Πρόγραμμα Μεταπτυχιακών Σπουδών
στις Σύγχρονες Ηλεκτρονικές Τεχνολογίες

Τμήμα Φυσικής
Σχολή Θετικών Επιστημών
Πανεπιστήμιο Ιωαννίνων

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ (Μ.Δ.Ε)

Ανάπτυξη ασύρματου κατανεμημένου συστήματος αυτοματισμού IoT με μικροελεγκτές.

ΑΠΟΣΤΟΛΟΣ ΓΟΥΛΑΣ
Αριθμός μητρώου: 837

Επιβλέπον: Ομότιμος Καθηγητής Μάνθος Νικόλαος

Ιωάννινα
Σεπτέμβριος 2025

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΤΙΤΛΟΣ: ΑΝΑΠΤΥΞΗ ΑΣΥΡΜΑΤΟΥ ΚΑΤΑΝΕΜΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΥΤΟΜΑΤΙΣΜΟΥ IoT ΜΕ ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΠΜΣ-ΣΗΤ

ΓΟΥΛΑΣ ΑΠΟΣΤΟΛΟΣ
Τμήμα Φυσικής

Πανεπιστήμιο Ιωαννίνων

ΓΟΥΛΑΣ ΑΠΟΣΤΟΛΟΣ
Τμήμα Φυσικής
Πανεπιστήμιο Ιωαννίνων

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία αφορά την ανάπτυξη ενός ασύρματου κατανεμημένου συστήματος αυτοματισμού βασισμένου στο Διαδίκτυο των Πραγμάτων (Internet of Things - IoT), το οποίο αξιοποιεί το πρωτόκολλο LoRa και μικροελεγκτές ESP32 για την ασύρματη συλλογή και αποστολή δεδομένων. Το σύστημα αποτελείται από δύο αυτόνομες περιφερειακές συσκευές, οι οποίες βασίζονται στη μονάδα μικροελεγκτή TTGO LoRa32 OLED V1. Οι περιφερειακές συσκευές λειτουργούν με μπαταρία και είναι εξοπλισμένες με αισθητήρες θερμοκρασίας, υγρασίας και έντασης φωτός, καθώς και με μονάδα GPS, ώστε κάθε μέτρηση να συνοδεύεται από ακριβές χρονικό στίγμα (timestamp).

Τα δεδομένα που συλλέγονται μεταδίδονται μέσω του πρωτοκόλλου LoRa σε μία κεντρική μονάδα λήψης (receiver), η οποία διακρίνει την προέλευση των δεδομένων και τα προβάλλει σε ενσωματωμένη οθόνη OLED. Στη συνέχεια, μέσω Wi-Fi, η κεντρική μονάδα αποστέλλει τα δεδομένα σε έναν τοπικό υπολογιστή μέσω πρωτοκόλλου UDP, όπου και αποθηκεύονται σε αρχείο τύπου CSV για περαιτέρω επεξεργασία.

Τέλος, έχει υλοποιηθεί μια διεπαφή χρήστη (user interface) βασισμένη σε HTML και JavaScript, η οποία επιτρέπει στον τελικό χρήστη να παρακολουθεί τα δεδομένα σε πραγματικό χρόνο μέσω φυλλομετρητή διαδικτύου, συνοδευόμενα από χρονικές - γραφικές παραστάσεις για ευκολότερη ανάλυση.

Το σύστημα που αναπτύχθηκε επιτυγχάνει αξιόπιστη και αποδοτική συλλογή, μετάδοση και απεικόνιση δεδομένων, αξιοποιώντας τεχνολογίες χαμηλής ενεργειακής κατανάλωσης και μεγάλης εμβέλειας, και αποτελεί μία λειτουργική πρόταση για εφαρμογές παρακολούθησης σε απομακρυσμένες ή δύσκολα προσβάσιμες περιοχές.

ABSTRACT

This thesis refers to development of a wireless distributed automation system based on Internet of Things (IoT) technologies, utilizing the LoRa communication protocol and the ESP32 microcontrollers. The system consists of two autonomous peripheral devices, using TTGO LoRa32 OLED V1 microcontroller module, powered by a battery and are equipped with environmental sensors (temperature, humidity, and light intensity), as well as GPS modules for timestamping each measurement.

The collected data is transmitted wirelessly via LoRa to a central receiving unit, which identifies the data source and displays it on an integrated OLED screen. Subsequently, the receiver uses Wi-Fi to send the data to a local computer via the UDP protocol. The data are stored in a CSV file for further processing.

Finally, a user interface has been developed using HTML and JavaScript, allowing users to access and visualize the data through a web browser, with graphical representations for enhanced readability and analysis.

The developed system achieves reliable and efficient data collection, transmission and visualization, utilizing low-energy consumption and long-range technologies, and constitutes a functional proposal for monitoring applications in remote or difficult-to-access areas.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Η ραγδαία ανάπτυξη της τεχνολογίας των αισθητήρων, των μικροελεγκτών και των πρωτοκόλλων ασύρματης επικοινωνίας έχει καταστήσει δυνατή τη δημιουργία έξυπνων και αποκεντρωμένων συστημάτων αυτοματισμού. Η σύζευξη αυτών των τεχνολογιών με το Διαδίκτυο των Πραγμάτων (Internet of Things – IoT) έχει ανοίξει νέες προοπτικές σε τομείς όπως η βιομηχανία, η γεωργία, οι έξυπνες πόλεις και η περιβαλλοντική παρακολούθηση [1], [2].

Στα κατανεμημένα συστήματα αυτοματισμού, η επεξεργασία των δεδομένων και η λήψη αποφάσεων δεν περιορίζονται σε έναν κεντρικό υπολογιστή, αλλά πραγματοποιούνται από πολλαπλούς, αυτόνομους κόμβους. Αυτή η αποκεντρωμένη προσέγγιση προσφέρει αυξημένη ευελιξία, ανθεκτικότητα σε βλάβες, επεκτασιμότητα και μειωμένες απαιτήσεις καλωδίωσης, ιδιαίτερα όταν συνδυάζεται με ασύρματες τεχνολογίες χαμηλής κατανάλωσης [3].

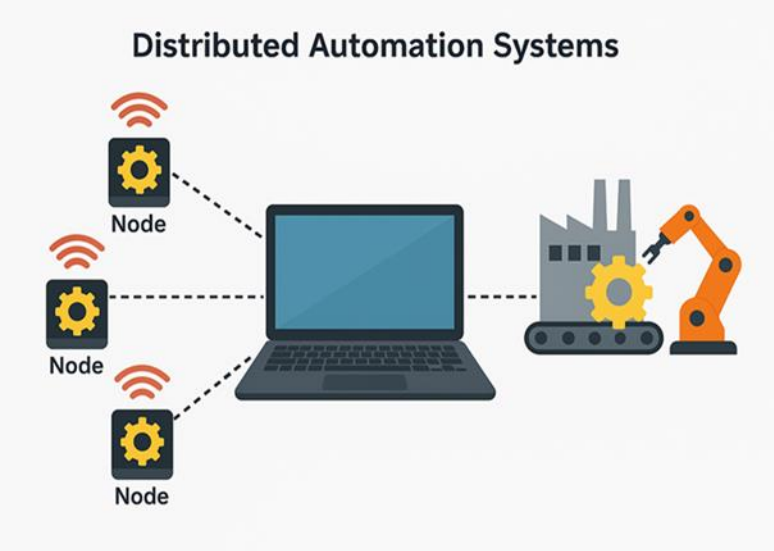
Η παρούσα εργασία εστιάζει στην ανάπτυξη ενός ασύρματου κατανεμημένου συστήματος αυτοματισμού IoT βασισμένου σε μικροελεγκτές ESP32 και στο πρωτόκολλο επικοινωνίας LoRa. Στόχος είναι η συλλογή δεδομένων από πολλαπλά σημεία, η μετάδοσή τους σε κεντρική μονάδα απευθείας ή μέσω αναμεταδοτών και η παρουσίασή τους στον χρήστη μέσω μιας φιλικής διεπαφής.

Στα επόμενα υποκεφάλαια παρουσιάζονται:

- Οι βασικές αρχές και η ιστορική εξέλιξη των κατανεμημένων συστημάτων.
- Τα κύρια χαρακτηριστικά και οι συνηθέστερες αρχιτεκτονικές τους.
- Η σύνδεσή τους με το IoT.
- Οι τεχνολογίες ασύρματης επικοινωνίας που αξιοποιούνται.
- Τα βασικά υλικά στοιχεία, όπως αισθητήρες και μικροελεγκτές.

1.1 Κατανεμημένα Συστήματα Αυτοματισμού

Τα κατανεμημένα συστήματα αυτοματισμού (Distributed Automation Systems – DAS) αποτελούν μια σύγχρονη κατηγορία τεχνολογικών λύσεων που βασίζονται στην αποκεντρωμένη συλλογή, επεξεργασία και αξιοποίηση δεδομένων μέσω ενός συνόλου αυτόνομων και αλληλοσυνδεμένων κόμβων. Στην εικόνα 1.1 περιγράφεται σχηματικά η λειτουργία ενός κατανεμημένου συστήματος αυτοματισμού. Κάθε κόμβος διαθέτει τοπικούς πόρους υπολογιστικής ισχύος, δυνατότητες επεξεργασίας και επικοινωνίας, καθώς και την ικανότητα να εκτελεί λειτουργίες ελέγχου ή λήψης αποφάσεων χωρίς την ανάγκη συνεχούς εποπτείας από έναν κεντρικό ελεγκτή.



Εικόνα 1.1 Αρχιτεκτονική Κατανεμημένου Αυτοματισμού

Σε αντίθεση με τα κεντρικοποιημένα συστήματα ελέγχου, στα οποία όλες οι διεργασίες εξαρτώνται από έναν ενιαίο υπολογιστικό πυρήνα, τα κατανεμημένα συστήματα βασίζονται στη διανομή της νοημοσύνης και της υπολογιστικής λειτουργικότητας σε πολλαπλά σημεία του δικτύου. Η προσέγγιση αυτή προσδίδει στα συστήματα αυξημένη ευελιξία, ανθεκτικότητα σε σφάλματα, επικοινωνιακή αποδοτικότητα και επεκτασιμότητα, καθιστώντας τα ιδανικά για εφαρμογές μεγάλης κλίμακας ή για λειτουργία σε δυναμικά και απομακρυσμένα περιβάλλοντα.

Η αξιοποίηση ασύρματων τεχνολογιών επικοινωνίας συμβάλλει καθοριστικά στη μείωση της πολυπλοκότητας εγκατάστασης και στη βελτίωση της ενεργειακής αποδοτικότητας. Παράλληλα, επιτρέπει την ανάπτυξη των συστημάτων αυτών σε περιοχές όπου η καλωδίωση είναι τεχνικά δύσκολη ή οικονομικά ασύμφορη. Η εξέλιξη των ενσωματωμένων συστημάτων (embedded systems), σε συνδυασμό με τη μείωση του κόστους αισθητήρων και μικροελεγκτών, έχει επιταχύνει τη μετάβαση προς κατανεμημένες αρχιτεκτονικές. Επιπλέον, η ανάπτυξη δικτύων χαμηλής ισχύος και μεγάλης εμβέλειας, όπως τα LoRa, ZigBee, NB-IoT και Wi-Fi HaLow [4] , έχει επιτρέψει τη δημιουργία ενεργειακά αποδοτικών και αξιόπιστων υποδομών επικοινωνίας για εφαρμογές του Διαδικτύου των Πραγμάτων (IoT).

Στο σύγχρονο ερευνητικό πλαίσιο, τα κατανεμημένα συστήματα αυτοματισμού εξελίσσονται σε ευφυείς, συνεργατικές υποδομές, όπου κάθε κόμβος μπορεί να λειτουργεί αυτόνομα, να ανταλλάσσει πληροφορίες και να συμβάλλει στη συνολική βελτιστοποίηση της λειτουργίας του συστήματος. Η ενσωμάτωση τεχνολογιών υπολογιστικού νέφους (cloud και edge computing) επιτρέπει την αποθήκευση, επεξεργασία και οπτικοποίηση μεγάλου όγκου δεδομένων σε πραγματικό χρόνο, ενώ η αξιοποίηση τεχνητής νοημοσύνης και μηχανικής μάθησης καθιστά εφικτή την ανάπτυξη μηχανισμών προβλεπτικού ελέγχου και έξυπνης συντήρησης.

Συνολικά, τα κατανεμημένα συστήματα αυτοματισμού αποτελούν θεμελιώδη συνιστώσα της μετάβασης προς την τέταρτη βιομηχανική επανάσταση (Industry 4.0), συνδυάζοντας αποδοτικότητα, αξιοπιστία και προσαρμοστικότητα. Η μελέτη και η περαιτέρω ανάπτυξή τους συμβάλλει καθοριστικά στη διαμόρφωση ευφυών, διασυνδεδεμένων και ενεργειακά βιώσιμων υποδομών, ικανών να υποστηρίξουν τις αυξανόμενες απαιτήσεις των σύγχρονων βιομηχανικών και περιβαλλοντικών εφαρμογών.

1.1.1 Βασικά Χαρακτηριστικά

Χαρακτηριστικά που καθιστούν τα κατανεμημένα συστήματα ιδιαίτερα ελκυστικά είναι [5], [6]:

- **Αποκέντρωση της υπολογιστικής ισχύος**: Η δυνατότητα κάθε κόμβου να επεξεργάζεται δεδομένα τοπικά μειώνει την ανάγκη για συνεχή επικοινωνία με μια κεντρική μονάδα και καθιστά το σύστημα πιο ευέλικτο και ανθεκτικό.
- **Ανοχή σε σφάλματα**: Η αποτυχία ενός κόμβου δεν οδηγεί απαραίτητα σε συνολική αποτυχία του συστήματος, καθώς οι υπόλοιποι κόμβοι συνεχίζουν να λειτουργούν κανονικά.
- **Επεκτασιμότητα**: Νέοι κόμβοι μπορούν να προστεθούν εύκολα στο σύστημα χωρίς να απαιτείται σημαντική τροποποίηση στην υπάρχουσα αρχιτεκτονική ή λογισμική υποστήριξη.
- **Απλοποίηση εγκατάστασης και συντήρησης**: Ιδιαίτερα όταν συνδυάζονται με ασύρματες τεχνολογίες, τα κατανεμημένα συστήματα περιορίζουν την ανάγκη για εκτεταμένη καλωδίωση, διευκολύνοντας την ανάπτυξη σε απομακρυσμένες ή δύσκολα προσβάσιμες περιοχές.
- **Ενεργειακή απόδοση**: Χρήση κόμβων χαμηλής κατανάλωσης ενέργειας.

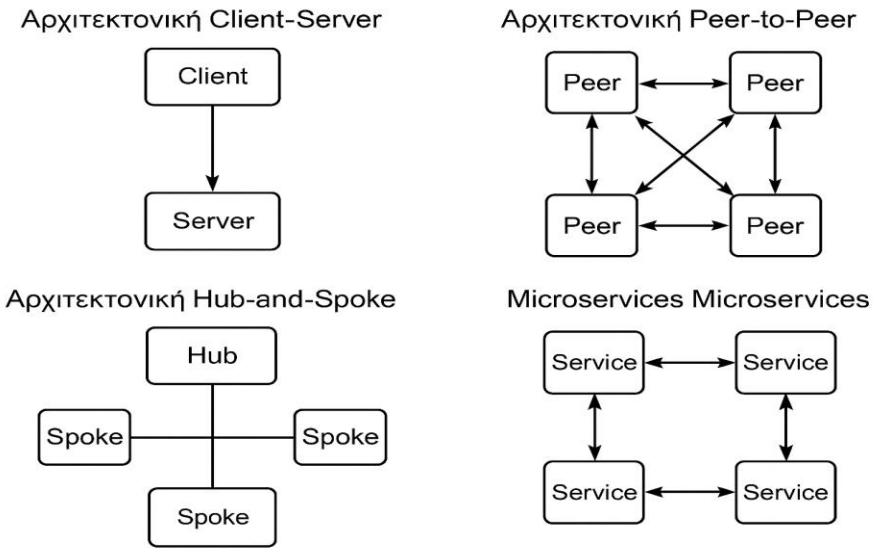
Παρά τα πολλαπλά θετικά του χαρακτηριστικά, η υλοποίηση τέτοιων συστημάτων συνεπάγεται και προκλήσεις – περιορισμούς όπως:

- **Διαχείριση ενέργειας:** Απαιτείται βέλτιστη χρήση ενέργειας, ιδιαίτερα όταν οι κόμβοι λειτουργούν με μπαταρία.
- **Ασφάλεια δεδομένων:** Η ασύρματη επικοινωνία ενέχει κινδύνους υποκλοπής ή αλλοίωσης δεδομένων.
- **Συγχρονισμός:** Η σωστή χρονική σύμπτωση των μετρήσεων είναι κρίσιμη για τη σωστή λειτουργεία των συστημάτων.
- **Συντήρηση και παρακολούθηση:** Απαιτείται συνεχής επίβλεψη για τον εντοπισμό ελαττωματικών κόμβων.

1.1.2 Αρχιτεκτονικές Κατανεμημένων Συστημάτων

Οι συνηθέστερες αρχιτεκτονικές περιλαμβάνουν [6]:

- **Client-server**: Οι κόμβοι (clients) στέλνουν δεδομένα σε έναν κεντρικό διακομιστή για επεξεργασία και αποθήκευση. Χρησιμοποιείται συχνά για την προβολή δεδομένων σε κεντρικά dashboards. Πλεονέκτημα αποτελεί η κεντρική διαχείριση, ενώ μειονέκτημα είναι η εξάρτηση από έναν κόμβο (single point of failure).
- **Peer-to-peer (P2P)**: Κάθε κόμβος μπορεί να επικοινωνεί απευθείας με άλλους. Είναι κατάλληλο για αποκεντρωμένα συστήματα χωρίς κεντρική εξάρτηση και προσφέρει αυξημένη ανθεκτικότητα, αλλά συνεπάγεται αυξημένη πολυπλοκότητα στον συγχρονισμό και τον έλεγχο της κατάστασης του συστήματος.
- **Hub-and-spoke**: Οι περιφερειακές συσκευές (spokes) στέλνουν δεδομένα σε έναν κόμβο-hub, ο οποίος συλλέγει, επεξεργάζεται και τα προωθεί. Είναι μια αρχιτεκτονική απλή στην υλοποίηση και διαχείριση, αλλά περιορίζεται από τη διαθεσιμότητα του hub. Η παρούσα εργασία υλοποιεί αυτήν την αρχιτεκτονική, με τις συσκευές TTGO ως περιφερειακούς κόμβους και την κεντρική μονάδα λήψης ως hub.
- **Service-Oriented / Microservices**: Κάθε λειτουργία ή κόμβος προσφέρει ανεξάρτητες υπηρεσίες με χρήση API. Αυτή η προσέγγιση προσφέρει μεγάλη ευελιξία, δυνατότητα επαναχρησιμοποίησης και ενσωμάτωσης με cloud εφαρμογές, αλλά απαιτεί καλή σχεδίαση και μηχανισμούς συντονισμού.
- **Event-Driven**: Οι κόμβοι ενεργοποιούνται και επικοινωνούν με βάση την εμφάνιση γεγονότων (π.χ. νέα μέτρηση αισθητήρα). Η αρχιτεκτονική αυτή είναι ιδιαίτερα χρήσιμη για real-time εφαρμογές και αυτοματισμούς, και βρίσκει εφαρμογή σε συστήματα IoT, όπως το παρόν, όπου τα δεδομένα αποστέλλονται μόνο όταν υπάρχει νέο περιεχόμενο.



Εικόνα 1.2 Σύγκριση δικτυακών αρχιτεκτονικών

Η επιλογή αρχιτεκτονικής εξαρτάται από παράγοντες όπως η αξιοπιστία, το εύρος κάλυψης, η κατανάλωση ενέργειας και οι απαιτήσεις συγχρονισμού. Στην πράξη, συχνά υιοθετείται υβριδική προσέγγιση με συνδυασμό περισσότερων από μία αρχιτεκτονικές, προκειμένου να αξιοποιηθούν τα πλεονεκτήματα κάθε μοντέλου.

Εκτός από τις βασικές αρχιτεκτονικές, όπως **client-server** και **P2P**, ιδιαίτερο ενδιαφέρον παρουσιάζει και η αρχιτεκτονική **hub-and-spoke**, κατά την οποία ένας κεντρικός κόμβος (hub) συλλέγει και επεξεργάζεται δεδομένα από περιφερειακούς κόμβους (spokes). Η αρχιτεκτονική αυτή εφαρμόστηκε στην παρούσα εργασία, όπου οι περιφερειακές μονάδες λειτουργούν ως κόμβοι συλλογής δεδομένων, ενώ ο κεντρικός κόμβος ενεργεί ως σημείο συγκέντρωσης, προβολής και προώθησης δεδομένων σε τοπικό υπολογιστή.

Παράλληλα, η χρήση του πρωτοκόλλου UDP και η ανάπτυξη διεπαφής χρήστη με HTML και JavaScript ενισχύουν τον χαρακτήρα **client-server**, προσδίδοντας στο σύστημα δυνατότητες διαμοιρασμένης επεξεργασίας και απομακρυσμένης παρακολούθησης. Τέλος, η αρχιτεκτονική του συστήματος, αν και απλή, υποστηρίζει επεκτασιμότητα, καθιστώντας δυνατή την ένταξη επιπλέον κόμβων.

1.1.3 Εφαρμογές

Τα κατανεμημένα συστήματα αυτοματισμού βρίσκουν εφαρμογή σε ποικίλους τομείς. Συγκεκριμένα περιλαμβάνουν [7], [8]:

- **Αγροτικές δραστηριότητες**: Παρακολούθηση υγρασίας εδάφους, θερμοκρασίας, επιπέδων φωτισμού και καιρικών συνθηκών. Επιτρέπουν την αυτόματη ενεργοποίηση ποτίσματος και τη βελτιστοποίηση των καλλιεργητικών πρακτικών.
- **Έξυπνες πόλεις (Smart Cities)**: Εφαρμογές όπως διαχείριση φωτισμού οδών, στάθμευσης, παρακολούθηση ποιότητας αέρα, έξυπνοι κάδοι απορριμμάτων και έλεγχος κυκλοφορίας. Αυτά τα συστήματα συμβάλλουν στη βιώσιμη και αποδοτική λειτουργία των πόλεων.
- **Βιομηχανία (Industry 4.0 [29])**: Παρακολούθηση γραμμών παραγωγής, έλεγχος ποιότητας, ανάλυση απόδοσης εξοπλισμού (OEE), και προγνωστική συντήρηση μέσω αισθητήρων και τεχνικών μηχανικής μάθησης.
- **Έξυπνα κτίρια (Smart Buildings)**: Αυτοματισμοί σχετικοί με τον φωτισμό, τον κλιματισμό, την ασφάλεια (ανίχνευση καπνού, κινήσεων), και τη διαχείριση ενέργειας, οι οποίοι μειώνουν το λειτουργικό κόστος και βελτιώνουν την εμπειρία του χρήστη.
- **Υγεία (e-Health)**: Φορητές συσκευές παρακολούθησης ζωτικών ενδείξεων, απομακρυσμένη παρακολούθηση ασθενών και αυτοματοποιημένη ειδοποίηση προσωπικού υγείας σε περιπτώσεις κρίσης.
- **Περιβαλλοντική παρακολούθηση**: Συστήματα που καταγράφουν ρύπους, στάθμη νερών, θερμοκρασία περιβάλλοντος και άλλες μεταβλητές, για χρήση από περιβαλλοντικές υπηρεσίες και ερευνητικούς φορείς.

Οι εφαρμογές αυτές αναδεικνύουν την ευελιξία και τη σημαντικότητα των κατανεμημένων συστημάτων στην υποστήριξη έξυπνων και αυτοματοποιημένων λύσεων, βελτιώνοντας την ποιότητα ζωής, μειώνοντας τα κόστη και αυξάνοντας την αποδοτικότητα.

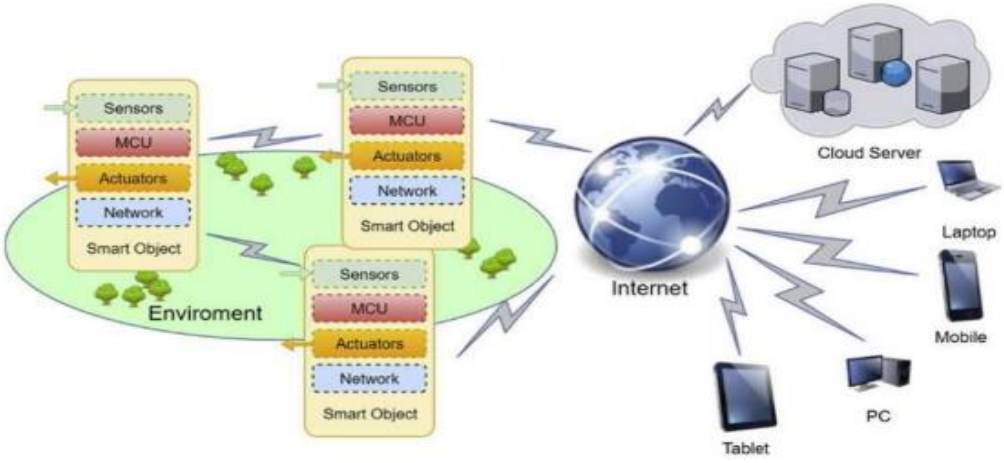
1.2 Το Διαδίκτυο των Πραγμάτων (IoT)

Το **Διαδίκτυο των Πραγμάτων (Internet of Things – IoT)** αποτελεί μια τεχνολογική προσέγγιση που συνδέει φυσικές συσκευές με το διαδίκτυο, επιτρέποντάς τους να συλλέγουν, να ανταλλάσσουν και να επεξεργάζονται δεδομένα αυτόνομα. Οι συσκευές αυτές, εξοπλισμένες με αισθητήρες, επεξεργαστές και δυνατότητες επικοινωνίας, σχηματίζουν ένα εκτεταμένο δίκτυο που υποστηρίζει πλήθος εφαρμογών σε βιομηχανικό, αστικό και οικιακό περιβάλλον χωρίς την ανάγκη συνεχούς ανθρώπινης παρέμβασης [1].

Τα κατανεμημένα συστήματα αυτοματισμού βρίσκονται στον πυρήνα της φιλοσοφίας του IoT, καθώς κάθε «έξυπνη» συσκευή λειτουργεί ως **κόμβος** ενός δικτύου, με δυνατότητα λήψης και αποστολής δεδομένων προς άλλες συσκευές ή προς κεντρικούς εξυπηρετητές. Η αποκεντρωμένη αρχιτεκτονική που χαρακτηρίζει τα κατανεμημένα συστήματα προσφέρει σημαντικά πλεονεκτήματα για το IoT, όπως:

- **Επεκτασιμότητα**: δυνατότητα εύκολης προσθήκης νέων συσκευών χωρίς εκτεταμένες αλλαγές στην υποδομή.
- **Αξιοπιστία**: διατήρηση λειτουργίας ακόμη και σε περίπτωση βλάβης ενός μέρους του συστήματος.
- **Μειωμένη καθυστέρηση**: τοπική επεξεργασία δεδομένων μειώνει την ανάγκη για συνεχή επικοινωνία με κεντρικούς διακομιστές.
- **Αποδοτική διαχείριση ενέργειας**: χρήση κόμβων χαμηλής κατανάλωσης σε εφαρμογές με μπαταρία.

Ο συνδυασμός IoT και κατανεμημένων συστημάτων αυτοματισμού οδηγεί στη δημιουργία **έξυπνων, προσαρμοστικών και αποδοτικών λύσεων** που βελτιώνουν την ποιότητα ζωής, αυξάνουν την παραγωγικότητα και μειώνουν το λειτουργικό κόστος. Από τις **έξυπνες πόλεις** και τα **βιομηχανικά δίκτυα αισθητήρων** μέχρι την **παρακολούθηση περιβαλλοντικών συνθηκών**, η ανάπτυξη αυτών των τεχνολογικών τομέων αποτελεί θεμέλιο για την εξέλιξη της τέταρτης βιομηχανικής επανάστασης (Industry 4.0) [30].



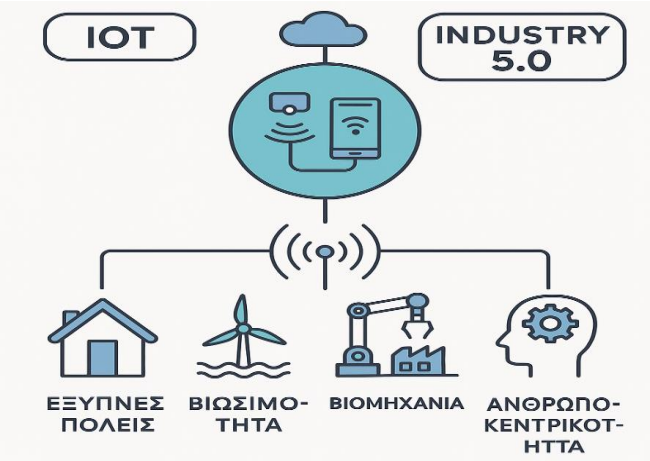
Εικόνα 1.3 Η δομή μιας βασικής πλατφόρμας έξυπνων αντικειμένων [31]

Η βασική ιδέα πίσω από το IoT είναι ότι οποιοδήποτε αντικείμενο μπορεί να καταστεί «έξυπνο» όταν του προστεθούν τα κατάλληλα υποσυστήματα για μέτρηση, ανάλυση και επικοινωνία. Στην πράξη, το IoT αποτελείται από τρία κύρια μέρη: τους αισθητήρες και τις συσκευές που συλλέγουν δεδομένα από το περιβάλλον, τις τεχνολογίες επικοινωνίας που μεταδίδουν αυτές τις πληροφορίες, και τις πλατφόρμες λογισμικού που αποθηκεύουν και αναλύουν τα δεδομένα, ώστε να παρέχονται χρήσιμες υπηρεσίες στον τελικό χρήστη.

Η λειτουργία του IoT ακολουθεί έναν κύκλο που ξεκινά από την ανάγνωση των δεδομένων (sensing), συνεχίζει με τη μετάδοσή τους (communication), ακολουθείται από την επεξεργασία τους είτε τοπικά (edge computing) είτε σε απομακρυσμένους διακομιστές (cloud computing) και καταλήγει στην εκτέλεση ενεργειών ή στην παροχή πληροφοριών στον χρήστη [32].

Οι εφαρμογές του IoT καλύπτουν ένα ευρύ φάσμα, από τα έξυπνα σπίτια και τις έξυπνες πόλεις, μέχρι τη βιομηχανική παραγωγή, την υγεία, την αγροτική παρακολούθηση και την περιβαλλοντική προστασία. Τα πλεονεκτήματα του IoT περιλαμβάνουν την αυτοματοποίηση, την καλύτερη αξιοποίηση πόρων, τη μείωση κόστους και τη βελτίωση της ποιότητας ζωής, ενώ οι προκλήσεις σχετίζονται με την ασφάλεια δεδομένων, τη διαλειτουργικότητα διαφορετικών συστημάτων, τη διαχείριση μεγάλων όγκων δεδομένων και την ενεργειακή απόδοση των συσκευών. Σε άμεση σύνδεση με τα κατανεμημένα συστήματα αυτοματισμού, το IoT επωφελείται από την αποκεντρωμένη επεξεργασία και τη δυνατότητα τοπικής λήψης αποφάσεων, δημιουργώντας ανθεκτικές, επεκτάσιμες και αποδοτικές υποδομές για ένα πλήθος σύγχρονων εφαρμογών.

Παράλληλα, η έννοια του **Industry 5.0** έρχεται να συμπληρώσει και να εξελίξει το πλαίσιο του Industry 4.0 [9], δίνοντας έμφαση στη συνεργασία ανθρώπου και μηχανής, στην εξατομίκευση της παραγωγής και στη βιωσιμότητα. Στο Industry 5.0, το IoT παίζει κεντρικό ρόλο, καθώς επιτρέπει τη συνεχή ροή δεδομένων από τις γραμμές παραγωγής, την παρακολούθηση σε πραγματικό χρόνο και τη λήψη αποφάσεων βασισμένων σε δεδομένα, προσφέροντας παράλληλα εργαλεία που προάγουν την ανθρώπινη δημιουργικότητα και την περιβαλλοντική υπευθυνότητα. Ο συνδυασμός IoT και Industry 5.0 διαμορφώνει ένα νέο οικοσύστημα παραγωγής, όπου η τεχνολογία και ο άνθρωπος συνυπάρχουν σε ένα δυναμικό, ευέλικτο και βιώσιμο περιβάλλον εργασίας.



Εικόνα 1.4 Το IoT ως βάση του Industry 5.0

1.3 Ασύρματες Επικοινωνίες

Οι ασύρματες επικοινωνίες αποτελούν μία από τις σημαντικότερες τεχνολογικές εξελίξεις στην ιστορία των τηλεπικοινωνιών και της πληροφορικής, επιτρέποντας τη μετάδοση δεδομένων χωρίς τη χρήση καλωδίων. Η βασική τους αρχή στηρίζεται στη χρήση ηλεκτρομαγνητικών κυμάτων για την αποστολή και λήψη πληροφοριών μεταξύ πομπών και δεκτών. Αυτή η δυνατότητα έχει βελτιώσει τον τρόπο με τον οποίο οι άνθρωποι και οι συσκευές συνδέονται, επιτρέποντας την ανάπτυξη κινητής τηλεφωνίας, δορυφορικών επικοινωνιών, ασύρματων δικτύων και, πιο πρόσφατα, του Διαδικτύου των Πραγμάτων (IoT).

Οι πρώτες μορφές ασύρματης επικοινωνίας εμφανίστηκαν στα τέλη του 19ου αιώνα με τα πειράματα του Guglielmo Marconi, ο οποίος το 1895 κατάφερε να στείλει σήμα ραδιοκυμάτων σε απόσταση αρκετών χιλιομέτρων. Στις αρχές του 20ού αιώνα, η τεχνολογία αυτή χρησιμοποιήθηκε για θαλάσσιες επικοινωνίες και στρατιωτικές εφαρμογές. Η δεκαετία του 1970 σηματοδότησε την αρχή της ευρείας χρήσης ασύρματων δικτύων, με τα πρώτα κυψελωτά δίκτυα κινητής τηλεφωνίας και τα συστήματα δορυφορικής επικοινωνίας. Στη δεκαετία του 1990 και 2000, η διάδοση του Wi-Fi και του Bluetooth έκανε την ασύρματη σύνδεση μέρος της καθημερινότητας, ενώ η ανάπτυξη τεχνολογιών όπως το Zigbee και το LoRa διεύρυνε τις δυνατότητες χαμηλής ισχύος και μεγάλης εμβέλειας, ειδικά για εφαρμογές IoT. Σήμερα, η άφιξη του 5G και των δικτύων χαμηλής ισχύος ευρείας περιοχής (LPWAN) δημιουργεί νέες προοπτικές για εφαρμογές σε βιομηχανία, υγεία, έξυπνες πόλεις και αυτόνομα οχήματα [10]. Στην εικόνα 1.5 περιγράφεται εν συντομία η εξέλιξη των ασύρματων επικοινωνιών.

Εικόνα 1.5 Η εξέλιξη των ασύρματων επικοινωνιών στο χρόνο

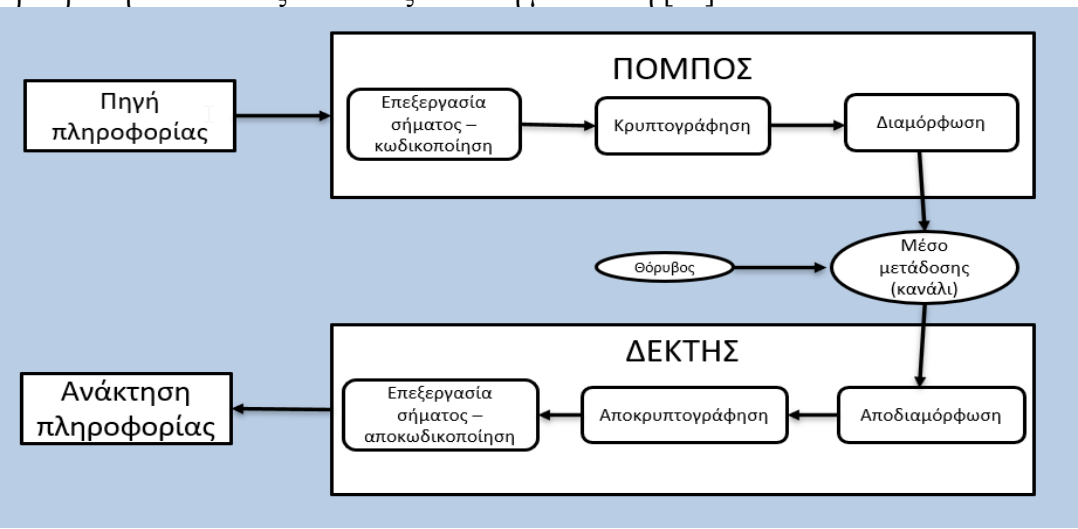
1.3.1 Χαρακτηριστικά

Τα βασικά χαρακτηριστικά των ασύρματων επικοινωνιών που επηρεάζουν την επιλογή τεχνολογίας είναι:

- **Εμβέλεια**: Καθορίζει τη μέγιστη απόσταση μετάδοσης χωρίς σημαντική απώλεια σήματος.
- **Ρυθμός μετάδοσης δεδομένων**: Εκφράζεται σε bits ανά δευτερόλεπτο (bps) και καθορίζει τον όγκο πληροφορίας που μεταδίδεται σε δεδομένο χρόνο.
- **Συχνότητα λειτουργίας**: Καθορίζει τη ζώνη συχνοτήτων στην οποία λειτουργεί η τεχνολογία (π.χ. 2.4 GHz, 868 MHz).
- **Κατανάλωση ενέργειας**: Σημαντικός παράγοντας για φορητές ή απομακρυσμένες συσκευές που τροφοδοτούνται με μπαταρίες.
- **Αξιοπιστία και αντοχή σε παρεμβολές**: Ικανότητα της επικοινωνίας να παραμένει σταθερή σε περιβάλλοντα με θόρυβο.
- **Ασφάλεια**: Προστασία δεδομένων μέσω κρυπτογράφησης και ελέγχου πρόσβασης.

1.3.2 Τρόπος Επικοινωνίας

Η ασύρματη επικοινωνία περιλαμβάνει τον πομπό, ο οποίος κωδικοποιεί την πληροφορία σε ηλεκτρικό σήμα, την κρυπτογραφεί για να διασφαλίσει την εμπιστευτικότητα και την ακεραιότητά της, και στη συνέχεια τη μετατρέπει σε ηλεκτρομαγνητική ακτινοβολία που μεταδίδεται μέσω του αέρα. Στην άλλη πλευρά της επικοινωνιακής αλυσίδας, ο δέκτης λαμβάνει το σήμα, πραγματοποιεί τη διαδικασία αποκρυπτογράφησης και αποκωδικοποίησης, ώστε να ανακτήσει την αρχική πληροφορία με τη μέγιστη δυνατή ακρίβεια. Η ενσωμάτωση μηχανισμών κρυπτογράφησης (όπως συμμετρικών ή ασύμμετρων αλγορίθμων, π.χ. AES ή RSA) αποτελεί κρίσιμο παράγοντα για την ασφάλεια των ασύρματων δικτύων, καθώς προστατεύει τα δεδομένα από μη εξουσιοδοτημένη πρόσβαση και πιθανές επιθέσεις κατά τη μετάδοση [11].

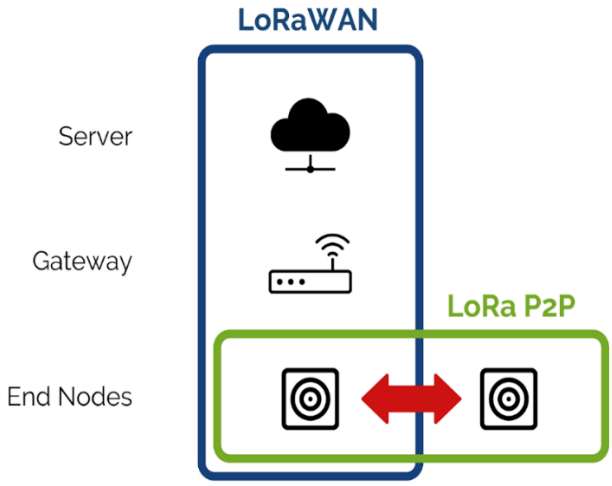


Εικόνα 1.6 Ροή πληροφορίας από τον πομπό στον δέκτη

Λεπτομέρειες για τα στάδια της ασύρματης επικοινωνίας αναφέρονται στο Παράρτημα Α.1.1.

1.3.3 Πρωτόκολλο Επικοινωνίας LoRa

Το **LoRa (Long Range)** είναι ένα ασύρματο πρωτόκολλο επικοινωνίας που αναπτύχθηκε από τη Semtech και είναι σχεδιασμένο για ασύρματες μεταδόσεις σχετικά μεγάλης εμβέλειας με χαμηλή κατανάλωση ενέργειας. Αποτελεί τη βάση για τα δίκτυα LoRa Peer-to-Peer (P2P) και LoRaWAN (Long Range Wide Area Network), τα οποία χρησιμοποιούνται εκτενώς σε εφαρμογές του Διαδικτύου των Πραγμάτων (IoT) και ιδιαίτερα σε περιπτώσεις όπου απαιτείται η αποστολή μικρών ποσοτήτων δεδομένων σε μεγάλες αποστάσεις[12].



Εικόνα 1.7 Σύγκριση LoRaWAN με LoRa P2P

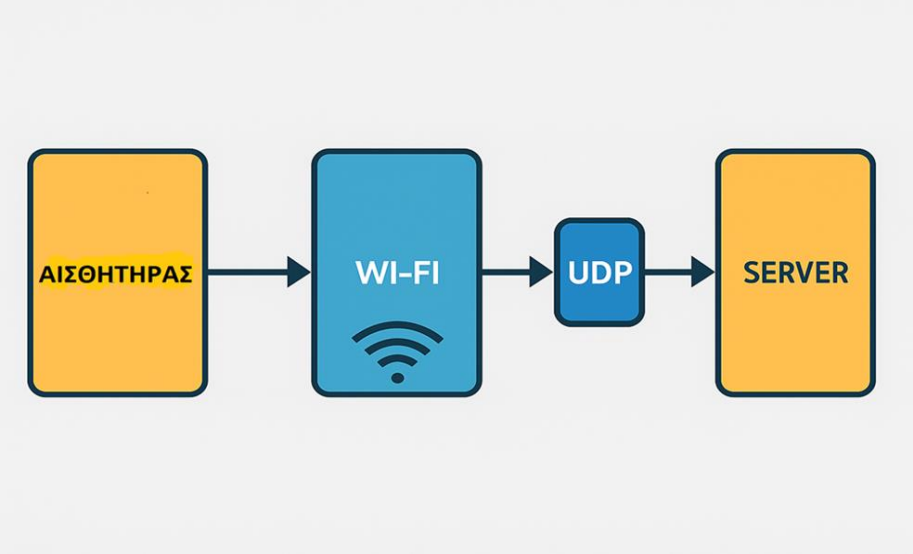
Σχετικά με το δίκτυο **LoRa** οι συσκευές LoRa επικοινωνούν μεταξύ τους σε συχνότητα 868 MHz (Ευρώπη). Η διαμόρφωση και οι παράμετροι επικοινωνίας (συχνότητα, ρυθμός μετάδοσης, bandwidth, spreading factor) πρέπει να είναι ίδιες και στις δύο πλευρές. ο ρυθμός Μετάδοσης (Data Rate) κυμαίνεται μεταξύ 0.3 kbps και 50 kbps, ανάλογα με τις παραμέτρους spreading factor (SF) και bandwidth (BW).Τα κύρια χαρακτηριστικά του είναι η χαμηλή πυκνότητα ισχύος, η πολύ μεγάλη εμβέλεια έως 10–15 km σε αγροτικές περιοχές και 2–5 km σε αστικά περιβάλλοντα, η χαμηλή κατανάλωση ενέργειας, η αντοχή σε θόρυβο και παρεμβολές λόγω της διαμόρφωσης Chirp Spread Spectrum (CSS) [33].

Επιπλέον τεχνικά χαρακτηριστικά το πρωτόκολλο LoRa Peer-to-Peer (P2P) και LoRaWAN περιγράφονται στο Παράρτημα Α.1.2.

1.3.4 Wi-Fi και UDP Πρωτόκολλο Επικοινωνίας

Το Wi-Fi αποτελεί μια ευρέως διαδεδομένη τεχνολογία ασύρματης δικτύωσης, προσφέροντας υψηλούς ρυθμούς μετάδοσης δεδομένων και συμβατότητα με πληθώρα συσκευών, γεγονός που το καθιστά κατάλληλο για εφαρμογές IoT. Σε συνδυασμό με το πρωτόκολλο UDP, το οποίο προσφέρει χαμηλή καθυστέρηση και απλότητα στη μετάδοση σε σχέση με το το άλλο βασικό πρωτόκολλο μεταφοράς του διαδικτύου TCP (Transmission Control Protocol), επιτυγχάνεται γρήγορη και αποδοτική επικοινωνία μεταξύ συσκευών και κεντρικού εξυπηρετητή. Αυτό γίνεται διότι το UDP δεν χρησιμοποιεί μηχανισμούς ελέγχου ροής, διόρθωσης σφαλμάτων ή δημιουργίας σύνδεσης, πράγμα που το καθιστά πιο ελαφρύ και γρήγορο, αν και λιγότερο αξιόπιστο. Ο συνδυασμός αυτός χρησιμοποιείται όταν προτεραιότητα έχει η ταχύτητα έναντι της πλήρους αξιοπιστίας, όπως στην παρούσα εργασία .

Η αρχιτεκτονική μιας τυπικής λύσης Wi-Fi/UDP περιλαμβάνει τον αισθητήρα μόνο του ή σε συνδυασμό με το έξυπνο αντικείμενο που συλλέγει τα δεδομένα, ένα Wi-Fi σημείο πρόσβασης που εξασφαλίζει τη σύνδεση με το τοπικό δίκτυο ή το Διαδίκτυο, και έναν απομακρυσμένο εξυπηρετητή (server) που λαμβάνει και επεξεργάζεται τα δεδομένα. Το Wi-Fi σημείο πρόσβασης λειτουργεί ως το φυσικό και συνδετικό μέσο, ενώ το πρωτόκολλο UDP καθορίζει τον τρόπο με τον οποίο μεταφέρονται τα πακέτα, προσφέροντας μια λύση που συνδυάζει υψηλή ταχύτητα, ευρεία συμβατότητα και χαμηλή καθυστέρηση. Η βασική πρόκληση είναι η διαχείριση της κατανάλωσης ενέργειας και η εξασφάλιση ικανοποιητικού επιπέδου αξιοπιστίας χωρίς να θυσιάζεται η απόδοση[13]. Στην εικόνα 1.8 περιγράφεται η μετάδοση των δεδομένων από έναν αισθητήρα προς έναν υπολογιστή - Server.



Εικόνα 1.8 Ροή πληροφορίας μέσω WiFi και UDP πρωτοκόλλου επικοινωνίας

Περισσότερες πληροφορίες σχετικά με το WiFi/UDP βρίσκονται στο Παράρτημα Α.1.3.

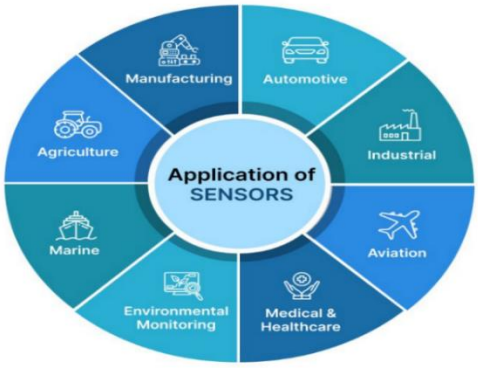
1.4 Αισθητήρες

Οι αισθητήρες αποτελούν βασικό στοιχείο κάθε συστήματος αυτοματισμού, ρομποτικής ή εφαρμογής του Διαδικτύου των Πραγμάτων (IoT), καθώς επιτρέπουν τη μέτρηση φυσικών, χημικών ή βιολογικών μεγεθών και τη μετατροπή τους σε ηλεκτρικά σήματα κατάλληλα για επεξεργασία. Η παρουσία τους είναι απαραίτητη σε οποιοδήποτε σύστημα που απαιτεί αλληλεπίδραση με το περιβάλλον, είτε για την παρακολούθηση συνθηκών είτε για την ενεργοποίηση αντιδράσεων.

Η ιστορία των αισθητήρων ξεκινά από τις πρώτες μηχανικές και ηλεκτρικές διατάξεις μέτρησης, όπως τα θερμόμετρα υδραργύρου, οι μηχανικοί διακόπτες και οι αναλογικές πιεζοηλεκτρικές διατάξεις. Στα μέσα του 20ού αιώνα, με την ανάπτυξη των ημιαγωγών, εμφανίστηκαν οι πρώτοι ηλεκτρονικοί αισθητήρες. Από τη δεκαετία του 1970 και έπειτα, η μικροηλεκτρονική και η τεχνολογία μικροεπεξεργαστών επέτρεψαν την κατασκευή ολοένα πιο ευαίσθητων και οικονομικών αισθητήρων, ενώ η ενσωμάτωση ψηφιακών κυκλωμάτων οδήγησε στην ανάπτυξη «έξυπνων αισθητήρων» δηλαδή συστημάτων αισθητήρων – επεξεργαστών που μπορούν όχι μόνο να μετρούν, αλλά και να επεξεργάζονται και να φιλτράρουν δεδομένα επιτόπου[14].

Σήμερα, η τεχνολογία των αισθητήρων βρίσκεται στην αιχμή της και εξελίσσεται ραγδαία. Η χρήση μικροηλεκτρομηχανικών συστημάτων (MEMS – Micro-Electro-Mechanical Systems) έχει μειώσει δραστικά το μέγεθος και το κόστος τους, επιτρέποντας την ενσωμάτωσή τους σε φορητές συσκευές, ιατρικά εμφυτεύματα και μικροσκοπικά drones. Παράλληλα, η ενσωμάτωση τεχνητής νοημοσύνης (AI) επιτρέπει στους αισθητήρες να αναγνωρίζουν μοτίβα και να λαμβάνουν βασικές αποφάσεις χωρίς να απαιτείται συνεχής επικοινωνία με κεντρικό υπολογιστικό σύστημα.
Οι αισθητήρες είναι πλέον παρόντες σε μια ευρεία γκάμα εφαρμογών όπως παρουσιάζεται και στην εικόνα 1.9[34] :

- **Βιομηχανία:** Παρακολούθηση θερμοκρασίας, πίεσης, δονήσεων και στάθμης σε γραμμές παραγωγής.
- **Ιατρική:** Καρδιογράφοι, οξύμετρα, αισθητήρες γλυκόζης και απεικονιστικά συστήματα.
- **Αυτοκίνηση:** ABS, συστήματα υποβοήθησης οδήγησης, ανίχνευση εμποδίων.
- **Καταναλωτικά προϊόντα:** Smartphones, wearables, οικιακός αυτοματισμός.
- **Περιβαλλοντική παρακολούθηση:** Μέτρηση ποιότητας αέρα, θερμοκρασίας, υγρασίας, ρύπανσης.



Εικόνα 1.9 Οι εφαρμογές των αισθητήρων

Ανάλογα με τον τρόπο που παράγουν και αποδίδουν την έξοδό τους, οι αισθητήρες διακρίνονται σε **αναλογικούς** και **ψηφιακούς**. Οι αναλογικοί παράγουν συνεχή σήματα που μεταβάλλονται ομαλά ανάλογα με το μετρούμενο μέγεθος, ενώ οι ψηφιακοί παράγουν διακριτά σήματα που μπορούν να διαβαστούν άμεσα από μικροελεγκτές και υπολογιστικά συστήματα. Και οι δύο κατηγορίες έχουν καθοριστικό ρόλο στη σύγχρονη τεχνολογία, και συχνά χρησιμοποιούνται συνδυαστικά για την επίτευξη υψηλής ακρίβειας και αξιοπιστίας.

Οι αναλογικοί αισθητήρες αποτελούν συσκευές που μετατρέπουν ένα φυσικό μέγεθος σε ηλεκτρικό σήμα, το οποίο μεταβάλλεται ομαλά και αναλογικά σε σχέση με το μετρούμενο φαινόμενο. Το σήμα αυτό μπορεί να αντιστοιχεί σε τάση, ρεύμα, αντίσταση ή χωρητικότητα, ανάλογα με τον τύπο και την αρχή λειτουργίας του αισθητήρα. Η κύρια ιδιότητά τους είναι ότι η παραγόμενη έξοδος μπορεί να πάρει απεριόριστο αριθμό τιμών μέσα σε ένα εύρος, αποτυπώνοντας ακόμη και τις πιο μικρές αλλαγές στο μετρούμενο μέγεθος. Ένα παράδειγμα αναλογικού σήματος παρουσιάζεται στην εικόνα 1.10.

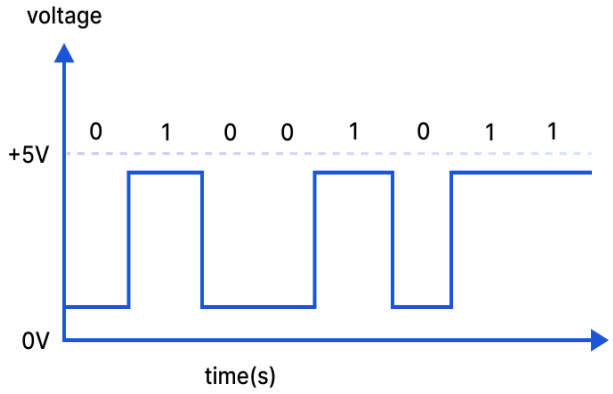
Η λειτουργία των αναλογικών αισθητήρων βασίζεται σε φυσικές αρχές όπως η μεταβολή της αντίστασης σε σχέση με τη θερμοκρασία (θερμίστορ), η αλλαγή της χωρητικότητας ανάλογα με την απόσταση ή το υλικό (αισθητήρες χωρητικότητας), ή η μεταβολή της επαγωγής σε σχέση με τη θέση μεταλλικών αντικειμένων (επαγωγικοί αισθητήρες). Το παραγόμενο σήμα συχνά χρειάζεται ενίσχυση και φιλτράρισμα πριν από την περαιτέρω επεξεργασία του, ενώ για τη χρήση του σε ψηφιακά συστήματα απαιτείται μετατροπέας A/D (Analog-to-Digital Converter) που το μετατρέπει σε ψηφιακή μορφή.

Οι αναλογικοί αισθητήρες χρησιμοποιούνται ευρέως σε εφαρμογές όπου απαιτείται υψηλή ανάλυση και συνεχής παρακολούθηση, όπως στη μέτρηση θερμοκρασίας με θερμίστορ ή θερμοζεύγη, στη μέτρηση φωτεινότητας με φωτοαντιστάσεις (LDR), στην καταγραφή πίεσης με πιεζοηλεκτρικούς μετατροπείς ή στην ανίχνευση ήχου με μικρόφωνα πυκνωτικού τύπου. Το βασικό τους πλεονέκτημα είναι η ικανότητα καταγραφής πολύ μικρών διαφορών στο μετρούμενο μέγεθος, κάτι που τους καθιστά ιδιαίτερα χρήσιμους σε εφαρμογές υψηλής ακρίβειας. Ωστόσο, μειονεκτούν σε περιβάλλοντα με έντονο ηλεκτρικό θόρυβο, καθώς το συνεχές σήμα τους μπορεί να αλλοιωθεί εύκολα από παρεμβολές, ενώ απαιτούνται πρόσθετα στάδια επεξεργασίας και μετατροπής για τη χρήση τους σε ψηφιακά συστήματα.

Στη σύγχρονη βιομηχανία και τεχνολογία, οι αναλογικοί αισθητήρες εξακολουθούν να κατέχουν κεντρικό ρόλο, ειδικά σε μετρήσεις φυσικών μεγεθών όπου η συνεχής παρακολούθηση και η υψηλή πιστότητα δεδομένων είναι καθοριστικής σημασίας. Παρά την αυξανόμενη χρήση ψηφιακών αισθητήρων, η ακρίβεια και η απλότητα των αναλογικών τους καθιστούν αναντικατάστατους σε πολλές εφαρμογές, από τον βιομηχανικό αυτοματισμό μέχρι τις επιστημονικές μετρήσεις.

Εικόνα 1.10 Αναπαράσταση Αναλογικού σήματος

Οι ψηφιακοί αισθητήρες αποτελούν συσκευές που ανιχνεύουν φυσικά, χημικά ή βιολογικά μεγέθη και τα μετατρέπουν απευθείας σε **ψηφιακή μορφή**, παρακάμπτοντας την ανάγκη εξωτερικού μετατροπέα A/D (Analog-to-Digital Converter). Σε αντίθεση με τους αναλογικούς αισθητήρες, των οποίων η έξοδος είναι ένα συνεχές σήμα, οι ψηφιακοί παρέχουν **διακριτές τιμές**, συνήθως κωδικοποιημένες σε μορφή δυαδικών δεδομένων (0 και 1), έτοιμες για επεξεργασία από μικροελεγκτές, υπολογιστές ή άλλα ψηφιακά συστήματα.



Εικόνα 1.11 Αναπαράσταση Ψηφιακού σήματος

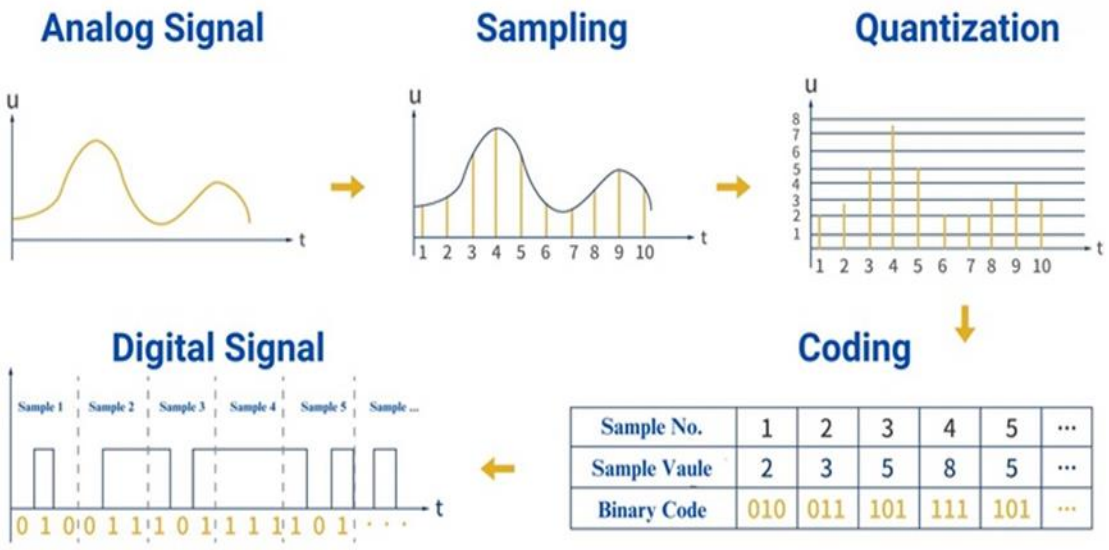
Η αρχή λειτουργίας τους βασίζεται στην ύπαρξη ενσωματωμένου κυκλώματος που εκτελεί τη μέτρηση, πραγματοποιεί τη μετατροπή σε ψηφιακή μορφή και, σε πολλές περιπτώσεις, εφαρμόζει επιπλέον επεξεργασία όπως φιλτράρισμα σήματος, βαθμονόμηση και υπολογισμό παραγώγων μεγεθών. Αυτό καθιστά τους ψηφιακούς αισθητήρες ιδιαίτερα αποδοτικούς σε συστήματα όπου η ταχύτητα εγκατάστασης, η ευκολία διασύνδεσης και η αξιοπιστία είναι κρίσιμες παράμετροι.

Ψηφιακοί αισθητήρες για παράδειγμα είναι ο **DHT22** για μέτρηση θερμοκρασίας και υγρασίας, ο **DS18B20** για ακριβή ψηφιακή μέτρηση θερμοκρασίας, ο **επιταχυνσιόμετρος MEMS** όπως ο MPU6050 για ανίχνευση κίνησης και κλίσης, καθώς και αισθητήρες απόστασης υπερήχων όπως ο HC-SR04. Οι περισσότεροι ψηφιακοί αισθητήρες επικοινωνούν μέσω τυποποιημένων σειριακών διαύλων όπως **I²C**, **SPI** ή **UART**, διευκολύνοντας τη σύνδεση με μικροελεγκτές και πλατφόρμες ανάπτυξης.

Τα πλεονεκτήματα των ψηφιακών αισθητήρων περιλαμβάνουν τη μειωμένη ευαισθησία σε θόρυβο, τη δυνατότητα απευθείας ανάγνωσης των δεδομένων χωρίς ανάγκη πρόσθετης αναλογικής επεξεργασίας, καθώς και τη συχνή ενσωμάτωση λειτουργιών αυτοδιάγνωσης και ρύθμισης. Ωστόσο, το βασικό τους μειονέκτημα είναι ότι η ανάλυσή τους εξαρτάται από τον εσωτερικό A/D μετατροπέα και τα όρια του ψηφιακού πρωτοκόλλου επικοινωνίας, ενώ σε ορισμένες εφαρμογές η καθυστέρηση απόκρισης μπορεί να είναι ελαφρώς μεγαλύτερη σε σύγκριση με έναν αντίστοιχο αναλογικό αισθητήρα.

Στη σύγχρονη εποχή, οι ψηφιακοί αισθητήρες έχουν κυριαρχήσει σε μεγάλο εύρος εφαρμογών, από απλές οικιακές λύσεις μέχρι βιομηχανικά συστήματα υψηλής τεχνολογίας, χάρη στην ευκολία ενσωμάτωσης, τη σταθερότητα και τη δυνατότητα άμεσης επικοινωνίας με «έξυπνες» πλατφόρμες και δίκτυα IoT.

Η διαδικασία μετατροπής ενός **αναλογικού σήματος** σε **ψηφιακό**, όπως συμβαίνει στους ψηφιακούς αισθητήρες περιγράφεται στην εικόνα 1.12. Αρχικά, το αναλογικό σήμα είναι μια συνεχής κυματομορφή που μεταβάλλεται ομαλά με τον χρόνο. Στη συνέχεια, μέσω της διαδικασίας του **δειγματοληψίας (sampling)**, λαμβάνονται συγκεκριμένες τιμές του σήματος σε τακτά χρονικά διαστήματα. Αυτά τα δείγματα περνούν από τη φάση της **ποσοτικοποίησης (quantization)**, όπου οι συνεχείς τιμές μετατρέπονται σε διακριτά επίπεδα. Έπειτα, εφαρμόζεται η **κωδικοποίηση (coding)**, η οποία αντιστοιχίζει κάθε επίπεδο ποσοτικοποίησης σε έναν **δυαδικό αριθμό**. Το αποτέλεσμα είναι ένα **ψηφιακό σήμα**, που αποτελείται από ακολουθίες bit (0 και 1), κατάλληλο για επεξεργασία από ηλεκτρονικά συστήματα και υπολογιστές. Έτσι, οι ψηφιακοί αισθητήρες μετατρέπουν τα φυσικά, αναλογικά φαινόμενα σε ψηφιακά δεδομένα, σε αντίθεση με τους αναλογικούς που παράγουν συνεχή σήματα.



Εικόνα 1.12 Η μετατροπή του αναλογικού σήματος σε ψηφιακό

Στον πίνακα 1.1 εμφανίζονται τα χαρακτηριστικά των αναλογικών και των ψηφιακών αισθητήρων καθώς και τα πλεονεκτήματα και τα μειονεκτήματα τους.

Πίνακας 1.1 Σύγκριση αναλογικών και ψηφιακών αισθητήρων

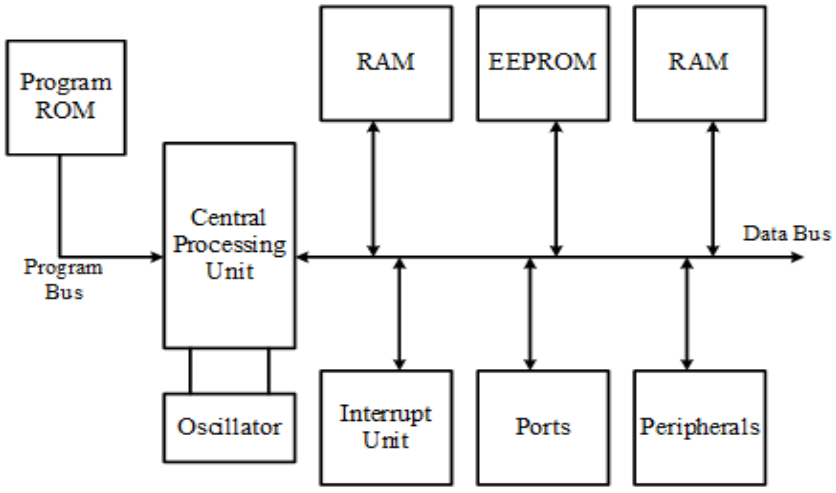
ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ

Χαρακτηριστικό	**Αναλογικοί Αισθητήρες**	**Ψηφιακοί Αισθητήρες**
Έξοδος	Συνεχές σήμα (τάση, ρεύμα, αντίσταση, χωρητικότητα)	Διακριτές ψηφιακές τιμές (0/1)
Ακρίβεια	Υψηλή, θεωρητικά απεριόριστη ανάλυση	Εξαρτάται από τον εσωτερικό A/D μετατροπέα
Αντοχή σε θόρυβο	Ευάλωτοι σε ηλεκτρομαγνητικές παρεμβολές	Υψηλή αντοχή σε παρεμβολές
Πολυπλοκότητα	Απαιτούν πρόσθετη επεξεργασία και A/D μετατροπή	Απλή σύνδεση απευθείας σε μικροελεγκτή
Κατανάλωση ενέργειας	Συνήθως χαμηλή	Μπορεί να είναι υψηλότερη λόγω ενσωματωμένων κυκλωμάτων
Τυπικές εφαρμογές	Μετρήσεις υψηλής ακρίβειας, επιστημονικά όργανα	IoT, καταναλωτικά προϊόντα, βιομηχανικός έλεγχος

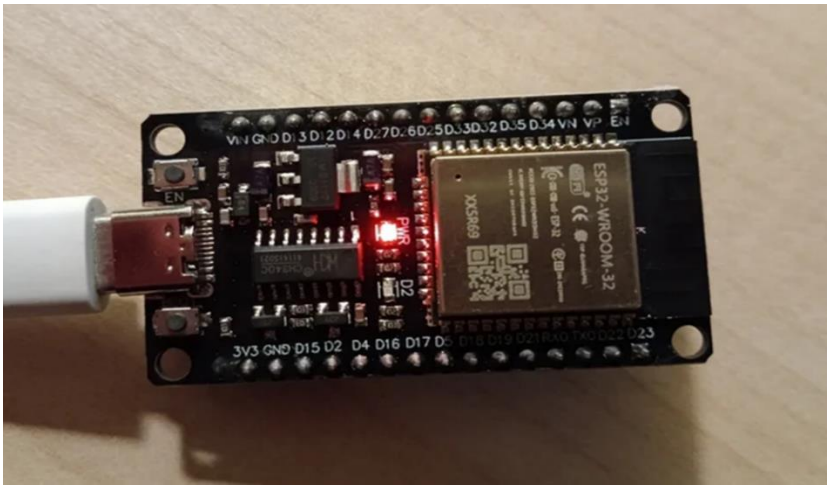
1.5 Μικροελεγκτές

Οι μικροελεγκτές (MCUs – *Microcontrollers*) αποτελούν ολοκληρωμένα κυκλώματα που περιλαμβάνουν σε ένα μόνο τσιπ κεντρική μονάδα επεξεργασίας (CPU), μνήμη και περιφερειακά εισόδου/εξόδου, επιτρέποντας τον άμεσο έλεγχο ηλεκτρονικών συσκευών και διεργασιών. Σε αντίθεση με τους μικροεπεξεργαστές (MPUs), οι οποίοι απαιτούν εξωτερικές μνήμες και υποσυστήματα για να λειτουργήσουν, οι μικροελεγκτές είναι σχεδιασμένοι ως «υπολογιστές σε ένα τσιπ» και χρησιμοποιούνται κυρίως σε ενσωματωμένα συστήματα (embedded systems), όπου η αξιοπιστία, η χαμηλή κατανάλωση και το μικρό κόστος είναι καθοριστικοί παράγοντες.

Στην εικόνα 1.13 παρουσιάζεται ένα γενικό διάγραμμα τυπικού μικροελεγκτή με επισήμανση CPU, μνήμης, GPIO και περιφερειακών. Στην εικόνα 1.14 βλέπουμε ένα παράδειγμα πλακέτας ESP32.

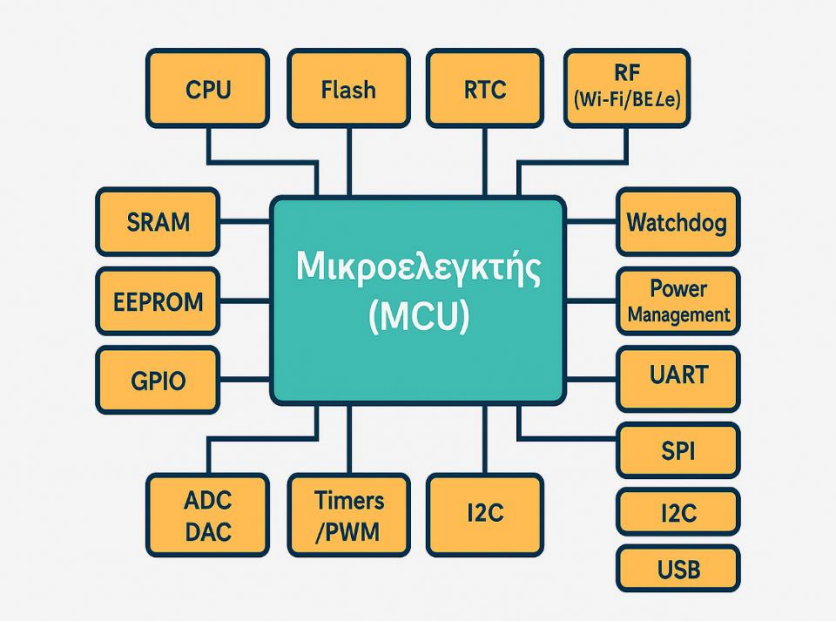


Εικόνα 1.13 Γενικό διάγραμμα τυπικού μικροελεγκτή.



Εικόνα 1.14 Πλακέτα μικροελεγκτή ESP32

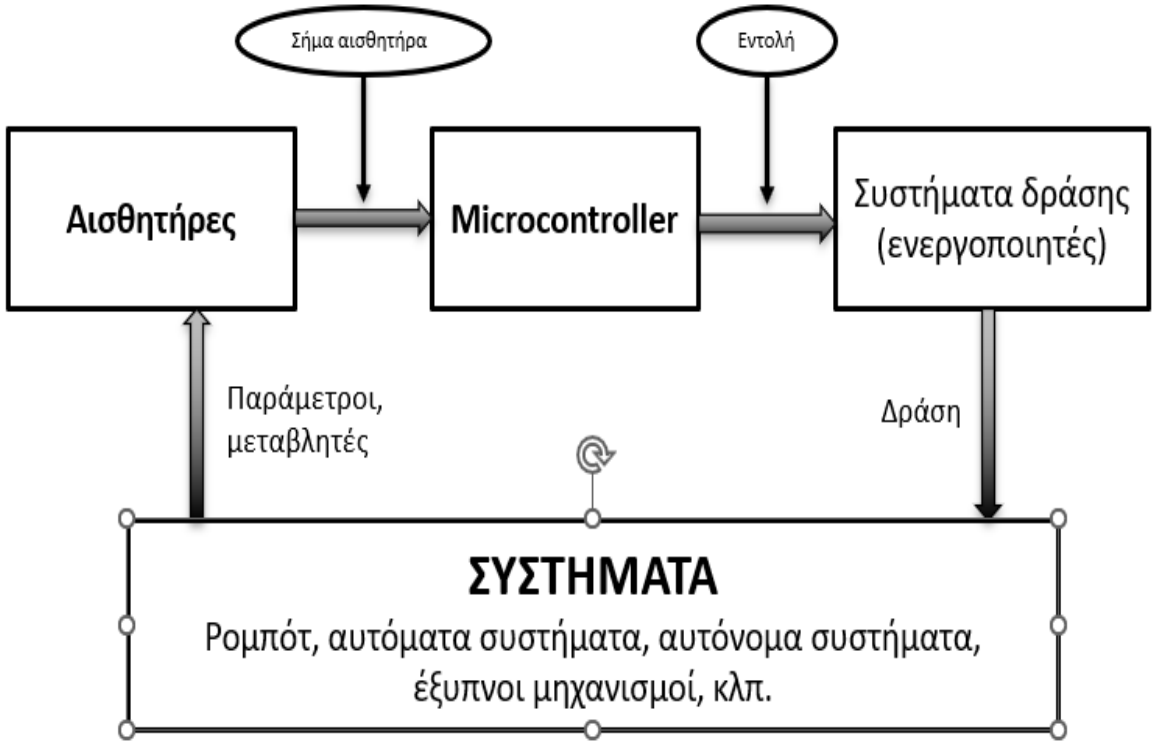
Η αρχιτεκτονική ενός μικροελεγκτή περιλαμβάνει επεξεργαστή 8, 16 ή 32 bit, με τους σύγχρονους να βασίζονται κυρίως σε πυρήνες ARM Cortex-M, RISC-V ή Xtensa για βελτιωμένη απόδοση. Η μνήμη Flash αποθηκεύει το firmware, η SRAM χρησιμοποιείται για δεδομένα κατά την εκτέλεση, ενώ συχνά ενσωματώνεται EEPROM για μόνιμες ρυθμίσεις. Τα εσωτερικά περιφερειακά περιλαμβάνουν γενικής χρήσης εισόδους/εξόδους (GPIO), μετατροπείς αναλογικού σε ψηφιακό (ADC) και μετατροπείς ψηφιακού σε αναλογικό (DAC), χρονιστές, μετρητές, εξόδους PWM, καθώς και διαύλους επικοινωνίας όπως UART, SPI, I²C, CAN και USB [15]. Σε πιο εξελιγμένες εκδόσεις, υποστηρίζονται Ethernet, SDIO (Secure Digital Input Output) και ασύρματες τεχνολογίες όπως Wi-Fi, Bluetooth ή LoRa.



Εικόνα 1.15 Γενικό διάγραμμα MCU

Ο προγραμματισμός τους γίνεται κυρίως σε C/C++, με εργαλεία όπως το Arduino IDE, το PlatformIO ή τα επίσημα SDKs (π.χ. STM32CubeIDE, MPLAB X, ESP-IDF). Σε πιο σύνθετες εφαρμογές χρησιμοποιούνται RTOS (Real-Time Operating Systems) όπως FreeRTOS ή Zephyr, που παρέχουν πολυνηματική εκτέλεση, ακριβή χρονισμό και προτεραιότητες διεργασιών. Οι μικροελεγκτές υποστηρίζουν καταστάσεις χαμηλής κατανάλωσης ενέργειας, κάτι που είναι ιδιαίτερα σημαντικό σε συστήματα που τροφοδοτούνται με μπαταρίες.

Στην πράξη, οι μικροελεγκτές συνδέονται με αισθητήρες και ενεργοποιητές, συλλέγοντας δεδομένα και εκτελώντας εντολές ελέγχου. Οι αναλογικοί αισθητήρες απαιτούν προσεκτική σχεδίαση για την προσαρμογή του σήματος στα όρια λειτουργίας του ADC, ενώ οι ψηφιακοί επικοινωνούν άμεσα μέσω σειριακών ή παράλληλων διαύλων. Η σύνδεση με ενεργοποιητές (όπως ρελέ, κινητήρες, σερβομηχανισμούς) γίνεται συνήθως με σήματα διαμόρφωσης εύρους παλμού PWM (Pulse Width Modulation) και κυκλώματα οδήγησης. Στην εικόνα 1.16 περιγράφεται ένα σύστημα που περιλαμβάνει αισθητήρες μικροελεγκτή και ενεργοποιητές.



Εικόνα 1.16 Διάγραμμα που δείχνει αισθητήρες, μικροελεγκτή, ενεργοποιητές.

Τα συστήματα μικροελεγκτών, όπως οι ESP32, έχουν επεκτείνει σημαντικά τις δυνατότητές τους ενσωματώνοντας Wi-Fi και Bluetooth, ενώ εκδόσεις με LoRa υποστηρίζουν και απομακρυσμένες εφαρμογές IoT. Επιπλέον, η αυξανόμενη υπολογιστική ισχύς επιτρέπει την υλοποίηση τεχνικών edge computing και TinyML, όπου μικρά μοντέλα μηχανικής μάθησης εκτελούνται τοπικά για άμεση λήψη αποφάσεων.

Η επιλογή του κατάλληλου μικροελεγκτή εξαρτάται από τις απαιτήσεις της εφαρμογής: διαθέσιμη μνήμη, αριθμός και τύπος περιφερειακών, κατανάλωση ισχύος, υποστήριξη πρωτοκόλλων επικοινωνίας και θερμοκρασιακό εύρος λειτουργίας. Στην παρούσα εργασία, η επιλογή του συστήματος ESP32 έγινε λόγω της ισχυρής επεξεργαστικής ικανότητας, της ενσωμάτωσης Wi-Fi και LoRa, και της μεγάλης υποστήριξης σε λογισμικό και βιβλιοθήκες.

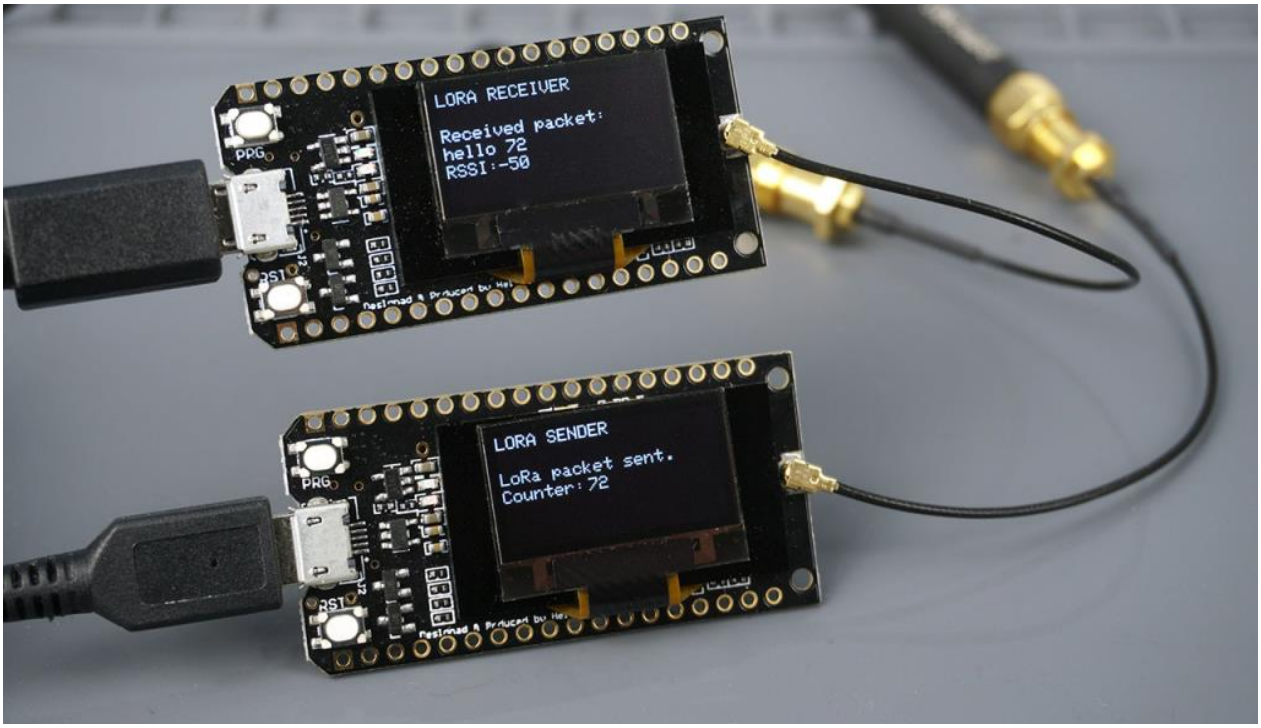
2. Ασύρματο Κατανεμημένο Σύστημα Αυτοματισμού IoT

Το σύστημα επίδειξης που αναπτύχθηκε αποτελείται από δύο αυτόνομους κόμβους μέτρησης και μία κεντρική μονάδα λήψης δεδομένων. Οι κόμβοι συλλέγουν περιβαλλοντικά δεδομένα (θερμοκρασία, υγρασία, φωτεινότητα, συντεταγμένες GPS) και τα αποστέλλουν ασύρματα με LoRa προς την κεντρική μονάδα. Εκεί πραγματοποιείται άμεση οπτική απεικόνιση σε OLED και προώθηση των πακέτων μέσω Wi-Fi/UDP σε τοπικό υπολογιστή για αποθήκευση (αρχεία CSV) και οπτικοποίηση (διεπαφή χρήστη σε φυλλομετρητή). Η αρχιτεκτονική του συστήματος είναι hub-and-spoke, δηλαδή οι περιφερειακοί κόμβοι (spokes) μεταδίδουν τα δεδομένα προς έναν δέκτη-hub, ο οποίος γεφυρώνει LoRa ↔ Wi-Fi.

2.1 Υλικό του Συστήματος

2.1.1 Το σύστημα συλλέκτη ESP32 TTGO LoRa32 OLED

Το σύστημα **ESP32 TTGO LoRa32 OLED** είναι μία ιδιαίτερα δημοφιλής μονάδα μικροελεγκτή που συνδυάζει πολλαπλές τεχνολογίες επικοινωνίας και απεικόνισης σε μία συμπαγή πλακέτα, σχεδιασμένη για εφαρμογές τηλεμετρίας **IoT** μεγάλης εμβέλειας.



Εικόνα 2.1 Μονάδα μικροελεγκτή TTGO LoRa32 OLED V1

Η πλακέτα ενσωματώνει [16], [17], [18], [19]:

Μικροελεγκτή ESP32

- Διπύρηνος επεξεργαστής 32-bit **Xtensa LX6**, 240 MHz.
- Ενσωματωμένο **Wi-Fi (2.4 GHz)** και **Bluetooth/BLE**.
- Υποστήριξη πολλαπλών διεπαφών: UART, SPI, I²C, ADC, DAC, PWM κ.ά.
- Μνήμη: 520 KB SRAM + 32 MB Flash (ανάλογα την έκδοση).

LoRa Module (Semtech SX1276)

- Δυνατότητα επικοινωνίας σε μεγάλες αποστάσεις (έως αρκετά χιλιόμετρα σε ανοιχτό πεδίο).
- Συχνότητες: 433 MHz, 868 MHz ή 915 MHz (ανά έκδοση).
- Συντελεστής Διασποράς (SF) 6–12 που καθορίζει πόσο «απλώνεται» το σήμα στον χρόνο
- Εύρος ζώνης (BW) έως 500 kHz.
- Χαμηλή κατανάλωση κατά τη μετάδοση/λήψη.

Οθόνη OLED με

- Διαγώνιο: 0.96″ ή 1.3″.
- Ανάλυση: 128×64 pixels.
- Σύνδεση μέσω I²C (στις GPIO 4 & 15).
- Για εμφάνιση μετρήσεων, κατάστασης σύνδεσης, δείκτη ισχύος λαμβανόμενου σήματος (RSSI) κ.λπ.

Διαχείριση Ενέργειας

- Θύρα micro-USB για τροφοδοσία.
- Υποδοχή μπαταρίας LiPo με ενσωματωμένο κύκλωμα φόρτισης(TP4054 chip).

Κεραία LoRa

- Εξωτερική κεραία τύπου SMA ή απευθείας wire antenna.
- Η σωστή κεραία είναι κρίσιμη για το εύρος και την ποιότητα σήματος.

Διάφορα

- Θύρα micro-USB για προγραμματισμό.
- Ρυθμιστής τάσης στα 3.3 V για ασφαλή λειτουργία του ESP32 και του LoRa.

Στον πίνακα 2.1 βρίσκονται συγκεντρωμένα τα βασικά χαρακτηριστικά της μονάδας μικροελεγκτή ESP32 TTGO LoRa32 OLED V1

Επιπλέον πληροφορίες για τους διεπαφές που αναφέρονται, βρίσκονται στο **ΠΑΡΑΡΤΗΜΑ Α.2** [20], [28].

Η διάταξη των ακροδεκτών (PINOUT), η αντιστοίχιση ακροδεκτών (Pin Mapping) και τα σχηματικά του συστήματος TTGO LORA32 OLED V1 υπάρχουν στο **ΠΑΡΑΡΤΗΜΑ Β.1** [20].

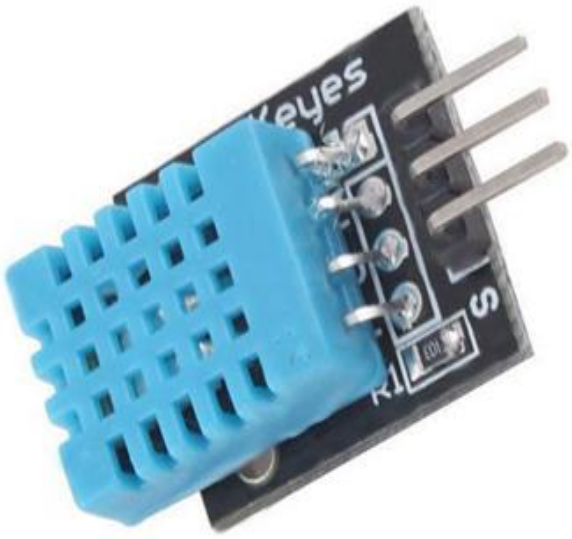
Πίνακας 2.1 Βασικά Χαρακτηριστικά του συστήματος ESP32 TTGO LoRa32 OLED V1

Χαρακτηριστικό	**Περιγραφή**
Μικροελεγκτής	ESP32 (διπύρηνος Xtensa LX6, 240 MHz με Wi-Fi και Bluetooth/BLE)
LoRa Module	SX1276 με συχνότητες 868–915 MHz με ισχύ εκπομπής έως +20 dBm και ευαισθησία έως –148 dBm
Μνήμη	Flash 32 MB
Οθόνη	OLED 0.96”, ανάλυσης 128×64 pixels, διεπαφή I^2C
USB Interface	CP2102 (USB-to-UART για προγραμματισμό/σειριακή επικοινωνία)
Κύκλωμα φόρτισης	TP4054, φόρτιση LiPo μέσω micro-USB, προστασία υπερφόρτισης/αποφόρτισης
Τάση λειτουργίας	3.3 V – 7 V
Θερμοκρασία λειτουργίας	–40 °C έως +90 °C
Wi-Fi Ρυθμοί μετάδοσης	150 Mbps (με πρότυπο WiFi 802.11n και εύρος καναλιού 40 MHz HT40), 72 Mbps (με πρότυπο WiFi 802.11n και εύρος καναλιού 20 MHz HT20), 54 Mbps (με πρότυπο WiFi 802.11g), 11 Mbps (με πρότυπο WiFi 802.11b)
Wi-Fi Ισχύς εκπομπής	19.5 dBm (11b), 16.5 dBm (11g), 15.5 dBm (11n)
Wi-Fi Ευαισθησία λήψης	έως –98 dBm
UDP Throughput	έως 135 Mbps
Διαστάσεις	~50 mm × 25 mm (ανάλογα με την έκδοση πλακέτας)
Υποστήριξη ανάπτυξης	Arduino IDE, PlatformIO, ESP-IDF

2.1.2 Digital Sensor DHT11

Ο **DHT11** είναι ένας ψηφιακός αισθητήρας θερμοκρασίας και υγρασίας. Συνδυάζει ένα στοιχείο μέτρησης υγρασίας τύπου αντίστασης και ένα θερμίστορ **NTC** για την μέτρηση της θερμοκρασίας. Επίσης έχει έναν ενσωματωμένο μικροελεγκτή 8-bit, με ADC που ψηφιοποιεί τις εν λόγω μετρήσεις. Ο αισθητήρας διαθέτει εύρος μέτρησης υγρασίας **20–90% RH** με ακρίβεια ±4-5%(ανάλογα με τη θερμοκρασία) και θερμοκρασιακό εύρος **0–50°C** με ακρίβεια ±2°C, επαρκή για εφαρμογές γενικής χρήσης. Η επικοινωνία με τον μικροελεγκτή πραγματοποιείται μέσω **μονοσύρματης σειριακής διεπαφής** (single-wire protocol), γεγονός που καθιστά την ενσωμάτωσή του απλή και γρήγορη, ενώ η τυπική περίοδος δειγματοληψίας του είναι 6 – 30 δευτερόλεπτα. Περισσότερες πληροφορίες σχετικά με τον αισθητήρα DHT11 βρίσκονται στον πίνακα 2.2.

Λόγω της μικρής κατανάλωσης ενέργειας (περίπου 1 mA κατά τη μέτρηση και 100 μA σε κατάσταση αναμονής) και του μικρού μεγέθους του, ο DHT11 είναι ιδανικός για φορητά ή αυτόνομα συστήματα με τροφοδοσία από μπαταρία[21]. Στον πίνακα 2.3 βρίσκονται οι ενεργειακές καταναλώσεις του αισθητήρα.



Εικόνα 2.2 Αισθητήρας υγρασίας – θερμοκρασίας DHT11

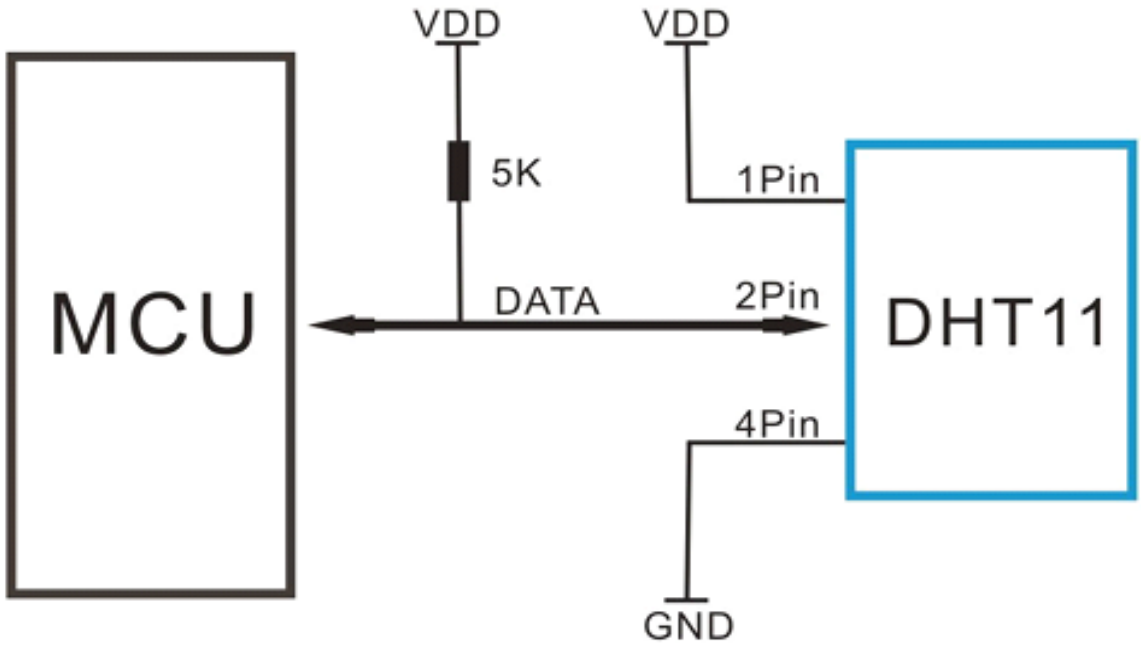
Πίνακας 2.2 Χαρακτηριστικά αισθητήρα DHT11

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Parameters	**Conditions**	**Minimum**	**Typical**	**Maximum**
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Πίνακας 2.3 Ενεργειακές καταναλώσεις του αισθητήρα DHT11

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		



Εικόνα 2.3 Διάγραμμα συνδεσμολογίας DHT11 με μικροελεγκτή.

Στην εικόνα 2.3 παρουσιάζεται η συνδεσμολογία του αισθητήρα **DHT11** στο σύστημα του **ESP32**, ώστε να καταγράφονται μετρήσεις θερμοκρασίας και υγρασίας και να αποστέλλονται για περαιτέρω επεξεργασία ή προβολή. Ο ακροδέκτης **VCC** του DHT11 συνδέεται στην έξοδο **3.3V** του ESP32 για τροφοδοσία, ενώ ο ακροδέκτης **GND** συνδέεται στη γείωση (GND) του μικροελεγκτή. Ο ακροδέκτης των δεδομένων (**DATA**) συνδέεται σε μία από τις ψηφιακές εισόδους του ESP32, και συνοδεύεται από μία **αντίσταση pull-up 5 kΩ** για σταθεροποίηση του σήματος.

Η επικοινωνία μεταξύ DHT11 και ESP32 πραγματοποιείται μέσω **μονοσύρματου πρωτοκόλλου (single-wire communication)**, όπου ο αισθητήρας αποστέλλει σειριακά τα δεδομένα θερμοκρασίας και υγρασίας στον μικροελεγκτή με καθορισμένους χρονισμούς παλμών. Το ESP32, με την υποστήριξη αντίστοιχων βιβλιοθηκών (όπως η DHT.h στην Arduino IDE ή η Adafruit_DHT στη MicroPython), αναλαμβάνει την αποκωδικοποίηση των δεδομένων και την περαιτέρω αξιοποίησή τους σε εφαρμογές IoT.

Η διαδικασία επικοινωνίας διαρκεί περίπου **4 ms** και περιλαμβάνει 40 bits δεδομένων, τα οποία περιέχουν τις πληροφορίες για τη σχετική υγρασία και τη θερμοκρασία, καθώς και ένα άθροισμα ελέγχου (checksum) για την επιβεβαίωση της ορθότητας της μετάδοσης.

Η ακολουθία έχει ως εξής [21]:

1. **Σήμα εκκίνησης (Start Signal):** Ο μικροελεγκτής (MCU) στέλνει παλμό χαμηλού επιπέδου διάρκειας τουλάχιστον **18 ms**, ώστε ο αισθητήρας να αναγνωρίσει το αίτημα ανάγνωσης. Στη συνέχεια, η γραμμή επιστρέφει σε υψηλή στάθμη για 20–40 μs, δίνοντας χρόνο στον αισθητήρα να προετοιμαστεί.

2. **Απάντηση αισθητήρα (Response Signal):** Ο DHT11 απαντάει με παλμό χαμηλού επιπέδου διάρκειας **80 μs**, ακολουθούμενο από παλμό υψηλού επιπέδου διάρκειας επίσης **80 μs**, υποδεικνύοντας ότι είναι έτοιμος να μεταδώσει δεδομένα.

3. **Μετάδοση δεδομένων:** Ο αισθητήρας στέλνει 40 bits οργανωμένα ως εξής:
   - 8-bit ακέραιης τιμής υγρασίας
   - 8-bit δεκαδικής τιμής υγρασίας
   - 8-bit ακέραιης τιμής θερμοκρασίας
   - 8-bit δεκαδικής τιμής θερμοκρασίας
   - 8-bit του αθροίσματος των προηγούμενων 4 byte (checksum)

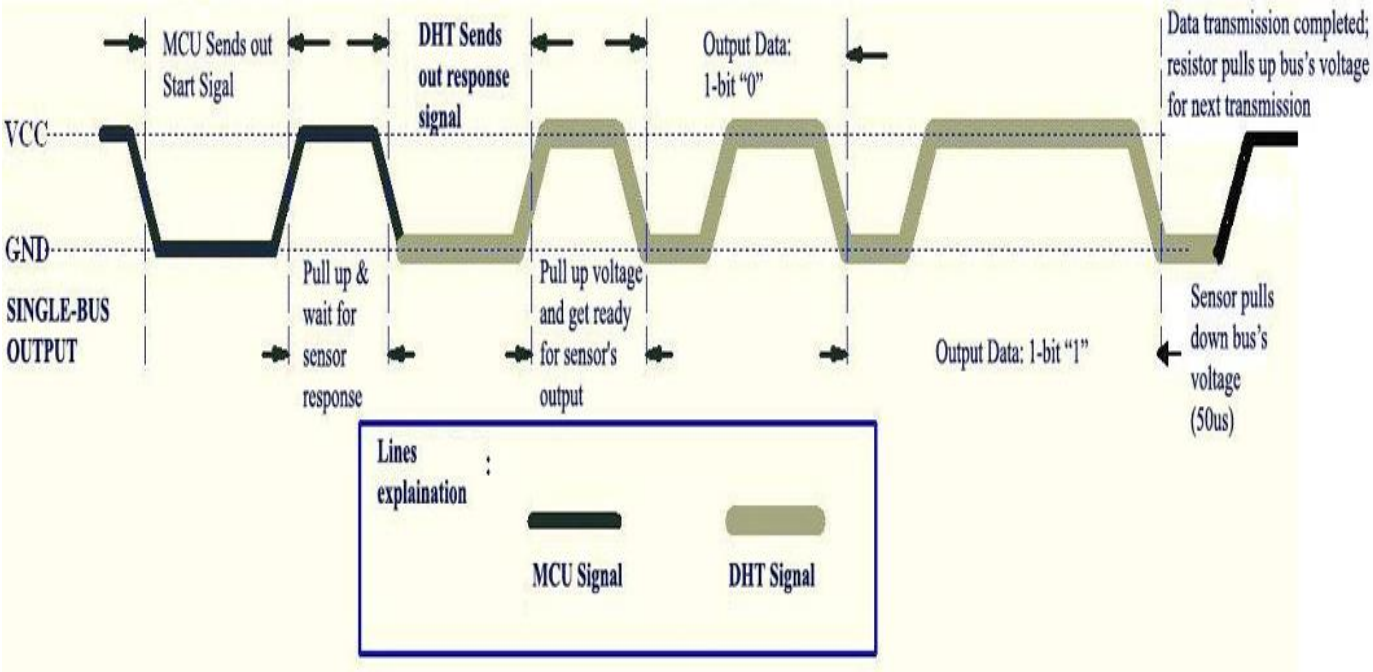
Κάθε bit ξεκινά με παλμό χαμηλής στάθμης διάρκειας **50 μs**. Η διάκριση μεταξύ «0» και «1» γίνεται από τη διάρκεια του επόμενου παλμού υψηλής στάθμης:

- **Bit "0":** ~26–28 μs υψηλή στάθμη
- **Bit "1":** ~70 μs υψηλή στάθμη

Μετά την ολοκλήρωση της μετάδοσης, ο αισθητήρας επαναφέρει τη γραμμή σε χαμηλή στάθμη για περίπου **50 μs** και στη συνέχεια επιστρέφει σε κατάσταση αδράνειας μέχρι να λάβει νέο σήμα εκκίνησης.

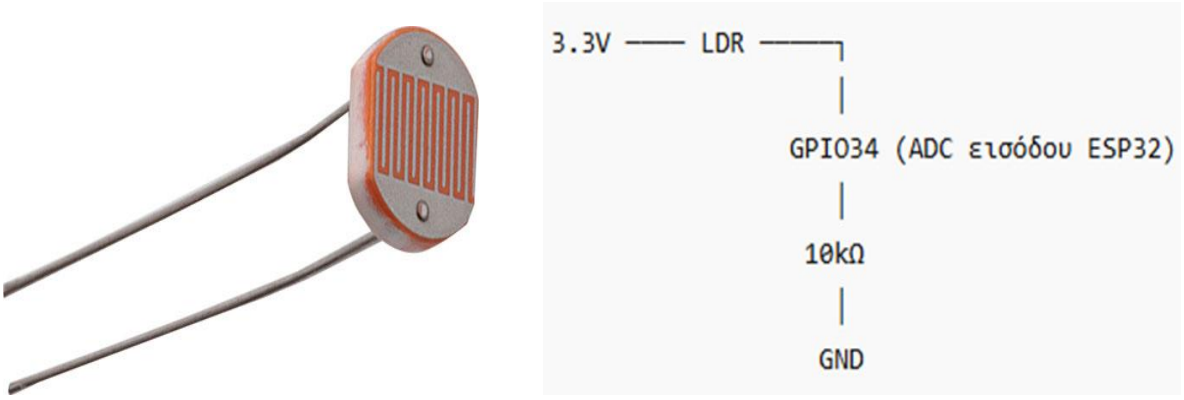
Αυτή η μέθοδος επικοινωνίας καθιστά τον DHT11 απλό στη χρήση, αφού απαιτεί μόνο ένα καλώδιο δεδομένων και μια αντίσταση pull-up. Παράλληλα, η αμφίδρομη λειτουργία του πρωτοκόλλου single-wire επιτρέπει στο μικροελεγκτή να ξεκινά τη διαδικασία και στον αισθητήρα να απαντά με τις μετρήσεις.

Στην εικόνα 2.4 φαίνεται το διάγραμμα χρονισμού των σημάτων του μικροελεγκτή και του DHT11.



Εικόνα 2.4 Συνολική διαδικασία επικοινωνίας μεταξύ του μικροελεγκτής και του αισθητήρα DHT11

2.1.3 Analog Sensor LDR



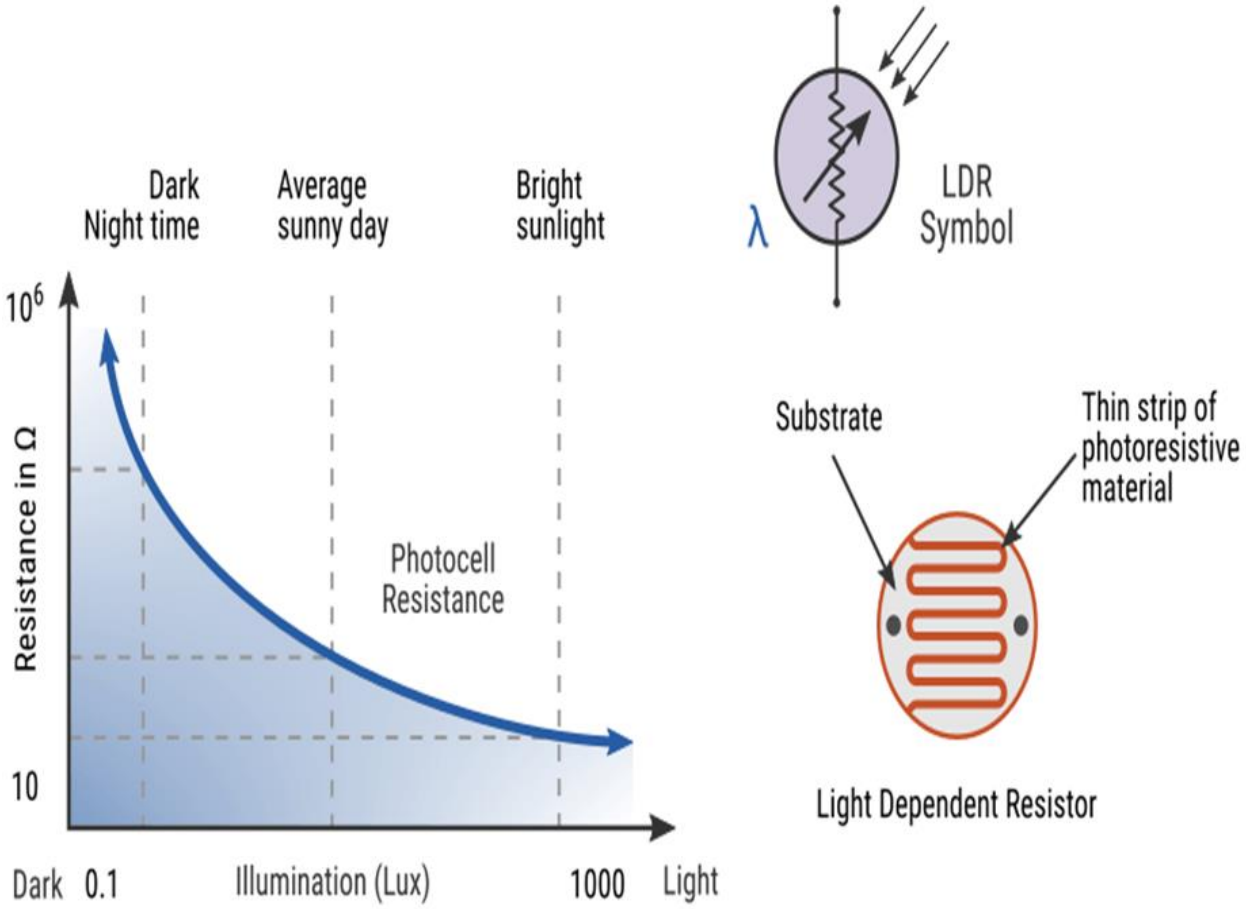
Εικόνα 2.5 Ο αισθητήρας έντασης φωτός LDR και το διάγραμμα σύνδεσης με το σύστημα ESP32

Ο **LDR Analog Sensor** είναι ένας αντιστάτης του οποίου η **αντίσταση μεταβάλλεται ανάλογα με την ένταση του φωτός** που πέφτει στην επιφάνειά της φωτοαντίστασης[22]. Η λειτουργία του βασίζεται στο φαινόμενο της φωτοαγωγιμότητας κατά το οποίο η μεταβολή της αντίστασης είναι αντιστρόφως ανάλογη με την ένταση του φωτός. Συνήθως κατασκευάζεται από φωταγωγικά υλικά όπως το Θειούχο Κάδμιο (CdS), Σεληνιούχο Κάδμιο (CdSe), Θειούχο Μόλυβδο (PbS) . Τα φωτόνια δημιουργούν ζεύγη ηλεκτρονίων–οπών στο υλικό της αντίστασης του LDR, αυξάνοντας την αγωγιμότητα και έτσι **η αντίσταση μειώνεται όσο το φως αυξάνεται** [23], [24]. Η εξάρτηση της αντίστασης από το φως δίνεται ως:

$$R_{LDR} \approx A \cdot (Lux)^{-\gamma} \ , \quad \gamma \approx 0{,}7 - 0{,}9$$

- R_{LDR}: η ηλεκτρική αντίσταση της φωτοαντίστασης.
- **Φ**: η φωτεινότητα στην LDR, σε lux (φωτομετρική μονάδα).
- **A**: σταθερά αναλογίας που εξαρτάται από τον συγκεκριμένο αισθητήρα (υλικό, επιφάνεια, κατασκευή) και τη γεωμετρία του κυκλώματος.
- **γ**: εκθέτης (0.7–0.9) που εκφράζει πόσο γρήγορα μειώνεται η αντίσταση όταν αυξάνει ο φωτισμός.

Στην εικόνα 2.6 φαίνεται ότι η φωτοαντίσταση LDR αλλάζει την αντίστασή της ανάλογα με το φως. Στο σκοτάδι αντιστέκεται πολύ στο ρεύμα, ενώ στο φως το ρεύμα περνά πιο εύκολα.
Επίσης, η φωτοαντίσταση έχει το σχήμα (σχήμα φιδιού) που βλέπουμε στο κάτω δεξιά μέρος της εικόνας 2.6, διότι έτσι αυξάνεται η ημιαγώγιμη περιοχή δηλαδή η **επιφάνεια** του φωτοευαίσθητου υλικού (π.χ. CdS) που εκτίθεται στο φως [35].



Εικόνα 2.6 Σχέση μεταξύ αντίστασης και έντασης φωτός

Τυπικές τιμές της αντίστασης είναι **>1–10 MΩ στο σκοτάδι** και **~1–10 kΩ σε έντονο φωτισμό**. Η φασματική απόκριση δηλαδή **σε ποια χρώματα ή περιοχές του φάσματος** είναι πιο ευαίσθητη η αντίσταση, κορυφώνεται περίπου στα **520–560 nm** (πράσινο), κοντά στη μεγίστη ευαισθησία του ανθρώπινου ματιού στο φως. Ο χρόνος απόκρισης της αντίστασης από το σκοτάδι (MΩ) σε φως (kΩ) είναι περίπου 10-100 ms και από το φως στο σκοτάδι είναι περίπου 0.5–2 s, γι' αυτό οι φωτοαντιστάσεις LDR δεν είναι κατάλληλες για πολύ γρήγορες μεταβολές φωτός.

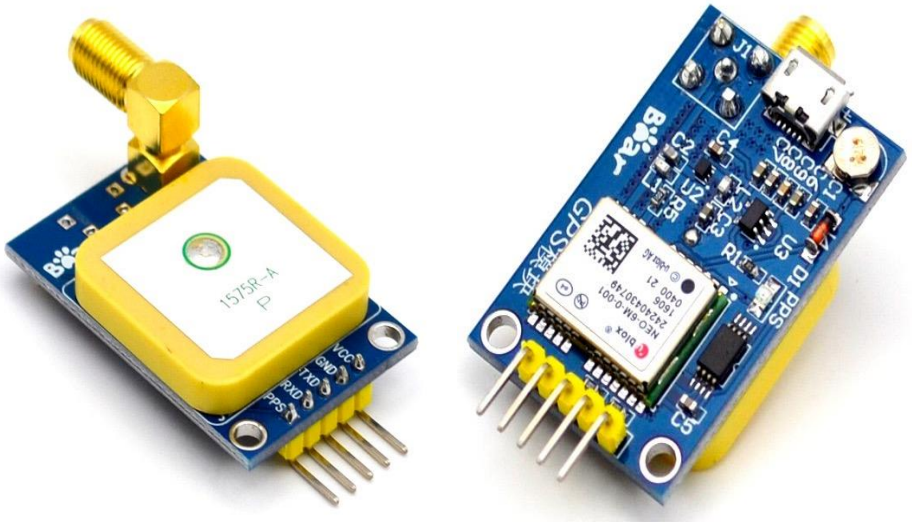
Στην εικόνα 2.5 φαίνεται το διάγραμμα διασύνδεσης του LDR με το σύστημα ESP32. Στο κύκλωμα χρησιμοποιήθηκε **διαιρέτης τάσης** για να μετατρέψει τη μεταβολή της αντίστασης της LDR σε μεταβολή τάσης, ώστε το ESP32 να μπορεί να τη διαβάσει μέσω του αναλογικού ακροδέκτη (ADC). Έτσι η τάση εξόδου που συνδέεται στο ADC (GPIO 34) είναι:

$$V_{out} = V_{CC} \cdot \frac{R_{fixed}}{R_{fixed} + R_{LDR}}$$

και το **Vout** μετριέται από τον ADC.

2.1.4 Μονάδα GPS Module

Η μονάδα **GPS** βασίζεται στον δέκτη της εταιρείας **u-blox**. Χρησιμοποιείται ευρέως σε Arduino, ESP32, Raspberry Pi και άλλα συστήματα για να λαμβάνει στίγμα, ώρα και ταχύτητα. Αποτελεί βασικό υποσύστημα του ασύρματου κατανεμημένου συστήματος αυτοματισμού, καθώς εξασφαλίζει την ακριβή χρονοσήμανση (timestamp) και γεωεντοπισμό κάθε μέτρησης. Η χρήση του επιτρέπει στο σύστημα όχι μόνο να καταγράφει περιβαλλοντικά δεδομένα (θερμοκρασία, υγρασία, φωτεινότητα), αλλά και να τα συνοδεύει με πληροφορίες θέσης και ακριβούς χρόνου, γεγονός που αυξάνει την αξιοπιστία και την αξία των δεδομένων.



Εικόνα 2.7 Μονάδα NEO 6M GPS της u-blox

Το σύστημα **NEO-6M GPS** φαίνεται στην εικόνα 2.9 και υποστηρίζει λήψη σήματος από δορυφόρους στη ζώνη L1 που είναι μια συγκεκριμένη ραδιοσυχνότητα (1575.42 MHz) που χρησιμοποιείται από τους δορυφόρους του **GPS (Global Positioning System)** για να μεταδίδουν τα σήματα τους προς τους δέκτες στη Γη.. Διαθέτει δυνατότητα επεξεργασίας διορθωτικών δεδομένων (SBAS), βελτιώνοντας την ακρίβεια εντοπισμού σε επίπεδο λίγων μέτρων (~2.5m). Χάρη στη μνήμη και την εφεδρική μπαταρία που ενσωματώνει, επιτυγχάνει ταχύτερη αρχικοποίηση και μειώνει τον χρόνο εντοπισμού θέσης σε διαδοχικές χρήσεις. Η επικοινωνία με μικροελεγκτές, όπως ο **ESP32**, γίνεται μέσω της σειριακής διεπαφής (UART) που διαθέτει [25], καθιστώντας τη διασύνδεση απλή και αποδοτική. Με αυτόν τον τρόπο, ο ESP32 είναι σε θέση να λαμβάνει και να επεξεργάζεται τα δεδομένα του δέκτη, δηλαδή γεωγραφικό πλάτος, μήκος, υψόμετρο και παγκόσμια ώρα (UTC) [26].
Το σχηματικό διάγραμμα του GPS Module βρίσκεται στο **ΠΑΡΑΡΤΗΜΑ Β.2** [27].
Στον πίνακα 2.4 συνοψίζονται τα βασικά τεχνικά χαρακτηριστικά του.

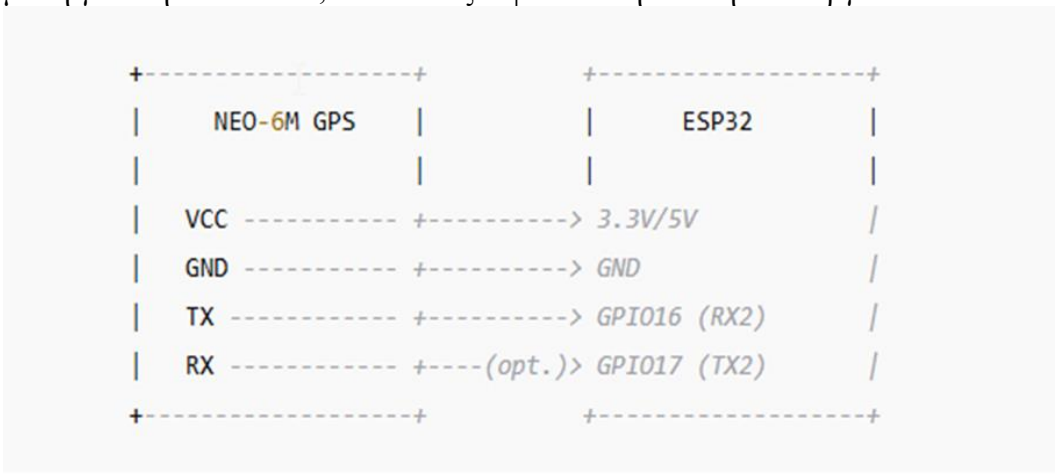
Πίνακας 2.4 Βασικά χαρακτηριστικά του συστήματος 6M GPS MODULE

Χαρακτηριστικό	**Τιμή / Περιγραφή**	**Επεξήγηση**
Συχνότητα λειτουργίας	L1 band (1575.42 MHz)	Η συχνότητα στη ζώνη L1 που χρησιμοποιούν οι δορυφόροι GPS για μετάδοση σήματος.
Ευαισθησία λήψης	έως –161 dBm	Πολύ υψηλή ευαισθησία — μπορεί να λαμβάνει πολύ ασθενή σήματα GPS, ακόμα και σε εσωτερικούς χώρους.
Ακρίβεια θέσης	~2.5 m (CEP)	Σφάλμα τοποθέτησης περίπου ±2,5 μέτρα υπό κανονικές συνθήκες. Το CEP (Circular Error Probable) σημαίνει ότι 50% των μετρήσεων βρίσκονται μέσα σε αυτόν τον κύκλο ακρίβειας.
Χρόνος εκκίνησης	Cold start ~27 s, Hot start < 1 s	Cold start: όταν δεν υπάρχουν προηγούμενα δεδομένα, χρειάζεται ~27 δευτ. να εντοπίσει δορυφόρους. Hot start: όταν έχει αποθηκευμένα δεδομένα, ξεκινά σε <1 δευτερόλεπτο.
Πρωτόκολλο επικοινωνίας	UART (NMEA sentences)	Επικοινωνεί σειριακά μέσω UART στέλνοντας τυποποιημένα NMEA μηνύματα με πληροφορίες θέσης, χρόνου και ταχύτητας.
Τυπικά μηνύματα NMEA	GPGGA, GPRMC, GPVTG	GPGGA: δεδομένα θέσης (συντεταγμένες, αριθμός δορυφόρων) GPRMC: ελάχιστα δεδομένα GPS (θέση, ταχύτητα, ώρα) GPVTG: ταχύτητα και κατεύθυνση πορείας
Τροφοδοσία	3.3 – 5 V	Τυπική τάση λειτουργίας, κατάλληλη για μικροελεγκτές όπως Arduino, ESP32, Raspberry Pi.
Κατανάλωση	~45 mA κατά τη λειτουργία	Πολύ χαμηλή κατανάλωση, ιδανικό για φορητές ή ενεργειακά αποδοτικές συσκευές.

Η λειτουργία του **GPS NEO-6M** απαιτεί σύνδεση με κεραία όπως φαίνεται και στην εικόνα 2.8 η οποία στο σύστημα μας είναι μια εξωτερική ενεργή κεραία GPS τύπου patch, που λειτουργεί στη ζώνη L1 (1575.42 MHz), διαθέτει ενσωματωμένο ενισχυτή σήματος (LNA) και τροφοδοτείται από το module GPS (NEO-6M) μέσω του καλωδίου της. Η κεραία αυτή λαμβάνει δορυφορικά σήματα και επιτρέπει την αξιόπιστη λήψη ακόμα και σε συνθήκες μερικής απόκρυψης ουρανού.

Εικόνα 2.8 Η κεραία BLUEBIRD GPS ANTENNA 1575.42MHz

Στο σύστημα που αναπτύχθηκε η μονάδα GPS τροφοδοτείται συνήθως με τάση από **3.3V έως 5V**, καθώς ενσωματώνει ρυθμιστή τάσης, ωστόσο η σειριακή επικοινωνία βασίζεται σε λογικά επίπεδα 3.3V, κάτι που το καθιστά απολύτως συμβατό με τον **ESP32**. Η επικοινωνία πραγματοποιείται μέσω **σειριακής διεπαφής (UART)**, όπου το pin **TX** του NEO-6M συνδέεται με το **RX** του ESP32 για τη μεταφορά των δεδομένων, ενώ η αντίστροφη σύνδεση (**RX του NEO-6M με TX του ESP32**) χρησιμοποιείται μόνο εφόσον απαιτείται η αποστολή εντολών ρύθμισης της μονάδας GPS από τον μικροελεγκτή. Η γείωση του module (**GND**) πρέπει να συνδεθεί κοινά με τη γείωση του ESP32, ώστε να εξασφαλιστεί η σωστή λειτουργία.



Εικόνα 2.9 Η συνδεσμολογία των ακροδεκτών του NEO 6M GPS με τους αντίστοιχους ακροδέκτες του ESP32.

Μέσω αυτής της διασύνδεσης, το GPS αποστέλλει περιοδικά δεδομένα σε μορφή χρονοσειρών **NMEA**, οι οποίες περιλαμβάνουν πληροφορίες γεωγραφικού πλάτους, μήκους, υψομέτρου, ταχύτητας και παγκόσμιας ώρας (UTC). Ο ESP32 ανακτά τα δεδομένα αυτά και να τα χρησιμοποιεί σε εφαρμογές.

Οι χρονοσειρές αυτές μεταδίδονται μέσω του διαύλου επικοινωνίας UART. Το σύστημα ESP32 μπορεί να τις διαβάσει και να τις αναλύσει με βιβλιοθήκες όπως η TinyGPS++, ώστε να εξάγει εύκολα συντεταγμένες, ώρα, ταχύτητα και κατεύθυνση.
Ένα παράδειγμα χρονοσειράς NMEA είναι:

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

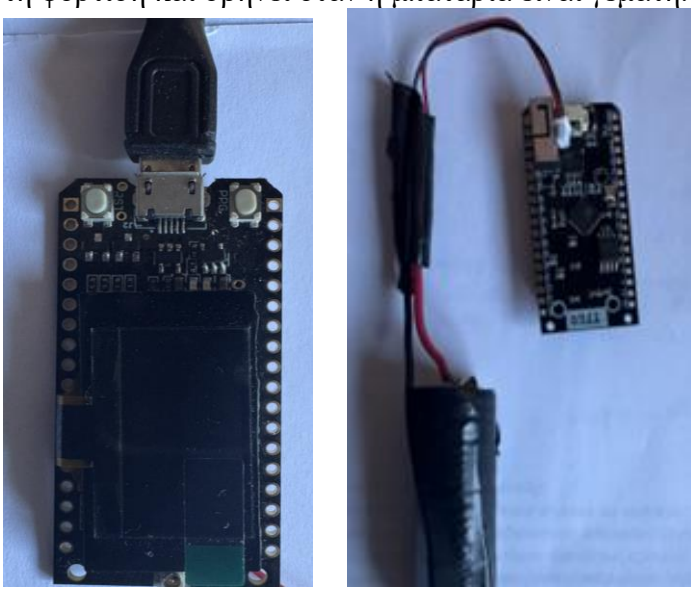
Στον πίνακα 2.5 παρέχονται οι πληροφορίες που έχει συλλέξει ο δέκτης GPS στο παραπάνω παράδειγμα.

Πίνακας 2.5 Παράδειγμα χρονοσειράς NMEA δεδομένων GPS

Πεδίο	**Ερμηνεία**	**Τιμή**
123519	Ώρα UTC	12:35:19
4807.038,N	Γεωγραφικό πλάτος	48°07.038′ Βόρεια
01131.000,E	Γεωγραφικό μήκος	11°31.000′ Ανατολικά
1	Είδος Fix (1 = ενεργό GPS Fix)	Ενεργό
08	Αριθμός δορυφόρων που χρησιμοποιούνται	8
0.9	Οριζόντια ακρίβεια (HDOP)	0.9
545.4,M	Υψόμετρο πάνω από τη στάθμη της θάλασσας	545.4μέτρα

2.1.5 Τροφοδοσία και Κατανάλωση Ενέργειας

Η σύστημα TTGO LoRa32 OLED V1 χρησιμοποιεί ένα ειδικό κύκλωμα φόρτισης, που βασίζεται στο ολοκληρωμένο TP4054 για τη διαχείριση μιας συνδεδεμένης μπαταρίας. Η τροφοδοσία μπορεί να παρέχεται μέσω USB ή μιας μπαταρίας LiPo 3,7-4,2V συνδεδεμένης σε μια υποδοχή JST 2 ακίδων όπως φαίνεται και στην εικόνα 2.10. Μια μπλε λυχνία LED υποδεικνύει τη φόρτιση και σβήνει όταν η μπαταρία είναι γεμάτη.



Εικόνα 2.10 Τροφοδοσία του συστήματος TTGO LORA32 OLED V1

Με το κύκλωμα φόρτισης που είναι ενσωματωμένο στο σύστημα, επιτυγχάνεται η απευθείας φόρτιση της μπαταρίας μέσω micro-USB, ενώ ταυτόχρονα παρέχει προστασία από υπερφόρτιση, αποφόρτιση και βραχυκύκλωμα. Η κεντρική μονάδα λήψης (receiver) του συστήματος τροφοδοτείται αποκλειστικά μέσω της USB θύρα καθώς βρίσκεται σε χώρο που υπάρχει αυτή η δυνατότητα. Από την ίδια θύρα δηλαδή πραγματοποιείται και η τροφοδοσία και ο προγραμματισμός την μονάδας.

Η κατανάλωση ισχύος εξαρτάται από τη λειτουργική κατάσταση κάθε υπομονάδας του συστήματος. Ο μικροελεγκτής **ESP32** υποστηρίζει πολλαπλές καταστάσεις λειτουργίας (active mode, modem sleep, light sleep, deep sleep), επιτρέποντας μεγάλη εξοικονόμηση ενέργειας όταν οι κόμβοι παραμένουν ανενεργοί για μεγάλα χρονικά διαστήματα. Έτσι, η διάρκεια της μπαταρίας μπορεί να αντιστοιχεί σε μήνες ανάλογα με της λειτουργίες του συστήματος.
Ενδεικτικά, οι καταναλώσεις ρεύματος του συστήματος φαίνονται στον πίνακα 2.6.

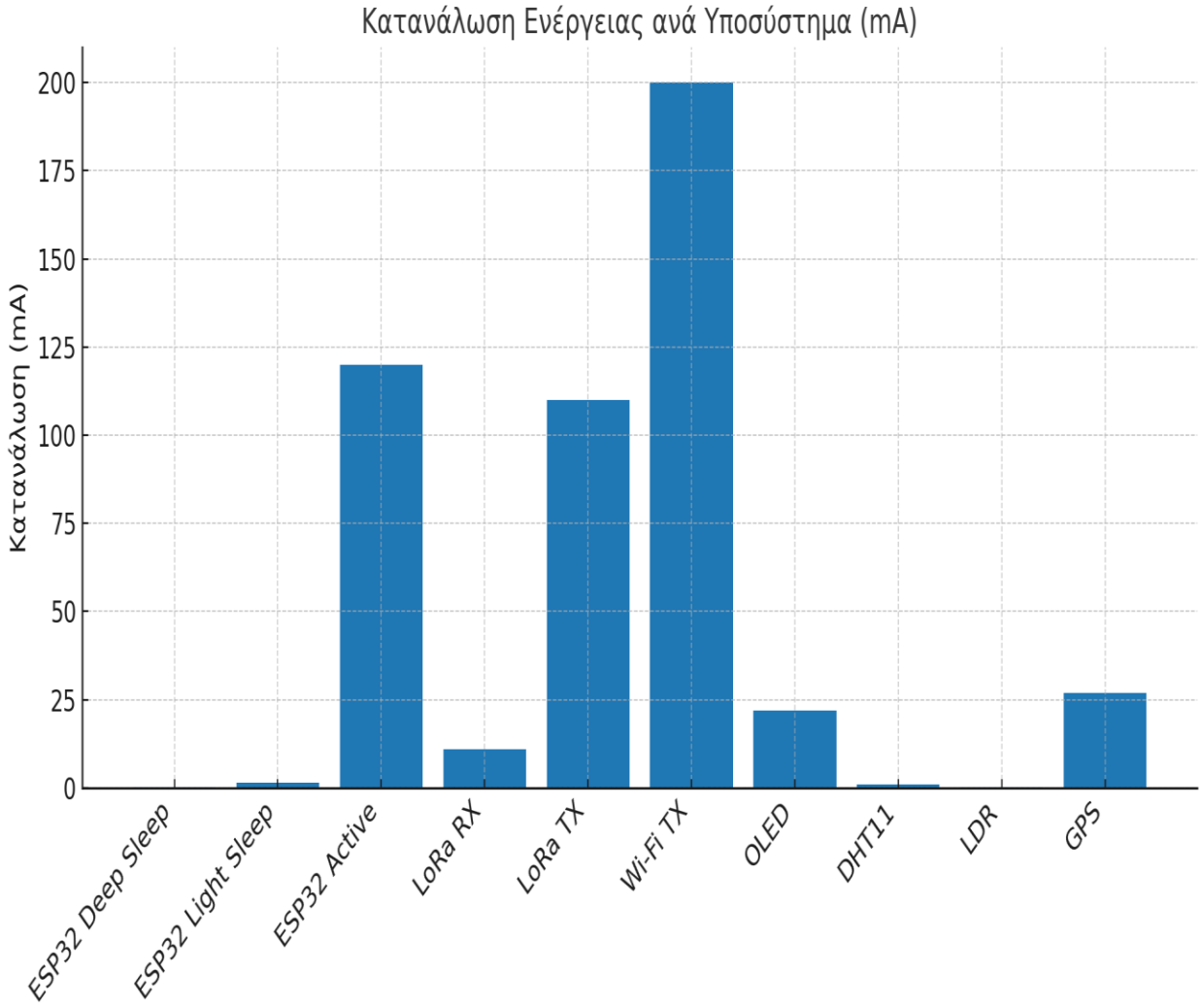
Πίνακας 2.6 Καταναλώσεις του συστήματος TTGO LORA32 OLED V1 και των περιφερειακών

Υποσύστημα / Λειτουργία	**Κατανάλωση ρεύματος (mA)**	**Περιγραφή**
ESP32 σε Deep Sleep	0.01 – 0.15	Ελάχιστη κατανάλωση, ο επεξεργαστής είναι απενεργοποιημένος και μόνο ο χρονοδιακόπτης ή ο αισθητήρας αφύπνισης παραμένει ενεργός.
ESP32 σε Light Sleep	0.8 – 2	Χαμηλή κατανάλωση, μερικές περιφερειακές μονάδες παραμένουν ενεργές (π.χ. RAM, Wi-Fi σε αναμονή).
ESP32 σε Active Mode (CPU 240 MHz)	80 – 160	Κανονική λειτουργία του επεξεργαστή, εκτέλεση προγραμμάτων, Wi-Fi ή Bluetooth ενεργό.
LoRa λήψη (RX)	10 – 12	Λειτουργία δέκτη LoRa, χαμηλή κατανάλωση καθώς μόνο το κύκλωμα λήψης είναι ενεργό.
LoRa μετάδοση (TX, 17 dBm)	100 – 120	Εκπομπή δεδομένων μέσω LoRa σε ισχύ +17 dBm. Μία από τις λειτουργίες με αυξημένη κατανάλωση.
Wi-Fi σύνδεση / μετάδοση	150 – 250	Ενεργή ασύρματη επικοινωνία Wi-Fi. Η πιο μεγάλη κατανάλωση
OLED οθόνη 0.96”	20 – 25	Ενεργή εμφάνιση δεδομένων σε οθόνη OLED μέσω διαύλου I^2C.
Αισθητήρας DHT11 (θερμοκρασίας/υγρασίας)	~1 (ενεργός), 0.1 (σε αναμονή)	Καταναλώνει ελάχιστο ρεύμα, κατάλληλος για χαμηλής ισχύος εφαρμογές.
Αισθητήρας LDR (φωτοαντίσταση)	<0.5 (παθητική κατανάλωση)	Δεν απαιτεί εξωτερική τροφοδοσία· η κατανάλωση εξαρτάται από τη φωτεινότητα.
GPS Module (ενεργό)	25 – 30	Ενεργός δέκτης GPS, σε λειτουργία λήψης δορυφορικών σημάτων στη ζώνη L1.

Ο αισθητήρας **LDR** (Light Dependent Resistor) είναι παθητική διάταξη, η οποία δεν καταναλώνει ουσιαστικά ρεύμα από μόνη της. Η κατανάλωση της περιορίζεται σε μA, που εξαρτάται από την τιμή της αντίστασης και του διαιρέτης τάσης. Έτσι, η συνολική ενεργειακή επιβάρυνση του συστήματος λόγω του LDR είναι πρακτικά αμελητέα.

Το σύστημα που αναπτύχθηκε ενεργοποιείται περιοδικά για να καταγράψει δεδομένα από τους αισθητήρες και να τα αποστείλει μέσω LoRa και Wi-Fi/UDP. Μετά την αποστολή εισέρχεται σε κατάσταση deep sleep. Με αυτόν τον τρόπο, μετά από την πλήρη φόρτιση της μπαταρίας μπορεί να λειτουργήσει αρκετές ημέρες ή και εβδομάδες, ανάλογα με τη χωρητικότητα της μπαταρίας και το διάστημα δειγματοληψίας.

Η ενεργειακή απόδοση και η σωστή διαχείριση τροφοδοσίας αποτελούν κρίσιμο παράγοντα για την πρακτική αξιοποίηση του συστήματος σε απομακρυσμένες περιοχές, όπου δεν υπάρχει σταθερή παροχή ρεύματος και η μπαταρία αποτελεί τη μοναδική πηγή ενέργειας.



Εικόνα 2.11 Ραβδόγραμμα με τις καταναλώσεις ενέργειας ανά υποσύστημα

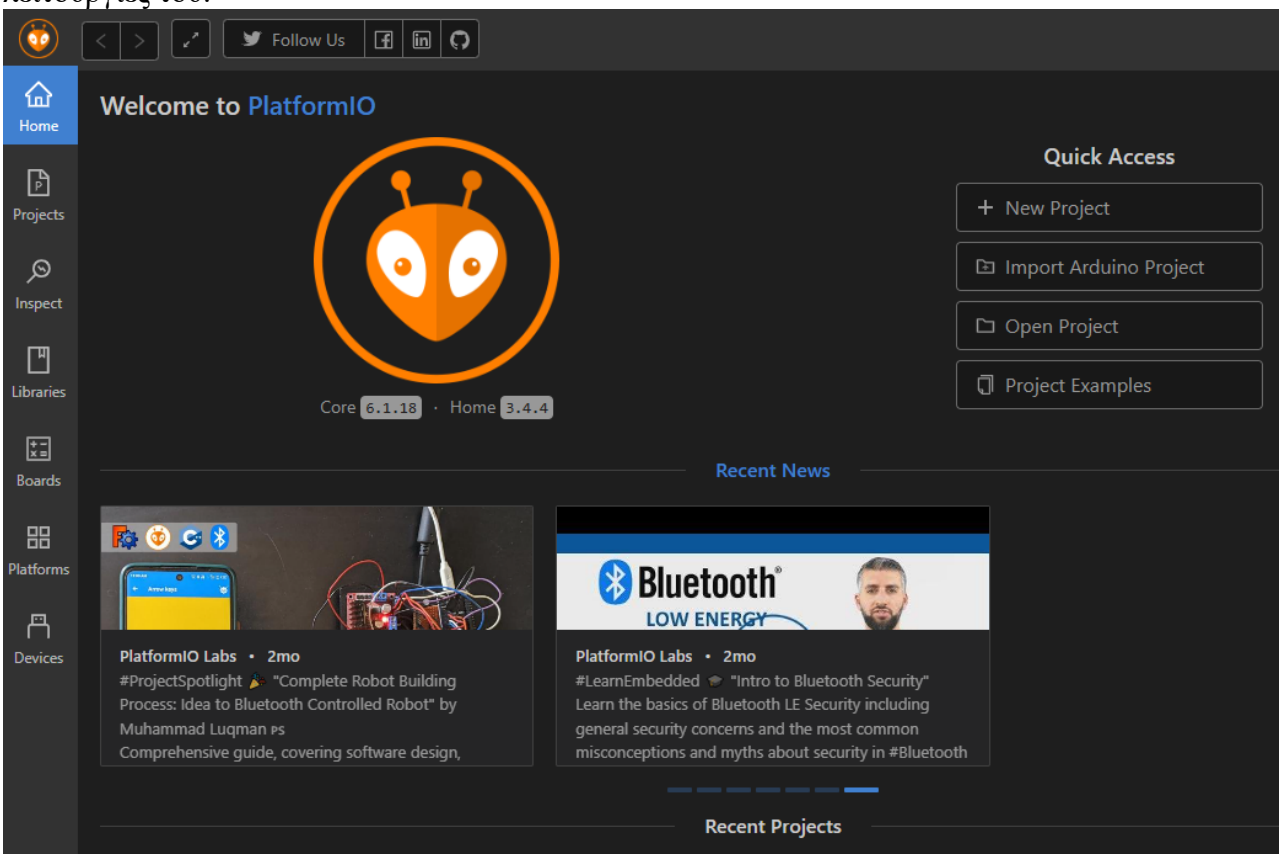
2.2 Περιβάλλον Λογισμικού του Συστήματος

Η υλοποίηση του λογισμικού του συστήματος βασίστηκε σε ένα συνδυασμό εργαλείων προγραμματισμού και περιβαλλόντων ανάπτυξης, τα οποία επιλέχθηκαν με γνώμονα τη λειτουργικότητα, την ευελιξία και την ευκολία χρήσης. Το λογισμικό διαρθρώνεται σε τρία κύρια επίπεδα:

- το **firmware** που εκτελείται στον μικροελεγκτή
- τον **server συλλογής δεδομένων** στον υπολογιστή.
- το **γραφικό περιβάλλον χρήστη (User Interface)** για την παρουσίαση των μετρήσεων.

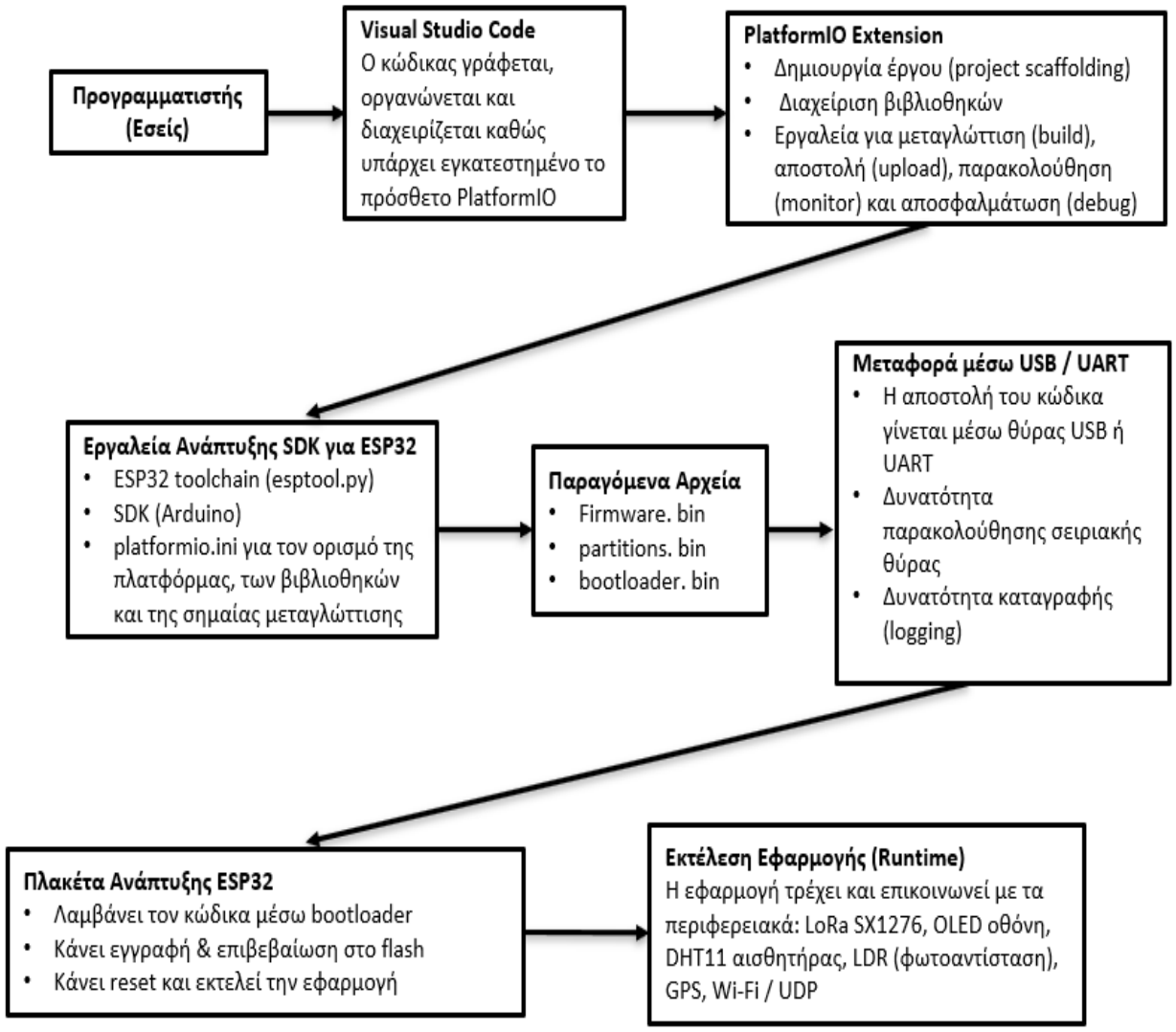
2.2.1 Visual Studio Code – PlatformIO extension

Για την ανάπτυξη του firmware του μικροελεγκτή ESP32 χρησιμοποιήθηκε το αναπτυξιακό περιβάλλον **Visual Studio Code (VS Code)** σε συνδυασμό με το **PlatformIO**. Το **PlatformIO extension** παρέχει ολοκληρωμένες δυνατότητες για ενσωματωμένα συστήματα, όπως αυτοματοποιημένη διαχείριση βιβλιοθηκών, εργαλεία μεταγλώττισης για μικροελεγκτές και υποστήριξη πολλών περιφερειακών, μεταξύ των οποίων και το TTGO LORA32 OLED. Στην εικόνα 2.12 βλέπουμε το πρόσθετο αναπτυξιακό περιβάλλον PlatfomIO και κάποιες από τις λειτουργίες του.



Εικόνα 2.12 PlatformIO extension στο περιβάλλον Visual Code Studio

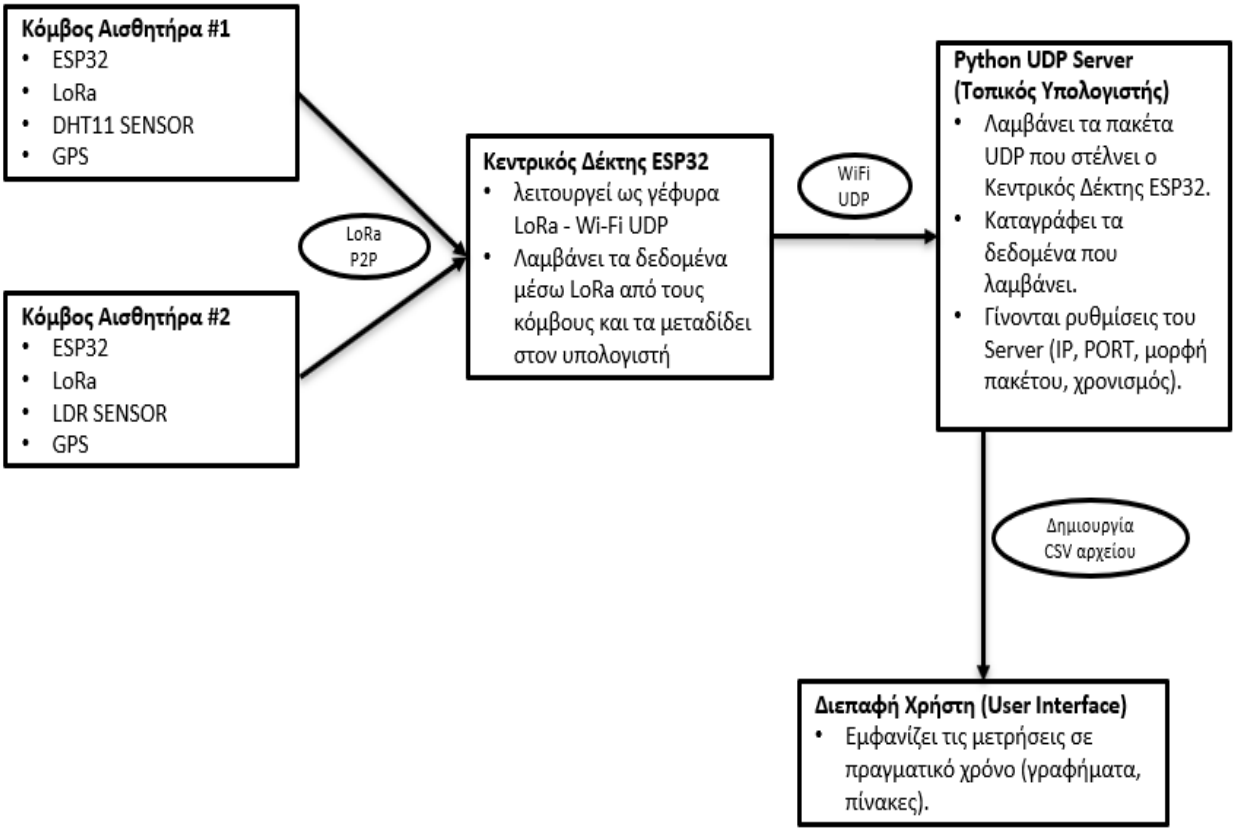
Η επιλογή του συγκεκριμένου αναπτυξιακού περιβάλλοντος προσφέρει ένα ενιαίο οικοσύστημα όπου ο προγραμματιστής μπορεί να γράψει, να μεταγλωττίσει και να φορτώσει τον κώδικα στον μικροελεγκτή, μειώνοντας σημαντικά τον χρόνο ανάπτυξης και αποσφαλμάτωσης. Στην εικόνα 2.13 βλέπουμε τη ροή ανάπτυξης και εκτέλεσης εφαρμογών ESP32 με το PlatformIO.



Εικόνα 2.13 Ροή ανάπτυξης και εκτέλεσης εφαρμογών ESP32 με PlatformIO

2.2.2 Python UDP Server

Για τη συλλογή και επεξεργασία των δεδομένων που μεταδίδονται από τον πομπό του ESP32 αναπτύχθηκε ένας **UDP server** σε γλώσσα προγραμματισμού **Python**. Η Python επιλέχθηκε διότι συνδυάζει απλότητα σύνταξης, υψηλή αναγνωσιμότητα και πληθώρα διαθέσιμων βιβλιοθηκών για δικτυακές επικοινωνίες, επεξεργασία δεδομένων και οπτικοποίηση. Ο server εκτελείται τοπικά στον υπολογιστή και έχει ως κύριο ρόλο να λαμβάνει πακέτα UDP, να τα αποκωδικοποιεί και να τα αποθηκεύει σε αρχεία, ώστε να είναι διαθέσιμα για περαιτέρω ανάλυση ή για ζωντανή παρουσίαση μέσω του User Interface. Το UDP (User Datagram Protocol) προσφέρει πλεονεκτήματα στην παρούσα εφαρμογή, καθώς πρόκειται για ένα ελαφρύ και γρήγορο πρωτόκολλο σε σύγκριση με το TCP/IP, κατάλληλο για εφαρμογές τηλεμετρίας όπου η ταχύτητα προέχει έναντι της απόλυτης αξιοπιστίας στη μετάδοση. Στην εικόνα 2.14 βλέπουμε το διάγραμμα ροής του συστήματος που αναπτύχθηκε και την λειτουργία του Python UDP Server

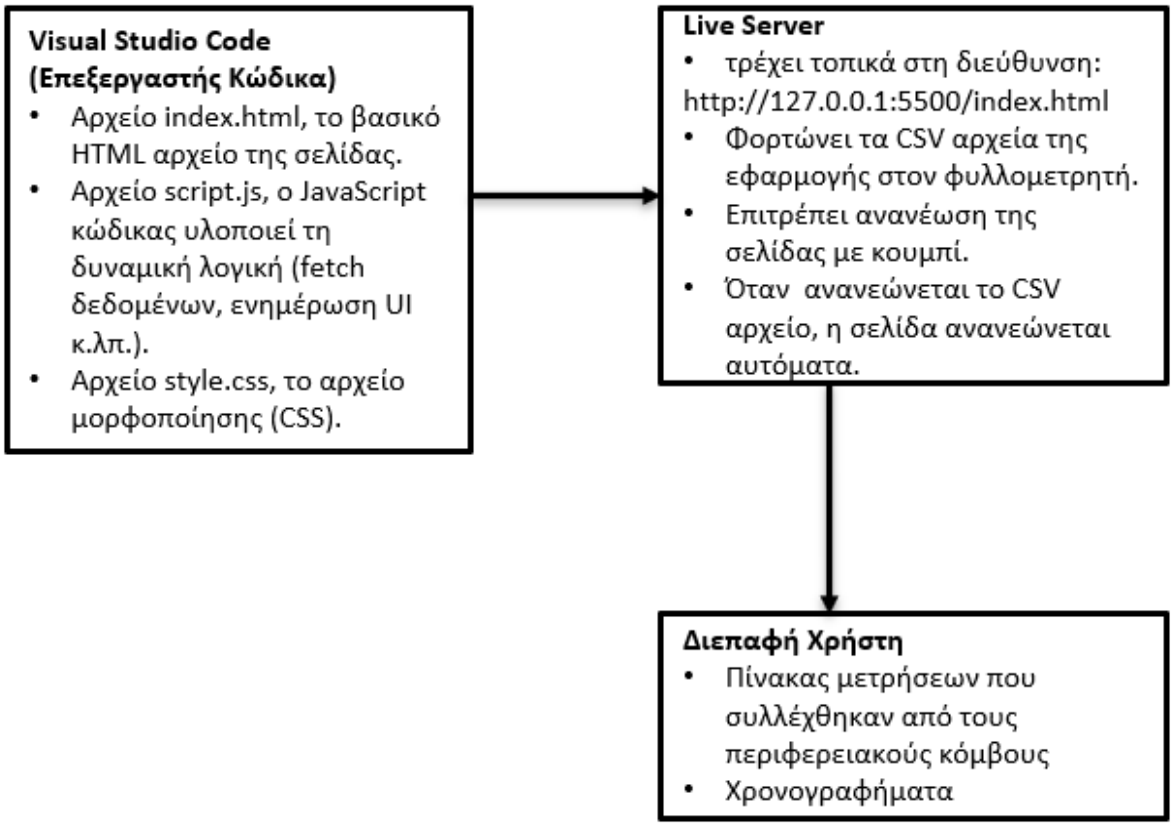


Εικόνα 2.14 Το διάγραμμα ροής του συστήματος που αναπτύχθηκε και η λειτουργία του Python UDP Server

2.2.3 Διεπαφή χρήστη με χρήση Visual Studio Code – Live Server extension

Η διεπαφή χρήστη (User Interface) αναπτύχθηκε με τις βασικές τεχνολογίες του ιστού (**HTML, CSS και JavaScript**), ώστε να είναι φορητή, ελαφριά και ανεξάρτητη από λειτουργικό σύστημα. Η ανάπτυξη έγινε μέσα από το **Visual Studio Code** [36], ενώ για την τοπική εκτέλεση και τη συνεχή προεπισκόπηση χρησιμοποιήθηκε το πρόσθετο **Live Server** [37]. Το τελευταίο επιτρέπει την άμεση ανανέωση της ιστοσελίδας με κάθε αλλαγή στον κώδικα, επιταχύνοντας τη διαδικασία δοκιμής και βελτιστοποίησης του περιβάλλοντος.

Μέσω της διεπαφής, ο τελικός χρήστης μπορεί να παρακολουθεί σε πραγματικό χρόνο τις μετρήσεις που λαμβάνονται από τους κόμβους, τόσο σε μορφή πινάκων όσο και σε γραφήματα, ενώ υπάρχει και η δυνατότητα χωρικής απεικόνισης μέσω χαρτών. Η επιλογή ανάπτυξης με τεχνολογίες ιστού προσφέρει ευκολία πρόσβασης, καθώς η διεπαφή χρήστη μπορεί να εκτελεστεί σε οποιονδήποτε φυλλομετρητή (browser) χωρίς ανάγκη εγκατάστασης επιπλέον λογισμικού. Στην εικόνα 2.17 φαίνεται η αρχιτεκτονική της διεπαφής χρήστη με VS code και Live Server.



Εικόνα 2.17 Η αρχιτεκτονική της διεπαφής χρήστη με VS Code και Live Server

3. Σχεδίαση του Συστήματος

Η σχεδίαση του ασύρματου κατανεμημένου συστήματος αυτοματισμού IoT απαιτεί την ανάλυση τόσο της αρχιτεκτονικής όσο και της λειτουργικής ροής της πληροφορίας, προκειμένου να εξασφαλιστεί η αξιοπιστία, η αποδοτικότητα και η επεκτασιμότητα της λύσης. Η διαδικασία περιλαμβάνει τον καθορισμό των επιμέρους υπομονάδων, τον τρόπο αλληλεπίδρασής τους, καθώς και την ενσωμάτωση τεχνολογιών επικοινωνίας χαμηλής κατανάλωσης και μεγάλης εμβέλειας.

3.1 Αρχιτεκτονική Συστήματος

Το σύστημα αποτελείται από τρία βασικά μέρη: τους κόμβους μέτρησης, την κεντρική μονάδα λήψης και τον υπολογιστή που αποθηκεύει και προβάλει τα δεδομένα.

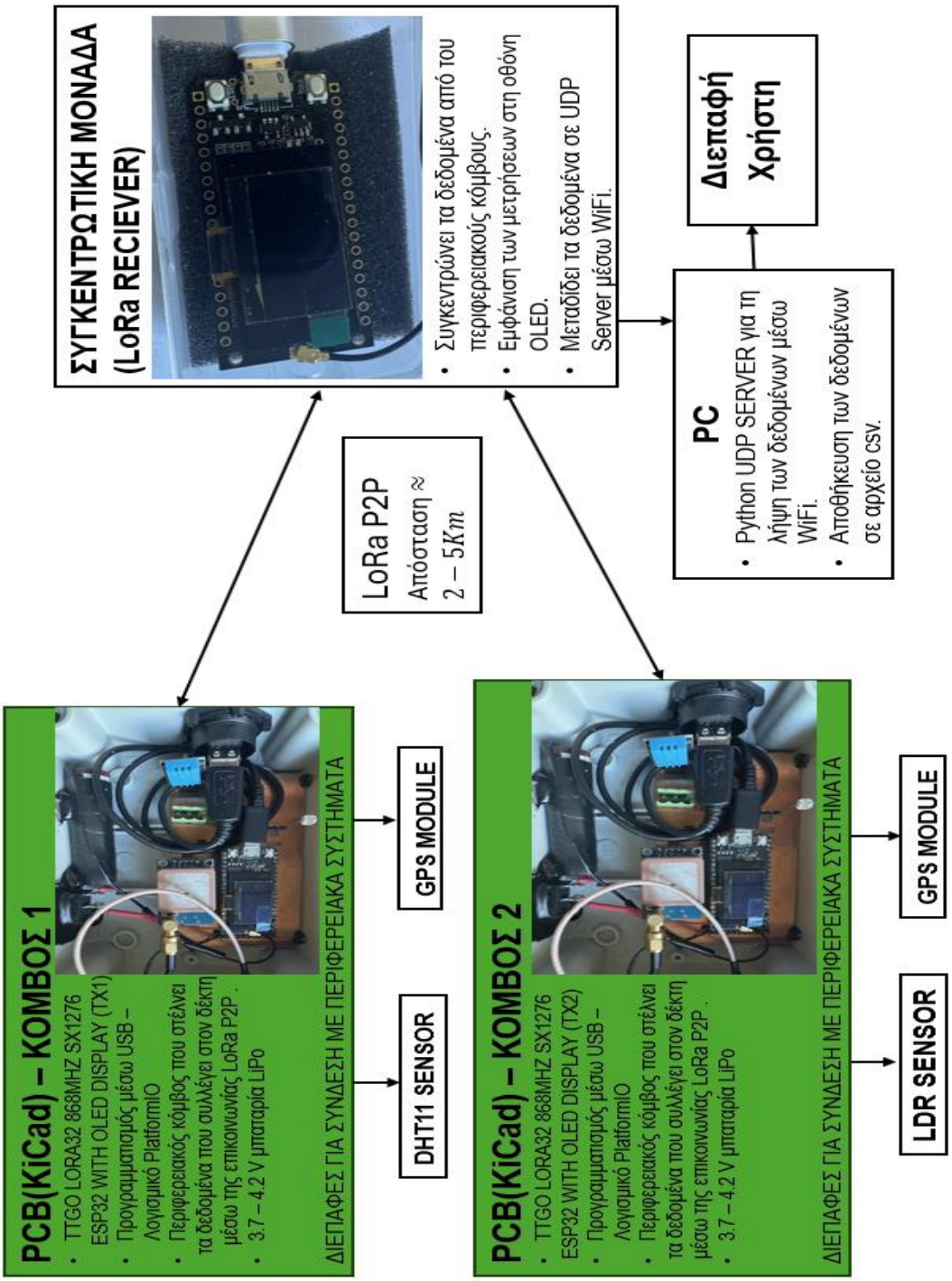
Οι κόμβοι μέτρησης είναι μικρές αυτόνομες συσκευές που λειτουργούν με μπαταρία. Κάθε κόμβος είναι εξοπλισμένος με αισθητήρες και με GPS για να καταγράφει τον ακριβή χρόνο και τη θέση κάθε μέτρησης. Ο μικροελεγκτής ESP32 του κόμβου φροντίζει να διαβάζει τις τιμές, να τις πακετάρει σε μικρά πακέτα δεδομένων και να τα στέλνει ασύρματα στην κεντρική μονάδα μέσω του πρωτοκόλλου LoRa. Το πρωτόκολλο LoRa επιλέχθηκε γιατί μπορεί να μεταδώσει δεδομένα σε μεγάλες αποστάσεις με χαμηλή κατανάλωση ενέργειας.

Η κεντρική μονάδα συγκεντρώνει όλα τα πακέτα που λαμβάνει από τους κόμβους, ελέγχει ότι τα δεδομένα είναι σωστά και τα εμφανίζει σε μία μικρή οθόνη OLED. Στη συνέχεια, χρησιμοποιεί Wi-Fi για να τα στείλει γρήγορα και με απλό τρόπο στον υπολογιστή, μέσω του πρωτοκόλλου UDP. Η χρήση του UDP εξασφαλίζει ότι η μετάδοση γίνεται χωρίς καθυστερήσεις, κάτι που είναι πολύ σημαντικό για εφαρμογές παρακολούθησης σε πραγματικό χρόνο.

Τέλος, στον υπολογιστή τρέχει ένας UDP server που αποθηκεύει τα δεδομένα σε αρχεία CSV και τα εμφανίζει μέσα από μια διεπαφή χρήστη που έχει φτιαχτεί σε HTML και JavaScript. Ο χρήστης μπορεί να βλέπει τις μετρήσεις σε γραφήματα, να φιλτράρει τα δεδομένα και να παρακολουθεί τις αλλαγές σε πραγματικό χρόνο. Με αυτόν τον τρόπο, το σύστημα προσφέρει μια ολοκληρωμένη λύση όπου οι κόμβοι συλλέγουν τις πληροφορίες, η κεντρική μονάδα τις συγκεντρώνει και τις μεταφέρει στον υπολογιστή ο οποίος τις αποθηκεύει και τις παρουσιάζει στη διεπαφή χρήστη.

Με λίγα λόγια, η αρχιτεκτονική του συστήματος είναι σχεδιασμένη έτσι ώστε το σύστημα να είναι επεκτάσιμο, επιτρέποντας την προσθήκη περισσότερων κόμβων ή περιφερειακών συγκεντρωτικών μονάδων.

Στην εικόνα 3.1 παρουσιάζεται το γενικό σχηματικό διάγραμμα του συστήματος που αναπτύχθηκε.



Εικόνα 3.1 Γενικό σχηματικό διάγραμμα του συστήματος που αναπτύχθηκε

3.2 Λειτουργία του συστήματος

Η λειτουργία του συστήματος ακολουθεί μια ευθύγραμμη ροή: οι περιφερειακοί κόμβοι μετρούν φυσικά μεγέθη και δημιουργούν πακέτα με χρονικό στίγμα/θέση (GPS), τα οποία μεταδίδουν ασύρματα μέσω LoRa στην συγκεντρωτική περιφερειακή μονάδα. Η συγκεντρωτική περιφερειακή μονάδα προβάλλει σύνοψη στην OLED και στέλνει τα δεδομένα σε τοπικό υπολογιστή για αποθήκευση (αρχείο CSV) και απεικόνιση μέσω φυλλομετρητή (HTML/JavaScript/CSS διεπαφή χρήστη). Στην πράξη, η ακολουθία είναι:

1. **Δειγματοληψία** από DHT11 (Θερμοκρασία/Υγρασία) και LDR (ένταση φωτός) + λήψη ώρας και ημερομηνίας (timestamp) από το GPS.
2. **Πακετοποίηση** των μετρήσεων από τον ESP32.
3. **Μετάδοση LoRa (P2P)** προς την κεντρική μονάδα.
4. **Λήψη & αναγνώριση προέλευσης** στην κεντρική μονάδα και **άμεση οπτική ένδειξη** στην OLED (μετρικές/RSSI).
5. **Wi-Fi – UDP** για προώθηση πακέτων στον τοπικό υπολογιστή.
6. **Αποθήκευση CSV & ζωντανή παρουσίαση** (Python UDP server και UI σε HTML/JS/CSS).

Η παραπάνω διαδρομή υλοποιεί την αρχιτεκτονική hub-and-spoke που έχει επιλεγεί για το σύστημα.

Η μετάδοση με το **LoRa P2P** δεν προσφέρει ένα αυστηρά τυποποιημένο επίπεδο «πακέτου» όπως τα πρωτόκολλα TCP/IP, αλλά το φυσικό στρώμα της LoRa δημιουργεί πλαίσια που έχουν ελάχιστη εσωτερική δομή. Συνήθως, η δομή διαμορφώνεται σε επίπεδο εφαρμογής. Στη συγκεκριμένη υλοποίηση, κάθε πακέτο περιλαμβάνει:

- **Πεδίο αναγνωριστικού κόμβου (Node ID)**: μοναδικό για κάθε κόμβο.
- **Χρονικό στίγμα**: παράγεται από το GPS (UTC time) και επιτρέπει συγχρονισμό και αλληλουχία.
- **Μετρήσεις αισθητήρων**: θερμοκρασία (°C), υγρασία (%), ένταση φωτός (raw ADC value).
- **Checksum:** απλός έλεγχος εγκυρότητας για να απορρίπτονται αλλοιωμένα πακέτα.

Η τελική μορφή είναι μια **συμβολοσειρά dataPacket** που δημιουργείται στον κώδικα, δηλαδή ένα **πακέτο δεδομένων (data packet)** σε **μορφή κειμένου (String)**, το οποίο περιέχει συνδυασμένες πληροφορίες (αισθητήρες και κατάσταση συσκευής). Το LoRa transceiver SX1276 που χρησιμοποιεί το σύστημα TTGO πλακέτα αναλαμβάνει να προσθέσει **προοίμιο (preamble)** 12.54ms SF7 και **checksum** ώστε ο δέκτης να μπορεί να αναγνωρίζει την έναρξη και το τέλος κάθε μετάδοσης.

Επίσης το LoRa χρησιμοποιεί **διαμόρφωση με ευρύ φάσμα συχνοτήτων (chirp spread spectrum – CSS)**.

Οι βασικές παράμετροι της επικοινωνίας LoRa που καθορίζονται σε κάθε κόμβο και στην συγκεντρωτική μονάδα είναι:

- **Συχνότητα λειτουργίας** (π.χ. 868 MHz στην Ευρώπη).
- **Spreading Factor (SF)**: καθορίζει το ρυθμό μετάδοσης και την ευαισθησία. Στο σύστημα χρησιμοποιήθηκε SF7 (SF7 → γρηγορότερη μετάδοση, SF12 → μεγαλύτερη εμβέλεια αλλά πιο αργή μετάδοση).
- **Bandwidth (BW)**: εύρος φάσματος (250 kHz).

- **Coding Rate (CR)**: πλεονασμός για διόρθωση σφαλμάτων (συνήθως 4/5 έως 4/8).
- **Output Power**: ισχύς εκπομπής (14 dBm).

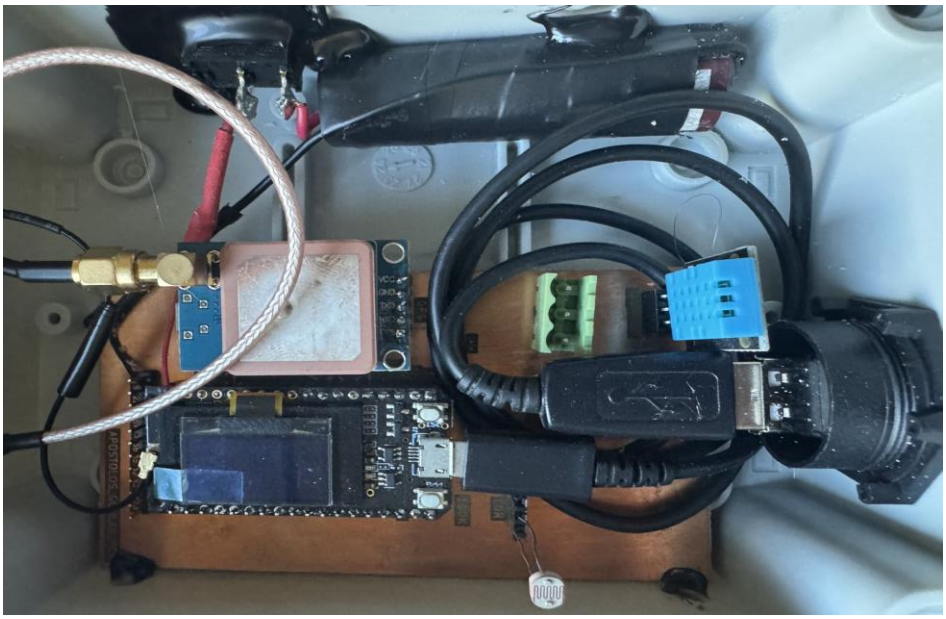
Η σωστή επιλογή των παραμέτρων αυτών καθορίζει την ισορροπία μεταξύ **αξιοπιστίας, κατανάλωσης ενέργειας και εμβέλειας**..

Τέλος η κεντρική μονάδα έχει άμεση επικοινωνία μέσω Wi-Fi με τον υπολογιστή στον οποίο εκτελείται ο **Python UDP server** που έχει οριστεί σε συγκεκριμένη διεύθυνση IP - PORT, παραλαμβάνει τα πακέτα, τα αποκωδικοποιεί και τα **αποθηκεύει σε αρχείο CSV**, το οποίο τροφοδοτεί το **Web UI** (HTML/JavaScript) για ζωντανά γραφήματα και φιλτράρισμα. Έτσι κλείνει ο βρόχος από τη δειγματοληψία μέχρι την παρουσίαση στον χρήστη.

3.3 Περιγραφή των Υπομονάδων

3.3.1 Περιφερειακές Συσκευές

Οι κόμβοι των αισθητήρων που αναπτύχθηκαν βασίζονται στο **ESP32 TTGO LoRa32 OLED V1**, μια πλακέτα που συνδυάζει μικροελεγκτή, LoRa module και οθόνη OLED. Οι κόμβοι είναι εξοπλισμένοι με αισθητήρες DHT11(υγρασίας, θερμοκρασίας), αισθητήρες LDR (ένταση φωτός) και μονάδες GPS. Στην εικόνα 3.2 φαίνεται η φωτογραφίας ενός κόμβου αισθητήρων του συστήματος.



Εικόνα 3.2 Φωτογραφία ενός κόμβου αισθητήρων

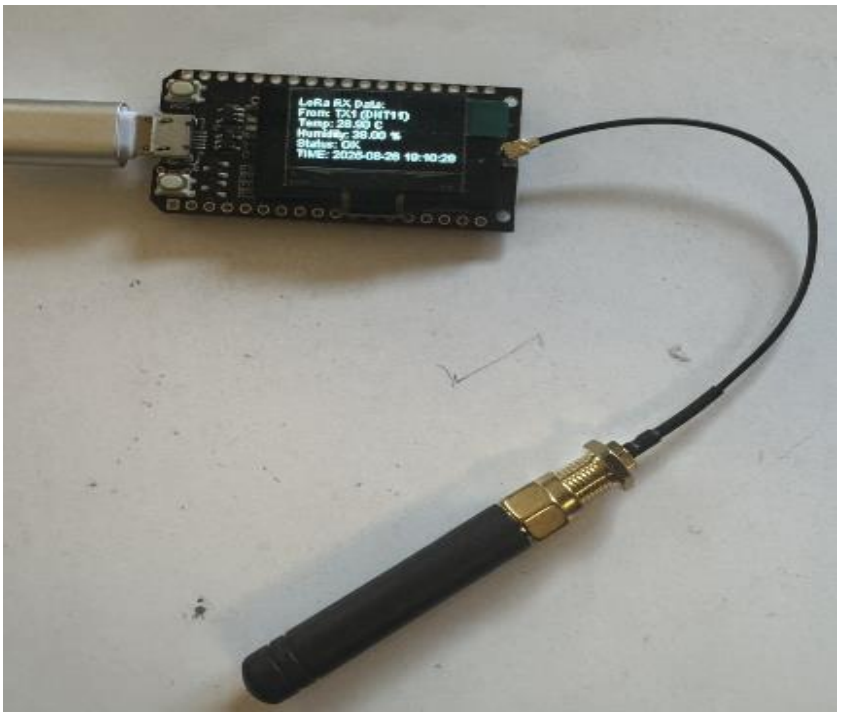
Η λειτουργία τους βασίζεται στη συλλογή περιβαλλοντικών δεδομένων με αισθητήρες σε καθορισμένα χρονικά διαστήματα, τα οποία πακετάρονται σε πλαίσια LoRa και μεταδίδονται ασύρματα προς την κεντρική μονάδα λήψης. Ο συνδυασμός των αισθητήρων με τη μονάδα GPS επιτρέπει την ακριβή ταυτοποίηση κάθε μέτρησης με συγκεκριμένο χρόνο και θέση, κάτι που καθιστά το σύστημα κατάλληλο για εφαρμογές παρακολούθησης σε απομακρυσμένες περιοχές.

3.3.2 Κεντρική Μονάδα Λήψης

Η περιφερειακή συγκεντρωτική μονάδα βασίζεται επίσης στο σύστημα ESP32 TTGO LoRa32 OLED. Ο ρόλος της περιλαμβάνει:

1. Τη λήψη πακέτων LoRa από πολλαπλούς κόμβους.
2. Τον διαχωρισμό της προέλευσης με βάση μοναδικά IDs που περιέχονται στα πακέτα.
3. Την εμφάνιση βασικών πληροφοριών στην ενσωματωμένη OLED (όπως θερμοκρασία, υγρασία, τιμή τουLDR, χρονικό στίγμα και κατάσταση κόμβου).
4. Την αναμετάδοση των δεδομένων μέσω **Wi-Fi**, χρησιμοποιώντας το **πρωτόκολλο UDP**, προς τον τοπικό υπολογιστή.

Με αυτόν τον τρόπο, η κεντρική μονάδα λειτουργεί ως γέφυρα μεταξύ **LoRa** και **Wi-Fi**, αξιοποιώντας την ανθεκτικότητα και την εμβέλεια του LoRa για τη συλλογή δεδομένων, ενώ παράλληλα χρησιμοποιεί την ταχύτητα και τη συμβατότητα του Wi-Fi για την επεξεργασία τους από τον υπολογιστή. Στην εικόνα 3.3 φαίνεται μια φωτογραφία της περιφερειακής συγκεντρωτικής μονάδας.



Εικόνα 3.4 Φωτογραφία της περιφερειακής συγκεντρωτικής μονάδας.

3.3.3 Τοπικός Υπολογιστής και Διεπαφή Χρήστη

Ο τοπικός υπολογιστής αποτελεί το τελικό στάδιο της αλυσίδας ροής πληροφορίας. Εκεί υλοποιείται:

- Ένας **server UDP σε Python**, που λαμβάνει τα πακέτα από την κεντρική μονάδα, τα αποκωδικοποιεί και τα αποθηκεύει σε αρχεία **CSV**. Αυτό επιτρέπει την εύκολη επεξεργασία και την ιστορική καταγραφή των δεδομένων.
- Μία **διεπαφή χρήστη (UI)**, υλοποιημένη σε **HTML, JavaScript και CSS**, η οποία επιτρέπει στον χρήστη να παρακολουθεί τα δεδομένα σε πραγματικό χρόνο μέσω οποιουδήποτε φυλλομετρητή. Η παρουσίαση γίνεται σε μορφή πινάκων και γραφημάτων, διευκολύνοντας την κατανόηση των μετρήσεων και την ανάλυση τους.

Επιπλέον, τα αποθηκευμένα αρχεία CSV μπορούν να αξιοποιηθούν για πιο προχωρημένες αναλύσεις σε εργαλεία όπως το Excel, η Python ή το MATLAB, επεκτείνοντας τις δυνατότητες του συστήματος. Στον πίνακα 3.1 φαίνονται η μετρήσεις που έχουν καταγραφεί στη διεπαφή χρήστη.

Πίνακας 3.1 Εμφάνιση δεδομένων στη Διεπαφή Χρήστη

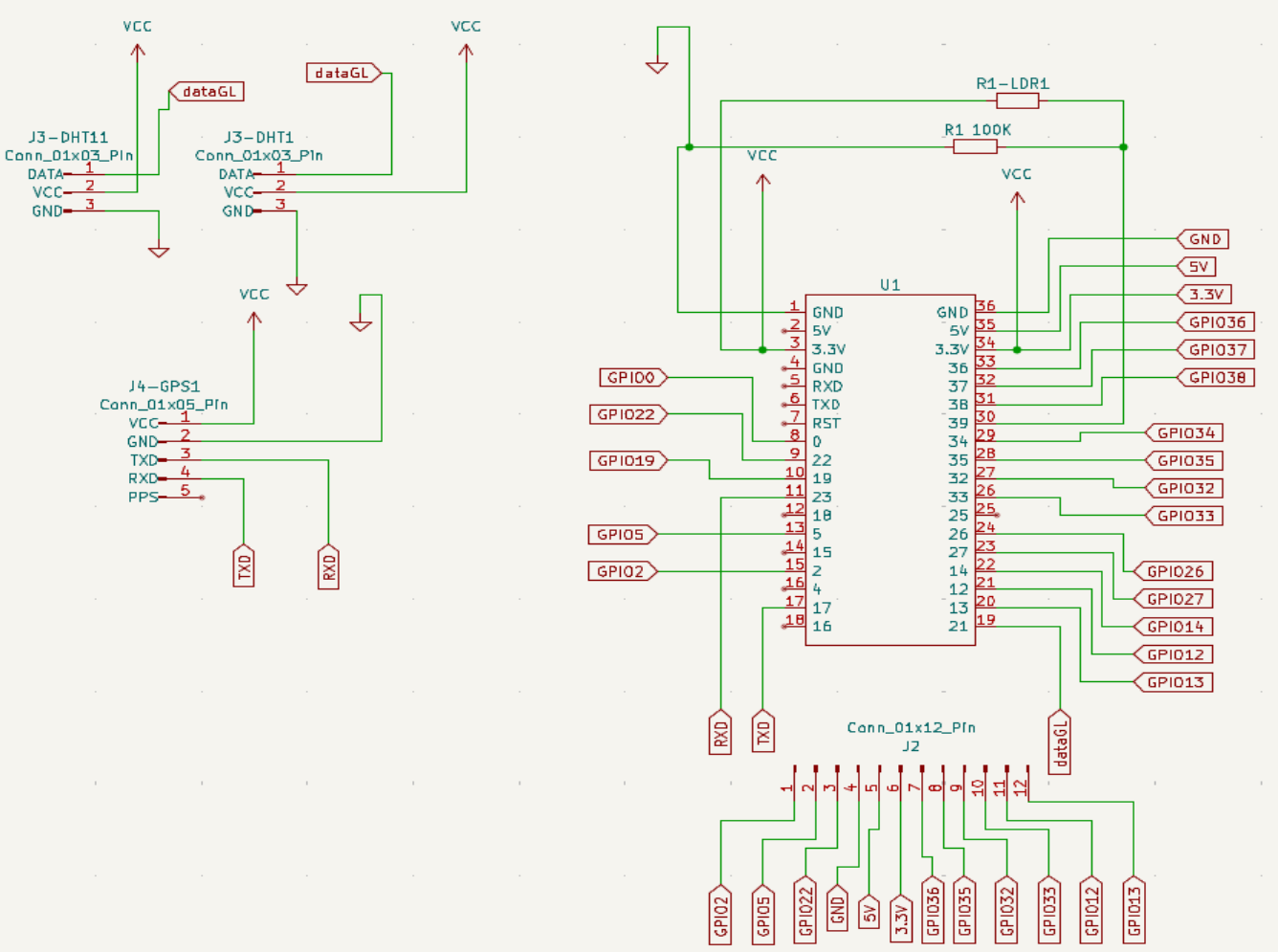
Timestamp	Device_ID	Sensor	Temperature (°C)	Humidity (%)	LDR_Value	BRIGHTNESS (%)	Status
9/15/2025 12:06	TX1	DHT11	27.6	55			OK
9/15/2025 12:06	TX2	LDR			3824	93	HIGH_LIGHT
9/15/2025 12:06	TX1	DHT11	27.6	55			OK
9/15/2025 12:07	TX1	DHT11	28	81			HIGH HUM
9/15/2025 12:07	TX2	LDR			3572	87	HIGH_LIGHT
9/15/2025 12:08	TX1	DHT11	27.6	75			HIGH HUM
9/15/2025 12:08	TX2	LDR			1824	44	OK
9/15/2025 12:08	TX1	DHT11	27.6	63			OK
9/15/2025 12:09	TX2	LDR			1751	42	OK
9/15/2025 12:09	TX1	DHT11	27.6	65			OK
9/15/2025 12:10	TX2	LDR			1812	44	OK
9/15/2025 12:10	TX1	DHT11	27.6	54			OK
9/15/2025 12:10	TX2	LDR			3632	88	HIGH_LIGHT
9/15/2025 12:11	TX1	DHT11	23.4	31			OK
9/15/2025 12:11	TX2	LDR			3621	88	HIGH_LIGHT
9/15/2025 12:11	TX1	DHT11	19	32			OK
9/15/2025 12:11	TX2	LDR			3662	89	HIGH_LIGHT
9/15/2025 12:12	TX1	DHT11	15.2	34			OK
9/15/2025 12:12	TX2	LDR			3651	89	HIGH_LIGHT
9/15/2025 12:13	TX2	LDR			3600	87	HIGH_LIGHT
9/15/2025 12:13	TX2	LDR			3625	88	HIGH_LIGHT
9/15/2025 12:13	TX1	DHT11	10.5	34			LOW TEMP
9/15/2025 12:14	TX1	DHT11	8.9	37			LOW TEMP
9/15/2025 12:14	TX2	LDR			4095	100	HIGH_LIGHT

4. Υλοποίηση

4.1 Κατασκευή πλακέτας με το πρόγραμμα KiCad

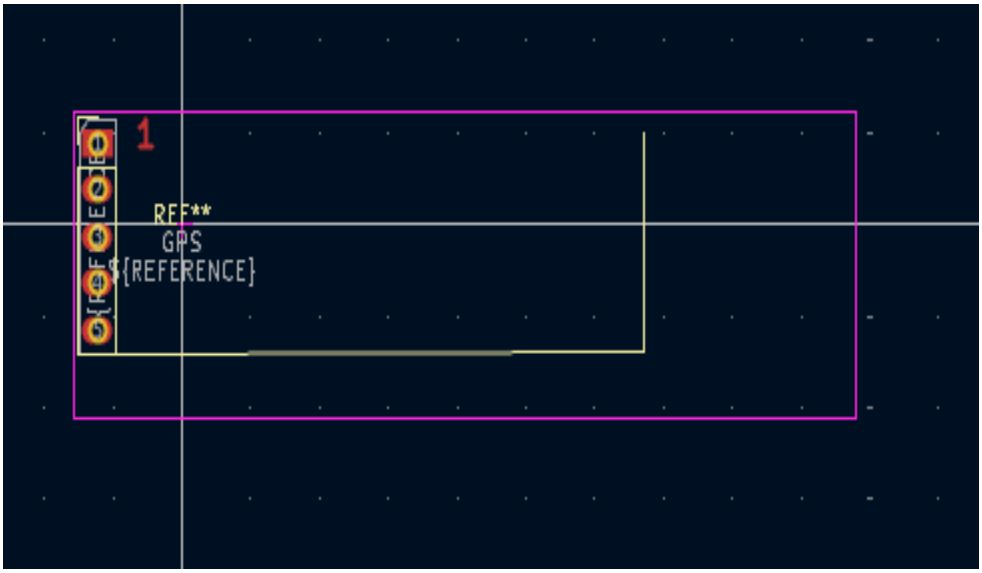
Η σχεδίαση και κατασκευή της πλακέτας τυπωμένου κυκλώματος (Printed Circuit Board – PCB) αποτέλεσε κρίσιμο στάδιο της υλοποίησης, καθώς σε αυτήν συγκεντρώνονται και διασυνδέονται όλα τα υποσυστήματα του ασύρματου κατανεμημένου συστήματος. Για τον σκοπό αυτό χρησιμοποιήθηκε το λογισμικό **KiCad 9.0**, το οποίο παρέχει ολοκληρωμένο περιβάλλον ανάπτυξης από το σχηματικό διάγραμμα έως την παραγωγή των αρχείων κατασκευής.

Το **σχηματικό διάγραμμα** όπως φαίνεται στην εικόνα 4.1 αποτυπώνει τις βασικές λειτουργικές ενότητες του συστήματος των αισθητήρων. Κεντρικό ρόλο κατέχει ο μικροελεγκτής **ESP32 TTGO LoRa32 OLED (U1)**, ο οποίος συνδέεται με τους αισθητήρες θερμοκρασίας και υγρασίας **DHT11 (J3)**, φωτεινότητας μέσω φωτοαντίστασης **LDR (R1)** και το **GPS module (J4)** για τον καθορισμό χωρικού και χρονικού στίγματος. Η τροφοδοσία της πλακέτας υποστηρίζεται από γραμμές ισχύος που υλοποιήθηκαν με κατάλληλους ακροδέκτες ώστε να εξασφαλίζεται σταθερή λειτουργία σε επίπεδα τάσης 3.3 V.

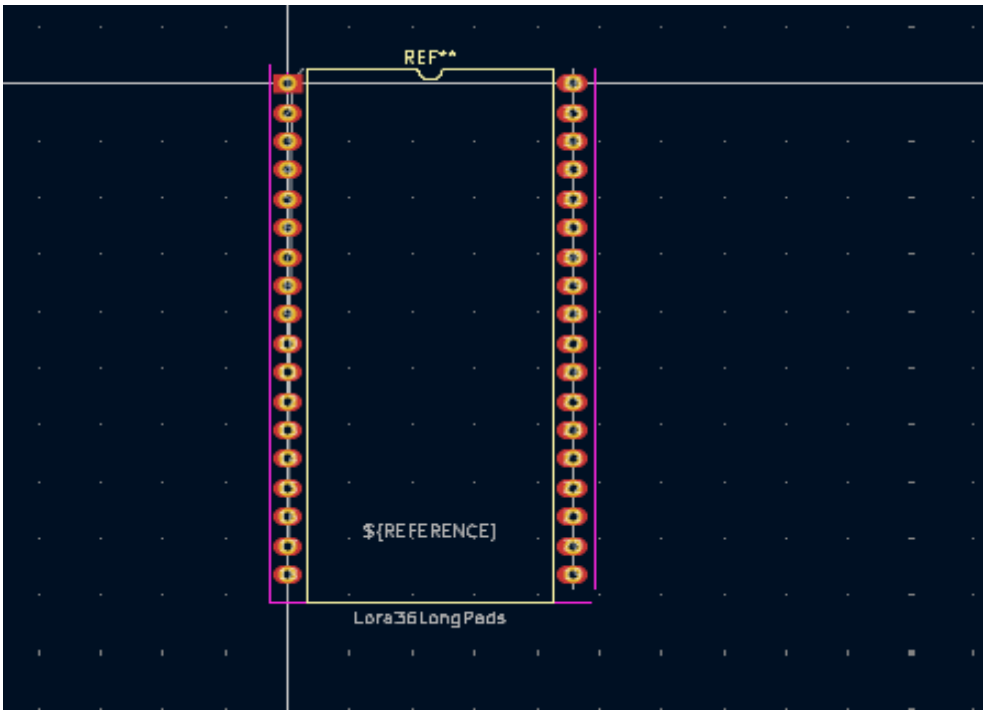


Εικόνα 4.1 Το σχηματικό για την κατασκευή της πλακέτας

Όλα τα εξαρτήματα αντιστοιχίστηκαν σε κατάλληλα αποτυπώματα (footprints) από τις βιβλιοθήκες του KiCad, ενώ όπου απαιτήθηκε δημιουργήθηκαν νέες καταχωρήσεις, ώστε να εξασφαλιστεί πλήρης αντιστοιχία με τα φυσικά εξαρτήματα. Στις εικόνες 4.2, 4.3 φαίνονται τα αποτυπώματα που δημιουργήθηκαν για την μονάδα GPS και και για τον σύστημα TTGO LORA32 OLEDV1 αντίστοιχα

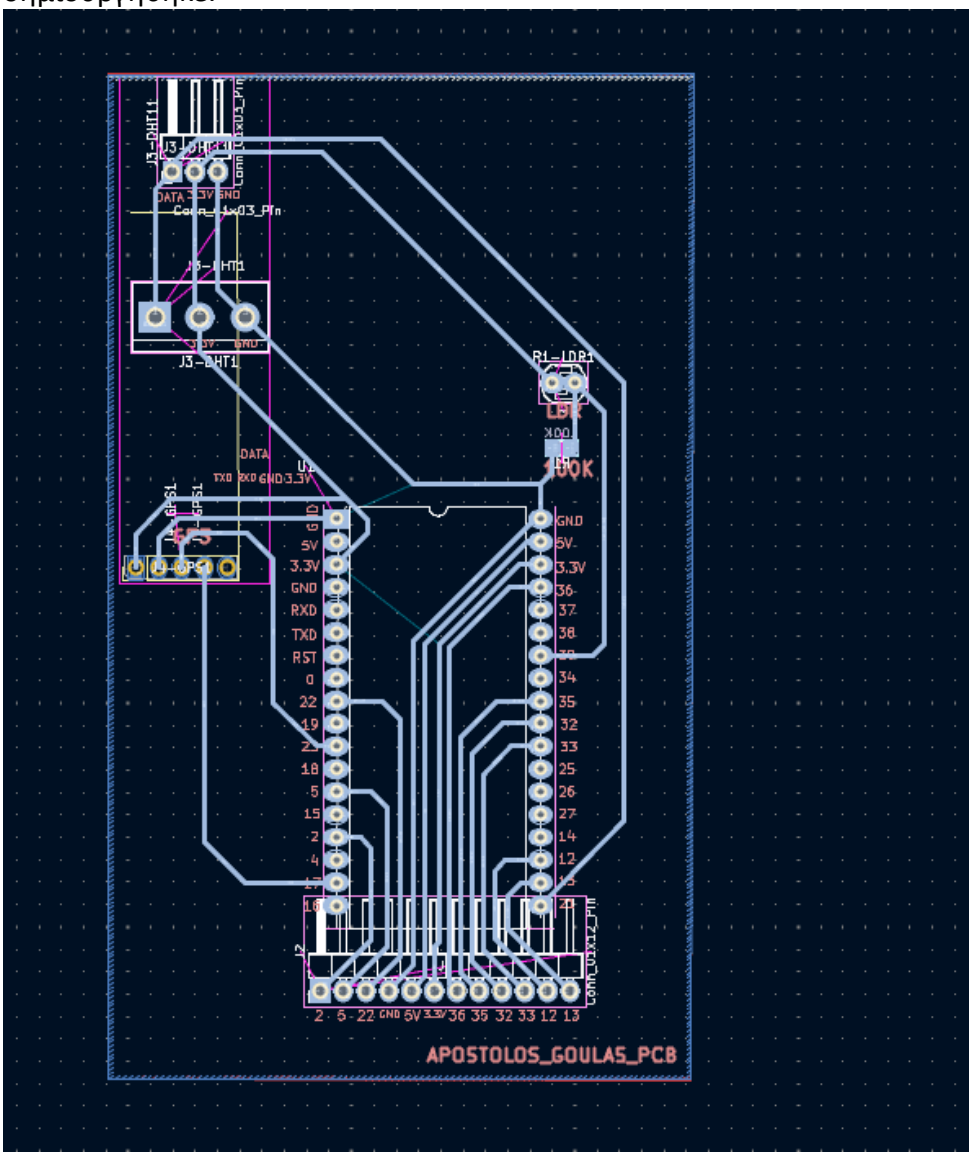


Εικόνα 4.2 Δημιουργία footprint για μονάδα GPS Module



Εικόνα 4.3 Δημιουργία footprint για το σύστημα TTGO LORA32 OLEDV1

Η **σχεδίαση του PCB** πραγματοποιήθηκε σε διπλής όψης πλακέτα, δηλαδή με δύο επίπεδα χαλκού και τυπικό πάχος 1.6 mm. Οι διαστάσεις και η γεωμετρία της πλακέτας καθορίστηκαν λαμβάνοντας υπόψη την τοποθέτηση σε προστατευτικό περίβλημα. Ιδιαίτερη προσοχή δόθηκε στην εργονομική τοποθέτηση των συνδέσμων (headers), ώστε να είναι δυνατή η εύκολη πρόσβαση στην κεραία LoRa και στις θύρες επικοινωνίας. Οι διαδρομές χαλκού (routing) πάχους 20 mils σχεδιάστηκαν με γνώμονα την ελαχιστοποίηση των ηλεκτρομαγνητικών παρεμβολών, με σύντομες και ευθείες διαδρομές για τα κρίσιμα σήματα, ενώ για τις γραμμές τροφοδοσίας εφαρμόστηκαν κατάλληλα πλάτη αγωγών για μείωση της πτώσης τάσης. Η ορθότητα της σχεδίασης επαληθεύτηκε μέσω των εργαλείων **Electrical Rules Check (ERC)** και **Design Rules Check (DRC)**, τα οποία επιβεβαίωσαν την απουσία σφαλμάτων τόσο στο σχηματικό όσο και στο layout. Στην εικόνα 4.4 φαίνεται το σχέδιο PCB της πλακέτας του συστήματος που δημιουργήθηκε.



Εικόνα 4.4 Το σχέδιο PCB της πλακέτας του συστήματος.

Από το τελικό σχέδιο εξήχθησαν τα **αρχεία Gerber**, μαζί με τα αρχεία διάτρησης (drill files) και τη λίστα υλικών (Bill of Materials – BOM). Χρησιμοποιήθηκε για την εκτύπωση της πλακέτας ο εξοπλισμός του εργαστήριου Φυσικής Υψηλών Ενεργειών του τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων.

Με την ολοκλήρωση της κατασκευής του PCB στο KiCad πραγματοποιήθηκαν οι απαραίτητες τρύπες και κολλήσεις στο χώρο του εν λόγω εργαστηρίου. Επιτεύχθηκε μια **συνεκτική και τεχνικά αξιόπιστη υλοποίηση**, η οποία ενσωματώνει όλα τα απαραίτητα υποσυστήματα σε μία ενιαία πλατφόρμα. Η διαδικασία αυτή συνέβαλε στη μετάβαση από το θεωρητικό σχεδιασμό στη λειτουργική κατασκευή του συστήματος..

Στο **ΠΑΡΑΡΤΗΜΑ Γ** αποτυπώνονται τα σχέδια της πάνω και κάτω όψης της πλακέτας τα οποια δημιουργήθηκαν στο PCB EDITOR του προγράμματος KiCad, τα drill files και οι φωτογραφίες του τυπωμένου κυκλώματος της πλακέτας.

4.2 Τοποθέτηση των πλακετών σε προστατευτικά κουτία IP65

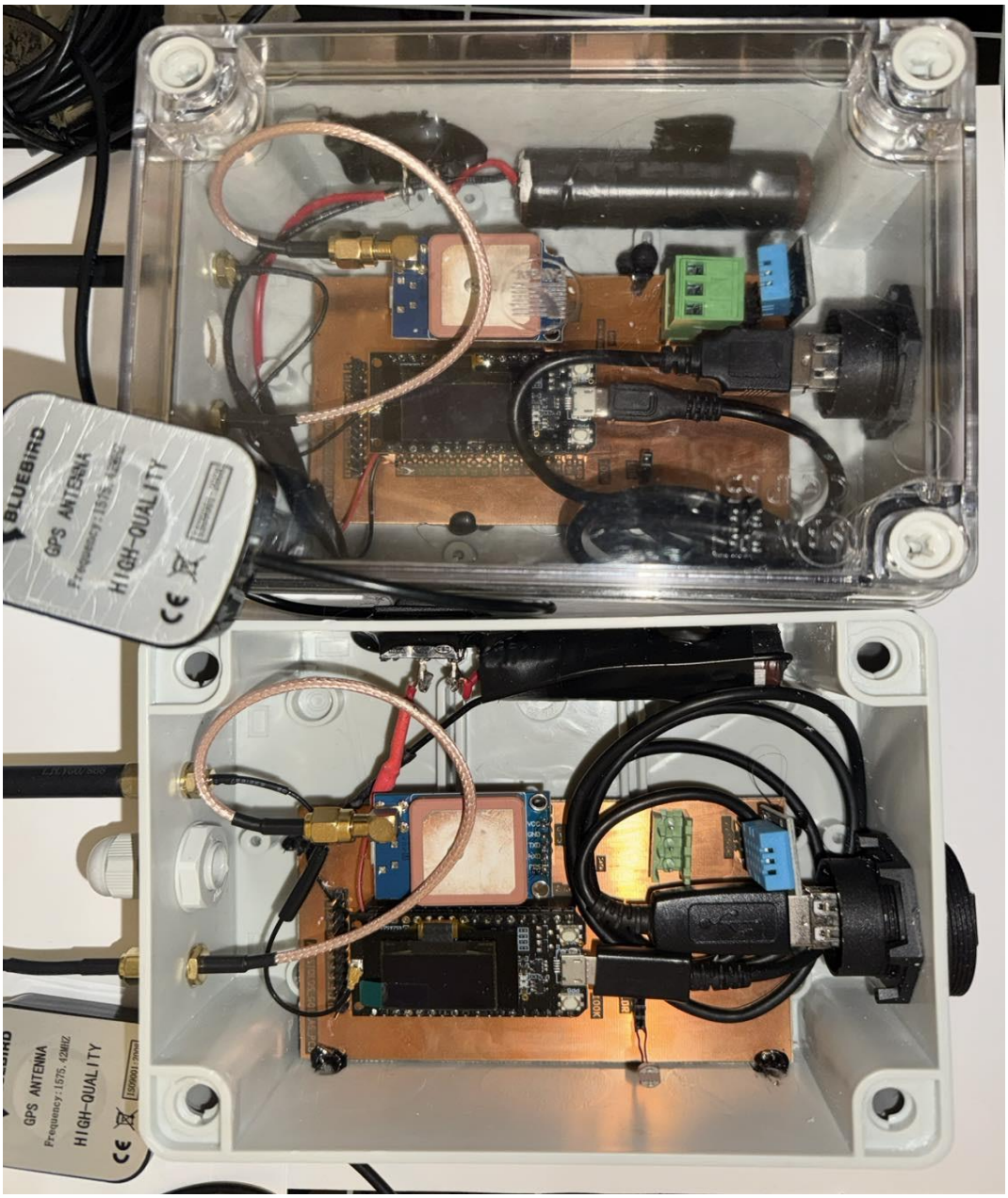
Οι πλακέτες που κατασκευάστηκαν τοποθετήθηκαν σε προστατευτικά κουτιά με βαθμό στεγανότητας IP65. Η πιστοποίηση **IP65** (Ingress Protection) δηλώνει προστασία έναντι της σκόνης και αντοχή σε εκτόξευση νερού χαμηλής πίεσης από οποιαδήποτε κατεύθυνση. Με τον τρόπο αυτό εξασφαλίζεται ότι οι ηλεκτρονικές πλακέτες παραμένουν λειτουργικές ακόμη και σε αντίξοες συνθήκες, όπως υγρασία, βροχή, σκόνη ή έντονη ηλιακή ακτινοβολία.

Κατά την τοποθέτηση των πλακετών στο εσωτερικό των κουτιών λήφθηκαν υπόψη τα ακόλουθα:

- **Σταθερή στήριξη:** Συγκόλληση των πλακετών στις βάσεις των κουτιών ώστε να μην μετακινούνται με τις δονήσεις, αποτρέποντας έτσι βραχυκυκλώματα και μηχανικές καταπονήσεις.
- **Διέλευση καλωδίων:** Τοποθέτηση ειδικών στυπιοθλιπτών (cable glands) για την είσοδο/έξοδο, των καλωδίων τροφοδοσίας, της κεραίας LoRa, της κεραίας του GPS και πιθανών αισθητήρων, χωρίς να χάνεται η στεγανότητα του κουτιού. Επίσης τοποθετήθηκαν διακόπτες ON/OFF για τον έλεγχο της τροφοδοσίας και αναμονές καλωδίων USB για εξωτερική τροφοδοσία, φόρτιση της μπαταρίας και προγραμματισμό.
- **Πρόσβαση για συντήρηση:** Η διάταξη των πλακετών και των συνδέσμων έγινε έτσι ώστε να είναι εύκολη η αφαίρεση του καπακιού και η πρόσβαση στο κύκλωμα χωρίς να απαιτείται αποσυναρμολόγηση.

Η χρήση κουτιών IP65 καθιστά τις συσκευές κατάλληλες για εφαρμογές πεδίου, όπως σε αγροτικές, βιομηχανικές ή αστικές εγκαταστάσεις, όπου οι περιβαλλοντικές συνθήκες δεν είναι ελεγχόμενες. Με αυτό τον τρόπο ενισχύεται σημαντικά η αξιοπιστία του ασύρματου κατανεμημένου συστήματος αυτοματισμού που υλοποιήθηκε.

Στην εικόνα 4.5 παρουσιάζονται τα κουτία IP65 που χρησιμοποιήθηκαν για την ολοκληρωμένη υλοποίηση των περιφερειακών κόμβων του συστήματος.



Εικόνα 4.5 Τα συστήματα των αισθητήρων ολοκληρωμένα εντός των κουτιών

5. Το λογισμικό του συστήματος

5.1 ESP32 TTGO LoRa32 OLED - DHT11 - LDR - GPS

Αναπτύχθηκε λογισμικό στον μικροελεγκτή **ESP32 TTGO LoRa32 OLED** με στόχο την ανάγνωση φυσικών παραμέτρων από τους αισθητήρες. Επίσης το λογισμικό διαβάζει το χρόνο από τη μονάδα GPS και μεταδίδει όλα τα δεδομένα.

Η ανάπτυξη του λογισμικού πραγματοποιήθηκε με το περιβάλλον **PlatformIO** και το framework **Arduino**. Για την υλοποίηση αξιοποιήθηκαν οι βιβλιοθήκες:

- **SSD1306Wire** για την οδήγηση της οθόνης OLED.
- **LoRa** (Sandeep Mistry) για την ασύρματη επικοινωνία στη συχνότητα 868 MHz.
- **Adafruit Unified Sensor** και **DHT** για τη μέτρηση θερμοκρασίας και υγρασίας.
- **TinyGPSPlus** για την αποκωδικοποίηση δεδομένων NMEA από το GPS.
- **Arduino** που είναι βασική βιβλιοθήκη Arduino/ESP32.
- **SPI** για SPI (LoRa).
- **Wire** για I2C OLED.
- **Images** για logo LoRa στην OLED
- **HardwareSerial** για Serial1/Serial2 στον ESP32.
- **esp_sleep** για Deep sleep λειτουργίες στον ESP32.

Η λειτουργία του προγράμματος χωρίζεται στα παρακάτω στάδια:

1. **Αρχικοποίηση συστημάτων**
   - Ρύθμιση σειριακής επικοινωνίας με τη μονάδα GPS στις θύρες RX/TX του ESP32 (pins 23 και 17).
   - Επαναφορά και αρχικοποίηση της οθόνης OLED. Κατά την εκκίνηση εμφανίζεται το λογότυπο που ορίζεται στο αρχείο images.h.
   - Εκκίνηση της διεπαφής LoRa με παραμέτρους που αντιστοιχούν στη ζώνη **868 MHz**, καθώς και αρχικοποίηση των αισθητήρων (DHT11 και LDR) και της μονάδας GPS.
2. **Συλλογή δεδομένων GPS**
   - Το πρόγραμμα διαβάζει τα δεδομένα που αποστέλλει το GPS.
   - Όταν εντοπιστεί ημερομηνία και ώρα, υπολογίζεται η τοπική ώρα βάσει διορθωτικού συντελεστή ζώνης (+3 ώρες).
3. **Συλλογή δεδομένων από τους αισθητήρες**
   - Διαβάζονται οι τιμές θερμοκρασίας, υγρασίας και LDR.
   - Ελέγχεται η εγκυρότητα των μετρήσεων και υπολογίζεται μια συμβολοσειρά **κατάστασης (STATUS)** ανάλογα με τις τιμές των αισθητήρων. Στην εικόνα 5.1 εμφανίζονται αυτές οι καταστάσεις.
4. **Δημιουργία και αποστολή πακέτου δεδομένων**
   - Οι μετρήσεις συγκεντρώνονται σε ένα πακέτο κειμένου το οποίο μεταδίδεται μέσω LoRa. Εάν δεν ληφθεί σήμα επιβεβαίωσης παραλαβής του πακέτου (**ACK)** από τον δέκτη, το σύστημα επαναλαμβάνει τη μετάδοση έως και τρεις φορές.

5. **Εμφάνιση στην οθόνη OLED**
   - Στην οθόνη προβάλλονται τα δεδομένα (τιμές αισθητήρων, κατάσταση, ώρα) και ένδειξη για την επιτυχία ή αποτυχία της λήψης **ACK**.
6. **Εξοικονόμηση ενέργειας**
   - Μετά την αποστολή, η συσκευή τίθεται σε **deep sleep mode** για όποιον επιθυμητό χρόνο. Έτσι, μειώνεται σημαντικά η κατανάλωση ενέργειας, γεγονός κρίσιμο για εφαρμογές απομακρυσμένων κόμβων με τροφοδοσία από μπαταρία

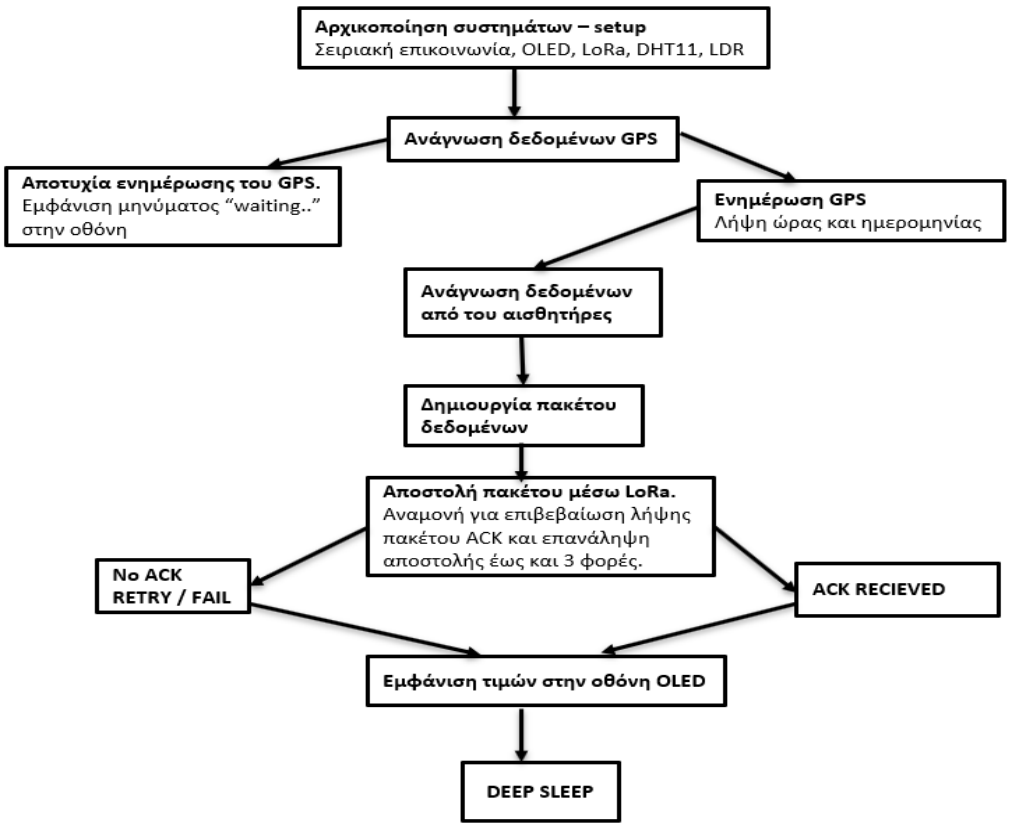
Στην εικόνα 5.2 φαίνεται το διάγραμμα ροής του λογισμικού του συστήματος των περιφερειακών κόμβων.

```
// Define STATUS
String status = "OK";  // Default status

if (isnan(temperature) || isnan(humidity)) {
    status = "ERROR";  // Sensor read failed
} else if (temperature > 30.0) {
    status = "HIGH TEMP";  // Temperature too high
} else if (temperature < 15.0) {
    status = "LOW TEMP";  // Temperature too low
} else if (humidity > 70.0) {
    status = "HIGH HUM";  // Humidity too high
} else if (humidity < 20.0) {
    status = "LOW HUM";  // Humidity too low
}
```

```
// Determine light status
String status = "OK";  // Default status
if (ldrPercent < 20) {
    status = "LOW_LIGHT";
} else if (ldrPercent > 80) {
    status = "HIGH_LIGHT";
}
```

Εικόνα 5.1 Οι καταστάσεις των αισθητήρων



Εικόνα 5.2 Διάγραμμα Ροής του Λογισμικού του συστήματος των περιφερειακών κόμβων.

Ο προγραμματισμός του κόμβου με τον αισθητήρα DHT11 έγινε σε C++ και το πρόγραμμα περιγράφεται αναλυτικά στο **ΠΑΡΑΡΤΗΜΑ Δ.1.**
Ο προγραμματισμός του κόμβου με τον αισθητήρα LDR έγινε σε C++ και το πρόγραμμα περιγράφεται αναλυτικά στο **ΠΑΡΑΡΤΗΜΑ Δ.2**

5.2 Λογισμικό για τον δέκτη

Η εφαρμογή λογισμικού του **δέκτη LoRa** υλοποιείται επίσης στο σύστημα ESP32 TTGO LoRa32 OLED. Ο δέκτης LoRa λαμβάνει τα πακέτα δεδομένων από τους περιφερειακούς κόμβους και τα παρουσιάζει στην OLED. Αποστέλλει **επιβεβαίωση (ACK)** προς τους πομπούς και ταυτόχρονα να τα **προωθεί μέσω πρωτοκόλλου UDP** σε υπολογιστή, όπου είναι διαθέσιμα στη διεπαφή χρήστη.

Η ανάπτυξη του κώδικα της εφαρμογής έγινε στο περιβάλλον ανάπτυξης **PlatformIO** (IDE) που υποστηρίζει πλήρως τον ESP32. Οι βασικές βιβλιοθήκες του PlatformIO που χρησιμοποιήθηκαν είναι:

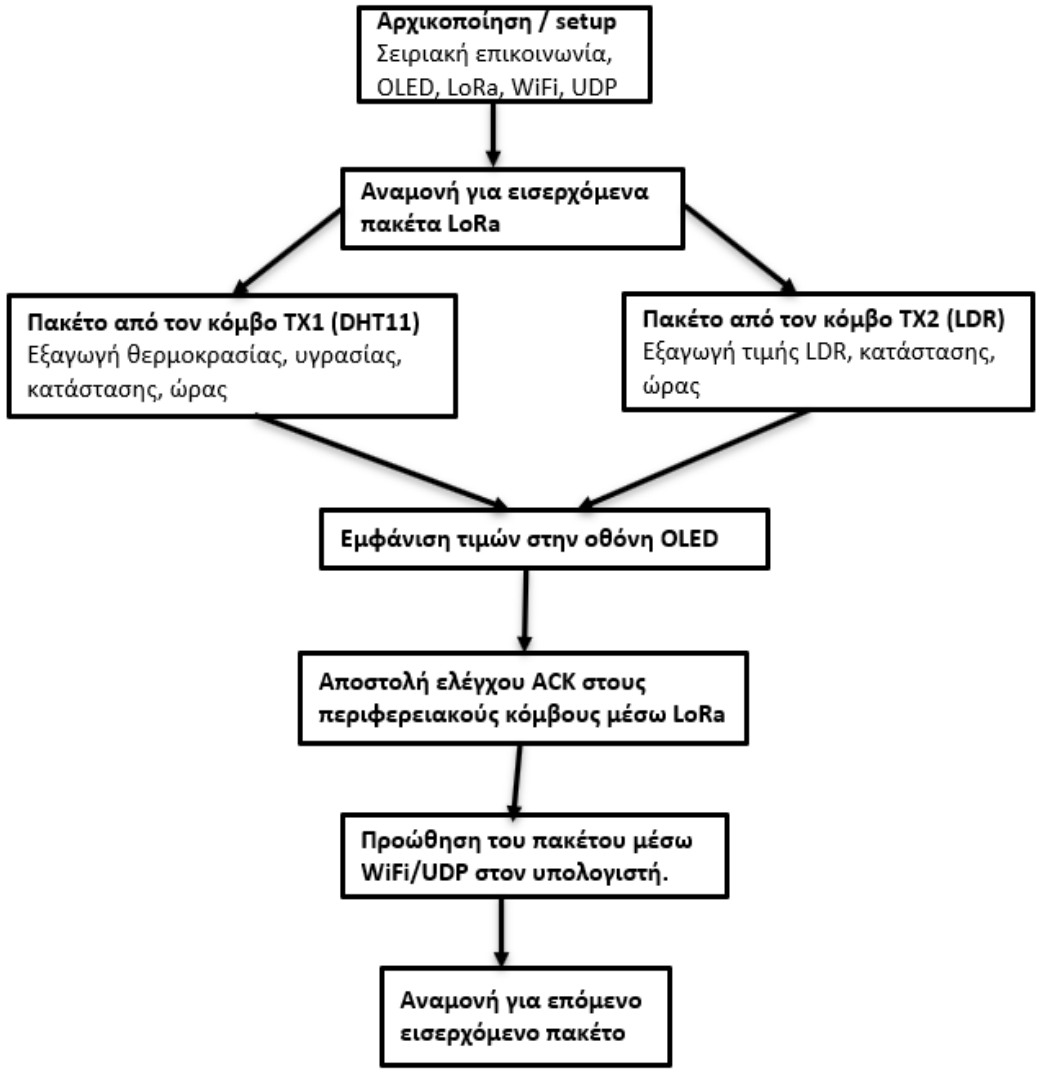
- **LoRa (Sandeep Mistry):** για τη ρύθμιση και διαχείριση επικοινωνίας με το LoRa module SX127x.
- **SSD1306Wire (ThingPulse):** για τον έλεγχο της οθόνης OLED SSD1306 μέσω διαύλου I²C.
- **WiFi και WiFiUDP:** για την ασύρματη σύνδεση του ESP32 σε τοπικό δίκτυο και την προώθηση των δεδομένων με χρήση πρωτοκόλλου UDP.
- **Arduino** που είναι βασική βιβλιοθήκη Arduino/ESP32
- **SPI** για SPI (LoRa).
- **Wire** για I2C OLED.
- **Images** για logo LoRa στην OLED

Η λειτουργία του προγράμματος οργανώνεται σε επιμέρους στάδια:

- **Αρχικοποιούνται** η σειριακή θύρα για έλεγχο και επιδιόρθωση σφαλμάτων (serial monitor), η οθόνη OLED, το LoRa module στα 868 MHz και η σύνδεση του συστήματος στο Wi-Fi με τις προκαθορισμένες παραμέτρους SSID και password
- **Λήψη πακέτων.** Στον βρόχο loop() που επαναλαμβάνεται, το πρόγραμμα ελέγχει συνεχώς αν υπάρχει εισερχόμενο πακέτο LoRa. Αν βρεθεί, το περιεχόμενο αποθηκεύεται σε πακέτο receivedData. Για τον περιφερειακό κόμβο **TX1 (DHT11)** το πακέτο (string) αναλύεται με substring για να εξαχθούν θερμοκρασία, υγρασία, κατάσταση και χρονική σήμανση.
  Για τον περιφερειακό κόμβο **TX2 (LDR)** χρησιμοποιείται η συνάρτηση sscanf για την εξαγωγή της τιμής ADC, του ποσοστού φωτεινότητας, της κατάστασης και της ώρας.
- **ACK και αξιοπιστία.** Μετά την επιτυχή λήψη, ο δέκτης αποστέλλει πίσω στον πομπό ένα **"ACK"**, ώστε να διασφαλιστεί η επιτυχής επικοινωνία. Αυτό υποστηρίζει το σχήμα **stop-and-wait** στους κόμβους, μειώνοντας την πιθανότητα απώλειας δεδομένων.
- **Προώθηση στο δίκτυο.** Κάθε πακέτο που λαμβάνεται μεταβιβάζεται μέσω πρωτοκόλλου **UDP** στον καθορισμένο serverIP και serverPort. Στον υπολογιστή τρέχει ένα πρόγραμμα σε Python που συλλέγει τα δεδομένα και τα αποθηκεύει σε αρχείο CSV και τα διαθέτει στη διεπαφή χρήστη.

- **Απεικόνιση στην OLED.** Η οθόνη OLED εμφανίζει συνοπτικά τις μετρήσεις που λαμβάνονται από κάθε κόμβο, μαζί με χρονική σήμανση, ώστε ο χειριστής να έχει άμεση εποπτεία των μετρήσεων επί τόπου, χωρίς να απαιτείται άνοιγμα του υπολογιστή.

Συνολικά, ο δέκτης λειτουργεί ως **γέφυρα** ανάμεσα στους απομακρυσμένους κόμβους LoRa και στο κεντρικό σύστημα (υπολογιστή), παρέχοντας **αξιοπιστία** μέσω ACK, **τοπική απεικόνιση** μέσω OLED, και **δικτυακή προώθηση** μέσω πρωτοκόλλου UDP. Το διάγραμμα Ροής του Λογισμικού του δέκτη φαίνεται στην εικόνα 5.3.
Ο προγραμματισμός του δέκτη έγινε σε C++ και το πρόγραμμα περιγράφεται αναλυτικά στο **ΠΑΡΑΡΤΗΜΑ Δ.3.**



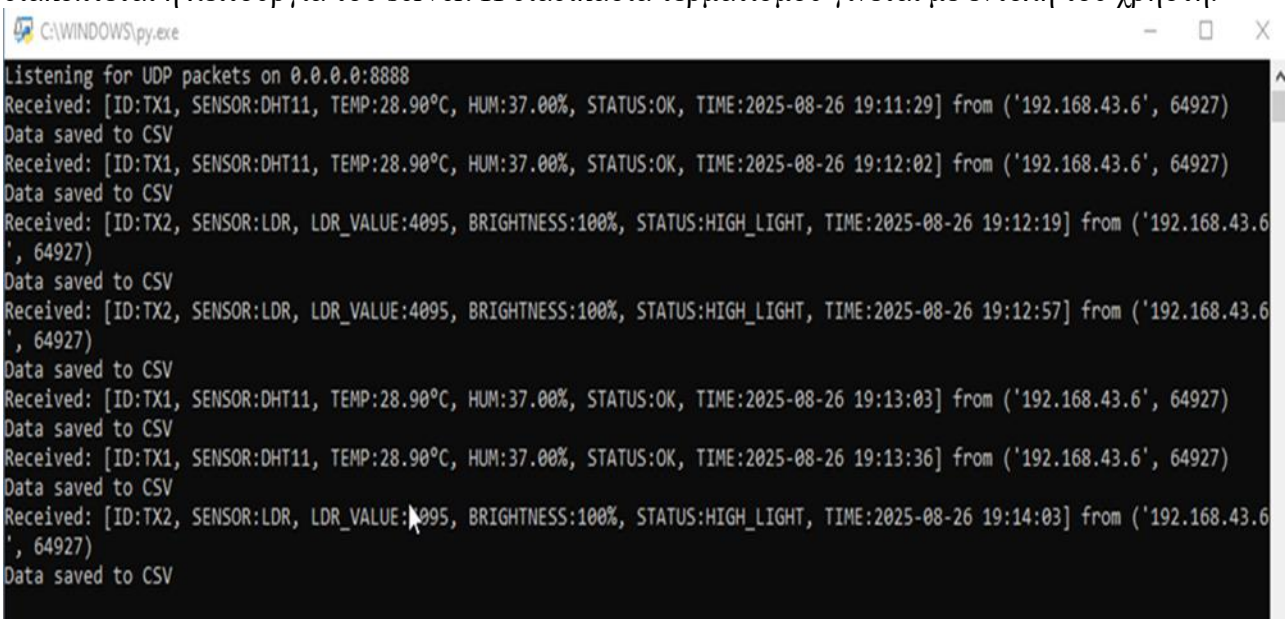
Εικόνα 5.3 Διάγραμμα Ροής Λογισμικού του Δέκτη

5.3 Διακομιστής UDP

Ο διακομιστής UDP που αναπτύχθηκε σε γλώσσα προγραμματισμού Python αποτελεί το ενδιάμεσο στάδιο μεταξύ του ESP32 και του υπολογιστικού συστήματος. Στην αρχικοποίηση του διακομιστή καθορίζονται η διεύθυνση και η θύρα ακρόασης, 0.0.0.0 και 8888 αντίστοιχα, ώστε να δέχεται πακέτα από οποιαδήποτε πηγή εντός του τοπικού δικτύου. Παράλληλα πραγματοποιείται έλεγχος για την ύπαρξη του φακέλου αποθήκευσης και του αρχείου δεδομένων. Σε περίπτωση που αυτά δεν υπάρχουν, δημιουργούνται αυτόματα. Το αρχείο CSV αρχικοποιείται με μια επικεφαλίδα που περιλαμβάνει όλα τα απαιτούμενα πεδία, όπως χρονική σήμανση, αναγνωριστικό συσκευής, τύπο αισθητήρα, τιμές θερμοκρασίας, υγρασίας και φωτεινότητας, ποσοστό φωτεινότητας, κατάσταση λειτουργίας και διεύθυνση προέλευσης του πακέτου.

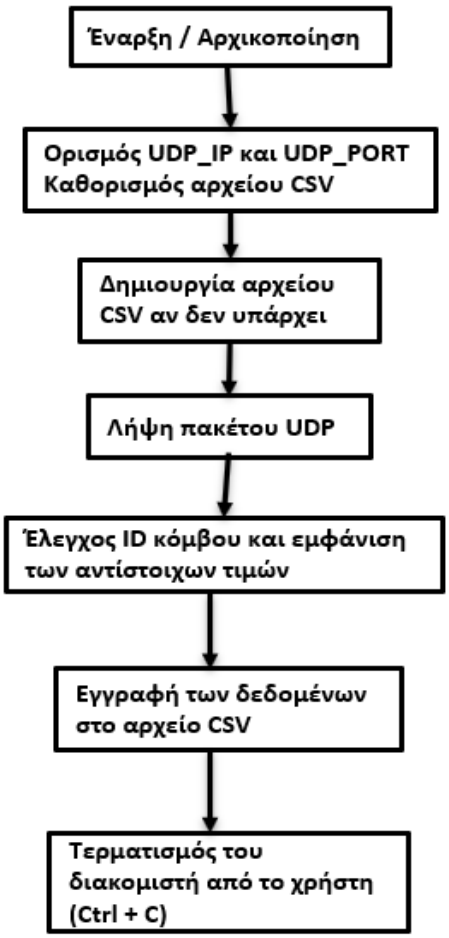
Στη συνέχεια ο διακομιστής δεσμεύει τη θύρα UDP και εισέρχεται σε βρόχο αδιάκοπης λειτουργίας, αναμένοντας εισερχόμενα datagrams μέσω της εντολής recvfrom(). Κάθε πακέτο που λαμβάνεται αποκωδικοποιείται σε μορφή UTF-8 και εμφανίζεται στην κονσόλα όπως βλέπουμε και στην εικόνα 5.4, παρέχοντας έτσι στον χρήστη δυνατότητα εποπτείας σε πραγματικό χρόνο. Με βάση το αναγνωριστικό της συσκευής καθορίζεται ο τύπος των δεδομένων που θα εξαχθούν. Όταν το αναγνωριστικό ID είναι TX1, ο server αναγνωρίζει ότι πρόκειται για τον κόμβο με αισθητήρα DHT11 και εξάγει τις μετρήσεις θερμοκρασίας και υγρασίας, ενώ όταν το αναγνωριστικό ID είναι TX2, τα δεδομένα προέρχονται από τον αισθητήρα LDR και περιλαμβάνουν την τιμή φωτεινότητας καθώς και το αντίστοιχο ποσοστό φωτεινότητας. Σε όλες τις περιπτώσεις καταγράφονται επιπλέον η χρονική σήμανση, η κατάσταση της συσκευής και η IP διεύθυνση του αποστολέα.

Μετά την ανάλυση τα δεδομένα αποθηκεύονται σε πίνακα του αρχείου CSV. Ο χρήστης ενημερώνεται κάθε φορά μέσω μηνύματος στην κονσόλα ότι η καταγραφή ολοκληρώθηκε επιτυχώς. Σε περίπτωση που η μορφή του πακέτου δεν ανταποκρίνεται στις προσδοκίες, ο κώδικας διαθέτει χειρισμό εξαιρέσεων ώστε να εμφανίζεται σχετικό μήνυμα σφάλματος χωρίς να διακόπτεται η λειτουργία του server. Η διαδικασία τερματισμού γίνεται με εντολή του χρήστη.



Εικόνα 5.4 Λειτουργία του διακομιστή UDP

Στην εικόνα 5.5 παρατίθεται το διάγραμμα ροής του λογισμικού του διακομιστή UDP.
Το πρόγραμμα σε γλώσσα προγραμματισμού python που έχει δημιουργηθεί για τον διακομιστή UDP βρίσκεται στο **ΠΑΡΑΡΤΗΜΑ Δ.4.**



Εικόνα 5.5 Διάγραμμα Ροής Λογισμικού UDP SERVER

5.4 Διεπαφή χρήστη (UI)

ΤΟ **γραφικό περιβάλλον χρήστη (User Interface – UI)**, βασίζεται σε τεχνολογίες διαδικτύου (HTML, CSS και JavaScript). Η διεπαφή διαβάζει τα δεδομένα των από το αρχείο sensor_data.csv. Τα παρουσιάζει σε μορφή πίνακα και σε μορφή **γραφημάτων** για καλύτερη κατανόηση και ανάλυση.

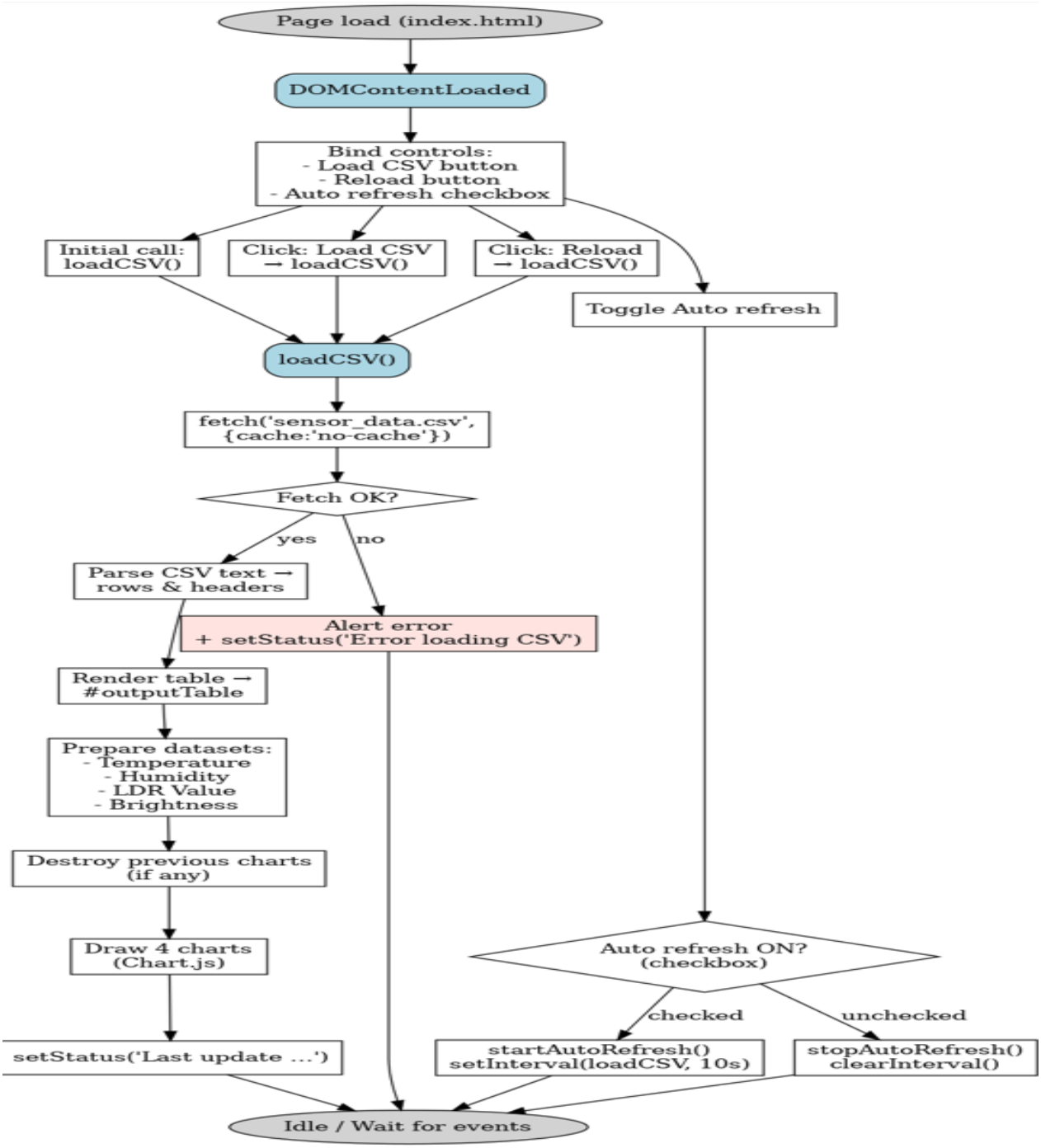
Η διεπαφή χρήστη λειτουργεί ως ένας ελαφρύς, καθαρά client-side "viewer" των μετρήσεων που περιέχει το αρχείο sensor_data.csv. Στο HTML αρχείο ορίζεται ένας απλός καμβάς εφαρμογής με τίτλο, μια μπάρα ελέγχου με τρία στοιχεία (κουμπί "Load sensor_data.csv" που καλεί τη loadCSV(), κουμπί "Reload" για άμεση επαναφόρτωση και checkbox "Auto refresh (10s)" που ενεργοποιεί/απενεργοποιεί αυτόματο συγχρονισμό), ένα κουμπί εμφάνισης των γραφημάτων, ένα placeholder όπου εγχέεται ο δυναμικός πίνακας και τέσσερις καμβάδες (chart1…chart4) για τα γραφήματα.

Η λειτουργία της JavaScript ξεκινάει όταν γίνεται η πρώτη φόρτωση αρχείου (loadCSV()). Η υποδομή auto-refresh βασίζεται σε setInterval με περίοδο 10 δευτερόλεπτα και σε επιλογές startAutoRefresh/stopAutoRefresh, με ενημέρωση μιας μικρής ένδειξης κατάστασης ("Last update). Το κουμπί "Reload" καλεί απλώς τη loadCSV() χωρίς να πειράζει το auto-refresh.

Η loadCSV() καλεί το αρχείο sensor_data.csv, μετατρέπει το περιεχόμενο του σε κείμενο και το περνά σε έναν απλό parser που σπάει το αρχείο σε γραμμές/κελιά. Έπειτα χτίζει έναν HTML πίνακα όπου η πρώτη γραμμή γίνεται κεφαλίδα και οι υπόλοιπες αποδίδονται ως σώμα, και τέλος σχεδιάζει τέσσερα γραφήματα γραμμής (Temperature, Humidity, LDR Value, Brightness) χρησιμοποιώντας τις στήλες που εντοπίζει. Πριν από κάθε νέο σχεδίασμα καταστρέφει τυχόν προϋπάρχοντα charts για να μη σωρεύονται αντικείμενα, ενώ τα γραφήματα αποδίδονται με απλά χρώματα (κόκκινο, μπλε, πράσινο, πορτοκαλί) και κοινές ρυθμίσεις (τίτλος, άξονες, μικρή καμπύλωση, χωρίς γέμισμα). Σε αποτυχία φόρτωσης εμφανίζεται ενημερωτικό alert που υπενθυμίζει ότι το αρχείο πρέπει να βρίσκεται στον ίδιο φάκελο και ότι η σελίδα πρέπει να φορτώνεται από τοπικό server, και η ένδειξη κατάστασης αλλάζει σε σφάλμα, ώστε ο χρήστης να καταλάβει τι συνέβη. Η παρουσίαση της διεπαφής χρήστη γίνεται με CSS

Με αυτόν τον τρόπο, το UI προσφέρει τρία πράγματα με ελάχιστη πολυπλοκότητα. Την χειροκίνητη φόρτωση του CSV, την αυτόματη περιοδική ανανέωση με ορατή κατάσταση και την σύγχρονη απεικόνιση τόσο σε μορφή πίνακα όσο και σε τέσσερα γραφήματα, όλα αποκλειστικά στον φυλλομετρητή. Με αυτόν τον τρόπο, ο χρήστης μπορεί εύκολα να εντοπίζει τάσεις, συσχετίσεις και ανωμαλίες στις μετρήσεις που καταγράφονται από τους αισθητήρες. Στη εικόνα 5.6 παρατίθεται το διάγραμμα ροής του λογισμικού της διεπαφής χρήστη.

Οι κώδικες των αρχείων **index.html**, **script.js** και **style.css** βρίσκονται αναλυτικά με περιγραφές στο **ΠΑΡΑΡΤΗΜΑ Δ.5.**



Εικόνα 5.6 Διάγραμμα Ροής Λογισμικού USER INTERFACE

6. Αποτίμηση του συστήματος

Για την αξιολόγηση του συστήματος, πραγματοποιήθηκε συλλογή πραγματικών μετρήσεων από τους αισθητήρες που ενσωματώθηκαν στους κόμβους του συστήματος (θερμοκρασίας, υγρασίας, φωτεινότητας και GPS). Χρησιμοποιήθηκε το πρωτόκολλο LoRa για την αποστολή των μετρήσεων προς την κεντρική μονάδα λήψης, όπου εμφανίστηκαν σε πραγματικό χρόνο στην οθόνη OLED και στη συνέχεια αποθηκεύτηκαν σε τοπικό υπολογιστή με τη χρήση πρωτοκόλλου UDP σε μορφή αρχείων CSV.

Η διαδικασία αξιολογήθηκε με βάσης την επεξεργασία και ανάλυση των δεδομένων που συλλέχθηκαν και ελέγχθηκαν:

- η αξιοπιστία και η σταθερότητα της μετάδοσης.
- η ακρίβεια και η συνέπεια των μετρήσεων των αισθητήρων.
- η ανταπόκριση του συστήματος σε πραγματικές συνθήκες λειτουργίας.

Για την καλύτερη κατανόηση και ερμηνεία των αποτελεσμάτων, τα δεδομένα παρουσιάστηκαν σε πίνακες και γραφήματα μέσω του περιβάλλοντος της διεπαφής χρήστη που υλοποιήθηκε. Το περιβάλλον αυτό δίνει τη δυνατότητα:

- εμφάνισης γραφημάτων σε πραγματικό χρόνο για κάθε μεταβλητή
- φιλτραρίσματος των δεδομένων ανά αισθητήρα ή χρονικό διάστημα
- προβολής ιστορικών μετρήσεων σε συνδυασμό με τις τρέχουσες τιμές
- άμεσης ανανέωσης των πληροφοριών χωρίς την ανάγκη πρόσθετης παρέμβασης του χρήστη.

Με αυτόν τον τρόπο, ο χρήστης έχει στη διάθεσή του ένα ολοκληρωμένο εργαλείο παρακολούθησης, που συνδυάζει την αποθήκευση των δεδομένων με την ευέλικτη και διαδραστική απεικόνισή τους.

Τα δεδομένα των αισθητήρων καταγράφηκαν σε πραγματικό χρόνο και απεστάλησαν μέσω του πρωτοκόλλου LoRa στην κεντρική μονάδα λήψης. Στη συνέχεια, μέσω Wi-Fi/UDP αποθηκεύτηκαν σε αρχεία CSV και αναπαραστάθηκαν γραφικά στο διεπαφή χρήστη. Στην εικόνα 6.1 φαίνονται οι μετρήσεις μέσω της σελίδας διεπαφής χρήστη.

Στις εικόνες 6.2, 6.3 αποτυπώνονται τα γραφήματα θερμοκρασίας, υγρασίας, τιμής LDR και φωτεινότητας (%) σε συνάρτηση με το χρόνο. Η συμπεριφορά των καμπυλών των γραφημάτων οφείλεται στις εναλλαγές των τιμών που βρίσκονται στην εικόνα 6.1.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ - ΤΜΗΜΑ ΦΥΣΙΚΗΣ

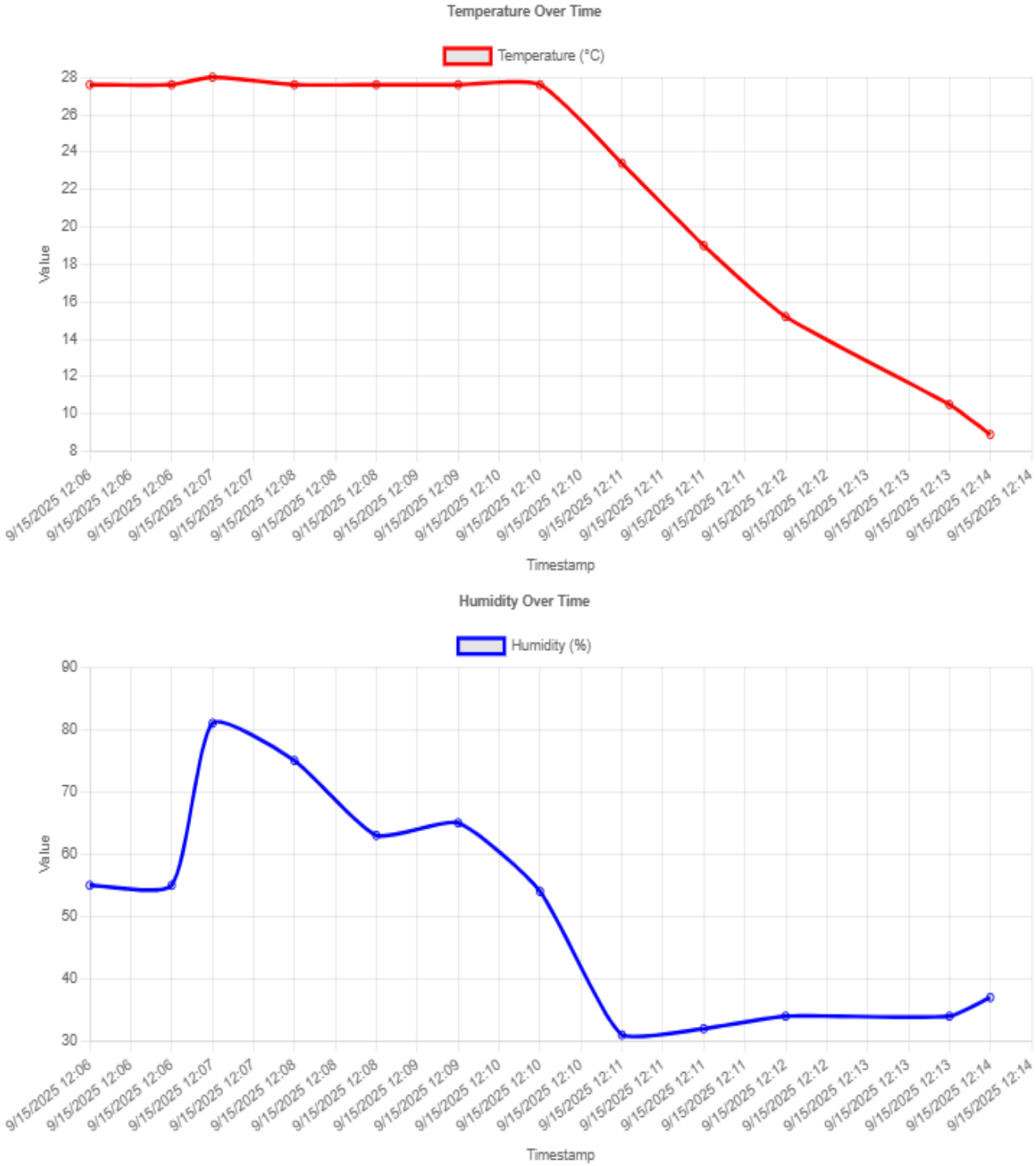
Μεταπτυχιακή Διπλωματική Εργασία

Ανάπτυξη Ασύρματου Κατανεμημένου Συστήματος Αυτοματισμού IoT με Μικροελεγκτές

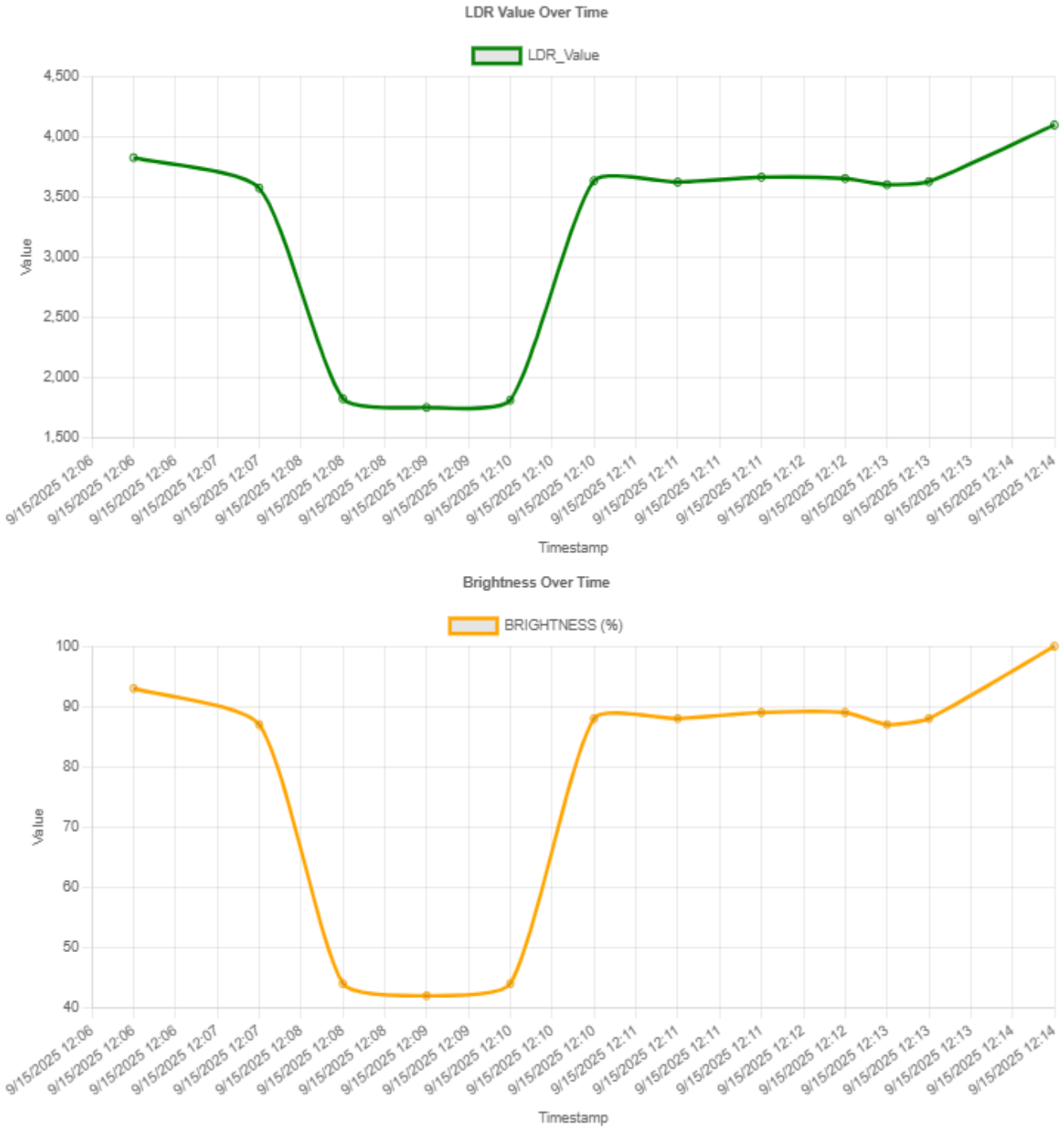
Load sensor_data.csv | Reload | ☐ Auto refresh (10s) | Last update: 10/12/2025, 10:54:34 PM

Timestamp	Device_ID	Sensor	Temperature (°C)	Humidity (%)	LDR_Value	BRIGHTNESS (%)	Status	Received From
9/15/2025 12:06	TX1	DHT11	27.6	55			OK	192.168.1.7:53958
9/15/2025 12:06	TX2	LDR			3824	93	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:06	TX1	DHT11	27.6	55			OK	192.168.1.7:53958
9/15/2025 12:07	TX1	DHT11	28	81			HIGH HUM	192.168.1.7:53958
9/15/2025 12:07	TX2	LDR			3572	87	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:08	TX1	DHT11	27.6	75			HIGH HUM	192.168.1.7:53958
9/15/2025 12:08	TX2	LDR			1824	44	OK	192.168.1.7:53958
9/15/2025 12:08	TX1	DHT11	27.6	63			OK	192.168.1.7:53958
9/15/2025 12:09	TX2	LDR			1751	42	OK	192.168.1.7:53958
9/15/2025 12:09	TX1	DHT11	27.6	65			OK	192.168.1.7:53958
9/15/2025 12:10	TX2	LDR			1812	44	OK	192.168.1.7:53958
9/15/2025 12:10	TX1	DHT11	27.6	54			OK	192.168.1.7:53958
9/15/2025 12:10	TX2	LDR			3632	88	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:11	TX1	DHT11	23.4	31			OK	192.168.1.7:53958
9/15/2025 12:11	TX2	LDR			3621	88	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:11	TX1	DHT11	19	32			OK	192.168.1.7:53958
9/15/2025 12:11	TX2	LDR			3662	89	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:12	TX1	DHT11	15.2	34			OK	192.168.1.7:53958
9/15/2025 12:12	TX2	LDR			3651	89	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:13	TX2	LDR			3600	87	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:13	TX2	LDR			3625	88	HIGH_LIGHT	192.168.1.7:53958
9/15/2025 12:13	TX1	DHT11	10.5	34			LOW TEMP	192.168.1.7:53958
9/15/2025 12:14	TX1	DHT11	8.9	37			LOW TEMP	192.168.1.7:53958
9/15/2025 12:14	TX2	LDR			4095	100	HIGH_LIGHT	192.168.1.7:53958

Εικόνα 6.1 Παρουσίαση των μετρήσεων στη διεπαφή χρήστη



Εικόνα 6.1 Χρονοσειρές μετρήσεων αισθητήρα θερμοκρασίας-υγρασίας DHT11



Εικόνα 6.2 Γραφήματα μετρήσεων αισθητήρα φωτός LDR

Συνοψίζοντας, τα αποτελέσματα των μετρήσεων καταδεικνύουν ότι το ασύρματο κατανεμημένο σύστημα αυτοματισμού που αναπτύχθηκε λειτουργεί αξιόπιστα και με συνέπεια. Οι περιβαλλοντικοί αισθητήρες απέδωσαν μετρήσεις εντός των αναμενόμενων ορίων, ενώ η ασύρματη επικοινωνία μέσω LoRa διατήρησε σταθερή σύνδεση μεταξύ των κόμβων και της κεντρικής μονάδας. Η απεικόνιση των δεδομένων στο User Interface διευκολύνει σημαντικά την ερμηνεία και την αξιολόγηση των αποτελεσμάτων.

Η δυνατότητα οπτικοποίησης σε πραγματικό χρόνο, σε συνδυασμό με την αξιοπιστία των μετρήσεων, επιβεβαιώνει ότι η αρχιτεκτονική που υλοποιήθηκε μπορεί να αποτελέσει μια πρακτική και αποδοτική λύση για εφαρμογές παρακολούθησης σε απομακρυσμένα περιβάλλοντα. Τα παραγόμενα γραφήματα όχι μόνο αποδεικνύουν τη σωστή λειτουργία του συστήματος, αλλά παράλληλα αποτελούν εργαλείο περαιτέρω ανάλυσης και βελτιστοποίησης.

6.1 Εμβέλεια του συστήματος

Η εμβέλεια του συστήματος προσδιορίστηκε αφού απομακρύνθηκε το σύστημα των αισθητήρων από το σύστημα του δέκτη σε απόσταση από 300m έως 2.5Km. Αξιολογούνται η μέγιστη απόσταση επιτυχούς επικοινωνίας, καθώς και οι δείκτες ποιότητας λήψης (RSSI, SNR) σε ημιαστικό περιβάλλον.
Οι ρυθμίσεις και η διάταξη για τις δοκιμές είναι:

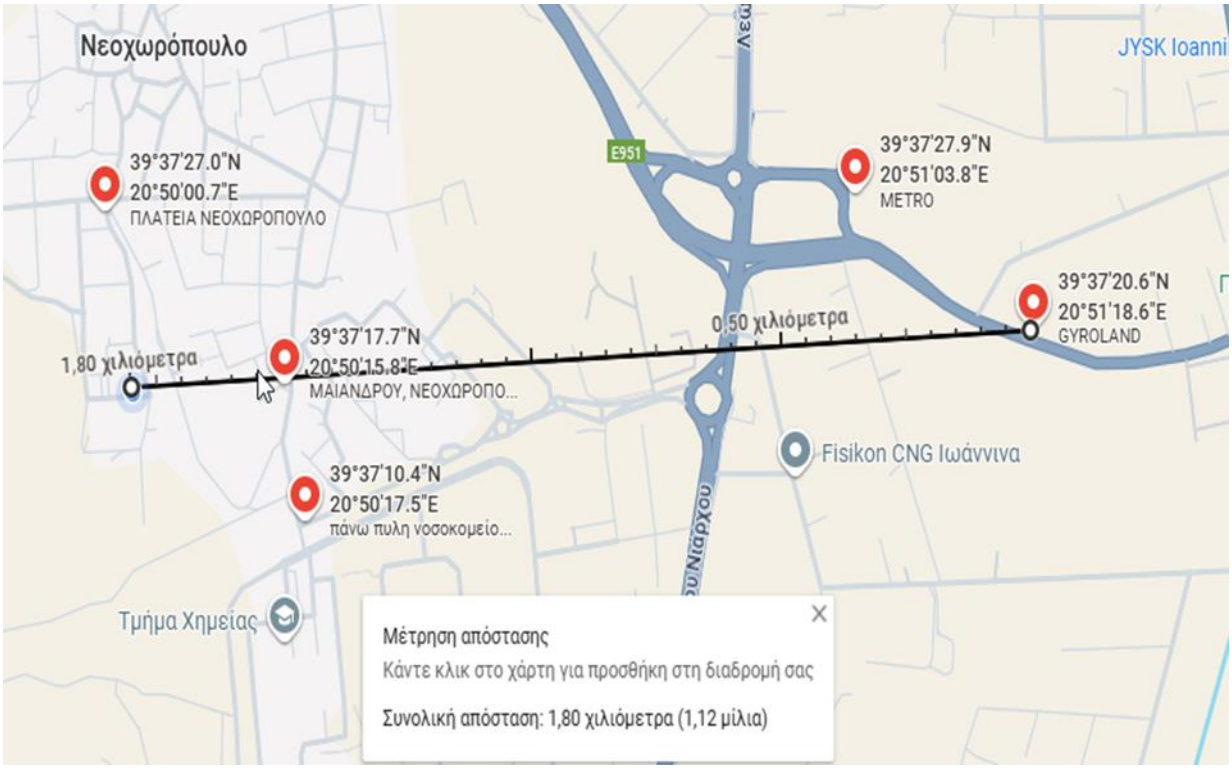
- Πλατφόρμες: TTGO LoRa32 OLED V1 (ESP32 + SX1276).
- Συχνότητα: **868.1 MHz** (ISM Ευρώπης).
- Παράμετροι LoRa (Max-Range): SF12, BW 125 kHz, CR 8, Preamble 12, CRC ON, SyncWord 0x12.
- Ισχύς εκπομπής: **20 dBm EIRP** (20 dBm για διερεύνηση μέγιστης εμβέλειας).
- Κεραίες: εξωτερικές 868 MHz, κάθετη πόλωση.
- Σημείο αναφοράς (δέκτης): **[LAT_0, LON_0]** (υψόμετρο ~**547,3 m**).
- Περιβάλλον: **ημιαστικό**

Μετρήθηκαν, ο Δείκτης Ισχύος Ληφθέντος Σήματος (RSSI) σε dBm που δείχνει πόσο ισχυρό είναι το σήμα που λαμβάνει η συσκευή από τον πομπό και ο Λόγος Σήματος προς Θόρυβο (SNR) σε dB που δείχνει πόσο καθαρό είναι το σήμα σε σχέση με τον ηλεκτρονικό “θόρυβο” του περιβάλλοντος. Ενδεικτικά για το RSSI, πρόκειται για αρνητικές τιμές (π.χ. -40 dBm είναι καλύτερο από -80 dBm). Για το SNR όσο μεγαλύτερο είναι τόσο καθαρότερο και σταθερότερο το σήμα. Είναι θετικός αριθμός στις καλές συνδέσεις (π.χ. 10–30 dB).
Στον πίνακα 6.2 φαίνονται οι μετρήσεις συντεταγμένων καθώς και οι δείκτες RSSI, SNR

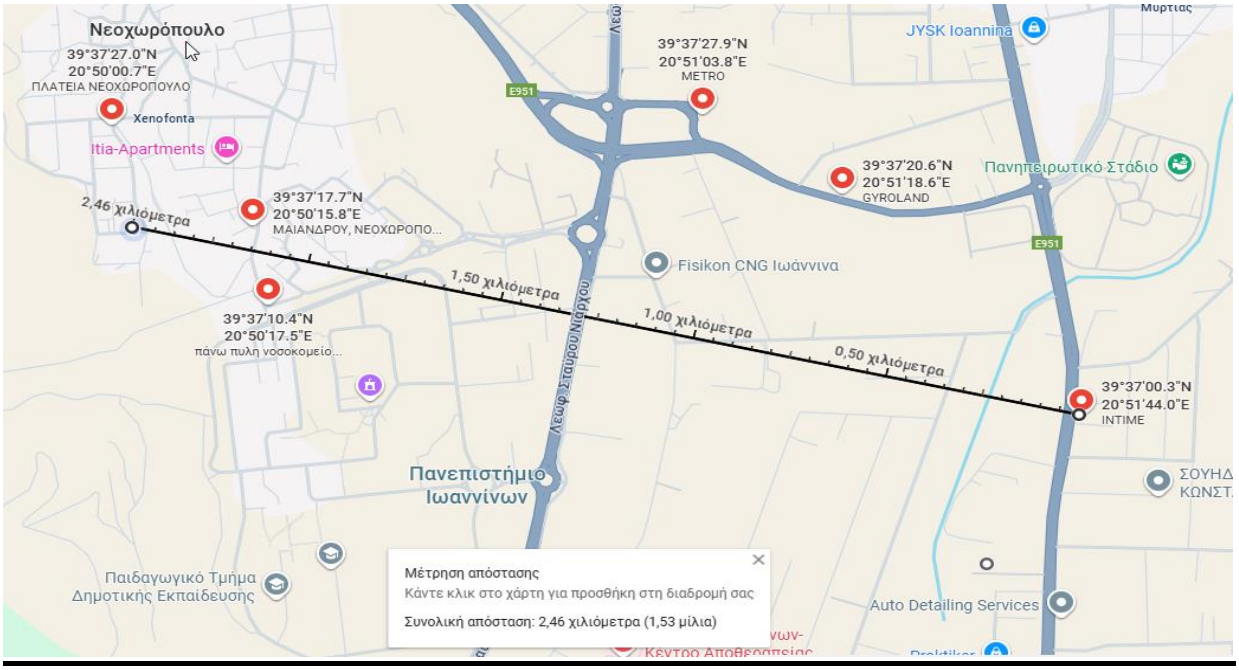
Συντεταγμένες του δέκτη (receiver) / Συντεταγμένες κινητού κόμβου μέτρησης	39.62147331444905, 20.83417308948236
39.62444323331782, 20.833559457755634	**328.44 m (RSSI: -68, SNR: 8.5)**
39.62191446682099, 20.837657873166243	**306.08 m(RSSI: -70, SNR: 15.3)**
39.61984841253434, 20.838151399629304	**385.15m(RSSI: -88, SNR: 13.3)**
39.624469960239246, 20.851039777874167	**1480m(RSSI: -100, SNR: -14.3)**
39.62238911392552, 20.8551752233836	**1800m(RSSI: -117, SNR: -22.3)**
39.6167457193227, 20.862225124248805	**2460m(RSSI: -117, SNR: -11.5)**

Πίνακας 6.1 Μέτρηση εμβέλειας του συστήματος βάση συντεταγμένων GPS

Ο υπολογισμός των αποστάσεων έγινε μέσω του Google Maps όπου τοποθετήθηκαν οι συντεταγμένες του δέκτη και του του κόμβου ανάλογα με το σημείο τοποθέτησης του. Στην Εικόνες 6.3, 6.4 παρουσιάζονται οι υπολογισμοί εμβέλειας του συστήματος μέσω Google Maps.



Εικόνα 6.3 Υπολογισμός εμβέλειας συστήματος μέσω Google Maps



Εικόνα 6.4 Υπολογισμός εμβέλειας συστήματος μέσω Google Maps

7. Συμπεράσματα – Συζήτηση

7.1 Συμπεράσματα - Γενικές παρατηρήσεις

Το αντικείμενο της εργασία είναι η ανάπτυξη ενός ασύρματου κατανεμημένου συστήματος αυτοματισμού βασισμένου στο Διαδίκτυο των Πραγμάτων (IoT), με αξιοποίηση μικροελεγκτών ESP32 και του πρωτοκόλλου LoRa. Το σύστημα συνδυάζει κόμβους συλλογής δεδομένων με κεντρική μονάδα λήψης και διεπαφή χρήστη και παρέχει μια ολοκληρωμένη, αξιόπιστης και ενεργειακά αποδοτική λύση για περιβαλλοντική παρακολούθηση σε πραγματικές συνθήκες.
Ειδικότερα:

- Η αρχιτεκτονική hub-and-spoke που εφαρμόστηκε ήταν ιδιαίτερα λειτουργική και επέτρεψε την αποδοτική διαχείριση δεδομένων από πολλαπλούς κόμβους με ελάχιστη πολυπλοκότητα.
- Το πρωτόκολλο LoRa προσέφερε ικανοποιητική κάλυψη και σταθερότητα στη μετάδοση.
- Η ενσωμάτωση αισθητήρων θερμοκρασίας, υγρασίας, φωτεινότητας και GPS στο σύστημα, έδειξαν τη χρήση του συστήματος δε περιβαλλοντικές ή άλλες εφαρμογές.
- Η ανάπτυξη διεπαφής χρήστη βασισμένης σε τεχνολογίες web (HTML/JavaScript/CSS) προσέφερε τη δυνατότητα παρακολούθησης σε πραγματικό χρόνο, με γραφικές απεικονίσεις που διευκολύνουν την καλύτερη περιβαλλοντική εποπτεία και την εξαγωγή συμπερασμάτων από τον χρήστη.
- Η αποθήκευση των δεδομένων σε μορφή CSV παρέχει συμβατότητα με εργαλεία επεξεργασίας και ανάλυσης, καθιστώντας το σύστημα χρήσιμο για μελλοντικές επιστημονικές ή βιομηχανικές εφαρμογές.

Συνολικά, η εργασία ανέδειξε τη δυνατότητα συνδυασμού φθηνών, χαμηλής κατανάλωσης συσκευών με ασύρματα πρωτόκολλα μεγάλης εμβέλειας, για τη δημιουργία κατανεμημένων συστημάτων αυτοματισμού που μπορούν να λειτουργήσουν αξιόπιστα σε απομακρυσμένα ή δύσκολα προσβάσιμα περιβάλλοντα.

Το σύστημα μπορεί να αναπτυχθεί περαιτέρω με:

1. **Εμπλουτισμό και διαφοροποίηση αισθητήρων**
   - Προσθήκη επιπλέον αισθητήρων, όπως πίεσης, ποιότητας αέρα, συγκέντρωσης CO_2, ρύπων και θορύβου, ώστε το σύστημα να παρέχει πληρέστερη εικόνα των περιβαλλοντικών συνθηκών.
   - Χρήση πιο εξελιγμένων αισθητήρων με μεγαλύτερη ακρίβεια και χαμηλότερη κατανάλωση.
2. **Ασφάλεια και αξιοπιστία δεδομένων**
   - Ενσωμάτωση μηχανισμών κρυπτογράφησης (AES, TLS) για την προστασία των μεταδιδόμενων δεδομένων.
   - Ανάπτυξη μεθόδων ανίχνευσης σφαλμάτων και ανακατανομής φορτίου μεταξύ κόμβων για αυξημένη αξιοπιστία.
3. **Ενεργειακή βελτιστοποίηση**
   - Υιοθέτηση στρατηγικών για ακόμη χαμηλότερη κατανάλωση (deep sleep modes, duty cycling).
   - Ενσωμάτωση ανανεώσιμων πηγών ενέργειας για αύξηση της αυτονομίας.
4. **Δικτυακή επεκτασιμότητα**
   - Ενσωμάτωση υποστήριξης LoRaWAN για μεγαλύτερη κλίμακα εφαρμογών και δυνατότητα σύνδεσης με δημόσιες υποδομές.
   - Διερεύνηση αρχιτεκτονικών mesh για αυξημένη ευελιξία και κάλυψη.
5. **Διασύνδεση με Cloud και Big Data**
   - Αποθήκευση και ανάλυση δεδομένων σε cloud υποδομές για δυνατότητες μακροχρόνιας μελέτης, απομακρυσμένης πρόσβασης και συνεργασίας πολλών χρηστών.
   - Εφαρμογή τεχνικών μηχανικής μάθησης (machine learning) για πρόβλεψη τάσεων και εντοπισμό ανωμαλιών.
6. **Edge Computing και Τεχνητή Νοημοσύνη**
   - Ενσωμάτωση τεχνικών TinyML στους κόμβους για τοπική ανάλυση δεδομένων και λήψη αποφάσεων σε πραγματικό χρόνο.
   - Ανάπτυξη ευφυών αλγορίθμων που προσαρμόζουν δυναμικά τη συχνότητα δειγματοληψίας ανάλογα με τις συνθήκες.
7. **Βελτίωση διεπαφής χρήστη**
   - Αναβάθμιση της διεπαφής με δυναμικά dashboards, ειδοποιήσεις (alerts) και δυνατότητα απομακρυσμένου ελέγχου.
   - Ενσωμάτωση υποστήριξης σε φορητές συσκευές (mobile applications) για μεγαλύτερη προσβασιμότητα.

Με αυτές τις βελτιώσεις, το σύστημα μπορεί να εξελιχθεί σε μια ολοκληρωμένη πλατφόρμα παρακολούθησης και αυτοματισμού, με εφαρμογές που εκτείνονται από την γεωργία ακριβείας και τις έξυπνες πόλεις έως τη βιομηχανική αυτοματοποίηση και την περιβαλλοντική προστασία, αποτελώντας σημαντική συνεισφορά στον τομέα των κατανεμημένων συστημάτων IoT.

8. Αναφορές

[1] A. F. Fitsilis, *Διαδίκτυο των Αντικειμένων (IoT)*. Διαθέσιμο: https://eclass.uth.gr/modules/document/file.php/DE_U_105/%CE%98%CE%95%CE%A9%CE%A1%CE%99%CE%91/%CE%A6%CE%B9%CF%84%CF%83%CE%B9%CE%BB%CE%AE%CF%82_%CE%9A%CE%B5%CF%86%CE%AC%CE%BB%CE%B1%CE%B9%CE%BF_9%20%CE%B4%CE%B9%CE%BF%CF%81%CE%B8%CF%89%CE%BC%CE%AD%CE%BD%CE%BF_%CE%A6%CE%99%CE%A4%CE%A3%CE%99%CE%9B%CE%97%CE%A3_V2.pdf?utm_source

[2] Σ. Μπαμπατσίκου, *Internet of Things: Τεχνολογίες, Πρωτόκολλα και Εφαρμογές*. Διαθέσιμο: https://bouras.upatras.gr/wp-content/uploads/2024/07/Babatsikou_thlematiki.pdf?utm

[3] A. Azari, Ç. Çavdar, "Self-organized Low-power IoT Networks: A Distributed Learning Approach", *arXiv preprint*, 2018. https://arxiv.org/pdf/1807.09333

[4] Ayanle Ibrahim Ali* and Sibel Zorlu Partal, Development and performance analysis of a ZigBee and LoRa-based smart building sensor network, Dept. of Electrical Engineering, Yildiz Technical University, Istanbul, Turkey https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2022.933743/full

[5] Σ. Μητρόπουλος (επιμ.), Μ. Δασυγένης, «Internet of Things Computing», Κάλλιπος, 2024. https://repository.kallipos.gr/bitstream/11419/13294/4/168-DASYGENIS-Internet-of-Things-Computing.pdf

[6] ΣΑΡΑΝΤΗΣ ΜΗΤΡΟΠΟΥΛΟΣ, Καθηγητής Ιονίου Πανεπιστημίου και ΧΡΗΣΤΟΣ ΔΟΥΛΗΓΕΡΗΣ, Καθηγητής Πανεπιστημίου Πειραιώς «Κατανεμημένα Πληροφοριακά Συστήματα και Διαχείρισή τους: Αρχιτεκτονική, Υπηρεσίες, Προγραμματισμός, Εφαρμογές, Ασφάλεια», Κάλλιπος (βιβλίο). https://repository.kallipos.gr/bitstream/11419/10301/4/11_Mitropoulos_Distributed-Information-Systems.pdf

[7] Σ. Μητρόπουλος, «Internet of Things and Applications», Κάλλιπος, 2023. [Online]. Διαθέσιμο: αποθετήριο Κάλλιπος. Τμήμα κεφαλαίου για κατανεμημένες αρχιτεκτονικές & τομείς εφαρμογών. https://repository.kallipos.gr/bitstream/11419/11095/1/11_Mitropoulos_Distributed-Information-Systems_CH11.pdf

[8] Διονυσόπουλος Γεώργιος, ΟΠΑ (Μεταπτ.), «Ασύρματες Τεχνολογίες σε Εφαρμογές IoT» (2021) https://mm.aueb.gr/master_theses/xylomenos/2022-dionisopoulos.pdf?utm

[9] Industry 5.0: A Transformative Vision for Europe ESIR Policy Brief No. 3 https://www.horizon-europe.gouv.fr/sites/default/files/2022-01/industry-5-0-pdf-5324.pdf

[10] Μπόχλος Αθανάσιος ΜΔΕ «Ανάπτυξη ενεργειακά αυτόνομου ασύρματου συστήματος αναμετάδοσης ψηφιακής πληροφορίας» Κεφ. 1 (2013)

[11] Δ. Κουτσόπουλος (επιμ.), *Βασικές Αρχές Ασύρματης Επικοινωνίας* (μετάφραση του Tse & Viswanath), Αθήνα: Εκδόσεις Κλειδάριθμος, 2012.

[12] Norah Huang, SEO συγγραφέας, IoT & Τεχνικός
https://www.mokolora.com/el/full-understanding-of-lora-and-lorawan/

[13] *Performance of TCP/UDP under Ad Hoc IEEE802.11* — Milenko Petrovic, Mokhtar Aboelaze. Μελετά τη συμπεριφορά UDP και TCP σε ad hoc Wi-Fi (IEEE 802.11).
https://arxiv.org/pdf/cs/0311049

[14] Χρήστος Π. Κουτσουραδής, ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ, Σχεδίαση και Ανάπτυξη Ασύρματου Συστήματος Αισθητήρων για Εφαρμογές Προγνωστικής Συντήρησης. Κεφ. 2
https://circuits.ece.ntua.gr/documents/DiplomaTheses/koutsouradis.pdf

[15] Δρ. Φασουλάς Ιωάννης , Μικροελεγκτές, διατάξεις και Μηχατρονικά συστήματα. Τεχνολογία και Εφαρμογές Μικροελεγκτών − Τμήμα Μηχανολόγων Μηχανικών
https://eclass.hmu.gr/modules/document/file.php/MECH133/%CE%94%CE%99%CE%91%CE%9B%CE%95%CE%9E%CE%95%CE%99%CE%A3/%5B%201%20%5D%20%CE%9C%CE%B9%CE%BA%CF%81%CE%BF%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%B5%CC%81%CF%82%2C%20%CE%B4%CE%B9%CE%B1%CF%84%CE%B1%CC%81%CE%BE%CE%B5%CE%B9%CF%82%20%CE%BA%CE%B1%CE%B9%20%CE%9C%CE%B7%CF%87%CE%B1%CF%84%CF%81%CE%BF%CE%BD%CE%B9%CE%BA%CE%B1%CC%81%20%CF%83%CF%85%CF%83%CF%84%CE%B7%CC%81%CE%BC%CE%B1%CF%84%CE%B1%20.pdf

[16] https://www.espboards.dev/esp32/ttgo-lora32/

[17] https://www.hellasdigital.gr/electronics/transceivers-and-communications/ttgo-lora32-868-915mhz-sx1276-esp32-oled-display-bluetooth-wifi-lora-el/?sl=en

[18] https://docs.platformio.org/en/latest/boards/espressif32/ttgo-lora32-v1.html

[19] https://lilygo.cc/products/lora-v1-3?srsltid=AfmBOoq8yoUsUhdcOwp-lhDyn1rN3lV_zBffahgQA3JWGuF6SYgrv4oH

[20] https://github.com/LilyGO/TTGO-LORA32/tree/master

[21] DHT11 Humidity & Temperature Sensor datasheet
https://www.alldatasheet.com/html-pdf/2193416/OSEPP/DHT11/172/1/DHT11.html

[22] https://www.hellasdigital.gr/electronics/sensors/light-and-color-sensors/gl5516-light-dependent-resistor-ldr-5mm/

[23] https://robocraze.com/blogs/post/what-is-the-ldr-sensor?srsltid=AfmBOopvTjDMIJrQ9oEDfqY3GeTARSOJKpbPz50XL0L-KRYaBEfPWmGN

[24] https://www.electronicsforu.com/technology-trends/learn-electronics/ldr-light-dependent-resistors-basics

[25] https://www.hellasdigital.gr/electronics/sensors/gps/neo-6m-gps-module-compatible-with-ublox/

[26]https://www.alldatasheet.com/html-pdf/1283987/U-BLOX/NEO-6M/250/1/NEO-6M.html

[27] https://www.edn.com/neo-6m-gps-module/

[28] Κολοβός Αθανάσιος ΜΔΕ «Ανάπτυξη ασύρματου ενεργειακά αυτόνομου συστήματος έγκαιρης προειδοποίησης με FPGA και IOT», ΠΑΡΑΡΤΗΜΑ Θ

[29] What is Industry 4.0? https://www.ibm.com/think/topics/industry-4-0

[30] Industry 4.0 Solutions from SAP
https://www.sap.com/products/scm/industry-4-0/what-is-industry-4-0.html

[31] Trends and Challenges in AIoT/IIoT/IoT Implementation
https://www.mdpi.com/1424-8220/23/11/5074

[32] “Edge-Computing Architectures for Internet of Things (ECAs-IoT)”
Hamdan, S. κ.ά. (2020).
https://pmc.ncbi.nlm.nih.gov/articles/PMC7696529/

[33] IoT Basic, March 12, 2020, What is the Technology Behind LoRa Frequency
https://www.mokosmart.com/lora-frequency/

[34] John Gieske Principal Engineer, Different Types of Sensors and Their Applications, May 20, 2024 https://nybsys.com/types-of-sensors/

[35] Abhishek Singh, 05/02/2025, Light Dependent Resistor (LDR) / Photoresistor Circuit Diagram & Working
https://hackatronic.com/light-dependent-resistor-ldr-photoresistor-circuit/

[36] Visual Code Studio, The open-source AI code editor
https://code.visualstudio.com/

[37] ritwickdey / vscode-live-server https://github.com/ritwickdey/vscode-live-server

ΠΑΡΑΡΤΗΜΑ Α

Α.1 Ασύρματες Επικοινωνίες – Τεχνικές λεπτομέρειες

Α.1.1 Περιγραφή του τρόπου επικοινωνίας

Η διαδικασία μπορεί να περιγραφεί σε διαδοχικά στάδια:

- **Κωδικοποίηση πληροφορίας (Data Encoding)** – Μετατροπή των ψηφιακών δεδομένων σε σήμα κατάλληλο για μετάδοση.
- **Διαμόρφωση σήματος (Modulation)** – Εφαρμογή τεχνικής μεταβολής παραμέτρων του κύματος-φορέα (πλάτος, συχνότητα, φάση) ώστε να «μεταφέρει» την πληροφορία. Τεχνικές όπως **FSK (Frequency Shift Keying)**, **QAM (Quadrature Amplitude Modulation)** ή **CSS (Chirp Spread Spectrum)** χρησιμοποιούνται σε διάφορα πρωτόκολλα.
- **Μετάδοση (Transmission)** – Εκπομπή του διαμορφωμένου σήματος μέσω κεραίας στην αντίστοιχη συχνότητα λειτουργίας.
- **Λήψη (Reception)** – Η κεραία του δέκτη συλλαμβάνει το σήμα, ακόμη και αν αυτό έχει υποστεί εξασθένηση ή θόρυβο.
- **Αποδιαμόρφωση (Demodulation)** – Επαναφορά των παραμέτρων του κύματος-φορέα στην αρχική τους κατάσταση, ώστε να εξαχθεί το αρχικό ψηφιακό σήμα.
- **Αποκωδικοποίηση (Decoding)** – Μετατροπή του σήματος σε χρήσιμη πληροφορία για το σύστημα ή τον χρήστη.

Η οργάνωση της επικοινωνίας μπορεί να ακολουθεί διαφορετικές τοπολογίες:

- **Σημείο-προς-Σημείο (Point-to-Point)**: Μία αποκλειστική σύνδεση μεταξύ δύο συσκευών (π.χ. σύνδεση δύο υπολογιστών μέσω Wi-Fi Direct).
- **Σημείο-προς-Πολλαπλά Σημεία (Point-to-Multipoint)**: Ένας κόμβος επικοινωνεί ταυτόχρονα με πολλούς (π.χ. ένας δρομολογητής Wi-Fi προς πολλαπλές συσκευές).
- **Δικτυακή τοπολογία πλέγματος (Mesh Networking)**: Κάθε κόμβος λειτουργεί και ως αναμεταδότης, επεκτείνοντας την κάλυψη του δικτύου και αυξάνοντας την αξιοπιστία. Χρησιμοποιείται σε Zigbee, Thread και ασύρματα αισθητηριακά δίκτυα.

Οι παράγοντες που επηρεάζουν την ποιότητα της επικοινωνίας είναι:

- **Εξασθένηση (Attenuation)**: Μείωση της ισχύος του σήματος με την απόσταση.
- **Θόρυβος (Noise)**: Παρεμβολές από άλλες πηγές που αλλοιώνουν το σήμα.
- **Πολυδιαδρομική διάδοση (Multipath Propagation)**: Αντανάκλαση του σήματος σε επιφάνειες, δημιουργώντας πολλαπλές διαδρομές προς τον δέκτη.
- **Κωδικοποίηση Διόρθωσης Σφαλμάτων (Error Correction Coding)**: Χρήση πλεοναζόντων δεδομένων για ανίχνευση και διόρθωση λαθών κατά τη λήψη.

Με βάση τις σύγχρονες τεχνολογικές τάσεις η εξέλιξη των ασύρματων επικοινωνιών ενσωματώνει:

- **MIMO (Multiple Input Multiple Output)** για ταυτόχρονη χρήση πολλαπλών κεραιών, αυξάνοντας την απόδοση και την εμβέλεια (Wi-Fi 6, 5G).
- **Adaptive Modulation** που προσαρμόζει τη διαμόρφωση ανάλογα με την ποιότητα του καναλιού.
- **Low-Power Wide Area Networks (LPWAN)** όπως LoRa και NB-IoT, για ελάχιστη κατανάλωση ενέργειας και μεγάλη εμβέλεια.
- **Edge Computing** που φέρνει την επεξεργασία πιο κοντά στον αισθητήρα, μειώνοντας τον όγκο δεδομένων που μεταδίδονται.

Ο τρόπος επικοινωνίας που θα επιλεγεί για ένα ασύρματο κατανεμημένο σύστημα αυτοματισμού εξαρτάται από τις απαιτήσεις σε εμβέλεια, κατανάλωση, αξιοπιστία και όγκο δεδομένων, καθώς και από τις περιβαλλοντικές συνθήκες λειτουργίας.

Α.1.2 Πρωτόκολλο επικοινωνίας LoRa

Τεχνικά Χαρακτηριστικά:

- **Τοπολογία:** Point-to-Point ή Point-to-Multipoint (με χρήση broadcast).
- **Συχνότητες λειτουργίας:** 433 MHz, 868 MHz (Ευρώπη), 915 MHz (ΗΠΑ).
- **Bandwidth:** Συνήθως 125 kHz, αλλά μπορεί να είναι και 250/500 kHz.
- **Spreading Factor (SF):** Από SF7 έως SF12 (μεγαλύτερο SF → μεγαλύτερη εμβέλεια, μικρότερος ρυθμός δεδομένων).
- **Coding Rate (CR):** 4/5 έως 4/8 για διόρθωση σφαλμάτων.
- **Μέγιστος ρυθμός δεδομένων:** Περίπου 27 kbps (με SF7 και BW 500 kHz).

Πλεονεκτήματα P2P:

- Δεν απαιτείται δικτυακή υποδομή (gateways, servers).
- Πολύ χαμηλή κατανάλωση ενέργειας.
- Ιδανικό για απλά σενάρια όπως τηλεχειρισμός, μέτρηση σε μικρά δίκτυα αισθητήρων.

Μειονεκτήματα P2P:

- Περιορισμένη δυνατότητα κλιμάκωσης.
- Έλλειψη μηχανισμών ασφαλείας και αυθεντικοποίησης σε βασικό επίπεδο.
- Δεν υπάρχει κεντρική διαχείριση δεδομένων.

Σχετικά με το δίκτυο **LoRaWAN** είναι ένα ανοιχτό πρωτόκολλο δικτύου επιπέδου MAC που ορίζεται από το **LoRa Alliance** και χρησιμοποιεί την τεχνολογία LoRa στο φυσικό επίπεδο. Είναι σχεδιασμένο για μεγάλης κλίμακας δίκτυα χαμηλής κατανάλωσης (LPWAN).

Αρχιτεκτονική:

- **End Devices:** Αισθητήρες/ενεργοποιητές που συλλέγουν δεδομένα.
- **Gateways:** Σταθμοί βάσης που λαμβάνουν σήματα LoRa και τα μετατρέπουν σε IP πακέτα.
- **Network Server:** Επεξεργάζεται, φιλτράρει και διαχειρίζεται τα δεδομένα, αποτρέποντας διπλά πακέτα και εφαρμόζοντας πολιτικές ασφαλείας.
- **Application Server:** Κάνει αποκρυπτογράφηση δεδομένων εφαρμογής και τα προωθεί στις κατάλληλες πλατφόρμες.

Τεχνικές Παράμετροι:

- **Συχνότητες λειτουργίας:** Όπως στο P2P (433, 868, 915 MHz).
- **Bandwidth:** 125 kHz (τυπικό), 250/500 kHz σε ειδικές περιπτώσεις.
- **Spreading Factor:** SF7–SF12, αυτόματα επιλεγόμενο μέσω ADR (Adaptive Data Rate).
- **Ρυθμός δεδομένων:** Από 0.3 kbps έως 50 kbps.
- **Κλάσεις Συσκευών:**
    - **Class A:** Λιγότερη κατανάλωση, δύο παράθυρα λήψης μετά από κάθε μετάδοση.
    - **Class B:** Προγραμματισμένα παράθυρα λήψης.
    - **Class C:** Συνεχής λήψη, χαμηλή καθυστέρηση.

Ασφάλεια:

- Διπλή κρυπτογράφηση AES-128 (NwkSKey για δικτυακό επίπεδο, AppSKey για δεδομένα εφαρμογής).
- Διαδικασία OTAA (Over-The-Air Activation) ή ABP (Activation By Personalization) για σύνδεση.

Πλεονεκτήματα LoRaWAN:

- Κλιμάκωση σε χιλιάδες συσκευές.
- Υψηλή ασφάλεια.
- Κεντρική διαχείριση και δρομολόγηση δεδομένων.
- Εξαιρετική εμβέλεια.

Μειονεκτήματα LoRaWAN:

- Χαμηλός ρυθμός δεδομένων.
- Απαίτηση για υποδομή (gateways, server).
- Πολυπλοκότητα σε σχέση με P2P.

Χαρακτηριστικό	LoRa P2P	LoRaWAN
Υποδομή	Όχι	Ναι
Ασφάλεια	Βασική	Ισχυρή (AES-128)
Κλιμάκωση	Περιορισμένη	Πολύ μεγάλη
Διαχείριση Δεδομένων	Τοπική	Κεντρική
Εμβέλεια	Υψηλή	Υψηλή
Κατανάλωση	Πολύ χαμηλή	Χαμηλή έως μέτρια
Πολυπλοκότητα	Χαμηλή	Υψηλή

Πίνακας Α.1 Σύγκριση LoRa P2P με LoRaWAN

A.1.3 Wi-Fi και UDP – Τεχνικές λεπτομέρειες

Το **Wi-Fi** αποτελεί μία από τις πιο διαδεδομένες τεχνολογίες ασύρματης δικτύωσης, βασισμένη στα πρότυπα **IEEE 802.11**, και χρησιμοποιείται ευρέως για την παροχή πρόσβασης σε τοπικά ασύρματα δίκτυα (WLAN) και στο Διαδίκτυο. Η τεχνολογία αυτή επιτρέπει υψηλούς ρυθμούς μετάδοσης δεδομένων, λειτουργεί κυρίως στις ζώνες συχνοτήτων των **2.4 GHz** και **5 GHz** (ενώ οι νεότερες εκδόσεις υποστηρίζουν και τα **6 GHz** μέσω Wi-Fi 6E) και αξιοποιεί τεχνικές όπως η πολυπλεξία πολλαπλών καναλιών και το **MIMO (Multiple Input Multiple Output)** για αυξημένη χωρητικότητα και εμβέλεια. Στο πλαίσιο των ασύρματων κατανεμημένων συστημάτων και των εφαρμογών IoT, το Wi-Fi παρέχει το πλεονέκτημα της ευρείας διαθεσιμότητας υποδομών και της συμβατότητας με μεγάλο εύρος συσκευών, αν και η κατανάλωση ενέργειας του είναι υψηλότερη σε σχέση με τεχνολογίες χαμηλής ισχύος όπως το LoRa ή το Zigbee.
Στον τομέα της μεταφοράς δεδομένων, το **UDP (User Datagram Protocol)** αποτελεί πρωτόκολλο του επιπέδου μεταφοράς (Transport Layer) που λειτουργεί πάνω από το **IP (Internet Protocol)** και είναι ιδιαίτερα κατάλληλο για εφαρμογές που απαιτούν χαμηλή καθυστέρηση και ταχεία αποστολή δεδομένων. Σε αντίθεση με το TCP (Transmission Control Protocol), το UDP είναι **connectionless**, δεν πραγματοποιεί μηχανισμούς επιβεβαίωσης λήψης, ούτε επανεκπομπή πακέτων σε περίπτωση απώλειας. Αυτή η απλότητα το καθιστά εξαιρετικά αποδοτικό για πραγματικού χρόνου επικοινωνία, καθώς η καθυστέρηση είναι ελάχιστη και η επικεφαλίδα των πακέτων περιορίζεται σε μόλις 8 bytes, αφήνοντας περισσότερο χώρο για το ωφέλιμο φορτίο.

Ωστόσο, η απουσία ελέγχου αξιοπιστίας σημαίνει ότι δεν εγγυάται την παράδοση όλων των πακέτων, κάτι που πρέπει να ληφθεί υπόψη στον σχεδιασμό της εφαρμογής.
Ο συνδυασμός Wi-Fi και UDP χρησιμοποιείται εκτενώς σε σενάρια όπου η άμεση μετάδοση δεδομένων υπερέχει σε σημασία έναντι της πλήρους ακεραιότητας. Παραδείγματα περιλαμβάνουν την ασύρματη μετάδοση βίντεο από drones σε σταθμούς ελέγχου, την αποστολή τηλεμετρικών δεδομένων από αισθητήρες IoT που ενημερώνονται ανά δευτερόλεπτο, καθώς και την επικοινωνία σε βιομηχανικά συστήματα αυτοματισμού όπου η καθυστέρηση μπορεί να επηρεάσει την ασφάλεια ή την απόδοση. Σε τέτοια περιβάλλοντα, η αξιοπιστία μπορεί να ενισχυθεί με μηχανισμούς στο επίπεδο εφαρμογής, όπως η περιοδική αποστολή κρίσιμων πακέτων ή η χρήση κωδικοποίησης διόρθωσης σφαλμάτων (FEC).

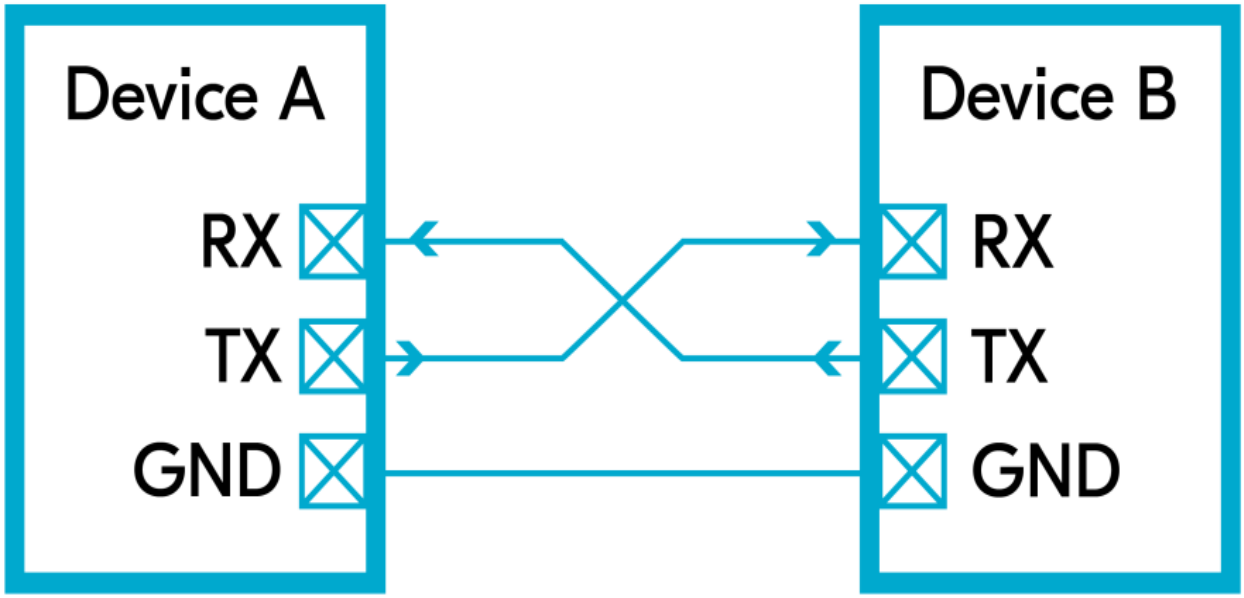
Α.2 Δίαυλοι επικοινωνίας

Για τη διασύνδεση και επικοινωνία του μικροελεγκτή με τα υπόλοιπα μέρη του συστήματος χρησιμοποιήθηκαν τα πρωτόκολλα επικοινωνίας UART, SPI, I²C, ADC και Single-wire.
Ο δίαυλος UART χρησιμοποιείται από το GPS του συστήματος, ο SPI για την επικοινωνία με το LoRa module (SX1276), ο I²C για την επικοινωνία με την OLED οθόνη, ο ADC (αναλογική είσοδος) για τον LDR αισθητήρας και ο Single-wire για τη επικοινωνία με τον αισθητήρα DHT11.

Α.2.1 Δίαυλος UART

Ο **UART (Universal Asynchronous Receiver/Transmitter)** είναι ένας απλός και ευρέως χρησιμοποιούμενος σειριακός δίαυλος επικοινωνίας για σύνδεση δύο συσκευών. Η επικοινωνία γίνεται μέσω δύο γραμμών, **TX** (πομπός) και **RX** (δέκτης), με κοινή γείωση (**GND**). Χαρακτηρίζεται ως **ασύγχρονος**, γιατί δεν χρησιμοποιεί κοινό ρολόι· ο συγχρονισμός επιτυγχάνεται με ειδικά **start και stop bits** που περιβάλλουν κάθε byte. Ένα τυπικό πλαίσιο μετάδοσης είναι η μορφή **8N1** (8 data bits, No parity, 1 stop bit) σε ταχύτητες που καθορίζονται από τον ρυθμό **baud** (π.χ. 9600 baud).

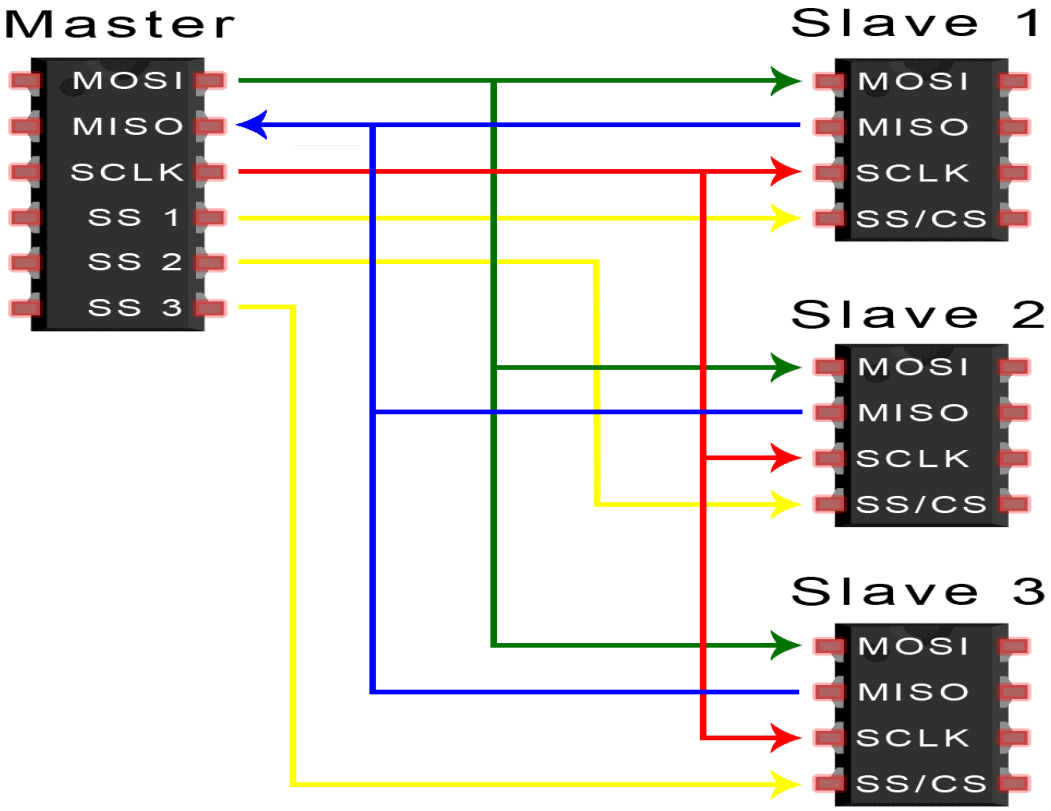
Ο UART επιτρέπει **πλήρη αμφίδρομη επικοινωνία (full duplex)**, αφού οι γραμμές TX και RX είναι ανεξάρτητες. Λόγω της απλότητας, της χαμηλής κατανάλωσης και της αξιοπιστίας του, χρησιμοποιείται εκτεταμένα σε μικροελεγκτές και περιφερειακά (όπως GPS modules, Bluetooth και αισθητήρες), παρέχοντας μια αποτελεσματική λύση για μετάδοση δεδομένων σε μικρές αποστάσεις με λίγα καλώδια.



Εικόνα Α.1 Διασύνδεση διαύλου UART

Α.2.2 Δίαυλος SPI

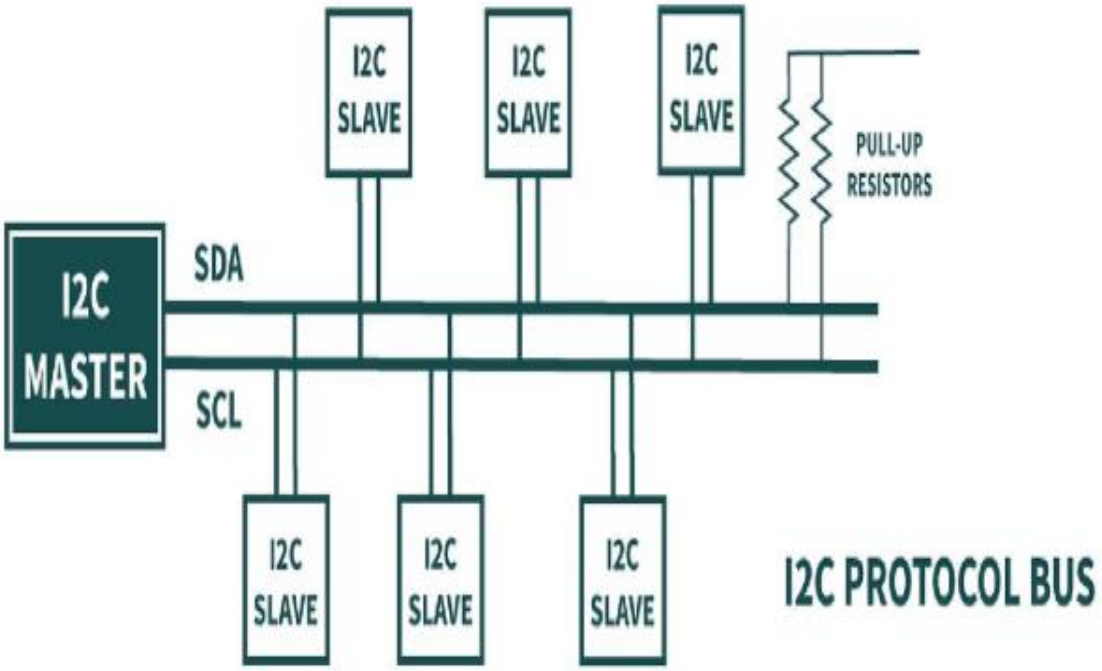
Ο **SPI (Serial Peripheral Interface)** είναι ένας σύγχρονος σειριακός δίαυλος επικοινωνίας που χρησιμοποιείται ευρέως σε ενσωματωμένα συστήματα λόγω της απλότητάς του και της υψηλής ταχύτητας μετάδοσης δεδομένων που προσφέρει. Ακολουθεί αρχιτεκτονική τύπου **Master–Slave**, όπου ο Master (π.χ. μικροελεγκτής) ελέγχει τον ρυθμό επικοινωνίας μέσω του σήματος ρολογιού (**SCK**) και επιλέγει τον εκάστοτε Slave με τη γραμμή **SS/CS** (Slave Select/Chip Select). Η επικοινωνία πραγματοποιείται μέσω τεσσάρων βασικών γραμμών: **MOSI (Master Out Slave In)** για τη μετάδοση δεδομένων από τον Master προς τον Slave, **MISO (Master In Slave Out)** για τη μετάδοση δεδομένων από τον Slave προς τον Master, **SCK** για τον συγχρονισμό και **SS/CS** για την ενεργοποίηση του επιθυμητού Slave. Το SPI υποστηρίζει **full-duplex λειτουργία**, δηλαδή ταυτόχρονη αποστολή και λήψη δεδομένων, και επιτυγχάνει πολύ υψηλές ταχύτητες σε σύγκριση με άλλους διαύλους όπως το I²C ή το UART. Παρά τα πλεονεκτήματά του, απαιτεί περισσότερες γραμμές σύνδεσης και δεν διαθέτει ενσωματωμένο μηχανισμό διευθυνσιοδότησης, γεγονός που το καθιστά πιο κατάλληλο για επικοινωνία μικρών αποστάσεων σε επίπεδο πλακέτας. Στο πλαίσιο του συστήματος που υλοποιήθηκε, ο δίαυλος SPI χρησιμοποιείται για την επικοινωνία του ESP32 με το ασύρματο module **LoRa (SX1276)**, εξασφαλίζοντας αξιόπιστη και γρήγορη μετάδοση δεδομένων.



Εικόνα Α.2 Διασύνδεση διαύλου SPI

Α.2.3 Δίαυλος I²C

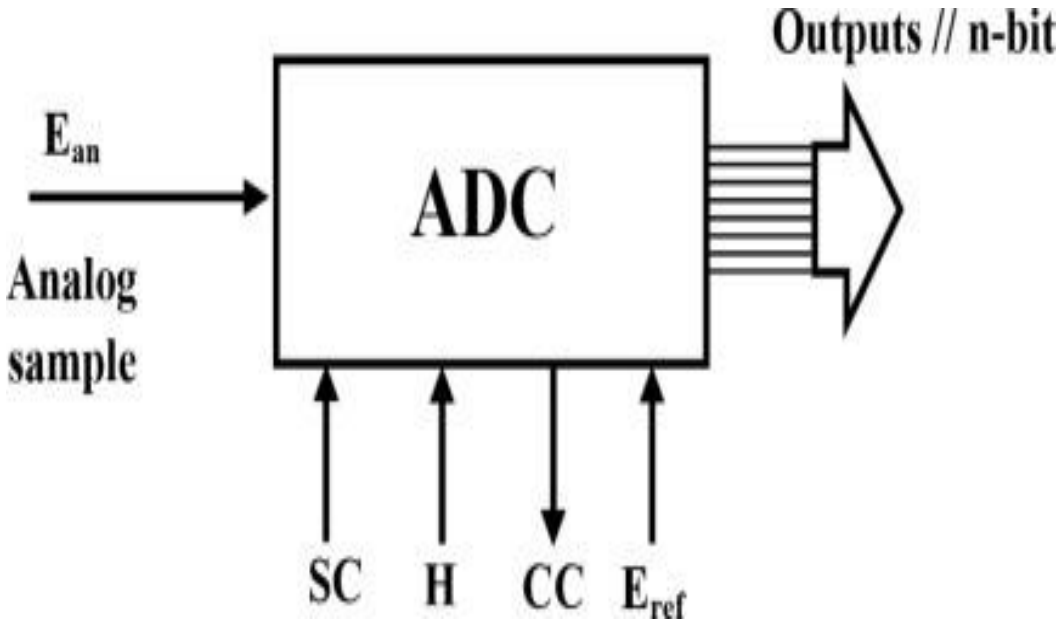
Ο I²C (Inter-Integrated Circuit) είναι ένας σύγχρονος σειριακός δίαυλος επικοινωνίας που αναπτύχθηκε για τη διασύνδεση ολοκληρωμένων κυκλωμάτων με απλό και αποδοτικό τρόπο, χρησιμοποιώντας μόνο δύο γραμμές: τη SDA (Serial Data) για τη μεταφορά δεδομένων και τη SCL (Serial Clock) για τον συγχρονισμό. Λειτουργεί σε αρχιτεκτονική Master–Slave, όπου ο Master (συνήθως ο μικροελεγκτής) παράγει το σήμα ρολογιού και ελέγχει τη ροή της επικοινωνίας, ενώ κάθε Slave αναγνωρίζεται μέσω μιας μοναδικής διεύθυνσης. Η επικοινωνία είναι ημιαμφίδρομη και πραγματοποιείται με τη μορφή πλαισίων (frames) που περιλαμβάνουν διεύθυνση συσκευής, εντολή ανάγνωσης ή εγγραφής και τα αντίστοιχα δεδομένα. Ο I²C υποστηρίζει τη σύνδεση πολλών συσκευών πάνω στον ίδιο δίαυλο, γεγονός που τον καθιστά ιδιαίτερα ευέλικτο για εφαρμογές όπου απαιτείται η ταυτόχρονη λειτουργία πολλών αισθητήρων ή περιφερειακών. Αν και οι ταχύτητές του (συνήθως από 100 kHz έως 400 kHz, ή και υψηλότερα σε Fast/High-Speed modes) είναι χαμηλότερες σε σχέση με το SPI, πλεονεκτεί σε απλότητα配 καλωδίωσης και σε δυνατότητα επέκτασης. Στο σύστημα που υλοποιήθηκε, ο δίαυλος I²C χρησιμοποιείται για την επικοινωνία του ESP32 με την οθόνη OLED SSD1306, επιτρέποντας την απεικόνιση πληροφοριών σε πραγματικό χρόνο με μόλις δύο καλώδια δεδομένων.



Εικόνα Α.3 Διασύνδεση διαύλου I²C

A.2.4 ADC

Ο ADC (Analog to Digital Converter – Μετατροπέας Αναλογικού σε Ψηφιακό) δεν είναι ακριβώς δίαυλος επικοινωνίας, αλλά ένας ενσωματωμένος μηχανισμός των μικροελεγκτών που επιτρέπει τη μέτρηση αναλογικών σημάτων και τη μετατροπή τους σε ψηφιακές τιμές ώστε να μπορούν να επεξεργαστούν από το σύστημα. Η λειτουργία του βασίζεται στη δειγματοληψία της αναλογικής τάσης που εφαρμόζεται σε μια είσοδο και στην αντιστοίχισή της σε έναν ακέραιο αριθμό μέσα σε ένα προκαθορισμένο εύρος (π.χ. 0–4095 για 12-bit ADC). Με τον τρόπο αυτό, αισθητήρες που παράγουν αναλογικά σήματα, όπως φωτοαντιστάσεις (LDR), θερμίστορ ή ποτενσιόμετρα, μπορούν να διασυνδεθούν άμεσα με τον μικροελεγκτή. Στον ESP32, ο ADC υποστηρίζει πολλαπλές εισόδους και ανάλυση έως 12 bit, προσφέροντας ικανοποιητική ακρίβεια για εφαρμογές παρακολούθησης και ελέγχου. Στο σύστημα που αναπτύχθηκε, ο δίαυλος ADC χρησιμοποιείται για την ανάγνωση της τιμής της φωτοαντίστασης (LDR), μετατρέποντας τη μεταβολή της φωτεινότητας σε αντίστοιχη ψηφιακή τιμή, η οποία στη συνέχεια υπολογίζεται ως ποσοστό φωτεινότητας και απεικονίζεται στην οθόνη OLED αλλά και αποστέλλεται μέσω LoRa.



Εικόνα Α.4 Διασύνδεση ADC

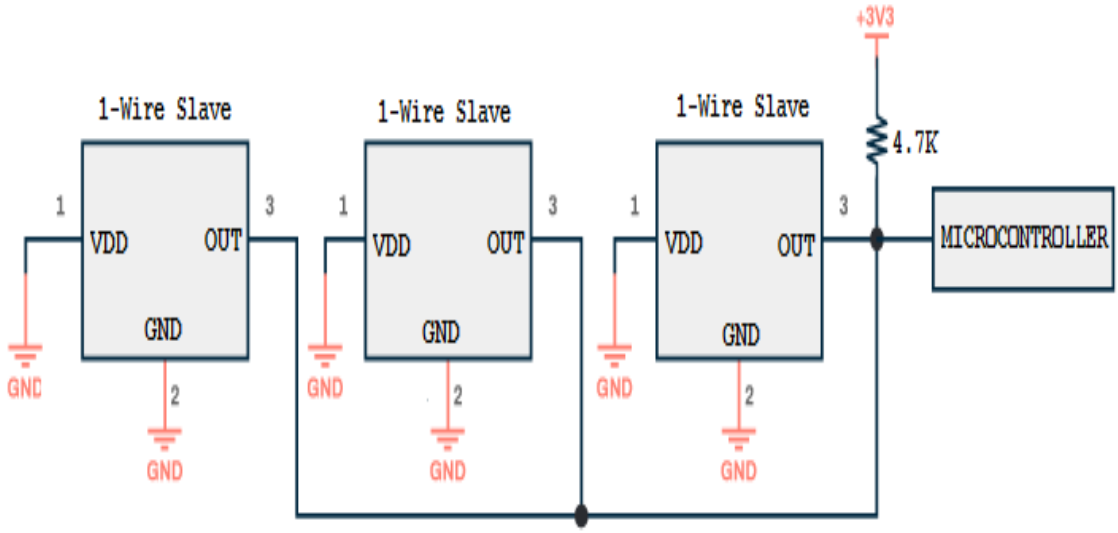
A.2.5 Single – Wire

Το **Single-Wire πρωτόκολλο επικοινωνίας** είναι μια μέθοδος ανταλλαγής δεδομένων ανάμεσα σε έναν μικροελεγκτή και έναν αισθητήρα ή περιφερειακή συσκευή χρησιμοποιώντας **μόνο μία γραμμή δεδομένων**, εκτός από την τροφοδοσία και τη γείωση. Η γραμμή αυτή είναι **διπλής κατεύθυνσης** (bidirectional) και χρησιμοποιείται τόσο για την αποστολή όσο και για τη λήψη δεδομένων, με τη βοήθεια κατάλληλου χρονοπρογραμματισμού και pull-up αντιστάσεων. Η επικοινωνία γίνεται ασύγχρονα και βασίζεται στη διάρκεια των παλμών υψηλού ή χαμηλού επιπέδου, που αντιστοιχούν σε λογικά **0** ή **1**.

Ένα χαρακτηριστικό παράδειγμα υλοποίησης του πρωτοκόλλου είναι ο αισθητήρας **DHT11/DHT22** για μέτρηση θερμοκρασίας και υγρασίας. Στην περίπτωση αυτή, ο μικροελεγκτής ξεκινά την επικοινωνία με ένα σήμα έναρξης (start signal), και ο αισθητήρας απαντά στέλνοντας μια ακολουθία bit που περιέχει τις μετρήσεις. Κάθε bit μεταδίδεται με τη μορφή παλμών συγκεκριμένης διάρκειας, ενώ η χρονική ακρίβεια είναι κρίσιμη για τη σωστή αποκωδικοποίηση.

Τα πλεονεκτήματα του Single-Wire πρωτοκόλλου είναι η **απλότητα στην καλωδίωση** (μόνο ένα καλώδιο δεδομένων) και το **χαμηλό κόστος**, καθιστώντας το ιδανικό για μικρούς αισθητήρες χαμηλής ταχύτητας. Ωστόσο, παρουσιάζει περιορισμούς σε **ταχύτητα μετάδοσης**, σε **απόσταση επικοινωνίας**, καθώς και σε **αξιοπιστία** σε περιβάλλοντα με ηλεκτρικό θόρυβο, σε σύγκριση με πιο εξελιγμένα πρωτόκολλα όπως το I²C ή το SPI.

Στο πλαίσιο του συστήματος που υλοποιήθηκε, το πρωτόκολλο Single-Wire χρησιμοποιείται για την επικοινωνία του ESP32 με τον αισθητήρα **DHT11**, ώστε να συλλέγονται οι μετρήσεις θερμοκρασίας και υγρασίας και να μετατρέπονται σε ψηφιακή μορφή για περαιτέρω επεξεργασία και μετάδοση.



Εικόνα Α.5 Διασύνδεση επικοινωνίας Single Wire

ΠΑΡΑΡΤΗΜΑ Β

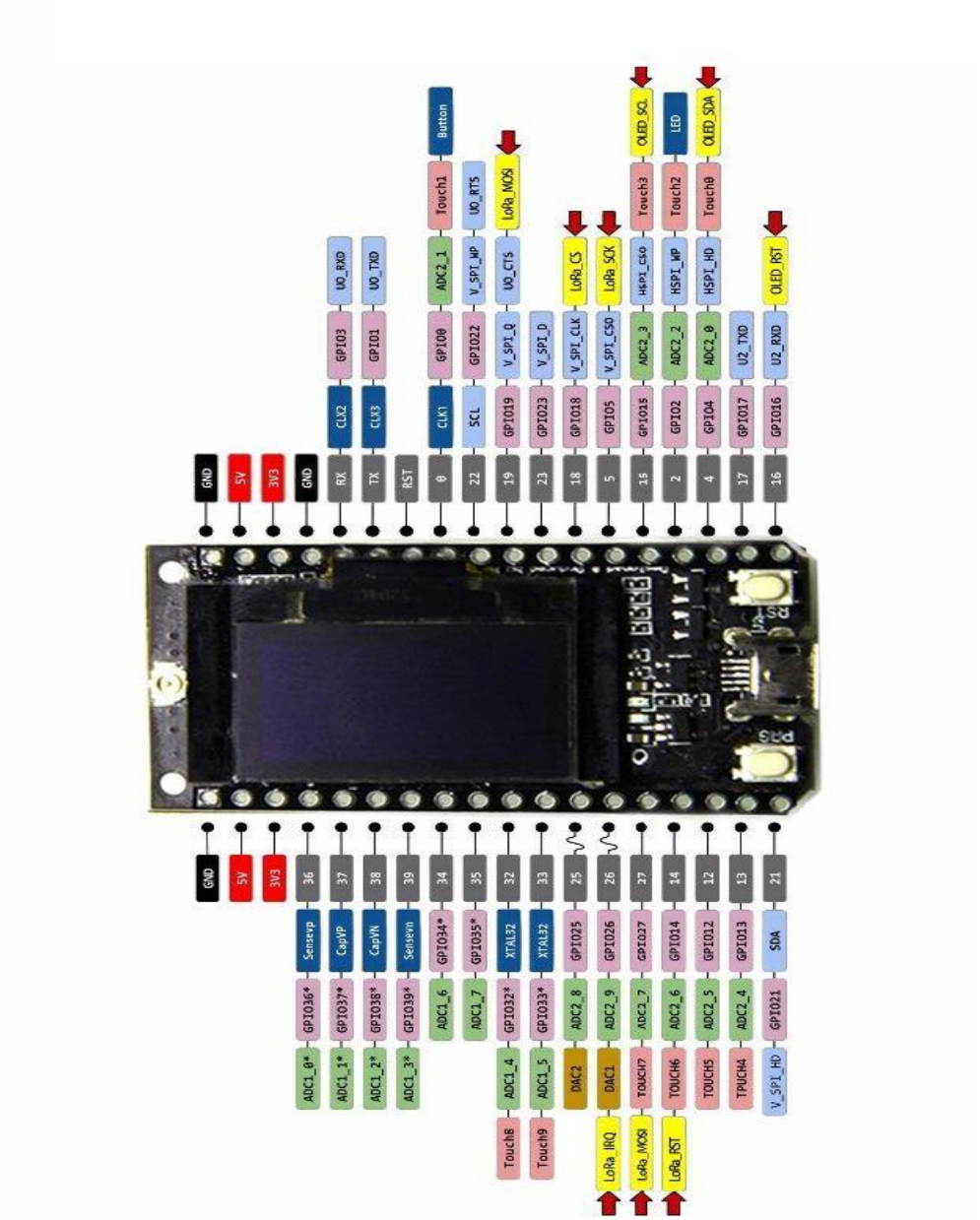
Β.1 TTGO LORA32 OLED V1

This development board provides **digital IO pins**, out of which **can be used as external interrupt pins** and **as analog input pins** .

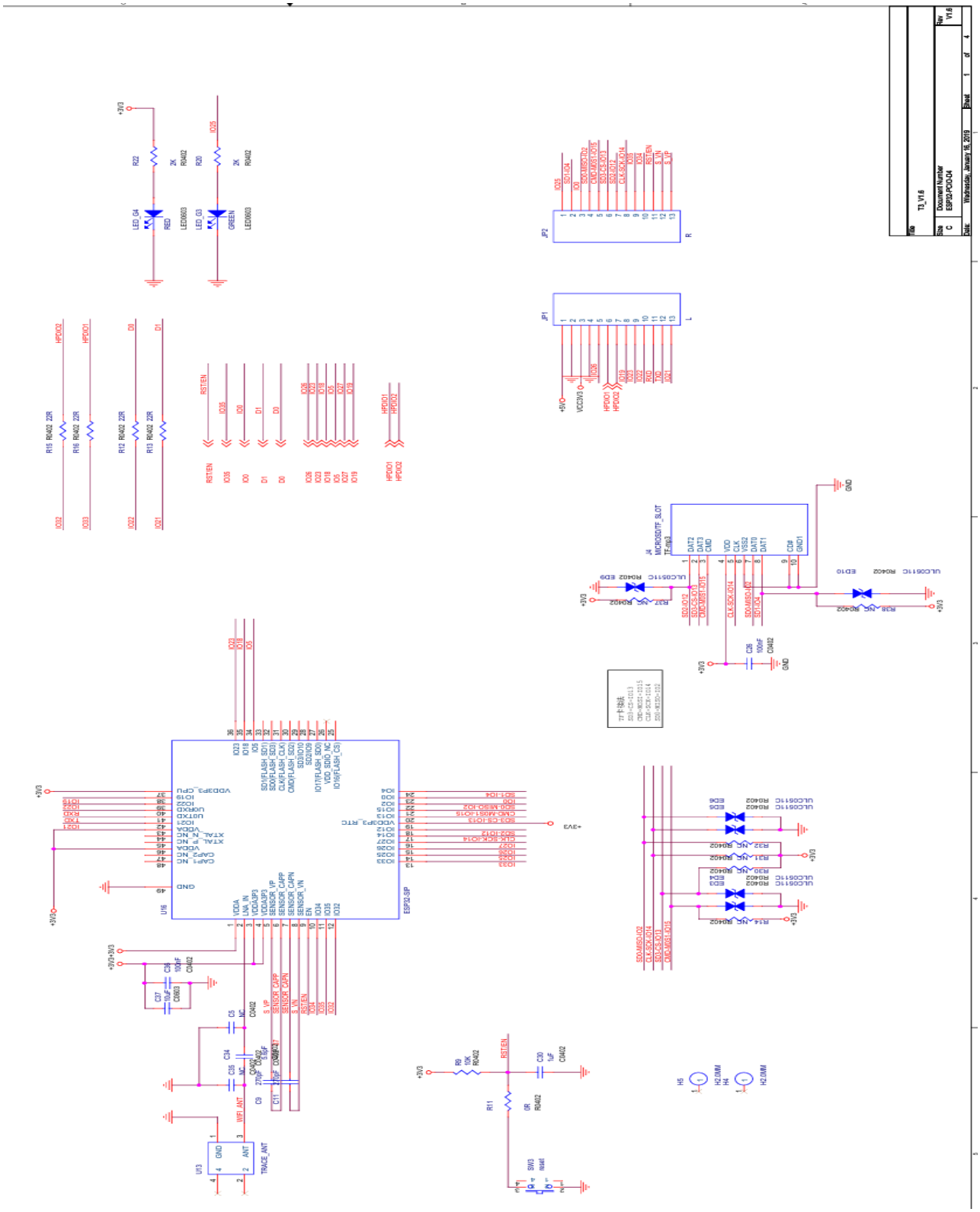
Pin	Function	ESP Pin	Input/Output	Description
1	OLED RST	GPIO16	output	OLED Reset (V1.0 only)
2	OLED SDA	GPIO4	bidirectional	I2C Data Line (V1.0 only)
3	OLED SDA	GPIO21	bidirectional	I2C Data Line (V1.2+ only)
4	OLED SCL	GPIO15	bidirectional	I2C Clock Line (V1.0 only)
5	OLED SCL	GPIO22	bidirectional	I2C Clock Line (V1.2+ only)
6	SDCard CS	GPIO13	output	SPI Chip Select (V1.6+)
7	SDCard MOSI	GPIO15	bidirectional	SPI Master Out Slave In (V1.6+)
8	SDCard MISO	GPIO2	bidirectional	SPI Master In Slave Out (V1.6+)
9	SDCard SCLK	GPIO14	bidirectional	SPI Clock Line (V1.6+)
10	DS3231 SDA	GPIO21	bidirectional	I2C Data Line (V1.2 T-Fox only)
11	DS3231 SCL	GPIO22	bidirectional	I2C Clock Line (V1.2 T-Fox only)
12	LORA MOSI	GPIO27	bidirectional	SPI Master Out Slave In
13	LORA MISO	GPIO19	bidirectional	SPI Master In Slave Out
14	LORA SCLK	GPIO5	bidirectional	SPI Clock Line
15	LORA CS	GPIO18	output	SPI Chip Select
16	LORA RST	GPIO14	output	LoRa Reset (V1.0 only)
17	LORA RST	GPIO23	output	LoRa Reset (V1.2+ only)
18	LORA DIO0	GPIO26	input	LoRa Interrupt Pin

Function Pin Function ESP Pin Pin on ESP32 I/O Input/Output Pin Description Pin Description

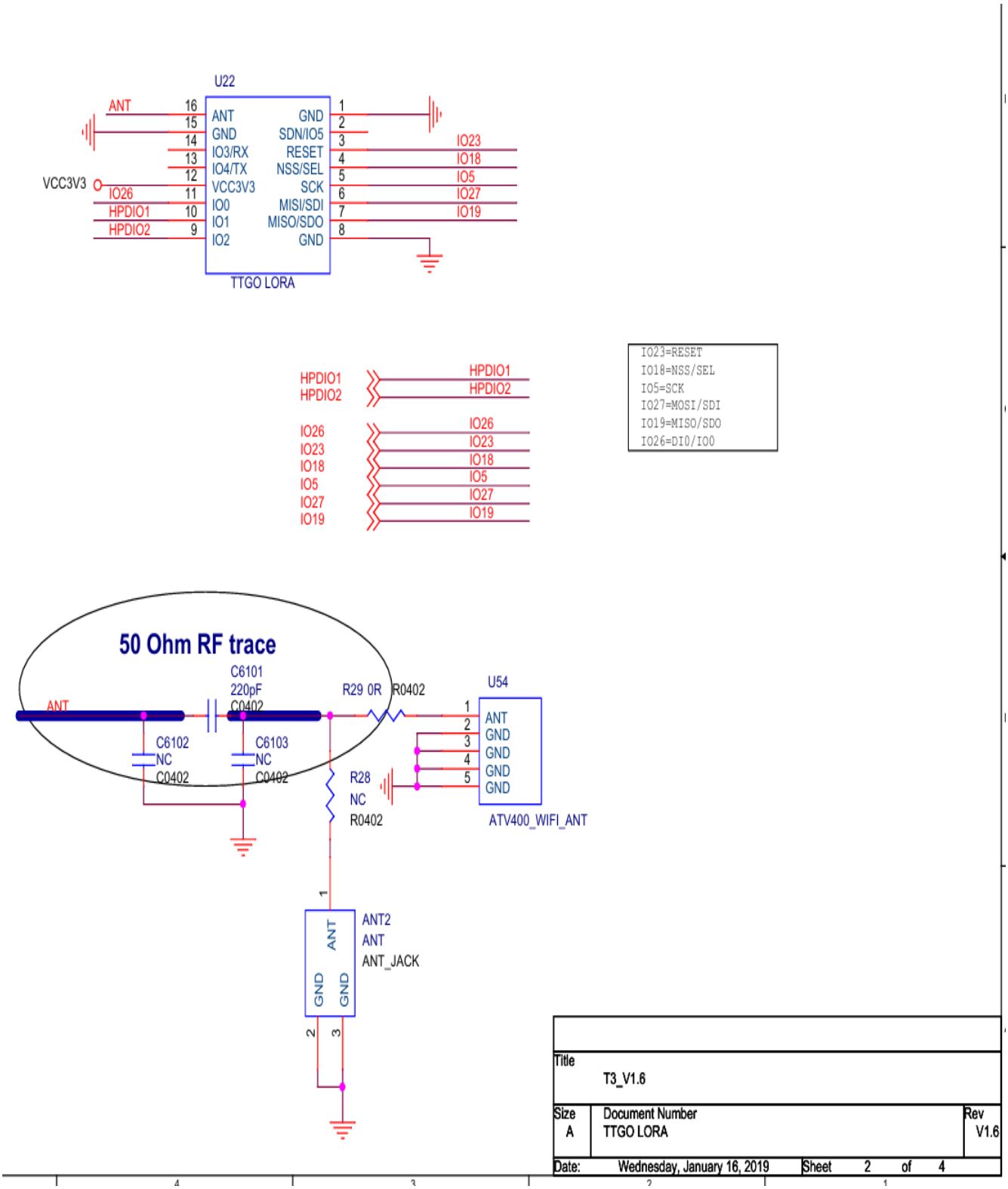
Εικόνα Β.1 Pin Mapping TTGO LORA32 OLED



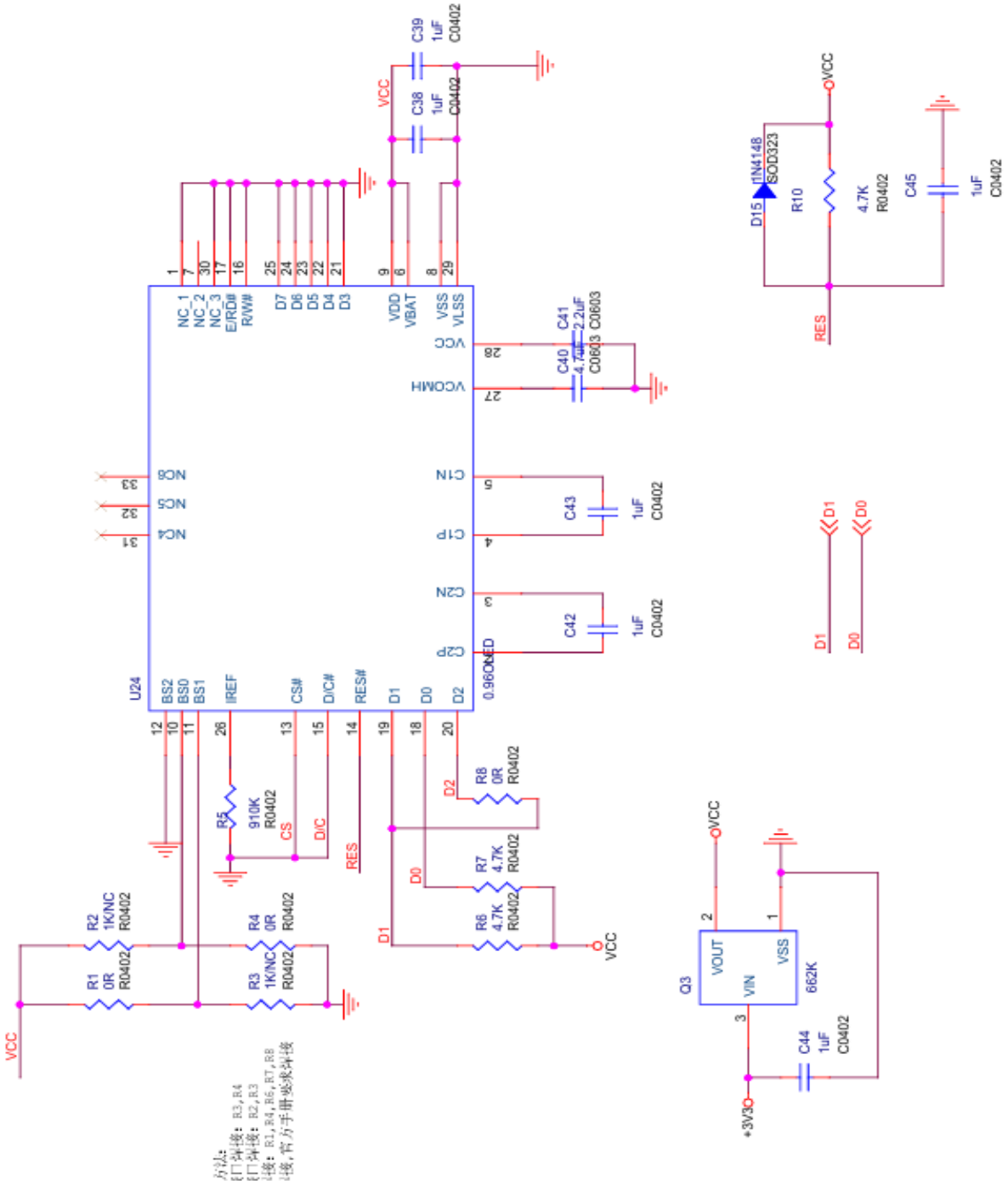
Εικόνα Β.2 PINOUT TTGO LORA32 OLED V1



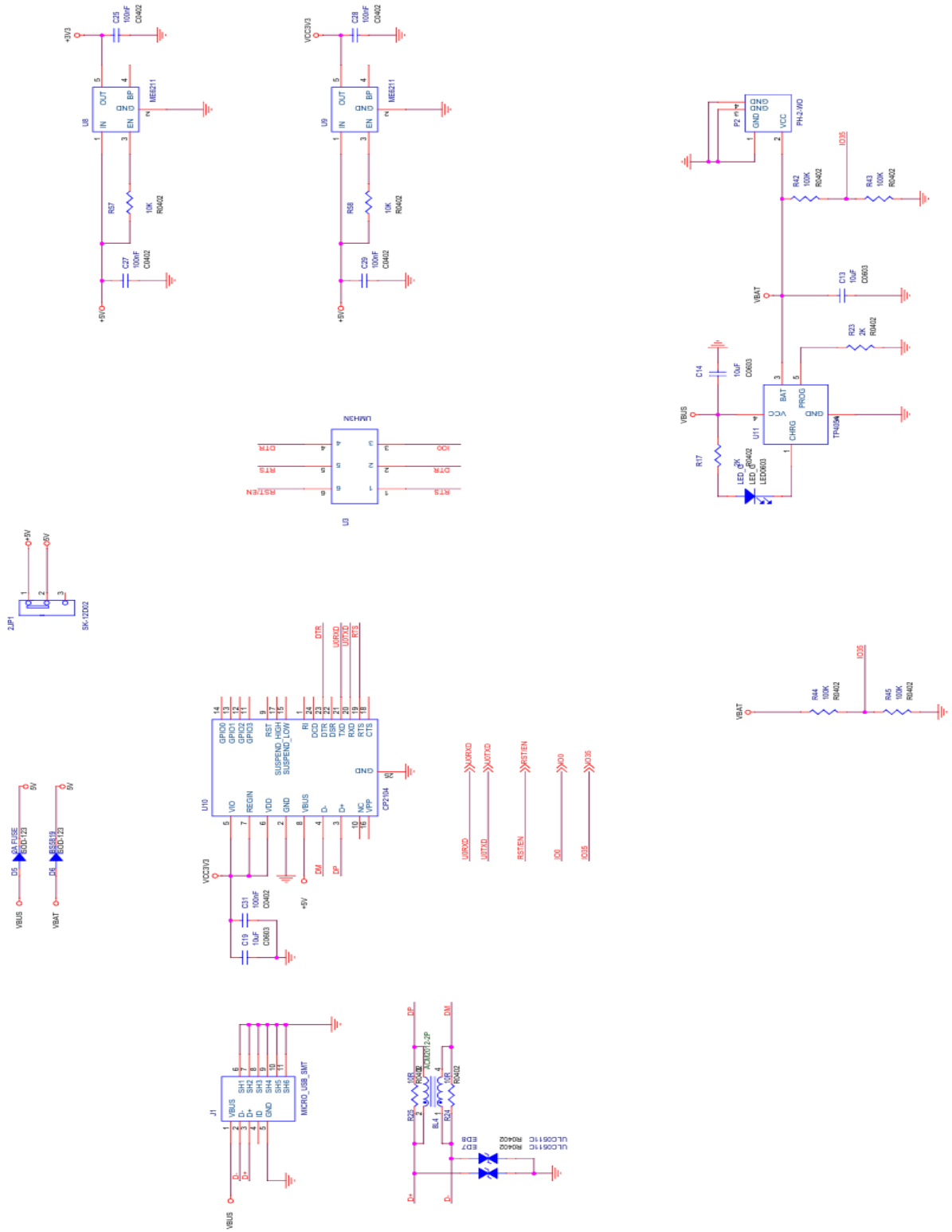
Εικόνα Β.3 Σχηματικό διάγραμμα TTGO LORA32 OLED V1 (1/4)



Εικόνα Β.4 Σχηματικό διάγραμμα TTGO LORA32 OLED V1 (2/4)

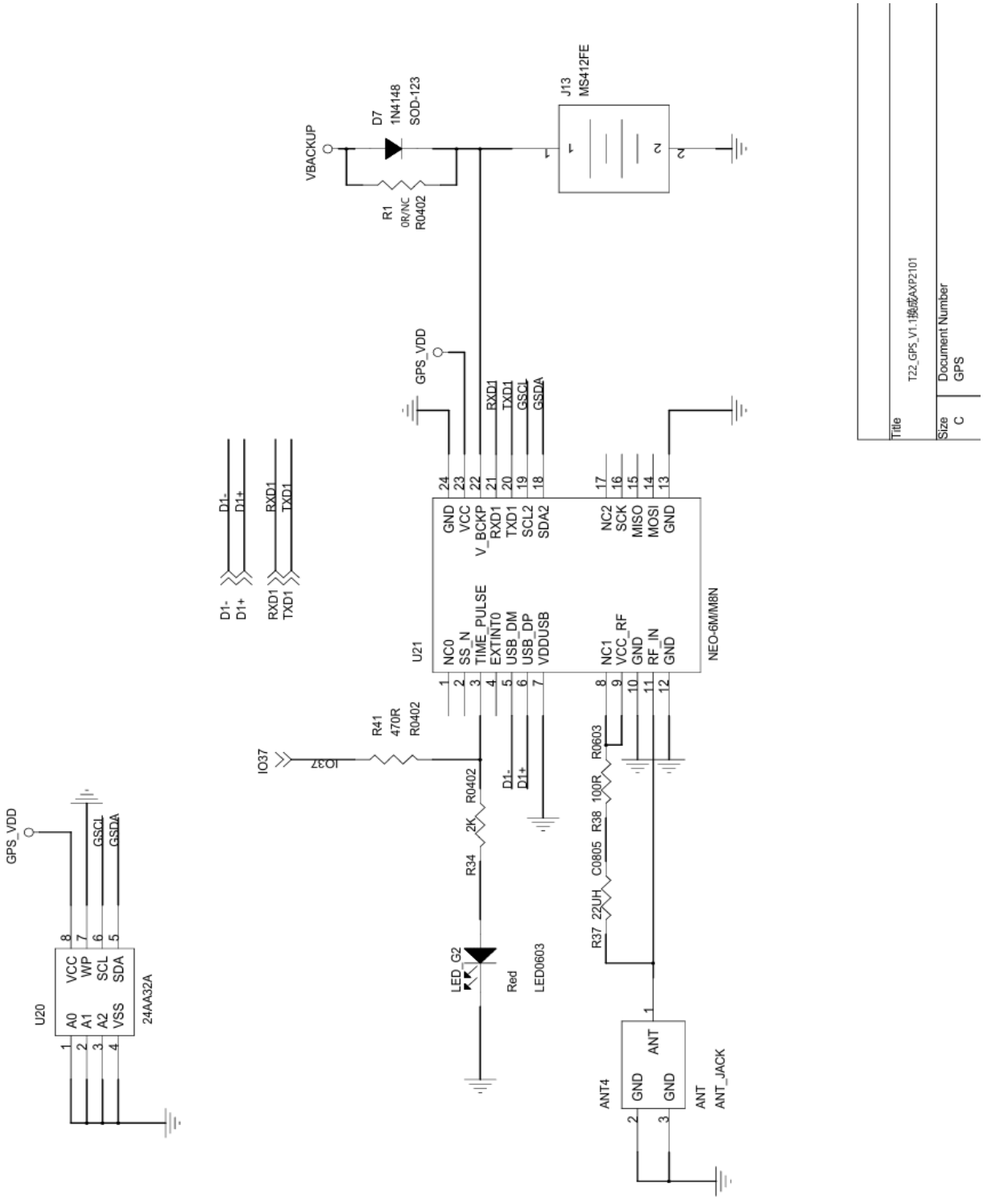


Εικόνα Β.5 Σχηματικό διάγραμμα TTGO LORA32 OLED V1 (3/4)



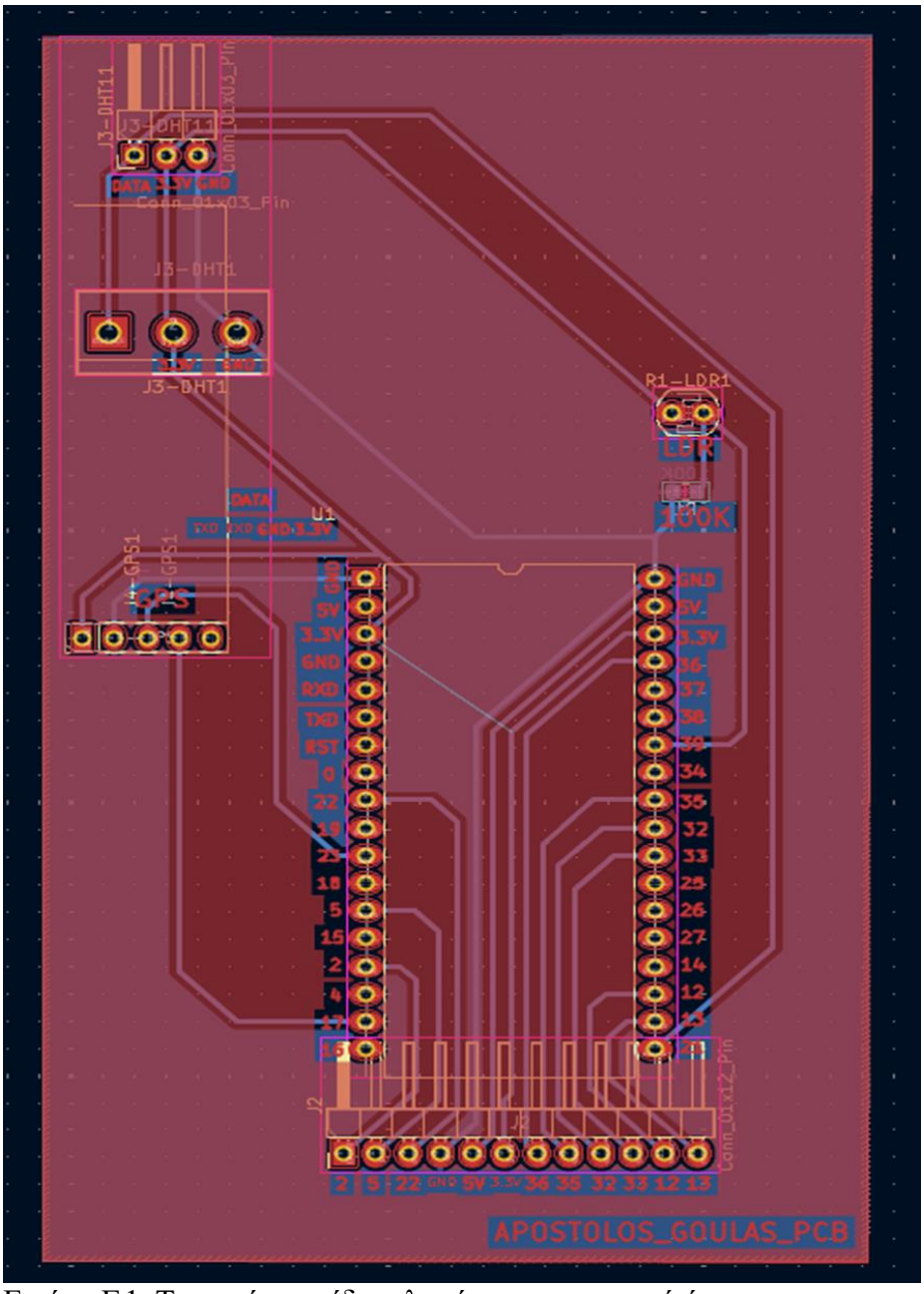
Εικόνα Β.6 Σχηματικό διάγραμμα TTGO LORA32 OLED V1 (4/4)

B.2. NEO 6M GPS module

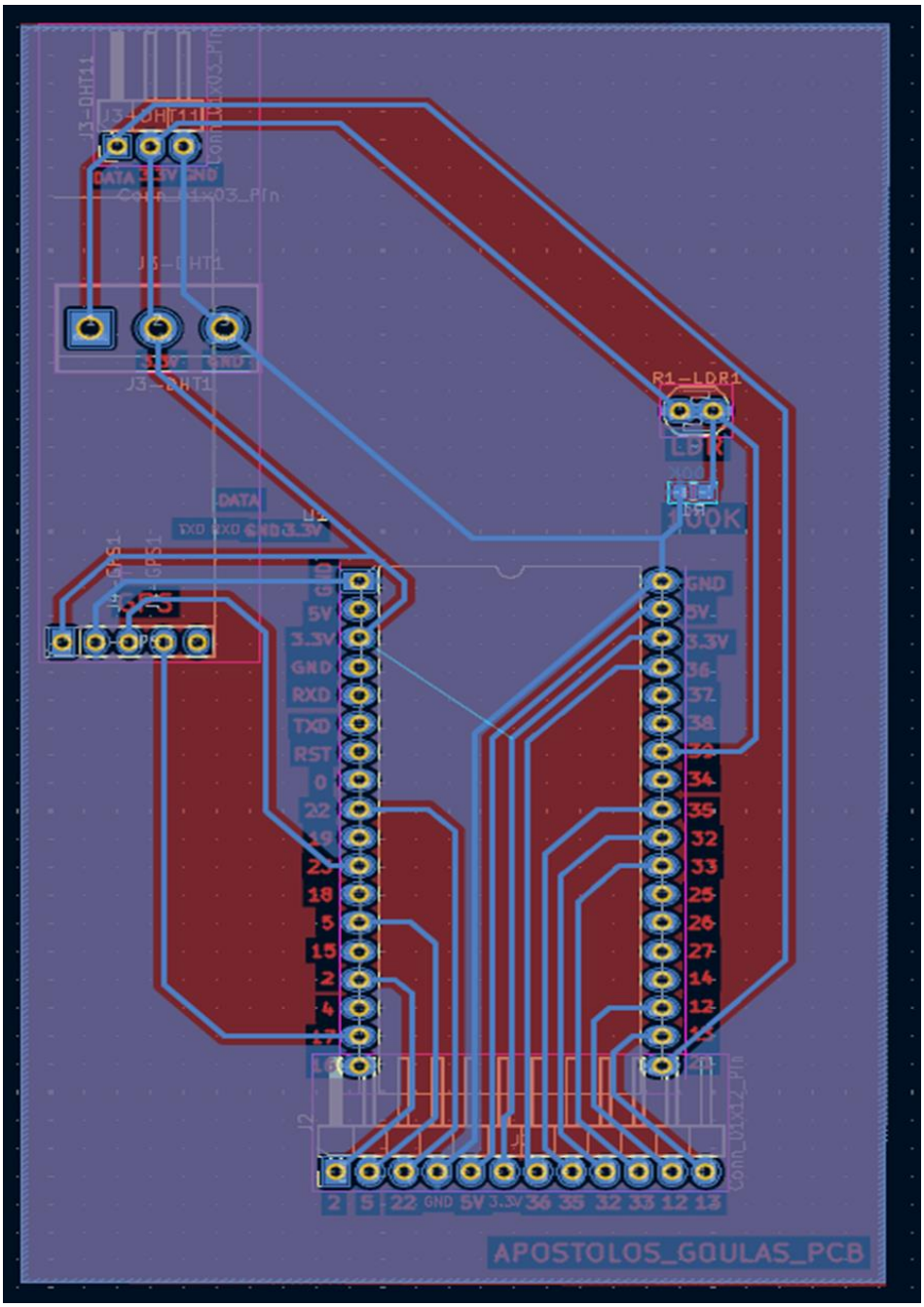


Εικόνα Β.7 Σχηματικό διάγραμμα NEO 6M GPS Module

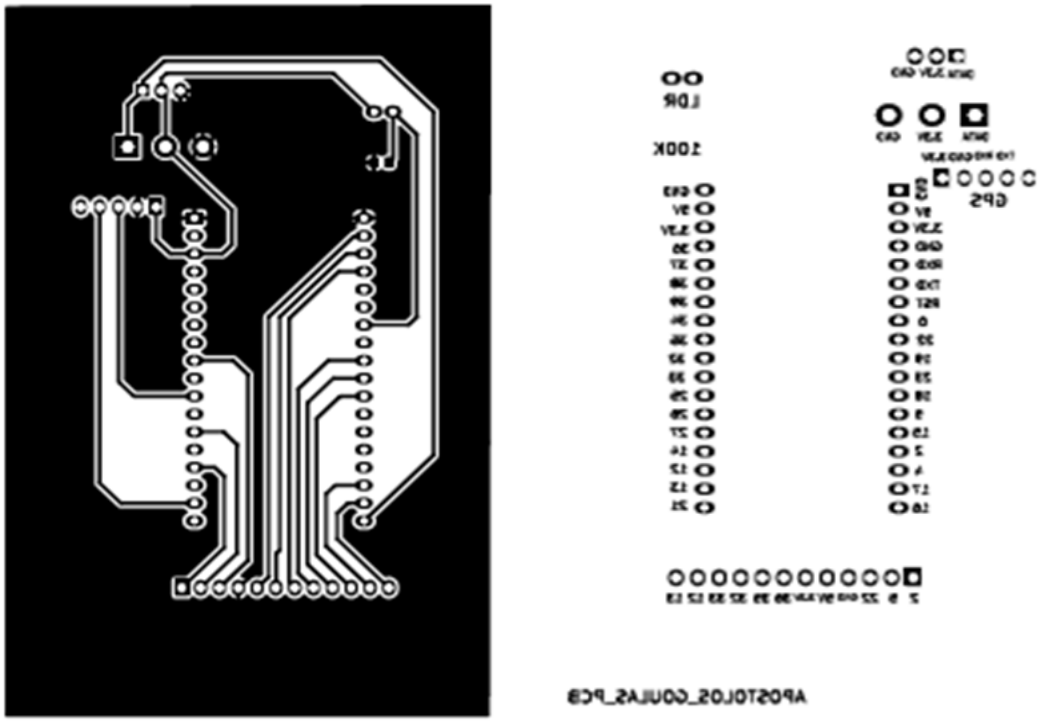
ΠΑΡΑΡΤΗΜΑ Γ



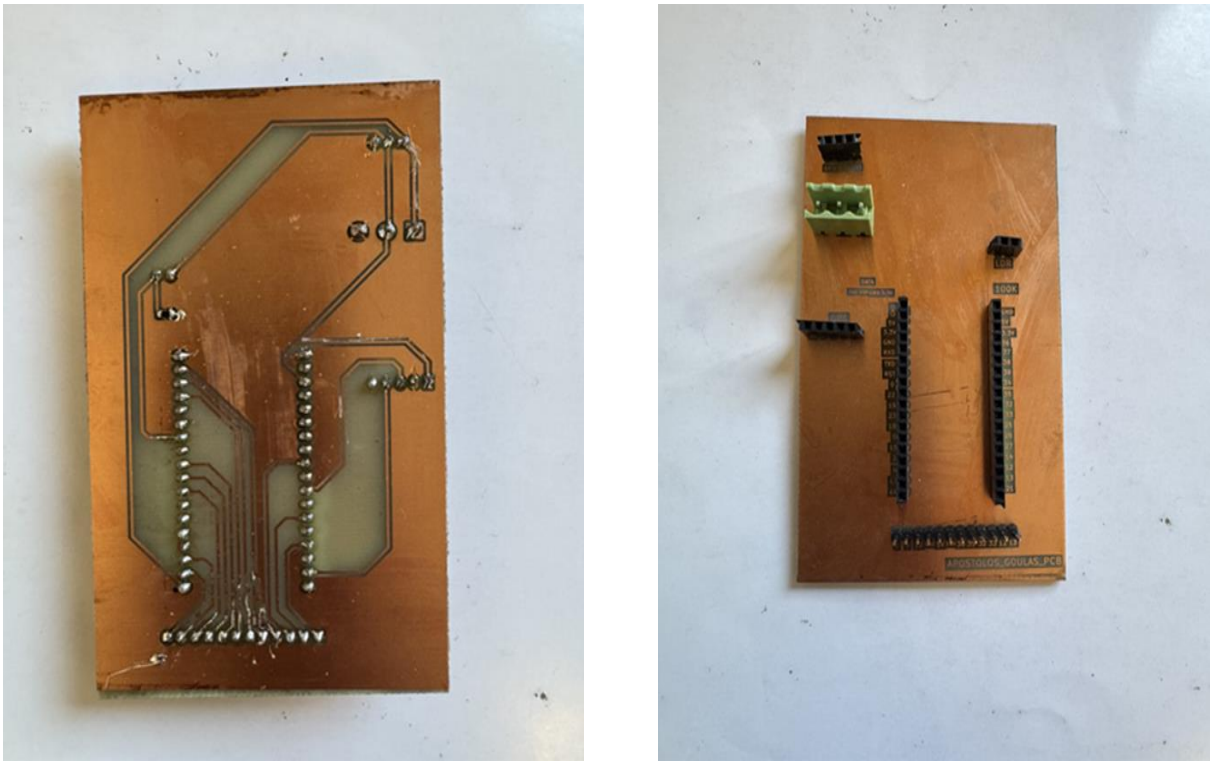
Εικόνα Γ.1. Τυπωμένο σχέδιο πλακέτας – μπροστινή όψη



Εικόνα Γ.2. Τυπωμένο σχέδιο πλακέτας – πίσω όψη



Εικόνα Γ.3. Εξαγωγή drill files σε PDF



Εικόνα Γ.4. Φωτογραφία της τυπωμένης πλακέτας

ΠΑΡΑΡΤΗΜΑ Δ

Δ.1. ESP32 TTGO LoRa32 OLED - DHT11 – GPS

Παρατίθεται ο κώδικας σε C++ που λειτουργεί στον πρώτο περιφερειακό κόμβο ESP32 TTGO LoRa32 OLED - DHT11 – GPS:

```
#include <Arduino.h>                  // Βασική βιβλιοθήκη Arduino/ESP32
#include <SPI.h>                      // Βιβλιοθήκη SPI (LoRa)
#include <Wire.h>                     // Βιβλιοθήκη I2C (OLED)
#include "SSD1306Wire.h"              // Βιβλιοθήκη OLED SSD1306 (ThingPulse)
#include <LoRa.h>                     // Βιβλιοθήκη LoRa
#include "images.h"                   // Αρχείο εικόνων/bitmap για OLED
#include <Adafruit_Sensor.h>          // Βασική βιβλιοθήκη αισθητήρων Adafruit
#include <DHT.h>                      // Βιβλιοθήκη αισθητήρα DHT11
#include <TinyGPSPlus.h>              // GPS NMEA parser
#include <HardwareSerial.h>           // Για Serial1/Serial2 στο ESP32
#include <esp_sleep.h>                // Deep sleep λειτουργίες ESP32

#define SLEEP_TIME 30 * 1000000ULL    // Διάρκεια deep sleep: 30 δευτερόλεπτα
// OLED Display Settings
#define I2C_ADDRESS 0x3C              // Default I2C διεύθυνση OLED
#define OLED_SDA    4                 // SDA pin για OLED
#define OLED_SCL    15                // SCL pin για OLED
#define OLED_RESET  16                // Reset pin για OLED
SSD1306Wire display(0x3C, OLED_SDA, OLED_SCL);  // Αντικείμενο για OLED
void showLogo();                      // Δήλωση συνάρτησης εμφάνισης λογότυπου

// LoRa Module Pins για TTGO LoRa32
#define LORA_SCK   5                  // SPI SCK
#define LORA_MISO  19                 // SPI MISO
#define LORA_MOSI  27                 // SPI MOSI
#define SS 18                          // LoRa Chip Select
#define RST 14                         // LoRa Reset
#define DIO0 26                        // LoRa IRQ pin
#define LORA_BAND 868E6                // LoRa συχνότητα (868MHz Ευρώπη)

#define DHTPIN 21                      // Pin δεδομένων του DHT11
#define DHTTYPE DHT11                  // Τύπος αισθητήρα DHT11

DHT dht(DHTPIN, DHTTYPE);              // Αντικείμενο DHT11
// GPS Module (NEO-6M)
#define RXD2 23                        // GPS TX → ESP32 RX
#define TXD2 17                        // GPS RX → ESP32 TX
```

```
HardwareSerial gpsSerial(1);          // Χρήση Serial1 για GPS
TinyGPSPlus gps;                      // Αντικείμενο GPS
char timeStr[20];                     // Buffer για ημερομηνία/ώρα (string)

void setup() {
    Serial.begin(115200);             // Εκκίνηση Serial Monitor στα 115200 baud
    gpsSerial.begin(9600, SERIAL_8N1, RXD2, TXD2); // Εκκίνηση GPS UART στα 9600
baud
    // Ρύθμιση OLED reset: HIGH → LOW → HIGH
    pinMode(OLED_RESET, OUTPUT);      // Ορισμός pin reset OLED
    digitalWrite(OLED_RESET, HIGH);   // HIGH
    delay(100);                       // Καθυστέρηση 100ms
    digitalWrite(OLED_RESET, LOW);    // LOW
    delay(100);                       // Καθυστέρηση 100ms
    digitalWrite(OLED_RESET, HIGH);   // Ξανά HIGH

    Wire.begin(4, 15);                // Εκκίνηση I2C (SDA=4, SCL=15)

    display.init();                   // Αρχικοποίηση OLED
    display.flipScreenVertically();   // Αναστροφή οθόνης (αν χρειάζεται)
    showLogo();                       // Εμφάνιση λογότυπου
    delay(2000);                      // Αναμονή 2s
    display.clear();                  // Καθαρισμός OLED
    display.setFont(ArialMT_Plain_10);// Ρύθμιση γραμματοσειράς
    display.drawString(0, 0, "TTGO LoRa32 and LoRaTx"); // Μήνυμα αρχικοποίησης
    display.display();                // Εμφάνιση στην OLED
    delay(1000);                      // Μικρή καθυστέρηση
    // Εκκίνηση LoRa
    SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI); // Εκκίνηση SPI pins
    LoRa.setPins(SS, RST, DIO0);      // Δήλωση pins LoRa
    if (!LoRa.begin(LORA_BAND)) {     // Προσπάθεια εκκίνησης LoRa
        Serial.println("LoRa init failed!");   // Αποτυχία
        display.drawString(0, 20, "LoRa Failed!"); // Μήνυμα στην OLED
        display.display();            // Εμφάνιση
        while (1);                    // Σταμάτα εδώ
    }
    Serial.println("LoRa Initialized");   // Επιτυχία
    display.drawString(0, 20, "LoRa TX Ready!"); // Μήνυμα στην OLED
    display.display();                // Εμφάνιση
    delay(2000);                      // Μικρή καθυστέρηση

    dht.begin();                      // Εκκίνηση αισθητήρα DHT11
}
```

```
void loop() {
    while (gpsSerial.available()) {   // Όσο υπάρχουν δεδομένα GPS
        gps.encode(gpsSerial.read()); // Δώστα στον TinyGPS++
    }

    if (gps.date.isValid() && gps.time.isValid()) { // Έγκυρη ημερομηνία/ώρα;
        int timeZoneOffset = 3;       // Ζώνη ώρας (+3 ώρες)
        int hourLocal = (gps.time.hour() + timeZoneOffset) % 24; // Μετατροπή UTC
→ τοπική

        snprintf(timeStr, sizeof(timeStr), "%04d-%02d-%02d %02d:%02d:%02d", //
Δημιουργία string ώρας
                 gps.date.year(), gps.date.month(), gps.date.day(),
                 hourLocal, gps.time.minute(), gps.time.second());

        Serial.print("GPS Date & Time: "); // Εμφάνιση στο serial
        Serial.println(timeStr);

        // Ανάγνωση DHT11
        float temperature = dht.readTemperature(); // Θερμοκρασία
        float humidity = dht.readHumidity();       // Υγρασία

        String status = "OK";          // Default κατάσταση

        // Έλεγχος ορίων
        if (isnan(temperature) || isnan(humidity)) {
            status = "ERROR";          // Αποτυχία μέτρησης
        } else if (temperature > 30.0) {
            status = "HIGH TEMP";      // Πολύ ζεστό
        } else if (temperature < 15.0) {
            status = "LOW TEMP";       // Πολύ κρύο
        } else if (humidity > 70.0) {
            status = "HIGH HUM";       // Υγρασία υψηλή
        } else if (humidity < 20.0) {
            status = "LOW HUM";        // Υγρασία χαμηλή
        }

        if (isnan(temperature) || isnan(humidity)) { // Έλεγχος NaN
            Serial.println("Failed to read from DHT sensor!"); // Μήνυμα
σφάλματος
            return;                // Επιστροφή από loop
        }
```

```
// Δημιουργία packet LoRa
        String dataPacket = "[ID:TX1, SENSOR:DHT11, TEMP:" + String(temperature)
+
                            "°C, HUM:" + String(humidity) + "%, STATUS:" + status
+
                            ", TIME:" + String(timeStr) + "]";

        Serial.println("Sending Packet: " + dataPacket); // Log

        int maxRetries = 3;          // Μέγιστες προσπάθειες
        bool ackReceived = false;    // Flag ACK

        for (int attempt = 1; attempt <= maxRetries; attempt++) { // Βρόχος
retries
            Serial.printf("Attempt %d: Sending -> %s\n", attempt,
dataPacket.c_str());

            LoRa.beginPacket();      // Έναρξη packet
            LoRa.print(dataPacket);  // Γράψε δεδομένα
            LoRa.endPacket();        // Στείλε packet

            unsigned long startTime = millis(); // Χρονόμετρο αναμονής
            while (millis() - startTime < 500) { // Αναμονή 500ms
                int packetSize = LoRa.parsePacket(); // Έλεγχος εισερχόμενου
packet
                if (packetSize) {    // Αν υπάρχει απάντηση
                    String ackResponse = ""; // Buffer
                    while (LoRa.available()) { // Διάβασε όλο το payload
                        ackResponse += (char)LoRa.read();
                    }
                    if (ackResponse == "ACK") { // Αν είναι ACK
                        Serial.println("ACK received!");
                        ackReceived = true; // Επιτυχία
                        break;              // Βγες από το loop
                    }
                }
            }

            if (ackReceived) break;  // Αν πήραμε ACK, σταματάμε
            Serial.println("No ACK received, retrying..."); // Αλλιώς retry
        }

        // Εμφάνιση στην OLED
        display.clear();             // Καθαρισμός
```

```
        display.setTextAlignment(TEXT_ALIGN_LEFT); // Στοίχιση
        display.setFont(ArialMT_Plain_10);         // Γραμματοσειρά
        display.drawString(0, 0, ackReceived ? "ACK Received!" : "No ACK!"); //
Μήνυμα ACK
        display.drawString(0, 10, "Temp: " + String(temperature) + " °C");    //
Θερμοκρασία
        display.drawString(0, 20, "Humi: " + String(humidity) + " %");        //
Υγρασία
        display.drawString(0, 30, "STATUS: " + status);                       //
Κατάσταση
        display.drawString(0, 40, "Time: " + String(timeStr));                //
Ώρα
        display.display();           // Ενημέρωση OLED
        delay(5000);                 // Καθυστέρηση 5s
    }
    else {                           // Αν GPS δεν είναι έγκυρο
        Serial.println("Waiting for valid GPS time..."); // Log
        display.clear();             // Καθαρισμός
        display.setFont(ArialMT_Plain_10);              // Γραμματοσειρά
        display.drawString(0, 0, "Waiting for valid GPS time..."); // Μήνυμα
        display.display();           // Ενημέρωση
        delay(1000);                 // Αναμονή 1s
    }

    Serial.println("Entering Deep Sleep for 30 seconds..."); // Log
    display.clear();                 // Καθαρισμός OLED
    display.setFont(ArialMT_Plain_10);// Γραμματοσειρά
    display.drawString(0, 0, "Sleeping for 30s..."); // Μήνυμα
    display.display();               // Ενημέρωση
    delay(2000);                     // Αναμονή 2s

    esp_sleep_enable_timer_wakeup(SLEEP_TIME); // Ρύθμιση wake-up
    esp_deep_sleep_start();         // Είσοδος σε deep sleep (reset)
}

void showLogo() {
  uint8_t x_off = (display.getWidth() - logo_width) / 2;  //Υπολογισμός κέντρου X
  uint8_t y_off = (display.getHeight() - logo_height) / 2;//Υπολογισμός κέντρου Y

  display.clear();                // Καθαρισμός OLED
  display.drawXbm(x_off, y_off, logo_width, logo_height, logo_bits); // Εμφάνιση
bitmap
  display.display();              // Ενημέρωση OLED
}
```

Παρακάτω παρατίθεται η συνάρτηση **void showLogo() που** χρησιμοποιεί το αρχείο images.h για την εμφάνιση του λογότυπου LoRa κατά την εκκίνηση του συστήματος. Το αρχείο **images.h** αποτυπώνεται παρακάτω.

```
// This is the LoRa(tm) image
#include <cstdint>          // Χρήση τυποποιημένων τύπων ακεραίων (uint8_t κ.λπ.)
#define logo_width 99         // Πλάτος εικόνας σε pixels (1-bit per pixel)
#define logo_height 64        // Ύψος εικόνας σε pixels

// Ακατέργαστα δεδομένα XBM 1-bit. PROGMEM = αποθήκευση στη flash (όχι RAM).
// Σε ESP32/Arduino, αυτά είναι κατάλληλα για drawXbm() της βιβλιοθήκης SSD1306.
const uint8_t logo_bits[] PROGMEM= {
   // Κάθε byte περιέχει 8 pixels (LSB-first). Τα bytes διατρέχουν την εικόνα ανά
γραμμή.
  // δυαδικά δεδομένα bitmaps που παρήχθησαν από εργαλείο μετατροπής (XBM
exporter).
  // ↓↓↓ Ακολουθεί το ωφέλιμο payload του bitmap ↓↓↓
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xD7, 0x06,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x98, 0x29,
  0x35, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x64,
  0x00, 0xB4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
  0x0B, 0x00, 0x80, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x01, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0xA8, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0xA0, 0x77, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x7A, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x00, 0x00, 0x00, 0x00, 0xB8, 0x6D,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0xFC, 0x01, 0xFC,
  0xFF, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0xC0, 0x01, 0x06,
  0xFC, 0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00,
  0x08, 0xFC, 0xFF, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00,
  0x00, 0x00, 0xF8, 0xBB, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00,
  0x00, 0x00, 0x00, 0xFC, 0x00, 0x7F, 0x00, 0x00, 0x04, 0x00, 0x00, 0xFC,
  0x00, 0x00, 0xFE, 0x00, 0xFC, 0x00, 0x7E, 0xC0, 0x3F, 0x90, 0x00, 0x00,
  0xFC, 0x00, 0x80, 0xFF, 0x07, 0xFC, 0x00, 0x7E, 0xF0, 0xFF, 0x44, 0x00,
  0x00, 0xFC, 0x00, 0xC0, 0xFF, 0x0F, 0xFC, 0x00, 0x7E, 0xF8, 0xFF, 0x21,
```

```
  0x00, 0x00, 0xFC, 0x00, 0xE0, 0xFF, 0x1F, 0xFC, 0x00, 0x7E, 0xF8, 0xF1,
  0x03, 0x00, 0x00, 0xFC, 0x00, 0xE0, 0x87, 0x1F, 0xFC, 0x00, 0x7F, 0x7C,
  0xF0, 0x01, 0x00, 0x00, 0xFC, 0x00, 0xF0, 0x03, 0x3F, 0xFC, 0xBB, 0x3F,
  0x68, 0xE0, 0x03, 0x00, 0x00, 0xFC, 0x00, 0xF0, 0x01, 0x3E, 0xFC, 0xFF,
  0x1F, 0x00, 0xE0, 0x03, 0x00, 0x00, 0xFC, 0x00, 0xF0, 0x01, 0x3E, 0xFC,
  0xFF, 0x1F, 0x80, 0xFF, 0x01, 0x00, 0x00, 0xFC, 0x00, 0xF8, 0x01, 0x7C,
  0xFC, 0xFF, 0x07, 0xE0, 0xFF, 0x03, 0x00, 0x00, 0xFC, 0x00, 0xF0, 0x01,
  0x3E, 0xFC, 0xFF, 0x03, 0xF8, 0xFF, 0x03, 0x00, 0x00, 0xFC, 0x00, 0xF8,
  0x00, 0x3E, 0xFC, 0xE0, 0x07, 0xFC, 0xE1, 0x03, 0x00, 0x00, 0xFC, 0x00,
  0xF0, 0x01, 0x7E, 0xFC, 0xE0, 0x0F, 0xFC, 0xF0, 0x03, 0x00, 0x00, 0xFC,
  0x00, 0xF0, 0x01, 0x3E, 0xFC, 0xE1, 0x0F, 0x7E, 0xF0, 0x01, 0x00, 0x00,
  0xFC, 0x96, 0xF1, 0x03, 0x3F, 0xFC, 0xC0, 0x1F, 0x3C, 0xF0, 0x03, 0x00,
  0x00, 0xFC, 0xFF, 0xE3, 0x87, 0x1F, 0xF8, 0x80, 0x1F, 0x7C, 0xF8, 0x03,
  0x00, 0x00, 0xFC, 0xFF, 0xE1, 0xFF, 0x1F, 0xFC, 0x80, 0x3F, 0xFC, 0xFF,
  0x01, 0x00, 0x00, 0xFC, 0xFF, 0xC3, 0xFF, 0x0F, 0xFC, 0x00, 0x7F, 0xFC,
  0xFF, 0x03, 0x00, 0x00, 0xFC, 0xFF, 0x81, 0xFF, 0x07, 0xFC, 0x00, 0x7F,
  0xF8, 0xEF, 0x03, 0x00, 0x00, 0xA8, 0x29, 0x00, 0xFE, 0x00, 0x54, 0x00,
  0x12, 0xE0, 0x43, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
  0x01, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0xFC, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x08, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x18, 0x00, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xE0, 0x01, 0x52, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x95, 0x0D, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x03, 0x00, 0x00, 0x02, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x19, 0x00, 0x80, 0x06,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xA4, 0x00, 0xA0,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x58, 0x5D,
  0x6D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
  0xB5, 0x0B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00,
};
```

Δ.2. ESP32 TTGO LoRa32 OLED – LDR SENSOR – GPS

Παρακάτω φαίνεται ο κώδικας σε C++ που λειτουργεί στον δεύτερο περιφερειακό κόμβο ESP32 TTGO LoRa32 OLED - LDR – GPS:

```
#include <Arduino.h>                        // Βασικός πυρήνας Arduino/ESP32
#include <Wire.h>                           // I2C επικοινωνία (για OLED)
#include "SSD1306Wire.h"                    // Βιβλιοθήκη OLED SSD1306 (ThingPulse)
#include <SPI.h>                            // SPI (για LoRa)
#include <LoRa.h>                           // Βιβλιοθήκη LoRa
#include "images.h"                         // Bitmap/logo για την OLED
#include <TinyGPSPlus.h>                    // Βιβλιοθήκη GPS (NMEA parsing)
#include <HardwareSerial.h>                 // Πρόσβαση σε Serial1/2 στο ESP32
#include <esp_sleep.h>                      // Deep sleep API ESP32

#define SLEEP_TIME 30 * 1000000ULL          // Διάρκεια deep sleep = 30s
                                            // 1s = 1.000.000 μικροδευτερόλεπτα

// LDR Sensor Pin
#define LDR_PIN 39                          // ADC pin για LDR (GPIO39, μόνο input)

// Define Min & Max ADC values for LDR
#define LDR_MIN 0                           // Ελάχιστη τιμή ADC (σκοτάδι)
#define LDR_MAX 4095                        // Μέγιστη τιμή ADC (φως)

#define LED_PIN 2                           // Ενσωματωμένο LED στο GPIO2

// LoRa Module Pins
#define LORA_SCK   5                        // SPI SCK για LoRa
#define LORA_MISO  19                       // SPI MISO για LoRa
#define LORA_MOSI  27                       // SPI MOSI για LoRa
#define SS 18                                // LoRa CS (chip select)
#define RST 14                               // LoRa reset pin
#define DIO0 26                              // LoRa DIO0 (IRQ/packet-done)

// OLED Display Settings
#define I2C_ADDRESS 0x3C                    // Default διεύθυνση I2C OLED
#define OLED_SDA    4                       // SDA pin OLED
#define OLED_SCL    15                      // SCL pin OLED
#define OLED_RST  16                       // Reset pin OLED
SSD1306Wire display(0x3C, OLED_SDA, OLED_SCL); // Δημιουργία αντικειμένου οθόνης
void showLogo();                            // Δήλωση συνάρτησης εμφάνισης logo
```

```
// GPS Module (NEO-6M)
#define RXD2 23                          // GPS TX → ESP32 RX (GPIO23)
#define TXD2 17                          // GPS RX → ESP32 TX (GPIO17)
HardwareSerial gpsSerial(1);             // Χρήση Serial1 για GPS
TinyGPSPlus gps;                         // Parser GPS
char timeStr[20];                        // Buffer "YYYY-MM-DD hh:mm:ss"

void setup() {
  Serial.begin(115200);                  // Εκκίνηση Serial monitor
  gpsSerial.begin(9600, SERIAL_8N1, RXD2, TXD2); // Εκκίνηση GPS UART στα 9600 baud

  Serial.println("TTGO and LDR");        // Μήνυμα εκκίνησης

  // Configure OLED reset: HIGH → LOW → HIGH
  pinMode(OLED_RST, OUTPUT);             // ΠΡΟΣΟΧΗ: πιθανό typo — έχεις ορίσει OLED_RESET, όχι OLED_RST
  digitalWrite(OLED_RST, HIGH);          // Θέσε reset pin HIGH
  delay(100);                            // Μικρή καθυστέρηση
  digitalWrite(OLED_RST, LOW);           // Reset χαμηλά
  delay(100);                            // Καθυστέρηση
  digitalWrite(OLED_RST, HIGH);          // Επιστροφή HIGH

  // Initialize I2C communication
  Wire.begin(4, 15);                     // Εκκίνηση I2C με SDA=4, SCL=15

  // Initialize OLED display
  display.init();                        // Αρχικοποίηση OLED
  display.flipScreenVertically();        // Αναστροφή κάθετη (αν χρειάζεται)
  showLogo();                            // Εμφάνιση λογότυπου
  delay(2000);                           // Περίμενε 2s
  display.clear();                       // Καθάρισε οθόνη
  display.setFont(ArialMT_Plain_10);     // Ρύθμιση γραμματοσειράς
  display.drawString(0, 0, "TTGO and LDR"); // Τίτλος
  delay(1000);                           // Μικρή καθυστέρηση
  display.display();                     // Ανανέωση οθόνης

  // Initialize LoRa
  Serial.println("Starting LoRa...");    // Log για LoRa
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI); // Εκκίνηση SPI με custom pins
  LoRa.setPins(SS, RST, DIO0);           // Δήλωση pins στο LoRa αντικείμενο
  if (!LoRa.begin(868E6)) {              // Έναρξη LoRa στα 868MHz (Ευρώπη)
    Serial.println("LoRa initialization failed!"); // Αποτυχία init
```

```
    while (1);                              // Μένει εδώ αν αποτύχει
  }

  Serial.println("LoRa Initialized.");     // Επιτυχής αρχικοποίηση
  display.clear();                         // Καθάρισε οθόνη
  display.setFont(ArialMT_Plain_10);       // Γραμματοσειρά
  display.drawString(0, 0, "LoRa Initialized."); // Μήνυμα OLED
  delay(1000);                             // Καθυστέρηση
  display.display();                       // Ανανέωση OLED

  pinMode(LED_PIN, OUTPUT);                // LED ενσωματωμένο ως έξοδος
  delay(2000);                             // Καθυστέρηση πριν την loop
}

void loop()  {
  while (gpsSerial.available()) {          // Όσο υπάρχουν bytes από GPS
    gps.encode(gpsSerial.read());          // Δώστα στον TinyGPS++ parser
  }

  if (gps.date.isValid() && gps.time.isValid()) { // Έγκυρη ημερομηνία & ώρα από
GPS;
    int timeZoneOffset = 3;                // Τοπική ζώνη ώρας (+3, Ελλάδα)
    int hourLocal = (gps.time.hour() + timeZoneOffset) % 24; // Μετατροπή
UTC→τοπική

    snprintf(timeStr, sizeof(timeStr), "%04d-%02d-%02d %02d:%02d:%02d", // Format
ώρας
             gps.date.year(), gps.date.month(), gps.date.day(),
             hourLocal, gps.time.minute(), gps.time.second());

    Serial.print("GPS Date & Time: ");     // Εμφάνιση στο serial
    Serial.println(timeStr);               // Εκτύπωση ώρας

    int ldrValue = analogRead(LDR_PIN);    // Ανάγνωση αναλογικής τιμής LDR (0–
4095)
    int ldrPercent = map(ldrValue, LDR_MIN, LDR_MAX, 0, 100); // Ποσοστό
φωτεινότητας 0–100%

    // Determine light status
    String status = "OK";                  // Προεπιλογή κατάστασης
    if (ldrPercent < 20) {                 // Αν χαμηλή φωτεινότητα
        status = "LOW_LIGHT";              // Θέσε LOW_LIGHT
    } else if (ldrPercent > 80) {          // Αν πολύ φωτεινό
        status = "HIGH_LIGHT";             // Θέσε HIGH_LIGHT
```

```
    }

    // Control built-in LED based on light percentage
    if (ldrPercent < 30) {               // Αν σκοτεινά (<30%)
      digitalWrite(LED_PIN, HIGH);       // Άναψε LED
    } else {
      digitalWrite(LED_PIN, LOW);        // Σβήσε LED
    }

    // Create JSON-like Data Packet
    String dataPacket = "[ID:TX2, SENSOR:LDR, LDR_VALUE:" + String(ldrValue) +
                        ", BRIGHTNESS:" + String(ldrPercent) + "%, STATUS:" +
                        status + ", TIME:" + String(timeStr) + "]"; // Πακέτο για
LoRa
    Serial.println("Sending Packet: " + dataPacket); // Log πακέτου

    // Retry sending packet if no ACK received
    int maxRetries = 3;                  // Μέγιστες προσπάθειες
    bool ackReceived = false;            // Flag ACK

    for (int attempt = 1; attempt <= maxRetries; attempt++) { // Βρόχος retries
        Serial.printf("Attempt %d: Sending -> %s\n", attempt,
dataPacket.c_str()); // Log

        // Send data via LoRa
        LoRa.beginPacket();              // Έναρξη packet
        LoRa.print(dataPacket);          // Εγγραφή δεδομένων
        LoRa.endPacket();                // Αποστολή πακέτου

        // Wait for ACK
        unsigned long startTime = millis(); // Χρονόμετρο αναμονής
        while (millis() - startTime < 1000) { // Αναμονή έως 1000ms για απάντηση
            int packetSize = LoRa.parsePacket(); // Έλεγχος εισερχόμενου πακέτου
            if (packetSize) {                    // Αν υπάρχει πακέτο
                String ackResponse = "";         // Buffer ACK
                while (LoRa.available()) {       // Διάβασε όλο το payload
                    ackResponse += (char)LoRa.read();
                }
                if (ackResponse == "ACK") {      // Έλεγχος αν είναι "ACK"
                    Serial.println("ACK received!"); // Μήνυμα
                    ackReceived = true;          // Θέσε επιτυχία
                    break;                       // Βγες από την αναμονή
                }
            }
```

```
        }

        if (ackReceived) break;                 // Αν πήραμε ACK, σταματάμε
retries
        Serial.println("No ACK received, retrying..."); // Αλλιώς ενημέρωση
    }

    // Display LDR value on OLED
    display.clear();                            // Καθάρισε οθόνη
    display.drawString(0, 0, ackReceived ? "ACK Received!" : "No ACK!"); //
Ενημέρωση ACK
    display.drawString(0, 10, "LDR: " + String(ldrValue));              // Τιμή
ADC
    display.drawString(0, 20, "BRIGHTNESS: " + String(ldrPercent) + "%"); //
Ποσοστό
    display.drawString(0, 30, "Status: " + status);                     //
Κατάσταση
    display.drawString(0, 40, "Time: " + String(timeStr));              // Ώρα
    display.drawProgressBar(10, 55, 100, 10, map(ldrValue, 0, 4095, 0, 100)); //
Progress bar
    display.display();                           // Ανανέωση
    delay(10000);                                // Αναμονή 10s στην οθόνη
  }
  else {                                         // Αν ΔΕΝ υπάρχει έγκυρη ώρα GPS
    Serial.println("Waiting for valid GPS time..."); // Ενημέρωση
    display.clear();                             // Καθαρισμός OLED
    display.setFont(ArialMT_Plain_10);           // Γραμματοσειρά
    display.drawString(0, 0, "Waiting for valid GPS time..."); // Μήνυμα αναμονής
    display.display();                           // Ανανέωση
    delay(1000);                                 // Μικρή καθυστέρηση
  }

  Serial.println("Entering Deep Sleep for 30 seconds..."); // Log (το σχόλιο λέει
30s, αλλά είναι 20s πάνω)
  display.clear();                               // Καθαρισμός
  display.setFont(ArialMT_Plain_10);             // Γραμματοσειρά
  display.drawString(0, 0, "Sleeping for 30s..."); // Μήνυμα (οπτικά 30s)
  display.display();                             // Ανανέωση
  delay(1000);                                   // Μικρή καθυστέρηση για να φανεί
το μήνυμα

  // Turn off OLED before sleeping
  display.displayOff();                          // Σβήσε την OLED για
εξοικονόμηση
```

```
  Serial.println("OLED turned off!");           // Log

  esp_sleep_enable_timer_wakeup(SLEEP_TIME);    // Ρύθμιση αφύπνισης σε
SLEEP_TIME (20s)
  esp_deep_sleep_start();                       // Είσοδος σε deep sleep (reset
μετά το ξύπνημα)
}

void showLogo() {
  uint8_t x_off = (display.getWidth() - logo_width) / 2;   // Κεντράρισμα X του
bitmap
  uint8_t y_off = (display.getHeight() - logo_height) / 2; // Κεντράρισμα Y του
bitmap

  display.clear();                              // Καθάρισε οθόνη
  display.drawXbm(x_off, y_off, logo_width, logo_height, logo_bits); // Ζωγράφισε
bitmap
  display.display();                            // Εμφάνισε στην OLED
}
```

Η συνάρτηση **void showLogo()** χρησιμοποιεί το αρχείο images.h για την εμφάνιση του λογότυπου LoRa κατά την εκκίνηση του συστήματος. Το αρχείο **images.h** είναι το ίδιο που αναφέρεται και στο ΠΑΡΑΡΤΗΜΑ Δ.1**.**

Δ.3. ESP32 TTGO LoRa32 OLED – RECIEVER

Παρακάτω παρατίθεται ο κώδικας σε C++ που τρέχει στον δέκτη ESP32 TTGO LoRa32 OLED - RECIEVER:

```
#include <Arduino.h>                              // Πυρήνας Arduino/ESP32
#include <SPI.h>                                  // SPI (για LoRa)
#include <Wire.h>                                 // I2C (για OLED)
#include "SSD1306Wire.h"                          // Βιβλιοθήκη OLED SSD1306
(ThingPulse)
#include <LoRa.h>                                 // Βιβλιοθήκη LoRa
#include "images.h"                               // Bitmap/logo για OLED
#include <WiFi.h>                                 // Wi-Fi stack ESP32
#include <WiFiUdp.h>                              // UDP πάνω από Wi-Fi

const char* ssid = "COSMOTE-167230-2.4G";         // SSID Wi-Fi δικτύου
const char* password = "goulas@1995";             // Κωδικός Wi-Fi (προσοχή: εμφανές
στο source)

const char* serverIP = "192.168.1.8";             // IP του PC που θα λάβει τα UDP
πακέτα
const int serverPort = 8888;                      // Θύρα UDP στον υπολογιστή
WiFiUDP udp;                                      // UDP socket αντικείμενο

// OLED ρυθμίσεις
#define I2C_ADDRESS 0x3C                           // Default I2C διεύθυνση OLED
#define OLED_SDA    4                              // SDA pin I2C για OLED
#define OLED_SCL    15                             // SCL pin I2C για OLED
#define OLED_RST    16                             // Reset pin της OLED
SSD1306Wire display(0x3C, OLED_SDA, OLED_SCL);    // Αντικείμενο οθόνης SSD1306 με
SDA/SCL
void showLogo();                                   // Πρωτότυπο συνάρτησης εμφάνισης
λογότυπου

// LoRa Pins για TTGO LoRa32
#define LORA_SCK   5                               // SPI clock για το LoRa module
(SCK)
#define LORA_MISO  19                              // SPI MISO (LoRa → ESP32)
#define LORA_MOSI  27                              // SPI MOSI (ESP32 → LoRa)
#define SS         18                              // LoRa NSS/CS (chip select)
#define RST        14                              // LoRa reset pin
#define DIO0       26                              // LoRa DIO0/IRQ (packet done)
```

```
void setup() {
  Serial.begin(115200);                          // Εκκίνηση σειριακής για debug

  // Reset ακολουθία OLED: HIGH → LOW → HIGH
  pinMode(OLED_RST, OUTPUT);                     // Ορισμός pin reset της OLED ως
έξοδος
  digitalWrite(OLED_RST, HIGH);                  // Θέσε HIGH
  delay(100);                                    // Καθυστέρηση 100 ms
  digitalWrite(OLED_RST, LOW);                   // Pulse LOW
  delay(100);                                    // Καθυστέρηση 100 ms
  digitalWrite(OLED_RST, HIGH);                  // Πίσω σε HIGH

  Wire.begin(4, 15);                             // Εκκίνηση I2C (SDA=4, SCL=15)

  display.init();                                // Αρχικοποίηση OLED
  display.flipScreenVertically();                // Αναστροφή κάθετη (αν
χρειάζεται)
  showLogo();                                    // Εμφάνιση λογότυπου
  delay(2000);                                   // Μικρή παύση για να φανεί
  display.clear();                               // Καθαρισμός οθόνης
  display.setFont(ArialMT_Plain_10);             // Επιλογή γραμματοσειράς
  display.drawString(0, 0, "TTGO and LoRaRx");   // Τίτλος
  display.display();                             // Ανανέωση οθόνης
  delay(2000);                                   // Μικρή καθυστέρηση

  // Εκκίνηση LoRa
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI);     // Εκκίνηση SPI με custom pins
για το LoRa (SCK, MISO, MOSI)
  LoRa.setPins(SS, RST, DIO0);                   // Δήλωση pins LoRa (CS/RST/DIO0)
  if (!LoRa.begin(868E6)) {                      // Συχνότητα 868 MHz (Ευρώπη)
       Serial.println("LoRa init failed!");      // Αν αποτύχει η αρχικοποίηση
       display.drawString(0, 20, "LoRa Failed!");// Μήνυμα στην OLED
       display.display();                        // Εμφάνιση
       while (1);                                // Σταμάτημα εκτέλεσης
  }
  Serial.println("LoRa Initialized");            // Επιτυχής init LoRa
  display.drawString(0, 20, "LoRa RX Ready!");   // Μήνυμα “δέκτης έτοιμος”
  display.display();                             // Ανανέωση OLED
  delay(2000);                                   // Μικρή καθυστέρηση

  WiFi.begin(ssid, password);                    // Σύνδεση στο Wi-Fi AP

  while (WiFi.status() != WL_CONNECTED) {        // Περιμένε μέχρι να συνδεθεί
      delay(2000);                               // Περίμενε 2s
```

```
      Serial.print(".");                          // Εκτύπωσε τελεία ως πρόοδο
  }
  Serial.println("\nWiFi Connected!");            // Συνδέθηκε
  Serial.print("IP address: ");                   // Εμφάνιση της IP
  Serial.println(WiFi.localIP());                 // Τοπική IP ESP32
  display.clear();                                // Καθαρισμός OLED
  display.setFont(ArialMT_Plain_10);              // Γραμματοσειρά
  display.drawString(20, 40, "\nWiFi Connected!");// Μήνυμα στην οθόνη
  display.display();                              // Ανανέωση OLED
  delay(2000);                                    // Μικρή καθυστέρηση
}

void loop() {
   int packetSize = LoRa.parsePacket();           // Έλεγχος αν ήρθε εισερχόμενο
LoRa πακέτο
   if (packetSize) {                              // Αν υπάρχει packet…
       String receivedData = "";                  // Buffer για το payload
       while (LoRa.available()) {                 // Όσο υπάρχουν bytes στο RX
           receivedData += (char)LoRa.read();     // Διάβασε χαρακτήρα-χαρακτήρα
       }
       Serial.println("Received: " + receivedData); // Εκτύπωση του ληφθέντος

        // Μεταβλητές για parsing περιεχομένου
        String deviceID = "";                     // ID πομπού (TX1/TX2)
        String sensorType = "";                   // Τύπος αισθητήρα (DHT11/LDR)
        float temperature = 0.0;                  // Θερμοκρασία (για TX1)
        float humidity = 0.0;                     // Υγρασία (για TX1)
        String status = "";                       // Κατάσταση (OK/HIGH/LOW/ERROR)
        String timeData;                          // Χρόνος (όπως εστάλη)

        // Έλεγχος για TX1 (DHT11)
        if (receivedData.indexOf("ID:TX1") != -1) {          // Αν περιέχει
“ID:TX1”
          deviceID = "TX1";                                  // Ορισμός ID
          sensorType = "DHT11";                              // Τύπος αισθητήρα

          String tempStr = receivedData.substring(receivedData.indexOf("TEMP:") +
5, receivedData.indexOf(", HUM:"));      // Απόσπαση θερμοκρασίας
          String humStr  = receivedData.substring(receivedData.indexOf("HUM:")  +
4, receivedData.indexOf(", STATUS:"));   // Απόσπαση υγρασίας
          String statusStr =
receivedData.substring(receivedData.indexOf("STATUS:") + 7,
receivedData.indexOf(", TIME:")); // Απόσπαση status
```

```
          String timeDataStr =
receivedData.substring(receivedData.indexOf("TIME:") +
5);                                   // Απόσπαση time έως το τέλος

          temperature = tempStr.toFloat();                    // Μετατροπή σε
float
          humidity    = humStr.toFloat();                     // Μετατροπή σε
float
          status      = statusStr;                            // Αντιγραφή
status
          timeData    = timeDataStr;                          // Αντιγραφή
χρόνου

          Serial.println("From: " + deviceID + " (DHT11)");    // Log πηγής
          Serial.print("Temp: ");     Serial.println(temperature); // Log
θερμοκρασίας
          Serial.print("Humidity: ");  Serial.println(humidity);   // Log
υγρασίας
          Serial.print("Status: ");    Serial.println(status);     // Log
κατάστασης
          Serial.print("TIME: ");      Serial.println(timeData);   // Log χρόνου

          display.clear();                                     // Καθάρισε OLED
          display.drawString(0, 0,  "LoRa RX Data:");          // Τίτλος
          display.drawString(0, 10, "From: " + deviceID + "
(DHT11)");           // Πηγή
          display.drawString(0, 20, "Temp: " + String(temperature) + "
C");      // Θερμοκρασία
          display.drawString(0, 30, "Humidity: " + String(humidity) + "
%");     // Υγρασία
          display.drawString(0, 40, "Status: " +
status);                          // Κατάσταση
          display.drawString(0, 50, "TIME: " +
timeData);                          // Χρόνος
          display.display();                                    // Ανανέωση
οθόνης
      }

       // Έλεγχος για TX2 (LDR)
       else if (receivedData.indexOf("ID:TX2") != -1) {   // Αν περιέχει “ID:TX2”
        deviceID   = "TX2";                                  // Ορισμός ID
        sensorType = "LDR";                            // Τύπος αισθητήρα
```

```
        // Αναμονή μορφής:
        // [ID:TX2, SENSOR:LDR, LDR_VALUE:%d, BRIGHTNESS:%d%%, STATUS:%[^,],
TIME:%[^]]
        int  ldrValue, ldrPercent;                              // Τιμή ADC &
ποσοστό φωτεινότητας
        char statusBuffer[20], timeBuffer[25];                   // Buffers για
status/time
        sscanf(receivedData.c_str(),
               "[ID:TX2, SENSOR:LDR, LDR_VALUE:%d, BRIGHTNESS:%d%%, STATUS:%[^,],
TIME:%[^]]",
               &ldrValue, &ldrPercent,
statusBuffer,  timeBuffer);              // Εξαγωγή πεδίων με sscanf
        String status  = String(statusBuffer);                   // Μετατροπή
status σε String
        String timeStr = String(timeBuffer);                    // Μετατροπή
time σε String

        display.clear();                                         // Καθάρισε OLED
        display.drawString(0, 0,  "LoRa RX Data:");              // Τίτλος
        display.drawString(0, 10, "From: " + deviceID + "
(LDR)");              // Πηγή
        display.drawString(0, 20, "LDR Value: " +
String(ldrValue));            // Τιμή ADC
        display.drawString(0, 30, "LDR Percent: " + String(ldrPercent) + " %");
// Ποσοστό
        display.drawString(0, 40, "Status: " +
status);                          // Κατάσταση
        display.drawString(0, 50, "Time: " +
timeStr);                            // Χρόνος
        display.display();                                       // Ανανέωση OLED
    }

    // Αποστολή ACK στον πομπό
    delay(100);                                 // Μικρή καθυστέρηση για αποφυγή
σύγκρουσης
    LoRa.beginPacket();                         // Έναρξη πακέτου απάντησης
    LoRa.print("ACK");                          // Payload “ACK”
    LoRa.endPacket();                           // Αποστολή
    Serial.println("ACK sent!");                // Log
```

```
    // Προώθηση του ληφθέντος payload μέσω UDP στο PC
    Serial.println("Sending via UDP: " + receivedData); // Log για UDP
    udp.beginPacket(serverIP, serverPort);        // Προετοιμασία UDP packet προς
serverIP:serverPort
    udp.print(receivedData);                      // Γράψε το μήνυμα στο packet
    udp.endPacket();                              // Στείλε το UDP packet
    delay(3000);                                  // Μικρή καθυστέρηση πριν
ξαναδιαβάσεις
   }                                              // Τέλος if(packetSize)
}                                                 // Τέλος loop

void showLogo() {
  uint8_t x_off = (display.getWidth() - logo_width) / 2;  // Κεντράρισμα X του
bitmap
  uint8_t y_off = (display.getHeight() - logo_height) / 2;// Κεντράρισμα Y του
bitmap
  display.clear();                                        // Καθάρισε οθόνη
  display.drawXbm(x_off, y_off, logo_width, logo_height, logo_bits); // Ζωγράφισε
bitmap
  display.display();                                      // Εμφάνισε στην OLED
}
```

Η συνάρτηση **void showLogo()** χρησιμοποιεί το αρχείο images.h για την εμφάνιση του λογότυπου LoRa κατά την εκκίνηση του συστήματος. Το αρχείο **images.h** είναι το ίδιο που αναφέρεται και στο ΠΑΡΑΡΤΗΜΑ Δ.1.

Δ.4. UDP SERVER

Παρακάτω αποτυπώνεται και ο κώδικας python του UDP SERVER με επιπρόσθετες περιγραφές της λειτουργίας του:

```
import socket                          # Βασική βιβλιοθήκη δικτύου (sockets)
import csv                             # Εγγραφή/ανάγνωση CSV αρχείων
import sys                            # Πρόσβαση σε ροές/εξόδους/εξόδους σφαλμάτων
import signal                          # Διαχείριση σημάτων (graceful shutdown)
from pathlib import Path               # Αντικείμενα διαδρομών αρχείων (pathlib)
from datetime import datetime                # Χειρισμός ημερομηνίας/ώρας

# Settings
UDP_IP = "0.0.0.0"               # Διεύθυνση listen (0.0.0.0 = όλες οι διεπαφές)
UDP_PORT = 8888                              # Θύρα UDP για λήψη πακέτων

# Fixed path to save the CSV file (Windows example)
CSV_DIR = Path(r"C:\Users\GOULAS\Desktop\project_html")  # Φάκελος αποθήκευσης
CSV (Windows)
CSV_FILE = CSV_DIR / "sensor_data.csv"     # Πλήρης διαδρομή αρχείου CSV

# CSV header
CSV_HEADER = [                               # Επικεφαλίδες στηλών CSV
    "Timestamp", "Device_ID", "Sensor",
    "Temperature (°C)", "Humidity (%)",
    "LDR_Value", "BRIGHTNESS (%)",
    "Status", "Received From"
]

# Max UDP packet size to read
MAX_UDP_BYTES = 2048                     # Μέγιστο μέγεθος UDP payload σε bytes

# Helpers
def ensure_csv_with_header(csv_path: Path, header: list[str]) -> None:  #
Βεβαιώνει ότι υπάρχει CSV με header
    csv_path.parent.mkdir(parents=True, exist_ok=True)  # Δημιουργεί φακέλους αν
δεν υπάρχουν
# Write header if file doesn't exist or is empty
    if not csv_path.exists() or csv_path.stat().st_size == 0:  # Αν δεν υπάρχει ή
είναι άδειο
        with csv_path.open(mode="w", newline="", encoding="utf-8") as f:  #
Άνοιγμα για εγγραφή
            writer = csv.writer(f)                   # Δημιουργία writer CSV
```

```
            writer.writerow(header)           # Εγγραφή γραμμής επικεφαλίδων

def parse_payload(payload: str) -> dict:          # Κάνει parsing του κειμένου payload σε dict
    """
    Parses payloads like:
    [ID: TX1, SENSOR: DHT11, TEMP: 24.1°C, HUM: 55%, STATUS: OK, TIME: 2025-09-06 10:15:00]
    Returns dict with stripped keys/values. Brackets are optional.
    """                                           # Περιγραφή μορφής εισόδου/εξόδου
    text = payload.strip()                        # Αφαίρεση κενών στην αρχή/τέλος
    if text.startswith("[") and text.endswith("]"): # Αν έχει αγκύλες [...]
        text = text[1:-1]                         # Αφαίρεσε τις αγκύλες

    # Split on commas that separate fields (tolerate spaces)
    parts = [p.strip() for p in text.split(",")]# Χώρισε σε πεδία με βάση το ,
    data = {}                                     # Λεξικό αποτελέσματος
    for part in parts:                            # Για κάθε τμήμα
        if not part:                              # Αν είναι κενό, συνέχισε
            continue
        if ":" not in part:  # Αν δεν περιέχει ':', αγνόησέ το (κακοδιαμορφωμένο)
            # Skip malformed chunk
            continue
        key, val = part.split(":", 1)     # Χώρισε στο πρώτο ':' → (κλειδί, τιμή)
        data[key.strip().upper()] = val.strip()      # Φύλαξε με UPPER κλειδί & trimmed τιμή
    return data                                   # Επιστροφή dict πεδίων

def cleanse_numeric_suffix(value: str, suffix: str) -> str:  # Αφαιρεί κατάληξη π.χ. '%' ή '°C'
    """Strip a trailing suffix like '%' or '°C' if present."""# Περιγραφή
    if not value:                                 # Αν κενό value
        return ""
    v = value.strip()                             # Trim κενών
    if v.endswith(suffix):                        # Αν τελειώνει με το suffix
        v = v[: -len(suffix)]                     # Κόψε το suffix
    return v.strip()                               # Επέστρεψε καθαρό string

def write_row(csv_path: Path, row: list):      # Προσθέτει μια γραμμή στο CSV
```

```
    with csv_path.open(mode="a", newline="", encoding="utf-8") as f:  # Άνοιγμα για προσθήκη
        writer = csv.writer(f)                      # CSV writer
        writer.writerow(row)                        # Γράψε τη γραμμή

def format_addr(addr_tuple) -> str:   # Μορφοποιεί το address tuple σε "ip:port"
    try:
        ip, port = addr_tuple                       # Απόσπαση IP και θύρας
        return f"{ip}:{port}"                       # Επιστροφή μορφής "ip:port"
    except Exception:
        return str(addr_tuple)                      # Fallback σε απλό str()

# Graceful shutdown
_stop = False                               # Global flag για τερματισμό βρόχου
def _handle_signal(signum, frame):          # Handler για σήματα (Ctrl+C, SIGTERM)
    global _stop
    _stop = True                    # Θέτει το flag ώστε να τερματίσει ο βρόχος

signal.signal(signal.SIGINT, _handle_signal)        # Πιάσε SIGINT (Ctrl+C)
if hasattr(signal, "SIGTERM"):                  # Αν υπάρχει SIGTERM σε αυτό το OS
    signal.signal(signal.SIGTERM, _handle_signal)   # Πιάσε SIGTERM

# Main
def main():                                         # Κυρίως σημείο εκτέλεσης
    ensure_csv_with_header(CSV_FILE, CSV_HEADER)    # Βεβαίωσε ότι το CSV υπάρχει με header

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Δημιουργία UDP socket
    try:
        sock.bind((UDP_IP, UDP_PORT))               # Δέσμευση στη διεύθυνση/θύρα
    except OSError as e:                            # Σε αποτυχία bind
        print(f"Failed to bind {UDP_IP}:{UDP_PORT} -> {e}", file=sys.stderr)  # Αναφορά σφάλματος
        sys.exit(1)                                 # Έξοδος με κωδικό σφάλματος

    print(f"Listening for UDP packets on {UDP_IP}:{UDP_PORT}")  # Πληροφορία εκκίνησης

    try:
        while not _stop:                # Κύριος βρόχος μέχρι να δοθεί σήμα stop
            try:
                sock.settimeout(1.0)    # Timeout 1s για να ελέγχει συχνά το _stop
```

```
                data, addr = sock.recvfrom(MAX_UDP_BYTES)  # Λήψη δεδομένων
(bytes) + διεύθυνση αποστολέα
            except socket.timeout:                  # Αν λήξει ο χρόνος αναμονής
                continue                    # Συνέχισε τον βρόχο (ξαναδοκίμασε)
            except Exception as e:                  # Άλλα σφάλματα socket
                print(f"Socket error: {e}", file=sys.stderr)  # Αναφορά σφάλματος
                continue                      # Παράκαμψη πακέτου / συνέχιση

            sender = format_addr(addr)      # Μορφοποιημένη διεύθυνση αποστολέα

            try:
                decoded = data.decode("utf-8", errors="replace").strip()  #
Αποκωδικοποίηση bytes σε str
            except Exception as e:         # Σε περίπτωση αποτυχίας decode
                print(f"Decode error from {sender}: {e}", file=sys.stderr)  #
Αναφορά σφάλματος
                continue                            # Παράκαμψη πακέτου

            if not decoded:                         # Αν κενό μήνυμα
                print(f"Ignored empty packet from {sender}")   # Ενημέρωση
                continue                            # Παράκαμψη

            print(f"Received: {decoded} from {sender}")   # Log ληφθέντος payload

            try:
                parsed = parse_payload(decoded)        # Parsing σε dict
(ID/SENSOR/TEMP/HUM/…)
                                        # Επιστρέφει keys σε UPPER case

                device_id = parsed.get("ID", "Unknown")       # Πάρε ID συσκευής
                sensor = parsed.get("SENSOR", "Unknown")    # Πάρε τύπο αισθητήρα
                status = parsed.get("STATUS", "Unknown")    # Πάρε κατάσταση
                received_time = parsed.get("TIME", "") or
datetime.now().strftime("%Y-%m-%d %H:%M:%S")  # Χρόνος (από payload ή τώρα)

                # Defaults
                temperature = ""                    # Προεπιλογή θερμοκρασίας
                humidity = ""                       # Προεπιλογή υγρασίας (κενό)
                ldr_value = ""                      # Προεπιλογή LDR value (κενό)
                ldr_percent = ""                    # Προεπιλογή BRIGHTNESS %
                # Populate depending on device
                dev_upper = device_id.upper() if isinstance(device_id, str) else
str(device_id)  # Κανονικοποίηση ID σε UPPER
```

```
                if dev_upper == "TX1":              # Αν πομπός TX1 (DHT11)
                    temperature = cleanse_numeric_suffix(parsed.get("TEMP", ""),
"°C")  # Αφαίρεσε '°C'
                    humidity = cleanse_numeric_suffix(parsed.get("HUM", ""),
"%")       # Αφαίρεσε '%'
                elif dev_upper == "TX2":            # Αν πομπός TX2 (LDR)
                    ldr_value = parsed.get("LDR_VALUE",
"").strip()                     # Τιμή LDR raw
                    ldr_percent = cleanse_numeric_suffix(parsed.get("BRIGHTNESS",
""), "%")  # Αφαίρεσε '%'
                else:     # Άγνωστη συσκευή: δοκίμασε να διαβάσεις ό,τι υπάρχει
                    temperature = cleanse_numeric_suffix(parsed.get("TEMP", ""),
"°C")  # Θερμοκρασία (αν υπάρχει)
                    humidity = cleanse_numeric_suffix(parsed.get("HUM", ""),
"%")       # Υγρασία (αν υπάρχει)
                    ldr_value = parsed.get("LDR_VALUE",
"").strip()                     # LDR value (αν υπάρχει)
                    ldr_percent = cleanse_numeric_suffix(parsed.get("BRIGHTNESS",
""), "%")  # BRIGHTNESS (αν υπάρχει)

                write_row(                          # Γράψε μια γραμμή στο CSV
                    CSV_FILE,
                    [
                        received_time, device_id, sensor,  # Χρόνος/ID/Αισθητήρας
                        temperature, humidity,              # Θερμ./Υγρασία
                        ldr_value, ldr_percent,            # LDR/Φωτεινότητα
                        status, sender                     # Κατάσταση/Αποστολέας
                    ],
                )

                print("Data saved to CSV")          # Επιβεβαίωση αποθήκευσης

            except Exception as e:          # Σφάλμα κατά το parsing/αποθήκευση
                print(f"Error parsing/saving data from {sender}: {e}",
file=sys.stderr)  # Αναφορά σφάλματος

    finally:
        sock.close()                    # Κλείσιμο socket κατά τον τερματισμό
        print("\nServer stopped.")                  # Μήνυμα τερματισμού

if __name__ == "__main__":              # Εκτέλεση όταν τρέχει ως κύριο script
    main()                                          # Κλήση main()
```

Δ.5. USER INTERFACE

Παρακάτω αποτυπώνεται ο κώδικας (αρχείο index.html) για τη δημιουργία της σελίδας SENSOR DATA VIEWER.

```
<!DOCTYPE html> <!-- Δήλωση HTML5 για τυποποιημένη, σύγχρονη απόδοση από τον browser -->
<html lang="en"> <!-- Γλώσσα εγγράφου: Αγγλικά (χρήσιμο για προσβασιμότητα/SEO) -->
<head> <!-- Έναρξη κεφαλίδας: μεταδεδομένα, τίτλος, συνδέσεις CSS/JS που φορτώνονται στο <head> -->
  <meta charset="UTF-8" /> <!-- Κωδικοποίηση χαρακτήρων UTF-8 (σωστή εμφάνιση ελληνικών/ειδικών χαρακτήρων) -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/> <!-- Responsive ρυθμίσεις για κινητές συσκευές -->
  <title>Sensor Data Viewer - UOI - APOSTOLOS GOULAS</title> <!-- Τίτλος καρτέλας/σελίδας του browser -->

  <!-- Σύνδεση με το φύλλο στυλ -->
  <link rel="stylesheet" href="style.css"> <!-- Εισαγωγή του CSS που μορφοποιεί το UI (τυπογραφία, πίνακας, διαγράμματα) -->
</head>
<body> <!-- Έναρξη σώματος σελίδας: ό,τι είναι ορατό στον χρήστη -->

<h1>Sensor Data Viewer - UOI - APOSTOLOS GOULAS</h1> <!-- Κύριος τίτλος που εμφανίζεται στην κορυφή της σελίδας -->

<div class="controls"> <!-- Μπάρα ελέγχου με κουμπιά/επιλογές για φόρτωση/αναφόρτωση/auto-refresh -->
  <button id="btnLoad" onclick="loadCSV()">Load sensor_data.csv</button> <!-- Κουμπί που καλεί τη loadCSV() για να φορτώσει το CSV -->
  <button id="btnReload" onclick="reloadNow()">Reload</button> <!-- Κουμπί για άμεση αναφόρτωση δεδομένων (χωρίς alert) -->
  <label style="margin-left:12px;"> <!-- Ετικέτα που περιέχει checkbox για αυτόματη ανανέωση -->
    <input type="checkbox" id="autoRefresh" onchange="toggleAutoRefresh(this.checked)"> <!-- Ενεργοποίηση/απενεργοποίηση auto-refresh -->
    Auto refresh (10s) <!-- Κείμενο περιγραφής του checkbox -->
  </label>
  <span id="status" style="margin-left:12px; font-size:0.95em; opacity:.8;"></span> <!-- Περιοχή εμφάνισης κατάστασης (π.χ. Last update) -->
```

```
</div>

<div id="outputTable"></div> <!-- Placeholder container: εδώ εγχέεται δυναμικά ο HTML πίνακας με τα δεδομένα CSV -->

<div id="chartContainer"> <!-- Περιέκτης για τα 4 διαγράμματα (Chart.js canvases) -->
  <canvas id="chart1" width="800" height="400"></canvas> <!-- Καμβάς διαγράμματος #1 (π.χ. Temperature) -->
  <canvas id="chart2" width="800" height="400"></canvas> <!-- Καμβάς διαγράμματος #2 (π.χ. Humidity) -->
  <canvas id="chart3" width="800" height="400"></canvas> <!-- Καμβάς διαγράμματος #3 (π.χ. LDR Value) -->
  <canvas id="chart4" width="800" height="400"></canvas> <!-- Καμβάς διαγράμματος #4 (π.χ. Brightness) -->
</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <!-- Εισαγωγή βιβλιοθήκης Chart.js από CDN για σχεδίαση γραφημάτων -->
<script src="script.js"></script> <!-- Εισαγωγή του κύριου αρχείου JavaScript με τη λογική: loadCSV/parse/display/drawCharts -->
<!-- (Προσοχή να ταιριάζει το όνομα: αν το αρχείο σου είναι script1.js ή script3.js, άλλαξε αντίστοιχα εδώ) -->

</body> <!-- Λήξη σώματος σελίδας -->
</html> <!-- Λήξη εγγράφου HTML -->
```

Η λειτουργικότητα της σελίδας παρέχεται μέσω του αρχείου JavaScript, το οποίο καλείται μέσα στον κώδικα html.
Παρακάτω αποτυπώνεται ο κώδικας του αρχείου **script.js:**

```
// === Ρυθμίσεις auto-refresh ===
const REFRESH_MS = 10_000;  // 10s                          // Περίοδος αυτόματης ανανέωσης σε ms
let refreshTimer = null;    // setInterval handle           // Χειριστής για το setInterval (ώστε να τον καθαρίζουμε)
let charts = [];            // Chart instances για destroy() πριν από redraw  // Αποθήκη ενεργών γραφημάτων Chart.js

// Βοηθητικά
const statusEl = () => document.getElementById('status');   // Συντόμευση: στοιχείο εμφάνισης κατάστασης
```

```
function setStatus(txt) { const el = statusEl(); if (el) el.textContent = txt || ''; } // Ενημέρωση/εκκαθάριση status κειμένου

function startAutoRefresh(periodMs = REFRESH_MS) {          // Εκκινεί auto-refresh με συγκεκριμένο διάστημα
  stopAutoRefresh();                                        // Ασφαλές: σταματά τυχόν προηγούμενο interval
  refreshTimer = setInterval(() => loadCSV().catch(console.error), periodMs); // Κάθε periodMs → νέα φόρτωση CSV
  setStatus(`Auto refresh: every ${periodMs/1000}s`);       // Ενημέρωση status (π.χ. “κάθε 10s”)
}
function stopAutoRefresh() {                                // Διακοπή του auto-refresh
  if (refreshTimer) { clearInterval(refreshTimer); refreshTimer = null; } // Καθαρισμός interval
  setStatus('Auto refresh: off');                           // Ενημέρωση status
}

// --- Συνδεδεμένα με το HTML ---
function reloadNow() {                                      // Κουμπί “Reload”: άμεση ανανέωση
  loadCSV().catch(console.error);            // στιγμιαίο reload // Καλεί την loadCSV χωρίς αλλαγή στο auto-refresh
}
function toggleAutoRefresh(on) {                            // Checkbox auto-refresh: on/off
  if (on) startAutoRefresh(REFRESH_MS); else stopAutoRefresh(); // Εναλλαγή κατάστασης ανάλογα με την τιμή
}

// Εκκίνηση: 1η φόρτωση (κρατώ το auto-refresh όπως το έχεις ήδη ή άφησε μόνο την πρώτη γραμμή)
window.addEventListener('DOMContentLoaded', () => {         // Όταν φορτωθεί το DOM...
  loadCSV().catch(console.error);            // 1η φόρτωση    // Φόρτωση δεδομένων αμέσως
  // Προαιρετικά: αν θες auto-start, άφησε την επόμενη γραμμή, αλλιώς σβήστην.
  // startAutoRefresh(REFRESH_MS);                            // Αυτόματο ξεκίνημα auto-refresh
});

// --- ΚΥΡΙΑ ΡΟΗ ---
```

```
async function loadCSV() {                                    // Κύρια συνάρτηση φόρτωσης του CSV
  try {
    const resp = await fetch('sensor_data.csv', { cache: 'no-cache' }); // Λήψη CSV (χωρίς cache)
    const csv = await resp.text();                            // Μετατροπή απάντησης σε απλό κείμενο
    const data = parseCSV(csv);                               // Parsing σε δισδιάστατο πίνακα [γραμμές][στήλες]
    displayTable(data);                                       // Απόδοση HTML πίνακα στο #outputTable
    drawCharts(data);                                         // Δημιουργία/αναδημιουργία 4 διαγραμμάτων
    setStatus(`Last update: ${new Date().toLocaleString()}`); // Ενημέρωση χρόνου τελευταίας ανανέωσης
  } catch (error) {
    alert("Error loading sensor_data.csv. Make sure it's in the same folder and you're using a local server."); // Μήνυμα λάθους
    console.error(error);                                     // Καταγραφή σφάλματος στην κονσόλα
    setStatus('⚠ Error loading CSV');                         // Ενημέρωση status για αποτυχία
  }
}

function parseCSV(text) {    // Απλός parser CSV (χωρίζει με newline και κόμμα)
  const lines = text.trim().split('\n');             // Σπάει σε γραμμές, αφαιρώντας κενά άκρα
  return lines.map(line => line.split(','));              // Για κάθε γραμμή, σπάει τα κελιά με κόμμα
}

function displayTable(data) {     // Απόδοση δεδομένων σε HTML πίνακα
  const output = document.getElementById('outputTable');    // Container εξόδου πίνακα
  let html = '<table><thead><tr>';                // Έναρξη πίνακα + κεφαλίδων
  data[0].forEach(header => html += `<th>${header}</th>`);  // Δημιουργία κεφαλίδων από την πρώτη γραμμή
  html += '</tr></thead><tbody>';     // Κλείσιμο κεφαλίδων, έναρξη σώματος
  data.slice(1).forEach(row => {      // Για κάθε υπόλοιπη γραμμή δεδομένων...
    html += '<tr>' + row.map(cell => `<td>${cell}</td>`).join('') + '</tr>'; // Δημιουργία κελιών
  });
  html += '</tbody></table>';                                 // Κλείσιμο πίνακα
```

```
  output.innerHTML = html;                                    // Εγχυση του πίνακα στο DOM
}

function drawCharts(data) {                                   // Δημιουργία τεσσάρων line charts με Chart.js
  // καθάρισμα παλιών charts
  charts.forEach(ch => ch.destroy());                         // Αν υπάρχουν προηγούμενα διαγράμματα, κατέστρεψέ τα
  charts = [];                                                // Εκκαθάριση λίστας

  const headers = data[0];                                    // Επικεφαλίδες (πρώτη γραμμή CSV)
  const timeIndex       = headers.findIndex(h => h.toLowerCase().includes("timestamp"));   // Θέση στήλης χρόνου
  const timeIndex       = headers.findIndex(h => h.toLowerCase().includes("temperature")); // Θέση στήλης θερμοκρασίας
  const humIndex        = headers.findIndex(h => h.toLowerCase().includes("humidity"));    // Θέση στήλης υγρασίας
  const ldrIndex        = headers.findIndex(h => h.toLowerCase().includes("ldr"));         // Θέση στήλης LDR value
  const brightnessIndex = headers.findIndex(h => h.toLowerCase().includes("brightness"));  // Θέση στήλης φωτεινότητας

  const labels = [];                                          // Ετικέτες άξονα X (timestamps)
  const tempData = [], humData = [], ldrData = [], brightData = []; // Σειρές δεδομένων για Y

  for (let i = 1; i < data.length; i++) {                     // Διάσχιση όλων των γραμμών δεδομένων (εκτός headers)
    labels.push(data[i][timeIndex]);                          // Προσθήκη χρονικής σφραγίδας
    tempData.push(parseFloat(data[i][tempIndex]));            // Μετατροπή τιμής σε αριθμό (Temperature)
    humData.push(parseFloat(data[i][humIndex]));              // (Humidity)
    ldrData.push(parseFloat(data[i][ldrIndex]));              // (LDR)
    brightData.push(parseFloat(data[i][brightnessIndex]));    // (Brightness)
  }

  const chartConfigs = [                                      // Περιγραφές για 4 διαγράμματα
    { id: 'chart1', label: headers[tempIndex],       data: tempData,   color: 'red',    title: 'Temperature Over Time' },
```

```
    { id: 'chart2', label: headers[humIndex],        data: humData,    color:
'blue',   title: 'Humidity Over Time' },
    { id: 'chart3', label: headers[ldrIndex],        data: ldrData,    color:
'green',  title: 'LDR Value Over Time' },
    { id: 'chart4', label: headers[brightnessIndex], data: brightData, color:
'orange', title: 'Brightness Over Time' }
  ];

  chartConfigs.forEach(config => {                // Για κάθε διάγραμμα...
    const ctx = document.getElementById(config.id).getContext('2d'); // Απόκτηση
2D context από τον αντίστοιχο καμβά
    const chart = new Chart(ctx, {     // Δημιουργία νέου Chart.js αντικειμένου
      type: 'line',                      // Τύπος γραφήματος: γραμμή
      data: {
        labels,                                      // Χρονικές ετικέτες (X)
        datasets: [{
          label: config.label,                // Ετικέτα dataset (από headers)
          data: config.data,                             // Τιμές Y
          borderColor: config.color,                     // Χρώμα γραμμής
          fill: false,                       // Χωρίς γέμισμα κάτω από τη γραμμή
          tension: 0.2,                            // Ελαφριά καμπύλωση γραμμής
          spanGaps: true  // Συνέχεια γραμμής πάνω από NaN κενά (χωρίς "σπάσιμο")
        }]
      },
      options: {
        responsive: true,            // Προσαρμογή στο μέγεθος container
        // maintainAspectRatio: false, // ενεργοποίησέ το αν μικρύνεις τα charts
μέσω CSS containers
        plugins: { title: { display: true, text: config.title } }, // Τίτλος
διαγράμματος
        scales: {
          x: { title: { display: true, text: 'Timestamp' } },   // Τίτλος άξονα X
          y: { title: { display: true, text: 'Value' } }       // Τίτλος άξονα Y
        }
      }
    });
    charts.push(chart);  // Αποθήκευση αναφοράς για μελλοντικό destroy()
  });
}
```

Η αισθητική και εργονομία της διεπαφής ρυθμίζεται από το **style.css**, του οποίου ο κώδικας αποτυπώνεται παρακάτω:

```
body {                                  /* Βασικό container όλης της σελίδας */
    font-family: Arial, sans-serif;     /* Ορισμός βασικής γραμματοσειράς για
ευανάγνωστο κείμενο */
    padding: 20px;                      /* Εσωτερικό περιθώριο ώστε το
περιεχόμενο να μην ακουμπά τα άκρα */
  }
button {                               /* Στυλ για όλα τα κουμπιά της σελίδας */
    padding: 10px 15px;                 /* Μεγαλύτερη «επιφάνεια» κλικ και
καλύτερη εργονομία */
    margin-bottom: 20px;                /* Κάθετο κενό κάτω από το κουμπί ώστε
να μην κολλάει στα επόμενα στοιχεία */
  }

table {                                 /* Γενικός κανόνας για τον HTML πίνακα
δεδομένων */
    width: 100%;                        /* Χρήση όλου του διαθέσιμου πλάτους του
container */
    border-collapse: collapse;          /* Ενιαία γραμμή περιγράμματος (χωρίς
διπλά borders στα κελιά) */
    margin-top: 20px;                   /* Απόσταση από τα στοιχεία που
προηγούνται (π.χ. κουμπιά/τίτλοι) */
  }

th, td {                              /* Κοινό στυλ για κεφαλίδες (th) και κελιά
(td) */
    padding: 8px;                       /* Εσωτερικό κενό για καλύτερη
αναγνωσιμότητα του περιεχομένου */
    border: 1px solid #ddd;             /* Απαλό περίγραμμα που οριοθετεί τα
κελιά */
    text-align: left;                   /* Στοίχιση κειμένου αριστερά (τυπικό
για πίνακες δεδομένων) */
  }

#chartContainer {                       /* Περιέκτης για τους καμβάδες των
διαγραμμάτων (Chart.js) */
    width: 100%;                      /* Καταλαμβάνει όλο το διαθέσιμο πλάτος */
    max-width: 900px;                   /* Αποφεύγει υπερβολικά πλατιά
διαγράμματα σε μεγάλες οθόνες */
    margin-top: 40px;                   /* Απόσταση από τον πίνακα/κουμπιά για
καθαρό διαχωρισμό */
  }
```

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΤΙΤΛΟΣ: ΑΝΑΠΤΥΞΗ ΑΣΥΡΜΑΤΟΥ ΚΑΤΑΝΕΜΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΥΤΟΜΑΤΙΣΜΟΥ IoT ΜΕ ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΠΜΣ-ΣΗΤ	ΓΟΥΛΑΣ ΑΠΟΣΤΟΛΟΣ Τμήμα Φυσικής	Πανεπιστήμιο Ιωαννίνων