Τμήμα Μηχανολόγων Μηχανικών Τ.Ε. Τ.Ε.Ι. Κρήτης

Δρ. Φασουλάς Γιάννης

jfasoulas@staff.teicrete.gr

Μηχατρονικά Συστήματα Ι

Εργαστήριο Συστημάτων Ελέγχου & Ρομποτικής
Σχολή Τεχνολογικών Εφαρμογών
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Ηράκλειο Κρήτης, Ελλάς

Η υπολογιστική πλατφόρμα Arduino

Η υπολογιστική πλατφόρμα Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει μικρή εμπειρία στον προγραμματισμό και στους μικροελεγκτές.

Πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια, καθώς και το software που χρειάζεται για την λειτουργία του, διανέμονται ελεύθερα και δωρεάν ώστε να μπορεί να κατασκευαστεί από τον καθένα. Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Μικροελεγκτής – η καρδιά του Arduino

Το Arduino βασίζεται στον ATmega328, έναν 8-bit RISC μικροελεγκτή, τον οποίο χρονίζει στα 16MHz. Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- **2Kb μνήμης SRAM** που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά σας για να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή αν γίνει reset.
- **1Kb μνήμης EEPROM** η οποία μπορεί να χρησιμοποιηθεί για «ωμή» εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματά σας κατά το runtime. Σε αντίθεση με την SRAM,

η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset οπότε είναι το ανάλογο του σκληρού δίσκου.

- **32Kb μνήμης Flash**, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των δικών σας προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή σας. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset. Επίσης, ενώ η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime μέσα από τα προγράμματά σας, λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), έχει σχεδιαστεί μια βιβλιοθήκη που επιτρέπει την χρήση όσου χώρου περισσεύει (30Kb μείον το μέγεθος του προγράμματός σας σε μεταγλωττισμένη μορφή).

Διαφορές στις προτεινόμενες εκδόσεις του Arduino

Το **Arduino Diecimila, UNO** έχει ουσιαστικά δύο βασικές διαφορές με το Duemilanove:

- Βασίζεται στον μικροελεγκτή ATmega168, ο οποίος διαθέτει ακριβώς την μισή μνήμη από τον ATmega328, δηλαδή 1Kb SRAM, 512bytes EEPROM και 16Kb Flash (14 ελεύθερα λόγω του bootloader).
- Δεν επιλέγει αυτόματα μεταξύ της εξωτερικής τροφοδοσίας και της τροφοδοσίας μέσω της θύρας USB. Το Diecimila διαθέτει ειδικό jumper με το οποίο μπορείτε να επιλέξετε χειροκίνητα την πηγή τροφοδοσίας.

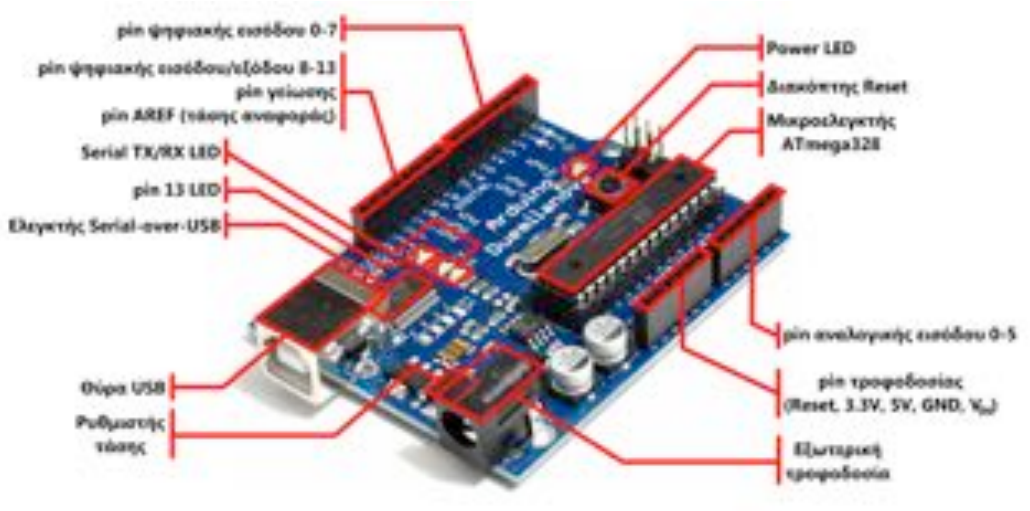
Το **Arduino Mega** είναι η πιο εξελιγμένη έκδοση με τον μικροελεγκτή ATmega1280 και αρκετά μεγαλύτερο μέγεθος. Οι διαφορές του από το Duemilanove είναι:

- Τετραπλάσια μνήμη (8Kb SRAM, 4Kb EEPROM, 128Kb Flash).
- 40 επιπλέον ψηφιακά pin εισόδου/εξόδου (σύνολο 54)
- 10 επιπλέον pin αναλογικής εισόδου (σύνολο 16)
- Υποστήριξη ψευδοαναλογικής εξόδου PWM σε 8 ακόμα ψηφιακά pin (σύνολο 14 PWM pin)
- Υποστήριξη εξωτερικού interrupt σε 4 ακόμα ψηφιακά pin (σύνολο 6 interrupt)
- 3 επιπλέον σειριακά interface (σύνολο 4) από τα οποία το ένα προωθείται στον ελεγκτή Serial-Over-USB όπως στο Duemilanove για σύνδεση με τον υπολογιστή.

Σημειώστε ότι το Arduino Mega είναι συμβατό με τα περισσότερα shield που έχουν κυκλοφορήσει για το Arduino αλλά όχι με το Ethernet Shield, το οποίο είναι ένα αρκετά σημαντικό μειονέκτημα για όσους θέλουν να φτιάξουν εφαρμογές με πρόσβαση στο internet ή σε κάποιο άλλο δίκτυο. Από τις ανεπίσημες εκδόσεις, το **Freeduino 1.16** και το **Seeeduino** βασίζονται στο Diecimila οπότε ισχύουν οι ίδιες διαφορές που έχει αυτό με το Duemilanove. Το Freeduino είναι ακριβής κλώνος του Diecimila, ενώ το Seeeduino είναι μια βελτιωμένη έκδοση του Diecimila με κύρια διαφορά την προσθήκη 2 επιπλέον pin αναλογικής εισόδου.

Είσοδοι – Έξοδοι

Καταρχήν το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.



Επιπλέον, στην πάνω πλευρά του Arduino βρίσκονται **14 θηλυκά pin**, αριθμημένα **από 0 ως 13**, που μπορούν να λειτουργήσουν **ως ψηφιακές είσοδοι και έξοδοι**. Λειτουργούν **στα 5V** και καθένα μπορεί να παρέχει ή να δεχτεί **το πολύ 40mA**. Ως ψηφιακή έξοδος, ένα από αυτά τα pin μπορεί να τεθεί από το πρόγραμμά σας σε κατάσταση HIGH ή LOW, οπότε το Arduino θα ξέρει αν πρέπει να διοχετεύσει ή όχι ρεύμα στο συγκεκριμένο pin. Με αυτόν τον τρόπο μπορείτε λόγου χάρη να ανάψετε και να σβήσετε ένα LED που έχετε συνδέσει στο συγκεκριμένο pin. Αν πάλι ρυθμίσετε ένα από αυτά τα pin ως ψηφιακή είσοδο μέσα από το πρόγραμμά σας, μπορείτε με την κατάλληλη εντολή να διαβάσετε την κατάστασή του (HIGH ή LOW) ανάλογα με το αν η εξωτερική συσκευή που έχετε συνδέσει σε αυτό το pin διοχετεύει ή όχι ρεύμα στο pin (με αυτόν τον τρόπο λόγου χάρη μπορείτε να «διαβάζετε» την κατάσταση ενός διακόπτη). **Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές είσοδοι/έξοδοι έχουν και δεύτερη λειτουργία.** Συγκεκριμένα:

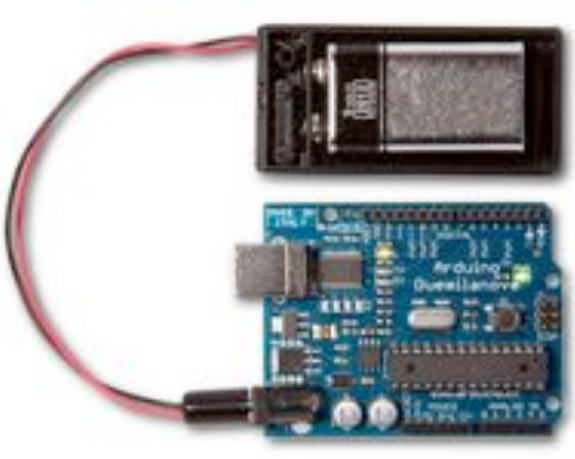
- **Τα pin 0 και 1 λειτουργούν ως RX και TX** της σειριακής όταν το πρόγραμμά σας ενεργοποιεί την σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμά σας στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά σας ενεργοποιήσετε το σειριακό interface, χάνετε 2 ψηφιακές εισόδους/εξόδους.
- **Τα pin 2 και 3 λειτουργούν και ως εξωτερικά interrupt** (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορείτε να τα ρυθμίσετε μέσα από το πρόγραμμά σας 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 interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- **Τα pin 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, μπορείτε να συνδέσετε λόγου χάρη ένα LED σε κάποιο από αυτά τα pin και να ελέγξετε πλήρως την φωτεινότητά του με **ανάλυση 8bit (256 καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο)** αντί να έχετε απλά την δυνατότητα αναμμένο-σβηστό που παρέχουν οι υπόλοιπές ψηφιακές έξοδοι. Είναι σημαντικό να καταλάβετε ότι τ**ο PWM δεν είναι πραγματικά αναλογικό σήμα** 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. Η συχνότητα του PWM στα περισσότερα pin είναι 490Hz. Στον Arduino Uno στα pin 5 και 6 η συχνότητα του PWM είναι 980Hz.

Στην κάτω πλευρά του Arduino, με τη **σήμανση ANALOG IN**, θα βρείτε μια ακόμη σειρά από **6 pin**, 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 ενσωματωμένο στον μικροελεγκτή**. Για παράδειγμα, μπορείτε να τροφοδοτήσετε ένα από αυτά με μια τάση την οποία μπορείτε να κυμάνετε με ένα ποτενσιόμετρο από 0V ως μια τάση αναφοράς Vref η οποία, αν δεν κάνετε κάποια αλλαγή είναι προρυθμισμένη στα 5V. Τότε, μέσα από το πρόγραμμά σας μπορείτε να «διαβάσετε» την τιμή του pin ως ένα ακέραιο αριθμό **ανάλυσης 10-bit**, **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 τάση επιθυμείτε (μεταξύ 2 και 5V) τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας.** Έτσι, αν τροφοδοτήσετε το pin AREF με 3.3V και στην συνέχεια δοκιμάσετε να διαβάσετε κάποιο pin αναλογικής εισόδου στο οποίο εφαρμόζετε τάση 1.65V, το Arduino θα σας επιστρέψει την τιμή 512. Τέλος, καθένα από τα 6 αυτά pin, 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 pin μετονομάζονται από 0~5 σε 14~19 αντίστοιχα.

Τροφοδοσία

Το Arduino μπορεί να τροφοδοτηθεί με ρεύμα είτε από τον υπολογιστή **μέσω της σύνδεσης USB**, είτε από **εξωτερική τροφοδοσία** που παρέχεται μέσω μιας υποδοχής φις των 2.1mm (θετικός πόλος στο κέντρο) και βρίσκεται στην κάτω-αριστερή γωνία του Arduino.



Για να μην υπάρχουν προβλήματα, η εξωτερική τροφοδοσία πρέπει να είναι **από 7 ως 12V** και μπορεί να προέρχεται από ένα κοινό μετασχηματιστή του εμπορίου, από μπαταρίες ή οποιαδήποτε άλλη **πηγή DC**.

Δίπλα από τα pin αναλογικής εισόδου, υπάρχει μια ακόμα συστοιχία από 6 pin με την σήμανση POWER. Η λειτουργία του καθενός έχει ως εξής:

- Το πρώτο, με την **ένδειξη RESET**, όταν γειωθεί (σε οποιοδήποτε από τα 3 pin με την ένδειξη GND που υπάρχουν στο Arduino) έχει ως αποτέλεσμα την επανεκκίνηση του Arduino.
- Το δεύτερο, με την **ένδειξη 3.3V**, μπορεί να τροφοδοτήσει τα εξαρτήματά σας 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**.
- Το τρίτο, με την **ένδειξη 5V**, μπορεί να τροφοδοτήσει τα εξαρτήματά σας 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.
- Το τέταρτο και το πέμπτο pin, με την **ένδειξη GND**, είναι φυσικά γειώσεις.
- Το έκτο και τελευταίο pin, με την **ένδειξη Vin έχει διπλό ρόλο**. Σε συνδυασμό με το pin γείωσης δίπλα του, μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας του Arduino, στην περίπτωση που δεν σας βολεύει να χρησιμοποιήσετε την υποδοχή του φις των 2.1mm. Αν όμως έχετε ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, μπορείτε να χρησιμοποιήσετε αυτό το pin για να τροφοδοτήσετε εξαρτήματα με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

Ενσωματωμένα κουμπιά και LED

Πάνω στην πλακέτα του Arduino υπάρχει ένας διακόπτης micro-switch και 4 μικροσκοπικά LED

επιφανειακής στήριξης. Η λειτουργία του διακόπτη (που έχει την σήμανση RESET) και του ενός LED με την σήμανση POWER είναι μάλλον προφανής. Τα **δύο LED με τις σημάνσεις TX και RX,** χρησιμοποιούνται ως ένδειξη λειτουργίας του σειριακού interface, καθώς ανάβουν όταν το Arduino στέλνει ή λαμβάνει (αντίστοιχα) δεδομένα μέσω USB. Σημειώστε ότι τα LED 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 του αναθέσετε να αναβοσβήνει ένα LED (θα το δείτε αυτό στην συνέχεια όταν θα φτιάξετε την πρώτη εφαρμογή σας). Για να μπορείτε να το κάνετε αυτό από την πρώτη στιγμή, χωρίς να συνδέσετε τίποτα πάνω στο Arduino, οι κατασκευαστές του σκέφτηκαν να ενσωματώσουν ένα LED στην πλακέτα, το οποίο σύνδεσαν στο ψηφιακό pin 13. Έτσι, ακόμα και αν δεν έχετε συνδέσει τίποτα πάνω στο φυσικό pin 13, αναθέτοντάς του την τιμή HIGH μέσα από το πρόγραμμά σας, θα ανάψει αυτό το ενσωματωμένο LED.

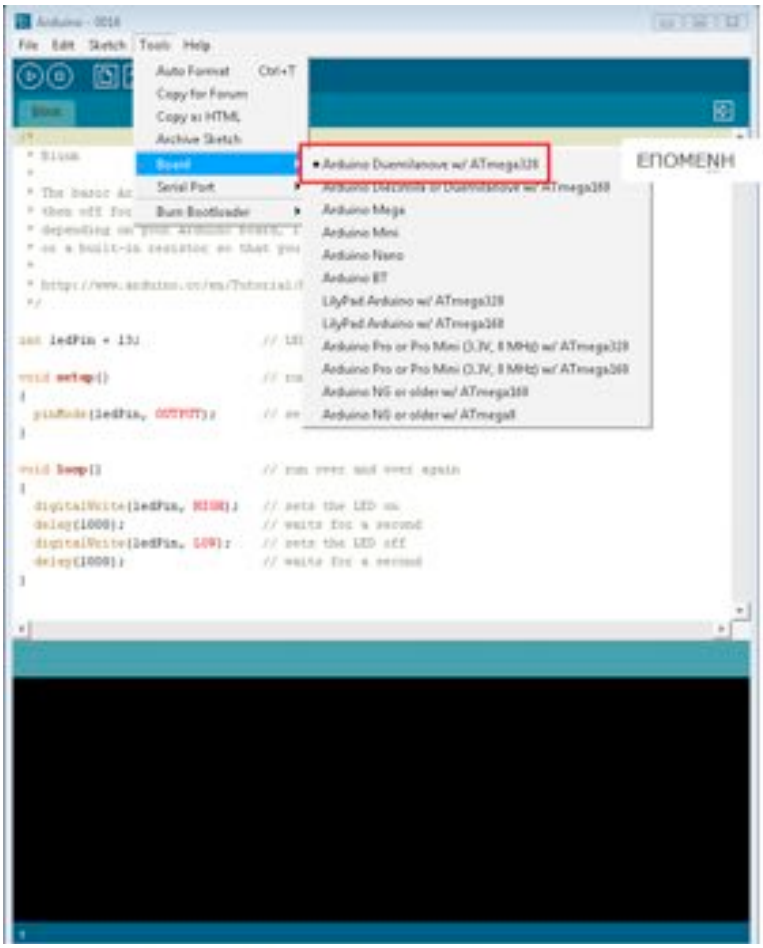
Arduino IDE (Integrated Design Environment) και σύνδεση με τον υπολογιστή

Ότι χρειάζεστε για την διαχείριση του Arduino από τον υπολογιστή σας το παρέχει **το Arduino IDE**, την τελευταία έκδοση του οποίου μπορείτε να κατεβάσετε από το επίσημο site για καθένα από τα τρία δημοφιλέστερα λειτουργικά συστήματα. Το Arduino IDE είναι βασισμένο σε Java και συγκεκριμένα παρέχει:

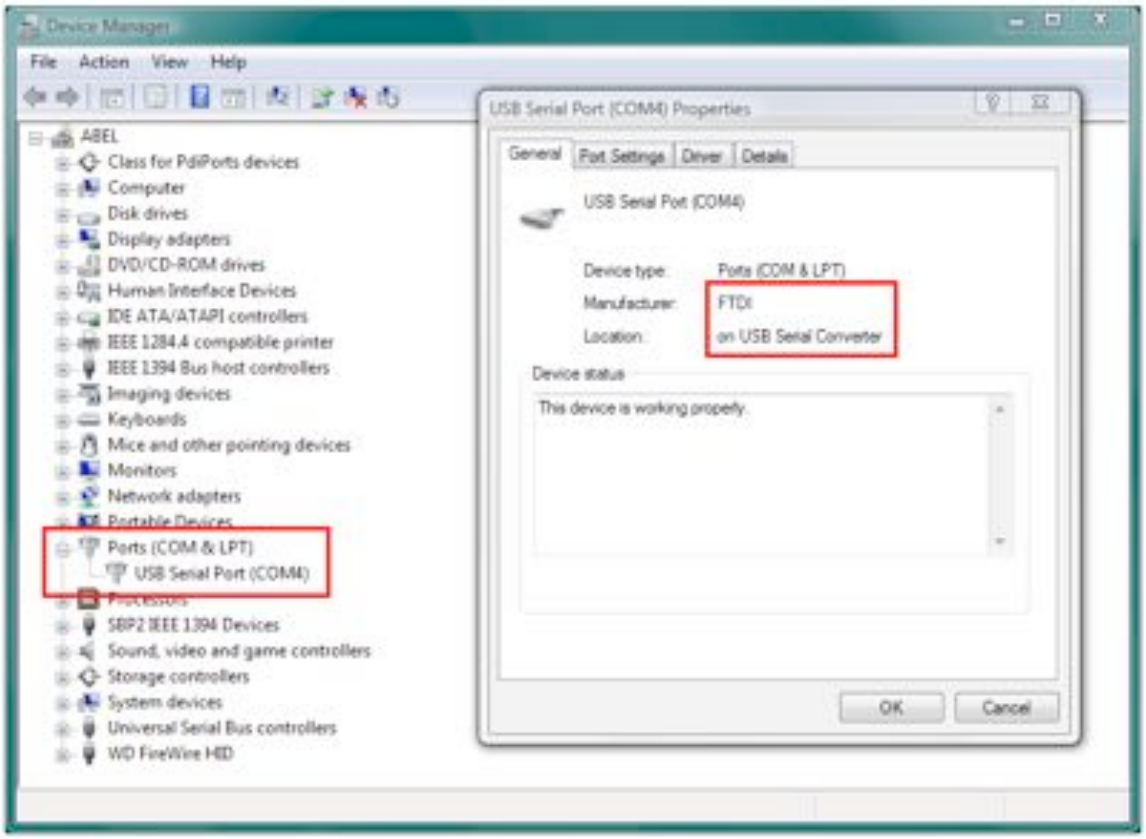
- ένα π**ρακτικό περιβάλλον** για την συγγραφή των προγραμμάτων σας (τα οποία **ονομάζονται sketch** στην ορολογία του Arduino) με συντακτική χρωματική σήμανση,
- αρκετά **έτοιμα παραδείγματα**,
- μερικές **έτοιμες βιβλιοθήκες** για προέκταση της γλώσσας και για να χειρίζεστε εύκολα μέσα από τον κώδικά σας τα εξαρτήματα που συνδέετε στο Arduino,
- **τον compiler** για την μεταγλώττιση των sketch σας,
- **ένα serial monitor** που παρακολουθεί τις επικοινωνίες της σειριακής (USB), αναλαμβάνει να στείλει αλφαριθμητικά της επιλογής σας στο Arduino μέσω αυτής και είναι ιδιαίτερα χρήσιμο για το debugging των sketch σας
- και την επιλογή να ανεβάσετε (**upload**) το μεταγλωττισμένο sketch στο Arduino.

Για τα δύο τελευταία χαρακτηριστικά βέβαια, το Arduino πρέπει να έχει συνδεθεί σε μια από τις θύρες USB του υπολογιστή και, λόγω του ελεγκτή Serial-over-USB, θα πρέπει να αναγνωριστεί από το λειτουργικό σας σύστημα ως εικονική σειριακή θύρα. Για την σύνδεση θα χρειαστείτε ένα καλώδιο USB από Type A σε Type B, όπως αυτό των εκτυπωτών. Για την αναγνώριση από το λειτουργικό θα χρειαστεί να εγκαταστήσετε τον οδηγό του FTDI chip (δηλαδή του ελεγκτή Serial-over-USB) ο οποίος υπάρχει στον

φάκελο drivers του Arduino IDE που κατεβάσατε. Την τελευταία έκδοση αυτού του οδηγού μπορείτε επίσης να κατεβάσετε για κάθε λειτουργικό σύστημα από το site της FTDI. Σημειώστε ότι στους τελευταίους πυρήνες του Linux υπάρχει εγγενής υποστήριξη του συγκεκριμένου ελεγκτή. Αν όλα έγιναν σωστά, το κεντρικό παράθυρο του Arduino IDE θα εμφανιστεί όταν το εκτελέσετε και στο μενού Tools –> Serial Port θα πρέπει να εμφανίζεται η εικονική σειριακή θύρα (συνήθως COM# για τα Windows, /dev/ttyusbserial## για το MacOS και /dev/ttyusb## για το Linux). Επιλέξτε αυτή την εικονική θύρα και στην συνέχεια επιλέξτε τον τύπο του Arduino σας (Arduino Duemilanove w/ ATmega328) από το μενού Tools –> Board. Το Arduino είναι πλέον έτοιμο να δεχτεί τα sketch σας. Αν εμφανίστηκε οποιοδήποτε πρόβλημα διαβάστε τις αναλυτικές οδηγίες εγκατάστασης για κάθε λειτουργικό σύστημα στη διεύθυνση http://arduino.cc/en/Guide/HomePage.



Το κεντρικό παράθυρο του Arduino IDE χωρίζεται σε δύο μέρη. Στο πάνω μέρος γράφετε τα sketch σας και στο κάτω μέρος εμφανίζονται πιθανά λάθη κατά την διαδικασία της μεταγλώττισης. Αμέσως μετά την πρώτη του εκτέλεση, δηλώστε την έκδοση του Arduino σας (όπως φαίνεται στην εικόνα) και την εικονική σειριακή που χρησιμοποιεί.



Αν ο οδηγός του ελέγχη Serial-over-USB είναι σωστά εγκατεστημένος στα Windows, το Arduino θα αναγνωρίζεται από τον Device Manager όπως στην εικόνα. Εκεί μπορείτε να δείτε και τον αριθμό της εικονικής σειριακής θύρας που του ανατέθηκε.
Εικόνα 9 από 13

ΚΛΕΙΣΙΜΟ

Η 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 μπορείτε να χρησιμοποιήσετε ουσιαστικά τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπους 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 (Vref) 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 Vref.
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: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW) • RISING (όταν από LOW γίνει HIGH) • FALLING (όταν από HIGH γίνει LOW) • 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.

Επιπλέον, στην γλώσσα του Arduino κάθε πρόγραμμα αποτελείται από δύο βασικές ρουτίνες ώστε να έχει την γενική δομή:

```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών...

void setup()
{
  // ...
}

void loop()
{
  // ...
}

// Υπόλοιπες συναρτήσεις...
```

Η βασική ρουτίνα setup() εκτελείται μια φορά μόνο κατά την εκκίνηση του προγράμματος ενώ η βασική ρουτίνα loop() περιέχει τον βασικό κορμό του προγράμματος και η εκτέλεσή της επαναλαμβάνεται συνέχεια σαν ένας βρόγχος while(true). Αν και πρόκειται μόνο για τις πιο βασικές λειτουργίες της γλώσσας του Arduino, με αυτές και με λίγες βασικές γνώσεις C θα μπορέσετε να δημιουργήσετε το sketch ακόμα και για κάποιο αρκετά περίπλοκο project. Για το πλήρες reference πάντως, επισκεφτείτε την σχετική σελίδα ενώ ακόμα περισσότερες πληροφορίες μπορείτε να βρείτε στο site της Wiring καθώς και στο εγχειρίδιο της βιβλιοθήκης AVR Libc.

Το παράδειγμα Hello World!

Έφτασε η στιγμή να δημιουργήσετε το πρώτο σας sketch, το οποίο -παραδοσιακά- πρέπει να εξάγει το μήνυμα «Hello World». Βέβαια -μέχρι να προσθέσετε εσείς μια- το Arduino δεν διαθέτει οθόνη ώστε να

εμφανίσει κάποιο μήνυμα. Η μόνη συσκευή εξόδου που είναι ενσωματωμένη στην πλακέτα του Arduino είναι το LED του pin 13. Έτσι, το Arduino σας θα χαιρετίσει την οικουμένη αναβοσβήνοντας απλά το LEDάκι του. Ανοίξτε το IDE του Arduino και -για να μην πληκτρολογείτε- επιλέξτε File –> Sketchbook –> Examples –> Digital –> Blink. Θα ανοίξει ένα sketch με τον παρακάτω κώδικα:

```
1  int ledPin = 13;
2  void setup()
3  {
4    pinMode(ledPin, OUTPUT);
5  }
6  void loop()
7  {
8    digitalWrite(ledPin, HIGH);
9    delay(1000);
10   digitalWrite(ledPin, LOW);
11   delay(1000);
12 }
```

Όπως κάθε "Hello World" πρόγραμμα, το sketch αυτό είναι αρκετά straightforward. Αρχικά, στην ρουτίνα setup() ρυθμίζεται το pin στο οποίο είναι συνδεδεμένο το LED ως pin εξόδου (γραμμή 4). Στην συνέχεια η κύρια ρουτίνα loop(), η εκτέλεση της οποίας επαναλαμβάνεται συνέχεια, ανάβει το LED (γραμμή 8) και στην συνέχεια το σβήνει (γραμμή 10). Δύο εντολές delay ρυθμίζουν τον χρόνο που το LED θα μένει αναμμένο ή σβηστό στις γραμμές 9 και 11 (1000ms δηλαδή 1 δευτερόλεπτο). Για να δείτε το πρόγραμμα στην πράξη, εφόσον έχετε ήδη συνδέσει το Arduino με τον υπολογιστή επιλέξτε File –> Upload to I/O Board (εναλλακτικά πατήστε Ctrl-U ή κάντε κλικ στο ανάλογο εικονίδιο της toolbar). Με αυτή την ενέργεια, το sketch θα μεταγλωττιστεί και θα σταλεί αυτόματα στο Arduino, γεγονός που μπορείτε να επαληθεύσετε από την δραστηριότητα των TX και RX LED πάνω στην πλακέτα του Arduino. Τα προγράμματα που «ανεβάζετε» στο Arduino εκτελούνται αυτόματα από τον bootloader αμέσως μετά την λήψη τους και έτσι, χωρίς καθυστέρηση, θα πρέπει να δείτε το LED με τη σήμανση 13 να ανάβει και να σβήνει συνεχόμενα με περίοδο 2 δευτερολέπτων, δηλαδή όπως ακριβώς ορίζει το sketch. Αν επιμένετε ότι ένα LED που αναβοσβήνει δεν αποτελεί πρέπον χαιρετισμό και θέλετε σώνει και καλά να δείτε το "Hello World" γραμμένο, υπάρχει μια λύση. Μπορείτε να το στείλετε μέσω της σειριακής (USB) στον υπολογιστή και να το δείτε στην οθόνη σας. Και σαν bonus, το Arduino θα στέλνει και την κατάσταση του LED στον υπολογιστή. Προσθέστε απλά τις γραμμές:

```
Serial.begin(9600);
Serial.println("Hello World! - Are you happy now?");
```

αμέσως μετά την γραμμή με την εντολή pinMode και πριν κλείσει το άγκιστρο της ρουτίνας setup().

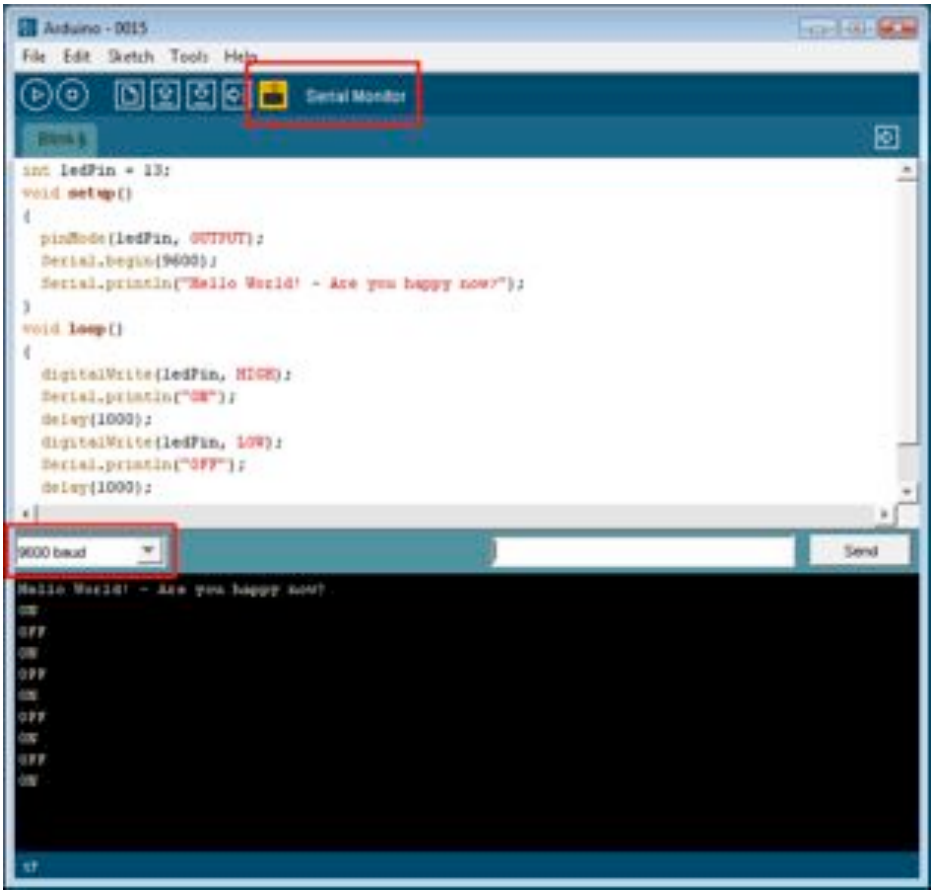
Επίσης, προσθέστε την γραμμή:

```
Serial.println("ON”);
```

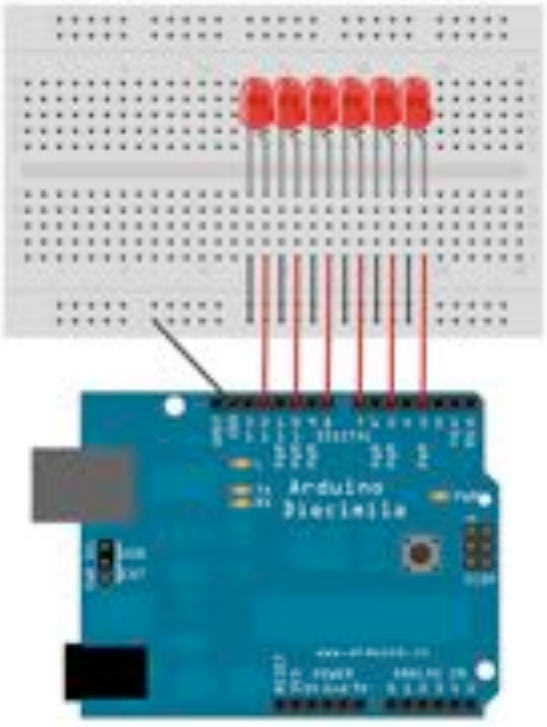
και την γραμμή:

```
Serial.println("OFF”);
```

αμέσως μετά την πρώτη και την δεύτερη digitalWrite αντίστοιχα.



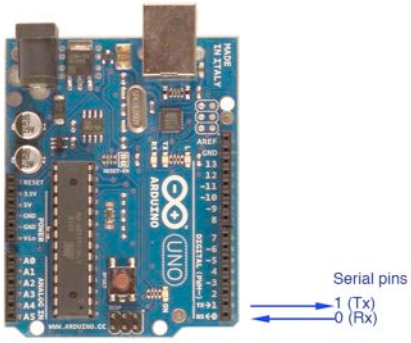
Αφού κάνετε τις αλλαγές επιλέξτε όπως και πριν το Upload to I/O Board από το IDE για να γίνει ξανά μεταγλώττιση και να ανέβει το νέο binary στο Arduino. Αμέσως μετά, κάντε κλικ στο τελευταίο κουμπί της toolbar με επεξήγηση Serial Monitor για να μετατρέψετε το κάτω τμήμα του παραθύρου του IDE σε σειριακή κονσόλα και σύντομα θα δείτε το Arduino να σας στέλνει τα μηνύματά του.



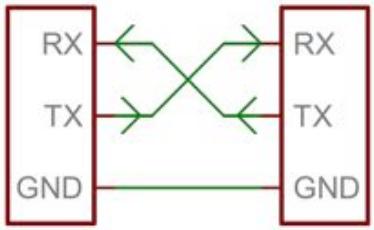
Μπορείτε να πειραματιστείτε με το sketch, να φτιάξετε ωραία pattern με τα οποία αναβοσβήνει το LED, να το βάλετε να στέλνει διαφορετικές πληροφορίες στην σειριακή κ.λπ. αλλά ακόμα και αν μάθετε στο Arduino να σας μιλάει με κώδικα Morse, δεν παύει να είναι ένα LED που αναβοσβήνει και ενίοτε φλυαρεί στην σειριακή – δεν έχει κάτι σημαντικό να σας πει. Αυτό φυσικά συμβαίνει επειδή δεν έχετε συνδέσει ακόμα περιφερειακά στο Arduino σας και έτσι είναι σαν να έχετε ένα υπολογιστή χωρίς οθόνη, ποντίκι και πληκτρολόγιο.

Σειριακή επικοινωνία με περιφερειακές συσκευές

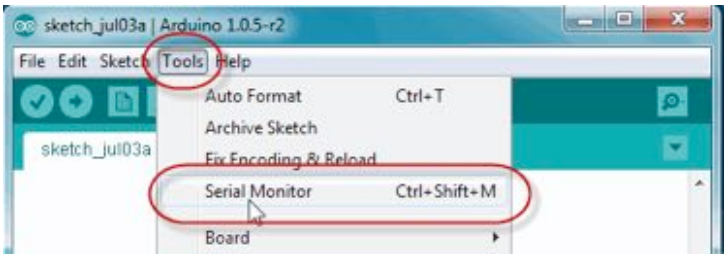
Συχνά στις κατασκευές μας θα χρειαστεί να στείλουμε ή να λάβουμε δεδομένα στο Arduino uno. Αυτή η επικοινωνία γίνεται μέσω της σειριακής θύρας του Arduino και πιο συγκεκριμένα μέσω των pins Rx και Tx που υπάρχουν πάνω σε αυτό.



Τα δεδομένα λαμβάνονται απ' το Arduino uno μέσω του pin Rx (Receive) και στέλνονται απ' αυτό μέσω του pin Tx (Transmit). Για να γίνει σωστά η επικοινωνία θα πρέπει το Rx και Tx του Arduino να συνδεθούν σταυρωτά με τα αντίστοιχα Rx και Tx της μονάδας που θέλουμε να γίνει η σειριακή σύνδεση (πχ GPS, Bluetooth module).



Η επικοινωνία μέσω του Η/Υ γίνεται μέσα απ' την σειριακή οθόνη (serial monitor) του Arduino IDE. Δεν χρειάζεται να κάνετε τίποτα με τα Rx και Tx, απλά συνδέστε το καλώδιο usb.



Για να επιτευχθεί η σωστή επικοινωνία θα πρέπει και οι δύο πλευρές, Η/Υ και Arduino, να έχουν τον ίδιο ρυθμό μετάδοσης δεδομένων (baud rate). Για εμάς πάντα ο ρυθμός μετάδοσης θα είναι τα 9600kbps.

Σε προγραμματιστικό επίπεδο, ώστε να μπορέσουμε να πούμε στο Arduino πως θα επικοινωνήσουμε με αυτό, η εντολή που χρησιμοποιούμε είναι η **Serial.begin(9600);** Η τιμή 9600 αναφέρεται στον ρυθμό μετάδοσης που είπαμε παραπάνω. Την εντολή αυτή την βάζουμε πάντα στο κομμάτι της void setup(). Για να μας εκτυπώσει το Arduino ένα μήνυμα στην σειριακή οθόνη, θα πρέπει μέσα στον κώδικα να κάνουμε χρήση της εντολής **Serial.print("Hello World");**. Αν θέλουμε το επόμενο μήνυμα να ξεκινάει στην επόμενη γραμμή τότε κάνουμε χρήση της print**ln**.

```
void setup(){ Serial.begin(9600);
              Serial.println("Hello World");}
```

Έστω ότι θέλουμε το Arduino να μας στέλνει εναλλάξ, ανά 2 δευτερόλεπτα, το μήνυμα "Blacκ" και "White" (δεν υποστηρίζονται ελληνικοί χαρακτήρες). Ο κώδικας που θα φορτώσουμε στο arduino uno είναι ο εξής:

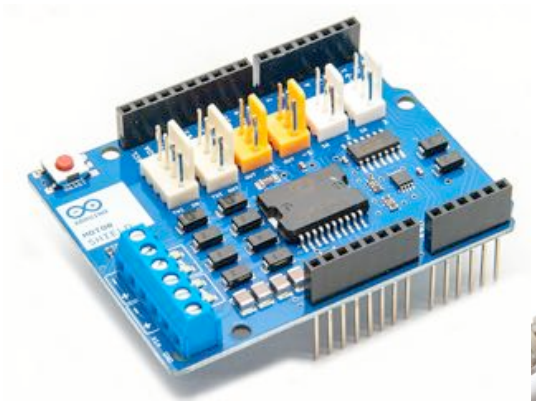
```
void setup(){ Serial.begin(9600); }
```

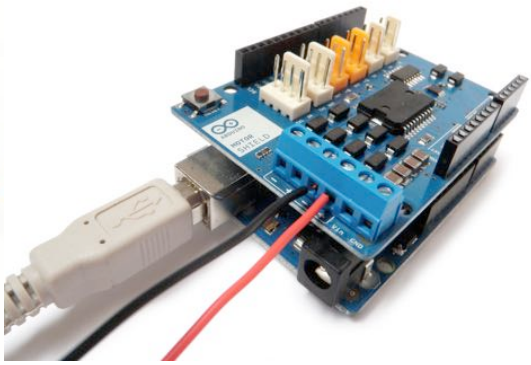
```
void loop(){
       Serial.println("Black");
       delay(2000);
       Serial.println("White");
       delay(2000);
           }
```

Πλακέτες επέκτασης (Arduino-Shield)

Τα shield είναι ολοκληρωμένες πλακέτες που είναι σχεδιασμένες ώστε να κουμπώνουν πάνω στο Arduino προεκτείνοντας την λειτουργικότητά του. Είναι η hardware αντίστοιχη έννοια των plugin, addon και extension που υπάρχουν στο software. Μερικά από τα πιο δημοφιλή shield που κυκλοφορούν στο εμπόριο για το Arduino είναι:

- **Ethernet shield**: Δίνει στο Arduino την δυνατότητα να δικτυωθεί σε ένα LAN ή στο internet μέσω ενός τυπικού καλωδίου Ethernet.
- **WiFi shield**: Όμοιο με το Ethernet shield, χωρίς φυσικά το καλώδιο.
- Διάφορα **shield οθόνης**: Προσθέτουν οθόνη στο Arduino. Κυκλοφορούν από απλές οθόνες τύπου calculator μέχρι OLED touchscreen υψηλής ανάλυσης τύπου iPhone.
- **Wave shield**: Δίνει στο Arduino την δυνατότητα να παίζει ήχους/μουσική από κάρτες SD.
- **GPS shield**: Προσθέτει GPS δυνατότητες στο Arduino (εντοπισμό στίγματος).
- Διάφορα **Motor Shields**: Σας επιτρέπουν να οδηγήσετε εύκολα μοτέρ διάφορων τύπων (απλά DC, servo, stepper κ.λπ.) από το Arduino.
- **ProtoShield**: Μια προσχεδιασμένη πλακέτα πρωτοτυποποίησης, συμβατή στις διαστάσεις του Arduino και χωρίς εξαρτήματα για να φτιάξετε το δικό σας shield.





Τα shield είναι σχεδιασμένα ώστε αφού κουμπωθούν πάνω στο Arduino να προωθούν τις υποδοχές του, ώστε να μπορείτε να συνδέσετε επιπλέον τα δικά σας εξαρτήματα ή να κουμπώσετε και επόμενο shield. Φυσικά, το κάθε shield χρησιμοποιεί ορισμένους από τους πόρους συνδεσιμότητας του Arduino και έτσι

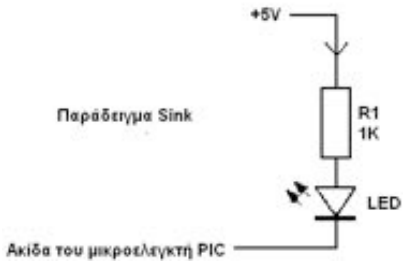
δεν μπορείτε να συνδέσετε απεριόριστα shield. Μάλιστα κάποια shield μπορεί να μην είναι συμβατά μεταξύ τους γιατί χρησιμοποιούν τα ίδια pin του Arduino για επικοινωνία με αυτό. Επίσης, επειδή κάποια shield δεν προωθούν τις συνδέσεις του Arduino (όπως π.χ. οι οθόνες οι οποίες δεν έχουν νόημα αν τις καλύψετε από πάνω με ένα επόμενο shield), υπάρχουν ειδικά extender shield που κουμπώνουν στο Arduino και δίνουν την δυνατότητα σε δύο άλλα shield να κουμπώσουν πάνω τους, λειτουργώντας σαν πολύπριζα. Όπως και για το ίδιο το Arduino, το βασικό πλεονέκτημα των shield δεν είναι τόσο το προφανές πλεονέκτημα του έτοιμου hardware όσο ότι συνοδεύονται συνήθως από έτοιμες βιβλιοθήκες που σας επιτρέπουν να προγραμματίζετε τα sketch σας σε high level. Έτσι, λόγου χάρη, δεν χρειάζεται να διαβάζετε datasheet για να συνδέσετε και να λειτουργήσετε ένα GPS module πάνω στο Arduino. Απλά συνδέετε το shield, εγκαθιστάτε τη βιβλιοθήκη που το συνοδεύει και χρησιμοποιείτε μια έτοιμη συνάρτηση -του στυλ getLocation- για να πάρετε το γεωγραφικό στίγμα και να το επεξεργαστείτε περαιτέρω στο sketch σας. Τα shield σας λύνουν τα χέρια όταν θέλετε να δημιουργήσετε εύκολα ένα πραγματικά πρακτικό project. Αυτός είναι και ο λόγος που δεν συνιστάται η αγορά κάποιας έκδοσης του Arduino που δεν είναι 100% συμβατή με τα shield.

Ακίδες εξόδου μικροελεγκτή

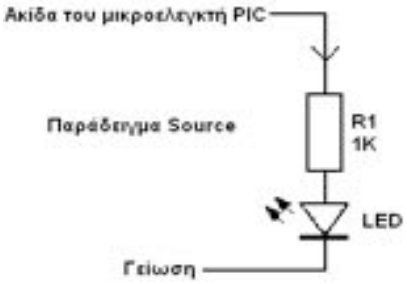
Το ρεύμα που μπορεί να περάσει μέσα από τους περισσότερους μικροελεγκτές όταν γίνεται είσοδος ή έξοδος δεδομένων από τους ακροδέκτες I/O ανέρχεται από 25mA μέχρι 40mA. Στα εγχειρίδια των κατασκευαστών αναφέρονται:

- Υψηλό επίπεδο (high=λογικό 1), που είναι μια τάση περί τα 5V
- Χαμηλό επίπεδο (low=λογικό 0), που είναι μια τάση περί τα 0V
- Το ρεύμα που μπορεί να περάσει[1] δια μέσου του ακροδέκτη εξόδου του μικροελεγκτή όταν αυτός τοποθετείται σε χαμηλή στάθμη με το λογικό μηδέν (Σχήμα 21.3)
- Το ρεύμα που μπορεί να περάσει[2] δια μέσου του ακροδέκτη εξόδου του μικροελεγκτή όταν αυτός τοποθετείται υψηλή στάθμη με το λογικό ένα (Σχήμα 21.4)

[1] Μπορεί να πει κανείς το ρεύμα που μπορεί να τραβήξει το τσιπ. Αναφέρεται στη βιβλιογραφία με τον όρο **'sink'**.
[2] Το ρεύμα που μπορεί να δώσει το τσιπ. Αναφέρεται στη βιβλιογραφία με τον όρο **'source'**.



Σχήμα 21. 1 Τρόπος σύνδεσης διόδου LED με μικροελεγκτή. Το LED ανάβει σε περίπτωση που σταλεί στην ακίδα του μικροελεγκτή το λογικό μηδέν



Σχήμα 21. 2 Τρόπος σύνδεσης διόδου LED με μικροελεγκτή. Το LED ανάβει σε περίπτωση που σταλεί στην ακίδα του μικροελεγκτή το λογικό ένα. Το ρεύμα πηγάζει από την ακίδα του μικροελεγκτή

Οι αντιστάσεις 1K χρησιμοποιούνται για τον περιορισμό του ρεύματος προκειμένου να μην "καεί"» το LED και να μην καταστραφεί ο μικροελεγκτής σύμφωνα με τον τύπο:

$$\mathrm{I} = \frac{V_{cc} - V_{diod}}{R} = \frac{5 - 0.7}{1000} = 4,3mA$$

Σε περίπτωση μη χρησιμοποίησης της αντίστασης R1 τόσο η ακίδα του μικροελεγκτή όσο και η δίοδος LED θα καούν. Το ρεύμα 4,3mA αρκεί για ορισμένα ειδικά LED χαμηλής κατανάλωσης. Για τα κοινά LED το ρεύμα θα πρέπει να είναι της τάξεως των 15mA και ως εκ τούτου η αντίσταση θα πρέπει να είναι περί τα 300Ω.

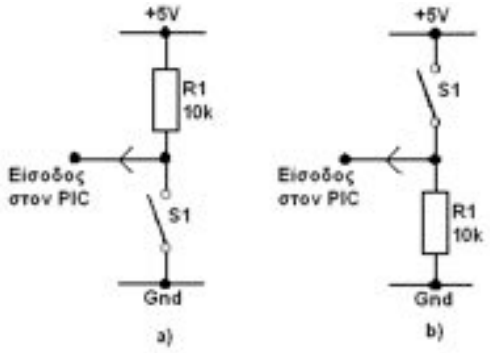
Όταν πρόκειται να οδηγηθεί μικρό φορτίο όπως τα LED τα πράγματα είναι απλά, αφού τα 25mA του μικροελεγκτή υπεραρκούν για την οδήγηση. Όταν όμως απαιτούνται μεγαλύτερα ρεύματα, τότε θα πρέπει να χρησιμοποιηθούν ενισχυτικές διατάξεις ρεύματος. Η πιο απλή ενίσχυση ρεύματος επιτυγχάνεται με τρανζίστορ.

Ακίδες εισόδου του μικροελεγκτή

Υπάρχουν οι δύο βασικές εναλλακτικές λύσεις για είσοδο δεδομένων που φαίνονται στο Σχήμα 21.7. Με την πρώτη γίνεται είσοδος της χαμηλής στάθμης (LOW, λογικού μηδέν), ενώ με τη δεύτερη γίνεται είσοδος της υψηλής στάθμης (HIGH) στην ακίδα εισόδου. Οι αντιστάσεις R1 ονομάζονται **pullups**. Οι αντιστάσεις αυτές δεν χρειάζονται απαραιτήτως τις περισσότερες φορές, επειδή υπάρχουν εσωτερικά στις περισσότερες εισόδους των μικροελεγκτών.

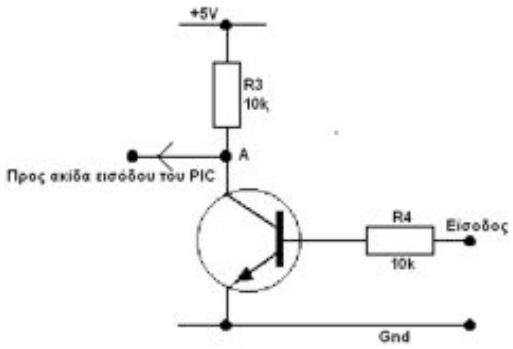
Στο Σχήμα 21.7 α) η ακίδα του PIC-μικροελεγκτή τραβιέται στην χαμηλή (LOW, είσοδος λογικό μηδέν) μόλις πατηθεί ο διακόπτης S1 και η R1 ονομάζεται **pullup αντίσταση**, ενώ στο b) τραβιέται η ακίδα του

PIC-μικροελεγκτή υψηλά (είσοδος λογικό ένα HIGH) μόλις πατηθεί ο διακόπτης S1 ενώ R1 ονομάζεται τώρα **pulldown αντίσταση**.



Σχήμα 21. 3 Οι δύο τρόποι εισαγωγής του λογικού 'μηδέν' και του λογικού 'ένα' με το κλείσιμο του διακόπτη

Δεν είναι απαραίτητο η είσοδος να προέρχεται από το κλείσιμο διακόπτη, αλλά μπορεί να είναι η έξοδος ενός άλλου μικροελεγκτή, ή άλλου ολοκληρωμένου ή ενός τρανζίστορ (Σχήμα 21.32). Όταν η είσοδος είναι +5V, γίνεται η ακίδα του μικροελεγκτή μηδέν (LOW) επειδή τίθεται σε αγωγιμότητα το τρανζίστορ, ενώ όταν η είσοδος είναι σε χαμηλή στάθμη (LOW) τίθεται το τρανζίστορ σε αποκοπή και η ακίδα του PIC-μικροελεγκτή βρίσκεται σε υψηλή στάθμη (HIGH).



Σχήμα 21. 32 Είσοδος στο μικροελεγκτή από τρανζίστορ. Όταν είναι η είσοδος του τρανζίστορ υψηλά το δυναμικό του σημείου Α είναι σχεδόν 0V, ενώ όταν είναι η είσοδος χαμηλά, το δυναμικό του σημείου Α είναι 5V.

Βασικά Παραδείγματα με τον μικροελεγκτή Arduino.

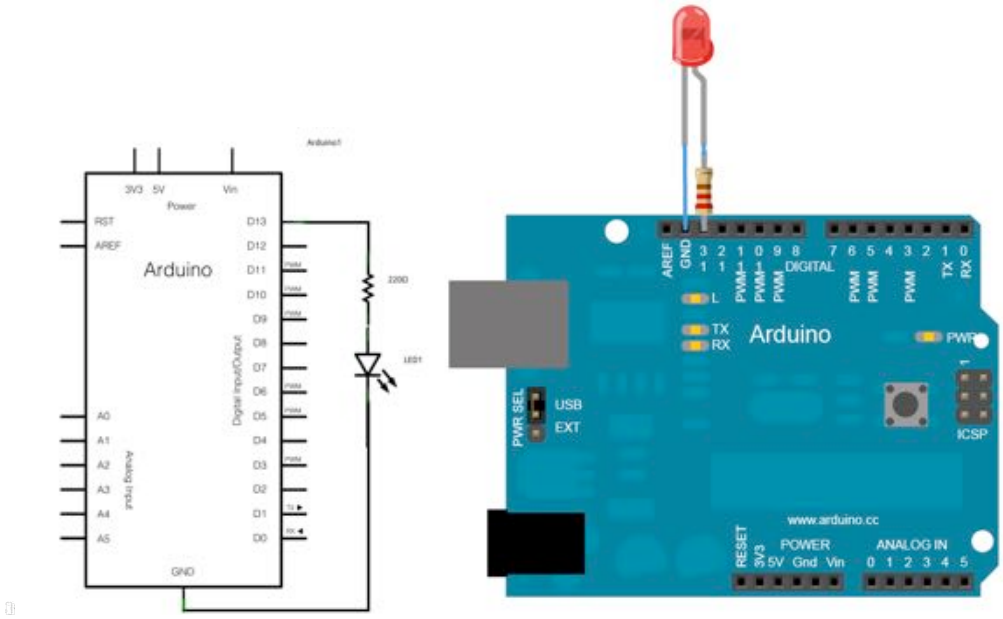
Παράδειγμα 1: Συνεχές αναβόσβημα ενός Led με περίοδο 1sec.

Το Arduino αποτελείται από δεκατρία ψηφιακά pin, τα οποία μπορούμε να τα χρησιμοποιήσουμε το κάθε ένα ξεχωριστά, είτε για είσοδο είτε για έξοδο. Μπορούμε να τα προγραμματίσουμε να συμπεριφέρονται όπως εμείς θέλουμε, αρκεί να κάνουμε τις σωστές δηλώσεις στο κώδικα που θα

φορτώσουμε στη πλακέτα. Η έξοδος του κάθε pin μπορεί να προγραμματιστεί να δίνει τιμές HIGH ή LOW. Λέγοντας HIGH ενώνουμε το δυαδικό '1' και έχουμε τάση εξόδου 5V DC, ενώ το LOW είναι το δυαδικό '0' και έχει τάση εξόδου 0V DC (ground).

Παρακάτω θα δούμε ένα παράδειγμα για να μπορέσουμε να κατανοήσουμε τον προγραμματισμό των ψηφιακών pin του Arduino. Θα ενώσουμε ένα led στη board και θα το προγραμματίσουμε να ανάβει και να σβήνει σε χρονικά διαστήματα ενός δευτερολέπτου. Η εφαρμογή παρουσιάζει το απλούστερο πράγμα που μπορείτε να κάνετε με ένα Arduino για να δείτε μια ψηφιακή θύρα του μικροελεγκτή να αναβοσβήνει ένα LED.

Σχηματικό Κύκλωμα: Για να υλοποιήσετε το κύκλωμα, συνδέστε μια αντίσταση 220Ω με το pin 13. Κατόπιν συνδέστε τον μακρύτερο ακροδέκτη (άνοδο) με την αντίσταση. Συνδέστε το κοντό ακροδέκτη (κάθοδο) με τη γείωση. Κατόπιν συνδέστε το Arduino με τον υπολογιστή σας, ξεκινήστε το πρόγραμμα Arduino, και πληκτρολογείστε τον παρακάτω κώδικα. Τα περισσότερα Arduino ήδη έχουν ένα LED στο pin 13. Εάν τρέχετε αυτό το παράδειγμα χωρίς εξωτερικό LED και αντισταση, πρέπει να δείτε το ενσωματωμένο LED να αναβοσβήνει.



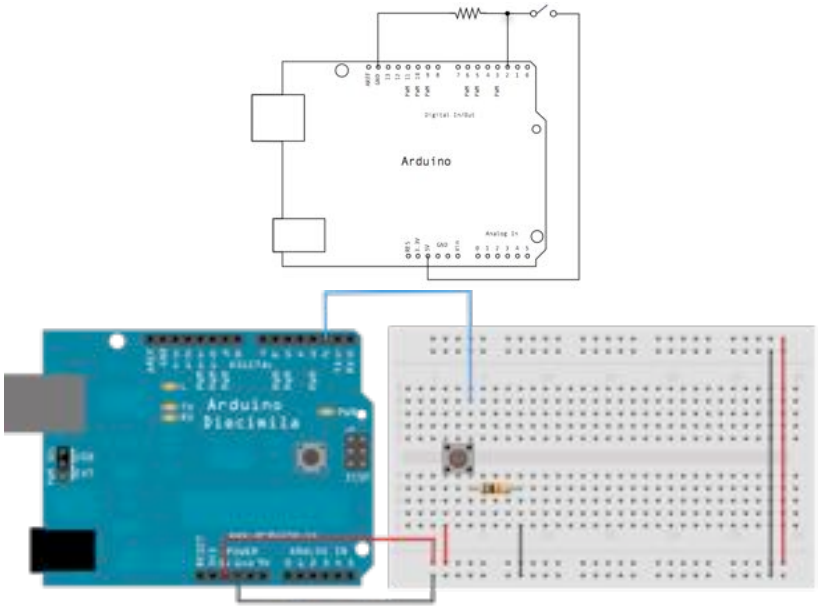
Για την δημιουργία της εικόνας χρησιμοποιήθηκε το προγραμμα Fritzing. Για περισσότερα παραδείγματα κυκλωμάτων, δείτε: http://fritzing.org

Παρακάτω δίνεται ένας πρώτυπος κώδικας που επιτυγχάνει το αναβόσβημα ενός Led με περίοδο 1sec (Οι γραμμές με πορτοκαλί είναι κώδικας-εντολές όλα τα άλλα είναι σχόλια)

```
  // Στους περισσότερους Arduino στο pin 13 είναι, πάνω στην PCB πλακέτα, συνδεδεμένο
 // ένα LED. Ας δώσουμε λοιπόν στο pin 13 ένα όνομα:
int led = 13; // Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα led και
               // της δίνουμε την τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω
              // στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led.

       // Η ρουτίνα void setup εκτελείται μονάχα μία φορά και χρησιμοποιείται
      // για να  αχρικοποιήσουμε τις παραμέτρους του Arduino δηλαδή, αποτελεί
     // τη συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή)

void setup() {
pinMode(led, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή led, ότι το pin
                      //  13 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
                }
    //Η ρουτίνα void loop εκτελείται συνεχώς και για όσο χρόνο υπάρχει τροφοδοσία στον
   //Arduino.
void loop() {
   digitalWrite(led,HIGH);//Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led
   delay(1000);             // Συνάρτηση καθυστέρησης χρόνου 1000ms
   digitalWrite(led, LOW);//Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το led
   delay(1000);              // Συνάρτηση καθυστέρησης χρόνου 1000ms
            }
```

Παράδειγμα 2: Άναμμα και σβήσιμο Led ελεγχόμενο από εξωτερικό διακόπτη (button) με την διαδικασία *polling* δηλαδή με αλλεπάλληλους ελέγχους από το κυρίως πρόγραμμα.

Παρακάτω δίνεται το σχηματικό κύκλωμα για την σύνδεση του διακόπτη καθώς και ένας πρότυπος κώδικας για τον προγραμματισμό του Arduino. Ο κώδικας αναλύεται διεξοδικά προκειμένου να καλυφθούν οποιεσδήποτε απορίες.



Σχηματικό διάγραμμα για την χρήση του διακόπτη επαναφοράς

```
// Οι μεταβλητές που ορίζονται ως connstants δεν αλλάζουν και μπορούν να χρησιμοποιηθούν
για τον ορισμό των αριθμών που αντιστοιχούν στα pin του Arduino:

 const int buttonPin = 2; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με
                           //όνομα buttonPin και της δίνουμε την τιμή 2 με σκοπό
                          //να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να
                         //δηλώσουμε που θα συνδέσουμε το button.

 const int ledPin = 13; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με
                        // όνομα ledPin και της δίνουμε την τιμή 13 με σκοπό να
                       // την χρησιμοποιήσουμε παρακάτω στον κώδικα και να
                      // δηλώσουμε που θα συνδέσουμε το led.

// Παρακάτω έχω μεταβλητές οι οποίες αλλάζουν τιμή κατά την εκτέλεση του προγράμματος

int buttonState = 0; //Ορίζουμε μία μεταβλητή τύπου ? Integer (Ακεραίου) με όνομα
                     // buttonState και της δίνουμε αρχικά την τιμή 0 με σκοπό να
                    // την χρησιμοποιήσουμε παρακάτω στον κώδικα για να
                   // αποθηκεύουμε την κατάσταση λειτουργίας του button (on-off)

// Ακολουθεί η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή

 void setup() {
      pinMode(ledPin, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή
                               // ledPin, ότι το pin 13 θα λειτουργεί σαν
                              // ψηφιακή έξοδος 0-5V

     pinMode(buttonPin, INPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή
                                // buttonPin, ότι το pin 2 θα λειτουργεί σαν
                               // ψηφιακή είσοδος 0-5V
               }
// Ακολουθεί η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
 void loop(){
      buttonState = digitalRead(buttonPin); // Διαβάζουμε την ψηφιακή τιμή
                                             //που έχει το Pin 2 (εκεί που
                                            //συνδέεται το button) και την
                                           //αποθηκεύουμε στην μεταβλητή
                                          //buttonState
//Στην συνέχεια ελέγχουμε αν το button είναι πατημένο, αν ναι τότε: άναψε το  LED
        if (buttonState == HIGH) {
                     digitalWrite(ledPin, HIGH); // Βγάλε λογικό 1 (5V)
                                                  //στο pin 13, δηλαδή
                                                 // άναψε το led
                                    }
            else { // αλλιώς: σβήσε το LED
              digitalWrite(ledPin, LOW); //Βγάλε λογικό 0 (0V) στο pin 13
                   }
             }
```

Οι Βιβλιοθήκες της υπολογιστικής πλατφόρμας Arduino

Όλες οι βιβλιοθήκες της **υπολογιστικής πλατφόρμας Arduino** βρίσκονται στο διαδίκτυο και μπορεί να τις κατεβάσει ο καθένας ελεύθερα και σκοπός τους είναι η ευκολία στη χρήση διαφόρων παρελκόμενων (οθόνη lcd, κινητήρες rc-servo κ.α.) μαζί τους. Το περιβάλλον Arduino (IDE) μπορεί να επεκταθεί με τη χρήση βιβλιοθηκών, ακριβώς όπως οι περισσότερες πλατφόρμες προγραμματισμού, αφού οι βιβλιοθήκες παρέχουν επιπλέον λειτουργίες για τη χρήση τους απλοποιώντας σημαντικά την

διαδικασία του προγραμματισμού. Μια σειρά από βιβλιοθήκες υπάρχουν προεγκατεστημένες, και είναι πολύ εύκολο να τις κατεβάσετε και να τις επεξεργαστείτε ή ακόμα και να δημιουργήσετε τη δική σας. Στην συνέχεια θα παρουσιάσουμε συνοπτικά κάποιες σημαντικές βιβλιοθήκες.

Βιβλιοθήκη των DC κινητήρων τύπου «RC-SERVO»

Όπως έχει προαναφερθεί και στις προηγούμενες βιβλιοθήκες έτσι και η βιβλιοθήκη των κινητήρων τύπου **«RC-Servo»** υπάρχει προ-εγκατεστημένη στο περιβάλλον προγραμματισμού του Arduino (IDE). Οι περισσότεροι κινητήρες τύπου «servo» κινούνται μεταξύ 0 και 180 μοιρών με καλή ακρίβεια και σε διαφορετικές ταχύτητες. Η βιβλιοθήκη των κινητήρων τύπου «servo» υποστηρίζει έως 12 κινητήρες για τις περισσότερες πλακέτες Arduino ενώ για τον Mega έως 48. **Με την ενεργοποίηση της βιβλιοθήκης στην υπολογιστική πλατφόρμα Arduino, εκτός του Arduino Mega, απενεργοποιείται η λειτουργία «PWM» στις θύρες 9 και 10 ακόμα και αν δεν υπάρχει κινητήρας συνδεδεμένος**, ενώ στον Mega μέχρι και 12 κινητήρες μπορούν να δουλέψουν χωρίς αυτή την δυσλειτουργία, εάν όμως χρησιμοποιηθούν 12 έως 23 κινητήρες απενεργοποιείται η λειτουργία «PWM» στις θύρες 11 και 12. Για την σωστή χρήση της βιβλιοθήκης ακολουθούμε την εξής διαδικασία:

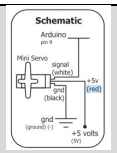
- Δηλώνουμε την βιβλιοθήκη στην αρχή του κώδικα ως **#include <Servo.h>**
- Δηλώνουμε το όνομα του κινητήρα σαν μία μεταβλητή τύπου **Servo π.χ. Servo myservo_name;**
- Καθορίζουμε μέσα στην συνάρτηση **setup()** τη θύρα που είναι συνδεδεμένος ο κινητήρας π.χ. **myservo.attach(9);**
- Καθορίζουμε μέσα στην συνάρτηση **loop()** την επιθυμητή τιμή για την γωνία του κινητήρα π.χ. **myservo.write(180);**

Παράδειγμα χρήσης της βιβλιοθήκης «RC-SERVO»

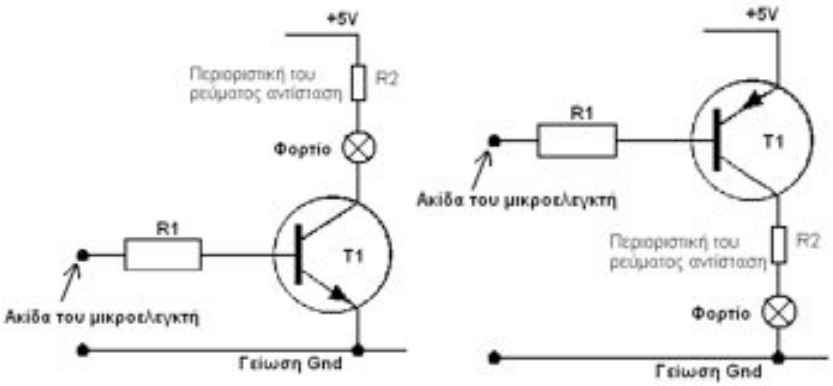
Παρακάτω παραθέτουμε ένα απλό παράδειγμα χρήσης της βιβλιοθήκης rc-servo, ώστε να γίνει αντιληπτός ο τρόπος με τον οποίο δηλώνουμε της θύρες του Arduino αλλά και οι συναρτήσεις της βιβλιοθήκης. Αρχικά ορίζεται το pin σύνδεσης του κινητήρα rc-servo (Pin 9) και στην συνέχεια με την βοήθεια ενός επαναληπτικού βρόγχου ο κινητήρας αλλάζει την γωνία περιστροφής του από 0^{o} σε 180^{o} με βήμα μίας μοίρας και αντίστροφα.

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

int pos = 0;    // variable to store the servo position
```

```
void setup()
          {
        myservo.attach(9);// attaches the servo on pin 9 to the
servo object
          }


void loop()
{
  for(pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
     {                              // in steps of 1 degree
      myservo.write(pos); // tell servo to go to position in variable 'pos'
      delay(15);          // waits 15ms for the servo to reach the position
     }
  for(pos = 180; pos>=1; pos-=1) // goes from 180 degrees to 0 degrees
     {
      myservo.write(pos); // tell servo to go to position in variable 'pos'
      delay(15);          // waits 15ms for the servo to reach the position
     }
}
```



Δύο απλοί τρόποι οδήγησης φορτίου με μικροελεγκτή



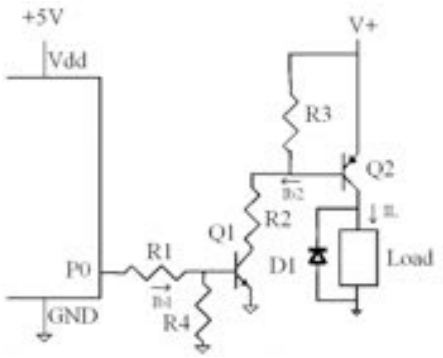
α) *β)*

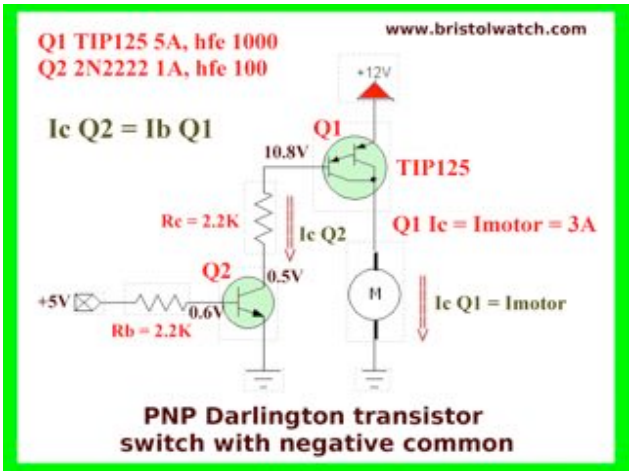
Σχήμα 21. 4 Δύο τρόποι α) και β) οδήγησης φορτίου με μικροελεγκτή.

Σύμφωνα με το Σχήμα 21.4 α) η πρώτη περίπτωση είναι η ευκολότερη, αφού όταν η ακίδα του μικροελεγκτή (η βάση του τρανζίστορ) γίνεται λογικό 1 περνάει ρεύμα μέσα από την R1, μεταβαίνει σε αγωγιμότητα το τρανζίστορ **συνδέοντας το φορτίο με τη γείωση**. **Η τάση που εφαρμόζεται στο φορτίο δεν είναι απαραίτητο να είναι 5V αλλά μπορεί π.χ. να ανέρχεται σε 24V ή μεγαλύτερη**. Αυτό που θα πρέπει να προσέξει κανείς είναι **η γείωση της τροφοδοσίας του τρανζίστορ να είναι κοινή με τη γείωση της τροφοδοσίας του μικροελεγκτή**.

Σύμφωνα με το Σχήμα 21.4 β), όταν η ακίδα του μικροελεγκτή (η βάση του τρανζίστορ pnp) γίνεται λογικό 1 δεν περνάει ρεύμα μέσα από την R1, μεταβαίνει το τρανζίστορ σε αποκοπή, ενώ όταν η ακίδα του μικροελεγκτή γίνεται λογικό 0 περνάει ρεύμα μέσα από την R1 φέρνοντας το τρανζίστορ στον κόρο **συνδέοντας το φορτίο με την τροφοδοσία**. Σε αντίθεση με την προηγούμενη περίπτωση του npn τρανζίστορ τώρα **η τροφοδοσία του φορτίου πρέπει να είναι η ίδια με την τροφοδοσία του μικροελεγκτή**, επειδή για να τεθεί το τρανζίστορ σε αποκοπή θα πρέπει η ακίδα **να έχει υψηλό λογικό επίπεδο**. Δηλαδή, όταν θέλουμε το φορτίο απενεργοποιημένο, η βάση του τρανζίστορ θα πρέπει να είναι **σε τάση κοντά ή υψηλότερη από αυτή του εκπομπού**. Όταν όμως το φορτίο πρέπει να οδηγηθεί από τον μικροελεγκτή με υψηλότερη τάση π.χ. Vcc=12 volt τότε δεν μπορούμε να χρησιμοποιήσουμε το παραπάνω κύκλωμα διότι πολύ πιθανόν ο μικροελεγκτής να υποστεί βλάβη αφού στην ακίδα εισόδου του θα εφαρμοστεί τάση $Vcc-V_{be}$= Vcc- 0,7 volt η οποία είναι μεγαλύτερη από τα 5 volt που αυτός μπορεί να διαχειριστεί. Λύση αποτελούν τα παρακάτω δύο κυκλώματα ενώ περισσότερες λεπτομέρειες σχετικά με αυτά μπορούν να βρεθούν στους παρακάτω συνδέσμους:

http://www.w9xt.com/page_microdesign_pt12_hv_pnp_switching.html
http://www.bristolwatch.com/ele/transistor_drivers.htm

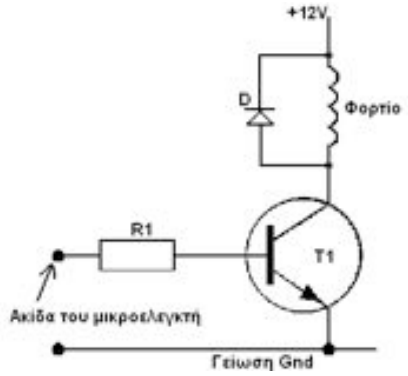




Όταν το φορτίο είναι επαγωγικό π.χ. ένα ρελαί, ή ένας κινητήρας κατά την αλλαγή κατάστασης της ακίδας του μικροελεγκτή δημιουργείται ανάστροφη (υπέρ)τάση[3], η οποία θα καταστρέψει το τρανζίστορ και ίσως και το μικροελεγκτή. Για το λόγο αυτό, θα πρέπει να συνδεθεί παράλληλα με το φορτίο μια δίοδος D η οποία θα γεφυρώσει και θα βραχυκυκλώσει την τάση αυτή (Σχήμα 21.6). Έτσι, το παραγόμενο

[3] Μπορεί να ανέρχεται σε ορισμένες εκατοντάδες Volt.

ρεύμα από την ανάστροφη υπέρταση θα αναλωθεί στο βρόγχο δίοδος-φορτίο θερμαίνοντας την αντίσταση του φορτίου.



Σχήμα 21. 5 Οδήγηση επαγωγικού φορτίου από μικροελεγκτή. Η δίοδος χρησιμοποιείται για την προστασία του τρανζίστορ από αντίστροφα ρεύματα

Η τεχνική της Διαμόρφωσης Εύρους Παλμών (P.W.M)

Η **Διαμόρφωση Εύρους Παλμών** γνωστή και ως **«PWM»** («Pulse Width Modulation») ή αλλιώς Pulse Duration Modulation («PDM») είναι μια τεχνική δημιουργίας ενός ψευδοαναλογικού σήματος το οποίο ουσιαστικά αντιστοιχεί σε ένα σήμα τετραγωνικού παλμού συγκεκριμένου πλάτους 5volt, σταθερής συχνότητας (για τον Arduino είναι περίπου 490Hz ή 960Hz ανάλογα την ακίδα εξόδου), αλλά με δυνατότητα μεταβολής του εύρους παλμού, αλλάζοντας με αυτόν τον τρόπο τη μέση τιμή της τάσης κατά την διάρκεια μιας περιόδου. Το σήμα **PWM** ουσιαστικά μπορεί να δημιουργηθεί από το άνοιγμα και το κλείσιμο ενός διακόπτη, για καθορισμένα χρονικά διαστήματα t_{on} και t_{off}, κρατώντας σταθερή τη συχνότητα, στην περίπτωσή μας η λειτουργία του διακόπτη πραγματοποιείται με ηλεκτρονικό τρόπο.

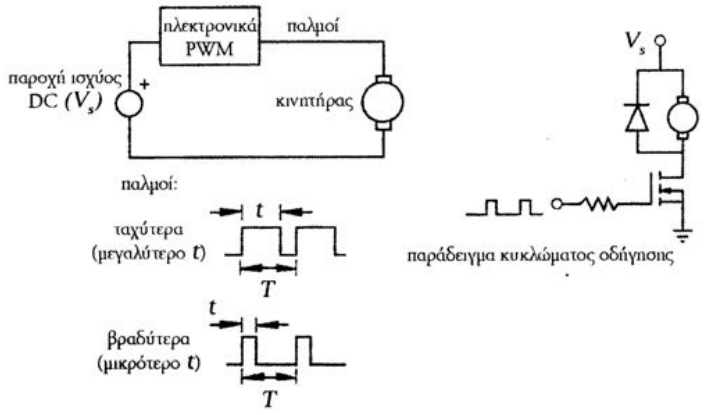
Γενικώς, η αρχή λειτουργίας ενός ενισχυτή PWM παρουσιάζεται στο σχήμα 5.16. Μια τάση DC, μεταβάλλεται με δεδομένη συχνότητα **f** μεταξύ δυο τιμών (ON, OFF). Στις περισσότερες περιπτώσεις η συχνότητα του σήματος υπερβαίνει το **1 kHz.** Η υψηλή τιμή διατηρείται κατά τη διάρκεια ενός παλμού μεταβλητού πλάτους **t** με περίοδο **T**, όπου

$$T = \frac{1}{f}$$

Η προκύπτουσα **ασύμμετρη κυματομορφή** έχει «κύκλο λειτουργίας» ή **duty cycle,** ο οποίος ορίζεται ως το κλάσμα μεταξύ της χρονικής διάρκειας όπου η τάση λαμβάνει τιμή ON και της περιόδου της κυματομορφής

$$Duty\ cycle = \frac{t_{on}}{T} 100\%$$

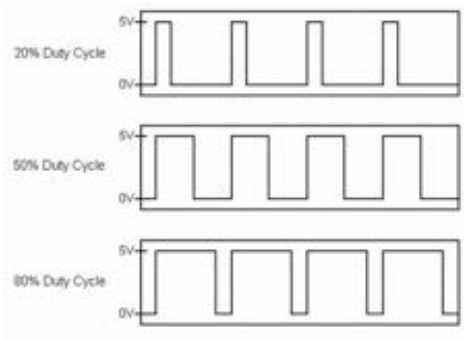
π.χ. για Duty cycle 50% πρέπει $t_{on}=t_{off}$.



Σχήμα 5.16: Διαμόρφωση παλμικού πλάτους ενός κινητήρα DC

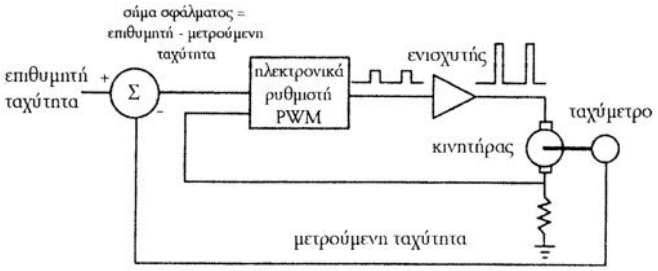
Με τη μεταβολή του **duty cycle** από τον ελεγκτή, μεταβάλλεται και το μέσο ρεύμα που διαρρέει τον κινητήρα, προκαλώντας έτσι μεταβολές στην ταχύτητα και τη ροπή στην έξοδο του. Για τον έλεγχο της ταχύτητας του κινητήρα κυρίως χρησιμοποιείται ο **duty cycle** και όχι η τιμή της τάσης. Σήμα PWM μπορούμε να παράγουμε εύκολα με τον μικροελεγκτή Arduino χρησιμοποιώντας την εντολή analogWrite (pin_number,0-255).

Στην Εικόνα 1.13.2 εμφανίζονται τρία διαφορετικά **P.W.M** σήματα, το πρώτο έχει κύκλο λειτουργίας (duty cycle) 20% της περιόδου, το δεύτερο έχει 50% ενώ το τρίτο το 80%. Τα σήματα έχουν την ίδια συχνότητα, αλλά διαφέρουν ως προς το εύρος του τετραγωνικού παλμού.



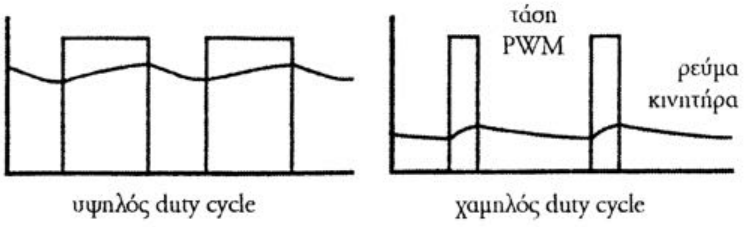
Εικόνα 1.13.2 Παράδειγμα του «duty cycle»

Στο σχήμα 5.17 απεικονίζεται το διάγραμμα ενός συστήματος ελέγχου ανάδρασης ταχύτητας PWM για έναν κινητήρα DC. Μια ταχογεννήτρια παράγει τάση, που είναι γραμμικά ανάλογη της ταχύτητας του κινητήρα. Αυτή συγκρίνεται με την επιθυμητή ταχύτητα (το set point), η οποία είναι μια τάση, που μπορεί να καθοριστεί, είτε χειροκίνητα, είτε μέσω υπολογιστή. Το σφάλμα ταχύτητας εκφρασμένο σε τάση και το ρεύμα που διαρρέει τον κινητήρα εκφρασμένο σε τάση πάνω στην αντίσταση, χρησιμοποιούνται από έναν ρυθμιστή διαμόρφωσης πλάτους παλμού (μικροελεγκτή ή κατάλληλο ολοκληρωμένο κύκλωμα), του οποίου η έξοδος είναι ένα κύμα τετραγωνικών παλμών διαμορφωμένο ως προς το εύρος του παλμού. Το εύρος ρυθμίζεται ανάλογα με από το σφάλμα ταχύτητας και ενισχύεται σε επίπεδα κατάλληλα για την οδήγηση του κινητήρα.



Σχήμα 5.17: Έλεγχος ανάδρασης ταχύτητας PWM

Σ' έναν ελεγκτή κινητήρα PWM η τάση του οπλισμού μεταβάλλεται με μεγάλη ταχύτητα, έτσι ώστε το ρεύμα, που διαρρέει τον κινητήρα να επηρεάζεται από την αντίσταση και την επαγωγή του. Επειδή η ταχύτητα μεταβολής είναι μεγάλη, η ένταση του ρεύματος, που διαρρέει τον κινητήρα παρουσιάζει μια μικρή διακύμανση γύρω από μια μέση τιμή, όπως παριστάνεται στο διάγραμμα 5.18. Καθώς μεγαλώνει ο duty cycle, μεγαλώνει και η μέση ένταση με αποτέλεσμα να αυξάνεται τελικά και η ταχύτητα του κινητήρα. Θυμηθείτε ότι σε ένα DC κινητήρα το ρεύμα που διαρρέει το ρότορα είναι ανάλογο με την ροπή που εμφανίζει στον άξονά του.



Σχήμα 5.18: Τάση PWM και ρεύμα κινητήρα

Οι κινητήρες τύπου «RC SERVO»

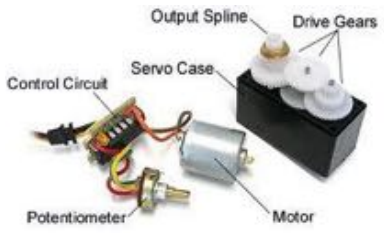
Η λέξη «servo» προέρχεται από την λατινική λέξη «servus» που σημαίνει υπηρέτης, μπορεί να θεωρηθεί η λειτουργία του κινητήρα έτσι αφού ο σερβοκινητήρας είναι ένας κινητήρας ο οποίος μπορεί να κινηθεί με καλή ακρίβεια ακολουθώντας τις εντολές που θα του δοθούν. Οι σερβοκινητήρες είναι μια

εξελιγμένη μορφή των απλών κινητήρων με βελτιωμένα λειτουργικά χαρακτηριστικά. Είναι κινητήρες συνεχούς ρεύματος με ένα ενσωματωμένο σύστημα γραναζιών και ένα κύκλωμα ελέγχου με ανάδραση θέσης. Σε γενικές γραμμές το μέγεθος τους είναι μικρό (9 με 55 gr) σε σχέση με την ροπή (1.6 kg-cm με 9.4 kg-cm) που μπορούν να δημιουργήσουν και την ταχύτητα που μπορούν να περιστραφούν (60^o/0.12 sec με 60^o/0.20 sec) ενώ σε συνδυασμό με το χαμηλό τους κόστος (3 ευρώ με 60 ευρώ) αποτελούν την καλύτερη λύση για την χρήση σε ρομποτικά συστήματα μικρής κλίμακας.

Εικόνα 4.1.1 Κινητήρας τύπου «RC Servo»

Η συνδεσμολογία τους είναι απλή αφού αποτελείται από **τρία** καλώδια, το πρώτο είναι η γείωση (-), το δεύτερο η τάση τροφοδοσίας (+), και το τρίτο το σήμα PWM με το οποίο ελέγχουμε την θέση. Ένας σερβοκινητήρας αποτελείται από έξη βασικά μέρη:

- το κύκλωμα ελέγχου («Control Circuit»), το οποίο διαβάζει τα σήματα PWM που δέχεται ο κινητήρας και αναλόγως τον περιστρέφει στην επιθυμητή θέση
- τον κινητήριο άξονα («Output Spline»)
- το σύστημα γραναζιών («Drive Gears»), το οποίο μεταφέρει την κίνηση στον άξονα
- το ποτενσιόμετρο («Potentiometer»), το οποίο συνδέεται με τον κινητήριο άξονα και περιστρέφεται μ' αυτόν έτσι ώστε να μεταβάλλει η αντίσταση του
- τον κινητήρα («Motor»)



Εικόνα 4.1.3 Εσωτερικό ενός κινητήρα τύπου «RC SERVO»

Ο έλεγχος του σερβοκινητήρα γίνεται με τη χρήση της τεχνολογίας «Pulse Width Modulation» (PWM),η λειτουργία της οποίας έχει αναφερθεί λεπτομερώς στο Κεφάλαιο 1 (1.4.8.3), το σήμα τετραγωνικών παλμών είναι 50Hz άρα περίοδο 20ms και ανάλογα με το εύρος του προκύπτει η επιθυμητή γωνία. Στους

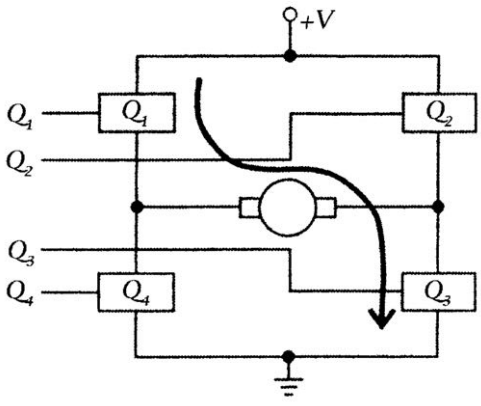
περισσότερους σερβοκινητήρες το εύρος του παλμού κυμαίνεται από 1ms έως 2ms, με δεδομένο ότι η γωνία μπορεί να είναι από 0 έως 180 μοίρες προκύπτει ότι αν το εύρος του παλμού είναι 1ms τότε η γωνία θα αντιστοιχεί σε 0 μοίρες, 1,5ms σε 90 μοίρες και 2ms σε 180 μοίρες[15].

Εικόνα 4.1.4 Εύρος παλμών του κινητήρα τύπου «RC SERVO»

Οδήγηση κινητήρα DC με Η γέφυρα (H-bridge)

Το πιο σύνηθες πρόβλημα στα μηχαντρονικά συστήματα αποτελεί αυτό του ελέγχου ενός κινητήρα συνεχούς ρεύματος. Στις περισσότερες περιπτώσεις, ο αντικειμενικός σκοπός μπορεί να είναι ο έλεγχος της ταχύτητας, της φοράς περιστροφής, της γωνίας και της ροπής του κινητήρα. Παρόλο, που είναι εφικτός ο σχεδιασμός και η κατασκευή ενός κυκλώματος οδήγησης του κινητήρα με διακριτά τμήματα ελέγχου και ισχύος, υπάρχουν πολλά ολοκληρωμένα κυκλώματα τα οποία εξοικονομούν χρόνο και χρήμα στο σχεδιασμό του μηχατρονικού συστήματος. Για να ελέγξουμε την ταχύτητα είναι απαραίτητο να μπορούμε να μεταβάλλουμε την παροχή ρεύματος στον κινητήρα ενώ για τον έλεγχο της φοράς περιστροφής πρέπει να είναι αναστρέψιμη η κατεύθυνση του ρεύματος. Αυτό απαιτεί έναν ενισχυτή ρεύματος και κάποια μέσα για την αλλαγή της φοράς του. Αυτές τις απαιτήσεις έρχεται να καλύψει η γέφυρα **Η** (H-bridge).

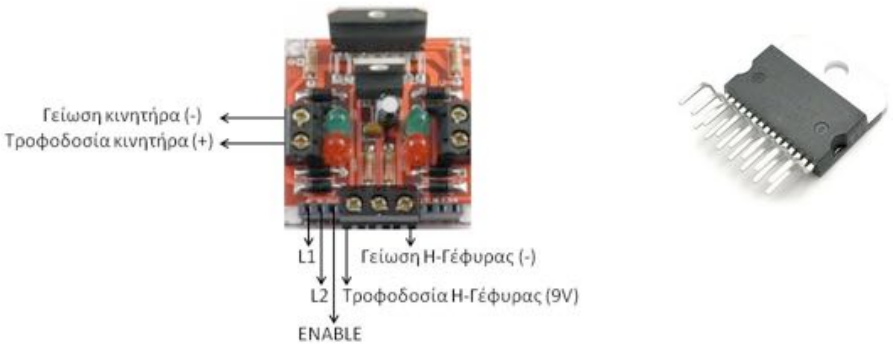
Η γέφυρα **Η** χρησιμοποιεί τέσσερα **transistors** ισχύος τοποθετημένα σε διάταξη Η, γύρω από τον κινητήρα DC και ενεργοποιεί εναλλάξ ένα ζευγάρι τη φορά, ώστε το ρεύμα να έχει την επιθυμητή. Εάν τα transistors Q1, Q3 είναι σε κατάσταση ON και τα Q2, Q4 σε OFF, η ροή του ρεύματος έχει τη φορά που φαίνεται στο σχήμα και ο κινητήρας περιστρέφεται προς τη μία φορά. Στην άλλη



περίπτωση (Q1, Q3: OFF - Q2, Q4: ON) ο κινητήρας περιστρέφεται αντίστροφα. Η κατασκευή μιας διακριτής γέφυρας είναι δυνατή, αλλά είναι δύσκολη η επιλογή των κατάλληλων transistors (BJTs, MOSFETs). Για το λόγο αυτό χρησιμοποιούμε διάφορα ολοκληρωμένα κυκλώματα (ICs) ελέγχου κίνησης τα οποία μπορούν να προσαρμοστούν άνετα για την οδήγηση κινητήρων DC. Κάποια από αυτά είναι η H - γέφυρες L298, LMD18200 και L293D, που είναι ειδικά σχεδιασμένες για τη οδήγηση κινητήρων DC και βηματικών. Επιτρέπουν τον έλεγχο της φοράς του ρεύματος και παρέχουν τη δυνατότητα ανίχνευσης υπερθέρμανσης, υπερτάσης (εκτός το L293D) και δυναμικού φρεναρίσματος του κινητήρα. Η λογική με την οποία χρησιμοποιούνται τα παραπάνω ολοκληρωμένα είναι σχεδόν η ίδια όπως θα αναδειχθεί και στην συνέχεια.

Οδήγηση DC κινητήρα με χρήση της H-Γέφυρας L298

Το ολοκληρωμένο κύκλωμα L298 χρησιμοποιείται σε εμπορικά διαθέσιμα κυκλώματα όπως αυτό της «Solarbotics» που παρουσιάζεται παρακάτω και μπορεί να οδηγήσει μέχρι και δύο κινητήρες, λειτουργεί από 6 Volt έως 26 Volt, με συνολικό ρεύμα οδήγησης 2 Ampere ανά κινητήρα.



Εικόνα 3.5.2 « Η - Γέφυρα» Solarbotics L298 με το ολοκληρωμένο κύκλωμα

Η λειτουργία του κυκλώματος βασίζεται στον παρακάτω πίνακα αληθείας όπου ανάλογα με το σήμα στις θύρες **Enable**, **L1**, **L2**, **L3** και **L4**, (HIGH ή LOW) επιφέρει κατάλληλο αποτέλεσμα στην κίνηση των κινητήρων, επίσης στην είσοδο Enable η H-Γέφυρα μπορεί να οδηγηθεί και με σήμα P.W.M. ρυθμίζοντας με αυτόν τον τρόπο την ταχύτητα περιστροφής του κινητήρα.

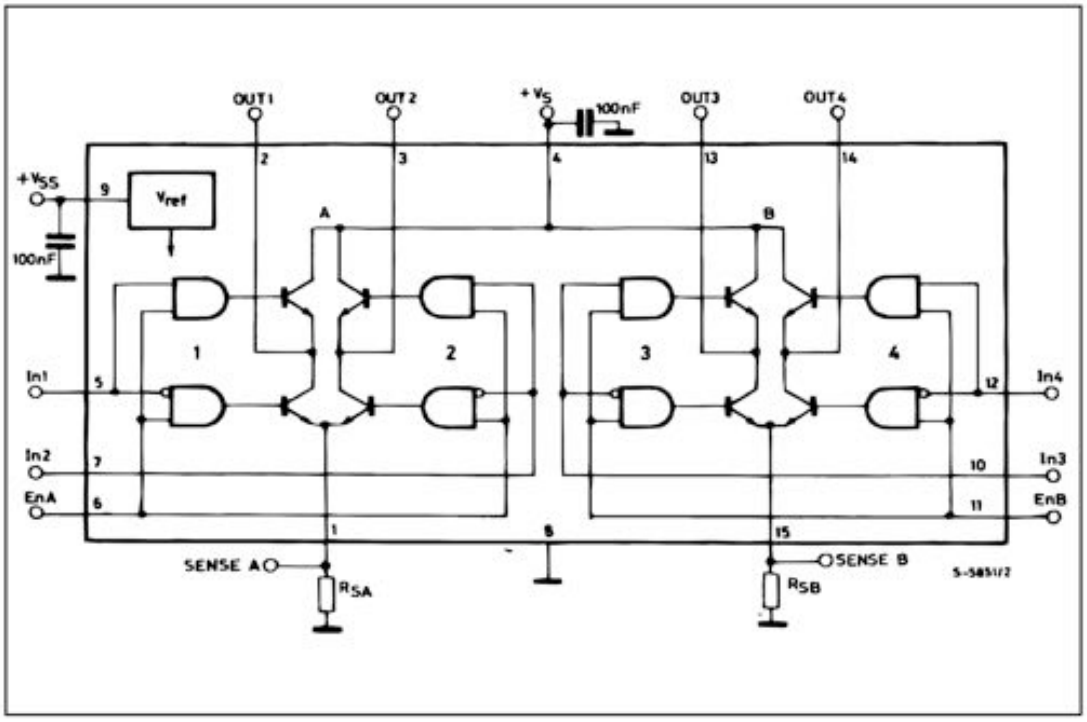
Logic Table

ENABLE	L1	L2	Result
L	L	L	OFF
L	L	H	OFF
L	H	L	OFF
L	H	H	OFF
H	L	L	BRAKE
H	L	H	FORWARD
H	H	L	BACKWARD
H	H	H	BRAKE
PWM	L	L	PULSE-BRK
PWM	L	H	FWD-SPD
PWM	H	L	BCK-SPD
PWM	H	H	PULSE-BRK

ENABLE	L3	L4	RESULT
L	L	L	OFF
L	L	H	OFF
L	H	L	OFF
L	H	H	OFF
H	L	L	BRAKE
H	L	H	FORWARD
H	H	L	BACKWARD
H	H	H	BRAKE
PWM	L	L	PULSE-BRK
PWM	L	H	FWD-SPD
PWM	H	L	BCK-SPD
PWM	H	H	PULSE-BRK

Εικόνα 3.5.3 Πίνακας αληθείας Solarbotics L298

Παρακάτω ακολουθεί το λειτουργικό διάγραμμα του ολοκληρωμένου L298 το οποίο περιλαμβάνεται στο παραπάνω κύκλωμα και ενσωματώνει δύο πλήρεις γέφυρες για την οδήγηση αντίστοιχων κινητήρων με δυνατότητα αλλαγής της φοράς κίνησης και μέτρησης των ρευμάτων που τους διαρρέουν. Στους ακροδέκτες OUT(1-2) συνδέεται ο κινητήρας 1, τα In(1-2) = L(1-2) καθορίζουν την κατάσταση κίνησης του (αριστερόστροφα, δεξιόστροφα, εξαναγκασμένο φρενάρισμα ή ελεύθερο φρενάρισμα) ενώ το EnA που ενεργοποιεί και απενεργοποιεί την λειτουργία της γέφυρας μπορεί να οδηγηθεί και με σήμα P.W.M.. Το ρεύμα του κινητήρα μπορεί να μετρηθεί σε μορφή τάσης από το SENSE A ενώ αντίστοιχα ισχύουν και για την σύνδεση του δεύτερου κινητήρα.

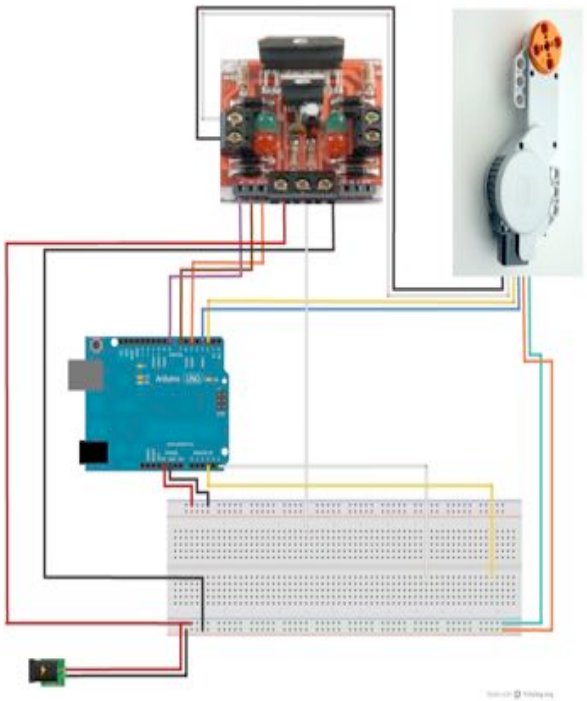


Εικόνα: το λειτουργικό διάγραμμα του ολοκληρωμένου L298

Στην συνέχεια, σύμφωνα με τον πίνακα αληθείας της προηγούμενης παραγράφου παραθέτουμε ένα απλό παράδειγμα χρήσης της **Η-Γέφυρας** από τον μικροελεγκτή Arduino ώστε να γίνει κατανοητή η

χρήση της για την οδήγηση ενός DC κινητήρα μόνιμου μαγνήτη της Lego. Ο κινητήρας συνδέεται στους ακροδέκτες της Η-Γέφυρας όπως φαίνεται στην Εικόνα 3.5.4 ενώ η **Η-Γέφυρα** ελέγχεται από τον μικροελεγκτή μέσα από τις θύρες L1, L2, και Enable (Pin 7, 8 και 5). Με το συγκεκριμένο πρόγραμμα ο κινητήρας θα κινείται συνεχόμενα μπροστά (FORWARD) καθώς το PIN 5 (Enable) το ορίζουμε HIGH, το PIN 7 (L1) το ορίζουμε LOW και το PIN 8 (L2) το ορίζουμε HIGH.

```
int motorPin0 = 7; //δήλωση της θύρας που είναι συνδεδεμένο το L1 της γέφυρας
int motorPin1 = 8; //δήλωση της θύρας που είναι συνδεδεμένο το L2 της γέφυρας
int motorPin2 = 5; //δήλωση της θύρας που είναι συνδεδεμένο το ENABLE της γέφυρας

void setup(){
pinMode(motorPin0, OUTPUT);
pinMode(motorPin1, OUTPUT);
pinMode(motorPin2, OUTPUT);
            }

Void loop (){
digitalWrite(motorPin0,LOW); //δήλωση του PIN 7 του κινητήρα ως LOW
digitalWrite(motorPin1,HIGH); //δήλωση του PIN 8 του κινητήρα ως HIGH
digitalWrite(motorPin2,HIGH); //δήλωση του PIN 5 του κινητήρα ως HIGH
          }
```

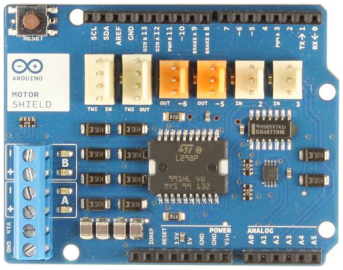


Εικόνα 3.5.4 Συνδεσμολογία της «Η-Γέφυρας» με τον κινητήρα NXT και τον Arduino Uno

Οδήγηση DC κινητήρα με χρήση Η-Γέφυρας του motor shield

Στόν παρακάτω πίνακα παρουσιάζονται τα τεχνικά χαρακτηριστικά και οι ακροδέκτες ελέγχου του motor shield σύνδεσης με τον μικροελεγκτή Arduino, το οποίο χρησιμοποιεί το **ολοκληρωμένο L298P** και μπορεί να οδηγήσει αντίστοιχα δύο κινητήρες μόνιμου μαγνήτη μέσα από τα κανάλια A και B, ή όπως θα δούμε αργότερα ένα βηματικό κινητήρα.

Operating Voltage Vin	5V to 12V
Motor controller	L298P, Drives 2 DC motors or 1 stepper motor
Max current	2A per channel or 4A max (with external power supply)
Current sensing	1.65V/A, AD converter in 3V3 for max current 2A
Free running stop and brake function	



Function	**Channel A**	**Channel B**
Direction	Digital 12	Digital 13
Speed (PWM)	Digital 3	Digital 11
Brake	Digital 9	Digital 8
Current Sensing	Analog 0	Analog 1

Παρατηρήστε οτι τα A0 και A1 είναι δεσμευμένα από το shield για την μέτρηση των ρευμάτων στους κινητήρες και δεν μπορούν να χρησιμοποιηθούν για την εισαγωγή κάποιου αναλογικού σήματος. Στο παρακάτω παράδειγμα γίνεται οδήγηση ενός κινητήρα αρχικά προς την μία κατεύθυνση με την μέγιστη ταχύτητα ενώ στην συνέχεια φρενάρει και περιστρέφεται ανάποδα με την μισή ταχύτητα.

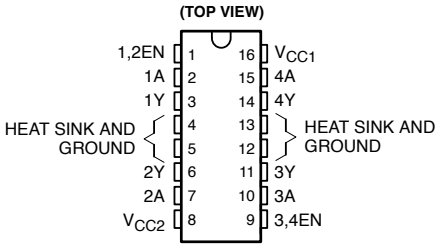
```
void setup(){ pinMode(12, OUTPUT); //DIRECTION Motor Channel A
              pinMode(9, OUTPUT); // BRAKE Motor Channel A
            }
void loop()  {

  //forward @ full speed
  digitalWrite(12, HIGH); //Establishes forward direction of Channel A
  digitalWrite(9, LOW);    //Disengage the Brake for Channel A
  analogWrite(3, 255);    //Spins the motor on Channel A at full speed

  delay(3000);
  digitalWrite(9, HIGH); //Engage the Brake for Channel A
  delay(1000);

  //backward @ half speed
  digitalWrite(12, LOW); //Establishes backward direction of Channel A
  digitalWrite(9, LOW);    //Disengage the Brake for Channel A
  analogWrite(3, 123);    //Spins the motor on Channel A at half speed

  delay(3000);
  digitalWrite(9, HIGH); //Eengage the Brake for Channel A
  delay(1000);}
```

Οδήγηση DC κινητήρα με χρήση της H-Γέφυρας L293D

Το ολοκληρωμένα **L293D**, αποτελεί έναν οδηγό ισχύος τύπου γέφυρας (**H-BridgeDrivers**) της εταιρείας **TexasInstruments**. Η μορφή του ολοκληρωμένου κυκλώματος φαίνεται στο επόμενο σχήμα (Σχήμα 4.3).

Σχήμα 4.3: *Οι οδηγοί ισχύος τύπου γέφυρας .*

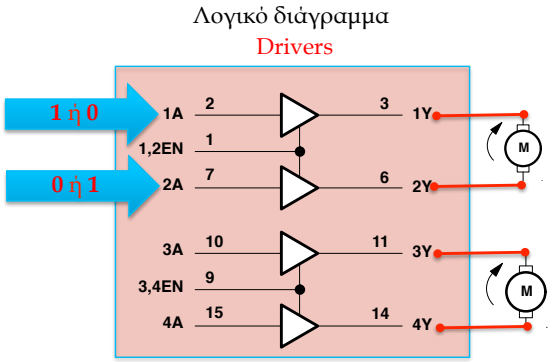
Τα ολοκληρωμένα αυτά είναι δεκαέξι ακροδεκτών, τέσσερις από τους οποίους (είσοδοι) ελέγχονται από ψηφιακά σήματα. Κάθε ένας από τους ακροδέκτες αυτούς ελέγχει έναν ακροδέκτη (έξοδος), στον οποίο συνδέεται το επαγωγικό φορτίο. Χαρακτηριστικά παραδείγματα επαγωγικών φορτίων αποτελούν τα ρελέ, τα πηνία, οι DC κινητήρες και οι διπολικοί βηματικοί κινητήρες. Επιπλέον, τα ολοκληρωμένα αυτά διαθέτουν δύο ακροδέκτες, οι οποίοι τα τροφοδοτούν με την αναγκαία ισχύ (pins 8, 16) και άλλους τέσσερις ακροδέκτες (pins 4, 5, 12, 13), οι οποίοι γειώνονται. Όλα αυτά δίνονται στο παρακάτω σχήμα (Σχήμα 4.4).



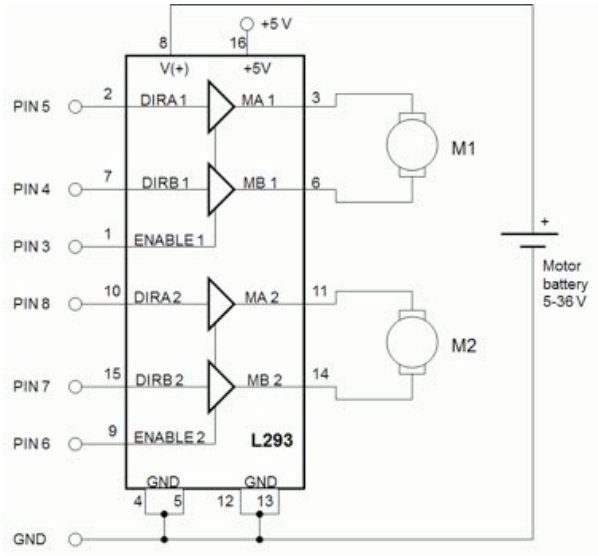
Σχήμα 4.4: Διάταξη ακροδεκτών των οδηγών ισχύος τύπου γέφυρας L293D.

Παρατηρώντας κανείς το παραπάνω σχήμα μπορεί να διακρίνει τέσσερις οδηγούς: ο πρώτος αποτελείται από τους ακροδέκτες 2 και 3, ο δεύτερος από τους ακροδέκτες 6 και 7, ο τρίτος από τους ακροδέκτες 10 και 11 και ο τέταρτος από τους ακροδέκτες 14 και 15. Για να λειτουργήσει κάποιος από τους τέσσερις οδηγούς απαιτείται τροφοδοσία στους ακροδέκτες V_{CC1} και V_{CC2} (pins 8, 16). Ο πρώτος δέχεται τάσεις από 4.5 Volts έως 7 Volts και ο δεύτερος από V_{CC1}Volts έως 36 Volts. Στην παρούσα σύνδεση λήφθηκαν V_{CC1}=5 Volts και V_{CC2}=9 Volts. Επιπλέον, οι ακροδέκτες 4, 5, 12 και 13 γειώνονται από κοινού. Οι ακροδέκτες 1 και 9 ελέγχουν (ενεργοποιούν και απενεργοποιουν) τη λειτουργία των τεσσάρων οδηγών. Εφαρμόζοντας σε έναν από αυτούς τάση 5 Volts ενεργοποιείται το αντίστοιχο ζεύγος οδηγών. Με τον

τρόπο αυτό, στέλνοντας ψηφιακό HIGH ή LOW στην κατάλληλη είσοδο (2 ή 6), η αντίστοιχη έξοδος (3 ή 7) λαμβάνει τάση V_{CC2} ή LOW και βρίσκεται σε φάση με την είσοδο. Ο πίνακας αλήθειας και το λογικό διάγραμμα του ολοκληρωμένου L293D δίνονται στο παρακάτω σχήμα (Σχήμα 4.5).



Σχήμα 4.5: *Το λογικό διάγραμμα των* ***οδηγών ισχύος*** *τύπου γέφυρας L293D.*



Στις εξόδους των οδηγών 1:(MA1) και 2:(MB1) μπορεί να συνδεθεί ο κινητήρας μας με αποτέλεσμα, εφαρμόζοντας την pwm λειτουργία στην είσόδο ENABLE1 των οδηγών αυτών, να επιτυγχάνεται αμφίδρομη περιστροφή των κινητήρων η οποία καθορίζεται από τις εισόδους DIRA1 και DIRB1 των οδηγών. Οι έξοδοι των οδηγών παρέχουν έως και 600 mA στα 9 Volts, ισχύς επαρκής, τόσο για τη λειτουργία μικρών κινητήρων.

Σημειώνεται πως οι δίοδοι εξόδου υψηλής ταχύτητας SES5001 χρησιμοποιούνται για την καταστολή του μεταβατικού φαινομένου κατά τη λειτουργία των ολοκληρωμένων L293D **και βρίσκονται ενσωματωμένες** σε αυτά παρόλο που δεν εμφανίζονται στο παραπάνω σχήμα. Παρακάτω

παρουσιάζεται ένα ενδεικτικό πρόγραμμα ελέγχου DC κινητήρα με την γέφυρα L293D και τον μικροελεγκτή Arduino.

```
/************************
Exercise the motor using
the L293 chip
************************/

#define ENABLE 3 //controller's output pins
#define DIRB 4
#define DIRA 5

void setup() {
  int i;

  pinMode(ENABLE,OUTPUT);
  pinMode(DIRA,OUTPUT);
  pinMode(DIRB,OUTPUT);
            }


   void loop() { • • • • • • }
```

```
//---back and forth example
  digitalWrite(ENABLE,HIGH); // enable on

for (i=0;i<5;i++) {
    digitalWrite(DIRA,HIGH); //one way
    digitalWrite(DIRB,LOW);
    delay(500);
    digitalWrite(DIRA,LOW);  //reverse
    digitalWrite(DIRB,HIGH);
    delay(500);
                    }
  digitalWrite(ENABLE,LOW); // disable
  delay(4000);
```

```
//---fast/slow stop example
digitalWrite(ENABLE,HIGH); //enable on
digitalWrite(DIRA,HIGH); //one way
digitalWrite(DIRB,LOW);
delay(1000);

digitalWrite(ENABLE,LOW); //slow stop
delay(3000);
digitalWrite(ENABLE,HIGH); //enable on
digitalWrite(DIRA,HIGH); //one way
digitalWrite(DIRB,LOW);
delay(1000);

digitalWrite(DIRA,LOW); //fast stop
delay(3000);
```

```
//---PWM example, full speed then slow
  digitalWrite(ENABLE,HIGH); //enable on
  digitalWrite(DIRA,HIGH); //one way
  digitalWrite(DIRB,LOW);
  delay(2000);
  analogWrite(ENABLE,128); //half speed
  delay(2000);
  digitalWrite(ENABLE,LOW); //all done
```

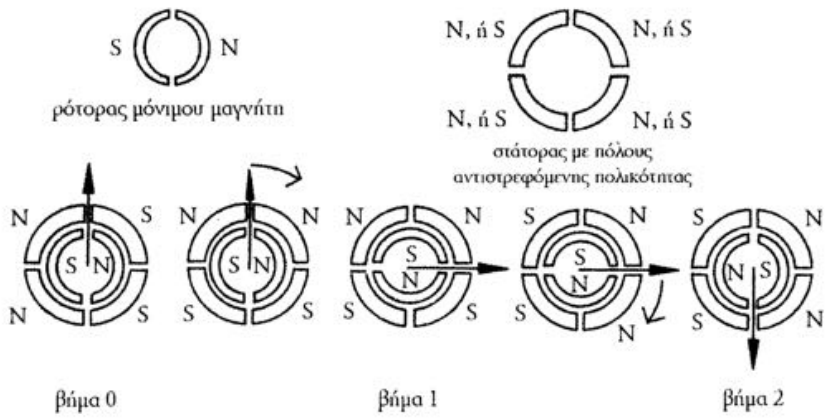
Βηματικοί κινητήρες

Μια ειδική κατηγορία κινητήρων DC είναι οι βηματικοί κινητήρες· πρόκειται για κινητήρες DC μόνιμου μαγνήτη, οι οποίοι έχουν τα εξής χαρακτηριστικά λειτουργίας: μπορούν να περιστραφούν και στις δυο κατευθύνσεις, να κινηθούν με ακρίβεια κατά συγκεκριμένη γωνία, να διατηρήσουν ροπή συγκράτησης υπό μηδενική ταχύτητα και να ελεγχθούν με ψηφιακά κυκλώματα. Η ακριβής κίνηση κατά τμήματα συγκεκριμένης γωνίας, που ονομάζονται βήματα (steps), γίνεται με την εφαρμογή ψηφιακών παλμών σε ένα ηλεκτρικό κύκλωμα οδήγησης. Ο αριθμός και ο ρυθμός των παλμών ελέγχουν τη θέση και την ταχύτητα του άξονα του κινητήρα. Γενικά οι βηματικοί κινητήρες κατασκευάζονται με δυνατότητα 12, 24, 72, 144, 180 και 200 βημάτων ανά πλήρη περιστροφή, που σημαίνει ότι ο άξονας περιστρέφεται 30°, 15°, 5°, 2.5°, 2° και 1.8° ανά βήμα. Εάν υπάρχει ανάγκη ακριβέστερης κίνησης, είναι δυνατό να σχεδιαστούν κυκλώματα μικρο-βηματισμού (micro-stepping), που να επιτρέπουν πολύ περισσότερα βήματα ανά περιστροφή· ο αριθμός μπορεί να φθάνει, ή και να ξεπερνά τα 10000 steps/rev.

Οι βηματικοί κινητήρες είναι είτε μονοπολικοί απαιτώντας μόνο μια πηγή τροφοδοσίας, είτε διπολικοί απαιτώντας δυο πηγές ενέργειας ή μια πηγή με δυνατότητα αλλαγής της πολικότητας,. Τροφοδοτούνται από πηγές DC και χρειάζονται ψηφιακά κυκλώματα για την εκκίνηση των διαδικασιών ενεργοποίησης των πηνίων, ώστε να περιστραφεί ο κινητήρας. Η ανάδραση δεν είναι πάντα απαραίτητη, αλλά η χρήση ενός κωδικοποιητή, ή κάποιου άλλου αισθητή θέσης μπορεί να εξασφαλίσει ακρίβεια στις εφαρμογές, που ο ακριβής έλεγχος θέσης αποτελεί βασική προϋπόθεση. Το πλεονέκτημα της λειτουργίας χωρίς ανάδραση είναι, ότι δε χρειάζεται το σύστημα ελέγχου κλειστού βρόχου. Γενικά οι βηματικοί κινητήρες αποδίδουν λιγότερο από 1 hp (746 W) με αποτέλεσμα να χρησιμοποιούνται μόνο σε εφαρμογές ελέγχου θέσης χαμηλής ισχύος.

Ένας βηματικός κινητήρας, που διατίθεται στο εμπόριο, αποτελείται από έναν μεγάλο αριθμό πόλων, που καθορίζει έναν αντίστοιχα μεγάλο αριθμό θέσεων ισορροπίας του κινητήρα. Στην περίπτωση ενός βηματικού κινητήρα μόνιμου μαγνήτη, ο στάτορας αποτελείται από πόλους με περιέλιξη, ενώ οι πόλοι του ρότορα είναι μόνιμοι μαγνήτες. Η διέγερση διαφορετικών συνδυασμών περιελίξεων μετακινεί και σταθεροποιεί τον ρότορα σε διαφορετικές θέσεις.

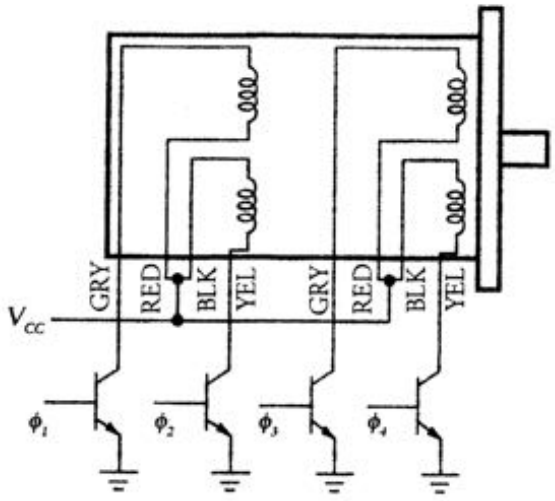
Για να κατανοήσετε την κατά βήματα περιστροφή ενός βηματικού κινητήρα, θεωρήστε μια απλή διάταξη, η οποία αποτελείται από τέσσερις πόλους στάτορα (κάθε ζεύγος πόλων του στάτορα δημιουργεί ένα ζεύγος Βόρειου - Νότιου πόλου) και έναν ρότορα μόνιμου μαγνήτη, όπως φαίνεται στο σχήμα 5.19.



Σχήμα 5.19: Ακολουθία βηματικού κινητήρα

Στο βήμα 0 ο ρότορας είναι σε ισορροπία, διότι οι αντίθετοι πόλοι του στάτορα και του ρότορα είναι απέναντι και έλκονται. Ο κινητήρας μπορεί να μείνει σ' αυτή τη θέση και να ανταπεξέλθει σε ροπή, της οποίας η μέγιστη τιμή ονομάζεται **ροπή συγκράτησης** (holding torque), εφόσον οι πολικότητες των πηνίων του στάτορα δε μεταβληθούν. Σε περίπτωση αλλαγής της πολικότητας του στάτη (βήμα 0 → βήμα 1), εφαρμόζεται ροπή στον ρότορα, αναγκάζοντάς τον να κινηθεί στην ωρολογιακή διεύθυνση κατά 90°, σε μια νέα θέση ισορροπίας (βήμα 1). Όταν μεταβληθούν ξανά οι **πολικότητες** του στάτορα (βήμα 1 → βήμα 2), τότε ο ρότορας υπόκειται σε ροπή, που τον μεταφέρει στο βήμα 2. Αλλάζοντας διαδοχικά τις πολικότητες μ' αυτό τον τρόπο, ο ρότορας μπορεί να μετακινηθεί σε διαδοχικές θέσεις ισορροπίας στην ωρολογιακή φορά. Η αντι-ωρολογιακή κίνηση μπορεί να επιτευχθεί αλλάζοντας τις πολικότητες στην αντίθετη κατεύθυνση. Η ροπή του κινητήρα συνδέεται άμεσα με τη ισχύ του μαγνητικού πεδίου των πόλων και του ρότορα και το ρεύμα στα πηνία των πόλων του βηματικού κινητήρα.

Στην περίπτωση που η εφαρμογή του κινητήρα περιλαμβάνει ταχεία εκκίνηση και παύση, ταχεία αλλαγή, ή κλιμάκωση της ταχύτητας, ή το χειρισμό μεγάλων, ή μεταβαλλόμενων φορτίων πρέπει να ληφθούν σοβαρά υπόψη η δυναμική απόκριση του κινητήρα και το φορτίο. Λόγω της αδράνειας του ρότορα και του φορτίου του κινητήρα, είναι δυνατό η περιστροφή να υπερβεί τον επιθυμητό αριθμό βημάτων.



Σχήμα 5.22: Σχηματικό διάγραμμα πηνίου, που προκαλεί πεδίο σε μονοπολικό βηματικό κινητήρα

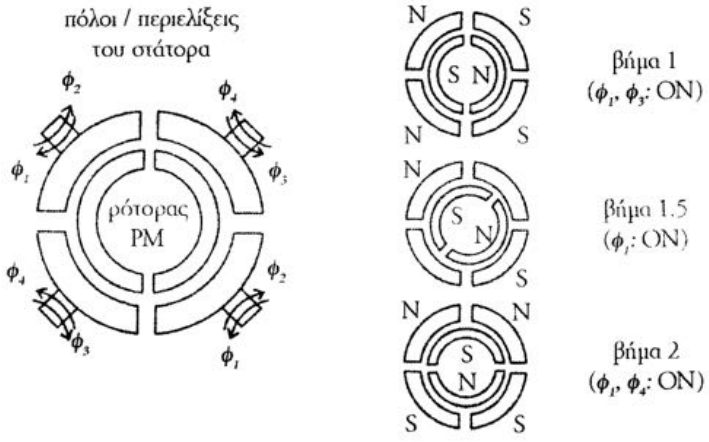
Στο σχήμα 5.23 παρουσιάζεται η κατασκευή και η ακολουθία των βημάτων ενός μονοπολικού βηματικού κινητήρα τεσσάρων φάσεων. Αποτελείται από έναν διπολικό ρότορα μόνιμου μαγνήτη και έναν στάτορα τεσσάρων πόλων, καθένας από τους οποίους τροφοδοτείται από δυο συμπληρωματικές περιελίξεις (στις φ1 και φ2 διαρρέονται από ρεύμα αντίθετης κατεύθυνσης). Στον πίνακα 5.1 καταχωρείται η ακολουθία φάσεων για την πλήρη περιστροφή του κινητήρα, όπου ενεργοποιούνται δυο από τις τέσσερις φάσεις (κατάσταση ON) για να πολωθεί κάθε πόλος του στάτορα. Στον πίνακα 5.2 καταχωρείται η ακολουθία φάσεων για τον βηματισμό μισού βήματος (half-stepping) του κινητήρα, όπου για κάθε πλήρες βήμα είναι ενεργοποιημένη μόνο μια φάση και ενεργοποιούνται μόνο δυο από τους πόλους του στάτορα. Η διακριτότητα, ή ο αριθμός των βημάτων είναι διπλάσιος στην ημι-βηματική λειτουργία (8 steps/rev στις 45°) σε σχέση με τη βηματική λειτουργία (full-step mode: 4 steps/rev στις 90°). Στην ημιβηματική λειτουργία η ροπή λειτουργίας και η ροπή συγκράτησης μεταβάλλονται μεταξύ δυο τιμών σε εναλλασσόμενους κύκλους. Μια άλλη τεχνική, που χρησιμοποιείται για την αύξηση του αριθμού των βημάτων είναι ο μικρο-βηματισμός (micro-stepping), όπου τα φασικά ρεύματα ελέγχονται από κλασματικά μεγέθη, αντί των ON και OFF με αποτέλεσμα να δημιουργούνται περισσότερες θέσεις μαγνητικής ισορροπίας μεταξύ των πόλων. Στην πράξη, εφαρμόζονται στις φάσεις, διακριτά ημιτονοειδή κύματα αντί των τετραγωνικών. Οι συνηθέστεροι βηματικοί κινητήρες, που διατίθενται στο εμπόριο παρέχουν τη δυνατότητα 200 steps/rev σε βηματική λειτουργία και συχνά αναφέρονται ως βηματικοί 1.8° (360°/200). Στην μικρο-βηματική λειτουργία μπορούν να επιτευχθούν περισσότερα από 10000 steps/rev.

Βήμα	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF

Πίνακας 5.1: Ακολουθία φάσεων μονοπολικού κινητήρα (full step)

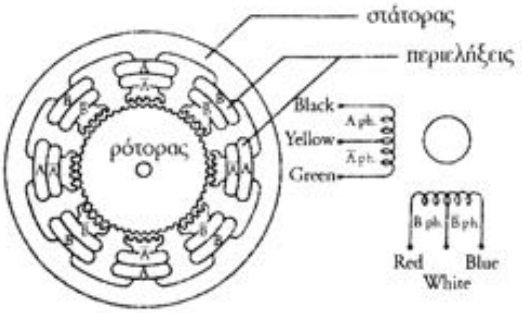
Βήμα	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	ON	OFF	ON	OFF
1.5	ON	OFF	OFF	OFF
2	ON	OFF	OFF	ON
2.5	OFF	OFF	OFF	ON
3	OFF	ON	OFF	ON
3.5	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
4.5	OFF	OFF	ON	OFF

Πίνακας 5.2: Ακολουθία φάσεων μονοπολικού κινητήρα (half step)

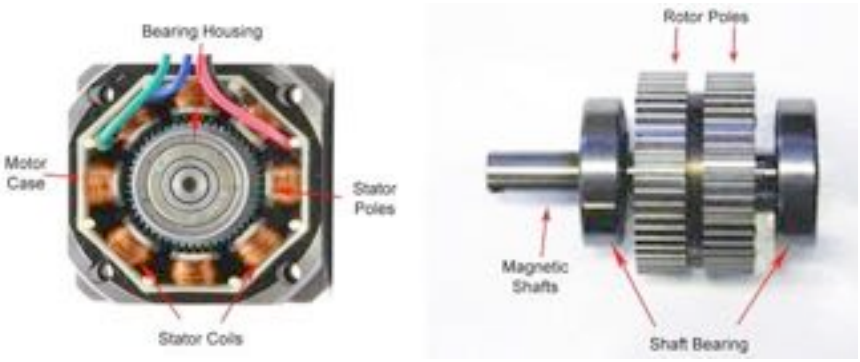


Σχήμα 5.23: Παράδειγμα μονοπολικού βηματικού κινητήρα

Στο σχήμα 5.24 παρουσιάζονται με περισσότερη λεπτομέρεια, η κατασκευή, η γεωμετρία των πόλων και οι συνδέσεις του πηνίου ενός πραγματικού βηματικού κινητήρα. Ο συγκεκριμένος κινητήρας μπορεί να διαμορφωθεί βάσει συνδεσμολογίας, ως μονοπολικός κινητήρας τεσσάρων φάσεων, ή διπολικός κινητήρας δυο φάσεων. Στην εικόνα 5.25 φαίνεται ένας χωριστός ρότορας 50 δοντιών, του οποίο η μια πλευρά έχει βόρεια πολικότητα και η άλλη νότια



Σχήμα 5.24: Τυπική διάταξη ρότορα και στάτορα βηματικού κινητήρα (ιδιοκτησία της Oriental Motor, Torrance, CA)



Βιβλιοθήκη βηματικών κινητήρων τύπου «STEPPER»

Υπάρχουν δύο είδη βηματικών κινητήρων τύπου **«Stepper»**, οι μονοπολικοί («Unipolar») και οι διπολικοί («Bipolar»), όπως οι προηγούμενες βιβλιοθήκες έτσι και η βιβλιοθήκη αυτών των κινητήρων υπάρχει προεγκατεστημένη στο περιβάλλον προγραμματισμού του Arduino (IDE). Η βιβλιοθήκη αποτελείται από τις παρακάτω τέσσερις συναρτήσεις:

- **Stepper(steps, pin1, pin2)**
- **Stepper(steps, pin1, pin2, pin3, pin4)**
- **setSpeed(rpms)**
- **step(steps)**

Η βιβλιοθήκη και για τους δύο τύπους κινητήρων είναι ίδια με μοναδική διαφορά τα τέσσερα καλώδια σύνδεσης που χρησιμοποιεί ο διπολικός κινητήρας σε αντίθεση με τα έξη που χρησιμοποιεί ο μονοπολικός. Για την σωστή χρήση της βιβλιοθήκης ακολουθούμε την εξής διαδικασία:

- Δηλώνουμε την βιβλιοθήκη στην αρχή του κώδικα ως **#include <Stepper.h>**

- Ορίζουμε τα συνολικά βήματα (steps) του κινητήρα για μία περιστροφή και τις θύρες που συνδέουμε τον κινητήρα στην υπολογιστική πλατφόρμα Arduino πριν την συνάρτηση **void setup()** :

  Π.χ. για τον μονοπολικό : **Stepper(steps, pin1, pin2)**

  Π.χ. για τον διπολικό : **Stepper(steps, pin1, pin2, pin3, pin4)**

- Ορίζουμε μέσα στην συνάρτηση **void setup()** την ταχύτητα κίνησης του κινητήρα σε στροφές ανά λεπτό (rpms) :

  Π.χ. **setSpeed(rpms)**

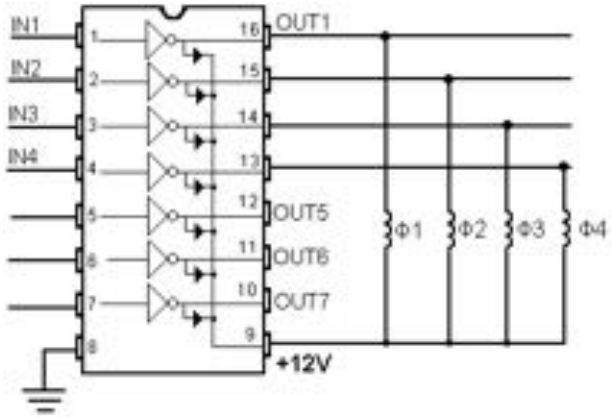
- Τέλος μέσα στην συνάρτηση **void loop()** δίνουμε τη επιθυμητή τιμή βημάτων (number_of_steps) ώστε να κινηθεί ο κινητήρας για τον αριθμό βημάτων που επιθυμούμε (για θετική τιμή ο κινητήρας στρέφεται αντίθετα από ότι για αρνητική τιμή βημάτων),π.χ. **step(number_of_steps)** :

  Π.χ. **Stepper: step(100**) ώστε ο κινητήρας να κάνει 100 βήματα

Κυκλώματα οδήγησης βηματικών κινητηρίων

Κυκλώματα οδήγησης μονοπολικών βηματικών κινητήρων με το το ULN2003

Στο παρακάτω σχήμα φαίνεται το κύκλωμα οδήγησης ενός μικρού μονοπολικού βηματικού κινητήρα 12V 0,5A με τη χρησιμοποίηση του ULN2003A.



Σχήμα A : *Οδήγηση των πηνίων μονοπολικού βηματικού κινητήρα χρησιμοποιώντας το ULN2003. κάθε πηνίο ενεργοποιείται όταν γίνει από το μικροελεγκτή η αντίστοιχη είσοδος INx =HIGH*

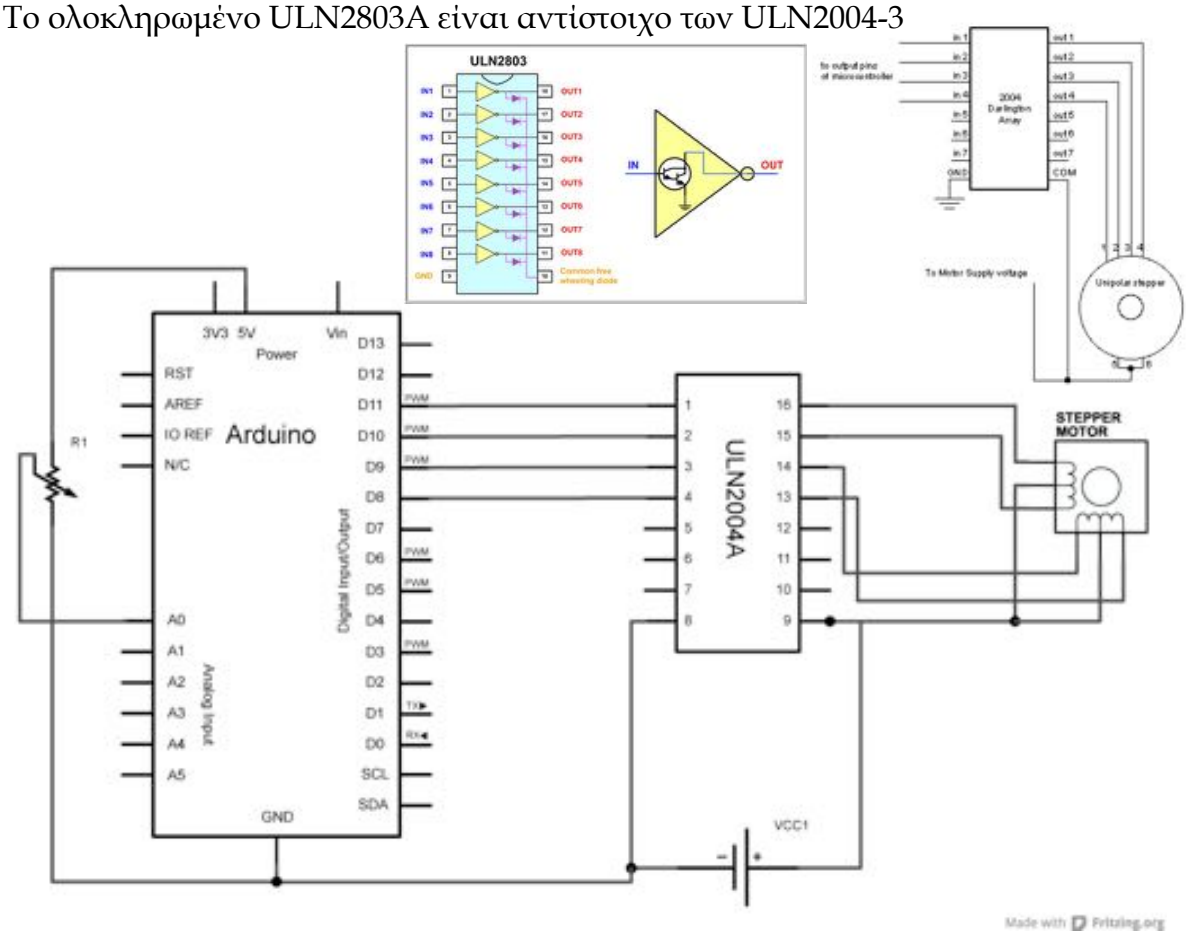
Το ολοκληρωμένο ULN2003A (Σχήμα 21.50), αποτελείται από επτά ενισχυτές DARLINGTON με κοινούς συλλέκτες. Οι έξοδοί του είναι τύπου ανοικτού συλλέκτη και κάθε μια από αυτές μπορεί να δώσει ρεύμα της τάξεως των 0,5A ανά DARLINGTON, ενώ στιγμιαία μπορεί να δώσει ρεύμα της τάξεως των 0,6A. Σε περίπτωση που απαιτούνται υψηλότερα ρεύματα μπορούν π.χ. δύο έξοδοι να συνδεθούν παράλληλα διπλασιάζοντας το παρεχόμενο ρεύμα. Κάθε ένας ενισχυτής διαθέτει στην έξοδό του μια δίοδο για την προστασία κατά την οδήγηση επαγωγικών φορτίων. Η οδήγηση λοιπόν, γίνεται **χωρίς να χρειάζεται η τοποθέτηση διόδων προστασίας** παράλληλα με τα πηνία, αφού αυτές είναι ενσωματωμένες στο ULN2003A.

Σύμφωνα με το παραπάνω για να βρεθεί το πηνίο Φ1 υπό τάση μεγαλύτερη από 0V θα πρέπει να έχει τοποθετηθεί η είσοδος IN1 = HIGH, προκειμένου να γίνει ο ακροδέκτης OUT1 = LOW, έτσι ώστε να βρεθεί το πηνίο Φ1 υπό τάση 12V. Η τάση που εφαρμόζεται στο πηνίο εξαρτάται από την τάση που εφαρμόζεται στην ακίδα 9 του ULN2003. Έτσι αν η τάση είναι π.χ. 24V, η τάση στα τυλίγματα θα ανέρχεται σε 24V, όποτε μια από τις εισόδους γίνει HIGH.

Τα ολοκληρωμένα ULN2003A (5V TTL, CMOS), ULN2004A (6–15V CMOS, PMOS), ULN2001A (DTL, TTL, PMOS, CMOS) και ULN2002A (14-25V PMOS), είναι κατάλληλα για την οδήγηση πηνίων, ηλεκτρονόμων, LED, κεφαλές θερμικών εκτυπωτών και πολλών άλλων συσκευών. Για το ULN2003A η διαφορά δυναμικού στην έξοδό του λαμβάνει την τιμή της τάσης που εφαρμόζεται στην ακίδα 9 (Σχήμα Α) όταν η αντίστοιχη είσοδός του είναι το λογικό 1 (HIGH).

Κύκλωμα σύνδεσης **μονοπολικού κινητήρα** με το ολοκληρωμλενο οδήγησης ULN2003-4 (Darlington Array) και τον μικροελεγκτή Arduino για τον έλεγχό με την χρήση της βιβλιοθήκης stepper.

- Το ολοκληρωμένο ULN2803A είναι αντίστοιχο των ULN2004-3



Εικόνα 1: Σύνδεση μονοπολικού βηματικού κινητήρα με το ολοκληρωμένο κύκλωμα οδήγησης ULN2004 (Darlington Array max 500mA per driver channel) για τον έλεγχό του από την βιβλιοθήκη ***<Stepper.h>****. Όπου Vcc1 η τάση τροφοδοσίας του βηματικού κινητήρα (https://www.arduino.cc/en/Tutorial/MotorKnob)*

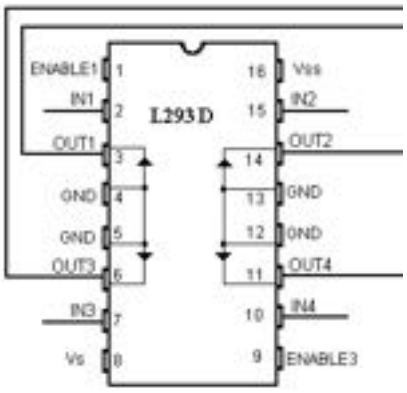
Το παρακάτω πρόγραμμα οδηγεί ένα μονοπολικό (ή διπολικό βηματικό) κινητήρα περιστρέφοντάς τον

με περιοδική εναλλαγή της φοράς περιστροφής του κάθε 500msec.

```
/*
 Stepper Motor Control - one revolution
 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 - 11 of the Arduino.
 The motor should revolve one revolution in one direction, then
 one revolution in the other direction.
 */

#include <Stepper.h>

const int stepsPerRevolution = 200; //change this to fit the number of steps
                                    //per revolution for your motor
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);// initialize the stepper
                                                    //library on pins 8 through 11:
void setup() {
     Serial.begin(9600); // initialize the serial port}
myStepper.setSpeed(60); // set the speed at 60 rpm:

void loop() {
  Serial.println("clockwise"); // step one revolution in one direction:
       myStepper.step(stepsPerRevolution);
       delay(500);
  Serial.println("counterclockwise"); //step one revolution in the other
                                      //direction:
       myStepper.step(-stepsPerRevolution);
       delay(500);
            }
```

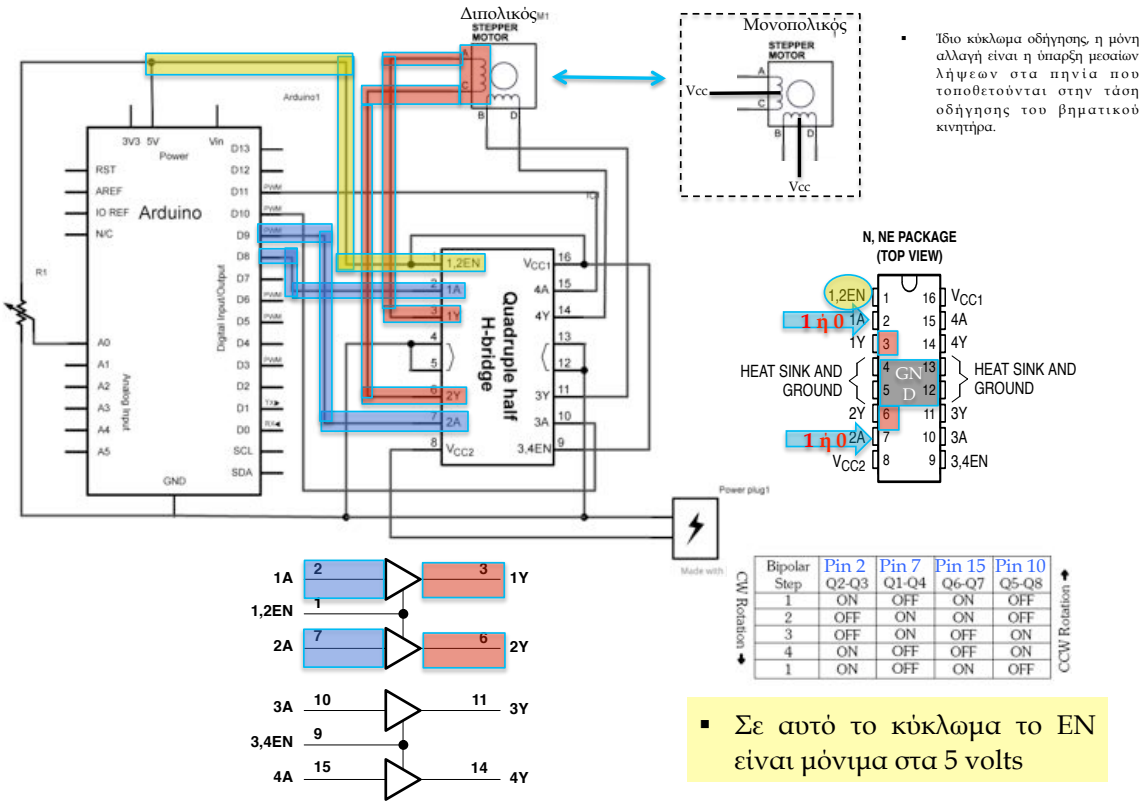
Οδήγηση διπολικών και μονοπολικών κινητήρων με το ολοκληρωμένο L293D

Το ολοκληρωμένο L293D που φαίνεται παρακάτω είναι κατάλληλο για την οδήγηση μονοπολικών και διπολικών βηματικών κινητήρων και όχι μόνον. Διατίθεται στο εμπόριο σε διάφορες εκδόσεις με την ικανότητα οδήγησης με ρεύμα έως 1A συνεχώς και 1,5A στιγμιαία. Στην έκδοση αυτή **υπάρχουν ενσωματωμένες δίοδοι προστασίας** συνδεδεμένες παράλληλα με το φορτίο για την οδήγηση επαγωγικών φορτίων.



Συγκεκριμένα, διαθέτει 4 εισόδους και 4 εξόδους με την έξοδο να λαμβάνει την τιμή που έχει η τάση Vs (από 5 μέχρι 30V), ενώ η τάση Vss ανέρχεται σε 5V.

Στην περίπτωση που θέλουμε να οδηγήσουμε ένα **διπολικό βηματικό κινητήρα** απαιτείται να χρησιμοποιήσουμε την παρακάτω συνδεσμολογία στην οποία χρησιμοποιείται το ολοκληρωμένο κύκλωμα **L293D** αντί αυτό του ULN2004 (Darlington Array).

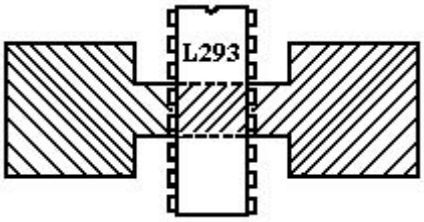


Bipolar Step	Pin 2 Q2-Q3	Pin 7 Q1-Q4	Pin 15 Q6-Q7	Pin 10 Q5-Q8
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON
1	ON	OFF	ON	OFF

Όπου Vcc2 η τάση τροφοδοσίας του βηματικού κινητήρα ενώ θέτουμε Vcc1 =5volt.

Είναι αξιοσημείωτο να πούμε ότι όταν θέλουμε να αλλάξουμε τον διπολικό με έναν μονοπολικό βηματικό κινητήρα, θα πρέπει να προσέξουμε οτι οι επιπρόσθετες μεσαίες λήψεις των πηνίων πρέπει να συνδεθούν στην τάση Vcc2 τροφοδοσίας του βηματικού κινητήρα, ενώ δεν χρειάζεται καμία αλλαγή στην συνδεσμολογία και το πρόγραμμα του μικροελεγκτή.

Όταν το ρεύμα είναι μεγάλο για αποφυγή υπερθέρμανσης του L293 θα πρέπει η κοινή γείωση των ακροδεκτών 4, 5, 12 και 13 να συνδέεται με μεγάλη μεταλλική επιφάνεια στην πλακέτα όπως φαίνεται στο παρακάτω για την απαγωγή της θερμότητας.



Σχήμα: Σύνδεση του L293 με μεγάλη σκιαγραφημένη επιφάνεια χαλκού στην πλακέτα στις ακίδες γείωσης για την επίτευξη ψύξης του ολοκληρωμένου

Σε περίπτωση που ο κινητήρας είναι μεγάλος και απορροφά μεγάλο ρεύμα μπορεί να χρησιμοποιηθεί το ολοκληρωμένο **L298**, που μπορεί να δώσει ρεύμα της τάξεως των 4A με τάση έως 46V.

ΠΑΡΑΡΤΗΜΑ Α:

Βασικές εντολές του Arduino με απλά παραδείγματα χρήσης τους

Εντολές Δομής Προγράμματος

- Εντολή **Void setup()**

Αυτή η εντολή εκτελείται μόνο 1 φορά στην αρχή του προγράμματος και αφορά την αρχικοποίηση των μεταβλητών ή των ακίδων (pins) ως έξοδοι.

Παράδειγμα

```
int buttonPin = 3;

void setup()
       {
         Serial.begin(9600);
         pinMode(buttonPin, INPUT);
       }

void loop()
{
  // ...
}
```

- Εντολή **Void loop()**

Αυτή η εντολή μας καθορίζει τις εντολές που θα εκτελούνται συνεχώς και όσο ο Arduino είναι σε λειτουργία.

Παραδειγμα

```
int buttonPin = 3;

void setup()// setup initializes serial and the button pin

{
  beginSerial(9600);
  pinMode(buttonPin, INPUT);
}

void loop()// loop checks the button pin each time,
           // and will send serial if it is pressed

{
  if (digitalRead(buttonPin) == HIGH)
      serialWrite('H');
  else
      serialWrite('L');

  delay(1000);
}
```

Εντολές Ελέγχου Προγράμματος

- Η εντολή **if**

Αυτή η εντολή χρησιμοποιείται για να τον έλεγχο αλήθειας μιας αριθμητικής ή λογικής έκφρασης.

Παράδειγμα

```
if (someVariable > 50)
{
  // do something here
}
```

Το πρόγραμμα θα ελέγξει εάν κάποια μεταβλητή έχει τιμή πάνω από 50. Εάν η έκφραση μέσα στην παρένθεση είναι αληθής τότε το πρόγραμμα θα εκτελεστεί κανονικά εάν όχι δηλαδή, είναι ψευδής το πρόγραμμα παρακάμπτει το σώμα εντολών της if και εκτελεί τον επόμενο κώδικα. Ακολουθούν παραλλαγές του τρόπου εγγραφής της εντολής if:

```
if (x > 120) digitalWrite(LEDpin, HIGH);
if (x > 120){ digitalWrite(LEDpin, HIGH); }
if (x > 120){
               digitalWrite(LEDpin1, HIGH);
               digitalWrite(LEDpin2, HIGH);
             }
```

Σύμβολα σύγκρισης και εκφράσεις που μπορούμε να χρησιμοποιήσουμε παρουσιάζονται παρακάτω:

```
x == y (το x είναι ίσο με το y)
x != y (το x δεν είναι ίσο με το y)
x <  y (το x είναι μικρότερο από το y)
x >  y (το x είναι μεγαλύτερο από το y)
x <= y (το x είναι μικρότερο η ίσο με το y)
x >= y (το x μεγαλύτερο η ίσο με το y)
```

Προσοχή!

Σε περίπτωση που στο πρόγραμμα γράψετε (`x=10`) και όχι (`x==10`) τότε το πρόγραμμα καταχωρεί στην μεταβλητή `x` τον αριθμό `10` και έτσι δεν γίνεται η σύγκριση αν το `x` είναι ίσο με τον αριθμό `10`.

- Η εντολή **if/else**

Αυτή η εντολή χρησιμοποιείται όπως η εντολή `if` με την διαφορά ότι σε περίπτωση που η έκφραση ελέγχου `if` είναι ψευδής εκτελείται το σώμα εντολών που ακολουθεί την `else` και έπειτα συνεχίζεται κανονικά η ροή του προγράμματος.

Παράδειγμα

```
if (pinFiveInput < 500) {  // σώμα εντολών Α }
else                    {// σώμα εντολών Β   }
```

Άν η έκφραση ελέγχου είναι αληθής τότε εκτελείται το σώμα εντολών Α, αν οχι τότε εκτελείται με την εντολή else το σώμα εντολών Β. Μπορούμε επίσης να θέσουμε και πιο πολλά τεστ ταυτόχρονα χρησιμοποιώντας την εντολή else if.

Παράδειγμα

```
if (pinFiveInput < 500)         {  // do Thing A }
else if (pinFiveInput >= 1000) { // do Thing B  }
else                            { // do Thing C  }
```

Μπορούμε να χρησιμοποιεισουμε την εντολή else if οσες φορες θελουμε. Οποτε παρατηρούμε οτι μπορούμε να εκτελέσουμε απειρία τεστ σε μια εντολή εως οτου βρεθεί αληθής η συνθήκη μας.

- Η εντολή **for**

Η εντολη for χρησιμοποιειται για να κανει επαναληψη ενα block η μια εντολη η οποια βρισκεται αναμεσα σε {}.Καθε φορα που εκτελειται μια φορα η εντολη for χρησιμοποιειται στην αρχη μετα της επαναληψης μια μεταβλητη που τη αυξανει η την μειωνει καθε φορα εως οτου τερματισει το loop(block η την εντολη).

Παράδειγμα

```
// Dim an LED using a PWM pin
int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10

void setup() { // no setup needed }
void loop()  {
                for (int i=0; i <= 255; i++){
                   analogWrite(PWMpin, i);
                   delay(10);
                                          }
             }
```

Καθε φορα που πραγματοποιειται μια επαναληψη η συνθηκη i<=255 ελέγχεται και εαν ειναι αληθης τοτε η μεταβλητη μας i αυξανεται κατά ένα. Οταν η συνθηκη γινει ψευδεις τοτε η επαναληψη σταματαει.

Παράδειγμα

Πως μπορουμε επισης να κανουμε ενα LED να χανει το χρωμα του σιγα σιγα. (fade out)

```
void loop()
{
   int x = 1;
   for (int i = 0; i > -1; i = i + x){
                           analogWrite(PWMpin, i);
                           if (i == 255) x = -1; // switch direction at peak
                           delay(10);
                                       }
}
```

- Η εντολή **switch/case**

Είναι παρόμοια με τη εντολή if. Η switch(συγκρίνει την τιμή μιας μεταβλητής var με τις τιμές που ορίζονται στις δηλώσεις case και εκτελεί την ανάλογη. Μετά το τέλος κάθε εντολής case πρέπει να ακολουθεί η εντολή break.

Παράδειγμα

```
switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
```

- Η εντολή **while**

Η εντολή ελγέγχει τη συνθήκη μέσα στις () και αν είναι αληθής εκτελείται το σώμα εντολών που της αντιστοιχεί. Πρέπει επίσης να υπάρχει μια μεταβλητή που να αυξάνει κάθε φορά γιατί αλλιώς η εντολή while θα μας εγκλωβίσει σε ένα ατέρμονα βρόχο επανάλυψης.

Παράδειγμα

```
var = 0;
while(var < 200){ // do something repetitive 200 times
                  var++;
                 }
```

- Η εντολή **do-while**

Κάνει την ίδια λειτουργία με την εντολή while. Η μοναδική διαφορά τους είναι ότι η συνθήκη ελέγχεται στο τέλος της επανάληψης εξασφαλίζοντας έτσι ότι το σώμα εντολών θα εκτελεστεί τουλάχιστον μία φορά.

Παράδειγμα

```
Do    {
        delay(50);           // wait for sensors to stabilize
        x = readSensors();   // check the sensors
      } while (x < 100);
```

- Η εντολή **break**

Χρησιμοποιείται για να τερματίσει την επανάληψη εντολών όπως do, for, while και switch.

Παράδειγμα

```
for (x = 0; x < 255; x ++)
      {
       digitalWrite(PWMpin, x);
       sens = analogRead(sensorPin);
                 if (sens > threshold){      // bail out on sensor detect
                                       x = 0;
                                       break;
```

```
                                        }
        delay(50);
       }
```

- Η εντολή **continue**

Η εντολή αυτή προσπερνάει το υπόλοιπο της τρέχουσας μιας επανάληψης (do, for, while) και συνεχίζει ελέγχοντας τη συνθήκη έκφρασης της επανάληψης ώστε να γίνουν οι επόμενες επαναλήψεις.

Παράδειγμα

```
for (x = 0; x < 255; x ++)
{
    if (x > 40 && x < 120){      // create jump in values
        continue;
    }

    digitalWrite(PWMpin, x);
    delay(50);
}
```

- Η εντολή **return**

Τερματίζει την λειτουργία μίας συνάρτησης επιστρέφοντας την τιμή/μεταβλητή που την ακολουθεί.

Παράδειγμα

```
int checkSensor(){
                    if (analogRead(0) > 400) { return 1;}
                    else                     { return 0;}
                  }
```

- Η εντολή **#define**

Αυτή η εντολή σου δίνει το δικαίωμα να αντικαταστήσεις ένα όνομα με μια μεταβλητή κατά την διάρκεια της μεταγλώττισης του προγράμματος. Όταν χρησιμοποιούμε αυτήν την εντολή δεν βάζουμε στο τέλος ; ή = .

Παράδειγμα

```
#define ledPin 3
   // The compiler will replace any mention of ledPin with the value 3 at
   // compile time.
```

- Η εντολή **#include**

σου δίνει το δικαίωμα να εισάγεις νέες βιβλιοθήκες.

Λογικοί τελεστές

- Λογικό ΚΑΙ, **&&**

Μπορουν να χρησιμοποιηθουν σε μια εντολη που περιεχει το if.

Παράδειγμα

```
if (digitalRead(2) == HIGH  && digitalRead(3) == HIGH) { // σώμα εντολών }
```

Πρέπει να ισχύει και το $1^{ο}$ και το $2^{ο}$ μέρος της συνθήκης ώστε να εκτελεστεί το σώμα εντολών μέσα στα άγκιστρα.

- Λογικό Ή, **||**

Μπορούν να χρησιμοποιηθούν σε μια εντολή που περιείχε το if.

Παράδειγμα

```
if (x > 0 || y > 0) { // σώμα εντολών}
```

Για να εκτελεστεί το σώμα εντολών μέσα στα άγκιστρα, θα πρέπει η το $1^{ο}$ μερος της συνθήκης ελέγχου να είναι αληθές ή το$2^{ο}$ μέρος να είναι αληθές.

- Λογικό ΟΧΙ, **!**

Μπορούν να χρησιμοποιηθούν σε μια εντολή που περιείχε το if. Για να εκτελεστεί το σώμα εντολών μέσα στα άγκιστρα θα πρέπει η μεταβλητή ελέγχου x να ειναι ψευδης.

Παράδειγμα

```
if (!x) {   // σώμα εντολών }
```