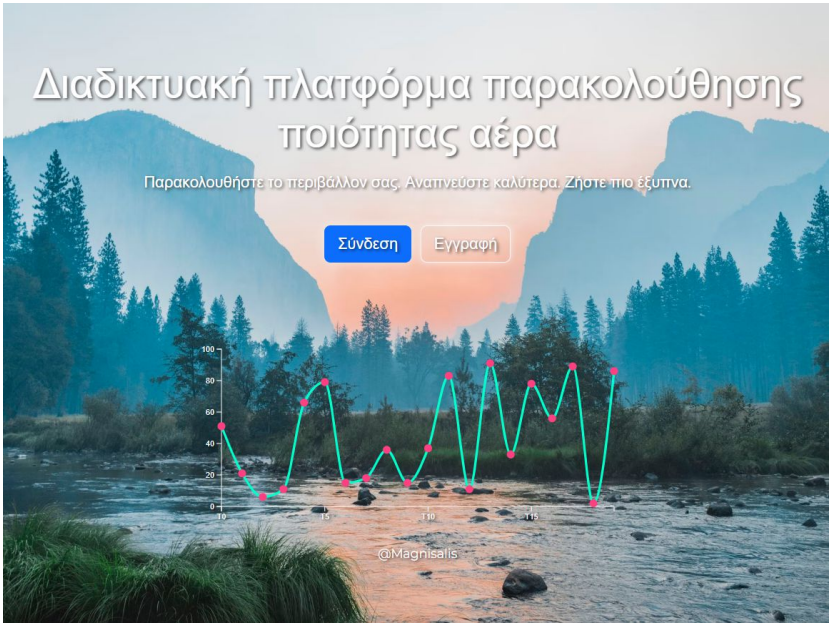
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και υλοποίηση διαδικτυακής πλατφόρμας παρακολούθησης ποιότητας αέρα»



Φοιτητής

ΙΩΑΝΝΗΣ ΜΑΓΝΗΣΑΛΗΣ - 512170

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Ιούνιος 2025

Σχεδιασμός και υλοποίηση διαδικτυακής πλατφόρμας παρακολούθησης ποιότητας αέρα

Κωδικός: 25153

Φοιτητής: Μαγνήσαλης Ιωάννης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 08-03-2025

Ημερομηνία περάτωσης Π.Ε. 31-05-2025

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής 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.

Περίληψη

Η εργασία αυτή αφορά τη δημιουργία ενός σύγχρονου συστήματος παρακολούθησης της ποιότητας του αέρα που βασίζεται σε συσκευές ESP32 και έναν κεντρικό server υλοποιημένο με Python. Οι συσκευές μετρούν βασικές περιβαλλοντικές παραμέτρους όπως τα σωματίδια PM2.5 και PM10, και αποστέλλουν τα δεδομένα στον server. Οι μετρήσεις αποθηκεύονται σε βάση δεδομένων και προβάλλονται μέσα από ένα εύχρηστο web περιβάλλον. Οι χρήστες μπορούν να παρακολουθούν σε πραγματικό χρόνο τις μετρήσεις, να βλέπουν γραφήματα και πίνακες και να κατεβάζουν τα δεδομένα για περαιτέρω ανάλυση. Το σύστημα υποστηρίζει επίσης την αναγνώριση του κάθε χρήστη και των συσκευών του. Με αυτόν τον τρόπο επιτυγχάνεται καλύτερη προβολή της ποιότητας του αέρα και ενημέρωση των πολιτών για πιθανούς κινδύνους που σχετίζονται με την ατμόσφαιρα.

« Design and implementation of an online air quality monitoring platform »

Abstract

This work concerns the creation of a modern air quality monitoring system, which is based on ESP32 devices and a central server implemented in Python. The devices measure basic environmental parameters, such as PM2.5 and PM10 particles, and send the data to the server. The measurements are stored in a database and displayed through an easy-to-use web environment. Users can monitor the measurements in real time, view graphs and tables and download the data for further analysis. The system also supports the identification of each user and their devices offering personalized information. In this way, a better understanding of air quality is achieved and citizens are informed about potential risks related to the atmosphere.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου και του φίλους μας και τους καθηγητές μου για την πολύτιμη βοήθεια τους.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η ποιότητα του αέρα που αναπνέουμε αποτελεί έναν από τους σημαντικότερους παράγοντες που επηρεάζουν άμεσα την ανθρώπινη υγεία και την ποιότητα ζωής. Η βιομηχανική δραστηριότητα εντείνεται και η ατμοσφαιρική ρύπανση αυξάνεται συνεχώς και δημιουργεί σοβαρούς κινδύνους για το περιβάλλον αλλά και για την υγεία των ανθρώπων. Η παρακολούθηση της ποιότητας του αέρα είναι πλέον απαραίτητη όχι μόνο σε μεγάλες πόλεις αλλά και σε μικρότερες κοινότητες καθώς η ρύπανση δεν έχει γεωγραφικά όρια.

Ένα από τα βασικά στοιχεία που συνθέτουν την ποιότητα του αέρα είναι τα αιωρούμενα σωματίδια, γνωστά και ως PM (Particulate Matter). Τα σωματίδια αυτά διακρίνονται ανάλογα με το μέγεθός τους σε PM1.0, PM2.5 και PM10, όπου ο αριθμός δηλώνει τη διάμετρο των σωματιδίων σε μικρόμετρα (μm). Όσο μικρότερα είναι τα σωματίδια, τόσο πιο επικίνδυνα είναι για την ανθρώπινη υγεία καθώς μπορούν να διεισδύσουν βαθύτερα στο αναπνευστικό σύστημα.

- Τα σωματίδια **PM10** εισχωρούν στους ανώτερους αεραγωγούς και μπορούν να προκαλέσουν αναπνευστικές ενοχλήσεις.
- Τα **PM2.5** είναι πιο επικίνδυνα καθώς εισχωρούν βαθύτερα στους πνεύμονες και έχουν συνδεθεί με καρδιοαναπνευστικές παθήσεις.
- Τα ακόμη μικρότερα **PM1.0** μπορούν να φτάσουν μέχρι και την κυκλοφορία του αίματος, προκαλώντας σοβαρά προβλήματα υγείας, όπως καρδιαγγειακά νοσήματα και επιπτώσεις στο νευρικό σύστημα.

Η συνεχής μέτρηση αυτών των σωματιδίων είναι ζωτικής σημασίας για την πρόληψη προβλημάτων υγείας, την έγκαιρη ενημέρωση του πληθυσμού και την εφαρμογή κατάλληλων μέτρων προστασίας.

Στην αγορά σήμερα διατίθενται διάφορα συστήματα μέτρησης της ποιότητας του αέρα. Αυτά κυμαίνονται από επαγγελματικούς σταθμούς υψηλού κόστους και ακρίβειας που χρησιμοποιούνται σε εθνικά και διεθνή δίκτυα παρακολούθησης έως και φορητές ή οικιακές συσκευές που είναι για προσωπική χρήση. Παρά την ύπαρξη τέτοιων λύσεων τα περισσότερα από αυτά τα συστήματα παρουσιάζουν σημαντικούς περιορισμούς: είτε είναι πολύ ακριβά και δύσχρηστα για τον απλό πολίτη, είτε δεν παρέχουν δυνατότητα αποθήκευσης και ανάλυσης ιστορικών δεδομένων σε διαδικτυακό περιβάλλον.

Η ανάγκη για άμεση και έγκυρη ενημέρωση των πολιτών σχετικά με την ποιότητα του αέρα έχει οδηγήσει στην ανάπτυξη διαδικτυακών συστημάτων και εφαρμογών που συλλέγουν, αναλύουν και

παρουσιάζουν μετρήσεις από διάφορες τοποθεσίες. Μέσω αυτών των πλατφορμών, οι χρήστες μπορούν να ενημερωθούν για τα επίπεδα ρύπανσης σε πραγματικό χρόνο, να δουν ιστορικά δεδομένα και να λάβουν ειδοποιήσεις σε περίπτωση που οι τιμές υπερβαίνουν τα επιτρεπτά όρια. Όμως ακόμα και αυτά τα συστήματα συχνά δεν καλύπτουν πλήρως τις ανάγκες εξατομικευμένης παρακολούθησης και διαχείρισης των συσκευών από τους ίδιους τους χρήστες.

Για να καλυφθεί αυτό το κενό, αναπτύχθηκε το παρόν σύστημα παρακολούθησης ποιότητας αέρα, το οποίο προσφέρει μια οικονομική λύση. Το σύστημα χρησιμοποιεί αισθητήρες συνδεδεμένους με μικροελεγκτές ESP32 οι οποίοι μετρούν τα επίπεδα PM1.0, PM2.5 και PM10 και αποστέλλουν τα δεδομένα σε έναν κεντρικό server. Εκεί τα δεδομένα αποθηκεύονται, αναλύονται και προβάλλονται μέσω μιας εύχρηστης διαδικτυακής εφαρμογής.

Μία σημαντική προσθήκη που εισάγει το παρόν σύστημα σε σύγκριση με άλλες αντίστοιχες εφαρμογές είναι η δυνατότητα αναγνώρισης και καταγραφής του χρήστη για κάθε συσκευή (ESP32). Κάθε συσκευή έχει ένα μοναδικό αναγνωριστικό (UUID) και συνδέεται με έναν συγκεκριμένο χρήστη. Αυτό επιτρέπει:

- Να γνωρίζει ο χρήστης ποιες συσκευές του ανήκουν.
- Να προσθέτει νέες συσκευές στο προσωπικό του προφίλ.
- Να αποδεσμεύει συσκευές που δεν χρησιμοποιεί πλέον.
- Να έχει πρόσβαση μόνο στα δεδομένα που αφορούν τις δικές του συσκευές.

Η λειτουργία αυτή ενισχύει την ασφάλεια, διευκολύνει την παρακολούθηση και επιτρέπει στον χρήστη να έχει πλήρη έλεγχο των συσκευών του και των μετρήσεών τους. Ακόμη μέσω της διαδικτυακής πλατφόρμας δίνεται δυνατότητα γραφικής απεικόνισης των δεδομένων, ανάλυσης ιστορικών τιμών και λήψης αποφάσεων για την προστασία της υγείας του.

1.2 Η δομή του κειμένου

Στο πρώτο κεφάλαιο παρουσιάζεται η γενική εισαγωγή στο θέμα της ποιότητας του αέρα και αναδεικνύεται η σημασία της παρακολούθησης των αιωρούμενων σωματιδίων για την προστασία της ανθρώπινης υγείας. Ακολουθεί η ανάλυση της δομής του κειμένου ώστε ο αναγνώστης να γνωρίζει τι θα ακολουθήσει στα επόμενα κεφάλαια.

Στο δεύτερο κεφάλαιο παρουσιάζονται σχετικά συστήματα παρακολούθησης ποιότητας αέρα που έχουν αναπτυχθεί σε ερευνητικό ή πρακτικό επίπεδο καθώς και η λειτουργία του Εθνικού Δικτύου Παρακολούθησης Ατμοσφαιρικής Ρύπανσης.

Το τρίτο κεφάλαιο επικεντρώνεται στην ανάλυση των τεχνολογικών εργαλείων που χρησιμοποιήθηκαν στο έργο, όπως η πλακέτα esp32, ο αισθητήρας pms5003, η γλώσσα προγραμματισμού Python και τα εργαλεία Django, Google Charts και Tabulator.js, τα οποία υποστηρίζουν την ανάπτυξη και οπτικοποίηση του συστήματος.

Στο τέταρτο κεφάλαιο αναλύεται λεπτομερώς η διαδικτυακή πλατφόρμα που αναπτύχθηκε για την παρακολούθηση της ποιότητας του αέρα. Παρουσιάζεται η συνολική λειτουργία του συστήματος, καθώς και οι επιμέρους λειτουργίες τόσο από την πλευρά της συσκευής esp32 όσο και από την πλευρά του server. Επίσης, γίνεται αναφορά στη βάση δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των μετρήσεων και των στοιχείων των χρηστών.

Στο πέμπτο και τελευταίο κεφάλαιο γίνεται αξιολόγηση του συστήματος, παρουσιάζονται τα πλεονεκτήματα και οι αδυναμίες του και προτείνονται πιθανές βελτιώσεις που θα μπορούσαν να ενισχύσουν περαιτέρω την απόδοση και την αποτελεσματικότητά του.

Η εργασία ολοκληρώνεται με τη βιβλιογραφία, όπου καταγράφονται όλες οι πηγές που χρησιμοποιήθηκαν, και το παράρτημα το οποίο περιλαμβάνει συμπληρωματικό υλικό που υποστηρίζει το περιεχόμενο της μελέτης.

Κεφάλαιο 2ο: Συστήματα παρακολούθησης ποιότητας αέρα

2.1 Σύστημα παρακολούθησης αιωρούμενων σωματιδίων σε εσωτερικούς χώρους (Marques et al., 2018)

Η συνεχής παρακολούθηση της ποιότητας του αέρα στους εσωτερικούς χώρους αποτελεί ζήτημα ολοένα και μεγαλύτερης σημασίας όπου οι άνθρωποι περνούν μεγάλο μέρος της ημέρας εντός κτηρίων – είτε πρόκειται για σπίτια, σχολεία, γραφεία ή νοσοκομεία. Η εργασία των Marques, Roque Ferreira και Pitarma (2018) αναγνωρίζει αυτή την ανάγκη και παρουσιάζει ένα καινοτόμο σύστημα βασισμένο στο Διαδίκτυο των Πραγμάτων (IoT), το οποίο σχεδιάστηκε για την παρακολούθηση της συγκέντρωσης αιωρούμενων σωματιδίων σε πραγματικό χρόνο, εστιάζοντας ειδικά σε εσωτερικά περιβάλλοντα [1].

Το σύστημα που περιγράφεται στην εργασία βασίζεται στη χρήση αισθητήρων για τη μέτρηση συγκεντρώσεων PM (κυρίως PM2.5 και PM10), οι οποίοι είναι συνδεδεμένοι με μικροελεγκτές που υποστηρίζουν ασύρματη επικοινωνία μέσω Wi-Fi. Αυτοί οι αισθητήρες συλλέγουν δεδομένα με μεγάλη συχνότητα και τα αποστέλλουν σε μια διαδικτυακή πλατφόρμα που είναι υπεύθυνη για την αποθήκευση, οπτικοποίηση και ανάλυση των δεδομένων. Ένα από τα βασικά πλεονεκτήματα του συστήματος είναι ότι παρέχει τη δυνατότητα συνεχούς εποπτείας ακόμη και όταν ο χρήστης δεν βρίσκεται στον ίδιο χώρο με τον αισθητήρα.

Οι συγγραφείς τονίζουν ιδιαίτερα τη χρησιμότητα της άμεσης ενημέρωσης των χρηστών σε περίπτωση που οι συγκεντρώσεις σωματιδίων υπερβαίνουν τα αποδεκτά όρια που έχουν θεσπιστεί από διεθνείς οργανισμούς υγείας. Αυτό επιτρέπει στους χρήστες να λαμβάνουν άμεσα μέτρα για τη βελτίωση της ποιότητας του αέρα, όπως το άνοιγμα παραθύρων, η ενεργοποίηση εξαερισμού ή η χρήση συσκευών καθαρισμού αέρα. Η δυνατότητα παρακολούθησης και αντίδρασης σε πραγματικό χρόνο καθιστά το σύστημα ιδιαίτερα χρήσιμο σε ευάλωτους πληθυσμούς, όπως παιδιά, ηλικιωμένους και άτομα με αναπνευστικά προβλήματα.

Το σύστημα που περιγράφεται είναι σχεδιασμένο με γνώμονα την προσβασιμότητα και το χαμηλό κόστος. Οι συγγραφείς επισημαίνουν ότι η χρήση οικονομικών αισθητήρων και απλών μικροελεγκτών (όπως Arduino ή ESP8266) καθιστά τη λύση αυτή κατάλληλη για ευρεία εφαρμογή σε οικιακά και μικρής κλίμακας περιβάλλοντα χωρίς να απαιτούνται εξειδικευμένες τεχνικές γνώσεις. Η αρχιτεκτονική του συστήματος είναι ευέλικτη και μπορεί να επεκταθεί ώστε να παρακολουθεί και άλλες παραμέτρους, όπως θερμοκρασία, υγρασία ή διοξείδιο του άνθρακα (CO_2), προσφέροντας έτσι μια πιο ολοκληρωμένη εικόνα για τις συνθήκες ενός εσωτερικού περιβάλλοντος.

Η εργασία των Marques et al. (2018) αποτελεί χαρακτηριστικό παράδειγμα της μετάβασης από τα παραδοσιακά, σταθερά και υψηλού κόστους συστήματα μέτρησης σε ευέλικτες, έξυπνες και προσβάσιμες λύσεις βασισμένες στο IoT. Αναδεικνύει την αξία της τεχνολογίας ως εργαλείο

προστασίας της υγείας και δημιουργεί το υπόβαθρο για την ανάπτυξη νέων εφαρμογών και βελτιωμένων εκδόσεων του συστήματος. Παράλληλα, θέτει τις βάσεις για την ένταξη των πολιτών σε μια ενεργή διαδικασία παρακολούθησης και βελτίωσης του περιβάλλοντός τους δίνοντάς τους τη δύναμη της πληροφόρησης μέσα από εύχρηστες τεχνολογικές λύσεις.

Η εργασία αυτή συνδέεται άμεσα με τη λογική του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας μελέτης και αξιοποιεί παρόμοιες τεχνολογίες δίνοντας στον χρήστη πρόσβαση σε δεδομένα σε πραγματικό χρόνο και στοχεύοντας στη βελτίωση της ποιότητας του αέρα στους χώρους όπου ζει και εργάζεται.

2.2 Παρακολούθηση της ποιότητας του αέρα με τη χρήση τεχνολογιών του Διαδικτύου των Πραγμάτων (Divan etc 2021)

Σύμφωνα με την εργασία των Divan, Sanchez-Reynoso, Panebianco και Mendez (2021) [2], εξετάζονται σύγχρονες προσεγγίσεις για την παρακολούθηση της ποιότητας του αέρα με τη χρήση τεχνολογιών του Διαδικτύου των Πραγμάτων (IoT) εστιάζοντας ιδιαίτερα στον εντοπισμό και την ανάλυση των αιωρούμενων σωματιδίων και της επίδρασής τους στην ανθρώπινη υγεία. Η εργασία αυτή δίνει έμφαση στην ανάγκη για ανάπτυξη έξυπνων συστημάτων που όχι μόνο συλλέγουν δεδομένα, αλλά και επεξεργάζονται τις πληροφορίες ώστε να παρέχουν χρήσιμα συμπεράσματα για την πρόληψη επιβλαβών συνεπειών από τη ρύπανση.

Στο πλαίσιο της μελέτης τους, οι ερευνητές παρουσιάζουν διάφορα παραδείγματα υλοποίησης συστημάτων IoT για την παρακολούθηση των σωματιδίων PM1.0, PM2.5 και PM10. Η έρευνα τονίζει ότι οι συγκεντρώσεις αυτών των σωματιδίων έχουν άμεση σχέση με σοβαρές παθήσεις, όπως καρδιοαναπνευστικά νοσήματα, εγκεφαλικά επεισόδια και αναπνευστικά προβλήματα, ιδιαίτερα σε ευαίσθητες κοινωνικές ομάδες, όπως παιδιά, ηλικιωμένοι και άτομα με προϋπάρχουσες χρόνιες ασθένειες.

Η εργασία των Divan et al. επισημαίνει επίσης ότι ένα βασικό πρόβλημα των παραδοσιακών σταθμών μέτρησης είναι το υψηλό κόστος εγκατάστασης και συντήρησής τους, καθώς και η αδυναμία παροχής άμεσης πληροφόρησης στον πολίτη. Αντίθετα τα IoT συστήματα παρέχουν τη δυνατότητα δημιουργίας ενός εκτεταμένου δικτύου αισθητήρων, που μπορούν να τοποθετηθούν εύκολα σε διάφορες περιοχές, ακόμα και σε απομακρυσμένα σημεία, εξασφαλίζοντας μεγαλύτερη χωρική κάλυψη και καλύτερη καταγραφή των επιπέδων ρύπανσης.

Η μελέτη δίνει ιδιαίτερη έμφαση στη σημασία της ανάλυσης των συλλεγόμενων δεδομένων. Δεν αρκεί απλώς η συγκέντρωση μετρήσεων αλλά είναι κρίσιμο να εφαρμόζονται τεχνικές ανάλυσης για να εντοπίζονται τάσεις, να προβλέπονται μελλοντικές αυξήσεις των ρύπων και να λαμβάνονται εγκαίρως

προληπτικά μέτρα. Οι ερευνητές αναφέρονται επίσης στην ενσωμάτωση αλγορίθμων τεχνητής νοημοσύνης για την έξυπνη επεξεργασία των δεδομένων προκειμένου να εντοπίζονται αυτόματα επικίνδυνες καταστάσεις και να αποστέλλονται ειδοποιήσεις στους πολίτες ή στις αρμόδιες αρχές.

Ένα ακόμη σημαντικό στοιχείο που αναφέρεται στην εργασία είναι η ανάγκη δημιουργίας ανοικτών και συνεργατικών πλατφορμών, οι οποίες θα επιτρέπουν την εύκολη πρόσβαση των πολιτών και των επιστημόνων στα δεδομένα που συλλέγονται. Με αυτόν τον τρόπο ενισχύεται η συμμετοχή του κοινού στην κατανόηση και την αντιμετώπιση των προβλημάτων που σχετίζονται με την ποιότητα του αέρα, ενώ παράλληλα διευκολύνεται η επιστημονική έρευνα.

Η εργασία των Divan et al. υπογραμμίζει ότι τα συστήματα παρακολούθησης που βασίζονται σε τεχνολογίες IoT αποτελούν πλέον ένα ισχυρό εργαλείο για την προστασία της δημόσιας υγείας και την κατανόηση των συνεπειών της ατμοσφαιρικής ρύπανσης. Τα συστήματα αυτά προσφέρουν δυνατότητες για άμεση ενημέρωση των πολιτών, πρόβλεψη κρίσιμων καταστάσεων και υιοθέτηση πολιτικών που στοχεύουν στη μείωση των επιπτώσεων της ρύπανσης.

Η συγκεκριμένη εργασία ενισχύει το σκεπτικό της παρούσας μελέτης σύμφωνα με το οποίο είναι απαραίτητη η ανάπτυξη ευέλικτων και οικονομικών λύσεων παρακολούθησης της ποιότητας του αέρα, με δυνατότητες ανάλυσης δεδομένων και ενεργής συμμετοχής των πολιτών. Το σύστημα που παρουσιάζεται στην παρούσα εργασία κινείται προς αυτή την κατεύθυνση, προσφέροντας τη δυνατότητα όχι μόνο παρακολούθησης της ποιότητας του αέρααλλά και αναγνώρισης των χρηστών μέσω των συσκευών τους, ενισχύοντας τη διαχείριση.

2.3 Σύστημα υψηλής ανάλυσης για τη συλλογή και παρακολούθηση αιωρούμενων σωματιδίων PM10 και PM2.5, (Ghizlane etc 2022)

Σύμφωνα με την εργασία των Ghizlane, Mabrouki, Ghrissi και Azrour (2022) [3], προτείνεται ένα προηγμένο σύστημα υψηλής ανάλυσης για τη συλλογή και παρακολούθηση αιωρούμενων σωματιδίων PM10 και PM2.5, το οποίο συγκρίνεται με υβριδικά συστήματα βασισμένα σε τεχνολογίες του Διαδικτύου των Πραγμάτων (IoT). Η μελέτη επικεντρώνεται σε μια ιδιαίτερα ρυπασμένη περιοχή, τα λατομεία της δυτικής Rif στο Μαρόκο, όπου τα επίπεδα αιωρούμενων σωματιδίων είναι εξαιρετικά υψηλά λόγω της έντονης ανθρώπινης δραστηριότητας και των εξορυκτικών εργασιών.

Οι ερευνητές αναγνωρίζουν ότι η παρακολούθηση της ποιότητας του αέρα σε τέτοιες περιοχές απαιτεί συστήματα υψηλής ακρίβειας και πυκνής χωρικής καταγραφής για να παρέχεται μια ολοκληρωμένη εικόνα της κατανομής των ρύπων. Για αυτόν τον λόγο προτείνεται η ανάπτυξη ενός συστήματος που αξιοποιεί πολλαπλούς αισθητήρες χαμηλού κόστους οι οποίοι τοποθετούνται στρατηγικά σε διάφορα σημεία, προσφέροντας δεδομένα υψηλής ανάλυσης τόσο σε επίπεδο χρόνου όσο και χώρου.

Η εργασία τονίζει ότι τα παραδοσιακά σταθερά συστήματα παρακολούθησης, παρότι παρέχουν υψηλής ακρίβειας αποτελέσματα, δεν είναι κατάλληλα για εφαρμογή σε μεγάλες εκτάσεις με έντονη περιβαλλοντική επιβάρυνση, λόγω του υψηλού κόστους εγκατάστασης και συντήρησης. Η προτεινόμενη λύση βασίζεται σε ένα υβριδικό μοντέλο που συνδυάζει τη χρήση έξυπνων IoT αισθητήρων με κεντρικές μονάδες ανάλυσης δεδομένων.

Η πλατφόρμα που προτείνεται από τους ερευνητές επιτρέπει την εύκολη συγκέντρωση και ανάλυση δεδομένων σε πραγματικό χρόνο με ιδιαίτερη έμφαση στην αξιοποίηση αλγορίθμων για τη βελτίωση της ακρίβειας των μετρήσεων ακόμη και όταν χρησιμοποιούνται αισθητήρες χαμηλού κόστους. Εφαρμόζονται μέθοδοι βαθμονόμησης και διόρθωσης σφαλμάτων ώστε τα δεδομένα να είναι αξιόπιστα και συγκρίσιμα με εκείνα των επαγγελματικών συστημάτων.

2.4 Εθνικό Δίκτυο Παρακολούθησης Ατμοσφαιρικής Ρύπανσης

Η ιστοσελίδα του Υπουργείου Περιβάλλοντος και Ενέργειας (ΥΠΕΝ) παρέχει πρόσβαση σε δεδομένα μετρήσεων ατμοσφαιρικής ρύπανσης που συλλέγονται από το Εθνικό Δίκτυο Παρακολούθησης Ατμοσφαιρικής Ρύπανσης (ΕΔΠΑΡ). Οι μετρήσεις πραγματοποιούνται σε συνεχή βάση καθ' όλη τη διάρκεια του 24ώρου, με χρόνο απόκρισης των αυτομάτων αναλυτών της τάξης του ενός λεπτού [4].

Εικόνα 2.1: Δεδομένα Μετρήσεων Ατμοσφαιρικής Ρύπανσης [4]

Ωστόσο, υπάρχουν ορισμένα ζητήματα που περιορίζουν την αποτελεσματικότητα του συστήματος: οι συστάσεις προς το κοινό για υπερβάσεις ρύπων εκδίδονται την επόμενη ημέρα, μειώνοντας την άμεση χρηστικότητά τους· δεν παρέχεται ανάλυση της διακύμανσης των ρύπων σε μηνιαία ή ετήσια βάση, καθιστώντας δύσκολη την παρακολούθηση των τάσεων· και παρόλο που υπάρχουν σταθμοί μέτρησης από διάφορους φορείς (π.χ. δήμοι, Εγνατία Οδός), δεν είναι σαφές αν τα δεδομένα αυτά ενσωματώνονται σε μια ενιαία βάση για πληρέστερη παρακολούθηση.

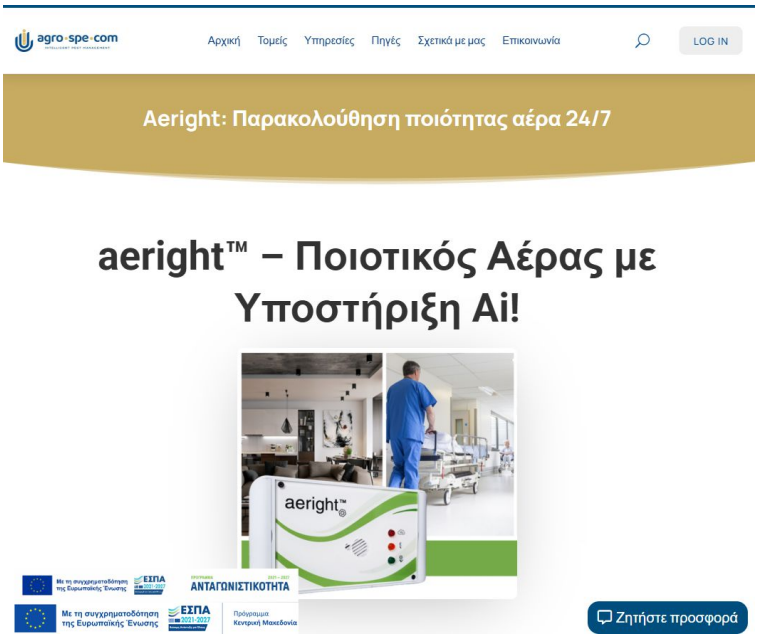
Επιπλέον, το Εργαστήριο Μηχανικής της Αειφορίας του ΑΠΘ έχει αναπτύξει ένα καινοτόμο Σύστημα Διαχείρισης Ποιότητας Αέρα, το οποίο παρέχει προβλέψεις για τις συνθήκες ποιότητας αέρα στη Θεσσαλονίκη. Παρά την επιτυχία του συστήματος, δεν έχει ενσωματωθεί πλήρως στις υποδομές της Περιφέρειας Κεντρικής Μακεδονίας, περιορίζοντας την αξιοποίησή του.

Το ΥΠΕΝ διαθέτει ένα βασικό σύστημα παρακολούθησης της ατμοσφαιρικής ρύπανσης και υπάρχουν περιθώρια βελτίωσης, όπως η έγκαιρη ενημέρωση του κοινού, η ανάλυση των δεδομένων σε βάθος χρόνου και η ενσωμάτωση τοπικών δεδομένων για μια ολοκληρωμένη εικόνα της ποιότητας του αέρα.

2.5 Σύστημα υψηλής ανάλυσης για τη συλλογή και παρακολούθηση αιωρούμενων σωματιδίων PM10 και PM2.5, (Ghizlane etc 2022)

Το Aeright™ της Agrospecom είναι ένα σύστημα παρακολούθησης της ποιότητας του εσωτερικού αέρα, σχεδιασμένο για συνεχή λειτουργία 24/7. Στοχεύει στη διασφάλιση υγιεινών συνθηκών σε χώρους με αυξημένη ανθρώπινη παρουσία, όπως γραφεία, σχολεία, ιατρικά κέντρα και κατοικίες [5].

Η συσκευή του Aeright™ μετρά σε πραγματικό χρόνο βασικές παραμέτρους όπως η θερμοκρασία, η υγρασία και η συγκέντρωση διοξειδίου του άνθρακα (CO_2). Όταν ανιχνεύεται υποβάθμιση της ποιότητας του αέρα, το σύστημα ειδοποιεί τους παρευρισκόμενους μέσω ενός οπτικού σήματος τύπου "φανάρι" με χρώματα πράσινο, πορτοκαλί και κόκκινο, υποδεικνύοντας το επίπεδο ποιότητας του αέρα. Αυτό επιτρέπει την άμεση λήψη διορθωτικών μέτρων, όπως ο αερισμός του χώρου ή η ενεργοποίηση συστημάτων καθαρισμού αέρα.



Εικόνα 2.2: Aeright [5]

Το Aeright™ συνδέεται ασύρματα με την πλατφόρμα cloud της Centaur Analytics, επιτρέποντας τη συλλογή και ανάλυση δεδομένων. Χρησιμοποιώντας τεχνητή νοημοσύνη, το σύστημα μπορεί να προβλέψει την ποιότητα του αέρα σε ολόκληρο τον χώρο με τη χρήση ενός μόνο αισθητήρα. Επιπλέον, παρέχεται η δυνατότητα πλήρως αυτοματοποιημένου ελέγχου των συσκευών κλιματισμού,

συμβάλλοντας στη διατήρηση υγιεινών συνθηκών χωρίς την ανάγκη συνεχούς ανθρώπινης παρέμβασης.

Η μέτρηση του CO_2 είναι ιδιαίτερα σημαντική, καθώς αποτελεί δείκτη της ποιότητας του αέρα σε κλειστούς χώρους. Υψηλά επίπεδα CO_2 μπορεί να υποδηλώνουν ανεπαρκή αερισμό, αυξάνοντας τον κίνδυνο μετάδοσης αερομεταφερόμενων παθογόνων, όπως ο SARS-CoV-2. Το Aeright™ είναι προγραμματισμένο σύμφωνα με ευρωπαϊκά και αμερικανικά πρότυπα ποιότητας αέρα, προσφέροντας αξιόπιστες μετρήσεις και ειδοποιήσεις.

Το σύστημα είναι κατάλληλο για χρήση σε ποικίλους χώρους, όπως νοσοκομεία, σχολεία, γραφεία, καταστήματα, ξενοδοχεία, γυμναστήρια, κομμωτήρια, εκκλησίες και κατοικίες, τόσο στην Ελλάδα όσο και στο εξωτερικό.

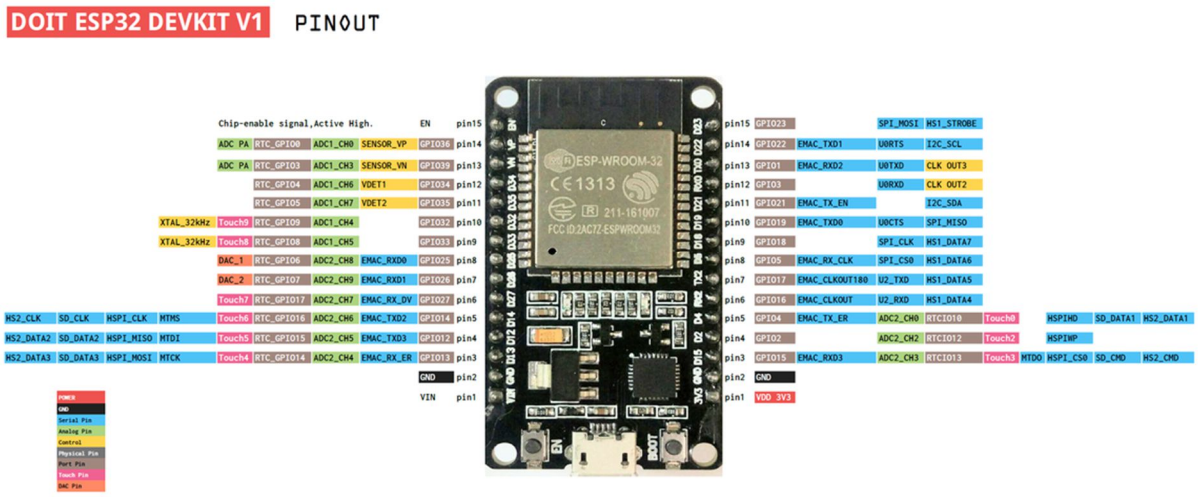
Κεφάλαιο 3ο: Χρήση τεχνολογικών εργαλείων στο έργο

3.1 esp32

Το ESP32 είναι ένα ισχυρό και ευέλικτο μικροελεγκτή σύστημα-σε-τσιπ (SoC) που έχει σχεδιαστεί από την Espressif Systems. Ενσωματώνει δυνατότητες Wi-Fi και Bluetooth και το κάνει ιδανικό για εφαρμογές Internet of Things (IoT), αυτοματισμούς και φορητές συσκευές [6-8].

Τεχνικά Χαρακτηριστικά

- **Επεξεργαστής**: Διπύρηνος ή μονοπύρηνος Xtensa LX6 32-bit, με συχνότητα έως 240 MHz.
- **Μνήμη**: 520 KiB SRAM και 448 KiB ROM.
- **Συνδεσιμότητα**: Υποστήριξη Wi-Fi 802.11 b/g/n και Bluetooth v4.2 BR/EDR και BLE.
- **GPIOs**: Έως 34 προγραμματιζόμενες εισόδους/εξόδους.
- **Αναλογικές Είσοδοι/Εξοδοι**: 2 × 12-bit SAR ADCs (έως 18 κανάλια), 2 × 8-bit DACs.
- **Άλλες Διεπαφές**: SPI, I²S, I²C, UART, SD/SDIO/MMC, Ethernet MAC, CAN bus 2.0.
- **Αισθητήρες**: Ενσωματωμένοι αισθητήρες αφής (10 κανάλια), αισθητήρας Hall.
- **Λειτουργίες Εξοικονόμησης Ενέργειας**: Υποστήριξη "deep sleep" με κατανάλωση ρεύματος μόλις 5 μA.



Εικόνα 3.1: esp32 ακροδέκτες

[https://www.cableworks.gr/images/detailed/27/pinout_esp32.png]

Η οικογένεια ESP32 περιλαμβάνει διάφορες παραλλαγές, όπως:

- ESP32-WROOM-32: Η πιο διαδεδομένη έκδοση με 4 MB flash μνήμη.
- ESP32-WROVER: Περιλαμβάνει επιπλέον 4 ή 8 MB PSRAM, κατάλληλο για εφαρμογές με αυξημένες απαιτήσεις μνήμης.
- ESP32-S2: Μονοπύρηνη έκδοση με βελτιωμένη ασφάλεια και υποστήριξη USB OTG.
- ESP32-CAM: Ενσωματώνει κάμερα OV2640, ιδανική για εφαρμογές παρακολούθησης.
- ESP32-PICO-D4: Μικρού μεγέθους module με ενσωματωμένα όλα τα απαραίτητα εξαρτήματα για εξοικονόμηση χώρου.

Το ESP32 χρησιμοποιείται ευρέως σε ποικίλες εφαρμογές, όπως:

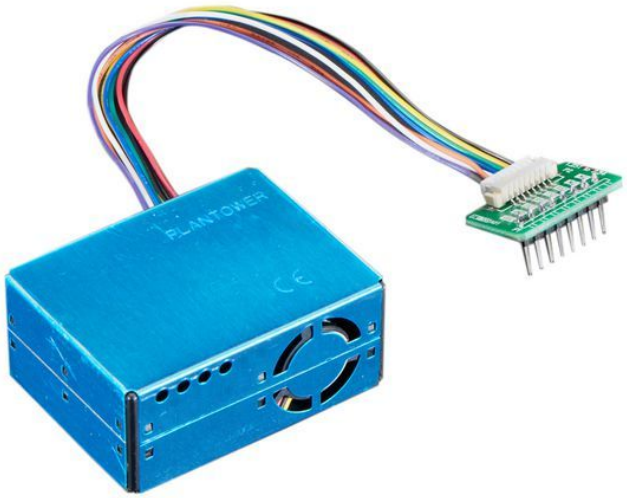
- Αυτοματισμοί Κατοικιών: Έλεγχος φωτισμού, θερμοκρασίας, ρολών και άλλων συσκευών μέσω Wi-Fi ή Bluetooth.
- Φορητές Συσκευές: Καταγραφή δεδομένων από αισθητήρες, όπως θερμοκρασία, υγρασία, ποιότητα αέρα.
- Βιομηχανικές Εφαρμογές: Παρακολούθηση και έλεγχος βιομηχανικών διαδικασιών με απομακρυσμένη πρόσβαση.
- Ασφάλεια: Συστήματα παρακολούθησης, ανίχνευση κίνησης, αναγνώριση προσώπου.
- Αναγνώριση Φωνής και Ήχου: Εφαρμογές με επεξεργασία ήχου, όπως έξυπνοι βοηθοί.

Το ESP32 υποστηρίζει διάφορα περιβάλλοντα ανάπτυξης, όπως:

- ESP-IDF: Το επίσημο framework της Espressif για ανάπτυξη εφαρμογών σε C/C++.
- Arduino IDE: Δημοφιλές περιβάλλον για αρχάριους και προχωρημένους χρήστες.
- MicroPython: Ελαφριά υλοποίηση της Python 3 για μικροελεγκτές.
- PlatformIO: Επαγγελματικό περιβάλλον ανάπτυξης με υποστήριξη πολλών πλατφορμών.

3.2 pms5003

Ο PMS5003 είναι ένας ψηφιακός αισθητήρας μέτρησης συγκέντρωσης σωματιδίων στον αέρα, σχεδιασμένος από την Plantower. Χρησιμοποιεί την αρχή της σκέδασης λέιζερ για την ανίχνευση και μέτρηση σωματιδίων διαμέτρου από 0,3 έως 10 μικρόμετρα και παρέχει αξιόπιστα δεδομένα σε πραγματικό χρόνο για την ποιότητα του αέρα [9-10].



Εικόνα 3.2: PMS5003

[https://media.distrelec.com/Web/WebShopImages/landscape_large/3-/01/Adafruit-3686-30139173-01.jpg]

Τεχνικά Χαρακτηριστικά

- Εύρος μέτρησης: 0,3–1,0 μm, 1,0–2,5 μm, 2,5–10 μm
- Αποδοτικότητα μέτρησης: 50% για σωματίδια 0,3 μm, 98% για ≥0,5 μm
- Ακρίβεια: ±10% για συγκεντρώσεις 100–500 μg/m³, ±10 μg/m³ για 0–100 μg/m³
- Ανάλυση: 1 μg/m³
- Χρόνος απόκρισης: <1 δευτερόλεπτο (με πλήρη σταθεροποίηση σε ≤10 δευτερόλεπτα)
- Τροφοδοσία: 5V DC (εύρος 4,5–5,5V)
- Κατανάλωση ρεύματος: ≤100 mA σε λειτουργία, ≤200 μA σε αναμονή
- Διαστάσεις: 50 × 38 × 21 mm
- Θερμοκρασία λειτουργίας: -10°C έως +60°C
- Υγρασία λειτουργίας: 0–99%

Ο αισθητήρας λειτουργεί με τη μέθοδο της σκέδασης λέιζερ: ένας ενσωματωμένος ανεμιστήρας διοχετεύει τον αέρα μέσω του αισθητήρα, όπου ένα λέιζερ φωτίζει τα αιωρούμενα σωματίδια. Το φως που σκεδάζεται συλλέγεται και αναλύεται από έναν μικροεπεξεργαστή ο οποίος υπολογίζει τη συγκέντρωση και το μέγεθος των σωματιδίων.

Η επικοινωνία με μικροελεγκτές, όπως το ESP32, πραγματοποιείται μέσω σειριακής διεπαφής UART στα 9600 bps. Η έξοδος του αισθητήρα περιλαμβάνει:

- Συγκεντρώσεις PM1.0, PM2.5 και PM10 σε μικρογραμμάρια ανά κυβικό μέτρο (μg/m³).
- Αριθμό σωματιδίων ανά 0,1 λίτρο αέρα για μεγέθη >0,3 μm, >0,5 μm, >1,0 μm, >2,5 μm, >5,0 μm και >10 μm.

Ο αισθητήρας μπορεί να λειτουργεί σε ενεργή ή παθητική λειτουργία:

- Ενεργή λειτουργία: Αποστέλλει δεδομένα αυτόματα σε καθορισμένα διαστήματα.
- Παθητική λειτουργία: Αποστέλλει δεδομένα μόνο κατόπιν αιτήματος από τον κεντρικό ελεγκτή.

Ο PMS5003 διαθέτει 8 ακίδες, με τις βασικές να είναι:

- VCC: Τροφοδοσία 5V
- GND: Γείωση

- TX: Μετάδοση δεδομένων (3,3V λογικό επίπεδο)
- RX: Λήψη δεδομένων (3,3V λογικό επίπεδο)
- SET: Επιλογή λειτουργίας (υψηλό ή αιωρούμενο για κανονική λειτουργία, χαμηλό για λειτουργία ύπνου)
- RESET: Επαναφορά αισθητήρα (χαμηλό για επανεκκίνηση)

Αν και ο αισθητήρας λειτουργεί με τροφοδοσία 5V, οι λογικές στάθμες για την επικοινωνία είναι 3,3V, καθιστώντας τον συμβατό με μικροελεγκτές όπως το ESP32.

Ο PMS5003 είναι ιδανικός για:

- Παρακολούθηση ποιότητας αέρα: Σε εσωτερικούς και εξωτερικούς χώρους, όπως κατοικίες, γραφεία και σχολεία.
- Συστήματα αυτοματισμού: Ενεργοποίηση καθαριστών αέρα ή συστημάτων εξαερισμού βάσει των μετρήσεων.
- Έργα IoT: Συλλογή και ανάλυση δεδομένων ποιότητας αέρα μέσω πλατφορμών cloud.

3.3 Python

Η γλώσσα προγραμματισμού Python αποτελεί ένα από τα πιο διαδεδομένα εργαλεία ανάπτυξης λογισμικού στη σύγχρονη εποχή. Δημιουργήθηκε από τον Guido van Rossum και παρουσιάστηκε για πρώτη φορά το 1991. Το όνομά της δεν προέρχεται από το φίδι «πύθωνας», όπως πολλοί νομίζουν, αλλά από την κωμική σειρά «Monty Python's Flying Circus», την οποία εκτιμούσε ιδιαίτερα ο δημιουργός της. Η Python σχεδιάστηκε με στόχο να είναι απλή, ευανάγνωστη και ταυτόχρονα ισχυρή, γεγονός που την καθιστά ιδανική τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές.

Η απλότητα της σύνταξής της είναι ένα από τα σημαντικότερα πλεονεκτήματά της. Ένας απλός υπολογισμός ή ακόμα και μια σύνθετη λειτουργία μπορούν να υλοποιηθούν με λιγότερες γραμμές κώδικα σε σύγκριση με άλλες γλώσσες, όπως η C++ ή η Java. Διαθέτει μια τεράστια γκάμα βιβλιοθηκών και εργαλείων που καλύπτουν σχεδόν κάθε τομέα της πληροφορικής, από την ανάλυση δεδομένων και την τεχνητή νοημοσύνη έως την ανάπτυξη ιστοσελίδων και τις ενσωματωμένες συσκευές [11-15].

Η Python είναι μια γλώσσα υψηλού επιπέδου. Αυτό σημαίνει ότι ο κώδικας δεν χρειάζεται να μεταγλωττιστεί πριν εκτελεστεί, αλλά διαβάζεται και εκτελείται απευθείας από τον διερμηνέα της

γλώσσας. Έτσι, διευκολύνεται η ανάπτυξη και ο έλεγχος εφαρμογών, καθώς οποιοδήποτε λάθος εντοπίζεται γρήγορα κατά την εκτέλεση. Παράλληλα, υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, όπως τον αντικειμενοστραφή, τον διαδικαστικό και τον λειτουργικό προγραμματισμό, προσφέροντας ευελιξία στην υλοποίηση διαφόρων ειδών έργων.

Η κοινότητα της Python είναι μια από τις μεγαλύτερες και πιο ενεργές στον κόσμο της πληροφορικής. Αυτό σημαίνει πως υπάρχουν αμέτρητες πηγές εκμάθησης, έτοιμα παραδείγματα κώδικα, βιβλιοθήκες και frameworks που διευκολύνουν σημαντικά την ανάπτυξη λογισμικού. Ενδεικτικά, στον τομέα της ανάλυσης δεδομένων χρησιμοποιούνται βιβλιοθήκες όπως τα Pandas και Numpy, στην τεχνητή νοημοσύνη το TensorFlow και το PyTorch, ενώ για την ανάπτυξη ιστοσελίδων είναι ιδιαίτερα γνωστά τα Flask και Django.

Παρά τα σημαντικά πλεονεκτήματά της, η Python έχει και μειονεκτήματα. Ένα από τα κύρια είναι η ταχύτητα εκτέλεσης, η οποία μπορεί να είναι χαμηλότερη σε σχέση με άλλες γλώσσες όπως η C ή η Java, ειδικά σε εφαρμογές που απαιτούν επεξεργασία μεγάλου όγκου δεδομένων σε πραγματικό χρόνο. Ωστόσο, η ευκολία ανάπτυξης και η δυνατότητα ενσωμάτωσης κώδικα άλλων γλωσσών, όπως C ή C++, με τη χρήση κατάλληλων διεπαφών (APIs), αντισταθμίζουν συχνά αυτό το μειονέκτημα.

3.4 Django Framework

Το Django είναι ένα από τα πιο ισχυρά και δημοφιλή web frameworks που έχουν δημιουργηθεί για τη γλώσσα προγραμματισμού Python. Δημιουργήθηκε το 2005 από μια ομάδα προγραμματιστών που εργαζόταν σε διαδικτυακά ειδησεογραφικά portals, με σκοπό να διευκολύνει την ταχεία ανάπτυξη ασφαλών και συντηρήσιμων web εφαρμογών. Από τότε, το Django έχει εξελιχθεί σε ένα από τα πλέον καταξιωμένα εργαλεία στον χώρο της ανάπτυξης ισχυρών και επεκτάσιμων ιστοσελίδων και web εφαρμογών [16-21].

Το Django βασίζεται στη φιλοσοφία DRY (Don’t Repeat Yourself) και στο αρχιτεκτονικό πρότυπο MVC (Model-View-Controller), το οποίο στο Django προσαρμόζεται ως MTV (Model-Template-View).

- **Model**: Καθορίζει τη δομή των δεδομένων και τις σχέσεις τους (αντίστοιχο με τον πίνακα της βάσης δεδομένων).
- **Template**: Ασχολείται με το presentation layer, δηλαδή πώς εμφανίζονται τα δεδομένα στον τελικό χρήστη.

- **View**: Περιέχει τη λογική της εφαρμογής και χειρίζεται τα αιτήματα που δέχεται από τον χρήστη.

Με αυτή την αρχιτεκτονική, ο κώδικας παραμένει καθαρός, καλά οργανωμένος και ευκολότερα επεκτάσιμος.

Δυνατότητες του Django

Ενσωματωμένο Admin Panel

Το Django διαθέτει ένα πλήρως λειτουργικό περιβάλλον διαχείρισης (Admin Interface) που δημιουργείται αυτόματα με βάση τα μοντέλα που ορίζει ο προγραμματιστής. Αυτό επιτρέπει την εύκολη διαχείριση περιεχομένου, χρηστών και δεδομένων χωρίς επιπλέον κώδικα.

Ασφάλεια από Σχεδίαση

Προστατεύει τις εφαρμογές από κοινές απειλές όπως:

- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Clickjacking

  Παρέχει επίσης ισχυρούς μηχανισμούς για την ασφαλή διαχείριση κωδικών και συνεδριών χρηστών.

ORM (Object-Relational Mapping)

Ο μηχανισμός ORM επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων μέσω Python αντικειμένων, χωρίς την ανάγκη γραφής καθαρών SQL εντολών. Αυτό καθιστά την ανάπτυξη γρηγορότερη και πιο ασφαλή.

URL Routing

Το Django προσφέρει έναν ευέλικτο μηχανισμό καθορισμού των URL routes, διευκολύνοντας τη δημιουργία φιλικών προς τον χρήστη URLs και την καθαρή δόμηση των εφαρμογών.

Υποστήριξη για REST APIs

Με την επέκταση Django REST Framework (DRF), οι προγραμματιστές μπορούν να δημιουργήσουν ισχυρά και ασφαλή APIs για την υποστήριξη mobile εφαρμογών και άλλων υπηρεσιών.

Διαχείριση Χρηστών και Δικαιωμάτων

Παρέχεται πλήρες σύστημα διαχείρισης χρηστών, ομάδων και δικαιωμάτων. Μπορεί εύκολα να επεκταθεί με custom models και να ενσωματωθεί με μηχανισμούς πιστοποίησης όπως OAuth2.

Εφαρμογή Django: Σύστημα Καταγραφής Περιβαλλοντικών Μετρήσεων

Βήμα 1: Δημιουργία του Project

```
django-admin startproject environment_monitor
cd environment_monitor
python manage.py startapp sensors
```

Ορισμός του Model

```
# sensors/models.py
from django.db import models

class Measurement(models.Model):
    device_id = models.CharField(max_length=50)
    pm25 = models.FloatField()
    pm10 = models.FloatField()
    temperature = models.FloatField()
    humidity = models.FloatField()
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
```

```
        return f"Device {self.device_id} at {self.timestamp}"
```

Δημιουργία και Εφαρμογή Migration

```
python manage.py makemigrations
python manage.py migrate
```

Προσθήκη της Εφαρμογής στο settings.py

```
INSTALLED_APPS = [
    ...,
    'sensors',
]
```

Δημιουργία View για Λήψη και Εμφάνιση Δεδομένων

```
# sensors/views.py
from django.shortcuts import render
from django.http import JsonResponse
from .models import Measurement
import json

def receive_measurement(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        Measurement.objects.create(
            device_id=data['device_id'],
            pm25=data['pm25'],
            pm10=data['pm10'],
            temperature=data['temperature'],
            humidity=data['humidity']
```

```
        )
        return JsonResponse({'status': 'success'})
    return JsonResponse({'error': 'Invalid request'}, status=400)
```

Ορισμός των URLs

```
# sensors/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('api/measurements/', views.receive_measurement, name='receive_measurement'),
]
```

Γενικά το Django αποτελεί ένα ολοκληρωμένο εργαλείο που καλύπτει πλήρως τις ανάγκες ανάπτυξης μιας σύγχρονης web εφαρμογής, από τη διαχείριση χρηστών και την αλληλεπίδραση με βάσεις δεδομένων, μέχρι την ασφαλή υλοποίηση RESTful APIs.

Η επιλογή του Django στο έργο μας επιτρέπει:

- Ταχεία ανάπτυξη αξιόπιστων και ασφαλών εφαρμογών.
- Οργανωμένη διαχείριση του κώδικα και των δεδομένων.
- Διευκόλυνση της συντήρησης και της μελλοντικής επέκτασης του συστήματος.

3.5 Google Charts

Η βιβλιοθήκη Google Charts αποτελεί μια από τις πιο διαδεδομένες λύσεις για την οπτικοποίηση δεδομένων στο διαδίκτυο. Παρέχεται δωρεάν από την Google και προσφέρει έναν ισχυρό μηχανισμό δημιουργίας γραφημάτων απευθείας μέσα από τον browser, χωρίς την ανάγκη εγκατάστασης πρόσθετων εργαλείων ή εφαρμογών. Η δημοτικότητά της οφείλεται τόσο στην ευκολία ενσωμάτωσής της σε οποιαδήποτε ιστοσελίδα, όσο και στη μεγάλη ποικιλία τύπων γραφημάτων που υποστηρίζει. Η

βιβλιοθήκη επιτρέπει την απεικόνιση δεδομένων με γραφήματα γραμμής, ράβδων, πίτας, περιοχής, χάρτες, οργανωτικά διαγράμματα και πολλά άλλα, καλύπτοντας έτσι κάθε πιθανή ανάγκη παρουσίασης [22-23].

Ένα από τα βασικά πλεονεκτήματα της Google Charts είναι η δυνατότητα δημιουργίας διαδραστικών γραφημάτων, τα οποία ανταποκρίνονται άμεσα σε ενέργειες του χρήστη όπως hover, click ή zoom. Αυτό επιτρέπει τη δημιουργία δυναμικών dashboards όπου τα δεδομένα δεν εμφανίζονται απλώς στατικά, αλλά προσαρμόζονται ανάλογα με τις επιλογές του χρήστη. Η βιβλιοθήκη υποστηρίζει επίσης real-time ενημέρωση δεδομένων, κάτι ιδιαίτερα χρήσιμο σε εφαρμογές παρακολούθησης μετρήσεων ή στατιστικών σε πραγματικό χρόνο.

Η ενσωμάτωση της Google Charts σε ένα έργο απαιτεί την προσθήκη του σχετικού script από τη Google και την κατάλληλη χρήση JavaScript για τη διαμόρφωση και την εμφάνιση των γραφημάτων. Παρότι η διαχείριση των δεδομένων γίνεται μέσω JavaScript, τα δεδομένα μπορούν εύκολα να αντλούνται από backend συστήματα, όπως εφαρμογές γραμμένες σε Python με Flask ή Django, μέσω AJAX κλήσεων και μορφοποίησης σε JSON. Αυτό επιτρέπει την απευθείας σύνδεση των γραφημάτων με βάσεις δεδομένων και τη συνεχή ενημέρωση των αποτελεσμάτων που προβάλλονται στον χρήστη.

Στο έργο μας, η Google Charts μπορεί να χρησιμοποιηθεί για την οπτικοποίηση μετρήσεων που συλλέγονται από τις συσκευές ESP32. Οι μετρήσεις αυτές, όπως οι συγκεντρώσεις σωματιδίων PM2.5 και PM10, η θερμοκρασία και η υγρασία, μπορούν να παρουσιαστούν με γραφήματα γραμμής που δείχνουν την πορεία των τιμών στο χρόνο. Με αυτόν τον τρόπο, ο χρήστης αποκτά άμεση και σαφή εικόνα των μεταβολών που συμβαίνουν στο περιβάλλον, ενώ παράλληλα διευκολύνεται η εξαγωγή συμπερασμάτων σχετικά με την ποιότητα του αέρα και τις συνθήκες που επικρατούν.

Η απλότητα στη χρήση της Google Charts σε συνδυασμό με τις μεγάλες δυνατότητες παραμετροποίησης την καθιστούν ιδανική λύση για την ανάπτυξη σύγχρονων και εντυπωσιακών διαδικτυακών εφαρμογών.

Η Google Charts αποτελεί ένα ιδιαίτερα καλό εργαλείο για την οπτικοποίηση δεδομένων σε web εφαρμογές. Η δυνατότητα άμεσης ενσωμάτωσης με backend σε Python και η υποστήριξη για real-time δεδομένα και η παραγωγή εντυπωσιακών πλήρως διαδραστικών γραφημάτων κάνουν την βιβλιοθήκη αυτή ένα σοβαρό κομμάτι κάθε σύγχρονου συστήματος πληροφόρησης και παρακολούθησης.

3.6 Πίνακες με tabulator.js

Το Tabulator.js είναι μια προηγμένη JavaScript βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία δυναμικών και διαδραστικών πινάκων δεδομένων σε ιστοσελίδες. Σε αντίθεση με παραδοσιακές λύσεις που απαιτούν πολύπλοκο χειρισμό του DOM και εκτενή χρήση JavaScript, το Tabulator προσφέρει μια

ολοκληρωμένη και εύχρηστη λύση για την εμφάνιση, διαχείριση και επεξεργασία δεδομένων σε μορφή πίνακα, με ενσωματωμένες λειτουργίες ταξινόμησης, φιλτραρίσματος, αναζήτησης, επεξεργασίας κελιών και εξαγωγής δεδομένων [24].

Ένα από τα μεγαλύτερα πλεονεκτήματα του Tabulator.js είναι η ευκολία προσθήκης του σε οποιοδήποτε έργο. Ο προγραμματιστής μπορεί να δημιουργήσει έναν πλήρως λειτουργικό πίνακα με λίγες μόνο γραμμές κώδικα, καθορίζοντας απλά τα δεδομένα και τις στήλες. Τα δεδομένα μπορούν να οριστούν απευθείας στον κώδικα ή να φορτώνονται δυναμικά από εξωτερικές πηγές όπως APIs ή βάσεις δεδομένων. Υποστηρίζεται η πλήρης παραμετροποίηση της εμφάνισης του πίνακα, ώστε να ενσωματώνεται αρμονικά στο γενικότερο design της ιστοσελίδας.

Το Tabulator παρέχει έτοιμες λειτουργίες για ταξινόμηση δεδομένων κατά στήλη, φιλτράρισμα με βάση συγκεκριμένα κριτήρια, αναζήτηση λέξεων κλειδιών και σελιδοποίηση, διευκολύνοντας έτσι την οργάνωση και προβολή μεγάλων συνόλων δεδομένων. Όλες αυτές οι λειτουργίες είναι ενσωματωμένες και δεν απαιτούν πρόσθετο προγραμματισμό, κάτι που μειώνει σημαντικά τον χρόνο ανάπτυξης. Προσφέρει δυνατότητες inline επεξεργασίας κελιών, επιτρέποντας στον χρήστη να τροποποιεί τα δεδομένα απευθείας μέσα στον πίνακα, χωρίς να απαιτείται φόρτωμα νέας σελίδας ή modal παραθύρων.

Η βιβλιοθήκη υποστηρίζει επίσης την εξαγωγή δεδομένων σε διάφορες μορφές όπως CSV, JSON, XLSX και PDF, παρέχοντας εύκολες λύσεις για την αποθήκευση ή τη μεταφορά των δεδομένων. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη σε εφαρμογές που απαιτούν την παραγωγή αναφορών ή την αποστολή δεδομένων σε άλλες πλατφόρμες. Υποστηρίζεται και η λειτουργία εκτύπωσης των πινάκων με ειδική μορφοποίηση που προσαρμόζεται για έντυπη αναπαραγωγή.

Στο έργο μας το Tabulator.js μπορεί να χρησιμοποιηθεί για την προβολή των μετρήσεων που συλλέγονται από τις συσκευές ESP32. Οι πίνακες μπορούν να εμφανίζουν δεδομένα όπως ημερομηνία και ώρα μέτρησης, συγκεντρώσεις PM2.5 και PM10, θερμοκρασία, υγρασία και άλλες περιβαλλοντικές παραμέτρους. Με την ενσωμάτωση δυνατοτήτων φιλτραρίσματος και αναζήτησης, οι χρήστες μπορούν να εντοπίζουν εύκολα τις πληροφορίες που τους ενδιαφέρουν, ενώ μέσω της εξαγωγής σε αρχεία μπορούν να κρατούν ιστορικά αρχεία ή να αναλύουν περαιτέρω τα δεδομένα σε άλλες εφαρμογές.

Η χρήση του Tabulator σε συνδυασμό με backend εφαρμογές όπως αυτές που υλοποιούνται με Django ή Flask, επιτρέπει την απευθείας φόρτωση των δεδομένων από APIs. Μέσω AJAX κλήσεων, τα δεδομένα μπορούν να ενημερώνονται σε πραγματικό χρόνο και να δίνουν στους χρήστες άμεση και ακριβή πληροφόρηση για τις συνθήκες που επικρατούν στο πεδίο των μετρήσεων. Αυτή η δυνατότητα καθιστά το Tabulator ιδανική επιλογή για συστήματα παρακολούθησης περιβαλλοντικών παραμέτρων, διαχείρισης χρηστών, παρουσίασης στατιστικών και άλλων εφαρμογών που απαιτούν προβολή μεγάλου όγκου πληροφοριών με φιλικό και αποδοτικό τρόπο.

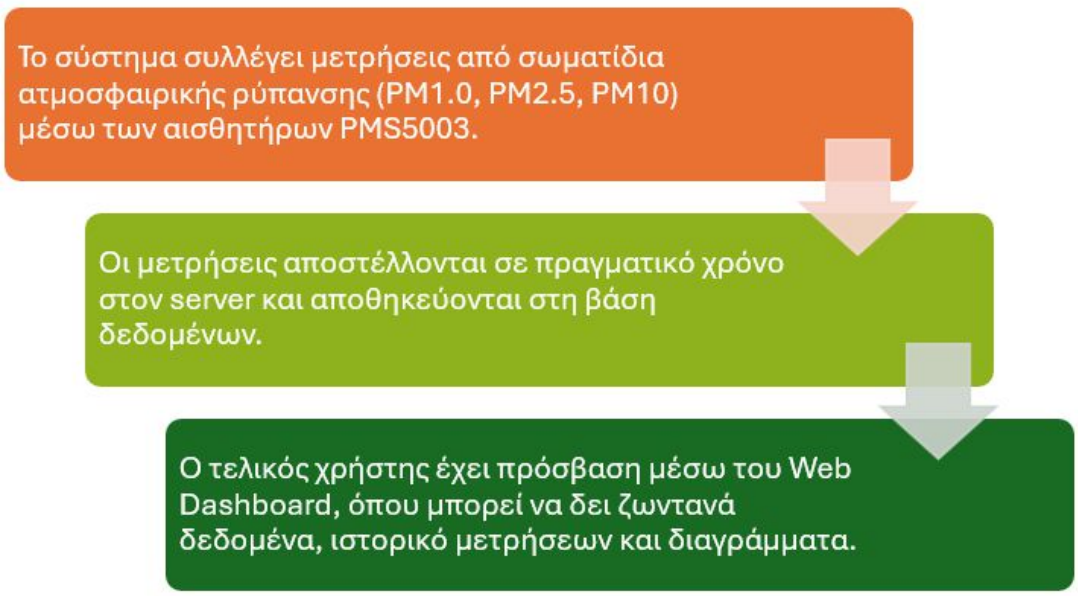
Κεφάλαιο 4ο: Η διαδικτυακή πλατφόρμα παρακολούθησης ποιότητας αέρα

4.1 Η λειτουργία του συστήματος

Η ενότητα αυτή παρουσιάζει αναλυτικά τη λειτουργία του συστήματος παρακολούθησης ποιότητας αέρα το οποίο σχεδιάστηκε με στόχο την άμεση καταγραφή, αποθήκευση και προβολή μετρήσεων σωματιδίων ατμοσφαιρικής ρύπανσης. Το σύστημα αξιοποιεί σύγχρονες τεχνολογίες τόσο στο υλικό (hardware) όσο και στο λογισμικό (software), εξασφαλίζοντας τη ροή δεδομένων από τους αισθητήρες μέτρησης μέχρι την τελική απεικόνισή τους στον χρήστη.

Συγκεκριμένα μέσω κατάλληλων αισθητήρων και του μικροελεγκτή ESP32, πραγματοποιούνται μετρήσεις των αιωρούμενων σωματιδίων PM1.0, PM2.5 και PM10 οι οποίες αποστέλλονται σε πραγματικό χρόνο σε έναν κεντρικό server. Μέσω ενός API που έχει αναπτυχθεί με το πλαίσιο Django, τα δεδομένα ελέγχονται, οργανώνονται και αποθηκεύονται σε σχεσιακή βάση δεδομένων MySQL.

Οι χρήστες έχουν τη δυνατότητα, μέσω ενός φιλικού και λειτουργικού διαδικτυακού περιβάλλοντος, να παρακολουθούν τις μετρήσεις των συσκευών τους, να διαχειρίζονται τις συσκευές που τους ανήκουν καθώς και να έχουν πρόσβαση σε ιστορικά δεδομένα και διαγράμματα. Με αυτόν τον τρόπο παρέχεται μια καλή λύση για την παρακολούθηση της ποιότητας του αέρα συμβάλλοντας στην έγκαιρη ενημέρωση των χρηστών.

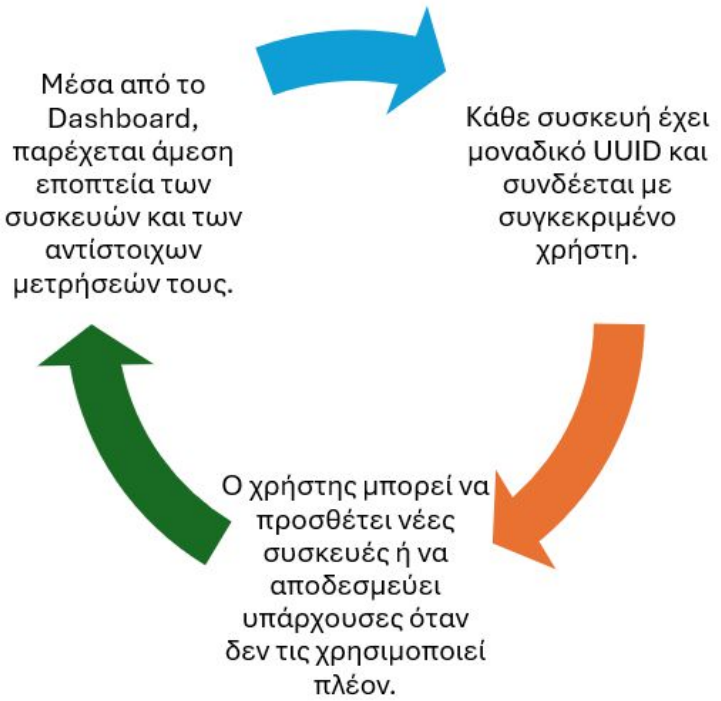


Εικόνα 4.1: Παρακολούθηση Ποιότητας Αέρα σε Πραγματικό Χρόνο

Στην Εικόνα 4.1 παρουσιάζεται με απλό και κατανοητό τρόπο η διαδικασία παρακολούθησης της ποιότητας του αέρα σε πραγματικό χρόνο, όπως υλοποιείται από το προτεινόμενο σύστημα. Αρχικά, το σύστημα συλλέγει μετρήσεις μέσω ειδικών αισθητήρων PMS5003, οι οποίοι καταγράφουν την παρουσία αιωρούμενων σωματιδίων στον αέρα, διαχωρισμένα σε τρεις βασικές κατηγορίες: PM1.0, PM2.5 και PM10. Αυτά τα σωματίδια σχετίζονται άμεσα με την ποιότητα του αέρα και έχουν σημαντικές επιπτώσεις στην ανθρώπινη υγεία, ειδικά όταν οι συγκεντρώσεις τους ξεπερνούν τα αποδεκτά όρια.

Στη συνέχεια οι μετρήσεις αποστέλλονται σε πραγματικό χρόνο στον κεντρικό server, όπου αποθηκεύονται οργανωμένα στη βάση δεδομένων. Με αυτόν τον τρόπο διασφαλίζεται ότι όλα τα δεδομένα είναι άμεσα διαθέσιμα για περαιτέρω επεξεργασία και ανάλυση και διατηρείται το ιστορικό των μετρήσεων για κάθε συσκευή.

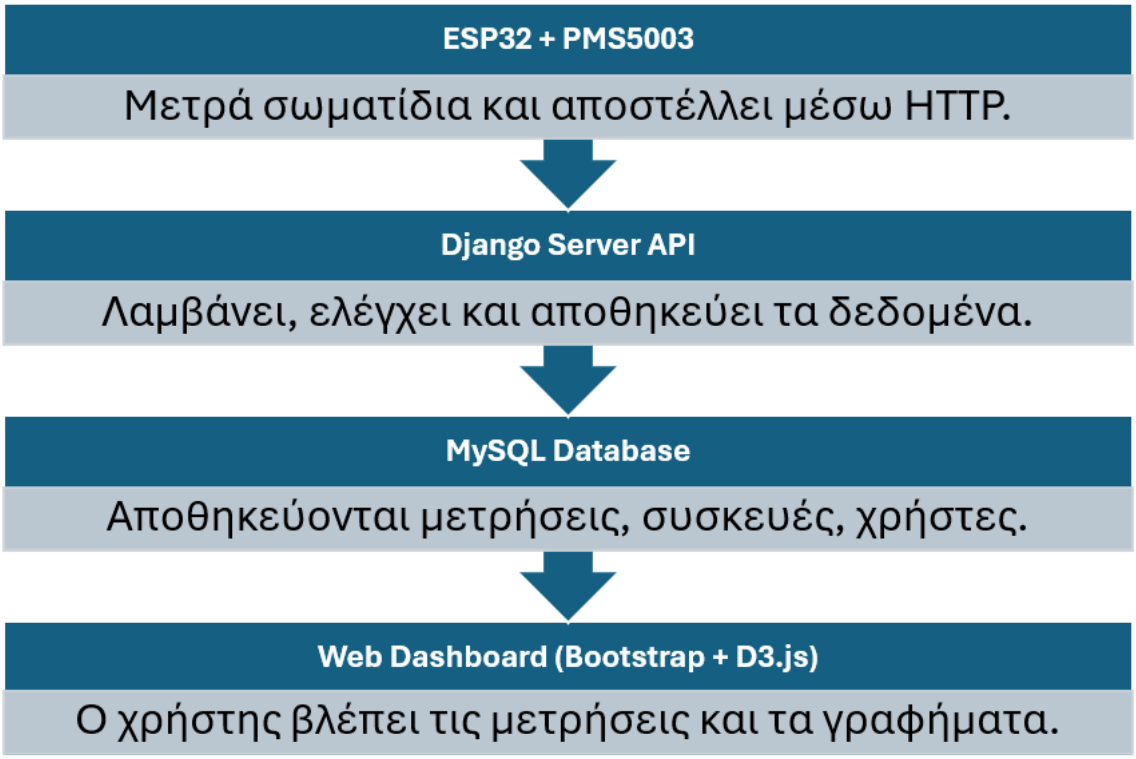
Ο τελικός χρήστης έχει τη δυνατότητα να συνδεθεί στην ειδικά διαμορφωμένη διαδικτυακή πλατφόρμα (Web Dashboard) και να έχει πλήρη εικόνα της ποιότητας του αέρα. Μέσα από την πλατφόρμα μπορεί να παρακολουθεί ζωντανά τις νέες μετρήσεις, να ανατρέχει σε παλαιότερα δεδομένα και να μελετά διαγράμματα που αποτυπώνουν την πορεία των μετρήσεων. Η δυνατότητα αυτή παρέχει στον χρήστη πολύτιμες πληροφορίες για το περιβάλλον του, συμβάλλοντας στη λήψη καλύτερων αποφάσεων για την προστασία της υγείας του και τη βελτίωση των συνθηκών διαβίωσης.



Εικόνα 4.2: Διαχείριση και Κατοχύρωση Συσκευών

Στην Εικόνα 4.2 παρουσιάζεται ο κύκλος διαχείρισης και κατοχύρωσης συσκευών στην πλατφόρμα παρακολούθησης ποιότητας αέρα. Κάθε συσκευή διαθέτει ένα μοναδικό αναγνωριστικό (UUID), το οποίο χρησιμοποιείται για να συνδεθεί με έναν συγκεκριμένο χρήστη. Μέσω αυτής της διαδικασίας διασφαλίζεται ότι κάθε χρήστης μπορεί να παρακολουθεί και να διαχειρίζεται μόνο τις συσκευές που του ανήκουν.

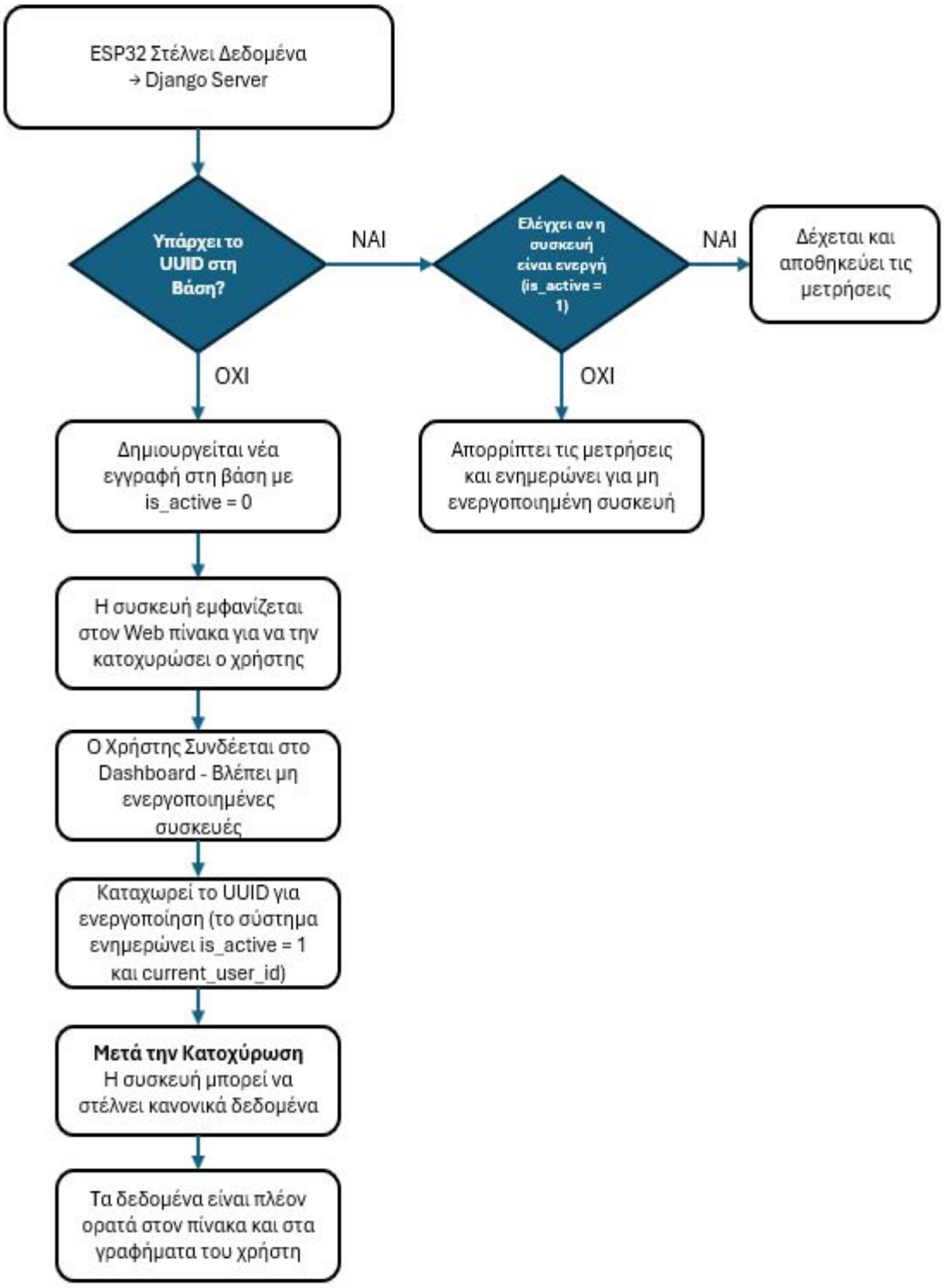
Ο χρήστης έχει τη δυνατότητα να προσθέτει νέες συσκευές στην πλατφόρμα καταχωρώντας το UUID τους, αλλά και να αποδεσμεύει συσκευές που δεν χρησιμοποιεί πλέον, ώστε αυτές να μπορούν να κατοχυρωθούν από άλλους χρήστες. Όλη η διαδικασία γίνεται με απλό και άμεσο τρόπο μέσα από το Web Dashboard, το οποίο προσφέρει πλήρη εποπτεία των συσκευών και των αντίστοιχων μετρήσεών τους. Αυτός ο κύκλος διαχείρισης διευκολύνει την αποτελεσματική παρακολούθηση της ποιότητας του αέρα και την ευελιξία στη χρήση των συσκευών από διαφορετικούς χρήστες.



Εικόνα 4.3: Διάγραμμα Ροής Δεδομένων

Στην Εικόνα 4.3 παρουσιάζεται το διάγραμμα ροής δεδομένων του συστήματος παρακολούθησης ποιότητας αέρα, το οποίο απεικονίζει τη συνολική διαδικασία συλλογής, αποστολής, αποθήκευσης και προβολής των μετρήσεων. Η διαδικασία ξεκινά με τη συσκευή ESP32 που, σε συνδυασμό με τον

αισθητήρα PMS5003, πραγματοποιεί μετρήσεις των αιωρούμενων σωματιδίων PM1.0, PM2.5 και PM10 και αποστέλλει τα δεδομένα μέσω πρωτοκόλλου HTTP.



Εικόνα 4.4: Διάγραμμα Διαδικασίας Εγγραφής & Αναγνώρισης Κόμβου

Στη συνέχεια, τα δεδομένα λαμβάνονται από τον Django Server API, όπου πραγματοποιείται έλεγχος εγκυρότητας και αποθήκευση των μετρήσεων. Τα δεδομένα οργανώνονται και φυλάσσονται στη MySQL Database, η οποία περιλαμβάνει πληροφορίες για τις μετρήσεις, τις συσκευές και τους χρήστες του συστήματος.

Ο χρήστης έχει τη δυνατότητα να δει όλα τα δεδομένα μέσω του Web Dashboard, το οποίο έχει υλοποιηθεί με χρήση των τεχνολογιών Bootstrap και D3.js. Μέσα από το γραφικό περιβάλλον, παρέχεται πλήρης εποπτεία των μετρήσεων με δυνατότητα παρουσίασης ζωντανών δεδομένων, ιστορικού και διαγραμμάτων, προσφέροντας μία ολοκληρωμένη εικόνα για την ποιότητα του αέρα.

Στην Εικόνα 4.4 παρουσιάζεται αναλυτικά η διαδικασία εγγραφής και αναγνώρισης ενός κόμβου (συσκευής) στο σύστημα παρακολούθησης ποιότητας αέρα. Η διαδικασία ξεκινά όταν η συσκευή ESP32 αποστέλλει δεδομένα στον Django Server. Ο server ελέγχει εάν το UUID της συσκευής υπάρχει ήδη στη βάση δεδομένων.

Εάν το UUID δεν υπάρχει, δημιουργείται νέα εγγραφή με την τιμή is_active = 0 και η συσκευή εμφανίζεται στον Web πίνακα ως μη ενεργοποιημένη, ώστε να την κατοχυρώσει ο χρήστης.

Αν το UUID υπάρχει, ο server ελέγχει αν η συσκευή είναι ενεργή (is_active = 1).

Εάν είναι ενεργή, τα δεδομένα γίνονται αποδεκτά και αποθηκεύονται.

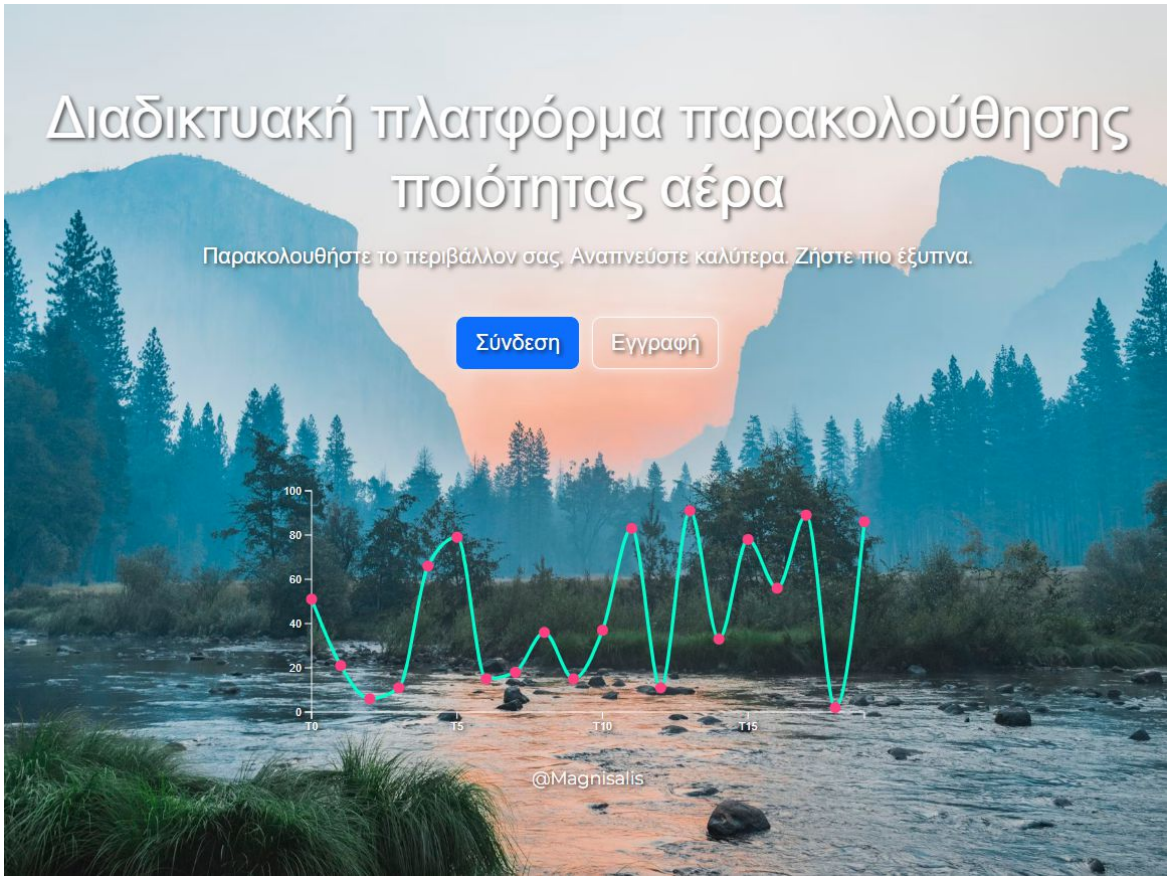
Εάν δεν είναι ενεργή, τα δεδομένα απορρίπτονται και ο server ενημερώνει ότι η συσκευή δεν έχει ενεργοποιηθεί.

Ο χρήστης, αφού συνδεθεί στο Dashboard, μπορεί να δει τις μη ενεργοποιημένες συσκευές και να κατοχυρώσει τη συσκευή του, εισάγοντας το UUID. Το σύστημα ενημερώνει τότε το is_active = 1 και καταγράφει τον current_user_id, ολοκληρώνοντας τη διαδικασία ενεργοποίησης.

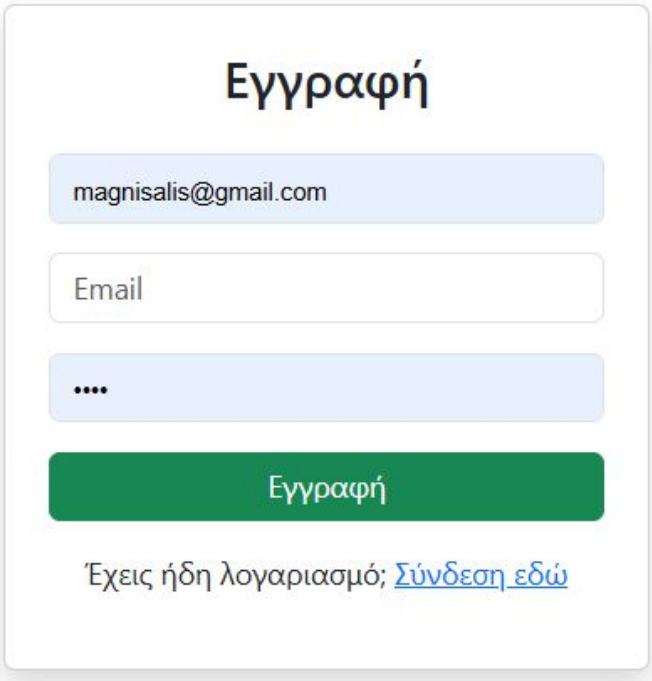
Μετά την κατοχύρωση, η συσκευή μπορεί πλέον να στέλνει κανονικά δεδομένα, τα οποία γίνονται άμεσα ορατά στον πίνακα μετρήσεων και στα διαγράμματα του χρήστη, διευκολύνοντας την παρακολούθηση της ποιότητας του αέρα σε πραγματικό χρόνο.

4.2 Η λειτουργία του συστήματος

Στην Εικόνα 4.5 παρουσιάζεται η αρχική σελίδα υποδοχής της διαδικτυακής πλατφόρμας παρακολούθησης ποιότητας αέρα. Έχει γραφικό περιβάλλον σε συνδυασμό με την προβολή ενός ενδεικτικού γραφήματος μετρήσεων. Οι χρήστες έχουν τη δυνατότητα να επιλέξουν άμεσα μεταξύ σύνδεσης και εγγραφής και διευκολύνουν την πρόσβασή τους στο σύστημα.



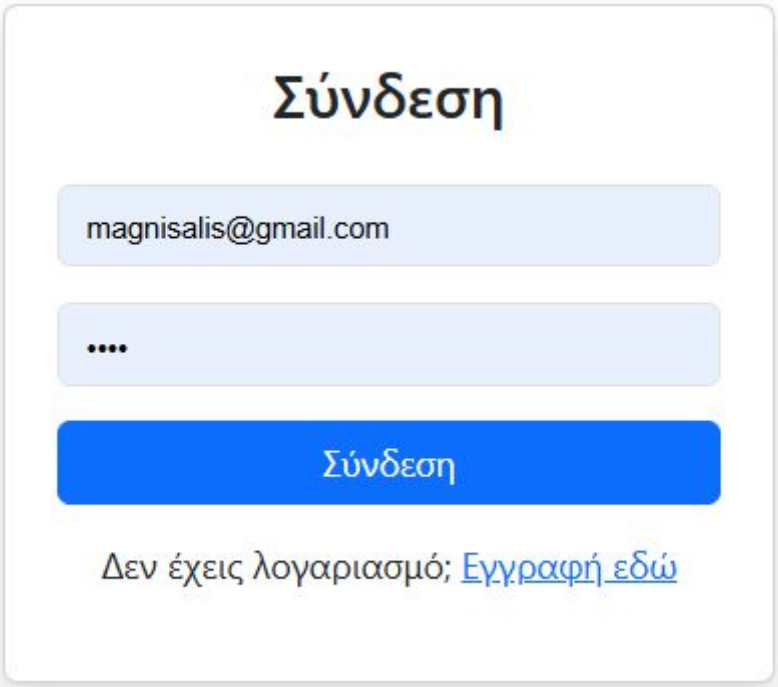
Εικόνα 4.5: Αρχική σελίδα



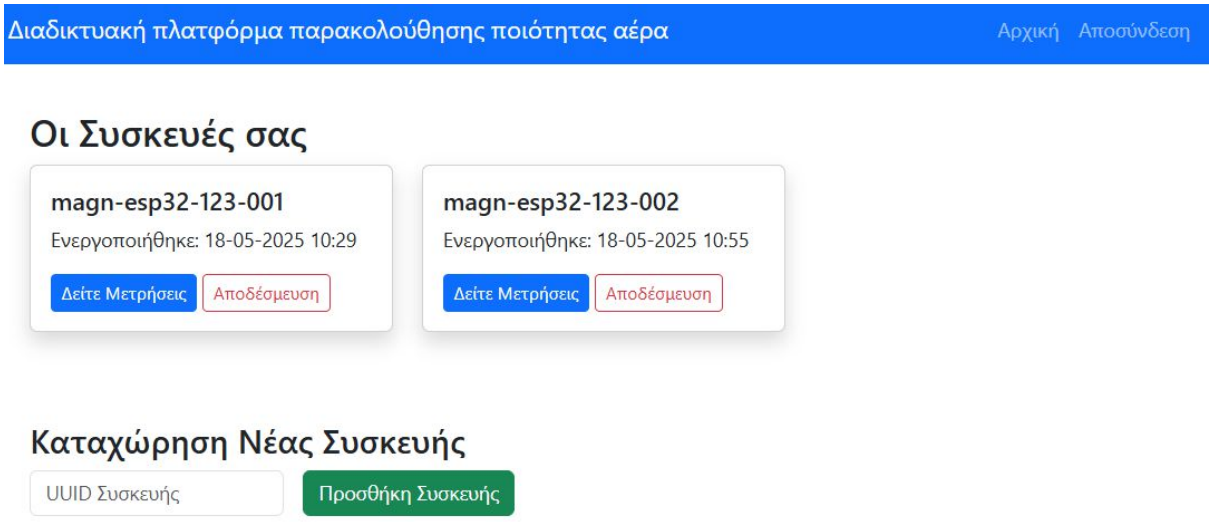
Εικόνα 4.6: Εγγραφή Χρήστη

Στην Εικόνα 4.6 παρουσιάζεται η φόρμα εγγραφής νέου χρήστη στην πλατφόρμα παρακολούθησης ποιότητας αέρα. Ο χρήστης καλείται να εισάγει τα απαραίτητα στοιχεία, όπως όνομα χρήστη, διεύθυνση email και κωδικό πρόσβασης, προκειμένου να δημιουργήσει λογαριασμό. Η διαδικασία είναι απλή και φιλική προς τον χρήστη, με στόχο την άμεση πρόσβαση στις υπηρεσίες της πλατφόρμας και τη δυνατότητα διαχείρισης των συσκευών μέτρησης που θα συνδέσει στον λογαριασμό του.

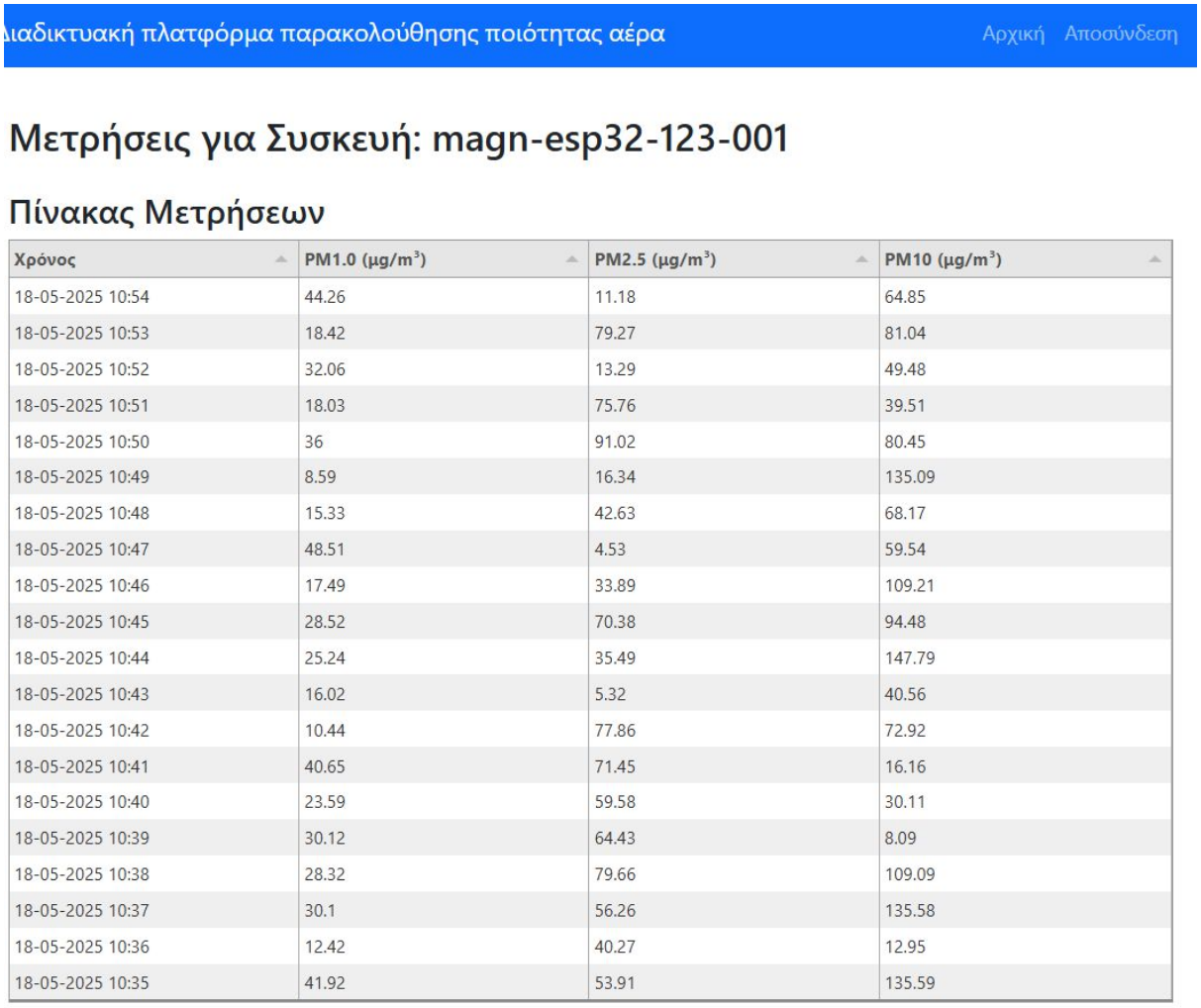
Στην Εικόνα 4.7 παρουσιάζεται η φόρμα σύνδεσης χρήστη στην πλατφόρμα παρακολούθησης ποιότητας αέρα. Ο χρήστης εισάγει τη διεύθυνση email και τον κωδικό πρόσβασης που έχει δηλώσει κατά την εγγραφή, προκειμένου να αποκτήσει πρόσβαση στις διαθέσιμες λειτουργίες του συστήματος. Μέσα από τη διαδικασία σύνδεσης, διασφαλίζεται η προστασία των προσωπικών δεδομένων και η ασφαλής πρόσβαση στα στοιχεία των συνδεδεμένων συσκευών και στις μετρήσεις τους.



Εικόνα 4.7: Σύνδεση Χρήστη



Εικόνα 4.8: Πίνακας Ελέγχου των συσκευών

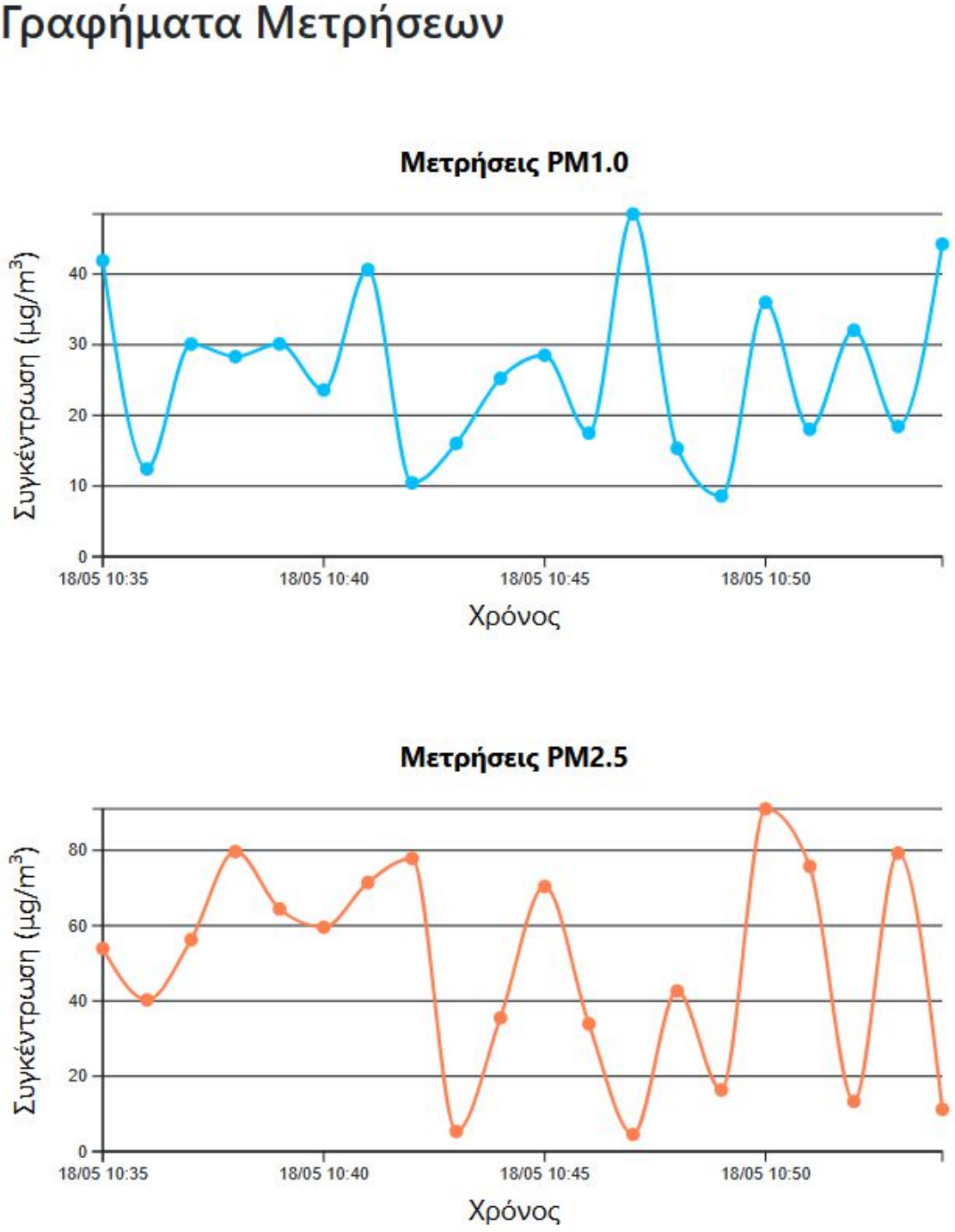


Χρόνος	PM1.0 (μg/m³)	PM2.5 (μg/m³)	PM10 (μg/m³)
18-05-2025 10:54	44.26	11.18	64.85
18-05-2025 10:53	18.42	79.27	81.04
18-05-2025 10:52	32.06	13.29	49.48
18-05-2025 10:51	18.03	75.76	39.51
18-05-2025 10:50	36	91.02	80.45
18-05-2025 10:49	8.59	16.34	135.09
18-05-2025 10:48	15.33	42.63	68.17
18-05-2025 10:47	48.51	4.53	59.54
18-05-2025 10:46	17.49	33.89	109.21
18-05-2025 10:45	28.52	70.38	94.48
18-05-2025 10:44	25.24	35.49	147.79
18-05-2025 10:43	16.02	5.32	40.56
18-05-2025 10:42	10.44	77.86	72.92
18-05-2025 10:41	40.65	71.45	16.16
18-05-2025 10:40	23.59	59.58	30.11
18-05-2025 10:39	30.12	64.43	8.09
18-05-2025 10:38	28.32	79.66	109.09
18-05-2025 10:37	30.1	56.26	135.58
18-05-2025 10:36	12.42	40.27	12.95
18-05-2025 10:35	41.92	53.91	135.59

Εικόνα 4.9: Μετρήσεις για μια συσκευή

Στην Εικόνα 4.8 παρουσιάζεται ο πίνακας ελέγχου των συσκευών, μέσω του οποίου ο χρήστης μπορεί να διαχειρίζεται εύκολα και αποδοτικά τις συσκευές που έχει καταχωρήσει στην πλατφόρμα. Για κάθε συσκευή εμφανίζονται πληροφορίες, όπως το μοναδικό αναγνωριστικό (UUID) και η ημερομηνία ενεργοποίησής της. Ο χρήστης έχει τη δυνατότητα να επιλέξει την προβολή των μετρήσεων κάθε συσκευής ή να την αποδεσμεύσει από τον λογαριασμό του, όταν δεν τη χρησιμοποιεί πλέον. Επίσης δίνεται η δυνατότητα άμεσης καταχώρησης νέας συσκευής μέσω της εισαγωγής του UUID της και βοηθά στην επέκταση του συστήματος με επιπλέον σημεία μέτρησης.

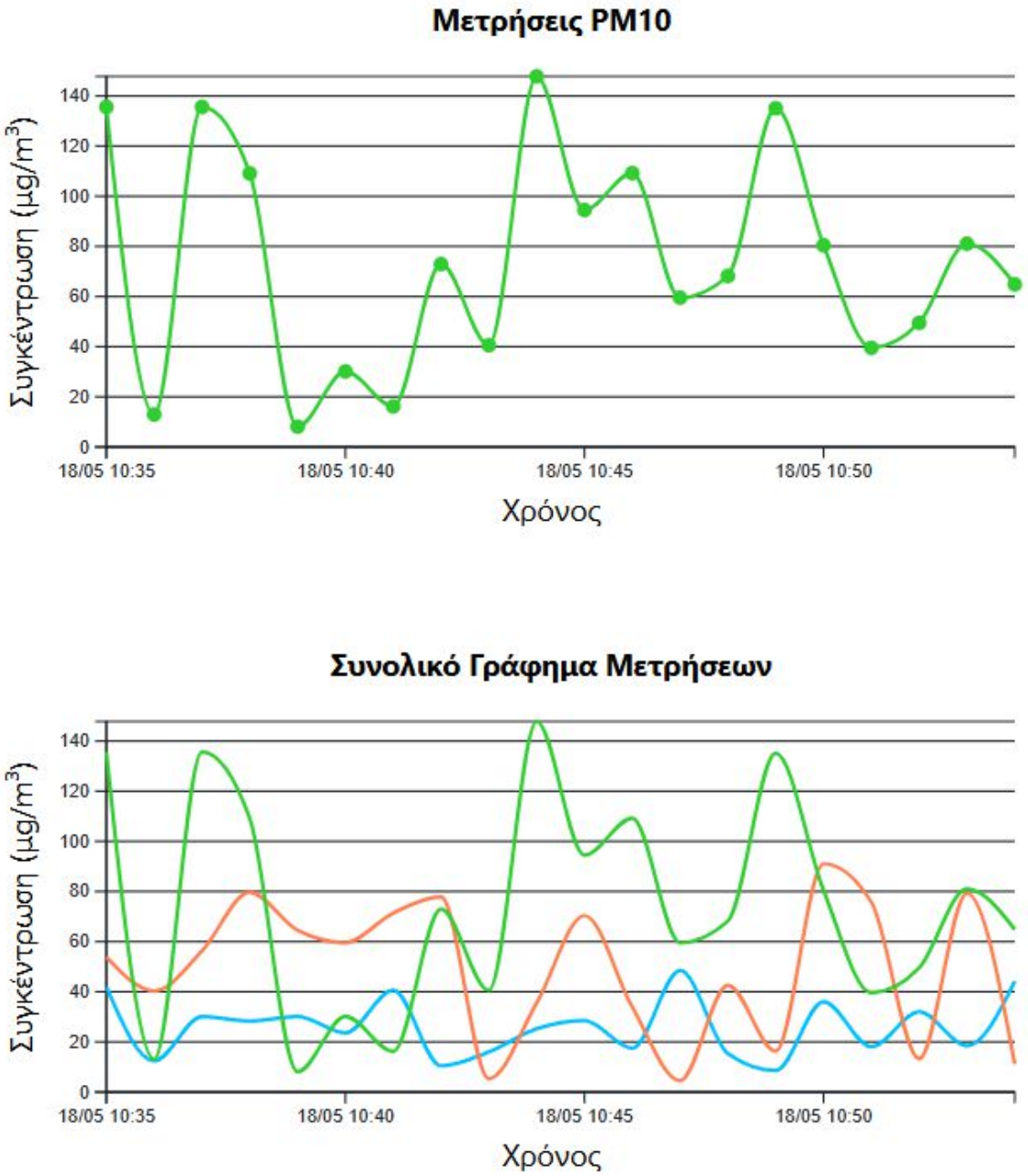
Στην Εικόνα 4.9 παρουσιάζεται η σελίδα προβολής των μετρήσεων για μια επιλεγμένη συσκευή. Ο χρήστης μπορεί να δει σε μορφή πίνακα τις καταγεγραμμένες μετρήσεις οι οποίες περιλαμβάνουν την ημερομηνία και ώρα λήψης και τις συγκεντρώσεις σωματιδίων PM1.0, PM2.5 και PM10 σε μονάδες $\mu g/m^3$. Ο πίνακας έχει παρουσίαση των δεδομένων διευκολύνοντας τον χρήστη να παρακολουθεί την ποιότητα του αέρα σε συγκεκριμένες χρονικές στιγμές και να εντοπίζει εύκολα περιόδους με αυξημένα επίπεδα ρύπων.



Εικόνα 4.10: Γραφήματα Μετρήσεων 1

Στην Εικόνα 4.10 παρουσιάζεται το πρώτο σετ γραφημάτων μετρήσεων της επιλεγμένης συσκευής. Κάθε γράφημα απεικονίζει τη μεταβολή της συγκέντρωσης ενός τύπου αιωρούμενων σωματιδίων (PM1.0, PM2.5 ή PM10) σε σχέση με τον χρόνο. Ο άξονας Χ αντιστοιχεί στον χρόνο καταγραφής των μετρήσεων, ενώ ο άξονας Υ δείχνει τη συγκέντρωση των σωματιδίων σε μονάδες μg/m³. Τα γραφήματα περιλαμβάνουν τίτλους, ετικέτες αξόνων και πλέγμα για καλύτερη ανάλυση των δεδομένων,

προσφέροντας στον χρήστη τη δυνατότητα να παρακολουθήσει την πορεία της ποιότητας του αέρα ανά χρονική περίοδο.



Εικόνα 4.11: Γραφήματα Μετρήσεων 2

Στην Εικόνα 4.11 παρουσιάζεται το συγκεντρωτικό γράφημα μετρήσεων, στο οποίο απεικονίζονται ταυτόχρονα οι συγκεντρώσεις των αιωρούμενων σωματιδίων PM1.0, PM2.5 και PM10 σε συνάρτηση με τον χρόνο. Κάθε τύπος σωματιδίων διακρίνεται με διαφορετικό χρώμα ενώ στο γράφημα

περιλαμβάνεται και σχετικός υπόμνημα (legend) για την εύκολη αναγνώριση των τιμών. Ο άξονας X παρουσιάζει τον χρόνο καταγραφής των μετρήσεων και ο άξονας Y τη συγκέντρωση των σωματιδίων σε μg/m³. Το γράφημα διαθέτει τίτλο, ετικέτες αξόνων και πλέγμα, επιτρέποντας στον χρήστη να πραγματοποιεί άμεσες συγκρίσεις μεταξύ των τριών τύπων σωματιδίων και να εντοπίζει περιόδους με αυξημένα επίπεδα ρύπανσης.

4.3 Από τη μεριά του esp32

Η δομή των αρχείων του έργου είναι οργανωμένη σύμφωνα με τις βέλτιστες πρακτικές του Django Framework, ώστε να διασφαλίζεται η ευκολία ανάπτυξης, η επεκτασιμότητα και η σωστή οργάνωση του κώδικα.

Παρακάτω παρουσιάζεται η βασική δομή των φακέλων και των αρχείων, με περιγραφή του ρόλου του καθενός:

Ο κύριος φάκελος του έργου που περιέχει τις ρυθμίσεις του Django, τα modules, και τα στατικά αρχεία.

- **manage.py**
  Το βασικό εκτελέσιμο script του Django για τη διαχείριση του έργου (εκκίνηση server, migrations, κ.ά.).
- **air_quality/** (Main Django Project Folder)
  - **settings.py**
    Περιέχει όλες τις βασικές ρυθμίσεις του έργου, όπως ρυθμίσεις βάσης δεδομένων, στατικών αρχείων και ενεργοποιημένων εφαρμογών (INSTALLED_APPS).
  - **urls.py**
    Ορίζονται τα global routes του έργου και γίνεται σύνδεση με τα URLs των επιμέρους εφαρμογών (π.χ., api/).

api/

Αυτή η εφαρμογή διαχειρίζεται την επιχειρησιακή λογική του συστήματος, τις διεπαφές με τον χρήστη και το API για την παραλαβή μετρήσεων.

- **models.py**
  Ορισμός των μοντέλων της βάσης δεδομένων (Device, Measurement). Κάθε μοντέλο αντιπροσωπεύει έναν πίνακα στη βάση.
- **views.py**
  Περιέχει τις συναρτήσεις (views) που χειρίζονται τα HTTP αιτήματα, τόσο για τα API endpoints όσο και για τις HTML σελίδες (π.χ., welcome_view, receive_measurement, dashboard).

- **urls.py**
  Ορισμός των routes της εφαρμογής (π.χ., /api/measurements/, /dashboard/, /login/).
- **templates/**
  Περιέχει όλα τα HTML αρχεία του frontend, οργανωμένα σε σελίδες:
  - **welcome.html** – Αρχική σελίδα καλωσορίσματος.
  - **login.html** – Σελίδα σύνδεσης.
  - **register.html** – Σελίδα εγγραφής νέου χρήστη.
  - **dashboard.html** – Πίνακας ελέγχου συσκευών.
  - **measurements.html** – Προβολή μετρήσεων και γραφημάτων.
- **static/**
  Περιέχει στατικά αρχεία, όπως CSS, JavaScript (π.χ., D3.js scripts), εικόνες και custom frontend resources.

```
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

const char* serverUrl = "http://127.0.0.1:8000/api/measurements/";
const char* deviceUUID = "magn-esp32-123-002";
```

Σε αυτό το τμήμα ορίζονται οι βασικές πληροφορίες για τη σύνδεση της συσκευής στο ασύρματο δίκτυο (Wi-Fi), καθώς και η διεύθυνση URL του server στον οποίο θα αποστέλλονται οι μετρήσεις. Επιπλέον, καθορίζεται το μοναδικό αναγνωριστικό (UUID) της συσκευής, που χρησιμοποιείται για την αναγνώρισή της από το σύστημα.

```
#define PMS_RX 16
#define PMS_TX 17
SoftwareSerial pmsSerial(PMS_RX, PMS_TX);
```

Ορίζονται οι ακίδες του ESP32 που χρησιμοποιούνται για την επικοινωνία με τον αισθητήρα PMS5003. Η βιβλιοθήκη SoftwareSerial επιτρέπει την επικοινωνία μέσω εικονικής σειριακής θύρας, απελευθερώνοντας τις βασικές σειριακές θύρες του ESP32 για άλλες λειτουργίες.

```
void setup() {
```

```
  Serial.begin(115200);
  pmsSerial.begin(9600);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected!");
}
```

Η συνάρτηση setup() εκτελείται μία φορά κατά την εκκίνηση της συσκευής. Αρχικοποιεί την επικοινωνία με τον αισθητήρα PMS5003 και τη σειριακή θύρα για αποστολή μηνυμάτων στον υπολογιστή (debugging). Στη συνέχεια, επιχειρεί σύνδεση στο ασύρματο δίκτυο και περιμένει μέχρι να επιτευχθεί επιτυχώς.

```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    if (readAndSendData()) {
      Serial.println("Data sent successfully!");
    } else {
      Serial.println("Failed to send data.");
    }
  } else {
    WiFi.reconnect();
  }

  delay(60000);
}
```

Η συνάρτηση loop() εκτελείται συνεχώς μετά την setup(). Ελέγχει αν η συσκευή είναι συνδεδεμένη στο διαδίκτυο και, αν ναι, διαβάζει μετρήσεις από τον αισθητήρα και τις αποστέλλει στον server. Σε περίπτωση αποσύνδεσης, προσπαθεί να επανασυνδεθεί. Η διαδικασία αυτή επαναλαμβάνεται κάθε 60 δευτερόλεπτα.

```
bool readAndSendData() {
  uint8_t buffer[32];
  if (pmsSerial.available() >= 32) {
    if (pmsSerial.read() == 0x42 && pmsSerial.read() == 0x4D) {
      buffer[0] = 0x42;
      buffer[1] = 0x4D;
      pmsSerial.readBytes(&buffer[2], 30);

      uint16_t pm1_0 = (buffer[10] << 8) | buffer[11];
      uint16_t pm2_5 = (buffer[12] << 8) | buffer[13];
      uint16_t pm10 = (buffer[14] << 8) | buffer[15];

      String payload = "{";
      payload += "\"uuid\":\"" + String(deviceUUID) + "\",";
      payload += "\"pm1_0\":" + String(pm1_0) + ",";
      payload += "\"pm2_5\":" + String(pm2_5) + ",";
      payload += "\"pm10\":" + String(pm10) + "}";

      return sendToServer(payload);
    }
  }
  return false;
}
```

Η συνάρτηση αυτή διαβάζει τα δεδομένα που στέλνει ο αισθητήρας PMS5003 μέσω της σειριακής θύρας. Ελέγχει αν το πακέτο δεδομένων είναι έγκυρο, εξάγει τις μετρήσεις PM1.0, PM2.5 και PM10 και δημιουργεί ένα JSON αντικείμενο με τις μετρήσεις και το UUID της συσκευής. Τέλος, καλεί τη συνάρτηση sendToServer() για την αποστολή των δεδομένων στον server.

```
bool sendToServer(String jsonPayload) {
  HTTPClient http;
  http.begin(serverUrl);
```

```
  http.addHeader("Content-Type", "application/json");

  int httpResponseCode = http.POST(jsonPayload);
  Serial.printf("HTTP Response Code: %d\n", httpResponseCode);

  http.end();
  return (httpResponseCode == 200 || httpResponseCode == 201);
}
```

Η συνάρτηση sendToServer() αναλαμβάνει την αποστολή των δεδομένων μέσω HTTP POST αιτήματος στον server. Χρησιμοποιεί τη βιβλιοθήκη HTTPClient για να δημιουργήσει και να στείλει το αίτημα, ορίζοντας το περιεχόμενο σε μορφή JSON. Επιστρέφει true αν η αποστολή ήταν επιτυχής (κωδικός απόκρισης 200 ή 201), αλλιώς επιστρέφει false.

4.4 Από τη μεριά του Server

```
@csrf_exempt
def receive_measurement(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            uuid = data.get('uuid')
            pm1_0 = data.get('pm1_0')
            pm2_5 = data.get('pm2_5')
            pm10 = data.get('pm10')

            if not uuid:
                return JsonResponse({'status': 'error', 'message': 'UUID is required.'}, status=400)

            device = Device.objects.filter(uuid=uuid).first()

            if not device:
```

```
            # Νέα συσκευή, δεν είναι καταχωρημένη
            Device.objects.create(uuid=uuid, is_active=False)
            return JsonResponse({'status': 'error', 'message': 'Device not registered.'}, status=403)

        if not device.is_active:
            return JsonResponse({'status': 'error', 'message': 'Device not yet activated.'}, status=403)

        Measurement.objects.create(
            device_id=device.id,
            pm1_0=pm1_0,
            pm2_5=pm2_5,
            pm10=pm10,
            timestamp=timezone.now()
        )

        return JsonResponse({'status': 'success', 'message': 'Data saved successfully.'}, status=200)

    except Exception as e:
        return JsonResponse({'status': 'error', 'message': str(e)}, status=500)

return JsonResponse({'status': 'error', 'message': 'Invalid request method.'}, status=405)
```

Αυτή η συνάρτηση είναι το βασικό API endpoint για την παραλαβή μετρήσεων από τις συσκευές.

- Ελέγχει αν το UUID της συσκευής υπάρχει στη βάση.
- Αν δεν υπάρχει, δημιουργεί νέα εγγραφή και επιστρέφει σφάλμα για μη καταχωρημένη συσκευή.
- Αν η συσκευή δεν έχει ενεργοποιηθεί, απορρίπτει τις μετρήσεις.
- Όταν όλα είναι έγκυρα, αποθηκεύει τη μέτρηση στη βάση δεδομένων.

```
def login_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
```

```
        with connection.cursor() as cursor:
            cursor.execute("SELECT id FROM users WHERE email=%s AND password_hash=%s", [email, password])
            result = cursor.fetchone()
            if result:
                request.session[SESSION_USER_ID] = result[0]
                return redirect('dashboard')
            else:
                messages.error(request, 'Invalid credentials.')

    return render(request, 'login.html')
```

Η συνάρτηση αυτή χειρίζεται τη διαδικασία σύνδεσης χρηστών.

- Ελέγχει τα στοιχεία εισόδου από τη φόρμα.
- Αν βρεθεί ο χρήστης στη βάση, δημιουργείται session και ο χρήστης μεταφέρεται στον πίνακα ελέγχου.
- Αν όχι, εμφανίζεται μήνυμα λάθους.

```
def register_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        username = request.POST.get('username')

        if not email or not password or not username:
            messages.error(request, 'All fields are required.')
            return render(request, 'register.html')

        with connection.cursor() as cursor:
            cursor.execute("SELECT id FROM users WHERE email=%s", [email])
            if cursor.fetchone():
                messages.error(request, 'Email already registered.')
```

```
            return render(request, 'register.html')


        cursor.execute(
            "INSERT INTO users (username, email, password_hash, created_at) VALUES (%s, %s, %s, NOW())",
            [username, email, password]
        )
        messages.success(request, 'Registration successful. Please log in.')
        return redirect('login')


    return render(request, 'register.html')
```

Η συνάρτηση αυτή διαχειρίζεται την εγγραφή νέων χρηστών.

- Ελέγχει ότι όλα τα πεδία της φόρμας είναι συμπληρωμένα.
- Αν το email υπάρχει ήδη, εμφανίζει σχετικό μήνυμα.
- Διαφορετικά, καταχωρεί νέο χρήστη στη βάση.

```
def dashboard(request):
    user_id = request.session.get(SESSION_USER_ID)
    if not user_id:
        return redirect('login')


    devices = Device.objects.filter(current_user_id=user_id)
    return render(request, 'dashboard.html', {'devices': devices})
```

Εμφανίζει στον συνδεδεμένο χρήστη τον πίνακα με τις συσκευές που του ανήκουν. Αν δεν υπάρχει ενεργό session, ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης.

4.5 Η βάση δεδομένων που χρησιμοποιήθηκε

Η βάση δεδομένων αποτελεί τον πυρήνα αποθήκευσης και διαχείρισης όλων των απαραίτητων πληροφοριών του συστήματος παρακολούθησης ποιότητας αέρα. Υλοποιήθηκε με χρήση του συστήματος διαχείρισης σχεσιακών βάσεων MySQL, το οποίο παρέχει καλή απόδοση, αξιοπιστία και ευκολία στη διαχείριση μεγάλων όγκων δεδομένων.

Η σχεδίαση της βάσης δεδομένων βασίστηκε σε αρχές απλότητας και επεκτασιμότητας, αποφεύγοντας τη χρήση σύνθετων περιορισμών (όπως foreign keys), ώστε να διευκολύνεται η διαχείριση και η ανάπτυξη νέων λειτουργιών.

Πίνακας users

Ο πίνακας users αποθηκεύει τα βασικά στοιχεία των χρηστών που έχουν πρόσβαση στην πλατφόρμα.

Πεδίο	Τύπος	Περιγραφή
id	INT (Primary)	Μοναδικό αναγνωριστικό χρήστη.
username	VARCHAR(100)	Όνομα χρήστη.
email	VARCHAR(150)	Διεύθυνση email.
password_hash	VARCHAR(150)	Κωδικός πρόσβασης (αποθηκευμένος ως hash).
created_at	DATETIME	Ημερομηνία εγγραφής.

Για λόγους ασφαλείας, οι κωδικοί πρέπει να αποθηκεύονται με ασφαλή hash, όπως bcrypt ή sha256.

Πίνακας devices

Ο πίνακας devices αποθηκεύει τις συσκευές μέτρησης, καθεμία από τις οποίες έχει ένα μοναδικό UUID.

Πεδίο	Τύπος	Περιγραφή
id	INT (Primary)	Μοναδικό αναγνωριστικό συσκευής.
uuid	VARCHAR(100)	Μοναδικό αναγνωριστικό συσκευής (UUID).
name	VARCHAR(100)	Προαιρετική φιλική ονομασία.
first_activation	DATETIME	Ημερομηνία πρώτης ενεργοποίησης.
is_active	BOOLEAN	Κατάσταση ενεργοποίησης (0/1).
current_user_id	INT	ID του κατόχου χρήστη (χωρίς foreign key).
created_at	DATETIME	Ημερομηνία καταχώρησης στη βάση.

Η χρήση του πεδίου is_active επιτρέπει τον έλεγχο αν η συσκευή έχει κατοχυρωθεί ή παραμένει ανενεργή.

Πίνακας measurements

Ο πίνακας measurements καταγράφει τις μετρήσεις ατμοσφαιρικών ρύπων που αποστέλλονται από τις συσκευές.

Πεδίο	Τύπος	Περιγραφή
id	INT (Primary)	Μοναδικό αναγνωριστικό μέτρησης.
device_id	INT	Αναγνωριστικό συσκευής (χωρίς foreign key).
timestamp	DATETIME	Χρόνος λήψης της μέτρησης.
pm1_0	FLOAT	Συγκέντρωση PM1.0 (μg/m³).
pm2_5	FLOAT	Συγκέντρωση PM2.5 (μg/m³).
pm10	FLOAT	Συγκέντρωση PM10 (μg/m³).
extra_data	JSON (Nullable)	Πρόσθετα δεδομένα (προαιρετικά).

Το πεδίο extra_data μπορεί να χρησιμοποιηθεί για μελλοντική αποθήκευση επιπλέον παραμέτρων, όπως θερμοκρασία, υγρασία κ.λπ.

Η επιλογή αποφυγής της χρήσης περιορισμών FOREIGN KEY έγινε συνειδητά για τους εξής λόγους:

- Απλότητα στη διαχείριση των σχέσεων και ευκολότερη αντιμετώπιση λαθών εγγραφών.
- Δυνατότητα αποδέσμευσης συσκευών από χρήστες με απλή ενημέρωση του πεδίου current_user_id.
- Βελτίωση της απόδοσης σε περιβάλλοντα με μεγάλες ποσότητες δεδομένων και συχνές αλλαγές σχέσεων.

Όλες οι σχέσεις διαχειρίζονται μέσω του λογισμικού επιπέδου Django και ελέγχονται με σχετικούς ελέγχους στους controllers (views).

Κεφάλαιο 5ο: Αξιολόγηση Συστήματος και Προτεινόμενες Βελτιώσεις

Το σύστημα παρακολούθησης ποιότητας αέρα που αναπτύξαμε επιτρέπει τη μέτρηση και την παρακολούθηση της ατμοσφαιρικής ρύπανσης με τη βοήθεια έξυπνων συσκευών και μιας διαδικτυακής πλατφόρμας. Στη συνέχεια, γίνεται μια αξιολόγηση για το πώς λειτουργεί το σύστημα σήμερα, ποια πλεονεκτήματα προσφέρει αλλά και τι θα μπορούσε να βελτιωθεί στο μέλλον.

Πλεονεκτήματα του Συστήματος

Άμεση Ενημέρωση Χρηστών:

Οι χρήστες μπορούν να δουν τις μετρήσεις για την ποιότητα του αέρα σε πραγματικό χρόνο. Αυτό σημαίνει ότι γνωρίζουν άμεσα αν η ατμόσφαιρα γύρω τους είναι καθαρή ή αν υπάρχει ρύπανση.

Εύχρηστο Περιβάλλον:

Η ιστοσελίδα της εφαρμογής είναι απλή και εύκολη στη χρήση. Οι χρήστες μπορούν να δουν τις συσκευές τους, τις μετρήσεις και όμορφα γραφήματα που βοηθούν στην κατανόηση των δεδομένων.

Ασφαλής Διαχείριση Συσκευών:

Κάθε συσκευή έχει ένα μοναδικό όνομα (UUID) και μπορεί να συνδεθεί μόνο από τον χρήστη που την έχει κατοχυρώσει. Αυτό προστατεύει τα δεδομένα και την ιδιωτικότητα των χρηστών.

Απλή Εγκατάσταση και Λειτουργία:

Το σύστημα μπορεί να εγκατασταθεί εύκολα σε έναν μικροελεγκτή όπως το ESP32 και να λειτουργήσει με απλό τρόπο χωρίς ειδικές γνώσεις.

Προβλήματα και Περιορισμοί που Παρατηρήθηκαν

Έλλειψη Ασφαλών Κωδικών:

Αυτή τη στιγμή, οι κωδικοί των χρηστών δεν αποθηκεύονται με ασφάλεια (δεν χρησιμοποιούνται κρυπτογραφημένες μέθοδοι). Αυτό μπορεί να είναι επικίνδυνο αν κάποιος προσπαθήσει να υποκλέψει δεδομένα.

Αποστολή Δεδομένων χωρίς Κρυπτογράφηση:

Η επικοινωνία μεταξύ των συσκευών και του server γίνεται μέσω απλού HTTP και όχι με ασφαλή σύνδεση (HTTPS). Αυτό σημαίνει ότι κάποιος κακόβουλος μπορεί να δει τα δεδομένα που αποστέλλονται.

Περιορισμένη Ανάλυση Δεδομένων:

Αν και υπάρχουν γραφήματα, δεν παρέχονται πιο σύνθετες αναλύσεις, όπως μέσος όρος ανά ημέρα, προειδοποιήσεις σε περίπτωση υψηλής ρύπανσης ή συγκρίσεις με τα επιτρεπτά όρια ρύπων.

Μη Υποστήριξη Πολλαπλών Τύπων Δεδομένων:

Το σύστημα αυτή τη στιγμή μετρά μόνο τα σωματίδια PM1.0, PM2.5 και PM10. Δεν καταγράφονται άλλες σημαντικές παράμετροι, όπως η θερμοκρασία, η υγρασία ή τα αέρια του περιβάλλοντος (CO_2, NO_2 κ.λπ.).

Προτάσεις για Βελτίωση του Συστήματος

Καλύτερη Ασφάλεια Δεδομένων:

- Να χρησιμοποιούνται ασφαλείς κωδικοί (με κρυπτογράφηση).
- Να γίνει μετάβαση σε ασφαλή σύνδεση HTTPS για την αποστολή των μετρήσεων.

Προσθήκη Νέων Αισθητήρων:

- Να ενσωματωθούν και άλλοι αισθητήρες, ώστε να μετρώνται και άλλες σημαντικές περιβαλλοντικές παράμετροι όπως θερμοκρασία, υγρασία, και επίπεδα CO_2.

Πιο Προηγμένη Ανάλυση Δεδομένων:

- Να δημιουργηθούν γραφήματα που να δείχνουν τον μέσο όρο μετρήσεων ανά ημέρα ή εβδομάδα.
- Να υπάρχουν ειδοποιήσεις στον χρήστη όταν οι μετρήσεις είναι επικίνδυνες για την υγεία.

Δημιουργία Εφαρμογής για Κινητά:

- Να αναπτυχθεί μια απλή εφαρμογή για κινητά τηλέφωνα, ώστε οι χρήστες να λαμβάνουν άμεσα ειδοποιήσεις και να βλέπουν τα δεδομένα από όπου κι αν βρίσκονται.

Καλύτερη Οργάνωση Ιστορικών Δεδομένων:

- Να υπάρχει η δυνατότητα να κατεβάζει ο χρήστης τα ιστορικά δεδομένα σε μορφή αρχείων (CSV, Excel) για προσωπική ανάλυση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Marques, G., Roque Ferreira, C., & Pitarma, R. (2018). A system based on the internet of things for real-time particle monitoring in buildings. International journal of environmental research and public health, 15(4), 821.

[2] Divan, M. J., Sanchez-Reynoso, M. L., Panebianco, J. E., & Mendez, M. J. (2021). IoT-based approaches for monitoring the particulate matter and its impact on health. IEEE Internet of Things Journal, 8(15), 11983-12003.

[3] Ghizlane, F., Mabrouki, J., Ghrissi, F., & Azrour, M. (2022). Proposal for a high-resolution particulate matter (PM10 and PM2. 5) capture system, comparable with hybrid system-based internet of things: case of quarries in the western rif, Morocco. Pollution, 8(1), 169-180.

[4] https://ypen.gov.gr/perivallon/poiotita-tis-atmosfairas/dedomena-metriseon-atmosfairikis-rypansis/

[5] https://agrospecom.gr/aeright-parakolouthisi-poiotitas-aera-24-7/

[6] https://es.wikipedia.org/wiki/ESP32

[7] https://randomnerdtutorials.com/getting-started-with-esp32/

[8] https://learnesp32.com/

[9] https://www.adafruit.com/product/3686

[10] https://aqicn.org/sensor/pms5003-7003/el/

[11] https://www.w3schools.com/python/

[12] https://en.wikipedia.org/wiki/Python_(programming_language)

[13] https://realpython.com/

[14] https://www.codecademy.com/catalog/language/python

[15] https://serokell.io/blog/python-pros-and-cons

[16] https://www.djangoproject.com/

[17] https://en.wikipedia.org/wiki/Django_(web_framework)

[18] https://www.w3schools.com/django/django_intro.php

[19] https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Introduction

[20] https://realpython.com/get-started-with-django-1/

[21] https://krify.co/advantages-and-disadvantages-of-django/

[22] https://www.w3schools.com/graphics/plot_google_chart.asp

[23] https://developers.google.com/chart

[24] https://tabulator.info/

ΠΑΡΑΡΤΗΜΑ

Views.py

```
from django.shortcuts import render, redirect
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.contrib import messages
from django.utils import timezone
from .models import Device, Measurement
from django.db import connection
import json

# Dummy User Authentication (Assuming user ID 1 is logged in for demo)
SESSION_USER_ID = 'user_id'

def welcome_view(request):
    return render(request, 'welcome.html')

@csrf_exempt
def receive_measurement(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            uuid = data.get('uuid')
            pm1_0 = data.get('pm1_0')
            pm2_5 = data.get('pm2_5')
            pm10 = data.get('pm10')

            if not uuid:
                return JsonResponse({'status': 'error', 'message': 'UUID is required.'}, status=400)

            device = Device.objects.filter(uuid=uuid).first()

            if not device:
                # New Device - Not Registered
                Device.objects.create(uuid=uuid, is_active=False)
               return JsonResponse({'status': 'error', 'message': 'Device not registered.'}, status=403)

            if not device.is_active:
```

```
                return JsonResponse({'status': 'error', 'message': 'Device not yet activated.'}, status=403)

            Measurement.objects.create(
                device_id=device.id,
                pm1_0=pm1_0,
                pm2_5=pm2_5,
                pm10=pm10,
                timestamp=timezone.now()
            )

           return JsonResponse({'status': 'success', 'message': 'Data saved successfully.'}, status=200)

        except Exception as e:
            return JsonResponse({'status': 'error', 'message': str(e)}, status=500)

    return JsonResponse({'status': 'error', 'message': 'Invalid request method.'}, status=405)

def login_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        # Simple User Check (replace with proper validation)
        with connection.cursor() as cursor:
           cursor.execute("SELECT id FROM users WHERE email=%s AND password_hash=%s", [email, password])
            result = cursor.fetchone()
            if result:
                request.session[SESSION_USER_ID] = result[0]
                return redirect('dashboard')
            else:
                messages.error(request, 'Invalid credentials.')

    return render(request, 'login.html')

def register_view(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        username = request.POST.get('username')

        if not email or not password or not username:
```

```
            messages.error(request, 'All fields are required.')
            return render(request, 'register.html')

        with connection.cursor() as cursor:
            # Check if email already exists
            cursor.execute("SELECT id FROM users WHERE email=%s", [email])
            if cursor.fetchone():
                messages.error(request, 'Email already registered.')
                return render(request, 'register.html')

            # Insert new user (In production, use hashed passwords!)
            cursor.execute(
                 "INSERT INTO users (username, email, password_hash, created_at) VALUES (%s, %s, %s,
NOW())",
                [username, email, password]
            )
            messages.success(request, 'Registration successful. Please log in.')
            return redirect('login')

    return render(request, 'register.html')

def logout_view(request):
    request.session.flush()
    return redirect('welcome')

def dashboard(request):
    user_id = request.session.get(SESSION_USER_ID)
    if not user_id:
        return redirect('login')

    devices = Device.objects.filter(current_user_id=user_id)
    return render(request, 'dashboard.html', {'devices': devices})

def add_device(request):
    user_id = request.session.get(SESSION_USER_ID)
    if not user_id:
        return redirect('login')

    if request.method == 'POST':
        uuid = request.POST.get('uuid')
        device = Device.objects.filter(uuid=uuid).first()
```

```
        if device and not device.is_active:
            device.is_active = True
            device.current_user_id = user_id
            device.first_activation = timezone.now()
            device.save()
            messages.success(request, 'Device registered successfully.')
        else:
            messages.error(request, 'Device not found or already assigned.')

    return redirect('dashboard')

def release_device(request, device_id):
    user_id = request.session.get(SESSION_USER_ID)
    if not user_id:
        return redirect('login')

    device = Device.objects.filter(id=device_id, current_user_id=user_id).first()
    if device:
        device.current_user_id = None
        device.is_active = False
        device.save()
        messages.success(request, 'Device released successfully.')
    else:
        messages.error(request, 'Device not found or not owned by you.')

    return redirect('dashboard')

def device_measurements(request, device_id):
    user_id = request.session.get(SESSION_USER_ID)
    if not user_id:
        return redirect('login')

    device = Device.objects.filter(id=device_id, current_user_id=user_id).first()
    if not device:
        messages.error(request, 'Device not found or not owned by you.')
        return redirect('dashboard')

    measurements = Measurement.objects.filter(device_id=device_id).order_by('-timestamp')[:50]
    return render(request, 'measurements.html', {'device': device, 'measurements': measurements})
```

models.py

```
from django.db import models
from django.utils import timezone


class Device(models.Model):
    uuid = models.CharField(max_length=100, unique=True)
    name = models.CharField(max_length=100, null=True, blank=True)
    first_activation = models.DateTimeField(null=True, blank=True)
    is_active = models.BooleanField(default=False)
    current_user_id = models.IntegerField(null=True, blank=True)  # No Foreign Key
    created_at = models.DateTimeField(default=timezone.now)
    class Meta:
        db_table = 'devices'


class Measurement(models.Model):
    device_id = models.IntegerField()  # No Foreign Key
    timestamp = models.DateTimeField(default=timezone.now)
    pm1_0 = models.FloatField()
    pm2_5 = models.FloatField()
    pm10 = models.FloatField()
    extra_data = models.JSONField(null=True, blank=True)
    class Meta:
        db_table = 'measurements'
```

urls.py

```
from django.urls import path
from . import views


urlpatterns = [
    path('welcome/', views.welcome_view, name='welcome'),
    path('api/measurements/', views.receive_measurement, name='receive_measurement'),
    path('', views.welcome_view, name='welcome'),
    path('login/', views.login_view, name='login'),
```

```
    path('register/', views.register_view, name='register'),
    path('logout/', views.logout_view, name='logout'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('add_device/', views.add_device, name='add_device'),
    path('release_device/<int:device_id>/', views.release_device, name='release_device'),
                        path('device_measurements/<int:device_id>/',          views.device_measurements,
name='device_measurements'),
]
```