ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

Σύστημα ελέγχου περιβαλλοντικών παραμέτρων σε νοσοκομειακές δομές

Τσαβδαρίδης Κωνσταντίνος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων:
Κακαρούντας Αθανάσιος

Λαμία, 2021

UNIVERSITY OF THESSALY
SCHOOL OF SCIENCE
INFORMATICS AND COMPUTATIONAL BIOMEDICINE

Environmental parameters monitoring system, in health care facilities

Tsavdaridis Konstantinos

MASTER THESIS

Supervisor's name:
Kakarountas Athanasios

Lamia, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

Σύστημα ελέγχου περιβαλλοντικών παραμέτρων σε νοσοκομειακές δομές.

Τσαβδαρίδης Κωνσταντίνος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων:
Κακαρούντας Αθανάσιος

Λαμία, 2021

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο [«τίτλος εργασίας»] αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ:

Τσαβδαρίδης Κωνσταντίνος

Ημερομηνία:

Υπογραφή

Τριμελής Επιτροπή:

Κακαρούντας Αθανάσιος

Τασουλής Σωτήριος

Καρανίκας Χαράλαμπος

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας τέθηκε ο σχεδιασμός και η κατασκευή ενός συστήματος ελέγχου των περιβαλλοντικών συνθηκών σε κάποια νοσοκομειακή δομή. Τα κριτήρια που θεωρήθηκαν απαραίτητα είναι η δυνατότητα μέτρησης της θερμοκρασίας, της υγρασίας και της συγκέντρωσης μονοξειδίου του άνθρακα του περιβάλλοντος, καθώς και η δυνατότητα καταγραφής αυτών, για το σκοπό του ελέγχου των εν λόγω των συνθηκών, όπου αυτές θεωρούνται κρίσιμες (πχ εντός κάποιας νοσοκομειακής δομής ή εργαστηρίου). Σημαντικό ρόλο στον σχεδιασμό παίζει το κόστος της κατασκευής, η διαθεσιμότητα των απαραίτητων εξαρτημάτων, αλλά και η ευκολία παραμετροποίησης των μετρήσεων που το σύστημα θα μπορεί να συλλέξει.

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος διαγραμμάτων

Κατάλογος εικόνων

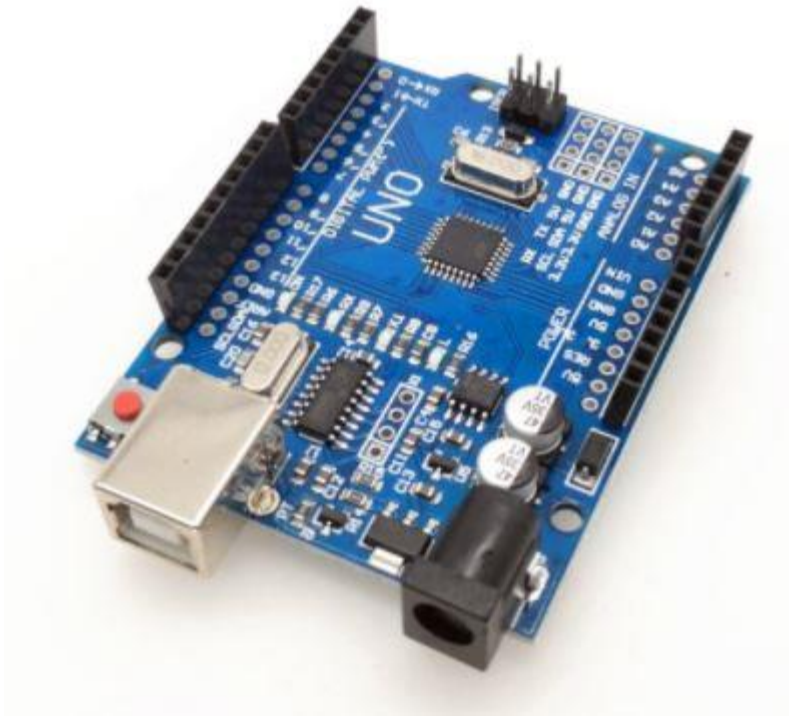
Κεφάλαιο 1: Θεωρητικός σχεδιασμός

Κατά τον σχεδιασμό της εκτέλεσης της διπλωματικής εργασίας, τέθηκαν τα βασικά χαρακτηριστικά του υπό κατασκευή συστήματος. Στόχος ήταν η δημιουργία ενός συστήματος, το οποίο δύναται να παρέχει ακριβείς μετρήσεις των δεδομένων που θέλουμε να μελετήσουμε, τα οποία είναι η θερμοκρασία, η υγρασία και η ποσότητα μονοξειδίου του άνθρακα σε κάποιο συγκεκριμένο χώρο μια νοσοκομειακής δομής ή εργαστηρίου. Βασική παράμετρος στην επιλογή της πλατφόρμας που χρησιμοποιήθηκε είναι το κόστος αυτής. Με πληθώρα συσκευών μετρήσεων στην αγορά κρίθηκε αναγκαίο το κόστος να διατηρηθεί όσο το δυνατό χαμηλότερα. Άλλη μία παράμετρος ιδιαίτερης βαρύτητας είναι η δυνατότητα παραμετροποίησης του συστήματος από τον τελικό χρήστη, καθώς τα διάφορα εμπορικά συστήματα δεν δίνουν πλήρη ελευθερία σε αυτόν τον τομέα. Με βάσει τα παραπάνω και για λόγους που θα αναλυθούν στη συνέχεια, επιλέχθηκε η πλατφόρμα Arduino σαν καταλληλότερη βάση για την υλοποίηση του συστήματος. Η Arduino είναι μια ανοιχτού λογισμικού (open-source) πλατφόρμα δημιουργίας ηλεκτρονικών συστημάτων που συνδυάζει το υλισμικό (hardware) με το λογισμικό (software). Όντας μία τέτοια πλατφόρμα (ανοιχτού λογισμικού) δεν διέπεται από κοστοβόρες άδειες χρήσης και ειδικά (ιδιόκτητα και πατενταρισμένα) εργαλεία κατασκευής, το οποίο δίνει σημαντικό οικονομικό πλεονέκτημα έναντι ανταγωνιστικών πλατφορμών (πχ Raspberry Pi). Μερικές σημαντικές διαφορές των δύο είναι:

- Το Arduino είναι μια πλακέτα μικροελεγκτή (microcontroller board), ενώ το Raspberry Pi ένας μικρός υπολογιστής βασισμένος σε μικροεπεξεργαστή (microprocessor).
- Ο μικροελεγκτής του Arduino περιέχει τον επεξεργαστή και δύο τύπους μνήμης (RAM και ROM). Όλο το επιπλέον υλισμικό της πλακέτας είναι για την τροφοδοσία ρεύματος, τον προγραμματισμό και τη δημιουργία συνδεσιμότητας (IO connectivity) μέσω θυρών εισόδου/εξόδου. Αντιθέτως, το Raspberry Pi έχει όλα τα στοιχεία ενός υπολογιστή με επεξεργαστή, μνήμη, αποθηκευτικό χώρο, κάρτα γραφικών και έτοιμες συνδέσεις στην πλακέτα του.

- Το Raspberry Pi απαιτεί πλήρες λειτουργικό σύστημα, συνήθως κάποια ειδική διανομή Linux (Raspberry Pi OS), η οποία εξελίσσεται από το Raspberry Pi Foundation, ενώ το Arduino χρειάζεται μόνο τον κώδικα εκτέλεσης των προγραμματισμένων λειτουργιών.
- Το Raspberry Pi υπερτερεί στην ανάπτυξη εφαρμογών λογισμικού με τη χρήση γλώσσας προγραμματισμού Python, ενώ το Arduino στην διεπαφή (interface) διαφόρων αισθητήρων, φωτισμού και κινητήρων.
- Επίσης, οι ανάγκες τροφοδοσίας των δύο συστημάτων διαφέρουν. Αν και τα δύο συστήματα έχουν είσοδο τροφοδοσίας μέσω θύρας USB, το Raspberry Pi χρειάζεται αρκετά μεγαλύτερη ένταση ρεύματος σε σχέση με το Arduino, απαιτώντας συνήθως την παρουσία εξωτερικού τροφοδοτικού, σε αντίθεση με το Arduino το οποίο μπορεί να λειτουργήσει με την τροφοδοσία ρεύματος μέσω μιας απλής θύρας USB ενός υπολογιστή. Η ακούσια διακοπή της τροφοδοσίας ενός Raspberry Pi μπορεί να προκαλέσει ζημιά στο υλικό (hardware), στο λογισμικό (software) ή σε κάποια εφαρμογή. Το Raspberry Pi πρέπει να τερματιστεί σωστά (shutdown) πριν αφαιρεθεί η τροφοδοσία ρεύματος, όπως κάθε υπολογιστής, ενώ το Arduino σε περίπτωση διακοπής της τροφοδοσίας, απλά επανεκκινεί και εκτελεί τον αποθηκευμένο κώδικα.
- Το κόστος απόκτησης του πρώτου Arduino UNO ήταν $23, αλλά υπάρχουν διαθέσιμοι αρκετοί κλώνοι αυτού με κόστος λιγότερο από $4, χωρίς ουσιαστικές διαφορές. Το αρχικό Raspberry Pi ξεκίνησε με κόστος $35 με τα επόμενα μοντέλα αυτού να ξεκινούν σε διαφορετικές τιμές ($55 ή $75 για το Raspberry Pi 4 Model B), αναλόγως και της ποσότητας εγκατεστημένης μνήμης.

Με βάση τα παραπάνω λοιπόν, αποφασίσθηκε η χρήση της πλατφόρμας Arduino για την υλοποίηση της κατασκευής. Συγκεκριμένα, επιλέχθηκε ο κλώνος μικροελεγκτή Arduino UNO (κόστος €2,60) σαν βάση της κατασκευής:

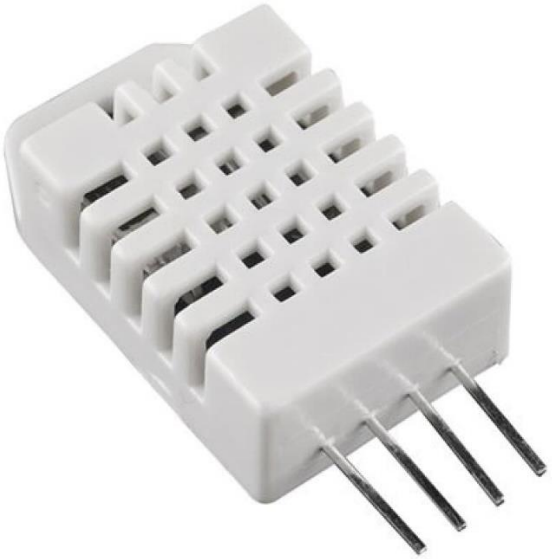


Εικόνα 1.1: Arduino UNO

Επιπροσθέτως, χρησιμοποιήθηκαν οι εξής αισθητήρες:

- Αισθητήρας DHT-22 (κόστος €4,40):

Πρόκειται για ένα ψηφιακό αισθητήρα μέτρησης θερμοκρασίας και υγρασίας με ευαισθησία της τάξεως του 0,1 βαθμών. Έχει εύρος μέτρησης θερμοκρασίας από -0^{o}C έως 80^{o}C και σχετικής υγρασίας (RH) 0% έως 100%. Ο αισθητήρας αυτός προτιμήθηκε, διότι είναι ψηφιακός και βαθμονομημένος, κάτι που εγγυάται ακρίβεια της τάξεως των ±0.5°C στην μέτρηση της θερμοκρασίας και του ±2% κατά την μέτρηση της σχετικής υγρασίας.



Εικόνα 1.2: αισθητήρας DHT-22

- Αισθητήρας MQ-7 (κόστος €0,90):

Πρόκειται για έναν αναλογικό αισθητήρα ανίχνευσης μονοξειδίου του άνθρακα (CO). Ο αισθητήρας αυτός αποτελείται από δύο μέρη, η λειτουργία των οποίων είναι αλληλένδετη. Το ένα μέρος είναι ένα κύκλωμα θέρμανσης και το δεύτερο μέρος ένα κύκλωμα μέτρησης. Ο αισθητήρας αυτός έχει εύρος μέτρησης 10ppm CO έως 1000ppm CO και θεωρείται ακριβής όταν μετράει σε συνθήκες περιβάλλοντος από τους -20°C έως τους 50°C και σχετική υγρασία έως 95%.

Εικόνα 1.3: αισθητήρας MQ-7

Εκτός των απαραίτητων αισθητήρων αποφασίστηκε να αξιοποιηθεί η δυνατότητα προγραμματισμού του Arduino UNO για την προσθήκη οπτικών και ηχητικών ενδείξεων/προειδοποιήσεων. Οι οπτικές ενδείξεις αφορούν στη λειτουργία 2 LED, που δηλώνουν την κατάσταση λειτουργίας της συσκευής και το αν η τελευταία μέτρηση εμπίπτει εντός των καθορισμένων από εμάς ορίων, ενώ η ηχητική προειδοποίηση αφορά σε ένα ηχείο, το οποίο παράγει ένα σήμα κινδύνου όταν η τελευταία μέτρηση είναι εκτός των προκαθορισμένων από εμάς ορίων.

- LED για οπτικές ενδείξεις (κόστος: €0,10)

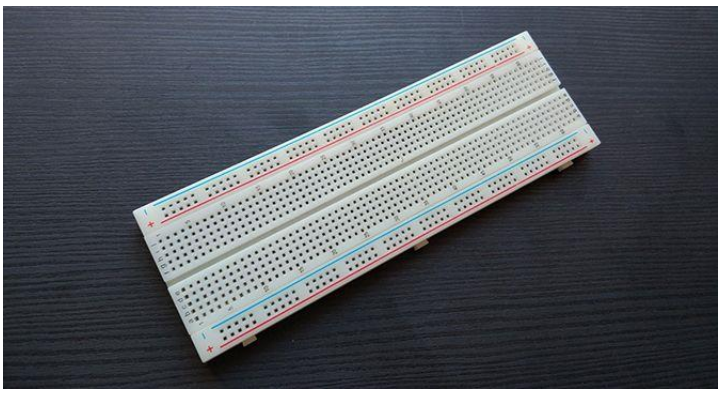
Εικόνα 1.4: LED οπτικών ενδείξεων

- Ηχείο για ακουστική προειδοποίηση (κόστος €0,30)

Εικόνα 1.5: ηχείο προειδοποίησης

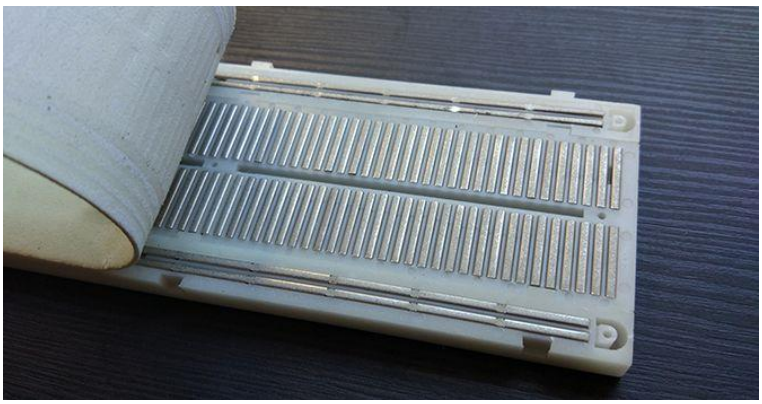
Κεφάλαιο 2: Υλοποίηση κατασκευής

Μετά την επιλογή των κατάλληλων υποσυστημάτων και σύμφωνα με τις τεχνικές οδηγίες συναρμολόγησης αυτών, προχωρήσαμε στην κατασκευή του συστήματος μέτρησης. Για πρακτικούς αλλά και για λόγους ευκολίας, κατά τις δοκιμές επίδειξης αποφασίστηκε κατασκευή ανοιχτού τύπου επάνω σε breadboard. Το breadboard είναι μια απλή κατασκευή, η οποία μας επιτρέπει να δημιουργούμε κυκλώματα χωρίς την ανάγκη συγκολλήσεων. Το κόστος του, αναλόγως και του μεγέθους αυτού, είναι περίπου €2,00



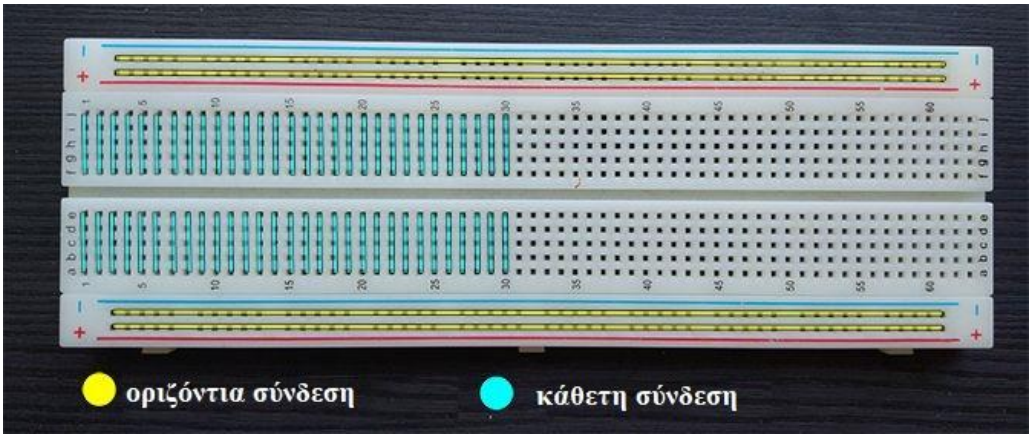
Εικόνα 2.1: breadboard

Η αρχή λειτουργίας του breadboard είναι πολύ απλή, όπως μπορούμε να δούμε στην εικόνα 2.2 και ουσιαστικά αποτελείται από δύο διαφορετικά σετ καλωδίων.



Εικόνα 2.2: breadboard, κάτω όψη

Το πρώτο σετ είναι τοποθετημένο οριζόντια και αφορά στην παροχή τροφοδοσίας ρεύματος, ενώ το δεύτερο σετ είναι τοποθετημένο κάθετα και παρέχει τις απαραίτητες ενώσεις των διαφόρων προς κατασκευή κυκλωμάτων (εικόνα 2.3).



Εικόνα 2.3: breadboard, αρχή λειτουργίας.

Τέλος, απαιτήθηκαν διάφορα καλώδια για την επικοινωνία των αισθητήρων με το Arduino, μερικές αντιστάσεις, δίοδοι και πυκνωτές επιπλέον κόστους €3,00. Το συνολικό κόστος όλων των εξαρτημάτων που έχουν αναφερθεί, ανέρχεται περίπου στα €14,00.

Πριν προχωρήσουμε στη συναρμολόγηση του συστήματος, θα πρέπει να αναφέρουμε ότι βάσει των τεχνικών προδιαγραφών και του τρόπου λειτουργίας του αισθητήρα MQ-7, απαιτείται μια τροποποίηση αυτού. Συγκεκριμένα, ο κατασκευαστής του περιγράφει ότι ο αισθητήρας πρέπει να λειτουργεί σε δύο φάσεις, με σκοπό να παράγει χρήσιμα αποτελέσματα. Οι δύο αυτές φάσεις αφορούν στη θερμική κατάσταση του αισθητήρα, όπου κατά την «ψυχρή» φάση το CO συσσωρεύεται στο στοιχείο μέτρησής του και κατά τη «θερμή» φάση το CO αποβάλλεται, ουσιαστικά καθαρίζοντας τον αισθητήρα για την επόμενη αυτή της μέτρησης. Η αρχή λειτουργίας σε απλοποιημένη μορφή, μπορεί να αποδοθεί στα παρακάτω βήματα και θα αποτελέσει την βάση σχεδιασμού του προγράμματός ελέγχου της συσκευής:

1. Εφαρμογή τάσης 5V για 60 δευτερόλεπτα – δεν υπολογίζουμε αυτές τις μετρήσεις
2. Εφαρμογή τάσης 1,4V για 90 δευτερόλεπτα – οι μετρήσεις αυτές μας ενδιαφέρουν
3. Επιστροφή στο βήμα 1.

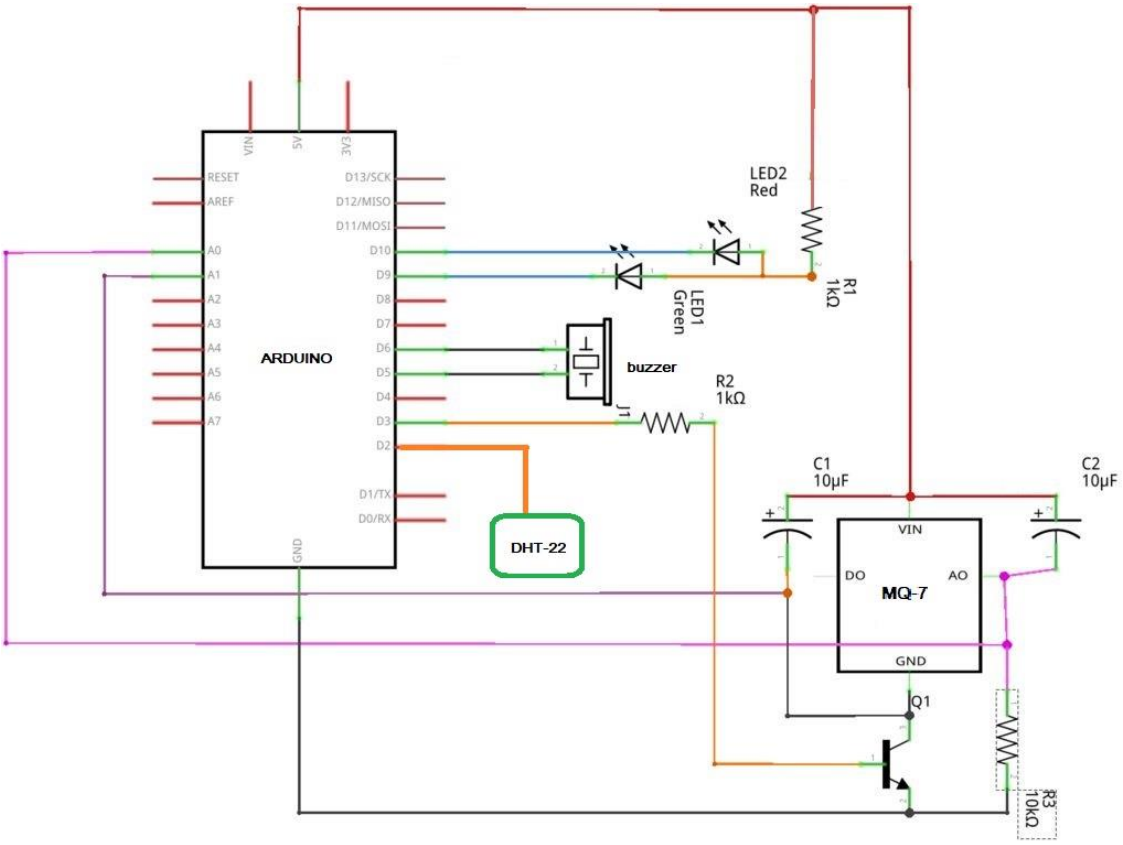
Οι παραπάνω απαιτήσεις για τη σωστή λειτουργία του αισθητήρα προσκρούουν σε 2 περιορισμούς. Ο πρώτος είναι, ότι ο αισθητήρας κατά τη «θερμή» του φάση απαιτεί ένταση ρεύματος περίπου 150mA, ενώ το Arduino μπορεί να αποδώσει μόνο 40mA με σχετική ασφάλεια. Μεγαλύτερη τάση ρεύματος θα έχει ως συνέπεια να καταστραφεί το συγκεκριμένο pin του Arduino που είναι υπεύθυνο για την τροφοδοσία ρεύματος του αισθητήρα MQ-7. Ο δεύτερος περιορισμός είναι πως κατά την «ψυχρή» φάση του αισθητήρα, αυτός απαιτεί τροφοδοσία τάσης 1,4V, κάτι που εγγενώς δεν μπορεί να παρέχει το Arduino. Ο μόνος τρόπος για να πάρουμε αξιόπιστα αυτή την τάση, χωρίς να εισάγουμε μεγάλο πλήθος αναλογικών εξαρτημάτων στην κατασκευή, είναι να χρησιμοποιήσουμε την τεχνική του PWM (pulse width modulation – διαμόρφωση πλάτους παλμού).

Οι δύο αυτοί περιορισμοί ξεπερνιόνται με τη χρήση ενός NPN transistor. Όταν το transistor είναι ενεργό, η τάση ρεύματος που καταλήγει στον αισθητήρα MQ-7 είναι σταθερή στα 5V, όπως επιβάλλεται για τη «θερμή» του φάση. Χρησιμοποιώντας την τεχνική του PWM, η ένταση του ρεύματος αυξομειώνεται, στη συνέχεια εξισορροπείται από τον πυκνωτή και διατηρείται επίσης σταθερή για την απρόσκοπτη τροφοδοσία του αισθητήρα. Χρησιμοποιώντας PWM υψηλής συχνότητας (62.5KHz) και λαμβάνοντας το μέσο όρο πολλών αναλογικών μετρήσεων (περισσότερων από 1000), τότε το αποτέλεσμα της μέτρησης μπορεί να θεωρηθεί αξιόπιστο και κατά συνέπεια αξιοποιήσιμο.

Ολοκληρώνοντας τον σχεδιασμό της συσκευής, αξίζει να αναφέρουμε τα παρακάτω:

- Τα pin D9 και D10 θα χρησιμοποιηθούν για τις φωτεινές ενδείξεις των LED.
- Τα pin D5 και D6 θα χρησιμοποιηθούν για το ηχείο προειδοποίησης.
- Το pin D3 εξυπηρετεί το PWM του τρανζίστορ το οποίο ελέγχει την τάση στον αισθητήρα MQ-7.
- Η αντίσταση R1 χρησιμοποιείται για τον έλεγχο φωτεινότητας των LED.
- Η αντίσταση R2 εξυπηρετεί τον περιορισμό της έντασης ρεύματος του τρανζίστορ, ώστε να μην υπερφορτωθεί το αντίστοιχο pin του Arduino.
- Η αντίσταση R3 τοποθετείται παράλληλα με την πλάκα μέτρησης του αισθητήρα, ώστε να δημιουργήσει μία διαφορά τάσης.
- Η τάση στην έξοδο του αισθητήρα ισούται με R3 / (R3 + Rs) * 5V, όπου Rs είναι η αντίσταση του αισθητήρα MQ-7 που χρησιμοποιούμε.

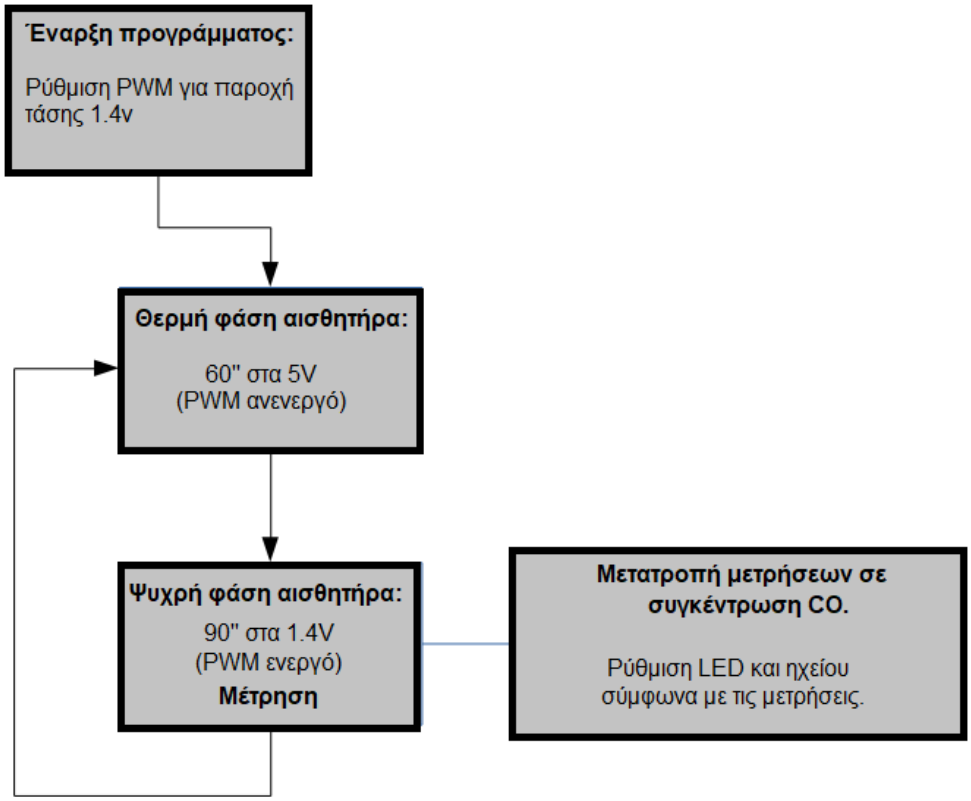
- Η αντίσταση αυτή εξαρτάται από την συγκέντρωση CO, οπότε η τάση μεταβάλλεται αντίστοιχα.
- Ο πυκνωτής C1 χρησιμοποιείται για να εξισορροπήσει την τάση εισόδου PWM στον αισθητήρα MQ-7.
- Ο πυκνωτής C2 χρησιμοποιείται για να εξομαλύνει την αναλογική έξοδο του αισθητήρα (η τάση εξόδου εξαρτάται από την τάση εισόδου και έχοντας αρκετά υψηλό PWM, αυτό επηρεάζει τις μετρήσεις), οπότε ο πυκνωτής C2 απαιτεί να είναι ίδιας χωρητικότητας με τον C1.
- Το NPN τρανζίστορ είτε παρέχει συνεχή ένταση ρεύματος για τη «θερμή» φάση του αισθητήρα, είτε λειτουργεί σε PWM, ελαττώνοντας την ένταση ρεύματος που καταλήγει στον αισθητήρα, για την «ψυχρή» φάση αυτού.
- Τέλος, ο αισθητήρας θερμοκρασίας/υγρασίας DHT-22, όντας ψηφιακός, δεν απαιτεί επιπλέον τροποποιήσεις και η έξοδός του συνδέεται στο pin D2 του Arduino, ενώ λαμβάνει τροφοδοσία ρεύματος απευθείας από το breadboard.



Εικόνα 2.4: τροποποίηση και σύνδεση αισθητήρα CO (MQ-7)

Κεφάλαιο 3: Προγραμματισμός συσκευής

Έχοντας πλέον ολοκληρώσει την κατασκευή της συσκευής, είμαστε έτοιμοι να σχεδιάσουμε και να ολοκληρώσουμε τον προγραμματισμό της συσκευής μας. Η αρχή λειτουργίας του προγράμματός μας μπορεί να αποδοθεί στο παρακάτω σχεδιάγραμμα:



Εικόνα 3.1: Αρχή λειτουργίας του προγράμματος

Αρχικά οφείλουμε να ρυθμίσουμε την PWM λειτουργία της συσκευής μας, ώστε να λάβουμε την (απαραίτητη για τον αισθητήρα MQ-7) τάση λειτουργίας 1.4V. Στη συνέχεια, η συσκευή εκτελεί συνεχείς εναλλαγές των δύο φάσεων λειτουργίας του αισθητήρα (θερμή/ψυχρή φάση), όπως αυτές απαιτούνται από τον κατασκευαστή του. Έχουμε επομένως τη θερμή φάση, όπου ο αισθητήρας λαμβάνει 5V τάσης ρεύματος για 60", ακολουθουμένη από την ψυχρή φάση, όπου ο αισθητήρας λαμβάνει 1.4V μέσω της PWM λειτουργίας για 90". Σε αυτή τη φάση, το CO που πιθανώς υπάρχει στην ατμόσφαιρα, εναποτίθεται επάνω στο στοιχείο μέτρησης του αισθητήρα και αυτός

παράγει μία τιμή την οποία μπορούμε να μετατρέψουμε σε συγκέντρωση CO στην ατμόσφαιρα. Για να πραγματοποιηθεί αυτό, χρειαζόμαστε κάποιο χρονόμετρο, κάτι που εγγενώς το Arduino δεν παρέχει, οπότε αυτό πρέπει να υλοποιηθεί προγραμματιστικά. Στο πρόγραμμά μας εκτελούνται 3 λειτουργίες χρονομέτρου, τις οποίες έχουμε ονομάσει **setTimer0PWM**, **setTimer1PWM** και **setTimer2PWM**. Το καθένα από αυτά χρονομετρεί και εφαρμόζει τις ρυθμίσεις του PWM, σύμφωνα με τις απαιτήσεις μας. Ο έλεγχος των δύο φάσεων λειτουργίας του αισθητήρας MQ-7 εκτελείται με την χρήση των λειτουργιών **startMeasurementPhas**e και **startHeatingPhase**, οι οποίες ορίζουν τη διάρκεια των χρονομέτρων για την εναλλαγή των PWM φάσεων μεταξύ 5V και 1,4V. Η κατάσταση των LED ελέγχεται από τη λειτουργία **setLEDs**, η οποία δέχεται είσοδο φωτεινότητας σε ποσοστό 1-100 για κόκκινο ή πράσινο φωτισμό. Η κατάσταση του ηχείου ελέγχεται από τις λειτουργίες **buzz_on, buzz_off** και **buzz_beep**. Οι λειτουργίες on/off διαχειρίζονται τον ήχο, ενώ η λειτουργία beep παράγει συγκεκριμένη αλληλουχία προειδοποιητικού ήχου με διάρκεια 1,5", η οποία ενεργοποιείται σύμφωνα με τα όρια που έχουμε θέσει για τη μέτρηση CO. Το πρόγραμμά μας αρχικά εκτελεί τη λειτουργία **pwm_adjust**, η οποία βρίσκει το κατάλληλο εύρος PWM, ώστε να επιτευχθεί η απαραίτητη τάση 1,4V για τον φάση της μέτρησης. Μόλις γίνει αυτό, παράγεται ένας ήχος που υποδεικνύει ότι ο αισθητήρας είναι έτοιμος και στη συνέχεια ξεκινάει η εκτέλεση και η εναλλαγή των δύο επαναλαμβανόμενων φάσεων (loop) του προγράμματος. Κατά την εκτέλεση του προγράμματος ελέγχεται από αυτό συνεχώς, αν έχουμε παραμείνει αρκετό χρόνο σε κάθε μία από τις δύο φάσεις (90" για τη ψυχρή φάση/φάση μέτρησης και 60" για την θερμή φάση) και αν αυτό ισχύει τότε η φάση εναλλάσσεται. Επιπροσθέτως, το πρόγραμμα ενημερώνεται συνεχώς για τις μετρήσεις του αισθητήρα και εφαρμόζει εκθετική εξομάλυνση με βάση τον τύπο: $\mathbf{T = 0{,}999 * T_{v-1} + 0{,}001 * T_v}$, όπου T η νέα μέτρηση, T_{v-1} η μέτρηση στον προηγούμενο κύκλο και T_v η στιγμιαία μέτρηση, στον τρέχων κύκλο. Με βάσει αυτές τις παραμέτρους και τους χρόνους των δύο κύκλων μέτρησης, παίρνουμε μέτρηση που αντιστοιχεί στα τελευταία 300milliseconds.

Ακολουθεί ο κώδικας του προγράμματος, με τα σχόλιά του:

```
//Γενική προειδοποίηση: κάθε αισθητήρας είναι διαφορετικός
//Κάθε αισθητήρας ΠΡΕΠΕΙ να βαθμονομηθεί και να καθοριστεί η τιμή του
sensor_reading_clean_air πριν από κάθε χρήση.

#include <DHT.h> //αυτό συμπεριλαμβάνει την βιβλιοθήκη για τον
αισθητήρα DHT-22

// Σταθερές
#define DHTPIN 2     //σε ποιο pin του Arduino έχουμε συνδεθεί
#define DHTTYPE DHT22   // ρισμός του τύπου αισθητήρα: DHT 22
(AM2302)
DHT dht(DHTPIN, DHTTYPE); ////αρχικοποίηση του αισθητήρα DHT για χρήση
με 16mhz Arduino


int time_scale = 8; //time scale: μεταβάλουμε το χρονόμετρο του
συστήματος, έτσι ώστε όλες οι λειτουργίες όπως millis(), delay() κλπ
//νομίζουν ότι ο χρόνος κινείται 8 φορές πιο γρήγορα από το κανονικό
//για να διορθώσουμε αυτό χρησιμοποιούμε τη μεταβλητή time_scale
variable:
//για να εισάγουμε delay 1000ms ζητούμε: delay(1000*time_scale)

int chk;
float hum;  //αποθηκεύει την τιμή της υγρασίας
float temp; //αποθηκεύει την τιμή της θερμοκρασίας

void setTimer0PWM(byte chA, byte chB) // pin D5 και pin D6
{
  TCCR0A = 0b10110011; //OCA κανονικό,OCB ανεστραμμένο, γρήγορο pwm
  TCCR0B = 0b010; //8 prescaler – σε αντίθεση με τον κανονικό
prescaler 64, έτσι ο χρόνος κινείται 8 φορές ταχύτερα
  OCR0A = chA; //0..255
  OCR0B = chB;
}


void setTimer2PWM(byte chA, byte chB) //pin D11 και D3
{
  TCCR2A = 0b10100011; //OCA,OCB, γρήγορο pwm
  TCCR2B = 0b001; //χωρίς prescaler
  OCR2A = chA; //0..255
  OCR2B = chB;
}

void setTimer1PWM(int chA, int chB) //pin D9 και D10
{
  TCCR1A = 0b10100011; //OCA,OCB, γρήγορο pwm
  TCCR1B = 0b1001; //χωρίς prescaler
  OCR1A = chA; //0..1023
  OCR1B = chB;
}

float opt_voltage = 0;
byte opt_width = 240; //default reasonable value

void pwm_adjust()
```

```
//αυτή η λειτουργία δοκιμάζει διάφορες συχνότητες PWM και μας δείχνει
//την τάση που επιτεύχθηκε, στη συνέχει επιλέγει την πιο σωστή και
//χρησιμοποιεί αυτή για την διάρκεια εκτέλεσης του κώδικα
{
  float previous_v = 5.0; //τάση στην προηγούμενη προσπάθεια
  float raw2v = 5.0 / 1024.0;// //σταθερά που μετατρέπει το analogRead
του Arduino σε τάση σε volts
  for(int w = 0; w < 250; w++)
  {
    setTimer2PWM(0, w);
    float avg_v = 0;
    for(int x = 0; x < 100; x ++) //μέτρηση για 100ms ώστε να έχουμε
σταθερό αποτέλεσμα
    {
      avg_v += analogRead(A1);
      delay(time_scale);
    }
    avg_v *= 0.01;
    avg_v *= raw2v;
    Serial.print("adjusting PWM w=");
    Serial.print(w);
    Serial.print(", V=");
    Serial.println(avg_v);
    if(avg_v < 3.6 && previous_v > 3.6) //εύρεση βέλτιστου πλάτους PWM
    {
      float dnew = 3.6 - avg_v; //τώρα πρέπει να βρούμε αν το τρέχον
      float dprev = previous_v - 3.6;//ή το προηγούμενο πλάτος είναι
σωστότερο
      if(dnew < dprev) //αν το νέο πλάτος είναι πιο κοντά στο 1.4v
τότε επιλέγεται
      {
        opt_voltage = avg_v;
        opt_width = w;
        return;
      }
      else //αλλιώς επιστρέφουμε στο προηγούμενο πλάτος PWM
      {
        opt_voltage = previous_v;
        opt_width = w-1;
        return;
      }
    }
    previous_v = avg_v;
  }
}

float alarm_ppm_threshold = 100; //όριο συγκέντροωσης CO για την
παραγωγή ηχητικού σήματος
float red_threshold = 400; //όριο εναλλαγής από πράσινο σε κόκκινο LED
float reference_resistor_kOhm = 10.0; //ορισμός τιμής αντίστασης που
χρησιμοποιήθηκε

float sensor_reading_clean_air = 600; // εισαγωγή της raw μέτρησης του
αισθητήρα στο τέλος της ψυχρής φάσης (πριν αρχίσει η θερμή φάση) σε
καθαρό αέρα. Αυτό είναι κρίσιμο για το σωστό υπολογισμό
float sensor_reading_100_ppm_CO = -1; // //αν μπορεί να γίνει
καλιμπράρισμα της μέτρησης για ορισμένο δείγμα CO 100ppm, τότε
εισάγουμε την raw τιμή του αισθητήρα μας εδώ
//εναλλακτικά το αφήνουμε -1 και χρησιμοποιείται η default τιμή που
ορίσαμε
```

```
float sensor_100ppm_CO_resistance_kOhm; //υπολογισμένο από τη
μεταβλητή sensor_reading_100_ppm_CO
float sensor_base_resistance_kOhm; // υπολογισμένο από τη μεταβλητή
sensor_reading_clean_air

byte phase = 0; //1 – υψηλή τάση, 0 – χαμηλή τάση μέτρησης
unsigned long prev_ms = 0; //milliseconds προηγούμενης φάσης
unsigned long sec10 = 0; // αυτό το χρονόμετρο ανανεώνεται 10 φορές /
sec
//όταν εξαντληθεί το πρόγραμμα ενδέχεται να σταματήσει να λειτουργεί
//κάτι τέτοιο μπορεί να συμβεί μετά από ~13 χρόνια λειτουργίας, παρόλα
αυτά
//πρέπει να ληφθεί υπόψιν
unsigned long high_on = 0; //χρόνος έναρξης φάσης υψηλής θερμοκρασίας
unsigned long low_on = 0; // χρόνος έναρξης φάσης χαμηλής θερμοκρασίας
unsigned long last_print = 0; //χρόνος από τον οποίον παράγαμε μέτρηση
στην σειριακή έξοδο

float sens_val = 0; //τρέχουσα εξομαλυμένη τιμή αισθητήρα
float last_CO_ppm_measurement = 0; //συγκέντρωση CO στο τέλος της
προηγούμενης φάσης μέτρησης

float raw_value_to_CO_ppm(float value)
{
  if(value < 1) return -1; //μήνυμα λάθους τιμής
  sensor_base_resistance_kOhm = reference_resistor_kOhm * 1023 /
sensor_reading_clean_air - reference_resistor_kOhm;
  if(sensor_reading_100_ppm_CO > 0)
  {
    sensor_100ppm_CO_resistance_kOhm = reference_resistor_kOhm * 1023
/ sensor_reading_100_ppm_CO - reference_resistor_kOhm;
  }
  else
  {
    sensor_100ppm_CO_resistance_kOhm = sensor_base_resistance_kOhm *
0.5;

  }
  float sensor_R_kOhm = reference_resistor_kOhm * 1023 / value -
reference_resistor_kOhm;
  float R_relation = sensor_100ppm_CO_resistance_kOhm / sensor_R_kOhm;
  float CO_ppm = 100 * (exp(R_relation) - 1.648);
  if(CO_ppm < 0) CO_ppm = 0;
  return CO_ppm;
}

void startMeasurementPhase()
{
  phase = 0;
  low_on = sec10;
  setTimer2PWM(0, opt_width);
}

void startHeatingPhase()
{
  phase = 1;
  high_on = sec10;
  setTimer2PWM(0, 255);
}
void setLEDs(int br_green, int br_red)
{
```

```
  if(br_red < 0) br_red = 0;
  if(br_red > 100) br_red = 100;
  if(br_green < 0) br_green = 0;
  if(br_green > 100) br_green = 100;

  float br = br_red;
  br *= 0.01;
  br = (exp(br)-1) / 1.7183 * 1023.0;
  float bg = br_green;
  bg *= 0.01;
  bg = (exp(bg)-1) / 1.7183 * 1023.0;
  if(br < 0) br = 0;
  if(br > 1023) br = 1023;
  if(bg < 0) bg = 0;
  if(bg > 1023) bg = 1023;

  setTimer1PWM(1023-br, 1023-bg);
}
void buzz_on()
{
  setTimer0PWM(128, 128);
}
void buzz_off()
{
  setTimer0PWM(255, 255);
}
void buzz_beep()
{
  byte sp = sec10%15;
  if(sp == 0)
    buzz_on();
  if(sp == 1)
    buzz_off();
  if(sp == 2)
    buzz_on();
  if(sp == 3)
    buzz_off();
  if(sp == 4)
    buzz_on();
  if(sp == 5)
    buzz_off();
}

void setup() {



  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  setTimer1PWM(1023, 0);
  analogReference(DEFAULT);
  Serial.begin(9600);

  dht.begin();

  pwm_adjust();
```

```
  Serial.print("PWM result: width ");
  Serial.print(opt_width);
  Serial.print(", voltage ");
  Serial.println(opt_voltage);
  Serial.println("Data output: raw A0 value, heating on/off (0.1 off
1000.1 on), CO ppm from last measurement cycle");
  //beep buzzer in the beginning to indicate that it works
  buzz_on();
  delay(100*time_scale);
  buzz_off();
  delay(100*time_scale);
  buzz_on();
  delay(100*time_scale);
  buzz_off();
  delay(100*time_scale);
  buzz_on();
  delay(100*time_scale);
  buzz_off();
  delay(100*time_scale);

  startMeasurementPhase(); //επανέναρξη μέτρησης
}

void loop()
{

  unsigned long ms = millis();
  int dt = ms - prev_ms;

  if(dt > 100*time_scale || dt < 0)
  {
    prev_ms = ms; //αποθηκεύει την τιμή του προηγούμενου κύκλου
    sec10++; //αυξάνει την μέτρηση κατά 0.1s
    if(sec10%10 == 1) //θέλουμε τα LED να αναβοσβήνουν περιοδικά
    {
      setTimer1PWM(1023, 1023); //αναβοσβήνει τα LED 1 φορά/sec
      //μπορούμε να χρησιμοποιήσουμε %100 για να αναβοσβήσουν 1
      //φορά/10sec ή %2 για να αναβοσβήσουν 5 φορές/sec

    }
    else //όλες τις υπόλοιπες φορές καθορίζει την κατάσταση των
         //LED και του προειδοποιητικού ηχείου
    {
      int br_red = 0, br_green = 0; //φωτεινότητα από 1 έως 100, η
λειτουργία setLEDs function το μετατρέπει σε ρύθμιση χρόνου
      if(last_CO_ppm_measurement <= red_threshold) //άναψε πράσινο LED
για < 30 PPM
      {//όσο πιο φωτεινό, τόσο χαμηλότερη συγκέντρωση
        br_red = 0; //σβήσε το κόκκινο LED
        br_green = (red_threshold -
last_CO_ppm_measurement)*100.0/red_threshold; //όσο χαμηλότερη η
συγκέντρωση, τόσο μεγαλύτερη είναι η τιμή
        if(br_green < 1) br_green = 1; //μη σβήνεις τελείως
      }
      else //για μέτρηση πάνω από το προκαθορισμένο όριο, άναψε
κοκκινο
      {
        br_green = 0; //και διατήρησε το πράσσινο σβηστό
```

```
        br_red = (last_CO_ppm_measurement-
red_threshold)*100.0/red_threshold; //όσο μεγαλύτερη η συγκέντρωση,
τόσο μεγαλύτερη είναι αυτή η τιμή
        if(br_red < 1) br_red = 1; // μη σβήνεις τελείως

      }

      if(last_CO_ppm_measurement > alarm_ppm_threshold) //αν στα 50sec
του κύκλου μέτρησης είμαστε πάνω από το όριο
        buzz_beep();
      else
        buzz_off();

      setLEDs(br_green, br_red); //όρισε φωτεινότητα LED
    }
  }

  if(phase == 1 && sec10 - high_on > 600) //στο τέλος των 60sec της
θερμής φάσης
    startMeasurementPhase();
  if(phase == 0 && sec10 - low_on > 900) //στο τέλος των 90sec της
ψυχρής φάσης
  {
    last_CO_ppm_measurement = raw_value_to_CO_ppm(sens_val);
    startHeatingPhase();
  }

  float v = analogRead(A0); //διάβασε την μέτρηση
  sens_val *= 0.999; //εφαρμογή εκθετικής εξομάλυνσης με βάση τον τύπο
  sens_val += 0.001*v; // average = old_average*a + (1-a)*new_reading
  if(sec10 - last_print > 9) //έξοδος αποτελεσμάτων 2 φορές/sec
  {
    last_print = sec10;


//Λήψη δεδομένων αισθητήρα DHT-22 και αποθήκευση αυτών στις μεταβλητές
hum και temp
hum = dht.readHumidity();
temp = dht.readTemperature();

//έξοδος αποτελεσμάτων

//έξοδος αποτελεσμάτων αισθητήρα MQ-7
    Serial.print("RAW CO data: ");
    Serial.print(sens_val);
    Serial.print("; phase: ");
    Serial.print(0.1 + phase*1000);
    Serial.print(" ");
    Serial.print("; Last CO Measurement (in ppm): ");
    Serial.println(last_CO_ppm_measurement);
//Print temp and humidity values to serial monitor
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.print("% & Temperature: ");
  Serial.print(temp);
  Serial.println(" Degrees Celsius");

  }
```

Κεφάλαιο 4: Λειτουργία της συσκευής

Φορτώνοντας και εκτελώντας τον παραπάνω κώδικα στην συσκευή, αρχίζουμε να βλέπουμε τα εξής αποτελέσματα: Αρχικά, στη serial monitor έξοδο του Arduino λαμβάνουμε τις παρακάτω ενδείξεις, οι οποίες είναι σχετικές με τη ρύθμιση του PWM. Το Arduino βρίσκει τη σωστή συχνότητα παλμού, η οποία παράγει την απαραίτητη 1,4V τάση για την ψυχρή φάση του αισθητήρα MQ-7. Η τάση αυτή υπολογίζεται ως η διαφορά από τα (εξ' ορισμού) 5V τάσης παροχής του Arduino. Επομένως, αναζητούμε μια τιμή όσο πιο κοντά γίνεται στα 3.6V.

```
11:16:31.865 -> adjusting PWM w=0, V=5.00
11:16:32.661 -> adjusting PWM w=1, V=4.98
11:16:33.458 -> adjusting PWM w=2, V=4.92
11:16:34.302 -> adjusting PWM w=3, V=4.86
11:16:35.099 -> adjusting PWM w=4, V=4.79
11:16:35.896 -> adjusting PWM w=5, V=4.74
11:16:36.739 -> adjusting PWM w=6, V=4.70
11:16:37.536 -> adjusting PWM w=7, V=4.64
11:16:38.368 -> adjusting PWM w=8, V=4.59
11:16:39.164 -> adjusting PWM w=9, V=4.53
11:16:39.961 -> adjusting PWM w=10, V=4.48
11:16:40.805 -> adjusting PWM w=11, V=4.42
11:16:41.602 -> adjusting PWM w=12, V=4.37
11:16:42.398 -> adjusting PWM w=13, V=4.33
11:16:43.242 -> adjusting PWM w=14, V=4.26
11:16:44.039 -> adjusting PWM w=15, V=4.21
11:16:44.883 -> adjusting PWM w=16, V=4.16
11:16:45.680 -> adjusting PWM w=17, V=4.11
11:16:46.476 -> adjusting PWM w=18, V=4.05
11:16:47.320 -> adjusting PWM w=19, V=3.99
11:16:48.117 -> adjusting PWM w=20, V=3.95
11:16:48.914 -> adjusting PWM w=21, V=3.89
11:16:49.757 -> adjusting PWM w=22, V=3.84
11:16:50.554 -> adjusting PWM w=23, V=3.80
11:16:51.351 -> adjusting PWM w=24, V=3.75
11:16:52.195 -> adjusting PWM w=25, V=3.70
11:16:52.992 -> adjusting PWM w=26, V=3.66
11:16:53.788 -> adjusting PWM w=27, V=3.61
```

```
11:16:54.585 -> adjusting PWM w=28, V=3.56
11:16:54.585 -> PWM result: width 27, voltage 3.61
```

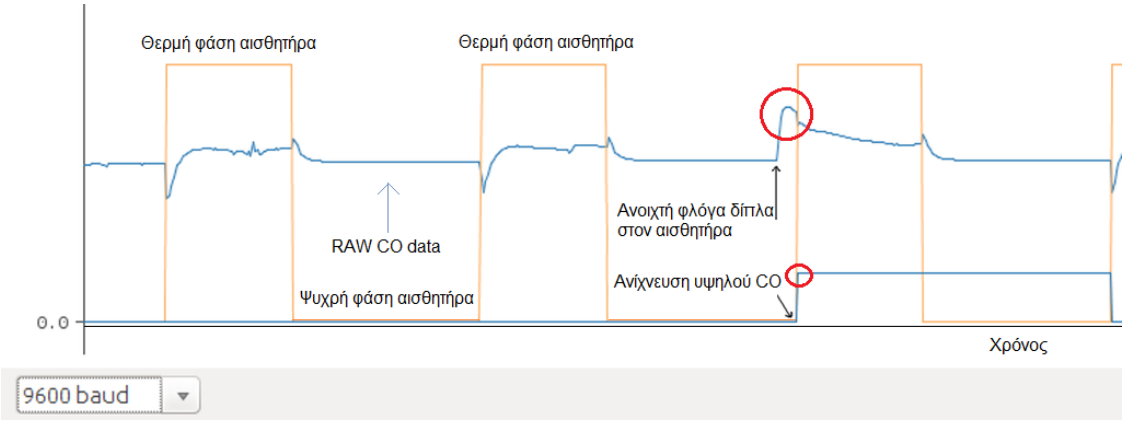
Όταν η ιδανική συχνότητα παλμού εντοπιστεί (στο παράδειγμά μας w=27 με διαφορά τάσης 3.61V), το Arduino κλειδώνει σε αυτήν και την χρησιμοποιεί μέχρι τέλους.

Αμέσως μετά λαμβάνουμε τα αποτελέσματα των μετρήσεων. Ο απλούστερος προγραμματιστικά τρόπος είναι να τα εξάγουμε από την ίδια έξοδο του Arduino (serial monitor). Τα αποτελέσματα αυτά είναι της μορφής:

```
11:27:58.072 -> Humidity: 51.70% & Temperature: 28.20 Degrees
Celsius
11:27:59.009 -> RAW CO data: 783.41; phase: 0.10 ; Last CO
Measurement (in ppm): 3.65
```

Οι πληροφορίες που λαμβάνουμε είναι: η ώρα της μέτρησης, η σχετική υγρασία (Humidity), η θερμοκρασία (Temperature), η RAW μέτρηση του αισθητήρα CO (RAW CO data), η φάση του αισθητήρα κατά τη στιγμή της συγκεκριμένης μέτρησης (0.10 για την ψυχρή φάση, 1000.10 για τη θερμή φάση) και η μετατροπή της RAW μέτρησης CO σε αναλογία ppm. Αυτά τα δεδομένα είναι που χρησιμοποιούμε στο πείραμα αυτής της εργασίας. Επιπροσθέτως, το Arduino παρέχει εγγενώς τη δυνατότητα παραγωγής αποτελεσμάτων σε plotter. Εκμεταλλευόμενοι αυτό, μπορούμε να δείξουμε τι ακριβώς συμβαίνει με τα δεδομένα μας μέσω ενός γραφήματος. Μια τυπική απεικόνιση αυτών φαίνεται παρακάτω. Με πορτοκαλί χρώμα απεικονίζεται η φάση του αισθητήρα MQ-7, με μία θερμή φάση 60” να διαδέχεται μία ψυχρή φάση 90” διαρκώς. Με γαλάζιο χρώμα απεικονίζεται η RAW CO μέτρηση του αισθητήρα. Παρατηρούμε ότι πλησιάζοντας μια ανοιχτή φλόγα από αναπτήρα κοντά στον αισθητήρα MQ-7, αυτό οδηγεί σε αύξηση των σωματιδίων CO που επικάθονται στο στοιχείο μέτρησης αυτού, με αντίστοιχη αύξηση της RAW ένδειξης. Αυτή η μέτρηση μετατρέπεται σε ποσοτική μέτρηση (ppm) στο τέλος κάθε κύκλου, ενώ εφόσον αυτή ξεπεράσει το όριο που έχουμε προκαθορίσει στο πρόγραμμά μας, τότε οδηγεί και σε ηχητική/οπτική προειδοποίηση στο Arduino, μέσω των LED και του ηχείου που έχουμε εντάξει και προγραμματίσει στην συσκευή μας. Αυτό καθιστά την παρακολούθηση των μετρήσεων σε πραγματικό χρόνο μη αναγκαία, εφόσον όταν υπάρχει υπέρβαση κάποιας μέτρηση (στο παράδειγμά μας της

συγκέντρωσης CO στον αέρα), το Arduino θα μας ειδοποιήσει οπτικά (κόκκινο LED) και ηχητικά (συναγερμός από το ηχείο).



Εικόνα 4.1: γραφική απεικόνιση μετρήσεων (serial plotter)

Αξίζει να αναφέρουμε εδώ, ότι η συσκευή θα συνεχίσει να λειτουργεί όσο της παρέχεται ρεύμα. Τυχόν διακοπή οδηγεί σε επανεκκίνηση αυτής, χωρίς την παρέμβαση του χρήστη και την εκτέλεση του αποθηκευμένου κώδικα από την αρχή. Δεν απαιτείται σύνδεση με υπολογιστή για την προειδοποιητική λειτουργία της συσκευής, μόνο για την λήψη δεδομένων.

Κεφάλαιο 5: Μετρήσεις

Ακολουθώντας τις οδηγίες του κατασκευαστή του αισθητήρα MQ-7, απαιτείται να λειτουργήσει για τουλάχιστον 48 ώρες με εναλλαγή των θερμών/ψυχρών φάσεων, ώστε οι μετρήσεις του να έχουν αξιοπιστία. Στην περίπτωσή μας παρατηρήθηκε μια διαφορά της τάξεως του 30% στην έξοδο του αισθητήρα (RAW CO data) τις πρώτες 10 ώρες λειτουργίας του συστήματος, ενώ στη συνέχεια η μέτρηση αυτή σταθεροποιήθηκε, με διακύμανση της τάξεως των 2-5 μονάδων ανά ώρα. Έχοντας ολοκληρώσει αυτή τη διαδικασία (burn-in), ακολουθεί η βαθμονόμηση του αισθητήρα. Η βαθμονόμηση μπορεί να γίνει με δύο τρόπους:

- Απόλυτη βαθμονόμηση: Χρησιμοποιώντας έναν ήδη βαθμονομημένο μετρητή CO και ένα δείγμα με ορισμένη συγκέντρωση CO (για παράδειγμα 100ppm) τροφοδοτούμε το δείγμα αυτό με μια σύριγγα σε ένα κλειστό δοχείο όπου βρίσκεται ο αισθητήρας μας και καταγράφουμε την μέτρηση που αντιστοιχεί σε αυτό το συγκεκριμένο δείγμα. Την τιμή αυτή στη συνέχεια την εισάγουμε στην μεταβλητή **sensor_reading_100_ppm_CO** του κώδικά μας, σαν τιμή αναφοράς για τη συγκεκριμένη συγκέντρωση CO. Η διαδικασία αυτή είναι εξαιρετικά σημαντική, εφόσον μας ενδιαφέρει η απόλυτη ακρίβεια μέτρησης (για παράδειγμα σε συνθήκες εργαστηρίου ή σε κάποια επαγγελματική εφαρμογή του συστήματος), αλλά μικρότερης σημασίας αν μας ενδιαφέρει μόνο η ύπαρξη ή όχι CO στον αέρα (για παράδειγμα κατά τη χρήση στο σπίτι, όπου κανονικά δεν θα πρέπει να ανιχνεύεται καθόλου CO).
- Σχετική βαθμονόμηση: σε περίπτωση που δεν έχουμε πρόσβαση σε βαθμονομημένο μετρητή CO για απόλυτη βαθμονόμηση των μετρήσεων του αισθητήρα MQ-7, μπορούμε να πάρουμε τις ενδείξεις του μετρητή μας (RAW CO data) σε συνθήκες καθαρού αέρα. Την μέτρηση αυτή την εισάγουμε στην μεταβλητή **sensor_reading_clean_air** του κώδικά μας. Το πρόγραμμα υπολογίζει όλες τις παραμέτρους του αισθητήρα και παράγει αποτέλεσμα συγκέντρωσης σε ppm. Η ακρίβεια με αυτόν τον τρόπο βαθμονόμησης είναι χαμηλότερη, αλλά είναι ικανή να διακρίνει διαφορές της τάξεως των 10ppm ή 100ppm.

Καθώς η πρόσβαση σε βαθμονομημένο μετρητή CO δεν είναι εύκολη και δεδομένου ότι η απόλυτη ακρίβεια της συγκέντρωσης CO δεν είναι απαιτούμενη στην εργασία αυτή, αποφασίσαμε να προχωρήσουμε με τη μέθοδο της σχετικής βαθμονόμησης.

Επανειλημμένες μετρήσεις της συγκέντρωσης CO στον αέρα έδειξαν τιμές (raw CO data) στον αισθητήρα μας 580 έως 620, οπότε και αποφασίστηκε να χρησιμοποιηθεί η μέση τιμή των 600 σαν τιμή αναφοράς στον κώδικά μας ως μηδενική συγκέντρωση CO σε ppm (**sensor_reading_clean_air = 600)**. Αξίζει να αναφέρουμε εδώ, ότι εφόσον έχουμε πρόσβαση σε βαθμονομημένο μετρητή αερίων, μπορούμε ανά πάσα στιγμή να επέμβουμε και να αλλάξουμε τον τρόπο υπολογισμού του CO σε ppm, αυξάνοντας έτσι την ακρίβεια της συσκευής μας.

Η συσκευή μας χρησιμοποιήθηκε για την εκτέλεση τριών πειραμάτων. Ακολουθεί μια σύντομη περιγραφή αυτών και της μεθοδολογίας τους:

Πείραμα 1:

Η συσκευή τοποθετήθηκε σε απόσταση 2 μέτρων από την έξοδο καυσαερίων οικιακού λέβητα θέρμανσης αερίου. Ο σκοπός αυτού του πειράματος, είναι να καταγραφεί η ύπαρξη και η συγκέντρωση μονοξειδίου του άνθρακα στην ατμόσφαιρα, ένα προϊόν ατελούς καύσης και βλαβερό για την υγεία. Η μέτρηση αυτή πραγματοποιήθηκε στις 07/03/2021 και είχε διάρκεια 24 ώρες.

Εικόνα 5.1: τοποθέτηση συσκευής για το πείραμα 1

Πείραμα 2:

Η συσκευή τοποθετήθηκε εντός οικιακού ψυγείου. Σκοπός αυτού του πειράματος είναι η καταγραφή της θερμοκρασίας ψύξης και υγρασίας αυτού, καθώς και οι μεταβολές που μπορεί να επέλθουν κατά το άνοιγμα της πόρτας. Η μέτρηση αυτή πραγματοποιήθηκε στις 06/06/2021 και είχε διάρκεια 12 ώρες.

Εικόνα 5.2: τοποθέτηση συσκευής για το πείραμα 2

Πείραμα 3:

Το πείραμα αυτό πραγματοποιήθηκε το Φεβρουάριο του 2021. Η συσκευή τοποθετήθηκε σε μπαλκόνι, προστατευμένη από τα καιρικά φαινόμενα και αφέθηκε να καταγράφει για όλη τη διάρκεια του μήνα. Κύριος σκοπός αυτού του πειράματος, ήταν η απόδειξη της σταθερότητας της συσκευής στην δημιουργία αξιόπιστων μετρήσεων διαρκείας.

Κεφάλαιο 6: Ανάλυση μετρήσεων

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα αποτελέσματα των μετρήσεων για τα 3 πειράματα, καθώς και η ερμηνεία/ανάλυση αυτών.

Πείραμα 1:

Έχοντας αφήσει τη συσκευή να καταγράφει για 24 ώρες, τα δεδομένα που συλλέξαμε έχουν την εξής μορφή:

```
15:14:43.466 -> RAW CO data: 599.00; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:43.532 -> Humidity: 34.40% & Temperature: 17.30
Degrees Celsius
15:14:44.477 -> RAW CO data: 598.20; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:44.544 -> Humidity: 34.40% & Temperature: 17.40
Degrees Celsius
15:14:45.478 -> RAW CO data: 598.81; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:45.546 -> Humidity: 34.40% & Temperature: 17.40
Degrees Celsius
15:14:46.485 -> RAW CO data: 597.48; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:46.552 -> Humidity: 34.40% & Temperature: 17.40
Degrees Celsius
15:14:47.492 -> RAW CO data: 595.97; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:47.559 -> Humidity: 34.40% & Temperature: 17.40
Degrees Celsius
15:14:48.495 -> RAW CO data: 594.66; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
15:14:48.562 -> Humidity: 34.40% & Temperature: 17.40
Degrees Celsius
15:14:49.500 -> RAW CO data: 592.46; phase: 0.10 ; Last CO
Measurement (in ppm): 0.00
```

Η σειριακή έξοδος του Arduino δίνει τη δυνατότητα να πάρουμε αυτά τα δεδομένα (είτε σε πραγματικό χρόνο, είτε στο τέλος της μέτρησης) σε μορφή κειμένου και στη συνέχεια να τα μετατρέψουμε σε κάποια άλλη μορφή (για παράδειγμα .csv για εισαγωγή στο Excel). Κατά την εισαγωγή των δεδομένων αποφασίστηκε η ελάττωση του dataset σε μία εγγραφή / 10 λεπτά, καθώς η ανάλυση των μετρήσεων ανά ένα δευτερόλεπτο δεν κρίθηκε απαραίτητη. Τα δεδομένα αυτά επομένως διαμορφώνονται, όπως φαίνονται στον πίνακα 6.1, ενώ με αυτή τη μορφή θα παρουσιάζονται εφεξής και για τα επόμενα πειράματα.

Time	Temperature (° Celsius)	Humidity (%, relative)	RAW CO	CO (in ppm)
15:15:00	17,4	34,7	594,42	0,00
15:25:00	18,0	32,9	604,48	0,00
15:35:00	17,1	33,8	751,18	0,00
15:45:00	16,8	34,5	585,89	0,00
15:55:00	15,9	35,6	593,51	0,00
16:05:00	15,9	35,1	597,44	0,00
16:15:00	15,6	35,4	571,27	0,00

Πίνακας 6.1: μορφή δεδομένων

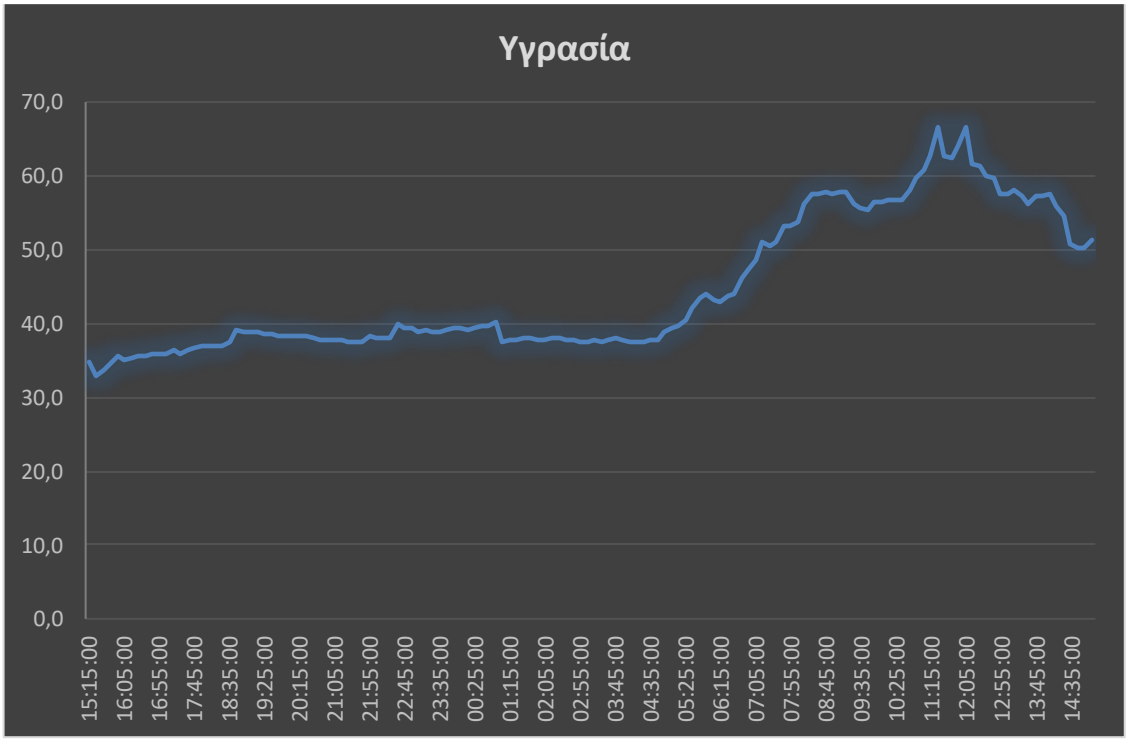
Έχοντας λοιπόν μετατρέψει τα δεδομένα μας σε μια πιο ευέλικτη προς επεξεργασία μορφή, η παρουσίασή τους με μορφή πινάκων είναι εύκολη:

Ελάχιστη θερμοκρασία	Μέγιστη θερμοκρασία	Μέση θερμοκρασία
8,8° C	18,0° C	10,90° C
Ελάχιστη υγρασία	**Μέγιστη υγρασία**	**Μέση υγρασία**
32,9%	66,6%	44,69%
Ελάχιστο CO	**Μέγιστο CO**	**Μέσο CO**
0,00	2,68	0,06

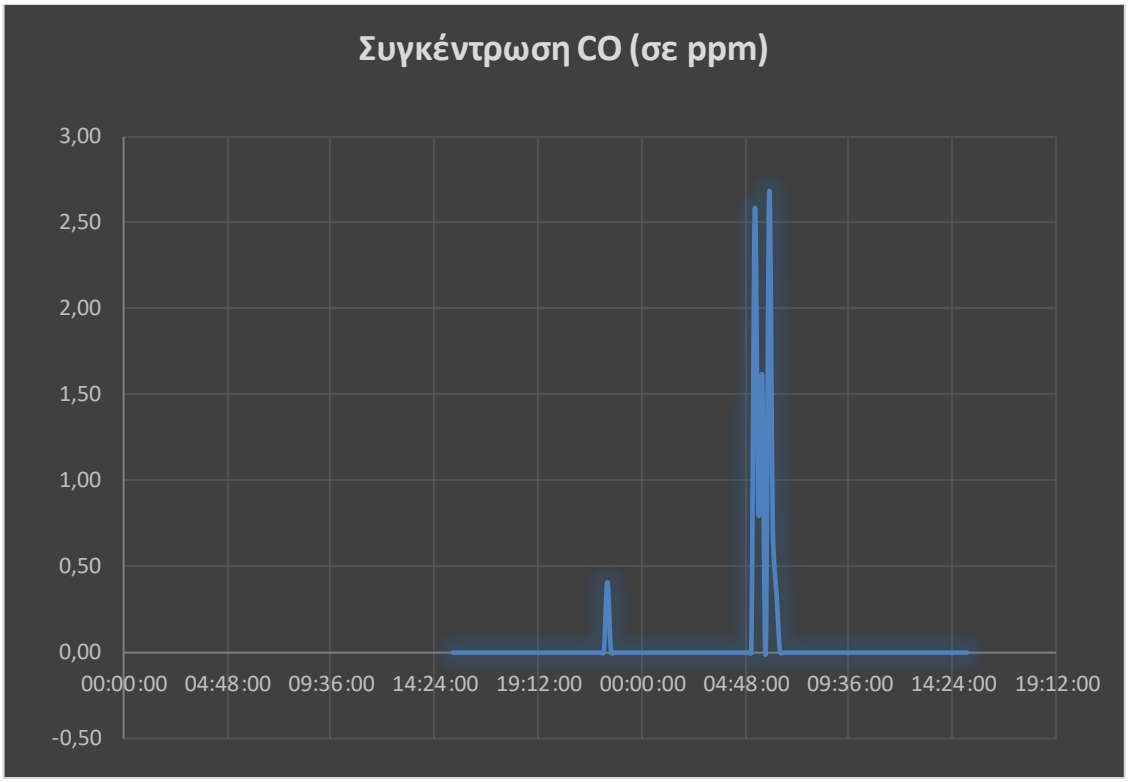
Πίνακας 6.2: ελάχιστη, μέγιστη και μέση τιμή μετρήσεων πειράματος 1

Επιπλέον των πινάκων, μπορούμε να δημιουργήσουμε γραφήματα απεικόνισης των δεδομένων μας, όπως παρακάτω:

Διάγραμμα 6.1: απεικόνιση θερμοκρασίας πειράματος 1



Διάγραμμα 6.2: απεικόνιση υγρασίας πειράματος 1



Διάγραμμα 6.3: απεικόνιση συγκέντρωσης CO πειράματος 1

Με προσεχτική μελέτη των δεδομένων μας παρατηρούμε ότι η μέση (average) θερμοκρασία για το 24ωρο των μετρήσεων του πειράματος 1 ήταν 10,9ο C, με ελάχιστη τους 8,8ο C στις 00:55 και μέγιστη τους 18ο C στις 15:25. Ταυτόχρονα, στις 15:25, καταγράφεται το χαμηλότερο ποσοστό υγρασίας με τιμή 32,9%, ενώ η μέγιστη υγρασία καταγράφεται στις 11:25 της ερχόμενης ημέρας, με τιμή 66,6%, όταν η θερμοκρασία είναι 9,8ο C. Αξίζει να σημειωθεί ότι αυτές οι τιμές δεν εκφράζουν τις καιρικές συνθήκες που επικρατούσαν κατά το διάστημα της μέτρησης, καθώς με την τοποθέτηση της συσκευής στα 2m από την έξοδο καυσαερίων ενδέχεται η υγρασία και η θερμοκρασία που καταγράφεται να επηρεάζεται από τα καυσαέρια του λέβητα, τα οποία αποτελούνται κυρίως από CO_2 σε υψηλή θερμοκρασία και H_2O λόγω της καύσης του αερίου. Οι καιρικές συνθήκες το διάστημα της μέτρησης, όπως καταγράφηκαν από τον μετεωρολογικό σταθμό της ΔΕΘ σε απόσταση 1,5χλμ από το σημείο της εκτέλεσης του πειράματος αναφέρουν μέγιστη θερμοκρασία 16,1ο C, ελάχιστη θερμοκρασία 7,7ο C, μέγιστη σχετική υγρασία 82% και ελάχιστη σχετική υγρασία 43%. Αυτή η μικρή απόκλιση σε σχέση με τις μετρήσεις της συσκευής στο σημείο όπου αυτή τοποθετήθηκε θεωρείται μη σημαντική.

Το σημαντικότερο εύρημα του πειράματος είναι η ανίχνευση CO σαν παράγωγο καύσης. Οι συγκεντρώσεις που ανιχνεύθηκαν ήταν 0,41ppm CO στις 22:25, 2,58ppm CO στις 05:15, 0,81ppm CO στις 05:25, 1,61ppm CO στις 05:35, 2,68ppm CO στις 05:55, 0,70ppm CO στις 06:05 και 0,34ppm CO στις 06:15.

Time	CO (in ppm)
05:15:00	**2,58**
05:25:00	**0,81**
05:35:00	**1,61**
05:45:00	0,00
05:55:00	**2,68**
06:05:00	**0,70**
06:15:00	**0,34**

Πίνακας 6.3: ανίχνευση CO

Οι συγκεντρώσεις αυτές δεν θεωρούνται σημαντικές, ώστε να αποτελέσουν κίνδυνο για την υγεία του ανθρώπου, ειδικά εφόσον ανιχνεύονται σε ανοιχτό χώρο. Παρόλα αυτά υποδηλώνουν κάποια δυσλειτουργία του συστήματος καύσης, καθώς θεωρητικά δεν θα έπρεπε να ανιχνεύσουμε καθόλου CO στην ατμόσφαιρα. Το πείραμα επαναλήφθηκε μετά από την ετήσια συντήρηση του καυστήρα στις 03/05/2021, με τοποθέτηση της συσκευής σε απόσταση 1m από την έξοδο καυσαερίων, χωρίς να σημειωθεί ανίχνευση CO. Αυτό υποδηλώνει κάποια πιθανή δυσλειτουργία, η οποία διορθώθηκε κατά τη συντήρηση αυτή. Επιβεβαιώνεται επίσης, η απαίτηση από την εταιρεία διανομής αερίου Θεσσαλονίκης όλων των καυστήρων αερίου να βρίσκονται σε εξωτερικό χώρο, καθώς παρόμοιες συγκεντρώσεις σε βάθος χρόνου σε κλειστό χώρο (πχ σε κάποιο μη επαρκώς αεριζόμενο υπόγειο) μπορούν να οδηγήσουν σταδιακά σε μεγάλη και πιθανώς επιζήμια συγκέντρωση CO για τον άνθρωπο.

Πείραμα 2:

Σε αυτό το πείραμα η συσκευή τοποθετήθηκε εντός ψυγείου, με σκοπό να καταλάβουμε την μεταβολή στη θερμοκρασία αυτού κατά την αυτόματη λειτουργία του θερμοστάτη του. Αναμένουμε να δούμε μια μικρή διακύμανση όσο αυτός ενεργοποιείται/απενεργοποιείται, αλλά κυρίως ενδιαφερόμαστε να μην υπάρχει μεγάλη απόκλιση μεταξύ των ακραίων τιμών και της μέσης (average) θερμοκρασίας. Επίσης, ανοίξαμε την πόρτα του ψυγείου για 30" δύο φορές, για να δούμε πως αυτή η κίνηση επηρεάζει τη θερμοκρασία εντός και πόση ώρα χρειάζεται για να επανέλθει αυτή στα καθορισμένα επίπεδα. Κατά τις 12 ώρες διάρκειας του πειράματος αυτού δεν υπήρχε ρύθμιση της εξωτερικής θερμοκρασίας (του χώρου δηλαδή στον οποίο βρίσκεται το ψυγείο), ώστε να διαπιστώσουμε αν αυτό μπορεί να διατηρεί τη ψύξη του σε συνθήκες κανονικής λειτουργίας. Οι μετρήσεις μας έδειξαν τα εξής:

Συνολικές μετρήσεις		
Ελάχιστη θερμοκρασία	**Μέγιστη θερμοκρασία**	**Μέση θερμοκρασία**
3,3^{o} C	6,5^{o} C	4,96^{o} C
Ελάχιστη υγρασία	**Μέγιστη υγρασία**	**Μέση υγρασία**
16,20%	78,6%	28,72%

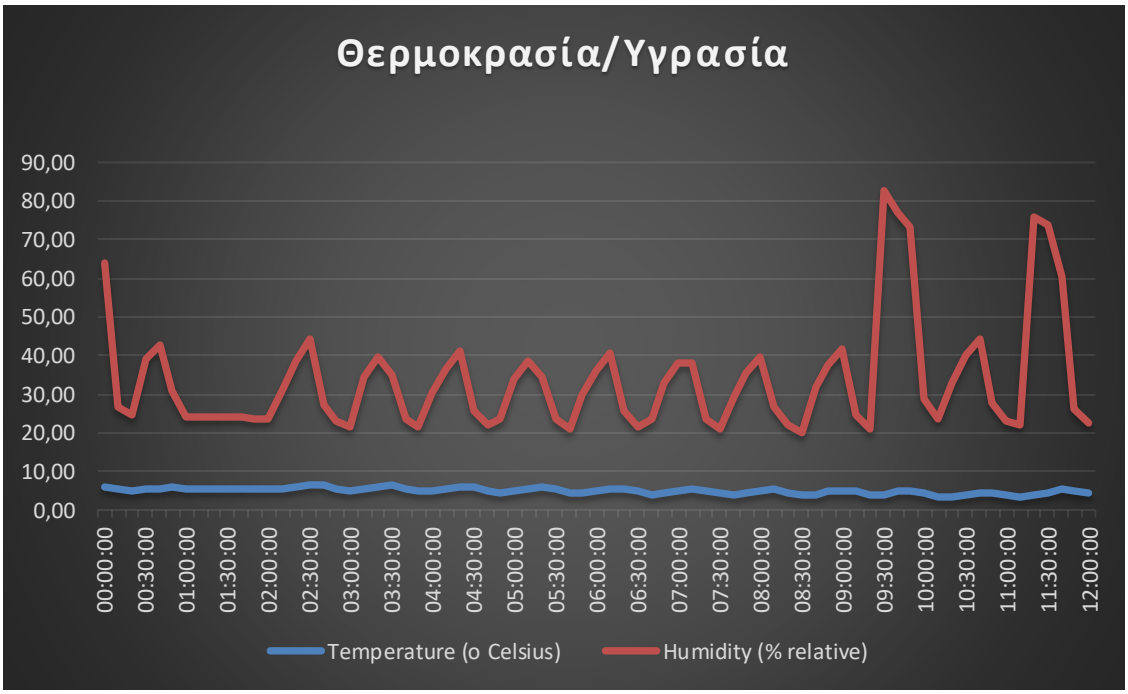
Πίνακας 6.4: ελάχιστη, μέγιστη και μέση τιμή μετρήσεων πειράματος 2

Κατά την εκτέλεση του πειράματος ανοίξαμε την πόρτα του ψυγείου στην αρχή, ώστε να τοποθετήσουμε τη συσκευή και στη συνέχεια επιπλέον 2 φορές, η πρώτη στις 9 ώρες και 30 λεπτά από την έναρξη των μετρήσεων και η δεύτερη 1 ώρα και 50 λεπτά αργότερα από την πρώτη, στις 11:20 των μετρήσεων. Σκοπός μας ήταν, αφενός να έχουμε έναν ικανό αριθμό μετρήσεων στην αρχή του πειράματος, όπου δεν αλληλοεπιδράσαμε με το ψυγείο (πόρτα διαρκώς κλειστή), οι οποίες μας δίνουν μια εικόνα για την απόδοση του ψυγείου σε ιδανικές συνθήκες, αφετέρου θέλαμε να δούμε τι συμβαίνει όταν ανοίξει η πόρτα για 30''και πόσος χρόνος χρειάζεται για να επανέλθει το ψυγείο σε θερμοκρασία παρόμοια με πριν. Χρησιμοποιώντας λοιπόν ένα υποσύνολο των δεδομένων μας, ξεκινώντας μία ώρα από την έναρξη του πειράματος (ώρα 01:00) και μέχρι πριν το πρώτο άνοιγμα της πόρτας (ώρα 09:00), έχουμε έναν ικανό αριθμό μετρήσεων για 8 συνεχόμενες ώρες με κλειστή πόρτα. Οι μετρήσεις μας σε αυτή την περίπτωση διαμορφώνονται ως εξής:

Μετρήσεις συνεχούς λειτουργίας με κλειστή πόρτα		
Ελάχιστη θερμοκρασία	**Μέγιστη θερμοκρασία**	**Μέση θερμοκρασία**
3,8^{o} C	6,5^{o} C	5,15^{o} C
Ελάχιστη υγρασία	**Μέγιστη υγρασία**	**Μέση υγρασία**
16,20%	37,90%	24,47%

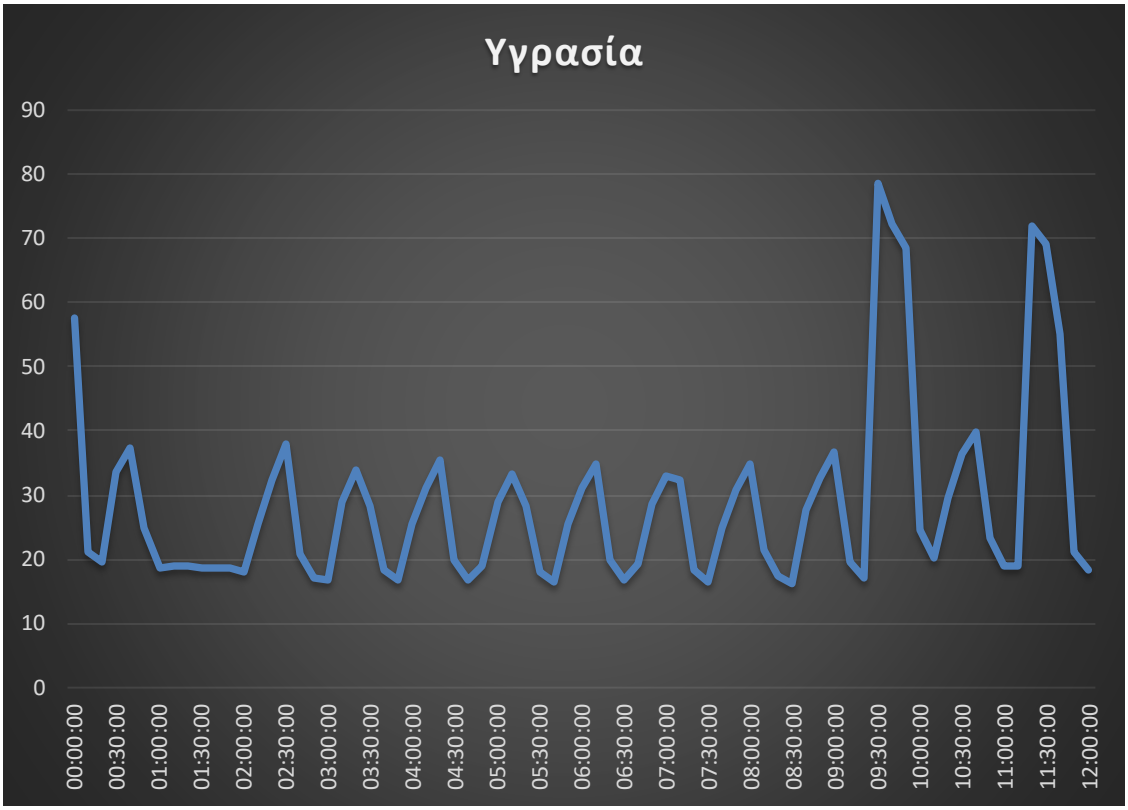
Πίνακας 6.5: ελάχιστη, μέγιστη και μέση τιμή μετρήσεων με διαρκώς κλειστή πόρτα

Εύκολα παρατηρείται ότι το ψυγείο παρουσιάζει μεγάλη σταθερότητα στον τομέα διατήρησης σταθερής της υγρασίας, όσο η πόρτα παραμένει κλειστή, με τη μέγιστη υγρασία στο 37,90% για το υπό εξέταση χρονικό διάστημα, αρκετά κοντά στη μέση υγρασία 24,7%, σε αντίθεση με τη συνολική μέγιστη υγρασία 78,6% η οποία παρατηρείται αμέσως μετά το πρώτο άνοιγμα της πόρτας στις 09:29. Παρόλα αυτά παρατηρείται ότι η ελάχιστή του θερμοκρασία είναι ελάχιστα υψηλότερη από τη συνολική (3,8^{o} C έναντι 3,3^{o} C), κάτι που ανεβάζει τη μέση θερμοκρασία από 4,96^{o} C στους 5,15^{o} C. Το ότι ελάχιστη θερμοκρασία παρατηρείται 1 ώρα και 40’ αφότου ανοίξουμε την πόρτα (δηλαδή στις 11:10), υποδεικνύει ότι ο συμπιεστής του ψυγείου δουλεύει εντονότερα για να επαναφέρει τη θερμοκρασία στα προκαθορισμένα επίπεδα, ξεπερνώντας την χαμηλότερη μέτρηση που έχουμε λάβει όσο το ψυγείο λειτουργούσε με σταθερά κλειστή την πόρτα του. Επίσης, παρατηρείται ότι χρειάζονται περίπου 40’ μετά το άνοιγμα της πόρτας, ώστε να επανέλθει η θερμοκρασία και η υγρασία στις προηγούμενες τιμές της.



Διάγραμμα 6.4: συνολική απόδοση ψυγείου

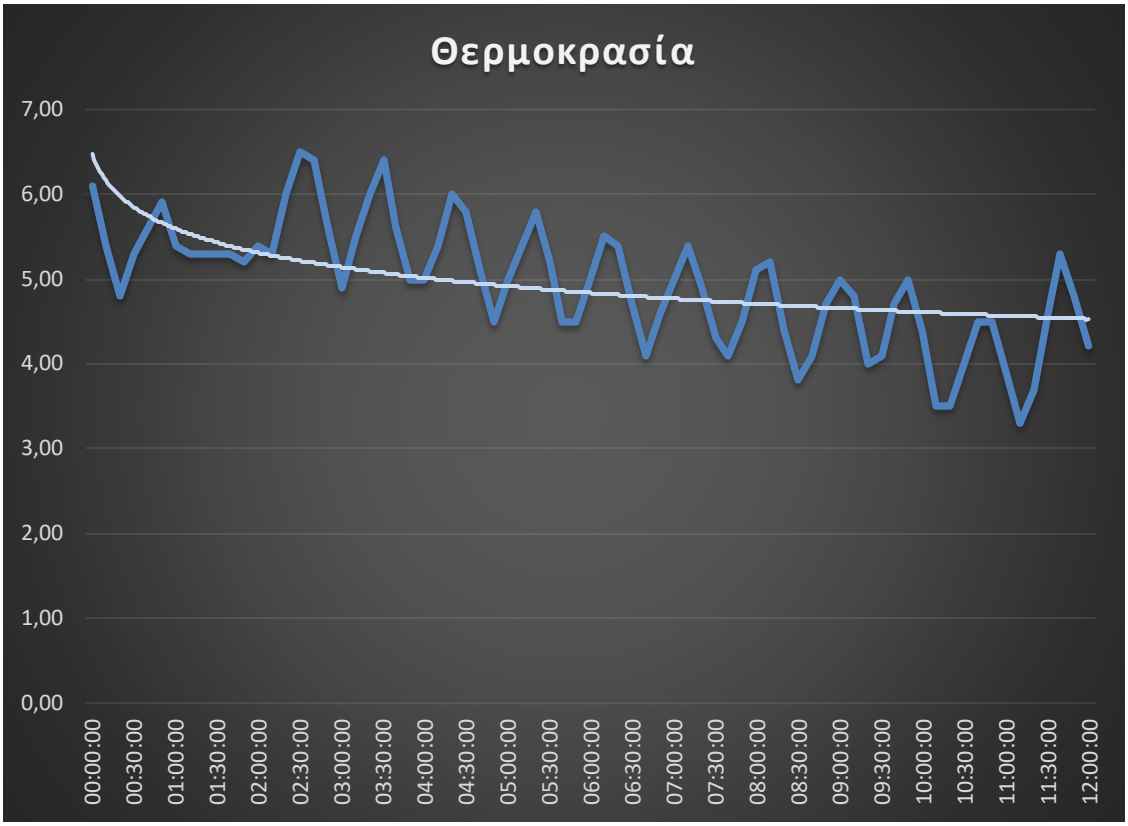
Στο διάγραμμα 6.4 παρατηρούμε τις διακυμάνσεις θερμοκρασίας/υγρασίας κατά το συνολικό χρόνο του πειράματος. Βλέπουμε ότι συμβαδίζουν, μειωμένες όσο ο συμπιεστής του ψυγείου λειτουργεί και αυξανόμενες κατά τα υπόλοιπα διαστήματα.



Διάγραμμα 6.5: απεικόνιση υγρασίας εντός του ψυγείου

Αναλυτικότερα, στο διάγραμμα 6.5, βλέπουμε την διακύμανση της υγρασίας εντός του ψυγείου, με δύο μεγάλες διακυμάνσεις στις 09:30 και στις 11:20 όπου ανοίγουμε την πόρτα αυτού.

Στο διάγραμμα 6.6 παρατηρούμε κάτι αντίστοιχο με την θερμοκρασία, η οποία μειώνεται κατά το διάστημα λειτουργίας του συμπιεστή και αυξάνεται όσο αυτός είναι ανενεργός και μέχρι να ενεργοποιηθεί ξανά. Παρατηρούμε επίσης, ότι στο πρώτο άνοιγμα της πόρτας στις 09:30 αυτή δεν μεταβάλλεται σημαντικά, στη συνέχεια ο συμπιεστής δουλεύει εντονότερα και πετυχαίνει την απόλυτα χαμηλότερη θερμοκρασία στις 11:10, ακριβώς πριν ανοίξουμε ξανά την πόρτα στις 11:20.



Διάγραμμα 6.6: απεικόνιση θερμοκρασίας εντός του ψυγείου

Τα παραπάνω δεδομένα μας δίνουν μια καλή εικόνα για την απόδοση του ψυγείου, ενώ η συγκεκριμένη άσκηση μπορεί να βρει εφαρμογή σε ψυγεία όπου η συντήρηση των δειγμάτων και η καταγραφή του ιστορικού συντήρησης αυτών έχει ιδιαίτερα μεγάλη σημασία. Αυτά μπορεί να είναι ψυγεία διαγνωστικών κέντρων, μεταφοράς ή συντήρησης εμβολίων, ακόμα και ψυγεία που αφορούν τη διακίνηση τροφίμων. Με τις κατάλληλες τροποποιήσεις του κώδικα μπορεί να προστεθεί ηχητική και οπτική προειδοποίηση, όταν οι μετρήσεις είναι εκτός των προκαθορισμένων από εμάς ορίων, όπως και με την υλοποίηση του κώδικα για το πείραμα 1. Με μικρό επιπλέον κόστος μπορεί να προστεθεί και οθόνη ένδειξης των μετρήσεων σε πραγματικό χρόνο.

Πείραμα 3:

Το πείραμα αυτό είχε κύριο στόχο να αποδείξει την αξιοπιστία της συσκευής μας, καθώς και να αναδείξει εναλλακτικές χρήσεις της πλατφόρμας Arduino.

Η συσκευή τοποθετήθηκε σε ταράτσα οικίας και αφέθηκε να καταγράφει. Η δειγματοληψία ορίστηκε στα 15’ και έτσι καταφέραμε να λάβουμε έναν πολύ μεγάλο όγκο περιβαλλοντικών δεδομένων (2688 εγγραφές) καθ' όλη τη διάρκεια του Φεβρουαρίου του 2021.

ΗΜΕΡΑ	ΜΕΓΙΣΤΗ ΘΕΡΜΟΚΡΑΣΙΑ	ΕΛΑΧΙΣΤΗ ΘΕΡΜΟΚΡΑΣΙΑ	ΜΕΓΙΣΤΗ ΥΓΡΑΣΙΑ	ΕΛΑΧΙΣΤΗ ΥΓΡΑΣΙΑ	ΜΕΓΙΣΤΟ CO
1	17,7	9,5	89	50	0
2	16,6	10,1	85	61	0
3	16,9	8,1	90	56	0
4	15,3	9,1	90	71	0
5	16	9,1	89	58	0
6	13,8	10,1	90	77	0
7	12,3	9,4	92	84	0
8	14,7	10,5	91	73	0
9	19,1	10,4	79	44	0
10	18,3	9,8	75	47	0
11	19,1	7	78	45	0
12	7,4	2,8	57	30	0
13	2,7	-1,9	82	43	0
14	-1,8	-3,8	86	72	0
15	4,5	-2,5	71	38	0
16	5,1	-0,6	58	33	0
17	6,3	-1,3	75	32	0
18	14,4	3,4	70	38	0
19	13,4	3,8	76	41	0
20	12,2	8,9	84	69	0
21	12,4	9	83	65	0
22	13,6	6,3	87	64	0
23	16,9	5,9	88	52	0
24	20,4	7,4	86	32	0
25	17,6	8,3	87	43	0
26	17	8,3	84	50	0
27	18	7,8	85	38	0
28	14,1	7,6	80	51	0

Πίνακας 6.6: Δεδομένα πειράματος 3

Αξίζει να σημειωθεί ότι η συσκευή ελεγχόταν καθημερινά για τη σωστή λειτουργία της και δεν παρουσίασε καμία βλάβη ή δυσλειτουργία στο διάστημα των 28 αυτών ημερών.

Με μικρό επιπλέον κόστος και σχετική τροποποίηση του κώδικα είναι δυνατόν να υπάρχει απομακρυσμένος έλεγχος της συσκευής, ώστε να μην απαιτείται η φυσική μας παρουσία για τη λήψη των μετρήσεων.

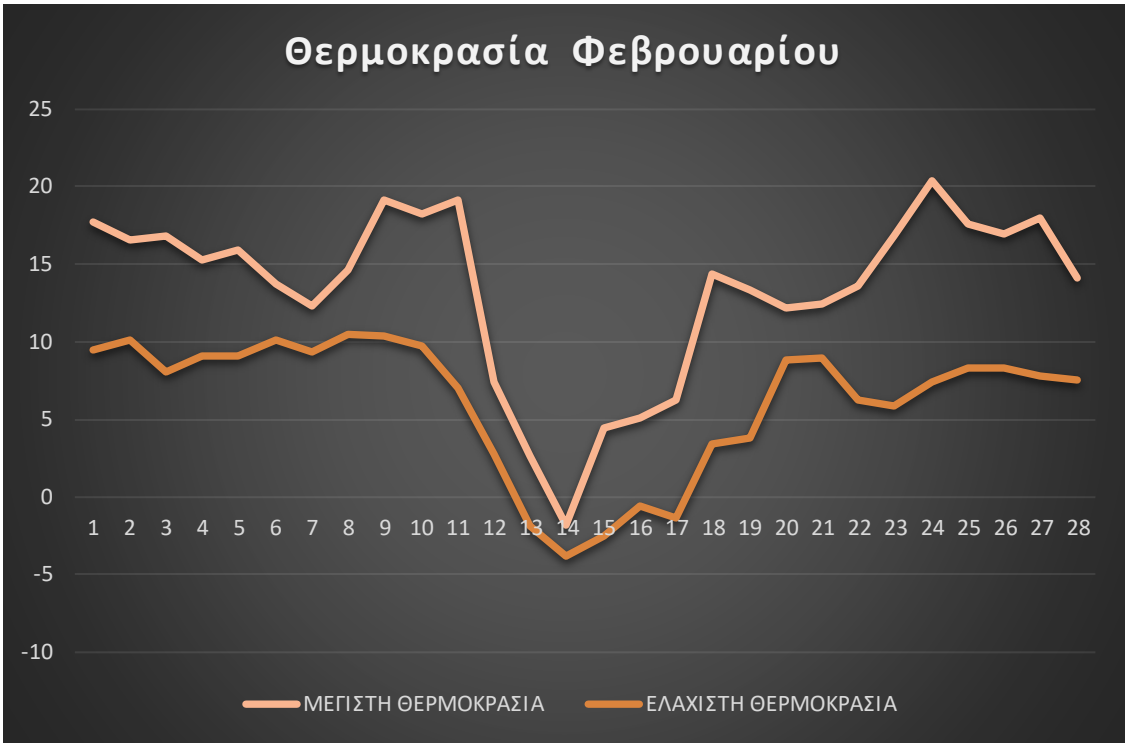
Διάγραμμα 6.7: μετρήσεις συστήματος για το Φεβρουάριο του 2021

Από τα δεδομένα που συνέλλεξε το σύστημά μας, προκύπτουν τα παρακάτω στοιχεία:

Μέγιστη θερμοκρασία Φεβρουαρίου 2021	**Ελάχιστη θερμοκρασία Φεβρουαρίου 2021**
20,4^{o} C	-3,8^{o} C
Μέγιστη υγρασία Φεβρουαρίου 2021	**Ελάχιστη υγρασία Φεβρουαρίου 2021**
92 %	30 %

Πίνακας 6.7: μέγιστες και ελάχιστες τιμές

Ενδιαφέρον προκαλεί και ο συσχετισμός της θερμοκρασίας με την υγρασία. Επί παραδείγματι, για τις 14/02/2021, όπου καταγράφονται οι χαμηλότερες θερμοκρασίες του μήνα, βλέπουμε μια απότομη αύξηση της συνολικής υγρασίας.



Διάγραμμα 6.8: μετρήσεις θερμοκρασίας Φεβρουαρίου 2021

Διάγραμμα 6.9: μετρήσεις υγρασίας Φεβρουαρίου 2021

Κεφάλαιο 7: Συμπεράσματα

Κρίνοντας την απόδοση του συστήματός μας σε διάρκεια αρκετών μηνών, μπορούμε με βεβαιότητα να δηλώσουμε, ότι ο στόχος αυτής της εργασίας επετεύχθη. Ο σχεδιασμός και η κατασκευή ενός συστήματος ελέγχου περιβαλλοντικών συνθηκών είναι δυνατός από κάποιον χωρίς ιδιαίτερες γνώσεις ηλεκτρονικής και προγραμματισμού. Το κόστος υλοποίησης επίσης κρίνεται ικανοποιητικό και μπορεί να μειωθεί περισσότερο αν αποφασιστεί η κατασκευή περισσότερων του ενός συστήματος, όπου η μαζική προμήθεια των απαραίτητων εξαρτημάτων είναι ακόμα οικονομικότερη. Η ευελιξία του συστήματος επίσης θεωρείται σημαντική, καθώς σε αυτό μπορούν να προστεθούν επιπλέον αισθητήρες και εργαλεία και με τον κατάλληλο προγραμματισμό μπορούμε να λάβουμε τα δεδομένα που μας ενδιαφέρουν, καθιστώντας την όποια επένδυση πάνω στην πλατφόρμα Arduino διαχρονική και συνεχούς αξίας. Η συσκευή μας σχεδιάστηκε για χρήση σε νοσοκομειακές δομές (πείραμα 1), αλλά αποδείξαμε ότι με μικρές μετατροπές στον κώδικά της, μπορεί να έχει επιπλέον χρήσεις, όπως την πιστοποίηση σωστής θερμικής συντήρησης ευαίσθητων ουσιών/δειγμάτων (πείραμα 2) ή την καταγραφή περιβαλλοντικών συνθηκών (πείραμα 3).

Βιβλιογραφία

Yuko Saso, Hiroshi Gotoda and Yoshio Ogawa (2005). Effect of Oxygen Concentration on the Carbon Monoxide Yields from Methane and Methanol Flames

Michael Leung, Alan H.S. Chan (2006). Control and management of hospital indoor air quality

L. J. McCann, R. Close, L. Staines, M. Weaver, G. Cutter, and G. S. Leonardi (2013). Indoor Carbon Monoxide: A Case Study in England for Detection and Interventions to Reduce Population Exposure

Διαδικτυακοί τόποι

https://www.raspberrypi.org
https://www.arduino.cc
https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf
https://www.epa.gov/indoor-air-quality-iaq/carbon-monoxides-impact-indoor-air-quality