ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

“Ηλεκτρονικά ελεγχόμενος θάλαμος απόσταξης”

Του φοιτητή
Δημήτρη Ουζούνη
Αρ. Μητρώου: 52113m

Επιβλέπων
Αργύρης Χατζόπουλος
Επίκουρος καθηγητής

Σεπτέμβριος 2025

Τίτλος Δ.Ε. Ηλεκτρονικά ελεγχόμενος θάλαμος απόσταξης
Κωδικός Δ.Ε. 24139
Ονοματεπώνυμο φοιτητή Δημήτριος Ουζούνης
Ονοματεπώνυμο εισηγητή Αργύρης Χατζόπουλος
Ημερομηνία ανάληψης Δ.Ε. 05-03-2024
Ημερομηνία περάτωσης Δ.Ε.

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στην οικογένειά μου, τους καθηγητές μου και τους φίλους μου»

Πρόλογος

Στο πλαίσιο της διπλωματικής μου εργασίας, επέλεξα να αναπτύξω ένα ολοκληρωμένο σύστημα που θα μου επέτρεπε να αξιοποιήσω στο μέγιστο τις γνώσεις που απέκτησα τόσο κατά τη διάρκεια των προπτυχιακών μου σπουδών όσο και μέσα από την εμπειρία και την εξειδίκευση που προσέφερε το μεταπτυχιακό πρόγραμμα. Στόχος μου ήταν να συνδυάσω θεωρία και πράξη, εστιάζοντας σε τεχνολογίες που με συναρπάζουν: το Διαδίκτυο των Πραγμάτων, η Υπολογιστική Νέφους, η Ανάπτυξη Λογισμικού, οι Βιομηχανικοί Αυτοματισμοί και οι Τηλεπικοινωνίες.

Η εργασία μου περιλάμβανε τον σχεδιασμό της ηλεκτρονικής πλακέτας του συστήματος και τον προγραμματισμό της συσκευής IoT, καθώς και την ανάπτυξη του λογισμικού με χρήση σύγχρονων τεχνολογιών και μοντέρνων γλωσσών προγραμματισμού όπως η JavaScript και η Python. Μέσα από αυτή τη διαδικασία, ενίσχυσα ουσιαστικά τις τεχνικές μου δεξιότητες, αλλά και τις προσωπικές μου ικανότητες, εξελισσόμενος τόσο ως μηχανικός όσο και ως άνθρωπος.

Η συνεργασία μου με τον επιβλέποντα καθηγητή, κ. Αργύρη Χατζόπουλο, Επίκουρο Καθηγητή στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος, υπήρξε καθοριστική. Η καθοδήγησή του συνέβαλε ουσιαστικά στην επιστημονική και επαγγελματική μου ωρίμανση, προσδίδοντας στην εργασία μου βάθος και ποιότητα.

Περίληψη

Η παρούσα εργασία παρουσιάζει την ανάπτυξη ενός ηλεκτρονικά ελεγχόμενου θαλάμου απόσταξης, βασισμένου σε μικροελεγκτή Raspberry Pi 4, με χρήση της γλώσσας Python για τον έλεγχο και την απεικόνιση δεδομένων. Η θερμοκρασία παρακολουθείται μέσω δύο αισθητήρων PT100 και κυκλώματος προσαρμογής με μετατροπέα MAX31865, ενώ σχεδιάστηκε και κατασκευάστηκε πλακέτα επέκτασης για την ολοκληρωμένη λειτουργία του συστήματος. Το πείραμα επικεντρώθηκε στην απόσταξη νερού, με στόχο την αξιολόγηση της ακρίβειας και της σταθερότητας του συστήματος.

Το έργο αξιοποιεί αποκλειστικά open-source τεχνολογίες, ενισχύοντας τη δυνατότητα επαναχρησιμοποίησης και εξέλιξης από την κοινότητα. Επιπλέον, διερευνάται η προοπτική εφαρμογής του συστήματος σε βιομηχανικές διεργασίες, όπως η παραγωγή αιθέριων ελαίων και η φαρμακευτική απόσταξη, με στόχο την αυτοματοποίηση, την ενεργειακή αποδοτικότητα και την απομακρυσμένη παρακολούθηση μέσω IoT τεχνολογιών.

Η εργασία συνδυάζει στοιχεία θερμοδυναμικής, τηλεπικοινωνιών, αυτοματισμού και ανάπτυξης λογισμικού, προσφέροντας μια σύγχρονη και προσβάσιμη λύση για πειραματικές και επαγγελματικές εφαρμογές.

“Electronically controlled distillation chamber”

Dimitrios Ouzounis

Abstract

This study presents the development of an electronically controlled distillation chamber based on a Raspberry Pi 4 microcontroller, utilizing the Python programming language for system control and data visualization. Temperature monitoring is achieved through two PT100 sensors and an interface circuit with the MAX31865 converter, while a custom expansion board was designed and built to support the system’s full functionality. The experiment focused on water distillation, aiming to evaluate the system’s accuracy and stability.

The project relies exclusively on open-source technologies, enhancing its potential for reuse and further development by the open-source community. Additionally, it explores the system’s applicability to industrial processes such as essential oil production and pharmaceutical distillation, with an emphasis on automation, energy efficiency, and remote monitoring via IoT technologies.

The work integrates elements of thermodynamics, telecommunications, automation, and software development, offering a modern and accessible solution for both experimental and professional applications.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες προς την οικογένειά μου για την αδιάκοπη υποστήριξη, την υπομονή και την ενθάρρυνση που μου προσέφεραν καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερη μνεία οφείλω στον επιβλέποντα καθηγητή κ. Αργύρη Χατζόπουλο, για την εμπιστοσύνη που μου έδειξε, την εποικοδομητική συνεργασία, την πολύτιμη καθοδήγηση και την αμέριστη βοήθειά του καθ' όλη τη διάρκεια της παρούσας εργασίας.

Ευχαριστώ θερμά τους διδάσκοντες του Μεταπτυχιακού Προγράμματος Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» για την επιστημονική καθοδήγηση και τη γνώση που απλόχερα μου προσέφεραν. Ομοίως, ευχαριστώ τους συμφοιτητές μου για την άψογη συνεργασία και την κοινή πορεία μας σε αυτό το σύντομο αλλά ουσιαστικό ακαδημαϊκό ταξίδι.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς τους φίλους και τους ανθρώπους του προσωπικού μου περιβάλλοντος, των οποίων η παρουσία μου έλειψε λόγω των αυξημένων υποχρεώσεων, για την κατανόηση και την υπομονή τους.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

ADC	Analog to Digital Converter
AI	Artificial Intelligence
ARM	Advanced RISC Machines
AWS	Amazon Web Services
CRM	Customer Relationship Management
CS	Chip Select
DSS	Decision Support System
ERP	Enterprise Resource Planning
GB	Gigabyte
GIS	Geographic Information System
GPIO	General Purpose Input Output
I/O	Input / Output
IC	Integrated Circuit
IEC	International Electrotechnical Commission
IoT	Internet Of Things, Διαδίκτυο των Πραγμάτων
ISO	International Organization for Standardization
JS	JavaScript
JSON	JavaScript Object Notation
LAN	Local Area Network
LED	Light Emitting Diode
LCD	Liquid Crystal Display
POR	Power On Reset
RISC	Reduced Instruction Set Computer
RTD	Resistance Temperature Detector
SBC	Single Board Computer
SD	Secure Digital
SDI	Serial Data Input
SDO	Serial Data Output
SMD	Surface Mount Devices
SS	Slave Select
SSR	Solid State Relay
ΕΗΣ	Εφαρμοσμένα Ηλεκτρονικά Συστήματα
Η/Υ	Ηλεκτρονικός Υπολογιστής
ΚΜΕ	Κεντρική Μονάδα Ελέγχου
ΠΜΣ	Πρόγραμμα Μεταπτυχιακών Σπουδών
ΠΣ	Πληροφοριακά Συστήματα
ΤΝ	Τεχνητή Νοημοσύνη

Κεφάλαιο 1ο

1.1 Εισαγωγή

Η συνεχής εξέλιξη της επιστήμης και της τεχνολογίας, έχει οδηγήσει τον άνθρωπο να κατασκευάζει συστήματα και μηχανήματα που υπερτερούν από τα παλαιότερα. Η πρόοδος που σημειώνουν εδώ και χρόνια οι τομείς της πληροφορικής, των τηλεπικοινωνιών και των ηλεκτρονικών συστημάτων, έχει ως συνέπεια την σημερινή εποχή, η βιομηχανία να βιώνει την τέταρτη βιομηχανική επανάσταση, Industry 4.0. Τεχνολογίες όπως το διαδίκτυο των πραγμάτων, η υπολογιστική νέφους και η τεχνητή νοημοσύνη, είναι πλέον από τα βασικά συστατικά των βιομηχανικών συστημάτων και των συστημάτων αυτόματου ελέγχου.

Στην παρούσα διπλωματική εργασία παρουσιάζεται ένα σύγχρονο και μοντέρνο σύστημα απόσταξης, που προσφέρει αξιοπιστία και ακρίβεια, στη μέτρηση και επόπτευση της θερμοκρασίας. Η χρήση των καινοτόμων τεχνολογιών IoT και της υπολογιστικής νέφους, συνδυάζονται με την παράδοση, προσφέροντας την δυνατότητα παραγωγής αποσταγμάτων υψηλής ποιότητας με ακρίβεια και ασφάλεια. Ο ηλεκτρονικός θάλαμος απόσταξης είναι εξοπλισμένος με τα πλέον σύγχρονα χαρακτηριστικά, που επιτρέπουν την εύκολη παρακολούθηση και ρύθμιση της διαδικασίας απόσταξης, μέσω μιας απλής και φιλικής προς τον χρήστη διεπαφής.

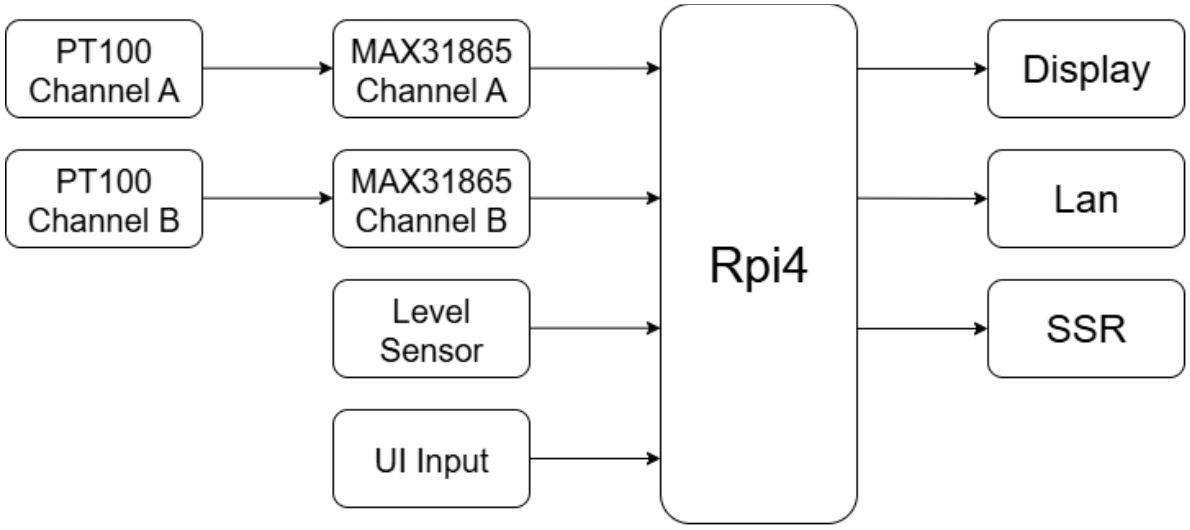
Εικόνα 1: Το σύστημα απόσταξης.

1.2 Συνοπτική περιγραφή συστήματος

Το σύστημα απαρτίζεται από την δεξαμενή συγκέντρωσης της πρώτης ύλης, που λειτουργεί και ως θάλαμος βρασμού. Η επόπτευση της θερμοκρασίας στον θάλαμο βρασμού πραγματοποιείται με την βοήθεια του συστήματος μετρήσεων. Μέσω της επόπτευσης, ελέγχουμε την μεταφορά της θερμότητας από την πηγή στον θάλαμο. Για αυτόν τον λόγο, στο σύστημα έχουν εγκατασταθεί δύο αισθητήρες θερμοκρασίας, μεταβλητής αντίστασης, τύπου PT100, για την ορθή και αξιόπιστη μέτρηση της, από 0 ºC έως 150 ºC, στον θάλαμο βρασμού και στον θάλαμο συμπύκνωσης. Οι αισθητήρες συνδέονται στην μετρητική διάταξη του συστήματος, όπου και γίνεται η ανάγνωση και επεξεργασία των πληροφοριών που μας δίνουν. Στο σύστημα

υπάρχει διαθέσιμη υποδομή για την εγκατάσταση ψηφιακού αισθητήρα στάθμης στη δεξαμενή βρασμού. Στο πλαίσιο του παρόντος πειράματος δεν χρησιμοποιήθηκε ο αισθητήρας. Αντ' αυτού, η λειτουργία του προσομοιώθηκε με έναν απλό διακόπτη.

Ο πίνακας ελέγχου της κατασκευής, στεγάζει όλα τα ηλεκτρονικά συστήματα και αποτελεί παράλληλα την φυσική διεπαφή χρήστη.



Εικόνα 2: Το block διάγραμμα του συστήματος ελέγχου και μέτρησης θερμοκρασίας.

1.2.1 Κεντρική μονάδα του συστήματος - Πίνακας ελέγχου

Η κεντρική μονάδα του συστήματος αποτελείται από έναν πίνακα ελέγχου, ο οποίος ενσωματώνει όλα τα ηλεκτρονικά υποσυστήματα και εξαρτήματα της κατασκευής. Το κυτίο είναι κατασκευασμένο από πλαστικό υψηλής αντοχής, κατάλληλο για βιομηχανική χρήση, και φέρει στεγανό καπάκι με βιδωτή σύνδεση μέσω τεσσάρων πλαστικών βιδών τύπου 1/4 στροφής, εξασφαλίζοντας εύκολη πρόσβαση στο εσωτερικό. Η στεγανότητα του περιβλήματος ενισχύεται με στυπιοθλίπτες, οι οποίοι επιτρέπουν την ασφαλή διέλευση καλωδίων, προστατεύοντας το εσωτερικό από υγρασία και σωματίδια. Ο βαθμός προστασίας είναι IP55, σύμφωνα με το πρότυπο IEC 60529, καθιστώντας το κατάλληλο για χρήση σε περιβάλλοντα με περιορισμένη έκθεση σε νερό και σκόνη. Ο πίνακας έχει υλοποιηθεί με βάση έναν μικροϋπολογιστή τύπου Raspberry Pi 4 Model B, ο οποίος λειτουργεί ως το βασικό στοιχείο επεξεργασίας και ελέγχου.

Το λογισμικό που διαχειρίζεται τόσο τη λειτουργία του υλικού (hardware) όσο και τη λογική της εφαρμογής έχει αναπτυχθεί στη γλώσσα προγραμματισμού Python. Μέσω αυτού, επιτυγχάνεται ο προγραμματισμένος έλεγχος των περιφερειακών μονάδων, η συλλογή και αποθήκευση δεδομένων, καθώς και η διασύνδεση με την εφαρμογή χρήστη. Οι περιφερειακές συσκευές και τα επιμέρους εξαρτήματα που περιλαμβάνονται στον πίνακα ελέγχου παρουσιάζονται αναλυτικά στον Πίνακα 1.1.

Πίνακας 1.1	
Ποσότητα	**Περιγραφή**
1	Πλακέτα επέκτασης
2	Αισθητήρας PT100, 3-wire
1	Rotary encoder
1	Solid state relay, 16A
3	Μπουτόν χειρισμού

5	Led indicators
1	Οθόνη LCD 4*20, I^2C
1	Διακόπτης προσομοίωσης αισθητήρας στάθμης
1	Ηλεκτρική συσκευή θέρμανσης
1	Μπουτόν εκτάκτου ανάγκης
1	Κυτίο στέγασης, IP55

Η μονάδα απεικόνισης αποτελείται από οθόνη τύπου LCD 4 γραμμών και 20 χαρακτήρων. Η σύνδεση με την κεντρική μονάδα πραγματοποιείται με το πρωτόκολλο I^2C και παρέχει σε πραγματικό χρόνο:

- Τιμές θερμοκρασίας από τους αισθητήρες μέτρησης
- Την τρέχουσα ώρα του συστήματος
- Μηνύματα κατάστασης και ειδοποιήσεις λειτουργίας

Ο rotary encoder χρησιμοποιείται για τη ρύθμιση της επιθυμητής θερμοκρασίας λειτουργίας, με βήμα μεταβολής ±1°C. Η περιστροφή του encoder μεταβάλλει την τιμή, ενώ η ενσωματωμένη λειτουργία push button επιτελεί τον ρόλο της επιβεβαίωσης επιλογής (Enter/OK).

Το μπουτόν έναρξης λειτουργίας (Start) ενεργοποιεί το σύστημα και εκκινεί τη διαδικασία ελέγχου θερμοκρασίας.

Το μπουτόν παύσης λειτουργίας (Stop) διακόπτει τη λειτουργία του συστήματος. Σε περίπτωση ενεργοποίησης του EMG (Emergency Button), το σύστημα εκτελεί εντολή επαναφοράς (reset), επανερχόμενο στην αρχική κατάσταση λειτουργίας, διαγράφοντας προσωρινές ρυθμίσεις και τερματίζοντας κάθε ενεργή διεργασία. Η λειτουργία αυτή ενισχύει την ασφάλεια του χειριστή και την προστασία του εξοπλισμού.

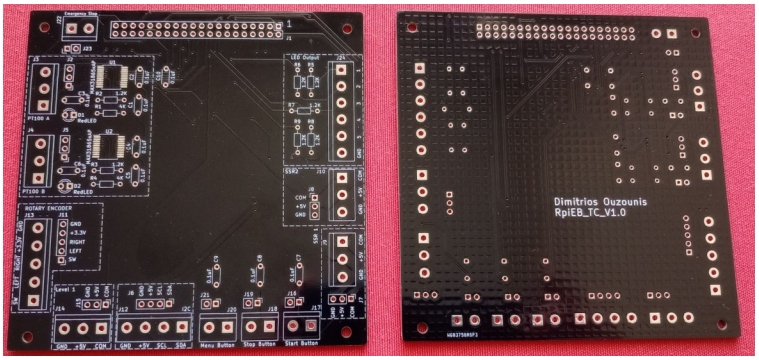
Το μπουτόν Menu αποτελεί βασικό στοιχείο διεπαφής του χρήστη με το σύστημα και παρέχει πρόσβαση στο μενού επιλογών και ρυθμίσεων. Η ενεργοποίησή του επιτρέπει την πλοήγηση σε προκαθορισμένες λειτουργίες, μέσω της οθόνης LCD και του rotary encoder. Οι διαθέσιμες επιλογές του μενού είναι:

- Χειροκίνητη σύνδεση με τον διακομιστή: Ο χρήστης έχει τη δυνατότητα να εκκινήσει χειροκίνητα τη διαδικασία σύνδεσης με τον απομακρυσμένο διακομιστή (server), για την αποστολή και λήψη των δεδομένων. Η λειτουργία αυτή είναι χρήσιμη σε περιπτώσεις όπου η αυτόματη σύνδεση έχει αποτύχει ή απαιτείται παρέμβαση από τον χειριστή.
- Τερματισμός λειτουργίας συστήματος: Μέσω της αντίστοιχης επιλογής, ο χρήστης μπορεί να διακόψει πλήρως τη λειτουργία του συστήματος, απενεργοποιώντας τις ενεργές διεργασίες και οδηγώντας το σύστημα σε κατάσταση απενεργοποίησης.

Εικόνα 3: Εμπρόσθια όψη του πίνακα ελέγχου.

1.2.3 Η πλακέτα επέκτασης

Η πλακέτα επέκτασης του συστήματος μετρήσεων περιέχει όλες τις φυσικές συνδέσεις των περιφερειακών συσκευών και ηλεκτρονικών εξαρτημάτων της κατασκευής. Το κύκλωμα προσαρμογής, των αισθητήρων μέτρησης θερμοκρασίας βρίσκεται στην πλακέτα. Ο αισθητήρας στάθμης, η οθόνη απεικόνισης των δεδομένων, τα κουμπιά χειρισμού, ο ηλεκτρονόμος (SSR) και οι λυχνίες ένδειξης συνδέονται με την πλακέτα επέκτασης. Η πλακέτα με την σειρά της συνδέεται με τον Η/Υ ενσύρματα, με ένα καλώδιο 20 pin (flat cable).



Εικόνα 4: Η πλακέτας επέκτασης.

Η σχεδίαση της υλοποιήθηκε με την εφαρμογή KiCad 7 και η κατασκευή του pcb έγινε από την εταιρεία PCBWay. Για την συναρμολόγηση της πλακέτας καθώς επίσης και για τις κολλήσεις των εξαρτημάτων χρησιμοποιήθηκαν αποκλειστικά εργαλεία χειρός.

1.2.4 Το σύστημα θέρμανσης

Για τη θέρμανση του λέβητα του συστήματος, χρησιμοποιήθηκε μία κοινή οικιακή ηλεκτρική εστία, η οποία παρείχε σταθερή και ελεγχόμενη θερμική ισχύ καθ’ όλη τη διάρκεια του πειράματος. Η επιλογή της

συγκεκριμένης πηγής θερμότητας εξασφάλισε σταθερότητα στις συνθήκες βρασμού και διευκόλυνε την παρακολούθηση της θερμοκρασιακής εξέλιξης μέσω των εγκατεστημένων αισθητήρων.

1.2.5 Η δεξαμενή

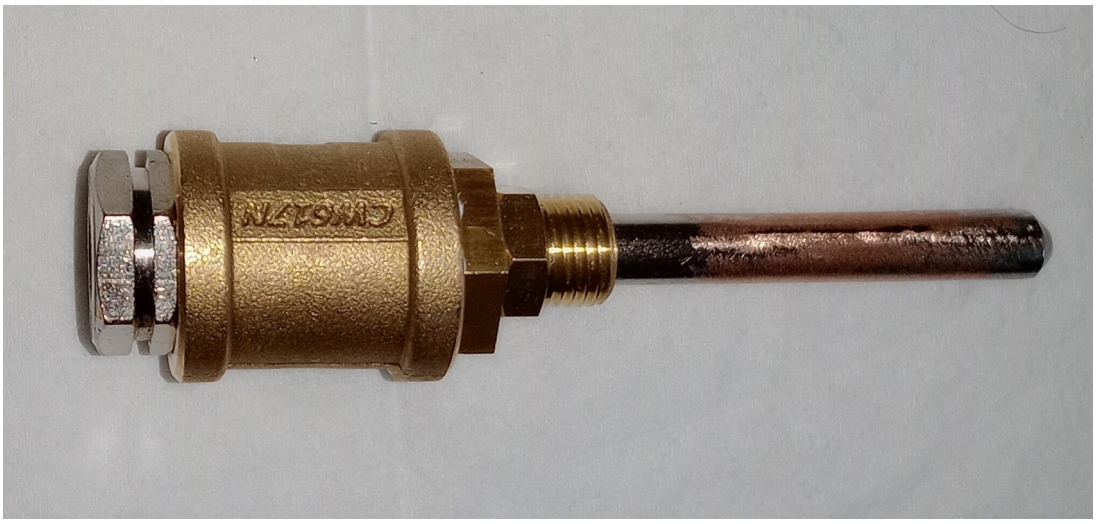
Οι δεξαμενές απόσταξης αλκοολούχων ποτών παίζουν καθοριστικό ρόλο στην παραγωγή αλκοολούχων ποτών, όπως τσίπουρο, ούζο και βότκα. Αυτές οι δεξαμενές κατασκευάζονται κυρίως από ανοξείδωτο χάλυβα, λόγω της ανθεκτικότητας τους στη διάβρωση και των ιδιοτήτων τους που δεν αλλοιώνουν το τελικό προϊόν.

Για το παρόν πείραμα χρησιμοποιήθηκε μια συσκευή απόσταξης, οι οποία προορίζεται για οικιακή χρήση και είναι κατασκευασμένη από ανοξείδωτο ατσάλι. Είναι μία οικονομική και αξιόπιστη επιλογή για πειραματισμό σε διάφορες τεχνικές απόσταξης, με ακρίβεια, ποιότητα και ευκολία στο χειρισμό (βλ. Παράρτημα Α για πληροφορίες κατασκευαστή).

Χαρακτηριστικά:

- Είναι κατασκευασμένη από ανοξείδωτο ατσάλι και χαλκό, διασφαλίζοντας ανθεκτικότητα και ψηλή θερμική αγωγιμότητα.
- Η χωρητικότητα της είναι 11,4 λίτρα, καθιστώντας την κατάλληλη για μικρές παραγωγές.
- Το σύστημα ψύξης στον θάλαμο συμπύκνωσης, απαρτίζεται από χάλκινο πηνίο.
- Αναλογικό θερμόμετρο διπλής ένδειξης θερμοκρασίας σε βαθμούς Κελσίου και Φαρενάιτ
- Σύστημα σφράγισης με ισχυρές πόρπες και φλάντζα από σιλικόνη τροφίμων για αποτελεσματική στεγανοποίηση.
- Ιδανική για απόσταξη κρασιού, μπράντι, ρούμι, ουίσκι, βότκα, τεκίλα, νερό και αιθέρια έλαια.

Η τοποθέτηση του αισθητήρα θερμοκρασίας PT100, πραγματοποιήθηκε στην θέση του αναλογικού θερμόμετρου που έχει τοποθετήσει ο κατασκευαστής στη δεξαμενή. Για την στερέωση του αισθητήρα χρησιμοποιήθηκε ειδική θήκη (κυάθιο) και θερμοαγώγιμη πάστα.



Εικόνα 5: Η θήκη του PT100.

1.2.6 Το πληροφοριακό σύστημα

Τα δεδομένα που συλλέγονται καθ' όλη την διάρκεια λειτουργίας του συστήματος αποστέλλονται σε απομακρυσμένο Η/Υ, στον οποίο έχει εγκατασταθεί ο διακομιστής (server) του συστήματος. Στον server επεξεργάζονται και αποθηκεύονται όλα τα δεδομένα του συστήματος και μέσω της εφαρμογής χρήστη, δίνεται η δυνατότητα για απομακρυσμένη επόπτευση και πλήρη έλεγχο όλων των λειτουργιών του συστήματος.

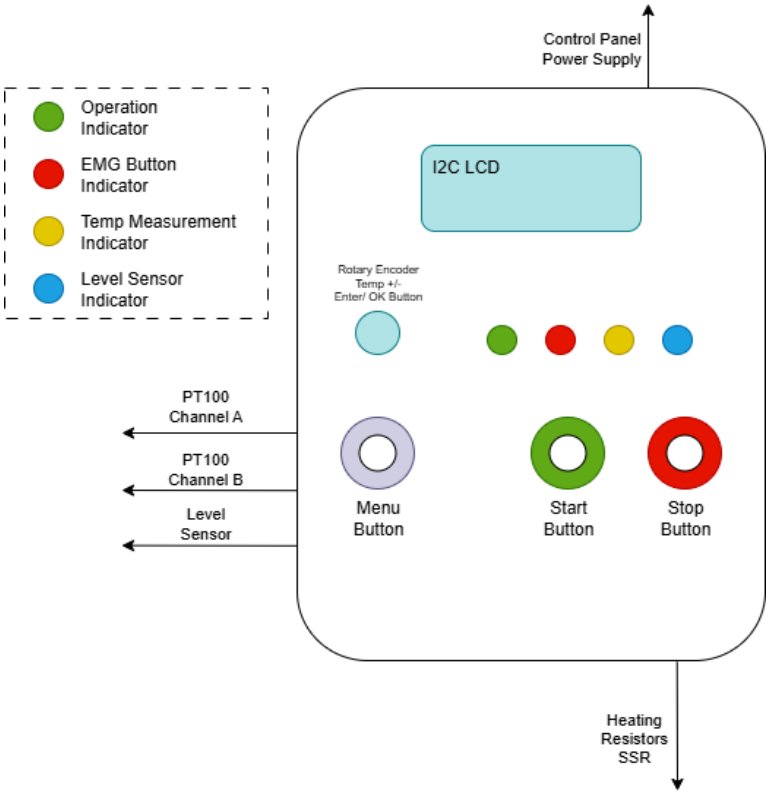
Η τεχνολογία που χρησιμοποιήθηκε για την υλοποίηση του διακομιστή και της εφαρμογής χρήστη είναι η γλώσσα προγραμματισμού JavaScript και το framework Node.js, με χρήση πρόσθετων βιβλιοθηκών για την ανάπτυξη της διεπαφής χρήστη και των επιπλέων λειτουργιών της εφαρμογής χρήστη. Ο Node.js server χρησιμοποιεί την V8 JavaScript engine της Google για να εκτελεί JavaScript κώδικα εκτός του προγράμματος περιήγησης.

Για την υλοποίηση της βάσης δεδομένων, όπου αποθηκεύονται οι πληροφορίες που καταγράφει το σύστημα, χρησιμοποιήθηκε το λογισμικό διαχείρισης σχεσιακών βάσεων δεδομένων MySQL.

1.3 Περιγραφή λειτουργίας

Η διαδικασία λειτουργίας του συστήματος απόσταξης ξεκινά με την τροφοδοσία και ενεργοποίηση της συσκευής. Μετά την εκκίνηση, το σύστημα εισέρχεται σε κατάσταση αναμονής, κατά την οποία η οθόνη απεικονίζει την τρέχουσα ώρα, τις μετρήσεις θερμοκρασίας από τους αισθητήρες PT100, καθώς και την κατάσταση λειτουργίας (ενεργή ή ανενεργή). Παράλληλα, η κίτρινη λυχνία LED αναβοσβήνει κάθε φορά που καταγράφεται νέα μέτρηση θερμοκρασίας, για να επιβεβαιώσει και οπτικά στον χρήστη την ενεργή λειτουργία του συστήματος.

Η ρύθμιση της επιθυμητής θερμοκρασίας πραγματοποιείται μέσω του διαθέσιμου περιστροφικού επιλογέα (rotary encoder), με δυνατότητα αυξομείωσης κατά ±1°C. Με την ενεργοποίηση του κουμπιού Start, εκκινείται η διαδικασία απόσταξης. Ο ηλεκτρονόμος ενεργοποιείται, επιτρέποντας στο σύστημα θέρμανσης να θερμάνει τον θάλαμο βρασμού. Η θέρμανση συνεχίζεται έως ότου η θερμοκρασία φτάσει στο προκαθορισμένο επίπεδο, οπότε και το σύστημα θέρμανσης απενεργοποιείται. Εφόσον η θερμοκρασία πέσει κάτω από το όριο ρύθμισης, η θέρμανση επανεκκινείται αυτόματα. Καθ' όλη τη διάρκεια της διαδικασίας, η πράσινη λυχνία LED παραμένει ενεργή, υποδεικνύοντας την ομαλή λειτουργία του συστήματος.



Εικόνα 6: Το σχέδιο του πίνακα ελέγχου.

Η διακοπή της λειτουργίας πραγματοποιείται με το πάτημα του μπουτόν Stop, το οποίο τερματίζει τη διαδικασία και επαναφέρει το σύστημα σε κατάσταση αναμονής, διατηρώντας τις ρυθμίσεις του χρήστη.

Σε περίπτωση έκτακτης ανάγκης, ο χρήστης έχει τη δυνατότητα να ενεργοποιήσει το ειδικό κουμπί ασφαλείας. Η χρήση του προβλέπεται σε περιπτώσεις που απειλείται η υγεία του χειριστή, η ακεραιότητα της παραγωγικής διαδικασίας, ο εξοπλισμός ή οι εγκαταστάσεις. Με την ενεργοποίηση του κουμπιού, όλες οι λειτουργίες του συστήματος διακόπτονται άμεσα και η κόκκινη λυχνία LED αρχίζει να αναβοσβήνει, σηματοδοτώντας την κατάσταση συναγερμού. Η επαναφορά του συστήματος πραγματοποιείται με παρατεταμένο πάτημα του κουμπιού Stop για δύο δευτερόλεπτα, οπότε και το σύστημα επιστρέφει σε κατάσταση αναμονής με τις αρχικές ρυθμίσεις.

Η μπλε λυχνία LED λειτουργεί ως οπτική ένδειξη της κατάστασης του αισθητήρα στάθμης. Όταν η στάθμη του υγρού εντός του θαλάμου πλήρωσης φτάσει στο προκαθορισμένο όριο, ο αισθητήρας ενεργοποιείται και, ως αποτέλεσμα, η μπλε λυχνία LED ανάβει. Παράλληλα ενημερώνεται και η ένδειξη στην οθόνη LCD. Η λειτουργία αυτή επιτρέπει στον χρήστη να παρακολουθεί σε πραγματικό χρόνο την πλήρωση του συστήματος, εξασφαλίζοντας την ακρίβεια και την ασφάλεια της διαδικασίας.

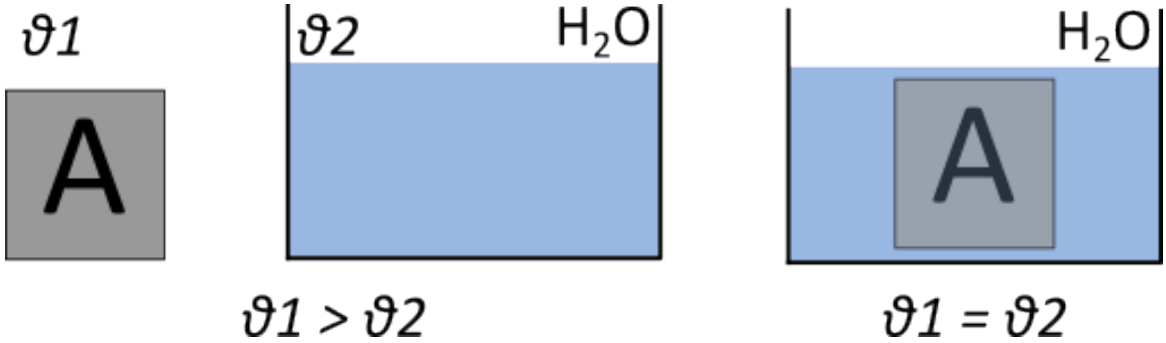
Η πρόσβαση στο μενού ρυθμίσεων πραγματοποιείται πιέζοντας το κουμπί Menu, το οποίο επιτρέπει στον χρήστη να εισέλθει στο περιβάλλον παραμετροποίησης του συστήματος. Από το μενού αυτό, παρέχεται η δυνατότητα χειροκίνητης σύνδεσης με τον διακομιστή, καθώς και η ασφαλής απενεργοποίηση του συστήματος. Η πλοήγηση εντός του μενού γίνεται μέσω του rotary encoder, ο οποίος επιτρέπει την επιλογή των διαθέσιμων λειτουργιών. Η επιβεβαίωση της εκάστοτε επιλογής πραγματοποιείται με το πάτημα του ενσωματωμένου push button, ενεργοποιώντας τη λειτουργία OK. Η διαδικασία αυτή εξασφαλίζει την ορθή και ασφαλή διαχείριση του συστήματος, προσφέροντας στον χρήστη πλήρη έλεγχο των βασικών λειτουργιών μέσω ενός απλού και εργονομικού μηχανισμού αλληλεπίδρασης.

Κεφάλαιο 2ο : Θεωρητικό μέρος

2.1 Θερμοκρασία και μετάδοση θερμότητας

Η θερμοκρασία και η μετάδοση θερμότητας είναι θεμελιώδεις έννοιες στη φυσική. Η θερμοκρασία είναι ένα φυσικό μέγεθος το οποίο μετράει πόσο ψυχρό και πόσο θερμό είναι ένα αντικείμενο, ένα σύστημα ή ένα υλικό και να αποδώσουμε ποσοτική έννοια στο κρύο και το ζεστό. Η μεταβολή της τιμής της οφείλεται στην μεταφορά ενέργειας και συγκεκριμένα στη μεταφορά θερμότητας, από ένα αντικείμενο υψηλότερης θερμοκρασίας, ζεστό προς ένα χαμηλότερης, κρύο.

Εάν τοποθετήσουμε ένα θερμό αντικείμενο μέσα σε ένα δοχείο που περιέχει κρύο νερό, το αντικείμενο θα αρχίσει να ψύχεται και το νερό να θερμαίνεται. Αυτό το φαινόμενο θα συμβαίνει έως ότου και τα δύο αποκτήσουν την ίδια θερμοκρασία και βρεθούν σε κατάσταση θερμικής ισορροπίας. Η μεταφορά ενέργειας που οφείλεται στην διαφορά θερμοκρασίας ονομάζεται μετάδοση θερμότητας [1, 2].



Εικόνα 7: Μεταφορά θερμότητας.

Υπάρχουν τρεις διαφορετικοί τρόποι μετάδοσης θερμότητας. Αυτοί είναι η αγωγή (conduction), η συναγωγή (convection) και η θερμική ακτινοβολία (thermal radiation). Η μέθοδος της αγωγής αφορά την διάδοση θερμότητας μεταξύ δύο σωμάτων ή αντικειμένων σε επαφή. Ένα παράδειγμα μεταφοράς θερμότητας με αγωγή είναι ένα μεταλλικό κουτάλι που γίνεται ζεστό όταν το βυθίζουμε σε ζεστό νερό. Η μέθοδος της συναγωγής αφορά την μεταφορά θερμότητας μεταξύ σταθερής επιφάνειας και κινούμενου ρευστού. Για παράδειγμα, ο ζεστός αέρας που ανεβαίνει από ένα καλοριφέρ. Η τρίτη μέθοδος, της θερμικής ακτινοβολίας, πραγματοποιείται μεταξύ δύο σωμάτων ή συστημάτων χωρίς την ανάγκη να υπάρχει φυσική

επαφή ή την παρουσία κάποιου μέσου για την μεταφορά θερμότητας. Με αυτόν τον τρόπο γίνεται η μεταφορά θερμότητας από τον Ήλιο στη Γη [1, 2]. Η μέθοδος που χρησιμοποιήθηκε για την θέρμανση της δεξαμενής στο παρόν πείραμα είναι αυτός της αγωγής.

Η ορθή και αξιόπιστη μέτρηση της θερμοκρασίας απασχολεί την επιστημονική κοινότητα μέχρι σήμερα. Για την μέτρηση της γίνεται χρήση ειδικών συστημάτων μέτρησης και το διεθνές σύστημα μονάδων ορίζει ως μονάδα μέτρησης της θερμοκρασίας τους βαθμούς Κέλβιν (K). Στην πλειοψηφία των εφαρμογών όμως, γίνεται χρήση της κλίμακας βαθμών Κελσίου (°C). Η μετατροπή από βαθμούς Κέλβιν σε βαθμούς Κελσίου γίνεται με την σχέση :

$$K = {}^{o}C + 273{,}15 \text{ [3]} \tag{2.1}$$

Μία ακόμη κλίμακα που χρησιμοποιείται συχνά για την μέτρηση της θερμοκρασίας είναι η κλίμακα των βαθμών Φαρενάιτ (°F). Η μετατροπή από την κλίμακα Κελσίου σε Φαρενάιτ γίνεται με την σχέση :

$$^{o}F = {}^{o}C \times (9/5) + 32 \text{ [3]} \tag{2.2}$$

Και για την μετατροπή από την κλίμακα Φαρενάιτ σε Κελσίου με την σχέση :

$$^{o}C = (^{o}F + 32) \times (5/9) \text{ [3]} \tag{2.3}$$

Στο παρόν πείραμα η θερμοκρασία αποδίδεται σε βαθμούς της κλίμακας Κελσίου.

2.2 Αισθητήρες μέτρησης θερμοκρασίας

Στην παρακάτω εικόνα μπορούμε να δούμε την βασική δομή ενός συστήματος μέτρησης φυσικών μεγεθών.

Εικόνα 8: Δομή συστήματος μέτρησης.

Αιχμή του δόρατος για ένα σύστημα μετρήσεων είναι ο αισθητήρας, ο οποίος μετατρέπει το φυσικό μέγεθος σε πληροφορία με τη μορφή ηλεκτρικού σήματος. Στη συνέχεια η βαθμίδα της προσαρμογής σήματος θα αποδώσει το κατάλληλο εύρος τιμών στο ηλεκτρικό σήμα, ώστε αυτό να είναι στην κατάλληλη μορφή για επεξεργασία, συλλογή και τελική απεικόνιση με την χρησιμοποίηση συστημάτων υψηλής τεχνολογίας υλικού και λογισμικού.
Οι αισθητήρες χωρίζονται σε κατηγορίες ανάλογα με την ανιχνεύσιμη μορφή ενέργειας που μετράνε:

- ηλεκτρική
- θερμική
- μηχανική
- μαγνητική
- χημική
- ακτινοβολία

Επίσης χωρίζονται σε παθητικούς και ενεργούς. Οι παθητικοί αισθητήρες ανιχνεύουν και αντιδρούν σε ερεθίσματα από το φυσικό περιβάλλον, παράγοντας ένα ηλεκτρικό σήμα ως απόκριση των μεταβολών αυτών, χωρίς την απαίτηση ηλεκτρικής ισχύς. Το θερμοζεύγος ή θερμοστοιχείο, ανήκει σε αυτή την κατηγορία αισθητήρων.

Οι ενεργοί αισθητήρες σε αντίθεση με τους παθητικούς, απαιτούν την παρουσία εξωτερικής μονάδας τροφοδοσίας ενέργειας, προκειμένου να παράγουν σήμα πληροφορίας στην έξοδο. Σε αυτή την κατηγορία ανήκει ο ψηφιακός αισθητήρας θερμοκρασίας DS18b20.

Οι αισθητήρες μέτρησης θερμοκρασίας χρησιμοποιούνται ευρέως για την παρακολούθηση και τον έλεγχο της θερμοκρασίας σε διάφορες εφαρμογές, από βιομηχανικά συστήματα και ιατρικές συσκευές έως οικιακές συσκευές και κλιματιστικά συστήματα.

Υπάρχουν πάρα πολλοί τρόποι που χρησιμοποιούνται στην καθημερινότητα για την μέτρηση της θερμοκρασίας. Ένας από τους πιο δημοφιλής είναι με την χρήση διατάξεων μέτρησης με αισθητήρες μεταβλητής αντίστασης. Οι αισθητήρες τύπου RTD (Resistance Temperature Detectors) είναι αισθητήρες των οποίων η αντίσταση μεταβάλλεται ανάλογα με την θερμοκρασία και ανήκουν στην κατηγορία των παθητικών αισθητήρων. Για την κατασκευή των RTD, μπορούν να χρησιμοποιηθούν υλικά όπως ο χαλκός και το νικέλιο. Το εύρος μέτρησης του αισθητήρα εξαρτάται από το υλικό κατασκευής του. Στον πίνακα 2.1 παρουσιάζεται το εύρος μέτρησης της θερμοκρασίας στα πιο δημοφιλή μέταλλα κατασκευής των RTD αισθητήρων [3].

Πίνακας 2.1	
Υλικό	**Εύρος μέτρησης θερμοκρασίας**
Πλατίνα	-270 ºC έως +850 ºC
Χαλκός	-100 ºC έως +200 ºC
Νικέλιο	-150 ºC έως +300 ºC
Βολφράμιο	-270 ºC έως +1100 ºC

Το συνηθέστερο υλικό κατασκευής τους είναι η πλατίνα, η οποία προσφέρει εξαιρετική ακρίβεια, εύρος μέτρησης της θερμοκρασίας και σχετική γραμμικότητα. Ενδεικτικά, η ακρίβεια τους μπορεί να φτάσει έως και τα ±0.1°C. Τα RTD πλατίνας συχνά αναφέρονται ως PT-RTD. Οι αισθητήρες μέτρησης θερμοκρασίας PT-RTD, με ονομαστική αντίσταση στους 0 ºC τα 100Ω και 1000Ω, είναι οι δημοφιλέστεροι και αναφέρονται και ως PT100. Είναι ιδανική επιλογή για συστήματα που το εύρος μέτρησης της θερμοκρασίας είναι από -200 ºC μέχρι +800 ºC. Χρησιμοποιούνται σε βιομηχανίες όπως η χημική, η φαρμακευτική, των τροφίμων και της ενέργεια. Επίσης, βρίσκουν εφαρμογή σε εργαστήρια, ερευνητικά κέντρα και σε συστήματα αυτόματου ελέγχου. Η εγκατάστασή τους είναι σχετικά εύκολη, ενώ μπορούν να συνδυαστούν με συστήματα ελέγχου για συνεχή παρακολούθηση και ανάλυση της θερμοκρασίας σε πραγματικό χρόνο.

Εικόνα 9: Αισθητήρας PT100.

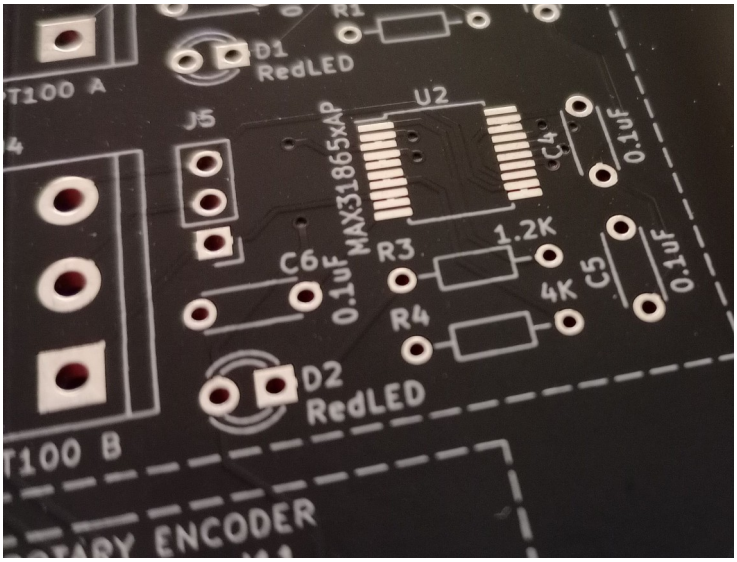
Οφείλουμε όμως να λάβουμε υπόψη μας και όλους τους παράγοντες που μπορεί να μας οδηγήσουν σε μετρήσεις με σφάλματα. Ένας αισθητήρας PT100 στους 0 ºC έχει 100Ohm αντίσταση. Η μέτρηση της τιμής της αντίστασης μας δίνει την δυνατότητα να υπολογίζουμε την τιμή της θερμοκρασίας. Αυτό σημαίνει ότι ένα σφάλμα στην μέτρηση της αντίστασης του αισθητήρα, θα μας οδηγήσει σε λάθος υπολογισμό στην τιμή της θερμοκρασίας.

Οι αιτίες που μπορούν να οδηγήσουν σε σφάλμα μέτρησης ποικίλουν ανάλογα την εφαρμογή, τις απαιτήσεις της, το περιβάλλον που θα τοποθετηθεί ο αισθητήρας και το σύστημα μέτρησης. Ένας βασικός παράγοντας που πρέπει να ληφθεί υπόψιν, κατά τη λήψη μετρήσεων, είναι η αντίσταση των αγωγών σύνδεσης του αισθητήρα. Για τον ορθό υπολογισμό της τιμής της θερμοκρασίας πρέπει να λάβουμε υπόψιν και την αντίσταση των αγωγών. Η σύνδεση του αισθητήρα με τα ηλεκτρονικά συστήματα είναι ενσύρματη με την χρήση δύο, τριών ή τεσσάρων καλωδίων. Ανάλογα τη σύνδεση ποικίλει και η ακρίβεια. Για την παρούσα κατασκευή επιλέχθηκε η σύνδεση τριών καλωδίων.

Άλλοι δημοφιλής αισθητήρες μέτρησης θερμοκρασίας είναι το θερμοζεύγος, τα θερμίστορ, τα διμεταλλικά ελάσματα και οι ψηφιακοί αισθητήρες θερμοκρασίας σε μορφή ολοκληρωμένου κυκλώματος. Υπάρχουν φυσικά και άλλοι μέθοδοι για τη μέτρηση της θερμοκρασίας με χρήση θερμομέτρων ακτινοβολίας και μέτρηση θερμοκρασίας με μεταβολή της πίεσης [2].

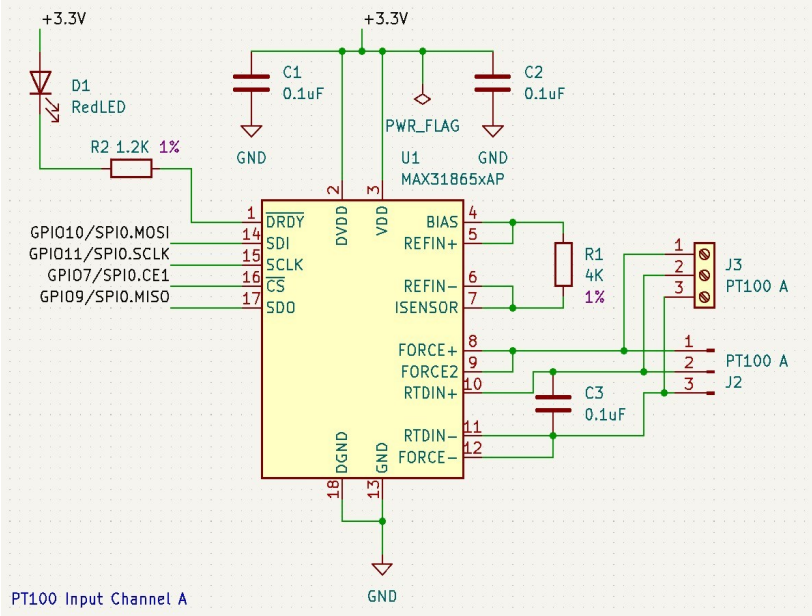
2.3 Το κύκλωμα προσαρμογής

Για την ορθή και αξιόπιστη μέτρηση της θερμοκρασίας στο παρόν πείραμα υλοποιήθηκε μία μετρητική διάταξη βασισμένη σε αισθητήρα τύπου PT100. Το κύκλωμα προσαρμογής των αισθητήρων PT100, βασίστηκε πάνω στο ολοκληρωμένο εξάρτημα MAX31865, της εταιρείας Maxim Integrated [17]. Σε αντίθετη περίπτωση θα μπορούσε να γίνει χρήση περίπλοκων ηλεκτρονικών κυκλωμάτων που περιλαμβάνουν υποσυστήματα με διατάξεις ενισχυτών για συστήματα μετρήσεων, φίλτρα για το φιλτράρισμα του ηλεκτρικού σήματος, βαθμίδα γραμμικοποίησης σήματος και σύστημα μετατροπής σήματος από αναλογικό σε ψηφιακό.



Εικόνα 10: Φωτογραφία της πλακέτα

Οι πυκνωτές των 0.1μF που χρησιμοποιούνται στο κύκλωμα είναι απόζευξης. Σκοπός τους είναι να προστατεύουν το MAX31865, φιλτράροντας τον υψηλής συχνότητας θόρυβο, απομονώνοντας τα AC σήματα που μπορεί να βρίσκονται στην ίδια γραμμή. Επίσης διασφαλίζουν ότι η τάση τροφοδοσίας θα παραμείνει σταθερή και καθαρή, απορροφώντας μεταβολές που θα επηρεάσουν την λειτουργία του κυκλώματος, όπως την αλλοίωση των ψηφιακών σημάτων [17, 18].

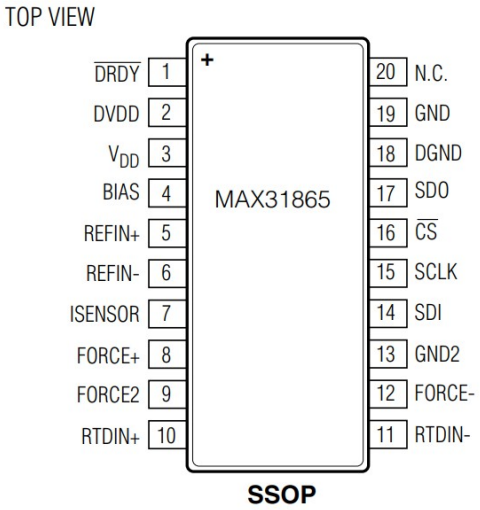


Εικόνα 11: Το κύκλωμα προσαρμογής.

2.3.1 Περιγραφή MAX31865

Το περιεχόμενο της παρούσας ενότητας βασίζεται αποκλειστικά στο επίσημο data sheet του IC MAX31865, όπως παρέχεται από τον κατασκευαστή [17].

Το IC MAX31865 είναι ένας ψηφιακός μετατροπέας αντίστασης RTD σε ψηφιακό σήμα, που δημιουργήθηκε από την Analog Devices. Χρησιμοποιείται για την μετατροπή της αντίστασης ενός RTD σε ψηφιακή τιμή, επιτρέποντας την ακριβή μέτρηση της θερμότητας με ανάλυση 15 bit. Επίσης διαθέτει προστασία εισόδου, ψηφιακό ελεγκτή, πρωτόκολλο επικοινωνίας SPI και οχτώ καταχωρητές των 8 bit για την αποθήκευση δεδομένων και τον προγραμματισμό της λειτουργίας του. Ιδανική επιλογή για την υλοποίηση του συστήματος μέτρησης θερμοκρασίας του πειράματος για την απλότητα που προσφέρει στον σχεδιασμό του ηλεκτρονικού κυκλώματος, την ακρίβεια στην μετατροπή του αναλογικού σήματος σε ψηφιακό, την ενσύρματη επικοινωνία με την κεντρική μονάδα και το μικρό συνολικό κόστος της κατασκευής [17, 18].



Εικόνα 12: Το IC MAX31865 [17].

Στον πίνακα 2.2 παρουσιάζονται οι ακροδέκτες του ολοκληρωμένου κυκλώματος MAX31865.

Πίνακας 2.2		
Ακροδέκτης	**Όνομα**	**Περιγραφή**
1	DRDY	Ακροδέκτης εξόδου. Όταν είναι 'LOW', υπάρχει διαθέσιμη μία νέα μέτρηση στον καταχωρητή δεδομένων.
2	DVDD	Ψηφιακή τροφοδοσία εισόδου. Συνδέεται με παροχή +3.3V. Συνδέστε ένα πυκνωτή 0.1μF μεταξύ αυτού και του ακροδέκτη DGND.
3	VDD	Τροφοδοσία εισόδου. Συνδέεται με παροχή +3.3V. Συνδέστε ένα πυκνωτή 0.1μF μεταξύ αυτού και του ακροδέκτη GND1.
4	BIAS	Έξοδος τάσης πόλωσης V_{BIAS}.
5	REFIN+	Θετική είσοδος τάσης αναφοράς. Σύνδεση αντίστασης αναφοράς R_{REF}, μεταξύ REFIN+ και REFIN-.
6	REFIN-	Αρνητική είσοδος τάσης αναφοράς. Σύνδεση αντίστασης αναφοράς R_{REF}, μεταξύ REFIN+ και REFIN-.
7	ISENSOR	'Χαμηλή πλευρά' (low side) της αντίστασης αναφοράς R_{REF}, Σύνδεση με τον ακροδέκτη REFIN-.
8	FORCE+	'Υψηλή πλευρά' (high side) του αισθητήρα θερμοκρασίας. Στη σύνδεση τριών καλωδίων συνδέεται με τον ακροδέκτη FORCE2.
9	FORCE2	Θετική είσοδος μόνο για την συνδεσμολογία τριών καλωδίων. Συνδέεται με τον ακροδέκτη FORCE+ και έχει ±45V προστασία εισόδου. Για την σύνδεση δύο ή τεσσάρων καλωδίων, ο ακροδέκτης συνδέεται με την γείωση.
10	RTDIN+	Θετική είσοδος του αισθητήρα θερμοκρασίας, με ±45V προστασία εισόδου.
11	RTDIN-	Αρνητική είσοδος του αισθητήρα θερμοκρασίας, με ±45V προστασία εισόδου.
12	FORCE-	'Χαμηλή πλευρά' (low side) της επιστροφής του αισθητήρα θερμοκρασίας, με ±45V προστασία εισόδου.
13	GND2	Ακροδέκτης γείωσης, είναι συνδεδεμένο με τον ακροδέκτη GND1.
14	SDI	Ακροδέκτης εισόδου δεδομένων της σειριακής θύρας.
15	SCLK	Το ρολόι της σειριακής εισόδου.
16	CS	Ακροδέκτης ενεργοποίησης της σειριακής επικοινωνίας, όταν είναι σε κατάσταση 'low'.
17	SDO	Ακροδέκτης εξόδου δεδομένων της σειριακής θύρας.
18	DGND	Ακροδέκτης ψηφιακής γείωσης.
19	GND1	Ακροδέκτης γείωσης, είναι συνδεδεμένος με τον ακροδέκτη GND2.
20	N. C.	Ασύνδετος ακροδέκτης.

Το MAX31865 διαθέτει οχτώ καταχωρητές των 8 bit. Στους καταχωρητές αυτούς αποθηκεύονται δεδομένα όπως η κατάσταση λειτουργίας, οι παράμετροι λειτουργίας και τα δεδομένα μετατροπής. Ο προγραμματισμός της συσκευής γίνεται επιλέγοντας τις κατάλληλες διευθύνσεις των καταχωρητών που επιθυμούμε. Για να κάνουμε ανάγνωση του περιεχομένου των καταχωρητών, χρησιμοποιούμε τις διευθύνσεις με το πρόθεμα '0xh'. Για να αποθηκεύσουμε πληροφορίες στους καταχωρητές, χρησιμοποιούμε τις διευθύνσεις με το πρόθεμα '8xh'. Τα δεδομένα των καταχωρητών γράφονται και διαβάζονται πρώτα στα περισσότερο σημαντικά bit (MSB). Στον πίνακα 2.3 παρουσιάζονται οι διευθύνσεις των εσωτερικών καταχωρητών [17, 18].

Πίνακας 2.3				
Όνομα καταχωρητή	**Διεύθυνση ανάγνωσης**	**Διεύθυνση εγγραφής**	**Κατάσταση Power On Reset**	**Εγγραφή / Ανάγνωση**
Configuration	00h	80h	00h	R/W
RTD MSBs	01h	-	00h	R
RTD LSBs	02h	-	00h	R
High Fault Threshold MSB	03h	83h	FFh	R/W
High Fault Threshold LSB	04h	84h	FFh	R/W
Low Fault Threshold MSB	05h	85h	00h	R/W
Low Fault Threshold LSB	06h	86h	00h	R/W
Fault Status	07h	-	00h	R

2.3.1.1 Καταχωρητής παραμέτρων

Ο καταχωρητής παραμέτρων, Configuration Register, είναι υπεύθυνος για την επιλογή λειτουργίας του ολοκληρωμένου. Επιλέγει τον τύπο σύνδεσης του RTD και ενεργοποιεί ή απενεργοποιεί τον ακροδέκτη εξόδου BIAS. Εκτελεί μετατροπές θερμοκρασίας τύπου 1-shot, επιλέγει τη συχνότητα λειτουργίας του φίλτρου τύπου notch και εκτελεί πλήρη ανίχνευση σφάλματος. Στον πίνακα 2.4 παρουσιάζονται τα bit ελέγχου του καταχωρητή ρυθμίσεων.

Πίνακας 2.4							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
V_{BIAS} 1=ON 0=OFF	Τρόπος λειτουργίας 1 = AUTO 0 = Normally Off	1-shot 1 = 1-shot (auto clear)	3-wire 1=3-wire RTD	Έλεγχος κύκλου ανίχνευσης σφαλμάτων (Πίνακας 2.5).		Fault Status Clear 1=Clear (auto clear)	Επιλογή συχνότητας φίλτρου 1 = 50Hz 0 = 60Hz

Με το bit D7, ελέγχουμε τον ακροδέκτη εξόδου BIAS. Από τον ακροδέκτη αυτόν, ελέγχουμε την τάση πόλωσης V_{BIAS}. Όταν δεν λαμβάνει χώρα κάποια μετατροπή, η V_{BIAS}, μπορεί να απενεργοποιηθεί για την μείωση της κατανάλωσης ισχύος στο κύκλωμα. Θέτουμε το bit D7 σε '1' για να ενεργοποιήσουμε την τάση πόλωσης, πριν από την έναρξη της μετατροπής τύπου 1-shot. Όταν επιλέξουμε λειτουργία αυτόματης μετατροπής σήματος, η τάση V_{BIAS}, παραμένει σε κατάσταση '1'.

Από το bit D6 έχουμε την δυνατότητα να επιλέξουμε την εκκίνηση της αυτόματης διαδικασίας ή 'Normally Off' διαδικασίας. Όταν θέσουμε το bit D6 σε '1', τότε θα έχουμε επιλέξει την λειτουργία της αυτόματης μετατροπής, η οποία υλοποιείται σε κύκλο 50Hz ή 60Hz. Όταν θέσουμε το bit D6 σε '0', τότε παύει η λειτουργία αυτόματης μετατροπής και ενεργοποιείται η λειτουργία 'Normally Off'. Όταν επιλέξουμε λειτουργία 'Normally Off', έχουμε την δυνατότητα να εκκινήσουμε την διαδικασία 1-shot. Η λειτουργία 1-shot, εκτελεί μία μετατροπή στην μέτρηση της αντίστασης του αισθητήρα. Για να συμβεί αυτό θα πρέπει η επιλεγμένη λειτουργία να είναι 'Normally Off' και να θέσουμε το bit D5 σε κατάσταση '1'. Η μετατροπή θα εκκινήσει, όταν η κατάσταση του ακροδέκτη CS γίνει '1', αφού γράψουμε στα περιεχόμενα αυτού του bit '1'. Όταν η τάση V_{BIAS} είναι ενεργή και ο ακροδέκτης CS εισέλθει σε κατάσταση '1', λαμβάνεται δείγμα από την τάση του RTD και αρχίζει η μετατροπή του σήματος. Όταν η τάση πόλωσης V_{BIAS} είναι απενεργοποιημένη, οι πυκνωτές που έχουν τοποθετηθεί στις RTDIN εισόδους, θα χρειαστεί πρώτα να φορτίσουν πριν λάβει χώρα μία μετατροπή σήματος με ακρίβεια και χωρίς σφάλματα. Έτσι όταν ενεργοποιούμε την V_{BIAS}, χρειάζεται να περιμένουμε 10.5 σταθερές χρόνου από το RC κύκλωμα εισόδου, συν επιπλέον 1ms, πριν αρχίσουμε να εκτελούμε μετατροπή σήματος. Μία μετατροπή χρειάζεται περίπου 52ms χρόνο, για συχνότητα φίλτρου στα 60Hz και 62.5ms για συχνότητα φίλτρου στα 50Hz.

Το bit D4 μας δίνει την δυνατότητα να θέσουμε τη λειτουργία τριών αγωγών σύνδεσης του αισθητήρα με το MAX31865. Σε αυτή τη λειτουργία η τάση μεταξύ FORCE+ και RTDIN+, αφαιρείται από το υπόλοιπο της αφαίρεσης της τάσης του RTDIN+ μείον του RTDIN-, για να ισοσταθμιστεί το σφάλμα που προκαλείται από την χρήση ενός καλωδίου/αγωγού στην σύνδεση στον ακροδέκτη FORCE- και RTDIN-. Για χρήση σύνδεσης δύο ή τεσσάρων αγωγών θέτουμε '0'. Τα bit D3 και D2, ελέγχουν την λειτουργία του κύκλου ανίχνευσης σφαλμάτων. Η λειτουργία αυτή μπορεί να εκκινηθεί από την μονάδα ελέγχου.

Όταν θέσουμε bit D1 σε κατάσταση '1' και παράλληλα τα bit D5, D3 και D2 σε '0', επιστρέφουμε όλα τα bit κατάστασης σφάλματος από τον καταχωρητή Fault Status Register. Ακολούθως, το bit D0 στον καταχωρητή RTD LSB, μπορεί να ρυθμιστεί ξανά, αμέσως μετά το reset, αν κάποιο σφάλμα υπέρτασης ή υπότασης παραμείνει. Το bit D1 επανέρχεται σε κατάσταση '0' αυτόματα. Για να επιλέξουμε την συχνότητα αποκοπής στο φίλτρο απόρριψης θορύβου και των αρμονικών, μεταξύ 50Hz ή 60Hz θα πρέπει να θέσουμε την τιμή του σε '0' ή '1' αντίστοιχα. Να αποφεύγονται οι αλλαγές στην συχνότητα του φίλτρου κατά τη διάρκεια της αυτόματης λειτουργίας.

2.3.1.2 Καταχωρητής του RTD

Σε αυτόν τον καταχωρητή των 16 bit, μπορούμε να βρούμε τα δεδομένα που αφορούν την αντίσταση του RTD αισθητήρα. Πρακτικά, ο καταχωρητής αποτελείται από δύο καταχωρητές των 8 bit. Τον RTD MSB και τον RTD LSB. Το πρώτο bit, είναι για τον έλεγχο σφάλματος. Αν είναι '0' τα περιεχόμενα το καταχωρητή αγνοούνται. Τα υπόλοιπα 15 από τα 16 bit, αφορούν την πληροφορία που μας ενδιαφέρει και είναι ο λόγος της αντίστασης του RTD προς την αντίσταση αναφοράς R_{REF}.

2.3.1.3 Καταχωρητής Fault Threshold

Ο συγκεκριμένος καταχωρητής ρυθμίζει το κατώφλι σφάλματος, στην διαδικασία ανίχνευσης σφάλματος στον αισθητήρα RTD. Τα αποτελέσματα από τις μετατροπές της τιμής του RTD, συγκρίνονται με τις τιμές αυτού του καταχωρητή για να δημιουργηθούν τα bit του σφάλματος, στον καταχωρητή Fault Status (D[7:6]). Ο συγκεκριμένος καταχωρητής έχει τα ίδια χαρακτηριστικά με των καταχωρητή του RTD.

2.3.1.4 Καταχωρητής Fault Status

Σε αυτόν τον καταχωρητή αποθηκεύονται τα bit των σφαλμάτων που μπορούν να ανιχνευθούν. Αν θέσουμε '1' το bit Fault Status Clear, στον καταχωρητή Configuration Register, ο καταχωρητής θα επιστρέψει όλα τα bit σφάλματος του σε κατάσταση '0'.

2.3.2 Μέτρηση θερμοκρασίας

Το εύρος μέτρησης της θερμοκρασίας που μας ενδιαφέρει είναι από 0 ^{o}C μέχρι 130 ^{o}C. Το κύκλωμα προσαρμογής του αναλογικού σήματος, είναι σχεδιασμένο για να λειτουργεί καλύτερα με αισθητήρες RTD, PT100 αλλά και PT1000. Επίσης το IC, υποστηρίζει και λειτουργία με αισθητήρες τύπου θερμίστορ. Για την μέτρηση της αντίστασης του PT100 έχουμε συνδέσει μία αντίσταση αναφοράς (R_{Ref}), σε σειρά με τον αισθητήρα, όπως ορίζει το σχέδιο του κυκλώματος στο φύλλο οδηγιών του ολοκληρωμένου [].

Εφαρμόζουμε την τάση πόλωσης (V_{bias}) στην R_{Ref}. Το ρεύμα που διέρχεται από την αντίσταση αναφοράς, διέρχεται επίσης και από τον αισθητήρα RTD. Η τάση στην αντίσταση αναφοράς R_{ref}, είναι η τάση αναφοράς για τον ADC. Η τάση του RTD, εφαρμόζεται στην διαφορική είσοδο RTDIN+ και RTDIN-. Έτσι ο ADC παράγει μία ψηφιακή έξοδο ίση με την αναλογία της R_{RTD} στην R_{Ref}. Για αισθητήρες RTD πλατίνας η βέλτιστη τιμή για την αντίσταση αναφοράς είναι ίση με τέσσερις φορές την τιμή της αντίστασης του RTD. Για PT100, R_{Ref} = 400Ω και για PT1000 R_{Ref} = 4kΩ.

Για την σύνδεση του αισθητήρα με το ολοκληρωμένο αν χρησιμοποιήσουμε σύνδεση δύο καλωδίων τα αποτελέσματα μέτρησης θα είναι αποδεκτά, αν το RTD μας είναι κοντά στο ολοκληρωμένο. Και αυτό γιατί όσο μεγαλύτερο είναι το μήκος των καλωδίων, τόσο μεγαλύτερο είναι και το σφάλμα μέτρησης που οφείλεται στην αντίσταση των καλωδίων. Αν συνδέσουμε μία αντίσταση 0.4 Ω σε σειρά με τον αισθητήρα μας, θα έχουμε σφάλμα μέτρησης περίπου 1 ^{o}C.

Για τον υπολογισμό της αντίστασης του καλωδίου μπορούμε να χρησιμοποιήσουμε την παρακάτω μαθηματική σχέση:

$$R = \rho * \frac{L}{A} \text{ [8]} \quad (2.4)$$

ρ: είναι η ηλεκτρική αντίσταση του υλικού (Ω x m)
L: είναι το μήκος του αγωγού σε μέτρα (m)
A: είναι το εμβαδόν της επιφάνειας διατομής του αγωγού σε τετραγωνικά μέτρα (m^2)

Για παράδειγμα, αν έχουμε αγωγό σύνδεσης PT100 με μήκος L = 1 m και διάμετρο 5mm. Η ηλεκτρική αντίσταση του χαλκού είναι $\rho = 1.68 \times 10^{-8} \Omega*m$, και επιφάνεια διατομής $A = (0,005)^2$.

$$R = 1.68 \times 10^{-8} \times (1/0.000025) = 0.000672\ \Omega.$$

Οπότε, για την περίπτωση σύνδεσης δύο αγωγών, η τιμή της αντίστασης των καλωδίων μεταξύ των σημείων σύνδεσης του αισθητήρα μέχρι και τους ακροδέκτες, θα πρέπει να συνυπολογιστεί στις μετρήσεις του συστήματος. Σε ειδικότερες περιπτώσεις συνιστάται να λαμβάνονται υπόψιν και τα θερμικά χαρακτηριστικά των καλωδίων σύνδεσης. Τα πιο σημαντικά από αυτά είναι ο συντελεστής θερμικής αγωγιμότητας, θερμική αντοχή, θερμική εκτασιμότητα, ειδική θερμότητα του υλικού, θερμική ικανότητα και η μέγιστη θερμοκρασία λειτουργίας του.

Η σύνδεση τεσσάρων καλωδίων εξαλείφει τα σφάλματα μέτρησης που προκύπτουν από την αντίσταση του καλωδίου σύνδεσης, αλλά αυξάνει το κόστος της εγκατάστασης. Η χρήση τριών αγωγών προσφέρει μία μέση λύση μεταξύ δύο και τεσσάρων αγωγών και είναι αυτή που χρησιμοποιήθηκε σε αυτήν την εφαρμογή. Για την αντιστάθμιση της πτώσης της τάσης σε όλο το καλώδιο επιστροφής, η τάση μεταξύ των ακροδεκτών *FORCE+* και *RTDIN+*, αφαιρείται από την τάση *RTDIN+* - *RTDIN-*. Αυτό επιτυγχάνεται με την χρήση της εισόδου δειγματοληψίας *FORCE2*. Αν οι αντιστάσεις των αγωγών είναι ίσες, το σφάλμα μέτρησης που οφείλεται στις αντιστάσεις των καλωδίων ακυρώνεται.

Για την επιλογή λειτουργίας σύνδεσης τριών καλωδίων, θέτουμε και το αντίστοιχο bit σε "1", στον καταχωρητή ρύθμισης παραμέτρων λειτουργίας (Configuration Register).

2.3.3 Γραμμικοποίηση των δεδομένων

Η σχέση μεταξύ θερμοκρασίας και αντίστασης στους αισθητήρες, όπως οι RTDs (Resistance Temperature Detectors) και τα θερμίστορ, δεν είναι πάντα γραμμική. Αυτό σημαίνει ότι η μεταβολή της αντίστασης δεν είναι πάντα άμεσα ανάλογη με τη μεταβολή της θερμοκρασίας. Η μη γραμμικότητα μπορεί να επηρεάσει την ακρίβεια και την ευαισθησία των μετρήσεων θερμοκρασίας. Οι αισθητήρες PT100, παρουσιάζουν μια σχετικά γραμμική σχέση μεταξύ θερμοκρασίας και αντίστασης σε ένα συγκεκριμένο εύρος θερμοκρασιών. Ωστόσο, αυτή η σχέση δεν είναι απολύτως γραμμική και μπορεί να περιγραφεί καλύτερα με τη χρήση πολυωνυμικών εξισώσεων, όπως αυτή των Callendar / Van Dusen [17].

Για εύρος μέτρησης θερμοκρασίας από -100 ºC έως 100 ºC, μία καλή προσέγγιση μπορεί να γίνει με την χρήση της παρακάτω εξίσωσης :

$$T \approx (\text{ADC Code} / 32) - 256 \text{ [17]} \quad (2.5)$$

T: Θερμοκρασία σε ºC
ADC Code: Η τιμή του μετατροπέα αναλογικού σήματος.

Η παραπάνω εξίσωση δίνει 0 ºC σφάλμα μέτρησης στους 0 ºC, -1.75 ºC σφάλμα μέτρησης στους -100 ºC και -1.4 ºC σφάλμα μέτρησης στους +100 ºC, για αισθητήρα RTD IEC751 και αντίσταση αναφοράς $R_{ref} = 4xR_{RTD}$ στους 0ºC. Για υψηλή ακρίβεια στη μέτρηση, συνιστάται η χρήση της εξίσωσης Callendar / Van Dusen ή ένας πίνακας αναζήτησης τιμών (lookup table), για την διόρθωση της προβλέψιμης μη γραμμικότητας. Η εξίσωση Callendar / Van Dusen για μέτρηση θερμοκρασίας κάτω από 0ºC:

$$R_{(T)} = R_0(1 + a * T + b * T + c * (T - 100) * T^3) \text{ [3, 17]} \quad (2.6)$$

και για άνω των 0°C

$$R_{(T)} = R_0(1+a \times T+b \times T^2) \text{ [17]} \quad (2.7)$$

T: η θερμοκρασία σε °C
$R_{(T)}$: η αντίσταση του αισθητήρα στους T °C
R_0: η αντίσταση του αισθητήρα στους 0 °C
a: ο συντελεστής θερμοκρασίας της αντίστασης, 3.90830×10^{-3}
b: -5.77500×10^{-7}
c: -4.18301×10^{-12}

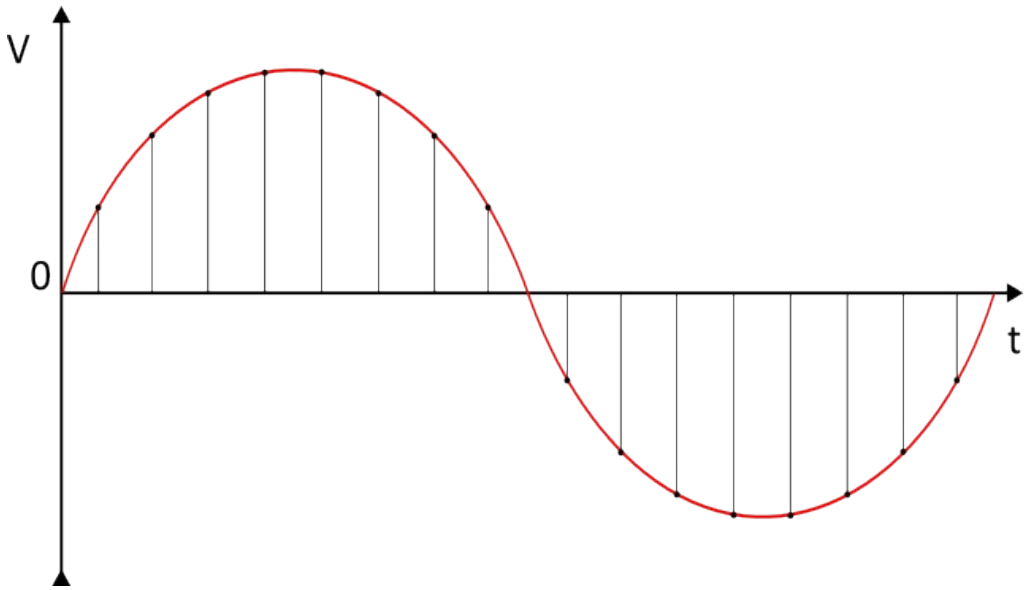
Οι τιμές των a, b, c, είναι από το πρότυπο IEC-751. Η τιμή του a, εξαρτάται από το πόσο καθαρό είναι το υλικό της πλατίνας, που χρησιμοποιήθηκε για την κατασκευή του RTD, από ακαθαρσίες και ορίζεται ως η μεταβολή της αντίστασης ανά μονάδα θερμοκρασίας.

2.3.4 Μετατροπή αναλογικού σήματος σε ψηφιακό

Οι πληροφορίες, όταν είναι σε ψηφιακή μορφή μπορούν πολύ εύκολα να επεξεργαστούν, να απεικονιστούν και να μεταδοθούν. Για να μπορέσει ένας ηλεκτρονικός υπολογιστής ή ένα ψηφιακό σύστημα να προβεί σε αυτές τις ενέργειες σε αναλογικής μορφής δεδομένα ή σήματα, θα πρέπει αυτά να μετατραπούν σε ψηφιακή μορφή. Για να πραγματοποιηθεί μία τέτοια μετατροπή, χρειάζεται ένα σύστημα μετατροπής αναλογικού σήματος σε ψηφιακό (ADC). Η μετατροπή ενός αναλογικού σήματος σε ψηφιακό περιλαμβάνει τα εξής στάδια [3, 4, 7]:

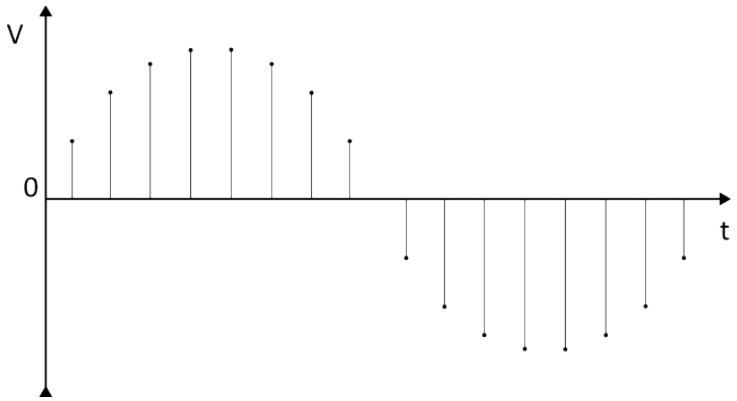
- Δειγματοληψία
- Κβαντοποίηση
- Κωδικοποίηση

Η δειγματοληψία ή ιχνηθέτηση, είναι το πρώτο στάδιο στη μετατροπή ενός αναλογικού σήματος. Το στάδιο αυτό αφορά τη λήψη δειγμάτων από το αναλογικό σήμα σε καθορισμένο χρόνο, που ονομάζεται περίοδος δειγματοληψίας (T_S) και η συχνότητα, συχνότητα δειγματοληψίας (f_S) [3, 4, 16].



Εικόνα 13: Δειγματοληψία αναλογικού σήματος

Η διαδικασία της δειγματοληψίας μετατρέπει το αναλογικό σήμα σε διακριτό σήμα. Διακριτό σήμα ονομάζεται το σήμα που έχει διακριτές τιμές σε συνεχή χρόνο. Ένα παράδειγμα σήματος διακριτού χρόνου είναι το radar [3, 7, 16].



Εικόνα 14: Διακριτό σήμα

Για να μπορέσουμε να μετατρέψουμε το ψηφιακό σήμα ξανά σε αναλογικό χωρίς να παραποιήσουμε την μορφή του από την διαδικασία μετατροπής, η συχνότητα δειγματοληψίας, σύμφωνα με το θεώρημα του C. Shannon, πρέπει να είναι ίση ή μεγαλύτερη από το διπλάσιο της μέγιστης συχνότητας f_{max} του αναλογικού σήματος [3, 4, 7]:

$$f_s \geq 2 \text{ x } f_{max} \tag{2.8}$$

Με άλλα λόγια, το θεώρημα αυτό μας λέει ότι για να αποθηκεύσουμε ή να μεταδώσουμε ένα αναλογικό σήμα χωρίς να χαθεί ή να χαθούν πληροφορίες, πρέπει να έχουμε τουλάχιστον δύο δείγματα ανά κύκλο. Από αυτή τη μαθηματική έκφραση μπορούμε να υπολογίσουμε και τη μέγιστη συχνότητα που μπορεί να αποκτηθεί από το σήμα, η συχνότητα αποκοπής ή συχνότητα Nyquist (f_N) [16]:

$$f_{max} = f_s / 2 \tag{2.9}$$

Στην διαδικασία της κβαντοποίησης, οι συνεχείς τιμές ενός σήματος περιορίζονται σε ένα πεπερασμένο σύνολο διακριτών επιπέδων. Κάθε επίπεδο αντιστοιχεί σε ένα συγκεκριμένο εύρος τιμών του αρχικού σήματος. Η κβαντοποίηση σήματος επιτρέπει τη μείωση της ποσότητας των δεδομένων που χρειάζονται για την αναπαράσταση ενός σήματος, καθιστώντας το πιο εύκολο να αποθηκευτεί ή να μεταδοθεί. Ωστόσο, συνοδεύεται και από ένα φαινόμενο γνωστό και ως "θόρυβος κβαντοποίησης", που είναι η διαφορά μεταξύ του αρχικού και του κβαντισμένου σήματος.
Τέλος στην διαδικασία της κωδικοποίησης, αντιστοιχίζεται μία δυαδική τιμή σε κάθε στάθμη που έχει υποστεί κβαντοποίηση [4, 16].

Για τη σχεδίαση ενός συστήματος μετατροπής σήματος από αναλογικό σε ψηφιακό, πρέπει να εξεταστούν προσεκτικά τα εξής χαρακτηριστικά [3, 7, 16]:

- Ακρίβεια
- Ανάλυση
- Πεδίο τιμών
- Κέρδος
- Διακριτική ικανότητα
- Σφάλμα κβαντοποίησης
- Σφάλμα κέρδους
- Σφάλμα πλήρους κλίμακας
- Διαφορική μη γραμμικότητα
- Ολοκληρωτική μη γραμμικότητα
- Ρυθμός μετατροπής

- Σφάλμα τάσης αναφοράς
- Κλίση θερμοκρασίας
- Συνολική αρμονική παραμόρφωση
- Λόγος σήματος προς θόρυβο
- Λόγος σήματος προς θόρυβο και παραμόρφωση

Ο ADC μετατροπέας έχει μία διαφορική αναλογική είσοδο, που είναι οι ακροδέκτες RTDIN+, RTDIN- και μία διαφορική είσοδο για την αντίσταση αναφοράς, στους ακροδέκτες REFIN+ και REFIN-. Η έξοδος του μετατροπέα είναι ο λόγος της τάσης στην αναλογική είσοδο, προς την τάση στην είσοδο που είναι συνδεμένη η αντίσταση αναφοράς:

$$ADC_{OUT} = V_{RTDIN}/V_{REFIN} \quad (2.10)$$

Σε περίπτωση αρνητικής τάσης εισόδου, η έξοδος του μετατροπέα είναι μηδέν. Όταν η τάση είσόδου είναι μεγαλύτερη από την τάση της αντίστασης αναφοράς V_{REFIN}, τότε θα έχουμε στην έξοδο του ADC τη μέγιστη κλίμακα (full scale output)[17].

Ένα τρίτης τάξης ψηφιακό φίλτρο εξασθενεί τον θόρυβο στην είσοδο του μετατροπέα. Θόρυβος από τα 50Hz ή 60Hz της τάσης τροφοδοσίας, συμπεριλαμβανομένων και των αρμονικών από την θεμελιώδη συχνότητα της ac ισχύος, εξασθενούν κατά 82dB.

Τα αποτελέσματα των μετατροπών του ADC μπορούν να βρεθούν στον καταχωρητή Data Register του RTD.

Οι τιμές αυτές μπορούν να μετατραπούν σε θερμοκρασία με μερικούς υπολογισμούς. Πρώτα καθορίζουμε την τιμή της αντίστασης του RTD με την παρακάτω εξίσωση:

$$R_{RTD} = (ADC\_code \times R_{REF})/2^{15} \quad (2.11)$$

ADC_code: 15 bit από τον καταχωρητή δεδομένων του RTD
R_{REF}: Η τιμή της αντίστασης αναφοράς

Μόλις γίνει γνωστή η τιμή της αντίστασης του RTD, με τις ιδιότητες του RTD που έχουμε επιλέξει, μπορούμε να υπολογίσουμε την θερμοκρασία. Αυτό μπορεί να υλοποιηθεί είτε υπολογίζοντάς την με μαθηματικό τρόπο είτε με πίνακα αναζήτησης τιμών (lookup table).

2.3.5 Ανίχνευση σφαλμάτων

Η ανίχνευση σφαλμάτων στη διαδικασία μετατροπής σήματος από αναλογικό σε ψηφιακό είναι σημαντική για την ακρίβεια και αξιοπιστία των δεδομένων. Το IC MAX31865 ανιχνεύει μία πληθώρα σφαλμάτων, τα οποία μπορούν να λάβουν χώρα με χρήση εξωτερικού αισθητήρα PT100, ανεξάρτητα από το αν έχουμε επιλέξει τη σύνδεση δύο, τριών ή τεσσάρων καλωδίων. Κάποια σφάλματα ανιχνεύονται σε κάθε μετατροπή, ενώ κάποια άλλα ανιχνεύονται μόνο όταν ζητηθεί από τη μονάδα ελέγχου να εκκινήσει ο κύκλος ανίχνευσης σφαλμάτων. Κατά τη διάρκεια αυτού του κύκλου, το ολοκληρωμένο έχει την δυνατότητα να ‘ανοίγει’ τον εσωτερικό διακόπτη στον ακροδέκτη εισόδου FORCE- από τον ακροδέκτη GND2, με τη ενεργοποίηση ενός εσωτερικού διακόπτη. Παρακάτω ακολουθούν οι συνθήκες που παράγουν σφάλματα:

- Σφάλματα που παρακολουθούνται δυναμικά καθ’ όλη τη διάρκεια της μέτρησης.

- Συνθήκες υπέρτασης ή υπότασης στους ακροδέκτες εισόδου FORCE+, FORCE2, RTDIN+, RTDIN- ή FORCE-.

- Σε κάθε μετατροπή του ADC όταν στην έξοδο έχουμε αποτέλεσμα μεγαλύτερο ή ίσο με το ανώτερο όριο μετατροπής ή αποτέλεσμα ίσο ή λιγότερο από το κατώτερο όριο μετατροπής.

- Σφάλματα που ανιχνεύθηκαν κατόπιν εντολής της μονάδας ελέγχου, στη διαδικασία του κύκλου ανίχνευσης σφαλμάτων (Fault Detection Cycle).

Έχει δύο τρόπους λειτουργίας, αυτόματη και χειροκίνητη. Στην περίπτωση που το εξωτερικό κύκλωμα διεπαφής του RTD, συμπεριλαμβάνει φίλτρο εισόδου με σταθερά χρόνου μεγαλύτερη των 100μs, ο χρονισμός της ανίχνευσης σφαλμάτων θα πρέπει να είναι στο χειροκίνητο τρόπο λειτουργίας. Για την εκκίνηση της επιθυμητής διαδικασίας ανίχνευσης σφαλμάτων, θέτουμε τα bit D3 και D2 του καταχωρητή ρυθμίσεων (Configuration Register), στην αντίστοιχη τιμή όπως δείχνει ο πίνακας 2.2.

Πίνακας 2.5

D3	**D2**	**Καταχωρητής Ρυθμίσεων**	**Εντολή**	**Ανάγνωση**
0	0	xxxx00xxb	Καμία δράση.	Ολοκλήρωση ανίχνευσης σφαλμάτων.
0	1	100x010xb	Εκκίνηση ανίχνευσης σφαλμάτων με αυτόματη καθυστέρηση.	Η διαδικασία αυτόματης ανίχνευσης σφαλμάτων εξακολουθεί να είναι ενεργή.
1	0	100x100xb	Εκκίνηση ανίχνευσης σφαλμάτων με χειροκίνητη καθυστέρηση (1ος κύκλος).	Αν η χειροκίνητη διαδικασία του 1ου κύκλου είναι ακόμα σε λειτουργία, περιμένει η συσκευή τον χρήστη να γράψει 11.
1	1	100x110xb	Ολοκλήρωση ανίχνευσης σφαλμάτων (2ος κύκλος).	Η χειροκίνητη διαδικασία του 2ου κύκλου συνεχίζει να είναι ενεργή.

Η ανίχνευση σφαλμάτων ελέγχει για τρία σφάλματα κάνοντας τις εξής συγκρίσεις τάσης, παραμετροποιώντας τα αντίστοιχα bits στον καταχωρητή Fault Status:

- Είναι η τάση στον ακροδέκτη REFIN- μεγαλύτερη από την τάση V_{BIAS} x 0.85; (Fault Status Register bit D5)
- Είναι η τάση στον ακροδέκτη REFIN- μικρότερη από την τάση V_{BIAS} x 0.85; (Fault Status Register bit D4)
- Είναι η τάση στον ακροδέκτη RTDIN- μικρότερη από την τάση V_{BIAS} x 0.85, όταν ο διακόπτης εισόδου στον ακροδέκτη FORCE- είναι ανοιχτός; (Fault Status Register bit D5)

Αφού ολοκληρωθεί ο κύκλος ανίχνευσης σφαλμάτων τα bit D3 και D2 αλλάζουν την τιμή τους σε 00 αυτόματα.

Για την έναυση του κύκλου ανίχνευσης σφαλμάτων σε χειροκίνητη λειτουργία, θα πρέπει να έχουμε ενεργοποιήσει πρώτα την έξοδο V_{BIAS}, τουλάχιστον πέντε σταθερές χρόνου. Έπειτα θέτουμε '100x100x' στον καταχωρητή Configuration Register. Ο ADC είναι σε λειτουργία 'Normally OFF'. Το MAX31865 ελέγχει για σφάλματα, όσο ο εσωτερικός διακόπτης στον ακροδέκτη FORCE-, είναι κλειστός. Με το που ολοκληρωθεί ο έλεγχος σφαλμάτων, ο διακόπτης ανοίγει. Τα bit D3 και D2 παραμένουν σε κατάσταση '10'. Επίσης θα χρειαστεί αναμονή πέντε σταθερών χρόνου και έπειτα γράφουμε στον καταχωρητή Configuration Register '100x110x'. Το IC MAX31865 τώρα ελέγχει για σφάλματα καθώς ο εσωτερικός διακόπτης του ακροδέκτη FORCE- είναι ανοιχτός. Όταν ο έλεγχος ολοκληρωθεί, ο διακόπτης στην είσοδο FORCE-, κλείνει και τα bits D3 και D2, του κύκλου ανίχνευσης σφαλμάτων παίρνουν την τιμή '00' αυτόματα. Στην περίπτωση που γράψουμε '1' στο bit D5 (λειτουργία 1-shot), και τα bit D3 και D2, σε μία εγγραφή, οι εντολές αγνοούνται.

Αν θέσουμε στον καταχωρητή '100x110x', χωρίς την αρχικοποίηση της χειροκίνητης διαδικασίας, θα ενεργοποιηθεί η αυτόματη διαδικασία. Να αναφέρουμε επίσης ότι οι είσοδοι FORCE+, FORCE2,

FORCE-, RTDIN+ και RTDIN-, έχουν προστασία από υπερτάσεις μέχρι ±45V. Σήματα που εφαρμόζονται σε αυτούς τους ακροδέκτες, προστατεύονται από αναλογικούς διακόπτες οι οποίοι είναι ανοιχτοί, όταν η εφαρμοζόμενη τάση είναι μεγαλύτερη από την τάση V_{DD} + 100mV ή λιγότερη από GND1 – 400mV.

Όταν έχουμε σφάλμα που οφείλεται στην τάση, τα κυκλώματα προστασίας θα επιτρέψουν ροή ρεύματος μέχρι 350 μA περίπου. Αυτό το ρεύμα διαρροής δεν προκαλεί βλάβη στο ολοκληρωμένο.
Σε συνθήκες υπέρτασης ή υπότασης, το bit D2 του καταχωρητή Fault Status παίρνει την τιμή '1' και ο ADC σταματάει την λειτουργία του μέχρι το σφάλμα σταματήσει να ανιχνεύεται. Έπειτα θα συνεχιστεί η λειτουργία του ADC.

Στην συνδεσμολογία τριών ή τεσσάρων καλωδίων, μία κομμένη σύνδεση ή ένα αποσυνδεμένο καλώδιο, έχει ως αποτέλεσμα την μη πολωμένη θετική είσοδο του ADC. Αυτό έχει ως συνέπεια απρόβλεπτα αποτελέσματα στις μετατροπές σήματος του ADC. Μεγάλη επιρροή σε αυτά μπορούν να έχουν ο εξωτερικός θόρυβος, η σχεδίαση της πλακέτας και η θερμοκρασία του χώρου λειτουργίας. Αυτή η κατάσταση σφάλματος μπορεί να περάσει απαρατήρητη λόγω της τιμής ρύθμισης του κατωφλίου σφάλματος, που έχει ρυθμιστεί στον καταχωρητή Fault Threshold Register. Για τον έλεγχο αυτού του σφάλματος, προσθέτουμε μία αντίσταση 10MOhm από την είσοδο του RTDIN+ στον ακροδέκτη BIAS. Αυτό έχει ως αποτέλεσμα, τη μέτρηση της αντίστασης του RTD, σε μέγιστη κλίμακα, στην περίπτωση που η σύνδεση του RTDIN+ έχει σφάλμα.

Η ανίχνευση σφαλμάτων σε περιπτώσεις ανοιχτού ή βραχυκυκλωμένου στοιχείου RTD, ανιχνεύεται σε κάθε μετατροπή με βάση τα δεδομένα της αντίστασης. Σε ένα 'ανοιχτό' στοιχείο RTD, συνήθως το αποτέλεσμα είναι η μέτρηση πλήρους κλίμακας. Για την ανίχνευση αυτού του σφάλματος, θα χρειαστεί να ρυθμίσουμε το κατώφλι ανίχνευσης σφάλματος, στον καταχωρητή High Fault Threshold Register. Αν το αποτέλεσμα της μετατροπής είναι μεγαλύτερο από ή ίσο με την τιμή του κατωφλίου, το υψηλό bit του RTD στον καταχωρητή Fault Status Register, θέτεται στο τέλος του κύκλου μετατροπής του ADC. Για την ανίχνευση σφάλματος επίσης ανοιχτού RTD στοιχείου κατά απαίτηση, ελέγχουμε $V_{REFIN-} > V_{BIAS}$ x 0.85. Ένα βραχυκυκλωμένο RTD στοιχείο μας δίνει αποτέλεσμα κοντά στο μηδέν. Η ρύθμιση του κατωφλίου για αυτή τη περίπτωση, λαμβάνει χώρα στον καταχωρητή Low Fault Threshold Register.

Τα bit κατάστασης σφάλματος μανδαλώνονται έως ότου θέσουμε το bit Fault Clear, στον καταχωρητή Configuration Register. Αυτό μας επιτρέπει να ανιχνεύσουμε διακοπτόμενα, μή επαναλαμβανόμενα σφάλματα.

2.4 Επικοινωνία SPI

Το SPI (Serial Peripheral Interface) είναι ένα πρωτόκολλο σειριακής επικοινωνίας. Χρησιμοποιείται κυρίως στα ενσωματωμένα συστήματα, για την αποστολή δεδομένων μεταξύ ενός κεντρικού επεξεργαστή και των περιφερειακών συσκευών, όπως αισθητήρες, οθόνες, και μνήμες.
Τα βασικά χαρακτηριστικά του SPI είναι:

- Αρχιτεκτονική master-slave: Το SPI χρησιμοποιεί μια κεντρική συσκευή (master) που ελέγχει την επικοινωνία με μία ή περισσότερες υποδεέστερες συσκευές (slaves).

- Χρησιμοποιεί τέσσερις κύριες γραμμές:
  - SCLK (Serial Clock): Σειριακό ρολόι που παράγεται από τον master.
  - SDI (Serial Data Input): Σειριακή είσοδος, τα δεδομένα έχουν ροή από την συσκευή master προς την συσκευή slave.
  - SDO (Serial Data Output): Σειριακή έξοδος, τα δεδομένα έχουν ροή από την συσκευή slave προς την συσκευή master.
  - CS (Chip Select): Γραμμή επιλογής slave συσκευής που ενεργοποιεί τον αντίστοιχο slave.

- Δυνατότητα Πολλαπλών slaves: Ο master μπορεί να επικοινωνήσει με πολλούς slaves, καθορίζοντας ποιος slave θα είναι ενεργός κάθε φορά μέσω της γραμμής CS.

Το SPI είναι γνωστό για την υψηλή του ταχύτητα και την ευελιξία του, καθιστώντας το ιδανικό για εφαρμογές που απαιτούν γρήγορη και αξιόπιστη μετάδοση δεδομένων.

2.4.1 SDI

Ο SDI (Serial Data In), είναι ένας από τους ακροδέκτες ή γραμμές που χρησιμοποιούνται στο πρωτόκολλο SPI (Serial Peripheral Interface). Είναι ο ακροδέκτης στον οποίο ο slave λαμβάνει δεδομένα από τον master. Εναλλακτικά, αυτός ο ακροδέκτης μπορεί να αναφέρεται και ως MOSI (Master Out Slave In), όταν μιλάμε από την πλευρά του master. Το SDI αποτελεί κρίσιμη γραμμή για την μετάδοση δεδομένων σε συστήματα SPI, καθιστώντας την επικοινωνία γρήγορη και αξιόπιστη.

2.4.2 SDO

Ο ακροδέκτης SDO (Serial Data Out) είναι μία από τις γραμμές στο πρωτόκολλο SPI (Serial Peripheral Interface). Χρησιμοποιείται από την slave συσκευή για την αποστολή δεδομένων πίσω στην master. Από την πλευρά της master, αυτός ο ακροδέκτης αναφέρεται συχνά ως MISO (Master In Slave Out). Η SDO είναι κρίσιμη για την μετάδοση δεδομένων από την slave στην master συσκευή, επιτρέποντας την αμφίδρομη επικοινωνία στο πρωτόκολλο SPI.

2.4.3 CS

Το CS (Chip Select), επίσης γνωστό ως SS (Slave Select), είναι μια από τις βασικές γραμμές στο πρωτόκολλο SPI (Serial Peripheral Interface).
Λειτουργία του CS:

- Επιλογή slave: Το CS χρησιμοποιείται από την κύρια συσκευή (master) για να επιλέξει ποια υποδεέστερη συσκευή (slave) θα είναι ενεργή και θα επικοινωνήσει.

- Ενεργοποίηση: Η γραμμή CS είναι συνήθως ενεργή σε κατάσταση ‘LOW’, που σημαίνει ότι η υποδεέστερη συσκευή επιλέγεται όταν το σήμα CS είναι στο λογικό 0.

- Αποτροπή Συγκρούσεων: Η γραμμή CS βοηθά να αποτραπεί η σύγκρουση δεδομένων μεταξύ πολλών υποδεέστερων συσκευών, εξασφαλίζοντας ότι μόνο μία συσκευή είναι ενεργή τη φορά.

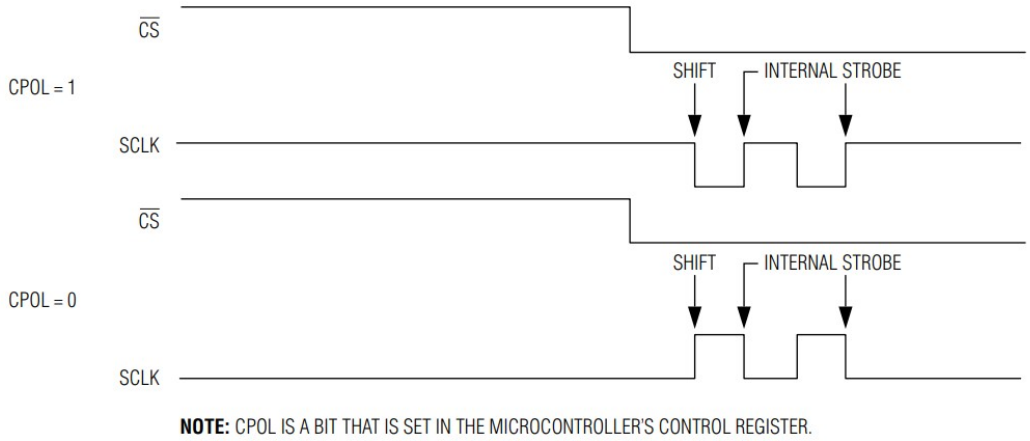
Ρόλος του CS στην Επικοινωνία SPI:

- Αρχή και Τέλος Επικοινωνίας: Όταν η master συσκευή θέλει να επικοινωνήσει με μία συγκεκριμένη slave συσκευή, θέτει την γραμμή CS σε λογικό ‘0’. Η επικοινωνία ξεκινά και συνεχίζεται μέχρι η γραμμή CS να επανέλθει στο λογικό ‘1’.

- Μέγιστη Συμβατότητα: Η χρήση του CS επιτρέπει στη master συσκευή να ελέγχει πολλαπλές υποδεέστερες συσκευές χωρίς προβλήματα σύγκρουσης, αυξάνοντας την ευελιξία και τη λειτουργικότητα του συστήματος.

2.4.4 SCLK

Η SCLK (Serial Clock) είναι μία από τις βασικές γραμμές στο πρωτόκολλο SPI (Serial Peripheral Interface). Είναι το ρολόι που παράγεται από την κύρια συσκευή (master) και συγχρονίζει τη μεταφορά δεδομένων μεταξύ του master και των περιφερειακών συσκευών (slaves). Όταν το ρολόι κάνει ένα κύκλο ή ‘χτύπο’, τα δεδομένα αποστέλλονται ή λαμβάνονται από τις slave συσκευές SPI. Αυτό σημαίνει ότι κάθε bit δεδομένων αντιστοιχεί σε έναν χτύπο του SCLK. Η διεύθυνση διαδρομής δεδομένων (Data Address Path), εξασφαλίζει ότι τα δεδομένα αποστέλλονται και λαμβάνονται ταυτόχρονα από τις συσκευές, αποτρέποντας προβλήματα έλλειψης συγχρονισμού. Το SCLK αποτελεί βασικό στοιχείο του πρωτοκόλλου SPI, καθορίζοντας τον ρυθμό με τον οποίο τα δεδομένα μεταφέρονται μεταξύ των συσκευών και είναι ενεργό μόνο όταν ο ακροδέκτης CS, είναι σε κατάσταση ‘LOW’, κατά την διαδικασία διευθυνσιοδότησης ή μεταφοράς δεδομένων σε οποιαδήποτε συσκευή είναι συνδεμένη στο δίκτυο SPI.



Εικόνα 15: Το σειριακό ρολόι ως συνάρτηση της πολικότητας του ρολογιού του μικροελεγκτή [17].

Η πολικότητα του ανενεργού ρολογιού, είναι προγραμματιζόμενη σε μερικούς μικροελεγκτές. Το MAX31865, όταν ο ακροδέκτης CS, είναι ‘HIGH’, παίρνει δείγμα από το SCLK με σκοπό να ρυθμίσει την πολικότητα του ρολογιού, όταν ο CS γίνει ‘LOW’. Τα δεδομένα εισόδου μανδαλώνονται στην πρώτη αιχμή του παλμού του ρολογιού και τα δεδομένα εξόδου στην δεύτερη αιχμή. Η πολικότητα του ρολογιού είναι είτε ‘0’ είτε ‘1’. Για κάθε ένα bit δεδομένων που μεταφέρεται υπάρχει και ένας παλμός του ρολογιού. Οι διευθύνσεις και τα bit δεδομένων μεταφέρονται σε ομάδες των 8 bit, με τα περισσότερο σημαντικά (MSB) στην αρχή.

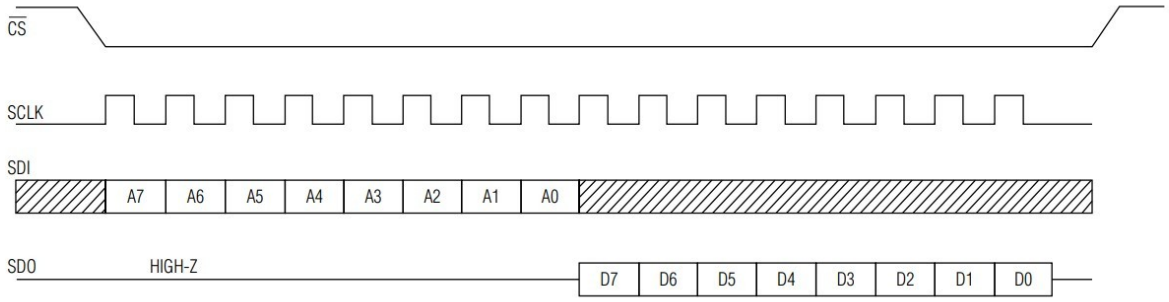
Πίνακας 2.

MODE	CS	SCLK	SDI	SDO
Απενεργοποιημένο Reset	High	Απενεργοποιημένη είσοδος	Απενεργοποιημένη είσοδος	Υψηλή σύνθετη αντίσταση
Εγγραφή	Low	CPOL=1, SCLK rising	Μανδάλωση των bit δεδομένων	Υψηλή σύνθετη αντίσταση
	Low	CPOL=0, SCLK falling		
Ανάγνωση	Low	CPOL=1, SCLK falling	X	Μετατόπιση bit δεδομένων
	Low	CPOL=0, SCLK rising		

2.4.5 Διευθύνσεις και bit δεδομένων

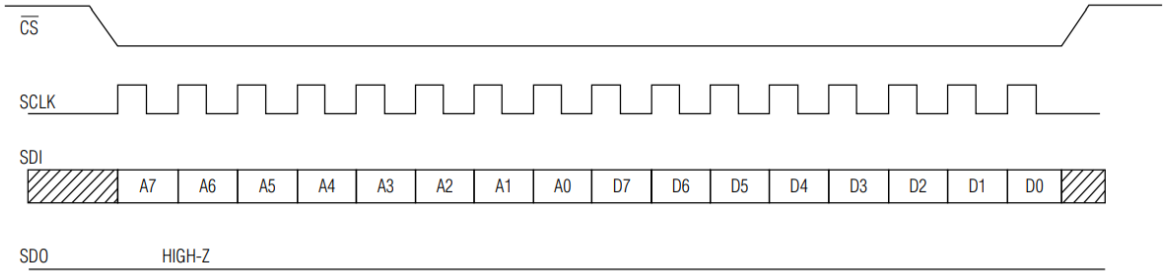
Οι διευθύνσεις και τα bit δεδομένων μεταδίδονται στην είσοδο (SDI) και έξοδο (SDO) με πρώτο το πιο σημαντικό ψηφίο (MSB). Κάθε μεταφορά δεδομένων χρειάζεται την διεύθυνση του byte που απευθύνεται και την διευκρίνηση αν πρόκειται για ενέργεια εγγραφής ή ανάγνωσης. Τέλος ακολουθεί ένα ή περισσότερα byte δεδομένων. Τα δεδομένα που μεταφέρονται από τον ακροδέκτη SDO είναι για ανάγνωση και από τον ακροδέκτη SDI για εγγραφή.

Το byte της διεύθυνσης του καταχωρητή είναι πάντα το πρώτο byte που μεταδίδεται, αμέσως μόλις ο ακροδέκτης CS, τεθεί σε κατάσταση ‘LOW’. Το πιο σημαντικό bit, το A7 από αυτό το byte, καθορίζει αν το ακόλουθο byte είναι για ανάγνωση ή εγγραφή. Εάν το bit A7 είναι ‘0’, μία ή περισσότερες ενέργειες ανάγνωσης ακολουθούν το byte διεύθυνσης. Εάν το bit A7 είναι ‘1’, μία ή περισσότερες ενέργειες εγγραφής ακολουθούν το byte διεύθυνσης. Για μεταφορά ενός μόνο byte, ένα byte εγγράφεται ή διαβάζεται και έπειτα ο ακροδέκτης CS πηγαίνει σε κατάσταση ‘HIGH’.



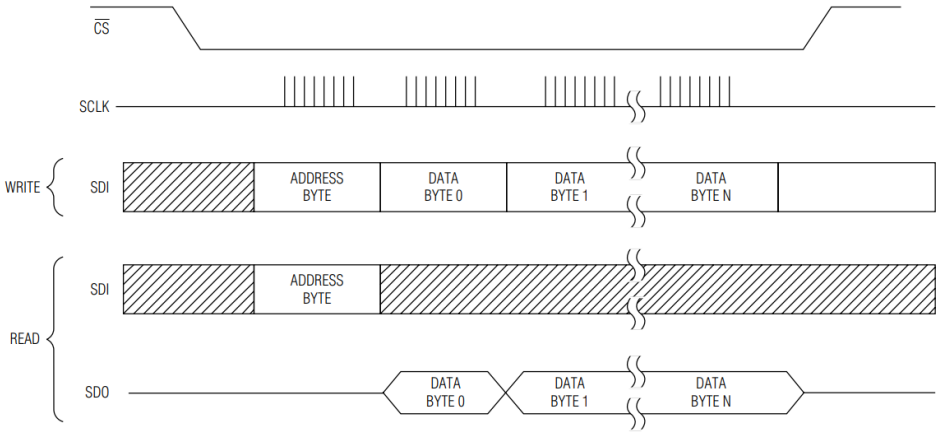
Εικόνα 16: Ανάγνωση ενώς byte [17].

Είναι επίσης εφικτή και η μετάδοση πολλών byte είτε για ανάγνωση είτε για εγγραφή, αφού γίνει πρώτα η εγγραφή του byte διεύθυνσης. Όσο ο ακροδέκτης CS παραμένει σε κατάσαση 'LOW', ο αριθμός της διεύθυνσης αυξάνεται σε όλες τις διαθέσιμες θέσεις της μνήμης.



Εικόνα 17: Εγγραφή ενός byte [17].

Αν η ροή των δεδομένων είτε προς την είσοδο είτε προς την έξοδο, είναι εντός ή εκτός χρονισμού, η διεύθυνση επιστρέφει από την θέση '7Fh/FFh' στη θέση '00h/80h'. Σε περιστατικό εσφαλμένης θέσης μνήμης, επιστρέφει η τιμή 'FFh'. Όποια προσπάθεια εγγραφής λάβει χώρα σε καταχωρητή που είναι μόνο για ανάγνωση, τα περιεχόμενα του μένουν αναλλοίωτα.



Εικόνα 18: Μετάδοση πολλών bytes [17].

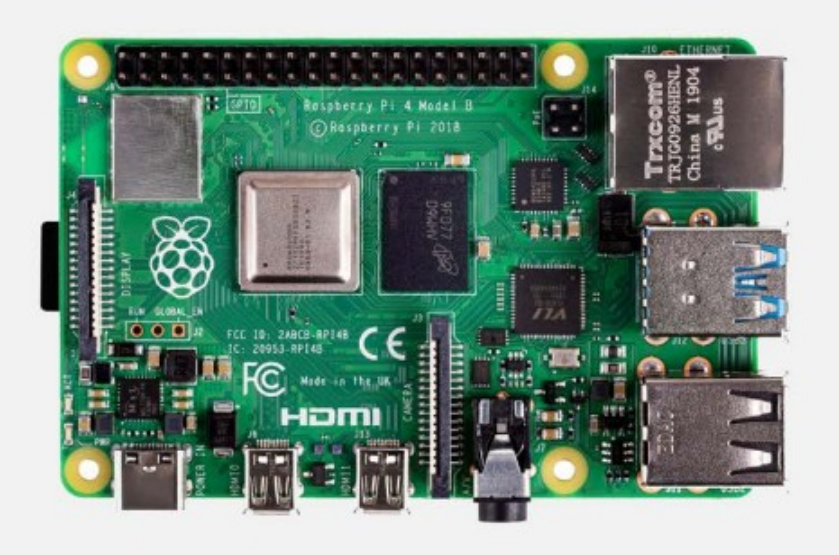
2.5 Ο μικροελεγκτής

Οι μικροελεγκτές και τα ενσωματωμένα συστήματα αποτελούν βασικά στοιχεία την σημερινή εποχή. Οι μικροελεγκτές είναι υπολογιστές μικρού μεγέθους που συμπεριλαμβάνουν επεξεργαστή, μνήμη

RAM και ROM, εισόδους και εξόδους. Μπορούμε να τους συναντήσουμε σε οικιακές ηλεκτρικές συσκευές, οχήματα, ιατρικό εξοπλισμό, καθώς επίσης σε βιομηχανικό και στρατιωτικό εξοπλισμό. Τα ενσωματωμένα συστήματα σχεδιάζονται συνήθως για να εκτελούν ένα συγκεκριμένο σκοπό κυρίως σε συνθήκες που απαιτούν αντίδραση σε πραγματικό χρόνο.

Ο μικροελεγκτής που επιλέχθηκε ως κεντρική μονάδα ελέγχου, στο παρόν πείραμα, είναι ένα Raspberry Pi 4, Model B. Το Raspberry Pi 4 είναι ο τέταρτος κατά σειρά υπολογιστής της σειράς Raspberry Pi, που δημιουργήθηκε από το ίδρυμα Raspberry Pi, για να προσφέρει φθηνές και ισχυρές λύσεις στον τομέα των υπολογιστών. Το συγκεκριμένο μοντέλο είναι μία ευέλικτη και ισχυρή συσκευή που μπορεί να χρησιμοποιηθεί για πολλούς σκοπούς, όπως η εκτέλεση προγραμμάτων με χειροκίνητη επαφή, χρήση ως προσωπικός Η/Υ ή για την διαχείριση ενός συστήματος. Τα τεχνικά χαρακτηριστικά του Raspberry Pi 4 Model B είναι τα εξής:

- Quad core 64-bit ARM-Cortex A72, 1.5GHz
- 4 GB LPDDR4 RAM
- 802.11 b/g/n/ac Wireless LAN
- 1x SD Card 32GB
- 1x Gigabit Ethernet port
- Bluetooth 5.0



Εικόνα 19: Raspberry Pi 4, Model B. [30]

Η εφαρμογή που είναι υπεύθυνη για την λειτουργία του συστήματος έχει γραφτεί στην γλώσσα Python. Επιπρόσθετες βιβλιοθήκες έχουν εγκατασταθεί και χρησιμοποιούνται στο σύστημα και φορούν την λειτουργία των περιφερειακών συσκευών καθώς επίσης και την επικοινωνία του συστήματος με τον έξω κόσμο. Παρακάτω παραθέτονται σε αλφαβητική σειρά οι βιβλιοθήκες που έχουν χρησιμοποιηθεί:

- *adafruit_max31865*: η βιβλιοθήκη adafruit_max31865 επιτρέπει την διαχείριση του ολοκληρωμένου κυκλώματος MAX31865, Μας επιτρέπει την ανάγνωση και τον προγραμματισμό τον λειτουργιών του. Περιέχει συναρτήσεις που είναι χρήσιμες για την ανάπτυξη εφαρμογών που απαιτούν ακρίβεια στην μέτρηση της θερμοκρασίας.
  https://docs.circuitpython.org/projects/max31865/en/latest/

- *board*: η παρούσα είναι μέρος της CircuitPython της Adafruit Blinka. Επιτρέπει την πρόσβαση και διαχείριση των I/O pins των μικροελεγκτών και SBCs όπως το Raspberry Pi. Παρέχει έναν απλοποιημένο και ενιαίο μηχανισμό διαχείρισης των pins, ενώ παραμένει πλήρως συμβατή με άλλες πλατφόρμες μικροελεγκτών.
  https://docs.circuitpython.org/en/latest/shared-bindings/board/index.html

- *digitalio*: και αυτή η βιβλιοθήκη όπως η προηγούμενη, αποτελεί μέρος του οικοσυστήματος της CircuitPython της Adafuit. Χρησιμοποιείται για την διαχείριση των ψηφιακών I/O pins. Επιτρέπει την ρύθμιση ως ψηφιακές εισόδους ή εξόδους και αντίστοιχα την ανάγνωση η εγγραφή.
  https://docs.circuitpython.org/en/latest/shared-bindings/digitalio/

- *gpiozero*: η βιβλιοθήκη αυτή διευκολύνει τη διαχείριση και τον έλεγχο των GPIO pins του Raspberry Pi. Παρέχει μία απλή διεπαφή για την εκτέλεση διεργασιών όπως είναι η ανάγνωση από αισθητήρες, η οδήγηση συσκευών και εξόδων.
  - *LED*: η κλάση αυτή επιτρέπει τον έλεγχο LED που είναι συνδεδεμένα στην GPIO του RaspberryPI.
  - *Button*: επιτρέπει την ανίχνευση της κατάστασης ενός κουμπιού που είναι συνδεδεμένο στην GPIO του Raspberry Pi.
  - *CPUTemperature*: επιτρέπει να παρακολουθούμε την θερμοκρασία του επεξεργαστή στο Raspberry Pi.
  - *RotaryEncoder*: χρησιμοποιείται για την ανάγνωση και διαχείριση δεδομένων από rotary encoders.

  https://gpiozero.readthedocs.io/en/latest/

- *json*: η συγκεκριμένη βιβλιοθήκη είναι ένα εργαλείο που επιτρέπει την διαχείριση δεδομένων τύπου JSON. Ο τύπος JSON είναι μία μορφή δεδομένων που είναι ευκολη για ανάγνωση και επεξεργασία από ανθρώπους και υπολογιστές. Παρέχει συναρτήσεις για τη μετατροπή αντικειμένων της Python σε μορφή JSON και αντίστροφα.
  https://docs.python.org/3/library/json.html

- *os*: είναι η διεπαφή για την αλληλεπίδραση με το λειτουργικό σύστημα. Είναι μέρος της standard βιβλιοθήκης της Python και προσφέρει μία πληθώρα συναρτήσεων που επιτρέπουν την εκτέλεση λειτουργιών όπως είναι η διαχείριση αρχείων και φακέλων, η εκτέλεση εντολών του συστήματος και η ανάκτηση πληροφοριών για το περιβάλλον του.
  https://docs.python.org/3/library/os.html

- *RPLCD.i2c*: επιτρέπει την εύκολη διαχείριση LCD οθονών που χρησιμοποιούν το ολοκληρωμένο Hitachi HD44780 μέσω του I^2C bus. Είναι ιδανική για χρήση με Raspberry Pi ή αλλά ενσωματωμένα συστήματα που υποστηρίζουν το πρωτόκολλο επικοινωνίας I^2C.
  - *CharLCD*: η κλάση αυτή μας δίνει τη δυνατότητα να γράψουμε κείμενο στην LCD οθόνη, να ρυθμίσουμε τη θέση του κέρσορα, να καθαρίσουμε την οθόνη και πολλές άλλες λειτουργίες.

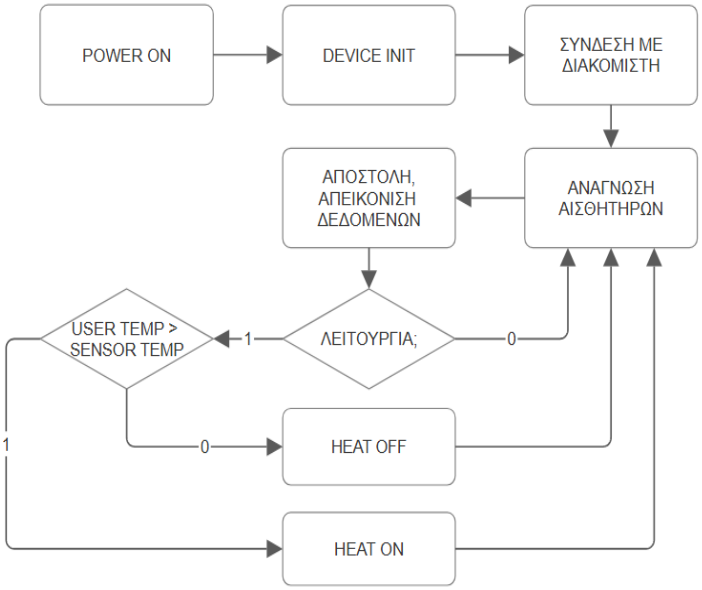
  https://rplcd.readthedocs.io/en/stable/index.html

- *python-socketio*: η βιβλιοθήκη αυτή είναι ένα πανίσχυρο εργαλείο για την ανάπτυξη εφαρμογών που απαιτούν επικοινωνία σε πραγματικό χρόνο. Προσφέρει υποστήριξη τόσο για τον διακομιστή όσο και για τον πελάτη, επιτρέποντας τη δημιουργία αμφίδρομης επικοινωνίας μέσω δικτύου. https://python-socketio.readthedocs.io

- *schedule*: η συγκεκριμένη βιβλιοθήκη μας επιτρέπει να προγραμματίζουμε και να εκτελούμε λειτουργίες σε συγκεκριμένα χρονικά διαστήματα ή στιγμές. Η σύνταξη της βιβλιοθήκης είναι απλή και κατανοητή, καθιστώντας την εύκολη στη χρήση.
  https://schedule.readthedocs.io/

- *threading*: η βιβλιοθήκη αυτή παρέχει εργαλεία για τη δημιουργία και διαχείρισή νημάτων (threads). Η χρήση νημάτων είναι σημαντική γιατί επιτρέπει την παράλληλη εκτέλεση πολλαπλών τμημάτων κώδικα, βελτιώνοντας την απόδοση των εφαρμογών που περιλαμβάνουν παραλληλισμό και ασύγχρονη εκτέλεση εργασιών.
  https://docs.python.org/3/library/threading.html#

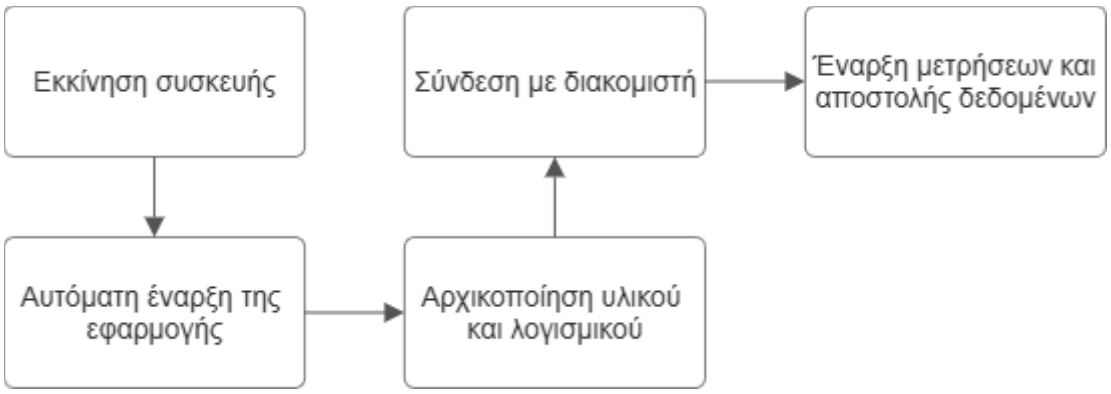
- *time*: η time παρέχει μία σειρά από συναρτήσεις για τον χειρισμό του χρόνου και των ημερομηνιών. Είναι ενσωματωμένη στην βασική βιβλιοθήκη της γλώσσας Python και προσφέρει χρήσιμες συναρτήσεις για τον χειρισμό χρονικών δεδομένων όπως είναι η ανάκτηση της τρέχουσας ώρας και ημερομηνίας και τη ρύθμιση καθυστερήσεων.
  https://docs.python.org/3/library/time.html



Εικόνα 20: Το μπλόκ διάγραμμα του προγράμματος λειτουργίας.

2.5.1 Περιγραφή λειτουργιών

Η έναρξη λειτουργίας της εφαρμογής πραγματοποιείται αυτόματα κατά την εκκίνηση του Raspberry Pi, μέσω της εντολής "*python3 client.py &*" που προστέθηκε στο αρχείο συστήματος rc.local, του λειτουργικού συστήματος linux, που τρέχει στον μικροελεγκτή.



Εικόνα 21: Η διαδικασίας εκκίνησης του μικροελεγκτή.

Είναι μία σύνθετη εργασία και περιλαμβάνει τη ρύθμιση του λογισμικού καθώς επίσης και του υλικού και περιλαμβάνει τις παρακάτω λειτουργίες:

- Αρχικοποίηση των παραμέτρων και των μεταβλητών του προγράμματος. Αυτή η διαδικασία περιλαμβάνει την ενσωμάτωση των πρόσθετων βιβλιοθηκών στην εφαρμογή, την αρχικοποίηση των μεταβλητών της εφαρμογής, την παραμετροποίηση των εισόδων και εξόδων της θύρας GPIO του μικροελεγκτή, της διεπαφής του χρήστη και των επικοινωνιών.

- Αρχικοποίηση της διεπαφής του αισθητήρα θερμοκρασίας και έναρξη της διαδικασίας λήψης μετρήσεων. Κατά την εκκίνηση του συστήματος, πραγματοποιείται αρχικοποίηση του αισθητήρα θερμοκρασίας. Αυτό περιλαμβάνει την ρύθμιση των παραμέτρων λειτουργίας του MAX31865 και την ενεργοποίηση της διαδικασίας συνεχούς παρακολούθησης και καταγραφής της θερμοκρασίας.

```
# Sensor A initialization
sensorA = adafruit_max31865.MAX31865(spi,
                                     csA,
                                     rtd_nominal=100,
                                     ref_resistor=470.6,
                                     wires=3)
# Sensor B initialization
sensorB = adafruit_max31865.MAX31865(spi,
                                     csB,
                                     rtd_nominal=100,
                                     ref_resistor=470.0,
                                     wires=3)
```

Εικόνα 22: Αρχικοποίηση του MAX31865.

- Σύνδεση με τον διακομιστή με WebSockets, αποστολή και λήψη δεδομένων: Η λειτουργία αυτή επιτρέπει την δημιουργία μιας αμφίδρομης επικοινωνίας μεταξύ της συσκευής και του διακομιστή, χρησιμοποιώντας το πρωτόκολλο WebSockets. Με τη μέθοδο αυτή, η συσκευή μπορεί να στέλνει δεδομένα μετρήσεων ή αιτήματα στον διακομιστή, καθώς και να λαμβάνει πληροφορίες ή οδηγίες από αυτόν σε πραγματικό χρόνο.

```
HOST = '192.168.1.100' #set hostname
PORT = 3000 #set port number
DEVICE_NAME = 'DEV001' #set device name
CONNECTION_QUERY = f'http://{str(HOST)}:{str(PORT)}' #connection string
```

Εικόνα 23: Τα στοιχεία για την σύνδεση με τον διακομιστή.

```
def connect():
    global connection
    try:
        print(f'Connecting to server... {str(HOST)}:{str(PORT)}')
        connection = mysocket.connect(CONNECTION_QUERY)
        print(CONNECTION_QUERY)
    except:
        print(f'Connection Error.')
        connection = '0'
        return connection
    else:
        print(f'Connected to server {str(HOST)}:{str(PORT)}')
        return connection
```

Εικόνα 24: Σύνδεση με τον διακομιστή.

- Αρχικοποίηση της οθόνης απεικόνισης των πληροφοριών και εμφάνιση πληροφοριών. Η συσκευή ενσωματώνει μία οθόνη που ενεργοποιείται με την εκκίνηση. Στην οθόνη αυτή εμφανίζονται κρίσιμες πληροφορίες, όπως οι μετρήσεις της θερμοκρασίας, η κατάσταση των αισθητήρων και μηνύματα του συστήματος.

```
lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1,
              cols=20, rows=4, dotsize=10, auto_linebreaks=False,
              backlight_enabled=True)
lcd.cursor_mode = 'hide'
```

Εικόνα 25: Αρχικοποίηση οθόνης LCD.

- Έλεγχος της κατάστασης του διακόπτη εκτάκτου ανάγκης. Το σύστημα πραγματοποιεί έλεγχο του διακόπτη εκτάκτου ανάγκης για να διασφαλίσει ότι βρίσκεται σε λειτουργική κατάσταση. Ο διακόπτης χρησιμοποιείται για άμεση απενεργοποίηση του συστήματος σε περίπτωση ανάγκης και την οπτική ένδειξη της κατάστασης με την ενεργοποίηση της λυχνίας ένδειξης εκτάκτου ανάγκης.

```
def emgPressed():
    global emgFlag, menuFlag
    emgFlag = True
    menuFlag = False
    led2.blink(on_time=1, off_time=1)
    stopBtnPressed()
    print(f'_____Emergency Button pressed!')

emgButton = gpiozero.Button(12, bounce_time = bounceTime)
# EMERGENCY button condition NC
emgButton.when_released = emgPressed
```

Εικόνα 26: Μπουτόν εκτάκτου ανάγκης.

- Έλεγχος κατάστασης αισθητήρα στάθμης. Ενσωματωμένος αισθητήρας στάθμης ελέγχει το επίπεδο του περιεχομένου της δεξαμενής. Τα δεδομένα του παρέχουν ενημέρωση στο χρήστη και μπορούν να ενεργοποιήσουν την παύσης της διαδικασίας πλήρωσης, εάν οι στάθμη ξεπεράσει τα καθορισμένα όρια.

```
def levelMax():
    global maxLevel, levelStr
    if levelSensor.is_active is True:
        led4.blink(on_time=0.5, off_time=0.5)
        maxLevel = True
        levelStr = 'MAX'
    elif levelSensor.is_active is False:
        led4.off()
        maxLevel = False
        levelStr = 'OK'

levelSensor = gpiozero.Button(22)
levelSensor.when_pressed = levelMax
levelSensor.when_released = levelMax
```

Εικόνα 27: Έλεγχος της κατάστασης του αισθητήρα στάθμης.

- Ρύθμιση της θερμοκρασίας από τον χρήστη. Ο χρήστης έχει τη δυνατότητα να ορίζει τα επιθυμητά επίπεδα θερμοκρασίας μέσω μιας φιλικής προς το χρήστη διεπαφής, τον διακόπτη rotary encoder. Το σύστημα προσαρμόζει τις λειτουργίες του ώστε να ανταποκριθεί στις ρυθμίσεις αυτές.

- Αυτόματος έλεγχος ηλεκτρονόμου. Η συσκευή ελέγχει τη λειτουργία του ηλεκτρονόμου, συγκρίνοντας τη μετρούμενη θερμοκρασία, με την θερμοκρασία ρύθμισης του χρήστη. Ο ηλεκτρονόμος λειτουργεί ως διακόπτης, για την τροφοδοσία του συστήματος θέρμανσης της δεξαμενής. Στη σύγκριση της θερμοκρασίας έχει προστεθεί 0.5 °C υστέρηση.

```
def checkTemp():
    # if (float(tempA) + 0.5 ) <= newTemp and led1.is_active == True:
    if (float(tempA) + 0.5 ) >= newTemp and startStatus == True:
        output1.off() # Relay 1
        # output2.off() # Optional Relay 2
    # elif (float(tempA) - 0.5 ) >= newTemp and led1.is_active == True:
    elif (float(tempA) - 0.5 ) <= newTemp and startStatus == True:
        output1.on() # Relay 1
        # output2.on() # Optional Relay 2
```

Εικόνα 28: Σύγκριση της θερμοκρασίας και έλεγχος της λειτουργίας του ηλεκτρονόμου.

- Μενού επιλογών συσκευής. Το μενού παρέχει επιλογές για τη σύνδεση με το διακομιστή, διασφαλίζοντας την άμεση και αποτελεσματική επικοινωνία με το δίκτυο. Επίσης προσφέρει μία σαφή διαδικασία για τον τερματισμό της λειτουργίας συσκευής, εξασφαλίζοντας την ομαλή και ασφαλή απενεργοποίηση της. Η διαδικασία αυτή είναι αναγκαία για την προστασία των δεδομένων και των πόρων της συσκευής από πιθανές βλάβες που μπορεί να προκύψουν από απότομο τερματισμό.

2.6 Το πληροφοριακό σύστημα

Σύμφωνα με το εκπαιδευτικό υλικό της Microsoft στην πλατφόρμα Microsoft Learn [21], τα πληροφοριακά συστήματα είναι συστήματα που συνδυάζουν τεχνολογία, ανθρώπους και διαδικασίες για να συλλέγουν, αποθηκεύουν, διαχειρίζονται και επεξεργάζονται δεδομένα. Ο βασικός σκοπός τους είναι η υποστήριξη των οργανισμών στη λήψη αποφάσεων, τη βελτίωση της αποδοτικότητας και την παροχή πληροφοριών σε χρήστες. Τα συστατικά στοιχεία ενός πληροφοριακού συστήματος είναι το υλικό, το λογισμικό, τα δεδομένα, οι διαδικασίες και οι άνθρωποι ή χρήστες.

Παραδείγματα πληροφοριακών συστημάτων περιλαμβάνουν τα Συστήματα Διαχείρισης Επιχειρησιακών Πόρων, τα Συστήματα Διαχείρισης Πελατειακών Σχέσεων, τα Διοικητικά Συστήματα Υποστήριξης Αποφάσεων και τα Γεωγραφικά Συστήματα Πληροφοριών. Καθένα από αυτά τα συστήματα έχει σχεδιαστεί για να καλύπτει διαφορετικές ανάγκες και λειτουργίες ενός οργανισμού, καθιστώντας τα απαραίτητα εργαλεία σε σύγχρονες επιχειρήσεις και οργανισμούς.

Μπορούν να διαδραματίσουν σημαντικό ρόλο στην διαδικασία απόσταξης αλκοολούχων ποτών, ενισχύοντας την παραγωγική διαδικασία μέσω της αυτοματοποίησης και της ανάλυσης δεδομένων. Μερικές βασικές εφαρμογές τους στην ποτοποιία είναι:

- Παρακολούθηση και έλεγχος κρίσιμων παραμέτρων, όπως η θερμοκρασία.
- Συλλογή και ανάλυση δεδομένων από τις διαφορετικές φάσης της απόσταξης.
- Διαχείριση αποθεμάτων πρώτων υλών και τελικών προϊόντων.
- Συστήματα διαχείρισης ποιότητας και προτύπων.

Συνολικά, τα πληροφοριακά συστήματα ενισχύουν την αποδοτικότητα, την ακρίβεια και την ευελιξία στη διαδικασία της απόσταξης, συμβάλλοντας στη διατήρηση της ποιότητας και της ανταγωνιστικότητας, στην βιομηχανία παραγωγής αλκοολούχων ποτών.

2.6.1 Υπολογιστική νέφους

Η υπολογιστική νέφους, είναι μία τεχνολογία που αναφέρεται στη χρήση απομακρυσμένων διακομιστών και υποδομών μέσω του διαδικτύου, για την επεξεργασία και διαχείριση δεδομένων, αντί να χρησιμοποιούνται τοπικοί υπολογιστές ή φυσικοί διακομιστές. Με άλλα λόγια, οι χρήστες και οι οργανισμοί μπορούν να χρησιμοποιούν υπολογιστικούς πόρους, όπως υπολογιστική ισχύ, αποθηκευτικό χώρο και εφαρμογές, μέσω του διαδικτύου, χωρίς να χρειάζεται να διαχειρίζονται ή να κατέχουν την υποκείμενη υλική υποδομή.

Η υπολογιστική νέφους χωρίζεται σε διάφορους τύπους, οι οποίοι εξυπηρετούν διαφορετικές ανάγκες και σενάρια χρήσης:

- Δημόσιο νέφος, Public Cloud.
- Ιδιωτικό νέφος, Private Cloud.
- Υβριδικό νέφος, Hybrid Cloud.
- Πολυμορφικό νέφος, Multi – Cloud.

Στο δημόσιο νέφος, οι υπηρεσίες και οι υποδομές παρέχονται από εξωτερικούς πάροχους, όπως είναι οι Amazon Web Services (AWS), Microsoft Azure και Google Cloud. Οι πόροι μοιράζονται μεταξύ πολλών χρηστών μέσω του διαδικτύου. Πλεονεκτεί για την ευκολία στην πρόσβαση που προσφέρει, την επεκτασιμότητα και το χαμηλό κόστος συντήρησης. Προκαλεί όμως ανησυχία για την ασφάλεια και την ιδιωτικότητα, το γεγονός ότι τα δεδομένα αποθηκεύονται σε data centers, εκτός των οργανισμών και των επιχειρήσεων.

Το ιδιωτικό νέφος χρησιμοποιείται αποκλειστικά από έναν συγκεκριμένο οργανισμό ή επιχείρηση, τοπικά. Αυτό έχει ως αποτέλεσμα, όλοι οι πόροι του συστήματος να είναι διαθέσιμοι για τους τοπικού χρήστες. Επίσης, προσφέρει προσαρμογή στις ανάγκες του οργανισμού, βελτιωμένη ασφάλεια και μεγαλύτερο έλεγχο. Τα αδύνατα του σημεία είναι η περιορισμένη δυνατότητα κλιμάκωσης σε συνδυασμό με το χρονοβόρο και υψηλό κόστος ανάπτυξης και συντήρησης των υλικών υποδομών.

Το υβριδικό νέφος συνδυάζει τα χαρακτηριστικά του δημόσιου και ιδιωτικού νέφους, επιτρέποντας την μεταφορά δεδομένων και εφαρμογών μεταξύ τους. Προσφέρει ακόμα μεγαλύτερη ευελιξία, βελτίωση του κόστους και προσαρμόζεται στις ανάγκες του οργανισμού ή επιχείρησης. Στα αρνητικά χαρακτηριστικά είναι η πολυπλοκότητα διαχείρισης των συστημάτων και οι απαιτήσεις για συμβατότητα μεταξύ τους.

Τέλος το πολυμορφικό νέφος, περιλαμβάνει την χρήση πολλαπλών δημόσιων cloud από διάφορους πάροχους, για την βελτίωση της ανθεκτικότητας και της απόδοσης των συστημάτων. Αυτό μειώνει τον κίνδυνο εξάρτησης από ένα πάροχο και δίνει την δυνατότητα να συνδυαστούν οι βέλτιστες και αξιόπιστες υπηρεσίες από κάθε πάροχο. Έτσι όμως, αυξάνεται η πολυπλοκότητα διαχείρισης των συστημάτων αυτών και δημιουργούνται ανάγκες για ενοποίηση των υπηρεσιών. Αυτοί οι τύποι υπολογιστικής νέφους παρέχουν διαφορετικές δυνατότητες και επιλογές ανάλογα με τις ανάγκες του κάθε οργανισμού ή χρήστη.
Τα πλεονεκτήματα της υπολογιστικής νέφους είναι:

- Ευκολία πρόσβασης: Οι χρήστες μπορούν να έχουν πρόσβαση στα δεδομένα τους και τις εφαρμογές τους από οποιοδήποτε σημείο του κόσμου, αν διαθέτουν σύνδεση στο διαδίκτυο.
- Κόστος: Οι επιχειρήσεις μπορούν να μειώσουν το κόστος εξοπλισμού και συντήρησης των υποδομών τους.
- Ευελιξία και ελαστικότητα: Υπάρχει η δυνατότητα, οι υπηρεσίες και οι πόροι, να μπορούν να προσαρμοστούν ανάλογα με τις απαιτήσεις και τις ανάγκες των οργανισμών και των επιχειρήσεων.
- Ασφάλεια: Οι πάροχοι υπηρεσιών υπολογιστικής νέφους, προσφέρουν υψηλά επίπεδα ασφάλειας για τα δεδομένα των χρηστών.

Η υπολογιστική νέφους προσφέρει διάφορα μοντέλα υπηρεσιών. Η παροχή υποδομών μέσω διαδικτύου (IaaS, Infrastructure as a Service), όπως διακομιστές, αποθηκευτικό χώρο και λοιπών υποδομών. Η παροχή πλατφορμών ανάπτυξης, δοκιμής και φιλοξενίας εφαρμογών (PaaS, Platform as a Service), χωρίς

να μπούμε σε διαδικασία διαχείρισης των υπολοίπων πόρων. Και τέλος η παροχή λογισμικού μέσο διαδικτύου πάνω από το διαδίκτυο (SaaS, Software as a Service), όπως το Google Docs, Office365 και Teams. Είναι μία τεχνολογία που αλλάζει τον τρόπο που εργαζόμαστε και αλληλεπιδρούμε με τα δεδομένα, παρέχοντας παράλληλα πολλές δυνατότητες και ευκολίες. Συνδυαστικά, τα πληροφοριακά συστήματα και η υπολογιστική νέφους δίνουν στους οργανισμούς τη δυνατότητα να διαχειρίζονται και να επεξεργάζονται δεδομένα αποτελεσματικά, προσφέροντας ισχυρά εργαλεία για την ενίσχυση της λειτουργίας και την επίτευξη στόχων. Οι επιχειρήσεις μπορούν να επωφεληθούν από τις δυνατότητες και την απόδοση της τεχνολογίας της υπολογιστικής νέφους, εξασφαλίζοντας ταυτόχρονα ασφάλεια και προσαρμοστικότητα στις ανάγκες τους.

2.6.2 Ο διακομιστής

Οι διακομιστές, είναι γνωστοί και ως servers. Είναι ισχυροί υπολογιστές ή συστήματα που παρέχουν πόρους ή υπηρεσίες σε άλλους υπολογιστές ή χρήστες, μέσω ενός δικτύου. Οι χρήσεις των διακομιστών είναι:

- Φιλοξενία ιστοσελίδων και παροχή του περιεχομένου στους χρήστες.
- Αποθήκευση δεδομένων με δυνατότητα προσπέλασης από απομακρυσμένες συσκευές.
- Αποστολή και λήψη ηλεκτρονικής αλληλογραφίας (e-mail).
- Εκτέλεση εφαρμογών ή υπηρεσιών.

Οι διακομιστές χωρίζονται και σε κατηγορίες ανάλογα με τη χρήση τους. Τα είδη τους είναι:

- Διακομιστές ιστού, για την εξυπηρέτηση ιστοσελίδων.
- Διακομιστές αρχείων, για την αποθήκευση και διαχείριση αρχείων.
- Διακομιστές βάσεων δεδομένων, για τη διαχείριση και αποθήκευση βάσεων δεδομένων.
- Διακομιστές παιχνιδιών, για την υποστήριξη παιχνιδιών πολλών χρηστών.

2.6.2.1 Node.js

Το Node.js [22] είναι το runtime περιβάλλον που μας επιτρέπει να τρέχουμε εφαρμογές JavaScript εκτός των εφαρμογών περιήγησης, δηλαδή στον διακομιστή. Αναπτύχθηκε πάνω στην “μηχανή” V8 της Google και χρησιμοποιείται κυρίως για τη δημιουργία back-end εφαρμογών. Τα κύρια χαρακτηριστικά του είναι:

- Γρήγορο και ελαφρύ λόγω του μηχανισμού V8.
- Η ασύγχρονη λειτουργία του το καθιστά ιδανικό για εφαρμογές με πολλά αιτήματα.
- Διαθέτει μεγάλο οικοσύστημα με πρόσθετα πακέτα.
- Ιδανικό για εφαρμογές πραγματικού χρόνου.

Χρησιμοποιείται κυρίως για τη δημιουργία REST APIs και εφαρμογών πραγματικού χρόνου. Επίσης μαζί με τη χρήση πρόσθετων πακέτων, μπορεί να χρησιμοποιηθεί ως web server, για την ανάπτυξη backend scripts και εργαλείων. Παρακάτω παραθέτονται σε αλφαβητική σειρά οι βιβλιοθήκες που έχουν χρησιμοποιηθεί:

- *body-parser*: η βιβλιοθήκη αυτή χρησιμοποιείται για την ανάλυση του σώματος των εισερχομένων αιτήσεων HTTP. Ειδικότερα, χρησιμοποιείται για την επεξεργασία δεδομένων που αποστέλλονται μέσω POST, PUT, ή άλλων HTTP μεθόδων, και εξάγει τα δεδομένα αυτών των αιτήσεων ώστε να είναι διαθέσιμα ως JavaScript αντικείμενα στον server
  https://www.npmjs.com/package/body-parser

- *chalk*: ένα πολύ δημοφιλές εργαλείο που χρησιμοποιείται για τη μορφοποίηση και τον χρωματισμό κειμένου στην κονσόλα. Μας επιτρέπει να δημιουργούμε πιο ευανάγνωστες και καλοσχεδιασμένες εξόδους, χρησιμοποιώντας χρώματα, στυλ κειμένου, και πολλά άλλα. Είναι εξαιρετικά χρήσιμη για την καταγραφή μηνυμάτων σε εφαρμογές που τρέχουν από τη γραμμή εντολών.

https://www.npmjs.com/package/chalk

- *ejs*: η βιβλιοθήκη EJS (Embedded JavaScript Templates) είναι ένα δημοφιλές εργαλείο για την ανάπτυξη εφαρμογών στον Node.js. Χρησιμοποιείται για τη δημιουργία δυναμικών HTML σελίδων μέσω ενσωματωμένων JavaScript scripts, μέσα στα αρχεία HTML. Επιτρέπει την παραγωγή HTML κώδικα, με δεδομένα που παρέχονται από τον server και η σύνταξη είναι κατανοητή για προγραμματιστές που γνωρίζουν HTML και JavaScript.
  https://www.npmjs.com/package/ejs

- express: η βιβλιοθήκη Express είναι ένα από τα πιο ευρέως χρησιμοποιούμενα frameworks για Node.js. Είναι ελαφρύ, ευέλικτο, και έχει σχεδιαστεί για να διευκολύνει την ανάπτυξη web εφαρμογών και API s. Αποτελεί τον κορμό πολλών εφαρμογών που βασίζονται στο Node.js. Παρέχει ένα ισχυρό σύστημα για τον καθορισμό διαδρομών και την απόκριση σε διαφορετικές HTTP μεθόδους. Διαθέτει επίσης μεγάλη κοινότητα με πληθώρα παραδειγμάτων και επεκτάσεων.
  https://expressjs.com/

- *http*: η βιβλιοθήκη http για Node.js είναι ένα από τα βασικά modules που παρέχονται από προεπιλογή, χωρίς την ανάγκη εγκατάστασης επιπλέον πακέτων. Χρησιμοποιείται για τη δημιουργία και τη διαχείριση διακομιστών (servers) και την αποστολή αιτήσεων (requests). Αποτελεί τον θεμέλιο λίθο για την κατασκευή εφαρμογών web και APIs στο Node.js.
  https://nodejs.org/api/http.html

- *socketio*: ένα εργαλείο που επιτρέπει την αμφίδρομη επικοινωνία πραγματικού χρόνου, μεταξύ client και server. Είναι ιδανική για εφαρμογές που απαιτούν άμεση ανταλλαγή δεδομένων, όπως εφαρμογές chat, ειδοποιήσεις ή multiplayer παιχνίδια.
  https://www.npmjs.com/package/socket.io

- *mysql2*: είναι ένα γρήγορο και αποδοτικό πακέτο για σύνδεση με MySQL βάσεις δεδομένων στο Node.js. Είναι μία βελτιωμένη έκδοση του mysql πακέτου, με υποστήριξη για non-utf8 encoding, binary log protocol, compression, ssl, connection pooling και ταχύτερη εκτέλεση ερωτημάτων.
  https://www.npmjs.com/package/mysql2

2.6.2.2 Περιγραφή λειτουργιών διακομιστή

Ο διακομιστής της εφαρμογής αναπτύχθηκε με τη χρήση της πλατφόρμας Node.js, αξιοποιώντας τη μοντέρνα αρχιτεκτονική της JavaScript στον server για υψηλή απόδοση και επεκτασιμότητα. Μία από τις βασικές λειτουργίες του είναι η ζωντανή σύνδεση με τη μονάδα IoT μέσω web sockets, επιτρέποντας την αμφίδρομη επικοινωνία σε πραγματικό χρόνο. Μέσω αυτού του καναλιού, ο διακομιστής λαμβάνει συνεχώς δεδομένα αισθητήρων από τη συσκευή, όπως τις μετρήσεις θερμοκρασίας και πληροφορίες λειτουργίας, καταγράφοντας τα σε σχεσιακή βάση δεδομένων για αποθήκευση και μελλοντική ανάλυση.

Πέρα από την αποθήκευση, ο διακομιστής εκτελεί επεξεργασία των δεδομένων με στόχο την εξαγωγή βασικών στατιστικών, όπως η μέγιστη, ελάχιστη και μέση θερμοκρασία ανά χρονική περίοδο. Τα αποτελέσματα αυτής της επεξεργασίας καθιστούν εφικτή την αποδοτική εποπτεία της λειτουργίας της IoT συσκευής και τη λήψη αποφάσεων βασισμένων σε δεδομένα.

Παράλληλα, υποστηρίζεται απομακρυσμένος έλεγχος της συσκευής μέσω εντολών που διαβιβάζονται μέσω του ίδιου καναλιού web sockets, επιτρέποντας την ενεργοποίηση ή απενεργοποίηση μηχανισμών της μονάδας, καθώς και την τροποποίηση παραμέτρων λειτουργίας από απόσταση.

Επιπρόσθετα, ο διακομιστής είναι υπεύθυνος για την παράδοση της εφαρμογής χρήστη, παρέχοντας μία εύχρηστη και διαδραστική διεπαφή μέσω browser, στον τελικό χρήστη. Μέσω αυτής της διεπαφής, τα δεδομένα μεταδίδονται στον χρήστη σε πραγματικό χρόνο, ενισχύοντας την εποπτεία και την άμεση ανταπόκριση σε κρίσιμες συνθήκες. Η ενσωμάτωση όλων των λειτουργιών σε μια ενοποιημένη υποδομή προσφέρει μία σύγχρονη, ευέλικτη και επεκτάσιμη λύση για εφαρμογές IoT. Οι λειτουργίες του διακομιστή:

- Σύνδεση με την μονάδα IoT με web sockets: ο διακομιστής αναλαμβάνει τη διαχείριση της αμφίδρομης επικοινωνίας με τη μονάδα IoT μέσω του πρωτοκόλλου websockets, το οποίο επιτρέπει συνεχή σύνδεση σε πραγματικό χρόνο.
- Αποστολή και λήψη δεδομένων με τη μονάδα IoT: η χρήση websockets δίνει τη δυνατότητα στο διακομιστή να λαμβάνει και να αποστέλλει δεδομένα ταυτόχρονα, χωρίς καθυστέρηση.
- Αποθήκευση δεδομένων σε βάση δεδομένων MySQL: τα ληφθέντα δεδομένα καταγράφονται στη βάση δεδομένων για αρχειοθέτηση και ανάλυση.
- Επεξεργασία δεδομένων θερμοκρασίας: ο διακομιστής αναλαμβάνει την ανάλυση των μετρήσεων θερμοκρασίας που λαμβάνει από τη μονάδα IoT, προκειμένου να εξάγει στατιστικά μεγέθη που είναι χρήσιμα για την εποπτεία και τη διαχείριση της συσκευής. Συγκεκριμένα, οι υπολογισμοί περιλαμβάνουν αλγόριθμο που εντοπίζει την υψηλότερη και χαμηλότερη καταγεγραμμένη τιμή θερμοκρασίας, καθώς επίσης και υπολογισμός του μέσου όρου της θερμοκρασίας.
- Απομακρυσμένος έλεγχος και εποπτεία της συσκευής IoT: η αρχιτεκτονική του συστήματος επιτρέπει στον χρήστη να αλληλεπιδρά με τη συσκευή IoT μέσω της εφαρμογής, χωρίς φυσική παρουσία στον χώρο εγκατάστασης.
- Φιλοξενία της εφαρμογής χρήστη: η εφαρμογή χρήστη φιλοξενείται σε διαδικτυακό περιβάλλον με στόχο την ευκολότερη πρόσβαση, διαθεσιμότητα και επεκτασιμότητα. Η υποδομή φιλοξενίας σχεδιάστηκε ώστε να υποστηρίζει τη λειτουργία σε πραγματικό χρόνο και την ασφαλή αλληλεπίδραση με τη συσκευή IoT.
- Αποστολή δεδομένων στην εφαρμογή χρήστη με το πρωτόκολλο επικοινωνίας HTTP: Ο διακομιστής επικοινωνεί με την εφαρμογή χρήστη μέσω του πρωτοκόλλου HTTP, παρέχοντας δομημένη και ασφαλή μεταφορά δεδομένων που απαιτούνται για την απεικόνιση και τον έλεγχο της συσκευής IoT.

Η συγγραφή και ανάπτυξη του κώδικα πραγματοποιήθηκε στο περιβάλλον Visual Studio Code (VS Code), ένα σύγχρονο και ισχυρό εργαλείο επεξεργασίας κώδικα, το οποίο προσφέρει ευρεία υποστήριξη για JavaScript/Node.js. Για τη διευκόλυνση της ανάπτυξης και την καλύτερη διαχείριση του κώδικα, αξιοποιήθηκαν οι παρακάτω επεκτάσεις του Visual Studio Code:

- Prettier – Code Formatter: Εργαλείο αυτόματης μορφοποίησης κώδικα που εξασφαλίζει ομοιομορφία και καθαρότητα σε όλο το έργο. Εφαρμόστηκε σε αρχεία JavaScript, EJS, JSON και CSS.
  https://prettier.io/

- EJS Language Support: Παρέχει υποστήριξη σύνταξης (syntax highlighting), αποσπάσματα κώδικα και δομική ευκρίνεια για τα αρχεία .ejs, τα οποία χρησιμοποιούνται για τη δημιουργία δυναμικών HTML σελίδων στον διακομιστή. Με αυτόν τον τρόπο διευκολύνεται η συγγραφή των ιστοσελίδων του frontend που εξυπηρετείται από το Node.js backend.
  https://marketplace.visualstudio.com/items?itemName=DigitalBrainstem.javascript-ejs-support

2.6.3 Η εφαρμογή χρήστη

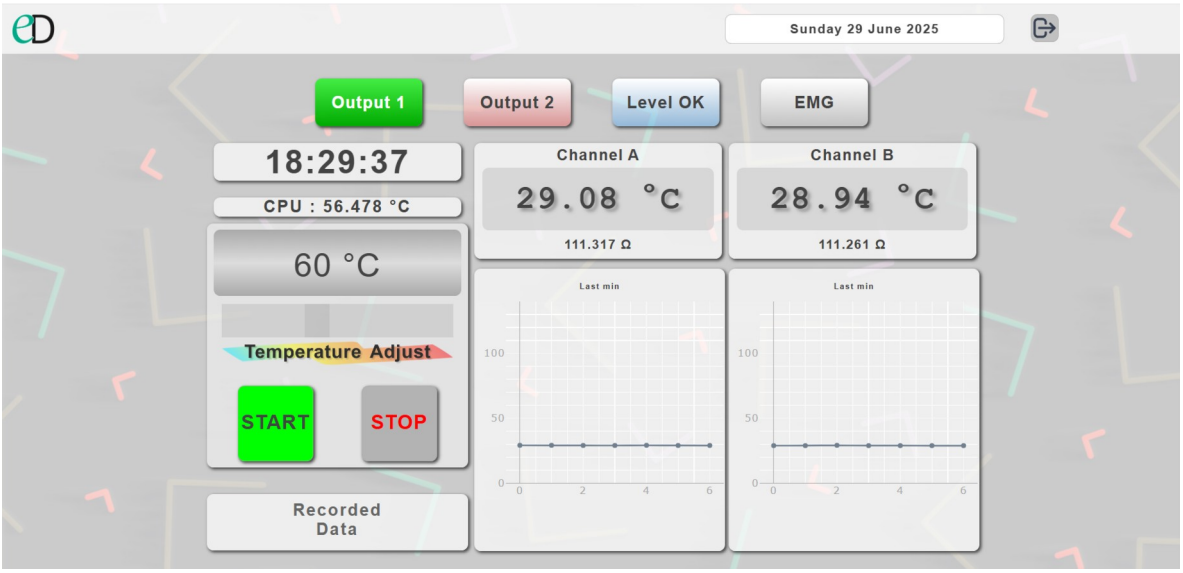
Η εφαρμογή χρήστη επιτρέπει την απομακρυσμένη παρακολούθηση και διαχείριση των μετρήσεων, καθώς επίσης και τον έλεγχο της συσκευής. Ο χρήστης μπορεί να παρακολουθεί τις μετρήσεις της θερμοκρασίας σε πραγματικό χρόνο, να έχει πρόσβαση σε ιστορικά δεδομένα και οπτικοποιήση με γραφήματα. Χρησιμοποιεί σύστημα ταυτοποίησης για την προστασία και την ασφάλεια των δεδομένων και της συσκευής.

Για την ανάπτυξη της web εφαρμογής έγινε χρήση HTML, CSS και Javascript με βιβλιοθήκες όπως η Jquery και Plotly. Η αποθήκευση των δεδομένων όπως προαναφέρθηκε γίνεται σε MySQL και για την επικοινωνία με τον μικροελεγκτή χρησιμοποιείται το πρωτόκολλό HTTP.

2.6.3.1 Περιγραφή διεπαφής χρήστη

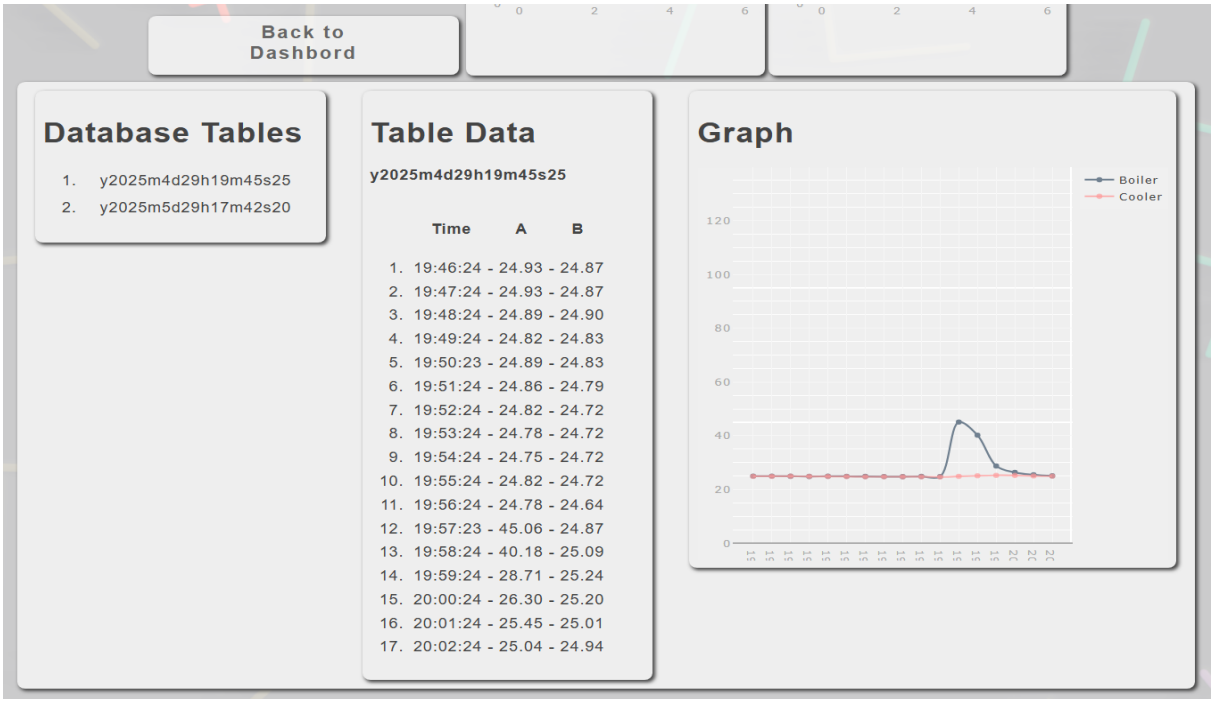
Η παρακάτω διεπαφή αποτελεί το γραφικό περιβάλλον της web εφαρμογής που εξυπηρετείται από τον διακομιστή Node.js. Μέσω αυτής, ο χρήστης μπορεί να παρακολουθεί σε πραγματικό χρόνο τα δεδομένα

της συνδεδεμένης IoT μονάδας, να εποπτεύει κρίσιμες μετρήσεις, και να πραγματοποιεί απομακρυσμένες ενέργειες ελέγχου με ευχρηστία και αξιοπιστία. Η δυναμική σύνδεση με τον server επιτρέπει τη συνεχή ροή και απεικόνιση των δεδομένων, προσφέροντας μια πλήρως διαδραστική εμπειρία.



Εικόνα 29: Η κεντρική οθόνη της διεπαφής του χρήστη.

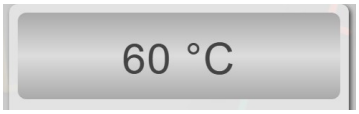
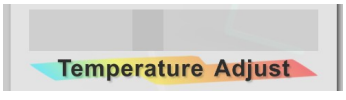
Η παρακάτω εικόνα, παρουσιάζει τα καταγεγραμμένα δεδομένα λειτουργίας της IoT συσκευής, όπως συλλέγονται και επεξεργάζονται σε πραγματικό χρόνο από τον διακομιστή. Η δυναμική οπτικοποίηση των μετρήσεων επιτρέπει την εύκολη παρακολούθηση των παραμέτρων λειτουργίας.

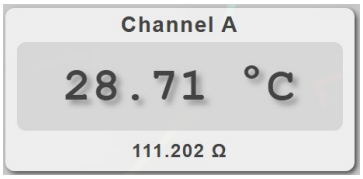
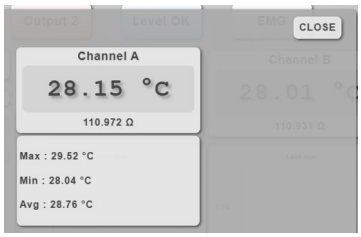
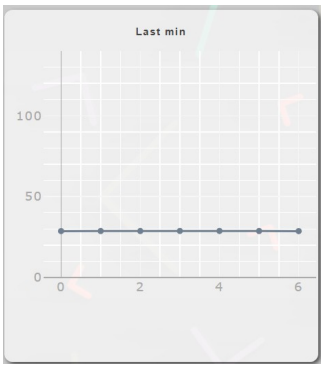
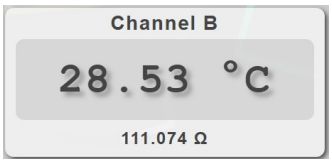
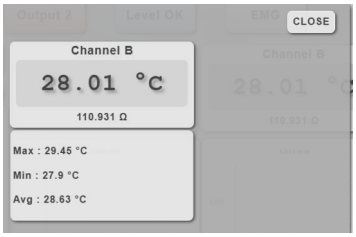


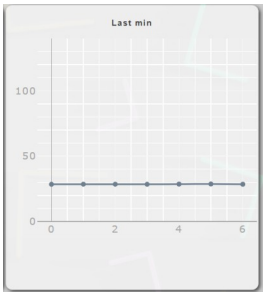
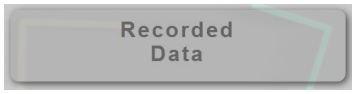
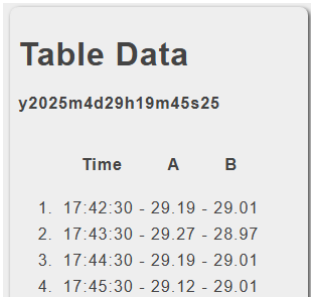
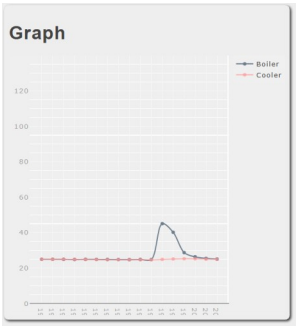
Εικόνα 30: Η οθόνη με τα καταγεγραμμένα δεδομένα λειτουργίας.

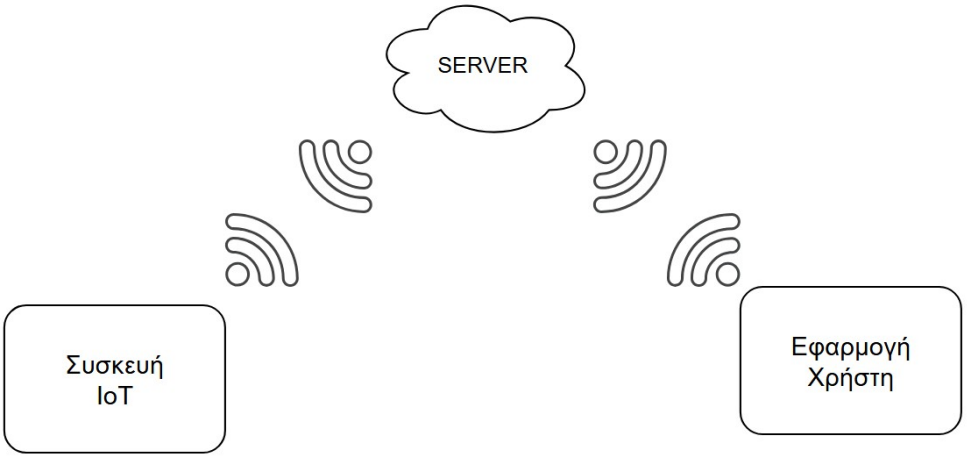
Η διεπαφή χρήστη του πίνακα ελέγχου, περιλαμβάνει περιλαμβάνει τα μπουτόν Start / Stop, ενδείξεις θερμοκρασίας με γραφήματα, ενδείξεις με την κατάσταση λειτουργίας, καθώς επίσης και γενικές

πληροφορίες, όπως είναι η ημερομηνία και η ώρα. Στον παρακάτω πίνακα παρουσιάζονται αναλυτικά, όλα τα στοιχεία της διεπαφής.

Πίνακας 2.6.3.1	
	Ένδειξη τρέχουσας ημερομηνίας.
	Μπουτόν αποσύνδεσης – εξόδου του χρήστη από την εφαρμογή.
	Ένδειξη λειτουργίας ηλεκτρονόμου εξόδου 1.
	Ένδειξη λειτουργίας ηλεκτρονόμου εξόδου 2.
	Ένδειξη αισθητήρα στάθμης, για τον έλεγχο της πλήρωσης της δεξαμενής
	Ένδειξη μπουτόν εκτάκτου ανάγκης.
	Ένδειξη της τρέχουσας ώρας.
	Ένδειξη της θερμοκρασίας του επεξεργαστή της κεντρικής μονάδας RaspberryPi.
	Η θερμοκρασία ρύθμισης του χρήστη.
	Μπάρα ρύθμισης της επιθυμητής θερμοκρασίας λειτουργίας.
	Μπουτόν εκκίνησης του συστήματος.

	Μπουτόν παύσης λειτουργίας του συστήματος.
	Ένδειξη θερμοκρασίας και αντίστασης του αισθητήρα PT100 του καναλιού Α. Αν επιλέξουμε το συγκεκριμένο widget θα μας ανοίξει το παρακάτω αναδυόμενο παράθυρο, το οποίο παραθέτει τη μέγιστη, την ελάχιστη και τον μέσο όρο της θερμοκρασίας.
	Το αναδυόμενο παράθυρο που απεικονίζει τη μέγιστη, την ελάχιστη και τον μέσο όρο της θερμοκρασίας.
	Γραφική απεικόνιση της θερμοκρασίας του καναλιού Α για το τελευταίο λεπτό λειτουργίας.
	Ένδειξη θερμοκρασίας και αντίστασης του αισθητήρα PT100 του καναλιού Β. Αν επιλέξουμε το συγκεκριμένο widget θα μας ανοίξει το παρακάτω αναδυόμενο παράθυρο, το οποίο παραθέτει την μέγιστη, την ελάχιστη και τον μέσο όρο της θερμοκρασίας.
	Το αναδυόμενο παράθυρο που απεικονίζει τη μέγιστη, την ελάχιστη και τον μέσο όρο της θερμοκρασίας.

	Γραφική απεικόνιση της θερμοκρασίας του καναλιού Β για το τελευταίο λεπτό λειτουργίας.
	Μετάβαση στη σελίδα απεικόνισης των καταγεγραμμένων δεδομένων λειτουργίας.
	Οι διαθέσιμοι πίνακες που έχουν καταγραφεί τα δεδομένα λειτουργίας.
	Πίνακας με τα καταγεγραμμένα δεδομένα.
	Γραφική απεικόνιση των θερμοκρασιών που καταγράφηκαν στη βάση δεδομένων.
	Επιστροφή στην αρχική σελίδα του πίνακα ελέγχου.



Εικόνα 31: Το σχεδιο του δικτύου.

2.6.3.2 Περιγραφή λειτουργιών της εφαρμογής χρήστη

Η εφαρμογή χρήστη σχεδιάστηκε με σκοπό την αμφίδρομη επικοινωνία μεταξύ του τελικού χρήστη και της συσκευής, παρέχοντας ένα ολοκληρωμένο περιβάλλον εποπτείας και ελέγχου. Οι βασικές λειτουργίες της εφαρμογής είναι οι εξής:

- Λήψη δεδομένων από τον διακομιστή: η εφαρμογή συνδέεται με τον απομακρυσμένο διακομιστή και λαμβάνει τις μετρήσεις και τα στοιχεία λειτουργίας κάθε ένα δευτερόλεπτο. Η επικοινωνία βασίζεται στο πρωτόκολλο επικοινωνίας HTTP.
- Απεικόνιση μετρήσεων σε πραγματικό χρόνο: οι ζωντανές μετρήσεις παρουσιάζονται στον χρήστη μέσω των οργάνων απεικόνισης καθώς επίσης και με γράφημα.
- Απεικόνιση ιστορικών δεδομένων από τη βάση δεδομένων: Η εφαρμογή παρέχει στον χρήστη δυνατότητα πρόσβασης στα ιστορικά δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων, με την πληροφορία να παρουσιάζεται τόσο σε μορφή πίνακα όσο και μέσω γραφικής απεικόνισης.
- Απομακρυσμένος έλεγχος της λειτουργίας της συσκευής: μέσω της εφαρμογής, ο χρήστης μπορεί να εκτελεί εντολές ελέγχου προς τη συσκευή, όπως η ενεργοποίηση και η απενεργοποίηση της λειτουργίας του συστήματος.
- Σύστημα ταυτοποίησης χρήστη: το σύστημα ταυτοποίησης αποτελεί βασικό μηχανισμό ασφαλείας της εφαρμογής, εξασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση στις λειτουργίες εποπτείας και ελέγχου της συσκευής IoT.

Κεφάλαιο 3ο : Πειραματική διαδικασία

3.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζονται οι πειραματικές διαδικασίες που πραγματοποιήθηκαν στο πλαίσιο της αξιολόγησης του Ηλεκτρονικού Θαλάμου Απόσταξης. Οι δοκιμές περιλαμβάνουν αρχικά τη μέτρηση θερμοκρασίας σε συνθήκες δωματίου, με χρήση αισθητήρων PT100, θερμόμετρου χειρός υπερύθρων και θερμομέτρου με αισθητήρα DHT22, με σκοπό την επαλήθευση της ακρίβειας και της σταθερότητας των μετρήσεων. Παράλληλα, εξετάστηκε η λειτουργία του API, του server και της αποθήκευσης δεδομένων σε βάση MySQL, ώστε να διασφαλιστεί η ομαλή επικοινωνία μεταξύ υλικού και λογισμικού. Στη συνέχεια, πραγματοποιήθηκε πείραμα απόσταξης νερού, με στόχο την αξιολόγηση της συνολικής απόδοσης του συστήματος υπό πραγματικές συνθήκες λειτουργίας. Οι διαδικασίες, τα αποτελέσματα και οι παρατηρήσεις που προέκυψαν από τα πειράματα αναλύονται διεξοδικά στις επόμενες ενότητες.

3.2 Πείραμα πρώτο - Έλεγχος καλής λειτουργίας συστήματος

Στο πλαίσιο της αρχικής πειραματικής αξιολόγησης του Ηλεκτρονικού Θαλάμου Απόσταξης, πραγματοποιήθηκε δοκιμή διάρκειας δέκα λεπτών με σκοπό την επαλήθευση της ακρίβειας και της συνέπειας των αισθητήρων θερμοκρασίας. Οι μετρήσεις καταγράφηκαν ανά δέκα δευτερόλεπτα, χρησιμοποιώντας τρεις διαφορετικές μεθόδους: τους ενσωματωμένους αισθητήρες PT100 του συστήματος, ένα θερμόμετρο χειρός υπερύθρων, καθώς και έναν αισθητήρα τύπου DHT22. Η παράλληλη χρήση των τριών τεχνολογιών επέτρεψε τη σύγκριση των αποτελεσμάτων και την αξιολόγηση της αξιοπιστίας των PT100 σε πραγματικές συνθήκες περιβάλλοντος. Τα δεδομένα που συλλέχθηκαν αποτελούν τη βάση για την περαιτέρω βαθμονόμηση και βελτιστοποίηση του συστήματος.

3.3 Πειραματική μεθοδολογία πρώτου πειράματος

Η πρώτη φάση της πειραματικής αξιολόγησης του συστήματος πραγματοποιήθηκε σε οικιακό περιβάλλον, με στόχο την προκαταρκτική επαλήθευση της λειτουργίας των αισθητήρων θερμοκρασίας και της συνολικής σταθερότητας του συστήματος. Η επιλογή του σπιτιού ως χώρου δοκιμής επέτρεψε την παρατήρηση της συμπεριφοράς του θαλάμου σε συνθήκες καθημερινής χρήσης, χωρίς εργαστηριακό έλεγχο, αλλά με πραγματικές θερμοκρασιακές μεταβολές και εξωτερικές παρεμβολές.

Η δοκιμή διήρκεσε δέκα λεπτά, με καταγραφή μετρήσεων θερμοκρασίας κάθε δέκα δευτερόλεπτα. Χρησιμοποιήθηκαν τρεις διαφορετικές μέθοδοι μέτρησης:

- Αισθητήρες PT100: Ενσωματωμένοι στο σύστημα, συνδεδεμένοι με τη μονάδα ελέγχου για αυτόματη και συνεχή καταγραφή.
- Θερμόμετρο υπερύθρων χειρός: Χρησιμοποιήθηκε για επιτόπιες μετρήσεις στην εξωτερική επιφάνεια των αισθητήρων PT100.
- Αισθητήρας DHT22: Τοποθετημένος πλησίον των αισθητήρων PT100, συνδεδεμένος σε ξεχωριστό κύκλωμα με μικροελεγκτή τύπου Arduino για ανεξάρτητη μέτρηση της θερμοκρασίας.

Η αποθήκευση των μετρήσεων πραγματοποιήθηκε μέσω τοπικού δικτύου σε βάση δεδομένων MySQL, επιτρέποντας την άμεση πρόσβαση και ανάλυση των αποτελεσμάτων. Η παράλληλη χρήση των τριών τεχνολογιών επέτρεψε τη σύγκριση των τιμών και την αξιολόγηση της αξιοπιστίας των PT100 σε συνθήκες μη ελεγχόμενου περιβάλλοντος.

Κατά τη διάρκεια της δεκάλεπτης δοκιμής, συλλέχθηκαν συνολικά 60 μετρήσεις θερμοκρασίας από κάθε μέθοδο, με χρονική απόσταση 10 δευτερολέπτων μεταξύ τους. Τα δεδομένα καταγράφηκαν και αποθηκεύτηκαν σε βάση MySQL, επιτρέποντας την εύκολη εξαγωγή και ανάλυση. Ο παρακάτω πίνακας που δημιουργήθηκε περιλαμβάνει τις θερμοκρασιακές μετρήσεις από τους τρεις διαφορετικούς αισθητήρες και η καταγραφή έγινε σε πραγματικό χρόνο.

Πίνακας 3.1				
TIME	**Channel A (°C)**	**Channel B (°C)**	**DHT22 (°C)**	**Θερμόμετρο IR (°C)**
20:19:05	27.22	27.38	27.5	27.4
20:19:15	27.26	27.31	27.5	27.1
20:19:25	27.26	27.38	27.5	27.2
20:19:35	27.22	27.35	27.5	27.5
20:19:45	27.26	27.35	27.5	27.4
20:19:55	27.26	27.38	27.5	27.1
20:20:05	27.18	27.31	27.5	27.4
20:20:16	27.22	27.42	27.5	27.5
20:20:26	27.26	27.31	27.5	27.7
20:20:35	27.26	27.31	27.5	27.6
20:20:45	27.26	27.27	27.5	27.5
20:20:55	27.22	27.38	27.5	27.4
20:21:05	27.29	27.35	27.5	27.6
20:21:15	27.22	27.35	27.5	27.3
20:21:25	27.22	27.38	27.5	27.2
20:21:35	27.26	27.68	27.5	27.4
20:21:46	27.22	27.35	27.5	27.7
20:21:56	27.26	27.38	27.5	27.1
20:22:06	27.26	27.27	27.5	27.3
20:22:15	27.26	27.35	27.5	27.7
20:22:25	27.22	27.38	27.5	27.2
20:22:35	27.29	27.35	27.5	27.4
20:22:45	27.22	27.35	27.5	27.2
20:22:55	27.26	27.31	27.5	27.5
20:23:05	27.29	27.31	27.5	27.4
20:23:15	27.29	27.42	27.5	27.6
20:23:26	27.22	27.38	27.5	27.5
20:23:36	27.26	27.31	27.5	27.3
20:23:46	27.22	27.35	27.5	27.2
20:23:56	27.26	27.31	27.5	27.4
20:24:05	27.29	27.35	27.5	27.4
20:24:15	27.22	27.35	27.5	27.6
20:24:25	27.22	27.35	27.5	27.1
20:24:35	27.41	27.31	27.5	27.5
20:24:45	27.26	27.35	27.5	27.4

20:24:55	27.22	27.35	27.5	27.5
20:25:06	27.26	27.35	27.5	27.7
20:25:16	27.22	27.35	27.5	27.6
20:25:26	27.22	27.35	27.5	27.6
20:25:36	27.26	27.31	27.5	27.7
20:25:45	27.18	27.35	27.5	27.6
20:25:55	27.26	27.31	27.5	27.8
20:26:05	27.37	27.35	27.5	27.6
20:26:15	27.18	27.35	27.5	27.1
20:26:25	27.18	27.53	27.5	27.5
20:26:36	27.22	27.31	27.5	27.3
20:26:46	27.22	27.38	27.5	27.4
20:26:56	27.22	27.42	27.5	27.7
20:27:06	27.15	27.35	27.5	27.6
20:27:16	27.26	27.31	27.5	27.3
20:27:25	27.22	27.53	27.5	27.5
20:27:35	27.44	27.35	27.4	27.2
20:27:45	27.18	27.38	27.4	27.1
20:27:55	27.18	27.31	27.4	27.4
20:28:05	27.18	27.24	27.4	27.5
20:28:16	27.18	27.53	27.4	27.6
20:28:26	27.33	27.53	27.4	27.4
20:28:36	27.33	27.35	27.4	27.2
20:28:46	27.18	27.24	27.4	27.5
20:28:56	27.15	27.27	27.4	27.6

3.3.1 Υπολογισμός Μέσης Θερμοκρασίας ανά αισθητήρα

Η μέση θερμοκρασία αποτελεί έναν βασικό δείκτη για την αξιολόγηση της θερμικής συμπεριφοράς του συστήματος κατά τη διάρκεια της απόσταξης. Μέσω της ανάλυσης των μετρήσεων από τους δύο αισθητήρες μπορούμε να κατανοήσουμε τη συνολική απόδοση της διαδικασίας και τη σταθερότητα των θερμοκρασιών.

$$\Theta = \Theta_{SUM} / n \quad (3.1)$$

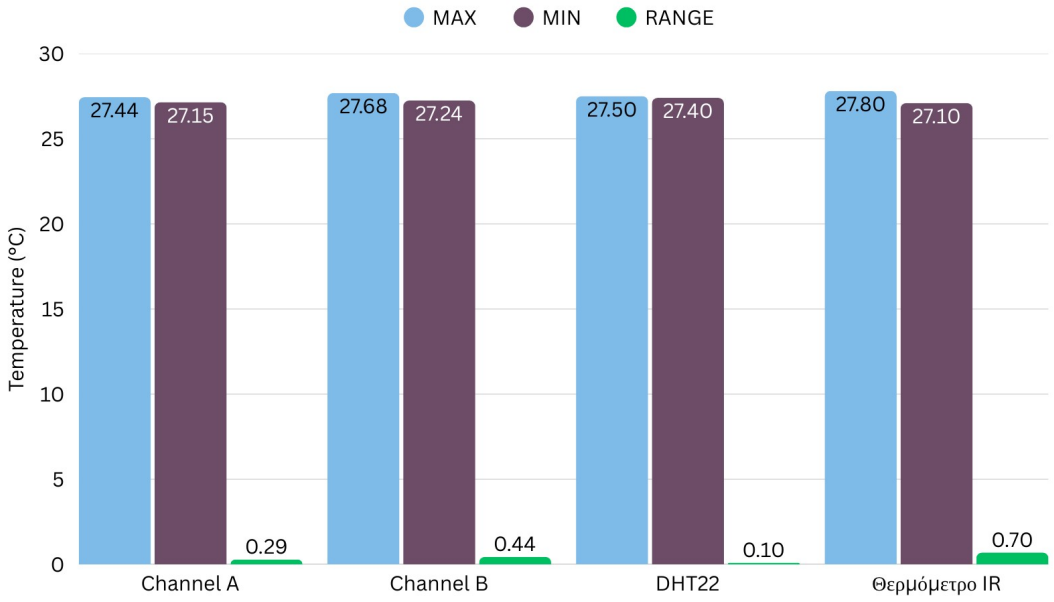
Θ: Μέση θερμοκρασίας
Θ_{SUM}: Θερμοκρασιακό άθροισμα
n: Το πλήθος των μετρήσεων

Πίνακας 3.2		
Αισθητήρας	**Θ_{SUM} (°C)**	**Μέση Θερμοκρασία (°C)**
Channel A	1633.1	27.22
Channel B	1641.66	27.36
DHT22	1649.2	27.49
Θερμόμετρο IR	1648.1	27.47

3.3.2 Ελάχιστη, μέγιστη θερμοκρασία ανά αισθητήρα και εύρος τιμών

Το εύρος τιμών είναι η διαφορά μεταξύ της μέγιστης και της ελάχιστης τιμής της θερμοκρασίας σε κάθε αισθητήρα.

Πίνακας 3.3			
Αισθητήρας	**Ελάχιστη (°C)**	**Μέγιστη (°C)**	**Εύρος (°C)**
Channel A	27.15	27.44	0.29
Channel B	27.24	27.68	0.44
DHT22	27.4	27.5	0.10
Θερμόμετρο IR	27.1	27.8	0.70



Εικόνα 32: Γράφημα μέγιστης / ελάχιστης θερμοκρασίας και το εύρος τιμών για το πρώτο πείραμα.

3.3.3 Τυπική απόκλιση

Η τυπική απόκλιση δείχνει πόσο διασκορπισμένες είναι οι τιμές γύρω από τη μέση θερμοκρασία. Ο υπολογισμός της τυπικής απόκλισης υλοποιήθηκε με τον παρακάτω μαθηματικό τύπο:

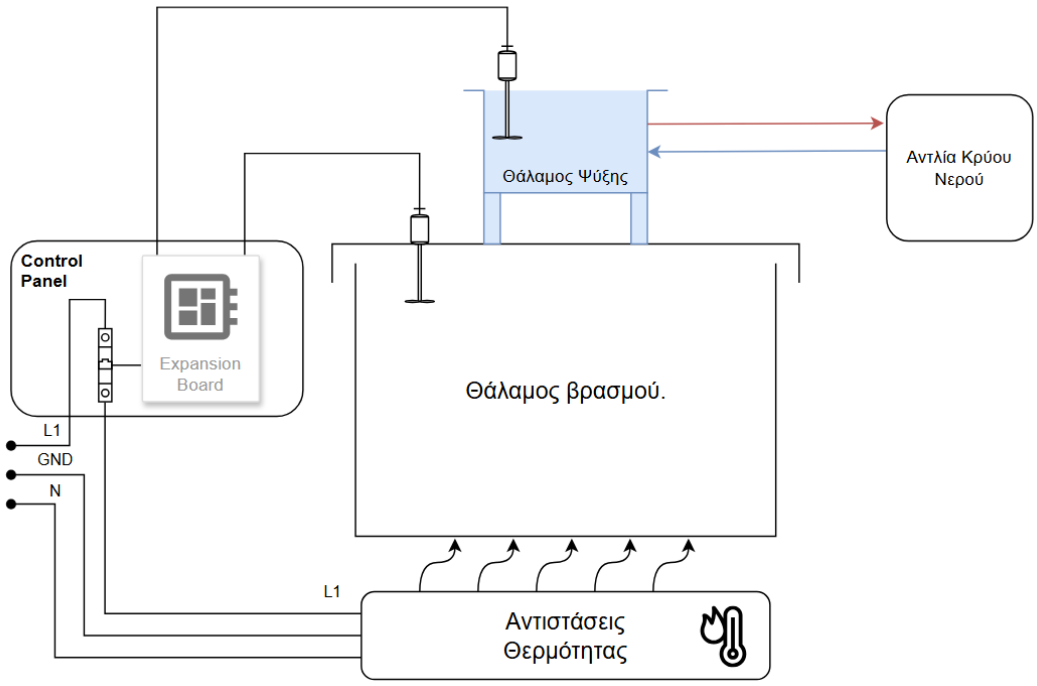
$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_i - \bar{T})^2} \quad (3.2)$$

σ: Η τυπική απόκλιση
n: Το πλήθος των μετρήσεων
T_i: Η επιμέρους τιμή της θερμοκρασίας
T: Η μέση τιμή της θερμοκρασίας

Πίνακας 3.4		
Αισθητήρας	**Μέση Θερμοκρασία (°C)**	**Τυπική Απόκλιση (°C)**
Channel A	27.22	0.06
Channel B	27.36	0.11
DHT22	27.49	0.03
Θερμόμετρο IR	27.47	0.18

3.4 Πείραμα δεύτερο – Απόσταξη νερού

Στο δεύτερο στάδιο της πειραματικής διαδικασίας, πραγματοποιήθηκε δοκιμή απόσταξης νερού με στόχο την αξιολόγηση της συνολικής λειτουργίας του Ηλεκτρονικού Θαλάμου Απόσταξης υπό πραγματικές συνθήκες. Η διαδικασία επέτρεψε την παρακολούθηση κρίσιμων παραμέτρων, όπως η θερμοκρασία και ο ρυθμός θέρμανσης, ενώ παράλληλα δοκιμάστηκε η ανταπόκριση των αισθητήρων και η αποδοτικότητα του συστήματος ελέγχου. Το πείραμα αυτό αποτέλεσε σημαντικό βήμα για την επιβεβαίωση της πρακτικής εφαρμογής της κατασκευής και την προετοιμασία για πιο σύνθετες διεργασίες απόσταξης.



Εικόνα 33: Γενικό σχέδιο της κατασκευής.

3.5 Πειραματική μεθοδολογία δεύτερου πειράματος

Η πειραματική διάταξη περιλαμβάνει τέσσερις αισθητήρες τοποθετημένους σε κρίσιμα σημεία του συστήματος, με στόχο την ακριβή καταγραφή θερμοκρασιακών μεταβολών κατά τη διάρκεια της διαδικασίας βρασμού και συμπύκνωσης. Ο αισθητήρας στο κανάλι μέτρησης Α βρίσκεται στον θάλαμο συμπύκνωσης, καταγράφοντας τη θερμοκρασία κατά τη φάση μετάβασης από αέρια σε υγρή μορφή. Ο αισθητήρας PT100, από το κανάλι Β, είναι εγκατεστημένος στο καπάκι του θαλάμου βρασμού, επιτρέποντας την παρακολούθηση της θερμοκρασίας των παραγόμενων ατμών. Ο αισθητήρας DHT22 είναι τοποθετημένος στον περιβάλλοντα χώρο και παρέχει δεδομένα θερμοκρασίας και σχετικής υγρασίας, τα οποία χρησιμοποιούνται για την εκτίμηση εξωτερικών επιδράσεων στο σύστημα. Τέλος, το θερμόμετρο χειρός υπερύθρων χρησιμοποιείται για τη μέτρηση της θερμοκρασίας του θαλάμου συμπύκνωσης και του συστήματος ψύξης.

Πίνακας 3.5		
Αισθητήρας	**Θέση**	**Σκοπός μέτρησης**
Channel A	Θάλαμος συμπύκνωσης	Θερμοκρασία συμπύκνωσης
Channel B	Καπάκι θαλάμου βρασμού	Θερμοκρασία ατμού / εξόδου
DHT22	Εσωτερικός χώρος (δωμάτιο)	Θερμοκρασία και υγρασία περιβάλλοντος
Θερμόμετρο υπερύθρων	Θάλαμος συμπύκνωσης	Επόπτευση θερμοκρασίας ψύξης

Εικόνα 34: Η τοποθέτηση του αισθητήρα PT100 στο καπάκι του λέβητα.

Κατά τη διάρκεια του πειράματος χρησιμοποιήθηκε 1 λίτρο νερό, με συνολική διάρκεια 3 ώρες, 26 λεπτά και 32 δευτερόλεπτα. Μετά την ολοκλήρωση της απόσταξης, συλλέχθηκαν 0,6 λίτρα καθαρού αποστάγματος, ενώ 0,3 λίτρα παρέμειναν στον λέβητα. Το υπόλοιπο 0,1 λίτρο αντιστοιχεί σε απώλειες που προέκυψαν κατά τη διαδικασία, καθώς και σε ποσότητα που παρέμεινε στον θάλαμο συμπύκνωσης. Οι πλήρεις θερμοκρασιακές μετρήσεις από τους δύο αισθητήρες παρατίθενται στον πίνακα Β.1 στο Παράρτημα Β.

3.5.1 Υπολογισμός Μέσης Θερμοκρασίας ανά αισθητήρα

Στον παρακάτω πίνακας παρουσιάζεται ο υπολογισμός της μέσης θερμοκρασίας για το δεύτερο πείραμα, βάσει των καταγεγραμμένων τιμών από τους δύο αισθητήρες (βλ. παράρτημα Β).

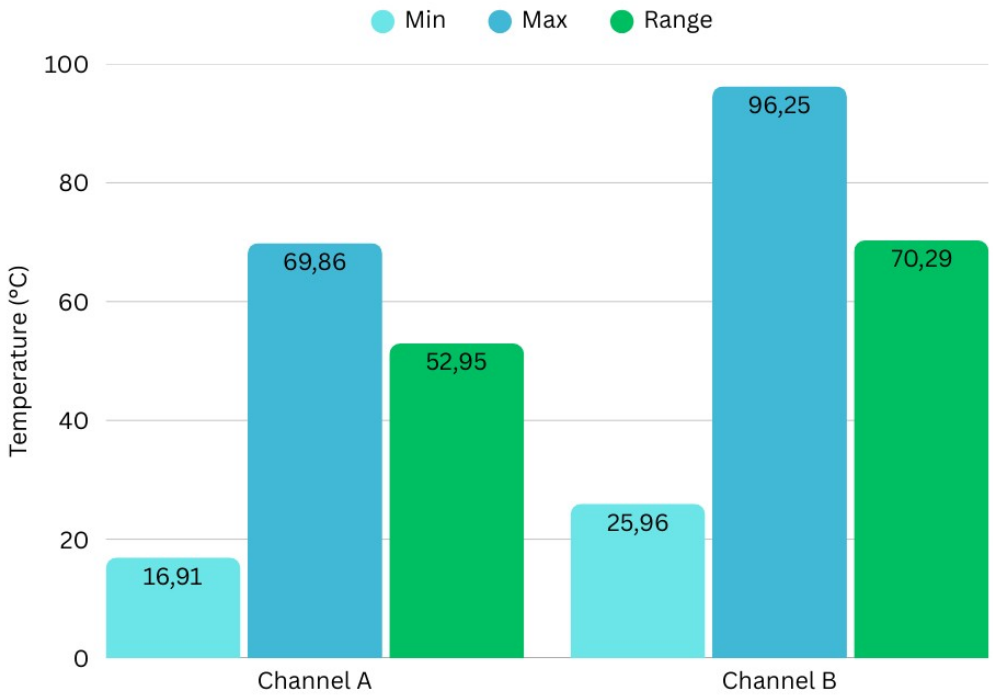
Πίνακας 3.6		
Αισθητήρας	**Θέση**	**Μέση Θερμοκρασία (°C)**
Channel A	Συμπυκνωτής	43.16
Channel B	Λέβητας	66.82

3.5.2 Ελάχιστη, μέγιστη θερμοκρασία ανά αισθητήρα και εύρος τιμών

Ο παρακάτω πίνακας παρουσιάζει την ελάχιστη και τη μέγιστη θερμοκρασία που καταγράφτηκε από τους δύο αισθητήρες, καθώς επίσης και τους υπολογισμούς του εύρους τιμών.

Πίνακας 3.7				
Αισθητήρας	**Θέση**	**Ελάχιστη (°C)**	**Μέγιστη (°C)**	**Εύρος (°C)**
Channel A	Συμπυκνωτής	16.91	69.86	52.95
Channel B	Λέβητας	25.96	96.25	70.29

Ακολουθεί το γράφημα της μέγιστης και ελάχιστης θερμοκρασίας και το εύρος τιμών.



Εικόνα 35: Γράφημα μέγιστης / ελάχιστης θερμοκρασίας και το εύρος τιμών για το δεύτερο πείραμα.

3.5.3 Οι φάσεις της απόσταξης

Κατά τη διάρκεια του πειράματος απόσταξης, παρατηρούνται τέσσερις διακριτές θερμικές φάσεις. Στην αρχική φάση θέρμανσης, οι θερμοκρασίες στα κανάλια B του λέβητα και A του συμπυκνωτή ξεκινούν

από χαμηλά επίπεδα, 26 °C και 17 °C αντίστοιχα, χωρίς να έχουμε παραγωγή ατμού. Στη συνέχεια, στη φάση βρασμού, ο λέβητας φτάνει σε θερμοκρασίες άνω των 90 °C, ενεργοποιώντας τη δημιουργία ατμού που μεταφέρεται προς τον συμπυκνωτή, ο οποίος αρχίζει να καταγράφει αυξημένες τιμές θερμοκρασίας. Κατά τη φάση συμπύκνωσης, ο ατμός ψύχεται αποτελεσματικά, με τις θερμοκρασίες στο κανάλι A να σταθεροποιούνται μεταξύ 60–70 °C, ενώ το υγρό προϊόν αρχίζει να συλλέγεται. Τέλος, στη φάση σταθεροποίησης και ψύξης, οι θερμοκρασίες μειώνονται σταδιακά, με τον λέβητα να πέφτει κάτω από τους 70 °C και τον συμπυκνωτή να επιστρέφει σε χαμηλότερα επίπεδα 35 με 40 °C, σηματοδοτώντας την ολοκλήρωση του θερμικού κύκλου.

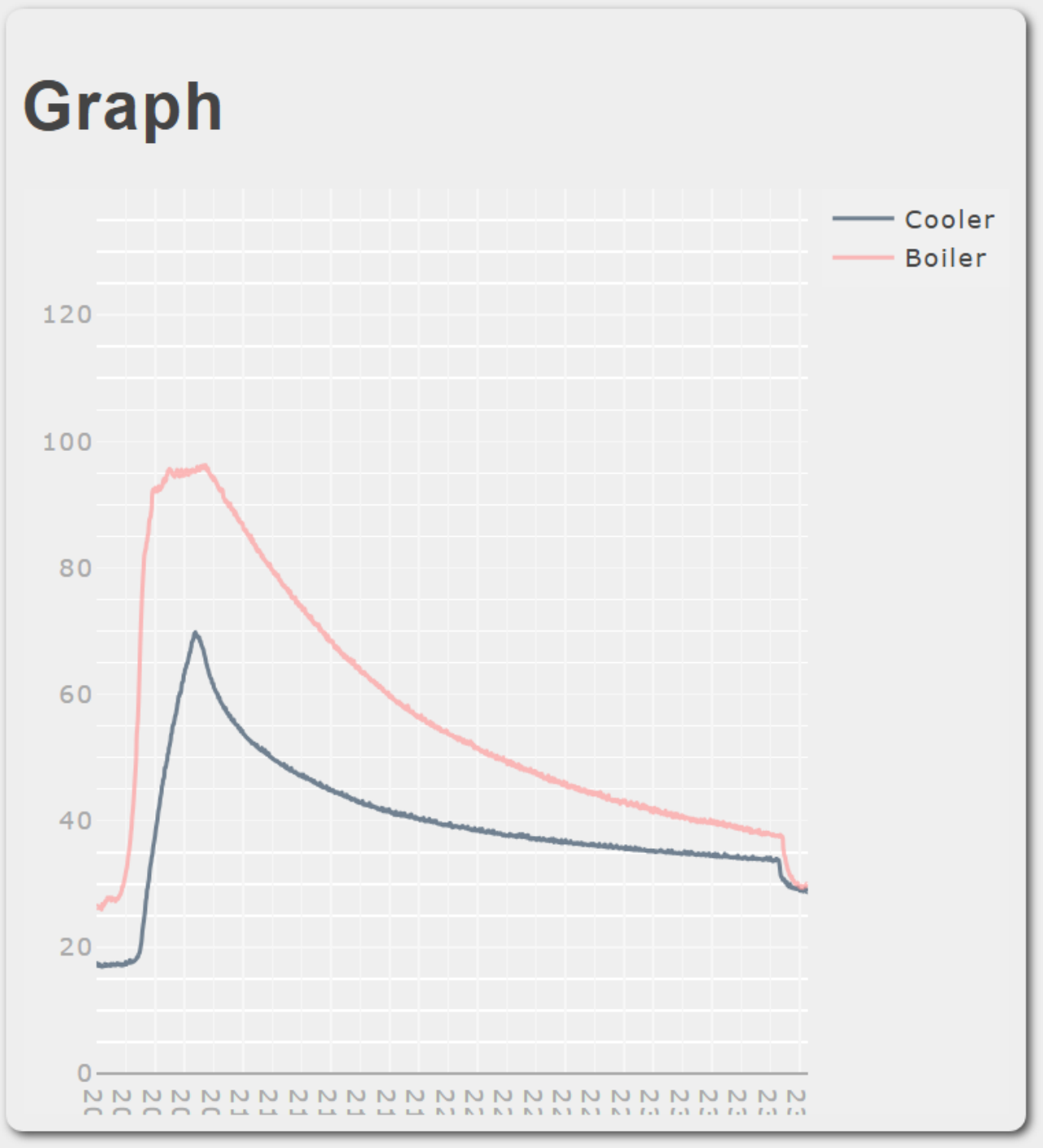


Figure 36: Το γράφημα του δεύτερου πειράματος.

Κεφάλαιο 4ο : Συμπεράσματα

4.1 Συμπεράσματα πρώτου πειράματος

Η επεξεργασία των πειραματικών δεδομένων ανέδειξε διαφοροποιήσεις ως προς τη σταθερότητα, την ακρίβεια και τη λειτουργική αξιοπιστία των χρησιμοποιούμενων αισθητήρων. Οι αισθητήρες τύπου PT100 παρουσίασαν εξαιρετική σταθερότητα, με ελάχιστες διακυμάνσεις μεταξύ των διαδοχικών μετρήσεων και συνεπή απόκριση καθ' όλη τη διάρκεια της δοκιμής. Η συμπεριφορά τους επιβεβαιώνει την καταλληλότητά τους για εφαρμογές που απαιτούν συνεχή και ακριβή παρακολούθηση θερμοκρασίας εντός του θαλάμου.

Ο αισθητήρας DHT22, ο οποίος χρησιμοποιήθηκε για την καταγραφή της θερμοκρασίας και της σχετικής υγρασίας του περιβάλλοντος χώρου, εμφάνισε υψηλή ομοιογένεια στις μετρήσεις του. Οι τιμές του παρέμειναν σταθερές γύρω από τη μέση θερμοκρασία, γεγονός που επιβεβαιώνει τη θερμική σταθερότητα του περιβάλλοντος και την αξιοπιστία του αισθητήρα σε συνθήκες χαμηλής μεταβλητότητας.

Αντιθέτως, το υπέρυθρο θερμόμετρο χειρός παρουσίασε μεγαλύτερη μεταβλητότητα στις μετρήσεις, γεγονός που αποδίδεται σε παραμέτρους όπως η απόσταση μέτρησης, η γωνία πρόσπτωσης και η μορφολογία της επιφάνειας στόχευσης. Παρά τις ενδογενείς αβεβαιότητες της μεθόδου, η χρήση του IR θερμομέτρου συνέβαλε ουσιαστικά στην ενίσχυση της αξιοπιστίας των συνολικών μετρήσεων, προσφέροντας τη δυνατότητα ταχείας και χωρίς επαφή καταγραφή θερμοκρασιών σε εξωτερικές επιφάνειες.

Συνολικά, η συγκριτική αξιολόγηση των αισθητήρων κατέδειξε ότι οι PT100 αποτελούν την πλέον αξιόπιστη επιλογή για συνεχή μέτρηση της θερμοκρασίας εντός του θαλάμου, ενώ οι υπόλοιπες μέθοδοι μπορούν να αξιοποιηθούν επικουρικά, είτε για επιβεβαίωση των μετρήσεων είτε για γρήγορο έλεγχο σε σημεία όπου η εγκατάσταση σταθερού αισθητήρα δεν είναι εφικτή.

4.2 Συμπεράσματα δεύτερου πειράματος

Με βάση την ανάλυση των μετρήσεων, η μέση θερμοκρασία στον λέβητα καταγράφεται σε υψηλά επίπεδα, όπως είναι αναμενόμενο, δεδομένου ότι αποτελεί το σημείο όπου πραγματοποιείται η θέρμανση και παραγωγή ατμού. Αντίθετα, η μέση θερμοκρασία στον συμπυκνωτή παραμένει αισθητά χαμηλότερη, γεγονός που υποδηλώνει ότι η ψύξη και συμπύκνωση του ατμού λειτουργούν αποτελεσματικά, διατηρώντας τη θερμική ισορροπία του συστήματος. Η μέση θερμική διαφορά μεταξύ των δύο καναλιών, η οποία αγγίζει τους 23.66 °C, αποτελεί ισχυρή ένδειξη για την καλή θερμική απόδοση και την αποτελεσματικότητα της διαδικασίας απόσταξης.

Επίσης παρατηρείται ότι στο κανάλι B, το οποίο αντιστοιχεί στον λέβητα, οι θερμοκρασίες προσεγγίζουν το σημείο βρασμού του νερού, 96.25 °C. Η επίτευξη τόσο υψηλών τιμών αποτελεί ένδειξη ότι η θέρμανση του συστήματος είναι αποδοτική, επιτρέποντας την παραγωγή ατμού με συνέπεια και σταθερότητα. Η θερμική απόδοση του λέβητα είναι κρίσιμη για την επιτυχή έναρξη της διαδικασίας απόσταξης.

Αντίθετα, στο κανάλι A, που αντιστοιχεί στον συμπυκνωτή, οι θερμοκρασίες παραμένουν σημαντικά χαμηλότερες, με μέγιστη τιμή 69.86 °C. Το γεγονός αυτό είναι απολύτως αναμενόμενο, καθώς ο συμπυκνωτής έχει ως ρόλο την ψύξη του ατμού και τη μετατροπή του σε υγρή μορφή. Η διατήρηση χαμηλών θερμοκρασιών σε αυτό το σημείο του συστήματος επιβεβαιώνει την αποτελεσματικότητα της ψύξης και την ομαλή λειτουργία της συμπύκνωσης.

Τέλος, το εύρος τιμών που καταγράφηκε σε κάθε κανάλι είναι μεγάλο 52.95 °C για τον συμπυκνωτή και 70.29 °C για τον λέβητα. Αυτή η διακύμανση αποδεικνύει ότι οι αισθητήρες κατέγραψαν ολόκληρο τον θερμικό κύκλο της διαδικασίας: από την αρχική φάση θέρμανσης, στη φάση βρασμού και παραγωγής ατμού, έως την τελική φάση ψύξης και υγροποίησης.

Η θερμοκρασιακή συμπεριφορά των δύο καναλιών αποτυπώνει με ακρίβεια τις φυσικές μεταβολές που συντελούνται κατά την απόσταξη. Η διακριτή μετάβαση από τη θέρμανση στον βρασμό, και από την παραγωγή ατμού στη συμπύκνωση, επιβεβαιώνει την ορθή λειτουργία του συστήματος και την αξιοπιστία των αισθητήρων.

4.3 Γενικά συμπεράσματα

Η παρούσα εργασία ανέδειξε την αποτελεσματικότητα της αξιοποίησης τεχνολογιών και εργαλείων ανοικτού κώδικα στην ανάπτυξη εφαρμογών βιομηχανικού χαρακτήρα, προσφέροντας σημαντικά πλεονεκτήματα όπως ευελιξία σχεδιασμού, χαμηλό κόστος υλοποίησης και δυνατότητα επεκτασιμότητας. Η σχεδιασμένη πλακέτα επέκτασης επιδεικνύει πλήρη συμβατότητα με όλα τα μοντέλα Raspberry Pi, καθώς και με μικροελεγκτές που υποστηρίζουν τα πρωτόκολλα επικοινωνίας SPI και I^2C, γεγονός που διευρύνει το φάσμα των πιθανών εφαρμογών της σε συστήματα αυτοματισμού και ελέγχου. Επιπλέον, η εργονομική και λειτουργικά αποδοτική κατασκευή της συμβάλλει ουσιαστικά στην ευκολία συντήρησης, επισκευής και μελλοντικής αναβάθμισης, καθιστώντας την κατάλληλη για χρήση σε απαιτητικά τεχνικά περιβάλλοντα.

4.4 Προτάσεις βελτίωσης

Για την περαιτέρω ενίσχυση της κατασκευής, προτείνονται παρεμβάσεις που αποσκοπούν στη βελτιστοποίηση της ακρίβειας, της απόδοσης και του βαθμού αυτοματοποίησης του συστήματος. Η εφαρμογή χρήστη να εμπλουτιστεί με προηγμένες δυνατότητες στατιστικής επεξεργασίας, επιτρέποντας την εις βάθος ανάλυση των μετρήσεων και την εξαγωγή πιο τεκμηριωμένων συμπερασμάτων. Η χρήση ηλεκτρονικών εξαρτημάτων υψηλής ακρίβειας, καθώς και η επιλογή ποιοτικότερων υλικών κατασκευής, αναμένεται να συμβάλει ουσιαστικά στη βελτίωση της σταθερότητας και της αξιοπιστίας των καταγραφών.

Επιπλέον, η ενσωμάτωση αποδοτικότερου συστήματος ψύξης, αυτόματου μηχανισμού πλήρωσης της δεξαμενής και αισθητήρων στάθμης για τον έλεγχο ορίων υψηλής και χαμηλής στάθμης, θα επιτρέψει την πλήρη αυτοματοποίηση της διαδικασίας, μειώνοντας την ανάγκη για ανθρώπινη παρέμβαση. Τέλος, η δυνατότητα επιλογής μεταξύ δύο τύπων αισθητήρων θερμοκρασίας, PT100 και PT1000, θα προσφέρει ευελιξία και προσαρμοστικότητα στις απαιτήσεις διαφορετικών εφαρμογών, ενισχύοντας τη λειτουργική επεκτασιμότητα του συστήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] H. D. Young, *Πανεπιστημιακή Φυσική, Τόμος Α, Μηχανική – Θερμοδυναμική*, 8ή Έκδοση, Εκδόσεις Παπαζήση, 1992.
[2] Α. Α. Ζήσος, *Φυσική Ι, Μηχανική – Θερμότητα*, 2ή Έκδοση, Σύγχρονη Εκδοτική, 2006.
[3] Κ. Καλοβρέκτης, Ν. Κατέβας, *Αισθητήρες Μέτρησης και Ελέγχου*, 2ή Έκδοση, Εκδόσεις Τζιόλα, 2017.
[4] Μ. Ν. Σπάσος, *Αναλογική Επεξεργασία Σημάτων Αισθητηρίων*, Εκδόσεις Αϊβάζη, 2018.
[5] A. P. Malvino, *Βασική Ηλεκτρονική*, 4ή Έκδοση, Εκδόσεις Τζιόλα, 1999.
[6] Sedra, Smith, *Μικροηλεκτρονικά Κυκλώματα*, Εκδόσεις Παπασωτηρίου, 1993.
[7] Kaufman, Seidman, *Εγχειρίδιο Ηλεκτρονικής*, 2ή Έκδοση, Εκδόσεις Τζιόλα, 1992.
[8] Δ. Ρήγας, *Τεχνολογία Ηλεκτρονικών Εξαρτημάτων*, Εκδόσεις Τζιόλα, 1995.
[9] Χ. Καραΐσκος, *Ο μικροελεγκτής 8051*, Εκδόσεις Σύγχρονη Εκδοτική, 2010.
[10] Δρ. Σταμάτης Αλατσαθιανός, *Ανάπτυξη Συστημάτων Με Μικροελεγκτές 8051*, 1ή Έκδοση, Έκδοση Σταμάτης Αλατσαθιανός,
[11] L. Lemay, R. Colburn, J. Kyrnin, *Πλήρες Εγχειρίδιο HTML 5, CSS & JavaScript*, 7ή Έκδοση, Εκδόσεις Μ. Γκιούρδας, 2016.
[12] M. Moncur, *Μάθετε την JavaScript σε 24 ώρες*, 4ή Έκδοση, Εκδόσεις Μ. Γκιούρδας, 2007.
[13] Μ. Καφές, *Εξερεύνηση της python*, Εκδόσεις Κλειδάριθμος, 2017.
[14] J. C. Melonie, *Μάθετε PHP, MySQL και Apache Όλα σε Ένα*, 5η Έκδοση, Εκδόσεις Μ. Γκιούρδας, 2014.
[15] J. Edmonds, *Αλγόριθμοι*, Εκδόσεις Κριτική, 2016.
[16] Γ. Καραγιάννης, Κ. Τζιτζιράχου, *Εισαγωγή στα Σήματα και Συστήματα*, Εκδόσεις Παπασωτηρίου, 2003.

Data Sheet

[17] Maxim Integrated Products, Inc., "RTD-to-Digital Converter", MAX31865 datasheet, Nov. 2015.
[18] Maxim Integrated Products, Inc., "MAX31865 Evaluation Kit ", datasheet, Nov. 2012.

Internet Site

[19] MDN Web Docs, [Online]. Available: https://developer.mozilla.org/en-US/.
[20] W3Schools, [Online]. Available: https://www.w3schools.com/.
[21] Microsoft Learn [Online]. Available: https://learn.microsoft.com/en-us/.
[22] Node.js documentation, [Online]. Available: https://nodejs.org/docs/latest/api/.
[23] JavaScript documentation, [Online]. Available: https://devdocs.io/javascript/.
[24] MySQL documentation, [Online]. Available: https://dev.mysql.com/doc/.
[25] Python documentation, [Online]. Available: https://docs.python.org/3/.
[26] RPLCD Python library documentation, [Online]. Available: https://rplcd.readthedocs.io/en/stable/index.html.
[27] gpiozero Python library documentation, [Online]. Available: https://gpiozero.readthedocs.io/en/latest/index.html.
[28] schedule Python library documentation [Online]. Available: https://schedule.readthedocs.io/en/stable/.
[29] Embedded JavaScript library documentation [Online]. Available: https://www.npmjs.com/package/ejs.
[30] Raspberry Pi documentation [Online]. Available: https://www.raspberrypi.com/.

Paper in Conference Proceedings

[31] Xiujun Li, Gerard C.M. Meijer, "A Low-Cost, High-Precision Sensor Interface for Platinum Temperature Sensors", Faculty of Information Technology and Systems, Delfl University of Technology, IEEE, 2003.
[32] Jiguang Liu, Yukun Li, Hongyan Zhao, "A Temperature Measurement System Based on PT100", Liaoning Provincial College of Communications, International Conference on Electrical and Control Engineering, Shenyang, China, IEEE, 2010.
[33] Jun Zhang, Shengjie Jiao, Min Ye, Xiaodong Zhang, Member, IEEE, Jiangcheng Chen, Bin Pang, Jinping Li, Xinxin Xu, "Multichannel Subgrade Temperature Acquisition System Based on LabVIEW and

Serial Communication", IEEE International Conference on Automation Science and Engineering (CASE), 2013.
[34] Yu Qian, Zhongming Luo, Zhuofu Liu, Hailu Zhao The higher educational key laboratory for Measuring & Control Technology and Instrumentations of Heilongjiang Province, Chenyang Li, Yang Song, Desheng Nan, Jingwei Wei School of Measurement–Control Technology and Communications Engineering, "Application of RTD sensor in the real time measurement and wireless transmission", 4th International Conference on Instrumentation and Measurement, Computer, Communication and Control, Harbin University of Science and Technology Harbin, China, 2014.
[35] Daniel F. Merchan, Jonnathan A. Peralta, Andres Vazquez-Rodas, Luis I. Minchala, Darwin Astudillo-Salinas, "Open source SCADA system for advanced monitoring of industrial processes", International Conference on Information Systems and Computer Science 2017, Universidad de Cuenca, Department of Electrical, Electronic and Telecommunications Engineering, Cuenca, Ecuador, IEEE, 2017.
[36] Wenli Zhang, Tingting Cheng, Huamin Chen, Xiang Guo, Guohua Gao, "Design of Whole Chain Temperature Monitoring System for Raw Milk", Proceedings of ICSP2018, The college of information and communications engineering, Faculty of Information Technology, Beijing University of Technology, Beijing, China, The college of mechanical Engineering and Applied Electronics Technology, Beijing University of Technology,Beijing ,China, IEEE, 2018.
[37] Tipparat Junsing, "Interface Circuit for Three-Wire Resistance Temperature Detector with Lead Wire Resistance Compensation", College of Industrial Technology, King Mongkut's University of Technology North Bangkok Bangsue, Bangkok, Thailand, IEEE, 2019.
[38] Qing Wang 1, Gang Lu Wang 1, Xin Xiu Xie 1, LiLi Zhou, "Design and simulation for temperature measurement and control system based on PT100", IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2019), School of electrical and energy engineering, Nantong Institute of technology, Nantong 226002, China, IEEE, 2019.

Journal Articles

[39] Fred Lacy, "Evaluating the Resistivity-Temperature Relationship for RTDs and Other Conductors", IEEE Sensors Journal, Vol. 11, No. 5, May 2011.
[40] Alan Shen, Member, IEEE, Seung Bum Kim, Callum Bailey, Anson W. K. Ma, and Sameh Dardona, "Direct Write Fabrication of Platinum-Based Thick-Film Resistive Temperature Detectors", IEEE Sensors Journal, Vol. 18, No. 22, Nov 15, 2018.

ΠΑΡΑΡΤΗΜΑ Α: Χαρακτηριστικά αποστακτήρα

Πίνακας Α.1	
Χαρακτηριστικό	**Περιγραφή**
Κατασκευαστής / Μοντέλο	VEVOR Moonshine Still Distiller – 11,4 L / 3 Gal
Υλικά Κατασκευής	Ανοξείδωτο ατσάλι, κόκκινος χαλκός
Χωρητικότητα λέβητα	11,4 L / 3 Gal
Χωρητικότητα συμπυκνωτή	3,5 L / 0.92 Gal
Διαστάσεις συμπυκνωτή	20 x 11 cm
Πάχος λέβητα	0.7 mm
Διάμετρος πηνίου χαλκού	10 mm
Βάρος	3,74 kg
Επιπλέον χαρακτηριστικά	Ενσωματωμένο θερμόμετρο, σφράγιση με σιλικόνη, ευέλικτη χρήση.
Σελίδα κατασκευαστή	https://eur.vevor.com/alcohol-distiller-c_10688/moonshine-3-gal-2pots-distiller-copper-wine-maker-water-still-boiler-p_010299814502

Λίστα Εργαλείων Λογισμικού

Στο πλαίσιο της ανάπτυξης και υλοποίησης της κατασκευής, αξιοποιήθηκε πλήθος εργαλείων λογισμικού, τα οποία κάλυψαν τις ανάγκες σχεδίασης, προγραμματισμού, απομακρυσμένης διαχείρισης, τεκμηρίωσης και δοκιμών. Ο ακόλουθος πίνακας συνοψίζει τα εργαλεία που χρησιμοποιήθηκαν.

Πίνακας Α.2		
Εργαλείο / Λογισμικό	**Χρήση στην κατασκευή**	**Σχόλια / Παρατηρήσεις**
Inkscape	Σχεδίαση διαγραμμάτων, σχημάτων	Ελεύθερο λογισμικό
GIMP	Επεξεργασία φωτογραφιών, εικόνων	Ελεύθερο λογισμικό
draw.io	Σχεδίαση μπλοκ διαγραμμάτων και flowchart	Web εφαρμογή με φιλικό περιβάλλον χρήστη και δωρεάν χρήση με βασικά εργαλεία.
canva.com	Σχεδιασμός των γραφημάτων	Web εφαρμογή με φιλικό περιβάλλον χρήστη και δωρεάν χρήση με βασικά εργαλεία.
KiCad	Σχεδίαση της πλακέτας PCB	Ελεύθερο λογισμικό
Visual Studio Code	Ανάπτυξη και επεξεργασία κώδικα σε Python, HTML, JavaScript	Ελεύθερο λογισμικό, υποστήριξη επεκτάσεων και debugging
WinSCP	Απομακρυσμένη σύνδεση με Raspberry Pi	Ελεύθερο λογισμικό, χρήση πρωτοκόλλου SCP/SSH
LibreOffice Writer	Σύνταξη και μορφοποίηση της διπλωματικής	Ελεύθερο λογισμικό, εναλλακτική του MS Word
LibreOffice Calc	Μαθηματικοί υπολογισμοί	Ελεύθερο λογισμικό, εναλλακτική του MS Excel
MySQL Workbench	Διαχείριση της βάσης δεδομένων	Ελεύθερο λογισμικό, οπτική διεπαφή για MySQL
Web Browsers	Δοκιμή εφαρμογής χρήστη (UI/UX)	Ελεύθερο λογισμικό, Chrome, Brave, Edge, Safari, Firefox

ΠΑΡΑΡΤΗΜΑ Β: Οι μετρήσεις του δεύτερου πειράματος.

ΠΙΝΑΚΑΣ Β.1		
TIME	**Channel A (°C)**	**Channel B (°C)**
20:23:20	17.06	26.46
20:23:30	17.46	26.54
20:23:40	17.13	26.24
20:23:50	17.02	26.20
20:24:00	17.06	26.42
20:24:10	17.32	26.24
20:24:20	17.21	26.28
20:24:30	17.06	26.20
20:24:40	16.95	26.17
20:24:49	16.95	25.96
20:25:00	17.02	26.83
20:25:10	16.91	26.87
20:25:20	17.06	26.49
20:25:30	17.02	26.76
20:25:40	17.32	26.81
20:25:50	17.32	27.06
20:26:00	17.02	27.31
20:26:10	17.17	27.28
20:26:20	17.24	27.44
20:26:29	17.17	27.82
20:26:40	17.10	27.63
20:26:50	17.13	27.79
20:27:00	17.10	27.83
20:27:10	17.10	27.86
20:27:20	17.06	27.68
20:27:30	17.35	27.59
20:27:40	17.10	27.33
20:27:50	17.21	27.86
20:28:00	17.28	27.57
20:28:10	17.35	27.66
20:28:20	17.24	27.60
20:28:30	17.17	27.52

20:28:40	17.21	27.60
20:28:50	17.17	27.24
20:29:00	17.13	27.60
20:29:10	17.24	27.71
20:29:20	17.50	27.75
20:29:30	17.28	27.57
20:29:40	17.32	27.71
20:29:50	17.21	27.97
20:30:00	17.35	28.16
20:30:10	17.17	28.23
20:30:20	17.21	28.45
20:30:30	17.32	28.75
20:30:40	17.17	29.00
20:30:50	17.10	29.67
20:31:00	17.24	29.52
20:31:10	17.32	30.00
20:31:21	17.28	30.33
20:31:30	17.21	30.85
20:31:40	17.24	31.25
20:31:50	17.69	31.81
20:32:00	17.28	32.25
20:32:10	17.50	32.69
20:32:20	17.54	33.39
20:32:30	17.54	34.35
20:32:40	17.54	34.76
20:32:50	17.54	35.39
20:33:01	17.98	36.16
20:33:10	17.54	36.98
20:33:20	17.69	38.19
20:33:30	17.57	39.01
20:33:40	17.80	40.49
20:33:50	17.69	41.41
20:34:00	17.76	42.52
20:34:10	17.72	43.78
20:34:20	17.87	45.26

20:34:31	17.91	46.78
20:34:41	18.02	48.26
20:34:51	18.13	49.86
20:35:00	18.24	53.61
20:35:10	18.46	54.79
20:35:20	18.42	55.91
20:35:30	18.79	58.21
20:35:40	18.97	61.15
20:35:50	19.16	64.28
20:36:01	19.60	67.56
20:36:11	20.19	70.39
20:36:21	20.70	72.82
20:36:31	21.58	74.99
20:36:40	22.65	76.93
20:36:50	23.35	78.65
20:37:00	23.94	80.07
20:37:10	24.64	81.76
20:37:20	25.38	82.36
20:37:31	26.41	82.79
20:37:41	27.41	83.11
20:37:51	27.85	83.98
20:38:01	29.10	84.19
20:38:11	29.36	85.02
20:38:20	30.06	85.26
20:38:30	30.54	86.48
20:38:40	31.50	87.56
20:38:50	32.42	87.84
20:39:01	32.98	88.03
20:39:11	33.46	88.76
20:39:21	34.20	89.41
20:39:31	34.53	91.52
20:39:41	35.31	92.37
20:39:51	36.08	92.11
20:40:01	36.53	92.04
20:40:10	37.04	92.14

20:40:20	37.67	92.67
20:40:31	38.45	92.42
20:40:41	39.08	92.09
20:40:51	39.60	92.15
20:41:01	40.19	92.32
20:41:11	41.08	92.11
20:41:21	41.71	92.89
20:41:31	42.11	92.87
20:41:41	42.71	92.60
20:41:50	43.52	92.34
20:42:01	44.00	92.64
20:42:11	44.48	92.71
20:42:21	45.49	92.88
20:42:31	45.74	93.21
20:42:41	46.56	93.63
20:42:51	46.67	94.07
20:43:01	47.08	93.50
20:43:11	48.45	93.69
20:43:21	48.30	93.84
20:43:31	48.71	93.76
20:43:41	49.31	94.68
20:43:51	49.60	94.67
20:44:01	50.53	95.34
20:44:11	50.64	95.11
20:44:21	51.50	95.51
20:44:31	51.79	95.69
20:44:41	52.24	95.45
20:44:51	52.76	95.52
20:45:01	53.69	95.38
20:45:11	53.87	95.12
20:45:21	54.69	94.85
20:45:31	55.18	94.70
20:45:41	55.18	94.55
20:45:51	55.55	94.67
20:46:01	56.07	94.37

20:46:11	56.59	94.74
20:46:21	56.81	95.08
20:46:31	57.30	95.01
20:46:42	57.85	95.48
20:46:51	58.52	95.49
20:47:01	58.90	94.89
20:47:11	59.75	94.59
20:47:21	59.64	94.44
20:47:31	60.27	94.52
20:47:41	60.09	94.67
20:47:51	60.38	94.68
20:48:01	61.20	95.58
20:48:11	61.84	95.17
20:48:22	61.91	94.52
20:48:32	62.25	95.15
20:48:41	63.14	94.60
20:48:51	63.25	94.63
20:49:01	63.92	95.08
20:49:11	64.11	95.38
20:49:21	64.30	95.49
20:49:31	64.89	95.12
20:49:41	64.82	95.49
20:49:52	65.19	94.97
20:50:02	65.71	94.67
20:50:12	66.09	95.31
20:50:21	66.65	95.02
20:50:31	66.50	95.24
20:50:41	67.21	95.37
20:50:51	67.54	95.08
20:51:01	68.25	95.23
20:51:11	68.33	95.57
20:51:22	68.66	95.27
20:51:32	68.92	95.46
20:51:42	69.60	95.23
20:51:52	69.22	95.34

20:52:01	69.86	95.19
20:52:11	69.52	95.31
20:52:21	69.37	95.38
20:52:31	69.30	95.91
20:52:41	69.00	95.98
20:52:52	68.96	95.53
20:53:02	68.81	95.76
20:53:12	69.00	95.38
20:53:22	68.33	95.68
20:53:32	68.44	95.83
20:53:42	67.95	96.10
20:53:51	67.65	95.98
20:54:01	67.47	95.79
20:54:11	67.24	95.83
20:54:22	67.06	96.21
20:54:32	66.57	95.83
20:54:42	66.16	95.76
20:54:52	65.83	95.91
20:55:02	65.42	96.15
20:55:12	64.89	95.71
20:55:22	64.63	95.61
20:55:31	64.22	95.31
20:55:41	64.00	95.79
20:55:52	63.63	95.16
20:56:02	63.22	95.00
20:56:12	62.96	94.97
20:56:22	62.66	94.67
20:56:32	62.43	94.55
20:56:42	62.47	94.33
20:56:52	61.80	94.59
20:57:02	61.73	94.29
20:57:11	61.69	93.91
20:57:21	61.17	93.95
20:57:32	60.91	94.25
20:57:42	60.79	93.80

20:57:52	60.53	93.91
20:58:02	60.38	93.46
20:58:12	60.27	93.43
20:58:22	60.01	93.13
20:58:32	60.05	93.01
20:58:42	59.42	92.67
20:58:52	59.38	92.52
20:59:02	59.53	92.34
20:59:12	58.97	92.19
20:59:22	58.82	92.52
20:59:32	58.67	92.49
20:59:42	58.52	92.36
20:59:52	58.34	92.38
21:00:02	58.26	91.77
21:00:12	57.96	91.13
21:00:22	57.78	90.98
21:00:32	57.70	90.83
21:00:42	57.52	90.46
21:00:52	57.82	90.50
21:01:02	57.26	90.61
21:01:12	57.11	90.23
21:01:22	56.92	90.38
21:01:32	57.03	89.97
21:01:42	56.77	89.71
21:01:52	56.48	90.01
21:02:03	56.44	89.63
21:02:12	56.66	90.20
21:02:22	56.22	89.48
21:02:32	55.99	89.18
21:02:42	55.92	89.26
21:02:52	55.77	89.33
21:03:02	55.66	89.18
21:03:12	55.99	88.99
21:03:22	55.47	88.36
21:03:32	55.36	88.92

21:03:43	55.29	88.66
21:03:53	55.14	88.54
21:04:02	55.10	88.17
21:04:12	55.10	87.87
21:04:22	55.06	87.76
21:04:32	54.65	87.46
21:04:42	54.58	87.72
21:04:52	54.69	87.27
21:05:02	54.69	87.26
21:05:13	54.13	87.20
21:05:23	54.10	87.16
21:05:33	54.06	86.93
21:05:42	54.28	87.12
21:05:52	53.76	86.56
21:06:02	53.76	86.11
21:06:12	53.61	86.03
21:06:22	53.54	85.88
21:06:32	53.43	85.77
21:06:43	53.24	86.03
21:06:53	53.21	85.62
21:07:03	53.06	85.47
21:07:13	52.98	85.39
21:07:22	52.95	85.06
21:07:32	52.87	85.02
21:07:42	52.69	85.21
21:07:52	52.76	84.49
21:08:02	52.61	84.91
21:08:13	52.54	85.09
21:08:23	52.24	84.08
21:08:33	52.31	84.23
21:08:43	52.35	84.53
21:08:53	52.16	83.89
21:09:03	52.20	83.67
21:09:12	52.09	83.67
21:09:22	51.94	83.71

21:09:32	52.02	83.11
21:09:42	51.72	83.48
21:09:53	52.02	82.70
21:10:03	51.79	82.73
21:10:13	51.50	82.58
21:10:23	51.46	82.88
21:10:33	51.35	82.70
21:10:43	51.35	82.58
21:10:52	51.27	82.40
21:11:02	51.12	82.28
21:11:13	51.53	81.76
21:11:23	51.42	81.87
21:11:33	51.09	82.02
21:11:43	50.94	81.42
21:11:53	50.98	81.46
21:12:03	50.79	81.38
21:12:13	51.20	81.20
21:12:23	50.68	81.05
21:12:32	50.61	81.01
21:12:43	50.49	80.75
21:12:53	50.79	80.82
21:13:03	50.46	80.56
21:13:13	50.38	80.30
21:13:23	50.61	80.22
21:13:33	50.23	80.34
21:13:43	50.09	80.64
21:13:53	50.09	80.19
21:14:03	50.05	79.74
21:14:13	49.94	79.85
21:14:23	49.83	79.70
21:14:33	49.83	79.48
21:14:43	49.75	79.25
21:14:53	49.64	79.06
21:15:03	49.60	79.21
21:15:13	49.53	79.29

21:15:23	49.42	78.80
21:15:33	49.38	78.73
21:15:43	49.49	78.76
21:15:53	49.31	78.58
21:16:03	49.31	78.88
21:16:13	49.19	78.32
21:16:23	49.19	78.09
21:16:33	49.08	78.02
21:16:43	49.01	78.09
21:16:53	48.82	77.75
21:17:03	48.82	77.68
21:17:13	48.79	77.34
21:17:24	48.75	77.38
21:17:33	48.79	77.31
21:17:43	49.01	77.01
21:17:53	48.64	77.23
21:18:03	48.45	77.08
21:18:13	48.53	76.82
21:18:23	48.30	76.82
21:18:33	48.67	76.74
21:18:43	48.23	76.45
21:18:53	48.30	76.74
21:19:04	48.16	76.18
21:19:14	48.16	76.07
21:19:23	48.04	75.96
21:19:33	47.93	76.37
21:19:43	48.01	75.88
21:19:53	48.34	75.44
21:20:03	47.90	75.21
21:20:13	47.82	75.47
21:20:23	47.82	75.17
21:20:34	47.75	75.29
21:20:44	47.64	75.03
21:20:54	47.64	75.25
21:21:03	47.75	75.32

21:21:13	47.60	74.46
21:21:23	47.41	74.65
21:21:33	47.56	74.80
21:21:43	47.38	74.43
21:21:53	47.19	74.58
21:22:03	47.30	73.94
21:22:14	47.12	74.02
21:22:24	47.23	74.17
21:22:34	47.15	73.98
21:22:44	47.34	74.02
21:22:53	47.08	73.90
21:23:03	46.97	73.31
21:23:13	46.89	73.38
21:23:23	47.34	73.57
21:23:33	46.86	73.46
21:23:44	46.89	73.57
21:23:54	47.01	73.23
21:24:04	46.97	72.75
21:24:14	46.60	72.71
21:24:24	46.71	72.75
21:24:33	46.89	72.52
21:24:43	46.67	72.52
21:24:53	46.60	72.19
21:25:03	46.56	72.45
21:25:14	46.45	72.00
21:25:24	46.38	71.92
21:25:34	46.56	72.30
21:25:44	46.49	72.22
21:25:54	46.38	71.59
21:26:04	46.15	71.63
21:26:13	46.26	71.36
21:26:23	46.04	71.44
21:26:33	46.34	71.10
21:26:44	46.08	71.10
21:26:54	45.93	71.03

21:27:04	45.97	71.18
21:27:14	45.86	70.92
21:27:24	45.78	70.92
21:27:34	45.71	71.03
21:27:44	45.71	70.69
21:27:54	45.67	71.03
21:28:03	45.60	70.13
21:28:14	46.00	70.51
21:28:24	45.41	70.39
21:28:34	45.49	70.17
21:28:44	45.71	69.95
21:28:54	45.41	69.98
21:29:04	45.26	69.87
21:29:14	45.30	69.65
21:29:24	45.26	69.46
21:29:34	45.23	69.76
21:29:43	45.26	69.61
21:29:54	45.19	69.05
21:30:04	45.49	69.09
21:30:14	45.15	68.68
21:30:24	44.97	68.60
21:30:34	44.93	69.31
21:30:44	44.82	68.60
21:30:54	45.15	68.39
21:31:04	44.89	68.37
21:31:14	44.93	68.32
21:31:24	44.82	68.59
21:31:34	44.71	68.45
21:31:44	44.71	68.19
21:31:54	44.74	68.38
21:32:04	44.85	67.93
21:32:14	44.63	67.78
21:32:24	44.52	67.37
21:32:34	44.41	67.34
21:32:44	44.48	67.52

21:32:55	44.56	67.60
21:33:04	44.45	67.11
21:33:14	44.41	67.34
21:33:24	44.48	67.07
21:33:34	44.30	67.34
21:33:44	44.22	66.81
21:33:54	44.34	66.78
21:34:04	44.30	66.89
21:34:14	44.15	66.56
21:34:24	44.15	66.40
21:34:35	44.48	66.37
21:34:44	44.08	66.48
21:34:54	43.93	66.33
21:35:04	44.00	65.88
21:35:14	43.85	66.29
21:35:24	43.93	65.84
21:35:34	43.93	65.73
21:35:44	44.11	65.96
21:35:54	44.11	65.99
21:36:05	43.67	65.88
21:36:15	43.74	65.66
21:36:24	43.78	65.32
21:36:34	43.67	65.40
21:36:44	43.74	65.40
21:36:54	43.48	65.14
21:37:04	43.45	65.62
21:37:14	43.56	65.36
21:37:24	43.30	65.25
21:37:35	43.37	64.80
21:37:45	43.41	65.14
21:37:55	43.34	64.88
21:38:05	43.41	65.36
21:38:14	43.48	64.61
21:38:24	43.30	64.61
21:38:34	43.11	64.17

21:38:44	43.11	64.24
21:38:54	43.00	64.43
21:39:04	43.19	64.24
21:39:15	43.04	64.47
21:39:25	42.85	63.94
21:39:35	42.85	64.28
21:39:45	43.00	64.13
21:39:54	42.82	63.72
21:40:04	42.85	63.42
21:40:14	42.89	63.56
21:40:24	42.82	63.50
21:40:34	43.15	63.31
21:40:45	42.56	63.27
21:40:55	42.74	63.53
21:41:05	42.63	63.24
21:41:15	42.48	63.46
21:41:25	42.52	63.01
21:41:34	42.59	63.09
21:41:44	42.45	63.05
21:41:54	42.45	62.98
21:42:04	42.48	62.83
21:42:14	42.52	62.75
21:42:25	42.34	62.60
21:42:35	42.82	62.34
21:42:45	42.48	62.57
21:42:55	42.37	62.27
21:43:05	42.19	62.12
21:43:15	42.22	62.16
21:43:24	42.37	62.12
21:43:34	42.34	62.01
21:43:44	42.19	62.19
21:43:55	42.15	61.82
21:44:05	42.15	62.01
21:44:15	42.08	61.78
21:44:25	42.15	61.56

21:44:35	42.11	61.49
21:44:45	42.08	61.77
21:44:55	41.82	61.44
21:45:04	41.89	61.40
21:45:14	41.89	61.08
21:45:24	41.85	61.15
21:45:35	41.74	61.38
21:45:45	41.78	61.19
21:45:55	41.82	61.15
21:46:05	42.04	60.89
21:46:15	41.67	60.97
21:46:25	41.52	60.93
21:46:35	41.93	60.52
21:46:45	41.52	60.44
21:46:54	41.67	60.56
21:47:05	41.56	60.37
21:47:15	41.67	60.30
21:47:25	41.45	60.15
21:47:35	41.71	60.07
21:47:45	41.45	60.18
21:47:55	41.48	59.92
21:48:05	41.45	60.33
21:48:15	41.82	59.74
21:48:25	41.41	59.55
21:48:34	41.37	59.74
21:48:45	41.45	59.48
21:48:55	41.19	59.51
21:49:05	41.22	59.74
21:49:15	41.22	59.25
21:49:25	41.30	59.41
21:49:35	41.15	59.19
21:49:45	40.93	59.16
21:49:55	41.19	59.12
21:50:05	41.08	58.96
21:50:14	41.45	58.96

21:50:25	40.89	58.81
21:50:35	41.04	58.64
21:50:45	40.97	58.59
21:50:55	40.93	58.78
21:51:05	40.97	58.59
21:51:15	40.93	58.81
21:51:25	41.26	58.36
21:51:35	40.78	58.55
21:51:45	40.93	58.44
21:51:55	41.19	58.21
21:52:05	40.82	58.40
21:52:15	41.04	58.03
21:52:25	40.89	57.95
21:52:35	40.78	57.89
21:52:45	41.19	57.87
21:52:55	40.89	57.80
21:53:05	40.78	57.66
21:53:15	40.71	58.14
21:53:26	40.67	57.58
21:53:35	40.74	57.58
21:53:45	40.60	58.21
21:53:55	40.63	57.47
21:54:05	40.78	57.62
21:54:15	40.52	57.32
21:54:25	40.60	57.21
21:54:35	40.97	57.14
21:54:45	40.97	57.21
21:54:56	40.56	57.06
21:55:06	40.67	57.30
21:55:15	40.52	56.91
21:55:25	40.48	56.93
21:55:35	40.48	56.89
21:55:45	40.34	56.70
21:55:55	40.63	56.69
21:56:05	40.34	56.50

21:56:15	40.37	56.69
21:56:26	40.41	56.32
21:56:36	40.30	56.65
21:56:46	40.37	56.39
21:56:56	40.30	56.35
21:57:05	40.22	56.17
21:57:15	40.08	56.35
21:57:25	40.30	56.17
21:57:35	40.26	56.73
21:57:45	40.22	56.21
21:57:55	40.26	55.87
21:58:06	40.11	56.17
21:58:16	40.48	55.98
21:58:26	40.00	56.06
21:58:36	40.08	56.13
21:58:45	39.89	56.13
21:58:55	40.11	55.80
21:59:05	40.00	55.57
21:59:15	40.04	55.39
21:59:25	39.93	55.54
21:59:36	39.93	55.39
21:59:46	40.04	55.39
21:59:56	40.34	55.57
22:00:06	39.93	55.17
22:00:16	39.85	55.69
22:00:25	39.82	55.02
22:00:35	39.89	55.31
22:00:45	39.97	55.02
22:00:55	39.89	55.43
22:01:06	39.74	54.94
22:01:16	39.89	55.20
22:01:26	40.00	54.83
22:01:36	39.82	54.68
22:01:46	39.78	54.72
22:01:56	39.97	54.83

22:02:06	39.63	54.91
22:02:15	39.56	54.76
22:02:25	39.60	54.50
22:02:36	39.56	54.83
22:02:46	39.48	54.65
22:02:56	39.52	54.61
22:03:06	39.45	54.68
22:03:16	39.56	54.16
22:03:26	39.48	54.31
22:03:36	39.37	54.16
22:03:46	39.30	54.24
22:03:55	39.41	54.20
22:04:06	39.34	54.05
22:04:16	39.30	54.16
22:04:26	39.45	54.01
22:04:36	39.30	54.24
22:04:46	39.34	54.27
22:04:56	39.37	54.16
22:05:06	39.34	54.24
22:05:16	39.23	53.79
22:05:26	39.34	53.68
22:05:35	39.23	53.72
22:05:46	39.63	53.64
22:05:56	39.48	53.64
22:06:06	39.63	53.53
22:06:16	39.19	53.57
22:06:26	39.04	53.53
22:06:36	38.93	53.57
22:06:46	39.19	53.49
22:06:56	38.97	53.31
22:07:06	39.04	53.20
22:07:16	39.15	53.23
22:07:26	39.11	53.27
22:07:36	39.15	53.23
22:07:46	39.08	53.01

22:07:56	39.15	53.23
22:08:06	39.04	52.97
22:08:16	39.08	52.75
22:08:26	38.97	52.86
22:08:36	38.93	53.16
22:08:47	38.86	52.90
22:08:56	38.86	53.05
22:09:06	39.04	52.71
22:09:16	38.89	52.79
22:09:26	39.04	52.75
22:09:36	39.19	52.60
22:09:46	39.11	52.34
22:09:56	38.93	52.83
22:10:06	38.97	52.68
22:10:17	38.97	52.31
22:10:27	38.82	52.38
22:10:36	38.82	52.34
22:10:46	38.86	52.23
22:10:56	38.82	52.60
22:11:06	39.08	52.27
22:11:16	38.67	52.20
22:11:26	38.71	52.12
22:11:36	38.63	52.20
22:11:46	38.60	52.27
22:11:57	38.67	52.60
22:12:07	38.67	52.05
22:12:17	38.49	52.38
22:12:26	38.56	51.86
22:12:36	38.52	52.10
22:12:46	38.49	52.06
22:12:56	38.49	51.45
22:13:06	38.86	51.82
22:13:16	38.45	51.56
22:13:27	38.63	51.60
22:13:37	38.41	51.42

22:13:47	38.45	51.60
22:13:57	38.41	51.38
22:14:06	38.45	51.56
22:14:16	38.41	51.27
22:14:26	38.71	51.27
22:14:36	38.60	51.23
22:14:46	38.26	51.12
22:14:56	38.34	50.97
22:15:07	38.30	51.19
22:15:17	38.74	51.01
22:15:27	38.30	50.90
22:15:37	38.30	50.94
22:15:46	38.23	50.89
22:15:56	38.26	50.93
22:16:06	38.19	50.60
22:16:16	38.19	50.71
22:16:26	38.12	50.60
22:16:37	38.19	51.04
22:16:47	38.37	50.71
22:16:57	38.34	50.56
22:17:07	38.45	50.30
22:17:17	38.26	50.45
22:17:27	38.15	50.34
22:17:36	38.19	50.38
22:17:46	38.19	50.30
22:17:56	38.19	50.71
22:18:06	38.04	50.23
22:18:17	38.41	50.23
22:18:27	38.15	50.15
22:18:37	38.41	50.23
22:18:47	37.93	50.38
22:18:57	37.97	50.12
22:19:07	38.00	50.15
22:19:16	38.00	49.78
22:19:26	38.04	50.12

22:19:36	38.08	50.30
22:19:46	37.93	49.82
22:19:57	37.97	49.82
22:20:07	38.00	49.71
22:20:17	37.97	49.86
22:20:27	37.93	49.97
22:20:37	38.04	49.86
22:20:47	37.86	49.75
22:20:56	37.82	49.67
22:21:06	37.82	49.75
22:21:16	37.67	49.49
22:21:27	37.82	50.04
22:21:37	37.82	49.52
22:21:47	37.78	49.45
22:21:57	37.75	49.52
22:22:05	37.60	49.52
22:22:17	37.63	49.45
22:22:27	37.67	49.52
22:22:37	37.67	48.89
22:22:46	37.71	49.15
22:22:57	37.56	49.23
22:23:07	37.63	49.41
22:23:17	37.93	49.12
22:23:27	37.71	48.93
22:23:37	37.63	49.41
22:23:47	37.78	49.04
22:23:57	37.63	48.75
22:24:07	37.71	48.89
22:24:17	37.56	49.19
22:24:26	37.56	49.00
22:24:37	37.56	48.93
22:24:47	37.52	48.60
22:24:57	37.67	48.75
22:25:07	37.49	48.49
22:25:17	37.41	48.56

22:25:27	37.52	48.52
22:25:37	37.78	48.41
22:25:47	37.52	48.97
22:25:57	37.49	48.56
22:26:07	37.82	48.52
22:26:17	37.86	48.15
22:26:27	37.63	48.67
22:26:37	37.56	48.19
22:26:47	37.41	48.56
22:26:57	37.86	48.30
22:27:07	37.75	48.15
22:27:17	37.45	48.08
22:27:27	37.52	47.97
22:27:37	37.41	48.11
22:27:47	37.38	48.26
22:27:57	37.67	48.00
22:28:07	37.34	48.19
22:28:17	37.38	47.93
22:28:27	37.67	47.93
22:28:37	37.71	47.97
22:28:47	37.34	47.93
22:28:57	37.12	47.89
22:29:07	37.27	47.78
22:29:18	37.23	48.23
22:29:27	37.19	47.74
22:29:37	37.08	47.74
22:29:47	37.27	47.63
22:29:57	37.12	47.74
22:30:07	37.19	47.52
22:30:17	37.19	47.86
22:30:27	37.23	47.74
22:30:37	37.12	47.86
22:30:47	37.19	47.82
22:30:58	36.97	47.52
22:31:07	36.97	47.26

22:31:17	37.23	47.52
22:31:27	37.08	47.41
22:31:37	37.12	47.30
22:31:47	37.15	47.15
22:31:57	37.30	47.19
22:32:07	36.93	47.45
22:32:17	36.97	47.00
22:32:28	37.01	47.11
22:32:38	36.90	47.45
22:32:48	36.97	47.00
22:32:57	37.04	47.04
22:33:07	37.23	46.67
22:33:17	37.01	46.82
22:33:27	37.01	46.89
22:33:37	36.90	46.85
22:33:47	36.97	46.63
22:33:58	36.86	46.78
22:34:08	36.93	46.56
22:34:18	36.90	46.52
22:34:28	36.86	47.15
22:34:37	37.01	46.93
22:34:47	36.78	46.26
22:34:57	36.78	46.41
22:35:07	36.93	46.30
22:35:17	37.12	46.22
22:35:28	36.82	46.48
22:35:38	36.82	46.78
22:35:48	36.78	46.45
22:35:58	37.15	46.15
22:36:08	36.60	46.60
22:36:18	36.82	46.63
22:36:27	36.82	46.26
22:36:37	36.97	46.11
22:36:47	36.75	46.36
22:36:58	36.71	46.31

22:37:08	36.82	46.27
22:37:18	36.67	46.02
22:37:28	36.60	46.30
22:37:38	36.71	46.08
22:37:48	36.64	46.28
22:37:58	36.67	45.93
22:38:07	36.49	46.00
22:38:17	36.93	45.71
22:38:28	36.53	45.93
22:38:38	36.45	45.97
22:38:48	36.60	45.63
22:38:58	36.56	46.22
22:39:08	36.56	45.74
22:39:18	36.60	45.85
22:39:28	36.93	45.52
22:39:38	36.82	45.82
22:39:47	36.53	46.04
22:39:57	36.60	45.26
22:40:08	36.53	45.59
22:40:18	36.49	45.53
22:40:28	36.45	45.45
22:40:38	36.49	45.22
22:40:48	36.49	45.48
22:40:58	36.78	45.41
22:41:08	36.53	45.56
22:41:18	36.53	45.30
22:41:28	36.49	45.52
22:41:38	36.38	45.34
22:41:48	36.30	45.30
22:41:58	36.45	45.11
22:42:08	36.41	45.48
22:42:18	36.49	45.00
22:42:28	36.38	45.04
22:42:38	36.34	45.22
22:42:48	36.45	45.26

22:42:58	36.53	44.97
22:43:08	36.45	44.85
22:43:18	36.30	45.19
22:43:28	36.38	45.22
22:43:38	36.34	44.72
22:43:48	36.30	44.97
22:43:58	36.53	44.66
22:44:08	36.34	44.71
22:44:18	36.45	44.89
22:44:28	36.16	44.62
22:44:39	36.16	44.85
22:44:48	36.30	44.82
22:44:58	36.23	45.00
22:45:08	36.23	44.59
22:45:18	36.30	44.71
22:45:28	36.27	44.45
22:45:38	36.16	44.78
22:45:48	36.19	44.52
22:45:58	36.16	44.67
22:46:08	36.19	44.22
22:46:19	36.08	44.67
22:46:29	36.23	44.59
22:46:38	36.19	44.59
22:46:48	36.16	44.52
22:46:58	36.34	44.56
22:47:08	36.19	44.19
22:47:18	36.08	44.56
22:47:28	36.19	44.48
22:47:38	36.38	44.15
22:47:49	36.08	44.11
22:47:59	36.27	44.48
22:48:09	36.04	44.22
22:48:18	36.04	44.30
22:48:28	36.04	44.22
22:48:38	36.08	44.45

22:48:48	36.12	44.15
22:48:58	35.90	43.97
22:49:08	36.08	44.11
22:49:19	36.34	43.93
22:49:29	35.97	44.45
22:49:39	36.04	44.52
22:49:49	36.01	44.22
22:49:58	35.97	43.71
22:50:08	36.01	43.82
22:50:18	36.01	43.97
22:50:28	35.97	43.63
22:50:38	35.90	44.04
22:50:49	36.23	43.63
22:50:59	35.93	43.67
22:51:09	36.19	43.56
22:51:19	35.86	43.74
22:51:29	35.79	43.30
22:51:39	35.82	43.82
22:51:48	35.79	43.62
22:51:58	35.79	44.04
22:52:08	35.90	43.52
22:52:19	35.97	43.39
22:52:29	36.23	43.34
22:52:39	36.16	43.29
22:52:49	35.86	43.20
22:52:59	35.79	43.25
22:53:09	35.71	43.12
22:53:19	35.82	43.37
22:53:28	35.68	43.15
22:53:38	35.79	43.30
22:53:48	35.79	43.26
22:53:59	35.93	43.30
22:54:09	35.82	43.15
22:54:19	36.08	43.04
22:54:29	35.86	43.19

22:54:39	35.79	43.30
22:54:49	35.68	43.08
22:54:59	35.68	43.15
22:55:08	35.60	43.11
22:55:18	35.68	42.67
22:55:29	35.79	42.93
22:55:39	35.82	43.04
22:55:49	35.68	43.08
22:55:59	35.71	42.93
22:56:09	35.75	43.23
22:56:19	35.60	42.89
22:56:29	35.64	43.15
22:56:39	35.71	43.15
22:56:49	35.49	43.23
22:56:59	35.93	42.78
22:57:09	35.75	42.89
22:57:19	35.49	42.52
22:57:29	35.60	42.41
22:57:39	35.68	42.45
22:57:49	35.56	42.71
22:57:59	35.49	42.67
22:58:09	35.49	42.60
22:58:19	35.82	42.85
22:58:29	35.45	43.00
22:58:39	35.56	42.82
22:58:49	35.49	42.60
22:58:59	35.49	42.52
22:59:09	35.38	42.34
22:59:19	35.49	42.52
22:59:29	35.38	42.63
22:59:39	35.82	42.34
22:59:49	35.45	42.52
23:00:00	35.38	42.45
23:00:09	35.64	42.89
23:00:19	35.45	42.71

23:00:29	35.38	42.23
23:00:39	35.34	42.41
23:00:49	35.38	42.23
23:00:59	35.60	41.97
23:01:09	35.53	42.04
23:01:19	35.34	41.93
23:01:30	35.34	42.08
23:01:40	35.31	42.00
23:01:50	35.34	42.52
23:01:59	35.31	42.04
23:02:09	35.38	42.52
23:02:19	35.49	42.19
23:02:29	35.31	41.93
23:02:39	35.27	41.97
23:02:49	35.27	42.30
23:02:59	35.34	41.89
23:03:10	35.19	42.15
23:03:20	35.23	41.86
23:03:30	35.27	42.00
23:03:39	35.16	42.04
23:03:49	35.16	41.78
23:03:59	35.27	41.78
23:04:09	35.31	41.41
23:04:19	35.16	42.00
23:04:29	35.27	41.49
23:04:40	35.16	41.63
23:04:50	35.27	41.89
23:05:00	35.27	41.26
23:05:10	35.27	41.56
23:05:19	35.16	42.08
23:05:29	35.05	41.52
23:05:39	35.23	41.47
23:05:49	35.05	41.43
23:05:59	35.12	41.41
23:06:10	35.12	41.86

23:06:20	35.12	41.30
23:06:30	35.19	41.56
23:06:40	35.16	41.67
23:06:50	35.27	41.41
23:07:00	35.34	41.45
23:07:09	35.38	41.08
23:07:19	35.12	41.15
23:07:29	35.16	41.34
23:07:40	35.16	41.30
23:07:50	35.08	41.23
23:08:00	35.08	41.04
23:08:10	35.08	41.12
23:08:20	35.01	41.00
23:08:30	34.97	41.60
23:08:40	34.97	41.12
23:08:49	34.94	41.00
23:08:59	35.01	40.82
23:09:10	35.31	40.86
23:09:20	35.27	41.04
23:09:30	35.01	41.23
23:09:40	34.94	40.93
23:09:50	34.97	40.86
23:10:00	35.01	41.12
23:10:10	34.94	41.23
23:10:20	35.34	41.12
23:10:30	34.94	40.52
23:10:39	34.90	40.82
23:10:50	34.86	40.56
23:11:00	34.97	40.67
23:11:10	34.90	40.78
23:11:20	34.94	41.12
23:11:30	34.86	40.71
23:11:40	34.86	40.56
23:11:50	34.83	40.86
23:12:00	34.90	41.08

23:12:10	34.86	40.63
23:12:20	34.86	40.78
23:12:30	34.94	40.52
23:12:40	34.86	40.38
23:12:50	34.86	40.59
23:13:00	34.75	40.53
23:13:10	34.86	40.82
23:13:20	34.75	40.86
23:13:30	34.86	40.49
23:13:40	34.83	40.52
23:13:50	34.90	40.56
23:14:00	35.16	40.63
23:14:10	35.05	40.27
23:14:20	35.08	40.45
23:14:30	34.75	40.49
23:14:40	34.71	40.44
23:14:50	34.79	40.40
23:15:00	34.86	40.42
23:15:10	35.05	40.30
23:15:21	34.68	40.36
23:15:31	34.71	40.30
23:15:40	34.71	40.38
23:15:50	34.86	39.97
23:16:00	35.12	40.03
23:16:10	34.79	40.27
23:16:20	34.97	40.30
23:16:30	34.75	40.27
23:16:40	34.79	40.30
23:16:50	34.68	39.97
23:17:01	34.75	40.23
23:17:11	34.64	40.12
23:17:20	34.68	39.86
23:17:30	34.57	40.23
23:17:40	34.64	39.75
23:17:50	34.64	39.90

23:18:00	34.86	39.93
23:18:10	34.68	39.90
23:18:20	34.71	39.86
23:18:30	34.60	39.78
23:18:41	34.60	40.19
23:18:51	34.75	40.01
23:19:00	34.57	40.08
23:19:10	34.71	39.90
23:19:20	34.57	39.75
23:19:30	34.86	39.75
23:19:40	34.64	40.15
23:19:50	34.68	40.23
23:20:00	34.71	39.92
23:20:11	34.46	39.90
23:20:21	34.71	39.86
23:20:31	34.71	39.71
23:20:41	34.60	39.67
23:20:50	34.57	39.78
23:21:00	34.49	39.67
23:21:10	34.83	39.82
23:21:20	34.46	39.86
23:21:30	34.42	40.19
23:21:40	34.46	39.71
23:21:51	34.46	39.67
23:22:01	34.42	39.60
23:22:11	34.38	40.01
23:22:21	34.64	39.53
23:22:30	34.57	39.97
23:22:40	34.34	39.71
23:22:50	34.79	39.68
23:23:00	34.64	39.53
23:23:10	34.38	39.90
23:23:21	34.31	39.38
23:23:31	34.42	39.86
23:23:41	34.71	39.30

23:23:51	34.42	39.82
23:24:01	34.31	39.56
23:24:10	34.34	39.60
23:24:20	34.27	39.38
23:24:30	34.31	39.60
23:24:40	34.31	39.64
23:24:50	34.27	39.71
23:25:01	34.27	39.27
23:25:11	34.34	39.38
23:25:21	34.34	39.82
23:25:31	34.42	39.16
23:25:41	34.79	39.34
23:25:51	34.34	39.27
23:26:00	34.27	39.53
23:26:10	34.53	39.38
23:26:20	34.34	39.01
23:26:31	34.53	39.38
23:26:41	34.27	39.45
23:26:51	34.23	39.23
23:27:01	34.27	39.27
23:27:11	34.23	39.49
23:27:21	34.20	39.16
23:27:31	34.23	39.12
23:27:40	34.23	39.04
23:27:50	34.23	39.08
23:28:01	34.16	39.20
23:28:11	34.16	39.18
23:28:21	34.23	38.90
23:28:31	34.20	39.04
23:28:41	34.23	39.30
23:28:51	34.23	38.79
23:29:01	34.20	38.86
23:29:11	34.12	39.19
23:29:20	34.09	38.90
23:29:31	34.53	38.75

23:29:41	34.09	39.01
23:29:51	34.12	38.71
23:30:01	34.12	38.68
23:30:11	34.12	39.01
23:30:21	34.09	38.82
23:30:31	34.09	38.97
23:30:41	34.09	39.04
23:30:51	33.98	38.90
23:31:01	34.12	38.56
23:31:11	34.16	38.45
23:31:21	34.12	38.45
23:31:31	34.09	38.45
23:31:41	34.31	38.49
23:31:51	34.05	38.82
23:32:01	34.05	38.49
23:32:11	34.05	38.90
23:32:21	33.90	38.64
23:32:31	34.01	38.42
23:32:41	33.90	38.82
23:32:51	33.98	38.71
23:33:01	34.01	38.49
23:33:11	34.09	38.08
23:33:21	34.01	38.68
23:33:31	34.20	38.42
23:33:41	33.90	38.23
23:33:51	34.34	38.12
23:34:01	33.98	38.22
23:34:12	34.01	38.28
23:34:21	33.98	38.27
23:34:31	34.01	38.31
23:34:41	34.09	38.19
23:34:51	33.94	38.42
23:35:01	33.94	38.42
23:35:11	34.05	38.19
23:35:21	33.90	38.08

23:35:31	33.94	38.08
23:35:41	33.98	38.56
23:35:52	33.98	38.08
23:36:02	33.94	37.90
23:36:11	33.90	38.01
23:36:21	33.90	37.86
23:36:31	33.86	37.94
23:36:41	33.83	37.79
23:36:51	33.90	38.12
23:37:01	33.79	38.08
23:37:12	33.94	37.94
23:37:22	34.01	37.86
23:37:32	34.01	37.83
23:37:42	33.90	38.16
23:37:51	34.16	38.01
23:38:01	33.86	37.94
23:38:11	33.83	37.94
23:38:21	33.86	38.12
23:38:31	34.05	37.97
23:38:41	34.12	37.90
23:38:52	33.79	37.86
23:39:02	34.16	37.79
23:39:12	33.75	37.79
23:39:22	33.72	37.68
23:39:31	33.75	37.64
23:39:41	33.61	37.90
23:39:51	33.75	37.75
23:40:01	33.75	37.68
23:40:11	33.75	37.60
23:40:22	33.72	37.57
23:40:32	33.94	37.64
23:40:42	33.94	37.68
23:40:52	33.98	37.64
23:41:02	33.83	37.68
23:41:12	33.79	37.57

23:41:21	33.72	37.49
23:41:31	33.24	37.68
23:41:41	32.61	37.57
23:41:51	31.76	37.75
23:42:02	31.24	37.49
23:42:12	31.06	37.42
23:42:22	30.98	37.53
23:42:32	30.65	37.27
23:42:42	30.69	35.65
23:42:52	30.76	35.05
23:43:01	30.50	34.46
23:43:11	30.32	34.28
23:43:21	30.25	33.80
23:43:32	30.14	33.28
23:43:42	30.32	32.99
23:43:52	29.80	32.58
23:44:02	29.80	32.21
23:44:12	29.69	32.03
23:44:22	29.54	31.73
23:44:32	29.62	31.40
23:44:42	29.99	31.25
23:44:51	29.58	31.33
23:45:02	29.43	30.85
23:45:12	29.43	31.18
23:45:22	29.47	30.66
23:45:32	29.36	30.63
23:45:42	29.32	30.52
23:45:52	29.40	30.33
23:46:02	29.32	30.18
23:46:12	29.32	30.00
23:46:22	29.21	29.96
23:46:32	29.18	29.96
23:46:42	29.40	29.89
23:46:52	29.47	30.15
23:47:02	29.14	29.74

23:47:12	29.10	29.67
23:47:22	29.07	29.72
23:47:32	28.92	29.52
23:47:42	29.03	29.41
23:47:52	28.95	29.39
23:48:02	28.95	29.74
23:48:12	28.95	29.48
23:48:22	28.92	29.37
23:48:32	28.81	29.48
23:48:42	28.92	29.63
23:48:52	29.03	29.59
23:49:02	29.21	29.56
23:49:12	29.29	29.63
23:49:22	28.84	30.00
23:49:33	28.73	29.96
23:49:42	28.92	29.82
23:49:52	28.84	29.89

ΠΑΡΑΡΤΗΜΑ Γ: Το πρόγραμμα λειτουργίας του μικροελεγκτή.

```
# ---------------------------------------------
# Title   : Distillation Chamber Controller
# Author  : Dimitrios Ouzounis
# Date    : 2025-09
# Version : 1.0
# ---------------------------------------------

import socketio #import socketio methods
import time #import time methods
import json #import json methods
import os
import digitalio #import gpio methods
import board #import SPI
from RPLCD.i2c import CharLCD
import adafruit_max31865 #import MAX31865 methods
import threading
import schedule
import gpiozero
from  gpiozero import LED, Button, CPUTemperature, RotaryEncoder

HOST = '192.168.1.100' #set hostname
PORT = 3000 #set port number
DEVICE_NAME = 'DEV001' #set device name
CONNECTION_QUERY = f'http://{str(HOST)}:{str(PORT)}' #connection string

# Starting conditions
global newTemp, emgFlag, maxLevel, levelStr, startStatus, stopStatus, menuFlag, menuOptions,
bounceTime
bounceTime = 0.04
levelStr = 'OK'
newTemp = 10
menuOptions = 0
emgFlag = False
menuFlag = False
startStatus = False
stopStatus = True
mysocket = socketio.Client()
lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1,
            cols=20, rows=4, dotsize=10, auto_linebreaks=False,
            backlight_enabled=True)
```

```
lcd.cursor_mode = 'hide'
spi = board.SPI()

# set led output pin 1 On/Off indicator
led1 = LED(5)
# set led output pin 2
led2 = LED(25)
# set led output pin 3 Level sensor indicator
led3 = LED(24)
# set led output pin 4
led4 = LED(23)
# set led output pin 5 Emergency/Error Indicator
led5 = LED(18)

# relay output 1
output1 = gpiozero.DigitalOutputDevice(13, initial_value=False)
# relay output 2
output2 = gpiozero.DigitalOutputDevice(19, initial_value=False)
# ------------------------------------------------------
# START BUTTON
def startBtnPressed():
    global startStatus, dataFileName, stopStatus
    if stopButton.is_pressed is True:
        return 0
    else:
        if emgFlag is False:
            if startStatus is False:
                startStatus = True
                stopStatus = False
                mysocket.emit('startBtn')
                checkTemp()
                led1.on()
                output1.on() # relay output 1 ON
                output2.on() # relay output 2 ON
                # All data are saved to a json file locally. The name of the file is genareted from date and
time
                # print(time.strftime('%Y%j%H%M%S'))
                theData = {
                    'action': 'Start Button',
                    'EMG': emgFlag,
                    'timeStamp': now,
                    'sensorA': [tempA, resA],
```

```
            'sensorB': [tempB, resB],
            'status': startStatus,
            'cpuTemp': cpu,
            'userTemp': newTemp,
            'tankLevel': maxLevel,
            'out1': output1.value,
            'out2': output2.value
        }
        theData = json.dumps(theData)
        dataFileName = f'{time.strftime("%Y%j%H%M%S")}'
        print(startStatus)
        print(f'Data to be saved ----> {theData}')
        f = open(f'{dataFileName}.json', 'w')
        f.write('{\n\t"Op_details" : [\n\t\t' + theData + ',' + '\n')
        f.close

    print(f'_____Start Button pressed! {led1.is_active}')

startButton = gpiozero.Button(27, bounce_time = bounceTime)
startButton.when_pressed = startBtnPressed

# ---------------------------------------------------------
# STOP BUTTON
def stopBtnPressed():
    global startStatus, stopStatus, emgFlag
    if stopStatus is False:
        startStatus = False
        stopStatus = True
        led1.off()
        output1.off() # relay output 1 OFF
        output2.off() # relay output 2 OFF
        # TO-DO add other activities to stop
        theData = {
            'action': 'Stop Button',
            'EMG': emgFlag,
            'timeStamp': now,
            'sensorA': [tempA, resA],
            'sensorB': [tempB, resB],
            'status': startStatus,
            'cpuTemp': cpu,
            'userTemp': newTemp,
            'tankLevel': maxLevel,
```

```
            'out1': output1.value,
            'out2': output2.value
        }
        theData = json.dumps(theData)
        print(startStatus)
        print(f'Data to be saved ----> {theData}')

        f = open(f'{dataFileName}.json', 'a')
        f.write('\t\t' + theData + '\n\t]\n}')
        f.close
    print(f'_____________________Stop Button pressed! {led1.is_active}')

# Reset function
def resetDevice():
    global newTemp, emgFlag, menuFlag
    if emgFlag is True: # check if emergency button is pressed, to perform error reset
        lcd.clear()
        newTemp = 10
        emgFlag = False
        menuFlag = False
        led2.blink(on_time=0.5, off_time=0.5, n = 4)

stopButton = gpiozero.Button(16, bounce_time = bounceTime)
stopButton.when_pressed = stopBtnPressed
stopButton.hold_time = 2 # Hold stop button for 2 seconds to perform error reset
stopButton.when_held = resetDevice
# ---------------------------------------------------------
# rotary encoder RIGHT
def rotaryRight():
    global newTemp, menuOptions
    if menuFlag is False:
        newTemp = newTemp + 1
        print(f'_____________________Rotary Right! New Temp : {newTemp}')
    elif menuFlag is True:
        if menuOptions >= 2:
            menuOptions = 2
        else:
            menuOptions = menuOptions + 1
        print(f'_____________________Rotary Left! New Option : {menuOptions}')


# rotary encoder LEFT
```

```
def rotaryLeft():
    global newTemp, menuOptions
    if menuFlag is False:
        if newTemp <= 0:
            newTemp = 0
        else:
            newTemp = newTemp - 1
        print(f'_____Rotary Left! New Temp : {newTemp}')
    elif menuFlag is True:
        if menuOptions <= 0:
            menuOptions = 0
        else:
            menuOptions = menuOptions - 1
        print(f'_____Rotary Left! New Option : {menuOptions}')



rotary = RotaryEncoder(20, 21, bounce_time = bounceTime, threshold_steps=(0,4))
rotary.when_rotated_clockwise = rotaryRight
rotary.when_rotated_counter_clockwise = rotaryLeft

def sendUserTemp():
    print(f'_____** -- SEND USER TEMP --
**_____')

# ---------------------------------------------------------
# rotary encoder push button
def rotaryPressed():
    global menuFlag, menuOptions
    if menuFlag is True:
        if menuOptions == 0:
            if connection == '0':
                connectThread()
            menuFlag = False
            lcd.clear()
            print(f'_____Rotary pressed! Selected :{menuOptions} -
Connect')
        if menuOptions == 1:
            # TO-DO
            menuFlag = False
            lcd.clear()
```

```
            print(f'_____Rotary pressed! Selected :{menuOptions} -
Shutdown')
            print(f'_____SHUTDOWN INACTIVE!')
        if menuOptions == 2:
            menuFlag = False
            lcd.clear()
            print(f'_____Rotary pressed! Selected :{menuOptions} - Exit')
    else:
        sendUserTemp()

rotarySW = gpiozero.Button(26, bounce_time = bounceTime)
rotarySW.when_pressed = rotaryPressed

# ------------------------------------------------------
# EMERGENCY button
def emgPressed():
    global emgFlag, menuFlag
    emgFlag = True
    menuFlag = False
    led2.blink(on_time=1, off_time=1)
    stopBtnPressed()
    print(f'_____Emergency Button pressed!')

emgButton = gpiozero.Button(12, bounce_time = bounceTime)
# EMERGENCY button condition NC
emgButton.when_released = emgPressed

# ------------------------------------------------------
# TO-DO: Change to MENU button
# Menu button
def menuPressed():
    global menuFlag
    menuFlag = True
    lcd.clear()

menuBtn = gpiozero.Button(6, bounce_time = bounceTime)
menuBtn.when_pressed = menuPressed

# ------------------------------------------------------
# Level sensor
def levelMax():
    global maxLevel, levelStr
```

```
    if levelSensor.is_active is True:
        led4.blink(on_time=0.5, off_time=0.5)
        maxLevel = True
        levelStr = 'MAX'
    elif levelSensor.is_active is False:
        led4.off()
        maxLevel = False
        levelStr = 'OK'

levelSensor = gpiozero.Button(22)
levelSensor.when_pressed = levelMax
levelSensor.when_released = levelMax

# ------------------------------------------------------
# set SPI CS pin for channel A
csA = digitalio.DigitalInOut(board.D7)
csA.direction = digitalio.Direction.INPUT
# set SPI CS pin for channel B
csB = digitalio.DigitalInOut(board.D1)
csB.direction = digitalio.Direction.INPUT

# Sensor A initialization
sensorA = adafruit_max31865.MAX31865(spi,
                                     csA,
                                     rtd_nominal=100,
                                     ref_resistor=468.4,
                                     wires=3)
# Sensor B initialization
sensorB = adafruit_max31865.MAX31865(spi,
                                     csB,
                                     rtd_nominal=100,
                                     ref_resistor=468.0,
                                     wires=3)


# connect to server function
def connect():
    global connection
    try:
        print(f'Connecting to server... {str(HOST)}:{str(PORT)}')
        connection = mysocket.connect(CONNECTION_QUERY)
        print(CONNECTION_QUERY)
```

```
    except:
        print(f'Connection Error.')
        connection = '0'
        return connection
    else:
        print(f'Connected to server {str(HOST)}:{str(PORT)}')
        return connection

# send data to server
def sendMeasures(theData):
    if connection != 0:
        try:
            mysocket.emit('measures', theData)
            print(f'Sending ----> {theData}')

        except:
            time_stamp = time.strftime('%H:%M:%S')
            print(f'Error! {time_stamp}')
            print(f'Error sending ----> {theData}')
            print('\n')

        else:
            time_stamp = time.strftime('%H:%M:%S')
            print(f'Success! {time_stamp}')
            print('\n')
    else:
        time_stamp = time.strftime('%H:%M:%S')
        print('\n')
        print (f'Connection Error! Not connected to server!\n')
        print(f'Error! {time_stamp}')
        print(f'Error sending ----> {theData}')
        printData()


@mysocket.on('connected')
def handle_json(data):
    print(f'Server response : {data}')

@mysocket.on('start')
def startDistil():
    startBtnPressed()
```

```
@mysocket.on('stop')
def stopDistil():
    stopBtnPressed()

# Add user temperature change value from slider
@mysocket.on('sliderChange')
def setUserTemp(data):
    global newTemp
    print(f'---------- Slider Change Function!!!! {data}')
    newTemp = int(data)

# Add Emergency Button web-socket endpoint
# Add Reset Button web-socket endpoint

def measure():
    global tempA, tempB, resA, resB, now, theData, cpu, s, startStatus
    led3.on()
    #led4.on()
    now = time.strftime('%H:%M:%S')
    cpu = CPUTemperature().temperature
    tempA = "{:0.2f}".format(sensorA.temperature)
    resA = "{:0.3f}".format(sensorA.resistance)
    tempB = "{:0.2f}".format(sensorB.temperature)
    resB ="{:0.3f}".format(sensorB.resistance)
    theData = {
            'timeStamp': now,
            'sensorA': [tempA, resA],
            'sensorB': [tempB, resB],
            'status': startStatus,
            'cpuTemp': cpu,
            'userTemp': newTemp,
            'tankLevel': maxLevel,
            'out1': output1.value,
            'out2': output2.value,
            'emgStatus': emgFlag
            }
    theData = json.dumps(theData)
    if startStatus is True:
        s = f' < ON  >'
    else:
        s = f' < OFF  >'
    printData()
```

```
    printDataLCD()
    printTimeLCDThread()
    sendData()

def sendData():
    sendMeasures(theData)
    led3.off()
    #led4.off()

def printDataLCD():
    if menuFlag is False:
        lcd.cursor_pos = (0, 11)
        lcd.write_string(f'{s}')
        lcd.cursor_pos = (1, 0)
        lcd.write_string(f'A : {tempA} C {resA}')
        lcd.cursor_pos = (2, 0)
        lcd.write_string(f'B : {tempB} C {resB}')
        lcd.cursor_pos = (3, 0)
        try:
            lcd.write_string(f'User Temp : {newTemp} ')
        except:
            lcd.write_string(f'User Temp : {newTemp}')
    elif menuFlag is True:
        lcd.cursor_pos = (1, 0)
        lcd.write_string(f'MENU')
        if menuOptions == 0:
            lcd.cursor_pos = (2, 0)
            lcd.write_string(f'Connect ')
        elif menuOptions == 1:
            lcd.cursor_pos = (2, 0)
            lcd.write_string(f'Shutdown')
        elif menuOptions == 2:
            lcd.cursor_pos = (2, 0)
            lcd.write_string(f'Exit    ')

def printTimeLCD():
    lcd.cursor_pos = (0, 0)
    lcd.write_string(f'{time.strftime("%H:%M:%S")}')

# Print data to terminal
def printData():
    print('-------------------------')
```

```
    print(f'Last measure  : {now}')
    print(f'CPU           : {cpu} °C')
    print(f'Session ID    : {mysocket.sid}')
    print(f'Sensor A      : {tempA} °C  {resA} Ω')
    print(f'Sensor B      : {tempB} °C  {resB} Ω')
    print(f'Status        : {startStatus}')
    print(f'Tank Level    : {levelStr}')
    print(f'EMG STOP      : {emgFlag}')
    print(f'Output 1      : {output1.value}')
    print(f'Output 1      : {output2.value}')
    try:
        print(f'Threshhold    : {newTemp} °C')
    except:
        print(f'Threshhold    : Reading... °C')
    print('\n')

# Check temperature
def checkTemp():
    # if (float(tempA) + 0.5 ) <= newTemp and led1.is_active == True:
    if (float(tempA) + 0.5 ) >= newTemp and startStatus == True:
        output1.off() # Relay 1
        # output2.off() # Optional Relay 2
    # elif (float(tempA) - 0.5 ) >= newTemp and led1.is_active == True:
    elif (float(tempA) - 0.5 ) <= newTemp and startStatus == True:
        output1.on() # Relay 1
        # output2.on() # Optional Relay 2

# Set a thread to read the sensors
def readSensorsThread():
    readSensors = threading.Thread(target=measure)
    readSensors.start()

# Set a thread to display time to LCD
def printTimeLCDThread():
    displayTime = threading.Thread(target=printTimeLCD)
    displayTime.start()

# Set a thread to manually connect to server
def connectThread():
    connectThr = threading.Thread(target=connect)
    connectThr.start()
```

```
lcd.cursor_pos = (1,0)
lcd.write_string(f'Loading...')
lcd.cursor_pos = (2,0)
lcd.write_string(f'Connecting...')
connect()
lcd.clear()
levelMax()
measure()

# Schedule the thread
schedule.every(0.5).seconds.do(readSensorsThread)

try:
    while True:
        checkTemp()
        schedule.run_pending()
        time.sleep(0.9)
except KeyboardInterrupt:
    print('Exiting operation...')
```

ΠΑΡΑΡΤΗΜΑ Δ: Το πρόγραμμα λειτουργίας του διακομιστή και της εφαρμογής χρήστη.

index.js.

```
// ---------------------------------------
// Title   : Distillation Server Controller
// Author  : Dimitrios Ouzounis
// Date    : 2025-09
// Version : 1.0
// ---------------------------------------

const express = require("express");
const session = require('express-session');
const mysql = require('mysql2/promise');
const chalk = require("chalk");
const server = require("http").createServer(express);
const bodyParser = require("body-parser");
const io = require("socket.io")(server);
const ejs = require("ejs");
// API instance
const app = express();
// Web Socket server port
const wsPort = 3000;
// Express server port
const httpPort = 3001;
// Express router
const buttonRouter = require("./routers/buttons")(io, opStatus, tableName, mysql, sqlCon(),
newTable());

// Connect to database
async function sqlCon() {
    const connection = await mysql.createConnection({
        host: "localhost", // MySQL host
        user: "root", // MySQL username
        password: "rootadmin", // MySQL password
        database: "thesis" // The name of the database
    });
    console.log(`----> Connected to the database!`);
    return connection;
};
// Create new database table
async function newTable() {
```

```
    const d = new Date();
    newTableName = `Y${d.getFullYear()}M${d.getMonth()}D${d.getDate()}H${d.getHours()}M${d.getMinutes()}S${d.getSeconds()}`;
    const createTableQuery = `
        CREATE TABLE ${newTableName.split(" ").join("")} (
        id INT AUTO_INCREMENT PRIMARY KEY,
        timeStamp VARCHAR(12),
        channelA VARCHAR(10),
        channelB VARCHAR(10));`;
    const connection = await sqlCon();
    await connection.execute(createTableQuery);
    // console.log(`New table with name ${tableName} created`);
    return newTableName;
};
//Express static views
app.use(express.static("views"));
// Middleware to access req.body
app.use(bodyParser.urlencoded({ extended:true}));
// Middleware to access buttons
app.use(buttonRouter);

// Sample user
const users = {username: "admin", password: "124paok"};

// User session
app.use(session({
    secret: "124paok",
    resave: false, // Force save session
    saveUninitialized: true, // Save uninitialized session
    cookie: { secure: false }
}));

// Check if there is a logged user
function isLoggedIn(req, res, next) {
    if (req.session.user) return next();
    res.redirect("/");
}

// Starting conditions
const errorMsg = "Reading...";
const errorLogin = "Invalid Username or Password";
// 10000 = 10sec
```

```
const myInterval = 10000;
//10000 = 10sec 60000 = 1min, 300000 = 5min, 600000 = 10min
const dataBaseInterval = 10000;
var opStatus = 0;
var cpuTemp = 0;
var out1 = 0;
var out2 = 0;
var userTemp = 10;
var tempA = 0;
var tempB = 0;
var resA = 100;
var resB = 100;
var tankLevel = 0;
var emg = 0;
var sumTempA = 0;
var sumTempB = 0;
var countA = 0;
var countB = 0;
var avgTempA = 0;
var avgTempB = 0;
var maxA = 0;
var minA = 0;
var maxB = 0;
var minB = 0;
var lastTenChanA = new Array(7);
var lastTenChanB = new Array(7);
var timeStamp = "";
var newTableName;
var tables;
var tablesData;

// Action when the client is connected
io.on("connection", (socket) => {
    console.log(chalk.inverse.green("---- User connected ----"));
    console.log(`Connection Time  : ${socket.handshake.time}`);
    console.log(`Socket ID       : ${socket.id}`);
    console.log(`Client IP       : ${socket.handshake.address.slice(7)}`);

// Response to client when connected
    io.emit("connected", "Connection OK");
    // io.emit("stop");
```

```
// Action when the client is disconnected
    socket.on("disconnect", () => {
        console.log(chalk.inverse.red("---- User disconnected ----"));
        console.log(`Client IP       : ${socket.handshake.address.slice(7)}`);
    });

// Action when the client sends measured data
    socket.on("measures", (data) => {
        try {
            measureData = JSON.parse(data);
            timeStamp = measureData.timeStamp;
            tempA = measureData.sensorA[0];
            resA = measureData.sensorA[1];
            tempB = measureData.sensorB[0];
            resB = measureData.sensorB[1];
            opStatus = measureData.status;
            cpuTemp = measureData.cpuTemp;
            userTemp = measureData.userTemp;
            tankLevel = measureData.tankLevel;
            out1=measureData.out1;
            out2=measureData.out2;
            emg=measureData.emgStatus;
            console.log(`Timestamp          : ${timeStamp}`);
            console.log(`CPU Temperature     : ${cpuTemp} °C`);
            console.log(chalk.green(`Channel A  : ${tempA} °C  ${chalk.inverse(resA)} Ω`));
            maxTempA();
            minTempA();
            averageTempA();
            console.log(`Last Min A ${lastTenChanA.length} : ${lastTenChanA}`);
            console.log(`------------------------------`)
            console.log(chalk.yellow(`Channel B  : ${tempB} °C  ${chalk.inverse(resB)} Ω`));
            maxTempB();
            minTempB();
            averageTempB();
            console.log(`Last Min B ${lastTenChanB.length} : ${lastTenChanB}`);
            opStatus === true || opStatus === 1?
                console.log(`Status : ${chalk.inverse.green(opStatus).padStart(39)}`):
                console.log(`Status : ${chalk.inverse.red(opStatus).padStart(39)}`);
            tankLevel === true || tankLevel === 1?
                console.log(`Tank Level : ${chalk.inverse.red("MAX").padStart(35)}`):
                console.log(`Tank Level : ${chalk.inverse.green("OK").padStart(35)}`);
            console.log(`Threshold  : ${userTemp.toString().padStart(16)} °C`);
```

```
            console.log(`Output 1           : ${out1}`);
            console.log(`Output 2           : ${out2}`);
            console.log(`EMG Status          : ${emg}`);
            console.log(`------------------------------------------------------------`);
            console.log(`------------------------------------------------------------`);
        }
        catch (e) {
            console.log(`${errorMsg} values!`);
        }
    });

    //Start button
    socket.on("startBtn", () => {
        console.log(chalk.green("----> Start Button pressed!"));
        newTable();
        console.log(`New table with name ${tableName} created`);
    });
});

app.get("/", (req, res) => {
    res.render("login.ejs");
});

app.get("/main", isLoggedIn, (req, res) => {
    try {
        console.log(JSON.stringify(req.body));
        console.log(`Status: ${opStatus}`);
        res.render("index.ejs", { user: req.session.user, userTemp: userTemp, tempA: tempA, resA:
resA, maxA: maxA, minA: minA, avgTempA: avgTempA, tempB: tempB, resB: resB, maxB:
maxB, minB: minB, avgTempB: avgTempB, sts: opStatus, cpuTemp: cpuTemp, out1: out1, out2:
out2, tankLevel: tankLevel, emg: emg, lastTenChanA: lastTenChanA, lastTenChanB: lastTenChanB
});
    }
    catch (e) {
        console.log(`-------------- > ${e}`)
        console.log(chalk.inverse.orange("Reading..."));
        res.render("index.ejs", { user: req.session.user, userTemp: errorMsg, tempA: errorMsg, resA:
errorMsg, maxA: errorMsg, minA: errorMsg, avgTempA: errorMsg, tempB: errorMsg, resB:
errorMsg, maxB: errorMsg, minB: errorMsg, avgTempB: errorMsg, sts: opStatus, cpuTemp:
errorMsg, out1: out1, out2: out2, tankLevel: tankLevel, emg: emg });
    }
});
```

```
app.post("/login", (req, res) => {
   console.log(" ----> Submit Button");
   const { username, password } = req.body;
   if (username === users.username && password === users.password) {
      req.session.user = username;
      res.redirect("/main");
   } else {
      res.render("errorLogin.ejs");
  }
});

app.get("/exitBtn", (req, res) => {
   console.log(" ----> Exit Button");
   req.session.destroy((err) => {
      if (err) {
         console.log(err);
         return res.redirect("/");
      }
      console.log(" ----> Exit Button");
      res.clearCookie("connect.sid");
      res.redirect("/");
    });
});

app.get("/updateTemp", (req, res) => {
   try {
      console.log(chalk.inverse.cyan("Update data request from Web client OK"));
      res.send({ userTemp: userTemp, tempA: tempA, resA: resA, maxA: maxA, minA: minA,
avgTempA: avgTempA, tempB: tempB, resB: resB, maxB: maxB, minB: minB, avgTempB:
avgTempB,sts: opStatus, cpuTemp: cpuTemp, out1: out1, out2: out2, tankLevel: tankLevel, emg:
emg, lastTenChanA: lastTenChanA, lastTenChanB: lastTenChanB });
   }
   catch (e) {
      console.log(chalk.inverse.red("Update data request from Web client failed!"));
      res.send({ userTemp: errorMsg, tempA: errorMsg, resA: errorMsg, maxA: errorMsg, minA:
errorMsg, avgTempA: errorMsg, tempB: errorMsg, resB: errorMsg, maxB: errorMsg, minB:
errorMsg, avgTempB: errorMsg, sts: opStatus, cpuTemp: errorMsg, out1: out1, out2: out2,
tankLevel: tankLevel, emg: emg });
   }
});
```

```
app.get("/userTemp", (req, res) => {
    console.log(chalk.inverse.yellow(`User temperature : `));
});

app.get("/getTableNames", isLoggedIn, async (req, res) => {
    const connection = await sqlCon();
    const [rows] = await connection.execute("SHOW TABLES");
    tables = rows.map(row => Object.values(row)[0]);
    console.log(`Tables in database: ${tables}`);
    await connection.end();
    // res.redirect("/history");
    res.render("history.ejs", {tables: tables, user: req.session.user, userTemp: userTemp, tempA:
tempA, resA: resA, maxA: maxA, minA: minA, avgTempA: avgTempA, tempB: tempB, resB:
resB, maxB: maxB, minB: minB, avgTempB: avgTempB, sts: opStatus, cpuTemp: cpuTemp, out1:
out1, out2: out2, tankLevel: tankLevel, emg: emg, lastTenChanA: lastTenChanA, lastTenChanB:
lastTenChanB });
});
var tableName;
// Get table data
app.get("/getTableData", isLoggedIn, async (req, res) => {
    tableName = req.query.table;
    const connection = await sqlCon();
    console.log(`Table name : ${tableName}`);
    const [rows] = await connection.execute(`SELECT * FROM ${tableName}`);
    tablesData = rows;
    console.log(tablesData);
    await connection.end();
    res.render("history2.ejs", {tableName:tableName, tablesData: tablesData, tables: tables, user:
req.session.user, userTemp: userTemp, tempA: tempA, resA: resA, maxA: maxA, minA: minA,
avgTempA: avgTempA, tempB: tempB, resB: resB, maxB: maxB, minB: minB, avgTempB:
avgTempB, sts: opStatus, cpuTemp: cpuTemp, out1: out1, out2: out2, tankLevel: tankLevel, emg:
emg, lastTenChanA: lastTenChanA, lastTenChanB: lastTenChanB });
});

app.get("/showTableData", (req, res) => {
    console.log(`Table name : ${tableName}`);
    console.log(typeof(tablesData));
    console.log(tablesData);
    res.render("history2.ejs", { tableName: tableName, tablesData: tablesData, tables: tables, user:
req.session.user, userTemp: userTemp, tempA: tempA, resA: resA, maxA: maxA, minA: minA,
avgTempA: avgTempA, tempB: tempB, resB: resB, maxB: maxB, minB: minB, avgTempB:
```

```
avgTempB, sts: opStatus, cpuTemp: cpuTemp, out1: out1, out2: out2, tankLevel: tankLevel, emg:
emg, lastTenChanA: lastTenChanA, lastTenChanB: lastTenChanB });
});

// Send slider value to IOT hardware
app.post("/sliderChange", (req, res, data) => {
   let userTemp = req.body.userTemp;
   io.emit("sliderChange", userTemp);
   res.send({ userTemp: userTemp });
   console.log(chalk.red(`Slider changed!! ${userTemp}`));
});

app.get("/levelSensor", (req, res) => {
   console.log(chalk.inverse.redBright(`Level sensor --------`));
});

app.get("/rotaryEncoder", (res, req) => {
   console.log(chalk.inverse.redBright(`Rotary push button --------`));
})

server.listen(wsPort, () => {
   console.log(chalk.inverse(`WS Server running on port ${wsPort}.`));
});

app.listen(httpPort, '0.0.0.0', () => {
   console.log(chalk.inverse.cyan(`Http Server running on port ${httpPort}.`));
});

function maxTempA() {
   if (maxA <= parseFloat(tempA)) maxA = parseFloat(tempA);
   maxA = String(maxA);
   console.log(`Max A      : ${maxA} °C`);
};

function minTempA() {
   if (minA != 0) {
      if (minA >= parseFloat(tempA)) minA = parseFloat(tempA);
   } else {
      minA = parseFloat(tempA);
   ;}
   console.log(`Min A      : ${minA} °C`);
};
```

```
function maxTempB() {
   if (maxB <= parseFloat(tempB)) maxB = parseFloat(tempB);
   console.log(`Max B      : ${maxB} °C`);
};

function minTempB() {
   if (minB != 0) {
      if (minB >= parseFloat(tempB)) minB = parseFloat(tempB);
   } else {
      minB = parseFloat(tempB);
   ;}
   console.log(`Min B      : ${minB} °C`);
};

function averageTempA() {
   countA++;
   sumTempA = sumTempA + parseFloat(tempA);
   avgTempA = sumTempA / countA;
   avgTempA = avgTempA.toFixed(2);
   console.log(`Channel A avg : ${avgTempA} °C`);
}

function averageTempB() {
   countB++;
   sumTempB = sumTempB + parseFloat(tempB);
   avgTempB = sumTempB / countB;
   avgTempB = avgTempB.toFixed(2);
   console.log(`Channel B avg : ${avgTempB} °C`);
}

function lastTenA () {
   lastTenChanA.unshift(tempA);
   lastTenChanA.pop();
}

function lastTenB () {
   lastTenChanB.unshift(tempB);
   lastTenChanB.pop();
}

async function addToDatabase() {
```

```
    if (opStatus == true || opStatus == 1)
    {
        const connection = await sqlCon();
        console.log(`----> Add to database`);
        console.log(`----> ${newTableName}`);
        var dataQuery = `INSERT INTO ${newTableName} (timeStamp, channelA, channelB) values
('${timeStamp}', '${tempA}', '${tempB}')`;
        await connection.execute(dataQuery);
        console.log(chalk.inverse(`----> Record inserted to database : ${dataQuery}`))
        await connection.end();
    }
    else
    {
        console.log(chalk.inverse(`----> Did not add to database. Operation Status : ${opStatus}`));
    }
}

setInterval(lastTenA, myInterval);
setInterval(lastTenB, myInterval);
setInterval(addToDatabase, dataBaseInterval); // Save to database
process.on('SIGINT', () => {
    console.log("Control + C detected. Closing database connection...");
    process.exit(); // Exit the app
});
```

login.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <script src="/jquery-3.7.1.js"></script>
    <title>ED-1 User Login</title>
  </head>
  <body>
    <!-- Header -->
    <header>
      <img class="logo" src="images/ed-logo-01.png" />
      <h6 class="date"></h6>
    </header>
    <div class="login-form-div">
```

```
    <h2>User Login</h2>
    <form class="login-form" action="/login" method="POST">
      <input
        class="input-field"
        id="username"
        type="text"
        name="username"
        value=""
        example="User name"
      />
      <br />
      <input
        class="input-field"
        id="password"
        type="password"
        name="password"
      />
      <br />
      <input class="submit-button" type="submit" value="Submit" />
      <input class="submit-button" type="reset" value="Reset" />
    </form>
  </div>
  <!-- Footer -->
  <%- include("components/footer") %>
  <script src="/scripts.js"></script>
 </body>
</html>
```

index.ejs

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" type="text/css" href="css/index.css" />
  <script src="/plotly-3.0.1.min.js" charset="utf-8"></script>
  <script src="/jquery-3.7.1.js" charset="utf-8"></script>
  <title>ED-1 Dashboard</title>
 </head>
 <body>
  <!-- Header -->
  <%- include("components/header") %>
```

```
    <!-- Channel A Modal -->
    <%- include("components/modal-A") %>
    <!-- Channel B Modal -->
    <%- include("components/modal-B") %>
    <!-- Indicator Container -->
    <%- include("components/indicator") %>
    <div class="root-div">
      <div>
        <p class="my-time"></p>
        <!-- CPU Temperature Indicator -->
        <%- include("components/cpu-temp") %>
        <!-- Control panel -->
        <%- include("components/control-panel") %>

        <a href="/getTableNames">
          <div class="historyBtn">Recorded<br />Data</div>
        </a>
      </div>
      <div class="data-display-container">
        <!-- Channel A temperature indicator -->
        <%- include("components/channel-A-temp") %>
        <!-- Channel A temperature graph -->
        <%- include("components/channel-A-graph")%>
      </div>
      <div class="data-display-container">
        <!-- Channel B temperature indicator -->
        <%- include("components/channel-B-temp") %>
        <!-- Channel B temperature graph -->
        <%- include("components/channel-B-graph")%>
      </div>
    </div>
    <br />
    <!-- Footer -->
    <%- include("components/footer") %>
    <script src="/scripts.js"></script>
  </body>
</html>
```

history.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" type="text/css" href="css/index.css" />
  <script src="/plotly-3.0.1.min.js" charset="utf-8"></script>
  <script src="/jquery-3.7.1.js" charset="utf-8"></script>
  <title>ED-1 Recorder Data</title>
</head>
<body>
  <!-- Header -->
  <%- include("components/header") %>
  <!-- Channel A Modal -->
  <%- include("components/modal-A") %>
  <!-- Channel B Modal -->
  <%- include("components/modal-B") %>
  <!-- Indicator Container -->
  <%- include("components/indicator") %>
  <div class="root-div">
    <div>
      <p class="my-time"></p>
      <!-- CPU Temperature Indicator -->
      <%- include("components/cpu-temp") %>
      <!-- Control panel -->
      <%- include("components/control-panel") %>
      <a href="/main">
        <div class="historyBtn">Back to<br />Dashbord</div>
      </a>
    </div>
    <div class="data-display-container">
      <!-- Channel A temperature indicator -->
      <%- include("components/channel-A-temp") %>
      <!-- Channel A temperature graph -->
      <%- include("components/channel-A-graph")%>
    </div>
    <div class="data-display-container">
      <!-- Channel B temperature indicator -->
      <%- include("components/channel-B-temp") %>
      <!-- Channel B temperature graph -->
      <%- include("components/channel-B-graph")%>
    </div>
  </div>
  <div class="tablesSection">
    <div class="tableNames">
```

```
      <h1>Database Tables</h1>
      <ol>
       <% tables.forEach(table => { %>
       <li class="tableList">
        <a
         class="aList"
         href="/showTableData"
         onclick="handleTableClick('<%= table %>')"
        >
         <%= table %>
        </a>
       </li>
       <% }); %>
      </ol>
     </div>
    </div>
    <br />
    <!-- Footer -->
    <%- include("components/footer") %>
    <script src="/scripts.js"></script>
  </body>
</html>
```

history2.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <script src="/plotly-3.0.1.min.js" charset="utf-8"></script>
    <script src="/jquery-3.7.1.js" charset="utf-8"></script>
    <title>ED-1 Recorder Data</title>
  </head>
  <body>
    <!-- Header -->
    <%- include("components/header") %>
    <!-- Channel A Modal -->
    <%- include("components/modal-A") %>
    <!-- Channel B Modal -->
    <%- include("components/modal-B") %>
    <!-- Indicator Container -->
```

```
<%- include("components/indicator") %>
<div class="root-div">
  <div>
    <p class="my-time"></p>
    <!-- CPU Temperature Indicator -->
    <%- include("components/cpu-temp") %>
    <!-- Control panel -->
    <%- include("components/control-panel") %>
    <a href="/main">
      <div class="historyBtn">Back to<br />Dashbord</div>
    </a>
  </div>
  <div class="data-display-container">
    <!-- Channel A temperature indicator -->
    <%- include("components/channel-A-temp") %>
    <!-- Channel A temperature graph -->
    <%- include("components/channel-A-graph")%>
  </div>
  <div class="data-display-container">
    <!-- Channel B temperature indicator -->
    <%- include("components/channel-B-temp") %>
    <!-- Channel B temperature graph -->
    <%- include("components/channel-B-graph")%>
  </div>
</div>
<div>
  <div class="tablesSection">
    <div class="tableNames">
      <h1>Database Tables</h1>
      <ol>
        <% tables.forEach(table => { %>
        <li class="tableList">
          <a
            class="aList"
            href="/showTableData"
            onclick="handleTableClick('<%= table %>')"
          >
            <%= table %>
          </a>
        </li>
        <% }); %>
      </ol>
```

```
    </div>
    <div class="table-data">
      <!-- to do -->
      <h1>Table Data</h1>
      <h4><%= tableName %></h4>
      <ol>
        <div class="tableDataTitle">
          <h4>Time</h4>
          <h4>A</h4>
          <h4>B</h4>
        </div>
        <% tablesData.forEach(tableData => { %>
        <li class="theList">
          <%= tableData.timeStamp %> - <%= tableData.channelA %> -
          <%=tableData.channelB %>
        </li>
        <% }); %>
      </ol>
    </div>
    <div class="plots">
      <h1>Graph</h1>
      <div id="tablePlot"></div>
    </div>
  </div>
</div>
<br />
<!-- Footer -->
<%- include("components/footer") %>
<script src="/scripts.js"></script>
<script>
  var tablesData = JSON.parse(`<%- JSON.stringify(tablesData) %>`);
  console.log(`TEST : ${tablesData[0].timeStamp}`);
  tablesData = tablesData.map((t) => ({
    ...t,
    timeStamp: t.timeStamp,
    channelA: t.channelA,
    channelB: t.channelB,
  }));
  console.log("Formatted tablesData:", tablesData);

  var tempTraceA = {
    y: tablesData.map((t) => t.channelA),
```

```
      x: tablesData.map((t) => t.timeStamp),
      type: "scatter",
      name: "Cooler",
      line: {
        color: "slategray",
        width: 2,
        shape: "spline",
      },
    };
    var tempTraceB = {
      y: tablesData.map((t) => t.channelB),
      x: tablesData.map((t) => t.timeStamp),
      type: "scatter",
      name: "Boiler",
      line: {
        color: "#f99a",
        width: 2,
        shape: "spline",
      },
    };
    var datas = [tempTraceA, tempTraceB];
    Plotly.newPlot("tablePlot", datas, myPlotLayout2, {
      displayModeBar: false,
    });
  </script>
 </body>
</html>
```

errorLogin.ejs

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" type="text/css" href="css/index.css" />
  <script src="/jquery-3.7.1.js"></script>
  <title>ED-1 User Login</title>
 </head>
 <body>
  <!-- Header -->
  <header>
```

```
    <img class="logo" src="images/ed-logo-01.png" />
    <h6 class="date"></h6>
   </header>
   <div class="login-form-div">
    <h2>User Login</h2>
    <h3 class="error-message">Invalid username or password</h3>
    <div class="home-link-div">
     <h3>
      <a class="home-link" href="/" rel="noreferrer">Back to login</a>
     </h3>
    </div>
   </div>
   <!-- Footer -->
   <%- include("components/footer") %>
   <script src="/scripts.js"></script>
  </body>
</html>
```

channel-A-graph.ejs

```
<div class="container-item last-ten">
  <!-- <h3 class="">Channel A</h3> -->
  <h4 class="">Last min</h4>
  <div id="last-ten-A"></div>
  <!-- <span class="lastTenA"><%=lastTenChanA %></span> -->
</div>
```

channel-A-temp.ejs

```
<div class="container-item" onclick="dataOnclickEventA()">
  <h3>Channel A</h3>
  <h2 class="temp-display tempA"><%=tempA %> °C</h2>
  <h2 class="res-display resA"><%=resA %> Ω</h2>
</div>
```

channel-B-graph.ejs

```
<div class="container-item last-ten">
  <!-- <h3 class="">Channel B</h3> -->
  <h4 class="">Last min</h4>
  <div id="last-ten-B"></div>
  <!-- <span class="lastTenB"><%=lastTenChanB %></span> -->
</div>
```

channel-B-temp.ejs

```
<div class="container-item" onclick="dataOnclickEventB()">
  <h3>Channel B</h3>
  <h2 class="temp-display tempB"><%=tempB %> °C</h2>
  <h4 class="res-display resB"><%=resB %> Ω</h4>
</div>
```

control-panel.ejs

```
<div class="control-panel">
  <div class="user-temp-display">
    <span class="user-temp"><%=userTemp %> °C</span>
  </div>
  <div class="slider-container">
    <input
      class="slider"
      oninput="sliderInput()"
      type="range"
      min="0"
      max="150"
      value="<%=userTemp %>"
    />
    <!-- <label>- Temperature Adjust +</label> -->
    <img class="slider-label" src="images/slider-label.png" />
  </div>
  <div class="startBtn" onclick="startDist()"><h2>START</h2></div>
  <div class="stopBtn" onclick="stopDist()"><h2>STOP</h2></div>
</div>
```

cpu-temp.ejs

```
<div class="cpu-temp">
  CPU :
  <span class="cpuTemp"><%=cpuTemp %> °C</span>
</div>
```

exit-btn.ejs

```
<a class="exit-btn" href="/exitBtn">
  <div class="exit-btn-div"></div>
</a>
```

footer.ejs

```
<footer>
  <div class="footer-container">
    <h4 class="copyrights">
      <a
        class="aLinks"
        href="https://www.linkedin.com/in/jimouz/"
        target="_blank"
        rel="noreferrer"
      >
        Dimitris Ouzounis
      </a>
      &copy; 
    </h4>
    <h4 class="copy-date"></h4>
  </div>
</footer>
```

header.ejs

```
<header>
  <img class="logo" src="images/ed-logo-01.png" />
  <h6 class="date"></h6>
  <%- include("exit-btn") %>
</header>
```

indicator.ejs

```
<div class="indicator-container">
  <div id="out1" class="indicator output-off">Output 1</div>
  <div id="out2" class="indicator output-off">Output 2</div>
  <div id="tank" class="indicator level-off">Level...</div>
  <div id="emg" class="indicator emg-off">EMG</div>
</div>
```

modal-A.ejs

```
<div class="modal" id="modalA">
  <div class="close-modal" onclick="closeModal()">CLOSE</div>
  <br /><br /><br />
  <div
    class="container-item-modal"
    onclick="dataOnclickEvent()"
    style="margin: 4px auto"
  >
```

```
    <h3>Channel A</h3>
    <h2 class="temp-display tempA"><%=tempA %> °C</h2>
    <h2 class="res-display resA"><%=resA %> Ω</h2>
  </div>
  <div class="modal-data">
    <h4>Max : <span class="maxA"><%=maxA %> °C</span></h4>
    <h4>Min : <span class="minA"><%=minA %> °C</span></h4>
    <h4>Avg : <span class="averageA"><%=avgTempA %></span></h4>
  </div>
</div>
```

modal-B.ejs

```
<div class="modal" id="modalB">
  <div class="close-modal" onclick="closeModal()">CLOSE</div>
  <br /><br /><br />
  <div
    class="container-item-modal"
    onclick="dataOnclickEvent()"
    style="margin: 4px auto"
  >
    <h3>Channel B</h3>
    <h2 class="temp-display tempB"><%=tempB %> °C</h2>
    <h2 class="res-display resB"><%=resB %> Ω</h2>
  </div>
  <div class="modal-data">
    <h4>Max : <span class="maxB"><%=maxB %> °C</span></h4>
    <h4>Min : <span class="minB"><%=minB %> °C</span></h4>
    <h4>Avg : <span class="averageB"><%=avgTempB %></span></h4>
  </div>
</div>
```

index.css

```
html {
    scroll-behavior: smooth;
    background-image: url('../images/bg-01.png');
    background-attachment: fixed;
    background-position: top;
    background-size: cover;
    background-blend-mode: darken;
    background-color: #ff48;
    color: #444;
```

```
    font-family: Arial, Helvetica, sans-serif;
    letter-spacing: 1px;
    text-decoration: none;
    margin: 0;
    cursor: auto;
    height: max-content;
}
body {
    background-color: #fffb;
    margin: 0;
    padding-top: 10px;
    min-height: max-content;
}
header {
    position: fixed;
    top: 0;
    width: 100vw;
    text-align: center;
    background-color: #fffb;
    display: grid;
    grid-template-columns: 60% auto auto;
    text-align: left;
    z-index: 99;
}
.logo {
    width: 52px;
    padding: 12px;
}
.exit-btn {
    background-image: url('../images/logout-icon.png');
    background-size: cover;
    background-position: center;
    padding: 8px;
    width: 16px;
    height: 16px;
    text-align: center;
    border-radius: 8px;
    border: 1px solid #aaaa;
    background-color: #aaaa;

    margin: 12px 20px 12px 12px;
    cursor: pointer;
```

```
}
.exit-btn {
    font-size: 16px;
    font-weight: bold;
}
.date {
    border-radius: 8px;
    border: 1px solid #aaaa;
    background-color: #fff;
    text-align: center;
    padding: 8px;
    font-size: 16px;
    margin: 12px 20px 12px 12px;
}
.head {
    margin: 10px 0px 10px 10px;
}
.error-message {
    color: red;
}
.home-link-div {
    margin: auto;
    margin-top: 40px;
    height: 40px;
    width: 200px;
}
.home-link{
    background-color: #fffa;
    text-align: center;
    letter-spacing: 2px;
    border-radius: 8px;
    margin-top: 8px;
    padding: 8px;
    box-shadow: #444 2px 2px 4px;
}
.login-form-div {
    background-color: #aaaa;
    margin-top: 120px;
    margin-bottom: 400px;
    margin-left: auto;
    margin-right: auto;
    min-width: 280px;
```

```
    min-height: 300px;
    max-width: 360px;
    /* border: 1px solid black; */
    border-radius: 8px;
    text-align: center;
    box-shadow: #444 2px 2px 4px;
}
.login-form-div > h2 {
    padding-top: 20px;
    color: #555;
    text-shadow: #0005 2px 2px 4px;
    letter-spacing: 2px;

}
.login-form {
    margin-top: 20px;
    max-width: 360px;

}
.input-field {
    margin-top: 10px;
    margin-bottom: 10px;
    min-height: 36px;
    min-width: 290px;
    border-style: groove;
    border-color: #aaaa;
    border-width: 0px 0px 1px 1px;
    border-radius: 0 8px 0px 8px;
    background-color: #fff9;
    box-shadow: #444 2px 2px 4px;
}

.submit-button, .historyBtn {
    margin-top: 32px;
    border: none;
    height: 32px;
    width: 120px;
    border-radius: 8px;
    box-shadow: #444 2px 2px 4px;
    letter-spacing: 2px;
    font-size: 16px;
    font-weight: bold;
```

```
    color: #666;
}
.submit-button:hover, .historyBtn:hover {
    background-color: #4444;
    box-shadow: 2px 2px 8px #444;
    transition: 0.2s;
    cursor: pointer;
}
.historyBtn {
    height: 52px;
    width: 300px;
    font-size: 20px;
    color: #666;
    padding: 8px;
    background-color: #fffa;
}
a {
    color: #444;
}
.indicator-container {
    column-span: 2;
    display: grid;
    grid-template-columns: auto auto auto auto;
    grid-template-rows: auto;
    grid-gap: 20px;
    margin: 80px auto 20px auto;
    width: 700px;
    text-align: center;

}
.indicator {
    margin: 0 auto;
    font-size: 20px;
    font-weight: bold;
    padding-top: 18px;
    min-width: 130px;
    max-width: 130px;
    height: 40px;
    box-shadow: #444 2px 2px 4px;
    border-radius: 8px;
}
.output-off {
```

```
    background: linear-gradient(#fff, #f004);
}
.level-off {
    background: linear-gradient(#fff, #08f4);
}
.emg-off {
    background: linear-gradient(#fff, #aaa4);
}
.root-div {
    /* margin-top: 10px; */
    margin-left: auto;
    margin-right: auto;
    width: auto;
    display: grid;
    grid-template-columns: auto auto auto;
    grid-template-rows: auto auto;
    grid-column-gap: 8px;
    grid-row-gap: 8px;
    text-align: center;
    justify-content: center;
}
.control-panel {
    max-width: 300px;
    border-radius: 8px;
    display: flex;
    flex-wrap: wrap;
    justify-content:start;
    margin-left: auto;
    margin-right: auto;
    padding: 8px;
    margin-top: 8px;
    box-shadow: #444 2px 2px 4px;
    background-color: #eeea;
}
.user-temp-display {
    height: 60px;
    width: 100%;
    padding-top: 20px;
    border-radius: 8px;
    /* border: 2px groove #4444; */
    background: linear-gradient(#aaa, #ddd, #aaa);
}
```

```
.user-temp {
    margin-left: auto;
    margin-right: auto;
    font-size: 40px;

}
.slider {
    margin-left: auto;
    margin-right: auto;
    margin-top: 10px;
    -webkit-appearance: none;
    appearance: none;
    width: 100%;
    height: 40px;
    background-color: #aaa8;
    opacity: 0.6;
    -webkit-transition: .2s;
    transition: opacity .2s;
}
.slider:hover {
    opacity: 1;
    box-shadow: 2px 2px 8px #444;
}
/* The slider handle (use -webkit- (Chrome, Opera, Safari, Edge) and -moz- (Firefox) to override
default look) */
.slider::-webkit-slider-thumb {
    -webkit-appearance: none;
    outline: none;
    width: 30px;
    height: 40px;
    background-color: #9998;
    border: none;
    cursor: pointer;
}
.slider::-moz-range-thumb {
    outline: none;
    width: 30px;
    height: 40px;
    background-color: #9998;
    border: none;
    cursor: pointer;
    border-radius: 0px;
```

```
}
.slider-label {
    width: 280px;
    opacity: 0.8;
}
.startBtn, .stopBtn {
    margin: 0 auto;
    font-weight: bold;
    width: 90px;
    height: 90px;
    border: 1px solid #aaa8;
    border-radius: 8px;
    background-color: #8888;
    box-shadow: #444 2px 2px 4px;
}
.startBtn > h2, .stopBtn > h2 {
    margin-top: 30px;
}
.stopBtn {
    color: #f00;
}
.startBtn:hover, .stopBtn:hover {
    box-shadow: 2px 2px 12px #000;
    cursor: pointer;
    border-color: #aaaf;
}
.slider-container {
    width: 280px;
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 20px;
}
label {
    font-weight: bold;
}
.data-display-container {
    margin-left: auto;
    margin-right: auto;
    display: grid;
    grid-template-rows: auto auto;
    grid-template-columns: auto;
    grid-column-gap: 8px;
```

```
    grid-row-gap: 12px;
    max-width: 300px;
    text-align: center;
}
.container-item, .container-item-modal {
    width: 300px;
    border-radius: 8px;
    background-color: #fffa;
    display: grid;
    grid-template-rows: auto auto auto;
    grid-template-columns: auto;
    grid-column-gap: 8px;
    grid-row-gap: 0px;
    align-content: start;
    box-shadow: #444 2px 2px 4px;
}
.container-item:hover {
    cursor: pointer;
    background-color: rgba(255, 226, 215, 0.4);
    transition: 0.3s;
}
.container-item > h3, .container-item-modal > h3 {
    margin: 4px auto;
}

.temp-display, .res-display, .my-time {
    border-radius: 8px;
    font-size: 40px;
    margin-left: auto;
    margin-right: auto;
    text-align: center;
}
.temp-display {
    background-color: #ccca;
    min-width: 280px;
    height: 60px;
    font-family: 'Courier New', Courier, monospace;
    font-weight: 900;
    margin: auto;
    padding-top: 16px;
    text-shadow: 4px 4px 4px #888c;
}
```

```
.res-display {
    margin: 4px auto;
    font-size: 16px;
    max-width: 120px;
    padding: 4px;
}
.my-time, .cpu-temp {
    margin-left: auto;
    margin-right: auto;
    background-color: #eee;
    display: block;
    width: 300px;
    color: #444;
    font-weight: bold;
    text-align: center;
    letter-spacing: 2px;
    border-radius: 8px;
}
.last-ten {
    padding: 2px;
    width: 300px;
    font-size: x-small;
}
#last-ten-A, #last-ten-B {
    height: 300px;
}
.my-time {
    min-height: 46px;
    margin-top: 0px;
    box-shadow: #444 2px 2px 4px;
}
.cpu-temp {
    box-shadow: #444 2px 2px 4px;
    margin-top: -20px;
    font-size: 20px;
}
footer {
    margin-top: 100px;
    position: relative;
    bottom: 0px;
    left: 0;
    height: 30vh;
```

```
    width: 100%;
    background: linear-gradient(#999b, #fffe);
}
.footer-container {
    margin-left: auto;
    margin-right: auto;
    display: flex;
    width: fit-content;
}
.copy-date, .copyrights {
    width: fit-content;
}
.tablesSection {
    display: flex;
    flex-direction: row;
    align-items: flex-start;
    max-width: 1200px;
    margin: auto;
    border-radius: 8px;
    background-color: #eee;
    box-shadow: #444 2px 2px 4px;
}
.tableNames, .table-data, .plots {
    margin: 10px auto;
    padding: 8px;
    border-radius: 8px;
    background-color: #eee;
    font-size: 16px;
    width: 280px;
    box-shadow: #444 2px 2px 4px;
    /* border: 1px solid black; */
}
.plots {
    width: 40%;
}
.tableList {
    border-radius: 8px;
    /* border: 1px solid black; */
    max-width: 400px;
    padding: 6px;
}
.tableDataTitle {
```

```
    max-width: 200px;
    display: flex;
    flex-direction: row;
    justify-content: space-around;
}
.aList {
    border-radius: 8px;
    /* border: 1px solid black; */
    width: auto;
    padding: 6px 12px;
}
.theList {
    padding: 4px;
}
.aList:hover {
    background-color: coral;
}
.modal {
    display: none;
    position: absolute;
    top: 100px;
    left: 29%;
    /* left: auto;
    right: auto; */
    background-color: #aaae;
    height: 400px;
    /* width: 80%; */
    width: 800px;
    margin-left: auto;
    margin-right: auto;
    padding: 4px;
    border-radius: 8px;
    box-shadow: #444 2px 2px 4px;
    z-index: 999;
}
.close-modal {
    position: absolute;
    right: 16px;
    width: fit-content;
    padding: 8px;
    /* text-align: right; */
    margin: 8px;
```

```
    border-radius: 8px;
    border: 1px solid #fffa;
    background-color: #fffa;
    font-weight: bold;
    box-shadow: #444 2px 2px 4px;
}
.close-modal:hover {
    cursor: pointer;
    background-color: #f00;
    transition: 0.3s;
}
.modal-data {
    width: 290px;
    margin: 0px auto;
    padding: 4px;
    border-radius: 8px;
    background-color: #fffa;
    box-shadow: #444 2px 2px 4px;
}
.aLinks:hover {
    cursor: pointer;
    color: #fff;
    transition: 0.3s;
}
a {
    text-decoration: none;
}
@media screen and (max-width: 800px) {
    .root-div {
        grid-template-rows:  auto auto;
        grid-template-columns: auto;
    }
    .header {
        grid-template-rows: auto;
        grid-template-columns: auto auto 50%;
    }
    .head {
        margin-top: 12px;
    }
    .modal {
        width: 90%;
        left: 4%;
```

```
        top: 660px;
    }
    .date {
        margin: 10px;
        grid-column: 2 / 3;
        text-align: left;
        max-width: fit-content;
        font-size: 10px;
    }
    .exit-btn {
        margin: 12px 4px;
    }
    .data-display-container {
        grid-template-rows:  auto auto auto auto;
        grid-template-columns: auto;
    }
    .container-item {
        padding-bottom: 10px;
    }
    .indicator-container {
        grid-template-columns:  auto auto;
        grid-template-rows: auto auto;
        width: 300px;
    }
    .tablesSection {
        max-width: 320px;
        flex-direction: column;
    }
    footer {
        height: 100px;
    }
}
@media screen and (max-width: 1400px) and (min-width: 600px){
    .modal {
        width: 500px;
        left: 20%;
        top: 260px;
    }
}
```

scripts.js

```
// ----------------------------------
// UI jquery scripts
// ----------------------------------

let sliderValue = 0;
let tables = "";
let tableData = "";
$(document).ready( () => {
    setInterval(myDate, 1000);
    setInterval(myClock, 1000);
    setInterval(update_temp, 1000);
    // Slider input
    document.querySelector('.slider').oninput = function() {
        sliderValue = document.querySelector('.slider').value;
        $('.user-temp').text(`${document.querySelector('.slider').value} °C`);
        sliderInput();
    }
});
// Update all values
function update_temp(){
    $.ajax({
        url: '/updateTemp',
        method: 'GET',
        success: (data) => {
            console.log(data);
            $('.cpuTemp').text(`${data.cpuTemp} °C`);
            document.querySelector('.slider').value=Number(data.userTemp);
            $('.user-temp').text(`${document.querySelector('.slider').value} °C`);
            $('.tempA').text(`${data.tempA} °C`);
            $('.resA').text(`${data.resA} Ω`);
            $('.maxA').text(`${data.maxA} °C`);
            $('.minA').text(`${data.minA} °C`);
            $('.averageA').text(`${(data.avgTempA)} °C`);
            $('.lastTenA').text(`${(data.lastTenChanA)} °C`);
            //-----------------------------------------------
            $('.tempB').text(`${data.tempB} °C`);
            $('.resB').text(`${data.resB} Ω`);
            $('.maxB').text(`${data.maxB} °C`);
            $('.minB').text(`${data.minB} °C`);
            $('.averageB').text(`${(data.avgTempB)} °C`);
            $('.lastTenB').text(`${(data.lastTenChanB)} °C`);
            if (data.sts === 1 || data.sts === true)
```

```
        {
            $('.startBtn').css('background-color', '#0f0');
        }
        else if (data.sts === 0 || data.sts === false)
        {
            $('.startBtn').css('background-color', '#8888');
        }
        (data.emg === 1 || data.emg === true)?
            $('#emg').css('background', 'red').css('color', 'white'):
            $('#emg').css('background', 'linear-gradient(#fff, #aaa4)').css('color', '#444');
        (data.out1 === 1 || data.out1 === true)?
            $('#out1').css('background', 'linear-gradient(#0f0a, #0a0)').css('color',"#fff"):
            $('#out1').css('background', 'linear-gradient(#fff, #f004)').css('color',"#444");
        // (data.out2 === 1 || data.out2 === true)?
        //     $('#out2').css('background', 'linear-gradient(#0f0a, #0a0)').css('color',"#fff"):
        //     $('#out2').css('background', 'linear-gradient(#fff, #f004)').css('color',"#444");
        (data.tankLevel === 1 || data.tankLevel === true)?
            $('#tank').text('Level MAX').css('background', 'linear-gradient(#08f8,
#08f)').css('color',"#ff0"):
            $('#tank').text('Level OK').css('background', 'linear-gradient(#fff,
#08f4)').css('color',"#444");
        plotA(data.lastTenChanA);
        plotB(data.lastTenChanB);
      }
   });
}
// Logout / Exit button
function exitBtn () {
   $.ajax({
      url: '/exitBtn',
      method: 'GET',
      success: () => {
         console.log(`----> Exit button clicked.`)
      }
   })
}
// Slider input
function sliderInput() {
   $.ajax({
      url: '/sliderChange',
      method: 'POST',
      data: {
```

```
            userTemp: sliderValue
        },
        success: (data) => {
            console.log(`Slider changed! ${data.userTemp}`);
            $('.user-temp').text(`${data.userTemp}`);
        }
    })
}
// Start button
function startDist() {
    $.ajax({
        url: '/startBtn',
        method: 'GET',
        success: (data) => {
            console.log(`Start Button pressed! ${data} `);
            $('.startBtn').prop('disabled', true);
        }
    });
}
//  Stop button
function stopDist() {
    $.ajax({
        url: '/stopBtn',
        method: 'GET',
        success: (data) => {
            console.log(`Stop Button pressed! ${data}`);
            $('.startBtn').prop('disabled', false);
        }
    });
};
//  History button
function historyBtn() {
    $.ajax({
        url: '/getTableNames',
        method: 'GET',
        success: (data) => {
            console.log(`History Button pressed! ${data}`);
            console.log(data.tables);
            $('.table-names').text(`${data.tableNames}`);
            $('.table-names').text(`Test`);
        }
    });
```

```
};
// Click a table to display data
function handleTableClick(tableName) {
    $.ajax({
        url: `/getTableData?table=${tableName}`,
        method: 'GET',
        success: (data) => {
            tablePlot(data);
            console.log(`Table Button pressed! ${tableName}`);
            console.log(`Table Data : ${data.tables}`);
            console.log(`Table Data : ${data.tablesData}`);
            tablePlot(data);
        }
    })
};
//  Clock
function myClock() {
    var date = new Date();
    function addZero(x) {
        if (x < 10) {
            return x = '0' + x;
        } else {
            return x;
        }
    }
    var h = addZero(date.getHours());
    var m = addZero(date.getMinutes());
    var s = addZero(date.getSeconds());
    $('.my-time').text(h + ':' + m + ':' + s);
}
function myDate() {
    const month = ['January', 'February',
                   'March', 'April', 'May',
                   'June', 'July', 'August',
                   'September', 'October',
                   'November', 'December'];
    const weekday = ['Sunday', 'Monday',
                   'Tuesday', 'Wednesday',
                   'Thursday', 'Friday', 'Saturday'];
    const newDate = new Date();
    fullDate = (`${weekday[newDate.getDay()]} ${newDate.getDate()} $
{month[newDate.getMonth()]} ${newDate.getFullYear()}`);
```

```
    fullDateSmall = (`${newDate.getDate()} ${month[newDate.getMonth()].slice(0, 3)} $
{newDate.getFullYear()}`);
    screen.width < 600 ? $('.date').text(fullDateSmall): $('.date').text(fullDate);
    $('.copy-date').text(`${newDate.getFullYear()}`);
}
function dataOnclickEventA() {
    $('.modal').css('display', 'none');
    $('#modalA').css('display', 'block');
}
function dataOnclickEventB() {
    $('.modal').css('display', 'none');
    $('#modalB').css('display', 'block');
}
function closeModal() {
    $('.modal').css('display', 'none');
}
var myPlotLayout = {
    yaxis: {range: [0, 140], color: "#aaa", minor: {gridcolor: "#fff"}},
    xaxis: {color: "#aaa", minor: {gridcolor: "#fff"}},
    height: 240,
    margin: {pad:0, r:0, l:36, b:20, t:0},
    paper_bgcolor: "#fff1",
    plot_bgcolor: "#fff0",
    showgrid: true,
    gridcolor: "#000",
};
var myPlotLayout2 = {
    yaxis: {range: [0, 140], color: "#aaa", minor: {gridcolor: "#fff"}},
    xaxis: {color: "#aaa", minor: {gridcolor: "#fff"}},
    height: "auto",
    margin: {pad:0, r:0, l:36, b:20, t:0},
    paper_bgcolor: "#fff1",
    plot_bgcolor: "#fff0",
    showgrid: true,
    gridcolor: "#000",
};
function plotA(tempDataA) {
    var lastTenNumberA = tempDataA;
    var tempTrace = {
        y: lastTenNumberA,
        x: [0, 1, 2, 3, 4, 5, 6],
        type: 'scatter',
```

```
        line: {
              color: 'slategray',
              width: 2,
              shape: 'spline'},
        };
    var data = [tempTrace];
    Plotly.newPlot('last-ten-A', data, myPlotLayout, {displayModeBar: false});
};
function plotB(tempDataB) {
    var lastTenNumberB = tempDataB;
    var tempTrace = {
        y: lastTenNumberB,
        x: [0, 1, 2, 3, 4, 5, 6],
        type: 'scatter',
        line: {
              color: 'slategray',
              width: 2,
              shape: 'spline'},
        };
    var data = [tempTrace];
    Plotly.newPlot('last-ten-B', data, myPlotLayout,  {displayModeBar: false, scrollZoom: false});
};
```

buttons.js

```
const express = require("express");
const router = new express.Router();
const chalk = require("chalk");
// const mysql = require('mysql2/promise');

module.exports = (io) => {
    // Define my routes here

    // Start button route
    router.get("/startBtn", async (req, res) => {
        io.emit("start");
        console.log(chalk.green("Start Button pressed!"));
        opStatus = true;
        res.send({ sts: opStatus });
    });
    // Stop button route
    router.get("/stopBtn", async (req, res) => {
        io.emit("stop");
```

```
        console.log(chalk.red("Stop Button pressed!"));
        opStatus = false;
        res.send({ sts: opStatus });
    });
    // Emergency button route
    router.get("/emgBtn", (req, res) => {
        console.log(chalk.inverse.red("EMERGENCY Button pressed!"));
        opStatus = false;
        res.send({ sts: opStatus });
    });
    return router;
  };
```

history.js

```
// -----------------------------------
// UI jquery scripts for history.ejs
// -----------------------------------

let sliderValue = 0;
let tables = "";
let tableData = "";
$(document).ready( () => {
    setInterval(myDate, 1000);
    setInterval(myClock, 1000);
    setInterval(update_temp, 1000);
    // Slider input
    document.querySelector('.slider').oninput = function() {
        sliderValue = document.querySelector('.slider').value;
        $('.user-temp').text(`${document.querySelector('.slider').value} °C`);
        sliderInput();
    }
});
// Update all values
function update_temp(){
    $.ajax({
        url: '/updateTemp',
        method: 'GET',
        success: (data) => {
            console.log(data);
            $('.cpuTemp').text(`${data.cpuTemp} °C`);
            document.querySelector('.slider').value=Number(data.userTemp);
            $('.user-temp').text(`${document.querySelector('.slider').value} °C`);
```

```
            $('.tempA').text(`${data.tempA} °C`);
            $('.resA').text(`${data.resA} Ω`);
            $('.maxA').text(`${data.maxA} °C`);
            $('.minA').text(`${data.minA} °C`);
            $('.averageA').text(`${(data.avgTempA)} °C`);
            $('.lastTenA').text(`${(data.lastTenChanA)} °C`);
            //-----------------------------------------------
            $('.tempB').text(`${data.tempB} °C`);
            $('.resB').text(`${data.resB} Ω`);
            $('.maxB').text(`${data.maxB} °C`);
            $('.minB').text(`${data.minB} °C`);
            $('.averageB').text(`${(data.avgTempB)} °C`);
            $('.lastTenB').text(`${(data.lastTenChanB)} °C`);
            if (data.sts === 1 || data.sts === true)
            {
               $('.startBtn').css('background-color', '#0f0');
            }
            else if (data.sts === 0 || data.sts === false)
            {
               $('.startBtn').css('background-color', '#8888');
            }
            (data.emg === 1 || data.emg === true)?
               $('#emg').css('background', 'red').css('color', 'white'):
               $('#emg').css('background', 'linear-gradient(#fff, #aaa4)').css('color', '#444');
            (data.out1 === 1 || data.out1 === true)?
               $('#out1').css('background', 'linear-gradient(#0f0a, #0a0)').css('color',"#fff"):
               $('#out1').css('background', 'linear-gradient(#fff, #f004)').css('color',"#444");
            // (data.out2 === 1 || data.out2 === true)?
            //    $('#out2').css('background', 'linear-gradient(#0f0a, #0a0)').css('color',"#fff"):
            //    $('#out2').css('background', 'linear-gradient(#fff, #f004)').css('color',"#444");
            (data.tankLevel === 1 || data.tankLevel === true)?
               $('#tank').text('Level MAX').css('background', 'linear-gradient(#08f8,
#08f)').css('color',"#ff0"):
               $('#tank').text('Level OK').css('background', 'linear-gradient(#fff,
#08f4)').css('color',"#444");
            plotA(data.lastTenChanA);
            plotB(data.lastTenChanB);
         }
      });
}
// Logout / Exit button
function exitBtn () {
```

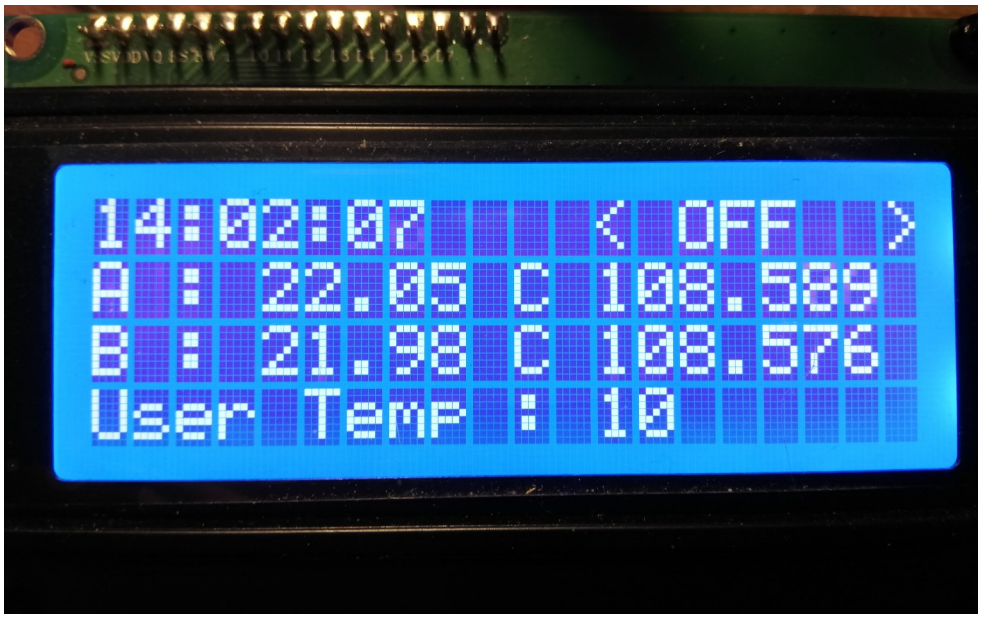
```
    $.ajax({
        url: '/exitBtn',
        method: 'GET',
        success: () => {
            console.log(`----> Exit button clicked.`)
        }
    })
}
// Slider input
function sliderInput() {
    $.ajax({
        url: '/sliderChange',
        method: 'POST',
        data: {
            userTemp: sliderValue
        },
        success: (data) => {
            console.log(`Slider changed! ${data.userTemp}`);
            $('.user-temp').text(`${data.userTemp}`);
        }
    })
}
// Start button
function startDist() {
    $.ajax({
        url: '/startBtn',
        method: 'GET',
        success: (data) => {
            console.log(`Start Button pressed! ${data} `);
            $('.startBtn').prop('disabled', true);
        }
    });
}
//  Stop button
function stopDist() {
    $.ajax({
        url: '/stopBtn',
        method: 'GET',
        success: (data) => {
            console.log(`Stop Button pressed! ${data}`);
            $('.startBtn').prop('disabled', false);
        }
```

```
    });
}
//  History button
function historyBtn() {
    $.ajax({
        url: '/getTableNames',
        method: 'GET',
        success: (data) => {
            console.log(`History Button pressed! ${data}`);
            console.log(data.tables);
            $('.table-names').text(`${data.tables}`);
        }
    });
}
// Click a table to display data
function handleTableClick(tableName) {
    $.ajax({
        url: `/getTableData?table=${tableName}`,
        method: 'GET',
        success: (data) => {
            tablePlot(data);
            console.log(`Table Button pressed! ${tableName}`);
            console.log(`Table Data : ${data.tables}`);
            console.log(`Table Data : ${data.tablesData}`);
            tablePlot(data);
        }
    })
};
//  Clock
function myClock() {
    var date = new Date();
    function addZero(x) {
        if (x < 10) {
            return x = '0' + x;
        } else {
            return x;
        }
    }
    var h = addZero(date.getHours());
    var m = addZero(date.getMinutes());
    var s = addZero(date.getSeconds());
    $('.my-time').text(h + ':' + m + ':' + s);
```

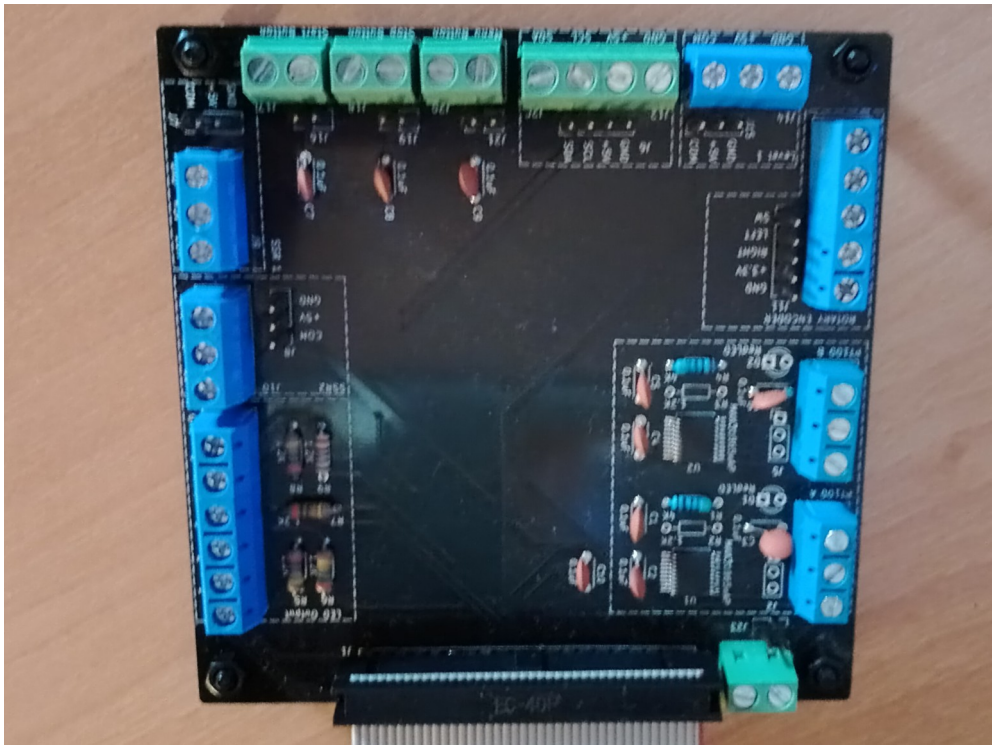
```
}
function myDate() {
    const month = ['January', 'February',
                   'March', 'April', 'May',
                   'June', 'July', 'August',
                   'September', 'October',
                   'November', 'December'];
    const weekday = ['Sunday', 'Monday',
                   'Tuesday', 'Wednesday',
                   'Thursday', 'Friday', 'Saturday'];
    const newDate = new Date();
    fullDate = (`${weekday[newDate.getDay()]} ${newDate.getDate()} $
{month[newDate.getMonth()]} ${newDate.getFullYear()}`);
    fullDateSmall = (`${newDate.getDate()} ${month[newDate.getMonth()].slice(0, 3)} $
{newDate.getFullYear()}`);
    screen.width < 600 ? $('.date').text(fullDateSmall): $('.date').text(fullDate);
    $('.copy-date').text(`${newDate.getFullYear()}`);
}
function dataOnclickEventA() {
    $('.modal').css('display', 'none');
    $('#modalA').css('display', 'block');
}
function dataOnclickEventB() {
    $('.modal').css('display', 'none');
    $('#modalB').css('display', 'block');
}
function closeModal() {
    $('.modal').css('display', 'none');
}
var myPlotLayout = {
    yaxis: {range: [0, 140], color: "#aaa", minor: {gridcolor: "#fff"}},
    xaxis: {color: "#aaa", minor: {gridcolor: "#fff"}},
    height: 240,
    margin: {pad:0, r:0, l:36, b:20, t:0},
    paper_bgcolor: "#fff1",
    plot_bgcolor: "#fff0",
    showgrid: true,
    gridcolor: "#000",

};
var myPlotLayout2 = {
    yaxis: {range: [0, 140], color: "#aaa", minor: {gridcolor: "#fff"}},
```

```
    xaxis: {color: "#aaa", minor: {gridcolor: "#fff"}},
    height: auto,
    margin: {pad:0, r:0, l:36, b:20, t:0},
    paper_bgcolor: "#fff1",
    plot_bgcolor: "#fff0",
    showgrid: true,
    gridcolor: "#000",
};
function plotA(tempDataA) {
    var lastTenNumberA = tempDataA;
    var tempTrace = {
        y: lastTenNumberA,
        x: [0, 1, 2, 3, 4, 5, 6],
        type: 'scatter',
        line: {
                color: 'slategray',
                width: 2,
                shape: 'spline'},
        };
    var data = [tempTrace];
    Plotly.newPlot('last-ten-A', data, myPlotLayout, {displayModeBar: false});
};
function plotB(tempDataB) {
    var lastTenNumberB = tempDataB;
    var tempTrace = {
        y: lastTenNumberB,
        x: [0, 1, 2, 3, 4, 5, 6],
        type: 'scatter',
        line: {
                color: 'slategray',
                width: 2,
                shape: 'spline'},
        };
    var data = [tempTrace];
    Plotly.newPlot('last-ten-B', data, myPlotLayout,  {displayModeBar: false, scrollZoom: false});
};
```

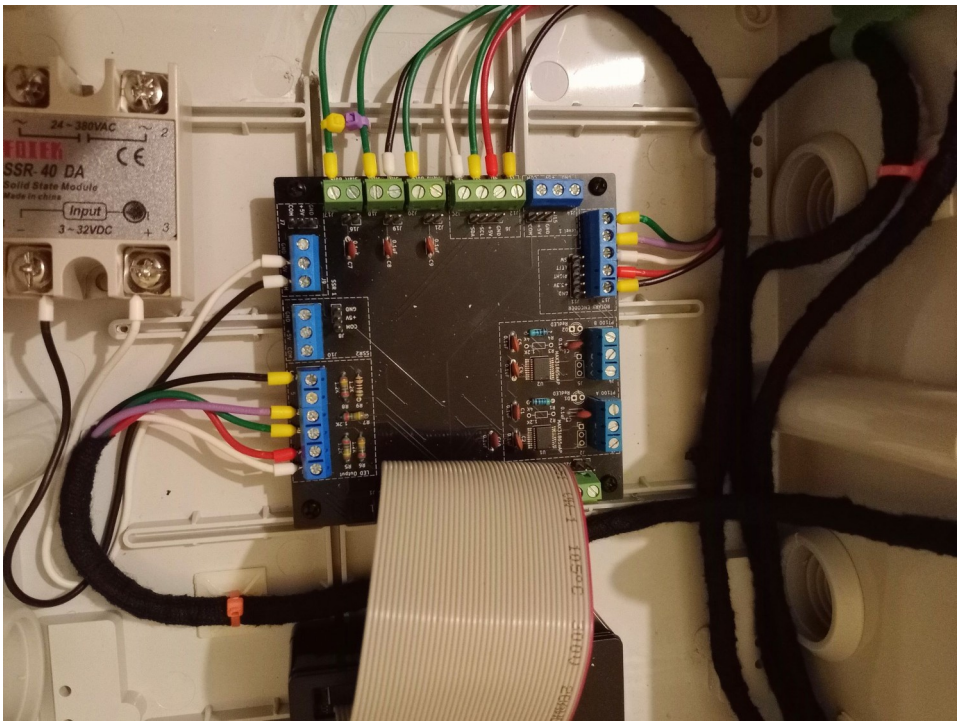
ΠΑΡΑΡΤΗΜΑ Ε: Φωτογραφίες της κατασκευής.



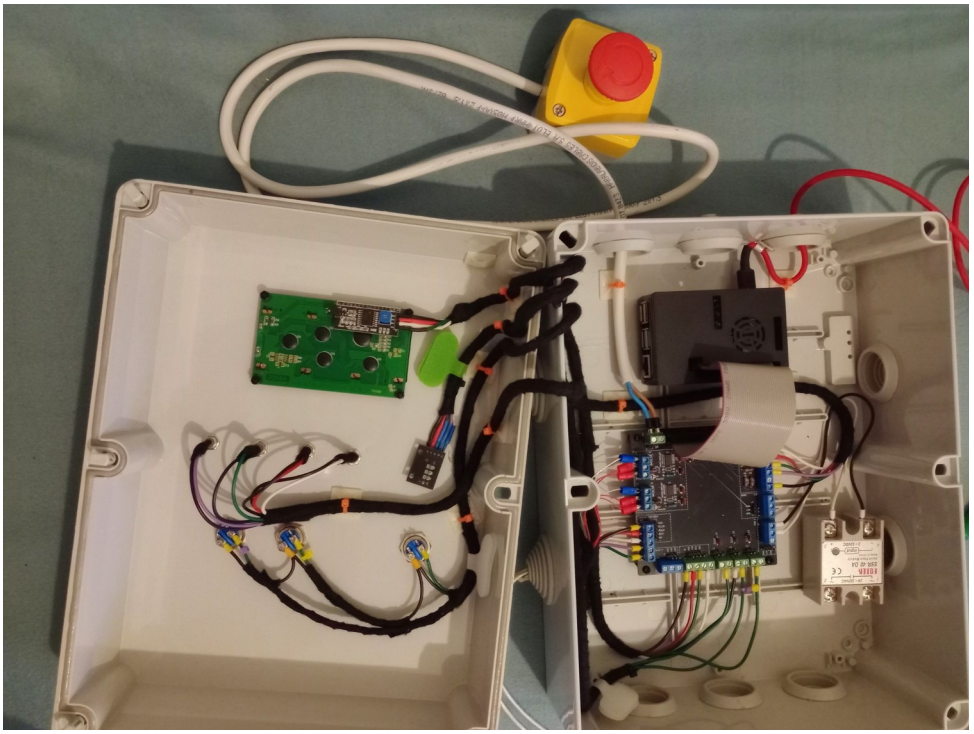
Εικόνα 37: Η οθόνη ενδείξεων.



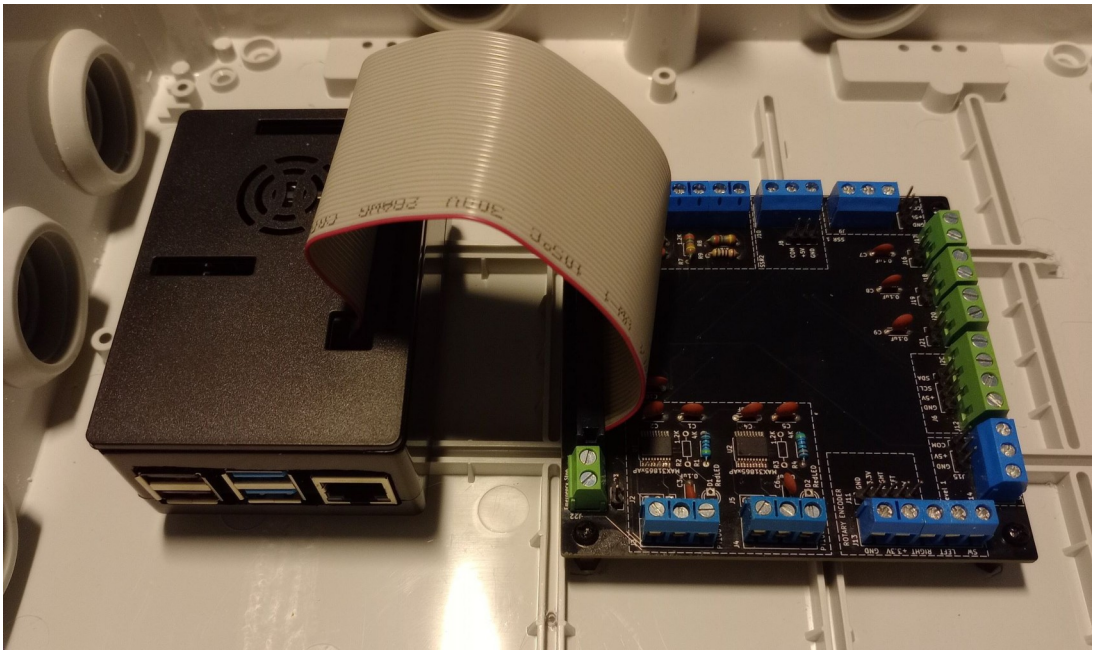
Εικόνα 38: Η πλακέτα επέκτασης



Εικόνα 39: Εσωτερικό πίνακα ελέγχου.

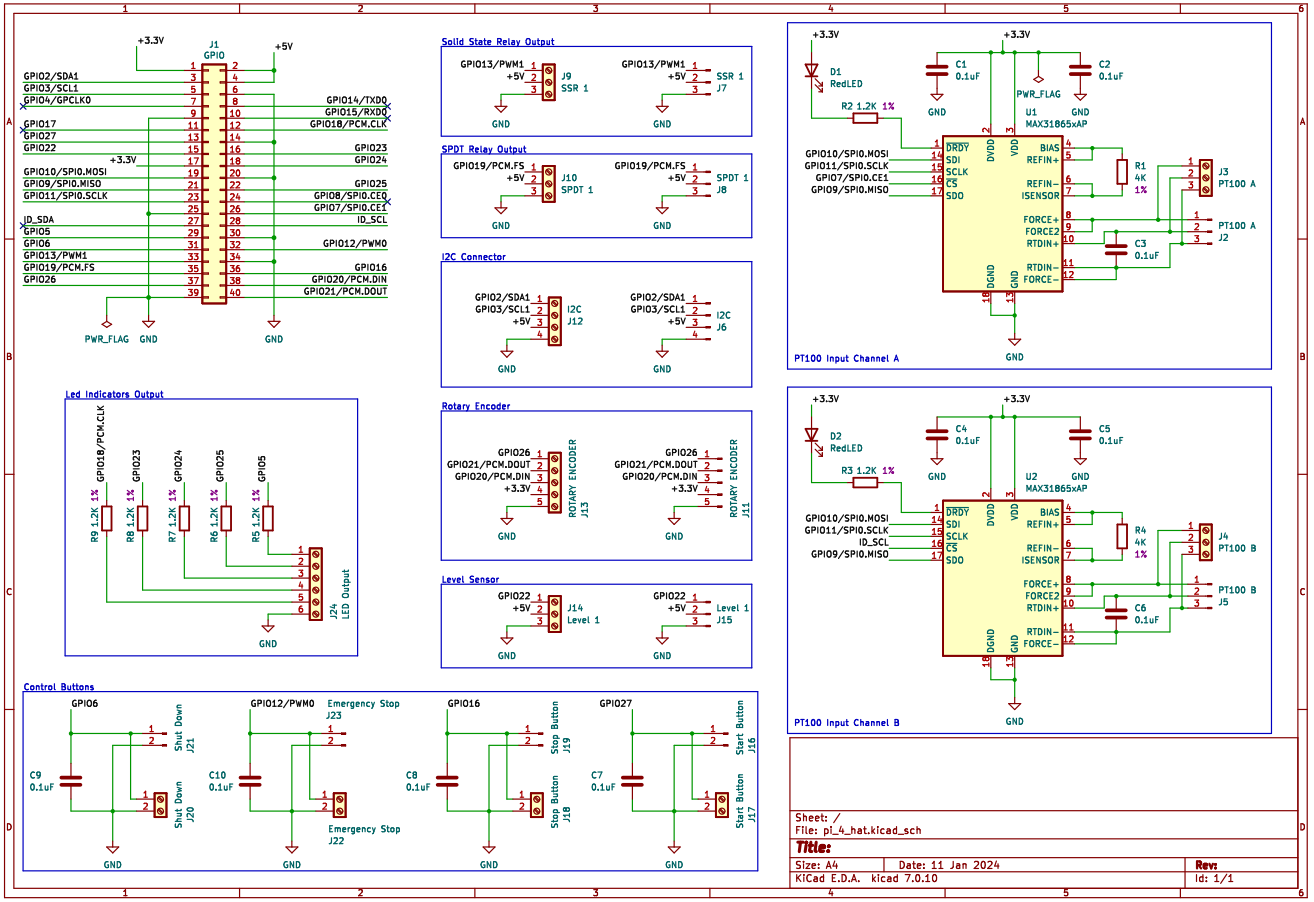


Εικόνα 40: Εσωτερικό πίνακα ελέγχου και το κόμβιο εκτάκτου ανάγκης.



Εικόνα 41: Το Raspberry Pi 4 και η πλακέτα επέκτασης.

ΠΑΡΑΡΤΗΜΑ ΣΤ: Τα σχέδια της πλακέτας επέκτασης.



Εικόνα 42: Το σχηματικό της πλακέτας επέκτασης.

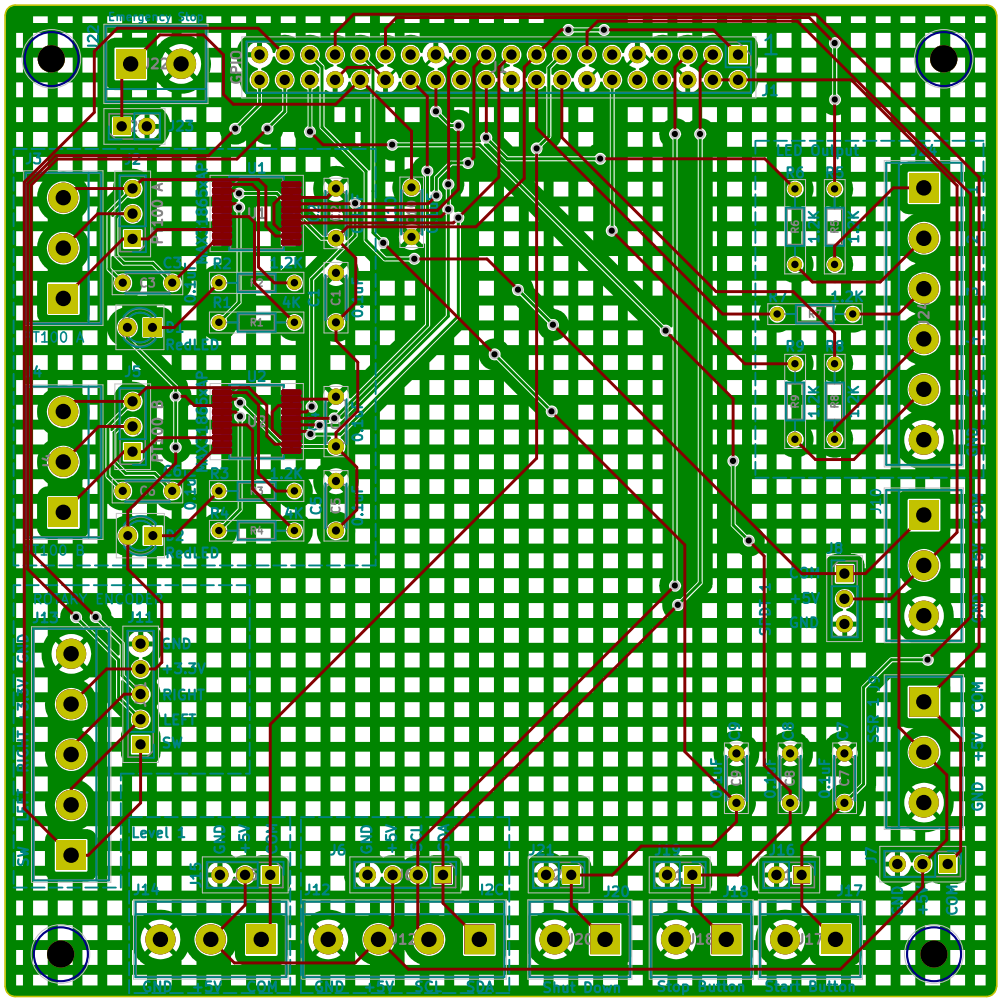


Figure 43: Το PCB της πλακέτας επέκτασης.