ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
στις Σύγχρονες Ηλεκτρονικές Τεχνολογίες

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ασύρματο Σύστημα Ανίχνευσης και Ανάλυσης RF Σημάτων μέσω Διαδικτυακής εφαρμογής

Σπυρίδων Μαρτζούκος
ΑΜ: 876

Επιβλέπων καθηγητής: Βασίλειος Χριστοφιλάκης

Ιωάννινα
2025-26

Περίληψη

Το αντικείμενο της παρούσας μεταπτυχιακής διατριβής ήταν η ανάπτυξη ενός ολοκληρωμένου συστήματος καταγραφής σημάτων, στην περιοχή των ραδιοσυχνοτήτων, προς μελέτη της εξασθένησης διαδρομής τους και τη σύγκριση με θεωρητικά μοντέλα σε ένα περιβάλλον διαδικτύου. Προς επίτευξη αυτού, δημιουργήθηκε κατάλληλο hardware δεκτών, για την ανίχνευση της ισχύος σήματος, και παράλληλα αναπτύχθηκε μία εφαρμογή διαδικτύου για την καταγραφή των μετρήσεων και την ανάλυση τους. Οι δέκτες πρόκειται για δύο πανομοιότυπες και ανεξάρτητες πλακέτες, οι οποίες βασίζονται σε μικροελεγκτή ESP και φέρουν δύο μετρητές ισχύος διαφορετικής ευαισθησίας, αλλά κοινής περιοχής συχνοτήτων, 10 έως 8000 MHz. Για την εκπομπή και λήψη σήματος χρησιμοποιήθηκαν δύο κατευθυντικές κεραίες εύρους 700 - 6000 MHz. Η υλοποίηση των πλακετών προβλέπει την ασύρματη επικοινωνία, μέσω Wi-Fi, με τον πυρήνα του συστήματος, έναν μικροϋπολογιστή Raspberry Pi, ο οποίος φιλοξενεί την εφαρμογή και αναλαμβάνει τη δικτύωση του χρήστη με τις πηγές μετρήσεων. Η ανταλλαγή δεδομένων από άκρη σε άκρη εφαρμόζεται στο πρωτόκολλο MQTT. Το ρόλο του διακομιστή αναλαμβάνει ο backend συνδυασμός Python-Flask-NGINX για βέλτιστη απόδοση στα πλαίσια του Raspberry Pi. Το frontend σχεδιάστηκε ώστε να προσφέρει ένα διαδραστικό και φιλικό προς τον χρήστη περιβάλλον διεξαγωγής πειραμάτων path loss και άμεση ανάλυση αυτών με προσαρμογή θεωρητικών μοντέλων διάδοσης. Το πειραματικό μέρος της εργασίας πραγματεύεται σενάρια διάδοσης RF σήματος σε διάφορους εσωτερικούς χώρους, με τις μετρήσεις να εστιάζουν στο εύρος συχνοτήτων 1 - 5 GHz. Εξετάστηκε, επίσης, η ικανότητα δύο σημαντικών εμπειρικών μοντέλων διάδοσης - του μοντέλου εξασθένησης διαδρομής ελεύθερου χώρου και του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης - να περιγράψουν τη συμπεριφορά του σήματος σε αυτές τις συνθήκες.

Abstract

The objective of this Master's thesis was the development of an integrated radio frequency (RF) signal recording system to study path loss and compare it with theoretical models within a web-based environment. To achieve this, specialized receiver hardware was developed for signal strength detection, alongside a web application for measurement logging and analysis. The receivers consist of two identical and independent boards based on the ESP microcontroller, featuring two power meters of varying sensitivity but sharing a common frequency range of 10 to 8000 MHz. Two directional antennas with a range of 700 – 6000 MHz were utilized for signal transmission and reception. The hardware design facilitates wireless communication via Wi-Fi with the system's core, a Raspberry Pi microcomputer, which hosts the application and manages the networking between the user and the measurement sources. End-to-end data exchange is implemented using the MQTT protocol. On the server side, a Python-Flask-NGINX backend stack was employed for optimal performance within the Raspberry Pi environment. The frontend was designed to provide an interactive and user-friendly interface for conducting path loss experiments and performing immediate analysis through the fitting of theoretical propagation models. The experiments focus on RF signal propagation scenarios in various indoor environments, with measurements centered on the 1-5 GHz frequency range. Additionally, the ability of two significant propagation models – the Free Space Path Loss (FSPL) model and the Log-Distance Path Loss model – to describe signal behavior under these conditions was evaluated.

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1: Εισαγωγή

1.1 Ηλεκτρομαγνητικά κύματα

Μέχρι τα μέσα του 19ου αιώνα δεν υπήρχε ξεκάθαρη συσχέτιση των αντικειμένων του ηλεκτρισμού και του μαγνητισμού. Η διατύπωση των ηλεκτρομαγνητικών κυμάτων έγινε το 1864 από τον Σκοτσέζο φυσικό και μαθητικό James Clerk Maxwell, ο οποίος μέσω του έργου του «A dynamical Theory of the Electromagnetic Field» ενοποίησε όλες τις πρόδρομες πειραματικές μελέτες ενός αιώνα σε ένα σύνολο εξισώσεων, γνωστές ως Εξισώσεις του Maxwell. Το 1887, ο Heinrich Hertz επιβεβαίωσε τη θεωρία με μια σειρά πειραμάτων με ραδιοκύματα. Πλέον, είναι καθολικά αποδεκτό ότι όλα τα ηλεκτρομαγνητικά φαινόμενα υπόκεινται σε αυτές τις εξισώσεις.

$$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \tag{1.1}$$

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{1.2}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{1.3}$$

$$\vec{\nabla} \times \vec{B} = \mu_0 \left(\vec{J} + \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \right) \tag{1.4}$$

Οι σχέσεις 1.1-1.4 αποτελούν τις εξισώσεις του Maxwell όπως αυτές διατυπώνονται σε διαφορική μορφή, όπου $\vec{E}$ και $\vec{B}$ το ηλεκτρικό και το μαγνητικό πεδίο αντίστοιχα, ε_0 η διηλεκτρική σταθερά του κενού, μ_0 η μαγνητική διαπερατότητα του κενού, ρ η πυκνότητα φορτίου και $\vec{J}$ η πυκνότητα ρεύματος. Η καθοριστική συμβολή του Maxwell ήταν η προσθήκη του όρου ρεύματος μετατόπισης ($\varepsilon_0 \, \partial \vec{E} / \partial t$) στην εξίσωση (1.4), ο οποίος δείχνει ότι ένα μεταβαλλόμενο ηλεκτρικό πεδίο επάγει ένα μαγνητικό πεδίο. Αυτή επέτρεψε την πρόβλεψη της ύπαρξης αυτοσυντηρούμενων κυμάτων που μπορούν να διαδοθούν στο κενό, τα ηλεκτρομαγνητικά κύματα.

Σε περιοχές του χώρου μακριά από πηγές, όπου η πυκνότητα φορτίου και η πυκνότητα ρεύματος μηδενίζεται (δηλαδή στο κενό), οι εξισώσεις του Maxwell οδηγούν σε δύο ομογενείς κυματικές εξισώσεις για το ηλεκτρικό και το μαγνητικό πεδίο:

$$\nabla^2 \vec{E} = \mu_0 \varepsilon_0 \frac{\partial^2 E}{\partial t^2} \qquad \nabla^2 \vec{B} = \mu_0 \varepsilon_0 \frac{\partial^2 B}{\partial t^2} \tag{1.5 \& 1.6}$$

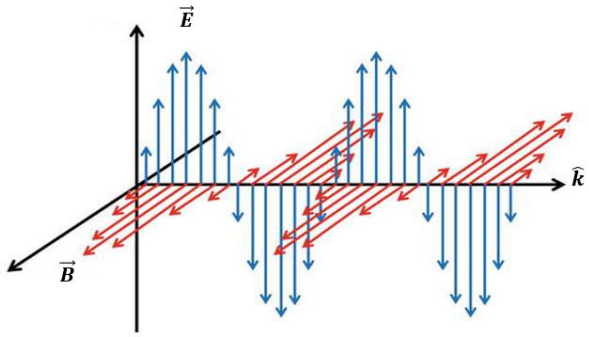
Η λύση τους περιγράφει το κύμα που διαδίδεται σε κάθε περίπτωση με ταχύτητα $\upsilon = 1/\sqrt{\varepsilon_0 \mu_0}$, ίση με την ταχύτητα του φωτός c. Η πιο απλή λύση της κυματικής εξίσωσης είναι το επίπεδο αρμονικό κύμα. Τα διανύσματα του ηλεκτρικού και του μαγνητικού πεδίου είναι

κάθετα μεταξύ τους, καθώς και κάθετα προς τη διεύθυνση διάδοσης του κύματος. Τα φυσικά πεδία σε ένα επίπεδο κύμα περιγράφονται από τις σχέσεις:

$$\vec{E} = E_0 \cos(k \cdot r - \omega \cdot t + \delta)\, \hat{n} \tag{1.7}$$

$$\vec{B} = B_0 \cos(k \cdot r - \omega \cdot t + \delta)\, (\hat{k} \times \hat{n}) \tag{1.8}$$

όπου E_0 και B_0 τα πλάτη των πεδίων που συνδέονται με τη σχέση $B_0 = E_0/c$, $\hat{k}$ το διάνυσμα διάδοσης με μέτρο τον κυματικό αριθμό k και $\hat{n}$ το διάνυσμα πόλωσης.



Σχήμα 1.1: Διάδοση ενός επίπεδου ηλεκτρομαγνητικού κύματος [36]

Στα ηλεκτρομαγνητικά πεδία μεταφέρεται ενέργεια. Ο ρυθμός μεταφοράς της ενέργειας ανά μονάδα χρόνου και ανά μονάδα εμβαδού περιγράφεται από το διάνυσμα Pointing, το οποίο ορίζεται ως το εξωτερικό γινόμενο:

$$\vec{S} = \frac{1}{\mu_0}\left(\vec{E} \times \vec{B}\right) \tag{1.9}$$

Το διάνυσμα Pointing λαμβάνει μονάδες πυκνότητας ισχύος (W/m^2) και η διεύθυνση του ταυτίζεται με τη διεύθυνση διάδοσης του κύματος. Δεδομένου ότι τα πεδία μεταβάλλονται με πολύ υψηλές συχνότητες (π.χ. στην περίπτωση των μικροκυμάτων), οποιαδήποτε μακροσκοπική μέτρηση περικλείει μεγάλο αριθμό κύκλων. Συνεπώς, το χρονικό μέσο της πυκνότητας ισχύος αποτελεί μια πιο χρήσιμη ένδειξη. Η μέση τιμή της ισχύος ανά μονάδα εμβαδού που μεταφέρεται από ένα ηλεκτρομαγνητικό κύμα δίνεται από τη σχέση:

$$I = \langle S \rangle = \frac{1}{2} c \varepsilon_0 E_0^2 \tag{1.10}$$

Η ισχύς της ακτινοβολίας υπολογίζεται ολοκληρώνοντας το διάνυσμα Pointing πάνω σε μία κλειστή επιφάνεια που περιβάλλει την πηγή:

$$P_{rad} = \oint \vec{S} \cdot d\vec{a} \tag{1.11}$$

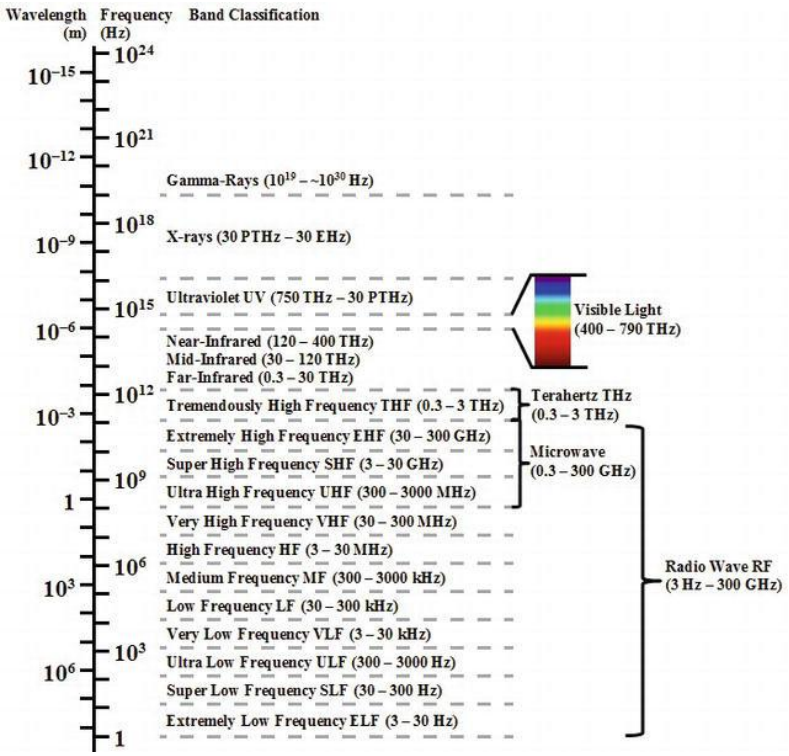
Θεωρώντας μία ισοτροπική πηγή που εκπέμπει ομοιόμορφα προς όλες τις κατευθύνσεις, στο κενό, τα μέτωπα κύματος είναι σφαιρικά. Λόγω της αρχής διατήρησης της ενέργειας, η συνολική ισχύς παραμένει σταθερή καθώς το κύμα απομακρύνεται από την πηγή και η ενέργεια κατανέμεται ομοιόμορφα στην περικλειόμενή σφαιρική επιφάνεια. Συνεπώς, η ένταση της ακτινοβολίας σε μεγάλες ακτινικές αποστάσεις r, όπου τα κύματα θεωρούνται κατά προσέγγιση επίπεδα, είναι:

$$\langle S \rangle = \frac{P_{rad}}{4\pi r^2} \tag{1.12}$$

Προκύπτει, λοιπόν, ότι στην υπόθεση μιας ισοτροπικής πηγής στον ελεύθερο χώρο η πυκνότητα ισχύος των ηλεκτρομαγνητικών κυμάτων μειώνεται ως συνάρτηση του αντίστροφου τετραγώνου της απόστασης – ένα φαινόμενο που αποτελεί τη βάση για τη μελέτη της εξασθένησης διαδρομής στις τηλεπικοινωνιακές ζεύξεις.

1.2 Ηλεκτρομαγνητικό φάσμα

Τα ηλεκτρομαγνητικά κύματα οργανώνονται σε μία συνεχή κατανομή συχνοτήτων, γνωστή ως ηλεκτρομαγνητικό φάσμα. Το ηλεκτρομαγνητικό φάσμα χωρίζεται σε ξεχωριστές ζώνες συχνοτήτων (ή μηκών κύματος), οι οποίες δεν έχουν διακριτά όρια, καθώς ο διαχωρισμός τους βασίζεται στον τρόπο με τον οποίο η ακτινοβολία αλληλοεπιδρά με την ύλη και τις μεθόδους που χρησιμοποιούνται για την παραγωγή και την ανίχνευσης της. Το ηλεκτρομαγνητικό φάσμα εκτείνεται από τα ραδιοκύματα χαμηλής συχνότητας και πολύ μεγάλου μήκους κύματος (της τάξης των ~10^8m), έως τα κύματα πολύ υψηλών συχνοτήτων και μεγάλης ενέργειας των ακτινών γάμμα ($\lambda<10^{-10}$m). Στο σχήμα 1.2 απεικονίζονται όλες οι περιοχές του φάσματος και οι επιμέρους ζώνες τους. Με την αύξηση της συχνότητας, αυξάνεται και η ενέργεια του φωτονίου, με αποτέλεσμα η ακτινοβολία πολύ υψηλών συχνοτήτων (ακτίνες γ και X) να προκαλεί ιονισμό των ατόμων. Η ακτινοβολία αυτή καλείται και ιονίζουσα ακτινοβολία.



Σχήμα 1.2: Το ηλεκτρομαγνητικό φάσμα με τις επιμέρους ζώνες των ραδιοσυχνοτήτων [36]

Στο πλαίσιο των ασύρματων επικοινωνιών, το ενδιαφέρον εστιάζεται στην περιοχή των ραδιοσυχνοτήτων (RF), η οποία συμβατικά ορίζεται από τα 3 kHz έως τα 300 GHz. Η περιοχή αυτή χαρακτηρίζεται από φωτόνια χαμηλής ενέργειας, επομένως πρόκειται για μη-ιονίζουσα ακτινοβολία. Η αλληλεπίδραση των ραδιοκυμάτων με την ύλη εκδηλώνεται μέσω φαινομένων που εξαρτώνται από τις ιδιότητες του υλικού και τα χαρακτηριστικά του κύματος, όπως η απορρόφηση, η ανάκλαση, η διάθλαση, η σκέδαση και η επαγωγή ρεύματος σε αγωγούς.

Οι συχνότητες που εξετάζονται στην παρούσα εργασία ανήκουν στην υποζώνη των μικροκυμάτων, η οποία καλύπτει το εύρος από 300 MHz έως 300 GHz. Η πλειονότητα των σύγχρονων ασύρματων εφαρμογών, όπως τα δίκτυα κινητής τηλεφωνίας, οι δορυφορικές επικοινωνίες, τα ραντάρ, το Wi-Fi και το Bluetooth χρησιμοποιούν τα μικροκύματα. Το βασικότερο πλεονέκτημα έγκειται στο υψηλό διαθέσιμο εύρος ζώνης, το οποίο επιτρέπει υψηλούς ρυθμούς μετάδοσης δεδομένων. Επιπλέον, το μικρό μήκος κύματος καθιστά εφικτή τη χρήση κεραιών μικρών διαστάσεων για την ενσωμάτωση τους σε φορητές συσκευές ή τη δημιουργία κεραιοσυστοιχιών.

1.3 Μηχανισμοί Διάδοσης

Οι τρεις βασικοί μηχανισμοί που επηρεάζουν τη διάδοση του σήματος σε ένα τηλεπικοινωνιακό σύστημα είναι η ανάκλαση, η περίθλαση και η σκέδαση. Όταν το ηλεκτρομαγνητικό κύμα διαδίδεται στον χώρο, η ενέργεια που φτάνει στο δέκτη δεν προέρχεται μόνο μέσω της απευθείας διαδρομής (Line-of-Sight), αλλά αποτελεί το συνολικό αποτέλεσμα των παραπάνω φαινομένων. Στις περιπτώσεις απουσίας οπτικής επαφής πομπού - δέκτη (Non-Line-of-Sight), η συμβολή αυτών των μηχανισμών γίνεται εντονότερη.

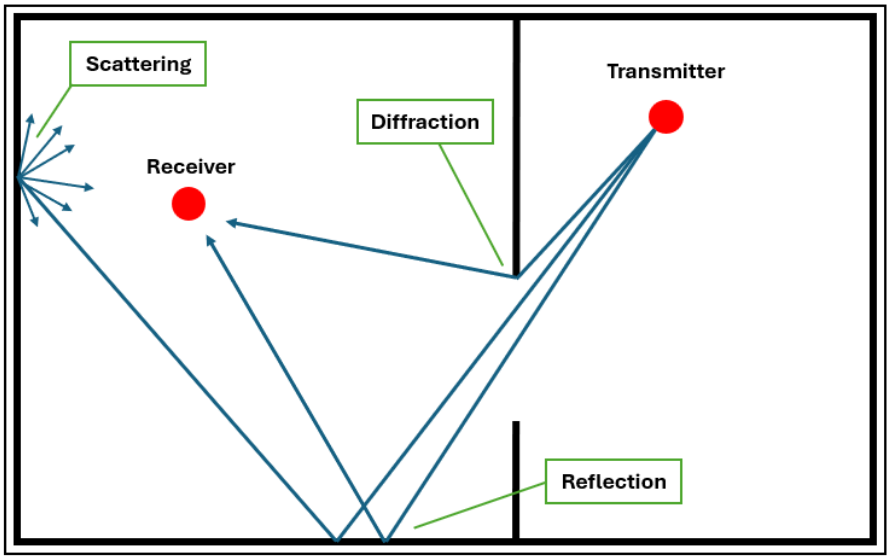
Η ανάκλαση συμβαίνει όταν το ηλεκτρομαγνητικό κύμα που διαδίδεται σε ένα μέσο προσπίπτει σε ένα άλλο μέσο με διαφορετικές ηλεκτρομαγνητικές ιδιότητες. Ένα μέρος της ενέργειας του, συνήθως, μεταδίδεται στο δεύτερο μέσο ή απορροφάται, οδηγώντας σε ένα εξασθενημένο ανακλώμενο κύμα. Εάν το κύμα προσπίπτει σε ένα τέλειο διηλεκτρικό υλικό, δεν υπάρχει απορρόφηση ενέργειας παρά μόνο διαχωρισμός. Αντίθετα, αν το υλικό ανάκλασης πρόκειται για έναν τέλειο αγωγό, όλη η προσπίπτουσα ενέργεια ανακλάται και επιστρέφει στο αρχικό μέσο. Η ένταση και η φάση του ανακλώμενου κύματος εξαρτώνται από τις ιδιότητες του υλικού, τη γωνία πρόσπτωσης, την πόλωση και τη συχνότητα του. Το φαινόμενο της ανάκλασης συναντάται σε διάφορα υλικά του περιβάλλοντα χώρου. Στη θεωρία διάδοσης των ραδιοκυμάτων, ανάκλαση συμβαίνει όταν το κύμα προσπίπτει σε ένα αντικείμενο αρκετά μεγαλύτερων διαστάσεων από το μήκος κύματος του. Στους εσωτερικούς χώρους, οι τοίχοι, τα δάπεδα και τα διάφορα εμπόδια διαμορφώνουν σε μεγάλο βαθμό το φαινόμενο της ανάκλασης.

Η περίθλαση συμβαίνει όταν η διαδρομή ενός διαδιδόμενου ραδιοκύματος εμποδίζεται από ένα αντικείμενο με γωνίες και ακμές. Τα δευτερεύοντα κύματα που προκύπτουν από αυτά τα σημεία της επιφάνειας διαδίδονται προς όλες τις κατευθύνσεις, ακόμα και πίσω από το εμπόδιο (περιοχές σκίασης), προκαλώντας την “κάμψη” του ραδιοκύματος γύρω από αυτό. Σε υψηλές συχνότητες, η περίθλαση εξαρτάται από τη γεωμετρία του αντικειμένου καθώς και από το πλάτος, τη φάση και την πόλωση του προσπίπτοντος κύματος. Το φαινόμενο είναι πιο έντονο όταν το μέγεθος του εμποδίου είναι συγκρίσιμο με το μήκος κύματος. Ραδιοκύματα χαμηλότερων συχνοτήτων παρουσιάζουν συνήθως εντονότερα το φαινόμενο της περίθλασης. Χαρακτηριστικό παράδειγμα, τα ηλεκτρομαγνητικά κύματα της ραδιοφωνίας μπορούν να παρακάμπτουν μεγάλα εμπόδια (π.χ. λόφους) και να διαδίδονται πίσω από αυτά μέσω της περίθλασης, σε αντίθεση με τα ραδιοκύματα μικρότερου μήκους κύματος που χρησιμοποιεί η τηλεφωνία.

Το φαινόμενο της σκέδασης παρατηρείται όταν το ηλεκτρομαγνητικό κύμα προσπίπτει πάνω σε τραχιές επιφάνειες ή σε αντικείμενα με διαστάσεις συγκρίσιμες ή μικρότερες από αυτό το μήκος κύματος. Ως αποτέλεσμα της πρόσκρουσης, η ενέργεια του κύματος διαχέεται προς όλες τις κατευθύνσεις, αποτελώντας μία ακόμη πηγή ακτινοβολίας στο χώρο. Αντικείμενα σκέδασης των ραδιοκυμάτων είναι συνήθως τα φυτά, τα έπιπλα ακόμη και η τραχύτητα ενός τοίχου.

1.4 Πολλαπλές Διαδρομές

Αποτέλεσμα των παραπάνω παραγόντων αποτελεί το φαινόμενο πολλαπλών διαδρομών (Multipath), κατά το οποίο η κεραία λήψης λαμβάνει σήματα που έχουν ακολουθήσει διαφορετικές διαδρομές. Συνεπώς, το τελικό σήμα αποτελεί μία υπέρθεση χρονικά μετατοπισμένων αντιγράφων του αρχικού σήματος, όπου κάθε συνιστώσα χαρακτηρίζεται από το δικό της πλάτος και φάση. Κατά αυτόν τον τρόπο, μπορεί να προκύψουν στο σημείο της λήψης φαινόμενα ενισχυτικής ή αποσβεστικής συμβολής, προκαλώντας σημαντικές διακυμάνσεις (fading) από σημείο σε σημείο του χώρου. Ακόμη και για ένα σταθερό σημείο, η μεταβολή ενός περιβαλλοντικού παράγοντα (π.χ. η μετακίνηση ενός ανθρώπου στον χώρο) μπορεί να επηρεάσει το λαμβανόμενο σήμα. Το πολυδιαδρομικό φαινόμενο έχει πιο έντονη παρουσία σε περιβάλλοντα πυκνής διάταξης, όπως στο εσωτερικό των κτιρίων, και φυσικά όταν δεν υπάρχει κυρίαρχη διαδρομή λόγω οπτικής επαφής πομπού - δέκτη.



Σχήμα 1.3: Το φαινόμενο πολλαπλών διαδρομών (Multipath) σε έναν εσωτερικό χώρο.

1.5 Εξασθένηση Διαδρομής (Path Loss)

Το φαινόμενο της εξασθένησης (ή απωλειών) διαδρομής εμφανίζεται όταν ένα διαδιδόμενο σήμα υφίσταται εξασθένηση ως συνάρτηση της απόστασης που διανύει καθώς και των χαρακτηριστικών του καναλιού που ανήκει. Συγκεκριμένα, ορίζεται ως η μείωση της ισχύος ενός ηλεκτρομαγνητικού κύματος καθώς διαδίδεται στο χώρο. Αυτή η απώλεια μπορεί να είναι αποτέλεσμα ενός συνδυασμού παραγόντων, συμπεριλαμβανομένου της φυσικής εξάπλωσης της ηλεκτρομαγνητικής ακτινοβολίας στο χώρο, του εξασθενημένου σήματος περίθλασης που προκαλείται από φυσικά εμπόδια, της απορρόφησης ενέργειας από ύλες που αλληλεπιδρούν με τα ηλεκτρομαγνητικά κύματα και άλλες επιδράσεις που βασίζονται στη συχνότητα και το περιβάλλον. Γενικότερα, το σήμα που συγκεντρώνεται στην περιοχή λήψης

είναι αποτέλεσμα υπέρθεσης πολλών επιμέρους σημάτων που έχουν ακολουθήσει διαφορετικές διαδρομές (multipath), επομένως παρουσιάζει αποκλίσεις από το αρχικό.

1.6 Μοντέλα Εξασθένησης Διαδρομής

Η εξασθένηση διαδρομής αποτελεί απαραίτητο στοιχείο για το σχεδιασμό του link budget ενός τηλεπικοινωνιακού συστήματος. Στην προσπάθεια πρόβλεψης της, χρησιμοποιούνται διάφορα μοντέλα που μπορεί να βασίζονται σε: (α) ηλεκτρομαγνητική θεωρία, (β) στατιστικές μεθόδους και (γ) πειραματικά δεδομένα. Το βασικότερο μοντέλο διάδοσης περιγράφει την εξασθένηση στο ιδεατό κανάλι ελεύθερου χώρου. Πρόκειται για το μοντέλο εξασθένησης διαδρομής ελεύθερου χώρου (Free Space Path Loss) το οποίο βασίζεται στην εξίσωση μετάδοσης του Friis, η οποία διατυπώθηκε από τον Harald T. Friis (1946). Η σχέση έχει ως εξής:

$$P_r = P_t\, G_t\, G_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{1.1}$$

Όπου:
P_t, **P_r**: η εκπεμπόμενη και η λαμβανόμενη ισχύς
G_t, **G_r**: το κέρδος των κεραιών
d: απόσταση
λ: το μήκος.

Σε λογαριθμική κλίμακα δίνεται από τη σχέση:

$$PL = \frac{P_t}{P_r} = -G_t - G_r - 20\, log_{10}\left(\frac{\lambda}{4\pi d}\right) \tag{1.2}$$

όπου PL η εξασθένηση διαδρομής. Σε πολλές εφαρμογές παραλείπεται το κέρδος των κεραιών από τη σχέση, θεωρώντας ότι είναι ισοτροπικές. Κατά συνέπεια οι σχέσεις (1.1) και (1.2) τροποποιούνται ως ακολούθως:

$$FSPL = \frac{P_t}{P_r} = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{1.3}$$

$$FSPL = 20\, log_{10}(d_m) + 20\, log_{10}(f_{MHz}) + 32.44 \tag{1.4}$$

Το μοντέλο θεωρεί ότι το μέσο διάδοσης είναι το κενό και ο δέκτης βρίσκεται σε τέτοια απόσταση από τον πομπό, όπου το ηλεκτρομαγνητικό κύμα προσεγγίζει πλέον ένα ευθύγραμμο μέτωπο. Επίσης, εξηγεί μόνο τη εξασθένηση που οφείλεται στη γεωμετρική εξάπλωση του κύματος. Η εξάρτηση από το μήκος κύματος οφείλεται στη χρήση της ενεργού επιφάνειας της κεραίας (και κατ' επέκταση του κέρδους της) στην εξίσωση.

Στην πράξη, τα κανάλια επικοινωνίας δε βρίσκονται στο κενό αλλά σε σύνθετα περιβάλλοντα με πολλαπλούς παράγοντες να επηρεάζουν την εξασθένηση των ραδιοκυμάτων. Η ποικιλομορφία που συναντάται στα περιβάλλοντα και η δυναμική μεταβλητότητα τους περιπλέκει τη μοντελοποίηση της διάδοσης, οπότε η χρήση εμπειρικών δεδομένων ή μίας ντετερμινιστικής φόρμουλας δεν αρκεί πάντα. Μοντέλα που περιγράφουν τη διάδοση κυμάτων εντός ενός κτιρίου ή γενικότερα σε αστικές περιοχές, χρησιμοποιούν είτε ακριβείς γεωγραφικές πληροφορίες είτε εισάγουν εργαλεία στατιστικής πρόβλεψης. Ο δεύτερος τύπος προσφέρει μια πιο γενικευμένη μέθοδο μοντελοποίησης, στην οποία ανήκει και το μοντέλο εξασθένησης διαδρομής λογαριθμικής απόστασης (Log-distance Path Loss).

Το μοντέλο λογαριθμικής απόστασης αποτελεί γενίκευση του μοντέλου FSPL και χρησιμοποιείται κυρίως για την πρόβλεψη της διάδοσης σήματος σε εσωτερικούς χώρους, καθώς συνυπολογίζει και τον περιβαλλοντικό παράγοντα. Το μοντέλο λογαριθμικής απόστασης εισάγει την έννοια του εκθέτη εξασθένησης *n*, ο οποίος εξαρτάται από το περιβάλλον. Δίνεται από τη:

$$PL(d) = PL(d_0) + 20\,n\,log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \qquad (1.5)$$

Όπου:
PL(d_0) η εξασθένηση διαδρομής σε μία απόσταση αναφοράς, συνήθως το 1 μέτρο.
n ο εκθέτης εξασθένησης διαδρομής
X_σ μία τυχαία μεταβλητή που ακολουθεί την κανονική κατανομή (Gaussian) με μέση τιμή μηδέν και τυπική απόκλιση σ (dB).

Ως σημείο αναφοράς χρησιμοποιείται είτε ένα αρχικό πειραματικό σημείο, είτε η θεωρητική τιμή FSPL στο 1 μέτρο. Ο εκθέτης εξασθένησης διαδρομής εκφράζει το ρυθμό μεταβολής της εξασθένησης συναρτήσει της απόστασης και είναι χαρακτηριστικός για κάθε περιβάλλον. Η προσθήκη του όρου X_σ γίνεται για να περιγράψει το φαινόμενο της σκίασης (shadowing), δηλαδή την εξασθένηση που οφείλεται σε αντικείμενα μεγάλης κλίμακας (large-scale fading) λόγω περίθλασης και των πολλαπλών διαδρομών που προκύπτουν. Ο εκθέτης *n* και η τυπική απόκλιση «σ» είναι οι δύο βασικές παράμετροι που χαρακτηρίζουν το περιβάλλον διάδοσης. Η τιμή της τυπικής απόκλισης μπορεί να παρουσιάζει μεγάλη διακύμανση από σημείο σε σημείο ενός κτιρίου, με τη βιβλιογραφία να αναφέρει τιμές που κυμαίνονται συνήθως, από 3 dB έως και 10 dB, για τη ζώνη sub-6 GHz [12-14]. Στον Πίνακα 1.1 αναγράφονται τυπικές τιμές του εκθέτη εξασθένησης για διάφορα περιβάλλοντα.

Περιβάλλον	*n*
Ελεύθερος χώρος	2
Εσωτερικός χώρος – LOS	1.6 - 1.8
Εσωτερικός χώρος – NLOS	4 – 6
Αστική περιοχή	2.7 - 3.5
Εργοστάσιο	2 – 3

Πίνακας 1.1: Τιμές του εκθέτη εξασθένησης *n* για ποικίλα περιβάλλοντα [8]

Στο μοντέλο εξασθένησης διαδρομής ελεύθερου χώρου και στο μοντέλο λογαριθμικής απόστασης, η εξασθένηση εκφράζεται συναρτήσει του δεκαδικού λογαρίθμου της απόστασης. Τα δύο αυτά μεγέθη μπορούν να συνδεθούν μέσω μιας γραμμικής σχέσης της μορφής:

$$PL = a\,log_{10}(d) + b \tag{1.6}$$

Στην περίπτωση του μοντέλου ελεύθερου χώρου οι παράμετροι a, b προσδιορίζονται ως:

$$\begin{cases} a = 20 \\ b = 20\,log_{10}\left(\frac{4\pi}{\lambda}\right) \end{cases}$$

Η τιμή της κλίσης είναι σε κάθε περίπτωση σταθερή και ίση με 20. Αντίθετα, η παράμετρος b (ή ο σταθερός όρος της ευθείας) εξαρτάται από τη συχνότητα του διαδιδόμενου σήματος. Για τη σύγκριση του μοντέλου με τα πειραματικά δεδομένα γίνεται προσαρμογή μιας καμπύλης γραμμικής παλινδρόμησης – συνήθως με τη μέθοδο ελαχίστων τετραγώνων - ώστε να βρεθεί η «καλύτερη» ευθεία που περιγράφει τα δεδομένα.

Για τη διερεύνηση του μοντέλου λογαριθμικής απόστασης, ακολουθείται παρόμοια διαδικασία γραμμικής προσαρμογής. Εδώ, χρησιμοποιείται ως εξαρτημένη μεταβλητή ο λόγος d/d_0, ενώ δεν υπάρχει σταθερός όρος b προς εύρεση, καθότι η τιμή $PL(d_0)$ είναι προκαθορισμένη. Επομένως, στη μέθοδο ελαχίστων τετραγώνων η σταθερά $PL(d_0)$ ενσωματώνεται ως σταθερά στην εξαρτημένη μεταβλητή. Ουσιαστικά γίνεται εύρεση της ευθείας $Y = 10n\,X$, όπου $Y = PL - PL(d_0)$ και $X = \log_{10}\left(\frac{d}{d_0}\right)$. Ο υπολογισμός της κλίσης ακολουθεί τη σχέση:

$$10n = \frac{\sum_{i=1}^{N} X_i Y_i}{\sum_{i=1}^{N} X_i^2} \tag{1.7}$$

Ο στατιστικός όρος X_σ μπορεί να προστεθεί στην καμπύλη μετά την προσαρμογή, βάσει της τυπικής απόκλισης (1.8) που θα προκύψει μεταξύ των πειραματικών σημείων και της προσαρμοσμένης καμπύλης.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}\left(PL_{measured} - PL_{fit}\right)^2}{N-1}} \tag{1.8}$$

1.7 Internet of Things

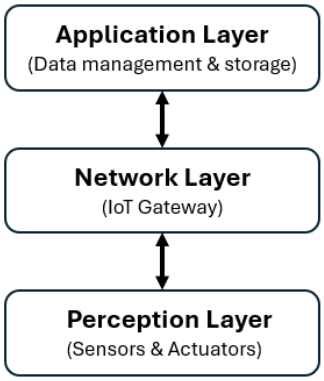
Το «Internet of Things» (IoT) αποτελεί μία έννοια η οποία έχει εδραιωθεί στο χώρο της τεχνολογίας και των υπηρεσιών και αφορά τον τρόπο που αλληλοεπιδρά ο χρήστης με τις συσκευές και οι ίδιες μεταξύ τους. Πρόκειται, για ένα δίκτυο «επικοινωνίας» στο οποίο ένα πλήθος διασυνδεδεμένων αντικειμένων συνεργάζεται με σκοπό τη συλλογή, επεξεργασία, ανάλυση και την διαχείριση δεδομένων με ελάχιστη ή μηδαμινή παρέμβαση του ανθρώπου. Αυτά τα αντικείμενα χαρακτηρίζονται και ως «έξυπνες συσκευές» και αντιστοιχούν σε ενσωματωμένα ηλεκτρονικά συστήματα παντός είδους, εξοπλισμένα με αισθητήρες, επεξεργαστική ισχύ και μονάδες δικτύωσης, ώστε να συλλέγουν και να ανταλλάσσουν δεδομένα σε πραγματικό χρόνο.

Η αποδοτικότητα ενός IoT συστήματος εξαρτάται από την ικανότητα του να συλλέγει και να αναλύει πληροφορίες, οι οποίες συνήθως προέρχονται από τον περιβάλλοντα χώρο. Οι παράγοντες που καθορίζουν την «έξυπνη» λειτουργία μιας συσκευή σχετίζονται τόσο με το λογισμικό την υποστηρίζει όσο και με το δίκτυο που διασφαλίζει τη διακίνηση των δεδομένων και την αλληλεπίδραση με τον χρήστη. Το λογισμικό συμμετέχει ενεργά στην αποτελεσματική διαχείριση και οργάνωση της πληροφορίας, με κύριο στόχο την επίτευξη μιας όσο τον δυνατόν περισσότερο αυτοματοποιημένης λειτουργίας. Παράλληλα, η ιδιαίτερη έμφαση στην επικοινωνία στις IoT εφαρμογές προβλέπει τη σύνδεση σε ένα ευρύτερο δίκτυο αντικειμένων (διαδίκτυο ή τοπικό δίκτυο), γεγονός που επιτρέπει τη μεταφορά και την απομακρυσμένη επεξεργασία των δεδομένων σε άλλες συσκευές.

Ένα τυπικό σύστημα IoT δομείται σε διάφορα επίπεδα τα οποία εξασφαλίζουν τη ροή της πληροφορίας από την ανίχνευση της προς στον τελικό χρήστη. Η πιο κοινή αρχιτεκτονική είναι αυτή των τριών επιπέδων [1]:

- **Επίπεδο αντίληψης** (Perception Layer): Περιλαμβάνει το hardware (π.χ. αισθητήρες) που αλληλοεπιδρά με το περιβάλλον και συλλέγει την πληροφορία.
- **Επίπεδο Δικτύου** (Network Layer): Αναλαμβάνει τη διαβίβαση των δεδομένων και τη διασύνδεση των αντικειμένων μέσω ασύρματων ή ενσύρματων τεχνολογιών

- **Επίπεδο Εφαρμογής** (Application Layer): Είναι υπεύθυνο για την παροχή εφαρμογών και υπηρεσιών που περιλαμβάνουν τη συλλογή, την ανάλυση και την οπτικοποίηση της πληροφορίας, ανάλογα τις ανάγκες του χρήστη.



Σχήμα 1.4: Η συμβατική δομή τριών επιπέδων ενός IoT συστήματος

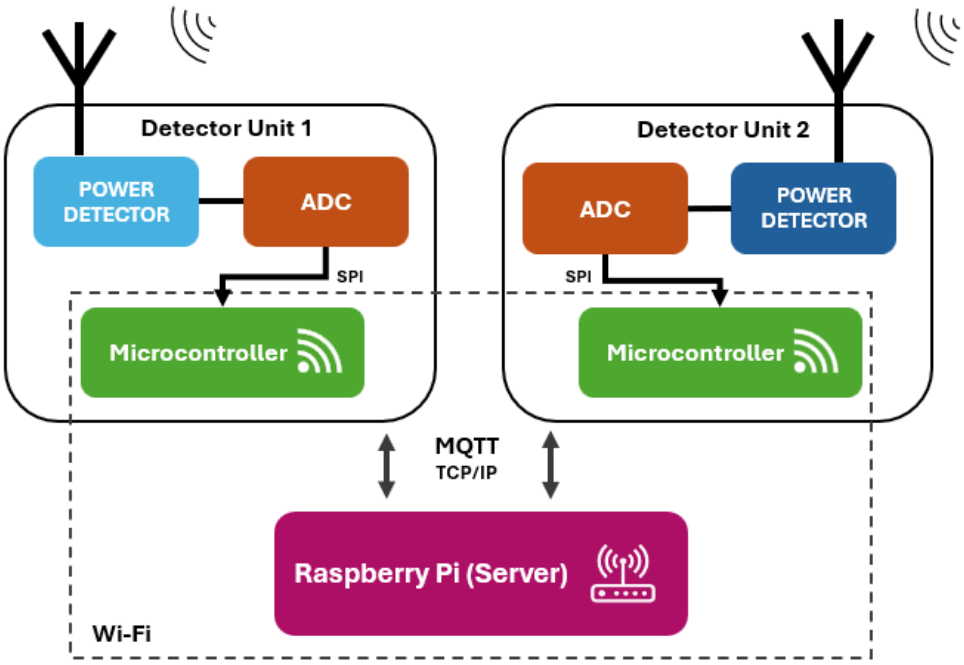
1.8 Εφαρμογές Διαδικτύου

Σε αντίθεση με τα παραδοσιακά λογισμικά που εκτελούνται τοπικά στη συσκευή χρήστη, οι εφαρμογές διαδικτύου φιλοξενούνται σε απομακρυσμένους διακομιστές (servers) και είναι προσβάσιμες μέσω προγραμμάτων περιήγησης (browsers). Αντιπροσωπεύουν ένα γενικότερο τεχνολογικό ρεύμα στον τρόπο σχεδίασης και διάθεσης λογισμικού, μεταφέροντας την υπολογιστική λογική από μία συσκευή σε κατανεμημένες υποδομές δικτύου. Η δομή μιας σύγχρονης εφαρμογής διαδικτύου βασίζεται στο μοντέλο πελάτη-εξυπηρετητή (client/server), όπου ο browser χρησιμοποιείται ως το πρόγραμμα πελάτη εκτελώντας μία διαδραστική υπηρεσία σε συνεργασία με τους συνδεδεμένους servers, μέσω του διαδικτύου ή ενός περιορισμένου δικτύου. Ουσιαστικά, το σύστημα διακρίνεται σε δύο οντότητες, με την μία να αφορά το πυρήνα της εφαρμογής, με σενάρια και υπηρεσίες που εκτελούνται στην πλευρά του διακομιστή (διαχείριση της πληροφορίας από και προς τον χρήστη, αλληλεπίδραση με βάσεις δεδομένων) και την άλλη να αφορά τη λογική που εκτελείται στον πελάτη και περιλαμβάνει τα μέσα αλληλεπίδρασης του χρήστη με την εφαρμογή (User Interface). Η διασύνδεση των δύο στηρίζεται, συνήθως, στο πρωτόκολλο επικοινωνίας του διαδικτύου, το HTTP, το οποίο ακολουθεί το μοντέλο αίτησης-απόκρισης. Ενώ μία απλή ιστοσελίδα παρέχει το περιεχόμενο από στατικά αρχεία, μία εφαρμογή διαδικτύου παρουσιάζει το περιεχόμενο δυναμικά, βάσει τα αιτήματα και τις παραμέτρους που θέτει ο χρήστης και της καταγραφόμενης συμπεριφοράς του.

Κεφάλαιο 2: Σχεδίαση & Υλοποίηση των μονάδων ανιχνευτών ισχύος

2.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφεται το hardware που χρησιμοποιήθηκε για τη λήψη μετρήσεων ισχύος σήματος. Το σύστημα αποτελείται από δύο μονάδες μέτρησης, τη γεννήτρια RF σήματος καθώς και τις κεραίες λήψης και εκπομπής. Οι δύο μονάδες είναι πανομοιότυπες μεταξύ τους, με μόνη διαφορά τον ανιχνευτή ισχύος (power detector) που ενσωματώνουν. Κύρια λειτουργία των μονάδων είναι η πραγματοποίηση μετρήσεων ισχύος του RF σήματος και η αποστολή τους μέσω τοπικού δικτύου Wi-Fi στην κεντρική μονάδα του συστήματος, το Raspberry Pi. Στόχος της παρούσας μεταπτυχιακής διπλωματικής εργασίας είναι η υλοποίηση δύο ανεξάρτητων ηλεκτρονικών τυπομένων κυκλωμάτων μικρών διαστάσεων, με ενσωματωμένο μικροελεγκτή για τη λήψη και τη μετάδοση των δεδομένων από τον ανιχνευτή ισχύος. Στο παρακάτω σχήμα 2.1 απεικονίζεται το σχηματικό διάγραμμα της διάταξης του συστήματος.



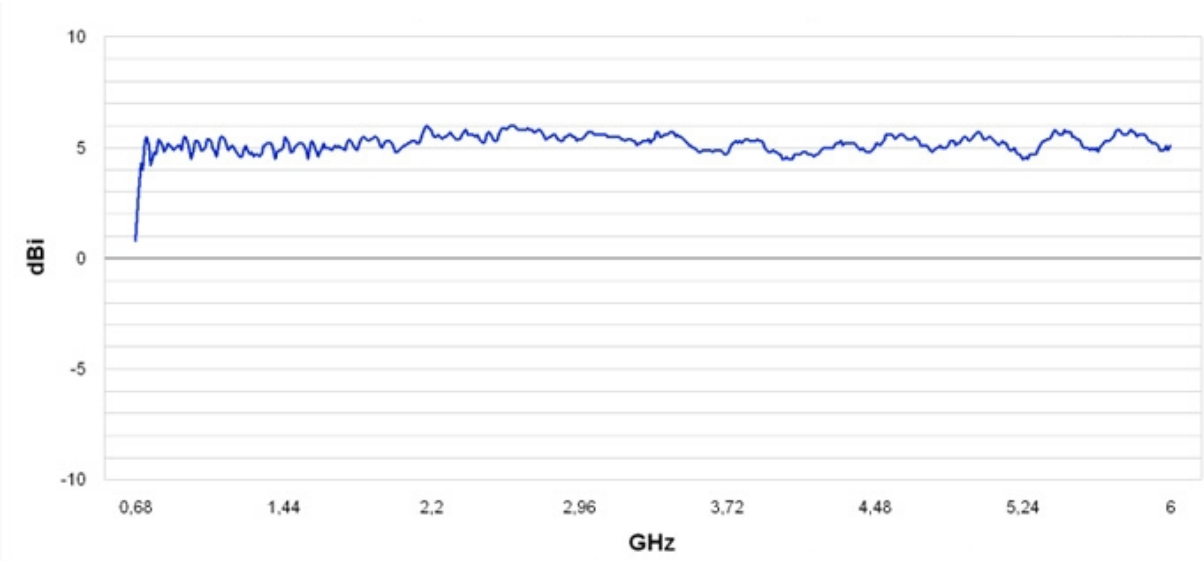
Σχήμα 2.1: Σχηματικό διάγραμμα με τα μέρη που συμμετέχουν στη λήψη μετρήσεων.

2.2 Η κεραία HyperLOG 7060

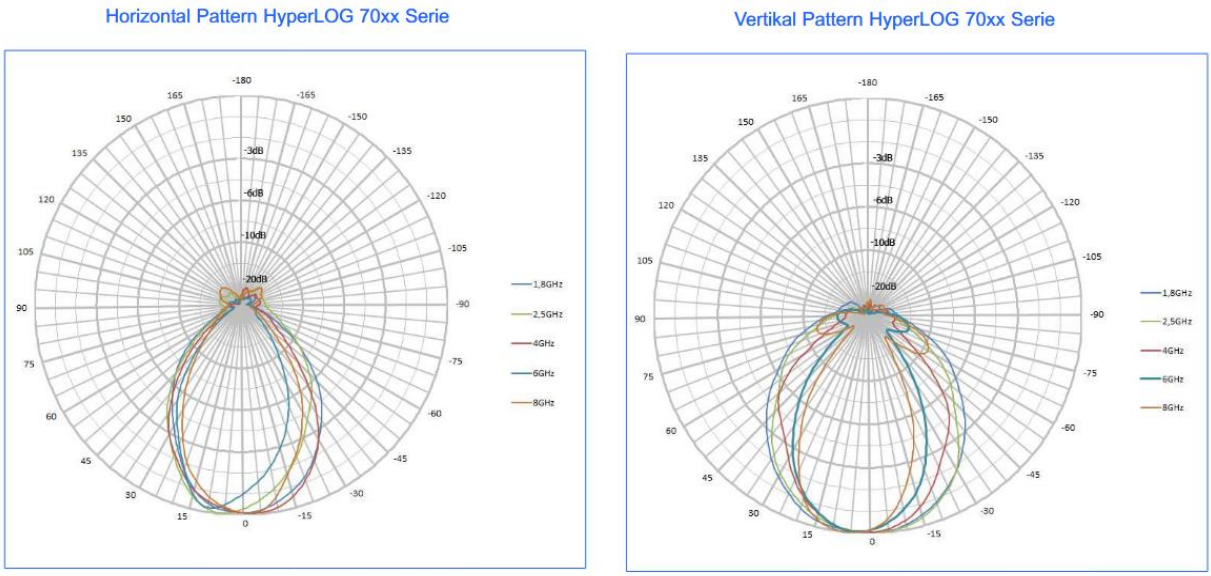
Για τη λήψη και εκπομπή του σήματος χρησιμοποιήθηκαν οι κεραίες HyperLOG 7060 της γερμανικής εταιρείας Aaronia. Η κεραία HyperLOG 7060 είναι κατευθυντική κεραία ευρείας ζώνης. Ειδικότερα ακολουθεί τη μορφή λογαριθμικά περιοδικής διπολικής διάταξης (Log Periodic Dipole Array), αποτελούμενη από έναν αριθμό στοιχείων διπόλου τοποθετημένα παράλληλα μεταξύ τους και σε απόσταση που ακολουθεί μία λογαριθμική περιοδικότητα.

Σχήμα 2.2: Η κεραία HyperLOG 7060 [29].

Το μοντέλο HyperLOG 7060 καλύπτει το εύρος συχνοτήτων από 700 έως 6000 MHz. Πρόκειται για παθητική κεραία, δηλαδή η λειτουργία της βασίζεται στη φυσική της απολαβή, η οποία κυμαίνεται περίπου στα 5dBi. Το διάγραμμα κέρδους του σχήματος 2.3 αποτυπώνει το μετρούμενο κέρδος της κεραίας συναρτήσει της συχνότητας. Η εμπέδηση της είναι 50 Ω, ενώ ο λόγος στάσιμου κύματος τάσης (VSWR) διατηρείται σε τιμές μικρότερες του 2. Στα διαγράμματα (Σχήμα 2.4) αποτυπώνεται η κατανομή της ακτινοβολίας της κεραίας στο οριζόντιο (αζιμουθιακό) και κάθετο επίπεδο (επίπεδο ανύψωσης) για διάφορες συχνότητες. Είναι εμφανές ότι η ένταση της ακτινοβολίας παρουσιάζει μεταβολές κατά μήκος των συχνοτήτων, με σημαντικότερη διαπίστωση ότι ο κύριος λοβός ακτινοβολίας είναι στενότερος σε μεγαλύτερες συχνότητες.



Σχήμα 2.3: Το διάγραμμα κέρδους της κεραίας συναρτήσει της συχνότητας λειτουργίας [29].



Σχήμα 2.4: Διάγραμμα ακτινοβολίας της κεραίας στο αζιμουθιακό επίπεδο (φ) (αριστερά) και στο πολικό επίπεδο (θ) (δεξιά) [30].

Η κεραία διατίθεται σε μία προστατευτική θήκη (ραδιοθόλος), η οποία έχει αμελητέα επιρροή στις μετρήσεις και επιτρέπει τη χρήση της με περιφερειακά στηρίγματα όπως χειρολαβή ή τρίποδο. Δεδομένου ότι η κεραία είναι γραμμικά πολωμένη, με τη χρήση της βάσης τριπόδου μπορεί εύκολα να μεταβληθεί η πόλωση, με περιστροφή της κεραίας κατά 90 μοίρες. Οι διαστάσεις της κεραίας είναι 340 x 200 x 25 mm.

2.3 Γεννήτρια Σήματος Synth USB3

Το SynthUSB3, της εταιρείας Windfreak Technologies, είναι μία χαμηλού κόστους γεννήτρια RF σήματος, στο μέγεθος ενός USB στικ. Καλύπτει συχνότητες από 12.5 MHz έως 6 GHz με ανάλυση 0.01 Hz και μπορεί να αποδώσει ισχύ σήματος έως +8 dBm, ανάλογα με τη συχνότητα. Η συσκευή τροφοδοτείται μέσω της θύρας USB, από την οποία πραγματοποιείται και ο προγραμματισμός της, με τη χρήση του ανοικτού κώδικα λογισμικού LabView GUI που το συνοδεύει. Μέσω αυτού μπορεί να ρυθμιστεί η συχνότητα και η ισχύς ενός σταθερού σήματος, να επιλεγεί η συσκευή να κάνει “frequency sweep”, δηλαδή να παράγει ένα σήμα που σαρώνει ένα εύρος συχνοτήτων και τέλος να γίνει διαμόρφωση πλάτους ή συχνότητας.

Σχήμα 2.5: Η γεννήτρια RF σήματος SynthUSB3 [37].

2.4 Ο μικροελεγκτής ESP32

Οι ESP32 είναι μια οικογένεια χαμηλού κόστους και κατανάλωσης μικροελεγκτών, οι οποίοι συνδυάζουν δυνατότητες Wi-Fi και Bluetooth, κάτι που τους καθιστά ιδανικούς για εφαρμογές IoT. Η σχεδίαση της σειράς ESP32 γίνεται από την κινεζική εταιρία Espressif Systems, ενώ την παραγωγή αναλαμβάνει η εταιρία TSMC. Ο μικροελεγκτής διατίθεται, συνήθως, ενσωματωμένος σε αναπτυξιακές πλακέτες (development boards) με διαμορφώσεις που ποικίλουν ανάλογα το μοντέλο και τον κατασκευαστή.

Οι μικροελεγκτές ESP32 διαθέτουν μικροεπεξεργαστές Xtensa (32-bit) LX6 ή LX7 σε εκδόσεις με μονό ή διπλό πυρήνα και συχνότητα ρολογιού έως 240MHz. Επιπλέον, ενσωματώνουν στοιχεία απαραίτητα για την ασύρματη επικοινωνία όπως διακόπτες κεραίας, ενισχυτή ισχύος, φίλτρα, ενισχυτή λήψης χαμηλού θορύβου κ.α. Για την ευκολότερη αξιοποίηση τους, η Espressif Systems παρέχει ολοκληρωμένα modules τα οποία περιλαμβάνουν το ESP32 SoC, εξωτερική μνήμη flash και PCB κεραία. Στα πλαίσια της μεταπτυχιακής εργασίας χρησιμοποιήθηκε η μονάδα ESP32-WROOM-32, η οποία ενσωματώνει διπύρηνο επεξεργαστή Xtensa LX6, Wi-Fi 802.11 b/g/n και Bluetooth 4.2. Στον παρακάτω πίνακα αναγράφονται τα βασικά χαρακτηριστικά του ESP32-WROOM-32 module:

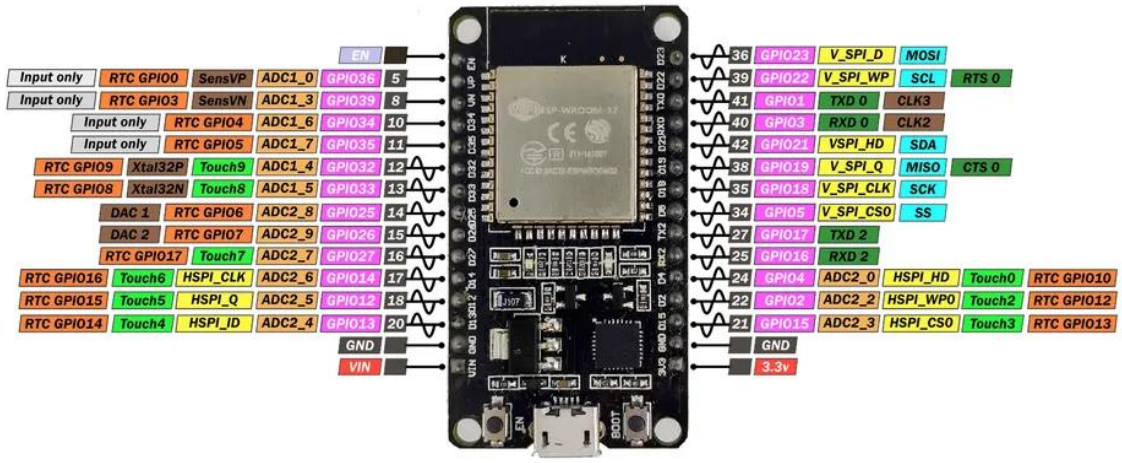
Πίνακας 2.1: Τα τεχνικά χαρακτηριστικά του μικροελεγκτή ESP32-WROOM32 [28].

CPU & SoC memory	
Microprocessor	Xtensa dual-core 32-bit LX6, up to 240 MHz
ROM	448 KB
SRAM	520 KB

Wi-Fi	
Protocols	802.11b/g/n
Bit rate	up to 150 Mbps
Frequency range	2412~2484 MHz
Bluetooth	
Protocols	V4.2 BR/EDR & Bluetooth Low Energy
Hardware	
Pins & GPIOs	38 pins, up to 32 GPIOs
Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR
On-board clock	40 MHz crystal
Operating voltage	2.7 ~ 3.6 V
Operating current	Average: 80 mA
Minimum current supplied	500 mA
Software	
Wi-Fi mode	Station/SoftAP/SoftAP+Station/P2P
Wi-Fi Security	WPA/WPA2/WPA2-Enterprise/WPS
Network Protocols	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT

2.5 Η αναπτυξιακή πλακέτα DOIT ESP32 DEVKIT V1

Στα πλαίσια της εργασίας χρησιμοποιήθηκε η αναπτυξιακή πλακέτα ESP32-DEVKIT-V1 της εταιρείας DOIT, στην έκδοση με τα 30 GPIOs, η οποία φέρει τη μονάδα ESP32-WROOM-32. Όσον αφορά τις διεπαφές, η πλακέτα διαθέτει δύο ακιδοσειρές (pin headers) που αντιστοιχούν στους ακροδέκτες της μονάδας ESP32 και μια θύρα USB-to-UART, βασισμένη στο τσιπ CP2102, για τον προγραμματισμό και τη τροφοδοσία του μικροελεγκτή. Εναλλακτικά, η τροφοδοσία μπορεί να γίνει είτε από τον ακροδέκτη “Vin”, εφαρμόζοντας τάση 5V, όπου ένας ρυθμιστής τάσης φροντίζει να παρέχει τα 3.3V που απαιτεί η μονάδα, είτε από τον ακροδέκτη “3V3” που συνδέεται άμεσα με αυτήν.



Σχήμα 2.6: Η αναπτυξιακή πλακέτα ESP32-DEVKIT-V1 και οι διεπαφές της [38].

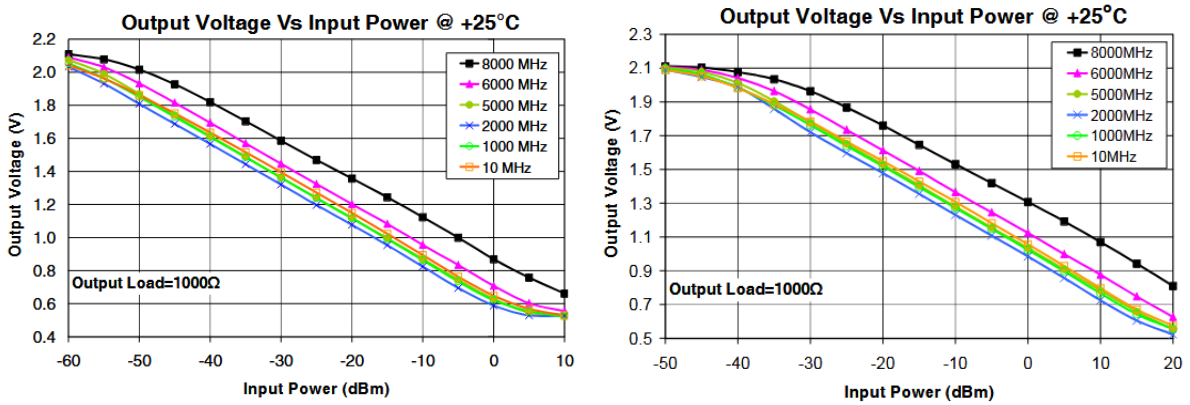
2.6 Mini-Circuits ZX47 Ανιχνευτές Ισχύος

Η σειρά ZX47, της εταιρείας Mini-Circuits, είναι μία οικογένεια μονάδων ανιχνευτών ισχύος που μετατρέπουν την ισχύ ενός σήματος στο φάσμα ραδιοσυχνοτήτων, σε μία DC τάση εξόδου. Πρόκειται για ανιχνευτές που καλύπτουν μία ευρεία περιοχή των ραδιοσυχνοτήτων, από τα 10 MHz έως τα 8000 MHz της μικροκυματικής ζώνης. Επιτρέπουν μετρήσεις σε ένα δυναμικό εύρος ~60dBm με όρια που ποικίλουν ανά μοντέλο, κατά μήκος του οποίου παρουσιάζουν υψηλή γραμμικότητα στη διαδικασία μετατροπής. Βασικό χαρακτηριστικό προς την επιλογή τους είναι η εύκολη ενσωμάτωση και χρήση τους, καθώς ο ανιχνευτής ισχύος είναι ήδη διαμορφωμένος σε μία συμπαγή και στιβαρή μονάδα, με ενσωματωμένο αισθητήρα θερμοκρασίας, απαιτώντας μόνο τροφοδοσία για τη λειτουργία της. Στο μεταλλικό τους περίβλημα βρίσκεται υπό τη μορφή SMA συνδέσμων (εμπέδησης 50 Ω) η είσοδος σήματος (RF IN) και η έξοδος DC τάσης (DC OUT), οι ακροδέκτες για την τροφοδοσία και ένας ακροδέκτης (DC τάσης) για την ανάγνωση της θερμοκρασίας.

Σχήμα 2.7: Ο ανιχνευτής ισχύος ZX47-60LN-S+. Όλα τα μοντέλα της σειράς ZX47 χρησιμοποιούν το ίδιο περίβλημα [39].

Τα μοντέλα που χρησιμοποιήθηκαν στην εργασία είναι οι ανιχνευτές ισχύος ZX47-60LN-S+ και ZX47-40LN-S+. Και οι δύο μονάδες αποτελούν μέρος των "LN" εκδόσεων της οικογένειας, που δίνουν έμφαση στη διατήρηση χαμηλού θορύβου (Low Noise) κατά την έξοδο - ο κατασκευαστής δίνει τυπική τιμή ripple ~20 mVp-p στα 10 MHz - με τίμημα πιο αργούς χρόνους παλμικής απόκρισης (pulse response). Η ειδοποιός διαφορά τους έγκειται στο δυναμικό εύρος τους, δηλαδή στα επίπεδα μέτρησης ισχύος που επιτρέπουν. Ο ZX47-60LN-S+ πρόκειται για τον πιο «ευαίσθητο» ανιχνευτή της σειράς με περιοχή λειτουργίας σημάτων ισχύος από -60dBm έως +5dBm. Αντίθετα, ο ZX47-40LN-S+ είναι σχεδιασμένος να ανιχνεύει υψηλότερα επίπεδα ισχύος σήματος, συγκεκριμένα στην περιοχή -40 έως +20 dBm. Φυσικά και στους δύο ανιχνευτές τα επίπεδα ανοχής ενός σήματος στην είσοδο είναι αρκετά μεγαλύτερα από το μέγιστο του δυναμικού τους εύρους (+15dBm και +27dBm αντίστοιχα). Για τη λειτουργία τους απαιτούν σταθερή τάση +5V (4.5 - 5.5V), ενώ η κατανάλωση ρεύματος κυμαίνεται στα ~100 mA.

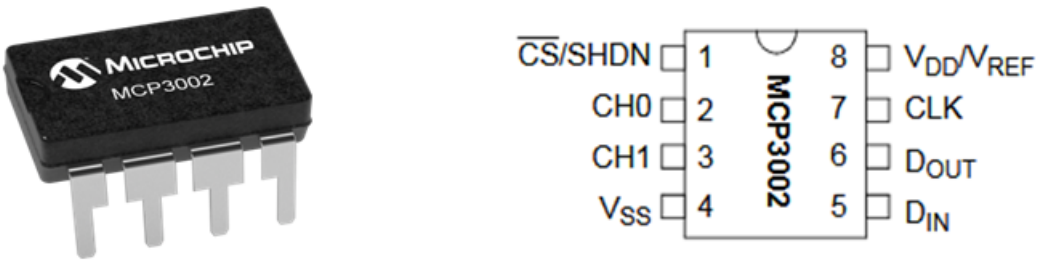
Στο μεγαλύτερο μέρος της περιοχής ανίχνευσης, οι ανιχνευτές ZX47 είναι γραμμικοί, δηλαδή η τάση εξόδου εκφράζεται συναρτήσει της ισχύος σήματος εισόδου με μία γραμμική σχέση. Πρόκειται για μια σχέση βαθμονόμησης και είναι απαραίτητη για την μετατροπή των τιμών τάσης που διαβάζει ο χρήστης, στην ισχύ σήματος που λαμβάνεται στην είσοδο του ανιχνευτή. Από τα διαγράμματα της καμπύλης μεταφοράς που παρέχονται στο φύλλο προδιαγραφών των ανιχνευτών, παρατηρείται ότι η γραμμική σχέση έχει σταθερή αρνητική κλίση, η οποία όμως, μεταβάλλεται (ελαχιστοποιείται) στα άκρα του δυναμικού εύρους. Επομένως, για την αξιόπιστη χρήση του ανιχνευτή, το μετρούμενο σήμα πρέπει να εμπίπτει στα επίπεδα ισχύος όπου η σχέση παραμένει γραμμική. Επίσης, μεταξύ των συχνοτήτων παρουσιάζονται διαφορές ως προς την καμπύλη, κυρίως στον σταθερό όρο αυτής - η κλίση ως επί τω πλείστων διατηρείται σταθερή. Για τη βέλτιστη ακρίβεια στον υπολογισμό της ισχύος σήματος, λοιπόν, πρέπει να χρησιμοποιείται διαφορετική σχέση βαθμονόμησης ανά συχνότητα. Ο κατασκευαστής παρέχει ήδη κάποιες μετρήσεις τάσης συναρτήσει της ισχύος, σε διάφορες συχνότητες, οι οποίες βρέθηκαν σύμφωνες με μετρήσεις που έγιναν στο εργαστήριο, επομένως χρησιμοποιήθηκαν οι καμπύλες του φύλλου προδιαγραφών. Για οποιαδήποτε άλλη συχνότητα, ωστόσο, που χρησιμοποιήθηκε έγιναν εκ νέου μετρήσεις βαθμονόμησης.



Σχήμα 2.8: Γράφημα της εξάρτησης της τάσης εξόδου από την ισχύ σήματος στην είσοδο για τον ανιχνευτή ZX47-60LN-S+ (**αριστερά**) και τον ZX47-40LN-S+ (**δεξιά**) [32][33].

2.7 MCP3002 Analog-to-Digital Converter

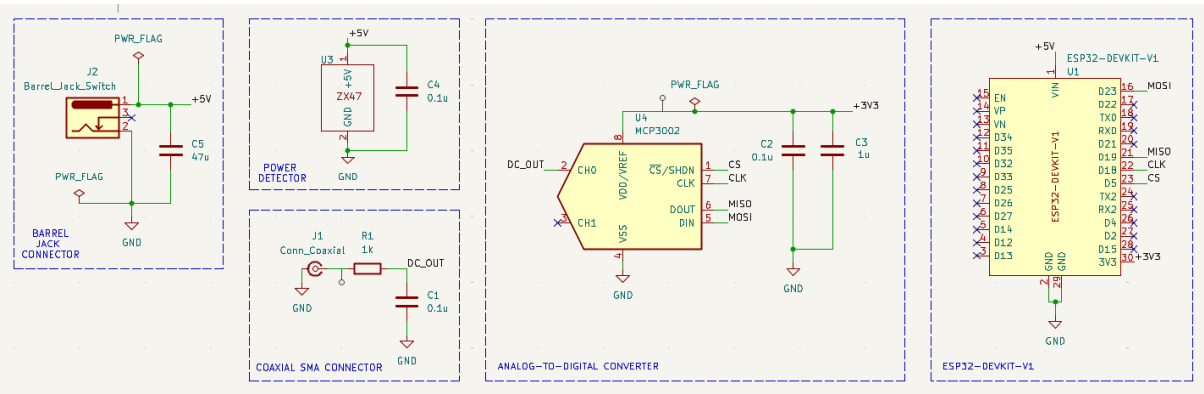
Ο MCP3002, της εταιρείας Microchip, είναι ένας 10-bit μετατροπέας αναλογικού σήματος σε ψηφιακό με ενσωματωμένο κύκλωμα δειγματοληψίας και συγκράτησης. Βασικότερο χαρακτηριστικό είναι ότι συνδυάζει υψηλή απόδοση και χαμηλή κατανάλωση ενέργειας καθιστώντας τον ιδανικό σε εφαρμογές ενσωματωμένων συστημάτων. Βασίζεται στην αρχιτεκτονική SAR (Successive Approximation Register) για τη γρήγορη μετατροπή του σήματος σε ψηφιακό, ενώ χρησιμοποιεί διεπαφή SPI για την επικοινωνία του (π.χ. με έναν μικροελεγκτή). Ο MCP3002 διαθέτει δύο κανάλια εισόδου αναλογικού σήματος, που μπορούν να διαμορφωθούν ως ανεξάρτητα μεταξύ τους ή ως ένα ψευδο-διαφορικό ζεύγος. Μπορεί να λειτουργήσει σε ένα εύρος τάσεων, από 2.7V έως 5.5V, με τη κατανάλωση ρεύματος σε ενεργή κατάσταση να αγγίζει τα 375 μA . Πρόκειται για έναν ADC ταχείας μετατροπής που μπορεί να φτάσει τα 200 ksps (kilo-samples per second) στα 5V ή τα 75ksps στα 2.7V.



Σχήμα 2.9: Ο ADC MCP3002 και οι διεπαφές του [35].

2.8 Διασύνδεση κυκλώματος - Σχεδίαση πλακετών

Κάθε πλακέτα ενσωματώνει μία αναπτυξιακή πλακέτα μικροελεγκτή, τη ESP32-DEVKIT-V1, έναν ανιχνευτή ισχύος της σειράς ZX47 και έναν μετατροπέα MCP3002 για τη μετατροπή του αναλογικού σήματος του ανιχνευτή σε ψηφιακό πριν την ανάγνωσή του από τον μικροελεγκτή. Για τη σχεδίαση της πλακέτας χρησιμοποιήθηκε το σχεδιαστικό πακέτο KiCad 9.0, όπου αρχικά δημιουργήθηκε το σχηματικό του κυκλώματος και στη συνέχεια βάσει αυτού το σχέδιο της PCB.



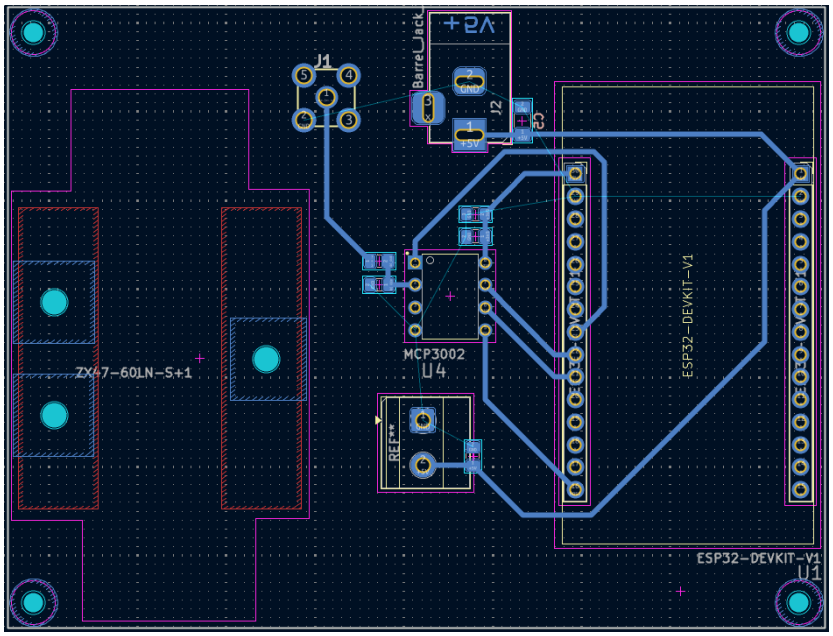
Σχήμα 2.10: Το σχηματικό του κυκλώματος της πλακέτας στο KiCad.

Ο ανιχνευτής ισχύος ZX47 απαιτεί τροφοδοσία 5V για τη λειτουργία του όπως και η πλακέτα του μικροελεγκτή ESP32, οπότε χρησιμοποιείται κοινή δίαυλος τροφοδοσίας συνδέοντας τον ανιχνευτή με την “+5V” υποδοχή της πλακέτας ESP. Αντίθετα, ο μετατροπέας MCP3002 χρειάζεται τροφοδοσία 3.3V την οποία λαμβάνει από τον “3V3” ακροδέκτη του ESP. Επομένως η πλήρης τροφοδοσία της πλακέτας μπορεί να γίνει είτε μέσω της USB θύρας του ESP, είτε μέσω εξωτερικής θύρας συνδεδεμένη με τον “5V” δίαυλο. Ως εξωτερική θύρα τροφοδοσίας χρησιμοποιείται ένα βύσμα jack 2.5 mm, συνοδευόμενο από έναν πυκνωτή με σχετικά μεγάλη χωρητικότητα (47μF) για να διατηρεί σταθερή τη τάση. Η μονάδα του ανιχνευτή διαθέτει εξωτερικούς ακροδέκτες (turret terminals) πάνω στο μεταλλικό του περίβλημα, οπότε η προσθήκη μιας κλέμας 2 υποδοχών βοηθά τη σύνδεση τους με το υπόλοιπο κύκλωμα της pcb.

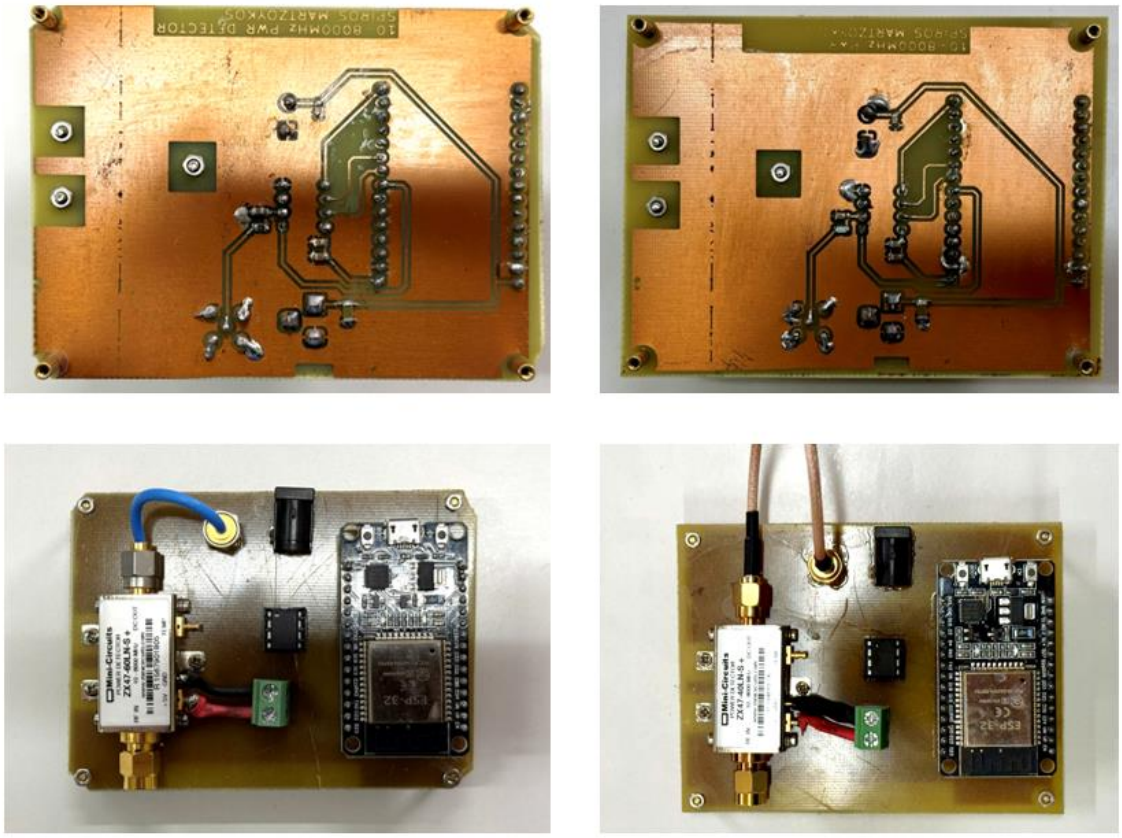
Η έξοδος του ανιχνευτή (DC OUT) είναι θύρα τύπου SMA (θηλυκό) για ομοαξονικά καλώδια, εμπέδησης 50 Ω. Για τη σύνδεση με τη πλακέτα χρησιμοποιείται ένας αντίστοιχος SMA σύνδεσμος, ο οποίος διαθέτει ακίδες (4 γείωσης & 1 σήματος) για through hole υλοποιήσεις σε pcb. Τους δύο παραπάνω συνδέσμους ενώνει ένα ομοαξονικό καλώδιο (male-to-male) μικρού μήκους. Μετά τον σύνδεσμο το σήμα οδηγείται στο πρώτο κανάλι (channel 0) του ADC MCP3002. Πριν την είσοδο αυτού υπάρχει ένα RC φίλτρο, αποτελούμενο από μία αντίσταση 1 kΩ και έναν πυκνωτή 0.1 μF, για να αποκόπτει τυχόν θόρυβο συνοδεύει το σήμα DC τάσης από τον ανιχνευτή.

Η υπόλοιπη συνδεσμολογία του ADC περιλαμβάνει τη τροφοδοσία του, η οποία, όπως αναφέρθηκε, πραγματοποιείται μέσω της εξόδου 3.3V της πλακέτας ESP. Παράλληλα στη γραμμή τροφοδοσίας τοποθετήθηκαν δύο πυκνωτές απόζευξης, χωρητικότητας 0.1 μF και 1 μF, διασφαλίζοντας τη σταθερότητα της παρεχόμενης τάσης. Για την επικοινωνία του μετατροπέα με τον μικροελεγκτή μέσω του υποστηριζόμενου SPI πρωτοκόλλου οι διεπαφές CS, CLK, DOUT, DIN συνδέονται απευθείας με τις αντίστοιχες διεπαφές του ESP32. Πρόκειται για τον κύριο δίαυλο SPI που χρησιμοποιούν οι περισσότερες βιβλιοθήκες και βρίσκεται τους ακροδέκτες D5, D18, D19, D23 (CS, CLK, MISO, MOSI). Οι υπολειπόμενοι δύο ακροδέκτες του MCP3002 γειώνονται, καθότι ο ένας πρόκειται για τη γείωση της μονάδας και ο άλλος για το αχρησιμοποίητο δεύτερο κανάλι του μετατροπέα.

Ο ADC MCP3002 που χρησιμοποιείται, όπως και η πλακέτα ESP32-DEVKIT-V1 διαθέτουν ακίδες στους διεπαφές, επομένως η τοποθέτηση τους απαιτεί through hole υλοποίηση. Αντί για απευθείας συγκόλληση των μονάδων πάνω στην πλακέτα, τοποθετήθηκαν ακιδοσειρές με θηλυκές υποδοχές ώστε να «κουμπώσουν» οι μονάδες πάνω σε αυτές. Όσον αφορά τη τοποθέτηση του ανιχνευτή πάνω στην pcb, αυτή γίνεται μέσω των οπών που διαθέτει η μεταλλική θήκη του. Αντίστοιχες οπές διαμέτρου 2.7 mm έγιναν στην πλακέτα, πάνω στην οποία η μονάδα σταθεροποιήθηκε με βίδες. Τέλος, οι πυκνωτές και αντιστάσεις που χρησιμοποιούνται είναι τύπου SMD (surface-mounted devices). Στο σχήμα 2.11 απεικονίζεται το τελικό σχέδιο της πλακέτας, όπως αναπτύχθηκε στον “PCB Editor” του KiCad. Η διασύνδεση των στοιχείων έγινε στη μία όψη της πλακέτας, ενώ στην άλλη τοποθετήθηκαν όλα τα “through hole” στοιχεία.



Σχήμα 2.11: Το τελικό σχέδιο της πλακέτας στον pcb editor του KiCad.



Σχήμα 2.12: Εμπρός και πίσω όψη της πλακέτας του ανιχνευτή ZX47-60LN-S+ (αριστερά) και του ZX4740LN-S+ (δεξιά)

2.9 Το λογισμικό του ESP32

Τελευταίο στάδιο στην υλοποίηση των πλακετών μέτρησης είναι η ανάπτυξη του firmware για τον μικροελεγκτή ESP32. Αυτή έγινε στο περιβάλλον Arduino IDE – το επίσημο περιβάλλον ανάπτυξης λογισμικού για μικροελεγκτές Arduino. Ωστόσο, λόγω της open-source φύσης του, επιτρέπει το προγραμματισμό πλακετών από τρίτους με τη χρήση των κατάλληλων βιβλιοθηκών. Το Arduino IDE είναι φιλικό προς τον χρήστη προσφέροντας αμέτρητες βιβλιοθήκες προς δοκιμή και εύκολη διαχείριση των πλακετών. Η ανάπτυξη του προγράμματος έγινε στη γλώσσα C++. Βασική αρμοδιότητα του ESP32 είναι να λαμβάνει δείγματα από τον εξωτερικό ADC, να μετατρέπει τη ψηφιακή τιμή σε τιμή τάσης και να την αποστέλλει, μέσω Wi-Fi, στο Raspberry Pi. Για την επικοινωνία των ESP με το Raspberry προτιμήθηκε το πρωτόκολλο MQTT καθώς είναι αποδοτικό και αξιόπιστο σε συνεχή ροή δεδομένων και παράλληλα ελαφρύ για τον μικροελεγκτή.

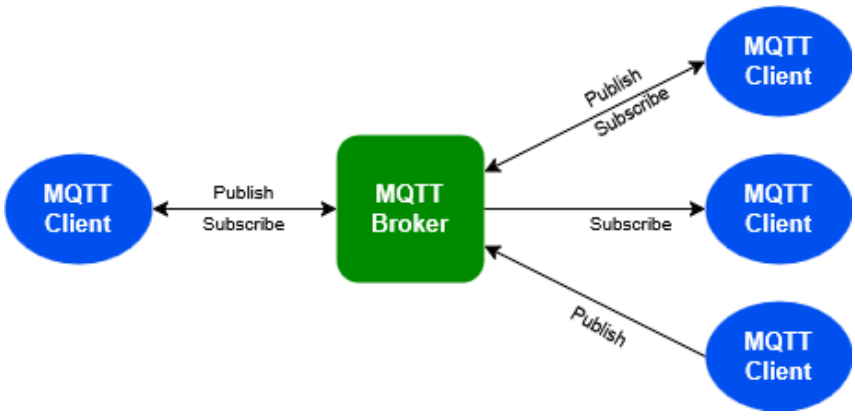
2.9.1 Πρωτόκολλο MQTT

Το πρωτόκολλο MQTT (Message Queue Telemetry Transport) ορίζει μια μορφή ασύγχρονης επικοινωνίας. Ακολουθεί το πρότυπο publish-subscribe κατά το οποίο οι publishers (αποστολείς ή εκδότες) και οι subscribers (παραλήπτες ή συνδρομητές) ανταλλάσσουν μηνύματα χωρίς να επικοινωνούν απευθείας. Εφευρέθηκε το 1999 από τους Andy Stanford-Clark και Arlen Nipper για τη χρήση σε χαμηλής κατανάλωσης αισθητήρες βιομηχανίας και την επικοινωνία με δορυφόρους. Οι αρχικές εκδόσεις χρησιμοποιήθηκαν εσωτερικά από την IBM. Το 2010 η έκδοση 3.1 δόθηκε ελεύθερα στο κοινό, ενώ το 2014 η έκδοση 3.1.1 έγινε πρώτη φορά πρότυπο από τον οργανισμό OASIS standards. Τα βασικότερα χαρακτηριστικά του MQTT πρωτοκόλλου είναι η απλή υλοποίηση του, οι χαμηλές απαιτήσεις σε υπολογιστική ισχύ και δικτυακού εύρους, τα διαθέσιμα επίπεδα ποιότητας υπηρεσίας παράδοσης μηνυμάτων και ότι είναι ανεξάρτητο του είδους των δεδομένων που μεταφέρονται (data agnostic).

Κάθε μήνυμα ανήκει σε ένα θέμα (topic). Ο εκδότης παράγει ένα μήνυμα και το κατηγοριοποιεί σε ένα θέμα. Αντίστοιχα, ένας συνδρομητής εγγράφεται σε ένα ή περισσότερα θέματα και λαμβάνει μόνο μηνύματα αυτών των θεμάτων. Υπάρχει και μία ενδιάμεση οντότητα, ο broker, που γνωρίζουν τόσο οι εκδότες όσο και οι συνδρομητές. Ο broker έχει άποψη για όλους τους συνδρομητές του συστήματος και προωθεί τα μηνύματα που λαμβάνει από τους εκδότες. Επομένως, οι αποστολείς και οι παραλήπτες δεν είναι ρητά συνδεδεμένοι μεταξύ τους και το μόνο που αρκεί να γνωρίζουν για να επικοινωνήσουν είναι ο broker. Στο σύστημα της εργασίας έγινε χρήση του broker “Eclipse Mosquitto”, ο οποίος είναι ανοικτού κώδικα λογισμικό και αποτελεί τη πιο διαδεδομένη λύση για IoT εφαρμογές.

Ένας MQTT client μπορεί να είναι εκδότης ή συνδρομητής ή και τα δύο ταυτόχρονα. Server είναι μόνο ο broker και μόνο με αυτόν συνδέονται οι clients. Client μπορεί να είναι οποιοδήποτε είδος συσκευής η οποία φέρει μια βιβλιοθήκη υλοποίησης του MQTT και

μπορεί να συνδέεται μέσω οποιουδήποτε δικτύου με τον broker. Χρησιμοποιείται το πρωτόκολλο μεταφοράς TCP/IP με θύρα συνήθως την 1883.



Σχήμα 2.13: Σχηματικό διάγραμμα της επικοινωνίας μεταξύ των οντοτήτων στην MQTT επικοινωνία

Η σύνδεση ενός client και ενός broker γίνεται, αφού αρχικά εγκαθιδρυθεί μία TCP σύνδεση μεταξύ τους, με την αποστολή ενός πακέτου 'CONNECT' από τον client. Ο broker θα απαντήσεις με ένα πακέτο 'CONNACK' και ένα κωδικό κατάστασης. Η σύνδεση θα παραμείνει ανοιχτή μέχρι ο πελάτης να αποσυνδεθεί. Σε κάθε μήνυμα που αποστέλλει ο publisher υπάρχει το όνομα του θέματος στο οποίο κοινοποιείται. Επίσης εκτός από πακέτα περιεχομένου ανταλλάσσονται και πακέτα ελέγχου, τα οποία επιτελούν διαφορετικές λειτουργίες του πρωτοκόλλου.

Το επίπεδο ποιότητας υπηρεσίας (Quality of Service Level, **QoS**) είναι μία συμφωνία μεταξύ του αποστολέα και του παραλήπτη για το επίπεδο εγγύησης μεταφοράς ενός μηνύματος. Αυτό είναι ιδιαίτερα χρήσιμο σε περιβάλλοντα ασταθούς σύνδεσης όπου απαιτείται αναμετάδοση μηνύματος. Στην προκειμένη περίπτωση για τη μετάδοση των μετρήσεων χρησιμοποιήθηκε το "QoS 0" κατά το οποίο ο αποστολέας δεν περιμένει πίσω κανέναν πακέτο αναγνώρισης ή εγγύησης αποστολής του μηνύματος. Αυτό επιτρέπει τη μέγιστη ταχύτητα μετάδοσης εφόσον τα μηνύματα δεν είναι κρίσιμα και οποιαδήποτε μικρή απώλεια είναι αποδεκτή.

2.9.2 Περιγραφή προγράμματος

Πρωταρχικό βήμα στην ανάπτυξη του προγράμματος είναι η προσθήκη των απαραίτητων βιβλιοθηκών. Αυτές είναι η "WiFi.h" για τη διαχείριση της σύνδεσης Wi-Fi, η "PubSubClient.h" για την υλοποίηση του πρωτοκόλλου MQTT και η "MCP3XXX" για την πρόσβαση στον ADC και την SPI επικοινωνία.

Η δομή του προγράμματος ακολουθεί την ίδια φιλοσοφία με τα Arduino sketches και αποτελείται από δύο κύρια μέρη. Το “setup” εκτελείται μία φορά κατά την εκκίνηση ή στο reset της πλακέτας και περιλαμβάνει αρχικοποιήσεις συσκευών, σειριακής επικοινωνίας κ.α. Το “loop” πρόκειται για το κεντρικό σώμα του προγράμματος που εκτελείται σε βρόχο, οπότε περιλαμβάνει τη βασική ρουτίνα του προγράμματος.

2.9.3 Αρχικές Ρυθμίσεις

Στο setup γίνεται αρχικοποίηση του ADC, ορίζοντας τις διεπαφές SPI που θα χρησιμοποιηθούν. Επίσης, καθορίζεται ο ρυθμός της σειριακής επικοινωνίας σε 115200 bps, όπως απαιτεί ο ESP32. Ακολουθεί η διαμόρφωση της σύνδεσης Wi-Fi και της επικοινωνίας MQTT με το ESP ως client. Για τη σύνδεση στο Access Point του Raspberry Pi απαιτείται μόνο το όνομα (SSID) και ο κωδικός πρόσβασης. Η ρύθμιση του MQTT Client είναι πιο σύνθετη. Αρχικά ορίζεται η διεύθυνση του broker, δηλαδή του Raspberry Pi, στη συνήθη port “1883”. Έπειτα ορίζονται παράμετροι σχετικά με την αξιοπιστία της σύνδεσης, όπως ο χρόνος που η επικοινωνία μπορεί να μείνει αδρανής πριν τερματιστεί (συνάρτηση setKeepAlive), ή το μέγιστο διάστημα μετάδοσης ενός πακέτου (setSocketTimeout). Τέλος, μόλις ο client συνδεθεί στον MQTT broker αρχικοποιείται η λειτουργία “Last Will and Testament (LWT)” όπου ο broker αποθηκεύει το μήνυμα κατάστασης “offline”, το οποίο δημοσιεύει στο “status” topic μόλις ο client αποσυνδεθεί. Στην αρχή της σύνδεσης δημοσιεύει το μήνυμα “online”. Κατά αυτό τον τρόπο οι υπόλοιποι clients γνωρίζουν τη κατάσταση της μονάδας.

2.9.4 Η «ρουτίνα» του προγράμματος

Ο βρόχος του προγράμματος έχει τους εξής βασικούς ρόλους: να διατηρεί τη σύνδεση MQTT ενεργή, να λαμβάνει μετρήσεις τάσης και να τις δημοσιεύει περιοδικά στο αντίστοιχο topic. Πιο αναλυτικά, σε κάθε επανάληψη του βρόχου ελέγχεται η επικοινωνία με τον broker, ώστε να γίνει η επαναφορά της εάν χρειαστεί. Για τη συντήρηση του MQTT Client είναι απαραίτητη η κλήση της συνάρτησης “client.loop” μέσα στο βρόχο. Είναι υπεύθυνη για τη διατήρηση της επικοινωνίας (αποστολή ping βάσει του επιλεγμένου “keep alive” και έλεγχος τήρησης των χρονικών ορίων “socket timeout”) και την διαχείριση των εισερχόμενων μηνυμάτων. Έπειτα, ακολουθεί η διαδικασία δημοσίευσης μιας μέτρησης του ανιχνευτή στο topic των μετρήσεων. Δεδομένου ότι υπάρχουν δύο ανιχνευτές που μπορούν να λάβουν μετρήσεις, χρησιμοποιούνται ξεχωριστά topics για κάθε συσκευή. Πρώτο βήμα, λοιπόν, για την αποστολή ενός μηνύματος μέτρησης είναι η δειγματοληψία από τον ADC. Για πιο αντιπροσωπευτικές μετρήσεις, λαμβάνεται ένα σύνολο δειγμάτων από το οποίο προκύπτει η μέση τιμή. Η 10bit-τιμή μετατρέπεται στο δεκαδικό σύστημα και έπειτα σε τιμή τάσης βάσει της σχέσης 2.1, όπου V_{ref} η τιμή τάσης που τροφοδοτεί τον ADC. Η τιμή τάσης αποστέλλεται στον broker και δημοσιεύεται στο topic της συσκευής από όπου προήλθε. Τέλος, το ο βρόχος καλεί μία συνάρτηση η οποία κοινοποιεί κάθε 30 δευτερόλεπτα ένα μήνυμα “online” στο

status topic. Αυτή λειτουργεί συμπληρωματικά με το “LWT” ώστε να προσφέρεται συνεχής ενημέρωση της κατάστασης της σύνδεσης, όχι μόνο στην αρχή της.

$$V = V_{ref} \frac{X}{2^N - 1} = V_{ref} \frac{X}{1023} \quad (2.1)$$

Για κάθε περίπτωση μηνύματος που μεταδίδεται χρησιμοποιείται διαφορετικό Quality of Service, αναλόγως το σκοπό που εξυπηρετεί. Για τα μηνύματα που είναι κρίσιμα και πρέπει να διασφαλιστεί η μετάδοση τους χρησιμοποιείται “QoS 1”, κατά το οποίο ο αποστολέας στέλνει το μήνυμα και περιμένει επιβεβαίωση. Αν δε λάβει μέσα σε ορισμένο χρονικό διάστημα, το ξαναστέλνει. Τέτοια κρίσιμα μηνύματα είναι οι εντολές που λαμβάνει ο ESP και το αρχικό LWT πακέτο. Από την άλλη, σε περιπτώσεις που η ταχύτητα της μετάδοσης κρίνεται πιο σημαντική από την απώλεια ενός μηνύματος, χρησιμοποιείται “QoS 0”, όπως κατά τη περιοδική μετάδοση των μετρήσεων και της online κατάστασης.

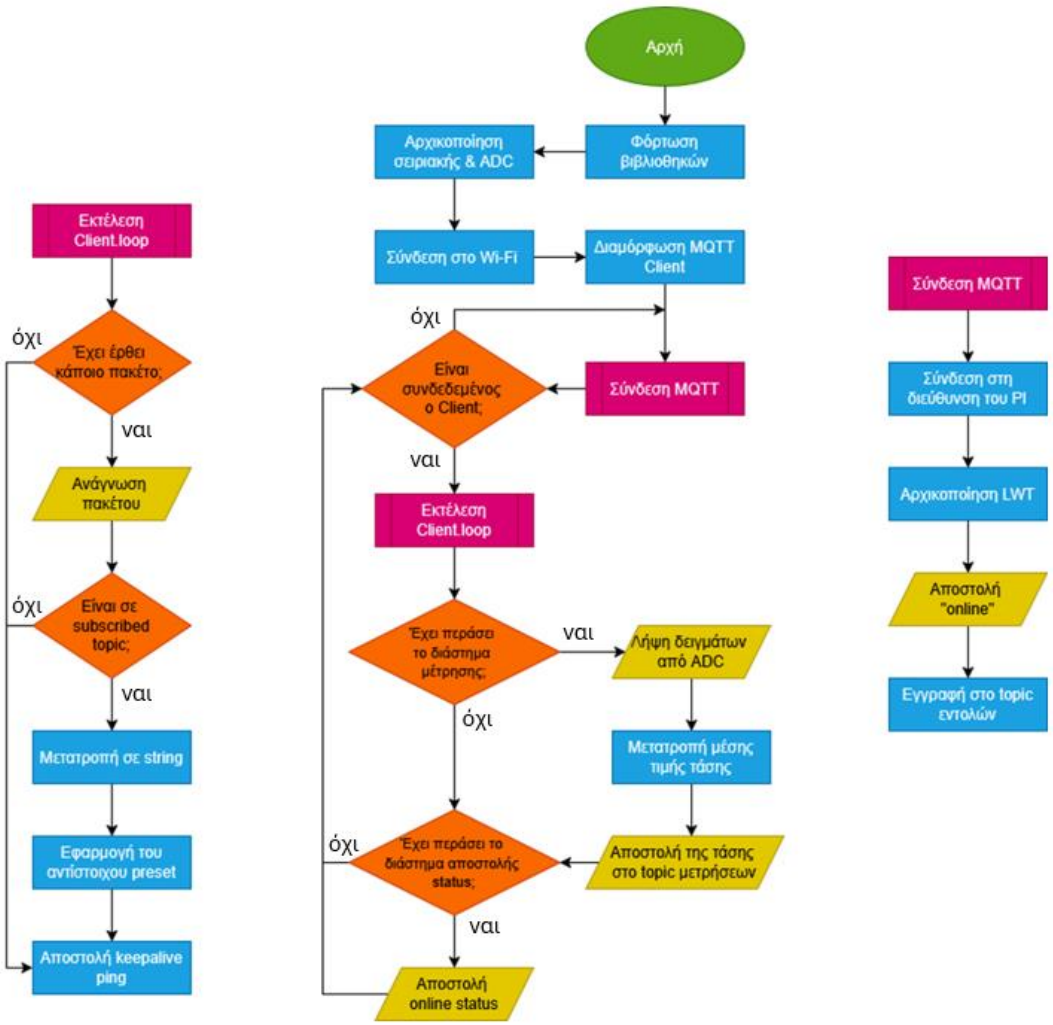
2.9.5 Δυναμικός ρυθμός μετάδοσης μετρήσεων

Αναφορικά με τα μηνύματα που λαμβάνουν οι “ESP clients”, ο broker δρομολογεί μηνύματα/εντολές στον ESP σχετικά με το ρυθμό δειγματοληψίας και αποστολής μετρήσεων. Πιο συγκεκριμένα, έχει προστεθεί η δυνατότητα ο χρήστης να μπορεί να μεταβάλλει τον ρυθμό με τον οποίο λαμβάνει μετρήσεις. Ο ESP εγγράφεται στο αντίστοιχο topic και περιμένει ένα μήνυμα (π.χ. «10» για ρυθμό 10 Hz). Έχει δημιουργηθεί ένας πίνακας προκαθορισμένων τιμών, διότι μαζί με τον ρυθμό που γίνεται publish μετρήσεων αλλάζει και ο αριθμός των δειγμάτων που λαμβάνονται και η καθυστέρηση μεταξύ αυτών. Αυτό γίνεται ώστε ο χρόνος λήψης δειγμάτων να παραμένει μικρότερος από το διάστημα αποστολής μεταξύ μετρήσεων. Για μεγάλους ρυθμούς μετάδοσης μειώνεται ο αριθμός των δειγμάτων και ο χρόνος λήψης τους, επομένως οι μετρήσεις έχουν μεγαλύτερη ανάλυση στον χρόνο αλλά και περισσότερο θόρυβο.

2.9.6 Διάγραμμα Ροής

Στα παρακάτω διαγράμματα ροής (Σχήμα 2.14) συνοψίζεται η πλήρης δομή του προγράμματος των μικροελεγκτών ESP, με όλα τα στάδια που συνθέτουν τον βασικό βρόγχο λειτουργίας του. Τα βασικά σημεία του προγράμματος είναι τα εξής:

1. Αρχικοποίηση του προγράμματος και των παραμέτρων του
2. Διατήρηση της σύνδεσης MQTT
3. Έλεγχος για ληφθέν μήνυμα σχετικό με την αλλαγή ρυθμού δειγματοληψίας
4. Δειγματοληψία από ADC για συγκεκριμένα χρονικά διαστήματα
5. Υπολογισμός της μέσης τιμής τάσης και μετατροπή της σε volts
6. Αποστολή της τιμής τάσης στο MQTT topic



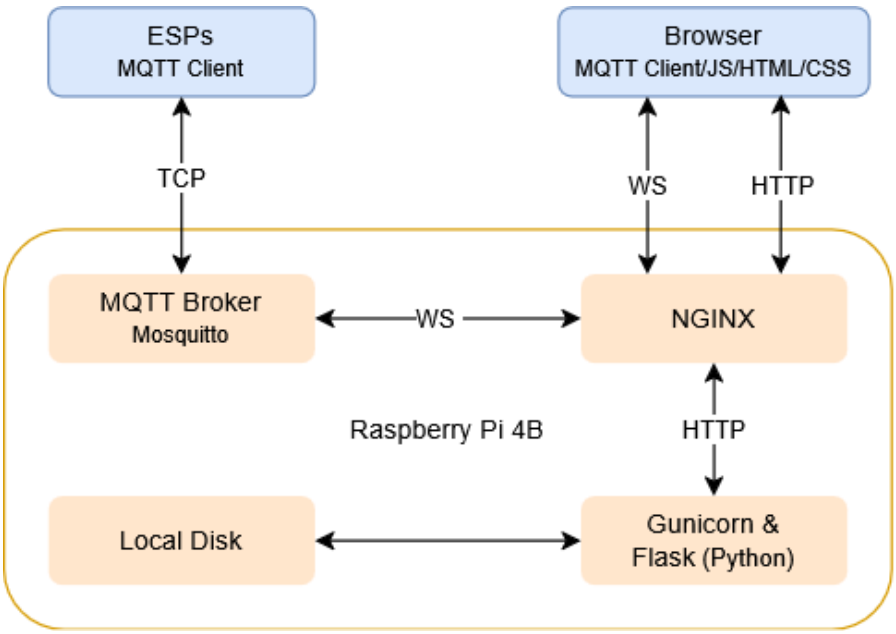
Σχήμα 2.14: Διάγραμμα ροής του προγράμματος των μικροελεγκτών ESP32.

Κεφάλαιο 3: Ανάπτυξη Διαδικτυακής Εφαρμογής

3.1 Εισαγωγή

Η διαδικτυακή εφαρμογή σχεδιάστηκε με στόχο να αποτελεί το κέντρο ελέγχου του συστήματος και ένα περιβάλλον συγκέντρωσης και ανάλυσης των μετρήσεων. Αυτή αναπτύχθηκε και λειτουργεί στο περιβάλλον ενός μικροϋπολογιστή Raspberry Pi. Αυτός αναλαμβάνει και το ρόλο του κεντρικού κόμβου MQTT (broker) που εξυπηρετεί τους διασυνδεδεμένους MQTT πελάτες (ESPs και φυλλομετρητές), δηλαδή να διατηρεί την επικοινωνία μεταξύ του χρήστη και της πηγής δεδομένων. Η εφαρμογή, για την ανάπτυξη της και τη διάθεση της στο δίκτυο, απαιτεί τη συνεργασία μιας σειράς εργαλείων και υπηρεσιών, που εκτελούνται στο Raspberry Pi και στο φυλλομετρητή του χρήστη. Σε αυτό το κεφάλαιο γίνεται περιγραφή αυτής της διαδικασίας και των τεχνολογιών που χρησιμοποιήθηκαν. Το παρακάτω σχηματικό διάγραμμα (Σχήμα 3.1) αποδίδει μια γενική εικόνα του συστήματος και τον τρόπο διασύνδεσης των διαφόρων υπηρεσιών που το συνθέτουν

.

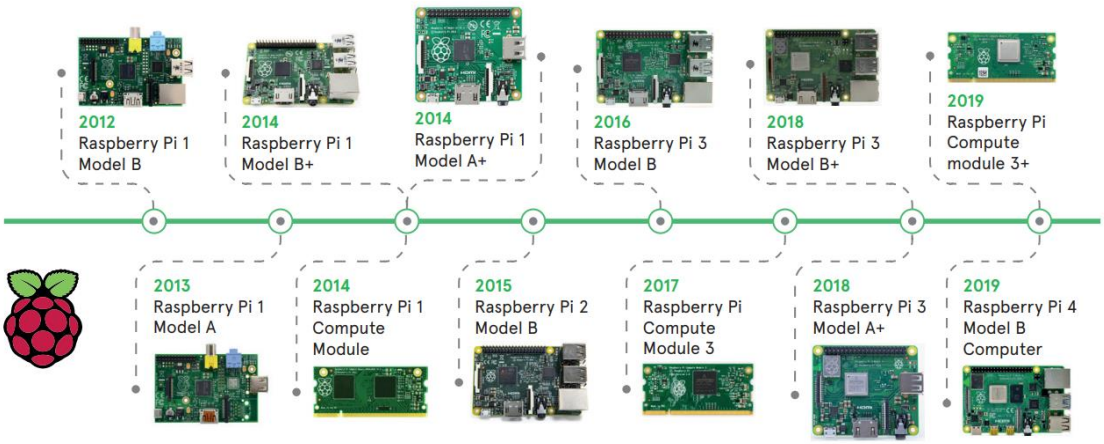


Σχήμα 3.1: Σχηματικό διάγραμμα των επιμέρους στοιχείων που συνθέτουν την εφαρμογή διαδικτύου

3.2 Αναπτυξιακή Πλακέτα Raspberry Pi

Οι συσκευές Raspberry Pi είναι μία σειρά μικροϋπολογιστών που αποτελούνται από έναν ενιαίο πίνακα κυκλωμάτων με επεξεργαστή, μνήμη, είσοδο/έξοδο και άλλα χαρακτηριστικά που συνθέτουν έναν λειτουργικό επεξεργαστή στο μέγεθος μιας πιστωτικής κάρτας. Η ανάπτυξη τους ξεκίνησε το 2009, στο Ηνωμένο Βασίλειο, από το Raspberry Pi Foundation για τη προώθηση της διδασκαλίας της βασικής πληροφορικής στα σχολεία και στις αναπτυσσόμενες χώρες. Αργότερα, η εξέλιξη της τεχνολογίας ανατέθηκε στην εταιρεία Raspberry Pi Trading με CEO τον Eben Upton, τον εμπνευστή της ιδέας. Το Raspberry Pi είχε ως στόχο να εμπνεύσει το ενδιαφέρον για τη τεχνολογία και τον προγραμματισμό, ιδιαίτερα στους νέους, γρήγορα όμως έγινε δημοφιλές τόσο σε χομπίστες όσο και σε εμπορικές εφαρμογές edge computing, όπως στη ρομποτική. Η προσιτή τιμή, το μικρό μέγεθος και η ευκολία στη χρήση του, το καθιστούν ιδανικό εργαλείο για την εκμάθηση της πληροφορικής και του προγραμματισμού καθώς και τη δημιουργία καινοτόμων εφαρμογών.

Από το 2012 που κυκλοφόρησε η πρώτη εμπορική πλακέτα, έχουν ακολουθήσει αρκετές γενιές Raspberry Pi. Σήμερα διατίθενται διάφορες εκδόσεις στην αγορά με τη βασική ειδοποιό διαφορά τον αριθμό θυρών I/O και συνεπώς το μέγεθος. Όλα τα μοντέλα διαθέτουν το Broadcom SoC με ενσωματωμένη κεντρική μονάδα επεξεργασίας, συμβατή με την ARM (CPU) και ενσωματωμένη μονάδα επεξεργασίας γραφικών on-chip (GPU).



Σχήμα 3.2: Η εξέλιξη των βασικών μοντέλων Raspberry Pi [41].

Το επίσημο και προτεινόμενο λειτουργικό σύστημα των Raspberry Pi μικροϋπολογιστών είναι το Raspberry Pi OS (ή με τη παλιά ονομασία “Raspbian”), μια διανομή Linux (32 & 64-bit) βασιζόμενη στο λειτουργικό Debian. Φυσικά υποστηρίζονται και μια πληθώρα λειτουργικών τρίτων κατασκευαστών όπως το Ubuntu, Windows 10 IoT Core, RISC OS κ.α. Το Raspberry Pi OS είναι ένα λειτουργικό ανοιχτού κώδικα, προσαρμοσμένο πάνω στην αρχιτεκτονική ARM και προσφέρει ένα φιλικό προς τον χρήστη περιβάλλον περιήγησης και προγραμματισμού, προωθώντας την Python και την Scratch ως τις κύριες γλώσσες. Διατίθεται σε “lite” έκδοση χωρίς περιβάλλον επιφάνειας εργασίας και σε “full” έκδοση με

μία πλήρη σουίτα προεγκατεστημένων λογισμικών. Για τη παρούσα εργασία χρησιμοποιήθηκε το Raspberry Pi OS αξιοποιώντας τη δυνατότητα απομακρυσμένης χρήσης μέσω VNC (Virtual Network Computing) και SSH (Secure Shell Protocol).

3.2.1 Το μοντέλο Raspberry PI 4B

Στο πλαίσιο της εργασίας χρησιμοποιήθηκε το μοντέλο μικροϋπολογιστή Raspberry PI 4 Model B, για τις ανάγκες του διακομιστή (server) του συστήματος. Στο μοντέλο βρίσκεται το chipset Broadcom BCM2711 με τον τετραπύρηνο 64-bit επεξεργαστή ARM Cortex-A72, χρονισμένο στα 1.5 GHz. Διατίθεται με ενσωματωμένη μνήμη 1, 2, 4 & 8 GB LPDDR4-3200 SDRAM (στην προκειμένη περίπτωση 4 GB), ενώ ως μόνιμη μνήμη για την αποθήκευση του λειτουργικού συστήματος και λοιπών δεδομένων δέχεται κάρτα μνήμης microSD. Περιλαμβάνει δύο θύρες USB 2.0 και δύο USB 3.0, θύρα gigabit Ethernet, ένα ζεύγος θυρών micro HDMI και μία θύρα USB-C για τη τροφοδοσία του. Επάνω στην πλακέτα διατίθεται και μία ακιδοσειρά 40 GPIO με υποστήριξη UART, I2C, SPI επικοινωνίας. Πέρα από Ethernet η διασυνδεσιμότητα του επεκτείνεται με Wi-Fi (2.4/5 GHz) και Bluetooth 5.0 υποστήριξη.

Σχήμα 3.3: Κάτοψη του μοντέλο Raspberry Pi 4B [40].

3.2.2 Προετοιμασία Raspberry Pi

Ο μικροϋπολογιστής Raspberry Pi 4B φιλοξενεί τον διακομιστή ιστού (web server) της εφαρμογής, ενώ ταυτόχρονα δικτυώνεται με τις δύο μονάδες ανίχνευσης συλλέγοντας τα δεδομένα μετρήσεων. Η επικοινωνία πραγματοποιείται μέσω ενός τοπικού δικτύου Wi-Fi που δημιουργεί το Raspberry ως Access Point. Μεταξύ των δύο ζωνών 2.4 και 5 GHz που υποστηρίζει το “Pi 4” μοντέλο, επιλέχθηκε η χαμηλότερη συχνότητα την οποία

υποστηρίζουν τα ESP. Το Wi-Fi AP δημιουργείται στην εκκίνηση του Raspberry. Όταν η συσκευή είναι συνδεδεμένη μέσω Ethernet σε ένα ευρύτερο δίκτυο, οι χρήστες του Wi-Fi έχουν επίσης πρόσβαση σε αυτό. Ουσιαστικά, το Raspberry λειτουργεί ως δρομολογητής δημιουργώντας ένα δικό του υποδίκτυο (192.168.4.x), όπου "μεταφράζει" τη κίνηση μεταξύ των δύο (Network Address Translation). Ως εκ τούτου, η εφαρμογή μπορεί να είναι προσβάσιμη μέσω μιας συσκευής που είναι συνδεδεμένη απευθείας στο Wi-Fi AP ή στο δίκτυο που έχει πρόσβαση το Raspberry μέσω του Ethernet.

Η εγκατάσταση όλων των απαραίτητων βιβλιοθηκών έγινε μέσα σε ένα εικονικό περιβάλλον (virtual environment), όπως προτρέπει το Raspberry Pi OS για να αποφευχθούν τυχόν προβλήματα με πακέτα του λειτουργικού. Πέρα από τις βιβλιοθήκες που χρησιμοποιούνται στο backend της εφαρμογής, έγινε εγκατάσταση και των frontend assets - αντί να φορτώνονται online από τον πελάτη - ώστε η εφαρμογή να είναι πλήρως λειτουργική χωρίς σύνδεση στο διαδίκτυο, στα πλαίσια του τοπικού δικτύου του Raspberry.

Εν κατακλείδι, το Raspberry Pi διαμορφώθηκε έτσι ώστε να αποτελεί τον κεντρικό κόμβο του συστήματος, όπου φιλοξενούνται οι διάφορες υπηρεσίες που χρησιμοποιούνται (MQTT Broker, Web Server, κτλ.), δημιουργείται η δικτύωση του χρήστη με τις πηγές δεδομένων και αποθηκεύονται όλα τα αρχεία μετρήσεων που δημιουργεί ο χρήστης μέσω της εφαρμογής.

3.3 Διαμόρφωση Backend

Στο παρασκήνιο του Raspberry Pi (backend) ένα σύνολο εργαλείων συνεργάζονται για να υποστηρίξουν τη λειτουργικότητα της εφαρμογής και να διαχειρίζονται αποτελεσματικά τη ροή δεδομένων από τη πηγή στον τελικό χρήστη. Ειδικότερα, είναι το μέρος όπου αναπτύσσεται η λογική της εφαρμογής, δέχεται τα αιτήματα των χρηστών και διαχειρίζεται την αποθήκευση και ανάκτηση των δεδομένων. Παρακάτω αναλύεται η αρχιτεκτονική του backend και οι τεχνολογίες που το αποτελούν.

3.3.1 Flask

Το Flask είναι ένα ελαφρύ web framework για την ανάπτυξη εφαρμογών σε Python. Αποτελεί ένα από τα πιο διαδεδομένα Python frameworks καθώς είναι σχεδιασμένο να παρέχει εύκολες και γρήγορες λύσεις στην ανάπτυξη backend λογισμικού. Χαρακτηρίζεται ως micro-framework γιατί δεν εξαρτάται από βιβλιοθήκες ή εργαλεία και δεν επιβάλλει κάποια συγκεκριμένη δομή, αλλά κρατάει τον πυρήνα της εφαρμογής μικρό και επιτρέπει τη προσθήκη λειτουργιών μέσω πακέτων και επεκτάσεων (βάσεις δεδομένων, τεχνολογίες authentication κ.α.). Βασίζεται στη βιβλιοθήκη της Python "Werkzeug" για την υλοποίηση του προτύπου WSGI (Web Server Gateway Interface) που συνδέει τη Python εφαρμογή με διακομιστές web, τη διαχείριση του συστήματος routing και της HTTP επικοινωνίας κ.α. Το

Flask προσφέρει ενσωματωμένο έναν απλό HTTP server της βιβλιοθήκης Werkzeug, ο οποίος προορίζεται μόνο για το στάδιο της ανάπτυξης της εφαρμογής.

Στο τρέχων σύστημα, το flask υλοποιεί τη λογική λειτουργία της εφαρμογής. Βασικός ρόλος είναι η διαχείριση των REST endpoints που καλεί το frontend. Με τον όρο endpoint αναφερόμαστε στη διαδρομή σε μία διεύθυνση URL, η οποία επιτρέπει την επικοινωνία με έναν συγκεκριμένο πόρο (resource) του backend χρησιμοποιώντας μεθόδους του πρωτοκόλλου HTTP. Για παράδειγμα, ο πελάτης στέλνει ένα αίτημα για ανάκτηση δεδομένων ενός πειράματος (resource) μέσω της αντίστοιχης URL (π.χ. '/experiment-data?<name>') χρησιμοποιώντας τη HTTP μέθοδο GET και ο server (εδώ το flask) αποκρίνεται επιστρέφοντας το CSV αρχείο του πειράματος. Κατά αυτό τον τρόπο υλοποιείται ένα REST API ακολουθώντας το μοντέλο διακομιστή-πελάτη και με την επικοινωνία να βασίζεται στη λογική αιτήματος-απόκρισης.

Όσον αφορά την οργάνωση του flask, αυτή ακολουθεί έναν modular σχεδιασμό, δηλαδή χωρίζεται σε λειτουργικές ενότητες ώστε να είναι πιο εύκολα διαχειρίσιμο και επεκτάσιμο. Αυτό έγινε με τη βοήθεια της κλάσης "Blueprints", όπου σε κάθε module ορίζεται από ένα blueprint με τις δικές του αντίστοιχες διαδρομές (routes) που αφορούν κάποιο λειτουργικό τμήμα της εφαρμογής (π.χ. διαχείριση πειραμάτων). Στο κεντρικό αρχείο ("app.py") αρκεί να γίνει κλήση των επιμέρους blueprints για την ενεργοποίηση τους.

Είναι σημαντικό να αναφερθεί ότι το flask δεν εμπλέκεται στη ροή των μετρήσεων από τις μονάδες ESP στο UI του χρήστη. Αντ' αυτού, τα δεδομένα οδηγούνται απευθείας μέσω της MQTT υπηρεσίας στο frontend. Αυτή η επιλογή έγινε για την απλοποίηση της flask λογικής και την αποφυγή καθυστερήσεων και σημείων συμφόρησης στο σύστημα.

Τέλος, αν και το flask διαθέτει δικό του διακομιστή για δοκιμές κατά την ανάπτυξη, στην τελική υλοποίηση χρησιμοποιήθηκε ένας κανονικός HTTP server, το Gunicorn. Αποτελεί μια διαδεδομένη λύση για Python εφαρμογές υποστηρίζοντας το πρότυπο WSGI για HTTP επικοινωνία. Πρόκειται, λοιπόν, για έναν WSGI server, ο οποίος αναλαμβάνει να μεταβιβάζει τα HTTP αιτήματα στο flask και τις απαντήσεις από αυτό. Συγκριτικά με τον ενσωματωμένο διακομιστή του flask, το Gunicorn είναι σαφώς πιο αποδοτικό, καθώς είναι σχεδιασμένο για συνεχή λειτουργία και μπορεί να διαχειρίζεται πολλαπλά ταυτόχρονα αιτήματα.

3.3.2 NGINX

Μεταξύ της εφαρμογής και του δικτύου μεσολαβεί το NGINX. Πρόκειται για ένα λογισμικό ανοικτού κώδικα, το οποίο χρησιμοποιείται κυρίως ως διακομιστής ιστού (web server), ενώ διαθέτει και άλλες λειτουργίες ιστού όπως αυτή του αντίστροφου μεσολαβητή (reverse proxy). Έχει σχεδιαστεί έτσι ώστε να διαχειρίζεται αποτελεσματικά πολλαπλές συνδέσεις με χαμηλή κατανάλωση πόρων.

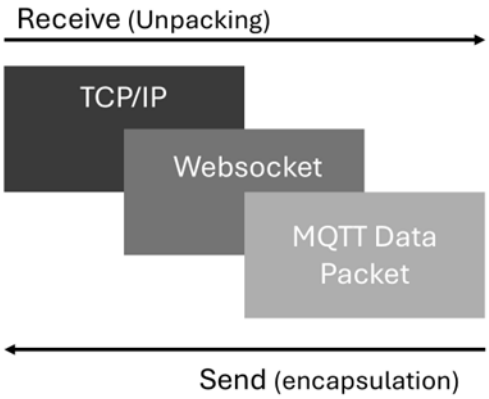
Ο ρόλος του Nginx στο σύστημα είναι να διαχειρίζεται όλες τις συνδέσεις των χρηστών με το backend της εφαρμογής, αποτελώντας το σημείο επικοινωνίας της εφαρμογής προς το δίκτυο. Επιτελεί χρέη web server διανέμοντας τα στατικά αρχεία της εφαρμογής (π.χ. HTML, CSS, βιβλιοθήκες) στον φυλλομετρητή, μειώνοντας έτσι τον φόρτο από το flask. Λειτουργεί και ως reverse proxy ώστε να προωθεί τα HTTP αιτήματα στο Gunicorn και να επιστρέφει τις απαντήσεις. Επίσης, διαμεσολαβεί στις websocket συνδέσεις του MQTT μεταξύ broker (Raspberry) και πελατών (browser), προωθώντας τα αιτήματα στον Websocket Listener χωρίς να εκτίθεται απευθείας η διεύθυνση του broker στο δίκτυο. Συνεπώς, το NGINX δρα σαν ένα τείχος προστασίας της εφαρμογής από το δίκτυο, καθώς είναι υπεύθυνο για τη δρομολόγηση κάθε σύνδεσης πελάτη με την αντίστοιχη backend υπηρεσία και την απομόνωση τους από δημόσια πρόσβαση.

Μία ακόμη λειτουργία του NGINX που υλοποιείται στην εφαρμογή είναι το “Basic Authentication”. Πρόκειται για μία απλή μέθοδο προστασίας μιας ιστοσελίδας κατά την οποία ο browser ζητά από τον χρήστη την εισαγωγή username/password πριν την επίσκεψη σε αυτή. Εφαρμόζεται μόνο στο NGINX, επομένως η πρόσβαση στην εφαρμογή αποκτάται μόλις επαληθευτεί ο χρήστης.

3.3.3 MQTT over Websockets

Το Websocket είναι ένα πρωτόκολλο επικοινωνίας υπολογιστών το οποίο επιτρέπει την αμφίδρομη ανταλλαγή δεδομένων μέσω μιας μόνιμης TCP σύνδεσης. Εφόσον, οι φυλλομετρητές δεν υποστηρίζουν απευθείας TCP κανάλια για λόγους ασφαλείας, το websocket λειτουργεί στο επίπεδο εφαρμογής ώστε να παρέχει στη σύνδεση τη συμβατότητα που απαιτούν οι browsers. Το websocket ξεκινά με ένα HTTP αίτημα από τον πελάτη, το οποίο ελέγχει ο διακομιστής και αποφασίζει αν θα ξεκινήσει η σύνδεση. Αν και είναι διαφορετικό πρωτόκολλο από το HTTP χρησιμοποιεί κοινές θύρες (80 & 443).

Το πρωτόκολλο MQTT δεν εφαρμόζεται μόνο στην επικοινωνία του Raspberry με τις μονάδες ESP, αλλά και με τους φυλλομετρητές. Δηλαδή το MQTT χρησιμοποιείται για τη δημιουργία ενός ενιαίου δικτύου διασύνδεσης των ESPs με τον browser (clients) με μεσολαβητή το Raspberry (broker). Ένας browser μπορεί να αναλάβει το ρόλο MQTT πελάτη μόνο μέσω websocket πρωτοκόλλου. Ουσιαστικά, η υλοποίηση “MQTT over websocket” πρόκειται για έναν συνδυασμό των δύο πρωτοκόλλων, όπου τα πακέτα MQTT ενθυλακώνονται μέσα σε websocket πακέτα, τα οποία με τη σειρά του μεταδίδονται μέσα σε TCP/IP πακέτα. Ο Mosquitto broker διατηρεί ενεργούς “websocket listeners” για τη δημιουργία καναλιών και τη λήψη μηνυμάτων μέσα σε αυτά. Με την επέκταση του MQTT δικτύου μέχρι το frontend της εφαρμογής, το UI μπορεί να λαμβάνει τις μετρήσεις παθητικά μέσω του subscribe/publish μηχανισμού, μειώνοντας το φόρτο του συστήματος και τις καθυστερήσεις μέσα σε αυτό.



Σχήμα 3.4: Η διακίνηση των πακέτων σε μία σύνδεση MQTT over Websocket.

3.4 Frontend

Η εφαρμογή ολοκληρώνεται με την ανάπτυξη του περιβάλλοντος αλληλεπίδρασης με τον χρήστη, δηλαδή το frontend. Περιλαμβάνει τη σχεδίαση του οπτικού μέρους της διεπαφής (User Interface), αλλά και τη λειτουργικότητα που λαμβάνει μέρος στην πλευρά του χρήστη ώστε να εξυπηρετεί τα αιτήματα του και να επικοινωνεί με το Backend. Εδώ, κατανέμεται και το μεγαλύτερο ποσοστό της λογικής της εφαρμογής. Αυτή περιλαμβάνει την παρακολούθηση των δεδομένων πραγματικού χρόνου, τη καταγραφή πειραματικών μετρήσεων ισχύος σήματος και την ανάλυση αυτών ως προς την εξασθένιση διαδρομής.

Ο πυρήνας της διεπαφής βασίζεται στον συνδυασμό των γλωσσών Javascript, HTML και CSS, επιτρέποντας την εκτέλεση της σε οποιοδήποτε σύγχρονο φυλλομετρητή. Η HTML δημιουργεί το περιεχόμενο στη σελίδα και καθορίζει τη δομή του, ενώ η CSS αναλαμβάνει τη μορφοποίηση του. Μέσω της Javascript προστίθεται λειτουργικότητα στη σελίδα, προσφέροντας χειρισμό συμβάντων και δυναμική τροποποίηση του περιεχομένου. Συμπληρωματικά με τη Javascript γίνεται αξιοποίηση κάποιων εξειδικευμένων βιβλιοθηκών, όπως η MQTT.js , η οποία προσθέτει στον browser τη δυνατότητα MQTT επικοινωνίας.

3.4.1 Γενική εικόνα της σελίδας

Η σελίδα χωρίζεται σε δύο λειτουργικά μέρη - το “Live” και το “Experiment”. Όπως υποδηλώνουν και τα ονόματα τους, το πρώτο πρόκειται για το περιβάλλον προβολής των μετρήσεων σε πραγματικό χρόνο, ενώ το δεύτερο είναι ο χώρος εκτέλεσης και ανάλυσης των πειραμάτων εξασθένισης διαδρομής σήματος. Δεν αφορούν ξεχωριστές σελίδες αλλά διαφορετικές καταστάσεις με δική τους συμπεριφορά που εκτελείται αποκλειστικά στο μέρος του πελάτη. Τα δύο “modes” μοιράζονται αρκετά στοιχεία διεπαφής και λειτουργικότητας.

Σχήμα 3.5: Το αρχικό περιβάλλον της ιστοσελίδας (Live) με τα δύο γραφήματα παρακολούθησης της ζωντανής ροής από τους δύο ανιχνευτές – του ZX47-60LN-S+ (**πάνω**) και του ZX47-40LN-S+ (**κάτω**).

Το βασικό περιεχόμενο της σελίδας, που είναι κοινό στα δύο modes, περιλαμβάνει:

- **Μπάρα ελέγχου**: Διαθέτει διάφορα στοιχεία ελέγχου (όπως κουμπιά, checkboxes, στοιχεία εισόδου κειμένου) σχετιζόμενα με την επιλεγμένη λειτουργία. Η μπάρα διατηρείται πάντοτε στη κορυφή της σελίδας και επιτρέπει την εναλλαγή μεταξύ Live-Experiment.

- **Δύο γραφήματα χρόνου**: Τοποθετούνται το ένα κάτω από το άλλο και αναλαμβάνουν την απεικόνιση των δεδομένων των ανιχνευτών σε πραγματικό χρόνο. Κάθε διάγραμμα αντιστοιχεί σε μία μονάδα ανιχνευτή. Αποτελούν μέρος της βιβλιοθήκης "Plotly.js", μιας ανοικτού κώδικα βιβλιοθήκης (στη γλώσσα Javascript), η οποία χρησιμοποιείται για τη δημιουργία διαδραστικών γραφημάτων και την οπτικοποίηση δεδομένων.
- **Πίνακες δεδομένων**: Δίπλα σε κάθε διάγραμμα βρίσκεται ένας πίνακας στον οποίο εισάγονται οι μετρήσεις των ανιχνευτών παράλληλα με την απεικόνιση τους στο διάγραμμα. Ο πίνακας είναι "κρυμμένος" από προεπιλογή, αλλά ο χρήστης μπορεί να τον εμφανίσει με το αντίστοιχο κουμπί "table" πάνω από το γράφημα.
- **Μπάρα συσκευών**: Βρίσκεται στο κάτω μέρος της οθόνης και περιέχει τις τελευταίες μετρήσεις από κάθε ανιχνευτή, καθώς και ένα checkbox για τον καθένα, με το οποίο εμφανίζεται ή αποκρύπτεται το αντίστοιχο διάγραμμα. Συνήθως, γίνεται χρήση ενός ανιχνευτή, οπότε αυτό μειώνει τη πληροφορία στην οθόνη.
- **Ένδειξη κατάστασης**: Πάνω από κάθε διάγραμμα υπάρχει μία ένδειξη που υποδεικνύει τη κατάστασης της μονάδας του ανιχνευτή. Αν το ESP βρίσκεται σε λειτουργία και είναι συνδεδεμένο στο δίκτυο, η ένδειξη είναι πράσινη και αναγράφει "online", αλλιώς είναι κόκκινη με την ετικέτα "offline".

Σχήμα 3.6: Το Live περιβάλλον με μόνο ένα κανάλι δεδομένων ενεργοποιημένο και τον αναδυόμενο πίνακα δεδομένων.

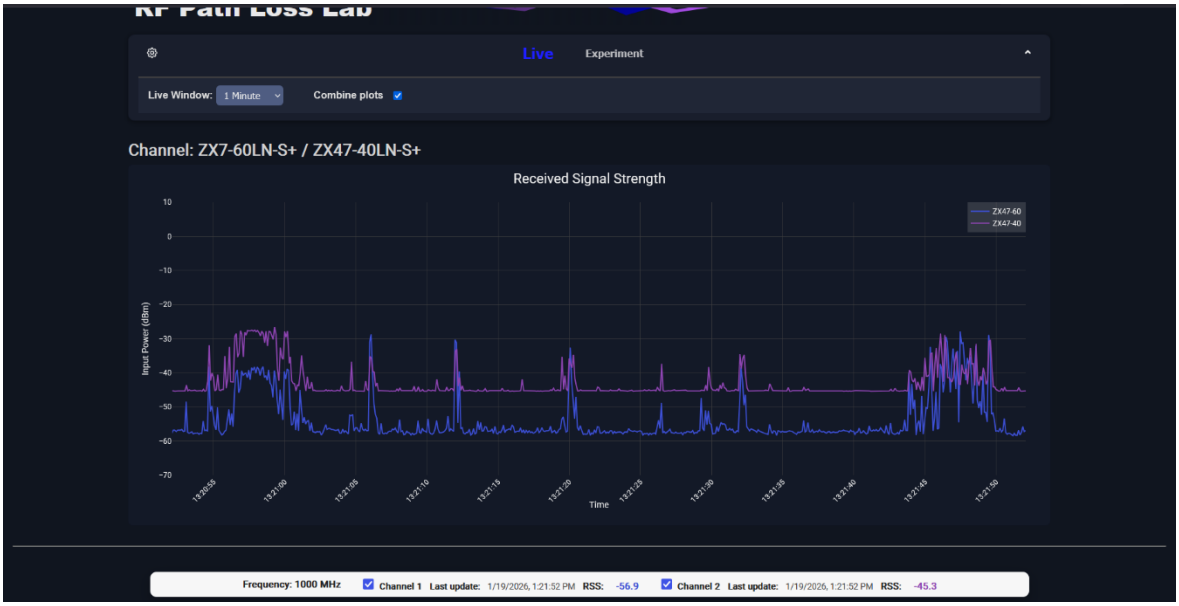
3.4.2 MQTT.js

Η MQTT.js είναι μία ανοικτού κώδικα βιβλιοθήκη για τη χρήση του πρωτοκόλλου MQTT σε περιβάλλοντα που βασίζονται σε Javascript (φυλλομετρητές, Node.js). Η τεχνολογία αναπτύχθηκε και συντηρείται από την open source κοινότητα και αποτελεί ένα από τα πιο δημοφιλή εργαλεία για την επικοινωνία IoT συσκευών με εφαρμογές διαδικτύου. Παρέχει ένα απλό τρόπο υλοποίησης του πρωτοκόλλου στο μέρος του πελάτη ώστε να μπορεί να συνδέεται σε έναν broker, να δημοσιεύει και να εγγράφεται σε MQTT topics ως ένας ασύγχρονος MQTT Client. Υποστηρίζει συνδέσεις MQTT/TCP, MQTT/TLS και MQTT/Websocket, ωστόσο σε εφαρμογές που εκτελούνται απευθείας στον φυλλομετρητή μόνο η σύνδεση μέσω Websocket είναι εφικτή.

3.5 Παρακολούθηση ανιχνευτών σε πραγματικό χρόνο

Η εφαρμογή εισάγει τον χρήστη στο “Live” περιβάλλον παρακολούθησης των ανιχνευτών σε πραγματικό χρόνο. Εάν κάποια μονάδα ανιχνευτή είναι συνδεδεμένη στο δίκτυο του Raspberry Pi τότε ξεκινά αυτόματα η εμφάνιση τιμών στο σχετικό διάγραμμα. Κατά τη φόρτωση της σελίδας πραγματοποιείται η σύνδεση MQTT μεταξύ του φυλλομετρητή και του broker (Raspberry). Ο browser, ως MQTT πελάτης, αρχικοποιεί τις παραμέτρους της σύνδεσης σύμφωνα με τη κλάση “mqttClient” της βιβλιοθήκης “mqtt.js”. Η διαδικασία περιλαμβάνει τη δημιουργία του καναλιού Websocket με τον broker και έπειτα την εγγραφή στα κατάλληλα θέματα (topics). Συγκεκριμένα, ο browser εγγράφεται στα topics μετρήσεων και status για κάθε μονάδα, με QoS 0, όπως ακριβώς και τα ESP.

Στη σελίδα “Live” ο χρήστης συναντά, πέρα από τα γραφήματα και τους πίνακες δεδομένων, μερικά στοιχεία επιλογής μέσα στην επέκταση της μπάρας ελέγχου. Ένα dropdown μενού δίνει τη δυνατότητα επιλογής του χρονικού παραθύρου που θα εμφανίζουν τα διαγράμματα. Δίνονται συγκεκριμένα διαστήματα, σχετικά μικρού χρόνου (1, 3 & 5 λεπτά), ώστε να μην συσσωρεύονται πολλά σημεία στο γράφημα και επιβαρύνεται η μνήμη του συστήματος και παράλληλα να μην επηρεάζεται η διακριτική ικανότητα. Δίπλα στο μενού υπάρχει ένα checkbox, το οποίο κατά την επιλογή του ενοποιεί τα δύο διαγράμματα σε ένα. Με αυτό τον τρόπο, όταν και τα δύο κανάλια είναι ενεργά, η σύγκριση των μετρήσεων από τις δύο πηγές είναι πιο εύκολη.



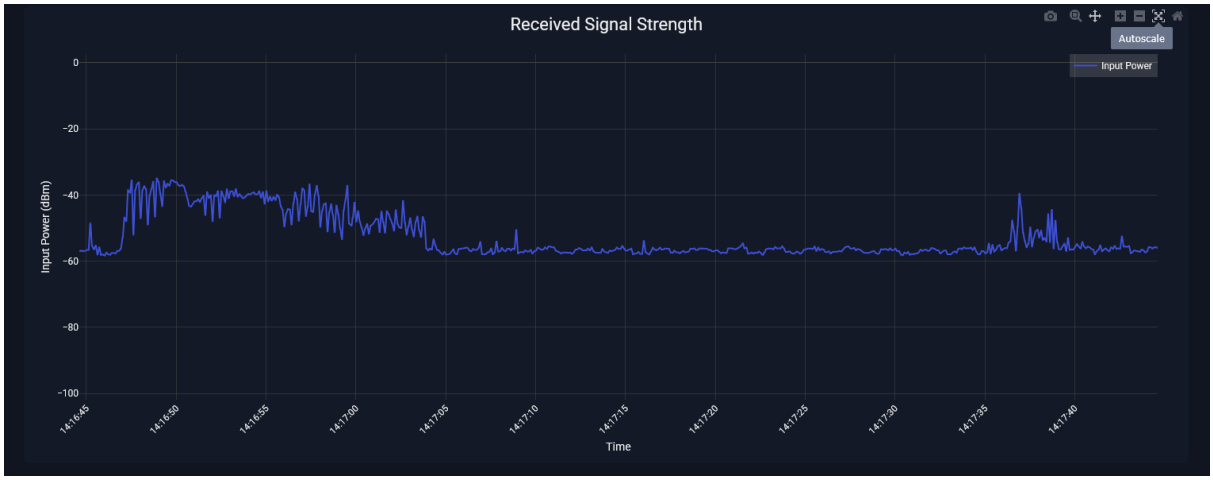
Σχήμα 3.7: Προβολή των δύο καναλιών δεδομένων σε ένα γράφημα για άμεση σύγκριση.

Με τη χρήση Websocket το κανάλι επικοινωνίας με το Raspberry Pi παραμένει ανοιχτό καθόλη τη διάρκεια λειτουργίας της εφαρμογής, επιτρέποντας την ανταλλαγή δεδομένων χωρίς τη προήγηση αιτημάτων. Για κάθε μήνυμα που λαμβάνεται στον browser, γίνεται διαχωρισμός του «ωφέλιμου φορτίου» (payload) ώστε να διευκρινιστεί αν πρόκειται για μήνυμα κατάστασης ή μέτρησης. Στη πρώτη περίπτωση γίνεται ενημέρωση της “online” κατάστασης της συσκευής. Αν το μήνυμα περιέχει μέτρηση, δηλαδή μια τιμή τάσης, σε αυτή προστίθεται η τρέχουσα σφραγίδα χρόνου (timestamp) και εισάγεται στο buffer του καναλιού.

Η διαχείριση των δεδομένων οργανώνεται σε δύο κανάλια, ένα για κάθε μονάδα ανιχνευτή, τα οποία λειτουργούν ανεξάρτητα μεταξύ τους και τροφοδοτούν το αντίστοιχο διάγραμμα και πίνακα. Για κάθε νέα μέτρηση που λαμβάνεται στο κανάλι δε γίνεται επανασχεδιασμός του γραφήματος, αντ’ αυτού η Plotly προσφέρει έναν πιο αποδοτικό τρόπο εισαγωγής δεδομένων μέσω της συνάρτησης “extentTraces”. Το νέο σημείο προστίθεται στο τέλος του γραφήματος και παράλληλα αφαιρείται το παλαιότερο, ώστε τα δεδομένα να εμπίπτουν στο επιλεγμένο χρονικό παράθυρο. Παρόμοια τακτική ακολουθείται και στους πίνακες, οι οποίοι διατηρούν ίδιο αριθμό γραμμών με τα σημεία στα γραφήματα.

Η διαδραστικότητα είναι ένα βασικό χαρακτηριστικό των Plotly γραφημάτων, καθώς προσφέρουν στον χρήστη μια σειρά εργαλείων χωρίς επιπλέον ανάπτυξη κώδικα. Κάποιες σημαντικές λειτουργίες είναι:

- **Γραμμή εργαλείων**: Βρίσκεται στο πάνω μέρος των γραφημάτων και συγκεντρώνει διάφορες επιλογές όπως zoom in/out, autoscale, επαναφορά αξόνων, λήψη εικόνας του γραφήματος.
- **Zoom & Pan**: Επιλογή και εστίαση σε μία συγκεκριμένη περιοχή του γραφήματος και μετακίνηση μέσα σε αυτό.
- **Hover**: Τοποθετώντας το ποντίκι πάνω σε ένα σημείο των δεδομένων εμφανίζεται ένα πλαίσιο με τις ακριβείς τιμές του.
- **Legend Filter**: Μέσω του υπομνήματος (legend) μπορεί να γίνει απόκρυψη ή εμφάνιση συγκεκριμένων σειρών δεδομένων.

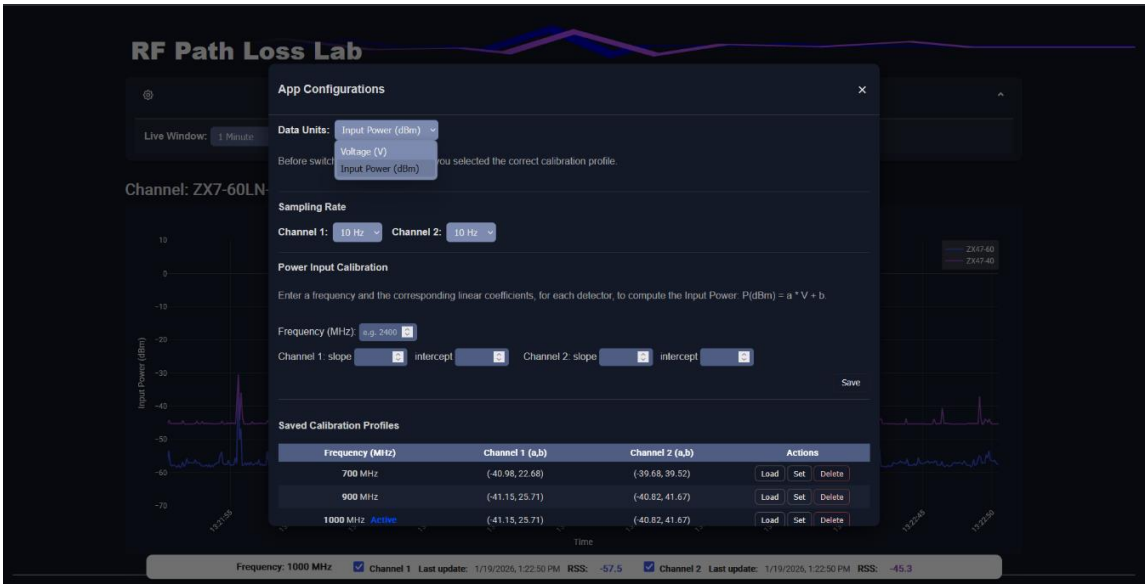


Σχήμα 3.8: Η γραμμή εργαλείων που προσφέρει η Plotly στο πάνω δεξιά μέρος του διαγράμματος

3.6 Παράμετροι εφαρμογής και διαμόρφωση

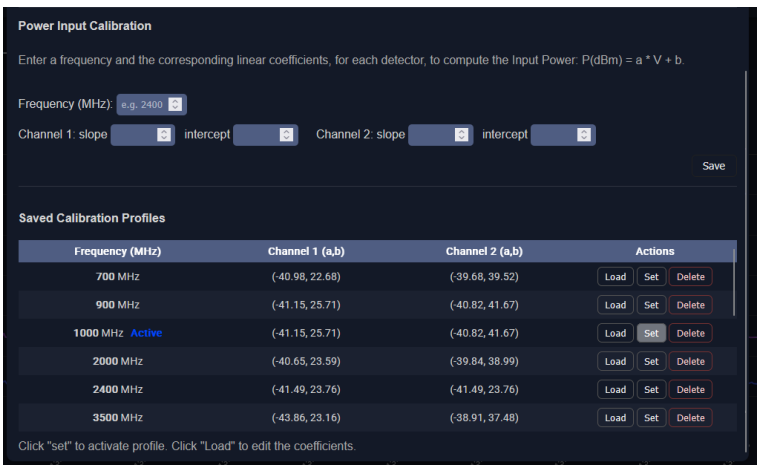
Στην μπάρα ελέγχου υπάρχει το κουμπί “Configurations” υπό το εικονίδιο γραναζιού, το οποίο ανοίγει το παράθυρο διαμόρφωσης. Εδώ ο χρήστης μπορεί να αλλάξει κάποιες μεταβλητές της εφαρμογής που αφορούν την απεικόνιση και τη ποιότητα των δεδομένων.

Ειδικότερα, ο χρήστης μπορεί να επιλέξει τις μονάδες μέτρησης που εμφανίζονται τα δεδομένα στην εφαρμογή. Οι μετρήσεις λαμβάνονται από το ESP όπως ακριβώς εξάγονται από τον ανιχνευτή ισχύος, δηλαδή σε τιμές τάσεις. Ωστόσο, για τη σωστή ερμηνεία της μέτρησης πρέπει να γίνει η μετατροπή σε μία διαχειρίσιμη μονάδα έκφρασης ισχύος σήματος, όπως τα dBm.



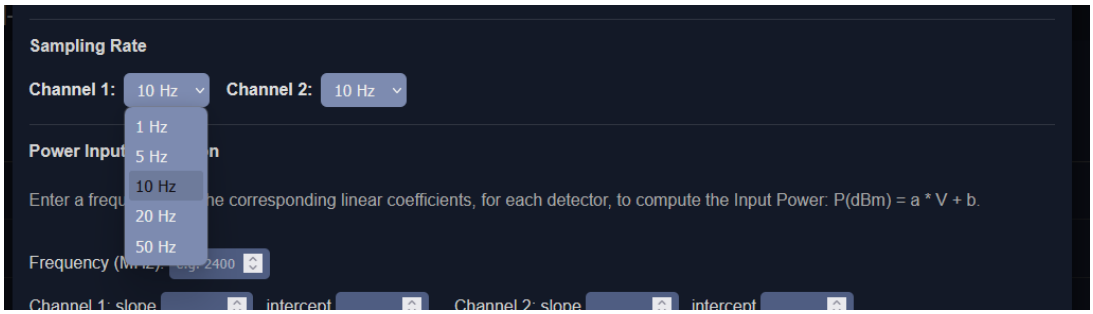
Σχήμα 3.9: Το παράθυρο διαμόρφωσης της εφαρμογής.

Πριν τη μετάβαση από Volts σε dBm, είναι απαραίτητη η επιλογή των γραμμικών παραμέτρων της μετατροπής. Πρόκειται στην ουσία για τη κλίση και τον σταθερό όρο της ευθείας που εκφράζει τη τάση εξόδου συναρτήσει της ισχύος σήματος και δίνονται στο φύλλο προδιαγραφών του ανιχνευτή για διαφορετικές συχνότητες. Μέσα από το παράθυρο “Configurations”, λοιπόν, ο χρήστης δύναται να επιλέξει το κατάλληλο προφίλ βαθμονόμησης (Calibration Profile) της συχνότητας στην οποία εργάζεται. Για τη δημιουργία ενός τέτοιου προφίλ υπάρχουν πέντε στοιχεία εισόδου (αριθμού): ένα για τη συχνότητα (σε MHz) και ένα για τη κλίση και τον σταθερό όρο της καμπύλης κάθε ανιχνευτή. Τα προφίλ που δημιουργούνται αποθηκεύονται στον διακομιστή (στο Raspberry Pi) και κατά το άνοιγμα του παραθύρου φορτώνονται και εμφανίζονται σε έναν πίνακα. Για κάθε συχνότητα μπορεί οριστεί μόνο ένα προφίλ βαθμονόμησης. Δίπλα από κάθε αποθηκευμένο προφίλ υπάρχει η επιλογή ορισμού ως ενεργό ή διαγραφής του.



Σχήμα 3.10: Το μέρος διαχείρισης των προφίλ βαθμονόμησης των ανιχνευτών.

Μία ακόμη παράμετρος που μπορεί να ρυθμιστεί είναι ο ρυθμός λήψης μετρήσεων. Όπως αναφέρθηκε στην ενότητα 2.9.5, ο μικροελεγκτής έχει τη δυνατότητα να μεταβάλλει το ρυθμό δειγματοληψίας και αποστολής μετρήσεων μεταξύ προκαθορισμένων τιμών. Στο παράθυρο διαμόρφωσης υπάρχει ένα dropdown μενού με αυτές τις τιμές. Επιλέγοντας την επιθυμητή, αυτή δημοσιεύεται στο σχετικό topic, λαμβάνεται από τα ESP και προσαρμόζονται αναλόγως. Οι τιμές ρυθμού κυμαίνονται μεταξύ του 1 Hz και των 50 Hz. Για ρυθμούς μεγαλύτερους από 50 Hz παρατηρούνταν θόρυβος στη δειγματοληψία (λόγω μικρότερου averaging των δειγμάτων) και ενδείξεις υπερφόρτωσης CPU από τον browser (λόγω πολύ γρήγορου rendering των γραφημάτων). Ο ιδανικός ρυθμός δειγματοληψίας είναι τα 10 Hz καθώς αποδίδει επαρκή ανάλυση στις μεταβολές των μετρήσεων (για τις ανάγκες των πειραμάτων), ενώ δεν απαιτούνται σημαντικοί πόροι από τη συσκευή για την απεικόνιση τους.



Σχήμα 3.11: Dropdown μενού για την αλλαγή του ρυθμού μετάδοσης των μετρήσεων.

3.7 Πειράματα εξασθένησης διαδρομής σήματος

Στο περιβάλλον “Experiment” κατέχει το κύριο ρόλο της εφαρμογής, καθώς ο χρήστης μπορεί να πραγματοποιήσει πειράματα υπολογισμού της εξασθένησης διαδρομής (Path Loss) RF σήματος. Συνοπτικά, η διεπαφή έχει σχεδιαστεί ώστε ο χρήστης να μπορεί εύκολα να δημιουργεί σετ μετρήσεων από τους ανιχνευτές, μέσα στο πλαίσιο του σεναρίου διάδοσης που έχει δημιουργήσει. Μπορεί να χρησιμοποιηθεί ένας ανιχνευτής ή και οι δύο ταυτόχρονα. Τα σετ μετρήσεων αποθηκεύονται μόνιμα στον server και δίνεται η δυνατότητα ανάλυσης τους μέσα στο περιβάλλον της εφαρμογής. Η ανάλυση περιλαμβάνει τον υπολογισμό της εξασθένησης διαδρομής και τη προσαρμογή μοντέλων διάδοσης σήματος.

3.7.1 Σχεδιαστική λογική

Το UI του “Experiment” ακολουθεί παρόμοια σχεδιαστική λογική με αυτή του “Live”. Στη μπάρα ελέγχου περιέχονται όλα τα στοιχεία εισαγωγής παραμέτρων που απαιτεί ένα πείραμα, όπως το όνομα του και πληροφορίες αναφορικά με το Link Budget (απόσταση πομπού με κάθε δέκτη, ισχύς εκπομπής πομπού, κέρδος κεραίας πομπού, κέρδος κεραίας κάθε δέκτη, εξασθένηση συστήματος). Στο ίδιο πλαίσιο υπάρχουν και δύο κουμπιά, το “Browse” και το “Export”. Το πρώτο ανοίγει ένα παράθυρο με ένα πίνακα που περιέχει όλα τα αποθηκευμένα πειράματα. Δίπλα από κάθε όνομα αναγράφεται η τελευταία τροποποίηση του πειράματος και το μέγεθος του, ενώ υπάρχουν και δύο κουμπιά για τη διαγραφή ή τη χρήση του. Πάνω από τη λίστα υπάρχει το “Import” μέσω του οποίου μπορεί να γίνει εισαγωγή ενός αρχείου πειράματος από τη τοπική συσκευή. Με το “Export” κουμπί, μπορεί να γίνει λήψη του προβαλλόμενου πειράματος, δηλαδή ενός πειράματος που μόλις έχει ολοκληρωθεί ή έχει γίνει φόρτωση του από τον server. Στο κέντρο του πλαισίου αναγράφονται πληροφορίες σχετικά με τη κατάσταση του πειράματος και κατά την εκτέλεση εμφανίζεται ένα ρολόι που καταγράφει τη διάρκεια του.

Σχήμα 3.12: Το περιβάλλον “Experiment” για την εκτέλεση και ανάλυση πειραμάτων εξασθένησης διαδρομής.

Στη δεξιά πλευρά της σελίδας τοποθετείται ένα πλαίσιο με τα κουμπιά που ελέγχουν τη ροή ενός πειράματος, τα “start”, “pause” και “stop”. Το pause button χρησιμοποιείται για τη προσωρινή παύση της λήψης δεδομένων στο κανάλι. Μιας και τα πειράματα εξασθένισης διαδρομής περιλαμβάνουν μετρήσεις σε διαφορετικές αποστάσεις πομπού-δέκτη, είναι αναγκαία η τακτική διακοπή της καταμέτρησης για τη μεταβολή της απόστασης των

κεραιών. Σε κάθε παύση, το “pause” μετατρέπεται σε “resume” για τη συνέχιση της διαδικασίας.

Όπως στο “Live mode”, έτσι και εδώ, τα γραφήματα αποτυπώνουν τη ροή των μετρήσεων σε πραγματικό χρόνο, με τη διαφορά όμως ότι αυτή είναι ελεγχόμενη. Τα δεδομένα ξεκινούν να λαμβάνονται με την εκκίνηση του πειράματος και σταματούν με την ολοκλήρωση του. Εν τέλει, το διάγραμμα εμφανίζει το σετ μετρήσεων συναρτήσει του χρόνου του ολοκληρωμένου πειράματος ή ενός παλαιότερου που μεταφορτώθηκε.

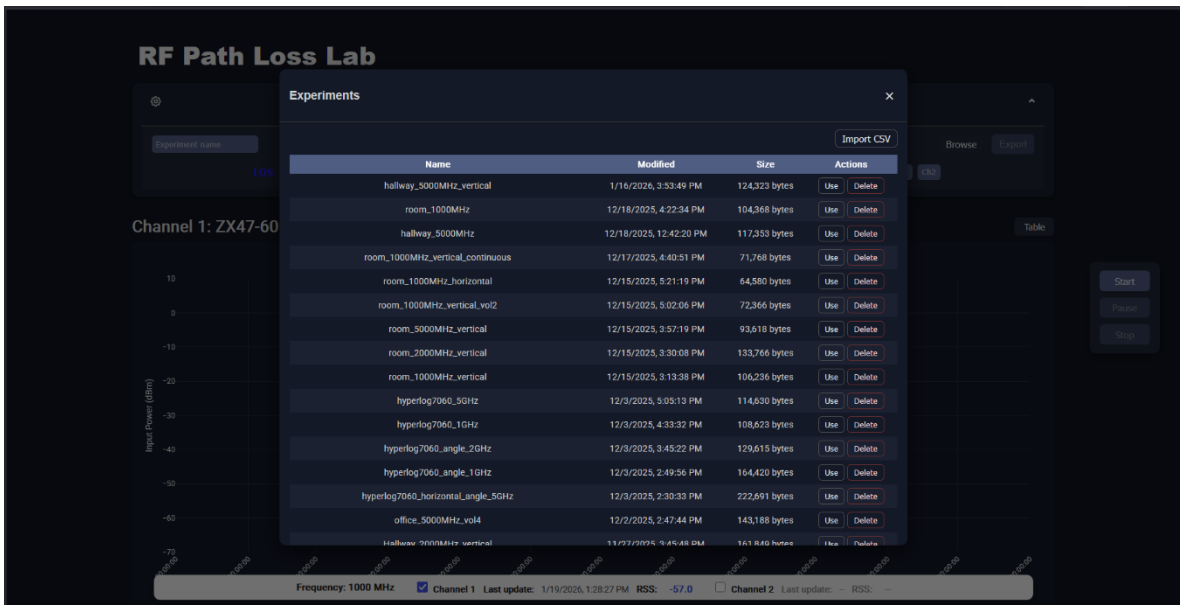
3.7.2 Αποθήκευση πειραμάτων

Με την ολοκλήρωση ενός πειράματος, γίνεται αυτόματα αποθήκευση του στον τοπικό δίσκο του Raspberry Pi. Οι μετρήσεις αποθηκεύονται υπό τη μορφή CSV (Comma - separated values) αρχείου που δημιουργείται στη μεριά του πελάτη και μεταφορτώνεται στον server. Αυτό περιλαμβάνει τις εξής εννιά στήλες: σφραγίδα χρόνου, συσκευή, τιμή τάσης (V), τιμή ισχύος (dBm), συχνότητα (MHz), κλίση και σταθερός όρος καμπύλης βαθμονόμησης, απόσταση (m) και εξασθένηση (dB). Στο αρχείο καταγράφονται, επίσης, οι παράμετροι του Link Budget και η συχνότητα καθώς απαιτούνται στη μετέπειτα ανάλυση των μετρήσεων. Για την αποφυγή δημιουργίας επιπλέον στηλών για μερικές σταθερές, προστίθεται στην αρχή του αρχείου μια γραμμή σαν σχόλιο, με το κείμενο σε JSON μορφή.

```
#exp-info: {"freqMHz":5000,"budget":{"Pt":16,"Gt":5,"rss60":{"Gr":5,"Lsys":3}}}
timestamp,device,volt,dbm,freq_mhz,a,b,distance_m,path_loss_db
2026-01-16T13:19:29.851Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:29.954Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.054Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.157Z,esp32_60,1.28,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.262Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.331Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.485Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.541Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.670Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.735Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.875Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:30.932Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:31.079Z,esp32_60,1.28,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:31.133Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:31.256Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:31.386Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
2026-01-16T13:19:31.452Z,esp32_60,1.279,-26.5,5000,-40.32,25.09,1.5,49.5
```

Σχήμα 3.13: Το CSV αρχείο όπως διαμορφώνεται κατά την αποθήκευση ενός πειράματος

Μέσω του “Import” γίνεται η μεταφόρτωση ενός CSV αρχείου από τη τοπική συσκευή στον διακομιστή της εφαρμογής. Το αρχείο οφείλει να τηρεί τη φόρμουλα που ακολουθούν τα CSV αρχεία των πειραμάτων για να γίνει αποδεκτό. Αυτή η λειτουργία εξυπηρετεί τον χρήστη όταν θέλει να τροποποιήσει το αρχείο ενός πειράματος (π.χ. να αφαιρέσει γραμμές μετρήσεων) στον υπολογιστή του και να το “ξανα-ανεβάσει” στην εφαρμογή.



Σχήμα 3.13: Το παράθυρο διαχείρισης όλων των αποθηκευμένων πειραμάτων.

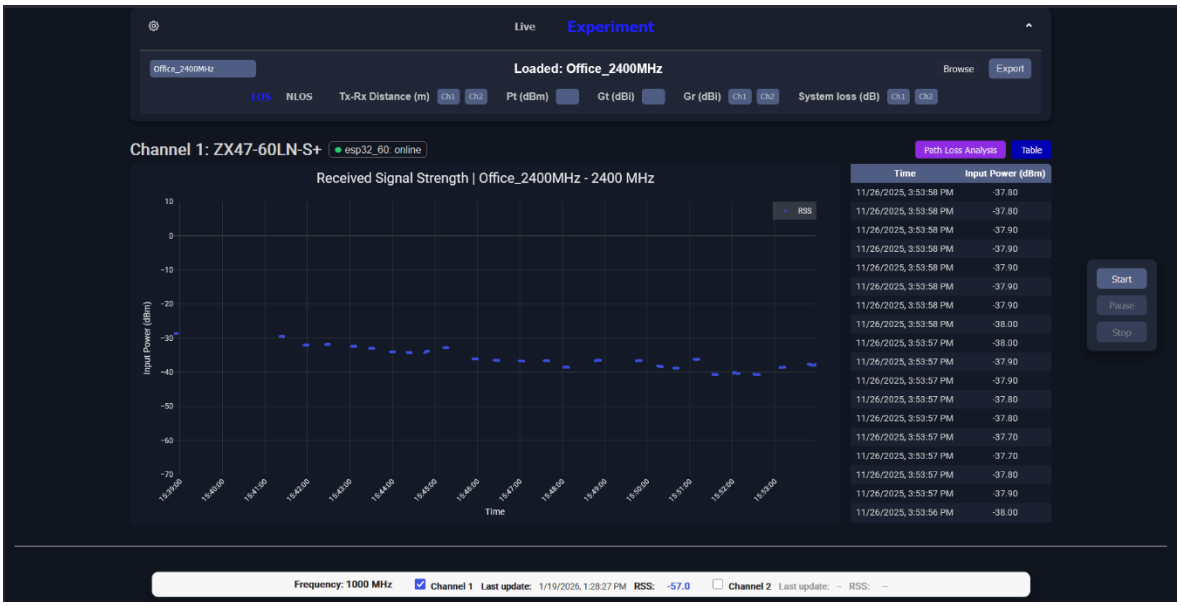
3.7.3 Εκτέλεση ενός πειράματος

Πριν ξεκινήσει ένα πείραμα πρέπει να έχει γίνει η κατάλληλη προετοιμασία της εφαρμογής. Πρωταρχικό βήμα είναι ο ορισμός ενός προφίλ βαθμονόμησης, ώστε να γνωρίζει το σύστημα ποια συχνότητα αφορούν οι μετρήσεις και πως θα γίνει η ακριβής μετατροπή της τάσης σε ισχύ σήματος. Επίσης, είναι σημαντικό να έχει επιλεχθεί από τη «μπάρα συσκευών» ο ανιχνευτής που θα πραγματοποιεί τις μετρήσεις, ώστε να ενεργοποιηθεί το αντίστοιχο κανάλι δεδομένων. Αν είναι επιλεγμένα και τα δύο κανάλια αλλά στην πραγματικότητα μόνο μία μονάδα ανιχνευτή είναι “online”, τότε η εφαρμογή καταλαβαίνει ότι μόνο το ένα κανάλι θα χρησιμοποιηθεί. Τέλος, πριν την έναρξη του πειράματος πρέπει να δοθεί ένα μοναδικό όνομα και να συμπληρωθούν οι παράμετροι του Link Budget. Με την ολοκλήρωση των παραπάνω βημάτων, η εφαρμογή επιτρέπει το πάτημα του “start”. Μία τυπική διαδικασία πειράματος μέτρησης της εξασθένησης ισχύος περιλαμβάνει την εξής διαδικασία:

- Πάτημα “start” και λήψη μερικών δειγμάτων στην αρχική απόσταση πομπού-δέκτη.
- “Pause” και αλλαγή της απόστασης στο σχετικό prompt.
- “Resume” και συνέχιση λήψης μετρήσεων.
- Επανάληψη των παραπάνω δύο βημάτων μέχρι τη τελική απόσταση.
- Πάτημα “stop” για την ολοκλήρωση και αποθήκευση των μετρήσεων.

Αν χρειάζεται τροποποιήσεις το σετ μετρήσεων, τότε ο χρήστης “κατεβάζει” το CSV αρχείο στον υπολογιστή του και έπειτα το ξανα-μεταφορτώνει με αντικατάσταση του προηγούμενου ή αποθήκευση με άλλο όνομα. Με την ολοκλήρωση του πειράματος εμφανίζεται ένα

πλήκτρο “Path Loss Analysis” πάνω από κάθε διάγραμμα. Αυτό εισάγει τον χρήστη στο παράθυρο ανάλυσης των πειραματικών μετρήσεων.



Σχήμα 3.14: Το περιβάλλον “Experiment” μετά την ολοκλήρωση ή τη φόρτωση ενός πειράματος.

3.8 Περιβάλλον ανάλυσης πειραμάτων

Η εφαρμογή ολοκληρώνεται με τον χώρο ανάλυσης των πειραματικών δεδομένων. Εδώ δίνεται η δυνατότητα μελέτης της εξασθένησης διαδρομής σήματος συναρτήσει της απόστασης πομπού - δέκτη. Αυτή περιλαμβάνει τη σχεδίαση της συνάρτησης σε γράφημα, τη σύγκριση με θεωρητικά μοντέλα διάδοσης σήματος και την εξαγωγή στατιστικών παραμέτρων που χαρακτηρίζουν το κανάλι διάδοσης. Όλη η επεξεργασία των δεδομένων λαμβάνει χώρα στον φυλλομετρητή και έχει υλοποιηθεί αποκλειστικά στο κώδικα της Javascript.

3.8.1 Απεικόνιση δεδομένων

Κατά το άνοιγμα του παραθύρου “Path Loss Analysis”, το γράφημα “Path-loss Vs.Distance” είναι ακόμη άδειο. Ο χρήστης καλείται να επιλέξει κάποιες παραμέτρους που αφορούν τη σχεδίαση της γραφικής:

- Raw/Binned data: Μπορεί να γίνει απεικόνιση όλων των καταγεγραμμένων μετρήσεων ή αυτές να ομαδοποιηθούν για κάθε απόσταση σε μία τιμή.

- Mean/Median: Η ομαδοποίηση ανά απόσταση μπορεί να γίνει είτε με τον υπολογισμό του μέσου όρου (mean) της εξασθένησης ή της διαμέσου (median).
- Linear/Log x-axis: Επιλογή απεικόνισης του οριζόντιου άξονα σε γραμμική ή λογαριθμική κλίμακα.

Με τη ρύθμιση των παραπάνω επιλογών και το πάτημα του “Compute” δημιουργείται η γραφική παράσταση σε ένα γράφημα Plotly. Δίνεται επίσης η δυνατότητα αλλαγής του τίτλου του γραφήματος και χρήση λευκού φόντου σε αυτό (light mode).

Σχήμα 3.15: Η γραφική παράσταση της εξασθένησης διαδρομής συναρτήσει της απόστασης στο παράθυρο ανάλυσης.

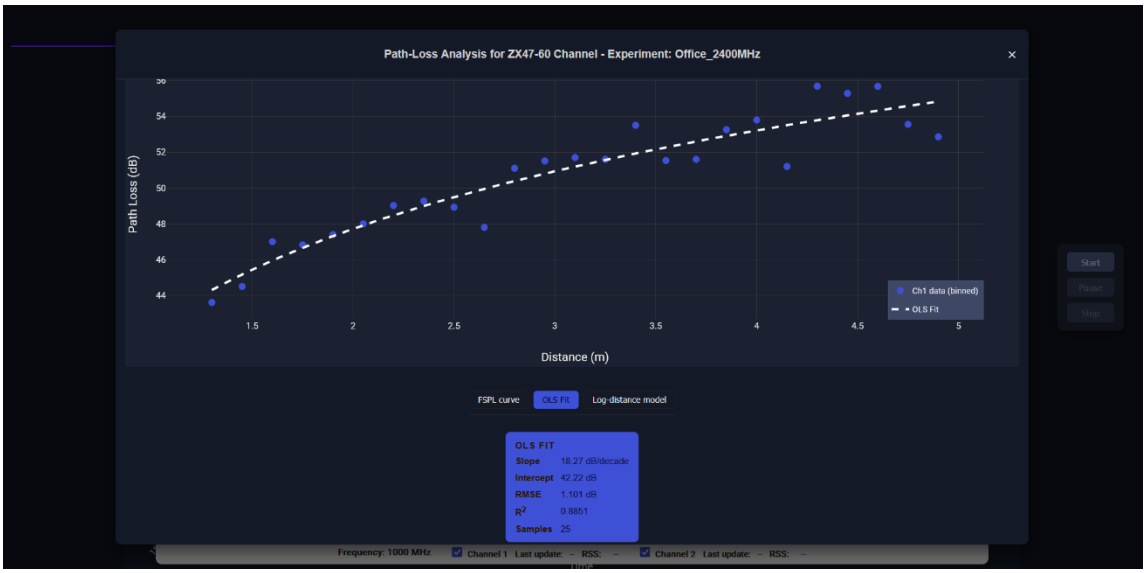
3.8.2 Προσαρμογή Θεωρητικών Μοντέλων

Κάτω από το πλαίσιο του γραφήματος βρίσκονται μερικές επιλογές σχετικά με τη προσαρμογή θεωρητικών καμπυλών πάνω στα θεωρητικά δεδομένα. Μαζί με τη προσθήκη κάθε καμπύλης εμφανίζεται και ένα πλαίσιο που συγκεντρώνει πληροφορίες της συνάρτησης και στατιστικά αποτελέσματα της προσαρμογής. Οι επιλογές περιλαμβάνουν τη προσθήκη των εξής:

- Προσαρμογή καμπύλης ελαχίστων τετραγώνων (OLS)
- Καμπύλη μοντέλου εξασθένησης διαδρομής ελεύθερου χώρου (FSPL)
- Προσαρμογή του μοντέλου εξασθένησης λογαριθμικής απόστασης (Log-distance PL)

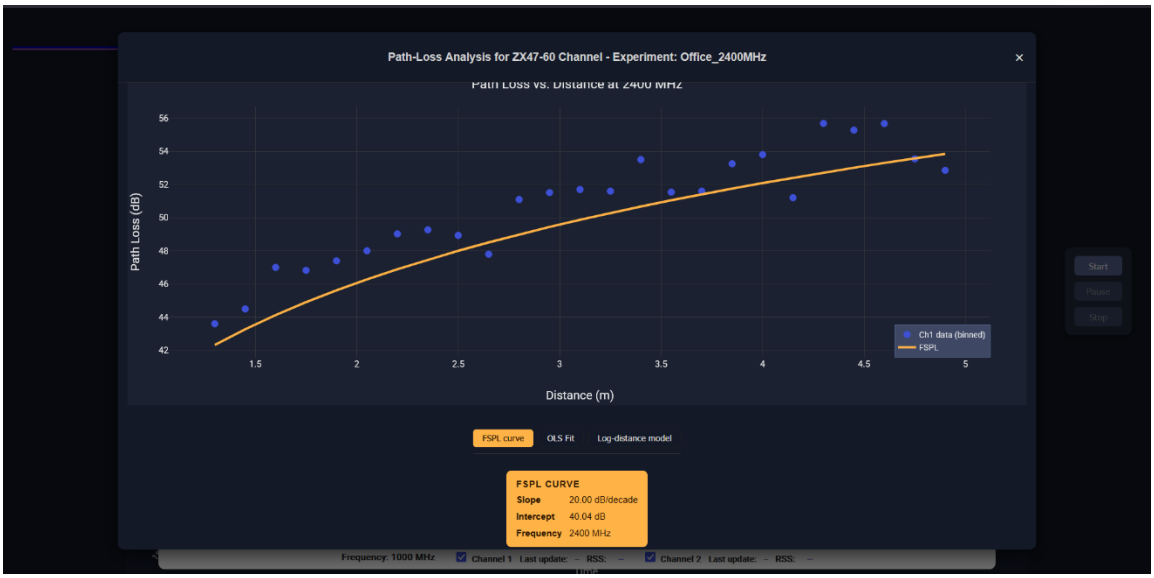
Μέσω του “OLS Fit”, η εφαρμογή προσαρμόζει μία γραμμική καμπύλη στα πειραματικά δεδομένα χρησιμοποιώντας τη μέθοδο των ελαχίστων τετραγώνων. Ως ανεξάρτητη μεταβλητή χρησιμοποιείται ο δεκαδικός λογάριθμος της απόστασης $log_{10}(d)$ και ως

εξαρτημένη η εξασθένηση διαδρομής. Στο πλαίσιο αποτελεσμάτων αναγράφεται ο σταθερός όρος και η κλίση της προκύπτουσας ευθείας, ενώ υπολογίζεται και ο συντελεστής προσδιορισμού R^2 και η ρίζα μέσου τετραγωνικού σφάλματος (RMSE).



Σχήμα 3.16: Προσαρμογή «καλύτερης» καμπύλης στα δεδομένα με τη μέθοδο ελαχίστων τετραγώνων και προβολή των αποτελεσμάτων στο "summary box".

Η επιλογή "FSPL curve" προσθέτει στο γράφημα τη θεωρητική καμπύλη όπως προκύπτει από το μοντέλο διάδοσης του ελεύθερου χώρου. Στον υπολογισμό των σημείων χρησιμοποιούνται τα δεδομένα απόστασης και συχνότητας. Στο αντίστοιχο πλαίσιο σημειώνονται οι παράμετροι της καμπύλης.



Σχήμα 3.17: Προσθήκη καμπύλης βάσει του μοντέλου εξασθένησης διαδρομής ελεύθερου χώρου (FSPL)

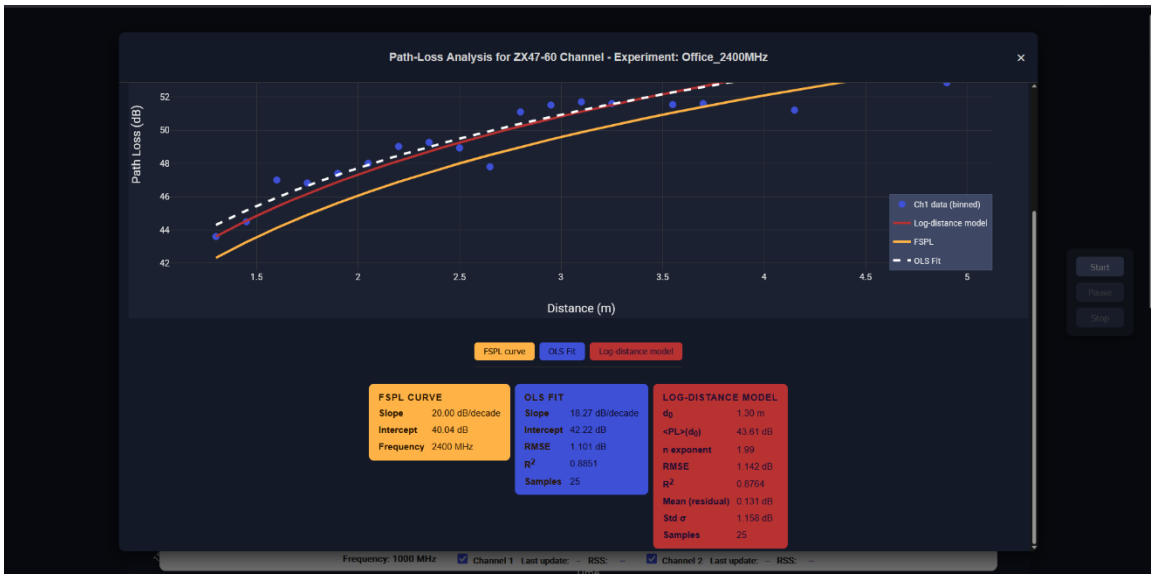
Τελευταία επιλογή είναι η “Log-distance model”, κατά την οποία πραγματοποιείται προσαρμογή του μοντέλου εξασθένησης λογαριθμικής απόστασης. Στο πάτημα της ανοίγει ένα μικρό πλαίσιο για τον ορισμό απαραίτητων μεταβλητών του μοντέλου. Ο χρήστης πρέπει να επιλέξει ποια τιμή θα χρησιμοποιηθεί ως μέση εξασθένηση διαδρομής σημείου αναφοράς $PL(d_0)$. Έχει τη δυνατότητα να εισάγει ο ίδιος μία τιμή απόστασης d_0 και εξασθένησης $PL(d_0)$, ωστόσο δίνονται και οι επιλογές να οριστεί αυτόματα το πρώτο σημείο των μετρήσεων ή η τιμή FSPL σε απόσταση 1 m, όπως συχνά συναντώνται στη βιβλιογραφία. Επίσης, δύναται να επιλέξει, αν επιθυμεί, να προχωρήσει με κανονική προσαρμογή του συντελεστή n στα δεδομένα ή να εισάγει ο ίδιος μια τιμή. Στη δεύτερη περίπτωση δεν συμβαίνει κάποιο fitting, απλώς σχεδιάζεται η καμπύλη σύμφωνα με τις δοθέντες παραμέτρους (χρήσιμο για δοκιμή θεωρητικών τιμών n). Τέλος, ο χρήστης μπορεί να προσθέσει στο μοντέλο τον συντελεστή σκίασης (shadowing) μέσω μιας τιμής τυπικής απόκλισης, εάν επιθυμεί να απεικονιστεί το διάστημα +1σ γύρω από τη καμπύλη.

Σχήμα 3.18: Το πλαίσιο εισαγωγής των παραμέτρων του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης.

Σχήμα 3.19: Η προσαρμογή και τα αποτελέσματα του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης.

Η προσαρμογή του μοντέλου βασίζεται στη γραμμική παλινδρόμηση, εφόσον η θεωρητική σχέση εκφράζεται ως γραμμική εξίσωση, με ανεξάρτητη μεταβλητή το $log_{10}(d)$. Η εφαρμογή ακολουθεί και εδώ τη μέθοδο των ελαχίστων τετραγώνων, με τη διαφορά όμως ότι ο σταθερός όρος είναι προκαθορισμένος και δε τίθεται προς υπολογισμό.

Το πλαίσιο σύνοψης συγκεντρώνει τα εξής αποτελέσματα βάσει της προσαρμογής του Log-distance μοντέλου: Το σημείο αναφοράς που χρησιμοποιήθηκε, ο προκύπτων εκθέτης n, οι συντελεστές RMSE και R^2, η τυπική απόκλιση σ και η μέση τιμή των υπολοίπων (residuals).



Σχήμα 3.20: Το περιβάλλον ανάλυσης με την προσθήκη όλων των καμπυλών.

Κεφάλαιο 4: Μελέτη της εξασθένησης διαδρομής σε εσωτερικό περιβάλλον

4.1 Εισαγωγή

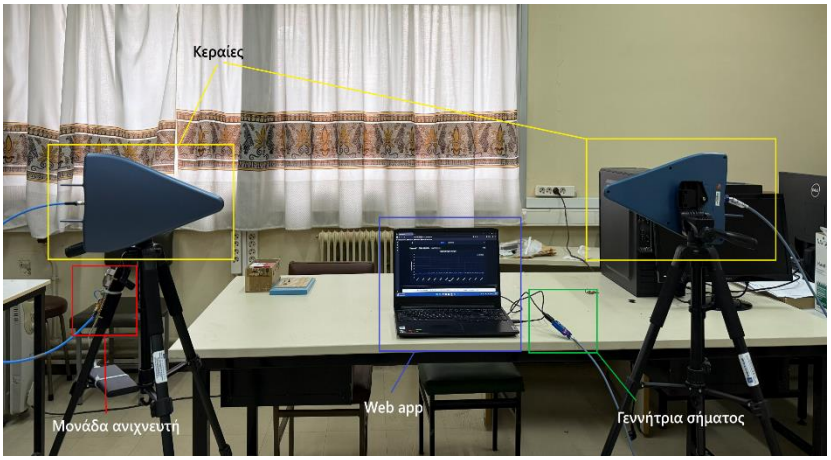
Μετά την ολοκλήρωση της υλοποίησης του συστήματος, ακολουθεί η δοκιμή του σε πραγματικές συνθήκες διάδοσης, με τη διεξαγωγή πειραμάτων μελέτης της εξασθένησης του RF σήματος. Ως εκ τούτου, πραγματοποιήθηκαν πειράματα μέτρησης της εξασθένησης διαδρομής μικροκυμάτων συναρτήσει της απόστασης, για συχνότητες 1, 2 και 5 GHz. Τα περιβάλλοντα που εξετάστηκαν αφορούν εσωτερικούς χώρους κτιρίων. Πιο συγκεκριμένα, επιλέχθηκαν τρεις εσωτερικοί χώροι του Εργαστηρίου Ηλεκτρονικής - Τηλεπικοινωνιών και Εφαρμογών του Τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων - (α) εργαστήριο, (β) γραφείο και (γ) διάδρομος - με σκοπό να παρουσιάζουν διαφοροποιήσεις ως προς τη γεωμετρία και την πυκνότητα αντικειμένων, ενώ ταυτόχρονα να παρέχουν επαρκή χώρο για μετρήσεις σε διαφορετικές αποστάσεις.

Η ανάλυση των μετρήσεων πραγματοποιήθηκε απευθείας στο περιβάλλον της εφαρμογής και περιλαμβάνει τη σύγκριση των αποτελεσμάτων με τα δύο θεωρητικά μοντέλα εξασθένησης διαδρομής, ελεύθερου χώρου και λογαριθμικής απόστασης. Βασικό παράγοντα σύγκρισης αποτελεί ο εκθέτης *n* που προκύπτει από την προσαρμογή του μοντέλου λογαριθμικής απόστασης, καθώς και η τυπική απόκλιση των δειγμάτων.

Όλα τα πειράματα ακολούθησαν μία σταθερή μεθοδολογία μετρήσεων. Σε κάθε χώρο, έγινε επιλογή ενός σημείου όπου οι κεραίες μπορούν να τοποθετηθούν σε οπτική ευθεία και υπάρχει ο κατάλληλος χώρος να μεταβάλλεται η απόσταση τους πάνω σε αυτή, για ένα διάστημα μερικών μέτρων. Ο πομπός περιλαμβάνει την προγραμματιζόμενη γεννήτρια RF σήματος μαζί με την κατευθυντική κεραία και έναν φορητό υπολογιστή. Ο δέκτης αποτελείται μόνο από την κεραία και τη μονάδα ανίχνευσης μαζί την τροφοδοσία της. Είναι σημαντικό να αναφερθεί ότι οι κεραίες HyperLOG 7060 τοποθετήθηκαν σε κάθετη διάταξη (όπως φαίνεται στο Σχήμα 4.1), με τη βοήθεια του τριπόδου, καθότι η ακτινοβολία στο επίπεδο ανύψωσης έχει μεγαλύτερο άνοιγμα. Συνεπώς, με την περιστροφή της κεραίας κατά 90 μοίρες, το επίπεδο ανύψωσης αποτελεί πλέον το αζιμουθιακό, προσφέροντας μεγαλύτερη ανοχή σε σφάλματα ευθυγράμμισης των κεραιών. Οι κεραίες βρίσκονταν σε ύψος 1.2 m. Τέλος, το Raspberry Pi τοποθετούνταν σε σημείο τέτοιο, ώστε η εμβέλεια του Wi-Fi του να καλύπτει τη μονάδα λήψης. Η πρόσβαση στην εφαρμογή πραγματοποιούνταν μέσω του φορητού υπολογιστή.

Στα πειράματα συχνότητας 5 GHz χρησιμοποιήθηκε ένας ενισχυτής, στην πλευρά του πομπού, καθώς η ισχύς του εκπεμπόμενου σήματος δεν επαρκούσε για τη διατήρηση της στα όρια γραμμικής λειτουργίας του ανιχνευτή. Η γεννήτρια SynthUSB3 μπορεί να αποδώσει μέγιστη ισχύ σήματος +6 dBm στα 5 GHz, η οποία λόγω της εντονότερης εξασθένησης που υφίσταται, τίθεται εκτός του δυναμικού εύρους του ZX47-60LN-S+ ανιχνευτή σε μεγάλες αποστάσεις. Εκτός αυτού, στη χαμηλή περιοχή ευαισθησίας εμφανίζονται παρεμβολές από

διάφορα άλλα σήματα (δίκτυο κινητής τηλεφωνίας, Wi-Fi) δυσχεραίνοντας τη λήψη ακριβούς μέτρησης. Για αυτό, έγινε προσθήκη του ενισχυτή Mini-Circuits ZX60-83LN-S+ μετά την έξοδο της γεννήτριας, τροφοδοτούμενος με 6V DC από εργαστηριακό τροφοδοτικό. Το τελικό σήμα εκπομπής ενισχύθηκε κατά ~21 dBm.



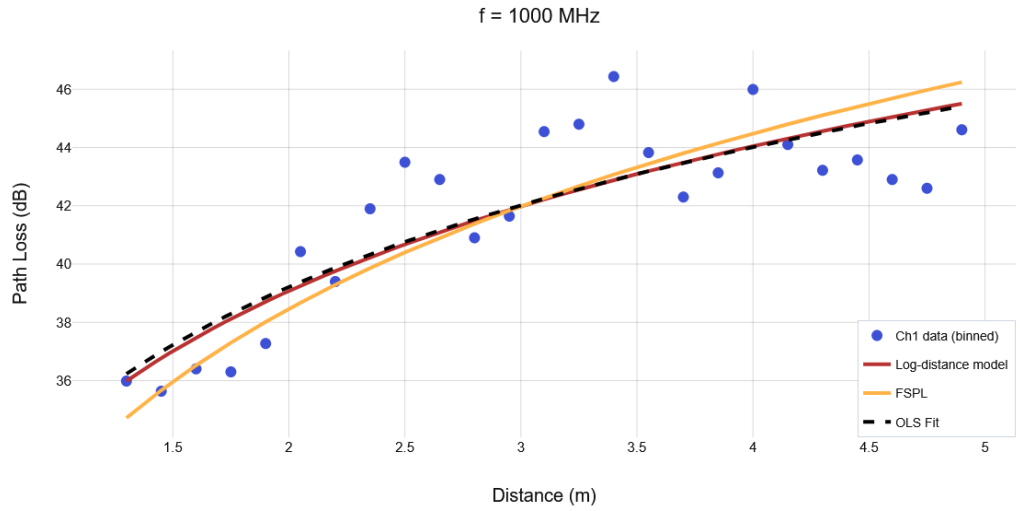
Σχήμα 4.1: Επιμέρους μονάδες της πειραματικής διάταξης.

4.2 Μετρήσεις σε χώρο εργαστήριου με οπτική επαφή (Indoor LoS Lab)

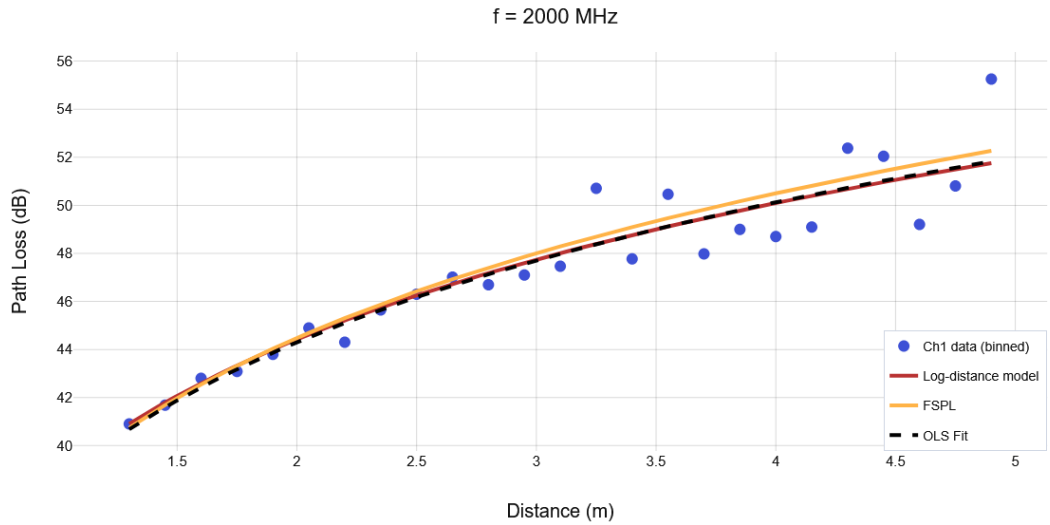
Το πρώτο σενάριο διάδοσης RF σήματος αφορά το απλούστερο παράδειγμα εσωτερικού χώρου, αυτό ενός άδειου δωματίου, τμήματος του εργαστηρίου Ηλεκτρονικής – Τηλεπικοινωνιών και Εφαρμογών. Οι μετρήσεις πραγματοποιήθηκαν για τις συχνότητες 1,2 και 5 GHz, σε αποστάσεις 1.3 έως 4.9 m, με βήμα 0.15 m. Οι κεραίες τοποθετήθηκαν στο κέντρο του δωματίου, με τη μετακίνηση του δέκτη να γίνεται πάνω σε ευθεία διατηρώντας ελεύθερο χώρο χωρίς παρεμβολές μεταξύ τους. Κατά τη διάρκεια των μετρήσεων και της καταγραφής του σήματος δεν υπήρχε κίνηση στο δωμάτιο αλλά ούτε αλλαγές στη διάταξη των αντικειμένων.


Σχήμα 4.2: Πειραματική διάταξη εντός του εργαστηρίου Ηλεκτρονικής – Τηλεπικοινωνιών και Εφαρμογών.

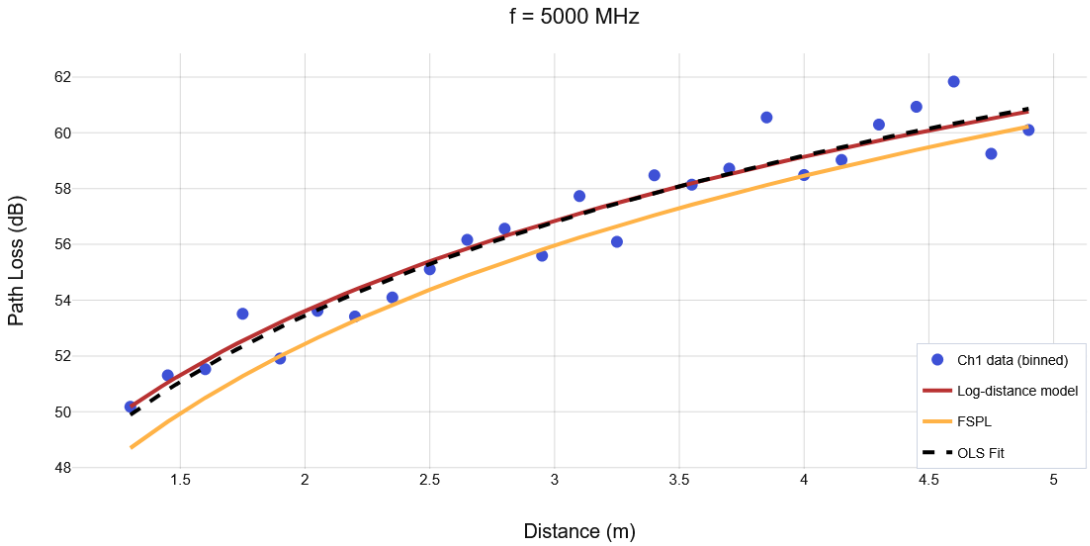
Η ανάλυση βασίστηκε στη μέση τιμή των δειγμάτων. Στο περιβάλλον της εφαρμογής γίνεται προσαρμογή της βέλτιστης ευθείας που περιγράφει τα δεδομένα βάσει της μεθόδου ελαχίστων τετραγώνων. Επίσης, εφαρμόστηκε το μοντέλο εξασθένησης διαδρομής λογαριθμικής απόστασης, όπου επιλέχθηκε ως σημείο αναφοράς η αρχική απόσταση των δεδομένων, δηλαδή το σημείο απόστασης 1.3 m. Τέλος, για τη σύγκριση με τη θεωρία διάδοσης στον ελεύθερο χώρο, έγινε προσθήκη και της καμπύλης FSPL. Η προσθήκη των παραπάνω καμπυλών έγινε απευθείας μέσω της εφαρμογής όπως περιγράφεται λεπτομερώς στην υποενότητα 3.8.2. Οι μετρήσεις και οι καμπύλες των μοντέλων αποτυπώνονται στα παρακάτω τρία γραφήματα και στους πίνακες 4.1α, 4.1β συγκεντρώνονται τα αποτελέσματα της ανάλυσης:



Σχήμα 4.1: Πειραματικά δεδομένα εξασθένησης διαδρομής στο εργαστήριο, για συχνότητα 1 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων.



Σχήμα 4.2: Πειραματικά δεδομένα εξασθένησης διαδρομής στο εργαστήριο, για συχνότητα 2 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων



Σχήμα 4.3 Πειραματικά δεδομένα εξασθένησης διαδρομής στο εργαστήριο, για συχνότητα 5 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων

f (GHz)	a_{FSPL}	b_{FSPL}	a_{fit}	b_{fit}	R^2
1	20	32.44	15.94	34.42	0.722
2	20	38.46	19.35	38.47	0.876
5	20	46.42	19.04	47.72	0.935

Πίνακας 4.1α: Οι παράμετροι της καμπύλης FSPL και τα αποτελέσματα προσαρμογής της βέλτιστης καμπύλης για τον χώρο του εργαστηρίου.

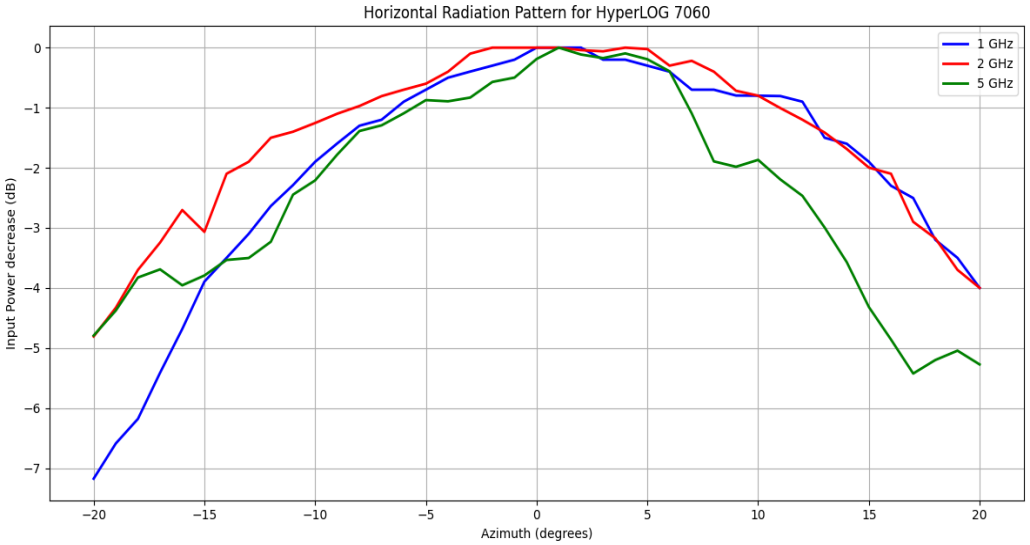
f (GHz)	PL(1.3)	n	R^2	σ (dB)
1	35.99	1.65	0.721	1.69
2	40.90	1.88	0.875	1.25
5	50.18	1.84	0.934	0.86

Πίνακας 4.1β: Τα αποτελέσματα της προσαρμογής του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης για τον χώρο του εργαστηρίου.

Από τις γραφικές παραστάσεις και τα αποτελέσματα των πινάκων προκύπτουν τα ακόλουθα συμπεράσματα: Η καμπύλη προσαρμογής των δεδομένων και η καμπύλη του μοντέλου λογαριθμικής απόστασης είναι σχεδόν πανομοιότυπη, με τον συντελεστή προσδιορισμού R^2 σχεδόν να ταυτίζεται. Εστιάζοντας στην προσαρμογή του μοντέλου, μεγαλύτερο R^2

προκύπτει στα 5 GHz και μικρότερο στο 1 GHz, γεγονός που συνδέεται άμεσα με την αύξηση της τυπικής απόκλισης με τη μείωση της συχνότητας.

Εστιάζοντας στην τυπική απόκλιση και τη διασπορά των μετρήσεων μεταξύ των συχνοτήτων, προκύπτουν δύο βασικές παρατηρήσεις. Πρώτον, οι τιμές της τυπικής απόκλισης είναι σημαντικά μικρότερες σε σχέση με τις τιμές (εντός του εύρους 3-8 dB) αντίστοιχων πειραμάτων εξασθένησης διαδρομής σε εσωτερικούς χώρους της βιβλιογραφίας [12-14]. Δεύτερον, παρατηρείται τάση μείωσης της διασποράς με την αύξηση της συχνότητας. Η χαμηλή διασπορά αποδίδεται, κυρίως, στη χρήση κατευθυντικών κεραιών. Οι κεραίες αυτές συγκεντρώνουν την εκπεμπόμενη και λαμβανόμενη ισχύ σε στενότερη γωνιακή περιοχή, με αποτέλεσμα να περιορίζεται η συνεισφορά των ανακλώμενων συνιστωσών του πεδίου. Κατά συνέπεια, μειώνεται ο αριθμός και η ισχύς των έμμεσων διαδρομών, γεγονός που οδηγεί σε μικρότερες διακυμάνσεις της λαμβανόμενης ισχύος λόγω συμβολής. Η παρατηρούμενη μείωση της διασποράς με την αύξηση της συχνότητας μπορεί να ερμηνευτεί από δύο παράγοντες. Αφενός, σε μεγαλύτερες συχνότητες η εξασθένηση των ανακλώμενων συνιστωσών είναι εντονότερη, αφετέρου σύμφωνα με το διάγραμμα ακτινοβολίας της κεραίας (Σχήμα 2.4) το άνοιγμα της δέσμης φαίνεται να είναι μικρότερο στις υψηλότερες συχνότητες. Προς επιβεβαίωση της παραπάνω ερμηνείας, πραγματοποιήθηκαν μετρήσεις της λαμβανόμενης ισχύος συναρτήσει της αζιμουθιακής γωνίας για τις συχνότητες 1, 2 και 5 GHz. Στο γράφημα του σχήματος 4.4 απεικονίζονται οι καμπύλες ισχύος ως προς τη γωνία για εύρος 20° γύρω από τη γωνία μέγιστης ακτινοβολίας. Για λόγους συγκρισιμότητας, οι τιμές ισχύος κανονικοποιήθηκαν ως προς τη μέγιστη τιμή κάθε συνόλου μετρήσεων.



Σχήμα 4.4: Πειραματικά δεδομένα της μεταβολής λαμβανόμενης ισχύος συναρτήσει της αζιμουθιακής γωνίας της κεραίας, για f = 1,2,5 GHz.

Σύμφωνα με το γράφημα, η πράσινη γραμμή, η οποία αναπαριστά τη μείωση της ισχύος για τα 5 GHz, εμφανίζει σημαντικά γρηγορότερη πτώση καθώς απομακρύνεται από τη γωνία μέγιστης ακτινοβολίας. Αυτό επιβεβαιώνει τη διατύπωση ότι ο λοβός ακτινοβολίας της κεραίας είναι στενότερος στα 5 GHz από ότι στα 1 και 2 GHz. Συνεπώς, οι έμμεσες διαδρομές λόγω ανάκλασης και περίθλασης είναι λιγότερες και ταυτόχρονα η απευθείας διαδρομή υφίσταται μικρότερη επίδραση.

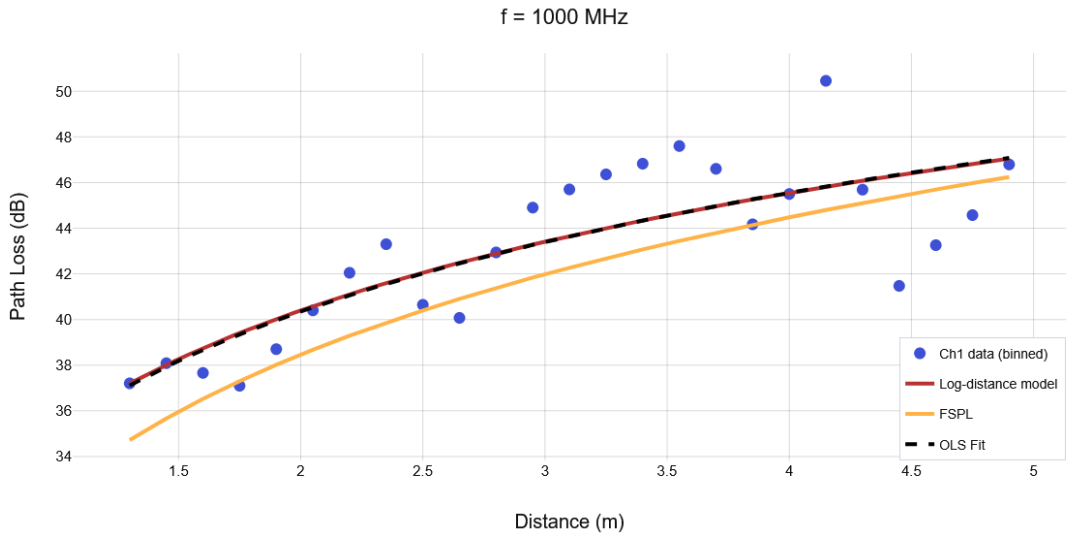
Συνεχίζοντας την ανάλυση, η προσαρμογή του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης και για τις τρεις συχνότητες έδωσε εκθέτη εξασθένησης διαδρομής *n* μικρότερο του 2, που προβλέπει το μοντέλο εξασθένησης διαδρομής ελεύθερου χώρου. Όπως είναι αναμενόμενο, ένας εσωτερικός χώρος που περιβάλλεται από κοντινά τοιχώματα, σε συνθήκες οπτικής επαφής, λειτουργεί ως κυματοδηγός λόγω των πολλών ανακλάσεων που προστίθενται εποικοδομητικά στο βασικό σήμα. Αυτό συνεπάγει ότι, η ισχύς του λαμβανόμενου σήματος φθίνει βραδύτερα συναρτήσει της απόστασης. Παράλληλα, η επίδραση της κεραίας που σχολιάστηκε προηγουμένως φαίνεται να έχει άμεση επίδραση στο ρυθμό εξασθένησης, καθώς στο 1 GHz, όπου η διασπορά είναι μεγαλύτερη, διαπιστώνεται εντονότερο το κυματοδηγικό φαινόμενο.

4.3 Μετρήσεις σε χώρο γραφείου με οπτική επαφή (Indoor LoS Office)

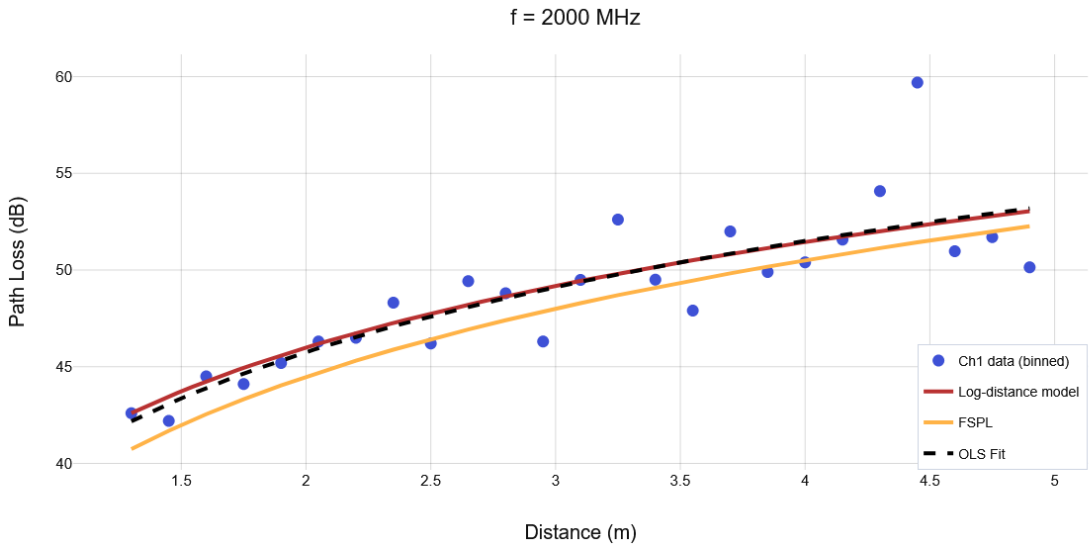
Ο επόμενος εσωτερικός χώρος που μελετάται είναι το περιβάλλον γραφείου. Πρόκειται για ένα δωμάτιο ανάλογων διαστάσεων με το προηγούμενο και με σαφώς πυκνότερη παρουσία αντικειμένων (γραφεία, υπολογιστές, εργαστηριακός εξοπλισμός κ.α.). Διατηρήθηκε η ίδια πειραματική διαδικασία με μετρήσεις σε αποστάσεις στο διάστημα [1.3, 4.9] m και για συχνότητες 1, 2 & 5 GHz. Οι κεραίες τοποθετήθηκαν στο κέντρο του χώρου, σε ευθεία, όσο το δυνατόν απομακρυσμένες από τα έπιπλα, με ελεύθερη περιοχή διάδοσης μεταξύ Tx και Rx.

Σχήμα 4.3: Η πειραματική διάταξη στο εσωτερικό χώρο του γραφείου.

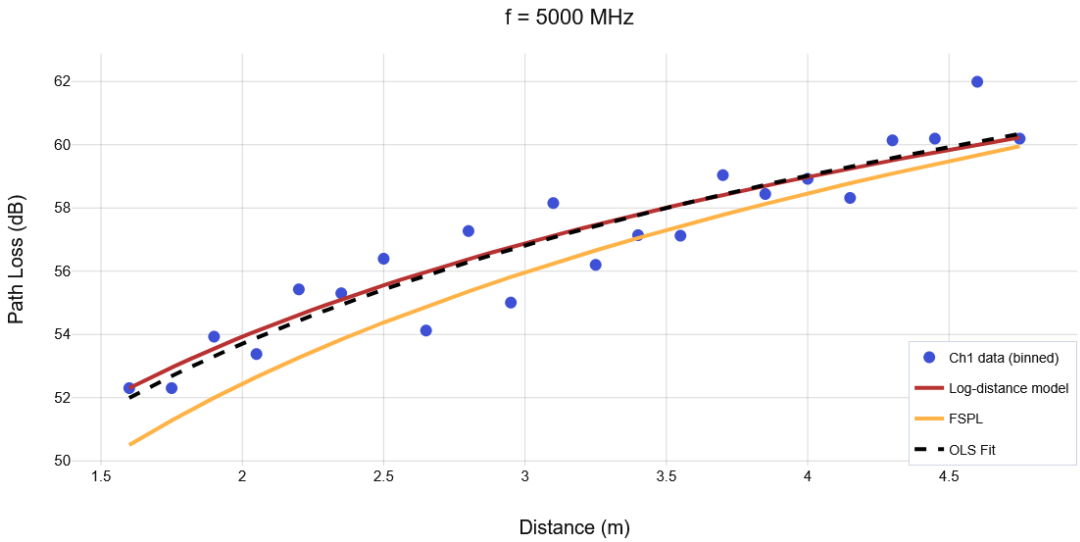
Τα πειραματικά δεδομένα με τις προσαρμοσμένες καμπύλες των μοντέλων αποτυπώνονται στα παρακάτω γραφήματα, για κάθε συχνότητα. Η ανάλυση που ακολουθήθηκε είναι ίδια με εκείνη του προηγούμενου πειράματος, με τα αποτελέσματα της να παρουσιάζονται στους πίνακες 4.2α και 4.2β.



Σχήμα 4.5: Πειραματικά δεδομένα εξασθένησης διαδρομής στο γραφείο, για συχνότητα 1 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων.



Σχήμα 4.6: Πειραματικά δεδομένα εξασθένησης διαδρομής στο γραφείο, για συχνότητα 2 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων.



Σχήμα 4.7: Πειραματικά δεδομένα εξασθένησης διαδρομής στο γραφείο, για συχνότητα 5 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων.

f (GHz)	a_{FSPL}	b_{FSPL}	a_{fit}	b_{fit}	R^2
1	20	32.44	17.30	35.14	0.651
2	20	38.46	19.11	40.00	0.709
5	20	46.42	17.67	48.39	0.874

Πίνακας 4.2α: Οι παράμετροι της καμπύλης FSPL και τα αποτελέσματα προσαρμογής της βέλτιστης καμπύλης για τον χώρο του γραφείου.

f (GHz)	PL(1.3)	n	R^2	σ (dB)
1	37.20	1.71	0.651	2.16
2	42.60	1.81	0.707	2.09
5	52.30	1.68	0.871	0.97

Πίνακας 4.2β: Τα αποτελέσματα της προσαρμογής του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης για τον χώρο του γραφείου

Συγκρίνοντας τα αποτελέσματα της προσαρμογής του μοντέλου λογαριθμικής απόστασης με αυτά της προσαρμογής της καμπύλης ελαχίστων τετραγώνων, παρατηρείται συμφωνία τόσο ως προς τον ρυθμό εξασθένησης όσο και στη συμπεριφορά των δεδομένων ανάμεσα στις συχνότητες. Οι τιμές του εκθέτη εξασθένησης διαδρομής κυμαίνονται κάτω από τη θεωρητική τιμή ελεύθερου χώρου (<2) και εντός του εύρους [1.6, 1.8] που προτείνεται συχνά στη βιβλιογραφία. Η τυπική απόκλιση εμφανίζει ανάλογη συμπεριφορά με αυτή του άδειου δωματίου, αφού είναι μεγαλύτερη στις χαμηλές συχνότητες, από ότι στα 5 GHz. Βέβαια οι τιμές παραμένουν σε αρκετά χαμηλά επίπεδα, ιδιαίτερα στην περίπτωση των 5 GHz, γεγονός που αποδόθηκε στην κατευθυντική φύση των κεραιών και στη μεταβλητότητα του λοβού ακτινοβολίας κατά μήκος των συχνοτήτων. Συγκρίνοντας τα δύο περιβάλλοντα, η τυπική απόκλιση είναι ελαφρώς μεγαλύτερη στις μετρήσεις του γραφείου, εξαιτίας του πυκνότερου σε αντικείμενα περιβάλλοντος που ενισχύει την παρουσία έμμεσων διαδρομών.

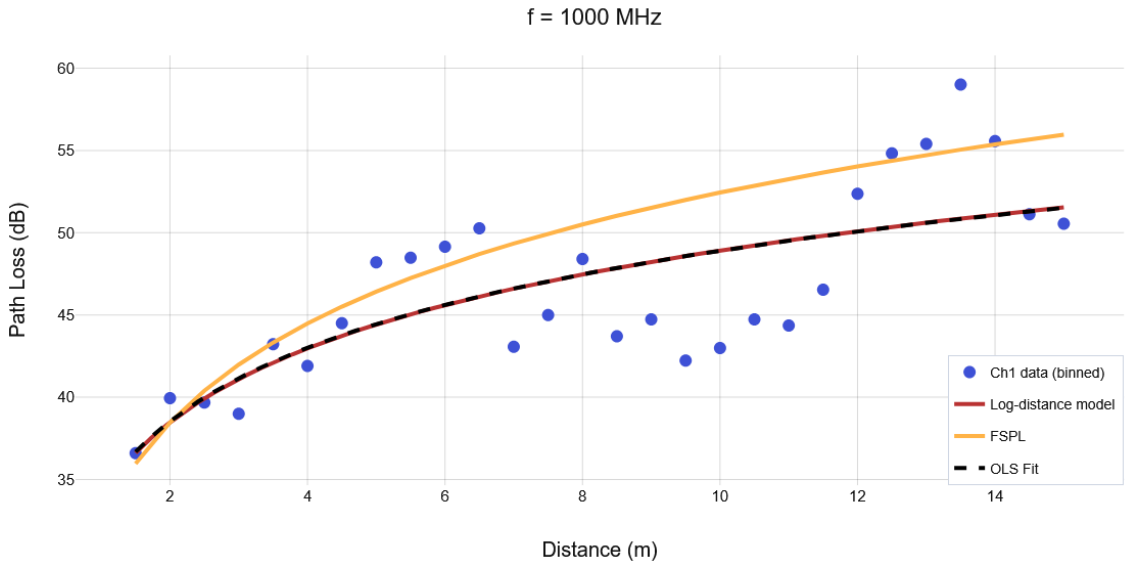
Εστιάζοντας στο μοντέλο εξασθένησης διαδρομής ελεύθερου χώρου παρατηρείται ότι η καμπύλη βρίσκεται 1-2 dB χαμηλότερα στις μικρές αποστάσεις σε σχέση με την καμπύλη προσαρμογής των πειραματικών δεδομένων. Εμφανίζεται ως συστηματικό σφάλμα και στις τρεις γραφικές παραστάσεις, οπότε μπορεί να αφορά μία απόκλιση στις παραμέτρους link budget που να οδηγεί σε υψηλότερη εκτίμηση της εξασθένησης διαδρομής. Ειδικότερα, το πραγματικό κέρδος των κεραιών δεν παραμένει σταθερό στα 5 dBi αλλά παρουσιάζει μεταβολές ανάλογα με τη συχνότητα. Ομοίως, οι απώλειες συστήματος, που οφείλονται στα καλώδια και στους συνδέσμους που χρησιμοποιούνται και για τις οποίες δόθηκε προσεγγιστικά σε όλα τα πειράματα η τιμή 3 dB, επηρεάζονται επίσης από τη συχνότητα του σήματος.

4.4 Μετρήσεις σε διάδρομο με οπτική επαφή (Indoor LoS Corridor)

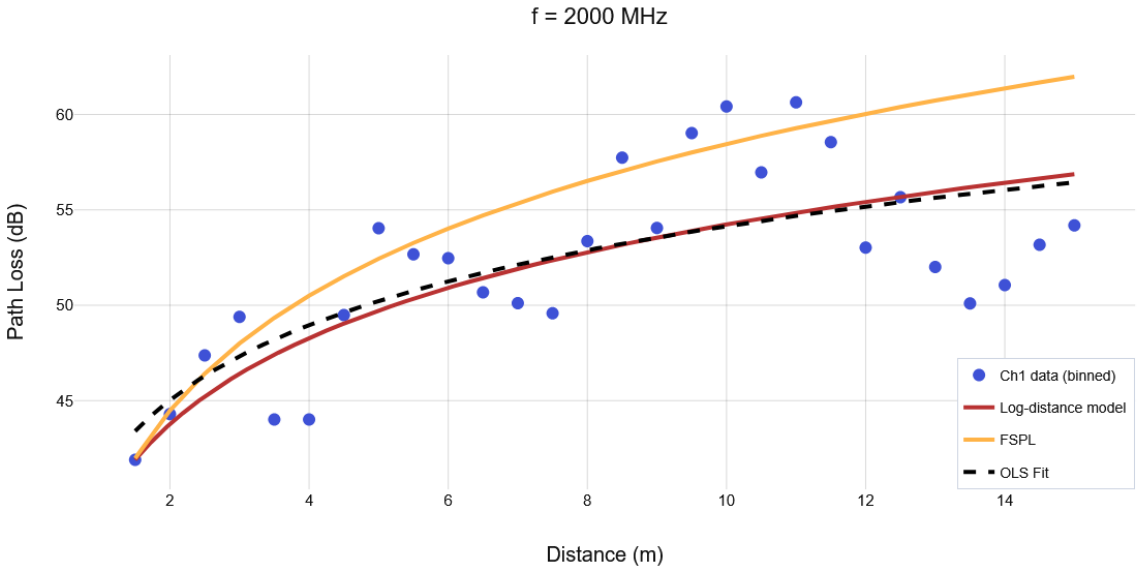
Το τελευταίο σετ πειραμάτων αφορά το σενάριο διάδοσης RF σήματος σε διάδρομο εντός του κτιρίου. Το περιβάλλον θα μπορούσε να θεωρηθεί πιο περίπλοκο σε σχέση με ένα κλασικό διάδρομο, καθότι στη μία πλευρά βρίσκονται διαδοχικές κολώνες που μπορεί να δημιουργούν περιοχές σκίασης στο σήμα (Σχήμα 4.4). Στην προκειμένη περίπτωση οι μετρήσεις έγιναν για ένα μεγαλύτερο διάστημα, από 1.5 έως 15 m, με βήμα 0.5 m. Ο πομπός τοποθετήθηκε στην αρχή του διαδρόμου, ενώ ο δέκτης ήταν ελεύθερος προς μετακίνηση πάνω στην οπτική ευθεία.

Σχήμα 4.4: Η πειραματική διάταξη σε διάδρομο εντός του Τμήματος Φυσικής

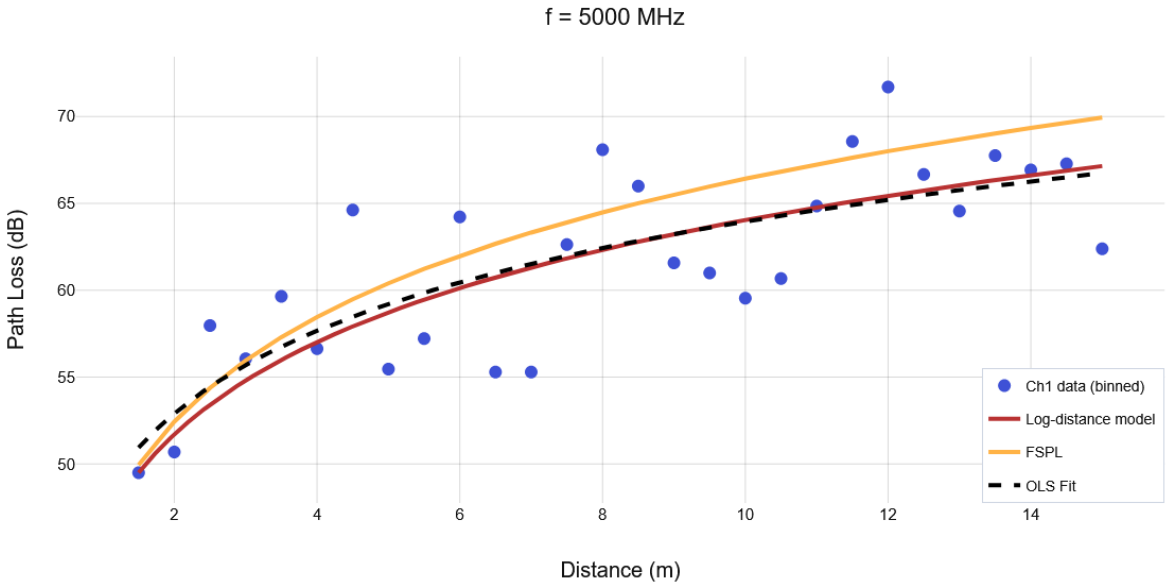
Στα γραφήματα των σχημάτων 4.8 έως 4.10 αποτυπώνονται οι πειραματικές μετρήσεις για κάθε συχνότητα μαζί με τις προσαρμοσμένες σε αυτά καμπύλες. Στους πίνακες 4.3α και 4.3β δίνονται οι μετρήσεις και τα αποτελέσματα της ανάλυσης για τον χώρο του διαδρόμου.



Σχήμα 4.8: Πειραματικά δεδομένα εξασθένησης διαδρομής στον διάδρομο, για συχνότητα 1 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων



Σχήμα 4.9: Πειραματικά δεδομένα εξασθένησης διαδρομής στον διάδρομο, για συχνότητα 2 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων



Σχήμα 4.10: Πειραματικά δεδομένα εξασθένησης διαδρομής στον διάδρομο, για συχνότητα 5 GHz και οι προσαρμοσμένες καμπύλες των μοντέλων ελεύθερου χώρου και λογαριθμικής απόστασης καθώς και της καμπύλης των δεδομένων

f (GHz)	a_{FSPL}	b_{FSPL}	a_{fit}	b_{fit}	R^2
1	20	32.44	14.83	34.07	0.543
2	20	38.46	13.04	41.11	0.525
5	20	46.42	15.80	48.16	0.603

Πίνακας 4.3α: Οι παράμετροι της καμπύλης FSPL και τα αποτελέσματα προσαρμογής της βέλτιστης καμπύλης για τον χώρο του διαδρόμου.

f (GHz)	PL(1.3)	N	R^2	σ (dB)
1	36.61	1.49	0.543	3.76
2	41.90	1.50	0.512	3.46
5	49.50	1.76	0.593	3.58

Πίνακας 4.3β: Τα αποτελέσματα της προσαρμογής του μοντέλου εξασθένησης διαδρομής λογαριθμικής απόστασης για τον χώρο του διαδρόμου

Παρατηρώντας τις γραφικές παραστάσεις διαπιστώνεται άμεσα ότι οι μετρήσεις έχουν μεγαλύτερη διασπορά σε όλες τις συχνότητες συγκριτικά με τα προηγούμενα δύο πειράματα. Η στενόμακρη διάταξη του διαδρόμου δημιουργεί πολλές ανακλώμενες δέσμες και τις κατευθύνει κατά μήκος του, δημιουργώντας έντονα φαινόμενα ενισχυτικής και καταστροφικής συμβολής. Ως εκ τούτου, παρουσιάζονται πολλές διακυμάνσεις στις μετρήσεις ισχύος (υψηλότερη τυπική απόκλιση & χαμηλότερος συντελεστής R^2). Και στις τρεις συχνότητες, η τιμή της τυπικής απόκλισης κυμαίνεται στα 3.5 dB - με το σήμα των 5 GHz να μη διαφοροποιείται αυτή τη φορά σε αυτόν τον παράγοντα. Πιθανή εξήγηση σε αυτό αποτελεί η επίδραση του χώρου. Τα τοιχώματα του διαδρόμου βρίσκονται αρκετά κοντά στις κεραίες με αποτέλεσμα το λίγο μικρότερο άνοιγμα της δέσμης ακτινοβολίας να μην μετριάζει τις ανακλώμενες διαδρομές.

Όσον αφορά τον εκθέτη εξασθένησης διαδρομής, αυτός διατηρείται και στις τρεις συχνότητες μικρότερος του 2 και μάλιστα στις δύο χαμηλότερες λαμβάνει μία αρκετά μικρή τιμή, $n \approx 1.5$ dB. Το κυματοδηγικό φαινόμενο είναι εντονότερο στο περιβάλλον του διαδρόμου, όπου οι πολλαπλές διαδρομές του σήματος «αναγκάζονται» να διαδίδονται κατά μήκος του, συσσωρεύοντας την ηλεκτρομαγνητική ακτινοβολία σε μία κατεύθυνση. Συνεπώς, η ισχύς του λαμβανόμενου σήματος φθίνει με μικρότερο βαθμό συναρτήσει της απόστασης.

Κεφάλαιο 5: Συμπεράσματα & Μελλοντική εργασία

5.1 Συμπεράσματα

Στην παρούσα μεταπτυχιακή διπλωματική εργασία υλοποιήθηκε ένα ολοκληρωμένο σύστημα μέτρησης της ισχύος RF σήματος και μελέτης της εξασθένησης διαδρομής μέσω μιας ειδικά σχεδιασμένης διαδικτυακής πλατφόρμας. Το σύστημα επιτρέπει στον χρήστη να διαχειρίζεται τη ροή δεδομένων για έναν επιλεγμένο ανιχνευτή ισχύος σήματος, να πραγματοποιεί γρήγορα πειράματα με μετρήσεις εξασθένησης διαδρομής συναρτήσει της απόστασης, να τα αποθηκεύει και να προβαίνει σε άμεση ανάλυση τους, με προσαρμογή θεωρητικών μοντέλων διάδοσης.

Προς επίτευξη αυτού, αναπτύχθηκαν δύο εξειδικευμένες πλακέτες μέτρησης, βασιζόμενες σε μικροελεγκτές ESP για ασύρματη λειτουργία και σε ανιχνευτές ισχύος. Οι δύο ανιχνευτές που επιλέχθηκαν καλύπτουν μία ευρεία περιοχή ραδιοσυχνοτήτων, από 10 MHz έως 8 GHz, προσφέροντας παρόμοιο δυναμικό εύρος ~60 dBm, ο καθένας όμως σε διαφορετικά επίπεδα ισχύος, καλύπτοντας συνδυαστικά μία εκτενή περιοχή από -55 dBm έως +20 dBm. Κάθε πλακέτα αποτελεί μία ανεξάρτητη πηγή ανίχνευσης, η οποία αποστέλλει τα δεδομένα σε έναν μικροϋπολογιστή Raspberry Pi, μέσω τοπικού δικτύου Wi-Fi που ο ίδιος δημιουργεί. Μέσα σε αυτόν αναπτύχθηκε και φιλοξενείται η εφαρμογή διαδικτύου που διαχειρίζεται τις μετρήσεις, η λογική της οποίας εξυπηρετείται από το ελαφρύ framework του flask. Με την αξιοποίηση του πρωτοκόλλου MQTT έγινε απευθείας διασύνδεση των μονάδων μέτρησης και του περιηγητή ιστού του χρήστη, δίχως τη δημιουργία κόμβων στον server.

Η απεικόνιση των δεδομένων γίνεται μέσα από το περιβάλλον της εφαρμογής που είναι προσβάσιμο από οποιονδήποτε σύγχρονο browser. Με την επιλογή της κατάλληλης καμπύλης βαθμονόμησης επιτυγχάνεται ακριβής παρουσίαση της τιμής ισχύος σήματος που μετράται στον ανιχνευτή σε πραγματικό χρόνο. Οι δυνατότητες της εφαρμογής επεκτείνονται με την εκτέλεση πειραμάτων καταγραφής της ισχύος συναρτήσει της απόστασης πομπού-δέκτη, προς μελέτη της εξασθένησης διαδρομής σε διάφορα σενάρια διάδοσης σήματος. Επίσης, για τον χαρακτηρισμό του καναλιού γίνεται άμεση ανάλυση των πειραματικών δεδομένων με προσαρμογή δύο διαδεδομένων θεωρητικών μοντέλων διάδοσης - αυτό της εξασθένησης διαδρομής ελεύθερου χώρου και της εξασθένησης διαδρομής λογαριθμικής απόστασης.

Το σύστημα δοκιμάστηκε και αξιολογήθηκε υπό πραγματικές συνθήκες, με την εκτέλεση πειραμάτων εξασθένησης RF σήματος, συχνότητας 1, 2 & 5 GHz, σε σενάρια διάδοσης εσωτερικών χώρων. Για τη μελέτη της εξάρτησης της εξασθένησης διαδρομής από την απόσταση, επιλέχθηκαν τρία διαφορετικά περιβάλλοντα του Πανεπιστημίου: ένα άδειο δωμάτιο, ένα γραφείο και ένας διάδρομος. Στην πειραματική ανάλυση έγινε σύγκριση των πειραματικών δεδομένων με το μοντέλο εξασθένησης διαδρομής ελεύθερου χώρου, ενώ έγινε ερμηνεία αυτών βάσει του μοντέλου λογαριθμικής απόστασης. Μέσα από τις

μετρήσεις, διαπιστώθηκε ότι ο εκθέτης εξασθένησης *n* κυμαίνεται σε τιμές κάτω του 2 – της αναμενόμενης τιμής για τον ελεύθερο χώρο – και ως επί το πλείστον μέσα στο εύρος [1.6, 1.8] που αναφέρεται στη βιβλιογραφία για LOS συνθήκες σε εσωτερικούς χώρους. Η γεωμετρία του περιβάλλοντος κατέχει καθοριστικό ρόλο στον ρυθμό εξασθένησης, καθώς οι μετρήσεις και των τριών συχνοτήτων παρουσιάζουν παρόμοιες μεταβολές από χώρο σε χώρο. Μεγαλύτερη διαφοροποίηση διαπιστώθηκε στις μετρήσεις του διαδρόμου, όπου η διακύμανση τους ήταν αισθητά μεγαλύτερη, 1.5-2.5 dB μεγαλύτερη τυπική απόκλιση σε σχέση με τις μετρήσεις γραφείου. Παράλληλα, ο ρυθμός εξασθένησης έλαβε τις χαμηλότερες τιμές n~1.5 για συχνότητες 1 και 2 GHz. Συνολικά, το μοντέλο λογαριθμικής απόστασης ακολουθώντας μία απλή φόρμουλα προσέφερε ικανοποιητική περιγραφή της εξασθένησης διαδρομής του σήματος μεταξύ των συχνοτήτων στις υπό διερεύνηση συνθήκες. Ωστόσο, το γεγονός ότι δε συνεκτιμά περιβαλλοντικούς παράγοντες και διατάξεις μπορεί να μειώσει την αξιοπιστία του σε πιο σύνθετους χώρους με εμπόδια μεταξύ των ζεύξεων.

Συνοψίζοντας, το σύστημα αποτελεί μία ολοκληρωμένη και εύχρηστη λύση για τη μελέτη απωλειών RF σήματος. Βασικότερα χαρακτηριστικά είναι η αυτονομία και η φορητότητα του. Πέρα από την τροφοδοσία του Raspberry Pi και της πλακέτας ανίχνευσης, το σύστημα δε βασίζεται στη χρήση τρίτων μέσων για τη λειτουργία του, ενώ η πρόσβαση στην εφαρμογή μπορεί να γίνει ακόμη και με σύνδεση στο τοπικό δίκτυο Wi-Fi του Raspberry Pi. Μάλιστα, λόγω της διαδικτυακής υπόστασης της εφαρμογής, αυτή είναι άμεσα διαθέσιμη από τον browser οποιασδήποτε συσκευής. Παράλληλα, λόγω της ασύρματης διασύνδεσης τόσο του χρήστη όσο και των υποσυστημάτων, η προσαρμογή της διάταξης στον υπό μελέτη χώρο καθίσταται εύκολη υπόθεση. Η λειτουργικότητα, βέβαια, πλαισιώνεται με το προσβάσιμο περιβάλλον της web εφαρμογής, η οποία δομείται σε ένα φιλικό προς χρήση UI, με σαφή και καθοδηγητικό χαρακτήρα προς την πραγματοποίηση γρήγορων πειραμάτων. Άλλωστε, η έρευνα σχετικά με τον χαρακτηρισμό ασύρματων δικτύων και τη βελτίωση των μοντέλων διάδοσης, ιδιαίτερα σε εσωτερικούς χώρους, απαιτεί εύχρηστα μέσα με ακρίβεια στη μέτρηση για βέλτιστη αποδοτικότητα.

5.2 Βελτιώσεις & Μελλοντική εργασία

Αν και στην τρέχουσα έκδοση του συστήματος, παρέχεται ένα πλήρως λειτουργικό πακέτο καταγραφής και μελέτης RF σήματος, μπορούν να γίνουν αναβαθμίσεις για αύξηση της εμβέλειας, της ευελιξίας και της ακρίβειας:

1. Επέκταση εμβέλειας

Μία κρίσιμη τεχνική βελτίωση σε επίπεδο συστήματος αφορά την άρση των περιορισμών εμβέλειας που θέτει η χρήση ενός τοπικού δικτύου Wi-Fi. Για την επικοινωνία των μονάδων ανίχνευσης με τον κεντρικό υπολογιστή σε μεγαλύτερες αποστάσεις ή σύνθετα περιβάλλοντα, το σύστημα θα μπορούσε να υιοθετήσει ένα πρωτόκολλο επικοινωνίας χαμηλής ισχύος και μεγάλης εμβέλειας, όπως το LoRaWAN. Με αυτή την επέκταση, η

εμβέλεια λήψης μετρήσεων θα μπορούσε να καλύπτει υπαίθριους ή εσωτερικούς χώρους, χωρίς την εξάρτηση από ένα κοντινό Access Point.

2. Εξέλιξη του λογισμικού

Στο επίπεδο λογισμικού, η διαδικτυακή εφαρμογή θα μπορούσε να εμπλουτιστεί με περισσότερες παραμετροποιήσεις και διευκολύνσεις, προς βελτίωση της εμπειρίας χρήστη. Αυτές μπορεί να αφορούν περισσότερες επιλογές τροποποίησης των γραφημάτων, δυνατότητα επεξεργασίας των δεδομένων εντός της πλατφόρμας κ.α.

3. Επέκταση μοντέλων

Μία ακόμη φυσική εξέλιξη της ανάλυσης των μετρήσεων θα ήταν και η επέκταση της βιβλιοθήκης μοντέλων προς προσαρμογή με την προσθήκη πιο σύνθετων μοντέλων διάδοσης που λαμβάνουν υπόψιν χωρικούς παράγοντες (π.χ. το μοντέλο ITU-R indoor). Επίσης, η δυνατότητα παραγωγής δισδιάστατων χαρτών ραδιοκάλυψης (heatmaps), μέσω της συσχέτισης των μετρήσεων ισχύος με δεδομένα θέσης ενός χώρου, θα ενίσχυε τη χρήση του συστήματος ως εργαλείο σχεδιασμού και βελτιστοποίησης δικτύων.

4. Υποστήριξη πολλών χρηστών

Τέλος, ένα σημαντικό βήμα για την εφαρμογή θα ήταν η μετάβαση από το πειραματικό “setup” της τρέχουσας μορφής, σε μία πλατφόρμα ικανή να εξυπηρετεί πολλούς χρήστες. Αυτό θα απαιτούσε τη μεταφορά της εφαρμογής από το τοπικό περιβάλλον του Raspberry Pi σε αποδοτικότερους διακομιστές ή ακόμη και σε cloud υποδομές, τη διαμόρφωση προς διαχείριση και οργάνωση μεγάλου όγκου δεδομένων και την ενίσχυση της ασφάλειας στην πρόσβαση και στη μεταφορά δεδομένων.

Βιβλιογραφία

[1] T. Domínguez-Bolaño, O. Campos, V. Barral, C. J. Escudero, and J. A. García-Naya, "An overview of IoT architectures, technologies, and existing open-source projects," *Internet of Things*, vol. 20, p. 100626, Oct. 2022, doi: 10.1016/j.iot.2022.100626.

[2] M. Jazayeri, "Some Trends in Web Application Development," in *Future of Software Engineering (FOSE'07)*, 2007, pp. 199-213, doi: 10.1109/FOSE.2007.26.

[3] C. A. Balanis, *Advanced Engineering Electromagnetics*, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.

[4] D. J. Griffiths, *Εισαγωγή στην Ηλεκτροδυναμική*, Β' αναθ. έκδ., μετάφρ. Σ. Αρβανιτίδης, Α. Λαυρέντζος, επιμ. Π. Δήτσας, Χ. Κουρκουμέλη. Ηράκλειο: Πανεπιστημιακές Εκδόσεις Κρήτης, 2017.

[5] A. Saakian, *Radio Wave Propagation Fundamentals*, 2nd ed. Boston, MA, USA: Artech House, 2021.

[6] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[7] A. Bensky, "Radio propagation," in *Short-range Wireless Communication*, 3rd ed. Newnes, 2019, ch. 2, pp. 11–41, doi: 10.1016/B978-0-12-815405-2.00002-6.

[8] R. Desimone, B. M. Brito and J. Baston, "Model of indoor signal propagation using log-normal shadowing," *2015 Long Island Systems, Applications and Technology*, Farmingdale, NY, USA, 2015, pp. 1-4, doi: 10.1109/LISAT.2015.7160217.

[9] S. L. Cotton and W. G. Scanlon, "A Statistical Analysis of Indoor Multipath Fading for a Narrowband Wireless Body Area Network," *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, Helsinki, Finland, 2006, pp. 1-5, doi: 10.1109/PIMRC.2006.254266.

[10] J. S. Seybold, *Introduction to RF Propagation*. New York: Wiley-Interscience, 2005.

[11] S. Ju, O. Kanhere, Y. Xing, and T. S. Rappaport, "A Millimeter-Wave Channel Simulator NYUSIM with Spatial Consistency and Human Blockage," *Heliyon*, vol. 8, no. 12, p. e11761, Dec. 2022, doi: 10.1016/j.heliyon.2022.e11761.

[12] Y. L. C. De Jong, J. A. Pugh, M. Bennai and P. Bouchard, "2.4 to 61 GHz Multiband Double-Directional Propagation Measurements in Indoor Office Environments," in *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 9, pp. 4806-4820, Sept. 2018, doi: 10.1109/TAP.2018.2851279.

[13] S. Sun *et al*., "Investigation of Prediction Accuracy, Sensitivity, and Parameter Stability of Large-Scale Propagation Path Loss Models for 5G Wireless Communications," in *IEEE Transactions on Vehicular Technology*, vol. 65, no. 5, pp. 2843-2860, May 2016, doi: 10.1109/TVT.2016.2543139.

[14] N. Moraitis, A. Rogaris, I. Popescu and K. S. Nikita, "Measurements and Channel Characterization in Indoor Environments in the Upper Mid-Band," in *IEEE Wireless Communications Letters*, vol. 14, no. 6, pp. 1758-1762, June 2025, doi: 10.1109/LWC.2025.3555009.

[15] D. Shakya *et al*., "Comprehensive FR1(C) and FR3 Lower and Upper Mid-Band Propagation and Material Penetration Loss Measurements and Channel Models in Indoor Environment for 5G and 6G," in *IEEE Open Journal of the Communications Society*, vol. 5, pp. 5192-5218, 2024, doi: 10.1109/OJCOMS.2024.3431686.

[16] Oladimeji TT, Kumar P, Elmezughi MK. Path Loss Measurements and Model Analysis in an Indoor Corridor Environment at 28 GHz and 38 GHz. Sensors (Basel). 2022 Oct 9;22(19):7642. doi: 10.3390/s22197642. PMID: 36236740; PMCID: PMC9573193.

[17] D. Parikh, *Raspberry Pi and MQTT Essentials*. Birmingham, UK: Packt Publishing, 2022.

[18] Cameron, N. (2023). ESP32 Microcontroller. In: ESP32 Formats and Communication. Maker Innovations Series. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-9376-8_1

[19] G. C. Hillar, *MQTT Essentials - A Lightweight IoT Protocol*. Birmingham, UK: Packt Publishing, 2017.

[20] M. Copperwaite and C. Leifer, *Learning Flask Framework*. Birmingham, UK: Packt Publishing, 2015.

[21] J. N. Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

[22] EMQX Blog "JavaScript MQTT Client: A Beginner's Guide to MQTT.js" https://www.emqx.com/en/blog/mqtt-js-tutorial (πρόσβαση Ιαν., 2026).

[23] MQTT.js Organization, "node and Javascript MQTT client and parser" https://github.com/mqttjs (πρόσβαση Ιαν., 2026).

[24] C. Baker, "An Arduino Library supporting the MCP3XXX series of ADC SPI chips" https://github.com/bakercp/MCP3XXX (πρόσβαση Ιαν., 2026).

[25] ESPBoards, "ESP32 DOIT DevKit V1 Guide" https://www.espboards.dev/esp32/esp32doit-devkit-v1/ (πρόσβαση Δεκ., 2025).

[26] Wikipedia. "Flask (web framework)" https://en.wikipedia.org/wiki/Flask_(web_framework) (πρόσβαση Δεκ, 2025).

[27] Raspberry Pi Foundation. "Raspberry Pi Documentation" https://www.raspberrypi.com/documentation/ (πρόσβαση Δεκ, 2025).

[28] Espressif Systems, "ESP32-WROOM-32 Datasheet" https://documentation.espressif.com/esp32-wroom-32_datasheet_en.pdf (πρόσβαση Δεκ, 2025).

[29] Aaronia AG, "HyperLOG 70 Series Logarithmic Periodic Antennas Datasheet" https://downloads.aaronia.com/datasheets/antennas/HyperLOG/Aaronia_HyperLOG_70_Logper_Antennas.pdf (πρόσβαση Δεκ., 2025).

[30] Test Equipment Depot, “Series HyperLOG 7060 Datasheet” https://fotronic.asset.akeneo.cloud/pdfs/media/aaronia_hyperlog7000series_datasheet.pdf (πρόσβαση Ιαν., 2026)

[31] Windfreak Technologies, "SynthUSB3 - 12.5MHz to 6.4GHz RF Signal Generator Controller Datasheet" https://windfreaktech.com/wp-content/uploads/2020/10/synthusb3-rf-signal-generator-detector.pdf (πρόσβαση Δεκ., 2025).

[32] Mini-Circuits, "ZX47-60+: Coaxial Power Detector, 10 to 8000 MHz" https://www.minicircuits.com/pdfs/ZX47-60+.pdf (πρόσβαση Δεκ, 2025).

[33] Mini-Circuits, "ZX47-40+: Coaxial Power Detector, 10 to 8000 MHz" https://www.minicircuits.com/pdfs/ZX47-40+.pdf (πρόσβαση Δεκ, 2025).

[34] Mini-Circuits, " AN47-001 - A 10 MHz to 6 GHz Power Meter" Application Note, https://www.minicircuits.com/app/AN47-001.pdf (πρόσβαση Δεκ, 2025).

[35] Microchip Technology Inc., "MCP3002 - 2.7V Dual Channel 10-Bit A/D Converter with SPI Serial Interface" https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21294E.pdf (πρόσβαση Δεκ, 2025).

[36] IntechOpen. (2020). Electromagnetic Spectrum and Its Radio Frequency Bands. Διαθέσιμο σε: https://www.intechopen.com/chapters/66104

[37] Saeling Company, Inc, “SynthUSB3 12.5MHz 6.4GHz RF Signal Generator”. Διαθέσιμο σε: https://www.saelig.com/product/synthusb3

[38] Renzo Mischianti (2021). “DOIT ESP32 DEV KIT v1: high resolution pinout and specs”. Διαθέσιμο σε: https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

[39] Mouser Electronics. “Mini-Circuits ZX47-60LN-S+”. Διαθέσιμο σε: https://eu.mouser.com/ProductDetail/Mini-Circuits/ZX47-60LN-S+

[40] Raspberry Pi Foundation. “Raspberry Pi 4”. Διαθέσιμο σε: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[41] element14 Community (2022). “10 Years of Raspberry Pi - History of Raspberry Pi and element14 Community”. Διαθέσιμο σε: https://community.element14.com/products/raspberry-pi/w/documents/27523/

Παράρτημα Ι

Κώδικας μικροελεγκτών ESP32

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <MCP3XXX.h>

MCP3002 adc;

static const float VREF = 3.33f;     // measured value of 3.3V-pin
static const uint8_t CHANNEL = 0;    // CH0 of mcp is used
int cfg_samples = 25;                // ADC samples to average per
measurement
int cfg_delay_us = 1000;             // delay between ADC samples
volatile unsigned long cfg_publish_period_ms = 50;  // minimum time between
publications

const char* ssid        = "RFDetectorNet";
const char* password    = "pi2651008746";
const char* mqtt_server = "192.168.4.1";  // MQTT broker adress

// Topic namespace per device
String DEVICE_ID    = "esp32_60";  // or "esp32_40"
String STATUS_TOPIC = "home/rfdetector/" + DEVICE_ID + "/status";
String DETECTOR_TOPIC = "home/rfdetector/" + DEVICE_ID + "/voltage";
String CMD_RATE_HZ  = "home/rfdetector/" + DEVICE_ID + "/rate_hz";

const unsigned long STATUS_ms = 5000; // period of re-publishing "online"
status
unsigned long lastStatus = 0;

WiFiClient espClient;
PubSubClient client(espClient);

// Preset definition & mapping: each specific rate (Hz) corresponds to a
profile
struct Preset {
  int rate_hz;
  int samples;
  int delay_us;
};

static const Preset PRESETS[] = {
  {  1, 200, 1000},  // 200 ms burst
  {  5,  50, 1000},  // 50 ms burst
  { 10,  25, 1000},  // 25 ms burst
  { 20,  25,  500},  // 12.5 ms burst
  { 50,  10,  500},  // 5 ms burst
};
// Apply preset by requested rate
bool applyPresetByRate(int rateHz) {
  for (auto &p : PRESETS) {
    if (p.rate_hz == rateHz) {
      cfg_samples = p.samples;
      cfg_delay_us = p.delay_us;
      cfg_publish_period_ms = (unsigned long)(1000L / max(1, rateHz));
      Serial.printf("Preset applied: %d Hz\n", rateHz);
      return true;
```

```
    }
  }
  return false;
}
// Connect to Raspberry Pi Access Point
void connectWifi(){
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
}
// Connect to MQTT broker and initialize LWT
bool connectToMQTT() {
  const char* willTopic = STATUS_TOPIC.c_str(); // where "offline" will be
published
  const char* willMsg   = "offline";  // Publoshed by broker on
disconnection
  int   willQos         = 1;
  bool  willRetain      = true;       // Retain last known status

  if (client.connect(DEVICE_ID.c_str(), willTopic, willQos, willRetain,
willMsg)){
    client.publish(STATUS_TOPIC.c_str(), "online", true);
    client.subscribe(CMD_RATE_HZ.c_str(), 1);
    lastStatus = millis();
    return true;
  } else {
    return false;
  }
}
// Publish "online" status
void publishStatus() {
  if (millis() - lastStatus >= STATUS_ms) {
    client.publish(STATUS_TOPIC.c_str(), "online", true);
    lastStatus = millis();
  }
}
// Read and average the samples from ADC and convert to voltage
float readAverageVoltage(uint8_t ch, int samples, int delay_us) {
  uint32_t acc = 0;
  for (int i = 0; i < samples; ++i) {
    acc += adc.analogRead(ch);
    delayMicroseconds(delay_us);
  }
  float avgCode = acc / float(samples);
  return (avgCode / 1023.0f) * VREF;
}
// Publish detector measurements
void publishDetector() {
  static unsigned long lastPub = 0;
  if (millis() - lastPub < cfg_publish_period_ms) return;
  lastPub = millis();
  float volts = readAverageVoltage(CHANNEL, cfg_samples, cfg_delay_us);
  char payload[16];
  snprintf(payload, sizeof(payload), "%.3f", volts);
  client.publish(DETECTOR_TOPIC.c_str(), payload); // default QoS is 0
  Serial.println(payload);
}
```

```
// Parse the rate received in subscribed topic and apply the matching
preset
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  char buf[16];
  length = min(length, (unsigned int)(sizeof(buf) - 1));
  memcpy(buf, payload, length);
  buf[length] = '\0';
  // Ignore other topics
  if (strcmp(topic, CMD_RATE_HZ.c_str()) != 0) {
    return;
  }
  long rate = strtol(buf, nullptr, 10);// Convert ASCII payload to integer
  if (!applyPresetByRate((int)rate)) {
    return;
  }
}
// ----------------- Arduino setup/loop -----------------
void setup() {
  Serial.begin(115200);
  delay(200);
  adc.begin(5, 23, 19, 18);  // SPI: [CS, MOSI, MISO, SCLK]
  connectWifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(mqttCallback);
  // Connection maintenance
  client.setKeepAlive(3);
  client.setSocketTimeout(5);
  WiFi.setSleep(false);
  // Establishing MQTT connection
  while(!connectToMQTT()){
    delay(3000);
  }
}

void loop() {
  if (!client.connected()) {
    connectToMQTT();
  }
  // Keep MQTT alive and process incoming messages
  client.loop();
  publishDetector();
  publishStatus();
}
```

Παράρτημα ΙΙ

Κώδικας Python – Backend υλοποίηση με Flask

Αρχείο main.py

```python
# app.py - Main Flask application setup and blueprint registration

from flask import Flask, render_template
from server.routes.experiments import bp as experiments_bp
from server.routes.calibration import bp as calibration_bp
from server.routes.devices import bp as devices_bp
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

app = Flask(
    __name__,
    template_folder=os.path.join(BASE_DIR, "templates"),
    static_folder=os.path.join(BASE_DIR, "static")
)
# Register blueprints
app.register_blueprint(experiments_bp)
app.register_blueprint(calibration_bp)
app.register_blueprint(devices_bp)

@app.route('/')
def index():
    return render_template('index.html', title="RF Path Loss Lab")

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=5000, debug=True, use_reloader=False)
```

Αρχείο devices.py

```python
# routes/devices.py - Blueprint for device management endpoints
import paho.mqtt.publish as publish
from flask import Blueprint, jsonify, request
from server.config import MQTT_BROKER, MQTT_PORT
from server.utils import load_sample_rates, save_sample_rates

bp = Blueprint('devices', __name__)

SAMPLE_RATES = {1, 5, 10, 20, 50}

@bp.get('/devices/sample-rates')
def get_sample_rates():
    """Get sample rates for all devices and the allowed rate options."""
    rates = load_sample_rates()
    return jsonify({
        'rates': rates,
        'allowedRates': sorted(SAMPLE_RATES),
    })

@bp.post('/devices/<string:device_id>/sample-rate')
```

```
def set_sample_rate(device_id: str):
    """Set a specific device's sample rate via MQTT and persist it.
    Expects JSON: {"rateHz": <int>}.
    """
    payload = request.get_json(silent=True) or {}
    rate = payload.get('rateHz')
    try:
        rate = int(rate)
    except (TypeError, ValueError):
        return jsonify({'error': 'rateHz must be a number.'}), 400

    if rate not in SAMPLE_RATES:
        return jsonify({'error': f'Invalid rate. Allowed values:
{sorted(SAMPLE_RATES)}'}), 400

    topic = f'home/rfdetector/{device_id}/rate_hz'
    try:
        publish.single(
            topic,
            payload=str(rate),
            hostname=MQTT_BROKER,
            port=MQTT_PORT,
            qos=1,
            retain=True,
        )
    except Exception as exc:
        return jsonify({'error': f'Failed to publish command: {exc}'}), 502

    rates = load_sample_rates()
    rates[device_id] = rate
    if not save_sample_rates(rates):
        return jsonify({'error': 'Failed to persist sample rate.'}), 500

    return jsonify({'ok': True, 'deviceId': device_id, 'rateHz': rate})
```

Αρχείο calibration.py

```
# routes/calibration.py - Blueprint for managing calibration profiles
from flask import Blueprint, request, jsonify
from server.utils import (
    load_calibrations, update_calibration_profile,
delete_calibration_profile
)

bp = Blueprint('calibration', __name__)

@bp.get('/calibration/profiles')
def get_profiles():
    """Return all saved calibration profiles."""
    data = load_calibrations()
    return jsonify({"profiles": data.get("profiles", {})})

@bp.post('/calibration/profiles')
def save_profile():
    """Create or update a calibration profile.
```

```
    Expects JSON: {"freqMHz": <number>, "rss60": <object>, "rss40":
<object>}.
    """
    payload = request.get_json(force=True) or {}
    freq = payload.get("freqMHz")
    rss60 = payload.get("rss60") or {}
    rss40 = payload.get("rss40") or {}
    if freq is None:
        return jsonify({"error": "freqMHz required"}), 400
    prof = update_calibration_profile(freq, rss60, rss40)
    return jsonify({"ok": True, "profile": {str(int(freq)): prof}})

@bp.delete('/calibration/profiles/<int:freq_mhz>')
def remove_profile(freq_mhz: int):
    """Delete the calibration profile for a specific frequency."""
    ok = delete_calibration_profile(freq_mhz)
    if not ok:
        return jsonify({"error": "not found"}), 404
    return jsonify({"ok": True})
```

Αρχείο experiments.py

```
# routes/experiments.py - Blueprint for managing experiment CSV files
# Handles upload, download, listing, and deletion of experiment data.
import os
from datetime import datetime, timezone
from flask import Blueprint, jsonify, request, send_file, abort
from server.config import DATA_DIR

bp = Blueprint('experiments', __name__)

def _csv_path(exp_id: str) -> str:
    """Return the filesystem path for the given experiment's CSV file."""
    from werkzeug.utils import secure_filename
    return os.path.join(DATA_DIR, f"{secure_filename(exp_id)}.csv")

def _list_csv_experiments():
    """Scan DATA_DIR for .csv experiments and return some info for the
modal."""
    items = []
    os.makedirs(DATA_DIR, exist_ok=True)
    for fn in os.listdir(DATA_DIR):
        if not fn.lower().endswith(".csv"):
            continue
        path = os.path.join(DATA_DIR, fn)
        try:
            st = os.stat(path)
        except OSError:
            continue
        exp_id = os.path.splitext(fn)[0]
        items.append({
            "id": exp_id,
            "size_bytes": st.st_size,
            "mtime": datetime.fromtimestamp(st.st_mtime, tz=timezone.utc)
                             .isoformat().replace("+00:00", "Z"),
        })
    items.sort(key=lambda x: x["mtime"], reverse=True)
```

```
    return items

@bp.get('/experiments')
def list_experiments():
    """List all stored experiment CSV files."""
    return jsonify(_list_csv_experiments())

@bp.post('/api/experiments/<exp_id>/upload')
def upload_experiment_csv(exp_id):
    """Receive the client-side CSV at stop and save it as <exp_id>.csv."""
    if 'file' not in request.files:
        return jsonify({"ok": False, "error": "missing file"}), 400
    f = request.files['file']
    if not f.filename.lower().endswith('.csv'):
        return jsonify({"ok": False, "error": "must be .csv"}), 400
    path = _csv_path(exp_id)
    if os.path.exists(path):
        return jsonify({"ok": False, "error": "experiment already
exists"}), 409
    try:
        f.save(path)
    except Exception as e:
        return jsonify({"ok": False, "error": f"save failed: {e}"}), 500
    return jsonify({"ok": True, "path": path})

@bp.post('/api/experiments/import')
def import_experiment_csv():
    """Allow users to upload a CSV from their device into experiment
storage."""
    from werkzeug.utils import secure_filename

    if 'file' not in request.files:
        return jsonify({"ok": False, "error": "missing file"}), 400

    file = request.files['file']
    if not (file and file.filename):
        return jsonify({"ok": False, "error": "missing filename"}), 400

    if not file.filename.lower().endswith('.csv'):
        return jsonify({"ok": False, "error": "must be .csv"}), 400

    # Determine experiment ID
    raw_id = (request.form.get('exp_id') or '').strip()
    if not raw_id:
        raw_id = os.path.splitext(file.filename)[0]

    # Make the experiment ID filesystem-safe
    safe_id = secure_filename(raw_id) or
secure_filename(os.path.splitext(file.filename)[0])
    if not safe_id:
        return jsonify({"ok": False, "error": "invalid experiment id"}),
400

    # Check for overwrite flag
    overwrite = str(request.args.get('overwrite', '')).lower() in ('1',
'true', 'yes', 'on')
    path = _csv_path(safe_id)

    if os.path.exists(path) and not overwrite:
        return jsonify({
            "ok": False,
```

```
            "error": "Experiment already exists on server.",
            "id": safe_id,
            "conflict": True
        }), 409

    # Save the uploaded file
    try:
        os.makedirs(DATA_DIR, exist_ok=True)
        file.save(path)
    except Exception as e:
        return jsonify({"ok": False, "error": f"save failed: {e}"}), 500

    return jsonify({
        "ok": True,
        "id": safe_id,
        "path": path
    })

@bp.get('/api/experiments/<exp_id>/download')
def download_experiment_csv(exp_id):
    """Download an experiment CSV by experiment id."""
    path = _csv_path(exp_id)
    if not os.path.exists(path):
        abort(404)
    return send_file(path, as_attachment=True,
download_name=f"{exp_id}.csv")

@bp.get('/experiment-data')
def experiment_data():
    """
    CSV-based endpoint to download experiment data.
    CSV schema:
      timestamp,device,volt,dbm,freq_mhz,a,b,distance_m,path_loss_db
    Comment lines (starting with '#') are skipped.
    """
    exp_id = request.args.get('exp_id')
    filename = request.args.get("filename", exp_id or "experiment")
    if not exp_id: return "Missing exp_id", 400

    path = _csv_path(exp_id)
    if not os.path.exists(path):
        return "Experiment CSV not found on server", 404

    return send_file(path, as_attachment=True,
download_name=f"{filename}.csv")

@bp.delete('/experiment/<exp_id>')
def delete_experiment(exp_id):
    """Delete an experiment CSV by experiment id."""
    if exp_id.endswith('.csv'):
        exp_id = exp_id[:-4]
    try:
        os.remove(_csv_path(exp_id))
        return jsonify({"ok": True, "deleted": exp_id})
    except FileNotFoundError:
        return jsonify({"error": f"Experiment {exp_id} not found."}), 404
    except Exception as exc:
        return jsonify({"error": f"Failed to delete experiment: {exc}"}),
500
```

Αρχείο utils.py

```
# utils.py - Utility functions for calibration and sample rate management
import json, math, os
from server.config import CALIBRATIONS_FILE, SAMPLE_RATES_FILE

CAL_DEFAULT = {"profiles": {}}

def load_json_file(path):
    """Load and parse JSON, returning 'None' if it can't be read."""
    try:
        with open(path, "r") as f:
            return json.load(f)
    except Exception:
        return None

def save_json_file(path, data):
    """Write 'data' as JSON, creating the parent directory if needed."""
    try:
        directory = os.path.dirname(path)
        if directory:
            os.makedirs(directory, exist_ok=True)
        with open(path, "w") as f:
            json.dump(data, f, indent=2)
        return True
    except Exception as e:
        print(f"Failed to save {path}: {e}")
        return False

def load_calibrations() -> dict:
    """Load calibration profiles from 'CALIBRATIONS_FILE' (creating
defaults if missing)."""
    data = load_json_file(CALIBRATIONS_FILE)
    if not data:
        os.makedirs(os.path.dirname(CALIBRATIONS_FILE), exist_ok=True)
        save_json_file(CALIBRATIONS_FILE, CAL_DEFAULT)
        return json.loads(json.dumps(CAL_DEFAULT))
    # Ensure keys exist
    data.setdefault("profiles", {})
    return data

def save_calibrations(data: dict) -> bool:
    """Save calibration profiles to 'CALIBRATIONS_FILE'."""
    data.setdefault("profiles", {})
    return save_json_file(CALIBRATIONS_FILE, data)

def update_calibration_profile(freq_mhz: int, rss60: dict, rss40: dict) ->
dict:
    """Create or update the calibration profile for 'freq_mhz' and persist
it."""
    data = load_calibrations()
    key = str(int(freq_mhz))
    prof = data["profiles"].setdefault(key, {})
    if "a" in rss60 and "b" in rss60:
        prof["rss60"] = {"a": float(rss60["a"]), "b": float(rss60["b"])}
    if "a" in rss40 and "b" in rss40:
        prof["rss40"] = {"a": float(rss40["a"]), "b": float(rss40["b"])}
    save_calibrations(data)
    return prof

def delete_calibration_profile(freq_mhz: int) -> bool:
```

```
    """Delete the calibration profile for 'freq_mhz'."""
    data = load_calibrations()
    key = str(int(freq_mhz))
    existed = data["profiles"].pop(key, None)
    save_calibrations(data)
    return existed is not None

def load_sample_rates() -> dict:
    """Load per-device sample rates from 'SAMPLE_RATES_FILE'."""
    data = load_json_file(SAMPLE_RATES_FILE)
    return data if isinstance(data, dict) else {}

def save_sample_rates(rates: dict) -> bool:
    """Save per-device sample rates to 'SAMPLE_RATES_FILE'."""
    payload = {}
    # ensure all rates are integers
    for dev, rate in (rates or {}).items():
        try:
            payload[str(dev)] = int(rate)
        except (TypeError, ValueError):
            continue
    return save_json_file(SAMPLE_RATES_FILE, payload)
```

Αρχείο config.py

```
# server/config.py - Configuration values
import os

DATA_DIR = os.getenv("DATA_DIR", "/home/pi4/rfWebApp/data")
CALIBRATIONS_FILE = os.path.join(DATA_DIR, "calibrations.json")
MQTT_BROKER = os.getenv("MQTT_BROKER", "XXX.XXX.X.X")
MQTT_PORT = int(os.getenv("MQTT_PORT", "1883"))
SAMPLE_RATES_FILE = os.path.join(DATA_DIR, "sample_rates.json")
```

Παράρτημα ΙΙΙ

Κώδικας HTML – Η δομή της διεπαφής χρήστη της εφαρμογής

Αρχείο index.html

```
<!DOCTYPE html>
{% from "_macros.html" import chart_block %}
<html>

<head>
  <meta charset="utf-8">
  <title>RF Path Loss Lab</title>
  <script src="{{ url_for('static', filename='lib/plotly-2.27.0.min.js')
}}"></script>
  <script src="{{ url_for('static', filename='lib/mqtt.min.js')
}}"></script>
  <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
</head>

<body>
  <header class="brand-header">
    <img class="heartbeat-svg" src="{{ url_for('static',
filename='img/heartbeat.svg') }}" alt="">
    <div class="fade-in"></div>
    <div class="fade-out"></div>
    <h1 class="brand-text">RF Path Loss Lab</h1>
  </header>

  <div class="mode-bar">
    <div class="mode-bar-top">
      <button id="configBtn" class="btn btn--panel">
        <svg width="20" height="20" viewBox="0 0 24 24" aria-hidden="true"
fill="none" stroke="currentColor"
          stroke-width="1.8" stroke-linecap="round" stroke-
linejoin="round">
          <use href="{{ url_for('static', filename='img/icons.svg')  }}
#icon-config"></use>
        </svg>
      </button>
      <div id="modeSwitch" class="mode-select">
        <button class="mode-item active" data-mode="live">Live</button>
        <button class="mode-item" data-
mode="experiment">Experiment</button>
      </div>
      <button id="togglePanelBtn" class="btn btn--panel">
        <svg width="16" height="16" viewBox="0 0 24 24">
          <use href="{{ url_for('static', filename='img/icons.svg')
}}#icon-chevron-up" fill="none"
            stroke="currentColor" stroke-width="4" stroke-linecap="round"
stroke-linejoin="round">
          </use>
        </svg>
      </button>
    </div>
```

```
    <div id="livePanel" class="mode-panel hidden">
      <label class="mc-field">Live Window:
        <select id="timeRange" class="form-control form-control--select">
          <option value="1">1 Minute</option>
          <option value="3">3 Minutes</option>
          <option value="5">5 Minutes</option>
        </select>
      </label>
      <label id="combinePlotsField" class="mc-field">
        <span>Combine plots</span>
        <input id="combineToggle" type="checkbox">
      </label>
    </div>

    <div id="experimentPanel" class="mode-panel hidden">
      <div class="expPanel-top">
        <div class="left">
          <input id="expId" class="form-control form-control--input width-
20ch" type="text"
            placeholder="Experiment name">
        </div>
        <div class="center">
          <span class="exp-status-text">No active experiment</span>
          <span id="expTimer" class="exp-timer"></span>
        </div>
        <div class="right">
          <button id="browseExperimentsBtn" class="btn btn--panel"
type="button">Browse</button>
          <a id="downloadExpCsv" class="btn btn--panel hidden"
download>Export</a>
        </div>
      </div>

      <div class="expPanel-bottom">
        <div class="group">
          <span class="lbl">Tx-Rx Distance (m)</span>
          <input id="distInput_rss60" class="form-control form-control--
input width-5ch" type="number" min="0"
            placeholder="Ch1">
          <input id="distInput_rss40" class="form-control form-control--
input width-5ch" type="number" min="0"
            placeholder="Ch2">
        </div>
        <div class="group">
          <span class="lbl">Pt (dBm)</span>
          <input id="txPowerDbm" class="form-control form-control--input
width-5ch" type="number">
        </div>
        <div class="group">
          <span class="lbl">Gt (dBi)</span>
          <input id="txGainDbi" class="form-control form-control--input
width-5ch" type="number">
        </div>
        <div class="group">
          <span class="lbl">Gr (dBi)</span>
          <input id="rxGain_rss60" class="form-control form-control--input
width-5ch" type="number" placeholder="Ch1">
          <input id="rxGain_rss40" class="form-control form-control--input
width-5ch" type="number" placeholder="Ch2">
        </div>
        <div class="group">
```

```
          <span class="lbl">System loss (dB)</span>
          <input id="sysLoss_rss60" class="form-control form-control--input
width-5ch" type="number" placeholder="Ch1">
          <input id="sysLoss_rss40" class="form-control form-control--input
width-5ch" type="number" placeholder="Ch2">
        </div>
      </div>
    </div>
  </div>

  <div class="exp-action-panel">
    <button id="startExpBtn" class="btn btn--panel">Start</button>
    <button id="pauseExpBtn" class="btn btn--panel" disabled>Pause</button>
    <button id="stopExpBtn" class="btn btn--panel" disabled>Stop</button>
  </div>

  <div id="channelToggleGroup" class="live-readings">
    <div class="reading">
      <span class="freq-display">Frequency: -</span>
    </div>
    <div class="reading" data-channel="rss60">
      <label class="channel-toggle">
        <input type="checkbox" data-channel-toggle="rss60" checked>
        <span>Channel 1</span>
      </label>
      <div class="reading-data">
        <span>Last update:</span>
        <span id="rss60Timestamp">--</span>
        <span>RSS:</span>
        <span id="rss60" class="reading-value">--</span>
      </div>
    </div>
    <div class="reading" data-channel="rss40">
      <label class="channel-toggle">
        <input type="checkbox" data-channel-toggle="rss40" checked>
        <span>Channel 2</span>
      </label>
      <div class="reading-data">
        <span>Last update:</span>
        <span id="rss40Timestamp">--</span>
        <span>RSS:</span>
        <span id="rss40" class="reading-value">--</span>
      </div>
    </div>
  </div>

  {{ chart_block('rss60', 'Channel 1: ZX47-60LN-S+', 'statusBar1') }}
  {{ chart_block('rss40', 'Channel 2: ZX47-40LN-S+', 'statusBar2') }}

  <div id="combinedBlock" class="chart-table-block hidden">
    <div class="block-header">
      <h2 class="chart-title">Channel: ZX47-60LN-S+ & ZX47-40LN-S+</h2>
    </div>
    <div class="block-content">
      <div id="combinedPlot" class="plot-box"></div>
    </div>
  </div>
  <hr>

  {% include "modals/exp_modal.html" %}
  {% include "modals/cfg_modal.html" %}
```

```
  {% include "modals/analysis_modal.html" %}

  <script>
    window.MQTT_URL = 'ws://' + location.host + '/mqtt';
    window.MQTT_TOPIC = 'home/rfdetector/+/voltage';
    window.MQTT_DEV_A = 'esp32_60';
    window.MQTT_DEV_B = 'esp32_40';
  </script>
  <script type="module" src="{{ url_for('static', filename='js/main.js')
}}"></script>

</body>
</html>
```

Αρχείο _macros.html – Χρήση Jinja2 macro template για τη δημιουργία των δύο πανομοιότυπων διαγραμμάτων & πινάκων

```
{% macro chart_block(key, title, status_bar_id) -%}
<div id="{{ key }}Block" class="chart-table-block">
  <div class="block-header">
    <h2 class="chart-title">{{ title }}</h2>
    <div id="{{ status_bar_id }}" class="status-bar"></div>
    <div class="header-actions">
      <button class="btn btn--panel analyze-btn hidden" data-channel="{{
key }}">Path Loss Analysis</button>
      <button class="btn btn--panel table-toggle" data-target="{{ key
}}">Table</button>
    </div>
  </div>
  <div class="block-content">
    <div id="{{ key }}Plot" class="plot-box"></div>
    <div id="{{ key }}TablePanel" class="table-box">
      <table class="table">
        <thead>
          <tr>
            <th>Time</th>
            <th><span id="{{ key }}TableUnit">Voltage (V)</span></th>
          </tr>
        </thead>
        <tbody id="{{ key }}TableBody"></tbody>
      </table>
    </div>
  </div>
</div>
{%- endmacro %}
```

Αρχείο exp_modal.html – Παράθυρο διαχείρισης αποθηκευμένων πειραμάτων

```
<div id="expModal" class="modal hidden">
  <div class="modal-backdrop"></div>
  <div class="modal-box">
    <div class="modal-header-top">
      <h3>Experiments</h3>
      <button id="expModalClose" class="modal-close">&times;</button>
    </div>
    <div class="modal-body">
      <button id="importCsvBtn" class="btn btn--modal">Import CSV</button>
      <input id="importCsvInput" type="file" accept=".csv" class="hidden">
      <div id="expModalEmpty" class="muted">No experiments found.</div>
      <table id="expTable" class="table hidden">
        <thead>
          <tr>
            <th>Name</th>
            <th>Modified</th>
            <th>Size</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody id="expTableBody"></tbody>
      </table>
    </div>
  </div>
</div>
```

Αρχείο cfg_modal.html – Παράθυρο διαμόρφωσης παραμέτρων της εφαρμογής

```
<div id="cfgModal" class="modal hidden">
  <div class="modal-backdrop"></div>
  <div class="modal-box">
    <div class="modal-header-top">
      <h3>App Configurations</h3>
      <button id="cfgCloseBtn" class="modal-close">&times;</button>
    </div>
    <div class="modal-body">
      <section class="cfg-section">
        <div class="modal-header">
          <h4>Data Units</h4>
        </div>
        <label>
          View:
          <select id="cfgUnits" class="form-control form-control--select">
            <option value="volt">Voltage (V)</option>
            <option value="dbm">Input Power (dBm)</option>
          </select>
        </label>
        <p class="muted">
          Before switching to dBm make sure you have selected the correct
calibration profile.
        </p>
      </section>
      <div class="cfg-divider"></div>
      <section class="cfg-section">
        <div class="modal-header">
          <h4>Sampling Rate</h4>
        </div>
```

```
        <label>
          Channel 1:
          <select id="cfgRate60" class="form-control form-control--select">
            <option value="1">1 Hz</option>
            <option value="5">5 Hz</option>
            <option value="10">10 Hz</option>
            <option value="20">20 Hz</option>
            <option value="50">50 Hz</option>
          </select>
          Channel 2:
          <select id="cfgRate40" class="form-control form-control--select">
            <option value="1">1 Hz</option>
            <option value="5">5 Hz</option>
            <option value="10">10 Hz</option>
            <option value="20">20 Hz</option>
            <option value="50">50 Hz</option>
          </select>
        </label>
      </section>
      <div class="cfg-divider"></div>
      <section class="cfg-section">
        <div class="modal-header">
          <h4>Power Input Calibration</h4>
        </div>
        <p class="muted">
          Enter a frequency and the corresponding linear coefficients for
each detector, to compute the Input Power:
          P(dBm) = a * V + b.
        </p>
        <label>
          Frequency (MHz):
          <input id="cfgFreq" class="form-control form-control--input
width-13ch" type="number" min="10" max="8000"
            placeholder="e.g. 2400">
        </label>
        <div>
          <label>
            Channel 1: slope
            <input id="cfg60_a" class="form-control form-control--input
width-13ch" type="number" step="0.01">
          </label>
          <label>
            intercept
            <input id="cfg60_b" class="form-control form-control--input
width-13ch" type="number" step="0.01">
          </label>
        </div>
        <div>
          <label>
            Channel 2: slope
            <input id="cfg40_a" class="form-control form-control--input
width-13ch" type="number" step="0.0001">
          </label>
          <label>
            intercept
            <input id="cfg40_b" class="form-control form-control--input
width-13ch" type="number" step="0.01">
          </label>
        </div>
        <button id="cfgSaveBtn" class="btn btn--modal">Save</button>
      </section>
```

```
      <div class="cfg-divider"></div>
      <section class="cfg-section">
        <div class="modal-header">
          <h4>Saved Calibration Profiles</h4>
        </div>
        <div class="table-wrap">
          <table class="table">
            <thead>
              <tr>
                <th>Frequency (MHz)</th>
                <th>Channel 1 (a,b)</th>
                <th>Channel 2 (a,b)</th>
                <th class="col-actions">Actions</th>
              </tr>
            </thead>
            <tbody id="cfgProfilesTbody">
            </tbody>
          </table>
        </div>
        <p class="muted">
          Click "set" to activate profile. Click "Load" to edit the
coefficients.
        </p>
      </section>
    </div>
  </div>
</div>
```

Αρχείο analysis_modal.html – Παράθυρο μελέτης της εξασθένησης διαδρομής

```
<div id="analysisModal" class="modal hidden">
  <div class="modal-backdrop"></div>
  <div class="modal-box">
    <div class="modal-header-top">
      <h3 id="analysisModalTitle">Path-loss analysis for experiment: -</h3>
      <button id="analysisCloseBtn" class="modal-close">&times;</button>
    </div>
    <div class="modal-body">
      <div>
        <label>View data:
          <select id="plView" class="form-control form-control--select">
            <option value="raw" selected>Raw</option>
            <option value="binned">Binned</option>
          </select>
        </label>
        <label>Binning:
          <select id="plStat" class="form-control form-control--select">
            <option value="mean" selected>Mean</option>
            <option value="median">Median</option>
          </select>
        </label>
        <label>X-Scale:
          <select id="plXscale" class="form-control form-control--select">
            <option value="linear" selected>Linear</option>
            <option value="log">Log</option>
          </select>
        </label>
```

```
        <label>Title:
          <input id="plTitleInput" class="form-control form-control--input
width-20ch" type="text"
            placeholder="Path Loss vs. Distance">
        </label>
        <label>
          <span>Light mode</span>
          <input id="analysisLightThemeToggle" type="checkbox">
        </label>
        <button id="analysisComputeBtn" class="btn btn--
modal">Compute</button>
      </div>

      <div id="plChart_rss60" class="plot-box"></div>
      <div id="plChart_rss40" class="plot-box hidden"></div>

      <div class="analysis-toggles">
        <button id="fsplToggle" class="btn btn--modal">FSPL curve</button>
        <button id="bestCurveToggle" class="btn btn--modal">OLS
Fit</button>
        <div class="logDistance">
          <button id="logDistanceToggle" class="btn btn--modal">Log-
distance model</button>
          <div id="logDistancePanel" class="analysis-logdistance-panel
hidden">
            <label class="mc-field">
              <span class="analysis-logdistance-label">d<sub>0</sub>
(m)</span>
              <input id="logDistanceD0" class="form-control form-control--
input width-9ch" type="number" min="0" step="0.01">
            </label>
            <label class="mc-field">
              <span class="analysis-logdistance-label">PL(d<sub>0</sub>)
(dB)</span>
              <input id="logDistancePl0" class="form-control form-control--
input width-9ch" type="number" step="0.01">
            </label>
            <label class="mc-field">
              <input id="logDistanceUseFirst" type="checkbox">
              <span>Use data's first point</span>
            </label>
            <label class="mc-field">
              <input id="logDistanceUseFsplRef" type="checkbox">
              <span>Use FSPL 1m reference</span>
            </label>
            <label class="mc-field">
              <span class="analysis-logdistance-label">n exponent</span>
              <input id="logDistanceExponent" class="form-control form-
control--input width-9ch" type="number" step="0.01">
            </label>
            <label class="mc-field">
              <input id="logDistanceFitExponent" type="checkbox">
              <span>Fit n exponent</span>
            </label>
            <label class="mc-field">
              <span class="analysis-logdistance-label">Shadowing &sigma;
(dB)</span>
              <input id="logDistanceSigma" class="form-control form-
control--input width-9ch" type="number" min="0" step="0.1"
                placeholder="0">
            </label>
```

```
            <button id="logDistanceFitBtn" class="btn btn--
modal">Fit</button>
            <button id="logDistanceClearBtn" class="btn btn--
modal">Clear</button>
          </div>
        </div>
      </div>
      <div class="analysis-results">
        <div id="fsplSummary" class="analysis-result-box analysis-result-
box--fspl hidden">
        </div>
        <div id="olsSummary" class="analysis-result-box analysis-result-
box--ols hidden">
        </div>
        <div id="logDistanceSummary" class="analysis-result-box analysis-
result-box--log hidden">
        </div>
      </div>
    </div>
  </div>
</div>
```

Παράρτημα IV

Κώδικας Javascript – Client side υλοποίηση της λειτουργικότητας της εφαρμογής

Αρχείο main.js

```
// main.js - entry point for the web app. Handles mode switching, config
modal, and wiring up UI interactions.

import { ModeManager } from './core/modeManager.js';
import { Analysis } from './core/analysis.js';
import * as Live from './modes/live.js';
import * as Experiment from './modes/experiment.js';
import { renderTable } from './core/plot.js';
import { Calibration, SERIES } from './core/state.js';
import { calDeleteProfile, calSaveProfile, getSampleRates, setSampleRate }
from './core/api.js';
import { CHANNEL_KEYS, getChannelState, setChannelEnabled } from
'./core/channels.js';

ModeManager.register('live', Live);
ModeManager.register('experiment', Experiment);
let panelExpanded = false;

// Show/hide the main panel for the current mode.
function showPanelFor(mode, expand = panelExpanded){
  const panels = {
    live: document.getElementById('livePanel'),
    experiment: document.getElementById('experimentPanel'),
  };
  panelExpanded = !!expand;
  Object.values(panels).forEach(p => p && p.classList.add('hidden'));
  const p = panels[mode];
  if (!p) return;
  p.classList.toggle('hidden', !panelExpanded);
  const btn = document.getElementById('togglePanelBtn');
  if (btn) btn.setAttribute('aria-expanded', panelExpanded ? 'true' :
'false');
}
// Wire up the mode selector UI to switch between live/experiment modes
function attachModeSwitch(){
  const switchEl = document.getElementById('modeSwitch');
  if (!switchEl) return;
  switchEl.addEventListener('click', async (e)=>{
    const btn = e.target.closest('.mode-item');
    if (!btn) return;
    const mode = btn.dataset.mode;
    if (!ModeManager.modes[mode]) return;

    switchEl.querySelectorAll('.mode-item').forEach(b =>
b.classList.toggle('active', b.dataset.mode === mode));
    document.body.classList.toggle('is-live', mode==='live');
    document.body.classList.toggle('is-experiment', mode==='experiment');

    await ModeManager.switchTo(mode);
    showPanelFor(mode, panelExpanded);
  });
}
// Wire the panel toggle button to show/hide the expandable panel
```

```
function wirePanelToggle(){
  const btn = document.getElementById('togglePanelBtn');
  if (!btn) return;
  btn.addEventListener('click', ()=>{
    const mode = ModeManager.current || 'live';
    const panel = ({
      live: document.getElementById('livePanel'),
      experiment: document.getElementById('experimentPanel'),
    })[mode];
    if (!panel) return;
    panelExpanded = panel.classList.contains('hidden');
    showPanelFor(mode, panelExpanded);
  });
}
// Syncs the channel UI with the current enabled/disabled channel state.
function updateChannelReadouts(){
  const state = getChannelState();
  CHANNEL_KEYS.forEach((key) => {
    const input = document.querySelector(`input[data-channel-
toggle="${key}"]`);
    if (input) input.checked = !!state[key];
    const reading = document.querySelector(`.reading[data-
channel="${key}"]`);
    if (reading) reading.classList.toggle('is-disabled', !state[key]);
    const def = SERIES[key];
    if (def && !state[key]) {
      const valueEl = document.getElementById(def.liveEl);
      if (valueEl) valueEl.textContent = '--';
      const tsEl = document.getElementById(def.liveTsEl);
      if (tsEl) tsEl.textContent = '--';
    }
    document.body.classList.toggle(`channel-${key}-enabled`, !!state[key]);
    document.body.classList.toggle(`channel-${key}-disabled`, !state[key]);
  });
}
// Schedule a resize of visible charts on the next animation frame when the
window resizes
let resize = false;
function queueVisiblePlotResize() {
  if (resize) return;
  if (!window.Plotly || !window.Plotly.Plots?.resize) return;
  resize = true;
  requestAnimationFrame(() => {
    resize = false;
    const plotNodes = document.querySelectorAll('.chart-table-
block:not(.hidden) .plot-box');
    plotNodes.forEach((node) => {
      try {
        window.Plotly.Plots.resize(node);
      } catch {}
    });
  });
}
// Wire up the channel toggle checkboxes to show/hide channels and update
readouts accordingly
let channelUiattached = false;
function wireChannelToggles(){
  if (channelUiattached) return;
  const group = document.getElementById('channelToggleGroup');
  if (!group) return;
  group.addEventListener('change', (e) => {
```

```
    const input = e.target.closest('input[data-channel-toggle]');
    if (!input) return;
    const channel = input.dataset.channelToggle;
    const want = input.checked;
    const ok = setChannelEnabled(channel, want);
    if (!ok) {
      input.checked = !want;
      alert(`At least one channel must remain enabled.`);
    }
  });
  window.addEventListener('channel-visibility-changed', (ev) => {
    updateChannelReadouts();
    if (typeof Experiment.onChannelVisibilityChange === 'function') {
      try {
        Experiment.onChannelVisibilityChange(ev?.detail || null);
      } catch {}
    }
    queueVisiblePlotResize();
  });
  updateChannelReadouts();
  queueVisiblePlotResize();
  channelUiattached = true;
}
// Set up the behavior of the configurations modal
function wireConfigModal(){
  const openBtn = document.getElementById('configBtn');
  const modal   = document.getElementById('cfgModal');
  const closeBtn= document.getElementById('cfgCloseBtn');
  const saveBtn = document.getElementById('cfgSaveBtn');
  const table   = document.getElementById('cfgProfilesTbody');
  const freqEl = document.getElementById('cfgFreq');
  const a60El = document.getElementById('cfg60_a');
  const b60El = document.getElementById('cfg60_b');
  const a40El = document.getElementById('cfg40_a');
  const b40El = document.getElementById('cfg40_b');
  const sampleRateSelectors = [
    { el: document.getElementById('cfgRate60'), deviceId: window.MQTT_DEV_A
|| 'esp32_60' },
    { el: document.getElementById('cfgRate40'), deviceId: window.MQTT_DEV_B
|| 'esp32_40' },
  ];
  // Refresh the sample rate selectors with current values from the server
  async function refreshSampleRateSelectors() {
    if (!sampleRateSelectors.some(({ el }) => el)) return;
    try {
      const resp = await getSampleRates();
      const rates = resp?.rates || {};
      sampleRateSelectors.forEach(({ el, deviceId }) => {
        if (!el) return;
        const current = Number(rates?.[deviceId]);
        if (Number.isFinite(current)) {
          el.value = String(current);
          el.dataset.prev = String(current);
        } else {
          el.value = '';
          el.dataset.prev = '';
        }
      });
    } catch {}
  }
  // Update sample rate on server when changed
```

```
  sampleRateSelectors.forEach(({ el, deviceId }) => {
    if (!el || !deviceId) return;
    el.addEventListener('change', async () => {
      const next = Number(el.value);
      if (el.dataset.prev === String(next)) return;
      el.disabled = true;
      try {
        await setSampleRate(deviceId, next);
        el.dataset.prev = String(next);
      } catch {
        alert('Failed to update sample rate. Please try again.');
        if (el.dataset.prev != null) {
          el.value = el.dataset.prev;
        }
      } finally {
        el.disabled = false;
      }
    });
  });
  // Open the modal and reset the input fields
  function open(){
    modal?.classList.remove('hidden');
    modal?.setAttribute('aria-hidden','false');

    if (freqEl) freqEl.value = '';
    if (a60El)  a60El.value  = '';
    if (b60El)  b60El.value  = '';
    if (a40El)  a40El.value  = '';
    if (b40El)  b40El.value  = '';

    refreshSampleRateSelectors();
    renderProfilesTable();
  }
  function close(){ modal?.classList.add('hidden');
modal?.setAttribute('aria-hidden','true'); }

  openBtn?.addEventListener('click', open);
  closeBtn?.addEventListener('click', close);
  modal?.querySelector('.modal-backdrop')?.addEventListener('click',
close);
  // Handle clicks on action button in the profiles table
  table?.addEventListener('click', async (e)=>{
    const btn = e.target.closest('button[data-action]');
    if (!btn) return;
    const tr = btn.closest('tr[data-freq]');
    if (!tr) return;
    const freq = Number(tr.dataset.freq);

    switch (btn.dataset.action) {
      case 'load': {
        freqEl.value = freq;
        const p = Calibration.profiles[String(freq)] || {};
        const c60 = p.rss60 || {}, c40 = p.rss40 || {};
        a60El.value = (Number.isFinite(c60.a) ? c60.a : '');
        b60El.value = (Number.isFinite(c60.b) ? c60.b : '');
        a40El.value = (Number.isFinite(c40.a) ? c40.a : '');
        b40El.value = (Number.isFinite(c40.b) ? c40.b : '');
        break;
      }
      case 'set': {
        // Only allow if both channels complete
```

```
        const p = Calibration.profiles[String(freq)] || {};
        const ok = Number.isFinite(p?.rss60?.a) &&
Number.isFinite(p?.rss60?.b)
                && Number.isFinite(p?.rss40?.a) &&
Number.isFinite(p?.rss40?.b);
        if (!ok) { alert('Profile is incomplete (both channels need
parameters).'); return; }
        await Calibration.setActiveFreq(freq);
        renderProfilesTable();
        updateFreqDisplay();
        break;
      }
      case 'delete': {
        if (String(Calibration.activeFreqMHz||'') === String(freq)) {
          alert('Cannot delete the active profile. Set another active
first.');
          return;
        }
        if (!confirm(`Delete profile ${freq} MHz?`)) return;
        try {
          await calDeleteProfile(freq);
          delete Calibration.profiles[String(freq)];
          renderProfilesTable();
        } catch (err) {
          alert('Failed to delete profile.');
        }
        break;
      }
    }
  });
  // Handle the save button to add/update a profile based on input fields
  saveBtn?.addEventListener('click', async ()=>{
    const f = parseFloat(freqEl?.value);
    const a60 = parseFloat(a60El?.value), b60 = parseFloat(b60El?.value);
    const a40 = parseFloat(a40El?.value), b40 = parseFloat(b40El?.value);
    if (!Number.isFinite(f) || f <= 0) { alert('Enter a valid frequency
(MHz).'); return; }
    if (![a60,b60,a40,b40].every(Number.isFinite)) { alert('Fill all four
coefficients.'); return; }

    try {
      await calSaveProfile({ freqMHz: f, rss60: {a:a60,b:b60}, rss40:
{a:a40,b:b40} });
      Calibration.updateLocalProfile(f, { rss60:{a:a60,b:b60},
rss40:{a:a40,b:b40} });
      renderProfilesTable();
      if (freqEl) freqEl.value = ''; if (a60El)  a60El.value  = ''; if
(b60El)  b60El.value  = '';
      if (a40El)  a40El.value  = ''; if (b40El)  b40El.value  = '';
    } catch (e) {
      alert('Failed to save calibration.');
    }
  });
}
// Handle the calibration profiles table in the configuration modal
function renderProfilesTable() {
  const tb = document.getElementById('cfgProfilesTbody');
  const active = Calibration.activeFreqMHz != null ?
String(Calibration.activeFreqMHz) : null;
  if (!tb) return;
```

```
  const entries = Object.entries(Calibration.profiles)
    .sort((a,b)=> Number(a[0]) - Number(b[0])); // ascending MHz

  const fmt = (a,b) => (Number.isFinite(a) && Number.isFinite(b))
    ? `(${a}, ${b})` : `<span class="muted">-</span>`;

  const rowsHtml = entries.map(([f, p]) => {
    const a60 = p?.rss60?.a, b60 = p?.rss60?.b;
    const a40 = p?.rss40?.a, b40 = p?.rss40?.b;
    const ok60 = Number.isFinite(a60) && Number.isFinite(b60);
    const ok40 = Number.isFinite(a40) && Number.isFinite(b40);
    const complete = ok60 && ok40;
    const isActive = active === f;

    return `<tr data-freq="${f}" class="${isActive ? 'is-active' : ''}">
      <td><strong>${f}</strong> MHz ${isActive ? '<span class="badge"
style="margin-left:6px; color:#0048FF; font-weight:700;">Active</span>' :
''}</td>
      <td>${fmt(a60,b60)}</td>
      <td>${fmt(a40,b40)}</td>
      <td>
        <div class="btn-row">
          <button class="btn btn--modal action-btn use-exp" data-
action="load">Load</button>
          <button class="btn btn--modal action-btn use-exp" data-
action="set" ${complete ? '' : 'disabled'}>Set</button>
          <button class="btn btn--modal action-btn danger del-exp" data-
action="delete" ${isActive ? 'disabled' : ''}>Delete</button>
        </div>
      </td>
    </tr>`;
  }).join('');
  tb.innerHTML = rowsHtml || `<tr><td colspan="4" class="muted">No profiles
saved yet.</td></tr>`;
}
// Update the side-table header to the active unit
function updateUnitsLabels(){
  const isDbm = !!Calibration.enabled;
  const unitLabel = isDbm ? 'Input Power (dBm)' : 'Voltage (V)';
  const t60 = document.getElementById('rss60TableUnit');
  const t40 = document.getElementById('rss40TableUnit');
  if (t60) t60.textContent = unitLabel;
  if (t40) t40.textContent = unitLabel;
}
// Wire up the units dropdown menu to switch between voltage and dBm
display modes.
function wireUnitsToggle(){
  const sel = document.getElementById('cfgUnits');
  sel.value = 'volt';

  sel.addEventListener('change', async () => {
    const wantDbm = sel.value === 'dbm';
    await Calibration.setEnabled(wantDbm);
    updateUnitsLabels();
  });
}
// Sync the frequency display in the UI.
function updateFreqDisplay() {
  const freq = Number.isFinite(+Calibration.activeFreqMHz) ?
Math.round(+Calibration.activeFreqMHz) : null;
```

```
  const hasProfile = freq != null &&
!!Calibration.profiles?.[String(freq)];
  const text = hasProfile ? `Frequency: ${freq} MHz` : 'Frequency: -';
  document.querySelectorAll('.freq-display').forEach((el) => {
el.textContent = text; });
}
// Update the units toggle to match selected units
function updateUnitsToggle() {
  const sel = document.getElementById('cfgUnits');
  if (sel) sel.value = Calibration.enabled ? 'dbm' : 'volt';
}
// Wire up the table toggle buttons to show/hide tables and resize plots.
function wireTableToggles() {
  document.addEventListener('click', (e)=>{
    const btn = e.target.closest('.table-toggle');
    if (!btn) return;
    const key = btn.dataset.target;
    const block = document.getElementById(key + 'Block');
    if (!block) return;
    block.classList.toggle('show-table');
    btn.classList.toggle('active');
    const plotId = ({rss60:'rss60Plot', rss40:'rss40Plot'})[key];
    const el = document.getElementById(plotId);
    if (el && window.Plotly) setTimeout(()=> Plotly.Plots.resize(el), 0);
    if (block.classList.contains('show-table')) { renderTable(key); }
  });
}

window.addEventListener('resize', queueVisiblePlotResize);
// After units are changed, update labels and refresh tables to show new
units.
window.addEventListener('units-changed', async (e)=>{
  updateUnitsToggle();
  updateUnitsLabels();
});
// After calibration profile is updated, refresh the frequency display.
window.addEventListener('calibration-updated', ()=>{
  updateFreqDisplay();
});
// On page load, initialize the app, set up event handlers, and switch to
the default mode.
document.addEventListener('DOMContentLoaded', async ()=>{
  Analysis.init();
  attachModeSwitch();
  wirePanelToggle();
  wireChannelToggles();
  wireConfigModal();
  wireUnitsToggle();
  wireTableToggles();
  await Calibration.initFromServer();
  updateUnitsToggle();
  updateUnitsLabels();
  updateFreqDisplay();
  // default mode: live
  document.querySelector('#modeSwitch .mode-item[data-
mode="live"]')?.classList.add('active');
  await ModeManager.switchTo('live');
  showPanelFor('live', panelExpanded);
});
```

Αρχείο live.js

```
// live.js - Detector's real-time data management (plots & tables)
import { SERIES, S, Calibration } from '../core/state.js';
import { numericOrNaN, maybeBreak, trimOld,
relayoutWindowAll,renderVisibleTables,
        clearAllTables, combinedMainTraces, newPlotWithUnit,
renderLatestReading
        } from '../core/plot.js';
import { startMqttConnection, subscribeMqttValues } from
'./mqttConnection.js';
import { attachStatusFromMqtt } from '../core/status.js';
import { getEnabledChannels, isChannelEnabled } from '../core/channels.js';

let suppress = false;
let minutes = 1;
let combine = false;
const combineToggleEl = document.getElementById('combineToggle');

// Resets live plots with correct units and empty data
function resetLivePlots(){
  Object.keys(S).forEach(k => { S[k].x = []; S[k].y = []; });
  const unit = Calibration.enabled ? 'dbm' : 'volt';

  if (combine) {
    const baseDef = { plotId: 'combinedPlot', title: 'Received Signal
Strength' };
    const traces = combinedMainTraces(SERIES);
    newPlotWithUnit(baseDef, traces, unit);
  } else {
    Object.keys(SERIES).forEach(k => {
      const def = SERIES[k];
      const main = { x:[], y:[], type:'scatter', mode:'lines',
                     name: (unit === 'volt') ? "Voltage" : "Input Power",
line:{width:2, color:def.color},
                     connectgaps:false };
      newPlotWithUnit(def, [main], unit);
    });
  }
}
// Wire up the combine charts toggle checkbox
function wireCombineToggle(){
  if (!combineToggleEl) return;
  if (enter._onCombineToggle) {
    combineToggleEl.removeEventListener('change', enter._onCombineToggle);
  }
  enter._onCombineToggle = () => {
    combine = combineToggleEl.checked;
    applyPlotVisibility();
    resetLivePlots();
  };
  combineToggleEl.checked = combine;
  combineToggleEl.addEventListener('change', enter._onCombineToggle);
}
// Show/hide plots based on channel visibility and combine toggle state
function applyPlotVisibility({ force = false } = {}){
  if (!force && !document.body.classList.contains('is-live')) return;
  const combined = document.getElementById('combinedBlock');
  const b60 = document.getElementById('rss60Block');
  const b40 = document.getElementById('rss40Block');
  const ch60 = isChannelEnabled('rss60');
```

```
  const ch40 = isChannelEnabled('rss40');
  const bothEnabled = ch60 && ch40;

  if (combineToggleEl) {
    combineToggleEl.disabled = !bothEnabled;
    if (!bothEnabled && combine) {
      combine = false;
      combineToggleEl.checked = false;
    }
  }
  const showCombined = combine && bothEnabled;
  if (showCombined) {
    combined?.classList.remove('hidden');
  } else {
    combined?.classList.add('hidden');
  }
  if (b60) b60.classList.toggle('hidden', !ch60 || showCombined);
  if (b40) b40.classList.toggle('hidden', !ch40 || showCombined);
}
// Handle incoming sensor data, update latest readings, extend plots, and
trim old data
function onSensorUpdate(data){
  if (suppress) return;
  const ts   = new Date(data.timestamp);
  const tsText = ts.toLocaleString();
  const vals = {};
  if (Object.prototype.hasOwnProperty.call(data, 'zx47_60')) {
    vals.rss60 = numericOrNaN(data.zx47_60);
  }
  if (Object.prototype.hasOwnProperty.call(data, 'zx47_40')) {
    vals.rss40 = numericOrNaN(data.zx47_40);
  }
  // Latest readings
  Object.keys(vals).forEach(k => {
    if (!isChannelEnabled(k)) return;
    const v = vals[k];
    renderLatestReading(
      k,
      v,
      tsText,
      (rawValue, channelKey) => (Calibration.enabled ?
Calibration.toDbm(channelKey, rawValue) : rawValue),
      { updateTimestampOnInvalid: false }
    );
  });
  // plot
  Object.keys(vals).forEach(k => {
    if (!isChannelEnabled(k)) return;
    const def = SERIES[k], st = S[k], v = vals[k];
    if (!def || !Number.isFinite(v)) return;
    let value = Calibration.enabled ? Calibration.toDbm(k, v) : v;
    if (!Number.isFinite(value)) return;

    st.x.push(ts); st.y.push(value);

    if (combine) {
      const idx = (k === 'rss60') ? 0 : 1;
      maybeBreak('combinedPlot', st, ts, idx);
      Plotly.extendTraces('combinedPlot', { x: [[ts]], y: [[value]] },
[idx]);
    } else {
```

```
      maybeBreak(def.plotId, st, ts);
      Plotly.extendTraces(def.plotId, { x: [[ts]], y: [[value]] }, [0]);
    }
  });
  // trim and relayout
  getEnabledChannels().forEach(k => trimOld(S[k], ts, 5*60_000));
  relayoutWindowAll(minutes);
  if (combine) { Plotly.relayout('combinedPlot', { 'xaxis.range': [new
Date(ts.getTime() - minutes*60_000), ts]}); }
  renderVisibleTables(getEnabledChannels());
}
// Entry point when navigating to live mode - sets up UI, event listeners,
and MQTT bridge
export async function enter() {
  document.body.classList.add('is-live');
  const timeSel = document.getElementById('timeRange');
  if (timeSel) { timeSel.disabled = false; minutes =
parseInt(timeSel.value||'1'); }
  applyPlotVisibility();
  resetLivePlots();
  wireCombineToggle();

  if (!enter._onChannelChanged) {
    enter._onChannelChanged = () => {
      if (!document.body.classList.contains('is-live')) return;
      clearAllTables();
      applyPlotVisibility();
      resetLivePlots();
    };
    window.addEventListener('channel-visibility-changed',
enter._onChannelChanged);
  }
  if (!enter._onUnitsChanged) {
    enter._onUnitsChanged = () => {
      if (!document.body.classList.contains('is-live')) return;
      resetLivePlots();
    };
    window.addEventListener('units-changed', enter._onUnitsChanged);
  }
  clearAllTables();
  startMqttConnection();
  enter._detachStatus = attachStatusFromMqtt();
  enter._detachMqttValues = subscribeMqttValues(({ channelKey, value,
timestamp }) => {
    if (suppress) return;
    const data = { timestamp };
    if (channelKey === 'rss60') data.zx47_60 = value;
    if (channelKey === 'rss40') data.zx47_40 = value;
    onSensorUpdate(data);
  }, { ensureConnection: false });
  suppress = false;
  timeSel?.addEventListener('change', onTimeChange);
}
// Handle time window selection change
function onTimeChange(){
  const timeSel = document.getElementById('timeRange');
  minutes = parseInt(timeSel.value||'1');
  const ends = Object.keys(S)
    .map(k => S[k].x.length ? S[k].x[S[k].x.length-1] : null)
    .filter(Boolean);
  if (ends.length) relayoutWindowAll(minutes);
```

```
}
// Exit live mode - remove event listeners, clear plots, and reset UI
export async function exit() {
  suppress = true;
  if (enter._onUnitsChanged) {
    window.removeEventListener('units-changed', enter._onUnitsChanged);
    enter._onUnitsChanged = null;
  }
  if (enter._detachStatus) {
    enter._detachStatus();
    enter._detachStatus = null;
  }
  if (enter._detachMqttValues) {
    enter._detachMqttValues();
    enter._detachMqttValues = null;
  }
  if (enter._onChannelChanged) {
    window.removeEventListener('channel-visibility-changed',
enter._onChannelChanged);
    enter._onChannelChanged = null;
  }
  if (combineToggleEl && enter._onCombineToggle) {
    combineToggleEl.removeEventListener('change', enter._onCombineToggle);
    enter._onCombineToggle = null;
  }
  combine = false;
  applyPlotVisibility({ force: true });
  if (window.Plotly?.purge) {
    Plotly.purge('combinedPlot');
  }
  const timeSel = document.getElementById('timeRange');
  timeSel && timeSel.removeEventListener('change', onTimeChange);
  document.body.classList.remove('is-live');
}
```

Αρχείο experiment.js

```
// experiment.js - Experiment recording, plotting and CSV management
import { SERIES, S, Calibration, Distance } from '../core/state.js';
import { newPlotWithUnit, initPlots, numericOrNaN, renderVisibleTables,
         clearAllTables, renderTable, clearAllSeries, renderLatestReading,
         drawExperimentArrays, relayoutExperimentFullSpan,
redrawExperimentOnUnitsChanged } from '../core/plot.js';
import { CsvRecorder } from '../core/csvRecorder.js';
import { Analysis } from '../core/analysis.js';
import { getEnabledChannels, isChannelEnabled } from '../core/channels.js';
import { expList, expImportCsv, expLoadParsed, expUploadCsv, expDelete }
from '../core/api.js';
import { startMqttConnection, subscribeMqttValues } from
'./mqttConnection.js';
import { attachStatusFromMqtt } from '../core/status.js';

// ----------------- DOM refs (panel + modal) -----------------
const panel       = document.getElementById('experimentPanel');
const idInput     = document.getElementById('expId');
const startBtn    = document.getElementById('startExpBtn');
const pauseBtn    = document.getElementById('pauseExpBtn');
```

```
const stopBtn     = document.getElementById('stopExpBtn');
const timerEl     = document.getElementById('expTimer');
const exportBtn   = document.getElementById('downloadExpCsv');
const browseBtn   = document.getElementById('browseExperimentsBtn');
const importBtn   = document.getElementById('importCsvBtn');
const importInput = document.getElementById('importCsvInput');

const expModal      = document.getElementById('expModal');
const expModalClose = document.getElementById('expModalClose');
const expTable      = document.getElementById('expTable');
const expTableBody  = document.getElementById('expTableBody');
const expModalEmpty = document.getElementById('expModalEmpty');

let timerInterval = null;
let startTime = null;
let currentExp = null;   // Global variable to hold the current experiment
data and state
let plotsInited = false;
let latestDataTs = null; // Timestamp of the latest received data point
let csv = null;
let EXP_SNAPSHOT = null; // active calibration profile
let EXP_FREQ_MHz = null; // active frequency for experiment

const CHANNEL_DEVICE_LABELS = { rss60: 'ZX47-60', rss40: 'ZX47-40' };
const CHANNEL_UI = Object.freeze({
  rss60: { distanceInputId: 'distInput_rss60',
           gainInputId: 'rxGain_rss60',
           lossInputId: 'sysLoss_rss60',
           blockId: 'rss60Block' },
  rss40: { distanceInputId: 'distInput_rss40',
           gainInputId: 'rxGain_rss40',
           lossInputId: 'sysLoss_rss40',
           blockId: 'rss40Block' } });
// Attach MQTT listener for receiving live detector values during an active
experiment
// and recording them into the current experiment data and CSV.
let expMqttDetach = null;
let expDetachStatus = null;
function attachMqtt() {
  if (!document.body.classList.contains('is-experiment')) return;
  if (expMqttDetach) return;
  expMqttDetach = subscribeMqttValues(({ device, channelKey, value, ts,
timestamp }) => {
    if (!currentExp || currentExp.end || currentExp.paused) return;
    if (!isChannelEnabled(channelKey)) return;

    const data = { timestamp };
    if (channelKey === 'rss60') data.zx47_60 = value;
    if (channelKey === 'rss40') data.zx47_40 = value;
    onSensorUpdate(data);

    const iso = new Date(ts).toISOString();
    const budget = (window.currentExp && window.currentExp.budget) || null;

    if (csv) {
      const dbm = Calibration.toDbmWithSnapshot(EXP_SNAPSHOT, channelKey,
value);
      if (!Number.isFinite(dbm)) return;
      const coeffs = EXP_SNAPSHOT?.devices?.[channelKey] || null;
      const aCoeff = (coeffs && Number.isFinite(+coeffs.a)) ? +coeffs.a :
'';
```

```
      const bCoeff = (coeffs && Number.isFinite(+coeffs.b)) ? +coeffs.b :
'';
      const dist = Distance[channelKey] ?? '';
      const pl   = (budget && budget[channelKey])
                   ? (budget.Pt + budget.Gt + budget[channelKey].Gr -
budget[channelKey].Lsys - dbm)
                   : NaN;
      csv.appendRow([ iso, device, value, dbm,
                      EXP_FREQ_MHz ?? '', aCoeff, bCoeff, dist,
(Number.isFinite(pl) ? Math.round(pl*10)/10 : '') ]);
    }
  }, { ensureConnection: true });
}
// Detach the MQTT listener when experiment ends or is paused to stop
recording live data
function detachMqtt() {
  if (!expMqttDetach) return;
  expMqttDetach();
  expMqttDetach = null;
}
// Get the current experiment object from the global state
function getExp() {
  if (!window.currentExp) window.currentExp = {};
  return window.currentExp;
}
// Check if there is an active experiment that has started but not yet
ended
function isExperimentActive() {
  return !!(currentExp && !currentExp.end);
}
// Determine the active frequency for the experiment.
function updateExperimentFrequency() {
  const freqMHz = (Number.isFinite(+Calibration.activeFreqMHz) &&
+Calibration.activeFreqMHz > 0)
    ? Math.round(+Calibration.activeFreqMHz)
    : null;
  EXP_FREQ_MHz = freqMHz;
  window.EXP_FREQ_MHz = EXP_FREQ_MHz;
  if (currentExp) {
    if (freqMHz != null) {
      currentExp.freqMHz = freqMHz;
    } else if (currentExp.freqMHz == null) {
      currentExp.freqMHz = null;
    }
    window.currentExp = currentExp;
  }
  EXP_SNAPSHOT = Calibration.getActiveProfile(freqMHz);
}
// ----------------- Panel status / timer -----------------
// Check if an experiment is currently active (started but not ended)
function isStreaming() {
  return !!(currentExp && !currentExp.end && !currentExp.paused);
}
// Update the timer label to show elapsed time since experiment start
(mm:ss)
function setTimerLabel() {
  if (!startTime) { timerEl.textContent = ''; return; }
  const diff = Math.floor((Date.now() - startTime.getTime()) / 1000);
  const mins = Math.floor(diff / 60);
  const secs = diff % 60;
```

```
  timerEl.textContent =
`(${String(mins).padStart(2,'0')}:${String(secs).padStart(2,'0')})`;
}
function startTimer() {
  stopTimer({ clearLabel: false });
  setTimerLabel();
  timerInterval = setInterval(setTimerLabel, 1000);
}
function stopTimer({ clearLabel = true } = {}) {
  if (timerInterval) { clearInterval(timerInterval); timerInterval = null;
}
  if (clearLabel) { timerEl.textContent = ''; }
}
function setStatusText(txt) {
  const span = panel?.querySelector('.exp-status-text');
  if (span) span.textContent = txt;
}
function resetPauseButton() {
  if (!pauseBtn) return;
  pauseBtn.disabled = true;
  pauseBtn.textContent = 'Pause';
}
// Reset the experiment form to its initial state
function resetExperimentForm() {
  stopTimer();
  startTime = null;
  if (idInput) idInput.disabled = false;
  if (startBtn) startBtn.disabled = false;
  if (stopBtn) stopBtn.disabled = true;
  resetPauseButton();
  setDistanceInputsEnabled(true);
  setBudgetInputsEnabled(true);
}
// Update the UI to reflect that there is no active experiment
function showFormMode() {
  setStatusText('No active experiment');
  Analysis.setAvailability(false);
  resetExperimentForm();
  applyExportRule();
}
// Update the UI to reflect that the experiment is paused
function showPausedMode(exp) {
  setStatusText(`Paused: ${exp.id || '(no id)'}`);
  Analysis.setAvailability(false);
  idInput.disabled = true;
  startBtn.disabled = true;
  stopBtn.disabled = false;
  pauseBtn.disabled = false;
  pauseBtn.textContent = 'Resume';
  setDistanceInputsEnabled(true);
}
// Update the UI to reflect that the experiment is running
function showRunningMode(exp) {
  setStatusText(`Running: ${exp.id || '(no id)'}`);
  Analysis.setAvailability(false);
  idInput.disabled = true;
  startBtn.disabled  = true;
  stopBtn.disabled   = false;
  pauseBtn.disabled  = false;
  pauseBtn.textContent = 'Pause';
  startTime = new Date(exp.start);
```

```
  startTimer();
  setDistanceInputsEnabled(false);
}
// Parse a numeric value from an input field by ID, returning null if
invalid
function parseNumber(elId) {
  const el = document.getElementById(elId);
  if (!el) return null;
  const raw = el.value;
  if (raw == null || String(raw).trim() === '') return null;
  const v = Number(raw);
  return (Number.isFinite(v) && v >= 0) ? v : null;
}
// Read distance values from the input fields for enabled channels before
starting or resuming
// and alert if any required distance is missing based on the enabled
channels.
function readAndSetDistancesOrAlert(actionLabel) {
  const missing = [];
  const enabled = getEnabledChannels();

  Object.entries(CHANNEL_UI).forEach(([channel, { distanceInputId }]) => {
    const value = parseNumber(distanceInputId);
    if (enabled.includes(channel)) {
      if (value === null) missing.push(CHANNEL_DEVICE_LABELS[channel]);
      else Distance[channel] = value;
    } else {
      Distance[channel] = null;
    }
  });
  if (missing.length) {
    alert(`Enter all parameters for: ${missing.join(', ')} before
${actionLabel}.`);
    return false;
  }
  return true;
}
// Read link budget parameters from the input fields before starting
// and alert if any required parameter is missing based on the enabled
channels.
function readAndSetBudgetOrAlert() {
  const Pt   = parseNumber('txPowerDbm');
  const Gt   = parseNumber('txGainDbi');
  const Gr60 = parseNumber('rxGain_rss60');
  const Gr40 = parseNumber('rxGain_rss40');
  const L60  = parseNumber('sysLoss_rss60');
  const L40  = parseNumber('sysLoss_rss40');
  const missing = [];
  const enabled = getEnabledChannels();

  if (enabled.length) {
    if (Pt   == null) missing.push('Tx power');
    if (Gt   == null) missing.push('Tx antenna gain');
  }
  if (enabled.includes('rss60')) {
    if (Gr60 == null) missing.push('Rx gain Ch1');
    if (L60  == null) missing.push('System loss Ch1');
  }
  if (enabled.includes('rss40')) {
    if (Gr40 == null) missing.push('Rx gain Ch2');
    if (L40  == null) missing.push('System loss Ch2');
```

```
  }

  if (missing.length) {
    alert(`Enter all link-budget parameters:\n${missing.join(', ')}`);
    return false;
  }

  const exp = getExp();
  const budget = {
    Pt: Pt ?? null,
    Gt: Gt ?? null,
  };
  if (enabled.includes('rss60')) {
    budget.rss60 = { Gr: Gr60, Lsys: L60 };
  }
  if (enabled.includes('rss40')) {
    budget.rss40 = { Gr: Gr40, Lsys: L40 };
  }
  exp.budget = budget;
  return true;
}
// Enable/disable distance input fields based on enabled channels and
whether the experiment is running.
function setDistanceInputsEnabled(enabled) {
  Object.entries(CHANNEL_UI).forEach(([channel, { distanceInputId }]) => {
    const el = document.getElementById(distanceInputId);
    if (!el) return;
    const allow = enabled && isChannelEnabled(channel);
    el.disabled = !allow;
  });
}
// Enable/disable budget input fields based on whether the experiment is
running and channel visibility.
function setBudgetInputsEnabled(enabled) {
  ['txPowerDbm','txGainDbi'].forEach(id => {
    const el = document.getElementById(id);
    if (el) el.disabled = !enabled;
  });
  Object.entries(CHANNEL_UI).forEach(([channel, { gainInputId, lossInputId
}]) => {
    [gainInputId, lossInputId].forEach(id => {
      const el = document.getElementById(id);
      if (!el) return;
      const allow = enabled && isChannelEnabled(channel);
      el.disabled = !allow;
    });
  });
}
// Clear distance input fields and reset Distance values
function clearExperimentInputsOnEnter() {
  const inputEl = document.getElementById('expId');
  if (inputEl) inputEl.value = '';

  Object.entries(CHANNEL_UI).forEach(([channel, { distanceInputId }]) => {
    const el = document.getElementById(distanceInputId);
    if (el) el.value = '';
    Distance[channel] = null;
  });
  // Enable all inputs when experiment is not active.
  setDistanceInputsEnabled(true);
  setBudgetInputsEnabled(true);
```

```
}
// Show/hide channel block based on channel visibility and enable/disable
based on experiment state.
function applyChannelVisibilityExperiment() {
  if (!document.body.classList.contains('is-experiment')) return;
  const combined = document.getElementById('combinedBlock');
  combined?.classList.add('hidden');
  if (window.Plotly?.purge) {
    Plotly.purge('combinedPlot');
  }
  Object.entries(CHANNEL_UI).forEach(([channel, { blockId }]) => {
    const block = document.getElementById(blockId);
    if (!block) return;
    block.classList.toggle('hidden', !isChannelEnabled(channel));
  });

  const running = isStreaming();
  setDistanceInputsEnabled(!running);
  setBudgetInputsEnabled(!running);
}
// Handle changes in channel visibility by redrawing plots, updating tables
// and applying experiment-specific UI adjustments.
export function onChannelVisibilityChange() {
  if (!document.body.classList.contains('is-experiment')) return;
  applyChannelVisibilityExperiment();
  drawExperimentArrays(getEnabledChannels());
  relayoutExperimentFullSpan(getEnabledChannels(), {
    isStreaming: isStreaming(),
    latestDataTs
  });
  renderVisibleTables(getEnabledChannels());
}
// ----------------- Export link helpers -----------------
// Update the export link and button state based on whether there is a
valid experiment to export.
function updateExportLink(exp) {
  if (!exportBtn) return;
  if (!exp || !exp.id) {
    setExportDisabled(true);
    exportBtn.href = '#';
    exportBtn.removeAttribute('download');
    return;
  }
  const safeId = String(exp.id).trim().replace(/\s+/g,'_');
  exportBtn.href =
`/api/experiments/${encodeURIComponent(exp.id)}/download`;
  exportBtn.download = `${safeId}.csv`;
}
// Enable/disable the export button
function setExportDisabled(disabled) {
  if (!exportBtn) return;
  exportBtn.classList.toggle('disabled', !!disabled);
  exportBtn.setAttribute('aria-disabled', disabled ? 'true' : 'false');
  if (disabled) {
    exportBtn.removeAttribute('href');
    exportBtn.removeAttribute('download');
  }
}
// Apply the export button state based on whether there is a finished
experiment available.
async function applyExportRule() {
```

```
  if (!exportBtn) return;
  if (currentExp && currentExp.end) { setExportDisabled(false); return; }
  setExportDisabled(true);
}
// ----------------- Modal helpers -----------------
function openExpModal() {
  if (!expModal) return;
  expModal.classList.remove('hidden');
  expModal.setAttribute('aria-hidden','false');
  loadExperimentsIntoModal();
}
function closeExpModal() {
  if (!expModal) return;
  expModal.classList.add('hidden');
  expModal.setAttribute('aria-hidden','true');
}
function fmtDt(s){ try { return s ? new Date(s).toLocaleString() : '--'; }
catch { return s||'--'; } }
// Load the list of experiments from the server
async function loadExperimentsIntoModal() {
  if (!expTableBody || !expModalEmpty || !expTable) return;
  expTableBody.innerHTML = '';
  expModalEmpty.classList.add('hidden');
  expTable.classList.add('hidden');
  try {
    const exps = await expList();
    if (!exps || !exps.length) {
      expModalEmpty.textContent = 'No experiments found.';
      expModalEmpty.classList.remove('hidden');
      return;
    }
    // Sort experiments by modification time
    exps.slice()
      .sort((a,b) => new Date(b.mtime || 0) - new Date(a.mtime || 0))
      .forEach(exp => {
        const tr = document.createElement('tr');
        tr.innerHTML = `
          <td>${exp.id || '--'}</td>
          <td>${fmtDt(exp.mtime)}</td>
          <td>${(exp.size_bytes ?? 0).toLocaleString()} bytes</td>
          <td>
            <button class="btn btn--modal action-btn use-exp" data-
id="${exp.id}">Use</button>
            <button class="btn btn--modal action-btn danger del-exp" data-
id="${exp.id}">Delete</button>
          </td>`;
        expTableBody.appendChild(tr);
      });
    expTable.classList.remove('hidden');
  } catch {
    expModalEmpty.textContent = 'Failed to load experiments.';
    expModalEmpty.classList.remove('hidden');
  }
}
// ----------------- Import CSV Files -----------------
importBtn?.addEventListener('click', () => importInput?.click());
importInput?.addEventListener('change', async () => {
  const file = importInput.files?.[0];
  if (!file) return;
  importInput.value = '';
  if (!file.name.toLowerCase().endsWith('.csv')) {
```

```
    alert('Please select a CSV file.');
    return;
  }

  const suggested = (file.name.replace(/\.csv$/i, '') ||
'experiment').trim();
  let id = prompt('Enter an ID for this experiment:', suggested);
  if (id === null) return;
  id = id.trim();
  if (!id) {
    alert('Experiment name is required.');
    return;
  }
  const buildFormData = () => {
    const fd = new FormData();
    fd.append('file', file, file.name);
    fd.append('exp_id', id);
    return fd;
  };
  let overwrite = false;
  while (true) {
    let resp;
    try {
      resp = await expImportCsv(buildFormData(), { overwrite });
    } catch {
      alert('Failed to import CSV. Please check your connection and try
again.');
      return;
    }
    if (resp.status === 409) {
      let payload = null;
      try { payload = await resp.json(); } catch (e) { /* ignore */ }
      const confirmMsg = payload?.error ? `${payload.error} Overwrite it?`
        : 'An experiment with this ID already exists. Overwrite it?';
      if (confirm(confirmMsg)) {
        overwrite = true;
        continue;
      }
      return;
    }
    if (!resp.ok) {
      const text = await resp.text().catch(() => '');
      alert(text ? `Failed to import CSV: ${text}` : 'Failed to import
CSV.');
      return;
    }
    try {
      await loadExperimentsIntoModal();
    } catch {}
    alert(`CSV imported successfully as "${id}".`);
    return;
  }
});
// ----------------- Charts -----------------
function onUnitsChangedExperiment() {
  if (!document.body.classList.contains('is-experiment')) {
    return;
  }
  redrawExperimentOnUnitsChanged(getEnabledChannels());
}
```

```
// Parse the data arrays for active channels and redraw the plots with
appropriate units.
function onSensorUpdate(data) {
  if (!currentExp || currentExp.end || currentExp.paused) return;
  const ts = new Date(data.timestamp);
  latestDataTs = ts;
  const tsText = ts.toLocaleString();
  const vals = {};
  // Read the incoming detector values for each channel and store them.
  if (Object.prototype.hasOwnProperty.call(data, 'zx47_60')) {
    vals.rss60 = numericOrNaN(data.zx47_60);
  }
  if (Object.prototype.hasOwnProperty.call(data, 'zx47_40')) {
    vals.rss40 = numericOrNaN(data.zx47_40);
  }
  Object.keys(vals).forEach(k => {
    if (!isChannelEnabled(k)) return;
    const def = SERIES[k], st = S[k], v = vals[k];
    if (!def || !Number.isFinite(v)) return;

    if (!st.yVolt) st.yVolt = [];
    if (!st.yDbm)  st.yDbm  = [];
    if (!st.dist) st.dist = [];
    if (!st.freq) st.freq = [];
    if (!st.pl) st.pl = [];

    const dbm = Calibration.toDbmWithSnapshot(EXP_SNAPSHOT, k, v);
    const vShow = Calibration.enabled ? (Number.isFinite(dbm) ? dbm : NaN)
: v;

    st.x.push(ts);
    st.yVolt.push(v);
    st.yDbm.push(Number.isFinite(dbm) ? dbm : NaN);
    st.dist.push(Distance[k] ?? null);
    st.freq.push(EXP_FREQ_MHz ?? null);
    const budget = currentExp?.budget;
    const pl = (budget && budget[k] && Number.isFinite(+dbm))
      ? (budget.Pt + budget.Gt + budget[k].Gr - budget[k].Lsys - dbm) :
NaN;
    st.pl.push(Number.isFinite(pl) ? Math.round(pl*10)/10 : '');

    const unit = Calibration.enabled ? 'dbm' : 'volt';
    const gd   = document.getElementById(def.plotId);
    if (!gd || !gd.data || !gd.data.length ||
!Array.isArray(gd.data[0]?.y)) {
      newPlotWithUnit(def, [{
        x: [ts], y: [vShow], type: 'scatter', mode: 'markers', connectgaps:
false, marker: { size: 4, color: def.color },
        name: 'RSS', fill: 'none'
      }], unit);
      return; // next message will use extendTraces
    }
    Plotly.extendTraces(def.plotId, { x: [[ts]], y: [[vShow]] }, [0]);
  });
  relayoutExperimentFullSpan(getEnabledChannels(), {
    isStreaming: isStreaming(),
    latestDataTs
  });
  renderVisibleTables(getEnabledChannels());
  // Update the latest value display for each active channel in the UI.
  Object.keys(vals).forEach(k => {
```

```
    if (!isChannelEnabled(k)) return;
    const v = vals[k];
    renderLatestReading(
      k,
      v,
      tsText,
      (rawValue, channelKey) => (
        Calibration.enabled
          ? Calibration.toDbmWithSnapshot(EXP_SNAPSHOT, channelKey,
rawValue)
          : rawValue
      )
    );
  });
}
// ----------------- Load a Finished Experiment -----------------
async function loadFinishedExperiment(id) {
  Analysis.setAvailability(false);
  Analysis.resetForNewExperiment();
  clearAllTables();
  try {
    const parsed = await expLoadParsed(id);

    window.currentExp = window.currentExp || {};
    window.currentExp.freqMHz = +parsed.freqMHz;
    currentExp = window.currentExp;

    ['rss60','rss40'].forEach(k => {
      S[k].x     = parsed.byDevice[k].x.map(ms => new Date(ms));
      S[k].yVolt = parsed.byDevice[k].vVolt;
      S[k].yDbm  = parsed.byDevice[k].vDbm;
      S[k].dist  = parsed.byDevice[k].vDist;
      S[k].freq  = parsed.byDevice[k].vFreq;
      S[k].pl    = parsed.byDevice[k].vPL;
    });

    latestDataTs = null;
    onChannelVisibilityChange();
    applyChannelVisibilityExperiment();
    Analysis.setAvailability(true);
    return;
  } catch (_) {
    throw new Error('csv could not be loaded');
  }
}
// ----------------- Mode API -----------------
// Enter the experiment mode. Set up the UI, attach MQTT listeners and
prepare to start an experiment.
export async function enter() {
  document.body.classList.add('is-experiment');
  Analysis.setExperimentMode(true);
  if (panel) panel.classList.remove('hidden');
  clearExperimentInputsOnEnter();
  document.getElementById('timeRange')?.setAttribute('disabled','true');

  if (!plotsInited) { initPlots(); plotsInited = true; }

  currentExp = null;
  window.currentExp = null;
  EXP_FREQ_MHz = null;
  window.EXP_FREQ_MHz = null;
```

```
  showFormMode();
  clearAllSeries();
  clearAllTables();
  onChannelVisibilityChange();
  applyChannelVisibilityExperiment();
  detachMqtt();
  startMqttConnection();
  if (!expDetachStatus) {
    expDetachStatus = attachStatusFromMqtt();
  }

  window.addEventListener('units-changed', onUnitsChangedExperiment);

  window.handleTableToggle = (key) => {
    if (!document.body.classList.contains('is-experiment')) return;
    renderTable(key);
    // resize plot when table is shown/hidden
    const plotEl = document.getElementById(SERIES[key].plotId);
    if (plotEl && window.Plotly) setTimeout(() =>
Plotly.Plots.resize(plotEl), 0);
  };
}
// Exit the experiment mode. Clean up the UI, detach MQTT listeners and
reset state.
export async function exit() {
  document.body.classList.remove('is-experiment');
  Analysis.setExperimentMode(false);
  Analysis.setAvailability(false);
  if (panel) panel.classList.add('hidden');
  if (!isExperimentActive()) {
    setDistanceInputsEnabled(true);
  }
  delete window.handleTableToggle;
  detachMqtt();
  if (expDetachStatus) {
    expDetachStatus();
    expDetachStatus = null;
  }
  window.removeEventListener('units-changed', onUnitsChangedExperiment);
}
// ----------------- Panel events -----------------
// Handle the export button click.
exportBtn?.addEventListener('click', (e) => {
  if (exportBtn.classList.contains('disabled')) {
    e.preventDefault();
    alert('No completed experiment available for export.');
    return;
  }
});
// Check if an experiment with the given id already exists by listing the
experiments from the server.
async function experimentIdExists(id) {
  try {
    const exps = await expList();
    const safeId = id.replace(/[^\w.-]+/g, '_');
    return exps.some(e => e && String(e.id || '').trim() === safeId);
  } catch { return false; }
}
// Handle the start button click. Validate inputs, set up the experiment
state,
// attach MQTT listeners and start recording data
```

```
startBtn?.addEventListener('click', async () => {
  const id = idInput.value.trim();
  if (!id) { alert('Please enter a name for the experiment.'); return; }

  if (await experimentIdExists(id)) {
    alert('An experiment with this name already exists. Please choose a
different ID.');
    return;
  }
  if (!Calibration.activeFreqMHz) {
    alert('Activate a calibration profile before starting an experiment.');
    return;
  }
  if (!readAndSetDistancesOrAlert('starting')) return;
  if (!readAndSetBudgetOrAlert()) return;
  setBudgetInputsEnabled(false);

  const safeId = id.replace(/[^\w.-]+/g, '_');
  currentExp = {
    ...(window.currentExp || {}),
    id: safeId,
    start: new Date().toISOString(),
    end: null,
    paused: false,
  };
  window.currentExp = currentExp;
  try {
    pauseBtn && (pauseBtn.disabled = false, pauseBtn.textContent =
'Pause');
    showRunningMode(currentExp);
    setExportDisabled(true);
    updateExportLink(null);
    applyExportRule();
    clearAllSeries();
    clearAllTables();

    updateExperimentFrequency();
    drawExperimentArrays(getEnabledChannels());
    // create a comment with experiment info for the csv
    const expInfo = (() => {
      const snapshot = EXP_SNAPSHOT.freqMHz != null
        ? { freqMHz: EXP_SNAPSHOT.freqMHz }
        : {};
      const budget = currentExp?.budget || null;
      if (budget) snapshot.budget = budget;
      return Object.keys(snapshot).length ? snapshot : null;
    })();
    csv = new CsvRecorder({
      filename: `${safeId}.csv`,
      header:
['timestamp','device','volt','dbm','freq_mhz','a','b','distance_m','path_lo
ss_db'],
      comment: expInfo ? [`#exp-info:${JSON.stringify(expInfo)}`] : [],
    });
    csv.start();
    attachMqtt();
  } catch (err) {
    alert(err.message || 'Failed to start experiment');
  }
});
// Handle the pause button click. Toggle between paused and running states.
```

```
pauseBtn?.addEventListener('click', async () => {
  if (!currentExp || currentExp.end) return;
  if (currentExp.paused) {
    if (!readAndSetDistancesOrAlert('resuming')) return;
    currentExp.paused = false;
    updateExperimentFrequency();
    showRunningMode(currentExp);
  } else {
    currentExp.paused = true;
    showPausedMode(currentExp);
  }
  window.currentExp = currentExp;
  applyExportRule();
});
// Handle the stop button click. Finalize the experiment data,
// upload the CSV to the server, reset the UI and update the export link.
stopBtn?.addEventListener('click', async () => {
  if (!confirm('Stop the current experiment?')) return;
  try {
    if (csv && currentExp) {
      const out = csv.stop();
      if (out && currentExp.id) {
        const fd = new FormData();
        fd.append('file', out.blob, `${currentExp.id}.csv`);
        fd.append('exp_id', currentExp.id);
        try {
          const resp = await expUploadCsv(currentExp.id, fd);
          if (!resp.ok) await resp.text().catch(()=> '');
        } catch {}
      }
      csv = null;
    }
    currentExp.end = new Date().toISOString();
    latestDataTs = null;
    detachMqtt();
    resetExperimentForm();
    updateExportLink(currentExp);
    setExportDisabled(false);
    setStatusText(`Finished: ${currentExp.id || '(no id)'}`);
    Analysis.setAvailability(true);
  } catch (err) {
    alert(err.message || 'Failed to stop experiment');
  }
});
// ----------------- Browse Modal wiring -----------------
browseBtn?.addEventListener('click', openExpModal);
expModalClose?.addEventListener('click', closeExpModal);
// Close the browse modal when clicking outside the content area
expModal?.addEventListener('click', (e) => {
  if (e.target === expModal || e.target.classList.contains('modal-
backdrop')) closeExpModal();
});
// Handle clicks on the experiment list in the modal for loading or
deleting experiments.
expTableBody?.addEventListener('click', async (e) => {
  const useBtn = e.target.closest('.use-exp');
  const delBtn = e.target.closest('.del-exp');
  if (useBtn) {
    const id = useBtn.dataset.id;
    const nextExp = { id, end: new Date().toISOString(), paused: false };
    currentExp = nextExp;
```

```
    window.currentExp = nextExp;

    updateExportLink(nextExp);
    setExportDisabled(false);
    applyExportRule();
    detachMqtt();
    resetExperimentForm();
    closeExpModal();
    setStatusText(`Loaded: ${id}`);

    try {
      await loadFinishedExperiment(id);
    } catch (e) {
      alert('Failed to load experiment data.');
      return;
    }
  }
  if (delBtn) {
    const id = delBtn.dataset.id;
    if (!id) return;
    if (!confirm('Delete this experiment? This cannot be undone.')) return;
    try {
      const resp = await expDelete(id);
      if (!resp.ok) {
        const err = await resp.json().catch(()=>({error:'Failed to
delete'}));
        alert(err.error || 'Failed to delete experiment.');
        return;
      }
      if (exportBtn?.href.includes(id)) updateExportLink(null);
      await loadExperimentsIntoModal();
      if (currentExp && currentExp.id === id && currentExp.end) {
        currentExp = null;
        clearAllSeries();
        drawExperimentArrays(getEnabledChannels());
        clearAllTables();
      }
    } catch { alert('Failed to delete experiment.'); }
  }
});
// Initial panel state
exportBtn?.classList.remove('hidden');
setExportDisabled(true);
```

Αρχείο mqttConnection.js

```
// mqttConnection.js - Bridge between MQTT messages and in-browser events
for live/experiment modes

const DEV_A = window.MQTT_DEV_A || 'esp32_60';
const DEV_B = window.MQTT_DEV_B || 'esp32_40';
// Map MQTT device identifiers to channel keys
function deviceToChannel(device) {
  if (device === DEV_A) return 'rss60';
  if (device === DEV_B) return 'rss40';
  return null;
}
```

```
// Connect to MQTT broker as client and set up topic subscriptions and
message handling
export function startMqttConnection() {
  const url = window.MQTT_URL;
  const topicData  = window.MQTT_TOPIC;
  const topicState = 'home/rfdetector/+/status';
  if (!url || !topicData || !window.mqtt) {
    return;
  }
  const existing = window.__mqttClient;
  if (existing && !existing.disconnected) {
    return existing;
  }

  const client = window.mqtt.connect(url, { keepalive: 25, reconnectPeriod:
1000 });
  window.__mqttClient = client;

  client.on('connect', () => {
    client.subscribe(topicData, { qos: 0 });
    client.subscribe(topicState, { qos: 0 });
    window.dispatchEvent(new CustomEvent('mqtt:conn', { detail: {
connected: true }}));
  });
  client.on('reconnect', () => {
    window.dispatchEvent(new CustomEvent('mqtt:conn', { detail: {
reconnecting: true }}));
  });
  client.on('close', () => {
    window.dispatchEvent(new CustomEvent('mqtt:conn', { detail: {
connected: false }}));
    window.__mqttClient = null;
  });
  client.on('message', (t, payload) => {
    const parts = String(t || '').split('/');
    const dev = parts[2];
    const leaf = parts[3];
    if (dev !== DEV_A && dev !== DEV_B) return;

    if (leaf === 'status') {
      const s = payload.toString().trim().toLowerCase();
      const online = (s === 'online');
      window.dispatchEvent(new CustomEvent('mqtt:status', {
        detail: { device: dev, online }
      }));
      return;
    }
    if (leaf === 'voltage') {
      const value = Number(payload.toString().trim());
      if (!Number.isFinite(value)) return;
      window.dispatchEvent(new CustomEvent('mqtt:value', {
        detail: { device: dev, value, ts: Date.now() }
      }));
    }
  });
}
// Subscribe to normalized mqtt:value samples used by live/experiment
modes.
// Returns a detach function.
export function subscribeMqttValues(onSample, { ensureConnection = true } =
{}) {
```

```
  if (typeof onSample !== 'function') return () => {};
  if (ensureConnection) startMqttConnection();

  const listener = (ev) => {
    const { device, value, ts } = ev.detail || {};
    if (typeof value !== 'number' || !device) return;
    const channelKey = deviceToChannel(device);
    if (!channelKey) return;
    const epoch = ts || Date.now();
    onSample({
      device,
      channelKey,
      value,
      ts: epoch,
      timestamp: new Date(epoch).toISOString()
    });
  };

  window.addEventListener('mqtt:value', listener);
  return () => window.removeEventListener('mqtt:value', listener);
}
```

Αρχείο modeManager.js

```
// core/modeManager.js - Mode registry and switching controller
export const ModeManager = {
  current: null,
  modes: {},
  register(name, mod) { this.modes[name] = mod; },
  async switchTo(name) {
    if (name === this.current) return;
    if (this.current && this.modes[this.current]?.exit) {
      await this.modes[this.current].exit();
    }
    this.current = name;
    if (this.modes[name]?.enter) {
      await this.modes[name].enter();
    }
  }
};
```

Αρχείο state.js

```
// core/state.js - Application state management
// Contains global state variables and calibration profile management
import { calGetProfiles } from './api.js';

// ------ Plotting Series Definitions -------------
export const YRANGE = Object.freeze({ volt: [0,3], dbm: [-70, 10] });
export const SERIES = {
  rss60: {
    plotId: 'rss60Plot',
    title: 'Received Signal Strength',
    yTitle: 'Voltage (V)',
    color: '#3E51D6',
```

```
    liveEl: 'rss60',
    liveTsEl: 'rss60Timestamp',
    tableBody: 'rss60TableBody',
    fmt: v => v == null || isNaN(v) ? '--' : v.toFixed(2),
  },
  rss40: {
    plotId: 'rss40Plot',
    title: 'Received Signal Strength',
    yTitle: 'Voltage (V)',
    color: '#8F46B3',
    liveEl: 'rss40',
    liveTsEl: 'rss40Timestamp',
    tableBody: 'rss40TableBody',
    fmt: v => v == null || isNaN(v) ? '--' : v.toFixed(2),
  }
};
export const Distance = { rss60: null, rss40: null, };
export const S = Object.fromEntries(Object.keys(SERIES).map(k => [k, { x:
[], y: [] }]));
export const SERIES_KEYS = Object.freeze(Object.keys(SERIES));

// ------ Calibration Profile Management -------------
const CALIBRATION_ACTIVE_KEY = 'calibration_active'; // { freqMHz:
number|null, unit: 'volt'|'dbm' }
// Read/write active profile to localStorage
function readActiveProfile() {
  try { return JSON.parse(localStorage.getItem(CALIBRATION_ACTIVE_KEY)) ||
null; }
  catch { return null; }
}
function writeActiveProfile({ freqMHz = null, unit = 'volt' } = {}) {
  const payload = { freqMHz: Number.isFinite(+freqMHz) ?
Math.round(+freqMHz) : null,
                    unit: unit === 'dbm' ? 'dbm' : 'volt' };
  try { localStorage.setItem(CALIBRATION_ACTIVE_KEY,
JSON.stringify(payload)); } catch {}
  return payload;
}
// Calibration management object
export const Calibration = {
  enabled: false,            // toggle globally (UI sets this)
  activeFreqMHz: null,       // current frequency (User sets this)
  profiles: {},              // { [freqMHz]: { rss60:{a,b}, rss40:{a,b} } }
  // Load profiles from server and set active profile from localStorage if
a valid one exists
  async initFromServer() {
    try {
      const profilesResp = await calGetProfiles();
      this.profiles = profilesResp?.profiles || {};
      // active comes from localStorage (per client) but must also exist in
profiles
      const local = readActiveProfile() || { freqMHz: null, unit: 'volt' };
      const localFreq = Number.isFinite(+local.freqMHz) ?
Math.round(+local.freqMHz) : null;
      const prof = localFreq != null ? this.profiles?.[String(localFreq)] :
null;
      const hasBothCoeffs = !!(
        prof
        && Number.isFinite(prof?.rss60?.a) &&
Number.isFinite(prof?.rss60?.b)
```

```
        && Number.isFinite(prof?.rss40?.a) &&
Number.isFinite(prof?.rss40?.b)
      );
      this.activeFreqMHz = hasBothCoeffs ? localFreq : null;
      this.enabled = hasBothCoeffs && (local.unit === 'dbm');
      writeActiveProfile({ freqMHz: this.activeFreqMHz, unit: this.enabled
? 'dbm' : 'volt' });
      window.dispatchEvent(new CustomEvent('calibration-loaded', {
        detail: { freqMHz: this.activeFreqMHz, unit: this.enabled ? 'dbm' :
'volt' }
      }));
    } catch {}
  },
  // Set enabled/disabled state and update localStorage
  setEnabled(on) {
    this.enabled = !!on;
    writeActiveProfile({ freqMHz: this.activeFreqMHz, unit: this.enabled ?
'dbm' : 'volt' });
    window.dispatchEvent(new CustomEvent('units-changed', {
      detail: { unit: this.enabled ? 'dbm' : 'volt' }
    }));
  },
  // Set active frequency and update localStorage
  setActiveFreq(mhz) {
    const f = Number.isFinite(+mhz) ? Math.round(+mhz) : null;
    this.activeFreqMHz = f;
    writeActiveProfile({ freqMHz: f, unit: this.enabled ? 'dbm' : 'volt'
});
    window.dispatchEvent(new CustomEvent('calibration-updated', { detail: {
freqMHz: f } }));
  },
  // Add or update a calibration profile
  updateLocalProfile(freqMHz, { rss60, rss40 }) {
    const f = Number(freqMHz);
    if (!Number.isFinite(f)) return;
    const key = String(Math.round(f));
    const existing = this.profiles[key] || {};
    this.profiles[key] = {
      ...existing,
      ...(rss60 ? { rss60 } : {}),
      ...(rss40 ? { rss40 } : {}),
    };
  },
  // Get calibration coefficients for active frequency
  getCoeffs(kind) {
    if (!this.activeFreqMHz) return null;
    const prof = this.profiles[String(this.activeFreqMHz)];
    return prof ? (prof[kind] || null) : null;
  },
  // Check if calibration coefficients exist for active frequency
  hasCoeffs(kind) {
    const c = this.getCoeffs(kind);
    return !!(c && Number.isFinite(c.a) && Number.isFinite(c.b));
  },
  // Check if both rss60 and rss40 coefficients exist for active frequency
  hasActiveProfileForBoth() {
    return this.hasCoeffs('rss60') && this.hasCoeffs('rss40');
  },
  // Convert voltage to dBm using active profile
  toDbm(kind, v) {
    const c = this.getCoeffs(kind);
```

```
    const n = Number(v);
    if (!c || !Number.isFinite(n)) return NaN;
    const raw = c.a * n + c.b;
    return Math.round(raw * 100) / 100;
  },
  // Get the active calibration profile parameters
  getActiveProfile(freqMHz) {
    const f = Number.isFinite(+freqMHz) ? Math.round(+freqMHz) :
this.activeFreqMHz;
    const prof = this.profiles?.[String(f)] || {};
    const snap = {
      freqMHz: f || null,
      devices: {
        rss60: prof.rss60 && Number.isFinite(prof.rss60.a) &&
Number.isFinite(prof.rss60.b) ? { a: +prof.rss60.a, b: +prof.rss60.b } :
null,
        rss40: prof.rss40 && Number.isFinite(prof.rss40.a) &&
Number.isFinite(prof.rss40.b) ? { a: +prof.rss40.a, b: +prof.rss40.b } :
null,
      }
    };
    return Object.freeze(snap);
  },
  // Convert voltage to dBm using provided snapshot
  toDbmWithSnapshot(snapshot, deviceKey, volt) {
    const d = snapshot?.devices?.[deviceKey];
    const v = Number(volt);
    if (!d || !Number.isFinite(d.a) || !Number.isFinite(d.b) ||
!Number.isFinite(v)) return NaN;
    const raw = d.a * v + d.b;
    return Math.round(raw * 100) / 100;
  },
};
```

Αρχείο plot.js

```
// core/plot.js - Plotting utilities for RSS data visualization & table
rendering
import { SERIES, S, SERIES_KEYS, YRANGE, Calibration } from './state.js';

// Layout generation based on units
export function makeLayout(def, {unit = 'volt'}={} ) {
  let chosenY;
  if (unit === 'dbm') {
    chosenY = {
      type: 'linear',
      autorange: false,
      range: [...YRANGE.dbm],
      title: 'Input Power (dBm)',
    };
  } else if (unit === 'volt') {
    chosenY = {
      type: 'linear',
      autorange: false,
      range: [...YRANGE.volt],
      title: def.yTitle || 'Voltage (V)',
    };
  }
```

```
  return {
    margin: { t: 60, l: 70, r: 40, b: 80 },
    plot_bgcolor: '#131927',
    paper_bgcolor: '#131927',
    font: { size: 12, family: 'Roboto, sans-serif', color: '#f0f0f0' },
    legend: { x: 1, y: 1, xanchor: 'right', yanchor: 'top', bgcolor:
'rgba(180,180,180,0.2)', font: { family: 'Roboto, sans-serif', size: 12 }
},
    showlegend: true,
    title: { text: def.title, font: { family: 'Roboto, sans-serif', size:
22 } },
    yaxis: { ...chosenY, showgrid: true, gridcolor: 'rgba(68,68,68,0.6)',
tickfont: { size: 12 }, titlefont: { size: 16 } },
    xaxis: { title: 'Time', type: 'date', tickformat: '%H:%M:%S',
hoverformat: '%H:%M:%S', showgrid: true, gridcolor: 'rgba(68,68,68,0.6)',
tickangle: -45, automargin: true, nticks: 20, tickfont: { size: 10 },
titlefont: { size: 16 } },
    uirevision: `fixed-${unit}-${def.plotId || 'plot'}`,
  };
}
// Main trace generator
export function mainTrace(def, state) {
  return {
    x: state.x, y: state.y, type: 'scatter', mode: 'lines',
    connectgaps: false,
    line: { width: 2, color: def.color, shape: 'linear' },
    name: def.legend || def.title,
    showlegend: true,
    fill: 'none'
  };
}
// Initialize plots for all series
export function initPlots() {
  Object.keys(SERIES).forEach(k => {
    const def = SERIES[k];
    const traces = [ mainTrace(def, S[k]) ];
    Plotly.newPlot(def.plotId, traces, makeLayout(def), { responsive: true,
displaylogo: false, doubleClick: 'reset' });
    ensureUnitResetHandler(def.plotId);
  });
}
// Get last timestamp from series
export function lastTs(state, fallbackDate=null) {
  return state.x.length ? state.x[state.x.length - 1] : fallbackDate;
}
export const GAP_MS = 2_000;
// Insert gap in data if time difference exceeds threshold
export function maybeBreak(plotId, state, ts, traceIndex = 0) {
  if (!state.x.length) return;
  const last = state.x[state.x.length - 1];
  if (ts - last > GAP_MS) {
    state.x.push(ts); state.y.push(null);
    Plotly.extendTraces(plotId, { x: [[ts]], y: [[null]] }, [traceIndex]);
  }
}
// Trim old data from series
export function trimOld(state, refTime, ms) {
  const cutoff = refTime.getTime() - ms;
  while (state.x.length && state.x[0].getTime() < cutoff) {
    state.x.shift(); state.y.shift();
  }
```

```
}
// Relayout plots to show data within specified time window
export function relayoutWindowAll(minutes, keys = SERIES_KEYS) {
  const targets = Array.isArray(keys) && keys.length ? keys : SERIES_KEYS;
  targets.forEach((k) => {
    const def = SERIES[k];
    const st = S[k];
    if (!def || !st || !Array.isArray(st.x) || !st.x.length) return;
    const end = lastTs(st);
    if (!end) return;
    const start = new Date(end.getTime() - minutes * 60_000);
    Plotly.relayout(def.plotId, { 'xaxis.range': [start, end] });
  });
}
// Redraw experiment plots for the provided channel keys using the current
unit.
export function drawExperimentArrays(keys = SERIES_KEYS) {
  const targets = Array.isArray(keys) && keys.length ? keys : SERIES_KEYS;
  const unit = Calibration.enabled ? 'dbm' : 'volt';
  targets.forEach((k) => {
    const def = SERIES[k];
    const st = S[k];
    if (!def || !st) return;
    const ys = unit === 'dbm' ? (st.yDbm || []) : (st.yVolt || []);
    const traces = [{
      x: st.x || [],
      y: ys,
      type: 'scatter',
      mode: 'markers',
      connectgaps: false,
      marker: { size: 4, color: def.color },
      name: 'RSS',
      fill: 'none'
    }];
    newPlotWithUnit(def, traces, unit);
  });
}
// Relayout experiment plots to the full data span, with optional streaming
clamp.
export function relayoutExperimentFullSpan(keys = SERIES_KEYS, {
isStreaming = false, latestDataTs = null } = {}) {
  const targets = Array.isArray(keys) && keys.length ? keys : SERIES_KEYS;
  const allX = targets.flatMap((k) => S[k]?.x || []);
  if (!allX.length) return;
  const start = allX.reduce((a, b) => (a < b ? a : b));
  const dataEnd = allX.reduce((a, b) => (a > b ? a : b));
  const shouldClamp = isStreaming && latestDataTs instanceof Date;
  const end = shouldClamp && dataEnd > latestDataTs ? latestDataTs :
dataEnd;
  targets.forEach((k) => {
    const def = SERIES[k];
    if (!def) return;
    Plotly.relayout(def.plotId, { 'xaxis.range': [start, end] });
  });
}
// Replot experiment traces when units change and refresh visible tables.
export function redrawExperimentOnUnitsChanged(keys = SERIES_KEYS) {
  const targets = Array.isArray(keys) && keys.length ? keys : SERIES_KEYS;
  drawExperimentArrays(targets);
  renderVisibleTables(targets);
}
```

```
// Parse numeric value or return NaN
export function numericOrNaN(v) {
  const n = parseFloat(v);
  return Number.isFinite(n) ? n : NaN;
}
// Combined traces for both series
export function combinedMainTraces(SERIES) {
  const t60 = {
    x: [], y: [], type:'scatter', mode:'lines',
    name: 'ZX47-60', line: { width: 2, color: SERIES.rss60.color },
    connectgaps: false
  };
  const t40 = {
    x: [], y: [], type:'scatter', mode:'lines',
    name: 'ZX47-40', line: { width: 2, color: SERIES.rss40.color },
    connectgaps: false
  };
  return [t60, t40];
}
// Render latest numeric readout & timestamp for a channel.
export function renderLatestReading(channelKey, rawValue, tsText,
toDisplayValue,
  { updateTimestampOnInvalid = true } = {}) {
  const id = SERIES[channelKey]?.liveEl;
  if (!id) return;
  const el = document.getElementById(id);
  if (!el) return;

  if (!Number.isFinite(rawValue)) {
    el.textContent = '--';
    if (!updateTimestampOnInvalid) return;
  } else {
    const displayValue = toDisplayValue(rawValue, channelKey);
    el.textContent = Number.isFinite(displayValue) ?
displayValue.toFixed(2) : '--';
  }
  const tsId = SERIES[channelKey]?.liveTsEl;
  if (!tsId) return;
  const tsEl = document.getElementById(tsId);
  if (tsEl) tsEl.textContent = tsText;
}
//-------- Table Helpers ---------------------------------
// Render table for a specific series
export function renderTable(kind, limit = 200) {
  const def = SERIES[kind];
  const body = document.getElementById(def.tableBody);
  if (!body) return;
  const xs = S[kind].x || [];
  const ys = Calibration?.enabled ? (S[kind].yDbm ?? S[kind].y ?? []) :
(S[kind].yVolt ?? S[kind].y ?? []);
  body.innerHTML = '';
  for (let i = xs.length - 1; i >= Math.max(0, xs.length - limit); i--) {
    const tr = document.createElement('tr');
    const yv = ys[i];
    tr.innerHTML =
`<td>${xs[i].toLocaleString()}</td><td>${def.fmt(yv)}</td>`;
    body.appendChild(tr);
  }
}
// Checks which tables are visible and renders them
export function renderVisibleTables(keys = SERIES_KEYS) {
```

```
  keys.forEach(k => {
    const block = document.getElementById(k + 'Block');
    if (block?.classList.contains('show-table')) renderTable(k);
  });
}
// Clear tables for specified series
export function clearAllTables(keys = SERIES_KEYS) {
  keys.forEach(k => {
    const body = document.getElementById(SERIES[k].tableBody);
    if (body) body.innerHTML = '';
  });
}
//------- Other Helpers ----------------------------
// Clear all data series
export function clearAllSeries() {
  SERIES_KEYS.forEach(k => {
    S[k].x = [];
    if (S[k].y) S[k].y = [];
    if (S[k].yVolt) S[k].yVolt = [];
    if (S[k].yDbm)  S[k].yDbm  = [];
    if (S[k].dist)  S[k].dist  = [];
    if (S[k].freq)  S[k].freq  = [];
    if (S[k].pl)    S[k].pl    = [];
  });
}
// Ensure reset Y-axis to correct unit
export function ensureUnitResetHandler(plotId){
  const el = document.getElementById(plotId);
  if (!el || el._unitResetHooked) return;
  el._unitResetHooked = true;
  el.on?.('plotly_doubleclick', () => {
    const unit = Calibration?.enabled ? 'dbm' : 'volt';
    const range = unit === 'dbm' ? [...YRANGE.dbm] : [...YRANGE.volt];
    Plotly.relayout(plotId, {
      'yaxis.autorange': false,
      'yaxis.range': range
    });
    return false;
  });
}
// Create new plot with specified unit
export function newPlotWithUnit(def, traces, unit) {
  const layout = makeLayout(def, { unit });
  if (typeof document !== 'undefined' &&
document.body?.classList?.contains('is-experiment')) {
    layout.title = layout.title || {};
    layout.title.text = buildExpTitle(def);
  }
  Plotly.purge(def.plotId);
  return Plotly.newPlot(
    def.plotId,
    traces,
    layout,
    { responsive: true, displaylogo: false, doubleClick: 'reset' }
  );
}
// Build experiment-specific title for plots
function buildExpTitle(def) {
  const exp = globalThis.currentExp;
  const id = (exp && exp.id) ? ` | ${exp.id}` : '';
  const expFreq = Number(exp?.freqMHz);
```

```
  const fallbackFreq = Number(globalThis.EXP_FREQ_MHz);
  const f = (Number.isFinite(expFreq) && expFreq > 0)
    ? expFreq
    : (Number.isFinite(fallbackFreq) && fallbackFreq > 0 ? fallbackFreq :
null);
  const freq = (f != null) ? ` - ${f} MHz` : '';
  return `${def.title}${id}${freq}`;
}
```

Αρχείο channel.js

```
// core/channels.js - Manages channel visibility state and persistence for
RSS data visualization

const STORAGE_KEY = 'rf_channels_enabled';
const CHANNEL_KEYS = Object.freeze(['rss60', 'rss40']);
const CHANNEL_LABELS = Object.freeze({
  rss60: 'Channel 1',
  rss40: 'Channel 2',
});
// Read stored channel visibility state from localStorage
function readStoredState() {
  try {
    const raw = localStorage.getItem(STORAGE_KEY);
    if (!raw) return null;
    const parsed = JSON.parse(raw);
    if (parsed && typeof parsed === 'object') {
      const state = {};
      CHANNEL_KEYS.forEach((key) => {
        state[key] = !!parsed[key];
      });
      return state;
    }
  } catch {}
  return null;
}

let state = readStoredState() || { rss60: true, rss40: true };
// Persist current channel visibility state to localStorage
function persist() {
  try {
    localStorage.setItem(STORAGE_KEY, JSON.stringify(state));
  } catch {}
}
// Get a copy of the current channel visibility state
export function getChannelState() {
  return { ...state };
}
// Check if a specific channel is enabled
export function isChannelEnabled(channelKey) {
  return !!state[channelKey];
}
// Get a list of currently enabled channels
export function getEnabledChannels() {
  return CHANNEL_KEYS.filter((key) => !!state[key]);
}
// Set channel visibility state and persist changes
export function setChannelEnabled(channelKey, enabled) {
```

```
  if (!CHANNEL_KEYS.includes(channelKey)) return false;
  const next = !!enabled;
  if (!next) {
    const othersEnabled = CHANNEL_KEYS
      .filter((key) => key !== channelKey)
      .some((key) => state[key]);
    if (!othersEnabled) {
      return false;
    }
  }
  if (state[channelKey] === next) {
    return true;
  }
  state = { ...state, [channelKey]: next };
  persist();
  const detail = { channel: channelKey, enabled: next, state:
getChannelState() };
  window.dispatchEvent(new CustomEvent('channel-visibility-changed', {
detail }));
  return true;
}

export { CHANNEL_KEYS, CHANNEL_LABELS };
```

Αρχείο status.js

```
// core/status.js - Status badge rendering & MQTT status handling

const STATUS_BAR_IDS = { 'esp32_60': 'statusBar1', 'esp32_40': 'statusBar2'
};

// Build a status badge element for a device
export function buildStatusBadge(devLabel, info) {
  let state = (info && info.state) ;
  if (!['online', 'offline'].includes(state)) state = 'unknown';
  const badge = document.createElement('div');
  badge.className = `status-badge status-${state}`;
  badge.title = `device: ${devLabel}\nstate: ${state}\n`;
  badge.innerHTML = `<span class="status-dot"></span><span class="status-
text">${devLabel}: ${state}</span>`;
  return badge;
}
// Map device IDs to status bar element IDs
export function renderStatusFor(deviceId, info, mapping = STATUS_BAR_IDS) {
  const barId = mapping[deviceId];
  const bar = document.getElementById(barId);
  if (!bar) return;
  bar.innerHTML = '';
  bar.appendChild(buildStatusBadge(deviceId, info));
}
// Attach MQTT status listener and render status badges accordingly
export function attachStatusFromMqtt(mapping = STATUS_BAR_IDS) {
  const onStatus = (ev) => {
    const { device, online } = ev.detail || {};
    if (!device) return;
    renderStatusFor(device, { state: online ? 'online' : 'offline' },
mapping);
  };
```

```
  window.addEventListener('mqtt:status', onStatus);
  // cleanup function to remove listener on detach
  return () => window.removeEventListener('mqtt:status', onStatus);
}
```

Αρχείο api.js

```
// core/api.js - API client functions for calibration profiles and device
settings

// Get all calibration profiles from the server
export async function calGetProfiles() {
  const r = await fetch('/calibration/profiles');
  if (!r.ok) throw new Error('Failed to load profiles');
  return r.json(); // { profiles: { "2400": {rss60:{a,b}, rss40:{a,b}}, ...
} }
}
// Save a calibration profile to the server
export async function calSaveProfile({ freqMHz, rss60, rss40 }) {
  const r = await fetch('/calibration/profiles', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ freqMHz, rss60, rss40 })
  });
  if (!r.ok) throw await r.json().catch(()=>({error:'save failed'}));
  return r.json(); // { ok:true, profile: { "2400": { ... } } }
}
// Delete a calibration profile from the server by frequency
export async function calDeleteProfile(freqMHz) {
  const r = await
fetch(`/calibration/profiles/${encodeURIComponent(freqMHz)}`, { method:
'DELETE' });
  if (!r.ok) throw await r.json().catch(()=>({error:'delete failed'}));
  return r.json(); // { ok:true }
}
// get sample rates for all devices from the server
export async function getSampleRates() {
  const r = await fetch('/devices/sample-rates');
  if (!r.ok) throw new Error('Failed to load sample rates');
  return r.json();
}
// set the sample rate for a specific device on the server
export async function setSampleRate(deviceId, rateHz) {
  const r = await fetch(`/devices/${encodeURIComponent(deviceId)}/sample-
rate`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ rateHz })
  });
  if (!r.ok) throw await r.json().catch(() => ({ error: 'Failed to update
sample rate' }));
  return r.json();
}
// list available experiment files/records
export async function expList() {
  const r = await fetch('/experiments');
  if (!r.ok) throw new Error('Failed to load experiments');
  return r.json();
```

```
}
// upload CSV to import endpoint; caller handles status (e.g. 409 overwrite
flow)
export async function expImportCsv(formData, { overwrite = false } = {}) {
  return fetch(`/api/experiments/import?overwrite=${overwrite ? '1' :
'0'}`, {
    method: 'POST',
    body: formData
  });
}
// get raw CSV content for a specific experiment id
export async function expGetCsvText(expId) {
  const r = await fetch(`/experiment-
data?exp_id=${encodeURIComponent(expId)}`);
  if (!r.ok) throw new Error('csv not found');
  return r.text();
}
// parse experiment CSV text to structured per-device arrays
function parseExperimentCsvText(text) {
  const lines = text.trim().split(/\r?\n/);
  let i = 0;
  while (i < lines.length && lines[i].startsWith('#')) i++;
  const header = (lines[i++] || '').split(',').map((s) =>
s.trim().toLowerCase());
  const idx = {
    ts: header.indexOf('timestamp'),
    dev: header.indexOf('device'),
    volt: header.indexOf('volt'),
    dbm: header.indexOf('dbm'),
    fmhz: header.indexOf('freq_mhz'),
    dist: header.indexOf('distance_m'),
    pl: header.indexOf('path_loss_db'),
  };

  const out = {
    freqMHz: null,
    byDevice: {
      rss60: { x: [], vVolt: [], vDbm: [], vDist: [], vPL: [], vFreq: [] },
      rss40: { x: [], vVolt: [], vDbm: [], vDist: [], vPL: [], vFreq: [] },
    }
  };

  for (; i < lines.length; i++) {
    const row = lines[i];
    if (!row || row.startsWith('#')) continue;
    const cols = row.split(',');
    const devRaw = (cols[idx.dev] || '').trim();
    const dev = devRaw === 'esp32_60' ? 'rss60' : (devRaw === 'esp32_40' ?
'rss40' : null);
    if (!dev) continue;

    const t = Date.parse(cols[idx.ts] || '');
    if (!Number.isFinite(t)) continue;
    out.byDevice[dev].x.push(t);
    out.byDevice[dev].vVolt.push(idx.volt >= 0 &&
Number.isFinite(+cols[idx.volt]) ? +cols[idx.volt] : NaN);
    out.byDevice[dev].vDbm.push(idx.dbm >= 0 &&
Number.isFinite(+cols[idx.dbm]) ? +cols[idx.dbm] : NaN);
    out.byDevice[dev].vDist.push(idx.dist >= 0 &&
Number.isFinite(+cols[idx.dist]) ? +cols[idx.dist] : NaN);
```

```
    const fm = idx.fmhz >= 0 && Number.isFinite(+cols[idx.fmhz]) ?
+cols[idx.fmhz] : NaN;
    out.byDevice[dev].vFreq.push(Number.isFinite(fm) ? fm : NaN);
    if (Number.isFinite(fm) && out.freqMHz == null) out.freqMHz = fm;
    out.byDevice[dev].vPL.push(idx.pl >= 0 &&
Number.isFinite(+cols[idx.pl]) ? +cols[idx.pl] : NaN);
  }
  return out;
}
// fetch and parse experiment CSV by id
export async function expLoadParsed(expId) {
  const text = await expGetCsvText(expId);
  return parseExperimentCsvText(text);
}
// upload finalized experiment CSV for a given id
export async function expUploadCsv(expId, formData) {
  return fetch(`/api/experiments/${encodeURIComponent(expId)}/upload`, {
    method: 'POST',
    body: formData
  });
}
// delete experiment by id
export async function expDelete(expId) {
  return fetch(`/experiment/${encodeURIComponent(expId)}`, {
    method: 'DELETE'
  });
}
```

Αρχείο analysis.js

```
// analysis.js - Path Loss Analysis Module
// Handles computation and plotting of path-loss models and statistics

import {S, SERIES} from './state.js'
import { mean, median, stddev, computeBestFit, generateLineXs,
         computeLogDistanceModel, computeLogDistanceShadowingBand,
         fitLogDistanceSlope, } from './statistics.js';

const state = {
  buttons: [],
  isExperimentMode: false,
  isAvailable: false,
  activeChannel: null,
  initialized: false,
  fsplEnabled: false,
  bestCurveEnabled: false,
  logDistance: {
    enabled: false,
    panelOpen: false,
    params: { d0: null, manualPl0: null, n: null, useFirstPoint: false,
              useFsplReference: false, fitN: false, shadowingSigma: 0 },
  },
  analysisLightTheme: false,
};
// Some helper functions to access DOM elements and data
function getModal() { return document.getElementById('analysisModal'); }
function getFsplButton() { return document.getElementById('fsplToggle'); }
```

```
function getBestCurveButton() { return
document.getElementById('bestCurveToggle'); }
function getLogDistanceButton() { return
document.getElementById('logDistanceToggle'); }
function getLogDistancePanel() { return
document.getElementById('logDistancePanel'); }
function getFsplSummaryEl() { return
document.getElementById('fsplSummary'); }
function getOlsSummaryEl() { return document.getElementById('olsSummary');
}
function getLogDistanceSummaryEl() { return
document.getElementById('logDistanceSummary'); }
function getTitleInput() { return document.getElementById('plTitleInput');
}
function getThemeToggle() { return
document.getElementById('analysisLightThemeToggle'); }
function formatNumber(value, digits = 2) {
  return Number.isFinite(value) ? value.toFixed(digits) : '-';
}
function getLogDistanceInputs() {
  return {
    d0: document.getElementById('logDistanceD0'),
    pl0: document.getElementById('logDistancePl0'),
    n: document.getElementById('logDistanceExponent'),
    useFirst: document.getElementById('logDistanceUseFirst'),
    useFspl: document.getElementById('logDistanceUseFsplRef'),
    fitN: document.getElementById('logDistanceFitExponent'),
    sigma: document.getElementById('logDistanceSigma'),
  };
}
function getAxisValues(series) {
  if (!series) return [];
  return series?.dist || [];
}
function isValidAxisValue(val) {
  return Number.isFinite(val) && val > 0;
}
function getChartElements() {
  return [
    document.getElementById('plChart_rss60'),
    document.getElementById('plChart_rss40'),
  ].filter(Boolean);
}
function getExperimentFreqMHz() {
  const raw = window.currentExp?.freqMHz ?? window.EXP_FREQ_MHz ?? '';
  const num = Number(raw);
  return Number.isFinite(num) && num > 0 ? num : null;
}
function buildDefaultTitle() {
  const freqVal = getExperimentFreqMHz();
  return `Path Loss vs. Distance at ${formatNumber(freqVal, 0)} MHz`;
}
function getChannelColor(channelKey) {
  const fallback = '#50a5faff';
  if (!channelKey) return fallback;
  const raw = SERIES?.[channelKey]?.color;
  if (typeof raw !== 'string') return fallback;
  const color = raw.trim();
  return color || fallback;
}
```

```
// Update summary sections
function updateFsplSummary(data) {
  const el = getFsplSummaryEl();
  if (!el) return;
  if (!data) {
    el.classList.add('hidden');
    el.innerHTML = '';
    return;
  }
  const { slope, intercept, freqMHz } = data;
  el.innerHTML = `
    <div class="analysis-result-title">FSPL CURVE</div>
    <dl>
      <dt>Slope</dt><dd>${formatNumber(slope, 2)} dB/decade</dd>
      <dt>Intercept</dt><dd>${formatNumber(intercept, 2)} dB</dd>
      <dt>Frequency</dt><dd>${formatNumber(freqMHz, 0)} MHz</dd>
    </dl>
  `;
  el.classList.remove('hidden');
}
function updateLogDistanceSummary(data) {
  const el = getLogDistanceSummaryEl();
  if (!el) return;
  if (!data) {
    el.classList.add('hidden');
    el.innerHTML = '';
    return;
  }
  const title = '<div class="analysis-result-title">Log-distance
model</div>';
  const { d0, pl0, n, datasetSize, rmse, r2, residualMean, residualSigma,
shadowingSigma } = data;
  const sigmaMarkup = (Number.isFinite(shadowingSigma) && shadowingSigma >
0) //The 'σ' for shadowing band
    ? `<dt>&sigma; shadowing</dt><dd>${formatNumber(shadowingSigma)}
dB</dd>`
    : '';
  el.innerHTML = `
    ${title}
    <dl>
      <dt>d<sub>0</sub></dt><dd>${formatNumber(d0)} m</dd>
      <dt>&lt;PL&gt;(d<sub>0</sub>)</dt><dd>${formatNumber(pl0)} dB</dd>
      <dt>n exponent</dt><dd>${formatNumber(n)}</dd>
      <dt>RMSE</dt><dd>${formatNumber(rmse, 3)} dB</dd>
      <dt>R<sup>2</sup></dt><dd>${formatNumber(r2, 4)}</dd>
      <dt>Mean (residual)</dt><dd>${formatNumber(residualMean, 3)} dB</dd>
      <dt>Std \u03C3</dt><dd>${formatNumber(residualSigma, 3)} dB</dd>
      ${sigmaMarkup}
      <dt>Samples</dt><dd>${datasetSize ?? '-'}</dd>
    </dl>
  `;
  el.classList.remove('hidden');
}
function updateOlsSummary(data, channelKey) {
  const el = getOlsSummaryEl();
  if (!el) return;
  if (!data) {
    el.classList.add('hidden');
    el.innerHTML = '';
    el.style.removeProperty('--analysis-ols-color');
    el.style.removeProperty('--analysis-ols-bg');
```

```
    el.style.removeProperty('border-color');
    el.style.removeProperty('background-color');
    return;
  }
  const color = getChannelColor(channelKey);
  el.style.setProperty('--analysis-ols-bg', color);
  const { slope, intercept, rmse, r2, count } = data;
  el.innerHTML = `
    <div class="analysis-result-title">OLS FIT</div>
    <dl>
      <dt>Slope</dt><dd>${formatNumber(slope, 2)} dB/decade</dd>
      <dt>Intercept</dt><dd>${formatNumber(intercept, 2)} dB</dd>
      <dt>RMSE</dt><dd>${formatNumber(rmse, 3)} dB</dd>
      <dt>R<sup>2</sup></dt><dd>${formatNumber(r2, 4)}</dd>
      <dt>Samples</dt><dd>${count ?? '-'}</dd>
    </dl>
  `;
  el.classList.remove('hidden');
}
// Handle log-distance model fitting
function handleLogDistanceFit() {
  const inputs = getLogDistanceInputs();
  const nVal = parseFloat(inputs?.n?.value);
  const useFsplReference = !!inputs?.useFspl?.checked;
  const useFirstPoint = useFsplReference ? false :
!!inputs?.useFirst?.checked;
  const fitExponent = !!inputs?.fitN?.checked;
  const rawSigma = parseFloat(inputs?.sigma?.value);
  if (Number.isFinite(rawSigma) && rawSigma < 0) {
    alert('Enter a non-negative sigma value for the shadowing effect.');
    return;
  }
  const sigmaVal = Number.isFinite(rawSigma) ? rawSigma : 0;
  const rawD0 = parseFloat(inputs?.d0?.value);
  const manualD0 = Number.isFinite(rawD0) ? rawD0 : null;
  const rawPl0 = parseFloat(inputs?.pl0?.value);
  const manualPl0 = Number.isFinite(rawPl0) ? rawPl0 : null;
  if (!useFirstPoint && !useFsplReference && (!Number.isFinite(manualD0) ||
manualD0 <= 0)) {
    alert('Enter a valid reference distance d0 greater than 0.');
    return;
  }
  if (!useFirstPoint && !useFsplReference && !Number.isFinite(manualPl0)) {
    alert('Enter a valid PL(d0) value or enable "Use first PL mean
point".');
    return;
  }
  if (useFsplReference && !Number.isFinite(getExperimentFreqMHz())) {
    alert('A valid experiment frequency is required to use the 1 m FSPL
reference.');
    return;
  }
  if (!fitExponent && !Number.isFinite(nVal)) {
    alert('Enter a valid path-loss exponent n or enable fitting.');
    return;
  }
  state.logDistance.enabled = true;
  state.logDistance.params = {
    d0: useFsplReference ? 1 : manualD0,
    manualPl0: useFsplReference ? null : manualPl0,
    n: fitExponent ? null : nVal,
```

```
    useFirstPoint,
    useFsplReference,
    fitN: fitExponent,
    shadowingSigma: sigmaVal,
  };
  updateLogDistanceButton();
  const channel = state.activeChannel;
  if (!channel) return;
  const opts = getAnalysisOpts();
  computeAndPlotForChannel(channel, opts);
}
// Clear summaries and charts
function clearSummaries() {
  updateFsplSummary(null);
  updateOlsSummary(null, null);
  updateLogDistanceSummary(null);
}
function clearCharts() {
  const charts = getChartElements();
  charts.forEach((el) => {
    if (window.Plotly?.purge) {
      try { window.Plotly.purge(el); } catch (_) {}
    }
    el.innerHTML = '';
    el.classList.add('hidden');
  });
}
// Update button states
function updateFsplButton() {
  const btn = getFsplButton();
  if (!btn) return;
  btn.disabled = false;
  btn.classList.remove('hidden');
  btn.setAttribute('aria-pressed', state.fsplEnabled ? 'true' : 'false');
  btn.classList.toggle('fspl-active', !!state.fsplEnabled);
}
function updateBestCurveButton() {
  const btn = getBestCurveButton();
  if (!btn) return;
  btn.setAttribute('aria-pressed', state.bestCurveEnabled ? 'true' :
'false');
  btn.classList.toggle('bestcurve-active', !!state.bestCurveEnabled);
  if (state.bestCurveEnabled) {
    const color = getChannelColor(state.activeChannel);
    btn.style.setProperty('--analysis-bestcurve-color', color);
  } else {
    btn.style.removeProperty('--analysis-bestcurve-color');
  }
}
function updateLogDistanceButton() {
  const btn = getLogDistanceButton();
  if (!btn) return;
  const enabled = !!state.logDistance?.enabled;
  btn.classList.toggle('logdistance-active', enabled);
  const expanded = !!state.logDistance?.panelOpen;
  btn.setAttribute('aria-expanded', expanded ? 'true' : 'false');
}
function toggleLogDistancePanel(force) {
  const panel = getLogDistancePanel();
  const btn = getLogDistanceButton();
  if (!panel || !btn) return;
```

```
  const desired = (typeof force === 'boolean') ? force :
!state.logDistance.panelOpen;
  panel.classList.toggle('hidden', !desired);
  panel.setAttribute('aria-hidden', desired ? 'false' : 'true');
  state.logDistance.panelOpen = desired;
  btn.setAttribute('aria-expanded', desired ? 'true' : 'false');
  updateLogDistanceButton();
}
// Enable/disable inputs based on current selections
function syncLogDistanceInputs() {
  const { pl0, d0, useFirst, useFspl, n, fitN } = getLogDistanceInputs();
  if (!pl0 || !useFirst) return;
  const useFirstChecked = !!useFirst.checked;
  const useFsplChecked = !!useFspl?.checked;
  const disablePlInputs = useFirstChecked || useFsplChecked;
  pl0.disabled = disablePlInputs;
  if (d0) {
    d0.disabled = disablePlInputs;
    if (disablePlInputs) {
      d0.setAttribute('aria-disabled', 'true');
    } else {
      d0.removeAttribute('aria-disabled');
    }
  }
  if (disablePlInputs) {
    pl0.setAttribute('aria-disabled', 'true');
  } else {
    pl0.removeAttribute('aria-disabled');
  }
  const fitChecked = !!fitN?.checked;
  if (n) {
    n.disabled = fitChecked;
    if (fitChecked) {
      n.setAttribute('aria-disabled', 'true');
    } else {
      n.removeAttribute('aria-disabled');
    }
  }
}
function clearLogDistance() {
  state.logDistance.enabled = false;
  state.logDistance.params = { d0: null, manualPl0: null, n: null,
useFirstPoint: false, useFsplReference: false, fitN: false, shadowingSigma:
0 };
  updateLogDistanceSummary(null);
  updateLogDistanceButton();
}
// Compute first mean point from distance-PL data
function computeFirstPointFromPairs(pairs) {
  if (!Array.isArray(pairs) || pairs.length === 0) return null;
  let minDist = Infinity;
  for (const p of pairs) {
    const d = p?.distance;
    if (Number.isFinite(d) && d > 0 && d < minDist) {
      minDist = d;
    }
  }
  if (!Number.isFinite(minDist) || minDist === Infinity) return null;
  const tolerance = Math.max(1e-9, Math.abs(minDist) * 1e-6);
  const values = [];
  for (const p of pairs) {
```

```
    const d = p?.distance;
    const v = p?.pl;
    if (!Number.isFinite(d) || !Number.isFinite(v)) continue;
    if (Math.abs(d - minDist) <= tolerance) {
      values.push(v);
    }
  }
  if (!values.length) return null;
  return { distance: minDist, meanPl: mean(values) };
}
// Fit log-distance exponent n
function fitLogDistanceExponent(viewXs, viewYs, d0, pl0) {
  if (!Number.isFinite(d0) || d0 <= 0 || !Number.isFinite(pl0)) return
null;
  if (!Array.isArray(viewXs) || !Array.isArray(viewYs)) return null;
  const fit = fitLogDistanceSlope(viewXs, viewYs, d0, pl0);
  return (fit && Number.isFinite(fit.n)) ? fit : null;
}
// Ensure buttons are cached
function ensureButtons() {
  if (!state.buttons.length) {
    state.buttons = Array.from(document.querySelectorAll('.analyze-btn'));
  }
  return state.buttons;
}

// Toggle placeholder visibility based on active channel
function togglePlaceholders(activeKey) {
  const map = {
    rss60: document.getElementById('plChart_rss60'),
    rss40: document.getElementById('plChart_rss40'),
  };
  Object.entries(map).forEach(([key, el]) => {
    if (!el) return;
    if (activeKey && key === activeKey) {
      el.classList.remove('hidden');
    } else {
      el.classList.add('hidden');
    }
  });
}
// Main computation and plotting function
function computeAndPlotForChannel(k, opts) {
  const chartId = (k === 'rss60') ? 'plChart_rss60'
                : (k === 'rss40') ? 'plChart_rss40'
                : null;
  if (!chartId) return;

  const st = S?.[k];
  const axisValuesRaw = getAxisValues(st);
  const dist = st?.dist || [];
  const pl   = st?.pl   || [];
  const channelColor = getChannelColor(k);
  const xs = [], ys = [];
  const pairs = [];
  // Collect valid distance-PL pairs
  for (let i = 0; i < Math.min(axisValuesRaw.length, pl.length); i++) {
    const axisVal = axisValuesRaw[i];
    const p = pl[i];
    if (!Number.isFinite(p) || !isValidAxisValue(axisVal)) continue;
    xs.push(axisVal);
```

```
    ys.push(p);
    pairs.push({ axis: axisVal, pl: p, distance: dist[i] });
  }
  const el = document.getElementById(chartId);
  if (!el) return;

  if (xs.length === 0) {
    if (window.Plotly?.purge) Plotly.purge(el);

    el.innerHTML = `<div class="muted" style="padding:12px;text-
align:center;">
      No path-loss data available for this channel (need distance & PL per
sample).
    </div>`;
    if (state.logDistance?.enabled) {
      updateLogDistanceSummary({ error: 'Not enough data to compute the
log-distance model.' });
    } else {
      updateLogDistanceSummary(null);
    }
    return;
  }

  let traces;
  let fsplSourceXs = [];
  let viewXs = [];
  let viewYs = [];
  let logDistanceSummary = null;
  let logDistanceError = null;
  const baseHoverTemplate = `${`Distance`}: %{x:.2f}${` m`}<br>PL: %{y:.2f}
dB<extra></extra>`;
  const isLight = !!opts.lightTheme;
  const gridColor = isLight ? 'rgba(0,0,0,0.15)' : 'rgba(68,68,68,0.6)';
  const fontColor = isLight ? '#111111' : '#f0f0f0';
  const plotBg = isLight ? '#ffffff' : '#1b2130';
  const legendBg = isLight ? '#ffffff' : '#3E4865';
  const legendBorder = isLight ? '#cfd6e3' : '#4F5D82';

  if (opts.view === 'binned') {
    const buckets = new Map();
    for (let i=0;i<xs.length;i++){
      const decimals = 3;
      const key = Number(xs[i].toFixed(decimals));
      if (!buckets.has(key)) buckets.set(key, []);
      buckets.get(key).push(ys[i]);
    }
    const bx = [], by = [];
    for (const [d, vals] of [...buckets.entries()].sort((a,b)=>a[0]-b[0]))
{
      if (vals.length === 0) continue;
      by.push(opts.stat === 'median' ? median(vals) : mean(vals));
      bx.push(d);
    }
    fsplSourceXs = bx;
    viewXs = bx;
    viewYs = by;
    const trace = {
      x: bx,
      y: by,
      type: 'scatter',
      mode: 'markers',
```

```
      marker: { size: 12, color: channelColor },
      name: (k==='rss60'?'Ch1 data':'Ch2 data') + ' (binned)',
      hovertemplate: baseHoverTemplate,
    };
    traces = [trace];
  } else {
    fsplSourceXs = xs;
    viewXs = xs;
    viewYs = ys;
    traces = [{
      x: xs,
      y: ys,
      type: 'scatter',
      mode: 'markers',
      marker: { size: 4, opacity: 0.8, color: channelColor },
      name: (k==='rss60'?'Ch1 data':'Ch2 data'),
      hovertemplate: baseHoverTemplate,
    }];
  }
  const datasetLen = Math.min(viewXs.length, viewYs.length);
  const freqVal = getExperimentFreqMHz();

  // Handle log-distance model
  if (state.logDistance?.enabled) {
    const params = state.logDistance.params || {};
    const fitExponent = !!params?.fitN;
    let nVal = fitExponent ? null : params?.n;
    const useFsplReference = !!params?.useFsplReference;
    const useFirstPoint = useFsplReference ? false :
!!params?.useFirstPoint;
    const manualD0 = params?.d0;
    let pl0Val = params?.manualPl0;
    const shadowingSigma = Number.isFinite(params?.shadowingSigma) ?
Math.max(0, params.shadowingSigma) : 0;
    let firstPointInfo = null;
    let d0Val = useFsplReference ? 1 : manualD0;
    if (useFsplReference && !Number.isFinite(freqVal)) {
      logDistanceError = 'A valid frequency is required to compute the FSPL
reference at 1 m.';
    } else if (useFsplReference) {
      pl0Val = 32.44 + 20 * Math.log10(freqVal) - 60;
    } else if (useFirstPoint) {
      firstPointInfo = computeFirstPointFromPairs(pairs);
      pl0Val = firstPointInfo?.meanPl ?? null;
      d0Val = firstPointInfo?.distance ?? null;
    }
    if (!Number.isFinite(d0Val) || d0Val <= 0) {
      logDistanceError = useFirstPoint
        ? 'First mean point not available to estimate reference distance.'
        : useFsplReference
        ? 'Reference distance is fixed at 1 m when using FSPL.'
        : 'Enter a valid reference distance d0 greater than 0.';
    } else if (!Number.isFinite(pl0Val)) {
      logDistanceError = useFirstPoint
        ? 'First mean point not available to estimate PL(d0).'
        : useFsplReference
        ? 'A valid frequency is required to compute FSPL at 1 m.'
        : 'Enter a valid PL(d0) value.';
    } else {
      let fitInfo = null;
      if (fitExponent) {
```

```
        fitInfo = fitLogDistanceExponent(viewXs, viewYs, d0Val, pl0Val);
        if (fitInfo && Number.isFinite(fitInfo.n)) {
          nVal = fitInfo.n;
        } else {
          logDistanceError = 'Unable to fit the path-loss exponent n from
the current data.';
        }
      }
      if (!fitExponent && !Number.isFinite(nVal)) {
        logDistanceError = 'Enter a valid path-loss exponent n.';
      } else if (Number.isFinite(nVal)) {
        let lineXs = generateLineXs(viewXs, opts.xscale);
        if (!lineXs.length) {
          lineXs = generateLineXs(xs, opts.xscale);
        }
        if (!lineXs.length) {
          lineXs = [...new Set(pairs.map(p => p.distance).filter(v =>
Number.isFinite(v) && v > 0))].sort((a, b) => a - b);
        }
        const model = computeLogDistanceModel(lineXs, { d0: d0Val, pl0:
pl0Val, n: nVal });
        if (model && Array.isArray(model.xs) && Array.isArray(model.ys) &&
model.xs.length) {
          const ldXs = model.xs;
          const ldYs = model.ys;
          const logColor = 'rgb(185,50,50)';
          const shadowingBand = computeLogDistanceShadowingBand(ldYs,
shadowingSigma);
          if (shadowingBand) {
            const bandColor = 'rgba(185,50,50,0.2)';
            traces.push({
              x: ldXs,
              y: shadowingBand.lower,
              type: 'scatter',
              mode: 'lines',
              line: { width: 0, color: 'rgba(0,0,0,0)' },
              hoverinfo: 'skip',
              showlegend: false,
              legendgroup: 'logdistance',
            });
            traces.push({
              x: ldXs,
              y: shadowingBand.upper,
              type: 'scatter',
              mode: 'lines',
              line: { width: 0, color: 'rgba(0,0,0,0)' },
              hoverinfo: 'skip',
              fill: 'tonexty',
              fillcolor: bandColor,
              name: '\u00B1\u03C3 shadowing',
              legendgroup: 'logdistance',
            });
          }
          traces.push({
            x: ldXs,
            y: ldYs,
            type: 'scatter',
            mode: 'lines',
            line: { width: 4, color: logColor },
            name: 'Log-distance model',
            hovertemplate: baseHoverTemplate,
```

```
            legendgroup: 'logdistance',
          });
          const metricPoints = [];
          for (let i = 0; i < Math.min(viewXs.length, viewYs.length); i++)
{
            const dx = viewXs[i];
            const dy = viewYs[i];
            if (!Number.isFinite(dx) || dx <= 0 || !Number.isFinite(dy))
continue;
            const pred = pl0Val + 10 * nVal * Math.log10(dx / d0Val);
            if (!Number.isFinite(pred)) continue;
            metricPoints.push({ actual: dy, predicted: pred });
          }
          let rmse = null;
          let r2 = null;
          let residualMean = null;
          let residualSigma = null;
          if (metricPoints.length) {
            const residuals = [];
            const meanY = metricPoints.reduce((acc, p) => acc + p.actual,
0) / metricPoints.length;
            let sse = 0;
            let sst = 0;
            for (const p of metricPoints) {
              const err = p.actual - p.predicted;
              residuals.push(err);
              sse += err * err;
              const dy = p.actual - meanY;
              sst += dy * dy;
            }
            rmse = Math.sqrt(sse / metricPoints.length);
            r2 = (sst === 0) ? 1 : 1 - (sse / sst);
            residualMean = mean(residuals);
            residualSigma = stddev(residuals);
          }
          logDistanceSummary = {
            d0: d0Val,
            pl0: pl0Val,
            n: nVal,
            useFirstPoint,
            datasetSize: metricPoints.length || datasetLen || pairs.length
|| null,
            firstPointDistance: firstPointInfo?.distance ?? null,
            rmse,
            r2,
            residualMean,
            residualSigma,
            fitExponent,
            useFsplReference,
            shadowingSigma,
          };
        } else {
          logDistanceError = 'Unable to generate the log-distance model
with the current inputs.';
        }
      }
    }
  }
  // Handle FSPL curve
  let fsplSummary = null;
  let fsplLine = null;
```

```
  if (state.fsplEnabled) {
    if (Number.isFinite(freqVal)) {
      const uniqXs = [...new
Set(fsplSourceXs.filter(isValidAxisValue))].sort((a, b) => a - b);
      if (uniqXs.length) {
        const fsplYs = uniqXs.map((d) => 32.44 + 20 * Math.log10(freqVal) +
20 * Math.log10(d / 1000));
        fsplLine = { xs: uniqXs, ys: fsplYs };
        traces.push({
          x: uniqXs,
          y: fsplYs,
          type: 'scatter',
          mode: 'lines',
          line: { width: 4, color: '#ffb347' },
          name: 'FSPL',
          hovertemplate: `${`Distance`}: %{x:.2f}${` m`}<br>FSPL: %{y:.2f}
dB<extra></extra>`,
        });
        fsplSummary = {
          slope: 20,
          intercept: 32.44 + 20 * Math.log10(freqVal) - 60,
          freqMHz: freqVal,
        };
      }
    }
  }
  // Handle OLS fit
  let regression = null;
  if (state.bestCurveEnabled) {
    regression = computeBestFit(viewXs, viewYs);
    if (regression) {
      const lineXs = generateLineXs(viewXs, opts.xscale);
      if (lineXs.length) {
        const lineYs = lineXs.map(d => regression.intercept +
regression.slope * Math.log10(d));
        traces.push({
          x: lineXs,
          y: lineYs,
          type: 'scatter',
          mode: 'lines',
          line: { width: 4, color: (isLight ? '#111111' : '#ffffff'), dash:
"dash" },
          name: 'OLS Fit',
          hoverinfo: 'skip',
          legendgroup: 'ols',
        });
      }
    }
  }
  // Finalize summaries and layout
  if (state.logDistance?.enabled) {
    const summaryData = logDistanceSummary
      || (logDistanceError ? { error: logDistanceError } : { error: 'Unable
to compute the log-distance model.' });
    updateLogDistanceSummary(summaryData);
  } else {
    updateLogDistanceSummary(null);
  }
  updateFsplSummary(state.fsplEnabled ? fsplSummary : null);
  updateOlsSummary(state.bestCurveEnabled ? regression : null, k);
  const customTitle = (opts.title || '').trim();
```

```
  const layoutTitle = customTitle || buildDefaultTitle();
  const layout = {
    title: {text: layoutTitle, font: { size: 22 }},
    xaxis: {
       title: 'Distance (m)',
       type: (opts.xscale === 'linear' ? 'linear' : 'log'),
       autorange: true,
       gridcolor: gridColor,
       tickfont: { size: 15 },
       titlefont: { size: 20 },
    },
    yaxis: { title: 'Path Loss (dB)', gridcolor: gridColor, tickfont: {
size: 15 }, titlefont: { size: 20 } },
    margin: { l: 70, r: 60, t: 60, b: 80 },
    autosize: true,
    plot_bgcolor: plotBg,
    paper_bgcolor: plotBg,
    font: { family: "Roboto, sans-serif", color: fontColor },
    showlegend: true, legend: { x:1, xanchor: 'right', y:0, bgcolor:
legendBg, bordercolor: legendBorder, borderwidth: 1,
                                font: { size: 14, color: fontColor } },
  };
  Plotly.newPlot(el, traces, layout, { responsive: true, displaylogo: false
});
}
function updateButtons() {
  const buttons = ensureButtons();
  const show = state.isExperimentMode && state.isAvailable;
  buttons.forEach(btn => {
    if (!btn) return;
    btn.classList.toggle('hidden', !show);
    btn.disabled = !show;
  });
}
// Modal open/close functions
function closeModal() {
  const modal = getModal();
  if (!modal) return;
  modal.classList.add('hidden');
  modal.setAttribute('aria-hidden', 'true');
  modal.removeAttribute('data-channel');
  state.activeChannel = null;
  togglePlaceholders(null);
  if (state.logDistance?.panelOpen) {
    toggleLogDistancePanel(false);
  }
  clearSummaries();
}
function resetForNewExperiment() {
  state.fsplEnabled = false;
  state.bestCurveEnabled = false;
  clearLogDistance();
  closeModal();
  clearCharts();
  updateFsplButton();
  updateBestCurveButton();
  updateLogDistanceButton();
}
function openModal(channelKey) {
  if (!state.isExperimentMode || !state.isAvailable) return;
  const modal = getModal();
```

```
  if (!modal) return;

  state.activeChannel = channelKey || null;
  modal.dataset.channel = channelKey || '';
  // Update title and inputs
  const titleEl = document.getElementById('analysisModalTitle');
  if (titleEl) {
    const expId = window.currentExp?.id ? String(window.currentExp.id) : '-
';
    titleEl.textContent = `Path-loss analysis for experiment: ${expId}`;
  }
  const titleInput = getTitleInput();
  if (titleInput) {
    const defaultTitle = buildDefaultTitle();
    titleInput.placeholder = defaultTitle;
    titleInput.value = defaultTitle;
  }
  const themeToggle = getThemeToggle();
  if (themeToggle) themeToggle.checked = state.analysisLightTheme;
  togglePlaceholders(channelKey);
  updateFsplButton();
  updateBestCurveButton();
  updateLogDistanceButton();
  syncLogDistanceInputs();
  clearSummaries();
  modal.classList.remove('hidden');
  modal.setAttribute('aria-hidden', 'false');
  const opts = getAnalysisOpts();
  computeAndPlotForChannel(channelKey, opts);
}
// Event handlers
function handleButtonClick(e) {
  const btn = e.currentTarget;
  if (!btn) return;
  const channel = btn.dataset.channel || null;
  openModal(channel);
}
function handleBackdropClick(e) {
  if (e.target && (e.target === e.currentTarget ||
e.target.classList?.contains('modal-backdrop'))) {
    closeModal();
  }
}
function handleKeydown(e) {
  if (e.key !== 'Escape') return;
  const modal = getModal();
  if (modal && !modal.classList.contains('hidden')) {
    closeModal();
  }
}
// Retrieve current analysis options from inputs
function getAnalysisOpts() {
  const view   = document.getElementById('plView')?.value || 'raw';
  const stat   = document.getElementById('plStat')?.value || 'mean';
  const xscale = document.getElementById('plXscale')?.value || 'linear';
  const titleInput = getTitleInput();
  const title = (titleInput?.value ?? '').trim();
  const lightTheme = !!getThemeToggle()?.checked;
  return { view, stat, xscale, title, lightTheme };
}
// One-time setup for event listeners & UI state
```

```
function init() {
  if (state.initialized) return;
  ensureButtons().forEach(btn => btn.addEventListener('click',
handleButtonClick));

  const modal = getModal();
  modal?.querySelector('.modal-backdrop')?.addEventListener('click',
handleBackdropClick);

  const closeBtn = document.getElementById('analysisCloseBtn');
  closeBtn?.addEventListener('click', closeModal);

  const computeBtn = document.getElementById('analysisComputeBtn');
  computeBtn?.addEventListener('click', () => {
    const k = state.activeChannel;
    if (!k) return;
    const opts = getAnalysisOpts();
    if (typeof togglePlaceholders === 'function') togglePlaceholders(k);
    computeAndPlotForChannel(k, opts);
  });
  const titleInput = getTitleInput();
  titleInput?.addEventListener('change', () => {
    const k = state.activeChannel;
    if (!k) return;
    const opts = getAnalysisOpts();
    computeAndPlotForChannel(k, opts);
  });
  const viewSelect = document.getElementById('plView');
  viewSelect?.addEventListener('change', (e) => {
    const k = state.activeChannel;
    if (k) {
      const opts = getAnalysisOpts();
      computeAndPlotForChannel(k, opts);
    }
  });
  const themeToggle = getThemeToggle();
  themeToggle?.addEventListener('change', () => {
    state.analysisLightTheme = !!themeToggle.checked;
    const k = state.activeChannel;
    if (k) {
      const opts = getAnalysisOpts();
      computeAndPlotForChannel(k, opts);
    }
  });
  const fsplBtn = getFsplButton();
  fsplBtn?.addEventListener('click', () => {
    state.fsplEnabled = !state.fsplEnabled;
    updateFsplButton();
    const k = state.activeChannel;
    if (k) {
      const opts = getAnalysisOpts();
      computeAndPlotForChannel(k, opts);
    }
  });
  const bestBtn = getBestCurveButton();
  bestBtn?.addEventListener('click', () => {
    state.bestCurveEnabled = !state.bestCurveEnabled;
    updateBestCurveButton();
    const k = state.activeChannel;
    if (k) {
      const opts = getAnalysisOpts();
```

```
      computeAndPlotForChannel(k, opts);
    }
  });
  const logBtn = getLogDistanceButton();
  logBtn?.addEventListener('click', () => {
    toggleLogDistancePanel();
  });
  const inputs = getLogDistanceInputs();
  inputs?.useFirst?.addEventListener('change', () => {
    if (inputs?.useFirst?.checked && inputs?.useFspl) {
      inputs.useFspl.checked = false;
    }
    syncLogDistanceInputs();
  });
  inputs?.useFspl?.addEventListener('change', () => {
    if (inputs?.useFspl?.checked && inputs?.useFirst) {
      inputs.useFirst.checked = false;
    }
    syncLogDistanceInputs();
  });
  inputs?.fitN?.addEventListener('change', () => {
    syncLogDistanceInputs();
  });
  const logFitBtn = document.getElementById('logDistanceFitBtn');
  logFitBtn?.addEventListener('click', handleLogDistanceFit);
  const logClearBtn = document.getElementById('logDistanceClearBtn');
  logClearBtn?.addEventListener('click', () => {
    clearLogDistance();
    const k = state.activeChannel;
    if (k) {
      const opts = getAnalysisOpts();
      computeAndPlotForChannel(k, opts);
    }
  });
  document.addEventListener('keydown', handleKeydown);
  updateButtons();
  updateFsplButton();
  updateBestCurveButton();
  updateLogDistanceButton();
  syncLogDistanceInputs();
  state.initialized = true;
}
// Public API to set experiment mode and availability
function setExperimentMode(on) {
  state.isExperimentMode = !!on;
  if (!state.isExperimentMode) {
    closeModal();
  }
  updateButtons();
  updateFsplButton();
  updateBestCurveButton();
  updateLogDistanceButton();
}
function setAvailability(on) {
  state.isAvailable = !!on;
  if (!state.isAvailable) {
    closeModal();
  }
  updateButtons();
  updateFsplButton();
  updateBestCurveButton();
```

```
  updateLogDistanceButton();
}
// Expose the public API
export const Analysis = {
  init,
  openModal,
  closeModal,
  setExperimentMode,
  setAvailability,
  resetForNewExperiment,
};
```

Αρχείο csvRecorder.js

```
// core/csvRecorder.js - A class for recording CSV data in-memory and
downloading as a file

export class CsvRecorder {
  constructor({ filename = 'experiment.csv', header = [], comment = [] } =
{}) {
    this.filename = filename;
    this.header = header;
    this.comment = Array.isArray(comment)
      ? comment.filter(Boolean).map((line) => String(line))
      : (comment ? [String(comment)] : []);
    this.parts = [];
    this.buffer = [];
    this.lines = 0;
    this._open = false;
  }
  // Start recording: prepare header and comment and mark as open for
appending rows
  start() {
    if (this._open) return;
    this._open = true;
    if (this.comment?.length) {
      this.comment.forEach((line) => {
        const text = line.endsWith('\n') ? line : `${line}\n`;
        this.parts.push(text);
      });
    }
    if (this.header?.length) {
      this.parts.push(this.header.join(',') + '\n');
      this.lines++;
    }
  }
  // Write a comment line to the CSV output
  writeComment(line) {
    if (!line) return;
    const text = line.endsWith('\n') ? line : `${line}\n`;
    if (!this._open) {
      this.parts.push(text);
      return;
    }
    this._flushBuffer();
    this.parts.push(text);
  }
```

```
  // Append a row of fields to the CSV output and flush buffer if it grows
too large
  appendRow(fields) {
    if (!this._open) return;
    const esc = (v) => {
      if (v == null) return '';
      const s = String(v);
      return /[,"\n]/.test(s) ? `"${s.replace(/"/g, '""')}"` : s;
    };
    this.buffer.push(fields.map(esc).join(',') + '\n');
    this.lines++;
    if (this.buffer.length >= 1000) this._flushBuffer();
  }
  // Flush the in-memory buffer of CSV rows to the parts array, which will
be used to create the final Blob
  _flushBuffer() {
    if (this.buffer.length) {
      this.parts.push(this.buffer.join(''));
      this.buffer.length = 0;
    }
  }
  // Stop recording and finalize the CSV data, returning a Blob URL for
download
  stop() {
    if (!this._open) return null;
    this._flushBuffer();
    this._open = false;
    const blob = new Blob(this.parts, { type: 'text/csv' });
    const url = URL.createObjectURL(blob);
    return { blob, url, lines: this.lines };
  }
  // Trigger a download of the recorded CSV data as a file with the
specified filename
  download() {
    const out = this.stop();
    if (!out) return null;
    const a = document.createElement('a');
    a.href = out.url;
    a.download = this.filename;
    a.style.display = 'none';
    document.body.appendChild(a);
    a.click();
    requestAnimationFrame(() => {
      URL.revokeObjectURL(out.url);
      a.remove();
    });
    return out;
  }
}
```

Αρχείο statistics.js

```
// Compute basic statistics: mean, median, standard deviation
export function mean(arr) {
  if (!Array.isArray(arr) || arr.length === 0) return 0;
  return arr.reduce((a, b) => a + b, 0) / arr.length;
}
export function median(arr) {
  if (!Array.isArray(arr) || arr.length === 0) return 0;
  const sorted = [...arr].sort((x, y) => x - y);
  const mid = Math.floor(sorted.length / 2);
  return (sorted.length % 2)
    ? sorted[mid]
    : (sorted[mid - 1] + sorted[mid]) / 2;
}
export function stddev(arr) {
  if (!Array.isArray(arr) || arr.length < 2) return 0;
  const m = mean(arr);
  const variance = arr.reduce((acc, val) => {
    const diff = val - m;
    return acc + diff * diff;
  }, 0) / (arr.length - 1);
  return Math.sqrt(variance);
}
// Compute best-fit line for log10(x), using least squares
export function computeBestFit(xs, ys) {
  const points = [];
  const len = Math.min(xs?.length ?? 0, ys?.length ?? 0);
  for (let i = 0; i < len; i++) {
    const x = xs[i];
    const y = ys[i];
    if (!Number.isFinite(x) || x <= 0 || !Number.isFinite(y)) continue;
    const lx = Math.log10(x);
    points.push({ lx, y });
  }
  if (points.length < 2) return null;

  const sum = points.reduce((acc, p) => {
    acc.lx += p.lx;
    acc.y += p.y;
    return acc;
  }, { lx: 0, y: 0 });
  const n = points.length;
  const meanLx = sum.lx / n;
  const meanY = sum.y / n;

  let num = 0;
  let den = 0;
  for (const p of points) {
    const dlx = p.lx - meanLx;
    const dy = p.y - meanY;
    num += dlx * dy;
    den += dlx * dlx;
  }
  if (den === 0) return null;
  const slope = num / den;
  const intercept = meanY - slope * meanLx;
  let sse = 0;
  let sst = 0;
  for (const p of points) {
    const pred = intercept + slope * p.lx;
```

```
    const err = p.y - pred;
    sse += err * err;
    const dy = p.y - meanY;
    sst += dy * dy;
  }
  const rmse = n > 0 ? Math.sqrt(sse / n) : NaN;
  const r2 = (sst === 0) ? 1 : 1 - (sse / sst);
  return { slope, intercept, rmse, r2, count: n};
}
// Generate linearly or logarithmically spaced x values
export function generateLineXs(xs, scale) {
  const valid = [...new Set((xs || []).filter(v => Number.isFinite(v) && v
> 0))].sort((a, b) => a - b);
  if (valid.length === 0) return [];
  if (valid.length === 1) return valid;

  const min = valid[0];
  const max = valid[valid.length - 1];
  const count = Math.max(10, Math.min(60, valid.length * 4));

  if (scale === 'log') {
    const logMin = Math.log10(min);
    const logMax = Math.log10(max);
    if (!Number.isFinite(logMin) || !Number.isFinite(logMax) || logMax ===
logMin) {
      return valid;
    }
    const step = (logMax - logMin) / (count - 1);
    const arr = [];
    for (let i = 0; i < count; i++) {
      arr.push(Math.pow(10, logMin + step * i));
    }
    return arr;
  }
  const step = (max - min) / (count - 1);
  const arr = [];
  for (let i = 0; i < count; i++) {
    arr.push(min + step * i);
  }
  return arr;
}
// Fit log-distance path loss model to data (least squares for fixed
intercept)
export function fitLogDistanceSlope(xs, ys, d0, pl0) {
  if (!Array.isArray(xs) || !Array.isArray(ys)) return null;
  if (!Number.isFinite(d0) || d0 <= 0 || !Number.isFinite(pl0)) return
null;
  const len = Math.min(xs.length, ys.length);
  let num = 0;
  let den = 0;
  let count = 0;
  for (let i = 0; i < len; i++) {
    const x = xs[i];
    const y = ys[i];
    if (!Number.isFinite(x) || x <= 0 || !Number.isFinite(y)) continue;
    const lx = Math.log10(x / d0);
    if (!Number.isFinite(lx)) continue;
    num += lx * (y - pl0);
    den += lx * lx;
    count += 1;
  }
```

```
  if (den === 0 || count < 2) return null;
  const slope = num / den;
  const n = slope / 10;
  return { n };
}
// Compute log-distance path loss model points
export function computeLogDistanceModel(xs, params) {
  if (!Array.isArray(xs) || !params) return null;
  const d0 = params.d0;
  const pl0 = params.pl0;
  const n = params.n;
  if (!Number.isFinite(d0) || d0 <= 0 || !Number.isFinite(pl0) ||
!Number.isFinite(n)) return null;
  const resultXs = [];
  const resultYs = [];
  for (const d of xs) {
    if (!Number.isFinite(d) || d <= 0) continue;
    const val = pl0 + 10 * n * Math.log10(d / d0);
    if (!Number.isFinite(val)) continue;
    resultXs.push(d);
    resultYs.push(val);
  }
  if (!resultXs.length) return null;
  return { xs: resultXs, ys: resultYs };
}
// Compute shadowing band for log-distance model
export function computeLogDistanceShadowingBand(ys, sigma) {
  if (!Array.isArray(ys) || !ys.length) return null;
  const sigmaVal = Number.isFinite(sigma) ? sigma : NaN;
  if (!(sigmaVal > 0)) return null;
  const upper = [];
  const lower = [];
  for (const y of ys) {
    if (!Number.isFinite(y)) {
      upper.push(NaN);
      lower.push(NaN);
      continue;
    }
    upper.push(y + sigmaVal);
    lower.push(y - sigmaVal);
  }
  return { upper, lower };
}
```

Παράρτημα V

Κώδικας CSS – Κανόνες μορφοποίησης του UI εφαρμογής

Αρχείο style.css

```
* { box-sizing: border-box; }
:root{
  --color-bg-page: #10151f;
  --color-text-body: LightGray;
  --color-text-light: #eee;
  --color-text-muted: #ddd;
  --color-form-control: #4F5D82;
  --color-form-control-hover: #7D8BB0;
  --color-focus: #1a1aff;
}
body {
  margin:0;
  background: var(--color-bg-page);
  color: var(--color-text-body);
  font-family: 'Roboto', sans-serif;
  padding: 1rem;
  padding-bottom: 3rem;
}
canvas { max-width: 100%; margin-bottom: 2rem; }

/*-------------- HEADER -----------------*/
.brand-header{
  margin: 0 0 .85rem;
  position: relative;
  display: grid;
  place-items: center;
  width: 100%;
  min-height: max(56px, 2rem);
  padding-block: .35rem;
  overflow: hidden;
}
.brand-text{
  position: relative;
  z-index: 2;
  line-height: 1.1;
  padding: .1rem .6rem;
  font-weight: 900;
  font-size: 36px;
  justify-self: start;
  margin-left: 10%;
  margin-bottom: 0;
  color: rgba(255,255,255,0.9);
}
/*------ SIGNAL-LIKE ANIMATION -------*/
.heartbeat-svg{
  position: absolute;
  left: 0; right: 0;
  top: 50%;
  transform: translateY(-50%);
  width: 100%;
  height: 56px;
  z-index: 0;
  pointer-events: none;
```

```
}
.fade-in {
  position: absolute;
  width: 100%;
  height: 100%;
  background-color: var(--color-bg-page);
  top: 0;
  right: 0;
  animation: heartRateIn 4.5s linear infinite;
  z-index: 1;
  pointer-events: none;
}
.fade-out {
  position: absolute;
  width: 140%;
  height: 100%;
  top: 0;
  left: -140%;
  z-index: 1;
  inset: 0;
  pointer-events: none;
  animation: heartRateOut 4.5s linear infinite;
  background: var(--color-bg-page);
  background: -moz-linear-gradient(left, var(--color-bg-page) 0%, var(--
color-bg-page) 50%, transparent 100%);
  background: -webkit-linear-gradient(left, var(--color-bg-page) 0%, var(--
color-bg-page) 50%, transparent 100%);
  background: -o-linear-gradient(left, var(--color-bg-page) 0%, var(--
color-bg-page) 50%, transparent 100%);
  background: -ms-linear-gradient(left, var(--color-bg-page) 0%, var(--
color-bg-page) 50%, transparent 100%);
  background: linear-gradient(to right, var(--color-bg-page) 0%, var(--
color-bg-page) 80%, transparent 100%);
}
@keyframes heartRateIn {
  0%   { width: 100%; }
  30%  { width: 0; }
  60%  { width: 0; }
  100% { width: 0; }
}
@keyframes heartRateOut {
  0%   { left: -140%; }
  30%  { left: -140%; }
  60%  { left: 0; }
  100% { left: 0; }
}
/*---------------MODE CONTROLS-----------------*/
.mode-bar {
  display: flex;
  flex-wrap: wrap;
  gap: 1.25rem ;
  align-items: center;
  position: sticky;
  top: 0;
  z-index: 5;
  backdrop-filter: blur(6px);
  background: rgba(27,33,48,.8);
  box-shadow: 0 2px 8px rgba(0,0,0,.4);
  padding: 0.75rem 1rem;
  border-radius: 10px;
  margin: 1rem auto 1.5rem;
```

```
  width: 80%;
  border: none;
}
.mode-bar label {
  font-weight: 600;
  color: var(--color-text-muted);
  cursor: pointer;
}
.mode-bar-top{
  display: grid;
  grid-template-columns: auto 1fr auto;
  grid-template-areas: "config switch toggle";
  align-items: center;
  gap: 1.25rem;
  width: 100%;
  position: relative;
}
#configBtn{
  align-self: center;
  background: transparent;
  border-color: transparent;
  grid-area: config;
  justify-self: start;
}
#modeSwitch{ grid-area: switch; justify-self: center; }

/* Mode selector (Live / Experiment) */
.mode-select{
  position:relative;
  display:inline-flex;
  border:none;
  border-radius:6px;
  background: transparent;
  overflow:hidden;
  user-select:none;
  --pad-y:.6rem; --pad-x:1.6rem;
}
.mode-select .mode-item{
  position:relative;
  appearance: none;
  font-weight: 700;
  -webkit-appearance: none;
  background:transparent;
  color: var(--color-text-light);
  opacity:.75;
  border:none;
  cursor:pointer;
  z-index:2;
  transition:color .2s, opacity .2s, transform .15s, background .2s;
  display:flex;
  align-items:center;
  gap:.45rem;
  padding:var(--pad-y) var(--pad-x);
  font-size: 1rem;
  line-height: .5;
}
.mode-select .mode-item:not(.active):hover{ color:#3385ff; opacity:1;
background: transparent; }
.mode-select .mode-item.active{
  background: transparent;
  color:#1f1fff;
```

```
  font-size: 1.5rem;
  opacity:1;
}
#togglePanelBtn{
  grid-area:toggle;
  justify-self: end;
  margin-left: 0;
  transition: transform .2s;
  background: transparent;
  border-color: transparent;
}
#togglePanelBtn[aria-expanded="false"] { transform: rotate(-180deg); }

.mode-panel {
  display: flex;
  padding: 0.75rem 1rem;
  background: rgba(34,40,56,.8);
  border-top: 2px solid #444;
  width: 100%;
  gap: 2rem;
  flex-wrap: wrap;
}

#livePanel.mode-panel{ gap: 16px; padding-top: 0.75rem; }
#combinePlotsField{ gap: .75rem; margin-left: 2rem; }

/* Button variants */
.btn{
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: .4rem;
  border-radius: 6px;
  border: none;
  line-height: 1.2;
  cursor: pointer;
  text-decoration: none;
  transition: background .2s, border-color .2s, color .2s, box-shadow .2s,
transform .2s;
}
.btn:disabled,
.btn.disabled{
  opacity: 0.45;
  pointer-events: none;
}
.btn--panel {
  padding: 0.45rem 0.75rem;
  font-size: 0.95rem;
  background: var(--color-form-control);
  color: var(--color-text-light);
  border: 1px solid transparent;
}
.btn--panel:hover:not(:disabled) {
  background: var(--color-form-control-hover);
  border-color: #666;
}
.btn--modal {
  padding: 0.25rem 0.6rem;
  font-size: 0.95rem;
  background: transparent;
  color: var(--color-text-light);
```

```
  border: 1px solid #555;
}
.btn--modal:hover:not(:disabled) {
  background: rgba(255,255,255,.4);
  color: white;
}

/* Input/Select Forms */
.form-control{
  border: none;
  outline: none;
  transition: background .2s, border-color .2s, color .2s, box-shadow .2s;
}
.form-control:focus{ outline: none; }
.form-control--select {
  padding: 0.45rem 0.75rem;
  font-size: 0.95rem;
  border-radius: 6px;
  border: none;
  background: var(--color-form-control);
  color: var(--color-text-light);
  line-height: 1.2;
  display: inline-flex;
  align-items: center;
  gap: .4rem;
}
.form-control--select:hover:not(:disabled) {
  background: var(--color-form-control-hover);
  border-color: #666;
}
.form-control--select:disabled {
  opacity: 0.45;
  pointer-events: none;
}
.form-control--input {
  padding: 0.4rem;
  border-radius: 5px;
  border: none;
  background: var(--color-form-control);
  color: white;
}
.form-control--input:focus { border: 1px solid #1a1aff; }
/* Hide number spinners in input fields*/
input[type="number"]::-webkit-outer-spin-button,
input[type="number"]::-webkit-inner-spin-button{
  -webkit-appearance: none; margin: 0;
}
input[type="number"]{
  -moz-appearance: textfield;
}

#experimentPanel.mode-panel {
  flex: 1 1 100%;
  width: 100%;
  flex-direction: column;
  flex-wrap: nowrap;
  gap: 0.75rem;
}
#experimentPanel.expPanel-bottom { width: 100%; }

.expPanel-top{
```

```
  display: grid;
  grid-template-columns: minmax(0,1fr) auto minmax(0,1fr);
  align-items: center;
  column-gap: 12px;
  width: 100%;
}
.expPanel-top .left{
  display: flex; align-items: center; gap: 12px; justify-self: start;
}
.expPanel-top .center{
  display: flex; align-items: center; gap: 8px;  justify-self: center;
}
.expPanel-top .right{
  display: flex; align-items: center; gap: 12px; justify-self: end;
}
.exp-action-panel{
  position: fixed;
  top: 50%;
  right: 2rem;
  transform: translateY(-50%);
  display: none;
  flex-direction: column;
  gap: 10px;
  padding: 0.9rem 1rem;
  background: rgba(34,40,56,.92);
  border: none;
  border-radius: 10px;
  box-shadow: 0 6px 18px rgba(0,0,0,.4);
  z-index: 6;
}
.exp-action-panel .btn{ width: 80px; }
.is-experiment .exp-action-panel{ display: flex; }

#experimentPanel .expPanel-bottom{
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
  gap: 12px 30px;
}
#experimentPanel .expPanel-bottom .group{
  display: inline-flex;
  align-items: center;
  gap: 8px;
}
#experimentPanel .expPanel-bottom .lbl{
  white-space: nowrap;
  font-weight: 600;
  margin-right: 4px;
  opacity: 0.9;
}
#experimentPanel .expPanel-bottom input[type="number"]{
  height: 25px;
}
.exp-status-text {
  color: white;
  font-size: 20px;
  font-weight: 700;
}
.exp-timer {
  color: var(--color-text-muted);
```

```
  font-weight: 600;
}
.mc-field{
  font-weight:600;
  color:var(--color-text-muted);
  display:flex;
  align-items:center;
  gap:.35rem;
}
.width-5ch { width: 5ch; }
.width-9ch { width: 9ch; }
.width-13ch { width: 13ch; }
.width-20ch { width: 20ch; }

/*------------- LIVE READINGS ---------------*/
.live-readings {
  display: inline-flex;
  position: fixed;
  left: 50%;
  bottom: 0.5rem;
  transform: translateX(-50%);
  width: fit-content;
  max-width: calc(100vw - 2rem);
  margin: 0 auto;
  gap: 2rem;
  flex-wrap: nowrap;
  overflow-x: auto;
  background: rgb(247,247,247);
  border-radius: 10px;
  padding: 0.5rem 1.5rem;
  font-family: 'Roboto', sans-serif;
  box-shadow: 0 8px 24px rgba(0,0,0,0.25);
  justify-content: center;
  text-align: center;
  z-index: 1000;
}
.live-readings .reading {
  display: inline-flex;
  align-items: center;
  gap: 0.75rem;
  font-size: 0.95rem;
  white-space: nowrap;
  color: black;
  font-weight: 700;
}
.live-readings .reading-data {
  display: inline-flex;
  align-items: baseline;
  gap: 0.75rem;
  font-size: 0.85rem;
  color: #454545;
}
.live-readings .reading.is-disabled { opacity: 0.45; }
.live-readings .reading[data-channel="rss60"] { --reading-accent: #3E51D6;
}
.live-readings .reading[data-channel="rss40"] { --reading-accent: #8F46B3;
}
.live-readings .reading-value {
  display: inline-block;
  color: var(--reading-accent, black);
  font-size: 0.95rem;
```

```
  transition: transform .2s;
}
.live-readings .reading-value:hover { transform: scale(1.05); }
.live-readings [id$="Timestamp"] { font-variant-numeric: tabular-nums; }

/*------------- CHART/TABLE BLOCK ----------------*/
.chart-table-block {
  margin: 2rem auto;
  width: 80%;
  position: relative;
}
.chart-title {
  margin: 0;
  font-size: 1.4rem;
  font-family: 'Roboto', sans-serif;
}
.block-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: .5rem;
  gap: .75rem;
}
.status-bar {
  display: flex;
  align-items: center;
  gap: .5rem;
}
.header-actions {
  margin-left: auto;
  display: flex;
  align-items: center;
  gap: .5rem;
}
.table-toggle.active { background: #1a1aff; }
.block-content {
  position: relative;
  box-sizing: border-box;
  padding-right: 0;
  overflow: hidden;
  border-radius: 6px;
  aspect-ratio: 23 / 9;
}
.plot-box{ width: 100%; height: 100%; }
.chart-table-block.show-table .block-content { padding-right: calc(320px +
1rem); }
.table-box {
  position: absolute;
  width: 320px;
  top: 0;
  right: 0;
  bottom: 0;
  overflow-y: auto;
  transform: translateX(100%);
  transition: transform .28s, opacity .28s;
  border: none;
  background: #131927;
}
.chart-table-block.show-table .table-box{
  transition-delay: .12s;
  transform: translateX(0);
```

```
  opacity: 1;
  pointer-events: auto;
}
.table {
  width: 100%;
  border-collapse: collapse;
  font-size: 0.9rem;
  font-family: 'Roboto', sans-serif;
}
.table thead th {
  position: sticky;
  top: 0;
  z-index: 1;
  background: var(--color-form-control);
  color: white;
  padding: 6px;
}
.table tbody td {
  padding: 6px 4px;
  text-align: center;
}
.table tbody tr:nth-child(even) {
  background-color: #1b2130;
}

.status-badge{
  display:flex;
  align-items:center;
  gap:.35rem;
  padding:.25rem .55rem;
  border-radius: 6px;
  background-color: transparent;
  border:1px solid #666;
  color:var(--color-text-muted);
  line-height:1;
  white-space:nowrap;
}
.status-dot{
  width:.6rem;
  height:.6rem;
  border-radius:50%;
  background:#777;
  flex:0 0 .6rem;
}
.status-online  .status-dot{ background:#2ecc71; }
.status-offline .status-dot{ background:#e74c3c; }
.status-unknown .status-dot{ background:#7f8c8d; }
.status-text{ font-weight:500; }

/*--- MODAL ---*/
.modal {
  position: fixed; inset: 0; z-index: 1000;
  display: flex; align-items: center; justify-content: center;
}
.modal-backdrop {
  position: absolute; inset: 0; background: rgba(0,0,0,0.55);
}
.modal-box {
  position: relative; width: min(1000px, 92vw); max-height: 80vh;
  background: #131927; border: none; border-radius: 10px;
  box-shadow: 0 10px 30px rgba(0,0,0,.6); overflow: hidden;
```

```
  display: flex; flex-direction: column;
}
.modal-header-top, .modal-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.modal-header-top {
  padding: .75rem 1rem;
  border-bottom: 1px solid #333;
}
.modal-box h4 { margin: 0.5rem 0; }
.modal-close {
  background: transparent;
  border: 0;
  color: var(--color-text-light);
  font-size: 1.4rem;
  cursor: pointer;
}
.modal-body {
  padding: .25rem 1rem; overflow: auto;
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.cfg-section {
  display: flex;
  flex-direction: column;
  gap: .75rem;
}
.cfg-divider { border-top: 1px solid rgba(255, 255, 255, 0.12); }
.table-wrap { max-height: 260px; overflow: auto; }
.col-actions { width: 220px; }
.modal .danger { border-color: #733; color: #f4cccc; }
.modal .danger:hover { background: #3a1f1f; }
.muted { color: #aaa }
#cfgRate60,#cfg60_a, #cfg40_a { margin-right: 16px; }
#importCsvBtn{ margin: 0.5rem 0 0.75rem auto; }
#cfgSaveBtn{ align-self: flex-end; margin-right:12px;}
/*--- ANALYSIS MODAL ---*/
#analysisModal .modal-box {
  width: 80%;
  max-width: none;
  max-height: 95vh;
  height: auto;
}
#analysisModal .modal-body{
  max-height: calc(92vh - 90px);
  overflow: auto;
}
#analysisModal .modal-header-top {
  justify-content: center;
  position: relative;
}
#analysisModal #analysisCloseBtn {
  position: absolute;
  right: 1rem;
}
#analysisModal .modal-body > div:first-child{
  display: flex;
  flex-wrap: wrap;
```

```
  gap: 0.875rem;
  align-items: center;
  padding: 0.75rem 1rem;
}
#analysisModal #analysisComputeBtn { margin-left: auto; }
#analysisModal .plot-box { height:auto; aspect-ratio: 23/9; }
.analysis-toggles {
  padding: 10px 0px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 8px;
  border-bottom: 1px solid #333;
}
.logDistance {
  position: relative;
  display: inline-flex;
  flex-direction: column;
  align-items: flex-end;
}
.logDistance > .btn { white-space: nowrap; }
.analysis-toggles .btn.fspl-active {
  background: #ffb347;
  border-color: #ffb347;
  color: #000;
}
.analysis-toggles .btn.bestcurve-active {
  background: var(--analysis-bestcurve-color);
  border-color: var(--analysis-bestcurve-color);
  color: #000;
}
.logDistance .btn.logdistance-active {
  background:  #b93232ff;
  border-color:  #b93232ff;
  color: #000;
}
.analysis-logdistance-panel {
  position: absolute;
  bottom: calc(100% + 8px);
  background: #262f45;
  border: 1px solid #3e4865;
  border-radius: 10px;
  box-shadow: 0 14px 30px rgba(0, 0, 0, 0.45);
  padding: 14px;
  z-index: 40;
  color: #f5f6fc;
  display: flex;
  flex-direction: column;
  gap: 10px;
}
.analysis-logdistance-panel .mc-field {
  gap: 10px;
  font-size: 0.92rem;
  white-space: nowrap;
}
.analysis-logdistance-panel .analysis-logdistance-label {
  flex: 1 1 auto;
}
.analysis-results {
  margin-top: 12px;
  display: flex;
```

```
  flex-wrap: wrap;
  gap: 8px;
  align-items: flex-start;
  justify-content: center;
}
.analysis-result-box {
  border-radius: 8px;
  padding: 12px 16px;
  border: none;
  font-size: 0.95rem;
  line-height: 1.45;
  color: #000;
  text-align: left;
  flex: 0 0 auto;
  max-width: min(100%, 340px);
}
.analysis-result-box .analysis-result-title {
  font-weight: 700;
  margin-bottom: 4px;
  text-transform: uppercase;
  font-size: 15px;
  letter-spacing: 0.08em;
}
.analysis-result-box dl {
  display: grid;
  grid-template-columns: auto 1fr;
  column-gap: 12px;
  row-gap: 6px;
  margin: 0;
}
.analysis-result-box dt { font-weight: 600; }
.analysis-result-box dd { margin: 0; }
.analysis-result-box--fspl { background: rgb(255, 179, 71); }
.analysis-result-box--ols { background: var(--analysis-ols-bg); }
.analysis-result-box--log { background: #b93232ff; }

.hidden { display: none; }
```