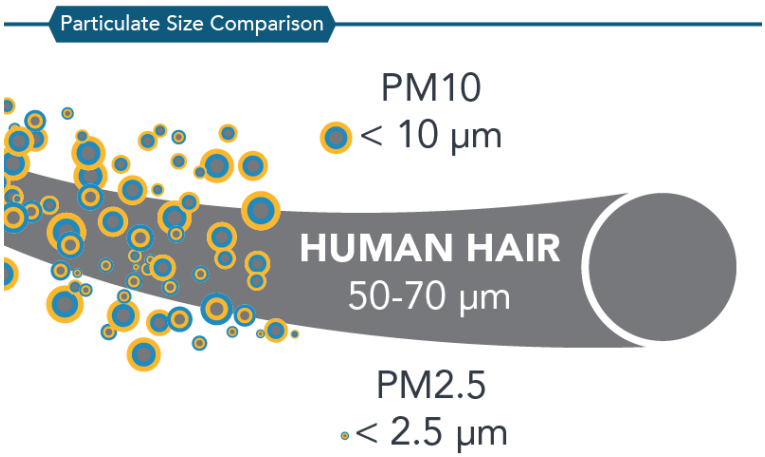
Εικόνα 4.12: Διάγραμμα ροής λειτουργίας οπτικού αισθητήρα

4.6.2 Ο ρόλος του οπτικού αισθητήρα στο σύστημα

Ο DFRobot PM2.5 Optical Sensor διαδραματίζει καθοριστικό ρόλο στο σύστημα μέτρησης ποιότητας αέρα, καθώς παρέχει πληροφορίες για τη σωματιδιακή ρύπανση, η οποία δεν μπορεί να ανιχνευθεί από αισθητήρες αερίων. Οι μετρήσεις PM1.0, PM2.5 και PM10 συμπληρώνουν τα δεδομένα των χημικών αισθητήρων, προσφέροντας μια ολοκληρωμένη εικόνα της ποιότητας του αέρα. Με τον τρόπο αυτό, το σύστημα μπορεί να αξιολογεί με μεγαλύτερη ακρίβεια τις συνθήκες του περιβάλλοντος και να υποστηρίζει εφαρμογές προειδοποίησης ή ανάλυσης σε πραγματικό χρόνο.

4.6.3 Particulate Matter (PM) και όρια έκθεσης

Το αιωρούμενο σωματιδιακό υλικό (Particulate Matter – PM) αποτελείται από μικροσκοπικά στερεά και υγρά σωματίδια που αιωρούνται στην ατμόσφαιρα και διακρίνονται κυρίως σε **PM10** και **PM2.5**, ανάλογα με τη διάμετρό τους. Τα σωματίδια PM10 (≤10 μm) μπορούν να εισέλθουν στο αναπνευστικό σύστημα και να προκαλέσουν επιδείνωση αναπνευστικών παθήσεων, ενώ τα λεπτότερα σωματίδια PM2.5 (≤2.5 μm) διεισδύουν βαθύτερα στους πνεύμονες και έχουν συσχετιστεί με σοβαρότερες επιπτώσεις στην υγεία, όπως καρδιοαναπνευστικές παθήσεις, άσθμα και αυξημένη θνησιμότητα. Οι κύριες πηγές των σωματιδίων αυτών περιλαμβάνουν διαδικασίες καύσης καυσίμων, εκπομπές από οχήματα και βιομηχανικές δραστηριότητες, καθώς και δευτερογενείς χημικές αντιδράσεις στην ατμόσφαιρα. [11]



Εικόνα 4.13: Σύγκριση μεγέθους αιωρούμενων σωματιδίων [11]

Μια μεμονωμένη, στιγμιαία μέτρηση συγκέντρωσης σωματιδίων PM δεν αρκεί για να καθοριστεί αν ο αέρας είναι πραγματικά ρυπασμένος ή επικίνδυνος για την υγεία. Τα διεθνώς χρησιμοποιούμενα πρότυπα ποιότητας αέρα – όπως αυτά θεσπίζονται από περιβαλλοντικές και υγειονομικές υπηρεσίες – βασίζονται στις **μέσες συγκεντρώσεις σε βάθος χρόνου**, συνήθως σε **24ωρη βάση**, επειδή αυτό αντικατοπτρίζει καλύτερα την έκθεση του ανθρώπου σε διάφορους περιβαλλοντικούς παράγοντες και μπορεί να συσχετιστεί με τεκμηριωμένα επιδημιολογικά δεδομένα. Μια παροδική αύξηση μπορεί να οφείλεται σε πρόσκαιρη πηγή (π.χ. κοντινό καύσιμο, εργασία, επεισόδιο καπνού) και να μην αποτελεί δείκτη υψηλού κινδύνου από μόνη της· η **συνεχής ή μέση έκθεση σε υψηλά επίπεδα 24 ώρες την ημέρα** είναι αυτή που αποδεικνύεται πιο έγκυρος δείκτης για εκτίμηση επιπτώσεων και λήψη αποφάσεων σχετικά με τη δημόσια υγεία.

	PM2.5 (μg/m³)		PM10 (μg/m³)	
	24-Hour Limit	Annual Limit	24-Hour Limit	Annual Limit
EU Standard (2008)	-	25	50	40
EU Standard (2024)	25	10	45	20
WHO Standard	15	5	45	15

Εικόνα 4.14: Επιτρεπόμενα όρια έκθεσης σε αιωρούμενα σωματίδια PM2.5 και PM10 σύμφωνα με τα παγκόσμια και ευρωπαϊκά πρότυπα ποιότητας αέρα (πηγή: Smart Air Filters).

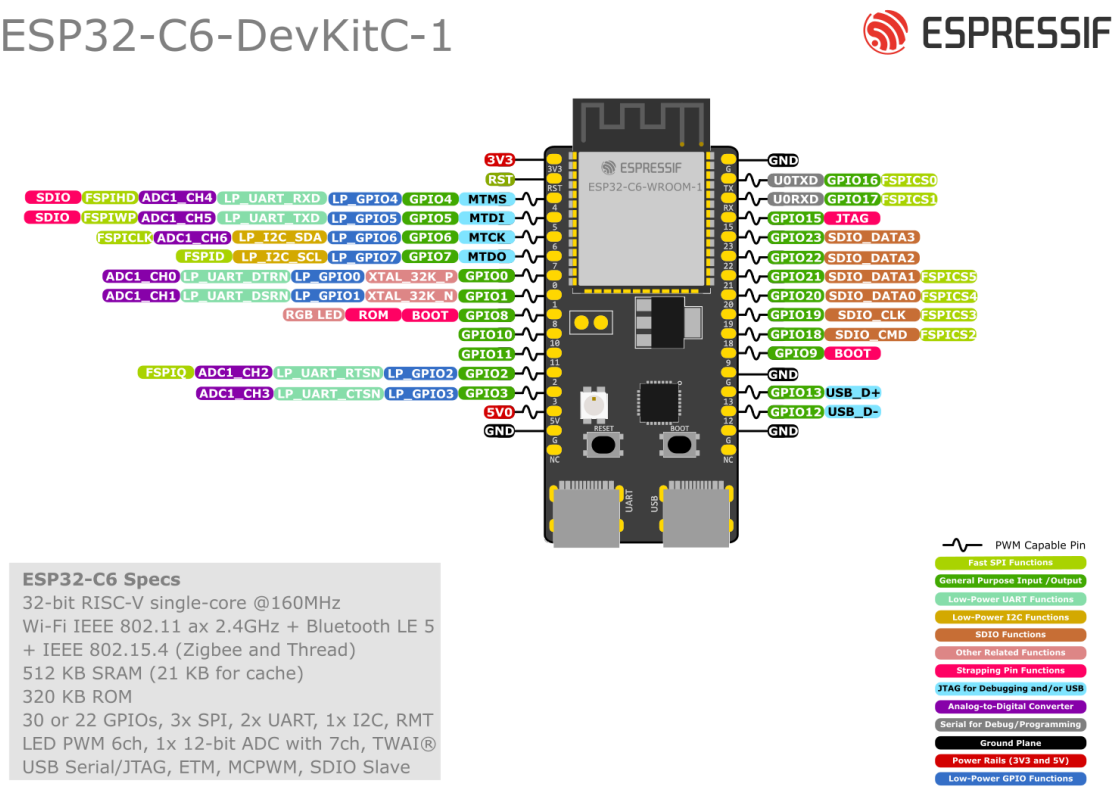
Έτσι, στην παρούσα εργασία, η εκτίμηση για την επιβάρυνση του αέρα από PM2.5/PM10 γίνεται με βάση τον **μέσο όρο 24ωρης μέτρησης**. Σε περίπτωση που ο μέσος όρος ξεπεράσει τα επιτρεπτά όρια, εμφανίζεται το αντίστοιχο μήνυμα. Την ίδια στιγμή ο χρήστης διατηρεί την ικανότητα να παρακολουθεί και τις στιγμιαίες τιμές.

4.7 ESP32

Ο ESP32-C6-DevKitC-1 είναι αναπτυξιακή πλακέτα της Espressif Systems, βασισμένη στο ολοκληρωμένο ESP32-C6 και προορίζεται για εφαρμογές Διαδικτύου των πραγμάτων (IOT). Ο μικροελεγκτής βασίζεται σε αρχιτεκτονική 32-bit RISC-V, προσφέροντας ικανοποιητική υπολογιστική ισχύ σε συνδυασμό με χαμηλή κατανάλωση ενέργειας, χαρακτηριστικό ιδιαίτερα σημαντικό για συστήματα συνεχούς παρακολούθησης περιβαλλοντικών παραμέτρων.

Η πλατφόρμα ενσωματώνει ασύρματη επικοινωνία Wi-Fi 6 (IEEE 802.11ax), επιτρέποντας αξιόπιστη σύνδεση στο διαδίκτυο και αποδοτική μετάδοση δεδομένων σε περιβάλλοντα με αυξημένη διαδικτυακή κίνηση. Επιπλέον υποστηρίζεται Bluetooth Low Energy (BLE) και IEEE 802.15.4, γεγονός που καθιστά τη συσκευή κατάλληλη και για μελλοντική επέκταση σε δίκτυα αισθητήρων.

Ο ESP32-C6-DevKitC-1 διαθέτει πληθώρα ακροδεκτών γενικής χρήσης (GPIO), μέσω των οποίων υποστηρίζονται πρωτόκολλα επικοινωνίας όπως I²C, SPI και UART, καθώς και αναλογικές είσοδοι μέσω μετατροπέων ADC. Αυτό επιτρέπει την άμεση διασύνδεση αισθητήρων ποιότητας αέρα. [12]



Εικόνα 4.15: Ο μικροελεγκτής ESP32 του παρόντος συστήματος

4.7.1 Ο ρόλος του μικροελεγκτή

Στο πλαίσιο της αυτής της εργασίας, ο ESP32-C6 αξιοποιείται ως κεντρική μονάδα ελέγχου και επικοινωνίας, αναλαμβάνοντας την επεξεργασία των μετρήσεων και την αποστολή τους μέσω διαδικτύου στην πλατφόρμα ThingSpeak για αποθήκευση, οπτικοποίηση και περαιτέρω ανάλυση. Η υποστήριξη καταστάσεων χαμηλής κατανάλωσης ενέργειας καθιστά το σύστημα κατάλληλο για συνεχή ή μακροχρόνια λειτουργία.

Πιο συγκεκριμένα ο μικροελεγκτής:

- Λαμβάνει δεδομένα από τους αισθητήρες αερίων, σωματιδίων και περιβαλλοντικών παραμέτρων (θερμοκρασία, υγρασία) μέσω αναλογικών εισόδων (ADC) ή ψηφιακών διαύλων (I^2C, UART).

- Εκτελεί την απαραίτητη επεξεργασία των μετρήσεων, όπως υπολογισμούς λόγων (π.χ. Rs/R0), φιλτράρισμα τιμών και έλεγχο ορίων.

- Διαχειρίζεται τη σύνδεση στο ασύρματο δίκτυο Wi-Fi και την επικοινωνία με απομακρυσμένους εξυπηρετητές.

- Αποστέλλει τα δεδομένα στην πλατφόρμα ThingSpeak, όπου αποθηκεύονται και παρουσιάζονται σε πραγματικό χρόνο.

- Λειτουργεί ως στοιχείο ελέγχου και συντονισμού όλων των υποσυστημάτων, διασφαλίζοντας την ομαλή και αξιόπιστη λειτουργία του συνολικού συστήματος.

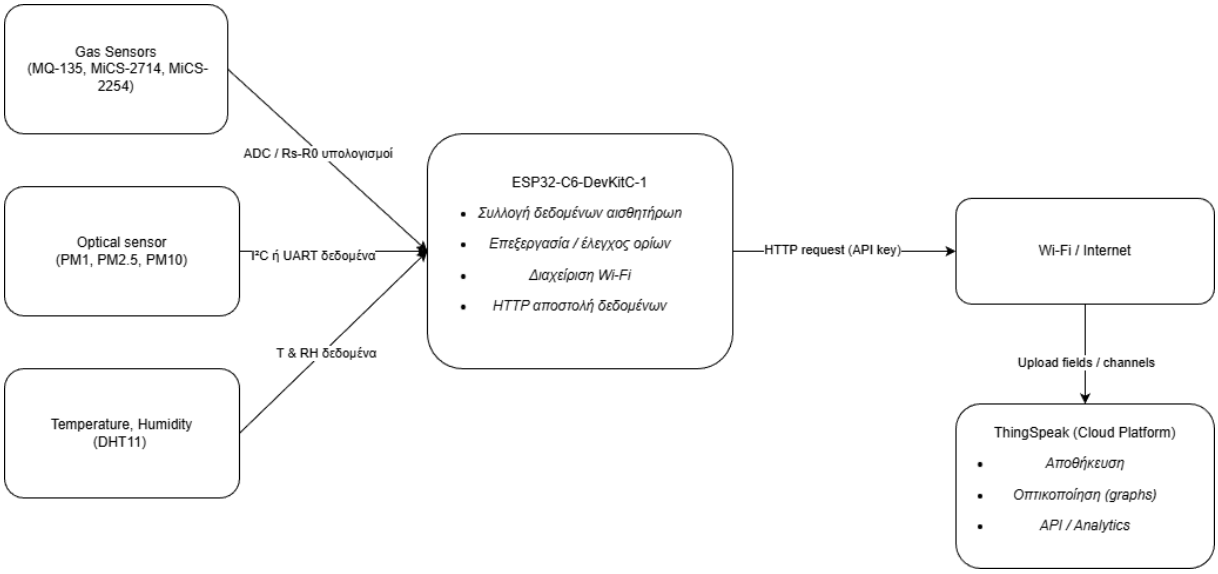
4.8 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκαν αναλυτικά οι αισθητήρες που συγκροτούν το σύστημα παρακολούθησης ποιότητας αέρα, καθώς και ο ρόλος του μικροελεγκτή **ESP32** ως κεντρικής μονάδας επεξεργασίας και επικοινωνίας. Περιγράφηκαν τα βασικά χαρακτηριστικά, η αρχή λειτουργίας και οι περιορισμοί κάθε αισθητήρα, με έμφαση στην ερμηνεία των μετρήσεων και στη σημασία της βαθμονόμησης μέσω κατάλληλων δεικτών, όπως ο λόγος Rs/R0. Ιδιαίτερη βαρύτητα δόθηκε στη συμπληρωματική χρήση διαφορετικών τεχνολογιών αισθητήρων (MOS, οπτικών και θερμοκρασίας/υγρασίας), ώστε να επιτυγχάνεται πιο αξιόπιστη και ολοκληρωμένη εκτίμηση της ποιότητας του αέρα, ειδικά σε βιομηχανικούς χώρους. Η ολοκλήρωση του κεφαλαίου αυτού θέτει το απαραίτητο θεωρητικό και τεχνικό υπόβαθρο για την κατανόηση της υλοποίησης, της συλλογής δεδομένων και της αξιολόγησης των μετρήσεων που ακολουθούν στα επόμενα κεφάλαια της εργασίας

Κεφάλαιο 5ο: Σχεδίαση και Αρχιτεκτονική του Συστήματος – Υλοποίηση

5.1 Εισαγωγή

Η συνολική αρχιτεκτονική του προτεινόμενου συστήματος παρακολούθησης ποιότητας αέρα παρουσιάζεται στο παρακάτω σχήμα (Εικόνα 5.1). Οι αισθητήρες αερίων και περιβαλλοντικών παραμέτρων (θερμοκρασία-υγρασία) τροφοδοτούν με μετρήσεις την κεντρική μονάδα ESP32-C6-DevKitC-1, η οποία αναλαμβάνει τη συλλογή και την επεξεργασία των δεδομένων (π.χ. υπολογισμούς/ελέγχους ορίων) και τη δικτυακή επικοινωνία. Στη συνέχεια, μέσω Wi-Fi, οι μετρήσεις αποστέλλονται με HHTP αιτήματα στην πλατφόρμα ThingSpeak, όπου πραγματοποιείται αποθήκευση και οπτικοποίηση των δεδομένων σε πραγματικό χρόνο.



Εικόνα 5.1: Block diagram του συστήματος παρακολούθησης ποιότητας αέρα με χρήση ESP32 και αποστολή δεδομένων στην πλατφόρμα ThingSpeak.

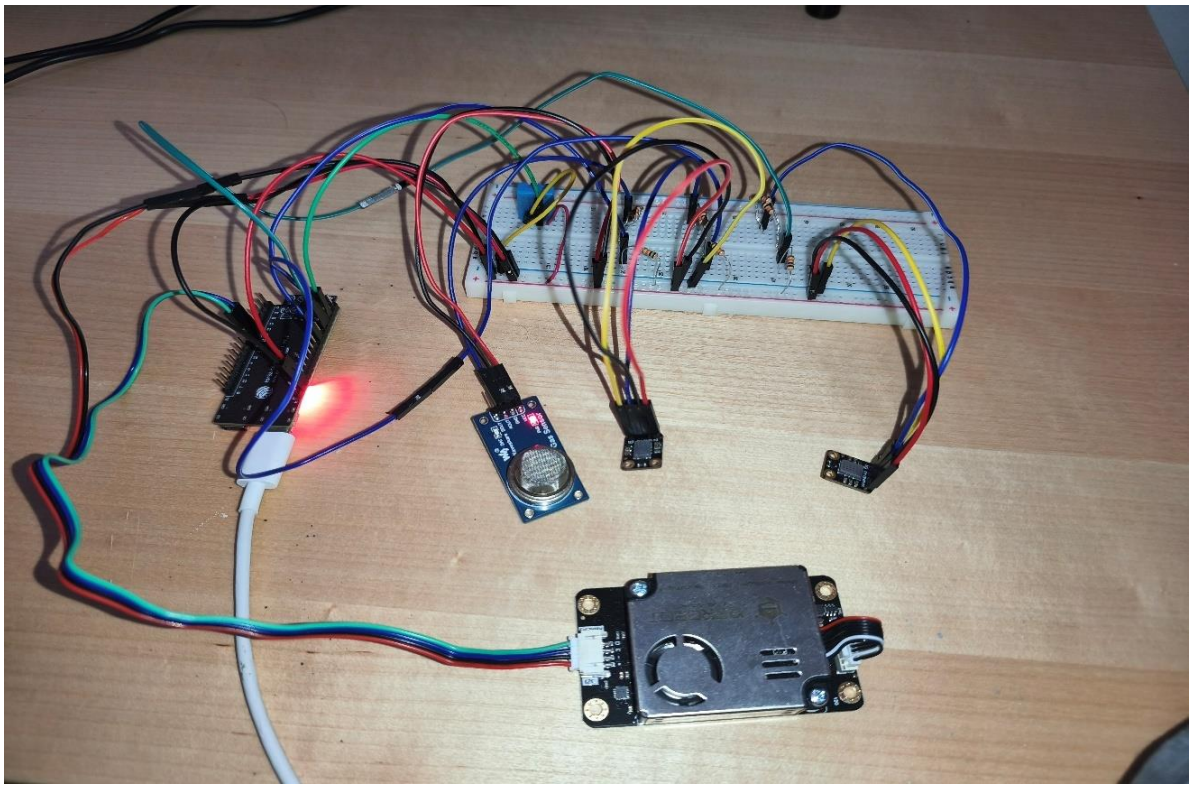
5.2 Γενική Δομή Υλικού (Hardware) Συστήματος

Στο σημείο αυτό περιγράφεται αναλυτικά η υλοποίηση του hardware του συστήματος παρακολούθησης ποιότητας αέρα. Παρουσιάζονται τα επιμέρους εξαρτήματα που χρησιμοποιήθηκαν, η μεταξύ τους συνδεσμολογία, καθώς και οι επιλογές που έγιναν σχετικά με την τροφοδοσία και τη διαχείριση σημάτων. Η ανάλυση αυτή βασίζεται στη θεωρητική περιγραφή των αισθητήρων και του μικροελεγκτή που παρουσιάστηκε στο προηγούμενο κεφάλαιο και εστιάζει στην πρακτική εφαρμογή τους σε ένα λειτουργικό πρωτότυπο.

Το σύστημα αποτελείται από τα ακόλουθα βασικά στοιχεία:

- Μικροελεγκτής (ESP32)
- Αισθητήρες αερίων (MiCS-2714, MiCS-5524)
- Αισθητήρας αιωρούμενων σωματιδίων
- Αισθητήρας θερμοκρασίας και σχετικής υγρασίας

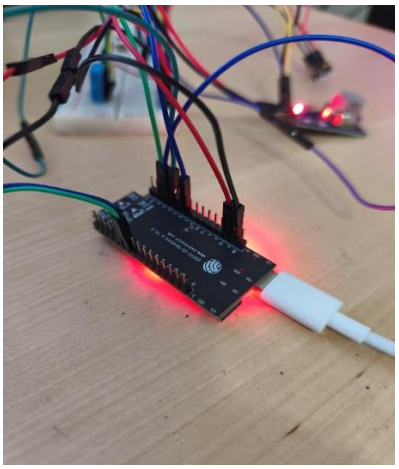
Η υλοποίηση πραγματοποιήθηκε σε πειραματικό κύκλωμα (breadboard), γεγονός που επιτρέπει την ευελιξία στις δοκιμές και τη μελλοντική επέκταση του συστήματος.



Εικόνα 5.2: Το ολοκληρωμένο σύστημα παρακολούθησης ποιότητας αέρα

Μικροελεγκτής (ESP32):

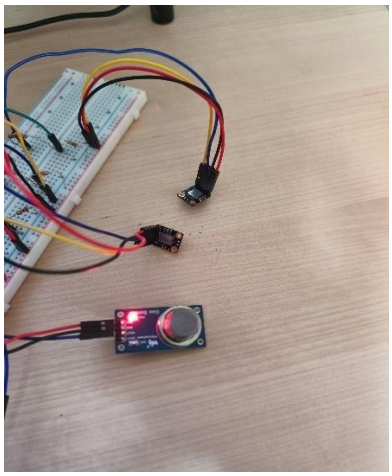
Η επιλογή του συγκεκριμένου μικροελεγκτή βασίστηκε στη χαμηλή κατανάλωση ενέργειας, στην ενσωματωμένη δυνατότητα ασύρματης επικοινωνίας και στη διαθεσιμότητα πολλαπλών διεπαφών (ADC, I2C, GPIO).



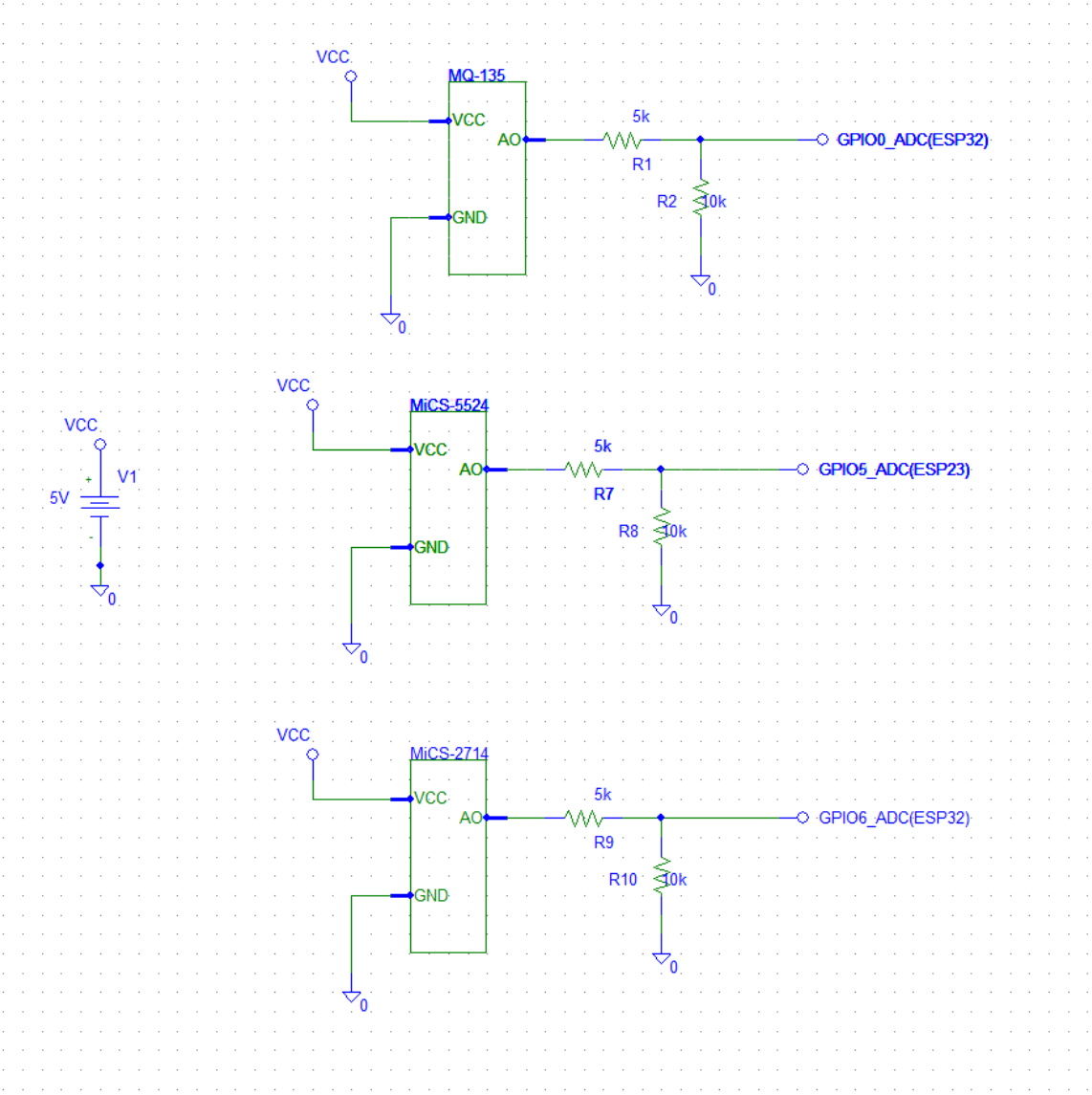
Εικόνα 5.3: Μικροελεγκτής ESP32 τροφοδοτούμενος μέσω USB. Διακρίνονται οι συνδέσεις τροφοδοσίας, οι αναλογικές είσοδοι από τους αισθητήρες αερίων και οι ψηφιακές γραμμές επικοινωνίας I2C.

Αισθητήρες Αερίων:

Οι αισθητήρες αερίων συνδέονται στις αναλογικές εισόδους του μικροελεγκτή και λειτουργούν με βάση τη μεταβολή της ηλεκτρικής τους αντίστασης ως απόκριση στην παρουσία ρύπων. Το σήμα εξόδου λαμβάνεται μέσω κατάλληλου κυκλώματος φορτίου και μετατρέπεται σε τάση, η οποία οδηγείται στον μετατροπέα αναλογικού σε ψηφιακό (ADC). Η τροφοδοσία κάθε αισθητήρα παρέχεται από τη γραμμή των 5V.



Εικόνα 5.4: Αισθητήρες αερίων (MiCS-2714,MiCS-5524 και MQ-135). Η έξοδος κάθε αισθητήρα υλοποιείται μέσω διαιρέτη τάσης.



Εικόνα 5.5:Διάγραμμα συνδεσμολογίας των αισθητήρων αερίων. Κάθε αισθητήρας μοντελοποιείται ως εξωτερική μονάδα με VCC, GND και αναλογική έξοδο (AO). Ένας διαιρέτης τάσης (5 kΩ / 10 kΩ) χρησιμοποιείται για την κλιμάκωση της εξόδου του αισθητήρα πριν τροφοδοτηθεί στην είσοδο ADC του μικροελεγκτή ESP32.

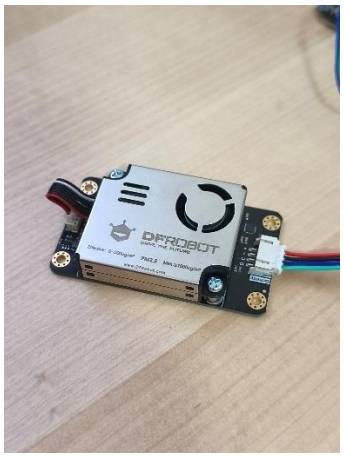
Προκειμένου να επιτευχθεί η επιθυμητή τάση λειτουργίας και να διασφαλιστεί η σωστή ανάγνωση του σήματος εξόδου των αισθητήρων αερίων **MQ-135**, **MiCS-2714** και **MiCS-5524**, υλοποιήθηκε κύκλωμα διαιρέτη τάσης σε κάθε αισθητήρα.

Συγκεκριμένα, σε σειρά με κάθε αισθητήρα τοποθετήθηκε αντίσταση R=5k και παράλληλα με αυτή μια R=10k σχηματίζοντας έναν διαιρέτη τάσης μεταξύ του αισθητήρα και του ESP32. Η τάση εξόδου λαμβάνεται από τον κόμβο μεταξύ R(5k) και RL(10k) και οδηγείται στην αναλογική είσοδο του μικροελεγκτή.

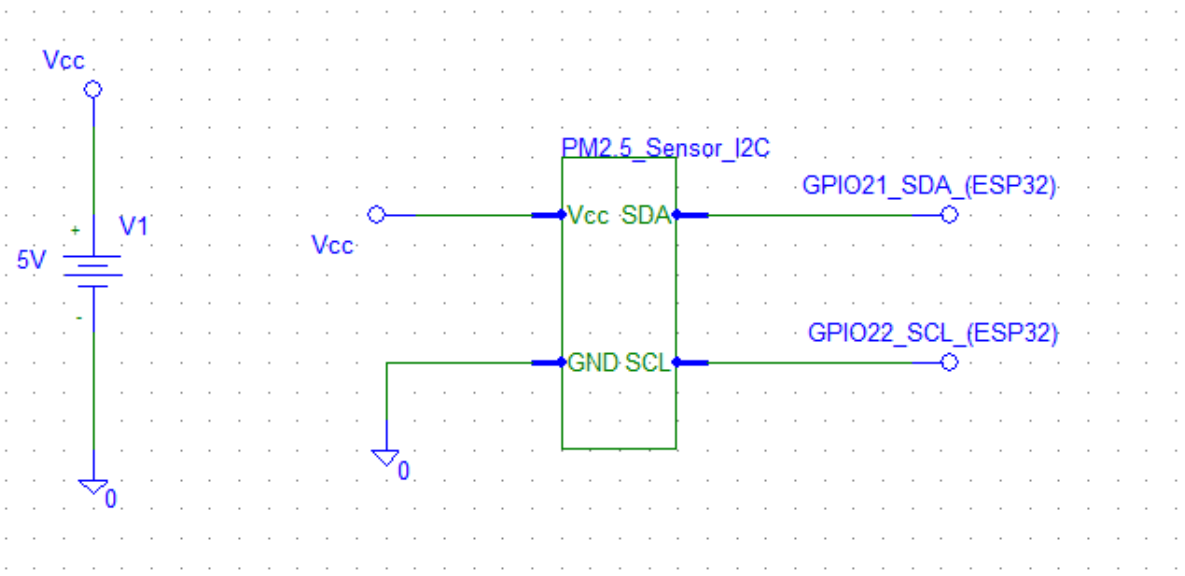
Η συγκεκριμένη διάταξη επιτρέπει τη μετατροπή των μεταβολών της αντίστασης του αισθητήρα, οι οποίες προκαλούνται από τη συγκέντρωση των ανιχνευόμενων αερίων, σε μεταβολές τάσης εντός των επιτρεπτών ορίων του μετατροπέα αναλογικού σε ψηφιακό (ADC) του μικροελεγκτή.

Αισθητήρας αιωρούμενων σωματιδίων:

Ο αισθητήρας αιωρούμενων σωματιδίων PM2.5 επικοινωνεί με τον μικροελεγκτή μέσω ψηφιακής διεπαφής (I2C). Η χρήση ψηφιακής επικοινωνίας μειώνει τον θόρυβο στις μετρήσεις και απλοποιεί τη συνδεσμολογία, καθώς απαιτούνται μόνο δύο γραμμές δεδομένων (SDA και SCL). Η τροφοδοσία παρέχεται από τη γραμμή των 5V.



Εικόνα 5.6: Οπτικός αισθητήρας αιωρούμενων σωματιδίων PM2.5 της DFRobot. Η επικοινωνία με τον μικροελεγκτή πραγματοποιείται μέσω διαύλου I2C, ενώ η τροφοδοσία παρέχεται από γραμμή 5V.



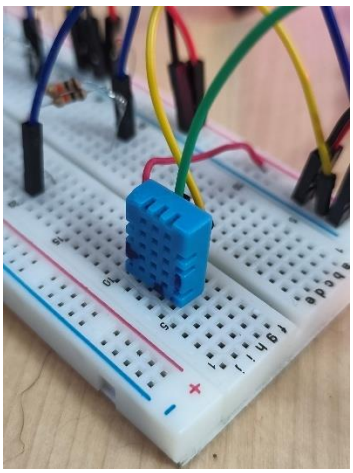
Εικόνα 5.7: Σχηματικό διάγραμμα συνδεσμολογίας PM2.5 DFRobot

Αισθητήρας θερμοκρασίας και σχετικής υγρασίας:

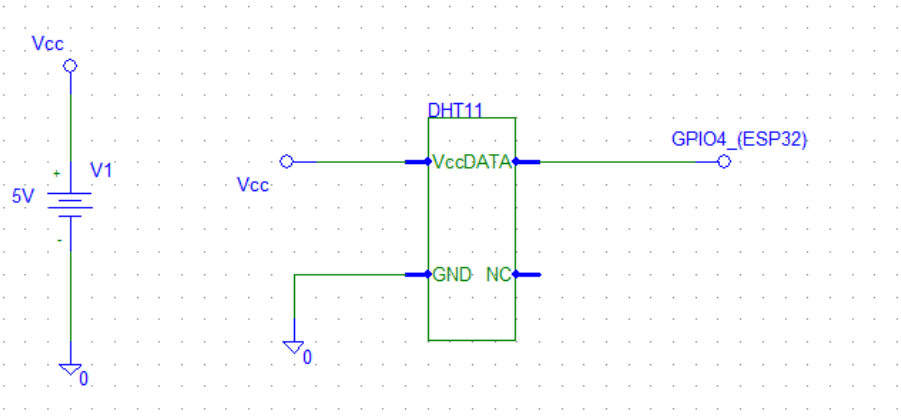
Ο αισθητήρας θερμοκρασίας και σχετικής υγρασίας χρησιμοποιείται για:

- την καταγραφή περιβαλλοντικών συνθηκών,
- την αξιολόγηση της αξιοπιστίας των μετρήσεων των αισθητήρων αερίων,
- την υποστήριξη λογικών ειδοποιήσεων σε περιπτώσεις ακραίων συνθηκών.

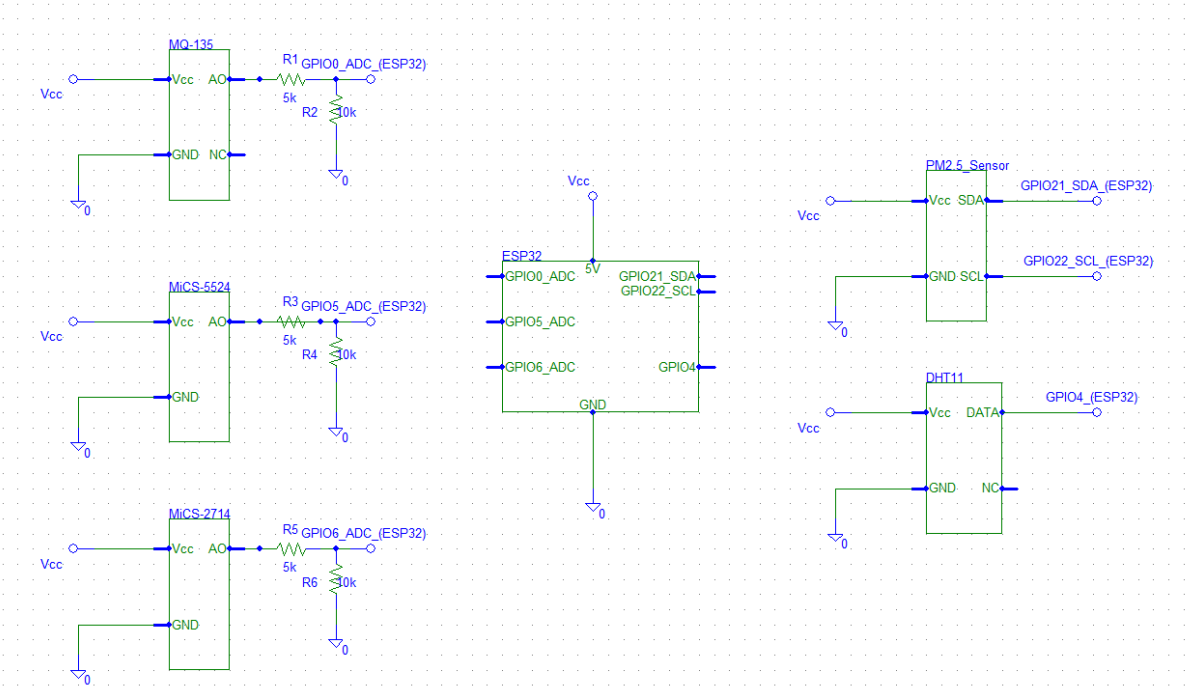
Η έξοδός του είναι ψηφιακή και συνδέεται σε κατάλληλη ψηφιακή είσοδο του μικροελεγκτή.



Εικόνα 5.8: Αισθητήρας θερμοκρασίας και σχετικής υγρασίας (DHT).



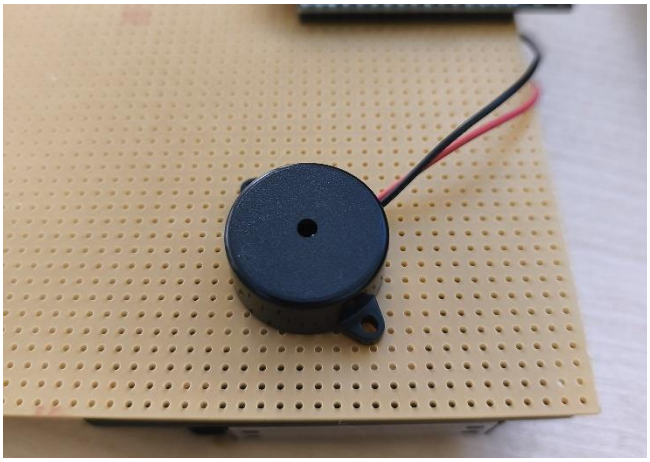
Εικόνα 5.9: Σχηματικό διάγραμμα συνδεσμολογίας DHT11



Εικόνα 5.10: Σχηματικό διάγραμμα της συνδεσμολογίας του συστήματος.

Buzzer προειδοποίησης

Τέλος, στο σύστημα ενσωματώθηκε μηχανισμός τοπικής ηχητικής προειδοποίησης μέσω buzzer, με σκοπό την άμεση ειδοποίηση του χρήστη σε περίπτωση παραβίασης προκαθορισμένων ορίων. Σε συνθήκες παραβίασης, το buzzer παράγει ηχητικό σήμα, ενημερώνοντας άμεσα τον χρήστη για την ύπαρξη ανθυγιεινού ή δυνητικά επικίνδυνου περιβάλλοντος. Η ηχητική ειδοποίηση παραμένει ενεργή όσο διαρκεί η παραβίαση και απενεργοποιείται αυτόματα όταν οι μετρούμενες τιμές επανέλθουν εντός αποδεκτών ορίων. Με αυτόν τον τρόπο, το σύστημα δεν περιορίζεται μόνο σε απομακρυσμένη καταγραφή και ειδοποίηση μέσω διαδικτύου, αλλά παρέχει και άμεση τοπική ανατροφοδότηση, η οποία είναι ιδιαίτερα χρήσιμη σε περιπτώσεις όπου το διαδίκτυο καταρρέει.



Εικόνα 5.11: Buzzer προειδοποίησης

5.3 Υλοποίηση λογισμικού

Το λογισμικό είναι υπεύθυνο για την αρχικοποίηση του μικροελεγκτή και των αισθητήρων, τη συλλογή και επεξεργασία των μετρήσεων, καθώς και τη μετάδοση των δεδομένων σε απομακρυσμένη πλατφόρμα αποθήκευσης και οπτικοποίησης (ThingSpeak). Η ανάπτυξη του κώδικα πραγματοποιήθηκε με γνώμονα τη modular δομή, τη σταθερότητα λειτουργίας και τη δυνατότητα μελλοντικής επέκτασης του συστήματος.

Ο μικροελεγκτής ESP32 προγραμματίστηκε σε γλώσσα **Python (MicroPython)**. Η επιλογή αυτή βασίστηκε:

- στην απλότητα και αναγνωσιμότητα του κώδικα
- στη διαθεσιμότητα βιβλιοθηκών για αισθητήρες και επικοινωνία Wi-Fi
- στη δυνατότητα γρήγορης ανάπτυξης και δοκιμών

Το πρόγραμμα εκτελείται απευθείας στον μικροελεγκτή και λειτουργεί σε συνεχή βρόχο, επιτρέποντας περιοδικές μετρήσεις και αποστολή δεδομένων.

5.3.1 Δομή και Αρχιτεκτονική Λογισμικού

Το λογισμικό οργανώνεται σε επιμέρους λειτουργικές ενότητες:

1) **Αρχικοποίηση υποσυστημάτων**
   Κατά την εκκίνηση, ο ESP32 αρχικοποιεί τις διεπαφές εισόδου/εξόδου (GPIO), τις αναλογικές εισόδους (ADC) για τους αισθητήρες αερίων, τον δίαυλο I2C για τον αισθητήρα PM2.5, καθώς και τον αισθητήρα DHT11. Παράλληλα, αρχικοποιούνται τα LEDs (status LEDs και NeoPixel) για οπτική ένδειξη κατάστασης λειτουργίας.

   Ιδιαίτερη σημασία δίνεται στον χρόνο σταθεροποίησης των αισθητήρων αερίων πριν την έναρξη των μετρήσεων.

```
PM sensor warm-up (recommended)...
MQ-135 calibration:
  Vclean = 0.354 V
  RL     = 10000 ohm
  Vc     = 5.0 V
  K_air  = 1.6
  Estimated R0_mq135 = 82026.8336 ohm
------------------------------------------------------------
MiCS-2714 warm-up (optional)... 60 sec
MiCS-2714 calibration (baseline R0_2714)...
  RL = 10000 ohm
  Vc = 5.0 V
  Estimated R0_2714 = 46557.58 ohm
------------------------------------------------------------
MiCS-5524 warm-up (optional)... 60 sec
MiCS-5524 calibration (baseline R0_5524)...
  RL = 10000 ohm
  Vc = 5.0 V
  Estimated R0_5524 = 72730.332 ohm
```

Εικόνα 5.12: Προθέρμανση και βαθμονόμηση των αισθητήρων πριν από την έναρξη των μετρήσεων. Εικόνα από το πρόγραμμα Thonny.

2) Σύνδεση σε ασύρματο δίκτυο Wi-Fi

Η σύνδεση στο Wi-Fi πραγματοποιείται δυναμικά, με σάρωση των διαθέσιμων δικτύων και προσπάθεια σύνδεσης σε προκαθορισμένη λίστα SSID. **Κατά τη διαδικασία παρέχεται οπτική ένδειξη (blinking)**. Σε περίπτωση αποσύνδεσης, το σύστημα επιχειρεί επανασύνδεση πριν συνεχίσει τη διαδικασία μετρήσεων και αποστολής.

```
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.disconnect()


    print("Starting Wi-Fi connection...")
    set_rgb_color(0, 0, 255)
    time.sleep(1)
    available = [net[0].decode("utf-8") for net in wlan.scan()]
    print("Found networks:", available)


    connected = False
    for net in WIFI_NETWORKS:
        ssid = net["ssid"]
        password = net["password"]
        if ssid in available:
            print(f"Trying to connect to {ssid} ...")
            wlan.connect(ssid, password)
            start = time.time()


            while not wlan.isconnected() and time.time() - start < 15:
                set_rgb_color(0, 0, 255)
                time.sleep_ms(300)
                set_rgb_color(0, 0, 0)
                time.sleep_ms(300)


            if wlan.isconnected():
                print(f"✅ Connected to {ssid}: {wlan.ifconfig()}")
                set_rgb_color(0, 255, 255)
                time.sleep(1)
                connected = True
                break
            else:
                print(f" Failed to connect to {ssid}")
        else:
            print(f" {ssid} not found in range")


    set_rgb_color(0, 0, 0)
    if not connected:
        print("No Wi-Fi connected")
    return wlan.isconnected()
```

3) Ανάγνωση αισθητήρων

Στο υποσύστημα της ανάγνωσης αισθητήρων, συλλέγονται δεδομένα από όλους τους αισθητήρες αερίων (MQ-135, MiCS-2714 και MiCS-5524), από τον αισθητήρα θερμοκρασίας και σχετικής υγρασίας (DHT11) καθώς και από τον αισθητήρα αιωρούμενων σωματιδίων (DFRobot)PM 2.5 Optical Sensor.

Για τους αισθητήρες των αερίων, οι οποίοι παρέχουν αναλογική έξοδο τάσης, έχει τοποθετηθεί εξωτερικός διαιρέτης τάσης πριν την είσοδο του ADC του ESP32. Για τον λόγο αυτό, η ανάγνωση πραγματοποιείται μέσω ειδικής συνάρτησης, η οποία ανακατασκευάζει την πραγματική τάση εξόδου του αισθητήρα:

```
def read_sensor_with_divider(adc):
    """
    Reads ADC voltage (after external divider R1/R2), then reconstructs real
sensor Vout.
    - v_div    : voltage actually seen by ADC (0..3.3V)
    - v_sensor : reconstructed real Vout of sensor divider (can be up to ~5V)
    """
    raw = adc.read()
    v_div = (raw / ((1 << ADC_BITS) - 1)) * ADC_REF
    v_sensor = v_div * ((R1 + R2) / R2)
    return raw, v_div, v_sensor
```

Η συνάρτηση επιστρέφει :
Την τιμή **raw,** την **τάση που «βλέπει» ο μετατροπέας ADC** και την **πραγματική τάση εξόδου** του αισθητήρα πριν τον διαιρέτη.

Στο κύριο πρόγραμμα, ανάγνωση των αισθητήρων αερίων γίνεται ως εξής:

```
raw_5524, v_div_5524, v_sensor_5524 = read_sensor_with_divider(adc_5524)
raw_2714, v_div_2714, v_sensor_2714 = read_sensor_with_divider(adc_2714)
raw_mq135, v_div_mq135, v_sensor_mq135 = read_sensor_with_divider(adc_mq135)
```

Η θερμοκρασία και η σχετική υγρασία ανακτώνται από τον αισθητήρα DHT11 με χρήση **try/except** για την αποφυγή σφαλμάτων αναγνώρισης:

```
try:
        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
except OSError:
        temp = None
        hum = None
```

Τέλος ο οπτικός αισθητήρας για τα αιωρούμενα σωματίδια διαβάζεται μέσω I2C και επιστρέφει τις συγκεντρώσεις των PM1.0, PM2.5 και PM10.

4) **Επεξεργασία και κανονικοποίηση δεδομένων**
Μετά την ανάγνωση των αισθητήρων, τα δεδομένα επεξεργάζονται ώστε να εξαχθεί ο λόγος **Rs/R0**, ο οποίος αποτελεί τον σχετικό δείκτη συγκέντρωσης ρύπων. Ο λόγος αυτός χρησιμοποιείται για ποιοτική κατηγοριοποίηση (“Πολύ καλή”, “Καλή”, κ.λπ.).

Η αντίσταση του αισθητήρα (Rs) υπολογίζεται με βάση το κύκλωμα διαιρέτη Rs–RL:

```
def mq135_compute_rs(vout, rl=MQ135_RL, vc=MQ135_VC):
    """
    Divider:
      Vout = Vc * RL / (Rs + RL)
      => Rs = RL * (Vc - Vout) / Vout
    """
    if vout is None or vout <= 0.0:
        return None
    if vout >= vc:
        vout = vc - 0.01
    return rl * (vc - vout) / vout
```

Αντίστοιχες συναρτήσεις χρησιμοποιούνται και για τους αισθητήρες MiCS-2714 και MiCS-5524.

Η τιμή αναφοράς **R0 (baseline)** υπολογίζεται αυτόματα κατά την εκκίνηση του συστήματος εφόσον το σύστημα παραμένει για κάποιο διάστημα σε περιβάλλον «καθαρού» αέρα.

Ενδεικτικά, για τον MQ-135 η διαδικασία υλοποιείται ως εξής:

```
def mq135_calibrate_r0(adc_obj, samples=MQ135_CAL_SAMPLES,
delay_s=MQ135_CAL_DELAY_S, k_air=K_AIR):
    """
    Calibrate R0 from measurements in "clean-ish" air:
      1) measure Vout_real (undo external divider)
      2) compute Rs for each sample
      3) R0 = average(Rs_clean) / K_AIR
    """
    s = 0.0
    n = 0
    for _ in range(samples):
        raw, v_div, vout_real = read_sensor_with_divider(adc_obj)
        rs = mq135_compute_rs(vout_real)
        if rs is not None:
            s += rs
            n += 1
        time.sleep(delay_s)
    if n == 0:
        return None
    rs_clean_avg = s / n
    return rs_clean_avg / k_air
```

Κατά τη λειτουργία του συστήματος, ο λόγος Rs/R0 υπολογίζεται σε πραγματικό χρόνο:

```
# MQ-135 Rs/R0 + quality
# ----------------------------
rs_mq135 = mq135_compute_rs(v_sensor_mq135)
rs_r0_mq135 = (rs_mq135 / R0_mq135) if (rs_mq135 is not None and R0_mq135 not in (None, 0)) else None
mq135_quality = mq135_quality_from_rs_r0(rs_r0_mq135)
```

Η κανονικοποιημένη αυτή τιμή χρησιμοποιείται για την ποιοτική αξιολόγηση της ποιότητας του αέρα μέσω προκαθορισμένων επιπέδων.

5) **Αποστολή δεδομένων στο cloud**
Η αποστολή στο cloud γίνεται μέσω HTTP GET αιτήματος προς την πλατφόρμα ThingSpeak, περνώντας τις μετρήσεις σε πεδία (fields). Πριν από την αποστολή ενεργοποιείται οπτική ένδειξη (LED upload) ώστε να είναι εμφανής η κατάσταση μετάδοσης (βλ κώδικα (410-429) ΠΑΡΑΡΤΗΜΑ Α).

```
def upload_data(temp, hum, raw_mq135, raw_5524, raw_2714, pm1_0, pm2_5, pm10):
    print("Uploading data to ThingSpeak...")
    for _ in range(3):
        led_upload.value(1)
        set_rgb_color(0, 255, 0)
        time.sleep_ms(300)
        led_upload.value(0)
        set_rgb_color(0, 0, 0)
        time.sleep_ms(300)
    try:
        url = (
            f"{THINGSPEAK_URL}?api_key={API_KEY}"

f"&field1={temp}&field2={hum}&field3={raw_mq135}&field4={raw_5524}&field5={raw
_2714}"
            f"&field6={pm1_0}&field7={pm2_5}&field8={pm10}"
        )
        response = urequests.get(url)
        print("ThingSpeak response:", response.text)
        response.close()
    except Exception as e:
        print("Upload failed:", e)
```

6) Main loop

Ο κεντρικός βρόχος (main loop) αποτελεί τον πυρήνα της αρχιτεκτονικής του λογισμικού και εκτελείται συνεχώς μετά την ολοκλήρωση της αρχικοποίησης και της βαθμονόμησης των αισθητήρων. Ο βρόχος αυτός αποτελεί τον κεντρικό μηχανισμό ελέγχου του λογισμικού και καθορίζει τη σειρά με την οποία πραγματοποιούνται οι επιμέρους λειτουργίες του συστήματος.

Οι ενέργειες που εκτελούνται στον κύριο βρόχο έχουν ως εξής:

Αρχικά ελέγχεται η κατάσταση της ασύρματης σύνδεσης Wi-Fi και επιχειρείται επανασύνδεση σε περίπτωση απώλειας επικοινωνίας. Στη συνέχεια πραγματοποιείται η ανάγνωση των αισθητήρων αερίων, του αισθητήρα αιωρούμενων σωματιδίων και του αισθητήρα θερμοκρασίας και σχετικής υγρασίας. Τα δεδομένα που συλλέγονται επεξεργάζονται και κανονικοποιούνται, με βασικό μέγεθος αναφοράς τον λόγο Rs/R0 για τους αισθητήρες αερίων, ενώ παράλληλα εφαρμόζεται ποιοτική κατηγοριοποίηση των μετρήσεων.

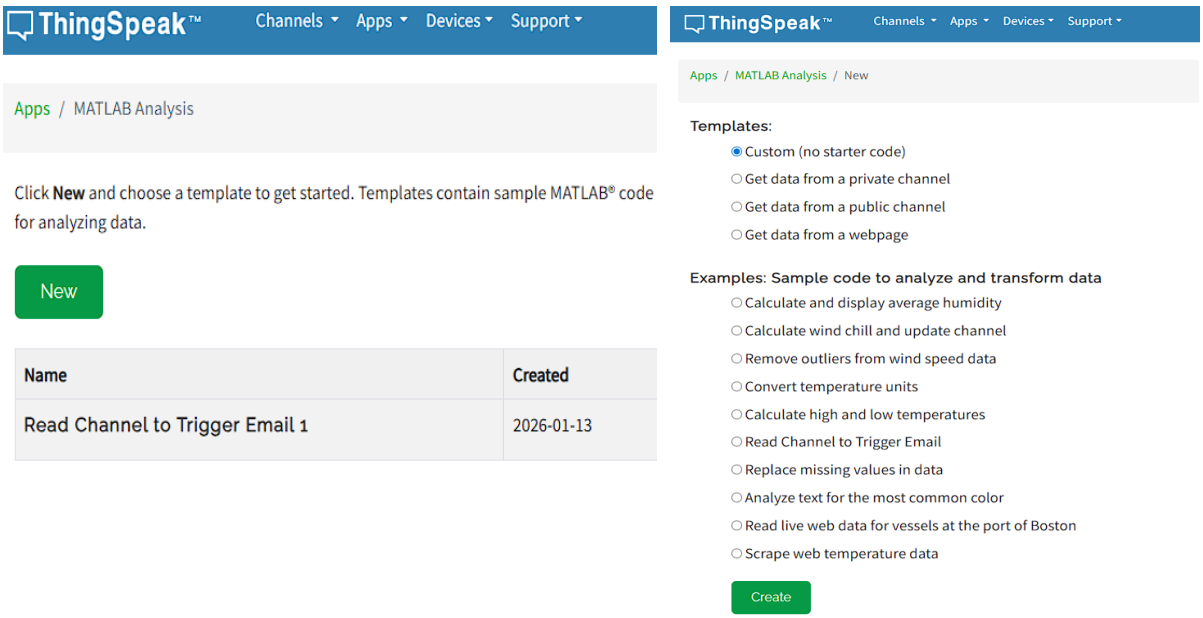
Στη συνέχεια, τα επεξεργασμένα δεδομένα αποστέλλονται σε απομακρυσμένη πλατφόρμα cloud για αποθήκευση και οπτικοποίηση. Μετά την ολοκλήρωση της αποστολής, το σύστημα εισέρχεται σε κατάσταση αναμονής για προκαθορισμένο χρονικό διάστημα, το οποίο καθορίζει τον ρυθμό δειγματοληψίας και μετάδοσης. Η διαδικασία επαναλαμβάνεται επ' αόριστον, εξασφαλίζοντας τη συνεχή και αυτοματοποιημένη λειτουργία του συστήματος.

```
while True:
    if not network.WLAN(network.STA_IF).isconnected():
        connect_wifi()
        time.sleep(2)

    # ----------------------------
    # Read gas sensors (ALL WITH DIVIDER)
    # ----------------------------
    raw_5524, v_div_5524, v_sensor_5524 = read_sensor_with_divider(adc_5524)
    raw_2714, v_div_2714, v_sensor_2714 = read_sensor_with_divider(adc_2714)
    raw_mq135, v_div_mq135, v_sensor_mq135 = read_sensor_with_divider(adc_mq135)

    # ----------------------------
    # MQ-135 Rs/R0 + quality
    # ----------------------------
    rs_mq135 = mq135_compute_rs(v_sensor_mq135)
    rs_r0_mq135 = (rs_mq135 / R0_mq135) if (rs_mq135 is not None and R0_mq135 not in (None, 0)) else None
    mq135_quality = mq135_quality_from_rs_r0(rs_r0_mq135)

    # ----------------------------
    # MiCS-2714 Rs/R0 + quality
    # ----------------------------
    rs_2714 = mics2714_compute_rs(v_sensor_2714)
    rs_r0_2714 = (rs_2714 / R0_2714) if (rs_2714 is not None and R0_2714 not in (None, 0)) else None
    mics2714_quality = mics2714_quality_from_rs_r0(rs_r0_2714)

    # ----------------------------
    # MiCS-5524 Rs/R0 + CO/VOC scenarios
    # ----------------------------
    rs_5524 = mics5524_compute_rs(v_sensor_5524)
    rs_r0_5524 = (rs_5524 / R0_5524) if (rs_5524 is not None and R0_5524 not in (None, 0)) else None

    lvl_co_5524 = mics5524_level_co(rs_r0_5524)
    lvl_voc_5524 = mics5524_level_voc_like(rs_r0_5524)
    lvl_final_5524 = mics5524_final_level(lvl_co_5524, lvl_voc_5524)
```

```
  try:
      dht_sensor.measure()
      temp = dht_sensor.temperature()
      hum = dht_sensor.humidity()
  except OSError:
      temp = None
      hum = None

  # ----------------------------
  # Read PM sensor
  # ----------------------------
  pm_vals, pm_raw, pm_mode = read_pm_values(debug=True)
  if pm_vals:
      pm1_0, pm2_5, pm10 = pm_vals
  else:
      pm1_0, pm2_5, pm10 = 0, 0, 0

  # ----------------------------
  # Print sensor values
  # ----------------------------
  print("MiCS-5524 -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs: {:.0f}Ω | Rs/R0: {:.3f}".format(
      raw_5524,
      v_div_5524,
      v_sensor_5524,
      rs_5524 if rs_5524 is not None else -1,
      rs_r0_5524 if rs_r0_5524 is not None else -1
  ))
  print("           CO scenario  level:", LEVEL_NUM_TO_TEXT_5524.get(lvl_co_5524, "UNK"))
  print("           VOC scenario level:", LEVEL_NUM_TO_TEXT_5524.get(lvl_voc_5524, "UNK"))
  print("           FINAL (worst-case) :", LEVEL_NUM_TO_TEXT_5524.get(lvl_final_5524, "UNK"))

  print("MiCS-2714 -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs: {:.0f}Ω | Rs/R0: {:.3f} | Level: {}".format(
      raw_2714,
      v_div_2714,
      v_sensor_2714,
      rs_2714 if rs_2714 is not None else -1,
      rs_r0_2714 if rs_r0_2714 is not None else -1,
      mics2714_quality
  ))
print("MQ-135    -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs: {:.0f}Ω | Rs/R0: {:.3f} | Quality: {}".format(
    raw_mq135,
    v_div_mq135,
    v_sensor_mq135,
    rs_mq135 if rs_mq135 is not None else -1,
    rs_r0_mq135 if rs_r0_mq135 is not None else -1,
    mq135_quality
))

if temp is not None and hum is not None:
    print("DHT11     -> Temp: {}°C | Humidity: {}%".format(temp, hum))
else:
    print("DHT11     -> Error reading sensor")

print("PM (mode: {}) -> PM1.0: {} | PM2.5: {} | PM10: {}".format(pm_mode, pm1_0, pm2_5, pm10))
print("-" * 60)

# ----------------------------
# Upload to ThingSpeak
# ----------------------------
upload_data(
    temp if temp is not None else 0,
    hum if hum is not None else 0,
    raw_mq135, raw_5524, raw_2714,
    pm1_0, pm2_5, pm10
)

# ----------------------------
# Waiting blink (red)
# ----------------------------
for _ in range(10):
    led_waiting.value(1)
    set_rgb_color(255, 0, 0)
    time.sleep(1)
    led_waiting.value(0)
    set_rgb_color(0, 0, 0)
    time.sleep(1)
```

5.4 Αποστολή ειδοποιήσεων μέσω e-mail

Για την έγκαιρη ειδοποίηση του χρήστη σε περιπτώσεις επιβάρυνσης της ποιότητας του αέρα ή λειτουργίας του συστήματος εκτός ορίων, υλοποιήθηκε **μηχανισμός αυτόματων ειδοποιήσεων μέσω email** αξιοποιώντας τις δυνατότητες της πλατφόρμας ThingSpeak. Η λογική υλοποίησης δεν εκτελείται στον μικροελεγκτή αλλά στο cloud, μέσω **MATLAB Analysis script (**ΠΑΡΑΡΤΗΜΑ Β), το οποίο ενεργοποιείται περιοδικά με **Time Control.** Με τον τρόπο αυτό επιτυγχάνεται ανεξαρτησία από τυχόν διακοπές του ESP32 και σταθερή λειτουργία ειδοποιήσεων, εφόσον οι μετρήσεις αποστέλλονται κανονικά στο κανάλι.



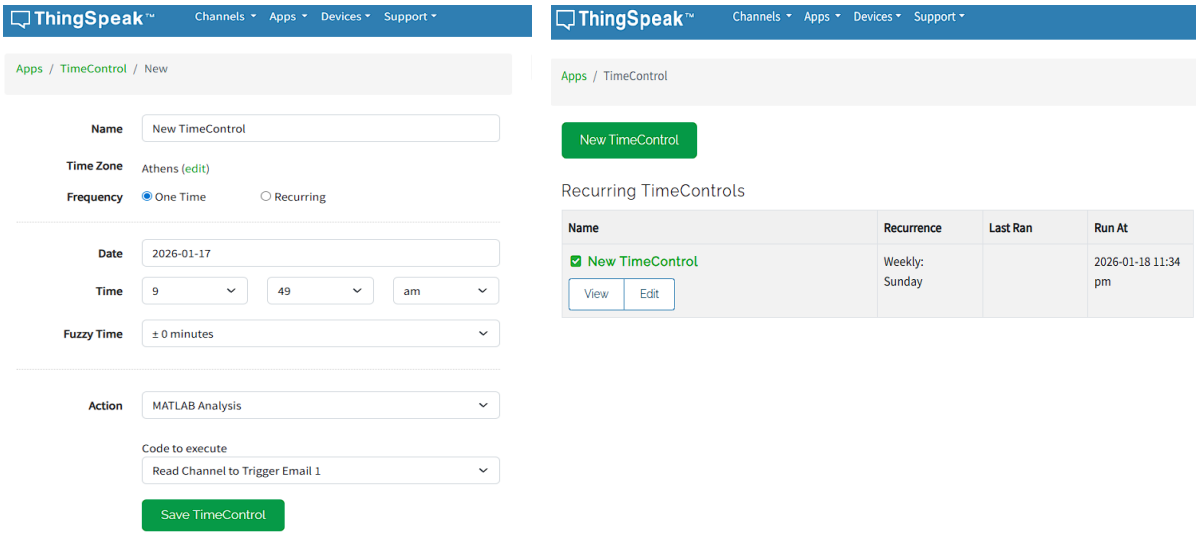
Εικόνα 5.13: Δημιουργία του script για αποστολή προειδοποιητικού email.

5.4.1 Περιοδική εκτέλεση script (Time Control)

Το script εκτελείται σε προκαθορισμένα χρονικά διαστήματα. Σε κάθε εκτέλεση:

1) Ανακτάται η πιο πρόσφατη εγγραφή από το κύριο κανάλι (τελευταίο δείγμα/last sample)
2) Ελέγχονται οι τιμές ως προς προκαθορισμένα όρια (thresholds)
3) Εάν εντοπιστεί παραβίαση, δημιουργείται μήνυμα ειδοποίησης και αποστέλλεται email μέσω του **ThingSpeak Alerts API**.

Η προσέγγιση αυτή εξασφαλίζει ότι η αξιολόγηση των ορίων γίνεται σε σταθερή συχνότητα, ενώ παράλληλα διευκολύνει τροποποιήσεις ορίων χωρίς αλλαγές στον κώδικα του ESP32.



Εικόνα 5.14: Εισαγωγή Time Cotrol

5.4.2 Χαρτογράφηση πεδίων και όρια ειδοποίησης

Το script χρησιμοποιεί αντιστοίχιση πεδίων από το κύριο κανάλι, ώστε οι μετρήσεις να ερμηνεύονται σωστά:

Field 1-2: Θερμοκρασία/Υγρασία
Field 3-5: τιμές αισθητήρων αερίων (RAW)
Field 6-8: PM1 / PM2.5 / PM10
Στη συνέχεια θέτονται τα όρια και εφαρμόζονται έλεγχοι σύμφωνα με τα παρακάτω:

Θερμοκρασία: Ειδοποίηση όταν η τιμή βρίσκεται εκτός του εύρους.

```
36 TEMP_LOW = 10;
37 TEMP_HIGH = 30;
```

Αιωρούμενα σωματίδια: ειδοποίηση όταν τα PM1, PM2.5 και PM10 υπερβαίνουν τα προκαθορισμένα όρια (σε αυτή την περίπτωση PM2.5=20 μg/m³ και PM10=45 μg/m³ ως 24ωρα επίπεδα προειδοποίησης σύμφωνα με τα διεθνή και ευρωπαϊκά όρια, ενώ το PM1 χρησιμοποιείται ως ενδεικτικός δείκτης).

```
32 PM25_WARN = 20; % WHO 2021 (24h guideline)
33 PM10_WARN = 45; % WHO 2021 (24h guideline)
34 PM1_WARN = 10; % indicative warning (no official WHO limit)
```

Αισθητήρες αερίων: Ο έλεγχός γίνεται εφόσον οριστεί baseline. Στον αρχικό κώδικα είναι `BASE_= 0`. Έτσι, αρχικά υπολογίζεται η τιμή **raw** σε συνθήκες καθαρού αέρα και συμπληρώνεται το αντίστοιχο πεδίο του κώδικα όπως φαίνεται παρακάτω:

```
BASE_MQ135 = 0; % set >0 to enable gas raw checks
BASE_MICS5524 = 0;
BASE_MICS2714 = 0;
```

Στη συνέχεια, εφόσον έχουν οριστεί τιμές baseline, ο έλεγχος γίνεται ως ποσοστιαία μεταβολή από τη baseline τιμή (Warning στο +30%, Alarm στο +60%).

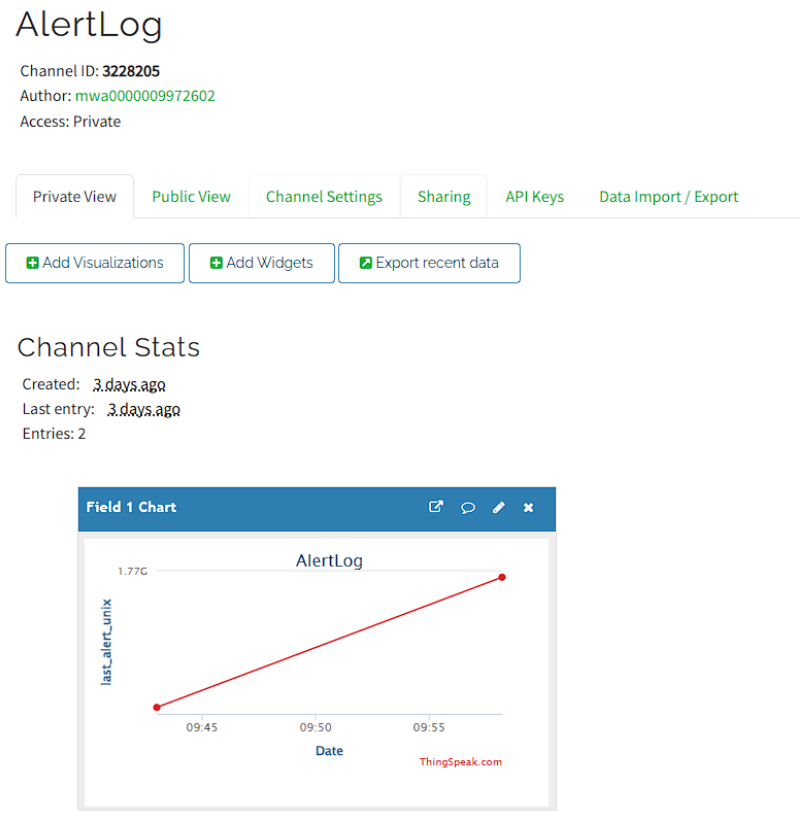
```
43 GAS_WARN_PCT = 0.30;
44 GAS_ALRM_PCT = 0.60;
```

Η έννοια της λίστας `issues{}` επιτρέπει να συμπεριληφθούν **πολλαπλά προβλήματα στο ίδιο email**, κάτι που δίνει σαφή εικόνα της κατάστασης χωρίς αποστολή πολλών μηνυμάτων.

Ολόκληρο το script παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Β.

5.4.3 Μηχανισμός αποφυγής "spam" (cooldown με AlertLog channel)

Για να αποφευχθεί συνεχής αποστολή emails σε περίπτωση παρατεταμένης ρύπανσης ή σταθερής υπέρβασης ορίων, εφαρμόστηκε μηχανισμός **cooldown**. Συγκεκριμένα, δημιουργήθηκε κανάλι "AlertLog", στο οποίο αποθηκεύεται η χρονική στιγμή της τελευταίας ειδοποίησης ως **Unix timestamp** (σε **Field1 = last_alert_unix**).



Εικόνα 5.15: Κανάλι AlertLog για την αποθήκευση των χρονικών στιγμών της τελευταίας ειδοποίησης

Η διαδικασία έχει ως εξής:

Το script υπολογίζει `now_unix` (τρέχων χρόνος σε UTC) → Διαβάζει από το AlertLog την τελευταία καταγραφή `last_unix` → Υπολογίζει το χρονικό διάστημα από την τελευταία ειδοποίηση σε λεπτά → Αν το διάστημα είναι μικρότερο από `COOLDOWN_MIN`, το script **δεν στέλνει email** και τερματίζει.

Έτσι αποτρέπεται η υπερβολική ειδοποίηση του χρήστη, ενώ διατηρείται η λειτουργία προειδοποίησης σε νέα γεγονότα μετά την παρέλευση του χρονικού παραθύρου.

5.4.4 Αποστολή email μέσω ThingSpeak Alerts API

Όταν εντοπιστούν παραβιάσεις και ο cooldown έλεγχος το επιτρέπει, δημιουργείται email με:

Subject: "Air Quality Alert (ThingSpeak)"
Body: τελευταία τιμή κάθε αισθητήρα + αναλυτική λίστα `Issues`

Η αποστολή πραγματοποιείται με **HTTP request** και αυθεντικοποίηση μέσω header: ThingSpeak-Alerts-API-Key.

Η μέθοδος αυτή επιτρέπει κεντρική διαχείριση ειδοποιήσεων από το περιβάλλον του ThingSpeak χωρίς να απαιτείται SMTP (Simple Mail Transfer Protocol) ρύθμιση στον ESP32.

Συνολικά: Το υποσύστημα ειδοποιήσεων ενισχύει τη λειτουργικότητα του συστήματος, καθώς μετατρέπει τις μετρήσεις σε άμεση ενημέρωση του χρήστη όταν εντοπίζονται επικίνδυνες ή μη επιθυμητές συνθήκες. Ο μηχανισμός Time Control σε συνδυασμό με το cooldown μέσω AlertLog εξασφαλίζει αξιοπιστία και αποφυγή υπερβολικών ειδοποιήσεων.

Κεφάλαιο 6ο: Συμπεράσματα και Προτάσεις Βελτίωσης του Συστήματος

Στο πλαίσιο αυτής της εργασίας σχεδιάστηκε και υλοποιήθηκε ένα ολοκληρωμένο σύστημα παρακολούθησης της ποιότητας του αέρα, βασισμένο σε μικροελεγκτή ESP32 και ένα συνδυασμό αισθητήρων αερίων, αιωρούμενων σωματιδίων και περιβαλλοντικών παραμέτρων (θερμοκρασία, σχετική υγρασία). Το σύστημα πέτυχε τη συνεχή καταγραφή δεδομένων, την απομακρυσμένη αποστολή τους σε πλατφόρμα IoT (ThingSpeak) καθώς και την ενεργοποίηση μηχανισμών ειδοποίησης σε περιπτώσεις υπέρβασης προκαθορισμένων ορίων.

6.1 Περιορισμοί του συστήματος

Ένα από τα βασικά συμπεράσματα που προκύπτουν αφορά στην **φύση και τους περιορισμούς των αισθητήρων** αερίων που χρησιμοποιήθηκαν. **Οι αισθητήρες MQ και MiCS** που χρησιμοποιήθηκαν στο σύστημα **δεν είναι σε θέση να ανιχνεύσουν έναν και μοναδικό ρύπο με απόλυτη επιλεκτικότητα, αλλά αντιδρούν σε ομάδες αερίων και μεταβάλλουν την έξοδό τους ως συνάρτηση της συνολικής παρουσίας αναγωγικών ή οξειδωτικών αερίων**. Για παράδειγμα, ο αισθητήρας MiCS-2714 που βρίσκεται σε ένα αντίστοιχο σύστημα εντός μιας εργοστασιακής μονάδας και εκτίθεται σε ρυπασμένο αέρα, θα καταγράψει αυξημένες μετρήσεις (υψηλότερες τιμές raw) χωρίς όμως να είναι σε θέση να «πει» ποιος είναι ο ακριβής ρύπος που ανίχνευσε. Η απόκριση του αισθητήρα, όπως αυτή αναλύεται στο κεφάλαιο 4, γίνεται με την παραδοχή ότι το διοξειδίο του αζώτου (NO_2) είναι ο συνηθέστερος ρύπος που απαντάται σε εργοστασιακά περιβάλλοντα ή έστω ότι είναι γνωστό πως το NO_2 είναι ο ρύπος που εκλύεται στον συγκεκριμένο χώρο. Το ίδιο ισχύει και για τους άλλους αισθητήρες αερίων (MiCS-5524 και MQ-135). Κατά συνέπεια, οι μετρήσεις που λαμβάνονται δεν μπορούν να αποδοθούν με απόλυτη βεβαιότητα σε έναν συγκεκριμένο ρύπο, αλλά παρέχουν κυρίως ποιοτική ή ημιποσοτική εκτίμηση της ρύπανσης του χώρου. Το γεγονός αυτό δεν αποτελεί σφάλμα σχεδίασης αλλά εγγενή περιορισμό της συγκεκριμένης κατηγορίας αισθητήρων χαμηλού κόστους.

6.2 Προτάσεις Βελτίωσης - Επέκτασης του συστήματος

Μια προφανής κατεύθυνση βελτιστοποίησης του συστήματος αφορά τη χρήση **ακριβότερων και πιο εξειδικευμένων αισθητήρων αερίων**, όπως **ηλεκτροχημικών** ή **φασματοσκοπικών αισθητήρων**, οι οποίοι προσφέρουν υψηλή επιλεκτικότητα και ακρίβεια ως προς τον υπό μέτρηση ρύπο. Οι αναφερόμενοι αισθητήρες δεν επιλέχθηκαν στην παρούσα εργασία κυρίως λόγω οικονομικών περιορισμών, καθώς το κόστος τους είναι σημαντικά υψηλότερο σε σχέση με τους αισθητήρες χαμηλού κόστους που χρησιμοποιήθηκαν.

Επιπλέον, θα μπορούσε να εξεταστεί **η εφαρμογή αλγορίθμων βαθμονόμησης και διόρθωσης βασισμένων σε δεδομένα από πιστοποιημένους σταθμούς μέτρησης**, ώστε να βελτιωθεί η αξιοπιστία των μετρήσεων. Ακόμη, **η χρήση τεχνικών μηχανικής μάθησης** για τη συσχέτιση των αποκρίσεων πολλαπλών αισθητήρων με συγκεκριμένα πρότυπα ρύπανσης αποτελεί επίσης μια ενδιαφέρουσα προοπτική.

Τέλος, μια σημαντική προοπτική εξέλιξης του συστήματος αφορά στη μετάβασή του από ένα σύστημα παρακολούθησης σε ένα **σύστημα ενεργής αντιμετώπισης της ρύπανσης του αέρα σε ελεγχόμενο χώρο**. Συγκεκριμένα, το σύστημα θα μπορούσε να επεκταθεί με την προσθήκη αυτοματοποιημένων μηχανισμών, οι οποίοι θα ενεργοποιούνται άμεσα σε περιπτώσεις παραβίασης προκαθορισμένων ορίων ρύπανσης. Ενδεικτικά, θα μπορούσε να εφαρμοστεί η αυτόματη ενεργοποίηση συστημάτων

εξαερισμού, ανεμιστήρων, φίλτρων αέρα ή άλλων μηχανισμών απομάκρυνσης ρύπων, **με στόχο την επιτόπου βελτίωση της ποιότητας του αέρα** στον ελεγχόμενο χώρο. Μια τέτοια προσέγγιση θα μετέτρεπε το σύστημα σε μια έξυπνη και αυτόνομη λύση περιβαλλοντικού ελέγχου, μειώνοντας τον χρόνο απόκρισης σε επικίνδυνες συνθήκες και περιορίζοντας την έκθεση των χρηστών/εργαζομένων σε επιβλαβείς ρύπους. Παράλληλα η λειτουργία αυτή θα μπορούσε να υλοποιηθεί με απλούς ενεργοποιητές (ρελέ, ηλεκτροβάνες) διατηρώντας το κόστος σε χαμηλά επίπεδα και ενισχύοντας την πρακτική αξία του συστήματος. ΄

6.3 Συνολική αποτίμηση

Συνοψίζοντας, το σύστημα αποδεικνύει ότι είναι δυνατή η υλοποίηση ενός λειτουργικού και επεκτάσιμου συστήματος παρακολούθησης της ποιότητας του αέρα με χαμηλό κόστος και ευέλικτο στην προσθήκη επιπλέον τεχνολογιών. Έτσι το σύστημα μπορεί να χρησιμοποιηθεί σε εργασιακά περιβάλλοντα (εργοστάσια, βιομηχανίες κ.α.) αλλά και για εκπαιδευτικούς, ερευνητικούς ή πιλοτικούς σκοπούς. Παρά τους περιορισμούς που εντοπίζονται, τα αποτελέσματα κρίνονται ικανοποιητικά και θέτουν τις βάσεις για μελλοντικές βελτιώσεις και περαιτέρω ερευνητική αξιοποίηση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Data Sheet

[1] Use introduction of HS-129 type gas sensitive components

[2] Microsoft Word - a1a-mics-2714_1_s1.doc

[3] ACROBOTIC | MiCS-5524 Datasheet

Internet Site

[1] Κ. Καλοβρέκτης και Ν. Κατέβας, "Χαρακτηριστικά αισθητήρων" [PDF], qr.tziola.gr, [Online]. Available: https://qr.tziola.gr/wp-content/uploads/2019/03/qr_1.1.pdf, Accessed: 09-12-2025.

[2] Zhengzhou Winsen Electronics Technology Co., Ltd, "Air Quality Gas Sensor（Model:MQ135）MANUAL", www.winsen-sensor.com, [Online]. Available: MQ135 (Ver1.4) - Manual.pdf, Accessed: 15-12-2025.

[3] T.K. Hareendran, "How To Use MQ-135 Gas Sensor", CORDEY Electronics, [Online]. Available: How To Use MQ-135 Gas Sensor - Codrey Electronics, Accessed: 14-12-2025.

[4] HANWEI ELECTRONICS CO., LTD, "TECHNICAL DATA MQ-135 GAS SENSOR," www.hwsensor.com, [Online]. Available: MQ-135, Accessed: 20-12-2025.

[5] "DHT11 Sensor Guide with Pinout, working, and Arduino Programming", www.electronicwings.com, [Online]. Available: DHT11 Sensor Guide with Pinout, working, and Arduino Programming ... , Accessed: 22-12-2025

[6] "MiCS-2714 NO2 Sensor" [PDF], www.cdiweb.com, [Online]. Available: Microsoft Word - a1a-mics-2714_1_s1.doc, Accessed: 28-12-2025

[7] "The MiCS-5524 is a compact MOS sensor." [PDF], SGX SENSORTECH, [Online]. Available: ACROBOTIC | MiCS-5524 Datasheet, Accessed: 28-12-2025

[8] Gas Detection Company, "Safety Protocols for Nitrogen Dioxide (NO_2) Gas Exposure," GasDetection.com. [Online]. Available: Safety Protocols for Nitrogen Dioxide (NO2) Gas Exposure. Accessed: 06-01-2026.

[9] DFRobot, "Fermion: MEMS Gas Sensor – MiCS-5524," DFRobot Wiki. [Online]. Available: Fermion: MEMS Gas Sensor - MiCS-5524 -DFRobot Product wiki. [Accessed: 28-Dec-2025].

[10] DFRobot, "Gravity: PM2.5 Air Quality Sensor (SEN0460)," *DFRobot Wiki*. [Online]. Available: Gravity: PM2.5 Air Quality Sensor PM2.5/PM10 Particle Sensor Wiki - DFRobot. [Accessed: 08-01-2026].

[11] California Air Resources Board, *Inhalable Particulate Matter and Health (PM2.5 and PM10)*. [Online]. Available: Inhalable Particulate Matter and Health (PM2.5 and PM10) | California Air Resources Board. [Accessed: 13-01-2026].

[12] Grobotronics, “ESP32-C6 DevKitC-1 N8 – 8MB SPI Flash,” *Grobotronics Online Store*. [Online]. Available: ESP32 Development Board - ESP32-C6-DevKitC-1-N8 - 8MB SPI Flash. [Accessed: 13-01-2026].

[13] Γαλανόπουλος, *Διπλωματική Εργασία*, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα, Ελλάδα, [Online]. Available: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/61821/%CE%93%CE%B1%CE%BB%CE%B1%CE%BD%CF%8C%CF%80%CE%BF%CF%85%CE%BB%CE%BF%CF%82_%CE%91%CE%BD%CE%B1%CF%83%CF%84%CE%AC%CF%83%CE%B9%CE%BF%CF%82_%CE%94%CE%B9%CF%80%CE%BB%CF%89%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE_%CE%95%CF%81%CE%B3%CE%B1%CF%83%CE%AF%CE%B1.pdf?sequence=1. Accessed: 14-12-2025.

[14] Winsen Electronics Technology Co., Ltd., “Semiconductor Sensors,” [Online]. Available: Αισθητήρες ημιαγωγών: Αρχές, τύποι και εφαρμογές Accessed: 14-12-2025.

[15] Π. Παπακάλος, *Διπλωματική Εργασία*, Πανεπιστήμιο, Ελλάδα, [Online]. Available: https://www.fire.gr/wp-content/uploads/2022/10/Petros_Papakalos_Univercity.pdf. Accessed: 24-01-2026

[16] Υπουργείο Περιβάλλοντος και Ενέργειας, «Ποιότητα της Ατμόσφαιρας», [Online]. Available: Ποιότητα της Ατμόσφαιρας - Υπουργείο Περιβάλλοντος και Ενέργειας. Accessed 24-01-2026.

[17] Συμβούλιο της Ευρωπαϊκής Ένωσης, «Ατμοσφαιρική ρύπανση στην ΕΕ», [Online]. Available: Ατμοσφαιρική ρύπανση στην ΕΕ: Στοιχεία και αριθμοί - Consilium. Accessed: 24-01-2026

Πηγές εικόνων

Εικόνα1.1: Ατμοσφαιρική ρύπανση: Οι πιο "μολυσμένες" ελληνικές πόλεις | News 24/7

Εικόνα 3.1: ME3-C2H6S2 Ηλεκτροχημικός αισθητήρας αερίου

Εικόνα 3.2: Carbon Monoxide sensor CO-B4 - Alphasense - B2B

Εικόνα 3.3: Waveshare MQ-135 Gas Sensor Module

Εικόνα 3.4: Fermion MEMS Gas Sensor - MiCS-2714

Εικόνα 3.5: MH-Z19C NDIR CO2 Module

Εικόνα 3.6: Senseair S8 Low Power CO2 Sensor – CO2 Meter

Εικόνα 3.7: Gravity Air Quality Sensor - PM2.5

Εικόνα 3.8: Particulate Matter Sensor PMS5003 with Cable

Εικόνα 3.9: 02_kalobrektis.pm6

Εικόνα 3.10: 02_kalobrektis.pm6

Εικόνα 3.11: 02_kalobrektis.pm6

Εικόνα 3.12: 02_kalobrektis.pm6

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας συστήματος (Micropython)

```
import machine
import neopixel
import network
import time
import urequests
import dht
from machine import ADC, Pin, I2C


# ----------------------------
# Wi-Fi credentials
# ----------------------------
WIFI_NETWORKS = [
    {"ssid": "INALAN_2.4G_9uAC9Z", "password": "X2XhsTKb"},
    {"ssid": "Redmi Note 14 Pro 5G", "password": "123454321"},
]


# ----------------------------
# ThingSpeak settings
# ----------------------------
API_KEY = "L9XPKJ6R2I4984CP"
THINGSPEAK_URL = "http://api.thingspeak.com/update"


# ----------------------------
# LED setup
# ----------------------------
led_waiting = Pin(5, Pin.OUT)
led_upload = Pin(7, Pin.OUT)


# Internal RGB LED (NeoPixel)
led_pin = Pin(8)
np = neopixel.NeoPixel(led_pin, 1)


def set_rgb_color(r, g, b):
    np[0] = (r, g, b)
    np.write()


# ----------------------------
# BUZZER setup
# ----------------------------
# Choose a free GPIO pin for the buzzer signal
BUZZER_PIN = 11
buzzer = Pin(BUZZER_PIN, Pin.OUT)
buzzer.value(0)


# Thresholds for buzzer activation
# adjustable limits
```

```
TH_TEMP_C = 30            # °C
TH_HUM_PCT = 80           # %
TH_PM25 = 35              # µg/m3 (or your sensor units)
TH_PM10 = 50              # µg/m3 (or your sensor units)


# Gas thresholds based on Rs/R0 ratio
# MQ-135: lower Rs/R0 => worse air. Trigger when below:
TH_MQ135_RS_R0_LOW = 0.7


# MiCS-2714: higher Rs/R0 => worse air (per your mapping). Trigger when above:
TH_MICS2714_RS_R0_HIGH = 2.0


# MiCS-5524: lower Rs/R0 => worse air. Trigger when below:
TH_MICS5524_RS_R0_LOW = 0.17


# ----------------------------
# Sensors setup (gas + DHT)
# ----------------------------


# ADC pins for gas sensors
ADC_PIN_5524 = 5
ADC_PIN_2714 = 6
ADC_PIN_MQ135 = 0


# Divider used BEFORE the ESP32 ADC
# Sensor AO -> R1 -> ADC node -> R2 -> GND
# ADC reads Vdiv; real sensor Vout = Vdiv * (R1+R2)/R2
R1 = 5000
R2 = 10000
ADC_REF = 3.3
ADC_BITS = 12


# ----------------------------
# MQ-135 parameters
# ----------------------------
MQ135_RL = 10000          # 10kΩ load resistor (sensor module divider)
MQ135_VC = 5.0            # Supply of the RS-RL divider (usually 5V). Change
to 3.3 if needed.
K_AIR = 1.6               # Normalization factor (semi-quantitative)


MQ135_CAL_SAMPLES = 60
MQ135_CAL_DELAY_S = 0.5
MQ135_WARMUP_S = 120     # warm-up before R0 calibration (seconds)


# ----------------------------
# MiCS-2714 parameters
# ----------------------------
MICS2714_RL = 10000       # 10kΩ load resistor
```

```
MICS2714_VC = 5.0         # assumed divider supply = 5V
MICS2714_CAL_SAMPLES = 60
MICS2714_CAL_DELAY_S = 0.5
MICS2714_WARMUP_S = 60    # optional short warm-up before R0 calibration
(seconds)


# ----------------------------
# MiCS-5524 parameters
# ----------------------------
MICS5524_RL = 10000       # 10kΩ load resistor
MICS5524_VC = 5.0         # assumed divider supply = 5V
MICS5524_CAL_SAMPLES = 60
MICS5524_CAL_DELAY_S = 0.5
MICS5524_WARMUP_S = 60    # optional short warm-up before R0 calibration
(seconds)


R0_mq135 = None
R0_2714 = None
R0_5524 = None


# ----------------------------
# ADC init
# ----------------------------
adc_5524 = ADC(Pin(ADC_PIN_5524))
adc_5524.atten(ADC.ATTN_11DB)
adc_5524.width(ADC.WIDTH_12BIT)


adc_2714 = ADC(Pin(ADC_PIN_2714))
adc_2714.atten(ADC.ATTN_11DB)
adc_2714.width(ADC.WIDTH_12BIT)


adc_mq135 = ADC(Pin(ADC_PIN_MQ135))
adc_mq135.atten(ADC.ATTN_11DB)
adc_mq135.width(ADC.WIDTH_12BIT)


# DHT11
dht_pin = Pin(4)
dht_sensor = dht.DHT11(dht_pin)


# ----------------------------
# PM Sensor (I2C) setup + functions
# ----------------------------
PM_I2C_SCL = 23
PM_I2C_SDA = 21
pm_i2c = I2C(0, scl=Pin(PM_I2C_SCL), sda=Pin(PM_I2C_SDA), freq=100000)


PM_CANDIDATE_ADDRS = [0x19, 0x12]
```

```
def pm_pick_address():
    found = pm_i2c.scan()
    print("PM I2C scan:", [hex(x) for x in found])
    for a in PM_CANDIDATE_ADDRS:
        if a in found:
            print("PM sensor address:", hex(a))
            return a
    if found:
        print("PM: no known addr; using first:", hex(found[0]))
        return found[0]
    print("PM: no I2C device found (check wiring/pins/power).")
    return None


PM_DEVICE_ADDR = pm_pick_address()


def pm_read32(addr, cmd=0x00, delay_ms=10):
    pm_i2c.writeto(addr, bytes([cmd]))
    time.sleep_ms(delay_ms)
    data = pm_i2c.readfrom(addr, 32)
    if len(data) != 32:
        raise RuntimeError("PM: invalid length")
    return data


def pm_parse_correct_little_endian(data):
    """
    Confirmed from your RAW dump:
      PM1.0  = bytes 06-07 (little-endian)
      PM2.5  = bytes 08-09 (little-endian)
      PM10   = bytes 10-11 (little-endian)
    """
    pm1_0 = data[6]  | (data[7]  << 8)
    pm2_5 = data[8]  | (data[9]  << 8)
    pm10  = data[10] | (data[11] << 8)
    return pm1_0, pm2_5, pm10


def read_pm_values(debug=False):
    if PM_DEVICE_ADDR is None:
        return None, None, "no_device"
    try:
        raw = pm_read32(PM_DEVICE_ADDR, cmd=0x00, delay_ms=10)
        vals = pm_parse_correct_little_endian(raw)
        return vals, raw, "i2c_le_fixed"
    except Exception as e:
        if debug:
            print("PM read error:", e)
        return None, None, "error"


# ----------------------------
# Sensor reading functions
# ----------------------------
def read_sensor_with_divider(adc):
    """
```

```
    Reads ADC voltage (after external divider R1/R2), then reconstructs real
sensor Vout.
    - v_div    : voltage actually seen by ADC (0..3.3V)
    - v_sensor : reconstructed real Vout of sensor divider (can be up to ~5V)
    """
    raw = adc.read()
    v_div = (raw / ((1 << ADC_BITS) - 1)) * ADC_REF
    v_sensor = v_div * ((R1 + R2) / R2)
    return raw, v_div, v_sensor


# ----------------------------
# MQ-135 Rs/R0 functions
# ----------------------------
def mq135_compute_rs(vout, rl=MQ135_RL, vc=MQ135_VC):
    """
    Divider:
      Vout = Vc * RL / (Rs + RL)
      => Rs = RL * (Vc - Vout) / Vout
    """
    if vout is None or vout <= 0.0:
        return None
    if vout >= vc:
        vout = vc - 0.01
    return rl * (vc - vout) / vout


def mq135_calibrate_r0(adc_obj, samples=MQ135_CAL_SAMPLES,
delay_s=MQ135_CAL_DELAY_S, k_air=K_AIR):
    """
    Calibrate R0 from measurements in "clean-ish" air:
      1) measure Vout_real (undo external divider)
      2) compute Rs for each sample
      3) R0 = average(Rs_clean) / K_AIR
    """
    s = 0.0
    n = 0
    for _ in range(samples):
        raw, v_div, vout_real = read_sensor_with_divider(adc_obj)
        rs = mq135_compute_rs(vout_real)
        if rs is not None:
            s += rs
            n += 1
        time.sleep(delay_s)
    if n == 0:
        return None
    rs_clean_avg = s / n
    return rs_clean_avg / k_air


def mq135_quality_from_rs_r0(rs_r0):
    if rs_r0 is None:
        return "N/A"
    if rs_r0 > 1.5:
        return "Πολύ καλή"
    elif rs_r0 > 1.0:
        return "Καλή"
```

```
    elif rs_r0 > 0.7:
        return "Μέτρια"
    elif rs_r0 > 0.4:
        return "Κακή"
    else:
        return "Πολύ κακή"


# ----------------------------
# MiCS-2714 Rs/R0 functions
# ----------------------------
def mics2714_compute_rs(vout, rl=MICS2714_RL, vc=MICS2714_VC):
    if vout is None:
        return None
    if vout <= 0.01:
        vout = 0.01
    if vout >= vc:
        vout = vc - 0.01
    return rl * (vc - vout) / vout


def mics2714_calibrate_r0(adc_obj, samples=MICS2714_CAL_SAMPLES,
delay_s=MICS2714_CAL_DELAY_S):
    s = 0.0
    n = 0
    for _ in range(samples):
        raw, v_div, vout_real = read_sensor_with_divider(adc_obj)
        rs = mics2714_compute_rs(vout_real)
        if rs is not None:
            s += rs
            n += 1
        time.sleep(delay_s)
    return (s / n) if n > 0 else None


def mics2714_quality_from_rs_r0(ratio):
    if ratio is None:
        return "N/A"
    if ratio < 0.5:
        return "Πολύ καλή"
    elif ratio < 1.0:
        return "Καλή"
    elif ratio < 2.0:
        return "Μέτρια (Warning)"
    elif ratio < 5.0:
        return "Κακή (High)"
    else:
        return "Πολύ κακή / Κρίσιμη (Critical)"


# ----------------------------
# MiCS-5524 Rs/R0 + scenario tables
# ----------------------------
def mics5524_compute_rs(vout, rl=MICS5524_RL, vc=MICS5524_VC):
    if vout is None:
        return None
    if vout <= 0.01:
```

```
        vout = 0.01
    if vout >= vc:
        vout = vc - 0.01
    return rl * (vc - vout) / vout


def mics5524_calibrate_r0(adc_obj, samples=MICS5524_CAL_SAMPLES,
delay_s=MICS5524_CAL_DELAY_S):
    s = 0.0
    n = 0
    for _ in range(samples):
        raw, v_div, vout_real = read_sensor_with_divider(adc_obj)
        rs = mics5524_compute_rs(vout_real)
        if rs is not None:
            s += rs
            n += 1
        time.sleep(delay_s)
    return (s / n) if n > 0 else None


LEVEL_NUM_TO_TEXT_5524 = {
    0: "Πολύ καλή / Normal",
    1: "Καλή / Caution",
    2: "Μέτρια / Warning",
    3: "Κακή / High",
    4: "Πολύ κακή / Critical",
    -1: "N/A"
}


def mics5524_level_co(ratio):
    if ratio is None:
        return -1
    if ratio >= 0.8:
        return 0
    elif ratio >= 0.45:
        return 1
    elif ratio >= 0.12:
        return 2
    elif ratio >= 0.038:
        return 3
    else:
        return 4


def mics5524_level_voc_like(ratio):
    if ratio is None:
        return -1
    if ratio >= 0.26:
        return 0
    elif ratio >= 0.17:
        return 1
    elif ratio >= 0.12:
        return 2
    elif ratio >= 0.09:
        return 3
    else:
```

```
        return 4


def mics5524_final_level(level_co, level_voc):
    if level_co < 0 and level_voc < 0:
        return -1
    if level_co < 0:
        return level_voc
    if level_voc < 0:
        return level_co
    return level_co if level_co > level_voc else level_voc


# ----------------------------
# Wi-Fi connection function
# ----------------------------
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.disconnect()


    print(" Starting Wi-Fi connection...")
    set_rgb_color(0, 0, 255)
    time.sleep(1)
    available = [net[0].decode("utf-8") for net in wlan.scan()]
    print("Found networks:", available)


    connected = False
    for net in WIFI_NETWORKS:
        ssid = net["ssid"]
        password = net["password"]
        if ssid in available:
            print(f"Trying to connect to {ssid} ...")
            wlan.connect(ssid, password)
            start = time.time()


            while not wlan.isconnected() and time.time() - start < 15:
                set_rgb_color(0, 0, 255)
                time.sleep_ms(300)
                set_rgb_color(0, 0, 0)
                time.sleep_ms(300)


            if wlan.isconnected():
                print(f" Connected to {ssid}: {wlan.ifconfig()}")
                set_rgb_color(0, 255, 255)
                time.sleep(1)
                connected = True
                break
            else:
                print(f" Failed to connect to {ssid}")
        else:
            print(f" {ssid} not found in range")
```

```
    set_rgb_color(0, 0, 0)
    if not connected:
        print(" No Wi-Fi connected")
    return wlan.isconnected()


# ----------------------------
# ThingSpeak upload function (fields 6-8 are PM)
# ----------------------------
def upload_data(temp, hum, raw_mq135, raw_5524, raw_2714, pm1_0, pm2_5, pm10):
    print("Uploading data to ThingSpeak...")
    for _ in range(3):
        led_upload.value(1)
        set_rgb_color(0, 255, 0)
        time.sleep_ms(300)
        led_upload.value(0)
        set_rgb_color(0, 0, 0)
        time.sleep_ms(300)
    try:
        url = (
            f"{THINGSPEAK_URL}?api_key={API_KEY}"

f"&field1={temp}&field2={hum}&field3={raw_mq135}&field4={raw_5524}&field5={raw
_2714}"
            f"&field6={pm1_0}&field7={pm2_5}&field8={pm10}"
        )
        response = urequests.get(url)
        print("ThingSpeak response:", response.text)
        response.close()
    except Exception as e:
        print("Upload failed:", e)


# ----------------------------
# MAIN
# ----------------------------
print("PM sensor warm-up (recommended)...")
time.sleep(10)


# MQ-135 warm-up + baseline R0 calibration (AUTO)
print("MQ-135 warm-up (recommended)...", MQ135_WARMUP_S, "sec")
time.sleep(MQ135_WARMUP_S)


print("MQ-135 calibration (baseline R0_mq135)...")
print("  RL =", MQ135_RL, "ohm")
print("  Vc =", MQ135_VC, "V")
print("  K_air =", K_AIR)
R0_mq135 = mq135_calibrate_r0(adc_mq135)
print("  Estimated R0_mq135 =", R0_mq135, "ohm")
print("-" * 60)


# MiCS-2714 warm-up + baseline R0 calibration
print("MiCS-2714 warm-up (optional)...", MICS2714_WARMUP_S, "sec")
time.sleep(MICS2714_WARMUP_S)
```

```
print("MiCS-2714 calibration (baseline R0_2714)...")
print("  RL =", MICS2714_RL, "ohm")
print("  Vc =", MICS2714_VC, "V")
R0_2714 = mics2714_calibrate_r0(adc_2714)
print("  Estimated R0_2714 =", R0_2714, "ohm")
print("-" * 60)


# MiCS-5524 warm-up + baseline R0 calibration
print("MiCS-5524 warm-up (optional)...", MICS5524_WARMUP_S, "sec")
time.sleep(MICS5524_WARMUP_S)


print("MiCS-5524 calibration (baseline R0_5524)...")
print("  RL =", MICS5524_RL, "ohm")
print("  Vc =", MICS5524_VC, "V")
R0_5524 = mics5524_calibrate_r0(adc_5524)
print("  Estimated R0_5524 =", R0_5524, "ohm")
print("-" * 60)


while True:
    if not network.WLAN(network.STA_IF).isconnected():
        connect_wifi()
        time.sleep(2)


    # ----------------------------
    # Read gas sensors (ALL WITH DIVIDER)
    # ----------------------------
    raw_5524, v_div_5524, v_sensor_5524 = read_sensor_with_divider(adc_5524)
    raw_2714, v_div_2714, v_sensor_2714 = read_sensor_with_divider(adc_2714)
    raw_mq135, v_div_mq135, v_sensor_mq135 =
read_sensor_with_divider(adc_mq135)


    # ----------------------------
    # MQ-135 Rs/R0 + quality
    # ----------------------------
    rs_mq135 = mq135_compute_rs(v_sensor_mq135)
    rs_r0_mq135 = (rs_mq135 / R0_mq135) if (rs_mq135 is not None and R0_mq135
not in (None, 0)) else None
    mq135_quality = mq135_quality_from_rs_r0(rs_r0_mq135)


    # ----------------------------
    # MiCS-2714 Rs/R0 + quality
    # ----------------------------
    rs_2714 = mics2714_compute_rs(v_sensor_2714)
    rs_r0_2714 = (rs_2714 / R0_2714) if (rs_2714 is not None and R0_2714 not
in (None, 0)) else None
    mics2714_quality = mics2714_quality_from_rs_r0(rs_r0_2714)


    # ----------------------------
```

```
    # MiCS-5524 Rs/R0 + CO/VOC scenarios
    # ----------------------------
    rs_5524 = mics5524_compute_rs(v_sensor_5524)
    rs_r0_5524 = (rs_5524 / R0_5524) if (rs_5524 is not None and R0_5524 not
in (None, 0)) else None


    lvl_co_5524 = mics5524_level_co(rs_r0_5524)
    lvl_voc_5524 = mics5524_level_voc_like(rs_r0_5524)
    lvl_final_5524 = mics5524_final_level(lvl_co_5524, lvl_voc_5524)


    # ----------------------------
    # Read DHT11
    # ----------------------------
    try:
        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
    except OSError:
        temp = None
        hum = None


    # ----------------------------
    # Read PM sensor
    # ----------------------------
    pm_vals, pm_raw, pm_mode = read_pm_values(debug=True)
    if pm_vals:
        pm1_0, pm2_5, pm10 = pm_vals
    else:
        pm1_0, pm2_5, pm10 = 0, 0, 0


    # ----------------------------
    # BUZZER trigger logic (ADDED)
    # ----------------------------
    alarm = False


    # DHT thresholds
    if temp is not None and temp > TH_TEMP_C:
        alarm = True
    if hum is not None and hum > TH_HUM_PCT:
        alarm = True


    # PM thresholds
    if pm2_5 is not None and pm2_5 > TH_PM25:
        alarm = True
    if pm10 is not None and pm10 > TH_PM10:
        alarm = True


    # Gas thresholds (ratios)
    if rs_r0_mq135 is not None and rs_r0_mq135 < TH_MQ135_RS_R0_LOW:
        alarm = True
```

```
    if rs_r0_2714 is not None and rs_r0_2714 > TH_MICS2714_RS_R0_HIGH:
        alarm = True
    if rs_r0_5524 is not None and rs_r0_5524 < TH_MICS5524_RS_R0_LOW:
        alarm = True


    buzzer.value(1 if alarm else 0)


    # ----------------------------
    # Print sensor values
    # ----------------------------
    print("MiCS-5524 -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs:
{:.0f}Ω | Rs/R0: {:.3f}".format(
        raw_5524,
        v_div_5524,
        v_sensor_5524,
        rs_5524 if rs_5524 is not None else -1,
        rs_r0_5524 if rs_r0_5524 is not None else -1
    ))
    print("           CO scenario  level:",
LEVEL_NUM_TO_TEXT_5524.get(lvl_co_5524, "UNK"))
    print("           VOC scenario level:",
LEVEL_NUM_TO_TEXT_5524.get(lvl_voc_5524, "UNK"))
    print("           FINAL (worst-case) :",
LEVEL_NUM_TO_TEXT_5524.get(lvl_final_5524, "UNK"))


    print("MiCS-2714 -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs:
{:.0f}Ω | Rs/R0: {:.3f} | Level: {}".format(
        raw_2714,
        v_div_2714,
        v_sensor_2714,
        rs_2714 if rs_2714 is not None else -1,
        rs_r0_2714 if rs_r0_2714 is not None else -1,
        mics2714_quality
    ))


    print("MQ-135    -> raw: {:4d} | Vdiv: {:.3f} V | Vout: {:.3f} V | Rs:
{:.0f}Ω | Rs/R0: {:.3f} | Quality: {}".format(
        raw_mq135,
        v_div_mq135,
        v_sensor_mq135,
        rs_mq135 if rs_mq135 is not None else -1,
        rs_r0_mq135 if rs_r0_mq135 is not None else -1,
        mq135_quality
    ))


    if temp is not None and hum is not None:
        print("DHT11     -> Temp: {}°C | Humidity: {}%".format(temp, hum))
    else:
        print("DHT11     -> Error reading sensor")
```

```
    print("PM (mode: {}) -> PM1.0: {} | PM2.5: {} | PM10: {}".format(pm_mode,
pm1_0, pm2_5, pm10))
    print("BUZZER    ->", "ON (ALARM)" if alarm else "OFF")
    print("-" * 60)


    # ----------------------------
    # Upload to ThingSpeak
    # ----------------------------
    upload_data(
        temp if temp is not None else 0,
        hum if hum is not None else 0,
        raw_mq135, raw_5524, raw_2714,
        pm1_0, pm2_5, pm10
    )


    # ----------------------------
    # Waiting blink (red)
    # ----------------------------
    for _ in range(10):
        led_waiting.value(1)
        set_rgb_color(255, 0, 0)
        time.sleep(1)
        led_waiting.value(0)
        set_rgb_color(0, 0, 0)
        time.sleep(1)
```

ΠΑΡΑΡΤΗΜΑ Β: Κώδικας MATLAB για την ειδοποίηση του χρήστη μέσω e-mail (ThingSpeak)

```
% ============================================================
% ThingSpeak MATLAB Analysis: Air Quality Alerts (FINAL)
% - Reads latest values from main channel
% - Checks thresholds
% - Cooldown using AlertLog (Field1 = last_alert_unix)
% - Sends email via ThingSpeak Alerts API (JSON + header)
% ============================================================


%% ===== USER SETTINGS =====
CHANNEL_ID = 3158331;
READ_API_KEY = ''; % main channel read key if private
ALERTS_API_KEY = 'TAKctwP0Ji2SnvYaiTs'; % Alerts API Key (profile)


% ---- AlertLog channel (Field1 must exist: last_alert_unix) ----
ALERT_LOG_CHANNEL_ID = 3228205;
ALERT_LOG_READ_KEY = 'VAFQ6P5KY4W7DACS'; % leave '' if public
ALERT_LOG_WRITE_KEY = 'TANL6U0KPXRBAVQA';


COOLDOWN_MIN = 60; % 1 alert per 15 minutes (increase if needed)


%% ===== FIELD MAPPING (main channel) =====
F_TEMP = 1;
F_HUM = 2;
F_MQ135 = 3;
F_MICS5524 = 4;
F_MICS2714 = 5;
F_PM1 = 6;
F_PM25 = 7;
F_PM10 = 8;


%% ===== THRESHOLDS =====
PM25_WARN = 20; % WHO 2021 (24h guideline)
PM10_WARN = 45; % WHO 2021 (24h guideline)
PM1_WARN = 10; % indicative warning (no official WHO limit)


TEMP_LOW = 10;
TEMP_HIGH = 30;


BASE_MQ135 = 0; % set >0 to enable gas raw checks
BASE_MICS5524 = 0;
BASE_MICS2714 = 0;


GAS_WARN_PCT = 0.30;
GAS_ALRM_PCT = 0.60;
```

```
%% ===== READ LAST SAMPLE (MAIN CHANNEL) =====
fields = [F_TEMP F_HUM F_MQ135 F_MICS5524 F_MICS2714 F_PM1 F_PM25 F_PM10];


if isempty(READ_API_KEY)
data = thingSpeakRead(CHANNEL_ID, 'Fields', fields, 'NumPoints', 1,
'OutputFormat', 'matrix');
else
data = thingSpeakRead(CHANNEL_ID, 'ReadKey', READ_API_KEY, 'Fields', fields,
'NumPoints', 1, 'OutputFormat', 'matrix');
end


data = data(end,:);


temp = data(1);
hum = data(2);
mq135 = data(3);
m5524 = data(4);
m2714 = data(5);
pm1 = data(6);
pm25 = data(7);
pm10 = data(8);


%% ===== CHECKS =====
issues = {};


if ~isnan(temp)
if (temp < TEMP_LOW) || (temp > TEMP_HIGH)
issues{end+1} = sprintf('Temperature out of range: %.1f C (target %d-%d C)',
temp, TEMP_LOW, TEMP_HIGH);
end
end


if ~isnan(pm1) && (pm1 >= PM1_WARN)
issues{end+1} = sprintf('PM1 elevated: %.1f ug/m3 (>= %d)', pm1, PM1_WARN);
end
if ~isnan(pm25) && (pm25 >= PM25_WARN)
issues{end+1} = sprintf('PM2.5 high: %.1f ug/m3 (WHO 24h guideline %d)', pm25,
PM25_WARN);
end
if ~isnan(pm10) && (pm10 >= PM10_WARN)
issues{end+1} = sprintf('PM10 high: %.1f ug/m3 (WHO 24h guideline %d)', pm10,
PM10_WARN);
end


if (BASE_MQ135 > 0) && ~isnan(mq135)
if mq135 >= BASE_MQ135*(1+GAS_ALRM_PCT)
issues{end+1} = sprintf('MQ-135 RAW ALARM: %d (baseline %d, +%.0f%%)',
round(mq135), round(BASE_MQ135), GAS_ALRM_PCT*100);
elseif mq135 >= BASE_MQ135*(1+GAS_WARN_PCT)
```

```
issues{end+1} = sprintf('MQ-135 RAW WARN: %d (baseline %d, +%.0f%%)',
round(mq135), round(BASE_MQ135), GAS_WARN_PCT*100);
end
end


if (BASE_MICS5524 > 0) && ~isnan(m5524)
if m5524 >= BASE_MICS5524*(1+GAS_ALRM_PCT)
issues{end+1} = sprintf('MiCS-5524 RAW ALARM: %d (baseline %d, +%.0f%%)',
round(m5524), round(BASE_MICS5524), GAS_ALRM_PCT*100);
elseif m5524 >= BASE_MICS5524*(1+GAS_WARN_PCT)
issues{end+1} = sprintf('MiCS-5524 RAW WARN: %d (baseline %d, +%.0f%%)',
round(m5524), round(BASE_MICS5524), GAS_WARN_PCT*100);
end
end


if (BASE_MICS2714 > 0) && ~isnan(m2714)
if m2714 >= BASE_MICS2714*(1+GAS_ALRM_PCT)
issues{end+1} = sprintf('MiCS-2714 RAW ALARM: %d (baseline %d, +%.0f%%)',
round(m2714), round(BASE_MICS2714), GAS_ALRM_PCT*100);
elseif m2714 >= BASE_MICS2714*(1+GAS_WARN_PCT)
issues{end+1} = sprintf('MiCS-2714 RAW WARN: %d (baseline %d, +%.0f%%)',
round(m2714), round(BASE_MICS2714), GAS_WARN_PCT*100);
end
end


%% ===== SEND EMAIL IF NEEDED (COOLDOWN via AlertLog Field1) =====
if isempty(issues)
disp('No alerts.');
return;
end


% ---- Read last alert unix seconds (avoid warning by not demanding 1 point) -
---
now_unix = posixtime(datetime('now','TimeZone','UTC'));
last_unix = NaN;


try
if isempty(ALERT_LOG_READ_KEY)
last_unix_mat = thingSpeakRead(ALERT_LOG_CHANNEL_ID, 'Fields', 1, 'NumPoints',
1, 'OutputFormat', 'matrix');
else
last_unix_mat = thingSpeakRead(ALERT_LOG_CHANNEL_ID, 'ReadKey',
ALERT_LOG_READ_KEY, 'Fields', 1, 'NumPoints', 1, 'OutputFormat', 'matrix');
end
if ~isempty(last_unix_mat)
last_unix = last_unix_mat(end,1);
end
catch
last_unix = NaN;
end
```

```
if isnan(last_unix) || last_unix <= 0
minutes_since_last = inf;
else
minutes_since_last = (now_unix - last_unix) / 60.0;
end


if minutes_since_last < COOLDOWN_MIN
disp(sprintf('Cooldown active: last alert %.1f min ago (< %d). Not sending.',
minutes_since_last, COOLDOWN_MIN));
return;
end


% Build issues text
issuesText = '';
for k = 1:numel(issues)
issuesText = [issuesText, sprintf('\n- %s', issues{k})]; %#ok<AGROW>
end


subject = 'Air Quality Alert (ThingSpeak)';
body = sprintf([ ...
'Alert triggered.\n\n' ...
'Latest values:\n' ...
'Temp: %.1f C\nHum: %.1f %%\n' ...
'PM1: %.1f ug/m3\nPM2.5: %.1f ug/m3\nPM10: %.1f ug/m3\n' ...
'MQ-135 RAW: %.0f\nMiCS-5524 RAW: %.0f\nMiCS-2714 RAW: %.0f\n\n' ...
'Issues:%s\n'], ...
temp, hum, pm1, pm25, pm10, mq135, m5524, m2714, issuesText);


% Pre-log to enforce cooldown even if sending fails / gets 429
try
thingSpeakWrite(ALERT_LOG_CHANNEL_ID, 1, now_unix, 'WriteKey',
ALERT_LOG_WRITE_KEY);
catch
% If this fails, you may retry too often. Keep your TimeControl interval
larger.
end


% Send alert
alertUrl = 'https://api.thingspeak.com/alerts/send';


options = weboptions( ...
'MediaType', 'application/json', ...
'Timeout', 20, ...
'HeaderFields', { ...
'ThingSpeak-Alerts-API-Key', ALERTS_API_KEY; ...
'Accept', 'application/json' ...
} ...
);


payload = struct('subject', subject, 'body', body);
```

```
try
resp = webwrite(alertUrl, payload, options);
disp('Alert sent.');
disp(resp);
catch ME
disp('Failed to send alert:');
disp(ME.message);


% If you hit 429, do NOT retry now. Cooldown will block next runs.
return;
end
```