***Υπεύθυνη Δήλωση*** *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή 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 πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.*

## ΠΡΟΛΟΓΟΣ

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η διερεύνηση και αξιοποίηση των δυνατοτήτων χαμηλού επιπέδου της ηλεκτρονικής πλατφόρμας Arduino για τη δημιουργία ενός Ραντάρ. Στη διαδικασία δημιουργίας του Ραντάρ είναι απαραίτητη η γνώση προγραμματιστικής γλώσσας του Arduino, η οποία βασίζεται στην C++ , καθώς και η λειτουργία των μικροσυσκευών που χρησιμοποιήθηκαν σε επίπεδο ‘’σχεδόν’’ υλικού. Παρόμοια εμπορικά συστήματα και εργαλεία διατίθενται ευρέως σε παγκόσμιο επίπεδο, όμως αυτό που κάνει το Arduino να ξεχωρίζει είναι το χαμηλότερο κόστος σε σχέση με τα υπόλοιπα συστήματα , όπως επίσης η άμεση επέμβαση του χρήστη στο προγραμματιστικό περιβάλλον του ίδιου του υλικού (hardware) του Arduino, με αποτέλεσμα το Arduino να έχει προσελκύσει αρκετά μεγάλο ενδιαφέρον.

Κατά την εκπόνηση της εργασίας παρουσιάστηκαν κάποιες δυσκολίες με σημαντικότερες τις παρακάτω.

- Δυσχέρεια πρόσβασης στην κατάλληλη βιβλιογραφία λόγω του μικρού χρονικού διαστήματος που είναι διαθέσιμη η συγκεκριμένη πλατφόρμα, στα πλαίσια της εργασίας. Οι πηγές περιορίζονται ουσιαστικά σε ξενόγλωσση αρθρογραφία και στο internet. Απαιτήθηκε κόπος και χρόνος για τη συλλογή του απαραίτητου υλικού, καθώς ήταν δύσκολα προσβάσιμο.
- Ευρύτητα εφαρμογών, αλλά μικρή εμπειρία. Μέχρι στιγμής στην Ελλάδα από ότι είναι γνωστό, δεν έχει καταγραφεί σε μεγάλη έκταση στο διαδίκτυο μελέτη ή έρευνα με τη χρήση της συγκριμένης ηλεκτρονικής πλατφόρμας με χρήση διαφόρων αισθητήρων ή άλλων μικροσυσκευών που χρησιμοποιήθηκαν και στην παρούσα εργασία. Είναι χαρακτηριστική η ιδιαίτερα περιορισμένη σχετική αρθρογραφία, ακόμα και για τη συγκεκριμένη εφαρμογή Ραντάρ.

Η εργασία αποτελείται από δύο μέρη: θεωρητικό και πειραματικό. Το κάθε μέρος αποτελείται από δύο κεφάλαια.

- Το πρώτο κεφάλαιο αναφέρεται στο μικροελεγκτή που χρησιμοποιούμε για τη κατασκευή μας. Αναλύουμε τα πλεονεκτήματα και τα μειονεκτήματα του μικροελεγκτή , την αρχιτεκτονική του, τα διάφορα μοντέλα που κυκλοφορούν στην αγορά, τα τεχνικά

χαρακτηριστικά, τον τρόπο επικοινωνίας του με τον υπολογιστή και το προγραμματιστικό του περιβάλλον.

- Το δεύτερο κεφάλαιο αναφέρεται στο αισθητήριο υπερήχων και τους σερβοκινητήρες. Γίνεται αναφορά στον τρόπο λειτουργίας τους, τα τεχνικά τους χαρακτηριστικά, τη συνδεσμολογία τους στη συγκεκριμένη κατασκευή, όπως επίσης και κάποια ιστορικά στοιχεία.
- Το τρίτο κεφάλαιο της εργασίας αναφέρεται στη διαδικασία περάτωσης της κατασκευής . Αναλύεται ο τρόπος λειτουργίας της κατασκευής, παρουσιάζονται φωτογραφίες από διάφορα στάδια κατασκευής όπως επίσης και προβλήματα που παρουσιάστηκαν κατά τη διάρκεια της κατασκευής.
- Το τέταρτο και τελευταίο κεφάλαιο περιέχει τον κώδικα της κατασκευής. Ο πρώτος κώδικας χρησιμοποιείται για τη κίνηση του σερβοκινητήρα και την συλλογή των δεδομένων από το αισθητήριο υπερήχων. Ο δεύτερος κώδικας χρησιμοποιείται για την παρουσίαση - «χαρτογράφηση» των δεδομένων .

**(ΟΝΟΜΑΤΑ-ΥΠΟΓΡΑΦΕΣ ΕΞΕΤΑΣΤΩΝ)**

**ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ**

# 1 . ΜΙΚΡΟΕΛΕΓΚΤΗΣ ARDUINO

## 1 . 1 Εισαγωγή

Ο μικροελεγκτής (microcontroller) είναι ένας τύπος επεξεργαστή, μια παραλλαγή ουσιαστικά του μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

Ο Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Ο Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών εφαρμογών αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing , Max/MSP, Pure Data, SuperCollider.

Πρόκειται για μια πρωτότυπη ηλεκτρονική πλατφόρμα διαμόρφωσης ανοικτού λογισμικού βασισμένη στο υλικό ενός μικροεπεξεργαστή, καθώς και σε κατάλληλο λογισμικό για τον προγραμματισμό του. Η χρήση του προορίζεται για σχεδιαστές , ανθρώπους που ασχολούνται με ηλεκτρονικά στον ελεύθερο χρόνο τους και κάθε ενδιαφερόμενο στη δημιουργία ψηφιακών εφαρμογών σε χαμηλό επίπεδο, χωρίς όμως να απαιτείται προγραμματισμός σε γλώσσα μηχανής.

Ο Arduino μπορεί να αισθανθεί το περιβάλλον προσλαμβάνοντας δεδομένα από μεγάλη ποικιλία αισθητήρων και μπορεί να επηρεάσει το χώρο με τον έλεγχο των φώτων, των μηχανών και άλλων ηλεκτρονικών συστημάτων. Ο μικροεπεξεργαστής στον πίνακα της πλατφόρμας είναι προγραμματισμένος στη χρήση ειδικής γλώσσας προγραμματισμού (γλώσσα του Arduino) και στο περιβάλλον ανάπτυξης Arduino. Τα προγράμματα στο περιβάλλον αυτό μπορούν να είναι αυτόνομα ή μπορούν αν επικοινωνήσουν με κατάλληλο λογισμικό με τον υπολογιστή.

### 1 . 2 Ιστορία του Arduino

Το 2005, στην Ivrea της Ιταλία (η ιστοσελίδα της εταιρείας υπολογιστών Olivetti), ένα έργο άρχισε να δημιουργείται, μια συσκευή για τον έλεγχο σχεδίων, χτισμένο από μαθητές με λιγότερα έξοδα σε σχέση με άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνην τη στιγμή. Από τον Μάιο του 2011, περισσότερα από 300.000 Arduino ήταν στην αγορά. Οι εφευρέτες Massimo Banzi και David Cuartielles, ονόμασαν το έργο τους Arduin of Ivrea. Το «Arduino» είναι επίσης ένα ιταλικό όνομα, που σημαίνει «γενναίος φίλος».

Η πλατφόρμα Arduino είναι μια παραγόμενη έκδοση της πλατφόρμας ανοικτού κώδικα Wiring Platform. Ο Κολομβιανός καλλιτέχνης και προγραμματιστής Hernando Barragán δημιούργησε συρμάτωση «Wiring» ως μια Μεταπτυχιακή διπλωματική εργασία στο Interaction Design Institute Ivrea υπό την εποπτεία του Massimo Banzi και του Casey Reas. Η Καλωδίωση «Wiring» βασίστηκε στην επεξεργασία και το ολοκληρωμένο περιβάλλον ανάπτυξης που είχε δημιουργηθεί από τον Casey Reas και τον Ben Fry.

Η πλατφόρμα Arduino χτίστηκε γύρω από το έργο της καλωδίωσης «Wiring» του Hernando Barragan. Η Καλωδίωση ήταν η διπλωματική εργασία του Hernando στο Interaction Design Institute Ivrea. Επρόκειτο να είναι μια ηλεκτρονική έκδοση της επεξεργασίας που χρησιμοποιήθηκε σε προγραμματιστικό περιβάλλον και ήταν η βάση για την σύνταξη επεξεργασίας. Η επεξεργασία σίγουρα δεν θα υπήρχε χωρίς τη γλώσσα προγραμματισμού Design By Numbers και τον John Maeda.,

### 1 . 3 Πλεονεκτήματα του Arduino σε σχέση με άλλους μικροελεγκτές

Υπάρχει πληθώρα άλλων μικροελεγκτών και αναπτυξιακών στο εμπόριο για να ασχοληθεί κάποιος. Ο Basic Stamp της Parallax, ο BX-24 της Netmedia, το Handyboard του MIT και πολλοί άλλοι όμοιας λειτουργικότητας. Όλα αυτά τα εργαλεία που προαναφέραμε είναι απλά και για τον αρχάριο χρήστη καθώς "κρύβουν" τις δύσκολες λεπτομέρειες της αρχιτεκτονικής και επιτρέπουν τον άμεσο προγραμματισμό του μικροελεγκτή, προσφέροντας τα πάντα σε ένα και μόνο "πακέτο" έτοιμο για χρήση. Ο Arduino διαφέρει από τους προηγούμενους γιατί απλοποιεί την διαδικασία να δουλεύει κάποιος με μικροελεγκτές. Κάποια πλεονεκτήματα που προσφέρει σε σχέση με άλλους μικροελεγκτές για χρήση από δασκάλους, μαθητές και άλλους hobbistες είναι τα παρακάτω:

- Οι πλακέτες του Arduino είναι εξαιρετικά φθηνές σε σχέση με άλλες πλατφόρμες μικροελεγκτών. Ειδικά δε μπορεί με τα σχηματικά που κυκλοφορούν στο Internet να

κατασκευάσει κάποιος την φθηνότερη εκδοχή ενός Arduino. Ωστόσο ακόμα και αν προμηθευτεί την έτοιμη (μονταρισμένη πλακέτα) αυτή θα κοστίσει το μέγιστο 50 Euro.

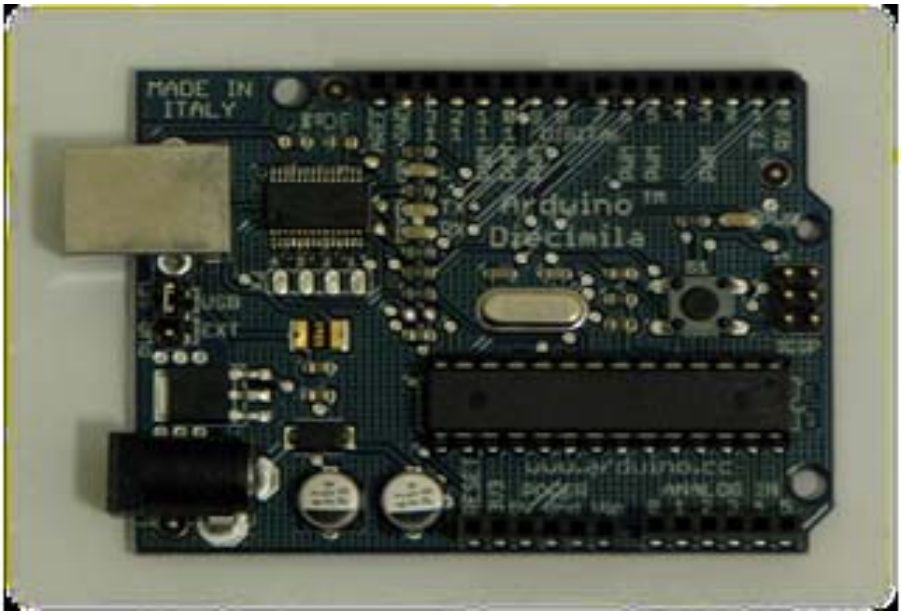
- Τρέχει σε διάφορα Λειτουργικά Συστήματα. Οι μηχανικοί λογισμικού, ανέπτυξαν το περιβάλλον προγραμματισμού του Arduino για Windows, Machintosh OSX και για λειτουργικά συστήματα Linux. Τα περισσότερα συστήματα ανάπτυξης Μικροελεγκτών περιορίζονται στα Windows.
- Απλό, ξεκάθαρο προγραμματιστικό περιβάλλον. Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχάριους, αλλά είναι ταυτόχρονα και ευέλικτο και για πιο προχωρημένους χρήστες.
- Ανοιχτού λογισμικού και λογισμικού που επεκτείνεται και παραμετροποιείται. Το software του Arduino διανέμεται με την μορφή εργαλείων ανοιχτού λογισμικού και είναι διαθέσιμο προς επέκταση για έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού του μπορεί να επεκταθεί διαμέσου των βιβλιοθηκών την C++ και οι άνθρωποι που θέλουν να ασχοληθούν περισσότερο με τους μικροελεγκτές μπορούν να μεταβούν από τον Arduino στην AVR C, που είναι για προγραμματισμό των Atmel Μικροελεγκτών και η γλώσσα στην οποία βασίστηκε το λογισμικό του Arduino. Ομοίως μπορεί κάποιος να προσθέσει κώδικα της AVR-C στο πρόγραμμα που έχει γράψει για τον Arduino του.
- Ανοιχτού Υλικού το οποίο μπορεί να επεκταθεί. Ο Arduino βασίζεται στους μικροελεγκτές της Atmel ATMEGA8 και ATMEGA168. Τα σχηματικά για τα αναπτυξιακά είναι κάτω από την άδεια της Creative Commons, επιτρέποντας σε έμπειρους σχεδιαστές να κατασκευάσουν το δικό τους αναπτυξιακό, εξελίσσοντας το ήδη υπάρχον χωρίς να έχουν νομικά προβλήματα. Η ακόμη καλύτερα όχι τόσο έμπειροι χρήστες μπορούν να επιδιώξουν την αντιγραφή και κατασκευή της πλακέτας σε ράστερ για να καταλάβουν την λειτουργία ενός Arduino.

### 1 . 4 Μοντέλα Arduino

Στην αγορά διατίθενται πολλά μοντέλα της πλατφόρμας και στην παρούσα εργασία χρησιμοποιήθηκε το Arduino Uno R3 το οποίο είναι και νεότερο τεχνολογικά. Παλαιότερα μοντέλα παρουσιάζουν κάποιες διαφορές μεταξύ τους και θα αναφερθούν παρακάτω με χρονολογική σειρά απ’ το νεότερο προς το παλαιότερο.

*Arduino Diecimila*

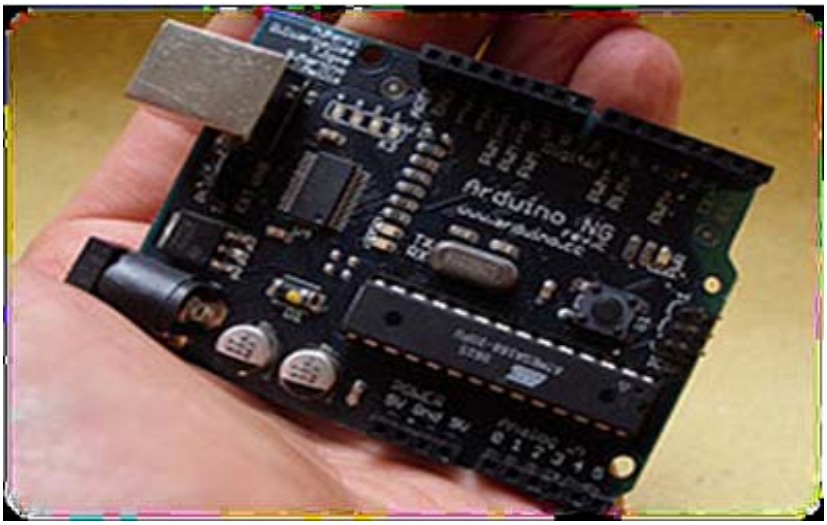
Το κύριο χαρακτηριστικό του Arduino Diecimila είναι ότι μπορεί να επαναρυθμιστεί από τον υπολογιστή χωρίς να πιεστεί το κουμπί reset που βρίσκεται στη πλατφόρμα. Το συγκεκριμένο μοντέλο έχει ένα ρυθμιστή τάσης χαμηλής ισχύος, ο οποίος μειώνει την κατανάλωση ενέργειας της πλατφόρμας όταν τροφοδοτείται από ηλεκτρονικό υπολογιστή ή μπαταρία. Παρέχει επίσης εισόδους για reset και για 3.3V και στην ψηφιακή είσοδο 13 βρίσκεται ενσωματωμένο led.




**Εικόνα 1-1** Η πλατφόρμα Arduino Diecimila

*Arduino NG Rev. C*

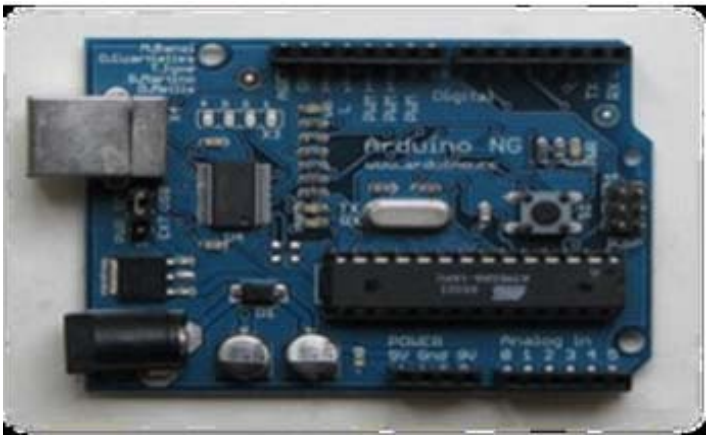
Το συγκεκριμένο μοντέλο δεν διαθέτει ενσωματωμένο led στην ψηφιακή είσοδο 13. Εντούτοις, διατίθεται 1000Ω αντίσταση στην είσοδο 13 και μπορεί να γίνει σύνδεση Led χωρίς εξωτερικό αντιστάτη.




**Εικόνα 1-2** Η πλατφόρμα Arduino NG Rev.C

*Arduino NG (Nuova Generazione)*

Το συγκεκριμένο μοντέλο χρησιμοποιεί τον USB – τμηματικό μετατροπέα FTDI FT232RL, ο οποίος απαιτεί λιγότερα εξωτερικά από τον FT232BM. Επίσης διαθέτει ενσωματωμένο led στην είσοδο 13, το οποίο μπορεί να παρεμποδίσει την επικοινωνία SPI.




**Εικόνα 1-3** Η πλατφόρμα Arduino NG

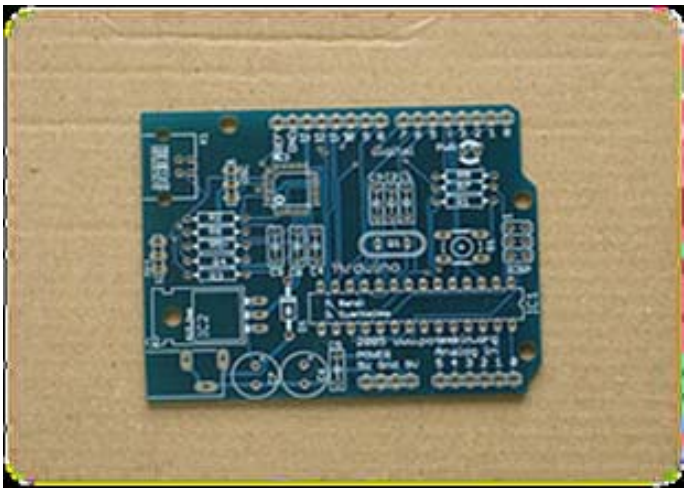
*Arduino Extreme*

Το συγκεκριμένο μοντέλο χρησιμοποιεί περισσότερα στοιχεία στην επιφάνεια, σε σχέση με τον Arduino USB και έχει θηλυκούς αποδέκτες στις εισόδους. Επίσης διαθέτει τα RX και TX LEDs, τα οποία υποδεικνύουν πότε στέλνονται τα δεδομένα στην πλατφόρμα ή από την πλατφόρμα αντίστοιχα.


**Εικόνα 1-4** Η πλατφόρμα Arduino Extreme

*Arduino USB*

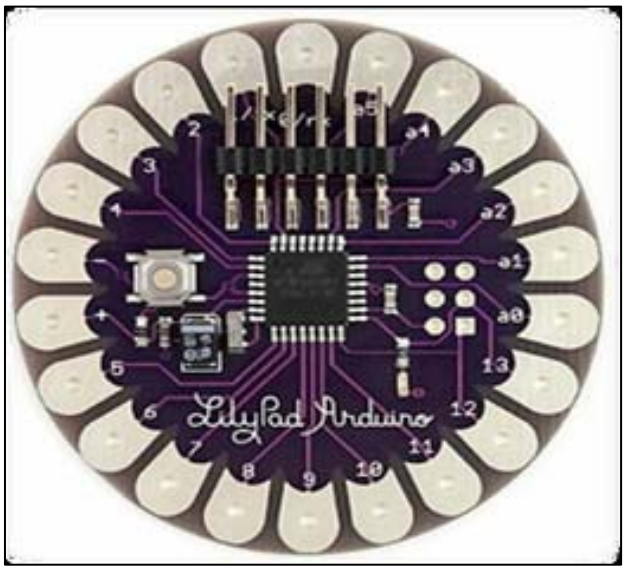
Το μοντέλο αυτό ήταν ο πρώτος πίνακας που ονομάστηκε Arduino. Αρχικά πουλήθηκαν σαν ανεξάρτητα κομμάτια που απαιτούσαν συναρμολόγηση. Η πρώτη έκδοση παρουσίαζε σφάλμα στη σύνδεση USB.



**Εικόνα 1-5** Η πλατφόρμα Arduino USB

*Arduino Lilypad*

Το αρχικό σχέδιο του μοντέλου είχε 10 ICSP εισόδους και χρησιμοποιούσε το εσωτερικό ρολόι του μικροεπεξεργαστή ATmega168, παρά έναν εξωτερικό ταλαντωτή. Επίσης, στο αρχικό σχέδιο χρησιμοποιούσε ως bootloader το bootloader NG.




**Εικόνα 1-6** Η πλατφόρμα Arduino Lilypad

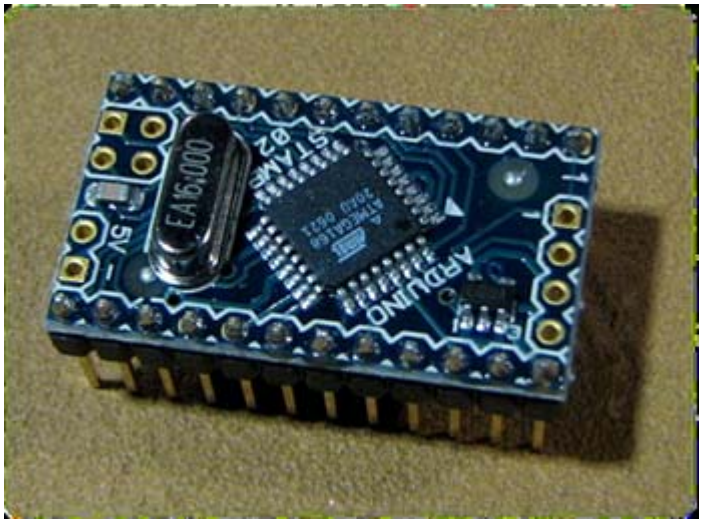
*Arduino Nano*

Το Arduino Nano περιέχει ένα τσιπ AT Mega 168 και FTDI USB , ενώ χαρακτηρίζεται ως ένα σχέδιο για τη χρήση με breadboard.


**Εικόνα 1-7** Η πλατφόρμα Arduino Nano

*Arduino Mini*

Το μοντέλο αυτός είναι ένας μικρός πίνακας μικροεπεξεργαστών που βασίζεται στον μικροεπεξεργαστή AT mega 168 και προορίζεται για τη χρήση με breadboards. Διαθέτει 14 ψηφιακές εισόδους – εξόδους (εκ των οποίων οι 6 μπορούν να χρησιμοποιηθούν ως PWM έξοδοι), 8 αναλογικές εισόδους και ένα ταλαντωτή κρυστάλλου των 16 MHz. Υπάρχει η δυνατότητα προγραμματισμού είτε με το Mini USB με τη χρήση μετατροπέα.




**Εικόνα 1-8** Η πλατφόρμα Arduino Mini

*Arduino Duemilanove*

Το Arduino Duemilanove (2009) είναι μια πλατφόρμα βασισμένη στον μικροεπεξεργαστή ATmega 168. Έχει 14 ψηφιακές εισόδους και εξόδους (εκ των οποίων οι έξι μπορούν να

χρησιμοποιηθούν ως αποτελέσματα PWM), 6 αναλογικές εισόδους , ένας ταλαντωτής κρυστάλλου 16 MHz, μια σύνδεση USB, μια είσοδο παροχής ενέργειας, ένα διασυνδετικό αγωγό ICSP , και ένα κουμπί αναστοιχειοθέτησης. Περιέχει όλα όσα απαιτούνται για να υποστηρίξουν τη λειτουργία του μικροεπεξεργαστή. Μπορεί να συνδεθεί με έναν υπολογιστή διαμέσω ενός καλωδίου USB ή να τροφοδοτηθεί με έναν προσαρμογέα ρεύμα-συνεχές ή με μία μπαταρία.


**Εικόνα 1-9** Η πλατφόρμα Arduino Duemilanove

### 1 . 5 Ο μικροελεγκτής Arduino Uno

Ο Arduino Uno είναι ο τύπος του Arduino που χρησιμοποιούμε για την εργασία αυτή. Ακολουθεί αναλυτική περιγραφή σχετικά με την αρχιτεκτονική του, τα μέρη απ' τα οποία αποτελείται, τον τρόπο επικοινωνίας και το πρόγραμμα με το οποίο μπορούμε να τον προγραμματίσουμε.

Η Αρχιτεκτονική του Arduino Uno

Όσον αφορά την αρχιτεκτονική του ,το Arduino είναι μια πλατφόρμα βασισμένη στον ATMEGA328. Περιέχει 14 ψηφιακούς ακροδέκτες I/O εισόδου-εξόδου, 6 αναλογικές εισόδους και 6 ψηφιακές εξόδους έναν ταλαντωτή των 16 MHz, υποδοχή σύνδεσης με USB καλώδιο, υποδοχή σύνδεσης με ρεύμα, δυνατότητα εντός του κυκλώματος σειριακού προγραμματισμού-ICSP (In-Circuit Serial Programming) και ένα κουμπί επανεκκίνησης (reset) σε περίπτωση που βραχυκυκλώσει χωρίς την θέλησή μας. Εμπεριέχει όλα όσα χρειάζονται για την υποστήριξη ενός μικροελεγκτή. Το μόνο που απαιτείται είναι η σύνδεσή του με έναν ηλεκτρονικό υπολογιστή μέσω ενός καλωδίου USB ή τροφοδοσία του μέσω ενός μετασχηματιστή τάσης για τη μετατροπή μεταξύ εναλλασσόμενου και ασυνεχούς ρεύματος (adaptor AC-to-DC-Alternating Current-to-Direct Current) ή μιας μπαταρίας.

Ακροδέκτες Ρεύματος Οι Ακροδέκτες ρεύματος αποτελούνται από (νούμερο 10 στην παρακάτω εικόνα 12)

**3.3V**: Μπορεί να τροφοδοτήσει τα εξαρτήματα που συνδέονται εκεί με τάση 3.3V. Η τάση αυτή δεν προέρχεται από την εξωτερική τροφοδοσία αλλά παράγεται από τον ελεγκτή Serial-over-USB και έτσι η μέγιστη ένταση που μπορεί να παρέχει είναι μόλις 50mA.

**5V**: Μπορεί να τροφοδοτήσει τα εξαρτήματα με τάση 5V. Ανάλογα με τον τρόπο τροφοδοσίας του ίδιου του Arduino, η τάση αυτή προέρχεται είτε άμεσα από την θύρα USB (που ούτως ή άλλως λειτουργεί στα 5V), είτε από την εξωτερική τροφοδοσία αφού αυτή περάσει από ένα ρυθμιστή τάσης για να την «φέρει» στα 5V.

**GND**: 2 γειώσεις

**VIN**: Σε συνδυασμό με τον ακροδέκτη γείωσης δίπλα του, μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας του Arduino, στην περίπτωση που δεν βολεύει να χρησιμοποιηθεί η υποδοχή του φις των 2.1mm. Αν όμως υπάρχει ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, μπορεί αυτό το pin να χρησιμοποιηθεί για να τροφοδοτηθούν εξαρτήματα με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

**RESET:** όταν γειωθεί (σε οποιοδήποτε από τα 3 pin με την ένδειξη GND που υπάρχουν στον Arduino) έχει ως αποτέλεσμα την επανεκκίνηση του Arduino.

Αναλογικοί Ακροδέκτες

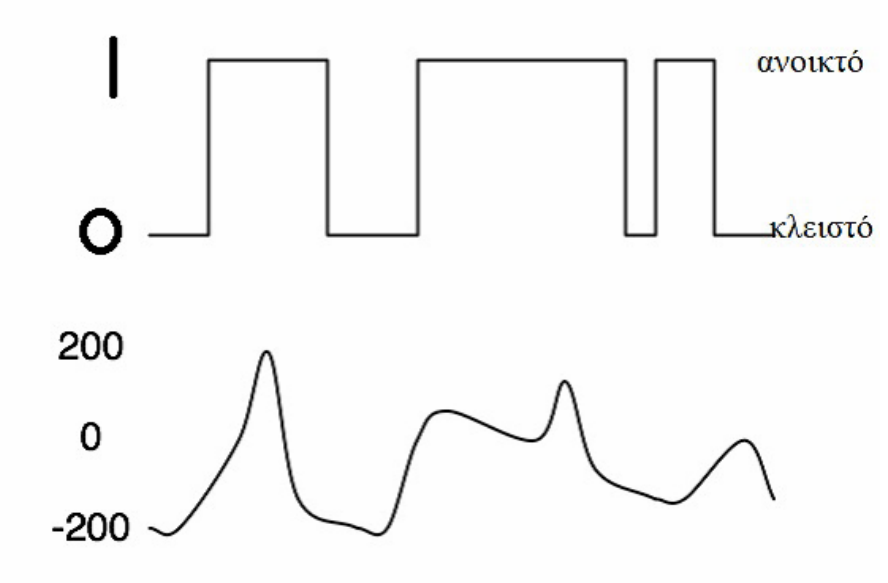
Στους αναλογικούς ακροδέκτες (νούμερο 11 στην εικόνα 12) θα συνδεθούν το αναλογικά συστήματα, όπως τα ποτενσιόμετρα και άλλοι αναλογικοί αισθητήρες. Αν και οι ψηφιακές είσοδοι / έξοδοι λειτουργούν μόνο με 0 και 1 τιμές, οι αναλογικές είσοδοι λειτουργούν με τιμές 0 έως 1023. Στην κάτω πλευρά του Arduino, με τη σήμανση ANALOG IN, υπάρχει μια ακόμη σειρά από 6 pins, αριθμημένα από το 0 ως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένος στον μικροελεγκτή. Για παράδειγμα, μπορεί να τροφοδοτηθεί ένα από αυτά με μια τάση η οποία μπορεί να μεταβάλλεται με ένα ποτενσιόμετρο από 0 V ως μια τάση αναφοράς Vref η οποία, αν δε γίνει κάποια αλλαγή είναι προρυθμισμένη στα 5V. Τότε, μέσα από το πρόγραμμά μπορεί να «διαβαστεί» η τιμή του ακροδέκτη ως ένας ακέραιος αριθμός ανάλυσης 10-bits, από 0 (όταν η τάση στο pin είναι 0V) μέχρι 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V, ή σε όποια τιμή τάσης μεταξύ 2V και 5V τροφοδοτώντας

εξωτερικά με αυτή την τάση τον ακροδέκτη με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτηθεί ο ακροδέκτης AREF με 3.3V και στην συνέχεια διαβαστεί κάποιος ακροδέκτης αναλογικής εισόδου στον οποίο εφαρμόζεται τάση 1.65V, ο Arduino θα επιστρέψει την τιμή 512. Τέλος, καθένας από τους 6 αυτούς ακροδέκτες, με κατάλληλη εντολή μέσα από το πρόγραμμα μπορεί να μετατραπεί σε ψηφιακό pin εισόδου/εξόδου όπως τα 14 που βρίσκονται στην απέναντι πλευρά και τα οποία περιγράφηκαν πριν. Σε αυτή την περίπτωση τα pins μετονομάζονται από 0~5 σε 14~19 αντίστοιχα.

Ψηφιακοί Ακροδέκτες

Ψηφιακοί ακροδέκτες (νούμερο 13 στην εικόνα 12) μπορεί να λειτουργήσουν ως εισροές ή εκροές και να οριστεί πώς θα λειτουργήσουν με την κατάλληλη εντολή pinMode ().

Οι πείροι που έχουν το "~" μπροστά από τα νούμερα είναι PWM (Pulse Width Modulation) έξοδοι, και μπορούν λειτουργήσουν ως αναλογικές έξοδοι με την εντολή analogWrite.




**Εικόνα 1-10** Παρουσίαση ψηφιακού σήματος

Τα pins 0 και 1 λειτουργούν ως RX και TX της σειριακής όταν το πρόγραμμα ενεργοποιεί την σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμά στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένας δεύτερος Arduino στο δικό του pin 1). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά ενεργοποιηθεί η σειριακή επικοινωνία χάνονται 2 ψηφιακές είσοδοι/έξοδοι.

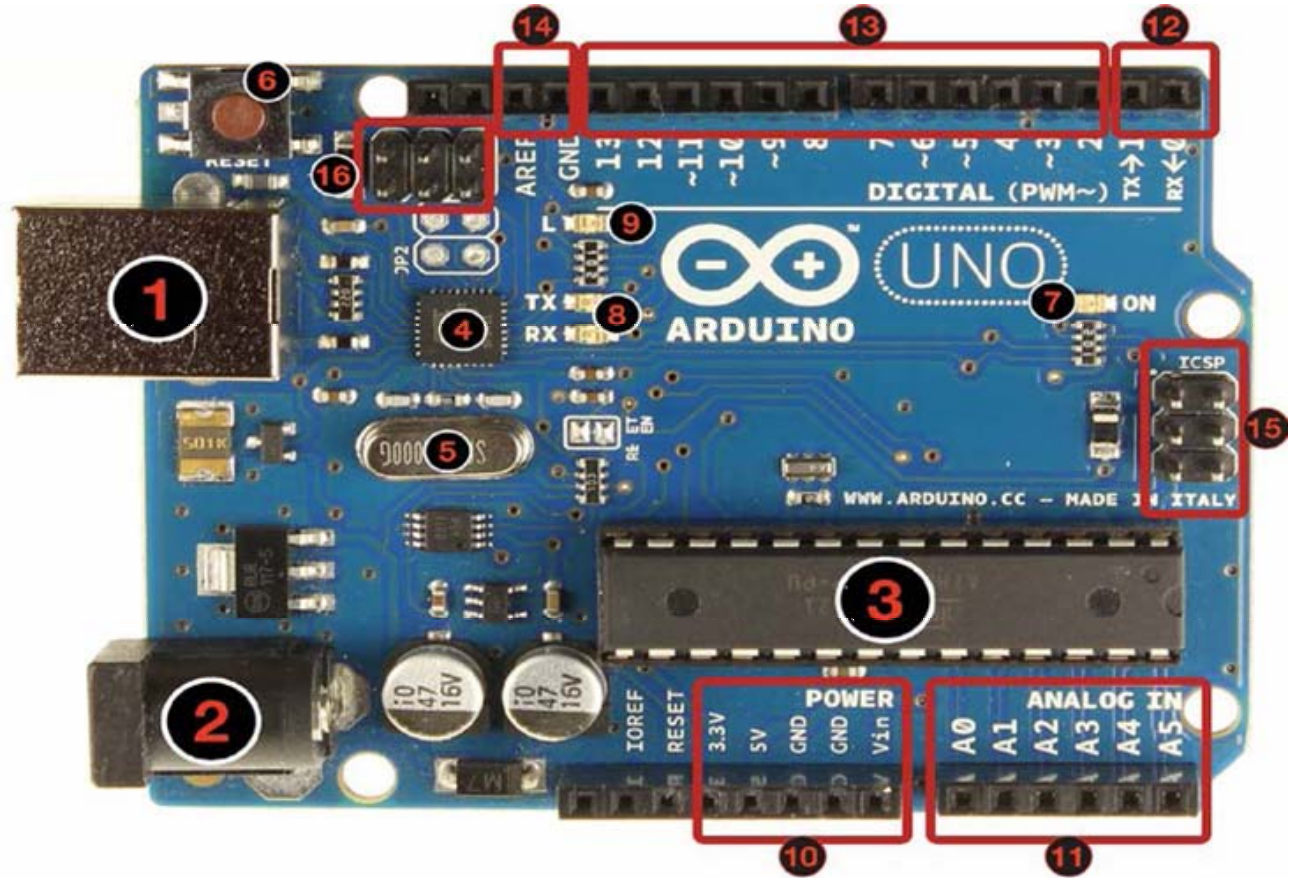
Τα pins 2 και 3 λειτουργούν και ως εξωτερικά interrupts (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορούν να ρυθμιστούν μέσα από το πρόγραμμα ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η

κανονική ροή του προγράμματος σταματάει *άμεσα* και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupts είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

Τα pins 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορεί να συνδεθεί λόγου χάρη ένα LED σε κάποιο από αυτά τα pins και να ελεγχθεί πλήρως η φωτεινότητά του με ανάλυση 8bits (256 καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο) αντί να υπάρχει απλά η δυνατότητα αναμμένο-σβηστό που παρέχουν οι υπόλοιπές ψηφιακές έξοδοι. Είναι σημαντικό ότι το PWM δεν είναι πραγματικά αναλογικό σύστημα και ότι θέτοντας στην έξοδο την τιμή 127, δεν σημαίνει ότι η έξοδος θα δίνει 2.5V αντί της κανονικής τιμής των 5V, αλλά ότι θα δίνει ένα παλμό που θα εναλλάσσεται με μεγάλη συχνότητα και για ίσους χρόνους μεταξύ των τιμών 0 και 5V.

Χαρακτηριστικά Arduino Uno

- Μικροελεγκτής ATMEGA328
- Τάση λειτουργίας 5V
- Τάση εισόδου (συνίσταται) 7-12V
- Όρια τάσης εισόδου 6-20V
- Ψηφιακοί ακροδέκτες I/O 14(6 εκ των οποίων PWM έξοδο)
- Αναλογικοί ακροδέκτες εισόδου 6
- Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη 40mA
- Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V 50mA
- Μνήμη flash 32KB(ATMEGA328)
- Μνήμη SRAM 2KB(ATMEGA328)
- Μνήμη EEPROM 1KB(ATMEGA328)
- Ταχύτητα ρολογιού 16MHz.




**Εικόνα 1-11** Παρουσίαση Arduino Uno

1. USB υποδοχή
2. Υποδοχή τροφοδοσίας
3. Επεξεργαστής
4. chip επικοινωνίας
5. 16 MHz κρυστάλλου
6. κουμπί επαναφοράς
7. Led ON
8. TX / NX Leds
9. Led
10. ακροδέκτες ρεύματος
11. Αναλογικές Είσοδοι
12. TX και RX ακροδέκτες
13. Ψηφιακές είσοδοι / έξοδοι μπροστά από τους αριθμούς είναι για PWM εξόδους.
14. Γείωση και AREF ακροδέκτες
15. ICSP για ATmega328
16. ICSP για διασύνδεση USB

Ο αριθμός των ακίδων (ψηφιακών in / out και αναλογικών ) ποικίλλουν ανάλογα με το μοντέλο.

**1 . 6 Ενσωματωμένα LED και κουμπιά**

Πάνω στην πλακέτα του Arduino υπάρχει ένας διακόπτης micro-switch και 4 μικροσκοπικά LEDs επιφανειακής στήριξης. Η λειτουργία του διακόπτη (που έχει την σήμανση RESET) και του ενός LED με την σήμανση POWER είναι μάλλον προφανής. Τα δύο LEDs με τις σημάνσεις TX και RX, χρησιμοποιούνται ως ένδειξη λειτουργίας του σειριακού interface, καθώς ανάβουν όταν ο Arduino στέλνει ή λαμβάνει (αντίστοιχα) δεδομένα μέσω USB. Τα LEDs αυτά ελέγχονται από τον ελεγκτή Serial-over-USB και συνεπώς δεν λειτουργούν όταν η σειριακή επικοινωνία γίνεται αποκλειστικά μέσω των ψηφιακών pin 0 και 1. Τέλος, υπάρχει το LED με την σήμανση L. Η βασική δοκιμή λειτουργίας του Arduino είναι να αναβοσβήνει ένα LED. Για να μπορεί να γίνει αυτό από την πρώτη στιγμή, χωρίς να συνδεθεί τίποτα πάνω στον Arduino, οι κατασκευαστές του σκέφτηκαν να ενσωματώσουν ένα LED στην πλακέτα, το οποίο σύνδεσαν στο ψηφιακό pin 13. Έτσι, ακόμα και αν δεν έχει συνδεθεί τίποτα πάνω στο φυσικό pin 13, αναθέτοντάς του την τιμή HIGH μέσα από το πρόγραμμα, θα ανάψει αυτό το ενσωματωμένο LED.

**1 . 7 Επικοινωνία**

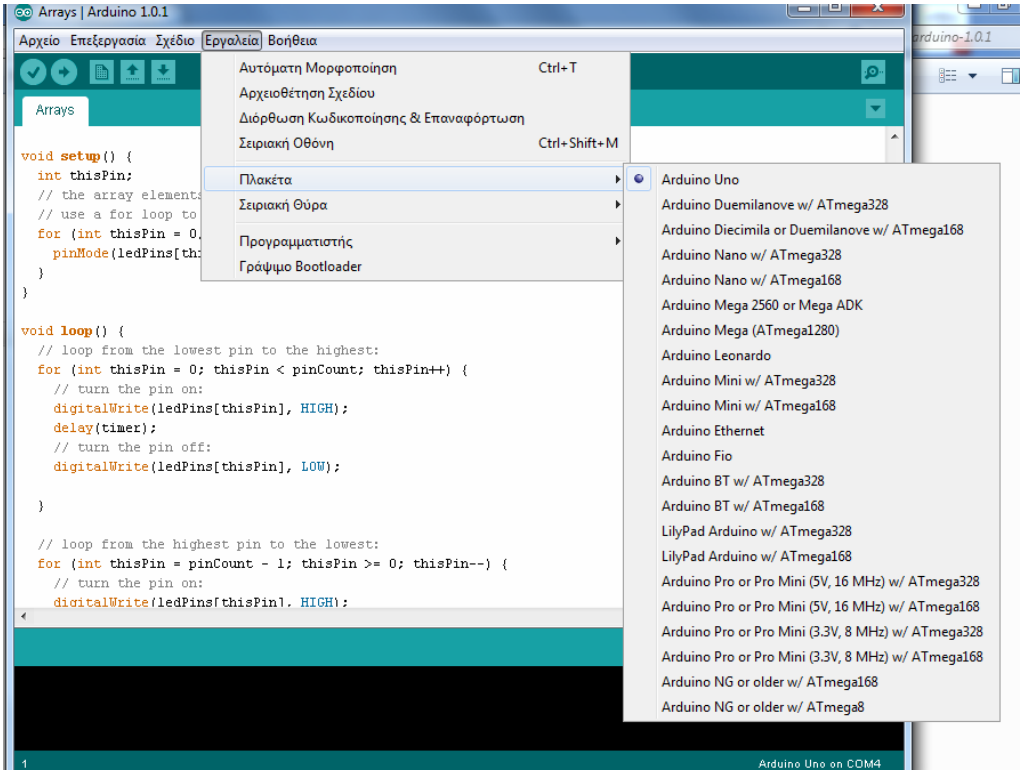
Ο Arduino Uno έχει διάφορους τρόπους για την επικοινωνία με έναν υπολογιστή, έναν άλλο Arduino ή άλλους μικροεπεξεργαστές. Οι ATmega168 και ATmega328 μικροεπεξεργαστές παρέχουν UART TLL (5V) σειριακή επικοινωνία , η οποία είναι διαθέσιμη στις ψηφιακές εισόδους 0(RX) και 1(TX). Ένα FTDI FT232RL στα κανάλια της πλατφόρμας, αυτή η σειριακή επικοινωνία με USB και οι οδηγοί FTDI (οι οποίοι συμπεριλαμβάνονται με το λογισμικό Arduino) παρέχουν μια εικονική είσοδο COM του λογισμικού στον υπολογιστή. Το λογισμικό Arduino περιλαμβάνει ένα σειριακό όργανο ελέγχου που επιτρέπει στα απλά δεδομένα κειμένου να αποσταλούν προς αλλά και από την πλατφόρμα Arduino. Τα RX και TX LEDs στην πλατφόρμα θα αναβοσβήσουν όταν διαβιβάζεται το στοιχείο μέσω του τσιπ FTDI και της σύνδεσης USB στον υπολογιστή (αλλά όχι για την σειριακή επικοινωνία των εισόδων 0 και 1). Η Software Serial βιβλιοθήκη επιτρέπει την σειριακή επικοινωνία σχετικά με οποιαδήποτε ψηφιακή είσοδο του UNO.

Οι ATmega168 και ATmega328 μικροεπεξεργαστές επίσης υποστηρίζουν την I2C (TWI) και SPI επικοινωνία . Το λογισμικό Arduino περιλαμβάνει βιβλιοθήκη Wire για να απλοποιηθεί η χρήση της I2C επικοινωνίας. Για να χρησιμοποιηθεί η SPI επικοινωνία, χρειάζεται η ανάγνωση και κατανόηση των οδηγιών λειτουργίας των ATmega168 ή ATmega328 μικροεπεξεργαστών.

### 1 . 8 Arduino IDE – σύνδεση με τον υπολογιστή

Η διαχείριση του Arduino από τον υπολογιστή παρέχεται από ένα IDE λογισμικό Arduino, την τελευταία έκδοση του οποίου μπορεί κανείς να κατεβάσει από το επίσημο site για καθένα από τα τρία δημοφιλέστερα λειτουργικά συστήματα. Το Arduino IDE είναι βασισμένο σε Java και συγκεκριμένα παρέχει:

- ένα πρακτικό περιβάλλον για την συγγραφή των προγραμμάτων (τα οποία ονομάζονται sketch στην ορολογία του Arduino) με συντακτική χρωματική σήμανση
- αρκετά έτοιμα παραδείγματα
- μερικές έτοιμες βιβλιοθήκες για προέκταση της γλώσσας και για να χειρίζεται κανείς εύκολα μέσα από τον κώδικά τα εξαρτήματα που συνδέονται στον Arduino
- ένα serial monitor που παρακολουθεί τις επικοινωνίες της σειριακής (USB), αναλαμβάνει να στείλει αλφαριθμητικά της επιλογής του χρήστη στο Arduino μέσω αυτής και είναι ιδιαίτερα χρήσιμο για το debugging των sketch σας
- και την επιλογή να ανεβεί το μεταγλωττισμένο sketch στο Arduino




**Εικόνα 1-12** Περιβάλλον Arduino Ide

### 1 . 9 Γλώσσα Προγραμματισμού

Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc. Λόγω της καταγωγής της από την C, στην γλώσσα του Arduino μπορεί κανείς να χρησιμοποιήσετε ουσιαστικά τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές όπως και στην C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino. Οι πιο σημαντικές από αυτές επεξηγούνται στον πίνακα που ακολουθεί:

| Όρισμα | Είδος | Τύπος | Παράμετροι | Περιγραφή |
|---|---|---|---|---|
| LOW | Σταθερά | int | - | Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false. |
| HIGH | Σταθερά | int | - | Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true. |
| INPUT | Σταθερά | int | - | Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false. |
| OUTPUT | Σταθερά | int | - | Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true. |
| pinMode | Εντολή | - | (*pin*, *mode*) | Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode* (INPUT ή OUTPUT αντίστοιχα). |
| digitalWrite | Εντολή | - | (*pin*, *pinstatus*) | Θέτει την κατάσταση *pinstatus* (HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*. |
| digitalRead | Συνάρτηση | int | (*pin*) | Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου. |
| analogReference | Εντολή | - | (*type*) | Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς ($V_{ref}$) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα) |

| | | | | |
|---|---|---|---|---|
| analogRead | Συνάρτηση | int | (*pin*) | Επιστρέφει έναν ακέραιο από 0 εώς 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως $V_{ref}$. |
| analogWrite | Εντολή | - | (*pin*, *value*) | Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο). |
| millis | Συνάρτηση | unsigned long | () | Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν. |
| delay | Εντολή | - | (*time*) | Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay. |
| attachInterrupt | Εντολή | - | (*interrupt*, *function*, *triggermode*) | Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*:<br>• LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW)<br>• RISING (όταν από LOW γίνει HIGH)<br>• FALLING (όταν από HIGH γίνει LOW)<br>• CHANGE (όταν αλλάξει κατάσταση γενικά) |
| detachInterrupt | Εντολή | - | (*interrupt*) | Απενεργοποιεί το συγκεκριμένο *interrupt*. |
| noInterrupts | Εντολή | - | () | Σταματά προσωρινά την λειτουργία όλων των interrupt |
| interrupts | Εντολή | - | () | Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts. |
| Serial.begin | Μέθοδος κλάσης | - | (*datarate*) | Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud) |
| Serial.println | Μέθοδος κλάσης | - | (*data*) | Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό. |

### 1 . 10 Βιβλιοθήκες

Μαζί με την εγκατάσταση του προγράμματος οδήγησης παρέχονται κάποιες βιβλιοθήκες και αυτές είναι οι εξής:

- EEPROM : Ανάγνωση και εγγραφή σε μόνιμη αποθήκευση (εγγραφή στη μνήμη της πλατφόρμας).
- Ethernet : Εφαρμόζεται για τη σύνδεση με το Διαδίκτυο χρησιμοποιώντας το Arduino Ethernet Shield
- Firmata: Χρησιμοποιείται στην επικοινωνία με τις εφαρμογές του υπολογιστή, που χρησιμοποιείται ένα τυποποιημένο τμηματικό πρωτόκολλο
- Liquid Crystal: Εφαρμόζεται για τον έλεγχο επιδείξεων υγρού κρυστάλλου (LCDs)
- Servo : Εφαρμόζεται για τον έλεγχο των servo μηχανών
- Software Serial : Εφαρμόζεται για την τμηματική ανακοίνωση σχετικά με οποιεσδήποτε ψηφιακές εισόδους
- Stepper : Εφαρμόζεται για τον έλεγχο των stepper κινητήρων
- Wire : Η διεπαφή δύο καλωδίων (TWI/I2C) έχει την δυνατότητα αποστολής και λήψης των δεδομένων εκτός των συσκευών και των αισθητήρων.

### 1 . 11 Το περιβάλλον του Processing

Η Processing είναι μια ανοικτή και δωρεάν γλώσσα προγραμματισμού - περιβάλλον για δημιουργικό προγραμματισμό. Χρησιμοποιείται για δημιουργία εικόνων, animation και διαδραστικών προγραμμάτων. Δημιουργήθηκε το 2001 και από τον Ben Fry και τον Casey Reas όταν και οι δύο ήταν φοιτητές του John Maeda στο MIT Media LabH Processing είναι γραμμένη σε Java (ουσιαστικά αποτελεί μια διεπαφή (interface) της Java). Σήμερα διδάσκεται σε πολλά πανεπιστήμια (New York University's graduate ITP program, UCLA, Lincoln Public Schools στη Nebraska , και Phoenix Country Day School στην Arizona) και αριθμεί δεκάδες χιλιάδες χρήστες σε όλο τον κόσμο. Αυτό είχε σαν αποτέλεσμα να δημιουργηθούν πολλές βιβλιοθήκες μέσω των οποίων διευρύνεται το πεδίο εφαρμογής της συνεχώς.

### 1 . 12 Βιβλιοθήκες του Processing

Οι παρακάτω βιβλιοθήκες περιλαμβάνονται στο λογισμικό του Processing . Για να χρησιμοποιήσουμε μια απ τις παρακάτω βιβλιοθήκες πηγαίνουμε στο μενού Sketch-> Import

Library και επιλέγουμε τη βιβλιοθήκη που θέλουμε να συμπεριλάβουμε στο πρόγραμμα μας. Οι βιβλιοθήκες είναι ανοιχτού κώδικα και διανέμονται με το Processing.

- *Video* : Η βιβλιοθήκη του βίντεο επιτρέπει στο Processing να αναπαράγει αρχείο βίντεο, να χρησιμοποιήσει βίντεο από μια κάμερα , και να δημιουργήσει βίντεο από ένα πρόγραμμα που τρέχει στον υπολογιστή. Τα βίντεο μπορούν να ‘’φορτωθούν’’ από βιντεοκάμερες που συνδέονται με υπολογιστή μέσω USB καλωδίου, από οποιοδήποτε μέσο αποθήκευσης καθώς και από το διαδίκτυο.
- *PDF Export* : Η συγκεκριμένη βιβλιοθήκη μας επιτρέπει να δημιουργήσουμε αρχεία PDF κατευθείαν μέσω του Processing. Μέσα στα αρχεία μπορούμε να συμπεριλάβουμε γραφικά σ’ οποιαδήποτε κλίμακα και σε πολύ υψηλή ανάλυση. Μπορεί να μετατρέψει δεδομένα 3D σε 2D. Για να εξάγουμε όμως δεδομένα 3D πρέπει να συμπεριλάβουμε και τη βιβλιοθήκη DFX.
- *Minim* : Είναι μία βιβλιοθήκη ήχου που μας επιτρέπει να χρησιμοποιήσουμε αρχεία ήχου όπως το mp3 στο πρόγραμμα μας. Αυτό το καταφέρνει χρησιμοποιώντας το JavaSound API , το Javazoom’s και το MP3SPI .
- *Network:* Επιτρέπει το read (διάβασμα) και write (εκτύπωση) δεδομένων με κάποια συσκευή, επικοινωνώντας μέσω του διαδικτύου. Επιτρέπει τη δημιουργία client –server . Ο server μπορεί να συνδεθεί σε μία λίστα από clients και να διαβάσει δεδομένα. Ο client μπορεί να διαβάσει και να εκτυπώσει δεδομένα στο server.
- *DXF Export* : Επιτρέπει τη δημιουργία DFX αρχείων(AutoCad) . Μπορούμε να σχεδιάσουμε σχήματα χρησιμοποιώντας τις συναρτήσεις beginRaw() και endRaw(). Όλα τα σχήματα που δημιουργούνται αποτελούνται από πολλά μικρά τρίγωνα.
- *Arduino* : Επιτρέπει την επικοινωνία και τον έλεγχο του μικροελεγκτή Arduino.
- *Serial* : Επιτρέπει το read and write δεδομένων από εξωτερικές συσκευές ένα byte κάθε φορά. Επιτρέπει δύο υπολογιστές να πάρουν και να στείλουν δεδομένα. Έχει ευελιξία καθώς μπορεί να χρησιμοποιήσει μικροελεγκτές σαν συσκευές εισόδου-εξόδου.
- *Netscape.JavaScript* : Επιτρέπει τη διασύνδεση προγραμμάτων Java με προγράμματα Processing.

Εκτός απ τις βιβλιοθήκες που αναφέραμε προηγουμένως και υπάρχουν ήδη εγκατεστημένες μαζί με το Processing , υπάρχουν και κάποιες άλλες βιβλιοθήκες οι οποίες είναι πιο εξεζητημένες . Αυτές για να τις χρησιμοποιήσουμε θα πρέπει να πάμε στο μενού Sketch-> Import Library-> Add Library , να τις κατεβάσουμε και να τις εγκαταστήσουμε.

**1 . 13 Εντολές που χρησιμοποιήθηκαν**

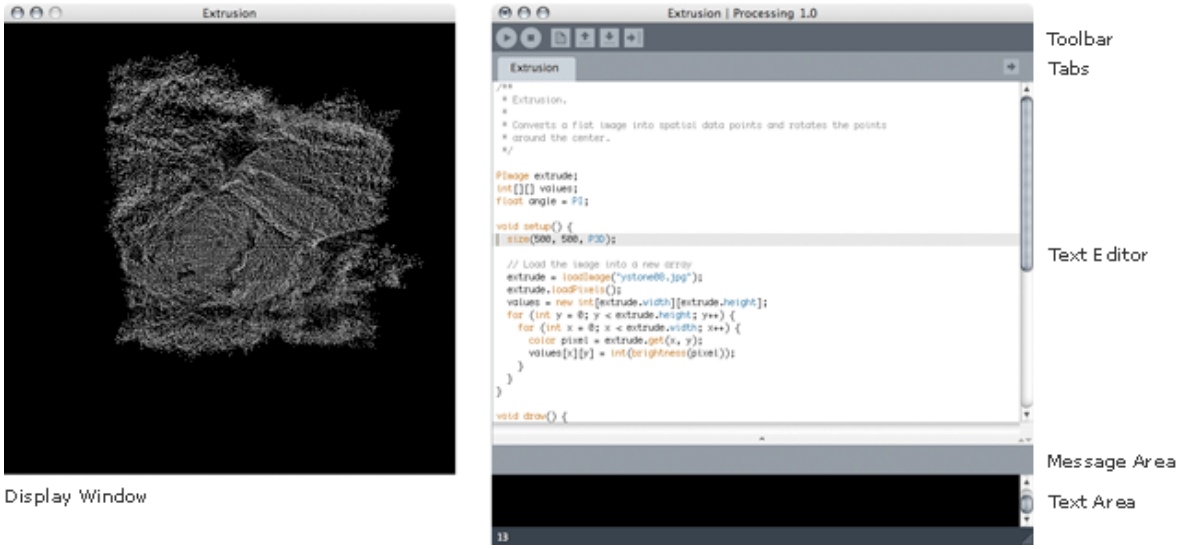
Παρακάτω θα αναφέρουμε τις εντολές που χρησιμοποιήσαμε στο πρόγραμμά μας και θα εξηγήσουμε αναλυτικά τι κάνει η κάθε μία . Για περισσότερες πληροφορίες όσον αφορά τις εντολές, υπάρχει αναλυτική περιγραφή όλων των εντολών του Processing στην επίσημη ιστοσελίδα της (http://processing.org/reference/) .

- size() : Καθορίζει το μέγεθος του παραθύρου της οθόνης σε pixel, όπου θα εμφανίσουμε τα δεδομένα μας. Αν δεν ρυθμιστεί τότε παίρνει αυτόματα την τιμή 100x100 pixels.
- Background() : Ρυθμίζει τον φόντο του παραθύρου της οθόνης όπου εμφανίζονται τα αποτελέσματα. Μπορεί να δεχτεί και εικόνα σαν φόντο, σ' αυτή τη περίπτωση όμως θα πρέπει να προσέξουμε η ανάλυση της εικόνας να ταιριάζει με την ανάλυση της οθόνης αποτελεσμάτων .
- Fill(): Ρυθμίζει το χρώμα γεμίσματος των σχημάτων. Δέχεται τιμές rgb με τη μορφή fill(r,g,b).
- rectMode(): Ρυθμίζει αν το τετράγωνο θα βρίσκεται στο κέντρο ή στη γωνία του παραθύρου δεχόμενη τις τιμές CORNER , CENTER , RADIUS.
- rect(): Σχεδιάζει ορθογώνιο παραλληλόγραμμο, δέχεται 4 τιμές με τη μορφή rect(a,b,c,d) όπου a και b είναι οι x,y συντεταγμένες του σχήματος και c,d οι πλευρές του σχήματος. Αν θέλουμε να έχουμε τις γωνίες στρογγυλεμένες τότε βάζουμε και μία παράμετρο ακόμα η οποία δείχνει το βαθμό στρογγυλοποίησης της γωνίας. Αν θέλουμε σε κάθε γωνία διαφορετική στρογγυλοποίηση τότε χρησιμοποιούμε 8 τιμές όπου οι 4 τελευταίες είναι για τη κάθε γωνία , ξεκινώντας από τη πάνω αριστερή γωνία και ακολουθώντας τη φορά του ρολογιού.
- stroke(): Ρυθμίζει το χρώμα του περιγράμματος των σχημάτων.
- line(): Σχεδιάζει γραμμή παίρνοντας 4 τιμές με τη μορφή line(x1,y1,x2,y2), όπου x1,y1 είναι οι συντεταγμένες της αρχής της γραμμής και x2y2 οι συντεταγμένες του τέλους της γραμμής.
- noStroke(): Αφαιρεί το περίγραμμα απ' το σχήμα.
- Ellipse(): Σχεδιάζει μια έλλειψη παίρνοντας 4 τιμές με τη μορφή ellipse(a,b,c,d) όπου a και b οι x-y συντεταγμένες του σχήματος , c-d πλάτος και ύψος σχήματος.
- beginShape()-endShape() : Ξεκινάμε να εγγράφουμε τις κορυφές για ένα σχήμα με την εντολή beginShape() και σταματάμε με την εντολή endShape().
- vertex(): Σχεδιάζει τις κορυφές συγκεκριμένων συντεταγμένων.

- cos(): Μας δίνει το συνημίτονο μιας γωνίας
- sin(): Μας δίνει το συνημίτονο μιας γωνίας
- radians(): Μετατρέπει τις μοίρες σε ακτίνια.
- text() : Εισάγει κείμενο παίρνοντας στην απλή μορφή κειμένου 3 τιμές με τη μορφή text(“a”,b,c) όπου το a είναι το κείμενο και b-c είναι οι συντεταγμένες του κειμένου.
- strokeweight() : Ρυθμίζει το πάχος του περιγράμματος παίρνοντας μια αριθμητική τιμή.

### 1 . 14 Processing IDE

Το περιβάλλον προγραμματισμού του Processing αποτελείται από ένα κειμενογράφο όπου γράφουμε τον κώδικα μας , ένα πεδίο μηνυμάτων, καρτέλες για να έχουμε ανοιχτά όσα αρχεία χρειαζόμαστε, μια γραμμή εργαλείων , και μενού.

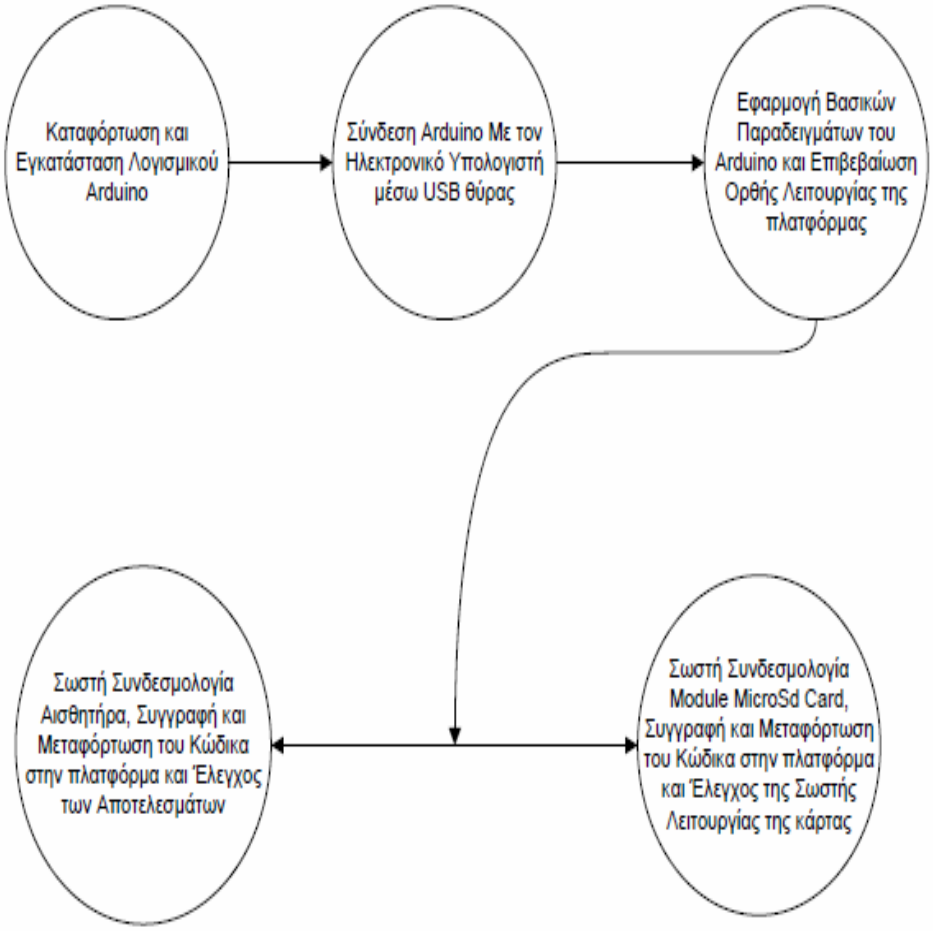



**Εικόνα 1-12** Περιβάλλον Processing Ide

Στο περιβάλλον του Processing τα προγράμματα που δημιουργούνται ονομάζονται sketch αποθηκεύονται στο sketchbook που είναι φάκελος μέσα στον υπολογιστή μας. Για τα προγράμματα μας μπορούμε να δημιουργήσουμε εκτελέσιμο αρχείο πηγαίνοντας στο μενού File->Export Application όπου επιλέγουμε το λειτουργικό σύστημα στο οποίο θα τρέχει το πρόγραμμά μας και πατάμε Export.

Το Processing έχει 3 διαφορετικούς τρόπους προγραμματισμού(modes) ώστε να γίνει εφικτή η ανάπτυξη των προγραμμάτων(sketches) σε διάφορες πλατφόρμες. Τα 3 modes είναι το Java, το JavaScript και το Android. Το Java mode δημιουργεί εφαρμογές της Java. Το JavaScript δημιουργεί εφαρμογές που τρέχουν μέσω HTML5 και WebGL, και το Android mode δημιουργεί

εφαρμογές οι οποίες τρέχουν σε Smartphone με λειτουργικό σύστημα Android όπως και σε ταμπλέτες με λειτουργικό σύστημα Android.




**Εικόνα 1-14** Αναπαράσταση ροής εργασιών

# 2.ΔΟΜΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΣΕΡΒΟΚΙΝΗΤΗΡΩΝ – ΑΙΣΘΗΤΗΡΑ

## ΣΕΡΒΟΚΙΝΗΤΗΡΕΣ

### 2 . 1 Εισαγωγή

Οι σερβοκινητήρες είναι κινητήρες υψηλής ποιότητας που χρησιμοποιούνται όπου απαιτούνται γρήγορες αλλαγές θέσης, ταχύτητας, επιτάχυνσης και παραγωγή μεγάλων ροπών δυνάμεων. Παρόλο που μοιάζουν κατασκευαστικά με τους κοινούς κινητήρες, διαφέρουν στο ότι ενσωματώνουν σύστημα ανάδρασης, το οποίο χρησιμοποιείται σε συνδυασμό με ένα σερβομηχανισμό οδήγησης με σκοπό να ελεγχθεί η ροπή, η ταχύτητα ή η θέση. Το χαρακτηριστικό αυτό τους καθιστά κατάλληλους για μια σειρά χρήσεων. Ως παραδείγματα θα μπορούσαν να αναφερθούν τα τηλε-κατευθυνόμενα αυτοκίνητα και αεροπλάνα, οι ηλεκτρικές οδοντόβουρτσες και ξυριστικές μηχανές, οι οδηγοί CD και DVD, διάφορες βιομηχανικές εφαρμογές ελέγχου κ.α. Η κύρια και πιο συνηθισμένη χρήση τους όμως βρίσκεται στην ρομποτική όπου θα μπορούσαμε να πούμε ότι οι σερβοκινητήρες παίζουν το ρόλο των «μυών» των ρομπότ.

#### 2 . 1 . 1 Ιστορικά Στοιχεία

Η λέξη “σέρβο” από τη λατινική λέξη “servus” που σημαίνει υπηρέτης. Ο σερβοκινητήρας μπορεί να θεωρηθεί ότι είναι ένας κινητήρας που “υπηρετεί” ακολουθώντας πιστά τις εντολές που του επιβάλλονται μέσω του συστήματος ελέγχου. Εδώ οι εντολές είναι η ροπή, η ταχύτητα κι η θέση ή συνδυασμός τους.

Ιστορικά, ο όρος σερβοκινητήρας “Le-Servomoteur” πρωτοχρησιμοποιήθηκε από τον Farcot το 1868 για να περιγράψει ένα σύστημα υδραυλικών ατμοκίνητων μηχανών για την πλοήγηση πλοίων. Ο πρώτος ηλεκτροκίνητος σερβομηχανισμός κατασκευάστηκε το 1898 από τον άγγλο H. Calendar για τη χρησιμοποίηση του σε μηχανοκίνητη καταγραφικό όργανο και το 1908 ο E. Sperry χρησιμοποίησε σερβοκινητήρες σε γυροσκοπική πυξίδα πλοίου.

### 2 . 2 Βασικές Έννοιες

Στο σημείο αυτό θα ήταν χρήσιμο να κάνουμε μια αναφορά σε κάποιες βασικές έννοιες, όπως τάση λειτουργίας, ένταση του ρεύματος, ταχύτητα και ροπή τις οποίες θα συναντήσουμε παρακάτω.

#### 2 . 2 . 1 Τάση λειτουργίας

Οι κινητήρες που χρησιμοποιούνται σε συστήματα μικρών αυτόνομων ρομπότ χαρακτηρίζονται από συνεχή τάση λειτουργίας. Χωρίζονται σε δύο κατηγορίες: στους μικρούς DC κινητήρες, οι οποίοι έχουν τάση 1,5 - 6 V και σε κάποιους πιο υψηλής ποιότητας με τάση 12 - 24

V. Το ενδιαφέρον των κατασκευαστών τέτοιων συστημάτων είναι στραμμένο στους κινητήρες που έχουν τάση λειτουργίας 1,5 - 12 V.

Οι περισσότεροι κινητήρες έχουν τη δυνατότητα να λειτουργούν ικανοποιητικά και σε υψηλότερες ή και σε χαμηλότερες τάσεις από ό,τι τους αναλογεί. Για παράδειγμα, ένας κινητήρας 12 V μπορεί να λειτουργήσει και στα 8 V αλλά δεν θα έχει την ίδια ροπή (με αποτέλεσμα να στρέφεται πιο αργά). Γενικά οι περισσότεροι κινητήρες δεν μπορούν να λειτουργήσουν με τάση κάτω από το 50% της τάσης λειτουργίας που τους αναλογεί.

Ομοίως, ένας κινητήρας 12 V μπορεί να λειτουργήσει στα 16 V. Ο κινητήρας θα έχει μεγαλύτερη ροπή με αποτέλεσμα να γυρίζει ακόμη πιο γρήγορα αλλά με κίνδυνο να κάψει τα τυλίγματα του τυμπάνου. Γενικά θα πρέπει να αποφεύγεται οι κινητήρες να λειτουργούν με τάση μεγαλύτερη των 30%-40% της επιτρεπτής μέγιστης τάσης λειτουργίας. Ο λόγος είναι ότι υπάρχουν υψηλές πιθανότητες υπερθέρμανσης και κατ' επέκταση πρόκλησης ζημίας.

**2 . 2 . 2 Ένταση του ρεύματος**

Η Ένταση του ρεύματος, μετράται σε Amps και είναι η ένταση του ρεύματος που διαρρέει τον κινητήρα όταν αυτός λειτουργεί. Η ένταση αυτή είναι ανάλογη του φορτίου του κινητήρα. Η ελάχιστη τιμή της αντιστοιχεί στην ελεύθερη περιστροφή του κινητήρα. Εάν ο κινητήρας έχει φορτίο η τιμή αυτή αυξάνεται. Για κάποια τιμή του φορτίου ο κινητήρας σταματά να περιστρέφεται, λόγω της μεγάλης αντίστασης, οπότε και η τιμή της έντασης μεγιστοποιείται. Η μέγιστη αυτή τιμή είναι ένα από τα σημαντικά χαρακτηριστικά του κινητήρα που πρέπει να είναι γνωστό για τη σωστή επιλογή της τροφοδοσίας του.

**2 . 2 . 3 Ταχύτητα**

Η ταχύτητα περιστροφής στους κινητήρες μετράται σε στροφές ανά λεπτό (Rounds Per Minute), στα σέρβο σε μοίρες ανά λεπτό και στους βηματικούς κινητήρες σε βήματα ανά δευτερόλεπτο. Πρόκειται για την ταχύτητα περιστροφής του άξονα του κινητήρα, όταν ο κινητήρας λειτουργεί υπό κανονική ηλεκτρική τάση και με δεδομένο φορτίο.

**2 . 2 .4 Ροπή**

Η ροπή, μετράται σε Nm (Newton meters) και είναι η ροπή που παράγει ο κινητήρας στις διάφορες ταχύτητες περιστροφής του. Η μέγιστη τιμή ροπής κάθε κινητήρα, που ονομάζεται ροπή ακινητοποίησης, είναι η ροπή που παράγει όταν το φορτίο που αντιμετωπίζει είναι τόσο μεγάλο, ώστε να τον ακινητοποιεί.

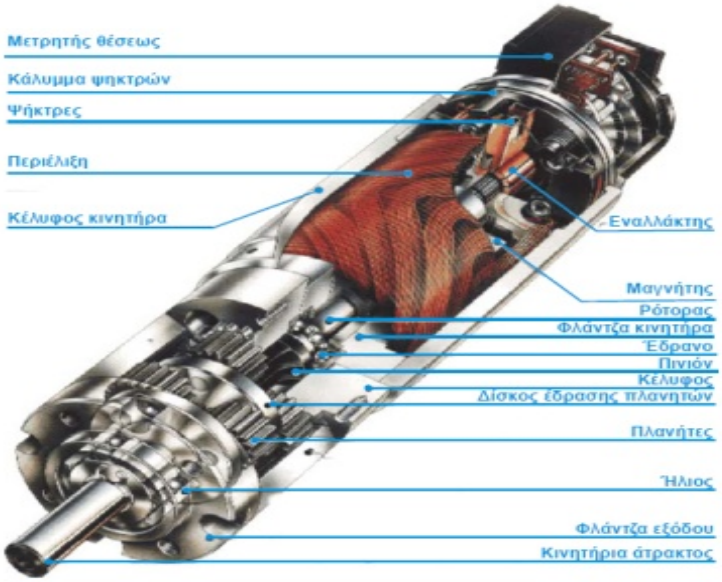
**2 . 3 Ηλεκτροκινητήρας**

Για να κατανοήσουμε καλύτερα τη λειτουργία των σερβοκινητήρων πρέπει πρώτα να κάνουμε μια αναφορά στους ηλεκτροκινητήρες, τα βασικά μέρη από τα οποία αποτελούνται και τον τρόπο

λειτουργίας τους. Ο στάτης, αποτελείται από το ζύγωμα, τους μαγνητικούς πόλους, τους βοηθητικούς πόλους, τον ψηκτροφορέα με τις ψήκτρες, το κιβώτιο ακροδεκτών και τα δύο καλύμματα. Το ζύγωμα είναι το κύριο μέρος του στάτη, αποτελεί τον κορμό της μηχανής μέσω του οποίου ενώνονται μηχανικά και μαγνητικά οι πόλοι. Κατασκευάζεται από χυτοχάλυβα ή μαλακό σίδηρο. Οι μαγνητικοί πόλοι αποτελούν τη διέγερση της μηχανής, δηλαδή την πρωτεύουσα πηγή μαγνητικής ροής στο διάκενο.

Οι βοηθητικοί πόλοι τοποθετούνται μεταξύ των κυρίων πόλων και χρησιμεύουν για την αποφυγή των σπινθηρισμών του συλλέκτη από τη μετακίνηση της ουδέτερης ζώνης, κατά την υπό φορτίο λειτουργία. Ο ψηκτροφορέας με τις ψήκτρες, χρησιμεύει για την προσαγωγή ή απαγωγή του ρεύματος των τυλιγμάτων του δρομέα της μηχανής. Οι ψήκτρες κατασκευάζονται από σκληρό άνθρακα, από γραφίτη ή από μίγμα άνθρακα και χαλκού. Τέλος, τα καλύμματα χρησιμεύουν για τη στήριξη του άξονα του δρομέα, του ψηκτροφορέα και την προφύλαξη του εσωτερικού της μηχανής

Οι βοηθητικοί πόλοι τοποθετούνται μεταξύ των κυρίων πόλων και χρησιμεύουν για την αποφυγή των σπινθηρισμών του συλλέκτη από τη μετακίνηση της ουδέτερης ζώνης, κατά την υπό φορτίο λειτουργία. Ο ψηκτροφορέας με τις ψήκτρες, χρησιμεύει για την προσαγωγή ή απαγωγή του ρεύματος των τυλιγμάτων του δρομέα της μηχανής. Οι ψήκτρες κατασκευάζονται από σκληρό άνθρακα, από γραφίτη ή από μίγμα άνθρακα και χαλκού. Τέλος, τα καλύμματα χρησιμεύουν για τη στήριξη του άξονα του δρομέα, του ψηκτροφορέα και την προφύλαξη του εσωτερικού της μηχανής.




**Εικόνα 2-1 Ηλεκτροκινητήρας**

Τα βασικά τυλίγματα μιας μηχανής συνεχούς ρεύματος είναι δύο, το τύλιγμα διέγερσης το οποίο βρίσκεται στο σταθερό μέρος (στάτη) και το τύλιγμα τυμπάνου στο δρομέα. Κατασκευαστικά τα τυλίγματα διέγερσης είναι συγκεντρωμένα, δηλαδή κάθε πόλος αποτελείται από μια ομάδα, όπου Ν ελίγματα διασυνδεδεμένα σε σειρά καταλήγουν σε δύο ακροδέκτες.

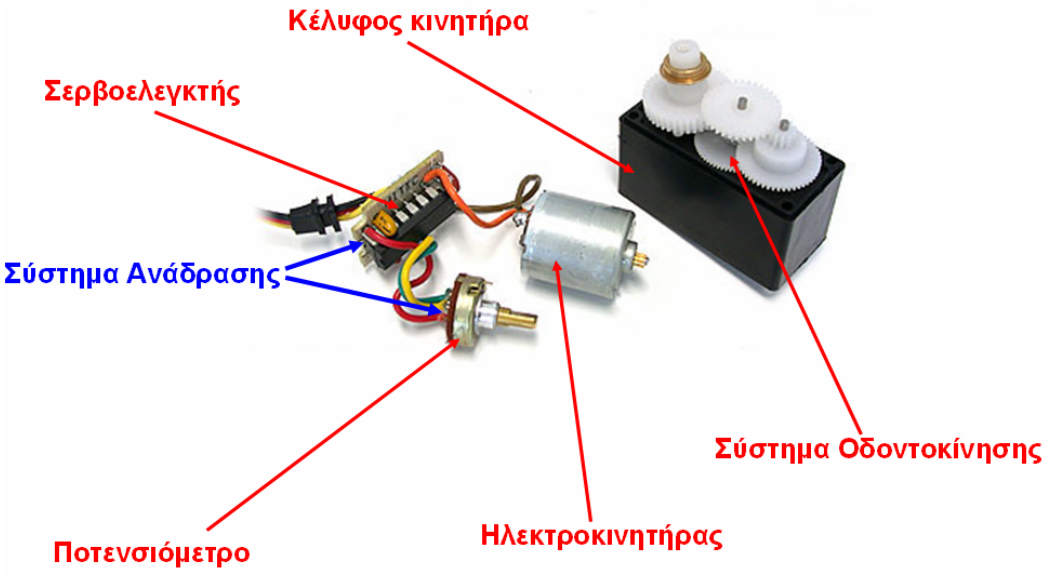
Ο αριθμός των ελιγμάτων αυτών, καθώς επίσης και η διατομή τους, καθορίζονται κυρίως από την επιθυμητή ισχύ, από τη γεωμετρία της μηχανής και από το είδος των σιδηρομαγνητικών υλικών.

Το τύλιγμα τυμπάνου, σε αντίθεση με το τύλιγμα διέγερσης, είναι διανεμημένο. Αποτελείται από ομάδες ομοιόμορφα διανεμημένες και κατάλληλα συνδεδεμένες στις οδοντώσεις (αύλακες), πάνω στην περιφέρεια του διακένου. Τα άκρα των ομάδων αυτών καταλήγουν στους τομείς του συλλέκτη. Οι βασικές κατηγορίες τυλιγμάτων τυμπάνου είναι δύο, τα βροχοτυλίγματα και τα κυματοτυλίγματα.

Οι ηλεκτρικοί κινητήρες μετατρέπουν την ηλεκτρική ενέργεια σε μηχανική. Στο εσωτερικό του κινητήρα ενεργούν μαγνητικά πεδία σε αγωγό που διαρρέεται από ρεύμα. Έτσι δημιουργούνται δυνάμεις που χρησιμοποιούνται για την παραγωγή περιστροφικής κίνησης στο εσωτερικό στρεφόμενο μέρος.

### 2 . 4 Σερβοκινητήρας

Από λειτουργική άποψη, αν εξετάσει κανείς τους σερβοκινητήρες από τη σκοπιά των ηλεκτροκινητήρων, κύριο γνώρισμά τους είναι η ικανότητα να αναπτύσσουν μεγάλες επιταχύνσεις όταν ξεκινούν από πλήρη ακινησία, δηλαδή, να έχουν μικρή ροπή αδράνειας και μεγάλη ροπή στρέψης.




**Εικόνα 2-2** Εσωτερικό του σερβοκινητήρα

Η έννοια ροπή αδράνειας αναφέρεται σε συγκεκριμένο σώμα και σε συγκεκριμένο άξονα. Μονάδα μέτρησης είναι το 1 $kg \cdot m^2$. Η τιμή της εξαρτάται από το «πώς κατανέμεται» η μάζα του στερεού σε σχέση με τον άξονα. Εκφράζει την αδράνεια του σώματος κατά τη μεταβολή της στροφικής κινητικής του κατάστασης γύρω από τον συγκεκριμένο άξονα. Όσο μεγαλύτερη ροπή αδράνειας παρουσιάζει το σώμα, τόσο μικρό-τερη θα είναι η γωνιακή επιτάχυνση την οποία θα «αποδεχθεί» υπό την επίδραση ορισμένης ροπής. Αυτό εκφράζεται και με την εξίσωση $\alpha_{γων} = M_{ολ}/ I$. Η ροπή αδράνειας του σώματος ως προς άξονα και συμβολίζεται με το κεφαλαίο I, αρχικό της λατινικής λέξης Inertia, το $M_{ολ}$ είναι η ολική ροπή της ηλεκτρομαγνητικής δύναμης και το $\alpha_{γων}$ η γωνιακή επιτάχυνση.

Για να πετύχουμε μικρή ροπή αδράνειας και μεγάλη ροπή στρέψης θα πρέπει:

- Ο ρότορας να έχει μεγάλο μήκος και μικρή διάμετρο
- Να υπάρχουν περιελίξεις αντισταθμίσεως, οι οποίες επιτρέποντας ανάπτυξη μεγαλύτερων ρευμάτων αυξάνουν τη ροπή στρέψης.
- Για μικρής ισχύος κινητήρες προβλέπεται μόνιμος μαγνήτης που περιβάλει τα τυλίγματα του ρότορα.
- Να είναι μικρή η σταθερά χρόνου $L \cdot R$ του τυλίγματος του ρότορα

Η επιλογή ενός σερβοκινητήρα γίνεται έχοντας υπόψη ότι η ισχύς του θα πρέπει να καλύπτει την ισχύ του φορτίου (ωφέλιμη) και τις τριβές (απώλειες) της διάταξης. Πέραν αυτού, ο σερβοκινητήρας πρέπει να λειτουργεί στις επιθυμητές ταχύτητες και να μπορεί να δίνει την απαραίτητη επιτάχυνση στο ρότορα και στο φορτίο.

Οι σερβοκινητήρες χωρίζονται στις εξής κατηγορίες: ηλεκτρικούς AC και DC, σε πνευματικούς και υδραυλικούς.

### 2 . 5 Σερβοκινητήρες DC

Οι μονοφασικοί σερβοκινητήρες συνεχούς ρεύματος είναι βασικά τεσσάρων τύπων:

- Ο πρώτος τύπος είναι αυτός που τα τυλίγματα του στάτορα τροφοδοτούνται από πηγή σταθερής τάσεως ή ρεύματος, ενώ το τύλιγμα του ρότορα από μια τάση ελέγχου Vε. Οι σερβοκινητήρες αυτοί είναι γνωστοί σαν ελεγχόμενοι από το ρότορα. Σε αυτούς τους σερβοκινητήρες αν διατηρούμε σταθερή την τάση ελέγχου Vε, η ροπή στρέψης μικραίνει γραμμικά καθώς αυξάνει η γωνιακή ταχύτητας ω του κινητήρα.
- Ο δεύτερος τύπος σερβοκινητήρα είναι ο ελεγχόμενος από το στάτορα. Σε αυτόν τον τύπο το τύλιγμα του ρότορα τροφοδοτείται από μια πηγή σταθερής τάσεως ή ρεύματος, ενώ το τύλιγμα του στάτορα από μια τάση ελέγχου. Σε αυτούς τους

σερβοκινητήρες η ροπή στρέψης είναι ανεξάρτητη από τη γωνιακή ταχύτητα του ρότορα και εξαρτάται μόνο από τη σταθερά Κ και το ρεύμα του στάτορα. Ωστόσο αν το μαγνητικό υλικό εργάζεται στον κόρο, η ροπή στρέψης επηρεάζεται και από τη γωνιακή ταχύτητα του στάτορα και μάλιστα, σε πολύ μεγάλες γωνιακές ταχύτητες η ροπή μικραίνει γιατί αυξάνει πάρα πολύ η αντιηλεκτρεγερτική δύναμη.

- Ο τρίτος τύπος είναι ο σερβοκινητήρας με τα τυλίγματα στάτορα και ρότορα σε σύνδεση σειράς: Οι σερβοκινητήρες αυτοί έχουν διπλό τύλιγμα στο στάτορα, έτσι που το καθένα να συνδέεται σε σειρά με το τύλιγμα του ρότορα με τη βοήθεια ηλεκτρονόμων. Η ροπή στρέψης του κινητήρα μεταβάλλεται εκθετικά και εξαρτάται από τα μεγέθη του ρεύματος ελέγχου και της γωνιακής ταχύτητας. Είναι πολύ μεγάλη κατά την εκκίνηση οπότε η γωνιακή ταχύτητα είναι μικρή, ενώ μικραίνει απότομα όταν η γωνιακή ταχύτητα μεγαλώνει. Χρησιμοποιείται κυρίως εκεί όπου απαιτείται μεγάλη ροπή κατά την εκκίνηση (όπου έχουμε περιστροφή μαζών), αφού η γραμμικότητα δεν παίζει κανένα ρόλο.
- Ένας ιδιαίτερα σημαντικός τύπος σερβοκινητήρας είναι αυτός με μόνιμο μαγνήτη. Ο σερβοκινητήρας του τύπου αυτού έχει αντί για τυλίγματα στάτορα μόνιμο μαγνήτη ενώ, ο ρότορας έχει κανονικό τύλιγμα μέσα από το οποίο ελέγχεται ο κινητήρας. Μοιάζει πολύ με τους ασύγχρονους κινητήρες παράλληλης διέγερσης και λόγω του μικρού όγκου του χρησιμοποιείται σε Συστήματα Αυτομάτου Ελέγχου πάνω σε αεροπλάνα. Ο μικρός όγκος του κινητήρα πετυχαίνεται με ειδικό κράμα μόνιμου μαγνήτη.

### 2 . 6 Σερβοκινητήρες AC

Οι σερβοκινητήρες εναλλασσομένου ρεύματος μπορούν να είναι μονοφασικοί ή τριφασικοί. Οι μονοφασικοί αποτελούνται από δύο τυλίγματα στο στάτη με τέτοια τοποθέτηση, ώστε να παρουσιάζουν διαφορά φάσεως 90 μοιρών με το ρότορα. Το ένα τύλιγμα ονομάζεται τύλιγμα αναφοράς και τροφοδοτείται από μια εναλλασσόμενη τάση σταθερής τιμής, ενώ το άλλο τύλιγμα είναι τύλιγμα ελέγχου και τροφοδοτείται από την τάση ελέγχου. Όταν λοιπόν εφαρμοστούν αυτές οι τάσεις στα τυλίγματα τότε δημιουργείται στρεφόμενο μαγνητικό πεδίο από τα δύο ρεύματα που διαρρέουν τα τυλίγματα και ο ρότορας περιστρέφεται. Οι μονοφασικοί σερβοκινητήρες χρησι-μοποιούνται κυρίως σε οικιακές συσκευές, όπως ψυγεία κ.α.

Οι τριφασικοί σερβοκινητήρες αποτελούνται από δύο τυλίγματα εναλλασσόμενου ρεύματος, ένα στο στάτη και ένα στο δρομέα. Το τύλιγμα του στάτη είναι γνωστό και ως τύλιγμα τυμπάνου.

Κατασκευαστικά το τύλιγμα του στάτη αποτελείται από τρία όμοια διανεμημένα μονοφασικά τυλίγματα, τοποθετημένα στο χώρο σχηματίζοντας μεταξύ τους γωνίες 120 μοιρών. Ο αριθμός των πόλων των τυλιγμάτων αυτών, για δεδομένη συχνότητα τροφοδοσίας, καθορίζει τις ονομαστικές στροφές του κινητήρα. Ανάλογα δε με τις τάσεις τροφοδοσίας και τον τύπο του κινητήρα, τα τυλίγματα του στάτη συνδέονται είτε σε αστέρα είτε σε τρίγωνο. Με τους τριφασικούς κινητήρες μπορούμε να επιτύχουμε υψηλότερη αποδοτικότητα και καλύτερο έλεγχο της κίνησης. Για τον λόγο αυτό χρησημοποιούνται κυρίως σε βιομηχανικές εφαρμογές.

Οι κινητήρες εναλλασσόμενου ρεύματος παρουσιάζουν μεγάλη ροπή στρέψης για μικρές γωνιακές ταχύτητες. Η σχέση μεταξύ ροπής και γωνιακής ταχύτητας είναι όμοια με αυτήν των σερβοκινητήρων συνεχούς ρεύματος που ελέγχονται από το ρότορα, δηλαδή η ροπή στρέψης μικραίνει γραμμικά σε συνάρτηση με την αύξηση της γωνιακής ταχύτητας.

Οι σερβοκινητήρες εναλλασσόμενου ρεύματος ποικίλλουν ανάλογα με την ταχύτητα του άξονα, τη ροπή, την τάση του ρεύματος, και την ισχύ εξόδου. Τέλος, οι αναστρεφόμενοι σερβοκινητήρες εναλλασσόμενου ρεύματος, έχουν τη δυνατότητα να επιτυγχάνουν ωρολογιακή και αντιωρολογιακή περιστροφή με τα ίδια σχεδόν χαρακτηριστικά λειτουργίας.

**2 . 7 Πνευματικοί Σερβοκινητήρες**

Οι πνευματικοί σερβοκινητήρες κάνουν χρήση της πνευματικής ενέργειας που παρέχεται από ένα συμπιεστή και τη μετασχηματίζουν σε μηχανική ενέργεια με τη βοήθεια πιστονιών ή πνευματικών επενεργητών. Οι κινητήρες αυτοί παρουσιάζουν ιδιαίτερες δυσκολίες στον έλεγχο εξαιτίας της αναπόφευκτης συμπιεστότητας του χρησιμοποιούμενου ρευστού. Για το λόγο αυτό δεν χρησιμοποιούνται συχνά εκτός από εφαρμογές που δεν απαιτούν υψηλή ακρίβεια.

**2 . 8 Υδραυλικοί Σερβοκινητήρες**

Οι υδραυλικοί σερβοκινητήρες μετασχηματίζουν την υδραυλική ενέργεια μιας δεξαμενής σε μηχανική με χρήση καταλλήλων αντλιών. Οι κινητήρες αυτοί μπορούν να υλοποιήσουν τόσο μεταφορική (μέσω ενός εμβόλου) όσο και περιστροφική κίνηση (μέσω αξονικών ή ακτινικών εμβόλων).

**2. 9 Έλεγχος θέσης Σερβοκινητήρα**

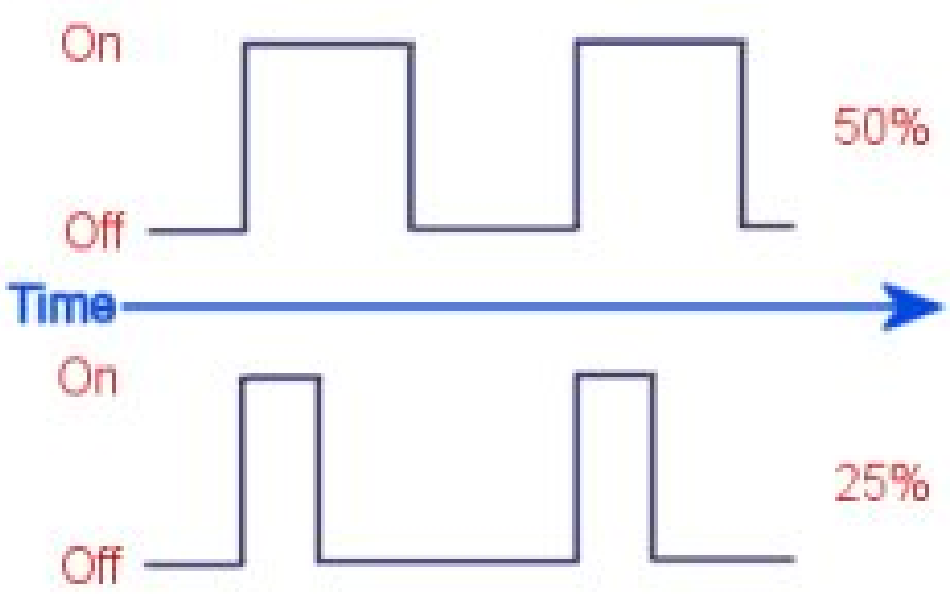
Οι σερβοκινητήρες συνήθως λειτουργούν με την αρχή λειτουργίας της αρνητικής ανάδρασης. Συγκεκριμένα, ο σερβοκινητήρας αποστέλλει σήματα ανατροφοδότησης στον ελεγκτή, ο οποίος ορίζει τη λειτουργία του κινητήρα, αυτός συγκρίνει το σήμα εισόδου (επιθυμητή τιμή) με το σήμα εξόδου του συστήματος (παραγόμενη τιμή). Οι διαφορές (σφάλματα) μεταξύ των πραγματικών και τον επιθυμητών δεδομένων λαμβάνονται υπόψη έτσι ώστε να οδηγηθεί στο σύστημα στην επιθυμητή θέση – σταθεροποίηση του άξονα του κινητήρα.

Η λειτουργία ανάδρασης σε ένα σερβοκινητήρα αποσκοπεί στο συνεχή έλεγχο των εντολών θέσης και ταχύτητας που δίνονται προς τον κινητήρα. Αυτό επιτυγχάνεται από τον ενισχυτή του σερβοσυστήματος που αποτελεί και το σύστημα οδήγησης του σερβοκινητήρα (servodrive).

Πιο συγκεκριμένα η λειτουργία των ενισχυτών του σερβοκινητήρα (servodrives) αποσκοπεί στο να διατηρεί σταθερές τις απαιτούμενες στροφές, να διατηρεί σταθερή τη ροπή σε όλη την περιοχή στροφών του κινητήρα, αλλά ταυτόχρονα να δίνει τη δυνατότητα της βηματικής κίνησης με απόλυτο έλεγχο των δύο προηγούμενων παραμέτρων.

Η διαφορά ενός ηλεκτρικού κινητήρα, με έναν σερβοκινητήρα είναι ότι στον ηλεκτρικό κινητήρα, όταν εφαρμόσουμε διαφορά δυναμικού, αυτός γυρίζει (πλήρεις περιστροφές) με συγκεκριμένη ταχύτητα και δεν χρειάζεται προγραμματισμό. Οι συμβατικοί DC κινητήρες χρειάζονται ορθή και ανάστροφη πολικότητα για ωρολογιακή και αντιωρολογιακή περιστροφή του άξονα Αντίθετα στο σερβοκινητήρα ο τελικός άξονας κίνησης δεν εκτελεί πλήρεις περιστροφές, αλλά περιστρέφεται μεταξύ δύο ακραίων θέσεων Α, Β. Για τη λειτουργία του σέρβο απαιτείται η παροχή της κατάλληλης ηλεκτρικής τάσης αλλά και ενός σήματος ελέγχου που καθορίζει τη θέση περιστροφής του τελικού άξονα.

Το σήμα αυτό είναι μια παλμοσειρά μεταβλητού εύρους (Pulse Width Modulation). Πρόκειται δηλαδή για μια παλμοσειρά με σταθερή περίοδο και με μεταβλητό εύρος παλμού (Duty Cycle). Διαμορφώνοντας το εύρος του παλμού μπορούμε να ελέγξουμε την θέση του κινητήρα.



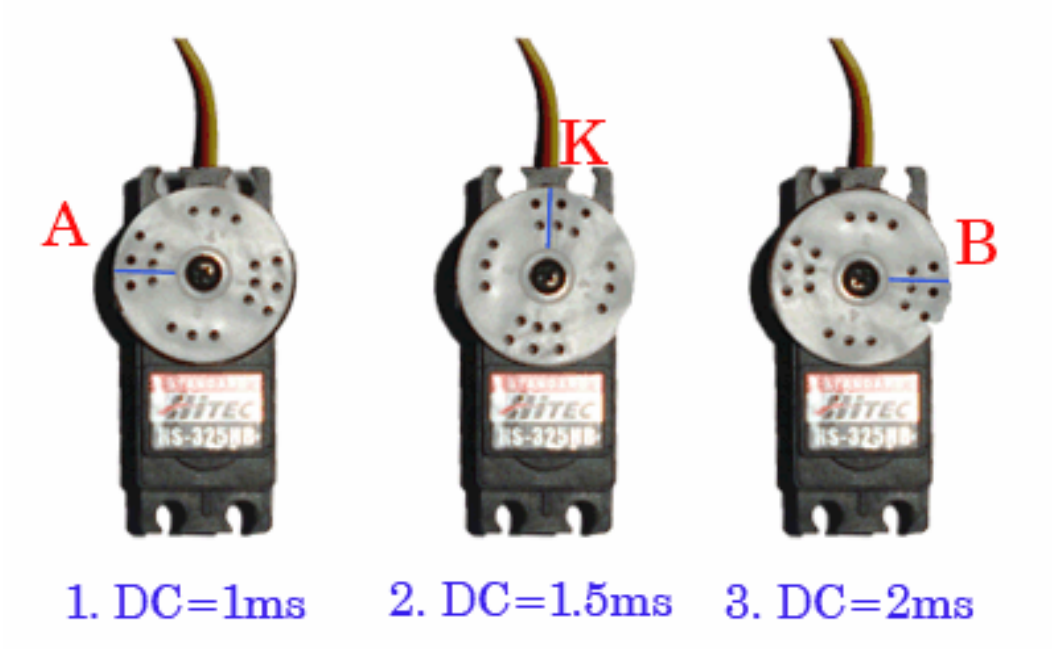

**Εικόνα 2-3** Παλμοσειρά μεταβλητού εύρους (Pulse Width Modulation)

Το εύρος του παλμού αναφέρεται συχνά και σαν ποσοστό επί τις περιόδου, ενώ εύκολα υπολογίζεται από τον τύπο:

$$\text{Duty Cycle} = \frac{t_{ON}}{t_{ON} + t_{OFF}}$$

**Εικόνα 2-4**

Συγκεκριμένα, ένας σερβοκινητήρας ελέγχεται με τη βοήθεια τριών καλωδίων 1. κόκκινο, 2. μαύρο, 3. λευκό ή πορτοκαλί (τα χρώματα των καλωδίων μπορεί να διαφέρουν ανάλογα με την εταιρία παραγωγής του σερβοκινητήρα). Εφαρμόζοντας τάση τροφοδοσίας στα δύο πρώτα και παλμό οδήγησης στο τρίτο ο κινητήρας μπορεί να στραφεί μεταβάλλοντας απλά τη διάρκεια του ενεργού παλμού που εφαρμόζεται στο λευκό καλώδιο εισόδου[1]. Ο κύκλος εργασίας του παλμού (Duty Cycle) καθορίζει την θέση στρέψης του σερβοκινητήρα. Για παλμός εύρους 1ms ο άξονας του κινητήρα θα κατευθυνθεί στη θέση Α (σχ. 1.1), ενώ για παλμό με εύρος 2ms θα κατευθυνθεί στη θέση Β. Οποιοσδήποτε παλμός με εύρος μεταξύ 1ms και 2ms θα δώσει και ανάλογη γωνία στρέψης. Παλμός οδήγησης με περίοδο λειτουργίας 1,5 ms θα μας δώσει το «κέντρο» (στην περίπτωση μας Κ). Στο σχήμα 1.1 μπορούμε να δούμε όσα περιγράψαμε παραπάνω.




**Εικόνα 2-4** Θέσης του άξονα σε αναλογία με το DC

Αν σταματήσουμε να στέλνουμε παλμό οδήγησης για παραπάνω από 50ms ο κινητήρας θα αδρανοποιηθεί. Αυτό σημαίνει ότι ο κινητήρας δεν θα ασκεί καμία δύναμη και μόνο οι τριβές θα τον σταθεροποιούν στη θέση του.

**2 . 10 Κιβώτιο Μειωτήρα**

Απαραίτητο και αναπόσπαστο κομμάτι ενός σερβοκινητήρα είναι το ενσωματωμένο κιβώτιο μειωτήρα ή αλλιώς σύστημα οδοντοκίνησης. Το κιβώτιο μειωτήρα περιλαμβάνει οδοντοτροχούς (γρανάζια) οι οποίοι μεταφέρουν την κίνηση από οδοντοτροχό σε οδοντοτροχό. Τα δόντια είναι κατασκευασμένα έτσι ώστε να μειώνονται οι τριβές, οι δονήσεις και ο θόρυβος, ενώ μεγιστοποιείται η αποδοτικότητα μετάδοσης της δύναμης. Επίσης ζεύγη οδοντοτροχών διαφορετικών μεγεθών μπορούν να χρησι-μοποιηθούν έτσι ώστε να επιτευχθεί μεγάλη ροπή στρέψης με χαμηλή ταχύτητα ή μικρή ροπή με μεγάλη ταχύτητα.

**Εικόνα 2-5** Κιβώτιο μειωτήρα σερβοκινητήρα

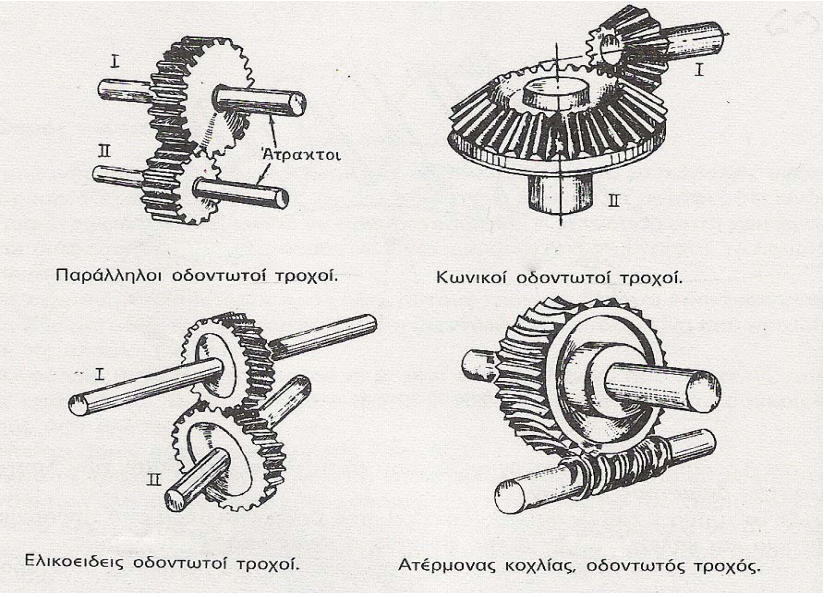
Η κίνηση που μεταδίδεται με τους οδοντωτούς τροχούς λέγεται οδοντοκίνηση και αποτελεί το δυσκολότερο τρόπο μετάδοσης κίνησης σε σύγκριση με άλλους τρόπους (τροχοί, ιμάντες, αλυσίδες).

Οδοντωτός τροχός λέγεται κάθε μεταλλικός ή και από οποιαδήποτε άλλη ανθεκτική ύλη κατασκευασμένος δίσκος, που η περιφέρειά του χωρίζεται κατά κανονικά διαστήματα σε εσοχές και εξοχές, δηλαδή σε δόντια. Όλα τα δόντια ενός τροχού πρέπει να έχουν την ίδια μορφή, δηλαδή να έχουν το ίδιο ύψος, πάχος και απόσταση μεταξύ τους.

Ανάλογα με τη θέση πού έχουν οι άτρακτοι στον χώρο καθορίζεται το είδος και η μορφή των δοντιών των τροχών και προκύπτουν οι εξής κατηγορίες: Παράλληλοι οδοντοτροχοί, κωνικοί οδοντοτροχοί και ελικοειδείς οδοντοτροχοί – ατέρμων κοχλίας. Στο παρακάτω σχήμα φαίνονται τα διάφορα είδη των οδοντοτροχών.

Στους σερβοκινητήρες συναντάμε τους παράλληλους οδοντοτροχούς. Όταν οι τροχοί αυτοί βρίσκονται σε κανονική εμπλοκή και εργάζονται, οι αρχικές τους περιφέρειες εφάπτονται μεταξύ τους**,** η δε κίνησή τους μπορεί να εξομοιωθεί με κύλιση της μιας αρχικής περιφέρειας επάνω στην άλλη.

Αφού λοιπόν η αρχική περιφέρεια του ενός τροχού διαμέτρου $d_1$ βρίσκεται συνεχώς σε επαφή με την αρχική περιφέρεια του άλλου τροχού διαμέτρου $d_2$, αυτό σημαίνει ότι και οι δύο οι δύο τροχοί έχουν στο σημείο επαφής τους την ίδια περιφερειακή ταχύτητα.

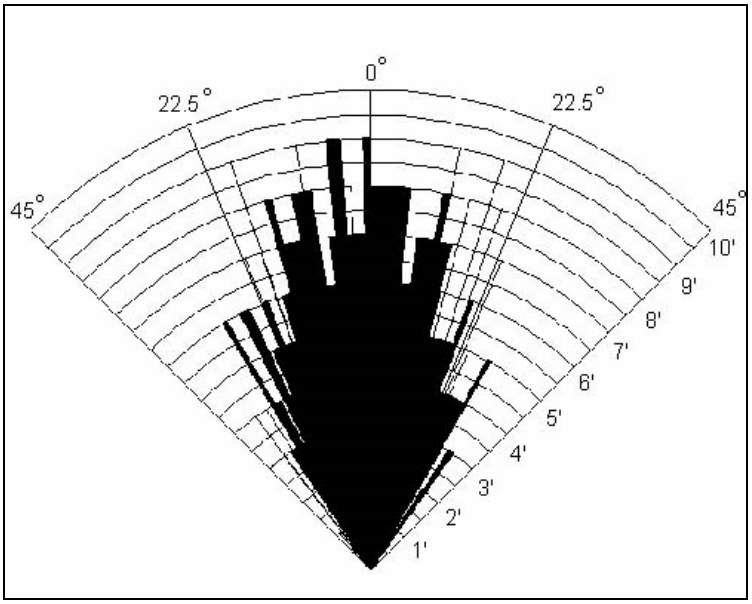



**Εικόνα 2-6** Τα είδη των οδοντοτροχών

**ΑΙΣΘΗΤΗΡΑΣ**

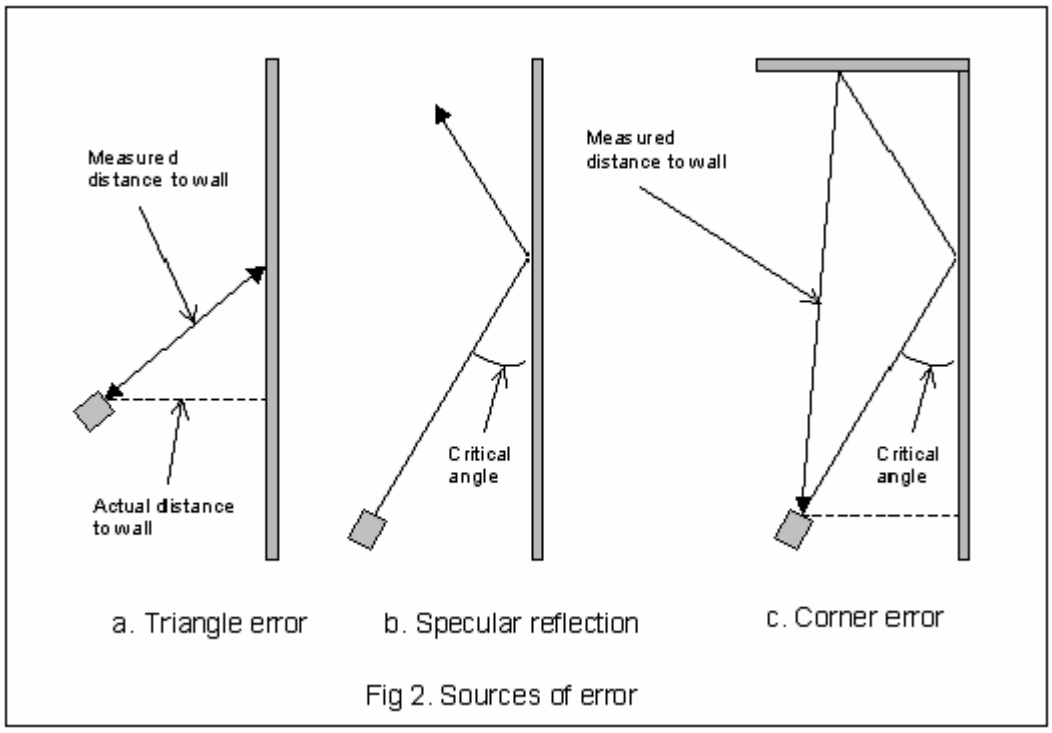
**2 . 11 Αισθητήρας Υπερήχων**

Για τις μετρήσεις αποστάσεων και εύρεσης εμποδίων με τη χρήση υπερήχων χρησιμοποιήθηκε ο αισθητήρας Devantech SRF05 και υποστηρίζει δυο τύπους λειτουργίας. Οι αισθητήρες υπερήχων είναι ηχοβολιστικά που το μήκος κύματος των ηχητικών σημάτων που στέλνουν δεν ανήκουν στο φάσμα συχνοτήτων που αντιλαμβάνεται το ανθρώπινο αυτί. Εκπέμπουν συνήθως το ηχητικό σήμα σε μορφή κώνου με γωνία έως και $30^o$ (Εικόνα 2-7).




**Εικόνα 2-7** Γράφημα Ηχ. σήματος

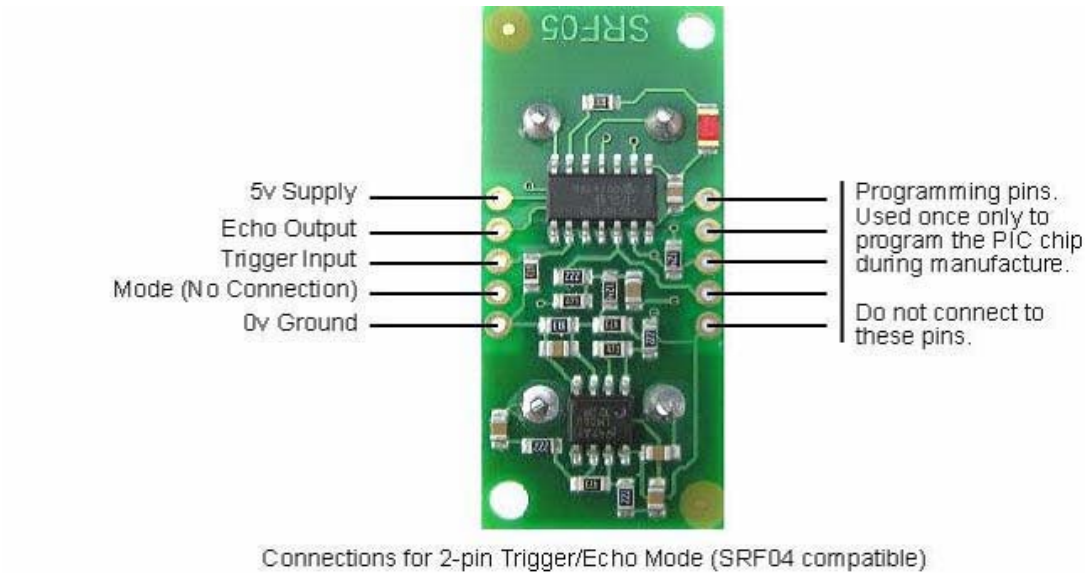
Μπορούν να αντιληφθούν αντικείμενα μέχρι και 4 μέτρα μακριά, εάν δεν υπάρχει αντικείμενο μέσα σε αυτά τα 4 μέτρα επιστρέφει την τιμή 4 μέτρα που ουσιαστικά είναι το άπειρο. Η εμβέλεια του κάθε αισθητήρα εξαρτάται από την ισχύ του σήματος που στέλνει. Στις περισσότερες περιπτώσεις το σήμα του αισθητήρα δεν προσπίπτει κάθετα στα αντικείμενα με αποτέλεσμα να ανακλάται σε άλλη κατεύθυνση ή μετά από διαδοχικές ανακλάσεις να επιστρέφει στον αισθητήρα δίνοντας όμως λάθος μέτρηση (Εικόνα 2-8). Για την αντιμετώπιση αυτών των περιπτώσεων μπορούμε να αλλάξουμε το περιβάλλον (ιδανική περίπτωση), να χρησιμοποιήσουμε πολλαπλούς αισθητήρες για επαλήθευση ή να χρησιμοποιήσουμε ενεργή αίσθηση.




**Εικόνα 2-8** Περιπτώσεις λάθος μετρήσεων

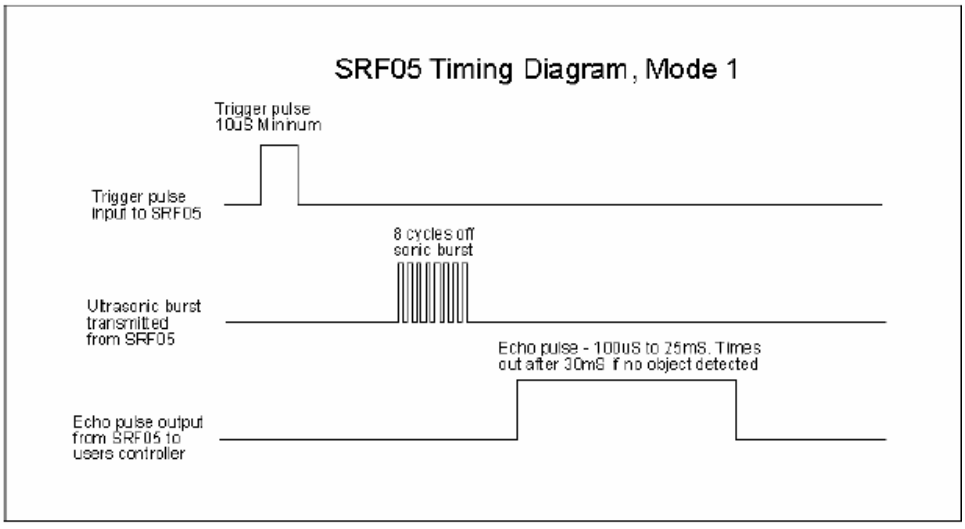
### 2 . 12 Αισθητήρας SRF-05

Στον ένα τύπο (MODE 1) χρησιμοποιείται ένας ακροδέκτης από όπου παίρνει το σήμα διέγερσης (παλμός σε υψηλό επίπεδο )και ένας άλλος ακροδέκτης για την έξοδο του σήματος της μέτρησης. Η συνδεσμολογία για αυτόν τον τύπο λειτουργίας φαίνεται στην Εικόνα 2-9.

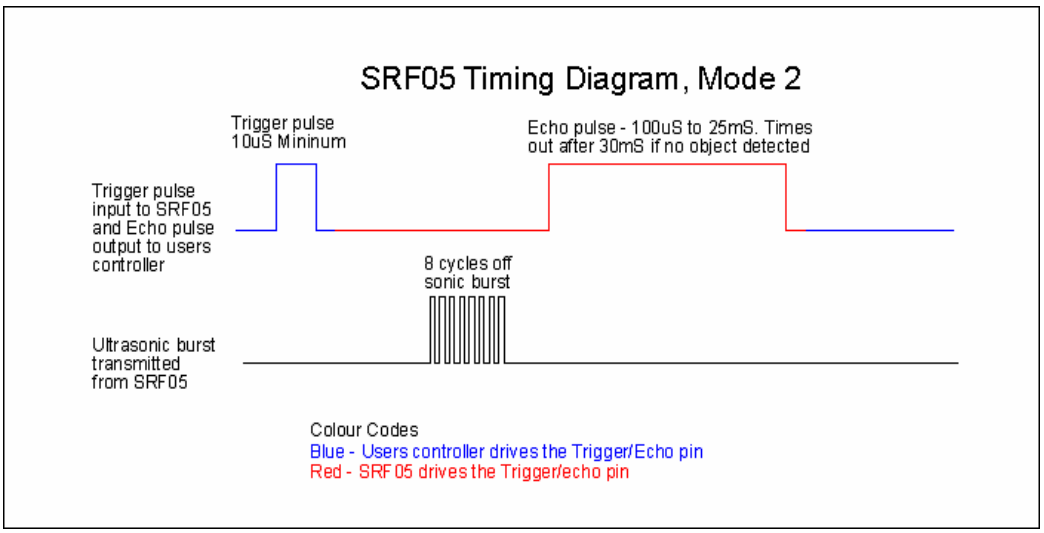



**Εικόνα 2-9** Αισθητήρας SRF-O5

Στην Εικόνα 2-10 φαίνεται η χρονική ακολουθία των παλμών κατά τον πρώτο τύπο λειτουργίας. Αρχικά αποστέλλεται το σήμα διέγερσης στον αισθητήρα (trigger), έπειτα ο αισθητήρας εκπέμπει το ηχητικό σήμα (sound), και τέλος εμφανίζεται ένας παλμός στο κανάλι εξόδου ο οποίος μπορεί να έχει διάρκεια από 100μs έως 18ms. Αναλόγως την διάρκεια του παλμού αυτού υπολογίζεται και η απόσταση. Στην Εικόνα 5 απεικονίζεται η χρονική ακολουθία των παλμών κατά τον δεύτερο τύπο λειτουργίας· η μόνη διαφορά είναι ότι το σήμα διέγερσης και το σήμα εξόδου βρίσκονται στο ίδιο κανάλι(Εικόνα 2-11).




**Εικόνα 2-10** Χρονοδιάγραμμα αισθητήρα σε Mode1




**Εικόνα 2-11** Χρονοδιάγραμμα αισθητήρα σε Mode2

### 2 . 12.1 Επιπλέον Ακροδέκτες

Οι πέντε επιπλέον ακροδέκτες χρησιμοποιήθηκαν αρχικά για τον προγραμματισμό της μνήμης του αισθητήρα και δεν πρέπει σε καμία περίπτωση να συνδεθεί κάτι σε αυτούς. Ο αισθητήρας λειτουργεί με 5Volt και η συχνότητα του ηχητικού σήματος είναι 40MHz. Η εμβέλειά του κυμαίνεται από ένα εκατοστό έως τέσσερα μέτρα (Εικόνα 2-12)

| Specifications | |
|---|---|
| Frequency | 40kHz |
| Max Range | 4 meters |
| Min Range | 3 centimeters |
| Input Trigger | 10uSec minimum, TTL level pulse |
| Echo Pulse | Positive TTL level signal, proportional to range |

**Εικόνα 2-12** Τεχνικά Χαρακτηριστικά SRF-05

### 2 . 12.2 Περιγραφή Μέτρησης Απόστασης

Όπως φαίνεται στις εικόνες ο παλμός trigger πρέπει να έχει διάρκεια τουλάχιστον 10 μs. Μετά τον παλμό πυροδότησης η γραμμή echo (ίδια με την trigger στο mode 2) θα παραμείνει αδρανής για 700 μs. Έπειτα εμφανίζεται ο παλμός echo με διάρκεια από 100μs μέχρι 25 ms, ανάλογα με την απόσταση του στόχου. Σε περίπτωση που δεν υπάρχει αντικείμενο (δηλαδή αντανάκλαση) στην εμβέλεια του σόναρ, η γραμμή echo θα μηδενιστεί μετά από 30 ms.

Εφόσον η απόσταση και η διάρκεια του παλμού είναι ανάλογες, ο υπολογισμός της πρώτης από τη δεύτερη **γίνεται με διαίρεση με την τιμή 58 σύμφωνα με τις προδιαγραφές του αισθητηρίου**.Η διαδικασία μπορεί να επαναλαμβάνεται κάθε 50 ms.

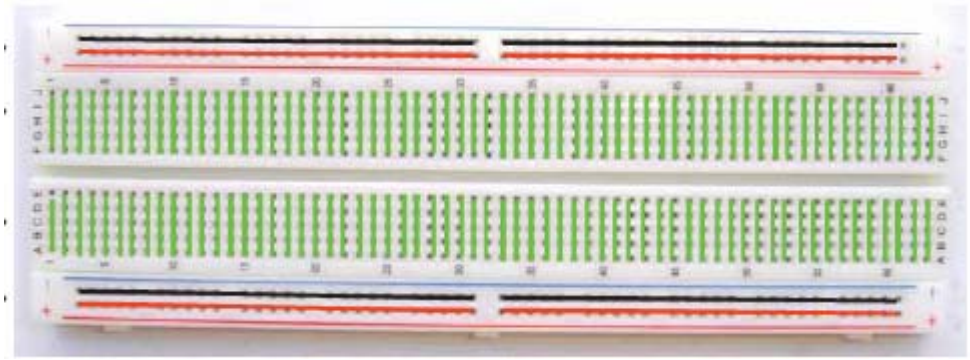
Διαιρούμε με το 58 διότι αν πάρουμε τον χρόνο που κάνει σε microsecond ένας παλμός για να αποσταλεί και να παραλειφθεί σε απόσταση ενός μέτρου είναι περίπου 5764 microseconds, διαιρούμε με το 100 για να το μετατρέψουμε σε εκατοστά και μας προκύπτει 57.64. Στρογγυλοποιώντας την τιμή αυτή καταλήγουμε στο 58.

# 3 . ΠΑΡΟΥΣΙΑΣΗ ΔΙΑΔΙΚΑΣΙΑΣ ΚΑΤΑΣΚΕΥΗΣ

## 3 . 1 Υλικά που χρησιμοποιήθηκαν

**- Πλακέτα σύνδεσης**

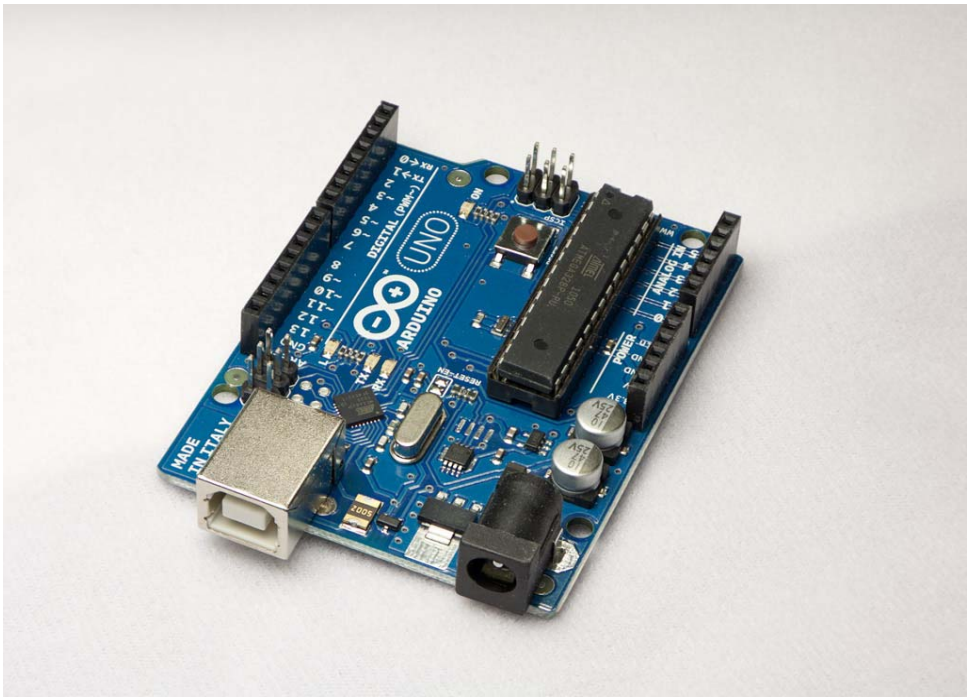
Επειδή η πλακέτα σύνδεσης δεν απαιτεί συγκόλληση, είναι επαναχρησιμοποιήσιμη , αυτό το καθιστά εύκολο στη χρήση για τη δημιουργία προσωρινών πρωτοτύπων και πειραμάτων με το σχεδιασμό του κυκλώματος.



**Εικόνα 3-1** Πλακέτα σύνδεσης

- **Μικροελεγκτής Arduino**

Ο Arduino Uno είναι ο μικροελεγκτής που χρησιμοποιούμε στη κατασκευή του Ραντάρ. Αναλυτικές πληροφορίες έχουν δοθεί στο κεφάλαιο 1.5.




**Εικόνα 3-2** Arduino Uno

- **Σερβοκινητήρας Futaba-s3113**

Αναλυτικές πληροφορίες για τους σερβοκινητήρες έχουν δοθεί στα κεφάλαια 2.1-2.10.

**Εικόνα 3-3** Servo Futaba s3113

- **Αισθητήρας Υπερήχων SRF-O5**

Αναλυτικές πληροφορίες για τους αισθητήρες έχουν δοθεί στο κεφάλαια 2.11-2.12

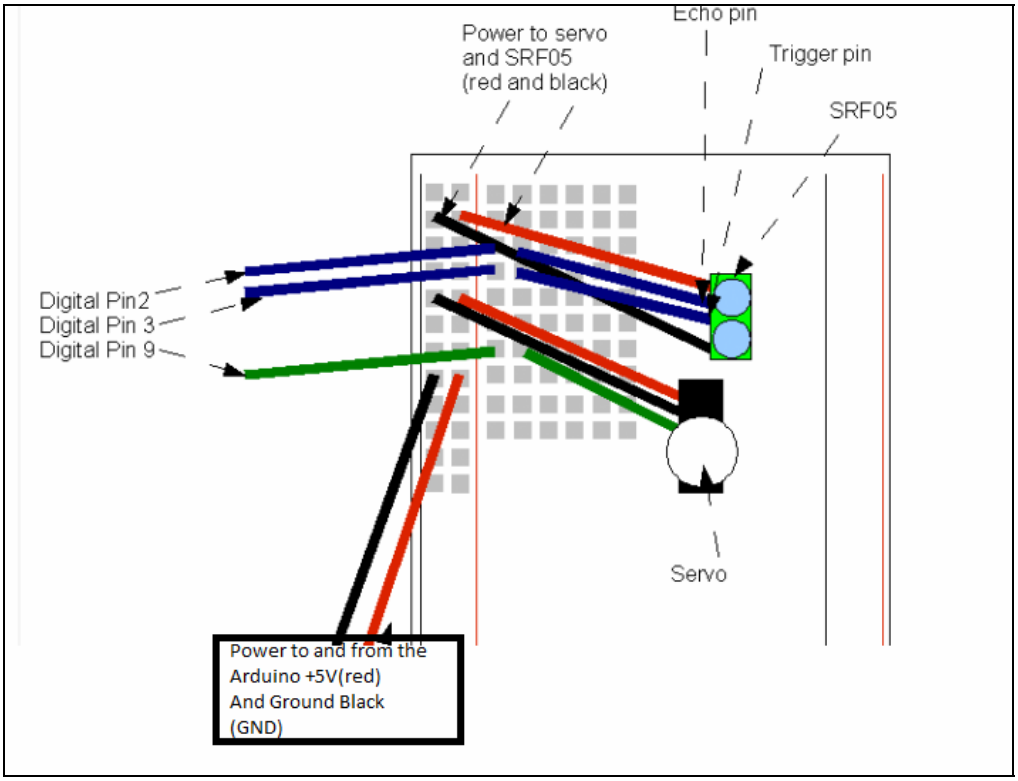
**Εικόνα 4-4** Αισθητήρας SRF-05

-Διάφορα αναλώσιμα υλικά όπως καλώδια , μονωτικές ταινίες , χαρτόνια, κτλ..

**Εικόνα 3-5** Αναλώσιμα

### 3 . 2 Συνδεσμολογία Κατασκευής

Το κύκλωμα μας παίρνει τάση από τον Μικροελεγκτή Arduino ισοδύναμη με +5V, και γείωση απ' το pin της γείωσης του Arduino. Συνδέουμε το σερβοκινητήρα και τον αισθητήρα στο κύκλωμα δίνοντας τα τάση +5V. Συνδέουμε το σερβοκινητήρα με το pin 9 του Arduino . Συνδέουμε το echo pin του αισθητήρα με το pin 2 του Arduino και το trigger pin με το pin 3 του Arduino.

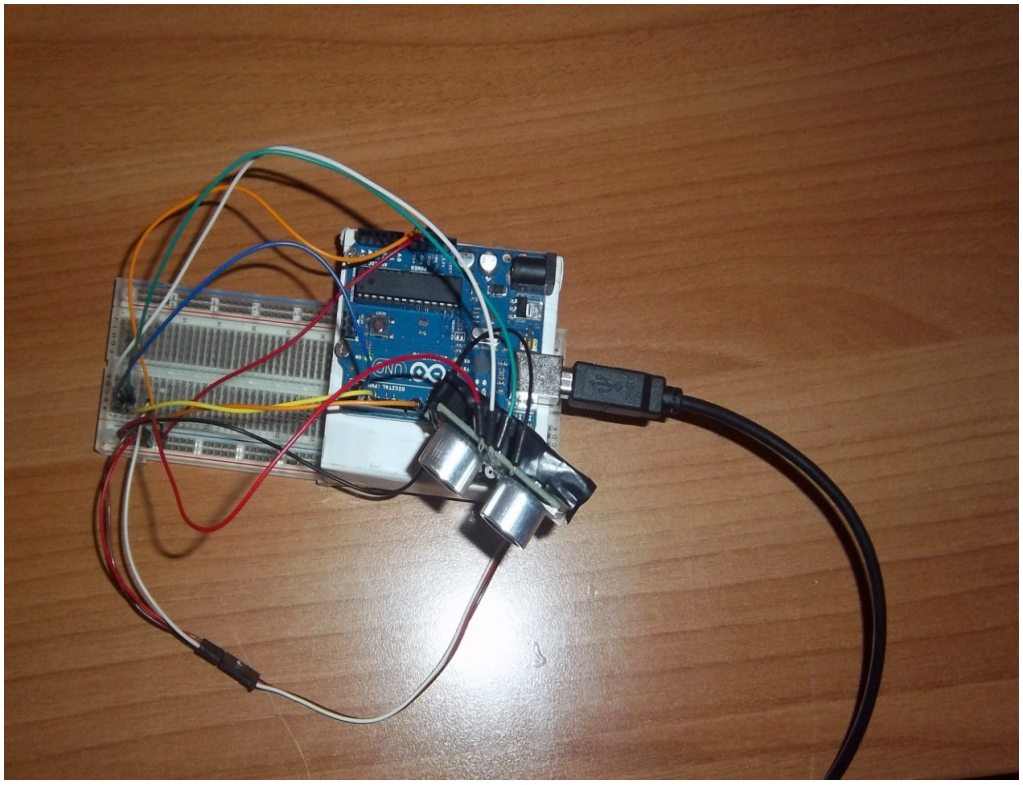



**Εικόνα 3-6** Κύκλωμα Ραντάρ

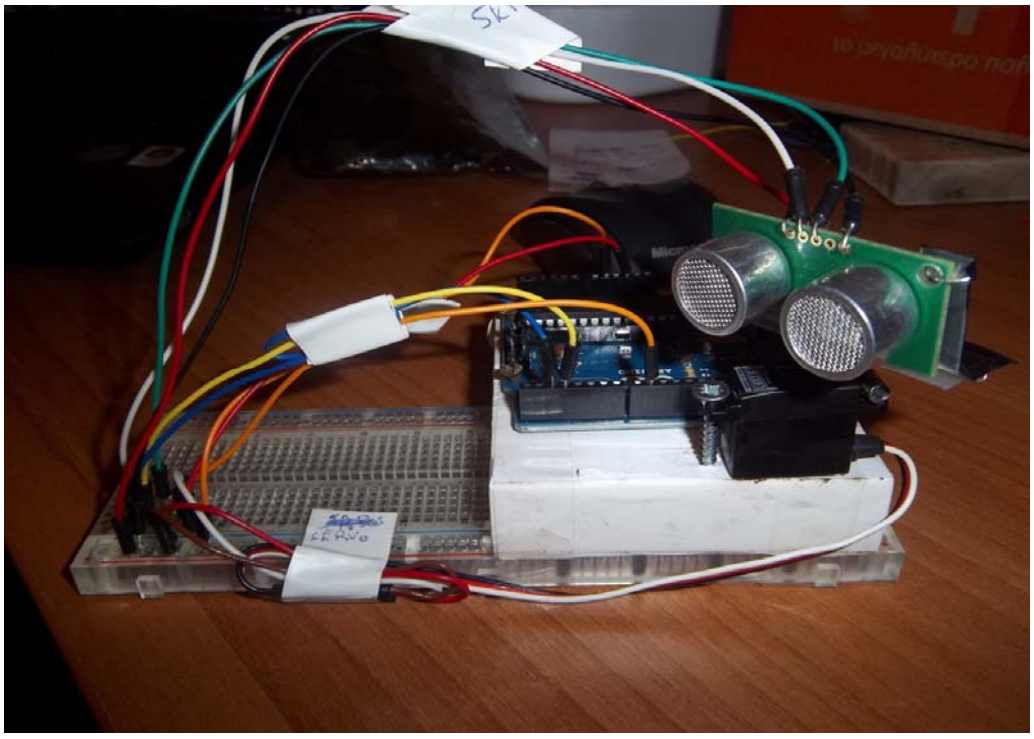
### 3 . 3 Διαδικασία Κατασκευής - Φωτογραφικό υλικό

Το πρώτο βήμα που κάναμε για να κατασκευάσουμε το ραντάρ είναι να στερεώσουμε τον αισθητήρα πάνω στο σερβοκινητήρα. Στη συνέχεια πήραμε ένα κομμάτι ξύλο το οποίο το κολλήσαμε πάνω στη πλακέτα σύνδεσης. Πάνω στο ξύλο τοποθετήσαμε τον μικροελεγκτή Arduino αλλά και το σερβοκινητήρα, στη συνέχεια τα στερεώσαμε χρησιμοποιώντας κάποιες βίδες ώστε η κατασκευή να είναι σταθερή. Επόμενο βήμα ήταν να κατασκευάσουμε κάτι το οποίο θα κάλυπτε όλο το κύκλωμα και θα άφηνε μόνο τον αισθητήρα να φαίνεται . Πήραμε ένα είδος ''χοντρού ''χαρτονιού και κατασκευάσαμε ένα ορθογώνιο παραλληλόγραμμο μέσα στον

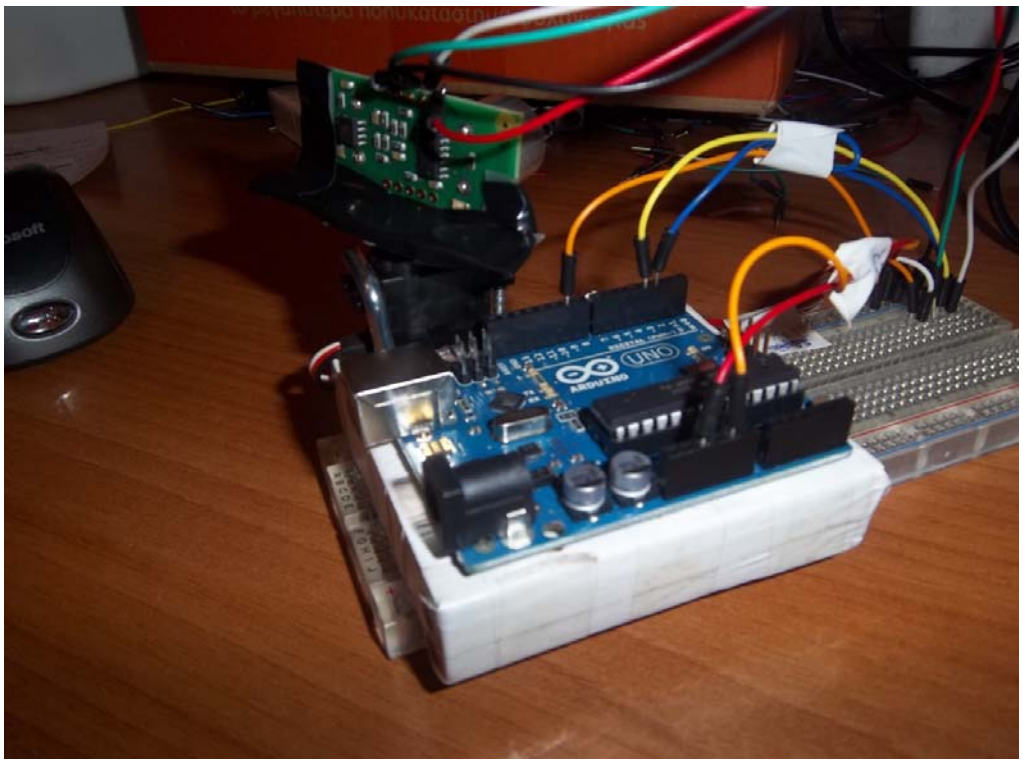
οποίο και τοποθετήσαμε το κύκλωμα μας. Ακολουθούν φωτογραφίες οι οποίες παρουσιάζουν τις διάφορες φάσεις της κατασκευής.



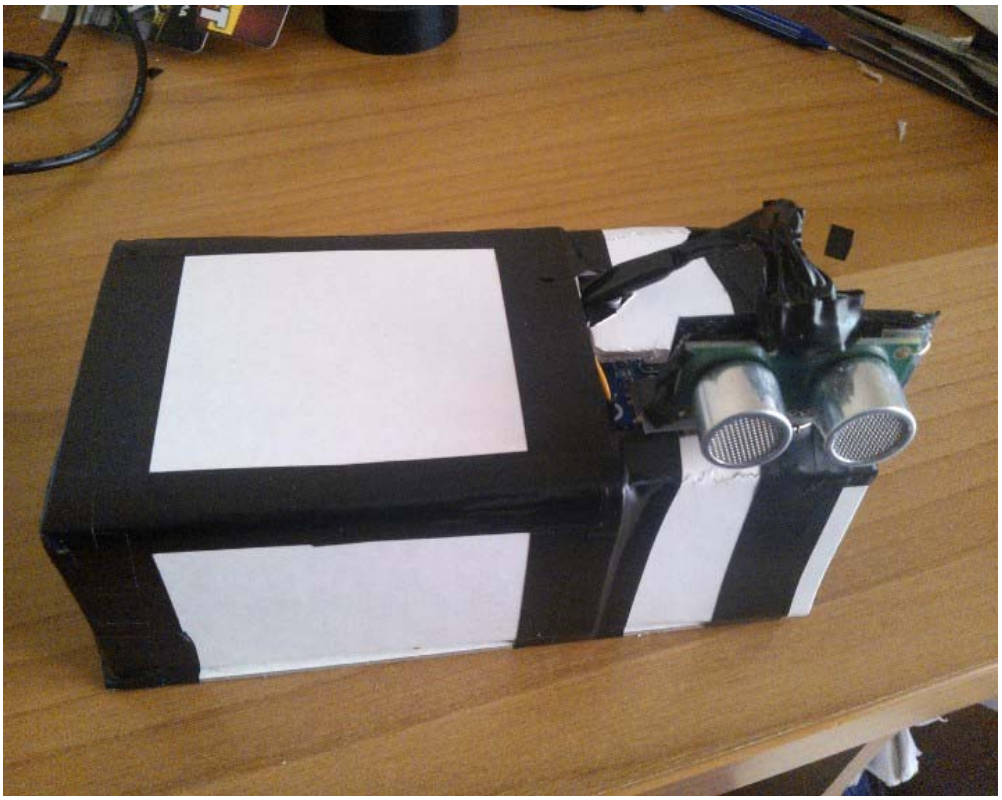
**Εικόνα 3-7**



**Εικόνα 3-8**




**Εικόνα 3-9**



**Εικόνα 3-10**

**Εικόνα 3-11**

**Εικόνα 3-12**

### 3 . 4 Σχολιασμός κώδικα

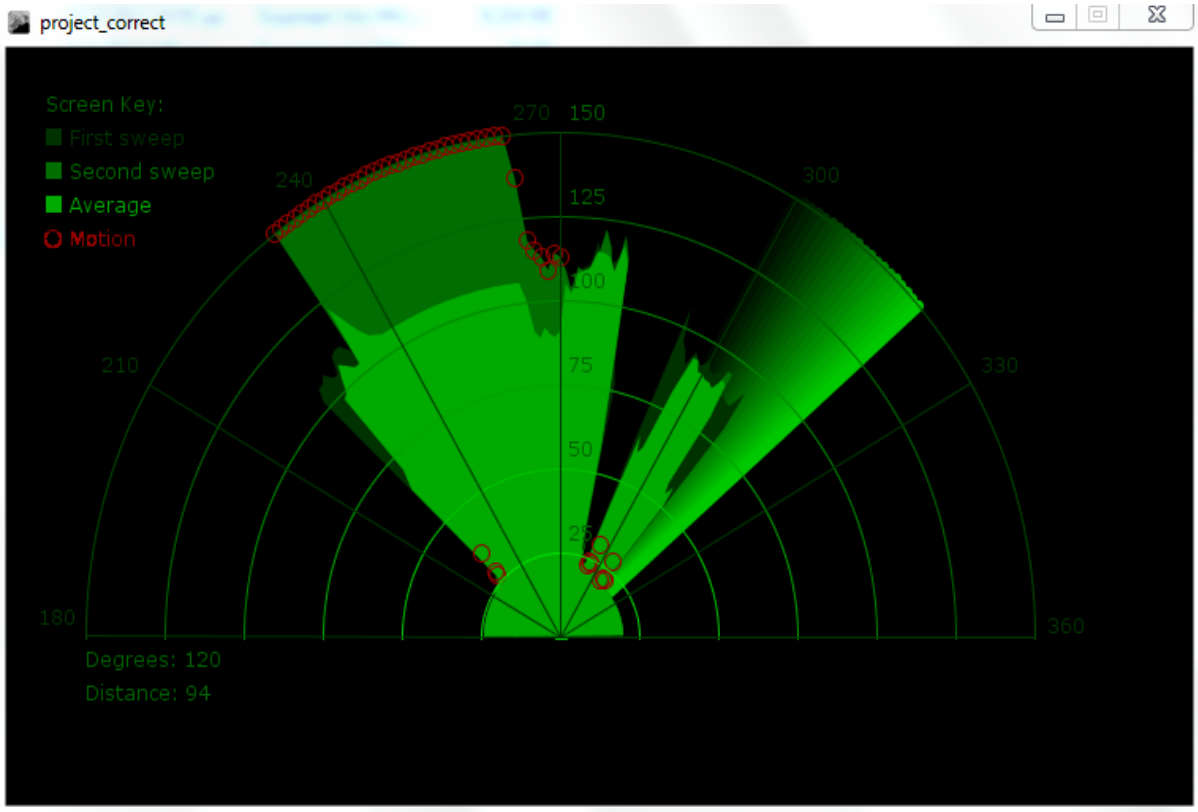
**ARDUINO**

Στο πρώτο κομμάτι του κώδικα που αφορά τον προγραμματισμό του Arduino, στην ουσία προγραμματίζουμε την κίνηση- περιστροφή του σερβοκινητήρα και στη συνέχεια παίρνουμε τα δεδομένα από τον αισθητήρα. Σε κάθε μία απ' τις 180 θέσεις που παίρνει ο σερβοκινητήρας διαβάζουμε δέκα φορές και στη συνέχεια βρίσκουμε το μέσο όρο των διαβασμάτων αυτών . Αυτό γίνεται για να μειώσουμε όσο είναι δυνατόν το ποσοστό σφάλματος στα δεδομένα που παίρνουμε.

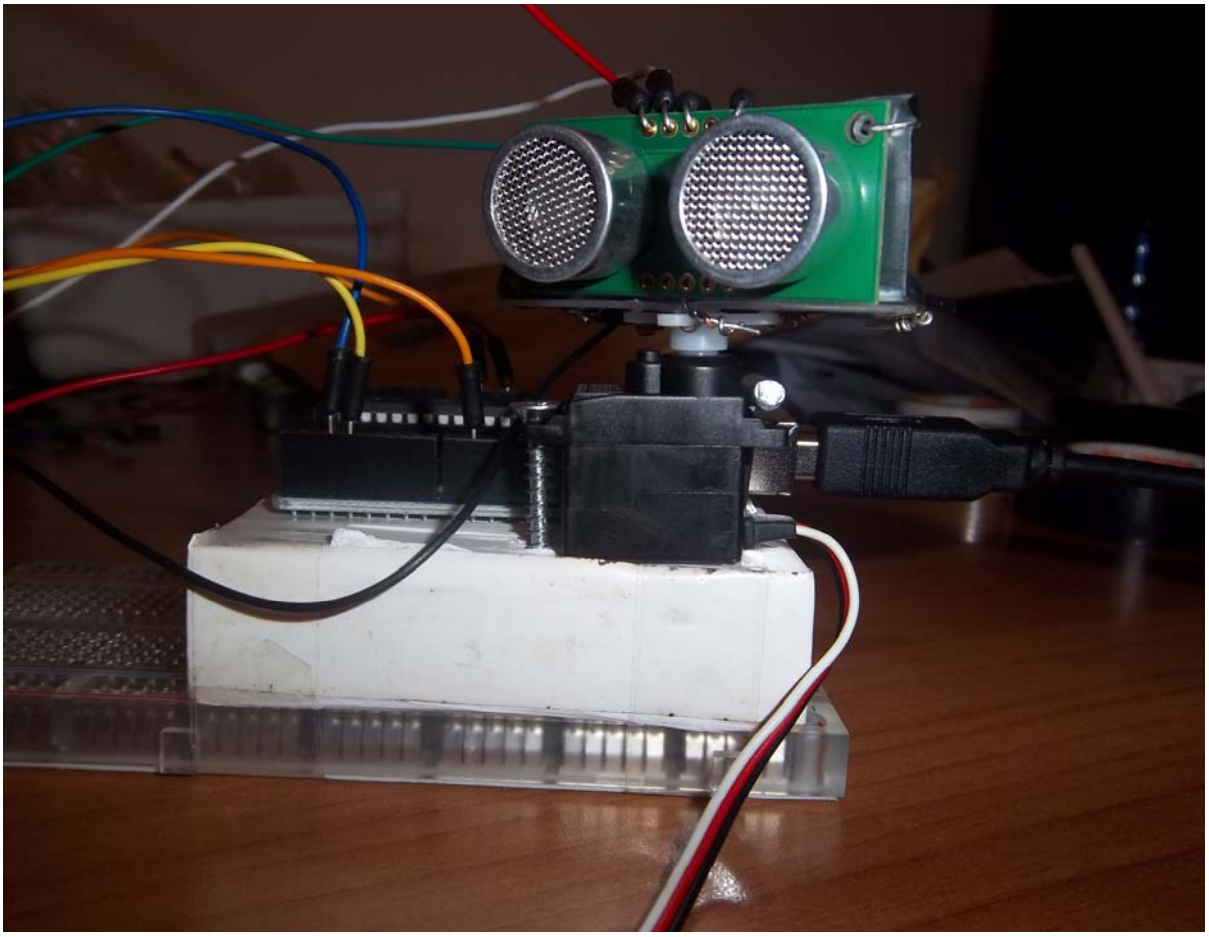
**PROCESSING**

Στο δεύτερο κομμάτι του κώδικα το οποίο αφορά το κομμάτι του σχεδιασμού και της παρουσίασης των δεδομένων τα πράγματα είναι λίγο πιο περίπλοκα.

Σύμφωνα με τους κανόνες του τριγωνομετρικού κύκλου, ένα σημείο Μ(α,β) πάνω στο κύκλο περιγράφεται απ το συνημίτονο της γωνίας που σχηματίζει η ακτίνα του σημείου σε σχέση με τον άξονα xx' (α), και το ημίτονο της γωνίας που σχηματίζει η ακτίνα του σε σχέση με τον άξονα yy' . Με αυτόν τον τρόπο βρίσκουμε τις κορυφές (σημεία όπου βρίσκει ο αισθητήρας εμπόδιο) ανά κάθε μοίρα , και τις απεικονίζουμε στο πρόγραμμα μας. Επίσης κρατάμε τις προηγούμενες τιμές του αισθητήρα σ' έναν άλλο πίνακα για να τα συγκρίνουμε με την επόμενη μέτρηση και σε περίπτωση που έχουμε αλλαγή μεγαλύτερη της τιμής του 35, τότε σε εκείνο το σημείο μπαίνει ένας μικρός κύκλος. Ο υπόλοιπος σχεδιασμός της ''οθόνης'' γίνεται χρησιμοποιώντας κάποια έτοιμα εργαλεία του Processing όπως η εντολή '' rect'' με την οποία σχεδιάζουμε παραλληλόγραμμο, η εντολή ''ellipse'' με την οποία σχεδιάζουμε κύκλους και η εντολή ''text'' με την οποία βάζουμε κείμενο μέσα στην οθόνη.




**Εικόνα 3-13** Οθόνη Ραντάρ



**Εικόνα 3-14**

# 4 . ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

## 4 . 1 Ανάλυση Κώδικα Προγραμματισμού Arduino

*Χρησιμοποιούμε τη βιβλιοθήκη Servo.h*

```
#include <Servo.h>
```

***Δημιουργούμε μια μεταβλητή τύπου servo για να ελέγχουμε το σερβοκινητήρα***

```
int leftRightPos = 0;
```

***Δημιουργία σταθεράς που ισούται με το 10 και δηλώνει το σύνολο των διαβασμάτων***

```
const int numReadings = 10;
```

***Δείκτης του τρέχοντος διαβάσματος***

```
int index = 0;
```

***Το σύνολο διαβασμένων αποστάσεων***

```
int total = 0
```

***Μέσος όρος των αποστάσεων που παίρνουμε σε κάθε θέση του servo***

```
int average = 0;
```

***Δήλωση του echo pin του αισθητήρα ότι θα συνδέεται με το pin 2 του μικροελεγκτή***

```
int echoPin = 2;
```

***Δήλωση του init pin του αισθητήρα ότι θα συνδέεται με το pin 3 του μικροελεγκτή***

```
int initPin = 3;
```

***Μεταβλητή στην οποία θα αποθηκεύουμε τον παλμό***

```
unsigned long pulseTime = 0;
```

***Μεταβλητή στην οποία θα αποθηκεύουμε την απόσταση***

```
unsigned long distance = 0;
void setup(){
```

***Σύνδεση της μεταβλητής leftRightServo τύπου Servo με το pin 9 του μικροελεγκτή***

```
leftRightServo.attach(9);
```

***Κάνουμε το pin(3) που είναι συνδεδεμένο με το init του αισθητήρα ΕΞΟΔΟ***

```
pinMode(initPin, OUTPUT);
```

***Κάνουμε το pin (2) που είναι συνδεδεμένο με το echo του αισθητήρα ΕΙΣΟΔΟ***

```
pinMode(echoPin, INPUT);
```

***Αρχικοποίηση της σειριακής θύρας ώστε να έχουμε επικοινωνία με 9600 bits/s***

```
Serial.begin(9600);
```

***Περιστροφή του σερβοκινητήρα για να πάρουμε τιμές απ τον αισθητήρα***

***Όσο η μεταβλητή leftRightPos είναι μικρότερη του 180 επαναλαμβάνεται η εξής διαδικασία***

```
void loop(){
for(leftRightPos = 0; leftRightPos < 180; leftRightPos++) {  // going left to right;
```

***Παίρνουμε τη τιμή της μεταβλητής leftRightPos***

```
leftRightServo.write(leftRightPos);
```

***Όσο η μεταβλητή index είναι μικρότερη ή ίση της μεταβλητής numReadings παίρνουμε τιμές απ τον αισθητήρα και βρίσκουμε το μέσο όρο αυτών***

```
for (index = 0; index<=numReadings;index++){
```

***Κλείσιμο σήματος του init pin***

```
digitalWrite(initPin, LOW);
Καθυστέρηση 50 Microsecond
delayMicroseconds(50);
Αποστολή σήματος  του init pin
digitalWrite(initPin, HIGH);
Καθυστέρηση 50 Microsecond
delayMicroseconds(50);
Κλείσιμο σήματος  του init pin
digitalWrite(initPin, LOW);
Με την εντολή pulseIn υπολογίζουμε το μήκος  του παλμού σε Microseconds που επιστρέφει με το echo pin και το αποθηκεύουμε  στη μεταβλητή pulseTime
pulseTime = pulseIn(echoPin, HIGH);
Διαιρώντας το αποτέλεσμα που πήραμε με το 58(βλέπε τεχνικά χαρακτηριστικά του αισθητήρα) παίρνουμε την απόσταση απ  το αντικείμενο
distance = pulseTime/58;
Αυξάνουμε την μεταβλητή total κατά distance
total = total + distance;
Καθυστέρηση για 10 δευτερόλεπτα
delay(10);
}
Βρίσκουμε τη μέση τιμή των αποστάσεων που διαβάσαμε
average = total/numReadings;
Μηδενίζουμε τους μετρητές μας για να τους χρησιμοποιήσουμε στην επόμενη μοίρα
if (index >= numReadings)  {
index = 0;
total = 0;
}
Εκτύπωση του γράμματος X (για μοίρες)
Serial.print("X");
Εκτύπωση τιμής της μεταβλητής leftRightPos
Serial.print(leftRightPos);
Εκτύπωση  του γράμματος V (για απόσταση)
Serial.print("V");
Εκτύπωση της τιμής της μεταβλητής average
Serial.println(average);
}
/* Ίδια διαδικασία με πάνω με το σερβοκινητήρα να περιστρέφεται  απ τα δεξιά στα αριστερά. */
for(leftRightPos = 180; leftRightPos > 0; leftRightPos--)
{
leftRightServo.write(leftRightPos);
for (index = 0; index<=numReadings;index++) {
digitalWrite(initPin, LOW);
delayMicroseconds(50);
digitalWrite(initPin, HIGH);
```

```
delayMicroseconds(50);
digitalWrite(initPin, LOW);
pulseTime = pulseIn(echoPin, HIGH);
distance = pulseTime/58;
total = total + distance;
delay(10);
}
average = total/numReadings;
if (index >= numReadings)  {
index = 0;
total = 0;
}
Serial.print("X");
Serial.print(leftRightPos);
Serial.print("V");
Serial.println(average);
}
```

**4 . 2  Ανάλυση Κώδικα Processing**

**Εισαγωγή της βιβλιοθήκης serial**

```
import processing.serial.*;
```

***Δήλωση σειριακής θύρας***

```
Serial myPort;
```

***Δήλωση μεταβλητών x και y  συντεταγμένων για τις κορυφες***

```
float x, y;
```

***Ρύθμιση ακτίνας του αντικειμένου***

```
int radius = 350;
```

***Θέτουμε μια τιμή( αυθαίρετη ) για τη τιμή του πλάτους***

```
int w = 300;
```

***Θέσεις σερβοκινητήρα σε μοίρες***

```
int degree = 0;
```

***Τιμή αισθητήρα (για την απόσταση)***

```
int value = 0;
```

***Ακέραια τιμή στην οποία αποθηκεύουμε  τη κατεύθυνση σάρωσης του σέρβο***

```
int motion = 0;
```

***Δημιουργία πίνακα για αποθήκευση κάθε νέας τιμής του αισθητήρα , για κάθε θέση που παίρνει ο σερβοκινητήρας***

```
int[] newValue = new int[181];
```

***Δημιουργία πίνακα για να κρατήσουμε τις προηγούμενες  τιμές του αισθητήρα***

```
int[] oldValue = new int[181];
```

***Ρύθμιση γραμματοσειράς για το Processing***

```
PFont myFont;
```

***Δήλωση τιμής για ρύθμιση ετικετών της απόστασης***

```
int radarDist = 0;
```
***Δήλωση τιμής για να αγνοήσουμε τις τιμές απ το triggering για τις πρώτες 2 σαρώσεις του servo***
```
int firstRun = 0;
```
***ΔΗΜΙΟΥΡΓΙΑ ΦΟΝΤΟΥ ΚΑΙ ΣΕΙΡΙΑΚΟΥ BUFFER***
```
void setup(){
```
***Ρύθμιση μεγέθους φόντου, χρώματος φόντου και γραμματοσειράς***
***Ρύθμιση μεγέθους***
```
size(750, 450);
Ρύθμιση χρώματος
background (0); // 0 = black
```
***Ρύθμιση γραμματοσειράς***
```
myFont = createFont("verdana", 12);
textFont(myFont);
```
***Ρύθμιση σειριακής θύρας και buffer***
***Ρύθμιση της σειριακής θύρας για αποστολή δεδομένων 9600bits/s***
```
myPort = new Serial(this, Serial.list()[1], 9600);
```
***Ρύθμιση buffer***
```
myPort.bufferUntil('\n');
}
```
***ΣΧΕΔΙΑΣΗ ΟΘΟΝΗΣ***
```
void draw(){
```
***Ρύθμιση ώστε τα σχήματα να είναι μαύρα***
```
fill(0);
```
***Ρύθμιση ώστε τα σχήματα να μην έχουν περίγραμμα***
```
noStroke();
```
***Σχεδιασμός έλλειψης με πλάτος και ύψος 750***
```
ellipse(radius, radius, 750, 750);
```
***Ρύθμιση ώστε το παραλληλόγραμμο που θα σχεδιαστεί παρακάτω να σχεδιαστεί στο κέντρο***
```
rectMode(CENTER);
```
***Σχεδιασμός παραλληλόγραμμου***
```
rect(350,402,800,100);
```
***ΕΛΕΓΧΟΣ ΓΙΑ ΤΗ ΚΑΤΕΥΘΥΝΣΗ ΠΕΡΙΣΤΡΟΦΗΣ ΤΟΥ ΣΕΡΒΟΚΙΝΗΤΗΡΑ***
***Αν η μεταβλητή degree είναι μεγαλύτερη ίση του 179 τότε η μεταβλητή motion παίρνει τη τιμή 1 και ο σερβοκινητήρας περιστρέφεται απ τα δεξιά στα αριστερά***
```
 if (degree >= 179){
motion = 1;
```
***Χρησιμοποιούμε την τριγωνομετρία για να δημιουργήσουμε σημεία γύρω απ' τον κύκλο. Παίρνουμε το άθροισμα της ακτίνας και του συνημίτονου απ' τη θέση του σερβοκινητήρα σε ακτίνια. Αφού τα ακτίνια 0 ξεκινούν απ' τις $90^{o}$ προσθέτουμε 180 για να ξεκινήσουμε απ ΄ τα αριστερά. Προσθέτουμε κάθε φορά κατά ένα ανάμεσα στις επαναλήψεις ώστε να είναι συγχρονισμένο με τη κίνηση του σέρβο. Το συνημίτονο υπολογίζει το και χ το ημίτονο το y.***

***Ρύθμιση του πάχους των γραμμών***
***Έλεγχος αν η κίνηση είναι απ τα αριστερά προς τα δεξιά***

```
if (motion == 0){
```
***Επαναληπτική διαδικασία κατά την οποία σχεδιάζουμε 20 γραμμές με ''ξεθώριασμα'' του χρώματος σε κάθε αλλαγή του i (είναι οι γραμμές που φαίνονται όταν σκανάρει ο αισθητήρας***
```
for (int i = 0; i <= 20; i++){
```
***Με την εντολή stroke ρυθμίζουμε το χρώμα της γραμμής να είναι πράσινο και να αυξανεται η τιμή του G ανάλογα με το i***
```
stroke(0, (10*i), 0);
```
***Σχεδιασμός γραμμής με τη διαδικασία που εξηγήσαμε παραπάνω.***
```
line(radius, radius, radius + cos(radians(degree+(180+i)))*w, radius + sin(radians(degree+(180+i)))*w);  //γραμμή(αρχή x, αρχή y, τέλος x, τέλος y)
```
***}***

***Κατεύθυνση σερβοκινητήρα απ τα δεξιά στα αριστερά Ίδιος κώδικα και ίδια διαδικασία με την προηγούμενη επαναληπτική διαδικασία***
```
} else{
for (int i = 20; i >= 0; i--){
stroke(0,200-(10*i), 0);
line(radius, radius, radius + cos(radians(degree+(180+i)))*w, radius + sin(radians(degree+(180+i)))*w);
}
}
```
***Ρύθμιση των σχημάτων που δημιουργήθηκαν απ τις τιμές του αισθητήρα***

***Αφαιρούμε το περίγραμμα***
```
noStroke();
```
***Πρώτο σκανάρισμα***

***Γεμίζουμε με πράσινο χρώμα το σχήμα (G=50)***
```
fill(0,50,0);
```
***Δημιουργία σχεδιασμού του σχήματος***
```
beginShape();
```
***Για κάθε μία μοίρα παίρνουμε τις συντεταγμένες χ και y των κορυφών και στη συνέχεια σχεδιάζουμε τις κορυφές***
```
for (int i = 0; i < 180; i++){
x = radius + cos(radians((180+i)))*((oldValue[i]*2));
y = radius + sin(radians((180+i)))*((oldValue[i]*2));
vertex(x, y);
}
endShape();
```
***Δεύτερο σκανάρισμα***

***Γεμίζουμε με πράσινο χρώμα το σχήμα (G=110)***
```
fill(0,110,0);
beginShape();
```
***Για κάθε μία μοίρα παίρνουμε τις συντεταγμένες χ και y των κορυφών και στη συνέχεια σχεδιάζουμε τις κορυφές***
```
for (int i = 0; i < 180; i++){
x = radius + cos(radians((180+i)))*(newValue[i]*2);
y = radius + sin(radians((180+i)))*(newValue[i]*2);
```

```
vertex(x, y);
}
endShape();
```
***Μέσος όρος των δύο σκαναρισμάτων***

***Γεμίζουμε με πράσινο χρώμα το σχήμα (G=170)***

```
fill(0,170,0);
```
***Για κάθε μία μοίρα παίρνουμε τις συντεταγμένες χ και y των κορυφών και στη συνέχεια σχεδιάζουμε τις κορυφές***

```
beginShape();
for (int i = 0; i < 180; i++) {
x = radius + cos(radians((180+i)))*(((newValue[i]+oldValue[i])/2)*2);
y = radius + sin(radians((180+i)))*(((newValue[i]+oldValue[i])/2)*2);
vertex(x, y);
}
endShape();
```
***Αν μετά τα δύο σκαναρίσματα υπάρχει αλλαγή των συντεταγμένων στα επόμενα σκαναρίσματα τότε στα σημεία διαφοράς σχεδιάζουμε ένα μικρό κύκλο για να δείξουμε ότι σ' εκείνο το σημείο είχαμε κίνηση.***

***Αν η τιμή της μεταβλητής firstRun είναι μεγαλύτερη του 360 τότε***

```
if (firstRun >= 360){
```
***Ρύθμιση ώστε το χρώμα της γραμμής να είναι κόκκινο***

```
stroke(150,0,0);
```
***Ρύθμιση ώστε το πάχος της γραμμής να είναι 1***

```
strokeWeight(1);
```
***Δεν έχουμε γέμισμα του σχήματος***

```
noFill();
```
***Επαναληπτική διαδικασία κατά την οποία ελέγχουμε αν η διαφορά νέας και παλιάς τιμής είναι μεγαλύτερη απ ' το 35 για κάθε θέση που παίρνει ο σέρβο. Αν ισχύει η συνθήκη τότε αποθηκεύουμε τις συντεταγμένες στις μεταβλητές x και y και σχηματίζουμε κύκλο με συντεταγμένες x y και με μέγεθος ύψους και πλάτους 10***

```
for (int i = 0; i < 180; i++) {
if (oldValue[i] - newValue[i] > 35 || newValue[i] - oldValue[i] > 35) {
 x = radius + cos(radians((180+i)))*(newValue[i]*2);
y = radius + sin(radians((180+i)))*(newValue[i]*2);
ellipse(x, y, 10, 10);
}
}
}
```
***Σχεδιασμός των δακτυλίων που δείχνουν την απόσταση ( 6 δακτύλια)***

```
for (int i = 0; i <=6; i++){
noFill();
```
***Πάχος γραμμής 1***

```
strokeWeight(1);
```
***Γέμισμα γραμμής περιγράμματος, αλλάζει ανάλογα με το i στις αποχρώσεις του πρασίνου***

```
    stroke(0, 255-(30*i), 0);
```
***Σχεδιασμός δακτυλίου με συντεταγμένες τη μεταβλητή radius και μέγεθος ύψους και πλάτους 100επί το i***
```
    ellipse(radius, radius, (100*i), (100*i));
```
***Γέμισμα χρώματος Πρασίνου με τιμή 100***
```
    fill(0, 100, 0);
```
***Χωρίς περίγραμμα***
```
    noStroke();
```
***Εκτύπωση τιμής radarDist στις συντεταγμένες 380 και 305 - δύο φορές τη τιμή της radarDist με ύψος και πλάτος 50***
```
    text(Integer.toString(radarDist+25), 380, (305-(radarDist*2)), 50, 50);
```
***Αυξάνεται η μεταβλητή radarDist κατά 25***
```
    radarDist+=25;
    }
```
***Μηδενισμός της μεταβλητής radarDIst***
```
    radarDist = 0;
```
***Σχεδιασμός γραμμών ανά $30^o$ βάζοντας και τις τιμές των γωνιών (6 γραμμές)***
```
    for (int i = 0; i <= 6; i++) {
```
***Μέγεθος γραμμής 1***
```
    strokeWeight(1);
```
***Χρώμα γραμμής περιγράμματος με Πράσινο χρώμα τιμή 55***
```
    stroke(0, 55, 0);
```
***Σχεδιασμός γραμμής με τον ίδιο τρόπο που είχαμε σχεδιάσει και παραπάνω***
```
    line(radius, radius, radius + cos(radians(180+(30*i)))*w, radius + sin(radians(180+(30*i)))*w);
```
***Γέμισμα γραμμής περιγράμματος με Πράσινο και τιμή 55***
```
    fill(0, 55, 0);
```
***Χωρίς περίγραμμα***
```
    noStroke();
       if (180+(30*i) >= 300){
```
***Εκτύπωση τιμών γύρω απ το δακτύλιο για τις μοιρες***
```
    text(Integer.toString(180+(30*i)), (radius+10) + cos(radians(180+(30*i)))*(w+10), (radius+10) +
sin(radians(180+(30*i)))*(w+10), 25,50);
    } else {
    text(Integer.toString(180+(30*i)), radius + cos(radians(180+(30*i)))*w, radius +
sin(radians(180+(30*i)))*w, 60,40);
    }
    }
```
***Εισαγωγή των τιμών και πληροφοριών***
***Δημιουργία ενός παραλληλογράμμου κάτω απ το σχήμα του αισθητήρα με μαύρο φόντο χωρις περιγραμμα***
```
    noStroke();
    fill(0);
    rect(350,402,800,100);
```
***Δημιουργία κειμένου Degrees : '' τιμή της μεταβλητής degrees '' η οποία έχει μετατραπεί σε string***

```
    fill(0, 100, 0);
    text("Degrees: "+Integer.toString(degree), 100, 380, 100, 50);
```
***Δημιουργία κειμένου Distance: '' τιμή της μεταβλητής Distance '' η οποία έχει μετατραπεί σε string***
```
    text("Distance: "+Integer.toString(value), 100, 400, 100, 50);
```
***Δημιουργία 3 μικρών τετραγώνων σε ξεκινώντας απ' ανοιχτή απόχρωση του πρασίνου και καταλήγοντας σε πιο σκούρα απόχρωση . Εγγραφή των λέξεων First sweep ,Second sweep, Average δίπλα απ τα τετράγωνα που δημιουργήθηκαν. Δημιουργία κύκλου κόκκινου περιγράμματος και δίπλα απ το κύκλο εγγραφή της λέξης Motion.***
```
    text("Screen Key:", 100, 50, 150, 50);
    fill(0,50,0);
    rect(30,53,10,10);
    text("First sweep", 115, 70, 150, 50);
    fill(0,110,0);
    rect(30,73,10,10);
    text("Second sweep", 115, 90, 150, 50);
    fill(0,170,0);
    rect(30,93,10,10);
    text("Average", 115, 110, 150, 50);
    noFill();
    stroke(150,0,0);
    strokeWeight(1);
    ellipse(29, 113, 10, 10);
    fill(150,0,0);
    text("Motion", 115, 130, 150, 50);
```
***}***
***Δημιουργία συνάρτησης για να πάρουμε τις τιμές απ τη θύρα***
```
    void serialEvent (Serial myPort){
```
***Λαμβάνουμε τιμές απ τη θύρα μέχρι να συναντήσουμε μια καινούργια γραμμή***
```
    String xString = myPort.readStringUntil('\n');
```
***Έλγχος για να μη είναι κενά τα δεδομένα***
```
    if (xString != null){
```
***Απομάκρυνση κενών χαρακτήρων***
```
    xString = trim(xString);
```
***Παίρνουμε την τιμή της θέσης του σερβοκινητήρα***
```
    String getX = xString.substring(1, xString.indexOf("V"));
```
***Παίρνουμε την τιμή του αισθητήρα***
```
    String getV = xString.substring(xString.indexOf("V")+1, xString.length());
```
***Αποθηκεύουμε τις τιμές που πήραμε σε μεταβλητές***
```
    degree = Integer.parseInt(getX);
    value = Integer.parseInt(getV);
```
***/*Αν οι τιμές που παίρνουμε είναι εκτός εμβέλειας του αισθητήρα τότε τα μετατρέπουμε είτε στη μεγαλύτερη τιμή (150) είτε στη μικρότερη(20) για καλύτερη απεικόνιση*/***
```
    if (value > 150){
    value = 150;
```

```
}
if (value < 20) {
value = 20;
}
```
***Αποθηκεύουμε τις θέσεις του servo σε πίνακα***
```
oldValue[degree] = newValue[degree];
newValue[degree] = value;
```
***Δημιουργία μετρητή για να μπορέσουμε να βρούμε τη κίνηση (motion) μετά τις δύο σαρώσεις***
```
firstRun++;
if (firstRun > 360){
firstRun = 360;
}
}
}
```

**ΒΙΒΛΙΟΓΡΑΦΙΑ – ΠΗΓΕΣ**


- www.arduinohnev.blogspot.com
- http://arduino.cc
- www.freeduino.org
- http://playground.arduino.cc/interfacing/processing
- **http://luckylarry.co.uk/**
- **http://processing.org/**
- **http://www.microplanet.gr**
- **http://www.t12online.com**
- **http://www.wiring.org.co/**
- Evans, B., (2007), Arduino Programming Notebook, 2st edition, California
- Greenberg, I., (2007), Processing: Creative Coding and Computational Art, 1st edition, New York
- Shiffman, D., (2008) Learning Processing: A Beginner"s Guide to Programming Images, Animation and Interaction, 1st edition, USAMike Riley (2012) , Programming Your Home
- Noble, J., (2009), Programming interactivity: A Designer"s Guide to Processing, Arduino and openFramework, 1st edition, California: O" Reilly Media
- *Banzi, M., (2009). Getting Started with Arduino, 1st edition, Italy*

## ΠΑΡΑΘΕΣΗ ΚΩΔΙΚΑ

## ARDUINO

```
#include <Servo.h>
Servo leftRightServo;        // set a variable to map the servo
int leftRightPos = 0;        // set a variable to store the servo position
const int numReadings = 10;   // set a variable for the number of readings to take
int index = 0;               // the index of the current reading
int total = 0;               // the total of all readings
int average = 0;             // the average
int echoPin = 2;             // the SRF05's echo pin
int initPin = 3;             // the SRF05's init pin
unsigned long pulseTime = 0;  // variable for reading the pulse
unsigned long distance = 0;   // variable for storing distance

/* setup the pins, servo and serial port */
void setup() {
  leftRightServo.attach(9);
  // make the init pin an output:
  pinMode(initPin, OUTPUT);
  // make the echo pin an input:
  pinMode(echoPin, INPUT);
  // initialize the serial port:
  Serial.begin(9600);
}

/* begin rotating the servo and getting sensor values */
void loop() {
  for(leftRightPos = 0; leftRightPos < 180; leftRightPos++) {  // going left to right.
    leftRightServo.write(leftRightPos);
     for (index = 0; index<=numReadings;index++) {          // take x number of readings from the
sensor and average them
       digitalWrite(initPin, LOW);
       delayMicroseconds(50);
       digitalWrite(initPin, HIGH);                    // send signal
       delayMicroseconds(50);                          // wait 50 microseconds for it to return
       digitalWrite(initPin, LOW);                     // close signal
       pulseTime = pulseIn(echoPin, HIGH);                 // calculate time for signal to return
       distance = pulseTime/58;                        // convert to centimetres
       total = total + distance;                      // update total
       delay(10);
     }
```

```
   average = total/numReadings;                    // create average reading

   if (index >= numReadings)  {                    // reset the counts when at the last item of the
array
     index = 0;
     total = 0;
   }
   Serial.print("X");                              // print leading X to mark the following value as degrees
   Serial.print(leftRightPos);                     // current servo position
   Serial.print("V");                              // preceeding character to separate values
   Serial.println(average);                        // average of sensor readings
  }
  /*
  start going right to left after we got to 180 degrees
  */
  for(leftRightPos = 180; leftRightPos > 0; leftRightPos--) {  // going right to left
   leftRightServo.write(leftRightPos);
   for (index = 0; index<=numReadings;index++) {
     digitalWrite(initPin, LOW);
     delayMicroseconds(50);
     digitalWrite(initPin, HIGH);
     delayMicroseconds(50);
     digitalWrite(initPin, LOW);
     pulseTime = pulseIn(echoPin, HIGH);
     distance = pulseTime/58;
     total = total + distance;
     delay(10);
   }
   average = total/numReadings;
   if (index >= numReadings)  {
     index = 0;
     total = 0;
   }
   Serial.print("X");
   Serial.print(leftRightPos);
   Serial.print("V");
   Serial.println(average);
  }
}
```

## PROCESSING

```
import processing.serial.*;     // import serial library
Serial myPort;                  // declare a serial port
float x, y;                     // variable to store x and y co-ordinates for vertices
int radius = 350;               // set the radius of objects
int w = 300;                    // set an arbitary width value
int degree = 0;                 // servo position in degrees
int value = 0;                  // value from sensor
int motion = 0;                 // value to store which way the servo is panning
int[] newValue = new int[181];  // create an array to store each new sensor value for each servo
position
int[] oldValue = new int[181];  // create an array to store the previous values.
PFont myFont;                   // setup fonts in Processing
int radarDist = 0;              // set value to configure Radar distance labels
int firstRun = 0;               // value to ignore triggering motion on the first 2 servo sweeps

/* create background and serial buffer */
void setup(){
  // setup the background size, colour and font.
  size(750, 450);
  background (0); // 0 = black
  myFont = createFont("verdana", 12);
  textFont(myFont);
  // setup the serial port and buffer
  myPort = new Serial(this, Serial.list()[1], 9600);
  myPort.bufferUntil('\n');
}

/* draw the screen */
void draw(){
  fill(0);                      // set the following shapes to be black
  noStroke();                   // set the following shapes to have no outline
```

```
 ellipse(radius, radius, 750, 750);    // draw a circle with a width/ height = 750 with its center position (x and y) set by the radius
 rectMode(CENTER);                  // set the following rectangle to be drawn around its center
 rect(350,402,800,100);              // draw rectangle (x, y, width, height)
 if (degree >= 179) {               // if at the far right then set motion = 1/ true we're about to go right to left
   motion = 1;                    // this changes the animation to run right to left
 }
 if (degree <= 1) {                 // if servo at 0 degrees then we're about to go left to right
   motion = 0;                    // this sets the animation to run left to right
 }
 /* setup the radar sweep */
 strokeWeight(7);                   // set the thickness of the lines
 if (motion == 0) {                 // if going left to right
   for (int i = 0; i <= 20; i++) {     // draw 20 lines with fading colour each 1 degree further round than the last
     stroke(0, (10*i), 0);           // set the stroke colour (Red, Green, Blue) base it on the the value of i
     line(radius, radius, radius + cos(radians(degree+(180+i)))*w, radius + sin(radians(degree+(180+i)))*w); // line(start x, start y, end x, end y)
   }
 } else {                           // if going right to left
   for (int i = 20; i >= 0; i--) {     // draw 20 lines with fading colour
     stroke(0,200-(10*i), 0);         // using standard RGB values, each between 0 and 255
     line(radius, radius, radius + cos(radians(degree+(180+i)))*w, radius + sin(radians(degree+(180+i)))*w);
   }
 }
 /* Setup the shapes made from the sensor values */
 noStroke();                        // no outline
 /* first sweep */
 fill(0,50,0);                      // set the fill colour of the shape (Red, Green, Blue)
 beginShape();                      // start drawing shape
```

```
  for (int i = 0; i < 180; i++) {     // for each degree in the array
    x = radius + cos(radians((180+i)))*((oldValue[i]*2)); // create x coordinate
    y = radius + sin(radians((180+i)))*((oldValue[i]*2)); // create y coordinate
    vertex(x, y);                  // plot vertices
  }
endShape();                        // end shape
/* second sweep */
fill(0,110,0);
beginShape();
  for (int i = 0; i < 180; i++) {
    x = radius + cos(radians((180+i)))*(newValue[i]*2);
    y = radius + sin(radians((180+i)))*(newValue[i]*2);
    vertex(x, y);
  }
endShape();
/* average */
fill(0,170,0);
beginShape();
  for (int i = 0; i < 180; i++) {
    x = radius + cos(radians((180+i)))*(((newValue[i]+oldValue[i])/2)*2); // create average
    y = radius + sin(radians((180+i)))*(((newValue[i]+oldValue[i])/2)*2);
    vertex(x, y);
  }
endShape();
/* if after first 2 sweeps, highlight motion with red circle*/
if (firstRun >= 360) {
  stroke(150,0,0);
  strokeWeight(1);
  noFill();
    for (int i = 0; i < 180; i++) {
      if (oldValue[i] - newValue[i] > 35 || newValue[i] - oldValue[i] > 35) {
        x = radius + cos(radians((180+i)))*(newValue[i]*2);
        y = radius + sin(radians((180+i)))*(newValue[i]*2);
```

```
        ellipse(x, y, 10, 10);
       }
      }
    }
  }
  /* set the radar distance rings and out put their values, 50, 100, 150 etc.. */
  for (int i = 0; i <=6; i++){
    noFill();
    strokeWeight(1);
    stroke(0, 255-(30*i), 0);
    ellipse(radius, radius, (100*i), (100*i));
    fill(0, 100, 0);
    noStroke();
    text(Integer.toString(radarDist+25), 380, (305-(radarDist*2)), 50, 50); // change this to
measure up to 150cm
    radarDist+=25;
  }
  radarDist = 0;
  /* draw the grid lines on the radar every 30 degrees and write their values 180, 210, 240 etc..
*/
  for (int i = 0; i <= 6; i++) {
    strokeWeight(1);
    stroke(0, 55, 0);
    line(radius, radius, radius + cos(radians(180+(30*i)))*w, radius + sin(radians(180+(30*i)))*w);
    fill(0, 55, 0);
    noStroke();
    if (180+(30*i) >= 300) {
      text(Integer.toString(180+(30*i)), (radius+10) + cos(radians(180+(30*i)))*(w+10), (radius+10)
+ sin(radians(180+(30*i)))*(w+10), 25,50);
    } else {
      text(Integer.toString(180+(30*i)), radius + cos(radians(180+(30*i)))*w, radius +
sin(radians(180+(30*i)))*w, 60,40);
    }
  }
```

```
  /* Write information text and values. */
  noStroke();
  fill(0);
  rect(350,402,800,100);
  fill(0, 100, 0);
  text("Degrees: "+Integer.toString(degree), 100, 380, 100, 50);        // use Integet.toString to
convert numeric to string as text() only outputs strings
  text("Distance: "+Integer.toString(value), 100, 400, 100, 50);        // text(string, x, y, width,
height)

  fill(0);
  rect(70,60,150,100);
  fill(0, 100, 0);
  text("Screen Key:", 100, 50, 150, 50);
  fill(0,50,0);
  rect(30,53,10,10);
  text("First sweep", 115, 70, 150, 50);
  fill(0,110,0);
  rect(30,73,10,10);
  text("Second sweep", 115, 90, 150, 50);
  fill(0,170,0);
  rect(30,93,10,10);
  text("Average", 115, 110, 150, 50);
  noFill();
  stroke(150,0,0);
  strokeWeight(1);
  ellipse(29, 113, 10, 10);
  fill(150,0,0);
  text("Motion", 115, 130, 150, 50);
}

/* get values from serial port */
void serialEvent (Serial myPort) {
```

```
  String xString = myPort.readStringUntil('\n');  // read the serial port until a new line
    if (xString != null) {  // if theres data in between the new lines
      xString = trim(xString); // get rid of any whitespace just in case
      String getX = xString.substring(1, xString.indexOf("V")); // get the value of the servo position
      String getV = xString.substring(xString.indexOf("V")+1, xString.length()); // get the value of
the sensor reading
      degree = Integer.parseInt(getX); // set the values to variables
      value = Integer.parseInt(getV);
        /*
        If our values are outside either end of the sensors range then convert them to the max/min
for a better display without the spikes
        */
        if (value > 150) {
          value = 150;
        }
        if (value < 20) {
          value = 20;
        }
      oldValue[degree] = newValue[degree]; // store the values in the arrays.
      newValue[degree] = value;
        /* sets a counter to allow for the first 2 sweeps of the servo */
      firstRun++;
      if (firstRun > 360) {
        firstRun = 360; // keep the value at 360
      }
  }
}
```