ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ
ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

Διπλωματική Εργασία

Συλλογή, επεξεργασία, ανάλυση και διαχείριση δεδομένων σε σύστημα Ψύξης/Θέρμανσης

Συγγραφέας:
ΓΕΩΡΓΙΟΣ ΔΗΜΑΣ
Αριθμός Μητρώου: 714222017029

Επιβλέπων:
Δρ. ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ
Καθηγητής

ΑΙΓΑΛΕΩ, Σεπτέμβριος 2024

UNIVERSITY OF WEST ATTICA
SCHOOL OF ENGINEERING
DEPARTMENT OF INDUSTRIAL DESIGN
AND PRODUCTION ENGINEERING

Diploma Thesis

Data collection, processing, analysis and management in a cooling/heating system

Author:
GEORGIOS DIMAS
Registration Number: 714222017029

Supervisor:
Dr. Evangelos Pallis
Professor

Egaleo, September 2024

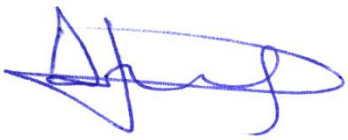
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γεώργιος Δήμας του Ιωάννη με αριθμό μητρώου 714222017029 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών
Γεώργιος Δήμας



Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	**ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ**	ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ Digitally signed by ΕΥΑΓΓΕΛΟΣ ΠΑΛΛΗΣ Date: 2024.10.11 16:48:09 +03'00'
2	**ΕΛΕΝΗ ΑΙΚΑΤΕΡΙΝΗ ΛΕΛΙΓΚΟΥ**	ELENI AIKATERINI LELIGKOU Digitally signed by ELENI AIKATERINI LELIGKOU Date: 2024.10.11 16:39:32 +03'00'
3	**ΧΡΗΣΤΟΣ ΔΡΟΣΟΣ**	Digitally signed by CHRISTOS DROSOS Date: 2024.10.11 14:55:03 +03'00'

Περίληψη

Στην παρούσα διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε ένα σύστημα συλλογής, επεξεργασίας, ανάλυσης και διαχείρισης δεδομένων κλιματισμού, κάνοντας χρήση του «Διαδικτύου των Πραγμάτων» (Internet of Things – IoT), υπολογιστικών νεφών και μηχανισμών τεχνητής νοημοσύνης, με στόχο την πρόβλεψη δυσλειτουργιών, την αποφυγή βλαβών και την έγκαιρη αντικατάσταση ελλαττωματικών μηχανισμών σε συστήματα ψύξης/θέρμανσης.

Πιο συγκεκριμένα, η συλλογή των δεδομένων πραγματοποιήθηκε μέσω αισθητήρων θερμοκρασίας και πίεσης σε επιλεγμένα σημεία ενός κλειστού κυκλώματος θέρμανσης, ενώ για την επεξεργασία τους σε τοπικό επίπεδο (τελική συσκευή – end device) χρησιμοποιήθηκε ο μικροελεγκτής ESP-32. Για την περαιτέρω επεξεργασία των δεδομένων, έγινε χρήση νεφοϋπολογιστικής υποδομής, η οποία επικοινωνούσε με την τελική συσκευή (ESP32) μέσω της τεχνολογίας Wi-Fi, κάνοντας χρήση του πρωτοκόλλου ανταλλαγής μηνυμάτων “MQTT”. Για την αποθήκευση των δεδομένων στο υπολογιστικό νέφος, σχεδιάστηκε και υλοποιήθηκε μια μη-σχεσιακή βάση δεδομένων - χρονοσειρών (NoSQL), ενώ για την περαιτέρω επεξεργασία και η ανάλυση τους έγινε χρήση αλγορίθμου τεχνητής νοημοσύνης και «Νευρωνικών Δικτύων Μακράς Βραχυπρόθεσμης Μνήμης» (LSTM Networks), με στόχο την έγκαιρη πρόβλεψη δυσλειτουργιών στο σύστημα ψύξης/θέρμανσης.

Στα πλαίσια επικύρωσης της εγκυρότητας της προτεινόμενης αρχιτεκτονικής και της αξιολόγησης των επιδόσεων του συστήματος, μελετήθηκε και υλοποιήθηκε μοντέλο προσομοίωσης κτηριακού συστήματος θέρμανσης «δωμάτιο φούρνου», στο οποίο χρησιμοποιήθηκαν κατάλληλα αισθητήρια, τόσο βιομηχανικών όσο και οικιακών προδιαγραφών. Η καταγραφή της συμπεριφοράς του μοντέλου σε συνδυασμό με την εκπαίδευση του αλγορίθμου τεχνητής νοημοσύνης, επέτρεψε την βελτιστοποίηση του μηχανισμού πρόβλεψης ανωμαλιών και δυσλειτουργιών, συγκρίνοντας τις τρέχουσες τιμές με εκείνες προηγούμενων (γνωστών) καταστάσεων υγειούς λειτουργίας. Οι δοκιμές και τα πειραματικά αποτελέσματα ανέδειξαν το εύρος του πεδίου εφαρμογής του Διαδικτύου των Πραγμάτων και της τεχνητής νοημοσύνης σε συστήματα ψύξης/θέρμανσης, και επιβεβαίωσαν την αποτελεσματικότητα της προτεινόμενης προγνωστικής μεθόδου, αναδεικνύοντας την αξία της στην αποδοτική λειτουργία και την έγκαιρη διάγνωση προβλημάτων. Τέλος από τα συμπεράσματα της έρευνας και των δοκιμών, η διπλωματική προσδιορίζει πεδία έρευνας και ανάπτυξης, με στόχο την βέλτιστη λειτουργία της προτεινόμενης προσέγγισης.

Λέξεις – κλειδιά

Διαδίκτυο των Πραγμάτων, Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Νευρωνικά Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης, Συστήματα Ψύξης- Θέρμανσης, Θέρμανση Αερισμός Κλιματισμός Ψύξη, Προγνωστική συντήρηση, Συντήρηση ηλεκτρομηχανολιγών συστημάτων, Μικροελεγκτής ESP-32,

Abstract

The aim of this diploma thesis was to design and implement a system for data collection, processing, analysis and management for HVAC platforms, by utilizing Internet of Things (IoT) technologies, cloud computing infrastructures and artificial intelligence mechanisms, towards predicting potential malfunctions, avoiding down-time periods, besides proactively replacing defective parts/equipment in Heating/Cooling systems.

More specifically, the collection of the data was accomplished by exploiting temperature and pressure sensors in specific points of a closed loop heating system, while processing of the collected data was performed locally at the end device level by utilizing the ESP-32 microcontroller. For the next level of data processing, cloud computing utilities were taken into account by connecting the end device (ESP-32) to the internet via Wi-Fi, the message transmission protocol "MQTT" were used. For the storage of the data in the cloud computing infrastructure, it was design and implemented a non-relational - time based (NoSQL) data base engine. As a next level of processing and analysis, Artificial Intelligence took place by utilizing "Long Short Term Memory Neural Networks" (LSTM Networks), the point was to promptly predict malfunctions in a heating/cooling system.

In the scope of confirming the validity of the recommended architecture and for evaluating the effective performance of the designed system, it was necessary to develop and construct a simulation model of a heating system for a building, "a room oven". In the design model, proper sensors were used based on industrial and residential infrastructure standards. The logging of the behavior from the simulation model in combination with the training of the built artificial intelligence model, allowed the optimization of the machine failure prediction mechanism by comparing the live measured data to previous measurements of know healthy operations. The tests and the experiment results point out the big advantage of IoT and AI in the Heating/Cooling Systems (HVAC). Furthermore, the effectiveness of the recommended predictive maintenance approach was came out highly confirmed due to the important role of the accurate and quick prediction of malfunctions. Finally, as a result of the studying and the experiments, the conclusion of this thesis diploma pointed out the next steps of development, which aim to a more optimized performance of the predictive maintenance approach.

Keywords

Internet of Things, Artificial Intelligence, Machine Learning, Long Short-Term Memory Neural Networks (LSTM), Heating and Cooling Systems, HVAC, Predictive Maintenance, Maintenance of Mechanical and Electrical Systems, Microcontroller ESP-32

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή μου, Δρ. Ευάγγελο Πάλλη, για την εμπιστοσύνη, την συνεχή υποστήριξη και την καθοδήγηση που έδειξε κατά την διάρκεια της διεκπεραίωση της διπλωματικής μου εργασίας.

Επιπλέον, θα ήθελα να ευχαριστήσω από καρδίας, τους καθηγητές του τμήματος «Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής» του πανεπιστημίου, την οικογένεια μου, τους φίλους μου και όσους άλλους στάθηκαν δίπλα μου, στα χρόνια των προπτυχιακών μου σπουδών αλλά και στο διάστημα της εκπόνησης της διπλωματικής μου εργασίας, για την απαράμιλλή στήριξη τους όταν το βεβαρυμμένο μου πρόγραμμα δεν μας επέτρεπε να περνάμε τον επιθυμητό χρόνο μαζί.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Αλφαβητικό Ευρετήριο

IEEE: The Institute for Electrical and Electronics Engineers

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια, ο τομέας της συντήρησης βιομηχανικών υποδομών παρουσιάζει ιδιαίτερο ενδιαφέρων και πολλές προκλήσεις, ιδιαίτερα όταν αυτές αφορούν στη σωστή και αδιάληπτη λειτουργία, τόσο των βασικών όσο και των βοηθητικών μονάδων παραγωγής. Λύση προς αυτή την κατεύθυνση μπορούν να αποτελέσουν οι τεχνολογίες επικοινωνιών και πληροφορικής, οι οποίες – σε αντίθεση με τις κλασσικές/τυπικές μεθόδους συντήρησης – μπορούν να προσφέρουν δυνατότητες προγνωστικού ελέγχου και προληπτικής συντήρησης των επί μέρους μηχανισμών, συμβάλλοντας στην ελαχιστοποίηση του κόστους λειτουργίας και στην βελτιστοποίηση της παραγωγικής διαδικασίας. Μεταξύ αυτώ των τεχνολογιών, πρωταγωνιστικό ρόλο έχει το Διαδίκτυο των Πραγμάτων και η ενοποίησή του με τις τεχνολογίες των υπολογιστικών νεφών και της τεχνικής νοημοσύνης. Η συμπεριφορά κάθε μηχανήματος, όπως και ενός συστήματος (π.χ. ψύξης ή θέρμανσης) μπορεί να μελετηθεί με μαθηματική ακρίβεια και είναι δυνατόν να δώσει χρήσιμα συμπεράσματα για την “υγεία” του εξοπλισμού. Οι μηχανικοί είναι σε θέση να έχουν καλύτερη εικόνα για την κατάσταση της υγείας του εξοπλισμού και μπορούν να προγραμματίσουν με περισσότερη ακρίβεια και αποτελεσματικότητα τις συντηρήσεις που είναι απαραίτητες να γίνουν. Μέσω της χρήσης κατάλληλων μηχανισμών, συλλέγονται δεδομένα από το πεδίο, αποστέλλονται στο διαδίκτυο, αποθηκεύονται στο υπολογιστικό σύννεφο και αναλύονται με την βοήθεια της μηχανικής μάθησης για την εκλογή χρήσιμων συμπερασμάτων για τον χρήστη και τον μηχανικό.

Αντικείμενο της διπλωματικής εργασίας

Η στοχευμένη υλοποίηση μεθόδων προγνωστικής συντήρησης σε συστήματα ψύξης και θέρμανσης μπορεί να προσφέρει σημαντικά οφέλη στην ανθρωπότητα. Η συνέλιξη του μικροελεγκτή ESP-32 της Espressif, των δομών ενός υπολογιστικού νέφους και της τεχνητής νοημοσύνης καλείται να αποδείξει κατά πόσο είναι εφικτό να προσφερθούν χρήσιμες λύσεις σε ευρεία κλίμακα με χαμηλό κόστος.

Σκοπός και στόχοι

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι να εξοικειωθεί ο φοιτητής με τις νέες τάσεις της τεχνολογίας, να μελετήσει τους τρόπους διασύνδεσης, να έρθει σε επαφή με τον μικροελεγκτή ESP-32, να εμβαθύνει στο περιβάλλον προγραμματισμού του και να ενσωματώσει την τεχνητή νοημοσύνη προσφέροντας μια ολοκληρωμένη λύση συλλογής, επεξεργασίας, ανάλυσης και διαχείρισης δεδομένων σε συστήματα ψύξης και θέρμανσης. Στόχος είναι η αξιολόγηση των τεχνολογιών που υπάρχουν ανά στάδιο υλοποίησης του πειραματικού μέρους της διπλωματικής εργασίας, η επιλογή των πλέον κατάλληλων μηχανισμών και ο εντοπισμός των λεπτομερειών σημείων που θα οδηγήσουν στην δημιουργία ενός αυτόνομου και συνάμα καινοτόμου συστήματος ελέγχου ηλεκτρομηχανολογικών συστημάτων, συγκεκριμένα συστημάτων ψύξης θέρμανσης.

1 ΚΕΦΑΛΑΙΟ 1ο : Διαδίκτυο των Πραγμάτων - Internet of Things

1.1 Εισαγωγή στο Διαδίκτυο των Πραγμάτων

Το Διαδίκτυο των Πραγμάτων ή στα αγγλικά Internet of Things (IoT) είναι ένα δίκτυο υλικών συσκευών, οχημάτων, οικιακών συσκευών και οποιονδήποτε άλλων αντικειμένων που φέρουν μαζί τους ηλεκτρονικά, λογισμικό, αισθητήρες, ενεργοποιητές και μια διάταξη δικτυακής σύνδεσης η οποία επιτρέπει στις συσκευές αυτές να συλλέγουν και να ανταλλάσσουν δεδομένα μεταξύ τους, με τους υπολογιστές ή και με τους ανθρώπους. [1]

Το Διαδίκτυο των Πραγμάτων χρησιμοποιώντας την υπάρχουσα δικτυακή υποδομή του εύκολα προσβάσιμου διαδικτύου, επιτρέπει σε πράγματα και αντικείμενα να αισθάνονται μεγέθη και καταστάσεις σε ένα απομακρυσμένο περιβάλλον και να αλληλεπιδρούν με αυτό εκτελώντας αντίστοιχες ενέργειες ή σενάρια. Η προσέγγιση αυτή, δημιουργεί ευκαιρίες για πιο άμεσες λύσεις ολοκλήρωσης του φυσικού κόσμου με τον κόσμο των υπολογιστών και της οργανωμένης πληροφορίας, φέροντας σαν αποτέλεσμα βελτιωμένη αποτελεσματικότητα, ακρίβεια χειρισμών και οικονομικά οφέλη, ενώ όλα αυτά πάντα βρίσκονται σε συνδυασμό με την μείωση της αλληλεπίδρασης της ανάγκης παρέμβασης του ανθρώπου.[1]

Η Ψηφιοποίηση σήμερα, έχει καθιερώσει το «έξυπνο» (smart) ως τον βασικό πυρήνα της σημερινής τεχνολογικής ανάπτυξης. Στις μέρες μας το Διαδίκτυο των Πραγμάτων λόγω της ευκαιρίας για καινοτομία και κοινωνική πρόοδο θεωρείτε ως ο κορμός γύρω από την 4η Βιομηχανική Επανάσταση. Το IoT είναι μια νέα υπόσχεση τεχνολογία βασισμένη στο Διαδίκτυο που έρχεται να ενώσει μεταξύ τους υλικές συσκευές οικιακές συσκευές, βιομηχανικό εξοπλισμό ή άλλα «πράγματα». Η κεντρική ιδέα πίσω από αυτή την εφεύρεση είναι ότι μέσα από την αξιοποίηση των κατάλληλων δικτυώσεων και αισθητηρίων, αυτές οι συσκευές θα μπορούσαν να προσφέρουν στους ανθρώπους ποικίλες υπηρεσίες και πολύτιμα δεδομένα. [2]

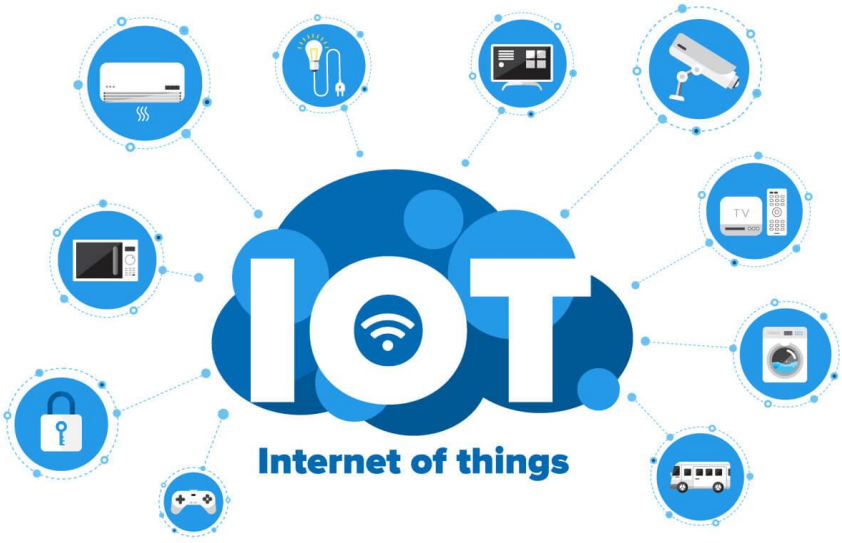
Σύμφωνα με το STATISTA, το 2020 υπήρχαν 15,1 δισεκατομμύρια συσκευές IoT σε παγκόσμια κλίμακα. Οι προβλέψεις αναφέρουν ότι μέχρι το 2030 θα υπάρχουν 29 δισεκατομμύρια συσκευές. Το 2030 μάλιστα μόνο οι πολίτες της Κίνα θα χρησιμοποιούν περίπου 8 δισεκατομμύρια καταναλωτικές συσκευές IoT. [3]

Η ιδέα ενός δικτύου έξυπνων συσκευών παρουσιάστηκε αρχικά το 1982, όταν ένα ανανεωμένο μηχάνημα αναψυκτικών τόπου κόλα στο Πανεπιστήμιο Carnegie Mellon έγινε το πρώτο μηχάνημα με σύνδεση στο διαδίκτυο. Μπορούσε να αναφέρει πληροφορίες σχετικά με τον κατάλογό του και αν τα γεμισμένα ποτά ήταν κρύα ή όχι. Ωστόσο, δεκαεφτά χρόνια μετά και συγκεκριμένα το 1999 ο Βρετανός επιχειρηματίας Kevin Ashton, εκτελεστικός διευθυντής του Auto-ID Center, είναι αυτός που χρεώνεται την καθιέρωση του όρου "Διαδίκτυο των Πραγμάτων". Στις μέρες μας, με την ανάπτυξη των ασύρματων μηχανισμών μετάδοσης και των μικρής κλίμακας, λιγότερο δυνητικών "γκάτζετ", η επικοινωνία μεταξύ παρεμφερών αντικειμένων ή η αποστολή οδηγιών σε οικιακές συσκευές δεν τίθεται καν προς αμφισβήτηση. Ωστόσο, στην πραγματικότητα, το Διαδίκτυο των πραγμάτων είναι ένα μέσο επικοινωνίας μεταξύ ανθρώπων και μηχανών.[4]

1.2 Εφαρμογές Διαδικτύου των Πραγμάτων

Αξιοποιώντας όλες τις δυνατότητες της επιστήμης των υπολογιστών οι εφαρμογές του διαδικτύου των πραγμάτων μπορούν να καλύψουν μια πληθώρα εφαρμογών όπως έξυπνες πόλεις, έξυπνη γεωργία, έξυπνες μεταφορές και έξυπνη συγκοινωνία, παροχή ασφάλειας και κλήση βοήθειας έκτακτης ανάγκης, οικιακοί αυτοματισμοί καθώς και είναι δυνατή η υλοποίηση εφαρμογών IoT στην ιατρική και το βιομηχανικό τομέα που να μπορούν να είναι προσιτές στο γενικό κοινό. [5]

Μελετητές συστήνουν ότι θα μπορούσαμε να λάβουμε υπόψιν μας το "πράγμα ή αντικείμενο" ως ένα αναπόσπαστο μείγμα υλικού, λογισμικού, δεδομένων και υπηρεσιών. Με βάση τα παραπάνω, το Διαδίκτυο των Πραγμάτων θα μπορούσε να καλύψει πάρα πολλές εφαρμογές και να γίνει μέρος της υλοποίησης έξυπνων πόλων, έξυπνων σχημάτων (smart grids), έξυπνων αυτοκινήτων, έξυπνων μετρητών ηλεκτρικής ενέργειας και πολλών άλλων πραγμάτων.[1] Το IoT είναι μια συνεχώς εξελίξιμη τεχνολογία, οι δυνατότητες που δίνει είναι απεριόριστες και γι' αυτό δεν είναι τυχαίο που μελετάται ιδιαίτερα τα τελευταία χρόνια. Η έμπνευση δεν έχει όρια και αυτό είναι που κάνει ακόμα και το Διαδίκτυο να τροποποιεί τα χαρακτηριστικά του και να μας ξανασυστήνεται σε μία πιο ολοκληρωμένη έκδοση του. Η ιδέα της διασύνδεσης κάθε πράγματος με το διαδίκτυο και της αλληλεπίδρασης των ευφυών μηχανών μεταξύ τους αποτελεί τεχνολογία αιχμής, αλλά η τεχνολογία πίσω από την οποία υλοποιείται το Διαδίκτυο των Πραγμάτων δεν μας είναι άγνωστη για εμάς. [6] Παρακάτω στην Εικόνα 1 βλέπουμε και την γραφική απεικόνιση του οικοσυστήματος του Διαδικτύου των Πραγμάτων, οπού το υπολογιστικό νέφος έχει τον κεντρικό ρόλο στην διασύνδεση των ολοένα και περισσότερων καθημερινών συσκευών ή και οχημάτων και εξοπλισμού ασφάλειας, προσφέροντας πληθώρα δυνατών εφαρμογών και υλοποιήσεων.



Εικόνα 1: Οικοσύστημα Διαδικτύου των Πραγμάτων [7]

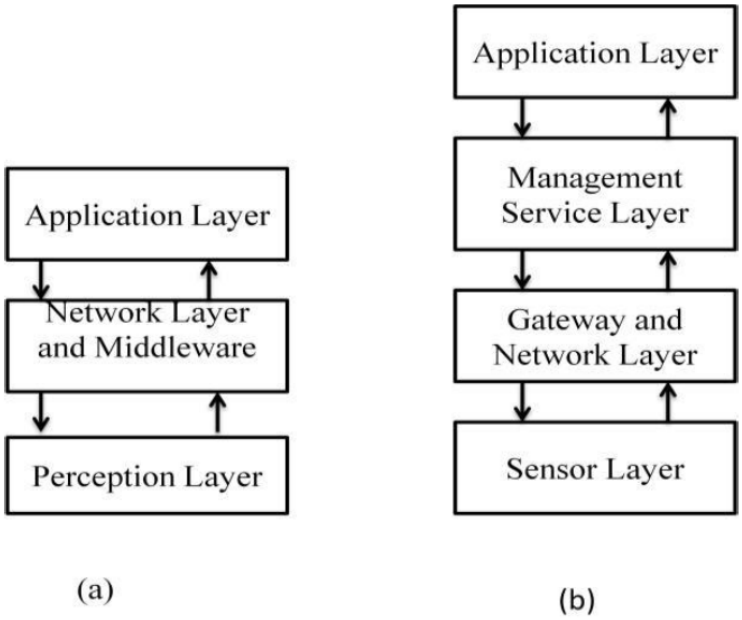
1.3 Αρχιτεκτονική Εφαρμογών στο Διαδίκτυο των Πραγμάτων

Η αρχιτεκτονική που θα ακολουθήσει μια εφαρμογή του Διαδικτύου των Πραγμάτων βασίζεται στην εφαρμογή που εξυπηρετεί και το πρόβλημα στο οποίο δίνει λύση. Στην Εικόνα 2 παρουσιάζονται τα δύο πιο γνωστά μοντέλα αρχιτεκτονικής υλοποίησης του Διαδικτύου των Πραγμάτων, χρησιμοποιώντας μια δομή τριών και τεσσάρων επιπέδων.

Δύο εφαρμογές του Διαδικτύου των Πραγμάτων όπου έχουν ως αντικείμενο την μέτρηση της ποιότητας και της σύστασης της ατμόσφαιρας μπορούν να ακολουθήσουν διαφορετική υλοποίηση, αναλόγως των απαιτήσεων και αναγκών της κάθε εφαρμογής ή της κρισιμότητας του προβλήματος στο οποίο δίνουν λύση.

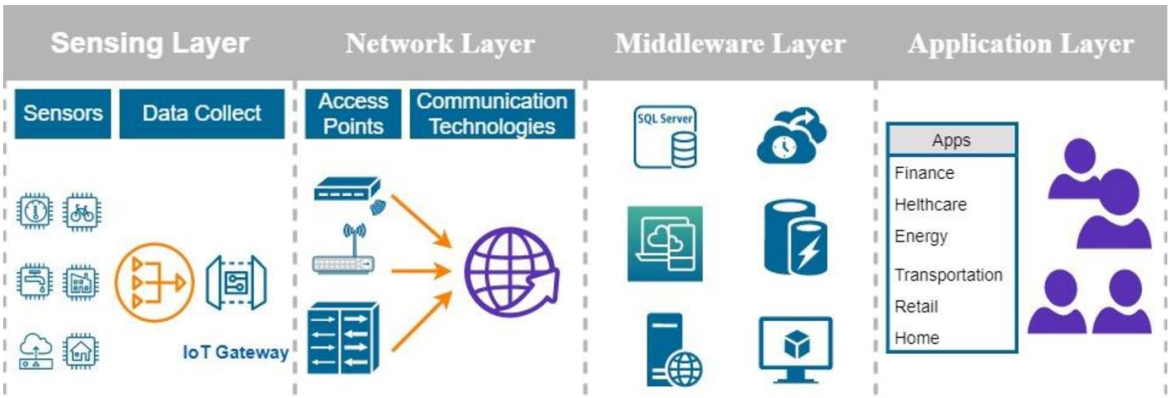
Μια εφαρμογή του Διαδικτύου των Πραγμάτων μπορεί να είναι οι έξυπνες συσκευές για την μόλυνση της ατμόσφαιρας, όπου αισθητήρια μετρούν το μονοξείδιο του άνθρακα, το διοξείδιο του αζώτου, το επίπεδο του θορύβου κ.α. μεγέθη και στέλνουν τα δεδομένα τους συνεχώς σε μια κεντρική βάση δεδομένων. Αυτά τα δεδομένα αναλύονται με κατάλληλους αλγορίθμους και εργαλεία και προκύπτουν πληροφορίες σχετικά με τη μόλυνση της ατμόσφαιρας ανά περιοχή, στέλνοντας την πληροφορία αυτή στην τροχαία. Η πληροφορία αυτή βοηθά την τροχαία να πάρει μέτρα όταν οι τιμές μόλυνσης υπερβούν τα κανονικά όρια. Εδώ το “sensors layer” χρησιμοποιεί αισθητήρια, λαμβάνει συνεχώς μετρήσεις της ατμόσφαιρας και στέλνει τα δεδομένα με ενσύρματη ή ασύρματη επικοινωνία σε μία βάση δεδομένων. Τα δεδομένα αυτά θα επεξεργαστούν και θα αναλυθούν και τα τελικά αποτελέσματα θα σταλούν στα έξυπνα τηλέφωνα των χρηστών μέσω της εφαρμογής «Έλεγχος Μόλυνσης Ατμόσφαιρας» του Υπουργείου Μεταφορών. Σε αυτή την περίπτωση, είναι απαραίτητη η υλοποίηση αρχιτεκτονικής τεσσάρων επιπέδων.

Μία άλλη πιο κρίσιμη αλλά αντίστοιχη εφαρμογή IoT αποτελεί έναν μηχανισμό για κουζίνες ή φιάλες αερίου, με διατάξεις ανίχνευσης διαρροής αερίου. Οποτεδήποτε ο αισθητήρας ανιχνεύσει αέριο, πρέπει να ειδοποιήσει μέσω συναγερμού τους ανθρώπους που μπορεί να βρίσκονται στον χώρο ή κοντά σε αυτό και να ειδοποιήσει και τον ιδιοκτήτη. Σε αυτή τη περίπτωση, η ανάλυση πρέπει να γίνει μέσα στο “sensor layer/ perception layer” και η υλοποίηση που θα χρειαστεί είναι η αρχιτεκτονική τριών επιπέδων. [1]



Εικόνα 2: Τα πιο γνωστά μοντέλα Αρχιτεκτονικής υλοποίησης IoT με 3 και 4 επίπεδα. [1]

Επιπλέον στην Εικόνα 3, βλέπουμε την αρχιτεκτονική και το πως υλοποιείται μια εφαρμογή Iot σε μοντέλο 4 επιπέδων. Οι συσκευές και οι διεργασίες σε κάθε επίπεδο διαδραματίζουν ένα ξεχωριστό ρόλο κάθε φορά ακολουθώντας μια τυποποίηση. [8]



Εικόνα 3: Υλοποίηση Αρχιτεκτονικής εφαρμογής IoT 4 επιπέδων **[8]**

1.4 Τεχνολογίες Επικοινωνίας στο Διαδίκτυο των Πραγμάτων

Το Διαδίκτυο των Πραγμάτων (IoT) αναφέρεται στη διαδικασία με την οποία πολλά πράγματα συνδέονται μεταξύ τους και ανταλλάσσουν τα δεδομένα τους με σκοπό την ανάπτυξη ευφυών συστημάτων που θα είναι ικανά να λειτουργούν ανεξάρτητα, να βελτιώνουν την αποδοτικότητα και να παρέχουν καινοτόμες υπηρεσίες. Αυτή η ανταλλαγή δεδομένων είναι απαραίτητη για τη λειτουργία των οικοσυστημάτων του Διαδικτύου των πραγμάτων, διότι επιτρέπει στις συσκευές να επικοινωνούν μεταξύ τους, να λαμβάνουν αποφάσεις με βάση τα δεδομένα που συλλέγονται σε πραγματικό χρόνο και να εκτελούν δραστηριότητες χωρίς την ανάγκη ανθρώπινης παρέμβασης. Αυτή η αλληλεπίδραση καθίσταται δυνατή με την ενσωμάτωση πολυάριθμων ασύρματων τεχνολογιών, όπως το WiFi, το LoRa, το GSM, το Bluetooth και το ZigBee, καθεμία από τις οποίες πληροί ένα ξεχωριστό σύνολο προδιαγραφών όσον αφορά τα σενάρια εφαρμογής, τον ρυθμό δεδομένων, την εμβέλεια και την κατανάλωση ενέργειας. Για παράδειγμα, το ZigBee και το LoRa είναι ιδιαίτερα γνωστά για τις χρήσεις τους σε συστήματα IoT και σε έξυπνα

περιβάλλοντα, προσφέροντας αντίστοιχα λύσεις που δίνουν προτεραιότητα στη μεγάλη εμβέλεια και τη χαμηλή κατανάλωση ενέργειας. [9]

Ακολούθως οι δικτυακές συσκευές που ονομάζονται gateways (πύλες) αποτελούν βασικά συστατικά των οικοσυστημάτων του Διαδικτύου των Πραγμάτων, καθώς χρησιμεύουν ως σύνδεσμοι μεταξύ των συσκευών και του δικτύου ή των υπηρεσιών cloud, επιτρέποντας τη συλλογή δεδομένων, τον έλεγχο των συσκευών και την πρόσβαση σε διάφορα πρωτόκολλα δικτύου. Παρέχουν τη δυνατότητα σε συσκευές με διαφορετικές τεχνολογίες επικοινωνίας να επεξεργάζονται δεδομένα τοπικά, να συνδέονται στο διαδίκτυο ή σε άλλα δίκτυα και να εκτελούν κρίσιμες εργασίες ασφαλείας. [9] Το IoT αποτελείται σήμερα από πληθώρα δικτύων που δεν έχουν ιδιαίτερα στενή σχέση μεταξύ τους. Επιπροσθέτως, πολλά διαφορετικά πρότυπα χρησιμοποιούνται και για τον χειρισμό των δικτύων και των υποδομών αυτών. Για παράδειγμα, τα δίκτυα χαμηλής ισχύος και μακρινών αποστάσεων (Low-Power Wide-Area) “LPWAN” είναι ένας νέος τύπος δικτύων ασύρματης επικοινωνίας. Ο νέος τύπος αυτός έρχεται να συμπληρώσει τα ασύρματα δίκτυα κυψελών και τα δίκτυα μικρής εμβέλειας, σε μία προσπάθεια να μπορούν να καλυφθούν οι ανάγκες των εφαρμογών κάτω από το πρίσμα του Διαδικτύου των Πραγμάτων. [10] Μέχρι σήμερα έχουν εφευρεθεί πολλά πρωτόκολλα επικοινωνίας πάνω στο LPWAN, μερικά από αυτά είναι τα: Narrow, Band-IoT, SigFox και LoRA. [10],[11]

Ωστόσο, η επιλογή της κατάλληλης τεχνολογίας, είτε πρόκειται για WiFi σε εφαρμογές υψηλού ρυθμού μετάδοσης, είτε για GSM με συνδεσιμότητα συσκευών σε μεγάλες εκτάσεις, είτε για Bluetooth για επικοινωνία μικρής εμβέλειας, είτε για ZigBee για ασύρματα δίκτυα πλέγματος (mesh) χαμηλής ισχύος, είτε για LoRa για εφαρμογές μεγάλης εμβέλειας και χαμηλής ισχύος, καθορίζεται από τις ειδικές απαιτήσεις της εκάστοτε εφαρμογής του Διαδικτύου των Πραγμάτων που υλοποιείται. [9]

Παρακάτω στον Πίνακας 1, βλέπουμε τα τεχνικά χαρακτηριστικά της κάθε τεχνολογίας μετάδοσης δεδομένων, συνδυάζοντας τα δεδομένα αυτά με τις απαιτήσεις της κάθε εφαρμογής μπορούμε να εκλέξουμε την πιο κατάλληλη λύση, είτε πρόκειται υλοποίηση του Διαδικτύου των πραγμάτων είτε όχι. [12]

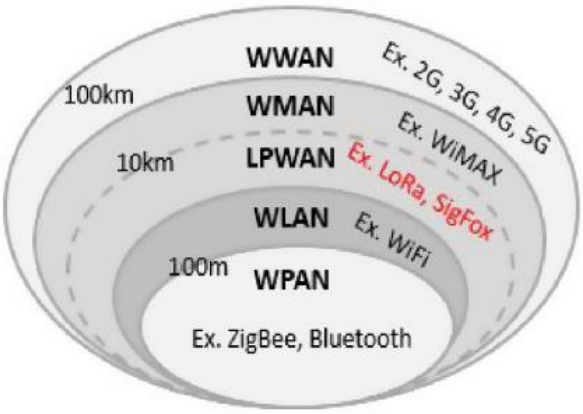
Πίνακας 1: Σύγκριση χαρακτηριστικών ασυρμάτων τεχνολογιών επικοινωνίας [12]

Wireless Standard	Power	Transmission Range (typical)	Data Rates
Bluetooth	Medium	1 to 100 m	1 to 3 Mbps
Bluetooth LE	Lower	>100 m	125 kbps to 2 Mbps
LoRaWAN	Low	10 km	0.3 to 50 kbps
NB-IoT	Low	<35 km	20 kbps to 5 Mbps
NFC	Low	<10 cm	106 to 424 kbps
Sigfox	Low	3 to 50 km	100 to 600 bps
6LoWPAN	Low	100 m	0 to 250 kbps
802.11/Wi-Fi	Medium	100 m to several km (with boosters)	10 to 100+ Mbps
802.15.4/Zigbee	Low	10 to 100 m	20 to 250 kbps
Z-Wave	Low	15 to 150 m	9.6 to 40 kbps

Πιο συγκεκριμένα, η χρήση της τεχνολογίας LoRa έχει προσελκύσει ενδιαφέρον λόγω της καταλληλότητας της για έξυπνα κτίρια και άλλες εφαρμογές του Διαδικτύου των Πραγμάτων που απαιτούν εκτεταμένη κάλυψη και υψηλή ενεργειακή απόδοση. Πρόσφατες έρευνες εξέτασαν την κατασκευή και την ανάλυση επιδόσεων ενός έξυπνου δικτύου αισθητήρων κτιρίων με βάση το LoRa και το ZigBee, γεγονός που αναδεικνύει τα αλληλένδετα πλεονεκτήματα των δύο τεχνολογιών στη σχεδίαση ευέλικτων και αποδοτικών συστημάτων του Διαδικτύου των Πραγμάτων. Λόγω της χαμηλής κατανάλωσης ενέργειας και της εκτεταμένης εμβέλειας μετάδοσης, το LoRa είναι μια εξαιρετική επιλογή για εφαρμογές έξυπνων κτιρίων που παρακολουθούν και ρυθμίζουν διάφορες παραμέτρους, όπως το φως, η υγρασία και η θερμοκρασία. Αυτές οι εφαρμογές συμβάλλουν στην εξοικονόμηση ενέργειας και παρέχουν καλύτερο περιβαλλοντικό έλεγχο. Αυτές οι θεμελιώδεις τεχνολογίες, καθώς και η λειτουργία που διαδραματίζουν τα gateways στις υποδομές του Διαδικτύου των Πραγμάτων, θέτουν τις βάσεις για μια πιο εμπεριστατωμένη διερεύνηση των δυνατοτήτων και των χρήσεων της τεχνολογίας LoRa, η οποία θα συζητηθεί στις επόμενες παραγράφους. [13]

1.4.1 Long Range (LoRa) – Ένα Ασύρματο Μητροπολιτικών Δίκτυο

Το LoRa (long range) είναι ένα πολλά υποσχόμενο πρωτόκολλο δικτύωσης μεγάλών αποστάσεων και χαμηλής ηλεκτρικής ισχύος, [5] πρόκειται για μια νέα τεχνολογία επικοινωνίας που προτάθηκε από το “LoRA Alliance” και αναπτύχθηκε από την “Semtech”. Ανήκει στην κατηγορία των Ασύρματων Μητροπολιτικών δίκτυών (WMAN) χρησιμοποιώντας το πρωτόκολλο του LPWAN. Όπως φαίνεται στην Εικόνα 4 και στο Πίνακας 1 το LoRa στοχεύει στην παροχή ασύρματης δικτύωσης σε συσκευές μεγάλης χρονικής ζωής, με χαμηλή ροή δεδομένων που βρίσκονται σε μακρινές αποστάσεις. [11]



Εικόνα 4. LPWAN και LORA έναντι άλλων ασύρματων δικτύων [11]

Το LoRa λειτουργεί στις συχνότητες 863-870MHz στην Ευρώπη και 902-928MHz στις ΗΠΑ, με τρία διαθέσιμα εύρη (bandwidths) 125KHz, 250KHz, 500KHz. Κάθε συχνότητα έχει προκαθορισμένο “duty cycle” και οι συσκευές που υιοθετούν το δίκτυο LoRa θα πρέπει να ακολουθούν πιστά τις συχνότητες που αντιστοιχούν στο LoRa, ανάλογα την χώρα που βρίσκονται. Η μετάδοση των σημάτων LoRa βασίζεται στο φάσμα εξάπλωσης chirp “CSS”, το οποίο έχει μεγάλη χρήση σε στρατιωτικές και διαστημικές εφαρμογές εξαιτίας της μεγάλης του εμβέλειας και την ανοχή του σε παρεμβολές. Αυτός ο τρόπος μετάδοσης, ο chirp, χρησιμοποιεί διάφορους συντελεστές ρυθμού μετάδοσης “Spreading factors” με

στόχο να αποτρέψει καταστροφικές συγκρούσεις σημάτων, όταν μεγάλο φορτίο σημάτων περνάει από το ίδιο κανάλι και με ίδιο SF. [10]

Οι παράμετροι μετάδοσης σημάτων LoRa αναλύονται παρακάτω:

- Συντελεστής ρυθμού μετάδοσης (Spread Factor). Είναι η αναλογία μεταξύ συμβόλων δεδομένων και ρυθμού chirp, και παίρνει τιμές μεταξύ 7 και 12. Το γεωγραφικό εύρος της επικοινωνίας επηρεάζεται τέντωνα από αυτή τη παράμετρο. Εφόσον το SF καθορίζει τον αριθμό των bits που θα χρησιμοποιηθούν για να κωδικοποιηθεί ένα σήμα, μια μεγάλη τιμή ως SF θα οδηγούσε σε μεγαλύτερο χρόνο μετάδοσης για κάθε μεταφορά ενός σύμβολου χαρακτήρα, προκαλώντας τα σήματα να ταξιδεύουν αργότερα και να φτάνουν μακρύτερα.
- Ισχύς Μετάδοσης (Transmit Power), η οποία λαμβάνει τιμές: 2, 5, 8, 11, 14 dBm.
- Μήκος κύματος (Band Width), το οποίο επηρεάζει το data rate της μετάδοσης και μπορεί να λάβει τρείς τιμές (125 kHz, 250 kHz, 500 kHz).
- Ρυθμός Κώδικα (Coding Rate), το οποίο αποτελεί το συντελεστή στον συντελεστή διόρθωσης λάθους και μπορεί να λαμβάνει τιμές: 5/4, 6/4, 7/4, 8/4.
- Κεντρική συχνότητα (Central Frequency) η οποία ορίζεται από την συχνότητα μετάδοσης που ισχύει για την συγκεκριμένη περιοχή. [10]

1.4.2 LoRAWAN

Οι όροι LoRa και LoRaWAN χρησιμοποιούνται εναλλάξ για τον ίδιο σκοπό, στην πραγματικότητα ωστόσο έχουν διαφορετική έννοια στην τεχνολογία της επικοινωνίας. Από τη μία πλευρά το LoRa είναι ένα μια ιδιόκτητη τεχνική ραδιομετάδοσης που αναπτύχθηκε από την SemTech και αναφέρεται στο φυσικό επίπεδο των επικοινωνιών. Από την άλλη πλευρά, το LoRaWAN είναι μια προδιαγραφή για το επίπεδο MAC (Medium Access Control). Πρόκειται για ένα LPWAN πρωτόκολλο που βασίζεται στο LORA και έχει επίσημα αναγνωριστεί από την Διεθνή Ένωση Τηλεπικοινωνιών (International Telecommunication Union). [11], [14]

Ο φορέας LoRa Alliance έχει αναλάβει την ανάπτυξη και την υποστήριξη του LoRaWAN, που βρίσκεται σήμερα στην έκδοση 1.1, στόχος είναι η βελτιστοποίηση της μετάδοσης δεδομένων με γνώμονα την ενεργειακή κατανάλωση και την εκπλήρωση των προδιαγραφών και τις απαιτήσεων των IoT συσκευών. Το LoRaWAN προσφέρει end-to-end ασφαλή αμφίδρομη επικοινωνία μέσω τοπολογίας δικτύου starts-of-stars, όπου πύλες (gateways / GWs) αναμεταδίδουν μνήματα μεταξύ των τελικών συσκευών και του διακομιστή δικτύου (Network Server). [14]

Τα τρία θεμελιώδη στοιχεία της αρχιτεκτονικής του LoRaWAN αναλύονται παρακάτω:

- Οι Τελικές συσκευές (End Devices) ονομάζονται τα αισθητήρια ή οι ενεργοποιητές που ασύρματα συνδέονται στο LoRaWAN δίκτυο. Μπορούν να στέλνουν (uplink) και να λαμβάνουν (downlink) δεδομένα προς ή από έναν διακομιστή δικτύου μέσω μιας πύλης LoRaWAN.
- Οι Πύλες (Gateways) αναλαμβάνουν έναν κρίσιμο ρόλο στο δίκτυο LoRaWAN. Λειτουργούν ως αναμεταδότες μεταξύ των τελικών συσκευών και του διακομιστή δικτύου. Αποκωδικοποιούν τα πακέτα που λαμβάνονται από τις συσκευές, τα κωδικοποιούν και τα μεταφέρουν παρακάτω ή αντίστροφα. Σε αντίθεση με τις

τελικές συσκευές που έχουν περιορισμένη ισχύ, οι πύλες είναι συνεχώς ηλεκτρικά τροφοδοτούμενες και συνδέονται στο Internet μέσω κλασικών γρήγορων μεθόδων επικοινωνίας, όπως Wi-Fi και Ethernet. Οι τελικές συσκευές επικοινωνούν με τις πύλες σε μοτίβου N-to-N (πολλά σε πολλά), οπού τα εξερχόμενα μηνύματα μεταδίδονται ευρέος (broadcast) και μπορούν να ληφθούν από οποιαδήποτε πύλη.

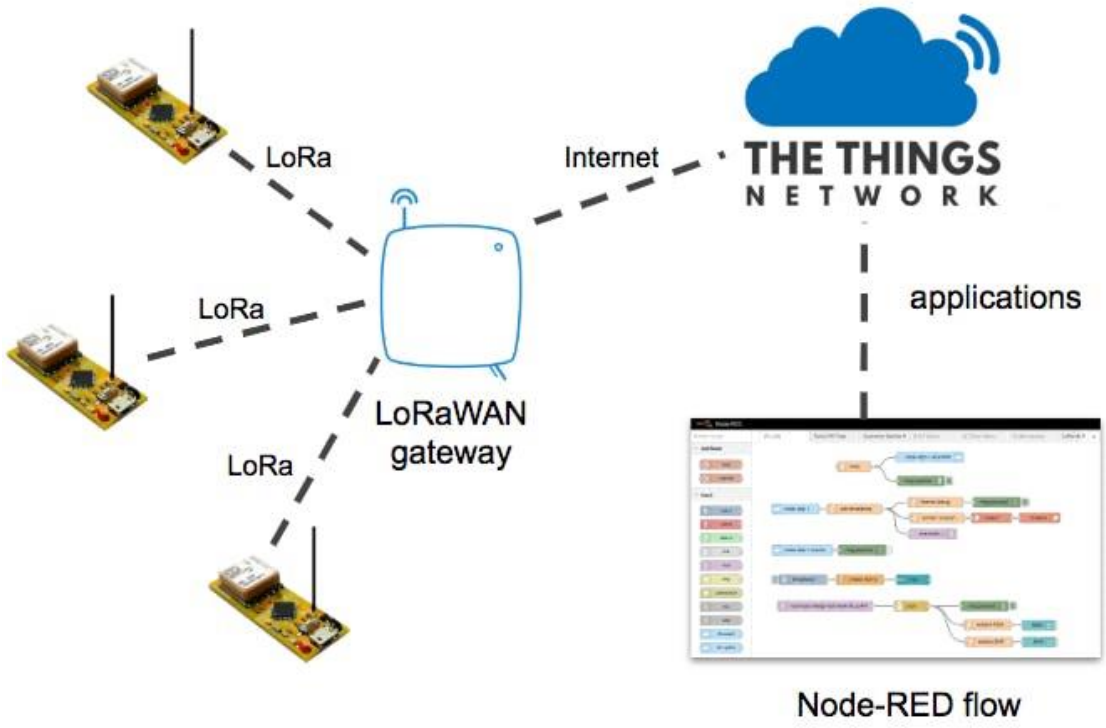
- Ο Διακομιστής δικτύου (Network Server) είναι το κύριο στοιχείο της αρχιτεκτονικής του LoRaWAN. Είναι αρμόδιο να λαμβάνει όλα τα εξερχόμενα μηνύματα των τελικών συσκευών και να τα μοιράζει στο επίπεδο εφαρμογής. Οι πύλες και ο διακομιστής δικτύου υλοποιούν επικοινωνία μοτίβου N-to-1 (πολλά σε ένα). Ωστόσο, τα εισερχόμενα μηνύματα των τελικών συσκευών (από την εφαρμογή προς τη συσκευή) μεταδίδονται σε αυτές μέσω της πύλης που έχει τις καλύτερες τιμές SNR (Signal Noise Ratio) για να προωθήσει τα δεδομένα.

Το πρωτόκολλο του LoRaWAN προσφέρει μια μέθοδο για ευέλικτο ρυθμό μεταφοράς δεδομένων (Adaptative Data Rate), με αυτό τον τρόπο οι συσκευές επιτρέπεται να επιλέγουν μόνες τους το ιδανικό συντελεστή ρυθμού μετάδοσης (Spread Factor) βάση των τιμών SNR που λαμβάνονται από την πύλη τους. Αυτή η μέθοδος στοχεύει στην χρήση του χαμηλότερου δυνατού συντελεστή ρυθμού μετάδοσης, εξοικονομώντας κατανάλωση ενέργειας και μεγιστοποιώντας την αποτελεσματικότητα της μετάδοσης των πακέτων. [14]

1.4.3 The Things Network

Το The Things Network (TTN) λειτουργεί ως η υποδομή του Διαδικτύου που επιτρέπει τη χρήση των τελικών συσκευών για πρακτικούς σκοπούς στο Διαδίκτυο. Οι τελικές συσκευές, γνωστές και ως κόμβοι, επικοινωνούν με μια κοντινή πύλη (gateway) που είναι συνδεδεμένη με την υποδομή του Διαδικτύου των Πραγμάτων. Μια εφαρμογή αναφέρεται σε οποιαδήποτε έννοια στο Διαδίκτυο που χρησιμοποιεί τα δεδομένα που παράγονται από τις τελικές συσκευές. Το “The Things Network” συνδέει τη ραδιοεπικοινωνία με την τεχνολογία του Διαδικτύου. Επιπλέον, η επικοινωνία δεν χρειάζεται να περιορίζεται σε μία κατεύθυνση- οι εφαρμογές έχουν τη δυνατότητα να στέλνουν δεδομένα και προς τις τελικές συσκευές. Ένα παράδειγμα εφαρμογής που χρησιμοποιείται τακτικά είναι ένας πίνακας οργάνων Cayenne, ο οποίος παρέχει έναν απλό και βολικό τρόπο οπτικοποίησης δεδομένων που προέρχονται από τις συσκευές IoT. Ενώ, το Node-RED είναι ένα ευρέως

χρησιμοποιούμενο εργαλείο για την κατασκευή λύσεων IoT, το οποίο επιτρέπει την απρόσκοπτη λήψη δεδομένων από το TTN.



Εικόνα 5: Ο ρόλος του TTN στο Διαδίκτυο των Πραγμάτων [15]

Στην Εικόνα 5 παρουσιάζεται η λειτουργία μιας πύλης (LoRaWAN gateway) στο TTN. Είναι απαραίτητο να ορίσουμε μια εφαρμογή στο TTN προκειμένου να συνδέσουμε κόμβους τελικών συσκευών, ακόμη και αν η πραγματική λειτουργικότητα της εφαρμογής μας υλοποιείται αλλού στο Διαδίκτυο. Σε μια εφαρμογή αποδίδεται συνήθως ένα όνομα και ένα μοναδικό αναγνωριστικό γνωστό ως “Application EUI”. Επιπλέον, δημιουργείται ένα κλειδί πρόσβασης εφαρμογής που μπορεί να χρησιμοποιηθεί για την απρόσκοπτη ενσωμάτωση με άλλες υπηρεσίες, όπως η Cayenne ή η Node-RED. [15]

2 ΚΕΦΑΛΑΙΟ 2ο : Υπολογιστικό Νέφος - Cloud Computing

2.1 Εισαγωγή στο Υπολογιστικό Νέφος - Cloud Computing

Το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) των Ηνωμένων Πολιτειών της Αμερικής, δίνει τον ορισμό του υπολογιστικού νέφους ακολούθως.

"Το υπολογιστικό νέφος είναι μια φιλοσοφία που εξασφαλίζει εύκολη, άμεση δικτυακή πρόσβαση σε μια κοινόχρηστη ομάδα διαμορφώσιμων υπολογιστικών πόρων, όπως διακομιστές (servers), δίκτυα, αποθηκευτικοί χώροι, εφαρμογές και υπηρεσίες, οι οποίοι μπορούν να παρέχονται και να διατίθενται γρήγορα με μικρή συμμετοχή των παρόχων της υπηρεσίας ή της διαχείρισης." [16]

Ωστόσο, αν και η ιδέα του υπολογιστικού νέφους υπάρχει εδώ και καιρό, δεν είναι απλώς μια καινούργια έννοια στον κόσμο των υπολογιστών. Το Υπολογιστικό Νέφος κέρδισε την προσοχή και τη δημοτικότητα στον κλάδο της πληροφορικής λίγα χρόνια μετά το σκάσιμο της φούσκας του ".com" το 1999, όταν εταιρείες όπως η Amazon, που είχαν κάνει σημαντικές επενδύσεις σε υπολογιστικές υποδομές, άρχισαν να εκμισθώνουν το αχρησιμοποίητο και ανεκμετάλλευτο υλικό τους με την τεχνολογία του νέφους. Για διάφορους λόγους, το υπολογιστικό νέφος έχει μετατραπεί σε περισσότερο από μια εναλλακτική λύση τα τελευταία χρόνια. Ένα παράδειγμα θα μπορούσε να είναι η μετάβαση στο "web 2.0", όπου οι πάροχοι υπηρεσιών χρησιμοποιούν διάφορες διαδικτυακές υπηρεσίες για να μεταφέρουν τις προσφορές τους από την τοπική υποδομή σε εξωτερικές υπηρεσίες. Πριν από μερικά χρόνια, μια διαδικτυακή επιχείρηση θα έπρεπε να υπογράψει δαπανηρές, μακροχρόνιες συμβάσεις με τράπεζες, εταιρείες πιστωτικών καρτών, εταιρείες ασφαλείας και άλλες επιχειρήσεις προκειμένου να φιλοξενήσει και να διαχειριστεί τη δική της πύλη πληρωμών και το δικό της σύστημα χρέωσης. Οποιοσδήποτε μπορεί πλέον να δέχεται πιστωτικές κάρτες χωρίς συμβόλαιο ή μακροπρόθεσμη δέσμευση και να χρησιμοποιεί τις υπηρεσίες πληρωμής κατά βούληση χάρη στην άνοδο εταιρειών όπως η PayPal, η BillDesk και η CCAvenue. Σήμερα σχεδόν κάθε νοικοκυριό έχει πλέον πρόσβαση στο Διαδίκτυο, ένα μέσω το οποίο θεωρείται πλέον εμπόρευμα, έτσι οι επιχειρήσεις μπορούν να μεταφέρουν τις υπηρεσίες λογισμικού τους με μεγαλύτερη ασφάλεια στο Διαδίκτυο, ανοίγοντας τη δυνατότητα χρήσης πλατφορμών υπολογιστικού νέφους. Το υπολογιστικό νέφος επωφελείται σημαντικά από τις οικονομίες κλίμακας. Οι πάροχοι υπολογιστικού νέφους μπορούν να μειώσουν τις τιμές τους με την κατασκευή μεγάλων κέντρων δεδομένων, ενώ οι πελάτες των υποδομών του υπολογιστικού νέφους μπορούν να λαμβάνουν υπολογιστική ισχύ ανάλογα με τις ανάγκες τους από οποιονδήποτε πάροχο υπολογιστικού νέφους χωρίς να χρειάζεται να κάνουν σημαντικές επενδύσεις σε υποδομές. Αυτό μπορεί να οδηγήσει σε εξοικονόμηση κόστους και για τα δύο μέρη. [17]

Λαμβάνοντας υπόψη τις ριζοσπαστικές δυνατότητές του, το υπολογιστικό νέφος έχει αναδειχθεί σε βασικό παράγοντα καινοτομίας σε διάφορους τομείς, όπως η μεταποίηση, η υγειονομική περίθαλψη, η χρηματοδότηση και η εκπαίδευση. Με τη βοήθεια αυτής της παραδειγματικής προσέγγισης, οι επιχειρήσεις μπορούν πλέον να εκμεταλλεύονται ισχυρές υπολογιστικές υποδομές, όποτε τις χρειάζονται, επιτυγχάνοντας πρωτοφανή επίπεδα παραγωγικότητας, προσαρμοστικότητας και επεκτασιμότητας κατά τον χειρισμό απαιτητικών προβλημάτων στην παροχή υπηρεσιών.

2.2 Ολοκλήρωση Υπολογιστικού Νέφος και Διαδικτύου των Πραγμάτων

Ο συνδυασμός του υπολογιστικού νέφους με το Διαδίκτυο των Πραγμάτων γίνεται όλο και πιο δημοφιλής σε διάφορες εφαρμογές. Με στόχο την παροχή πληροφοριών σε πραγματικό χρόνο, το Διαδίκτυο των πραγμάτων επιτρέπει τη συλλογή και τη μετάδοση δεδομένων, που κατόπιν μπορούν να αξιοποιηθούν στο υπολογιστικό νέφος. Σε ένα δικτυακό περιβάλλον όπως αυτό, οι IoT συσκευές χρησιμεύουν ως πηγές δεδομένων και οι εφαρμογές που βασίζονται στο υπολογιστικό νέφος χρησιμοποιούνται για να διευκολύνουν την επεξεργασία και την ανάλυση των δεδομένων.

Η ενσωμάτωση του Διαδικτύου των Πραγμάτων με το υπολογιστικό νέφος επιτρέπει τη δημιουργία συστημάτων κλειστού βρόχου (closed-loop) που είναι σε θέση να ανταποκρίνονται σε πραγματικό χρόνο στις πληροφορίες που παράγονται από τους αισθητήρες και τις συσκευές IoT. Η παρακολούθηση αλλά και ο έλεγχος των συσκευών σε πραγματικό χρόνο καθίστανται δυνατά από ένα τέτοιο ολοκληρωμένο σύστημα, το οποίο επιτρέπει με την σειρά του την υλοποίηση εφαρμογών όπως η προγνωστική συντήρηση, η πρόγνωση σφαλμάτων και η προγνωστική ανάλυση σε ένα μηχανολογικό εξοπλισμό. Επιπροσθέτως, οι συσκευές του Διαδικτύου των Πραγμάτων μπορούν να ελέγχονται μέσω ενός ασφαλούς και αξιόπιστου περιβάλλοντος υπολογιστικού νέφους, το οποίο επιτρέπει στους χρήστες να έχουν πρόσβαση στα δεδομένα τους από οποιαδήποτε τοποθεσία και ανά πάσα στιγμή.

Περαιτέρω, το υπολογιστικό νέφος προσφέρει επεκτασιμότητα και μειώνει το κόστος διαχείρισης υλικού και λογισμικού, γεγονός που καθιστά δυνατή την ανάπτυξη διασυνδεδεμένων διατάξεων σε ευρεία κλίμακα. Συνολικά, ο συνδυασμός του Διαδικτύου των Πραγμάτων (IoT) με το υπολογιστικό νέφος προσφέρει πληθώρα πλεονεκτημάτων και ανοίγει το δρόμο για την ανάπτυξη μοναδικών και συναρπαστικών εφαρμογών. Η υποδομή του Διαδικτύου των Πραγμάτων μπορεί να υλοποιηθεί και να συντηρηθεί γρήγορα με μικρότερους πόρους χάρη στο υπολογιστικό νέφος, γεγονός που το καθιστά πιο προσιτό και άμεσα διαθέσιμο για διάφορες εφαρμογές σε κάθε οργανισμό. Η αύξηση της επιχειρησιακής απόδοσης, η βελτίωση της προγνωστικής ανάλυσης και η αύξηση του ελέγχου και της ασφάλειας είναι όλα επιτεύγματα που μπορούν να πραγματοποιηθούν μέσω της αξιοποίησης του υπολογιστικού νέφους και του Διαδικτύου των πραγμάτων. [18]

2.2.1 Πρότυπες εφαρμογές IoT και υπολογιστικού νέφους

Μερικές πρότυπες εφαρμογές ως αποτέλεσμα της ολοκλήρωσης και του συνδυασμού του υπολογιστικού νέφους με το διαδίκτυο των πραγμάτων αναλύονται παρακάτω.

- Έξυπνα σπίτια και οικιακός αυτοματισμός: Οι πλατφόρμες νέφους του IoT συμβάλλουν στον οικιακό αυτοματισμό, προσφέροντας στους ιδιοκτήτες σπιτιών τον έλεγχο των έξυπνων θερμοστατών, του φωτισμού και των συστημάτων ασφαλείας. Αυτή η ενσωμάτωση βελτιστοποιεί τη χρήση ενέργειας, ενισχύει την ασφάλεια και εξασφαλίζει ευκολία μέσω δυνατοτήτων απομακρυσμένης διαχείρισης.
- Υγειονομική περίθαλψη και απομακρυσμένη παρακολούθηση: Οι πλατφόρμες IoT που βασίζονται στο νέφος φέρνουν επανάσταση στην υγειονομική περίθαλψη, επιτρέποντας την απομακρυσμένη παρακολούθηση ασθενών, την τηλεϊατρική και την εξατομικευμένη φροντίδα. Οι φορητές συσκευές υγείας και οι αισθητήρες

μεταδίδουν δεδομένα στο cloud για ανάλυση, βελτιώνοντας τα αποτελέσματα των ασθενών και την προσβασιμότητα στην υγειονομική περίθαλψη.

- Βιομηχανικός αυτοματισμός και έξυπνη παραγωγή: Στον βιομηχανικό τομέα, οι πλατφόρμες IoT στο νέφος διευκολύνουν την παρακολούθηση των μηχανημάτων σε πραγματικό χρόνο, την προγνωστική συντήρηση και τη βελτιστοποίηση της εφοδιαστικής αλυσίδας. Αυτό έχει ως αποτέλεσμα την αύξηση της παραγωγικότητας, τη μείωση του λειτουργικού κόστους και τη βελτίωση της ποιότητας των προϊόντων.
- Έξυπνες πόλεις: Το Διαδίκτυο των Πραγμάτων και το υπολογιστικό νέφος προωθούν πρωτοβουλίες έξυπνων πόλεων με τη διαχείριση συστημάτων κυκλοφορίας, τη διαχείριση απορριμμάτων και την περιβαλλοντική παρακολούθηση. Οι αναλύσεις δεδομένων σε πραγματικό χρόνο βελτιστοποιούν την κατανομή των πόρων, βελτιώνουν την αστική διαβίωση και υποστηρίζουν τις προσπάθειες βιωσιμότητας.
- Γεωργία ακριβείας: Αξιοποιώντας τις πλατφόρμες του διαδικτύου των πραγμάτων και του υπολογιστικού νέφους, η γεωργία ακριβείας αξιοποιεί δεδομένα αισθητήρων και εικόνες από μη επανδρωμένα αεροσκάφη για τη βελτιστοποίηση της άρδευσης, τον έλεγχο των παρασίτων και την παρακολούθηση της υγείας των καλλιεργειών. Η προσέγγιση αυτή οδηγεί σε αυξημένες αποδόσεις καλλιεργειών και σε αποτελεσματική χρήση των πόρων.[19]

2.2.2 Ενίσχυση του IoT μέσω του υπολογιστικού νέφους

Το υπολογιστικό νέφος διαδραματίζει καθοριστικό ρόλο στην ανάλυση των δεδομένων που παράγονται από το IoT, παρέχοντας επεκτάσιμη αποθήκευση, ισχυρούς υπολογιστικούς πόρους και προηγμένες δυνατότητες επεξεργασίας δεδομένων. Οι τεχνικές επεξεργασίας δεδομένων σε πραγματικό χρόνο και οι τεχνικές επεξεργασίας τμηματικής πληροφορίας που υποστηρίζονται από πλατφόρμες νέφους, επιτρέπουν τον αποτελεσματικό χειρισμό τεράστιων συνόλων δεδομένων IoT, παρέχοντας έγκαιρες πληροφορίες και διευκολύνοντας τις διαδικασίες λήψης αποφάσεων.

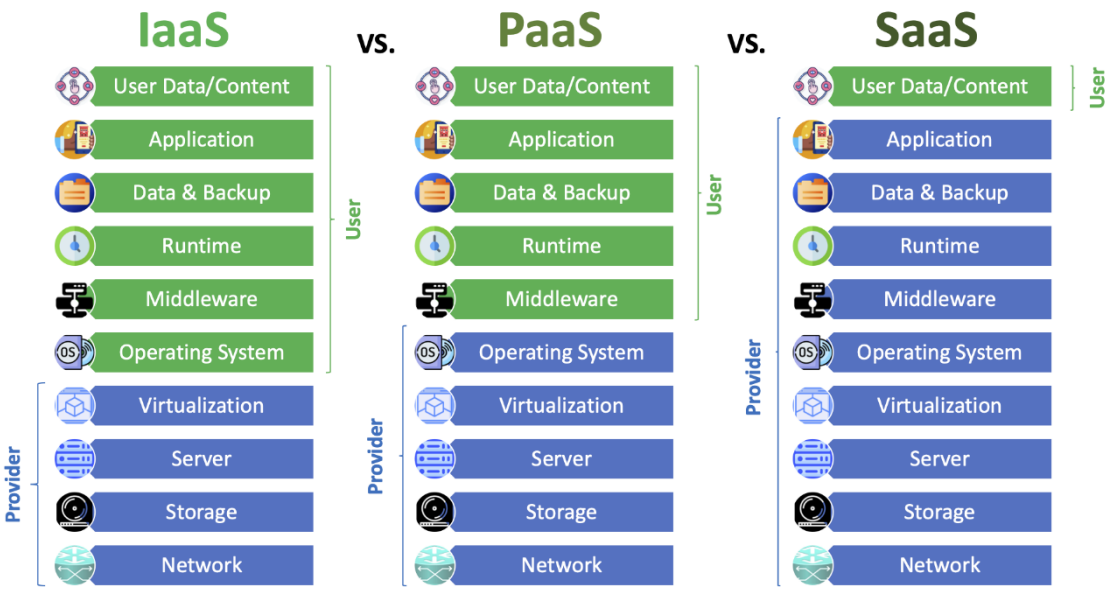
Η διαχείριση των συσκευών απλοποιείται μέσω πλατφορμών IoT που βασίζονται στο νέφος, προσφέροντας λειτουργίες για την παροχοποίηση των συσκευών, τη συνδεσιμότητα, την ασφάλεια και τις ενημερώσεις λογισμικού. Αυτό εξασφαλίζει την απρόσκοπτη λειτουργία των συσκευών του Διαδικτύου των Πραγμάτων, διατηρώντας την ασφάλεια και την αξιοπιστία τους.

Επιπλέον, η ενοποίηση του IoT με άλλες τεχνολογίες ενισχύεται σημαντικά από το υπολογιστικό νέφος. Οι πλατφόρμες cloud υποστηρίζουν τη σύγκλιση του IoT με το edge computing, την τεχνητή νοημοσύνη (AI) και τη μηχανική μάθηση (ML), προωθώντας την καινοτομία και επιτρέποντας την ανάπτυξη εξελιγμένων εφαρμογών IoT.

Συμπερασματικά, το υπολογιστικό νέφος όχι μόνο ενισχύει τις δυνατότητες των εφαρμογών IoT αλλά παρέχει επίσης ένα ισχυρό πλαίσιο για την ανάλυση, τη διαχείριση και την ενισχυμένη ενσωμάτωση των συσκευών IoT με άλλες τεχνολογίες αιχμής. Αυτή η σύμπραξη προωθεί την πρόοδο των εφαρμογών του Διαδικτύου των Πραγμάτων, προσφέροντας κλιμακούμενες, ασφαλείς και αποτελεσματικές λύσεις σε διάφορους τομείς. [19]

2.3 Μοντέλα αρχιτεκτονικής υπηρεσιών υπολογιστικού νέφους

Σύμφωνα με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας των Ηνωμένων Πολιτειών, το μοντέλο του υπολογιστικού νέφους μπορεί να υλοποιηθεί μέσω τριών διαφορετικών μοντέλων αρχιτεκτονικής.[16] Όπως βλέπουμε και στην Εικόνα 6, ανάλογα με το μοντέλο που ακολουθείται, ο χρήστης έχει περισσότερη ή λιγότερη πρόσβαση και έλεγχο στις υποδομές του παρόχου, οπού με την σειρά του και αυτός προσφέρει διαφορετικό όγκο υπηρεσιών. Παρακάτω θα αναλύσουμε από την πλευρά του πελάτη τα τρία μοντέλα που προσφέρονται από τους παρόχου υπολογιστικού νέφους.



Εικόνα 6: Σύγκριση μοντέλων IaaS, PaaS, και SaaS [20]

2.3.1 Infrastructure as a Service (IaaS)

Το Infrastructure as a Service, πιο γνωστό ως IaaS, είναι μια ευέλικτη και επεκτάσιμη υποδομή υπολογιστικού νέφους, η οποία βασίζεται σε εικονικές μηχανές.[21] Οι πελάτες έχουν τη δυνατότητα να κατανέμουν την επεξεργασία, την αποθήκευση, τα δίκτυα και άλλους βασικούς πόρους υπολογιστών, επιτρέποντάς τους να αναπτύσσουν και να υλοποιούν ένα ευρύ φάσμα εφαρμογών λογισμικού, όπως λειτουργικά συστήματα και προγράμματα. Ο πελάτης δεν έχει εξουσία ή έλεγχο επί της θεμελιώδους υποδομής του νέφους, αλλά έχει εξουσία επί των λειτουργικών συστημάτων, της αποθήκευσης και των εγκατεστημένων εφαρμογών. Επιπλέον, μπορεί να έχει περιορισμένο έλεγχο σε ορισμένα στοιχεία δικτύωσης, όπως τα τείχη προστασίας του κεντρικού υπολογιστή. [16]

Μια υπηρεσία IaaS μπορεί να βασιστεί σε ένα λειτουργικό σύστημα όπως είναι το Debian, μια εξαιρετική επιλογή λόγω της φήμης του για την αξιοπιστία, την ασφάλεια και τα μεγάλα αποθετήρια πακέτων. Ορισμένα πλεονεκτήματα από την ενσωμάτωσή του σε υποδομές IaaS είναι τα ακόλουθα:

- Αξιοπιστία και ασφάλεια: Ο ισχυρός σχεδιασμός του Debian και οι ενδελεχείς δοκιμές εγγυώνται ένα ασφαλές, σταθερό περιβάλλον για ζωτικής σημασίας εφαρμογές, μειώνοντας τις αστοχίες και τον χρόνο του downtime.

- Ευελιξία και προσαρμογή: Η αρχιτεκτονική ανοικτού κώδικα του Debian διευκολύνει σε μεγάλο βαθμό την παραμετροποίησή του ώστε να ανταποκρίνεται στις μοναδικές απαιτήσεις και προτιμήσεις της κάθε επιχείρησης. Αυτό είναι ιδιαίτερα χρήσιμο σε περιβάλλοντα Infrastructure as a Service (IaaS), όπου η ευελιξία είναι απαραίτητη.
- Κόστος-Αποδοτικότητα: Το Infrastructure as a Service (IaaS) με βάση το Debian μπορεί να είναι πιο προσιτό από κλασικές υπηρεσίες διαχείρισης υπολογιστικού νέφους, όπως το Amazon AWS ή το Microsoft Azure. Αυτό ισχύει ιδιαίτερα σε εταιρείες που προσπαθούν να μεγιστοποιήσουν τον προϋπολογισμό τους στον τομέα της πληροφορικής χωρίς να θυσιάσουν την απόδοση και την αξιοπιστία.
- Υποστήριξη από την κοινότητα: Λόγω της μεγάλης ομάδας χρηστών και της ισχυρής κοινότητας του Debian, οι χρήστες έχουν πρόσβαση σε μια ποικιλία κοινών γνώσεων και εργαλείων, σε ολοκληρωμένη τεκμηρίωση και σε βοήθεια από την ίδια την κοινότητα, όλα αυτά ενισχύουν την δημιουργικότητα και την ικανότητα επίλυσης προβλημάτων.
- Συμβατότητα και ενοποίηση: Οι διαχειριστές μπορούν να επωφελούνται πλήρως από το υπολογιστικό νέφος χάρη στη συμβατότητα του Debian με πολλές τεχνολογίες εικονικών μηχανών, όπως το KVM και το OpenVZ, γεγονός που εγγυάται την ομαλή ένωση και την αποτελεσματική λειτουργία μέσα σε πλαίσια IaaS.

Εξαιτίας αυτών των χαρακτηριστικών, το Debian είναι μια εξαιρετική επιλογή για επιχειρήσεις και όχι μόνο που θέλουν να επωφεληθούν από τα πλεονεκτήματα του IaaS χωρίς να εγκαταλείψουν τον έλεγχο των ιδιαιτεροτήτων της υποδομής του υπολογιστικού τους νέφους. [21], [22], [23], [24]

2.3.2 Platform as a Service (PaaS)

Στο Platform as a Service, πιο γνωστό ως PaaS, οι πελάτες έχουν τη δυνατότητα να εγκαθιστούν εφαρμογές, είτε αυτές δημιουργούνται από τον ίδιο τον καταναλωτή είτε αγοράζονται, μέσα από τις υποδομές του νέφους. Αυτές οι εφαρμογές αναπτύσσονται χρησιμοποιώντας γλώσσες προγραμματισμού, βιβλιοθήκες, υπηρεσίες και εργαλεία που υποστηρίζονται από τον πάροχο. Ο πελάτης δεν έχει τον έλεγχο ή την εξουσία επί της θεμελιώδους υποδομής του νέφους, συμπεριλαμβανομένου του δικτύου, των διακομιστών, των λειτουργικών συστημάτων και των μέσων αποθήκευσης. Ωστόσο, διαθέτει εξουσία επί των εγκατεστημένων εφαρμογών και ίσως επί των ρυθμίσεων διαμόρφωσης που αφορούν το περιβάλλον φιλοξενίας εφαρμογών. [16]

2.3.3 Software as a Service (SaaS)

Οι πελάτες έχουν τη δυνατότητα να χρησιμοποιούν τις εφαρμογές που προσφέρει ο πάροχος, οι οποίες φιλοξενούνται από την υποδομή του υπολογιστικού νέφους. Οι συσκευές των πελατών μπορούν να έχουν πρόσβαση στις εφαρμογές χρησιμοποιώντας είτε μια διεπαφή ενός ελαφριού τερματικού (thin client), όπως ένα πρόγραμμα περιήγησης στο διαδίκτυο (π.χ. ηλεκτρονικό ταχυδρομείο μέσω διαδικτύου), είτε μια διεπαφή προγράμματος. Με εξαίρεση τις περιορισμένες επιλογές διαμόρφωσης εφαρμογών που αφορούν συγκεκριμένους χρήστες, ο πελάτης δεν έχει τον έλεγχο της υποκείμενης

υποδομής του νέφους, η οποία περιλαμβάνει δίκτυο, διακομιστές, λειτουργικά συστήματα, αποθήκευση και συγκεκριμένες δυνατότητες εφαρμογών.[16]

2.4 Σημαντικοί πάροχοι υπηρεσιών υπολογιστικού νέφους

Ο κλάδος των υπηρεσιών υπολογιστικού νέφους έχει σημειώσει σημαντικές εξελίξεις από την αρχή του, με εξέχοντες παρόχους όπως την Amazon Web Services (AWS), το Microsoft Azure και άλλους να παίζουν καθοριστικό ρόλο στη διαμόρφωση της σημερινής εικόνας του κυβερνοχώρου. Οι υπηρεσίες υπολογιστικού νέφους έχουν μετατραπεί από απλές υποδομές σε ολοκληρωμένες πλατφόρμες διαχείρισης εφαρμογών, αποτελώντας απαραίτητα εργαλεία για εταιρείες και μεμονωμένους καταναλωτές σε παγκόσμιο επίπεδο. Η παρούσα ενότητα θα επικεντρωθεί στη σύγκριση των προσφερόμενων υπηρεσιών των κυριότερων παρόχων υπολογιστικού νέφους, στην αξιολόγηση των ιδιαίτερων πλεονεκτημάτων τους και στην παρουσίαση της θέσης τους στην αγορά, καταδεικνύοντας έτσι τη συμβολή τους στο διαμορφωμένο τεχνολογικό τοπίο.

2.4.1 Amazon Web Services (AWS)

Όταν οι υπηρεσίες Amazon Web Services (AWS) παρουσιάστηκαν για πρώτη φορά τον Ιούλιο του 2002, προσέφεραν μια μικρή επιλογή προϊόντων και υπηρεσιών. Μια πρόταση του 2003 από τους Chris Pinkham και Benjamin Black, η οποία κατέληξε σε ένα έγγραφο-ορόσημο και περιγράφει λεπτομερώς την τυποποίηση της λιανικής υπολογιστικής πολιτικής της Amazon μέσω αυτοματοποιημένων υπηρεσιών ιστού, είχε σημαντικό αντίκτυπο στην πρόοδο της πλατφόρμας. Αυτή η πρωτοποριακή προσπάθεια προετοίμασε το έδαφος το 2006 για το λανσάρισμα της AWS και την επανάσταση στην παροχή υπηρεσιών που βασίζονται στο υπολογιστικό νέφος για υποδομές πληροφορικής. Προκειμένου η γκάμα προϊόντων της AWS να παρέχει μια ολιστική προσέγγιση στο υπολογιστικό νέφος, διαιρείται σε τέσσερις βασικούς τομείς: υπολογισμοί (compute), αποθήκευση, βάσεις δεδομένων και υπηρεσίες δικτύωσης. [25]

Το 2006, η Amazon Web Services (AWS) γνώρισε μια σημαντική μεταμόρφωση με το ντεμπούτο τριών βασικών υπηρεσιών: Amazon S3, Amazon SQS (Simple Queue Service) και Amazon EC2 (Elastic Compute Cloud). Ο Andy Jassy, ιδρυτής και αντιπρόεδρος της AWS, τόνισε αυτή την ανάπτυξη τονίζοντας πώς το Amazon S3 μεταμόρφωσε τον τρόπο με τον οποίο οι προγραμματιστές αποθηκεύουν και διασφαλίζουν τα δεδομένα τους, αντιμετωπίζοντας τις συνήθεις ανησυχίες και καθιστώντας το περιβάλλον υπολογιστικού νέφους πιο αξιόπιστο και ασφαλές. [25]

Το Amazon EC2, ένα βασικό προϊόν της AWS, αντιπροσωπεύει τις κλιμακούμενες και ευέλικτες λύσεις υπολογιστικού νέφους της πλατφόρμας. Η απλή διεπαφή του EC2 επιτρέπει στους προγραμματιστές να αποκτούν, να ρυθμίζουν και να διαχειρίζονται γρήγορα υπολογιστικούς πόρους για υπολογισμούς διαδικτυακής κλίμακας. Αυτή η λύση παρέχει στους χρήστες πλήρη έλεγχο των υπολογιστικών πόρων τους και κλιμακώνεται γρήγορα για να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες. Είναι σημαντικό ότι το μοντέλο τιμολόγησης pay-as-you-go του Amazon EC2 επιτρέπει στους χρήστες να ελαχιστοποιούν τα έξοδα πληρώνοντας μόνο για τη δυναμικότητα που καταναλώνουν. Η οικονομική απόδοση του Amazon EC2 και τα εκτεταμένα εργαλεία για την κατασκευή εφαρμογών ανθεκτικών σε αστοχίες το καθιστούν ακρογωνιαίο λίθο των υπηρεσιών

υπολογιστικού νέφους της AWS, ενισχύοντας την αξιοπιστία, την ευελιξία και την οικονομική αποδοτικότητα της πλατφόρμας. [17], [25]

2.4.2 Microsoft Azure

Το Microsoft Azure, που αρχικά ονομαζόταν Project Red Dog, ξεκίνησε την πορεία του στα μέσα της δεκαετίας του 2000 με στόχο να ανταγωνιστεί την Amazon στον χώρο του υπολογιστικού νέφους. Η πλατφόρμα, η οποία παρουσιάστηκε για πρώτη φορά το 2008 από τον Ray Ozzie, επικεφαλής αρχιτέκτονα λογισμικού της Microsoft, έχει ως στόχο να προσφέρει μια ολοκληρωμένη γκάμα υπηρεσιών νέφους. Οι υπηρεσίες που προσέφερε το Azure ήταν το Windows Azure για υπηρεσίες υπολογιστικού νέφους, .NET υπηρεσίες για προγραμματιστές, κοινή χρήση αρχείων μέσω του Live Services, Microsoft SQL Services για τη διαχείριση βάσεων δεδομένων, Microsoft SharePoint Services και Dynamics CRM υπηρεσίες SaaS. Οι εν λόγω υπηρεσίες παρουσιάστηκαν σε ένα συνέδριο για τους επαγγελματίες προγραμματιστές της Microsoft. Αυτό το ευρύ φάσμα υπηρεσιών επισήμανε την αλλαγή στρατηγικής της Microsoft, καθιερώνοντας το Azure ως έναν ευέλικτο πάροχο υπηρεσιών υπολογιστικού νέφους.

Τον Φεβρουάριο του 2010, η Microsoft εισήλθε επίσημα στον κλάδο των υπηρεσιών νέφους με την κυκλοφορία της πλατφόρμας Windows Azure. Το Azure υπερέχει έναντι του ανταγωνισμού υποστηρίζοντας ένα μεγάλο εύρος γλωσσών προγραμματισμού, εργαλείων και frameworks και υποστηρίζοντας το λογισμικό ανοικτού κώδικα, γεγονός που το καθιστά μια δημοφιλή επιλογή για τις επιχειρήσεις. Το Azure εξυπηρετεί μια ευρεία σειρά αναγκών μέσα από τα βασικά του προϊόντα, τα οποία περιλαμβάνουν το λογισμικό ως υπηρεσία (SaaS), την πλατφόρμα ως υπηρεσία (PaaS) και την υποδομή ως υπηρεσία (IaaS). Το Azure ανταγωνίζεται τους άλλους τιτάνες του κλάδου, όπως το AWS, το Google Cloud και την IBM, αλλά παραμένει ξεχωριστό χάρη στις ευρείες δυνατότητες ολοκλήρωσης των υπηρεσιών του. Το Azure προσφέρει μια ισχυρή, καθολική πλατφόρμα που μπορεί να διαμορφωθεί σε διάφορες εφαρμογές και συνθέσεις δεδομένων εντός των παγκόσμιων κέντρων δεδομένων (datacenters) της Microsoft, είτε κάποιος δημιουργεί διαδικτυακές εφαρμογές, είτε αποθηκεύει δεδομένα, είτε βελτιώνει επιτόπιες εφαρμογές, είτε χρειάζεται σύνθετες λύσεις δικτύωσης. [17], [25]

3 ΚΕΦΑΛΑΙΟ 3ο : Επικοινωνία, συλλογή και αποθήκευση δεδομένων

3.1 Εισαγωγή στα πρωτόκολλα και στους τρόπους ανταλλαγής δεδομένων

Η μετάδοση μηνυμάτων μεταξύ των συσκευών είναι ζωτικής σημασίας στο Διαδίκτυο των Πραγμάτων, διότι, για να χειριστούμε ένα δίκτυο ή μια συσκευή του Διαδικτύου των Πραγμάτων πρέπει να μεταδώσει μια εντολή σε μια άλλη συσκευή. [4] Πέρα λοιπόν από το δίκτυο (Wi-Fi, LoRa, LTE, Bluetooth) το οποίο μια συσκευή του Διαδικτύου των Πραγμάτων θα χρησιμοποιήσει για να στείλει ή να λάβει δεδομένα προς ή από μια πύλη του διαδικτύου (gateway), χρειάζεται και η υλοποίηση ενός πρωτοκόλλου μεταφοράς των δεδομένων από ή προς τον διακομιστή της εφαρμογής. Δεδομένου ότι οι συσκευές IoT είναι κατασκευασμένες σε δίκτυα με περιορισμένο εύρος ζώνης, το πρωτόκολλο push είναι καλύτερη επιλογή για την ανταλλαγή μηνυμάτων από ό,τι το πρωτόκολλο polling. Αυτές οι υπηρεσίες μηνυμάτων push χρησιμοποιήθηκαν για την υλοποίηση των πρωτοκόλλων MQTT, XMPP και CoAP. Αυτές οι διαδικασίες μπορούν να χρησιμοποιηθούν σε διάφορες καταστάσεις. Συγκεκριμένα, το MQTT σχεδιάστηκε ως ένα ελαφρύ πρωτόκολλο που θα μπορούσε να λειτουργήσει σε συσκευές χαμηλής κατανάλωσης ενέργειας. Δεν είναι τυχαίο λοιπόν ότι έχει ενσωματωθεί σε πολλές συσκευές του Διαδικτύου των Πραγμάτων και σε συστήματα υπηρεσιών άμεσης ανταλλαγής μηνυμάτων. [4]

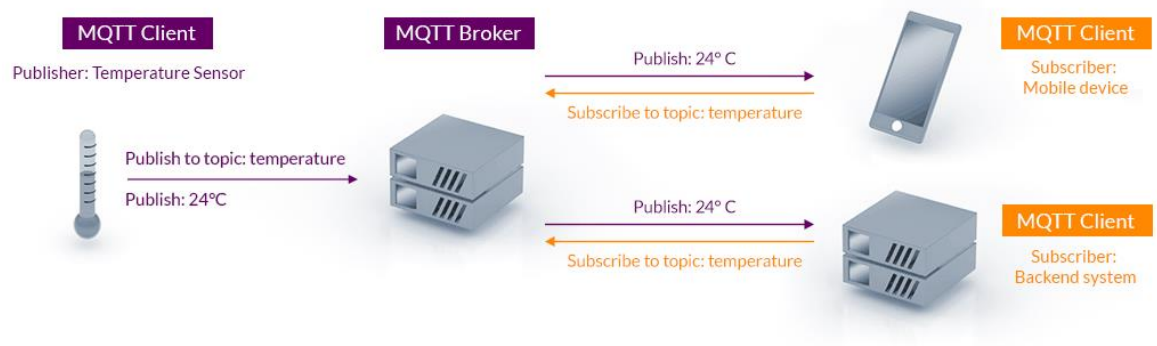
Σε σύγκριση με τα κανονικά δίκτυα, τα Ασύρματα Δίκτυα Αισθητήρων ("Wireless Sensor Networks", WSN) παρουσιάζουν μοναδικές προκλήσεις. Η επικοινωνία με επίκεντρο τα δεδομένα (Data-centric Communication) είναι ένα νέο πρότυπο επικοινωνίας που αναδύεται με σκοπό την επίλυση τέτοιου είδους ζητημάτων. Ενώ το σύστημα ανταλλαγής μηνυμάτων δημοσιεύσης/εγγραφής (publish/subscribe) είναι ένας από τους τύπους επικοινωνίας με επίκεντρο τα δεδομένα. Στην κατανεμημένη πληροφορική, τα συστήματα "publish/subscribe" είναι ευρέως διαδεδομένα σε σύγκριση με άλλους τύπους επικοινωνίας που επικεντρώνονται στα δεδομένα. Επομένως, η ενσωμάτωση εφαρμογών αισθητηρίων με άλλες κατανεμημένες εφαρμογές, θα είναι ευκολότερη εάν τα συστήματα "publish/subscribe" εξελιχθούν στα Ασύρματα Δίκτυα Αισθητήρων. [26]

3.2 Πρωτόκολλο ανταλλαγής μηνυμάτων MQTT

Το MQTT, ή αλλιώς «Message Queuing Telemetry Transport", είναι ένα ανοικτό "publish/subscribe" πρωτόκολλο που αναπτύχθηκε ειδικά για εφαρμογές τηλεμετρίας οπου χρησιμοποιούνται συσκευές περιορισμένων δυνατοτήτων. Λόγω του τρόπου με τον οποίο είναι κατασκευασμένο το MQTT, είναι αρκετά εύκολο να εγκατασταθεί στην πλευρά του πελάτη, από την μεριά ενός αισθητήρα ή ενεργοποιητή. Το MQTT προϋποθέτει ότι υπάρχει ένα υφιστάμενο δίκτυο που προσφέρει μια υπηρεσία επικοινωνίας από σημείο σε σημείο (point-to-point), η οποία είναι προσανατολισμένη στη σύνοδο (session oriented), με αυτόματη τμηματοποίηση δεδομένων και με παράδοση των δεδομένων σε σειρά (όπως το TCP/IP). Η χρήση μιας τέτοιας υποδομής είναι απαραίτητη για την ανταλλαγή μηνυμάτων με το πρωτόκολλο MQTT καθώς αυτή η τεχνολογία δεν ορίζει κανόνες δρομολόγησης και δεν περιλαμβάνει υποδομή δικτύωσης.

Το MQTT πρόκειται για ένα πρωτόκολλο γύρω από το θέμα "topic-based", που υποστηρίζει ιεραρχικά θέματα (topics) μέσω της χρήσης συμβολοσειρών χαρακτήρων. Αυτό διευκολύνει την εγγραφή (subscription) σε πολλά θέματα (topics) σε μια εφαρμογή του Διαδίκτυο των Πραγμάτων. Για παράδειγμα, ένας αισθητήρας θερμοκρασίας στο δωμάτιο "R248" στον όροφο "F2" μπορεί να χρησιμοποιήσει το ιεραρχικό θέμα "wsn/sensor/F2/R248/temperature" για να μεταδώσει τα δεδομένα του. Η κάθετος "/" χρησιμοποιείται για τη διαίρεση του θέματος στα συστατικά του μέρη. Στη συνέχεια, οποιοδήποτε στοιχείο της θεματικής ενότητας μπορεί να αντικατασταθεί με χαρακτήρες μπαλαντέρ. Για παράδειγμα, η συμβολοσειρά "wsn/sensor/F2/+/temperature" μπορεί να χρησιμοποιηθεί για να εγγραφεί μια συσκευή σε δεδομένα που παράγονται από οποιονδήποτε αισθητήρα θερμοκρασίας στον όροφο F2. Σε αυτή την περίπτωση, ο χαρακτήρας μπαλαντέρ "+" χρησιμοποιήθηκε για να αντιπροσωπεύσει κάθε δομή στο τέταρτο επίπεδο του θέματος. [26]

Παρακάτω στην Εικόνα 7 παρουσιάζεται η διαδικασία της δημοσίευσης και της εγγραφής του MQTT, ο διακομιστής ελέγχει το διαμοιρασμό των πληροφοριών και είναι κυρίως υπεύθυνος για τη λήψη όλων των μηνυμάτων από τον εκδότη, το φιλτράρισμά τους, την απόφαση για το ποιος ενδιαφέρεται και, στη συνέχεια, την αποστολή των μηνυμάτων σε όλους τους εγγεγραμμένους πελάτες.



Εικόνα 7: Δημοσίευση/Εγγραφή μέσω MQTT [27]

3.2.1 Αρχιτεκτονική του MQTT

Η τυπική αρχιτεκτονική του MQTT μπορεί να χωριστεί σε δύο κύρια στοιχεία, όπως φαίνεται στην Εικόνα 8. Επιπλέον, κάθε στοιχείο περιγράφεται συνοπτικά παρακάτω.

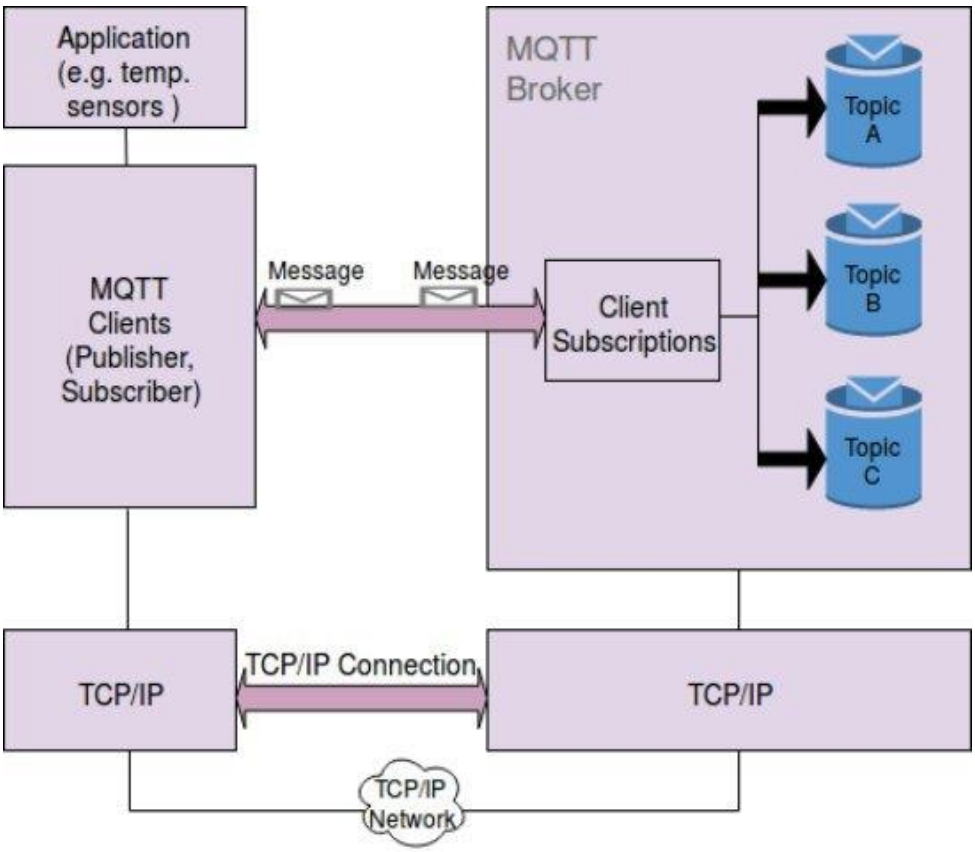
1) Πελάτης: Ο πελάτης μπορεί να είναι είτε εκδότης "publisher" είτε συνδρομητής "subscriber" και αναλαμβάνει πάντα να εγκαθιστά τη σύνδεση δικτύου με τον διακομιστή (broker).

- Ο πελάτης μπορεί να ολοκληρώσει τις παρακάτω εργασίες:
- Να δημοσιεύει μηνύματα για τους ενδιαφερόμενους χρήστες.
- Να εγγραφεί σε ένα ενδιαφερόμενο θέμα για τη λήψη μηνυμάτων.
- Να διαγραφτεί από ένα εγγεγραμμένο θέμα.
- Να αποσυνδεθεί από τον διαμεσολαβητή (Broker).

2) Διαμεσολαβητής/ Διακομιστής: Ο διαμεσολαβητής ελέγχει τη διανομή των πληροφοριών και είναι κυρίως υπεύθυνος για τη λήψη όλων των μηνυμάτων από τους εκδότες, το φιλτράρισμά τους, την απόφαση για το ποιος ενδιαφέρεται και, στη συνέχεια, την αποστολή των μηνυμάτων σε όλους τους εγγεγραμμένους πελάτες ανά θέμα.

Ο διαμεσολαβητής μπορεί να ολοκληρώσει τις παρακάτω εργασίες:

- Να δέχεται αιτήματα πελατών
- Να λαμβάνει δημοσιευμένα μηνύματα από χρήστες
- Να επεξεργάζεται διάφορα αιτήματα όπως η εγγραφή και η απεγράφη χρηστών
- Να προωθεί στους ενδιαφερόμενους πελάτες τα αντίστοιχα δεδομένα έπειτα από τη λήψη μηνυμάτων από τους άλλους χρήστες - πελάτες



Εικόνα 8: Αρχιτεκτονική MQTT [4]

3.2.2 Λειτουργικές Πτυχές του MQTT

Το MQTT υποστηρίζει μια απλή ποιότητα υπηρεσίας (“Quality of Service”, QoS) από άκρο σε άκρο. Το MQTT διακρίνει τρία επίπεδα QoS με βάση την αξιοπιστία με την οποία τα μηνύματα πρέπει να φτάνουν στους παραλήπτες τους. Το πιο βασικό επίπεδο QoS είναι το μηδέν, το οποίο παρέχει μια υπηρεσία παράδοσης με βάση τις καλύτερες συνθήκες, όπου τα μηνύματα καταλήγουν στον προορισμό τους είτε μία φορά είτε καθόλου. Σε αυτό το επίπεδο δεν ορίζεται η αναμετάδοση ή η επιβεβαίωση της λήψης. Μια πιο αξιόπιστη μεταφορά δεδομένων προσφέρεται από το επίπεδο QoS 1, όπου τα μηνύματα με αυτόν τον βαθμό αξιοπιστίας, επαναμεταδίδονται μέχρι οι παραλήπτες να επιβεβαιώσουν την παραλαβή τους. Ως αποτέλεσμα, τα μηνύματα επιπέδου QoS 1 θα φτάσουν σίγουρα, αλλά λόγω των αναμεταδόσεων, μπορεί να φτάσουν στον προορισμό περισσότερες από μία

φορές. Το υψηλότερο επίπεδο ποιότητας υπηρεσιών στο MQTT, το επίπεδο QoS 2, εγγυάται ότι τα μηνύματα θα παραληφθούν και ότι θα παραληφθούν από τον παραλήπτη μόνο μία φορά. Η κάθε εφαρμογή IoT όταν σχεδιάζεται οφείλει να ορίσει το κατάλληλο επίπεδο ποιότητας υπηρεσίας (QoS) για τις ενέργειες του "publish" και "subscribe". Παραδείγματος χάριν, μια εφαρμογή που παρακολουθεί τη θερμοκρασία μπορεί να επιλέξει να χρησιμοποιήσει το επίπεδο QoS 0 για τη μετάδοση (publish) συνήθων και τυπικών αναφορών μετρήσεων και το επίπεδο QoS 1 για την αποστολή προειδοποιητικών μηνυμάτων όταν η θερμοκρασία αυξάνεται πάνω από ένα ορισμένο σημείο. [26]

Το MQTT είναι ένα πρωτόκολλο προσανατολισμένο στη σύνδεση, και το οποίο σημαίνει ότι ένας πελάτης πρέπει να δημιουργήσει μια σύνδεση με τον διαμεσολαβητή πριν ανταλλάξει αιτήματα "publish" και "subscribes" με αυτόν. Ως εκ τούτου, προβλέπεται η ανταλλαγή ενός μηνύματος "CONNECT" για την δημιουργία μιας σύνδεσης και η ανταλλαγή ενός "Client Id" που βοηθά τον διαχειριστή/διαμεσολαβητή (broker) να ξεχωρίζει την κάθε συσκευή/πελάτη που είναι συνδεδεμένη. Όταν ένας πελάτης επανασυνδέεται κατά τη διάρκεια μίας δυσλειτουργίας του δικτύου, ο διακομιστής χρησιμοποιεί το αναγνωριστικό πελάτη, για να διασφαλίσει ότι οι δημοσιεύσεις επιπέδου QoS 1 και 2 παραδίδονται κατάλληλα. Ορίζοντας έναν χρονομετρητή "keep-alive", ο οποίος καθορίζει το μέγιστο χρονικό διάστημα που μπορεί να μεσολαβήσει μεταξύ δύο μηνυμάτων που λαμβάνονται από έναν πελάτη, ο διακομιστής μπορεί να παρακολουθεί αν ο πελάτης ή η σύνδεση είναι ενεργή. Ο πελάτης στέλνει ένα μήνυμα PING στον διακομιστή και ο διακομιστής το επιβεβαιώνει εάν δεν υπάρχουν μηνύματα σχετικά με δεδομένα προς αποστολή εντός αυτού του χρονικού πλαισίου. Ως αποτέλεσμα, ο διακομιστής μπορεί να εντοπίσει αν ο πελάτης ή η σύνδεση του δικτύου αποτυγχάνει χάρη στον χρονομετρητή keep-alive. [26]

Ένα συναφές και αξιοσημείωτο χαρακτηριστικό του MQTT είναι ότι υποστηρίζει την έννοια που συνήθως αναφέρεται ως "Wil". Ένας πελάτης μπορεί να ζητήσει από τον διακομιστή να αποθηκεύσει ένα "Will" θέμα καθώς και ένα "Will" μήνυμα κατά τη στιγμή της σύνδεσης. Οι συνδρομητές θα λάβουν αυτή τη δημοσίευση "Will" από τον διακομιστή μόνο σε περίπτωση που αυτός διαπιστώσει μια ασυνήθιστη απώλεια επικοινωνίας με τον πελάτη. Αυτή η λειτουργικότητα θα μπορούσε να χρησιμοποιηθεί από εφαρμογές για τον εντοπισμό προβλημάτων στις συσκευές και τις συνδέσεις τους. [26]

3.2.3 Προγράμματα διαμεσολαβητών MQTT "Mqtt Brokers"

Το κεντρικό στοιχείο κάθε υλοποίησης MQTT είναι ο διαμεσολαβητής "MQTT Broker". Προσφέρει μια διασύνδεση μεταξύ της υποδομής μιας εταιρείας και των εφαρμογών ή του υλικού. Η εγγραφή, οι καθορισμένες σύνοδοι, τα μηνύματα που χάνονται και η συνολική ασφάλεια, συμπεριλαμβανομένης της εξουσιοδότησης και του ελέγχου ταυτότητας, εμπίπτουν στην αρμοδιότητα των διακομιστών. Οι πιο δημοφιλείς μεσίτες περιλαμβάνονται στον ακόλουθο πίνακα μαζί με τα χαρακτηριστικά και τους περιορισμούς τους. [4]

Πίνακας 2: Χαρακτηριστικά διαμεσολαβητών MQTT [4]

Διαμεσολαβητής (Broker)	Γενικές πληροφορίες	Χαρακτηριστικά	Περιορισμοί
Mosquito	Αποτελεί λειτουργικό ανοιχτού κώδικα και υποστηρίζει την έκδοση MQTT 3.1	• Υποστηρίζει όλες τις επιλογές ποιότητας υπηρεσίας (QoS) • Λειτουργεία Γέφυρας (Bridge mode) • Δυναμικά θέματα • Λειτουργίες Web	• Συσταδοποίηση (Clustering) • Λιγότερη παραμετροποίηση που δεν επιτρέπει ταυτόχρονες συνδέσεις με αυθεντικοποιήση
RSMB (Really Small Message Broker)	Ένας πραγματικά πολύ μικρός διαμεσολαβητής που υποστηρίζει εκδόσεις MQTT 3 και 3.1	• Υποστηρίζει όλες τις επιλογές ποιότητας υπηρεσίας (QoS) • Λειτουργεία Γέφυρας (Bridge mode) • Δυναμικά θέματα	• Ασφάλεια • Λειτουργίες Web • Συσταδοποίηση (Clustering)
MQTT.js	Είναι ένας API διαμεσολαβητής MQTT μεταξύ πελάτη/εξυπηρετητή παραγωγής που συναντάτε σε κώδικα javascript	• Υποστηρίζει όλες τις επιλογές ποιότητας υπηρεσίας (QoS) • Λειτουργεία Γέφυρας (Bridge mode) • Πιστοποιητικά SSL	• Λειτουργεία Γέφυρας (Bridge mode) • Αυθεντικοποιήση • Συσταδοποίηση (Clustering)
HiveMQ	Δίνει τη δυνατότητα στους οργανισμούς να συνδέσουν όλες τις συσκευές και τις υπηρεσίες με ελάχιστη προσπάθεια	• Υποστηρίζει όλες τις επιλογές ποιότητας υπηρεσίας (QoS) • Λειτουργεία Γέφυρας (Bridge mode) • Δυναμικά θέματα • Πιστοποιητικά TLS/SSL • Συσταδοποίηση (Clustering)	• Ανοικτό πρότυπο • Μείωση της επίδοσης εξαιτίας του TLS
VerneMQ	Θα μπορούσε να είναι καλύτερος κατανεμημένος διαμεσολαβητής, υποστηρίζει την έκδοση MQTT 3.1 και 3.1.1.	• Υποστηρίζει όλες τις επιλογές ποιότητας υπηρεσίας (QoS) • Λειτουργεία Γέφυρας (Bridge mode) • Δυναμικά θέματα • Κρυπτογράφηση	• Μείωση της επίδοσης εξαιτίας του TLS

3.2.4 Μειονεκτήματα MQTT

Δεδομένου ότι οι συσκευές που συγκεντρώνουν το σύνολο των πόρων του Διαδικτύου των Πραγμάτων είναι περιορισμένες, το MQTT είναι το πιο συχνά χρησιμοποιούμενο πρωτόκολλο επικοινωνίας. Ωστόσο, έχει αρκετά μειονεκτήματα που πρέπει να αντιμετωπιστούν σε μια IoT εφαρμογή- αυτά παρατίθενται παρακάτω:

1. Λήξη ζωής μηνυμάτων: Επί του παρόντος, το MQTT δεν έχει δυνατότητα λήξης της ζωής των μηνυμάτων. Αυτό σημαίνει ότι αν στείλετε ένα μήνυμα σε έναν διακομιστή και ξεχάσετε να το λάβετε αργότερα ή αν δεν εμφανιστεί ποτέ κανείς για να το ανακτήσει, αυτό θα παραμείνει εκεί επ' αόριστόν. Εξαιτίας αυτού, ο διακομιστής έχει να διαχειριστεί υπερβολικό όγκο μηνυμάτων, γεγονός που επιφέρει μείωση της συνολικής του απόδοσης. [4]
2. Ασφάλεια: Το πρωτόκολλο MQTT προσφέρει ένα σύστημα ονόματος χρήστη και κωδικού πρόσβασης για τον έλεγχο της ταυτότητας, αλλά αρκετοί διακομιστές ενσωματώνουν επιπλέον μηχανισμούς ασφαλείας. Το επίπεδο ασφάλειας στο MQTT εξαρτάται από τη συγκεκριμένη περίπτωση χρήσης και την επιλογή του κατάλληλου διακομιστή. Οι περισσότεροι διαμεσολαβητές προσφέρουν κυρίως ασφάλεια μέσω TLS, ωστόσο, το TLS έχει αξιοσημείωτο αντίκτυπο στην απόδοση, ιδίως όσον αφορά τη χρήση της CPU κατά τη διαδικασία χειραψίας "handshake". [4]
3. Ταξινόμηση: Η αναδιάταξη των μηνυμάτων και η επαναποστολή μηνυμάτων που χάθηκαν κατά τη μετάδοση είναι τα κύρια ζητήματα που αντιμετωπίζει μια αποτελεσματική οργάνωση της μετάδοσης δεδομένων σε ένα σύστημα του Διαδικτύου των Πραγμάτων. Αν και η παράδοση μηνυμάτων μέσω του MQTT μπορεί να διασφαλιστεί, η διατήρηση της διάταξης των μηνυμάτων με τη σωστή σειρά, είναι ένα δύσκολο ζήτημα στο MQTT. [4]
4. Προτεραιότητα: Το MQTT δεν είναι συμβατό με τη λειτουργία που είναι γνωστή ως προτεραιότητα μηνύματος. Κάθε σύστημα που συλλέγει δεδομένα που είναι πιο κρίσιμα από τα δεδομένα των αισθητήρων θερμοκρασίας ή πίεσης πρέπει να είναι άμεσα διαθέσιμα σε όλους τους συνδρομητές. Για παράδειγμα, τα δεδομένα από ένα σύστημα συναγερμού πυρκαγιάς πρέπει να διατίθενται σε όλους τους παραλήπτες πριν από οποιαδήποτε άλλα δεδομένα. Επομένως, προκειμένου να μεταφέρονται τα δεδομένα με τη σωστή σειρά, τα μηνύματα πρέπει να έχουν προτεραιότητα. [4]

Τα παραπάνω μειονεκτήματα της τεχνολογίας που ακούει στο όνομα MQTT είναι πεπερασμένα. Συνεπώς οι μηχανικοί και οι μελετητές, μέσω κατάλληλης έρευνας μπορούν να καταλήξουν σε τεχνάσματα ώστε να ξεπεράσουν τους περιορισμούς αυτούς και να πετύχουν το καλύτερο δυνατό αποτέλεσμα για την εκάστοτε εφαρμογή.

3.3 Από το MQTT στην ολοκληρωμένη διασύνδεση του IoT

Το Διαδίκτυο των Πραγμάτων (IoT) προβλέπεται να παρέχει αυξημένη συνδεσιμότητα για συστήματα, συσκευές και υπηρεσίες, η οποία εκτείνεται πέρα από τις συνδέσεις μεταξύ μηχανών (M2M) και καλύπτει μια ποικιλία εφαρμογών, πρωτοκόλλων και τομέων. Λόγω της φύσης του ανοικτού κώδικα και της ευκολίας χρήσης του, το MQTT είναι ένα καλό πρωτόκολλο για εφαρμογές με πολλούς περιορισμούς, όπως για παράδειγμα το Διαδίκτυο των πραγμάτων, το οποίο έχει ελάχιστη ισχύ, μικρή μνήμη και ικανότητα επεξεργασίας και

περιορισμένο εύρος ζώνης.[4] Αυτή η θεμελιώδης λειτουργικότητα του MQTT δημιουργεί το πλαίσιο για μια πιο ολοκληρωμένη ανάλυση του περιβάλλοντος του Διαδικτύου των Πραγμάτων (IoT), που απαιτεί την ενσωμάτωση πολλών τεχνολογιών για την παροχή μιας ολοκληρωμένης λύσης IoT.

3.4 Έλεγχος και διαχείριση υποδομών IoT με τη χρήση του εργαλείου Node-RED

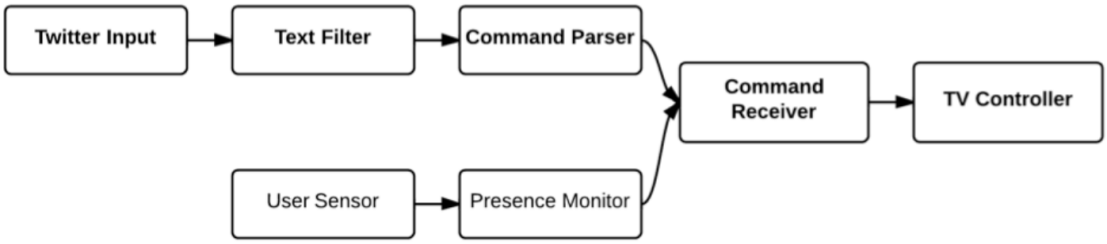
Η ενοποίηση διαδικτυακών υπηρεσιών με αισθητήρες και ενεργοποιητές (σχεδόν) πραγματικού χρόνου είναι απαραίτητη για πολλές τυπικές εφαρμογές του Διαδικτύου των Πραγμάτων, όπως διαδραστικά περιβάλλοντα, εταιρικές αναλύσεις σε πραγματικό χρόνο και οικιακός και βιομηχανικός αυτοματισμός. Αν και οι διαδραστικές εφαρμογές του Διαδικτύου των Πραγμάτων σε πραγματικό χρόνο μπορούν να γίνουν με συμβατικά εργαλεία προγραμματισμού, μπορεί να αποδειχθεί δύσκολο, καθώς οι προγραμματιστές πρέπει να σχεδιάσουν στοιχεία επεξεργασίας δεδομένων, να κατανοήσουν νέα πρωτόκολλα και API και να τα συνδέσουν μεταξύ τους. Προκειμένου να διευκολυνθεί η δημιουργία αυτής της κατηγορίας εφαρμογών IoT, οι επαγγελματίες αναπτύσσουν τώρα νέα εργαλεία και πλατφόρμες. Ορισμένα συστήματα επικεντρώνονται στη δρομολόγηση και τη σύνδεση, ενώ άλλα συνδέουν αντικείμενα και υπηρεσίες χρησιμοποιώντας το παράδειγμα της "δράσης υπό συνθήκη". [28]

Κατ' αυτόν τον τρόπο, διάφορες πλατφόρμες παρέχουν ένα πρότυπο προγραμματισμού ροής δεδομένων/πληροφορίας, στο οποίο τα προγράμματα υπολογιστών περιγράφονται ως κατευθυνόμενοι γράφοι που συνδέουν δίκτυα κόμβων "μαύρου κουτιού" που ανταλλάσσουν δεδομένα κατά μήκος συνδεδεμένων τόξων, ώστε να παρέχουν μεγαλύτερη ευελιξία διατηρώντας την απλότητα χρήσης. Αν και δεν δηλώνεται συχνά γραφικά, αυτή η απλή έννοια αποτελεί τη βάση πολλών οπτικών γλωσσών προγραμματισμού. Οι οπτικές γλώσσες προγραμματισμού ροής δεδομένων "VDFPLs", έχουν βρει χρήση σε ένα ευρύ φάσμα διαφόρων τομέων, όπως η μουσική, τα παιχνίδια, οι βιομηχανικές εφαρμογές και οι παράλληλοι υπολογιστές υψηλής ταχύτητας που εκμεταλλεύονται τους υπολογιστές πολλών πυρήνων. Συγκεκριμένα, τα διαδικτυακά συστήματα ροής δεδομένων WoTKit Processor και Node-RED αρχίζουν να χειρίζονται διαδραστικές καταστάσεις του Διαδικτύου των πραγμάτων όπως αυτές. Το Node-RED είναι μια εργαλειοθήκη για τη δημιουργία ροών δεδομένων σε διακομιστές και συσκευές, ενώ το WoTKit Processor είναι ένα σύστημα πολλαπλών χρηστών για την εκτέλεση προγραμμάτων ροής δεδομένων στο νέφος.[28]

Παρόλο που οι πλατφόρμες ροής δεδομένων έχουν αποδειχτεί επωφελείς από μόνες τους, πολλά σενάρια για το Διαδίκτυο των πραγμάτων απαιτούν το συντονισμό των υπολογιστικών πόρων που βρίσκονται σε διακομιστές και αντικείμενα σε όλο το δίκτυο, συμπεριλαμβανομένων εκείνων που φιλοξενούνται σε διακομιστές στο υπολογιστικό νέφος, πύλες "gateways" και έξυπνες συσκευές στο πεδίο. Οι συσκευές που είναι εξοπλισμένες με αισθητήρες μπορούν να μεταβάλλουν τα ακατέργαστα δεδομένα από τους αισθητήρες για να παραδώσουν μηνύματα όπως "η παρουσία". Οι πύλες είναι σε θέση να συνδυάζουν δεδομένα από διάφορους αισθητήρες και να εκτελούν βασική επεξεργασία δεδομένων- τα συστήματα που βασίζονται στο νέφος μπορούν να δημιουργήσουν συνδέσεις με υπηρεσίες διαδικτύου για να λαμβάνουν ροές από κοινωνικά δίκτυα και κανάλια ειδοποιήσεων όπως SMS και ηλεκτρονικό ταχυδρομείο σε πραγματικό χρόνο. [28]

3.4.1 Αρχιτεκτονική της Ροής Δεδομένων στο IoT

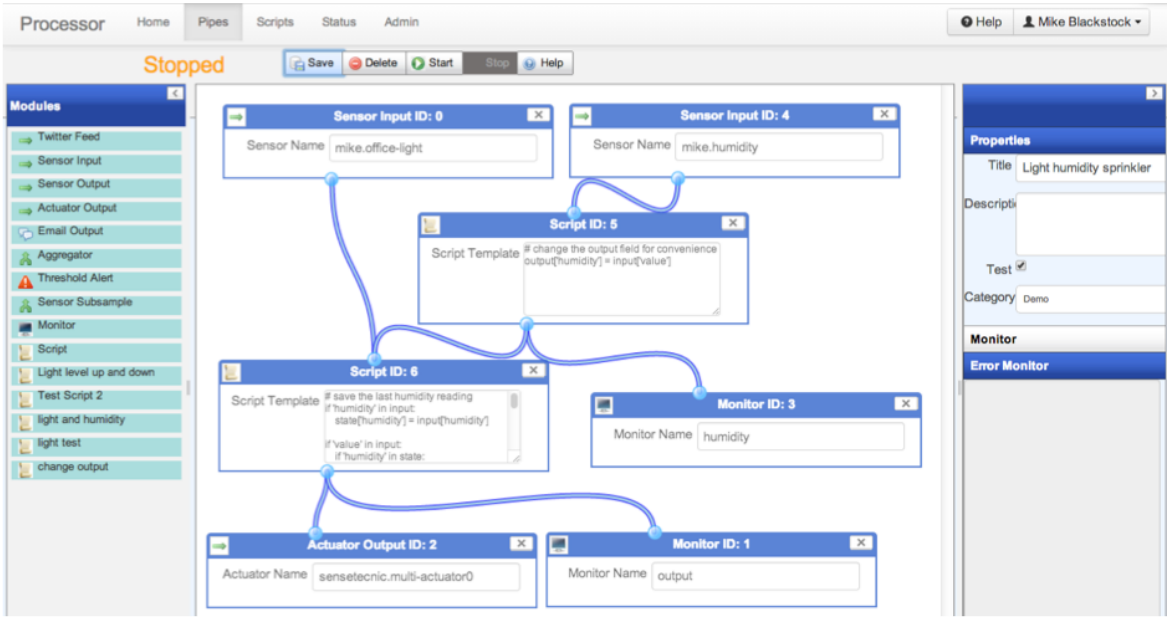
Τα μοντέλα προγραμματισμού βασισμένα στη ροή δεδομένων αποτελούν αντικείμενο έρευνας από τα μέσα της δεκαετίας του 1970, όταν οι ακαδημαϊκοί προσπαθούσαν να αξιοποιήσουν τον εκτενή παραλληλισμό που παρείχαν τα παράλληλα υπολογιστικά συστήματα. Τα παραδοσιακά προγράμματα εκφράζονται συνήθως ως μια ακολουθία εντολών γραμμένων σε κείμενο, με έμφαση στις πράξεις που πρέπει να εκτελεστούν. Αντίθετα, τα προγράμματα ροής δεδομένων αναπαρίστανται καλύτερα οπτικά, με κόμβους που αναπαριστούν εισόδους δεδομένων, εξόδους και λειτουργίες που συνδέονται με τόξα που καθορίζουν τη ροή δεδομένων μεταξύ των στοιχείων, όπως φαίνεται παρακάτω στην Εικόνα 9. Λόγω της ανεξάρτητης φύσης τους, οι διαδικασίες αυτές μπορούν εύκολα να αναπαραχθούν και να ενσωματωθούν σε άλλες εφαρμογές. Επιπλέον, η σημασιολογία της ροής δεδομένων των VDFPLs μπορεί να είναι ευκολότερα κατανοητή για όσους δεν είναι προγραμματιστές.[28]



Εικόνα 9: Απλή ροή δεδομένων που συνδέει το Twitter και ένα αισθητήρα παρουσίας για τον έλεγχο μιας τηλεόρασης. [28]

3.4.2 Webkit processor

Η πλατφόρμα WebKit συνοδεύεται από μια υπηρεσία πολλαπλών χρηστών που ονομάζεται WebKit Processor, η οποία επιτρέπει στους χρήστες να επεξεργάζονται δεδομένα αισθητήρων και να ανταποκρίνονται σε πραγματικό χρόνο σε ενημερώσεις από άλλα εξωτερικά συστήματα και αισθητήρες. Η μηχανή εκτέλεσης του επεξεργαστή WebKit βασίζεται στο Java Concurrency Framework και είναι κατασκευασμένη σε Java χρησιμοποιώντας το Spring Framework. Ένας οπτικός επεξεργαστής ροής δεδομένων με βάση το πρόγραμμα περιήγησης χρησιμεύει ως κύρια διεπαφή, όπως ακριβώς και σε άλλα συστήματα ροής δεδομένων. Οι χρήστες του επεξεργαστή αναπτύσσουν προγράμματα ροής δεδομένων που ονομάζονται pipes, τα οποία αποτελούνται από ενότητες που συνδέονται με καλώδια, αφού εμπνεύστηκαν από το Yahoo Pipes. [28] Παρακάτω στην Εικόνα 10, βλέπουμε το γραφικό περιβάλλον προγραμματισμού του WebKit Processor, οπού οι διεργασίες συσχετίζονται μέσω βελών γνωστών ως ροών.



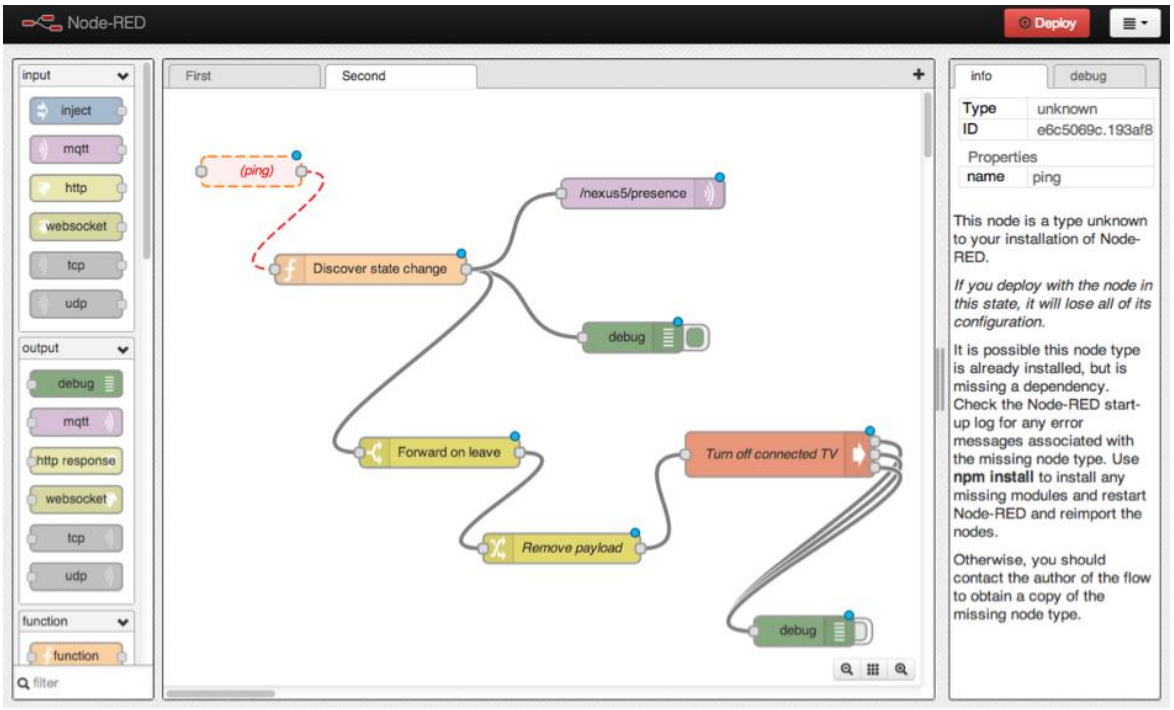
Εικόνα 10: Περιβάλλον προγραμματισμού WebKit Processor [28]

3.4.3 Node-RED

Το Node-RED, που αναπτύχθηκε από ειδικούς της IBM, είναι ένα περιβάλλον ανάπτυξης ανοικτού κώδικα για JavaScript που βασίζεται στο Node.js και είναι το πλέον κατάλληλο για τη δημιουργία εφαρμογών Internet of Things. [29] Βασίζεται στο πλαίσιο NodeJS, το οποίο προσφέρει λύσεις για τη δημιουργία συστημάτων του Διαδικτύου των Πραγμάτων σε όρους κόμβων και ροών και είναι επηρεασμένο από το πρότυπο προγραμματισμού με βάση τη ροή. [30] Πιο συγκεκριμένα, ενσωματώνοντας το υλικό και το λογισμικό, το Node-RED είναι ένα εικονικό περιβάλλον προγραμματισμού βασισμένο σε διεργασίες που δημιουργεί "ροές δεδομένων" από τον αισθητήρα προς το υπολογιστικό νέφος. Η επεξεργασία δεδομένων διευκολύνεται από το γεγονός ότι είναι κατάλληλη για τη συγγραφή διαδικασιών δεδομένων. [29] Οι κόμβοι συνδέονται μεταξύ τους κάθε φορά που χρειάζεται να συνεργαστούν ή να αλληλοεπιδράσουν για την εκτέλεση εργασιών- ως αποτέλεσμα, σχηματίζουν ροές, οι οποίες είναι τα μέρη ενός συστήματος που βρίσκουν νόημα όταν οργανώνουν τη λειτουργικότητα με μια συγκεκριμένη σειρά. [30]

Το Node-RED υποβλήθηκε από την IBM ως έργο ανοικτού κώδικα στο JS Foundation από το 2016 [31] και έκτοτε, κάθε μέρα ανεβαίνουν νέοι κόμβοι βασισμένοι στην τελευταία λέξη της τεχνολογίας από τη δυναμική κοινότητα που το υποστηρίζει. [30] Οι κόμβοι μπορούν να χρησιμοποιηθούν για την εκτέλεση ποικίλων λειτουργιών, όπως η ανάγνωση δεδομένων από μια βάση δεδομένων, η εκτέλεση ενός κώδικα Javascript, η λήψη τροφοδοσιών από έναν λογαριασμό Twitter, η δημιουργία σύνδεσης πρωτοκόλλου MQTT μεταξύ δύο συσκευών και πολλά άλλα. [30] Με το Node-RED, η λογική επεξεργασίας δεδομένων μπορεί να δημιουργηθεί γρήγορα και απλά. Τα επεξεργασμένα δεδομένα μπορούν στη συνέχεια να εμφανίζονται αμέσως ή να αποστέλλονται σε συστήματα ανώτερου επιπέδου (όπως διακομιστές SQL, εταιρικά συστήματα διαχείρισης, κεντρικοί συλλέκτες δεδομένων και υπηρεσίες που βασίζονται στο νέφος) μέσα σε λίγα λεπτά. Επιπλέον, αποτελείται από κόμβους που καθιστούν διαθέσιμες μια ποικιλία λειτουργιών, όπως MQTT broker, αποσφαλμάτωση, Raspberry Pi GPIO, TCP κ.ο.κ. Ο μηχανισμός που χρησιμοποιείται για την αποθήκευση των νέο-δημιουργημένων διεργασιών είναι οι δομές δεδομένων JSON. [29]

Λαμβάνοντας υπόψη την προσαρμοστικότητά του και την ταχύτητα με την οποία μπορεί να δημιουργήσει εφαρμογές, είναι ένα καλό εργαλείο για την κατασκευή πρωτοτύπων. Πριν δαπανηθούν σημαντικοί πόροι στην καινοτομία, η πλατφόρμα Node-RED μπορεί να δώσει στη βιομηχανία τη δυνατότητα να κατασκευάσει και να δοκιμάσει νέα πρωτότυπα που ενσωματώνονται ή συγχωνεύονται με τεχνολογίες που ήδη υπάρχουν. [29] Παρόμοια με οποιαδήποτε άλλη γλώσσα προγραμματισμού ή εργαλείο, το Node-RED επιτρέπει στον προγραμματιστή του συστήματος να δημιουργήσει ένα σύστημα σύμφωνα με τις δικές του προτιμήσεις. Για παράδειγμα, μπορεί να το σχεδιάσει με μια ξεχωριστή ροή που χρησιμοποιεί κόμβους συναρτήσεων κάθε φορά που πρέπει να αρχικοποιηθεί μια παγκόσμια μεταβλητή ή μπορεί να αποφασίσει να χωρίσει τη λογική του συστήματος σε διάφορες ροές και να τις αρχικοποιήσει με πιο κομψούς κόμβους. Αυτή η ευελιξία, από την άλλη πλευρά, έχει την ακούσια συνέπεια να επιτρέπει στον προγραμματιστή να δημιουργήσει ένα σύστημα που είναι λειτουργικό αλλά ακατανόητο. Έτσι, εξαλείφεται η πιθανότητα μικρών σφαλμάτων και αποκλίσεων από την αναμενόμενη συμπεριφορά του συστήματος, τα οποία μπορεί να είναι δύσκολο να εντοπιστούν και να διορθωθούν. [30] Παρακάτω στην Εικόνα 11Εικόνα 10, βλέπουμε το γραφικό περιβάλλον προγραμματισμού του Node-Red Processor, οπού οι διεργασίες συσχετίζονται μέσω βελών γνωστών ως ροών.



Εικόνα 11: Περιβάλλον προγραμματισμού Node-RED [28]

3.5 Βάσεις Δεδομένων στο Διαδίκτυο των Πραγμάτων: Προκλήσεις και λύσεις αντιμετώπισης

Το Διαδίκτυο των Πραγμάτων φανερώνει μια πληθώρα πρωτόγνωρων προκλήσεων, ιδίως για τα συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ). Οι προκλήσεις αυτές περιλαμβάνουν την εκτέλεση των λειτουργιών σε πραγματικό χρόνο, τον χειρισμό της ασφάλειας των δεδομένων και την επεξεργασία τεράστιου όγκου δεδομένων σε πραγματικό χρόνο. Ένα παράδειγμα θα μπορούσε να είναι οι αισθητήρες υγρασίας περιβάλλοντος ή οι αισθητήρες κλιματιστικών που βασίζονται στο Διαδίκτυο των Πραγμάτων (IoT) και είναι τοποθετημένοι σε διάφορες έξυπνες πόλεις. Αυτές οι συσκευές θα μπορούσαν να

μεταδώσουν σημαντικό όγκο δεδομένων σχετικά με τη θερμοκρασία και την υγρασία του περιβάλλοντος σε λιγότερο από ένα λεπτό. Αυτό οφείλεται εν μέρη και στο γεγονός ότι αυξάνεται ραγδαία ο αριθμός των πραγμάτων που συνδέονται στο διαδίκτυο και ενσωματώνονται στα smartphones πολυάριθμοι αισθητήρες και ενεργοποιητές που ανιχνεύουν, υπολογίζουν και επικοινωνούν πολύτιμα δεδομένα που συλλέγονται μέσω του διαδικτύου. Χρησιμοποιώντας αυτό το δίκτυο σε συνδυασμό με μια ποικιλία οικιακών προϊόντων που ενσωματώνουν αισθητήρες, οι άνθρωποι θα είναι σε θέση να δημιουργήσουν μια σειρά από καινοτόμες εφαρμογές που θα προκαλέσουν μια σημαντική επιρροή σε ολόκληρο τον πλανήτη.[32]

3.5.1 Ιδιαιτερότητες Βάσεων Δεδομένων στο IoT

Τα οφέλη του IoT εξαρτώνται σε μεγάλο βαθμό από τις τεράστιες ποσότητες δεδομένων που παράγουν οι εφαρμογές. Αυτό καθιστά ένα σύστημα διαχείρισης βάσεων δεδομένων της εφαρμογής πιο ευάλωτο σε προβλήματα, τα σημαντικότερα από τα οποία είναι η επεκτασιμότητα και η γρήγορη ταχύτητα εισροής δεδομένων.

Δεδομένου ότι τα δεδομένα των συσκευών είναι συχνά κάπως διαφοροποιημένα, η ικανότητα χειρισμού γρήγορα μεταβαλλόμενων δεδομένων είναι ιδιαίτερα χρήσιμη για την αποθήκευση δεδομένων του Διαδικτύου των Πραγμάτων. Η συχνότητα με την οποία μια βάση δεδομένων μπορεί να ανταποκριθεί σε αυτά τα αιτήματα εξαρτάται από διάφορους παράγοντες. Ο όγκος των δεδομένων που παράγονται και οι στρατηγικές κοινής χρήσης και αντιγραφής που χρησιμοποιούνται για τον διαχωρισμό και τη διανομή των δεδομένων εντός μιας συστάδας είναι οι κύριοι παράγοντες που περιορίζουν την επεκτασιμότητα. Η συνολική απόδοση ενός συστήματος βάσης δεδομένων εξαρτάται από διάφορους παράγοντες, όπως το υλικό, τα βασικά συστήματα (όπως η μηχανή αποθήκευσης και η γλώσσα προγραμματισμού), η επεκτασιμότητα, οι ανάγκες συνέπειας, ο πλούτος του μοντέλου δεδομένων και η χρήση ευρετηρίων. Για να επιτευχθούν οι απαιτήσεις χωρίς συμβιβασμούς, θα ήταν επιτακτική ανάγκη η βάση δεδομένων που θα απαιτηθεί για την παροχή υπηρεσιών IoT να σχεδιαστεί με γνώμονα αυτούς τους παράγοντες.[32]

Για την αποτελεσματική διαχείριση των δεδομένων του Διαδικτύου των Πραγμάτων πρέπει να επιλεγεί το σωστό είδος βάσης δεδομένων σε κάθε εφαρμογή. Ωστόσο, καθώς τα περιβάλλοντα IoT διαφέρουν σε μεγάλο βαθμό, η επιλογή της καλύτερης βάσης δεδομένων για εφαρμογές IoT μπορεί να αποδειχθεί δύσκολη πρόκληση. Κατά την επιλογή μιας βάσης δεδομένων για εφαρμογές του Διαδικτύου των Πραγμάτων, υπάρχουν διάφοροι παράγοντες που πρέπει να ληφθούν υπόψη. Η ικανότητα χειρισμού τεράστιων όγκων δεδομένων με κατάλληλες ταχύτητες, η επεκτασιμότητα, το ευέλικτο σχήμα, η ευκολία χρήσης με μια σειρά αναλυτικών εργαλείων, η ασφάλεια και το κόστος είναι οι πιο σημαντικότεροι εξ αυτών. [32]

3.5.2 Οι περιορισμοί των Παραδοσιακών Σχεσιακών Συστημάτων Διαχείρισης Βάσεων Δεδομένων στο IoT

Η υποστήριξη του Διαδικτύου των Πραγμάτων με Σχεσιακά ΣΔΒΔ (RDBMS) είναι μια εναλλακτική λύση- ωστόσο, συνοδεύεται από ένα περιοριστικό χαρακτηριστικό που, με την πάροδο του χρόνου, γίνεται σημαντικό εμπόδιο στην υλοποίηση του συνόλου των δυνατοτήτων που είναι δυνατόν να προκύψουν από κάθε είδους δεδομένα. [32]

Μια σχεσιακή βάση δεδομένων είναι ένας τύπος βάσης δεδομένων που αποθηκεύει πληροφορίες οι οποίες είναι οργανωμένες και διασυνδεδεμένες και επιπλέον είναι οργανωμένες με κατάλληλες συσχετίσεις. Όλες οι εγγραφές που πρόκειται να διατηρηθούν θα οργανωθούν σε μια πιο ολοκληρωμένη δομή, γνωστή ως πίνακας, και στη συνέχεια θα συνδεθούν μεταξύ τους χρησιμοποιώντας ξένα κλειδιά. Ενώ είναι δυνατόν μια ενιαία βάση δεδομένων να έχει έναν ή περισσότερους πίνακες. Η SQL, η οποία είναι μια γλώσσα βελτιστοποιημένη για σχεσιακές βάσεις δεδομένων, καθιστά εύκολη την πρόσβαση στην πληροφορία. Η SQL θεωρείται πλέον το πρότυπό της βιομηχανίας ενώ και άλλοι τύποι συστημάτων διαχείρισης βάσεων δεδομένων χρησιμοποιούν γλώσσες που μοιάζουν αρκετά με τη σύνταξη της SQL. [33]

Ενώ τα Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων αποτελούν τον ακρογωνιαίο λίθο των λύσεων αποθήκευσης δεδομένων, το κατά κανόνα άκαμπτο σχήμα τους και οι διασυνδεδεμένες δομές δεδομένων θέτουν σημαντικά εμπόδια σε μοντέλα IoT. Η ιεραρχική φύση των σχεσιακών βάσεων δεδομένων μπορεί να εμποδίσει τις ρευστές ανταλλαγές δεδομένων που είναι απαραίτητες για τις λειτουργίες του IoT. Αντιπαραβάλλοντας τα Σχεσιακά ΣΔΒΔ με τις αναδυόμενες εναλλακτικές τεχνολογίες NoSQL, είναι προφανές ότι η φύση των εν λόγω συστημάτων χωρίς σχηματοποίηση επιτρέπει πιο ευέλικτο και προσαρμόσιμο χειρισμό δεδομένων, ένα χαρακτηριστικό στο οποίο βασίζονται σε μεγάλο βαθμό οι εφαρμογές IoT λόγω των απρόβλεπτων και διαρκώς εξελισσόμενων απαιτήσεων των δεδομένων τους.

3.5.3 Η PostgreSQL ως Αντικειμενο-σχεσιακή γέφυρα των εφαρμογών IoT

Μια από τις λίγες βάσεις δεδομένων που παρέχουν μια αντικειμενο-σχεσιακή προσέγγιση είναι η PostgreSQL, η οποία δημιουργήθηκε στη γλώσσα προγραμματισμού C. Είναι προσβάσιμη στη συντριπτική πλειονότητα των λειτουργικών συστημάτων και διατίθεται με άδεια χρήσης ανοικτού κώδικα. Το εργαλείο που χρησιμοποιείται για τη διαχείριση της βάσης δεδομένων PostgreSQL είναι το pgAdmin, το οποίο είναι ένα από τα πιο διαδεδομένα εργαλεία το οποίο λειτουργεί ως διαδικτυακή εφαρμογή και μπορεί να ανοίξει σε ένα πρόγραμμα περιήγησης. [33]

Η PostgreSQL ξεχωρίζει ως μια γέφυρα μεταξύ ενός γνώριμου ΣΣΔΒΔ και των νέων απαιτήσεων του IoT. Οι αντικειμενο-σχεσιακές δυνατότητές της εισάγουν ένα επίπεδο ευελιξίας που συχνά απουσιάζει από τις παραδοσιακές σχεσιακές βάσεις δεδομένων. Η επεκτασιμότητα της PostgreSQL και η υποστήριξη ποικίλων τύπων δεδομένων την καθιστούν μια συναρπαστική επιλογή για εφαρμογές IoT που απαιτούν την αυστηρότητα του σχεσιακού μοντέλου σε συνδυασμό με την ευελιξία που χρειάζεται για να φιλοξενήσει πολύπλοκες και ετερογενείς δομές δεδομένων IoT.

3.5.4 Η Δυναμική των NoSQL Βάσεων Δεδομένων στο οικοσύστημα του IoT

Λόγω της ιδιαίτερης οριζόντιας επεκτασιμότητάς τους, τα συστήματα NoSQL είναι ιδανικά για το Διαδίκτυο των Πραγμάτων. Η αποδοτική χρήση της αποθήκευσης δεδομένων σε μνήμη, η οποία θα ήταν πολύ χρήσιμη για την ταχύτητα εγγραφής και την μείωση της καθυστέρησης, είναι ακόμη ένα πολύ δημοφιλές χαρακτηριστικό των βάσεων δεδομένων NoSQL. Τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων φαίνεται να είναι κατά κάποιο τρόπο ανεπαρκή, δεδομένου ότι δεν σχεδιάστηκαν για να επεξεργάζονται τον όγκο των δεδομένων ή τον ρυθμό με τον οποίο παράγονται τα δεδομένα. Ωστόσο, χάρη σε αυτές είναι δυνατή η δημιουργία ενός σχήματος με μια απλή προσθήκη ενός νέου πεδίου σε μια

βάση δεδομένων ή με την προσθήκη μιας μοναδικής ομάδας κλειδιών σε έναν πίνακα. Αυτό κάνει απλή και γρήγορη τη διαχείριση των πληροφοριών από το Διαδίκτυο των Πραγμάτων.[32]

Παρόλα αυτά, καθώς αποθηκεύονται τεράστιες ποσότητες μη δομημένων δεδομένων που αυξάνονται με γοργούς ρυθμούς, ξεπερνώντας τους περιορισμούς αποδοτικότητας και επεκτασιμότητας των άκαμπτων σχεσιακών βάσεων δεδομένων, τα συστήματα βάσεων δεδομένων NoSQL αρχίζουν να γίνονται όλο και πιο διαδεδομένα. Προκύπτει δε το ερώτημα κατά πόσον το σχεσιακό μοντέλο έχει φθάσει στο τέλος του. Από τη μία πλευρά, οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν μορφές κανονικότητας στην ιδέα της πληροφορίας που χωρίζεται σε πίνακες και εγγραφές πεδίων σύμφωνα με τους κανόνες κανονικοποίησης. Από την άλλη πλευρά, οι βάσεις δεδομένων NoSQL συνεχίζουν να παρέχουν μια αξιόπιστη εναλλακτική λύση απόδοσης, η οποία είναι επωφελής για την κανονικοποίηση και τον επανασχεδιασμό κλιμακούμενων εγκαταστάσεων. [32]

Το NoSQL, που σημαίνει "Not only SQL", είναι μια φράση που περιγράφει μια συγκεκριμένη κατηγορία σύγχρονων συστημάτων βάσεων δεδομένων που έχουν σχεδιαστεί για χρήση σε κατανεμημένα πληροφοριακά συστήματα. Αυτές οι βάσεις δεδομένων διαθέτουν συχνά μια αυτόνομα παραχθείσα δομή που δημιουργείται καθώς προστίθενται νέα δεδομένα. Όταν η δομή των δεδομένων δεν είναι άμεσα εμφανής ή είναι ιδιαίτερα διαφοροποιημένη, αυτή η δυνατότητα μπορεί να είναι αρκετά χρήσιμη. Αν και δεν υπάρχουν συνήθως συσχετίσεις στις NoSQL βάσεις δεδομένων, υπάρχουν αρκετές τεχνικές που μπορούν να μιμηθούν τις σχέσεις που είναι γνωστές από τα RDBMS. [33]

Οι βάσεις δεδομένων NoSQL δεν αφορούν μόνο τη διαχείριση τεράστιων ροών δεδομένων- ενσωματώνουν την ευελιξία που απαιτούν τα οικοσυστήματα IoT. Η σχεδίαση τους χωρίς κανονικοποίηση επιτρέπει την προσαρμογή στα δεδομένα που διαχειρίζονται, είτε πρόκειται για ταχέως εξελισσόμενη τηλεμετρία συσκευών είτε για γεγονότα που δημιουργούνται από χρήστες. Σε αντίθεση με τα RDBMS, τα οποία απαιτούν προκαθορισμένα σχήματα και συχνά πολύπλοκες τροποποιήσεις για την προσαρμογή νέων τύπων δεδομένων, οι βάσεις δεδομένων NoSQL είναι κατασκευασμένες για να απορροφούν αυτή την ποικιλομορφία χωρίς την ανάγκη ανατρεπτικών διαρθρωτικών αλλαγών. Αυτή η ευελιξία προσφέρεται για ταχεία καινοτομία και πειραματισμό εντός των εφαρμογών IoT, επιτρέποντας στους προγραμματιστές να επαναλαμβάνουν και να αναπτύσσουν νέα χαρακτηριστικά χωρίς περιορισμούς στο παρασκήνιο.

3.5.5 MongoDB: Μια Αντικειμενοστραφής Βάση δεδομένων και για το IoT

Το σύστημα διαχείρισης βάσεων δεδομένων ανοικτού κώδικα MongoDB, το οποίο κατασκευάστηκε στη γλώσσα προγραμματισμού C++, θεωρείται ευρέως ως ένα από τα πιο χαρακτηριστικά παραδείγματα ενός μη σχεσιακού συστήματος βάσεων δεδομένων. Όλες οι πληροφορίες διατηρούνται με τη μορφή εγγράφων που καταγράφονται σε μορφή JSON ή στη δυαδική έκδοση της μορφής BSON. Η βάση δεδομένων διαθέτει συλλογές, οι οποίες εξυπηρετούν παρόμοιο σκοπό με τους πίνακες. Τα έγγραφα μιας συλλογής διαφέρουν μεταξύ τους, καθώς δεν έχουν καθορισμένη μορφή και συμβατικές στήλες, ούτε ακολουθούν μια κανονικοποίηση. Η πιο συνηθισμένη γλώσσα που χρησιμοποιείται για την υποβολή ερωτημάτων σε μια μη σχεσιακή βάση δεδομένων είναι η JavaScript, ενώ η MongoDB διαθέτει οδηγούς προγραμμάτων για ένα ευρύ φάσμα άλλων γλωσσών. Οι συλλογές δεν συσχετίζονται μεταξύ τους, καθώς δεν υπάρχουν σχέσεις μεταξύ τους,

γεγονός που οδηγεί ενίοτε σε επανάληψη δεδομένων. Η εργασία με μεγάλες ποσότητες δεδομένων καθίσταται εφικτή χάρη στην ταχύτατη λειτουργία της MongoDB. [33]

Με βάση τα παραπάνω, ο ευέλικτος σχεδιασμός της επιτρέπει την ταχεία απορρόφηση και αναζήτηση δεδομένων που είναι τυπικά χαρακτηριστικά των συστημάτων IoT. Η MongoDB ευθυγραμμίζεται πλήρως με την ανάγκη για προσαρμογές των σχημάτων δεδομένων on-the-fly, καθώς οι συσκευές IoT εξελίσσονται και παράγουν ολοένα και πιο πολύπλοκα δεδομένα.

3.5.6 NoSQL Βάσεις Δεδομένων Χρονοσειρών στο οικοσύστημα του IoT

Οι βάσεις δεδομένων χρονοσειρών που βασίζονται στην τεχνολογία NoSQL έχουν αναπτυχθεί ως μια εναλλακτική λύση των παραδοσιακών βάσεων δεδομένων χρονοσειρών (time series databases). Αυτό οφείλεται στο γεγονός ότι τα χαρακτηριστικά που προσφέρονται από τα παραδοσιακά συστήματα δεν είναι πλέον σε θέση να ικανοποιήσουν τις ανάγκες που έχουν αναπτυχθεί. [34]

Μια βάση δεδομένων χρονοσειρών, είναι μια βάση δεδομένων που έχει είτε χρονοσφραγίδα είτε αποτελείται από βελτιστοποιημένα δεδομένα χρονοσειρών. Τα δεδομένα χρονοσειρών είναι απλώς γεγονότα ή μετρήσεις που κατηγοριοποιούνται, αποθηκεύονται και διαχειρίζονται με βάση τον χρόνο. Στατιστικά στοιχεία διακομιστών, παρακολούθηση επιδόσεων συστημάτων, δεδομένα δικτύου, δεδομένα αισθητήρων, συμβάντα, κλικ, συναλλαγές στην αγορά και διάφορα άλλα είδη δεδομένων ανάλυσης μπορεί να εμπίπτουν σε αυτή την κατηγορία.

Οι χρονοσήμανση των μεγεθών, των γεγονότων ή των μετρήσεων διαχειρίζονται κυρίως από τις βάσεις δεδομένων χρονοσειρών. Χαρακτηριστικά όπως η σύνοψη, η σάρωση των δεδομένων μεγάλου εύρους και η διαχείριση του κύκλου ζωής πολλών εγγραφών πληροφορίας καθιστούν τα δεδομένα χρονοσειρών ιδιαίτερα απαιτητικά από άλλα πρότυπα δεδομένων. [33]

Ο ρόλος των Βάσεων Δεδομένων Χρονοσειρών γίνεται όλο και πιο κρίσιμος στο τοπίο του IoT, καθώς προσφέρουν εξειδικευμένες υποδομές για την αποθήκευση και την ανάλυση χρονολογικών δεδομένων. Οι εφαρμογές IoT συχνά παράγουν δεδομένα με χρονοσήμανση που πρέπει να αναλύονται με την πάροδο του χρόνου για να διακρίνουν τάσεις και μοτίβα που είναι απαραίτητα για την προγνωστική συντήρηση, την παρακολούθηση της απόδοσης και την ανάλυση σε πραγματικό χρόνο.

Στην Εικόνα 12, βλέπουμε αρκετές Βάσεις Δεδομένων Χρονοσειρών σε συγκριτική κατάταξη μεταξύ τους σύμφωνα με το DB-Engines, έναν ολοκληρωμένο διαδικτυακό ιστότοπο που κατατάσσει τις βάσεις δεδομένων με βάση τις μηχανές αναζήτησης, τις αναφορές στα μέσα κοινωνικής δικτύωσης, την ποσότητα των προσφερόμενων εργασιών και τη συχνότητα των τεχνικών συζητήσεων.

☐ include secondary database models — 42 systems in ranking, February 2024

Rank Feb 2024	Rank Jan 2024	Rank Feb 2023	DBMS	Database Model	Score Feb 2024	Score Jan 2024	Score Feb 2023
1.	1.	1.	IninfluxDB	Time Series, Multi-model	28.48	+0.92	-0.96
2.	2.	↑ 3.	Prometheus	Time Series	8.61	-0.34	+1.55
3.	3.	↓ 2.	Kdb	Multi-model	8.27	+0.30	+0.97
4.	↑ 5.	↑ 5.	TimescaleDB	Time Series, Multi-model	5.62	+0.37	+1.11
5.	↓ 4.	↓ 4.	Graphite	Time Series	5.08	-0.18	-1.66
6.	6.	↑ 7.	DolphinDB	Time Series, Multi-model	4.06	-0.40	+1.21
7.	7.	↑ 8.	Apache Druid	Multi-model	3.43	-0.14	+0.94
8.	8.	↑ 10.	TDengine	Time Series, Multi-model	3.25	-0.18	+0.84
9.	9.	↓ 6.	RRDtool	Time Series	2.75	-0.05	-0.25
10.	10.	↑ 12.	QuestDB	Time Series, Multi-model	2.69	-0.03	+0.74

Εικόνα 12: Σύγκριση βάσεων δεδομένων χρονοσειρών συμφωνα με το DB-Engines [35]

3.5.7 Προηγμένες NoSQL Βάσεις Δεδομένων Χρονοσειρών: InfluxDB, Prometheus, KDB+

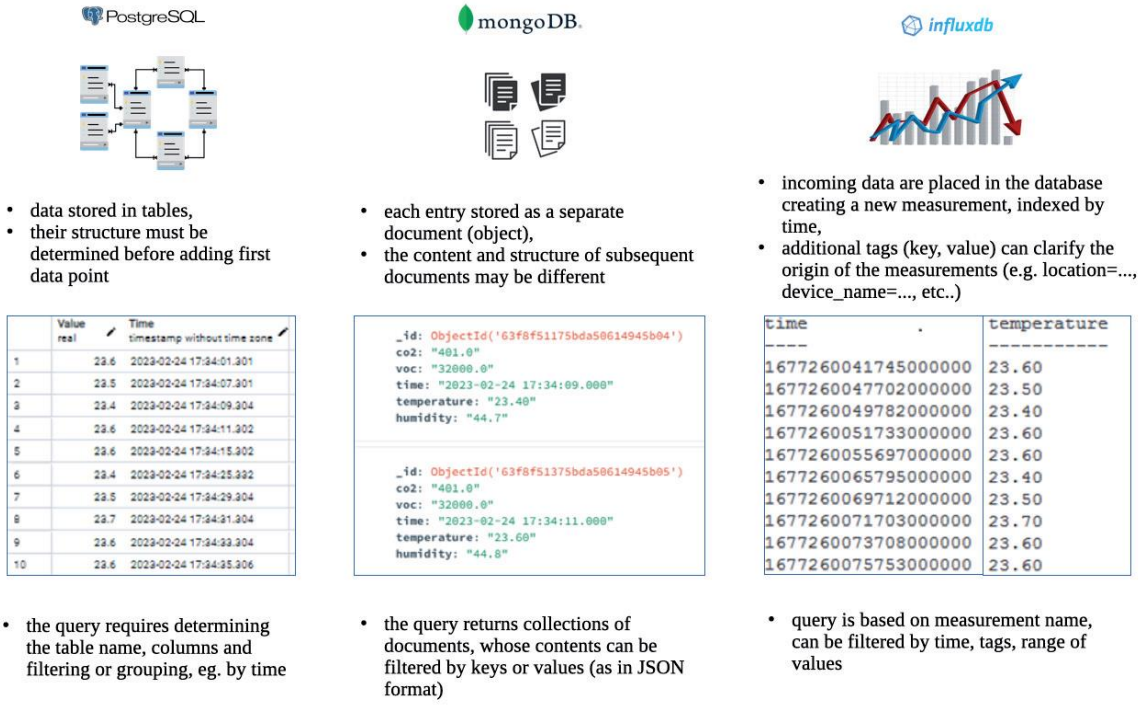
Στη συνέχεια, θα εμβαθύνουμε στις ιδιαιτερότητες τριών κορυφαίων βάσεων δεδομένων χρονοσειρών NoSQL, υπογραμμίζοντας τα μοναδικά χαρακτηριστικά τους και τον τρόπο με τον οποίο ευθυγραμμίζονται με τις απαιτήσεις των σύγχρονων εφαρμογών IoT.

- **Influx DB**: Η InfluxData δημιούργησε το 2013 την Influx DB, μια κατανεμημένη βάση δεδομένων ανοιχτού κώδικα για χρονοσειρές. Βασίζεται στην LevelDB, μια βάση δεδομένων κλειδιών-τιμών, και έχει δημιουργηθεί στη γλώσσα προγραμματισμού Go. Στους πελάτες παρέχεται ένα HTTP api και βιβλιοθήκες για τη συνεργασία με τη βάση δεδομένων πέρα από τη διεπαφή διαχείρισης. Η ικανότητα της InfluxDB να συγκεντρώνει αυτόματα πληροφορίες σε "κουβάδες στιγμής" (buckets) χωρίς ανθρώπινη παρέμβαση έχει αποδείξει ότι είναι το κύριο πλεονέκτημά της. Τα προγράμματα που μπορούν να έχουν πρόσβαση στην InfluxDB περιλαμβάνουν το Grafana, ένα ισχυρό front-end εργαλείο που προσφέρει δυνατότητες οπτικοποίησης δεδομένων χρονοσειρών. Η InfluxDB δεν έχει εσωτερικές ασυμβατότητες και η SQL χρησιμοποιείται για την αναζήτηση μιας δομής δεδομένων που περιλαμβάνει μετρήσεις, σειρές και στοιχεία. Κάθε στοιχείο αποτελείται από διάφορα ζεύγη κλειδιών-τιμών που ονομάζονται fieldset και timestamp. Οι τιμές μπορούν να είναι ακέραιοι αριθμοί 64-bit, κείμενα, Booleans και κινητά σημεία 64-bit. Τα σημεία ευρετηριάζονται με βάση τη στιγμή και το tagset τους. Τα δεδομένα αποθηκεύονται στην InfluxDB μέσω της αξιοποίησης των πρωτοκόλλων TCP, HTTP και UDP. [32]
- **Prometheus:** Η πλατφόρμα ανοικτού κώδικα Prometheus αναπτύχθηκε αρχικά στην Sound Cloud με σκοπό την παρακολούθηση και την ειδοποίηση διαφόρων συστημάτων εντός της εταιρείας. Από την ίδρυσή του το 2012, το Prometheus έχει συγκεντρώσει την υποστήριξη πολλών εταιρειών και οργανισμών, πέρα από μια ιδιαίτερα ενεργή ομάδα σχεδιαστών και χρηστών που συμμετέχουν ενεργά στο έργο. Εξελίχθηκε σε ένα αυτοτελές έργο ανοικτού κώδικα και συντηρείται ανεξάρτητα από οποιαδήποτε εταιρεία μπορεί να συνεισφέρει σε αυτό. [32]

- **KDB+:** Η Kdb+ είναι μια εμπορική σχεσιακή βάση δεδομένων χρονοσειρών με αρχιτεκτονική βασισμένη σε στήλες και δυνατότητες αποθήκευσης στη μνήμη, η οποία δημιουργήθηκε το 2003. Η εκτέλεση αθροιστικών ερωτημάτων, όπως το άθροισμα, η καταμέτρηση, ο μέσος όρος κ.ο.κ., είναι το σημείο όπου η σχεδίαση βάσης δεδομένων με βάση τις στήλες λάμπει περισσότερο. Όταν χρειάζεται να αντληθούν δεδομένα αυτής της βάσης δεδομένων, χρησιμοποιείται η γλώσσα ερωτημάτων Q. Όπως δηλώνουν τα άτομα που είναι υπεύθυνα για την ανάπτυξή της, οι βασικοί σχεδιαστικοί στόχοι της Q είναι η επίτευξη αποδοτικότητας, ταχύτητας και εκφραστικότητας. [32]

Κατά την εμβάθυνση στις βάσεις δεδομένων χρονοσειρών NoSQL, οι InfluxDB, Prometheus και KDB+ αναδεικνύονται σε πρωτοπόρους, προσφέροντας η καθεμία μοναδικά χαρακτηριστικά προσαρμοσμένα για τη διαχείριση δεδομένων χρονοσειρών στο IoT. Η InfluxDB υπερέχει στη συγκέντρωση δεδομένων, η Prometheus στην παρακολούθηση του συστήματος και στις ειδοποιήσεις και η KDB+ στην ανάλυση υψηλών ταχυτήτων μέσω των σχεσιακών χαρακτηριστικών της. Η έρευνα των εν λόγω βάσεων δεδομένων αποκαλύπτει τα ιδιαίτερα πλεονεκτήματά τους στην αντιμετώπιση των ποικίλων και απαιτητικών αναγκών των δεδομένων χρονοσειρών σε περιβάλλοντα IoT.

Παρακάτω στην Εικόνα 13, βλέπουμε την επιγραμματικά την σύγκριση μεταξύ των πιο διαδεδομένων βάσεων δεδομένων ανά κατηγορία: Η PostgreSQL διακρίνεται για τη σχεσιακή της δομή, η MongoDB για την ευελιξία της μορφής "εγγράφων" και η InfluxDB για την ειδίκευση της στις χρονοσειρές δεδομένων.



Εικόνα 13: Σύντομη επισκόπηση των χαρακτηριστικών γνωστών ΣΔΒΔ και παράδειγμα απεικόνισης των αποθηκευμένων δεδομένων. [33]

4 Μηχανική Μάθηση και Τεχνητή Νοημοσύνη

4.1 Εισαγωγή στη Τεχνητή Νοημοσύνη

Τα τελευταία δέκα χρόνια, η τεχνητή νοημοσύνη έχει κερδίσει μεγάλη δημοτικότητα τόσο εντός όσο και εκτός της επιστημονικής κοινότητας. Ο κλάδος της τεχνητής νοημοσύνης "*Artificial Intelligence*" (AI) και των μεθόδων της, της μηχανικής μάθησης "*Machine Learning*" (ML) και της βαθιάς μάθησης "*Deep Learning*" (DL) έχουν διερευνηθεί εκτενώς σε μεγάλο αριθμό άρθρων που έχουν δημοσιευθεί σε τεχνολογικές και μη τεχνολογικές εκδόσεις. Ωστόσο, εξακολουθεί να υπάρχει μεγάλη σύγχυση σχετικά με τους όρους DL, ML και AI. Παρόλο που οι έννοιες συνδέονται στενά μεταξύ τους, δεν μπορούν να χρησιμοποιούνται εναλλάξ. [36]

Ο στόχος της τεχνητής νοημοσύνης (AI) είναι η αυτοματοποίηση των διανοητικών εργασιών που συνήθως εκτελούνται από τον άνθρωπο. Ενώ η μηχανική μάθηση (ML) και η βαθιά μάθηση (DL) είναι συγκεκριμένες τεχνικές και μέθοδοι που μπορούν να εφαρμοστούν για την επίτευξη αυτού του στόχου. [36]

4.1.1 Τι είναι η μηχανική μάθηση κατά τους Samuel, Mitchell και Turing

Ο σκοπός της μηχανικής μάθησης είναι να κάνει προβλέψεις για μελλοντικά γεγονότα ή σενάρια που ο υπολογιστής δεν γνωρίζει εκ των προτέρων.
Το 1959, ο Arthur Samuel δήλωσε ότι: *"η μηχανική μάθηση (ML) είναι το πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν χωρίς να προγραμματίζονται ρητά (without being explicitly programmed)"* (Samuel 1959). Κατέληξε δε στο συμπέρασμα ότι οι περισσότερες από αυτές τις περίπλοκες εργασίες προγραμματισμού θα πρέπει κάποτε να είναι περιττές, δεδομένου ότι οι υπολογιστές θα είναι δυνατόν να προγραμματιστούν και να μαθαίνουν από την εμπειρία τους.

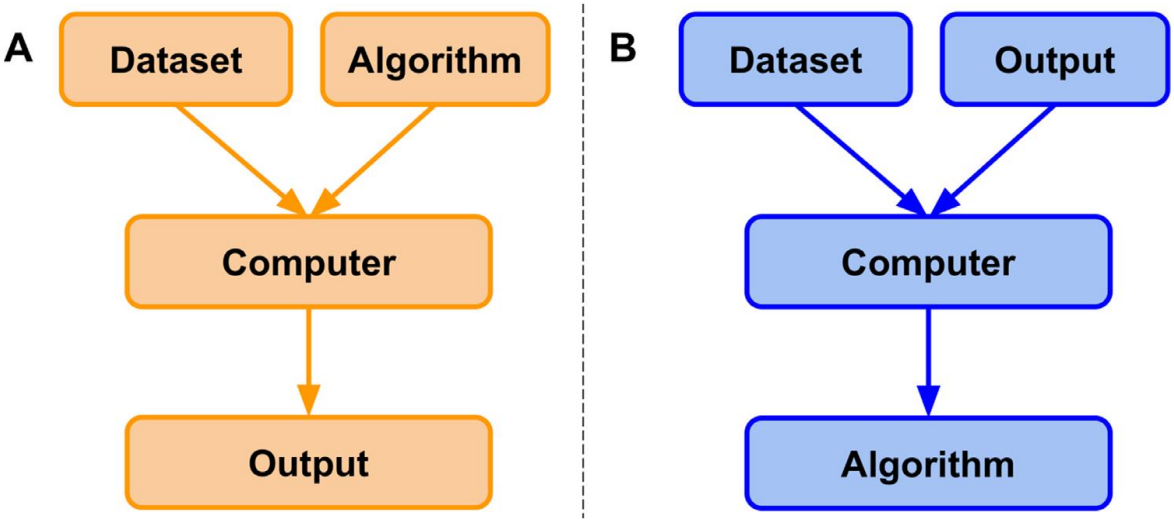
Σύμφωνα με τον Tom M. Mitchell και τον ορισμό του για τη Μηχανική Μάθηση (ML): *"Λέμε ότι ένα πρόγραμμα ενός υπολογιστή μαθαίνει από μια εμπειρία E (experience) σε σχέση με κάποια κατηγορία εργασιών T (tasks) και ένα μέτρο απόδοσης P (performance), αν η απόδοσή του στις εργασίες T, όπως μετράτε από το P, βελτιώνεται με την εμπειρία E."*.

Τέλος στο έργο-ορόσημο, Turing 1950, ο Alan Turing καθιέρωσε ένα θεμελίωμα - απαίτηση αναφοράς για την απόδειξη της ευφυΐας των μηχανών, δηλώνοντας ότι ένας υπολογιστής πρέπει να είναι έξυπνος και να ανταποκρίνεται με τρόπο που να καθιστά αδύνατη τη διάκρισή του από έναν άνθρωπο. [37]

4.1.2 Η Μηχανική Μάθηση έναντι του κλασικού προγραμματισμού

Στον κλασικό προγραμματισμό (Εικόνα 14Α), ο προγραμματιστής γράφει συγκεκριμένες οδηγίες ή κανόνες στον αλγόριθμο για να εκτελέσει επιθυμητές λειτουργίες ή να λύσει συγκεκριμένα προβλήματα. Αυτό απαιτεί από τον προγραμματιστή να κατανοεί εκ των προτέρων τη λογική πίσω από το αποτέλεσμα και να μεταφράσει αυτή τη λογική σε ακριβείς, αλγοριθμικές οδηγίες με το βέλτιστο κώδικα. Αυτή η προσέγγιση λειτουργεί καλά για εφαρμογές με πλήρως τεκμηριωμένα και αμετάβλητα προβλήματα, αλλά μπορεί να είναι περιοριστική σε πιο περίπλοκες ή δυναμικά μεταβλητές καταστάσεις. Από την άλλη πλευρά στη μηχανική μάθηση (Εικόνα 14Β) εκπαιδεύουμε τον υπολογιστή να αναγνωρίζει μόνος του τα πρότυπα και τις σχέσεις μέσα από τα προσφερόμενα δεδομένα, χρησιμοποιώντας τα δεδομένα αυτά ως εκπαιδευτικό υλικό. Κατ' αυτόν τον τρόπο ο αλγόριθμος της Μηχανικής

Μάθησης μαθαίνει να κάνει προβλέψεις ή να λαμβάνει αποφάσεις, προσαρμόζοντας τις αποφάσεις του με βάση τα μοτίβα και τα χαρακτηριστικά που ανακαλύπτει κατά την εκπαίδευση. Αυτό επιτρέπει στην ML να αντιμετωπίζει περίπλοκες και δυναμικά μεταβαλλόμενες καταστάσεις που θα ήταν δύσκολο ή ακόμα και αδύνατο να προγραμματιστούν εκτενώς. [38]



Εικόνα 14: Κλασικός προγραμματισμός (α) έναντι μηχανικής μάθησης (β) [38]

4.1.3 Εμβαθύνοντας στη Μηχανική και τη Βαθιά Μάθηση

Η μηχανική μάθηση βρίσκει ήδη εκτεταμένες εφαρμογές σε διάφορους τομείς, όπως, μεταξύ άλλων, στην ηλεκτρονική αναζήτηση, την τοποθέτηση διαφημίσεων, τη βαθμολόγηση πιστοληπτικής ικανότητας, την πρόβλεψη χρηματιστηριακών αγορών, την ανάλυση αλληλουχιών DNA, την ανάλυση συμπεριφοράς, τα έξυπνα κουπόνια, την ιατρική έρευνα, την πρόγνωση του καιρού, την ανάλυση μεγάλου όγκου δεδομένων και πολλά άλλα. [37]

Η τεχνητή νοημοσύνη είναι ένας αναπτυσσόμενος τομέας που περιλαμβάνει ένα φάσμα μεθόδων και τεχνολογιών που χρησιμοποιούνται για τη δημιουργία ευφυούς συμπεριφοράς και επιτρέπουν στις μηχανές να μιμηθούν την ανθρώπινη αντίληψη και τη γνωστική λειτουργία. Τα νευρωνικά δίκτυα, τα οποία αναλύουν δεδομένα με περίπλοκους τρόπους, αποτελούν τον πυρήνα πολλών συστημάτων τεχνητής νοημοσύνης (AI), συμπεριλαμβανομένης της μηχανικής μάθησης (ML) και της βαθιάς μάθησης (DL). Αυτές οι τεχνολογίες επιτρέπουν την ανάπτυξη εφαρμογών σε ένα ευρύ φάσμα τομέων, συμπεριλαμβανομένων των συστημάτων εμπειρογνωμόνων, της ρομποτικής και άλλων. Με τη χρήση αυτών των μεθόδων, η τεχνητή νοημοσύνη είναι σε θέση να κάνει προβλέψεις ή να κρίνει με βάση την ανάλυση και την ερμηνεία δεδομένων, συχνά με τη χρήση προγνωστικών στατιστικών μοντέλων. Αυτό επιτρέπει σε ρομπότ να εκτελούν εργασίες που κανονικά θα χρειάζονταν ανθρώπινη νοημοσύνη. [39]

4.1.4 Μοντέλα Μηχανικής Μάθησης (supervised, unsupervised, semi-supervised learning)

Καθώς συνεχίζουμε να διερευνούμε τους μηχανισμούς με τους οποίους οι μηχανές μαθαίνουν, είναι σημαντικό να κατανοήσουμε τις διαφορές μεταξύ των κύριων εργασιών στη μηχανική μάθηση, όπως περιεγράφηκαν από τον Bishop το 2006. Οι περισσότερες από αυτές τις εργασίες εμπίπτουν σε μία από τις δύο κατηγορίες: μάθηση με επίβλεψη

(supervised learning) ή μάθηση χωρίς επίβλεψη (unsupervised learning). Καθεμία έχει μια ξεχωριστή λειτουργία στη διδασκαλία των μηχανών και το πώς να επεξεργάζονται πληροφορίες ώστε να καταλήγουν σε συμπεράσματα. Στην επιβλεπόμενη μάθηση (supervised learning), ένα μοντέλο εκπαιδεύεται σε ένα σύνολο δεδομένων που περιέχει τόσο τις εισόδους (x) όσο και τις επιθυμητές εξόδους (y), έτσι ώστε το μοντέλο να μπορεί να προβλέψει αποτελέσματα για προηγουμένως άγνωστα δεδομένα. Αντίθετα, η μάθηση χωρίς επίβλεψη (unsupervised learning) δεν περιλαμβάνει αποτελέσματα που έχουν ήδη επισημανθεί. Αντίθετα, αναζητά δομές ή μοτίβα στα ίδια τα δεδομένα, όπως η μη επιβλεπόμενη ομαδοποίηση (unsupervised clustering), η οποία ομαδοποιεί συγκρίσιμα σημεία δεδομένων. Επιπλέον, είναι ενδιαφέρων ότι η μάθηση με ημιεπίβλεψη (Semi-supervised learning), η οποία χρησιμοποιεί τόσο επισημασμένα όσο και μη επισημασμένα δεδομένα για τη βελτίωση της μάθησης, καλύπτει το κενό μεταξύ αυτών των δύο μεθόδων. Αυτή η υβριδική προσέγγιση προσφέρει μια εφικτή μέθοδο για την αύξηση της ακρίβειας και της αποτελεσματικότητας του μοντέλου και είναι ιδιαίτερα χρήσιμη σε περιπτώσεις όπου η δημιουργία επισημασμένων δεδομένων είναι δύσκολη ή δαπανηρή. Η μάθηση με ημιεπίβλεψη (Semi-supervised learning) είναι κατάλληλη σε ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένων εκείνων όπως η προγνωστική συντήρηση που είναι ιδιαίτερα κρίσιμη, αφού έχει μεγάλο αντίκτυπο στις βιομηχανικές ρυθμίσεις όσον αφορά τη μείωση του κόστους και τη λειτουργική αποδοτικότητα. [40], [41]

Όπως εξέτασε εκτενώς ο Bishop το 2006 και στη συνέχεια αφού διερεύνησε επιπλέον στο πλαίσιο της προγνωστικής συντήρησης, η κατανόηση των διαφορών μεταξύ εποπτευόμενων (supervised), μη εποπτευόμενων (unsupervised) και ημι-εποπτευόμενων (semi-supervised) μαθησιακών εργασιών στο πλαίσιο της μηχανικής μάθησης είναι απαραίτητη για την αξιολόγηση του βάθους και των δυνατοτήτων της τεχνητής νοημοσύνης στις βιομηχανικές εφαρμογές. [40], [41]

4.2 Η υιοθέτηση μεθόδων Τεχνητής Νοημοσύνης στη Βιομηχανία

Όπως συζητείται ευρέως στον επιχειρηματικό κόσμο, η συνεχής αύξηση των δαπανών λειτουργίας και συντήρησης αποτελεί σήμερα ένα σοβαρό πρόβλημα για τον βιομηχανικό κλάδο. Η υιοθέτηση μεθόδων τεχνητής νοημοσύνης στις διαδικασίες λειτουργίας και συντήρησης των εγκαταστάσεων προσφέρει μια πιθανή λύση για το πρόβλημα αυτό. Ο κ. Thomas ανέφερε το 2018 ότι "η εφαρμογή μεθόδων προγνωστικής συντήρησης, θα μπορούσε να οδηγήσει σε εξοικονόμηση κόστους που κυμαίνεται από 15 έως 98% του συνολικού κόστους συντήρησης". Είναι πλέον γνωστό ότι η τεχνητή νοημοσύνη (AI) έχει τις δυνατότητες να μετασχηματίσει σημαντικά τις διαδικασίες συντήρησης.[39]

4.2.1 Ο ρόλος της τεχνικής νοημοσύνης στη μακροζωία των βιομηχανικών συστημάτων και στη μείωση του κόστους λειτουργίας

Οι μέθοδοι εξοικονόμησης κόστους είναι ιδιαίτερα σημαντικοί στον πολυσύνθετο κλάδο της βιομηχανίας, όπου οι δραστηριότητες δημιουργούν τεράστιους όγκους δεδομένων λόγω των αλληλεπιδράσεων μεταξύ πολλαπλών στοιχείων, συμπεριλαμβανομένων φυσικών, άυλων και ανθρώπινων πόρων. Μέσω της χρήσης προ-εκπαιδευμένων αλγορίθμων τεχνητής νοημοσύνης σε υπολογιστικές υποδομές, οι εταιρείες μπορούν να αξιοποιούν αποτελεσματικά τα δεδομένα για τη βελτίωση και την ενημέρωση των μεθόδων συντήρησης. Οι μέθοδοι της τεχνητής νοημοσύνης έχουν κερδίσει ευρεία αποδοχή για την

ικανότητά τους να εμπλουτίζουν τα συστήματα με τη ευφυΐα που απαιτείται για να μαθαίνουν από τις αλλαγές στο περιβάλλον και να προσαρμόζονται σε αυτές με βάση προηγούμενες ενέργειες και εμπειρίες. Η ικανότητα της τεχνητής νοημοσύνης να διαχειρίζεται σύνθετα δεδομένα, να μειώνει την πολυπλοκότητα, να ενισχύει την τεχνογνωσία και να εντοπίζει συσχετίσεις μεταξύ των διεργασιών, αναδεικνύει το όφελος και τη σημασία της στον κλάδο της βιομηχανίας. [42]

Μετά την εμβάθυνση στο ρόλο που διαδραματίζει η τεχνητή νοημοσύνη (AI) στην ενίσχυση των δυνατοτήτων του βιομηχανικού κλάδου -ιδιαίτερα, μέσω της προγνωστικής συντήρησης- είναι σημαντικό να διερευνηθεί και η θέση της τεχνητής νοημοσύνης στο ευρύτερο πλαίσιο των τακτικών συντήρησης στη βιομηχανία. Αξίζει λοιπόν να μελετήσουμε το πώς η τεχνητή νοημοσύνη έχει φέρει επανάσταση στον συγκεκριμένο κλάδο, μεταφέροντας το κέντρο προσοχής από τις έκτακτες συντηρήσεις στα μοντέλα προληπτικής και προγνωστικής συντήρησης. Η στροφή αυτή αναδεικνύει τα ασυναγώνιστα πλεονεκτήματα κόστους και αποδοτικότητας που παρέχει στις σύγχρονες επιχειρήσεις η προγνωστική συντήρηση με βάση την τεχνητή νοημοσύνη, πέραν της ανάδειξης των βελτιώσεων στις διαδικασίες συντήρησης.

4.2.2 Η Μηχανική Μάθηση και τα Μοντέλα συντήρησης στη βιομηχανία

Κάθε στρατηγική βιομηχανικής συντήρησης έχει πλεονεκτήματα και μειονεκτήματα τα οποία καθορίζουν τη διαδικασία εκλογής της καλύτερης δυνατής στρατηγικής διαχείρισης του εξοπλισμού. Οι βιομηχανίες ανέκαθεν βασίζονταν σε δύο βασικές τακτικές για να υποστηρίξουν τις διάφορες φιλοσοφίες λειτουργίας τους: την προληπτική συντήρηση και την έκτακτη συντήρηση ή αλλιώς λειτουργία μέχρι τέλους "run to failure".

Η προγνωστική συντήρηση (Predictive Maintenance, ή PdM), αποτελεί την προτιμώμενη επιλογή για τις σύγχρονες εταιρείες λόγω των βελτιώσεων στις διαδικασίες συντήρησης. Η διορθωτική και η έκτακτη συντήρηση, που σχετίζονται με τη λειτουργία μέχρι τέλους, είναι δύο όροι που αναφέρονται σε παρόμοιες εργασίες που εκτελούνται όταν παρουσιάζεται αποτυχία του εξοπλισμού ή απόδοση κάτω του μέσου όρου. Η προσέγγιση αυτή αξιοποιεί όσο το δυνατόν περισσότερο τον εξοπλισμό καθυστερώντας τη συντήρηση μέχρι να εμφανιστεί κάποιο πρόβλημα. Το πλεονέκτημα αυτής της κατάστασης είναι ότι επιτρέπει στον εξοπλισμό να συνεχίσει να λειτουργεί μέχρι να φτάσει στο σημείο βλάβης του, γεγονός που επιτρέπει τη μέγιστη ωφέλιμη ζωή του εξοπλισμού καθώς και την εξάλειψη του προγραμματισμού συντήρησης. Αυτή η τακτική, ωστόσο, έχει ως αποτέλεσμα απρόβλεπτες διακοπές λειτουργίας, μερικές φορές και κατά τη διάρκεια κρίσιμων περιόδων, οι οποίες μπορούν να επηρεάσουν σοβαρά τη λειτουργία και να αυξήσουν το κόστος επισκευής λόγω πιθανών σοβαρών ζημιών ίσως και ανεπανόρθωτα.

Από την άλλη πλευρά, η προληπτική συντήρηση παρέχει μια πιο προμελετημένη προσέγγιση με τον προγραμματισμό δραστηριοτήτων συντήρησης σε τακτά χρονικά διαστήματα. Αυτό επιτρέπει την υιοθέτηση μιας πιο προληπτικής στρατηγικής. Αυτό το σχέδιο, το οποίο βασίζεται σε υποδείξεις των μηχανικών ή των κατασκευαστών, αντικαθιστά εξαρτήματα σε τακτική βάση και προσπαθεί να αποτρέψει απρογραμμάτιστες βλάβες. Ένα από τα σημαντικότερα πλεονεκτήματα της προληπτικής συντήρησης είναι ότι αυξάνει την αξιοπιστία του εξοπλισμού και παρατείνει ταυτόχρονα τη συνολική διάρκεια ζωής του. Ωστόσο, η μέθοδος αυτή οδηγεί συχνά σε αχρείαστες δαπάνες συντήρησης και

προγραμματισμένες διακοπές λειτουργίας για συντήρηση που μπορεί να μην είναι τεχνικά απαραίτητες, γεγονός που μειώνει τη συνολική αποδοτικότητα της λειτουργίας.

Γίνεται λοιπόν, ολοένα και πιο σαφές ότι η προγνωστική συντήρηση είναι μια προτιμότερη εναλλακτική λύση σε σχέση με τις προαναφερθείσες συμβατικές προσεγγίσεις. Η προγνωστική συντήρηση χρησιμοποιεί εξελιγμένες αναλύσεις και υιοθετεί τακτικές παρακολούθησης δεδομένων σε πραγματικό χρόνο για την πρόβλεψη πιθανών βλαβών του εξοπλισμού πριν αυτές συμβούν. Επιτρέποντας τον ακριβή προγραμματισμό της συντήρησης με βάση τις πραγματικές απαιτήσεις του εξοπλισμού και όχι τα συνήθη διαστήματα, η στρατηγική αυτή μειώνει σημαντικά τον χρόνο διακοπών λειτουργίας. Επιπλέον, η προγνωστική συντήρηση μπορεί να μειώσει το κόστος συντήρησης, συγκεντρώνοντας υλικούς και ανθρώπινους πόρους μόνο σε μηχανήματα που αποκαλύπτουν έγκαιρα προειδοποιητικά σημάδια βλάβης και παρατείνοντας τη διάρκεια ζωής των μηχανημάτων, αποτρέποντας τις πρόωρες βλάβες. [39], [42]

4.2.3 Η προγνωστική συντήρηση ως ένα θεμελιώδες εργαλείο των συγχρόνων βιομηχανιών

Συνοψίζοντας, η προγνωστική συντήρηση είναι η πιο επιτυχημένη και οικονομικά συμφέρουσα στρατηγική για τη συντήρηση, ακόμη και αν οι κλασικές μέθοδοι όπως η μη προγραμματισμένη/διορθωτική και η προληπτική συντήρηση έχουν κάποια αξία στις βιομηχανικές διαδικασίες. Οι βιομηχανίες μπορούν να βρουν μια προσεκτική ισορροπία μεταξύ της μείωσης του χρόνου διακοπής λειτουργίας, της διαχείρισης των δαπανών συντήρησης και της αύξησης της μακροζωίας του εξοπλισμού τους με τη χρήση της προγνωστική συντήρηση. Η μετάβαση προς την προγνωστική συντήρηση αντιπροσωπεύει μια αξιοσημείωτη αλλαγή στις διαδικασίες συντήρησης προς την κατεύθυνση της μεγαλύτερης ευφυΐας και πιο άμεσης ανταπόκρισης, καθιερώνοντας έτσι ένα νέο σημείο αναφοράς για τη επιχειρηματική αριστεία στον βιομηχανικό κλάδο. [39], [42]

Ως αποτέλεσμα λοιπόν του γεγονότος ότι παράγει λιγότερες απρογραμμάτιστες βλάβες στον εξοπλισμό και περισσότερες δυνατότητες εξοικονόμησης χρημάτων στις δαπάνες λειτουργίας και συντήρησης, η προγνωστική συντήρηση είναι ένα στοιχείο που τα στελέχη των βιομηχανικών εγκαταστάσεων θέλουν να συμπεριλάβουν στη στρατηγική των υπηρεσιών τους. [39]

4.3 Η Αρχιτεκτονική της Προγνωστικής Συντήρησης με τη χρήση Τεχνικής Νοημοσύνης

Οι τεχνολογίες που είναι υπεύθυνες για την προγνωστική συντήρηση διακρίνονται από την Deloitte σε πέντε διαφορετικές κατηγορίες, προκειμένου να επιτευχθεί αυτός ο στόχος. Αυτές περιλαμβάνουν το δίκτυο, την ολοκλήρωση, τους αισθητήρες, την επαυξημένη νοημοσύνη και την επαυξημένη συμπεριφορά. Οι έξυπνοι αισθητήρες έχουν σχεδιαστεί για τη συλλογή δεδομένων από το περιβάλλον με τη χρήση εξωτερικών αισθητήρων ή αξιοποιώντας τα στοιχεία των μηχανών με τη χρήση ενσωματωμένων αισθητήρων. Το δίκτυο επιτρέπει την αποθήκευση δεδομένων καθώς και τη μετάδοση πληροφοριών μέσω της χρήσης Bluetooth και Wi-Fi. Η ενσωμάτωση των τεχνολογιών επιτρέπει τη διαχείριση και τη συλλογή δεδομένων μέσω του Διαδικτύου των πραγμάτων (IoT), την επαυξημένη νοημοσύνη για την υποβοήθηση της επεξεργασίας δεδομένων και της ανάλυσης δεδομένων και την επαυξημένη συμπεριφορά για την επίτευξη της εικονικοποίησης, της πληροφορικής

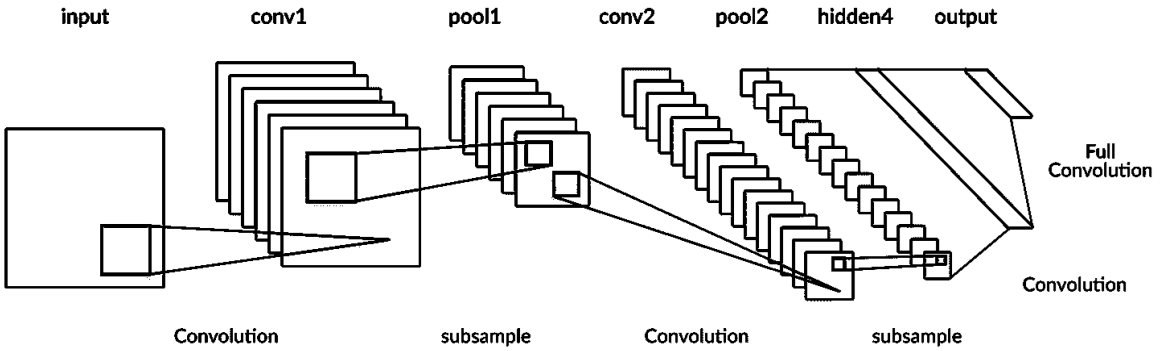
και της πλατφόρμας υπηρεσιών μέσω της χρήσης εφαρμογών, καθώς και για την έκδοση αιτημάτων για την υποβοήθηση του χειριστή. [43]

4.4 Η Βαθιά Μάθηση ως εργαλείο της προγνωστική συντήρησης

Σε αυτόν τον συνεχώς εξελισσόμενο κόσμο της βιομηχανικής συντήρησης, η εισαγωγή της τεχνολογίας της βαθιάς μάθησης (DL) έχει φέρει επαναστατικούς τρόπους για την προγνωστική συντήρηση . Οι προσεγγίσεις αυτές κάνουν χρήση πολύπλοκων δεδομένων προκειμένου να προβλέπουν και να αποφεύγουν προβλήματα στον εξοπλισμό. Δύο κρίσιμα μοντέλα που έχουν βελτιώσει σημαντικά την ικανότητα ανάλυσης και κατανόησης τεράστιων και περίπλοκων συνόλων δεδομένων είναι τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks - CNN) και τα δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long Short-Term Memory - LSTM). Τα δίκτυα αυτά βρίσκονται στο επίκεντρο αυτής της επανάστασης.

4.4.1 Συνελικτικά Νευρωνικά Δίκτυα (CNN)

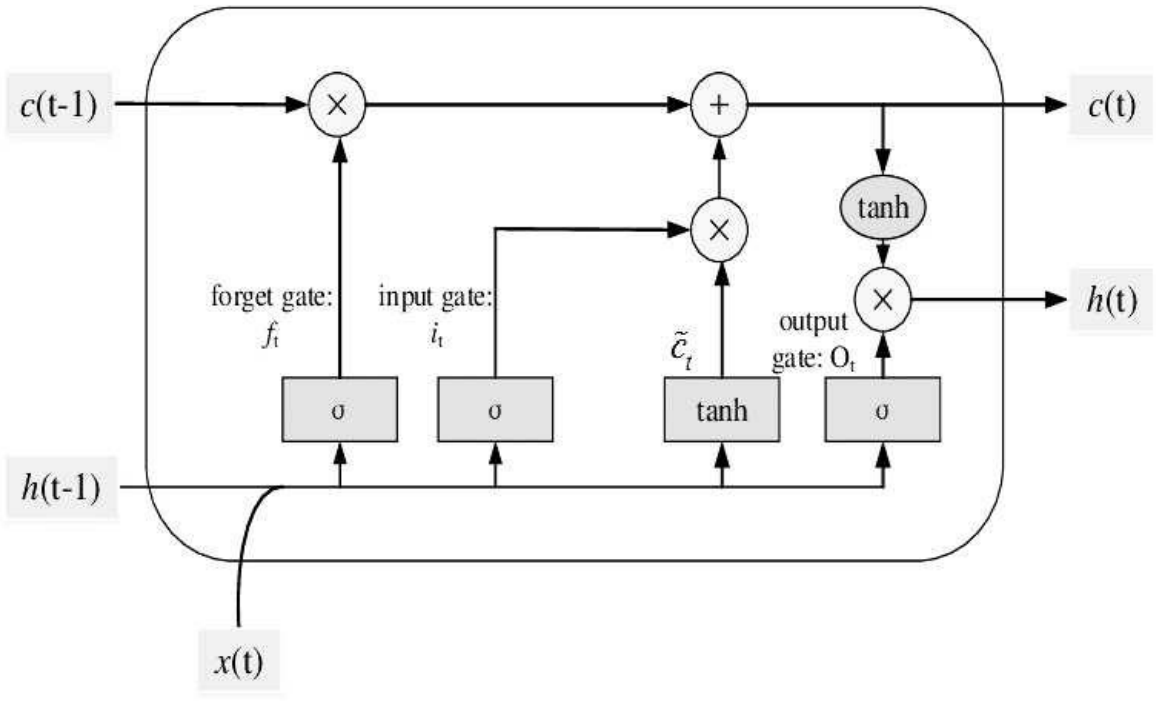
Τα CNN είναι ένας τύπος βαθιού νευρωνικού δικτύου που ειδικεύεται στην επεξεργασία δεδομένων με δομή που μοιάζει με πλέγμα, όπως οι εικόνες. Τα CNNs είναι πολύ καλά στον εντοπισμό μοτίβων για τη διάκριση μεταξύ διαφορετικών οπτικών πτυχών, καθώς ο σχεδιασμός τους διαμορφώνεται σύμφωνα με τον τρόπο που είναι δομημένος ο οπτικός φλοιός των ζώων. Ένα επίπεδο εισόδου, πολλά επίπεδα συνελεκτικής επεξεργασίας, επίπεδα συγκέντρωσης και ένα επίπεδο εξόδου είναι τα τυπικά συστατικά ενός CNN. Το επίπεδο εξόδου μπορεί να συνδεθεί με πλήρως συνδεδεμένα επίπεδα ή με συναρτήσεις ταξινόμησης, όπως η σιγμοειδής συνάρτηση. Η γραμμική πράξη μεταξύ πινάκων εφαρμόζεται από τα στρώματα συμβολής χρησιμοποιώντας μια μαθηματική διαδικασία γνωστή ως συνέλιξη. Μειώνοντας τον αριθμό των παραμέτρων, η προσέγγιση αυτή βελτιώνει την ταχύτητα επεξεργασίας του δικτύου για πολυδιάστατες ή μεγάλης κλίμακας εικόνες. Λόγω των εξαιρετικών επιδόσεών τους στον χειρισμό δεδομένων πολυμέσων, τα CNN έχουν αποδειχθεί ιδιαίτερα χρήσιμα και στους τομείς της επεξεργασίας σήματος. Αυτό αποδεικνύεται από τον μοναδικό σχεδιασμό τους, ο οποίος περιλαμβάνει την αρχιτεκτονική LeNet-5 που παρουσίασε ο LeCun στις αρχές της δεκαετίας του 1980. Στο Σχήμα 1 απεικονίζεται η συνολική αρχιτεκτονική των CNN, τα οποία μειώνουν αποτελεσματικά το περιθώριο σφάλματος με τη χρήση τεχνικών όπως η Ανάστροφη Διάδοση (backpropagation). [44]



Εικόνα 15: Δομή Συνελικτικών Νευρωνικών Δικτύων [44]

4.4.2 Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης (LSTM Networks)

Τα Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης, τα οποία αναφέρονται συχνά ως Δίκτυα LSTM, είναι μια υποκατηγορία των Αναδρομικών Νευρωνικών Δικτύων (RNN) που αναπτύχθηκαν συγκεκριμένα για την αντιμετώπιση προβλημάτων πρόβλεψης ακολουθιών μακρινών εξαρτήσεων. Λόγω του προβλήματος της φθίνουσας διαβάθμισης, τα κανονικά RNN δυσκολεύονται να διδαχτούν απο μακροχρόνιες εξαρτήσεις. Αντίθετα, τα LSTM δίκτυα διαθέτουν κύτταρα μνήμης που τους επιτρέπουν να αποθηκεύουν πληροφορίες για εκτεταμένες χρονικές περιόδους. Εξαιτίας αυτής της ιδιότητας, τα LSTM είναι ιδιαίτερα κατάλληλα για εργασίες στις οποίες ο συγχρονισμός και η αλληλουχία των γεγονότων είναι ζωτικής σημασίας, όπως η γλωσσική μοντελοποίηση, η αναγνώριση φωνής και κυρίως η προγνωστική συντήρηση, η οποία εξαρτάται σε μεγάλο βαθμό από δεδομένα αισθητήρων που συλλέγονται με την πάροδο του χρόνου. Το ζήτημα της εξαφάνισης των διαβαθμίσεων επιλύεται από τη δομή LSTM με την προσθήκη πυλών που ελέγχουν τη ροή πληροφοριών. Αυτές οι πύλες καθορίζουν ποια δεδομένα θα πρέπει να διατηρηθούν ή να απορριφθούν κατά την επεξεργασία των δεδομένων, εξασφαλίζοντας ότι η μάθηση μακράς ακολουθίας και η διατήρηση της μνήμης μπορούν να επιτευχθούν από τα LSTM χωρίς συμβιβασμούς στην αποδοτικότητα. Στο Σχήμα 2 παρουσιάζεται η διάταξη ενός μοντέλου δικτύου LSTM, συμπεριλαμβανομένων των πυλών εισόδου (input), λήθης (forget) και εξόδου (output), καθώς και των λειτουργιών τους για τη διατήρηση της ροής της πληροφορίας με την πάροδο του χρόνου. [44]



Εικόνα 16: Δομή Νευρώνων Δικτύων Μακράς Βραχυπρόθεσμης Μνήμης [44]

4.4.3 Προσέγγιση προγνωστικής συντήρησης με Υβριδικά Συνελικτικά Νευρωνικά Δίκτυα (CNN) και Μακράς Βραχυπρόθεσμης Μνήμης (LSTM)

Αξιοποιώντας τα διακριτά πλεονεκτήματα των CNN και των LSTM, το υβριδικό μοντέλο CNN-LSTM που προτάθηκε από τον κ. Nasser την ομάδα του, επιδιώκει να βελτιώσει τις τακτικές προγνωστικής συντήρησης. Μέσω του συνδυασμού της εξειδίκευσης των CNNs

στον χειρισμό χωρικών πληροφοριών και της ικανότητας των LSTMs να αναλύουν ακολουθιακά δεδομένα, το συγκεκριμένο μοντέλο παρέχει μια ενδελεχή μέθοδο για την αξιολόγηση των περίπλοκων δεδομένων που σχετίζονται με την παρακολούθηση της υγείας του εξοπλισμού. Ενώ το στοιχείο του LSTM αναλύει τις χρονικές συνδέσεις εντός των δεδομένων για να παρέχει ακριβείς προβλέψεις μελλοντικών βλαβών του εξοπλισμού, το στοιχείο του CNN συλλέγει αποτελεσματικά χαρακτηριστικά από ακατέργαστα δεδομένα, όπως εικόνες ή εισόδους αισθητήρων. Αυτή η υβριδική τεχνική μειώνει την πολυπλοκότητα του μοντέλου και ενισχύει την ακρίβεια πρόβλεψης, καθιστώντας το πιο αποτελεσματικό για πρακτικές εφαρμογές προγνωστικής συντήρησης. [44]

4.4.4 Χρήση αυτοκωδικοποιητών σε δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM)

Μετά την ανάλυση των δυνατοτήτων των δικτύων CNN και LSTM, οι αυτοκωδικοποιητές (autoencoders) αποτελούν μια ακόμη επαναστατική πρόοδο στην προγνωστική συντήρηση με βάση τη βαθιά μάθηση. Όταν οι αυτοκωδικοποιητές χρησιμοποιούνται μαζί με δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM), ανοίγουν νέες δυνατότητες για τον εντοπισμό βλαβών και την παρακολούθηση της κατάστασης του βιομηχανικού εξοπλισμού. Η προσέγγιση αυτή χρησιμοποιεί ένα πλαίσιο βασισμένο σε αυτόκωδικοποιητή για την κατηγοριοποίηση των δεδομένων μηχανών και αισθητήρων που συλλέγονται στο πεδίο σε μια σειρά από ετικέτες που σχετίζονται με την κατάσταση, γεγονός που διευκολύνει την κατανόηση της κατάστασης του εξοπλισμού με πιο εξελιγμένο τρόπο. Οι μεγάλες και περίπλοκες πληροφορίες από βιομηχανικές διεργασίες μπορούν να μετατραπούν σε ουσιαστικές γνώσεις μέσω της χρήσης αυτοκωδικοποιητών, γεγονός που καθιστά δυνατή την ακριβή εκτίμηση της εναπομένουσας ωφέλιμης ζωής (RUL) του εξοπλισμού.

Το βασικό στοιχείο αυτής της στρατηγικής είναι η ικανότητά της να χρησιμοποιεί την κωδικοποίηση και την αποκωδικοποίηση για την καταγραφή και την ανακατασκευή των προτύπων λειτουργίας των μηχανημάτων, γεγονός που επιτρέπει την έγκαιρη ανίχνευση ανωμαλιών και δυσλειτουργιών. Μια στρατηγική αυτού του είδους είναι απαραίτητη για τη μετάβαση από τα συμβατικά προγράμματα προληπτικής συντήρησης σε ένα καθεστώς συντήρησης που είναι πιο δυναμικό και προγνωστικό. Η προσέγγιση αυτή συμβάλλει σημαντικά για κάθε βιομηχανία, χρησιμοποιώντας αυτόματους κωδικοποιητές LSTM για την πρόβλεψη της υπολειπόμενης διάρκειας ζωής του εξοπλισμού με μεγαλύτερη ακρίβεια μέσω της ανάλυσης δεδομένων που συλλέγονται από πραγματικές λειτουργίες. Μια ολοκληρωμένη τεχνική για την αξιολόγηση της λειτουργικής κατάστασης του εξοπλισμού παραγωγής αναδεικνύεται από την ενσωμάτωση των αυτοκωδικοποιητών με δίκτυα LSTM. Η βαθιά μάθηση χρησιμοποιείται για την ανίχνευση ανωμαλιών, η οποία αντιστοιχεί άμεσα σε διάφορες προβλέψεις για την περίοδο ζωής του εξοπλισμού.

Επιπλέον, η τεχνική του αυτόματου κωδικοποιητή ενισχύεται με την πρακτική χρήση των συνεπτυγμένων νευρωνικών δικτύων (CNN) για την ανίχνευση ελαττωμάτων σε δεδομένα αισθητήρων πολλαπλών καναλιών, λόγω των μειωμένων αναγκών επεξεργασίας και της εγγενούς αποδοτικότητάς τους. Σε συνδυασμό, οι τεχνολογίες αυτές προσφέρουν μια ολοκληρωμένη προσέγγιση για την προγνωστική συντήρηση, καλύπτοντας τη ζήτηση για καθοδηγούμενες από δεδομένα, μη καταστροφικές γνώσεις σχετικά με την κατάσταση του εξοπλισμού. Εκτός από την απόδειξη της αποτελεσματικότητας της βαθιάς μάθησης σε

βιομηχανικές εφαρμογές, η πρωτοποριακή χρήση των CNN, των LSTM και των αυτοκωδικοποιητών στην προγνωστική συντήρηση αποτελεί σημαντική πρόοδο στη βελτίωση της λειτουργικής αξιοπιστίας και αποδοτικότητας. [45]

4.5 Αντιμετωπίζοντας τις προκλήσεις της βιομηχανίας με νέους κανόνες

Συμπερασματικά, ο συνδυασμός των δομών CNN, LSTM και υβριδικών δομών CNN-LSTM, σε συνδυασμό με επιπρόσθετες μεθόδους όπως οι αυτοκωδικοποιητές, σηματοδοτεί μια αξιοσημείωτη πρόοδο στον τομέα της προγνωστικής συντήρησης. Αυτές οι τεχνολογίες βαθιάς μάθησης παρέχουν ισχυρά μέσα για την ανάλυση περίπλοκων, πολυδιάστατων δεδομένων, τα οποία χρησιμεύουν ως βάση για μοντέλα πρόβλεψης ικανά να προβλέπουν βλάβες εξοπλισμού με πρωτοφανή ακρίβεια. Αξιοποιώντας αυτές τις τεχνολογίες, οι βιομηχανίες μπορούν να βελτιώσουν σημαντικά τα πλάνα συντήρησής τους, μειώνοντας τον χρόνο διακοπής λειτουργίας και αυξάνοντας τη διάρκεια ζωής σημαντικών μηχανημάτων. Ειδικότερα, η καινοτόμος χρήση των αυτοκωδικοποιητών σε συνδυασμό με τα δίκτυα LSTM διευρύνει ακόμη περισσότερο τα όρια της προγνωστικής συντήρησης, επιτρέποντας στις βιομηχανίες όχι μόνο να προβλέπουν και να σχεδιάζουν τις μελλοντικές ανάγκες συντήρησης, αλλά και να τις προβλέπουν και να ανταποκρίνονται σε αυτές. Όλα αυτά τα στοιχεία αλλάζουν εντελώς το παιχνίδι και θέτουν ένα εντελώς νέο πρότυπο για τη επιχειρησιακή αριστεία γύρω από τον χώρο της προγνωστικής συντήρησης. [44], [45]

5 Σχεδιασμός του προτεινόμενου συστήματος

5.1 Γενική Αρχιτεκτονική

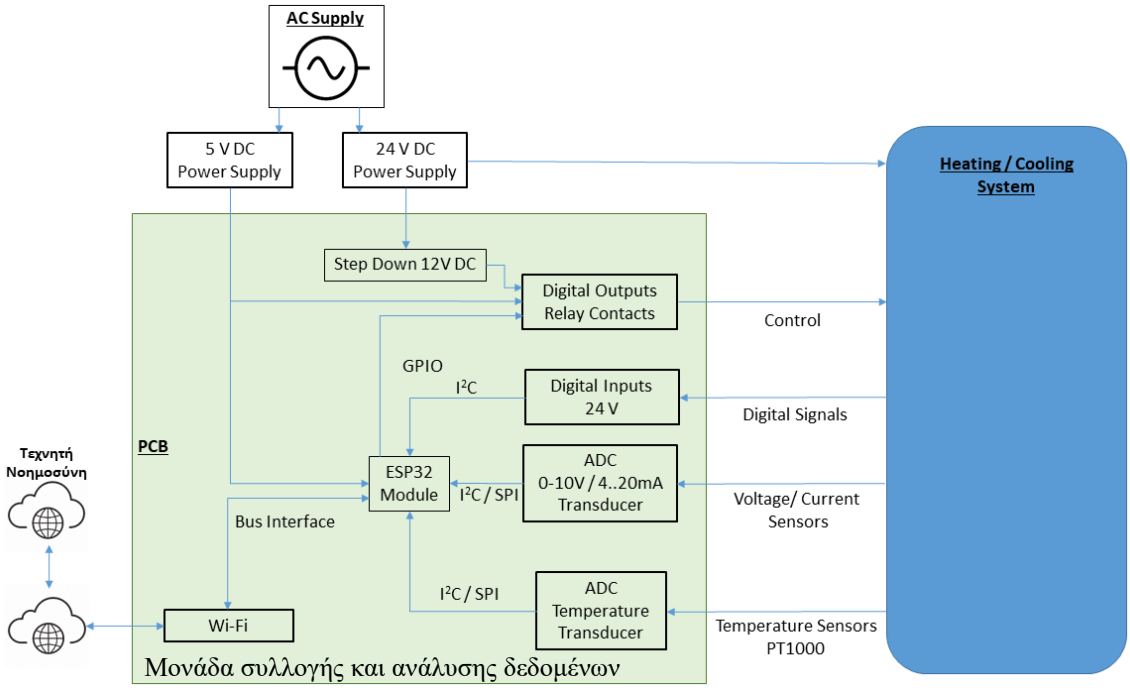
Ένα σύστημα το οποίο συλλέγει στοιχεία από ένα φυσικό περιβάλλον όπως ένα σύστημα ψύξης/θέρμανσης και αποθηκεύει τα δεδομένα αυτά σε μια υποδομή υπολογιστικού νέφους για περεταίρω ανάλυση, αποτελείται από το υλικό πεδίου (field hardware) και την υπολογιστική υποδομή που μπορεί να φιλοξενηθεί σε μια υποδομή cloud. Η συγκεκριμένη υλοποίηση απαιτεί δικτύωση και όταν θέλουμε να παρακολουθήσουμε περισσότερο από ένα σύστημα, χρειαζόμαστε μια υλοποίηση του μοντέλου Client-Server. Αν εξετάσουμε το πρότυπο του Διαδικτύου των Πραγμάτων, ο όρος υλικό πεδίου συσχετίζεται τόσο με το πράγμα “thing” όσο και με τη γέφυρα μεταξύ αντικειμένων και διαδικτύου.

Ως πράγμα, αντικείμενο ή “thing” θεωρούμε την συσκευή που θέλουμε να παρακολουθήσουμε και στη προκειμένη περίπτωση ένα σύστημα θέρμανσης/ψύξης. Τα λοιπά περιφερειακά ηλεκτρονικά στοιχεία όπως οι αισθητήρες και μια στοιχειώδης ηλεκτρονική διάταξη (πλακέτα) αποτελούν τη μονάδα συλλογής δεδομένων, δηλαδή τη γέφυρα του συστήματος ανάμεσα στον φυσικό και τον ψηφιακό κόσμο.

Στην άλλη πλευρά του συστήματος υπάρχει μια απαραίτητη υπολογιστική υποδομή η οποία στη συγκεκριμένη διπλωματική εργασία φιλοξενείται σε μια υπηρεσία νέφους, κάτι το οποίο συνηθίζεται σε ερευνητικές και εμπορικές εφαρμογές. Στη δομή υπολογιστικού νέφους υλοποιούνται οι λειτουργίες οι οποίες εξασφαλίζουν την συνεχή εισροή και αποθήκευση δεδομένων από το πεδίο. Τα δεδομένα φτάνουν στον εξυπηρετητή μέσω κατάλληλης δικτυακής υποδομής και αποθηκεύονται σε βάση δεδομένων για περεταίρω ανάλυση.

Το μεγαλύτερο εφόδιο των συστημάτων του Διαδικτύου των πραγμάτων είναι τα δεδομένα τους. Έτσι και στη συγκεκριμένη εφαρμογή, το σημαντικότερο μέρος μιας διάταξης που αναλαμβάνει την παρακολούθηση ενός συστήματος ψύξης/ θέρμανσης, είναι τα ίδια τα δεδομένα που συλλέγονται από το φυσικό περιβάλλον και τα μηχανήματα με την βοήθεια των αισθητήρων. Τα φυσικά μεγέθη αναλύονται με την ενσωμάτωση της τεχνητής νοημοσύνης και προκύπτουν συσχετίσεις εξ αυτών. Ο κατάλληλος αλγόριθμος τεχνικής νοημοσύνης θα μπορέσει να βρει τα μοτίβα εκείνα τα οποία επαληθεύουν την συμπεριφορά του εκάστοτε συστήματος ψύξης/θέρμανσης (έξοδοι = y) ως προς τις συνθήκες λειτουργίας και τις παραμέτρους του συστήματος (είσοδοι = x) λαμβάνοντας υπόψη δεδομένα θερμοκρασίας, πίεσης, υγρασίας, αλλά λοιπά μεγέθη που μπορούν να δώσουν συμπεράσματα για την συμπεριφορά και καταπόνηση των μηχανημάτων. Η κύρια ιδέα της εφαρμογής της τεχνικής νοημοσύνης βασίζεται στις αναρίθμητες χρήσιμες πληροφορίες που μπορεί να δώσει αυτή για την υγεία του μηχανήματος με σκοπό την αποφυγή βλαβών και την ανάγκη απρογραμμάτιστων συντηρήσεων. Η προσέγγιση αυτή κινείται γύρω από το ότι ένα σύστημα έχει παρατηρηθεί και μελετηθεί σε υγιή κατάσταση για μεγάλο χρονικό διάστημα, έχει αποκωδικοποιηθεί η συμπεριφορά του και έχει ενσωματωθεί αυτή σε ένα μοντέλο αλγορίθμου το οποίο είναι πλέον ικανό να προβλέψει την μελλοντική συμπεριφορά ενός μηχανήματος και να διακρίνει πιθανές βλάβες σε βάθος χρόνου.

Στο παρακάτω σχεδιάγραμμα στην Εικόνα 17 απεικονίζεται η αρχιτεκτονική του συστήματος.



Εικόνα 17: Γενική αρχιτεκτονική του συστήματος

5.2 Μονάδα συλλογής και ανάλυσης δεδομένων σε τοπικό επίπεδο

Σε κάθε σύστημα του Διαδικτύου των πραγμάτων, μια ηλεκτρονική διάταξη αναλαμβάνει τη λήψη μετρήσεων μέσω των αισθητήρων, τη συλλογή των δεδομένων και την αποστολή τους στο υπολογιστικό νέφος. Αν έχουμε υλοποίηση του μοντέλου client – server, τότε οι μονάδες συλλογής δεδομένων είναι τα τερματικά των συστήματος.

Η σχεδίαση της μονάδας αυτής περιλαμβάνει την εκλογή μιας κατάλληλης μονάδας επεξεργασίας, αυτή μπορεί να είναι ενδεικτικά μια από τις παρακάτω:

- μια πλακέτα Raspberry-Pi βασισμένη σε μικροεπεξεργαστή αρχιτεκτονικής ARM,
- μια πλακέτα Arduino βασισμένη σε έναν μικροελεγκτή Atmel,
- ένα PLC βασισμένο στο βιομηχανικό ελεγκτή Simatic της Siemens,
- ή μια πλακέτα της Espressif βασισμένη στο μικροελεγκτή ESP-32.

Ανάλογα την εφαρμογή, λαμβάνουμε υπόψιν παράγοντες όπως είναι η κρισιμότητα του μηχανήματος που παρακολουθούμε, το διαθέσιμο κεφάλαιο (budget) για το έργο, η επαναληψιμότητα, ο χαρακτήρας της εφαρμογής (βιομηχανικό, εμπορικό, ερευνητικό έργο) και τα τεχνολογικά χαρακτηριστικά των διαθέσιμων λύσεων. Για τις ανάγκες τις παρούσας διπλωματικής εργασίας επιλέχθηκε ο μικροελεγκτής ESP-32 της Espressif και συγκεκριμένα η πλακέτα “ESP32-S3-DevKitC-1U-N8R8”

Η συλλογή των μετρήσεων από το πεδίο γίνεται σε όλες τις περιπτώσεις με περιφερειακά εισόδου διακριτών ή αναλογικών σημάτων. Οι μικροελεγκτές, οι βιομηχανικοί ελεγκτές ή ακόμα και οι μικροεπεξεργαστές έχουν την δυνατότητα να διαβάσουν αναλογικές τιμές με τη βοήθεια διαιρετών τάσης ή ψηφιακά σήματα μέσω τρανζίστορ. Παρόλα αυτά οι δυνατότητες των αναφερθέντων ηλεκτρονικών κυκλωμάτων είναι περιορισμένες και οι ενσωματωμένες είσοδοι και έξοδοι των ολοκληρωμένων κυκλωμάτων χρησιμοποιούνται για την λειτουργία των ιδίων συσκευών, τον έλεγχο και χειρισμό μέσω λυχνιών και μπουτών, την αποθήκευση σε στοιχεία μόνιμης (rom/ flash memory) ή προσωρινής μνήμης

(ram), την χρήση επιταχυντών (hardware accelerators) για κρυπτογράφηση και άλλες μαθηματικές πράξεις και αρκετές επιπλέον λειτουργίες. Είναι δε συχνό να παρατηρούμε ότι οι εκδόσεις των συστημάτων ελέγχου που έχουν ενσωματωμένες «κάρτες» εισόδου και εξόδου είναι οι οικονομικότερες εξ αυτών στην αγορά. Η επιλογή μιας αρχιτεκτονικής μονάδας ελέγχου η οποία περιλαμβάνει εξωτερικές-περιφερειακές μονάδες εισόδου και εξόδου σημάτων, προσφέρει ευελιξία και προσαρμοστικότητα στη σχεδίαση του έργου, ιδιαίτερα σε ερευνητικές εφαρμογές όπως η παρούσα διπλωματική εργασία. Στη παρούσα εργασία επιλέχθηκαν για τις αναλογικές εισόδους, μικροτσίπ της Adafruit για μετατροπή σήματος αναλογικού αισθητηρίου 4…20mA σε ψηφιακή πληροφορία και επιπλέον μικροτσίπ της ιδίας εταιρείας για μετατροπή σήματος θερμοκρασίας από αισθητήρια RTD PT1000. Η επικοινωνία των περιφερειακών υλικών με τον μικροελεγκτή ESP-32 γίνεται μέσω των πρωτοκόλλων I2C και SPI. Η είσοδος ψηφιακών σημάτων στο σύστημα συλλογής δεδομένων δύναται να πραγματοποιείται μέσω του ολοκληρωμένου κυκλώματος του μικροεπεξεργαστή με pull-down/pull-up αντιστάσεις, χωρίς τη χρήση περιφερειακών.

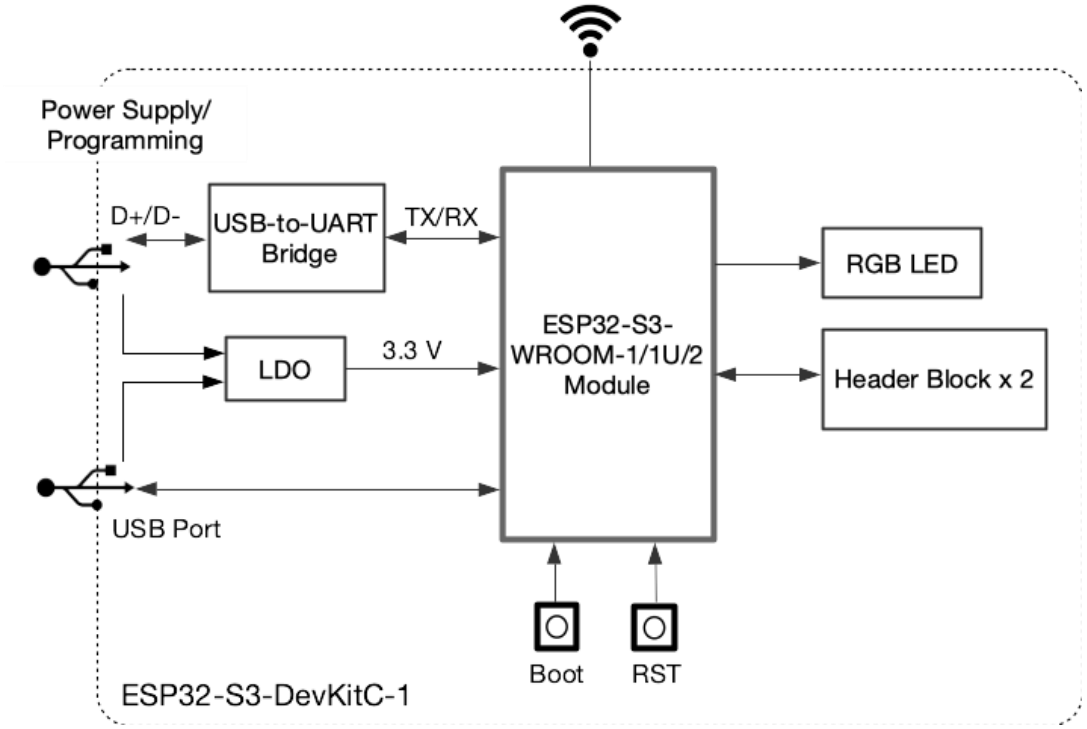
5.2.1 Μικροελεγκτής ESP-32

Ο μικροελεγκτής ESP-32 της Espressif είναι ιδιαίτερα επαναστατικός στο χώρο της του Διαδικτύου των Πραγμάτων. Οι ενσωματωμένες λειτουργικότητες Wi-Fi και Bluetooth σε συνδυασμό με την τρομερή επεξεργαστική ισχύ, το μέγεθος των μνημών ram και flash, την πληθώρα εισόδων και εξόδων, τη μικρή κατανάλωση ηλεκτρικής ισχύος και φυσικά το πολύ χαμηλό κόστος, καθιστούν τον ESP-32 ως «εργαλείο» ορόσημο για εμπορικές και ερευνητικές εφαρμογές του Διαδικτύου των Πραγμάτων. Οι μικροελεγκτές είναι πλέον η πιο οικονομική λύση για τη κατασκευή ενσωματωμένων (embedded) συστημάτων. Μικροελεγκτές μικρής κλίμακας αλλά γνωστοί στους αρχάριους προγραμματιστές, όπως ο ATMEL που χρησιμοποιεί η Arduino στα γνωστά εμπορικά της προϊόντα, παρουσιάζουν ιδιαίτερα απλή σχεδίαση με εύκολο προγραμματισμό από το περιβάλλον του Arduino IDE, ωστόσο οι δυνατότητες τους είναι περιορισμένες.

Η Espressif, μια εισηγμένη εταιρεία που ιδρύθηκε το 2008 στη Σανγκάη της Κίνας από τον “Teo Swee Ann”, έφερε την επανάσταση στους προγραμματιστές και στις εταιρείες πληροφορικής. Η έξυπνη κίνηση του να είναι τα προϊόντα της δυνατόν να προγραμματιστούν μέσου του περιβάλλοντος προγραμματισμού Arduino IDE, ήταν ένα τεράστιο πλεονέκτημα για την ανάπτυξη της εταιρείας. Κατ’ αυτό τον τρόπο όσοι είχανε εμπειρία γύρω από την Arduino και τον ATMEL, μπορούσαν με τις υπάρχοντες γνώσεις τους να εγκαταστήσουν εύκολα το κατάλληλο framework στους μικροελεγκτές της οικογένειας ESP και να αρχίσουν να απολαμβάνουν τις καινοτόμες δυνατότητες που τους δίνονταν, οι οποίες ξεπερνούσαν κατά πολύ τα μέχρι τότε οικονομικά προϊόντα της αγοράς.

Η Espressif προσφέρει στον κατάλογο της πολλές εκδόσεις του ESP32 και διανέμει την εκάστοτε έκδοση με τρείς διαφορετικούς τρόπους. Η οικονομικότερη λύση είναι να προμηθευτεί κανείς αυτό κάθε αυτού το ολοκληρωμένο κύκλωμα και στη συνέχεια να φροντίσει για τη κατασκευή πλακέτας, ο κωδικός ενός τέτοιου προϊόντος είναι ESP32-S3. Η επόμενη λύση είναι η προμήθεια μιας μικρής πλακέτα μεγέθους περίπου 2 επί 3 εκατοστά η οποία περιέχει το βασικό ολοκληρωμένο κύκλωμα και τις απαραίτητες διατάξεις όπως Flash memory, μεταλλική προστασία, κεραία Wi-Fi κα., ο αντίστοιχος κωδικός για αυτό το προϊόν μπορεί να είναι ESP32-S3-WROOM-1. Σε αντίθεση με την λύση του απλού ολοκληρωμένου που προτιμάτε από μεγάλες μαζικές παραγωγές, η ενδιάμεση αυτή λύση

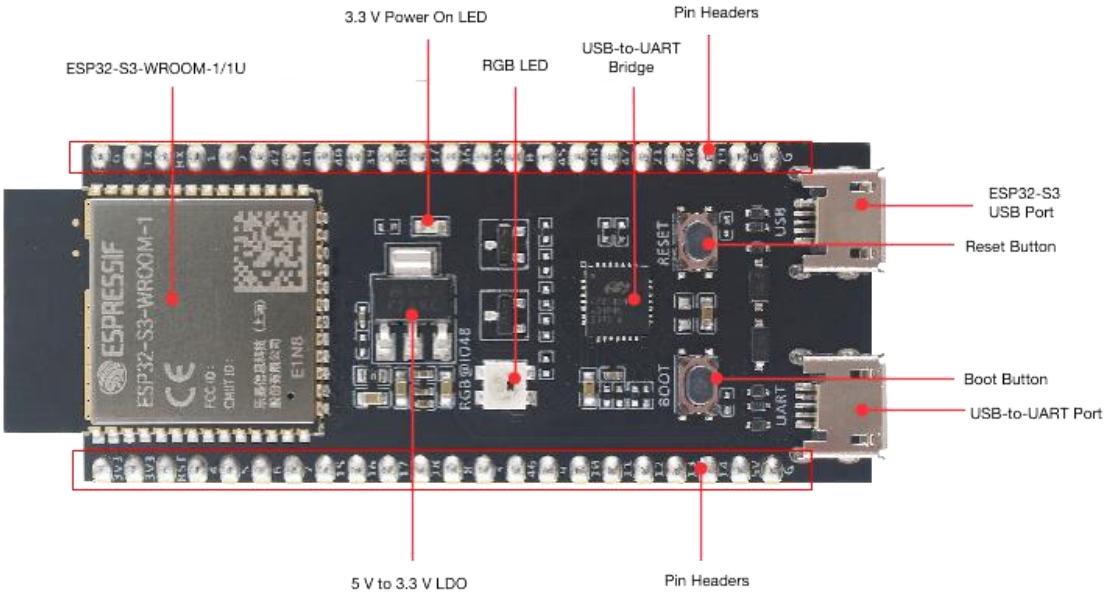
που συχνά αναφέρεται και ως ESP-WROOM, αποτελεί μια μέση κατεύθυνση για παραγωγές μεσαίου ή μικρού μεγέθους. Την πιο ολοκληρωμένη εκ των όλων λύση αποτελεί η πλακέτα ανάπτυξης ESP32-S3-DevKitC-1, η οποία με τέσσερις φορές μεγαλύτερο κόστος (περίπου 15 $) προσφέρει στον προγραμματιστή όλες εκείνες τις ηλεκτρονικές διατάξεις που χρειάζεται για το έργο του. Το ESP DevKit το οποίο απεικονίζεται διαγραμματικά και αναλύεται στην Εικόνα 18, περιλαμβάνει σταθεροποιητή τάσης, μπουτών, διεπαφή USB και λυχνίες.



Εικόνα 18: Διάγραμμα Αρχιτεκτονικής μικροελεγκτή ESP32-S3-DevKitC-1 [46]

Δεδομένων των παραπάνω, το μοντέλο μικροελεγκτή που εκλέχθηκε για τη διπλωματική εργασία είναι το ESP32-S3-DevKitC-1-N8R8.

Μία πιο καλύτερη αντίληψη του DevKit μπορεί να δοθεί και από την παρακάτω Εικόνα 19.

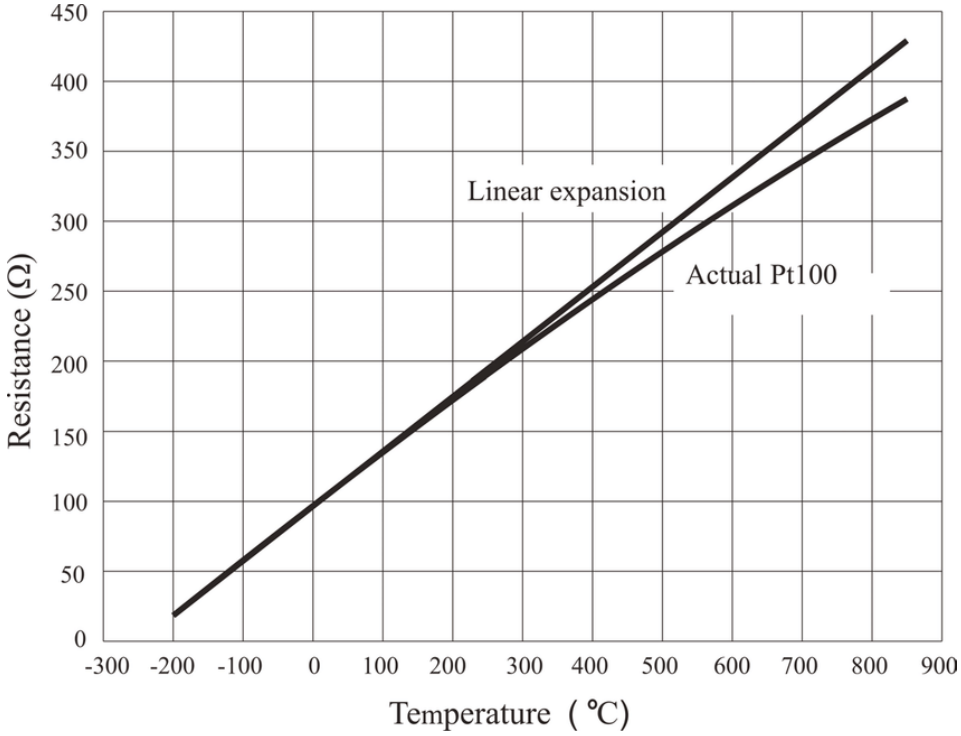


Εικόνα 19: Άνω Όψη του ESP32-S3-DevKitC-1 [46]

5.2.2 Αισθητήρια και κάρτες εισόδου αναλογικών σημάτων

Τα αισθητήρια είναι ο τρόπος επικοινωνίας του φυσικού περιβάλλοντος με τις μονάδες συλλογής δεδομένων. Η εκλογή των κατάλληλων τύπων αισθητηρίων προκύπτει μέσω της αποσαφήνισης των φυσικών μεγεθών που θέλουμε να μετρήσουμε. Πέρα από το ότι υπάρχουν ενεργητικά και παθητικά αισθητήρια, υπάρχουν και πολλές υποκατηγορίες αυτών ανάλογα το μέγεθος μέτρησης και το μέσο.

Αν θέλουμε να μετρήσουμε θερμοκρασία, υπάρχουν τα παθητικά αισθητήρια όπως τα Resistance Temperature Detector (PT100, PT500, PT1000, LG-Ni1000 της Siemens κ.α.) και τα Thermocouples, τα οποία είναι αντιστάσεις που μεταβάλουν την τιμή τους, άρα και την τάση που τα διαπερνάει ανάλογα την θερμοκρασιακή μεταβολή τους σχεδόν με απόλυτη γραμμικότητα, όπως φαίνεται παρακάτω στην Εικόνα 20.



Εικόνα 20: Γραμμικότητα αισθητηρίων RTD [47]

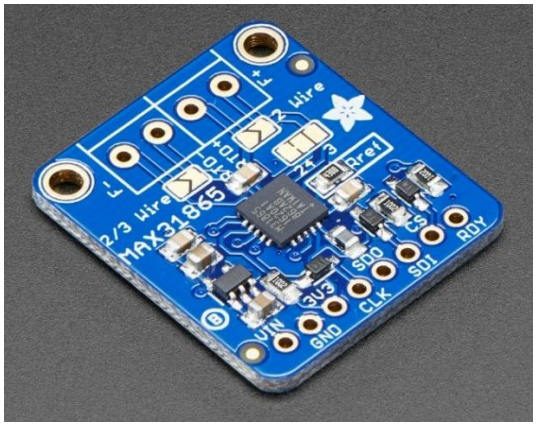
Ένας άλλος τύπος είναι τα ενεργητικά αισθητήρια, αν και αυτά χρησιμοποιούν μια διάταξη ημιαγωγών, είναι εξοπλισμένα με τη δική τους ηλεκτρονική διάταξη (μερικούς πυκνωτές και αντιστάσεις διαιρέτες τάσης και κάποια ολοκληρωμένα κυκλώματα) ώστε να υπολογίζουν την τιμή που μετράνε και να την μεταδίδουν στη κεντρική μονάδα μέσω ενός πρωτοκόλλου επικοινωνίας (πχ Modbus, I2C, SPI, Profinet, CanBus) σε ψηφιακή μορφή με bits. Οι κύριες διαφορές ανάμεσα στα ενεργητικά και παθητικά αισθητήρια είναι ότι τα ενεργητικά χρειάζονται τροφοδοσία ρεύματος και η έξοδος τους είναι ψηφιακή. Ανάλογα την εφαρμογή επιλέγουμε πάντα το πιο κατάλληλη κατηγορία και το αισθητηρίων. Στη παρούσα εργασία επιλέχθηκαν για την μέτρηση της θερμοκρασίας του υγρού, αισθητήρια RTD PT1000 τα οποία έχουν ιδιαίτερη χρήση στο βιομηχανικό αυτοματισμό και μέσω αυτών μπορούμε να μετρήσουμε θερμοκρασία ενός ρευστού, αφού αυτά έρχονται και μέσα σε σωληνίσκο (probe) όπως φαίνεται και παρακάτω στην Εικόνα 21.

Εικόνα 21: Αισθητήρια Μέτρησης Θερμοκρασίας RTD

Αν θέλουμε να μετρήσουμε πίεση ενός ρευστού σε ένα υπολογιστικό σύστημα, υπάρχουν μόνο ενεργητικά αισθητήρια. Μολονότι και αυτά χρησιμοποιούν ημιαγωγούς για την αίσθηση του φυσικού φαινομένου που ονομάζεται πίεση, η χρήση κατάλληλων στοιχείων και οι διενέργεια μαθηματικών πράξεων είναι απαραίτητη για τη «μετάφραση» των δεδομένων ενός πιεζοηλεκτρικού αισθητηρίου σε πίεση μονάδων Pascal και στη συνέχεια σε ένα εύρος bar, ανάλογα το εύρος μέτρησης. Τα αισθητήρια αυτά τα οποία χρησιμοποιούνται σε βιομηχανικούς αυτοματισμούς, λαμβάνουν μια τροφοδοσία λειτουργίας (συνήθως 12-30V DC) και επιστρέφουν έξοδο αναλογικού σήματος σε ρεύμα (4 … 20 mA) ή τάση (0-10V). Πλέον όμως μια άλλη διάταξη, μια περιφερειακή κάρτα εισόδων και εξόδων είναι απαραίτητη για την αναγνώριση του φυσικού μεγέθους σε ψηφιακή μορφή (bits) από την μονάδα ελέγχου. Παραδείγματα αισθητηρίων πίεσης με έξοδο ρεύματος φαίνονται παρακάτω στην Εικόνα 22.

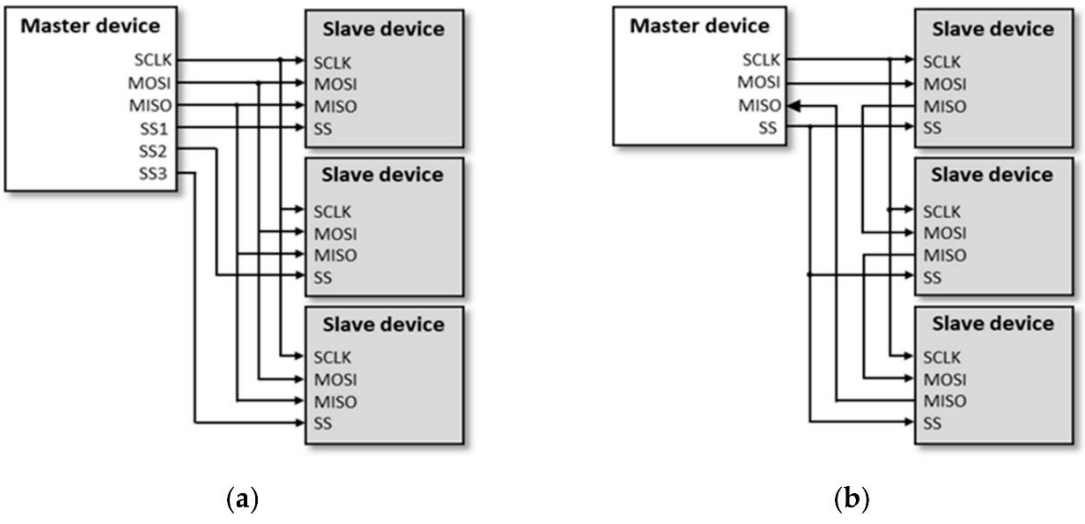
Εικόνα 22: Αισθητήρια Μέτρησης Πίεσης [48]

Η Αρχιτεκτονική του κάθε συστήματος είναι μοναδική, αν και στη παρούσα διπλωματική εργασία χρησιμοποιούμε αισθητήρια παθητικά αισθητήρια PT1000 και ενεργητικά αισθητήρια πίεσης, οφείλουμε να ενσωματώσουμε τις κατάλληλες περιφερειακές εισόδους στο μικροελεγκτή μας που θα μας επιτρέψουν να μεταφέρουμε σε αυτόν τα αναλογικά σήματα. Η ανάγνωση των τιμών των αισθητηρίων θερμοκρασίας θα γίνει με μια περιφερειακή πλακέτα της Adafruit με το ολοκληρωμένο κύκλωμα MAX31865 το οποίο απεικονίζεται παρακάτω στην Εικόνα 23.



Εικόνα 23: Ηλεκτρονική διάταξη Adafruit με μικροτσίπ MAX31865. [49]

Η συγκεκριμένη συσκευή λαμβάνει πληροφορίες από τον μικροελεγκτή μας για τον τύπο του αισθητηρίου RTD που είναι συνδεδεμένο και στέλνει ηλεκτρική τάση για να μετρήσει τη διαφορά δυναμικού στα άκρα της αντίστασης με τη βοήθεια διαιρέτη τάσης. Η πληροφορία αυτή μεταφέρεται με το πρωτόκολλο επικοινωνίας Serial Parallel Interface στον ESP-32, ενώ αυτός είναι που ρωτάει την εκάστοτε περιφερειακή συσκευή SPI για τις τιμές που έχει μετρήσει. Παρακάτω αναλύεται στην Εικόνα 24 η μοντελοποίηση του πρωτοκόλλου SPI.



Εικόνα 24: Μοντελοποίηση SPI Master-Slave [50]
Διάγραμμα διαμόρφωσης πολλαπλών slave συσκευών στο SPI: (α) αυτόνομη διαμόρφωση slave συσκευής SPI και (β) διαμόρφωση δαιδαλώδους αλυσίδας slave συσκευών SPI.

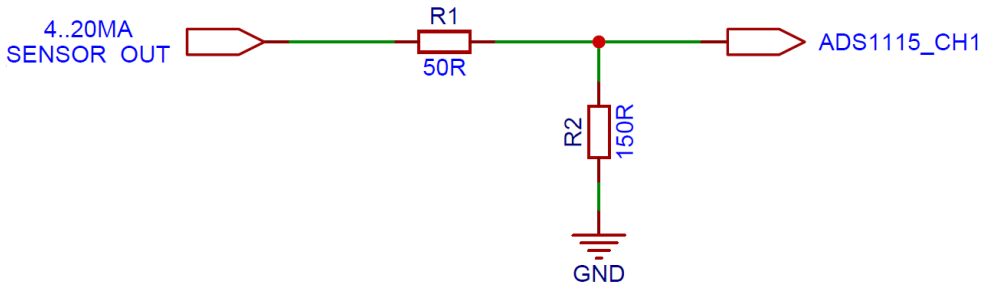
Αν και ο μικροελεγκτής ESP-32 μπορεί να διαβάσει αναλογικά σήματα στους ηλεκτρικούς συνδέσμους γενικής χρήσης εισόδων και εξόδων (GPIO Pins), έχει περιορισμό σε εύρος 0 έως 1V, ενώ δεν προβλέπεται η μέτρηση σημάτων έντασης ρεύματος (4..20mA). Υστέρα λοιπόν από μελέτη των διαθέσιμων λύσεων επιλέχθηκε η χρήση κυκλώματος διαιρετή

τάσης και της πλακέτας της Adafruit με το ολοκληρωμένο κύκλωμα ADS1115, όπως φαίνεται παρακάτω στην ηλεκτρονική διάταξη στην Εικόνα 25. Με σταθερή τάση 24V τροφοδοτούμε ένα αισθητήριο πίεσης με έξοδο ρεύματος και με δύο αντιστάσεις 50 και 150Ωμ διαιρούμε την τάση εξόδου στα 0,6 έως 3,0 Volt όπως φαίνεται βάση των εξισώσεων που ακολουθούν.

$$Vout = \mathrm{I} * \mathrm{R2} = \frac{R2}{\mathrm{R1} + \mathrm{R2}} * Vin$$

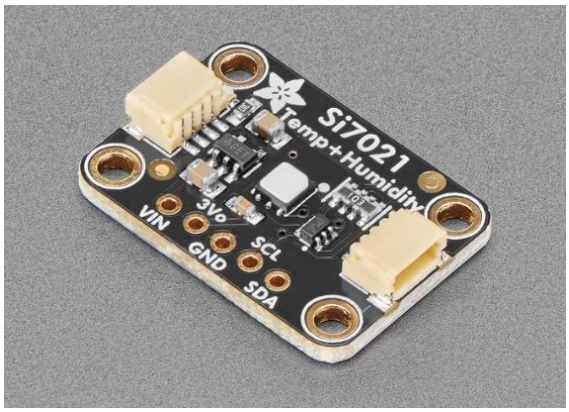
$$\overset{I=0,004}{\Longleftrightarrow} Vout = 0{,}004\mathrm{A} * 150\Omega = \frac{150\Omega}{50\Omega + 150\Omega} * 0{,}8V = 0{,}6V$$

$$\overset{I=0,020}{\Longleftrightarrow} Vout = 0{,}020\mathrm{A} * 150\Omega = \frac{150\Omega}{50\Omega + 150\Omega} * 4V = 3V$$



Εικόνα 25: Διαιρέτης τάσης για μέτρηση πίεσης με αισθητήρια 4..20 mA

Στην ίδια κατεύθυνση των περιφερειακών συσκευών με I2C και SPI, υπάρχουν και αυτόνομες διατάξεις με ενσωματωμένους αισθητήρες θερμοκρασίας, υγρασίας, βαρομετρικής πίεσης, ποιότητας αέρα ή ακόμα και αισθητήρες έντασης φωτός και πολλά άλλα. Τέτοιες συσκευές ονομάζονται απλώς αισθητήρες και το χαρακτηριστικό τους είναι ότι μπορούν να μετρήσουν κάτι μόνο σε περιβάλλον ατμόσφαιρας. Δεδομένου ότι οι ημιαγωγοί που αισθάνονται το κάθε μέγεθος είναι τοποθετημένοι πάνω στις πλακέτες, είναι αδύνατον να μετρήσουμε για παράδειγμα την θερμοκρασία ενός υγρού με αυτή τη διάταξη. Ενώ επίσης μπορούμε να μετρήσουμε τη πίεση της ατμόσφαιρας, δεν μπορούμε να μετρήσουμε την πίεση ενός αερίου σε κενό ή ένα πεπιεσμένο αέριο διότι τα ίδια τα ηλεκτρονικά της πλακέτας δεν αντέχουν σε μη ατμοσφαιρικές συνθήκες. Ωστόσο για περιπτώσεις κλιματισμού κτηρίων που θέλουμε να μετρήσουμε θερμοκρασία και υγρασία χώρου σε ανθρωπινά ανεκτές συνθήκες η επιλογή των αναφερθέντων αισθητηρίων είναι μια πολύ εύκολη και οικονομική λύση. Ο παρακάτω αισθητήρας στην Εικόνα 26 είναι μια εξαιρετική λύση για την μέτρηση των δυο αυτών μεγεθών στην πειραματική διάταξη.



Εικόνα 26: I2C Αισθητήρας “Si7021” για μέτρηση θερμοκρασίας και υγρασίας[51]

5.2.2.1 Περιβάλλον προγραμματισμού

Η εκάστοτε μονάδα ελέγχου και συλλογής δεδομένων έχει το δικό της τρόπο προγραμματισμού. Σήμερα ο προγραμματισμός των μικροελεγκτών και συγκεκριμένα του ESP-32 γίνεται μέσω υπολογιστή με τη χρήση ειδικού λογισμικού. Ένα από τα μεγαλύτερα πλεονεκτήματα της Espressif ήταν ότι έδωσε πολύ νωρίς τη δυνατότητα προγραμματισμού των προϊόντων της, με τη γλώσσα C++ μέσω της πλατφόρμας Arduino IDE. Οι προγραμματιστές που είναι εξοικειωμένοι στη πλατφόρμα δεν βρίσκουν δυσκολία στο προγραμματισμό του νέου αυτού μικροελεγκτή. Η Espressif μοιράζει ένα πακέτο αρχείων τα οποία φορτώνει ο προγραμματιστής στο Arduino IDE και το περιβάλλον αυτό μπορεί να μεταγλωττίσει τον κώδικα από C++ σε κατάλληλη γλώσσα μηχανής την οποία μπορεί στη συνέχεια να εκτελέσει ο ESP και να αποθηκεύσει τον αλγόριθμο στη μνήμη flash του. Η συγκεκριμένη ιδιότητα είναι πολύ χρήσιμη, ωστόσο στερείται μερικές φορές σε αξιοπιστία και σταθερότητα της λειτουργίας του προγράμματος, ενώ δεν είναι διαθέσιμες όλες οι δυνατότητες που προσφέρει η εταιρεία για τον μικροελεγκτή. Ο προτεινόμενος τρόπος προγραμματισμού σύμφωνα με τον κατασκευαστή είναι η χρήση του Development Framework ESP-IDF. Ο προγραμματιστής εγκαθιστά στον υπολογιστή του το συγκεκριμένο πακέτο και μέσω ενός περιβάλλοντος συγγραφής κώδικα σαν το Eclipse ή το Visual Studio Code της Microsoft, φτιάχνει προγράμματα στις γλώσσες C ή C++ τα οποία αντίστοιχα μεταγλωττίζονται σε γλώσσα μηχανής μέσω του ESP-IDF και το Visual Studio αναλαμβάνει την αποθήκευση τους στο σύστημα αρχείων του μικροελεγκτή. Αυτή η μέθοδος προγραμματισμού έχει αναρίθμητα πλεονεκτήματα. Οι εταιρείες λογισμικού που φτιάχνουν ολοκληρωμένες εμπορικές εφαρμογές προτιμάνε αυτόν τον επίσημο τρόπο μέσω του ESP-IDF καθώς έχουν όλες τις δυνατότητες για τον χειρισμό της συσκευής, μπορούν να απευθυνθούν σε μια μεγάλη και ώριμη κοινότητα έμπειρων και σοβαρών προγραμματιστών και επιπλέον έχουν στη διάθεση τους την τεχνική υποστήριξη της ιδίας της εταιρείας. Για τους σκοπούς της παρούσας εργασίας επιλέχθηκε ο προγραμματισμός μέσω της επίσημης πρότασης της εταιρείας με σκοπό να υπάρχει η πρόσβαση σε μια πιο καταρτισμένη κοινότητα και η παραγόμενη εφαρμογή να είναι πιο ώριμη για περεταίρω εξέλιξη. Ο κώδικας σε γλώσσα C++ του μικροελεγκτή βρίσκεται στο Παράρτημα Α’ – Κώδικας μικροελεγκτή

5.2.3 Δημιουργία πακέτου δεδομένων και αποστολή στο διαδίκτυο

Έπειτα από την συλλογή των μετρήσεων από τους αισθητήρες σε τοπικό επίπεδο, ο μικροελεγκτής θα πρέπει να ομαδοποιήσει τα δεδομένα να ελέγξει την εγκυρότητα τους και να δημιουργήσει μια ακολουθία πληροφορίας σε δυαδική μορφή. Η επικοινωνία των τερματικών συσκευών, δηλαδή των επιμέρους μικροελεγκτών που παρακολουθούν έκαστοι μια μονάδα ψύξης θέρμανσης και του κεντρικού υπολογιστή στο νέφος, γίνεται κάτω από αυστηρούς κανόνες οι οποίοι ορίζουν την ανταλλαγή συγκεκριμένης πληροφορίας για το κάθε σύστημα. Η πληροφορία στον μικροελεγκτή μορφοποιείται κάτω από μια νέα δομή δεδομένων με συγκεκριμένο αναγνωριστικό. Με αυτό τον τρόπο στην άλλη πλευρά του δικτύου, ο εξυπηρετητής μπορεί να αναγνωρίσει ανά σύστημα ψύξης/θέρμανσης τα εισερχόμενα δεδομένα και να τα χειριστεί ανάλογα.

5.3 Μονάδα συλλογής και ανάλυσης δεδομένων σε υπολογιστικό νέφος

Ως πλατφόρμα επεξεργασίας και ανάλυσης δεδομένων ονομάζεται η υποδομή εκείνη στην οποία καταλήγουν τα δεδομένα από τους μικροελεγκτές – τερματικά, τα διαχωρίζει και στη συνέχεια τα επεξεργάζεται και τα αναλύει κατάλληλα. Ο σωστός διαχωρισμός είναι ζωτικής σημασίας για κάθε σύστημα ψύξης θέρμανσης που παρακολουθείται. Είναι λοιπόν απαραίτητη μια σωστή διαχείριση των δεδομένων στην πλευρά του εξυπηρετητή σε ένα μοντέλο client-server.

Αναλύοντας τη συνθήκη οπού ακολουθίες δεδομένων εισέρχονται στον υπολογιστικό νέφος, καταλήγουμε εύκολα στο συμπέρασμα ότι πρέπει να ενσωματώσουμε την εκτέλεση ενός ή μιας ομάδα αλγορίθμων οι οποίοι θα αναλάβουν:

- την λήψη των δεδομένων,
- την αποθήκευση τους,
- την εκπαίδευση των αλγορίθμων της τεχνητής νοημοσύνης
- τον έλεγχο των νέων δεδομένων και την ανάλυση τους με τα μοντέλα της τεχνητής νοημοσύνης για την παραγωγή μηνυμάτων ενημερώσεις

Ένας τρόπος για να εκτελεστούν όλες οι παραπάνω εργασίες είναι με τη συγγραφή προγραμμάτων για παράδειγμα στη γλώσσα python. Ένας άλλος τρόπος και μάλιστα ιδιαίτερα δημοφιλής στην κατασκευή πρωτότυπων και ερευνητικών εφαρμογών είναι η εκτέλεση σεναρίων ροών δεδομένων με το Node-Red. Το NodeRed το οποίο έχει αναλυθεί στις προηγούμενες παραγράφους, επιλέχθηκε για στην παρούσα διπλωματική εργασία επειδή μας δίνει εύκολα και αξιόπιστα την δυνατότητα να κάνουμε λήψη των δεδομένων μέσω των πρωτοκόλλων επικοινωνίας και στη συνέχεια να τα αποθηκεύσουμε σε μία βάση δεδομένων λαμβάνοντας υπόψιν μας από ποιο σύστημα ψύξης/θέρμανσης έχουν προέλθει. Ωστόσο, ο τρόπος αυτός δεν ενδείκνυται για την εκτέλεση πιο πολύπλοκών διεργασιών όπως η εκπαίδευση αλγορίθμων τεχνητής νοημοσύνης. Για τον σκοπό αυτό επιλέχθηκε να χρησιμοποιήσουμε διαδικαστικό προγραμματισμό στη γλώσσα Python, η οποία έχει ιδιαίτερη απήχηση στους νέους προγραμματιστές με μεγάλη υποστήριξη σε εφαρμογές τεχνητής νοημοσύνης μέσω πληθώρας διαθέσιμων βιβλιοθηκών αλλά και εγχειριδίων στο διαδίκτυο.

Η παραμετροποίηση του NodeRed της παρούσας διπλωματικής εργασίας παρουσιάζεται στο Παράρτημα Β’ – Υποδομή Node Red

5.3.1 Λήψη δεδομένων με το πρωτόκολλο MQTT

Για τις ανάγκες της λήψης των δεδομένων από τους μικροελεγκτές πρέπει οι εκάστοτε δυο πλευρές του δικτύου να επικοινωνούν σε επίπεδο εφαρμογής με το ίδιο πρωτόκολλο. Ένα από τα πιο γνωστά πρωτόκολλα για εφαρμογές του διαδικτύου των πραγμάτων κάτω από την αρχιτεκτονική του μοντέλου πελάτη-εξυπηρετητή, είναι το MQTT, “Message Queuing Telemetry Transport”. Το συγκεκριμένο πρωτόκολλο επιλέχθηκε χάριν της ιδιότητας του για δημοσίευση και εγγραφή πληροφορίας σε κανάλια δεδομένων, κάτι που δίνει γενικότερα πολλά πλεονεκτήματα στις εφαρμογές του διαδικτύου, όπως και στην παρούσα εργασία. Εγκαθιστώντας λοιπόν ένα πρόγραμμα εξυπηρετητή MQTT στο υπολογιστικό νέφος, είναι πλέον δυνατό να λάβουμε δεδομένα από τους μικροελεγκτές. Εφόσον η επικοινωνία των δύο πλευρών γίνεται αυστηρά κάτω από το MQTT, μπορούμε να εγκαταστήσουμε οποιοδήποτε πρόγραμμα εξυπηρετητή MQTT για την πραγματοποίηση της επικοινωνίας.

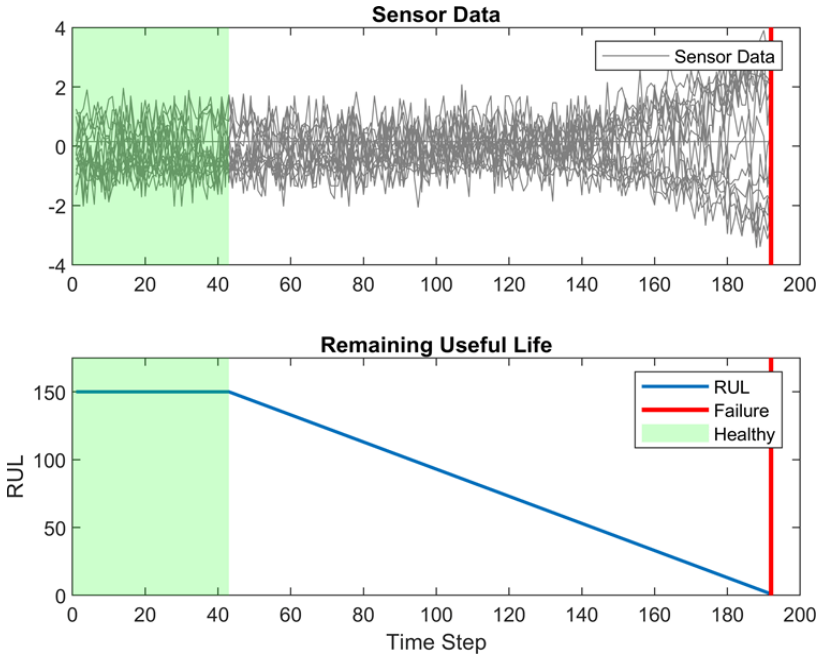
Για τους σκοπούς της εργασίας επιλέχθηκε ο Mosquito Server, ο οποίος είναι ιδιαίτερα γνωστός και απλός και η υλοποίηση του ήταν ιδιαίτερα εύκολη.

5.3.2 Αποθήκευση δεδομένων σε μη σχεσιακή βάση δεδομένων

Τα ληφθέντα πλέον δεδομένα της εφαρμογής, έχουν αποθηκευτεί στη προσωρινή μνήμη και πρέπει να αποθηκευτούν στον σκληρό δίσκο του κεντρικού υπολογιστή για περεταίρω διαχείριση και ανάλυση. Η εκλογή της κατάλληλης βάσης δεδομένων γίνεται σε κάθε περίπτωση λαμβάνοντας υπόψη τους κανόνες που χαρακτηρίζουν τα δεδομένα της εφαρμογής. Στις εφαρμογές παρακολούθησής συστημάτων του διαδικτύου των πραγμάτων αυτό που χαρακτηρίζει τα δεδομένα είναι η επαναληψιμότητα ανά κάποια χρονική στιγμή και η μη συσχέτιση των μετρήσεων ανά σύστημα. Εφόσον τα βασικά χαρακτηριστικά είναι ο χρόνος και η μη κανονικότητα των δεδομένων, δηλαδή ή μη συσχέτισή μεταξύ τους, προκύπτει ότι η κατάλληλη υποδομή είναι μια Μη Σχεσιακή Βάση Δεδομένων Χρονοσειρών (NoSQL Time Series Database). Παράδειγμα τέτοιου μοντέλου είναι η InfluxDB, ένα προϊόν της εταιρείας Influx Data με ιδιαίτερη θέση στον πίνακα κατάταξης των πιο δημοφιλών βάσεων δεδομένων. Η συγκεκριμένη λύση επιλέχθηκε ως η πιο δημοφιλής και πλέον κατάλληλη για εφαρμογές του διαδικτύου των πραγμάτων με βάση την βιβλιογραφία που αναλύθηκε σε προηγούμενα κεφάλαια.

5.3.3 Ανάλυση δεδομένων με Τεχνητή Νοημοσύνη

Η ενσωμάτωση της τεχνητής νοημοσύνης στη παρούσα διπλωματική εργασία έχει στόχο την βελτίωση της εναπομείναντος ωφέλιμης ζωής του μηχανολογικού εξοπλισμού, την αποτροπή ανεπιθύμητων βλαβών και την ενίσχυση της ικανότητας των μηχανικών να αντιμετωπίζουν έγκαιρα τις βλάβες που προκύπτουν. Τα ακόλουθα δύο γραφήματα στην Εικόνα 27, παρέχουν από κοινού μια ολοκληρωμένη εικόνα του τρόπου με τον οποίο η συμπεριφορά των αισθητήρων συσχετίζεται με την υγεία και τη διάρκεια ζωής μιας συσκευής ή ενός συστήματος. Η πράσινη περιοχή σηματοδοτεί μια περίοδο καλής λειτουργίας και υψηλής υπολειπόμενης ωφέλιμης ζωής, η οποία μειώνεται με την πάροδο του χρόνου καθώς αυξάνονται οι ανωμαλίες στον αισθητήρα, με αποκορύφωμα ένα συμβάν αστοχίας όπου η υπολειπόμενη ωφέλιμη ζωή φτάνει στο μηδέν. Αυτός ο τύπος ανάλυσης είναι ζωτικής σημασίας για την προγνωστική συντήρηση, επιτρέποντας παρεμβάσεις πριν από την εμφάνιση κρίσιμων βλαβών.



Εικόνα 27: Ανάλυση Κατάστασης Συστήματος μέσω δεδομένων από αισθητήρες. [52]

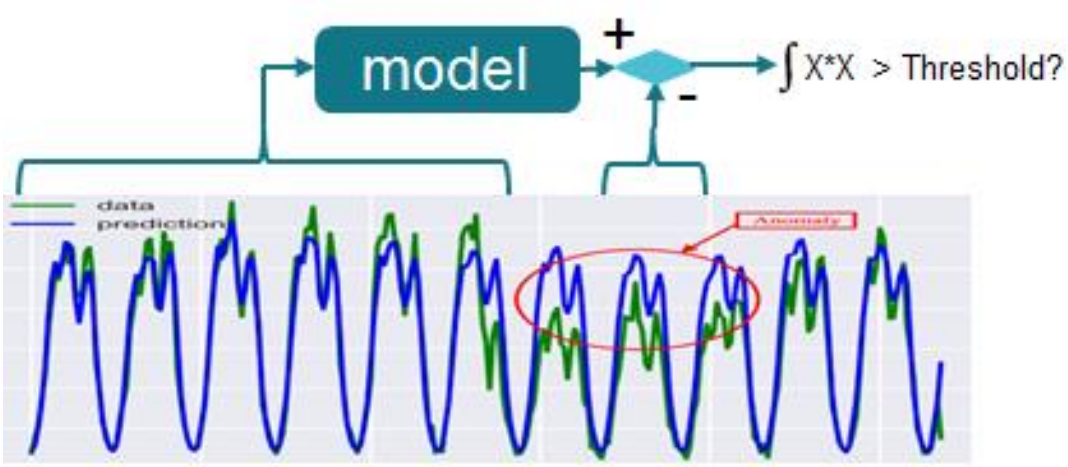
Όλα τα συστήματα παρακολούθησης δημιουργούν έναν τεράστιο όγκο δεδομένων στη μονάδα του χρόνου τα οποία μπορούν να δώσουν αναρίθμητα συμπεράσματα για την λειτουργία των συστημάτων αυτών. Τα δεδομένα αυτά που προκύπτουν είναι διακεκριμένα με βάση το χρόνο, έχουν όλα μια χρονοσφραγίδα “timestamp” και εκφράζουν τη μεταβολή σημάτων χρονοσειρών μέσα από την επιρροή συγκεκριμένων ποσοτικών και ποιοτικών μεταβλητών. Η αξία της ενσωμάτωσης της τεχνικής νοημοσύνης, έναντι της υλοποίησης απλών ελέγχων σε συγκεκριμένες τιμές αισθητηρίων είναι ότι δεν υπάρχει η ανάγκη εκτενούς προγραμματισμού για κάθε πιθανό σενάριο που προκαλεί κατάσταση σφάλματος. Αυτός είναι δε και ο ορισμός της τεχνικής νοημοσύνης τον οποίο έδωσε ο κος “Arthur Samuel” το 1959.

Η τεράστια ικανότητα των υπολογιστών να αναπτύξουν την δικιά τους νοημοσύνη και να μπορούν στη συνέχεια να ξεχωρίσουν τι θα πει σφάλμα και τι όχι, χωρίς να απαιτούν τον λεπτομερή τους προγραμματισμό κάνει αυτή τη τεχνολογία εξαιρετικά ανταγωνιστική στο περιβάλλον της πληροφορικής. Έρχεται μάλιστα να αντιμετωπίσει και το πρόβλημα όπου έμπειροι μηχανολόγοι, και ηλεκτρολόγοι μηχανικοί θα χρειαζόταν να κάνουν εξατομικευμένη έρευνα πάνω στη συμπεριφορά των συστημάτων, ώστε να βγάλουν συμπεράσματα και στη συνέχεια να με γνώσεις προγραμματισμού και αυτοματισμού να υλοποιήσουν χρήσιμους ελέγχους. Δεδομένου ότι δεν υπάρχουν τόσο πολλοί έμπειροι μηχανικοί στην αγορά και οι όλοι είναι επιφορτισμένοι με τον έλεγχο, την συντήρηση και την εγκατάσταση είναι αδιαμφισβήτητα ωφέλιμο να υπάρχει μια πλατφόρμα παρακολούθησης συστημάτων μηχανολογικού εξοπλισμού που θα μπορούν να δίνουν χρήσιμα συμπεράσματα στους λίγους διαθέσιμους μηχανικούς.

Με την υιοθέτηση της τεχνητής νοημοσύνης μπορεί να επιτευχθεί και η επέκταση της ζωής του εκάστοτε εξοπλισμού, κάτι που μπορεί να δώσει ιδιαίτερα οφέλη στη ταχεία ανάγκη μείωσης των αποβλήτων που παράγει η ανθρωπότητά και στη μεγάλη καταναλωτική ανάγκη που υπάρχει για παραγωγή νέων μονάδων. Μια τέτοια προσέγγιση φέρνει τις χώρες ολόκληρου του κόσμου πιο κοντά στη μείωση του αποτυπώματος άνθρακα με τη μείωση

των εκπομπών αερίων του θερμοκηπίου και τη μείωση των εξορύξεων, εφαρμόζοντας μια πιο πράσινη πολιτική.

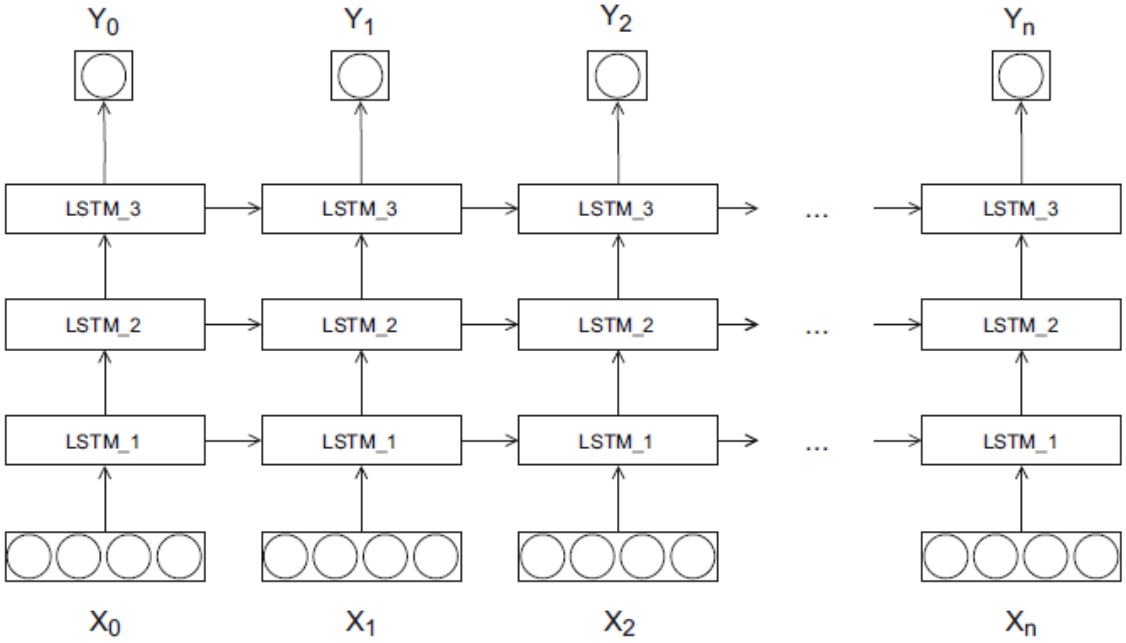
Η προσέγγιση της τεχνητής νοημοσύνης στη λειτουργία του μηχανολογικού εξοπλισμού βασίζεται στο πρότυπο της προγνωστικής συντήρησης (predictive maintenance). Για τον σκοπό αυτό είναι απαραίτητη η σχεδίαση ενός μοντέλου που βασίζεται σε έναν αλγόριθμο τεχνητής νοημοσύνης. Το μοντέλο αυτό έχει δική του, κατάλληλη αρχιτεκτονική που του επιτρέπει να εκπαιδευτεί μέσα από τις μετρήσεις ώστε να αναγνωρίσει τη συμπεριφορά του συστήματος ψύξης/ θέρμανσης. Όπως φαίνεται και στην Εικόνα 28, η εκπαίδευση του μοντέλου λαμβάνει χώρα με δεδομένα υγιούς συμπεριφοράς, ο αλγόριθμος μαθαίνει την αντίδραση του εκάστοτε συστήματος απέναντι στις πολυποίκιλες αλλαγές των δεδομένων αυτών και μπορεί να προβλέψει την συμπεριφορά στο μέλλον. Η προγνωστικής συντήρηση προκύπτει ως αποτέλεσμα της σύγκριση της πρόβλεψης έναντι των πραγματικών δεδομένων, ο έλεγχος του σφάλματος και η σύγκριση έναντι ενός ορίου είναι αυτό που ανιχνεύει τις ανωμαλίες στη λειτουργία και αναγνωρίζει βλάβες πριν καν αυτές προκύψουν.



Εικόνα 28: Ανίχνευση ανωμαλιών στις μετρήσεις δεδομένων [53]

5.3.3.1 Υλοποίηση προγνωστικής συντήρησης μέσω δικτύων μακράς βραχυπρόθεσμης μνήμης

Η Εικόνα 29 απεικονίζει ένα στοιβαγμένο δίκτυο μακράς βραχυπρόθεσμης μνήμης (LSTM), μια ισχυρή αρχιτεκτονική για την πρόβλεψη σφαλμάτων από ακολουθίες δεδομένων. Αυτή η αρχιτεκτονική είναι ιδιαίτερα κατάλληλη για διαδικασίες προγνωστικής συντήρησης σε συστήματα ψύξης θέρμανσης λόγω της ικανότητάς της να συλλαμβάνει μακροχρόνιες εξαρτήσεις σε δεδομένα χρονοσειρών.



Εικόνα 29: Κοινή αρχιτεκτονική βαθιάς μάθησης για προγνωστική συντήρηση με βάση επίπεδα LSTM [54]

Τα συστήματα ψύξης/ θέρμανσης λειτουργούν σύμφωνα με πολύπλοκες δυναμικές που επηρεάζονται από πολλαπλούς παράγοντες, όπως οι περιβαλλοντικές συνθήκες, οι λειτουργικές ρυθμίσεις και η μηχανική φθορά. Με την πάροδο του χρόνου, οι παράμετροι αυτοί συμβάλλουν στη δημιουργία μοτίβων στα δεδομένα που μπορεί να υποδεικνύουν επικείμενες βλάβες ή ανάγκες συντήρησης. Οι ακόλουθοι λόγοι φανερώνουν γιατί τα δίκτυα LSTM είναι καίριας σημασίας για αυτή την προγνωστική συντήρηση συστημάτων ψύξης θέρμανσης ή και άλλων βιομηχανικών μονάδων:

- Χρονικός χειρισμός δεδομένων: Τα συστήματα κλιματισμού δημιουργούν συνεχή δεδομένα με την πάροδο του χρόνου. Τα δίκτυα LSTM είναι ειδικά σχεδιασμένα για το χειρισμό τέτοιων διαδοχικών δεδομένων, όπου η σειρά των σημείων δεδομένων είναι κρίσιμη. Το κύτταρο μνήμης στα δίκτυα LSTM διατηρεί πληροφορίες για μεγάλα χρονικά διαστήματα, γεγονός ζωτικής σημασίας για την αναγνώριση μοτίβων που φανερώνουν σε πιθανές βλάβες του συστήματος.
- Πολυπλοκότητα επιπέδων: Όπως απεικονίζεται στο διάγραμμα, τα στοιβαγμένα επίπεδα των LSTM επιτρέπουν στο μοντέλο να μαθαίνει διαφορετικές πληροφορίες από κάθε αλληλουχία δεδομένων. Το πρώτο στρώμα μπορεί να καταγράφει βασικά χαρακτηριστικά, όπως οι διακυμάνσεις της θερμοκρασίας, ενώ βαθύτερα στρώματα μπορούν να ερμηνεύσουν πιο σύνθετα μοτίβα, όπως η συσχέτιση μεταξύ διαφορετικών λειτουργικών σημάτων.
- Δυνατότητες πρόβλεψης: Για την προγνωστική συντήρηση, η τελική έξοδος του δικτύου LSTM (Yn στο διάγραμμα) μπορεί να προβλέπει μελλοντικές καταστάσεις με βάση ιστορικά δεδομένα. Αυτή η ικανότητα επιτρέπει στους μηχανικούς συντήρησης να λαμβάνουν προληπτικές ενέργειες με βάση προβλέψεις για το πότε ένα εξάρτημα ενός συστήματος ψύξης θέρμανσης μπορεί να αποτύχει ή να χρειαστεί συντήρηση.

- Προσαρμοστικότητα σε νέα δεδομένα: Τα δίκτυα LSTM μπορούν να ενημερώνουν τους ιδίους τους μηχανισμούς με νέες πληροφορίες χωρίς να χρειάζεται να επανεκπαιδευτούν από την αρχή. Αυτό το χαρακτηριστικό είναι πολύ σημαντικό για συστήματα που οι συνθήκες μπορεί να αλλάζουν με την πάροδο του χρόνου, απαιτώντας από το μοντέλο να προσαρμόζεται συνεχώς στα νέα δεδομένα λειτουργίας.

Η αρχιτεκτονική LSTM, με τα πολλαπλά στρώματα και τις δυνατότητες διαδοχικής επεξεργασίας δεδομένων, αποτελεί ιδανική λύση για την αντιμετώπιση των προκλήσεων που θέτει η προγνωστική συντήρηση στα συστήματα ψύξης και θέρμανσης. Πιο σύνθετα μοντέλα τεχνικής νοημοσύνης με autoencoders ή υβριδικά αποφεύχθηκαν να χρησιμοποιηθούν πάρα του ότι μπορεί να είχαμε περισσότερα πλεονεκτήματα, για τον λόγω του ότι ήταν εκτός των απαιτήσεων της έρευνας της παρούσας διπλωματικής εργασίας για το συγκεκριμένο πεδίο εφαρμογής. Επιπλέον μια πιο σύνθετη υλοποίηση θα είχε νόημα να εξεταστεί σε επόμενο στάδιο, στη παρακολούθηση μιας μεγάλης βιομηχανικής μονάδας με πολλαπλάσια δεδομένα και στοιχεία.

5.3.3.2 Βιβλιοθήκες της Γλώσσας Python για την υλοποίηση του αλγορίθμου

Κατά τη διαδικασία σχεδιασμού του συστήματος της προγνωστικής συντήρησης για τον εξοπλισμό θέρμανσης και ψύξης, χρησιμοποιήθηκε ένα ευρύ φάσμα βιβλιοθηκών και εργαλείων ανοικτού κώδικα για τον χειρισμό των λειτουργιών δεδομένων, τη δημιουργία μοντέλων και τη λεπτομερή ρύθμιση αυτών των μοντέλων. Ακολουθεί επεξήγηση του τρόπου με τον οποίο χρησιμοποιείται κάθε βιβλιοθήκη:

- InfluxDB Client: Αυτή η βιβλιοθήκη ήταν απαραίτητη για την ανάκτηση δεδομένων χρονοσειρών από την InfluxDB, επιτρέποντας την αποτελεσματική εξαγωγή δεδομένων λειτουργίας από συστήματα ψύξης/ θέρμανσης που είναι αποθηκευμένα σε μια βάση δεδομένων χρονοσειρών.
- Pandas: Η Pandas έπαιξε καθοριστικό ρόλο στην επεξεργασία και την ανάλυση δεδομένων, τη δόμηση, τον καθαρισμό και την προετοιμασία των δεδομένων για μοντελοποίηση. Μετασχημάτισε τα ακατέργαστα δεδομένα αισθητήρων σε μορφή κατάλληλη για ανάλυση σε βάθος.
- Scikit-learn (sklearn): Χρησιμοποιήθηκε για την προ-επεξεργασία δεδομένων, συγκεκριμένα για την κλιμάκωση και την κανονικοποίηση των δεδομένων χρονοσειρών. Το Scikit-learn παρείχε επίσης εργαλεία για μετρήσεις επιδόσεων για την αξιολόγηση της αποτελεσματικότητας του μοντέλου.
- Joblib: Χρησιμοποιήθηκε για την αποθήκευση και τη φόρτωση αντικειμένων Python που αποθηκεύουν στοιχεία του μοντέλου, όπως οι κλίμακες δεδομένων, κάτι που είναι ζωτικής σημασίας για την ανάπτυξη του εκπαιδευμένου μοντέλου σε περιβάλλον παραγωγής.
- NumPy: Αναπόσπαστο στοιχείο για το χειρισμό αριθμητικών πράξεων σε πίνακες, υποστηρίζοντας λειτουργίες υψηλής απόδοσης σε δεδομένα, απαραίτητες τόσο για τις διαδικασίες εκπαίδευσης όσο και για τις διαδικασίες πρόβλεψης.
- TensorFlow Keras: Χρησίμευσε ως το βασικό πλαίσιο για τον σχεδιασμό και την εκπαίδευση μοντέλων νευρωνικών δικτύων μακράς βραχυπρόθεσμης μνήμης

(LSTM). Το Keras προσφέρει ένα υψηλού επιπέδου, φιλικό προς το χρήστη API για τη δημιουργία μοντέλων βαθιάς μάθησης στο πλαίσιο του TensorFlow.

- Keras Tuner: Αυτό το εργαλείο χρησιμοποιήθηκε για τον έλεγχο των υπερπαραμέτρων των μοντέλων LSTM για τη βελτιστοποίηση της απόδοσης, αυτοματοποιώντας την επιλογή των καλύτερων υπερπαραμέτρων για τα νευρωνικά δίκτυα για την ενίσχυση της ακρίβειας του μοντέλου.

Η ενσωμάτωση των βιβλιοθηκών αυτών κατέστησε δυνατή την πραγματοποίηση μιας στιβαρής ανάπτυξης του συστήματος προληπτικής συντήρησης. Διαπιστώθηκε ότι κάθε βιβλιοθήκη επιλέχθηκε λόγω της αποτελεσματικότητάς της στη διαχείριση δεδομένων, της ικανότητάς της για ισχυρή ανάλυση δεδομένων, της ικανότητάς της για την κατασκευή προηγμένων μοντέλων τεχνητής νοημοσύνης και της ικανότητάς της να βελτιστοποιεί τις παραμέτρους του μοντέλου προκειμένου να εγγυάται υψηλή ακρίβεια πρόβλεψης. Ένα σύστημα που είναι ικανό να αυξήσει σημαντικά τα διαστήματα συντήρησης και την αξιοπιστία λειτουργίας των συστημάτων ψύξης/ θέρμανσης δημιουργήθηκε με τη βοήθεια αυτής της σουίτας εργαλείων, τα οποία βοήθησαν από κοινού το σχεδιασμό του συστήματος.

Τα δύο προγράμματα σε γλώσσα Python 3 που ενσωματώνουν την τεχνητή νοημοσύνη για την κατασκευή και εκπαίδευσή του αλγορίθμου και την εφαρμογή προγνωστικής συντήρησης με εύρεση ανωμαλιών στις μετρήσεις σε πραγματικό χρόνο παρουσιάζονται στο

Παράρτημα Γ’ – Κώδικας .

6 Υλοποίηση πρότυπου πειραματικού περιβάλλοντος

6.1 Υλοποίηση πρότυπης πειραματικής διάταξης

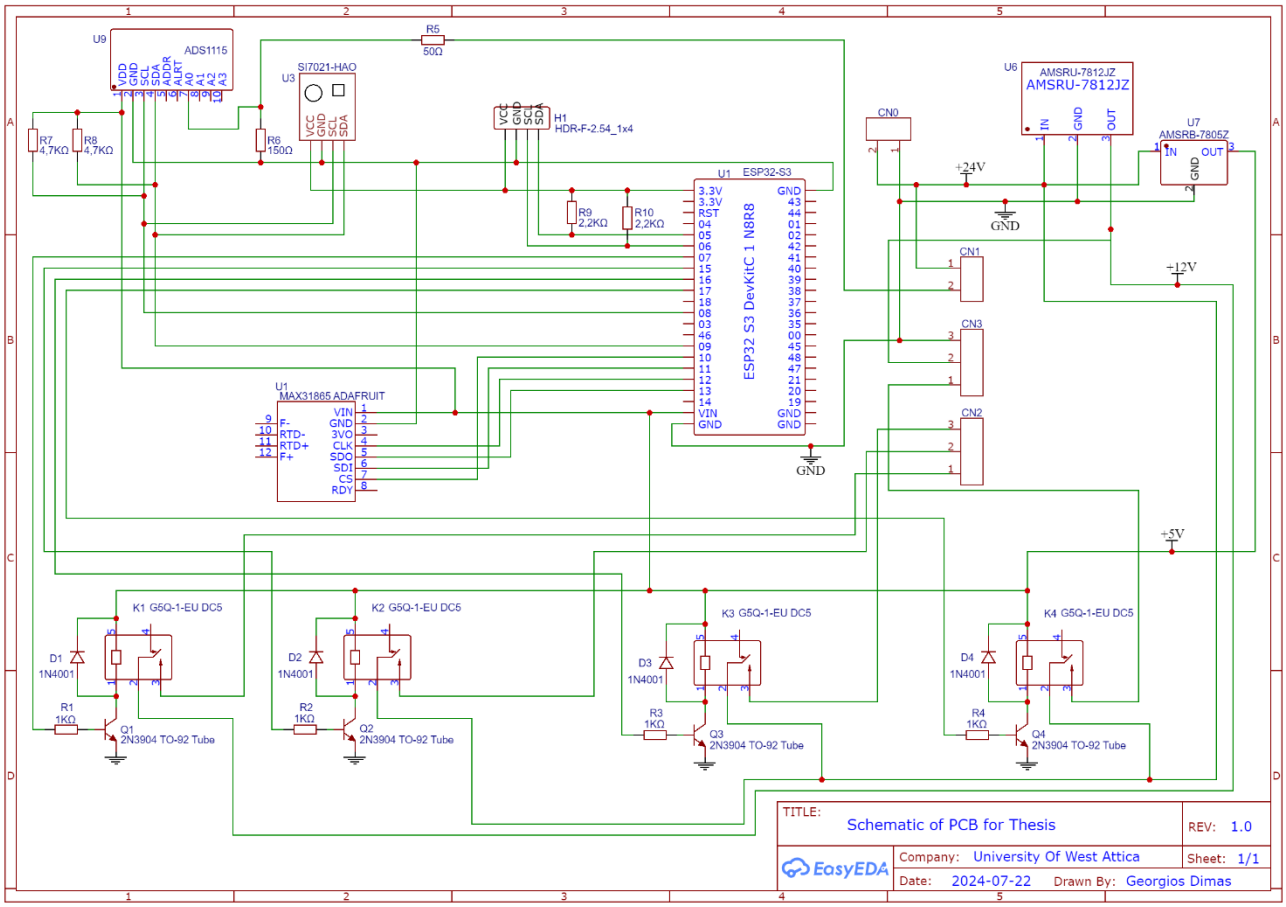
Με βάση την αρχιτεκτονική που παρουσιάστηκε στο προηγούμενο κεφάλαιο, υλοποιήθηκε ένα πρότυπο σύστημα που προσομοιάζει ένα ολοκληρωμένο σύστημα θέρμανσης χώρου με τη προσθήκη των απαραίτητων ηλεκτρονικών στοιχείων και των στοιχείων ελέγχου. Η κατασκευή ενός συστήματος χωρίζεται σε διάφορα βήματα όπως αυτά αποτελούν την προκατασκευή των επιμέρους στοιχείων και την ενοποίησή τους σε ένα ενιαίο περιβάλλον δοκιμών.

Αναφορικά με την μονάδα συλλογής και ανάλυσης δεδομένων σε τοπικό επίπεδο, ο μικροελεγκτής ESP-32 εκπονούσε δύο λειτουργίες: α) τον έλεγχο του συστήματος θέρμανσης, δηλαδή τη συνεχή μέτρηση της θερμοκρασίας στο υγρό και στο κλειστό δωμάτιο και την ενεργοποίηση των αντιστάσεων και της αντλίας έτσι ώστε η θερμοκρασία στο δωμάτιο να διατηρείται σε σταθερά θερμά επίπεδα, και β) τη συλλογή όλων των μετρήσεων και την αποστολή τους μέσω του διαδικτύου σε έναν διακομιστή με το πρωτόκολλό του MQTT. Ο διακομιστής αναλαμβάνει την συλλογή και επεξεργασία των δεδομένων με τεχνητή νοημοσύνη όπως αναλύθηκε.

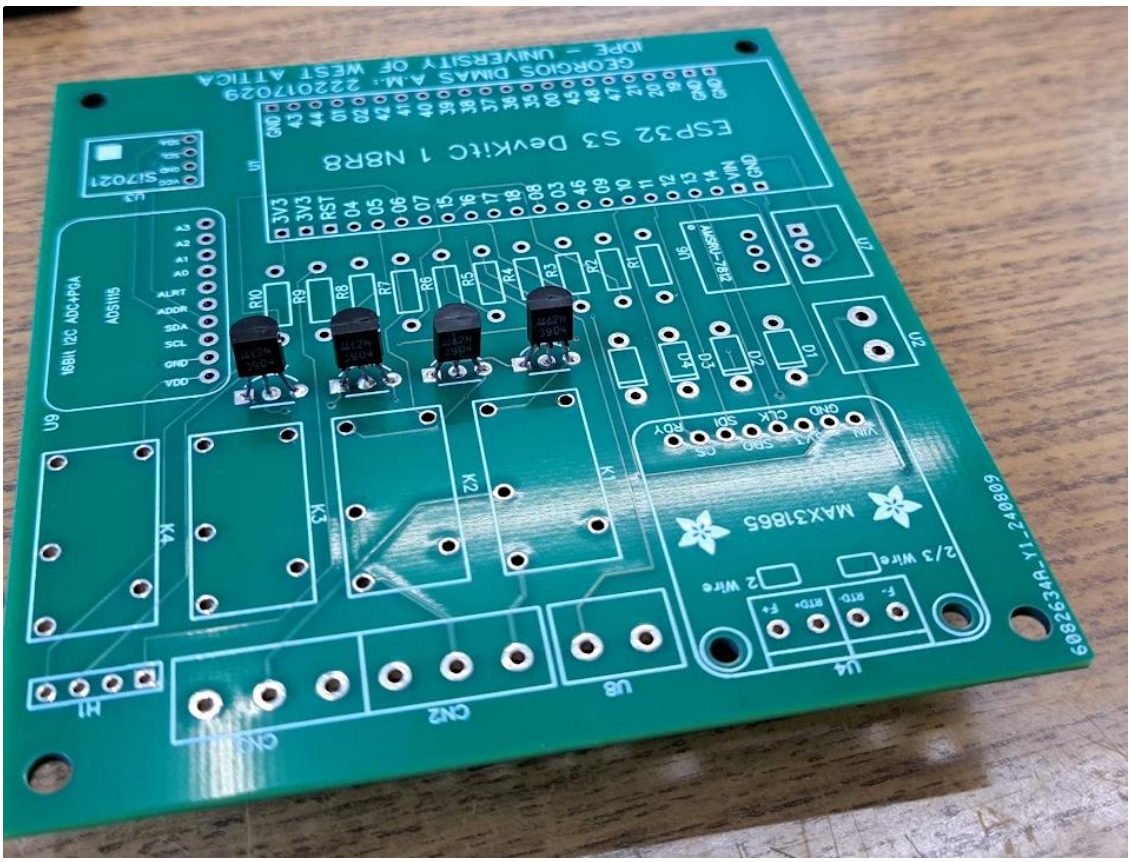
Αναφορικά με τον αλγόριθμο επεξεργασίας και ανάλυσης των δεδομένων στην νεφοϋπολογιστική υποδομή, επιλέχθηκε μηχανισμός μακράς βραχυπρόθεσμης μνήμης, ο οποίος έδωσε τη δυνατότητα να εκπαιδεύσουμε τον αλγόριθμο με τις πραγματικές τιμές σε βάθος χρόνου. Πιο συγκεκριμένα, ορίζοντας συγκεκριμένα χρόνο-βήματα και κανόνες εξάρτησης, εκπαιδεύσαμε τον αλγόριθμο αρχικά σε ομαλή λειτουργία του συστήματος, και μοντελοποιήθηκε η λειτουργία του σε χρονικές ακολουθίες των πέντε λεπτών, βάση επτά (7) μεγεθών "features" (θερμοκρασία του κυκλοφορών υγρού, πίεση του υγρού στη δεξαμενή, θερμοκρασία του θερμαινόμενου χώρου, υγρασία του θερμαινόμενου χώρου, θερμοκρασία του εξωτερικού περιβάλλοντος, ισχύς θέρμανσης, κατάστασης λειτουργίας της αντλίας κυκλοφορίας) και του πώς αυτά επηρεάζουν τα μεγέθη "targets" (θερμοκρασία και η πίεση του υγρού και η θερμοκρασία και υγρασία του εσωτερικού θερμαινόμενου χώρου). Πρέπει επίσης να σημειωθεί ότι τα μεγέθη "targets" επηρεάζονται και από άλλες περιβαλλοντικές παραμέτρους, όπως η μόνωση του χώρου θέρμανσης, η εσωτερική τριβή των εξαρτημάτων, οι υδραυλικές στενώσεις, τα βουλώματα στους σωλήνες, η ροή της αντλίας, η θερμοχωρητικότητα του υγρού κυκλοφορίας, η απόδοση του θερμαντήρα και του εναλλακτική νερού προς αέρα στο δωμάτιο, η ροή αέρα του ανεμιστήρα εκ του ψυγείου ή καλοριφέρ, κ.α. Σε μια ομαλή λειτουργία, όλες οι παραπάνω παράμετροι είναι σταθερές. Στα πλαίσια αυτά, ο μηχανισμός τεχνητής νοημοσύνης που υιοθετήθηκε σε αυτή τη διπλωματική εργασία παρακολουθεί τις τιμές των επτά "features", υπολογίζει/προβλέπει τις τιμές για τα τέσσερα "targets", και τελικά εκτιμά αν υπάρχει σφάλμα στη συνοχή των πραγματικών δεδομένων την παρούσα στιγμή. Αυτό επιτυγχάνεται μετρώντας συνεχώς το σφάλμα ανάμεσα στις πραγματικές και τις εκτιμηθείσες τιμές των targets, μια διαφοροποίηση ανά μέγεθος μπορεί να οδηγήσει στο συμπέρασμα ότι υπάρχει παρεμβολή στο σύστημα, το οποίο σημαίνει αλλοίωση των χαρακτηριστικών του συστήματος που αναφέρθηκαν στην αρχή της παραγράφου.

6.1.1 Υλοποίηση της μονάδα συλλογής και ανάλυσης δεδομένων σε τοπικό επίπεδο

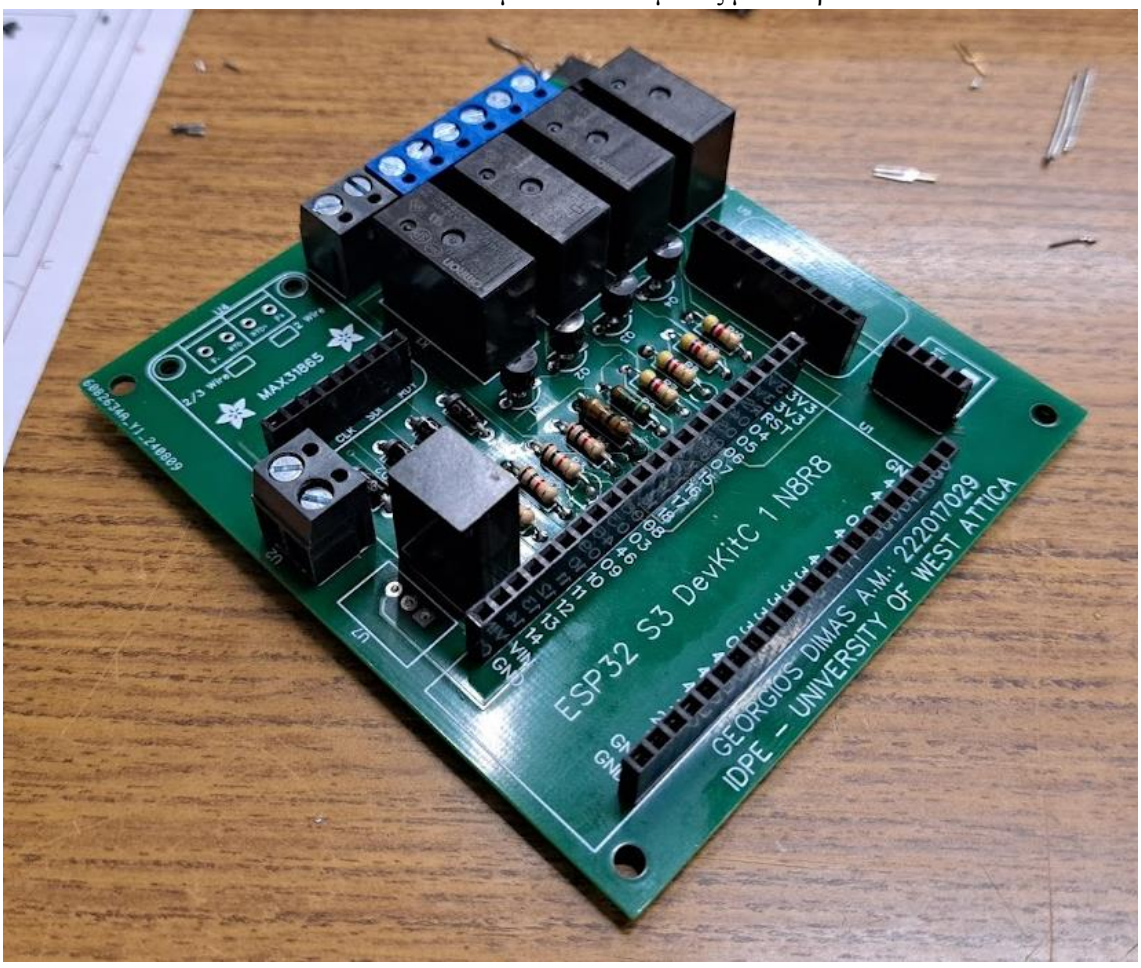
Αρχικά σχεδιάστηκε όπως φαίνεται στο παρακάτω διάγραμμα και στην Εικόνα 30 ένα ηλεκτρονικό κύκλωμα στο οποίο συνδέονται μεταξύ τους ο μικροελεγκτής, τα περιφερειακά στοιχεία εισόδων, οι μετατροπείς των αναλογικών σημάτων καθώς και λοιπά ηλεκτρονικά στοιχεία όπως αντιστάσεις, ρελέ και άλλα. Στη συνέχεια με την βοήθεια του προγράμματος EasyEDA, σχεδιάστηκε το τυπωμένο κύκλωμα – πλακέτα – το οποίο και παραγγέλθηκε προς κατασκευή στην Κίνα, μέσω της διαδικτυακής πλατφόρμας “JLCPCB”. Παρακάτω στην Εικόνα 31 φαίνεται η πλακέτα χωρίς υλικά ενώ στην Εικόνα 32 έχουν τοποθετηθεί τα βασικά παθητικά εξαρτήματα· αντιστάσεις, ρελέ, υποδοχές. Η τελική φάση συναρμολόγησης, όπου στην τοποθετημένη στην βάση πλακέτα έχουν τοποθετηθεί και τα εξαρτήματα λογικής· αισθητήρια, μετατροπείς και μικροελεγκτής φαίνεται στην Εικόνα 33.



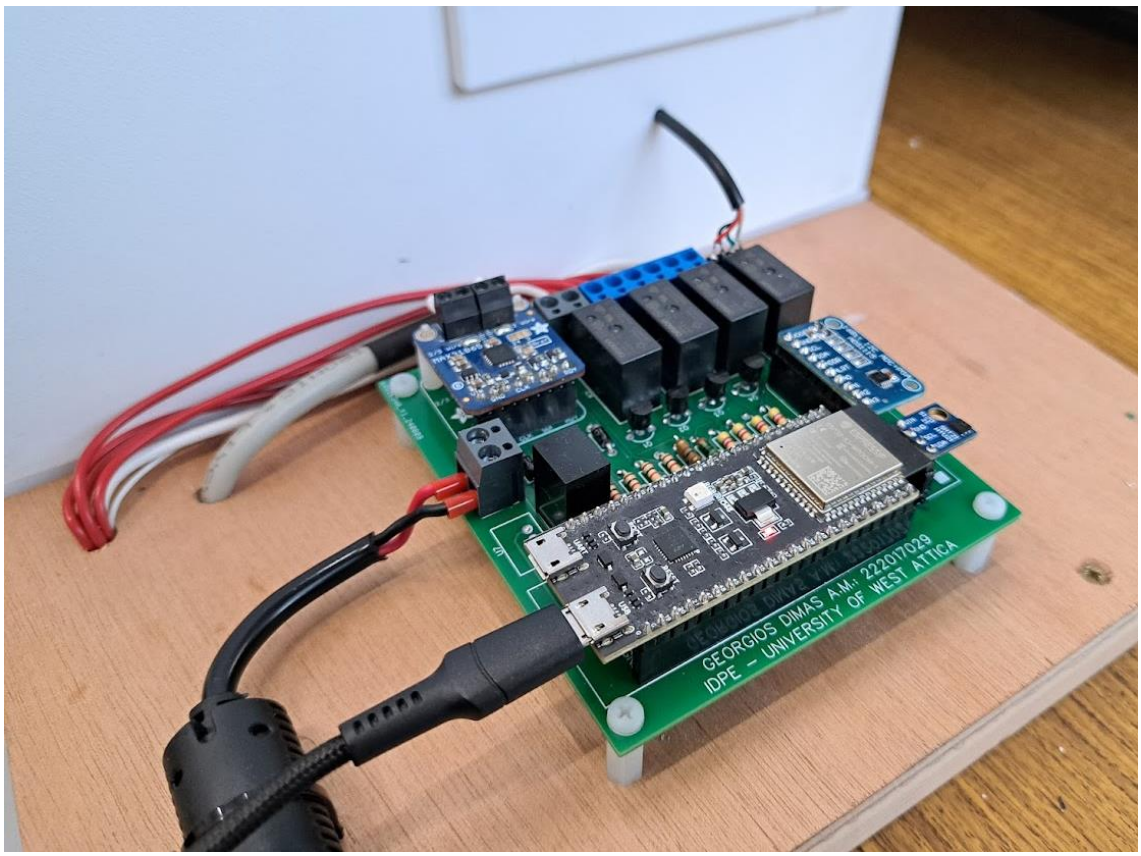
Εικόνα 30: Σχεδιάγραμμα ηλεκτρονικού κυκλώματος



Εικόνα 31: Πλακέτα τυπωμένου κυκλώματος με τα πρώτα υλικά



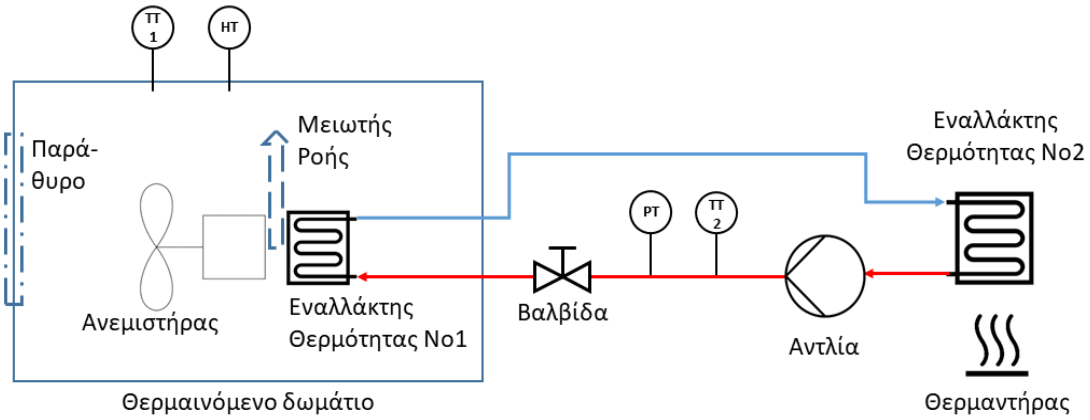
Εικόνα 32: Πλακέτα τυπωμένου κυκλώματος με τα βασικά παθητικά ηλεκτρονικά εξαρτήματα



Εικόνα 33: Πλήρως συναρμολογημένη πλακέτα

6.1.2 Υλοποίηση πρότυπου περιβάλλοντος εξομοίωσης και διεξαγωγής πειραμάτων

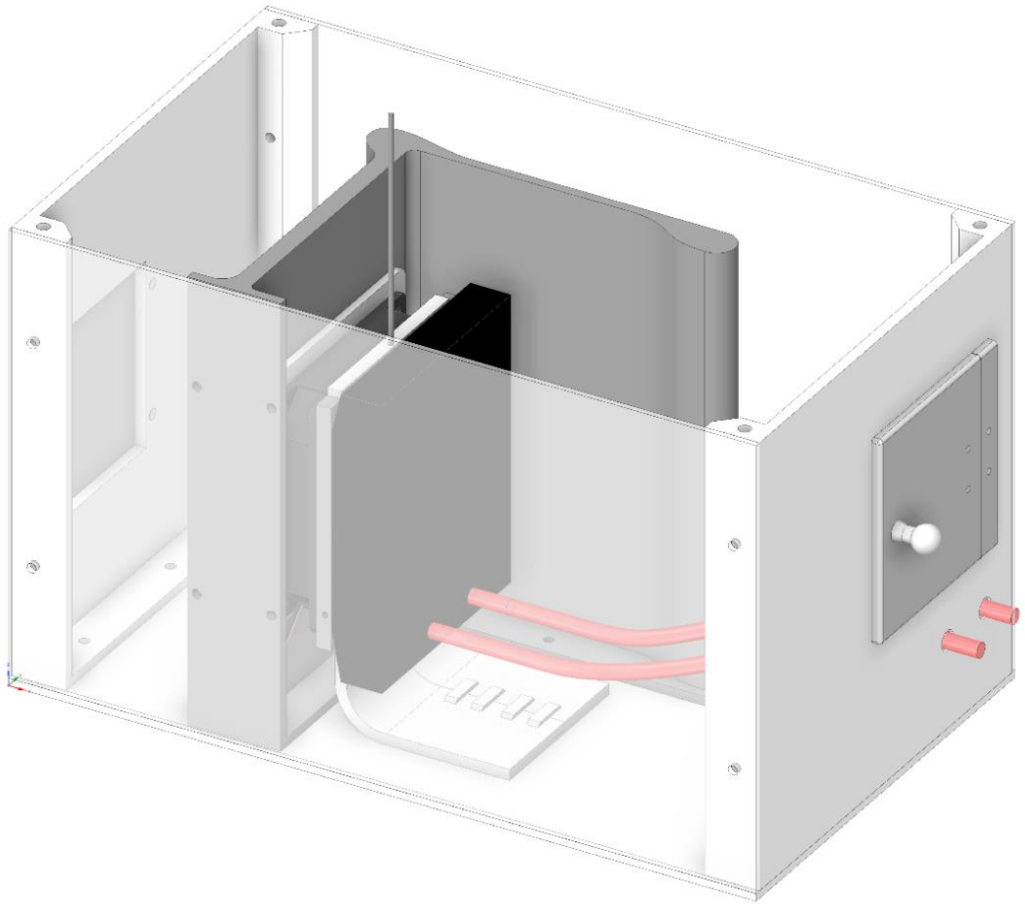
Με σκοπό την λήψη πραγματικών μετρήσεων που θα εκπαιδεύσουν τον αλγόριθμο υλοποιήθηκε ένα κλειστό κύκλωμα θέρμανσης εσωτερικού χώρου. Μολονότι η διπλωματική εργασία αφορά συστήματα ψύξης ή και θέρμανσης, υλοποιείται μόνο το σενάριο της θέρμανσης, καθώς αυτό είναι αρκετό για την απόδειξη της θεωρίας λειτουργίας της εφαρμογής. Όπως είναι δε γνωστό από τους νόμους της φυσικής η θερμότητα ρέει από το θερμό προς το κρύο. Επίσης μια διάταξη ψύξης είναι πιο δύσκολή στην υλοποίηση της από μια διάταξη θέρμανσης. Για τους λόγους αυτούς επιλέχθηκε η θέρμανση ενός χώρου και όχι η ψύξη του. Ως πλέον καλύτερο παράδειγμα δοκιμής επιλέχθηκε η σχεδίαση ενός κλειστού χώρου θερμικά μη αγώγιμου με αντοχή σε θερμοκρασία έως 80 βαθμούς κελσίου, και η χρήση ενός συστήματος υδρόψυξης υπολογιστή για την προσομοίωση του “fan coil”. Στην όλη διάταξη όπως αυτή απεικονίζεται στο παρακάτω σχεδιάγραμμα στην Εικόνα 34, έχουν προστεθεί αισθητήρια, βαλβίδες και άλλοι μηχανισμοί για την μέτρηση της συμπεριφοράς του συστήματος και την παρεμπόδιση της λειτουργίας του.



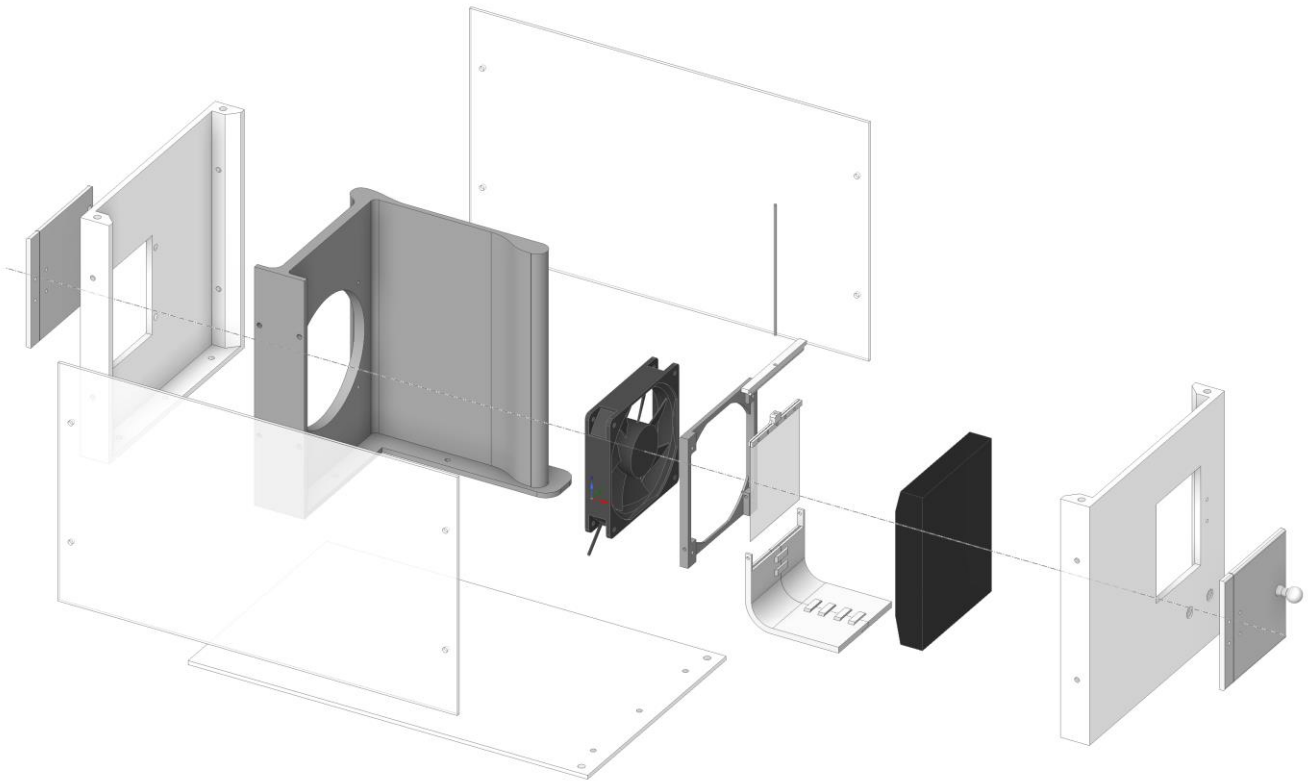
PT: Pressure Transmitter – Αισθητήρας Πίεσης
TT: Temperature Transmitter – Αισθητήρας Θερμοκρασίας
HT: Humidity Transmitter – Αισθητήρας Υγρασίας

Εικόνα 34: Διάγραμμα αρχικού πειραματικού μοντέλου δοκιμών

Με σκοπό την κατασκευή και τη συναρμολόγηση του πειραματικού μοντέλου χρειάσθηκε να γίνει επιλογή υλικών που θα είναι ικανά να υποστηρίξουν τις απαιτήσεις της σχεδίασης. Κατά την σχεδίαση του μοντέλου προέκυψε η ανάγκη κατασκευής πολύπλοκων μηχανικών εξαρτημάτων τα οποία επιλέχθηκε να κατασκευαστούν με τη μέθοδο της προσθετικής κατασκευής με 3D εκτυπωτή και συγκεκριμένα τον Bambu LAB. Για την σχεδίαση των διαφόρων εξαρτημάτων της κατασκευής, του συνόλου/ assembly και την κατασκευή του ψηφιακού δίδυμου του πειραματικού μοντέλου επιλέχθηκε το πρόγραμμα CAD, "Design Spark Mechanical 6.0" της "Space Claim". Παρακάτω στην Εικόνα 35, εμφανίζεται το assembly σχέδιο του μοντέλου μέσω του προαναφερθέντος προγράμματος. Ενώ στην Εικόνα 36 βλέπουμε ένα σχέδιο συναρμολόγησης «explosion chart» που δείχνει τα εξαρτήματα της κατασκευής και πως αυτά ενώνονται μεταξύ τους για την δημιουργία του μοντέλου. Η κατασκευή αποτελείται από ένα κλειστό χώρο με μια ειδική βάση του ψυγείου και του ανεμιστήρα. Η βάση είναι ειδικά σχεδιασμένη ώστε να είναι και διαχωριστικό. Η αεροδυναμική της σχεδίαση της επιτρέπει στον ανεμιστήρα να ανακυκλώνει πλήρως τον αέρα εντός του δωματίου. Μια συνεχή ροή του αέρα, σε κυκλική κατεύθυνση επιτρέπει σε αυτόν να λαμβάνει όλη τη θερμότητα από το ψυγείο. Αξιοποιώντας τις δυνατότητες της προσθετικής κατασκευής, ήταν δυνατόν να χρησιμοποιηθούν εξαρτήματα με περίεργα γεωμετρικά σχήματα που θα ήταν ιδιαίτερα δύσκολο και ασύμφορο να κατασκευαστούν με συμβατικές μεθόδους κατεργασίας. Στο μοντέλο ενσωματώθηκαν και κάποια στοιχεία που έχουν σκοπό την εκ θέλησης προσομοίωση βλαβών ή ανωμαλιών στην λειτουργία. Δυο παράθυρα στο ίδιο άξονα με τον ανεμιστήρα, προκαλούν θελημένες απώλειας θερμότητας, ενώ μια κατασκευή με ένα ολισθαίνον φύλλο PET προσομοιάζει ένα βουλωμένο ψυγείο.



Εικόνα 35: CAD Σχέδιο Μοντέλου



Εικόνα 36: Σχέδιο Συναρμολόγησης (Explosion Chart)

Ως προς την επιλογή των υλικών, αυτά έπρεπε να πληρούν τις προδιαγραφές της εφαρμογής, ως προς διάφορες συνθήκες και παραμέτρους. Η αντοχή σε θερμοκρασία και στο χρόνο είναι δυο απαραίτητα χαρακτηριστικά, για αυτό το λόγω ανάλογα το εξάρτημα, επιλέχθηκαν τα ακόλουθα υλικά με βάση την ποιότητα κατασκευής και το κόστος, όπως αναλύονται παρακάτω:

- Θερμαινόμενο Δωμάτιο Διαστάσεων 400 χ 240 χ 240 χιλιοστά: Ως κύριο υλικό κατασκευής επιλέχθηκε το Plexiglas με αντοχή σε θερμοκρασία έως 130C σε γαλακτερό πάχους 3mm για την οριοθέτηση του πατώματος και των δύο μακριών τοίχων, ενώ σχεδιάστηκε το καπάκι/ ταβάνι να είναι διάφανο πάχους 4mm. Έκαστος τα κομμάτια κόπηκαν από τον προμηθευτή σε διαστάσεις 400χ 240 mm με σταθερό τροχό. Το δεύτερο υλικό που επιλέχθηκε ήταν το ABS αντοχής 80-90C σε χρώμα λευκό για χρήση στον 3D εκτυπωτή. Άλλα υλικά όπως το αλουμίνιο ή το ξύλο δεν επιλέχθηκαν διότι θα ήτανε ακριβότερη η κατασκευή και δυσκολότερη η κατεργασία τους. Τα πλαστικά υλικά χρησιμοποιήθηκαν επιπλέον και για τον λόγω ότι είναι θερμικά μη αγώγιμα και περιορίζεται έτσι η απώλεια θερμότητας εκ του δωματίου προς τον περιβάλλοντα χώρο. Η μέθοδος του 3D printing διαδραμάτισε καθοριστικό ρόλο στην επιλογή ως βασικού υλικών, πλαστικά αντοχής σε υψηλές θερμοκρασίες.
- Κλειστό κύκλωμα θέρμανσης: Επιλέχθηκε η αγορά συστήματος υδρόψυξης υπολογιστή και ο κατακερματισμός / μετατροπή του για να καλύψει τις ανάγκες της σχεδίασης. Η υδρόψυξη ενός υπολογιστή είναι μια συσκευή κλειστού κυκλώματος που περιλαμβάνει αντλία, επιφάνεια εναλλάκτη θερμότητας η οποία τοποθετείται στον επεξεργαστή, ανεμιστήρα και εναλλάκτη θερμότητας αέρα-νερού κοινός αναφερόμενου ως ψυγείο. Η υδρόψυξη ενός υπολογιστή θεωρητικά είναι σχεδιασμένη από κατάλληλα υλικά ώστε να αντέχει τις υψηλές θερμοκρασίες που φτάνει η CPU ενός υπολογιστή. Το μοντέλο που επιλέχθηκε είναι το “Enermax LIQMAX III 120”, οπού και απεικονίζεται στην παρακάτω Εικόνα 37. Όπως θα αναφερθεί και στη συνέχεια, η έτοιμη αγορασμένη διάταξη της υδρόψυξης κατακερματίστηκε και συναρμολογήθηκε εκ νέου με τρόπο ώστε να εξυπηρετεί την θέρμανση του δωματίου και όχι την ψύξη ενός επεξεργαστή. Ο ανεμιστήρας και το ψυγείο λειτουργούν ως “Fan Coil” κλιματισμού και η αντλία με τον εναλλάκτη επιφανείας θερμαίνει μέσω αντιστάσεων θέρμανσης το υγρό και το κυκλοφορεί στο σύστημα.

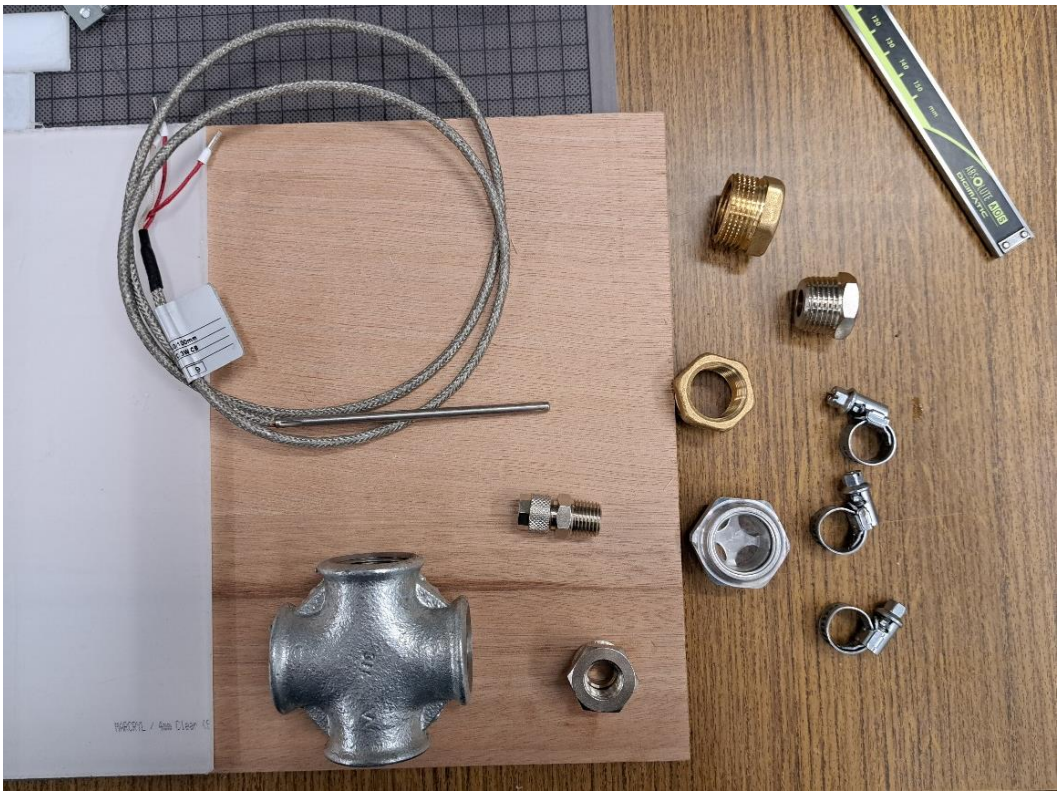
Εικόνα 37: Υδρόψυξη Enermax LIQMAX III 120 [55]

- Σωλήνες κυκλώματος θέρμανσης: Στη συνέχεια του αποκερμαρματισμού της υδρόψυξης και της ανάγκης της σύνθετου κυκλώματος και της τοποθέτησης αισθητηρίων επιλέχθηκε εύκαμπτος σωλήνας χαμηλής πίεσης που πωλείται με το μέτρο. Κατάλληλο υλικό θεωρείται το Teflon® (PTFE). Το PTFE αναφέρεται στην αγορά ως το ελαστικό με τις καλύτερες αντοχές στα χημικά, είναι εξαιρετικά αδρανές και ευσταθές με θερμοκρασία λειτουργία από τους -60 έως +260C. [56] Το συγκεκριμένο υλικό είναι αρκετά κοινό στην αγορά και χρησιμοποιείται σε συστήματα πεπιεσμένου αέρα, συνήθως μέχρι 10bar, οπότε και θεωρείται ανθεκτικό στη πίεση. Επιπλέον το υλικό είναι διάφανο και αυτό επιτρέπει να φαίνεται το υγρό κυκλοφορίας και προσδίδει μια ομορφιά στο μοντέλο.

Εικόνα 38: Σωλήνας Teflon® (PTFE) [56]

- Υδραυλικά εξαρτήματα και αισθητήρια δικτύου υγρού: Η σύνδεση των αισθητηρίων μπορεί προτείνεται να γίνει στην έξοδο της αντλίας και μετά τον θερμαντήρα του υγρού. Δεδομένου ότι υπάρχει η ανάγκη σύνδεση 2 αισθητηρίων και οπτικός έλεγχος, θα βοήθαγε η ύπαρξη κοινού «κολλεκτέρ» - δεξαμενής στην οποία θα συνδέονται τα αισθητήρια. Επιλέχθηκε λοιπόν τοποθέτηση ενός υδραυλικού

σταυρού G 3/4 της ίντσας με εσωτερική χωρητικότητα 25 mL και επιπλέον η διάνοιξη 5^{ου} σπειρώματος G1/4 της ίντσας στο κέντρο του σταυρού, η χρήση των κατάλληλων συστολών επιτρέπει την σύνδεση των αισθητήριων, ενώ δυο βάνες επιτρέπουν την δοκιμή και τη λειτουργεία του μοντέλου. Ως υλικά επιλέχθηκαν κοινά υλικά του εμπορείου όπως Γαλβανιζέ Σίδηρος, Ανοξείδωτο ατσάλι και Ορείχαλκος. Τα αισθητήρια που επιλέχθηκαν είναι ειδικά για την μέτρηση ρευστών/ υγρών και ανήκουν στην οικογένεια των αισθητηρίων του βιομηχανικού αυτοματισμού. Η θερμοκρασία μετριέται με αισθητήριο PT1000 3 αγωγών τοποθετημένο σε στέλεχος σωλήνα (probe) πάχους 4mm με καλώδιο fiberglass αντοχής μέχρι του 260C. Ενώ η πίεση μετριέται με αισθητήριο 1.6 bar abs με έξοδο σήματος 4..20mA.



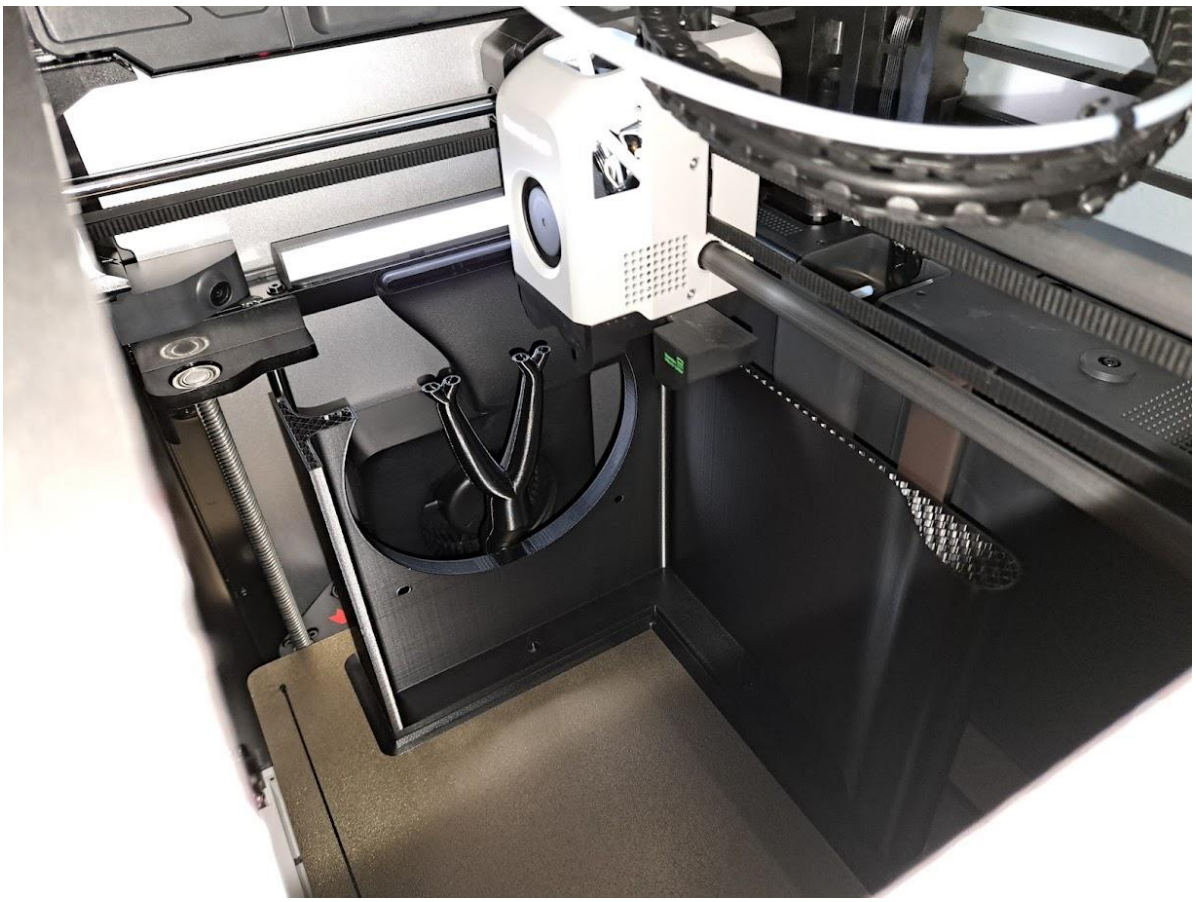
Εικόνα 39: Υδραυλικά εξαρτήματα και αισθητήριο θερμοκρασίας νερού

Εικόνα 40: Αισθητήριο Πίεσης [57]

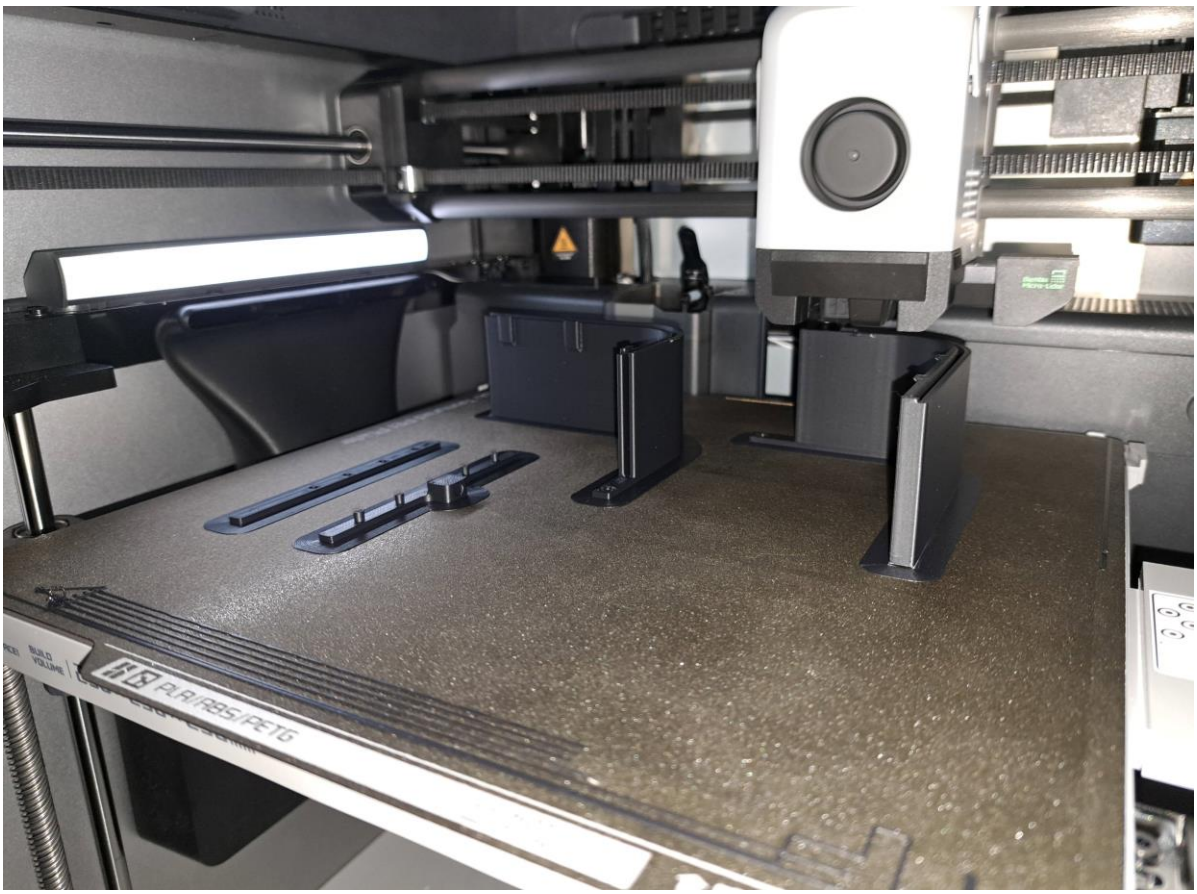
- Θερμαινόμενο υγρό: Σαν υγρό κυκλοφορίας στο σύστημα θέρμανσης επιλέχθηκε ροζ αντιψυκτικό υγρό για ψυγεία αυτοκινήτου, γνωστό και ως «*παραφλού*». Το συγκεκριμένο υγρό επιλέχθηκε κυρίως επειδή είναι χρωματιστό και θα είναι οπτικά αντιληπτό μέσα από τους σωλήνες PTFE. Το υγρό είναι αντοχής -40 έως +105C. Χρησιμοποιήθηκαν συνολικά 120ml «*παραφλού*» ως υγρό κυκλοφορίας.

Στη συνέχεια της σχεδίασης ακολουθεί η προμήθεια των υλικών και η κατασκευή μέρους των εξαρτημάτων με 3D εκτυπωτή. Παρακάτω στην Εικόνα 42 και Εικόνα 43, φαίνεται η διαδικασία εκτύπωσης των 3D εξαρτημάτων με τον εκτυπωτή BambuLab X1 που απεικονίζεται στην Εικόνα 41.

Εικόνα 41: 3D Εκτυπωτής BambuLab X1



Εικόνα 42: Εκτύπωση βάσης στήριξης fan coil από νήμα ABS



Εικόνα 43: Εκτύπωση περιφερειακών αντικειμένων από νήμα ABS

Έπειτα από την εκτύπωση – κατασκευή των δομικών εξαρτημάτων του μοντέλου και στη συνέχεια του αποκερματισμού της υδρόψυξης της Εικόνα 44 ακολουθεί η εκ νέου δόμηση του ψυγείου και του ανεμιστήρα πάνω στη βάση και τον διαχωριστή του αέρα.

Εικόνα 44: Ανεμιστήρας και εναλλάκτης αποκερματισμένης υδρόψυξης

Στις παρακάτω φωτογραφίες της Εικόνα 45 α’ και β’ φαίνεται το πλαίσιο στήριξης μεταξύ του ανεμιστήρα και του ψυγείου. Το πλαίσιο αυτό λειτουργεί ως αποστάτης και χρησιμεύει στο να κινείται εντός αυτού ένα φύλλο ανακυκλωμένου πλαστικού PET με αντοχή σε θερμοκρασία έως +105C.

Εικόνα 45α’ και 45β’: Πλαίσιο στήριξης με κλειστή και ανοιχτή κουρτίνα - ρολό

Ένα ειδικά σχεδιασμένο clipper από ABS συγκρατεί το εύκαμπτο φύλλο πλαστικού, ενώ ένα ραβδί 2,5 χιλιοστών από ορείχαλκο, κολλημένο με εποξική κολλά 2 συστατικών, βοηθάει στο να μετακινείται και να ολισθαίνει το φύλλο ως κάθετη κουρτίνα ή σαν ρολό. Ένα εκτυπωμένο πλαστικό εξάρτημα σε καμπύλο σχήμα, βοηθάει στην οδήγηση του φύλλου και στο να είναι κριμένο όταν δεν το χρειαζόμαστε, η συναρμογή των παραπάνω φαίνεται στην Εικόνα 46.

Εικόνα 46: Συναρμολογημένο πλαίσιο

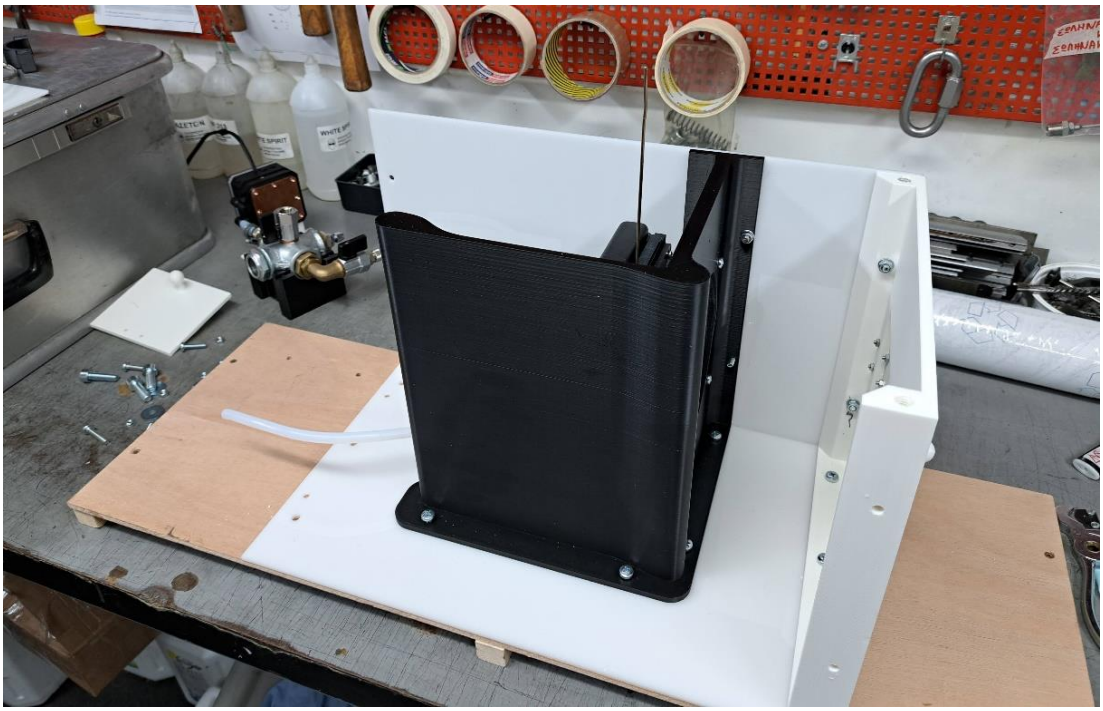
Η συναρμολόγηση του μοντέλου γίνεται σύμφωνα με το σχέδιο της Εικόνα 36 και οι παρακάτω φωτογραφίες απεικονίζουν τη διαδικασία του μονταρίσματος του εσωτερικού μέρους της μονάδας θέρμανσης.

Εικόνα 47: Συναρμολογημένη βάση μπλοκ θέρμανσης – όψη εναλλάκτη

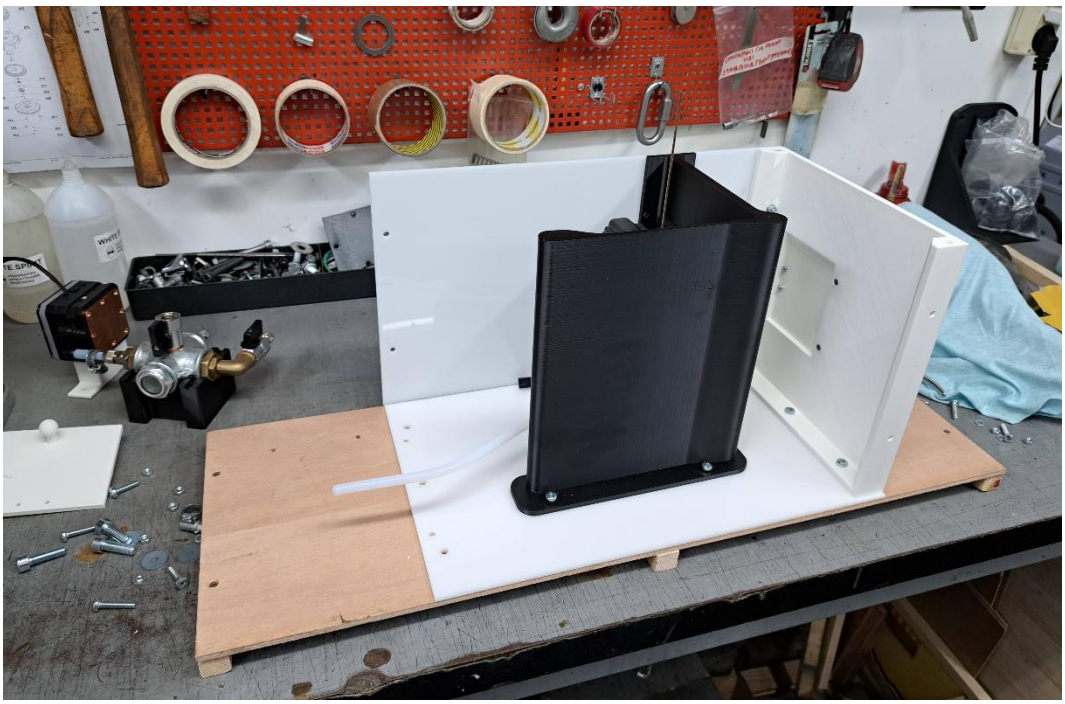
Εικόνα 48: Συναρμολογημένη βάση μπλοκ θέρμανσης – όψη ανεμιστήρα

Η όλη κατασκευή πρέπει να είναι σε μια βάση η οποία θα περιλαμβάνει όλα τα εξαρτήματα για λόγους εργονομίας με βέλτιστη διαχείριση χώρου. Έτσι λοιπόν χρησιμοποιήθηκε ένα φύλλο κόντρα πλακέ διαστάσεων 24 επι 65 εκατοστά με 3 πόδια από ξύλο οξιάς 20 επι 22 εκατοστά σταθεροποιημένα με ξυλόκολλα και ξυλόβιδες.

Εικόνα 49: Βάση πειραματικού μοντέλου

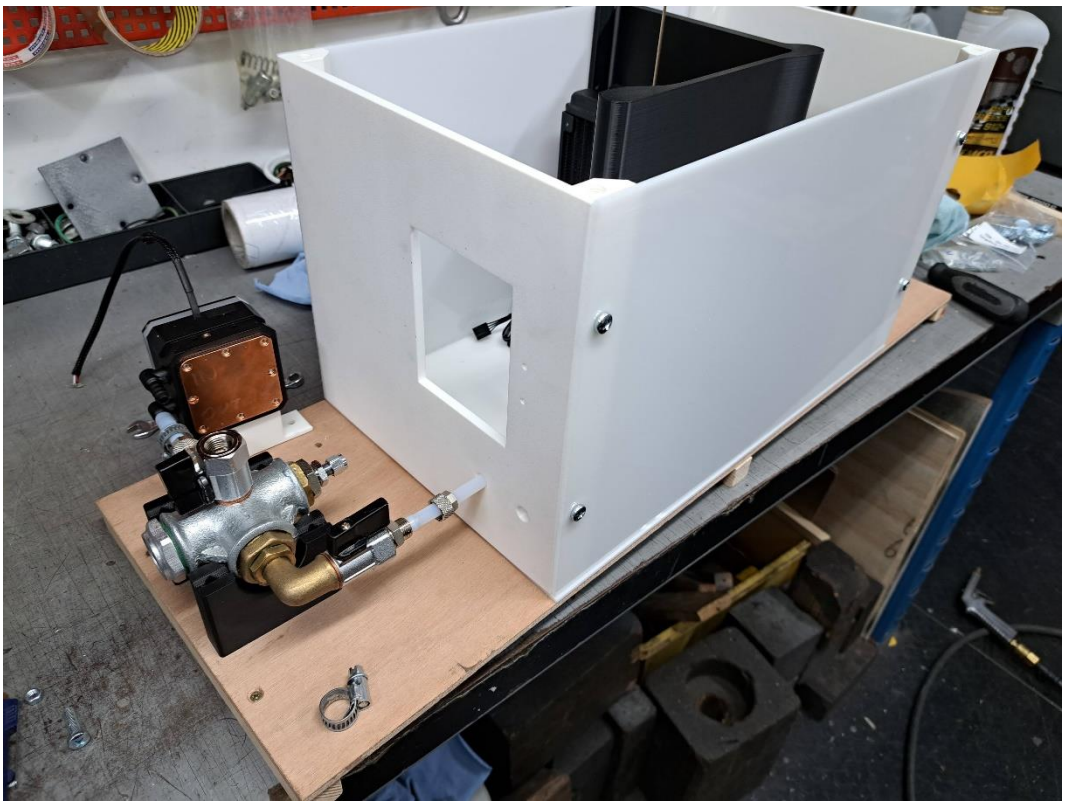


Εικόνα 50: Ενδιάμεσο στάδιο συναρμολόγησης – όψη 1



Εικόνα 51: Ενδιάμεσο στάδιο συναρμολόγησης – όψη 2

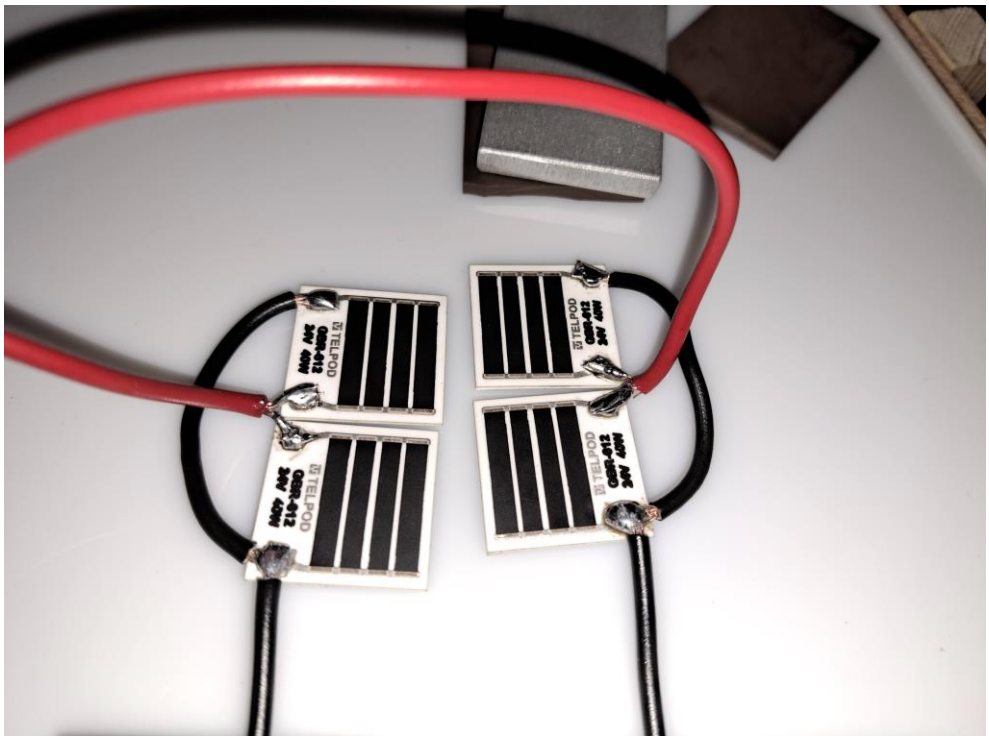
Στη συνέχεια όλα τα δομικά εξαρτήματα αρχίζουν να μπαίνουν στη θέση τους και μετρικές γαλβανιζέ βίδες χρησιμοποιούνται στη συναρμολόγηση. Καθώς τοποθετούνται τα πλαστικά περιβλήματα, θέση παίρνουν στην κατασκευή οι σωλήνες, ο υδραυλικός σταυρός με τα αισθητήρια και η αντλία της υδρόψυξης με τον εναλλάκτη. Ο υδραυλικός σταυρός στηρίζεται με ειδική βάση κατασκευασμένη από τον 3D printer και ακινητοποιείται. Οι παρακάτω φωτογραφίες απεικονίζουν ολόκληρη την μηχανολογική κατασκευή στη φάση της συναρμολόγησης.



Εικόνα 52: Ενδιάμεσο στάδιο συναρμολόγησης – όψη 3

Εικόνα 53: Ενδιάμεσο στάδιο συναρμολόγησης - κύκλωμα υγρού

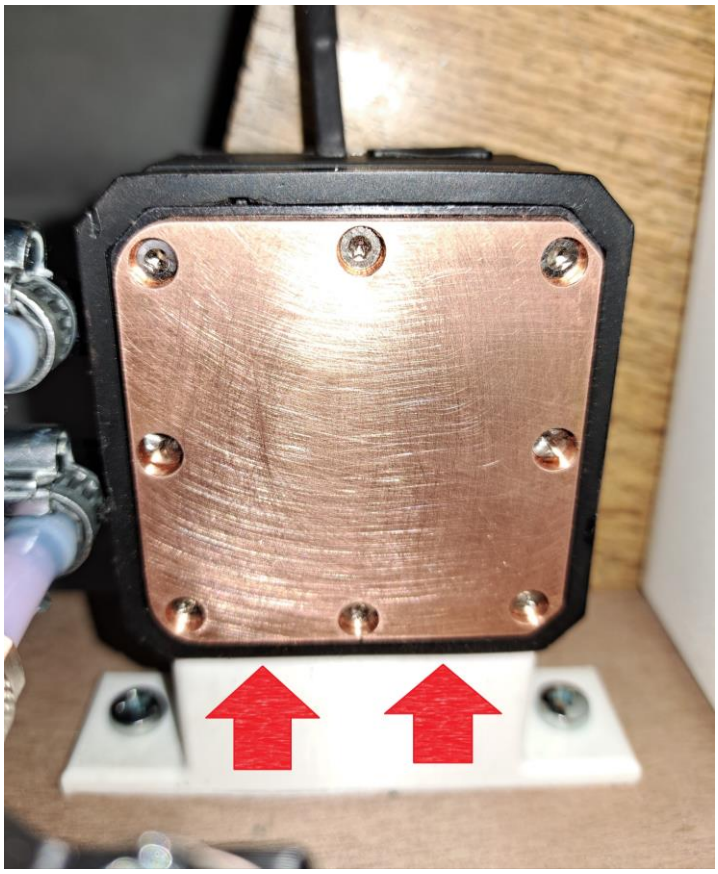
Τέλος για τον σκοπό της θέρμανσης του νερού επιλέχθηκαν 4 κεραμικές αντιστάσεις θέρμανσης της εταιρείας TELPOD, με κωδικό προϊόντος “GBR-612/24/40-1”. Έκαστος κάθε αντίσταση είναι 40 watt με τροφοδοσία 24 Volt, προσφέροντας συνολικά θερμότητα ίση με 160 Watt μείον τις απώλειες. Η σύνδεση των αντιστάσεων φαίνεται παρακάτω στην Εικόνα 54.



Εικόνα 54: Αντιστάσεις θέρμανσης TELPOD σε παράλληλη σύνδεση

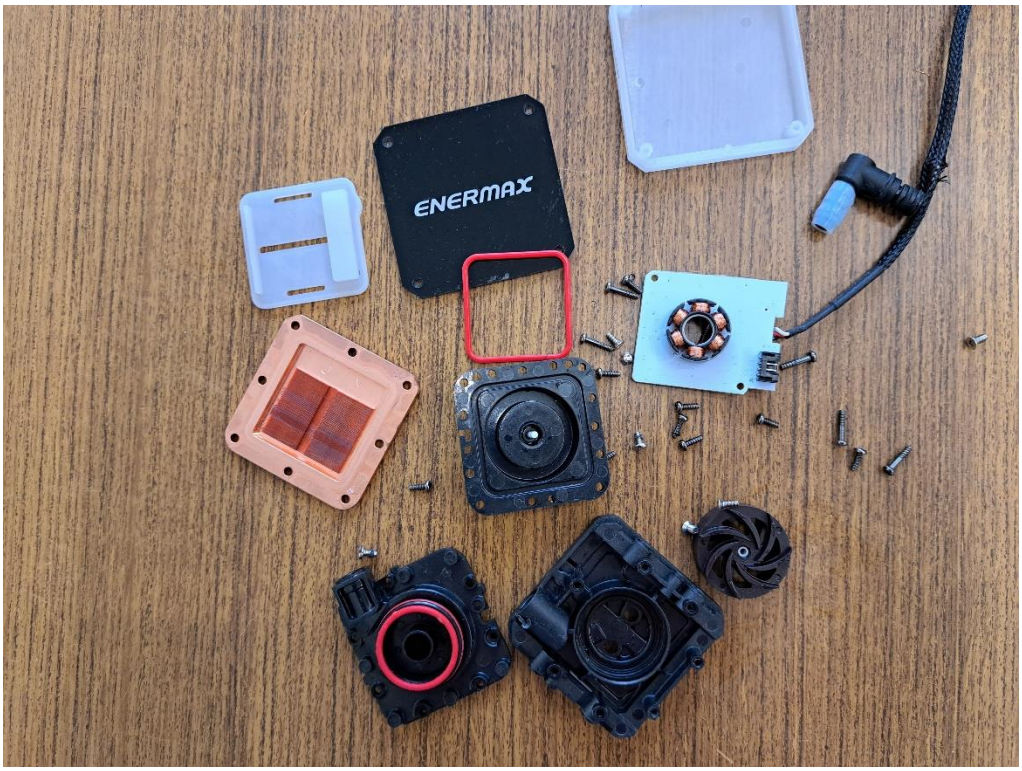
6.1.3 Δοκιμή Συστήματος – παρουσίαση διαρροής

Η δοκιμή του συστήματος είναι μια απαραίτητη διαδικασία κατά την οποία ελέγχεται με αναλογικά, ψηφιακά και οπτικά αισθητήρια η συμπεριφορά του συστήματος. Οι αντιστάσεις θερμαίνουν το υγρό κυκλοφορίας, η αντλία επιτρέπει την κυκλοφορία του υγρού, ενώ το ψυγείο με τον ανεμιστήρα εντός του δωματίου μεταφέρει την θερμότητα από το υγρό στο κλειστό δωμάτιο. Κατά την δοκιμή χρησιμοποιήθηκε θερμοζεύγος σε πολύμετρο fluke για την μέτρηση της θερμοκρασίας του δωματίου, και μετρήθηκε η πίεση στο νερό με μανόμετρο 0 έως 1 bar g. Η δοκιμαστική λειτουργεία του συστήματος διήρκησε 5 λεπτά, η λειτουργεία ήταν άριστη, ο αέρας που έβγαινε μέσα από το ψυγείο είχε θερμοκρασία είχε θερμοκρασία έως 65 βαθμούς και το υγρό θερμάνθηκε μέχρι περίπου μέχρι τους 70C. Η λειτουργία του κυκλώματος ήταν απόλυτα ικανοποιητική και ικανοποίησε πλήρως την θεωρία της πρόβλεψης βλαβών με χρήση αισθητηρίων. Ο χειρισμός των μηχανισμών παρεμπόδισης, της βάνα στην κατάθλιψη μετά το κολλεκτέρ, των παραθύρων και του ολισθαίνον φύλλου πλαστικού «κουρτίνα», έδωσε στα αισθητήρια μετρήσεις οι οποίες θα μπορούσαν σε συνδυασμό μεταξύ τους να χαρακτηρίσουν την συμπεριφορά του συστήματος ως μη ομαλή. Η πίεση στο υγρό αυξανόταν καθώς αυτό θερμαινόταν, ενώ επίσης αυξανόταν, αλλά με ακαριαία ταχύτητα αν έκλεινε η βάνα στην κατάθλιψη. Δυστυχώς όμως η δοκιμαστική λειτουργία του κυκλώματος θέρμανσης δεν κράτησε πολύ. Η αντλία με τον θερμαντήρα του νερού παρουσίασε διαρροή και υπήρξε μικρή αλλά συνεχή απώλεια του νερού. Η κινέζικη και η κακή ποιότητα κατασκευής και σχεδίασης της αντλίας κυκλοφορίας πρόδωσαν το πείραμα. Παρουσιάστηκε διαρροή του ροζ αντιψυκτικού υγρού από το κάτω μέρος, στο σημείο που φαίνεται και στα βέλη παρακάτω στην Εικόνα 55. Οι αντιστάσεις είχαν τοποθετηθεί πάνω στην χάλκινη επιφάνεια.



Εικόνα 55: Σημείο διαρροής αντλίας υδρόψυξης

Στη συνέχεια η αντλία αποσυναρμολογήθηκε στα εξόν συν ετέθη, οπού και έγινε αντιληπτή η χειριστη ποιότητα κατασκευής με πλαστικό κέλυφος, λασκαρισμένες βίδες και μη εργονομική σχεδίαση του κυκλώματος του θερμαντήρα και της αντλίας. Στην Εικόνα 56 φαίνονται τα εξαρτήματα της αντλίας μετα την αποσυναρμολόγηση.



Εικόνα 56: Αποσυναρμολογημένη αντλία υδρόψυξης

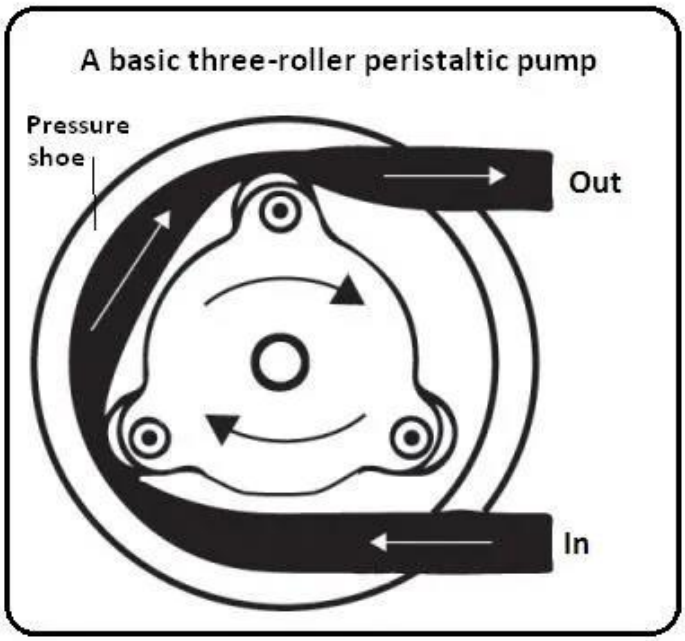
Το μόνο παρήγορο είναι ότι λίγα δευτερόλεπτα πριν εμφανιστεί η διαρροή, υπήρξε απώλεια της πίεσης του υγρού, κάτι το οποίο φάνηκε με το μανόμετρο. Προκύπτει λοιπόν ότι ακόμα και με σε αυτή την απρόβλεπτη βλάβη είχαμε τα καταλληλά σήματα για να μπορέσουμε να προβλέψουμε την διαρροή δευτερόλεπτα πριν αυτή εμφανισθεί. Βάση των παραπάνω η τεχνική της προγνωστικής συντήρησης φαίνεται πως έχει απόλυτη εφαρμογή στην μηχανική. Ενώ η χρήση της τεχνητής νοημοσύνης μπορεί να δώσει γρήγορα τα συμπεράσματα που δεν μπορεί να δώσει ο άνθρωπος ανά πάσα στιγμή.

6.1.4 Επανασχεδίαση και αναβάθμιση μηχανικής κατασκευής.

Στη συνέχεια της βλάβης έπρεπε να αλλαχθεί ο εξοπλισμός της αντλίας της υδρόψυξης ο οποίος αποτελεί την αντλία κυκλοφορίας και τον εναλλάκτη θερμότητας/ θερμαντήρα του συστήματος.

Βασικό και πιο δύσκολο εξάρτημα είναι η αντλία, από την δοκιμή φάνηκε ότι η ποιότητα της αντλίας της υδρόψυξης ήταν ελλαττωματική, θα έπρεπε ή να επιλεχθεί ακριβότερη υδρόψυξη (200-300€) ή να τοποθετηθεί άλλη αντλία. Οι φυγοκεντρικές αντλίες που υπάρχουν στην ελληνική αγορά είναι ερασιτεχνικές και συχνά χρησιμοποιούνται για μικρά σιντριβάνια νερού ή απλούς πειραματικούς αυτοματισμούς με κρύο νερό. Καμία διαθέσιμη αντλία μικρής παροχής δεν θα μπορούσε να καλύψει τις απαιτήσεις του μοντέλου δοκιμών. Με βάση την επαγγελματική εμπειρία επιλέχθηκε λοιπόν περισταλτική αντλία του Γερμανικού οίκου Thomas, και συγκεκριμένα το μοντέλο “SR10/30 DC Standard” με κωδικό προϊόντος “20300319”. Η αρχή λειτουργίας των περισταλτικών αντλιών βασίζεται στην παλμική ροή του υγρού μέσα από έναν ειδικό και ανθεκτικό σωλήνα, ο οποίος πιέζεται

από 3 ροδάκια και έτσι ωθείται το υγρό προς την εξαγωγή. Το παρακάτω σχεδιάγραμμα της Εικόνα 57 περιγράφει τα παραπάνω.

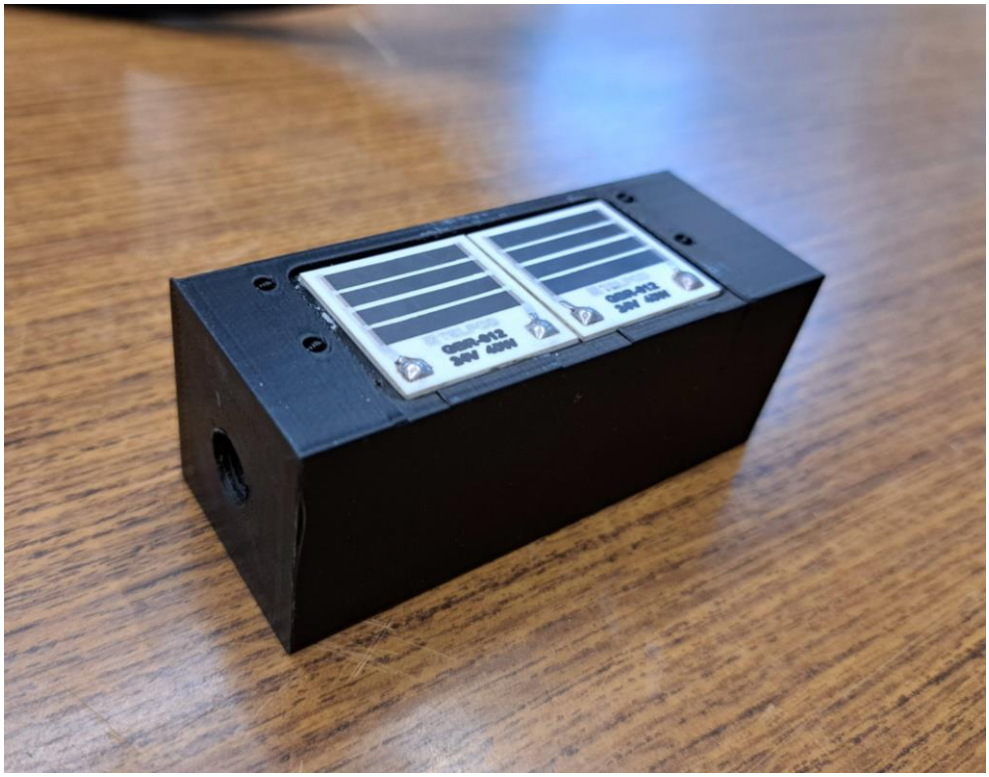


Εικόνα 57: Αρχή λειτουργίας περισταλτικής αντλίας. [58]

Στις συγκεκριμένες αντλίες δεν υπάρχει φτερωτή αλλά ούτε και πολλά εξαρτήματα προς στεγανοποίηση. Η ιδιαιτερότητα της μη επαφής του νερού με κινητά μέρη, κάνουν τις περισταλτικές αντλίες να είναι ιδανικές για μεταφορά χημικών, ενώ σύμφωνα με το datasheet της η συγκεκριμένη αντλία μπορεί να μεταφέρει με άνεση υγρό θερμοκρασίας μέχρι τους 65C. Μόνο μειονέκτημα της συγκεκριμένης αντλίας είναι ο περιορισμένος κύκλος λειτουργίας κατά 50%, πιο συγκεκριμένα η αντλία επιτρέπεται να λειτουργεί έως 15 λεπτά ανά φορά και υποχρεωτικά πρέπει να μην λειτουργεί για διάστημα τουλάχιστον ίσο με τον χρόνο λειτουργίας της. Επιλέχθηκε η αντλία να λειτουργεί ενάμιση λεπτό και να μένει κλειστή για άλλο ενάμιση λεπτό. Η παροχή της αντλίας κατά την λειτουργία της ισούται με 50ml ανά λεπτό και η μέγιστη επιτρεπτή πίεση που μπορεί να εφαρμόσει είναι 8μέτρα νερού, δηλαδή 800mbar. Στην Εικόνα 58 φαίνεται η αντλία που επιλέχθηκε.

Εικόνα 58: Περισταλτική αντλία Thomas SR10/30 DC Standard [59]

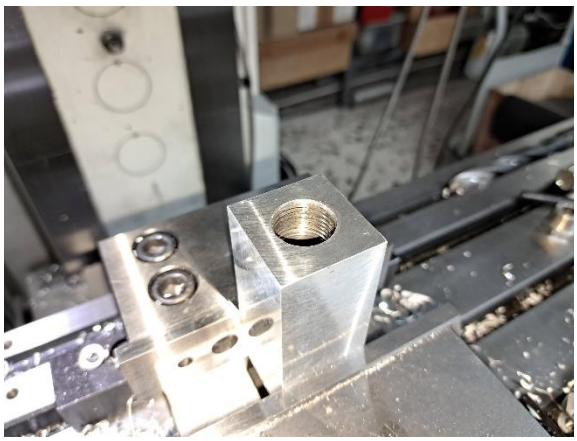
Το επόμενο εξάρτημα προς αντικατάσταση είναι ο εναλλάκτης – θερμαντήρας του νερού κυκλοφορίας. Δεδομένου ότι μέχρι πριν ο εναλλάκτης θερμότητας ήταν ενσωματωμένος με την αντλία, έπρεπε να αλλάξει η σχεδίαση του κυκλώματος. Επιλέχθηκε να κατασκευαστεί σε φρέζα 5 αξόνων εναλλάκτης θερμότητας από αεροπορικό αλουμίνιο 7075 διαστάσεων 34χ34 με 75cm μήκος. Στον εναλλάκτη τοποθετήθηκαν οι τέσσερις αντιστάσεις θέρμανσης με χρήση θερμοαγώγιμης κόλλας και σταθεροποιήθηκαν με ειδικό κάλυμμα από φύλλο PTFE και ένα λεπτό κομμάτι αλουμινίου ως καπάκι. Με γνώμονα ότι οι μεταλλικές κατασκευές είναι ακριβές και χρονοβόρες αν δεν γίνουν από CNC, συνίσταται ως πρώτο βήμα να κατασκευάζεται ένα πρωτότυπο από πλαστικό με 3D εκτυπωτή.



Εικόνα 59: 3D εκτύπωση πλαστικού πρωτότυπο εναλλάκτη θερμότητας με δυο αντιστάσεις

Στη συνέχεια δίνεται η ευκαιρία να γίνουν οι κατάλληλοι έλεγχοι και να προκύψουν διορθώσεις στο σχέδιο που θα κατασκευαστεί το τελικό εξάρτημα από αλουμίνιο. Παρακάτω ακολουθούν εικόνες από τις μηχανουργικές κατεργασίας στη φρέζα.

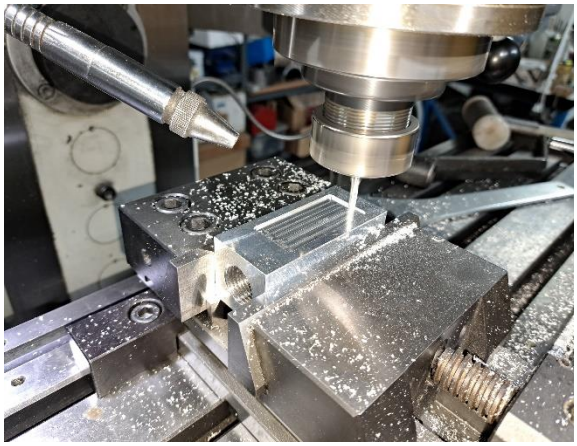
Εικόνα 60: Κατασκευή εναλλάκτη



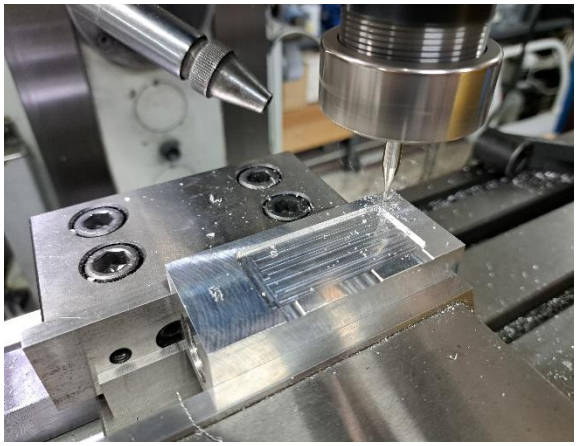
Εικόνα 61: Άνοιγμα σπειρώματος ½ ίντσας

Η χρήση στιβαρών υλικών και η δυνατότητα κατασκευής από την αρχή, μας επιτρέπει την σχεδίαση του εναλλάκτη σύμφωνα με συγκεκριμένες απαιτήσεις. Όπως φαίνεται και

παρακάτω στις Εικόνα 62 καιΕικόνα 63, με τη χρήση ειδικού εργαλείου «κονδύλι» ανοίχθηκε λούκι, θέση για την τοποθέτηση των αντιστάσεων και στη συνέχεια με τέσσερα σπειρώματα τοποθετείτε καπάκι προστασίας και συγκρατήσεις που φαίνεται στη συνέχεια.

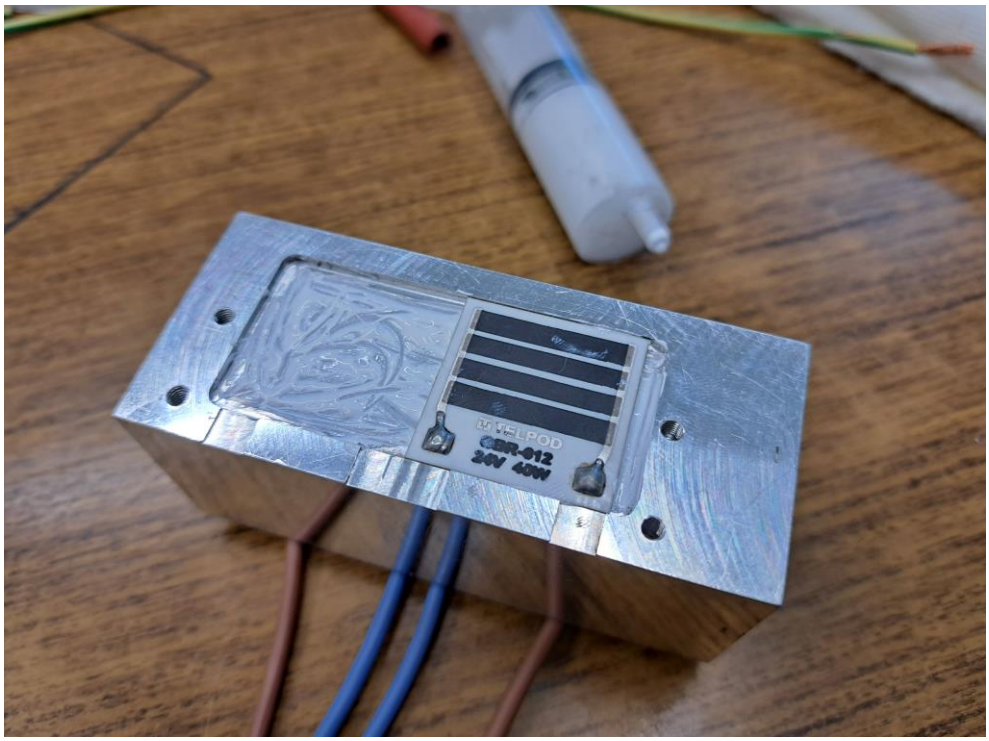


Εικόνα 62: Κατασκευή λουκιού αντιστάσεων



Εικόνα 63: Άνοιγμα σπειρωμάτων για τοποθέτηση καπακιού

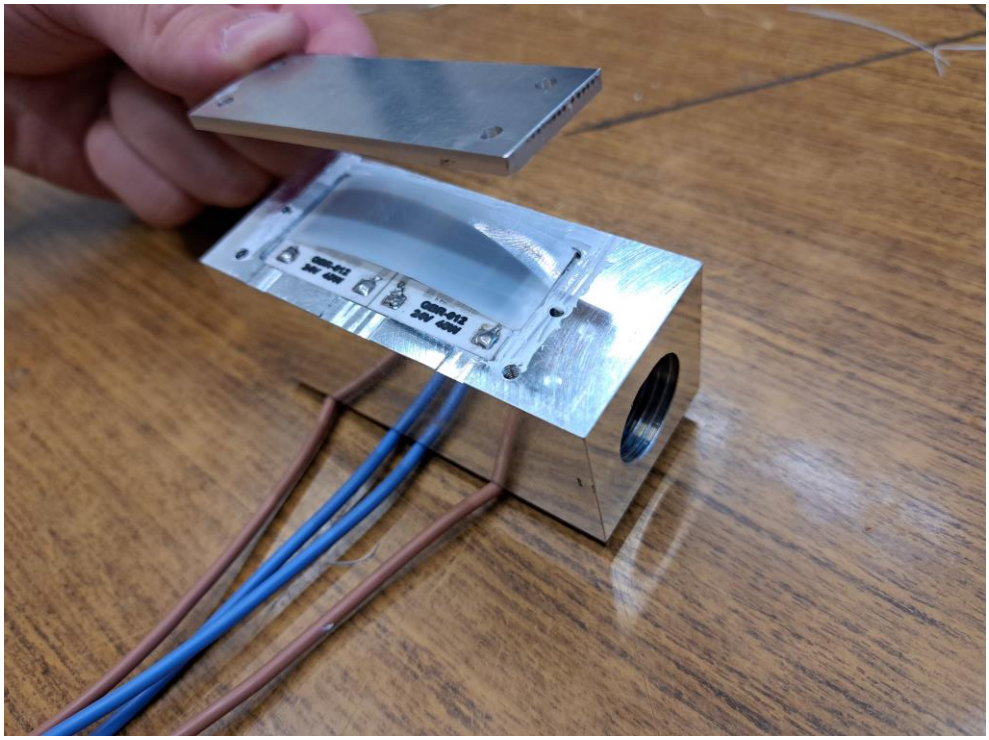
Στη συνέχεια της κατασκευής του εναλλάκτη αρχίζει η συναρμολόγηση του θερμαντήρα όπως φαίνεται στην Εικόνα 64 οι κεραμικές αντιστάσεις τοποθετούνται εντός του ορίου τους με την χρήση θερμοαγώγιμης πάστας και μπαίνουν με τρόπο ώστε οι ηλεκτρικές επαφές τροφοδοσίας να είναι προς μια πλευρά. Οι αντιστάσεις τοποθετούνται αντικρυστά με σκοπό την μέγιστη απορρόφηση της θερμότητας από το υγρό και την εκμετάλλευση της θερμοχωρητικότητας του θερμαντήρα ή αλλιώς καυστήρα.



Εικόνα 64: Τοποθέτηση κεραμικών αντιστάσεων 40W στον εναλλάκτη

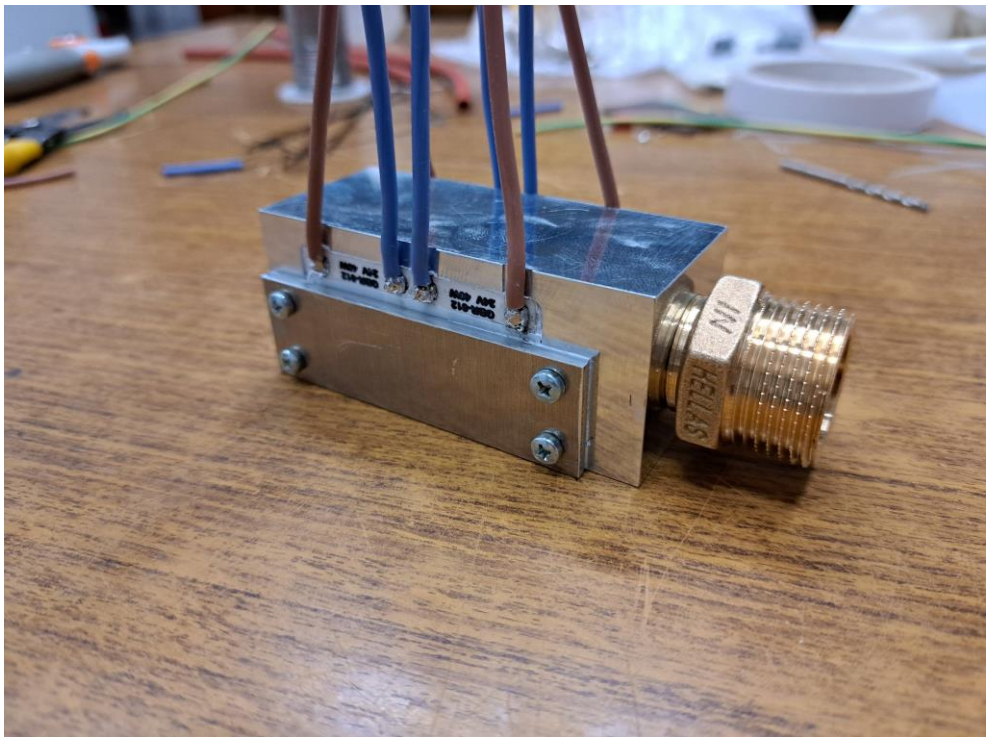
Οι συγκεκριμένες κεραμικές αντιστάσεις έχουν εκτεθειμένες από την άνω πλευρά τους τις ηλεκτρικές επαφές τους, συνεπώς το σύνολο του καλύμματος που τις συγκρατεί θα πρέπει να είναι μονωτής. Λόγω των παραπάνω χρησιμοποιήθηκε ένα φύλλο PTFE “Teflon” πάχους 0,2 χιλιοστά που λειτουργεί ως μονωτής και αντέχει σε θερμοκρασίες τουλάχιστον 150 ˚C.

Τέλος ένα λεπτό κομμάτι αλουμινίου πάχους 2 χιλιοστών τοποθετείται από πάνω και ακουμπάει πάνω στο φύλο PTFE και στον εναλλάκτη, αποτρέποντας την κίνηση των στοιχείων θέρμανσης. Τα παραπάνω φαίνονται αναλυτικά στην



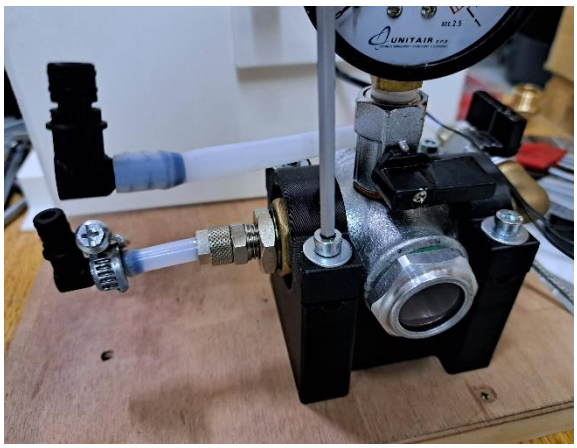
Εικόνα 65: Τελικό στάδιο συναρμολόγησης θερμαντήρα

Η επιλογή των κατάλληλων υλικών ήταν προτεραιότητα για την επιλογή της νέας αντλίας και την κατασκευή του εναλλάκτη. Μεταξύ άλλων τα καλώδια που χρησιμοποιήθηκαν είναι με μόνωση σιλικόνης που αντέχουν με ευκολία θερμοκρασίες 160 ˚C.

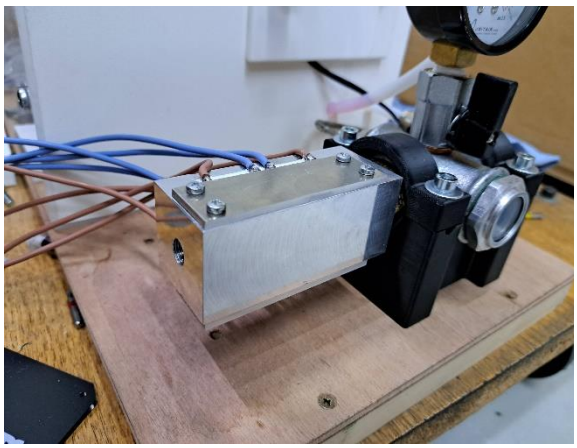


Εικόνα 66: Συναρμολογημένος θερμαντήρας

Παρακάτω στις Εικόνα 67 και Εικόνα 68 απεικονίζεται η διαδικασία αντικατάστασης του εξοπλισμό με την τοποθέτηση του νέου θερμαντήρα.



Εικόνα 67: Αφαίρεση εξαρτημάτων χαλασμένης αντλίας υδρόψυξης

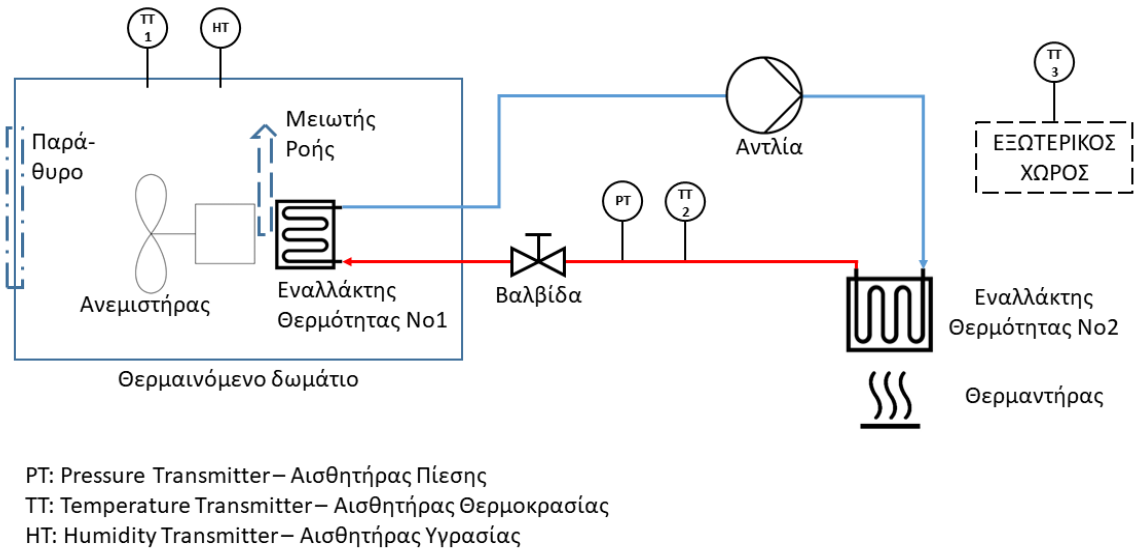


Εικόνα 68: Τοποθέτηση νέου θερμαντήρα

Όπως φαίνεται στην Εικόνα 68, το υγρό κυκλοφορίας εισέρχεται στον θερμαντήρα από μια οπή με σπείρωμα G 1/8" και στη συνέχεια αφού θερμανθεί οδεύει στον χώρο του υδραυλικού σταυρού ως συγκοινωνούν δοχείο μέσω μιας συστολής από ορείχαλκο που φαίνεται στην Εικόνα 66 του συναρμολογημένου θερμαντήρα.

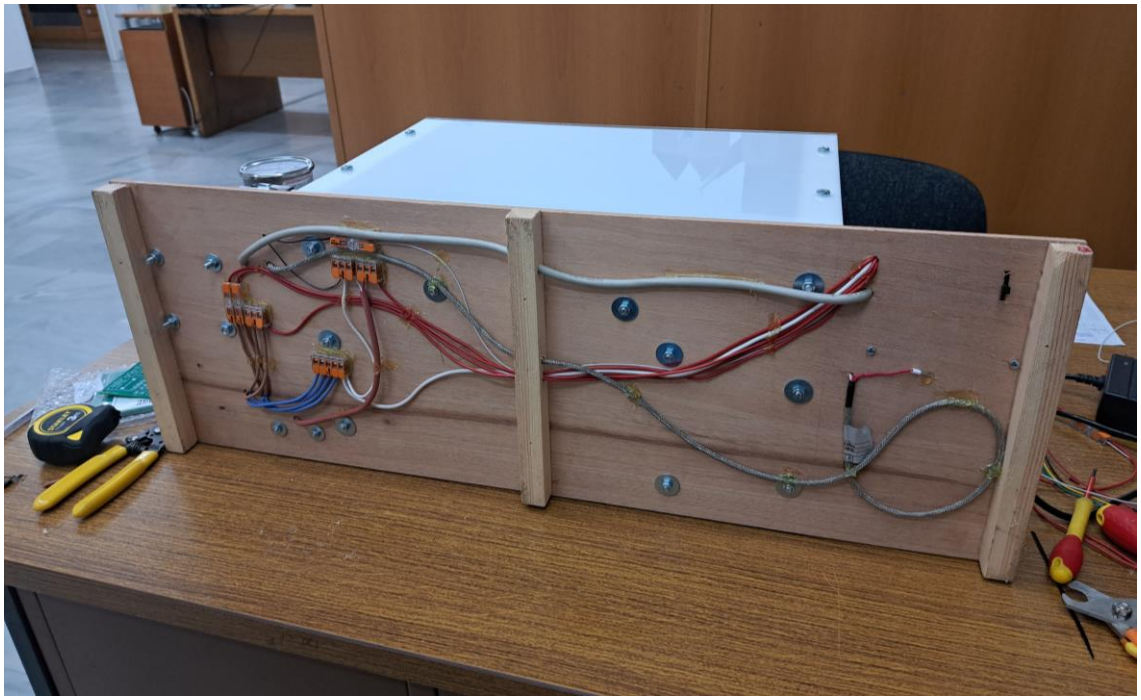
6.2 Τελική μορφή πειραματικής διάταξης

Στον απόηχο της ολοκλήρωσης του μηχανολογικού και του ηλεκτρονικού μέρους έχουμε την τελική μορφή του πειραματικού μοντέλου. Το μοντέλο βασίζεται στο παρακάτω διάγραμμα της Εικόνα 69, βασική διαφορά με την προηγούμενη έκδοση είναι ότι η αντλία είναι τοποθετημένη στο κύκλωμα του κρύου υγρού και δεν καταπονείται θερμικά αφού σε αυτή φτάνει το υγρό που έχει αποθέσει την θερμότητα του στο θερμαινόμενο δωμάτιο. Η συγκεκριμένη διαφοροποίηση έναντι της αρχικής σχεδίασης είναι πλεονέκτημα της μη χρήσης της αντλίας της υδρόψυξης· ενός συνόλου αντλίας και εναλλάκτη, η χρήση δυο εξαρτημάτων μας δίνει ευελιξία στη σχεδίαση, μέγιστη απόδοση και βέλτιστη λειτουργία.



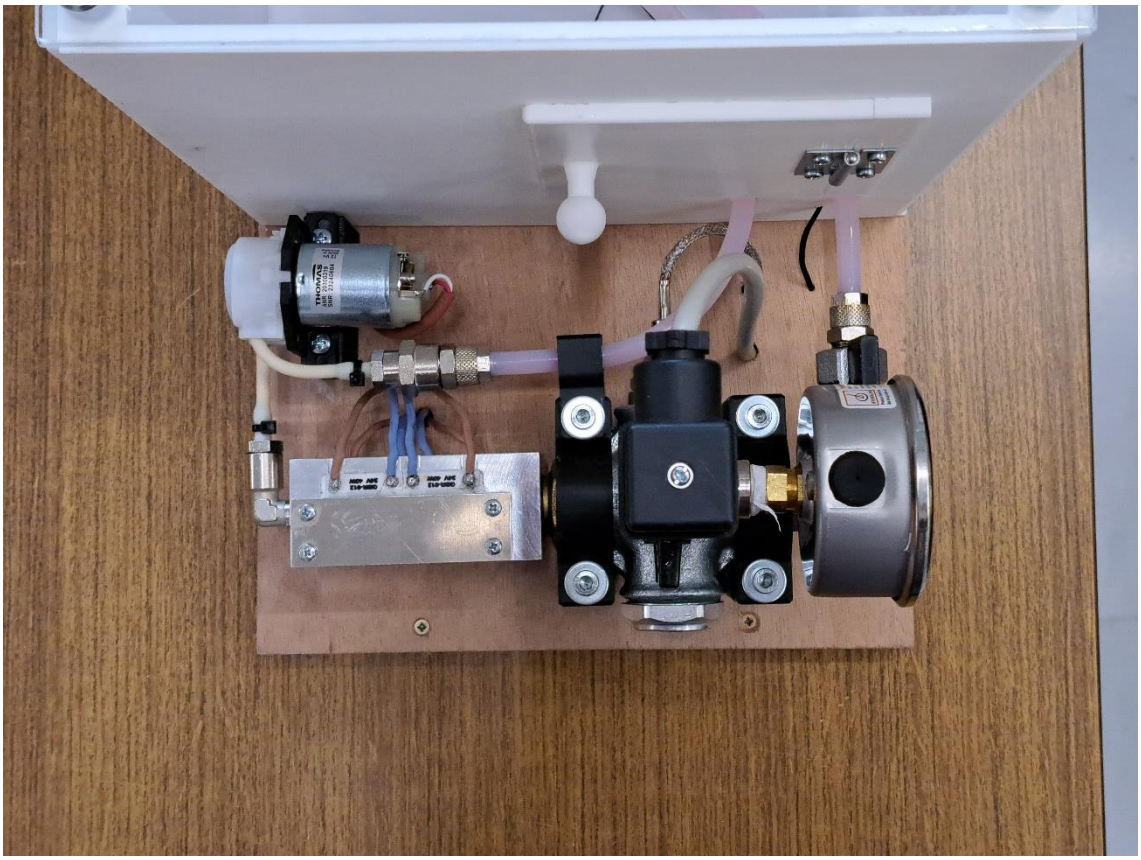
Εικόνα 69: Αναθεωρημένο διάγραμμα προτύπου μοντέλου δοκιμών

Η μορφή του πειραματικού μοντέλου δοκιμών προβάλλεται ανά τμήματα στις παρακάτω φωτογραφίες. Στην Εικόνα 70 βλέπουμε το ηλεκτρικό κύκλωμα, οπού οι αντιστάσεις, η αντλία, ο ανεμιστήρα και τα αισθητήρια συνδέονται με καλώδια που έρχονται από την πλακέτα.

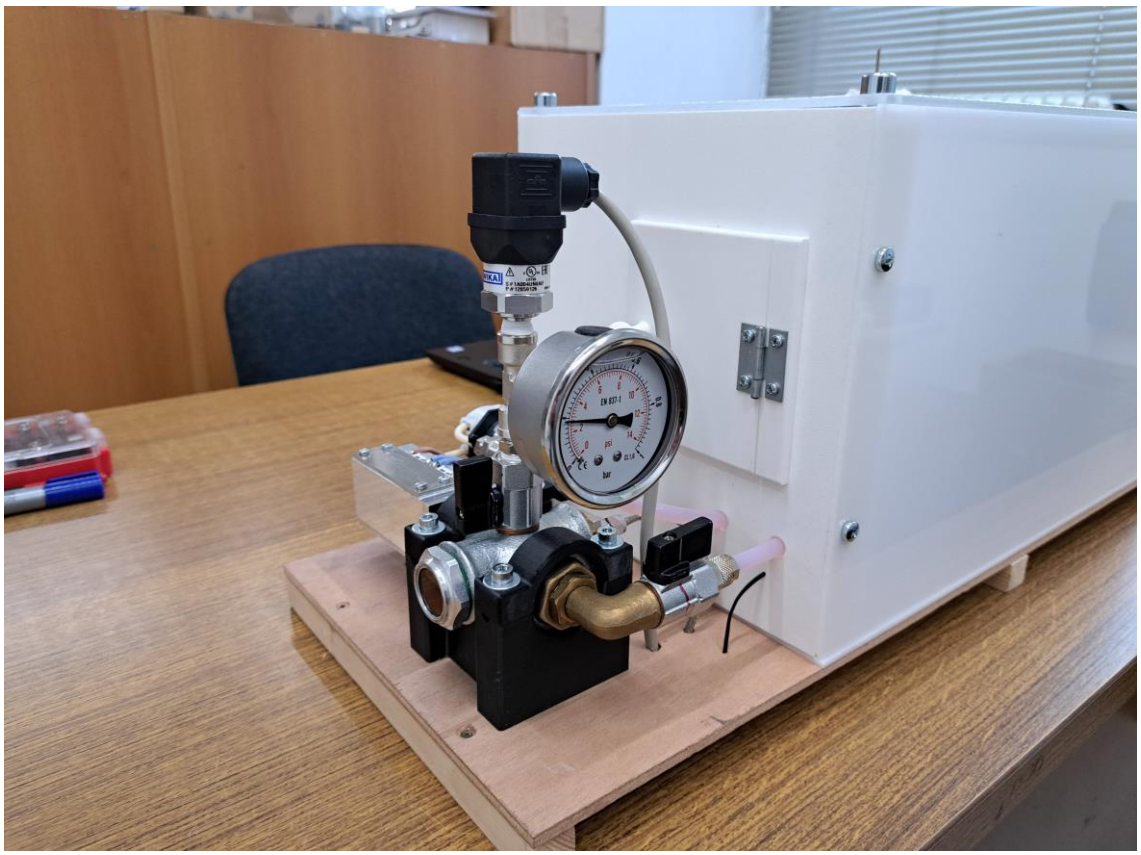


Εικόνα 70: Κάτω άποψη μοντέλου και ηλεκτρικού κυκλώματος

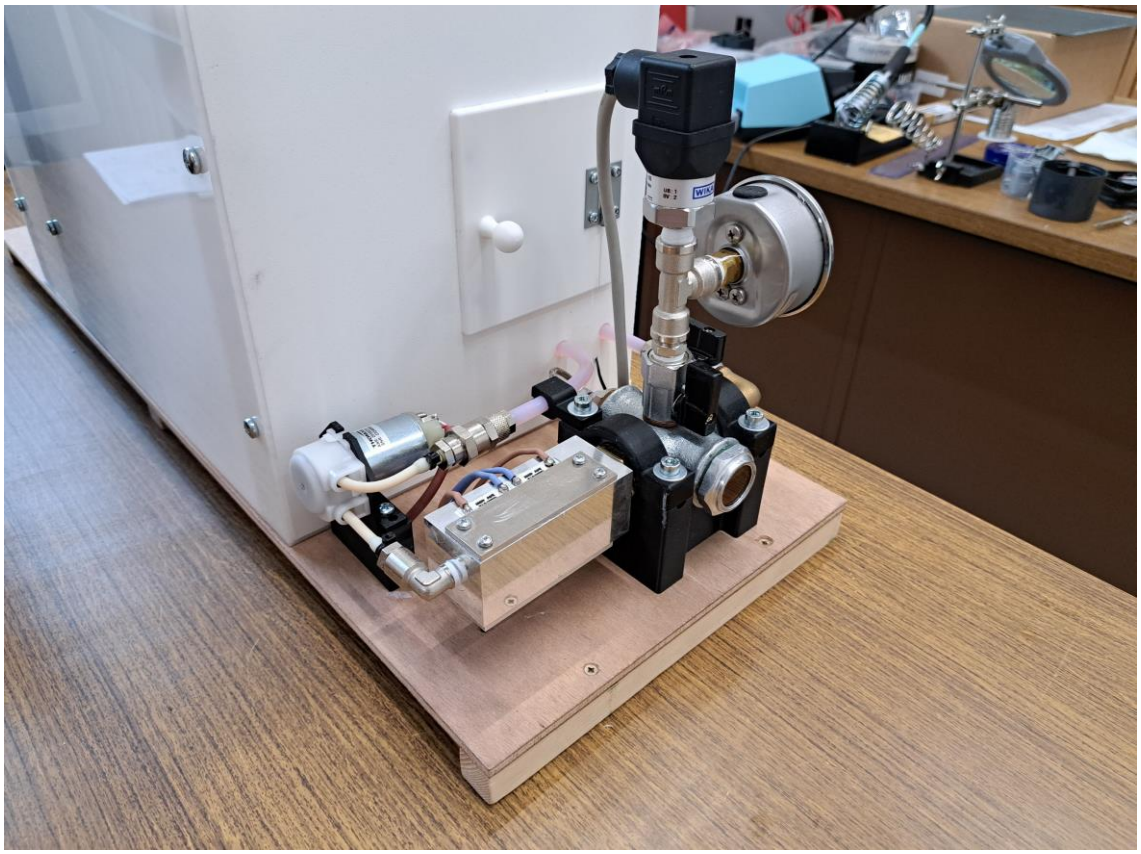
Στις Εικόνα 71,Εικόνα 72,Εικόνα 73 βλέπουμε από την πάνω πλευρά ολόκληρο το κύκλωμα θέρμανσης και κυκλοφορίας του υγρού, όπως κατασκευάστηκε.



Εικόνα 71: Άνω όψη υδραυλικού κυκλώματος θέρμανσης υγρού



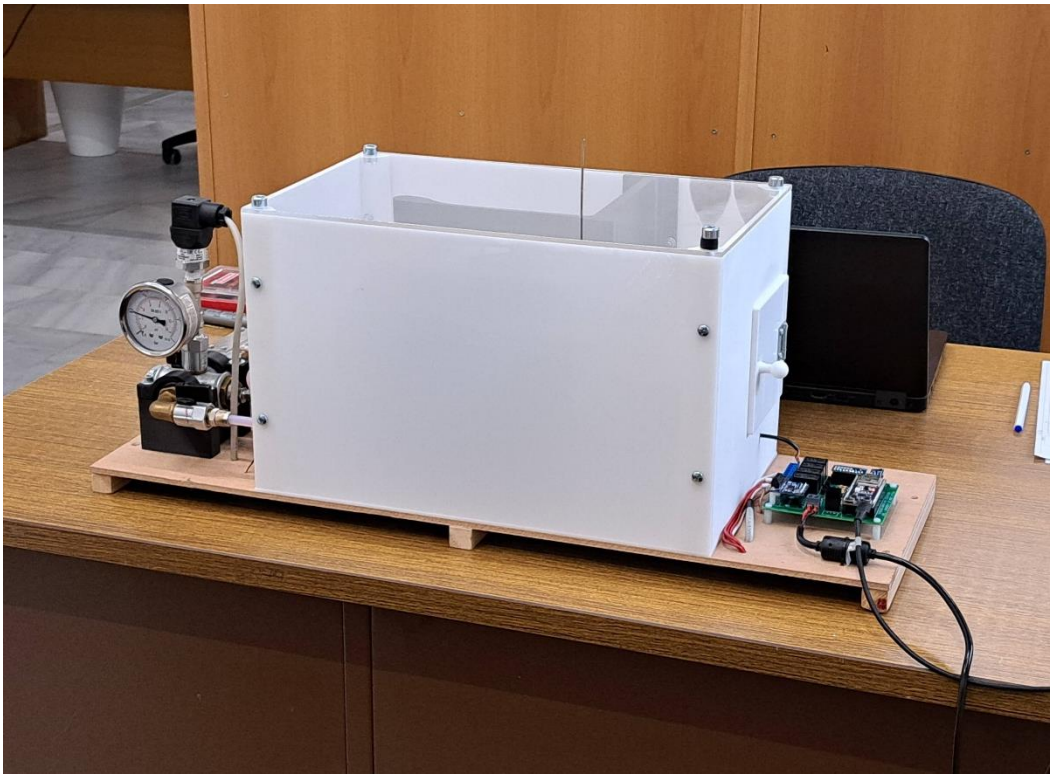
Εικόνα 72: Μπροστινή όψη υδραυλικού κυκλώματος θέρμανσης υγρού



Εικόνα 73: Όπισθεν όψη υδραυλικού κυκλώματος θέρμανσης υγρού

Μια συνολική άποψη του μοντέλου έχουμε πλέον στην Εικόνα 74, όπου ολόκληρο το μοντέλο είναι σε μια ξύλινη βάση όπως αναφέρθηκε εξ αρχής και δυο τροφοδοτικά δίνουν

ρεύμα στο σύστημα. Ένα τροφοδοτικό στα 24V των 200W παρέχει ρεύμα σε αντιστάσεις, αντλία, ανεμιστήρα και αισθητήρα πίεσης, ενώ όλα τα στοιχεία λογικής της πλακέτας, όπως ο μικροελεγκτής λειτουργούν με να USB τροφοδοτικό των 5V.



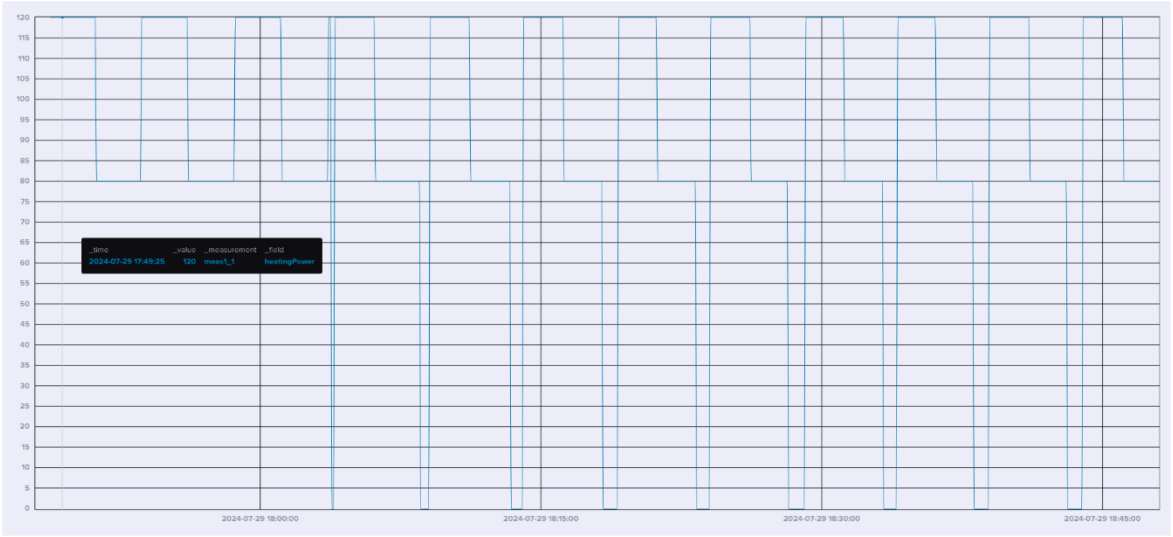
Εικόνα 74: Πειραματικό μοντέλο δοκιμών

6.3 Συλλογή δεδομένων για την εκπαίδευση του αλγορίθμου

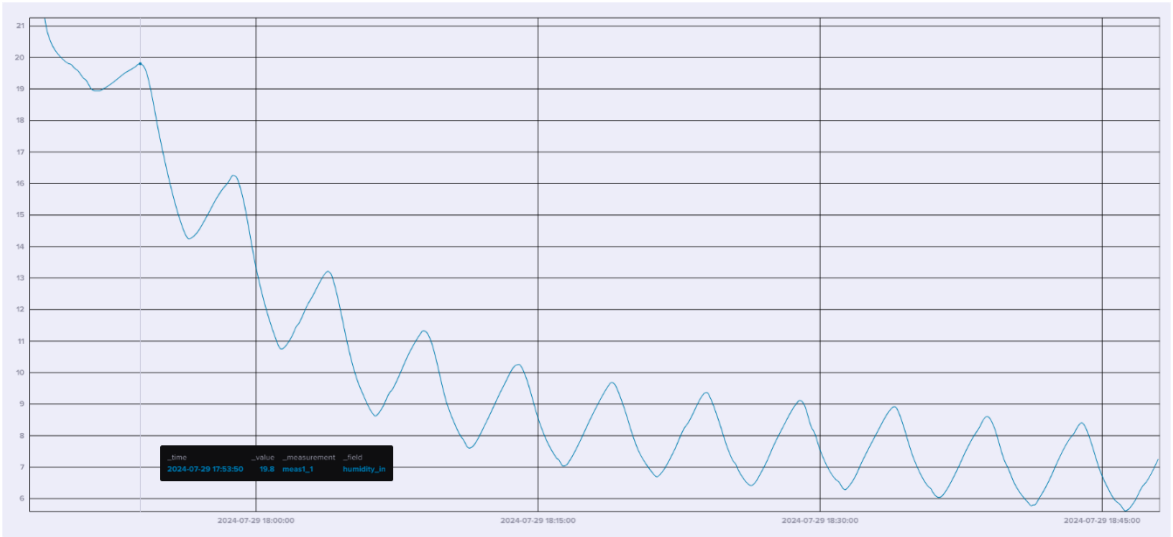
Μέσω της πλατφόρμας διεπαφής της βάσης δεδομένων της InfluxDB έχουμε την δυνατότητα να δούμε, όπως στην Εικόνα 75, τα δεδομένα που καταγράφονται κάθε 5 δευτερόλεπτα από το μικροελεγκτή μας και αποθηκεύονται στο υπολογιστικό νέφος.

Εικόνα 75: Μετρήσεις κατά την εκπαίδευση και την δοκιμή

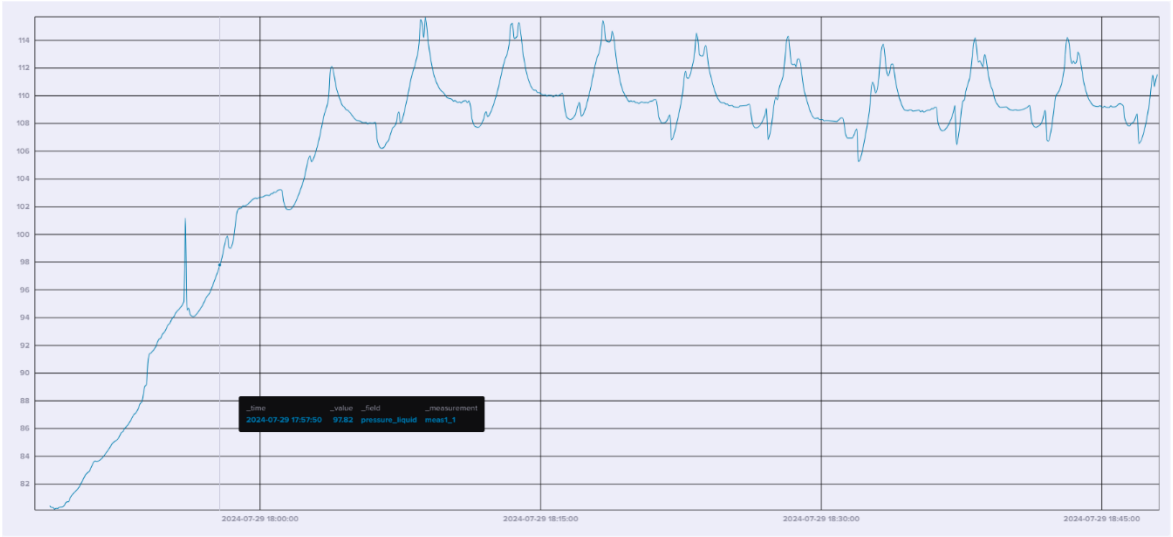
Στις επόμενες εικόνες βλέπουμε τα μεγέθη σε ένα συγκεκριμένο διάστημα μίας ώρας, κατά την καταγραφή δεδομένων για σκοπούς εκπαίδευσης του αλγορίθμου, το ακόλουθο δείγμα δεδομένων απεικονίζει υγιή λειτουργία του εξοπλισμού.



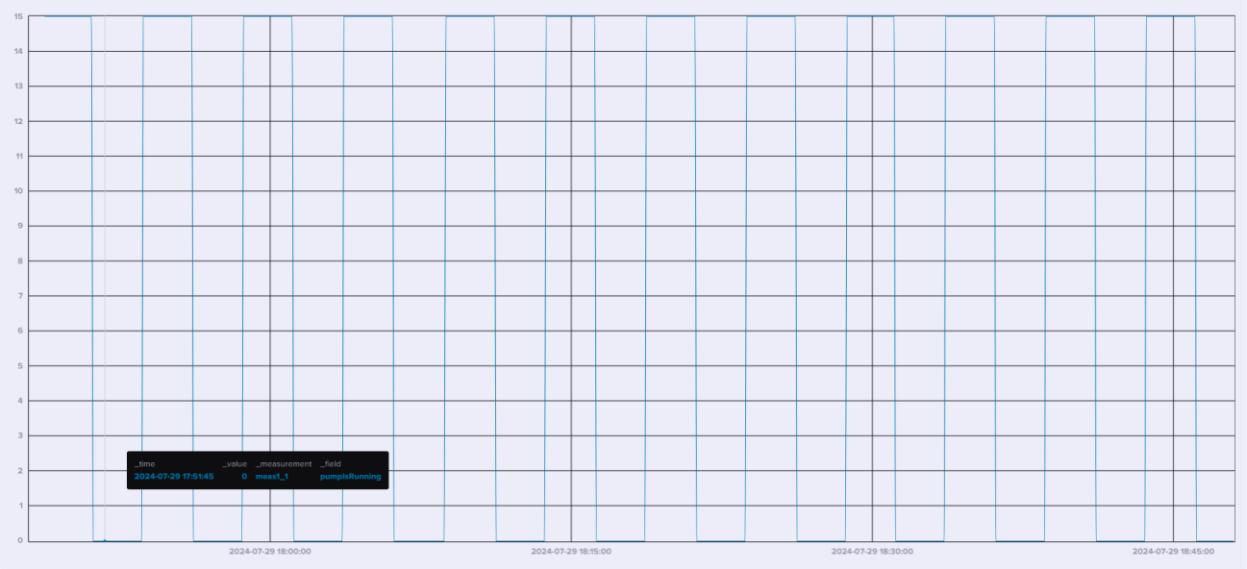
Εικόνα 76: Καμπύλη ισχύς θέρμανσης



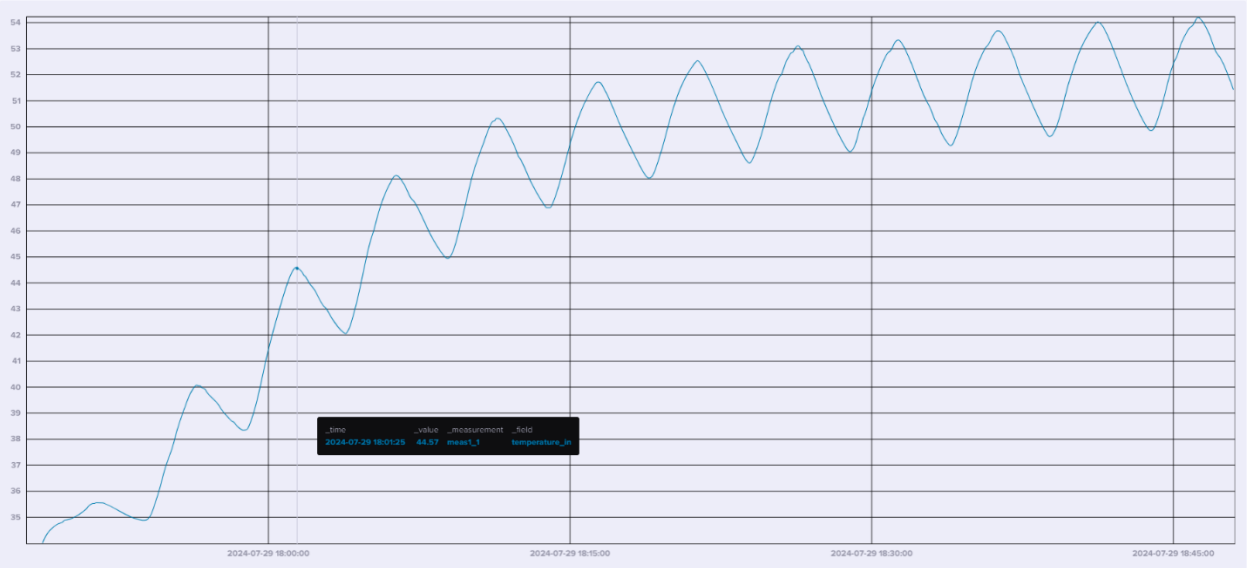
Εικόνα 77: Καμπύλη υγρασίας του θερμαινόμενου χώρου



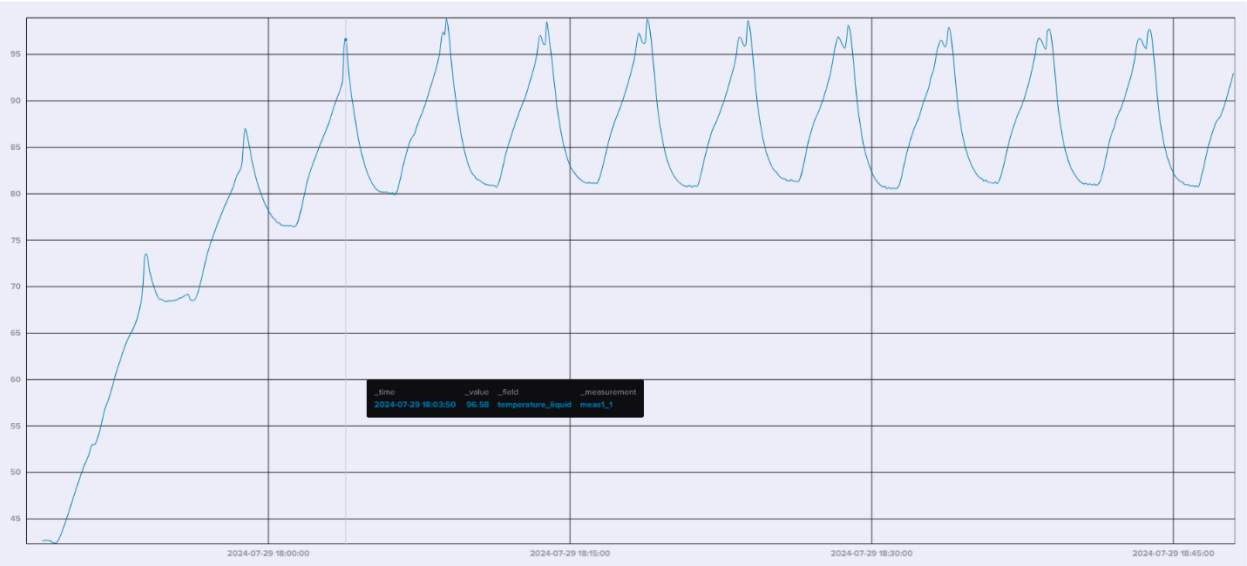
Εικόνα 78: Καμπύλη πίεσης του υγρού θέρμανσης/κυκλοφορίας



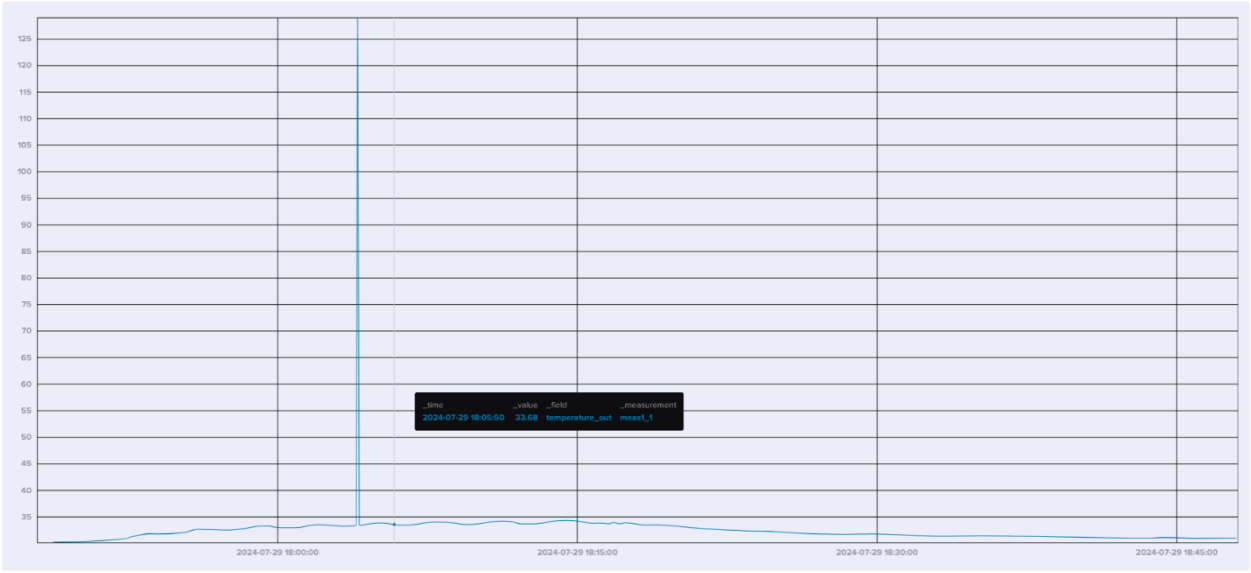
Εικόνα 79: Καμπύλη κατάστασης-λειτουργίας της αντλίας κυκλοφορίας



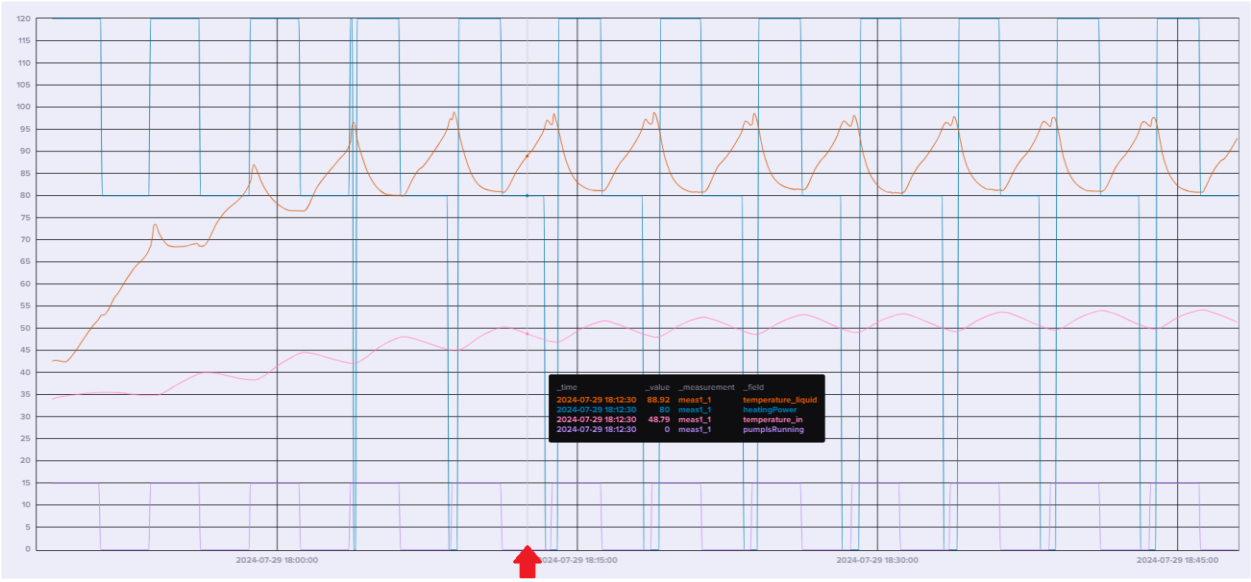
Εικόνα 80: Καμπύλη θερμοκρασίας του θερμαινόμενου χώρου



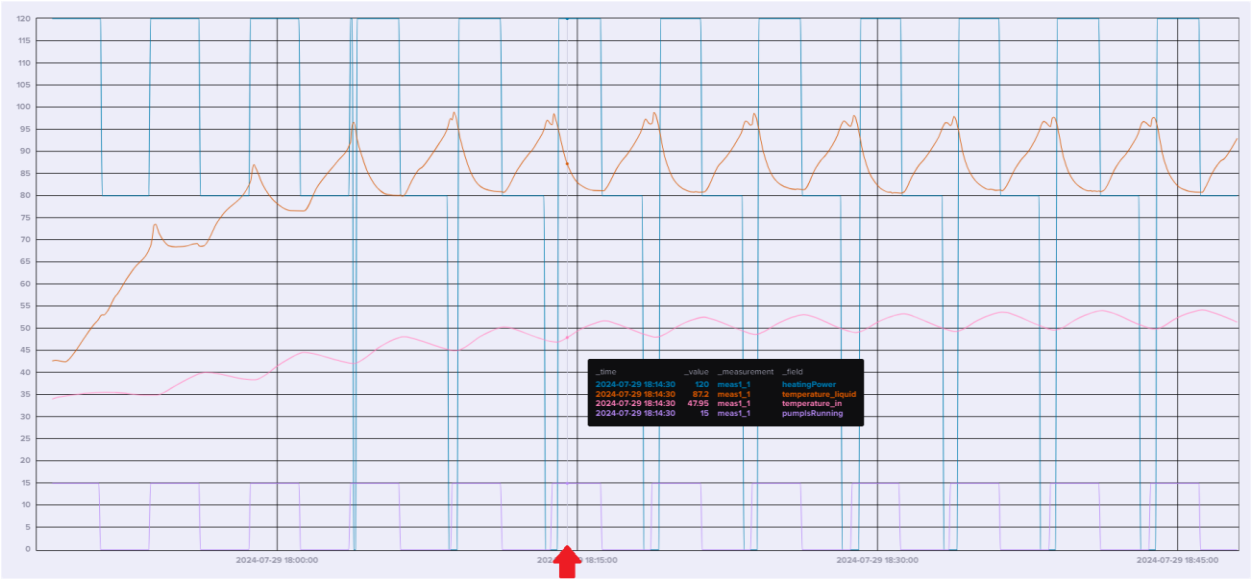
Εικόνα 81: Καμπύλη θερμοκρασίας υγρού θέρμανσης/κυκλοφορίας



Εικόνα 82: Καμπύλη θερμοκρασίας του εξωτερικού περιβάλλοντος

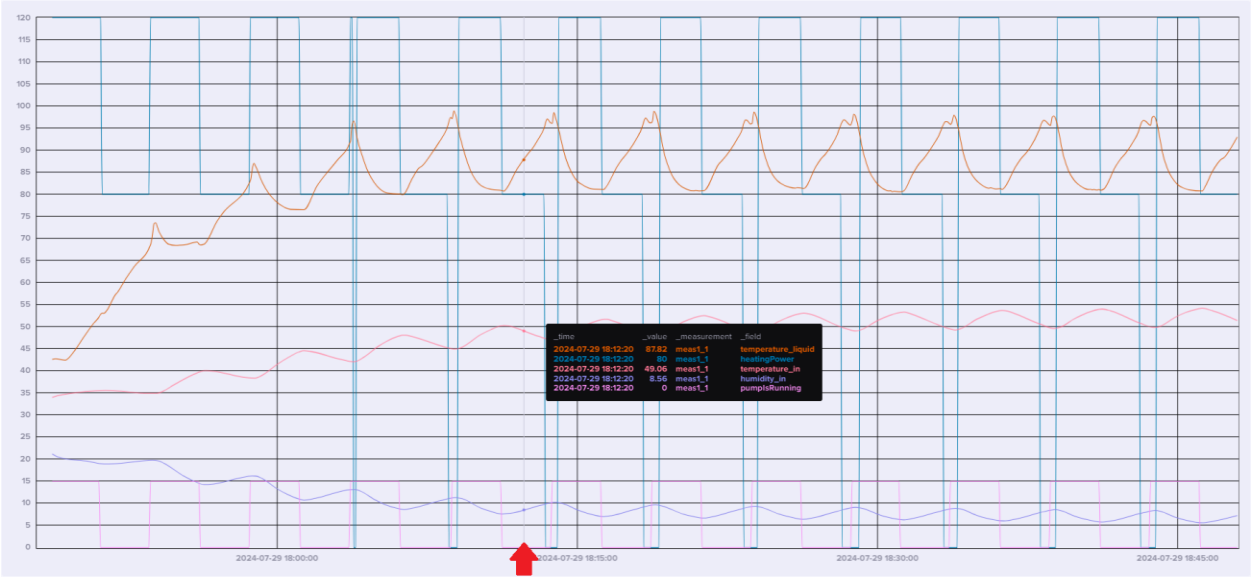


Εικόνα 83: Θερμοκρασία υγρού, Θερμική ισχύς και Θερμοκρασία δωματίου με κλειστή αντλία

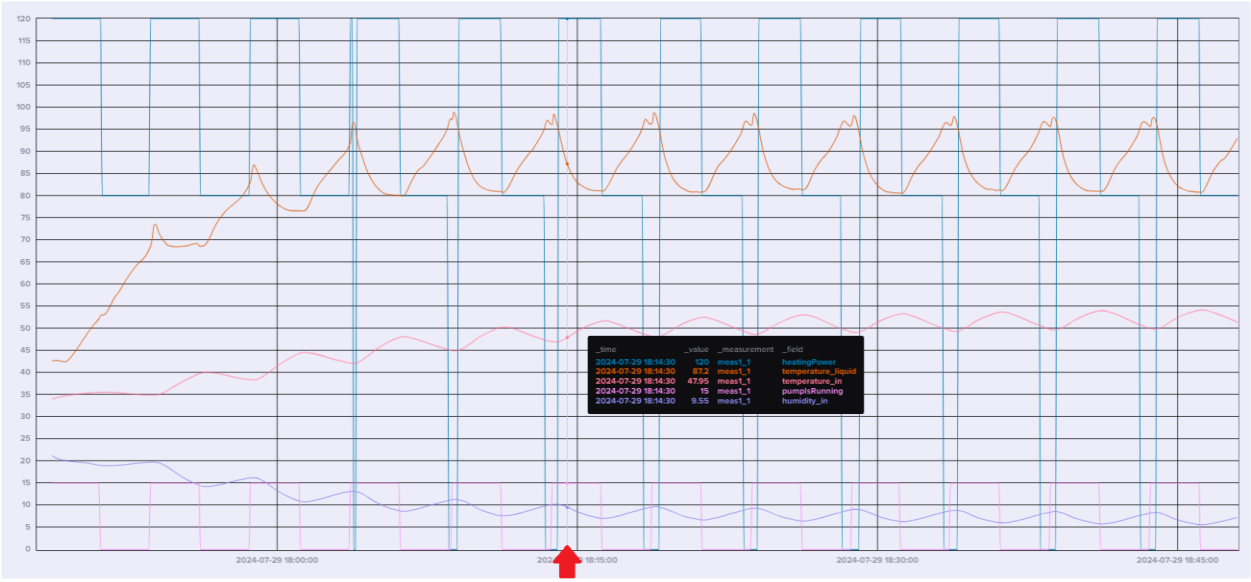


Εικόνα 84: Θερμοκρασία υγρού, Θερμική ισχύς και Θερμοκρασία δωματίου με ανοιχτή αντλία

Παραπάνω στις Εικόνα 83 και Εικόνα 84, βλέπουμε την σχέση των μεγεθών της θερμοκρασίας του υγρού κυκλοφορίας, της θερμικής ισχύς και της θερμοκρασίας του δωματίου σε συνάρτηση με τον χρόνο. Ενώ οι δύο εικόνες μεταξύ τους αποτυπώνουν την κατάσταση με την αντλία σε ανάπαυση (Εικόνα 83) και σε λειτουργία (Εικόνα 84).



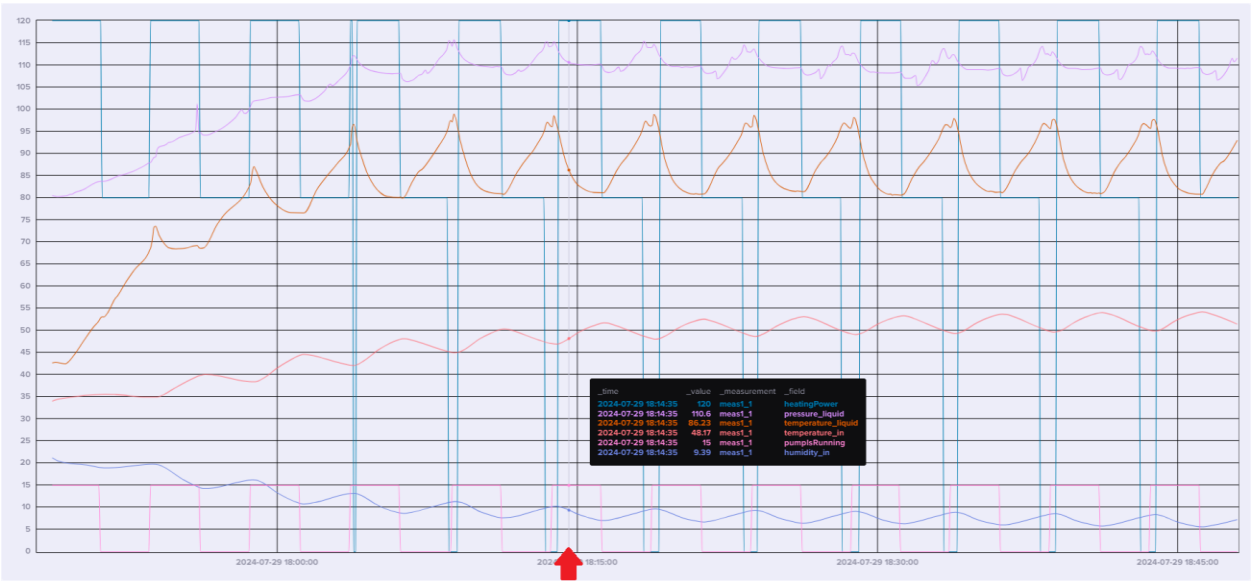
Εικόνα 85: Θερμοκρασία υγρού, Θερμική ισχύς, Θερμοκρασία και Υγρασία δωματίου με κλειστή αντλία



Εικόνα 86: Θερμοκρασία υγρού, Θερμική ισχύς, Θερμοκρασία και Υγρασία δωματίου με ανοιχτή αντλία

Στις Εικόνα 85 και Εικόνα 86, βλέπουμε την σχέση των προαναφερθέντων μεγεθών σε συνάρτηση με τον χρόνο, ενώ έχει προστεθεί και η υγρασία του θερμαινόμενου δωματίου. Οι δύο εικόνες μεταξύ τους αποτυπώνουν την κατάσταση με την αντλία σε ανάπαυση (Εικόνα 85) και σε λειτουργία (Εικόνα 86).

Εικόνα 87: Πίεση υγρού, Θερμοκρασία υγρού, Θερμική ισχύς, Θερμοκρασία και Υγρασία δωματίου με κλειστή αντλία



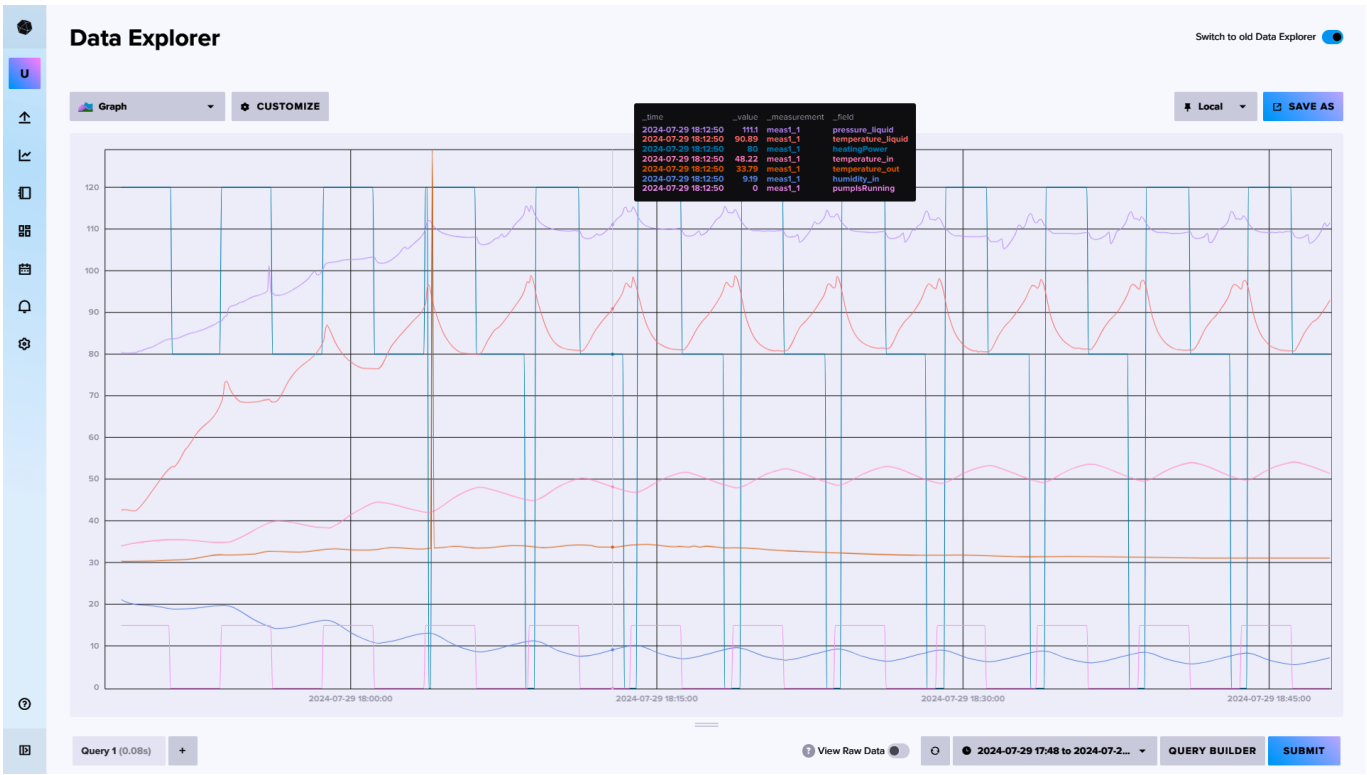
Εικόνα 88: Πίεση υγρού, Θερμοκρασία υγρού, Θερμική ισχύς, Θερμοκρασία και Υγρασία δωματίου με ανοιχτή αντλία

Στις Εικόνα 87 και Εικόνα 88, βλέπουμε την σχέση όλων των προαναφερθέντων μεγεθών σε συνάρτηση με τον χρόνο, ενώ έχει προστεθεί και η πίεση του υγρού κυκλοφορίας. Η πίεση απεικονίζεται σε centibars (100 centibars = 1 bar). Οι δύο εικόνες μεταξύ τους αποτυπώνουν την κατάσταση με την αντλία σε ανάπαυση (Εικόνα 87) και σε λειτουργία (Εικόνα 88).

Παρατηρούμε λοιπόν ότι προκύπτει συσχέτιση των μεγεθών. Η θερμοκρασία του υγρού και του κλειστού δωματίου έχουν αντίστροφο κυματισμό ως προς τον χρόνο, τα μεγέθη επηρεάζονται ανάλογα την θερμική ισχύ και την λειτουργία της αντλίας. Η καμπύλη της θερμοκρασίας του υγρού παρουσιάζει μορφή αιχμηρού βουνού όταν η θέρμανση είναι συνεχής, καθώς το υγρό φτάνει στην μέγιστη θερμοκρασία του, η θέρμανση διακόπτατε και η καμπύλη έχει μορφή κοντύτερου βουνού με περισσότερες από μία κορυφές. Η άνοδος της θερμοκρασίας του δωματίου ακολουθείται από ταυτόχρονη πτώση της θερμοκρασίας του υγρού, είναι φανερό ότι γίνεται μεταφορά της θερμότητας από το ένα μέσο στο άλλο· από το υγρό στον αέρα. Αν αλλάξει η απόδοση μεταφοράς της θερμότητας αυτό θα γίνει

αντιληπτό από την σχέση των καμπύλων των δύο θερμοκρασιών, μεταξύ τους. Η υγρασία του χώρου μειώνεται σε βάθος χρόνου και ακολουθεί τον κυματισμό της θερμοκρασίας του δωματίου. Η θερμοκρασία του εξωτερικού περιβάλλοντος παρατηρούμε ότι μένει σχετικά σταθερή και δεν επηρεάζει αρνητικά την συμπεριφορά του συστήματος. Η πίεση του υγρού είναι ένα μέγεθος με έντονη εξάρτηση στην θερμοκρασία του υγρού, την κατάσταση λειτουργίας της αντλίας και την ένταση της θέρμανσης. Όσο η αντλία δεν δουλεύει και η θέρμανση είναι ίση με 80 Watt η πίεση μειώνεται ελαφρός, ενώ στη συνέχεια ανεβαίνει απότομα καθώς συσσωρεύεται θερμότητα στο υγρό. Όταν ξεκινάει η αντλία και αρχίζει η κυκλοφορία του υγρού, η πίεση αυξάνεται στιγμιαία αλλά στη συνέχεια μειώνεται καθώς μειώνεται και η θερμοκρασία του υγρού· ως αποτέλεσμα της μεταφοράς της θερμότητας από το υγρό στο δωμάτιο. Επιπλέον όσο και αν πέσει η θερμοκρασία του υγρού από τους 100 έως προς τους 80C, η καμπύλη της πίεσης του υγρού ακολουθεί μορφή σχεδόν ευθεία γραμμή με πολύ μικρή πτώση.

Κατά την φάση εκπαίδευσης το σχήμα της εκάστοτε καμπύλης στο βάθος του χρόνου σε συνδυασμό με τις τιμές των υπόλοιπων μεγεθών ορίζουν την υγιή κατάσταση. Στη φάση του ελέγχου λειτουργίας, η αναγνώριση του αν το σχήμα της καμπύλης αυτής έχει εμφανιστεί ξανά στο παρελθών χαρακτηρίζουν αν η λειτουργία αυτή είναι ορθή ή όχι. Η πλήρης απεικόνιση των μετρήσιμων μεγεθών παρουσιάζεται στην Εικόνα 89.



Εικόνα 89: Πλήρης απεικόνιση μετρήσιμων μεγεθών

6.4 Εκπαίδευση αλγορίθμου τεχνητής νοημοσύνης

Η εκπαίδευση του αλγορίθμου περιλαμβάνει δυο στάδια. Το πρώτο στάδιο αφορά την δημιουργία της αρχιτεκτονικής του αλγορίθμου που θα μπορεί να αναγνωρίζει καλύτερα διαφορές στις τιμές, να αναγνωρίζει τα μετρήσιμα μεγέθη και να εφαρμόζει ελέγχους προγνωστικής συντήρησης. Το δεύτερο στάδιο αφορά την εκπαίδευση του αλγορίθμου με

τα πραγματικά δεδομένα των μετρήσεων με σκοπό να καταλαβαίνει ακριβώς ποιες είναι οι ανωμαλίες σε μια ακολουθία τιμών.

Παρακάτω στην Εικόνα 90, βλέπουμε την εκτέλεση του κώδικα όπου γίνονται τα δυο προαναφερθέντα βήματα. Πιο συγκεκριμένα ο κώδικα του προγράμματος σε γλώσσα python3 “dffrominflux_simple_meas5.py” όπως επισυνάπτεται και στο παράρτημα, διαβάζει όλα τα δεδομένα (features) τα χωρίζει σε ομάδες δεδομένων, τα κατατάσσει με βάση τον τύπο δεδομένων και βάση του ερωτήματος “Type 'load' to load an existing model, or 'train' to train a new one:” ρωτάει αν επιθυμούμε να δημιουργήσουμε μια νέα βελτιωμένη εκπαιδευμένη αρχιτεκτονική αλγορίθμου τεχνητής νοημοσύνης ή αν απλά θέλουμε να φορτώσουμε μια αποθηκευμένη αρχιτεκτονική και να ελέγξουμε την λειτουργία του.

```
root@vps6116:~# cd thesis/
root@vps6116:~/thesis# python3.9 -i dffrominflux_simple_meas5.py
2024-07-30 23:27:07.918564: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU will not be used.
2024-07-30 23:27:08.042080: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to
2024-07-30 23:27:08.042188: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to
2024-07-30 23:27:08.042316: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting
2024-07-30 23:27:08.075480: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU will not be used.
2024-07-30 23:27:11.370334: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-07-22 18:47:05+00:00         120.0        21.16          0.96842            1.0           32.50               45.01            28.77
2024-07-22 18:47:10+00:00         120.0        21.06          0.96850            1.0           32.62               44.67            28.77
2024-07-22 18:47:15+00:00         120.0        20.96          0.96958            1.0           32.70               44.26            28.78
2024-07-22 18:47:20+00:00         120.0        20.84          0.97008            1.0           32.79               43.78            28.76
2024-07-22 18:47:25+00:00         120.0        20.73          0.97058            1.0           32.89               43.34            28.77
timestamp
2024-07-22 18:47:05+00:00    0
2024-07-22 18:47:10+00:00    0
2024-07-22 18:47:15+00:00    0
2024-07-22 18:47:20+00:00    0
2024-07-22 18:47:25+00:00    0
Name: session_id, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8344 entries, 0 to 8343
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   heatingPower       8344 non-null   float64
 1   humidity_in        8344 non-null   float64
 2   pressure_liquid    8344 non-null   float64
 3   pumpIsRunning      8344 non-null   int64
 4   temperature_in     8344 non-null   float64
 5   temperature_liquid 8344 non-null   float64
 6   temperature_out    8344 non-null   float64
 7   session_id         8344 non-null   int64
dtypes: float64(6), int64(2)
memory usage: 521.6 KB
None
Training data shape: (6099, 60, 7), (6099, 4)
Test data shape: (1525, 60, 7), (1525, 4)
Type 'load' to load an existing model, or 'train' to train a new one: train
```

Εικόνα 90: Δημιουργία αρχιτεκτονικής μοντέλου τεχνητής νοημοσύνης

Στην Εικόνα 91, βλέπουμε την διαδικασία της εύρεσης της βέλτιστης αρχιτεκτονικής, το πρόγραμμα τρέχει 50 φορές “epochs=50”, όπου κάθε φορά δοκιμάζει διαφορετική δομή με περισσότερους ή λιγότερους νευρώνες, περισσότερα ή λιγότερα επίπεδα, διαφορετικό ποσοστό ζώνης απόρριψης ανά επίπεδο, διαφορετικό ρυθμό εκμάθησης και φορών εκπαίδευσης σε τελική φάση. Δοκιμάζοντας 50 φορές διαφορετικούς συνδυασμούς καταλήγει στις βέλτιστες υπερπαραμέτρους “hyperparameters”, δομώντας την αρχιτεκτονική με το μικρότερο δυνατό σφάλμα. Είναι όμως ξεκάθαρο ότι στην παρούσα περίπτωση δεν έχει γίνει κάποια συγκεκριμένη μελέτη με εκατομμύρια μετρήσεις, υπάρχει ο κίνδυνος της υπερπροσαρμογής. Σε περίπτωση εκτενούς μελέτης, είναι δυνατόν να προκύψει μια βέλτιστη αρχιτεκτονική για ελέγχους προγνωστικής συντήρησης με μεγαλύτερη αξιοπιστία. Η αυτόματη δημιουργία αρχιτεκτονικής με αυτόματη εκλογή των υπερπαραμέτρων είναι μια ικανοποιητική προσέγγιση για την κατασκευή αλγορίθμου για τους πειραματικούς σκοπούς της διπλωματικής εργασίας.

```
Epoch 8/10
191/191 [==============================] - 298s 2s/step - loss: 9.9081e-04 - val_loss: 6.4678e-04
Epoch 9/10
191/191 [==============================] - 299s 2s/step - loss: 9.4002e-04 - val_loss: 6.7593e-04
Epoch 10/10
191/191 [==============================] - 304s 2s/step - loss: 8.5145e-04 - val_loss: 3.9379e-04

Trial 29 Complete [00h 49m 20s]
val_loss: 0.00036704697413370013

Best val_loss So Far: 0.0002983174053952098
Total elapsed time: 08h 11m 39s

Search: Running Trial #30

Value             |Best Value So Far |Hyperparameter
224               |192               |units_1
0.2               |0                 |dropout_1
3                 |2                 |num_layers
352               |128               |units_2
0.2               |0                 |dropout_2
0.0035158         |0.0012481         |learning_rate
416               |128               |units_3
0.3               |0                 |dropout_3
256               |128               |units_4
0.4               |0.4               |dropout_4
10                |10                |tuner/epochs
0                 |4                 |tuner/initial_epoch
0                 |2                 |tuner/bracket
0                 |2                 |tuner/round

Epoch 1/10
191/191 [==============================] - 355s 2s/step - loss: 0.0712 - val_loss: 0.0032
Epoch 2/10
191/191 [==============================] - 332s 2s/step - loss: 0.0093 - val_loss: 0.0033
Epoch 3/10
191/191 [==============================] - 326s 2s/step - loss: 0.0075 - val_loss: 0.0032
Epoch 4/10
191/191 [==============================] - 345s 2s/step - loss: 0.0066 - val_loss: 0.0025
Epoch 5/10
191/191 [==============================] - 337s 2s/step - loss: 0.0054 - val_loss: 0.0020
Epoch 6/10
191/191 [==============================] - 347s 2s/step - loss: 0.0056 - val_loss: 0.0026
Epoch 7/10
191/191 [==============================] - 344s 2s/step - loss: 0.0046 - val_loss: 0.0030
Epoch 8/10
191/191 [==============================] - 354s 2s/step - loss: 0.0043 - val_loss: 0.0028
Epoch 9/10
191/191 [==============================] - 332s 2s/step - loss: 0.0040 - val_loss: 0.0028
Epoch 10/10
191/191 [==============================] - 346s 2s/step - loss: 0.0038 - val_loss: 0.0023

Trial 30 Complete [00h 57m 03s]
val_loss: 0.0019612438045442104

Best val_loss So Far: 0.0002983174053952098
Total elapsed time: 09h 08m 42s
48/48 [==============================] - 10s 169ms/step - loss: 2.9832e-04
Best model loss on test set: 0.0002983174053952098
```

Εικόνα 91: Εύρεση βέλτιστης αρχιτεκτονικής

7 Επικύρωση της προτεινόμενης προσέγγισης και αξιολόγηση των επιδόσεων του συστήματος

Για την δοκιμή του πειράματος με σκοπό τον έλεγχο εφαρμογής μεθόδων προγνωστικής συντήρησης στο σύστημα θέρμανσης που κατασκευάστηκε, γίνανε 3 μεμονωμένα πειράματα κατά τα οποία δημιουργήσαμε παρεμβολές στο σύστημα, η ανίχνευση ανωμαλιών στα μετρηθέντα μεγέθη θα σήμαινε αυτόματα την αναγνώριση των παρεμβολών.

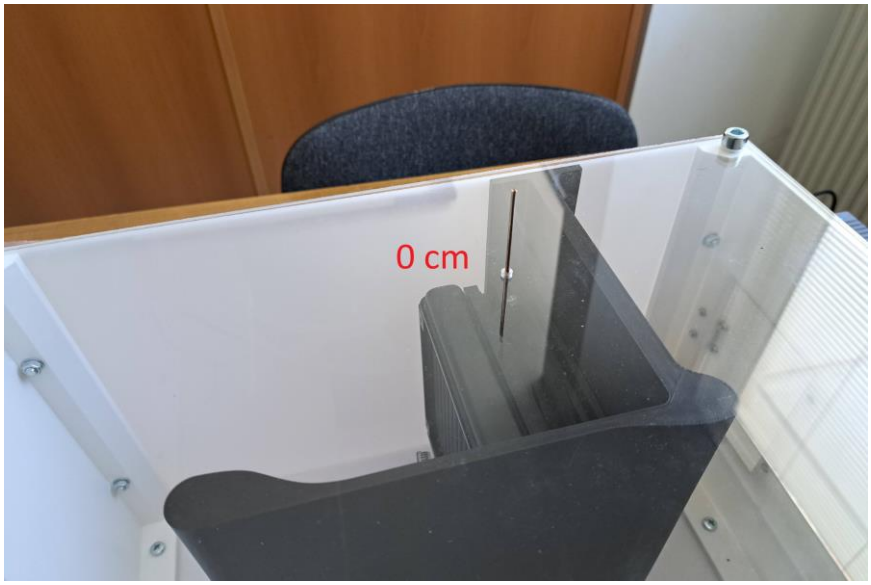
Τα πειράματα που διεξάχθηκαν περιλάμβαναν 3 σενάρια:

Σενάριο α) Μείωση ροής του ανεμιστήρα. Με την βοήθεια της κάθετης κουρτίνας ή αλλιώς ρολού, μειώνεται σταδιακά η ροή του αέρα μέσα από τον εναλλάκτη. Το πείραμα αυτό προσομοιάζει ένα βουλωμένο ψυγείο ή φίλτρα από σκόνη.

Σενάριο β) Μείωση ροής του υγρού κυκλοφορίας. Με την βοήθεια μιας βάνας στην έξοδο του κολλεκτέρ, περιορίζουμε την ροή και να αυξάνουμε την πίεση στο κολλεκτέρ απλά στραγγαλίζοντας τον σφαιροκρουνό. Το πείραμα αυτό προσομοιάζει μια κολλημένη βάνα ή βρώμα ή σκουριά στους σωλήνες του μειώνει συνεχώς την διατομή τους.

Σενάριο γ) Απώλεια θερμότητας. Ανοίγοντας τα δύο παράθυρα του δωματίου ο ανεμιστήρας ρουφάει κρύο αέρα απέξω και τον βγάζει επίσης έξω από το άλλο παράθυρο στην απέναντι πλευρά. Το πείραμα αυτό εφαρμόζει μια απώλεια θέρμανσης είτε από ανοικτό παράθυρο ή από χαλασμένη μόνωση ή τρίτη πηγή παραγωγής κρύου.

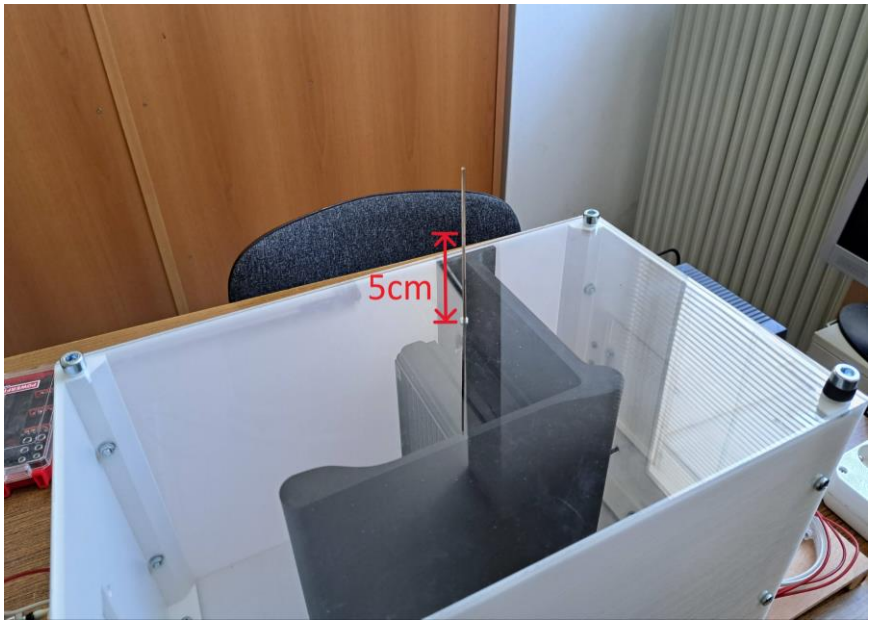
Στο πείραμα που ακολουθεί παρουσιάζεται το σενάριο α', της μείωσης της ροής του ανεμιστήρα. Η ορειχάλκινη ράβδος είναι αρχικά στο χαμηλότερο σημείο όπως φαίνεται στην Εικόνα 92, άρα το ρολό δεν είναι σηκωμένο, η λειτουργία μέχρι εκείνη την χρονική στιγμή είναι ομαλή.



Εικόνα 92: Θέση ρολού στο χαμηλότερο σημείο

Στην Εικόνα 93 που ακολουθεί, η ράβδος άρα και το ρολό είναι ανυψωμένα κατά 5 εκατοστά, αυτό σημαίνει μειωμένη επιφάνεια ροής αέρα εκ του ψυγείου κατά περίπου 50%,

δεν σημαίνει όμως ότι και ο αέρας που κυκλοφορεί είναι μειωμένος σε όγκο κατά 50%. Υπάρχει μια πτώση διαφορά πίεσης του αέρα η οποία δημιουργεί απώλειες στην απαγωγή της θερμότητας του νερού από τον αέρα που κυκλοφορεί.

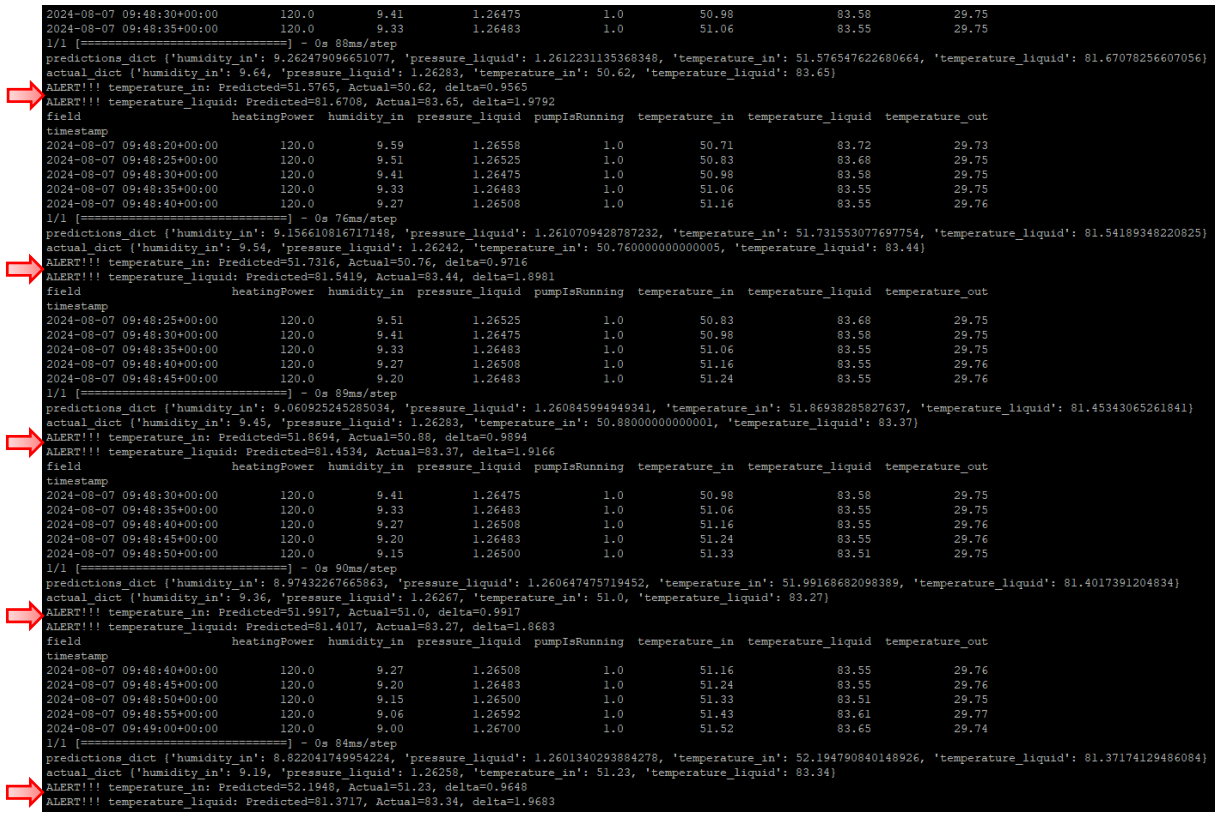


Εικόνα 93: Θέση ρολού ανυψωμένο κατά 5 εκατοστά

```
ALERT!!! temperature_in: Predicted=51.1773, Actual=50.26, delta=0.9173
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 09:23:15+00:00         120.0        11.51          1.26025            1.0           50.21               83.51            29.82
2024-08-07 09:23:20+00:00         120.0        11.39          1.26175            1.0           50.33               83.34            29.82
2024-08-07 09:23:25+00:00         120.0        11.30          1.26067            1.0           50.46               83.27            29.82
2024-08-07 09:23:30+00:00         120.0        11.19          1.26042            1.0           50.59               83.06            29.83
2024-08-07 09:23:35+00:00         120.0        11.09          1.26017            1.0           50.69               83.13            29.83
1/1 [==============================] - 0s 86ms/step
predictions_dict {'humidity_in': 10.500087589025497, 'pressure_liquid': 1.2599194169044494, 'temperature_in': 51.337342262268066, 'temperature_liquid': 81.74883842468262}
actual_dict {'humidity_in': 10.83, 'pressure_liquid': 1.2585, 'temperature_in': 50.39, 'temperature_liquid': 83.2}
ALERT!!! temperature_in: Predicted=51.3373, Actual=50.39, delta=0.9473
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 09:23:25+00:00         120.0        11.30          1.26067            1.0           50.46               83.27            29.82
2024-08-07 09:23:30+00:00         120.0        11.19          1.26042            1.0           50.59               83.06            29.83
2024-08-07 09:23:35+00:00         120.0        11.09          1.26017            1.0           50.69               83.13            29.83
2024-08-07 09:23:40+00:00         120.0        11.00          1.25975            1.0           50.79               83.20            29.83
2024-08-07 09:23:45+00:00         120.0        10.92          1.26083            1.0           50.91               83.10            29.81
1/1 [==============================] - 0s 85ms/step
predictions_dict {'humidity_in': 10.280710458755493, 'pressure_liquid': 1.2588528990745544, 'temperature_in': 51.614394187927246, 'temperature_liquid': 81.46944522857666}
actual_dict {'humidity_in': 10.61, 'pressure_liquid': 1.2595000000000003, 'temperature_in': 50.66, 'temperature_liquid': 82.89}
ALERT!!! temperature_in: Predicted=51.6144, Actual=50.66, delta=0.9544
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 09:23:30+00:00         120.0        11.19          1.26042            1.0           50.59               83.06            29.83
2024-08-07 09:23:35+00:00         120.0        11.09          1.26017            1.0           50.69               83.13            29.83
2024-08-07 09:23:40+00:00         120.0        11.00          1.25975            1.0           50.79               83.20            29.83
2024-08-07 09:23:45+00:00         120.0        10.92          1.26083            1.0           50.91               83.10            29.81
2024-08-07 09:23:50+00:00         120.0        10.83          1.26058            1.0           51.00               83.10            29.81
1/1 [==============================] - 0s 99ms/step
predictions_dict {'humidity_in': 10.184545069932938, 'pressure_liquid': 1.2584784626960754, 'temperature_in': 51.7330265045166, 'temperature_liquid': 81.38936519622803}
actual_dict {'humidity_in': 10.51, 'pressure_liquid': 1.25942, 'temperature_in': 50.78, 'temperature_liquid': 82.86}
ALERT!!! temperature_in: Predicted=51.733, Actual=50.78, delta=0.953
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 09:23:35+00:00         120.0        11.09          1.26017            1.0           50.69               83.13            29.83
2024-08-07 09:23:40+00:00         120.0        11.00          1.25975            1.0           50.79               83.20            29.83
2024-08-07 09:23:45+00:00         120.0        10.92          1.26083            1.0           50.91               83.10            29.81
2024-08-07 09:23:50+00:00         120.0        10.83          1.26058            1.0           51.00               83.10            29.81
2024-08-07 09:23:55+00:00         120.0        10.75          1.26058            1.0           51.05               83.20            29.83
1/1 [==============================] - 0s 70ms/step
predictions_dict {'humidity_in': 10.09584441781044, 'pressure_liquid': 1.2581194043159485, 'temperature_in': 51.83957576751709, 'temperature_liquid': 81.34181499481201}
actual_dict {'humidity_in': 10.41, 'pressure_liquid': 1.25833, 'temperature_in': 50.89, 'temperature_liquid': 82.89}
ALERT!!! temperature_in: Predicted=51.8396, Actual=50.89, delta=0.9496
ALERT!!! temperature_liquid: Predicted=81.3418, Actual=82.89, delta=1.5482
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 09:23:40+00:00         120.0        11.00          1.25975            1.0           50.79               83.20            29.83
2024-08-07 09:23:45+00:00         120.0        10.92          1.26083            1.0           50.91               83.10            29.81
2024-08-07 09:23:50+00:00         120.0        10.83          1.26058            1.0           51.00               83.10            29.81
2024-08-07 09:23:55+00:00         120.0        10.75          1.26058            1.0           51.05               83.20            29.83
2024-08-07 09:24:00+00:00         120.0        10.69          1.26083            1.0           51.13               83.17            29.83
1/1 [==============================] - 0s 87ms/step
predictions_dict {'humidity_in': 10.012937337716011, 'pressure_liquid': 1.2576234102249146, 'temperature_in': 51.93486690052124, 'temperature_liquid': 81.31876945495605}
actual_dict {'humidity_in': 10.35, 'pressure_liquid': 1.25817, 'temperature_in': 50.97, 'temperature_liquid': 82.96}
ALERT!!! temperature_in: Predicted=51.9349, Actual=50.97, delta=0.9649
ALERT!!! temperature_liquid: Predicted=81.3188, Actual=82.96, delta=1.6412
```

Εικόνα 94: Ανάδειξη ανωμαλιών στην λειτουργία με 50% βουλωμένο ψυγείο

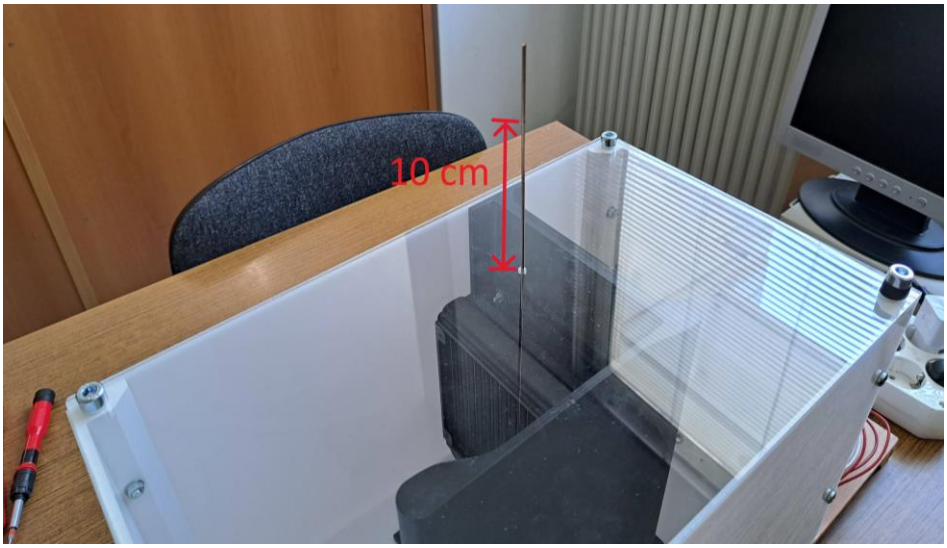
Στην Εικόνα 94 βλέπουμε αμέσως μετά την ανύψωση κατά 50% του ρολού, ότι η θερμοκρασία του δωματίου είναι σχεδόν 1˚C χαμηλότερη απ’ ότι αναμενόταν. Ενώ στη συνέχεια μαζί με την θερμοκρασία εμφανίζει και η θερμοκρασία του υγρού αυξανόμενη απόκλιση ίση με 1,6˚C.



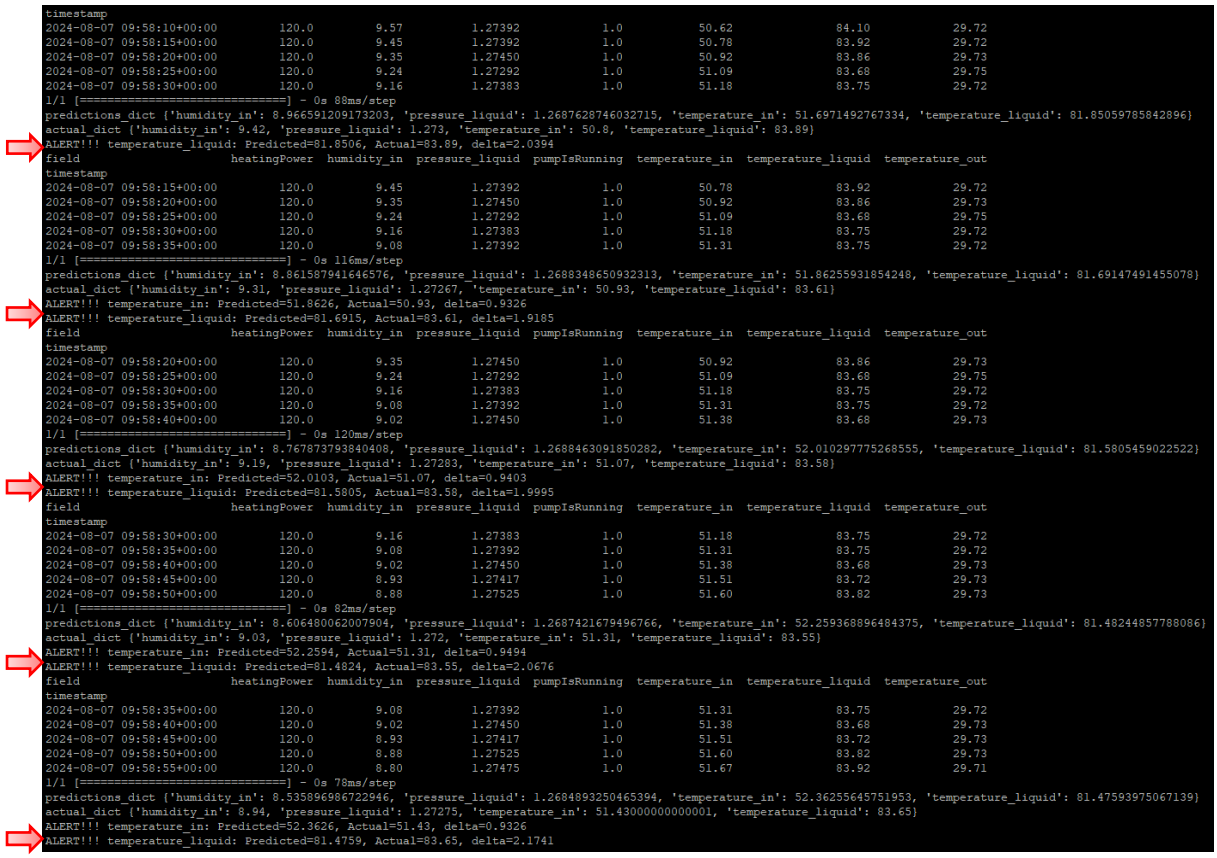
Εικόνα 95: Παγίωση ανωμαλιών στην λειτουργία με 50% βουλωμένο ψυγείο

Στην Εικόνα 95 παρατηρούμε ότι 15 ολόκληρα λεπτά μετά την ανύψωση του ρολού στα 5 εκατοστά, εμφανίζεται ακόμα μεγαλύτερη απόκλιση στη θερμοκρασία του υγρού, με το υγρό να είναι στους 83,65˚C και η πρόγνωση να είναι ίση με 81,67˚C, ενώ η θερμοκρασία του δωματίου είναι στην πραγματικότητα 50,62˚C με πρόγνωση για 51,57˚C. Η διαφορά της πρόγνωσης έναντι της πραγματικότητας με σχεδόν +2˚C στο υγρό και -1˚C στον αέρα του δωματίου εκφράζει ότι σε αντίθεση με μια ομαλή λειτουργία, εδώ το υγρό είναι πιο ζεστό αλλά το δωμάτιο πιο κρύο, η λειτουργία αυτή χαρακτηρίζεται ως μη υγιής.

Παρακάτω στην Εικόνα 96 βλέπουμε ότι το ρολό έχει ανυψωθεί στην υψηλότερη θέση, με το μέγιστό δυνατό μεγαλύτερο βούλωμα. Ενώ, στην Εικόνα 97 βλέπουμε πως το πρόγραμμα έχει πάλι εντοπίσει την προβληματική λειτουργία και η διαφορά της θερμοκρασίας του υγρού έχει ξεπεράσει τις αποκλείσεις της προηγούμενης φάσης είναι ίση με 2,2˚C.

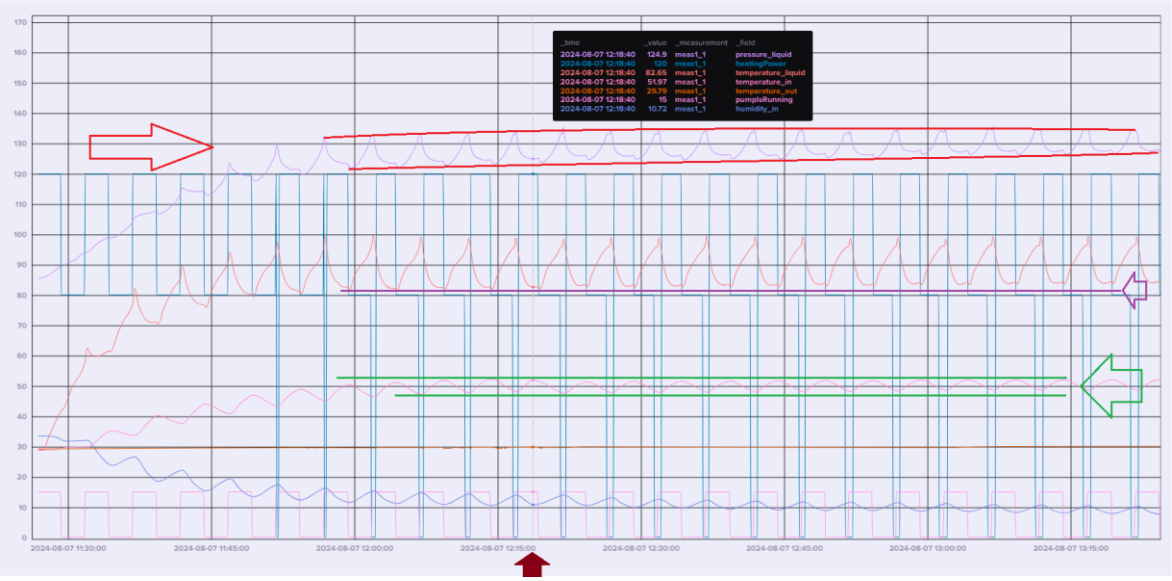


Εικόνα 96: Θέση ρολού ανυψωμένο στο μέγιστο επίπεδο ίσο με 10 εκατοστά



Εικόνα 97: Έντονες ανωμαλίες στην λειτουργία με πλήρες ανυψωμένο ρολό

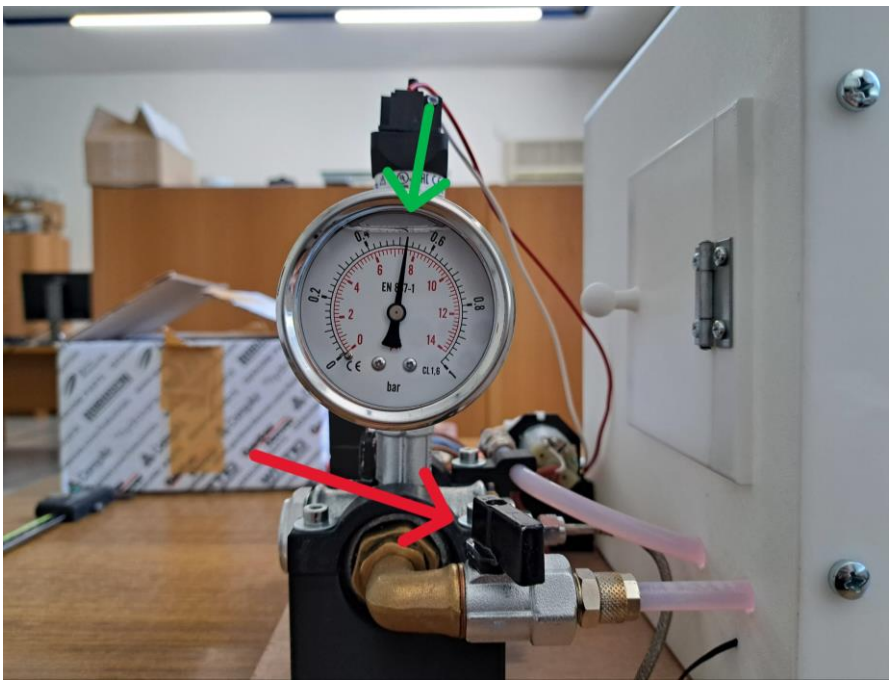
Όλα τα παραπάνω είναι αντιληπτά από έναν μηχανικό και στο παρακάτω διάγραμμα καμπυλών-μεγεθών της Εικόνα 98. Στο γράφημα, απεικονίζονται οι τιμές των μεγεθών της πίεσης, της ισχύς θέρμανσης και της θερμοκρασίας του υγρού, της θερμοκρασίας και υγρασίας του δωματίου και της θερμοκρασία του εξωτερικού χώρου και της κατάστασης λειτουργίας της αντλίας. Από τη χρονική στιγμή που είναι σημαδεμένη με ένα μπορντό βέλος επί του άξονα του χρόνου στις 12:18:40 αρχίζει η παρεμπόδιση της ροής του αέρα με το ρολό. Παρατηρούμε μία μικρότερη ταλάντωση στη καμπύλη της θερμοκρασίας του δωματίου περικλειόμενη από δυο πράσινες γραμμές, μια αύξηση του κατώτερου ορίου της καμπύλης της θερμοκρασίας του υγρού υπογραμμισμένη από μια μωβ γραμμή. Ενώ επίσης είναι αντιληπτή και μια μικρότερη ταλάντωση της πίεσης του υγρού θέρμανσης με παράλληλη αύξηση του κάτω ορίου στο χρόνο, η διαφορά αυτή φαίνεται ότι είναι εξαρτημένη από την θερμοκρασία του υγρού. Η ικανότητα του αλγορίθμου είναι να μπορεί να εντοπίζει εκ της γενέσεως τους όλες τις ανωμαλίες που ένας έμπειρος μηχανικός θα μπορούσε να αναγνωρίσει υστέρα από μελέτη του γραφήματος.



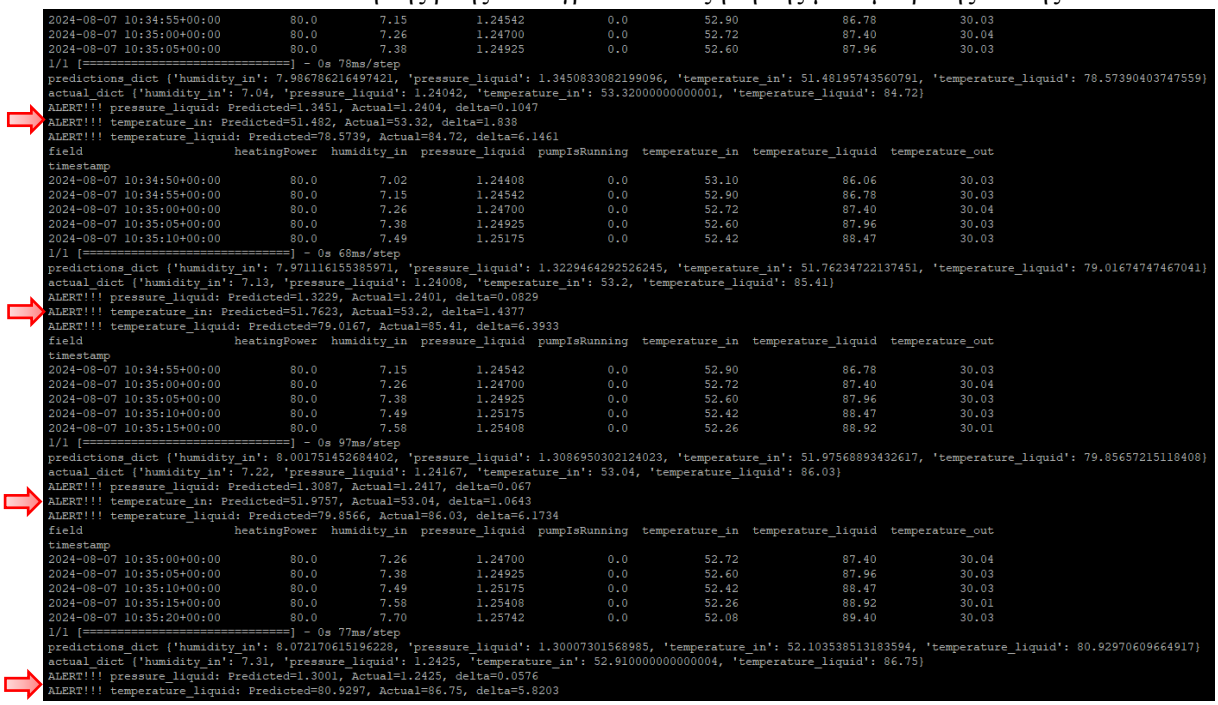
Εικόνα 98: Γραφική απεικόνιση του πειράματος α' για την απόδειξη λειτουργίας

Στο πείραμα που ακολουθεί παρουσιάζεται το σενάριο β', της μείωσης της ροής του υγρού κυκλοφορίας. Η ροή του υγρού περιορίζεται από το στραγγάλισμα ενός σφαιροκρουνού, η ροής μειώνεται στο σημείο εκείνο μειώνεται σε επίπεδο χαμηλότερο από αυτό την ροή της αντλίας. Η ρύθμιση αυτή έχει ως αποτέλεσμα λοιπόν κατά την διάρκεια λειτουργίας της αντλίας να αυξάνεται η πίεση στον χώρο του εναλλάκτη και του υδραυλικού σταυρού «κολλεκτέρ». Κατά την ρύθμιση λήφθηκε υπόψιν να μην ξεπεραστεί η μέγιστη επιτρεπτή πίεση λειτουργίας της περισταλτικής αντλίας κυκλοφορίας του υγρού.

Στην συνέχεια βλέπουμε στην Εικόνα 99 επι της πειραματικής διάταξης πως έκλεισε η βάνα «κόκκινο βέλος», με αυξημένη πίεση «πράσινο βέλος», ενώ στην Εικόνα 100 φαίνονται σε έντονο βαθμό ότι υπάρχουν αποκλείσεις σε τρία μεγέθη. Στην πίεση του υγρού υπάρχει απόκλιση στην πραγματικότητα έναντι της πρόβλεψης ίση με -83 mbar, η θερμοκρασία του δωματίου είναι 1,43˚C υψηλότερη, ενώ η θερμοκρασία του υγρού στον υδραυλικό σταυρό όπου μετριέται είναι υψηλότερη κατά 6,4˚C.



Εικόνα 99: Μείωση της ροής του υγρού και αύξηση της μανομετρικής πίεσης του

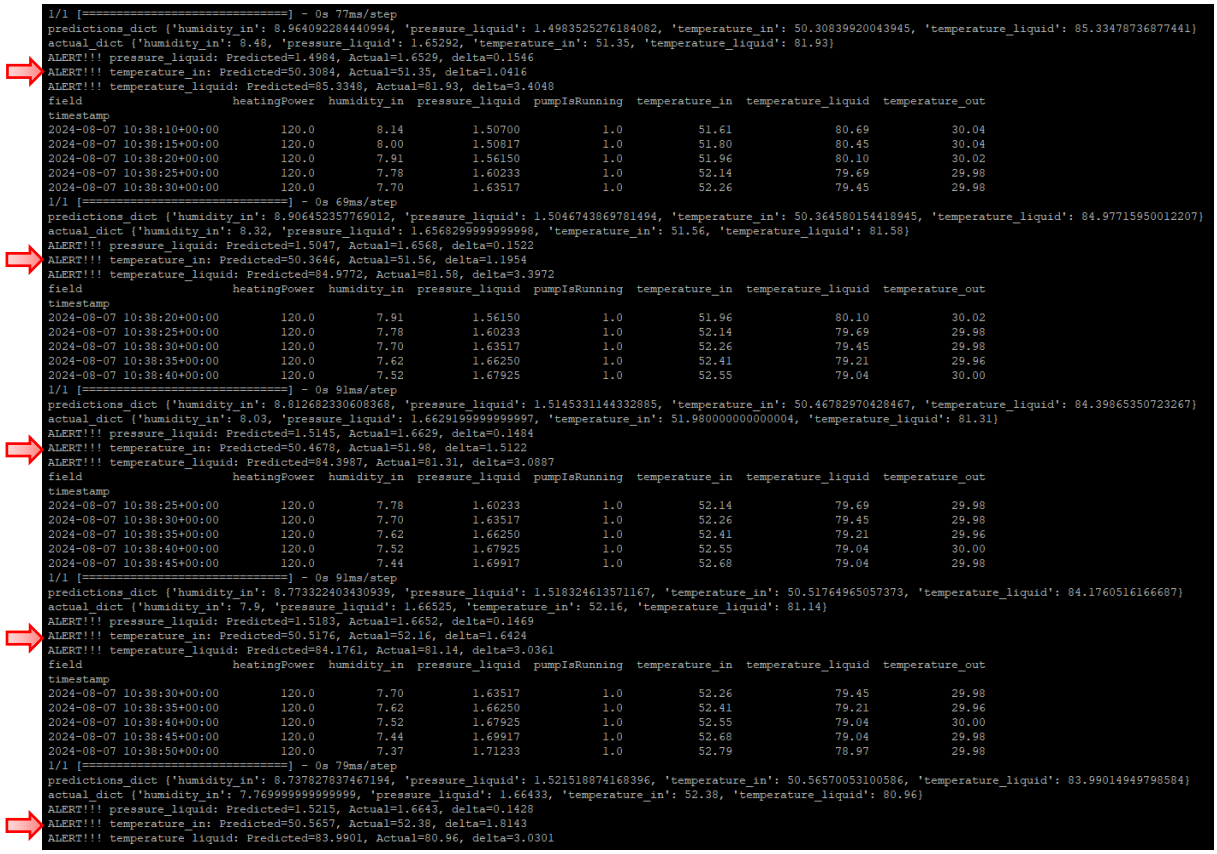


```
2024-08-07 10:34:55+00:00          80.0         7.15         1.24542           0.0          52.90          86.78          30.03
2024-08-07 10:35:00+00:00          80.0         7.26         1.24700           0.0          52.72          87.40          30.04
2024-08-07 10:35:05+00:00          80.0         7.38         1.24925           0.0          52.60          87.96          30.03
1/1 [==============================] - 0s 78ms/step
predictions_dict {'humidity_in': 7.986786216497421, 'pressure_liquid': 1.3450833082199096, 'temperature_in': 51.48195743560791, 'temperature_liquid': 78.57390403747559}
actual_dict {'humidity_in': 7.04, 'pressure_liquid': 1.24042, 'temperature_in': 53.32000000000001, 'temperature_liquid': 84.72}
ALERT!!! pressure_liquid: Predicted=1.3451, Actual=1.2404, delta=0.1047
ALERT!!! temperature_in: Predicted=51.482, Actual=53.32, delta=1.838
ALERT!!! temperature_liquid: Predicted=78.5739, Actual=84.72, delta=6.1461
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 10:34:50+00:00          80.0         7.02         1.24408           0.0          53.10          86.06          30.03
2024-08-07 10:34:55+00:00          80.0         7.15         1.24542           0.0          52.90          86.78          30.03
2024-08-07 10:35:00+00:00          80.0         7.26         1.24700           0.0          52.72          87.40          30.04
2024-08-07 10:35:05+00:00          80.0         7.38         1.24925           0.0          52.60          87.96          30.03
2024-08-07 10:35:10+00:00          80.0         7.49         1.25175           0.0          52.42          88.47          30.03
1/1 [==============================] - 0s 68ms/step
predictions_dict {'humidity_in': 7.971116155385971, 'pressure_liquid': 1.3229464292526245, 'temperature_in': 51.76234722137451, 'temperature_liquid': 79.01674747467041}
actual_dict {'humidity_in': 7.13, 'pressure_liquid': 1.24008, 'temperature_in': 53.2, 'temperature_liquid': 85.41}
ALERT!!! pressure_liquid: Predicted=1.3229, Actual=1.2401, delta=0.0829
ALERT!!! temperature_in: Predicted=51.7623, Actual=53.2, delta=1.4377
ALERT!!! temperature_liquid: Predicted=79.0167, Actual=85.41, delta=6.3933
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 10:34:55+00:00          80.0         7.15         1.24542           0.0          52.90          86.78          30.03
2024-08-07 10:35:00+00:00          80.0         7.26         1.24700           0.0          52.72          87.40          30.04
2024-08-07 10:35:05+00:00          80.0         7.38         1.24925           0.0          52.60          87.96          30.03
2024-08-07 10:35:10+00:00          80.0         7.49         1.25175           0.0          52.42          88.47          30.03
2024-08-07 10:35:15+00:00          80.0         7.58         1.25408           0.0          52.26          88.92          30.01
1/1 [==============================] - 0s 97ms/step
predictions_dict {'humidity_in': 8.001751452684402, 'pressure_liquid': 1.3086950302124023, 'temperature_in': 51.97568893432617, 'temperature_liquid': 79.85657215118408}
actual_dict {'humidity_in': 7.22, 'pressure_liquid': 1.24167, 'temperature_in': 53.04, 'temperature_liquid': 86.03}
ALERT!!! pressure_liquid: Predicted=1.3087, Actual=1.2417, delta=0.067
ALERT!!! temperature_in: Predicted=51.9757, Actual=53.04, delta=1.0643
ALERT!!! temperature_liquid: Predicted=79.8566, Actual=86.03, delta=6.1734
field                     heatingPower  humidity_in  pressure_liquid  pumpIsRunning  temperature_in  temperature_liquid  temperature_out
timestamp
2024-08-07 10:35:00+00:00          80.0         7.26         1.24700           0.0          52.72          87.40          30.04
2024-08-07 10:35:05+00:00          80.0         7.38         1.24925           0.0          52.60          87.96          30.03
2024-08-07 10:35:10+00:00          80.0         7.49         1.25175           0.0          52.42          88.47          30.03
2024-08-07 10:35:15+00:00          80.0         7.58         1.25408           0.0          52.26          88.92          30.01
2024-08-07 10:35:20+00:00          80.0         7.70         1.25742           0.0          52.08          89.40          30.03
1/1 [==============================] - 0s 77ms/step
predictions_dict {'humidity_in': 8.072170615196228, 'pressure_liquid': 1.30007301568985, 'temperature_in': 52.103538513183594, 'temperature_liquid': 80.92970609664917}
actual_dict {'humidity_in': 7.31, 'pressure_liquid': 1.2425, 'temperature_in': 52.910000000000004, 'temperature_liquid': 86.75}
ALERT!!! pressure_liquid: Predicted=1.3001, Actual=1.2425, delta=0.0576
ALERT!!! temperature_liquid: Predicted=80.9297, Actual=86.75, delta=5.8203
```

Εικόνα 100: Έντονες αποκλείσεις στις πραγματικές τιμές στη περιορισμένη ροή του υγρού

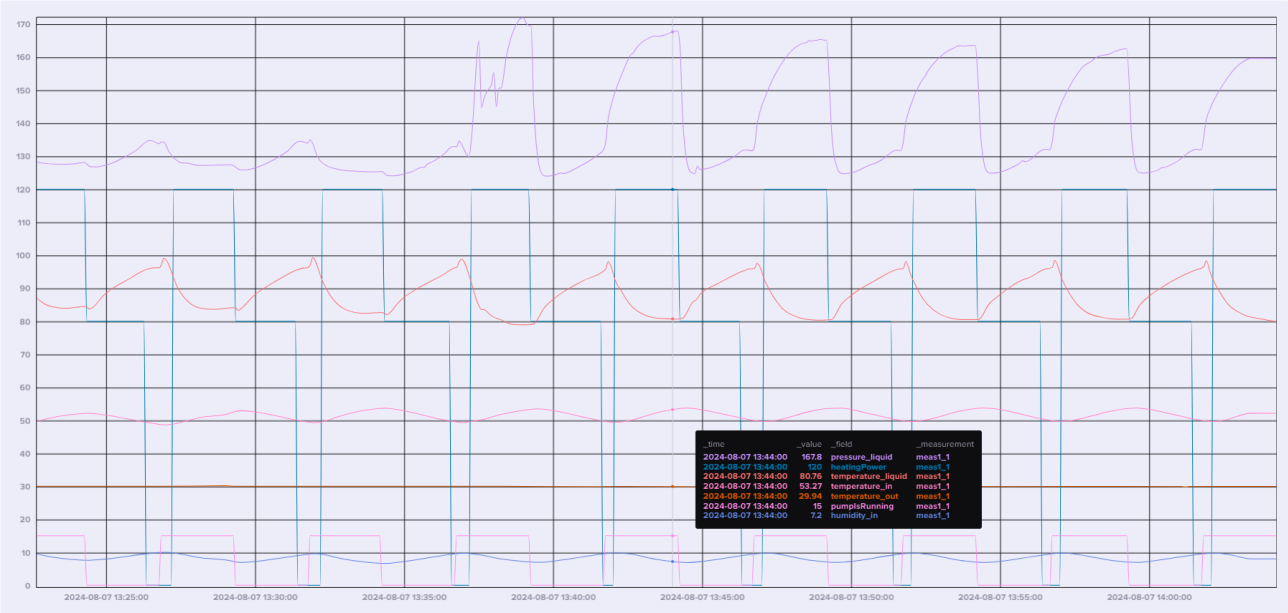
Συγκρίνοντας της αναφορές του προγράμματος στις Εικόνες Εικόνα 100 και Εικόνα 101, παρατηρούμε ότι στο πρώτο στιγμιότυπο 5 λεπτά μετα την παρεμβολή οι ανωμαλίες είναι ιδιαίτερα έντονες, καθώς περνάει ο χρόνος παρατηρούμε το δεύτερο στιγμιότυπο 8 λεπτά συνολικά μετα την παρεμβολή και βλέπουμε ότι αν και οι αποκλείσεις πλέον έχουν εδραιωθεί είναι μικρότερες. Ο αλγόριθμος εξετάζει πάντα τα τελευταία 5 λεπτά, εκείνη τη στιγμή τα δεδομένα αφορούν αποκλειστικά μια προβληματική συμπεριφορά, η ικανότητα του να εντοπίσει ραγδαίες διαφορές μειώνεται γιατί υπάρχει ο παράγοντας της μνήμης, ο αλγόριθμος εξετάζει την συμπεριφορά του συστήματος με βάση μια ήδη προβληματική συμπεριφορά. Ωστόσο επειδή η λειτουργία που παρατηρείται δεν είναι αναμενόμενη και δεν έχει παρατηρηθεί ποτέ ξανά· στην φάση της εκπαίδευσης, γι' αυτό τον λόγο γίνεται

αντιληπτή, απλά με μικρότερες διαφορές ανάμεσα στις εκτιμηθείσες και τις πραγματικές τιμές.



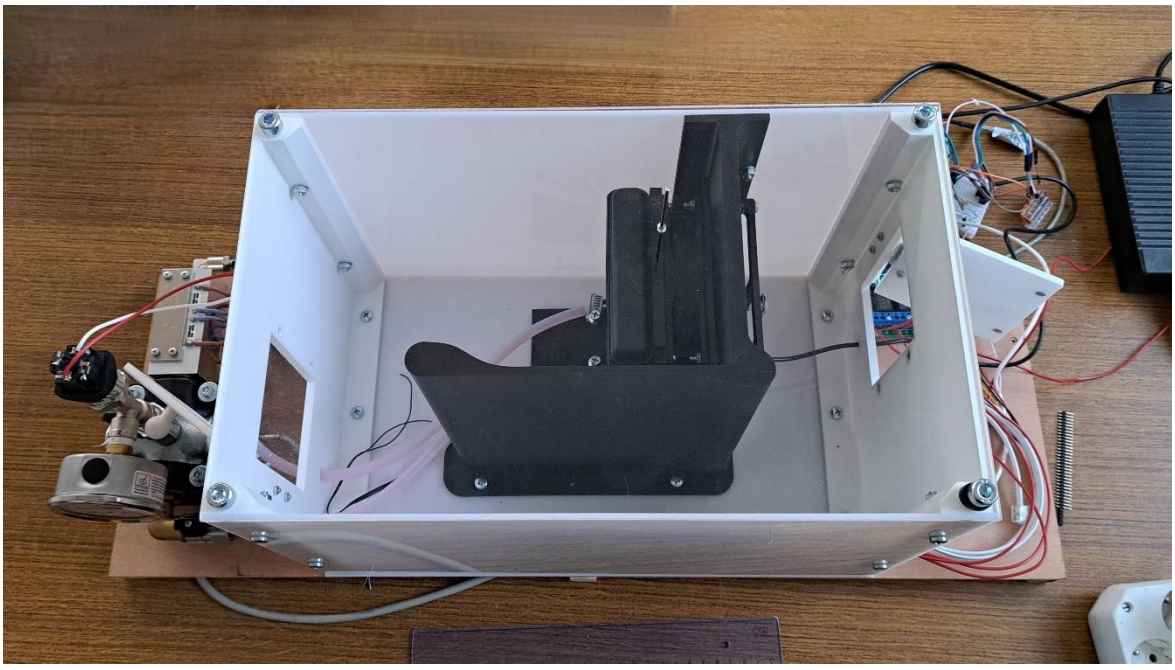
Εικόνα 101: Παγίωση των ανωμαλιών αλλά με μικρότερες αποκλείσεις στη περιορισμένη ροή υγρού

Όλα τα παραπάνω μπορούν να γίνουν αντιληπτά σε έναν μηχανικό αναλύοντας από το παρακάτω γράφημα της Εικόνα 102. Η πίεση όταν ξεκινάει η αντλία παρουσιάζει ραγδαία άνοδο και φτάνει τα 1678 mbar, έναντι ομαλής πτώσης στη νορμάλ λειτουργία. Όταν κλείνει η αντλία η πίεση κατρακυλάει στα προηγούμενα όρια λόγω της απουσίας εκβιασμένης ροής της αντλίας. Επίσης, η καμπύλη της θερμοκρασίας γίνεται πιο ομαλή με χαμηλότερες οριακές τιμές ανά περίοδο.

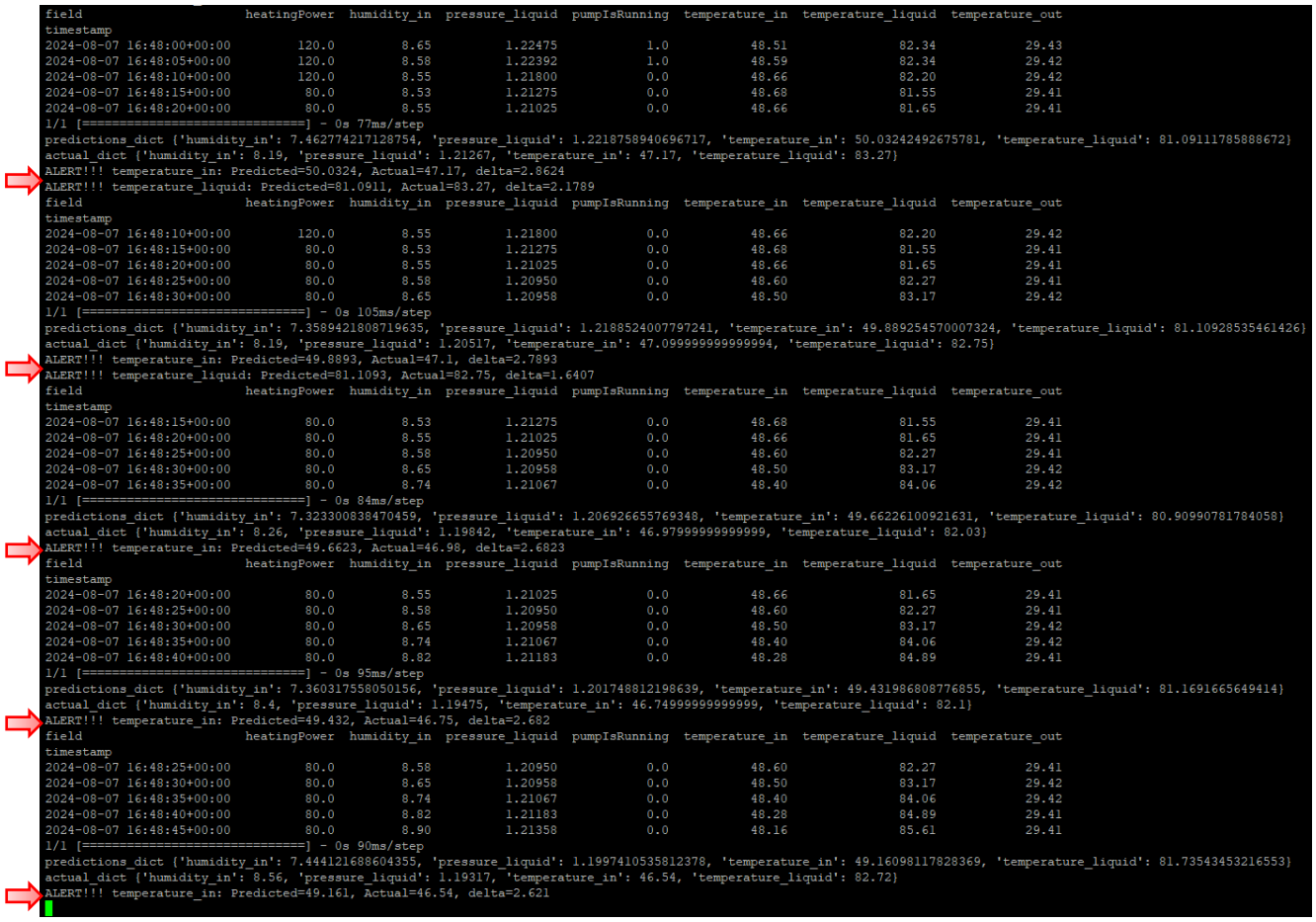


Εικόνα 102: Γραφική απεικόνιση του πειράματος β' για την απόδειξη λειτουργίας

Στο πείραμα που ακολουθεί παρουσιάζεται το σενάριο γ’, της απώλειας θερμότητας. Τα δύο απέναντι παράθυρα είναι ανοιχτά όπως φαίνεται και στην Εικόνα 103. Ένα σημαντικό μέρος αέρα από το εξωτερικό περιβάλλον, θερμοκρασίας 30˚C εισέρχεται στο δωμάτιο, θερμαίνεται και βγαίνει έξω. Οι τεράστιες θερμικές απώλειες σε αυτό το πείραμα είναι αντιληπτές στην αναφορά του αλγορίθμου στην Εικόνα 104.

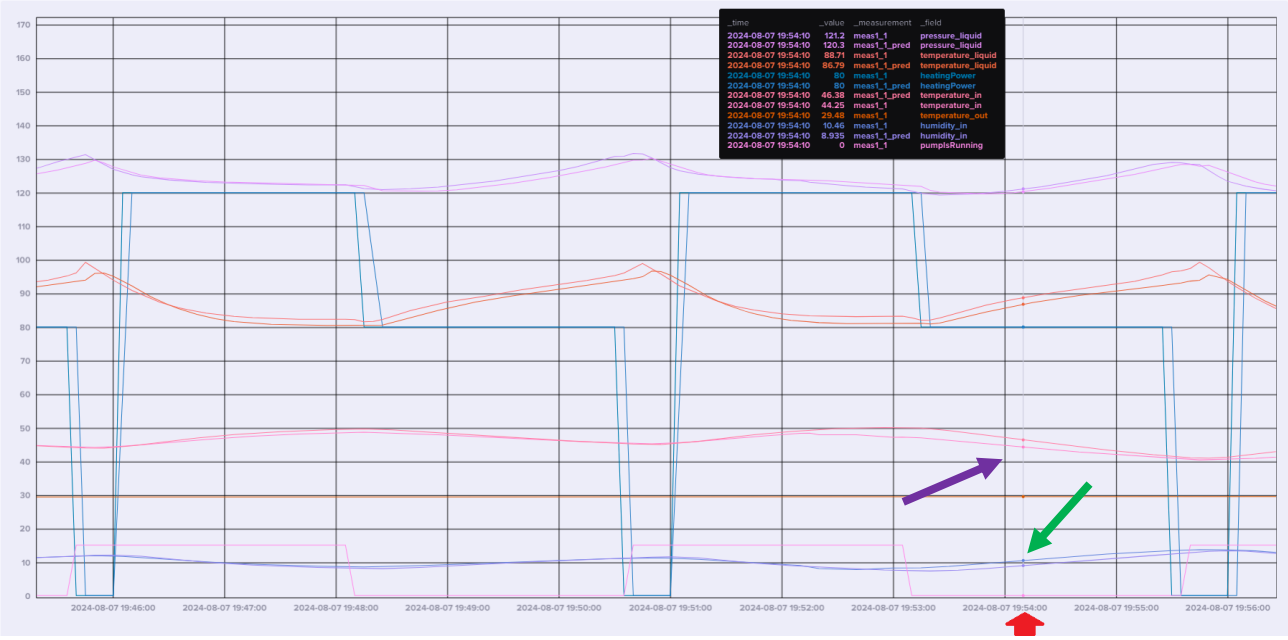


Εικόνα 103: Ανοικτά παράθυρα με σκοπό την απώλεια θερμότητας από το σύστημα



Εικόνα 104: Ανωμαλίες στις μετρήσεις κατά την απώλεια θερμότητας

Στο γράφημα παρακάτω στην Εικόνα 105 παρατηρούμε την διαφορά μεταξύ δυο ομάδων μετρήσεων, των πραγματικών “measurement: meas1_1” και των προβλεφθέντων “measurement: meas1_1_pred”, η εικόνα την χρονική στιγμή ολίγων μετα το άνοιγμα των παραθύρων που σηματοδοτείται με το κόκκινο βέλος στον άξονα του χρόνου, δείχνει μια διαφορά στις τιμές της θερμοκρασίας του χώρου «μωβ βέλος», ενώ το επίσης διαφορά παρατηρείται και στην υγρασία του χώρου «πράσινο βέλος».



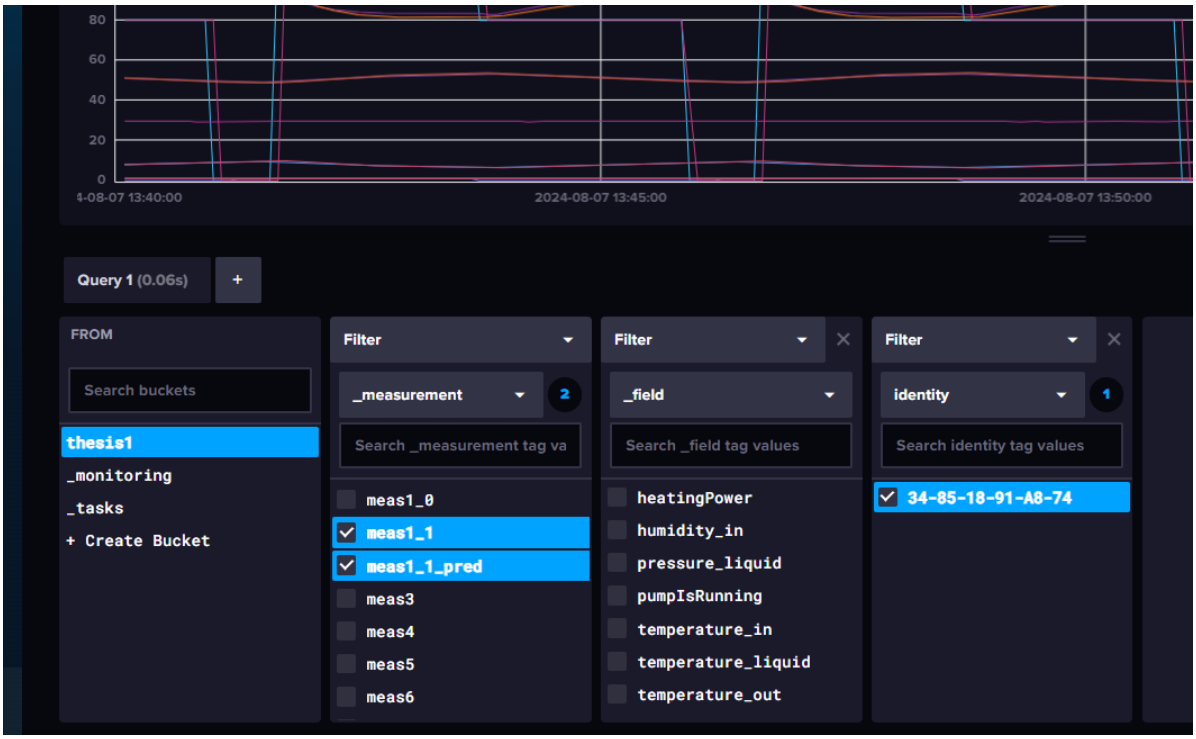
Εικόνα 105: Γραφική απεικόνιση του πειράματος γ' για την απόδειξη λειτουργίας

8 Συμπεράσματα

8.1 Συμπεράσματα από την αποθήκευση των δεδομένων

Η αποθήκευση των δεδομένων σε μια μη σχεσιακή, NoSQL βάση δεδομένων, δίνει την δυνατότητα όπως φάνηκε για ταχεία προσπέλαση των δεδομένων τόσο κατά την διάρκεια της εκπαίδευσης του αλγορίθμου αλλά και στην φάση του πειραματισμού όπου χρειάστηκε να ελέγξουμε την αρτιότητα των δεδομένων από τις μετρήσεις. Η ομαδοποίηση των δεδομένων της influxDB με φίλτρα κατά measurements και tags, έδωσε την δυνατότητα διαχωρισμού των μετρήσεων κάθε φορά που άλλαζε κάτι στον σχεδιασμό, δυο μετρήσεις (measurements) η “meas1_1” και “meas1_1_pred” διαφοροποιούν και ομαδοποιούν ως προς τον χρόνο τις πραγματικές μετρήσεις έναντι των εκτιμηθέντων. Ενώ ορίζοντας ως tag ενα αναγνωριστικό για τον κάθε μικροελεγκτή, την διεύθυνση MAC της κάρτας Wi-Fi του ESP-32 μπορούμε να διαχωρίσουμε πλήρως τα δεδομένα. Παρακάτω στην Εικόνα 106, μπορεί κανείς να καταλάβει με ευκολία την αξία αυτής της ομαδοποίησης.

Συνεπώς, βάση των δοκιμών που έγιναν το περιβάλλον βάσης δεδομένων InfluxDB χαρακτηρίζεται ως το πλέον κατάλληλο για την διπλωματική εργασία αλλά και για μελλοντικές εφαρμογές της προγνωστικής συντήρησης ηλεκτρομηχανολογικών συστημάτων σε ευρεία κλίμακα.

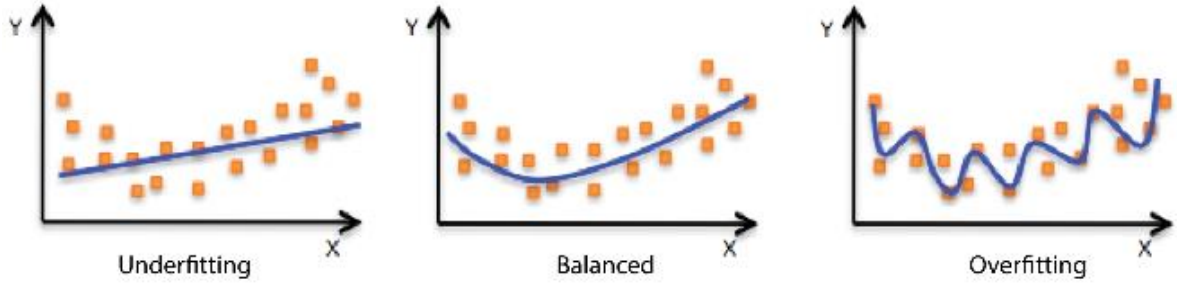


Εικόνα 106: Δυνατότητες φίλτρων InfluxDB

8.2 Συμπεράσματα από την εκπαίδευση του μοντέλου της τεχνητής νοημοσύνης

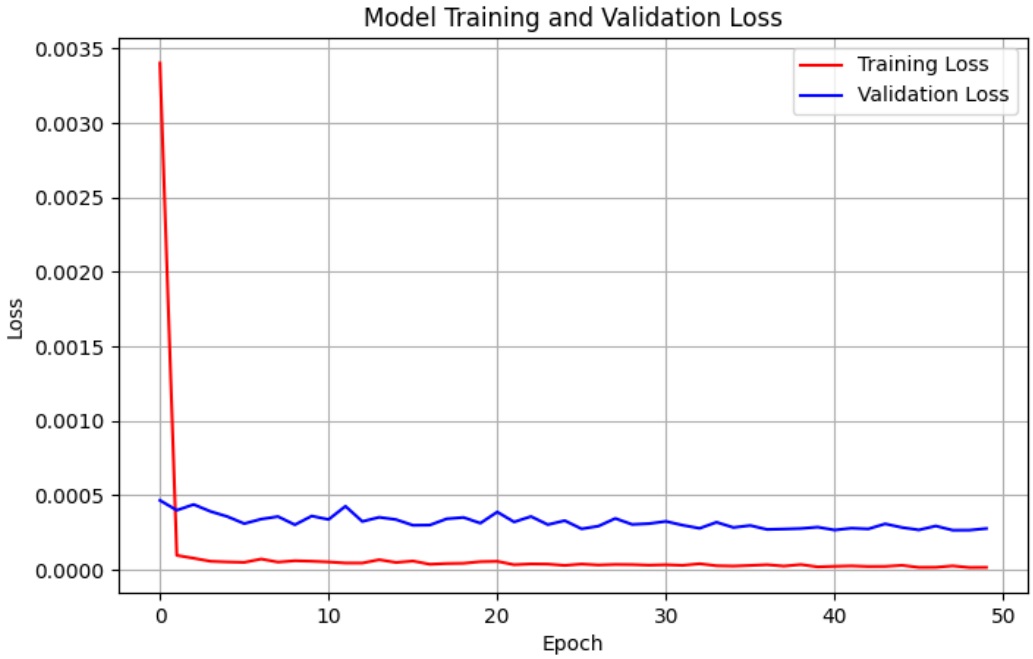
Η εκπαίδευση ενός μοντέλου τεχνητής νοημοσύνης αποτελεί μια από τις πιο σημαντικές διαδικασίες, ενώ θεωρείτε βήμα 0 για την ορθή λειτουργεία μιας εφαρμογής που ενσωματώνει διαδικασίες τεχνικής νοημοσύνης με μικρό σφάλμα. Η αρχιτεκτονική του μοντέλου πρέπει να μελετηθεί λεπτομερώς ενώ η χρήση όσο των δυνατών περισσότερων

δεδομένων επιτρέπει την καλύτερη εκπαίδευση του αλγορίθμου. Αφετέρου δε πρέπει να δοθεί έμφαση στην αποτροπή της υπερπροσαρμογής “overfitting” του μοντέλου πάνω στις μετρήσεις. Το πρόβλημα της υπό/υπέρ προσαρμογής μπορεί να γίνει καλύτερα αντιληπτό από τα διαγράμματα της Εικόνα 107.



Εικόνα 107: α: Υποπροσαρμογή, β: Βέλτιστη Προσαρμογή και γ:Υπερπροσαρμογή [60]

Στην Εικόνα 108, βλέπουμε ένα γράφημα που μπορέσαμε να εξάγουμε κατά την διάρκεια της εκπαίδευσης του αλγορίθμου και απεικονίζει το σφάλμα του αλγόριθμο και το σφάλμα των μετρήσεων ως θόρυβο. Παρατηρούμε ο αλγόριθμος που δημιουργήσαμε έχει μικρότερο σφάλμα από τα πραγματικά δεδομένα, προκύπτει λοιπόν ότι σε ένα βαθμό έχουμε υπερπροσαρμογή του αλγορίθμου “overfitting”.



Εικόνα 108: Σφάλμα εκπαίδευσης έναντι σφάλματος τιμών

8.3 Συμπεράσματα από την πρόβλεψη ανωμαλιών στη σχέση των δεδομένων

Όπως πολύ σωστά αναμενόταν ανάλογα την παρεμβολή που γινόταν στο μοντέλο διαφορετικά σφάλματα πρόκυπταν στις πραγματικές τιμές έναντι των προβλεφθείσων. Η εύρεση διαφορετικών ανωμαλιών, σηματοδοτεί μια πολύ ελπιδοφόρα βλέψη για την προγνωστική συντήρηση. Με κατάλληλο χαρακτηρισμό και διαφοροποίηση των ανωμαλιών που παρατηρούνται από τον αλγόριθμο και με εφαρμογή ευρετικών μεθόδων, μπορούμε να εντοπίσουμε καλύτερα, γρηγορότερα και με μεγαλύτερη ακρίβεια σε ποιο εξάρτημα του ηλεκτρομηχανολογικού εξοπλισμού μπορεί να είναι η βλάβη.

Παρατηρήθηκε επίσης πώς στην συνέχεια των παρεμβολών υπήρχε μια παγίωση των ανωμαλιών ανάμεσα στις πραγματικές τιμές και στις προβλεφθείσες με μείωση της διαφοράς αυτής στο βάθος του χρόνου. Η μείωση αυτή είναι φαινόμενο της μνήμης του αλγορίθμου ανά επίπεδο του νευρωνικού δικτύου όπου γίνεται συνέχιση ανάμεσα στο ποιες τιμές χαρακτηρίζουν ομαλή και μη λειτουργία στον πραγματικό χρόνο.

8.4 Μελλοντικές Κατευθύνσεις

Στον απόηχο της διπλωματικής εργασίας μπορέσαμε να έχουμε μια λεπτομερή εικόνα των περιορισμών που είχαμε και να βρούμε ποια θα μπορούσαν να είναι τα επόμενα βήματα με σκοπό την δημιουργία μίας εμπορικά διαθέσιμης εφαρμογής προληπτικής συντήρησης για συστήματα ψύξης θέρμανσης.

Οι μελλοντικές κατευθύνσεις αφορούν:

- Ένα ολοκληρωμένο περιβάλλον διεπαφής, όπου ο χρήστης θα έχει την δυνατότητα να βλέπει τις μετρήσεις με ευκολία ακόμα και από το κινητό του τηλέφωνο, να ενημερώνεται για τις βλάβες καθώς και να τις επαληθεύει ή να καταχωρεί συντηρήσεις με σκοπό την καλύτερη εκπαίδευση του αλγορίθμου.
- Τον χαρακτηρισμό των ανωμαλιών από ειδικούς και μετάφραση σε στοχευμένα μηνύματα για την υγεία του εξοπλισμού. Η διαδικασία αυτή δεν μπορεί να γίνει μόνο από έναν ειδικό στον τομέα της ηλεκτρονικής και της τεχνητής νοημοσύνης, ωστόσο η συνεργασία του με ένα ειδικό στους καυστήρες μπορεί να οδηγήσει στον απαραίτητο χαρακτηρισμό με σκοπό την βελτίωση της εμπειρίας του χρήστη.
- Την επέκταση και σε άλλες εφαρμογές μηχανοστασίων. Είναι αλήθεια πως τα συστήματα θέρμανσης και ψύξης είναι ένας τομέας με ιδιαίτερο ενδιαφέρον στην κοινωνία και στους μηχανικούς, λόγω της συχνότατης εμφάνισή τους στις κτηριακές μονάδες. Ωστόσο, όλα τα ηλεκτρομηχανολογικά συστήματα παρουσιάζουν κοινά με τα συστήματα θέρμανσης, είτε πρόκειται για έναν κινητήρα εσωτερικής καύσης σε ένα αυτοκίνητο είτε έναν ατμοστρόβιλο σε ένα θερμικό εργοστάσιο παραγωγής ηλεκτρικής ενέργειας, θέλουμε να εξασφαλίσουμε την ομαλή λειτουργία του εξοπλισμού, μπορούμε να μετρήσουμε μεγέθη όπως θερμοκρασία και πίεση και να καταλήξουμε σε χρήσιμα συμπεράσματα. Πιθανότητα κάθε εφαρμογή να απαιτεί την υλοποίηση διαφορετικής αρχιτεκτονική του μοντέλου της τεχνικής νοημοσύνης, ενώ η εκπαίδευση του κάθε συστήματος θα γίνεται ξεχωριστά για όσο διάστημα έχουμε ομαλή λειτουργία.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] T. Dr Aditya, C. Abhitesh, S. Mayank, M. Manish, K. C. Mayank, and T. Mohit, "Internet of Things (IoT): Research, Architectures and Applications," *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 3, pp. 23–27, Mar. 2018.

[2] D. Verma *et al.*, "Internet of things (IoT) in nano-integrated wearable biosensor devices for healthcare applications," *Biosensors and Bioelectronics: X*, vol. 11, p. 100153, Sep. 2022, doi: 10.1016/j.biosx.2022.100153.

[3] L. S. Vailshery, "IoT connected devices worldwide 2019-2030," Statista. Accessed: Sep. 03, 2023. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[4] D. Soni and A. Makwana, "A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)," presented at the International Conference of Telecommunication, Bharath Institute of Higher Education and Research, India, Apr. 2017.

[5] A. Prof, "A review paper on 'IOT' & It"s Smart Applications," 2016. Accessed: Mar. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/A-review-paper-on-%E2%80%9CIOT%E2%80%9D-%26-It%E2%80%9Fs-Smart-Applications-Prof/d778c2d305badd69ea1ad69edee9aef1a3e80507

[6] M. U.Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A Review on Internet of Things (IoT)," *IJCA*, vol. 113, no. 1, pp. 1–7, Mar. 2015, doi: 10.5120/19787-1571.

[7] N. Staff, "Concept of the Internet of Things (IoT) and its implications for smart homes," Physics Network. Accessed: May 06, 2024. [Online]. Available: https://physicsnetwork.org/concept-of-the-internet-of-things-iot-and-its-implications-for-smart-homes.html

[8] V.-D. Gavra and O. A. Pop, "Usage of ZigBee and LoRa wireless technologies in IoT systems," *2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 221–224, Oct. 2020, doi: 10.1109/SIITME50350.2020.9292150.

[9] A. I. Ali, S. Z. Partal, S. Kepke, and H. P. Partal, "ZigBee and LoRa based Wireless Sensors for Smart Environment and IoT Applications," in *2019 1st Global Power, Energy and Communication Conference (GPECOM)*, Jun. 2019, pp. 19–23. doi: 10.1109/GPECOM.2019.8778505.

[10] M. M. Elsherbini, B. M. Elhalawany, O. M. Ali, and S. I. Abd Elkarim, "Machine learning approaches for LoRa networks:a survey," *IJSCC*, vol. 14, no. 4, p. 10057436, 2023, doi: 10.1504/IJSCC.2023.10057436.

[11] M. Alenezi, K. K. Chai, Y. Chen, and S. Jimaa, "Ultra-dense LoRaWAN: Reviews and challenges," *IET Communications*, vol. 14, no. 9, pp. 1361–1371, Jun. 2020, doi: 10.1049/iet-com.2018.6128.

[12] S. Lemeš, *The Role of Software Engineering in Industry 4.0*. 2023. doi: 10.5644/PI2023.209.05.

[13] A. I. Ali and S. Zorlu Partal, "Development and performance analysis of a ZigBee and LoRa-based smart building sensor network," *Front. Energy Res.*, vol. 10, Aug. 2022, doi: 10.3389/fenrg.2022.933743.

[14] J. R. Santana, P. Sotres, J. Pérez, L. Sánchez, J. Lanza, and L. Muñoz, "Assessing LoRaWAN radio propagation for smart parking service: An experimental study," *Computer Networks*, vol. 235, p. 109962, Nov. 2023, doi: 10.1016/j.comnet.2023.109962.

[15] "Using LoRaWAN End Devices on The Things Network," Hackster.io. Accessed: Mar. 23, 2024. [Online]. Available: https://www.hackster.io/nootropicdesign/using-lorawan-end-devices-on-the-things-network-206a86

[16] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, NIST Special Publication (SP) 800-145, Sep. 2011. doi: 10.6028/NIST.SP.800-145.

[17] V. Gandhi and C. Kumbharana, "Comparative study of Amazon EC2 and Microsoft Azure cloud architecture," *IJANA*, pp. 117–123, Sep. 2018.

[18] D. Verma and A. Taqa, "Integrated System: IoT and Cloud Computing," vol. 9, pp. 2394–4331, Sep. 2022.

[19] P. Muralidhara, "IoT applications in cloud computing for smart devices," *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*, vol. 1, no. 1, Art. no. 1, Mar. 2017.

[20] "The History of Cloud Computing: Two Decades in Review (Part1) | LinkedIn." Accessed: May 06, 2024. [Online]. Available: https://www.linkedin.com/pulse/emergence-evolution-cloud-computing-two-decades-review-ben-david/

[21] M. Cakir, "Cloud Computing: IaaS general purpose VM performance comparison between Microsoft Azure, Amazon AWS, and Google Cloud GCP," Master Thesis Computer Science (Networks and Security), Staffordshire University, Stafford, 2023. Accessed: Feb. 07, 2024. [Online]. Available: https://rgdoi.net/10.13140/RG.2.2.34476.74885

[22] S. Goyal, "Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review," *IJCNIS*, vol. 6, no. 3, pp. 20–29, Feb. 2014, doi: 10.5815/ijcnis.2014.03.03.

[23] D. Rani, "A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 6, pp. 458–461, Jun. 2014.

[24] A. Kovari and P. Dukan, "KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE," in *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*, Sep. 2012, pp. 335–339. doi: 10.1109/SISY.2012.6339540.

[25] A. Kawser and K. Pinto, "A Comparative study one of the Hadoop distribution Hortonworks with Amazon Web Service (AWS) and Microsoft Azure," Nov. 2020, [Online]. Available: https://www.researchgate.net/publication/346049775_A_Comparative_study_one_of

_the_Hadoop_distribution_Hortonworks_with_Amazon_Web_Service_AWS_and_Microsoft_Azure

[26] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Jan. 2008, pp. 791–798. doi: 10.1109/COMSWA.2008.4554519.

[27] "MQTT - The Standard for IoT Messaging." Accessed: May 06, 2024. [Online]. Available: https://mqtt.org/

[28] M. Blackstock and R. Lea, "Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED)," in *Proceedings of the 5th International Workshop on Web of Things*, in WoT '14. New York, NY, USA: Association for Computing Machinery, Oct. 2014, pp. 34–39. doi: 10.1145/2684432.2684439.

[29] K. Ferencz and J. Domokos, "Using Node-RED platform in an industrial environment," presented at the XXXV. Jubileumi Kandó Konferencia, Sapientia Hungarian University of Transylvania, Tîrgu Mureș, Romania, Feb. 2020.

[30] D. Clerissi, M. Leotta, G. Reggio, and F. Ricca, "Towards an approach for developing and testing Node-RED IoT systems," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Ensemble-Based Software Engineering*, in EnSEmble 2018. New York, NY, USA: Association for Computing Machinery, Nov. 2018, pp. 1–8. doi: 10.1145/3281022.3281023.

[31] S. Chanthakit and C. Rattanapoka, "MQTT Based Air Quality Monitoring System using Node MCU and Node-RED," in *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, Jul. 2018, pp. 1–5. doi: 10.1109/ICT-ISPC.2018.8523891.

[32] M. Nasar and M. Abu Kausar, "Suitability Of Influxdb Database For IoT Applications," Aug. 2019, doi: 10.35940/ijitee.J9225.0881019.

[33] S. Włostowska, "Comparison of SQL, NoSQL and TSDB database systems for smart buildings and smart metering applications," *ELECTROTECHNICAL REVIEW*, vol. 1, no. 11, pp. 9–14, Nov. 2023, doi: 10.15199/48.2023.11.02.

[34] K. Rudolph, *A Comparison of NoSQL Time Series Databases*. 2015. Accessed: Feb. 25, 2024. [Online]. Available: https://www.grin.com/document/299975

[35] "DB-Engines Ranking," DB-Engines. Accessed: May 06, 2024. [Online]. Available: https://db-engines.com/en/ranking/time+series+dbms

[36] B. Mahesh, *Machine Learning Algorithms -A Review*. 2019. doi: 10.21275/ART20203995.

[37] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress, 2015.

[38] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," *Translational Vision Science & Technology*, vol. 9, no. 2, p. 14, Feb. 2020, doi: 10.1167/tvst.9.2.14.

[39] A. D. Scaife, "Improve predictive maintenance through the application of artificial intelligence: A systematic review," *Results in Engineering*, vol. 21, p. 101645, Mar. 2024, doi: 10.1016/j.rineng.2023.101645.

[40] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach Learn*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/s10994-019-05855-6.

[41] C. M. Bishop, *Pattern recognition and machine learning*. in Information science and statistics. New York: Springer, 2006.

[42] W. J. Lee, H. Wu, H. Yun, H. Kim, M. B. G. Jun, and J. W. Sutherland, "Predictive Maintenance of Machine Tool Systems Using Artificial Intelligence Techniques Applied to Machine Condition Data," *Procedia CIRP*, vol. 80, pp. 506–511, Jan. 2019, doi: 10.1016/j.procir.2018.12.019.

[43] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0," *Sustainability*, vol. 12, no. 19, Art. no. 19, Jan. 2020, doi: 10.3390/su12198211.

[44] A. Nasser and H. AL-Khazraji, "A hybrid of convolutional neural network and long short-term memory network approach to predictive maintenance," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, Art. no. 1, Feb. 2022, doi: 10.11591/ijece.v12i1.pp721-730.

[45] X. Bampoula, G. Siaterlis, N. Nikolakis, and K. Alexopoulos, "A Deep Learning Model for Predictive Maintenance in Cyber-Physical Production Systems Using LSTM Autoencoders," *Sensors*, vol. 21, no. 3, Art. no. 3, Jan. 2021, doi: 10.3390/s21030972.

[46] "ESP32-S3-DevKitC-1 - ESP32-S3 - — ESP-IDF Programming Guide v4.4.5 documentation." Accessed: May 06, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v4.4.5/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html

[47] R. Radetić, M. Pavlov-Kagadejev, and N. Milivojević, "The analog linearization of Pt100 working characteristic," *Serbian Journal of Electrical Engineering*, vol. 12, no. 3, pp. 345–357, 2015.

[48] K. M. GmbH, "Pressure transmitters," Mexico. Accessed: Aug. 10, 2024. [Online]. Available: https://mx.krohne.com/en/products/pressure-measurement/pressure-transmitters

[49] "Overview | Adafruit MAX31865 RTD PT100 or PT1000 Amplifier | Adafruit Learning System." Accessed: Aug. 10, 2024. [Online]. Available: https://learn.adafruit.com/adafruit-max31865-rtd-pt100-amplifier/overview

[50] "Electronics | Free Full-Text | A Novel Low-Power Synchronous Preamble Data Line Chip Design for Oscillator Control Interface." Accessed: Aug. 10, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/9/9/1509

[51] A. Industries, "Adafruit Si7021 Temperature & Humidity Sensor Breakout Board." Accessed: Aug. 10, 2024. [Online]. Available: https://www.adafruit.com/product/3251

[52] "Choose Training Configurations for LSTM Using Bayesian Optimization - MATLAB & Simulink - MathWorks América Latina." Accessed: May 06, 2024. [Online]. Available: https://la.mathworks.com/help/deeplearning/ug/experiment-with-training-configurations-for-sequence-regression.html

[53] "5.3.7. Predictive Maintenance Demo — Processor SDK Linux Documentation." Accessed: May 06, 2024. [Online]. Available: https://software-dl.ti.com/processor-sdk-linux/esd/docs/06_03_00_106/linux/Examples_and_Demos/Application_Demos/PdM_Anomaly_Detection.html

[54] O. Serradilla, E. Zugasti, J. Rodriguez, and U. Zurutuza, "Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects," *Appl Intell*, vol. 52, no. 10, pp. 10934–10964, Aug. 2022, doi: 10.1007/s10489-021-03004-y.

[55] "Enermax - LIQMAX III High Pressure - All-in-One RGB CPU Liquid Cooler," Enermax. Accessed: May 19, 2024. [Online]. Available: https://www.enermaxeu.com/products/cpu-cooling/liquid-cooling/liqmax-iii-hf/

[56] "ΣΩΛΗΝΕΣ ΤΕΦΛΟΝ (PTFE)," Tecnopneumatic A.E. Accessed: May 19, 2024. [Online]. Available: https://www.tecnopneumatic.gr/product/solinas-teflon-PTFE/

[57] "Pressure sensor - WIKA." Accessed: May 19, 2024. [Online]. Available: https://www.wika.com/en-tr/lp_pressure_sensor.WIKA

[58] "Peristaltic pump dispensing - Novanta IMS." Accessed: Aug. 31, 2024. [Online]. Available: https://www.novantaims.com/success-stories/peristaltic-pumps/, https://www.novantaims.com/success-stories/peristaltic-pumps/

[59] "Liquid Peristaltic Pumps | SR 10-30 Series," thomas. Accessed: Aug. 31, 2024. [Online]. Available: https://www.thomaspumps.com/en-us/peristaltic-liquid-pumps/sr-10-30-series

[60] "Model Fit: Underfitting vs. Overfitting - Amazon Machine Learning." Accessed: Sep. 01, 2024. [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Παράρτημα Α' – Κώδικας μικροελεγκτή

Στο παρών παράρτημα επισυνάπτεται ο κώδικα της CPP του μικροελεγκτή.

```
1   //    UNIVERSITY OF WEST ATTICA
2   //    Industrial Design and Production Engineering
3   //    Egaleo, September 2024
4   //    Author:   Georgios Dimas  /  Student Number: 222017029
5   //    Part of thesis diploma  "Data collection, processing, analysis and
6   //    management in a cooling/heating system"
7
8   #include <stdio.h>
9   #include "freertos/FreeRTOS.h"
10  #include "freertos/task.h"
11  #include <freertos/timers.h>
12  #include "freertos/semphr.h"
13  #include "freertos/queue.h"
14  #include "freertos/event_groups.h"
15  #include "sdkconfig.h"
16  #include "esp_log.h"
17  #include "driver/i2c.h"
18  #include "driver/gpio.h"
19  #include "driver/spi_master.h"
20  #include <stdint.h>
21  #include <stddef.h>
22  #include "esp_wifi.h"
23  #include "nvs_flash.h"
24  #include "esp_event.h"
25  #include "esp_netif.h"
26  #include "esp_system.h"
27  #include "esp_mac.h"
28
29  #include "mqtt_client.h"
30
31  #include "lwip/err.h"
32  #include "lwip/sys.h"
33
34  #include "lwip/sockets.h"
35  #include "lwip/dns.h"
36  #include "lwip/netdb.h"
37
38  #include <stdlib.h>
39  #include <string.h>
40
41  #include "Adafruit_ADS1X15.h"   //Credits to  author   K.Townsend (Adafruit
    Industries)
42  #include "Adafruit_MAX31865.h"  //Credits to  author   Limor Fried/Ladyada
    (Adafruit Industries)
43
44  #include <stdlib.h>
45  #include <esp_log.h>
46  #include <driver/i2c.h>
47
48  // Global variables to manage pump state
49  #define RELAY_PIN GPIO_NUM_7
50  #define RELAY_HEAT_PIN GPIO_NUM_15
51  #define RELAY_HEAT2_PIN GPIO_NUM_16
52  #define RELAY_HEAT3_PIN GPIO_NUM_17
53
54
```

```
55  #define TEMP_CHECK_PERIOD_MS 5000 // Time between temperature checks in
    milliseconds
56  #define MAX_RUN_TIME_MS 150000    // Maximum run time in milliseconds
57
58
59  //static bool pumpIsRunning = false;
60  static uint32_t var_wait_1 = 0; // Time to wait before turning on the pump
    again
61  static TickType_t var_timer_1 = 0; // Timestamp for when the pump was last
    turned on
62
63  #define PIN_NUM_CLK     GPIO_NUM_12
64  #define PIN_NUM_MISO    GPIO_NUM_13
65  #define PIN_NUM_MOSI    GPIO_NUM_11
66  #define PIN_NUM_CS      GPIO_NUM_10
67
68  // The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000
69  #define RREF      4300.0
70
71  // The 'nominal' 0-degrees-C resistance of the sensor
72  // 100.0 for PT100, 1000.0 for PT1000
73  #define RNOMINAL  1000.0
74
75  #define I2C_MASTER_NUM_0    I2C_NUM_0
76  #define I2C_MASTER_SDA_IO_0 GPIO_NUM_9
77  #define I2C_MASTER_SCL_IO_0 GPIO_NUM_8
78
79  #define I2C_MASTER_NUM_1    I2C_NUM_1
80  #define I2C_MASTER_SDA_IO_1 GPIO_NUM_5
81  #define I2C_MASTER_SCL_IO_1 GPIO_NUM_6
82
83  #define ADS1X15_ADDRESS 0x48
84  #define SI7021_ADDRESS 0x40
85
86  // Commands for Si7021
87  #define SI_7021_ADDRESS                 0x40
88  #define SI_7021_MEASURE_HUMIDITY        0xE5
89  #define SI_7021_MEASURE_TEMPERATURE     0xE3
90  #define SI_7021_TIMEOUT                 1000 / portTICK_PERIOD_MS
91
92  // for wifi
93  #define PROJECT_ESP_WIFI_SSID      "georgedimas"
94  #define PROJECT_ESP_WIFI_PASS      "george123"
95  #define PROJECT_ESP_MAXIMUM_RETRY  5
96
97  #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA2_PSK
98
99  #define WIFI_CONNECTED_BIT BIT0
100 #define WIFI_FAIL_BIT      BIT1
101
102 static const char *TAG = "max31865_ads1115_test_mqtt";
103
104 #define QUEUE_SIZE 10
105
106 // Define the queue handle globally
107 QueueHandle_t sensorDataQueue;
108
109 struct SensorData {
110     float temperature_out;
```

```
111         float temperature_in;
112         float humidity_in;
113         float temperature_liquid;
114         float pressure_liquid;
115         bool pumpIsRunning;
116         int heatingPower;
117     };
118
119     spi_device_handle_t      _spi;
120     int i2c_master_port_0 = I2C_MASTER_NUM_0;
121     int i2c_master_port_1 = I2C_MASTER_NUM_1;
122
123
124     static int s_retry_num = 0;
125     char unique_id_str[18]; // Global variable to hold the MAC address string
126     esp_mqtt_client_handle_t client; // Assuming mqtt client is globally
        available
127
128     /* FreeRTOS event group to signal when we are connected*/
129     static EventGroupHandle_t s_wifi_event_group;
130
131     //C functions
132     extern "C" void app_main();
133     //wifi functions
134     extern "C" void log_error_if_nonzero();
135     extern "C" void wifi_event_handler();
136     extern "C" void wifi_init_sta();
137     extern "C" void mqtt_event_handler();
138     extern "C" void mqtt_app_start();
139     extern "C" void publish_json_data();
140
141
142     //static
143     void log_error_if_nonzero(const char *message, int error_code)
144     {
145         if (error_code != 0) {
146             ESP_LOGE(TAG, "Last error %s: 0x%x", message, error_code);
147         }
148     }
149
150     void wifi_event_handler(void* arg, esp_event_base_t event_base, int32_t
        event_id, void* event_data)
151     {
152         if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
153             esp_wifi_connect();
154         } else if (event_base == WIFI_EVENT && event_id ==
            WIFI_EVENT_STA_DISCONNECTED) {
155             if (s_retry_num < PROJECT_ESP_MAXIMUM_RETRY) {
156                 esp_wifi_connect();
157                 s_retry_num++;
158                 ESP_LOGI(TAG, "retry to connect to the AP");
159             } else {
160                 xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
161             }
162             ESP_LOGI(TAG,"connect to the AP fail");
163         } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
164             ip_event_got_ip_t* event = (ip_event_got_ip_t*) event_data;
165             ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
166             s_retry_num = 0;
```

```
167             xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
168         }
169     }
170
171     void wifi_init_sta(void)
172     {
173         s_wifi_event_group = xEventGroupCreate();
174
175         ESP_ERROR_CHECK(esp_netif_init());
176
177         ESP_ERROR_CHECK(esp_event_loop_create_default());
178         esp_netif_create_default_wifi_sta();
179
180         wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
181         ESP_ERROR_CHECK(esp_wifi_init(&cfg));
182
183         esp_event_handler_instance_t instance_any_id;
184         esp_event_handler_instance_t instance_got_ip;
185         ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
            ESP_EVENT_ANY_ID, &wifi_event_handler, NULL, &instance_any_id));
186         ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT,
            IP_EVENT_STA_GOT_IP, &wifi_event_handler, NULL, &instance_got_ip));
187
188         wifi_config_t wifi_config = {
189             .sta = {
190                 .ssid = PROJECT_ESP_WIFI_SSID,
191                 .password = PROJECT_ESP_WIFI_PASS,
192                 .scan_method = WIFI_FAST_SCAN,           // or
                    WIFI_ALL_CHANNEL_SCAN
193                 .bssid_set = 0,
194                 .bssid = {0},
195                 .channel = 0,
196                 .listen_interval = 0,
197                 .sort_method = WIFI_CONNECT_AP_BY_SIGNAL, // or other
                    appropriate value
198                 .threshold= {
199                     .rssi = 0,
200                     .authmode = ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD,
201                 },
202                 .pmf_cfg = {
203                     .capable = true,
204                     .required = false
205                 },
206                 .rm_enabled = false,
207                 .btm_enabled = false,
208                 .mbo_enabled = false,
209                 .ft_enabled = false,
210                 .owe_enabled = false,
211                 .reserved = {0},
212                 .sae_pwe_h2e = WPA3_SAE_PWE_BOTH,
213             },
214         };
215
216         ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA) );
217         ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config) );
218         ESP_ERROR_CHECK(esp_wifi_start() );
219
220         ESP_LOGI(TAG, "wifi_init_sta finished.");
221
```

```
222       // Waiting until either the connection is established
          (WIFI_CONNECTED_BIT) or connection failed for the maximum
223       // number of re-tries (WIFI_FAIL_BIT). The bits are set by
          event_handler() (see above)
224       EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
225               WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
226               pdFALSE,
227               pdFALSE,
228               portMAX_DELAY);
229
230       // xEventGroupWaitBits() returns the bits before the call returned,
          hence we can test which event actually
231       // happened.
232       if (bits & WIFI_CONNECTED_BIT) {
233           ESP_LOGI(TAG, "connected to ap SSID:%s password:%s",
234                    PROJECT_ESP_WIFI_SSID, PROJECT_ESP_WIFI_PASS);
235       } else if (bits & WIFI_FAIL_BIT) {
236           ESP_LOGI(TAG, "Failed to connect to SSID:%s, password:%s",
237                    PROJECT_ESP_WIFI_SSID, PROJECT_ESP_WIFI_PASS);
238       } else {
239           ESP_LOGE(TAG, "UNEXPECTED EVENT");
240       }
241   }
242
243    // @brief Event handler registered to receive MQTT events
244    //
245    //  This function is called by the MQTT client event loop.
246    //
247    // @param handler_args user data registered to the event.
248    // @param base Event base for the handler(always MQTT Base in this example).
249    // @param event_id The id for the received event.
250    // @param event_data The data for the event, esp_mqtt_event_handle_t.
251
252   //static
253   void mqtt_event_handler(void *handler_args, esp_event_base_t base, int32_t
      event_id, void *event_data)
254   {
255       ESP_LOGD(TAG, "Event dispatched from event loop base=%s, event_id=%d",
          base, event_id);
256       //esp_mqtt_event_handle_t event = event_data;
257       esp_mqtt_event_handle_t event = reinterpret_cast<esp_mqtt_event_handle_t
          >(event_data);
258       client = event->client;
259       int msg_id;
260       // Prepare the full topic string
261       char full_topic[100]; // Adjust size as necessary
262       switch ((esp_mqtt_event_id_t)event_id) {
263       case MQTT_EVENT_CONNECTED:
264           ESP_LOGI(TAG, "MQTT_EVENT_CONNECTED");
265           // Prepare the full topic string
266           snprintf(full_topic, sizeof(full_topic), "id222017029/iot/%s/import"
              , unique_id_str);
267           msg_id = esp_mqtt_client_subscribe(client, full_topic, 0);
268           ESP_LOGI(TAG, "sent subscribe successful, msg_id=%d", msg_id);
269
270           break;
271       case MQTT_EVENT_DISCONNECTED:
272           ESP_LOGI(TAG, "MQTT_EVENT_DISCONNECTED");
273           break;
```

```
274
275         case MQTT_EVENT_SUBSCRIBED:
276             ESP_LOGI(TAG, "MQTT_EVENT_SUBSCRIBED, msg_id=%d", event->msg_id);
277
278             // Prepare the full topic string
279
280             snprintf(full_topic, sizeof(full_topic), "id222017029/iot/%s/report"
                , unique_id_str);
281             msg_id = esp_mqtt_client_publish(client, full_topic, "justboot", 0,
                0, 0);
282             ESP_LOGI(TAG, "sent publish successful, msg_id=%d", msg_id);
283             break;
284         case MQTT_EVENT_UNSUBSCRIBED:
285             ESP_LOGI(TAG, "MQTT_EVENT_UNSUBSCRIBED, msg_id=%d", event->msg_id);
286             break;
287         case MQTT_EVENT_PUBLISHED:
288             ESP_LOGI(TAG, "MQTT_EVENT_PUBLISHED, msg_id=%d", event->msg_id);
289             break;
290         case MQTT_EVENT_DATA:
291             ESP_LOGI(TAG, "MQTT_EVENT_DATA");
292             printf("TOPIC=%.*s\r\n", event->topic_len, event->topic);
293             printf("DATA=%.*s\r\n", event->data_len, event->data);
294             break;
295         case MQTT_EVENT_ERROR:
296             ESP_LOGI(TAG, "MQTT_EVENT_ERROR");
297             if (event->error_handle->error_type == MQTT_ERROR_TYPE_TCP_TRANSPORT
                ) {
298                 log_error_if_nonzero("reported from esp-tls", event->
                    error_handle->esp_tls_last_esp_err);
299                 log_error_if_nonzero("reported from tls stack", event->
                    error_handle->esp_tls_stack_err);
300                 log_error_if_nonzero("captured as transport's socket errno",
                    event->error_handle->esp_transport_sock_errno);
301                 ESP_LOGI(TAG, "Last errno string (%s)", strerror(event->
                    error_handle->esp_transport_sock_errno));
302
303             }
304             break;
305         default:
306             ESP_LOGI(TAG, "Other event id:%d", event->event_id);
307             break;
308         }
309     }
310
311     //static
312      void mqtt_app_start(void)
313     {
314         // Prepare the full client id
315         char full_client_id[100]; // Adjust size as necessary
316         snprintf(full_client_id, sizeof(full_client_id), "ESP32-S3-%s",
            unique_id_str);
317         const esp_mqtt_client_config_t mqtt_cfg = {
318             .broker = {
319                 .address = {
320                     .uri = "mqtt://79.129.0.122",    // MQTT broker URI
321                     .port = 1883,                    // MQTT broker port
322                 },
323             },
324             .credentials = {
```

```
325                 .username = "thesisuser",           // Username for MQTT
326                 .client_id = full_client_id,        // Client ID.
327                 .authentication = {
328                     .password = "thesis",         // Password for MQTT
329                 },
330             },
331         };
332
333         client = esp_mqtt_client_init(&mqtt_cfg);
334         // The last argument may be used to pass data to the event handler, in
        this example mqtt_event_handler
335         //esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID,
        mqtt_event_handler, NULL);
336         esp_mqtt_client_register_event(client, static_cast<esp_mqtt_event_id_t>(
        ESP_EVENT_ANY_ID), mqtt_event_handler, NULL);
337         esp_mqtt_client_start(client);
338
339     }
340
341     void publish_json_data(void *params) {
342         while (1) {
343             // Prepare the full topic string
344             char full_topic[100]; // Adjust size as necessary
345             snprintf(full_topic, sizeof(full_topic), "id222017029/iot/%s/export"
            , unique_id_str);
346
347
348             // JSON data
349             float temperature = 15.11;
350             float humidity = 55.61;
351             float pressure = 989.7;
352
353             // Prepare JSON string
354             char json_data[128]; // Adjust size as necessary
355             snprintf(json_data, sizeof(json_data), "{\"temperature\": %.2f,
            \"humidity\": %.2f, \"pressure\": %.1f}", temperature, humidity,
            pressure);
356
357             // Publish JSON data
358             int msg_id = esp_mqtt_client_publish(client, full_topic, json_data,
            0, 0, 0);
359             ESP_LOGI(TAG, "JSON data published of json, msg_id=%d", msg_id);
360
361             // Wait for 10 seconds
362             vTaskDelay(pdMS_TO_TICKS(10000)); // 10000 ms delay
363         }
364     }
365     // ... end of C function declarations ...
366
367     //spi functions
368     int initializeSPI(int mosi, int miso, int clk, int cs) {
369         esp_err_t ret;
370
371         spi_bus_config_t buscfg ;
372         memset( &buscfg, 0, sizeof(spi_bus_config_t) );
373
374         buscfg.mosi_io_num = mosi;
375         buscfg.miso_io_num = miso;
376         buscfg.sclk_io_num = clk;
```

```
377         buscfg.quadwp_io_num = -1;
378         buscfg.quadhd_io_num = -1;
379         buscfg.max_transfer_sz = 0; // Adjust as needed
380         buscfg.flags = SPICOMMON_BUSFLAG_MASTER;
381         buscfg.intr_flags = 0;
382 
383 
384         spi_device_interface_config_t devcfg;
385         memset( &devcfg, 0, sizeof(spi_device_interface_config_t) );
386 
387         devcfg.clock_speed_hz = 100000; // Clock speed in Hz
388         devcfg.mode = 1;
389         devcfg.spics_io_num = cs; // Set to your CS pin if used, otherwise -1
390         devcfg.queue_size = 1;    // Transaction queue size
391         devcfg.flags = SPI_DEVICE_HALFDUPLEX;
392         devcfg.address_bits = 8;
393         // Add other fields if needed, in the order they are declared in the
            struct
394 
395 
396         // Initialize the SPI bus
397         ret = spi_bus_initialize(SPI2_HOST, &buscfg, SPI_DMA_CH_AUTO);
398         ESP_ERROR_CHECK(ret);
399 
400         if (ret != ESP_OK) {
401             return 1;
402         }
403 
404         // Add the SPI device to the bus
405         ret = spi_bus_add_device(SPI2_HOST, &devcfg, &_spi);
406         ESP_ERROR_CHECK(ret);
407 
408         return 0;
409     }
410 
411     //i2c functions
412     static esp_err_t i2c_master_init(void)
413     {
414         i2c_config_t conf_0;
415         conf_0.mode = I2C_MODE_MASTER;
416         conf_0.sda_io_num = I2C_MASTER_SDA_IO_0;
417         conf_0.sda_pullup_en = GPIO_PULLUP_ENABLE;
418         conf_0.scl_io_num = I2C_MASTER_SCL_IO_0;
419         conf_0.scl_pullup_en = GPIO_PULLUP_ENABLE;
420         conf_0.master.clk_speed = 100000;
421         conf_0.clk_flags = 0; // for ESP32, this is optional
422 
423         esp_err_t ret = i2c_param_config(i2c_master_port_0, &conf_0);
424         if (ret != ESP_OK) {
425             ESP_LOGE(TAG, "Failed Parameter configuration_0: ",esp_err_to_name(
                ret),ret);
426             return ret;
427         }
428 
429         ret = i2c_driver_install(i2c_master_port_0, conf_0.mode, 0, 0, 0);
430         if (ret != ESP_OK) {
431             ESP_LOGE(TAG, "Failed Install driver for i2c_0 bus: %s",
                esp_err_to_name(ret),ret);
432             return ret;
```

```
433         }
434
435         ESP_LOGI(TAG, "Successful I2C_0 initialization");
436         printf("Successful I2C_0 initialization");
437
438
439         i2c_config_t conf_1;
440         conf_1.mode = I2C_MODE_MASTER;
441         conf_1.sda_io_num = I2C_MASTER_SDA_IO_1;
442         conf_1.sda_pullup_en = GPIO_PULLUP_ENABLE;
443         conf_1.scl_io_num = I2C_MASTER_SCL_IO_1;
444         conf_1.scl_pullup_en = GPIO_PULLUP_ENABLE;
445         conf_1.master.clk_speed = 100000;
446         conf_1.clk_flags = 0; // for ESP32, this is optional
447
448         //esp_err_t
449         ret = i2c_param_config(i2c_master_port_1, &conf_1);
450         if (ret != ESP_OK) {
451             ESP_LOGE(TAG, "Failed Parameter configuration_1: ",esp_err_to_name(
                ret),ret);
452             return ret;
453         }
454
455         ret = i2c_driver_install(i2c_master_port_1, conf_1.mode, 0, 0, 0);
456         if (ret != ESP_OK) {
457             ESP_LOGE(TAG, "Failed Install driver for i2c_1 bus: %s",
                esp_err_to_name(ret),ret);
458             return ret;
459         }
460
461         ESP_LOGI(TAG, "Successful I2C_1 initialization");
462         printf("Successful I2C_1 initialization");
463
464         return ret;
465     }
466
467     int a = initializeSPI( PIN_NUM_MOSI, PIN_NUM_MISO, PIN_NUM_CLK, PIN_NUM_CS );
468
469     //globally known
470     Adafruit_ADS1115 adc;
471     Adafruit_MAX31865 tempSensor ( _spi );
472
473     SensorData data_measure;
474
475     void sensor7021Task(){
476         printf("Hello!\n");
477         // Write measure humidity command
478         i2c_cmd_handle_t handle = i2c_cmd_link_create();
479         i2c_master_start(handle);
480
481         i2c_master_write_byte(handle,
482                     SI_7021_ADDRESS << 1 | I2C_MASTER_WRITE,
483                     I2C_MASTER_ACK);
484         i2c_master_write_byte(handle,
485                 SI_7021_MEASURE_HUMIDITY,
486                 I2C_MASTER_ACK);
487
488         i2c_master_stop(handle);
489         esp_err_t error = i2c_master_cmd_begin(i2c_master_port_1, handle,
```

```
        SI_7021_TIMEOUT);
490     i2c_cmd_link_delete(handle);
491
492     if (error != ESP_OK) {
493         ESP_LOGI(TAG, "Failed to write humidity command: %s",
            esp_err_to_name(error));
494     }
495
496     // Read two bytes from humidity sensor
497     uint8_t humMSB;
498     uint8_t humLSB;
499
500     handle = i2c_cmd_link_create();
501     i2c_master_start(handle);
502
503     i2c_master_write_byte(handle,
504                 SI_7021_ADDRESS << 1 | I2C_MASTER_READ,
505                 I2C_MASTER_ACK);
506
507     i2c_master_read_byte(handle, &humMSB, I2C_MASTER_ACK);
508     i2c_master_read_byte(handle, &humLSB, I2C_MASTER_NACK);
509
510     i2c_master_stop(handle);
511     error = i2c_master_cmd_begin(i2c_master_port_1, handle, SI_7021_TIMEOUT);
512     i2c_cmd_link_delete(handle);
513
514     if (error != ESP_OK) {
515         ESP_LOGI(TAG, "Failed to read humidity: %s", esp_err_to_name(error));
516     }
517
518     double humidity = ((uint16_t) humMSB << 8) | (uint16_t) humLSB;
519     humidity *= 125;
520     humidity /= 65536;
521     humidity -= 6; \
522
523     data_measure.humidity_in = humidity;
524
525     //ESP_LOGI(TAG, "humidity: %f", humidity);
526     //printf( "humidity: %f", humidity);
527
528
529
530     // Write measure temperature command
531     handle = i2c_cmd_link_create();
532     i2c_master_start(handle);
533
534     i2c_master_write_byte(handle,
535                 SI_7021_ADDRESS << 1 | I2C_MASTER_WRITE,
536                 I2C_MASTER_ACK);
537     i2c_master_write_byte(handle,
538             SI_7021_MEASURE_TEMPERATURE,
539             I2C_MASTER_ACK);
540
541     i2c_master_stop(handle);
542     error = i2c_master_cmd_begin(i2c_master_port_1, handle, SI_7021_TIMEOUT);
543     i2c_cmd_link_delete(handle);
544
545     if (error != ESP_OK) {
546         ESP_LOGI(TAG, "Failed to write temperature command: %s",
```

```
            esp_err_to_name(error));
547     }
548
549     // Read two bytes from the temperature and humidity sensor
550     uint8_t tempMSB;
551     uint8_t tempLSB;
552
553     handle = i2c_cmd_link_create();
554     i2c_master_start(handle);
555
556     i2c_master_write_byte(handle,
557                 SI_7021_ADDRESS << 1 | I2C_MASTER_READ,
558                 I2C_MASTER_ACK);
559
560     i2c_master_read_byte(handle, &tempMSB, I2C_MASTER_ACK);
561     i2c_master_read_byte(handle, &tempLSB, I2C_MASTER_NACK);
562
563     i2c_master_stop(handle);
564     error = i2c_master_cmd_begin(i2c_master_port_1, handle, SI_7021_TIMEOUT);
565     i2c_cmd_link_delete(handle);
566
567     if (error != ESP_OK) {
568         ESP_LOGI(TAG, "Failed to read Temperature: %s", esp_err_to_name(
            error));
569     }
570
571     double temperature = ((uint16_t) tempMSB << 8) | (uint16_t) tempLSB;
572     temperature *= 175.72;
573     temperature /= 65536;
574     temperature -= 46.85;
575
576     data_measure.temperature_in = temperature;
577
578     //SECOND SENSOR FOR EXTERNAL TEMP
579
580     // Write measure temperature command
581     i2c_cmd_handle_t handle_1 = i2c_cmd_link_create();
582     i2c_master_start(handle_1);
583
584     i2c_master_write_byte(handle_1,
585                 SI_7021_ADDRESS << 1 | I2C_MASTER_WRITE,
586                 I2C_MASTER_ACK);
587     i2c_master_write_byte(handle_1,
588             SI_7021_MEASURE_TEMPERATURE,
589             I2C_MASTER_ACK);
590
591     i2c_master_stop(handle_1);
592     error = i2c_master_cmd_begin(i2c_master_port_0, handle_1,
        SI_7021_TIMEOUT);
593     i2c_cmd_link_delete(handle_1);
594
595     if (error != ESP_OK) {
596         ESP_LOGI(TAG, "Failed to write temperature command: %s",
            esp_err_to_name(error));
597     }
598
599     // Read two bytes from the temperature sensor
600     uint8_t temp_outMSB;
601     uint8_t temp_outLSB;
```

```
602
603         handle_1 = i2c_cmd_link_create();
604         i2c_master_start(handle_1);
605
606         i2c_master_write_byte(handle_1,
607                               SI_7021_ADDRESS << 1 | I2C_MASTER_READ,
608                               I2C_MASTER_ACK);
609
610         i2c_master_read_byte(handle_1, &temp_outMSB, I2C_MASTER_ACK);
611         i2c_master_read_byte(handle_1, &temp_outLSB, I2C_MASTER_NACK);
612
613         i2c_master_stop(handle_1);
614         error = i2c_master_cmd_begin(i2c_master_port_0, handle_1,
            SI_7021_TIMEOUT);
615         i2c_cmd_link_delete(handle_1);
616
617         if (error != ESP_OK) {
618             ESP_LOGI(TAG, "Failed to read Temperature: %s", esp_err_to_name(
                error));
619         }
620
621         double temperature_out = ((uint16_t) temp_outMSB << 8) | (uint16_t)
            temp_outLSB;
622         temperature_out *= 175.72;
623         temperature_out /= 65536;
624         temperature_out -= 46.85;
625
626
627         ESP_LOGI(TAG, "temperature_out: %f", temperature_out);
628         printf( "temperature_out: %f", temperature_out);
629         if (temperature_out>20){
630             data_measure.temperature_out = temperature_out;
631         }
632     }
633
634     //collect the data from sensors;
635     void sensorTask(void *params) {
636         //QueueHandle_t sensorDataQueue = (QueueHandle_t)params;
637         printf("Start Sensor Task");
638
639         while (true) {
640             // ... Code to read from sensors
641             printf("Start Measuring");
642
643             //I2C data
644             uint16_t adcval = adc.readADC_SingleEnded(0);
645             float volts = adc.computeVolts(adcval);
646
647             if (adcval>0 && adcval<32767){
648                 float current = volts/0.150;
649                 float pressure = (current-4.0)/(20.0-4.0)*1.600;
650
651                 data_measure.pressure_liquid = pressure;
652
653                 printf( "Sensor ADC: %d, Voltage: %5.4f, Current: %5.3f mA,
                    Pressure: %4.3f bar\n", adcval, volts, current, pressure);
654             } else {
655                 printf( "The I2C bus is broken");
656                 i2c_master_init();
```

```
657             }
658
659             //SPI data
660             uint16_t rtd = tempSensor.readRTD();
661             // Check and print any faults
662             uint8_t fault = tempSensor.readFault();
663             if (fault) {
664                 printf("Fault 0x%X\n", fault);
665                 if (fault & MAX31865_FAULT_HIGHTHRESH) {
666                     printf("RTD High Threshold\n");
667                 }
668                 if (fault & MAX31865_FAULT_LOWTHRESH) {
669                     printf("RTD Low Threshold\n");
670                 }
671                 if (fault & MAX31865_FAULT_REFINLOW) {
672                     printf("REFIN- > 0.85 x Bias\n");
673                 }
674                 if (fault & MAX31865_FAULT_REFINHIGH) {
675                     printf("REFIN- < 0.85 x Bias - FORCE- open\n");
676                 }
677                 if (fault & MAX31865_FAULT_RTDINLOW) {
678                     printf("RTDIN- < 0.85 x Bias - FORCE- open\n");
679                 }
680                 if (fault & MAX31865_FAULT_OVUV) {
681                     printf("Under/Over voltage\n");
682                 }
683                 tempSensor.clearFault();
684             }
685
686             float tempMAX31865 = tempSensor.temperature(RNOMINAL, RREF);
687
688             if (rtd!=0 && rtd!=32767){
689                 printf( "Sensor: %d Temp: %12.4f Ratio: %12.4f\n", rtd,
                    tempMAX31865, rtd / 32768.0 * RREF );
690             } else {
691                 printf( "The spi slave is broken");
692             }
693
694             data_measure.temperature_liquid = tempMAX31865;
695
696             sensor7021Task();
697
698             printf("\nItem data: \nTemperature Liquid: %f, \nPressure Liquid:
                %f, \nRoom Temperature : %f, \nExternal Temperature : %f, \nRoom
                Humidity: %f\n\n",
699               data_measure.temperature_liquid,data_measure.pressure_liquid,
                  data_measure.temperature_in, data_measure.temperature_out,
                  data_measure.humidity_in);
700
701
702
703             vTaskDelay(1000 / portTICK_PERIOD_MS); // Sensor read interval
704
705         }
706     }
707
708     //send the data with mqtt
709     void mqttTask(void *params) {
710         while (true){
```

```
711
712             char full_topic[100]; // Adjust size as necessary
713             snprintf(full_topic, sizeof(full_topic), "id222017029/iot/%s/export"
                , unique_id_str);
714             if (data_measure.temperature_liquid>1){
715                 // Prepare JSON string
716                 char json_data[255]; // Adjust size as necessary
717                 snprintf(json_data, sizeof(json_data),
718                 "{\"temperature_out\": %.2f, \"temperature_in\": %.2f,
                    \"humidity_in\": %.2f, \"temperature_liquid\": %.2f,
                    \"pressure_liquid\": %.5f, \"heatingPower\": %d,
                    \"pumpIsRunning\": %d}",
719                 data_measure.temperature_out,
720                 data_measure.temperature_in,
721                 data_measure.humidity_in,
722                 data_measure.temperature_liquid,
723                 data_measure.pressure_liquid,
724                 data_measure.heatingPower,
725                 data_measure.pumpIsRunning ? 1 : 0 ); // Converts boolean to
                    "true" or "false"
726
727                 printf(json_data);
728
729                 // Publish JSON data
730                 int msg_id = esp_mqtt_client_publish(client, full_topic,
                    json_data, 0, 0, 0);
731                 ESP_LOGI(TAG, "JSON v1 data published of json, msg_id=%d",
                    msg_id);
732             }
733             vTaskDelay(5000 / portTICK_PERIOD_MS); // Sensor read interval
734         }
735     } //collect the data from sensors;
736
737
738     void controlPump(void *pvParameters) {
739         const int hyst = 10; // Hysteresis of 10 degrees
740         const float temp_in_threshold = 70; // 70 degrees threshold for room
            temperature
741
742         while (1) {
743             float temp_in = data_measure.temperature_in;       // Get actual
                sensor readings
744
745             ESP_LOGI(TAG, "Clock time: %u", xTaskGetTickCount());
746
747             // Check if temperatures are below the threshold minus hysteresis
                to turn the pump ON
748             if (temp_in < (temp_in_threshold - hyst) ) {
749                 if (var_wait_1 > TEMP_CHECK_PERIOD_MS) {
750                     vTaskDelay(TEMP_CHECK_PERIOD_MS / portTICK_PERIOD_MS);
751                     var_wait_1 -= TEMP_CHECK_PERIOD_MS;
752                     ESP_LOGI(TAG, "Pump wait for 5s and the timer var_wait_1 =
                        %d ms", var_wait_1);
753                 } else if (!data_measure.pumpIsRunning) {
754                     var_timer_1 = xTaskGetTickCount(); // Record the start time
755                     gpio_set_level(RELAY_PIN, 1);      // Turn pump ON
756                     data_measure.pumpIsRunning = true;
757                     ESP_LOGI(TAG, "Pump turned ON");
758                 } else {
```

```
759                 vTaskDelay(TEMP_CHECK_PERIOD_MS / portTICK_PERIOD_MS);
760                 uint32_t var_wait_2 = (xTaskGetTickCount() - var_timer_1) *
                    portTICK_PERIOD_MS;
761                 if (var_wait_2 > MAX_RUN_TIME_MS) {
762                     gpio_set_level(RELAY_PIN, 0); // Turn pump OFF
763                     data_measure.pumpIsRunning = false;
764                     var_wait_1 = var_wait_2;
765                     ESP_LOGI(TAG, "Pump turned OFF. Rest for %d ms",
                        var_wait_1);
766                 } else {
767                     ESP_LOGI(TAG, "Pump is running. Timer is %d ms",
                        var_wait_2);
768                 }
769             }
770         } else {
771             // Check if temperatures exceed the threshold to turn the pump
                OFF
772             if (data_measure.pumpIsRunning && (temp_in >= temp_in_threshold
                )) {
773                 uint32_t runtime = (xTaskGetTickCount() - var_timer_1) *
                    portTICK_PERIOD_MS;
774                 var_wait_1 = runtime; // Record how long the pump was running
775                 gpio_set_level(RELAY_PIN, 0); // Turn pump OFF
776                 data_measure.pumpIsRunning = false;
777                 ESP_LOGI(TAG, "Pump turned OFF");
778             } else {
779                 vTaskDelay(TEMP_CHECK_PERIOD_MS / portTICK_PERIOD_MS);
780                 if (var_wait_1 > TEMP_CHECK_PERIOD_MS) {
781                     var_wait_1 -= TEMP_CHECK_PERIOD_MS;
782                     ESP_LOGI(TAG, "HOT! Pump wait for 5s and the timer
                        var_wait_1 = %d s", var_wait_1);
783                 }
784             }
785         }
786         if (var_wait_1>900000){
787             var_wait_1 = MAX_RUN_TIME_MS;
788             ESP_LOGI(TAG, "OOhps reset");
789         }
790     }
791 }
792
793 void controlHeat(void *pvParameters) {
794     const int hyst = 5; // Hysteresis of 10 degrees
795     const float temp_liquid_threshold = 95; // 70 degrees threshold for
        room temperature
796     int heatPower = 0;
797
798     ESP_LOGI(TAG, "Control the heaaat!");
799
800     while (1) {
801         float temp_liquid = data_measure.temperature_liquid; // Get actual
            sensor readings
802
803         ESP_LOGI(TAG, "Heating Clock time: %u", xTaskGetTickCount());
804
805         // Check if temperatures are below the threshold minus hysteresis
            to turn the pump ON
806         if (temp_liquid < (temp_liquid_threshold - hyst) ) {
807                 if (data_measure.pumpIsRunning){
```

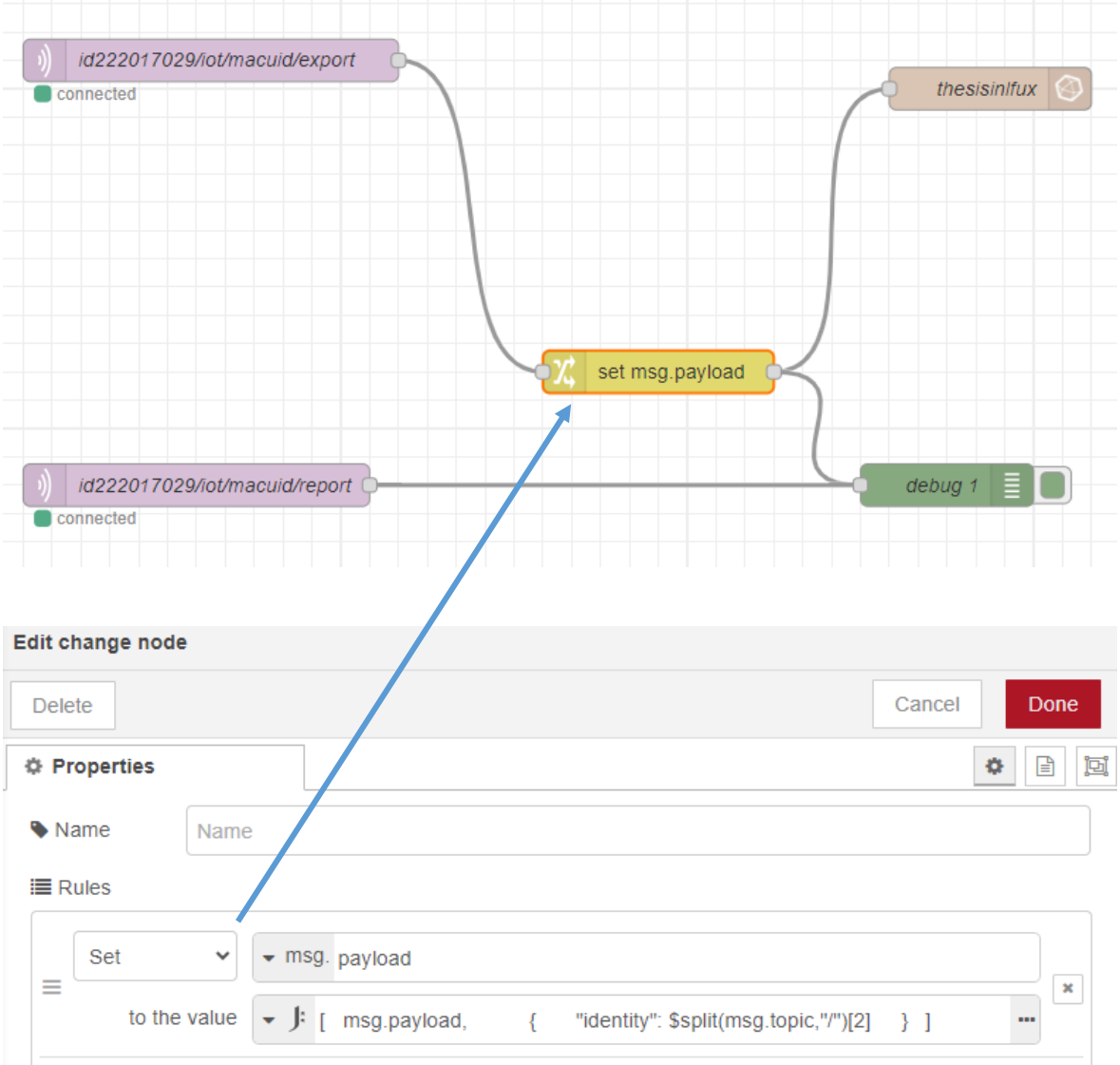
```
808                     heatPower = 120;
809                 } else {
810                     heatPower = 80;
811                 }
812             } else {
813                 // Check if temperatures exceed the threshold to turn the pump
                    OFF
814                 if (data_measure.heatingPower>0 && (temp_liquid >=
                    temp_liquid_threshold)) {
815                     heatPower = 0;
816                 } else {
817                     if (temp_liquid < temp_liquid_threshold){
818                         if (data_measure.pumpIsRunning){
819                             heatPower = 120;
820                         } else {
821                             heatPower = 80;
822                         }
823                     }
824                 }
825             }
826             ESP_LOGI(TAG, "Heating Power is %d ",heatPower);
827 
828             if (heatPower == 160){
829                 gpio_set_level(RELAY_HEAT_PIN, 1);      // Turn Heating ON for
                    160W
830                 gpio_set_level(RELAY_HEAT2_PIN, 1);      // Turn Heating ON for
                    160W
831                 gpio_set_level(RELAY_HEAT3_PIN, 1);      // Turn Heating ON for
                    160W
832                 data_measure.heatingPower = 120;
833                 ESP_LOGI(TAG, "Heating Power = 120W ");
834             } else if (heatPower == 120){
835                 gpio_set_level(RELAY_HEAT_PIN, 1);      // Turn Heating ON for
                    120W
836                 gpio_set_level(RELAY_HEAT2_PIN, 1);      // Turn Heating ON for
                    120W
837                 gpio_set_level(RELAY_HEAT3_PIN, 0);      // Turn Heating ON for
                    120W
838                 data_measure.heatingPower = 120;
839                 ESP_LOGI(TAG, "Heating Power = 120W ");
840             } else if (heatPower == 80){
841                 gpio_set_level(RELAY_HEAT_PIN, 1);      // Turn Heating ON for
                    800W
842                 gpio_set_level(RELAY_HEAT2_PIN, 0);      // Turn Heating ON for
                    80W
843                 gpio_set_level(RELAY_HEAT3_PIN, 0);      // Turn Heating ON for
                    80W
844                 data_measure.heatingPower = 80;
845                 ESP_LOGI(TAG, "Heating Power = 80W ");
846             } else if (heatPower == 0){
847                 gpio_set_level(RELAY_HEAT_PIN, 0);      // Turn Heating OFF
848                 gpio_set_level(RELAY_HEAT2_PIN, 0);      // Turn Heating
                    OFF
849                 gpio_set_level(RELAY_HEAT3_PIN, 0);      // Turn Heating ON for
                    120W
850                 data_measure.heatingPower = 0;
851                 ESP_LOGI(TAG, "Heating Power = 0W ");
852             }
853             vTaskDelay(TEMP_CHECK_PERIOD_MS / portTICK_PERIOD_MS);
```

```
854         }
855     }
856
857
858     void app_main(void)
859     {
860         ESP_LOGI(TAG, "[APP] Startup..");
861         ESP_LOGI(TAG, "[APP] Free memory: %d bytes", esp_get_free_heap_size());
862         ESP_LOGI(TAG, "[APP] IDF version: %s", esp_get_idf_version());
863
864         esp_log_level_set("*", ESP_LOG_INFO);
865         esp_log_level_set("mqtt_client", ESP_LOG_VERBOSE);
866         esp_log_level_set("MQTT_EXAMPLE", ESP_LOG_VERBOSE);
867         esp_log_level_set("TRANSPORT_BASE", ESP_LOG_VERBOSE);
868         esp_log_level_set("esp-tls", ESP_LOG_VERBOSE);
869         esp_log_level_set("TRANSPORT", ESP_LOG_VERBOSE);
870         esp_log_level_set("outbox", ESP_LOG_VERBOSE);
871
872         //Initialize NVS
873         esp_err_t ret = nvs_flash_init();
874         if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
            ESP_ERR_NVS_NEW_VERSION_FOUND) {
875           ESP_ERROR_CHECK(nvs_flash_erase());
876           ret = nvs_flash_init();
877         }
878         ESP_ERROR_CHECK(ret);
879
880         ESP_LOGI(TAG, "ESP_WIFI_MODE_STA");
881         wifi_init_sta();
882
883         //delay to let the wifi be established correctly and avoid errors
884         vTaskDelay(1000 / portTICK_PERIOD_MS);
885
886         // Buffer to hold the unique ID
887         uint8_t unique_id[6]; // ESP32's MAC address is 6 bytes
888         char unique_id_str_temp[18]; // Each byte in hex is 2 chars + 5
            separators and 1 null terminator
889
890         // Fetch the unique ID (MAC address in this case)
891         esp_err_t ret2 = esp_efuse_mac_get_default(unique_id);
892         ESP_ERROR_CHECK(ret2); // Check if there was an error in fetching the
            MAC address
893
894         // Convert the unique ID to a string
895         snprintf(unique_id_str_temp, sizeof(unique_id_str_temp),
896                 "%02X-%02X-%02X-%02X-%02X-%02X",
897                 unique_id[0], unique_id[1], unique_id[2],
898                 unique_id[3], unique_id[4], unique_id[5]);
899
900         // Print the unique ID
901         printf("Unique ID: %s\n", unique_id_str_temp);
902
903         // Copy the MAC address string to the global variable
904         strncpy(unique_id_str, unique_id_str_temp, sizeof(unique_id_str));
905         unique_id_str[sizeof(unique_id_str) - 1] = '\0'; // Ensure
            null-termination
906
907         mqtt_app_start();
908
```

```
909         //wait another 100ms for the SPI and I2C
910         vTaskDelay(100 / portTICK_PERIOD_MS);
911 
912         // Initialize I2C devices
913         if (i2c_master_init() != ESP_OK) {
914             ESP_LOGE(TAG, "First I2C initialization failed!");
915             return;
916         }
917 
918         adc.begin( i2c_master_port_0 );
919         adc.setGain(GAIN_ONE);
920 
921         tempSensor.begin(MAX31865_2WIRE);
922         tempSensor.enable50Hz(true);
923 
924         gpio_reset_pin(RELAY_PIN);
925         gpio_set_direction(RELAY_PIN, GPIO_MODE_OUTPUT);
926         gpio_set_level(RELAY_PIN, 0); // Ensure the pump is OFF at start
927 
928         gpio_reset_pin(RELAY_HEAT_PIN);
929         gpio_reset_pin(RELAY_HEAT2_PIN);
930         gpio_reset_pin(RELAY_HEAT3_PIN);
931         gpio_set_direction(RELAY_HEAT_PIN, GPIO_MODE_OUTPUT);
932         gpio_set_direction(RELAY_HEAT2_PIN, GPIO_MODE_OUTPUT);
933         gpio_set_direction(RELAY_HEAT3_PIN, GPIO_MODE_OUTPUT);
934         gpio_set_level(RELAY_HEAT_PIN, 0); // Ensure the heat is OFF at start
935         gpio_set_level(RELAY_HEAT2_PIN, 0); // Ensure the heat is OFF at start
936         gpio_set_level(RELAY_HEAT3_PIN, 0); // Ensure the heat is OFF at start
937 
938 
939         // Create the queue
940         sensorDataQueue = xQueueCreate(QUEUE_SIZE, sizeof(SensorData*));
941 
942         if (sensorDataQueue == NULL) {
943             // Queue was not created and must not be used.
944             // Handle the error
945             printf("Queue was not created and must not be used");
946         }
947 
948         // Create the sensor task
949         xTaskCreate(sensorTask, "Sensor Task", 4096, NULL, 5, NULL);
950 
951         xTaskCreate(controlPump, "controlPump", 4096, NULL, 10, NULL);
952         xTaskCreate(controlHeat, "controlHeat", 4096, NULL, 10, NULL);
953 
954         // Create the MQTT task
955         xTaskCreate(mqttTask, "MQTT Task", 4096, NULL, 5, NULL);
956     }
```

Παράρτημα Β' – Υποδομή Node Red

Στο παρών παράρτημα επισυνάπτεται η παραμετροποίηση της υποδομής του Node-Red.



Παράρτημα Γ' – Κώδικας τεχνητής νοημοσύνης

Στο παρών παράρτημα επισυνάπτεται ο κώδικας της python από την πλευρά του διακομιστή.

Ακολουθεί ο κώδικας για την κατασκευή και εκπαίδευση του αλγορίθμου.

```
#    UNIVERSITY OF WEST ATTICA
#    Industrial Design and Production Engineering
#    Egaleo, September 2024
#    Author:   Georgios Dimas  /  Student Number: 222017029
#    Part of thesis diploma  "Data collection, processing, analysis and
#    management in a cooling/heating system"

#Create a function that builds the LSTM model. This function will take a hp
(hyperparameters) argument that Keras Tuner uses to suggest hyperparame

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import keras_tuner as kt
import tensorflow as tf


import pandas as pd
from influxdb_client import InfluxDBClient


#Step 1: Retrieve Data from InfluxDB

# InfluxDB settings
url = 'http://192.168.0.66:8086'
token =
'n1ygZY41QPQY0wm4e-3EJixnZbM48ljPiYU6KtG1PQbyH540MOhESQXWuDMBxrJYC7Pbi67BApiZ
DEq7q2nfpg=='
org = 'UNIWA'

# Create a client instance
client = InfluxDBClient(url=url, token=token, org=org)

# Define your Flux query
query = '''
from(bucket: "thesis1")
  |> range(start: 2024-07-20, stop: 2024-08-07)
  |> filter(fn: (r) => r["_measurement"] == "meas1_1")
  |> filter(fn: (r) => r["identity"] == "34-85-18-91-A8-74")
  |> filter(fn: (r) => r["_field"] == "heatingPower" or r["_field"] ==
  "humidity_in" or r["_field"] == "pressure_liquid" or r["_field"] ==
  "pumpIsRunning" or r["_field"] == "temperature_in" or r["_field"] ==
  "temperature_liquid" or r["_field"] == "temperature_out")
  |> aggregateWindow(every: 5s, fn: mean, createEmpty: false)
  |> yield(name: "mean")
'''

# Query the data
result = client.query_api().query(org=org, query=query)
# Close client connection
client.close()
```

```
#Step 2: Dictionary Setup for Features and Targets and transform influx datatable to pandas dataframe.
var_details = {
    "heatingPower": {"type": "analog", "feature": True, "target": False, "scale_min": 0, "scale_max": 160},
    "humidity_in": {"type": "analog", "feature": True, "target": True, "scale_min": 0, "scale_max": 100},
    "pressure_liquid": {"type": "analog", "feature": True, "target": True, "scale_min": 0.8, "scale_max": 2.0},
    "pumpIsRunning": {"type": "binary", "feature": True, "target": False},
    "temperature_in": {"type": "analog", "feature": True, "target": True, "scale_min": 0, "scale_max": 80},
    "temperature_liquid": {"type": "analog", "feature": True, "target": True, "scale_min": 0, "scale_max": 120},
    "temperature_out": {"type": "analog", "feature": True, "target": False, "scale_min": 0, "scale_max": 80}
}


import pandas as pd

# Initialize a list to hold dictionaries for each record
data = []

# Iterate over each table, then over each record in the table
for table in result:
    for record in table.records:
        data.append({
            'timestamp': record.get_time(),
            'value': record.get_value(),
            'field': record['_field']
        })

# Create DataFrame
df = pd.DataFrame(data)

# Pivot the DataFrame to make fields into columns
df = df.pivot(index='timestamp', columns='field', values='value')


print(df.head())  # Show the first few rows to check the DataFrame

#Step 3: Format DataFrame
# Assuming 'df' is your DataFrame name instead of 'result' for clarity

# Convert binary variables, ensuring 'pumpIsRunning' is treated as an integer
if df['pumpIsRunning'].isnull().any():
    df['pumpIsRunning'].fillna(0, inplace=True)  # Assuming 0 as the default for missing
df['pumpIsRunning'] = df['pumpIsRunning'].astype(int)

# Sort by index if not already sorted (good practice even if it seems sorted)
df.sort_index(inplace=True)

# Identify sessions
df['session_id'] = (df.index.to_series().diff() > pd.Timedelta('30s')).cumsum()

# Compute the count of each session_id
session_counts = df['session_id'].value_counts()

# Find session_ids with counts greater than or equal to 60
```

```
valid_sessions = session_counts[session_counts >= 70].index

# Filter the DataFrame to only include rows with valid session_ids
df_filtered = df[df['session_id'].isin(valid_sessions)]

# Assign new contiguous session IDs
df_filtered['session_id'], _ = pd.factorize(df_filtered['session_id'])

# Now your session_id should be remapped from 0 to n-1 where n is the
number of unique sessions
print(df_filtered['session_id'].head())

df_filtered.reset_index(drop=True, inplace=True)

df=df_filtered

# Display updated DataFrame information to confirm changes
print(df.info())


#Step 4: Scale the Data
import joblib
import pandas as pd
from sklearn.preprocessing import MinMaxScaler


# Define sensor limits
sensor_limits = {
    'heatingPower': (0, 160),
    'humidity_in': (0, 100),
    'pressure_liquid': (0.8, 2.0),
    'pumpIsRunning': (0, 1),
    'temperature_in': (0, 80),
    'temperature_liquid': (0, 120),
    'temperature_out': (0, 80)
}

# Initialize a dictionary to store the scalers
scalers = {}

# Loop through each sensor, create a scaler, fit it, and transform the data
for sensor, limits in sensor_limits.items():
    # Create a DataFrame with the minimum and maximum values
    limits_df = pd.DataFrame({
        sensor: [limits[0], limits[1]]
    })

    # Create a MinMaxScaler for the specific range
    scaler = MinMaxScaler(feature_range=(0, 1))

    # Fit the scaler on a DataFrame that includes the sensor name as a column
    scaler.fit(limits_df)

    # Apply the scaler to the corresponding column in the DataFrame
    df[[sensor]] = scaler.transform(df[[sensor]])

    # Store the scaler in the dictionary
    scalers[sensor] = scaler

    # Optionally, save each scaler to a file
    joblib.dump(scaler, f"{sensor}_scaler.save")

```

```
print("All features scaled individually. Scalers saved.")

#Step 5: Create Datasets with Lookback

import numpy as npdef create_dataset(X, y, look_back=60):
    Xs, ys = [], []
    for session_id, group in X.groupby(X['session_id']):
        session_X = group[features]
        session_y = y.loc[group.index][targets]
        for i in range(len(session_X) - look_back):
            v = session_X.iloc[i:(i + look_back)].values
            #if session_X.iloc[i:(i + look_back)]['session_id'].nunique()
            == 1:
            Xs.append(v)
            ys.append(session_y.iloc[i + look_back].values)
    return np.array(Xs), np.array(ys)

# Assuming your targets are also part of your DataFrame
features = [col for col, details in var_details.items() if details['feature'
]]
targets = [col for col, details in var_details.items() if details['target']]

features = [f for f in features if f != 'session_id']
targets = [t for t in targets if t != 'session_id']

X, y = create_dataset(df, df, look_back=60)


#Step 6: Split Data into Training and Testing Sets
# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

print(f"Training data shape: {X_train.shape}, {y_train.shape}")
print(f"Test data shape: {X_test.shape}, {y_test.shape}")


#Step 7: Build the LSTM Model
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

def build_model(input_shape):
    model = Sequential()
    model.add(LSTM(100, return_sequences=True, input_shape=input_shape))
    model.add(Dropout(0.2))
    model.add(LSTM(50, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(50))
    model.add(Dropout(0.2))
    model.add(Dense(len(targets))) #4
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

from kerastuner import HyperModel
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam

```

```
225
226 class LSTMHyperModel(HyperModel):
227     def __init__(self, input_shape, num_targets):
228         self.input_shape = input_shape
229         self.num_targets = num_targets
230
231     def build(self, hp):
232         model = Sequential()
233         model.add(LSTM(
234             units=hp.Int('units_1', min_value=32, max_value=512, step=32),
235             return_sequences=True,
236             input_shape=self.input_shape
237         ))
238         model.add(Dropout(rate=hp.Float('dropout_1', min_value=0, max_value=
        0.5, step=0.1)))
239
240
241         for i in range(hp.Int('num_layers', 1, 3)):  # Number of additional
        LSTM layers
242             model.add(LSTM(
243                 units=hp.Int(f'units_{i+2}', min_value=32, max_value=512,
                    step=32),
244                 return_sequences=i < hp.Int('num_layers', 1, 3) - 1
245             ))
246             model.add(Dropout(rate=hp.Float(f'dropout_{i+2}', min_value=0,
                max_value=0.5, step=0.1)))
247
248         model.add(Dense(self.num_targets))
249         model.compile(
250             optimizer=Adam(
251                 hp.Float('learning_rate', min_value=1e-4, max_value=1e-2,
                    sampling='LOG')
252             ),
253             loss='mean_squared_error'
254         )
255         return model
256
257
258 # Prompt the user for input
259 user_input= ""
260 while user_input!="load" and user_input!="train":
261     user_input = input("Type 'load' to load an existing model, or 'train'
    to train a new one: ").strip().lower()
262
263 model_path = 'lstm_model_meas5'
264
265
266 if user_input == 'load':
267     if os.path.exists(model_path):
268         model = load_model(model_path)
269         print("Model loaded successfully.")
270     else :
271         print("No saved model found. Training a new one.")
272         user_input = 'train'
273
274 if user_input == 'train':
275     # Train the model
276     # Assume X_train, y_train, X_test, y_test are already defined
277     from kerastuner.tuners import Hyperband
278
279     hypermodel = LSTMHyperModel(input_shape=(X_train.shape[1], X_train.shape
    [2]), num_targets=y_train.shape[1])
```

```
280
281         tuner = Hyperband(
282             hypermodel,
283             objective='val_loss',
284             max_epochs=10,
285             factor=3,
286             directory='keras_tuner_dir',
287             project_name='lstm_tuning'
288         )
289
290         # Early stopping to avoid overfitting
291         stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
            patience=5)
292
293         # Execute the search
294         tuner.search(X_train, y_train, epochs=50, validation_data=(X_test,
            y_test), callbacks=[stop_early])
295
296         # Get the best model
297         model = tuner.get_best_models(num_models=1)[0]
298
299         # Evaluate the best model
300         loss = model.evaluate(X_test, y_test)
301         print(f'Best model loss on test set: {loss}')
302
303         # Save the best model
304         model.save(model_path)
305         print(f"Model trained and saved at {model_path}")
306
307
308         # Step 8: Evaluate and Visualize the Model Performance
309         import matplotlib.pyplot as plt
310
311         # Plot training and validation loss
312         plt.plot(history.history['loss'], label='train')
313         plt.plot(history.history['val_loss'], label='test')
314         plt.title('Model Loss')
315         plt.ylabel('Loss')
316         plt.xlabel('Epoch')
317         plt.legend()
318         plt.show()
319         plt.savefig('model_loss.png')
320         plt.close()
321
322         print(f"Test loss: {test_loss}")
323
324     # Continue with predictions or further processing
325
326
327     #Step 9: Inverse Transform Predictions
328     # Make predictions
329     train_predict = model.predict(X_train)
330     test_predict = model.predict(X_test)
331
332
333
334     import joblib
335     import numpy as np
336
337     # Load all saved scalers
338     scalers2 = {col: joblib.load(f"{col}_scaler.save") for col in var_details if
        var_details[col]['type'] == 'analog' and var_details[col]['feature']}
```

```
339
340   # Initialize dictionaries to store the original data
341   test_predict_original = {}
342   train_predict_original = {}
343   y_test_original = {}
344   y_train_original = {}
345
346   # Assuming test_predict, train_predict, y_test, and y_train are your
      model's outputs and true labels
347   # Ensure these arrays are shaped (number_of_samples, number_of_features)
348
349   # Apply inverse transformation for each feature
350   for col in var_details:
351       if var_details[col]['type'] == 'analog' and var_details[col]['target']:
352           col_index = targets.index(col)  # Get the correct column index for
              the target
353           # Reshape data for inverse transformation
354           test_predict_col = test_predict[:, col_index].reshape(-1, 1)
355           train_predict_col = train_predict[:, col_index].reshape(-1, 1)
356           y_test_col = y_test[:, col_index].reshape(-1, 1)
357           y_train_col = y_train[:, col_index].reshape(-1, 1)
358           # Inverse scaling
359           test_predict_original[col] = scalers2[col].inverse_transform(
              test_predict_col)
360           train_predict_original[col] = scalers2[col].inverse_transform(
              train_predict_col)
361           y_test_original[col] = scalers2[col].inverse_transform(y_test_col)
362           y_train_original[col] = scalers2[col].inverse_transform(y_train_col)
363
364   # Print or use the original values as needed
365   print("Original Test Predictions:", test_predict_original)
366   print("Original Train Predictions:", train_predict_original)
367   print("Original Test Labels:", y_test_original)
368   print("Original Train Labels:", y_train_original)
369
370
```

Ακολουθεί ο κώδικας για την εφαρμογή της προγνωστικής συντήρησης

```
#    UNIVERSITY OF WEST ATTICA
#    Industrial Design and Production Engineering
#    Egaleo, September 2024
#    Author:   Georgios Dimas  /  Student Number: 222017029
#    Part of thesis diploma  "Data collection, processing, analysis and
#    management in a cooling/heating system"


import time
import numpy as np
import pandas as pd
import joblib
from influxdb_client import InfluxDBClient
from tensorflow.keras.models import load_model
from influxdb_client import InfluxDBClient, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

# InfluxDB settings
url = 'http://192.168.0.66:8086'  # Replace with your InfluxDB URL
token =
'n1ygZY41QPQY0wm4e-3EJixnZbM48ljPiYU6KtG1PQbyH540MOhESQXWuDMBxrJYC7Pbi67BApiZ
DEq7q2nfpg==' # Replace with your InfluxDB token
org = 'UNIWA'                # Replace with your organization name
bucket= 'thesis1'

#Step 2: Dictionary Setup for Features and Targets and transform influx
datatable to pandas dataframe.
var_details = {
    "heatingPower": {"type": "analog", "feature": True, "target": False,
    "scale_min": 0, "scale_max": 160},
    "humidity_in": {"type": "analog", "feature": True, "target": True,
    "scale_min": 0, "scale_max": 100},
    "pressure_liquid": {"type": "analog", "feature": True, "target": True,
    "scale_min": 0.8, "scale_max": 2.0},
    "pumpIsRunning": {"type": "binary", "feature": True, "target": False},
    "temperature_in": {"type": "analog", "feature": True, "target": True,
    "scale_min": 0, "scale_max": 80},
    "temperature_liquid": {"type": "analog", "feature": True, "target": True
    , "scale_min": 0, "scale_max": 120},
    "temperature_out": {"type": "analog", "feature": True, "target": False,
    "scale_min": 0, "scale_max": 80}
}

# Load the scaler and the model
model = load_model('lstm_model_meas5')
#scalers = joblib.load('scalers.joblib')  # Assuming all scalers are saved
in one joblib file
scalers = {col: joblib.load(f"{col}_scaler.save") for col in var_details if
var_details[col]['type'] == 'analog' and var_details[col]['feature']}

# Define anomaly thresholds for each target
anomaly_thresholds = {
    'temperature_in': 0.9,
    'temperature_liquid': 1.5,
    'humidity_in': 2.0,
    'pressure_liquid': 0.05
```

```
}

# Function to fetch the latest data from InfluxDB
def fetch_latest_data():
    # Create a client instance
    client = InfluxDBClient(url=url, token=token, org=org)
    query = '''from(bucket: "thesis1")
    |> range(start: -6m)
  |> filter(fn: (r) => r["_measurement"] == "meas1_1")
  |> filter(fn: (r) => r["identity"] == "34-85-18-91-A8-74")
  |> filter(fn: (r) => r["_field"] == "heatingPower" or r["_field"] == "humidity_in" or r["_field"] == "pressure_liquid" or r["_field"] == "pumpIsRunning" or r["_field"] == "temperature_in" or r["_field"] == "temperature_liquid" or r["_field"] == "temperature_out")
  |> aggregateWindow(every: 5s, fn: mean, createEmpty: false)
  |> limit(n:61)
  |> yield(name: "mean")
'''
    result = client.query_api().query(org='UNIWA', query=query)
    client.close()
    # Process the result into a DataFrame
    data = []

    # Iterate over each table, then over each record in the table
    for table in result:
        for record in table.records:
            data.append({
                'timestamp': record.get_time(),
                'value': record.get_value(),
                'field': record['_field']
            })

    # Create DataFrame
    df = pd.DataFrame(data)

    # Pivot the DataFrame to make fields into columns
    df = df.pivot(index='timestamp', columns='field', values='value')

    print(df.head())  # Show the first few rows to check the DataFrame

    #Step 3: Format DataFrame
    # Assuming 'df' is your DataFrame name instead of 'result' for clarity

    # Convert binary variables, ensuring 'pumpIsRunning' is treated as an integer
    if df['pumpIsRunning'].isnull().any():
        df['pumpIsRunning'].fillna(0, inplace=True)  # Assuming 0 as the default for missing
    df['pumpIsRunning'] = df['pumpIsRunning'].astype(int)

    # Sort by index if not already sorted (good practice even if it seems sorted)
    df.sort_index(inplace=True)
    if df.empty:
        return None

    # Scale the data
    for sensor in scalers:
        if sensor in df.columns:
            df[sensor] = scalers[sensor].transform(df[[sensor]])

    return df
```

```
102
103   # Function to preprocess data and predict
104   def predict_and_detect_anomaly(df):
105       # Ensure all columns are in the expected order and scale them
106
107       # Check we have enough data
108       if df.shape[0] < 61:
109           return ["Not enough data for prediction."]
110
111       # Reshape data for LSTM: [samples, time steps, features]
112       X = np.array([df.iloc[:-1].values])  # Last 60 for prediction
113
114       # Predict using the LSTM model
115       predictions = model.predict(X)[0]
116
117       # Inverse transform predictions and actual last value
118       targets = [col for col, details in var_details.items() if details[
          'target']]
119       targets2 = ['heatingPower']
120       # Assuming 'predictions' is an array with each element corresponding to
          a target
121       if len(predictions) == len(targets):  # Safety check
122           predictions_dict = {col: scalers[col].inverse_transform([[pred]])[0
              ][0] for col, pred in zip(targets, predictions)}
123       else:
124           print("Error: The number of predictions does not match the number
              of targets")
125
126       actual_dict = {col: scalers[col].inverse_transform([[df.iloc[-1][col
          ]]])[0][0] for col in targets if col in df.columns}
127       actual_dict2 = {col: scalers[col].inverse_transform([[df.iloc[-1][col
          ]]])[0][0] for col in targets2 if col in df.columns}
128
129       print("predictions_dict",predictions_dict)
130       print("actual_dict",actual_dict)
131
132       #write the predicted data to InfluxDB using the timestamp of the latest
          actual data
133       # Assuming df is your DataFrame containing the actual values and the
          timestamp index
134       latest_timestamp = df.index[-1]  # The latest timestamp from actual data
135
136
137       # Prepare the Data
138       measurement_name = "meas1_1_pred"
139       tags = {
140           "identity": "34-85-18-91-A8-74"
141       }
142       fields = predictions_dict  # Assuming predictions_dict contains your
          predicted data fields
143       # And 'predictions_dict' is your dictionary containing the predicted
          values for other variables
144       fields['heatingPower'] = actual_dict2['heatingPower']  # Add actual
          heating power to predictions
145       timestamp = int(latest_timestamp.timestamp()) * 1_000_000_000  #
          Nanoseconds
146
147       #Format the Data for InfluxDB
148       line_protocol = f"{measurement_name}"
149       if tags:
150           tag_str = "," + ",".join([f"{k}={v}" for k, v in tags.items()])
151           line_protocol += tag_str
```

```
152         if fields:
153             field_str = " " + ",".join([f"{k}={v}" for k, v in fields.items() if
                 isinstance(v, (int, float))])
154             field_str += ",".join([f'{k}="{v}"' for k, v in fields.items() if
                isinstance(v, str)])
155             line_protocol += field_str
156         line_protocol += f" {timestamp}"
157 
158         #Write to InfluxDB
159             # Initialize client for write
160         client_2 = InfluxDBClient(url=url, token=token, org=org)
161 
162         # Get the write API
163         write_api = client_2.write_api(write_options=SYNCHRONOUS)
164 
165         # Write data
166         # Assuming the client and write_api are already set up
167         write_api.write(bucket, org, line_protocol, write_precision=
            WritePrecision.NS)
168 
169 
170         # Close the client
171         client_2.close()
172 
173 
174         #predictions = {col: scalers[col].inverse_transform([[pred]])[0][0] for
            col, pred in zip(scalers.keys(), predictions)}
175         #actual = {col:
            scalers[col].inverse_transform([[df.iloc[-1][col]]])[0][0] for col in
            scalers.keys() if col in df.columns and var_details[col]['type'] ==
            'analog' and var_details[col]['target']}
176 
177         # Compare predictions to actual values and check for anomalies
178         anomalies = []
179         for col in actual_dict:
180             delta=abs(predictions_dict[col] - actual_dict[col])
181             if delta > anomaly_thresholds.get(col, float('inf')):
182                 anomalies.append(f"ALERT!!! {col}: Predicted={round(
                    predictions_dict[col],4)}, Actual={round(actual_dict[col],4)},
                    delta={round(delta,4)}")
183 
184         return anomalies if anomalies else ["No anomalies detected."]
185 
186     # Main monitoring loop
187     while True:
188         try:
189             df = fetch_latest_data()
190             anomalies = predict_and_detect_anomaly(df)
191             for anomalie in anomalies:
192                 print(anomalie)
193         except Exception as e:
194             print(f"An error occurred: {e}")
195         time.sleep(5)  # Check every 5 sec
196 
```