ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αυτοματοποίηση παρασκευαστή παγοκύβων

«Ειδικά Θέματα σε Συστήματα Ηλεκτρικών Μετρήσεων» (ΗΡΥ602)

ΛΑΖΑΡΙΔΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Πρόλογος

Η εργασία αυτή αφορά την επισκευή μιας παγομηχανής. Συγκεκριμένα, είχε καταστραφεί η πλακέτα και μέρος του κάδου. Θα ήθελα να ευχαριστήσω προσωπικά τον καθηγητή Καλαϊτζάκη Κωσταντίνο για την ευκαιρία που μου έδωσε να συμμετέχω στο μάθημα αυτό.

Περίληψη

Το θέμα της εργασίας είναι η δημιουργία του κυκλώματος για τον έλεγχο μιας παγομηχανής. Ο μικροεπεξεργαστής που χρησιμοποιήθηκε, είναι το Arduino Nano το οποίο επεξεργάζεται τα δεδομένα από τους αισθητήρες και τις εισόδους που δίνει ο χρήστης ώστε να δημιουργήσει παγοκύβους. Στη συγκεκριμένη εργασία, φαίνεται η χρήση αλγορίθμου για την ανάγνωση της θερμοκρασίας καθώς και η εύρεση της καμπύλης θερμοκρασίας-αντίστασης από άγνωστο θερμόμετρο, η χρήση μεταλλικής γειωμένης πλάκας για την αποφυγή ηλεκτρομαγνητικών παρεμβολών, η μελέτη των εξαρτημάτων που χρησιμοποιήθηκαν, η κατασκευή της πλακέτας καθώς και ο προγραμματισμός ενός σύνθετου προβλήματος.

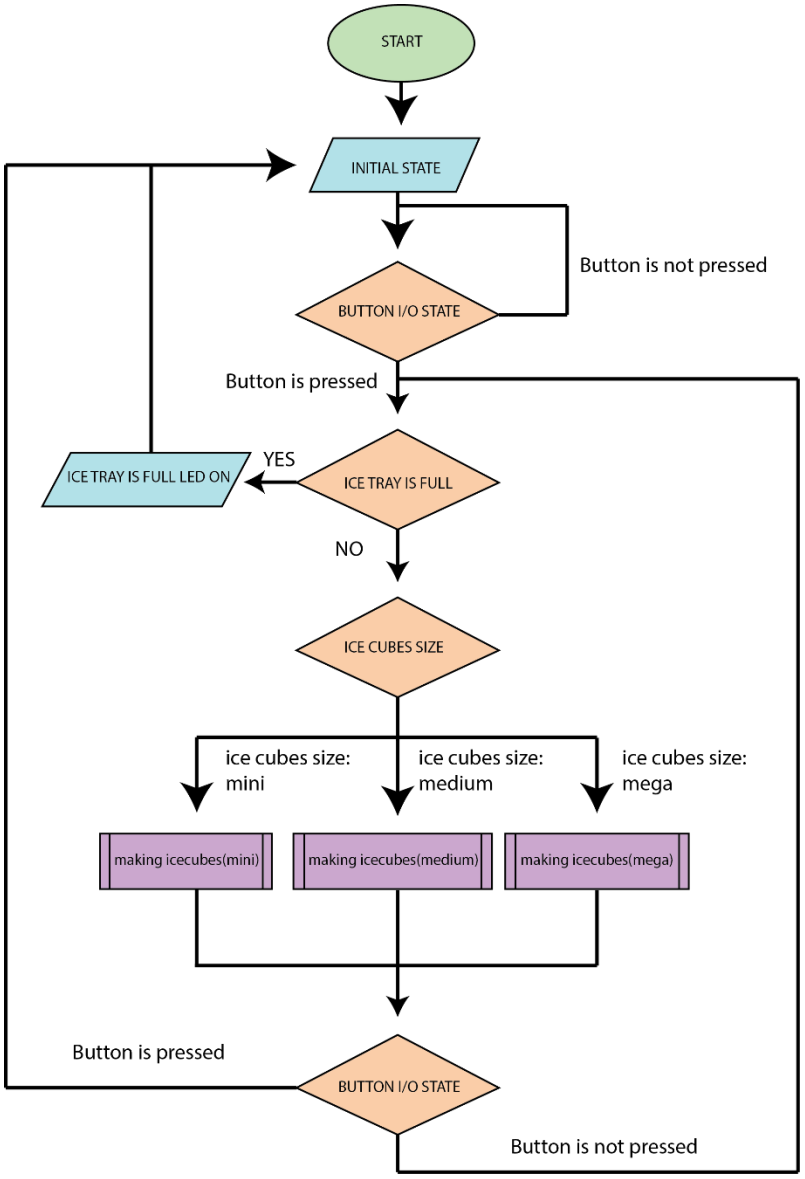
Περιεχόμενα

1. Γενικά

Η αρχή λειτουργίας της παγομηχανής είναι η συμπίεση του αέριου(δηλ. εφαρμογή υψηλής πίεσης) και στη συνέχεια η ψύξη του μέσω μιας ψύκτρας και ενός ανεμιστήρα. Πλέον η θερμοκρασία του συμπιεσμένου αερίου βρίσκετε κοντά στην θερμοκρασία του δωματίου. Καθώς το αέριο περνάει από μία βαλβίδα η πίεση και η θερμοκρασία του μειώνονται ραγδαία. Στη συνέχεια κατευθύνεται σε σωλήνες οι οποίες βρίσκονται σε επαφή με το νερό και λόγω της διαφοράς θερμοκρασίας, το αέριο απορροφά θερμότητα από το νερό και μέσα σε λίγα λεπτά το νερό παγώνει. Για να αφαιρεθούν οι παγοκύβοι από τα μεταλλικά στοιχεία υπάρχει μια βαλβίδα η οποία εξισορροπεί τις πιέσεις. Όταν οι πιέσεις εξισορροπηθούν τα στοιχεία αυτά θερμαίνονται ελάχιστα και η εσωτερική πλευρά των παγοκύβων λιώνει, αυτό είναι αρκετό ώστε να τους επιτραπεί να «γλιστρήσουν» και να πέσουν στο κάδο περισυλλογής.

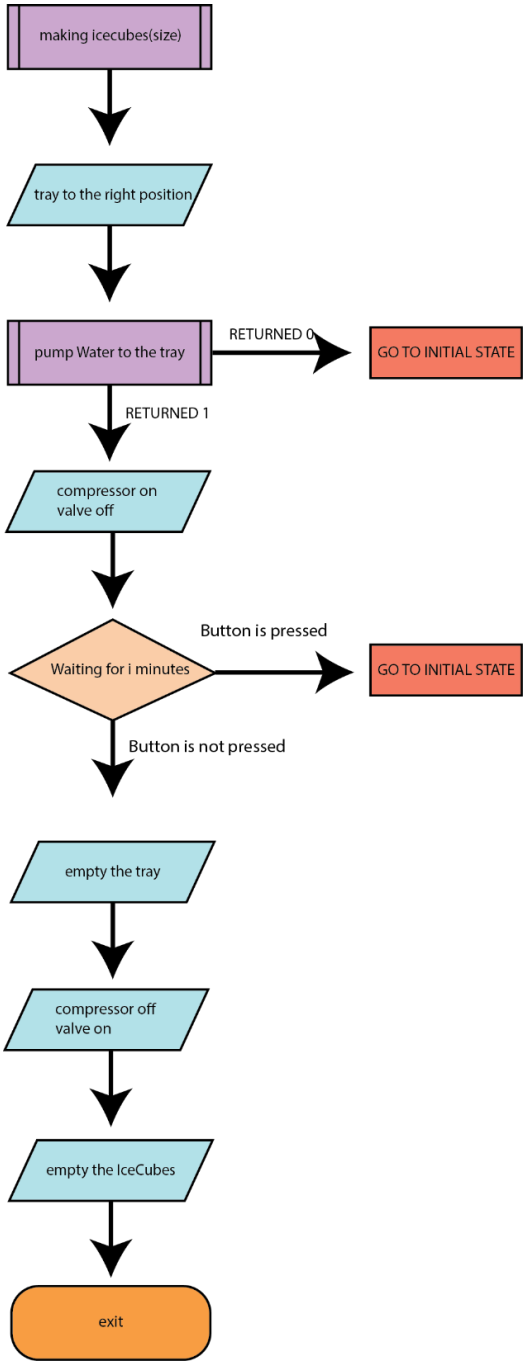
1. Αρχή λειτουργίας

Το παρακάτω διάγραμμα ροής περιγράφει τον αλγόριθμο λειτουργίας της παγομηχανής. Αρχικά υπάρχει μία κατάσταση InitialState στην οποία ο συμπιεστής είναι απενεργοποιημένος και η βαλβίδα αποσυμπίεσης είναι ανοιχτή, ώστε να μην υπάρχουν πιέσεις στο συμπιεστή, οι οποίες του προκαλούν μεγάλες φθορές. Επίσης σε αυτή την κατάσταση αδειάζει μια φορά το δοχείο ψύξης νερού, ώστε να μην υπάρχουν παγοκύβοι ή νερό και έτσι εξασφαλίζεται η ομαλή λειτουργία του. Όταν ο χρήστης πατήσει το κουμπί λειτουργίας, ο αλγόριθμος ελέγχει αν το δοχείο περισυλλογής παγοκύβων είναι γεμάτο. Στη περίπτωση που δεν είναι γεμάτο ξεκινά η διαδικασίας δημιουργίας παγοκύβων ανάλογα με το πιο μέγεθος έχει επιλέξει ο χρήστης. Αυτή η διαδικασία επαναλαμβάνεται έως ότου το δοχείο περισυλλογής παγοκύβων γεμίσει είτε ο χρήστης να πατήσει το διακόπτη παύσης.



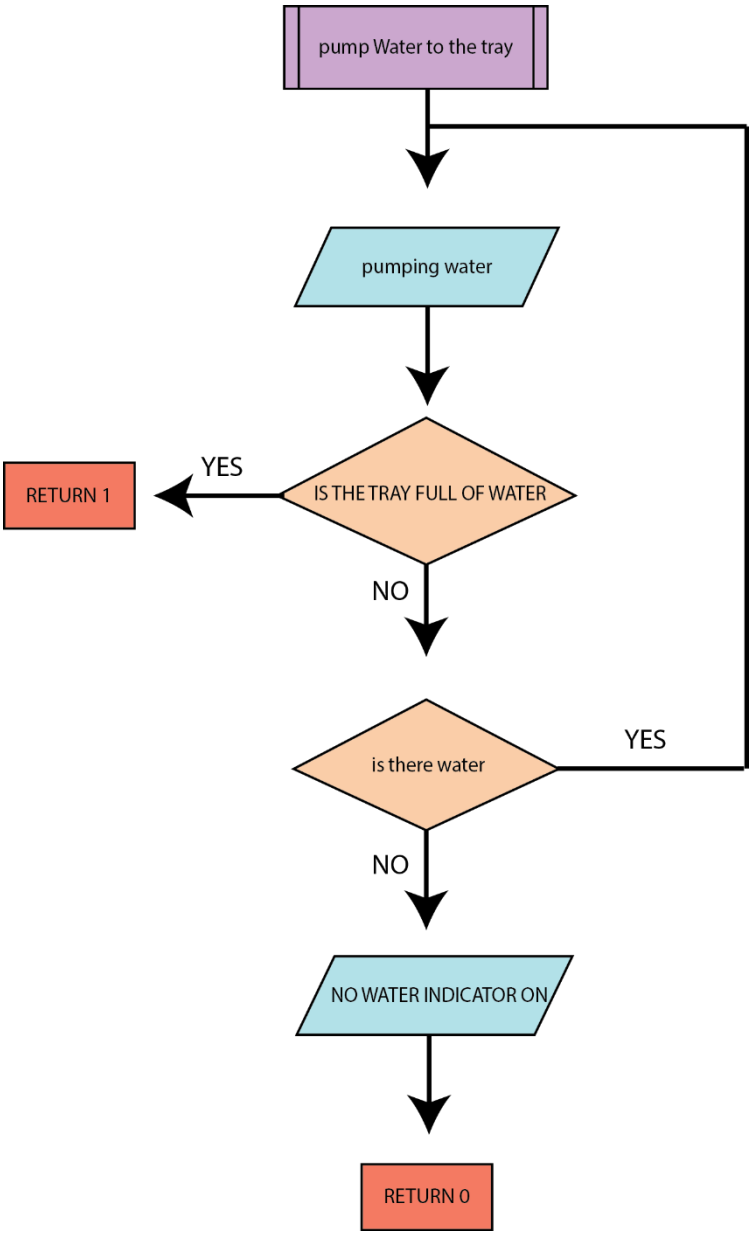
Εικόνα 1.1 Διάγραμμα ροής της λειτουργείας

Ο αλγόριθμος δημιουργίας παγοκύβων ξεκινά με την τοποθέτηση του δοχείου ψύξης νερού στη κατάλληλη θέση ώστε να γεμίσει με νερό. Στη συνέχεια, η αντλία λειτουργεί για καθορισμένο χρονικό διάστημα εκτός αν δεν υπάρχει νερό, οπότε απενεργοποιείτε και επιστρέφει στην InitialState. Αν υπάρχει νερό, ο συμπιεστής ενεργοποιείτε, η βαλβίδα αποσυμπίεσης απενεργοποιείτε για προκαθορισμένο χρόνο ανάλογα το μέγεθος παγοκύβων που επιθυμεί ο χρήστης. Τέλος ο συμπιεστής απενεργοποιείτε, η βαλβίδα αποσυμπίεσης ενεργοποιείτε και τα παγάκια μεταφέρονται στο καλάθι περισυλλογής.



Εικόνα 1.2 Διάγραμμα ροής Δημιουργίας Παγοκύβων

Ο αλγόριθμος χρήσης της αντλίας αρχικά την ενεργοποιεί και σε κάθε κύκλο ρολογιού ελέγχει αν υπάρχει νερό. Στη περίπτωση που δεν υπάρχει, απενεργοποιεί την αντλία και ενημερώνει τον χρήστη.



Εικόνα 1.3 Διάγραμμα ροής Αντλίας

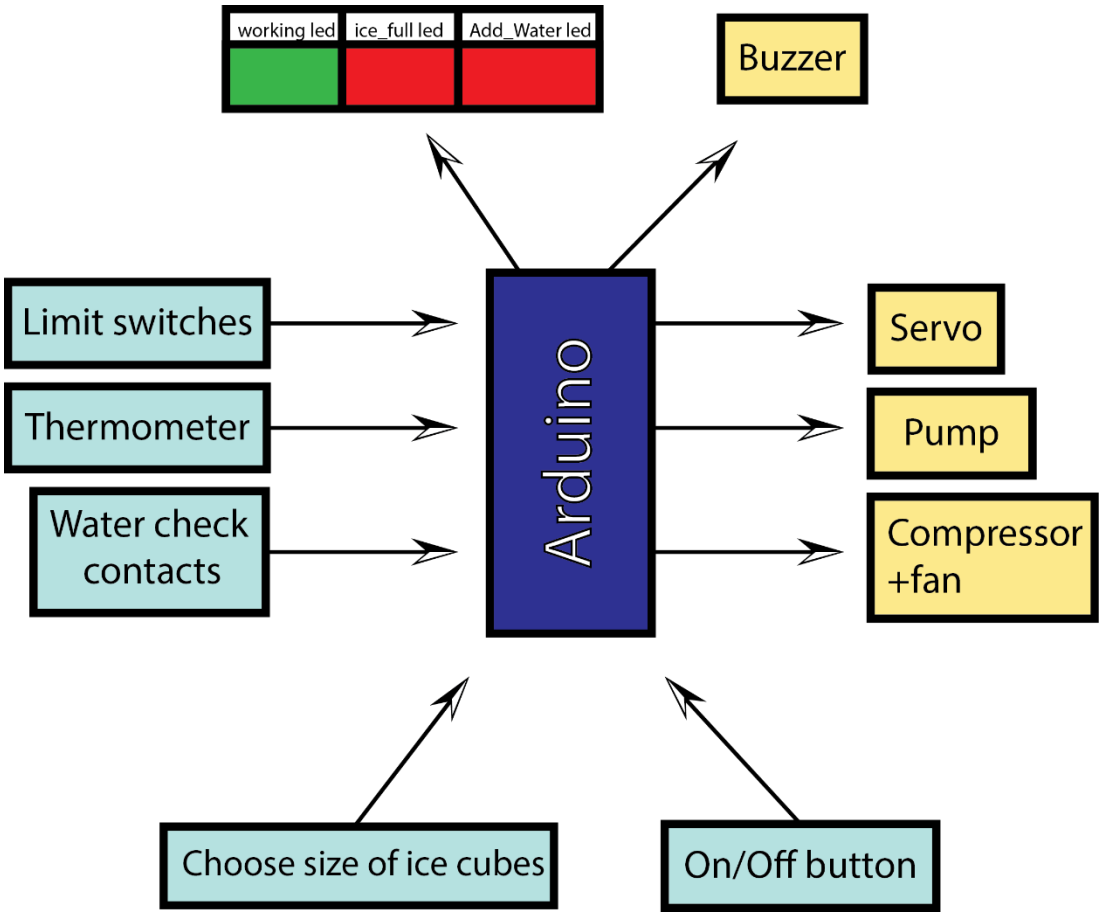
2. Εισαγωγικά

2.1 Γενικές παρατηρήσεις

Στη συγκεκριμένη διατριβή, παρατηρήθηκε ότι η ηλεκτρομαγνητική ακτινοβολία λόγω του συμπιεστή δημιουργούσε παρεμβολές στον μικροεπεξεργαστή και λόγω των ευαίσθητων κυκλωμάτων «καιγόταν». Λόγω αυτής της παρεμβολής απαραίτητη θεωρείται η χρήση μίας γειωμένης επιφάνειας αλουμινίου ώστε να γειώνονται οι παρεμβολές και να μην επηρεάζουν τον μικροεπεξεργαστή. Παρατηρήθηκε επίσης ότι στην αρχή λειτουργίας της αντλίας υπήρχε μια σημαντική πτώση τάσης, η οποία έκανε επανεκκίνηση στον μικροεπεξεργαστή με αποτέλεσμα την μη ορθή εκτέλεση του προγράμματος. Κάθε relay ελέγχεται από transistor ώστε να μην απορροφηθεί περισσότερο ρεύμα από τι μπορεί να διαθέσει ο μικροεπεξεργαστής. Παρατηρήθηκε ότι αν δεν καθαριστεί αρκετά καλά το flux (πάστα που βοηθά την κόλληση) μπορεί να προκαλέσει βραχυκλώματα που θα επηρεάσουν το κύκλωμα.

2.2 Επικοινωνία μικροελεγκτή με τα εξαρτήματα του συστήματος

Στο παρακάτω σχήμα φαίνονται οι συνδέσεις του μικροεπεξεργαστή με τα εξαρτήματα. Τα βέλη που κατευθύνονται στο Arduino είναι οι είσοδοι ενώ τα βέλη που ξεκινούν από αυτό είναι τα εξαρτήματα που ελέγχει.

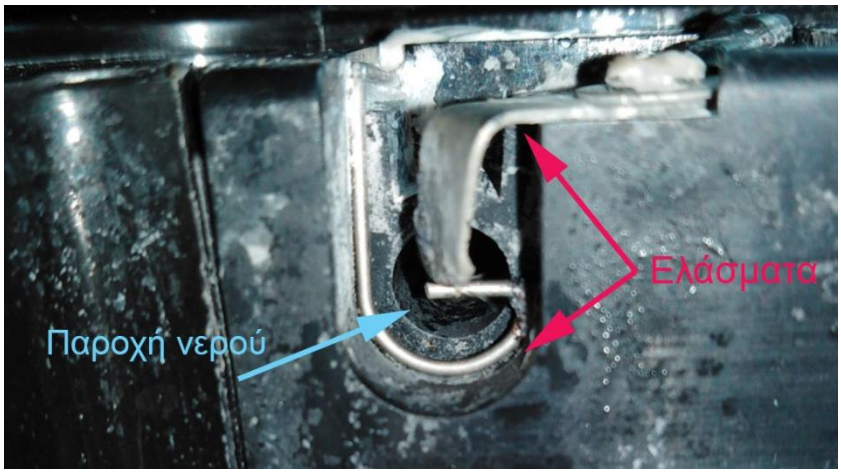


Εικόνα 2.1 Abstract σχέδιο κυκλώματος

2.3 Αισθητήρες που χρησιμοποιήθηκαν

- Αισθητήρας νερού:

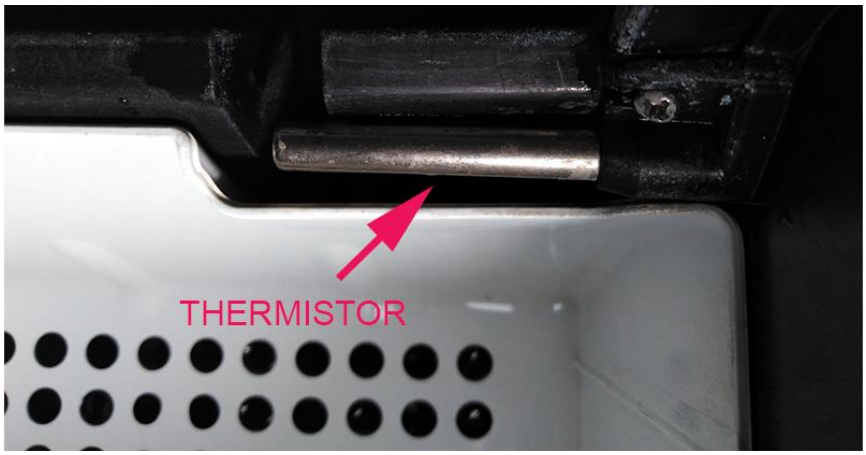
  Ο αισθητήρας αυτός αποτελείται από δύο ελάσματα τα οποία βρίσκονται αρκετά κοντά. Όταν διέρχεται από αυτόν τον αισθητήρα νερό, λόγω των ηλεκτρολυτών υπάρχει αντίσταση(της τάξης Mega Ohm) η οποία δημιουργείται στα δύο ελάσματα που προαναφέρθηκαν. Στη συνέχεια, ο μικροεπεξεργαστής εντοπίζει την αντίσταση αυτή, η οποία σηματοδοτεί την ύπαρξη νερού.



Εικόνα 2.2 Εικόνα Αισθητήρα Νερού

- Αισθητήρας θερμοκρασίας:

  Ο αισθητήρας αυτός είναι ένα θερμίστορ τύπου NTC το οποίο μετρά τη θερμοκρασία στην άκρη του δοχείου περισυλλογής παγοκύβων. Όταν η θερμοκρασία αυτή προσεγγίζει τους μηδέν βαθμούς κελσίου, το οποίο σημαίνει ότι κάποιο παγάκι είτε ακουμπάει είτε είναι αρκετά κοντά, σηματοδοτείται ότι το καλάθι είναι γεμάτο με παγάκια και η διαδικασία δημιουργίας παγοκύβων πρέπει να σταματήσει.

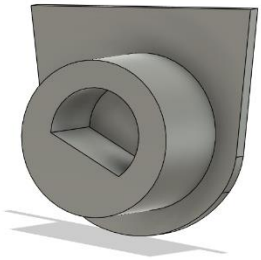


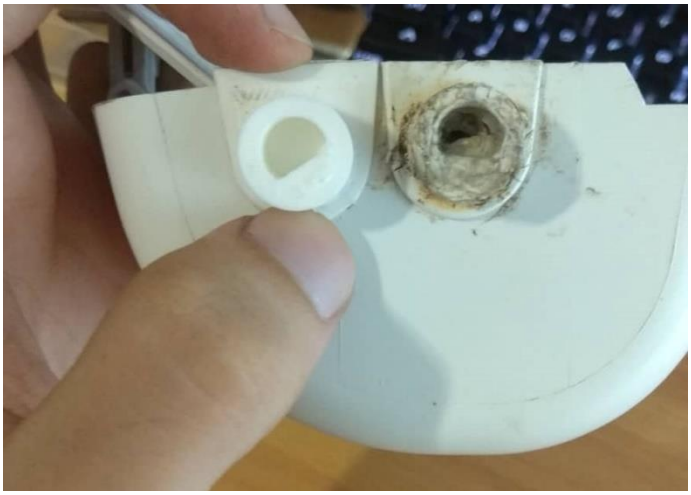
Εικόνα 2.3 Εικόνα Θερμίστορ

2.4 Μοτέρ για την κίνηση του δοχείου ψύξης νερού

- Servo 360°

  Το μοτέρ που κινεί το δοχείο αυτό είναι ένα servo τριακοσίων εξήντα μοιρών, το οποίο δέχεται ένα παλμικό σήμα από τον μικροεπεξεργαστή που καθορίζει την κατεύθυνση και την ταχύτητα. Το μοτέρ αυτό αντικατέστησε το προϋπάρχον, το οποίο ήταν ένα Ac-motor που παρέμενε στις τερματικές θέσεις έως ότου άλλαζε κατεύθυνση από το μαγνητικό πεδίο. Ωστόσο, αυτός ο μηχανισμός ήταν προβληματικός στο συγκεκριμένο σύστημα, καθώς η άσκηση πίεσης στις τερματικές θέσεις τις έφθειρε και τελικά έσπασε τμήμα του κάδου. Η βλάβη αναπληρώθηκε με τρισδιάστατα σχεδιασμένο και εκτυπωμένο εξάρτημα που παρατίθεται παρακάτω.



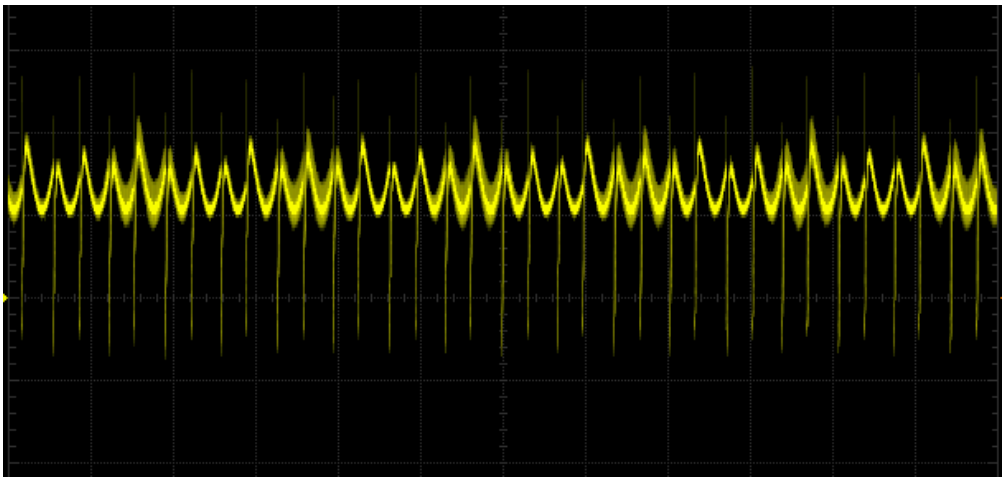


Εικόνα 2.4 Κάδος ψύξης νερού

3. Μελέτη εξαρτημάτων

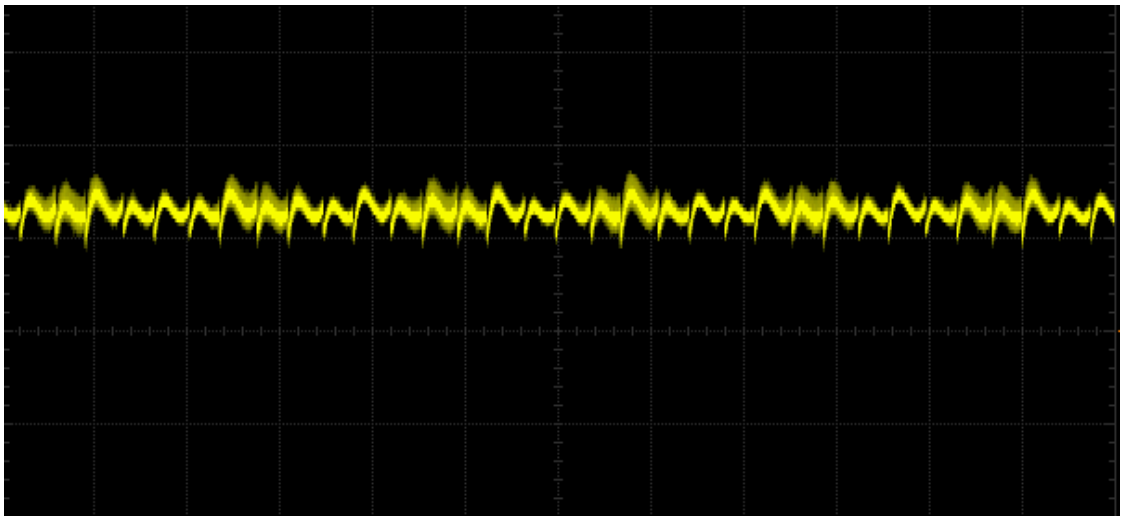
3.1 Μελέτη κυματομορφής αντλίας και βελτιστοποίηση

Λόγω της κατασκευής της αντλίας και συγκεκριμένα στο σημείο που ενώνονται οι συλλέκτες με τα καρβουνάκια, προκύπτουν περιοδικά στην κυματομορφή του ρεύματος σημεία μηδενισμού και έπειτα απότομης αύξησης του ρεύματος. Το γεγονός αυτό ενδέχεται να επηρεάσει το τροφοδοτικό που είναι power switching supply και να του μηδενίσει την τάση εξόδου.



Εικόνα 3.1 Κυματομορφή ρεύματος αντλίας

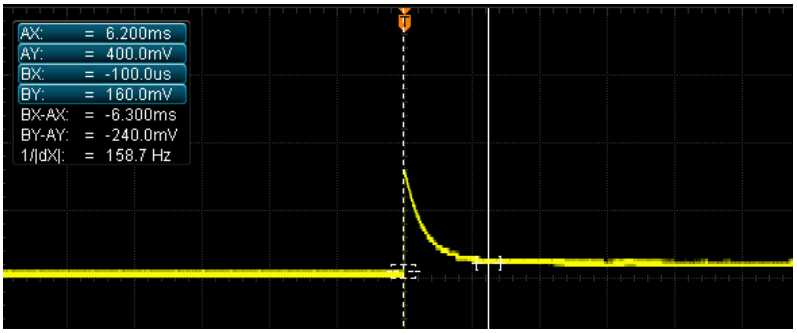
Για να ομαλοποιήθεί αυτή η απότομη μεταβολή του ρεύματος τοποθετήθηκε παράλληλα ένας πυκνωτής των 470μF ο οποίος επιλέχθηκε πειραματικά. Λόγω του ότι το θεμελιώδες εξάρτημα της αντλίας είναι ένα DC motor ενδέχετε να επιστρέψει τάση (kickback), για την μείωση της φθοράς του transistor τοποθετήθηκε μία δίοδος ώστε το ρεύμα αυτό να καταναλωθεί απ'τη δίοδο.



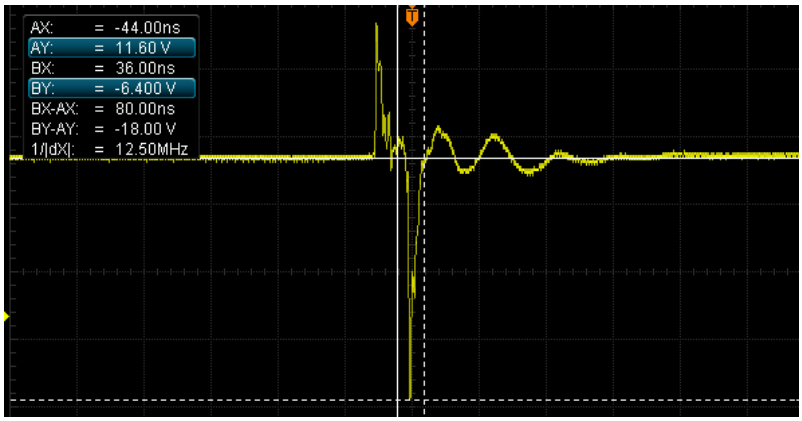
Εικόνα 3.2 Κυματομορφή ρεύματος αντλίας συνδεδεμένη με πυκνωτή και δίοδο παράλληλα

3.1 Υπολογισμός Πυκνωτή στην είσοδο του Μικροεπεξεργαστή

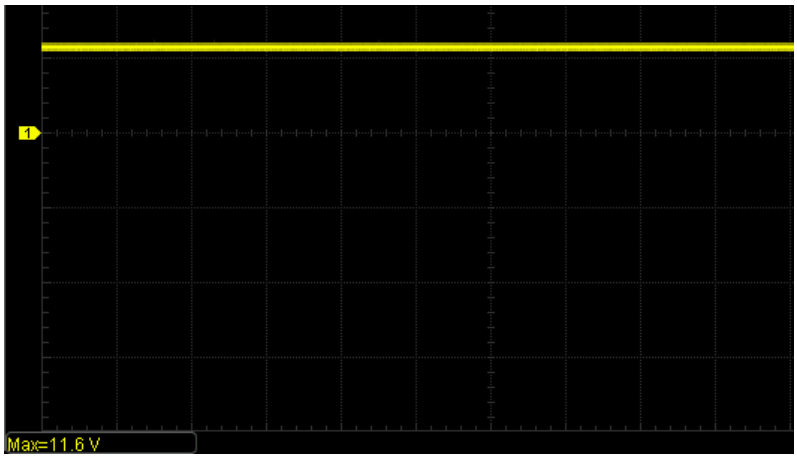
Παρακάτω παρατίθονται οι κυματομορφές ρεύματος της αντλίας και της τάσης του τροφοδοτικού. Παρατηρείται ότι για περίπου 6,3 ms καταναλώνει ένα μεγαλύτερο ρεύμα το οποίο δημιουργεί μία βύθιση στην τάση του τροφοδοτικού, όπως φαίνεται στην Εικόνα 3.4. Αυτή η μεταβολή της τάσης κάνει reset στο Arduino. Γι' αυτό το λόγο στην είσοδο τάσης του μικροεπεξεργαστή τοποθετήθηκε μια δίοδος και ένας πυκνωτής (Εικόνα 3.7) ώστε όταν γίνετε βύθιση, η τάση τροφοδοσίας του μικροεπεξεργαστή να είναι σταθερή.



Εικόνα 3.3 Κυματομορφή ρεύματος αντλίας

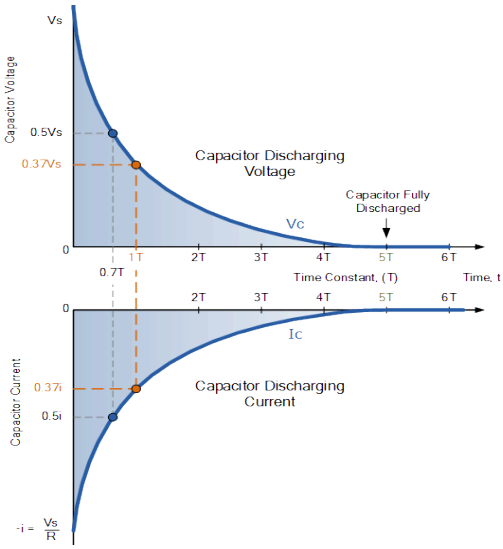


Εικόνα 3.4 Κυματομορφή τάσης τροφοδοσίας



Εικόνα 3.5 Κυματομορφή τάσης τροφοδοσίας μικροεπεξεργαστή

Για τον υπολογισμό του κατάλληλου πυκνωτή πρέπει να παρατηρηθεί η καμπύλη αποφόρτισης του. Φαίνεται ότι σε χρόνο 0.7T η τάση του πυκνωτή έχει μειωθεί κατά 50%. Η τάση τροφοδοσίας του Arduino ιδανικά πρέπει να κυμαίνεται από 12 έως 6 volt. Με δεδομένο ότι η τάση που παρέχεται σε αυτό είναι 12 V ο πυκνωτής πρέπει να μπορεί να κρατά την τροφοδοσία έως τα 6 V για τουλάχιστον 80ns (χρόνος βύθισης τάσης λόγω αντλίας). Αξιοσημείωτο είναι ότι για τον υπολογισμό αυτό μελετάται η ακραία περίπτωση, κατά την οποία ο μικροεπεξεργαστής καταναλώνει τη μέγιστη δυνατή τιμή ρεύματος που μπορεί να του παρέχεται μέσω του regulator, πράγμα που είναι αδύνατο στο συγκεκριμένο κύκλωμα και η κατανάλωση αυτή στην πραγματικότητα είναι υπερβολικά πιο μικρή. Γι' αυτό το λόγο η τιμή του πυκνωτή θα υπολογιστεί αποκλειστικά από την κυμματομορφή αποφόρτισης τάσης.



Εικόνα 3.6 Εκφόρτιση πυκνωτή

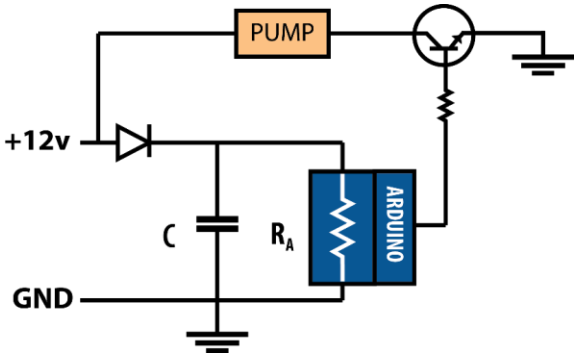
Αρχικά για τον υπολογισμό του πυκνωτή βρέθηκε πόσο είναι το μέγιστο ρεύμα που μπορεί να καταναλωθεί από το Arduino. Το ρεύμα αυτό είναι το μέγιστο που μπορεί να καταναλωθεί απ' το regulator το οποίο με την σειρά του παρέχει ρεύμα στο μικροελεγκτή. Η τιμή του είναι 1.3 A και η τάση που παρέχετε στο regulator στην ακραία περίπτωση είναι 6 V. Μέσω του νόμου του Ohm βρίσκεται η αντίσταση που θα βλέπει ως φορτίο ο πυκνωτής.

$$R_A = \frac{V}{I} = \frac{6}{1.3} = 4.61\ \Omega$$

Θεωρώντας το κύκλωμα ως RC και T=80ns χρόνος αποφόρτισης του πυκνωτή. Η τιμή του πυκνωτή θα υπολογιστεί από τον τύπο:

$$T = 0.7 \cdot R_A \cdot C \Rightarrow C = \frac{T}{0.7 \cdot R_A} = \frac{80ns}{0.7 \cdot 4.61\ \Omega} = 24.7\ nF$$

Για λόγους ασφαλείας η χωρητικότητα του πυκνωτή θα αυξηθεί και η τιμή του θα είναι 1000 μF.

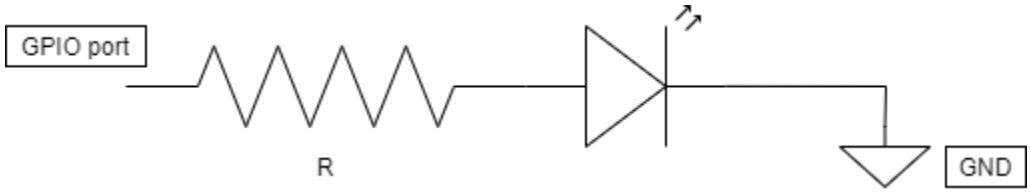


Εικόνα 3.7 Διάταξη Arduino Διόδου και πυκνωτή

3.2 Current limiting resistor LED

Για τον υπολογισμό των αντιστάσεων βρέθηκε η πτώση τάσης στο κάθε LED η οποία είναι 2.5V και το ρεύμα που είναι 3mA. Από τον νόμο του Ohm μπορεί να βρεθεί η αντίσταση που θα περιορίζει το ρεύμα:

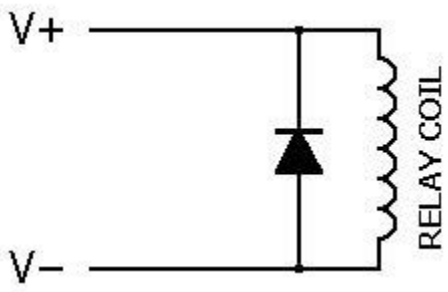
$$R = \frac{V_{GPIO} - V_F}{I_{LED}} = \frac{5 - 2.5}{0.003} = 830\Omega$$



Εικόνα 3.8 Διάταξη LED αντίστασης

3.3 Relay και Δίοδος

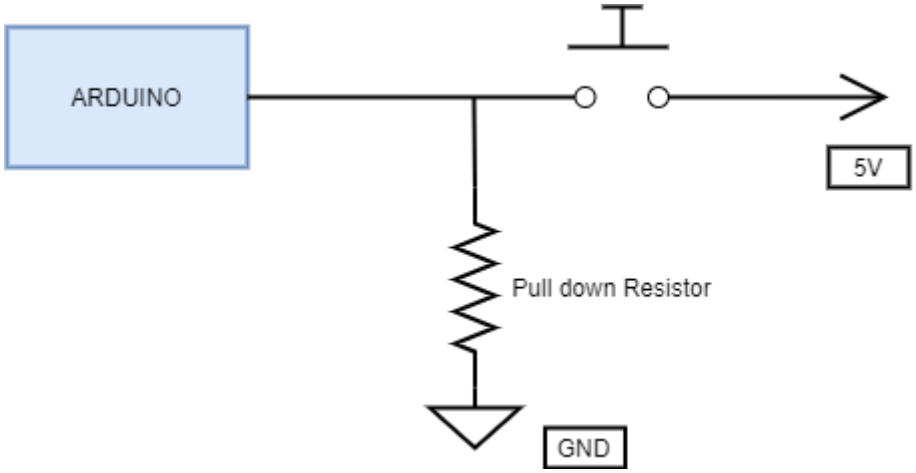
Το relay ή αλλιώς ηλεκτρονόμος είναι ένας ηλεκτρικός διακόπτης που ανοίγει και κλείνει ένα ηλεκτρικό κύκλωμα κάτω από τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος, στη συγκεκριμένη διατριβή τα GPIO Ports του Arduino. Αποτελείται από ένα πηνίο το οποίο όταν το διαρρέει ηλεκτρικό ρεύμα το παραγόμενο μαγνητικό πεδίο έλκει έναν οπλισμό που είναι μηχανικά συνδεδεμένος σε μια κινούμενη επαφή. Όπως προαναφέρθηκε όταν στο πηνίο του ηλεκτρονόμου υπάρχει τάση δημιουργείται μαγνητικό πεδίο. Όταν διακοπεί η τάση τότε το μαγνητικό πεδίο το οποίο υπήρχε γύρω του αντλείται από το πηνίο και δημιουργεί ένα ανάστροφο ρεύμα. Το ρεύμα αυτό περνάει μέσα από τη δίοδο και δεν επηρεάζει το υπόλοιπο κύκλωμα.



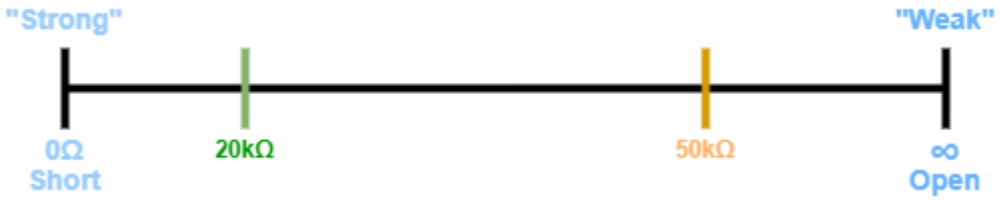
Εικόνα 3.9 Διάταξη relay διόδου

3.4 Pull-down Αντιστάσεις για τους διακόπτες

Στη συγκεκριμένη διατριβή χρησιμοποιήθηκαν διακόπτες Momentary-Push buttons όπου η συγκεκριμένη εντολή στον Μικροεπεξεργαστή δηλώνεται με λογικό «1». Αν στο διακόπτη αυτόν δεν χρησιμοποιηθεί αντίσταση ώστε το σήμα να γειώνετε, όταν δεν πιέζεται θα λαμβάνει σήματα από το γύρω ηλεκτρομαγνητικό θόρυβο το οποίο συνεπάγεται ασταθή συμπεριφορά του κυκλώματος. Γι' αυτό το λόγο τοποθετείται μια αντίσταση στο σήμα εισόδου και στη γείωση. Η αντίσταση αυτή δεν επιτρέπεται να είναι ούτε μηδενική (λόγω του ότι θα προκαλεί βραχυκύκλωμα στη πηγή και συνεπώς καταστροφή του regulator του Arduino) αλλά ούτε και άπειρη για τους λόγους που προαναφέρθηκαν. Στον επεξεργαστή, η ελάχιστη προτεινόμενη αντίσταση που μπορεί να τοποθετηθεί είναι 20 kΩ ενώ η μέγιστη είναι 50 kΩ. Όσο πλησιάζει η αντίσταση το μέγιστο όριο, τόσο θεωρείται πιο «αδύναμη» ενώ στην περίπτωση που πλησιάζει το ελάχιστο όριο θεωρείται «δυνατή». Η επιλογή της αντίστασης βρίσκετε πειραματικά.



Εικόνα 3.10 Διάταξη pull down resistors

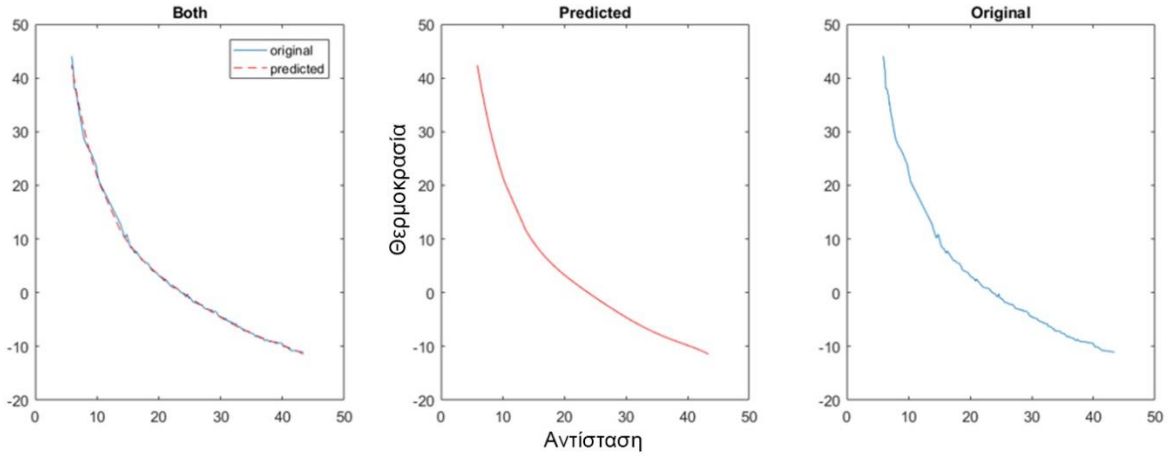


Εικόνα 3.11 Διάγραμμα pull down αντίστασης

3.5 Υπολογισμός καμπύλης αισθητήρα θερμοκρασίας αντίστασης πειραματικά

Λόγω του ότι η καμπύλη του αισθητήρα αυτού ήταν άγνωστη έπρεπε να βρεθεί πειραματικά. Στη συνέχεια έπρεπε να βρεθεί μια συνάρτηση που θα πλησιάζει αρκετά κοντά σε αυτή την καμπύλη, ώστε να μπορέσει ο μικροεπεξεργαστής να υπολογίζει τις θερμοκρασίες ανάλογα με την αντίσταση που θα έχει εκείνη την στιγμή ο αισθητήρας. Λόγω της μη γραμμικότητας της, η εύρεση της είναι αρκετά δύσκολη. Γι' αυτό τον λόγο υπολογίστηκε μέσω του Matlab με τεχνητή νοημοσύνη.

- Αρχικά έπρεπε να γίνουν μετρήσεις θερμοκρασίας και αντίστασης του θερμομέτρου. Για να υπάρχει μια ικανοποιητική ακρίβεια στις μετρήσεις χρησιμοποιήθηκαν 2 θερμόμετρα. Στη συνέχεια αφαιρέθηκαν κάποιες μετρήσεις οι οποίες υπήρχαν λόγω σφάλματος των οργάνων. Τέλος έγινε το γράφημα των μετρήσεων και βρέθηκε η εξίσωση η οποία τείνει στην αρχική καμπύλη των μετρήσεων.



Εικόνα 3.12 Διάγραμμα του αισθητήρα θερμοκρασίας Θερμοκρασία-Αντίσταση. Πειραματικά αποτελέσματα(Original) και η πολυονυμική εξίσωση που το προσομοιώνει (predicted)

Η εξίσωση είναι ένα πολυώνυμο 5^{ου} βαθμού:

$$P(R) = -0.0000039 \cdot R^5 + 0.0006 \cdot R^4 - 0.0351 \cdot R^3 + 1.026 \cdot R^2 - 15.7405 \cdot R^1 + 106.0322$$

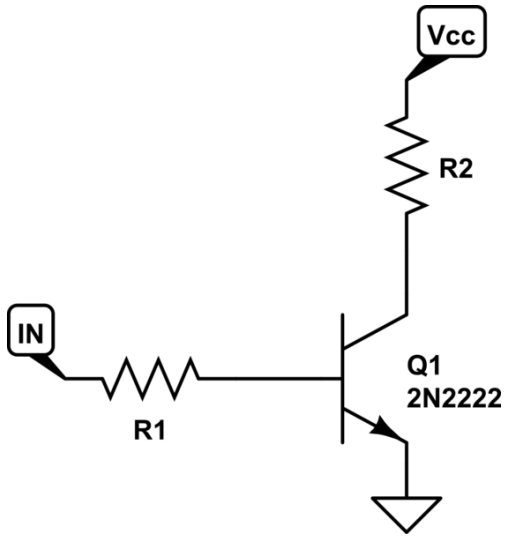
Για την εύρεση της εξίσωσης χρησιμοποιήθηκε η συνάρτηση polyfit του Matlab

```
6   R=data(:,1);
7   T=data(:,2);
8
9   %X = [ones(size(R,1),1) R];
10  %y = T;
11
12
13  p = polyfit(R,T,5);
14
15  predicted = polyval(p,R);
```

Εικόνα 3.13 Κώδικας Matlab

3.6 Υπολογισμός αντιστάσεων για τα transistors

Λόγω του ότι το κύκλωμα δεν έχει ιδιαίτερες απαιτήσεις ούτε στα ρεύματα, ούτε στις συχνότητες χρησιμοποιείται το 2n2222. Γενικά, το transistor λειτουργεί ως ενισχυτής. Όμως στη συγκεκριμένη περίπτωση το transistor χρησιμοποιείται ως διακόπτης με κατάσταση "ON" όταν η τάση στην είσοδο είναι υψηλή και "OFF" όταν είναι χαμηλή. Στα transistor υπάρχει μια περιοχή στην οποία όσο περισσότερο είναι το ρεύμα εισόδου στη βάση, τόσο μεγαλύτερο ρεύμα επιτρέπει να διαρρέεται μέσω αυτού. Στη συγκεκριμένη περίπτωση δε θα μελετηθεί αυτή η περιοχή, θα μελετηθεί μόνο στα Cut-off και στο Saturation. Μια τυπική σύνδεση του transistor φαίνεται παρακάτω όπου η τάση in συνδέεται στο GPIO port όπου R2 συμβολίζει το φορτίο.



Εικόνα 3.14 Διάταξη transistor αντίστασης

- **Υπολογισμός αντιστάσεως εισόδου R1**

  Για τον υπολογισμό της αντιστάσεως εισόδου βάσης του transistor θα θεωρηθεί ότι το transistor διαχειρίζεται το μέγιστο ρεύμα, το οποίο είναι 500mA. Η τάση τροφοδοσίας Vcc είναι 12V η οποία λόγω της πτώσης τάσης που υπάρχει στο transistor θα έχει μια πτώση τάσης κατά μέγιστο 1.2V αυτό είναι ανεκτό από τα στοιχεία που ελέγχει. Λόγω του ότι η τάση μειώνεται το ρεύμα θα αυξηθεί, το οποίο είναι ανεκτό διότι τα στοιχεία καταναλώνουν περίπου το μισό ρεύμα από τι μπορεί να διαχειριστεί το transistor. Η τάση στη βάση πριν την αντίσταση είναι η τάση του GPIO port του μικροελεγκτή που είναι 5V. Το datasheet του transistor δηλώνει ότι όταν το ρεύμα I_c είναι 500mA τότε το ρεύμα της βάσης I_B θα πρέπει να είναι 50mA, το μέγιστο όμως ρεύμα που μπορεί να παρέχει το Arduino είναι 40mA. Το V_F είναι η πτώση τάσης λόγου της διόδου που σχηματίζεται από την βάση προς τον εκπομπό και είναι ίση με 0.7V. Με τον νόμο του Ohm μπορεί να υπολογιστεί εύκολα η αντίσταση R_1.

$$R_1 = \frac{V_{GPIO} - V_F}{I_B} = \frac{5 - 0.7\,V}{40mA} = 107.5\Omega$$

3.7 Αισθητήρας Θερμοκρασίας-Νερού

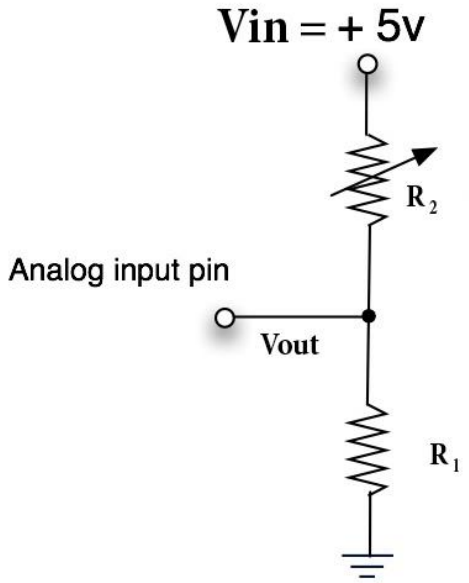
Ο αισθητήρας θερμοκρασίας τοποθετήθηκε σε διάταξη διαιρέτη τάσης όπου η μία αντίσταση R_1 είναι γνωστή. Λόγω του διαιρέτη τάσης είναι γνωστό ότι :

$$V_{OUT} = V_{IN} \cdot \left(\frac{R_2}{R_2 + R_1}\right)$$

Αν αυτή η εξίσωση λυθεί ώς προς R_2 :

$$R_2 = R_1 \cdot \left(\frac{V_{IN}}{V_{OUT}} - 1\right)$$

Οι αισθητήρες Θερμοκρασίας και Νερού ενώ ακολουθούν την αρχή διαφέρουν στην τιμή της γνωστής αντίστασης. Για τον αισθητήρα του νερού πρέπει να τοποθετηθεί μια μεγάλη αντίσταση R_1 για να υπάρχει πτώση τάσης που θα μπορεί να διαβάσει ο μικροεπεξεργαστής, διότι η αντίσταση του νερού είναι αρκετά μεγάλη (Προσοχή: αν είναι υπερβολικά μεγάλη η αντίσταση μπορεί να μη γειώνεται σωστά το σήμα και να επηρεάζεται από παρεμβολές των ηλεκτρομαγνητικών σημάτων). Αντίθετα η αντίσταση R_1 για τον αισθητήρα θερμοκρασίας πρέπει να είναι αρκετά μικρότερη ώστε να υπάρχει καλή ακρίβεια στην μετρήσεις.



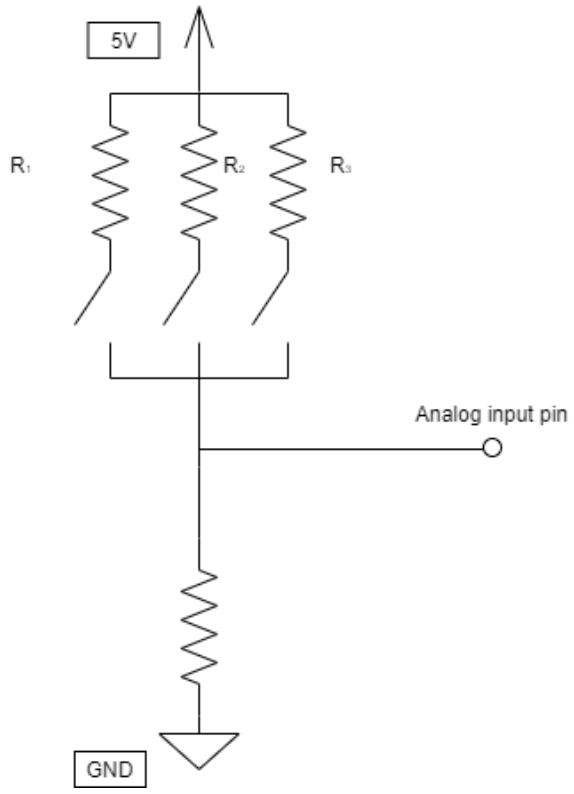
Εικόνα 3.15 Διάταξη διαιρέτη τάσης για αισθητήρες

3.8 Επιλογή μεγέθους Παγοκύβων

Λόγω του ότι χρησιμοποιούνται όλες οι ψηφιακές είσοδοι/έξοδοι του μικροεπεξεργαστή δεν είναι δυνατό να προστεθούν διακόπτες που θα επιλέγουν το μέγεθος των παγοκύβων. Γι' αυτό το λόγο θα χρησιμοποιηθεί μια αναλογική είσοδος του Arduino στην οποία θα συνδεθούν με την παρακάτω διάταξη τρεις διακόπτες με τρεις διαφορετικές αντιστάσεις. Όταν ο χρήστης επιλέξει ένα μέγεθος, τότε βραχυκυκλώνεται μία από αυτές με μια γνωστή αντίσταση και δημιουργείται πτώση τάσης, την οποία θα διαβάζει ο μικροεπεξεργαστής μέσω της αναλογικής εισόδου και θα υπολογίζει με τον νόμο του Ohm την αντίσταση όπου επιλέχθηκε, συνεπώς ποιο μέγεθος παγοκύβων. Με την μέθοδο αυτή δίνετε η δυνατότητα να χρησιμοποιούνται αρκετοί διακόπτες σε μόνο μία είσοδο ώστε να μην δεσμεύονται οι υπόλοιπες.

Ο τύπος εύρεσης αντιστάσεως είναι ο ίδιος με τον παραπάνω που υπολογίστηκε στη παράγραφο «Αισθητήρας Θερμοκρασίας-Νερού»:

$$R_2 = R_1 \cdot \left(\frac{V_{IN}}{V_{OUT}} - 1\right)$$

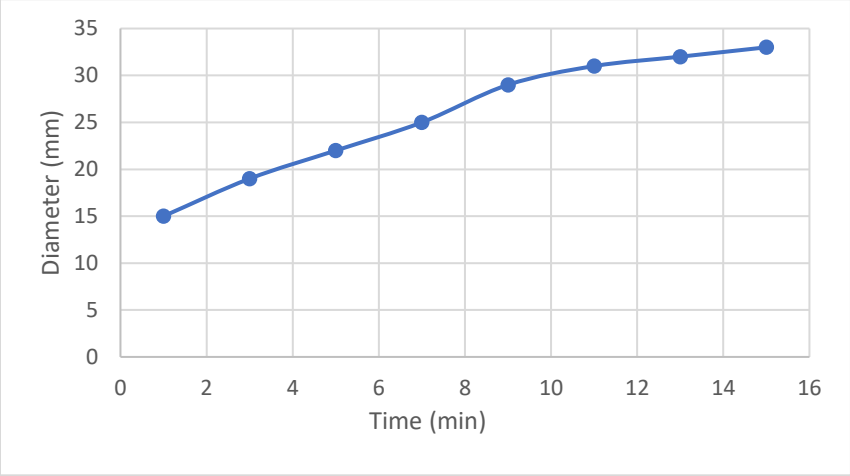


Εικόνα 3.16 Διάταξη τρειών κουμπιών επιλογής

3.9 Μέγεθος Παγοκύβων

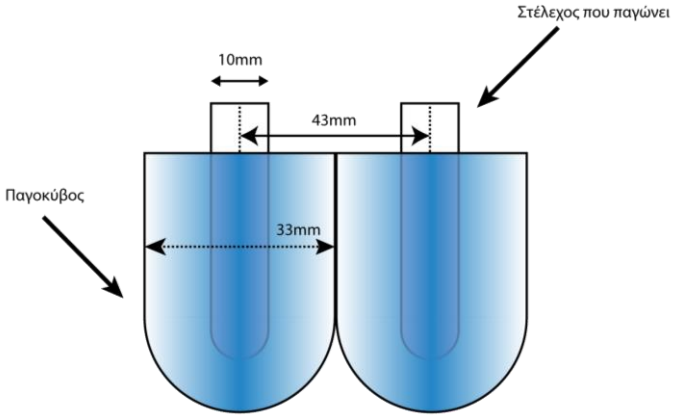
Έπειτα από μετρήσεις του μεγέθους παγοκύβων ανάλογα την ώρα που παγώνουν, παρατηρήθηκε ότι οι παγοκύβοι τείνουν να αυξάνονται με λογαριθμική σχέση. Το γεγονός αυτό είναι αναμενόμενο διότι καθώς αυξάνεται η διάμετρός απομακρύνεται το στέλεχος που ψύχετε απ' το υγρής μορφής νερό. Παρακάτω παρατίθονται οι μετρήσεις και το διάγραμμα Χρόνου-Διαμέτρου:

Minutes	Diameter
15 min	33 mm
13 min	32 mm
11 min	31 mm
9 min	29 mm
7 min	25 mm
5 min	22 mm
3 min	19 mm
1 min	15 mm



1η Παρατήρηση: Λόγω του ότι το δοχείο συλλογής νερού επαναχρησιμοποιεί το νερό από τα λιωμένα παγάκια, η θερμοκρασία του νερού μειώνεται με αποτέλεσμα το νερό να παγώνει πιο γρήγορα. Συμπερασματικά τα παραπάνω αποτελέσματα δηλώνουν την γενική εκτύλιξη της διαδικασίας και όχι τους χρόνους που χρειάζεται για να παράγει ένα παγοκύβο συγκεκριμένης διαμέτρου.

2η Παρατήρηση: Η μέγιστη διάμετρος παγοκύβου είναι συγκεκριμένη λόγω κατασκευής της παγομηχανής και δεν πρέπει να ξεπεραστεί διότι ενδέχεται να προκαλέσει βλάβη στο μηχάνημα.



Εικόνα 3.17 Σχήμα παγοκύβων και αποστάσεων

4. Κώδικας Προγραμματισμού

4.1 Συνάρτηση delayForMinutesCompressor ()

Η συνάρτηση delay() που χρησιμοποιεί το Arduino αδρανοποιεί το σύστημα και δεν επιτρέπει να εκτελεστεί άλλη εντολή μέχρι να τερματιστεί. Για τον λόγο αυτό δημιουργήθηκε μία συνάρτηση που δέχεται μια μεταβλητή float ως όρισμα, η οποία χρησιμοποιεί το εσωτερικό ρολόι του μικροεπεξεργαστή ώστε να περιμένει για το απαιτούμενο χρονικό διάστημα ελέγχοντας αν πατήθηκε ο διακόπτης παύσης.

```
void delayForMinutesCompressor(float minutes){
     Serial.print("minutes of delay = ");
     Serial.println(minutes);
     unsigned long StartTime = millis();
     unsigned long CurrentTime = millis();
     unsigned long ElapsedTime=0;
     int LastStateDe=State;
     //Serial.println("LastStateDe");
     //Serial.println(LastStateDe);

     while(ElapsedTime<(minutes*60000)){

     CurrentTime = millis();
     ElapsedTime = CurrentTime - StartTime;
     Serial.print("ElapsedTime : ");
     Serial.println(ElapsedTime);

       if(debounced()!=LastStateDe){
         Serial.println("Button pressed I am out");
         break;
       }
     }

     Serial.println("end of delay");
}
```

4.2 2.2 Συνάρτηση readTemp()

Η συνάρτηση αυτή είναι τύπου double και χρησιμοποιεί την εξίσωση που υπολογίστηκε παραπάνω (Υπολογισμός καμπύλης αισθητήρα θερμοκρασίας με αντίσταση). Μέσω του διαιρέτη τάσης υπολογίζεται η αντίσταση και στη συνέχεια μέσω της εξίσωσης η θερμοκρασία.

```
double readTemp(){
  VRT = analogRead(ThermistorPin);   //Acquisition analog value of VRT
  VRT = (5.00 / 1023.00) * VRT;      //Conversion to voltage
  RT = (R*(VCC/VRT - 1.0))/1000;           //finding the uknown resistor

  double rPowerFive=pow(RT, 5);   // r^5
  double rPowerFour=pow(RT, 4);   // r^4
  double rPowerThree=pow(RT, 3);  // r^3
  double rPowerTwo=pow(RT, 2);    // r^2

  TX =-0.00000398985138208*rPowerFive+0.00059799669742102*rPowerFour.....

  Serial.print("| Temperature");
  Serial.print("\t| ");
  Serial.print(TX);
  Serial.print("C |\t ");
  Serial.print(RT);
  Serial.println(" kΩ|\t\t ");
  return TX;
}
```

4.3 Συνάρτηση checkIfCartridgeIsFull()

Η συνάρτηση αυτή είναι τύπου Boolean και χρησιμοποιεί την παραπάνω συνάρτηση «readTemp()» ώστε όταν η θερμοκρασία είναι μικρότερη από την voudou μεταβλητή tempValueforFull να επιστρέψει true που υποδηλώνει ότι το δοχείο με τα παγάκια έχει γεμίσει.

```
boolean checkIfCartridgeIsFull(){

    if(readTemp()<=tempValueforFull){
      Serial.println("Cartridge Is Full");
      digitalWrite(led_IceFull,HIGH);    //ICE FULL light indicator on
      fullCartiegeSound();
      return true;
    }
    else{
      return false;
    }
}
```

4.4 Συνάρτηση readButtonSelect()

Προκυμμένου να γίνει η επιλογή μεγέθους παγοκύβου η συνάρτηση αυτή ελέγχει την τάση στην αναλογική είσοδο και με τον νόμο του Ohm υπολογίζει την αντίσταση η οποία τοποθετείται, ανάλογα με το ποιο κουμπί πατήθηκε από τον χρήστη. Εν συνεχεία αλλάζει την μεταβλητή sizeOfIceCube που υποδηλώνει το μέγεθος των παγοκύβων.

```
void readButtonSelect(){
  VRs = analogRead(SelectorButtons);     //Acquisition analog value of VRs
  VRs = (5.00 / 1023.00) * VRs;          //Conversion to voltage
  Rs = (R1*(Vsupply/VRs - 1.0))/1000;    //finding the uknown resistor

  if(Rs>0.5 && Rs<1.5){
    sizeOfIceCube=1;
    selectSound();
  }
  if(Rs>8 && Rs<12){
    sizeOfIceCube=2;
    for(int i=0; i<=1; i++){
      selectSound();
    }
  }
  if(Rs>60 && Rs<130){
    sizeOfIceCube=3;
    for(int i=0; i<=2; i++){
      selectSound();
    }
  }
```

4.5 Συνάρτηση toTheEndRight() - toTheEndLeft()

Οι συναρτήσεις αυτές ευθύνονται για την κίνηση του δοχείου ψύξης νερού. Ελέγχουν μέσω των τερματικών διακοπτών αν το δοχείο αυτό βρίσκετε στην σωστή θέση. Σε περίπτωση βλάβης κατά την οποία το μοτέρ δεν μπορεί να κινηθεί, για λόγους ασφαλείας πηγαίνει σε μία κατάσταση σφάλματος, όπου απενεργοποιούνται όλα τα εξαρτήματα και σημαίνει περιοδικό συναγερμό ώστε να ενημερώσει το χρήστη.

```
void toTheEndRight(){
  unsigned long StartTime = millis();
  unsigned long CurrentTime = millis();
  unsigned long ElapsedTime=0;

  Serial.println("Going Right");
  int rightSwitchState = digitalRead(rightSwitch);
  Serial.println(rightSwitchState);

  if(rightSwitchState==LOW){
  do{
    CurrentTime = millis();
    ElapsedTime = CurrentTime - StartTime;

    delay(100);
    rightSwitchState = digitalRead(rightSwitch);
    Serial.println("moving");
    trayMotor.writeMicroseconds(speedServoToTheRightMicrosecond);  //turn right

    if(ElapsedTime>maximumTimeForMoving){
      ErrorServoMove();
    }
  }while (rightSwitchState==LOW);
  trayMotor.writeMicroseconds(stopServo); //stop
  }
}


void ErrorServoMove(){
  trayMotor.writeMicroseconds(stopServo); //stop
  digitalWrite(valve,HIGH);      //heating valve off
  digitalWrite(Compressor,LOW); //Compressor On

  while(1){
    tone(buzzer,1000);
    delay(4000);
    Serial.println("ERROR WITH SERVO");
    digitalWrite(led_Work,HIGH);
    digitalWrite(led_IceFull,HIGH);
    digitalWrite(led_AddWater,HIGH);
    noTone(buzzer);
    delay(1000);
    digitalWrite(led_Work,LOW);
    digitalWrite(led_IceFull,LOW);
    digitalWrite(led_AddWater,LOW);
  }
}
```

4.6 Συνάρτηση checkNoWater()

Η συνάρτηση αυτή είναι υπεύθυνη να αναγνωρίσει αν υπάρχει νερό. Στη περίπτωση που υπάρχει επιστρέφει true. Αλλιώς επιστρέφει false με τα ανάλογα μηνύματα.

```
boolean checkNoWater(){

    float waterConductivity = analogRead(waterSensor);
    Serial.println(waterConductivity);
    if(waterConductivity<990){
      return true;
    }
    else{
     digitalWrite(led_AddWater,HIGH);
     Serial.println("NO WATER, ADD WATER PLEASE");
     return false;
    }
}
```

4.7 Συνάρτηση fillTheTrayWithWater()

Η συνάρτηση fillTheTrayWithWater() γεμίζει το δοχείο ψύξης με νερό. Χρησιμοποιώντας την παραπάνω συνάρτηση «checkNoWater()» ελέγχει την αντλία ώστε να μην λειτουργεί χωρίς νερό και κατά συνέπεια φθαρθεί.

```
boolean fillTheTrayWithWater(){
    unsigned long StartTime = millis();
    unsigned long CurrentTime = millis();
    unsigned long ElapsedTime=0;

    Serial.println("Fill the tray with water");
    Serial.println("Going to the correct possition");
    toTheEndLeft();
    Serial.println("Water pump ON");
    digitalWrite(waterPump,HIGH); //water pump on
    delay(700);
    Serial.println("check conductivity after the delay");

    while((ElapsedTime>(waterPumpTime+100) || ElapsedTime<(waterPumpTime-100))){

     if(checkNoWater()==false){
       digitalWrite(waterPump,LOW);  //water pump off
       Serial.println("Water pump OFF");
       Serial.println("No Water BREAK");
       waterFullSound();
       return false;
     }

    CurrentTime = millis();
    ElapsedTime = CurrentTime - StartTime;
    Serial.print("ElapsedTime : ");
    Serial.println(ElapsedTime);
    }

   digitalWrite(waterPump,LOW);  //water pump off
   Serial.println("Water pump OFF");
   Serial.println("End of filling water");
```

4.8 Συνάρτηση algorithmOfIcecubes ()

Η συνάρτηση αυτή είναι ο αλγόριθμος σχηματισμού παγοκύβων.

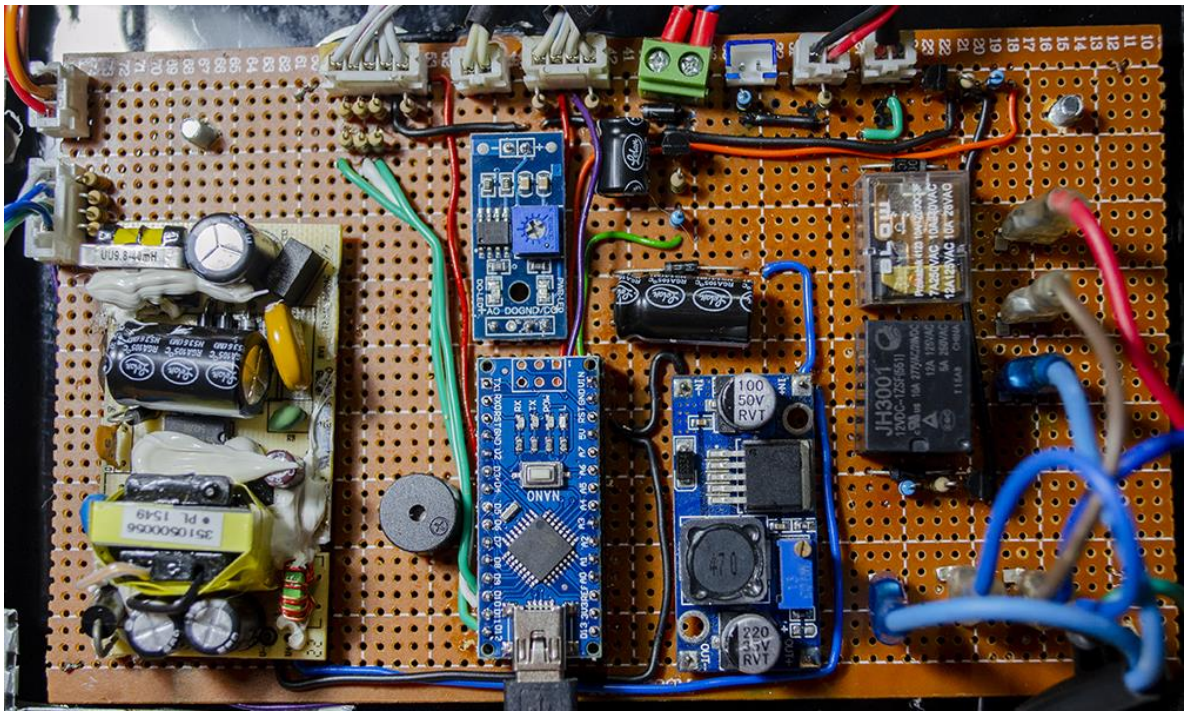
```
void algorithmOfIcecubes(float minutesForCubeSize){

  digitalWrite(valve,LOW);       //heating valve off
  Serial.println("Vlave OFF");
  delay(500);

  digitalWrite(Compressor,HIGH); //Compressor On
  Serial.println("Compressor ON");

  delayForMinutesCompressor(minutesForCubeSize); //Waiting to Froze the Ice Cubes

  digitalWrite(Compressor,LOW); //Compressor Off
  Serial.println("Compressor OFF");

  Serial.println("Emptying water");
  toTheEndRight(); //empty water

  delay(EmptyingWaterDelay); //wait to empty water
  Serial.println("Droping IceCubes");
  digitalWrite(valve,HIGH);      //heating valve off
  delayForMinutes(ValveDealayToHeat); //wait to heat the cubes to fall
  Serial.println("empty Icecubes from Cartrige");
  EmptyIceCubes();
```

4.9 Συνάρτηση MakeIceCubes ()

Η συνάρτηση αυτή συνδέει όλες τις παραπάνω συναρτήσεις ώστε να λειτουργεί σωστά η παγομηχανή

```
void MakeIceCubes(int sizeOfIceCube){

 int CartrigeFull = checkIfCartridgeIsFull(); //If Cartridge is not full proceed
 if (CartrigeFull){
  State=HIGH ;
  return;
 }

 int TrayFilled = fillTheTrayWithWater(); //If there is water proceed
 if (!TrayFilled){
  State=HIGH ;
  return;
 }

 Serial.println("Making IceCubes");
  switch (sizeOfIceCube) {

   case 1: //mini cubes
      Serial.println("Making Ice cubes number 1");
      algorithmOfIcecubes(delayForMiniCubes);
      break;

    case 2: //middle cubes
      Serial.println("Making Ice cubes number 2");
      algorithmOfIcecubes(delayForMiddleCubes);
      break;

    case 3: //mega cubes
      Serial.println("Making Ice cubes number 3");
      algorithmOfIcecubes(delayForMegaCubes);
      break;
```

5. Κατασκευή πλακέτας

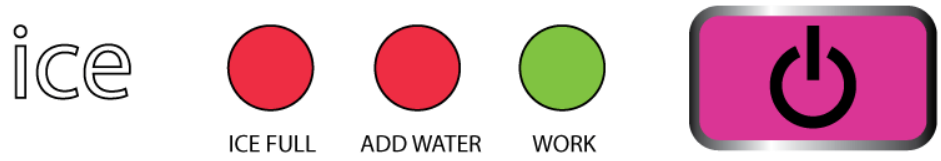


- Για την σύνδεση των εξαρτημάτων με την πλακέτα χρησιμοποιήθηκαν JST XH connectors.

- Η αντλία συνδέετε με ακροφύσια CT04 και με κλέμα.

Παρατηρήσεις : Για την κατασκευή της πλακέτας στις κολλήσεις χρησιμοποιήθηκε flux. Σε μερικά σημεία δεν είχε καθαριστεί καλά, κατά συνέπεια προκαλούσε βραχυκυκλώματα μη φανερής αιτίας.

6. Indications

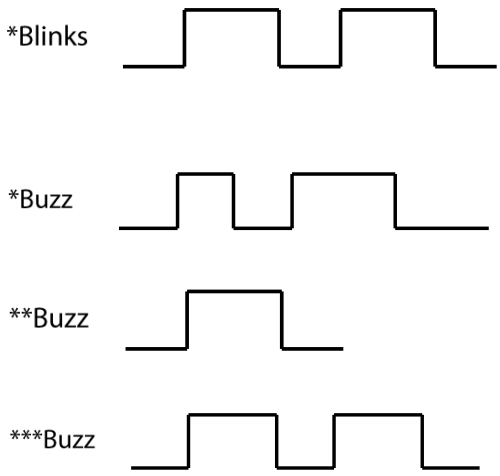


Normal Led Inications

Ice Full LED	Add Water LED	Work LED	Buzzer	Condition
OFF	OFF	ON	-	Making Ice Cubes
OFF	ON	OFF or ON	*buzz	Need Water
ON	OFF	OFF or ON	**buzz	The Tray is full of IceCubes
OFF	OFF	OFF	-	OFF or STBY

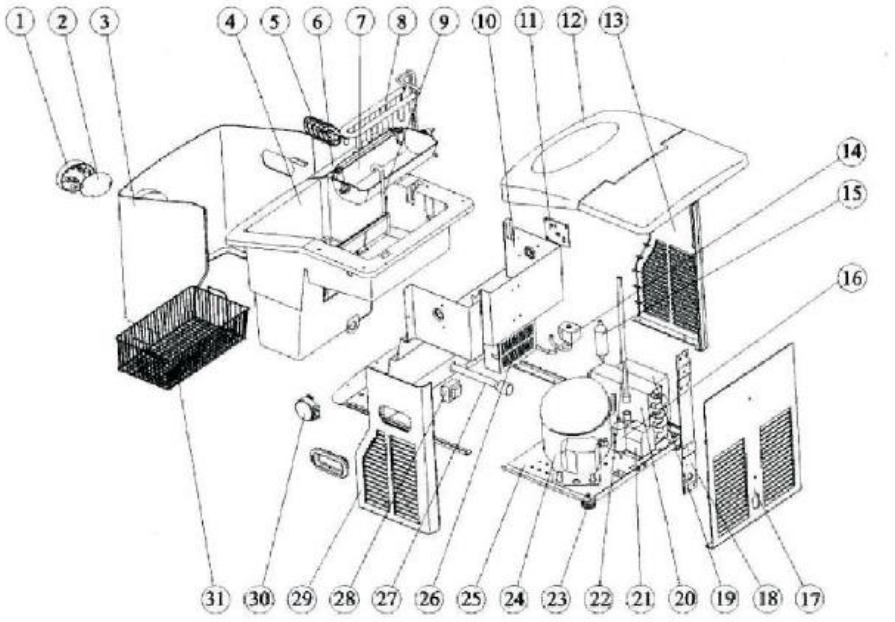
Abnormal Led Inications

Ice Full LED	Add Water LED	Work LED	Buzzer	Condition
*Blinks	*Blinks	*Blinks	***buzz	The Ice Tray cant Move

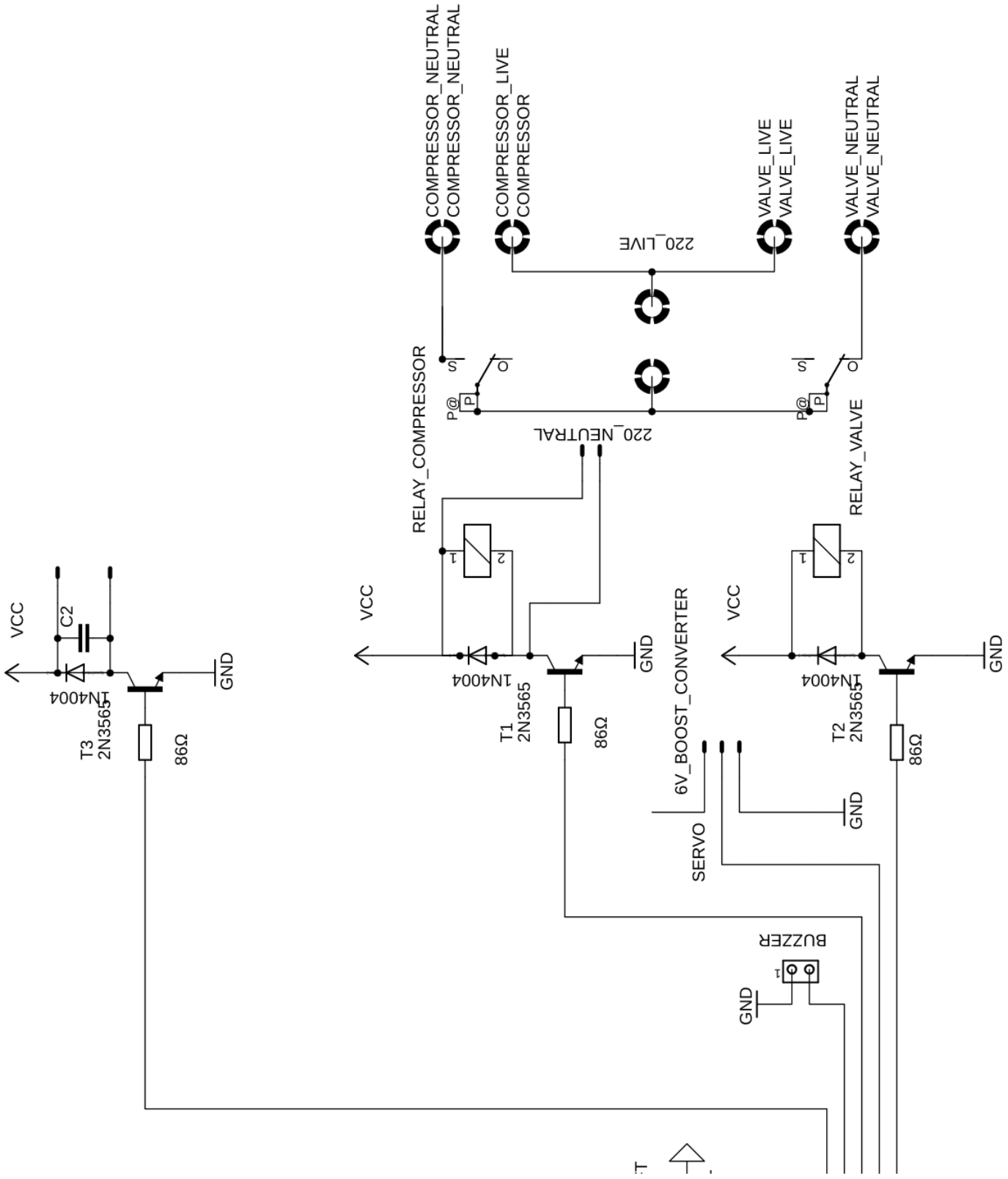


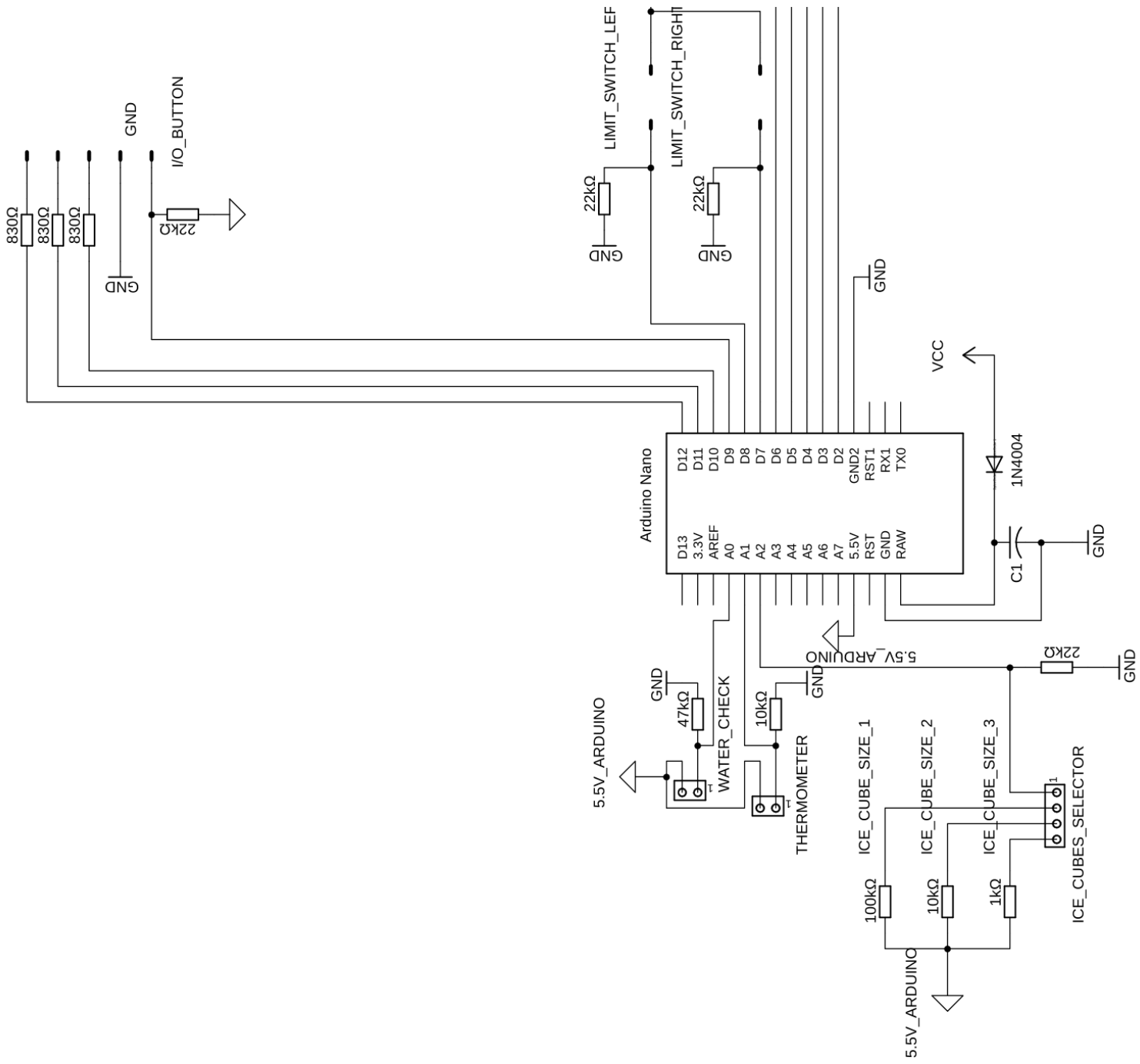
7. Μέρη παγομηχανής

1. Κουμπιά ελέγχου
2. Led indicators
3. Θήκη του πάνελ
4. Πλαστική βάση
5. Χερούλι
6. Φτυάρι πάγου
7. Δοχείο ψύξης νερού
8. Evaporator
9. Σωλήνας νερού
10. Μεταλλικές γωνίες
11. Τερματικοί διακόπτες
12. Καπάκι με διαφανές παράθυρο
13. Αριστερό πλαστικό panel
14. Φίλτρο
15. Σωληνοειδές
16. Μεταλλική βάση συμπιεστή
17. Πίσω πλαστικό panel
18. Μεταλλική γωνία στήριξης
19. Ανεμιστήρας
20. Ψύκτρα
21. Αντλία νερού
22. Σωλήνας νερού
23. Πλαστικά πόδια
24. Συμπιεστής
25. Μεταλλική βάση
26. Θήκη του μοτέρ
27. Σωλήνας νερού
28. Βάση στήριξης για το μοτέρ
29. Δεξί πλαστικό panel
30. Servo μοτέρ
31. Καλάθι παγοκύβων



8. Σχηματικό κυκλώματος




830Ω
830Ω
830Ω
GND
I/O_BUTTON
22kΩ
GND
22kΩ
LIMIT_SWITCH_LEF
GND
22kΩ
LIMIT_SWITCH_RIGHT
GND
GND
VCC
1N4004
Arduino Nano
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
GND2
RST1
RX1
TX0
D13
3.3V
AREF
A0
A1
A2
A3
A4
A5
A6
A7
5.5V
RST
GND
RAW
C1
GND
5.5V_ARDUINO
22kΩ
GND
GND
47kΩ
WATER_CHECK
10kΩ
GND
5.5V_ARDUINO
THERMOMETER
ICE_CUBE_SIZE_1
ICE_CUBE_SIZE_2
ICE_CUBE_SIZE_3
100kΩ
10kΩ
1kΩ
ICE_CUBES_SELECTOR
5.5V_ARDUINO

9. ΣΥΜΠΕΡΑΣΜΑΤΑ

Εν κατακλείδι η μηχανή παγοκύβων λειτουργεί ικανοποιητικά, καθώς υπάρχουν αρκετοί αλγόριθμοι που την προστατεύουν από πιθανά προβλήματα και κινδύνους. Ο κώδικας επικοινωνεί επαρκώς με την κονσόλα του IDE και ενημερώνει το χρήστη για κάθε βήμα που εκτελεί, με αυτόν τον τρόπο είναι πολύ εύκολη η ανίχνευση λαθών. Ο αλγόριθμος της θερμοκρασίας προσεγγίζει την πραγματική θερμοκρασία. Ο μικροεπεξεργαστής δεν είναι κολλημένος στην πλακέτα και οποιοδήποτε πρόβλημα μπορεί να αντιμετωπιστεί με μεγαλύτερη ευκολία. Στην μηχανή αυτή προστέθηκε η επιλογή του μεγέθους παγοκύβων ώστε να διευκολύνει το χρήστη. Ο αισθητήρας που ελέγχει την ύπαρξη νερού στην αντλία λειτουργεί ικανοποιητικά όμως λόγω της υψηλής αντίστασης του νερού υπάρχουν σφάλματα και ενδέχεται να υπάρχει νερό και να μην το αναγνωρίσει. Μια μελλοντική βελτίωση είναι να τοποθετηθεί ένας ενισχυτής ώστε να υπάρχει ενίσχυση αυτού του σήματος για πιο ακριβή αποτελέσματα, όπως και να τοποθετηθούν Led για την ένδειξη του μεγέθους των παγοκύβων.

10. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Καλαϊτζάκη Κ., Κουτρούλη Ε. "Ηλεκτρικές Μετρήσεις και Αισθητήρες", 2010

[2] Ανάλυση και σχεδίαση αναλογικών ολοκληρωμένων κυκλωμάτων

[3] Datasheet transistor 2n2222

[4] Επεξεργαστής ATMEGA328P

[5] https://www.arduino.cc/reference/en/language/functions/time/millis/