ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση web εφαρμογής για την παρουσίαση δεδομένων καιρού από αισθητήρες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Α. Ιωαννίδης

Επιβλέπων Καθηγητής:

Ευάγγελος Β. Χριστοφόρου

Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση web εφαρμογής για την παρουσίαση δεδομένων καιρού από αισθητήρες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Ιωαννίδης

Επιβλέπων Καθηγητής:
Ευάγγελος Β. Χριστοφόρου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Φεβρουαρίου 2025.

Ευάγγελος Χριστοφόρου	Ιωάννης Γκόνος	Εμμανουήλ Χουρδάκης
Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής
Ε.Μ.Π.	Ε.Μ.Π.	Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2025

.....

.....

Αλέξανδρος Α. Ιωαννίδης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα διπλωματική εργασία αφορά στις δυνατότητες της τεχνολογίας IoT και προτείνει το σύστημα DashPL. Πρόκειται για ένα εργαλείο παρακολούθησης και παρουσίασης δεδομένων σε χρονοσειρές. Στόχος είναι η χρήση και αξιοποίησή του από οποιονδήποτε θέλει να εμφανίσει στην οθόνη του μία μέτρηση σε τρέχοντα χρόνο από ασύρματα συνδεδεμένο αισθητήρα.

Στο εισαγωγικό κεφάλαιο παρουσιάζεται το πλαίσιο αναφοράς που είναι η τεχνολογία IoT. Σε αυτήν καίριο ρόλο διαδραματίζει η τεχνολογία των αισθητήρων που συνεπικουρούμενη και από άλλα δομικά στοιχεία υποστηρίζει μια διάταξη IoT τεχνολογίας.

Στο δεύτερο κεφάλαιο αναλύεται η δομή του IoT με ιδιαίτερη αναφορά στο επίπεδο δικτύου. Δίνονται τα κύρια χαρακτηριστικά μιας περίπτωσης χρήσης της συγκεκριμένης τεχνολογίας μέσω ενός παραδείγματος που αφορά στην κάλυψη μιας καλλιέργειας από αυτό το δίκτυο. Τέλος, περιγράφονται οι κύριες τεχνολογίες για τη δικτύωση αντίστοιχων διατάξεων.

Στο τρίτο κεφάλαιο παρουσιάζονται διατάξεις IoT που εφαρμόζονται σήμερα στη γεωργία για τη καλύτερη διαχείριση των πόρων και την αύξηση της ποιότητας της παραγωγής.

Στο τέταρτο κεφάλαιο περιγράφεται το σύστημα DashPL μαζί με το λογισμικό του και τις προδιαγραφές λειτουργίας του. Τέλος αναλύονται οι σχεδιαστικές επιλογές που ακολουθήθηκαν για την ανάπτυξη του συγκεκριμένου λογισμικού.

Λέξεις-κλειδιά: *διαδικτυακή εφαρμογή, JavaScript, frontend, μικροελεγκτής, παρακολούθηση-επιτήρηση, οικολογία, διαδίκτυο των πραγμάτων*

Abstract

This diploma dissertation explores the potential of IoT technology and introduces the DashPL system, a tool for monitoring and presenting data in time series. DashPL is designed for users who want to display real-time measurements on their screen from a wirelessly connected sensor.

The introductory chapter presents the IoT framework, with a key focus on sensor technology, which plays a crucial role in supporting IoT systems.

The second chapter analyzes the structure of the IoT, with particular emphasis on the network layer. It describes the main characteristics of IoT networks through a case study on crop monitoring. Additionally, it outlines the primary technologies used for networking such systems.

The third chapter examines IoT devices currently used in agriculture to improve resource management and enhance production quality.

The fourth chapter provides a detailed description of the DashPL system, including its software architecture and operational specifications. It also explains the design choices made during the development of this software.

Keywords: *web application, JavaScript, frontend, microcontroller, monitoring, ecology, Internet of Things*

Ευχαριστίες

Οφείλω ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή κ. Βαγγέλη Χριστοφόρου, ο οποίος με βοήθησε με τις πολύτιμες συμβουλές του και με ενίσχυσε με την εμπιστοσύνη που μου έδειξε.

Επιπλέον, θα ήθελα να ευχαριστήσω τον κ. Σπυρίδωνα Αγγελόπουλο που με τις σημαντικές υποδείξεις του συνέβαλε στον σχεδιασμό της εφαρμογής DashPL.

Πίνακας περιεχομένων

Ευρετήριο Εικόνων

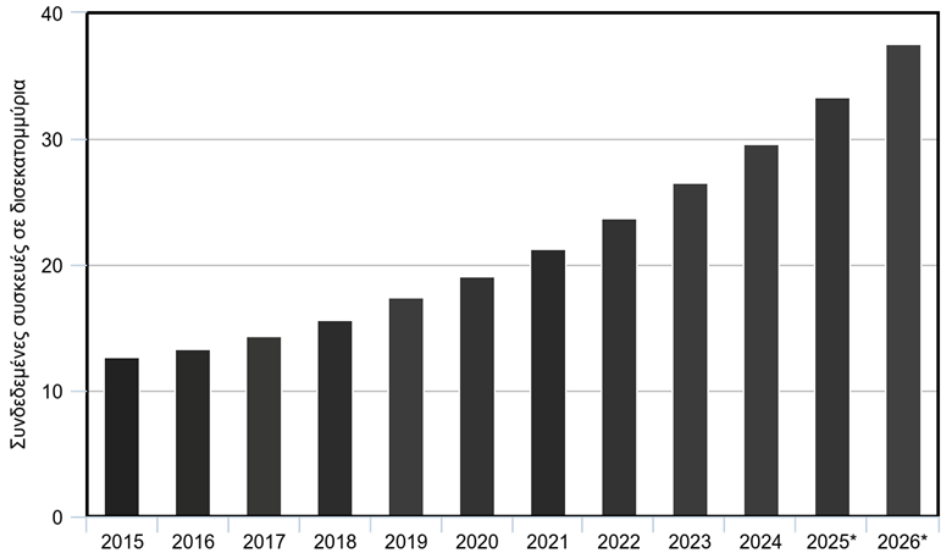
1 Κεφάλαιο

1.1 Εισαγωγή

1.1.1 Διαδίκτυο-των-Πραγμάτων

Ο όρος διαδίκτυο των πραγμάτων (αγγλικά: Internet of Things – IoT) αποτελεί μια τεχνολογική επανάσταση. Αναφέρεται σε συσκευές με πρόσθετες τεχνολογίες, οι οποίες συνδέονται με άλλες συσκευές και συστήματα συλλέγοντας και ανταλλάσσοντας δεδομένα μέσω του διαδικτύου.

Στο διαδίκτυο των πραγμάτων οι συσκευές εγκαθίστανται σε πολλά περιβάλλοντα, όπως για παράδειγμα σε νοσοκομεία, κατοικίες, εργοστάσια, αλλά και σε αυτοκίνητα, δρόμους, αεροπλάνα. Τα νούμερα που συνδέονται με τον όγκο των δεδομένων στον ψηφιακό χώρο αγγίζουν τα 100 zettabytes, ή ισοδύναμα με 100 δισεκατομμύρια terabytes[1].



Εικόνα 1: Εκτιμώμενος αριθμός IoT συσκευών[2]

[1]https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/
[2]https://www.statista.com/statistics/512650/worldwide-connected-devices-amount/

Ο όρος “Internet of Things” επινοήθηκε τη δεκαετία του 90’. Πλέον έχει γίνει δόκιμος. Παρακάτω αναπτύσσονται οι τεχνολογικοί όροι διαδίκτυο και “πράγμα”.

Ιστορικά, το διαδίκτυο βασίζεται στο όραμα του ARPANET. Ένα ερευνητικό πρόγραμμα που στόχο είχε να συνδέσει διάφορους ερευνητικούς οργανισμούς και πανεπιστήμια, επιτρέποντας την ανταλλαγή πληροφοριών και προάγοντας τη συνεργασία. Το πρόγραμμα που πήρε τη σκυτάλη είναι το NSFNET. Αυτό αποτέλεσε τον πυρήνα του σύγχρονου διαδικτύου. Ο όρος δίκτυο ξεκίνησε να μελετάται, σύμφωνα με τον καθηγητή Χρήστο Παπαδημητρίου το 1993. Το διαδίκτυο ωστόσο αν και ανταποκρίνεται στο όραμα της “άμεσης” δημοκρατίας και της απρόσκοπτης επικοινωνίας έχει και τη “σκοτεινή” πλευρά. Η εκμετάλλευση και η κεφαλαιοποίηση της πληροφορίας, η διάδοση του “μίσους”, ο εθισμός και ο κοινωνικός φορμαλισμός είναι μόνο μερικές από τις αρνητικές πτυχές του.

Το ‘The Thing’ επινοήθηκε και υλοποιήθηκε από τον Léon Theremin. Πρόκειται για μια παθητική συσκευή ακρόασης του 1945 που χρησιμοποιήθηκε για να καταγράφει συνομιλίες μυστικά. Είχε τη δυνατότητα να μπορεί και να ανιχνεύει ηχητικά κύματα από απόσταση, έχοντας μικρόφωνο και ραδιοφωνικό δέκτη. Αποτελεί τον σημαντικό προάγγελο της τεχνολογίας που εισήγαγε ο Charles Walton με την ονομασία radio frequency identification ή εν συντομία RFID [1]. Η συσκευή του Theremin, αν και δεν ταυτίζεται με τα RFID, αξιοποιούσε την ικανότητα των ηλεκτρομαγνητικών κυμάτων να μεταφέρουν δεδομένα, θέτοντας τις βάσεις για τα σύγχρονα συστήματα ασύρματης επικοινωνίας.

Έτσι, συνδυάστηκαν οι δύο παραπάνω όροι και προέκυψε ο νέος τομέας τεχνολογικής δραστηριότητας το διαδίκτυο των πραγμάτων (IoT).

Η τεχνολογία IoT βασίζεται σε πολλά θεμελιώδη στοιχεία, όπως είναι οι αισθητήρες, οι μικροελεγκτές, το δίκτυο, τα πρωτόκολλα επικοινωνίας και οι πλατφόρμες νέφους για την κεντρική επεξεργασία των δεδομένων. Οι αισθητήρες παίζουν καθοριστικό ρόλο στη λήψη δεδομένων σε πραγματικό χρόνο, όπως της θερμοκρασίας και της υγρασίας. Επιπλέον συνδυάζονται για επεξεργασία της πληροφορίας με τους μικροελεγκτές. Εκείνοι διαβιβάζουν τα

δεδομένα σε τοπικούς πίνακες ελέγχου για οπτικοποίηση είτε τα μεταδίδουν σε διακομιστές.

Η επικοινωνία μεταξύ τους είναι κεντρική και χρησιμοποιείται κυρίως η ασύρματη μετάδοση μέσω τεχνολογίας Wi-Fi ή Bluetooth ή ZigBee ή κα.

Οι λόγοι για τους οποίους το IoT έχει ισχυροποιηθεί και γιγαντωθεί σήμερα, οφείλεται κυρίως στη πρόοδο της τεχνολογίας των αισθητήρων και των μικροελεγκτών. Οι συσκευές του IoT είναι εξοπλισμένες με αισθητήρες, μικροελεγκτές με υπολογιστική ισχύ, λογισμικό και άλλες τεχνολογίες. Καθένα από αυτά έχει γίνει ενεργειακά αποδοτικότερο και σημαντικά οικονομικότερο λαμβάνοντας ολοένα και μεγαλύτερη αποδοχή τόσο από εταιρίες όσο και από άτομα που ενδιαφέρονται. Επιπλέον, η αύξηση της ζήτησης για αυτοματοποίηση των διαδικασιών και του απομακρυσμένου ελέγχου έχει κάνει το IoT ανάρπαστο. Σε αυτά έρχεται να συνδράμει η ευρεία διαθεσιμότητα του διαδικτύου, η πρόσβαση σε αυτό και η αποδοτική ανταλλαγή δεδομένων σε υψηλές ταχύτητες. Τέλος η ανάπτυξη της υπολογιστικής νέφους (Cloud Computing) και η εξέλιξη του ερευνητικού τομέα των Data Analytics, επιτρέπει την αποθήκευση, τη συσχέτιση και την ανάλυση μεγάλου όγκου δεδομένων (Big Data).

1.1.2 Περί αισθητήρων

Ο σκοπός ενός αισθητήρα είναι η μετατροπή μιας φυσικής διέγερσης σε μετρήσιμο ηλεκτρικό σήμα. Σύγχρονες συσκευές, όπως τα κινητά, που ενσωματώνουν τους αισθητήρες χαρακτηρίζονται έξυπνα, γιατί συνάπτουν στα ερεθίσματα από τα διάφορα εξωτερικά φαινόμενα αλγορίθμους που τρέχουν σε υλικό (hardware) [2]. Οι αισθητήρες βρίσκουν εφαρμογή στη βιομηχανία, στην ιατρική, στον στρατό, στην αγροκαλλιέργεια, στην παρακολούθηση του περιβάλλοντος.

Τα κύρια χαρακτηριστικά ενός αισθητήρα είναι:

- Η Ευαισθησία (sensitivity)

Η ευαισθησία αποδίδεται με μια τιμή. Εκφράζεται ως κλάσμα και αφορά τη λεπτότητα της αντίληψης που λαμβάνουμε από τον αισθητήρα για ένα

παρατηρούμενο φυσικό μέγεθος. Ορίζεται ως ρυθμός μεταβολής και είναι ένας λόγος (π.χ. 1 mV/°C). Αφορά και το παρατηρούμενο μέγεθος και το ποσό που αντιλαμβάνεται ο χρήστης του αισθητήρα για μικρομεσαία μεταβολή του φυσικού μεγέθους αυτού.

- Η Διακριτική Ικανότητα (resolution)

Ως διακριτική ικανότητα ενός αισθητήρα ορίζεται η ελάχιστη διακύμανση του σήματος εισόδου που μπορεί να εντοπιστεί από αυτόν (π.χ. 0.05 °C)

- Η Ακρίβεια (accuracy)

Για την Ακρίβεια δίνετε μια κύμανση σχετικά με τις τιμές του παρατηρούμενου φυσικού μεγέθους και τις τιμές που αντιλαμβάνεται ο χρήστης μέσω του αισθητήρα. Ορίζεται ως η μέγιστη απόκλιση μεταξύ μετρούμενης καταγεγραμμένης τιμής από την φυσική πραγματική τιμή (π.χ. ±0.1°C).

- Το Εύρος μέτρησης (span or dynamic range)

Το εύρος (π.χ. από -50°C έως +120°C) στο οποίο μπορεί να αξιοποιηθεί ο αισθητήρας για τις μετρούμενες τιμές του παρατηρούμενου φυσικού μεγέθους. Σήματα εκτός του εύρους προκαλούν σφάλματα ακρίβειας όσο και πιθανή καταστροφή του αισθητήρα.

- Η Γραμμικότητα (linearity)

Η γραμμικότητα περιγράφει πόσο γραμμικά η έξοδος του αισθητήρα αντιστοιχεί στην πραγματική μετρούμενη τιμή σε όλο το εύρος του. Ένας απόλυτα γραμμικός αισθητήρας θα έχει μια ευθύγραμμη σχέση μεταξύ εισόδου και εξόδου.

- Το Σφάλμα Υστέρησης (hysteresis error)

Το σφάλμα που προκύπτει από την εξάρτηση της εξόδου του αισθητήρα, χωρίς reset της κατάστασης ενδιάμεσα, με τη προηγούμενη κατάσταση του σήματος εξόδου.

- Χρόνος απόκρισης (response time)

Η χρονική διάρκεια που απαντάται στην ένδειξη που αντιλαμβάνεται ο χρήστης για μια μεταβολή του τρέχοντος φυσικού μεγέθους (π.χ. 2ms).

Τα παραπάνω χαρακτηριστικά καθιστούν τους αισθητήρες ένα ικανό “εργαλείο” για την πρόσληψη και απόδοση μετρήσεων των φυσικών φαινομένων. Η απόδοση γίνεται με λεπτότητα και ακρίβεια. Η φυσική αυτή διαδικασία δίνει την ικανότητα στο παρατηρητή να μετρά και να μελετά τα φυσικά φαινόμενα.

Οι μετρήσεις αυτών των ποσοτήτων περιγράφονται άλλοτε με έναν αριθμό και άλλοτε με την αναγραφή και της κατεύθυνσης. Στην πρώτη περίπτωση ονομάζονται βαθμωτά [3], ενώ στη δεύτερη διανυσματικά μεγέθη. Για τη πρώτη περίπτωση, τις απλουστευμένες περιγραφές εάν γίνουν αντικείμενο παρατηρήσεως μπορούν να οδηγήσουν σε οπτικοποίηση των δεδομένων. Παράδειγμα αποτελεί ένα γράφημα που αποτυπώνει τη διακύμανση της υγρασίας σ’ ένα χώρο κατά τη διάρκεια μιας εβδομάδας.

Οι αισθητήρες χωρίζονται σε κατηγορίες με κριτήριο διαφοροποίησης το υλικό κατασκευής τους και γενικότερα με βάση το είδος της μέτρησης. Στην πρώτη κατηγορία κατατάσσονται οι αισθητήρες από αγώγιμα, ημιαγώγιμα, υπεραγώγιμα, διηλεκτρικά και μαγνητικά υλικά. Στη δεύτερη παρουσιάζονται οι οπτικοί, οι ακουστικοί, οι ακτινοβολίας, οι θερμικοί, οι μηχανικοί, οι χημικοί και οι ηλεκτρικοί.

Πρώτο Μέρος

Περιβάλλον IoT

2 Κεφάλαιο

2.1 Τεχνικά χαρακτηριστικά

2.1.1 Βασική αρχιτεκτονική του Διαδικτύου των Πραγμάτων

Η αρχιτεκτονική του IoT αναφέρεται στο συνολικό σχεδιασμό και τη δομή ενός συστήματος IoT. Τα Πράγματα, δηλαδή αντικείμενα εξοπλισμένα με αισθητήρες και επεξεργαστές επικοινωνούν μεταξύ τους για να εξυπηρετήσουν έναν ουσιαστικό σκοπό και να εξασφαλίσουν συγκεκριμένες λειτουργίες. Η επικρατέστερη IoT αρχιτεκτονική είναι η αρχιτεκτονική πέντε επιπέδων [4].

Τα πέντε αυτά επίπεδα, ξεκινώντας από το χαμηλότερο προς το υψηλότερο, είναι τα εξής:

- Επίπεδο Αντίληψης: Σε αυτό το επίπεδο, οι αισθητήρες και οι συσκευές συλλέγουν δεδομένα από το περιβάλλον.
- Επίπεδο Δικτύου: Εδώ, τα δεδομένα μεταδίδονται από τις συσκευές στο νέφος(cloud) ή σε άλλες συσκευές μέσω διαφόρων τεχνολογιών επικοινωνίας.
- Επίπεδο Εφαρμογών: Μέσω διαφόρων εφαρμογών, ο χρήστης μπορεί να έρθει σε επαφή με τα έξυπνα αντικείμενα που παρέχει το επίπεδο αντίληψης.

Τα τρία αυτά επίπεδα είναι τα βασικά της φιλοσοφίας του IoT σε όλες τις αρχιτεκτονικές του. Για ερευνητικούς λόγους προστέθηκε ένα επίπεδο μεταξύ των επιπέδων του δικτύου και των εφαρμογών που ονομάζεται επίπεδο επεξεργασίας. Επιπλέον παρατίθεται και το επίπεδο επιχειρήσεων.

- Επίπεδο Επεξεργασίας: Αυτό το επίπεδο είναι κρίσιμο για την εξαγωγή αξίας από τα δεδομένα που συλλέγονται από τους αισθητήρες. Εδώ εφαρμόζονται τεχνολογίες όπως η ανάλυση δεδομένων, η μηχανική μάθηση και η τεχνητή νοημοσύνη για την αναγνώριση μοτίβων, την πρόβλεψη τάσεων και τη λήψη αποφάσεων.

- Επίπεδο Επιχειρήσεων: Αυτό το επίπεδο αφορά στη διαχείριση και στον συντονισμό ολόκληρου του συστήματος IoT, συμπεριλαμβανομένων των επιχειρηματικών μοντέλων και της ασφάλειας.

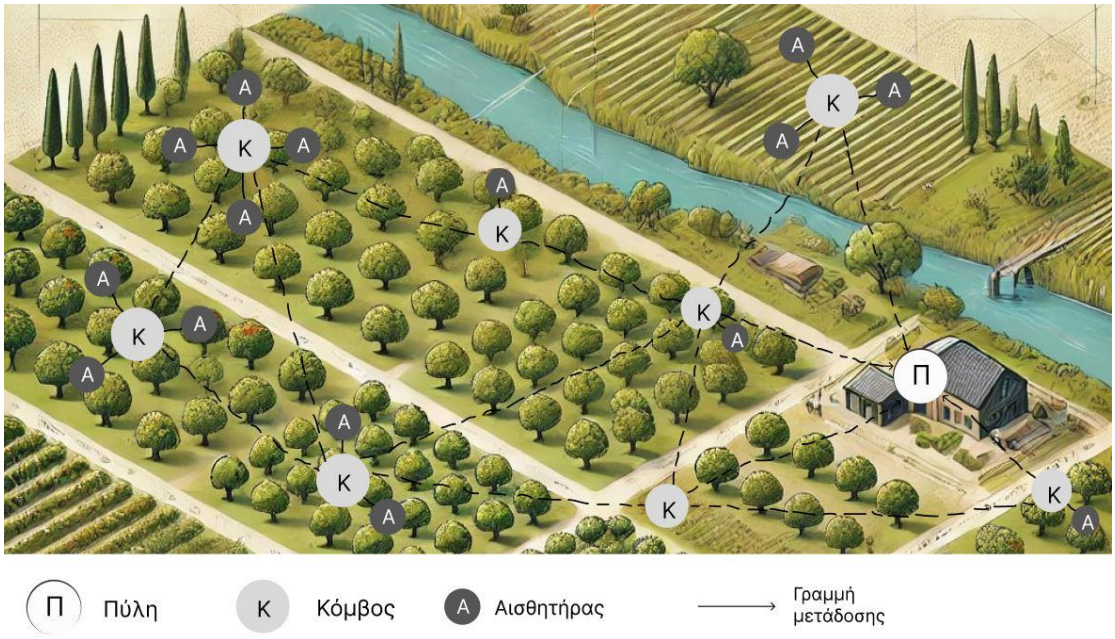
Ενώ η αρχιτεκτονική πέντε επιπέδων είναι ευρέως διαδεδομένη, υπάρχουν και άλλες αρχιτεκτονικές IoT που μπορεί να έχουν διαφορετική δομή ή να δίνουν έμφαση σε συγκεκριμένες πτυχές, όπως η ασφάλεια και η διαλειτουργικότητα [5].

2.1.2 Περίπτωση χρήσης αναφορικά με το επίπεδο δικτύου του IoT

Σε μία εφαρμογή του IoT σε γεωργική καλλιέργεια περιλαμβάνονται τα εξής:

- **Δίκτυο (Network):** Δίκτυο υπολογιστικών οντοτήτων που αλληλοεπιδρούν και συνεργάζονται μεταξύ τους με τη βοήθεια κεραιών για να καλύψουν σποραδικά μια έκταση γης και να προσφέρουν στον αγρότη εικόνα για τις συνθήκες μετρούμενων αγαθών/μεγεθών.
- **Αισθητήρας (Sensor):** Οντότητα που εξάγει πληροφορίες για μια παρατηρούμενη ιδιότητα.
- **Ενεργοποιητές (Actuators):** Μορφοτροπέας που μετατρέπει το ηλεκτρικό σήμα σε μηχανική κίνηση στο πραγματικό κόσμο.
- **Πύλη IP (IP Gateway):** Συσκευή του δικτύου που λειτουργεί ως σταθμός βάσης και προεπιλεγμένα ως πύλη που συνδέεται μέσω διαδικτύου στο cloud και σε άλλους κόμβους του δικτύου για να λαμβάνει δεδομένα από τις καλλιέργειες.
- **Κόμβος (Node):** Συσκευή του δικτύου που λειτουργεί μεσολαβητικά συλλέγοντας δεδομένα από προσαρτημένους αισθητήρες με σκοπό να αποσταλούν σε μια πύλη.
- **Πύλη (Gateway):** Συσκευή του δικτύου που συνδέεται με τις άλλες συσκευές μέσω κεραίας και δικτύου κινητής τηλεφωνίας.
- **Τοπολογία Πλέγματος Δικτύου (Mesh Network):** Το δίκτυο στο οποίο οι κόμβοι έχουν τη δυνατότητα να στείλουν από τον κόμβο έναρξης στο κόμβο προορισμού ένα μήνυμα μέσω διαφορετικών οδών κόμβων [6]

Ένα δίκτυο σε συνδυασμό με τους αισθητήρες και το λογισμικό αποτελούν ένα σύστημα παρακολούθησης. Για το σκοπό της παρακολούθησης σχεδιάζεται το δίκτυο ώστε να καλύπτει στοχευμένα και αποδοτικά τις καλλιέργειες. Ένα δίκτυο πλεγμάτων αποτελείται από μια πύλη (IP συνήθως η όχι) και επίσης από αρκετούς κόμβους, ο καθένας με μερικούς αισθητήρες. Οι κόμβοι έχουν τη δυνατότητα να συνεργάζονται μεταξύ τους, να συγκρατούν δεδομένα και να τα αποστέλλουν μαζικά στην πύλη. Οι κόμβοι τοποθετούνται σε σημείο υψηλό, ώστε να μην παρεμποδίζεται η επικοινωνία μεταξύ τους. Επίσης ενδεικτικά το εύρος στην μέγιστη απόσταση σύνδεσης τέτοιων συσκευών ξεκινά από αρκετές δεκάδες και εκατοντάδες μέτρα φτάνοντας έως και χίλια μέτρα. Οι αισθητήρες συνδέονται ασύρματα ή καλωδιωμένα σε κάποιο κόμβο του δικτύου για να αποσταλούν μέσω κεραιών στην πύλη.

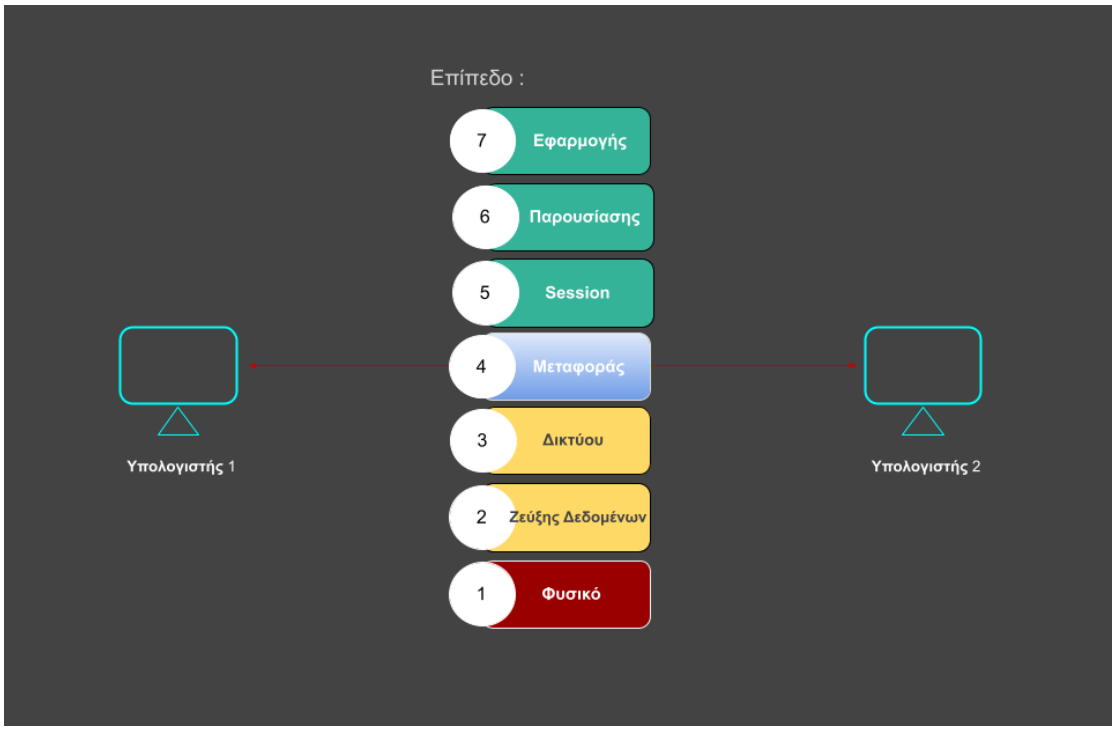


Εικόνα 2: Δικτυακή κάλυψη για τη μετάδοση συλλεγμένων δεδομένων

2.2 Περαιτέρω περί δικτύωσης

2.2.1 Το μοντέλο OSI

Το μοντέλο αναφοράς Ανοικτής Διασύνδεσης Συστημάτων (αγγλικά: Open Systems Interconnect model – OSI model) αποτελεί ένα θεωρητικό πλαίσιο που παρέχει έναν τρόπο σκέψης για τη δικτύωση. Διαχωρίζει την επικοινωνία δυο συσκευών μέσω δικτύου σε επτά αφαιρετικά επίπεδα.



Εικόνα 3: Διαστρωμάτωση 7 επιπέδων του Μοντέλου OSI

2.2.2 Πρωτόκολλα στο διαδίκτυο

Για τη λειτουργία του IoT απαιτείται μια ποικιλία τεχνολογιών. Μερικές από τις πιο συχνές τεχνολογίες παρουσιάζονται εδώ, ώστε να περιγράφει μια τυπική δράση του συστήματος, όταν ένας χρήστης κάνει αίτημα μιας web εφαρμογής στο διαδίκτυο.

Το Transmission Control Protocol (TCP) είναι το εξελιγμένο πρωτόκολλο στο στρώμα μεταφοράς και είναι αυτό που προσφέρει μια αξιόπιστη, και προσανατολισμένη στη σύνδεση υπηρεσία ροής Byte [7]. Μαζί με το TCP το

απλοποιημένο User Datagram Protocol (UDP) ως αποπλέκτης είναι το άλλο βασικό πρωτόκολλο στο στρώμα μεταφοράς του OSI.

Το κύριο πρωτόκολλο στο επίπεδο των εφαρμογών είναι το πρωτόκολλο μεταφοράς υπερκειμένου (αγγλικά: HyperText Transfer Protocol – HTTP). Είναι, επίσης, το κύριο πρωτόκολλο στη μετάδοση δεδομένων μεταξύ συσκευών και συγκεκριμένα φυλλομετρητών του παγκόσμιου ιστού. Σχεδιάστηκε για τη μεταφορά δεδομένων και σκοπό έχει τη σύνδεση του διακομιστή (server) και του πελάτη (client).

Όταν ένας χρήστης κάνει ένα αίτημα HTTP σε έναν διακομιστή ιστού, τα δεδομένα περνούν από έναν μετασχηματισμό σε κάθε επίπεδο του μοντέλου OSI. Πρώτα, το επίπεδο εφαρμογής προσθέτει την επικεφαλίδα HTTP που περιέχει πληροφορίες σχετικά με την αίτηση. Στη συνέχεια, το επίπεδο μεταφοράς ενθυλακώνει τα δεδομένα σε τμήματα με επικεφαλίδες TCP για αξιόπιστη παράδοση. Στη συνέχεια, το επίπεδο δικτύου προσθέτει μια επικεφαλίδα IP με διευθύνσεις IP πηγής και προορισμού για τη δρομολόγηση στο διαδίκτυο. Στο επίπεδο ζεύξης δεδομένων, προστίθεται μια επικεφαλίδα MAC που περιέχει τις διευθύνσεις Media Access Control για την επικοινωνία σε τοπικό δίκτυο. Τέλος, το φυσικό επίπεδο μεταδίδει τα ενθυλακωμένα δεδομένα ως ακατέργαστα bits στο καλώδιο δικτύου. Ο διακομιστής ιστού στο άκρο λήψης αντιστρέφει αυτή τη διαδικασία, εξάγοντας τα δεδομένα διαδοχικά από την πολυτμηματική κατασκευή τους για την επεξεργασία του αιτήματος HTTP.

2.2.3 Πρότυπα ασύρματης και ενσύρματης μετάδοσης

Στο φυσικό στρώμα του OSI και στο στρώμα ζεύξης η συνδεσιμότητα αφορά μηχανήματα και πύλες. Σχετικά με τεχνολογίες ασύρματων δικτύων προσωπικής περιοχής υπάρχουν παραδείγματα προτύπων όπως τα IEEE 802.15.x, ZigBee, Bluetooth. Από την άλλη στις τεχνολογίες τοπικών δικτύων υπάρχουν τα Meter-Bus (M-BUS), τα Power Line Communications και το Wi-Fi (IEEE 802.11x).

3 Κεφάλαιο

3.1 Εφαρμογή IoT στη γεωργία

3.1.1 Εισαγωγή

Η γεωργία έχει υποστεί ένα σημαντικό μετασχηματισμό με την έλευση των τεχνολογιών IoT. Ο μετασχηματισμός οφείλεται στην ανάγκη τον αγροτών να βελτιώσουν σε ορισμένες περιπτώσεις τις παραδοσιακές μεθόδους καλλιέργειας. Στις παραδοσιακές μεθόδους οι αγρότες δαπανούν περίσσιο χρόνο και ενέργεια. Αφενός βρίσκονται σε συνεχή επαγρύπνηση για την επιτήρηση της μετεωρολογικής συνθήκης που επικρατεί και αφετέρου ελέγχουν, εξασφαλίζουν και ρυθμίζουν τα ποιοτικά χαρακτηριστικά και τις παραμέτρους ανάπτυξης των καλλιεργειών τους. Αυτές οι ανάγκες αυξάνονται κλιμακωτά στις μεγαλύτερες καλλιέργειες. Ως εκ τούτου, για τις καλλιέργειες τους, οι αγρότες εκ των πραγμάτων χρειάζονται τόσο την άμεση πληροφόρηση για τις παραμέτρους του εδάφους, της κατάστασης των φυτών, τις ατμοσφαιρικές συνθήκες όσο και την αυτοματοποίηση.

Εικόνα 4: Παράμετροι για την ανάπτυξη καλλιεργειών

3.1.2 Παραδείγματα άρδευσης με IoT

Η τεχνολογία αισθητήρων διαδραματίζει καίριο ρόλο στην εξοικονόμηση νερού, ιδιαίτερα στη γεωργία ακριβείας. Αισθητήρες υγρασίας εδάφους παρέχουν δεδομένα σε πραγματικό χρόνο για την περιεκτικότητα του εδάφους σε νερό, επιτρέποντας στους αγρότες να ποτίζουν τις καλλιέργειες μόνο όταν είναι απαραίτητο. Αυτή η στοχευμένη προσέγγιση μειώνει σημαντικά τη σπατάλη νερού σε σύγκριση με τις παραδοσιακές μεθόδους άρδευσης.

Σε πιλοτική καλλιέργεια πατάτας στη Κύπρο, ερευνητές παρατήρησαν την δυνατότητα μείωσης 22% της ποσότητας νερού, συσχετίζοντας την υγρασία του εδάφους σε διάφορα βάθη με την παρακολούθηση της άρδευσης [8].

Η μελέτη ερευνητών παρουσιάζει ένα σύστημα αυτοματοποιημένης άρδευσης που ονομάζεται HidroBus®, το οποίο χρησιμοποιεί ασύρματα δίκτυα αισθητήρων για να επιτρέπει τον κεντρικό έλεγχο και την εποπτεία μεγάλων αρδευόμενων εκτάσεων. Το σύστημα αυτό έχει εφαρμοστεί στην Jumilla, στη περιοχή Murcia της Ισπανίας, και επιτυγχάνει εξοικονόμηση νερού 30-60% μέσω του ελέγχου χιλίων οκτακοσίων πενήντα πομονών σε πραγματικό χρόνο και της βελτιστοποίησης της άρδευσης ανάλογα με τις ανάγκες [9].

3.1.3 Τύποι αισθητήρων που χρησιμοποιούνται στη γεωργία

Ένας από τους πιο συχνά χρησιμοποιούμενους τύπους αισθητήρων στη γεωργία είναι ο αισθητήρας υγρασίας εδάφους. Αυτός αποτελεί τον πυρήνα των αυτοματοποιημένων συστημάτων και των νέων πρακτικών άρδευσης. Σε συνδυασμό με τους ελεγχόμενους ενεργοποιητές και τους αυτόματους κρουνούς βοηθά τους αγρότες να καθορίσουν τον χρόνο της άρδευσης και την ποσότητα του νερού. Για την αποτελεσματική αυτή νέα πρακτική αξιοποιούνται δεδομένα που σχετίζονται με την υγρασία της ατμόσφαιρας, τις βροχοπτώσεις, την υγρασία του εδάφους και τη θερμοκρασία.

Τοποθετούνται τόσο στις καλλιέργειες όσο και στα οικιακά φυτά. Παρακάτω εικονίζονται βυθισμένες σε δύο διαφορετικά βάθη σωλήνες που ενθυλακώνουν τον αισθητήρα υγρασίας εδάφους. Επιπρόσθετα εικονίζεται και ένας οικιακός αισθητήρας υγρασίας εδάφους.

Εικόνα 5: Μέτρηση υγρασίας εδάφους με αισθητήρες

Οι αισθητήρες καιρού αποτελούν μια άλλη ζωτικής σημασίας κατηγορία στα γεωργικά συστήματα IoT, παρέχοντας δεδομένα για παραμέτρους όπως η βροχόπτωση, η ταχύτητα του ανέμου, η θερμοκρασία και η ηλιακή ακτινοβολία. Μια τέτοια πληροφόρηση είναι απαραίτητη για τον μετριασμό των κινδύνων που εγκυμονούν τα ακραία καιρικά φαινόμενα.

Εικόνα 6: Αισθητήρας παραμέτρων καιρού

Οι καινοτομίες στην τεχνολογία των αισθητήρων επηρεάζουν σημαντικά και τους καλλιεργητές. Αισθητήρες που μετρούν τα επίπεδα θρεπτικών στοιχείων του εδάφους παρέχουν μια οικονομικά αποδοτική εναλλακτική λύση σε σχέση με τις παραδοσιακές εργαστηριακές δοκιμές. Το υγιές, και παραγωγικό έδαφος έχει πορώδη δομή και καλή ικανότητα να δέχεται και να αποθηκεύει νερό. Έχει επίσης πλούσια οργανική περιεκτικότητα και επιτρέπει την ανάπτυξη των φυτών και την υψηλή πρόσληψη θρεπτικών στοιχείων.

Οι αισθητήρες για την ανίχνευση παρασίτων και ασθενειών είναι ένα άλλο αναδυόμενο τεχνολογικό εργαλείο για τους καλλιεργητές. Έξυπνα συστήματα του είδους όχι μόνο αποτρέπουν τις απώλειες των καλλιεργειών αλλά και μειώνουν την ανάγκη για υπερβολική χρήση φυτοφαρμάκων.

Εικόνα 7: Μήλο και φουζικλάδιο

Εικόνα 8: Αισθητήρας φύλλου

3.1.4 Νέες δυνατότητες και διαχείριση πόρων

Η ενσωμάτωση αλγορίθμων μηχανικής μάθησης σε αισθητήρες ενισχύει περαιτέρω τη χρησιμότητα τους. Ο συνδυασμός αισθητήρων θρεπτικών ουσιών με προγνωστικά μοντέλα επιτρέπει την εκτίμηση σε πραγματικό χρόνο της διαθεσιμότητας θρεπτικών στοιχείων στο έδαφος, βελτιώνοντας τις στρατηγικές λίπανσης. Οπτικοί αισθητήρες σε συνδυασμό με αλγορίθμους βαθιάς μάθησης εντοπίζουν ασθένειες των φυτών με εξαιρετική ταχύτητα και ακρίβεια. Έτσι λοιπόν η τεχνολογία των αισθητήρων συνδυάζεται και με την τεχνητή νοημοσύνη και τα μεγάλα δεδομένα.

Οι αισθητήρες μέσω του IoT δίνουν τη δυνατότητα τόσο στους αγρότες να αποκτήσουν ακριβείς και σε τρέχοντα χρόνο πληροφορίες για τις συνθήκες που επηρεάζουν τις καλλιέργειες και το έδαφος. Επιπλέον, επιτρέπουν τη δημιουργία ιστορικού καταγραφής και συλλογής δεδομένων στο διαδίκτυο στοχεύοντας σε ευρύτερη χρήση τεχνολογιών τεχνητής νοημοσύνης στο πρότυπο των προηγούμενων μοτίβων.

Η αύξηση του πληθυσμού παγκοσμίως συνάδει με την ολοένα και αυξανόμενη ανάγκη για επισιτισμό. Σύμφωνα με την ερευνήτρια Marta κ.α. διερευνήθηκαν οι επιπτώσεις που έχει η αύξηση του πληθυσμού στις απαιτήσεις για τρόφιμα και νερό με ορίζοντα το 2050 [10]. Ωστόσο, νωρίτερα, ο καθηγητής Beddington [11] παρέχει μια πιο βραχυπρόθεσμη προοπτική, προβλέποντας αύξηση 50% στην παραγωγή τροφής και 30% στην απαιτούμενη ποσότητα νερού έως το 2030. Με την ευφυή γεωργία παρέχεται η δυνατότητα βελτίωσης της παραγωγικής διαδικασίας και των αποδόσεων της σε συνδυασμό με την ασφαλέστερη διαχείριση των πόρων. Έτσι με την εφαρμογή της μπορούν να αντιμετωπιστούν τα προβλήματα επισιτισμού στον παγκόσμιο πληθυσμό.

Εικόνα 9: Καλλιέργεια στους πρόποδες του Βερμίου

Δεύτερο Μέρος

Η υλοποίηση του DashPL

4 Κεφάλαιο

4.1 Σχεδίαση εφαρμογής DashPL

4.1.1 Σκοπός του λογισμικού

Το σύστημα DashPL έχει ως στόχο να παρουσιάζει τα δεδομένα μετρήσεων σε χρονοσειρές με ένα προσιτό και χρηστικό τρόπο. Η παρουσίαση γίνεται μέσω μιας σύγχρονης διαδικτυακής εφαρμογής ιστού συμβατή με φυλλομετρητές όπως Chrome, Mozzila, Safari. Συγκεκριμένα είναι δυνατή η δημιουργία γραφήματος του μετρούμενου μεγέθους με δυνατότητες αυτόματης προσαρμογής ορίων, εστίασης, καταγραφής δεδομένων και εξαγωγής αυτών σε μορφή csv. Επιπρόσθετα, είναι δυνατή και η εξαγωγή του γραφήματος σε εικόνα png. Καταληκτικά, η διασύνδεση και παρουσίαση των δεδομένων γίνεται σε ένα σύγχρονο και ελκυστικό διαδικτυακό περιβάλλον.

4.1.2 Λειτουργικές απαιτήσεις του λογισμικού

Το frontend περιβάλλον προορίζεται για τη διεπαφή με τον χρήστη. Πρόκειται για το βασικό κορμό του συστήματός και όλες οι δυνατότητες που προσφέρονται είναι αξιοποιήσιμες μέσα σε αυτό.

Το λογισμικό συσχετίζεται με τις εξής πηγές:

1. www.react.dev
2. www.rabbitmq.com
3. www.tailwindcss.com
4. www.github.com/recharts/recharts
5. www.docker.com
6. ESP32 – Wikipedia
7. Arduino IDE

Ο άμεσα εμπλεκόμενος Χρήστης είναι εκείνος που παρακολουθεί, παρατηρεί και καταγράφει τις μετρήσεις από το frontend, ενώ ο εμμέσως

εμπλεκόμενος χρήστης απαιτείται να συνδέσει τον αισθητήρα στο localhost μέσω του router.

Δεδομένα εισόδου:

- Όνομα του γραφήματος
- Περιγραφή του γραφήματος
- Οι οριοθετημένες ακραίες τιμές του γραφήματος

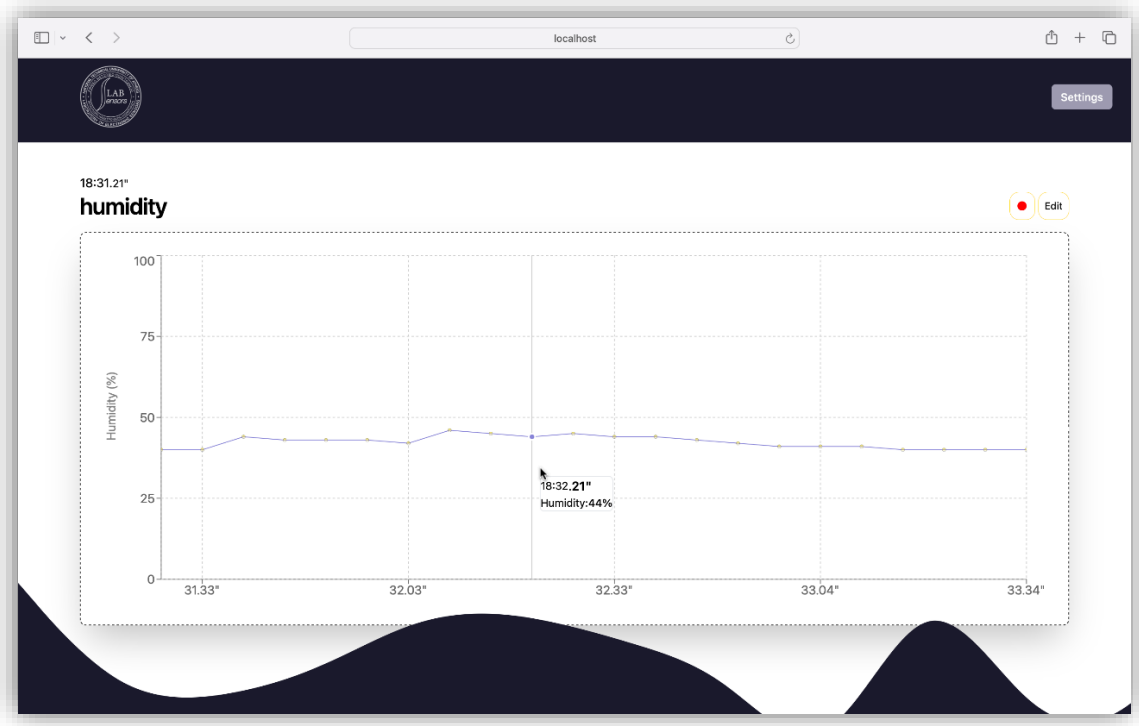
Δεδομένα εξόδου

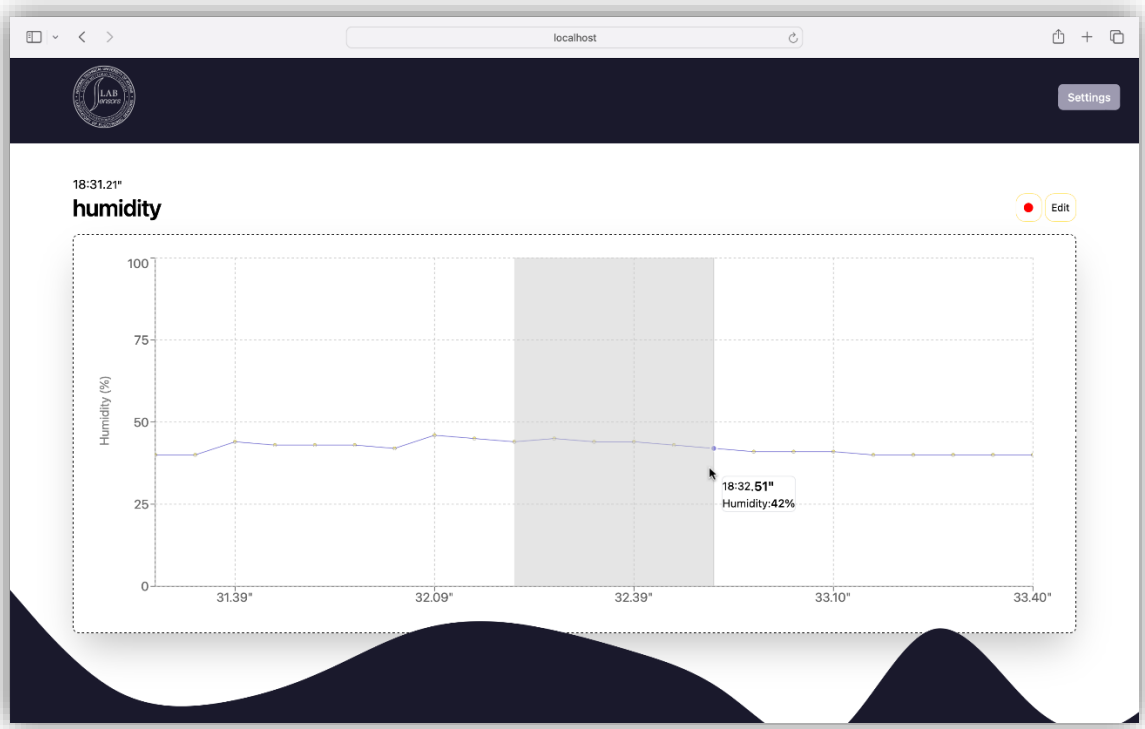
- Γράφημα

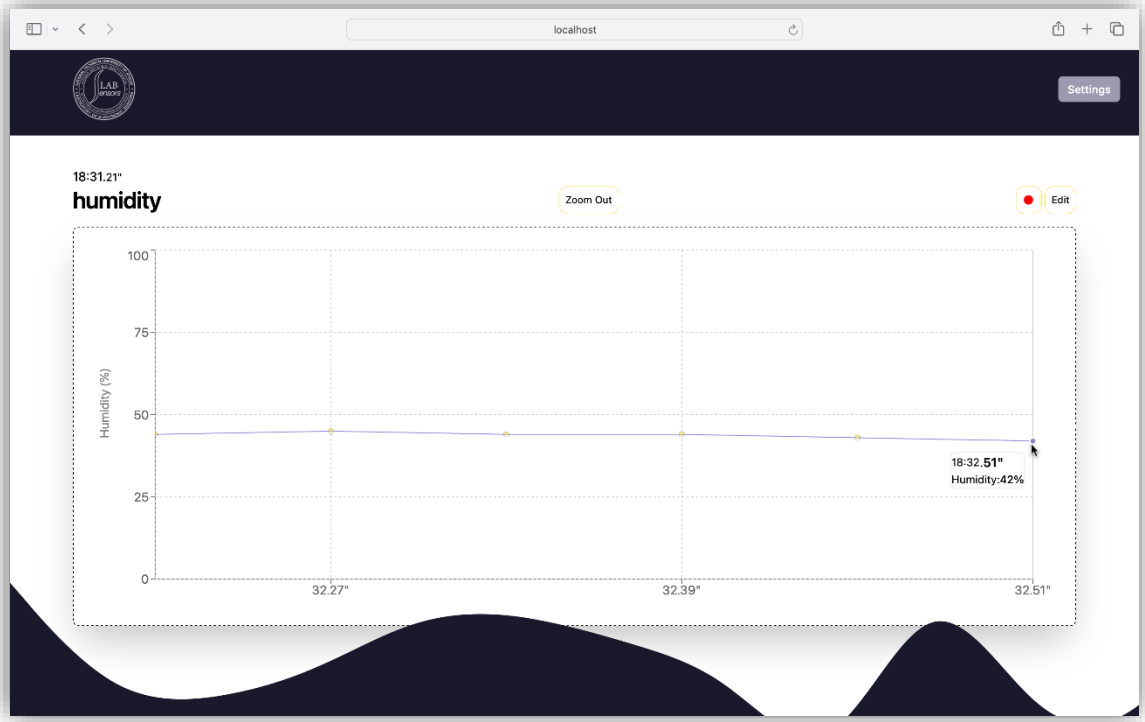
Περίπτωση χρήσης 1 Παρακολούθηση επιλεγμένων τιμών

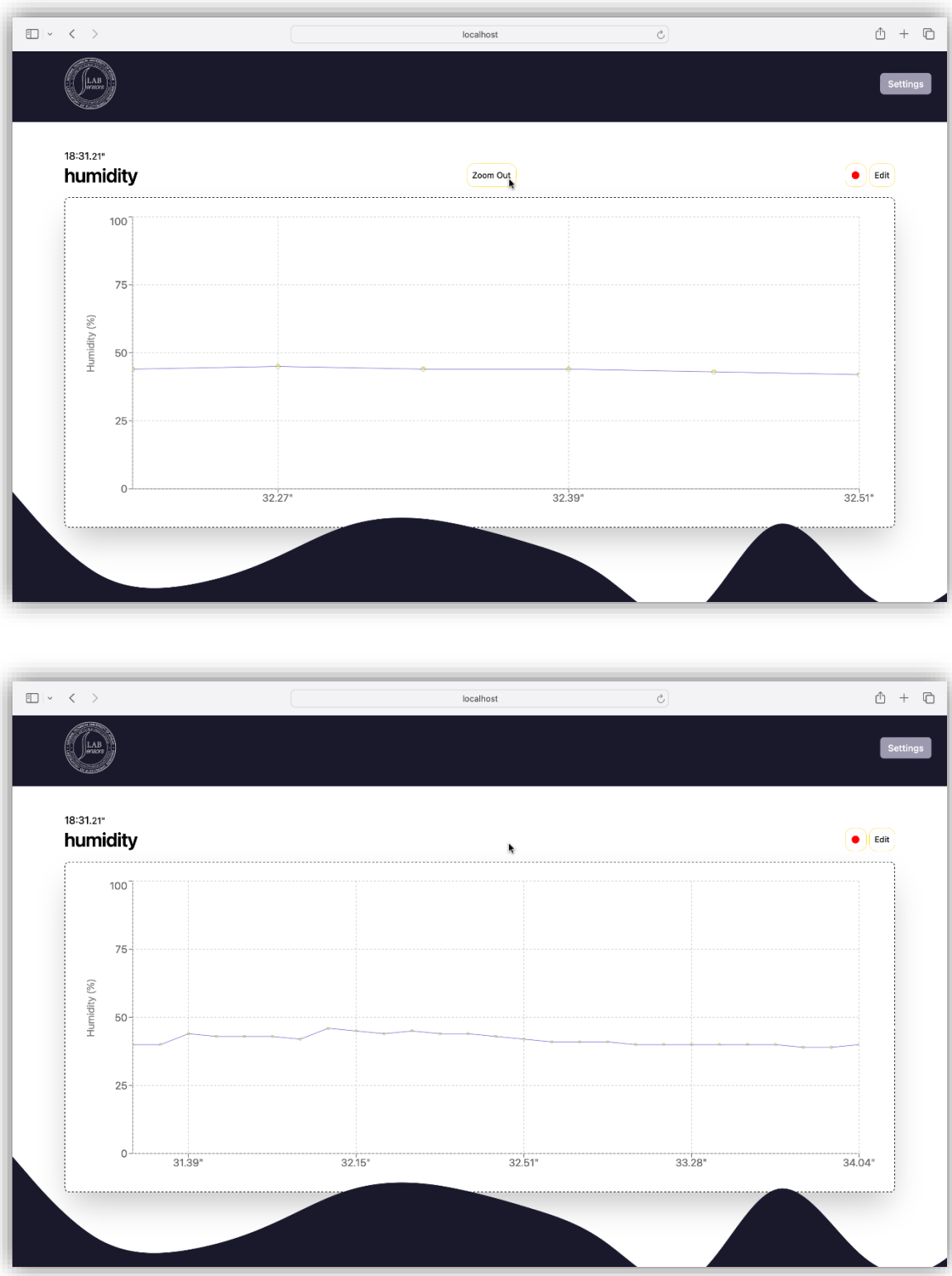
Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Ο χρήστης πλοηγείται στη σελίδα σύνδεσης. Παρακολουθεί στην οθόνη του τις τιμές του συνδεδεμένου αισθητήρα σε τρέχοντα χρόνο. Εν συνεχεία εστιάζει σε ορισμένο εύρος χρόνου. Στις μετρήσεις γίνεται το λεγόμενο “zoom in” και εμφανίζεται επιλογή για “zoom out”. Ο χρήστης μπορεί να κάνει επιπλέον εστίαση στα σημεία ενδιαφέροντος.




localhost
Settings
18:31.21"
humidity
Edit
100
75
50
25
0
Humidity (%)
31.39"
32.09"
32.39"
33.10"
33.40"
18:32.51"
Humidity:42%


localhost
Settings
18:31.21"
humidity
Zoom Out
Edit
100
75
50
25
0
Humidity (%)
32.27"
32.39"
32.51"
18:32.51"
Humidity:42%

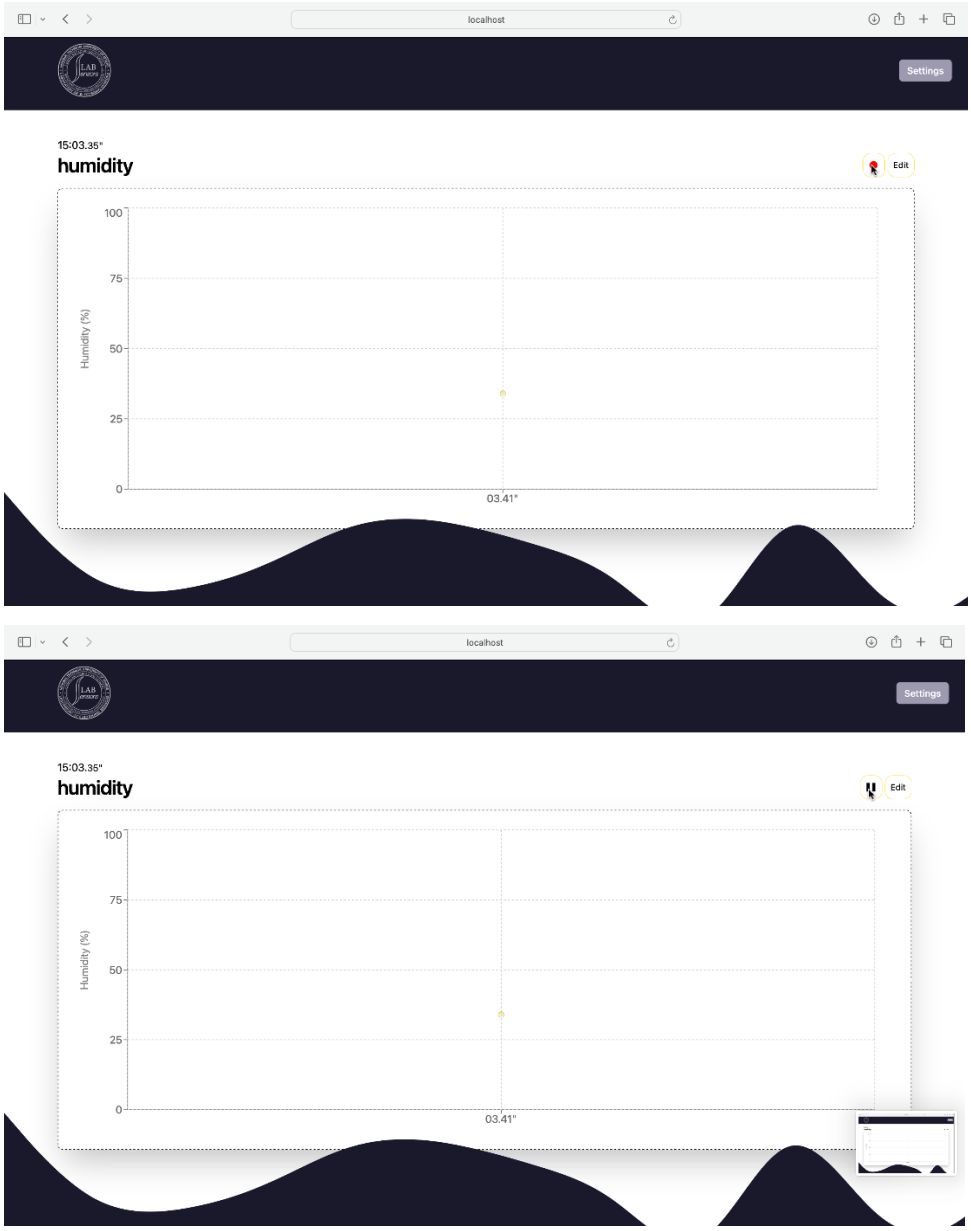


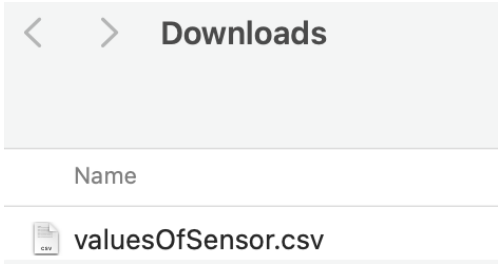
Εικόνα 10: Παρακολούθηση επιλεγμένων τιμών στο DashPL

Περίπτωση χρήσης 2 Εγγραφή τιμών και αποθήκευση σε αρχείο csv

Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Ο χρήστης αφού προηγηθεί στην ιστοσελίδα κάνει εγγραφή. Η εγγραφή νέων τιμών ξεκινά πατώντας το κόκκινο κουμπί εγγραφής. Όσο έρχονται νέες τιμές αυτές καταγράφονται στο παρασκήνιο. Με το που επιλέξει το “stop/pause button” παρουσιάζονται δύο επιλογές. Είτε να αποθηκεύσει στο τοπικό δίσκο τις τιμές, είτε να μην τις αποθηκεύσει και να επιστρέψει πάλι στο κόκκινο κουμπί εγγραφής.





valuesOfSensor

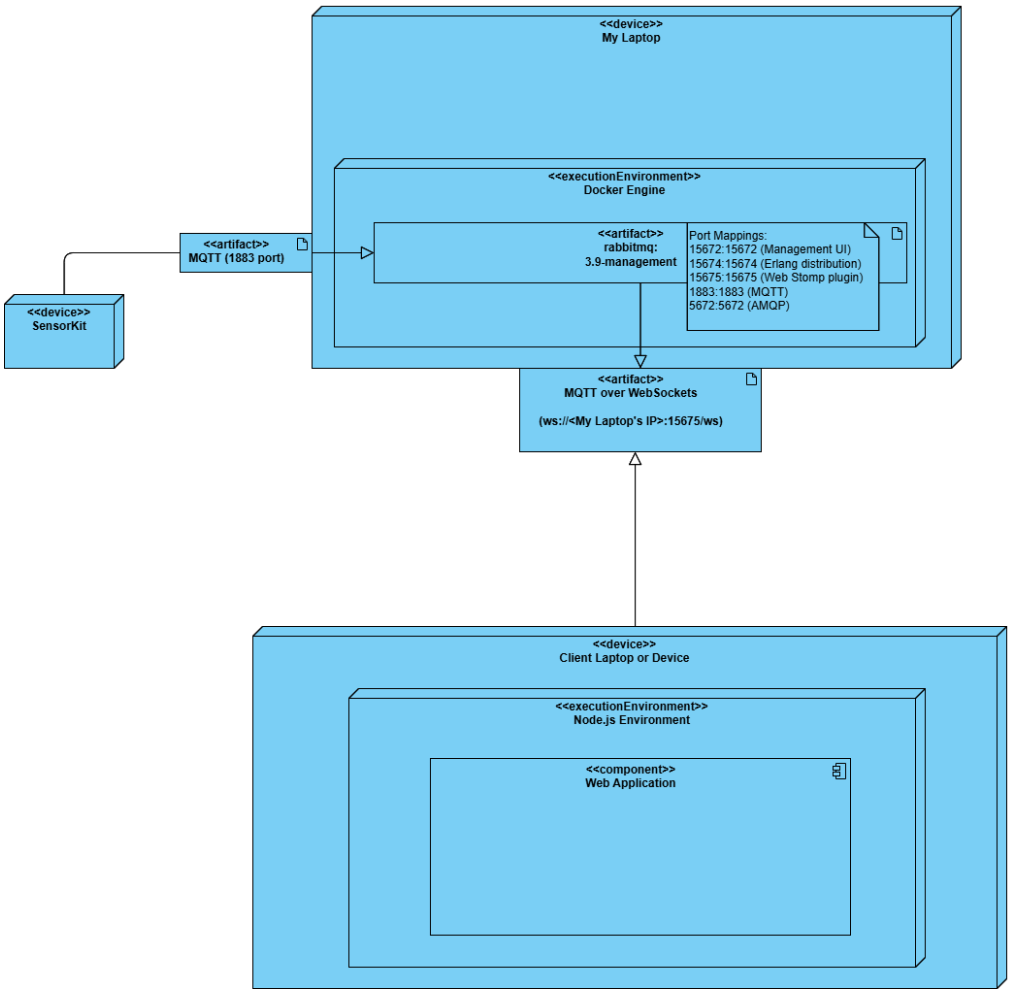
Timestamp	Humidity %
January 12 of 2025 at 2:59:34 PM GMT+2	35
January 12 of 2025 at 2:59:40 PM GMT+2	36
January 12 of 2025 at 2:59:46 PM GMT+2	36
January 12 of 2025 at 2:59:52 PM GMT+2	36
January 12 of 2025 at 2:59:58 PM GMT+2	35
January 12 of 2025 at 3:00:04 PM GMT+2	35
January 12 of 2025 at 3:00:10 PM GMT+2	34

Εικόνα 11: Εγγραφή τιμών και αποθήκευση σε αρχείο csv

4.2 Περί συστήματος

4.2.1 Ενοποιημένη Γλώσσα Σχεδιασμού Προτύπων

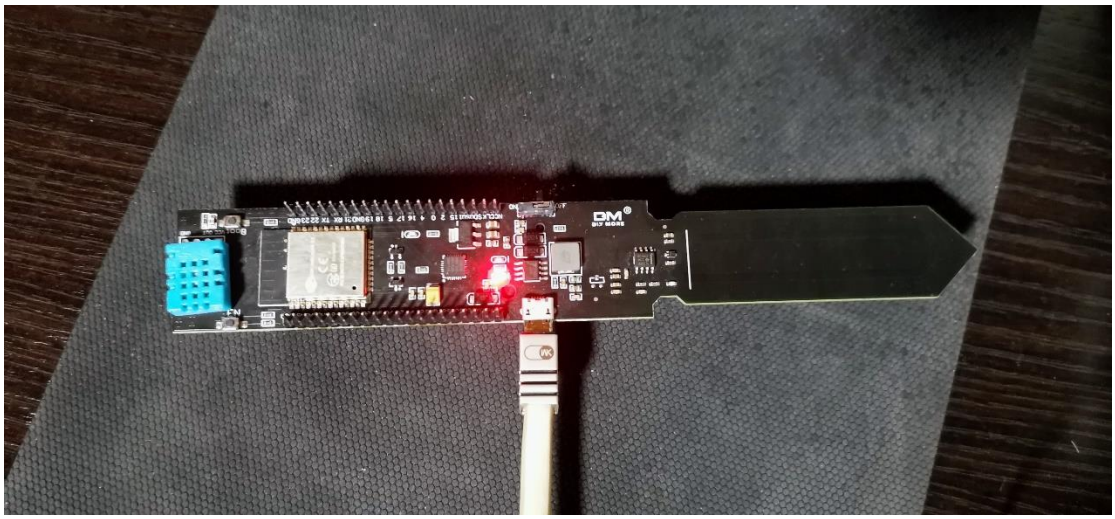
Η ενοποιημένη γλώσσα σχεδιασμού προτύπων (αγγλικά: unified modeling language – UML) στη μηχανική λογισμικού χρησιμοποιείται για τη δημιουργία διαγραμμάτων που απεικονίζουν τη δομή, τη συμπεριφορά και τις αλληλεπιδράσεις των στοιχείων ενός λογισμικού. Παρακάτω παρατίθεται το διάγραμμα ανάπτυξης του συστήματος με την γλώσσα UML. Σε αυτό οπτικοποιούνται οι διασυνδέσεις των κόμβων επεξεργασίας καθώς και τα στοιχεία που βρίσκονται σε αυτούς όσο τρέχει η εφαρμογή.



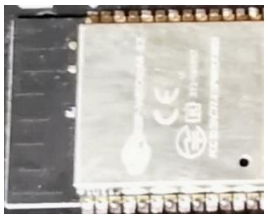
Εικόνα 12: UML deployment διάγραμμα

4.2.2 Περιγραφή στοιχείων συστήματος

Υπάρχει πληθώρα στοιχείων που συναποτελούν το σύστημα του DashPL. Η μέτρηση μέσω κάποιου αισθητήρα και η παραγωγή αναλογικού σήματος, η μετατροπή του αναλογικού σήματος σε ψηφιακό, η επεξεργασία του, η αποστολή του σε ένα διαμεσολαβητή, η κατανάλωση του δεδομένου αυτού από τον υπολογιστή του πελάτη και τέλος η παρουσίαση της πληροφορίας στην οθόνη.



Αρχικά, όπως και στο παραπάνω διάγραμμα, υπάρχει η συσκευή με κάποιους αισθητήρες. Η συσκευή αυτή διαθέτει επεξεργαστική ισχύ, μνήμη και υποστήριξη της συνδεσιμότητας στο διαδίκτυο με ασύρματο τρόπο. Για τις παραπάνω λειτουργίες είναι υπεύθυνο το chip του Esp32.

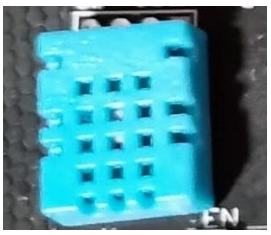


Αυτός ο μικροελεγκτής προγραμματίζεται μέσω του Arduino IDE με κώδικα C. Αυτό δεν αποκλείει και τη χρήση και άλλων γλωσσών προγραμματισμού για τον κώδικα. Ο κώδικας αρχικά αφορά κατά την έναρξη του μικροελεγκτή τα εξής βήματα εγκατάστασης:

1. Τον εντοπισμό γνωστού δικτύου Wi-Fi και τη σύνδεση σε αυτό με τους κωδικούς του.

Πρωτού ξεκινήσει η επικοινωνία με τον επιλεγμένο αισθητήρα έχουν προηγηθεί δύο βήματα.

2. Η λήψη την τρέχουσας ώρας μέσω του διαδικτύου
3. Η σύνδεση με το επόμενο στοιχείο του διαγράμματος, δηλαδή με έναν διαμεσολαβητή ο οποίος αναλύεται παρακάτω. Η διασύνδεση αυτή κάνει χρήση του MQTT, ενός πρωτοκόλλου κατάλληλου για το διαδίκτυο των πραγμάτων και την αποδοτική και ελαφριά σε όγκο και ενέργεια αποστολή των επιλεγμένων δεδομένων.
4. Με χρήση βιβλιοθήκης προγραμματισμού γίνεται έναρξη επικοινωνίας του μικροελεγκτή με τον αισθητήρα, εν προκειμένω τον DHT 11.



Ο DHT11 μετρά την υγρασία με τη χρήση τεχνολογίας των χωρητικών αισθητήρων υγρασίας. Η τεχνολογία αυτή δηλαδή χρησιμοποιεί δύο ηλεκτρόδια που σχηματίζουν ένα πυκνωτή. Μεταξύ αυτών υπάρχει ένα υλικό πολυμερούς που απορροφά την υγρασία από τον αέρα. Για την αρχή λειτουργίας του, όσο το επίπεδο υγρασίας στον αέρα αυξάνεται, τόσο μειώνεται η χωρητικότητα του πυκνωτή εφόσον το πολυμερές υλικό απορροφά τους αγώγιμους υδρατμούς. Η αλλαγή αυτή συλλέγεται και μεταδίδεται ενσύρματα σε μικροελεγκτή. Τελικά, αποδίδουμε στη μέτρηση του αισθητήρα μια τιμή ως ποσοστό σχετικής υγρασίας.

Τα επόμενα βήματα μπαίνουν σε επαναληπτική εκτελεστική διαδικασία.

5. Ο μικροελεγκτής Esp32 διαβάζει αρχικά το πρώτο δεδομένο σχετικής τιμής υγρασίας (εφόσον το βήμα εκτελείται για πρώτη φορά) από την ενσύρματη καλωδίωση με τον DHT11.
6. Ελέγχει και βεβαιώνει τη σύνδεση με το άλλο άκρο, εν προκειμένω τον διαμεσολαβητή μέσω Wi-Fi.

7. Δημιουργεί ένα προγραμματιστικό αντικείμενο με πεδία την τρέχουσα χρονοσφραγίδα και τις μετρήσεις από τους αισθητήρες της συσκευής.
8. Αποστέλλει το μήνυμα (το προγραμματιστικό αυτό αντικείμενο) στο διαμεσολαβητή μέσω Wi-Fi.
9. Ο μικροελεγκτής μέσω χρονοδιακόπτη σταματά να τρέχει για προγραμματισμένο πλήθος milli-second.

Εκτελούνται έπειτα επαναληπτικά τα βήματα 5 έως 9. Με αυτό τον τρόπο υπάρχει στο διαδίκτυο αποστολή δεδομένων από τους αισθητήρες ανά προγραμματισμένο (εν προκειμένω 6 δευτερόλεπτα) χρονικό διάστημα και επισυνάπτεται και η χρονοσφραγίδα της μέτρησης αυτής.

Για τη συγκεκριμένη συσκευή διατίθεται εκτός από τη μέτρηση της υγρασίας στον αέρα και η μέτρηση της θερμοκρασίας του αέρα καθώς και μέτρηση υγρασίας του εδάφους. Η τελευταία μέτρηση αφορά κυρίως τη σπιτική κηπουρική όπου ο αισθητήρας της συσκευής τοποθετείται κατακόρυφα και με το ένα τμήμα του (το μυτερό) μέσα στο χώμα.

Η συσκευή αυτή μπορεί να γίνει αυτόνομη ενεργειακά καθώς διαθέτει και δυνατότητα τροφοδότησης από μπαταρία τύπου 18650.

Έπειτα, όπως φαίνεται και στο διάγραμμα, υπάρχει το τμήμα του διαμεσολαβητή. Αυτός υφίσταται στον υπολογιστή μου και στόχο έχει τόσο να αποτρέψει την απώλεια μηνυμάτων σε περίπτωση που ο υπολογιστής του πελάτη διακόψει τη σύνδεση ή υπερφορτωθεί, όσο και να αποστέλλει πολλαπλά αντίγραφα σε άλλους πελάτες αναφορικά με την ίδια μέτρηση. Ο διαμεσολαβητής που χρησιμοποιείται είναι της RabbitMQ (RabbitMQ broker).

Ο RabbitMQ broker χρησιμοποιείται ευρέως για να εξασφαλίσει την αξιόπιστη ασύγχρονη επικοινωνία μεταξύ διαφορετικών συστημάτων ή υπηρεσιών που μπορούν και απομονώνονται και λειτουργούν ανεξάρτητα. Στο διαδίκτυο των πραγμάτων λειτουργεί ως προσωρινός χώρος αποθήκευσης επιτρέποντας μεγάλη διαχειριστική δύναμη στην επικοινωνία των υποσυστημάτων.

Το λογισμικό που χρησιμοποιήθηκε είναι το RabbitMQ management και εγκαταστάθηκε εύκολα μέσω του περιβάλλοντος Docker. Περιληπτικά το περιβάλλον αυτό περιγράφεται από:

- Docker Image: Ορίζεται η έκδοση του λογισμικού μαζί με τις εξαρτήσεις και το περιβάλλον στο οποίο θα “τρέξει” το προϊόν αυτό του κώδικα σε κάποια μηχανή.
- Docker Hub: Είναι ο διαδικτυακός χώρος όπου εταιρίες ανοιχτού κώδικα και μη ανεβάζουν τα λογισμικά τους ως image.
- Docker (Docker Desktop): Είναι το περιβάλλον στο οποίο αλληλοεπιδρούν οι χρήστες με τα λογισμικά αυτά. Πλεονέκτημα του Docker είναι ότι εκτελεί το λογισμικό σε πακέτα λογισμικού (τα container) ανεξάρτητα από το λειτουργικό σύστημα του εκάστοτε υπολογιστή.

Η διαδικασία εγκατάστασης και η μετέπειτα εκτέλεση του πακέτου RabbitMQ management παρακάμπτει πολλά βήματα καθώς οι εξαρτήσεις και οι βιβλιοθήκες του λογισμικού βρίσκονται μέσα στο πακέτο, το οποίο αποκτήθηκε μέσω του Docker Hub. Έτσι πλέον εκτελείται εύκολα και απομονωμένα στο Docker περιβάλλον.

Στο διάγραμμα της υποενότητας 5.2.1 φαίνεται και η σύνδεση των θυρών (ports) του πακέτου που τρέχει απομονωμένα στο Docker με τις θύρες του “My laptop”. Αυτές κρούονται τόσο από τη συσκευή με τον αισθητήρα όσο και από τους πιθανούς πελάτες που επιθυμούν να παρακολουθήσουν τις μετρήσεις στην οθόνη τους.

Στο Message Queue Telemetry Transport πρωτόκολλο (MQTT) πρωταγωνιστούν εκτός από το broker, εν προκειμένω της RabbitMQ, και οι πελάτες του broker, οι αποκαλούμενοι Publisher και Subscribers. Στο σύστημα μας υπάρχει ένας Publisher για κάθε πιθανό Esp32 που παράγει μηνύματα με τις μετρήσεις και τις χρονοσφραγίδες και πολλοί Subscribers εγγεγραμμένοι στο κανάλι επικοινωνίας του μοντέλου, οι οποίοι καταναλώνουν το μήνυμα αυτό. Το μοντέλο αυτό ονομάζεται Pub-Sub στο οποίο κάθε Publisher παράγει

μηνύματα, τα οποία καταναλώνονται από τους επιλεγμένους Subscribers μέσω ενός θεματικού διαύλου.

Στο διάγραμμα φαίνεται και ο πελάτης ο οποίος με τη συσκευή του μπορεί να δει στην οθόνη του μέσω του frontend λογισμικού την πληροφορία σε χρονοσειρά.

> Από τη σκοπιά του προγραμματιστή το παρόν κεφάλαιο εισάγει τη σχεδιαστική επιλογή της χρήσης μια μοντέρνας βιβλιοθήκης κώδικα, αυτής της React.

4.3 Σχεδιαστικές επιλογές στο DashPL

4.3.1 Ανασκόπηση της React στη στοίβα τεχνολογιών

Στη σύγχρονη ανάπτυξη λογισμικού για ιστοσελίδες έχει εδραιωθεί η χρήση του τρίπτυχου html, css, JavaScript. Ενώ η HTML (Hypertext Markup Language) περιγράφει τα δομικά στοιχεία μιας ιστοσελίδας, όπως οι χρονοσφραγίδες και ο τίτλος του γραφήματος που περιγράφονται ως συμβολοσειρές κειμένου, από την άλλη, με τη χρήση της CSS (Cascading Style Sheets) ο προγραμματιστής διατάσσει/μορφοποιεί τα παραπάνω δομικά στοιχεία στην οθόνη. Επιπλέον οι σύγχρονοι φυλλομετρητές με τη βοήθεια της JavaScript επεκτείνουν τις δυνατότητες τους και τον τρόπο με τον οποίο “βλέπουν” την HTML, ώστε να δώσουν ένα δυναμικό χαρακτήρα στις ιστοσελίδες. Έτσι και στην ιστοσελίδα του DashPL, ο χρήστης μπορεί και αλληλεπιδρά με τα δεδομένα. Καταληκτικά, τα δομικά στοιχεία μιας ιστοσελίδας μαζί με την διάταξη/μορφοποίηση αυτών και τις υποστηριζόμενες λειτουργίες για δυναμική αλληλεπίδραση και περιεχόμενο καθιστούν το DashPL μια μοντέρνα προσέγγιση στην ανάπτυξη του λογισμικού.

Κατά τη σχεδίαση επιλέχθηκε η χρήση της βιβλιοθήκης κώδικα της React.javascript (React.js) για την ανάπτυξη της ιστοσελίδας του DashPL. Με τη βοήθεια της βιβλιοθήκης αυτής ο κώδικας είναι δομημένος και ιεραρχημένος, ώστε ο προγραμματιστής έχει τη δυνατότητα να τον επεξεργάζεται και να τον διατηρεί ευκολότερα. Κατά τη διαδικασία συγγραφής κώδικα με τη React.js σε αντίθεση με την απλή javascript ο προγραμματιστής καθορίζει την επιθυμητή

διαδικασία ως στόχο και τα επιμέρους κομμάτια της διαδικασίας αφορούν πλέον μονάχα την εσωτερική λειτουργία της βιβλιοθήκης της React.js [12] & [13]. Πλέον η υλοποίηση λειτουργιών με κώδικα και η συντήρηση του είναι μια διαδικασία λιγότερο χρονοβόρα και ευκολότερη από παλαιότερα.

Η χρήση της React για τη σχεδίαση της διεπαφής χρήστη είναι απαραίτητη, καθώς με αυτή ο προγραμματιστής εκφράζει με τρόπο εύληπτο τις καταγεγραμμένες στον κώδικα διαδικασίες. Πλέον στο παρασκήνιο η διεπαφή χρήστη με το λογισμικό είναι μια συνάρτηση των δεδομένων. Με αυτήν την ιδιότητα που έχει η React απλοποιείται η διαδικασία σχεδίασης και υλοποίησης μιας ιστοσελίδας για τον προγραμματιστή [14].

4.3.2 Κώδικας στο περιβάλλον της React

Στην ιστοσελίδα του DashPL, τα δεδομένα και τα στοιχεία ενημερώνονται χωρίς αναμονή και επαναφορτώσεις. Δεν χρειάζεται η επαναφόρτωση της σελίδας, όπως γινόταν στα πρώιμα στάδια οι διαδικτυακές πλοηγήσεις. Η React ενημερώνει μονάχα τα απαραίτητα τμήματα/στοιχεία της ιστοσελίδας.

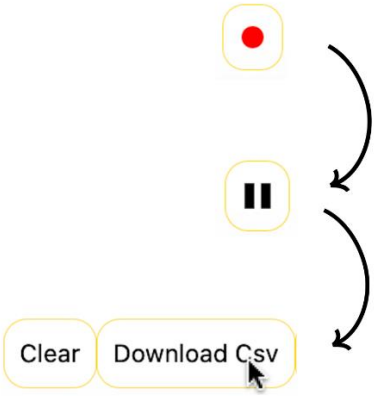
Τα συστατικά (αγγλικά: components) ενός κώδικα διασπούν τη σχεδίαση και υλοποίηση της διεπαφής χρήστη σε μικρότερα, διαχειρίσιμα κομμάτια. Τα κομμάτια αυτά είναι αυτοτελή και διασυνδέουν στοιχεία σχεδίασης μιας ιστοσελίδας της οθόνης με τμήματα κώδικα.

Εικόνα 13: 5 συστατικά ιεραρχημένα μιας ιστοσελίδας

Τα τμήματα αυτά του κώδικα της React στο DashPL εμπεριέχουν τόσο τα στοιχεία της HTML, μορφοποιημένα μέσω της tailwind CSS, όσο και πρωτίστως τη λογική διάδρασης που επιτρέπουν το χρήστη να πλοηγηθεί σε μια ιστοσελίδα. Ο κώδικας των συστατικών, επίσης, υπάρχει μέσα στα αρχεία με την προέκταση jsx. Στα αρχεία .jsx επιτρέπεται η HTML εντός JavaScript και με αυτόν το τρόπο καταγράφεται η λογική απεικόνισης στοιχείων με το περιεχόμενο των στοιχείων αυτών στο ίδιο μέρος.

Παρακάτω θα δοθεί επιλεγμένο τμήμα κώδικα από την εφαρμογή.

Το αρχείο RecButton.jsx αποδίδει σε κώδικα το δομικό στοιχείο ενός κουμπιού και τη λειτουργικότητα γύρω από αυτό. Το κόκκινο κουμπί εγγραφής αφορά την καταγραφή των νέων τρεχόντων τιμών και σε συνδυασμό με το κουμπί “Stop” παρουσιάζεται στην οθόνη νέα επιλογή, εκείνης για το κατέβασμα των παραπάνω διατηρηθέντων τιμών σε αρχείο csv.



Εικόνα 14: Απεικόνιση Recording Button

Για χάρη εστίασης στη λειτουργία ενός τέτοιου στοιχείου δίνεται το αντίστοιχο αρχείο RecButton.jsx με απλουστεύσεις.

```
export const RecButton = () => {
  const [isRecording, setIsRecording] = useState(false);
  const recData = useDashStore((state) => state.recInfo.recData);
  const newData = useDashStore((state) =>
state.sensorData.lines[1].newData);
```

```
  useEffect(() => {
    if (isRecording) {
      setRecAppendedData(newData);
    }
  }, [newData]);

  const handleRecordToggle = () => {
    setIsRecording((prev) => !prev);
  };

  return (
    <>
      {isRecording || recData.length === 0 ? (
        <div onClick={handleRecordToggle}>
          {isRecording ? "Stop Recording" : "Start Recording"}
        </div>
      ) : (
        <div className="flex">
          <Button
            label="Clear"
            onClick={() => {
              setClearRecData();
            }}
          />
          <ExportCsv />
        </div>
      )}
    </>
  );
};
```

Εικόνα 15: Αρχείο RecButton.jsx

Το συστατικό RecButton χρησιμοποιεί μια μνήμη. Δηλώνει αρχικά τη μεταβλητή isRecording σε false. Αυτή η δήλωση γίνεται μέσω του useState η οποία είναι συχνή στη React. Με δηλώσεις των μεταβλητών μέσω useState το συστατικό χρειάζεται να αλλάξει τι εμφανίζει στην οθόνη ως αποτέλεσμα μιας διάδρασης με το χρήστη. Στη προκειμένη περίπτωση χρήσης, πατώντας το κόκκινο κουμπί που αντιπροσωπεύεται από το κείμενο “Start Recording”, ενεργοποιείται η συνάρτηση handleRecordToggle και αλλάζει η μεταβλητή από false ανάποδα σε true. Αυτή η αλλαγή ενεργοποιεί την επανεμφάνιση του στοιχείου στην οθόνη και πλέον η javascript είναι υπεύθυνη στο να αλλάξει την ροή του κώδικα ώστε να εμφανιστεί το “Stop” αντί του “Start”.

Το συστατικό RecButton χρησιμοποιεί και άλλη μνήμη. Εκτός από την ιδιωτική μνήμη εντός του συστατικού γίνεται χρήση ενός εκ των εργαλείων κώδικα για μνήμη κοινού πλαισίου. Έτσι στην εφαρμογή DashPL επιλέχθηκε για απλό εργαλείο το zustand store, μεταξύ άλλων εργαλείων όπως το πολυπλοκότερο Redux, ή το useContext από το ίδιο το περιβάλλον της React. Με αυτά τα εργαλεία ο προγραμματιστής διαχειρίζεται στο κοινό πλαίσιο ενός project πολλών συστατικών μια κοινή μνήμη και κυρίως εύκολα προσβάσιμη για όλα τα συστατικά. Συγκεκριμένα στο RecButton.jsx γίνεται χρήση των συναρτήσεων setRecAppendedData και setClearRecData που αφορούν τη μνήμη που δηλώνουν οι μεταβλητές newData και recData. Οι δύο μεταβλητές προέρχονται από τη διαμοιρασμένη κοινή μνήμη πλαισίου της εφαρμογής με ονομασίες state.sensorData… και state.recInfo. Αυτές εμφανίζεται και στα αρχεία Dashboard.jsx και ExportCSV.jsx όπου και χρησιμοποιούνται εσωτερικά αναλόγως τη λειτουργία των συστατικών αυτών.

Η εισροή νέων δεδομένων-σημείων στην οθόνη αλλά και στο παρασκήνιο για το “κατέβασμα” σε ομαδοποιημένο αρχείο csv από τις μετρήσεις του αισθητήρα αφορά το τμήμα κώδικα του RecButton.jsx εσωτερικά της συνάρτησης useEffect. Αυτή η συνάρτηση χρησιμοποιείται συχνά στη React όταν χρειάζεται η λειτουργία συγχρονισμού της ιστοσελίδας στην οθόνη με εξωτερικές υπηρεσίες/συστήματα.

Το συγκεκριμένο παράδειγμα μελετήθηκε σε συνδυασμό με τον “κορμό” της React. Αυτή χρησιμοποιήθηκε και μελετήθηκε για να υλοποιηθεί το μεγαλύτερο κομμάτι του λογισμικού του DashPL.

5 Κεφάλαιο

Επίλογος - Συμπεράσματα

Η επίσκεψη σε συγκεκριμένη καλλιέργεια μήλων στους πρόποδες του Βερμίου Εορδαίας οδήγησε στο συμπέρασμα πώς η χρήση της τεχνολογίας ανταποκρίνεται πλήρως στις αυξανόμενες απαιτήσεις και νέες προκλήσεις που αναδύονται στο κλάδο της γεωργίας. Τα χιλιάδες δέντρα φυτεμένα σε γραμμές στο χώρο των στρεμμάτων αυτών θα ήταν πρακτικά αδύνατον να παρακολουθηθούν χωρίς τη βοήθεια της τεχνολογίας. Οι σύγχρονοι μέθοδοι που βασίζονται στο διαδίκτυο των πραγμάτων καθιστά εφικτή τη διαχείριση μεγάλων εκτάσεων. Η διαπιστωμένη πλεονεκτική ισχύς της τεχνολογίας στον τομέα της γεωργίας (αλλά και αλλού) ανοίγει το δρόμο σε μια πιο ευφυή πρακτική και ενδεχομένως σε μια ευρύτερη εφαρμογή της τεχνολογίας.

Η πλατφόρμα που υλοποιήθηκε στη παρούσα διπλωματική εργασία είναι ευέλικτη και επεκτάσιμη. Συγκεκριμένα, οι επεκτάσεις δυνητικά μπορούν να καλύψουν την παρακολούθηση οποιωνδήποτε αισθητήρων σε οποιοδήποτε πλαίσιο πέραν της γεωργίας. Μπορεί η πλατφόρμα να μετεξελιχθεί ώστε ο χρήστης να επιλέγει από πληθώρα αισθητήρων. Μπορεί επίσης να δημιουργηθούν και άλλες αναφορές και διαγράμματα, ώστε να παίρνονται αποφάσεις καθολικότερα για ένα παρακολουθούμενο φαινόμενο. Στόχος ήταν να δημιουργηθεί ένα εργαλείο προσιτό και εύχρηστο, το οποίο καλύπτει τις βασικές ανάγκες για την παρακολούθηση και παρουσίαση δεδομένων από αισθητήρες καιρού. Ωστόσο, αυτό δύναται να αξιοποιηθεί και για οποιεσδήποτε άλλες μετρήσεις, όπως για παράδειγμα στα ιατρικά σήματα.

6 Βιβλιογραφία

[1] Lemelson-MIT, «Charles Walton | Lemelson,» [Ηλεκτρονικό]. Available: https://lemelson.mit.edu/resources/charles-walton.

[2] J. Fraden, Handbook of Modern Sensors, Springer, 2016.

[3] «Scalars and Vectors,» [Ηλεκτρονικό]. Available: https://www.grc.nasa.gov/www/k-12/airplane/vectors.html.

[4] P. Sethi και S. R. Sarangi, «Internet of Things: Architectures, Protocols, and Applications,» *Journal of Electrical and Computer Engineering,* p. 25, Γεννάρης 2017.

[5] Wikipedia, «Διαλειτουργικότητα».

[6] D. Sharma, S. Verma και K. Sharma, «Network Topologies in Wireless Sensor Networks: A review,» *International Journal of Electronics & Communication Technology,* pp. 93-97, 2013.

[7] B. S. Davie και L. L. Peterson, Computer Networks: A Systems Approach.

[8] G. Adamides, N. Kalatzis, A. Stylianou, N. Marianos, F. Chatzipapadopoulos, M. Giannakopoulou, G. Papadavid, V. Vassiliou και D. Neocleous, «Smart Farming Techniques for Climate Change Adaptation in Cyprus.,» *Atmosphere,* p. 557, Ιούνιος 2010.

[9] M. Damas, A. Prados, F. Gómez και G. Olivares, «HidroBus system: fieldbus for integrated management of extensive areas of irrigated land,» *Microprocessors and Microsystems,* pp. 177-184, Μάρτιος 2001.

[10] M. Rivera-Ferré, L. Pereira, T. Karpouzoglou, K. Nicholas και S. Onzere, «A Vision for Transdisciplinarity in Future Earth: Perspectives from

Young Researchers,» *Commentaries on Food Systems Research Priorities,* p. 249–260, Οκτώβριος 2013.

[11] J. Beddington, «The Perfect Storm: Food, energy, water security and climate change,» 2009.

[12] I. Mundy, «Declarative vs Imperative Programming...Or wrong ways I was thinking about React,» 2017. [Ηλεκτρονικό]. Available: https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2. [Πρόσβαση 2024].

[13] «Imperative vs Declarative Programming,» [Ηλεκτρονικό]. Available: https://ui.dev/imperative-vs-declarative-programming.

[14] K. Narkevicius, «UI is a function of state,» 2015. [Ηλεκτρονικό]. Available: https://www.kn8.lt/blog/ui-is-a-function-of-data/. [Πρόσβαση 2024].

Παράρτημα – Αποθετήριο Κώδικα

src > App.jsx

```
import { useState, useRef } from "react";
import "./output.css";
import { setDialogState, setZoomedOut, useDashStore } from "./store";
import { Header } from "./Components/Navbar/Header";
import { Dialog } from "./Components/Dialog";
import { Button } from "./Components/Button";
import { RecButton } from "./Components/Navbar/RecButton";
import { ExportAsPNG } from "./Components/ExportAsPNG";
import { DashBoard } from "./Components/Dashboard";
import { useForm, FormProvider } from "react-hook-form";
import RabbitMQConsumer from "./Components/RabbitMQConsumer";

// now is a string that represents the current time in the format HH:MM:SS
const now = new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-
digit', second: '2-digit' });

function App() {
  const methods = useForm();
  console.log(methods.formState);
  const zoomedIn = useDashStore((state) => state.zoom.zoomedIn);
  const [title, setTitle] = useState("humidity");
  const [description, setDescription] = useState("Description");
  const dashboardRef = useRef(null);

  return (
    <div className="h-screen relative flex flex-col justify-between ">
      <FormProvider {...methods}>
        <Dialog setTitle={setTitle} setDescription={setDescription} />
      </FormProvider>
      <div>
        <Header />
        <div className=" h-52 z-50" ref={dashboardRef}>
          <div className=" py-10 px-20 ">
            <div>
              <p className="text-slate-500">{now.slice(0,5)+ `.`}
                <span className="text-sm">{now.slice(6,8) + `"`}</span>
              </p>
            </div>
            <RabbitMQConsumer/>
            <div className="flex justify-between">
              <h1 className="tracking-tight text-3xl font-semibold"
title={description}>
                {title}
              </h1>
              {zoomedIn && (
                <Button
                  label="Zoom Out"
```

```
                onClick={() => {
                  setZoomedOut(true);
                }}
              />
            )}
            <div className="flex">
              <RecButton />
              <ExportAsPNG elementRef={dashboardRef} fileName="dashboard" />
              <Button
                label="Edit"
                onClick={() => {
                  setDialogState(true);
                }}
              />
            </div>
          </div>
          <DashBoard />
        </div>
      </div>
    </div>
    <PageWaveFooter>{…}
    </PageWaveFooter>
  </div>
 );
}
```

src > store.js

```
import { create } from "zustand";

export const useDashStore = create(() => ({
  sensorData: {
    lines: [
      {
        name: 'Line 1',
        index: 0,
        data: [],
        showLine: true,
        newData: null,
      },
      {
        name: 'Line 2',
        index: 1,
        data: [],
        showLine: true,
        newData: null,
      },
      {
        name: 'Line 3',
        index: 2,
        data: [],
        showLine: true,
        newData: null,
```

```
      },
    ],
    lineCount: 3,
  },
  dialogIsOpen: false,
  recInfo: {
    isRecording: false,
    recData: [],
  },
  zoom: {
    zoomedIn: false,
    zoomedOut: true,
    left: "dataMin",
    right: "dataMax",
    refAreaLeft: 0,
    refAreaRight: 0,
    top: "dataMax+1",
    bottom: "dataMin-1",
    top2: "dataMax+20",
    bottom2: "dataMin-20",
    animation: true,
  },
}));

export const updateLineData = (newdata, lineIndex) => {
  return useDashStore.setState((state) => {
    const updatedLine = [...state.sensorData.lines];
    updatedLine[lineIndex] = { ...updatedLine[lineIndex], newData: newdata };
    return { sensorData: { ...state.sensorData, lines: updatedLine } };
  });
};

export const setAppendedData = (newdata, lineIndex) => {
  return useDashStore.setState((state) => {
    const updatedLine = [...state.sensorData.lines];
    updatedLine[lineIndex].data = [...updatedLine[lineIndex].data, newdata ];
    return { sensorData: { ...state.sensorData, lines: updatedLine } };
  });
};

export const addLine = () => {
  return useDashStore.setState((state) => ({
    sensorData: {
      ...state.sensorData,
      lines: [
        ...state.sensorData.lines,
        {
          name: `Line ${state.sensorData.lineCount + 1}`,
          index: state.sensorData.lineCount,
          data: [],
          newData: null,
          showLine: true,
        },
```

```
      ],
      lineCount: state.sensorData.lineCount + 1,
    },
  }));
};

export const setRecAppendedData = (newdata) => {
  return useDashStore.setState((state) => ({
    recInfo: { ...state.recInfo, recData: [...state.recInfo.recData, newdata]
},
  }));
};

export const setClearRecData = () => {
  return useDashStore.setState((state) => ({
    recInfo: { ...state.recInfo, recData: [] },
  }));
};

export const setZoomedIn = (zoomedIn) => {
  return useDashStore.setState(() => ({ zoom: { zoomedIn: zoomedIn } }));
};

export const setZoomedOut = (zoomedOut) => {
  return useDashStore.setState(() => ({ zoom: { zoomedOut: zoomedOut } }));
};

export const setDialogState = (isOpen) => {
  return useDashStore.setState(() => ({ dialogIsOpen: isOpen }));
};

export const setRecordButtonState = (isRecording) =>
  useDashStore.setState((state) => ({
    recInfo: { ...state.recInfo, isRecording: isRecording },
  }));

export const setRefAreaLeft = (refAreaLeft) =>
  useDashStore.setState((state) => ({
    zoom: { ...state.zoom, refAreaLeft: refAreaLeft },
  }));

export const setRefAreaRight = (refAreaRight) =>
  useDashStore.setState((state) => ({
    zoom: { ...state.zoom, refAreaRight: refAreaRight },
  }));

export const setLeftRight = (left, right) =>
  useDashStore.setState((state) => ({
    zoom: { ...state.zoom, left: left, right: right },
  }));
```

Dialoge.jsx

```
import { useDashStore, setDialogState } from "../../store";
import {
  Transition,
  TransitionChild,
  Dialog as DialogHeadlessUI,
  DialogPanel,
} from "@headlessui/react";
import EditForm from "../EditForm";

export const Dialog = ({ setTitle, setDescription }) => {
  const isOpen = useDashStore((state) => state.dialogIsOpen);
  return (
    <Transition appear show={isOpen}>
      <DialogHeadlessUI
        as="div"
        className=" z-50 focus:outline-none "
        onClose={() => setDialogState(false)}
      >
        <div className="absolute top-1/2 -translate-y-1/2 z-10 w-screen">
          <div className="flex min-h-full items-center justify-center p-4">
            <TransitionChild
              enter="ease-out duration-300"
              enterFrom="opacity-0 transform-[scale(95%)]"
              enterTo="opacity-100 transform-[scale(100%)]"
              leave="ease-in duration-200"
              leaveFrom="opacity-100 transform-[scale(100%)]"
              leaveTo="opacity-0 transform-[scale(95%)]"
            >
              <DialogPanel style={{backgroundColor: "#bbf2"}} className="w-full
max-w-md rounded-xl p-6 backdrop-blur-2xl">
                <EditForm setTitle={setTitle} setDescription={setDescription}/>

              </DialogPanel>
            </TransitionChild>
          </div>
        </div>
      </DialogHeadlessUI>
    </Transition>
  );
};
```

EditForm.jsx

```
import { useFormContext } from 'react-hook-form';
import { Button } from "../Button";
import { setDialogState } from "../../store";;

function EditForm({ setTitle, setDescription}) {
  const { register, handleSubmit, formState: { errors }, control } =
useFormContext()
```

```
  const onSubmit = (data) => {
    console.log(data);
    setTitle(data.title);
    setDescription(data.description);
    setDialogState(false);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)} className="select-none space-y-4">
      <div className="flex flex-col">
        <label className="mb-2 font-semibold">Title</label>
        <input
          {...register("title", { required: "Title is required" })}
          className="p-2 border border-gray-300 rounded-md focus:outline-none
focus:ring-2 focus:ring-blue-500"
        />
        {errors.title && <span className="text-red-
500">{errors.title.message}</span>}
      </div>
      <div className="flex flex-col">
        <label className="mb-2 font-semibold">Description</label>
        <input
          {...register("description")} // Fixed typo here
          className="p-2 border border-gray-300 rounded-md focus:outline-none
focus:ring-2 focus:ring-blue-500"
        />
      </div>
      <div className="flex justify-end">
        <Button type="submit" label="Save" className="px-4 py-2 bg-blue-500
text-white rounded-md hover:bg-blue-600" />
      </div>
    </form>
  );
}

export default EditForm;
```

Button.jsx

```
import { useState } from 'react';

export const Button = ({label, onClick, type}) => {
    return (
        <button type={type} onClick={onClick} className='px-2 py-1 text-black
rounded-xl border-1 select-none' style={{ fontSize: '80%', borderColor: 'gold',
borderWidth: '0.5px' }}>
            {label}
        </button>
    );
}
```

RecButton.jsx

```
import React, { useState, useEffect } from "react";
import { setRecAppendedData, setClearRecData, useDashStore } from
"../../store";
import { Button } from "../Button";
import { ExportCsv } from "../ExportCsv";

export const RecButton = () => {
  const [isRecording, setIsRecording] = useState(false);
  const recData = useDashStore((state) => state.recInfo.recData);
  const newData = useDashStore((state) => state.sensorData.lines[1].newData);

  useEffect(() => {
    if (isRecording) {
      setRecAppendedData(newData);
    }
  }, [newData]);

  const handleRecordToggle = () => {
    setIsRecording((prev) => !prev);
  };

  return (
    <>
      {isRecording || recData.length === 0 ? (
        <div onClick={handleRecordToggle}>
          {isRecording ? "Stop Recording" : "Start Recording"}
        </div>
      ) : (
        <div className="flex">
          <Button
            label="Clear"
            onClick={() => {
              setClearRecData();
            }}
          />
          <ExportCsv />
        </div>
      )}
    </>
  );
};
```

ExportCsv.jsx

```
import React from "react";
import { Button } from "./Button";
import { useJsonToCsv } from "react-json-csv";
import { useDashStore } from "../store";

export const ExportCSV = () => {
  const { saveAsCsv } = useJsonToCsv();
  const katagegrammena = useDashStore((state) => state.recInfo.recData);
```

```
  const filename = "valuesOfSensor";
  const fields = { time: "Timestamp", hum: "Humidity", temp: "Temperature" };

  return (
    <Button label="Download Csv" onClick={() => saveAsCsv({
      data: katagegrammena.map((d) => {
        return {
          // stylish way to replace commas with "of" in the csv file
          time: d.uv.toString().replace(/,/g, ' of'),
          hum: d.pv,
          temp: d.val,
        };
      }),
      fields: fields,
      filename: filename,
    })}>AL</Button>
  );
};
```

ExportAsPNG.jsx

```
import React from "react";
import { Button } from "./Button";
import domtoimage from "dom-to-image";
import FileSaver from "file-saver";

export const ExportAsPNG = ({ elementRef, fileName = "exported-image" }) => {
  const handleExport = () => {
    if (elementRef?.current) {
      domtoimage.toBlob(elementRef.current)
        .then((blob) => {
          FileSaver.saveAs(blob, `${fileName}.png`);
        })
        .catch((error) => {
          console.error("Error exporting PNG:", error);
        });
    } else {
      console.error("No elementRef provided or the ref is not attached.");
    }
  };

  return (
    <Button
      label="Image"
      onClick={handleExport}
      className="px-4 py-2 bg-blue-500 text-white font-semibold rounded-lg
hover:bg-blue-600"
    >
      Save as PNG
    </Button>
  );
};
```

RabbitMQConsumer.jsx

```
import { useEffect, useState } from 'react';
import { useDashStore } from "../../store";
import { updateLineData, setAppendedData } from "../../store";
import mqtt from 'mqtt';

function RabbitMQConsumer() { {…}
  const [newData, setNewData] = useState({});
  const sensorAllData = useDashStore((state) =>
state.sensorData.lines[1].data);

  useEffect(() => {
    if (Object.keys(newData).length > 0) {
      updateLineData(newData, 1);
      setAppendedData(newData, 1);
    }
  }, [newData, updateLineData, setAppendedData]); {…}  {…} //useEffect with
[sensorAllData.length, topic]);

  return (
    <></>
  );
}

export default RabbitMQConsumer;
```

Dashboard.jsx

```
import { useEffect, useRef, useState } from "react";
import { useDashStore } from "../../store";
import { updateLineData, setAppendedData } from "../../store";
import { setZoomedIn } from "../../store";
import { setLeftRight, setRefAreaLeft, setRefAreaRight } from "../../store";
import {
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  ResponsiveContainer,
  ReferenceArea,
  LineChart,
  Line,
} from "recharts";
import { fromUnixTime, getMinutes, getSeconds } from 'date-fns';
import CustomTooltip from "../CustomTooltip";

export const DashBoard = () => {
  const sensorAllData = useDashStore((state) =>
state.sensorData.lines[1].data);
  const [data, setData] = useState([]);
  const [mouseUped, setMouseUped] = useState(false);
  const {
```

```
    zoomedIn,
    zoomedOut,
    refAreaLeft,
    refAreaRight,
    top,
    bottom,
    top2,
    bottom2,
    left,
    right,
  } = useDashStore((state) => state.zoom);

  const prevDeps = useRef([false, 'dataMin', 'dataMax']);


  useEffect(() => {
    if (zoomedOut && !zoomedIn) {
      setData(sensorAllData);
      setLeftRight('dataMin', 'dataMax');
      //resetZoom();
    }
  }, [zoomedOut, sensorAllData, zoomedIn]);

  useEffect(() => {
    function zoom() {
      if (refAreaLeft < refAreaRight) {
        setZoomedIn(true);
        setData(
          sensorAllData.filter(
            (entry) => entry.name >= refAreaLeft && entry.name <= refAreaRight
          )
        );
        setLeftRight(refAreaLeft, refAreaRight);
      }
    }
    if (
      prevDeps.current[0] !== mouseUped &&
      prevDeps.current[1] !== refAreaLeft &&
      prevDeps.current[2] !== refAreaRight
    ) {
      if (mouseUped) {
        zoom();
        setMouseUped(false);
        setRefAreaLeft('');
        setRefAreaRight('');
      }
      prevDeps.current = [false, refAreaLeft, refAreaRight];
    }
  }, [mouseUped, refAreaLeft, refAreaRight]);

  const handleMouseUp = () => {
    if (refAreaLeft === refAreaRight || refAreaRight === '') {
      setRefAreaLeft('');
```

```
      setRefAreaRight('');
      return;
    }
    setMouseUped(true);
  };

  return (
    <div className="z-50 select-none mt-4 rounded-lg bg-white py-6 shadow-2xl
border border-black sm:px-6 h-[60vh] border-dashed">
      <ResponsiveContainer width="100%" height="100%">
        <LineChart
          onAnimationStart={() => {}}
          animationDuration={0}
          width={500}
          height={300}
          data={data}
          onMouseDown={(e) => setRefAreaLeft(e.activeLabel)}
          onMouseMove={(e) => refAreaLeft && setRefAreaRight(e.activeLabel)}
          onMouseUp={handleMouseUp}
          margin={{
            top: 5,
            right: 30,
            left: 20,
            bottom: 5,
          }}
        >
          <defs>
            <linearGradient id="colorUv" x1="0" y1="0" x2="0" y2="1">
              <stop offset="5%" stopColor="#f3e00a" stopOpacity={0.8} />
              <stop offset="95%" stopColor="#f3e00a" stopOpacity={0} />
            </linearGradient>
            <linearGradient id="colorPv" x1="0" y1="0" x2="0" y2="1">
              <stop offset="5%" stopColor="#01cbdd" stopOpacity={0.8} />
              <stop offset="95%" stopColor="#01cbdd" stopOpacity={0} />
            </linearGradient>
          </defs>
          <XAxis
            className="select-none"
            allowDataOverflow
            dataKey="name"
            domain={[left, right]}
            type="category"
            xAxisId="0"
          />
          <YAxis
            className="select-none"
            allowDataOverflow
            domain={[0, 100]}
            orientation="right"
            yAxisId="5"
            type="number"
            label={{ value: 'Humidity (%)', angle: -90, position: 'insideLeft',
dx: +10, dy: -60 }}
```

```
          />
          <YAxis
            className="select-none"
            allowDataOverflow
            domain={[10, 31]}
            yAxisId="4"
            dataKey="uv"
            type="number"
            label={{ value: 'Temprature (celsius)', angle: -90, position:
'insideLeft', dx: -10, dy: 30 }}
          />
          <CartesianGrid strokeDasharray="3 3" />
          <Tooltip
            content={<CustomTooltip />}
          />
          <Line
            name= "Temprature"
            isAnimationActive={false}
            type="linear"
            dataKey="val"
            stroke="#e57373"
            fillOpacity={1}
            fill="url(#colorUv)"
            dot={{
              strokeWidth: 0.2,
              stroke: '#555',
              r: 4.3 - 3.1 * (1 - Math.exp(-0.05 * data.length)),
            }}
            xAxisId="0"
            yAxisId="4"
          />
          <Line
            name="Humidity"
            animationDuration={0}
            type="linear"
            dataKey="pv"
            stroke="#8884d8"
            fillOpacity={1}
            fill="url(#colorPv)"
            dot={{
              strokeWidth: 0.2,
              stroke: '#555',
              r: 4.3 - 3.1 * (1 - Math.exp(-0.05 * data.length)),
            }}
            xAxisId="0"
            yAxisId="5"
          />
          {refAreaLeft && refAreaRight > refAreaLeft ? (
            <ReferenceArea
              x1={refAreaLeft}
              x2={refAreaRight}
              strokeOpacity={0.3}
              yAxisId="5"
```

```
            />
          ) : null}
        </LineChart>
      </ResponsiveContainer>
    </div>
  );
};
```

CustomTooltip.jsx

```
import React, { memo } from 'react';
import { fromUnixTime, getHours, getMinutes, getSeconds } from 'date-fns';

const CustomTooltip = memo(({ active, payload }) => {
  let timeShow = '';
  let bigSeconds = '';
  let hoursAndMinutes = '';
  if (active && payload && payload.length) {
    const epoch = payload[0]?.payload.name;

    if (!isNaN(epoch)) {
      const utcDate = fromUnixTime(epoch/1000);
      const hours = getHours(utcDate)
      const minutes = getMinutes(utcDate);
      const seconds = getSeconds(utcDate);
      hoursAndMinutes = `${hours.toString().padStart(2,
'0')}:${minutes.toString().padStart(2, '0')}`
      bigSeconds = `${seconds.toString().padStart(2, '0')}`
      timeShow = `${hours.toString().padStart(2,
'0')}:${minutes.toString().padStart(2, '0')}.${seconds.toString().padStart(2,
'0')}"`;
    } else {
      timeShow = epoch;
    }


    return (
      <div className="bg-white border border-gray-200 rounded-md p-3 shadow-md
z-20">
        <p className="text-gray-800 mb-2 flex items-center">
          <span className='text-sm'>{hoursAndMinutes}</span>
          <span className="mx-1">.</span>
          <span className="font-semibold ml-1">{bigSeconds}</span>
          <span className="font-semibold">"</span>
        </p>
        <p className="text-blue-500 text-sm flex items-center">
          Humidity: <span className="ml-1 font-
medium">{payload[1].value}%</span>
        </p>
        <p className="text-orange-500 text-sm flex items-center">
          Temprature: <span className="ml-1 font-
medium">{payload[0].value}%</span>
        </p>
```

```
      </div>
    );
  }
  return null;
});

export default CustomTooltip;
```