
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# “ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΔΙΕΡΓΑΣΙΑ ΠΛΗΡΩΣΗΣ ΔΟΧΕΙΩΝ ΜΕ ΡΕΥΣΤΟ, ΜΕ ΤΟΠΙΚΟ ΚΑΙ ΑΠΟΜΑΚΡΥΣΜΕΝΟ ΕΛΕΓΧΟ ”

Φιλίππου Θωμάς (ΑΜ: 7224)

ΕΠΙΒΛΕΠΩΝ:

Χαδέλλης Λουκάς

Τοπάλης Ευάγγελος

ΠΑΤΡΑ, 2022

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, 2022

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή
2. Ονοματεπώνυμο, Υπογραφή
3. Ονοματεπώνυμο, Υπογραφή

**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.

# Περίληψη

Η παρούσα πτυχιακή αφορά την υλοποίηση μιας αυτόματης γραμμής παραγωγής, στην οποία εκτελείτε μια διεργασία για την μεταφορά δοχείων δύο χρωμάτων, την αυτόματη πλήρωση συγκεκριμένου χρώματος δοχείων με ρευστό και στη συνέχεια τον διαχωρισμός τους. Η εργασία αυτή βασίζεται στην πλατφόρμα arduino καθώς επίσης έχει γίνει χρήση του λογισμικού Labview και της διαδικτυακής πλατφόρμας thingSpeak ώστε να υπάρχει δυνατότητα τοπικού και απομακρυσμένου ελέγχου καθώς και καταγραφή και επεξεργασία δεδομένων και τιμών αισθητήρων.



## ABSTRACT

This dissertation concerns the implementation of an automatic production line, in which a process is performed for the transfer of two-color containers, the automatic filling of a specific color of containers with fluid and then their separation. This work is based on the arduino platform as well as the use of Labview software and the online platform thingSpeak for the possibility of local and remote control as well as recording and processing of data and sensor values.



Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τους καθηγητές μου κ. Ευάγγελο Τοπάλη και κ. Λουκά Χαδέλλη για την βοήθεια και την καθοδήγηση κατά την διάρκεια της εργασίας, όπως επίσης και τον καθηγητή κ. Βασίλειο Καψάλη ως μέλος της εξεταστικής επιτροπής.

# Περιγραφή

Το θέμα της εργασίας αυτής είναι η κατασκευή μια γραμμής παραγωγής αποτελούμενη από έναν ταινιόδρομο, έναν ρομποτικό βραχίονα και αισθητήρες, η οποία θα εκτελεί μια αυτόματη διεργασία που θα μεταφέρει δοχεία δύο χρωμάτων (μπλε και κόκκινο), θα γεμίζει με ρευστό αυτά του κόκκινου χρώματος και στο τέλος θα τα διαχωρίζει.

Πιο συγκεκριμένα έχει κατασκευαστεί ένας ταινιόδρομος πάνω στον οποίο υπάρχουν δύο αισθητήρες κίνησης (ένας στην αρχή και ένας στο τέλος), ένας αισθητήρας αναγνώρισης χρώματος και ένα σωληνάκι το οποίο συνδέεται σε μια αντλία που αντλεί ρευστό από μια δεξαμενή που επίσης κατασκευάστηκε. Η δεξαμενή έχει και αυτή αισθητήρα θερμοκρασίας και αισθητήρα υπερήχων για τον έλεγχο της στάθμης. Στο τέλος του ταινιόδρομου υπάρχει ένας ρομποτικός βραχίονας που συναρμολογήθηκε από ένα αγορασμένο kit. Η διεργασία που εκτελείτε αναλυτικά είναι η εξής:

Ο ταινιόδρομος μεταφέρει τα δοχεία. Όταν ο πρώτος αισθητήρας κίνησης ανιχνεύσει κάποιο δοχείο, τότε ο ταινιόδρομος σταματά και ο αισθητήρας χρώματος αναγνωρίζει το χρώμα του δοχείου και το αποθηκεύει στον arduino. Αν το χρώμα είναι κόκκινο τότε η αντλία ξεκινά και γεμίζει με ρευστό το δοχείο και ύστερα ξεκινά και πάλι ο ταινιόδρομος. Αν το χρώμα είναι μπλε, ξεκινά ο ταινιόδρομος χωρίς να γεμίσει το δοχείο. Στη συνέχεια όταν φτάσουν στο τέλος του ταινιόδρομου και ο δεύτερος αισθητήρας που βρίσκετε εκεί τα ανιχνεύσει, τότε αν το χρώμα του είναι μπλε ο διάδρομος το αποσύρει αλλιώς αν είναι κόκκινο σταματά και ο ρομποτικός βραχίονας το παίρνει και το τοποθετεί σε διαφορετική θέση εκτός του ταινιόδρομου. Υπάρχουν επίσης δύο σενάρια, το πρώτο είναι αν η δεξαμενή αδειάσει και κάποιο κόκκινο δοχείο έχει φτάσει στο σημείο για να γεμίσει, τότε όλη η λειτουργία θα σταματήσει μέχρι να ξαναγεμίσει η δεξαμενή. Το δεύτερο σενάριο είναι ότι υπάρχουν δύο θέσεις στις οποίες αφήνει τα δοχεία ο βραχίονας, έτσι κάθε φορά που μεταφέρει κάποιο κόκκινο δοχείο θα εναλλάζει τις θέσεις, δηλαδή το πρώτο κόκκινο που θα πάρει θα το βάλει στην πρώτη θέση, το δεύτερο στην δεύτερη, το τρίτο πάλι στην πρώτη κλπ.

Επιπλέον θα μπορούμε να παρακολουθούμε τιμές μετρήσεων και καταστάσεις καθώς και να ελέγχουμε την διεργασία αυτή μέσω ενός περιβάλλοντος ελέγχου με εικονικά όργανα, που θα επικοινωνεί σειριακά, με την χρήση του λογισμικού Labview.
Τέλος κάποια δεδομένα με τιμές μετρήσεων και καταστάσεις ενεργοποιητών θα ανεβαίνουν στην πλατφόρμα Thingspeak μέσω σύνδεσης ethernet, όπου θα προβάλλονται σε διαγράμματα και θα μπορούν να επεξεργαστούν και να αναλύονται.

# Περιεχόμενα

## Ευρετήριο Πινάκων



## Ευρετήριο Σχημάτων - Εικόνων

# Κεφάλαιο 1ο Εισαγωγή

## 1.1 Σκοπός

Ο Σκοπός της εργασίας αυτής είναι η κατασκευή της γραμμής παραγωγής και η δημιουργία μιας διεργασίας που θα εκτελείτε ώστε να μεταφέρονται δοχεία κόκκινου και μπλε χρώματος, να γεμίζουν με υγρό μόνο τα κόκκινα δοχεία και στη συνέχεια να αποσύρονται τα μπλε δοχεία μέσω του ταινιόδρομου και να μετακινούνται τα κόκκινα σε άλλη θέση μέσω του ρομποτικού βραχίονα. Επίσης η εκμάθηση της πλατφόρμας arduino και ο προγραμματισμός του για την δημιουργία της διεργασίας που θα εκτελείτε και την επικοινωνία με άλλα λογισμικά και πλατφόρμες που θα χρησιμοποιηθούν, η γνωριμία και ο προγραμματισμός του λογισμικού Labview για την δημιουργία ενός περιβάλλοντος ελέγχου της γραμμής παραγωγής όπου θα απεικονίζονται οι μετρήσεις των αισθητήρων σε εικονικά όργανα και θα μπορούμε να χειριζόμαστε κάποιους ενεργοποιητές (κινητήρα, αντλία, button) από μακριά, και επιπλέον η εκμάθηση της πλατφόρμας thingspeak στην οποία θα ανεβαίνουν δεδομένα όπως τιμές μετρήσεων και καταστάσεις ενεργοποιητών τα οποία θα εμφανίζονται σε γραφήματα, και θα μπορούν να επεξεργάζονται και να αναλύονται ώστε να χρησιμοποιηθούν για κάποιες ανάγκες μας ή για την δημιουργία νέων γραφημάτων.

Τέλος ο συνδυασμός των παραπάνω πλατφορμών και λογισμικών και η ενσωμάτωσή τους σε διάφορα συστήματα με αισθητήρες και ενεργοποιητές, στην προκειμένη περίπτωση στην κατασκευή της εργασίας αυτής, έχει ως σκοπό την γνώση πάνω στην υλοποίηση συστημάτων αυτομάτου ελέγχου (τοπικού και απομακρυσμένου), την μελέτη της λειτουργίας των διαφόρων αυτών αισθητήρων, ενεργοποιητών και άλλων hardware και την καταγραφή και επεξεργασία δεδομένων ενός συστήματος όπως συναντάμε και σε βιομηχανίες.

## 1.2 Δομικά Μέρη της Εργασίας

Το πρώτο μέρος της εργασίας αυτής (**Κεφάλαιο 2**) και έχει να κάνει με τον ορισμό του αυτοματισμού, τα είδη και τις δυνατότητες, καθώς και την ιδέα για την υλοποίηση της κατασκευής. Το δεύτερο μέρος (**Κεφάλαιο 3**), αφορά την περιγραφή όλων των τεχνολογιών, πλατφορμών, λογισμικών και άλλων εργαλείων ανάπτυξης που χρησιμοποιήθηκαν για την εργασία αυτή.

Στο τρίτο μέρος (**Κεφάλαιο 4**), περιγράφονται αναλυτικά όλα μέρη της κατασκευής (μηχανικά, ηλεκτρολογικά, ηλεκτρονικά), καθώς επίσης και η υλοποίηση της κατασκευής και το τελικό κύκλωμα.

Το τέταρτο μέρος (**Κεφάλαιο 5**), αφορά τον προγραμματισμό του arduino, του Labview και το ανέβασμα μετρήσεων στο thingspeak με αναλυτική περιγραφή της διαδικασίας.

Το πέμπτο μέρος (**Κεφάλαιο 6**), έχει να κάνει με τον ThingSpeak και συγκεκριμένα με την διαχείριση και την επεξεργασία των γραφημάτων καθώς και την δημιουργία αναλύσεων.

Τέλος στο έκτο μέρος (**Κεφάλαιο 7**), υπάρχουν τα συμπεράσματα και κάποιες μελλοντικές επεκτάσεις που μπορούν να γίνουν για την εργασία αυτή.

# Κεφάλαιο 2ο Η Έννοια του Αυτοματισμού

## 2.1 Ορισμός

Με τον όρο αυτοματισμό εννοούμε την τυποποίηση μιας διαδικασίας με την εύρεση ενός αλγορίθμου δηλαδή μια σειρά καλώς ορισμένων ενεργειών που πρέπει να εκτελούνται σε πεπερασμένο χρόνο για να έχουμε κάποιο επιθυμητό αποτέλεσμα. Σκοπός της διαδικασίας αυτής είναι για παράδειγμα η κατασκευή μιας αυτόνομης μηχανής που για κάποια είσοδο που θα δοθεί θα εκτελεί αυτές τις εντολές χωρίς ανθρώπινη παρέμβαση. Επίσης ο αυτοματισμός σαν έννοια ασχολείται με τον έλεγχο διεργασιών και στηρίζετε στην θεωρία ελέγχου, που ασχολείται με την συμπεριφορά των δυναμικών συστημάτων και στους μηχανισμούς ανάδρασης, όπου ανάδραση είναι η ανατροφοδότηση της εξόδου ενός συστήματος, στην είσοδο του. [1]

Στις μέρες μας η σημασία του αυτοματισμού είναι μεγάλη στη βιομηχανία, χρησιμοποιεί πολλά και εξειδικευμένα ηλεκτρονικά, αισθητήρια, ενεργοποιητές, μικροϋπολογιστές, συστήματα πραγματικού χρόνου, εξειδικευμένους υπολογιστές και άλλα συστήματα τεχνολογίας πληροφοριών, για να ελέγχουν διεργασίες και μεταβλητές και να συγχρονίζουν τη ροή εισόδων από αισθητήρες με την ροή εντολών για τους ενεργοποιητές, όπως παράδειγμα ένας ρομποτικός βραχίονας. [1]

## 2.2 Είδη αυτοματισμού (κλασικός αυτοματισμός, αυτοματισμός με μικροϋπολογιστές)

Οι αυτοματισμοί που υπάρχουν σήμερα έχουν αλλάξει σε μεγάλο βαθμό από αυτούς πριν δεκαετίες όσον αφορά τα εργαλεία, τα ηλεκτρολογικά – ηλεκτρονικά εξαρτήματα και την εξέλιξη της τεχνολογίας. Συγκεκριμένα, πριν δεκαετίες οι αυτοματισμοί βασιζόταν αποκλειστικά σε ηλεκτρολογικά εξαρτήματα όπως ρελέ, χρονικά κλπ. και όλα αυτά συνδεόταν με καλώδια. Αυτός ο τρόπος αναφέρεται σήμερα ως κλασικός αυτοματισμός.

Πλέον τα ηλεκτρονικά, οι υπολογιστές και γενικά η τεχνολογία βρίσκονται σε τρομερή ανάπτυξη και έχουν επικρατήσει των αναλογικών. Οι απαιτήσεις για αυτοματισμούς είναι αυξημένες και πιο σύνθετες, έτσι λοιπόν οι τεχνολογίες αυτές έχουν επηρεάσει τον τομέα του αυτοματισμού και τους συναντάμε και στην βιομηχανία. Συγκεκριμένα έχουμε αυτοματισμούς με ηλεκτρονικές διατάξεις, με μικροϋπολογιστές που συνδυάζονται με τα κλασικά ηλεκτρολογικά κυκλώματα και έχουν δυνατότητες ασύρματης σύνδεσης, ψηφιακού προγραμματισμού και απομακρυσμένου ελέγχου. [2] [3]

### 2.2.1 Περιγραφή και δυνατότητες της κάθε κατηγορίας

Ο κλασικός αυτοματισμός όπως αναφέρετε  πιο πάνω, χρησιμοποιεί ηλεκτρολογικά στοιχεία και βασίζετε στην ενσύρματη λογική. Είναι απαραίτητο πρώτα να γίνει το ηλεκτρολογικό σχέδιο ώστε μετά να κατασκευαστεί και ο πίνακας.. Ύστερα αφού γίνει επιλογή των κατάλληλων στοιχείων θα πρέπει να γίνει η σύνδεση τους μέσα στον πίνακα με βάση το σχέδιο .  Σε περίπτωση που χρειαστεί να αλλάξουμε την λειτουργία του αυτοματισμού και να προστεθούν και νέα υλικά, θα πρέπει να ξαναγίνει σχέδιο και να ξαναφτιάξουμε την συνδεσμολογία από την αρχή. Επιπλέον αν υπάρχει κάποιο σφάλμα στο σύστημα, είναι πιο δύσκολη και χρονοβόρα η διάγνωσή του, όπως και η συντήρηση του.  [2] [3]

Στους αυτοματισμούς με μικροϋπολογιστές αρχικά πρέπει να επιλέξουμε τον κατάλληλο μικροϋπολογιστή που χρειάζεται για να καλυφθούν οι απαιτήσεις της εφαρμογής, να επιλέξουμε επίσης τα άλλα ηλεκτρολογικά – ηλεκτρονικά υλικά όπως αισθητήρες , ενεργοποιητές, ηλεκτρονικές διατάξεις για έλεγχο στροφών κινητήρων κλπ. Εδώ δεν είναι απαραίτητο το σχέδιο γιατί η συνδεσμολογία είναι απλή και είναι ίδια ανεξάρτητα από την λειτουργία που θέλουμε. Για την ακρίβεια όλα τα στοιχεία συνδέονται στις κατάλληλες εισόδους – εξόδους του μικροϋπολογιστή, ενώ η λειτουργία του αυτοματισμού καθορίζετε από τον προγραμματισμό του μικροϋπολογιστή.  Η διάγνωση σφαλμάτων εδώ είναι πιο εύκολη και γρήγορη.

Με τον τρόπο αυτό έχουμε και την δυνατότητα απομακρυσμένου ελέγχου και προγραμματισμού μέσω ασύρματης ή ενσύρματης δικτύωσης με διάφορα λογισμικά, την συλλογή, επεξεργασία και αποστολή  δεδομένων και τιμών αισθητηρίων. [2] [3]

### 2.2.2 Πλεονεκτήματα των αυτοματισμών με μικροϋπολογιστές.

Οι σύγχρονοι αυτοματισμοί με μικροϋπολογιστές παρουσιάζουν περισσότερα πλεονεκτήματα έναντι των κλασικών αυτοματισμών. Συγκεκριμένα:

- Μικρότερο κόστος και χρόνο κατασκευής του πίνακα.
- Μικρό κόστος συντήρησης.
- Εύκολη και γρήγορη διάγνωση σφαλμάτων.
- Δυνατότητα απομακρυσμένου ελέγχου και προγραμματισμού.
- Δυνατότητα ασύρματης και ενσύρματης δικτύωσης.
- Για να αλλάξουμε το σενάριο μιας εφαρμογής αρκεί να τροποποιήσουμε τον κώδικα.

- Καλύτερη και αποτελεσματικότερη διαχείριση της παραγωγής και κατανάλωσης ενέργειας.
- Ένας μικροϋπολογιστής μπορεί να ελέγχει περισσότερες από μία εφαρμογές.

Ένα μειονέκτημα των μικροϋπολογιστών που χρησιμοποιούνται για βιομηχανική χρήση, όπως τα PLC, είναι το σχετικά μεγάλο κόστος τους εάν χρησιμοποιηθούν για απλές εφαρμογές με ελάχιστες απαιτήσεις. Σε αυτή την περίπτωση συνήθως χρησιμοποιείτε ο κλασικός αυτοματισμός για καθαρά οικονομικό λόγο.

Υπάρχουν όμως κι άλλοι μικροϋπολογιστές - μικροελεγκτές με χαμηλό κόστος που συναντάμε κυρίως σε οικιακούς, γεωργικούς, για διάφορα projects και γενικά μη βιομηχανικούς αυτοματισμούς λόγω τον μικρότερων δυνατοτήτων τους, όπως arduino, raspberry, κλπ.. Στην περίπτωση αυτή συμφέρει να τους χρησιμοποιήσουμε έναντι του κλασικού αυτοματισμού. [2] [3]

Έτσι και για την κατασκευή αυτή επιλέχτηκε η πλατφόρμα arduino για τον λόγο ότι είναι οικονομική και ιδανική για τις απαιτήσεις μας.

## 2.3 Εφαρμογές των αυτοματισμών

- Στην βιομηχανία (παραγωγή αγαθών, κλπ),
- Στην γεωργία (έξυπνα θερμοκήπια, μελέτη εδάφους, κλπ.)
- Στις μεταφορές (συγκοινωνίες, ναυτιλία, αεροναυτιλία, αυτοκίνηση, κλπ )
- Στις επικοινωνίες (τηλεπικοινωνίες, κλπ)
- Στην κτηνοτροφία (αυτόματες εργασίες, κλπ)
- Στην ηλεκτροδότηση (ανανεώσιμες πηγές ενέργειας, προστασία γραμμών μεταφοράς, κλπ)
- Στα σπίτια μας (έξυπνο σπίτι, ηλεκτρικές συσκευές, θέρμανση, κλιματισμός, συναγερμός, κλπ)
- Σε κτήρια (ανελκυστήρες, συστήματα πυρανίχνευσης – πυρασφάλειας, συστήματα συναγερμών, κλπ)
- Στην ιατρική (ρομπότ που πραγματοποιούν επεμβάσεις, κλπ)

Όπως αναφέρεται και πιο πάνω, μία από τις πολλές εφαρμογές των αυτοματισμών είναι η βιομηχανία, η οποία σχετίζετε και με το θέμα της παρούσας εργασίας. Εκεί συναντάμε ταινιόδρομους και ρομποτικούς βραχίονες που σε συνδυασμό με αισθητήρες εκτελούν με μεγάλη ακρίβεια εργασίες, βαριές και μη, για τις διάφορες ανάγκες που μπορεί να υπάρχουν.

Έτσι η κατασκευή αυτή είναι μιας μικρής κλίμακας γραμμής παραγωγής που μοιάζει με αυτές των βιομηχανιών και η ιδέα ξεκίνησε από δύο βίντεο στο youtube με αυτοματισμό ταινιόδρομων. Το πρώτο βίντεο είναι από ένα κανάλι με όνομα "Make Robots" και δείχνει την λειτουργία ενός ταινιόδρομου βασισμένο σε arduino με δύο αισθητήρες υπερύθρων ο οποίος μεταφέρει κύβους λευκού και μαύρου χρώματος και όταν φτάσουν στο τέλος ένας ρομποτικός βραχίονας τα παίρνει, τα πηγαίνει πάνω στον αισθητήρα υπερύθρων για να αναγνωριστεί το χρώμα και στην συνέχεια τα διαχωρίζει. [4]

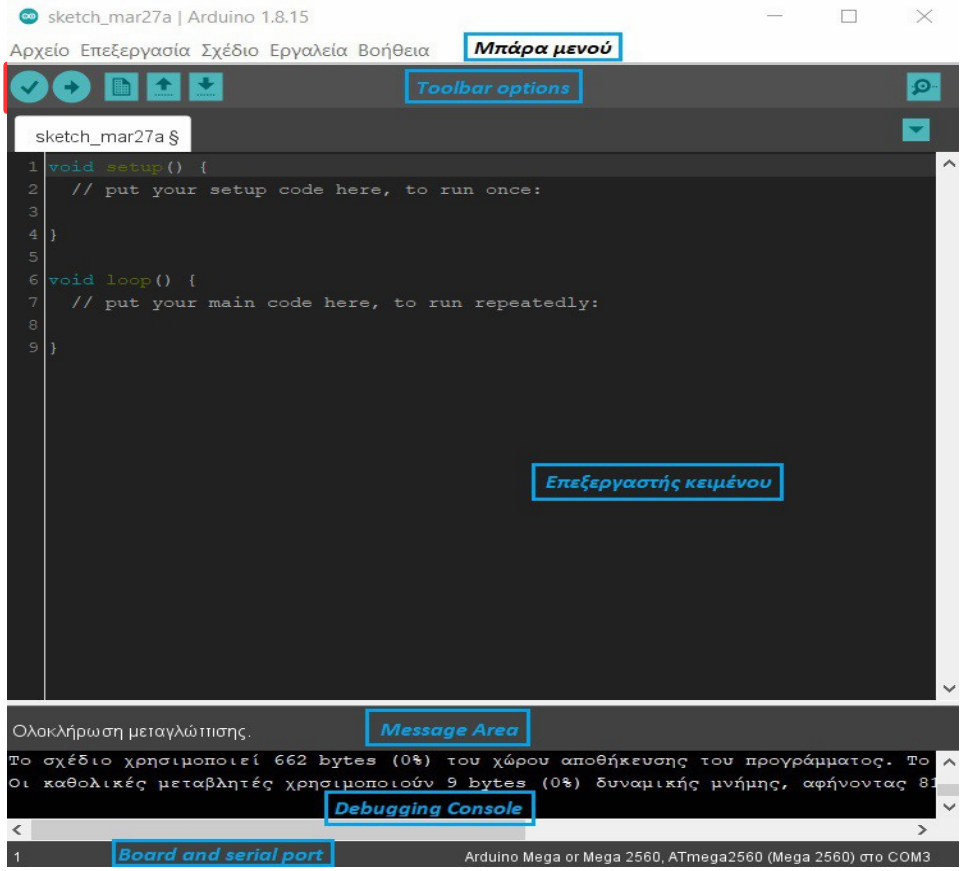
Το δεύτερο βίντεο είναι το κανάλι "Easy HomeMade Projects" και δείχνει την λειτουργία ενός ταινιόδρομου με arduino και δύο αισθητήρες υπερύθρων που μεταφέρει και γεμίζει βάζα με νερό. [5]

Με βάση τα παραπάνω βίντεο αποφασίστηκε να συνδυαστούν και οι δύο λειτουργίες και να δημιουργηθούν και νέα σενάρια προσθέτοντας περισσότερους αισθητήρες όπως αναγνώρισης χρώματος, θερμοκρασίας, υπερήχων για την στάθμη της δεξαμενής, με σκοπό τον μεγαλύτερο έλεγχο. Επίσης ήταν επιθυμητό η λειτουργία της κατασκευής αυτή να θυμίζει την λειτουργία μιας βιομηχανίας, έτσι λοιπόν χρησιμοποιήθηκε το λογισμικό Labview και η πλατφόρμα Thingspeak για να δημιουργηθεί τοπικός και απομακρυσμένος έλεγχος.

# Κεφάλαιο 3ο Τεχνολογίες, Πλατφόρμες και Εργαλεία ανάπτυξης που χρησιμοποιήθηκαν

## 3.1 Arduino IDE

Το Arduino IDE είναι ένα περιβάλλον προγραμματισμού. Το IDE είναι τα αρχικά των Integrated Development Environment που σημαίνει ολοκληρωμένο περιβάλλον ανάπτυξης. Περιλαμβάνει δυνατότητες όπως η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και μπορεί να κάνει μεταγλώττιση των προγραμμάτων και να τα φορτώνει στις πλακέτες. Τα προγράμματα που γράφονται στο περιβάλλον αυτό λέγονται sketches και σώζονται με την επέκταση (.ino). Το λογισμικό αυτό το κατεβάζουμε δωρεάν από την σελίδα του arduino. [6] [7]



Εικόνα 3.1: Περιβάλλον Arduino IDE (black theme)

Κάτω από την μπάρα των μενού είναι τα κουμπιά των εργαλείων . Το πρώτο είναι για να κάνει επικύρωση και μεταγλώττιση τον κώδικα, το δεύτερο για να ανεβάσει το πρόγραμμα στην πλακέτα, το τρίτο για να δημιουργήσουμε νέο sketch, το τέταρτο για να ανοίξουμε κάποιο αρχείο, το πέμπτο για να αποθηκεύσουμε και το τελευταίο στα δεξιά είναι για να ανοίξουμε το serial monitor.

Κάτω από αυτά είναι ο επεξεργαστής κειμένου. Εκεί γράφουμε τον κώδικα.

Μετά έχουμε την περιοχή μηνυμάτων όπου παίρνουμε πληροφορίες κατά την διάρκεια μεταγλώττισης και ανεβάσματος όπως επίσης και σφαλμάτων.

Από κάτω είναι η κονσόλα αποσφαλμάτωσης (debugging console) που δείχνει πληροφορίες για το σχέδιο και τον χώρο που καταλαμβάνει στην πλακέτα που έχουμε επιλέξει και επίσης σε περίπτωση σφάλματος δείχνει αναλυτικά το μήνυμα και πληροφορίες για αυτό.

Τέλος κάτω δεξιά βλέπουμε τον τύπο πλακέτας και την σειριακή θύρα που έχουμε επιλέξει. [7]

### 3.1.1 Γλώσσα Προγραμματισμού Arduino

Η γλώσσα προγραμματισμού του arduino είναι μια παραλλαγή – μίξη της C και C++ και είναι μια υλοποίηση της wiring. Το arduino IDE έρχεται με μια βιβλιοθήκη λογισμικού γραμμένη σε C/C++ που ονομάζεται wiring που κάνει τον προγραμματισμό ακόμη πιο εύκολο. [6]

### 3.1.2 Καταχώρηση βιβλιοθηκών

Οι βιβλιοθήκες παρέχουν επιπλέον λειτουργικότητα στο πρόγραμμά μας, όπως πιο εύκολο τρόπο προγραμματισμού και επικοινωνίας με διάφορα hardware, επεξεργασία δεδομένων και άλλα. Το arduino IDE έχει ήδη κάποιες εγκατεστημένες βιβλιοθήκες τις οποίες μπορούμε να συμπεριλάβουμε στο πρόγραμμά μας πατώντας στο μενού ***Σχέδιο > Συμπερίληψη βιβλιοθήκης*** , και να επιλέξουμε όποια χρειαζόμαστε. Θα την δούμε να εμφανίζετε στο πρόγραμμα με την δήλωση #include και το όνομα της βιβλιοθήκης μέσα σε ( < > ) ή σε ( “ “ ). [8]

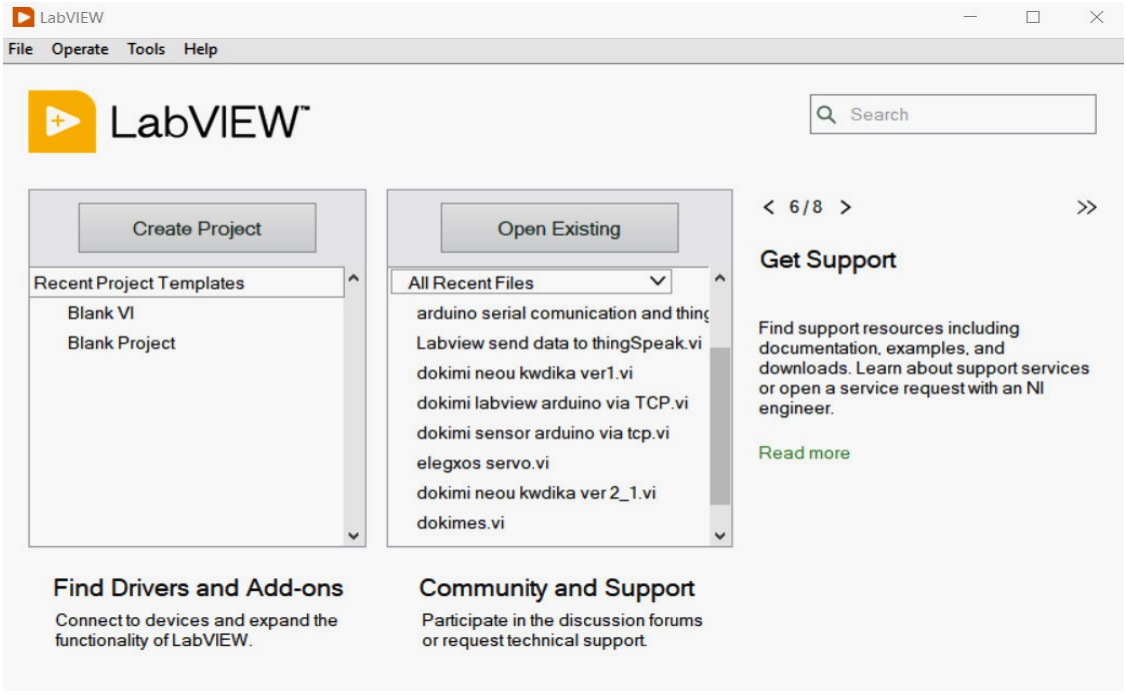
```
#include <Ethernet.h>
#include “Ethernet.h”
```

Επειδή οι βιβλιοθήκες ανεβαίνουν μαζί με το πρόγραμμα στην πλακέτα, καταλαμβάνουν χώρο στη μνήμη, οπότε αν δεν χρειαζόμαστε κάποια βιβλιοθήκη στον κωδικά μας , καλό είναι να την σβήνουμε.

## 3.2 Λογισμικό Labview

Το Labview ή αλλιώς (Laboratory Virtual Instrument Engineering Workbench) είναι ένα περιβάλλον σχεδίασης και ανάπτυξης εφαρμογών από την National Instruments, που έχουν την δυνατότητα να επικοινωνούν και με εξωτερικό hardware. Συγκεκριμένα περιλαμβάνει κάποιες λειτουργίες σύνδεσης και επικοινωνίας με βασικά πρότυπα ελέγχου όπως σειριακά (RS232, RS485 και RS422), USB, Ethernet, IEEE 1394, VISA, Modbus, GPIB, VXI, PXI και OPC servers. Χρησιμοποιεί μια ισχυρή γραφική γλώσσα προγραμματισμού που ονομάζετε "G", με πλήθος εργαλείων για συλλογή μετρήσεων, ανάλυση και παρουσίαση, έλεγχο οργάνων και άλλα, που μας επιτρέπει να δημιουργήσουμε εφαρμογές μετρήσεων και αυτοματισμού με την χρήση μπλοκ διαγραμμάτων (block diagrams). Το Labview μεταφράζει τα διαγράμματα αυτά σε κώδικα μηχανής. [9] [10] [11]

Στην παρακάτω εικόνα φαίνεται το αρχικό παράθυρο του Labview από το οποίο μπορούμε να ανοίξουμε κάποιο υπάρχον αρχείο ή να δημιουργήσουμε ένα νέο.




Εικόνα 3.2: Αρχικό παράθυρο Labview

Το Labview αποτελείτε από δύο παράθυρα, το Front panel και το Block diagram τα οποία είναι τα κύρια μέρη ενός προγράμματος. Στο Front panel ή αλλιώς παράθυρο γραφικών μπορούμε να μέσα από εικονικά όργανα (VI's) να παρακολουθούμε, να ελέγχουμε και να καταγράφουμε τις συνθήκες από κάποιο σύστημα αυτοματισμού.

Στο Block diagram προγραμματίζουμε την εφαρμογή με διάφορες συναρτήσεις σε μορφή block στοιχείων, με ρουτίνες, με καλώδια που μεταφέρουν πληροφορία και άλλα, τα οποία τα βρίσκουμε μέσα από παλέτες και υποπαλέτες όπως και στο Front panel.

### 3.2.1 National Instruments VISA

Το NI VISA (Virtual Instrument Software Architecture) ή στα ελληνικά (αρχιτεκτονική λογισμικού εικονικών οργάνων), είναι μια διεπαφή προγραμματισμού (API) για να μπορούμε μέσα από διάφορα λογισμικά ανάπτυξης εφαρμογών της NI όπως το LabView, το LabVIEW NXG, το LabWindows/CVI και το Measurement Studio να επικοινωνούμε και να ελέγχουμε hardware και όργανα μέσω διάφορων διασυνδέσεων όπως GPIB, VXI, PXI, Ethernet/LXI, σειριακά, USB, HiSLIP, VXI-11 (με TCP/IP). Το API εγκαθίσταται μέσω του προγράμματος οδήγησης NI-VISA, το οποίο περιλαμβάνει και κάποια βοηθητικά προγράμματα και λειτουργίες έλεγχου χαμηλού επιπέδου. [12] [13] [14]

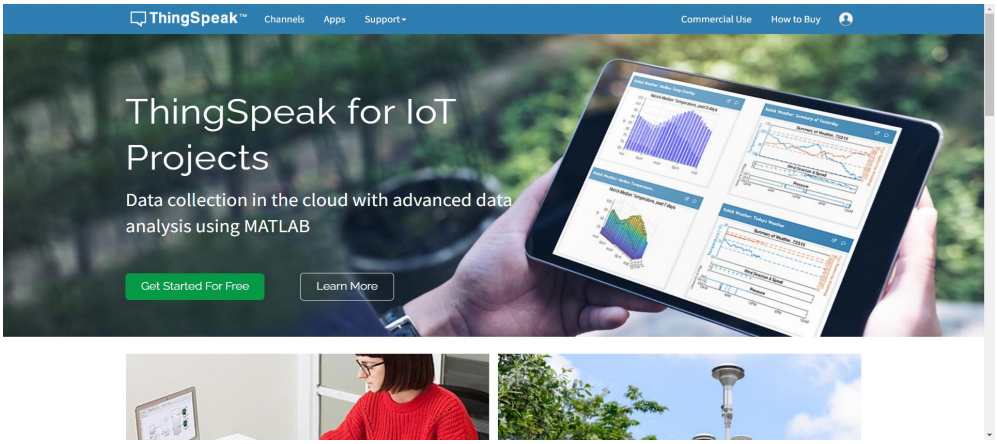
## 3.3 Διαδίκτυο των Πραγμάτων (IoT)

Το διαδίκτυο των πραγμάτων ή (Internet of things) αποτελεί ένα δίκτυο όπου διάφορες “έξυπνες” συσκευές, δηλαδή συσκευές που ενσωματώνουν αισθητήρες, λογισμικό, ηλεκτρονικά και συνδεσιμότητα σε δίκτυο, συνδέονται μεταξύ τους (τοπικό δίκτυο) ή στο διαδίκτυο με σκοπό την ανταλλαγή δεδομένων και την δυνατότητα να μπορούμε να τα ελέγχουμε από έναν υπολογιστή ή κινητό απομακρυσμένα. Τέτοιες συσκευές για παράδειγμα μπορεί να είναι οικιακές συσκευές, αυτοκίνητα με ενσωματωμένους αισθητήρες, κάμερες, φώτα, συστήματα ασφαλείας, έξυπνο ψυγείο και άλλα. Οι συσκευές αυτές συνδέονται σε μια πλατφόρμα, στην οποία στέλνουν τα δεδομένα τους και αυτή τα λαμβάνει τα αναλύει και μπορεί να στέλνει κατάλληλες και χρήσιμες πληροφορίες σε εφαρμογές που μπορεί να υπάρχουν για την αντιμετώπιση συγκεκριμένων αναγκών. [15]

Η χρησιμότητα του IoT στις μέρες μας είναι μεγάλη όπως και η ζήτηση από αγοραστές, αφού η χρήση του συμβάλει στην αυτονομία πολλών πραγμάτων γύρω μας, πράγμα που οι άνθρωποι πλέον αποζητούν. Η χρήση του επίσης επεκτείνετε και στις επιχειρήσεις οι οποίες εκμεταλλεύονται την δυνατότητα αποθήκευσης και επεξεργασίας δεδομένων από cloud συστήματα. [15]

### 3.3.1 ThingSpeak

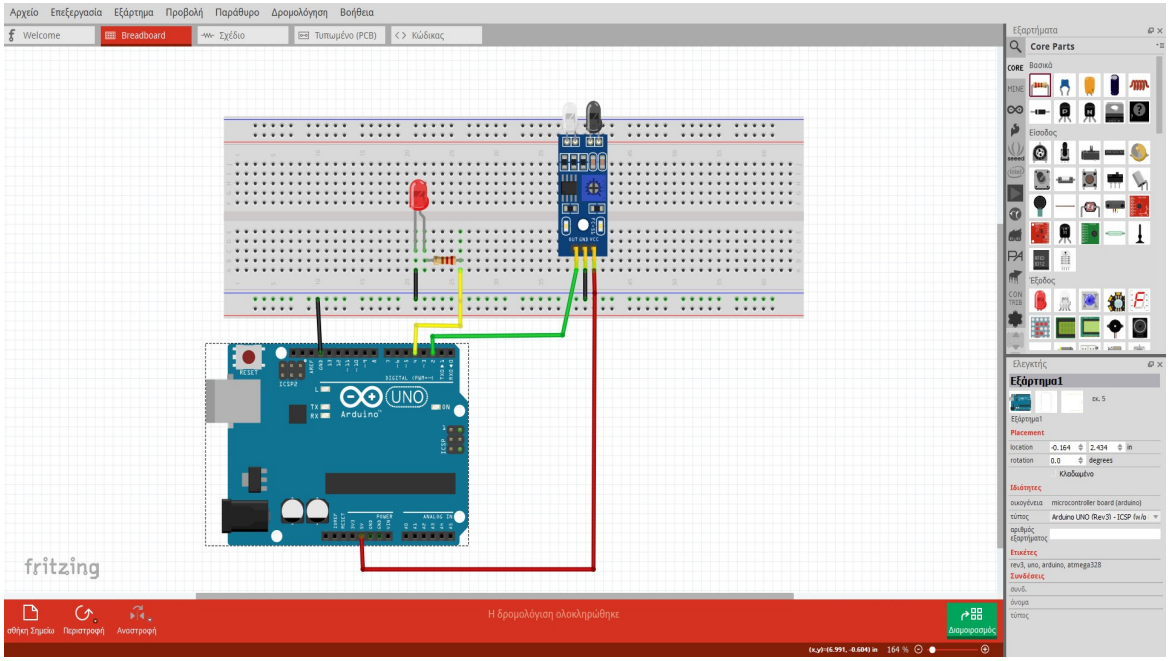
Το ThingSpeak είναι μια πλατφόρμα ανάλυσης IoT που μπορούμε να καταγράφουμε, να οπτικοποιούμε και να αναλύουμε ζωντανές ροές δεδομένων στο cloud. Κυκλοφόρησε αρχικά το 2010 από το ioBridge ως υπηρεσία υποστήριξης εφαρμογών IoT. Με το ThingSpeak μπορούμε να απεικονίσουμε τα δεδομένα μας που στέλνονται από τις συσκευές μας. Έχει ενσωματωμένη υποστήριξη από το λογισμικό Matlab της Mathworks που μας δίνει την δυνατότητα να εκτελούμε κώδικα Matlab για να αναλύουμε και να επεξεργαζόμαστε δεδομένα διαδικτυακά. Χρησιμοποιεί τα πρωτόκολλα HTTP και MQTT για να στέλνουμε εύκολα δεδομένα από τις συσκευές μας. Μπορούμε επίσης να συγκεντρώνουμε δεδομένα κατ' απαίτηση από άλλους, να εκτελούμε αναλύσεις αυτόματα με βάση χρονοδιαγράμματα ή συμβάντα, και να ενεργούμε αυτόματα στα δεδομένα μας και να επικοινωνούμε χρησιμοποιώντας υπηρεσίες τρίτων όπως το Twilio ή το Twitter. [16] [17]




Εικόνα 3.3: Αρχική σελίδα του ThingSpeak [Πηγή]

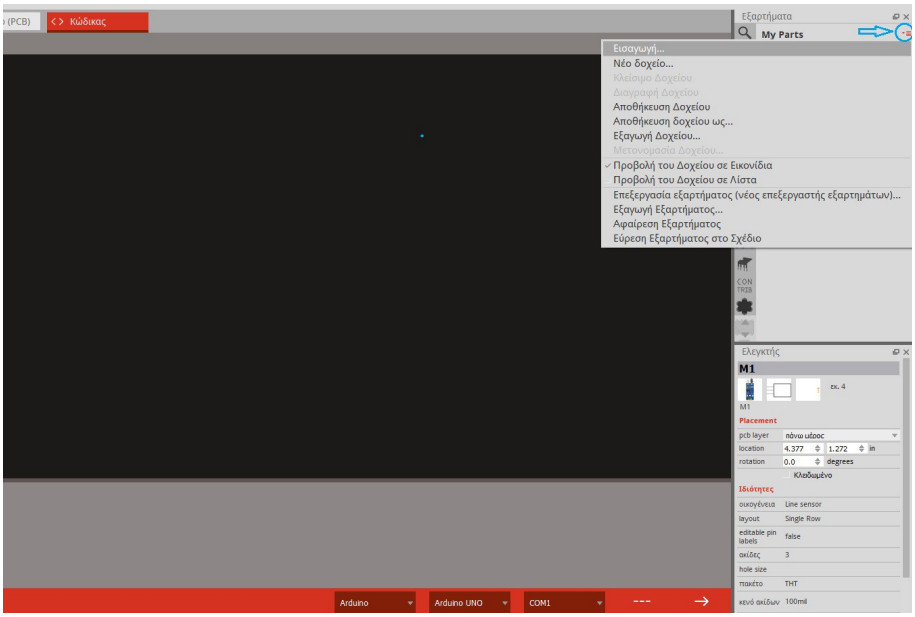
### 3.4 Fritzing

Το Fritzing είναι ένα λογισμικό σχεδιασμού (CAD) ανοιχτού κώδικα με το οποίο σχεδιάζουμε εικονικά ηλεκτρονικά κυκλώματα. Δημιουργήθηκε από την σχολή εφαρμοσμένων εφαρμοσμένων επιστημών του Πότσνταμ στην Γερμανία. Έχει δημιουργηθεί στο πνεύμα της γλώσσας προγραμματισμού Processing και του arduino και μας επιτρέπει να σχεδιάσουμε εικονικά κυκλώματα βασισμένα σε arduino. Δημιουργεί επίσης αυτόματα ένα ηλεκτρονικό σχέδιο, όπως επίσης και μια διάταξη PCB την οποία μπορούμε να παραγγείλουμε από το Fab.Fritzing.org. Τέλος μπορούμε να γράψουμε και κώδικα για arduino και να τον ανεβάσουμε από εκεί σε κάποια πλακέτα. Το λογισμικό είναι διαθέσιμο για Windows, MacOS X και Linux. Όλα τα κυκλώματα με arduino της εργασίας αυτής έχουν σχεδιαστεί με αυτό το πρόγραμμα. [18] [19]



Εικόνα 3.4: Καρτέλα Breadboard του Fritzing με κύκλωμα.

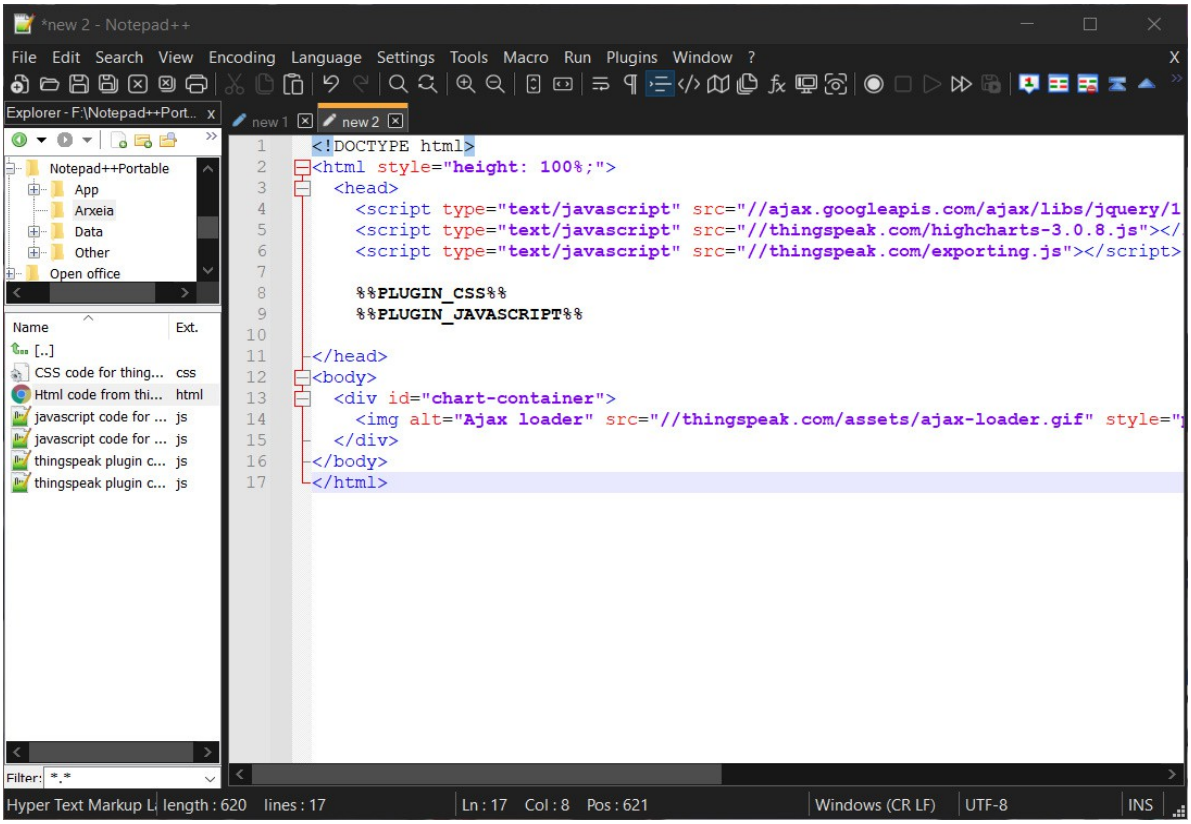
Το Fritzing έχει εγκατεστημένη βιβλιοθήκη εξαρτημάτων τα οποία χωρίζονται σε παλέτες και φαίνονται δεξιά στην εικόνα 3.4. Μπορούμε επίσης να προσθέσουμε νέα εξαρτήματα που θα βρούμε στο διαδίκτυο. Για να γίνει αυτό θα πρέπει να βρούμε και να κατεβάσουμε το εξάρτημα που θέλουμε. Θα είναι ένα αρχείο με την επέκταση (.fzp) ή (fzpz). Στη συνέχεια μέσα στο πρόγραμμα στον κατάλογο με τα εξαρτήματα  πατάμε πάνω δεξιά στο εικονίδιο που φαίνεται στην παρακάτω εικόνα, και επιλέγουμε εισαγωγή. Μέσα από το παράθυρο που θα ανοίξει βρίσκουμε το αρχείο που κατεβάσαμε και πατάμε άνοιγμα. Θα εμφανιστεί στην παλέτα MINE.



Εικόνα 3.5: Εισαγωγή νέου εξαρτήματος

## 3.5 Notepad++

Το Notepad++ είναι ένα δωρεάν λογισμικό επεξεργασίας κειμένου και κώδικα. Διαθέτει επισήμανση σύνταξης για 78 γλώσσες προγραμματισμού, αναδίπλωση κώδικα, περιορισμένη αυτόματη συμπλήρωση και επιτρέπει να δουλεύουμε με πολλές καρτέλες ανοιχτές σε ένα παράθυρο. Μπορεί να επανερμηνεύσει αρχεία κειμένου σε διάφορες κωδικοποιήσεις χαρακτήρων και να τα μετατρέψει σε ASCII, UTF-8 ή UCS-2. Υποστηρίζει επίσης μακροεντολές και plugins. [20]



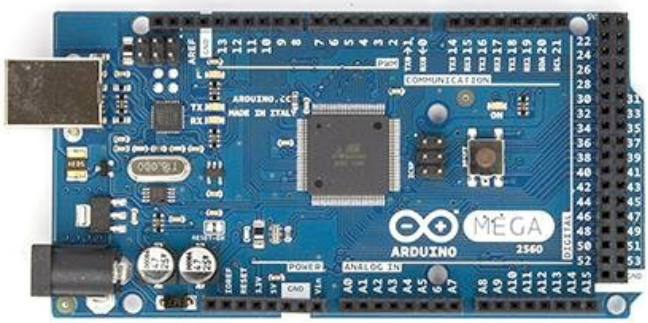
Εικόνα 3.6: Περιβάλλον Notepad++

# Κεφάλαιο 4ο Υλικά Μέρη Κατασκευής

## 4.1 Ηλεκτρολογικά και Ηλεκτρονικά εξαρτήματα

### 4.1.1 Arduino Mega 2560

Το Arduino Mega 2560 είναι μία πλακέτα μικροελεγκτή βασισμένη στον ATmega2560, έναν 8-bit AVR® RISC μικροελεγκτή υψηλής απόδοσης και χαμηλής κατανάλωσης. Έχει 54 ψηφιακά pin εισόδου/εξόδου, εκ των οποίων τα 15 μπορούν να χρησιμοποιηθούν σαν PWM έξοδοι. Επίσης έχει 16 αναλογικά pin εισόδου, 4 UARTs (Universal Asynchronous Receiver/Transmitter ) για σειριακή επικοινωνία με hardware, 6 pin για ICSP (In-Circuit Serial Programming), έναν 16 Mhz κρυσταλλικό ταλαντωτή, μία θύρα USB για την επικοινωνία και τον προγραμματισμό με υπολογιστή, έναν power jack κονέκτορα και ένα κουμπί για reset. [21]




Εικόνα 4.1: Πλακέτα Arduino Mega 2560 [Πηγή]

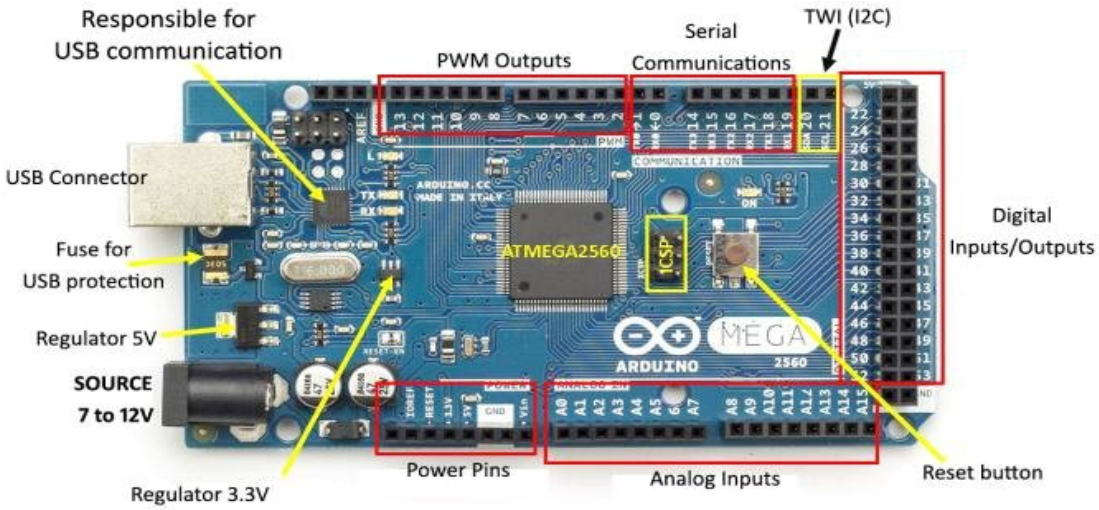
#### 4.1.1.1 Τεχνικά Χαρακτηριστικά

| | |
|---|---|
| **Microcontroller** | ATmega2560 |
| **USB connector** | USB-B |
| **Operating Voltage** | 5V |
| **Input Voltage (recommended)** | 7-12V |
| **Input Voltage (limit)** | 6-20V |
| **Digital I/O Pins** | 54 (of which 15 provide PWM output) |
| **Analog Input Pins** | 16 |
| **DC Current per I/O Pin** | 20 mA |
| **DC Current for 3.3V Pin** | 50 mA |
| **Flash Memory** | 256 KB of which 8 KB used by bootloader |

| | |
|---|---|
| **SRAM** | 8 KB |
| **EEPROM** | 4 KB |
| **Clock Speed Main Processor** | Atmega2560, 16 MHz |
| **USB-Serial Processor** | Atmega16U2, 16 MHz |
| **LED_BUILTIN** | 13 |
| **Communication UART** | Yes, 4 |
| **I2C** | Yes |
| **SPI** | Yes |
| **Length** | 101.52 mm |
| **Width** | 53.3 mm |
| **Weight** | 37 g |

Πίνακας 4.1: Τεχνικά χαρακτηριστικά του arduino mega

4.1.1.2 Ανάλυση της πλακέτας




Εικόνα 4.2: Εξαρτήματα πλακέτας arduino mega 2560 [Πηγή]

Η πλακέτα arduino όπως αναφέρεται και πιο πάνω είναι βασισμένη στον μικροελεγκτή ATmega2560. Ο ATmega2560 σε αυτή την πλακέτα είναι προγραμματισμένος με έναν bootloader που μας επιτρέπει να ανεβάζουμε τον κώδικα χωρίς εξωτερικό προγραμματιστή. Μπορούμε όμως να τον παρακάμψουμε και να ανεβάζουμε τον κώδικα μέσω ICSP με εξωτερικό προγραμματιστή

όπως arduino ISP. [21]

Διαθέτει 256 KB flash memory στην οποία αποθηκεύει τον κώδικα. Από τα 256 KB, τα 8 KB είναι για τον bootloader. Επίσης έχει 8 KB στατικής μνήμης SRAM και 4 KB EEPROM. [21]

Για την τροφοδοσία της πλακέτας μπορούμε να χρησιμοποιήσουμε την usb θύρα, ή το jack connector με κάποια εξωτερικό τροφοδοτικό ή και με μπαταρία 9V της οποίας οι ακροδέκτες θα συνδεθούν στο GND pin και στο Vin pin. Η πηγή τροφοδοσίας επιλέγετε αυτόματα. Η τάση που μπορεί να δεχτεί από εξωτερική τροφοδοσία είναι 6–12V, όμως η συνιστώμενη είναι από 7 έως 12V και αυτό γιατί κάτω από 7V, το 5V pin μπορεί να δίνει λιγότερο από 5V. Από την άλλη αν χρησιμοποιούμε πάνω από 12V τάση, τότε μπορεί να υπερθερμανθεί και να καταστραφεί ο ρυθμιστής τάσης (voltage regulator) που έχει η πλακέτα. [21]

Για την προστασία των usb θυρών του υπολογιστή, υπάρχει στην πλακέτα μια επαναρυθμιζόμενη ασφάλεια για την προστασία από βραχυκυκλώματα και υπερεντάσεις και κόβει αυτόματα την σύνδεση αν στην θύρα usb υπάρχει ρεύμα πάνω από 500mA. [21]

Τα pin τροφοδοσίας του arduino είναι:

- **Vin:** Είναι η είσοδος της τάσης για την τροφοδοσία της πλακέτας από εξωτερική πηγή. Επίσης αν τροφοδοτούμε μέσω του jack connector, τότε μπορούμε να έχουμε πρόσβαση στην τάση του jack από αυτό το pin.
- **5V:** Βγάζει 5V μέσω του ρυθμιστή τάσης.
- **3V3:** Βγάζει 3,3V μέσω του δεύτερου ρυθμιστή τάσης. Το μέγιστο ρεύμα που μπορεί να δώσει είναι 50 mA.
- **GND:** Είναι τα 2 pin της γείωσης.
- **IOREF:** Περιέχει την τάση αναφοράς με την οποία λειτουργεί ο μικροελεγκτής. Ένα shield μπορεί να διαβάσει την τάση αυτή και να επιλέξει την κατάλληλη τάση ή να ενεργοποιήσει του μεταφραστές τάσης στις εξόδους για να δουλέψει με τα 5V ή με τα 3,3V. [21]

Το arduino έχει 16 αναλογικές εισόδους με ονομασίες από A0 έως A15. Παρέχουν 10 bit ανάλυσης (δηλαδή τιμές από 0 έως 1024). Από προεπιλογή μετρούν από Gnd έως 5V, αλλά το όριο των 5V μπορεί να αλλάξει χρησιμοποιώντας τα pin AREF και την συνάρτηση analogReference(). Τα AREF pin βρίσκονται πάνω από τις pwm εξόδους και περιέχουν την τάση αναφοράς των αναλογικών εισόδων. [21]

Επίσης έχει 54 ψηφιακές εισόδους/εξόδους που λειτουργούν στα 5V. Κάθε pin μπορεί να δώσει ή να δεχτεί 20 mA ρεύμα (προτεινόμενο) και έχει και εσωτερική pull-up αντίσταση 20-50 Kohm που

είναι απενεργοποιημένη από προεπιλογή. Το ρεύμα δεν πρέπει να ξεπεράσει τα 40 mA γιατί μπορεί να καταστραφεί ο μικροελεγκτής. [21]

Τα 15 από τα 54 pin μπορούν λειτουργήσουν σαν έξοδοι και να δώσουν 8 bit PWM. Αυτά είναι τα 2 έως 13 και 44 έως 46. Κάποια από τα 54 pin έχουν και κάποιες άλλες ιδιότητες.

Τα pin 0, 1 και 14 έως 19 είναι για σειριακή επικοινωνία (UART) και 4 από αυτά έχουν το όνομα TX (transmite) ενώ τα άλλα 4 έχουν όνομα RX (receive). Τα pin 0 και 1 συνδέονται επίσης με τις αντίστοιχες ακίδες του chip ATmega16U2 που είναι υπεύθυνο για την σειριακή επικοινωνία με το usb. [21]

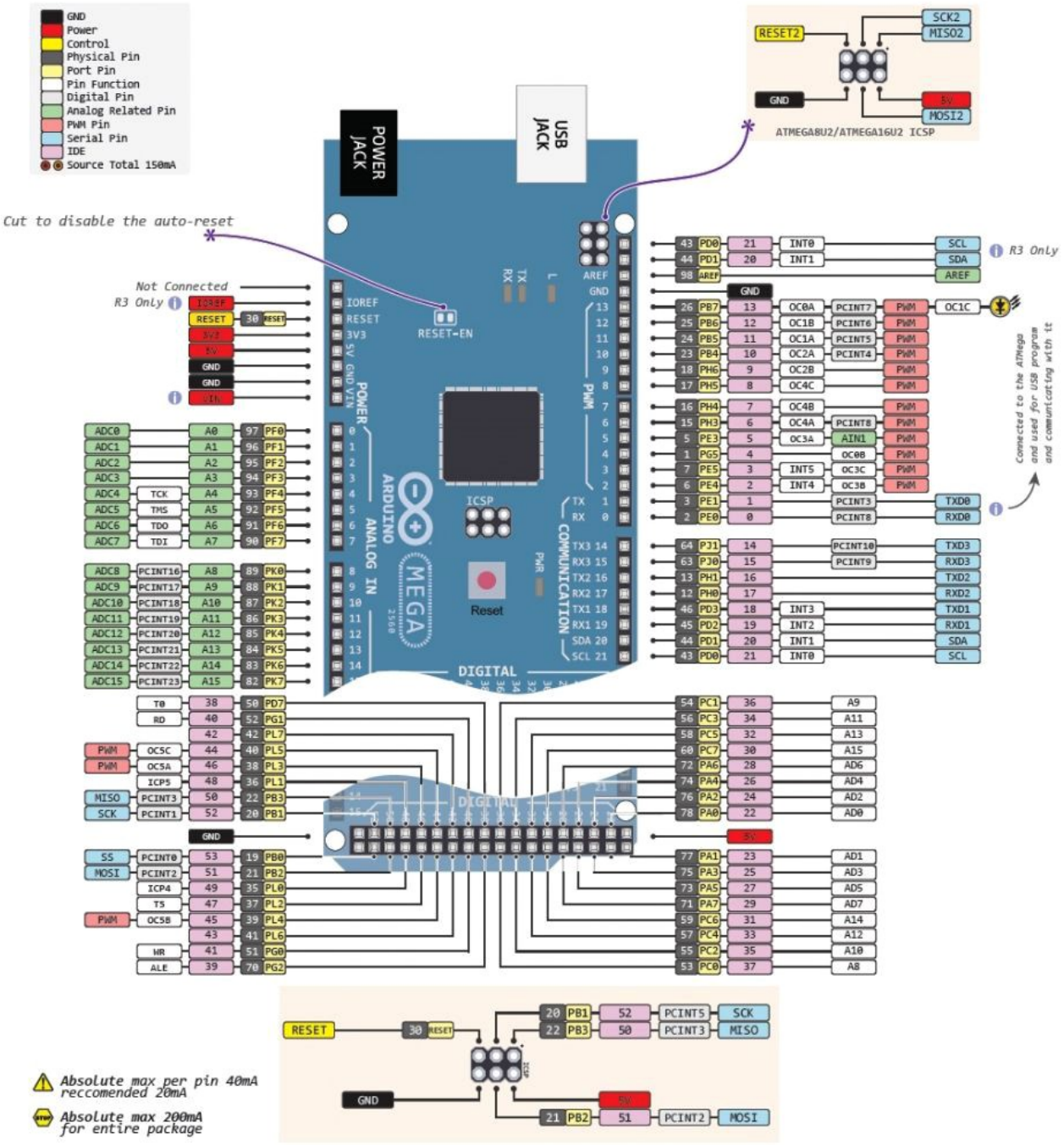
Τα pin 20 (SDA) και 21 (SCL) είναι για επικοινωνία μέσω του πρωτοκόλλου I2C χρησιμοποιώντας την βιβλιοθήκη Wire.

Τα pin 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) υποστηρίζουν επικοινωνία με το SPI πρωτόκολλο χρησιμοποιώντας την βιβλιοθήκη SPI. Τα pin αυτά τα βρίσκουμε επίσης και στο κέντρο της πλακέτας που είναι οι ICSP ακίδες. [21]

Το pin 13 είναι επίσης συνδεδεμένο με το led της πλακέτας. Έτσι όταν η τιμή του pin είναι HIGH, το led θα είναι αναμμένο και όταν η τιμή είναι LOW, θα σβήνει. [21]

Τέλος υπάρχουν κάποια pin που μπορούν να διαμορφωθούν ώστε να ενεργοποιούν μια διακοπή (External Interrupts) σε χαμηλό επίπεδο, μια ανερχόμενη ή καθοδική άκρη ή μια αλλαγή στο επίπεδο. Αυτά είναι τα 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), και 21 (interrupt 2). [21]

Όλα τα pin και οι λειτουργίες τους φαίνονται αναλυτικά στο διάγραμμα της παρακάτω εικόνας.



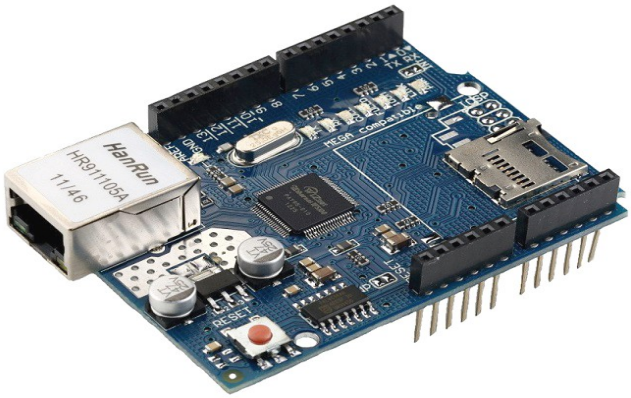

Εικόνα 4.3: Arduino Mega pinout diagram [Πηγή]

4.1.1.3 Πλεονεκτήματα Arduino

- **Χαμηλό κόστος**
- **Απλότητα:** Είναι εύκολο να δημιουργήσει κανείς ένα απλό project ακόμα κι αν είναι άπειρος.
- **Cross-platform:** Μπορούμε να το προγραμματίσουμε στα περισσότερα λογισμικά.

- **Ανοικτού κώδικα υλικό:** Τα σχέδια των πλακετών είναι διαθέσιμα για όλους, έτσι κάποιος έμπειρος μπορεί να τα τροποποιήσει ή να τα επεκτείνει.
- **Ανοικτού κώδικα λογισμικό:** Ο πηγαίος κώδικας είναι ανοιχτός για όλους συμβάλλοντας έτσι στην εξέλιξη της πλατφόρμας και επίσης αναπτύσσονται συνέχεια βιβλιοθήκες από ανθρώπους έμπειρους στον προγραμματισμό. [22]

### 4.1.2 Ethernet Shield

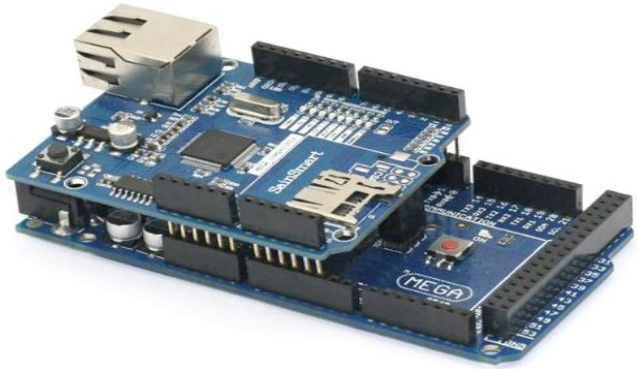



Εικόνα 4.4: Ethernet Shield [Πηγή]

Το Arduino Ethernet Shield είναι μια πλακέτα επέκτασης ανοιχτού υλικού κώδικα για arduino που επιτρέπει την σύνδεση του arduino με το διαδίκτυο.

#### 4.1.2.1 Πληροφορίες πλακέτας

Είναι βασισμένο στο Wiznet W5100 ethernet chip με ενσωματωμένο buffer 16K το οποίο παρέχει ένα δίκτυο ικανό για TCP/IP και για UDP, με ταχύτητα σύνδεσης 10/100Mb. Υποστηρίζει έως και 4 συνδέσεις πρίζας (socket). Η πλακέτα αυτή είναι συμβατή με τις πλακέτες arduino uno και arduino mega και εφαρμόζει πάνω στις πλακέτες αυτές χρησιμοποιώντας τις μακριές ακίδες που εκτείνονται μέσα από το Shield. Έτσι διατηρούνται ανέπαφα τα pin στο πάνω μέρος. Για να μπορεί το πρόγραμμά μας να χρησιμοποιεί τις δυνατότητες του shield θα πρέπει να κάνουμε χρήση της βιβλιοθήκης ethernet.h. [23]



Εικόνα 4.5: Σύνδεση ethernet shield με arduino mega [Πηγή]

Το shield έχει μια τυπική σύνδεση RJ-45 με ενσωματωμένο μετασχηματιστή γραμμής και ενεργοποιημένο PoE (Power over Ethernet), μια τεχνολογία η οποία μας επιτρέπει να μεταφέρουμε ισχύ μαζί με δεδομένα από συνεστραμμένο ζεύγος καλωδίου ethernet cat5 ώστε να μην χρειάζεται άλλο καλώδιο για τροφοδοσία. Τα χαρακτηριστικά της συγκεκριμένης τεχνολογίας στο Shield αυτό είναι:

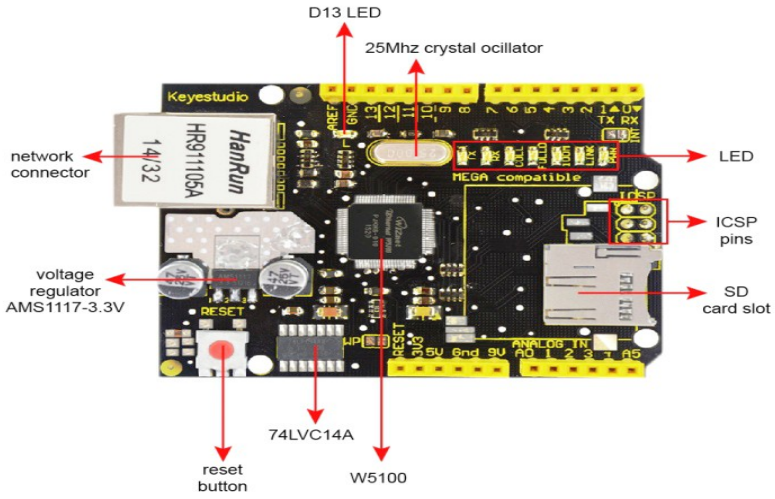
- Συμβατό με IEEE802.3af
- Κυματισμός και θόρυβος χαμηλής απόδοσης (100 mVpp)
- Εύρος τάσης εισόδου 36V έως 57V
- Προστασία από υπερφόρτωση και βραχυκύκλωμα
- Έξοδος 9V
- Μετατροπέας DC/DC υψηλής απόδοσης: τύπου 75% @ 50% φορτίο
- Απομόνωση 1500V (είσοδος προς έξοδο)

Περιέχει επίσης υποδοχή για micro-SD κάρτα στην οποία μπορούμε να αποθηκεύσουμε αρχεία και να τα προβάλουμε στο δίκτυο. Για να είναι δυνατή η ανάγνωση και εγγραφή της SD θα πρέπει να κάνουμε χρήση της βιβλιοθήκης SD. Όταν δουλεύουμε με αυτή την βιβλιοθήκη το SS (Slave Select ) είναι στο pin 4. [23]

Το arduino επικοινωνεί με το chip W5100 και με την κάρτα SD μέσω των ακίδων ICSP χρησιμοποιώντας τον δίαυλο SPI. Στον Mega τα pin αυτά είναι τα 50, 51 και 52. Το pin 10 SS (Slave Select) είναι για την επιλογή του W5100 και το 4 για την κάρτα SD. Αυτά τα pin δεν πρέπει να τα χρησιμοποιούμε σαν εισόδους/εξόδους γενικού σκοπού. Στον Mega εκτός του pin 10 υπάρχει και το 53 που είναι και αυτό SS, αλλά δεν χρησιμοποιείτε ούτε για την επιλογή του W5100 ούτε για την SD. Θα πρέπει όμως να διατηρηθεί ως έξοδος αλλιώς η διεπαφή SPI δεν θα λειτουργεί. [23]

Το chip W5100 και η SD μοιράζονται τον ίδιο δίαυλο SPI δεν μπορούν να δουλεύουν ταυτόχρονα. Αν θέλουμε να τα χρησιμοποιήσουμε και τα δύο θα πρέπει να ληφθεί μέριμνα από τις αντίστοιχες βιβλιοθήκες. [23]

Υπάρχει επίσης και ένα κουμπί reset, το οποίο κάνει reset και το shield αλλά και το arduino.



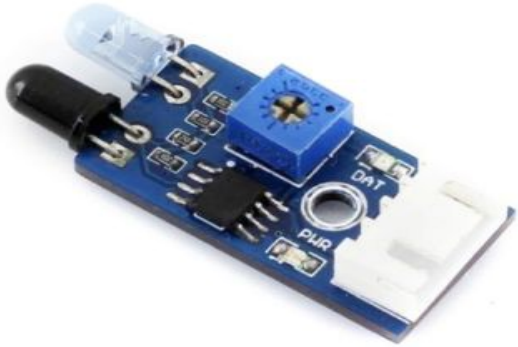

Εικόνα 4.6: Εξαρτήματα Ethernet Shield [Πηγή]

Τέλος στην πλακέτα υπάρχουν κάποια ενδεικτικά led. Αυτά είναι:

- **PWR:** μας δείχνει ότι το arduino και το shield είναι τροφοδοτημένα.
- **LINK:** όταν είναι αναμμένο μας ενημερώνει ότι υπάρχει σύνδεση με δίκτυο και όταν αναβοσβήνει σημαίνει ότι μεταδίδει ή στέλνει δεδομένα.
- **FULLD:** δείχνει ότι έχουμε πλήρως αμφίδρομη σύνδεση με το δίκτυο.
- **100M:** υποδηλώνει ότι είναι συνδεδεμένο σε δίκτυο ταχύτητας 100 Mb/s.
- **RX:** όταν αναβοσβήνει σημαίνει ότι το shield λαμβάνει δεδομένα.
- **TX:** όταν αναβοσβήνει σημαίνει ότι το shield στέλνει δεδομένα.
- **COLL:** αναβοσβήνει όταν υπάρχουν συγκρούσεις δικτύου (network collisions).

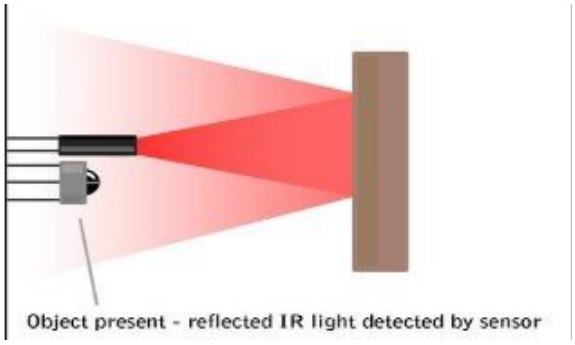
[23]

4.1.3 Waveshare IR Sensor




Εικόνα 4.7: Waveshare υπέρυθρος αισθητήρας απόστασης [Πηγή]

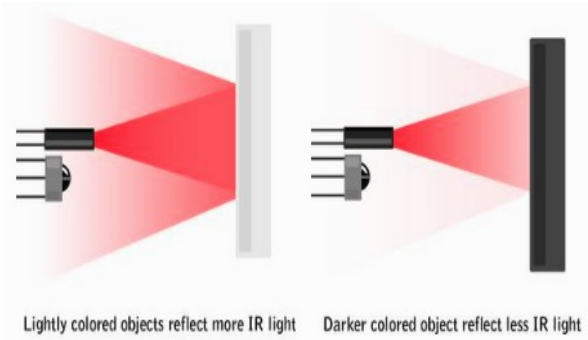
Πρόκειται για έναν αισθητήρα με ψηφιακή και αναλογική έξοδο που ανιχνεύει αντικείμενα στον χώρο και υπολογίζει την απόσταση μεταξύ αυτού και του αντικειμένου. Συγκεκριμένα εκπέμπει μια δέσμη υπερύθρων από το διάφανο led που είναι ένα led υπερύθρων και περιμένει να λάβει την ανακλώμενη δέσμη από την επιφάνεια ενός αντικειμένου στο μαύρο led που είναι ένας ανιχνευτής υπερύθρων ή αλλιώς φωτοδίοδος. Εάν λάβει κάποια δέσμη τότε ανάβει το ένα από τα led που έχει πάνω του και δίνει στην ψηφιακή έξοδο την τιμή 0 ή LOW για να μας δείξει ότι υπάρχει κάποιο αντικείμενο. Επιπλέον ανάλογα την ένταση της ανακλώμενης δέσμης μεταβάλετε η αντίσταση και η τάση εξόδου η οποία συγκρίνετε από τον συγκριτή τάσης LM393 και δίνει στην αναλογική έξοδο κάποια τιμή που αντιστοιχεί στην απόσταση. [24] [25]




Εικόνα 4.8: Τρόπος λειτουργίας του υπέρυθρου αισθητήρα [Πηγή]

Η ευαισθησία του αισθητήρα μπορεί να ρυθμιστεί από το μπλε ποτενσιόμετρο που έχει. Η ευαισθησία του αισθητήρα αλλάζει ανάλογα το χρώμα και το υλικό του αντικειμένου. Για παράδειγμα τα ανοιχτά χρώματα ανακλούν περισσότερη υπέρυθρη ακτινοβολία από τα σκούρα. Έτσι θα μπορεί να τα ανιχνεύει σε μεγαλύτερη απόσταση. Αντίστοιχα το γυαλί απορροφά την υπέρυθρη και δεν μπορεί να το ανιχνεύσει ο αισθητήρας. Αυτό βέβαια εξαρτάτε και από το μήκος

κύματος της ακτινοβολίας του αισθητήρα και από το πάχος του γυαλιού. Για την εργασία αυτή χρησιμοποιήθηκαν 2 τέτοιοι αισθητήρες και συνδέθηκαν μόνο τα digital output. [24] [25]




Εικόνα 4.9: Ευαισθησία του αισθητήρα στα χρώματα [Πηγή]

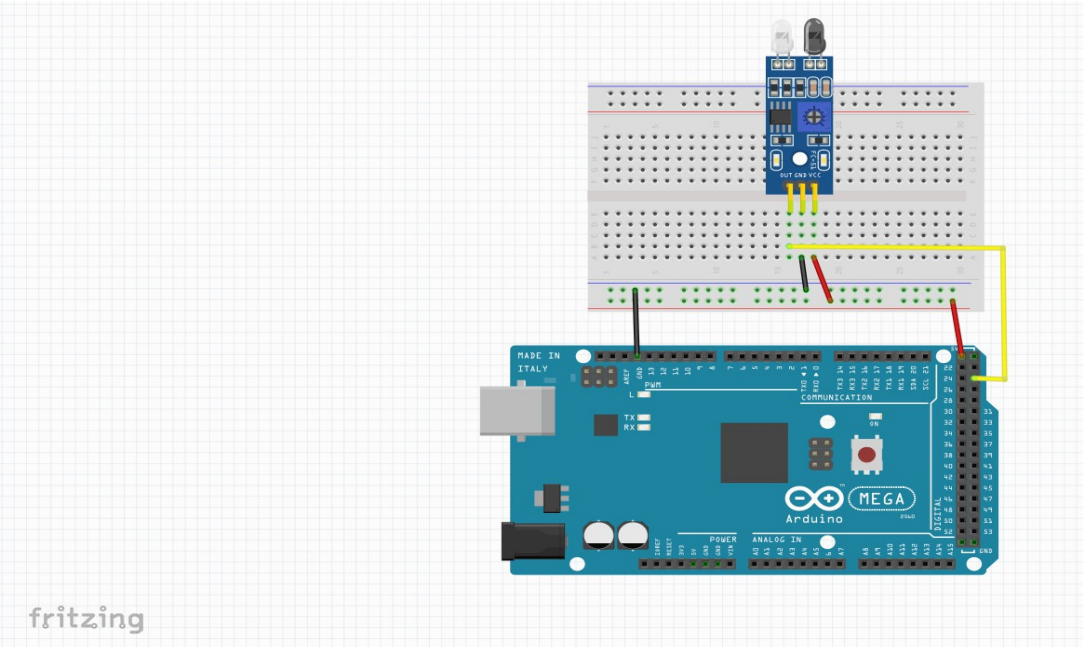
ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τύπος Αισθητήρα:** Απόστασης-Υπερύθρων
- **Τυπική Τάση Εισόδου:** 3.3VDC-5VDC
- **Διασύνδεση:** Αναλογική-Ψηφιακή
- **Ρυθμιζόμενη ευαισθησία**
- **Συγκριτής τάσης μεγάλου εύρους LM393**
- **Μέγεθος οπών τοποθέτησης:** 3mm
- **Εύρος ανίχνευσης:** 2 - 30 cm (ανάλογα με το χρώμα του εμποδίου, πιο μακριά για το λευκό)
- **Γωνία ανίχνευσης:** 35°
- **Προτεινόμενο περιβάλλον:** εσωτερικό, για αποφυγή της ηλιοφάνειας

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **VCC pin** → 3.3V ~ 5V
- **GND pin** → GND
- **AOUT pin** → analog output
- **DOUT pin** → digital output [26]

Στην παρακάτω εικόνα φαίνεται ή σύνδεση του αισθητήρα με τον arduino mega. Ο αισθητήρας της εικόνας δεν έχει το AOUT pin. Ωστόσο αν θέλουμε και αυτό το pin, αρκεί να το συνδέσουμε σε κάποια αναλογική είσοδο από A0 έως A15.




Εικόνα 4.10: Σύνδεση IR αισθητήρα με arduino mega.

Στο παρακάτω παράδειγμα κώδικα διαβάζουμε την ψηφιακή έξοδο του αισθητήρα και αν ανιχνεύσει αντικείμενο ανάβει το led του arduino και γράφει ανάλογο μήνυμα στο serial monitor. Διαβάζουμε επίσης και την αναλογική έξοδο και βλέπουμε τις τιμές όπως επίσης και κάποια μηνύματα στο serial monitor.

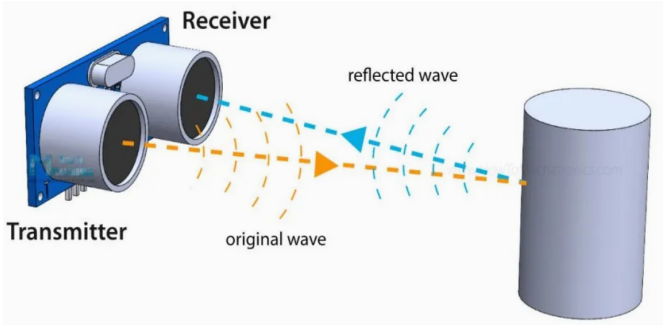
```
const int pinIRd = 25;    //δήλωση του ψηφιακού pin του αισθητήρα
const int pinIRa = A0;   //δήλωση του αναλογικού pin του αισθητήρα, το const δηλώνει
ότι η τιμή της μεταβλητής δεν μπορεί να αλλάξει
const int pinLED = 13;   // δήλωση του ενσωματωμένου led του Arduino
int IRvalueA = 0;        //δήλωση μεταβλητής για την τιμή που θα διαβάζουμε από το
αναλογικό pin
int IRvalueD = 0;       //δήλωση μεταβλητής για την τιμή του ψηφιακού pin
char Str1[15] ;          //δήλωση πίνακα χαρακτήρων χωρίς να τον αρχικοποιήσουμε
char Str2[14] ;

void setup() {
 Serial.begin(9600);          //ανοίγουμε την σειριακή επικοινωνία
 pinMode(pinIRd,INPUT);      //αρχικοποιούμε τα pin
 pinMode(pinIRa,INPUT);
 pinMode(pinLED,OUTPUT);
}

void loop() {
 IRvalueA = analogRead(pinIRa);  //διαβάζει την τιμή του αναλογικού pin (0 έως 1024)
ανάλογα την ρύθμιση της ευαισθησίας που έχουμε κάνει και αποθηκεύετε στην μεταβλητή
IRvalueA
 if (IRvalueA > 500){          // αν η τιμή είναι μεγαλύτερη από 500
  Str2 = " Out of Range";      // αποθηκεύετε η λέξη Out of Range στον Str2
  }else {                 //αλλιώς
   Str2 = "   ";         // ο Str2 θα περιέχει τον χαρακτήρα του κενού
   }
 IRvalueD = digitalRead(pinIRd);     // διαβάζει την τιμή του ψηφιακού pin 0 ή 1 και
αποθηκεύετε στην μεταβλητή IRvalueD
```

```
 if (IRvalueD == LOW) {       // αν είναι LOW δηλαδή 0
  Str1 = " ITEM DETECTED";    // και αποθηκεύετε η λέξη ITEM DETECTED στον  Str1
  } else {                  //αλλιώς
  Str1 = "  ";              // ο Str1 θα περιέχει τον χαρακτήρα του κενού
 }
  Serial.print("Analog Reading= ");  // τυπώνετε στο serial monitor η λέξη Analog
Reading= ,
  Serial.print(IRvalueA);        // στη συνέχεια η τιμή της μεταβλητής IRvalueA που
διαβάστηκε παραπάνω,
 Serial.print(Str2);           // το περιεχόμενο του Str2,
 Serial.print(" , ");          // οι χαρακτήρες του κενού και του (,)
  Serial.print(Str1);           //και το περιεχόμενο του Str1
 delay(1000);              // περιμένει 1 δευτερόλεπτο
}
```

### 4.1.4 HC-SR04 Ultrasonic sensor

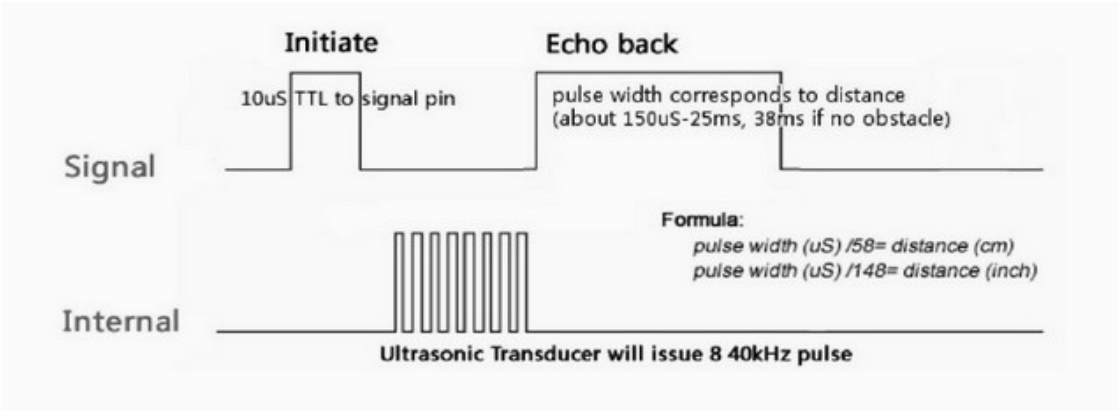

Εικόνα 4.11: Αισθητήρας Υπερήχων HC-SR04 [Πηγή]

Ο HC-SR04 είναι ένας αισθητήρας που χρησιμοποιεί υπερήχους για να προσδιορίσει την απόσταση ενός αντικειμένου με μεγάλη ακρίβεια και σε μεγάλο εύρος από 2 cm έως 400 cm . Όπως φαίνεται και στην εικόνα 4.11 ο αισθητήρας έχει έναν πομπό (transmitter) και έναν δέκτη (receiver) που μοιάζουν με μάτια. Ο πομπός εκπέμπει παλμούς υπερήχων συχνότητας 40 Khz και ο δέκτης ακούει τους παλμούς που εκπέμπονται. [27] [28]




Εικόνα 4.12: Τρόπος λειτουργίας του HC-SR04 [Πηγή]

Για να αρχίσει να μετράει θα πρέπει πρώτα να δοθεί στο Trig pin του αισθητήρα η τιμή HIGH (5V), για τουλάχιστον 10 microseconds (us). Τότε ο αισθητήρας θα εκπέμψει 8 παλμούς υπερήχων συχνότητας 40 Khz και θα περιμένει να δεχτεί κάποιον ανακλώμενο παλμό. Εάν λάβει παλμό τότε θέτει το Echo pin του αισθητήρα σε HIGH (5V) και περιμένει μέχρι να τελειώσει ο παλμός. Μετά θέτει το Echo pin σε LOW (0V). Με τον τρόπο αυτό έχει παράξει έναν παλμό εξόδου, το πλάτος του οποίου το χρησιμοποιούμε για να υπολογίσουμε την απόσταση από το αντικείμενο. Εάν δεν λάβει κανέναν παλμό τότε το Echo pin θα τεθεί σε LOW μετά από 38 millisecond (ms). Άρα ένας παλμός των 38 ms δηλώνει ότι δεν υπάρχει κάποιο αντικείμενο. [27] [28]




Εικόνα 4.13: Διάγραμμα χρονισμού [Πηγή]

Για τον υπολογισμό της απόστασης υπάρχουν διάφοροι τύποι. Ένας είναι αυτός που φαίνεται και στην παραπάνω εικόνα.

- **Απόσταση (cm) = Πλάτος παλμού (uS) / 58**

Ένας άλλος είναι να χρησιμοποιήσουμε την ταχύτητα του ήχου. Η ταχύτητα του ήχου στον αέρα είναι 340 m/sec ή 0,034 cm/uS. Ο τύπος της απόστασης είναι:

- **Απόσταση (cm) = [ ταχύτητα ήχου (cm/uS) * Πλάτος παλμού (uS) ] / 2**

Το 2 το βάζουμε γιατί ο υπέρηχος έχει κάνει 2 διαδρομές, από τον πομπό στο αντικείμενο και από το αντικείμενο πίσω στον δέκτη.

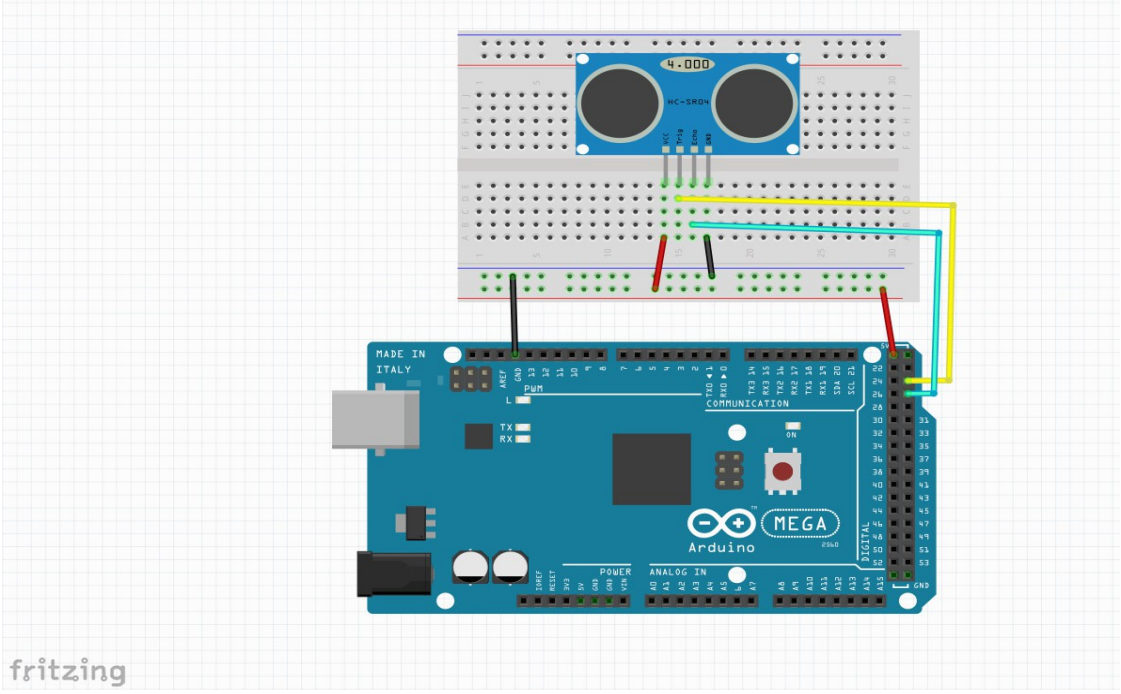
Η διαφορά αυτού του τύπου αισθητήρα με τον αισθητήρα υπερύθρων είναι ότι έχει μεγαλύτερο εύρος ανίχνευσης, δεν επηρεάζετε η λειτουργία του από το φως του ήλιου ούτε από το χρώμα του αντικειμένου. Βέβαια μαλακά ακουστικά υλικά όπως ρούχα μπορεί να είναι δύσκολο να τα εντοπίσει. [28]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τύπος Αισθητήρα:** Απόστασης
- **Τυπική Τάση Εισόδου:** 5VDC
- **Ρεύμα ηρεμίας:** <2 mA
- **Ρεύμα λειτουργίας :** 15 mA
- **Αποτελεσματική γωνία :** <15°
- **Απόσταση εμβέλειας:** 2 cm – 400 cm
- **Ανάλυση:** 0.3 cm
- **Γωνία μέτρησης:** 30°
- **Πλάτος παλμού εισόδου ενεργοποίησης:** 10uS
- **Συχνότητα Υπερήχων:** 40 Khz
- **Διαστάσεις:** 45mm x 20mm x 15mm

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **VCC** → 5V
- **Trig: (Trigger Pulse Input)** → digital output
- **Echo: (Echo Pulse Output )** → digital input
- **GND** → Ground [28]




Εικόνα 4.14: Σύνδεση HC-SR04 αισθητήρα με arduino mega.

Παράδειγμα Κώδικα

```
#define trigpin 25           // καθορίζουμε που έχουν συνδεθεί τα pin
#define echopin 27

void setup(){
Serial.begin(9600);          //ανοίγουμε την σειριακή επικοινωνία
pinMode(trigpin, OUTPUT);    //αρχικοποιούμε τις καταστάσεις των pin
pinMode(echopin, INPUT);
}

void loop(){
int duration, distance;  //δηλώνουμε τις μεταβλητές που θα έχουν την πληροφορία για
την διάρκεια και την απόσταση
digitalWrite(trigpin, HIGH);  // δίνει στο Trigger pin την τιμή HIGH και
delayMicroseconds(10);     //περιμένει 10 uS για να ξεκινήσει να στείλει 8 παλμούς
υπερήχων
digitalWrite(trigpin, LOW);   // δίνει στο Trigger pin την τιμή LOW

duration = pulseIn(echopin,HIGH); // διαβάζει έναν παλμό HIGH από το echopin, δηλαδή
αυτόν που επιστρέφει από το αντικείμενο, μέχρι αυτός να γίνει LOW και αποθηκεύει την
διάρκεια στην μεταβλητή duration

distance = ( duration *0.034 ) / 2;    // υπολογίζει την απόσταση
Serial.print("Η απόσταση είναι: ");    // εκτυπώνει στο serial monitor την λέξη μέσα
στα εισαγωγικά
Serial.print(distance);    // εκτυπώνει την τιμή της μεταβλητής distance που υπολόγισε
πιο πάνω
Serial.print(" cm");     //εκτυπώνει την λέξη στα μέσα στα εισαγωγικά

if (distance >= 395) {  // ελέγχει αν η τιμή της απόστασης είναι μεγαλύτερη ή ίση με
το 395 και
Serial.println("Out of Range"); //αν ισχύει η συνθήκη εκτυπώνει την λέξη μέσα στα
εισαγωγικά
}
```

4.1.5 DS18B20 Temperature sensor

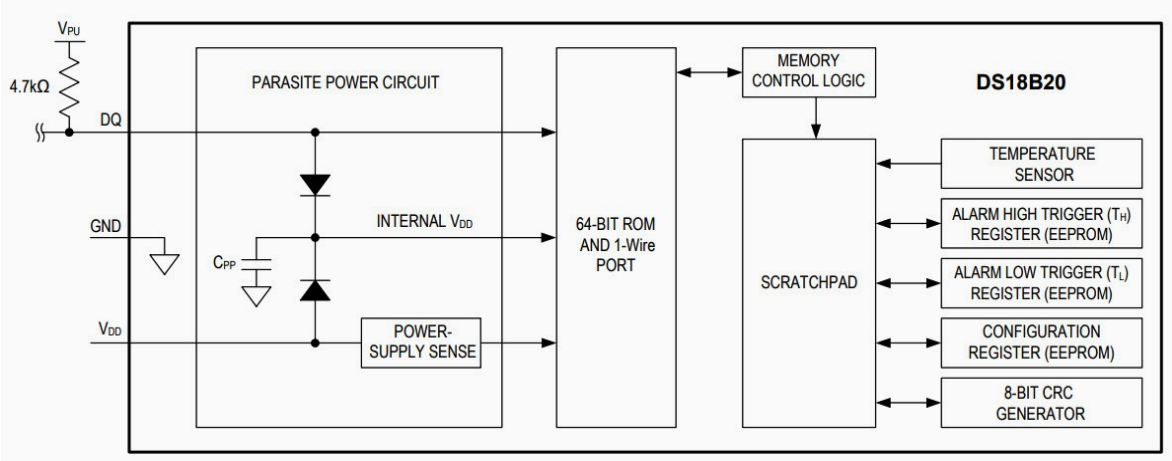
Εικόνα 4.15: Αισθητήρας θερμοκρασίας DS18B20 [Πηγή]

Ο DS18B20 είναι ένας ψηφιακός αισθητήρας θερμοκρασίας με εύρος μέτρησης από -55°C έως 125°C και ακρίβεια ±0,5°C στην περιοχή των -10°C έως 85°C. Ο συγκεκριμένος βρίσκετε μέσα σε μεταλλική θήκη που τον καθιστά αδιάβροχο. Επικοινωνεί με το πρωτόκολλο διαύλου one-wire το οποίο επιτρέπει την επικοινωνία με τον μικροεπεξεργαστή με μία μόνο γραμμή δεδομένων.

Επιπλέον μπορεί να τροφοδοτηθεί από τη γραμμή δεδομένων χωρίς να χρειάζεται εξωτερικό τροφοδοτικό. Κάθε αισθητήρας DS18B20 έχει έναν μοναδικό σειριακό κωδικό 64 bit και αυτό επιτρέπει να λειτουργούν πολλοί τέτοιοι αισθητήρες στον ίδιο δίαυλο one-wire. [29] [30]


Εικόνα 4.16: Ο DS18B20 και τα pin του. [Πηγή]

Ο DS18B20 είναι ένας direct to digital αισθητήρας και η βασική του λειτουργία είναι μα μετατρέπει την αναλογική τιμή της θερμοκρασίας σε ψηφιακή. Διαθέτει όμως και λειτουργία συναγερμού με μη πτητικά High και Low σημεία ενεργοποίησης που τα προγραμματίζει ο χρήστης. Η ανάλυση του αισθητήρα είναι 12 bit από προεπιλογή αλλά μπορεί να τον αλλάξουμε σε 9, 10 και 11 bit. Οι αριθμοί της ανάλυσης αντιστοιχούν σε αυξήσεις που μπορεί να αντιληφθεί ο αισθητήρας. Έτσι τα 9 bit αντιστοιχούν σε αύξηση 0,5°C, τα 10 σε 0,25°C, τα 11 σε 0,125°C και τα 12 σε 0,0625°C. Το μπλοκ διάγραμμα του φαίνεται στην παρακάτω εικόνα. [29] [30]




Εικόνα 4.17: DS18B20 block diagram [Πηγή]

Στη μνήμη ROM αποθηκεύετε ο σειριακός κωδικός του αισθητήρα. Η μνήμη Scratchpad είναι μια υψηλής ταχύτητας μνήμη στην οποία αποθηκεύονται προσωρινά δεδομένα. Εκεί βρίσκεται ο καταχωρητής θερμοκρασίας 2 byte που αποθηκεύει το ψηφιακό αποτέλεσμα του αισθητήρα. Η

Scratchpad παρέχει επίσης πρόσβαση και σε άλλους 3 καταχωρητές, τον Alarm High Trigger (TH) Reggister του ενός byte, τον Alarm Low Trigger (TL) Reggister του ενός byte και τον Configuration Register ο οποίος μας επιτρέπει να ορίσουμε την ανάλυση. Όπως αναφέρεται παραπάνω ο αισθητήρας μπορεί να τροφοδοτηθεί από την γραμμή δεδομένων δηλαδή το data pin και συγκεκριμένα από την αντίσταση pull up όταν το data pin είναι σε κατάσταση HIGH. Το σήμα αυτό φορτίζει τον εσωτερικό πυκνωτή (CPP), ο οποίος όταν το data pin είναι LOW τροφοδοτεί τη συσκευή. [29]

Για να κάνει μια μέτρηση θερμοκρασίας και να την μετατρέψει σε ψηφιακή, το DS18B20 δίνει μια εντολή Convert T [44h]. Αφού κάνει την μετατροπή, τα δεδομένα θερμοκρασίας βαθμονομούνται σε βαθμούς Κελσίου ως προσημασμένο συμπλήρωμα ως προς 2 των 16 bit και αποθηκεύονται στον καταχωρητή θερμοκρασίας. Το bit πρόσημου (S) δείχνει αν η θερμοκρασία είναι θετική (S=0) ή αρνητική (S=1). [29]

| TEMPERATURE (°C) | DIGITAL OUTPUT (BINARY) | DIGITAL OUTPUT (HEX) |
|---|---|---|
| +125 | 0000 0111 1101 0000 | 07D0h |
| +85* | 0000 0101 0101 0000 | 0550h |
| +25.0625 | 0000 0001 1001 0001 | 0191h |
| +10.125 | 0000 0000 1010 0010 | 00A2h |
| +0.5 | 0000 0000 0000 1000 | 0008h |
| 0 | 0000 0000 0000 0000 | 0000h |
| -0.5 | 1111 1111 1111 1000 | FFF8h |
| -10.125 | 1111 1111 0101 1110 | FF5Eh |
| -25.0625 | 1111 1110 0110 1111 | FE6Fh |
| -55 | 1111 1100 1001 0000 | FC90h |

*The power-on reset value of the temperature register is +85°C.

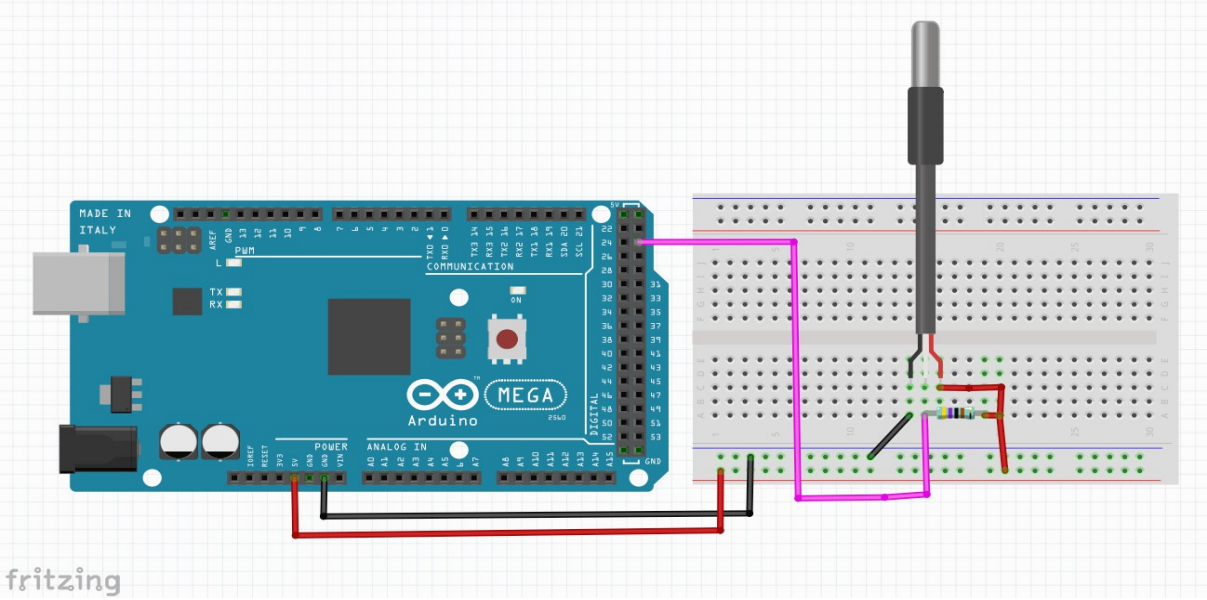
Πίνακας 4.2: Αντιστοιχία θερμοκρασιών / δεδομένων [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τύπος Αισθητήρα:** Θερμοκρασίας
- **Τυπική Τάση Εισόδου:** 3VDC - 3.3VDC - 5VDC - 5.5VDC
- **Ρεύμα Λειτουργίας:** 1.5mA
- **Διασύνδεση:** Ψηφιακή
- **Πρωτόκολλο Επικοινωνίας:** Dallas 1-Wire
- **Περιοχή Μέτρησης Θερμοκρασίας:** -55...125°C, (–67°F to +257°F)
- **Ακρίβεια Μέτρησης:** ±0.5°C (–10°C to +85°C)
- **Συνήθως για την σύνδεση με μικροελεγκτή απαιτείται μια αντίσταση 4.7Kohm.**

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **Gnd Pin** → Ground
- **Vcc Pin** → 3,3V ή 5V
- **Data Pin** → digital pin




Εικόνα 4.18: Σύνδεση DS18B20 με arduino mega

Για την επικοινωνία με τον arduino και τον προγραμματισμό του χρειάζονται 2 βιβλιοθήκες, η (OneWire.h) και η (DallasTemperature.h) .

Παράδειγμα Κώδικα

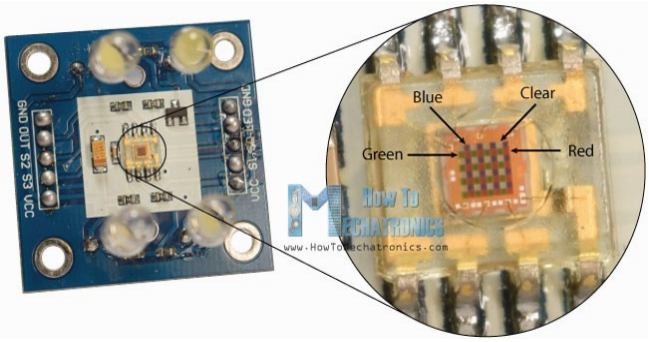
```
#include <OneWire.h>          // συμπερίληψη της βιβλιοθήκης OneWire και
#include <DallasTemperature.h>  // της DallasTemperature
#define ONE_WIRE_BUS 25 // καθορίζουμε την σύνδεση του data pin στον arduino αλλά του έχουμε δώσει όνομα ONE_WIRE_BUS
OneWire oneWire(ONE_WIRE_BUS);  // setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature Ics)
DallasTemperature sensors(&oneWire);  // Pass our oneWire reference to Dallas Temperature.

void setup() {
  Serial.begin(9600);
   sensors.begin();   // Εεκινά την βιβλιοθήκη
}
void loop() {
  sensors.requestTemperatures(); // στέλνει την εντολή για να πάρει θερμοκρασία
   float tempC = sensors.getTempCByIndex(0); //δηλώνουμε μια μεταβλητή τύπου float στην οποία θα αποθηκευτεί ο αριθμός της θερμοκρασίας που θα λάβουμε με την εντολή sensors.getTempCByIndex(0)
  if(tempC != DEVICE_DISCONNECTED_C)  // ελέγχει αν το διάβασμα της θερμοκρασίας ήταν επιτυχημένο
  {
```

```
    Serial.print("Temperature is: ");  // τυπώνει στο serial monitor την λέξη
στην παρένθεση
    Serial.println(tempC);             // και την θερμοκρασία
  }
  else {                          // αλλιώς αν δεν είναι επιτυχημένο το διάβασμα
    Serial.println("Error: Could not read temperature data"); // τυπώνει την λέξη
  }
  delay(2000);         // περιμένει 2 δευτερόλεπτα
}
```

### 4.1.6 TCS230 TCS3200 Color recognition sensor

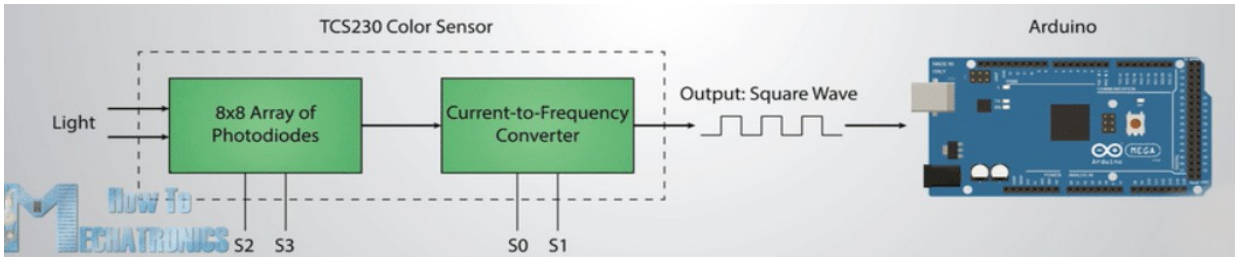
Εικόνα 4.19: TCS230 TCS3200 αισθητήρας [Πηγή]

Πρόκειται για έναν αισθητήρα ο οποίος αναγνωρίζει το χρώμα ενός αντικειμένου από το φως που ανακλάτε σε αυτόν. Έχει πάνω του 4 λευκά led με τα οποία μπορεί να φωτίσει το αντικείμενο ώστε να μπορεί να αναγνωρίζει το χρώμα του σε κάθε συνθήκη φωτισμού ακόμα και στο σκοτάδι. Στο κέντρο έχει μια συστοιχία φωτοδιόδων και συγκεκριμένα 64 σε διάταξη 8x8. Από αυτές 16 έχουν κόκκινο φίλτρο, 16 έχουν μπλε φίλτρο, 16 έχουν πράσινο φίλτρο και οι άλλες 16 δεν έχουν φίλτρο. [31]




Εικόνα 4.20: Συστοιχία φωτοδιόδων [Πηγή]

Ο τρόπος που λειτουργεί είναι ότι μετατρέπει την ένταση του φωτός που πέφτει πάνω του, σε συχνότητα. Στην έξοδό του παίρνουμε μια κυματομορφή πού είναι τετραγωνικού παλμού με 50% κύκλο λειτουργίας (duty cycle). Με τον μικροελεγκτή μπορούμε να μετρήσουμε την περίοδο του παλμού ώστε να πάρουμε την συχνότητα. Η συχνότητα όπως και ο τετραγωνικός παλμός μπορεί να διαφέρει ανάλογα το χρώμα και την ένταση του φωτός. [31] [32]



Εικόνα 4.21: Τρόπος λειτουργίας του αισθητήρα [Πηγή]

Ο αισθητήρας έχει 10 pin από τα οποία τα Vcc και Gnd που είναι για την τροφοδοσία υπάρχουν 2 φορές.

- Τα pin **S0/S1** ελέγχουν την κλίμακα της συχνότητας εξόδου.
- Τα pin **S2/S3** ελέγχουν πιο σετ φωτοδίοδων θα ενεργοποιηθούν για να μετρήσουν (κόκκινο, πράσινο, μπλε, διαφανές).
- Το pin **Out** είναι η έξοδος όπου λαμβάνουμε το σήμα εξόδου, δηλαδή τον τετραγωνικό παλμό.
- Το **p**in **Led** είναι για τον έλεγχο των led που έχει πάνω του ο αισθητήρας.

Υπάρχει μια σχέση μεταξύ εξόδου και έντασης του φωτός αφού η συχνότητα της εξόδου είναι ανάλογη του ρεύματος που διαρρέει τις φωτοδιόδους. Το εύρος της τυπική συχνότητας εξόδου είναι 2HZ~500KHZ. Μπορούμε να ορίσουμε διαφορετικό συντελεστή κλιμάκωσης της συχνότητας εξόδου δίνοντας διαφορετικούς συνδυασμούς τιμών στα pin S0 και S1. [31] [32]

| S0 | S1 | Output Frequency (fo) |
|---|---|---|
| L | L | Power down |
| L | H | 2% |
| H | L | 20% |
| H | H | 100% |

Πίνακας 4.3: Συνδυασμοί pin για friquency scaling [Πηγή]

Επειδή ο αισθητήρας έχει μία έξοδο, δεν μπορεί να κάνει μέτρηση ταυτόχρονα και με τα 4 κανάλια (κόκκινο, πράσινο, μπλε, διαφανές). Έτσι πρέπει να επιλέγουμε ένα φίλτρο χρώματος την φορά, να πάρουμε το αποτέλεσμα στην έξοδο και μετά να επιλέξουμε άλλο. Κάθε φίλτρο επιτρέπει σε ένα συγκεκριμένο χρώμα να περάσει και να διαβάσει την έντασή του. Για παράδειγμα όταν επιλέγουμε το μπλε φίλτρο, μόνο το μπλε χρώμα επιτρέπετε να περάσει και να διαβαστεί η έντασή του. Για να επιλέξουμε ποιο φίλτρο θα κάνει μέτρηση πρέπει να δώσουμε διαφορετικούς συνδυασμούς τιμών στα pin S2 και S3. [31] [32]

| S2 | S3 | Photodiode type |
|---|---|---|
| L | L | Red |
| L | H | Blue |
| H | L | Clear |
| H | H | Green |

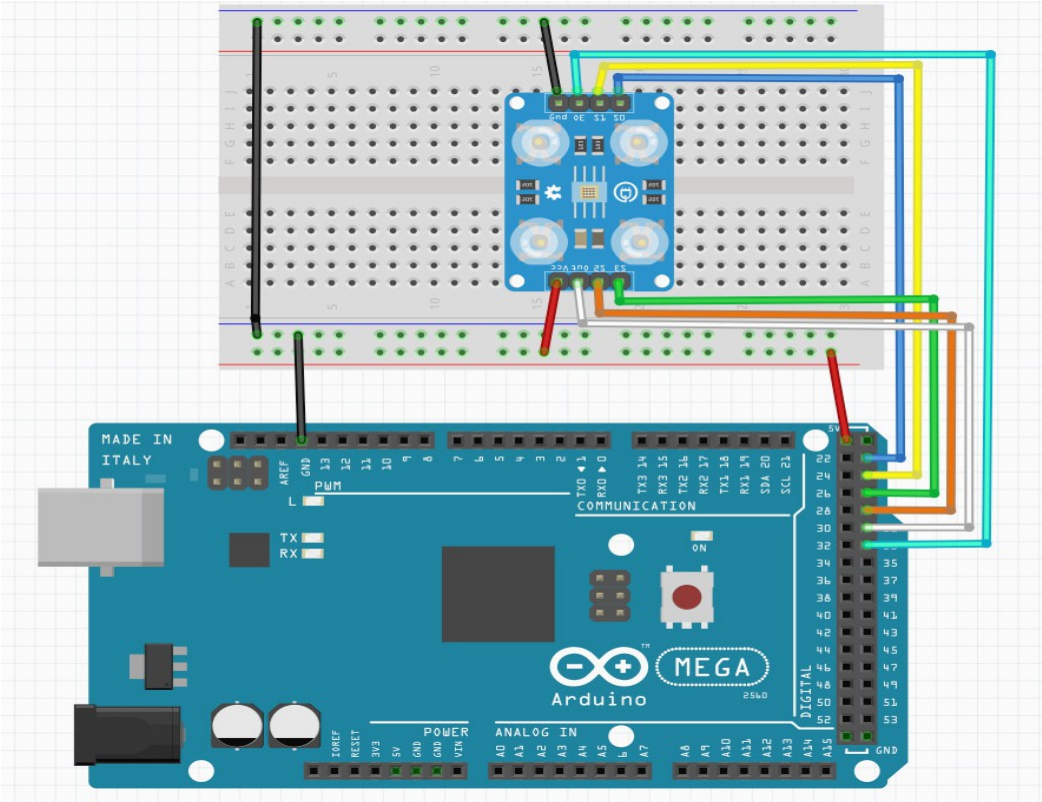
Πίνακας 4.4: Συνδυασμοί pin για επιλογή φίλτρου [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τάση εισόδου:** 3V ~ 5V.
- **Καλύτερη εμβέλεια ανίχνευσης:** 10 mm.
- **Μετατροπή της έντασης του φωτός σε συχνότητα υψηλής ανάλυσης.**
- **Προγραμματιζόμενο χρώμα και πλήρης κλίμακας συχνότητα εξόδου.**
- **Επικοινωνία απευθείας με μικροελεγκτή.**
- **Μέγεθος PCB (Μ x Π):** Περίπου 1,2 x 0,95 ίντσες.
- **Βάρος:** 4g

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **Vcc pin** → 3V ή 5V
- **Gnd pin** → Ground
- **S0, S1, S2, S3, Led pins** → Digital Outputs
- **Out pin** → Digital Input




Εικόνα 4.22: Κύκλωμα σύνδεσης TCS3200 με arduino mega

Παράδειγμα κώδικα

```
#define S0 23    //καθορίζουμε τα pin του αισθητήρα και που έχουν συνδεθεί
#define S1 24
#define S2 29
#define S3 27
#define sensorOut 31
#define led 33
int redfrequency = 0;  //δήλωση μεταβλητής στην οποία θα αποθηκευτεί η συχνότητα που
θα λάβουμε από το κόκκινο φίλτρο.
int greenfrequency = 0; // δήλωση μεταβλητής για την συχνότητα του πράσινου φίλτρου
int bluefrequency = 0;  // δήλωση μεταβλητής για την συχνότητα του μπλε φίλτρου

void setup() {
  pinMode(S0, OUTPUT);  //αρχικοποιούμε τις καταστάσεις των pin του αισθητήρα
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(sensorOut, INPUT);
  // Θέτουμε το frequency-scaling στο 20% με τον ακόλουθο συνδυασμό τιμών
  digitalWrite(S0,HIGH);
  digitalWrite(S1,LOW);
  Serial.begin(9600);  // Ανοίγουμε την σειριακή επικοινωνία
}
void loop() {
  digitalWrite (led,HIGH); // εντολή να ανάψουν τα led του αισθητήρα
//Επιλέγουμε την φωτοδίοδο κόκκινου φίλτρου να διαβαστεί με τον ακόλουθο συνδυασμό τιμών
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
```

```
  // διαβάζουμε την συχνότητα εξόδου για το κόκκινο
  redfrequency = pulseIn(sensorOut, LOW);
  delay(100);
//Επιλέγουμε την φωτοδίοδο πράσινου φίλτρου να διαβαστεί με τον ακόλουθο συνδυασμό τιμών
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // διαβάζουμε την συχνότητα εξόδου για το πράσινο
  greenfrequency = pulseIn(sensorOut, LOW);
  delay(100);
// Επιλέγουμε την φωτοδίοδο μπλε φίλτρου να διαβαστεί με τον ακόλουθο συνδυασμό τιμών
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // διαβάζουμε την συχνότητα εξόδου για το μπλε
  bluefrequency = pulseIn(sensorOut, LOW);
  // Εκτυπώνουμε τις τιμές στο serial monitor
  Serial.print("R= ");
  Serial.print(redfrequency);
  Serial.print("  ");
  Serial.print("G= ");
  Serial.print(greenfrequency);
  Serial.print("  ");
  Serial.print("B= ");
  Serial.print(bluefrequency);/
  Serial.println("  ");
  delay(1000);
}
```

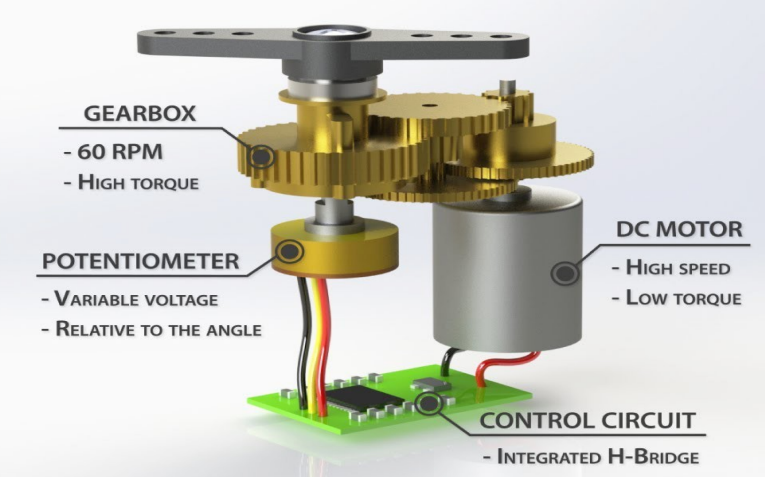
### 4.1.7 Waveshare MG996R Servo motor standard

Εικόνα 4.23: Σερβοκινητήρας MG996R [Πηγή]

Ο Waveshare MG996R είναι ένας σερβοκινητήρας υψηλής ροπής με μεταλλικά γρανάζια και εύρος κίνησης από 0° έως 180°.

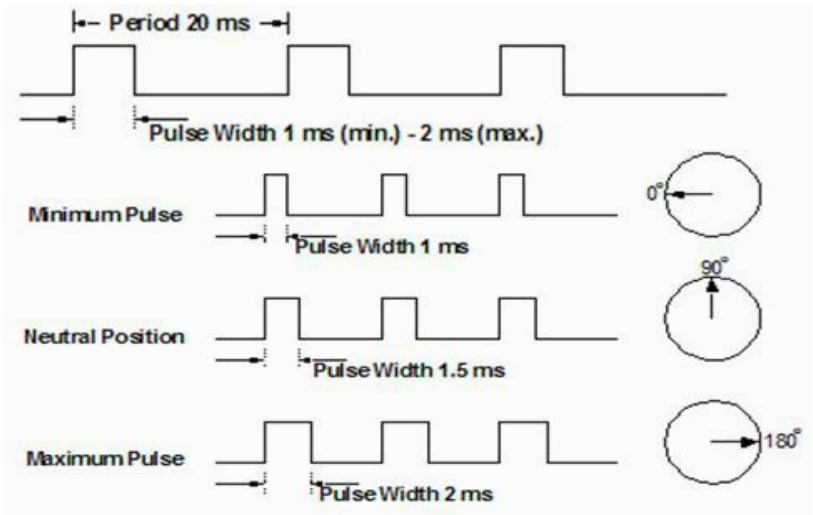
Ένας σερβοκινητήρας είναι ένα σύστημα που αποτελείτε από έναν DC κινητήρα, μία πλακέτα ελέγχου, έναν μηχανισμό ανάδρασης που συνήθως είναι ένα ποτενσιόμετρο και ένα κιβώτιο ταχυτήτων. Ο άξονας του DC κινητήρα είναι συνδεδεμένος με το κιβώτιο ταχυτήτων το οποίο αυξάνει την ροπή στον άξονα εξόδου. Ο κινητήρας οδηγείτε από την πλακέτα ελέγχου η οποία ερμηνεύει τα σήματα που στέλνει ο μικροελεγκτής. Ο ρόλος του ποτενσιόμετρου είναι για να γνωρίζει το κύκλωμα ελέγχου την θέση του άξονα εξόδου, ώστε να δίνει τις κατάλληλες εντολές να

διορθώνει την θέση και να υπάρχει ακρίβεια. [33]




Εικόνα 4.24: Το εσωτερικό του σερβοκινητήρα [Πηγή]

Ο σερβοκινητήρας έχει 3 ακροδέκτες. Από αυτούς οι 2 είναι για την τροφοδοσία (Vcc και Gnd) και ο τρίτος (PWM) είναι για τον έλεγχο θέσης. Για να κινηθεί σε κάποια θέση (γωνία), θα πρέπει να δώσουμε ένα σήμα ελέγχου PWM στο pin (PWM). Συγκεκριμένα το κύκλωμα ελέγχου του σερβοκινητήρα περιμένει να δει έναν παλμό ανά 20ms. Το πλάτος αυτού του παλμού καθορίζει την θέση και είναι συνήθως μεταξύ 1ms και 2ms. [33]




Εικόνα 4.25: Έλεγχος θέσης σερβοκινητήρα ανάλογα το πλάτος του παλμού [Πηγή]

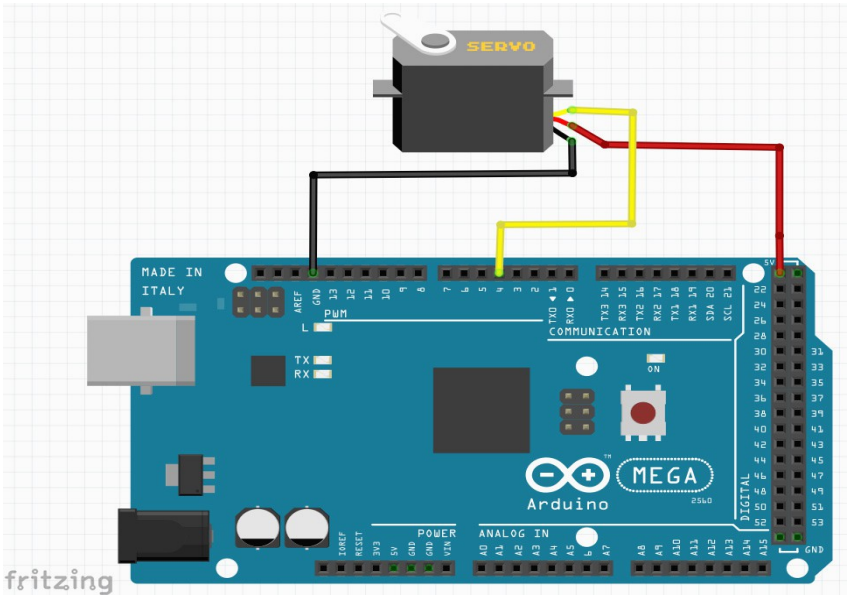
Για την κατασκευή αυτής της εργασίας και συγκεκριμένα για τον ρομποτικό βραχίονα χρησιμοποιήθηκαν 5 τέτοιοι σερβοκινητήρες.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Servo Standar:** 25T (Futaba/Feetech)
- **Μέγεθος Servo:** Standard
- **Περιστροφή:** 180°
- **Ροπή:** 9kg/cm(4.8V) 11kg/cm(6V)
- **Ταχύτητα:** 0.19s/60°(4.8V) 0.18s/60°(6V)
- **Τύπος Γραναζιών:** Μεταλλικά
- **Τάση Λειτουργίας:** 4.8 ~ 6V
- **Ρεύμα Λειτουργίας:** 500 mA –900 mA (6V)
- **Ρεύμα Στασιμότητας:** 2.5 A (6V)
- **Νεκρή Ζώνη:** 5us
- **Διπλό ρουλεμάν σταθερό και ανθεκτικό σε κραδασμούς**
- **Βάρος:** 55g
- **Διαστάσεις:** 40.7mm × 19.7mm × 42.9mm [34]

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **Vcc (κόκκινο)** → 5V ή 6V (από εξωτερικό τροφοδοτικό)
- **Gnd (μαύρο)** → Ground
- **PWM (πορτοκαλί)** → PWM Digital Output



Εικόνα 4.26: Κύκλωμα σύνδεσης σερβοκινητήρα με arduino mega

Για τον εύκολο έλεγχο θέσης και ταχύτητας με το arduino χρησιμοποιούμε βιβλιοθήκες. Η πιο γνωστή είναι η (Servo.h) , που υπάρχει εγκατεστημένη στο Arduino IDE. Άλλη μία είναι η (VarSpeedServo.h) η οποία είναι μια τροποποίηση της Servo.h και μπορούμε να την κατεβάσουμε από το GitHub. Η συγκεκριμένη βιβλιοθήκη επιτρέπει ταυτόχρονη, ασύγχρονη κίνηση μέχρι 8 σερβοκινητήρες, ρύθμιση της ταχύτητας τους και εντολή για τον αν ένας σερβοκινητήρας θα περιμένει έναν άλλο να ολοκληρώσει την κίνησή του πριν κάνει την δική του.

Για τον ρομποτικό βραχίονα αυτής της εργασίας χρησιμοποιήθηκε η βιβλιοθήκη (VarSpeedServo.h) για τους λόγους που αναφέρθηκαν.

Παράδειγμα κώδικα με χρήση της βιβλιοθήκης (Servo.h)

```
#include <Servo.h> //συμπερίληψη της βιβλιοθήκης Servo.h
Servo servo1;   //δήλωση μεταβλητής τύπου Servo με όνομα servo1
                //Ο τύπος μεταβλητής Servo είναι της βιβλιοθήκης
int pos1 = 90;  //δήλωση μεταβλητής για την θέση του σερβοκινητήρα με τιμή 90°

void setup() {
 servo1.attach(4);     //σύνδεση της μεταβλητής servo1 στο pin 4
 servo1.write(pos1);  //αρχικοποίηση της θέσης του σερβοκινητήρα, γράφοντας σε αυτόν
την τιμή της μεταβλητής pos1
}
void loop() {
 delay(4000);       //περιμένει 4 δευτερόλεπτα
 servo1.write(0);  //γράφει στον σερβοκινητήρα να πάει στην θέση 0°
 delay(1000);      //περιμένει 1 δευτερόλεπτο τον σερβοκινητήρα να φτάσει στην θέση του
 servo1.write(180); //γράφει στον σερβοκινητήρα να πάει στις 180°
 delay(1500);    //περιμένει 1,5 δευτερόλεπτο τον σερβοκινητήρα να φτάσει στην θέση του
 servo1.write(90);  //γράφει στον σερβοκινητήρα να πάει στις 90°
}
```

Παράδειγμα κώδικα με χρήση της βιβλιοθήκης (VarSpeedServo.h) και χρήση 2 σερβοκινητήρων για ασύγχρονη κίνηση.

```
#include <VarSpeedServo.h>  //συμπερίληψη της βιβλιοθήκης VarSpeedServo.h
VarSpeedServo servo1;  //δήλωση μεταβλητής τύπου VarSpeedServo με όνομα servo1
VarSpeedServo servo2;  //δήλωση μεταβλητής τύπου VarSpeedServo με όνομα servo2
                       //ο τύπος της μεταβλητής VarSpeedServo είναι της βιβλιοθήκης
void setup() {
 servo1.attach(4);  //σύνδεση της μεταβλητής servo1 στο pin4
 servo2.attach(5);  //σύνδεση της μεταβλητής servo2 στο pin5
//Αρχικοποίηση των θέσεων των σερβοκινητήρων
 servo1.write(0,40,false); //γράφει στον servo1 να πάει στις 0°, με ταχύτητα 40 και
να μην περιμένει να ολοκληρωθεί η κίνησή του
                           //δηλαδή όσο εκτελείτε η κίνησή του, εκτελείτε ταυτόχρονα
και η επόμενη κίνηση του servo2
 servo2.write(90,80,true); //γράφει στον servo2 να πάει στις 90°, με ταχύτητα 80 και
να περιμένει να ολοκληρωθεί η κίνησή του
                           //δηλαδή μόνο όταν εκτελεστεί η κίνησή του θα προχωρήσει
σε επόμενη κίνηση.
}
void loop() {
 delay(4000);   //περιμένει 4 δευτερόλεπτα
//Ταυτόχρονη και ασύγχρονη κίνηση των σερβοκινητήρων
```

```
 servo1.write(180,200,false); //γράφει στον servo1 να πάει στις 180°, με ταχύτητα
200 και να μην περιμένει να ολοκληρωθεί η κίνησή του
 servo2.write(180,20,false); //γράφει στον servo2 να πάει στις 1800°, με ταχύτητα 20
και να μην περιμένει να ολοκληρωθεί η κίνησή του
 delay(2000);   //περιμένει 2 δευτερόλεπτα
//Κίνηση ενός σερβοκινητήρα τη φορά
 servo1.write(90,50,true); //γράφει στον servo1 να πάει στις 90°, με ταχύτητα 50 και
να περιμένει να ολοκληρωθεί η κίνησή του
 servo2.write(0,150,true); //γράφει στον servo1 να πάει στις 0°, με ταχύτητα 150 και
να περιμένει να ολοκληρωθεί η κίνησή του
}
```

Ο αριθμός της ταχύτητας είναι από 0 έως 255, όπου 0 είναι η μέγιστη ταχύτητα, 1 είναι η ελάχιστη και 255 είναι επίσης η μέγιστη.

### 4.1.8 DC Gearmotor 9-12V 100rpm

Εικόνα 4.27: DC κινητήρας [Πηγή]

Είναι ένας δυνατός κινητήρας συνεχούς ρεύματος με κιβώτιο ταχυτήτων με μεταλλικά γρανάζια και σχέση μετάδοσης 75:1. Ο εξωτερικός άξονας είναι 4 mm. Μπορεί να τροφοδοτηθεί με συνεχή τάση από 9 έως 12V. Η ταχύτητα του στα 9V είναι 70 rpm ενώ στα 12V είναι 100rpm και η ροπή του 9 kg.cm.  Η χρήση του κινητήρα αυτού είναι η κίνηση του ταινιόδρομου και γι' αυτό χρειαζόμαστε χαμηλές στροφές αλλά μεγάλη ροπή.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Διάμετρος Κινητήρα:** 25mm
- **Τάση τροφοδοσίας:** 9 - 12Volt
- **Ρεύμα Στασιμότητας:** 2.1A
- **Ταχύτητα:** 100rpm±5% @ 12V,  70rpm±5% @ 9V
- **Ροπή:** 9kg.cm
- **Στυλ Κιβωτίου Ταχυτήτων:** Economy Spur

- **Υλικό Γραναζιών:** Μεταλλικά
- **Λόγος Μετάδοσης Κιβωτίου Ταχυτήτων:** 75:1
- **Κωδικοποιητής:** Όχι
- **Ρεύμα εν κενώ:** 70mA
- **Διάμετρος Άξονα:** 4mm (D)
- **Μήκος Άξονα:** 10mm
- **Μήκος κινητήρα (χωρίς τον άξονα):** 54.2mm
- **Διάμετρος κινητήρα:** 25mm
- **Βάρος:** 90gr [35]

Ο κινητήρας αυτός δεν μπορεί να συνδεθεί απευθείας με τον arduino. Για να γίνει σύνδεση απαιτείτε ένας οδηγός (motor driver), ο οποίος θα επικοινωνεί με τον arduino και επίσης θα τροφοδοτεί τον κινητήρα.

4.1.9 DC αντλία νερού micro 12V

Εικόνα 4.28: Liquid Pump Motor [Πηγή]

Πρόκειται για έναν DC κινητήρα με μια συνδεδεμένη αντλία στον άξονά του. Όπως φαίνεται στην εικόνα η αντλία έχει 2 υποδοχές για τους σωλήνες, η κάτω είναι για την εισαγωγή και η πάνω για την εξαγωγή. Μπορούμε να την τροφοδοτήσουμε με συνεχή τάση από 6 έως 12V και αντίστοιχα μπορεί να αντλήσει υγρά με μέγιστη ροή 1 έως 3 λίτρα το λεπτό και σε θερμοκρασία έως 80°C. Η μέγιστη απόσταση αναρρόφησης υγρού είναι 2 μέτρα και μπορεί να το αντλεί σε κατακόρυφο ύψος έως 3 μέτρα. Όταν αντλεί κάποιο υγρό δεν κάνει καθόλου θόρυβο.

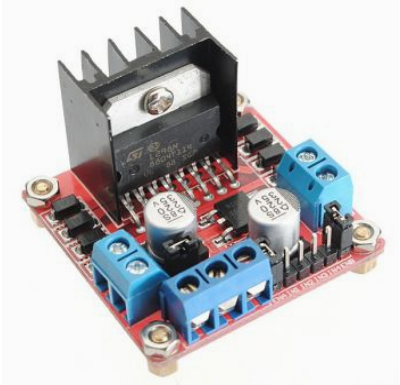
Η χρήση της αντλίας αυτής για την συγκεκριμένη εργασία είναι για το σύστημα της δεξαμενής όπου θα αντλεί υγρό και θα γεμίζει τα κόκκινα δοχεία. [36]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τάση Λειτουργίας:** 6-12V DC
- **Ρεύμα υπό φορτίο:** 0.5-0.7A
- **Μέγιστη Ροή:** 1-3L/Min
- **Μέγιστο Ύψος Άντλησης:** 3 m
- **Μέγιστο Εύρος Αναρρόφησης:** 2 m
- **Θερμοκρασία:** ≤80℃
- **Διάρκεια Ζωής:** 2500H
- **Εσωτερική Διάμετρος Σωλήνα:** 7mm
- **Διαστάσεις:** 86 x 43mm [36]

Για την σύνδεση με τον arduino απαιτείτε οδηγός (Motor driver).

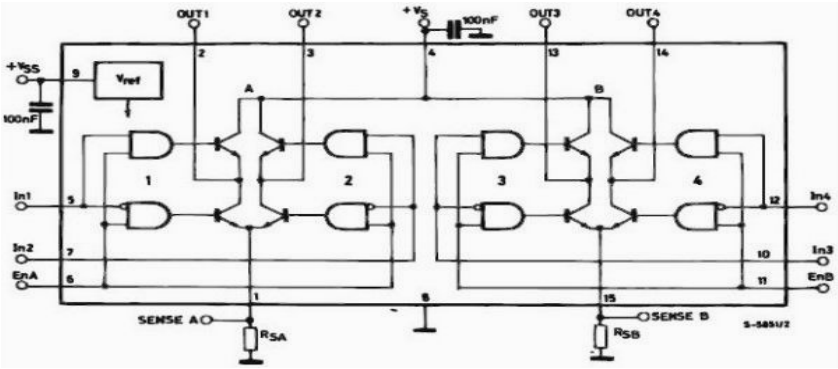
### 4.1.10 L298N dual h bridge driver



Εικόνα 4.29: L298N Dual Motor Driver [Πηγή]

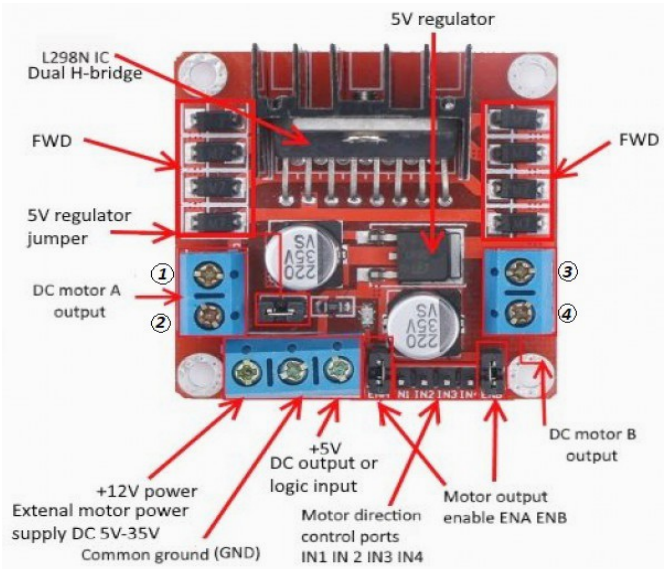
Η πλακέτα αυτή είναι ένας αμφίδρομος οδηγός κινητήρων με τον οποίο μπορούμε να ελέγχουμε την ταχύτητα και την φορά περιστροφής δύο DC κινητήρων ή ενός διπολικού βηματικού κινητήρα. Βασίζετε στο ολοκληρωμένο L298N, που είναι ένας οδηγός διπλής πλήρους γέφυρας (dual H bridge) υψηλής τάσης και ρεύματος. Το διάγραμμα του IC φαίνεται στην παρακάτω εικόνα. (Εικόνα 4.30)

Έχει 2 εισόδους για να μπορούμε να ενεργοποιούμε ή να απενεργοποιούμε την συσκευή χωρίς να χρησιμοποιούμε τα σήματα εισόδου. Επίσης υπάρχει πρόσθετη είσοδος τροφοδοσίας ώστε η λογική του να μπορεί να λειτουργεί σε χαμηλότερη τάση. [37]



Εικόνα 4.30: L298 Circuit Diagram [Πηγή]

Ο οδηγός αυτός μπορεί να ελέγξει ανεξάρτητα 2 κινητήρες συνεχούς ρεύματος με μέγιστο ρεύμα 2A ο καθένας και τάση μεταξύ 5 και 35V. Η πλακέτα αυτή έχει Led για ενδείξεις, ρυθμιστή τάσης 5V και διόδους ελεύθερης διέλευσης (FWD) για προστασία. [39]



Εικόνα 4.31: Ανάλυση Dual Motor Driver Module L298N [Πηγή]

**+12V Power Terminal:** Σύνδεση του θετικού δυναμικού της εξωτερικής τροφοδοσίας DC για τους κινητήρες από 5 έως 35V. Εάν τροφοδοτήσουμε με τάση μεγαλύτερη από 12V πρέπει να αφαιρέσουμε το 5V regulator jumper. Από αυτό τροφοδοτείτε η πλακέτα και οι κινητήρες. [38]

**GND Terminal:** Σύνδεση του αρνητικού δυναμικού (Ground) της εξωτερικής τροφοδοσίας αλλά και του arduino. Η γείωση πρέπει να είναι κοινή. [38]

**+5V Terminal:** Λειτουργεί σαν έξοδος 5V τάσης, εάν το 5V regulator jumper είναι στην θέση του. Αν έχουμε αφαιρέσει το 5V regulator jumper λόγο μεγαλύτερης τάσης τροφοδοσίας της πλακέτας,

τότε λειτουργεί σαν είσοδος που πρέπει να δώσουμε τάση 5V για να τροφοδοτήσουμε την λογική του ολοκληρωμένου (L298N) μιας και ο ρυθμιστής τάσης θα είναι απενεργοποιημένος. [38]

**DC motor A output terminal:**

1. Σύνδεση του θετικού ακροδέκτη (+) του πρώτου DC κινητήρα ή του ακροδέκτη A+ του βηματικού κινητήρα
2. Σύνδεση του αρνητικού ακροδέκτη (-) του πρώτου DC κινητήρα ή του ακροδέκτη A3 του βηματικού κινητήρα

**DC motor B output terminal:**

3. Σύνδεση του αρνητικού ακροδέκτη (-) του δεύτερου DC κινητήρα ή του ακροδέκτη B του βηματικού κινητήρα
4. Σύνδεση του θετικού ακροδέκτη (+) του δεύτερου DC κινητήρα ή του ακροδέκτη B+ του βηματικού κινητήρα [38]

**5V Regulator Jumper:** Όταν είναι συνδεδεμένο ενεργοποιεί την λειτουργία του ρυθμιστή τάσης των 5V ο οποίος τροφοδοτεί το λογικό κύκλωμα του ολοκληρωμένου (L298N) και επίσης έχουμε τάση στο +5V terminal με την οποία μπορούμε να τροφοδοτήσουμε τον arduino ή και αισθητήρες. Αν όμως τροφοδοτούμε τον οδηγό με τάση μεγαλύτερη των 12V, πρέπει να αφαιρεθεί και τότε απενεργοποιείτε ο ρυθμιστής τάσης για να μην καεί. Έτσι δεν μπορούμε να πάρουμε τάση από το +5V terminal αλλά θα πρέπει να δώσουμε εμείς τάση 5V. [38]

**Motor ENA Jumper:** Το αφήνουμε στην θέση του αν χρησιμοποιούμε βηματικό κινητήρα, αλλιώς αν χρησιμοποιούμε DC κινητήρα το αφαιρούμε και συνδέουμε το pin αυτό με μια έξοδο PWM του arduino για τον έλεγχο της ταχύτητας. [38]

**Motor ENB Jumper:** Το αφήνουμε στην θέση του αν χρησιμοποιούμε βηματικό κινητήρα, αλλιώς το αφαιρούμε και συνδέουμε το pin αυτό με μια έξοδο PWM του arduino για τον έλεγχο της ταχύτητας του κινητήρα. [38]

**IN1, IN2 pins:** Είναι για τον έλεγχο της φοράς περιστροφής του πρώτου κινητήρα. Συνδέονται σε ψηφιακές εξόδους του arduino. [38]

**IN3, IN4 pins:** Είναι για τον έλεγχο της φοράς περιστροφής του δεύτερου κινητήρα. Συνδέονται σε ψηφιακές εξόδους του arduino. [38]

Για να ελέγξουμε έναν κινητήρα πρέπει να δώσουμε κατάλληλες λογικές τιμές HIGH ή LOW (1 ή 0) στα pin IN1,IN2 και μία τιμή PWM από 0 έως 255 στο pin ENA για την ταχύτητα του κινητήρα. Αντίστοιχα για τον δεύτερο κινητήρα πρέπει να δώσουμε τους ίδιους συνδυασμούς τιμών αλλά στα

pin ENB, IN3,IN4. Οι συνδυασμοί των λογικών τιμών φαίνονται στον παρακάτω πίνακα. Η τιμή 1 στο ENA σημαίνει ότι αν δουλεύει ο κινητήρας θα έχει την μέγιστη ταχύτητα, δηλαδή αντιστοιχεί στην PWM τιμή 255.

| ENA | IN1 | IN2 | Description |
|---|---|---|---|
| 0 | N/A | N/A | Motor A is off |
| 1 | 0 | 0 | Motor A is stopped (brakes) |
| 1 | 0 | 1 | Motor A is on and turning backwards |
| 1 | 1 | 0 | Motor A is on and turning forwards |
| 1 | 1 | 1 | Motor A is stopped (brakes) |

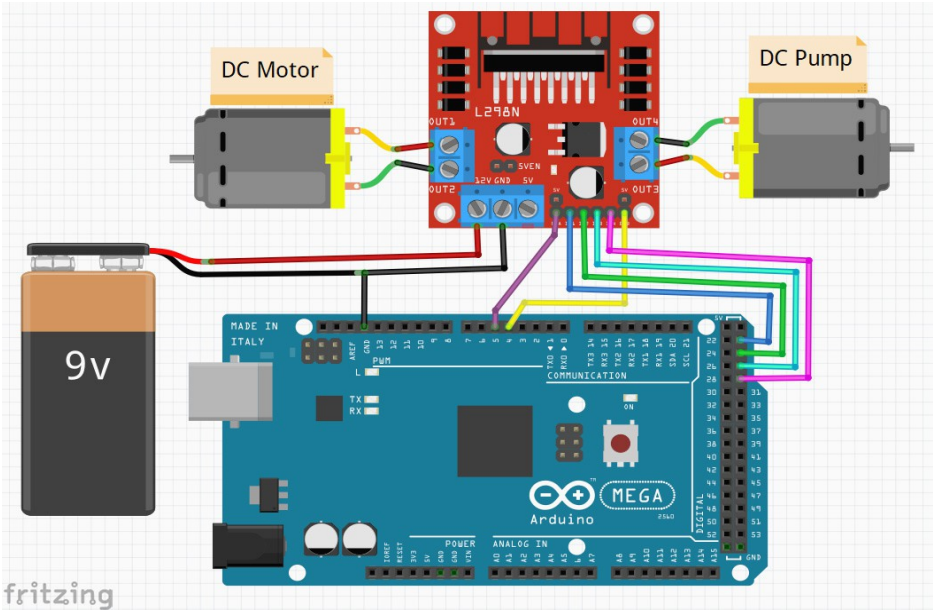
Πίνακας 4.5: Πίνακας αλήθειας του L298N [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τύπος οδηγού:** DC Motor - Stepper
- **Οδηγός:** L298N Dual H Bridge DC Motor Driver IC
- **Τάση εισόδου Vs:** 5 ~ 35V DC
- **Μέγιστο Ρεύμα ανά Κανάλι:** 2A
- **Ρεύμα Κορυφής (Peak):** 3A
- **Μέγιστη Ισχύς:** 20 W (όταν η θερμοκρασία T = 75 °C)
- **Τάση Λογικού Επιπέδου Vss:** +5 V ~ +7 V
- **Ρεύμα Λογικού Επιπέδου:** 0 - 36mA
- **Εύρος Σήματος Λογικής Εισόδου:** [ Low:-0.3V ≤ Vin ≤ 1.5V] [ High: 2.3V ≤ Vin ≤ Vss]
- **Θερμοκρασία Αποθήκευσης:** -25 °C ~ +130 °C.
- **Διαστάσεις:** 3,4cm x 4,3cm x 2,7cm [39]

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ARDUINO

- **ENA** → PWM Digital Output (αν ελέγχουμε κινητήρα)
- **IN1** → Digital Output
- **IN2** → Digital Output
- **ENB** → PWM Digital Output (αν ελέγχουμε κινητήρα)
- **IN3** → Digital Output
- **IN4** → Digital Output



Εικόνα 4.32: Κύκλωμα σύνδεσης L298N driver με arduino για έλεγχο κινητήρα και αντλίας

Παράδειγμα Κώδικα

```
//Δήλωση των pin για τον έλεγχο της αντλίας
int enB = 4;
int in3 = 27;
int in4 = 29;
//Δήλωση των pin για τον έλεγχο του κινητήρα
int enA = 5;
int in1 = 23;
int in2 = 25;

void setup() {
//Αρχικοποίηση των pin
pinMode(enB, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
pinMode(enA, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
)
void loop(){
//Η αντλία είναι σταματημένη
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
//Ο κινητήρας είναι σταματημένος
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
delay(3000);  //περιμένει 3 δευτερόλεπτα
//Η αντλία ξεκινά δεξιόστροφα με ταχύτητα 200 από 255
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
analogWrite(enB, 200);
//Ο κινητήρας ξεκινά δεξιόστροφα με ταχύτητα 120 από 255
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
analogWrite(enA, 120);
delay(5000);  //περιμένει 5 δευτερόλεπτα
```

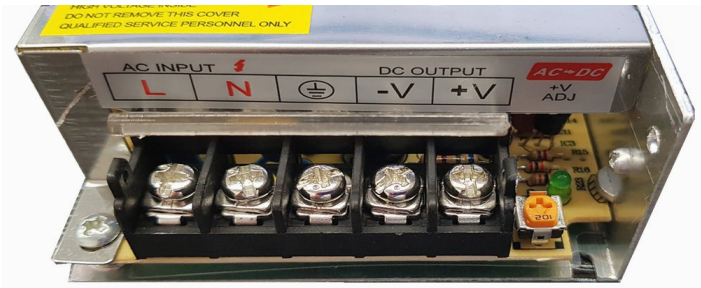
```
//Ο κινητήρας σταματά
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
delay(1000); // περιμένει 1 δευτερόλεπτο
//Ο κινητήρας ξεκινά αριστερόστροφα με πλήρη ταχύτητα
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(enA, HIGH); //ή μπορούμε να το γράψουμε ως analogWrite(enA, 255);
delay(2000); //περιμένει 2 δευτερόλεπτα
}
```

### 4.1.11 SN-HS-60-12 DC Τροφοδοτικό 12V


Εικόνα 4.33: DC Τροφοδοτικό SN-HS-60-12 [Πηγή]

Για την γενική τροφοδοσία της κατασκευής χρησιμοποιήθηκε το τροφοδοτικό συνεχούς ρεύματος SN-HS-60-12. Είναι ένα τροφοδοτικό με ρυθμιζόμενο εύρος τάσης εξόδου 12V και ρεύμα 5A. Παρέχει προστασία στην έξοδο για βραχυκύκλωμα, υπερένταση, υπέρταση και έχει επίσης EMI φίλτρο για να μειώνει τον ηλεκτρονικό θόρυβο υψηλής συχνότητας. Η τάση εξόδου είναι σταθερή και ακριβής και με μικρή κυμάτωση και θόρυβο. [40]

Οι ακροδέκτες σύνδεσης και το ποτενσιόμετρο ρύθμισης της τάσης φαίνονται στην παρακάτω εικόνα.




Εικόνα 4.34: Ακροδέκτες σύνδεσης του τροφοδοτικού [Πηγή]

Οι ακροδέκτες από αριστερά προς δεξιά είναι:

- **L:** φάση (είσοδος AC)
- **N:** ουδέτερος (είσοδος AC)
- **Gnd:** γείωση (είσοδος AC)
- **-V:** αρνητικό δυναμικό (έξοδος DC)
- **+V:** θετικό δυναμικό (έξοδος DC)

Δίπλα από τους ακροδέκτες είναι το ποτενσιόμετρο ρύθμισης της τάσης εξόδου με πορτοκαλί χρώμα και πίσω από αυτό υπάρχει ένα πράσινο led που δείχνει ότι λειτουργεί το τροφοδοτικό.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τάση εισόδου:** 100 – 240V AC
- **Συχνότητα εισόδου:** 50/60Hz
- **Τάση εξόδου που μπορεί να ρυθμιστεί:** ± 10V
- **Τάση εξόδου:** 12V
- **Ρεύμα εξόδου:** 5A
- **Ισχύς εξόδου:** 60W
- **Απόδοση:** 87%
- **Κυματισμός, Θόρυβος:** 100mV
- **Ρύθμιση τάσης:** (Πλήρες φορτίο) ≤ 0.5%
- **Προστασία υπερέντασης:** 105% - 150%
- **Προστασία υπέρτασης:** 105% - 150%
- **Προστασία έναντι βραχυκυκλώματος**
- **Ένταση μόνωσης P.S, P.C:** 1500VAC/1min
- **Ένταση μόνωσης S.C:** 500VAC/1min
- **Πρότυπα ασφαλείας:** Σύμφωνα με τα GB9443, UL60950, EN60950-1
- **Πρότυπα EMC:** Σύμφωνα με τα GB9254, EN55022 class A EN61347-2-13:2008
- **Θερμοκρασία λειτουργίας:** -20℃ ~ 60℃ / 20% ~ 90% RH
- **Θερμοκρασία αποθήκευσης:** -40℃ ~ 85℃ / 10% ~ 95% RH
- **Διαστάσεις:** 110 (L) * 78(W) * 36(H) mm
- **Βάρος:** 169 g [40]

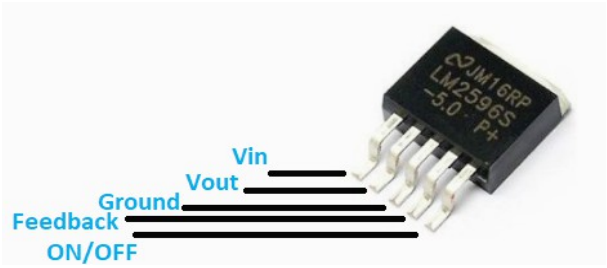
4.1.12 Μετατροπέας τάσης DC-DC step down

Εικόνα 4.35: Μετατροπέας DC-DC Step-Down [Πηγή]

Είναι ένας ρυθμιζόμενος μετατροπέας step down ή αλλιώς buck converter που μειώνει την τάση εισόδου στην έξοδο και μπορούμε να ρυθμίσουμε και το εύρος της. Το εύρος της τάσης εισόδου είναι 3,2V - 40V και της τάσης εξόδου του είναι 1,3V – 30V. Βασίζετε στον ρυθμιστή LM2596S.

Ένας buck converter (step down) είναι μια κατηγορία τροφοδοτικών μεταγωγής (SMPS) και συνήθως έχει ένα τρανζίστορ και μία δίοδο ενώ στην είσοδο αλλά και στη έξοδο υπάρχει ένας πυκνωτής ή πηνίο ή και τα δύο μαζί ως φίλτρο για να μειωθεί ο κυματισμός τάσης. Η απόδοση των μετατροπέων αυτών είναι πολύ μεγάλη, συνήθως πάνω από 90%. [41]

Ο ρυθμιστής τάσης LM2596S είναι ένα μονολιθικό ολοκληρωμένο κύκλωμα που περιέχει όλες τις λειτουργίες για έναν step down μετατροπέα. Μπορεί να χειρίζεται ρεύμα φορτίου 3A με εξαιρετική ρύθμιση τάσης και ρεύματος. Λειτουργεί σε συχνότητα μεταγωγής 150Khz. [42]




Εικόνα 4.36: LM2596S pinout [Πηγή]

**Vin:** Είσοδος τροφοδοσίας. Πρέπει να υπάρχει ένας κατάλληλος πυκνωτής εισόδου για να ελαχιστοποιεί τις μεταβατικές τάσεις και να τροφοδοτεί τα ρεύματα μεταγωγής που απαιτούνται από τον ρυθμιστή.

**Vout:** Έξοδος της μειωμένης τάσης. Η τάση σε αυτό το pin αλλάζει μεταξύ περίπου (+VIN −

VSAT) και περίπου −0,5 V, με κύκλο λειτουργίας VOUT / VIN.

**Ground:** Γείωση

**Feedback:** Ανιχνεύει την ρυθμισμένη τάση εξόδου για να ολοκληρώσει τον βρόχο ανάδρασης.

**ON/OFF:** Επιτρέπει την διακοπή λειτουργίας του κυκλώματος ρυθμιστή χρησιμοποιώντας λογικά σήματα. Το pin αυτό στην πλακέτα αυτού του μετατροπέα είναι συνδεδεμένο στην γείωση αφού δεν χρησιμοποιείτε αυτή η δυνατότητα. Έτσι ο ρυθμιστής είναι σε κατάσταση ON πάντα. [42]

Οι ακροδέκτες της πλακέτας αυτής είναι τέσσερις. Δύο για την είσοδο και άλλοι δύο για την έξοδο με τα αντίστοιχα ονόματα (in+, in-, out+, out-). Στη μέση στης πλακέτας βρίσκετε το ποτενσιόμετρο για την ρύθμιση της τάσης εξόδου. Ο σκοπός αυτό του μετατροπέα είναι να μειώσει την τάση των 12V από την γενική τροφοδοσία σε 6V, ώστε να τροφοδοτήσει τους 5 σερβοκινητήρες του ρομποτικού βραχίονα.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

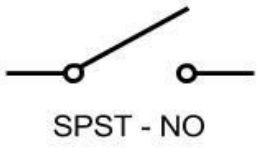
- **Τάση Εισόδου:** 3.2-40V
- **Τάση Εξόδου:** 1.3-30V
- **Ένταση Εξόδου:** 3A
- **Απόδοση:** 92%
- **Διακοπτική Συχνότητα:** 65Khz
- **Θερμοκρασία Λειτουργίας:** -45 ℃ ~ +85 ℃ [43]

4.1.13 Push Buttons

Εικόνα 4.37: Push Buttons πράσινο και κόκκινο [Πηγή] [Πηγή]

Τα button αυτά λειτουργούν σαν απλοί διακόπτες. Οι δύο ακροδέκτες τους είναι κανονικά ανοικτοί (NO) και όταν πατηθεί το button οι ακροδέκτες κλείνουν για όσο το button παραμένει πατημένο. Αυτή η διαμόρφωση των επαφών λέγετε SPST-NO (Single pole, single throw, normally open) και το

σύμβολο της φαίνεται στην παρακάτω εικόνα.

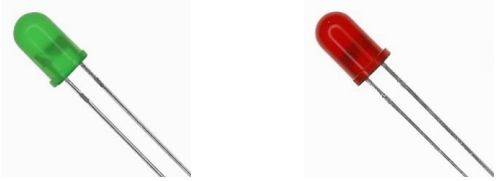



Εικόνα 4.38: Συμβολισμός επαφής SPST-NO [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Μέγιστο Ρεύμα:** 1A / 220V
- **Διαμόρφωση Επαφών** : SPST-NO
- **Μέθοδος Εναλλαγής :** OFF-(ON)
- **Επαφές για:** Soldering [44]

4.1.14 Λυχνίες Led



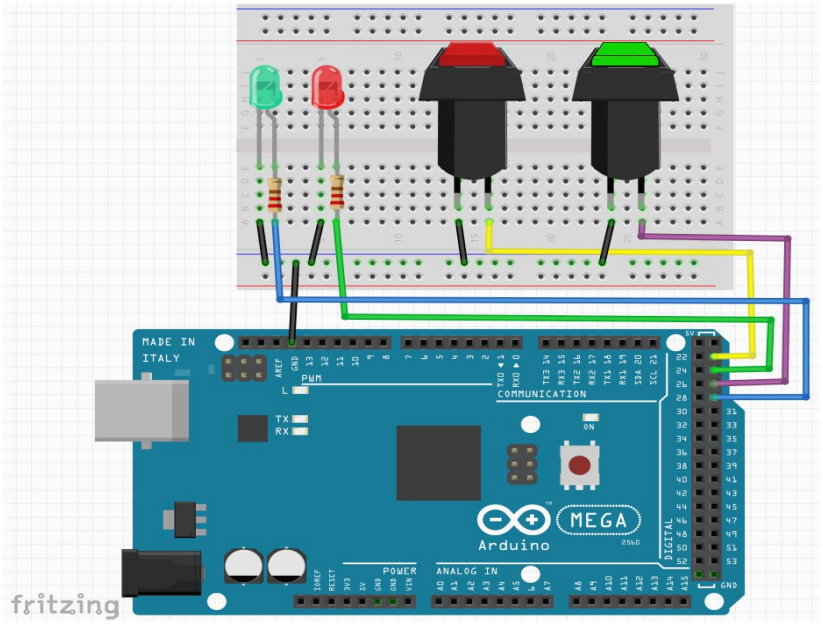
Εικόνα 4.39: Πράσινο και κόκκινο Led [Πηγή] [Πηγή]

Οι λυχνίες led ή αλλιώς δίοδοι εκπομπής φωτός είναι ημιαγωγοί οι οποίοι όταν εκπέμπουν φως σε διάφορα χρώματα όταν εφαρμόζετε τάση στους ακροδέκτες τους. Συγκεκριμένα θα πρέπει να πολωθεί ορθά, δηλαδή στον θετικό ακροδέκτη του led να συνδεθεί το θετικό δυναμικό της πηγή και στον αρνητικό ακροδέκτη το αρνητικό δυναμικό της πηγής. Εάν το ρεύμα που διέρχεται από την δίοδο είναι μεγάλο τότε μπορεί να καταστραφεί το led, για τον λόγο αυτό συνήθως συνδέουμε και μια αντίσταση στο κύκλωμα. [45]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Πτώση Τάσης:** 1.8-2.4 VDC
- **Μέγιστο Ρεύμα:** 20 mA
- **Προτεινόμενο Ρεύμα Χρήσης:** 16-18mA
- **Φωτεινή Ένταση:** 20 mcd [45]

ΣΥΝΔΕΣΜΟΛΟΓΙΑ BUTTON ΚΑΙ LED ΜΕ ARDUINO




Εικόνα 4.40: Κύκλωμα σύνδεσης led και button με arduino mega

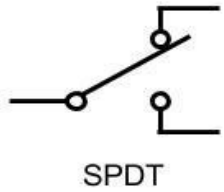
Παράδειγμα Κώδικα

```
#define GreenLedPin 29  //καθορίζουμε που έχουν συνδεθεί τα pin
#define RedLedPin 25
#define StopButtonPin 24
#define StartButtonPin 27
int StartButtonStatus = 0; //δήλωση μεταβλητών για τις καταστάσεις των button
int StopButtonStatus = 0;

void setup() {
//Αρχικοποιούμε τις καταστάσεις των led και button
pinMode(StopButtonPin, INPUT_PULLUP);  //Για τα button ενεργοποιούμε την εσωτερική
αντίσταση PULLUP
pinMode(StartButtonPin, INPUT_PULLUP); //γιατί δεν έχουμε συνδέσει εξωτερική
αντίσταση
pinMode(GreenLedPin, OUTPUT);
pinMode(RedLedPin, OUTPUT);
)
void loop() {
//Ελέγχουμε αν έχει πατηθεί το button stop
if (StopButtonStatus == digitalRead(StopButtonPin)){ //αν έχει πατηθεί, η τιμή που
θα διαβάσει θα είναι LOW
    delay(10);
    digitalWrite(GreenLedPin, LOW); //το πράσινο led σβήνει και
    digitalWrite(RedLedPin, HIGH);  //το κόκκινο led ανάβει
  }
//Ελέγχουμε αν έχει πατηθεί το button start
if (StartButtonStatus == digitalRead(StartButtonPin)){ //αν έχει πατηθεί, η τιμή
που θα διαβάσει θα είναι LOW
    delay(10);
    digitalWrite(GreenLedPin, HIGH); //το πράσινο led ανάβει και
    digitalWrite(RedLedPin, LOW);  //το κόκκινο led σβήνει
  }
}
```

4.1.15 Rocker Switch KCD5-102

Εικόνα 4.41: Διακόπτης KCD5-102 [Πηγή]

Πρόκειται για έναν διακόπτη μικρό σε μέγεθος που μπορεί να χειριστεί 3 Α ρεύμα και προσφέρει χαμηλό ρεύμα μεταγωγής. Έχει μεγάλη μηχανική διάρκεια ζωής σε βαριά φορτία. Ο συγκεκριμένος διακόπτης έχει 3 επαφές από τις οποίες η μία είναι είσοδος και μπορεί να αλλάζει μεταξύ δύο εξόδων. Η διαμόρφωση αυτή λέγεται SPDT (Single Pole Double Throw) και το σύμβολό της φαίνεται στην παρακάτω εικόνα. [46]




Εικόνα 4.42: Συμβολισμός επαφής SPDT [Πηγή]

Ένας τέτοιος διακόπτης μπορεί να λειτουργήσει με δύο τρόπους, σαν on-off διακόπτης χρησιμοποιώντας τις δύο από τις τρεις επαφές ή για να αλλάζει μεταξύ δύο λειτουργιών ενός κυκλώματος χρησιμοποιώντας και τις τρεις επαφές του.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Διαμόρφωση Επαφών:** SPDT
- **Μέθοδος Εναλλαγής:** ON-ON
- **Ρεύμα DC ή AC:** 6A 125VAC / 3A 250VAC
- **Επαφές για:** Flat Terminal
- **Αντίσταση Επαφής:** 50 mΩ max
- **Θερμοκρασία Λειτουργίας:** -25°C~+85°C
- **Αντίσταση Μόνωσης:** 100mΩ min (500VDC)

- **Ηλεκτρική Ζωή:** ≥ 10,000 Cycles
- **Διηλεκτρική Αντοχή:** 1500VAC 1 minute
- **Μηχανική Ζωή:** ≥ 100,000 Cycles
- **Πιστοποιήσεις:** CE, CQC, SGS [46]

### 4.1.16 Αντιστάσεις

Για την σύνδεση των λυχνιών led χρησιμοποιήθηκαν δύο αντιστάσεις τύπου carbon film με τιμή 220 Ω. Επίσης χρησιμοποιήθηκε και μία αντίσταση τύπου metal film με τιμή 4,7 ΚΩ για την σύνδεση του ψηφιακού αισθητήρα θερμοκρασίας.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

| **Αντίσταση τύπου Carbon Film** | **Αντίσταση τύπου Metal Film** |
|---|---|
| **Τιμή αντίστασης:** 220Ω | **Τιμή αντίστασης:** 4,7ΚΩ |
| **Ισχύς:** 0.5 - 1/2W | **Ισχύς:** 0.6 W |
| **Ανοχή:** ±5% | **Ανοχή:** ±1% |
| **Μέγιστη Τάση Λειτουργίας:** 350V | **Μέγιστη Τάση Λειτουργίας:** 250V |

Πίνακας 4.6: Ηλεκτρικά χαρακτηριστικά αντιστάσεων carbon και metal

### 4.1.17 Terminal Blocks

Εικόνα 4.43: Δυο τύποι terminal blocks [Πηγή] [Πηγή]

Τα terminal blocks ή αλλιώς κλέμες σύνδεσης τα χρησιμοποιούμε για να συνδέσουμε πολλά καλώδια σε κοινό κόμβο. Για την κατασκευή αυτή χρησιμοποιήθηκαν και τα δύο της εικόνας, σαν σημεία λήψης των διάφορων τάσεων (12V, 6V, 5V) και γείωσης που χρειάζονται τα διάφορα ηλεκτρονικά εξαρτήματα.

4.1.18 Καλώδια Σύνδεσης

Για τις συνδέσεις των ηλεκτρονικών – ηλεκτρικών εξαρτημάτων χρησιμοποιήθηκαν διάφορα καλώδια όπως, jumper wires ή αλλιώς καλώδια DuPont για τις συνδέσεις στον arduino, μονοπολικά καλώδια χαλκού πολύκλωνα διατομής 0,5mm$^2$, καλώδια επέκτασης για σερβοκινητήρες, καλώδιο τροφοδοσίας θηλυκό jack σε γυμνό καλώδιο για την τροφοδοσία του arduino και τριπολικό καλώδιο τροφοδοσίας AC PLUG διατομής 3 x 0.75mm$^2$ αρχιτεκτονικής CEE 7/7 (E/F) για την παροχή AC τάσης στο τροφοδοτικό.

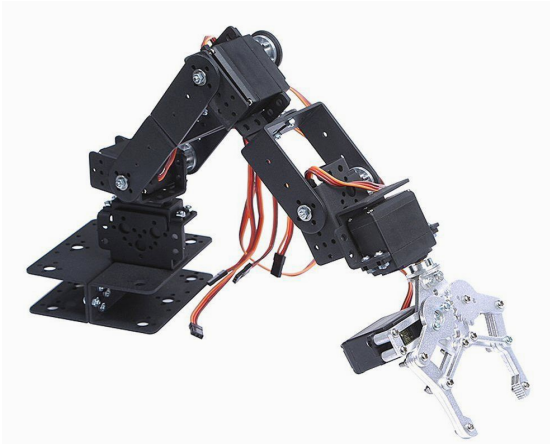
4.2 Μηχανολογικά και μηχανικά εξαρτήματα

4.2.1 Ρομποτικός Βραχίονας 6 Βαθμών ελευθερίας Kit

Εικόνα 4.44: Kit ρομποτικού βραχίονα [Πηγή]

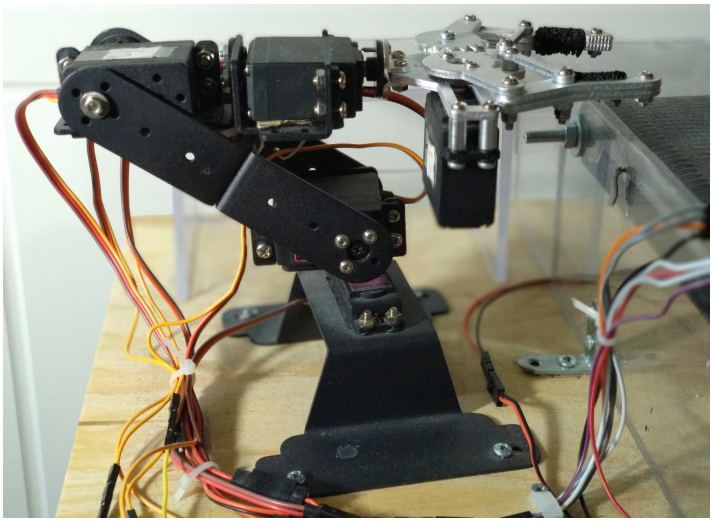
Το kit αυτό περιλαμβάνει διάφορα αλουμινένια bruckets, βάση, δαγκάνα, βίδες και παξιμάδια διάφορων μεγεθών και μικρά ρουλεμάν και με αυτά μπορούμε να συναρμολογήσουμε έναν ρομποτικό βραχίονα έως 6 βαθμούς ελευθερίας ή οτιδήποτε άλλο θέλουμε, μιας και τα bruckets έχουν πολλές τρύπες σε όλη τους την επιφάνεια ώστε να μπορούν να συνδεθούν μεταξύ τους με πολλούς τρόπους και επιπλέον οι βίδες και τα παξιμάδια είναι περισσότερα απ' ότι χρειάζεται για τον βραχίονα.

Οι σερβοκινητήρες που ταιριάζουν στα bruckets αυτά είναι μεγέθους standard και δεν περιλαμβάνονται στο kit. Επίσης θα πρέπει να είναι μεγάλης ροπής και με μεταλλικά γρανάζια ειδικά στα χαμηλά σημεία του βραχίονα γιατί θα πρέπει να σηκώνουν όλο το βάρος του.



Εικόνα 4.45: Συναρμολογημένος βραχίονας 6 βαθμών ελευθερίας [Πηγή]

Για την κατασκευή της εργασίας αυτή ο βραχίονας είναι 5 βαθμών ελευθερίας ώστε να μην είναι πολύ ψηλός.



Εικόνα 4.46: Ρομποτικός βραχίονας 5 βαθμών ελευθερίας της εργασίας αυτής

Αναλυτικά το kit περιλαμβάνει:

- 1 x mechanical claws
- 5 x multifunctional bracket
- 3 x long U bracket
- 1 x L bracket
- 1 x βάση
- 4 x cup bearing
- 1 set x mounting screws

4.2.2 Plexiglass

Για την κατασκευή του ταινιόδρομου, της δεξαμενής και άλλων μερών της κατασκευής, χρησιμοποιήθηκε διάφανο φύλλο plexiglass πάχους 5 mm.

Το plexiglass ή αλλιώς PMMA ( πολυ μεθακρυλικός μεθυλεστέρας) ανήκει σε μια ομάδα υλικών που λέγονται πλαστικά μηχανικής. Έχει μεγάλη αντοχή στον χρόνο και στην χρήση, το βάρος του είναι πιο χαμηλό από αυτό του κοινού γυαλιού και είναι και πιο καθαρό από το γυαλί. [47] [48]

Τα πλεονεκτήματά του είναι:

- Είναι θερμομονωτικό
- Αντέχει σε υψηλές θερμοκρασίες
- Αντέχει σε δυσμενείς καιρικές συνθήκες
- Πιο διαφανές από το γυαλί
- Μεγάλη αντοχή στη χρήση
- Δεν γίνεται θρύψαλα [48]

4.2.3 Εξαρτήματα για τον ταινιόδρομο

4.2.3.1 Aluminum tube

Για τα rollers του ταινιόδρομου χρησιμοποιήθηκαν δύο σωλήνες αλουμινίου που φαίνονται στην παρακάτω εικόνα.

Εικόνα 4.47: Σωλήνας αλουμινίου [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Μήκος:** 3" / 76,2mm
- **Εσωτερική Διάμετρος:** 0.82" / 20.8mm
- **Εξωτερική Διάμετρος:** 1" / 25.4mm
- **Πάχος:** 0.09" / 2.3mm
- **Υλικό:** Aluminum (Aircraft Grade) [49]

4.2.3.2 Ρουλεμάν - 627ZZ

Εικόνα 4.48: Ρουλεμάν [Πηγή]

Για τα rollers του ταινιόδρομου χρησιμοποιήθηκαν 3 τέτοια ρουλεμάν, τα οποία τοποθετήθηκαν χωνευτά στις άκρες των δύο αλουμινένιων σωλήνων για την ελάττωση της τριβής ώστε να υπάρχει ομαλή κύλιση.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Τύπος:** Deep Groove
- **Εσωτερική Διάμετρος (Άξονας):** 7mm
- **Εξωτερική Διάμετρος:** 22mm
- **Λίπανση:** Λάδι
- **Θωράκιση:** Metal Shielded
- **Πάχος:** 7mm
- **Υλικό:** Chrome Steel [50]

4.2.3.3 Shaft coupler solid

Για τη σύνδεση του άξονα του roler με τον άξονα του κινητήρα χρησιμοποιήθηκε αυτός ο shaft coupler ή ζεύκτης άξονα.

Εικόνα 4.49: Shaft coupler [Πηγή]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

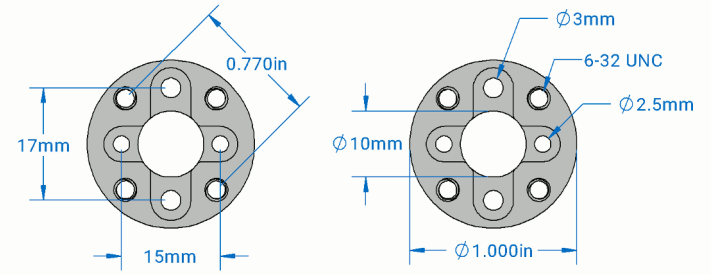
- **Εσωτερική Διάμετρος Α:** 4mm

- **Εσωτερική Διάμετρος B:** 5mm
- **Εξωτερική Διάμετρος:** 9mm
- **Μήκος:** 20mm
- **Τύπος Συνδέσμου:** Solid
- **Υλικό:** Brass [51]

4.2.3.4 Aluminum motor mount F

Εικόνα 4.50: Βάση κινητήρα [Πηγή]

Η βάση αυτή είναι συμβατή με δύο διαφορετικούς κινητήρες. Μπορεί να στερεωθεί στην πρόσοψη του κινητήρα χρησιμοποιώντας δύο από τις τρύπες που βρίσκονται μέσα στον σταυρό ανάλογα τον τύπο του κινητήρα. Ύστερα μπορούμε να στερεώσουμε την βάση με τον κινητήρα σε οποιοδήποτε εξάρτημα ή επιφάνεια που να ταιριάζει με τις εξωτερικές τρύπες. Στην παρακάτω εικόνα φαίνεται στο σχέδιο της βάσης αυτής.




Εικόνα 4.51: Μηχανολογικό σχέδιο της βάσης [Πηγής]

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

- **Διάμετρος Κινητήρα:** 25mm
- **Υλικό:** Aluminum 6061-T6
- **Βάρος:** 2.8g
- **Συμβατό με:** Economy Spur Gear Motors (25mm), Mini Economy Spur Gear Motors [52]

4.2.3.5 Άλλα εξαρτήματα και αναλώσιμα

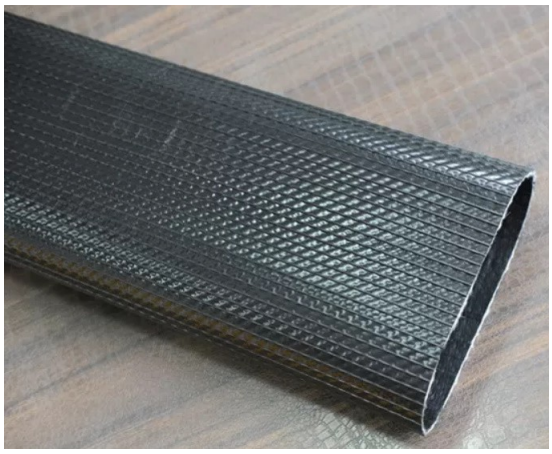
Για τον άξονα των roller του ταινιόδρομου χρησιμοποιήθηκαν 4 εξάγωνες βίδες (μία σε κάθε πλευρά των roller) τύπου DIN 933, μεγέθους m6 και μήκους 40 mm, καθώς και παξιμάδια και ροδέλες μεγέθους m6.

Για την εγκατάσταση του κινητήρα και των IR αισθητήρων στον ταινιόδρομο χρησιμοποιήθηκαν μεταλλικοί αποστάτες δύο μεγεθών (20mm και 30mm) τύπου θηλυκό σε θηλυκό με σπείρωμα m3 και σχήμα εξαγωνικό.

Για την στήριξη του ταινιόδρομου και των IR αισθητήρων χρησιμοποιήθηκαν γωνίες στήριξης μικρού μεγέθους.

Για την πλευρά του αλουμινένιου σωλήνα που θα κινεί ο κινητήρας δεν υπάρχει ρουλεμάν αλλά μια πλαστική τάπα ή οποία εφαρμόζει σφιχτά ώστε να μπορεί να περιστρέφει το roller. Στο κέντρο της τάπας ανοίχτηκε τρύπα διατομής 6 mm για να περάσει η βίδα m6 που είναι ο άξονας.

Για την ταινία μεταφοράς του ταινιόδρομου χρησιμοποιήθηκε ένα κομμάτι μάνικας PVC/ NBR layflat το οποίο κόπηκε στα δύο, στο κατάλληλο μήκος και πλάτος του ταινιόδρομου και ράφτηκε με κλωστή και πετονιά.



Εικόνα 4.52: Μάνικα PVC [Πηγή]

4.2.4 Εξαρτήματα για το σύστημα της δεξαμενής

Για την παροχή της δεξαμενής χρησιμοποιήθηκε ένας πλαστικός μαστός δεξαμενής με αρσενικό σπείρωμα και μονή φλάντζα στεγανοποίησης (Εικόνα 4.53). Το σπείρωμα του είναι 21mm και η εσωτερική του διάμετρος 14mm.

Εικόνα 4.53: Μαστός δεξαμενής με αρσενικό σπείρωμα και μονή φλάντζα [Πηγή]

Στο σπείρωμα του μαστού που βρίσκετε έξω από την δεξαμενή συνδέθηκε ένα ρακόρ πεταλούδα ορειχάλκινο με θηλυκό σπείρωμα 21mm (Εικόνα 4.54), για να συνδεθεί μετά το σωληνάκι που θα πάει στην αντλία. Η εξωτερική διάμετρος του ρακόρ για την σύνδεση του σωλήνα είναι 8mm και η εσωτερική είναι 6mm.

Εικόνα 4.54: Ρακόρ πεταλούδα ορειχάλκινο με θηλυκό σπείρωμα [Πηγή]

Για την σύνδεση του ρακόρ με την είσοδο της αντλίας χρησιμοποιήθηκε ένα σωληνάκι σιλικόνης με εσωτερική διάμετρο 7mm και εξωτερική 10mm και για την έξοδο της αντλίας χρησιμοποιήθηκε ένα άλλο σωληνάκι εσωτερικής διατομής 5mm και εξωτερικής 7mm .

Εικόνα 4.55: Σωληνάκι σιλικόνης 7mm και σωληνάκι 5mm [Πηγή] [Πηγή]

Τέλος για την κατασκευή της δεξαμενή χρησιμοποιήθηκε κόλλα στιγμής για να κολλήσουν τα κομμάτια plexiglass και διάφανη σιλικόνη αδιαβροχοποίησης η οποία μπήκε εσωτερικά και εξωτερικά στις ενώσεις των plexiglass για να μην υπάρχει διαρροή.

4.2.5 Άλλα αναλώσιμα και κουτιά

Για τον arduino mega χρησιμοποιήθηκε ένα κουτί – θήκη για arduino mega από πλαστικό ABS. Η θήκη αυτή αποτελείτε από δύο μέρη (το κάτω και το πάνω μέρος) τα οποία κουμπώνουν μεταξύ τους.

Εικόνα 4.56: Arduino mega case [Πηγή]

Είναι συμβατό με πλακέτες arduino mega αλλά και uno και μπορούμε να τις ασφαλίσουμε με βίδες αν και δεν είναι απαραίτητο. Παρέχει προστασία στην πλακέτα και έχουμε πρόσβαση σε όλα τα pin , στο usb, στο reset button και στο jack connector. [53]

Χρησιμοποιήθηκε επίσης ένα κουτί κατασκευής G1020B από ανθεκτικό υλικό ABS για το control panel.

Εικόνα 4.57: G1020B κουτί κατασκευής ABS [Πηγή]

Στην επιφάνεια του κουτιού εγκαταστάθηκαν τα button, τα led και ο διακόπτης (rocker switch) και στο εσωτερικό του έγιναν οι συνδέσεις. Έχει μήκος 83mm, πλάτος 54mm και ύψος 30mm. Η κατηγορία ευφλεκτότητας είναι UL94HB. [54]

Τέλος χρησιμοποιήθηκαν βίδες m3 τύπου DIN7985 με είδος κεφαλής (cheese head) και κοπή κεφαλής (Phillips) σε διάφορα μήκη και αντίστοιχα παξιμάδια m3. Επίσης βίδες UNC6-32 με είδος κεφαλής (cheese head) και κοπή κεφαλής (Phillips) καθώς και δεματικά (tire up).

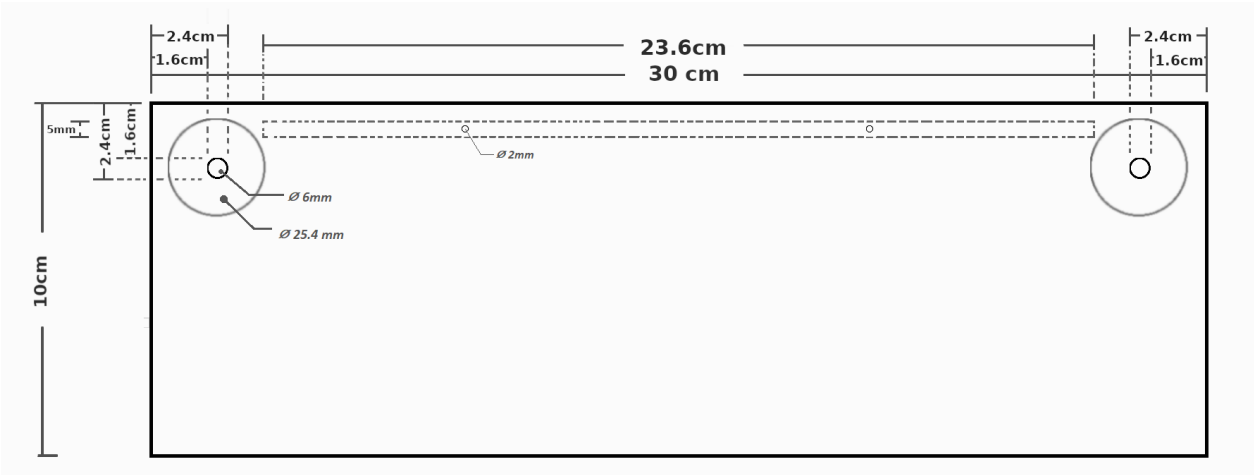
4.3 Κατασκευή

Ο ταινιόδρομος αποτελείτε από τρία κομμάτια plexiglass διαστάσεων (δύο 30cm x 10cm που είναι τα πλαϊνά μέρη και ένα 23.6cm x 8.8cm που είναι η οριζόντια επιφάνεια) και από δύο roller, ένα που κινεί ο κινητήρας και ένα που το κινεί η ταινία (παθητικό roller). Το παθητικό roller αποτελείτε από μια βίδα μ6 για άξονα, ροδέλα, ρουλεμάν και παξιμάδι και στις δύο πλευρές.

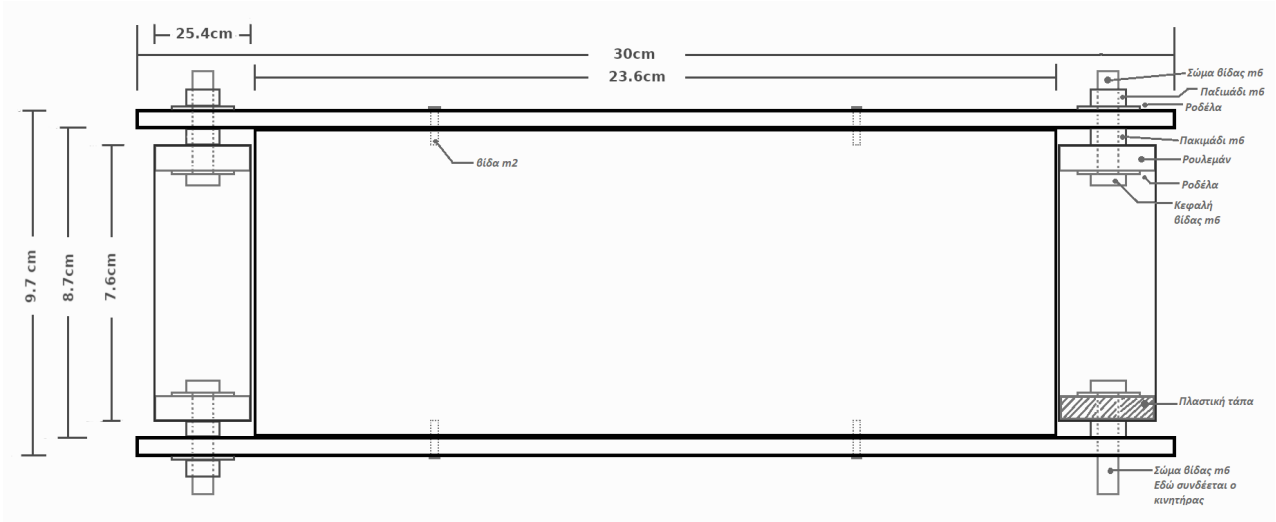
Εικόνα 4.58: Σχέδιο κατασκευής παθητικού roller ταινιόδρομου

Το άλλο roller που το κινεί ο κινητήρας είναι ίδιο με της παραπάνω εικόνας με την μόνη διαφορά ότι στην μία πλευρά δεν έχει ρουλεμάν αλλά μια πλαστική τάπα για να μπορεί να περιστρέφει το roller.

Η κάτοψη και η πλαϊνή όψη του ταινιόδρομου φαίνονται στις παρακάτω εικόνες (Εικόνες 4.59, 4.60).

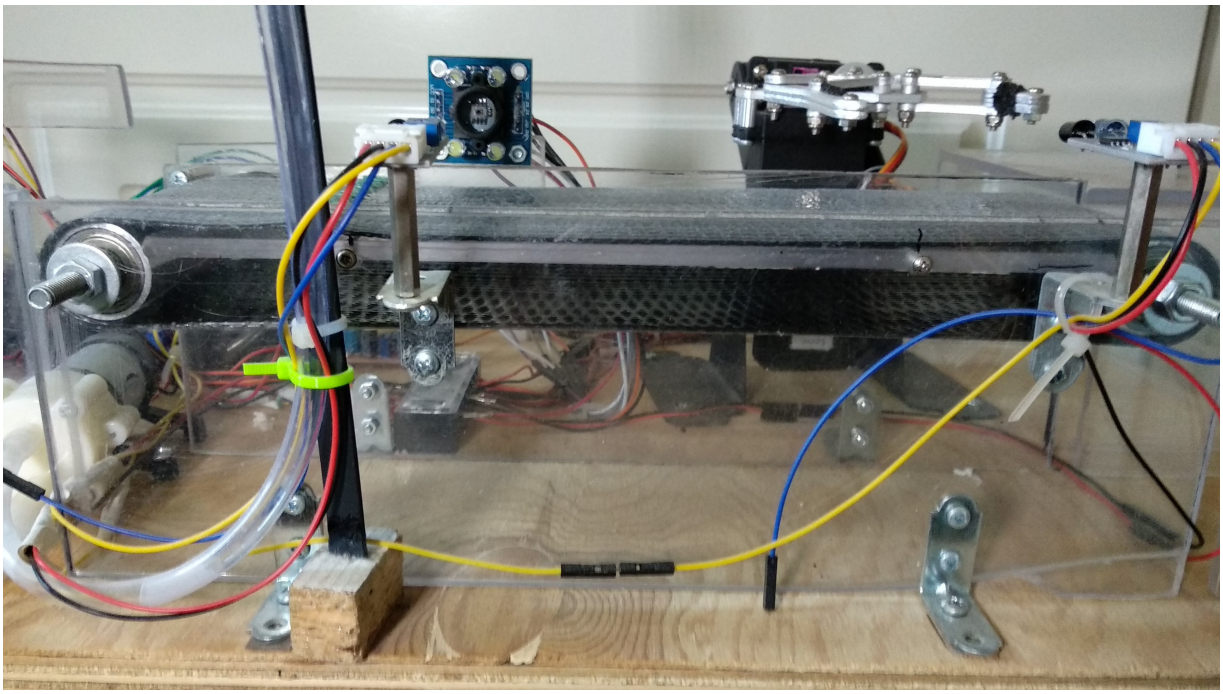



Εικόνα 4.59: Σχέδιο πλαϊνής όψης του ταινιόδρομου




Εικόνα 4.60: Σχέδιο κάτοψης του ταινιόδρομου

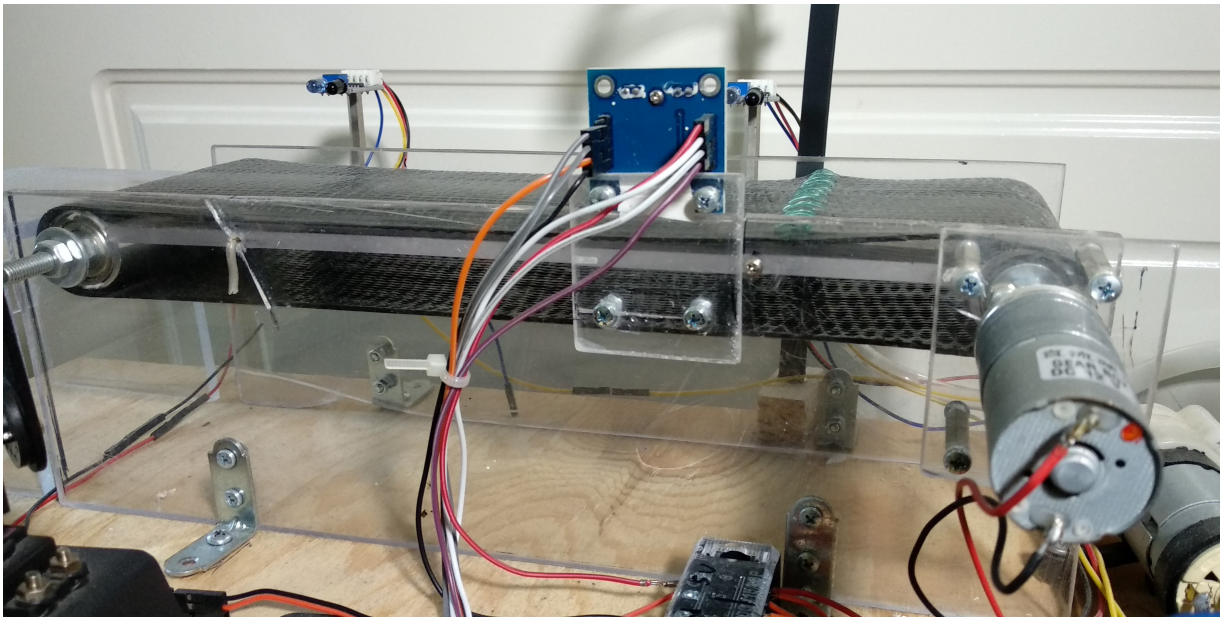
Ο ταινιόδρομος είναι βιδωμένος πάνω σε ένα κομμάτι ξύλο με δύο γωνίες στήριξης σε κάθε πλαϊνό μέρος του. Επίσης στη μία πλευρά είναι εγκατεστημένοι οι δύο IR αισθητήρες με γωνίες στήριξης και αποστάτες (Εικόνα 4.61) καθώς και ένα σίδερο στήριξης για το σωληνάκι της αντλίας που γεμίζει τα δοχεία.



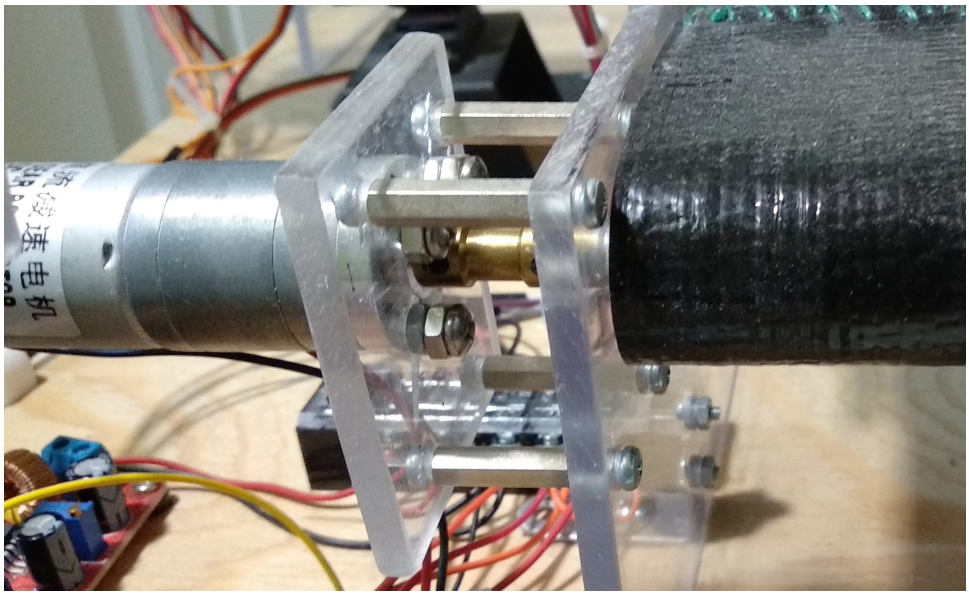
Εικόνα 4.61: Πλευρά του ταινιόδρομου με τους IR αισθητήρες

Στην άλλη πλευρά είναι εγκατεστημένος ο αισθητήρας χρώματος με ένα κομμάτι plexiglass για βάση και βίδες με παξιμάδια καθώς και ο κινητήρας (Εικόνα 4.62). Ο κινητήρας είναι βιδωμένος σε

ένα κομμάτι plexiglass χρησιμοποιώντας την βάση αλουμινίου (Aluminum motor mount F) και συνδέεται με τον ταινιόδρομο με τέσσερις αποστάτες και τον Shaft coupler (Εικόνα 4.63).



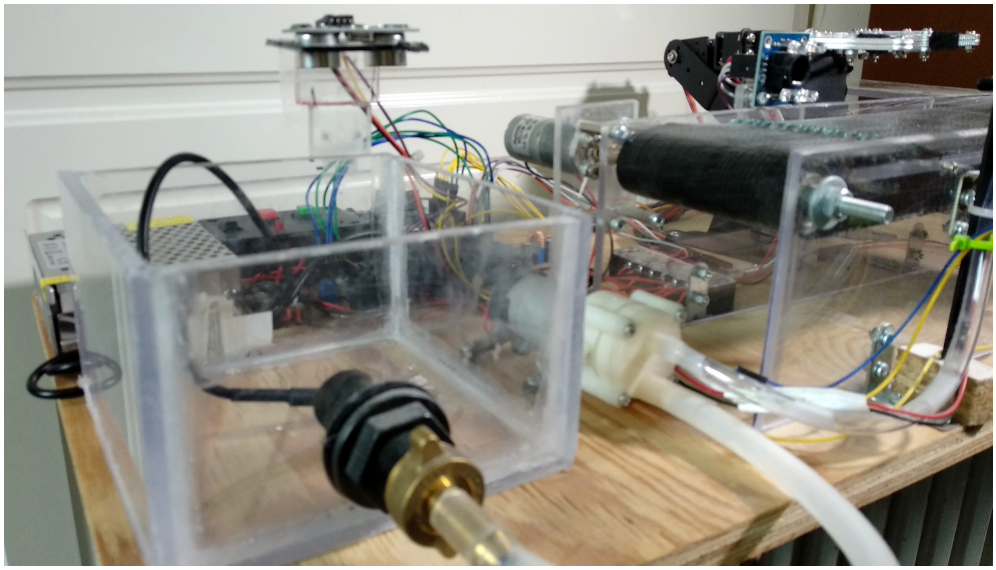
Εικόνα 4.62: Πλευρά του ταινιόδρομου με τον κινητήρα και τον αισθητήρα χρώματος



Εικόνα 4.63: Εγκατάσταση κινητήρα στον ταινιόδρομο

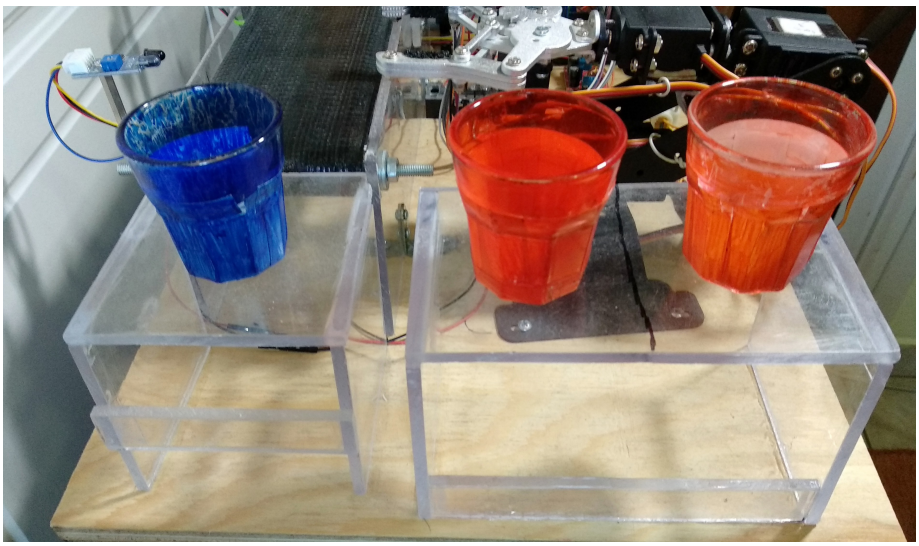
Για την δεξαμενή χρησιμοποιήθηκαν δύο κομμάτια plexiglass 8cm x 12cm, δύο κομμάτια 8cm x 13cm και 1 κομμάτι 13cm x 13cm για τον πάτο. Τα κομμάτια κολλήθηκαν με κόλλα στιγμής και οι ενώσεις σφραγίστηκαν με διάφανη σιλικόνη αδιαβροχοποίησης. Στο ένα πλαϊνό κομμάτι ανοίχτηκε μια τρύπα 21mm για να τοποθετηθεί ο μαστός και το ρακόρ για την παροχή. Εκεί συνδέετε το σωληνάκι με την αντλία και από την αντλία φεύγει το άλλο σωληνάκι το οποίο πάει στον

ταινιόδρομο και είναι δεμένο με tire up στο σίδερο στήριξης. Επίσης σε άλλο κομμάτι τοποθετήθηκε και ο αισθητήρας υπερήχων για την μέτρηση της στάθμης. Μέσα στην δεξαμενή υπάρχει και ο αισθητήρας θερμοκρασίας.



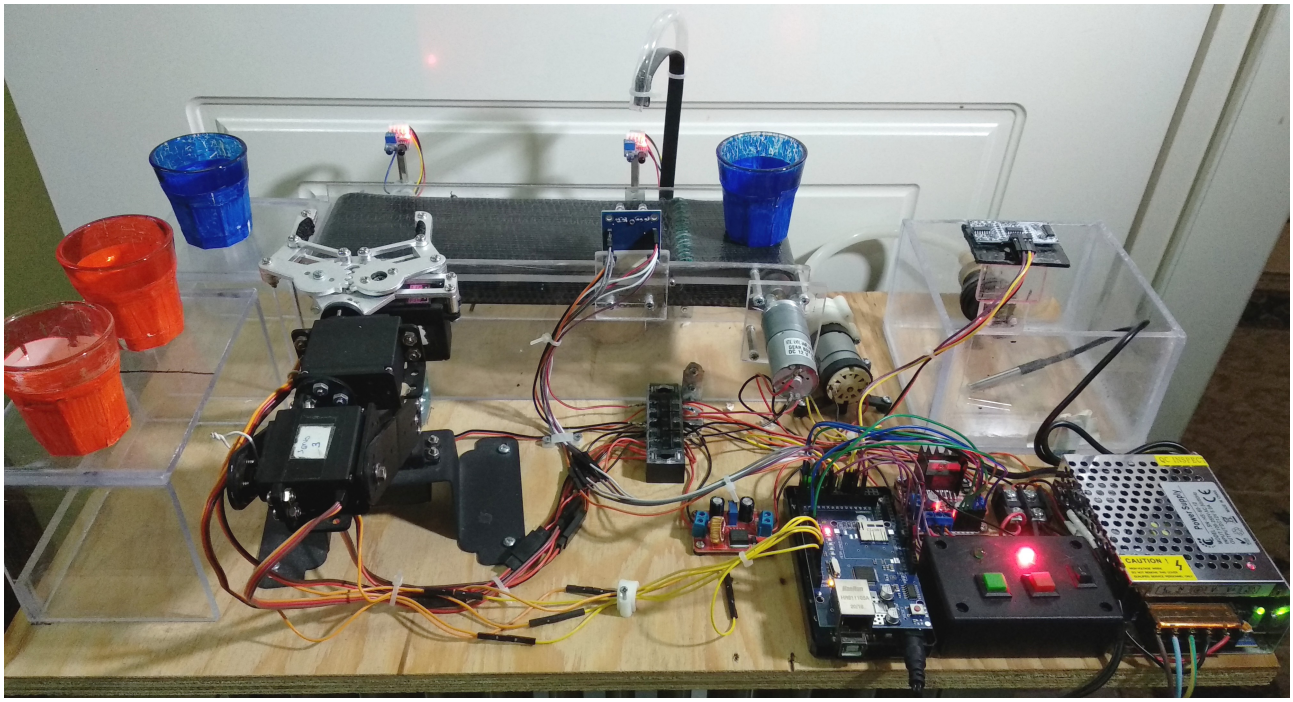
Εικόνα 4.64: Σύστημα δεξαμενής

Κατασκευάστηκαν επίσης δύο “πάγκοι” - θέσεις για τα δοχεία από plexiglass. Ο ένας είναι στο τέλος του ταινιόδρομου για να αποσύρονται τα μπλε δοχεία και ο άλλος είναι δίπλα και έχει δύο θέσεις για να αφήνει ο βραχίονας τα κόκκινα δοχεία. Όλα τα κομμάτια κολλήθηκαν με κόλλα στιγμής μεταξύ τους αλλά και στο ξύλο για να είναι σταθερά.



Εικόνα 4.65: Θέσεις δοχείων από plexiglass

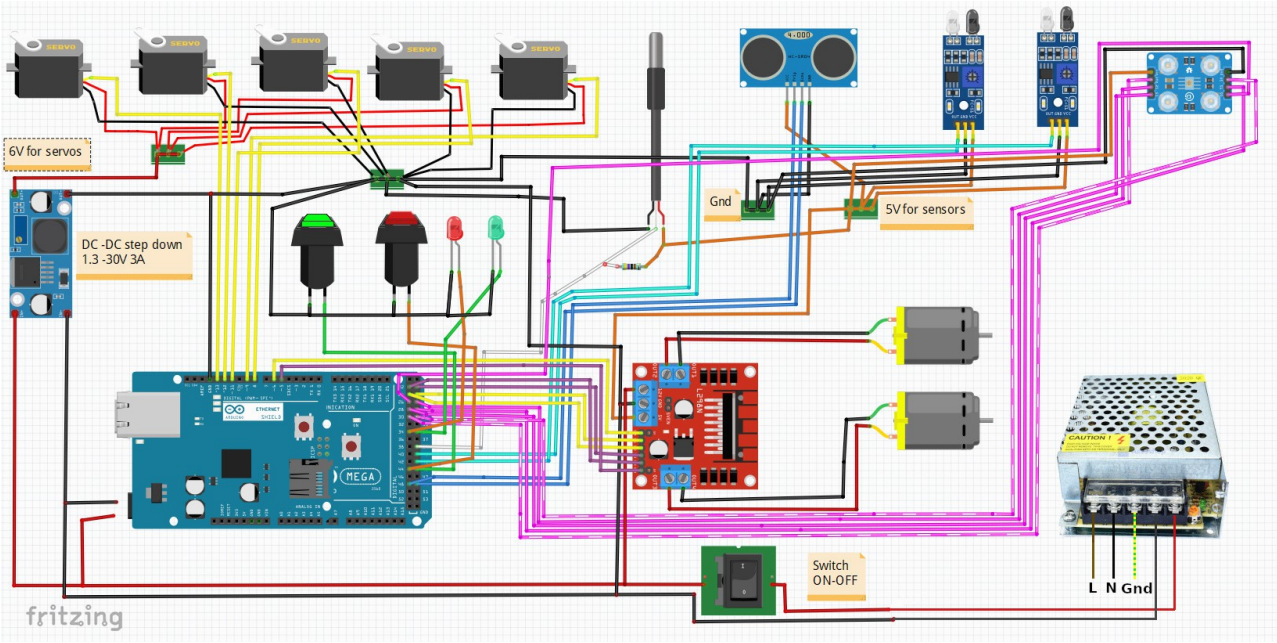
Τέλος όλα τα ηλεκτρονικά βρίσκονται μαζί και είναι βιδωμένα στο ξύλο. Ολόκληρη η κατασκευή φαίνεται στην παρακάτω εικόνα.



Εικόνα 4.66: Ολοκληρωμένη κατασκευή

## 4.4 Κύκλωμα της κατασκευής

Η τροφοδοσία του κυκλώματος ξεκινά από το τροφοδοτικό το οποίο δίνει 11V τάση στον arduino, στον motor driver και στον μετατροπέα τάσης αφού πρώτα περάσει από τον γενικό διακόπτη. Ο μετατροπέας μειώνει την τάση στα 6V για τους σερβοκινητήρες. Ο motor driver τροφοδοτεί με 11V τον κινητήρα και την αντλία και δίνει και 5V στον ακροδέκτη +5V, αφού η τάση που τροφοδοτείτε ο οδηγός είναι μικρότερη των 12V. Με τα 5V τροφοδοτούνται όλοι οι αισθητήρες. Οι τάσεις αυτές των 6V, 5V και της γείωσης πηγαίνουν στην μεγάλη κλέμμα της κατασκευής για να μπορούν να μοιράζονται.  Το τελικό κύκλωμα φαίνεται στην παρακάτω εικόνα.




Εικόνα 4.67: Τελικό κύκλωμα της κατασκευής

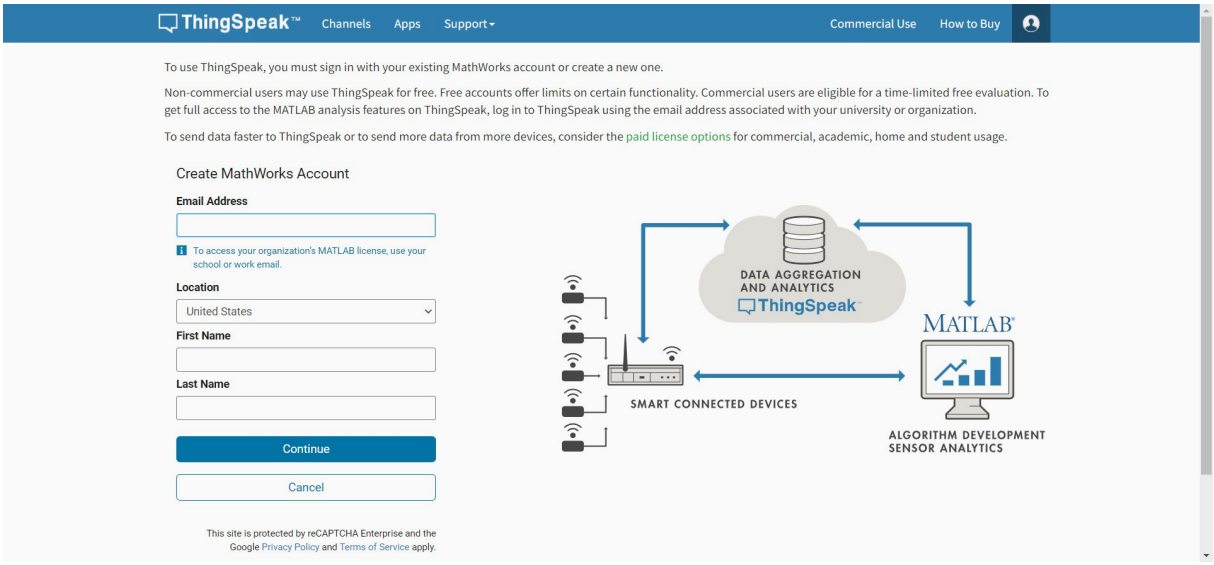
# Κεφάλαιο 5ο Προγραμματισμός της κατασκευής

## 5.1 Δημιουργία Λογαριασμού και καναλιού ThingSpeak

Πριν ασχοληθούμε με τον προγραμματισμό του arduino θα πρέπει να δημιουργήσουμε έναν λογαριασμό στο ThingSpeak και συγκεκριμένα στην MathWorks και ύστερα να φτιάξουμε ένα κανάλι, γιατί θα χρειαστούμε κάποια πράγματα όπως channel id και apikeys, τα οποία θα πρέπει να εισάγουμε στον κώδικα του arduino.

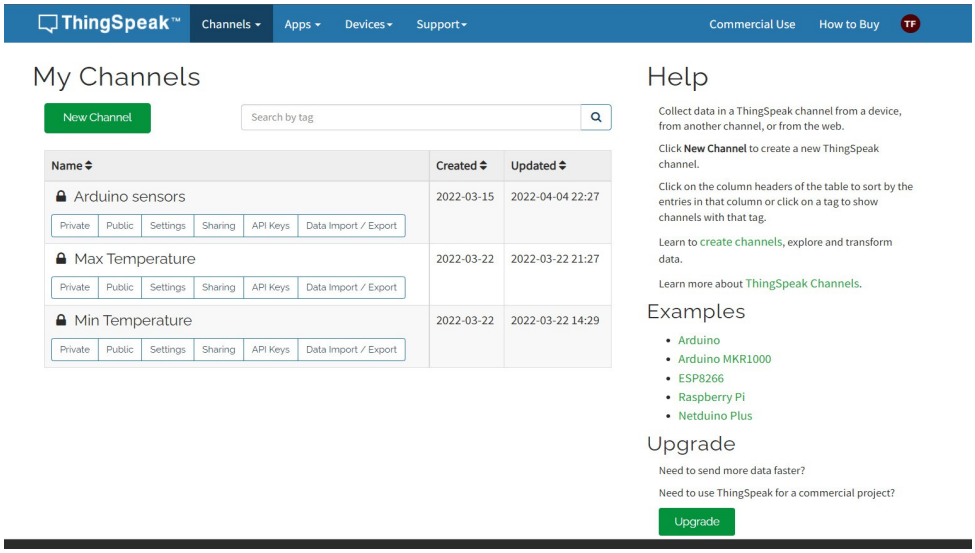
Έτσι από την αρχική σελίδα του ThingSpeak πατάμε πάνω δεξιά το Sign In και στην επόμενη σελίδα πατάμε Create One.

Στην επόμενη σελίδα (Εικόνα 5.1), συμπληρώνουμε τα στοιχεία που μας ζητάει και πατάμε continue.



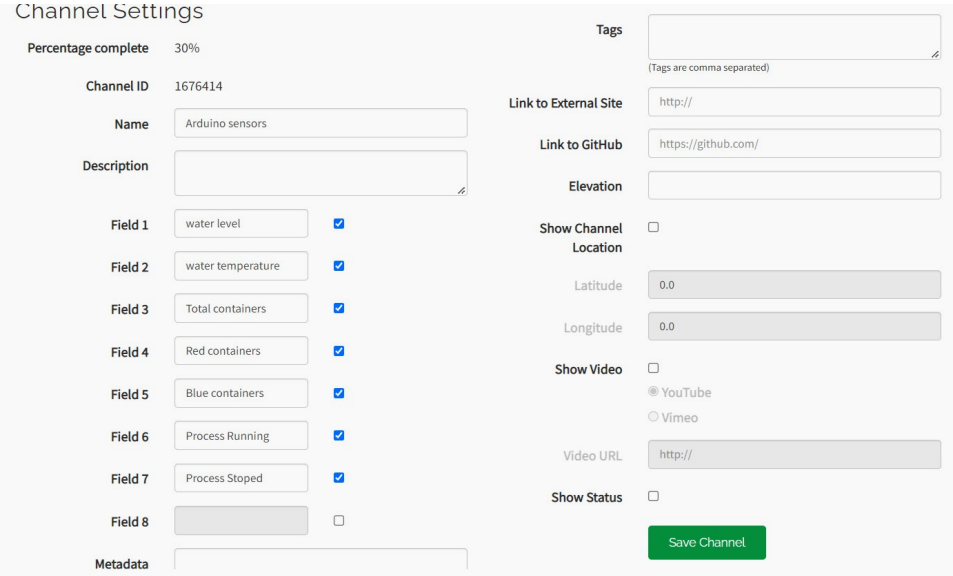

Εικόνα 5.1: Σελίδα δημιουργίας λογαριασμού ThingSpeak [Πηγή]

Αφού δημιουργήσουμε τον λογαριασμό πηγαίνουμε στην καρτέλα channels / my channels (Εικόνα 5.2). Εκεί δημιουργούμε νέα κανάλια ή βλέπουμε τα υπάρχοντα. Για να δημιουργήσουμε νέο κανάλι επιλέγουμε το πράσινο κουμπί New Channel.



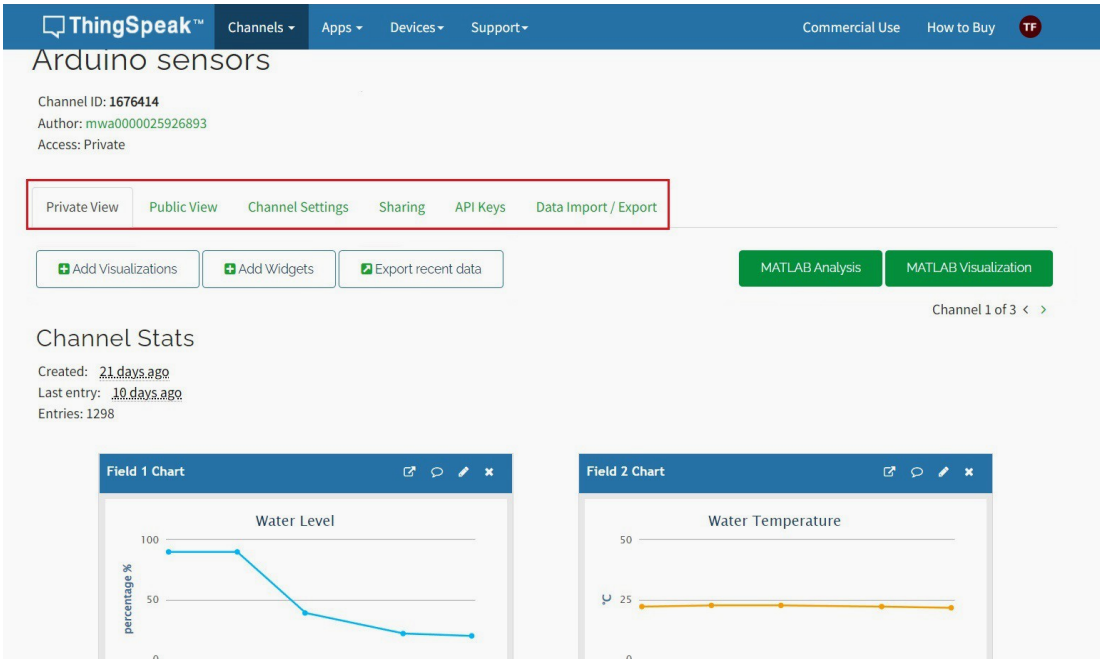

Εικόνα 5.2: Σελίδα με τα υπάρχοντα κανάλια

Η επόμενη σελίδα που θα μας ανοίξει είναι οι ρυθμίσεις του καναλιού. (Εικόνα 5.3) Εκεί δίνουμε όνομα στο κανάλι, προσθέτουμε πεδία για κάθε αισθητήρα που θα στέλνει δεδομένα, καθώς άλλες ρυθμίσεις που θα δούμε παρακάτω.



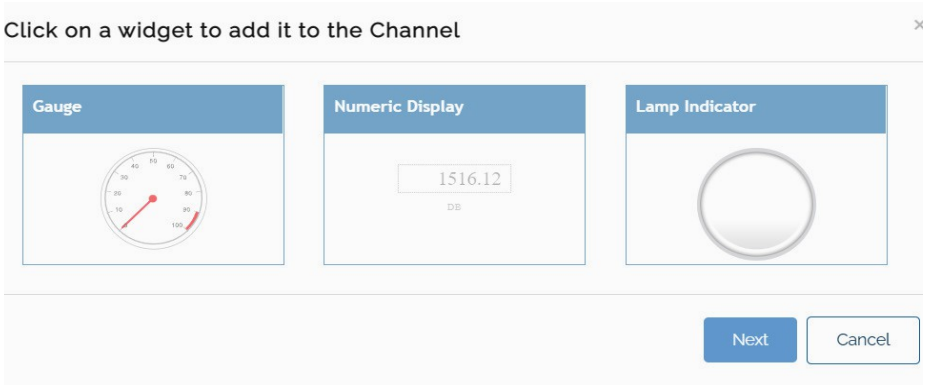

Εικόνα 5.3: Σελίδα ρυθμίσεων καναλιού

Το κανάλι της παραπάνω φωτογραφίας είναι αυτό που χρησιμοποιήθηκε για την εργασία αυτή. Όπως βλέπουμε στην εικόνα μετά το Channel ID δίνουμε όνομα στο κανάλι και από κάτω μπορούμε να δώσουμε και περιγραφή. Στη συνέχεια βλέπουμε 8 πεδία τα οποία μπορούμε να τα ενεργοποιήσουμε από το κουτάκι που έχουν δεξιά τους και να τους δώσουμε όνομα. Κάθε πεδίο αντιστοιχεί σε έναν αισθητήρα και σε αυτό θα ανεβαίνουν και θα προβάλλονται τα δεδομένα. Αν δεν μας φτάνουν τα 8 πεδία θα πρέπει να ανοίξουμε και δεύτερο κανάλι. Μετά έχουμε το πεδίο metadata στο οποίο μπορούμε να δώσουμε πληροφορίες για τα δεδομένα του καναλιού και δεδομένα JSON, XML,CSV. Στη συνέχεια είναι το πεδίο tags όπου μπορούμε να γράψουμε λέξεις κλειδιά που προσδιορίζουν το κανάλι. Στα επόμενα 2 πεδία μπορούμε να δώσουμε link για εξωτερικό site που θα έχει πληροφορίες για το κανάλι μας ή link για το github αν έχουμε κάποιο κώδικα αποθηκευμένο εκεί. Τέλος τα επόμενα 3 πεδία Latitude, Longitude και Elevation είναι για δεδομένα τοποθεσίας. Όταν τελειώσουμε με τις ρυθμίσεις πατάμε save channel. Το κανάλι έχει εισάγει αυτόματα άδεια γραφήματα όσα τα πεδία που έχουμε δημιουργήσει. Στα γραφήματα αυτά θα εμφανιστούν δεδομένα αυτόματα όταν συνδεθεί ο arduino με το thingspeak. Στην εικόνα 5.4 φαίνονται τα 2 πρώτα γραφήματα του καναλιού της εργασίας αυτής αλλά με δεδομένα.



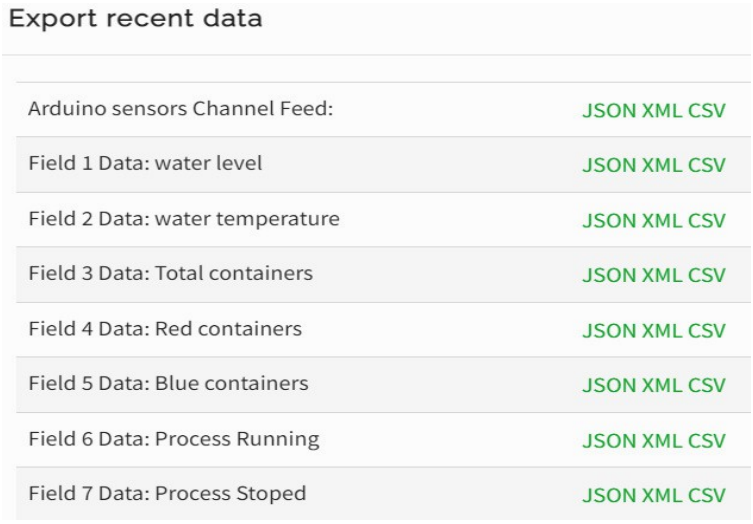

Εικόνα 5.4: Σελίδα γραφημάτων του καναλιού

Υπάρχουν επίσης κάποιες καρτέλες με το κόκκινο περίγραμμα. Η πρώτη είναι η Private view την οποία μόνο εμείς μπορούμε να βλέπουμε, και φαίνονται τα δύο γραφήματα και οι επιλογές Add Visualizations, Add Widgets, Export recent data, Matlab Analysis και Matlab visualizations. Με την επιλογή Add Visualizations εισάγουμε στο κανάλι νέα γραφήματα που έχουμε δημιουργήσει με Matlab Analysis ή Matlab visualizations. Με το Add Widgets μπορούμε να εισάγουμε γραφικά στοιχεία τύπου Gauge , Lamp Indicator και Numeric Display. (Εικόνα 5.5)




Εικόνα 5.5: Επιλογή Widgets

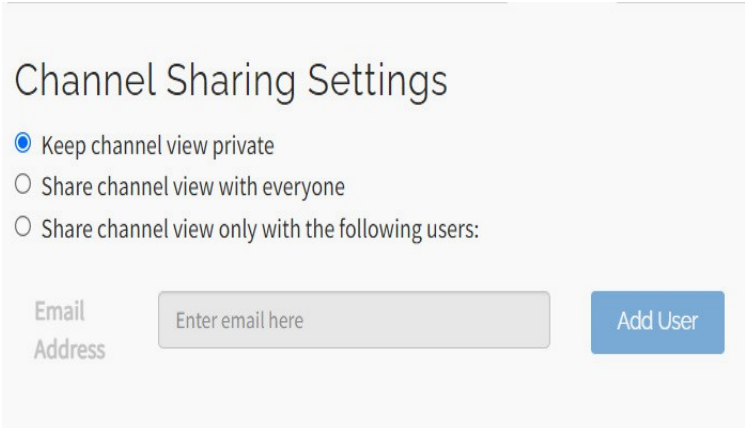
Με την επιλογή Export recent data μπορούμε να εξάγουμε όποια δεδομένα του καναλιού μας σε μορφές JSON, XML και CSV. (Εικόνα 5.6)



Εικόνα 5.6: Καρτέλα εξαγωγής δεδομένων του καναλιού

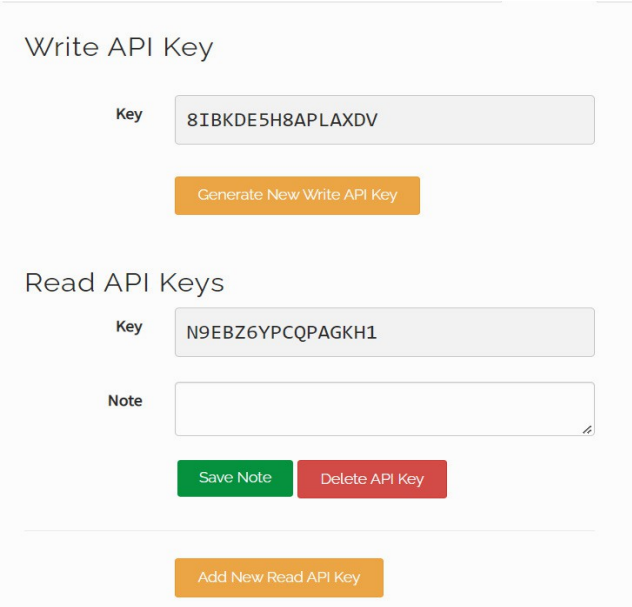
Οι επόμενες δύο επιλογές Matlab Analysis και Matlab visualizations είναι για να κάνουμε ανάλυση σε κάποια δεδομένα ή να δημιουργήσουμε γραφήματα με κώδικα Matlab.

Η δεύτερη καρτέλα είναι η Public view, της οποίας τα γραφήματα μπορούν να τα βλέπουνε όλοι. Η δημόσια προβολή είναι κλειδωμένη αλλά μπορούμε να την ανοίξουμε από την καρτέλα Sharing. Η καρτέλα Channel settings είναι αυτή της εικόνας 5.7 στην οποία κάναμε τις ρυθμίσεις του καναλιού και μπορούμε να κάνουμε και αλλαγές. Στην συνέχεια είναι η Sharing στην οποία έχουμε τις επιλογές να κάνουμε το κανάλι μας δημόσιο για όλους, να το μοιραστούμε με συγκεκριμένους χρήστες που θα ορίσουμε εμείς, ή να το κρατήσουμε ιδιωτικό.



Εικόνα 5.7: Καρτέλα Channel sharing settings

Μετά έχουμε την καρτέλα APIkeys η οποία περιέχει 2 κλειδιά, το WriteAPIkey που το χρειαζόμαστε για να γράψουμε στο κανάλι και το ReadAPIkey για να διαβάζουμε από το κανάλι. Αυτά τα κλειδιά θα τα χρειαστούμε και για την σύνδεση με τον arduino.



Εικόνα 5.8: Καρτέλα API keys

Τέλος στην καρτέλα Data Import / Export μπορούμε να εισάγουμε ή να εξάγουμε δεδομένα στο κανάλι μας.

## 5.2 Προγραμματισμός του Arduino

Η δομή ενός προγράμματος χωρίζετε σε τρία μέρη. Για αρχή πρέπει να γίνει δήλωση των μεταβλητών – σταθερών και των βιβλιοθηκών. Υπάρχουν αρκετοί τύποι μεταβλητών αλλά οι πιο βασικοί είναι:

- bool (παίρνει μόνο τιμές 0 και 1 ή TRUE και FALSE)
- byte (αριθμοί των 8 bit, 0 έως 255 )
- int (ακέραιοι αριθμοί από -32.768 έως 32.767 )
- float (δεκαδικοί αριθμοί από -3,4028235E+38 έως 3,4028235E+38 και αποθηκεύονται σαν 32 bit πληροφορίας)
- char (αποθηκεύει έναν χαρακτήρα του ενός byte)
- string (υπάρχουν διάφοροι τρόποι σύνταξης αλλά γενικά αποθηκεύει λέξεις ή πίνακα χαρακτήρων)

Παραδείγματα σύνταξης

```
int ledPin = 2;          //ορίζουμε μια ακέραια μεταβλητή με όνομα ledPin και τιμή 2.
```

```
float SensorValue = 1.567;          //δεκαδική μεταβλητή SensorValue με τιμή 1.567
```

```
String message = "Hello World";       //μεταβλητή message που αποθηκεύει την λέξη
                                      // Hello World
```

```
char myString[] = "Hello World";
```

```
char String2[8] = "Arduino";          //Το νούμερο στις αγκύλες είναι ο αριθμός των
                                      // χαρακτήρων και πάντα προσθέτουμε ένα παραπάνω
```

```
char String[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
```

Πίνακας 5.1: Παραδείγματα δήλωσης μεταβλητών στο arduino

Στη συνέχεια ακολουθεί το void setup(). Το void είναι και αυτό ένας τύπος μεταβλητής που χρησιμοποιείτε μόνο για δήλωση συναρτήσεων. Η setup() είναι μια συνάρτηση που τρέχει μόνο μια φορά στην αρχή του προγράμματος ή αν γίνει reset στην πλακέτα. Εδώ γίνετε η αρχικοποίηση καταστάσεων και μεταβλητών και γράφουμε τον κώδικα που θα τρέξει μόνο μια φορά. Η εντολή που χρησιμοποιούμε για να ορίσουμε εισόδους και εξόδους είναι η:

- pinMode (pin,mode); // καθορίζει αν το pin θα είναι είσοδος ή έξοδος.

Τα mode είναι τρία, INPUT, OUTPUT και INPUT_PULLUP όπου γίνετε ενεργοποίηση της εσωτερικής αντίστασης pullup που έχει το arduino και το χρησιμοποιούμε συνήθως στα buttons αν δεν έχουμε συνδέσει κάποια εξωτερική αντίσταση.

Μετά την συνάρτηση setup() έχουμε την loop() η οποία δηλώνετε ως void loop(). Εδώ γράφουμε τον κώδικα που θα τρέχει συνέχεια όσο το arduino έχει τροφοδοσία.

Κάποιες βασικές εντολές που χρησιμοποιούνται και στο void setup() αλλά και στο void loop() είναι οι εξής:

- digitalWrite (pin,value);

Η εντολή αυτή δίνει στο pin που έχουμε ορίσει την τιμή HIGH δηλαδή 5V ή 3.3V ανάλογα την πλακέτα, ή την τιμή LOW δηλαδή 0V.

- variable_name = digitalRead (pin);

Με την εντολή αυτή διαβάζουμε την τιμή από το pin (HIGH ή LOW) και αποθηκεύετε στην μεταβλητή variable_name.

```
int LedPin = 4;
int ButtonPin = 7;
int ButtonValue = 0;

void setup() {
  pinMode(LedPin, OUTPUT);  //ορίζει το pin 4 σαν έξοδο
  pinMode(ButtonPin, INPUT_PULLUP);     // ορίζει το pin 7 σαν είσοδο και
ενεργοποιείτε και η εσωτερική αντίσταση.
}
void loop() {
  ButtonValue = digitalRead(ButtonPin);    // διαβάζει την τιμή του pin 7
  digitalWrite(LedPin, ButtonValue);  // δίνει στο led την τιμή που διάβασε
από το pin 7, HIGH ή LOW
}
```

- analogWrite (pin,value);

Δίνει μια αναλογική τιμή PWM (από 0 έως 255) στο pin που έχουμε ορίσει.

- delay (ms);

Σταματά την εκτέλεση του προγράμματος για όσο χρόνο σε milliseconds υπάρχει μέσα στην παρένθεση.

- Millis()

Μας επιστέφει τον χρόνο σε millisecond που έχει περάσει από την ενεργοποίηση της λειτουργίας της πλακέτας.

```
unsigned long myTime;

void setup() {
  Serial.begin(9600);  //Ενεργοποιούμε την σειριακή επικοινωνία και ορίζουμε την
ταχύτητα σε bits per second
}
void loop() {
  Serial.print("Time: "); //εκτυπώνει στο Serial monitor την λέξη Time:
  myTime = millis();  //εκτελείτε η εντολή millis και αποθηκεύει τον χρόνο στην
μεταβλητή myTime
  Serial.println(myTime); //εκτυπώνει το περιεχόμενο της μεταβλητής myTime
  delay(1000);           // περιμένει 1 sec
}
```

- Συνθήκη ελέγχου if και if...else

if (συνθήκη) { εντολές } , ελέγχει αν ισχύει η συνθήκη στην παρένθεση και αν ισχύει εκτελεί τις εντολές μέσα στις αγκύλες, αλλιώς προχωράει παρακάτω.

if (συνθήκη) {εντολές} else if (συνθήκη 2) {άλλες εντολές} else {άλλες εντολές}, με τον τρόπο αυτό έχουμε μεγαλύτερο έλεγχο της ροή του κώδικα.

```
int PinButton_1 = 4;
```

```
int PinButton_2 = 5;
int greenLedPin = 6;
int redLedPin = 7;

void setup() {
pinMode(PinButton_1,INPUT_PULLUP);
pinMode(PinButton_2,INPUT_PULLUP);
pinMode(greenLedPin,OUTPUT);
pinMode(redLedPin,OUTPUT);
digitalWrite(greenLedPin,LOW); // αρχικοποιούμε τις καταστάσεις των led
digitalWrite(redLedPin,LOW);   // να είναι σβηστά
}

void loop() {
if (digitalRead(PinButton_1) == HIGH) { //ελέγχει αν έχει πατηθεί το button1
   digitalWrite(greenLedPin,HIGH);   //αν ισχύει η συνθήκη τότε ανάβει το πράσινο
led και
   digitalWrite(redLedPin,LOW);     // το κόκκινο led παραμένει σβηστό
}else                                // αλλιώς...
  if (digitalRead(PinButton_2) == HIGH) { //ελέγχει αν έχει πατηθεί το button2
     digitalWrite(greenLedPin,LOW);   //αν ισχύει η συνθήκη τότε σβήνει το πράσινο
led και
     digitalWrite(redLedPin,HIGH);   // το κόκκινο led ανάβει
}else {                             //αλλιώς...
   digitalWrite(greenLedPin,LOW); // και τα 2 σβήνουν
   digitalWrite(redLedPin,LOW);
}
}
```

Περισσότερες εντολές και παραδείγματα υπάρχουν στην σελίδα του arduino: Language Reference

### 5.2.1 Βιβλιοθήκες που χρησιμοποιήθηκαν

Οι βιβλιοθήκες που χρησιμοποιήθηκαν στο πρόγραμμα είναι οι εξής:

- **VarSpeedServo.h** → Χρησιμοποιείτε για τον έλεγχο σερβοκινητήρων. Επιτρέπει να χρησιμοποιούμε έως 8 σερβοκινητήρες οι οποίοι μπορούν να κινούνται ασύγχρονα, με δυνατότητα ρύθμιση της ταχύτητας και με δυνατότητα να περιμένουν να ολοκληρωθεί η κίνηση κάποιου σερβοκινητήρα.
- **OneWire.h** → Επιτρέπει την πρόσβαση σε συσκευές 1-wire κατασκευασμένες από την Maxim/Dallas όπως τον αισθητήρα θερμοκρασίας της κατασκευής αυτής.
- **DallasTemperature.h** → Βιβλιοθήκη plug and go για τα ολοκληρωμένα (ICs) θερμοκρασίας της Maxim/Dallas. Υποστηρίζει τις συσκευές DS18B20, DS18S20, DS1822, DS1820, MAX31820.
- **SPI.h** → Χρησιμοποιείτε για την επικοινωνία του arduino με διάφορες SPI συσκευές, όπως το ethernet shield, σαν master ή για την επικοινωνία μεταξύ δύο μικροελεγκτών.

- **Ethernet.h** → Επιτρέπει την σύνδεση του arduino στο διαδίκτυο είτε σαν server είτε σαν client. Το arduino επικοινωνεί με το shield μέσω του διαύλου SPI.
- **ThingSpeak.h** → Επιτρέπει στον arduino ή σε άλλες συσκευές όπως ESP8266 και ESP32 να γράφει και να διαβάζει δεδομένα από και προς τον ThingSpeak.

Οι βιβλιοθήκες συμπεριλαμβάνονται στην αρχή του προγράμματος.

```
#include <VarSpeedServo.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SPI.h>
#include <Ethernet.h>
#include <ThingSpeak.h>
```

### 5.2.2 Αρχικοποιήσεις μεταβλητών, ονοματοδοσία Pin και δηλώσεις Ethernet και Thingspeak

Στη συνέχεια δηλώνουμε σε ποια pin του arduino είναι συνδεδεμένες οι συσκευές και δίνουμε και ονόματα. Επίσης δηλώνουμε κάποιες μεταβλητές για καταστάσεις συσκευών, για το labview και άλλα καθώς και μεταβλητές για το Ethernet και τον Thingspeak.

```
 //temperature sensor
#define ONE_WIRE_BUS 39 //καθορίζουμε την σύνδεση του data pin στον arduino
OneWire oneWire(ONE_WIRE_BUS); //setup a oneWire instance to communicate with any
OneWire devices
DallasTemperature sensors(&oneWire); //Pass our oneWire reference to Dallas
Temperature.
//Ultrasonic sensor
#define trigpin 49
#define echopin 48
//Dc motor
int enB = 5;
int in3 = 23;
int in4 = 22;
//Dc Pump
int enA = 6;
int in1 = 25;
int in2 = 24;
//IR sensor_1
const int pinIRd1 = 41;
const int pinLED1 = 40; //δήλωση του ενσωματωμένου led του αισθητήρα
int IRvalueD1 = 0;  //μεταβλητή για την ψηφιακή τιμή του αισθητήρα
//IR sensor_2
const int pinIRd2 = 43;
const int pinLED2= 42; //δήλωση του ενσωματωμένου led του αισθητήρα
int IRvalueD2 = 0;  //μεταβλητή για την ψηφιακή τιμή του αισθητήρα
//buttons
int StartButtonStatus = 0;
int StopButtonStatus = 0;
// leds
int greenLed = 35;
int redLed = 34;
int greenledstate = 0;
int redledstate = 1;
```

```
//Color sensor
#define S0 26
#define S1 27
#define S2 29
#define S3 28
#define sensorOut 30
#define led 31
int redfrequency = 0;
int greenfrequency = 0;
int bluefrequency = 0;
int ledvalue = 0;
//Servos
VarSpeedServo servo1;
VarSpeedServo servo2;
VarSpeedServo servo3;
VarSpeedServo servo4;
VarSpeedServo servo5;
const int servoPin1 = 9;
const int servoPin2 = 8;
const int servoPin3 = 11;
const int servoPin4 = 12;
const int servoPin5 = 13;
int myarray[80] ; //δήλωση πίνακα μεγέθους 80 ακαίρεων αθιθμών. Εδώ αποθηκεύετε το
χρώμα των δοχείων με 0 ή 1.
int N = 0;  //δείκτης θέσης για την εγγραφή στον πίνακα.
int K = 0;  //δείκτης θέσης για την ανάγνωση του πίνακα.
int T = LOW; //μεταβλητή για την επιλογή μεταξύ των δύο θέσεων που θα αφήσει τα δοχεία
ο βραχίονας
int Red_color = 0;
int Blue_color = 0;
int Total_containers = 0;
int motorState = 0;
int pumpState = 0;
//Labview
char labview;
String bufer;
//Ethernet
byte mac[] = { 0xFE, 0xAD, 0xBF, 0xBE, 0xEE,0xED};  //δίνουμε μια διεύθυνση mac
στο ethernet shield, που να μην την έχει άλλη συσκευή στο δίκτυο
IPAddress ip(192,168,2,10);   //δίνουμε μια διεύθυνση IP που να μην την έχει άλλη
συσκευή στο ίδιο δίκτυο
EthernetClient client;
//ThingSpeak
unsigned long mySensorsChannelNumber = 1676414;  //Replase this code with your
thingSpeak channel ID
const char * myWriteAPIKey_sensors = "IW0YRSWAFG9AX7U0";  //Replace with your
channel Write API Key
long lastWriteTime=0;
```

### 5.2.3 Προσδιορισμός I/O και αρχικοποίηση καταστάσεων

Στη συνέχεια έχουμε το `void setup()`. Εδώ προσδιορίζουμε τον τρόπο που θα λειτουργούν τα pin, δηλαδή αν θα είναι είσοδοι, έξοδοι ή είσοδοι με ενεργοποιημένη την εσωτερική αντίσταση pull up. Επίσης αρχικοποιούμε κάποιες καταστάσεις όπως τις αρχικές θέσεις των σερβοκινητήρων του βραχίονα, το frequency scalling του αισθητήρα χρώματος. Τέλος αρχικοποιούμε τις βιβλιοθήκες ethernet και thingspeak και ανοίγουμε την σειριακή επικοινωνία.

```
void setup() {
//temperature sensor
sensors.begin();  //Ξεκινά την βιβλιοθήκη
//Ultrasonic sensor
pinMode(trigpin, OUTPUT);
pinMode(echopin, INPUT);
 //Dc motor
pinMode(enB, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
// Dc Pump
pinMode(enA, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
//buttons
pinMode(44, INPUT_PULLUP);
pinMode(45, INPUT_PULLUP);
//leds
pinMode(greenLed, OUTPUT);
pinMode(redLed, OUTPUT);
digitalWrite(greenLed, LOW);
digitalWrite(redLed,HIGH);
//IR sensor_1
pinMode(pinIRd1,INPUT);
pinMode(pinLED1,OUTPUT);
//IR sensor_2
pinMode(pinIRd2,INPUT);
pinMode(pinLED2,OUTPUT);
//color sensor
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(sensorOut, INPUT);
pinMode  (led,OUTPUT);
digitalWrite(led,LOW);
// Setting frequency-scaling to 20%
digitalWrite(S0,HIGH);
digitalWrite(S1,LOW);
//Robotic Arm Servos
servo1.attach(servoPin1);
servo1.write(73,80,false);
servo2.attach(servoPin2);
servo2.write(160,150,false);
servo3.attach(servoPin3);
servo3.write(0,150,false);
servo4.attach(servoPin4);
servo4.write(90,150,false);
servo5.attach(servoPin5);
servo5.write(110,100,false);
Serial.begin(9600);   //Άνοιγμα της σειριακής επικοινωνίας
Ethernet.begin(mac, ip); //Αρχικοποιεί τη βιβλιοθήκη Ethernet και τις ρυθμίσεις
δικτύου.
ThingSpeak.begin(client); //Αρχικοποιεί τη βιβλιοθήκη ThingSpeak
delay(1000);
}
```

### 5.2.4 Μετρήσεις αισθητήρων και σενάρια αυτοματισμών

Στο `void loop()` θα γράψουμε τον κώδικα που θα τρέχει συνέχεια για όσο το arduino έχει τροφοδοσία και δεν έχει γίνει reset στη πλακέτα. Στην αρχή υπάρχει ο κώδικας που αφορά τις μετρήσεις των αισθητήρων για την στάθμη της δεξαμενής και για την θερμοκρασία, καθώς και ο υπολογισμός του συνολικού αριθμού δοχείων.

```
float duration, distance, level; //δήλωση μεταβλητών για τον υπολογισμό της στάθμης
//Μέτρηση και υπολογισμός στάθμης της δεξαμενής
digitalWrite(trigpin, HIGH);
delayMicroseconds(10);
digitalWrite(trigpin, LOW);
duration = pulseIn(echopin,HIGH);
distance = duration/51;
distance = distance-3.2; // offset corection
distance = 10-distance;
level = 10*distance;     // distance in % , 0 - 100%
//Μέτρηση θερμοκρασίας
sensors.requestTemperatures();
float tempC = sensors.getTempCByIndex(0); //δήλωση μεταβλητής που θα αποθηκεύετε η θερμοκρασία
//Υπολογισμός των συνολικών δοχείων
Total_containers = Red_color + Blue_color;
```

Στη συνέχεια έχουμε τα σενάρια των αυτοματισμών. Αν πατηθεί το button stop, θέλουμε να ανάβει το κόκκινο led και να σταματάει η λειτουργία του κινητήρα και της αντλίας. Επίσης ενημερώνονται οι καταστάσεις των led, του κινητήρα και της αντλίας. Τα button αυτά όταν είναι πατημένα δίνουν σήμα LOW δηλαδή 0.

```
if (StopButtonStatus == digitalRead(44)){ //Διαβάσει το pin του button και ελέγχει την συνθήκη
  delay(10);
  digitalWrite(redLed, HIGH);
  digitalWrite(greenLed, LOW);
  greenledstate = 0;
  redledstate = 1;
  motorState = 0;
  pumpState = 0;
  digitalWrite(in3, LOW);     //Τα pin του κινητήρας
  digitalWrite(in4, LOW);
  digitalWrite(in1, LOW);     //Τα pin της αντλίας
  digitalWrite(in2, LOW);
  }
```

Αν πατηθεί το button start και κανένας από τους δύο IR αισθητήρες δεν έχει ανιχνεύσει αντικείμενο και η στάθμη της δεξαμενής είναι μεγαλύτερη ή ίση του 30%, θέλουμε να ανάβει το πράσινο led, να ξεκινά ο κινητήρας δεξιόστροφα και με συγκεκριμένη ταχύτητα και να ενημερώνονται οι αντίστοιχες καταστάσεις. Οι IR αισθητήρες όταν ανιχνεύσουν αντικείμενο δίνουν σήμα LOW.

```
//Διάβασμα των τιμών των pin από το button και τους αισθητήρες και έλεγχος της συνθήκης
if (StartButtonStatus == digitalRead(45)&& digitalRead(pinIRd1) == HIGH &&
digitalRead(pinIRd2) == HIGH && level >= 30){    //Το && είναι η λογική πράξη and
  delay(10);
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, HIGH);
  greenledstate = 1;
  redledstate = 0;
  motorState = 1;
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 150);
}
```

Στη συνέχεια αν υπάρχει κάποιο δοχείο στον πρώτο IR αισθητήρα που βρίσκετε στην αρχή του ταινιόδρομου, θέλουμε να αναγνωριστεί το χρώμα του από τον αισθητήρα χρώματος για να εκτελεστούν κάποια σενάρια με βάση το χρώμα. Σε περίπτωση όμως που κάποιο δοχείο βρίσκετε εκεί σταματημένο λόγο χαμηλής στάθμης της δεξαμενής δεν θέλουμε να αναγνωριστεί το χρώμα του αν δεν πατηθεί το button start. Για να γίνει αυτό ελέγχετε αν ο IR αισθητήρας έχει ανιχνεύσει δοχείο και αν το πράσινο led είναι αναμμένο ή το button start είναι πατημένο. Όταν το πράσινο led είναι αναμμένο σημαίνει ότι η διεργασία εκτελείτε κανονικά, άρα τότε θέλουμε να αναγνωριστεί το χρώμα του δοχείου, αλλά αν είναι σβηστό (τότε θα είναι αναμμένο το κόκκινο) σημαίνει ότι η διεργασία έχει σταματήσει, οπότε για να γίνει έλεγχος για το χρώμα θα πρέπει να πατηθεί το button start.

Στη συνέχεια συγκρίνονται οι τιμές των συχνοτήτων των φωτοδιόδων για να δούμε αν το χρώμα του δοχείου είναι κόκκινο ή μπλε και εκτελείτε αντίστοιχα κάποια από τις παρακάτω συνθήκες. Αν το χρώμα είναι κόκκινο καλείτε μια συνάρτηση με όνομα PumpEn για το γέμισμα των δοχείων και την αποθήκευση του χρώματος. Η συνάρτηση αυτή δηλώνετε ως void PumpEn() και γράφετε έξω από το `void loop()`. Η συνάρτηση αυτή περιγράφετε σε επόμενη ενότητα. Αν το χρώμα του δοχείου είναι μπλε, καταγράφετε η τιμή 0 στην θέση του πίνακα που δείχνει η μεταβλητή N, ύστερα αυξάνετε η τιμή του δείκτη και του αριθμού των μπλε δοχείων και ξεκινά ο κινητήρας.

```
//Έλεγχος της συνθήκης. Το σύμβολο || είναι η λογική πράξη or
if (digitalRead(pinIRd1) == LOW && (greenledstate == 1 || digitalRead(45) ==
LOW)) {
  delay(200);   //Περιμένει ώστε να σταματήσει στο κατάλληλο σημείο
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, HIGH);
  greenledstate = 1;
  redledstate = 0;
  motorState = 0;
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  digitalWrite (led,HIGH);
  delay(200);
 //Θέτουμε την κόκκινη φωτοδίοδο να διαβαστεί
```

```
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  redfrequency = pulseIn(sensorOut, LOW);  //Διαβάζετε η συχνότητα εξόδου
  delay(100);
  digitalWrite(S2,HIGH);  //Θέτουμε την πράσινη φωτοδίοδο να διαβαστεί
  digitalWrite(S3,HIGH);
  greenfrequency = pulseIn(sensorOut, LOW); //Διαβάζετε η συχνότητα εξόδου
  delay(100);
  digitalWrite(S2,LOW);  //Θέτουμε την μπλε φωτοδίοδο να διαβαστεί
  digitalWrite(S3,HIGH);
  bluefrequency = pulseIn(sensorOut, LOW);  //Διαβάζετε η συχνότητα εξόδου
  digitalWrite (led,LOW);
     if ((redfrequency>15 && redfrequency< 320)&& (greenfrequency>170 &&
greenfrequency< 830) && (bluefrequency>180 && bluefrequency< 700)) {
       PumpEn();   //Καλείτε η συνάρτηση για το γέμισμα των δοχείων και την αποθήκευση
του χρώματος.
    }else
  if ((bluefrequency>15 && bluefrequency< 300) && (greenfrequency>190 &&
greenfrequency< 640)&& (redfrequency>140 && redfrequency< 780)) {
       myarray[N] = 0;
       ++N;  //Αύξηση του δείκτη
       ++Blue_color;   //Αύξηση του αριθμού των μπλε δοχείων
       digitalWrite(in3, LOW);
       digitalWrite(in4, HIGH);
       analogWrite(enB, 150);
       motorState = 1;
       delay(500); //Περιμένει μέχρι να περάσει το δοχείο από τον IR αισθητήρα για να
μην ξαναγίνει trigger
     }
   }
```

Μετά ελέγχετε αν υπάρχει κάποιο δοχείο στον δεύτερο IR αισθητήρα στο τέλος του ταινιόδρομου. Εάν υπάρχει δοχείο και το πράσινο led είναι αναμμένο, δηλαδή η διεργασία είναι σε λειτουργία, τότε θέλουμε να σταματήσει ο κινητήρας και να διαβαστεί ή τιμή του πίνακα που σε αυτό το σημείο ορίζει ο δείκτης K. Ο δείκτης K υπάρχει για τον λόγο ότι ο δείκτης N που είναι για την εγγραφή στον πίνακα θα βρίσκετε πάντα σε μεγαλύτερη θέση από τον K. Για παράδειγμα αν περάσουν από τον αισθητήρα χρώματος δύο δοχεία διαφορετικού χρώματος στη σειρά, ο δείκτης N θα είναι δύο θέσεις πάνω από τον K. Έτσι όταν φτάσουν στο τέλος του ταινιόδρομου, ο K θα είναι στη σωστή θέση ώστε να διαβαστεί από τον πίνακα το αποθηκευμένο χρώμα του δοχείου που πέρασε πρώτο και μετά το δεύτερο. Αν η τιμή του πίνακα είναι 0 (για το μπλε χρώμα), αυξάνετε η τιμή του δείκτη K και ξεκινά ο κινητήρας για να αποσύρει το δοχείο. Αν όμως η τιμή του πίνακα είναι 1 (για το κόκκινο χρώμα), αυξάνετε η τιμή του δείκτη K, καλείτε μια συνάρτηση με όνομα Move η οποία είναι για την μετακίνηση του δοχείου από τον βραχίονα και μετά αντιστρέφεται η τιμή της μεταβλητής T και ξεκινά ο κινητήρας. Η μεταβλητή T είναι για να επιλέγετε σε ποια από τις δύο θέσεις θα αφήνει το δοχείο ο βραχίονας. Η αρχική τιμή του T είναι LOW που σημαίνει ότι το πρώτο δοχείο θα το αφήσει στην πρώτη θέση, μετά θα αντιστραφεί η τιμή του T ώστε την επόμενη φορά να αφήσει το δοχείο στην δεύτερη θέση. Η συνάρτηση Move περιγράφετε σε επόμενη ενότητα.

```
//Διάβασμα της τιμής του pin του αισθητήρα και έλεγχος της συνθήκης
if (digitalRead(pinIRd2) == LOW &&  greenledstate == 1) {
   digitalWrite(in3, LOW);
   digitalWrite(in4, LOW);
    if (myarray[K]== 0){ //διάβασμα της θέσης του πίνακα που ορίζει ο δείκτης K και
έλεγχος της συνθήκης
       ++K;  //αύξηση του δείκτη
       digitalWrite(in3, LOW);  //ξεκίνημα του κινητήρα
       digitalWrite(in4, HIGH);
       analogWrite(enB, 150);
       motorState = 1;
       delay(500);  //Περιμένει μέχρι να περάσει το δοχείο από τον IR αισθητήρα για
να μην ξαναγίνει trigger
    } else
     if (myarray[K]== 1){
       ++K;
       Move();  //Καλείτε η συνάρτηση
       T= !T;  //Αντιστροφή της τιμής της μεταβλητής.
       digitalWrite(in3, LOW); //ξεκίνημα του κινητήρα
       digitalWrite(in4, HIGH);
       analogWrite(enB, 150);
       motorState = 1;
       }
   }
```

5.2.5 Σειριακή επικοινωνία με το Labview

Για την επικοινωνία με το Labview θα πρέπει αρχικά να έχει γίνει σύνδεση από την πλευρά του labview με τα κατάλληλα στοιχεία που θα αναλυθούν σε επόμενη ενότητα. Από την πλευρά του arduino θα πρέπει να ανοίξουμε την σειριακή επικοινωνία και να ορίσουμε το baud rate. Αυτό έγινε στο void setup() με την εντολή Serial.begin(9600). Στη συνέχεια πρέπει να βρούμε έναν τρόπο να διαβάζουμε τα strings που στέλνει το Labview.  Για να γίνει αυτό ορίσαμε στην αρχή του προγράμματος μια μεταβλητή τύπου χαρακτήρα (char) που την ονομάσαμε labview και μια μεταβλητή τύπου string που την ονομάσαμε bufer. Την μεταβλητή bufer την γράψαμε με ένα (f) γιατί αλλιώς πιάνετε για εντολή του arduino.

Στη συνέχεια πρέπει να ελέγξουμε αν υπάρχουν δεδομένα που έχει στείλει το Labview στην σειριακή θύρα. Τα δεδομένα που στέλνει είναι συμβολοσειρές (strings) από τις οποίες διαβάζονται ένας ένας οι χαρακτήρες στη μεταβλητή labview και προστίθενται στη μεταβλητή “bufer” σχηματίζοντας την λέξη (string) . Στη συνέχεια ελέγχουμε αν η μεταβλητή “bufer” έχει την λέξη “sensors”. Τότε το arduino γράφει στην σειριακή θύρα τιμές των μετρήσεων των αισθητήρων και των διαφόρων καταστάσεων.

```
if (Serial.available() > 0)  // Ελέγχει αν υπάρχουν δεδομένα στην σειριακή θύρα
{
delay(100);
while (Serial.available() > 0) // Για όσο υπάρχουν δεδομένα
{
```

```
labview = Serial.read();  //η μεταβλητή labview διαβάζει έναν έναν τους χαρακτήρες
που στέλνει το LabView και
bufer += labview;   //τους σώζει στην μεταβλητή bufer σχηματίζοντας την λέξη
}
if (bufer == "sensors\n") {    // Εάν ο bufer έχει την λέξη sensors
  Serial.println(level);      // θα τυπώσει τις τιμές των αισθητήρων που θα διαβάσει,
  Serial.println(tempC);     // ώστε να τις διαβάσει το LabView με το Visa Read
  Serial.println(greenledstate);  // Το \n που υπάρχει είναι ένας χαρακτήρας ascii
για αλλαγή γραμμής
  Serial.println(redledstate);
  Serial.println(Total_containers);
  Serial.println(Red_color);
  Serial.println(Blue_color );
  Serial.println(pumpState);
  Serial.println(motorState);
  bufer = "";
   }
```

Το " \n " όπως βλέπουμε και στον παραπάνω κώδικα, είναι ένας χαρακτήρας ASCI για αλλαγή γραμμής. Στο Labview στέλνουμε κάθε λέξη βάζοντας και το αντίστοιχο σύμβολο Line Feed που θα δούμε σε επόμενη ενότητα για να δηλώσουμε αλλαγή γραμμής. Αυτό το κάνουμε για να δείξουμε που τελειώνει η λέξη. Έτσι και εδώ γράφουμε τον αντίστοιχο χαρακτήρα, ώστε να ξέρει μέχρι που θα διαβάσει το arduino.

Μετά ελέγχουμε αν η μεταβλητή "bufer" έχει την λέξη "stop". Τότε σταματά ο κινητήρας και η αντλία και ανάβει το κόκκινο led ενώ το πράσινο σβήνει. Επίσης ενημερώνονται οι καταστάσεις.

```
else if (bufer == "stop\n") {
       digitalWrite(redLed, HIGH);
       digitalWrite(greenLed, LOW);
       greenledstate = 0;
       redledstate = 1;
       motorState = 0;
       pumpState = 0;
       digitalWrite(in3, LOW);
       digitalWrite(in4, LOW);
       digitalWrite(in1, LOW);
       digitalWrite(in2, LOW);
       bufer = "";
    }
```

Με τον ίδιο τρόπο ελέγχουμε αν η μεταβλητή "bufer" έχει τις λέξεις "start", "pumpon" και "pumpoff". Αν έχει την λέξη start, ξεκινά ο κινητήρας, ανάβει το πράσινο led και ενημερώνονται οι καταστάσεις. Αν έχει την λέξη pumpon θέλουμε να ενεργοποιείτε η αντλία και να ενημερώνετε η κατάστασή της και αν η λέξη είναι pumpoff, θέλουμε να απενεργοποιείται η αντλία.

```
else if (bufer == "start\n") {
       digitalWrite(redLed, LOW);
       digitalWrite(greenLed, HIGH);
       greenledstate = 1;
       redledstate = 0;
       motorState = 1;
```

```
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
        analogWrite(enB, 150);  //Ταχύτητα κινητήρα
        bufer = "";
      }else
       if (bufer == "pumpon\n") {
         digitalWrite(in1, LOW);
         digitalWrite(in2, HIGH);
         analogWrite(enA, 140);  //Ταχύτητα αντλίας
         pumpState = 1;
         bufer = "";
       }else
       if (bufer == "pumpoff\n") {
         digitalWrite(in1, LOW);
         digitalWrite(in2, LOW);
         pumpState = 0;
         bufer = "";
       }
     bufer = "";
  }
```

### 5.2.6 Ανέβασμα μετρήσεων στον ThingSpeak

Για να ανεβάσουμε τις μετρήσεις των αισθητήρων και τις διάφορες καταστάσεις θα πρέπει να ελέγξουμε αν υπάρχει σύνδεση με τον thingspeak μέσω του ethernet. Επίσης θα πρέπει να ελέγχουμε τον χρόνο που έχει περάσει από το τελευταίο ανέβασμα, καθώς ο thingspeak στην δωρεάν άδεια επιτρέπει να ανεβάζουμε κάθε 15 δευτερόλεπτα. Για να γίνει αυτό χρησιμοποιούμε την συνάρτηση millis(), η οποία μας δείχνει τον χρόνο που έχει περάσει από την ενεργοποίηση του arduino σε (ms) και την μεταβλητή lastWriteTime που δηλώσαμε στην αρχή στην οποία θα αποθηκεύετε η τιμή του χρόνου μετά από κάθε ανέβασμα. Η τιμή αυτή συγκρίνετε με την τιμή του χρόνου που επιστρέφει η συνάρτηση millis() και αν είναι μεγαλύτερη από 15 δευτερόλεπτα τότε ο arduino στέλνει τις μετρήσεις.

```
if(!client.connected() &&(millis()-lastWriteTime>15000)) {
   ThingSpeak.setField(1,level);
   ThingSpeak.setField(2,tempC);
   ThingSpeak.setField(3,Total_containers);
   ThingSpeak.setField(4,Red_color);
   ThingSpeak.setField(5,Blue_color);
   ThingSpeak.setField(6,greenledstate);
   ThingSpeak.setField(7,redledstate);
       //Γράφει το channel ID και το Write API Key που δηλώσαμε στην αρχή
   ThingSpeak.writeFields(mySensorsChannelNumber, myWriteAPIKey_sensors);
   lastWriteTime=millis();   //αποθηκεύεται η τιμή του χρόνου μετά το ανέβασμα
   client.stop();
  }
}   //Κλείνει το void loop
```

### 5.2.7 Συναρτήσεις PumpEn() και Move()

Έξω από το void loop γράφουμε τις συναρτήσεις PumpEn() και Move() που καλούνται σε κάποια σημεία του κύριου προγράμματος. Η PumpEn() όταν καλείτε μετράει την στάθμη του υγρού της δεξαμενής και αν βρίσκετε κάτω από 30%, σταματά την διεργασία απενεργοποιώντας τον κινητήρα και την αντλία και ανάβοντας το κόκκινο led. Για να ξεκινήσει πάλι πρέπει να πατήσουμε το button start και η στάθμη του υγρού να έχει ξεπεράσει το 30%.

Αν όμως η στάθμη είναι ίση ή μεγαλύτερη από 30%, αποθηκεύετε η τιμή 1 (για κόκκινο δοχείο) στην θέση του πίνακα που ορίζει ο δείκτης N, ύστερα αυξάνετε ο N και ο αριθμός των κόκκινων δοχείων και ξεκινά η αντλία για 3 δευτερόλεπτα για να γεμίσει το δοχείο. Στη συνέχεια σταματά η αντλία και ξεκινά ο κινητήρας για να συνεχιστεί η διεργασία. Επίσης ενημερώνονται οι καταστάσεις.

```
void PumpEn() {
  motorState = 0;
  //Μέτρηση και υπολογισμός της στάθμης
  float duration, distance, level;
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin,HIGH);
  distance = duration/51;
  distance = distance-3.2;      // offset corection
  distance = 10-distance;
  level = 10*distance;     // distance in % , 0 - 100%
  if (level < 30) {
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
    digitalWrite(in3, LOW);  //Σταματά ο κινητήρας
    digitalWrite(in4, LOW);
    digitalWrite(in1, LOW);  //Σταματά η αντλία
    digitalWrite(in2, LOW);
    greenledstate = 0;
    redledstate = 1;
    pumpState = 0;
    }else
    if (level >= 30) {
    myarray[N] = 1;  //εγγραφή της τιμής 1 στον πίνακα
    ++N;   //Αύξηση του δείκτη
    ++Red_color;  //Αύξηση του αριθμού των κόκκινων δοχείων
    pumpState = 1;
    greenledstate = 1;
    redledstate = 0;
    digitalWrite(in1, LOW);  //Ξεκινά η αντλία
    digitalWrite(in2, HIGH);
    analogWrite(enA, 140);  //Ταχύτητα της αντλίας
    delay(3000);
    digitalWrite(in1, LOW); //Σταματά η αντλία
    digitalWrite(in2, LOW);
    pumpState = 0;
    digitalWrite(in3, LOW);  //Ξεκινά ο κινητήρας
    digitalWrite(in4, HIGH);
    analogWrite(enB, 150);   //Ταχύτητα του κινητήρα
```

```
    motorState = 1;
    delay(500);  //Περιμένει μέχρι να περάσει το δοχείο από τον IR αισθητήρα για να
μην ξαναγίνει trigger
    }
}
```

Η συνάρτηση Move() όταν καλείτε ελέγχει αν η μεταβλητή T, η οποία καθορίζει σε ποια από τις δύο θέσεις θα μετακινήσει το δοχείο ο βραχίονας, έχει την τιμή LOW ή HIGH. Για κάθε μία συνθήκη δίνονται οι κατάλληλες οδηγίες στους σερβοκινητήρες.

```
void Move() {
  if (T == LOW){   //1η θέση
     delay(300);
     servo1.write(67,10,true);    //(μοίρες, ταχύτητα, αναμονή για ολοκλήρωση της
κίνησης)
     servo3.write(12,10,false);
     servo2.write(111,10,true);
     servo3.write(20,10,true);
     delay(300);
     servo4.write(90,10,false);
     servo5.write(135,10,true);
     delay(300);
     servo3.write(34,10,true);
     servo2.write(140,10,false);
     servo3.write(22,6,true);
     servo1.write(180,10,true);
     servo2.write(138,6,false);
     servo3.write(8,6,true);
     servo5.write(108,20,true);
     delay(300);
     servo2.write(170,15,false);
     servo3.write(0,6,true);
     servo3.write(24,10,true);
     servo1.write(73,30,false);
     servo2.write(160,25,false);
     servo3.write(0,8,false);
     delay(300);
  }else
   if (T == HIGH){  //2η θέση
      delay(300);
      servo1.write(67,10,true);
      servo3.write(12,10,false);
      servo2.write(111,10,true);
      servo3.write(20,10,true);
      delay(300);
      servo4.write(90,10,false);
      servo5.write(135,10,true);
      delay(300);
      servo3.write(34,10,true);
      servo2.write(140,10,false);
      servo3.write(20,6,true);
      servo1.write(143,10,true);
      servo2.write(130,6,false);
      servo3.write(10,6,true);
      servo5.write(108,20,true);
      delay(300);
      servo2.write(170,15,false);
      servo3.write(0,6,true);
```

```
        servo3.write(20,10,true);
        servo1.write(73,30,false);
        servo2.write(160,25,false);
        servo3.write(0,8,false);
        delay(300);
    }
}
```

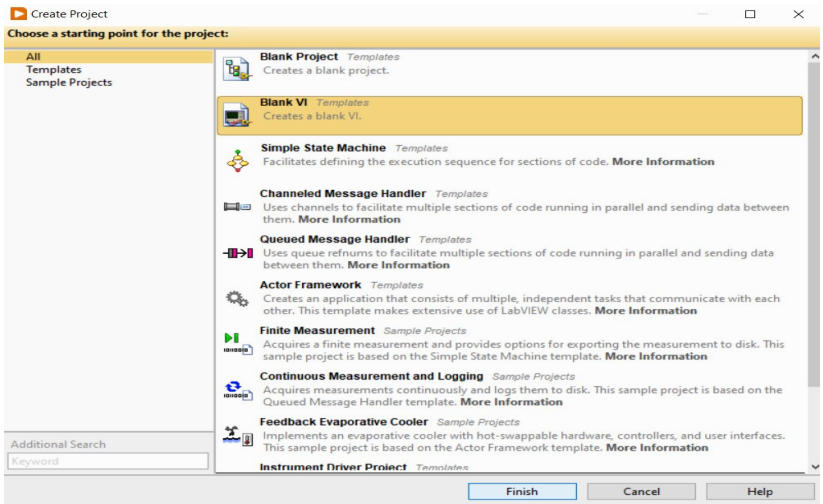
5.2.8 Ανέβασμα κώδικα στην πλακέτα.

Για να ανεβάσουμε τον κώδικα στην πλακέτα arduino, αρχικά πρέπει να έχουμε συνδέσει τον arduino με τον υπολογιστή με ένα κατάλληλο καλώδιο usb. Ύστερα πρέπει να έχουμε κάνει μεταγλώττιση τον κώδικα για να δούμε αν έχουμε σφάλματα. Αφού τα κάνουμε αυτά τότε επιλέγουμε από το μενού ***Εργαλεία,*** και στην καρτέλα που θα ανοίξει, επιλέγουμε την πλακέτα που έχουμε συνδέσει και την θύρα που έχει συνδεθεί. Ο επεξεργαστής επιλέγετε αυτόματα. Στη συνέχεια το μόνο που μένει είναι να πατήσουμε το κουμπί του upload. Αν προκύψει κάποιο σφάλμα θα φανεί κάτω στην περιοχή μηνυμάτων και στην κονσόλα αποσφαλμάτωσης.

5.3 Προγραμματισμός του Labview

Για τον έλεγχο της διεργασίας δημιουργήθηκε ένα περιβάλλον στο Labview με εικονικά όργανα (VIs), με τα οποία μπορούμε να παρακολουθούμε τις τιμές των αισθητήρων, τις καταστάσεις των led και των κινητήρων, πληροφορίες για τον αριθμό των δοχείων και επίσης μπορούμε να χειριζόμαστε τα button για τον κινητήρα και την αντλία.

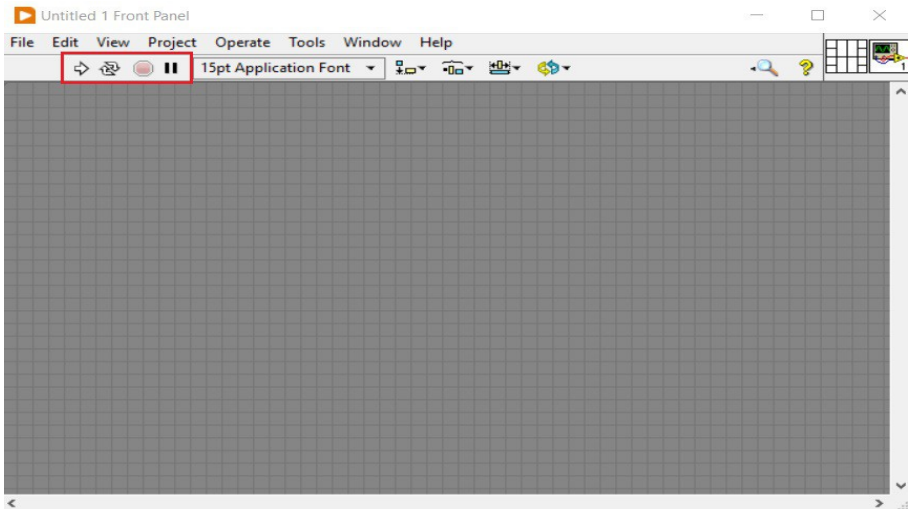
Ξεκινώντας δημιουργούμε ένα νέο project από το αρχικό παράθυρο του Labview πατώντας **Create Project** και στην συνέχεια στην καρτέλα που θα ανοίξει επιλέγουμε **Blank VI.**



Εικόνα 5.9: Παράθυρο δημιουργίας νέου project

5.3.1 Δημιουργία του Front Panel

Στη συνέχεια θα ανοίξουν δύο νέα παράθυρα, το Front Panel και το Block diagram.




Εικόνα 5.10: Front Panel του Labview

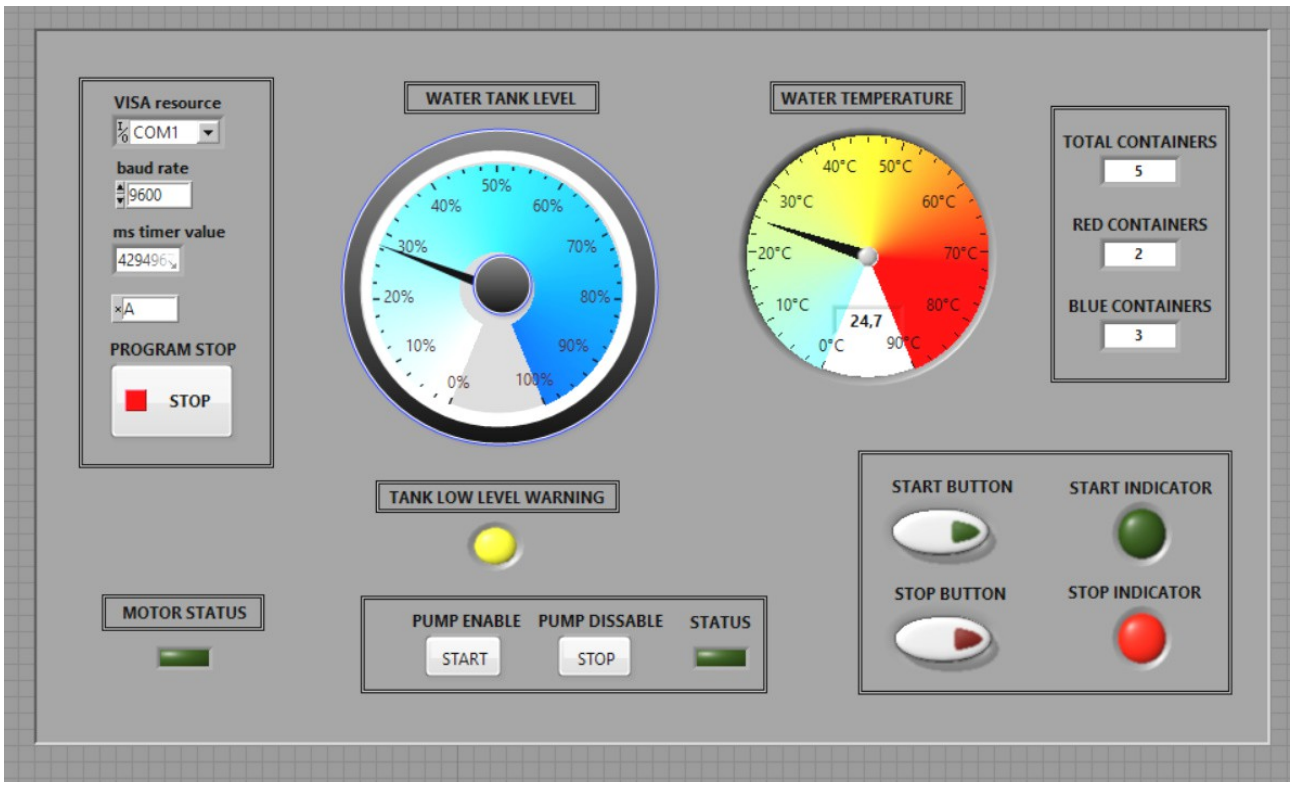
Στο Front Panel δημιουργήθηκε το περιβάλλον ελέγχου με τα όργανα μέτρησης, τα led και άλλα indicators, και τα button.

Τα όργανα αυτά τα εισάγουμε μέσα από βιβλιοθήκες (παλέτες) που διαθέτει το labview τις οποίες μπορούμε να δούμε πατώντας δεξί κλικ μέσα στο front panel. Τα εικονικά όργανα που υπάρχουν χωρίζονται σε δύο κατηγορίες με βάση την ιδιότητά τους. Η πρώτη είναι τα indicators που εξάγουν δεδομένα από το Front Panel και τα χρησιμοποιούμε για να παρατηρούμε καταστάσεις ή τιμές μετρήσεων. Σε αυτήν την κατηγορία ανήκουν Led, όργανα μετρήσεων, δεξαμενές, παλμογράφοι και άλλα γραφήματα, θερμόμετρα, οθόνες χαρακτήρων και άλλα.

Η δεύτερη κατηγορία είναι τα controls που εισάγουν δεδομένα από το Front panel και τα χρησιμοποιούμε για να ελέγχουμε κάποια εφαρμογή μας. Στην κατηγορία αυτή ανήκουν τα ποτενσιόμετρα, οι διακόπτες, οι οθόνες χαρακτήρων, πλήκτρα και άλλα.

Κάθε στοιχείο indicator μπορεί να γίνει control και αντίστροφα, πατώντας δεξί κλιπ πάνω στο στοιχείο και επιλέγουμε change to control ή change to indicator ανάλογα. Επίσης κάθε indicator ή control χειρίζονται δεδομένα συγκεκριμένου τύπου όπως λογικού (boolean), αριθμητικού (numeric) και συμβολοσειράς (string), τα οποία καθορίζονται από το χρώμα του τερματικού που έχουν τα στοιχεία και αντίστοιχα τα καλώδια.

Το περιβάλλον που δημιουργήθηκε για την εργασία αυτή φαίνεται στην παρακάτω εικόνα.




Εικόνα 5.11: Περιβάλλον χειρισμού και ελέγχου

Το όργανο Gauge για την στάθμη του νερού το εισάγουμε από την παλέτα **Silver / Numeric.** Μπορούμε να το μετακινήσουμε όπου θέλουμε και να το μεγαλώσουμε με το ποντίκι όπως τις εικόνες. Πατώντας δεξί κλιπ πάνω του και επιλέγοντας **Properties**, ανοίγει μια καρτέλα όπου μπορούμε να κάνουμε κάποιες ρυθμίσεις για την εμφάνισή του (βελόνες, text label, χρώματα, κ.α), για τον τύπο δεδομένων που θα χειρίζεται που στην περίπτωση μας είναι ο default, για την κλίμακα, για το display format που στην περίπτωσή μας είναι το (%) . Αντίστοιχα κάνουμε τα ίδια και για το δεύτερο Gauge το οποίο όμως το βρίσκουμε από την παλέτα **Modern / Numeric**.

Τα Start και Stop button είναι από την παλέτα **Modern / Boolean** και τα start και stop για την αντλία τα βρίσκουμε από την **Silver / Boolean** και είναι τα blank button. Τα button αυτά από default λειτουργούν σαν διακόπτες και όχι σαν button. Για να το αλλάξουμε αυτό πατάμε δεξί κλικ στον κάθε διακόπτη, επιλέγουμε **Mechanical Action / Switch Until Released.** Με τον τρόπο αυτό δεν παραμένουν ενεργοποιημένοι όταν τους πατήσουμε μια φορά.

Τα led τα βρίσκουμε και αυτά από την παλέτα **Modern / Boolean**.

Πάνω αριστερά υπάρχουν κάποια controls και indicators για την σειριακή επικοινωνία μέσω Visa και το button Stop για την διακοπή του προγράμματος και πάνω δεξιά έχουμε τρία String Indicators για τον αριθμό των δοχείων. Αυτά μπορούμε να τα βρούμε από την παλέτα **Modern / String & Path**, ή να τα εισάγουμε από το block diagram.

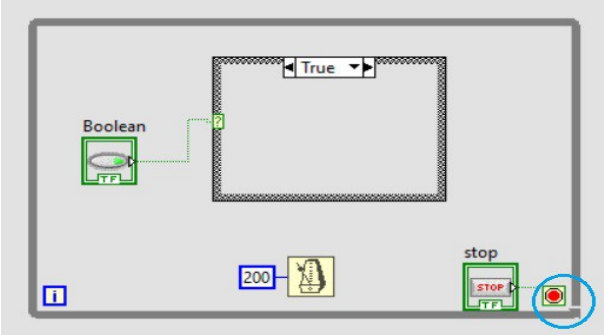
Τέλος το background και τα περιγράμματα είναι από την παλέτα decorations.

### 5.3.2 Προγραμματισμός στο Block Diagram

Στο Block Diagam του Labview έγινε το κύκλωμα της εφαρμογής αυτής με τα κατάλληλα block στοιχεία και συναρτήσεις. Ότι στοιχείο εισάγουμε στο front panel, εμφανίζετε αυτόματα και εδώ με την μορφή block στοιχείων τα οποία έχουν και τερματικά για σύνδεση. Πατώντας πάνω στα τερματικά μπορούμε να δημιουργήσουμε καλώδιο για να το συνδέσουμε με κάποιο άλλο στοιχείο. Το χρώμα στο τερματικό και στα καλώδια συμβολίζει τον τύπο δεδομένων. Στοιχεία και ακροδέκτες διαφορετικών τύπου δεδομένων δεν μπορούν να συνδεθούν μεταξύ τους εκτός αν υπάρχει κάποιο άλλο στοιχείο ανάμεσα που κάνει μετατροπή.

Οι συναρτήσεις και τα στοιχεία που θα χρησιμοποιηθούν για το πρόγραμμα είναι:

- Η While loop: Την βρίσκουμε από την παλέτα **Programming / Structures**. Κατά την εκτέλεση του προγράμματος, ότι υπάρχει μέσα σε αυτή τρέχει συνέχεια και σταματά μόνο αν υπάρχει μέσα κάποια συνθήκη ή κάποιον διακόπτη που θα βάλουμε. Επίσης μπορούμε να βάλουμε timer για να υπάρχει κάποια καθυστέρηση των επαναλήψεων που θα ορίσουμε.

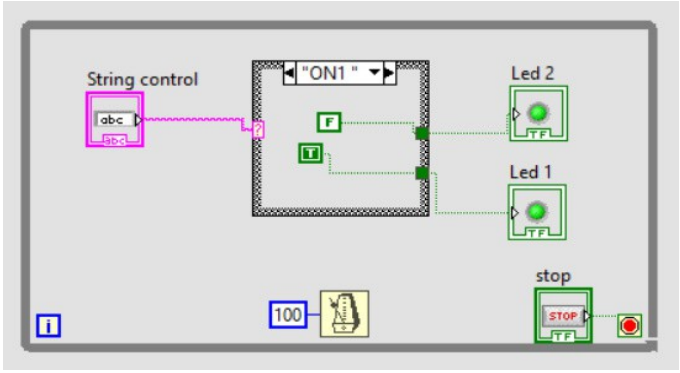


Εικόνα 5.12: While loop με case structure μέσα

Κάτω δεξιά μέσα στον μπλε κύκλο, είναι το loop condition. Εκεί συνδέουμε ένα button όπως φαίνεται και στη εικόνα για να μπορούμε να σταματάμε το πρόγραμμα. Για να το κάνουμε αυτό πατάμε δεξί κλικ στην αριστερή πλευρά του loop condition και επιλέγουμε create control. Αυτόματα θα εμφανιστεί ένα button στο front panel.

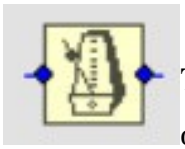
- Case Structure: Την βρίσκουμε και αυτή από την ίδια παλέτα με την while. Μπορεί να έχει πολλά υποδιαγράμματα που λέγονται cases (περιπτώσεις). Μια τιμή επιλογής ορίζει ποιο case θα εκτελεστεί. Στην παραπάνω εικόνα (εικόνα 5.4) για παράδειγμα, το case που θα εκτελεστεί ορίζετε από την τιμή του διακόπτη (Boolean) που είναι συνδεδεμένος στον

επιλογέα. Επειδή ο διακόπτης έχει μόνο δύο καταστάσεις (True or false), θα υπάρχουν δυο περιπτώσεις και στην case (true και false) όπως φαίνεται στο πάνω μέρος της. Επίσης η τιμή που καθορίζει πιο case θα εκτελεστεί μπορεί εκτός από boolean να είναι numeric, string και άλλος τύπος όπως στην παρακάτω εικόνα.



Εικόνα 5.13: Case structure με επιλογέα τύπου string

Στην εικόνα 23, επειδή ο επιλογέας δέχεται string τιμές δηλαδή χαρακτήρες ή λέξεις, έτσι και τα cases μπορεί να είναι όσα θέλουμε.

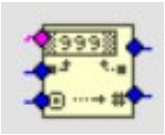


**Wait Until Next ms Multiple:** Το βρίσκουμε από την παλέτα **Programming / Timing**. Το χρησιμοποιούμε μέσα στην while loop για delay. Ορίζουμε στον αριστερό ακροδέκτη τον χρόνο σε milliseconds που θα περιμένει για να ξαναεκτελεστεί η whle.

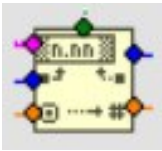
**Visa Write:** Το βρίσκουμε από την παλέτα **Instrument I/O /VISA**. Γράφει δεδομένα τύπου string στη συσκευή είναι συνδεδεμένη στο VISA.

**Visa Read:** Το βρίσκουμε από την παλέτα **Instrument I/O /VISA**. Διαβάζει τα byte δεδομένων που στέλνει η συσκευή η οποία είναι συνδεδεμένη στο Visa. Τον αριθμό των bytes τον ορίζουμε εμείς. Τα δεδομένα είναι τύπου string.

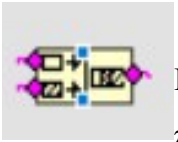
**Match True / False String:** Το βρίσκουμε από την παλέτα **Programming / String / Additional String Functions**. Ελέγχει αν η λέξη ταιριάζει με την αληθινή ή την ψευδή λέξη και επιστρέφει μια τιμή Boolean (True ή False) στην επιλογή ανάλογα με το αν λέξη ταιριάζει με την αληθινή ή την ψευδή.



**Decimal String To Number:** Το βρίσκουμε από την παλέτα **Programming / String / Number- String Conversion**. Μετατρέπει τους αριθμητικούς χαρακτήρες από τύπο string, σε δεκαδικό ακέραιο αριθμό (numeric). Έτσι μπορούμε να βλέπουμε τις τιμές των μετρήσεων μας σε κάποιο όργανο.



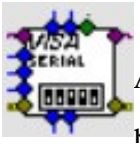
**Fract/Exp String To Number:** Το βρίσκουμε από την παλέτα **Programming / String / Number- String Conversion**. Ερμηνεύει τους χαρακτήρες 0 έως 9, συν, μείον, e, E και το δεκαδικό σημείο από μια συμβολοσειρά (string) και το επιστρέφει σε αριθμό κινητής υποδιαστολής (numeric floating point).



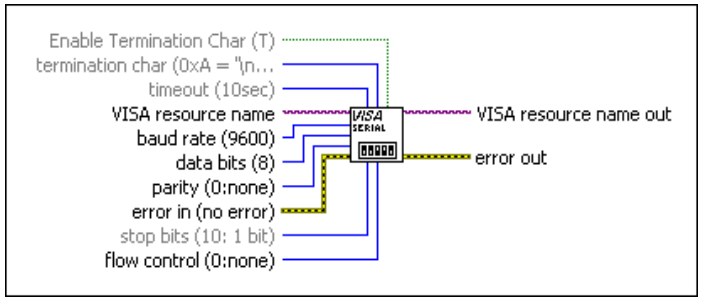
**Concatenate Strings:** Το βρίσκουμε από την παλέτα **Programming / String**. Ενώνει δύο ή περισσότερες σειμβολοσειρές (strings) σε μία ενιαία. Για να έχουμε περισσότερες εισόδους αρκεί να πατήσουμε δεξί κλιπ πάνω σε μία είσοδο και να επιλέξουμε Add Input, ή τραβώντας με το ποντίκι το κάτω μέρος του στοιχείου για να αλλάξει το μέγεθος του.

### 5.3.3 Σύνδεση LabView με Arduino

Για την σύνδεση του Labview με τον arduino, θα πρέπει να εισάγουμε δύο νέα στοιχεία για να στήσουμε έναν δίαυλο επικοινωνίας. Τα στοιχεία αυτά είναι:



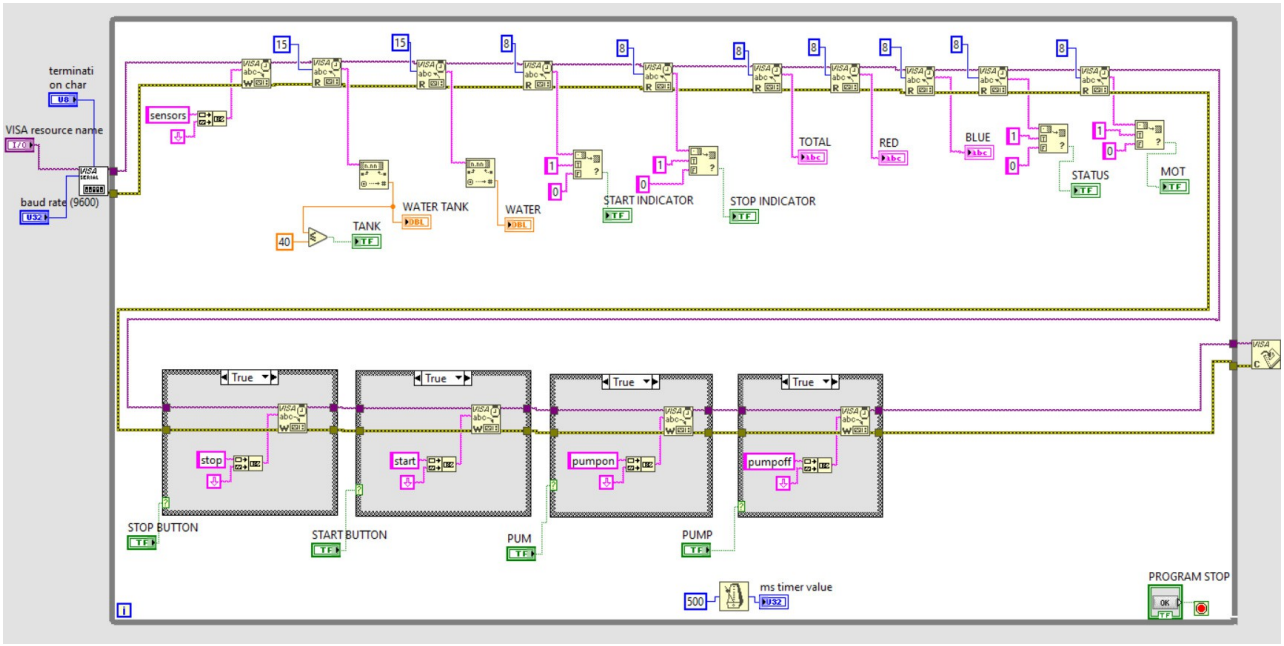
**VISA Configure Serial Port:** Το βρίσκουμε από την παλέτα **Instrument I/O / Serial**. Αρχικοποιούμε την σειριακή θύρα που έχουμε συνδέσει το arduino, η οποία καθορίζετε από το όνομα VISA resource name. Επίσης ορίζουμε το baud rate ανάλογα τον ρυθμό που υποστηρίζει το κάθε μοντέλο arduino (το πιο σύνηθες είναι 9600). Το βάζουμε στην αρχή του προγράμματος έξω από την while loop.




Εικόνα 5.14: VISA Configure Serial Port [Πηγή]

**VISA Close:** Το βρίσκουμε από την παλέτα **Instrument I/O / Visa / Advanced**. Κλείνει την σύνδεση με την συσκευή. Την βάζουμε στο τέλος του προγράμματος έξω από την while loop.

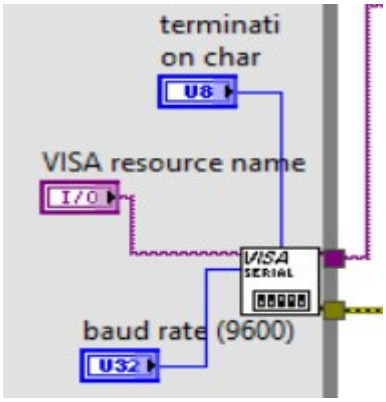
Το τελικό κύκλωμα του Block διαγράμματος φαίνεται στην παρακάτω εικόνα.

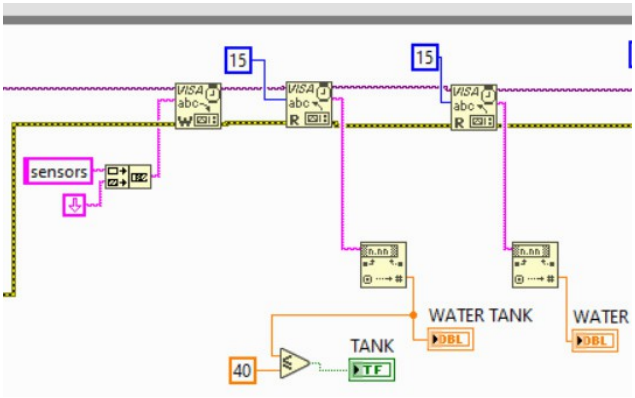
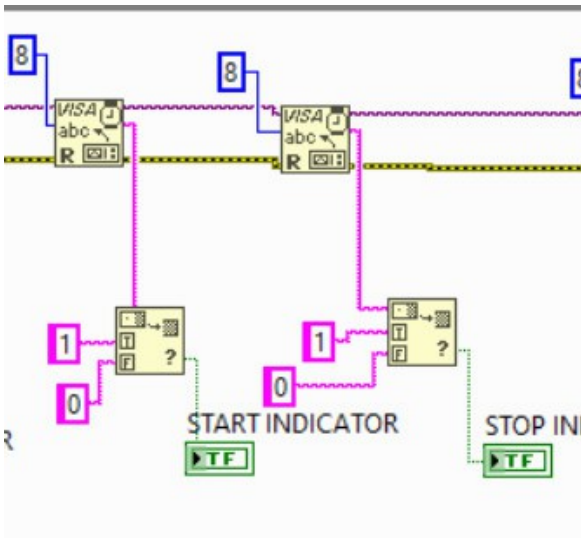


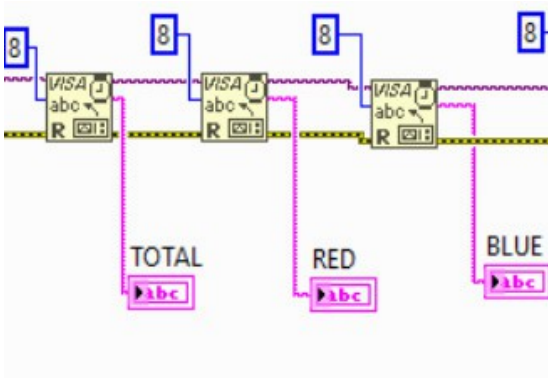

Εικόνα 5.15: Block Diagram του Labview

Όπως βλέπουμε στην εικόνα, όλο το πρόγραμμα τρέχει μέσα σε μια while loop. Το πάνω κομμάτι αφορά τις μετρήσεις των αισθητήρων και το κάτω κομμάτι αφορά τον χειρισμό. Έξω από την while loop υπάρχουν τα block στοιχεία για την σειριακή επικοινωνία μέσω Visa.

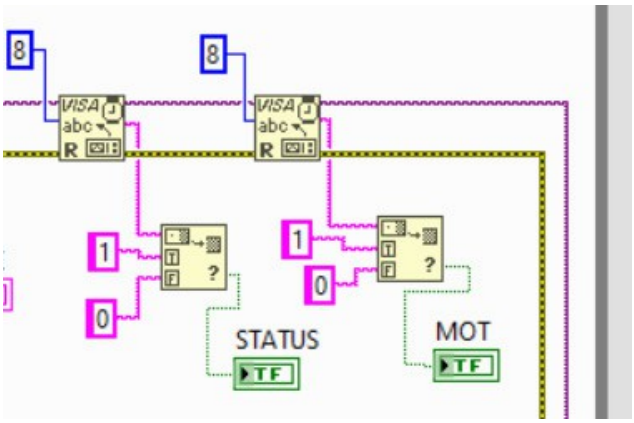
Αναλυτικά το κάθε κομμάτι του προγράμματος φαίνεται παρακάτω.

| | |
|---|---|
|  | Έξω από την while loop υπάρχει το VISA Configure Serial Port για την σειριακή επικοινωνία με τον arduino. Σε αυτό συνδέουμε τα απαραίτητα controls για τα δεδομένα που χρειαζόμαστε για την σύνδεση. Το VISA resource name είναι για να επιλέγουμε την θύρα που έχει συνδεθεί ο arduino. Το baud rate είναι η ταχύτητα σε bits ανά δευτερόλεπτο και είναι default στα 9600, αλλά μπορούμε να το |

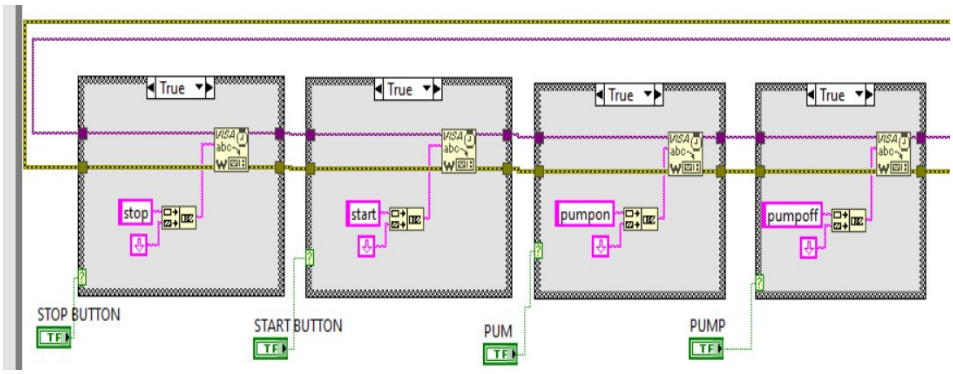
| | |
|---|---|
| | αλλάξουμε αν ο arduino υποστηρίζει και μεγαλύτερη ταχύτητα. Το termination char δεν χρειάζεται να μπει. |
|  | Μέσα στην while loop στο πάνω μέρος, το πρώτο στοιχείο που υπάρχει είναι το Visa write το οποίο στέλνει την λέξη “sensors” και έναν χαρακτήρα ASCI για αλλαγή γραμμής (Line Feed) στον arduino και περιμένει απάντηση. Στη συνέχεια έχουμε τα visa read τα οποία διαβάζουν από την σειριακή θύρα όσα byte δεδομένων έχουμε ορίσει. Η πρώτη read θα διαβάσει την απάντηση που θα στείλει ο arduino σε string για την στάθμη του νερού της δεξαμενής, ενώ η δεύτερη θα διαβάσει την θερμοκρασία του νερού. Η έξοδος των read πηγαίνει στα στοιχεία Fract/Exp String to Number τα οποία μετατρέπουν τα string δεδομένα σε αριθμό κινητής υποδιαστολής για να μπορούμε να τα απεικονίσουμε σε κάποιο όργανο όπως γίνετε και στην εικόνα. Για την στάθμη του νερού υπάρχει και ένας συγκριτής ο οποίος ενεργοποιεί ένα warning led αν η στάθμη είναι μικρότερη ή ίση του 40%. |
|  | Οι επόμενες δύο read είναι για τις καταστάσεις των start και stop led. Η έξοδος των read πηγαίνει στα στοιχεία Match True / False String όπου ελέγχονται τα strings που διαβάστηκαν από την σειριακή θύρα με τα string που έχουμε ορίσει εμείς (0 ή 1). Αν είναι 1, δηλαδή true, τότε ανάβουν τα led αλλιώς αν είναι 0, δηλαδή false, τότε σβήνουν τα led. |



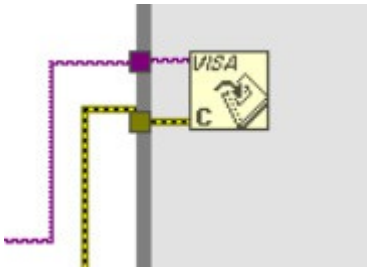
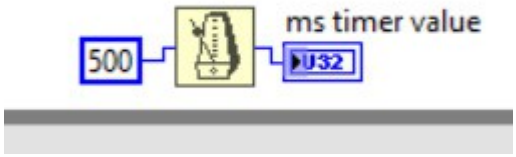

Στη συνέχεια έχουμε άλλες τρεις read οι οποίες διαβάζουν τον αριθμό των συνολικών, των κόκκινων και των μπλε δοχείων και τον απεικονίζουν σε τρεις string indicators. Για να δημιουργήσουμε έναν indicator πατάμε δεξί κλικ πάνω στην έξοδο της read και επιλέγουμε create indicator.




Μετά έχουμε τις δυο τελευταίες read που διαβάζουν τις καταστάσεις του κινητήρα και της αντλίας και ανάβουν ή σβήνουν τα αντίστοιχα led.




Στη συνέχεια στο κάτω μέρος της while υπάρχουν τέσσερα case structures για τον έλεγχο του κινητήρα και της αντλίας. Κάθε case έχει 2 σενάρια, το true και το false, τα οποία ενεργοποιούνται από τα button που είναι συνδεδεμένα σε κάθε case. Τα cases από default είναι false αφού και τα button δεν έχουν πατηθεί. Αν κάποιο button πατηθεί, τότε το αντίστοιχο case θα γίνει true και θα εκτελεστεί ότι υπάρχει μέσα σε αυτό. Σε όλα τα true cases υπάρχει ένα visa write που στέλνει στον arduino μία λέξη. Έτσι λοιπόν στο πρώτο στέλνει την λέξη “stop”, ώστε ο arduino να σταματήσει τον κινητήρα και όλη την διεργασία. Στο δεύτερο στέλνει την λέξη “start”, για να ξεκινήσει η διεργασία και ο κινητήρας. Στο τρίτο στέλνει την λέξη “pumpon”, για να ξεκινήσει η αντλία και στο τέταρτο την λέξη “pumpoff”, για να σταματήσει η αντλία. Τα fasle cases δεν έχουν τίποτα

| | |
|---|---|
| μέσα, αλλά υπάρχει μόνο η σύνδεση των καλωδίων. | |
|  | Τέλος έξω από την while loop υπάρχει το VISA close με το οποίο κλείνουμε την σύνδεση με τον arduino. |
|  | Κάποια επιπλέον αναγκαία στοιχεία μέσα στην while loop είναι ένα timer στο οποίο ορίζουμε ένα χρόνο καθυστέρησης σε ms, στην συγκεκριμένη περίπτωση 500ms. Έτσι το πρόγραμμα μέσα στην while loop θα επαναλαμβάνετε ανά 500ms για να μην υπάρχουν προβλήματα λόγω του διαφορετικού χρονισμού των ρολογιών. |
|  | Επίσης για να μπορούμε να σταματήσουμε το πρόγραμμα είναι αναγκαίο να δημιουργήσουμε ένα button στο loop condition κάτω δεξιά της while loop. |

Πίνακας 5.2: Ανάλυση των κομματιών του προγράμματος

Τέλος δημιουργήθηκε μια executable εφαρμογή του προγράμματος ώστε να τρέχει σε οποιονδήποτε υπολογιστή και χωρίς να χρειάζεται το labview. Για να γίνει αυτό, πατάμε από την μπάρα μενού του labview, **Tools** και επιλέγουμε **Build Application (EXE) from VI**.. Στο παράθυρο που θα μας ανοίξει , μπορούμε μέσα από τις διάφορες κατηγορίες να κάνουμε ρυθμίσεις. Οι μόνες ρυθμίσεις που έγιναν είναι να δώσουμε όνομα, η επιλογή του αποθηκευτικού χώρου και το Enable debugging από την κατηγορία Advanced το οποίο εξασφαλίζει πλήρη βελτιστοποίηση.

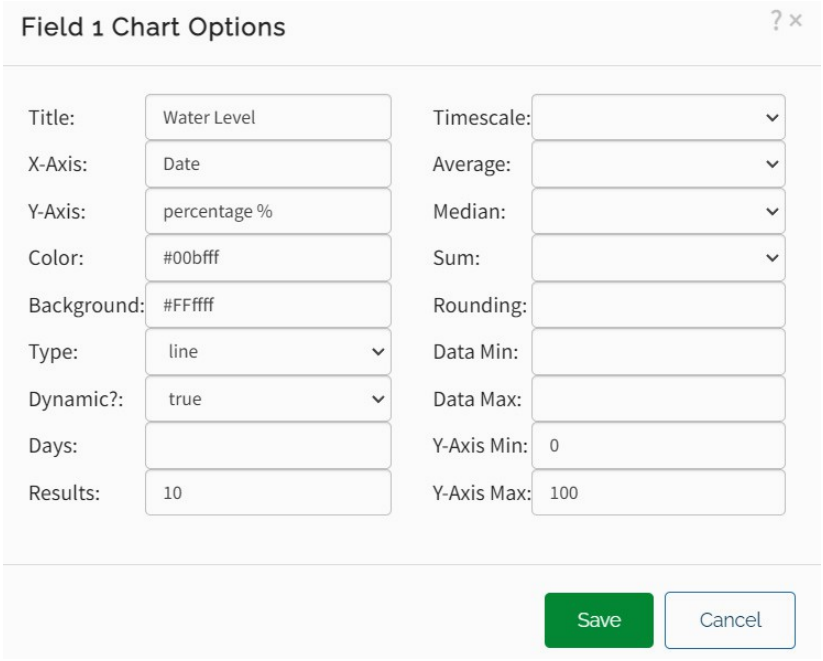
# Κεφάλαιο 6ο Διαχείριση και επεξεργασία γραφημάτων στο thingspeak

## 6.1 Εμφάνιση και παραμετροποίηση των γραφημάτων

Αφού έχουμε τελειώσει με τον προγραμματισμό του arduino, πλέον στο κανάλι του thingspeak που έχουμε δημιουργήσει στην ενότητα 5.1 του 5ου κεφαλαίου, ανεβαίνουν τα δεδομένα που στέλνει ο arduino κάθε 15 δευτερόλεπτα, στα αντίστοιχα πεδία που έχουμε δημιουργήσει. Τα δεδομένα αυτά είναι η στάθμη του νερού της δεξαμενής, η θερμοκρασία του νερού, ο αριθμός των συνολικών δοχείων, ο αριθμός των κόκκινων και μπλε δοχείων και οι καταστάσεις των led. Τα γραφήματα στα πεδία εμφανίζονται αυτόματα όταν ληφθούν τα δεδομένα και από default έχουν κόκκινο ή μπλε χρώμα και ο τύπος τους είναι γραμμικός (line). Μπορούμε όμως να τα μορφοποιήσουμε ώστε να έχουν διαφορετικό χρώμα, όνομα στους άξονες, μορφή, ακόμα και όρια στην κλίμακα. Για να το κάνουμε αυτό πατάμε στο γράφημα που θέλουμε, το εικονίδιο που φαίνεται μέσα στον κόκκινο κύκλο της εικόνας.


Εικόνα 6.1: Εικονίδιο για την επιλογή chart options

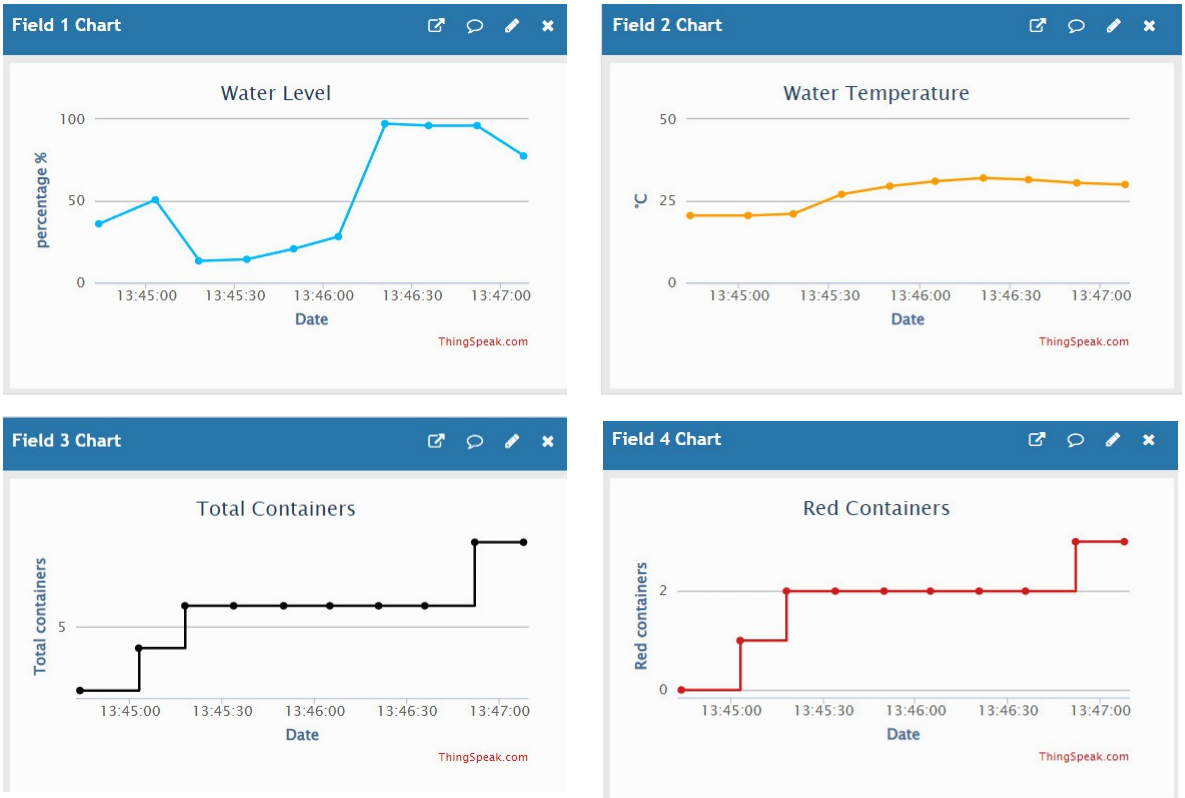
Θα ανοίξει μια καρτέλα (chart options) στην οποία μπορούμε να δώσουμε πληροφορίες για το γράφημα.



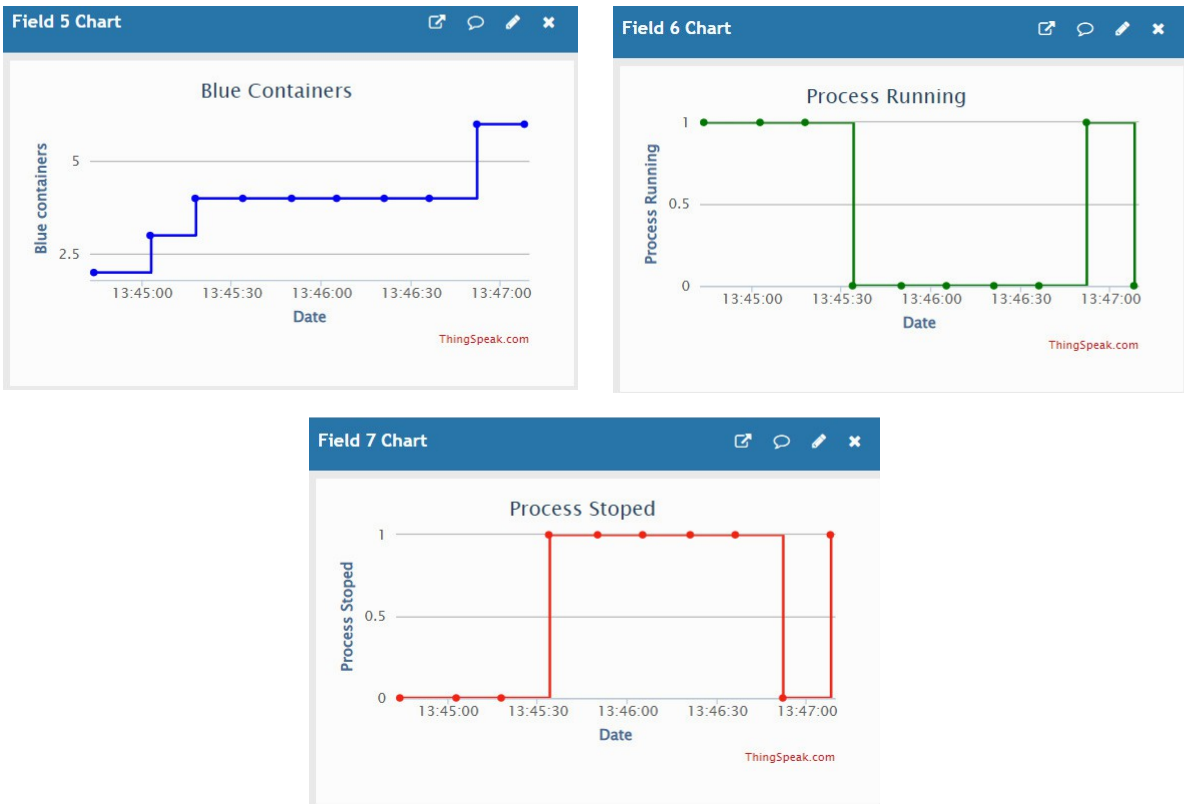

Εικόνα 6.2: Καρτέλα Chart Options

Η συγκεκριμένη καρτέλα της εικόνας είναι για το πεδίο 1 που έχει τα δεδομένα της στάθμης του νερού. Μπορούμε να δώσουμε τίτλο στο γράφημα, όνομα τους άξονες, χρώμα στο γράφημα και στο background με κωδικούς html. Στη συνέχεια διαλέγουμε τον τύπο του γραφήματος με επιλογές line (γραμμή), step (βηματική), column (στήλη), spline και bar (μπάρα). Στα γραφήματα για την στάθμη του νερού και της θερμοκρασίας (πεδία 1 και 2), ο τύπος είναι line ενώ στα υπόλοιπα είναι step. Μετά η επιλογή dynamic αν είναι true, θα γίνετε αυτόματη ενημέρωση του γραφήματος με βάση το χρονικό διάστημα που αναφέρεται στην ενημέρωση, στην περίπτωσή μας κάθε 15 δευτερόλεπτα. Στο results βάζουμε τον αριθμό των δεδομένων που θα φαίνονται στο γράφημα. Στις επιλογές timescale, average, median και sum, μπορούμε επιλέξουμε κάποιον από τους έτοιμους αριθμούς σε λεπτά που έχει, ώστε να παίρνουμε αντίστοιχα στο γράφημα τον χρόνο, τον μέσο όρο, την διάμεσο και το άθροισμα των τιμών στον χρόνο που έχουμε επιλέξει. Για παράδειγμα αν επιλέξω στο average χρόνο 10 λεπτά, τότε στο γράφημα θα έχω κάθε 10 λεπτά τη μέση τιμή από τις τιμές που θα ανεβαίνουν μέσα στο διάστημα των 10 λεπτών. Το rounding είναι για να επιλέξουμε πόσα δεκαδικά ψηφία θα έχουν οι τιμές του γραφήματος. Τέλος τα τελευταία τέσσερα πεδία είναι για την επιλογή του μέγιστου και ελάχιστου αριθμού των δεδομένων και των αξόνων.

Τα μορφοποιημένα γραφήματα του καναλιού “arduino sensors” φαίνονται στις παρακάτω εικόνες.



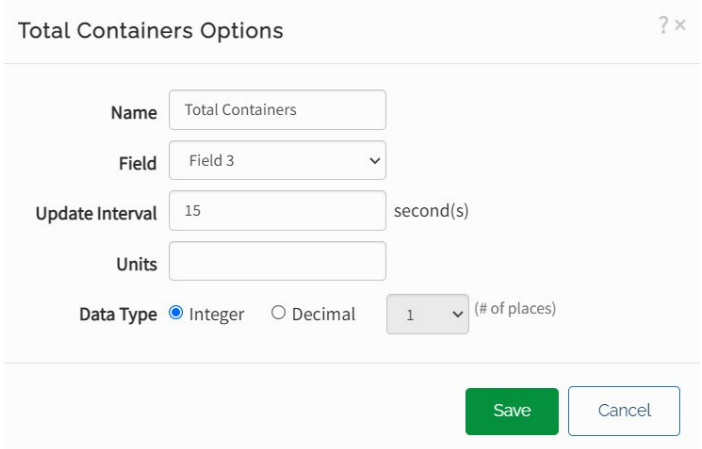

Εικόνα 6.3: Γραφήματα του καναλιού στον thingspeak 1




Εικόνα 6.4: Γραφήματα του καναλιού στον thingspeak 2

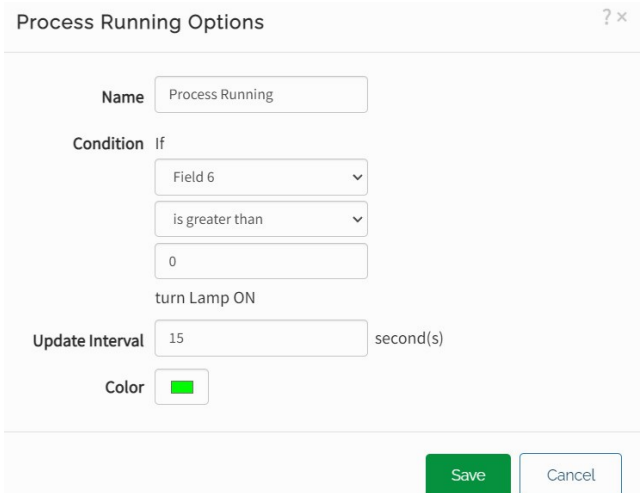
Επίσης στα γραφήματα με τον αριθμό των δοχείων και με τις καταστάσεις των led, δημιουργήθηκαν κάποια widgets (γραφικά στοιχεία) για την καλύτερη απεικόνιση τους. Για να προσθέσουμε widgets πατάμε στο Add Widget που υπάρχει μέσα στο κανάλι όπως αναφέρεται στην ενότητα 5.1 του 5ου κεφαλαίου. Για τον αριθμό των δοχείων προστέθηκαν τρία numeric display και για τις καταστάσεις των led, 2 lamp indicator.

Όταν εισάγουμε κάποιο widget ανοίγει μια καρτέλα με ρυθμίσεις διαφορετικές για τον κάθε τύπο. Για τα numeric display η καρτέλα φαίνεται στην παρακάτω εικόνα.



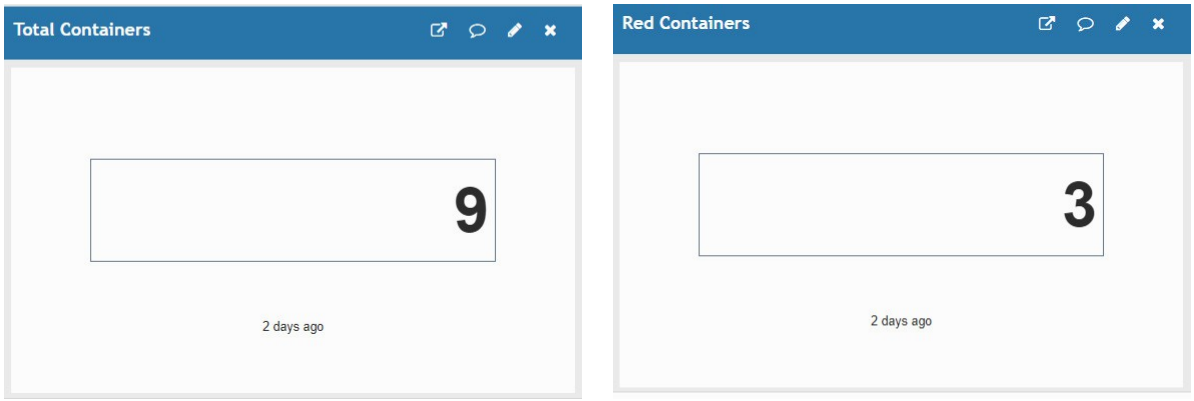

Εικόνα 6.5: Καρτέλα ρυθμίσεων των numeric display widget

Εκεί δίνουμε όνομα στο γραφικό στοιχείο, επιλέγουμε το πεδίο με το γράφημα για το οποίο θέλουμε να δημιουργηθεί το γραφικό στοιχείο, τον χρόνο ανανέωσης σε δευτερόλεπτα, την μονάδα μέτρησης και τον τύπο των δεδομένων. Για τα lamp indicators η καρτέλα φαίνεται στην παρακάτω εικόνα.

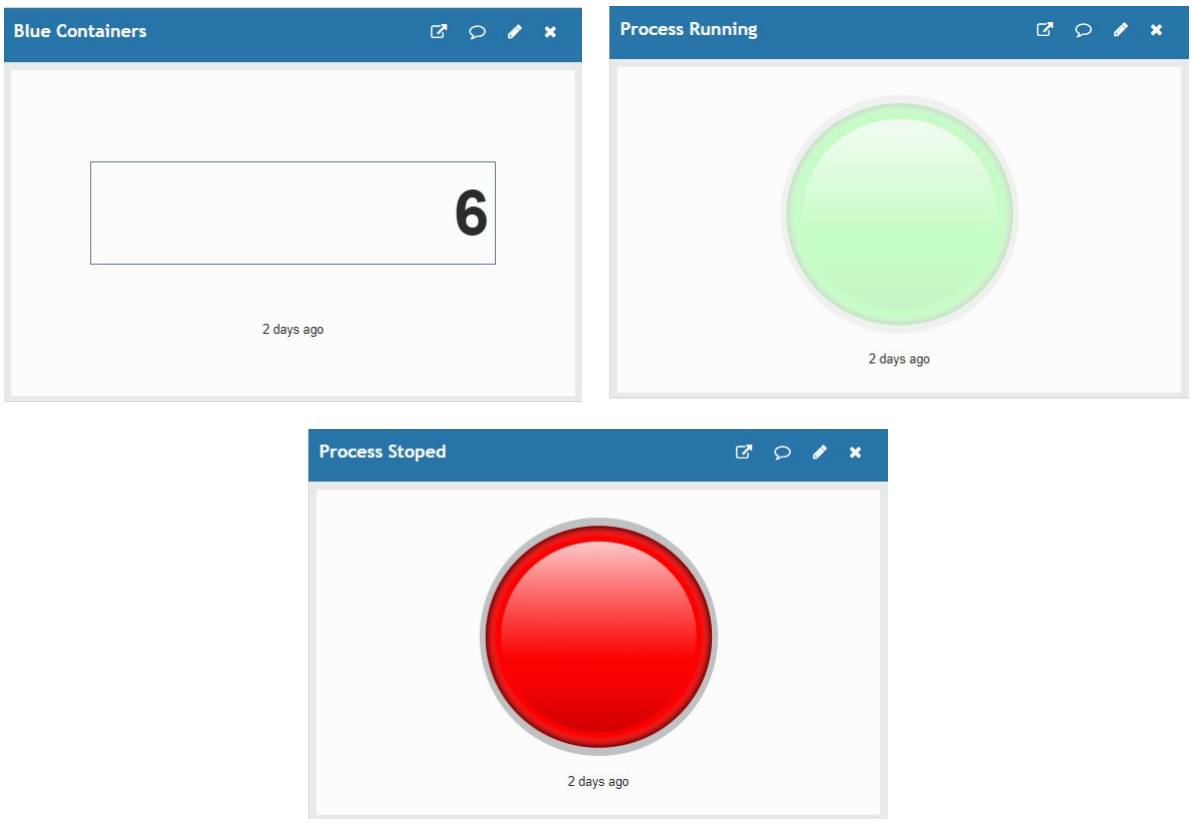



Εικόνα 6.6: Καρτέλα ρυθμίσεων των lamp indicator

Στην καρτέλα αυτή δίνουμε όνομα στο γραφικό στοιχείο και στη συνέχεια υπάρχει μια συνθήκη if στην οποία καθορίζουμε πότε θα ανάβει η εικονική λάμπα. Για την συνθήκη επιλέγουμε το πεδίο με το γράφημα για το οποίο θέλουμε να δημιουργηθεί το widget και του οποίου οι τιμές θα συγκρίνονται στην συνθήκη και στη συνέχεια την αριθμητική σύγκριση για την συνθήκη. Στη εικόνα 6.6 έχουμε επιλέξει “is greater than” και στη συνέχεια την τιμή που θέλουμε να συγκρίνει, στην περίπτωσή μας το 0. Έτσι αν η τιμή από το γράφημα του πεδίου 6 είναι μεγαλύτερη από το 0, τότε θα ανάβει η λάμπα. Το γράφημα του πεδίου 6 παίρνει τιμές 0 ή 1. Τέλος επιλέγουμε τον χρόνο ανανέωσης και το χρώμα της λάμπας. Τα widgets που δημιουργήθηκαν φαίνονται στην παρακάτω εικόνα.




Εικόνα 6.7: Widgets για τον αριθμό των δοχείων και τις καταστάσεις των led 1



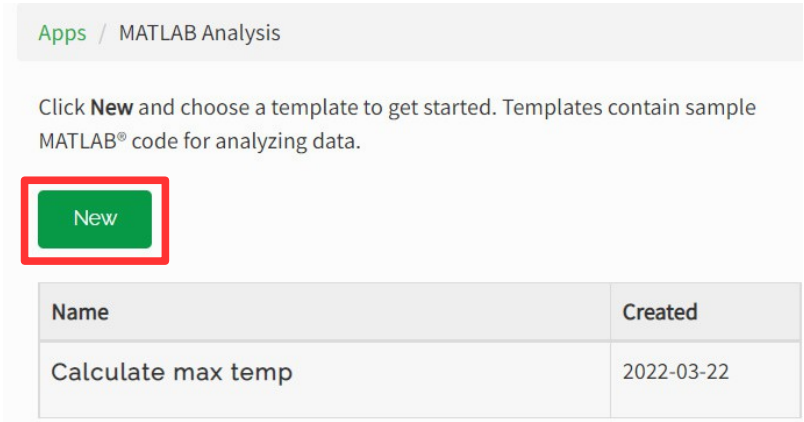

Εικόνα 6.8: Widgets για τον αριθμό των δοχείων και τις καταστάσεις των led 2

## 6.2 Εύρεση Μέγιστης – Ελάχιστης θερμοκρασίας με matlab analysis

Η εύρεση της μέγιστης και της ελάχιστης θερμοκρασίας του νερού της δεξαμενής έγινε με matlab analysis, μια δυνατότητα του thingspeak με την οποία χρησιμοποιώντας κώδικα matlab μπορούμε να αναλύουμε τα δεδομένα όποιου καναλιού και πεδίου ορίσουμε εμείς. Για την συγκεκριμένη ανάλυση υπάρχει ένα έτοιμο παράδειγμα με γραμμένο τον κώδικα τον οποίο τροποποιούμε για τα δικά μας δεδομένα. Το παράδειγμα αυτό υπολογίζει την μέγιστη και ελάχιστη τιμή της θερμοκρασίας και αποθηκεύει τα αποτελέσματα σε διαφορετικά κανάλια. Για τον λόγο αυτό πρέπει να δημιουργήσουμε πρώτα δύο νέα κανάλια με ένα πεδίο το καθένα. Στο ένα κανάλι θα αποθηκεύετε η μέγιστη τιμή της θερμοκρασίας και στο δεύτερο κανάλι η ελάχιστη τιμή. Ο τρόπος για την δημιουργία καναλιού έχει περιγραφεί στην ενότητα 5.1 του 5ου κεφαλαίου. Το κάθε κανάλι έχει ένα μοναδικό channel ID, Read API Key και Write API Key τα οποία τα χρησιμοποιούμε στον κώδικα της ανάλυσης μαζί με τον αριθμό του πεδίου (field) για να διαβάσουμε ή να αποθηκεύσουμε δεδομένα.

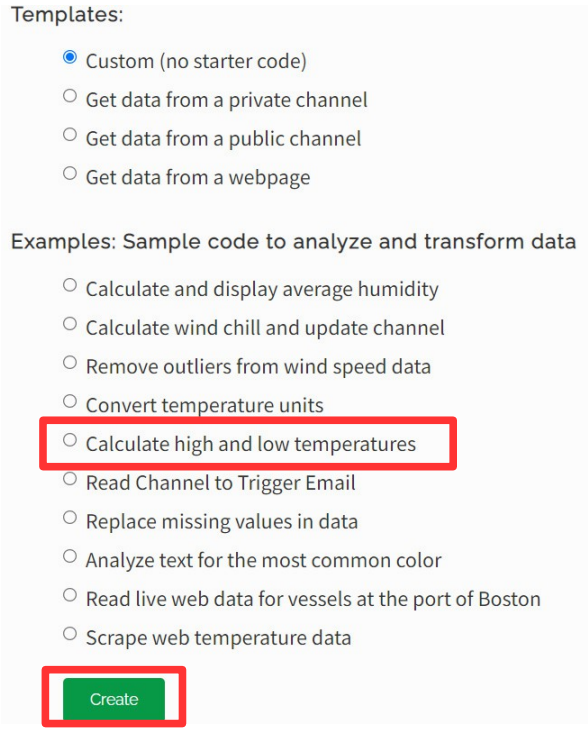
Στη συνέχεια για να δημιουργήσουμε μια ανάλυση επιλέγουμε στην μπάρα μενού του thingspeak, Aps και επιλέγουμε Matlab Analysis ή μπορούμε να το βρούμε μέσα από το κανάλι στην πάνω

δεξιά μεριά με πράσινο χρώμα. Στην καρτέλα που θα ανοίξει βλέπουμε τις υπάρχουσες αναλύσεις τις οποίες μπορούμε να επεξεργαστούμε ή πατάμε new για να δημιουργήσουμε μια νέα.




Εικόνα 6.9: Καρτέλα Matlab Analysis

Πατώντας new ανοίγει μια νέα καρτέλα με templates και έτοιμα παραδείγματα. Επιλέγουμε από τα παραδείγματα το Calculate high and low temperatures και πατάμε create. Στην καρτέλα που θα μας ανοίξει βρίσκεται ο κώδικας matlab, τον οποίο θα πρέπει να τροποποιήσουμε για να διαβάζει τα δεδομένα από το δικό μας κανάλι και πεδία και να αποθηκεύει τα αποτελέσματα στα νέα κανάλια που δημιουργήσαμε. Στην αρχή της καρτέλας, πριν τον κώδικα μπορούμε να αλλάξουμε το όνομα του παραδείγματος της ανάλυσης.



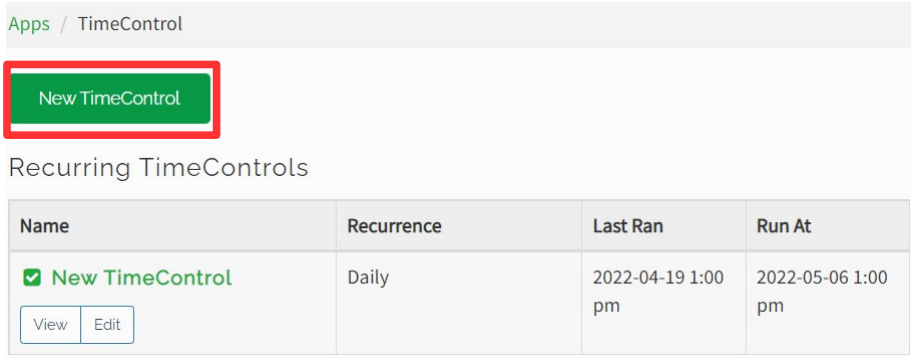

Εικόνα 6.10: Καρτέλα επιλογής παραδειγμάτων για matlab analysis

Ο κώδικας με τις αλλαγές για τον υπολογισμό της μέγιστης και ελάχιστης θερμοκρασίας είναι ο παρακάτω.

```
% Channel ID από το οποίο θα διαβάσει τα δεδομένα
readChannelID = 1676414;
% Field ID που έχει τα δεδομένα της θερμοκρασίας
TemperatureFieldID = 2;

% Channel Read API Key
readAPIKey = 'CZAI10GTNSHUG7R3';

% Διαβάζει τα δεδομένα της θερμοκρασίας μιας ημέρας χρησιμοποιώντας τις παραπάνω
πληροφορίες
[tempF,timeStamp] = thingSpeakRead(readChannelID,'Fields',TemperatureFieldID,..
'numDays',1,'ReadKey',readAPIKey);

% Υπολογισμός της μέγιστης και ελάχιστης θερμοκρασίας
[maxTempF,maxTempIndex] = max(tempF);
[minTempF,minTempIndex] = min(tempF);

% Επιλέγει τη χρονική στιγμή που μετρήθηκε η μέγιστη και ελάχιστη θερμοκρασία
timeMaxTemp = timeStamp(maxTempIndex);
timeMinTemp = timeStamp(minTempIndex);

% Γράφουμε τα channel ID των καναλιών της μέγιστης και ελάχιστης θερμοκρασίας
writeChannelIDmax = 1682414;
writeChannelIDmin = 1682416;
% Γράφουμε τα Write API Keys των καναλιών
writeAPIKeymax = '5KXS6NYBJKA1TL0C';
writeAPIKeymin = 'S89SG4UU20TXZG5D';
% Γράφει τα δεδομένα στα κανάλια που γράψαμε παραπάνω
thingSpeakWrite(writeChannelIDmax,maxTempF,'timestamp',timeMaxTemp,'WriteKey',w
riteAPIKeymax);
thingSpeakWrite(writeChannelIDmin,minTempF,'timestamp',timeMinTemp,'WriteKey',w
riteAPIKeymin);
```

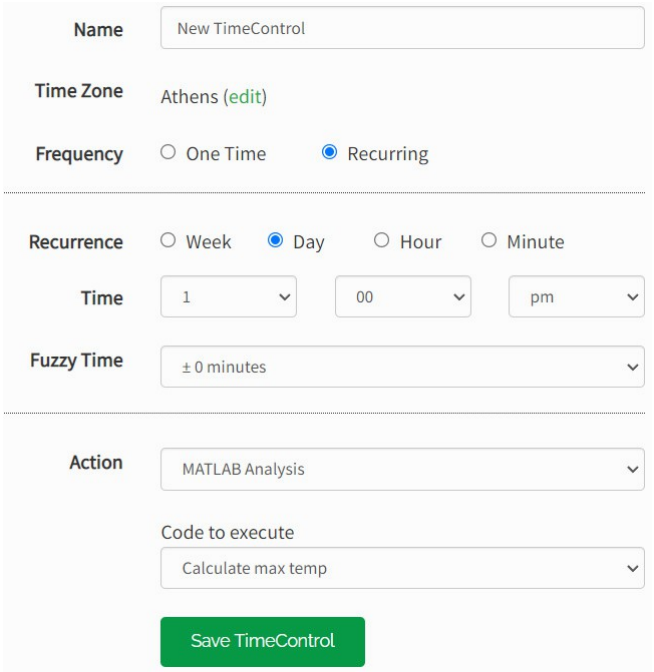
Αφού γράψουμε τον κώδικα πατάμε στο κάτω μέρος Save για να αποθηκευτεί, ή Save and run για να αποθηκευτεί και να τρέξει. Αν υπάρχει κάποιο σφάλμα θα βγάλει μήνυμα στο monitor, κάτω από τα κουμπιά save.

Για να τρέχει ο κώδικας της ανάλυσης καθημερινά και σε συγκεκριμένη ώρα, δημιουργήθηκε ένα Time control. Το Time control είναι μια δυνατότητα του thingspeak που μπορεί να ενεργοποιεί μια ενέργεια, στην περίπτωσή μας την ανάλυση, σε συγκεκριμένη χρονική στιγμή που ορίζουμε εμείς. Συγκεκριμένα μπορούμε να επιλέξουμε η ενέργεια να εκτελείτε μόνο μια φορά, κάθε εβδομάδα, κάθε μέρα και σε συγκεκριμένη ώρα, κάθε ώρα ή κάθε λεπτό. Για την δημιουργία time control πατάμε στην μπάρα μενού Apps και επιλέγουμε Time control. Στην καρτέλα που θα ανοίξει βλέπουμε τα υπάρχοντα time controls τα οποία μπορούμε να τα ανοίξουμε για να τα δούμε ή να τα επεξεργαστούμε. Για να δημιουργήσουμε νέο πατάμε στο κουμπί New TimeControl.




Εικόνα 6.11: Καρτέλα Time Control

Στην καρτέλα που θα ανοίξει υπάρχουν οι ρυθμίσεις για το time control. Από εκεί δίνουμε όνομα στο time control, επιλέγουμε τη ζώνη ώρας, τη συχνότητα που θα εκτελείτε (μία φορά ή επαναλαμβανόμενα), κάθε πότε θα επαναλαμβάνετε (κάθε εβδομάδα, κάθε μέρα, κάθε ώρα ή κάθε λεπτό), την ώρα που θα εκτελείτε, ένα χρόνο σε λεπτά (fuzzy time) γύρω από τον προγραμματισμένο χρόνο για να εκτελεστεί η ενέργεια και τέλος την ενέργεια που θέλουμε να εκτελεστεί και τον κώδικα αυτής της ενέργειας. Αφού ολοκληρώσουμε τις ρυθμίσεις πατάμε Save TimeControl.




Εικόνα 6.12: Καρτέλα ρυθμίσεων TimeControl

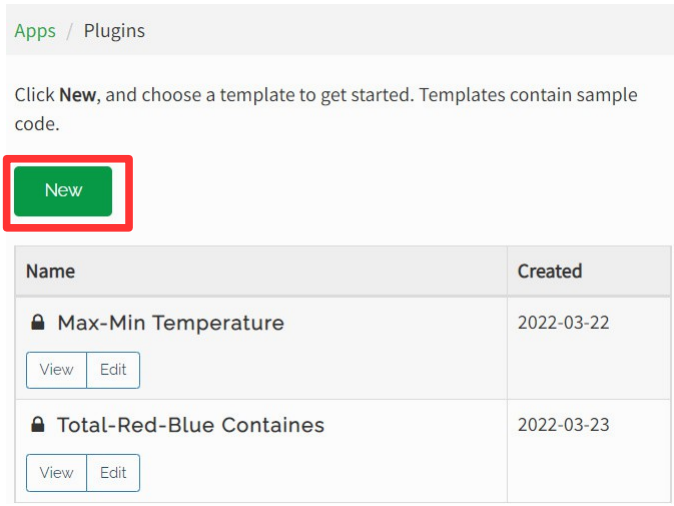
Στην περίπτωση μας θέλουμε η Matlab Analysis και συγκεκριμένα ο κώδικας για τον υπολογισμό της μέγιστης και ελάχιστης θερμοκρασίας να εκτελείτε κάθε μέρα στις 1μμ και τα αποτελέσματα θα

αποθηκεύονται στα κανάλια για την μέγιστη και ελάχιστη θερμοκρασία. Τα γραφήματα των καναλιών αυτών φαίνονται παρακάτω.


Εικόνα 6.13: Γραφήματα μέγιστης και ελάχιστης θερμοκρασίας

## 6.3 Συνδυασμοί Διαγραμμάτων

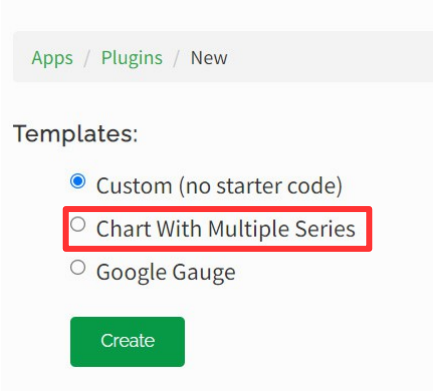
Τέλος δημιουργήθηκαν δύο κοινά διαγράμματα με χρήση plugin του thingspeak, όπου στο ένα προβάλουμε μαζί την μέγιστη και την ελάχιστη θερμοκρασία και στο άλλο τον συνολικό αριθμό των δοχείων μαζί με τον αριθμό των κόκκινων και των μπλε δοχείων. Για να δημιουργήσουμε κάποιο plugin πατάμε στη μπάρα μενού Apps και επιλέγουμε Plugins. Στην καρτέλα που θα ανοίξει βλέπουμε όλα τα υπάρχοντα plugin τα οποία μπορούμε να προβάλουμε πατώντας view ή να επεξεργαστούμε πατώντας edit. Για να δημιουργήσουμε νέο επιλέγουμε New.




Εικόνα 6.14: Καρτέλα Plugins

Πατώντας New θα ανοίξει μια άλλη καρτέλα στην οποία επιλέγουμε κάποιο από τα τρία template που υπάρχουν, custom (χωρίς έτοιμο κώδικα), Chart with multiple series ή Google gauge.

Επιλέγουμε Chart with multiple series και πατάμε create.




Εικόνα 6.15: Καρτέλα επιλογής template για plugin

Στη συνέχεια θα ανοίξει μια καρτέλα με τρία πεδία κώδικα. Το πρώτο θα είναι κώδικας HTML το δεύτερο CSS και το τρίτο JavaScript. Από αυτά θα τροποποιήσουμε τον κώδικα Javascript. Επίσης θα αλλάξουμε και κάποιες γραμμές του HTML γιατί ο αρχικός κώδικας δεν λειτουργεί και δεν εμφανίζει τα διαγράμματα. Ο κώδικας HTML φαίνεται παρακάτω.

```
<!DOCTYPE html>
<html style="height: 100%;">
  <head>
    <script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
    <script type="text/javascript" src="//thingspeak.com/highcharts-3.0.8.js"></script>
    <script type="text/javascript" src="//thingspeak.com/exporting.js"></script>
    %%PLUGIN_CSS%%
    %%PLUGIN_JAVASCRIPT%%
</head>
<body>
  <div id="chart-container">
    <img alt="Ajax loader" src="//thingspeak.com/assets/ajax-loader.gif" style="position: absolute; margin:
auto; top: 0; left: 0; right: 0; bottom: 0;" />
  </div>
</body>
</html>
```

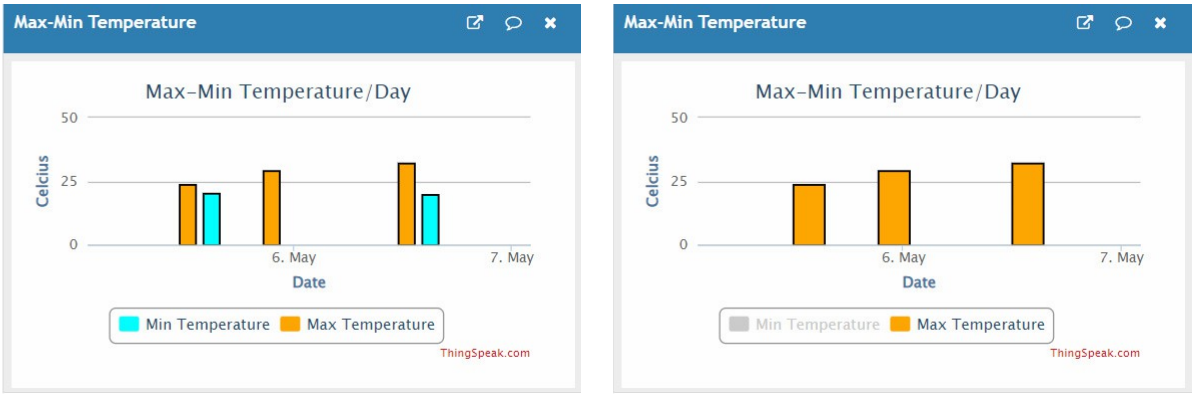
Για την αναπαράσταση της μέγιστης και ελάχιστης θερμοκρασίας, ο κώδικας JavaScript είναι ο παρακάτω.

```
<script type="text/javascript">
  //Μεταβλητές με τα στοιχεία του καναλιού της μέγιστης θερμοκρασίας
  var series_1_channel_id = 1682414;
  var series_1_field_number = 1;
  var series_1_read_api_key = 'CYD8J70INIVD57YE';
  var series_1_results = 5;
  var series_1_color = '#ffa500';
  //Μεταβλητές με τα στοιχεία του καναλιού της ελάχιστης θερμοκρασίας
```

```
 var series_2_channel_id = 1682416;
 var series_2_field_number = 1;
 var series_2_read_api_key = 'MWSMMHMHZ67SR1S5';
 var series_2_results = 5;
 var series_2_color = '#00ffff';
 // Μεταβλητή με τον τίτλο του διαγράμματος
 var chart_title = 'Max-Min Temperature/Day';
 // Μεταβλητή με τον τίτλο του άξονα Y
 var y_axis_title = 'Celcius';
 //Διαβάζει την ώρα του χρήστη
 var my_offset = new Date().getTimezoneOffset();
 // Μεταβλητή my_chart
 var my_chart;
 // when the document is ready
 $(document).on('ready', function() {
  //Προσθέτει ένα κενό διάγραμμα
  addChart();
  // Προσθέτει τα δεδομένα του πρώτου καναλιού στο διάγραμμα
  addSeries(series_1_channel_id, series_1_field_number, series_1_read_api_key, series_1_results,
series_1_color);
  // Προσθέτει τα δεδομένα του δεύτερου καναλιού στο διάγραμμα
  addSeries(series_2_channel_id, series_2_field_number, series_2_read_api_key, series_2_results,
series_2_color);
 });
 // add the base chart
 function addChart() {
  var localDate;
  // Χαρακτηριστικά του γραφήματος
  var chartOptions = {
   chart: {
    renderTo: 'chart-container',
    defaultSeriesType: 'column',
    backgroundColor: '#ffffff',
    events: { }
   },
   title: { text: chart_title },
   plotOptions: {
    column: {
     pointPadding: 0.8,   //Το κενό ανάμεσα από κάθε στήλη ή μπάρα στον άξονα x
      borderWidth: 1.5,  //Το πλάτος του περιγράμματος κάθε στήλης ή μπάρας
     borderColor: 'black', //Το χρώμα του πλαισίου,..φαίνεται μόνο αν το borderWidth >0
      groupPadding: 0.8,  //Το κενό ανάμεσα από κάθε ομάδα τιμών στον άξονα x

    }
   },
   tooltip: {
    formatter: function() {
     var d = new Date(this.x + (my_offset*60000));
     var n = (this.point.name === undefined) ? '' : '<br>' + this.point.name;
     return this.series.name + ':<b>' + this.y + '</b>' + n + '<br>' + d.toDateString() + '<br>' +
d.toTimeString().replace(/\(.*\)/, "");
    }
   },
   xAxis: {
    type: 'datetime',
    title: { text: 'Date' }
   },
   yAxis: { title: { text: y_axis_title } },
```

```
    exporting: { enabled: false },
    legend: { enabled: true}, //Εμφανίζει στο κάτω μέρος του γραφήματος υπόμνημα για τα δεδομένα του γραφήματος και επιλογή να κρύβουμε κάποιο και να βλέπουμε τα υπόλοιπα
    credits: {
     text: 'ThingSpeak.com',
     href: 'https://thingspeak.com/',
     style: { color: '#D62020' }
    }
   };
   // Δημιουργεί το γράφημα
   my_chart = new Highcharts.Chart(chartOptions);
 }
 // Προσθέτει τις γραμμές στο γράφημα
 function addSeries(channel_id, field_number, api_key, results, color) {
  var field_name = 'field' + field_number;
  // get the data with a webservice call
  $.getJSON('https://api.thingspeak.com/channels/' + channel_id + '/fields/' + field_number + '.json?offset=0&round=2&results=' + results + '&api_key=' + api_key, function(data) {
    // blank array for holding chart data
    var chart_data = [];
    // iterate through each feed
    $.each(data.feeds, function() {
     var point = new Highcharts.Point();
     // θέτει τις κατάλληλες τιμές
     var value = this[field_name];
     point.x = getChartDate(this.created_at);
     point.y = parseFloat(value);
     // Προσθέτει τοποθεσία αν είναι δυνατόν
     if (this.location) { point.name = this.location; }
     // if a numerical value exists add it
     if (!isNaN(parseInt(value))) { chart_data.push(point); }
    });
    // Προσθέτει τα δεδομένα του γραφήματος
    my_chart.addSeries({ data: chart_data, name: data.channel[field_name], color: color });
  });
 }
  // converts date format from JSON
 function getChartDate(d) {
  // offset in minutes is converted to milliseconds and subtracted so that chart's x-axis is correct
  return Date.parse(d) - (my_offset * 60000);
 }
</script>
```

Το διάγραμμα που δημιουργήθηκε φαίνεται στην παρακάτω εικόνα. Από το υπόμνημα μπορούμε να επιλέγουμε ποια στήλη θέλουμε να βλέπουμε κάνοντας κλικ. Αυτό συμβαίνει γιατί στο παραπάνω κώδικα έχουμε δώσει την τιμή true στο legend.




Εικόνα 6.16: Διάγραμμα μέγιστης και ελάχιστης θερμοκρασίας με υπόμνημα

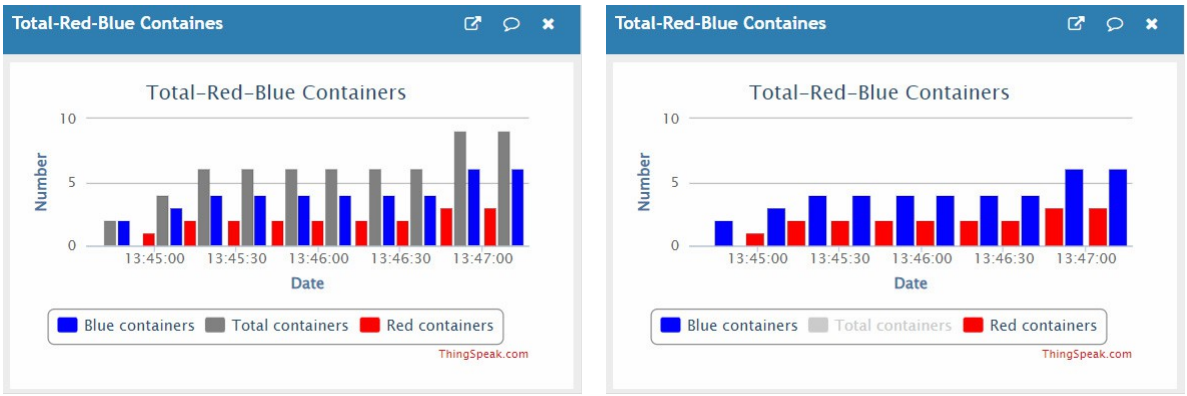
Για τον αριθμό των δοχείων (συνολικών, κόκκινων, μπλε) σε κοινό διάγραμμα, χρησιμοποιήθηκε ο ίδιος κώδικας με την μόνη διαφορά ότι στην αρχή προστέθηκαν κάποια στοιχεία για το τρίτο γράφημα. Ο κώδικας φαίνεται παρακάτω.

```
<script type="text/javascript">
  //Μεταβλητές με τα στοιχεία για τον συνολικό αριθμό δοχείων
  var series_1_channel_id = 1676414;
  var series_1_field_number = 3;
  var series_1_read_api_key = 'CZAI10GTNSHUG7R3';
  var series_1_results = 10;
  var series_1_color = '#808080';
  //Μεταβλητές με τα στοιχεία για τον αριθμό των κόκκινων δοχείων
  var series_2_channel_id = 1676414;
  var series_2_field_number = 4;
  var series_2_read_api_key = 'CZAI10GTNSHUG7R3';
  var series_2_results = 10;
  var series_2_color = '#ff0000';
    //Μεταβλητές με τα στοιχεία για τον αριθμό των μπλε δοχείων
  var series_3_channel_id = 1676414;
  var series_3_field_number = 5;
  var series_3_read_api_key = 'CZAI10GTNSHUG7R3';
  var series_3_results = 10;
  var series_3_color = '#0000ff';
  var chart_title = 'Total-Red-Blue Containers';
  var y_axis_title = 'Number';
  var my_offset = new Date().getTimezoneOffset();
  var my_chart;
  $(document).on('ready', function() {
    //Προσθέτει ένα κενό διάγραμμα
    addChart();
    //Προσθέτει τα δεδομένα για τον συνολικό αριθμό των δοχείων
    addSeries(series_1_channel_id, series_1_field_number, series_1_read_api_key, series_1_results,
series_1_color);
    //Προσθέτει τα δεδομένα για τον αριθμό των κόκκινων δοχείων
    addSeries(series_2_channel_id, series_2_field_number, series_2_read_api_key, series_2_results,
series_2_color);
    //Προσθέτει τα δεδομένα για τον αριθμό των μπλε δοχείων
    addSeries(series_3_channel_id, series_3_field_number, series_3_read_api_key, series_3_results,
series_3_color);
```

```
});
function addChart() {
  var localDate;
  var chartOptions = {
    chart: {
      renderTo: 'chart-container',
      defaultSeriesType: 'column',
      backgroundColor: '#ffffff',
      events: { }
    },
    title: { text: chart_title },
    plotOptions: {
     column: {
       pointPadding: 0.9,   //Το κενό ανάμεσα από κάθε στήλη ή μπάρα στον άξονα x
       borderWidth: 0.1,    //Το πλάτος του περιγράμματος κάθε στήλης ή μπάρας
       borderColor: 'black', //Το χρώμα του πλαισίου,..φαίνεται μόνο αν το borderWidth >0
       groupPadding: 1,  //Το κενό ανάμεσα από κάθε ομάδα τιμών στον άξονα x
      }
    },
    tooltip: {
      formatter: function() {
        var d = new Date(this.x + (my_offset*60000));
        var n = (this.point.name === undefined) ? '' : '<br>' + this.point.name;
        return this.series.name + ':<b>' + this.y + '</b>' + n + '<br>' + d.toDateString() + '<br>' +
d.toTimeString().replace(/\(.*\)/, "");
      }
    },
    xAxis: {
      type: 'datetime',
      title: { text: 'Date' }
    },
    yAxis: { title: { text: y_axis_title } },
    exporting: { enabled: false },
    legend: { enabled: true }, //Εμφανίζει στο κάτω μέρος του γραφήματος υπόμνημα για τα δεδομένα του
γραφήματος και επιλογή να κρύβουμε κάποιο και να βλέπουμε τα υπόλοιπα
    credits: {
      text: 'ThingSpeak.com',
      href: 'https://thingspeak.com/',
      style: { color: '#D62020' }
    }
  };
  my_chart = new Highcharts.Chart(chartOptions);
}
//Προσθέτει τις γραμμές στο γράφημα
function addSeries(channel_id, field_number, api_key, results, color) {
  var field_name = 'field' + field_number;
  $.getJSON('https://api.thingspeak.com/channels/' + channel_id + '/fields/' + field_number + '.json?
offset=0&round=2&results=' + results + '&api_key=' + api_key, function(data) {
    var chart_data = [];
    $.each(data.feeds, function() {
      var point = new Highcharts.Point();
      var value = this[field_name];
      point.x = getChartDate(this.created_at);
      point.y = parseFloat(value);
      if (this.location) { point.name = this.location; }
      if (!isNaN(parseInt(value))) { chart_data.push(point); }
    });
    //Προσθέτει τα δεδομένα του γραφήματος
```

```
    my_chart.addSeries({ data: chart_data, name: data.channel[field_name], color: color });
  });
}
function getChartDate(d) {
  return Date.parse(d) - (my_offset * 60000);
}
</script>
```

Το διάγραμμα που δημιουργήθηκε φαίνεται στην παρακάτω εικόνα. Επίσης και σε αυτό υπάρχει το υπόμνημα για να μπορούμε κάνοντας κλικ να επιλέγουμε ποιες στήλες θα βλέπουμε.




Εικόνα 6.17: Διάγραμμα Συνολικών – Κόκκινων – Μπλε δοχείων με υπόμνημα

Για να εισάγουμε σε κάποιο κανάλι μας τα διαγράμματα που φτιάξαμε, πατάμε στο πάνω μέρος του καναλιού Add Visualizations και θα μας ανοίξει μια καρτέλα με όλα τα plugins που έχουμε δημιουργήσει και δεν έχουν εισαχθεί στο κανάλι. Από εκεί επιλέγουμε όποιο θέλουμε και το εισάγουμε στο κανάλι.

# Κεφάλαιο 7ο: Συμπεράσματα και μελλοντικές επεκτάσεις

## 7.1 Ανακεφαλαίωση και Συμπεράσματα

Η πτυχιακή εργασία αυτή αφορούσε την κατασκευή μιας γραμμής παραγωγής με ταινιόδρομο, ρομποτικό βραχίονα, δεξαμενή με αντλία και διάφορους αισθητήρες, χρησιμοποιώντας την πλατφόρμα arduino, και τον προγραμματισμό της ώστε να εκτελείτε μια διεργασία για την μεταφορά δοχείων κόκκινου και μπλε χρώματος, την πλήρωση των κόκκινων δοχείων με ρευστό και στη συνέχεια την απόσυρση των μπλε δοχείων από τον ταινιόδρομο και την μετακίνηση των κόκκινων σε άλλη θέση από τον ρομποτικό βραχίονα. Επιπλέον χρησιμοποιώντας το λογισμικό Labview ανοίξαμε έναν δίαυλο επικοινωνίας με την πλακέτα arduino μέσω σειριακής επικοινωνίας (USB) με τη χρήση του πρωτοκόλλου VISA (Virtual Instrument Software Architecture) και δημιουργήθηκε ένα περιβάλλον ελέγχου της διεργασίας αυτής όπου ο χρήστης μπορεί να ελέγχει τιμές αισθητήρων και καταστάσεων ενεργοποιητών μέσω εικονικών οργάνων, αλλά και να επέμβει σε κάποιες λειτουργίες όπως τον κινητήρα του ταινιόδρομου και την αντλία. Τέλος ο arduino συνδέθηκε στο διαδίκτυο με τη χρήση του ethernet shield και επικοινωνεί με την διαδικτυακή πλατφόρμα ThingSpeak, στην οποία στέλνει κάθε δεκαπέντε δευτερόλεπτα τα δεδομένα από τις μετρήσεις των αισθητήρων, τις καταστάσεις των led, του κινητήρα και της αντλίας, τα οποία επεξεργάζονται και προβάλλονται σε γραφήματα και widgets, δημιουργήθηκε μια ανάλυση η οποία τρέχει σε καθημερινή βάση για τον υπολογισμό της μέγιστης και ελάχιστης θερμοκρασίας του υγρού στη δεξαμενή, και επίσης δημιουργήθηκαν συνδυασμοί διαγραμμάτων με χρήση plugin που διαθέτει το thingspeak.

Με την εργασία αυτή γνώρισα και έμαθα πως μπορούμε να δημιουργήσουμε ένα σύστημα αυτοματισμού για κάποια εφαρμογή με την πλατφόρμα arduino και το λογισμικό Labview, πως να επιλέξω τα κατάλληλα υλικά (αισθητήρες, ενεργοποιητές, οδηγούς κινητήρων και άλλα ηλεκτρονικά και μηχανολογικά υλικά) για τις ανάγκες της εφαρμογής. Έμαθα πως λειτουργούν και πως συνδέονται, τις δυνατότητες τους και τον προγραμματισμό τους. Επίσης κατανόησα την έννοια του IoT (Internet of Things) μέσω της πλατφόρμας ThingSpeak, ανακάλυψα και χρησιμοποίησα τις δυνατότητές της στην εργασία αυτή (δημιουργώντας αναλύσεις, widgets και γραφήματα) και τέλος είδα πως όλα αυτά μπορούν να συνδυαστούν και να χρησιμοποιηθούν από απλές καθημερινές μέχρι και βιομηχανικές εφαρμογές, και να τις κάνουν πιο "έξυπνες".

## 7.2 Μελλοντικές Επεκτάσεις και βελτιώσεις

Η παραπάνω κατασκευή σίγουρα δέχεται βελτιώσεις και επεκτάσεις. Όσον αφορά τις βελτιώσεις θα μπορούσαν να μπούνε καλύτερης ποιότητας αισθητήρες για ακριβέστερες μετρήσεις και καλύτεροι σερβοκινητήρες ή σερβοκινητήρες με αναλογικό feedback για ακριβέστερο έλεγχο θέσης καθώς και ένας arduino due που έχει 32 bit επεξεργαστή και θα είναι πιο γρήγορος ειδικά στην επικοινωνία με το Labview.

Επεκτάσεις μπορούν να γίνουν πολλές και εξαρτάτε από την φαντασία του καθενός. Κάποιες ενδεικτικές μπορεί να είναι η προσθήκη περισσότερων αισθητήρων όπως ρεύματος, τάσης, πίεσης ή και κάποια κάμερα με σκοπό να δημιουργηθούν περισσότερα σενάρια.

Από την πλευρά του Labview, θα μπορούσε να επεκταθεί το περιβάλλον και να έχουμε περισσότερες καταγραφές αισθητήρων ή και παρακολούθηση κάποιας κάμερας.

Τέλος θα μπορούσε το Labview να συνδεθεί με τον arduino μέσω tcp/ip πρωτοκόλλου και το Labview να στέλνει τα δεδομένα στον Thingspeak.

## Βιβλιογραφία

(1) [Ηλεκτρονικό]
https://el.wikipedia.org/wiki/Αυτοματισμός

(2)[Ηλεκτρονικό]
Πτυχιακή εργασία Ιωακείμ Μαμάτας, με θεμα Σχεδίαση και υλοποίηση εκπαιδευτικού αναπτύγματος με τη χρήση του PLC S7-200, για την προσομοίωση εφαρμογών αυτοματισμού

(3) [Ηλεκτρονικό]
Πτυχιακή εργασία Ακρίδας Αντώνιος, με θέμα Δομή και λειτουργία των PLC

(4) [Ηλεκτρονικό]
https://www.youtube.com/watch?v=9kWFaOGZn7Q

(5) [Ηλεκτρονικό]
https://www.youtube.com/watch?v=apJLyShtmB4

(6) [Ηλεκτρονικό]
https://el.wikipedia.org/wiki/Arduino#Λογισμικ%CF%8C

(7) [Ηλεκτρονικό]
https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics

(8) [Ηλεκτρονικό]
https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#libraries

(9) [Ηλεκτρονικό]
https://eclass.upatras.gr/modules/document/file.php/EE795/Useful_manual_Introduction_to_LabVIEW.pdf

(10) [Ηλεκτρονικό]
http://eclass.teipir.gr/openeclass/modules/document/file.php/HYS113/04%20%CE%95%CE%B3%CF%87%CE%B5%CE%B9%CF%81%CE%AF%CE%B4%CE%B9%CE%BF%20%CE%B3%CE%B9%CE%B1%20%CF%84%CE%B7%CE%BD%20%CE%95%CE%BA%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7%20%CF%84%CE%BF%CF%85%20Labview.pdf

(11) [Ηλεκτρονικό]
https://www.ni.com/en-us/innovations/white-papers/13/hardware-integration-with-ni-labview.html

(12) [Ηλεκτρονικό]
https://www.ni.com/en-us/support/documentation/supplemental/06/ni-visa-overview.html

(13) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/Virtual_instrument_software_architecture

(14) [Ηλεκτρονικό]
https://www.ni.com/en-us/support/downloads/drivers/download.ni-visa.html#442805

(15) [Ηλεκτρονικό]
https://el.wikipedia.org/wiki/Διαδίκτυο_των_πραγμάτων

(16) [Ηλεκτρονικό]
https://thingspeak.com/pages/learn_more

(17) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/ThingSpeak

(18) [Ηλεκτρονικό]
http://users.sch.gr/kgiannaras/fritzing/γνωριμία-με-το-fritzing.html

(19) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/Fritzing

(20) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/Notepad++

(21) [Ηλεκτρονικό]
https://www.arduino.cc/en/Main/arduinoBoardMega2560

(22) [Ηλεκτρονικό]
http://users.sch.gr/asal1/material/seminaria/teliko24_1.pdf

(23) [Ηλεκτρονικό]
https://docs.arduino.cc/retired/shields/arduino-ethernet-shield-without-poe-module

(24) [Ηλεκτρονικό]
https://www.electronicshub.org/ir-sensor/

(25) [Ηλεκτρονικό]
https://www.elprocus.com/lm393-ic-pin-configuration-circuit-diagram-and-its-working/

(26) [Ηλεκτρονικό]
https://www.waveshare.com/wiki/Infrared_Proximity_Sensor

(27) [Ηλεκτρονικό]
http://grobotronics.com/images/companies/1/HC-SR04Users_Manual.pdf

(28) [Ηλεκτρονικό]
https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

(29) [Ηλεκτρονικό]
http://grobotronics.com/images/companies/1/DS18B20.pdf

(30) [Ηλεκτρονικό]
https://www.elprocus.com/ds18b20-temperature-sensor/

(31) [Ηλεκτρονικό]
https://www.elecrow.com/wiki/index.php?title=TCS3200_Colour_Sensor_Module

(32) [Ηλεκτρονικό]
https://create.arduino.cc/projecthub/SurtrTech/color-detection-using-tcs3200-230-84a663

(33) [Ηλεκτρονικό]
https://www.sparkfun.com/servos

(34) [Ηλεκτρονικό]
https://grobotronics.com/servo-standard-11kg.cm-metal-gears-waveshare-mg996r.html

(35) [Ηλεκτρονικό]
https://grobotronics.com/metal-gearmotor-25mm-100rpm-9-12v.html

(36) [Ηλεκτρονικό]
https://grobotronics.com/liquid-pump-motor-micro-12v.html

(37) [Ηλεκτρονικό]
http://grobotronics.com/images/companies/1/datasheets/mXxwur.pdf?1515498866794

(38) [Ηλεκτρονικό]
http://grobotronics.com/images/companies/1/datasheets/L298N-module-informatie.pdf

(39) [Ηλεκτρονικό]
http://www.handsontec.com/dataspecs/L298N Motor Driver.pdf

(40) [Ηλεκτρονικό]
https://grobotronics.com/images/companies/1/SN-HS-60.pdf?1570443285273

(41) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/Buck_converter

(42) [Ηλεκτρονικό]
https://www.ti.com/lit/ds/symlink/lm2596.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1620718466883

(43) [Ηλεκτρονικό]
https://grobotronics.com/dc-dc-step-down-1.3-35v-3a-el.html

(44) [Ηλεκτρονικό]
https://grobotronics.com/push-button-11x13-el.html

(45) [Ηλεκτρονικό]
https://grobotronics.com/led-diffused-5mm-prasino.html

(46) [Ηλεκτρονικό]
https://grobotronics.com/rocker-switch-on-on-spdt-3a-250vac-mini-black.html

(47) [Ηλεκτρονικό]
https://en.wikipedia.org/wiki/Poly(methyl_methacrylate)

(48) [Ηλεκτρονικό]
https://www.plexiglas-petropoulos.gr/το-υλικό-plexiglass/

(49) [Ηλεκτρονικό]
https://grobotronics.com/aluminum-tube-1.0od-x-2.0l-clone.html

(50) [Ηλεκτρονικό]
https://grobotronics.com/627zz-7mm-bore-22mm-od.html

(51) [Ηλεκτρονικό]
https://grobotronics.com/shaft-coupler-solid-4mm-5mm.html

(52) [Ηλεκτρονικό]
https://grobotronics.com/aluminum-motor-mount-f.html

(53) [Ηλεκτρονικό]
https://grobotronics.com/arduino-mega-enclosure-black-plastic.html

(54) [Ηλεκτρονικό]
https://grobotronics.com/project-box-83x54x30mm-black-g1020b.html

## Συντομογραφίες

**ABS:** Acrylonitrile Butadiene Styrene

**API:** Application Programming Interface

**AREF:** Anaglog Reference

**ASCII:** American Standard Code for Information Interchange

**AVR:** Advanced Virtual RISC

**CAD:** Computer-Aided Design

**COLL:** Collisions

**CPP :** Capacitor with Polypropylene Film

**CSS:** Cascading Style Sheets

**CSV:** Comma-Separated Values

**DC:** Direct Current

**EEPROM:** Electrically Erasable Programmable Read-Only Memory

**EMI filter:** Electromagnetic Interference filter

**FULLD:** Full Duplex

**FWD:** Free Wheel Diode

**GPIB:** General Purpose Interface Bus

**HTML:** Hypertext Markup Language

**HTTP:** Hypertext Transfer Protocol

**ICSP:** In-Circuit Serial Programming

**ICs:** Intergrated Circuits

**I2C:** Inter-Integrated Circuit

**IDE:** Integrated Development Environment

**I/O:** Input/Output

**IOREF:** Input/Output Reference

**IoT:** Internet of Things

**IP:** Internet Protocol

**IR:** Infrared Radiation

**JSON:** JavaScript Object Notation

**Labview:** Laboratory Virtual Instrument Engineering Workbench

**LXI:** LAN eXtensions for Instrumentation

**MISO:** Master In Slave Out

**MOSI:** Master Out Slave In

**MQTT:** MQ Telemetry Transport

**NBR:** Nitrile-Butadiene Rubber

**NI:** Natioanl Instrumets

**OPC:** Open Platform Communications

**PCB:** Printed Circuit Board

**PMMA:** Poly-Methyl Methacrylate

**PLC:** Progammable Logic Controller

**PoE:** Power Over Ethernet

**PVC:** Polyvinyl Chloride

**PWM:** Pulse Width Modulation

**PWR:** Power

**PXI:** PCI eXtensions for Instrumentation

**RISC:** Reduced Instruction Set Computer

**RJ-45:** Registered Jack-45

**ROM:** Read Only Memory

**RX:** Reciever

**SCK:** Serial Clock

**SCL:** Serial Clock Line

**SD card:** Secure Digital

**SDA:** Serial Data Line

**SMPS:** Switched-Mode Power Supply

**SPDT:** Single Pole Double Throw

**SPST-NO:** Single pole, single throw, normally open

**SRAM:** Static Random Access Memory

**SS:** Slave Select

**TCP:** Transmission Control Protocol

**TX:** Transmiter

**UART:** Universal Asynchronous Receiver-Transmitter

**UCS-2:** Universal Character Set

**UDP:** User Datagram Protocol

**USB:** Universal Serial Bus

**UTF-8:** UCS Transformation Format 8

**VI:** Virtual Instrument

**VISA:** Virtual Instrument Software Architecture

**VXI:** VME eXtensions for Instrumentation

**XML:** Extensible Markup Language

## Παράρτημα Α

Τελικός Κώδικας

```
#include <VarSpeedServo.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SPI.h>
#include <Ethernet.h>
#include <ThingSpeak.h>
//temperature sensor
#define ONE_WIRE_BUS 39
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
//Ultrasonic sensor
#define trigpin 49
#define echopin 48
//Dc motor
int enB = 5;
int in3 = 23;
int in4 = 22;
//Dc pump
int enA = 6;
int in1 = 25;
int in2 = 24;
//IR sensor_1
const int pinIRd1 = 41;
const int pinLED1 = 40;
int IRvalueD1 = 0;
//IR sensor_2
const int pinIRd2 = 43;
const int pinLED2= 42;
int IRvalueD2 = 0;
//buttons
int StartButtonStatus = 0;
int StopButtonStatus = 0;
// leds
int greenLed = 35;
int redLed = 34;
int greenledstate = 0;
int redledstate = 1;
//color sensor
#define S0 26
#define S1 27
#define S2 29
#define S3 28
#define sensorOut 30
#define led 31
int redfrequency = 0;
int greenfrequency = 0;
int bluefrequency = 0;
int ledvalue = 0;
//Servos
VarSpeedServo servo1;
VarSpeedServo servo2;
VarSpeedServo servo3;
VarSpeedServo servo4;
```

```
VarSpeedServo servo5;
const int servoPin1 = 9;
const int servoPin2 = 8;
const int servoPin3 = 11;
const int servoPin4 = 12;
const int servoPin5 = 13;
int myarray[80] ;      //πίνακας που αποθηκεύετε το χρώμα των δοχείων.
int N = 0;       //δείκτης θέσης για την εγγραφή στον πίνακα.
int K = 0;      //δείκτης θέσης για την ανάγνωση του πίνακα.
int T = LOW;     //μεταβλητή για την επιλογή μεταξύ των δύο θέσεων που θα αφήσει τα δοχεία ο βραχίονας
int Red_color = 0;
int Blue_color = 0;
int Total_containers = 0;
int motorState = 0;
int pumpState = 0;
//Labview
char labview;
String bufer;
//Ethernet
byte mac[] = { 0xFE, 0xAD, 0xBF, 0xBE, 0xEE,0xED};
IPAddress ip(192,168,2,10);
EthernetClient client;
//ThingSpeak
unsigned long mySensorsChannelNumber = 1676414;
const char * myWriteAPIKey_sensors = "IW0YRSWAFG9AX7U0";
long lastWriteTime=0;

void setup() {
//temperature sensor
sensors.begin();
//Ultrasonic sensor
pinMode(trigpin, OUTPUT);
pinMode(echopin, INPUT);
 //Dc motor
pinMode(enB, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
// Dc pump
pinMode(enA, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
//buttons
pinMode(44, INPUT_PULLUP);
pinMode(45, INPUT_PULLUP);
//leds
pinMode(greenLed, OUTPUT);
pinMode(redLed, OUTPUT);
digitalWrite(greenLed, LOW);
digitalWrite(redLed,HIGH);
//IR sensor_1
pinMode(pinIRd1,INPUT);
pinMode(pinLED1,OUTPUT);
//IR sensor_2
pinMode(pinIRd2,INPUT);
pinMode(pinLED2,OUTPUT);
//color sensor
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
```

```
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(sensorOut, INPUT);
pinMode (led,OUTPUT);
digitalWrite(led,LOW);
// Setting frequency-scaling to 20%
digitalWrite(S0,HIGH);
digitalWrite(S1,LOW);
//Robotic Arm Servos
servo1.attach(servoPin1);
servo1.write(73,80,false);
servo2.attach(servoPin2);
servo2.write(160,150,false);
servo3.attach(servoPin3);
servo3.write(0,150,false);
servo4.attach(servoPin4);
servo4.write(90,150,false);
servo5.attach(servoPin5);
servo5.write(110,100,false);
Serial.begin(9600);
Ethernet.begin(mac, ip);
ThingSpeak.begin(client);
delay(1000);
}
void loop() {
float duration, distance, level;
//Μέτρηση και υπολογισμός στάθμης της δεξαμενής
digitalWrite(trigpin, HIGH);
delayMicroseconds(10);
digitalWrite(trigpin, LOW);
duration = pulseIn(echopin,HIGH);
distance = duration/51;
distance = distance-3.2;
distance = 10-distance;
level = 10*distance;
//Μέτρηση θερμοκρασίας
sensors.requestTemperatures();
float tempC = sensors.getTempCByIndex(0);
//Υπολογισμός των συνολικών δοχείων
Total_containers = Red_color + Blue_color;

if (StopButtonStatus == digitalRead(44)){
  delay(10);
  digitalWrite(redLed, HIGH);
  digitalWrite(greenLed, LOW);
  greenledstate = 0;
  redledstate = 1;
  motorState = 0;
  pumpState = 0;
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  }
if (StartButtonStatus == digitalRead(45)&& digitalRead(pinIRd1) == HIGH && digitalRead(pinIRd2) ==
HIGH && level >= 30) { //Το && είναι η λογική πράξη and
  delay(10);
  digitalWrite(redLed, LOW);
```

```
  digitalWrite(greenLed, HIGH);
  greenledstate = 1;
  redledstate = 0;
  motorState = 1;
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 150);
}
if (digitalRead(pinIRd1) == LOW && (greenledstate == 1 || digitalRead(45) == LOW)) {
  delay(200);
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, HIGH);
  greenledstate = 1;
  redledstate = 0;
  motorState = 0;
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  digitalWrite (led,HIGH);
  delay(200);
 //Διάβασμα του χρώματος των δοχείων
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  redfrequency = pulseIn(sensorOut, LOW);
  delay(100);
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  greenfrequency = pulseIn(sensorOut, LOW);
  delay(100);
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  bluefrequency = pulseIn(sensorOut, LOW);
  digitalWrite (led,LOW);
    if ((redfrequency>15 && redfrequency< 320)&& (greenfrequency>170 && greenfrequency< 830) &&
(bluefrequency>180 && bluefrequency< 700)) {
      PumpEn();
   }else
  if ((bluefrequency>15 && bluefrequency< 300) && (greenfrequency>190 && greenfrequency< 640)&&
(redfrequency>140 && redfrequency< 780)) {
      myarray[N] = 0;
      ++N ;
      ++Blue_color;
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
      analogWrite(enB, 150);
      motorState = 1;
      delay(500);
    }
  }
if (digitalRead(pinIRd2) == LOW &&  greenledstate == 1) {
   digitalWrite(in3, LOW);
   digitalWrite(in4, LOW);
    if (myarray[K]== 0 && digitalRead(pinIRd2) == LOW){
      ++K;
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
      analogWrite(enB, 150);
      motorState = 1;
      delay(500);
```

```
   } else
    if (myarray[K]== 1 && digitalRead(pinIRd2) == LOW){
      ++K;
      T= !T;
      Move();
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
      analogWrite(enB, 150);
      motorState = 1;
      }
  }
//Επικοινωνία με το Labview
if (Serial.available() > 0)
{
delay(100);
while (Serial.available() > 0)
{
labview = Serial.read();
bufer += labview;
}
if (bufer == "sensors\n") {
  Serial.println(level);
  Serial.println(tempC);
  Serial.println(greenledstate);
  Serial.println(redledstate);
  Serial.println(Total_containers);
  Serial.println(Red_color);
  Serial.println(Blue_color );
  Serial.println(pumpState);
  Serial.println(motorState);
  bufer = "";
   } else
   if (bufer == "stop\n") {
     digitalWrite(redLed, HIGH);
     digitalWrite(greenLed, LOW);
     greenledstate = 0;
     redledstate = 1;
     motorState = 0;
     pumpState = 0;
     digitalWrite(in3, LOW);
     digitalWrite(in4, LOW);
     digitalWrite(in1, LOW);
     digitalWrite(in2, LOW);
     bufer = "";
   } else
     if (bufer == "start\n") {
       digitalWrite(redLed, LOW);
       digitalWrite(greenLed, HIGH);
       greenledstate = 1;
       redledstate = 0;
       motorState = 1;
       digitalWrite(in3, LOW);
       digitalWrite(in4, HIGH);
       analogWrite(enB, 150);
       bufer = "";
     }else
      if (bufer == "pumpon\n") {
        digitalWrite(in1, LOW);
```

```
      digitalWrite(in2, HIGH);
      analogWrite(enA, 140);
      pumpState = 1;
      bufer = "";
    }else
    if (bufer == "pumpoff\n") {
      digitalWrite(in1, LOW);
      digitalWrite(in2, LOW);
       pumpState = 0;
      bufer = "";
    }
 bufer = "";
 }
//Ανέβασμα δεδομένων στον ThingSpeak
if(!client.connected() &&(millis()-lastWriteTime>15000)) {
  ThingSpeak.setField(1,level);
  ThingSpeak.setField(2,tempC);
  ThingSpeak.setField(3,Total_containers);
  ThingSpeak.setField(4,Red_color);
  ThingSpeak.setField(5,Blue_color);
  ThingSpeak.setField(6,greenledstate);
  ThingSpeak.setField(7,redledstate);
  ThingSpeak.writeFields(mySensorsChannelNumber, myWriteAPIKey_sensors);                  // write two values
  lastWriteTime=millis();              // store last write time
  client.stop();
 }
}
void PumpEn() {
 motorState = 0;
 //Μέτρηση και υπολογισμός της στάθμης
 float duration, distance, level;
 digitalWrite(trigpin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigpin, LOW);
 duration = pulseIn(echopin,HIGH);
 distance = duration/51;
 distance = distance-3.2;
 distance = 10-distance;
 level = 10*distance;
 if (level < 30) {
  digitalWrite(redLed, HIGH);
  digitalWrite(greenLed, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  greenledstate = 0;
  redledstate = 1;
  pumpState = 0;
  }else
  if (level >= 30) {
  myarray[N] = 1;
  ++N;
  ++Red_color;
  pumpState = 1;
  greenledstate = 1;
  redledstate = 0;
  digitalWrite(in1, LOW);
```

```
    digitalWrite(in2, HIGH);
    analogWrite(enA, 140);
    delay(3000);
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    pumpState = 0;
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    analogWrite(enB, 150);
    motorState = 1;
    delay(500);
    }
}
void Move() {
  if (T == LOW){   //1η θέση
     delay(300);
     servo1.write(67,10,true);
     servo3.write(12,10,false);
     servo2.write(111,10,true);
     servo3.write(20,10,true);
     delay(300);
     servo4.write(90,10,false);
     servo5.write(135,10,true);
     delay(300);
     servo3.write(34,10,true);
     servo2.write(140,10,false);
     servo3.write(22,6,true);
     servo1.write(180,10,true);
     servo2.write(138,6,false);
     servo3.write(8,6,true);
     servo5.write(108,20,true);
     delay(300);
     servo2.write(170,15,false);
     servo3.write(0,6,true);
     servo3.write(24,10,true);
     servo1.write(73,30,false);
     servo2.write(160,25,false);
     servo3.write(0,8,false);
     delay(300);
  }else
   if (T == HIGH){  //2η θέση
      delay(300);
      servo1.write(67,10,true);
      servo3.write(12,10,false);
      servo2.write(111,10,true);
      servo3.write(20,10,true);
      delay(300);
      servo4.write(90,10,false);
      servo5.write(135,10,true);
      delay(300);
      servo3.write(34,10,true);
      servo2.write(140,10,false);
      servo3.write(20,6,true);
      servo1.write(143,10,true);
      servo2.write(130,6,false);
      servo3.write(10,6,true);
      servo5.write(108,20,true);
      delay(300);
```

```
    servo2.write(170,15,false);
    servo3.write(0,6,true);
    servo3.write(20,10,true);
    servo1.write(73,30,false);
    servo2.write(160,25,false);
    servo3.write(0,8,false);
    delay(300);
  }
}
```