ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΡΟΜΠΟΤΙΚΟΣ ΦΥΛΑΚΑΣ ΣΕ ΒΙΟΜΗΧΑΝΙΚΟΥΣ ΧΩΡΟΥΣ

(ROBOTIC GUARD FOR INDUSTRIAL ENVIRONMENTS)

ΜΠΙΜΠΑ ΛΕΝΤΙΟ — ΑΜ: 133146

ΓΚΑΖΙΝΑΣ ΑΛΕΞΑΝΔΡΟΣ — ΑΜ:133175

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: *ΔΡΑΚΑΚΗ ΜΑΡΙΑ*

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

Περιεχόμενα

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να ευχαριστήσουμε βαθιά την κυρία Μαρία Δρακάκη, υπεύθυνη καθηγήτρια της πτυχιακής, η οποία στήριξε την ιδέα μας από την πρώτη στιγμή, στάθηκε δίπλα μας καθ' όλη την διάρκεια της ομαδικής προσπάθειας μας να την εκπληρώσουμε και συνεχίζει να μας παροτρύνει για επόμενα μελλοντικά βήματα.

Θέλουμε να πούμε και ένα μεγάλο ευχαριστώ σε όλους τους καθηγητές τους Τμήματος Μηχανικών Αυτοματισμού που καθ όλη την διάρκεια των σπουδών μας, μας παρείχαν πολύ υλικό για μελέτη, το οποίο συνέβαλε στην εκπόνηση της πτυχιακής μας.

Τέλος, χρωστάμε ένα μεγάλο ευχαριστώ στις οικογένειες μας που μας στήριζαν σε αυτήν μας την προσπάθεια και όχι μόνο.

ΠΕΡΙΛΗΨΗ

Σε αυτήν την πτυχιακή εργασίας θέλαμε να συνδυάσουμε όσο το δυνατών περισσότερες γνώσεις που λάβαμε κατά την διάρκεια των μαθημάτων μας στην σχολή. Όμως θέλαμε να συνδυάσουμε τεχνολογία και βιομηχανία. Τέλος καταλήξαμε σε ένα ρομπότ, το οποίο μπορεί να περπατάει στα δυο του πόδια και να παρατηρεί τον χώρο.

Μετά από κάποιο χρονικό διάστημα έρευνας που κάναμε για το πως θα μπορούσε να είναι αυτό το ρομπότ, καταλήξαμε σε μια κατασκευή που θα έχει 2 μεγάλα πόδια ώστε να πετύχουμε την ισορροπία του, χωρίς να χρειαστεί να εφαρμόσουμε άλλες τεχνικές (βλ. ανάστροφο εκκρεμές). Επίσης χρησιμοποιήθηκαν τέσσερεις σερβοκινητήρες για την επίτευξη της κίνησης. Για τον έλεγχο αυτών χρησιμοποιήθηκε ένα Arduino Mega.

Το επόμενο πρόβλημα ήταν η ανίχνευση εμποδίων ώστε να μην κολλάει το ρομπότ σε κάποιο σημείο. Έτσι προσθέσαμε ένα αισθητήριο υπερηχητικών κυμάτων ώστε να μπορεί αυτό να εντοπίζει εμπόδια και με μια τεχνική οπισθοχώρησης και αλλαγής κατεύθυνσης καταφέραμε να λύσουμε κ αυτό το πρόβλημα.

Η παρατήρηση του περιβάλλοντος γίνεται με 2 αισθητήρια, θερμοκρασίας & υγρασίας και εύφλεκτων αερίων. Αυτά ελέγχονται από ένα Arduino Uno το οποίο συλλέγει τα δεδομένα και μέσω μιας Wi-Fi κεραίας τα στέλνει σε μια online πλατφόρμα για την παρατήρηση, επεξεργασία και απεικόνισή τους.

1.ΕΙΣΑΓΩΓΗ

Είναι εμφανές ότι στον βιομηχανικό τομέα επενδύονται μεγάλα χρηματικά κεφάλαια στην βελτίωση της παραγωγικής διαδικασίας, του χώρου εργασίας και στην ασφάλεια των εργαζομένων. Σε αυτό το κομμάτι συμβάλει σε μεγάλο βαθμό η εξέλιξη της τεχνολογίας και συγκεκριμένα η ανάπτυξη αυτοματισμών.

Είναι ευνόητο πως θα μπορούσε ο αυτοματισμός να συμβάλει στην παραγωγική διαδικασία αλλά πως θα μπορούσε να βοηθήσει στην βελτίωση του χώρου εργασίας των εργαζομένων;

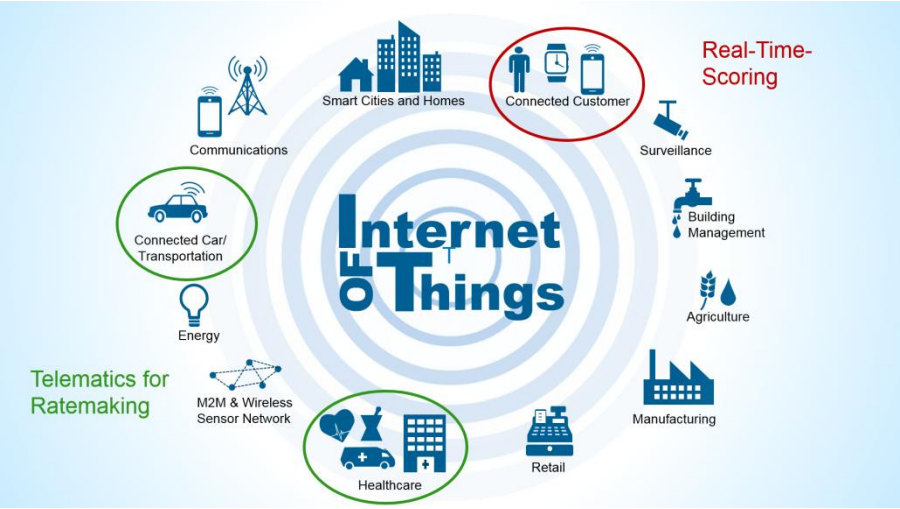
Σε πολλά εργοστάσια υπάρχουν χώροι απαγορευμένοι σε μη τεχνικά καταρτισμένο προσωπικό. Όπως επίσης υπάρχουν χώροι που απαγορεύεται η είσοδος χωρίς κατάλληλο εξοπλισμό. Επιπλέον σε μια βιομηχανία θα βρούμε κάποιον χώρο όπου φυλάγονται φιάλες εύφλεκτων αερίων, ίσως και χώρους με υψηλή θερμοκρασία (μη υποφερτή από τον ανθρώπινο οργανισμό). Ωστόσο, ακόμα και σε τόσο επικίνδυνους χώρους πρέπει να γίνονται εργασίες και συντηρήσεις με όσο το δυνατόν μειωμένο κίνδυνο προς τους εργαζόμενους.

Επισκεπτόμενοι πολλά εργοστάσια, έγινε αντιληπτή η ανάγκη ενός τρόπου εποπτείας αυτών των χώρων σε πραγματικό χρόνο αλλά και σε συγκεκριμένες χρονικές περιόδους. Επομένως, η ανάπτυξη ενός ρομπότ, το οποίο θα μπορεί να περιφέρεται στον χώρο μπορώντας να ξεπερνάει τα εμπόδια που θα βρίσκει μπροστά του αλλά και να παίρνει μετρήσεις για τα στοιχεία του χώρου (θερμοκρασία, υγρασία, επίπεδα εύφλεκτων αερίων) ήταν μια λύση σε ένα μεγάλο πρόβλημα.

Μετά από έρευνα και συζητήσεις με ανθρώπους μέσα από τον βιομηχανικό χώρο, (εργαζομένους, συντηρητές κτλ.) καταλήξαμε στην ανάπτυξη ενός ρομπότ το οποίο μπορεί να περπατάει στα 2 του πόδια, να ξεπερνάει εμπόδια, να παίρνει μετρήσεις θερμοκρασίας, υγρασίας και αερίων, τα οποία να τα στέλνει σε μία διαδικτυακή πλατφόρμα μέσω σύνδεσης Wi-Fi έτσι ώστε να μπορεί χειριστής του να βλέπει απομακρυσμένα την κατάσταση στο εργοστάσιο του. Επιπρόσθετα, το ρομπότ θα μπορεί να τριγυρνάει στον χώρο του εργοστασίου ακόμα και μετά το κλείσιμό του, ούτως ώστε να παίρνει πληροφορίες για τον χώρο και να προληφθεί κάποια έκρηξη ή κάποια μεγάλη διαρροή αερίου. Τέλος, το ρομπότ μπορεί να σταλεί σε κάποιον χώρο που πρέπει να γίνει μια συντήρηση και να αφεθεί εκεί για κάποιο χρονικό διάστημα ώστε να πάρουμε πληροφορίες για την κατάσταση του χώρου και αν είναι δυνατόν να γίνουν αυτές οι εργασίες.

Εν κατακλείδι, είναι φανερό ότι η ανάπτυξη ενός τέτοιου ρομπότ θα βοηθούσε στην αποφυγή πολλών εργατικών ατυχημάτων, στην πρόληψη αυτών αλλά και στην αποφυγή μεγάλων καταστροφών του βιομηχανικού χώρου και εξοπλισμού.

1.1 Internet of Things



Εικόνα 1: Εφαρμογές του Internet Of Things (www.betanews.com)

Ο όρος internet of things πρωτοεμφανίστηκε την δεκαετία του 1990 από τον Kevin Ashton. Ο όρος αυτός αναφέρεται σε ένα δίκτυο επικοινωνίας πληθώρας συσκευών, οικιακών συσκευών, αυτοκινήτων καθώς και κάθε αντικειμένου που ενσωματώνει ηλεκτρονικά μέσα, λογισμικό, αισθητήρες και συνδεσιμότητα σε δίκτυο ώστε να επιτρέπεται η σύνδεση και η ανταλλαγή δεδομένων. Απλούστερα, η φιλοσοφία του IoT είναι η σύνδεση όλων των ηλεκτρονικών συσκευών μεταξύ τους (τοπικό δίκτυο) ή με δυνατότητα σύνδεσης στο διαδίκτυο (παγκόσμιο ιστό).

Η έννοια "Things" (πράγματα) δεν είναι αυστηρά συνδεδεμένη με ορισμένα προϊόντα. Αναφέρεται σε μία ευρεία ποικιλία συσκευών εντελώς διαφορετικά μεταξύ τους, όπως για παράδειγμα αυτοκίνητα με ενσωματωμένους αισθητήρες, κάμερες, κλιματιστικά, φώτα, συστήματα ασφαλείας, “έξυπνα” ρολόγια (smartwatches) ακόμα και αυτοκίνητα των οποίων οι περίπλοκοι αισθητήρες εντοπίζουν αντικείμενα στην πορεία τους. Είναι μερικά από τα πολλά προϊόντα τεχνολογίας. Βασικό χαρακτηριστικό όλων είναι η σύνδεση μεταξύ τους με απώτερο σκοπό την δυνατότητα του χρήστη να τα ελέγχει από έναν υπολογιστή ή κινητό.

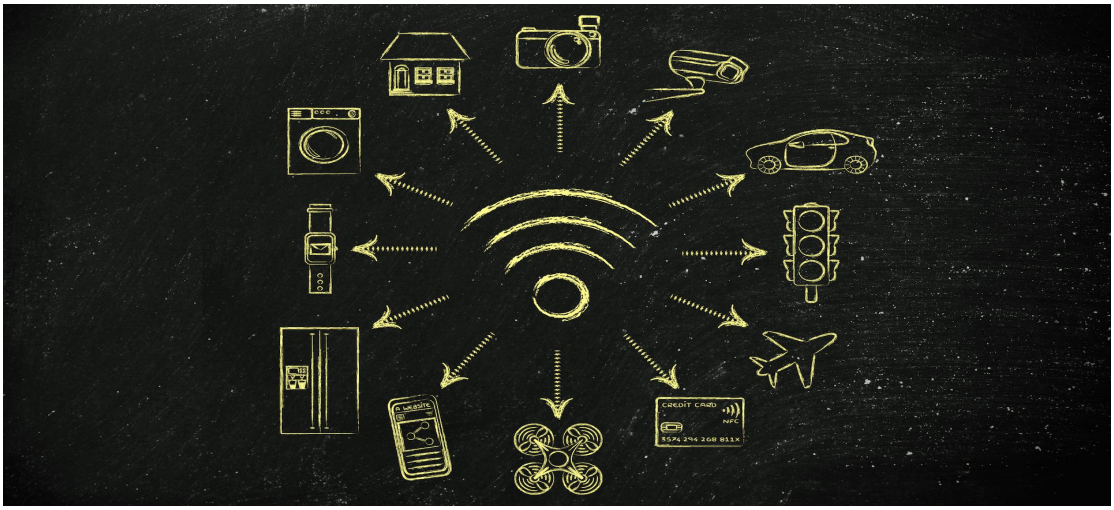
Η λειτουργία του είναι πολύ απλή μιας και συσκευές και αντικείμενα με ενσωματωμένους αισθητήρες συνδέονται με μια πλατφόρμα, η οποία περιλαμβάνει δεδομένα από τις διάφορες συσκευές και εφαρμόζει αναλυτικά στοιχεία για να μοιράζονται τις πιο πολύτιμες πληροφορίες με εφαρμογές που έχουν δημιουργηθεί για την αντιμετώπιση συγκεκριμένων αναγκών. Οι συσκευές IoT μπορούν να εντοπίσουν ακριβώς ποιες πληροφορίες είναι χρήσιμες και να τις εκμεταλλευτούν κατάλληλα. Η δυνατότητα αυτή μπορεί να αυτοματοποιήσει επαναλαμβανόμενες, χρονοβόρες ή ακόμα και επικίνδυνες εργασίες.

Μια μεγάλη πρόκληση είναι η αποθήκευση, η ανάλυση και η αξιοποίηση των δεδομένων που προέρχονται από όλες τις συνδεδεμένες συσκευές στο δίκτυο. Αυτός ο όγκος δεδομένων είναι τεράστιος και δεν μπορεί να αποθηκευτεί χωρίς να υποστεί κάποια διαλογή ή και επεξεργασία.

Το IoT μας ανοίγει την πόρτα της ψηφιακής εποχής και η χρησιμότητά του είναι μεγάλη αλλά η ζήτηση από τους υποψήφιους αγοραστές ακόμα μεγαλύτερη. Ως επί το πλείστον, οι άνθρωποι αποζητούν την αυτονομία σε πολλά πράγματα γύρω τους. Από ένα αυτόματο ξυπνητήρι μέχρι το έξυπνο ψυγείο που ενημερώνει το χρήστη για βασικές ελλείψεις ή ακόμα και την δυνατότητα ενεργοποίησης κλιματισμού πριν ακόμα ο χρήστης εισέλθει στο σπίτι. Είναι μερικές από τις δυνατότητες που προσφέρει το IoT. Η χρήση του δεν παραμένει μόνο εκεί, αλλά επεκτείνετε και στις επιχειρήσεις οι οποίες εκμεταλλεύονται την δυνατότητα αποθήκευσης και επεξεργασίας των δεδομένων από cloud συστήματα.

Παράδειγμα εφαρμογής IoT (πηγή Wikipedia)

Ο δήμος των ΗΠΑ που πέρασε σε έξυπνους μετρητές για την παρακολούθηση της χρήσης του νερού είδε άμεση και ασφαλή εξοικονόμηση χρημάτων. Η διαδικασία συλλογής δεδομένων εξελίχθηκε από μια χειρωνακτική διαδικασία, στην οποία οι τεχνικοί ταξίδευαν σε κάθε μετρητή, σε μία γρήγορη και τυποποιημένη εργασία από συστήματα IoT όπου οι αυτόματοι μετρητές κατέγραφαν και απέστελλαν σε μια κεντρική βάση δεδομένων τα αποτελέσματα των μετρήσεων. Αυτό εξοικονόμησε πολλά χρήματα, τόσο σε ώρες εργασίας όσο και σε εξοπλισμό, όπως φορτηγά. Η πόλη υπολογίζει συνολική εξοικονόμηση 28 εκατομμυρίων δολαρίων και καθαρή εξοικονόμηση περίπου 10 εκατομμυρίων δολαρίων κατά τη διάρκεια της πρωτοβουλίας.



Εικόνα 1.1: Παραδείγματα IoT με χρήση Wi-Fi.

2. ΥΛΙΚΟΣ ΕΞΟΠΛΙΣΜΟΣ

2.1 Ο κόσμος του Arduino

Το 2005 τέθηκε σε λειτουργία ένα πλάνο προκειμένου να κατασκευαστεί μία συσκευή για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από φοιτητές και μαθητές, το οποίο θα ήταν πιο οικονομικό από άλλα πρωτότυπα συστήματα, διαθέσιμα εκείνη την χρονική περίοδο. Το σχέδιο αυτό ονομάστηκε Arduin της Ivrea από τους δημιουργούς του Massimo Banzi και David Cueartielles. Έτσι, άρχισαν να κατασκευάζουν πλακέτες σε ένα μικρό εργοστάσιο στην βορειοδυτική Ιταλία, πιο συγκεκριμένα στην Ιβρέα, κωμόπολη της επαρχίας Τορίνο στην περιοχή Πεδεμόντιο στον ίδιο τόπο που στεγαζόταν και η επιχείρηση υπολογιστών Olivetti.

Όπως αναφέρουν οι ιδρυτές του, το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών, βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software. Απευθύνεται σε όλους όσους κατέχουν μια μικρή προγραμματιστική εμπειρία, βασικές γνώσεις ηλεκτρονικών και την θέληση να δημιουργήσουν διαδραστικά αντικείμενα ή περιβάλλοντα.

Ουσιαστικά, πρόκειται για ένα ηλεκτρονικό κύκλωμα που φέρει τον μικροελεγκτή ATmega της εταιρείας Atmel και του οποίου διακινούνται ελεύθερα και δωρεάν όλα τα σχέδια, καθώς και το software που απαιτείται για την λειτουργία του. Έτσι, μπορεί να κατασκευαστεί από οποιονδήποτε και αυτός είναι και ο λόγος για τον οποίο περιγράφετε με τον περίεργο – για hardware- χαρακτηρισμό: «ανοικτού κώδικα».

Ο Arduino, είναι ικανός να λειτουργήσει σαν ένας μικροσκοπικός υπολογιστής, επειδή ο χρήστης έχει την ικανότητα να συνδέσει πάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει τον μικροελεγκτή ώστε να δέχεται πληροφορίες από τις μονάδες εισόδου, να τις επεξεργάζεται και να δίνει τις κατάλληλες εντολές στις μονάδες εξόδου. Θα μπορούσε μάλιστα κάποιος να ισχυριστεί – και θα ήταν μια πετυχημένη δήλωση – ότι λειτουργικά το Arduino έχει πολλές ομοιότητες με το NXT Brick των Lego Mindstorms NXT. Εξάλλου, μια από τις πολλές εφαρμογές στις οποίες το Arduino διαπρέπει είναι η ρομποτική.

Το Arduino διαπιστωμένα, δεν είναι ο μοναδικός, άλλα ούτε και ο καλύτερος δυνατός τρόπος για την υλοποίηση μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Το κυριότερο πλεονέκτημά του είναι η μεγάλη κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια αντίστοιχη σε μέγεθος online γνωσιακή βάση. Αυτός είναι τελικά και ο παράγοντας ο οποίος οδήγησε στην ευρεία χρήση του Arduino διότι, δεν χρειάζονται πολλές (ηλεκτρονικές) γνώσεις πέρα από τις λίγες που έχει μάθει ο καθένας μας στο σχολείο.

Το Arduino είναι *μικροελεγκτής* μονής πλακέτας. Αυτό σημαίνει ότι αποτελείται από μια απλή *μητρική πλακέτα ανοικτού κώδικα* και έχει έναν ενσωματωμένο μικροελεγκτή, καθώς και εισόδους/εξόδους. Η πλακέτα αυτή μπορεί να προγραμματιστεί με τη γλώσσα Wiring η οποία αποτελείται από την πολύ γνωστή γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες της. Έτσι μας δίνεται η δυνατότητα να αναπτύξουμε ανεξάρτητα διαδραστικά αντικείμενα, αλλά και να συνδέσουμε το Arduino σε υπολογιστή, με την βοήθεια ποικίλων λογισμικών, όπως Processing, Max/MSP, PureData, SuperCollider. Κυκλοφορούν προ-συναρμολογημένες εκδόσεις του Arduino (έτοιμα συστήματα), ενώ για τους περισσότερο τολμηρούς χρήστες είναι διαθέσιμα διαγράμματα και οδηγίες συναρμολόγησης. (grhotels.gr)

Μετά την ευρεία χρήση και την αποδοχή του από ένα τεράστιο πλήθος χρηστών, τα Arduino άρχισαν να βγαίνουν και σε διάφορα μεγέθη και δυνατότητες. Με την πάροδο του χρόνου και τα προβλήματα της κάθε εφαρμογής οδηγηθήκαμε στο Arduino Nano, μια μικρή πλακέτα που είναι ιδανική για μικρές εφαρμογές που απαιτούν και μικρό όγκο. Επίσης, προέκυψε και το Arduino Mega, ένας επεξεργαστής με μεγαλύτερη ταχύτητα από τους προκατόχους του αλλά και μεγαλύτερο όγκο. Ο Mega έχει πολλές επιπλέον εισόδους-εξόδους αλλά και λειτουργίες και είναι ιδανικός σε μεγαλύτερες εφαρμογές όπου έχουμε μεγάλο όγκο δεδομένων προς επεξεργασίας αλλά και σε εφαρμογές που απαιτούν μεγαλύτερη ισχύ.

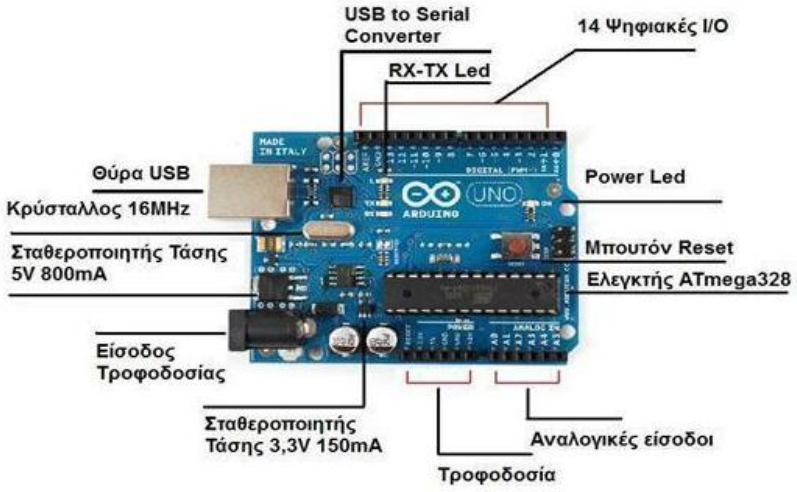
2.1.1 Εξοπλισμός του Arduino.

Όσο αφορά την υλική του δομή, το σύστημα Arduino περιλαμβάνει έναν γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz, ενώ κάποιες παραλλαγές έχουν κεραμικό αντηχητή. Για να μην απαιτείται εξωτερικός προγραμματιστής ο μικροελεγκτής φέρει ενσωματωμένο έναν φορτωτή εκκίνησης (bootloader).

Στην ακολουθία λογισμικού (software stack) που χρησιμοποιείται στις πλατφόρμες Arduino όλες οι πλάκες (boards) προγραμματίζονται με μία RS-232 σειριακή σύνδεση αλλά για κάθε διαφορετική εκδοχή υλικού εξοπλισμού η σύνδεση αυτή διαφέρει. Όσον αφορά τις σειριακές πλάκες, αυτές περιέχουν ένα απλό κύκλωμα μετατόπισης στάθμης (level shifter) που μετατρέπει το σήμα τάσης από RS-232 σε TTL, και το αντίστροφο. Πλέον τα Arduino προγραμματίζονται μέσω USB χρησιμοποιώντας εφαρμογές προσαρμογής chip USB-to-Serial, όπως το FTDIFT232. Όσον αφορά το ανεπίσημο Boarduino ή το Arduino mini

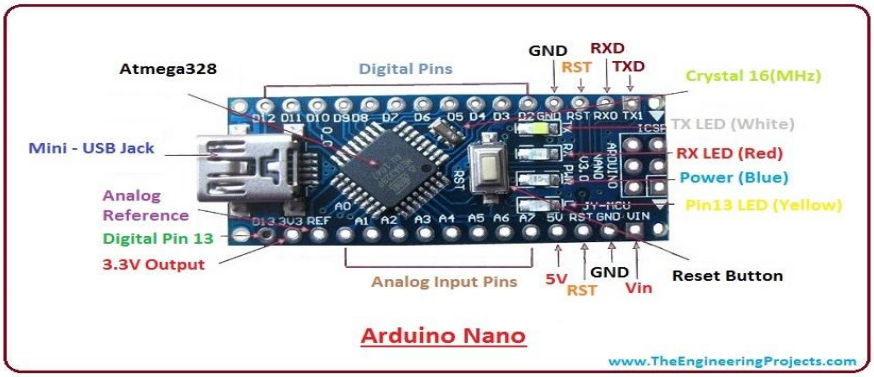
εφαρμόζονται άλλες μέθοδοι αλλά και ένα αφαιρούμενο USB-to-Serial καλώδιο ή πλάκα, ή ακόμα και σύνδεση με Bluetooth.

Σε σχέση με άλλα κυκλώματα, οι πίνακες Arduino περιέχουν περισσότερα microcontroller I/O pins, με διάφορες εφαρμογές plug-in ασπιδών να είναι διαθέσιμες. Τα Arduino Diecimila, Duemilanove και Uno έχουν 14 ψηφιακά I/O pins, έξι εκ των οποίων παράγουν pulse-width διαμορφωμένα σήματα, και έξι αναλογικά δεδομένα. Αυτά τα pins μπορούμε να τα βρούμε στην κορυφή του πίνακα με femaleheaders 0.1 ιντσών (2,2mm). Για την σύνδεση σε Bread boards τα Arduino Nano, Arduino-CompatibleBareBonesBoard και Boarduino Board, παρέχουν κάποιες φορές maleheader pins.



Εικόνα 2.1: hardware Arduino (myduino.com)

Εναλλακτικά του Arduino, υπάρχουν πολλοί πίνακες (boards) που είναι είτε συμβατά είτε προερχόμενα από αυτό. Η συνηθέστερη μετατροπή των Arduino είναι η προσθήκη καινοτόμων οδηγών εξόδων (output drivers) στα πλαίσια χρήσης τους στην εκπαίδευση, ώστε να απλοποιηθούν και να γίνουν περισσότερο προσιτές στους μαθητές κατασκευές buggies ή μικρών ρομπότ. Κάποιες από αυτές τις μετατροπές είναι ισάξιες αλλάζοντας μόνο τον παράγοντα μορφής κι έτσι επιτρέποντας την συνεχόμενη χρήση ασπίδων (shields). Γενικά υπάρχουν πολλές παραλλαγές που χρησιμοποιούν πληθώρα επεξεργαστών και ποικίλα επίπεδα συμβατότητας.



Εικόνα 2.2: Arduino Nano (theEngineeringProjects.com)

Arduino Mega 2560

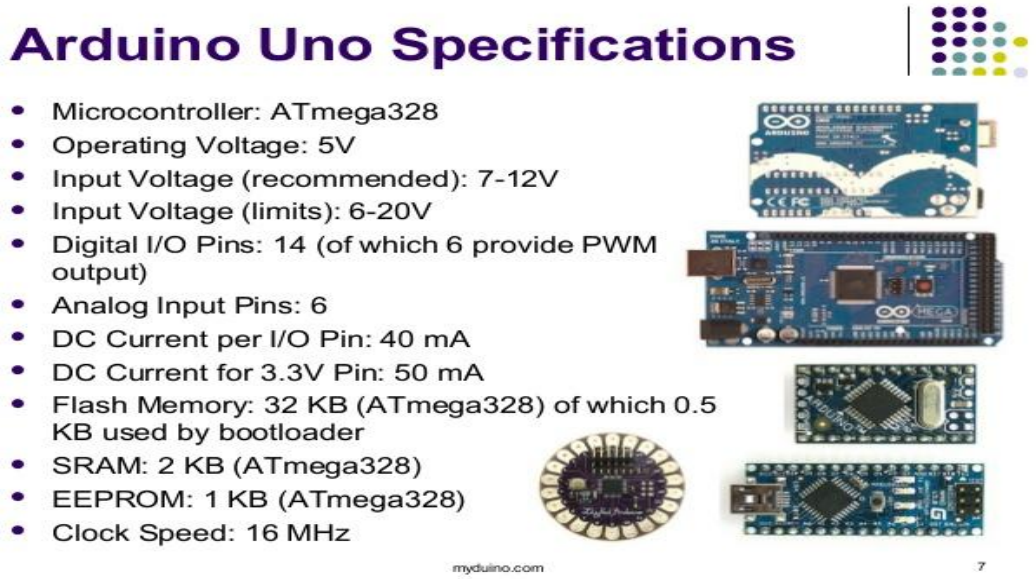
Specifications

- Atmega 2560 Microcontroller
- 54 I/O Pins (15 PWM O/p)
- 16 Analog I/P Pins
- 7-12V Input Voltage
- 5V Operating Voltage
- 256KB of Flash Memory
- 8KB SRam
- 4KB EEPROM
- 16MHz Clock Speed
- 2 8-Bit Timer Counters
- RTC with separate Oscialltor
- SPI/I2C Interface
- Serial / RxTx Interface
- ICSP Communication

About

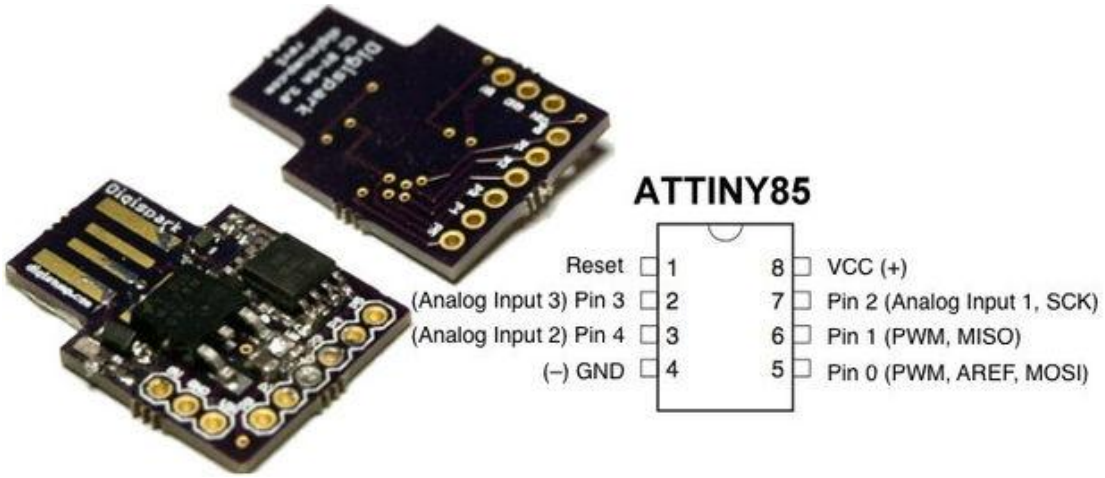
Arduino Mega is a Open source microcontroller board with AtMega2560 Microcontroller it has been manufactured with a main focus on internet of things and industrial automation. It can be programmed with Arduino IDE provided by the manufacturer. The read/write operation is achieved through Serial Communication(USB)

Εικόνα 2.3: Χαρακτηριστικά του Arduino Mega 2560 (myduino.com).



Εικόνα 2.4: Χαρακτηριστικά Arduino Uno (myduino.com).

Τέλος, θα αναφέρουμε τη νεότερη έκδοση του Arduino, που είναι το ATTiny85. Ένα Arduino απίστευτα μικρό:



Εικόνα 2.5: Arduino ATTINY85 (Wikipedia.com)

2.1.2 Πλεονεκτήματα χρήσης Arduino

Υπάρχουν πολλά συστήματα μικροελεγκτών και μικροεπεξεργαστών που προσφέρουν την δυνατότητα παρόμοιων λειτουργιών με το Arduino. Τέτοια είναι τα ParallelBasic, Stamp, Netmedia's BX-24, Phidgets και MIT's Handyboard. Οι ομοιότητα τους έγκειται στην απλοποίηση του χειρισμού των μικροελεγκτών μέσα από ένα χρηστικά εύκολο περιβάλλον λειτουργίας. Το Arduino όμως υπερέχει καθώς είναι εύχρηστο για μια πληθώρα ομάδων (καθηγητές, μαθητές, ενδιαφερόμενοι ερασιτέχνες) λόγω των εξής στοιχείων του:

Χαμηλό κόστος. Ένα πλήρες συναρμολογημένο Arduino κοστίζει λιγότερο από 15 ευρώ, μία τιμή που χαμηλώνει ακόμα περισσότερο αν συναρμολογηθεί από τον χρήστη (με το χέρι). Οι πλακέτες του είναι συγκριτικά με άλλες πλατφόρμες πιο φθηνές.

Γενική πλατφόρμα. Ενώ οι περισσότεροι μικροελεγκτές χρειάζονται λειτουργικό σύστημα (OperatingSystem, OS) Windows για να λειτουργήσουν, το Arduino είναι γνωστό πως δουλεύει σε Windows και Linux.

Απλό, κατανοητό προγραμματιστικό περιβάλλον. Ενώ είναι εύκολο στη χρήση για τους αρχάριους, το διακρίνει ένας ευέλικτος χαρακτήρας από τον οποίο μπορούν να επωφελούνται οι προχωρημένοι χρήστες. Η συγγένειά του με το περιβάλλον Processing το κάνει ιδανικό για καθηγητές και οικείο για τους μαθητές.

Εργαλείο Ανοιχτού κώδικα και επεκτάσιμου λογισμικού. Οι βιβλιοθήκες της C++ είναι πολύ βοηθητικό εργαλείο για την επέκταση του λογισμικού Arduino, σε συνάρτηση με τον ανοιχτό κώδικα που είναι διαθέσιμος προς εξέλιξη στους πιο έμπειρους προγραμματιστές. Ακόμη, η στενή συγγένεια με τις βιβλιοθήκες της C++ προσφέρει εύκολη μετάβαση και κατανόηση και για τα άτομα που ενδιαφέρονται από Arduino να μεταβούν στην AVR C γλώσσα προγραμματισμού στην οποία είναι βασισμένο. Αυτός είναι και ο λόγος που μπορεί κάποιος να προσθέσει απευθείας AVR-C κώδικα σε Arduino χωρίς πρόβλημα.

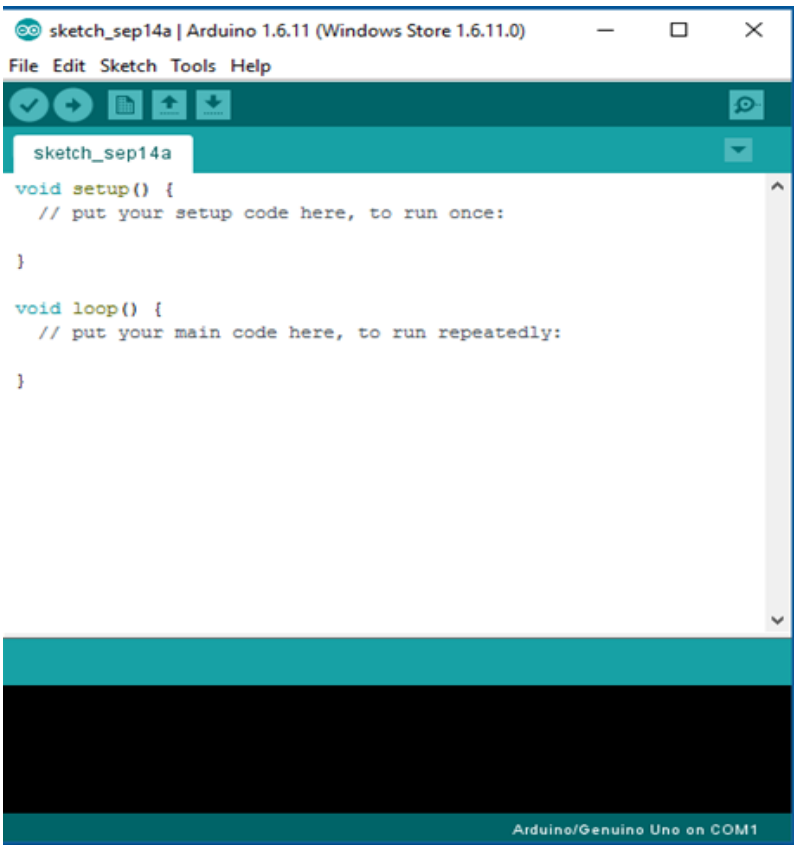
Εργαλείο Ανοιχτού κώδικα και επεκτάσιμου υλικού. Επιπλέον τα λειτουργικά σχέδια του λογισμικού Arduino είναι δημοσιευμένα υπό Δημιουργική Κοινή Άδεια (Creative Common License), Ταυτόχρονα είναι βασισμένο στους μικροελεγκτές ATMEGA8 και ATMEGA168 της Atmel, είναι εύκολο να τροποποιηθεί. Φυσικά μόνον οι έμπειροι σχεδιαστές και κατασκευαστές κυκλωμάτων μπορούν να δημιουργήσουν τροποποιημένες και βελτιωμένες εκδόσεις λειτουργικών τμημάτων του Arduino. (http://brain.ee.auth.gr)

Η συναρμολόγηση της μητρικής πλακέτας όμως, και η κατανόηση της βασικής δομής και λειτουργίας του, είναι πολύ εύκολη ενώ ταυτόχρονα μειώνει το κόστος του.

2.1.3 Arduino και προγραμματισμός.

Τα Arduino χρησιμοποιούν ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που έχει γραφτεί σε Java και είναι έτσι σχεδιασμένο ώστε να εισαγάγει στον προγραμματισμό άτομα που δεν είναι εξοικειωμένοι με το αντικείμενο όπως νέοι μαθητές. Το IDE αυτό λειτουργεί σε πολλές πλατφόρμες και προέρχεται από τον προγραμματισμό με την γλώσσα Processing πάνω στο περιβάλλον Wiring. Αποτελείται από ένα τύπο επεξεργασίας κώδικα που τον χαρακτηρίζει η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και είναι σε θέση να μεταγλωττίζει και να φορτώνει προγράμματα στην πλακέτα με το πάτημα ενός κουμπιού. Τα προγράμματα ή ο ίδιος ο κώδικας που γράφτηκε για συστήματα Arduino ονομάζονται σκίτσο (sketch) ενώ στις περισσότερες περιπτώσεις δεν υπάρχει ανάγκη επεξεργασίας ή χρήσης περιβάλλοντος γραμμής εντολών.

Το Arduino IDE τρέχει προγράμματα που είναι συνήθως γραμμένα σε C ή C++, αλλά με ειδικούς κανόνες δομής. Μαζί με το Arduino IDE έχουμε και μια βιβλιοθήκη λογισμικού που προέρχεται από το πρότζεκτ Wiring, για αυτό και έχουν πολλές κοινές διαδικασίες εισόδου/εξόδου (input/output).



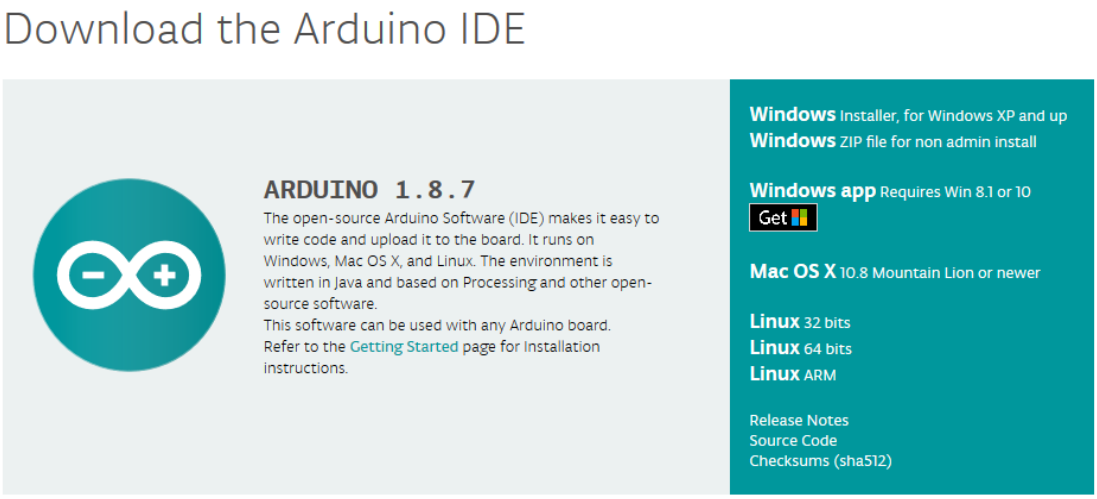
Οι χρήστες πρέπει να ορίσουν μόνο τις εξής δύο λειτουργίες για να κάνουν ένα πρόγραμμα κυκλικής εκτέλεσης:

-setup(): μία συνάρτηση που τρέχει μία φορά στην αρχή του προγράμματος και αρχικοποιεί τις ρυθμίσεις.

-loop(): μία συνάρτηση που καλείται συνέχεια μέχρι η πλακέτα να απενεργοποιηθεί.

Η βασική ρουτίνα setup() εκτελείται μια φορά μόνο κατά την εκκίνηση του προγράμματος ενώ η βασική ρουτίνα loop() περιέχει τον βασικό κορμό του προγράμματος και η εκτέλεσή της επαναλαμβάνεται συνέχεια σαν ένας βρόγχος while(true). (python.org.gr)

Σε αυτό το σημείο να αναφερθεί ότι το περιβάλλον προγραμματισμού του Arduino κυκλοφορεί ελεύθερο στο διαδίκτυο και μπορεί ο καθένας να το βρει για διαφορετικά λειτουργικά συστήματα:



Εικόνα 3.6: Σύνδεσμος λήψης λογισμικού (Arduino.cc).

Τέλος, αξίζει να σημειωθεί ότι υπάρχουν πολλά έτοιμα προγράμματα για τα Arduino, όπως και βιβλιοθήκες για άπειρες λειτουργίες, για τις οποίες υπάρχει η δυνατότητα επεξεργασίας και αναβάθμισης με αποτέλεσμα προσαρμογής στην κάθε εφαρμογή. Επιπλέον υπάρχουν πολλών ειδών οδηγοί εκμάθησης της γλώσσας προγραμματισμού, από βιβλία που κυκλοφορούν ελεύθερα στο διαδίκτυο μέχρι αναλυτικά βίντεο με εφαρμογές. Και όλα αυτά μπορούν να φανερωθούν με μια απλή αναζήτηση στο Google την λέξη Arduino.

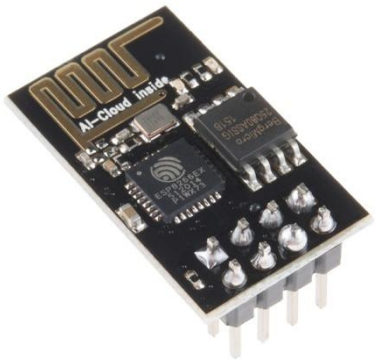
2.2 ESP 8266 Wi-Fi Module

Το ESP8266 είναι ένα χαμηλού κόστους Wi-Fi chip με ενσωματωμένα πρωτοκολλά TCP/IP το οποίο μπορεί να δώσει πρόσβαση στο Wi-Fi σε οποιοδήποτε μικροελεγκτή. Το ESP μπορεί να τρέξει το ίδιο μια εφαρμογή ή να φορτώσει μια, από άλλη συσκευή.

Κατασκευάζεται στη Σαγκάη από την εταιρία Espressif. Πρόκειται για ένα αρκετά σύγχρονο τσιπ, το οποίο εμφανίστηκε στη δυτική αγορά τον Αύγουστο του 2014, με το μοντέλο ESP-01 από την Al-Thinker. Είναι η πρώτη οικονομική μονάδα που επιτρέπει τη σύνδεση στο Wi-Fi αν και αρχικά δεν υποστήριζε την αγγλική γλώσσα. Η χαμηλή τιμή του και το μικρό μέγεθος του προσέλκυσε το ενδιαφέρον hacker , οι οποίοι τελικά το μετέφρασαν από την κινεζική στην αγγλική γλώσσα.

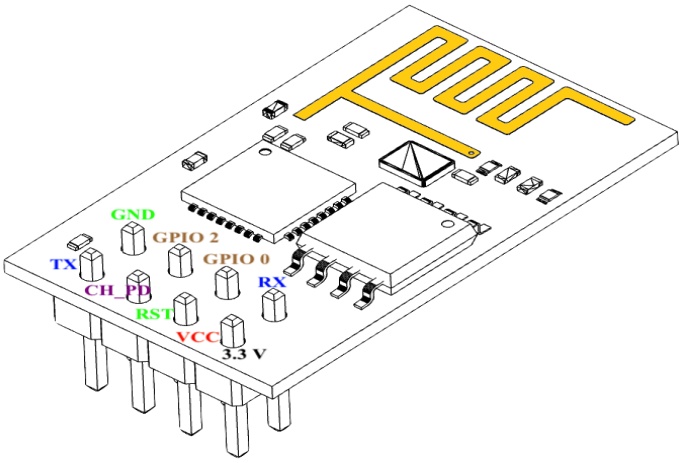
Έκτοτε, ολοένα και περισσότεροι προγραμματιστές, το χρησιμοποιούν σε ποικίλες εφαρμογές. Η ικανότητα για γρήγορη επεξεργασία και ο μεγάλος αποθηκευτικός χώρος (1MB), επιτρέπουν την διασύνδεση με πολλούς αισθητήρες και άλλες συσκευές εφαρμογών (πχ Arduino), μέσω των GPIOs, με ελάχιστο χρόνο φόρτωσης κατά τη διάρκεια λειτουργιάς.

Παράλληλα, το κύκλωμα του είναι σχεδιασμένο, ώστε να καταλαμβάνει τον ελάχιστο δυνατό χώρο πάνω σε μια πλακέτα PCB (gmelectronic.com).



Εικόνα 2.2.1: Πλακέτα ESP8266 (Wikipedia).

Υπάρχει πληθώρα πληροφοριών τόσο για τις δυνατότητες και τον τρόπο διασύνδεσης του με άλλες συσκευές και αισθητήρες, όσο και για τον προγραμματισμό του, οι οποίες παρέχονται κυρίως από την κοινότητα ανθρώπων που το χρησιμοποιούν. Στην κοινότητα μπορεί κανείς να διαβάσει πως μπορεί το ESP να μετατραπεί σε μια IoT συσκευή.



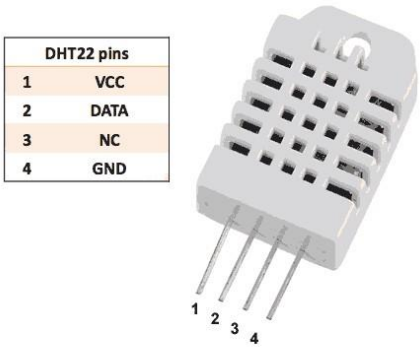
Εικόνα 2.2.2: Ακροδέκτες της πλακέτα ESP 8266 (Wikipedia).

Οι ακροδέκτες της συσκευής περιγράφονται παρακάτω:

1. VCC: τάση εισόδου (3,3V)
2. GND: γείωση (0 V)
3. RX: αποδοχή δεδομένου (data bit X)
4. TX: μετάδοση δεδομένου (data bit X)
5. CH_PD: απενεργοποίηση συσκευής
6. RST: επαναφορά (Reset)
7. GPIO 0: General-purpose input/output No. 0
8. GPIO 2: General-purpose input/output No. 2

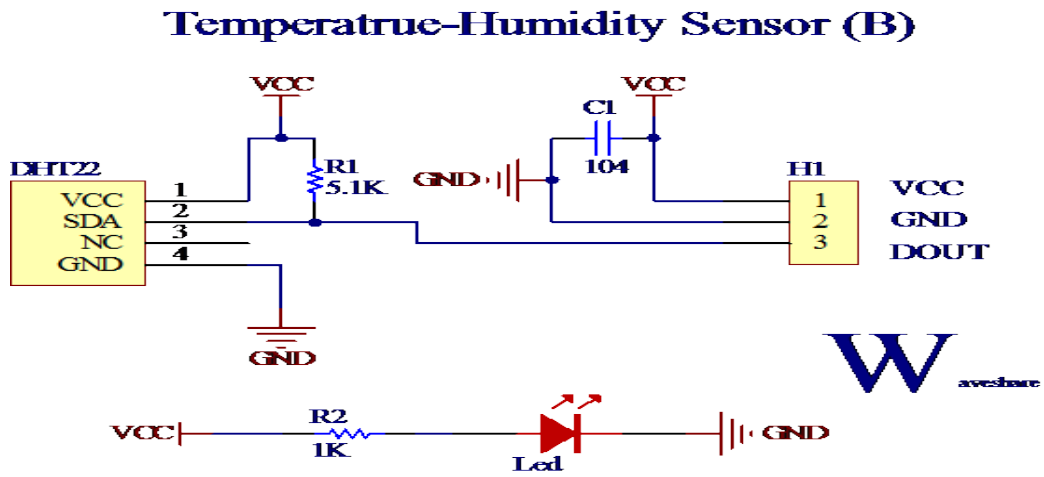
2.3 DHT-22

Το DHT-22 είναι ένα αισθητήριο θερμοκρασίας και υγρασίας, ευρείας χρήσης λόγω χαμηλού κόστους απόκτησης και χαμηλής κατανάλωσης. Επίσης η συνδεσμολογία του είναι πάρα πολύ απλή έχοντας μόνον τέσσερα pins και η ακρίβεια το πολύ αξιόπιστη. Αυτά κάνουν το συγκεκριμένο shield πολύ δημοφιλές και πρώτο στις προτιμήσεις των χρηστών.



Εικόνα 2.3.1: ακροδέκτες του DHT-22 (waveshare.com)

Ένα απλό παράδειγμα χρήσης του φαίνεται στην παρακάτω εικόνα:



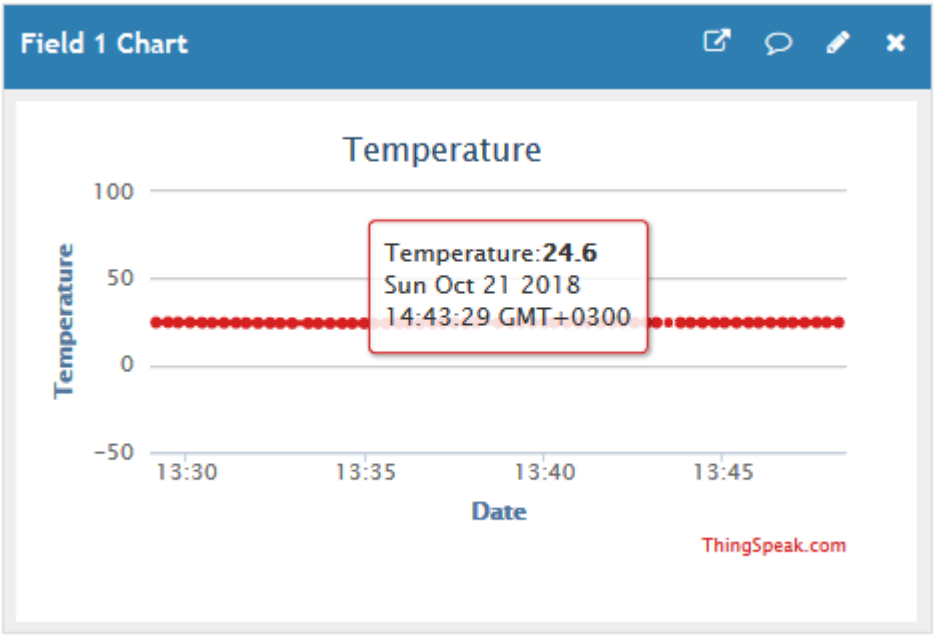
Εικόνα 2.3.2: παράδειγμα σύνδεσης DHT-22 (waveshare.com)

Τεχνικά χαρακτηριστικά του αισθητηρίου μας φαίνονται παρακάτω:

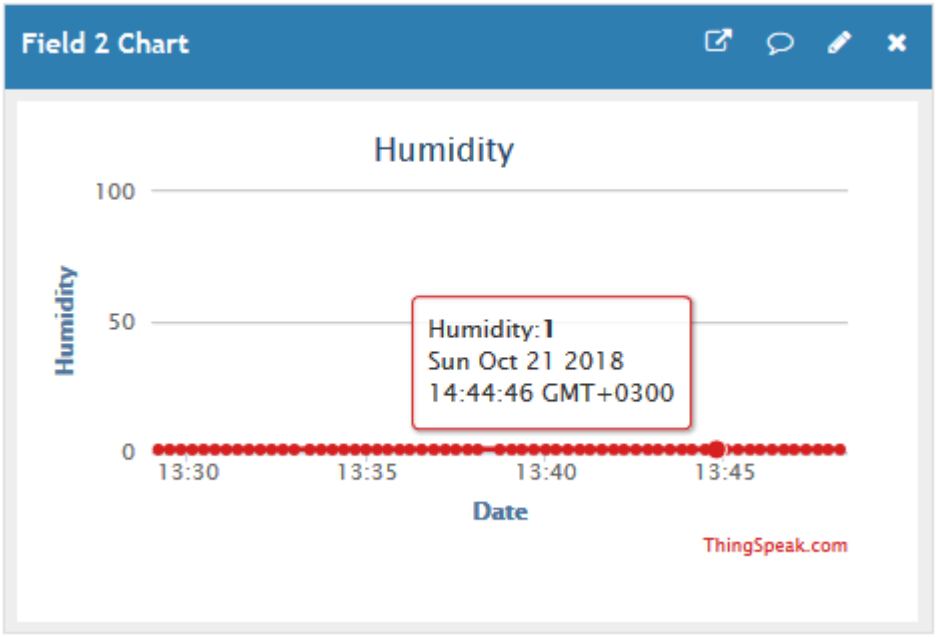
- 3.3-6V τάση εισόδου
- 1-1.5mA μετρούμενο ρεύμα
- 40-50 μA κατανάλωση σε λειτουργία αναμονής
- Μετρούμενη υγρασία 0-100% RH
- Μετρούμενη θερμοκρασία (-40 – 80) βαθμούς κελσίου
- +-2% Ακρίβεια υγρασίας.
- +-0.5 Ακρίβεια θερμοκρασίας.

Φυσικά υπάρχουν και άλλες εκδόσεις του DHT, όπως το DHT-11 το οποίο είναι πιο φθηνό από το DHT-22 αν και τα δυο αισθητήρια είναι ίδια στην χρήση τους, έχουν μια διαφορά στην ακρίβεια μέτρησης όπου το DHT-11 δίνει μόνο ακέραιες τιμές.

Στις παρακάτω εικόνες φαίνονται τα δεδομένα που έστειλε το DHT-22 στο ThingSpeak:



Εικόνα 2.3.3: Δείγματα θερμοκρασίας μαζί με ημερομηνία και ώρα.



Εικόνα 2.3.4: Δείγματα υγρασίας με ημερομηνία και ώρα.

2.4 MQ-2 Gas Sensor

Τα αισθητήρια MQ είναι μια οικογένεια αισθητηρίων που ανιχνεύουν οσμές, αέρια-καπνούς. Ποικίλουν ανάλογα την περίπτωση. Και αυτά, όπως σχεδόν κάθε Arduino shield είναι φθηνό, εύκολο στη χρήση και υποστηρίζεται από πολλές ελεύθερες βιβλιοθήκες και παραδείγματα.

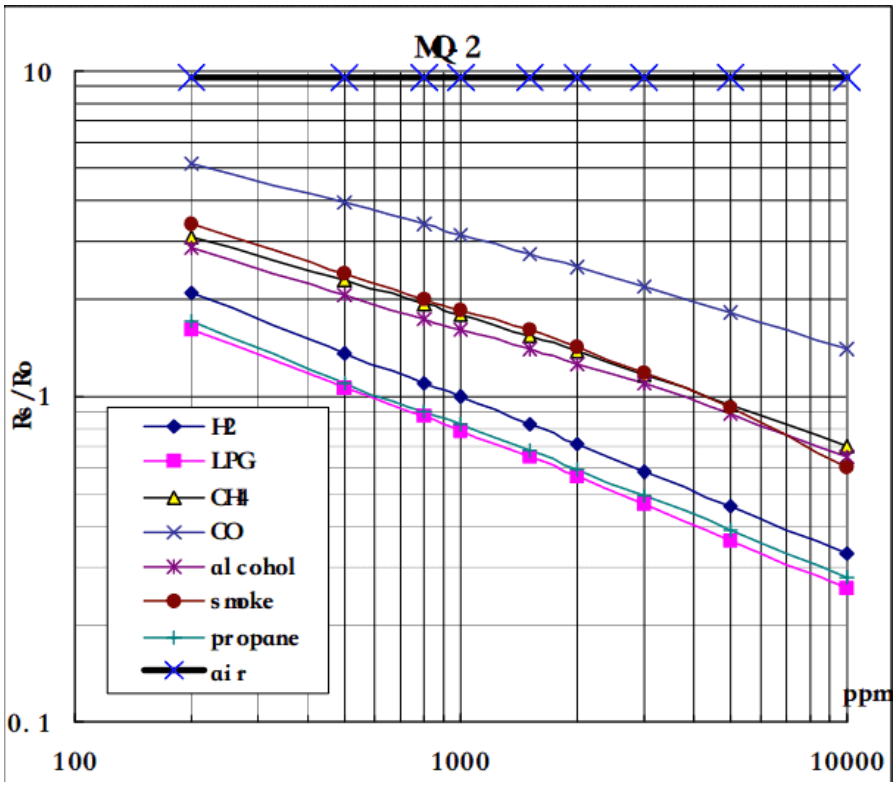
Κάποια από τα αισθητήρια της οικογένειας MQ φαίνονται στην παρακάτω εικόνα, μαζί με τις ιδιότητες τους:

Sensor	Gas Type
MQ2	Combustible Gas, Smoke
MQ3	Alcohol Vapor
MQ5	LPG, Natural Gas, Town Gas
MQ9	Carbon Monoxide, Coal Gas, Liquefied Gas

Εικόνα 2.4.1: Οικογένεια αισθητηρίων MQ

Το MQ-2 που χρησιμοποιήθηκε σε αυτήν την εφαρμογή είναι ένα αισθητήριο που ανιχνεύει καπνό. Υπάρχουν άλλα αισθητήρια που ανιχνεύουν φυσικό αέριο, προπάνιο, αλκοόλ, μονοξείδιο του άνθρακα κ.α.

Στην παρακάτω εικόνα φαίνονται τεχνικά χαρακτηριστικά των αισθητηρίων αυτών σε σχέση με το εκάστοτε αέριο:

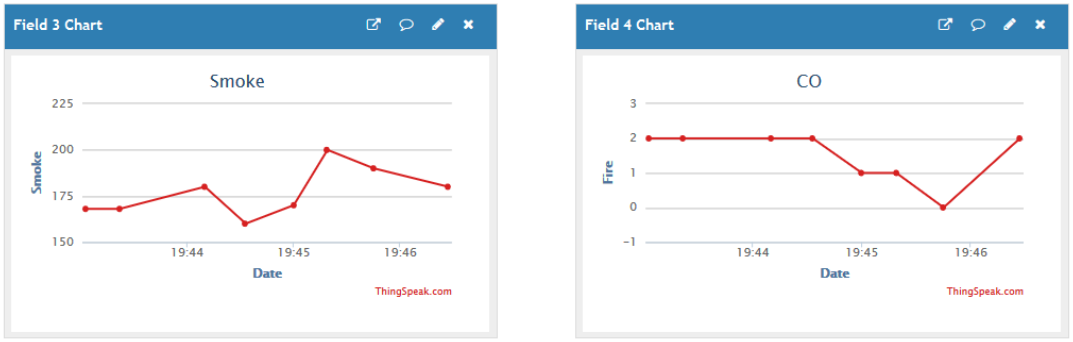


Εικόνα 2.4.2: Γραφική απεικόνιση (myduino.com)

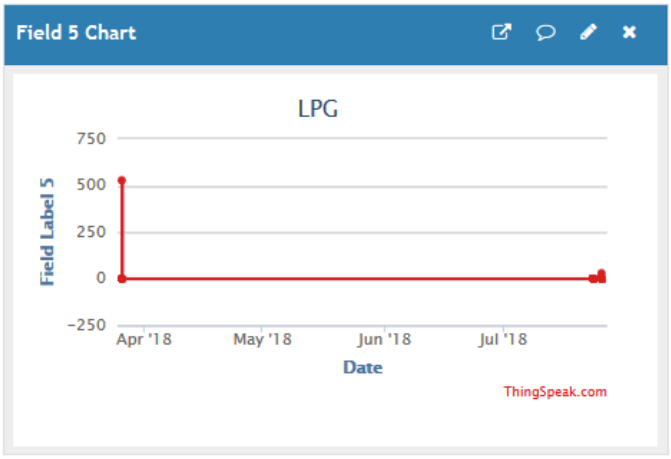
Η συνδεσμολογία του αισθητηρίου είναι πολύ απλή:

- VCC: 5V
- GND: Γείωση
- Dout: Ψηφιακή έξοδος
- Aout: Αναλογική έξοδος

Στις παρακάτω εικόνες φαίνονται δείγματα από τις τιμές που έστειλε το MQ-2 στο ThingSpeak:



Εικόνα 2.4.3: Δείγματα καπνού και μονοξειδίου του άνθρακα.

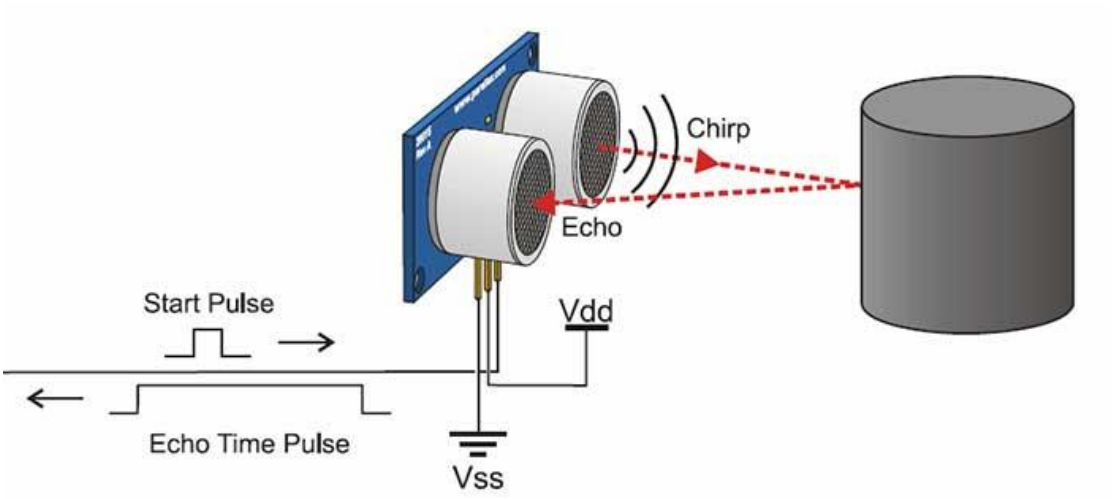


Εικόνα 2.4.4: Μετρήσεις αερίου (κίνησης).

2.5 UltraSonic Sensor

Το UltraSonic Sensor είναι ένα ευρέως γνωστό αισθητήριο ανίχνευσης εμποδίων με χαμηλό κόστος, εύκολο στη χρήση και στην τοποθέτηση αλλά και στην λειτουργία του.

Το αισθητήριο αυτό διαθέτει δυο «μάτια» τα όποια είναι και ολόκληρη η φιλοσοφία της μέτρησης του. Από το ένα «μάτι» εκπέμπει υπερηχητικά κύματα και με το άλλο «μάτι» τα λαμβάνει. Αυτή είναι μια ικανότητα που έχουν κάποια θηλαστικά ζώα της θάλασσας όπως οι φάλαινες και τα δελφίνια.



Εικόνα 2.5.1: Φιλοσοφία του UltraSonic (Wikipedia)

Για την σύνδεση του σε κάποια εφαρμογή το συγκεκριμένο αισθητήριο διαθέτει μόνον 4 Pins:

- Vcc: 5V
- GND: Γείωση
- Trig: Αποστολή υπερηχητικών κυμάτων.
- Echo: Λήψη των επιστρεφόμενων κυμάτων.

Εικόνα 2.5.2: Pin του UltraSonic (Wikipedia).

2.5.1 Δοκιμή ακρίβειας UltraSonic.

Μετά την τοποθέτηση και την σύνδεση του αισθητηρίου απόστασης, έπρεπε να ελεγχθεί η λειτουργία του και η ακρίβεια του ώστε να ρυθμιστεί κατάλληλο για το καλύτερο δυνατό αποτέλεσμα. Για την δοκιμή αυτή χρησιμοποιήθηκε η βιβλιοθήκη *<NewPing.h>,* η οποία μας δίνει την δυνατότητα ελέγχου του αισθητηρίου απόστασης. Κάποιες εντολές που μας παρέχει η συγκεκριμένη βιβλιοθήκη και είναι χρήσιμες σε αυτήν την εργασία φαίνονται παρακάτω:

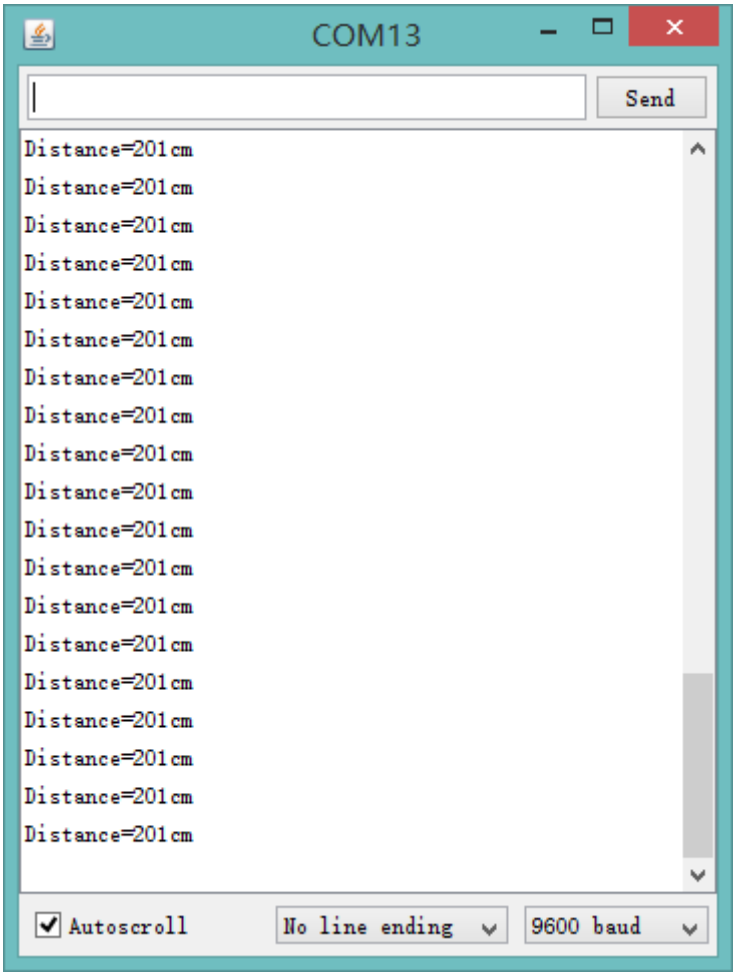
- sonar.ping(): Στέλνει μια παλμοσειρά και επιστρέφει την τιμή της απόστασης σε microseconds.
- sonar.ping_in(): Στέλνει μια παλμοσειρά και επιστρέφει την τιμή της απόστασης σε ίντσες.
- sonar.ping_cm(): Στέλνει μια παλμοσειρά και επιστρέφει την τιμή της απόστασης σε εκατοστά.

Περισσότερες εντολές και πληροφορίες περί αυτών μπορούν να βρεθούν στην σελίδα: *https://playground.arduino.cc/Code/NewPing*

Για την δοκιμή αυτή, πρέπει να ενεργοποιήσουμε την σειριακή επικοινωνία του Arduino με με ρυθμό μετάδοσης 115200 baud. Αυτό γίνεται πληκτρολογώντας την παρακάτω εντολή:

Serial.begin(115200); // open serial monitor at 115200 baud to see ping results.

Παράδειγμα αποστολής δεδομένων από το UltraSonic Sensor:



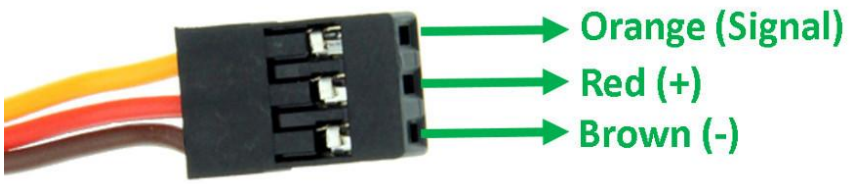
Εικόνα 2.5.3: Δείγματα του UltraSonic στραμμένο προς το ταβάνι.

2.6 Servo Motors

Σε αυτήν την εφαρμογή έχει χρησιμοποιηθεί για την κίνηση του ρομπότ σερβοκινητήρας. Για την ακρίβεια έχουν λάβει μέρος τέσσερεις σερβοκινητήρες.

Όσο απλή είναι η λειτουργία τους, τόσο καθοριστικό ρόλο παίζει στην κίνηση του ρομπότ, όπου θέλουμε να ελέγχουμε στροφές με βάση συγκεκριμένες μοίρες αλλά να γνωρίζουμε και την θέση που έχει ο σερβοκινητήρας εκείνη τη στιγμή.

Η τοποθέτηση του σερβοκινητήρα στην εφαρμογή μας ήταν απλή:



Εικόνα 2.6.1: Σύνδεση σερβοκινητήρα.

- (+): 5V
- (-): GND
- Signal: Σήμα ελέγχου από τον μικροελεγκτή.

Τεχνικά χαρακτηριστικά:

- Operating Speed:
  - 0.12sec/60degree (4.8V)
  - 0.10sec/60degree (6V)
- Stall Torque:
  - 1.3kg.cm/18.09oz.in(4.8V)
  - 1.5kg.cm/20.86oz.in(6V)
- Operating Voltage: 4.8V~6V
- Control System: Analog
- Direction: CCW
- Operating Angle: 120degree
- Required Pulse: 900us-2100us
- Bearing Type: None
- Gear Type: Plastic
- Motor Type: Metal

2.6.1 Δοκιμή λειτουργίας Servo Motors

Αφού τοποθετήθηκαν στην θέση τους οι κινητήρες και συνδέθηκαν κατάλληλα στην εφαρμογή, έπρεπε να γίνει μια δοκιμή λειτουργίας για να διαπιστωθεί ότι λειτουργούν σύμφωνα με τις προδιαγραφές και δεν έχουμε κάποια μεγάλη απόκλιση στις μοίρες κατά τον έλεγχο. Για την δοκιμή αυτή αλλά και τον μετέπειτα έλεγχο των κινητήρων χρησιμοποιήθηκε η βιβλιοθήκη *<Servo.h>*. Παρακάτω φαίνονται κάποιες εντολές που περιέχονται στην βιβλιοθήκη που χρησιμοποιήθηκαν σε αυτήν την εργασία:

- attach(int) - Turn a pin into a servo driver. Calls pinMode. Returns 0 on failure.
- detach() - Release a pin from servo driving.
- write(int) - Set the angle of the servo in degrees, 0 to 180.
- read() - return that value set with the last write().
- attached() - return 1 if the servo is currently attached.

Για περισσότερες εντολές *https://playground.arduino.cc/ComponentLib/Servo*

Αφού έγινε η εισαγωγή των βιβλιοθηκών στο πρόγραμμα, έγινε η δημιουργία 5 αντικειμένων για να μπορέσουμε να χρησιμοποιήσουμε τις κλάσεις των 2 βιβλιοθηκών (*<NewPing.h>* και *<Servo.h>),* όπως φαίνεται στην παρακάτω φωτογραφία.

```
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
NewPing sonar(TR\
```

Εν συνεχεία πρέπει να δηλώσουμε σε ποια έξοδο του Arduino αντιστοιχεί το κάθε σερβομοτέρ. Αυτό γίνεται εύκολα με τις παρακάτω εντολές:

```
void setup()
{
  myservo1.attach(9);    // Connect the signal wire of servo to pin 9
  myservo2.attach(10);    // Connect the signal wire of servo to pin 10
  myservo3.attach(11);    // Connect the signal wire of servo to pin 11
  myservo4.attach(12);    // Connect the signal wire of servo to pin 12
  Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
```

Η δοκιμή συνεχίζει πληκτρολογώντας την παρακάτω εντολή και για τα τέσσερα σέρβο ξεχωριστά:

myservo1.write();

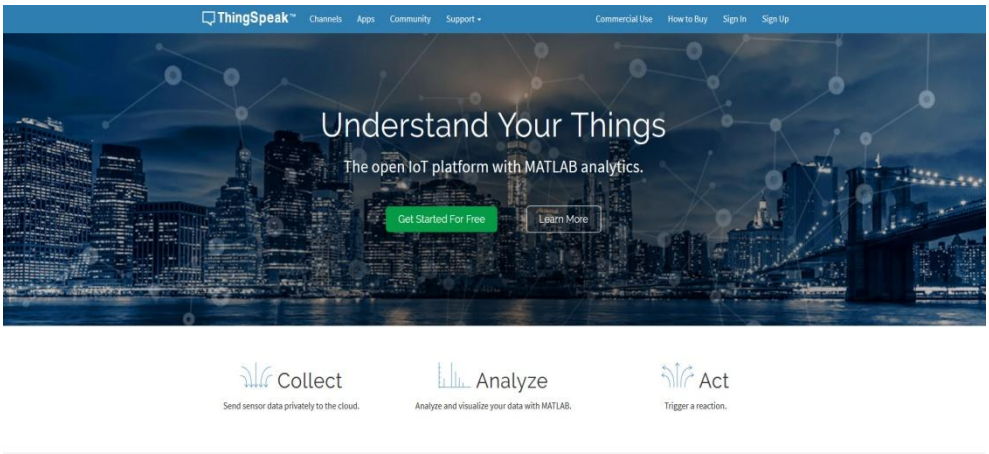
Με την οποία εντολή στέλνουμε τις μοίρες που επιθυμούμαι να κινηθεί το σέρβο.

Κατεβάζοντας το πρόγραμμα αυτό στο Arduino μπορούμε να παρατηρήσουμε με το μάτι την κίνηση των κινητήρων και αν αυτή η κίνηση ήταν η επιθυμητή.

3. ΛΟΓΙΣΜΙΚΟ

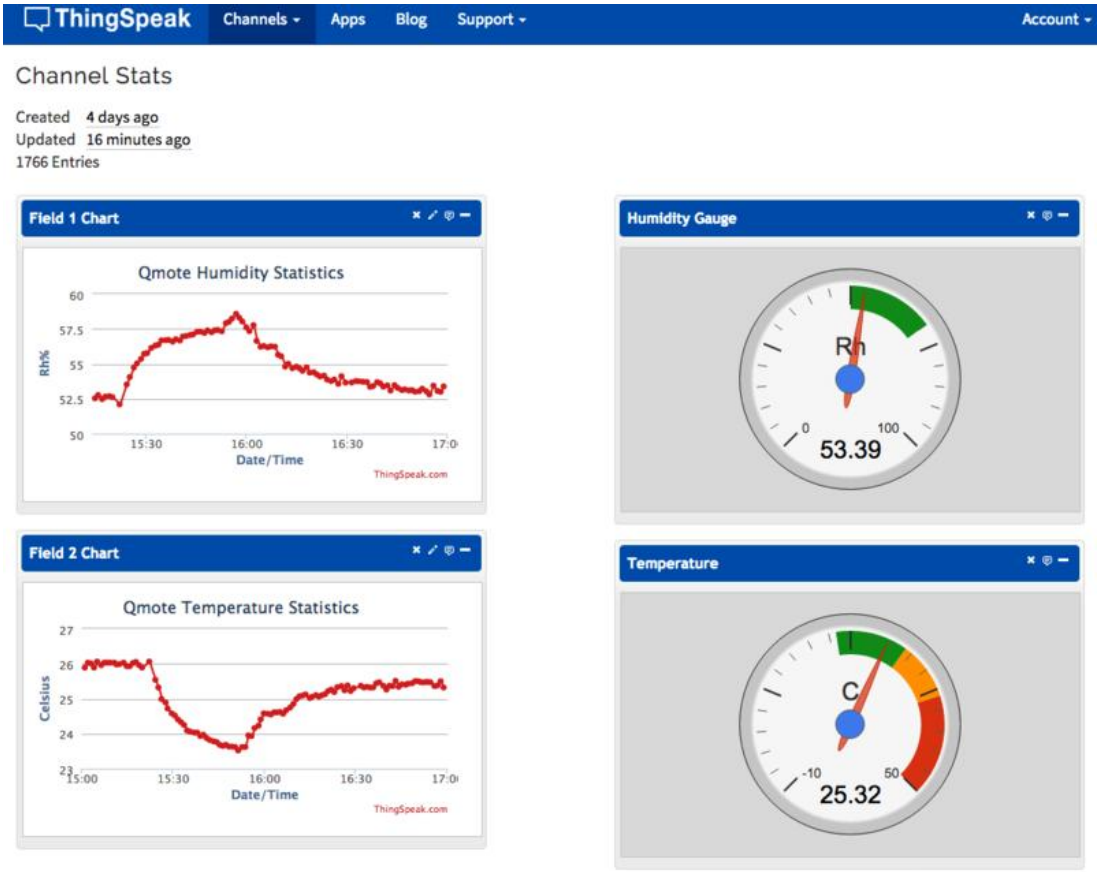
3.1 ThingSpeak

Το ThingSpeak είναι μια Internet of Things (IoT) πλατφόρμα που μας επιτρέπει να συλλέγουμε και από αποθηκεύουμε τα δεδομένα των αισθητηρίων μας σε ένα σύννεφο (cloud) και να δημιουργήσουμε IoT εφαρμογές. Η ThingSpeak™ IoT πλατφόρμα μας παρέχει εφαρμογές που μας επιτρέπουν να αναλύσουμε και να οπτικοποιήσουμε τα δεδομένα μας στο MATLAB®, και στην συνέχεια να ενεργήσουμε κατά βούληση πάνω σε αυτά. Τα δεδομένα των αισθητηρίων μπορούν να σταλούν στο ThingSpeak από Arduino®, Raspberry Pi™, BeagleBone Black και άλλους εξοπλισμούς.



Εικόνα 3.1.1: ThingSpeak.com

Χρησιμοποιώντας λειτουργίες του MATLAB, το ThingSpeak μπορεί να δημιουργεί πίνακες με τις τιμές των αισθητηρίων και να τους μετατρέπει σε γραφικές παραστάσεις. Επίσης, χάρις στο MATLAB Analysis μπορεί να μας υπολογίσει την μέση τιμή των δεδομένων που δέχτηκε ή να εξαλείψουμε ακραία δεδομένα από ένα κανάλι, με την χρήση των συναρτήσεων που παρέχει το MATLAB. Μετά την ανάλυση, μπορούμε να γράψουμε δεδομένα σε ένα κανάλι ή να δημιουργήσουμε μια οπτικοποίηση τους, όπως περιγράφετε στην MATLAB Visualization εφαρμογή.



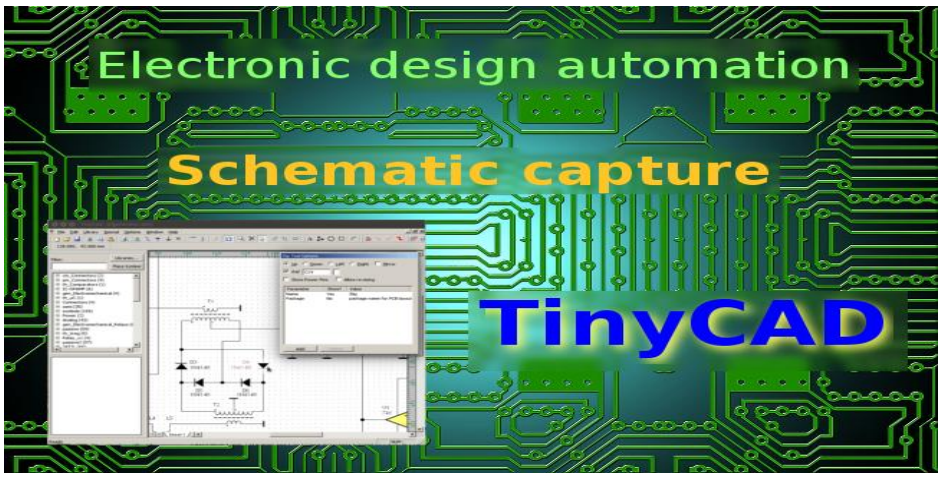
Εικόνα 3.1.2: Γραφικές παραστάσεις υγρασίας και θερμοκρασίας.

Χρησιμοποιούμε την MATLAB Visualizations εφαρμογή για οπτικοποιήσουμε τα δεδομένα μας σε ένα ThingSpeak κανάλι. Μπορούμε να δούμε και να περιηγηθούμε στα δεδομένα μας με την χρήση διαδραστικών οπτικοποιήσεων, όπως γραφικές παραστάσεις, διαγράμματα ροής ή διαγράμματα διασποράς σε στατιστική οπτικοποίηση, με την χρήση άλλων MATLAB διαγραμμάτων. Μπορούμε επίσης να θέσουμε τις οπτικοποιήσεις αυτές δημόσιες ούτος ώστε να χρησιμοποιήσουμε τον υπερσύνδεσμο (link) τους για να τα προσθέσουμε στην ιστοσελίδα μας.

Μπορούμε επίσης να προσθέσομε συναρτήσεις σε Scripts στο MATLAB Analysis και σε Visualization εφαρμογές ώστε να πετύχουμε τεχνικές τμηματικού προγραμματισμού.

Το ThingSpeak είναι μια IoT πλατφόρμα που χρησιμοποιεί κανάλια για την αποθήκευση δεδομένων που αποστέλλονται από εφαρμογές ή συσκευές. Με τις ρυθμίσεις που περιγράφονται στο Channel Configurations, δημιουργούμε ένα κανάλι και στην συνέχεια λαμβάνουμε ή στέλνουμε δεδομένα από και προς αυτό. Μπορούμε επίσης να θέσουμε το κανάλι μας δημόσιο για να δημοσιεύσουμε τα δεδομένα μας.

3.2 TinyCAD



Το TinyCAD είναι ένα ανοιχτού κώδικα πρόγραμμα σχεδιασμού ηλεκτρικών κυκλωμάτων και προσομοίωσης. Είναι απλό στην χρήση του, πολύ εύχρηστο για τον σχεδιασμό μικρών και απλών ηλεκτρικών κυκλωμάτων. Διαθέτει πολλές βιβλιοθήκες αλλά και επεκτάσεις. Επίσης περιέχει όλα τα βασικά σύμβολα με επιπλέον προσθήκες κάποια ολοκληρωμένα, τελεστικούς και φυσικά σύμβολα για τα Arduino. Τέλος, μπορεί κάποιος να βρει μέσα σε αυτό σύμβολα για αρσενικά και θηλυκά Pins σε περίπτωση που θελήσει να κάνει ένα πιο λεπτομερή σχεδιασμό.

Το πρόγραμμα μπορεί να ληφθεί από τον παρακάτω σύνδεσμο: https://sourceforge.net/projects/tinycad/

Στον οποίο σύνδεσμο υπάρχουν και βοηθήματα αλλά και ένα online forum για άμεση βοήθεια, τυχόν ερωτήσεις και απορίες ή και προβλήματα με το λογισμικό.

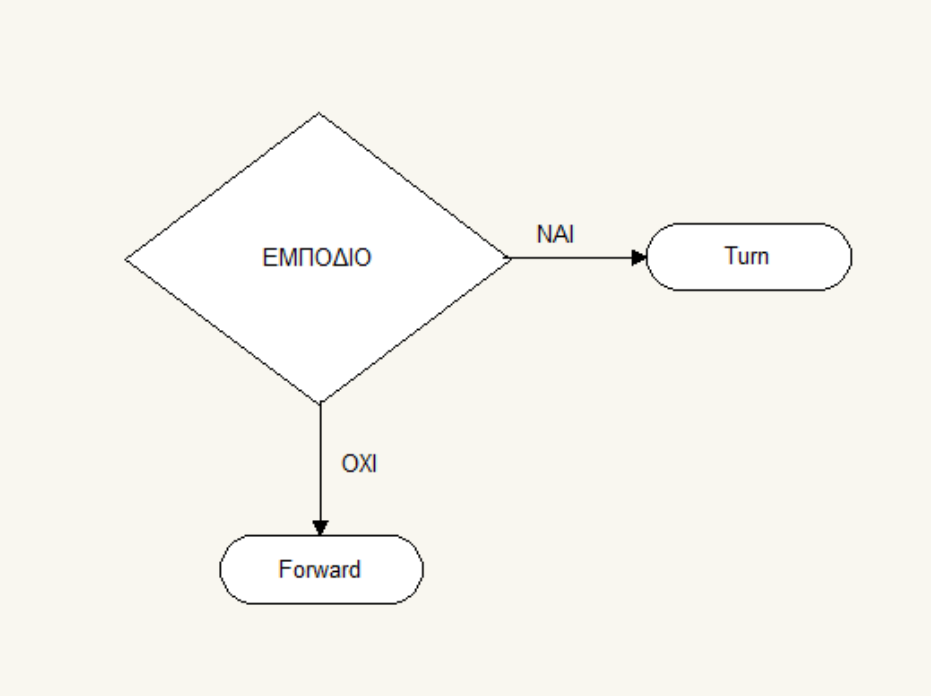
Είναι σημαντικό να αναφερθεί ότι το TinyCAD μπορεί να χρησιμοποιηθεί για την δημιουργία πλακέτας PCB και ότι σου προσφέρει την δυνατότητα σχεδιασμού δικού σου αντικειμένου, πχ κάποιου Arduino Shield.

4. ΚΙΝΗΣΗ ΤΟΥ ΡΟΜΠΟΤ

4.1 Φιλοσοφία της κίνησης

Η κίνηση και ο έλεγχος της είναι το μεγαλύτερο και δυσκολότερο μέρος αυτής της εργασίας. Μετά από κάποιο χρονικό διάστημα ερευνών που κάναμε, βρήκαμε μια κατασκευή η οποία έχει την δυνατότητα να στέκεται και να κινείται στα 2. Η κίνηση όμως έπρεπε να είναι ελεγχόμενη και να μπορεί το ρομπότ μας να κολλάει και να αχρηστεύεται εξαιτίας ενός εμποδίου που θα βρεθεί μπροστά του. Προσθέτοντας το αισθητήριο απόστασης UltraSonic και αξιοποιώντας το κατάλληλα μέσα στην κίνηση του ρομπότ, καταφέραμε να λύσουμε αυτό το πρόβλημα.

Πιο συγκεκριμένα, προσθέσαμε έναν έλεγχο της απόστασης που μας δίνει το UltraSonic με μια δική μας επιθυμητή τιμή (απόσταση). Έτσι, όταν η τιμή του αισθητηρίου είναι μικρότερη της επιθυμητής τιμής, σημαίνει ότι υπάρχει κάποιο εμπόδιο σε κοντινή απόσταση και το ρομπότ πρέπει να αλλάξει πορεία. Αν η τιμή του αισθητηρίου είναι μεγαλύτερη της επιθυμητής τιμής τότε δεν υπάρχει κάποιο πιθανό εμπόδιο σε κοντινή απόσταση και το ρομπότ μπορεί να συνεχίσει την ευθεία πορεία του.



Εικόνα 4.1.1: Λογικό διάγραμμα ελέγχου εμποδίων.

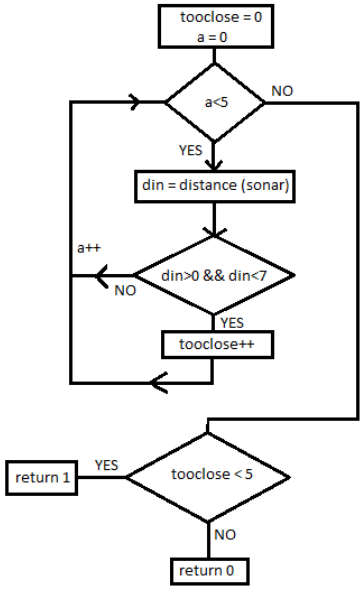
Όπως φαίνεται καθαρά και στο λογικό διάγραμμα, το ρομπότ θα συνεχίσει την ευθεία πορεία του όσο δεν αντιλαμβάνεται κάποιο εμπόδιο σε κοντινή απόσταση στην ευθεία που κινείται. Αλλιώς, θα κάνει πίσω αλλάζοντας κατεύθυνση και αφού ολοκληρωθεί η αλλαγή της πορείας, τότε συνεχίζει ευθεία μέχρι το επόμενο εμπόδιο.

Όμως, ο έλεγχος αυτός ήταν τόσο γρήγορος που θα είχε αρνητικά αποτελέσματα και ανακρίβεια στην αλλαγή πορείας του ρομπότ. Λόγο των υψηλών ταχυτήτων, μπορεί αυτός ο έλεγχος να τύχαινε όταν θα περνούσε κάτι τυχαία από το ρομπότ και αυτό θα το αντιλαμβανόταν ως εμπόδιο. Έτσι αυτός ο έλεγχος μπήκε σε μια δομή επανάληψης ώστε να πάρουμε δείγματα από το αποτέλεσμα του ελέγχου για μεγαλύτερη ακρίβεια. Έτσι κάναμε τον έλεγχο αυτόν για 6 φορές συνεχόμενα αυξάνοντας έναν μετρητή κάθε φορά που έβλεπε εμπόδιο κοντά του. Στο τέλος, έλεγχε αυτόν τον μετρητή με τον συνολικό αριθμό των μετρήσεων και αν το αποτέλεσμα ήταν μεγαλύτερο από έναν αριθμό (επαναλήψεων που επιλέξαμε εμείς) τότε υπήρχε όντως κάποιο εμπόδιο κοντά του. Αν όμως βρίσκαμε ότι ο μετρητής αυτός ήταν πχ 2, τότε θα καταλαβαίναμε ότι δεν υπάρχει σταθερό εμπόδιο μπροστά στο ρομπότ και ότι αυτό που αντιλήφθηκε ήταν κάτι κινούμενο, με αποτέλεσμα να μην χρειαστεί να αλλάξει την πορεία του.

```
bool TooClose()
{
    int tooclose = 0;
    for(int a=0; a<5; a++) {
        delay(50);
        int din = sonar.ping_in();
        if (din < 7 && din > 0) tooclose++;
    }
    if (tooclose < 5) return 1;
    return 0;
}
```

Εικόνα 4.1.2: Έλεγχος εμποδίου.

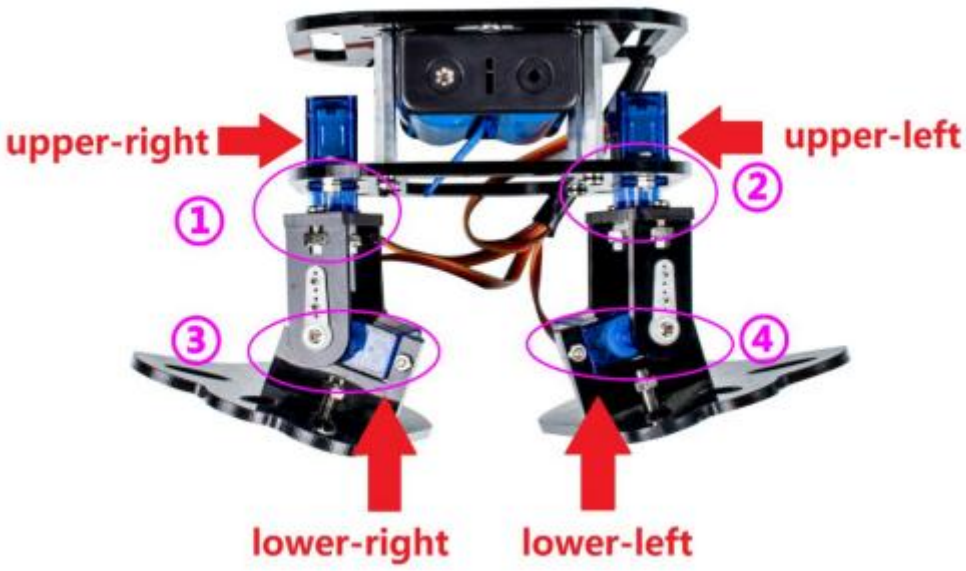
Τελικά, αυτό το ρομπότ είναι ικανό να κινείται ελεύθερα και αυτόνομα σε οποιονδήποτε χώρο κι αν αφεθεί έχοντας τη δυνατότητα να αποφεύγει τυχόν σταθερά εμπόδια, μιας και δεν θα αλλάζει πορεία σε κινούμενα αντικείμενα που πιθανόν να πέρασαν για κάποια στιγμή από μπροστά του.



Εικόνα 4.1.3: Λογικό διάγραμμα ελέγχου εμποδίου

4.2 Ανάλυση της κίνησης

Η κίνηση του ρομπότ βασίζεσαι στον συνδυασμό τεσσάρων σερβοκινητήρων. Άρα υπάρχουν 4 σημεία ελέγχου τα οποία αν οριστούν σωστά θα μπορεί να δημιουργηθεί ανθρωποειδής κίνηση σε κάποια προκαθορισμένα επαναλαμβανόμενα βήματα. Αυτό και κάναμε.



Εικόνα 4.2.1: Απεικόνιση θέσης σερβοκινητήρων.

Προτού φτάσουμε στον έλεγχο του κάθε σερβοκινητήρα χωριστά, πρέπει πρώτα να δηλώσουμε σε ποια έξοδο του Arduino θα αντιστοιχεί ο κάθε ένας (όπως απεικονίζονται στην εικόνα 4.2.1). Αυτό επιτυγχάνεται προσθέτοντας τις παρακάτω εντολές:

```
void Servo_Init()
{
    RU.attach(9);   // Connect the signal wire of the upper-right servo to pin 9
    RL.attach(10);   // Connect the signal wire of the lower-right servo to pin 10
    LU.attach(11);   // Connect the signal wire of the upper-left  servo to pin 11
    LL.attach(12);   // Connect the signal wire of the lower-left  servo to pin 12
}
```

Μετά από μελέτη και δοκιμές, καταλήξαμε σε μια βελτιωμένη έκδοση του πίνακα βημάτων που προϋπήρχε στα παραδείγματα του Arduino, πετυχαίνοντας την κίνηση «μπροστά» σε έξι επαναλαμβανόμενα βήματα ενώ την κίνηση «οπισθοχώρηση καιαλλαγή πορείας» σε πέντε επαναλαμβανόμενα βήματα.

Τα βήματα για την εμπρόσθια κίνηση φαίνονται στον παρακάτω πίνακα:

Upper-right	Lower-right	Upper-left	Lower-left
0	-40	0	-20
30	-40	30	-20
30	0	30	0
0	20	0	40
-30	20	-30	40
-30	0	-30	0

Τα βήματα για την οπισθοχώρηση και αλλαγή φοράς φαίνονται στον παρακάτω πίνακα:

Upper-right	Lower-right	Upper-left	Lower-left
-40	0	-20	0
-40	30	-20	30
0	30	0	30
30	0	30	0
0	0	0	0

Παρακάτω φαίνεται η δήλωση αυτών των πινάκων στο πρόγραμμα:

```
const int num1 = 6;
const int array_forward[num1][4] =
{
    {0,-40,0,-20},
    {30,-40,30,-20},
    {30,0,30,0},
    {0,20,0,40},
    {-30,20,-30,40},
    {-30,0,-30,0},
};

const int num2 = 5;
const int array_turn[num2][4] =
{
    {-40,0,-20,0},
    {-40,30,-20,30},
    {0,30,0,30},
    {30,0,30,0},
    {0,0,0,0},
};
```

Με βάση αυτούς τους πίνακες και με βάση την αρχική θέση που έχει ο κάθε κινητήρας, προσθέτουμε ή αφαιρούμε αυτές τις τιμές βήμα βήμα ούτως ώστε να έχουμε το επιθυμητό αποτέλεσμα.

Τέλος , πρέπει να αναφερθεί και η σωστή αρχικοποίηση της θέσης των σερβοκινητήρων η οποία γίνεται καταχωρώντας τις κατάλληλες τιμές στην παρακάτω εντολή:

```
const int array_cal[4] = {95,135,90,95};
```

Οι παραπάνω τιμές προέκυψαν μετά από δοκιμές και παρατήρηση της θέσης.

4.2.1 Εγκατάσταση και Προσαρμογή

Το πρόγραμμα της κίνησης του ρομπότ χωρίζεται σε τρία στάδια:

```
//#define INSTALL
//#define CALIBRATION
//#define RUN
```

Εγκατάσταση

Κατά την διαδικασία της εγκατάστασης τρέχουμε το πρόγραμμα με την εντολή #define INSTALL αφήνοντας τις υπόλοιπες 2 εντολές ως σχόλια. Αυτή η εντολή περνάει όλο το πρόγραμμα στο Arduino και ουσιαστικά ετοιμάζει το ρομπότ μας για το επόμενο βήμα.

Προσαρμογή

Τρέχουμε το πρόγραμμα αφήνοντας σε σχόλια τις υπόλοιπες 2 εντολές. Κατά την διαδικασία αυτή θα τεθούν οι αρχικές τιμές ή οι τιμές σε κανονικές συνθήκες και θα αρχικοποιηθεί η θέση των κινητήρων. Θα παρατηρήσουμε ότι τα πόδια κινούνται ούτως ώστε να ευθυγραμμιστούν με το έδαφος.

Λειτουργία

Τέλος, σε αυτό το στάδιο επιλέγουμε να τρέξουμε το πρόγραμμα με την εντολή RUN. Κατ αυτό το στάδιο το ρομπότ αρχίζει να κινείται στον χώρο κανονικά.

4.2.2 Ταχύτητα κίνησης

Στο συγκεκριμένο πρόγραμμα ο προγραμματιστής μπορεί να μεταβάλει την ταχύτητα με την οποία θα εκτελούνται τα βήματα από τους πίνακες κίνησης που δείξαμε παραπάνω, καθώς και την καθυστέρηση που θα υπάρχει σε κάθε λούπα κατά την επεξεργασία των πράξεων μεταξύ του πίνακα με τις αρχικές συνθήκες και των πινάκων των κινήσεων. Αυτό θα έχει σαν αποτέλεσμα το ρομπότ να κινείται πιο γρήγορα ή πιο αργά.

```
const int vel = 20, vel_Back = 10;
const int delay_Forward = 750, delay_Back = 1000;
```

Οι τιμές vel και delay_Forward αναφέρονται στην επιτάχυνση και καθυστέρηση για την ευθεία κίνηση ενώ οι vel_Back και delay_Back αναφέρονται στην κίνηση της στροφής.

Το αποτέλεσμα της παραμετροποίησης των παραπάνω μεταβλητών μεταβάλει την ταχύτητα της κίνησης , και αυτό μπορεί να φανεί παρακάτω:

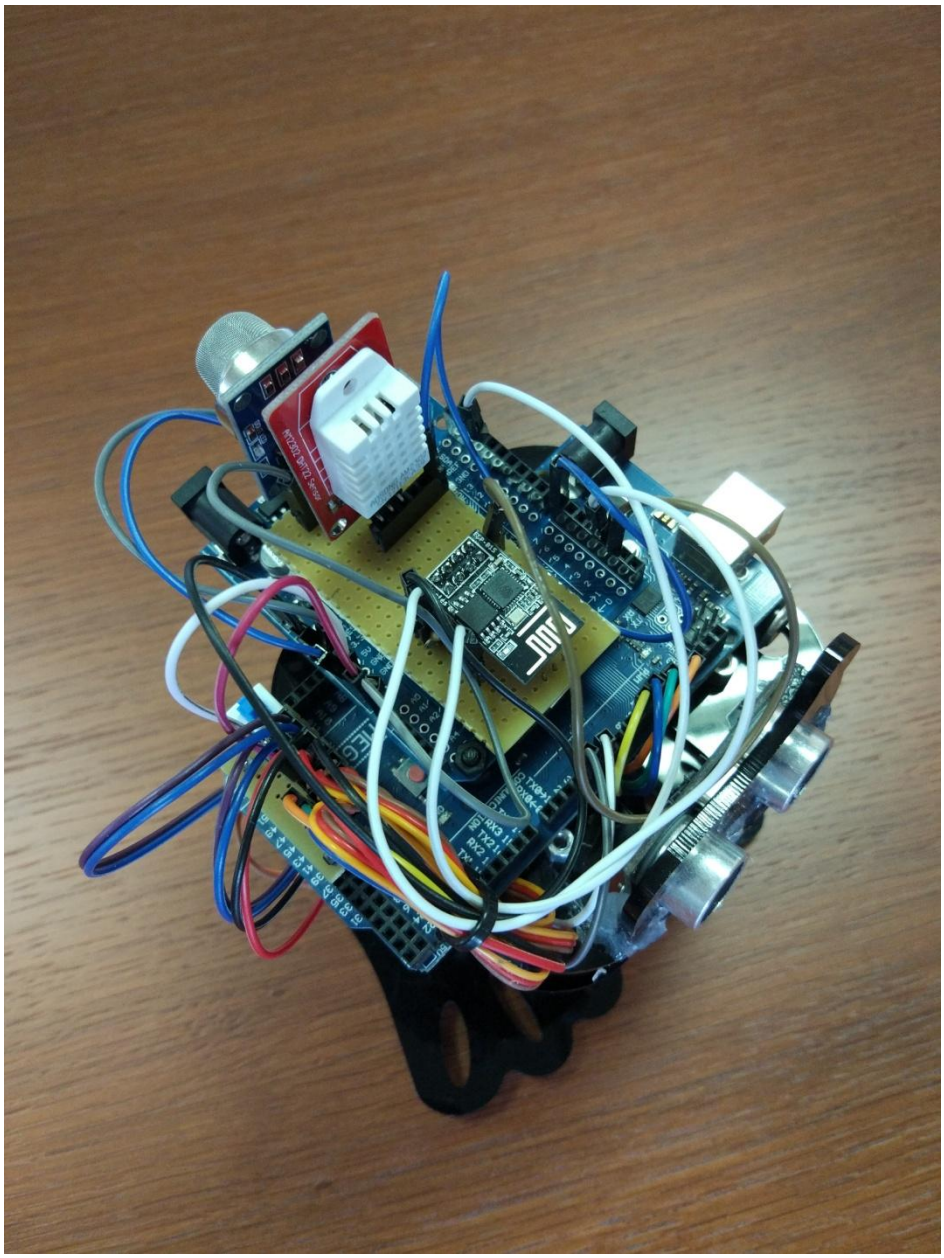
```
void Forward()
{
    for(int x=0; x<num1; x++) {
        RU.slowmove (array_cal[0] + array_forward[x][0] , vel);
        RL.slowmove (array_cal[1] + array_forward[x][1] , vel);
        LU.slowmove (array_cal[2] + array_forward[x][2] , vel);
        LL.slowmove (array_cal[3] + array_forward[x][3] , vel);
        delay(delay_Forward);
    }
}

void Backward()
{
    for(int z=0; z<4; z++) {
        for(int y=0; y<num2; y++) {
            RU.slowmove (array_cal[0] + array_turn[y][0] , vel_Back);
            RL.slowmove (array_cal[1] + array_turn[y][1] , vel_Back);
            LU.slowmove (array_cal[2] + array_turn[y][2] , vel_Back);
            LL.slowmove (array_cal[3] + array_turn[y][3] , vel_Back);
            delay(delay_Back);
        }
    }
}
```

Παρατηρούμε ότι αλλάζοντας οποιαδήποτε από τις παραπάνω μεταβλητές θα υπάρξει αύξηση ή μείωση της ταχύτητας κίνησης του ρομπότ.

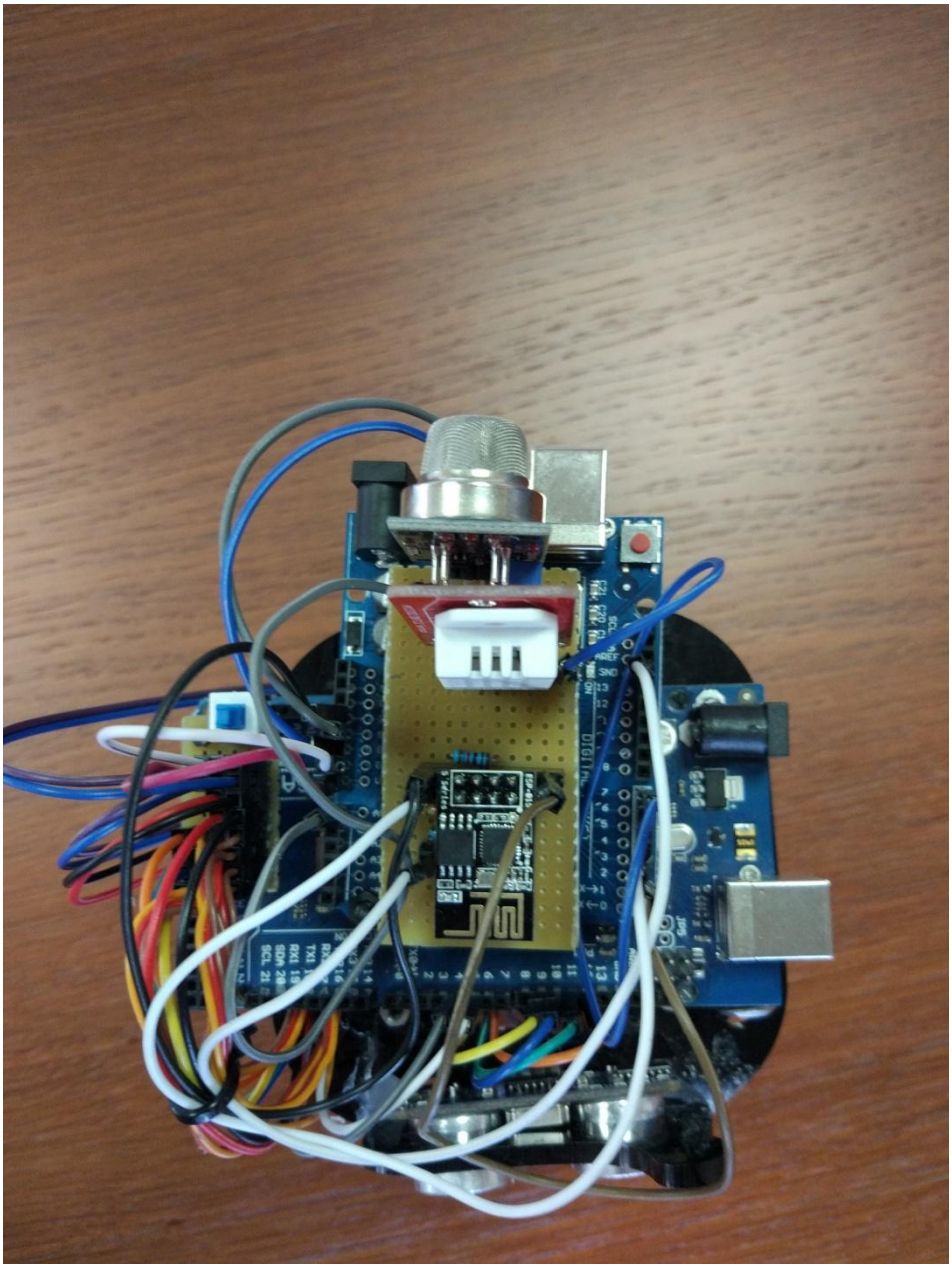
4.3 Αντιμετώπιση αστάθειας

Από την στιγμή που το ρομπότ στέκεται στα 2 του πόδια δικαίως ο καθένας μας θα σκεφτόταν ότι χρειάζεται κάποιος έλεγχος ισορροπίας (ανάστροφο εκκρεμές). Στην προκειμένη περίπτωση δεν χρειάζεται κάποια μέθοδος ισορροπίας μιας και το ρομπότ διαθέτει 2 πέλματα που καταλαμβάνουν μεγάλη επιφάνεια και μπορεί πατώντας σε αυτά να διατηρείται όρθιο χωρίς να πέσει.

Επιπλέον, η κατανομή των υλικών πάνω στο ρομπότ έπαιξε σημαντικό ρόλο στη διατήρηση της ισορροπίας του. Για αυτόν τον λόγο αποφασίσαμε να κάνουμε μια καλή κατανομή του χώρου ούτως ώστε το κέντρο βάρους της κατασκευή να παραμείνει το ίδιο και να μην υπάρχει εξοπλισμός που να βρίσκεται εκτός σκελετού.



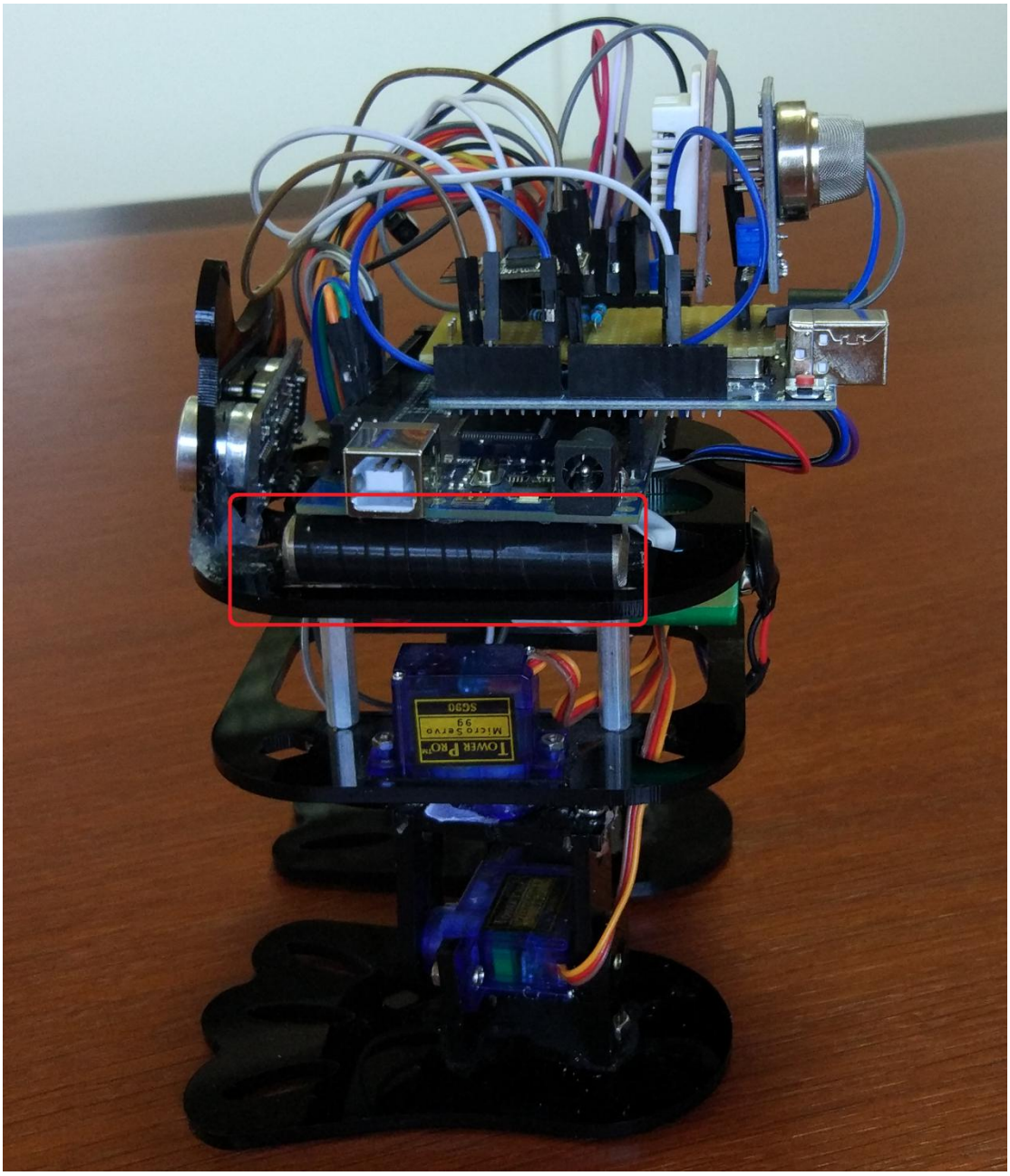
Εικόνα 4.3.1: Θέση εξαρτημάτων πάνω στην κατασκευή.



Εικόνα 4.3.2: Θέση εξαρτημάτων πάνω στην κατασκευή.

Παρά τα μεγάλα του πόδια, το ρομπότ δεν ήταν τόσο σταθερό κατά την διάρκεια της κίνησης του και σε κάποιες στιγμές έγερνε πολύ μπροστά (οριακά να πέσει). Μετά από παρατήρηση της κίνησης διαπιστώσαμε ότι χρειαζόταν ένα αντίβαρο για να μπορέσει να κρατήσει την ισορροπία του και να κάνει πιο σταθερά την κίνηση του.

Αφού μελετήσαμε τις πιθανές εκδοχές για την δημιουργία και του βαριδίου αυτού, καταλήξαμε σε ένα σχήμα το οποία ταίριαζε απόλυτα στον ελεύθερο χώρο που διαθέταμε και δεν επηρέαζε τα υπόλοιπα. Ωστόσο για το βάρος του βαριδίου καταλήξαμε μετά από κάποιες δοκιμές.



Εικόνα 4.3.3: Θέση και σχήμα αντίβαρου.

5. ΕΦΑΡΜΟΓΕΣ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

5.1 Παραδείγματα εφαρμογών.

Θα ήταν χρήσιμο να αναφερθούν κάποια πραγματικά παραδείγματα χρησιμότητας και βοήθειας που μπορεί να προσφέρει αυτό το ρομπότ.

Επειδή σε έναν βιομηχανικό χώρο παίζει σημαντικό ρόλο ο ανθρώπινος παράγοντας γίνονται και λάθη απροσεξία ή αμέλειας. Ένα τέτοιο λάθος θα ήταν να ξεχαστεί ανοιχτή μια μπουκάλα αερίου που χρησιμοποιείται για κοπή μετάλλων (πχ μπουκάλα προπανίου, οξυγόνου κτλ) ή και ένα πιο μικρό εργαλείο όπως είναι το φλόγιστρο. Αυτό θα μπορούσε να δημιουργήσει μεγάλα προβλήματα όπως εκρήξεις ή πρόκληση προβλημάτων υγείας σε κάποιον εργαζόμενο, αλλά και μικρότερα προβλήματα, όπως άδειασμα της μπουκάλας, καταστροφή του εξοπλισμού, περιττά έξοδα στην επιχείρηση ή ακόμα και διχόνοιες ανάμεσα στους εργαζομένους. Ένα τέτοιο πρόβλημα μπορεί να εντοπιστεί από το ρομπότ το οποίο θα αντιληφθεί το αέριο στο χώρο και θα ενημερώσει το σύστημα το οποίο παρακολουθείται από κάποιον χειριστή ή συντηρητή με αποτέλεσμα να βρεθεί αυτή η μικροδιαρροή και να εξαλειφθεί.

Κάτι αντίστοιχο θα μπορούσε να συμβεί και σε κάποιον βιομηχανικό χώρο όπου περιέχονται μεγάλες εγκαταστάσεις ψύξεως. Δηλαδή κάποιος εργαζόμενος που βιάζεται να σχολάσει να ξεχάσει μισόκλειστη την πόρτα μιας τέτοιας μονάδας και να υπάρχει απώλεια ψύξης. Αυτό μπορεί να εντοπιστεί από το ρομπότ το οποίο θα παρατηρήσει μια πτώση θερμοκρασίας και θα στείλει τα δεδομένα στην online πλατφόρμα απ’ όπου μπορεί ο χειριστής να το διαπιστώσει και να κατευθυνθεί προς τον χώρο με την πτώση θερμοκρασίας να ελέγξει μόνος του περί τίνος πρόκειται. Αυτό θα μπορούσε να αποτρέψει την επιχείρηση από μια δαπανηρή ενέργεια που προκλήθηκε από απροσεξία. Επίσης, έτσι μπορεί να προστατευτεί και ο εξοπλισμός ενός εργοστασίου.

Ομοίως, θα μπορούσε να παρατηρηθεί και η λειτουργία ενός θερμαντικού σώματος μέσα στον βιομηχανικό χώρο, ξεχασμένο από κάποιον εργαζόμενο.

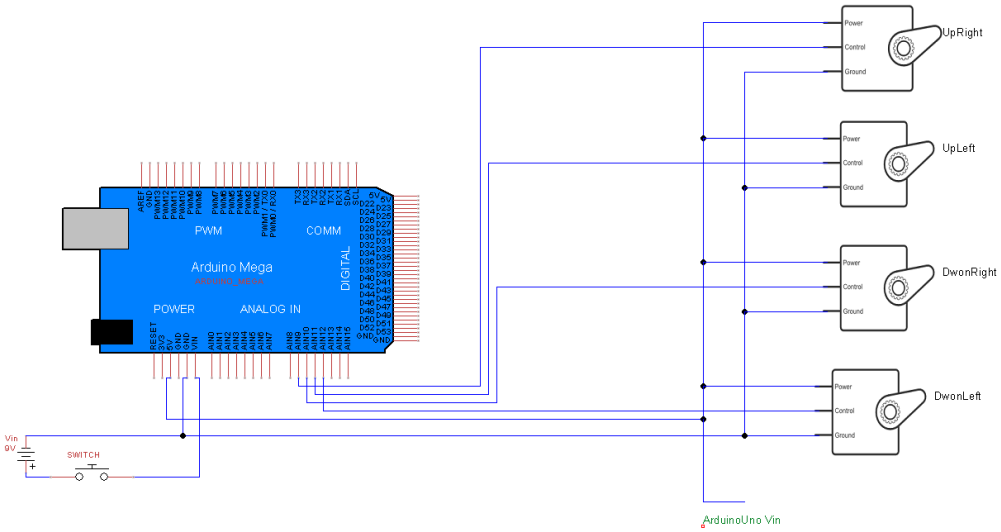
5.2 Προτάσεις εξέλιξης.

Μιας και θα ήταν πολύ βοηθητικό για τους συναδέλφους μας που θα ήθελαν να ασχοληθούμε με κάτι παρόμοιο στην πτυχιακή τους, θα θέλαμε εδώ να αναφέρουμε κάποιες ιδέες για αναβάθμιση και εξέλιξη της υπάρχουσας πτυχιακής.

- Γράψιμο κώδικα ώστε να βγάζει κατάλληλη ένδειξη ανάλογα τις τιμές που παίρνει από τα αισθητήρια και να το στέλνει σε κάποια Online πλατφόρμα ή ακόμα και σε κινητό τηλέφωνο (συντηρητή κτλ).
- Βέλτιστη κατανάλωση ρεύματος και εξοικονόμηση ενέργειας.
- Ανάπτυξη συστήματος αυτό-φόρτισης μέσω ενός φωτοβολταικού πάνελ (όπου θέλει κ έλεγχο φωτεινότητας για εντοπισμό του ήλιου).
- Ανάπτυξη δυνατότητας χαρτογράφησης του χώρου για ακρίβεια δεδομένων στον χώρο και γραφική απεικόνισή τους.
- Δημιουργία Android εφαρμογής για επικοινωνία και παρακολούθηση του ρομπότ.
- Έξυπνη συμπεριφορά του ρομπότ (πχ όταν αντιληφθεί υψηλή θερμοκρασία να τριγυρίζει εκεί κοντά για να μπορέσει να εντοπίσει πως προέκυψε).
- Δυνατότητα όρασης σε σκοτεινό περιβάλλον για νυχτερινή λειτουργία χωρίς ανάγκη φωτισμού.
- Προσθήκη δυνατότητας ακοής και απεικόνισης θορύβου (κατ’ αυτόν τον τρόπο μπορούν να παρθούν πολλές πληροφορίες για συμβάντα στον χώρο).

6. ΣΧΕΔΙΑ ΗΛΕΚΤΡΙΚΩΝ ΚΥΚΛΩΜΑΤΩΝ

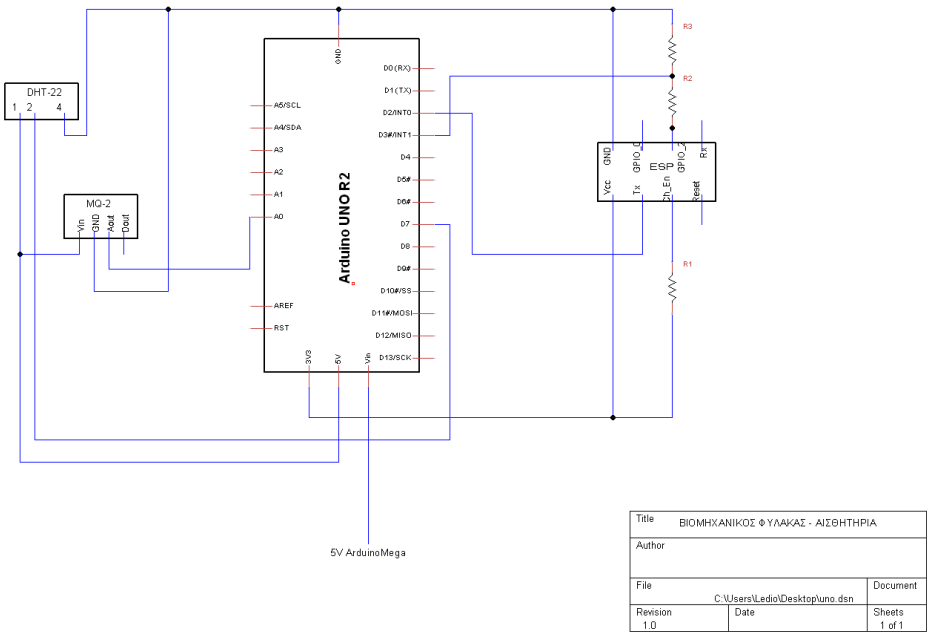
6.1 Ηλεκτρικό κύκλωμα κίνησης.



Title	ΒΙΟΜΗΧΑΝΙΚΟΣ ΦΥΛΑΚΑΣ- ΚΙΝΗΣΗ	
Author		
File	C:\Users\Ledio\Desktop\mega.dsn	Document
Revision 1.0	Date	Sheets 1 of 1

Εικόνα 6.1.1: Κύκλωμα κίνησης μέσω TinyCAD.

6.2 Ηλεκτρικό κύκλωμα σημάτων.



Εικόνα 6.2.1: Κύκλωμα σημάτων μέσω TinyCAD.

7. ΚΩΔΙΚΑΣ

7.1 Κώδικας κίνησης.

```
#include "VarSpeedServo.h"  //include the VarSpeedServo library

#include <NewPing.h>       //include the NewPing library

//#include <Servo.h>


VarSpeedServo RU;  //Right Upper

VarSpeedServo RL;  //Right Lower

VarSpeedServo LU;  //Left Upper

VarSpeedServo LL;  //Left Lower



NewPing sonar(5,4,200);              //vel(min), delay_Forward(max) = (5, 2000)


const int vel = 20, vel_Back = 10;               //vel(mid), delay_Forward(mid) = (20, 750)


const int delay_Forward = 750, delay_Back = 1000;
//vel(max), delay_Forward(min)= (256, 50)


 //wonderful ---> (10, 700) (50, 500) (100, 100) (100, 300) (100, 500)


const int array_cal[4] = {95,100,90,95};

int RU_Degree = 0, LU_Degree = array_cal[2] + 5;
```

```
const int num1 = 6;
const int array_forward[num1][4] =
{
   {0,-40,0,-20},
   {30,-40,30,-20},
   {30,0,30,0},
   {0,20,0,40},
   {-30,20,-30,40},
   {-30,0,-30,0},
};

const int num2 = 5;
const int array_turn[num2][4] =
{
   {-40,0,-20,0},
   {-40,30,-20,30},
   {0,30,0,30},
   {30,0,30,0},
   {0,0,0,0},
};

//#define INSTALL
#define CALIBRATION
//#define RUN
```

```
void Servo_Init()
{
   RU.attach(9);   // Connect the signal wire of the upper-right servo to pin 9
   RL.attach(10);   // Connect the signal wire of the lower-right servo to pin 10
   LU.attach(11);   // Connect the signal wire of the upper-left  servo to pin 11
   LL.attach(12);   // Connect the signal wire of the lower-left  servo to pin 12
}



void Adjust()                        // Avoid the servo's fast spinning in initialization
{                                    // RU,LU goes from array_cal[0] - 5 ,array_cal[2] + 5 degrees to
array_cal[0],array_cal[2] degrees
   for(RU_Degree = array_cal[0] - 5; RU_Degree <= array_cal[0]; RU_Degree += 1)
{
      RU.write(RU_Degree);           // in steps of 1 degree
      LU.write(LU_Degree--);
 // tell servo to go to RU_Degreeition, LU_Degreeition in variable 'RU_Degree', 'LU_Degree'
      delay(15);                     // waits 15ms for the servo to reach the RU_Degreeition
   }
}


bool TooClose()
{
   int tooclose = 0;
   for(int a=0; a<5; a++) {
      delay(50);
      int din = sonar.ping_in();
      if (din < 7 && din > 0) tooclose++;
```

```
    }
    if (tooclose < 5) return 1;
    return 0;
}

void Forward()
{
    for(int x=0; x<num1; x++) {
        RU.slowmove (array_cal[0] + array_forward[x][0] , vel);
        RL.slowmove (array_cal[1] + array_forward[x][1] , vel);
        LU.slowmove (array_cal[2] + array_forward[x][2] , vel);
        LL.slowmove (array_cal[3] + array_forward[x][3] , vel);
        delay(delay_Forward);
    }
}

void Backward()
{
    for(int z=0; z<4; z++) {
        for(int y=0; y<num2; y++) {
            RU.slowmove (array_cal[0] + array_turn[y][0] , vel_Back);
            RL.slowmove (array_cal[1] + array_turn[y][1] , vel_Back);
            LU.slowmove (array_cal[2] + array_turn[y][2] , vel_Back);
            LL.slowmove (array_cal[3] + array_turn[y][3] , vel_Back);
            delay(delay_Back);
        }
    }
```

```
}

void setup()
{
#ifdef INSTALL
   Servo_Init();

   RU.slowmove (90 , vel);
   RL.slowmove (90 , vel);
   LU.slowmove (90 , vel);
   LL.slowmove (90 , vel);
   while(1);
#endif

#ifdef CALIBRATION
   Servo_Init();
   Adjust();

   RL.slowmove (array_cal[1] , vel);
   LL.slowmove (array_cal[3] , vel);
   delay(2000);
   while(1);
#endif

#ifdef RUN
   Servo_Init();
   Adjust();
```

```
   RL.slowmove (array_cal[1] , vel);
   LL.slowmove (array_cal[3] , vel);
   delay(2000);
#endif
}

void loop()
{
   while(TooClose()) Forward();
   Backward();
}
```

7.2 Κώδικας σημάτων και αισθητηρίων.

```
#include <SoftwareSerial.h>

SoftwareSerial espSerial =  SoftwareSerial(2,3);


#define       MQ_PIN                  (0)     //define which analog input channel you are going to
use

#define       RL_VALUE                (5)     //define the load resistance on the board, in kilo
ohms

#define       RO_CLEAN_AIR_FACTOR     (9.83)  //RO_CLEAR_AIR_FACTOR=(Sensor
resistance in clean air)/RO,

                                      //which is derived from the chart in datasheet


#define       CALIBARAION_SAMPLE_TIMES    (50)    //define how many samples you are
going to take in the calibration phase

#define       CALIBRATION_SAMPLE_INTERVAL  (500)   //define the time interal(in
milisecond) between each samples in the

                                      //cablibration phase

#define       READ_SAMPLE_INTERVAL        (50)    //define how many samples you are going
to take in normal operation

#define       READ_SAMPLE_TIMES           (5)     //define the time interal(in milisecond)
between each samples in

                                      //normal operation


#define       GAS_LPG                 (0)

#define       GAS_CO                  (1)

#define       GAS_SMOKE               (2)


float         LPGCurve[3]  =  {2.3,0.21,-0.47};   //two points are taken from the curve.
```

```
                                    //with these two points, a line is formed which is
"approximately equivalent"
                                    //to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg200, 0.21), point2:
(lg10000, -0.59)
float       COCurve[3]  =  {2.3,0.72,-0.34};    //two points are taken from the curve.
                                    //with these two points, a line is formed which is
"approximately equivalent"
                                    //to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg200, 0.72), point2:
(lg10000,  0.15)
float       SmokeCurve[3] ={2.3,0.53,-0.44};    //two points are taken from the curve.
                                    //with these two points, a line is formed which is
"approximately equivalent"
                                    //to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg200, 0.53), point2:
(lg10000,  -0.22)
float       Ro          =  10;              //Ro is initialized to 10 kilo ohms


#include <DHT.h>
#define DHTPIN 7
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);



String apiKey = "VPOTQUGQ3OKCZWF5";


String ssid="MonkeyDLuffy";
String password ="HMG528HMG528";
```

```
boolean DEBUG=true;

int soundSensor = 9;
int soundLed = 7;

long sentTime = -16000;

int movement = 4;
int movementLed = 6;

const int sensorMin = 0;
const int sensorMax = 1024;

int sensorFlag = 0;

void showResponse(int waitTime){
   long t=millis();
   char c;
   while (t+waitTime>millis()){
     if (espSerial.available()){
       c=espSerial.read();
       if (DEBUG) Serial.print(c);
     }
   }

}
```

```
boolean thingSpeakWrite(float value1, float value2, int value3, int value4, int value5, int
value6, int value7){
 String cmd = "AT+CIPSTART=\"TCP\",\"";
 cmd += "184.106.153.149";
 cmd += "\",80";
 espSerial.println(cmd);
 if (DEBUG) Serial.println(cmd);
 if(espSerial.find("Error")){
  if (DEBUG) Serial.println("AT+CIPSTART error");
  return false;
 }


 String getStr = "GET /update?api_key=";
 getStr += apiKey;

 getStr +="&field1=";
 getStr += String(value1);
 getStr +="&field2=";
 getStr += String(value2);
 getStr +="&field3=";
 getStr += String(value3);
 getStr +="&field4=";
 getStr += String(value4);
 getStr +="&field5=";
 getStr += String(value5);
 getStr +="&field6=";
 getStr += String(value6);
```

```
 getStr +="&field7=";
 getStr += String(value7);

 getStr += "\r\n\r\n";

 // send data length
 cmd = "AT+CIPSEND=";
 cmd += String(getStr.length());
 espSerial.println(cmd);
 if (DEBUG)  Serial.println(cmd);

 delay(100);
 if(espSerial.find(">")){
  espSerial.print(getStr);
  if (DEBUG)  Serial.print(getStr);
 }
 else{
  espSerial.println("AT+CIPCLOSE");
  // alert user
  if (DEBUG)   Serial.println("AT+CIPCLOSE");
  return false;
 }
 return true;
}


float MQResistanceCalculation(int raw_adc)
```

```
{
  return ( ((float)RL_VALUE*(1023-raw_adc)/raw_adc));
}

float MQCalibration(int mq_pin)
{
  int i;
  float val=0;

  for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {          //take multiple samples
    val += MQResistanceCalculation(analogRead(mq_pin));
    delay(CALIBRATION_SAMPLE_INTERVAL);
  }
  val = val/CALIBARAION_SAMPLE_TIMES;                 //calculate the average value

  val = val/RO_CLEAN_AIR_FACTOR;                      //divided by RO_CLEAN_AIR_FACTOR yields
the Ro
                                                      //according to the chart in the datasheet

  return val;
}

float MQRead(int mq_pin)
{
  int i;
  float rs=0;

  for (i=0;i<READ_SAMPLE_TIMES;i++) {
```

```
    rs += MQResistanceCalculation(analogRead(mq_pin));
    delay(READ_SAMPLE_INTERVAL);
  }

  rs = rs/READ_SAMPLE_TIMES;

  return rs;
}

int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
  if ( gas_id == GAS_LPG ) {
     return MQGetPercentage(rs_ro_ratio,LPGCurve);
  } else if ( gas_id == GAS_CO ) {
     return MQGetPercentage(rs_ro_ratio,COCurve);
  } else if ( gas_id == GAS_SMOKE ) {
     return MQGetPercentage(rs_ro_ratio,SmokeCurve);
  }

  return 0;
}

int  MQGetPercentage(float rs_ro_ratio, float *pcurve)
{
  return (pow(10,( ((log(rs_ro_ratio)-pcurve[1])/pcurve[2]) + pcurve[0])));
}
```

```
static int initProgram() {
  long curTime = millis();

  if((curTime < sentTime && curTime > 16000) || curTime > 16000 + sentTime){
    sensors();
    sentTime = curTime;
  }
  int statusSensor = digitalRead(soundSensor);
  if(statusSensor == 1){
    digitalWrite(soundLed, HIGH);
  }
  else{
    digitalWrite(soundLed, LOW);
  }

  int var = digitalRead(movement);
  if (var == HIGH){
    sensorFlag = 1;
  }
  else{
    digitalWrite(movementLed, LOW);
  }
}

static int sensors() {
  float t = dht.readTemperature();
  float h = dht.readHumidity();
```

```
int mq1 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_LPG);
int mq2 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_CO);
int mq3 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_SMOKE);

int sensorReading = analogRead(A1);
int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

Serial.println("Temp="+String(t)+" *C");
Serial.println("Humidity="+String(h)+" %");

Serial.print("LPG:");
Serial.print(mq1);
Serial.println( "ppm" );
Serial.print("CO:");
Serial.print(mq2);
Serial.println( "ppm" );
Serial.print("SMOKE:");
Serial.print(mq3);
Serial.println( "ppm" );

switch (range) {
case 0:
  Serial.println("** Fire closer than 1.5 feet away. **");
  break;
case 1:
  Serial.println("** Fire between 1-3 feet away. **");
```

```
    break;
  case 2:
    Serial.println("No Fire detected.");
    break;
  }

  thingSpeakWrite(t, h, mq1, mq2, mq3, range, sensorFlag);
  sensorFlag = 0;
}

void setup() {
  DEBUG=true;
  Serial.begin(9600);

  dht.begin();

  espSerial.begin(9600);

  espSerial.println("AT+CWMODE=1");
  showResponse(1000);

  espSerial.println("AT+CWJAP=\""+ssid+"\",\""+password+"\"");
  showResponse(5000);

  if (DEBUG)  Serial.println("Setup completed");

  Serial.println("Calibrating...");
```

```
  Ro = MQCalibration(MQ_PIN);
  Serial.println("Calibration is done...");
  Serial.print("Ro=");
  Serial.print(Ro);
  Serial.println(" kohm");

  pinMode(soundSensor, INPUT);
  pinMode(soundLed, OUTPUT);
  pinMode(movement, INPUT);
  pinMode(movementLed, OUTPUT);

  Serial.println("\nSensors Started!\n");
}

void loop() {
  initProgram();
}
```

8. Υλικά Πτυχιακής

Τεμάχιο	Υλικό	Τιμή
2	Ultrasonic Module	0,83 €
1	Arduino Mega 2560 with USB Cable	8,80 €
1	Arduino Uno	1,75 €
1	SunFounder Robotic kit	46,64 €
2	AM2302 DHT22	4,86 €
2	MQ2 Gas Sensor	2,18 €
2	ESP WiFi module	2,86 €
3	Μπαταρία 9V	9,60 €
1	Διάτρητη πλακέτα	2,50 €
3	Σειρά αρσενικά pin	0,60 €
3	Σειρά θυλικά pin	0,60 €
1	Εύκαμπτο καλώδιο θυλικό αρσενικό	1,24 €
1	Εύκαμπτο καλώδιο αρσενικό αρσενικό	1,24 €
3	Διακόπτης ON-OFF	0,25 €
30	Αντιστάσεις	0,50 €
Συνολο		**84,45 €**

Επίσης στην εκπόνηση αυτής της εργασίας χρησιμοποιήθηκαν και πολλά εργαλεία όπως

- κολλητήρι
- καλάι
- πριόνι
- κατσαβίδια
- φαλτσέτα
- σιλικόνη
- πάγκος εργασίας
- αποστάτες
- βίδες
- μονωτική ταινία

τα οποία ανήκαν στην εταιρία που εργαζόμαστε και είχαμε την άδεια να τα χρησιμοποιήσουμε.

ΑΝΑΦΟΡΕΣ

Βιβλία

1. ΗΛΕΚΤΡΟΝΙΚΗ (7η Έκδοση) - Albert Malvino David J. Bates
2. Ψηφιακά Ηλεκτρονικά (5η Έκδοση) - Leach & Malvino
3. Σύστημα Βασισμένο σε Arduino για την παρατήρηση ροής υγρού - Βλαγοΐδης Νικόλαος
4. Ηλεκτρικές Μηχανές – Stephen Chapman
5. Προγραμματίζοντας με τον μικροελεγκτή Arduino - Εμμανουήλ Πουλάκης

WEBSITES

1. http://www.hlektronika.gr/ (Κοινότητα ηλεκτρονικών)
2. https://www.sas.com/el_gr/insights/big-data/internet-of-things.html
3. https:// wikipedia.org/
4. https://www.arduino.cc/
5. https://www.youtube.com/channel/UCEKKd7DJkhJ6fnQNX9mKvGw
6. https://grobotronics.com/esp8266-wifi-module.html?sl=el
7. https://playground.arduino.cc/Code/NewPing
8. https://playground.arduino.cc/ComponentLib/Servo

ΠΑΡΑΡΤΗΜΑ