ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

# ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

## «Σύστημα πλήρωσης χημικών υγρών με υπολογισμό μεταβαλλόμενης ροής»


**Του φοιτητή**

**Νικήσανλη Νικόλαου**

**Αρ. Μητρώου: 52112m**

**Επιβλέπων**

**Ονοματεπώνυμο: Χατζόπουλος Αργύριος**

**Βαθμίδα: Επίκουρος Καθηγητής**


**Ημερομηνία 29-2-2024**

Τίτλος Δ.Ε.: Σύστημα πλήρωσης χημικών υγρών με υπολογισμό μεταβαλλόμενης ροής
Κωδικός Δ.Ε.: 23285
Ονοματεπώνυμο φοιτητή: Νικήσανλης Νικόλαος
Ονοματεπώνυμο εισηγητή: Χατζόπουλος Αργύριος
Ημερομηνία ανάληψης Δ.Ε.: 26-10-2023
Ημερομηνία περάτωσης Δ.Ε.: 29-2-2024

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



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

# Πρόλογος

Το σύστημα αυτό έρχεται να λύσει το πρόβλημα που υπάρχει σε εργοστάσια παραγωγής χημικών υγρών κατά την διαδικασία της συσκευασίας. Ειδικά στα μικρά εργοστάσια παραγωγής χημικών υγρών στα οποία οι ποσότητες παραγωγής είναι περιορισμένες η διαδικασία συσκευασίας γίνεται με χειροκίνητο τρόπο. Αυτό κάνει την διαδικασία αργή και αναξιόπιστη διότι τις περισσότερες φορές τοποθετείται μεγαλύτερη ποσότητα από την απαιτούμενη σε κάθε δοχείο. Οπότε θα πρέπει να κατασκευαστεί ένα σύστημα που θα μπορεί να συσκευάσει τα δοχεία σε μικρότερο χρόνο, χωρίς να απαιτεί την συνεχόμενη προσοχή του προσωπικού και θα επιτυγχάνεται καλύτερη ακρίβεια στο γέμισμα. Έτσι το εργοστάσιο θα μπορεί να πέτυχει καλύτερη διαχείριση του προσωπικού, ακριβής τοποθέτηση υλικού, άρα περισσότερο κέρδος. Τα είδη χημικών υγρών όμως, ποικίλουν ανάλογα με την ρευστότητά τους. Αυτό κάνει το σύστημα πλήρωσης να χρειάζεται διαφορετικές ρυθμίσεις για κάθε είδος, πόσο μάλλον όταν η διαδικασία αυτή γίνεται με ελεύθερη ροή, οπότε μπαίνει μέσα και ο παράγοντας της βαρύτητας. Έτσι είναι αδύνατον μία απλή μηχανή πλήρωσης να παρέχει σταθερά αποτελέσματα καθ' όλη τη διάρκεια του γεμίσματος των δοχείων.

# Περίληψη

Σε αυτή τη διπλωματική εργασία κατασκευάστηκε ένα σύστημα το οποίο υπόσχεται να δώσει λύση σε ένα πρόβλημα που σχετίζεται με συσκευασία δοχείων χημικών υγρών. Το πρόβλημα εντοπίζεται στα εργοστάσια παραγωγής χημικών υγρών στα οποία η διαδικασία συσκευασίας των δοχείων γίνεται με χειροκίνητο τρόπο και με ελεύθερη ροή. Οπότε το σημαντικότερο δεδομένο που θα πρέπει να υπολογίζεται από το σύστημα, είναι ο ρυθμός ροή. Για να γίνει αυτό θα πρέπει να είναι γνωστή, η ρευστότητα του χημικού υγρού που επιλέχθηκε, η πίεση στην έξοδο της δεξαμενής όπου είναι τοποθετημένο και η θερμοκρασία του χημικού υγρού που έχει άμεση συσχέτιση με τη ρευστότητα. Το σύστημα συλλεγεί τις μετρήσεις θερμοκρασίας και πίεσης, οι οποίες επεξεργάζονται από κάποια κυκλώματα μετατροπής για να είναι διαχειρίσιμα από τον μικροελεγκτή. Τα δεδομένα της θερμοκρασίας αποστέλλονται σε έναν τοπικό server ο οποίος με την σειρά του στέλνει πίσω τα δεδομένα της ρευστότητας για να ολοκληρωθεί ο υπολογισμός του εκτιμώμενου ρυθμού ροής. Κατά την διάρκεια της διαδικασίας του γεμίσματος γίνεται μέτρηση του βάρους, και με βάση τον χρόνο, υπολογίζεται ο πραγματικός ρυθμός ροή για την αποφυγή σφαλμάτων. Τα δεδομένα για την επιλογή του κατάλληλου χημικού υγρού από τον χρήστη γίνεται μέσω ενός πληκτρολογίου ενώ η εμφάνιση των δεδομένων γίνεται σε μία οθόνη αποτελούμενη από πέντε ψηφία. Το γέμισμα των δοχείων γίνεται από μία βαλβίδα ελεγχόμενη με αέρα για αποφυγή σπινθηρισμών, και ο έλεγχος του αέρα γίνεται από μία ηλεκτροβαλβίδα τοποθετημένη σε ασφαλή απόσταση από το χημικό υγρό. Αυτό συμβαίνει διότι πολλές φορές τα χημικά υγρά είναι εύφλεκτα και μπορεί να προκληθεί φωτιά. Κατά τη διάρκεια των δοκιμών ανακαλύφθηκε ότι σε πιο υδαρή χημικά υγρά και σε μικρές ζητούμενες ποσότητες γεμίσματος υπήρχε πρόβλημα στον εντοπισμό της πραγματικής ροής. Για να αποφευχθεί αυτό το πρόβλημα θα πρέπει να τοποθετηθεί ρυθμιστής ροής πριν από την βαλβίδα εξόδου.

# «Chemical fluids filling system with variable flow calculation»

«Nikisanlis Nikolaos»

## Abstract

In this thesis, a system was developed to address a problem related to the packaging of chemical liquids. The issue arises in chemical production plants where container packaging is done manually and with free-flowing methods. Therefore, the most crucial parameter for the system to compute is the flow. To achieve this, the system needs to know the viscosity of the chosen chemical liquid, as well as the pressure at the tank outlet. Viscosity is directly correlated with the temperature of the chemical liquid. Thus, the system gathers temperature and pressure measurements, which are processed by conversion circuits to be manageable by the microcontroller. Temperature data is transmitted to a local server, which in turn sends back viscosity data to complete the calculation of the estimated flow rate. During the filling process, weight measurement is taken, and the actual flow rate is calculated based on time to prevent any potential errors. User input for selecting the appropriate chemical liquid is done through a keyboard, while data presentation occurs on a screen consisting of five digits. Container filling is carried out by an air-controlled valve to prevent sparks, and air control is managed by a solenoid valve placed at a safe distance from the chemical liquid. This precaution is necessary as many chemical liquids are flammable and could cause a fire. During testing, it was found that there was a problem in accurately detecting the flow rate in more watery chemical liquids and in small requested filling quantities. To address this issue, a flow adjusting device should be installed before the output valve.

# Περιεχόμενα

# Κατάλογος Σχημάτων και Εικόνων

# Κατάλογος Πινάκων και Γραφημάτων

# Συντομογραφίες

| | |
|---|---|
| ΔΕ | Διπλωματική Εργασία |
| ΟΚ | Ολοκληρωμένο Κύκλωμα |
| ΤΕ | Τελεστικός Ενισχυτής |
| ADC | Analog to Digital Converter |
| SMD | Surface Mount Device |
| $I^2C$ | Inter-Integrated Circuit |
| SDA | Serial DAta |
| SCL | Serial CLock |
| SPI | Serial Peripheral Interface |
| USB | Universal Serial Bus |

# ΚΕΦΑΛΑΙΟ 1

# Χημικά υγρά

## 1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει αναφορά στα χημικά και στα χαρακτηριστικά τους. Επίσης θα γίνει εξήγηση των σημαντικότερων χαρακτηριστικών και του υπολογισμού τη ροής που παίζει σημαντικό ρόλο στην διαδικασία του γεμίσματος.

## 1.2 Χημικό υγρό

Χημικά υγρά, επίσης γνωστά ως χημικές διαλύσεις ή χημικοί συνδυασμοί σε υγρή μορφή, είναι ουσίες που αποτελούνται από δύο ή περισσότερα χημικά συστατικά που αναμιγνύονται ή διαλύονται σε ένα υγρό διαλύτη. Αυτά τα υγρά μπορούν να διαφέρουν ευρέως στη σύνθεση, τις ιδιότητες και τις χρήσεις ανάλογα με τα συγκεκριμένα χημικά που εμπλέκονται.

## 1.3 Ιδιότητες χημικών υγρών

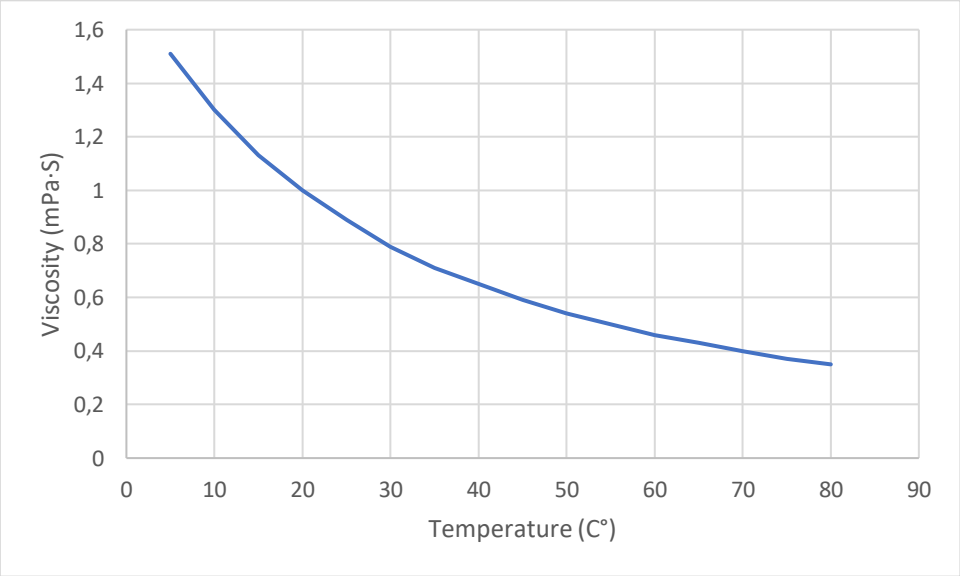
Οι ιδιότητες των χημικών υγρών εξαρτώνται από τις ιδιότητες των μεμονωμένων συστατικών και τις συγκεντρώσεις τους. Αυτές οι ιδιότητες μπορεί να περιλαμβάνουν το ιξώδες, την πυκνότητα, το pH, την αγωγιμότητα και τη χημική αντίδραση. Τα χημικά υγρά μπορεί να είναι όξινα, αλκαλικά, διαβρωτικά ή αδρανή ανάλογα με τη σύνθεσή τους.

### 1.3.1 Ρευστότητα

Η ρευστότητα, ή αλλιώς ιξώδες, είναι η αντίσταση ενός υγρού στη ροή. Δηλαδή αναφέρεται στο πόσο εύκολα ρέει ένα ρευστό. Τα υγρά με υψηλό ιξώδες ρέουν αργά, ενώ τα υγρά με χαμηλό ιξώδες ρέουν πιο εύκολα. Είναι μια βασική ιδιότητα των υγρών και παίζει σημαντικό ρόλο σε διάφορες βιομηχανικές διαδικασίες, επιστημονικές εφαρμογές και καθημερινά φαινόμενα. Επίσης το ιξώδες επηρεάζεται άμεσα από την θερμοκρασία του υγρού.

Πίνακας 1.1: Ενδεικτικός πίνακας ιξώδες νερού

| Temperature (C°) | Viscocity (mPa·S) |
|---|---|
| 5 | 1,51 |
| 10 | 1,3 |
| 15 | 1,13 |
| 20 | 1 |
| 25 | 0,89 |
| 30 | 0,79 |
| 35 | 0,71 |
| 40 | 0,65 |
| 45 | 0,59 |
| 50 | 0,54 |
| 55 | 0,5 |
| 60 | 0,46 |
| 65 | 0,43 |
| 70 | 0,4 |
| 75 | 0,37 |
| 80 | 0,35 |




Γράφημα 1.1: Σχετικό γράφημα ιξώδες νερού

### 1.3.2 Ειδικό βάρος

Το ειδικό βάρος για υγρά, γνωστό και ως μοναδιαίο βάρος υγρών, είναι ένα μέτρο του βάρους ανά μονάδα όγκου ενός ρευστού. Ποσοτικοποιεί τη δύναμη που ασκείται από τη βαρύτητα σε μια μονάδα όγκου του ρευστού. Το ειδικό βάρος είναι μια θεμελιώδης ιδιότητα των ρευστών και είναι ζωτικής σημασίας σε διάφορες μηχανικές και επιστημονικές εφαρμογές, ιδιαίτερα στη μηχανική των ρευστών και την υδροδυναμική.

## 1.4 Πίεση εξόδου

Η πίεση των ρευστών είναι μια θεμελιώδης έννοια στη μηχανική των ρευστών και αναφέρεται στη δύναμη που ασκείται από ένα ρευστό ανά μονάδα επιφάνειας. Διαδραματίζει καθοριστικό ρόλο σε διάφορες μηχανολογικές και επιστημονικές εφαρμογές, που κυμαίνονται από τα υδραυλικά συστήματα έως τη δυναμική της ατμόσφαιρας. Σε μία δεξαμενή η πίεση είναι ανάλογη με την ποσότητα του υγρού μέσα σε αυτή.

## 1.5 Υπολογισμός Ροής

Το σημαντικότερο δεδομένο στην διαδικασία γεμίσματος είναι ο ρυθμός ροή του υγρού. Για να υπολογιστεί η τιμή της ροής χρειάζονται τα δεδομένα της ρευστότητας του υγρού, η πίεση του υγρού στην έξοδο της δεξαμενής και η διατομή όπως και το μήκος του αγωγού εξόδου της δεξαμενής. Ο τύπος για τον υπολογισμό της ροής είναι[8]:

$$Q = \frac{\pi * R^4 * p}{8 * \mu * L} \tag{1.1}$$

Όπου **Q** είναι ο ρυθμός ροής(L/h), **R** είναι η διατομή(mm) και **L** το μήκος(m) του αγωγού, **μ** είναι η ρευστότητα(kg/m·s) και **p** είναι η πίεση εξόδου(Pa). Η τιμή της ροής συνήθως μετριέται σε μονάδα μέτρησης υγρών στο χρόνο, για παράδειγμα λίτρα ανά ώρα. Κάποιες φορές όμως πρέπει να μετατραπεί σε μονάδα μέτρησης βάρους στο χρόνο, για παράδειγμα κιλά ανά ώρα. Έτσι ο τύπος υπολογισμού είναι[8]:

$$m = Q * \rho \tag{1.2}$$

Όπου **m** ο ρυθμός ροής σε βάρος(kg/h) και **ρ** το ειδικό βάρος του υγρού(kg/m$^3$).

## 1.6 Επίλογος

Σε αυτό το κεφάλαιο έγινε επεξήγηση των ιδιοτήτων των χημικών υγρών και ο ρόλος που παίζουν στην διαδικασία γεμίσματός. Επίσης έγινε ο υπολογισμός της ροής του χημικού υγρού με βάση όλες αυτές τις ιδιότητες.

# ΚΕΦΑΛΑΙΟ 2

## Σχεδίαση κυκλωμάτων μέτρησης

### 2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει αναφορά στους αισθητήρες που θα χρησιμοποιηθούν, θα αναλυθούν τα κυκλώματα ενίσχυσης και προσαρμογής όπως επίσης και το κύκλωμα μετατροπής του αναλογικού σήματος σε ψηφιακό για την ανάγνωσή του από τον μικροελεγκτή.

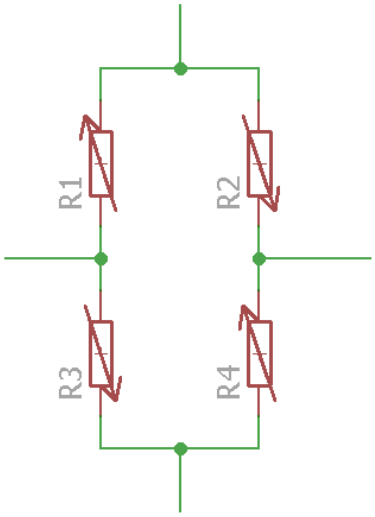
### 2.2 Αισθητήρες

Οι αισθητήρες που χρησιμοποιήθηκαν είναι οι παρακάτω τρεις.

- Δύναμης (βάρους)
- Θερμοκρασίας
- Πίεσης

Εικόνα 2.1: Αισθητήρας βάρους [9]

Για την μέτρηση του βάρους χρησιμοποιήθηκε ένας αισθητήρας φορτίου (load cell) ή αλλιώς μετρητής καταπόνησης με μέγιστο φορτίο 200 kg και ευαισθησία 2mV/V όταν εφαρμόζεται το μέγιστο βάρος. Η αντίσταση στον κάθε κλάδο της γέφυρας είναι 350Ω και η μέγιστη μεταβολή είναι 0.2%.




Σχήμα 2.1: Σχέδιο αντιστάσεων αισθητήρα βάρους

Εικόνα 2.2: Αισθητήρας θερμοκρασίας [9]

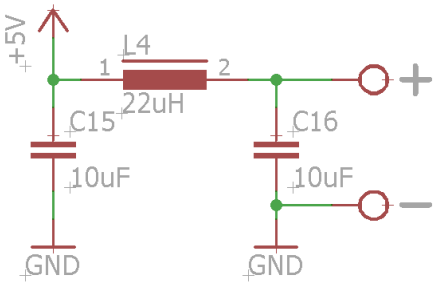
Για την μέτρηση της θερμοκρασίας χρησιμοποιήθηκε ένα NTC θερμίστορ με σπείρωμα ώστε να μπορεί να βιδωθεί στην σωλήνα εξόδου του χημικού υγρού παρέχοντας στεγανότητα. Η αντίσταση του αισθητηρίου στους 25ºC είναι 10kΩ και στον παρακάτω πίνακα φαίνεται η μεταβολή της αντίστασης για κάθε τιμή της θερμοκρασίας.

Πίνακας 2.1 Τιμές αντίστασης αισθητήρα θερμοκρασίας

| Θερμοκρασία (ºC) | Αντίσταση (Ω) |
|---|---|
| 0 | 32650 |
| 5 | 25395 |
| 10 | 19903 |
| 15 | 15714 |
| 20 | 12493 |
| 25 | 10000 |
| 30 | 8056 |
| 35 | 6530 |
| 40 | 5324 |
| 45 | 4366 |
| 50 | 3601 |
| 55 | 2985 |
| 60 | 2487 |
| 65 | 2082 |
| 70 | 1751 |
| 75 | 1480 |
| 80 | 1256 |
| 85 | 1070 |
| 90 | 916 |
| 95 | 786 |
| 100 | 678 |

Εικόνα 2.3: Αισθητήρας πίεσης [9]

Για την μέτρηση της πίεσης εξόδου χρησιμοποιήθηκε ένας αισθητήρας πίεσης υγρών με μέγιστη τιμή μέτρησης 1.6MPa. Ο αισθητήρας τροφοδοτείται με τάση 5V και η αναλογική έξοδος κυμαίνεται από 0.5V έως 4.5V.




Σχήμα 2.2: Φίλτρο τροφοδοσίας αισθητήρων

Όλοι οι αισθητήρες τροφοδοτούνται με τάση 5V μέσω ενός χαμηλοπερατού φίλτρου CLC σε διάταξη Π. Η συχνότητα αποκοπής του φίλτρου υπολογίζεται από τον παρακάτω τύπο:

$$f_c = \frac{1}{\pi\sqrt{LC}} \tag{2.1}$$

Αυτό το φίλτρο τοποθετείται για να υπάρχει σταθερή τροφοδοσία στους αισθητήρες εξαλείφοντας τυχόν παράσιτα από το παλμικό τροφοδοτικό στο οποίο θα γίνει αναφορά στο επόμενο κεφάλαιο.

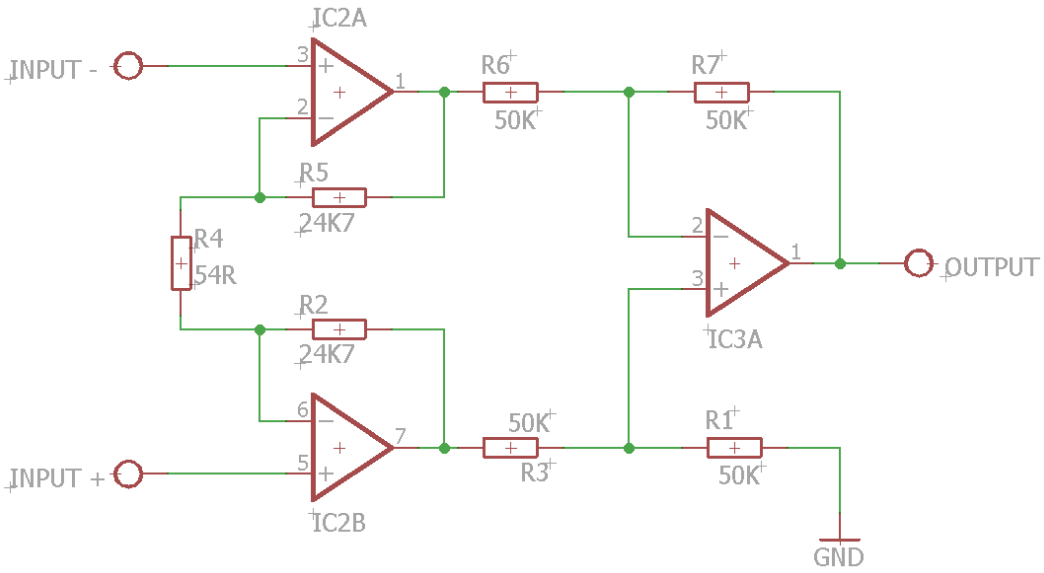
## 2.3 Κυκλώματα προσαρμογής και ενίσχυσης

Η αναλογική τάση που παράγεται από τους αισθητήρες τις περισσότερες φορές είναι αρκετά χαμηλή ή άλλες φορές είναι εκτός των ορίων που μπορεί να χρησιμοποιηθεί. Έτσι θα πρέπει να υλοποιηθεί κύκλωμα προσαρμογής του σήματος στα επιθυμητά όρια.

### 2.3.1 Μέτρηση βάρους

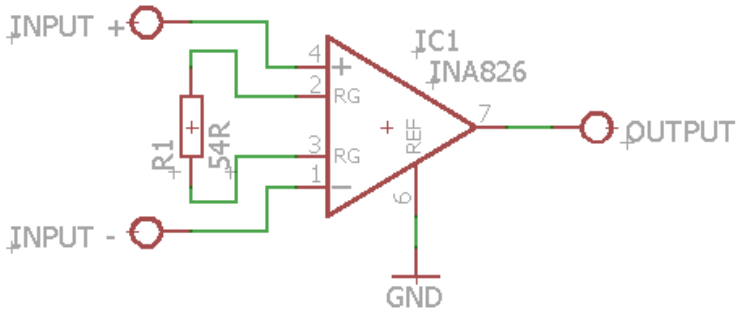
Ο αισθητήρας βάρους στο μέγιστο φορτίο του βγάζει 2mV/V και η τροφοδοσία του είναι 5V. Στο συγκεκριμένο σύστημα το ωφέλιμο βάρος είναι 90Kg το οποίο αντιστοιχεί σε 0,9mV/V. Οπότε η έξοδος του αισθητήρα θα έχει μέγιστη τιμή:

$$0{,}9 * 5 = 4{,}5mV \tag{2.2}$$




Σχήμα 2.3: Ενισχυτής οργανολογίας με τρείς τελεστικούς

Για την ενίσχυση αυτής της τάσης χρησιμοποιήθηκε ένας ενισχυτής οργανολογίας INA826, αποτελούμενος από τρεις τελεστικούς ενισχυτές, ενσωματωμένους σε ένα ολοκληρωμένο κύκλωμα.




Σχήμα 2.4: Ολοκληρωμένος διαφορικός ενισχυτής

Στην έξοδο του ενισχυτή οργανολογίας θα συνδεθεί ένας αναλογικός μετατροπέας ADC. Η μέγιστη τάση εισόδου του αναλογικού μετατροπέα είναι 4.096V. Οπότε έχοντας αυτά τα δεδομένα το κέρδος του διαφορικού ενισχυτή θα πρέπει να είναι:

$$\frac{4096}{4.5} = 910 \tag{2.3}$$

Ο διαφορικός ενισχυτή που χρησιμοποιήθηκε είναι ο INA826 της εταιρίας Texas Instruments. Για να υπολογιστεί η κατάλληλη αντίσταση για το επιθυμητό κέρδος είναι [6]:

$$G = 1 + \frac{49400}{R_G} \Rightarrow R_G = \frac{49400}{910-1} = 54{,}34\Omega \tag{2.4}$$

### 2.3.2 Μέτρηση πίεσης

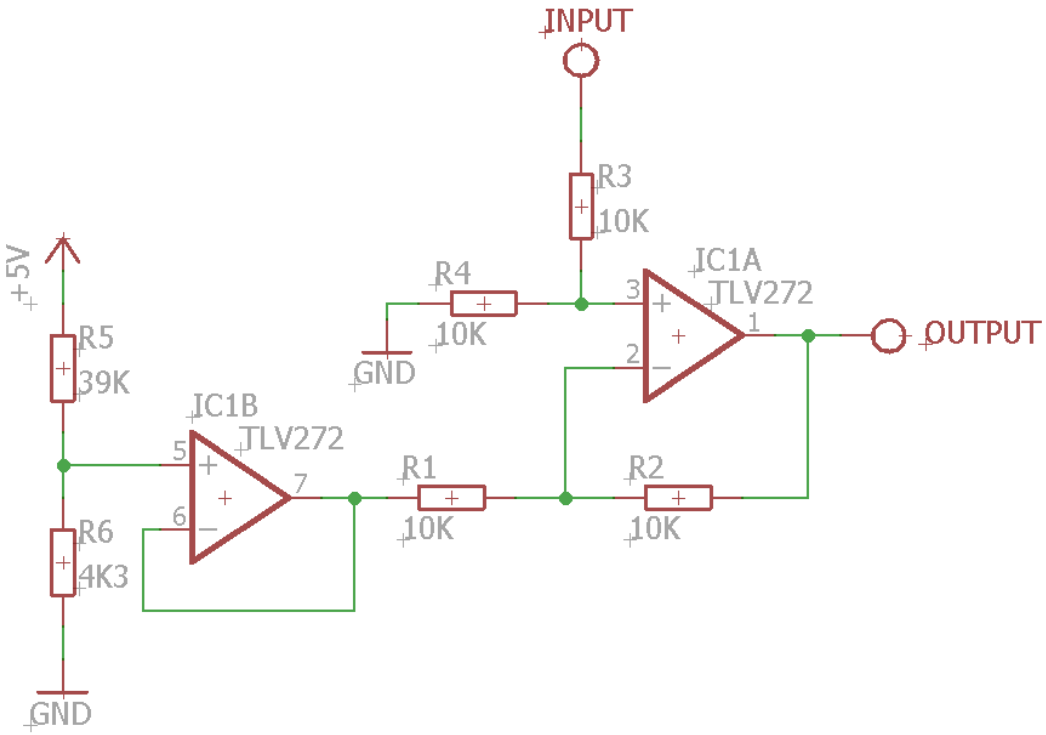
Το εύρος της τάσης εξόδου του αισθητήρα πίεσης είναι 0,5V – 4,5V ενώ η επιθυμητή τάση είναι 0V – 4V. Έτσι πρέπει να χρησιμοποιηθεί κύκλωμα που θα μετασχηματίσει την τάση εξόδου του αισθητήρα στην επιθυμητή τάση. Μετά από υπολογισμούς υπήρξε η ανάγκη να τοποθετηθεί στη είσοδο του κυκλώματος προσαρμογής ένας διαιρέτης τάσης για μειώσει στο μισό την τάση του αισθητήρα πίεσης. Έτσι οι τύποι με τους οποίους θα καθοριστεί το κύκλωμα προσαρμογή είναι οι παρακάτω:

$$0 = \frac{0,5}{2}\alpha + \beta \tag{2.5}$$

$$4 = \frac{4,5}{2}\alpha + \beta \tag{2.6}$$

Με βάση τους παραπάνω τύπους προκύπτει η εξίσωση:

$$y = 2x - 0,5 \tag{2.7}$$




Σχήμα 2.5: Κύκλωμα προσαρμογής τάσης αισθητήρα πίεσης

Οπότε το κύκλωμα που προκύπτει είναι το παραπάνω στο οποίο εφαρμόζεται η τάση του αισθητήρα στην θετική είσοδο του τελεστικού ενισχυτή μέσο ενός διαιρέτη τάσης και η τάση αναφοράς στην αρνητική είσοδο μέσω ενός δεύτερου τελεστικού ενισχυτή ως buffer. Με τους παρακάτω τύπους υπολογίσθηκαν οι αντιστάσεις του κυκλώματος.

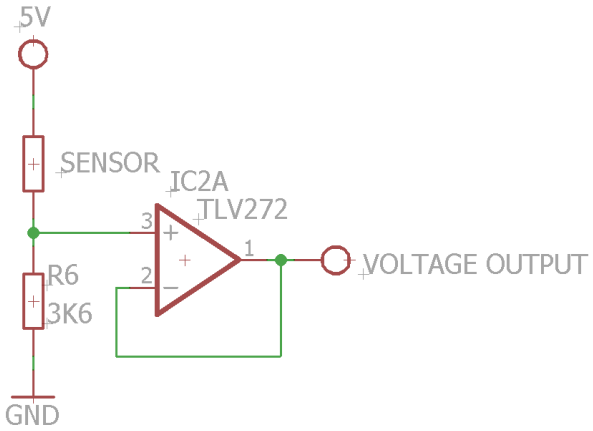
$$\alpha = 1 + \frac{R2}{R1} => \ R2 = (\alpha - 1) * R1 = (2 - 1) * 10K = 10k\Omega \tag{2.8}$$

$$\beta = 5 * \frac{R6}{R5+R6} * \frac{R2}{R1} => R5 = \frac{5*R6*\frac{R2}{R1}}{\beta*R6} = \frac{5*4.3k*\frac{10k}{10k}}{0.5*4.3k} = 39k\Omega \tag{2.9}$$

Στο παρακάτω γράφημα φαίνεται πως διαμορφώνεται η τάση εξόδου του κυκλώματος προσαρμογής σε σύγκριση με την τάση εξόδου του αισθητήρα πίεσης.


Γράφημα 2.1: Σύγκριση τάσεων εξόδου κυκλώματος αισθητήρα πίεσης

### 2.3.3 Μέτρηση θερμοκρασίας




Σχήμα 2.6: Μέτρηση θερμοκρασίας σε τάση

Ο αισθητήρας θερμοκρασίας είναι μία μεταβλητή αντίσταση η οποία μεταβάλλεται με την θερμοκρασία. Έτσι ο αισθητήρας τοποθετήθηκε σε διάταξη διαιρέτη τάσης για να μετατραπεί η μεταβολή της αντίστασης σε μεταβολή τάσης.

$$Vout = 5 * \frac{R6}{R6+SENSOR} \quad (2.10)$$

Πίνακας 2.2: Τάση εξόδου αισθητήρα θερμοκρασίας

| Θερμοκρασία (ºC) | Αντίσταση (Ω) | Τάση εξόδου (V) |
|---|---|---|
| 0 | 32650 | 0,50 |
| 5 | 25395 | 0,62 |
| 10 | 19903 | 0,77 |
| 15 | 15714 | 0,93 |
| 20 | 12493 | 1,12 |
| 25 | 10000 | 1,32 |
| 30 | 8056 | 1,54 |
| 35 | 6530 | 1,78 |
| 40 | 5324 | 2,02 |
| 45 | 4366 | 2,26 |
| 50 | 3601 | 2,50 |
| 55 | 2985 | 2,73 |
| 60 | 2487 | 2,96 |
| 65 | 2082 | 3,17 |
| 70 | 1751 | 3,36 |
| 75 | 1480 | 3,54 |
| 80 | 1256 | 3,71 |
| 85 | 1070 | 3,85 |
| 90 | 916 | 3,99 |
| 95 | 786 | 4,10 |
| 100 | 678 | 4,21 |

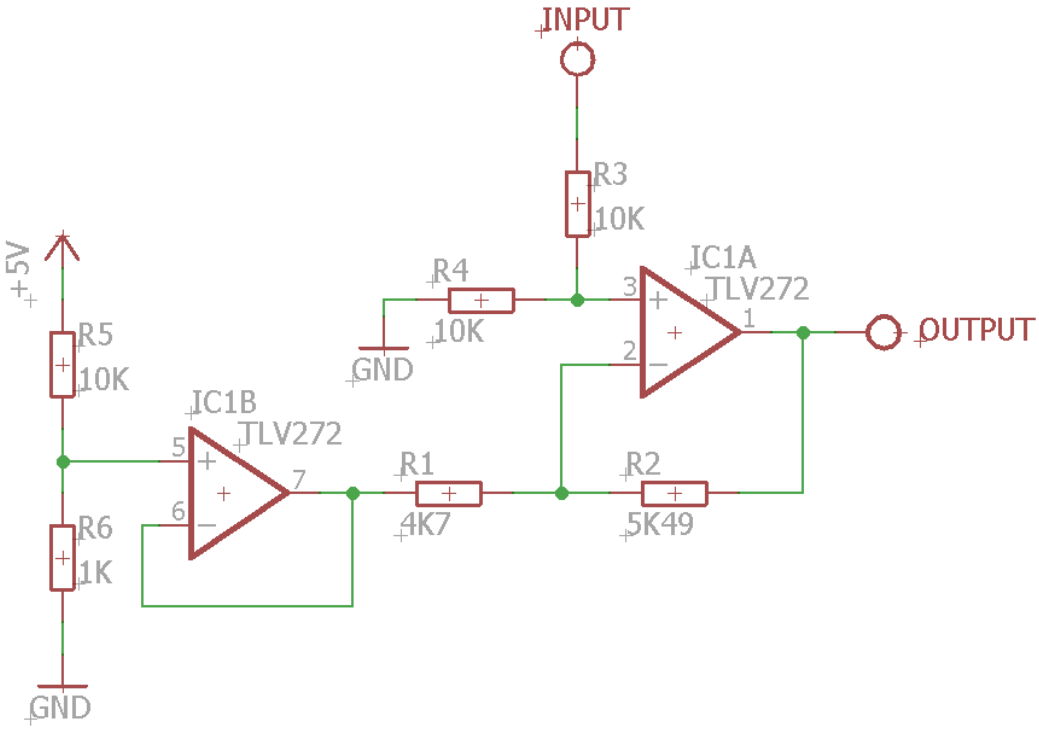
Στον πίνακα 2.2 φαίνεται η τάση εξόδου σε σύγκριση με την αντίσταση του αισθητήρα για κάθε τιμή θερμοκρασίας. Έτσι η τιμή της τάσης για θερμοκρασία από 0 - 100 Cº κυμαίνεται από 0,5 – 4,2 V. Με βάση αυτά τα δεδομένα, όπως και στο κύκλωμα του αισθητήρα πίεσης, είναι απαραίτητο να τοποθετηθεί ένας διαιρέτης τάσης στην είσοδο του κυκλώματος προσαρμογής για να μειώσει στο μισό την τάση που προέρχεται από το κύκλωμα του αισθητήρα θερμοκρασίας. Οι τύποι υπολογισμού του κυκλώματος είναι οι παρακάτω:

$$0 = \frac{0,5}{2}\alpha + \beta \tag{2.11}$$

$$4 = \frac{4,2}{2}\alpha + \beta \tag{2.12}$$

Με βάση τους παραπάνω τύπους προκύπτει η εξίσωση:

$$y = 2,16x - 0,54 \tag{2.13}$$



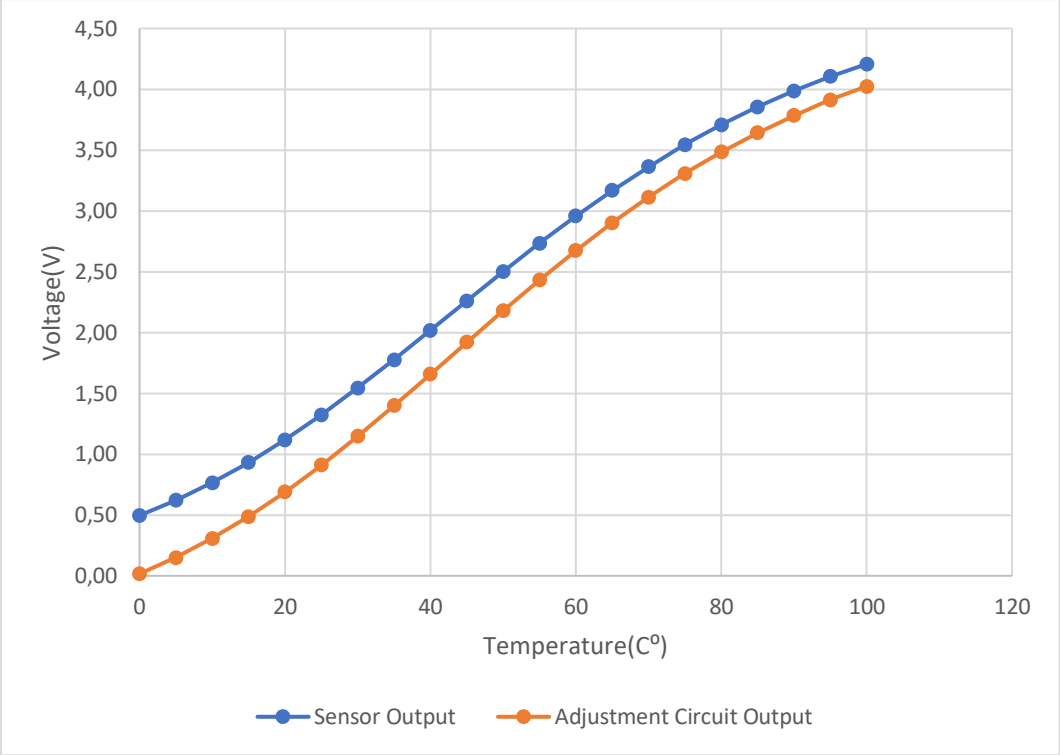

Σχήμα 2.7: Κύκλωμα προσαρμογής τάσης αισθητήρα θερμοκρασίας

Οπότε το κύκλωμα που προκύπτει είναι το παραπάνω, και είναι παρόμοιο με το κύκλωμα προσαρμογής του αισθητήρα πίεσης, στο οποίο εφαρμόζεται η τάση του αισθητήρα στην θετική είσοδο του τελεστικού ενισχυτή μέσω ενός διαιρέτη τάσης και η τάση αναφοράς στην αρνητική είσοδο μέσω ενός δεύτερου τελεστικού ενισχυτή ως buffer. Με τους παρακάτω τύπους υπολογίσθηκαν οι αντιστάσεις του κυκλώματος.

$$\alpha = 1 + \frac{R2}{R1} => \ R2 = (\alpha - 1) * R1 = (2{,}16 - 1) * 4{,}7K = 5{,}49k\Omega \quad (2.14)$$

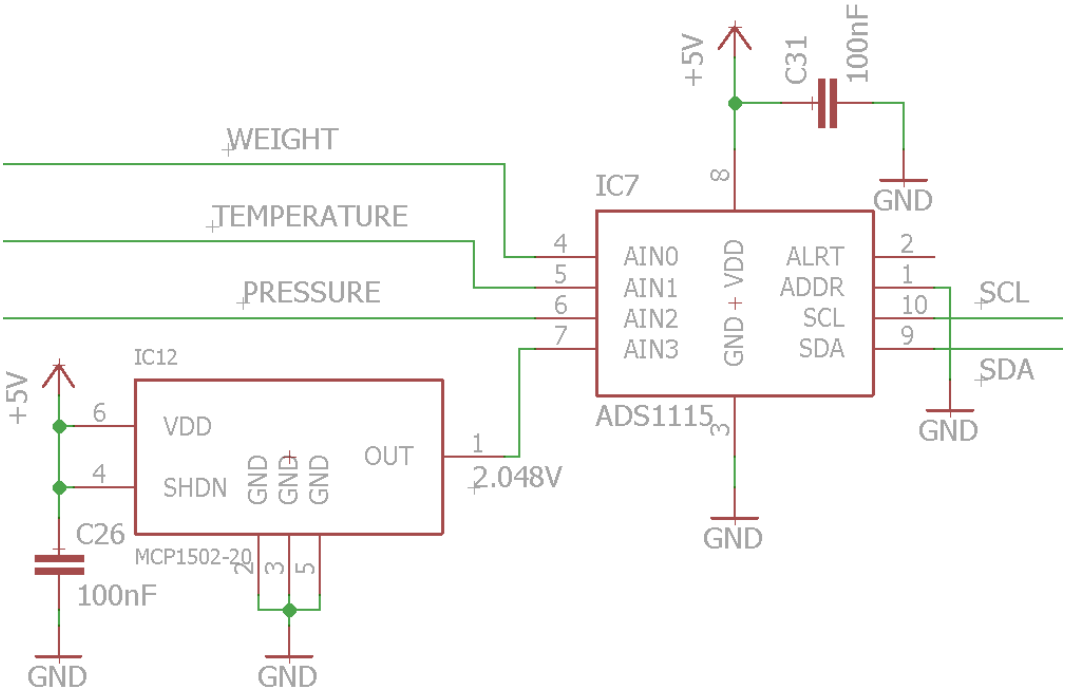
$$\beta = 5 * \frac{R6}{R5+R6} * \frac{R2}{R1} => R5 = \frac{5*R6*\frac{R2}{R1}}{\beta*R6} = \frac{5*1k*\frac{5{,}49k}{4{,}7k}}{0{,}54*1k} = 9{,}87k\Omega \quad (2.15)$$

Στο παρακάτω γράφημα φαίνεται πως διαμορφώνεται η τάση εξόδου του κυκλώματος προσαρμογής σε σύγκριση με την τάση εξόδου του αισθητήρα θερμοκρασίας.




Γράφημα 2.2: Σύγκριση τάσεων εξόδου κυκλώματος αισθητήρα θερμοκρασίας

## 2.4 Αναλογικός μετατροπέας

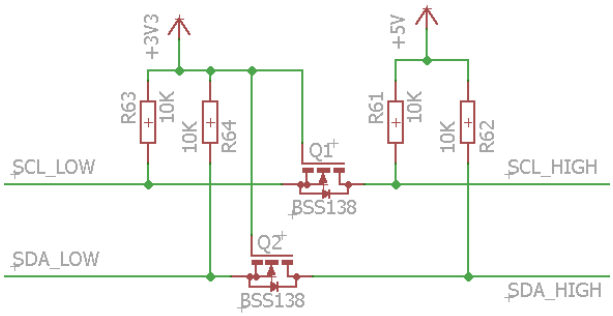



Σχήμα 2.8: Κύκλωμα μετατροπής αναλογικής τάσης σε ψηφιακό σήμα

Για να μπορέσει ο μικροελεγκτής να διαβάσει τις τάσεις εξόδου από τα κυκλώματα των αισθητήρων θα πρέπει να χρησιμοποιηθεί κύκλωμα μετατροπής του σήματος από αναλογικό σε ψηφιακό (ADC). Η ιδανική λύση θα ήταν να χρησιμοποιηθεί ο εσωτερικός ADC του μικροελεγκτή. Συνήθως οι εσωτερικοί ADCs των μικροελεγκτών έχουν μικρή ανάλυση οπότε αυτό τους κάνει να έχουν χαμηλή διακριτική ικανότητα. Για να επιτευχθεί καλύτερη διακριτική ικανότητα πρέπει να χρησιμοποιηθεί ένας εξωτερικός ADC. Έτσι χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα ADS1115 της εταιρίας Texas Instruments, το οποίο έχει τέσσερις αναλογικές εισόδους και ανάλυση 16 bits. Η επικοινωνία με τον μικροελεκτή γίνεται μέσω του πρωτοκόλλου $I^2C$. Επιπλέον, έχει την ικανότητα να μετρήσει την διαφορά τάσης μεταξύ δύο εισόδων, δηλαδή να κάνει διαφορική μέτρηση. Τέλος έχει την δυνατότητα για προγραμματιζόμενο εύρος τάσης εισόδου με τις παρακάτω επιλογές:

- ±6,144V
- ±4,096V
- ±2,048V
- ±1,024V
- ±0,512V
- ±0,256V

Όπως φαίνεται παραπάνω το εύρος μέτρησης τάσης εκτείνεται και προς τις αρνητικές τιμές ενώ η τάση εξόδου των κυκλωμάτων των αισθητήρων είναι θετικές τιμές. Αυτό κάνει τον ADC να εκμεταλλεύεται μόνο το μισό εύρος μέτρησής του. Για να μπορέσει να γίνει εκμετάλλευση ολόκληρου του εύρους, χρησιμοποιήθηκε διαφορική μέτρηση των καναλιών 0, 1 και 2 σε σύγκριση με το κανάλι 3. Στο κανάλι 3 εφαρμόστηκε τάση 2,048V η οποία προέρχεται από έναν σταθεροποιητή τάσης ακριβείας. Το εύρος που επιλέχθηκε είναι ±2,048V και με αυτόν τον τρόπο το εύρος μέτρησης είναι 0 έως 4,096V, όσο και η έξοδος από τα κυκλώματα των αισθητήρων.




Σχήμα 2.9: Μετατροπέας λογικού επιπέδου.

Η τροφοδοσία του ADC είναι 5V, άρα και η τάση των ψηφιακών σημάτων που στέλνονται μέσα στο κανάλι επικοινωνίας είναι 5V. Η τροφοδοσία του μικροελεγκτής είναι 3.3V, οπότε υπάρχει κίνδυνος καταστροφής του. Έτσι τοποθετήθηκε ένα κύκλωμα μετατροπής λογικού επιπέδου που μετατρέπει αυτά τα σήματα από τα 5V στα 3.3V.

## 2.5 Επίλογος

Σε αυτό το κεφάλαιο έγινε αναφορά στα χαρακτηριστικά του αισθητήρα δύναμης, του αισθητήρα θερμοκρασίας, του αισθητήρα πίεσης καθώς και στον τρόπο τροφοδοσίας τους. Έγινε υπολογισμός του διαφορικού ενισχυτή για την μέτρηση του βάρους όπως επίσης έγινε ανάλυση στον τρόπο υπολογισμού και σχεδίασης των κυκλωμάτων προσαρμογής για την μέτρηση της θερμοκρασίας και της πίεσης. Τέλος, έγινε αναφορά στον ADC που χρησιμοποιήθηκε για την ανάγνωση των αναλογικών τάσεων από τον μικροελεγκτή, ο τρόπος με τον οποίο επιτεύχθηκε η μέγιστη ακρίβεια και το πρωτόκολλο επικοινωνίας του.

# ΚΕΦΑΛΑΙΟ 3

## Κυκλώματα του μικροελεγκτή και τροφοδοσία

### 3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει αναφορά στα χαρακτηριστικά του μικροελεγκτή που χρησιμοποιήθηκε και στον τρόπο επικοινωνίας του με εξωτερική συσκευή. Επίσης θα αναλυθούν τα κυκλώματα εισαγωγής εντολών από τον χρήστη, εμφάνισης αποτελεσμάτων και εκτέλεσης εντολών. Τέλος, θα αναλυθούν τα κυκλώματα τροφοδοσίας του συστήματος.

### 3.2 Μικροελεγκτής


Εικόνα 3.1: ESP32

Ο μικροελεγκτής που χρησιμοποιήθηκε είναι ο ESP32-WROOM-32E της εταιρίας Espressif Systems. Ο ESP32 είναι μία ολοκληρωμένη μονάδα επεξεργασίας ενσωματωμένη μέσα σε ένα κέλυφος. Τα βασικά χαρακτηριστικά του είναι:

- Διπύρηνος επεξεργαστής στα 250MHz
- 448 KB μνήμη ROM
- 520 KB μνήμη RAM
- 40 MHz εσωτερικό κρυσταλλικό ταλαντωτή
- 8 MB μνήμη flash

Επίσης ο ESP32 έχει αρκετά περιφερειακά ενσωματωμένα. Κάποια από αυτά είναι:

- WIFI
- Bluetooth
- UART
- SPI
- I2C
- LED PWM
- GPIO
- ADC
- SD Card

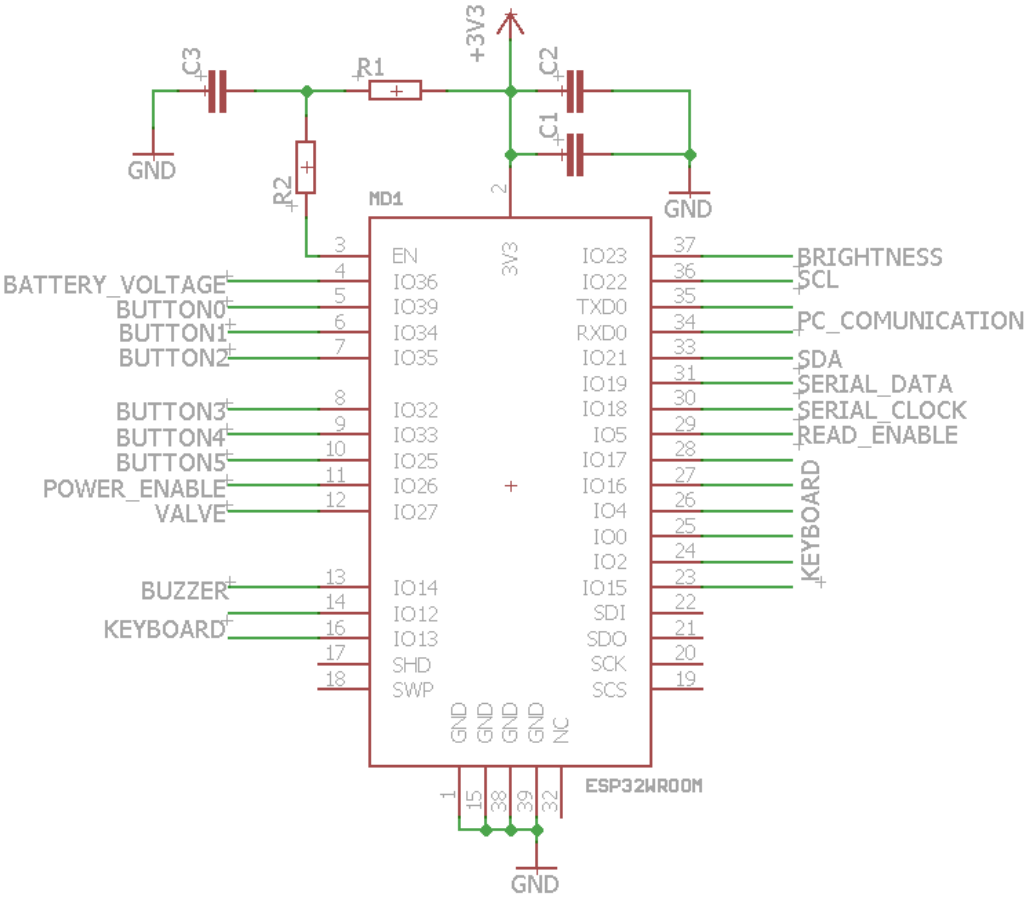
Γενικότερα οι είσοδοι – έξοδοι του μικροελεγκτή αποτελούνται από:

- 25 ψηφιακές εισόδους - εξόδους
- 17 αναλογικές εισόδους
- 1 κανάλι SD Card
- 2 κανάλια SPI επικοινωνίας
- 1 κανάλι I2C επικοινωνίας
- 2 κανάλια σειριακής επικοινωνίας

Στον πίνακα 3.1 και στο σχήμα 3.1 φαίνεται η αντιστοιχία των συσκευών του συστήματος με τις εισόδους – εξόδους του μικροελεγκτή.

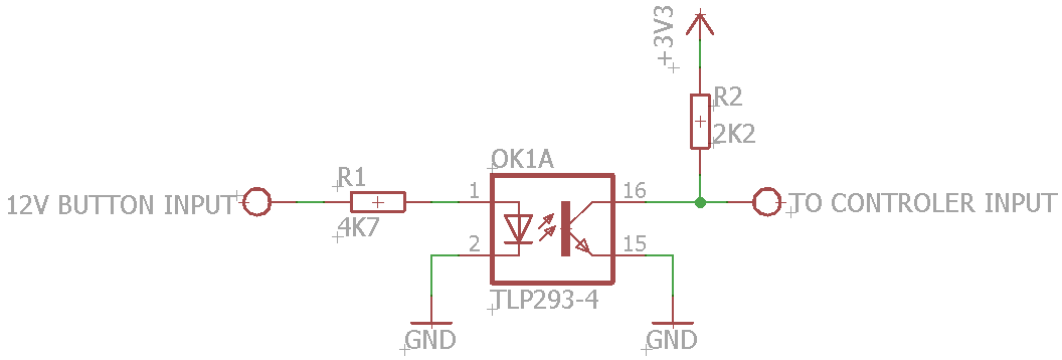
Πίνακας 3.1: Pin σύνδεσης συσκευών με τον μικροελεγκτή

| Device | Pins |
|---|---|
| Battery Voltage | IO36 |
| Button 0 – 5 | IO39, IO34, IO35, IO32, IO33, IO25 |
| Power enable | IO26 |
| Valve | IO27 |
| Buzzer | IO14 |
| Keyboard | IO12, IO13, IO15, IO2, IO0, IO4, IO16, IO17 |
| Display SPI | IO5, IO18, IO19, IO23 |
| ADC I2C | IO21, IO22 |
| PC Serial Communication | RXD0, TXD0 |



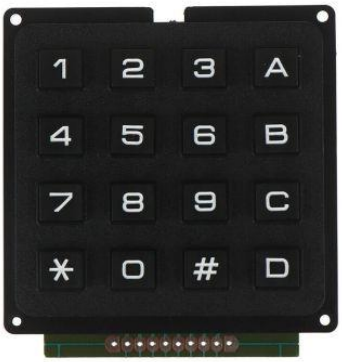

Σχήμα 3.1: Κύκλωμα σύνδεσης συσκευών στον μικροελεγκτή

## 3.3 Κυκλώματα εισόδου εντολών και δεδομένων



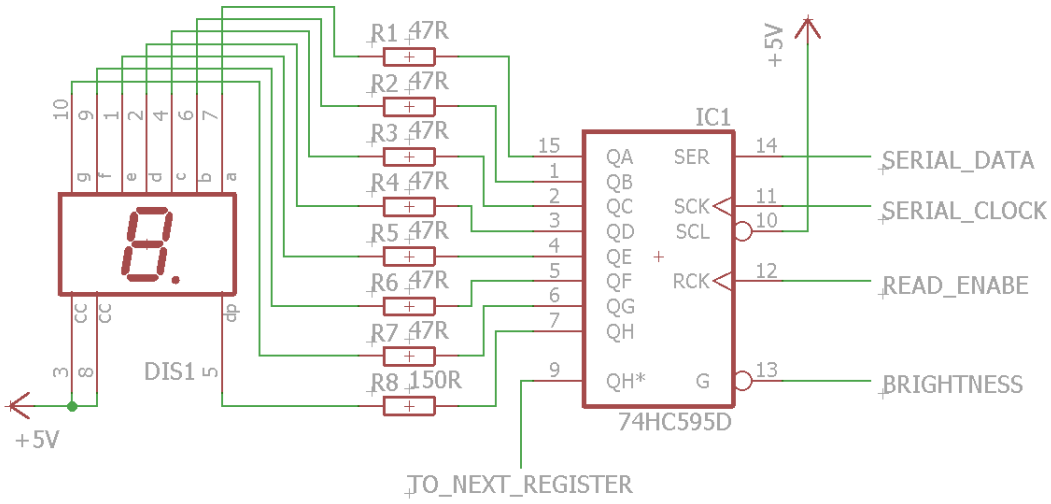

Σχήμα 3.2: Κύκλωμα οπτοζεύκτη

Οι εντολές από τον χρήστη δίνονται μέσω μπουτόν τα οποία κατά το πάτημά τους εισάγουν στην πλακέτα μία τάση 12V. Με αυτήν την τάση ενεργοποιείται ένας οπτοζεύκτης, μέσω μίας αντίστασης, ο οποίος γειώνει την είσοδο του μικροελεγκτή για να εκτελεστεί η εντολή. Τα δεδομένα που σχετίζονται με το γέμισμα εισάγονται από τον χρήστη μέσω ενός αριθμητικού πληκτρολογίου.



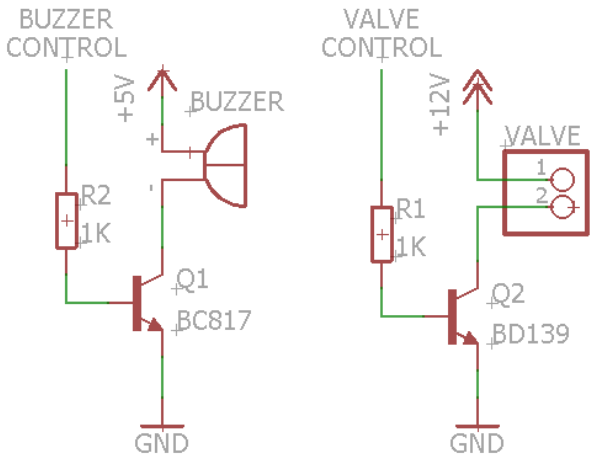

Εικόνα 3.2: Πληκτρολόγιο 4x4

## 3.4 Κυκλώματα εμφάνισης δεδομένων και εκτέλεσης εντολών



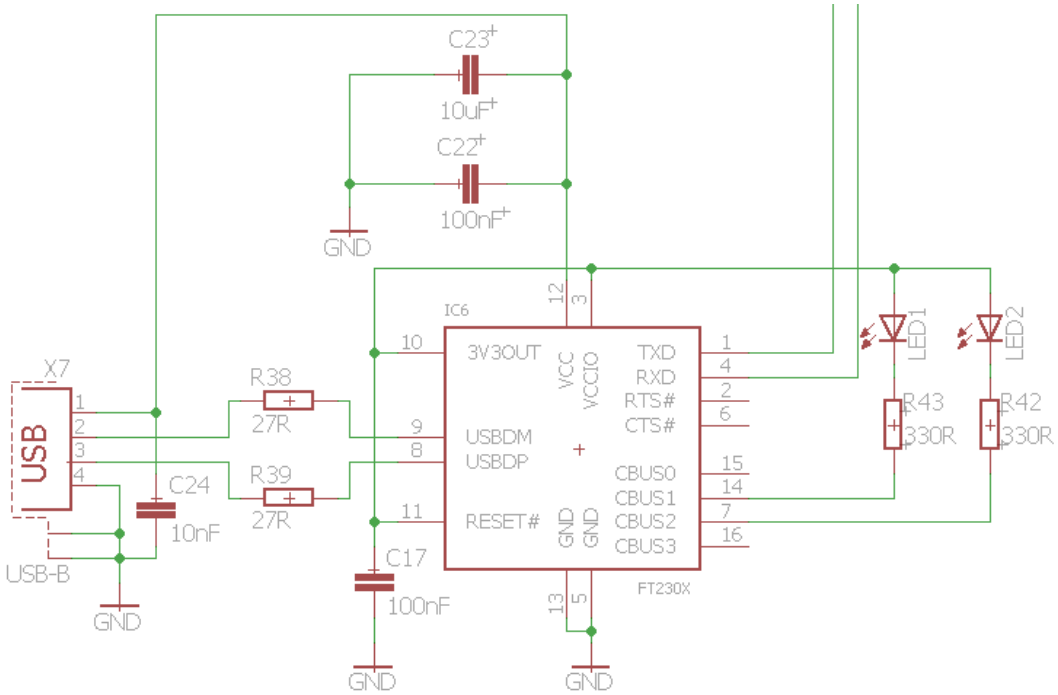

Σχήμα 3.3: Κύκλωμα shift register

Η εμφάνιση των δεδομένων γίνεται μέσω πέντε ενδεικτών επτά τομέων. Ο κάθε ενδείκτης συνδέεται σε έναν καταχωρητή. Ο καταχωρητής έχει οκτώ εξόδους με τις οποίες ελέγχεται ο κάθε τομέας του ενδείκτη, μία έξοδος η οποία συνδέεται στην σειριακή είσοδο δεδομένων του επόμενου καταχωρητή και τέσσερις εισόδους για την επικοινωνία με τον μικροελεγκχτή. Όλοι οι καταχωρητές είναι συνδεδεμένοι σε σειρά και η επικοινωνία γίνεται μέσω του πρωτοκόλλου Software SPI. Οι εντολές που εκτελούνται είναι η ενεργοποίηση και απενεργοποίηση της βαλβίδας γεμίσματος. Αυτό γίνεται μέσω ενός NPN transistor τοποθετημένο στην γείωση του πηνίου της ηλεκτροβαλβίδας. Τέλος υπάρχει ένα buzzer που παρέχει ηχητικές ενδείξεις στον χρήστη.




Σχήμα 3.4: Κύκλωμα ενεργοποίησης βαλβίδας και buzzer ηχητικής ένδειξης

## 3.5 Σειριακή επικοινωνία με υπολογιστή




Σχήμα 3.5: Κύκλωμα σειριακής επικοινωνίας

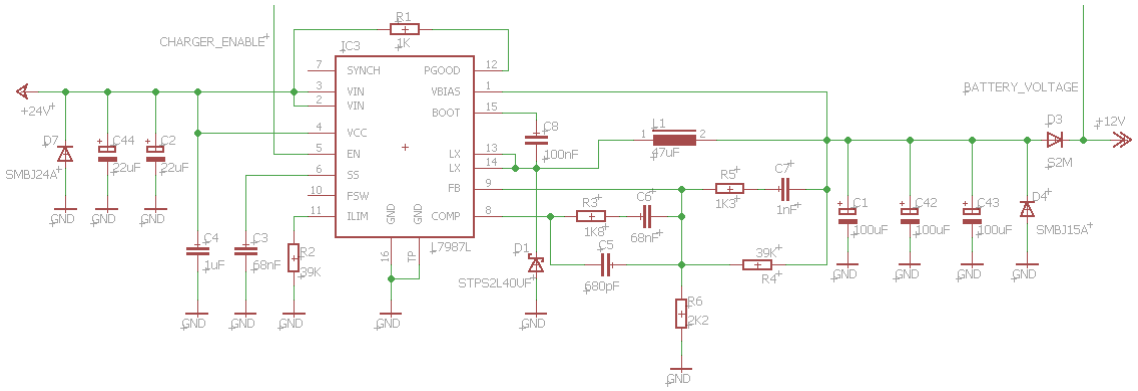
Για τον προγραμματισμό και την ρύθμιση του μικροελεγκτή χρησιμοποιήθηκε η εσωτερική σειριακή επικοινωνία του. Η σύνδεση με τον υπολογιστή γίνεται μέσω θύρας USB, με τη χρήση του

ολοκληρωμένου κυκλώματος FT230X της εταιρίας FTDI Chip, το οποίο είναι υπεύθυνο για την μετατροπή του σειριακού πρωτοκόλλου σε πρωτόκολλο USB.

## 3.6 Κυκλώματα τροφοδοσίας

Επειδή τα χημικά υγρά είναι εύφλεκτα, απαγορεύεται η παρουσία καλωδίων υψηλής τάσης, για την αποφυγή πρόκλησης φωτιά σε περίπτωση σπινθηρισμού. Έτσι η τροφοδοσία γίνεται αποκλειστικά από μπαταρία. Για αυτό το λόγο χρησιμοποιήθηκαν δύο παλμικά τροφοδοτικά. Το πρώτο τροφοδοτείται από την μπαταρία για να παράγει τις τάσεις που χρειάζεται το κύκλωμα και το δεύτερο είναι υπεύθυνο για την φόρτιση της μπαταρίας.

### 3.6.1 Φορτιστής μπαταρίας




Σχήμα 3.6: Παλμοτροφοδοτικό του φορτιστή

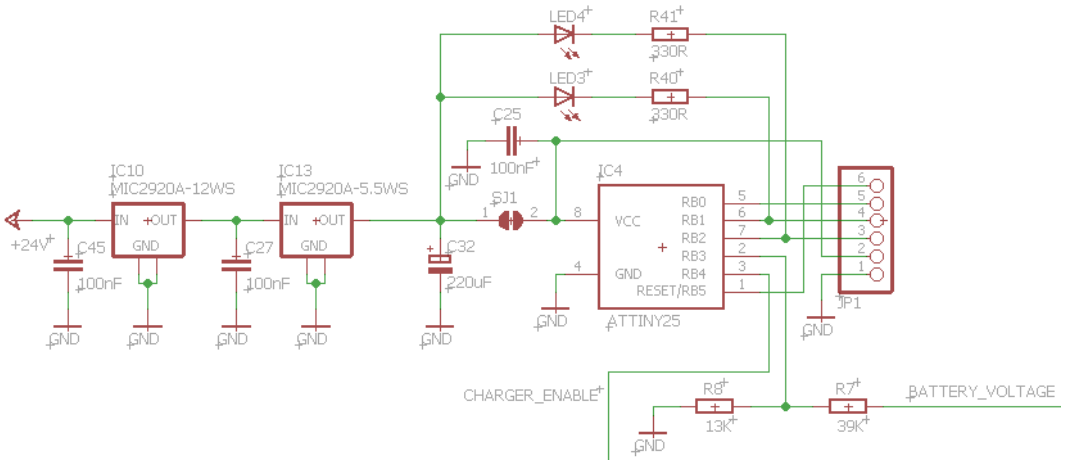
Για το τροφοδοτικό το οποίο είναι υπεύθυνο για την φόρτιση της μπαταρίας χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα L7987L της εταιρίας ST Microelectronics. Αυτό το κύκλωμα έχει τάση εισόδου από 4,5V έως 61V και τάση εξόδου από 0,8V έως την τάση τροφοδοσίας με μέγιστη έξοδο ρεύματος 2A. Επίσης έχει ρυθμιζόμενο όριο ρεύματος, ρυθμιζόμενη ομαλή έναρξη και ρυθμιζόμενη συχνότητα παλμού. Τέλος, έχει δυνατότητα ενεργοποίησης – απενεργοποίησης μέσω του enable pin που διαθέτει.

Η ρύθμιση της τάσης εξόδου γίνεται με δύο αντιστάσεις σε συνδεσμολογία διαιρέτη τάσης και ο υπολογισμός γίνεται με τον τύπο [5]:

$$R4 = \frac{(Vout-0{,}8)*R6}{0{,}8} = \frac{(15-0{,}8)*2{,}2}{0{,}8} = 39K\Omega \quad (3.1)$$

Η ρύθμιση του ορίου του ρεύματος γίνεται μέσω μίας αντίστασης η οποία υπολογίζεται με τον τύπο [5]:

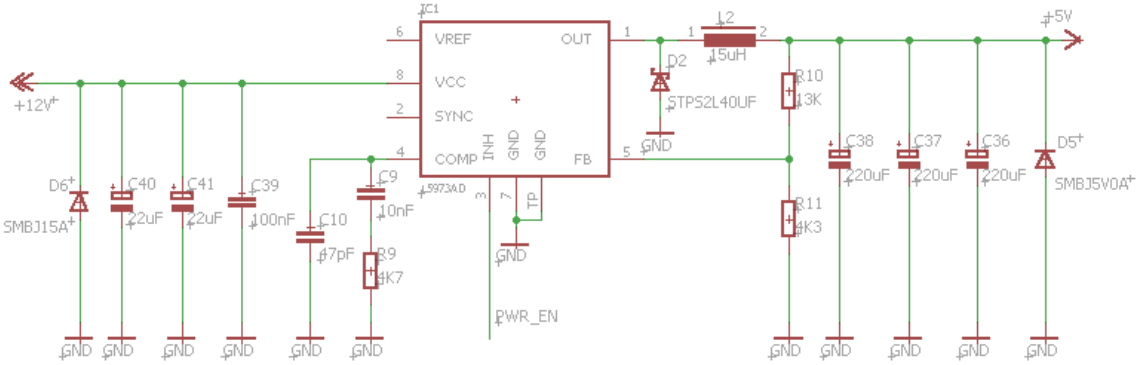
$$R2 = 27K\Omega\,\frac{3A}{I_{LIM}} = 27K\Omega\,\frac{3A}{2A} = 39K\Omega \quad (3.2)$$




Σχήμα 3.7: Controller ελέγχου φόρτισης

Ο φορτιστής διαθέτει έναν controller για την ομαλή φόρτιση της μπαταρίας. Κατά την διάρκεια της φόρτισης ελέγχει την κατάσταση της μπαταρίας μετρώντας την τάση της και ρυθμίζει τον χρόνο ενεργοποίησης του τροφοδοτικού μέσω του enable pin. Η ένδειξη της κατάστασης λειτουργίας του τροφοδοτικού γίνεται μέσο δύο led.

### 3.6.2 Κυρίως τροφοδοτικό

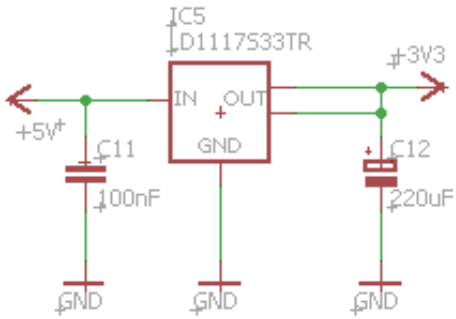



Σχήμα 3.8 Switching τροφοδοτικό 5 Volt

Το κύριο τροφοδοτικό του συστήματος είναι ένα παλμικό τροφοδοτικό τάσης εξόδου 5 Volt. Για την κατασκευή του χρησιμοποιήθηκε το chip L5973AD της εταιρίας ST Microelectronics το οποίο η τάση τροφοδοσίας του είναι από 4V έως 36V ενώ η τάση εξόδου του μπορεί να ρυθμιστεί από 1,235V έως 35V. Το μέγιστο ρεύμα εξόδου είναι τα 2A και η συχνότητα παλμού είναι σταθερή στα 500kHz.

Η ρύθμιση της τάσης εξόδου γίνεται με δύο αντιστάσεις σε συνδεσμολογία διαιρέτη τάσης και ο υπολογισμός γίνεται με τον τύπο [4]:

$$R10 = \frac{(Vout - 1{,}235) * R11}{1{,}235} = \frac{(5 - 1{,}235) * 4{,}3}{1{,}235} = 13K\Omega \qquad (3.3)$$




Σχήμα 3.9: Τροφοδοτικό 3.3 Volt

Η τάση λειτουργίας του μικροελεγκτή είναι 3.3V. Έτσι τοποθετήθηκε ένας σταθεροποιητής τάσης σε ολοκληρωμένο κύκλωμα. Ο σταθεροποιητής είναι ο LD1117 της εταιρίας ST Microelectronics που μας παρέχει τάση 3.3V με μέγιστο ρεύμα εξόδου 0.8A.

## 3.7 Επίλογος

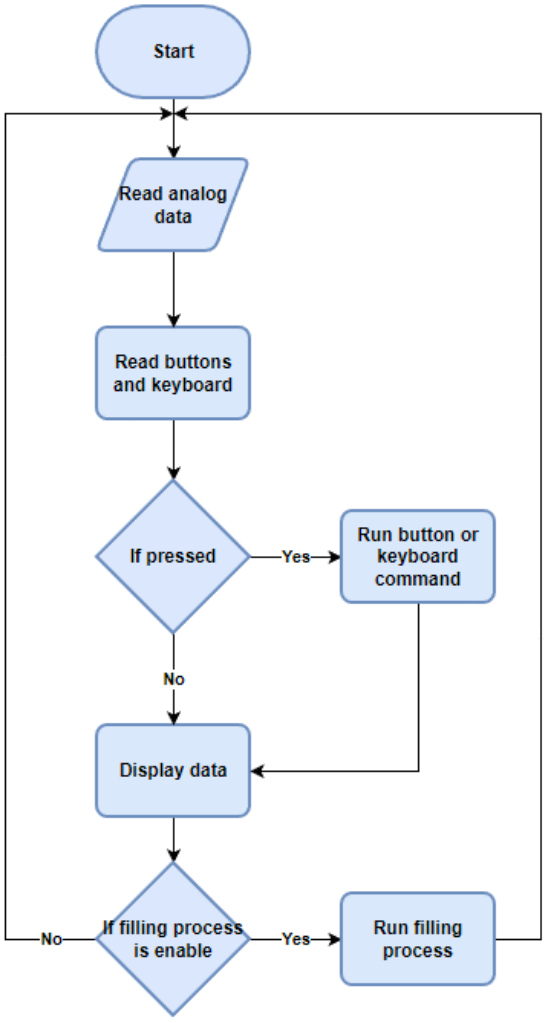
Σε αυτό το κεφάλαιο αναφέρθηκαν όλα τα σημαντικά χαρακτηρίστηκα του ESP32 και ο λόγος για τον οποίο επιλέχθηκε ο συγκεκριμένος μικτοελεγκτής σε αυτό το σύστημα. Αναλύθηκαν τα κυκλώματα που εισάγουν τις εντολές και τα δεδομένα από τον χρήστη καθώς και ο τρόπος που εμφανίζονται τα δεδομένα στους ενδείκτες και πως εκτελούνται οι εντολές γεμίσματος. Τέλος, έγινε αναφορά στο κύκλωμα επικοινωνίας του μικροελεγκτή με τον υπολογιστή μέσω θύρας USB και αναλύθηκαν τα κυκλώματα του φορτιστή της μπαταρίας και τα κυκλώματα της τροφοδοσίας όλων των εξαρτημάτων του συστήματος.

# ΚΕΦΑΛΑΙΟ 4

# Λογισμικό μικροελεγκτή

## 4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθούν όλες οι λειτουργίες που εκτελούνται από τον μικροελεγκτή. Θα γίνει αναφορά στου τρόπους ανάγνωσης και επεξεργασίας των εντολών και των δεδομένων. Τέλος, θα γίνει λεπτομερής ανάλυση της διαδικασίας γεμίσματος των δοχείων και του υπολογισμού τη ροής του χημικού υγρού. Στο παρακάτω διάγραμμα φαίνονται τα στάδια τα οποία εκτελούνται από το σύστημα.

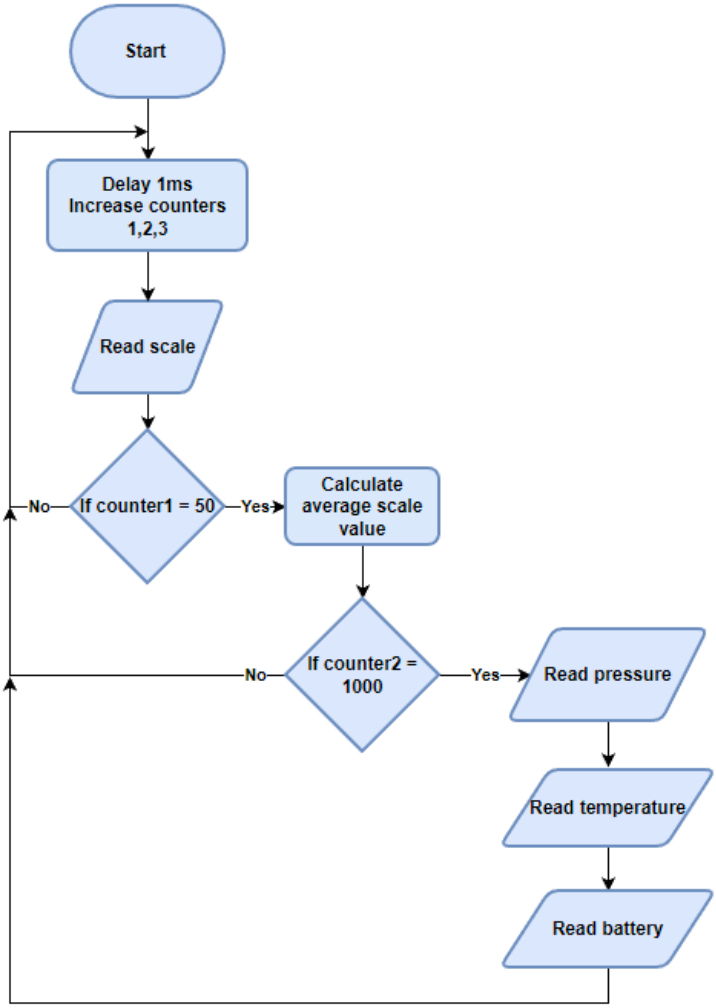



Σχήμα 4.1: Συνολικό διάγραμμα του κώδικα

## 4.2 Δεδομένα και εντολές

Τα δεδομένα που εισάγονται προς επεξεργασία στον μικροελεγκτή είναι δύο ειδών. Πρώτον είναι τα δεδομένα που εισάγονται από τους αισθητήρες. Δεύτερον είναι τα δεδομένα που εισάγονται από τον χρήστη, είτε μέσω μπουτόν είτε μέσω του πληκτρολογίου.

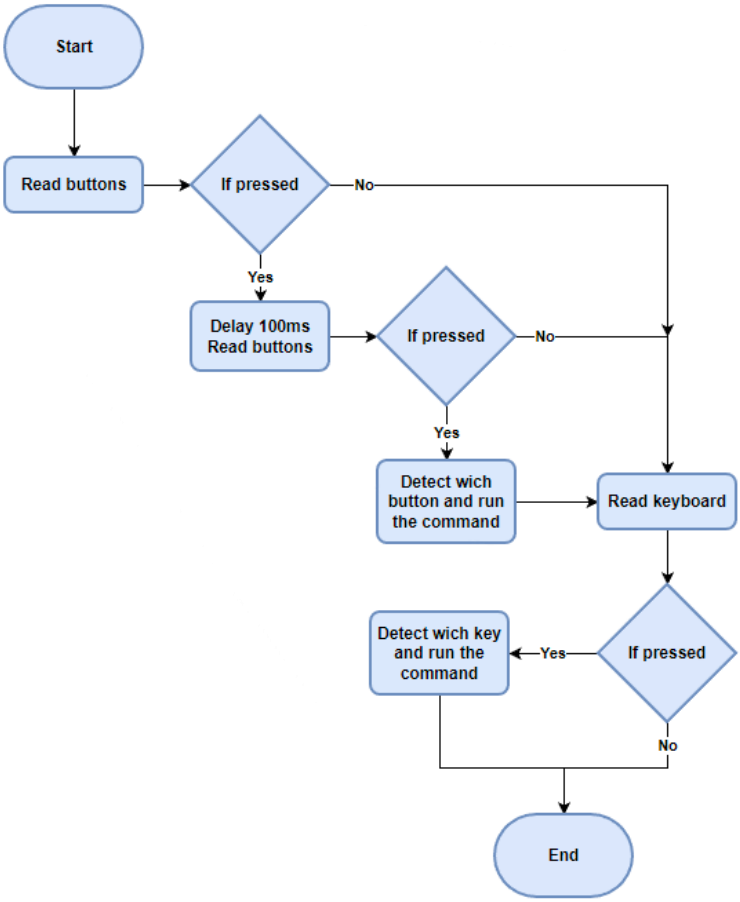
### 4.2.1 Ανάγνωση αναλογικών δεδομένων




Σχήμα 4.2: Διάγραμμα διαδικασίας ανάγνωσης αναλογικών δεδομένων

Η ανάγνωση των αναλογικών δεδομένων γίνεται με βάση δύο μετρητές που αυξάνονται με μία καθυστέρηση όπως φαίνεται και στο παραπάνω σχήμα. Οι μετρητές αυξάνονται κάθε 1 χιλιοστό του δευτερολέπτου όπως επίσης γίνεται και μέτρηση του βάρους. Όταν ο πρώτος μετρητής φτάσει την τιμή 10, που σημαίνει ότι έγιναν 10 μετρήσεις βάρους, τότε υπολογίζεται ο μέσος όρος και μηδενίζεται ο πρώτος μετρητής. Με αυτόν το τρόπο επιτυγχάνεται καλύτερη σταθερότητα στην μέτρηση του βάρους. Στην συνέχεια, όταν ο δεύτερος μετρητής φτάσει την τιμή 1000, δηλαδή όταν θα έχει περάσει 1 δευτερόλεπτο από την προηγουμένη μέτρηση, τότε γίνεται μέτρηση της πίεσης, της θερμοκρασίας και της τάσης της μπαταρίας, και μηδενίζεται ο δεύτερος μετρητής. Αυτό συμβαίνει διότι αυτές οι τρείς μετρήσεις είναι δευτερεύουσες σε σύγκριση με το βάρος, οπότε δεν χρειάζεται να επιβαρύνουν την λειτουργία του συστήματος με συχνές μετρήσεις.

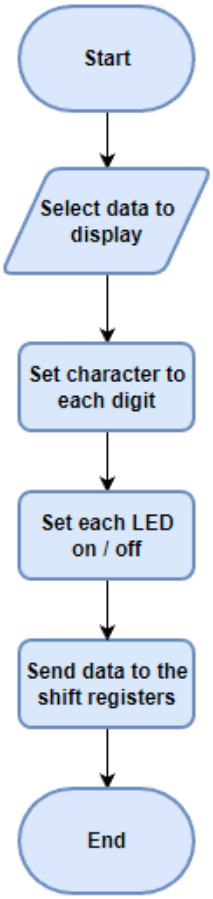
### 4.2.2 Εντολές από μπουτόν και πληκτρολόγιο




Σχήμα 4.3: Διάγραμμα ανάγνωση μπουτόν και πληκτρολογίου

Οι εντολές και τα δεδομένα που εισάγονται από τον χρήστη στο σύστημα γίνεται με δύο τρόπους. Ο πρώτος τρόπος είναι με μπουτόν. Όταν το σύστημα εντοπίσει ότι πατήθηκε κάποιο μπουτόν, τότε μετά από μία καθυστέρηση 0,1 δευτερόλεπτα επανελέγχει τα μπουτόν. Αυτό συμβαίνει για να εξαλειφθούν πιθανοί θόρυβοι που μπορεί να ενεργοποιήσουν κάποια εντολή χωρίς να πρέπει. Ο δεύτερος τρόπος είναι με ένα αριθμητικό πληκτρολόγιο. Ο εντοπισμός του κουμπιού του πληκτρολογίου που πατήθηκε γίνετε με την διαδικασία της σάρωσης. Στο πληκτρολόγιο ένα από τα κουμπιά παίζει το ρόλο shift. Έτσι, όταν είναι πατημένο, όλα τα υπόλοιπα κουμπιά εκτελούν επιπλέον λειτουργίες.

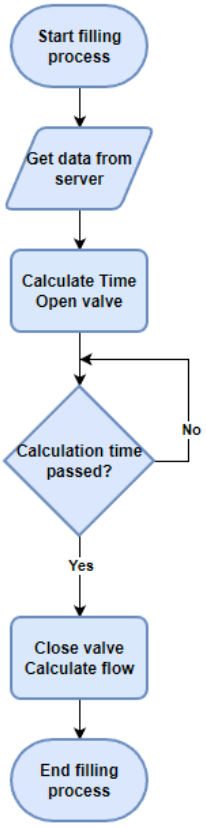
## 4.3 Εμφάνιση δεδομένων




Σχήμα 4.4: Διάγραμμα εμφάνισης δεομένων στα ψηφία

Τα δεδομένα εμφανίζονται μέσω πέντε ψηφίων. Στην αρχή το σύστημα επιλέγει τα δεδομένα που πρέπει να εμφανιστούν με βάση το μενού που έχει επιλεγεί από τον χρήστη. Στην συνέχεια γίνεται η κατάλληλη μετατροπή του αριθμού στην σωστή μονάδα μέτρησης. Έπειτα, με βάση αυτόν το αριθμό γίνεται η επιλογή του χαρακτήρα που θα εμφανιστεί στο καθένα από τα πέντε ψηφία. Το επόμενο βήμα είναι να ανάψουν τα κατάλληλα LED στα οποία φαίνεται η στάθμη της μπαταρίας, η κατάσταση λειτουργίας της συσκευής και η επιλεγμένη μονάδα μέτρησης. Τέλος όλα τα παραπάνω στοιχεία αποστέλλονται στους καταχωρητές για να εμφανιστούν. Όλη αυτή η διαδικασία επαναλαμβάνεται κάθε 100 χιλιοστά του δευτερολέπτου.

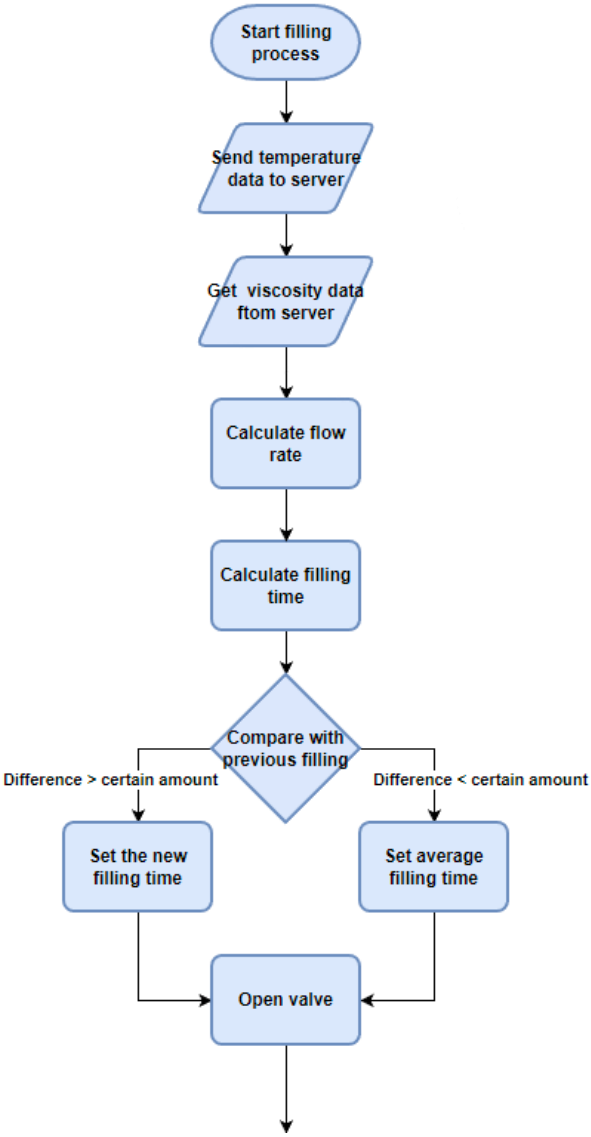
## 4.4 Διαδικασία γεμίσματος




Σχήμα 4.5: Διάγραμμα γενικής διαδικασία γεμίσματος

Η διαδικασία γεμίσματος είναι η κυριότερη λειτουργία του συστήματος. Σε αυτή την λειτουργία το σύστημα αναλαμβάνει να υπολογίζει από την αρχή της διαδικασίας μέχρι το τέλος όλες τις παραμέτρους που εισέρχονται για πετύχει τον στόχο που είναι το ακριβές γέμισμα των δοχείων. Αυτή η λειτουργία αποτελείται από δύο μέρη. Το πρώτο μέρος αναφέρεται στις λειτουργίες που εκτελούνται από την στιγμή που δοθεί εντολή έναρξης της διαδικασίας από τον χρήστη μέχρι την στιγμή που θα αρχίσει η ροή του υγρού. Το δεύτερο μέρος αναφέρεται στις λειτουργίες που εκτελούνται κατά την διάρκεια ροής του υγρού μέχρι το κλείσιμο της βαλβίδας και τον τελικό υπολογισμό.

### 4.4.1 Συλλογή δεδομένων από τον server




Σχήμα 4.6: Διάγραμμα συλλογής δεδομένων και έναρξης διαδικασίας

Κατά την εκκίνηση της διαδικασίας γεμίσματος το σύστημα στέλνει στον server την τιμή της θερμοκρασίας του χημικού υγρού που επιλέχθηκε. Εκεί γίνεται σύγκριση με τους πίνακες που έχουν αποθηκευτεί από τον χρήστη και αποστέλλεται πίσω στην συσκευή η τιμή της πυκνότητας του χημικού υγρού για την συγκεκριμένη τιμή θερμοκρασίας. Με βάση την τιμή της πυκνότητας και την τιμή της πίεσης που μετρήθηκε, η οποία βασίζεται στην ποσότητα του χημικού υγρού που βρίσκεται μέσα στην δεξαμενή, υπολογίζεται η ενδεικτική ροή και στην συνέχεια και ο εκτιμώμενος χρόνος που θα χρειαστεί για να γεμίσει η ζητούμενη ποσότητα. Ο χρόνος υπολογίζεται με τον τύπο:

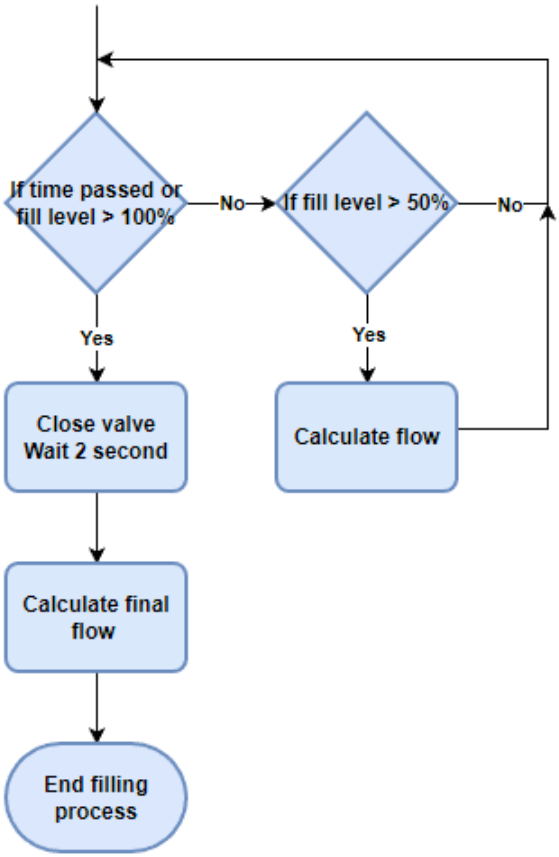
$$Time = \frac{TargetVolume * 1000}{Flow} \quad (4.1)$$

Όπου Time είναι ο χρόνος (ms), TargetVolume είναι η επιθυμητή τιμή γεμίσματος (gr) και το Flow είναι η ροή (gr/s).

Εάν η διαδικασία βρίσκεται στο πρώτο δοχείο ή ο χρόνος που υπολογίστηκε είναι πολύ διαφορετικός από τον χρόνο που χρειάστηκε το προηγούμενο δοχείο, τότε ορίζεται η καινούρια τιμή θεωρώντας την

προηγούμενη ως εσφαλμένη. Εάν η διαδικασία βρίσκεται σε εξέλιξη και η διαφορά του καινούριου χρόνου με τον προηγούμενο είναι ανάμεσα σε κάποια λογικά πλαίσια τότε υπολογίζεται ο μέσος όρος των δύο τιμών. Τέλος, ενεργοποιείται η βαλβίδα για να ξεκινήσει το γέμισμα του δοχείου.

### 4.4.2 Έλεγχος διαδικασίας και ολοκλήρωση γεμίσματος




Σχήμα 4.7: Διάγραμμα ελέγχου και ολοκλήρωσης διαδικασίας

Καθ' όλη την διάρκεια του γεμίσματος του δοχείου, γίνεται έλεγχος εάν η ποσότητα του χημικού υγρού έχει ξεπεράσει την επιθυμητή ποσότητα γεμίσματος και αυτό συμβαίνει για να αποφευχθεί η περίπτωση υπερχείλισης. Όταν η ποσότητα του χημικού υγρού φτάσει στο μέσο του δοχείου, γίνεται έλεγχος του χρόνου που χρειάστηκε για να γεμίσει αυτή η ποσότητα και υπολογίζεται ο χρόνος που υπολείπεται για την ολοκλήρωση του γεμίσματος. Η βαλβίδα κλείνει όταν συμπληρωθεί αυτός ο χρόνος. Έπειτα, γίνεται υπολογισμός της πραγματικής ροής και γίνεται σύγκριση της τελικής ποσότητας με την επιθυμητή ποσότητα του χημικού υγρού για τον υπολογισμό του σφάλματος. Ο υπολογισμός της πραγματικής ροής γίνεται με τον τύπο:

$$Flow = \frac{ScaleValue * 1000}{TimePassed} \quad (4.2)$$

Όπου Flow είναι η ροή (gr/s), ScaleValue είναι το βάρος του τελικού ποσού γεμίσματος (gr) και TimePassed ο χρόνος που χρειάστηκε για να ολοκληρωθεί το γέμισμα (ms).

Επίσης, υπάρχει ένα μετρητής που ελέγχει τον ρυθμό πτώσης της ροής από δοχείο σε δοχείο. Η τιμή της ροής, ο ρυθμός πτώσης και το σφάλμα που υπολογίστηκαν θα χρησιμοποιηθούν στο γέμισμα του επόμενου δοχείου για να επιτευχθεί μεγαλύτερη ακρίβεια.

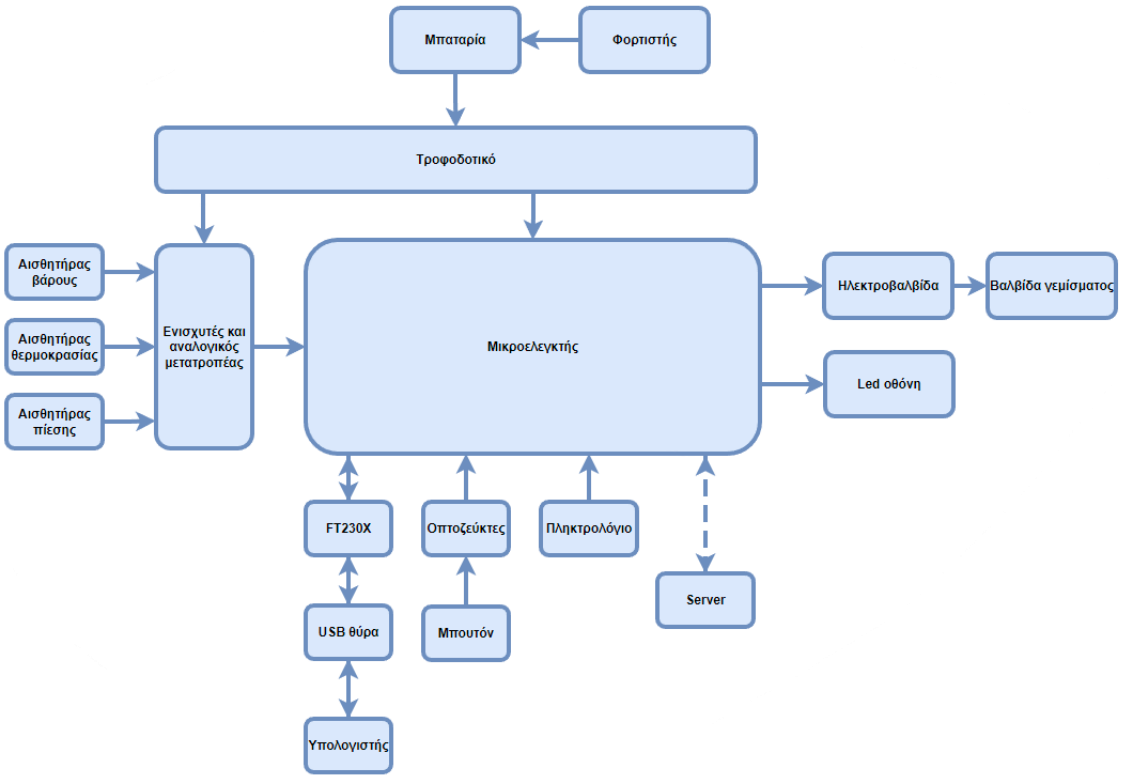
## 4.5 Επίλογος

Σε αυτό το κεφάλαιο έγινε αναλυτική αναφορά στα στάδια που ακολουθεί το λογισμικό του μικροελεγκτή. Στην αρχή έγινε ανάλυση στον τρόπο ανάγνωσης των αναλογικών δεδομένων από τους αισθητήρες. Έπειτα έγινε αναφορά στην διαδικασία ανάγνωσης των μπουτόν και του πληκτρολογίου, και στον τρόπο αποφυγής λανθασμένων εντολών. Έγινε αναφορά στον τρόπο εμφάνισης των δεδομένων στην οθόνη. Τέλος, έγινε λεπτομερής ανάλυση της σημαντικότερης διαδικασίας του συστήματος η οποία είναι το γέμισμα των δοχείων. Αναλύθηκε ο τρόπος συλλογής δεδομένων από τον server και υπολογισμού του εκτιμώμενου χρόνου γεμίσματος. Έγινε αναφορά στην μέθοδο υπολογισμού της πραγματικής ροής και του σφάλματος γεμίσματος για την επίτευξη καλύτερων αποτελεσμάτων.

# ΚΕΦΑΛΑΙΟ 5

## Κατασκευή και αποτελέσματα

### 5.1 Εισαγωγή

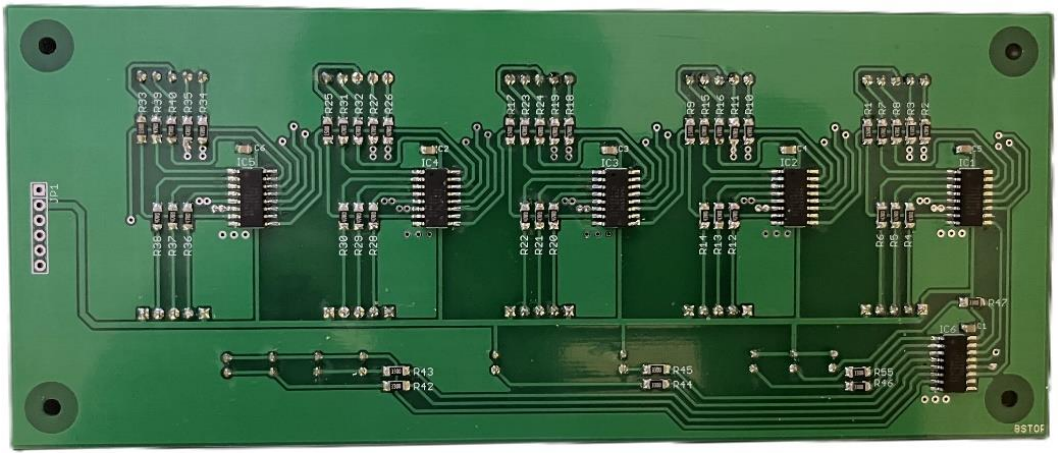



Σχήμα 5.1: Μπλοκ διάγραμμα όλων των συσκευών

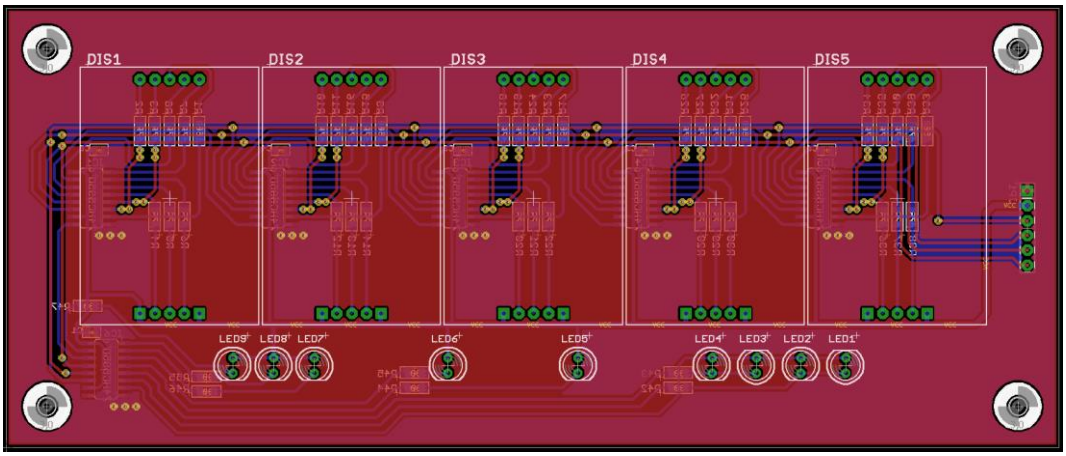
## 5.2 Ηλεκτρονικές πλακέτες του συστήματος

Το ηλεκτρονικό μέρος της κατασκευής αποτελείται από δύο πλακέτες τοποθετημένες η μία πάνω από την άλλη. Η πρώτη πλακέτα είναι αυτή που είναι υπεύθυνη για την ένδειξη των αποτελεσμάτων στον χρήστη. Στην μπροστινή μεριά της πλακέτας τοποθετήθηκαν τα ψηφία και τα led για τις ενδείξεις, και στο πίσω μέρος τοποθετήθηκαν οι καταχωρητές και τα υπόλοιπα υλικά. Παρακάτω φαίνεται η μπροστινή και η πίσω όψη της πλακέτας οθόνης.


Εικόνα 5.1: Μπροστινή όψη πλακέτας οθόνης

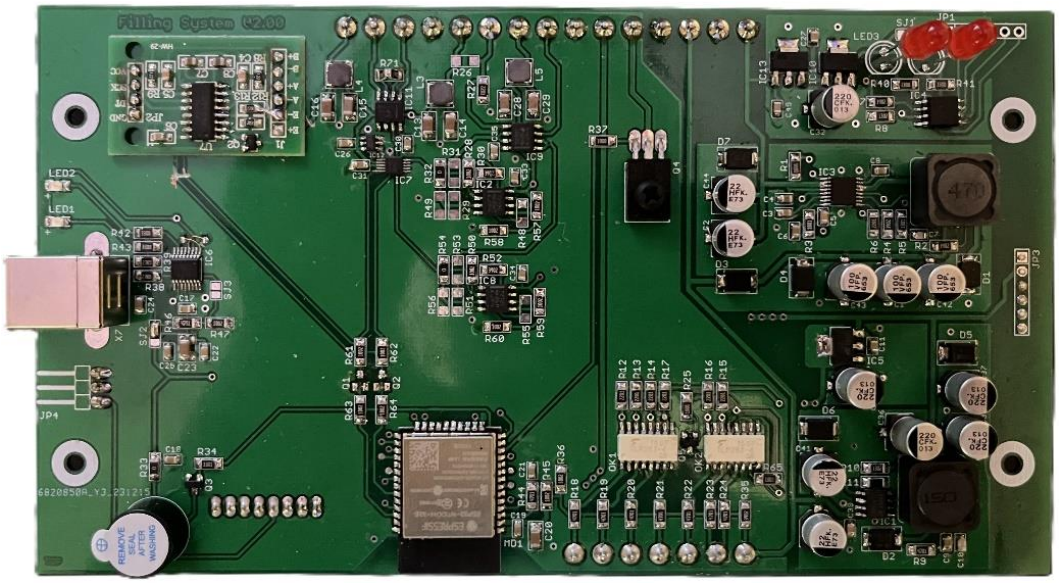


Εικόνα 5.2: Πίσω όψη πλακέτας οθόνης

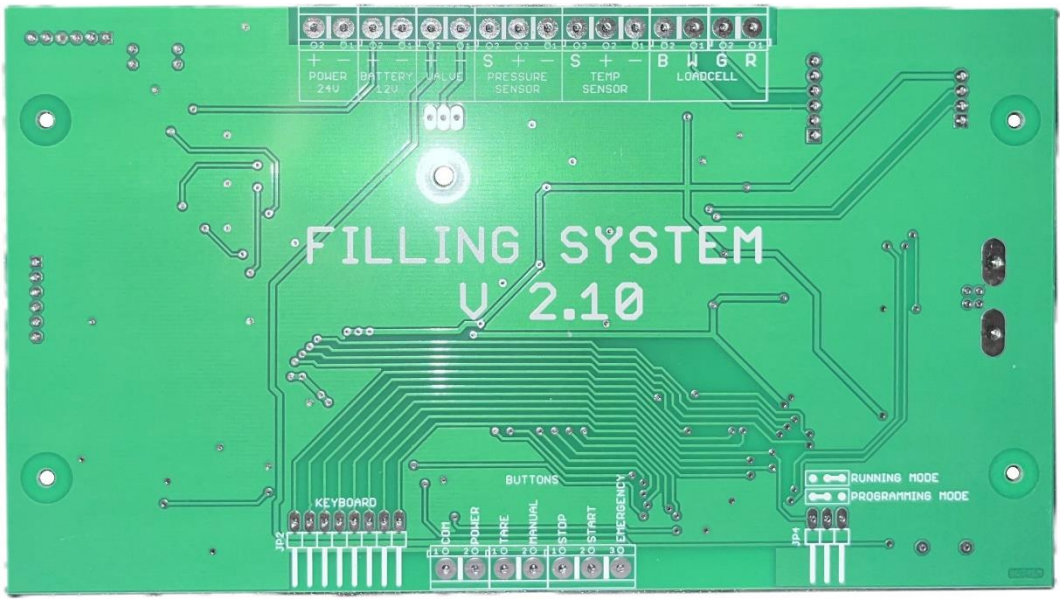



Εικόνα 5.3: Πλακέτα οθόνης στο σχεδιαστικό πρόγραμμα

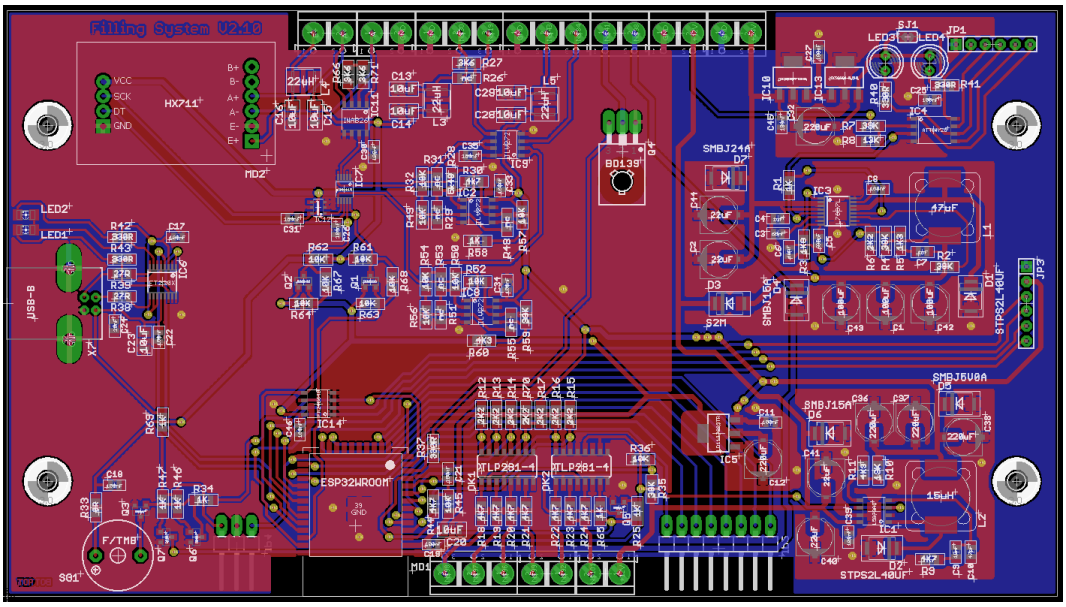
Στην δεύτερη πλακέτα, η οποία είναι και η κύρια πλακέτα, βρίσκονται όλα τα υπόλοιπα κυκλώματα που είναι απαραίτητα για την λειτουργία της συσκευής. Ο σχεδιασμός των πλακετών έγινε στο σχεδιαστικό πρόγραμμα Eagle της εταιρία Autodesk και όλα τα υλικά που χρησιμοποιήθηκαν, και στις δύο πλακέτες, είναι τεχνολογίας SMD. Παρακάτω φαίνεται η μπροστινή και η πίσω όψη τη κύριας πλακέτας καθώς και τα επιμέρους κυκλώματα που την απαρτίζουν.



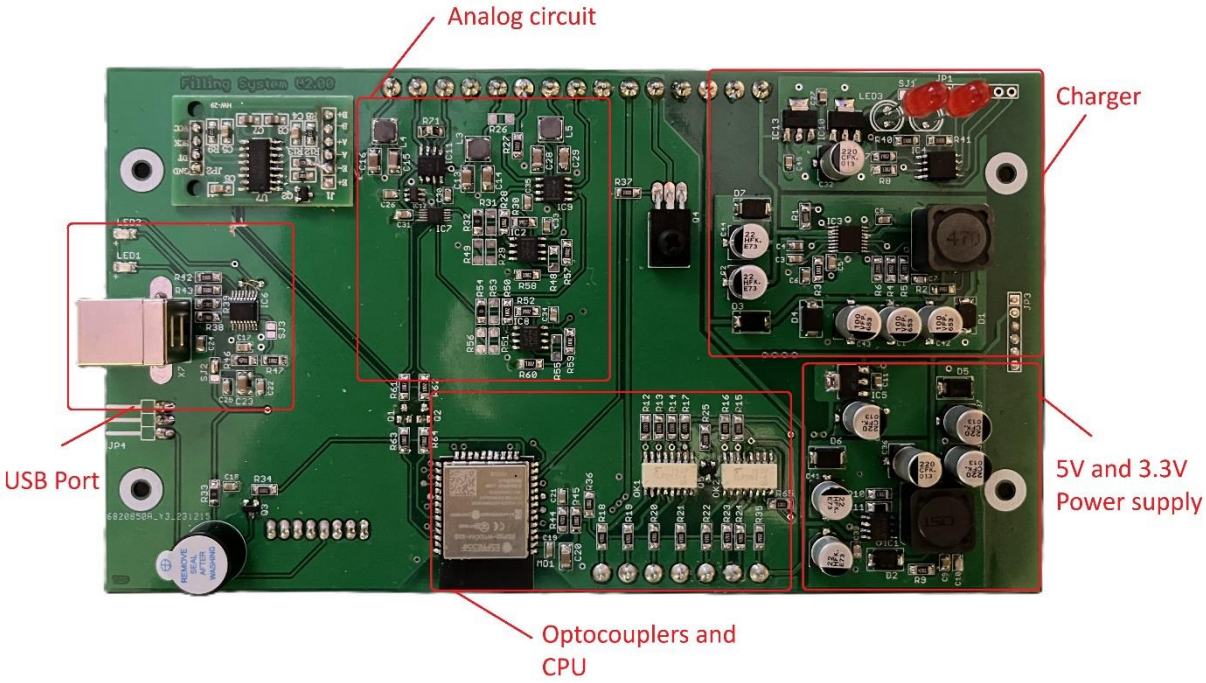
Εικόνα 5.4: Μπροστινή όψη κύριας πλακέτας



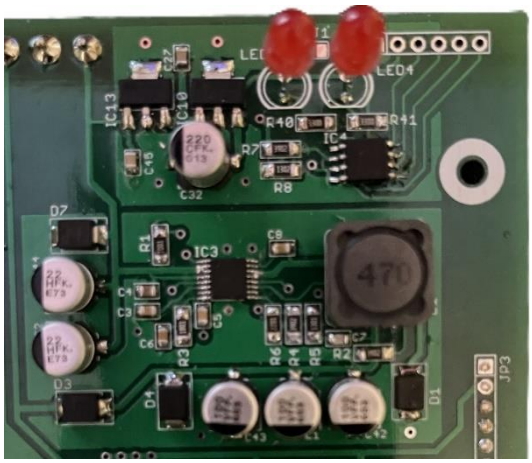

Εικόνα 5.5: Πίσω όψη κύριας πλακέτας




Εικόνα 5.6: Κύρια πλακέτα στο σχεδιαστικό πρόγραμμα

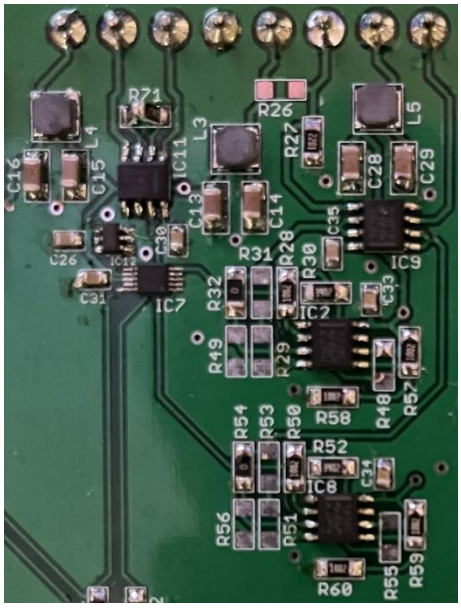



Εικόνα 5.7: Αναφορά διατάξεων πλακέτας

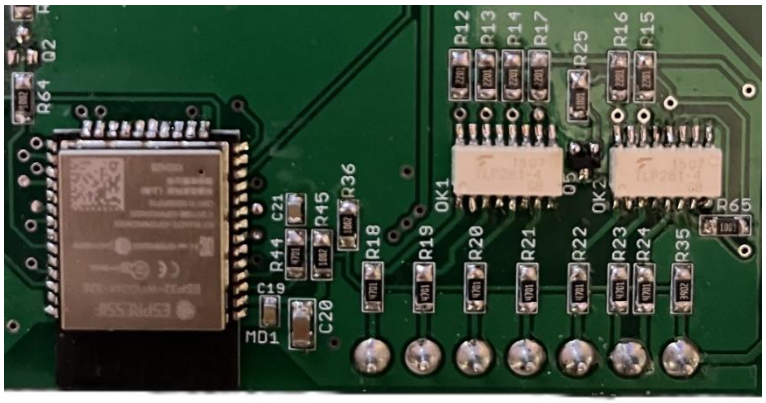



Εικόνα 5.8: Κύκλωμα φορτιστή μπαταρίας

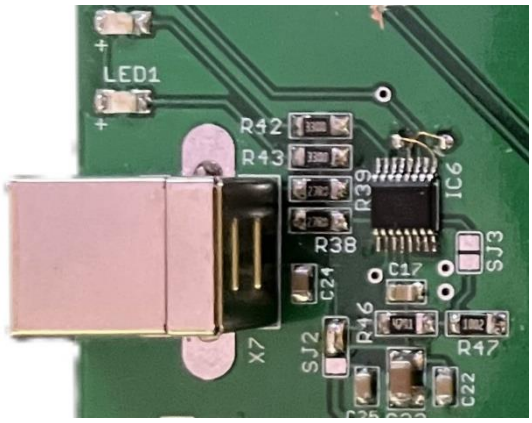

Εικόνα 5.9: Κύκλωμα κύριου τροφοδοτικού




Εικόνα 5.10: Αναλογικό κύκλωμα




Εικόνα 5.11: Κύκλωμα μικροελεγκτή και οπτοζεύκτες




Εικόνα 5.12: Κύκλωμα επικοινωνίας μέσω USB

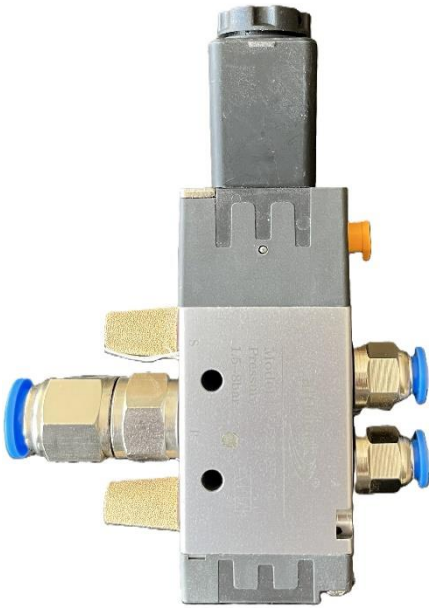
## 5.3 Εξωτερικά εξαρτήματα συστήματος


Εικόνα 5.13: Κουτί κατασκευής


Εικόνα 5.14: Εσωτερικό κουτιού κατασκευής

Εικόνα 5.15: Βαλβίδα ενεργοποίησης γεμίσματος

Το γέμισμα των δοχείων πραγματοποιείται από μία ειδική βαλβίδα η οποία έχει την ιδιότητα να κλείνει σχεδόν ακαριαία, και κατά το κλείσιμό της να μην επιτρέπει να πέσει καθόλου επιπλέον υγρό λόγω αποστράγγισης όπως συμβαίνει με άλλες βαλβίδες. Ο έλεγχος γίνεται με αέρα για δύο λόγους. Ο πρώτος είναι για να πετύχουμε το πολύ γρήγορο κλείσιμο και ο δεύτερος είναι να αποφύγουμε πιθανούς σπινθηρισμούς του ρεύματος οι οποίοι θα μπορούσαν να δημιουργήσουν ανάφλεξη σε κάποιο εύφλεκτο χημικό υγρό. Ο έλεγχος αυτής της βαλβίδας γίνεται με μία ηλεκτροβαλβίδα τοποθετημένη σε μία ασφαλή απόσταση από το χημικό υγρό.



Εικόνα 5.16: Ηλεκτροβαλβίδα ενεργοποίησης κυκλώματος αέρα

Εικόνα 5.17: Βάση ζύγισης


Εικόνα 5.18: Ολοκληρωμένη διάταξη κατασκευής

## 5.4 Προβλήματα και αντιμετώπιση

Κατά τη διάρκεια των δοκιμών ανέκυψαν ορισμένα προβλήματα. Τα χημικά υγρά με μεγάλη ρευστότητα, κατά το άνοιγμα της βαλβίδας δημιουργούσαν μία στιγμιαία μεγάλη μέτρηση στο βάρος λόγο της απότομης εκτόνωσης της πίεσης. Αυτό, στην περίπτωση μικρής ζητούμενης ποσότητας γεμίσματος, δημιουργούσε πρόβλημα εμφανίζοντας στιγμιαία υπερχείλιση του δοχείου. Έτσι, η βαλβίδα έκλεινε πολύ πιο γρήγορα από το αναμενόμενο με αποτέλεσμα να μην τοποθετείται η σωστή ποσότητα. Για να αποφευχθεί αυτό το πρόβλημα θα πρέπει να τοποθετηθεί ρυθμιστής ροής πριν από την βαλβίδα εξόδου έτσι ώστε να μειωθεί μεγάλη αρχική πίεση θυσιάζοντας βέβαια τον χρόνο διότι με αυτό τον τόπο η διαδικασία γεμίσματος των δοχείων θα διαρκεί περισσότερο.

# Βιβλιογραφία

**Βιβλία**

[1] T. G. Mezger, The Rheology Handbook. Vincentz Network, Cop, 2014.

[2] Μ. Σπάσος, Αναλογική Επεξεργασία Σημάτων Αισθητήρων. Εκδόσεις Αϊβάζη, 2018.

[3] A. Malvino and D. J. Bates, Ηλεκτρονική, Αρχές και Εφαρμογές. Εκδόσεις Τζιόλα, 2012.

**Data Sheet**

[4] ST Microelectronics, “2A switch step-down switching regulator,” L5973AD datasheet, November 2009.

[5] ST Microelectronics, “2A asynchronous step-down switching regulator with adjustable current limitation,” L7987L datasheet, May 2020.

[6] Texas Instruments, “Instrumentation Amplifier with Rail-to-Rail Output,” INA826 datasheet, August 2011.

[7] Texas Instruments, “16-Bit ADCs with Internal Reference, Oscillator, and Programmable Comparator,” ADS1115 datasheet, May 2009.

**Internet sites**

[8] Georgiev G.Z., "*Flow Rate Calculator*", [online] Available: https://www.gigacalculator.com/calculators/pipe-flow-rate-calculator.php

[9] https://grobotronics.com/

# Παράρτημα Α:

```
#define Display_G 23 //display britness
#define Display_SER 19 //display data
#define Display_RCK 18 //display read enable
#define Display_SCK 5 //display clock
#define KeypadRow1 17
#define KeypadRow2 16
#define KeypadRow3 4
#define KeypadRow4 0
#define KeypadCol1 2
#define KeypadCol2 15
#define KeypadCol3 13
#define KeypadCol4 12
#define Buzzer 14
#define Valve 27
#define PowerEnable 26
#define PowerButton 25
#define TareButton 33
#define ManualFillingButton 32
#define StopFillingButton 35
#define StartFillingButton 34
#define EmergencyButton 39
#define Battery 36

#include <EEPROM.h>
#include <Shift_Keypad.h>

#include "Analog.h"
#include "Buttons.h"
#include "Display.h"
#include "FillingProcess.h"
#include "Parameters.h"
```

```
#ifdef Sensor_HX711
#define LoadCell_DT 22 //loadcell data
#define LoadCell_SCK 21 //loadcell clock
#include <HX711_ADC.h>
HX711_ADC LoadCell(LoadCell_DT, LoadCell_SCK);
#endif

#ifdef Sensor_ADS1115
#include <ADS1X15.h>
ADS1115 AnalogConverter(0x48);
#endif

Shift_Keypad Keypad0(RowPins, ColumPins, Rows, Colums, makeKeymap(keypadButtons), makeKeymap(keypadShiftButtons), shiftButton);

int i, j;
int startFlag = 0;
int eepromAdress(int a) {
  int b = a * 4;
  return b;
}

void setup() {
  pinMode(Display_G, OUTPUT);
  pinMode(Display_SER, OUTPUT);
  pinMode(Display_RCK, OUTPUT);
  pinMode(Display_SCK, OUTPUT);
  pinMode(Valve, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(PowerEnable, OUTPUT);
  pinMode(PowerButton, INPUT);
  pinMode(TareButton, INPUT);
```

```
  _5VoltIn = analogRead(ManualFillingButton);
  pinMode(ManualFillingButton, INPUT);
  pinMode(StopFillingButton, INPUT);
  pinMode(StartFillingButton, INPUT);
  pinMode(EmergencyButton, INPUT);
  EEPROM.begin(512);
  Serial.begin(9600);
  initSensors();
  digitalWrite(PowerEnable, HIGH);
  digitalWrite(Valve, LOW);
  EEPROM.get(eepromAdress(0), calibrationValue);
  EEPROM.get(eepromAdress(1), unitsFlag);
  EEPROM.get(eepromAdress(2), tUnitFlag);
  EEPROM.get(eepromAdress(3), pipeLength);
  EEPROM.get(eepromAdress(4), pipeDiameter);
  EEPROM.get(eepromAdress(5), backlightValue);
  if(unitsFlag > 3 || unitsFlag < 0){
    unitsFlag = 1;
    EEPROM.put(eepromAdress(1), unitsFlag);
  }
  if(tUnitFlag > 1 || tUnitFlag < 0){
    tUnitFlag = 0;
    EEPROM.put(eepromAdress(2), tUnitFlag);
  }
  if(backlightValue != 2 || backlightValue != 4 || backlightValue != 6 || backlightValue != 8 ||
backlightValue != 10){
    backlightValue = 6;
    EEPROM.put(eepromAdress(5), backlightValue);
  }
  EEPROM.commit();
#ifdef WIFI
  //
#endif
```

```
#ifdef Sensor_HX711
  LoadCell.begin();
  LoadCell.start(1000);
  LoadCell.setCalFactor(calibrationValue);
#endif
#ifdef Sensor_ADS1115
  AnalogConverter.begin();
  AnalogConverter.setGain(2);
#endif
  while (digitalRead(PowerButton) == 0);
  ledOutput[7] = 0; ledOutput[6] = 0; ledOutput[5] = 0; ledOutput[4] = 0; ledOutput[3] = 0; ledOutput[2] = 1; ledOutput[1] = 1; ledOutput[0] = 1;
  for (j = 9; j >= 0; j--) {
    for (i = 4; i >= 0; i--) {
      dispNumber[i] = j;
    }
    send_register();
    Serial.println(j);
    delay(150);
  }
  dispNumber[4] = 30; dispNumber[3] = 30; dispNumber[2] = 30; dispNumber[1] = 30; dispNumber[0] = 30;
  ledOutput[2] = 0; ledOutput[1] = 1; ledOutput[0] = 1;
  send_register();
  delay(150);
  ledOutput[2] = 1; ledOutput[1] = 0; ledOutput[0] = 0;
  send_register();
  delay(150);
  ledOutput[2] = 1; ledOutput[1] = 0; ledOutput[0] = 1;
  send_register();
  delay(150);
  ledOutput[2] = 0; ledOutput[1] = 1; ledOutput[0] = 0;
  send_register();
  delay(150);
```

```
  ledOutput[7] = 1; ledOutput[6] = 1; ledOutput[5] = 1; ledOutput[4] = 1; ledOutput[3] = 1; ledOutput[2] = 1; ledOutput[1] = 1; ledOutput[0] = 1;
  dispNumber[4] = 21; dispNumber[3] = 21; dispNumber[2] = 21; dispNumber[1] = 21; dispNumber[0] = 21;
  send_register();
  delay(500);
}

void loop() {
  read_buttons();
  display_data();
  read_adcs();
  if (fillingStateFlag != 0) {
    fillingProcess();
  }
  if(startFlag == 0){
    myTare();
    startFlag = 1;
  }
}

long scaleValue = 0;
long temperatureValue = 0;
double temperatureValue0 = 0;
double pressureValue = 0;
double batteryValue = 0;
long scaleValue0 = 0;
long tareOffset = 0;
int reverceTimer = 0;
long calibrationKnownMass = 0;
float calibrationValue = 1.06;
bool holdState = false;
bool endState = false;
double scaleVar = 0;
```

```
double PressureVar = 0;
int _5VoltIn = 0;
double _5Volt = 0;
double _5mVolt = 0;
unsigned long adcTimerCounter = 0;
unsigned long adcCounter1 = 0;
unsigned long adcCounter2 = 0;

#ifdef Sensor_HX711
void read_adcs(void){
  if(LoadCell.update()){
    scaleValue = LoadCell.getData();
  }
  batteryValue = readBatteryVoltage();
}
void myTare(void){
  LoadCell.tareNoDelay();
}
#endif
#ifdef Sensor_ADS1115
void read_adcs(void){
  if(millis() - adcTimerCounter >= 1){
    scaleValue0 += readScale();
    if(++adcCounter1 == 50){
      scaleValue = scaleValue0 / 50;
      scaleValue0 = 0;
      adcCounter1 = 0;
    }
    if(++adcCounter2 == 1000){
      pressureValue = readPressureSensor();
      readThermometer();
      batteryValue = readBatteryVoltage();
      adcCounter2 = 0;
```

```
    }
    adcTimerCounter = millis();
  }
}
long readScale(void){
  long dataIn = AnalogConverter.readADC_Differential_0_3() + 32638 - tareOffset;
  double dataOut = dataIn * scaleVar * calibrationValue;
  return dataOut;
}
void readThermometer(void){
  long dataIn = AnalogConverter.readADC_Differential_1_3() + 32716;
  double dataA = (4096 * dataIn) / 65535;
  double dataB = (dataA - temperatureAmpB) / temperatureAmpA;
  double dataR = (((_5mVolt - dataB) * temperatureRsens) / dataB) * 0.96;
  readLookUpTable(dataR);
}
void readLookUpTable(long dataR){
  int a;
  int aR;
  int b;
  int bR;
  for(i = 0; i<= 30; i++){
    if(dataR >= temperatureSensorTable[i]){
      a = i;
      aR = temperatureSensorTable[a];
      b = i - 1;
      bR = temperatureSensorTable[b];
      i = 31;
    }
  }
  if(dataR - aR < bR - dataR){
    temperatureValue = a * 5;
  }else{
```

```
    temperatureValue = b * 5;
  }
  temperatureValue0 = (((dataR - aR) * 5.00) / (bR - aR)) + temperatureValue;
}
double readPressureSensor(void){
  long dataIn = AnalogConverter.readADC_Differential_2_3() + 32714;
  double dataOut = dataIn * PressureVar;
  return dataOut;
}
void myTare(void){
  long data0 = 0;
  for(int k = 0; k <= 9; k++){
    data0 += AnalogConverter.readADC_Differential_0_3() + 32638;
    delay(50);
  }
  tareOffset = data0 / 10;
}
#endif
void calibration(void){
  double data0;
  set_leds();
  for(i = 0; i <= 4; i++){
    ledOutput[i] = 1;
  }
  dispNumber[0] = 20;
  dispNumber[1] = 28;
  dispNumber[2] = 27;
  dispNumber[3] = 22;
  dispNumber[4] = 20;
  send_register();
  delay(2000);
  dispNumber[0] = 21;
  dispNumber[1] = 20;
```

```
dispNumber[2] = 10;
dispNumber[3] = 20;
dispNumber[4] = 21;
data0 = 0;
//holdState = true;
for(j = 9; j >= 0; j--){
  dispNumber[2] = j;
  send_register();
  myTare();
  data0 += tareOffset;
  delay(500);
}
tareOffset = data0 / 10;
holdState = true;
while(holdState == true){
  read_keypad();
  display_data();
}
if(endState == false){
#ifdef Sensor_HX711
LoadCell.update();
LoadCell.refreshDataSet();
calibrationValue = LoadCell.getNewCalibration(calibrationKnownMass);
#endif
#ifdef Sensor_ADS1115
dispNumber[0] = 21;
dispNumber[1] = 20;
dispNumber[2] = 10;
dispNumber[3] = 20;
dispNumber[4] = 21;
data0 = 0;
for(j = 9; j >=0; j--){
  dispNumber[2] = j;
```

```
    send_register();
    for(i = 9; i >= 0; i--){
      data0 += AnalogConverter.readADC_Differential_0_3() + 32638 - tareOffset;
      delay(100);
    }
  }
  Serial.println(data0);
  data0 /= 100.00;
  Serial.println(data0);
  Serial.println(calibrationKnownMass);
  calibrationValue = 1;
  if(data0 != 0){
    calibrationValue = calibrationKnownMass / (data0 * scaleVar);
  }
  Serial.println(calibrationValue);
  #endif
  EEPROM.put(eepromAdress(0), calibrationValue);
  EEPROM.commit();
  }
  endState = false;
}
double readBatteryVoltage(void){
  long dataIn = analogRead(Battery);
  double dataOut = dataIn / 23.4;
  return dataOut;
}
void initSensors(void){
  _5Volt = _5VoltIn / 1;
  _5Volt = 5.13;//temporary
  _5mVolt = _5Volt * 1000;
  double SmaxV = scaleSensorMaxOutput * _5Volt * scaleAmpGane;
  double SmaxW = (4096 * scaleSencorMax) / SmaxV;
  scaleVar = (SmaxW * 1000) / 65535;
```

```
  double PmaxV = (pressureSensorMaxOutput * pressureAmpA) + pressureAmpB;
  double PmaxPa = (4.096 * pressureSensorMax) / PmaxV;
  PressureVar = (PmaxPa * 1000000) / 65535;
  temperatureAmpB *= 1000;
}

int buttonPressedFlag = 0;
unsigned long buttonsTimerCounter;

const byte Rows = 4;
const byte Colums = 4;

byte RowPins[Rows] = {KeypadRow1,KeypadRow2,KeypadRow3,KeypadRow4};
byte ColumPins[Colums] = {KeypadCol1,KeypadCol2,KeypadCol3,KeypadCol4};

const char keypadButtons[Rows][Colums] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
const char shiftButton = '*';

const char keypadShiftButtons[Rows][Colums] = {
  {'E','F','G','H'},
  {'I','J','K','L'},
  {'M','N','O','P'},
  {'Q','R','S','T'}
};

char kData;
int kNumber[5];
double kNumberTemp = 0;
```

```
int decimalDigitsFlag = 0;
int decimalDigitsPosition = 0;
int menuFlag = 0;
int typingFlag = 0;

void read_buttons(void){
  while(digitalRead(EmergencyButton) == 1){
    digitalWrite(Valve, LOW);
    dispNumber[0] = 25;
    dispNumber[1] = 26;
    dispNumber[2] = 25;
    dispNumber[3] = 25;
    dispNumber[4] = 24;
    ledOutput[7] = 1;
    ledOutput[6] = 1;
    ledOutput[5] = 1;
    ledOutput[4] = 1;
    ledOutput[3] = 1;
    ledOutput[2] = 1;
    ledOutput[1] = 1;
    ledOutput[0] = 1;
    send_register();
    delay(10);
  }
  read_keypad();
  if(buttonPressedFlag == 0){
    if(digitalRead(PowerButton) == 0){
      buttonPressedFlag = 1;
      buttonsTimerCounter = millis();
    }else if(digitalRead(TareButton) == 0){
      buttonPressedFlag = 2;
      buttonsTimerCounter = millis();
    }else if(digitalRead(ManualFillingButton) == 0){
```

```
    buttonPressedFlag = 3;
    buttonsTimerCounter = millis();
   }else if(digitalRead(StopFillingButton) == 0){
    buttonPressedFlag = 4;
    buttonsTimerCounter = millis();
   }else if(digitalRead(StartFillingButton) == 0){
    buttonPressedFlag = 5;
    buttonsTimerCounter = millis();
   }
  }else if(buttonPressedFlag >= 2 && millis() - buttonsTimerCounter >= 100){
   if(buttonPressedFlag == 2 && digitalRead(TareButton) == 0){
    myTare();
    buttonPressedFlag = 0;
   }else if(buttonPressedFlag == 3 && digitalRead(ManualFillingButton) == 0){
    digitalWrite(Valve, HIGH);
   }else if(buttonPressedFlag == 3 && digitalRead(ManualFillingButton) == 1){
    digitalWrite(Valve, LOW);
    buttonPressedFlag = 0;
   }else if(buttonPressedFlag == 4 && digitalRead(StopFillingButton) == 0){
    fillingStateFlag = 0;
    digitalWrite(Valve, LOW);
    valveFlag = 1;
    buttonPressedFlag = 0;
   }else if(buttonPressedFlag == 5 && digitalRead(StartFillingButton)== 0){
    if(fillingProcessFlag == 0){
     fillingStateFlag = 1;
    }else{
     fillingStateFlag = 2;
    }
    buttonPressedFlag = 0;
   }else{
    buttonPressedFlag = 0;
   }
```

```
  }else if(buttonPressedFlag == 1 && millis() - buttonsTimerCounter >= 1000){
    if(digitalRead(PowerButton) == 0){
      dispNumber[4] = 29;
      dispNumber[3] = 21;
      dispNumber[2] = 0;
      dispNumber[1] = 23;
      dispNumber[0] = 23;
      send_register();
      delay(2000);
      digitalWrite(PowerEnable,LOW);
      while(digitalRead(PowerButton) == 0);
    }
    else{
      buttonPressedFlag = 0;
    }
  }
}

void read_keypad(void){
  kData = Keypad0.get_key();
  switch(kData){
    case 'A': //set fill level
    if(menuFlag == 0){
    menuFlag = 1;
    ledOutput[4] = 0;
    ledTimerCounter = millis();
    menuTimerCounter = millis();
    }
    break;
    case 'B': //set product
    if(menuFlag == 0){
    menuFlag = 2;
    ledOutput[3] = 0;
```

```
  ledTimerCounter = millis();
  menuTimerCounter = millis();
  }
  break;
  case 'C': //backspace
  if(decimalDigitsFlag == 1 && decimalDigitsPosition == 0){
   decimalDigitsFlag = 0;
  }else{
   if(decimalDigitsPosition != 0){
    decimalDigitsPosition--;
   }
   for(i = 0; i <= 4; i++){
    kNumber[i] = kNumber[i + 1];
   }
   kNumber[4] = 0;
   kNumberTemp = kNumber[4] * 10000 + kNumber[3] * 1000 + kNumber[2] * 100 + kNumber[1] *
10 + kNumber[0];
  }
  break;
  case 'D': //enter
  if(decimalDigitsFlag == 1){
   kNumberTemp /= (int)pow(10,decimalDigitsPosition);
  }
  switch(menuFlag){
   case 1:
   switch(unitsFlag){
    case 0:
    fillVolumeTarget = kNumberTemp;
    break;
    case 1:
    fillVolumeTarget = kNumberTemp * 1000.0;
    break;
    case 2:
```

```
  fillVolumeTarget = kNumberTemp * 28.35;
  break;
  case 3:
  fillVolumeTarget = kNumberTemp * 453.53;
  break;
}
fillingProcessFlag = 0;
break;
case 2:
productSelected = kNumberTemp;
fillingProcessFlag = 0;
break;
case 10:
switch(cUnitsFlag){
  case 0:
  calibrationKnownMass = kNumberTemp;
  break;
  case 1:
  calibrationKnownMass = kNumberTemp * 1000.0;
  break;
  case 2:
  calibrationKnownMass = kNumberTemp * 28.35;
  break;
  case 3:
  calibrationKnownMass = kNumberTemp * 453.53;
  break;
}
holdState = false;
break;
case 11:
pipeLength = kNumberTemp / 2.54;
EEPROM.put(eepromAdress(3), pipeLength);
EEPROM.commit();
```

```
  break;
  case 12:
  pipeDiameter = kNumberTemp;
  EEPROM.put(eepromAdress(4), pipeDiameter);
  EEPROM.commit();
  break;
}
kNumberTemp = 0;
kNumber[0] = 0;
kNumber[1] = 0;
kNumber[2] = 0;
kNumber[3] = 0;
kNumber[4] = 0;
decimalDigitsPosition = 0;
decimalDigitsFlag = 0;
typingFlag = 0;
menuFlag = 0;
break;
case 'E': //set brightness -
if(menuFlag == 0 || menuFlag == 3){
menuFlag = 3;
menuTimerCounter = millis();
if(backlightValue  != 2){
   backlightValue -= 2;
   EEPROM.put(eepromAdress(5), backlightValue);
   EEPROM.commit();
}
}
break;
case 'F': //set brightness +
 if(menuFlag == 0 || menuFlag == 3){
menuFlag = 3;
menuTimerCounter = millis();
```

```
if(backlightValue  != 10){
  backlightValue += 2;
  EEPROM.put(eepromAdress(5), backlightValue);
  EEPROM.commit();
}
}
break;
case 'G': //start calibration
if(menuFlag == 0){
menuFlag = 10;
calibration();
}
break;
case 'H': //clear fill level value
if(menuFlag == 0){
fillVolumeTarget = 0;
kNumberTemp = 0;
for(i = 0; i <= 4; i++){
  kNumber[i] = 0;
}
decimalDigitsFlag = 0;
decimalDigitsPosition = 0;
typingFlag = 0;
fillingProcessFlag = 0;
}
break;
case 'I': //set pipe length
if(menuFlag == 0){
menuFlag = 11;
menuTimerCounter = millis();
}
break;
case 'J': //set pipe diameter
```

```
if(menuFlag == 0){
menuFlag = 12;
menuTimerCounter = millis();
}
break;
case 'K': //display flow value
if(menuFlag == 0){
menuFlag = 4;
menuTimerCounter = millis();
}
break;
case 'L': //clear product value
if(menuFlag == 0){
productSelected = 0;
kNumberTemp = 0;
for(i = 0; i <= 4; i++){
  kNumber[i] = 0;
}
decimalDigitsFlag = 0;
decimalDigitsPosition = 0;
typingFlag = 0;
fillingProcessFlag = 0;
}
break;
case 'M': //display temperature value
if(menuFlag == 0 || menuFlag == 5){
if(menuFlag == 5){
  tUnitFlag = !tUnitFlag;
  EEPROM.put(eepromAdress(2), tUnitFlag);
  EEPROM.commit();
}
menuFlag = 5;
menuTimerCounter = millis();
```

```
}
break;
case 'N': //display pressure value
if(menuFlag == 0){
menuFlag = 6;
menuTimerCounter = millis();
}
break;
case 'O': //display battery voltage
if(menuFlag == 0){
menuFlag = 7;
menuTimerCounter = millis();
}
break;
case 'P': //clear all
if(menuFlag == 0){
fillVolumeTarget = 0;
productSelected = 0;
kNumberTemp = 0;
for(i = 0; i <= 4; i++){
  kNumber[i] = 0;
}
decimalDigitsFlag = 0;
decimalDigitsPosition = 0;
typingFlag = 0;
fillingProcessFlag = 0;
}
break;
case 'R': //display firmware
if(menuFlag == 0){
menuFlag = 8;
menuTimerCounter = millis();
}
```

```
break;
case 'S': //set g/kg/oz/pount
if(menuFlag == 10){
if(++cUnitsFlag == 4){
  cUnitsFlag = 0;
}
}else{
if(++unitsFlag == 4){
  unitsFlag = 0;
}
}
EEPROM.put(eepromAdress(1), unitsFlag);
EEPROM.commit();
break;
case 'T': //exit menu without saving
kNumberTemp = 0;
for(i = 0; i <= 4; i++){
  kNumber[i] = 0;
}
decimalDigitsFlag = 0;
decimalDigitsPosition = 0;
typingFlag = 0;
menuFlag = 0;
endState = true;
holdState = false;
break;
case '#': //dot
if(decimalDigitsFlag == 0){
  decimalDigitsFlag = 1;
}
break;
default: //numbers
if(menuFlag == 1 || menuFlag == 2 || menuFlag >= 10){
```

```
    if(kData >= '0' && kData <= '9'){
      if(typingFlag == 0){
        kNumber[0] = 0;
        kNumber[1] = 0;
        kNumber[2] = 0;
        kNumber[3] = 0;
        kNumber[4] = 0;
        decimalDigitsFlag = 0;
        decimalDigitsPosition = 0;
      }
      typingFlag = 1;
      for(i = 4; i >= 0; i--){
        kNumber[i] = kNumber[i-1];
      }
      kNumber[0] = kData - 48;
      if(decimalDigitsFlag == 1){
        decimalDigitsPosition++;
      }
      kNumberTemp = kNumber[4] * 10000 + kNumber[3] * 1000 + kNumber[2] * 100 + kNumber[1]
* 10 + kNumber[0];
    }
   }
  }
}


bool led[8] = {0,0,0,0,0,0,0,0};
const byte sevenSegments[31] = {
  0b00000011,//[0] = 0,O
  0b10011111,//[1] = 1
  0b00100101,//[2] = 2
  0b00001101,//[3] = 3
  0b10011001,//[4] = 4
  0b01001001,//[5] = 5
```

```
  0b01000001,//[6] = 6
  0b00011111,//[7] = 7
  0b00000001,//[8] = 8
  0b00001001,//[9] = 9
  0b00000010,//[10] = 0.
  0b10011110,//[11] = 1.
  0b00100100,//[12] = 2.
  0b00001100,//[13] = 3.
  0b10011000,//[14] = 4.
  0b01001000,//[15] = 5.
  0b01000000,//[16] = 6.
  0b00011110,//[17] = 7.
  0b00000000,//[18] = 8.
  0b00001000,//[19] = 9.
  0b11111101,//[20] = -
  0b11111111,//[21] = {off}
  0b01100011,//[22] = C
  0b01110001,//[23] = F
  0b01100001,//[24] = E
  0b11110101,//[25] = r
  0b11000101,//[26] = o
  0b00010001,//[27] = A
  0b11100010,//[28] = L.
  0b00110000,//[29] = P.
  0b11111110//[30] = .
};
byte registerOut[6];
unsigned long menuTimerCounter = 0;
unsigned long ledTimerCounter = 0;
unsigned long dispTimerCounter = 0;
int unitsFlag = 1;
bool tUnitFlag = 0;
int cUnitsFlag = 0;
```

```
int backlightValue = 2;
long dispNumberTemp;
int dispNumber[6] = {0,0,0,0,0,21};
int dispSymbolFlag = 0;
int dispDotPositionFlag = 0;
bool minusFlag;
bool ledOutput[8];
int ledOutputtemp;

void display_data(void){
  if(millis() - dispTimerCounter >= 100){
    set_digits();
    set_leds();
    send_register();
    dispTimerCounter = millis();
  }
}
void set_digits(void){
  if(typingFlag == 0){
    if(menuFlag != 0 && menuFlag != 10 && millis() - menuTimerCounter >= 3000){
       menuFlag = 0;
    }
    switch(menuFlag){
      case 0: //weight value
      switch(unitsFlag){
        case 0:
        dispNumberTemp = scaleValue;
        dispDotPositionFlag = 10;
        break;
        case 1:
        dispNumberTemp = scaleValue / 10;
        dispDotPositionFlag = 2;
        break;
```

```
  case 2:
  dispNumberTemp = scaleValue / 2.835;
  dispDotPositionFlag = 1;
  break;
  case 3:
  dispNumberTemp = scaleValue / 4.5359;
  dispDotPositionFlag = 2;
  break;
}
dispSymbolFlag = 0;
break;
case 1: //fill volume value
switch(unitsFlag){
  case 0:
  dispNumberTemp = fillVolumeTarget;
  dispDotPositionFlag = 10;
  break;
  case 1:
  dispNumberTemp = fillVolumeTarget / 10;
  dispDotPositionFlag = 2;
  break;
  case 2:
  dispNumberTemp = fillVolumeTarget / 2.835;
  dispDotPositionFlag = 1;
  break;
  case 3:
  dispNumberTemp = fillVolumeTarget / 4.5359;
  dispDotPositionFlag = 2;
  break;
}
dispSymbolFlag = 0;
break;
case 2: //product
```

```
dispNumberTemp = productSelected;
dispDotPositionFlag = 10;
dispSymbolFlag = 0;
break;
case 3: //brightness value
dispNumberTemp = backlightValue * 10;
dispDotPositionFlag = 10;
dispSymbolFlag = 0;
break;
case 4: //flow value
switch(unitsFlag){
  case 0:
  dispNumberTemp = flowValue;
  dispDotPositionFlag = 10;
  break;
  case 1:
  dispNumberTemp = flowValue / 10;
  dispDotPositionFlag = 2;
  break;
  case 2:
  dispNumberTemp = flowValue / 2.835;
  dispDotPositionFlag = 2;
  break;
  case 3:
  dispNumberTemp = flowValue / 4.5359;
  dispDotPositionFlag = 2;
  break;
}
dispSymbolFlag = 0;
break;
case 5: //temperature value
if(tUnitFlag == 0){
  dispNumberTemp = temperatureValue0 * 100;
```

```
    dispDotPositionFlag = 2;
    dispSymbolFlag = 2;
   }else if(tUnitFlag == 1){
    dispNumberTemp = ((temperatureValue0 * 1.8) + 32) * 100;
    dispDotPositionFlag = 10;
    dispSymbolFlag = 2;
   }
   break;
   case 6: //presure value
   dispNumberTemp = pressureValue/10;
   dispDotPositionFlag = 2;
   dispSymbolFlag = 0;
   break;
   case 7: //battery voltage
   dispNumberTemp = batteryValue;
   dispDotPositionFlag = 1;
   dispSymbolFlag = 0;
   break;
   case 8: //firmware
   dispNumberTemp = Firmware;
   dispDotPositionFlag = 10;
   dispSymbolFlag = 3;
   break;
   default:
   dispNumberTemp = 0;
   dispDotPositionFlag = 10;
 }
 if(dispNumberTemp < 0){
   minusFlag = 1;
   dispNumberTemp = abs(dispNumberTemp);
 }else{
   minusFlag = 0;
 }
```

```
for(i = 4; i >= 0; i--){
  dispNumber[i] = (dispNumberTemp / (int)pow(10,i)) % 10;
  if(dispDotPositionFlag == i){
    dispNumber[i] += 10;
  }
  if(dispNumber[i] == 0 && dispNumber[i + 1] == 21 && i != 0){
    dispNumber[i] = 21;
  }
}
switch(dispSymbolFlag){
  case 1:
  dispNumber[0] = 21;
  dispNumber[1] = 28;
  dispNumber[2] = 27;
  dispNumber[3] = 22;
  dispNumber[4] = 21;
  break;
  case 2:
  if(tUnitFlag == 0){
    dispNumber[0] = 22;
  }else if(tUnitFlag == 1){
    dispNumber[0] = 23;
  }
  break;
  case 3:
  for(i = 0; i <= 4; i++){
    if(dispNumber[i] == 21){
      dispNumber[i] = 23;
      i = 5;
    }
  }
  break;
}
```

```
    if(minusFlag == 1){
      for(i = 0; i <= 4; i++){
        if(dispNumber[i] == 21){
          dispNumber[i] = 20;
          i = 5;
        }
      }
    }
  }else{
    if(decimalDigitsFlag == 1){
      dispDotPositionFlag = decimalDigitsPosition;
    }else{
      dispDotPositionFlag =  5;
    }
    dispNumberTemp = kNumberTemp;
    for(i = 4; i >= 0; i--){
      dispNumber[i] = (dispNumberTemp / (int)pow(10,i)) % 10;
      if(dispDotPositionFlag == i){
        dispNumber[i] += 10;
      }
      if(dispNumber[i] == 0 && dispNumber[i + 1] == 21 && i != 0){
        dispNumber[i] = 21;
      }
    }
  }
}
void set_leds(void){
  if(batteryValue >= 100.5 && ledOutput[7] == 1){
    ledOutput[7] = 0;
  }else if(batteryValue <= 100 && ledOutput[7] == 0){
    ledOutput[7] = 1;
  }
  if(batteryValue >= 110.5 && ledOutput[6] == 1){
```

```
  ledOutput[6] = 0;
}else if(batteryValue <= 110 && ledOutput[6] == 0){
  ledOutput[6] = 1;
}
if(batteryValue >= 120.5 && ledOutput[5] == 1){
  ledOutput[5] = 0;
}else if(batteryValue <= 120 && ledOutput[5] == 0){
  ledOutput[5] = 1;
}
if(menuFlag == 1 && millis() - ledTimerCounter > 500){
  if(unitsFlag <= 1){
    ledOutput[1] = !ledOutput[1];
  }else if(unitsFlag >= 2){
    ledOutput[2] = !ledOutput[2];
  }
  ledOutput[4] = !ledOutput[4];
  ledTimerCounter = millis();
}
if(fillVolumeTarget != 0 && menuFlag != 1){
  ledOutput[4] = 0;
}else if(menuFlag != 1){
  ledOutput[4] = 1;
}
if(menuFlag == 2 && millis() - ledTimerCounter > 500){
  ledOutput[3] = !ledOutput[3];
  ledTimerCounter = millis();
}
if(productSelected != 0 && menuFlag != 2){
  ledOutput[3] = 0;
}else if(menuFlag != 2){
  ledOutput[3] = 1;
}
if(menuFlag == 3 && millis() - ledTimerCounter > 1000){
```

```
  if(unitsFlag <= 1){
    ledOutput[1] = !ledOutput[6];
  }else if(unitsFlag >= 2){
    ledOutput[2] = !ledOutput[5];
  }
  ledTimerCounter = millis();
}
if(menuFlag == 0){
  switch(unitsFlag){
    case 0:
    ledOutput[2] = 0;
    ledOutput[1] = 1;
    ledOutput[0] = 0;
    break;
    case 1:
    ledOutput[2] = 1;
    ledOutput[1] = 0;
    ledOutput[0] = 1;
    break;
    case 2:
    ledOutput[2] = 1;
    ledOutput[1] = 0;
    ledOutput[0] = 0;
    break;
    case 3:
    ledOutput[2] = 0;
    ledOutput[1] = 1;
    ledOutput[0] = 1;
    break;
  }
}else if(menuFlag == 1){
  switch(unitsFlag){
    case 0:
```

```
    ledOutput[2] = 0;
    ledOutput[0] = 0;
    break;
    case 1:
    ledOutput[2] = 1;
    ledOutput[0] = 1;
    break;
    case 2:
    ledOutput[1] = 0;
    ledOutput[0] = 0;
    break;
    case 3:
    ledOutput[1] = 1;
    ledOutput[0] = 1;
    break;
  }
}else if(menuFlag == 10){
  switch(cUnitsFlag){
    case 0:
    ledOutput[2] = 0;
    ledOutput[1] = 1;
    ledOutput[0] = 0;
    break;
    case 1:
    ledOutput[2] = 1;
    ledOutput[1] = 0;
    ledOutput[0] = 1;
    break;
    case 2:
    ledOutput[2] = 1;
    ledOutput[1] = 0;
    ledOutput[0] = 0;
    break;
```

```
      case 3:
      ledOutput[2] = 0;
      ledOutput[1] = 1;
      ledOutput[0] = 1;
      break;
    }
  }else{
    ledOutput[2] = 0;
    ledOutput[1] = 0;
    ledOutput[0] = 0;
  }
}
void send_register(void){
  for(i = 4; i >= 0; i--){
    registerOut[i] = sevenSegments[dispNumber[i]];
  }
  registerOut[5] = 128*ledOutput[7]+64*ledOutput[6]+32*ledOutput[5]+16*ledOutput[4]+8*ledOutput[3]+4*ledOutput[2]+2*ledOutput[1]+1*ledOutput[0];
  digitalWrite(Display_RCK, LOW);
  for(i = 5; i >= 0; i--){
    shiftOut(Display_SER, Display_SCK, LSBFIRST, registerOut[i]);
  }
  digitalWrite(Display_RCK, HIGH);
  analogWrite(Display_G, 255 - (backlightValue * 25.5));
}

long fillVolumeTarget = 0;
long productSelected = 0;
double flowValue = 0;
double flowValueMax = 0;
double flowValueAverage = 0;
int fillingStateFlag = 0;
int fillingProcessFlag = 0;
```

```
float pipeDiameter = 0;//cm
float pipeLength = 0;//cm
float viscosity = 0;//mpas
float indicativeFlow = 200.00;
unsigned long fillingStartTime = 0;
unsigned long fillingStopTime = 0;
unsigned long fillingCalcTime = 0;
unsigned long fillingCalcTimeTrue = 0;
unsigned long fillingCalcTimeOld = 0;
double fillingCalcTimeF = 0;
unsigned long delayFlowTime = 0;
bool valveFlag = 0;
long fillVolume = 0;
double flow = 0;
float temperatureValueOld;
float liquidDencity = 1;
float calculateState = 0;
unsigned long buzzerTimer = 0;
float R;
unsigned long fillingMidStartTime = 0;
unsigned long fillingMidStopTime = 0;
unsigned long fillingMidTime = 0;
unsigned long fillingMidCalcTime = 0;
unsigned long time0 = 0;

void fillingProcess(void){
  if(valveFlag == 1){
    if(scaleValue >= fillVolume + 0.5){
      valveFlag = 0;
      digitalWrite(Valve,LOW);
      fillingStopTime = millis();
      fillingStateFlag = 8;
      Serial.println("stoped by max fill value");
```

```
    Serial.print("filling stop TimePassed = ");
    Serial.print(millis() - fillingStartTime);
    Serial.print("  scaleValue = ");
    Serial.println(scaleValue);


   }
  }
  switch(fillingStateFlag){
   case 1:
   if(fillVolumeTarget == 0){
    fillingStateFlag = 0;
    break;
   }
   fillVolume = fillVolumeTarget;
   flowValue = indicativeFlow;
   #ifdef _EEPROM
   if(productSelected != 0){
    EEPROM.get(eepromAdress(productSelected + 10), flowValue);
    if(flowValue == 0){
     flowValue = indicativeFlow;
    }
   }
   #endif
   #ifdef _WIFI
   if(productSelected != 0){
    String serverPath =
"http://192.168.1.250:2312/addvaluefromget/?guid1=device1&field1="+String(productSelected,
2);//send product selected
    int httpResponseCode = http.GET();
    if (httpResponseCode>0) {
     String payload1 = http.getString();
     liquidDencity = payload1.toFloat();//take from server
    }
   }
```

```
#endif
fillingStateFlag = 2;
Serial.println("first start");
break;
case 2:
#ifdef _WIFI
if(abs(temperatureValue - temperatureValueOld) >= 5 || fillingProcessFlag == 0){
 String serverPath = "http://192.168.1.250:2312/addvaluefromget/?guid1=device1&field1="+String(productSelected, 2)+"&field2="+String(temperatureValue, 2);//send product,temperature value
  int httpResponseCode = http.GET();
  if (httpResponseCode>0) {
   String payload2 = http.getString();
   viscosity = payload2.toFloat();//take fron server
  }
}
flow = (((3.141 * pow(((pipeDiameter*0.032)/2),4)) * (pressureValue*0.02088) / (8 * (viscosity*0.000672) * (pipeLength*0.032))))* 28.31 * 1000;
flow *= liquidDencity;
if(fillingProcessFlag == 0){
  flowValue = flow;
}else{
  flowValue = abs(flowValue - flow) / 2;
}
#endif
fillingCalcTimeF = (fillVolume * 1000.00) / flowValue;
fillingCalcTime = fillingCalcTimeF;
fillingStateFlag = 3;
Serial.print("pressure =");
Serial.print(pressureValue);
Serial.print("  temperature = ");
Serial.print(temperatureValue0);
Serial.print("   viscosity = ");
Serial.println(viscosity);
```

```
break;
case 3:
digitalWrite(Valve,HIGH);
fillingStartTime = millis();
time0 = millis();
fillingCalcTimeF = fillingCalcTime * 0.50;
fillingMidCalcTime = fillingCalcTimeF;
fillingStateFlag = 4;
Serial.println("open ok");
break;
case 4:
if(millis() - time0 >= 150){
  if(scaleValue >= 100){
    valveFlag = 1;
    fillingStateFlag = 5;
    Serial.print("first drop scaleValue = ");
    Serial.print(scaleValue);
    Serial.print("   delayFlowTime = ");
    Serial.println(millis() - fillingStartTime);
  }else{
    time0 = millis();
  }
}
break;
case 5:
if(millis() - fillingStartTime >= fillingMidCalcTime){
  digitalWrite(Valve,LOW);
  valveFlag = 0;
  fillingMidStopTime = millis();
  fillingStateFlag = 6;
}
break;
case 6:
```

```
  if(millis() - fillingMidStopTime >= 2000){
    fillingCalcTimeOld = fillingCalcTime;
    fillingCalcTime = ((fillingMidStopTime - fillingStartTime) * scaleValue) / fillVolume;
    fillingMidStartTime = millis();
    fillingMidTime = fillingMidStartTime - fillingMidStopTime;
    digitalWrite(Valve,HIGH);
    valveFlag = 1;
    fillingStateFlag = 7;
  }
break;
case 7:
if(millis() - fillingStartTime - fillingMidTime >= fillingCalcTime){
  valveFlag = 0;
  digitalWrite(Valve,LOW);
  fillingStopTime = millis();
  fillingStateFlag = 8;
  Serial.print("filling stop TimePassed = ");
  Serial.print(millis() - fillingStartTime);
  Serial.print("  scaleValue = ");
  Serial.println(scaleValue);
}
break;
case 8:
if(millis() - fillingStopTime >= 2000){
  fillingCalcTimeTrue = fillingStopTime - fillingStartTime;
  flowValue = (scaleValue * 1000.00) / fillingCalcTimeTrue;
  if(scaleValue > fillVolumeTarget){
    fillVolume += (scaleValue - fillVolumeTarget);
  }else if(scaleValue > fillVolumeTarget){
    fillVolume -= (scaleValue - fillVolumeTarget);
  }
  fillingProcessFlag++;
  #ifdef _EEPROM
```

```
    if(fillingProcessFlag == 3 && productSelected != 0){
     EEPROM.get(eepromAdress(productSelected + 10), flow);
     if(abs(flow - flowValue) >= 100){
       EEPROM.put(eepromAdress(productSelected + 10), flowValue);
       EEPROM.commit();
     }
    }
    #endif
    buzzerTimer = millis();
    digitalWrite(Buzzer,HIGH);
    fillingStateFlag = 9;
    Serial.print("final calculation flow = ");
    Serial.println(flowValue);
    Serial.print(" scaleValue = ");
    Serial.println (scaleValue);
   }
   break;
   case 9:
   if(millis() - buzzerTimer >= 800){
    digitalWrite(Buzzer,LOW);
    fillingStateFlag = 0;
    Serial.println("next");
   }
  }
}

/*

*/
const int Firmware = 1;

//#define Sensor_HX711
#define Sensor_ADS1115
```

```
//#define _WIFI
#define _EEPROM
double scaleSencorMax = 200; // kg
double scaleSensorMaxOutput = 2; // mV/V
double scaleAmpGane = 918;

double pressureSensorMax = 1.6; // Mpa
double pressureSensorMaxOutput = 4.5; //V
double pressureAmpA = 1.25;
double pressureAmpB = -0.625;

double temperatureAmpA = 1.25;
double temperatureAmpB = -0.625;
double temperatureRsens = 2200; //Ohm
long temperatureSensorTable[31] = {
  32650,
  25395,
  19903,
  15714,
  12493,
  10000,
  8056,
  6530,
  5324,
  4366,
  3601,
  2985,
  2487,
  2082,
  1751,
  1480,
  1256,
  1070,
```

```
  916,
  786,
  678,
  587,
  510,
  444,
  388,
  340,
  299,
  264,
  234,
  207,
  184
};
```