ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

" Μελέτη, σχεδιασμός και υλοποίηση αυτόματου κομποστοποιητή (Study, design and implementation of an automatic composter)"

ΚΑΡΑΝΤΩΝΙΔΗΣ ΕΥΑΓΓΕΛΟΣ

ΕΠΙΒΛΕΠΩΝ: Λάμπρος Μπισδούνης, Καθηγητής

ΠΑΤΡΑ 2023

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, 22/9/2023

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Λάμπρος Μπισδούνης, Καθηγητής
2. Γεώργιος Σουλιώτης, Επίκουρος Καθηγητής
3. Ανδρέας Κατσαΐτης, Τεχνικός Εργαστηρίων

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσης πτυχιακής εργασίας είναι ο σχεδιασμός και υλοποίηση μιας συσκευής παρασκευής κομπόστ ή οποία θα μπορεί να λειτουργεί αυτόνομα θα παρέχει όλες τις απαραίτητες μετρήσεις για την κατάσταση του υλικού καθώς και θα δημιουργεί τις κατάλληλες συνθήκες όπως η απαραίτητη υγρασία και θερμοκρασία ώστε να επιταχύνει την διαδικασία και να υπάρχει το επιθυμητό αποτέλεσμα.

Για να επιτευχθεί το αποτέλεσμα αυτό θα γίνει χρήση πλήθους διαφορετικών αισθητήρων που ο καθένας από αυτούς θα εκτελεί διαφορετικό σκοπό, θα κατανεμηθούν σε δυο Arduino pro mini για να λαμβάνονται μετρήσεις και να φιλτράρονται, επίσης θα υπάρχουν έξοδοι όπως κινητήρας αντιστάσεις και ότι άλλο χρειαστεί. Με τη σειρά τους τα Arduino θα συνδεθούν σε ένα ESP 32 το οποίο θα είναι υπεύθυνο για τον συντονισμό του συστήματος καθώς και για την διεπαφή του χρήστη με τη συσκευή η οποία θα είναι μέσω Wi-Fi ώστε να μπορεί να γίνει έλεγχος από υπολογιστή ή από κινητό τηλέφωνο εύκολα.

Μια ακόμα λειτουργεία που θα εκτελεί το EPS32 θα είναι να στέλνει τα δεδομένα των μετρήσεων σε ένα υπολογιστικό φύλλο Excel για να μπορούν να αξιολογηθούν τα αποτελέσματα.

Στόχος της εργασίας είναι να κατασκευαστεί μια λειτουργική πρωτότυπη συσκευή η οποία θα έχει περιθώρια ανάπτυξης όσο σε υλικό όσο και σε λογισμικό, επίσης θα γίνει προσπάθεια να γίνει όσο το δυνατόν μεγαλύτερη εξοικονόμηση ενέργειας.

Περιεχόμενα

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

1.1 Το κομπόστ

Το κομπόστ είναι το φυσικό λίπασμα που παράγεται από την αποσύνθεση των οργανικών υλικών όπως φύλλα, κλαδιά, φρούτα, λαχανικά, κατακάθια καφέ, γενικά υπολείμματα κουζίνας κλπ., όπου γίνεται από μικροοργανισμούς του εδάφους όπως μύκητες, βακτήρια, γαιοσκώληκες.

Είναι μια πλούσια σκούρα ουσία γνωστή επίσης ως και εδαφοβελτιωτικό. Το κομπόστ μπορεί να έχει πολύ καλά ποιοτικά χαρακτηριστικά και μπορεί να χρησιμοποιηθεί για κάθε είδους καλλιέργεια.

Εικόνα 1: Κομπόστ

1.2 Χρήσεις του κομπόστ

Χρησιμοποιείται στην καλλιέργεια για βελτίωση του εδάφους σε θρεπτικά συστατικά για τα φυτά έτσι ώστε να μπορούν να αναπτύξουν ριζικό σύστημα. Επίσης βοηθάει στην συγκράτηση υγρών στο έδαφος και στον εξαερισμό του το οποίο είναι σημαντικό για τα φυτά ειδικά σε αμμώδες έδαφος.

1.3 Η διαδικασία της κομποστοποίησης

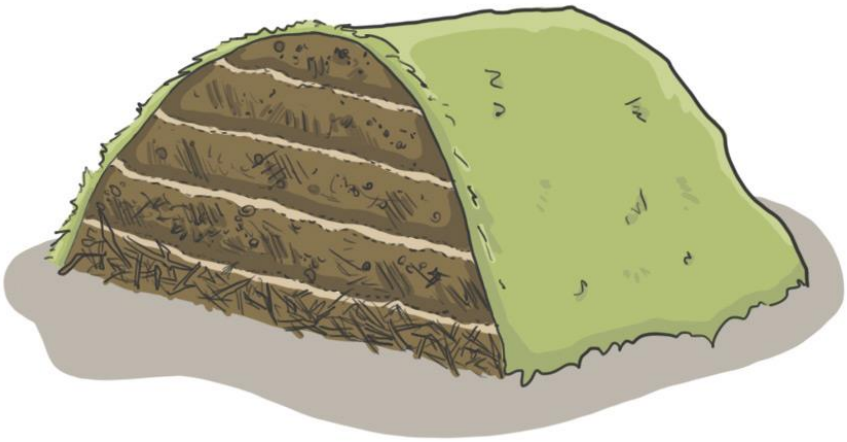
Κομποστοποίηση είναι η διαδικασία κατά την οποία γίνεται αερόβια βιολογική αποικοδόμηση και σταθεροποίηση των οργανικών υλικών, που πραγματοποιείται υπό τις φυσικές και χημικές συνθήκες που ευνοούν τη ανάπτυξη συγκεκριμένων θερμόφιλων, θερμοάντοχων και μεσόφιλων μικροβιακών πληθυσμών και η οποία μετατρέπει τα οργανικά υλικά σε μια πλούσια σκούρα ύλη δηλαδή το κομπόστ. Η ποιότητά του εξαρτάται κυρίως από το είδος και την ποιότητα των υλικών που προστίθενται στη φάση της κομποστοποίησης, αλλά και άλλους παράγοντες όπως πχ. ο τρόπος στοίβαξης, ο χώρος παρασκευής, η ταχύτητα αποδόμησης των υλικών και η πορεία της ζύμωσης.

Η κομποστοποίηση γίνεται και πολλούς τρόπους, οι πιο συνηθισμένοι είναι δύο, είτε με την μέθοδο του κομποστοσωρού, είτε με χρήση κομποστοποιητή.

Κατά την διάρκεια της κομποστοποίησης υπάρχει πιθανότητα να υπάρξει παραγωγή μεθανίου λόγω χώνεψης απουσία αέρα, το οποίο προκύπτει στην περίπτωση που κατά την διάρκεια της κομποστοποίησης δεν υπάρχει αρκετό οξυγόνο.

1.3.1 Η μέθοδος του κομποστοσωρού

Η μέθοδος του επιφανειακού σωρού ή κομποστοσωρού είναι η πιο απλή. Τα διάφορα υλικά στρώνονται σε μια επιφάνεια πάνω σε σκαλισμένο ή οργωμένο χώμα.
Ο σωρός πρέπει να παίρνει το σχήμα ενός λόφου οι διαστάσεις εξαρτώνται από το πόσο υλικό υπάρχει, τον χώρο ο οποίος υπάρχει, καθώς και τον τρόπο ανάδευσης.
Στη βάση τοποθετούνται χονδροειδή υλικά, π.χ. διάφορα κλαδιά, προκειμένου να κυκλοφορεί ο αέρας και από πάνω προσθέτουμε τα υλικά. Γενικά προτιμάμε υλικά μικρά σε μέγεθος ιδανικά θρυμματισμένα για καλύτερη απορρόφηση υγρών και διατήρηση υγρασίας.

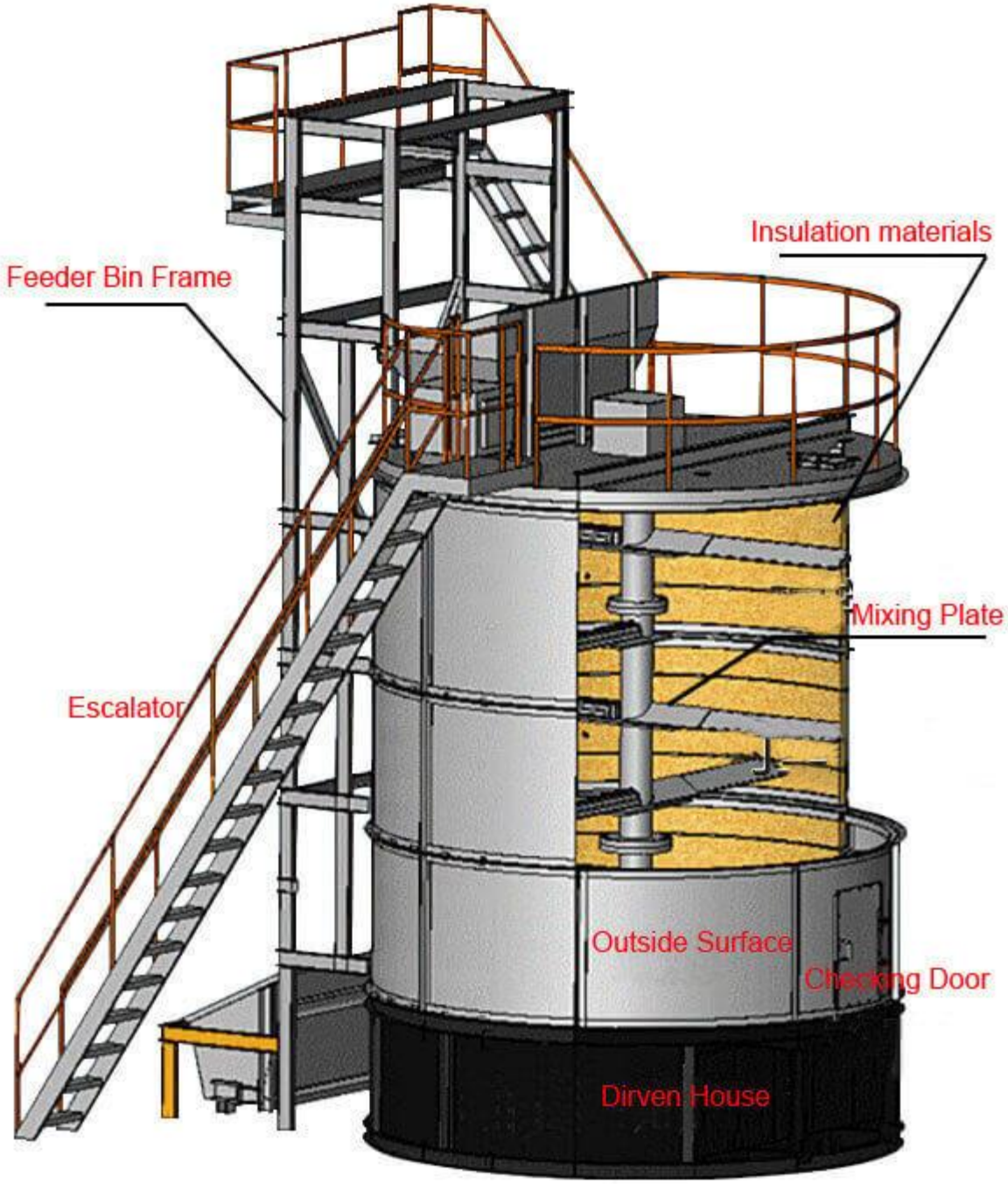


Εικόνα 2: Κομποστοσωρός

1.3.1 Η μέθοδος του κομποστοποιητή

Οι κομποστοποιητές ποικίλουν σε σχέδια και μεγέθη, ανάλογα με την ζητούμενη παραγωγή κομπόστ και τις διαστάσεις του χώρου εγκατάστασης.
Μπορεί να είναι ένας απλός πλαστικός κάδος κομποστοποίησης ή και αυτοσχέδιος από παλέτες, για οικιακή χρήση. Αρκεί να έχει κατάλληλο σχεδιασμό έτσι ώστε να οξυγονώνονται τα υλικά.
Οι κομποστοποιητές, δεν κάνουν κάτι διαφορετικό από την φύση. Η λειτουργία τους στηρίζεται στην φυσιολογική λειτουργία αποδόμησης μέσω μικροοργανισμών, κάνουν όμως την διαδικασία της αποδόμησης αρκετά γρηγορότερα απ' ότι γίνεται έξω στην φύση. Αυτό το επιτυγχάνουν κυρίως λόγω του σχεδιασμού τους, που επιτρέπει την εισροή του αέρα σε ολόκληρη την μάζα του κομπόστ και ευνοούν την ανάπτυξη και τον πολλαπλασιασμό των μικροοργανισμών.
Επιπλέον αναπτύσσοντας υψηλές θερμοκρασίες σκοτώνουν τα παράσιτα και τα καταστρέφουν την βλαστική ικανότητα των σπόρων ζιζανίων που περιέχονται πολλές φορές στο υλικό που πρόκειται να κομποστοποιηθεί.
Με αυτά τα πλεονεκτήματα οι κομποστοποιητές είναι η πλέον ενδεδειγμένη λύση για παραγωγή καθαρού και ποιοτικού κομπόστ και σε πιο σύντομο χρονικό διάστημα.

Εικόνα 3: Απλός πλαστικός κομποστοποιητής

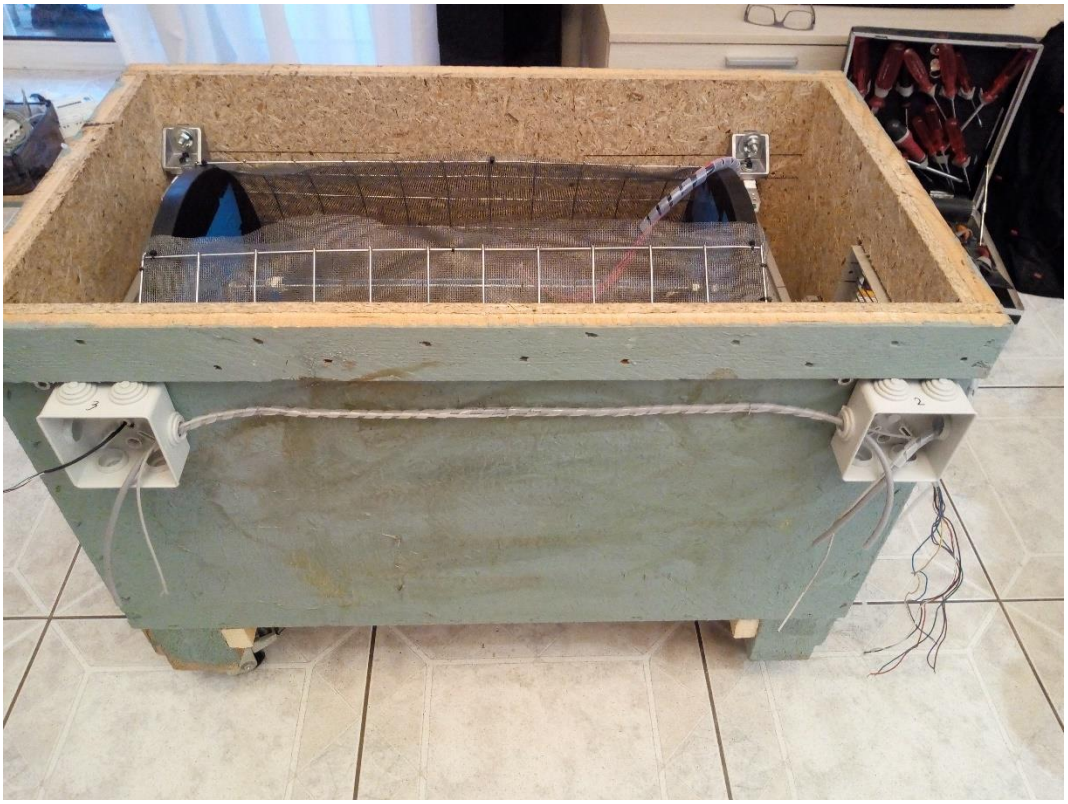


Εικόνα 4: Βιομηχανικός κομποστοποιητής

ΚΕΦΑΛΑΙΟ 2 : ΔΟΜΗ ΤΟΥ ΑΥΤΟΜΑΤΟΥ ΚΟΜΠΟΣΤΟΠΟΙΗΤΗ

2.1 ΤΟ ΕΞΩΤΕΡΙΚΌ ΚΕΛΥΦΟΣ

Η κατασκευή θα υλοποιηθεί μέσα ένα ξύλινο κουτί το οποίο βρέθηκε με διαστάσεις 80Χ46Χ54 εκατοστά (μήκος/πλάτος/ύψος)



Εικόνα 5: Το ξύλινο κουτί με τον κάδο ανάδευσης.

Οι εσωτερικές διαστάσεις του κουτιού είναι 74Χ40Χ54 εκατοστά (μήκος/πλάτος/ύψος), όποτε το σύστημα ανάδευσης θα πρέπει να διαμορφωθεί μέσα σε αυτές τις διαστάσεις, αναλόγως και το σύστημα μέτρησης βάρους.

Βάζοντας όλα τα απαραίτητα εξαρτήματα θα υπολογίσουμε και το ύψος του κάδου άρα από τα 74 εκατοστά χώρου αφαιρώ 10 εκατοστά που χρειάζονται για τον μειωτήρα του κινητήρα αφαιρώ και 3 εκατοστά από την κάθε πλευρά για στήριξη του κάδου στους ρουλεμάν καθώς και άλλα 3 εκατοστά για την στήριξη του συστήματος στο προφίλ αλουμινίου οπότε μένουν 55 εκατοστά για το ύψος του κάδου.

Η χωρητικότητά του κάδου είναι βάσει τα παραπάνω και από τον τύπο :

$$V=\pi*h*r^2$$

Με h=55 cm και r =19 cm, έχω V ίσο με 62376,32 κυβικά εκατοστά (cm^3).

Εικόνα 6: Block κινητήρα και μειωτήρα

Εικόνα 7:Ο κάδος ανάδευσης κατά την τοποθέτηση αντιστάσεων και θερμομέτρων

2.2 Ο ΚΑΔΟΣ ΑΝΑΔΕΥΣΗΣ

Ο κάδος ανάδευσης θα έχει μέσα του δύο αντιστάσεις ώστε να δημιουργηθούν οι ιδανικές θερμοκρασίες ώστε να μπορούν να αναπτυχθούν οι μικροοργανισμοί.
Για τον έλεγχο τους θα υπάρχουν αντίστοιχα δύο ψηφιακά θερμόμετρα καθώς και τον έλεγχο θερμοκρασίας του υλικού, τα θερμόμετρα και οι αντιστάσεις έχουν καλυφθεί με σύρμα χαλκού για καλύτερη επαγωγή θερμότητας.
Για τον αερισμό του θα καλυφθεί με σύρμα για μηχανική αντοχή και μετά με σίτα, με αυτόν τον τρόπο θα μπορεί να αερίζεται εύκολα το υλικό και η περίσσεια υγρασία θα μπορεί να αποβληθεί.

Εικόνα 8: Ο κάδος τελειοποιημένος

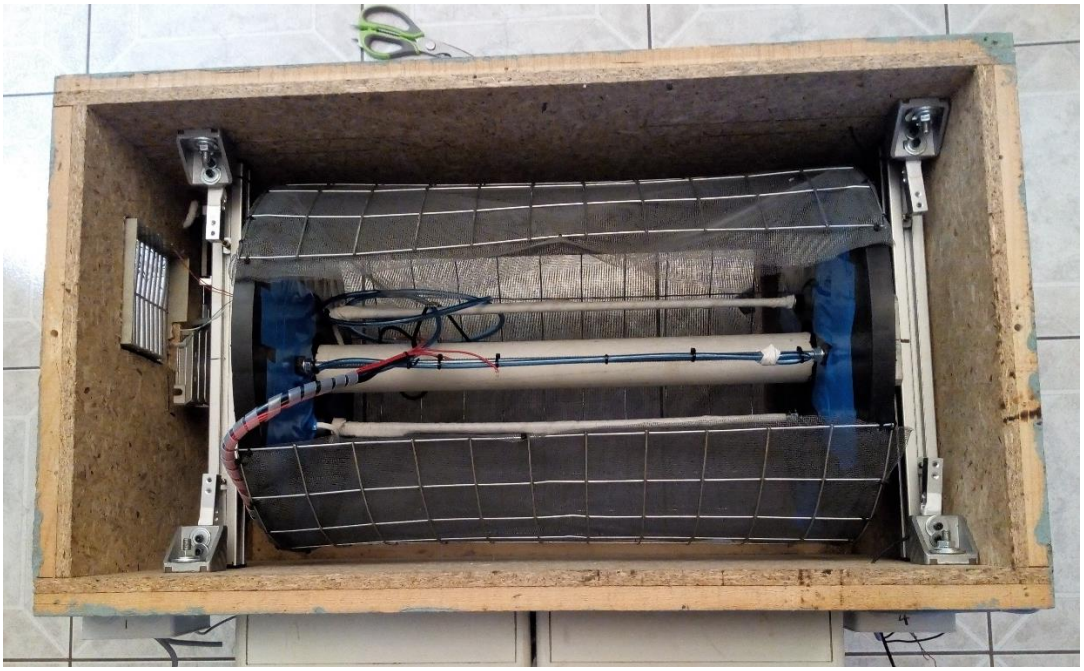
2.3 ΤΟ ΣΥΣΤΗΜΑ ΜΕΤΡΗΣΗΣ ΒΑΡΟΥΣ

Μια από τις πιο βασικές παραμέτρους της κομποστοποίησης είναι η υγρασία του υλικού η οποία πρέπει να διατηρείται σταθερή καθ' όλη τη διάρκεια λειτουργείας, όμως λόγω της φύσης των υλικών δεν γίνεται να μετρηθεί με αισθητήρες υγρασίας εδάφους, οπότε λαμβάνοντας υπ' όψη ότι το μόνο που μπορεί να μειώσει το βάρος δραματικά είναι η αποβαλλόμενη υγρασία.
Με αυτόν το τρόπο σε κάθε εκκίνηση του συστήματος μπορεί να μειωθεί το απόβαρο, έπειτα να γεμίσει ο κάδος με το υλικό και τα επιθυμητά λίτρα νερού (ανάλογα το υλικό), με αποτέλεσμα να υπάρχει παρακολούθηση βάρους αλλά και υγρασίας.

Στο σύστημα του κινητήρα και του κάδου είναι αναρτημένο σε 4 δυναμοκυψέλες των 20 κιλών η κάθε μία. Η επιλογή των κυψελών έγινε βάσει του μέσου βάρους του κομπόστ και δεδομένου ότι ο κάδος θα γεμίζει περίπου μέχρι την μέση ώστε να μπορεί να γίνει ανάδευση.

Volume of Compost:	Weight in lbs :	Weight in kg:
1 cubic yard (yd^3)	1250 lbs	1250 lbs
1 cubic foot (ft^3)	46 lbs	21 kg
1 cubic meter ($1m^3$)	956 lbs	434 kg
50 L bag	81 lbs	36 kg
40 L bag	65 lbs	29 kg
10 L bag	16 lbs	7 kg

Πίνακας 1: Πίνακας με προσεγγιστικές τιμές όγκου-βάρους κομπόστ (https://helpmecompost.com/compost/basics/how-much-does-compost-weigh/)



Εικόνα 9: Το σύστημα ζύγισης

Εικόνα 10 : Στήριξη δυναμοκυψέλης

2.4 ΔΕΞΑΜΕΝΗ ΝΕΡΟΥ

Εικόνα 11: Δεξαμενή νερού.

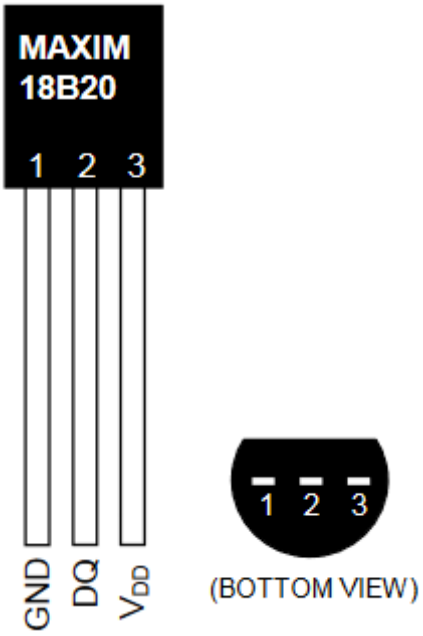
Η αποβαλλόμενη υγρασία από το υλικό θα πρέπει να αναπληρώνεται καθ' όλη τη διαδικασία για αυτό το λόγο κατασκευάστηκε μια αυτοσχέδια δεξαμενή με 5 αισθητήρες υγρασίας για να υπάρχει μερική επίβλεψη της στάθμης του νερού.
Μέσα στην δεξαμενή στην βάση της θα βρίσκεται μια αντλία νερού βυθισμένη για παροχή του νερού στην συσκευή.
Γενικά η δεξαμενή είναι φτιαγμένη έτσι ώστε να μπορεί να είναι σχετικά απομακρυσμένη από την συσκευή για να μπορεί γίνεται πλήρωση ευκολότερα και για αποφυγή υγρασίας στα ηλεκτρονικά κυκλώματα της μηχανής.

Εικόνα 12: Το εσωτερικό της δεξαμενής.

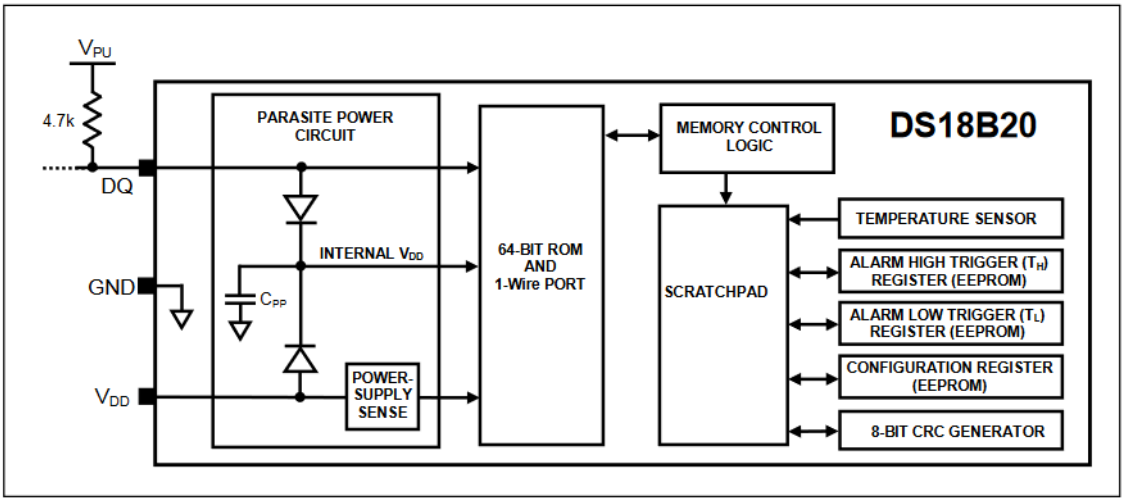
ΚΕΦΑΛΑΙΟ 3 : ΑΙΣΘΗΤΗΡΕΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΕΣ ΔΙΑΤΑΞΕΙΣ

3.1 ΑΙΣΘΗΤΗΡΑΣ ΘΕΡΜΟΚΡΑΣΙΑΣ DS18B20

Το θερμόμετρο DS18B20 είναι ένα ψηφιακό θερμόμετρο Direct–to–Digital που προσφέρει μετρήσεις σε βαθμούς κελσίου με εύρος από 9 έως 12 bit, θερμοκρασίες λειτουργείας είναι μεταξύ -55 και +125 βαθμούς κελσίου. Το εύρος μέτρησης με ακρίβεια 0,5 βαθμούς κελσίου είναι μεταξύ -10 °C και +85°C.
Οι λειτουργίες που υποστηρίζει είναι επικοινωνία 1-Wire οπού χρειάζεται μόνο μια είσοδο από τον μικροελεγκτή, έχει ξεχωριστό 64 bit σειριακό αριθμό ώστε να μπορούν να συνδεθούν στον ίδιο κόμβο πολλαπλά θερμόμετρα, εντολή αφύπνισης εάν η θερμοκρασία ξεπεράσει κάποιο εύρος που μπορεί να θέσει ο προγραμματιστής και επίσης μπορεί να τροφοδοτηθεί μέσω του διαύλου δεδομένων με παρασιτική ισχύ.
Η τάση λειτουργίας του είναι μεταξύ 3-5V και ρεύμα λειτουργίας είναι μέγιστο 1,5 mA.



Εικόνα 13: Θερμόμετρο DS18B20



Εικόνα 14: Block διάγραμμα του DS18B20

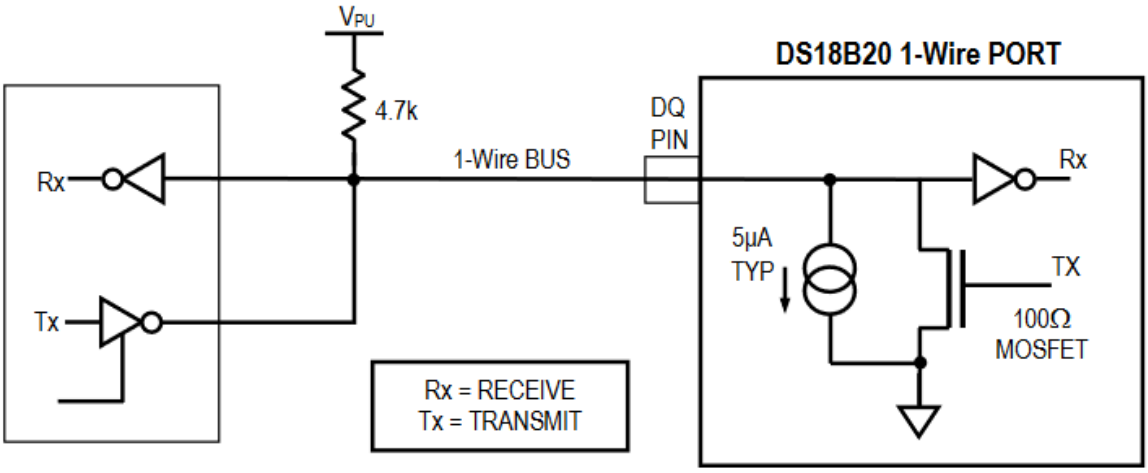
Οι αισθητήρες θερμοκρασίας Direct–to–Digital μετρούν τη θερμοκρασία μέσω μιας εξαγωγής μέτρησης θερμοκρασίας που χρησιμοποιείται ενσωματωμένη.
Κάθε αισθητήρας θερμοκρασίας μετρά τη θερμοκρασία μετρώντας τον αριθμό των κύκλων ρολογιού που διανύει ένας ταλαντωτής με χαμηλό συντελεστή θερμοκρασίας κατά τη διάρκεια μιας περιόδου πύλης που καθορίζεται από έναν ταλαντωτή με υψηλό συντελεστή θερμοκρασίας. Ο μετρητής είναι προρυθμισμένος με μια βασική καταμέτρηση που αντιστοιχεί στους -55°C. Εάν ο μετρητής φτάσει στο μηδέν πριν τελειώσει η περίοδος της πύλης, το αρχείο καταγραφής θερμοκρασίας, το οποίο είναι επίσης προρυθμισμένο στην τιμή -55°C, αυξάνεται κατά ένα, υποδεικνύοντας ότι η θερμοκρασία είναι υψηλότερη από -55°C. Παράλληλα, ο μετρητής προρυθμίζεται με μια τιμή που καθορίζεται από το κύκλωμα του συσσωρευτή κλίσης. Αυτό το κύκλωμα απαιτείται για να αντισταθμίσει την παραβολική συμπεριφορά των ταλαντωτών με την αλλαγή της θερμοκρασίας. Ο μετρητής ρολογιού ρυθμίζεται ξανά και πάλι μέχρι να φτάσει στο μηδέν. Εάν η περίοδος της πύλης δεν έχει ολοκληρωθεί ακόμα, τότε αυτή η διαδικασία επαναλαμβάνεται.

Ο συσσωρευτής κλίσης χρησιμοποιείται για να αντισταθμίσει τη μη γραμμική συμπεριφορά των ταλαντωτών με την αλλαγή της θερμοκρασίας, παρέχοντας μια μέτρηση θερμοκρασίας υψηλής ανάλυσης. Αυτό επιτυγχάνεται αλλάζοντας τον αριθμό των μετρήσεων που απαιτούνται για τον μετρητή για κάθε αυξημένο βαθμό στη θερμοκρασία. Για μια επιθυμητή ανάλυση, επομένως, πρέπει να είναι γνωστή τόσο η τιμή του μετρητή όσο και ο αριθμός των μετρήσεων ανά βαθμό C (η τιμή του συσσωρευτή κλίσης) σε μια δεδομένη θερμοκρασία.

$$TEMPERATURE = temp\_read - {}^{1}/_{2}LSB + \frac{(count\_per\_degree - count\_remain)}{count\_per\_degree}$$

Το 1-Wire σύστημα επικοινωνίας έχει έναν master ο οποίος κάνει τον έλεγχο του δίαυλου και από έναν ή περισσότερους slaves. Όταν υπάρχει μόνο ένας slave τότε το σύστημα λέγεται single-drop ενώ με περισσότερους λέγεται multi-drop.
Ο δίαυλος έχει μια μοναδική γραμμή δεδομένων. Κάθε συσκευή συνδέεται στη γραμμή δεδομένων μέσω μιας πόρτας open-drain ή 3-state. Αυτό επιτρέπει σε κάθε συσκευή να "απελευθερώσει" τη γραμμή δεδομένων όταν δεν μεταδίδει δεδομένα, έτσι ώστε ο δίαυλος να είναι διαθέσιμος για χρήση από μια άλλη συσκευή. Η θύρα 1-Wire του DS18B20 (το pin DQ) είναι open-drain με ένα εσωτερικό κύκλωμα παρόμοιο με αυτό που φαίνεται στο Σχήμα 10.

Ο δίαυλος 1-Wire απαιτεί μια εξωτερική αντίσταση pull-up περίπου 5kΩ. Επομένως, η κατάσταση ηρεμίας για τον δίαυλο είναι High. Εάν για οποιονδήποτε λόγο μια συναλλαγή πρέπει να ανασταλεί, ο δίαυλος πρέπει να αφεθεί στην κατάσταση αδράνειας για να συνεχιστεί η συναλλαγή. Ο χρόνος αδράνειας μπορεί να είναι απεριόριστος, αρκεί ο δίαυλος να βρίσκεται στην ανενεργή (high) κατάσταση κατά τη διάρκεια. Εάν ο δίαυλος κρατηθεί σε low κατάσταση για περισσότερα από 480μs, όλες οι συσκευές στον δίαυλο θα κάνουν reset.



Σχήμα 1: Το κύκλωμα επικοινωνίας 1-Wire

Εικόνα 15: Εικόνα του εξαρτήματος που έχει τοποθετηθεί στην κατασκευή. (https://www.hellasdigital.gr/electronics/sensors/temperature-sensors/waterproof-temperature-sensor-ds18b20-2m-for-arduino/)

Από την παραπάνω φωτογραφία φαίνεται ότι υπάρχει εξωτερικό μεταλλικό κάλυμμα για την προστασία του αισθητήρα από υγρασία και λεκέδες.

3.2 Μετατροπέας αναλογικού σήματος σε ψηφιακό (ADC) – HX711

Ο HX711 είναι ένας μετατροπέας αναλογικού σήματος σε ψηφιακό (ADC) 24-bit, κατάλληλος για ζυγαριές και βιομηχανικές εφαρμογές ελέγχου, προκειμένου να διασυνδεθεί απευθείας με έναν αισθητήρα γέφυρας.

Ο πολυπλέκτης εισόδου επιλέγει τη διαφορική είσοδο καναλιού Α ή Β για τον χαμηλού θορύβου ενισχυτή προγραμματιζόμενου κέρδους (PGA). Το κανάλι Α μπορεί να προγραμματιστεί με κέρδος 128 ή 64, αντίστοιχα για πλήρη διαφορική τάση εισόδου σε ±20mV ή ±40mV αντίστοιχα, όταν συνδέεται μια παροχή τροφοδοσίας 5V στον ακροδέκτη τροφοδοσίας AVDD.
Το κανάλι Β έχει σταθερό κέρδος 32.
Ο ενσωματωμένος ρυθμιστής τροφοδοσίας εξαλείφει την ανάγκη για εξωτερικό ρυθμιστή τροφοδοσίας για την παροχή αναλογικής τροφοδοσίας για τον ADC και τον αισθητήρα.
Η είσοδος ρολογιού είναι ευέλικτη. Μπορεί να προέρχεται από εξωτερική πηγή ρολογιού, κρύσταλλο ή τον ενσωματωμένο ταλαντωτή που δεν απαιτεί κανένα εξωτερικό εξάρτημα. Η ενσωματωμένη αρχή επαναφοράς τροφοδοσίας απλοποιεί την αρχικοποίηση της ψηφιακής διεπαφής.

Η διαφορική είσοδος καναλιού Α συνδέεται απευθείας με τη διαφορική έξοδο ενός αισθητήρα γέφυρας. Μπορεί να προγραμματιστεί με κέρδος 128 ή 64. Τα μεγάλα κέρδη χρειάζονται για να φιλοξενήσουν το μικρό σήμα εξόδου από τον αισθητήρα. Όταν χρησιμοποιείται μια παροχή 5V στον ακροδέκτη AVDD, αυτά τα κέρδη αντιστοιχούν σε μια πλήρη διαφορική τάση εισόδου εύρους ±20mV ή ±40mV αντίστοιχα.

Η διαφορική είσοδος καναλιού Β έχει ένα σταθερό κέρδος 32. Το πλήρες εύρος τάσης εισόδου είναι ±80mV, όταν χρησιμοποιείται μια παροχή 5V στον ακροδέκτη AVDD.

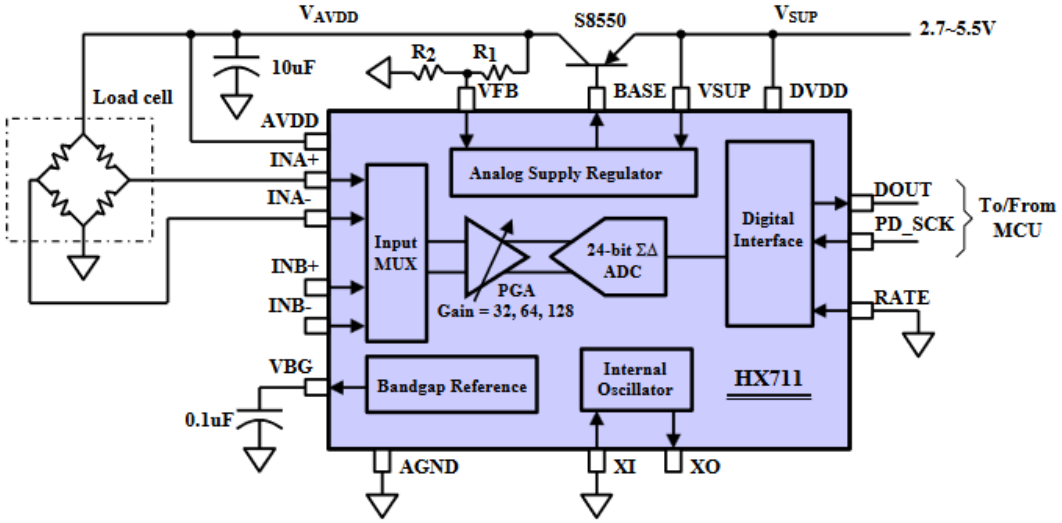
Κατά την χρήση του ενσωματωμένου ταλαντωτή, η ταχύτητα εξόδου των δεδομένων είναι συνήθως 10SPS (Samples Per Second) με RATE ίσο με λογικό 0 ή 80SPS με RATE ίσο με λογικό 1.

Όταν χρησιμοποιείται εξωτερικό ρολόι ή κρύσταλλος, η ταχύτητα εξόδου των δεδομένων είναι απευθείας ανάλογη με τη συχνότητα του ρολογιού ή του κρυστάλλου. Η χρήση ρολογιού ή κρυστάλλου 11.0592MHz οδηγεί σε μια ακριβή ταχύτητα εξόδου δεδομένων 10 (RATE=0) ή 80SPS (RATE=1).

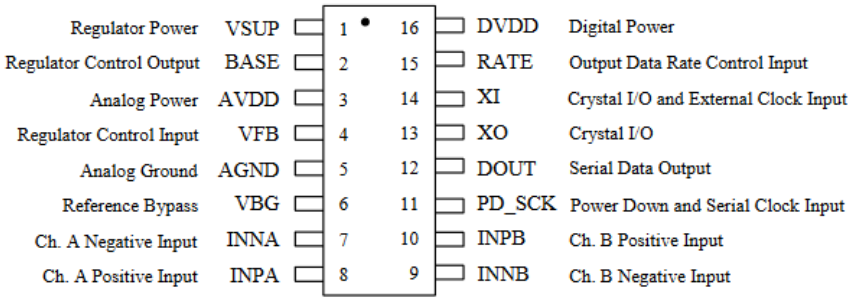
Τα 24 bits εξόδου δεδομένων βρίσκονται σε μορφή συμπληρώματος του 2. Όταν το διαφορικό σήμα εισόδου βγαίνει εκτός του εύρους των 24 bits, τα δεδομένα εξόδου θα κορεστούν στην τιμή 800000h (MIN) ή 7FFFFFh (MAX), μέχρι το σήμα εισόδου να επιστρέψει στο εύρος της εισόδου.

Οι ακροδέκτες PD_SCK και DOUT χρησιμοποιούνται για την ανάκτηση δεδομένων, την επιλογή εισόδου, την επιλογή ενίσχυσης και τους ελέγχους απενεργοποίησης της τροφοδοσίας.

Όταν τα δεδομένα εξόδου δεν είναι έτοιμα για ανάκτηση, ο ψηφιακός ακροδέκτης εξόδου DOUT είναι High. Το ρολόι σειράς εισόδου PD_SCK πρέπει να είναι Low. Όταν το DOUT πηγαίνει σε Low κατάσταση, υποδηλώνει ότι τα δεδομένα είναι έτοιμα για ανάκτηση. Εφαρμόζοντας 25~27 θετικές παλμικές ροές στον ακροδέκτη PD_SCK, τα δεδομένα μετατοπίζονται από τον ακροδέκτη εξόδου DOUT. Κάθε παλμός του PD_SCK μετατοπίζει ένα bit, ξεκινώντας από το πιο σημαντικό bit, μέχρι όλα τα 24 bits να μετατοπιστούν. Ο 25ος παλμός στην είσοδο PD_SCK θα επαναφέρει τον ακροδέκτη DOUT πίσω σε υψηλή κατάσταση. Ο έλεγχος της εισόδου και της επιλογής της ενίσχυσης ελέγχεται από τον αριθμό των παλμών της εισόδου PD_SCK (Πίνακας 3). Οι παλμοί ρολογιού PD_SCK δεν πρέπει να είναι λιγότεροι από 25 ή περισσότεροι από 27 μέσα σε ένα χρονικό διάστημα μετατροπής, για να αποφευχθεί η πρόκληση σφάλματος στη σειριακή επικοινωνία.



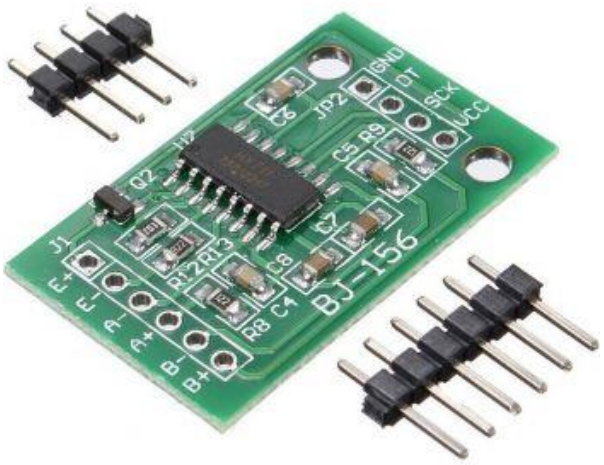
Σχήμα 2: Σχηματικό διάγραμμα του HX711



SOP-16L Package

Pin #	Name	Function	Description
1	VSUP	Power	Regulator supply: 2.7 ~ 5.5V
2	BASE	Analog Output	Regulator control output（NC when not used）
3	AVDD	Power	Analog supply: 2.6 ~ 5.5V
4	VFB	Analog Input	Regulator control input（connect to AGND when not used）
5	AGND	Ground	Analog Ground
6	VBG	Analog Output	Reference bypass output
7	INA-	Analog Input	Channel A negative input
8	INA+	Analog Input	Channel A positive input
9	INB-	Analog Input	Channel B negative input
10	INB+	Analog Input	Channel B positive input
11	PD_SCK	Digital Input	Power down control (high active) and serial clock input
12	DOUT	Digital Output	Serial data output
13	XO	Digital I/O	Crystal I/O (NC when not used)
14	XI	Digital Input	Crystal I/O or external clock input, 0: use on-chip oscillator
15	RATE	Digital Input	Output data rate control, 0: 10Hz; 1: 80Hz
16	DVDD	Power	Digital supply: 2.6 ~ 5.5V

Πίνακας 2: Ακροδέκτες του ολοκληρωμένου HX711



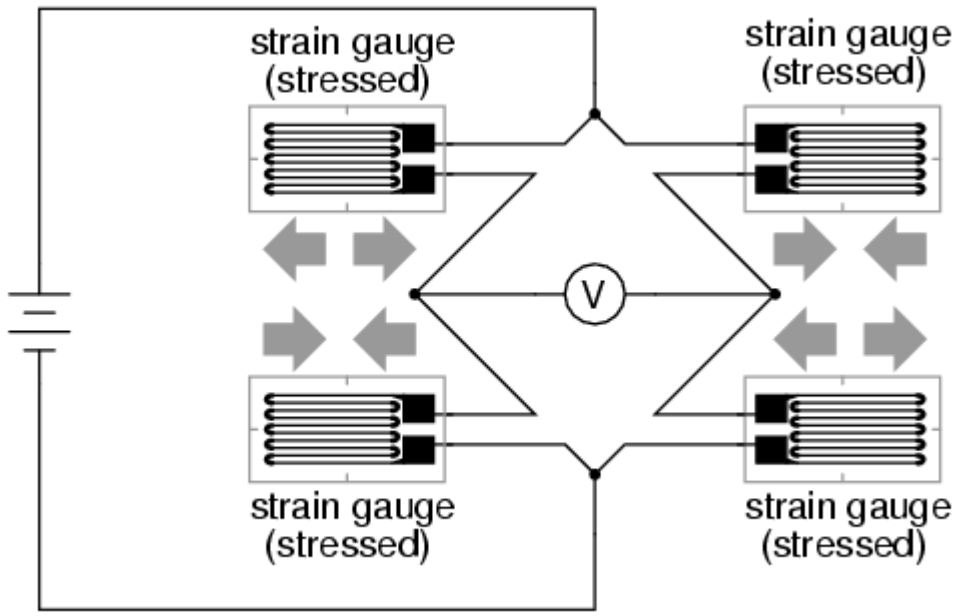
Εικόνα 16: Η πλακέτα με το ολοκληρωμένο. (https://grobotronics.com/)

3.3 Δυναμοκυψέλη

Οι δυναμοκυψέλες είναι αισθητήρια όργανα που χρησιμοποιούνται για να μετατρέψουν μια δύναμη σε ηλεκτρικό σήμα. Χρησιμοποιούνται ευρέως στα συστήματα ζύγισης.

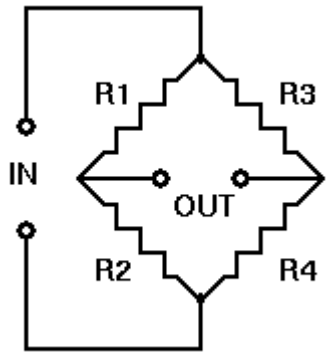
Η δυναμοκυψέλη, μετρά το βάρος του αντικειμένου (δοχείο με προϊόν, χοάνη κλπ.), μετρώντας τη δύναμη συμπίεσης που ασκείται. Υπάρχει δυνατότητα ζύγισης από 1 kg έως 100 ton. Ανάλογα με τη δυναμοκυψέλη και το εύρος ζύγισης, επιτυγχάνεται και η ανάλογη ακρίβεια.

Εικόνα 17: Δυναμοκυψέλη της συσκευής (https://grobotronics.com)



Εικόνα 18: Σχηματικό μιας δυναμοκυψέλης(https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all)

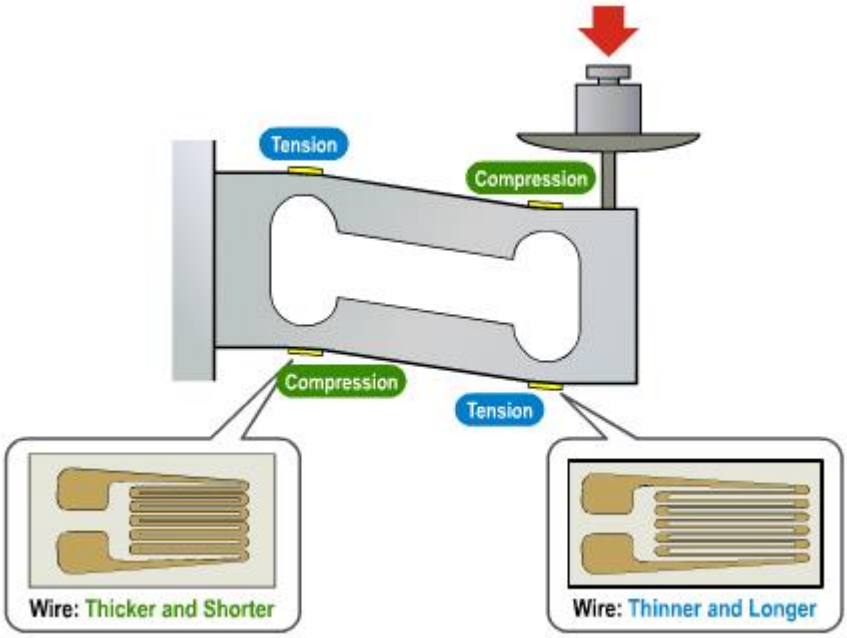
Από το παραπάνω σχήμα βλέπουμε ότι πρόκειται για μια γέφυρα Wheatstone, το οποίο σημαίνει ότι μπορεί να διασυνδεθεί με τον ADC HX711 και έπειτα στον μικροελεγκτή ώστε να πάρουμε μετρήσεις.



Εικόνα 19: Γέφυρα Wheatstone.(https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all)

Ο τύπος που υπολογίζει την τάση εξόδου είναι :

$$V_{out} = [R_3 / (R_3 + R_4) – R_1 /(R_1 + R_2)] \text{ x } V_{in}$$



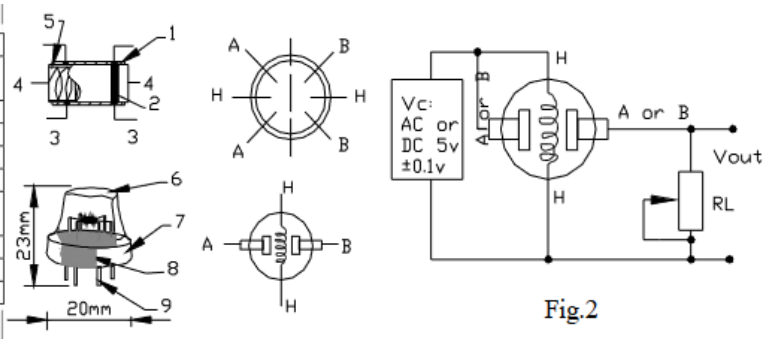
Εικόνα 20: Σχήμα με τις δυνάμεις που ασκούνται σε μία δυναμοκυψέλη. (https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all)

3.4 Αισθητήρας μεθανίου MQ-4

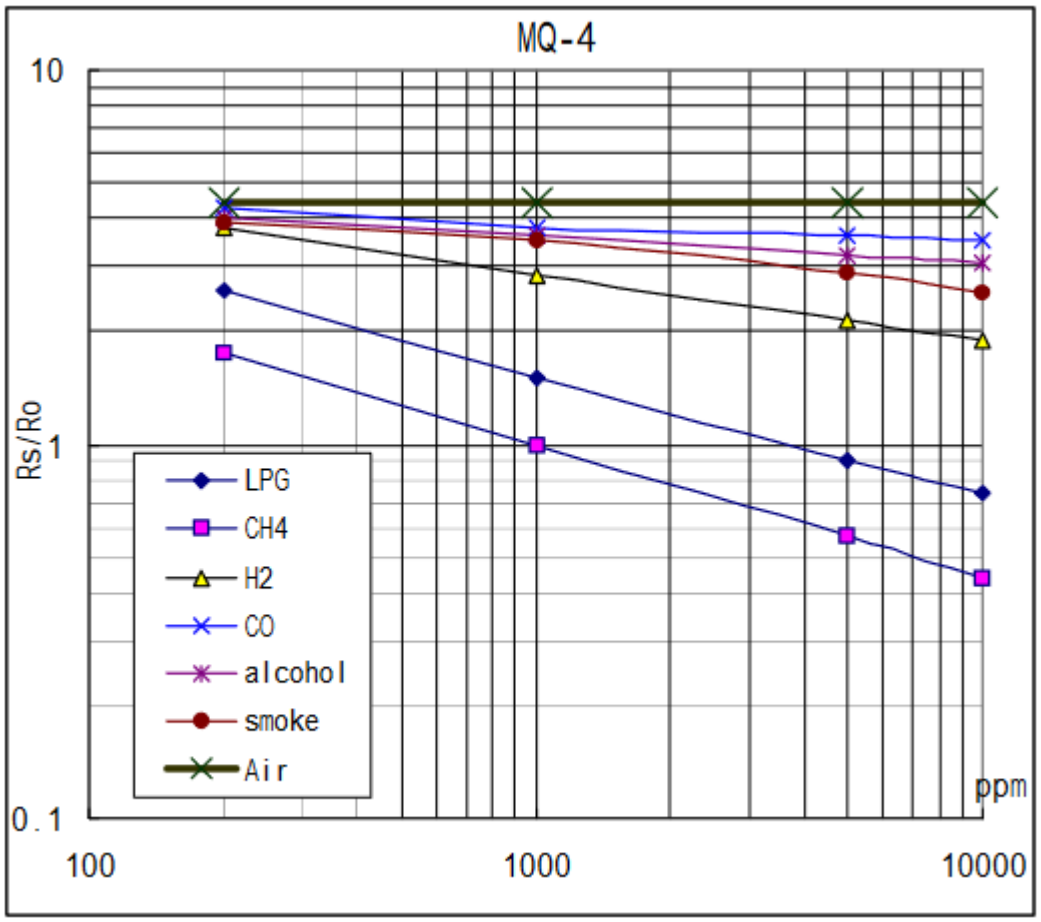
Εικόνα 21: Αισθητήρας μεθανίου MQ-4 (https://www.sparkfun.com/products/9404)

Ο συγκεκριμένος αισθητήρας είναι φτιαγμένος ώστε να ανιχνεύει την παρουσία του φυσικού αερίου και κυρίως του μεθανίου CH4. Το εύρος του είναι από 200 έως 10000ppm (Parts Per Million).
Τροφοδοτείται με 5V είτε DC, είτε AC.
Αποτελείται από ένα ηλεκτρόδιο αναφοράς, ένα ηλεκτρόδιο με επίστρωση διοξειδίου του κασσίτερου και μια αντίσταση η οποία ζεσταίνει τα ηλεκτρόδια ώστε να αυξάνει την ευαισθησία. Όταν το αέριο αντιδρά με τα ηλεκτρόδια τότε η αντίσταση τους μειώνεται, η αντίσταση είναι αντιστρόφως ανάλογη με την ποσότητα του αερίου.

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni



Εικόνα 22: Σχηματικό του αισθητήρα (https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf)



Πίνακας 3: Τυπικά χαρακτηριστικά ευαισθησίας για διάφορα αέρια.(https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf)

Οι παραπάνω μετρήσεις ισχύουν για θερμοκρασία 20°C, υγρασία 65%, συγκέντρωση O^2 21% και αντίσταση RL ίση με 20kΩ.
Η μεταβλητή Ro αντιπροσωπεύει την αντίσταση του αισθητήρα σε 1000ppm CH_4 σε καθαρό αέρα και η Rs την αντίσταση του αισθητήρα σε διάφορες συγκεντρώσεις από αέρια.

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
P_L	Load resistance	20KΩ	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 750mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10℃-50℃	
Tas	Storage Tem	-20℃-70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

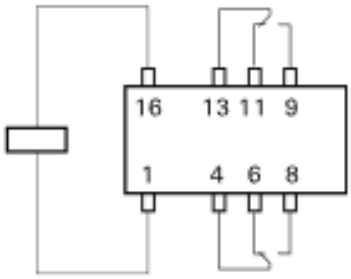
Symbol	Parameter name	Technical parameter	Ramark 2
Rs	Sensing Resistance	10KΩ - 60KΩ (1000ppm CH_4)	Detecting concentration scope: 200-10000ppm CH_4 , natural gas
α (1000ppm/ 5000ppm CH_4)	Concentration slope rate	≤0.6	
Standard detecting condition	Temp: 20℃±2℃ Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

Πίνακας 4: Τεχνικά χαρακτηριστικά του αισθητήρα. (https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf)

Βλέπω από P_H που είναι η κατανάλωση θερμάνσεως ότι είναι μέγιστο 750mw, για αυτό το λόγο επειδή το σύστημα θα τροφοδοτείται από μπαταρίες θα τοποθετηθεί ένα μικρορελέ το οποίο θα ενεργοποιεί τον αισθητήρα 2 λεπτά πριν πάρει μέτρηση ώστε να προλάβει να ζεσταθεί και αφού πάρει την μέτρηση θα απενεργοποιεί το ρελέ.

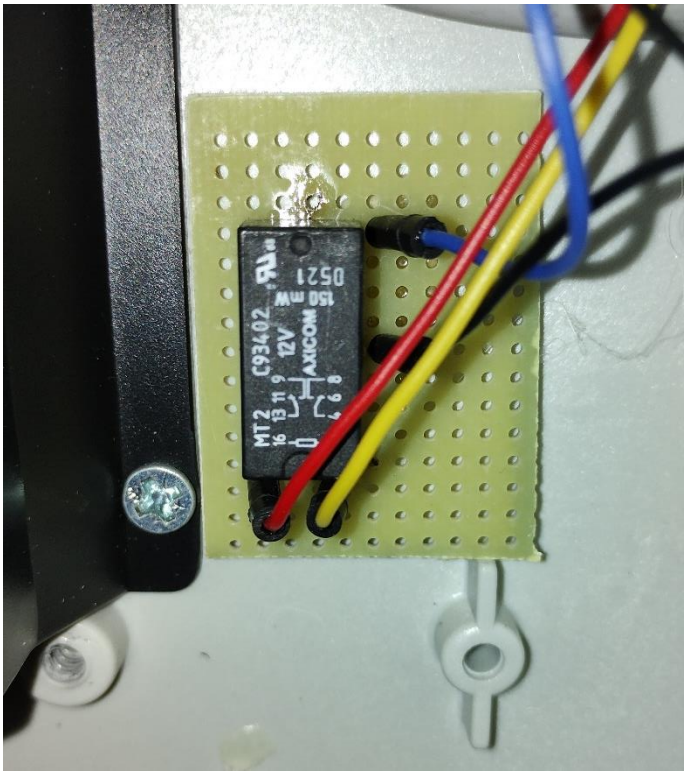
3.4.1 MT2 RELAY

Το εξάρτημα αυτό είναι ένα ρελέ που οπλίζει μια normally open και μια normally closed επαφή.



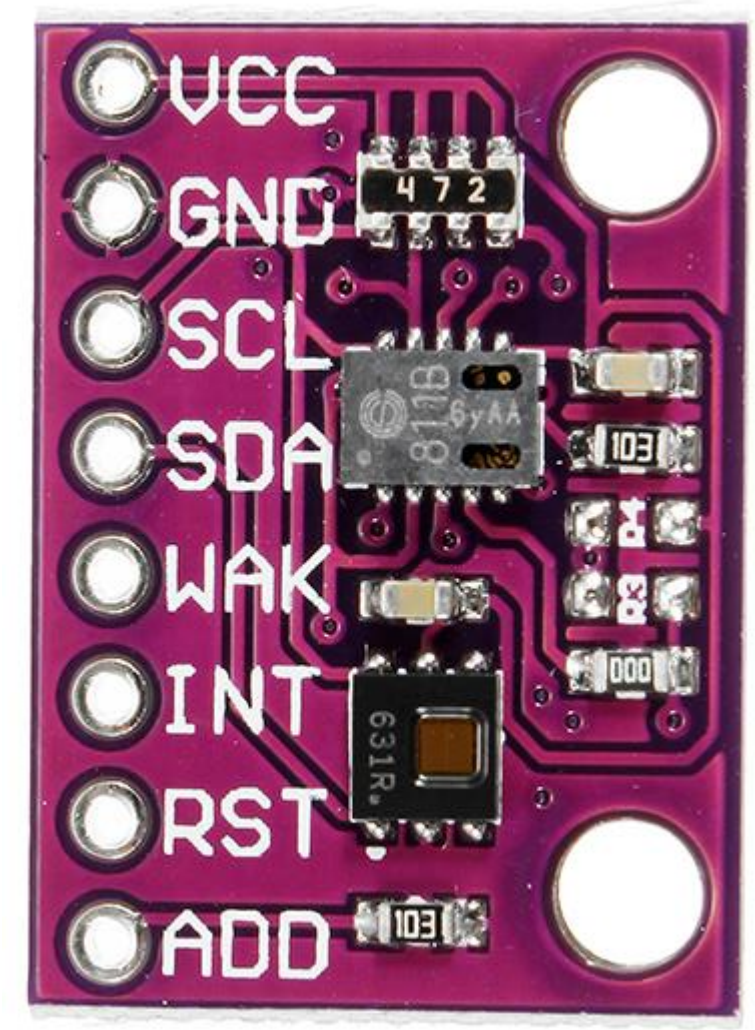
Εικόνα 23: Σχηματικό διάγραμμα του ρελέ.(https://www.digchip.com/datasheets/parts/datasheet/1721/MT2-C93402-pdf.php)

Η τάσεις στις οποίες οπλίζει είναι από 3V έως 48V συνεχής τάση.
Θα μπορούσε να τροφοδοτείται απ' ευθείας από κάποιο μικροελεγκτή, αντ' αυτού θα τροφοδοτείται από ένα mosfet control module (βλέπε κεφάλαιο 3.6) έτσι ώστε να μπορεί να υπάρχει οπτική επιβεβαίωση ότι λειτουργεί και για ηλεκτρική απομόνωση των μιρκοελεγκτών.



Εικόνα 24: Το ρελέ τοποθετημένο στην συσκευή.

3.5 ΑΙΣΘΗΤΗΡΑΣ CJMCU-8118



Εικόνα 25 : Ο αισθητήρας CJMCU-8118. (https://www.elecbee.com/en-26482-8118-CCS811-HDC1080-Carbon-Monoxide-CO-VOCs-Temperature-And-Humidity-Gas-Sensor-Module)

Ο αισθητήρας CJMCU-8118 είναι ουσιαστικά μια πλακέτα η οποία περιέχει δύο διαφορετικούς αισθητήρες, τον CCS811 και τον HDC1080. Ο ρόλος της πλακέτας είναι να παρέχει τις απαραίτητες εισόδους και εξόδους στους αισθητήρες δηλαδή την τάση λειτουργείας Vcc την γείωση τα SCL και SDA οπού είναι οι δίαυλοι επικοινωνίας για το πρωτόκολλο I^2C , το WAK είναι μια είσοδος με active low λογική οπού απενεργοποιεί τον αισθητήρα ώστε να έχει χαμηλή κατανάλωση της τάξης 100nA , το INT είναι επίσης μια active low λογικής πόρτα και χρησιμοποιείται από το ολοκληρωμένο σε περίπτωση που ξεπεραστεί κάποια καθορισμένη τιμή από τον χρήστη, το RST είναι μια active low είσοδος η οποία είναι συνδεδεμένη με το Vcc με μια pull up αντίσταση και τέλος η είσοδος ADD με την οποία γίνεται επιλογή διεύθυνσης για το I^2C του CCS811 για low είναι 0x5A (δεκαδικό 90) ενώ για high 0x5B (δεκαδικό 91).

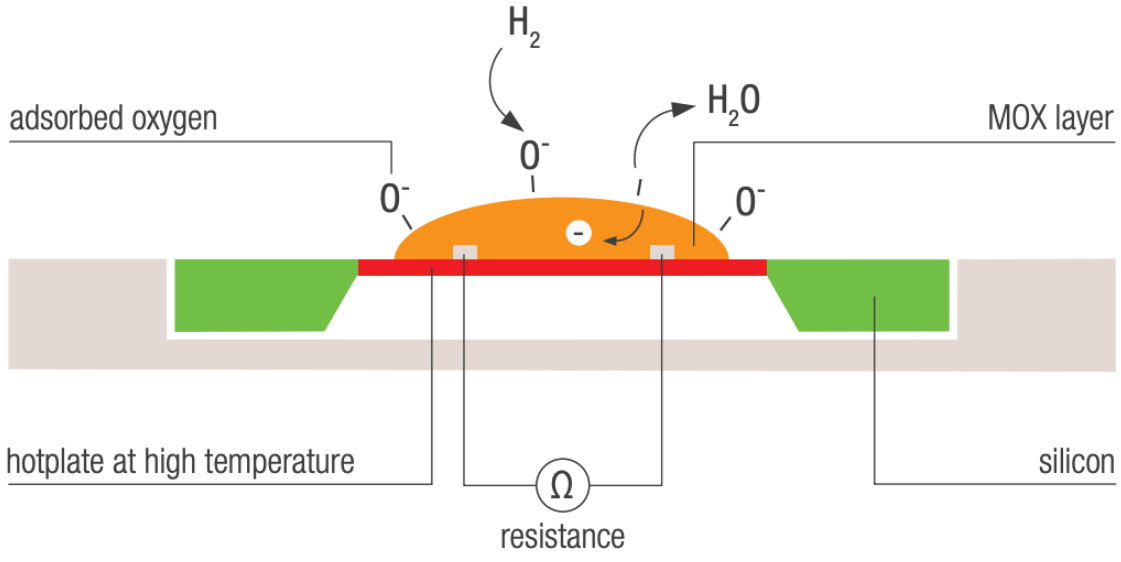
3.5.1 ΑΙΣΘΗΤΉΡΑΣ CSS811

Ο CSS811 είναι ένας αισθητήρας μέτρησης συνόλου πτητικών οργανικών ενώσεων (Total Volatile Organic Compounds – TVOC) τα οποία είναι μια μεγάλη ομάδα υγρών και αερίων, πολλές από τις οποίες είναι άχρωμες και άοσμες. Δεδομένου ότι είναι πτητικές ουσίες, τα υγρά εξατμίζονται εύκολα σε θερμοκρασία δωματίου.
Ορισμένες πτητικές οργανικές ενώσεις μπορεί να είναι άμεσα επιβλαβείς με τη μορφή καρκινογόνων ουσιών για την ανθρώπινη υγεία.

Για μια οικονομική λύση , η βιομηχανία έχει αναζητήσει διαφορετικούς τρόπους μέτρησης του διοξειδίου του άνθρακα. Αυτό μας οδηγεί σε μια παράμετρο που ονομάζεται eCO2 ή ισοδύναμο διοξείδιο του άνθρακα.
Πολλές συσκευές που δηλώνουν ότι μετρούν το CO2 στην πραγματικότητα αναφέρονται μόνο στο eCO2. Μια μέτρηση eCO2 προέρχεται από μια μέτρηση συνολικών αρωματικών οργανικών ενώσεων (TVOC).

Η αρχή λειτουργίας του αισθητήρα βασίζεται στην αρχή λειτουργίας των ημιαγωγών μετάλλου οξειδίου, η λειτουργία του βασίζεται στην αλλαγή της αγωγιμότητας του ευαίσθητου στον αέρα στρώματος ημιαγωγού MOX κατά την έκθεση στο αέριο, το οποίο μπορεί να μετρηθεί και να αναλυθεί.

Ανάλογα με τον τύπο του αερίου που θέλουμε να ανιχνευθεί αλλάζει και το ημιαγώγιμο υλικό.



Εικόνα 26: Δομή του αισθητηρίου. (https://atmotube.com/atmotube-support/how-does-atmotube-voc-sensor-work)

Symbol	Parameter	Min	Max	Units	Comments
Electrical Parameters					
V_{DD} (1)	Supply Voltage	1.8(2)	3.6	V	
I_{DD}	Supply Current		30	mA	In mode 1
P	Power Consumption		60	mW	In mode 1
Electrostatic Discharge					
ESD_{HBM}	Human Body Model	±2000		V	
Environmental Conditions					
T_{AMB}	Ambient Temperature for Operation	-5	50	°C	
T_{STRG}	Storage Temperature	-40	125	°C	
RH_{NC}	Relative Humidity (non-condensing)	10	95	%	
MSL	Moisture Sensitivity Level	1			Unlimited max. floor life time

Πίνακας 5: Χαρακτηριστικά του CSS811. (https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf)

Το αισθητήριο παρέχει 5 καταστάσεις :

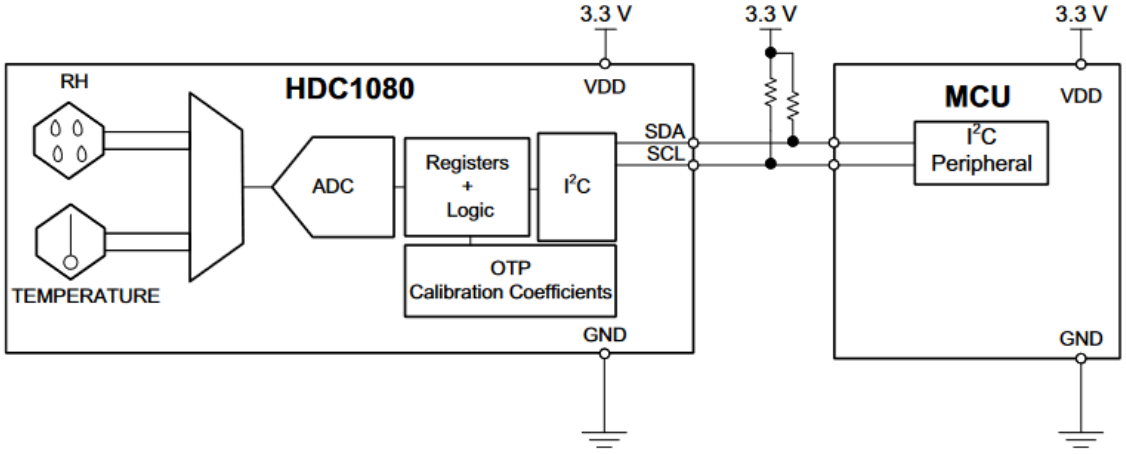
1. Ηρεμίας, λειτουργία χαμηλής ισχύος
2. Σταθερής ισχύος με μέτρηση ανά δευτερόλεπτο
3. Παλμικής θέρμανσης με μέτρηση ανά 10 δευτερόλεπτα
4. Χαμηλής ισχύος παλμική θέρμανση με μέτρηση ανά 60 δευτερόλεπτα
5. Σταθερής ισχύος με μέτρηση ανά 250ms

Τα εύρη μετρήσεων είναι για το eCO_2 είναι μεταξύ 400 – 8192ppm και οι τιμές εκτός ορίων αποκόπτονται και για τα TVOC είναι 0-1187ppb (Parts Per Billion).

Οι μετρήσεις ανακτώνται από τον μικροελεγκτή με την χρήση πρωτοκόλλου I^2C. (Βλέπε κεφάλαιο 4.4)

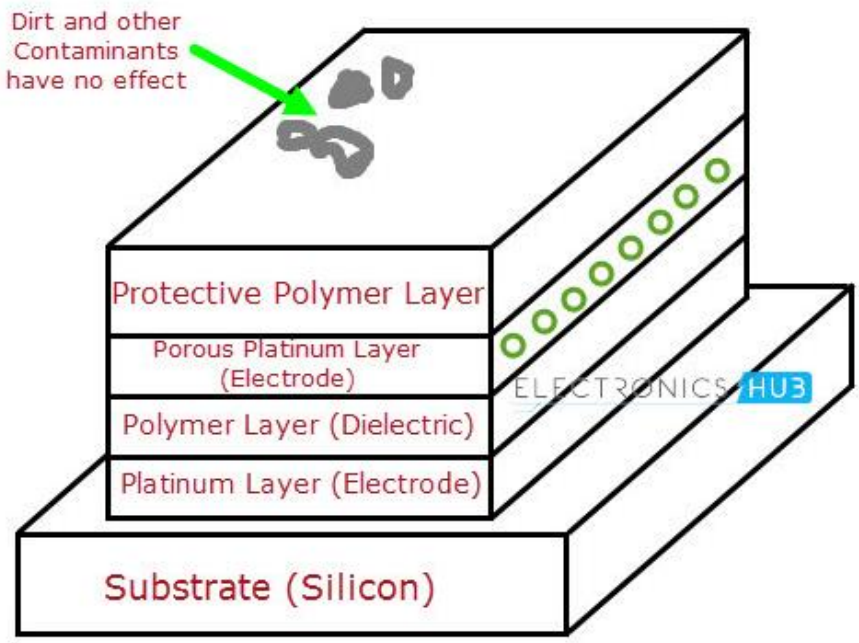
3.5.2 ΑΙΣΘΗΤΗΡΑΣ HDC1080

Ο HDC1080 είναι ένας αισθητήρας υγρασίας και θερμοκρασίας με αντίστοιχες ακρίβειες 2% και 0,2°C, έχει ενσωματωμένο μετατροπέα αναλογικό σε ψηφιακό σήμα με 14 bit υποστηρίζει κατάσταση χαμηλής ισχύος και συνδέεται σε δίαυλο I^2C. Το αισθητήριο της υγρασίας είναι ένας χωρητικός αισθητήρας υγρασίας και το θερμόμετρο ένας αισθητήρας με κενό ενεργειακής ζώνης πυριτίου.



Εικόνα 27 : Block διάγραμμα του αισθητήρα. (https://www.ti.com/lit/ds/symlink/hdc1080.pdf?ts=1692177762473&ref_url=https%253A%252F%252Fwww.google.com%252F)

Ο χωρητικός αισθητήρας υγρασίας μετράει την σχετική υγρασία χρησιμοποιώντας ένα πυκνωτή και ως διηλεκτρικό όπου η διηλεκτρική σταθερά του αλλάζει όταν αλλάζει η υγρασία του αέρα.



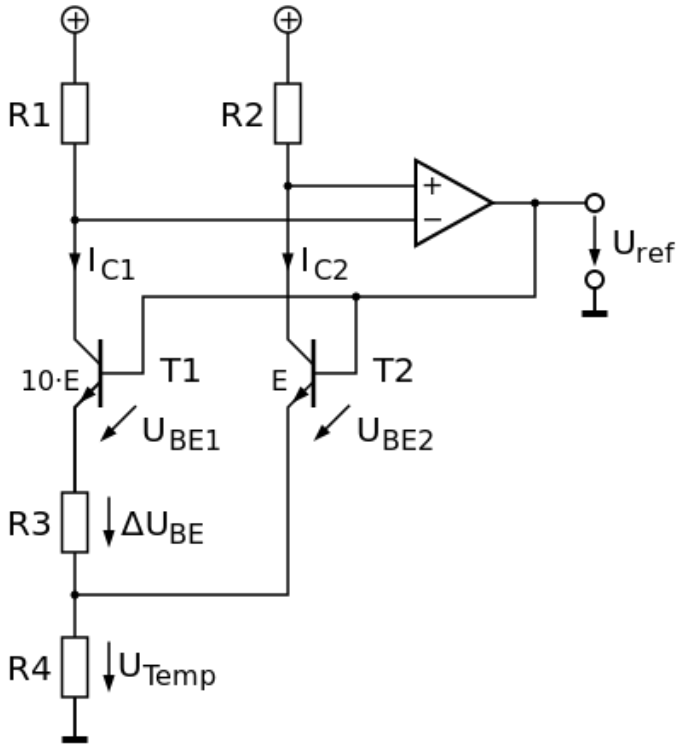
Εικόνα 28 : Δομή του αισθητήρα (https://www.electronicshub.org/humidity-sensor-types-working-principle/)

Σχετική υγρασία είναι το ποσό της υγρασίας στον αέρα σε σύγκριση με το πόση υγρασία μπορεί να κρατήσει ο αέρας στην ίδια θερμοκρασία, άρα καταλαβαίνουμε ότι οι είσοδοι για τον υπολογισμό της θερμοκρασίας είναι η υγρασία και η θερμοκρασία.

$$RH\,\% = \frac{P_{water\ vapor}}{P_{saturation\ water\ vapor}} \times 100$$

Αισθητήρας με κενό ενεργειακής ζώνης πυριτίου είναι ένας πολύ συνηθισμένος αισθητήρας θερμοκρασίας που χρησιμοποιείται σε ηλεκτρονικό εξοπλισμό το πλεονέκτημα του είναι ότι συμπεριλαμβάνεται σε ένα ενσωματωμένο σύστημα με πολύ μικρό κόστος.
Η αρχή λειτουργείας του βασίζεται στην τάση αγωγής μίας διόδου ή η τάση βάσης- εκπομπού σε ένα διπολικό τρανζίστορ, οπού εξαρτάται από την θερμοκρασία βάσει την παρακάτω συνάρτηση.

$$V_{BE} = V_{G0}\left(1 - \frac{T}{T_0}\right) + V_{BE0}\left(\frac{T}{T_0}\right) + \left(\frac{nkT}{q}\right)\ln\left(\frac{T_0}{T}\right) + \left(\frac{kT}{q}\right)\ln\left(\frac{I_C}{I_{C0}}\right)$$



Εικόνα 29 : Σχηματικό του κυκλώματος μέτρησης θερμοκρασίας.(https://en.wikipedia.org/wiki/File:Bandgap-reference.svg)

Συγκρίνοντας τις τάσεις Vbe1 και Vbe2 στην ίδια θερμοκρασία αλλά με δυο διαφορετικά ρεύματα η παραπάνω σχέση γίνεται :

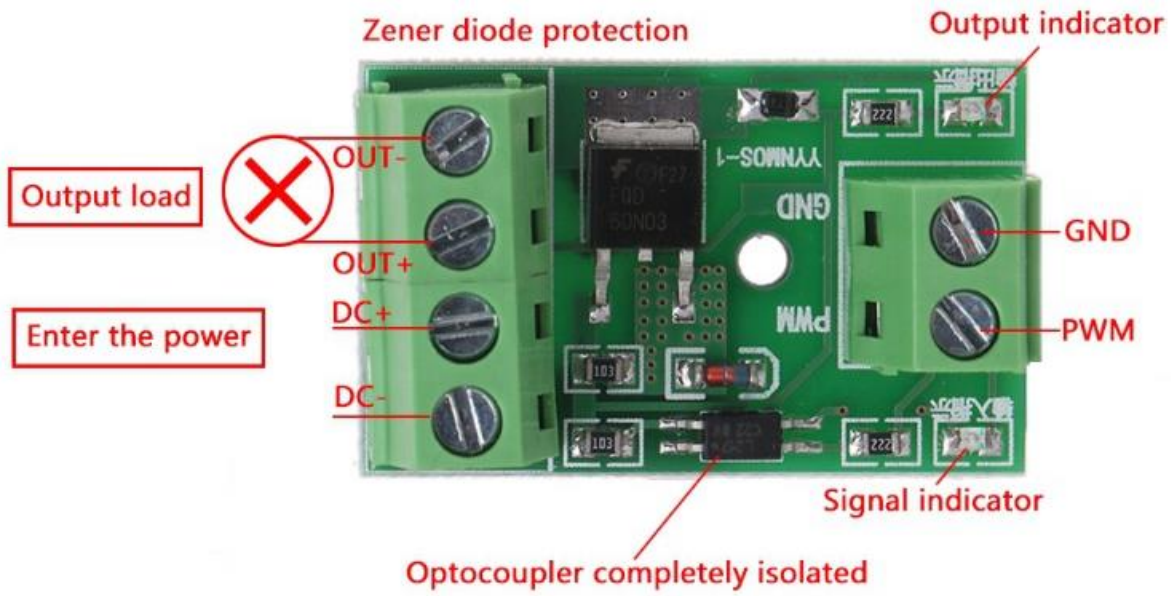
$$\Delta V_{BE} = \frac{kT}{q} \cdot \ln\left(\frac{I_{C1}}{I_{C2}}\right)$$

3.6 Mosfet Control Module

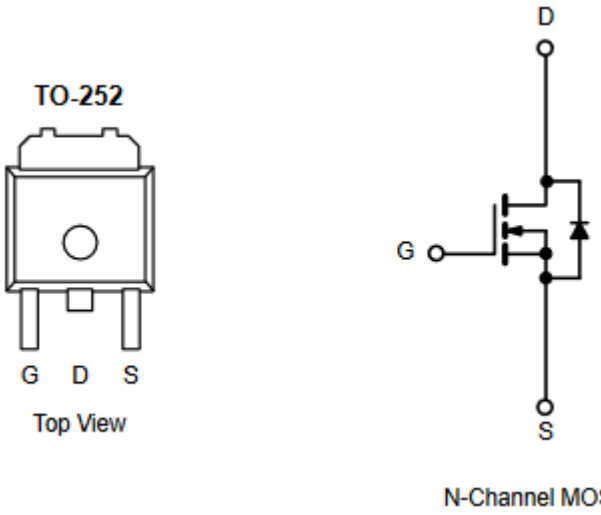
Είναι μια πλακέτα με ένα mosfet και η χρήση της είναι να τροφοδοτεί φορτία βάσει τις εξόδους του μικροελεγκτή.

Το σήμα εισόδου είναι μεταξύ 3,3 - 20 Volt PWM σήμα με συχνότητα 0 έως 20kHz, τροφοδοτείται με τάσεις 5 έως 27 Volt και μέγιστο ρεύμα 10A.
Όταν παλμοδοτείται με PWM μπορεί γίνει έλεγχος της ισχύος στην έξοδο του.

Για προστασία του μικροελεγκτή το κύκλωμα σηματοδότησης είναι απομονωμένο από το κύκλωμα ισχύος με optocoupler και επίσης το mosfet προστατεύεται με μια εσωτερική δίοδο Zener για προστασία από ανάστροφα ρεύματα.
Ο κωδικός του mosfet είναι 60N03.



Εικόνα 30: Η πλακέτα του mosfet. (https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/modules/converter/mosfet-with-optocoupler-isolation-driver-module-with-pwm-control-3-20v/)



Εικόνα 31: Σχηματικό του mosfet.(https://www.alldatasheet.com/datasheet-pdf/pdf/1255185/VBSEMI/60N03.html)

SPECIFICATIONS (T_J = 25 °C, unless otherwise noted)						
Parameter	**Symbol**	**Test Conditions**	**Min.**	**Typ.**	**Max.**	**Unit**
Static						
Drain-Source Breakdown Voltage	V_{DS}	V_{GS} = 0 V, I_D = 250 µA	30			V
V_{DS} Temperature Coefficient	$\Delta V_{DS}/T_J$	I_D = 250 µA		35		mV/°C
$V_{GS(th)}$ Temperature Coefficient	$\Delta V_{GS(th)}/T_J$			- 7.5		
Gate-Source Threshold Voltage	$V_{GS(th)}$	V_{DS} = V_{GS}, I_D = 250 µA	1.0		2.5	V
Gate-Source Leakage	I_{GSS}	V_{DS} = 0 V, V_{GS} = ± 20 V			± 100	nA
Zero Gate Voltage Drain Current	I_{DSS}	V_{DS} = 30 V, V_{GS} = 0 V			1	µA
		V_{DS} = 30 V, V_{GS} = 0 V, T_J = 55 °C			10	
On-State Drain Current[a]	$I_{D(on)}$	V_{DS} ≥ 5 V, V_{GS} = 10 V	90			A
Drain-Source On-State Resistance[a]	$R_{DS(on)}$	V_{GS} = 10 V, I_D = 38.8 A		0.005		Ω
		V_{GS} = 4.5 V, I_D = 37 A		0.006		
Forward Transconductance[a]	g_{fs}	V_{DS} = 15 V, I_D = 38.8 A		160		S

Πίνακας 6: Χαρακτηριστικά του mosfet. (https://www.alldatasheet.com/datasheet-pdf/pdf/1255185/VBSEMI/60N03.html)

Το optocoupler είναι το ACPL-217, είναι ένα dc ενός καναλιού φωτοτρανζίστορ οπτοσυζευκτήρας, περιλαμβάνει μια δίοδο εκπομπής φωτός (LED) που έχει οπτική σύζευξη με ένα φωτοτρανζίστορ.

Σημαντικό χαρακτηριστικό είναι η τάση μόνωσης εισόδου εξόδου που είναι στα 3000 Vrms και ο χρόνος απόκρισης είναι στα 2μs.

- Pin 1 Anode
- Pin 2 Cathode
- Pin 3 Emitter
- Pin 4 Collector

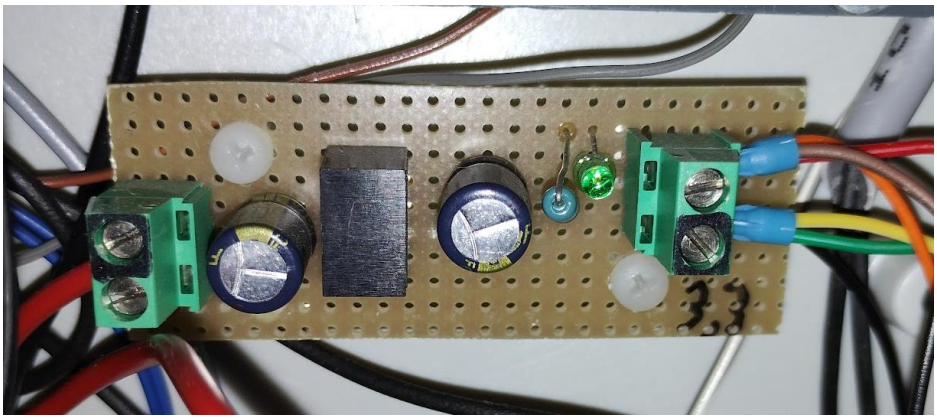
Εικόνα 32: Σχηματικό του optocoupler. (https://www.mouser.com/datasheet/2/38/AV02-0470EN-189984.pdf)

Parameter	Symbol	Min.	Typ.	Max.	Units	Test Conditions
Forward Voltage	V_F	-	1.2	1.4	V	$I_F = 20mA$
Reverse Current	I_R	-	-	10	μA	$V_R = 5V$
Terminal Capacitance	C_t	-	30	-	pF	$V = 0, f = 1MHz$
Collector Dark Current	I_{CEO}	-	-	100	nA	$V_{CE} = 48V, I_F = 0\ mA$
Collector-Emitter Breakdown Voltage	BV_{CEO}	80	-	-	V	$I_C = 0.5\ mA, I_F = 0\ mA$
Emitter-Collector Breakdown Voltage	BV_{ECO}	7	-	-	V	$I_E = 100\ \mu A, I_F = 0\ mA$
Current Transfer Ratio	CTR	50	-	600	%	$I_F = 5\ mA, V_{CE} = 5V$
Saturated CTR	CTR(sat)	-	100	-	%	I_F=1mA, $V_{CE} = 0.4V$
Collector-Emitter Saturation Voltage	V_{CE}(sat)	-	-	0.4	V	$I_F = 8mA, I_C = 2.4mA$
Isolation Resistance	R_{ISO}	$5x10^{10}$	$1x10^{11}$	-	Ω	DC500V, R.H. 40~60%
Floating Capacitance	C_F	-	0.6	1	pF	$V = 0, f = 1MHz$
Cut-off Frequency (-3dB)	F_C	-	80	-	kHz	$V_{CC} = 5V, I_C = 2\ mA,$ $R_L = 100Ω$
Response Time (Rise)	t_r	-	2	-	μs	$V_{CC} = 10V, I_C = 2\ mA,$ $R_L = 100Ω$
Response Time (Fall)	t_f	-	3	-	μs	
Turn-on Time	t_{on}	-	3	-	μs	
Turn-off Time	t_{off}	-	3	-	μs	
Turn-ON Time	t_{ON}	-	2	-	μs	$V_{CC} = 5V, I_F = 16\ mA,$ $R_L = 1.9kΩ$
Storage Time	T_S	-	25	-	μs	
Turn-OFF Time	t_{OFF}	-	40	-	μs	
Common Mode Rejection Voltage	CMR	-	10	-	kV/μs	Ta=25°C, R_L=470Ω, V_{CM}=1.5kV(peak), I_F=0mA, V_{CC}=9V, V_{np}=100mV

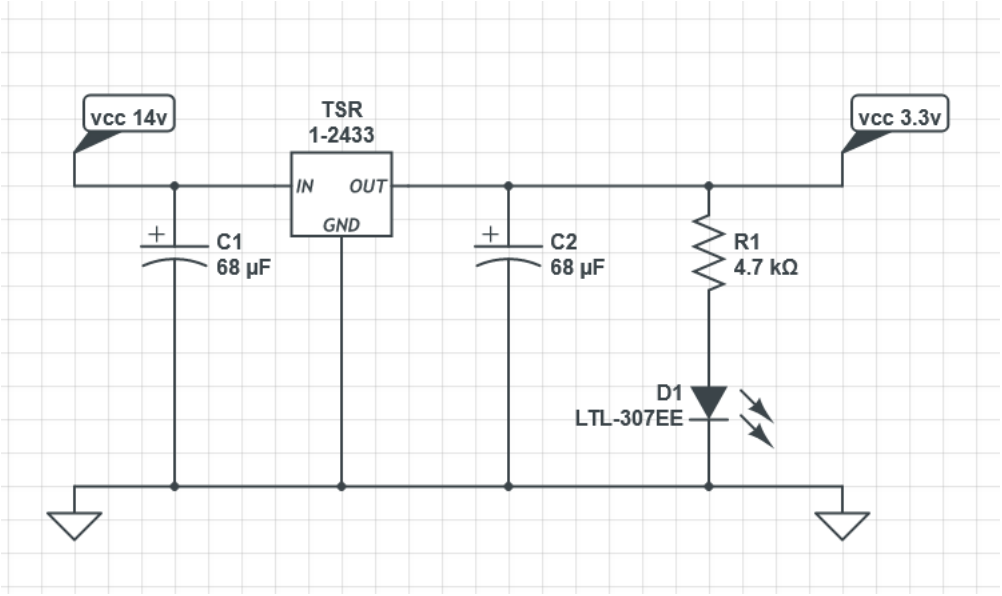
Πίνακας 7: Προδιαγραφές του οπτοσυζευκτήρα. (https://www.mouser.com/datasheet/2/38/AV02-0470EN-189984.pdf)

3.7 ΡΥΘΜΙΣΤΗΣ ΤΑΣΗΣ TSR 1-2433

Ο TSR 1-2433 πρόκειται για ένα step down ρυθμιστή συνεχούς τάσης για εύρος εισόδου 4,75 V – 36 V και με μέγιστη ισχύ εξόδου 1000mA. Έχει μεγάλη απόδοση στα 96% και επιτρέπει λειτουργεία σε πλήρη ισχύ στους 60^{o}C χωρίς να χρειάζεται επιπλέον ψύξη, η ακρίβεια εξόδου είναι στο 2% και το ρεύμα ηρεμίας είναι στα 2mA που το κάνει ιδανικό για χρήση σε κυκλώματα τροφοδοτούμενα από μπαταρίες.
Έχουν επίσης τοποθετηθεί στην είσοδο και στην έξοδο πυκνωτές για να γίνεται εξομάλυνση της τάσης καθώς και ένα LED για ενδεικτικό λειτουργείας.



Εικόνα 33: Ο ρυθμιστής στην συσκευή.



Εικόνα 34: Σχηματικό του ρυθμιστή όπως είναι υλοποιημένο για την συσκευή.

Output Specifications			
Voltage Set Accuracy			±2% max.
Regulation	- Input Variation (Vmin - Vmax)		0.2% max.
	- Load Variation (10 - 100%)		0.6% max. (1.2 & 1.5 Vout models)
			0.4% max. (other models)
Ripple and Noise (20 MHz Bandwidth)		9 Vin models:	50 mVp-p typ.
		12 Vin models:	50 mVp-p typ.
		24 Vin models:	75 mVp-p typ.
Capacitive Load			470 µF max.
Minimum Load			Not required
Temperature Coefficient			±0.015 %/K max.
Start-up Overshoot Voltage			1% max.
Short Circuit Protection			Continuous, Automatic recovery
Output Current Limitation			250% typ. of Iout max.
Transient Response	- Peak Variation		150 mV typ. / 200 mV max. (50% Load Step)
	- Response Time		250 µs typ. / 350 µs max. (50% Load Step)

EMC Specifications			
EMI Emissions	- Conducted Emissions		EN 55032 class A (with external filter)
	- Radiated Emissions		EN 55032 class A (with external filter)
		External filter proposal:	www.tracopower.com/overview/tsr1

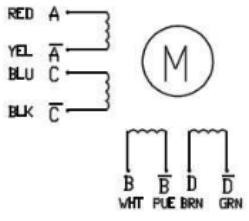
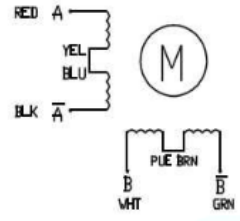
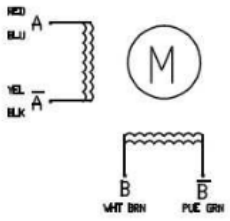
General Specifications			
Relative Humidity			95% max. (non condensing)
Temperature Ranges	- Operating Temperature		–40°C to +85°C
	- Storage Temperature		–55°C to +125°C
Power Derating	- High Temperature		2.4 %/K above 60°C
		See application note:	www.tracopower.com/overview/tsr1
Over Temperature Protection Switch Off	- Protection Mode		150°C typ. (Automatic recovery)
	- Measurement Point		Internal IC temperature
Cooling System			Natural convection (20 LFM)

Πίνακας 8: Χαρακτηριστικά του TSR 1-2433.(https://www.tracopower.com/sites/default/files/products/datasheets/tsr1_datasheet.pdf)

3.8 ΚΙΝΗΤΗΡΑΣ ΚΑΔΟΥ 60bygh450d-02

Εικόνα 35: Ο κινητήρας. (https://www.sorotec.de/shop/Stepping-Motor-4-2A-Bipolar-3-NM-DS-2599.html)

Ο συγκεκριμένος κινητήρας είναι ένας υβριδικός βηματικός κινητήρας συνεχούς ρεύματος με της δυνατότητα τροφοδοσίας 2 ή 4ων φάσεων. Στις 4 φάσεις κάθε τύλιγμα τροφοδοτείται ξεχωριστά ενώ για 2 φάσεις θα πρέπει να συνδεθούν μεταξύ τους ανά ζεύγη είτε σε σειρά είτε παράλληλα έτσι ώστε να γίνουν 2 τυλίγματα.
Στην κατασκευή έχουν συνδεθεί μεταξύ τους τα τυλίγματα σε σειρά.

	Unipolar	Bipolar Serial	Bipolar Parallel
			
No. of phases	4	2	2
Step Angle	1.8 ± 0.09°		
Rated Voltage (V)	4.38	6.13	3.07
Current (A)	3.0	2.1	4.2
Resistance (Ω)	1.46	2.92	0.73
Inductance (mH)	3.3	13	3.3
Holding Torque coils on DC (Nm)	2.4	3.3	3.3
Holding Torque step mode (Nm)	1.7	2.3	2.3
Rotor Inertia (gcm²)	840		
Insulation Class	B		
Insulation Resistance	100MΩ, 500VDC		
Weight (kg)	1.5		

Πίνακας 9: Τεχνικές πληροφορίες του κινητήρα. (https://mecheltron.com/sites/default/files/webresources/MechanicalElectroMech/StepperMotors/pdf/datasheets-steppermotors/60BYGH450D-02.pdf)

Ο Υβριδικός Κινητήρας είναι ο συνδυασμός δύο κινητήρων, όπως ένας μόνιμου μαγνήτη και ένας κινητήρας μεταβλητής αντίστασης. Η αρχή λειτουργείας του υβριδικού κινητήρα είναι ότι ο δρομέας του κινητήρα είναι μαγνητισμένος αξονικά, παρομοίως με τον κινητήρα μονίμου μαγνήτη, ενώ το στάτορας ενεργοποιείται ηλεκτρομαγνητικά, παρόμοια με έναν βηματικό κινητήρα. Έτσι, είναι μια μονάδα που μετατρέπει ηλεκτρικούς παλμούς σε γωνιακή μετατόπιση.

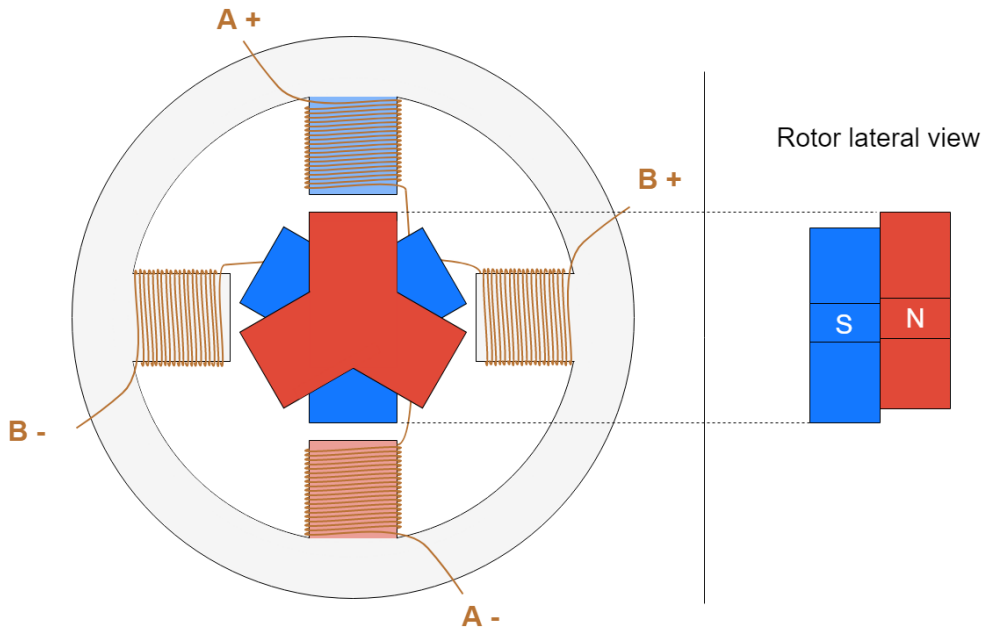
Σε σύγκριση με άλλους τύπους, αυτός ο τύπος κινητήρα παρέχει υψηλή ροπή με μικρότερη γωνία βηματισμού και έχει καλή δυναμική ιδιότητα.
Για την λειτουργεία του κινητήρα χρειάζεται να τροφοδοτηθεί από ένα σύστημα οδήγησης.

Τα πλεονεκτήματα του είναι ότι :

- Έχει μεγάλη ροπή
- Έχει ροπή αδράνειας τροφοδοτώντας τα τυλίγματα του όταν είναι σε στάση
- Έχει μικρή απόσταση μεταξύ των βημάτων
- Η απόδοση του κινητήρα είναι υψηλή σε μικρές ταχύτητες

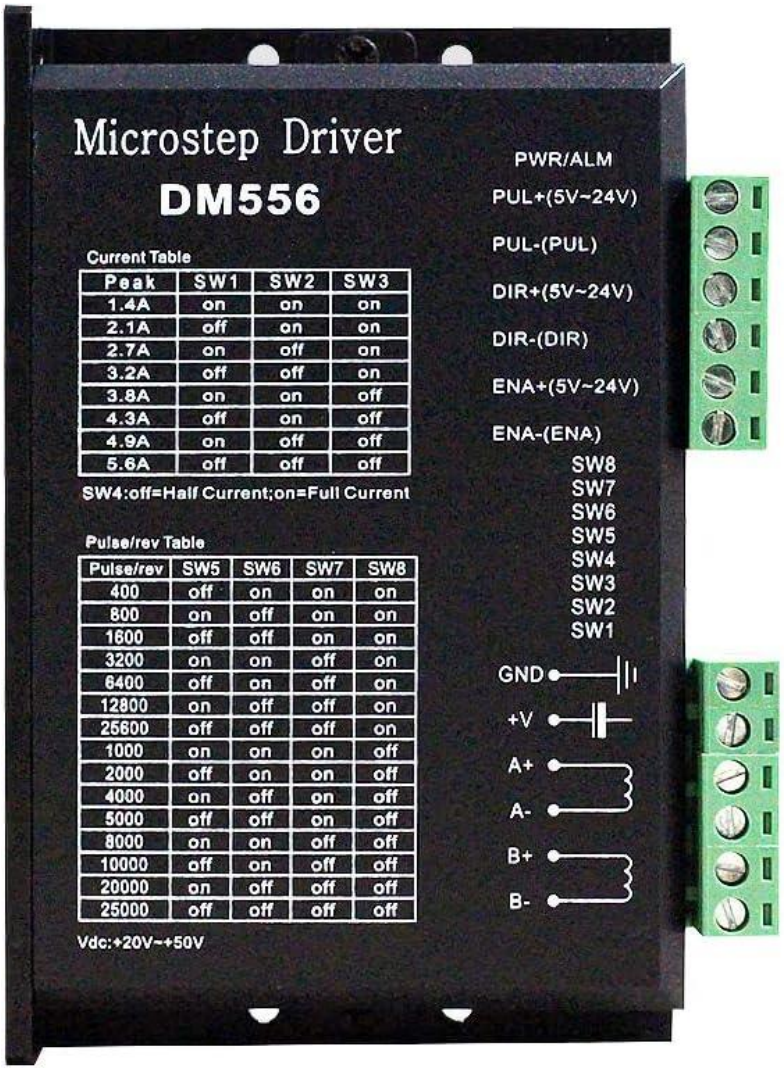
Τα μειονεκτήματα είναι :

- Ότι έχει μεγάλη αδράνεια
- Σαν σύστημα είναι βαρύ
- Οι επιδόσεις επηρεάζονται λόγω της μαγνητικής δύναμης
- Είναι ακριβός



Εικόνα 36: Απλοποιημένη δομή του κινητήρα. (https://www.monolithicpower.com/en/stepper-motors-basics-types-uses)

3.9 Σύστημα οδήγησης κινητήρα DM556

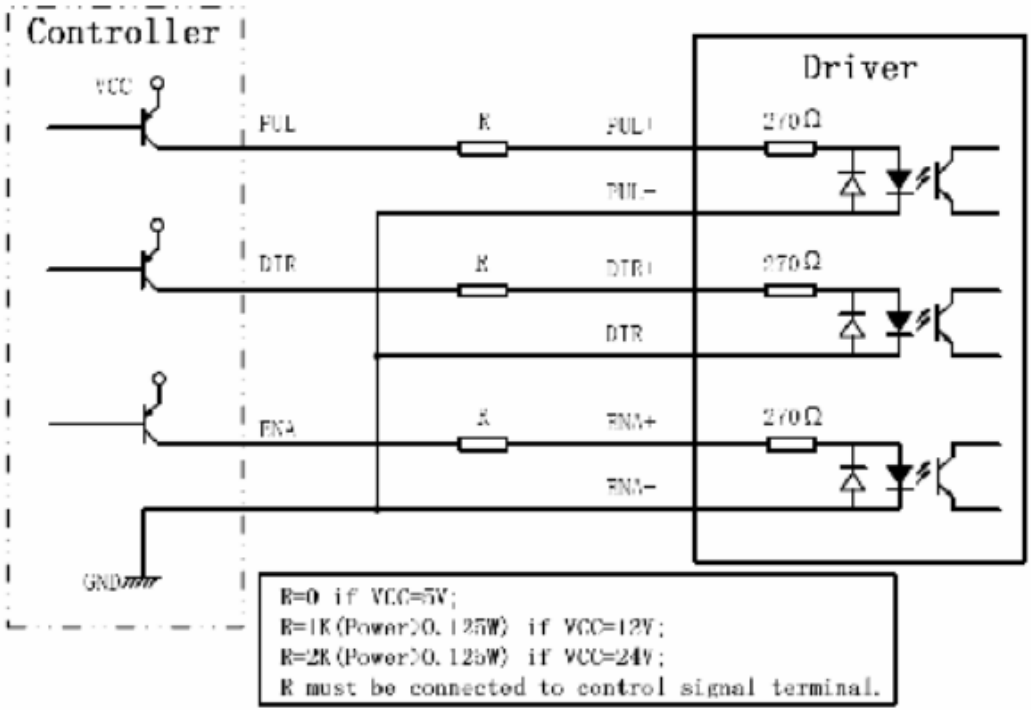


Εικόνα 37 : Το DM 556. (https://www.amazon.com/Digital-Stepper-Controller-2-phase-subdivision/dp/B08J3NFF9V)

Το DM556 είναι ένα πλήρως ψηφιακό σύστημα οδήγησης βηματικών κινητήρων, παρέχει ομαλότητα στο σύστημα και ιδανική ροπή για όλο το εύρος ταχυτήτων. Έχει την δυνατότητα αυτοδίαγνωσης και μπορεί να τροφοδοτήσει ένα ευρύ φάσμα βηματικών κινητήρων.
Οι δυνατότητες του είναι ότι έχει λειτουργεία μικρό-βημάτων για ακριβέστερη θέση, η έξοδος του μπορεί να είναι από 0.5A – 5.6A παλμοδότηση μέχρι τα 200 KHz, υποστηρίζει TTL πρωτόκολλο επικοινωνίας, μείωση ρεύματος αδράνειας και εναλλαγή δεξιόστροφης είτε αριστερόστροφης περιστροφής του κινητήρα.

Οι είσοδοι του είναι οι εξής :

- Vcc : Συνδέεται η τροφοδοσία του που θα πρέπει να είναι συνεχής τάση μεταξύ 20V και 50V, όμως στην κατασκευή τροφοδοτείται στα 14V και λειτουργεί κανονικά.
- GND : Συνδέεται η γείωση ή αλλιώς το πλην.
- Τα PUL+ και PUL- : Είναι για την παλμοδότηση του, θα πρέπει να είναι πλάτους 5V τετραγωνικοί παλμοί. Για παραπάνω τάσεις θα πρέπει να συνδεθούν αντιστάσεις για μείωση της τάσης. Στην κατασκευή οι παλμοί είναι στα 3,3V πλάτος. Κάθε παλμός είναι και ένα βήμα στον κινητήρα.
- Τα DIR+ και DIR- : Καθορίζει την φορά περιστροφής του κινητήρα. Χρειάζεται απλά ένα σήμα για να καθορίσει την κατάσταση του.
- Τα ENA+ και ENA- : Καθορίζουν αν η έξοδος για την τροφοδοσία του κινητήρα είναι ενεργή ή ανενεργή. Είναι ένα σήμα με active low λογική. Είναι ένας πολύ σημαντικός παράγοντας για το σύστημα αυτός διότι ο κινητήρας κατασκευαστικά όταν είναι σε στάση διατηρεί τα τυλίγματα του ενεργά με αποτέλεσμα να μένει κλειδωμένος σε μια θέση. Αυτό έχει ως αποτέλεσμα να καταναλώνει ρεύμα απενεργοποιώντας την έξοδο του οδηγού μπορούμε να έχουμε εξοικονόμηση ενέργειας.



Εικόνα 38: Συνδεσμολογία του συστήματος. (https://www.leadshine.com.ua/pdf/dm566.pdf)

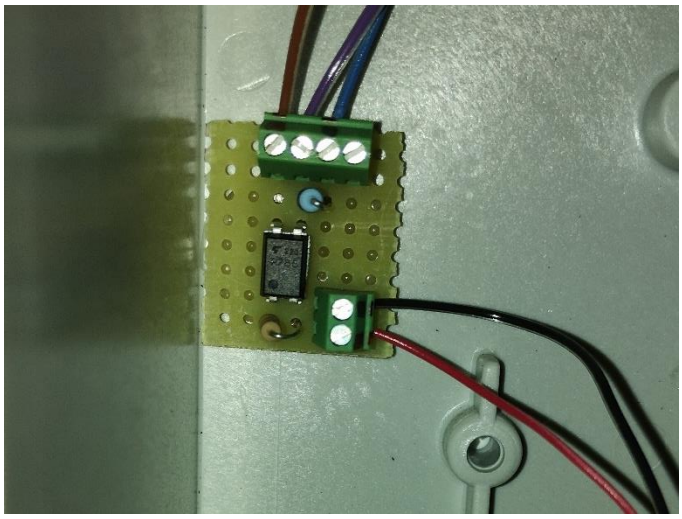
3.10 ΡΥΘΜΙΣΤΗΣ ΤΑΣΗΣ ΕΞΟΔΟΥ ΦΩΤΟΒΟΛΤΑΙΚΟΥ

Εικόνα 39: Ο ρυθμιστής τάσης.
(https://www.haitronic.cn/index.php?route=product/product&path=91_141&product_id=1375&limit=100)

Είναι ένα σύστημα ανεπτυγμένο από την Haitronic με κωδικό HS0055 το οποίο παίρνει σαν είσοδο την έξοδο ενός φωτοβολταϊκού και είτε την ρυθμίζει στην απαιτούμενη τάση εξόδου είτε φορτίζει τις μπαταρίες τις οποίες όταν δεν παρέχεται αρκετή ενέργεια από το φωτοβολταϊκό ξεκινάει και τις εκφορτίζει μέχρι να πέσει η τάση σε ένα ελάχιστο όριο, έπειτα απενεργοποιεί την έξοδο. Η κατάσταση της εξόδου ρυθμίζεται από την τάση των μπαταριών, την έξοδο του φωτοβολταϊκού, από τον φυσικό φωτισμό και την ώρα. Ως έξοδο έχει και δυο θύρες USB για τροφοδοσία 5V.

Η τάση λειτουργείας είναι στα 12V ή στα 24V και το ρεύμα εξόδου είναι στα 30A, αντίστοιχες θα πρέπει να είναι και οι τάσεις εξόδου των μπαταριών είτε στα 12V είτε στα 24V.
Έχει αισθητήρα φωτός και εσωτερικό ρολόι και παρέχει προστασία από υπερένταση, από βραχυκύκλωμα και από ανάστροφα ρεύματα.

3.10.1 ΦΩΤΟΣΥΖΕΥΚΤΉΡΑΣ TLP 785



Για να μπορεί να γνωρίζει ο μικροελεγκτής την κατάσταση της εξόδου του ρυθμιστή τοποθετήθηκε ένα optocoupler με μια pull down αντίσταση. Η τάση λειτουργείας είναι από 5V έως 24V.

Έγινε επιλογή optocoupler για ηλεκτρική απομόνωση της εξόδου από την είσοδο του ελεγκτή και για χαμηλή κατανάλωση ισχύος.

3.10.2 ΦΩΤΟΒΟΛΤΑΙΚΟ ΠΑΝΕΛ

Εικόνα 40: Το φωτοβολταϊκό πάνελ.

Ως παροχή ενέργειας για τη συσκευή είναι το φωτοβολταϊκό της φωτογραφίας. Είναι της εταιρίας SOLARFAM και είναι το μοντέλο SZ-50-36M.
Τα χαρακτηριστικά του είναι τα εξής :

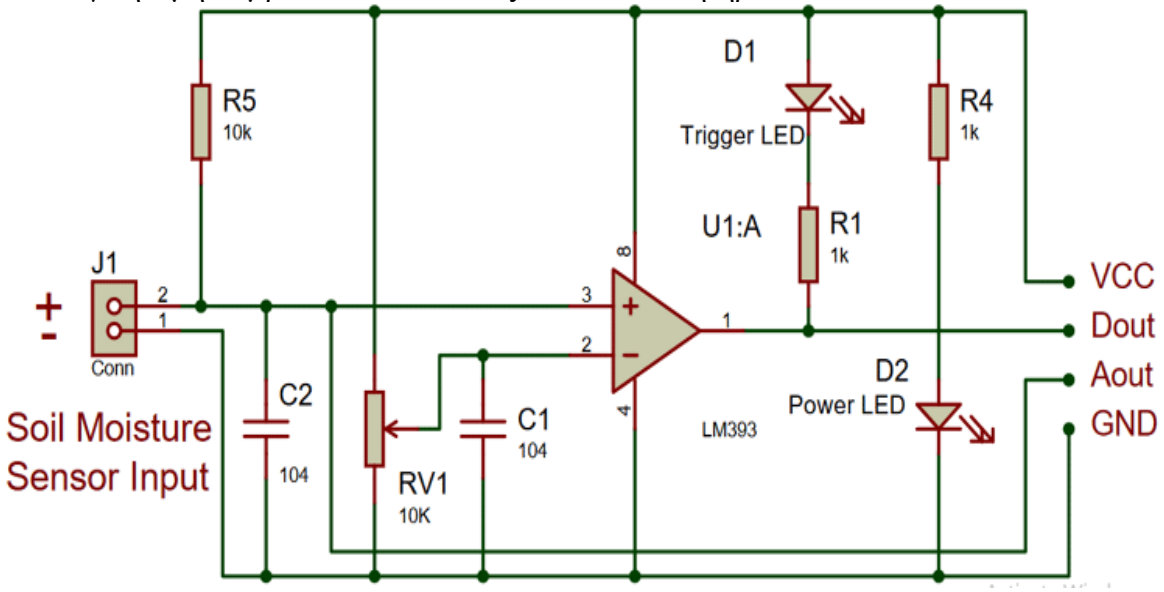
- Μέγιστη ισχύς 50W
- Μέγιστο ρεύμα 2.75A
- Μέγιστη τάση εξόδου 18.2V
- Τάση ανοιχτού κυκλώματος 21.5V
- Ρεύμα βραχυκύκλωσης 3.23A
- Διαστάσεις 670mm*540mm*30mm
- Θερμοκρασίες λειτουργείας -40^{o}C - +85^{o}C

3.11 ΑΙΣΘΗΤΗΡΑΣ ΥΓΡΑΑΣΙΑΣ ΕΔΑΦΟΥΣ

Η διαδικασία της κομποστοποίησης για να είναι επιτυχής χρειάζεται νερό για να υπάρχει η απαραίτητη υγρασία. Για αυτό το λόγο χρειάζεται μια δεξαμενή με νερό, έτσι για να μπορεί το σύστημα να ελέγχει την γενική στάθμη νερού που έχει διαθέσιμο τοποθετήθηκαν αισθητήρες υγρασίας.

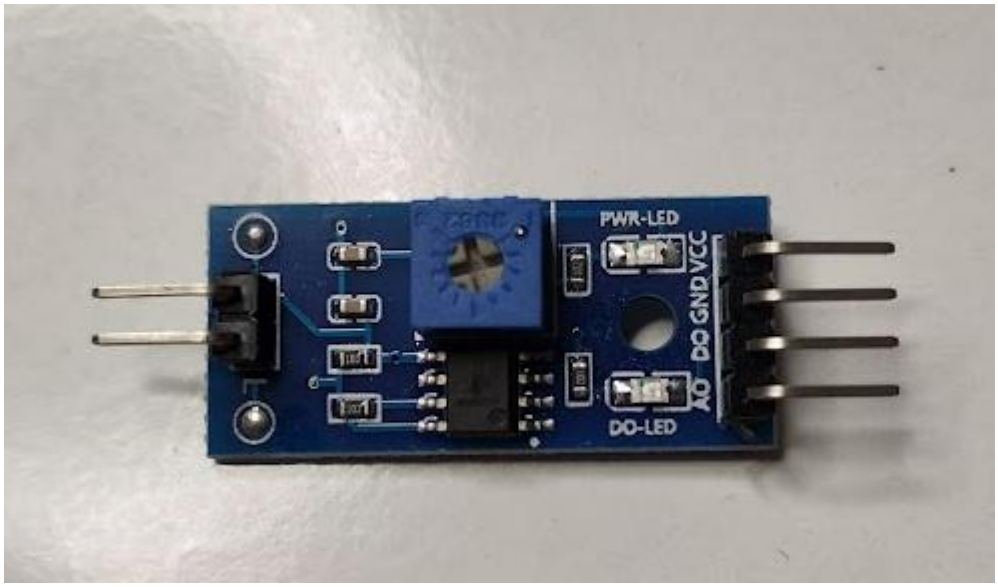
Ο αισθητήρας είναι ουσιαστικά δυο ακίδες που τοποθετούνται μέσα στο έδαφος που θέλουμε να μετρήσουμε την υγρασία, έτσι όσο περισσότερη υγρασία έχει το έδαφος τόσο πιο αγώγιμο γίνεται και περνάει περισσότερο ρεύμα και αντίστροφα. Ο αισθητήρας που χρησιμοποιήθηκε είναι ο FC-28.

Στη συνέχεια ο αισθητήρας συνδέεται σε έναν συγκριτή τάσεων, τον LM393 για να λειτουργήσει η ψηφιακή έξοδος του συστήματος.
Στην πλακέτα στην οποία είναι τοποθετημένος ένας ροοστάτης για τη ρύθμιση του ορίου ώστε να λειτουργήσει η ψηφιακή έξοδος σε επιθυμητή τιμή υγρασίας. Υπάρχει ένα LED για ένδειξη ότι λειτουργεί και ένα ακόμη LED για ένδειξη ότι έχει ξεπεραστεί το όριο για λαμβάνουμε στην ψηφιακή έξοδο λογικό 1.
Η αναλογική τιμή λαμβάνεται απ' ευθείας από τον αισθητήρα.



Σχήμα 3: Σχηματικό της πλακέτας του LM393. (https://components101.com/modules/soil-moisture-sensor-module)

Εικόνα 41: Οι ακίδες για την μέτρηση της υγρασίας.



Εικόνα 42: Η πλακέτα και ο συγκριτής LM393.

ΚΕΦΑΛΑΙΟ 4 : ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΚΑΙ ΣΥΣΤΗΜΑ

4.1 ESP 32

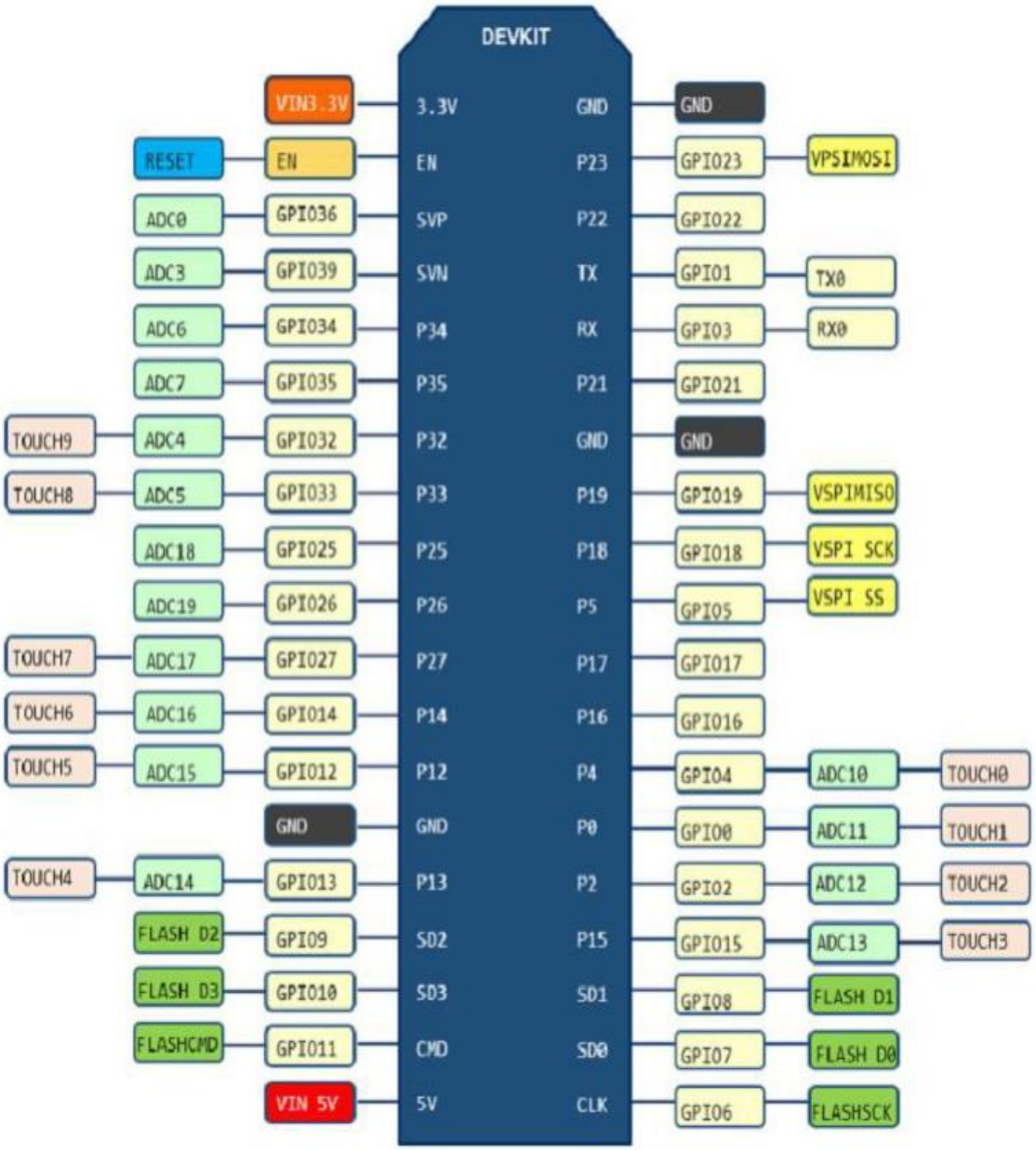
Η επεξεργαστική μονάδα αποτελεί την πρωταρχική συνιστώσα ενός αισθητήριου συστήματος. Για την ορθή της επιλογή πρέπει να ληφθούν αρκετές παράμετροι υπόψη όπως το κόστος, η κατανάλωση, η απόδοση, η απλότητα στη σύνδεση με τα υπόλοιπα συστήματα.
Ως επεξεργαστική μονάδα επιλέχτηκε η αναπτυξιακή πλακέτα NodeMCU-32S της εταιρίας Shenzhen Ai-Thinker. Η συγκεκριμένη πλακέτα είναι συμβατή με την πλατφόρμα Arduino και πολυχρησιμοποιημένη από την κοινότητα, πράγμα το οποίο διευκολύνει τον προγραμματισμό της και την ενσωμάτωση περιφερειακών σε αυτήν. Το τσιπ ESP32 περιλαμβάνει τον διπύρηνο 32-bit μικροεπεξεργαστή LX6 της εταιρίας Tensilica Xtensa, ο οποίος χρονίζεται από τα 80 στα 240MHz. Επίσης, διαθέτει 520KB SRAM, Wi-Fi πομποδέκτη 802.11b/g/n HT40 ο οποίος επιτρέπει τη δημιουργία Access Point και, τέλος, υποστηρίζει και Bluetooth 4.2 (BR/EDR/BLE).
Υποστηρίζει και τα ακόλουθα πρωτόκολλα επικοινωνίας UART, SPI, SDIO, I2C, I2S και IR.
Έχει ενσωματωμένους ADC και DAC.
Ο μετατροπέας αναλογικού σήματος σε ψηφιακό έχει 12 bit ακρίβεια και μπορεί να διαχειριστεί μέχρι 18 κανάλια.
Η θερμοκρασία λειτουργείας θα πρέπει να είναι μεταξύ -20 και +70^{o}C.
Οι τάση τροφοδοσίας θα πρέπει να είναι 5V από την θύρα USB της πλακέτας η οποία μειώνεται από την πλακέτα με έναν ρυθμιστή τάσης στα 3,3V, είτε μπορεί να τροφοδοτηθεί απ’ ευθείας με 3,3V.

Οι πόρτες εισόδων εξόδων του ESP32 είναι προγραμματιζόμενες δηλαδή μπορεί μέσω του κώδικα να καθορίσει το τί θα είναι η κάθε πόρτα εκτός αν χρειάζεται να εκτελεί κάποια εξειδικευμένη λειτουργεία δηλαδή σε περίπτωση που ο χρήστης θέλει να διαβάσει μια αναλογική είσοδο τότε θα πρέπει να χρησιμοποιήσει τον ADC του ESP άρα θα πρέπει να συνδέσει την είσοδο αυτή σε μια πόρτα που να υποστηρίζει ο ADC από ADC0 – ACD19.

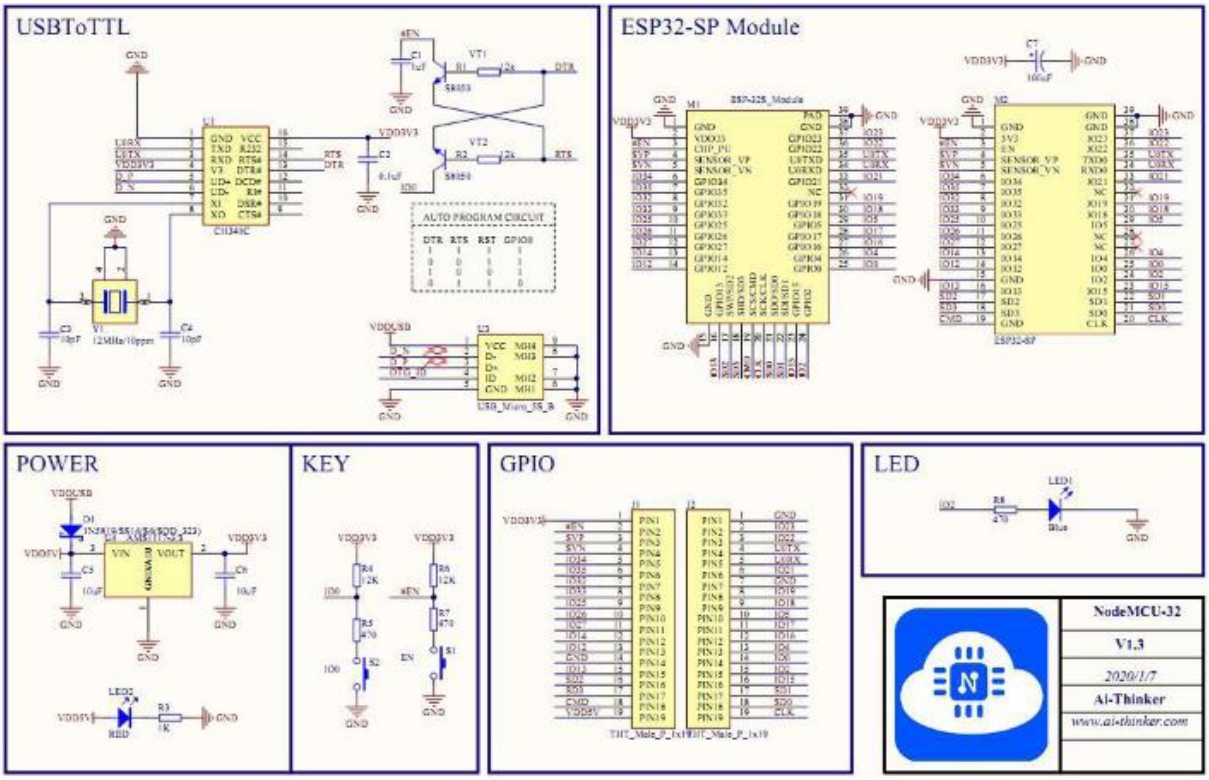
Για τον προγραμματισμό του ESP και επικοινωνία του με υπολογιστή υπάρχει στην είσοδο της USB το CH340C που είναι μετατροπέας από USB σε UART. Το UART πρωτόκολλο περιλαμβάνει 2 αγωγούς RX και TX για ανάγνωση και μετάδοση αντίστοιχα, είναι full

duplex επικοινωνία, χρειάζεται οι συσκευές που επικοινωνούν να έχουν κοινή γείωση για ρύθμιση των τάσεων, θα πρέπει να έχουν συγχρονιστεί να εκπέμπουν και να λαμβάνουν στους ίδιους χρόνους baud rate (bits/sec) καθώς και να χρησιμοποιούν ίδια bit ισοτιμίας για έλεγχο σφαλμάτων.

Τα πλεονεκτήματα της συγκεκριμένης επικοινωνίας είναι ότι είναι πολύ απλή διότι δεν χρειάζεται να δοθούν διευθύνσεις , είναι αμφίδρομη επικοινωνία, χρειάζονται μόνο 3 αγωγοί και μπορούν να είναι σε απόσταση έως και 1 χιλιόμετρο.
Τα αρνητικά είναι ότι μπορούν να συνδεθούν μόνο 2 συσκευές, ο ρυθμός μετάδοσης είναι σταθερός και ότι σε σχέση με άλλες επικοινωνίες όπως I2C και SPIείναι πιο αργή.



Εικόνα 43: Σχέδιο με τις εισόδους-εξόδους του NodeMCU-32.(https://docs.ai-thinker.com/_media/nodemcu32-s_specification_v1.3.pdf)



Εικόνα 44: Σχηματικά από τα κυκλώματα της πλακέτας.

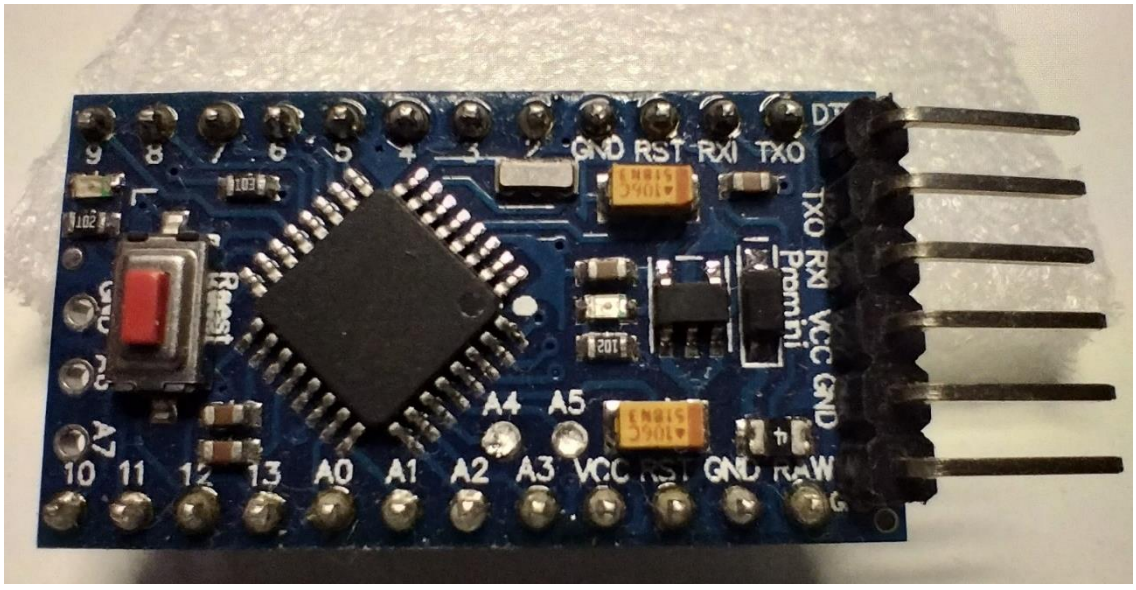
4.2 ARDUINO PRO MINI

Είναι μια αναπτυξιακή πλακέτα με βάση τον μικροεπεξεργαστή της Microchip (πρώην Atmel) ATMEGA328P.
Υπάρχουν δυο εκδόσεις η μια είναι χρονισμένη στα 8MHz και τροφοδοτείται στα 3.3V και η άλλη στα 16MHz και 5V. Έχει 32kB flash μνήμης, 2kB SRAM και
1kB EEPROM η οποία μπορεί να χρησιμοποιηθεί μόνο όταν προγραμματιστικά χρησιμοποιείται η βιβλιοθήκη την EEPROM library.

Από εισόδους και εξόδους έχει 14 ψηφιακές εισόδους εξόδους, 6 αναλογικές εισόδους, ταλαντωτή και κουμπί επαναφοράς. Επίσης έχει τα RX και TX για επικοινωνία σε TTL, έχει 6 εξόδους PWM με ανάλυση 8 bit, υποστηρίζει SPI επικοινωνία με τα pins SCLK το ρολόι του διαύλου, MOSI που είναι η έξοδος δεδομένων από τον ‘’κύριο’’ και είσοδος στον ‘’δούλο’’, MISO που είναι η έξοδος δεδομένων από τον ‘’ δούλο’’ και είσοδος στον ‘’ κύριο’’ και το CS που χρησιμοποιείται για την επιλογή του ‘’δούλου’’ για να επικοινωνήσει ο ‘’κύριος’’ και τέλος έχει τα pin A4 ως SDA και A5 ως SCL για επικοινωνία I2C.

Microcontroller	ATmega328P *
Board Power Supply	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
Circuit Operating Voltage	3.3V or 5V (depending on model)
Digital I/O Pins	14
PWM Pins	6
UART	1
SPI	1
I2C	1
Analog Input Pins	6
External Interrupts	2
DC Current per I/O Pin	40 mA
Flash Memory	32KB of which 2 KB used by bootloader *
SRAM	2 KB *
EEPROM	1 KB *
Clock Speed	8 MHz (3.3V versions) or 16 MHz (5V versions)

Πίνακας 10: Πίνακας με τα χαρακτηριστικά της αναπτυξιακής πλακέτας. (https://docs.arduino.cc/retired/boards/arduino-pro-mini)



Εικόνα 45: Η αναπτυξιακή πλακέτα Arduino pro mini.

Για την επικοινωνία του Arduino χρειάζεται να συνδεθεί ένας μετατροπέας από USB σε UART πρωτόκολλο.
Για την κατασκευή χρησιμοποιήθηκαν τα τσιπ της FTDI FT232RL που παρέχει ρύθμιση στο ρολόι του για να μπορεί να οδηγήσει μικροελεγκτές, μπορεί να στείλει 7 ή 8 bit δεδομένων, 1 ή 2 stop bits και μπορεί να γίνει επιλογή bit ισοτιμίας σε άρτια, περιττή, mark, space και χωρίς ισοτιμία.

- Υποστηρίζει baud rate από 300 έως 3Megabaud,
- Έχει 256byte μνήμη εισόδου και 128 byte μνήμη εξόδου για μεταφορά μεγάλου όγκου δεδομένων
- Επιλογή τάσης εξόδου στα 1.8V, 2.8V, 3.3V και στα 5V
- Είναι χαμηλής ισχύος
- Χαμηλής χρήσης εύρους ζώνης από την USB θύρα
- Μπορεί να αντιστρέψει τα σήματα στην έξοδο UART.
- Η θερμοκρασία λειτουργείας είναι από -40°C έως 85°C

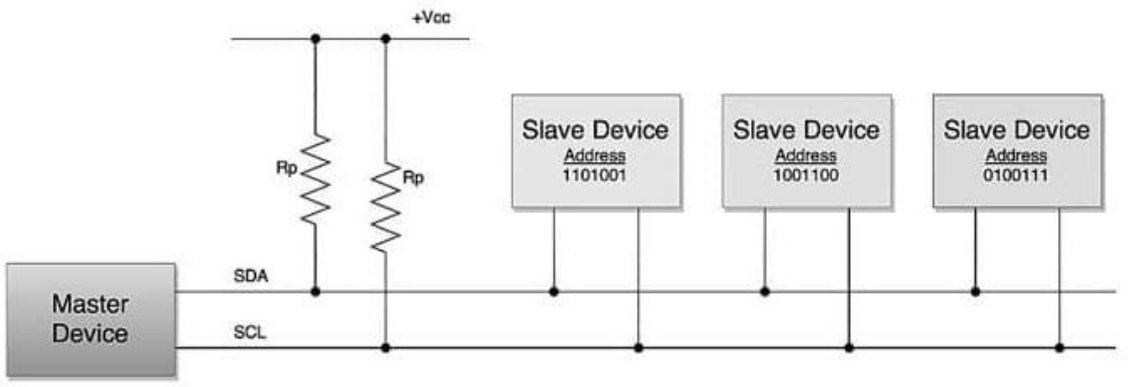
Εικόνα 46: Το FT232RL. (https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/modules/converter/ftdi-ft232rl-usb-to-ttl-serial-converter-adapter-module-5v-and-3.3v-for-arduino/)

4.4 ΛΕΙΤΟΥΡΓΕΙΑ I2C

Το πρωτόκολλο επικοινωνίας I2C, αναπτύχθηκε από την Philips το 1982. Αυτό το πρωτόκολλο επιτρέπει τη μεταφορά δεδομένων μεταξύ ενός κεντρικού επεξεργαστή και πολλαπλών ολοκληρωμένων κυκλωμάτων στην ίδια πλακέτα, χρησιμοποιώντας μόνο δύο κοινά καλώδια. Είναι ευρέως υιοθετημένο για τη σειριακή επικοινωνία συγχρονισμένης μικρής απόστασης μεταξύ μικροελεγκτών, πινάκων αισθητήρων, οθονών, συσκευών IoT (Internet of Things), EEPROMs κ.λπ. Ο πολλών "Master" I2C δίαυλος αποτελείται από δύο ενεργά καλώδια (Serial Data/SDA και Serial Clock/SCL) και τα δεδομένα μεταφέρονται bit by bit κατά μήκος ενός μόνου καλωδίου.
Ο σχεδιασμός I2C έχει χωρητικότητα διεύθυνσης 7 bit και χρησιμοποιείται κυρίως σε λειτουργίες ταχύτητας διαύλου, όπως η τυπική λειτουργία (100 kbit/s) και η γρήγορη λειτουργία (400 kbit/s). Υπάρχει μια λειτουργία χαμηλής ταχύτητας (10 kbit/s), αλλά γίνεται αποδεκτή οποιαδήποτε χαμηλή συχνότητα ρολογιού, και οι τελευταίες αναβαθμίσεις προσφέρουν ταχύτητες έως 5 Mbit/s. Ο χωρητικότητα των διευθύνσεων και ο περιορισμός της συνολικής χωρητικότητας του διαύλου περιορίζουν τον αριθμό των κόμβων που μπορούν να υπάρχουν σε έναν δίαυλο I2C και περιορίζουν επίσης την απόσταση επικοινωνίας σε λίγα μέτρα.
Τα δεδομένα που χρειάζεται να μεταφερθούν αποστέλλονται στο καλώδιο Serial Data Line (SDA) και συγχρονίζονται με το σήμα ρολογιού από το καλώδιο Serial Clock Line (SCL).

Οι συσκευές που συνδέονται στο δίκτυο I2C ονομάζονται "Masters" ή " Slaves ". Ένας μόνο διαχειριστής παραμένει ενεργός στο δίκτυο I2C σε οποιαδήποτε στιγμή. Ο διαχειριστής ελέγχει τη γραμμή ρολογιού SCL και τις γραμμές SDA και επικοινωνεί με τις συσκευές σκλάβους.
Κατά τη μεταφορά δεδομένων προς ή από μια συσκευή "Slave", ο διαχειριστής καθορίζει μια 7-bit διεύθυνση που αντιστοιχεί στη συσκευή σκλάβο στη γραμμή SDA και στη συνέχεια πραγματοποιεί τη μεταφορά δεδομένων.

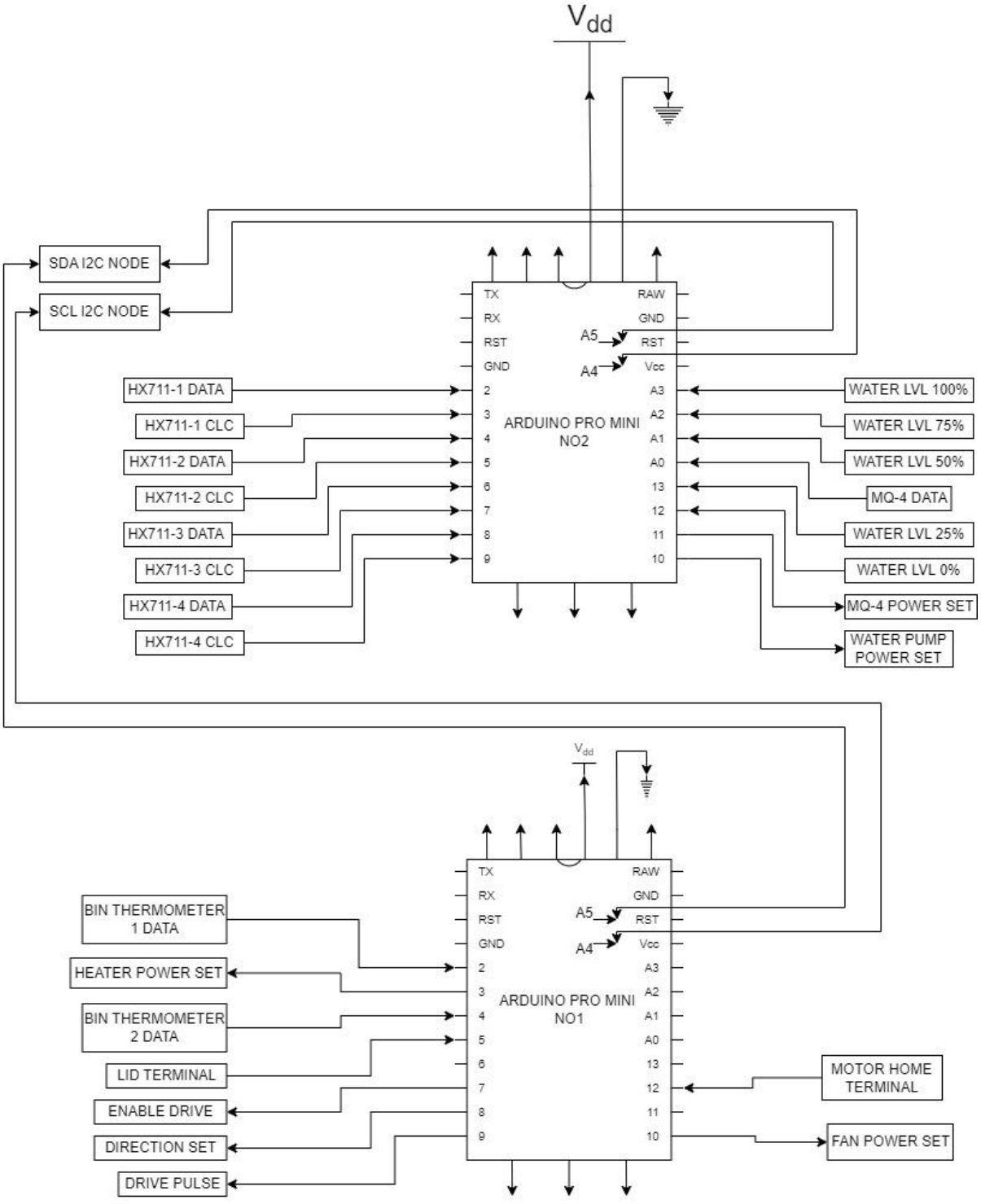
Η μεταφορά δεδομένων γίνεται σε μηνύματα, τα οποία διαιρούνται σε πλαίσια δεδομένων. Το μήνυμα περιλαμβάνει πλαίσιο διεύθυνσης που περιέχει τη δυαδική διεύθυνση του σκλάβου, συνθήκες έναρξης/τερματισμού, δυαδικά bits για ανάγνωση/εγγραφή και ACK/NACK (απάντηση αποδοχής/απόρριψης) ανάμεσα σε κάθε πλαίσιο δεδομένων.



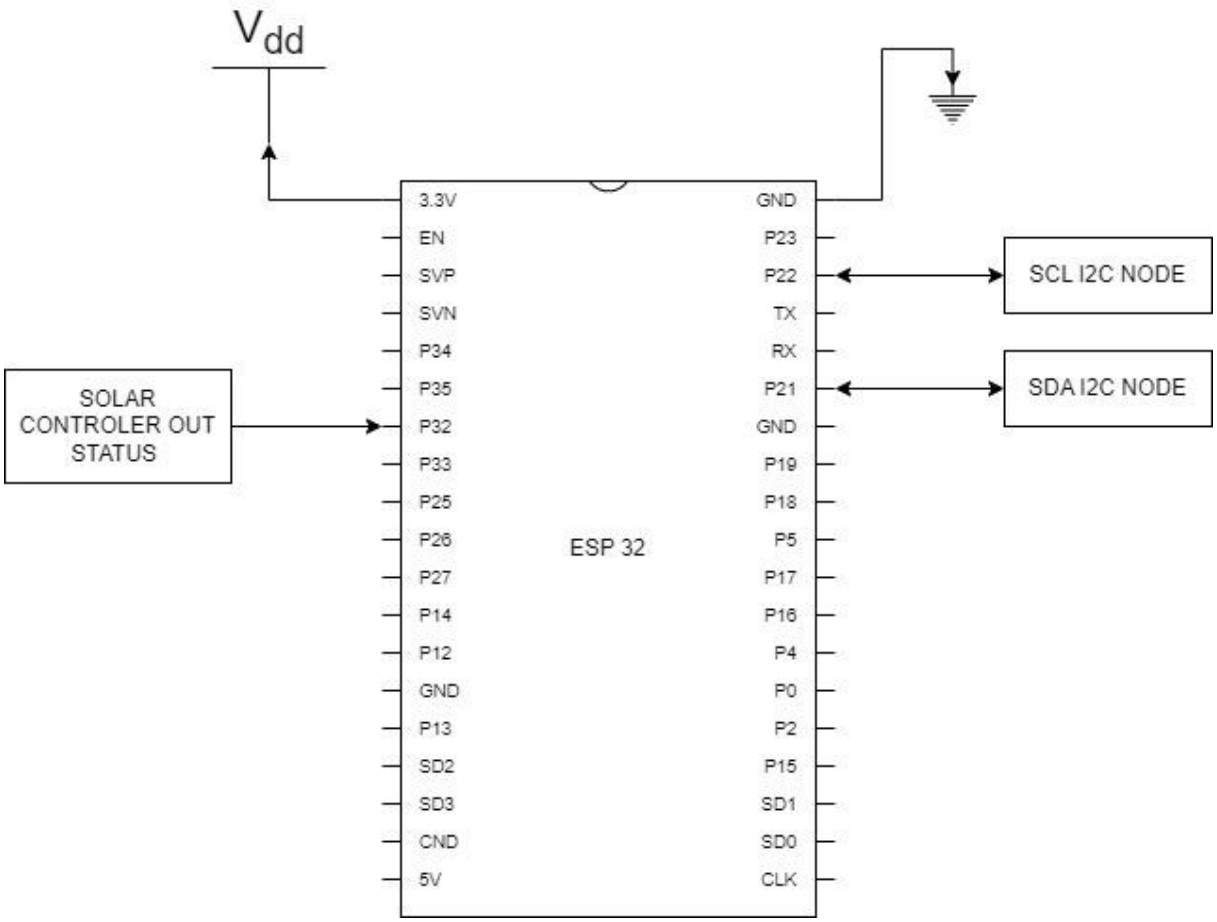
Εικόνα 47: Συνδεσμολογία του διαύλου I2C. (https://911electronic.com/how-i2c-works-i2c-protocol/)

4.3 ΔΙΑΓΡΑΜΜΑΤΑ ΣΥΝΔΕΣΕΩΝ

Παρακάτω παραθέτονται τα σχηματικά διαγράμματα συνδέσεων των ARDUINO και του ESP32.



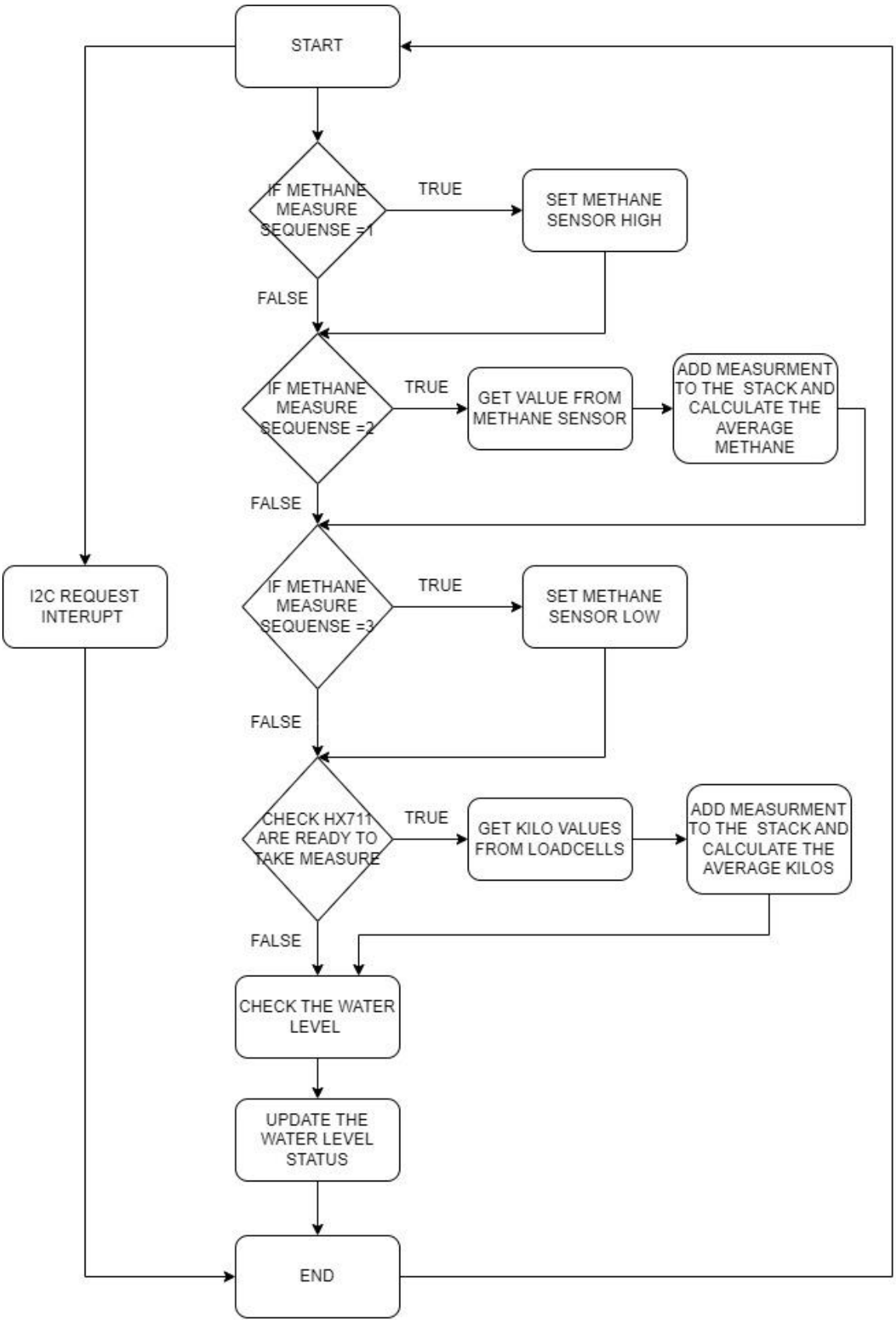
Σχήμα 4: Διασύνδεση των Arduino pro mini.



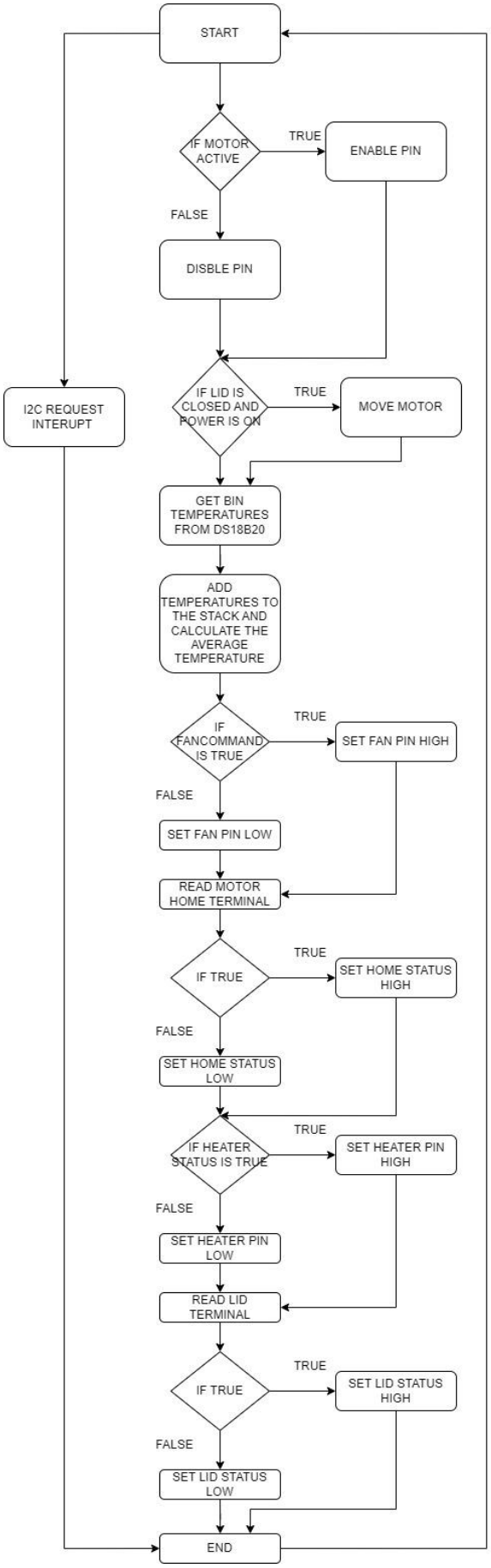
Σχήμα 5: διασύνδεση του ESP32.

ΚΕΦΑΛΑΙΟ 5 : ΚΩΔΙΚΕΣ ΚΑΙ ΔΙΑΓΡΑΜΑΤΑ ΡΟΗΣ

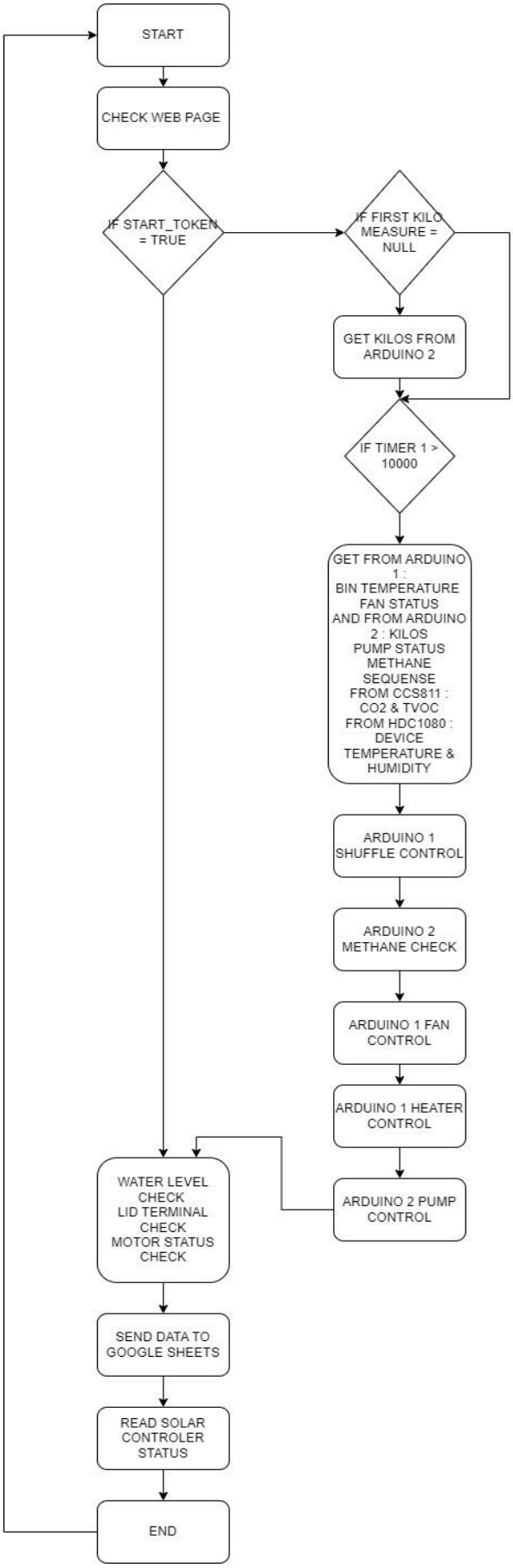
5.1 ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ



Σχήμα 6: Διάγραμμά ροής του Arduino 2.



Σχήμα 7: Διάγραμμά ροής του Arduino 1.



Σχήμα 8: Διάγραμμα ροής του ESP32.

5.2 ΚΩΔΙΚΕΣ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

5.2.1 ESP 32

5.2.1.1 PAGE 1

```
#include <WiFi.h>
#include <Wire.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <HTTPClient.h>
#include <Adafruit_CCS811.h>
#include <ClosedCube_HDC1080.h>
ClosedCube_HDC1080 hdc1080;
Adafruit_CCS811 ccs;


const char* WIFI_NAME= "THOMASNET"; //laptopaggelos
const char* WIFI_PASSWORD = "LGyr1276@!"; //00005555

WebServer server(80);

String status1Text = "Status 1: ...";
String status2Text = "Status 2: ...";
String status3Text = "Status 3: ...";
bool button1Pressed = false;
bool button2Pressed = false;
bool button3Pressed = false;
bool button4Pressed = false;
bool button5Pressed = false;

bool Start_token=false
,Pause_token=false,Empty_bin_token=false,Stop_token=true;

String header;
const String event = "esp32datatransfer" ;
unsigned long sendtoifttt =0;

const int MPPV_OUT = 32;

const int ARDUINO1_ADDR = 8;
const int ARDUINO2_ADDR = 9;

int state = 0;
int duration = 0;
```

```
unsigned long lasttime =0, methanetime=0, shufle_time=0,
checkevent=0,mppvevent=0;
float
Kilos,Bin_temperature,methane_measure,CO2,TVOC,DEVICE_TEMPERATURE,DEVICE_HUMID
ITY, first_kilos_measure;
short Water_level_status, methane_sensor_status/*1,2,3 */,Motor_status;
bool MPPV=false, Lid_status, Fan_status, Pump_status, Heater_status;


const byte COMMAND_MOTOR_STATUS = 11;
const byte COMMAND_BIN_TEMPERATURE = 12;
const byte COMMAND_LID_TERMINAL_STATUS = 13;
const byte COMMAND_FAN_STATUS = 15;
const byte COMMAND_HEATER_STATUS =16;

const byte COMMAND_KILOS = 21;
const byte COMMAND_WATER_LEVEL_CHECK = 22;
const byte COMMAND_PUMP_STATUS = 23;
const byte METHANE_MEASURE_SEQUENCE = 24;
const byte METHANE_MEASURE = 25;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Wire.begin(21,22);//sda,scl,adress

  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.on("/command", handleCommand);

  server.begin();
  Serial.println("HTTP server started");

  //mppv digital output status
  pinMode(MPPV_OUT,INPUT);

  if (!ccs.begin())
  {
    Serial.println("Failed to find CCS811 sensor!"); //CCS811 initialization
fails
    while(1);
```

```
  }
   while(!ccs.available());

  hdc1080.begin(0x40);

  Serial.println("Startup finished ");

}

void loop() {
  server.handleClient();
  handleCommand();
  StatusControl();
  if(Start_token==true)
    schedule();

  if(checkdelay(10000,checkevent) == true)
  {
    requestValueFromArduino2(COMMAND_WATER_LEVEL_CHECK);
    receiveintValueFromArduino2(COMMAND_WATER_LEVEL_CHECK);
    //Serial.print("Water_level_status  :");
    //Serial.println(Water_level_status);

    requestValueFromArduino1(COMMAND_LID_TERMINAL_STATUS);
    receiveboolValueFromArduino1(COMMAND_LID_TERMINAL_STATUS);
    //Serial.print("Lid_status  :");
    //Serial.println(Lid_status);

    requestValueFromArduino1(COMMAND_MOTOR_STATUS);
    receiveintValueFromArduino1(COMMAND_MOTOR_STATUS);
    //Serial.print("Motor_status  :");
    //Serial.println(Motor_status);
    checkevent = millis();
  }
  if((checkdelay(300000,sendtoifttt) == true)&&(Start_token == true))//Send
variables to google sheets
  {
    String url = "https://maker.ifttt.com/trigger/" + event +
"/with/key/eh1tjjNAJd34f0cEfQdqv4FF5Xn9Z0ZPWsXRUHxcc0h";
    url += "?value1=" + String(Kilos);
    url += "&value2=" + String(DEVICE_HUMIDITY);
    url += "&value3=" + String(Bin_temperature);
    //url += "&value4=" + String(CO2);
    //url += "&value5=" + String(TVOC);
    //url += "&value6=" + String(DEVICE_TEMPERATURE);
    HTTPClient http;
    http.begin(url);
```

```
    int httpResponseCode = http.GET();//Verification that data was sent
    if (httpResponseCode == HTTP_CODE_OK)
      Serial.println("Data sent successfully");
    else
      Serial.println("Error sending data");
    http.end();
    sendtoifttt = millis();
  }

  ////////mppv status
  if(((digitalRead(MPPV_OUT)==HIGH) && (MPPV == false)) ||( (MPPV==true) &&
(checkdelay(10000,mppvevent) == true)))
  {
    MPPV=true;
    sendInstructionToArduino1(9);//send to arduino 1 that mppv out is on
    sendInstructionToArduino2(9);//send to arduino 2 that mppv out is on

  }
  if((digitalRead(MPPV_OUT)==LOW) && (MPPV == true)|| ( (MPPV==false) &&
(checkdelay(10000,mppvevent) == true)))
  {
    MPPV = false;
    sendInstructionToArduino1(17);//send to arduino that 1 mppv out is off
    sendInstructionToArduino2(17);//send to arduino that 2 mppv out is off

  }
  if((Empty_bin_token==true)&&(Motor_status == 1))
  {
    sendInstructionToArduino1(2);//move to empty position
    requestValueFromArduino1(COMMAND_MOTOR_STATUS);
    receiveintValueFromArduino1(COMMAND_MOTOR_STATUS);
  }
  if((Empty_bin_token==false)&&(Motor_status == 2))
  {
    sendInstructionToArduino1(1);//move to home position
    requestValueFromArduino1(COMMAND_MOTOR_STATUS);
    receiveintValueFromArduino1(COMMAND_MOTOR_STATUS);
  }
  if(Stop_token==true)
  {
    first_kilos_measure=0;
    sendInstructionToArduino2(5);//deactivate pump
    requestValueFromArduino2(COMMAND_PUMP_STATUS);
    receiveboolValueFromArduino2(COMMAND_PUMP_STATUS);

    sendInstructionToArduino1(8);//8 to deactivate heater
    requestValueFromArduino1(COMMAND_HEATER_STATUS);
    receiveboolValueFromArduino1(COMMAND_HEATER_STATUS);
```

```
    sendInstructionToArduino1(6);//5 to activate fan and 6 to deactivate fan
    requestValueFromArduino1(COMMAND_FAN_STATUS);
    receiveboolValueFromArduino1(COMMAND_FAN_STATUS);

    sendInstructionToArduino2(3);//deactiveate methane sensor
    requestValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    receiveintValueFromArduino2(METHANE_MEASURE_SEQUENCE);
  }
}
```

5.2.1.2 PAGE 2

```
void schedule()
{
  if(first_kilos_measure==NULL||first_kilos_measure==0)
  {
    requestValueFromArduino2(COMMAND_KILOS);
    receivefloatValueFromArduino2(COMMAND_KILOS);
    first_kilos_measure=Kilos;
    Serial.println(Kilos);
  }
  if(checkdelay(10000,lasttime) == true)
  {
    // Request specific values from Arduino 1

    requestValueFromArduino1(COMMAND_BIN_TEMPERATURE);
    receivefloatValueFromArduino1(COMMAND_BIN_TEMPERATURE);

    requestValueFromArduino1(COMMAND_FAN_STATUS);
    receiveboolValueFromArduino1(COMMAND_FAN_STATUS);
    Serial.print("Fan_status  :");
    Serial.println(Fan_status);


    // Request specific values from Arduino 2
    requestValueFromArduino2(COMMAND_KILOS);
    receivefloatValueFromArduino2(COMMAND_KILOS);

    requestValueFromArduino2(COMMAND_PUMP_STATUS);
    receiveboolValueFromArduino2(COMMAND_PUMP_STATUS);

    requestValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    receiveintValueFromArduino2(METHANE_MEASURE_SEQUENCE);


    //Request values from CCS811/HDC1080
    CO2 = readCCS811CO2();
    Serial.print("CO2 :");
    Serial.println(CO2);
    TVOC = readCCS811TVOC();
    Serial.print("TVOC :");
    Serial.println(TVOC);
    DEVICE_TEMPERATURE = hdc1080.readTemperature();
    Serial.print("DEVICE_TEMPERATURE :");
    Serial.println(DEVICE_TEMPERATURE);
    DEVICE_HUMIDITY = hdc1080.readHumidity();
    Serial.print("DEVICE_HUMIDITY :");
```

```
    Serial.println(DEVICE_HUMIDITY);

    lasttime = millis();
  }
  if( ((checkdelay(180000,shufle_time) == true) || (methane_measure > 800)) &&
(Motor_status == 1)&&(MPPV==true)&&(Pause_token == false) )
  {
    sendInstructionToArduino1(3);//begin shufle
    requestValueFromArduino1(COMMAND_MOTOR_STATUS);
    receiveintValueFromArduino1(COMMAND_MOTOR_STATUS);
    shufle_time = millis();
  }
//////////////////////////methane measure
  if( ((checkdelay(180000,methanetime) == true && methane_sensor_status == 3)
|| (methane_measure>800 && checkdelay(10000,methanetime) == true) )&&
(MPPV==true) )
  {
    sendInstructionToArduino2(1); //begin the methane measure sequence
    requestValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    receiveintValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    methanetime = millis();
  }
  if((checkdelay(9000,methanetime)== true) && (methane_sensor_status == 1) )
  {
    sendInstructionToArduino2(2);//take measyre avrg
    requestValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    receiveintValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    methanetime = millis();
  }
  if((checkdelay(9000,methanetime)== true) && (methane_sensor_status == 2 ))
  {
    requestValueFromArduino2(METHANE_MEASURE);
    receivefloatValueFromArduino2(METHANE_MEASURE);
    sendInstructionToArduino2(3);//deactiveate methane sensor
    requestValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    receiveintValueFromArduino2(METHANE_MEASURE_SEQUENCE);
    methanetime = millis();
  }
////////////////////////////////////////////////////
  //fan control
  if(
((methane_measure>800)||(Bin_temperature>34)||(CO2>500)||(DEVICE_TEMPERATURE>3
4)||(DEVICE_HUMIDITY>90))&&(MPPV==true) )//over 800ppm//over 34 celcius// over
...ppm co2
  {
    sendInstructionToArduino1(5);//5 to activate fan and 6 to deactivate fan
    requestValueFromArduino1(COMMAND_FAN_STATUS);
    receiveboolValueFromArduino1(COMMAND_FAN_STATUS);
```

```
  }
  if(((methane_measure<800)&&(Bin_temperature<35)&&(CO2<500)&&(DEVICE_TEMPERAT
URE<38)&&(DEVICE_HUMIDITY<80))||(MPPV==false))//over 800ppm//over 34 celcius//
over ...ppm co2
  {
    sendInstructionToArduino1(6);//5 to activate fan and 6 to deactivate fan
    requestValueFromArduino1(COMMAND_FAN_STATUS);
    receiveboolValueFromArduino1(COMMAND_FAN_STATUS);
  }
  //heater control
  if((Bin_temperature<30)&&(MPPV==true))//temperature in the bin less than 30
  {
    sendInstructionToArduino1(7);//7 to activate heater and 8 to deactivate
    requestValueFromArduino1(COMMAND_HEATER_STATUS);
    receiveboolValueFromArduino1(COMMAND_HEATER_STATUS);
  }
  if((Bin_temperature>30)||(MPPV==false))
  {
    sendInstructionToArduino1(8);//8 to deactivate heater
    requestValueFromArduino1(COMMAND_HEATER_STATUS);
    receiveboolValueFromArduino1(COMMAND_HEATER_STATUS);
  }
  //humidity control - watering control
  if(((first_kilos_measure<0.1*Kilos)&&(Water_level_status>1))&&(MPPV==true))
  {
    sendInstructionToArduino2(4);//activate pump
    requestValueFromArduino2(COMMAND_PUMP_STATUS);
    receiveboolValueFromArduino2(COMMAND_PUMP_STATUS);
    if(Motor_status == 1)
    {
      sendInstructionToArduino1(4);
      requestValueFromArduino1(COMMAND_MOTOR_STATUS);
      receiveintValueFromArduino1(COMMAND_MOTOR_STATUS);
    }
  }
  if(((first_kilos_measure>0.1*Kilos)||(Water_level_status<2))||(MPPV==false))
  {
    sendInstructionToArduino2(5);//deactivate pump
    requestValueFromArduino2(COMMAND_PUMP_STATUS);
    receiveboolValueFromArduino2(COMMAND_PUMP_STATUS);
  }
}
```

5.2.1.3 PAGE 3

```
// Function to send an instruction to Arduino 1
void sendInstructionToArduino1(int instruction) {
  Wire.beginTransmission(ARDUINO1_ADDR);
  Wire.write(instruction);
  Wire.write((byte*)&instruction, sizeof(int));
  Wire.endTransmission();
}
//--------------------------------------------------------------------------
--------------------------------------------------------------------------//

// Function to send an instruction to Arduino 2
void sendInstructionToArduino2(int instruction) {
  Wire.beginTransmission(ARDUINO2_ADDR);
  Wire.write(instruction);
  Wire.write((byte*)&instruction, sizeof(int));
  Wire.endTransmission();
}

//--------------------------------------------------------------------------
--------------------------------------------------------------------------//

// Function to request a specific value from Arduino 1
void requestValueFromArduino1(int command) {
  Wire.beginTransmission(ARDUINO1_ADDR);
  Wire.write(command);
  Wire.endTransmission();
}
//--------------------------------------------------------------------------
--------------------------------------------------------------------------//
// Function to request a specific value from Arduino 2
void requestValueFromArduino2(int command) {
  Wire.beginTransmission(ARDUINO2_ADDR);
  Wire.write(command);
  Wire.endTransmission();
}
//--------------------------------------------------------------------------
--------------------------------------------------------------------------//
void receivefloatValueFromArduino2(byte command) {
  Wire.requestFrom(ARDUINO2_ADDR, sizeof(float)); // Request a single float
value from Arduino 2

  while (Wire.available()) {
    float value = 0.0;
    Wire.readBytes((byte*)&value, sizeof(float));
```

```
    // Process the received value from Arduino 2 based on the command
    switch (command) {
      case COMMAND_KILOS:
        Kilos = value;
        Serial.println(Kilos);
        // Process kilos value
        //21
      break;
      case METHANE_MEASURE:
        methane_measure = value;
        //methane value obtain
        //25
      break;
    }
  }
}
void receiveintValueFromArduino2(byte command) {
  Wire.requestFrom(ARDUINO2_ADDR, sizeof(short)); // Request a single float
value from Arduino 2

  while (Wire.available()) {
    short value = 0;
    Wire.readBytes((byte*)&value, sizeof(short));

    // Process the received value from Arduino 2 based on the command
    switch (command) {
      case COMMAND_WATER_LEVEL_CHECK:
        //1 for crit , 2 for 0, 3 for 25 , 4 for 50, 5 for 75 , 6 for 100
        Water_level_status = value;
        //22
      break;
      case METHANE_MEASURE_SEQUENCE:
        methane_sensor_status = value;
        //1 active seq, 2 measuring, 3 inactive
        //24
      break;
    }
  }
}
void receiveboolValueFromArduino2(byte command) {
  Wire.requestFrom(ARDUINO2_ADDR, sizeof(bool)); // Request a single float
value from Arduino 2

  while (Wire.available()) {
    bool value = false;
    Wire.readBytes((byte*)&value, sizeof(bool));
```

```
    // Process the received value from Arduino 2 based on the command
    switch (command) {
      case COMMAND_PUMP_STATUS:
        Pump_status = value;
        //0 inactive , 1 active
        //23
      break;
    }
  }
}
//-------------------------------------------------------------------------------
-------------------------------------------------------------------------------//


void receivefloatValueFromArduino1(byte command) {
  Wire.requestFrom(ARDUINO1_ADDR, sizeof(float)); // Request a single float
value from Arduino 1

  while (Wire.available()) {
    float value = 0.0;
    Wire.readBytes((byte*)&value, sizeof(float));

    // Process the received value from Arduino 1 based on the command
    switch (command) {
      case COMMAND_BIN_TEMPERATURE:
        Bin_temperature = value;
        Serial.println(Bin_temperature);
        // Process temperature value
        //12
        break;

    }
  }
}
void receiveintValueFromArduino1(byte command) {
  Wire.requestFrom(ARDUINO1_ADDR, sizeof(short)); // Request a single int
value from Arduino 1

  while (Wire.available()) {
    short value = 0;
    Wire.readBytes((byte*)&value, sizeof(short));

    // Process the received value from Arduino 1 based on the command
    switch (command) {
      case COMMAND_MOTOR_STATUS:
        Motor_status = value;
        //Serial.println(Motor_status);
        // Process motor status value..... integer : 1 for home position , 2
for empty position , 3 for shuffling, 4 watering, 5 moving to position
```

```
        //11
        break;
    }
  }
}
void receiveboolValueFromArduino1(byte command) {
  Wire.requestFrom(ARDUINO1_ADDR, sizeof(bool)); // Request a single float
value from Arduino 1

  while (Wire.available()) {
    bool value = false;
    Wire.readBytes((byte*)&value, sizeof(bool));

    // Process the received value from Arduino 1 based on the command
    switch (command) {
      case COMMAND_LID_TERMINAL_STATUS:
        //0 for closed, 1 for open
        Lid_status = value;
        //13
        break;
      case COMMAND_FAN_STATUS:
        //0 for deactive, 1 for active
        Fan_status = value;
        //15
      case COMMAND_HEATER_STATUS://0 for deactive, 1 for active
        Heater_status = value;
        //16
        break;
    }
  }
}
//-------------------------------------------------------------------------
-------------------------------------------------------------------------//
float readCCS811CO2()
{
  if (ccs.available())
    if (!ccs.readData())
    {
      float co2 = ccs.geteCO2();
      return co2;
    }
}
//-------------------------------------------------------------------//
float readCCS811TVOC()
{
  if (ccs.available())
    if (!ccs.readData())
    {
```

```
      float tvoc = ccs.getTVOC();
      return tvoc;
    }

  //return -1.0;
}
```

5.2.1.4 PAGE 4

```
void handleRoot() {
  String html = "<html><head><title>Compost Control</title>";
  html += "<style>body { display: flex; justify-content: center; align-items:
center; height: 100vh; }</style>";
  html += "</head><body>";
  html += "<div style='text-align: center;'>";
  html += "<h1>Compost Control</h1>";
  html += "<button onclick=\"sendCommand('Button1')\">START</button>";
  html += "<button onclick=\"sendCommand('Button2')\">PAUSE</button>";
  html += "<button onclick=\"sendCommand('Button3')\">STOP</button>";
  html += "<button onclick=\"sendCommand('Button4')\">EMPTY</button>";
  html += "<button onclick=\"sendCommand('Button5')\">HOME</button>";
  html += "<h2><span id=\"status1\">" + status1Text + "</span></h2>";
  html += "<h2><span id=\"status2\">" + status2Text + "</span></h2>";
  html += "<h2><span id=\"status3\">" + status3Text + "</span></h2>";
  html += "</div>";
  html += "<script>function sendCommand(sercommand) {";
  html += "var xhttp = new XMLHttpRequest();";
  html += "xhttp.open('GET', '/serommand?cmd=' + sercommand, true);";
  html += "xhttp.send();}";
  // AJAX code to periodically update the status strings(Asynchronous
JavaScript And XML)
  html += "setInterval(function() {";
  html += "var xhttpStatus = new XMLHttpRequest();";
  html += "xhttpStatus.onreadystatechange = function() {";
  html += "if (this.readyState == 4 && this.status == 200) {";
  html += "var statusData = JSON.parse(this.responseText);";
  html += "document.getElementById('status1').innerText =
statusData.status1;";
  html += "document.getElementById('status2').innerText =
statusData.status2;";
  html += "document.getElementById('status3').innerText =
statusData.status3;";
  html += "}";
  html += "};";
  html += "xhttpStatus.open('GET', '/getStatus', true);";
  html += "xhttpStatus.send();";
  html += "}, 1000);"; // Update status every 1 second
  html += "</script>";
  html += "</body></html>";
  server.send(200, "text/html", html);
}


void handleCommand() {
```

```
  String sercommand = server.arg("cmd");
  if (sercommand == "Button1") { //start
    //button1Pressed = !button1Pressed;
    Start_token = true;
    Pause_token = false;
    Stop_token = false;
    //Serial.println("start pressed");
  } else if (sercommand == "Button2") { //pause
    //button2Pressed = !button2Pressed;
    Pause_token = true;
  } else if (sercommand == "Button3") { //stop
    //button3Pressed = !button3Pressed;
    Start_token = false;
    Pause_token = false;
    Stop_token = true;
  } else if (sercommand == "Button4") { //empty
    //button4Pressed = !button4Pressed;
    Empty_bin_token = true;
  } else if (sercommand == "Button5") { //home
    //button5Pressed = !button5Pressed;
    Empty_bin_token = false;
  }
  StatusControl();
  String json = "{\"status1\":\"" + status1Text + "\",\"status2\":\"" +
status2Text + "\",\"status3\":\"" + status3Text + "\"}";
  server.send(200, "application/json", json);

}

void StatusControl(){
  if ((Start_token == true) && (Pause_token == false)) {
    status1Text = "THE DEVICE HAS STARTED";
  }
  if ((Start_token == true) && (Pause_token == true)) {
    status1Text = "THE DEVICE IS AT PAUSE";
  }
  if (Stop_token==true) {
    status1Text = "THE DEVICE IS IDLE";
  }

  if (Motor_status == 2) {
    status2Text = "THE BIN IS AT EMPTY POSITION";
  }
  if(Motor_status ==1) {
    status2Text = "THE BIN IS AT HOME POSITION";
  }
  if(Motor_status >= 3 && Motor_status <= 5) {
    status2Text = "THE BIN IS MOVING";
```

```
  }

  if (Water_level_status == 6) {
    status3Text = "WATER CAPACITY 100%";
  }
  if (Water_level_status == 5) {
    status3Text = "WATER CAPACITY 75%";
  }
  if (Water_level_status == 4) {
    status3Text = "WATER CAPACITY 50%";
  }
  if (Water_level_status == 3) {
    status3Text = "WATER CAPACITY 25%";
  }
  if (Water_level_status == 2) {
    status3Text = "WATER CAPACITY 0%";
  }
  if (Water_level_status == 1) {
    status3Text = "CRITICAL WATER LEVEL. PUMPING WATER UNAVAILABLE";
  }
}
```

5.2.1.5 PAGE 5

```
bool checkdelay(int amount,unsigned long prevtime)
{
  if(prevtime > millis())
  {
    unsigned long fixed_time = 4294967295 - prevtime;
    fixed_time += millis();
    if (millis()-fixed_time >= amount)
      return true;
    else
      return false;
  }
  else if (millis()-prevtime >= amount)
    return true;
  else
    return false;
}
```

5.2.2 ARDUINO No1

5.2.2.1 PAGE 1

```
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>

//i2c connection
const int SLAVE_ADDR = 8;
int command,sendcommand;


bool Lid_status,MPPV,Pause,home_status, fancommand=false;
bool Fan_status = false;
float Bin_temperature ;
int motormove=0;
int motorStatus=1;
float MOTEMP;

const int air_mosfet = 10;
const int ONE_WIRE_BUS1 = 2;
const int ONE_WIRE_BUS2 = 4;
const int TEMPERATURE_PRECISION = 9;
OneWire oneWire1(ONE_WIRE_BUS1);
OneWire oneWire2(ONE_WIRE_BUS2);
DallasTemperature sensor_1(&oneWire1);
DallasTemperature sensor_2(&oneWire2);

//motor
const int enable_Pin = 7;
const int direction_Pin = 8;
const int pulse_Pin = 9;
const int home_term = 12;
const int lid_term = 5;
bool pulse_status = false, direction_status = false;

float temp_readings[10];
int indextemp=0;
float temp_sum=0.0;//sum of temperatures
float temp_avrg = 0.0;// average temperatura value

const int heaterPin = 3;
int Heater_status;

unsigned long lastpulse_up = 0, lastpulse_down=0;
```

```
long pulse=0;
int cycle=0;
bool half_rotation1=false;

void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.println("Starting...");
//////////////////i2c setup
  Wire.begin(SLAVE_ADDR); // Initialize I2C communication with slave address
  Wire.onRequest(sendData); // Register the sendData function as the request
event handler
  Wire.onReceive(receiveData);// Register the receiveData function as the
receive event handler
//&&&&&&&&&&&&&&&&&----motor pins----&&&&&&&&&&&&&&&&&//
  pinMode(enable_Pin, OUTPUT);
  pinMode(direction_Pin, OUTPUT);
  pinMode(pulse_Pin, OUTPUT);
  pinMode(home_term, INPUT);
  pinMode(lid_term, INPUT_PULLUP);
  pinMode(heaterPin, OUTPUT);


  pinMode(air_mosfet, OUTPUT);

  digitalWrite(pulse_Pin,LOW);
//-------------------------------------------------------------------------------
-------------------------------------------------------------//
                                                          //tempsensors//
  sensor_1.begin();
  sensor_2.begin();

  Serial.println("Startup is complete");

}
//#############################################################################
#############################################################//
                                                            //MAIN LOOP//
void loop() {

  //Wire.onReceive(receiveData);
  //Wire.onRequest(sendData);

//-------------------------------------------------------------------------------
-------------------------------------------------------------//
  if((motormove>0)&&(motormove<5))
    digitalWrite(enable_Pin,LOW);
```

```
  else
    digitalWrite(enable_Pin,HIGH);

  if((Lid_status==false)&&(MPPV==true))
  {
    motor_control();
  }
//tempsensors
  sensor_1.requestTemperatures();
  sensor_2.requestTemperatures();
  MOTEMP = ((sensor_1.getTempCByIndex(0)+sensor_2.getTempCByIndex(0))/2);
  //Serial.println(MOTEMP);

  temp_sum-=temp_readings[indextemp];
  temp_readings[indextemp] = MOTEMP;
  temp_sum+=temp_readings[indextemp];

  indextemp = (indextemp + 1)%10;

  temp_avrg= temp_sum/10;
  Serial.println(temp_avrg);

//-------------------------------------------------------------------------------
-----------------------------------------------------------------//
  if(fancommand==true)
    digitalWrite(air_mosfet,HIGH);
  else
    digitalWrite(air_mosfet,LOW);

//-------------------------------------------------------------------------------
-----------------------------------------------------------------//
  ////////lid treminal check
  if(digitalRead(home_term)==HIGH)
  {
    home_status=true;
    Serial.println(home_status);
  }
  if(digitalRead(home_term)==LOW)
  {
    home_status=false;
    Serial.println(home_status);
  }

  if(Heater_status==1)
    digitalWrite(heaterPin,HIGH);
  else
    digitalWrite(heaterPin,LOW);
  ////////////////////////////////
```

```
  if(digitalRead(lid_term)==HIGH)
  {
    Lid_status=true;
    Serial.println(Lid_status);
  }
  if(digitalRead(lid_term)==LOW)
  {
    Lid_status=false;
    Serial.println(Lid_status);
  }
  Serial.print("motorStatus is ");
  Serial.println(motorStatus);
  Serial.print("motormove is ");
  Serial.println(motormove);
  Serial.print("mppv is ");
  Serial.println(MPPV);

}
```

5.2.2.2 PAGE 2

```
void motor_pulse_func(int x)
{
  while(pulse <= x)
  {
    if((pulse_status == false)&&
(checkdelay(1,lastpulse_up))&&(digitalRead(lid_term)==LOW)&&(MPPV==true))
    {
      digitalWrite(pulse_Pin,HIGH);
      //lastpulse_up= millis();
      pulse_status = true;
      Serial.println("pulse is high" );

      //return;
    }
    if((pulse_status == true)&&
(checkdelay(1,lastpulse_down))&&(digitalRead(lid_term)==LOW)&&(MPPV==true))
    {
      digitalWrite(pulse_Pin,LOW);
      //lastpulse_down= millis();
      pulse_status = false;
      Serial.println("pulse is low");
      //return;
      pulse++;
    }
  }
  pulse=0;
  return;
}

void motor_control()
{
  switch(motormove)
  {
    case 1://move to home position
      motorStatus=5;
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      if(digitalRead(home_term)== HIGH )
      {
        digitalWrite(direction_Pin,LOW);
        direction_status = false;
        motorStatus=1;
        motormove=0;
      }
      while(digitalRead(home_term)== LOW)
```

```
      {
        motor_pulse_func(17);
      }
      break;

////////////////////////////////////////////////////////////////////////////

    case 2://move to empty position
      motorStatus=5;//bis is moving
      digitalWrite(direction_Pin,LOW);
      direction_status = false;
      motor_pulse_func(5700);
      motormove=0;
      motorStatus = 2;//bin is at empty position
    break;

////////////////////////////////////////////////////////////////////////////

    case 3: // shuffling
    motorStatus=3;//bin is shuffling
    if((half_rotation1==false))
    {
      motor_pulse_func(3000);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      half_rotation1=true;
    }
    if(cycle==false)
    {
      motor_pulse_func(6000);
      digitalWrite(direction_Pin,LOW);
      direction_status = false;
      motor_pulse_func(6000);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      motor_pulse_func(6000);
      digitalWrite(direction_Pin,LOW);
      direction_status = false;
      motor_pulse_func(6000);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      cycle=true;
    }
    if(cycle==true)
    {
      while(digitalRead(home_term)== LOW )
      {
        motor_pulse_func(17);
```

```
      }
      if(digitalRead(home_term)== HIGH )
      {
        digitalWrite(direction_Pin,LOW);//normal direction status and reset
all flags
        direction_status = false;
        half_rotation1=false;
        cycle=0;
        motormove=0;
        motorStatus = 1;//bin is at home position
      }
    }
    break;
////////////////////////////////////////////////////////////////////////////
////////////////////
    case 4: //watering  2800 5600
    motorStatus = 4;//bin is watering
    if((half_rotation1==false))
    {
      motor_pulse_func(2500);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      half_rotation1=true;
    }
    if(cycle==false)
    {
      motor_pulse_func(5000);
      digitalWrite(direction_Pin,LOW);
      direction_status = false;
      motor_pulse_func(5000);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      motor_pulse_func(5000);
      digitalWrite(direction_Pin,LOW);
      direction_status = false;
      motor_pulse_func(5000);
      digitalWrite(direction_Pin,HIGH);
      direction_status = true;
      cycle=true;
    }
    if(cycle==true)
    {
      while(digitalRead(home_term)== LOW )
      {
        motor_pulse_func(17);
      }
      if(digitalRead(home_term)== HIGH )
      {
```

```
        digitalWrite(direction_Pin,LOW);//normal direction status and reset
all flags
        direction_status = false;
        half_rotation1=false;
        cycle=0;
        motormove=0;
        motorStatus = 1;//bin is at home position
      }
    }
    break;

    default: break;
  }
}
```

5.2.2.3 PAGE 3

```
void sendData()
{
  // Send the requested value based on the command
  switch (sendcommand)
  {
    case 11: // COMMAND_MOTOR_STATUS
      Wire.write((byte*)&motorStatus, sizeof(int));
      sendcommand=0;
    break;
    case 12: // COMMAND_BIN_TEMPERATURE
      Wire.write((byte*)&temp_avrg, sizeof(float));
      sendcommand=0;
    break;
    case 13: // COMMAND_LID_TERMINAL_STATUS
      Wire.write((byte*)&Lid_status, sizeof(bool));
      sendcommand=0;
    break;
    case 15: // COMMAND_FAN_STATUS
      Wire.write((byte*)&fancommand, sizeof(bool));
      sendcommand=0;
    break;
    case 16: // COMMAND_CO2
      Wire.write((byte*)&Heater_status, sizeof(bool));
      sendcommand=0;
    break;
    default:
    break;
  }
}
void receiveData(int byteCount) {
  while (Wire.available()) {
    byte command = Wire.read();
    sendcommand = command;
    switch (command)
    {
      case 1: //move to home position
        motormove = 1;
        command=0;
      break;
      case 2://move to empty position
        motormove = 2;
        command=0;
      break;
      case 3://shufle
        motormove = 3;
```

```
        command=0;
      break;
      case 4://water move
        motormove = 4;
        command=0;
      break;
      case 5://activate fan
        fancommand = true;
        command=0;
      break;
      case 6: //deactivate fan
        fancommand = false;
        command=0;
      break;
      case 7://activate heater
        Heater_status=1;
        command=0;
      break;
      case 8://deactivate heater
        Heater_status=0;
        command=0;
      case 9://solar controler output on
        MPPV=true;
        command=0;
      break;
      case 17://solar controler output off
        MPPV=false;
        command=0;
      break;

      default: break;
    }
  }
}
```

5.2.2.4 PAGE 4

```
bool checkdelay(int amount,unsigned long prevtime)
{
  if(prevtime > millis())
  {
    unsigned long fixed_time = 4294967295 - prevtime;
    fixed_time += millis();
    if (millis()-fixed_time >= amount)
      return true;
    else
      return false;
  }
  else if (millis()-prevtime >= amount)
    return true;
  else
    return false;
}
```

5.2.3 ARDUINO NO2

5.2.3.1 PAGE 1

```
#include <HX711_ADC.h>
#include <EEPROM.h>
#include <Wire.h>
//i2c adress
const int SLAVE_ADDR = 9;
int command,sendcommand;

//HX711 constructor (dout pin, sck pin)
HX711_ADC LoadCell_1(2, 3); //HX711 1
HX711_ADC LoadCell_2(4, 5); //HX711 2
HX711_ADC LoadCell_3(6, 7); //HX711 3
HX711_ADC LoadCell_4(8, 9); //HX711 4
bool newDataReady=0;

float kilos_readings[10];
int indexkilos=0;
float kilos_sum=0.0;//sum of kilos
float kilos_avrg = 0.0;// average kilos value
float kilos;

//MQ-4
int methane_measure_sequense=3;
const int meth_PWM = 11;


float meth_readings[10];
int indexmeth=0;
float meth_sum=0.0;//sum of meth
float meth_avrg = 0.0;// average meth value

//PUMP
const int pumpPWM = 10;
bool pump_status=false;
//WATEL LEVEL
int water_level;

bool MPPV;

void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.println("Starting...");
```

```
  LoadCell_1.begin();
  LoadCell_2.begin();
  LoadCell_3.begin();
  LoadCell_4.begin();
  unsigned long stabilizingtime = 2000; // tare preciscion can be improved by
adding a few seconds of stabilizing time
  boolean _tare = false; //set this to false if you don't want tare to be
performed in the next step
  byte loadcell_1_rdy = 0;
  byte loadcell_2_rdy = 0;
  byte loadcell_3_rdy = 0;
  byte loadcell_4_rdy = 0;
  while ((loadcell_1_rdy + loadcell_2_rdy + loadcell_3_rdy + loadcell_4_rdy) <
4)
    { //run startup, stabilization and tare, all modules simultaniously
      if (!loadcell_1_rdy) loadcell_1_rdy =
LoadCell_1.startMultiple(stabilizingtime, _tare);
      if (!loadcell_2_rdy) loadcell_2_rdy =
LoadCell_2.startMultiple(stabilizingtime, _tare);
      if (!loadcell_3_rdy) loadcell_3_rdy =
LoadCell_3.startMultiple(stabilizingtime, _tare);
      if (!loadcell_4_rdy) loadcell_4_rdy =
LoadCell_4.startMultiple(stabilizingtime, _tare);
    }

  LoadCell_1.setCalFactor(101.23); // user set calibration value (float)
  LoadCell_2.setCalFactor(101.34); // user set calibration value (float)
  LoadCell_3.setCalFactor(101.38); // user set calibration value (float)
  LoadCell_4.setCalFactor(101.22); // user set calibration value (float)
  LoadCell_1.tareNoDelay();
  LoadCell_2.tareNoDelay();
  LoadCell_3.tareNoDelay();
  LoadCell_4.tareNoDelay();

  //MQ-4
  pinMode(A0,INPUT);
  pinMode(meth_PWM, OUTPUT);
  digitalWrite(meth_PWM,LOW);

  //PUMP
  pinMode(pumpPWM, OUTPUT);
  digitalWrite(pumpPWM,LOW);

  //water level
  pinMode(A3,INPUT);//100%
  pinMode(A2,INPUT);//75%
  pinMode(A1,INPUT);//50%
  pinMode(13,INPUT);//25%
```

```
  pinMode(12,INPUT);//0%

////////i2c
  Wire.begin(SLAVE_ADDR); // Initialize I2C communication with slave address
  Wire.onRequest(sendData); // Register the sendData function as the request
event handler
  Wire.onReceive(receiveData); // Register the receiveData function as the
receive event handler
  Serial.println("Startup is complete");
}

void loop()
{
////////////////////////////////////methane proc
  if(methane_measure_sequense == 1)
  {
    digitalWrite(meth_PWM,HIGH);
  }
  if(methane_measure_sequense == 2)
  {
    meth_sum-=meth_readings[indexmeth];
    meth_readings[indexmeth] = analogRead(A0);
    meth_sum+=meth_readings[indexmeth];

    indexmeth = (indexmeth + 1)%10;

    meth_avrg= meth_sum/10;
    //Serial.println(meth_avrg);
    Serial.print("ppm are  ");
    Serial.println(meth_avrg);
  }
  if(methane_measure_sequense == 3)
  {
    digitalWrite(meth_PWM,LOW);
    //methane_measure_sequense == 3;
  }

////////////////////////////////////weight proc
  if
((LoadCell_1.update())&&(LoadCell_2.update())&&(LoadCell_3.update())&&(LoadCel
l_4.update()))
  {
    newDataReady = true;
  }
  if ((newDataReady))
  {
    float a = LoadCell_1.getData();
    float b = LoadCell_2.getData();
```

```
    float c = LoadCell_3.getData();
    float d = LoadCell_4.getData();
    newDataReady=0;
    kilos=a+b+c+d;
    //Serial.println(kilos);

    kilos_sum-=kilos_readings[indexkilos];
    kilos_readings[indexkilos] = kilos;
    kilos_sum+=kilos_readings[indexkilos];

    indexkilos = (indexkilos + 1)%10;

    kilos_avrg= kilos_sum/10;
    Serial.print("kilos are ");
    Serial.println(kilos_avrg);
  }
////////////////////////////////////WATER LEVEL
  if((digitalRead(12))&&(digitalRead(13))&&(digitalRead(A1))&&(digitalRead(A2)
)&&(digitalRead(A3)))
    water_level=1;//lit works with reverse logic if there is water returns 0
if not then 1
  if((digitalRead(12))&&(digitalRead(13))&&(digitalRead(A1))&&(digitalRead(A2)
)&&(!digitalRead(A3)))
    water_level=2;
  if((digitalRead(12))&&(digitalRead(13))&&(digitalRead(A1))&&(!digitalRead(A2
))&&(!digitalRead(A3)))
    water_level=3;
  if((digitalRead(12))&&(digitalRead(13))&&(!digitalRead(A1))&&(!digitalRead(A
2))&&(!digitalRead(A3)))
    water_level=4;
  if((digitalRead(12))&&(!digitalRead(13))&&(!digitalRead(A1))&&(!digitalRead(
A2))&&(!digitalRead(A3)))
    water_level=5;
  if((!digitalRead(12))&&(!digitalRead(13))&&(!digitalRead(A1))&&(!digitalRead
(A2))&&(!digitalRead(A3)))
    water_level=6;
////////////////////////////////////
//pump control
  if(pump_status==true)
  {
    digitalWrite(pumpPWM,HIGH);
  }
  if(pump_status==false)
  {
    digitalWrite(pumpPWM,LOW);
  }
}
```

5.2.3.2 PAGE 2

```
void sendData() {
  //int command = Wire.read(); // Read the command sent by the master

  // Send the requested value based on the command
  switch (sendcommand)
  {
    case 21: // COMMAND_KILOS
      Wire.write((byte*)&kilos, sizeof(float));
      sendcommand=0;
      break;
    case 22: // COMMAND_WEIGHT
      Wire.write((byte*)&water_level, sizeof(int));
      sendcommand=0;
      break;
    case 23: // COMMAND_CO2
      Wire.write((byte*)&pump_status, sizeof(bool));
      sendcommand=0;
      break;
    case 24: // COMMAND_CO2
      Wire.write((byte*)&methane_measure_sequense, sizeof(int));
      sendcommand=0;
      break;
    case 25: // COMMAND_CO2
      Wire.write((byte*)&meth_avrg, sizeof(float));
      sendcommand=0;
      break;
  }
}
void receiveData(int byteCount) {
  while (Wire.available()) {
    byte command = Wire.read(); // Read the command sent by the master
    sendcommand = command;

    // Process the received command and perform the corresponding action
    switch (command) {
      case 1: //activate mq4 measure seq
        methane_measure_sequense=1;
        command=0;
      break;
      case 2://take measures;
        methane_measure_sequense=2;
        command=0;
      break;
      case 3://deactivate mq4 measure seq;
        methane_measure_sequense=3;
        command=0;
```

```
      break;
      case 4://activate pump
        pump_status=true;
        command=0;
      break;
      case 5://deactivate pump
        pump_status=false;
        command=0;
      break;
      case 9://solar controler output on
        MPPV=true;
        command=0;
      break;
      case 17://solar controler output off
        MPPV=false;
        command=0;
      break;

      default: break;
    }
  }
}
```

5.3 ΕΠΕΞΗΓΗΣΗ ΤΟΥ ΚΑΘΕ ΚΩΔΙΚΑ

5.3.1 ESP32

Το ESP32 έχει τον ρόλο του συντονιστή, έχει αναλάβει να στέλνει αιτήματα μέσω του I2C στα Arduino για να του στείλουν πίσω δεδομένα μετρήσεων είτε για να εκτελέσουν κάποια λειτουργεία όπως να περιστρέφει τον κάδο είτε να ενεργοποιηθεί ο ανεμιστήρας, οι εντολές αυτές προκύπτουν από τις μετρήσεις τις οποίες λαμβάνει. Στον ίδιο δίαυλο συνδέεται και το CJMCU-8118 το οποίο το διαχειρίζεται επίσης το ESP32 για να λαμβάνει μετρήσεις.
Επόμενη λειτουργεία που εκτελεί είναι η ιστοσελίδα, να λαμβάνει αιτήματα και να ανανεώνει τα status ανάλογα. Για τον μόνο αισθητήρα που είναι υπεύθυνο το ESP, είναι για τον αισθητήρα εξόδου του solar converter όπου ανάλογα της κατάστασης του θα εξαρτηθούν και τα αιτήματα που θα κάνει στα Arduino.
Τέλος ανά ένα χρονικό διάστημα στέλνει τις μετρήσεις που έχει λάβει σε ένα υπολογιστικό φύλο Excel μέσω του IFTTT που από εκεί μπορεί να γίνει ανάλυση.

5.3.2 ARDUINO 1

Σε αυτό τον μικροελεγκτή είναι συνδεδεμένοι οι αισθητήρες θερμοκρασίας του κάδου, ένας τερματικός διακόπτης για το καπάκι της συσκευής για λόγους ασφαλείας, οι έξοδοι για τον έλεγχο του κινητήρα, ένας επαγωγικός διακόπτης normally open ως θέση εκκίνησης και οι έξοδοι για χειρισμό του ανεμιστήρα και των αντιστάσεων.
Οι μετρήσεις που λαμβάνει από τα θερμόμετρα προθέτονται μεταξύ τους και υπολογίζεται ο μέσος όρος τους έπειτα προστίθενται σε μια δομή δεδομένων FIFO (first in first out) 10 στοιχείων και έπειτα υπολογίζεται από όλα τα στοιχεία της δομής ο μέσος όρος.
Έχει εφαρμοστεί αυτή η μέθοδος έτσι ώστε σε περίπτωση που υπάρξει εσφαλμένη μέτρηση να μειωθεί το σφάλμα και να αποφευχθούν ανεπιθύμητες ενέργειες από το ESP.
Έπειτα στον κινητήρα έχει προγραμματιστεί έτσι ώστε να μην ξεπερνάει το εύρος μοιρών που μπορεί να περιστραφεί, καθώς και η μέθοδος με την οποία γίνεται η παλμοδότηση είναι βήμα βήμα και χωρίς την χρήση εντολών delay λόγω του ότι σταματάει όλο το πρόγραμμα και έχουμε κατάσταση αδράνειας μέχρι να τελειώσει ο χρόνος αναμονής της εντολής delay.
Προσοχή έχει δοθεί στον τερματικό ασφαλείας του καπακιού διότι σε περίπτωση που ανοίξει το καπάκι της συσκευής σταματάει να περιστρέφεται ο κάδος για αποφυγή ατυχήματος.

5.3.3 ARDUINO 2

Σε αυτό το Arduino έχουν συνδεθεί οι δυναμοκυψέλες, η έξοδος του αισθητήρα μεθανίου, η έξοδος για ενεργοποίηση του και οι αισθητήρες υγρασίας από τη δεξαμενή του νερού.
Τα δεδομένα από τις δυναμοκυψέλες προστίθενται μεταξύ τους και μετά αντιμετωπίζονται ακριβώς όπως οι θερμοκρασίες στο Arduino 1.
Οι αισθητήρες υγρασίας έχουν δυο μόνο καταστάσεις αφού χρησιμοποιείται η ψηφιακή έξοδος που σημαίνει είτε υπάρχει νερό είναι όχι, έτσι αφού είναι τοποθετημένες κατακόρυφα στην δεξαμενή τότε μόνο οι δυναμοκυψέλες που είναι βυθισμένες στο νερό στέλνουν σήμα και κατά συνέπεια ο μικροελεγκτής καταλαβαίνει σε ποια στάθμη είναι το νερό. Η ανάλυση είναι ανά 25%, οπότε τα επίπεδα είναι 100%, 75%, 50%, 25%, 0%. Στο 0% απαγορεύει το ESP32 να λειτουργήσει η αντλία νερού.
Ο αισθητήρας μεθανίου ελέγχεται κατά κύριο λόγο από το ESP32, ουσιαστικά για να υπάρχει εξοικονόμηση ισχύος πρέπει να λειτουργεί ανά μεγαλύτερα διαστήματα από τους άλλους αισθητήρες, όμως το γεγονός ότι πρέπει να θερμανθεί για 2 λεπτά είναι ένα ακόμα πρόβλημα που πρέπει να αντιμετωπιστεί. Με βάση τα παραπάνω προέκυψε η εξής λύση, η διαδικασία μέτρησης να γίνεται σε στάδια.
Το πρώτο στάδιο είναι το να ενεργοποιηθεί ο αισθητήρας και να ξεκινήσει να ζεσταίνεται, το δεύτερο στάδιο είναι ότι αφού ζεσταθεί το Arduino να ξεκινήσει να λαμβάνει μετρήσεις και να τις βάζει σε μια δομή δεδομένων για υπολογισμό μέσου όρου και στο μεταξύ το ESP32 να είναι σε αναμονή ώστε να προλάβει να υπολογιστεί ο μέσος όρος.
Το τρίτο στάδιο είναι να αποσταλούν τα δεδομένα στο ESP32 και να απενεργοποιηθεί ο αισθητήρας, έπειτα βάσει των μετρήσεων το ESP32 θα ρυθμίσει τον χρόνο για την επόμενη μέτρηση.

5.4 ΛΟΓΙΣΜΙΚΌ ΑΝΑΠΤΥΞΗΣ

Το λογισμικό που χρησιμοποιήθηκε είναι το Arduino IDE το οποίο είναι ένα δωρεάν λογισμικό ανάπτυξης κώδικα. Η γλώσσα στην οποία μπορεί κανείς να γράψει είναι η Wiring η οποία είναι μια παραλλαγή της C/C++ για μικροελεγκτές αρχιτεκτονικής AVR, όπως ο Atmega και υποστηρίζει όλες τις βασικές δομές της C, καθώς και μερικά χαρακτηριστικά της C++.
Για αυτή τη γλώσσα βέβαια έχουν γραφτεί πολλές ακόμα βιβλιοθήκες που μπορούν να υποστηρίξουν πληθώρα μικροελεγκτών όπως και ο ESP32 και άλλοι ακόμα, πέρα από τις βιβλιοθήκες αυτές υπάρχουν και έτοιμες βιβλιοθήκες μέσα από τις ίδιες τις εταιρείες που παράγουν αισθητήρες για διευκόλυνση του προγραμματιστή.

Η βασική αρχή λειτουργείας είναι ότι έχει δύο βασικά υποπρογράμματα το ένα είναι το setup στο οποίο γράφεται κώδικας ο οποίος θα εκτελεστεί μια μόνο φορά εκεί συνήθως γίνονται όλες οι αρχικοποιήσεις των εισόδων και των εξόδων και των μεταβλητών γίνεται η χειραψία σε περίπτωση που είναι συνδεδεμένο το σύστημα μαζί με άλλα συστήματα για να μπορεί να τρέξει στο main loop.
Στο main loop γράφεται ο κώδικας που θα εκτελείται κυκλικά όσο λειτουργεί ο μικροελεγκτής. Εκτελείται αφού πρώτα τρέξει ο κώδικας του setup.

5.5 ΙΣΤΟΣΕΛΙΔΑ

Για τον έλεγχο της συσκευής χρειάζεται μια διεπαφή χρήστη, οι δυνατότητες του ESP32 επιτρέπουν να συνδεθεί σε τοπικό Wi-Fi δίκτυο και να λειτουργήσει σε λειτουργεία σταθμού, το οποίο σημαίνει ότι για να το διαχειριστεί κάποιος θα πρέπει να είναι στο ίδιο δίκτυο συνδεδεμένος και να έχει πρόσβαση στην IP του μικροελεγκτή.

Το μενού επιλογών είναι απλό για να μην περιοριστεί η μνήμη του ESP και να είναι όσο το δυνατόν πιο γρήγορη η απόκριση του.

Η δοκιμή και η κατασκευή του κώδικα έγινε στην ιστοσελίδα w3schools με την εφαρμογή tryit (https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic).

Εικόνα 48: Το μενού επιλογών.

Ο κώδικας αυτός ουσιαστικά τυπώνεται στην ιστοσελίδα μια φορά από τον μικροελεγκτή και ύστερα για να γίνουν οι απαραίτητες αλλαγές θα πρέπει να πατηθεί κάποιο πλήκτρο και να σταλεί αίτημα από τη σελίδα ή να στείλει αίτημα η σελίδα μέσω της **setInterval(function()** για ενημέρωση των στοιχείων ανά 1 δευτερόλεπτο.
Έπειτα αφού λάβει αίτημα συντάσσει ένα κείμενο JSON και το στέλνει στην σελίδα, μετά η σελίδα το διαβάζει και αλλάζει τις τιμές ανάλογα.

JSON (JavaScript Object Notation) είναι ένα κείμενο, που μπορεί να διαβαστεί από τον άνθρωπο, για τη μετάδοση δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικών-τιμών και τύπου δεδομένων συστοιχιών (ή οποιασδήποτε άλλης σειριοποίησιμης τιμής). Πρόκειται για ένα πολύ κοινό μορφότυπο δεδομένων που χρησιμοποιείται για την ασύγχρονη επικοινωνία περιηγητή - διακομιστή.

Το σύστημα που επεξηγήθηκε παραπάνω ονομάζεται AJAX (Asynchronous JavaScript and XML), με τη χρήση AJAX τεχνικών, οι εφαρμογές διαδικτύου μπορούν να στέλνουν και να ανακτούν δεδομένα από έναν διακομιστή ασύγχρονα χωρίς να παρεμβαίνουν στην εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Με την αποσύνδεση του επιπέδου των δεδομένων που έχουν την δυνατότητα αλλαγής από το επίπεδο παρουσίασης της σελίδας, η AJAX επιτρέπει σε ιστοσελίδες, και κατ' επέκταση σε εφαρμογές διαδικτύου, να αλλάζουν το περιεχόμενο τους δυναμικά, χωρίς να χρειάζεται να φορτωθεί εκ νέου ολόκληρη η σελίδα. Στην πράξη, οι σύγχρονες εφαρμογές συνήθως χρησιμοποιούν JSON , αντί για XML, λόγω των πλεονεκτημάτων του JSON που υπάρχουν έτοιμα στην γλώσσα JavaScript.

```
<!DOCTYPE html>
<html>
<head>
  <title>Compost Control</title>
  <style>body { display: flex; justify-content: center; align-items: center; height: 100vh; }</style>
</head>
<body>
  <div style='text-align: center;'>
    <h1>Compost Control</h1>
    <button onclick="sendCommand('Button1')">START</button>
    <button onclick="sendCommand('Button2')">PAUSE</button>
    <button onclick="sendCommand('Button3')">STOP</button>
    <button onclick="sendCommand('Button4')">EMPTY</button>
    <button onclick="sendCommand('Button5')">HOME</button>
    <h2><span id="status1">Status 1 Text</span></h2>
    <h2><span id="status2">Status 2 Text</span></h2>
    <h2><span id="status3">Status 3 Text</span></h2>
  </div>
  <script>
    function sendCommand(sercommand) {
      var xhttp = new XMLHttpRequest();
      xhttp.open('GET', '/serommand?cmd=' + sercommand, true);
      xhttp.send();
    }

    setInterval(function() {
      var xhttpStatus = new XMLHttpRequest();
      xhttpStatus.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          var statusData = JSON.parse(this.responseText);
          document.getElementById('status1').innerText = statusData.status1;
          document.getElementById('status2').innerText = statusData.status2;
          document.getElementById('status3').innerText = statusData.status3;
        }
      };
      xhttpStatus.open('GET', '/getStatus', true);
      xhttpStatus.send();
    }, 1000); // Update status every 1 second
  </script>
</body>
</html>
```

Εικόνα 49: Ο κώδικας για το μενού επιλογών.

5.6 WEBHOOK IFTTT

Το IFTTT, συντομογραφία των λέξεων “If This Then That”, είναι μια πλατφόρμα αυτοματισμού που επιτρέπει στους χρήστες να δημιουργούν προσαρμοσμένες ροές εργασίας συνδέοντας διάφορες εφαρμογές και υπηρεσίες μεταξύ τους. Λειτουργεί χρησιμοποιώντας ένα συγκεκριμένο έναυσμα σε μια εφαρμογή ή υπηρεσία προκαλεί μια ενέργεια σε μια άλλη εφαρμογή ή υπηρεσία.
Από τεχνικής άποψης, το IFTTT είναι μια διαδικτυακή υπηρεσία που χρησιμοποιεί ένα RESTful API για την επικοινωνία μεταξύ διαφορετικών εφαρμογών και υπηρεσιών. Προσφέρει ένα ευρύ φάσμα με περισσότερες από 600 διαφορετικές υπηρεσίες, συμπεριλαμβανομένων πλατφορμών μέσων κοινωνικής δικτύωσης, έξυπνων οικιακών συσκευών, εργαλείων παραγωγικότητας και άλλων.
Ένα από τα βασικά χαρακτηριστικά του IFTTT είναι η απλότητα και η ευκολία χρήσης του. Οι χρήστες μπορούν να δημιουργήσουν προσαρμοσμένες εφαρμογές με λίγα μόνο κλικ, χωρίς να χρειάζονται γνώσεις προγραμματισμού. Αυτό επιτυγχάνεται μέσω της φιλικής προς το χρήστη διεπαφής του IFTTT.

Στην εργασία αυτή το έναυσμα είναι το ESP32 όπου στέλνει τα δεδομένα και ανακατευθύνονται σε ένα φύλλο Excel σε έναν λογαριασμό στην Google.

ΚΕΦΑΛΑΙΟ 6 : ΠΡΟΤΑΣΕΙΣ ΠΕΡΕΤΑΙΡΩ ΜΕΛΕΤΗΣ

Με βάση την παραπάνω μελέτη η συσκευή θα μπορούσε να θεωρηθεί ικανή να παράγει το υλικό για το οποίο είναι διαμορφωμένη, βέβαια όμως λόγω της επιλογής των υλικών οι δυνατότητες είναι περιορισμένες και η απόδοση σχετικά μικρή.

Σε επίπεδο υλικών θα μπορούσε να αντικατασταθεί το ξύλινο περίβλημα με ένα μεταλλικό με αντοχή στην οξείδωση και με κατάλληλη επικάλυψη ώστε να είναι και θερμομονωτικό για να επιτευχθεί καλύτερη διατήρηση της θερμοκρασίας εντός της συσκευής.
Ο κάδος θα μπορούσε να διαμορφωθεί με ένα καπάκι, και με προσθήκη στον άξονα ενός περιστρεφόμενου συνδέσμου (slip ring) θα μπορεί να περιστρέφεται ελεύθερα πράγμα που δεν μπορεί να γίνει σε αυτή την κατασκευή.
Σε επίπεδο λογισμικού θα μπορούσε να γίνει βελτιστοποίηση στους κώδικες για να μπορεί να τρέχει πιο ομαλά και να έχει καλυτέρους χρόνους απόκρισης και ταυτόχρονα να χρειάζεται λιγότερη ισχύ χρησιμοποιώντας τις λειτουργείες αναστολής όσο στους μικροελεγκτές όσο και στους αισθητήρες.
Τέλος άλλη μια πρόταση είναι να μελετηθεί η αυτονομία του για χρήση κατάλληλου φωτοβολταϊκού πάνελ και αντίστοιχης χωρητικότητας μπαταριών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Waqas, M., Hashim, S., Humphries, U. W., Ahmad, S., Noor, R., Shoaib, M., ... & Lin, H. A. (2023). Composting Processes for Agricultural Waste Management: A Comprehensive Review. Processes, 11(3), 731.
- Καλοβρέκτης Κ., Κατέβας Ν. Αισθητήρες μέτρησης και ελέγχου, 3η Έκδοση, Εκδόσεις Τζιόλα.
- Haug, RogerTim. *The practical handbook of compost engineering*. Routledge, 2018.
- Ευφυής Γεωργία και Κυκλική Βιοοικονομία – SmartBIC -Agricultural University of Athens – Σημειώσεις AUA Open eClass

- https://else.fcim.utm.md/pluginfile.php/11966/mod_resource/content/1/app105.pdf
- https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf
- https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf
- https://el.wikipedia.org/wiki/%CE%A0%CF%84%CE%B7%CF%84%CE%B9%CE%BA%CE%AE_%CE%BF%CF%81%CE%B3%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%AD%CE%BD%CF%89%CF%83%CE%B7
- https://www.umweltsensortechnik.de/en/gas-sensors/mox-gas-sensors-functional-principle.html
- https://www.ti.com/lit/ds/symlink/hdc1080.pdf?ts=1692177762473&ref_url=https%253A%252F%252Fwww.google.com%252F
- https://www.ti.com/lit/an/snaa216/snaa216.pdf?ts=1692182098041
- https://e2e.ti.com/support/sensors-group/sensors/f/sensors-forum/1044307/hdc1080-type-of-temperature-sensor
- https://en.wikipedia.org/wiki/Brokaw_bandgap_reference
- https://e2e.ti.com/support/sensors-group/sensors/f/sensors-forum/969182/hdc1080-technical-information-technology/3580342#3580342
- https://www.electronicshub.org/humidity-sensor-types-working-principle/
- https://www.mouser.com/datasheet/2/38/AV02-0470EN-189984.pdf
- https://mecheltron.com/sites/default/files/webresources/MechanicalElectroMech/SteppErMotors/pdf/datasheets-steppermotors/60BYGH450D-02.pdf
- https://www.elprocus.com/what-is-hybrid-stepper-motor-working-its-applications/
- https://www.leadshine.com.ua/pdf/dm566.pdf
- https://www.haitronic.cn/index.php?route=product/product&path=91_141&product_id=1375&limit=100
- https://www.mouser.com/catalog/specsheets/TLP785_datasheet_en_20120220.pdf
- https://docs.arduino.cc/retired/boards/arduino-pro-mini

- https://picockpit.com/raspberry-pi/el/uart-the-universal-asynchronous-receiver-transmitter/
- https://ftdichip.com/products/ft232rl/
- https://datasheethub.com/wp-content/uploads/2022/08/SEN0114_Web.pdf
- https://components101.com/modules/soil-moisture-sensor-module
- https://el.wikipedia.org/wiki/JSON
- https://el.wikipedia.org/wiki/Ajax_(%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CF%8C%CF%82)
- https://topreviews.gr/ifttt/