ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΜΕΛΕΤΗ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ 10 ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΜΕ ΤΗ ΜΟΡΦΗ ΦΥΛΛΩΝ ΕΡΓΟΥ ΓΙΑ ΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ ARDUINO

Πτυχιακή Εργασία των

Ανδρέα Χριστοφιλίδη (Α.Ε.Μ. 812)

Χαράλαμπου Δεμενίδη (Α.Ε.Μ. 839)

Επιβλέπων: Μαδεμλής Ιωάννης, Εργαστηριακός Συνεργάτης

ΣΕΡΡΕΣ, 2013

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ:

Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχουμε αναφέρει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνουμε ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται τη μελέτη της αναπτυξιακής υπολογιστικής πλατφόρμας ARDUINO. Η πλατφόρμα ARDUINO στηρίζεται σε μικροελεγκτή Atmel και αποτελεί ένα ολοκληρωμένο αναπτυξιακό σύστημα ανοιχτού λογισμικού και ευέλικτης προσαρμογής υλικού (open-source hardware), προσφέροντας δυνατότητες υλοποίησης εφαρμογών σε πολλούς τομείς της ηλεκτρονικής. Στο πλαίσιο της πτυχιακής ορίστηκαν και υλοποιήθηκαν ένα σύνολο από παραδείγματα που αξιοποιούν και επιδεικνύουν τις δυνατότητες της πλατφόρμας. Περιγράφηκε αναλυτικά η αρχή λειτουργίας της, ο τρόπος διασύνδεσης με μία λίστα περιφερειακών και ο προγραμματισμός της σε γλώσσα wiring με το Ολοκληρωμένο Περιβάλλον Ανάπτυξης Arduino IDE. Τα προγράμματα συντάχθηκαν σε μορφή εργαστηριακών ασκήσεων.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Για την εργασία αυτή εργάστηκαν οι: Δεμενίδης Χαράλαμπος και Χριστοφιλίδης Ανδρέας υπό την επίβλεψη του κ. Ιωάννη Μαδεμλή. Χρησιμοποιήθηκε η ανοιχτού υλικού και ανοιχτού λογισμικού, αναπτυξιακή υπολογιστική πλατφόρμα ARDUINO Uno Revision 3 βασισμένη στον ATmega328P της ATMEL, έναν 8-bit AVR RISC-based 16MHz μικροελεγκτή και το Ολοκληρωμένο Περιβάλλον Ανάπτυξης λογισμικού Arduino IDE 1.0.2.

Σκοπός της πτυχιακής εργασίας είναι η μελέτη και η δημιουργία 10 εργαστηριακών ασκήσεων, με τη μορφή φύλλων έργου, για τον προγραμματισμό της πλατφόρμας Arduino. Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω ειδικών προγραμμάτων. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες, ωστόσο το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστούμε τον επιβλέποντα καθηγητή κ. Ιωάννη Μαδεμλή για την αγαστή συνεργασία και τη βοήθεια που παρείχε κατά τη διάρκεια συγγραφής του παρόντος πονήματος, καθώς και τους καθηγητές στο ΤΕΙ Σερρών για τις γνώσεις που μας παρείχαν και τα κίνητρα για τις δεξιότητες που αναπτύξαμε κατά τη διάρκεια της φοίτησής μας.

ΠΕΡΙΓΡΑΦΗ ΚΕΦΑΛΑΙΩΝ

Στα κεφάλαια της πτυχιακής εργασίας που ακολουθούν αναλύθηκαν παράμετροι όπως το υλικό που χρησιμοποιήθηκε, για παράδειγμα κάρτες και κυκλώματα, το λογισμικό και ο κώδικας αυτών και τέλος αναφέρθηκαν συμπεράσματα και παρατηρήσεις.

Πιο αναλυτικά, στο πρώτο κεφάλαιο αναλύθηκε το αναπτυξιακό σύστημα του Arduino Uno, μελετήθηκε η αρχιτεκτονική του, τα στοιχεία από τα οποία αποτελείται και ο τρόπος με τον οποίο λειτουργεί.

Στο δεύτερο κεφάλαιο μελετήθηκε και αναλύθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης και η γλώσσα προγραμματισμού του, στοιχεία απαραίτητα για τον προγραμματισμό του Arduino. Έγινε ανάλυση στις βασικές δομές της γλώσσας προγραμματισμού, τις διάφορες βιβλιοθήκες της πλατφόρμας Arduino. Τέλος το κεφάλαιο συμπληρώθηκε με την ενδελεχή ανάλυση ενός εισαγωγικού προγράμματος με το οποίο ο αναγνώστης καλείται να εξοικειωθεί.

Στο τρίτο κεφάλαιο έγινε η παρουσίαση και η επεξήγηση όλων των περιφερειακών εξαρτημάτων που χρησιμοποιούνται καθώς και του κώδικα προγραμματισμού τους. Η παρουσίαση έγινε με τη μορφή εργαστηριακών φύλλων έργου. Συγγράφηκαν 10 εργαστηριακές ασκήσεις, κάθε μία εκ των οποίων αναφέρθηκε στον τρόπο διασύνδεσης με το Arduino και τον προγραμματισμό μιας ομάδας περιφερειακών συσκευών. Συνολικά προγραμματίστηκαν 22 περιφερειακά και παρουσιάστηκαν 27 κώδικες. Σε κάθε περίπτωση υλοποιήθηκαν τα πρόσθετα κυκλώματα διασύνδεσης.

Πιο αναλυτικά τα επιμέρους περιφερειακά τα οποία μελετήθηκαν και προγραμματίστηκαν είναι:

1. Χρωματιστά LED.
2. Πολύχρωμη φωτοδίοδος RGB.
3. Οθόνη ενδείξεων 7-seg.
4. Βομβητής.
5. Μεγάφωνο.
6. Πιεζοηλεκτρικός δίσκος.
7. Κινητήρας συνεχούς ρεύματος.
8. SN754410 H-Γέφυρα.
9. Βηματικός κινητήρας.
10. ULN2003 Darlington ολοκληρωμένο.
11. Σερβοκινητήρας.
12. Οθόνη υγρών κρυστάλλων.
13. Οθόνη LED Dot Matrix.
14. Υπερηχητικός αισθητήρας.
15. Δέκτης υπερύθρων.

16. Ρολόι πραγματικού χρόνου.
17. Αριθμητικό πληκτρολόγιο.
18. SD κάρτα επέκτασης.
19. Φωτοαντίσταση.
20. Θερμίστορ.
21. Ηλεκτρονόμος.
22. 74HC595 καταχωρητής ολίσθησης.

Αμέσως μετά, παρατέθηκαν οι εντυπώσεις από την εκπόνηση της εργασίας και επισημάνθηκαν οι παρατηρήσεις και τα συμπεράσματα κατά τη διάρκεια διεκπεραίωσης της.

Τέλος, ακολουθούν τα ευρετήρια για τις εικόνες και τους πίνακες, οι βιβλιογραφικές αναφορές καθώς και τα παραρτήματα στα οποία υπάρχουν οι φωτογραφίες των κατασκευών μας, ο ενδεικτικός τιμοκατάλογος και τα φύλλα δεδομένων.

ΚΕΦΑΛΑΙΟ 1^{ο} : Η ΤΕΧΝΟΛΟΓΙΑ ΚΑΙ Η ΧΡΗΣΗ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ ARDUINO

1.1 Παρουσίαση των μικροελεγκτών

1.1.1 Τι είναι τα Ενσωματωμένα Συστήματα;

Με τον όρο «ενσωματωμένα συστήματα» εννοούμε την ενσωμάτωση κάποιου μικροελεγκτή / μικροεπεξεργαστή στη λειτουργία ενός ολόκληρου συστήματος με ηλεκτρονικά, μηχανολογικά και άλλα μέρη, αλλά μας αφορούν μόνο οι λειτουργίες του ελεγκτή και η διεπαφή του με το εξωτερικό περιβάλλον. Δηλαδή ενσωματωμένα συστήματα είναι αυτά στα οποία, κάποιος μικροελεγκτής λειτουργεί σαν μέρος μίας ολότητας, επιτελώντας συγκεκριμένο έργο και στον οποίο εν γένει ο χρήστης δεν έχει πρόσβαση για να αλλάξει το πρόγραμμα ή τη λειτουργικότητα του συστήματος.

Μικροεπεξεργαστής είναι η υλοποίηση ενός γενικής χρήσεως επεξεργαστή σε ένα ολοκληρωμένο κύκλωμα με κύριο κριτήριο την απόδοση. Δηλαδή στον μικροεπεξεργαστή δόθηκε έμφαση στην υπολογιστική ισχύ. Έτσι αν συνδυαστεί με τις κατάλληλες εξωτερικές περιφερειακές συσκευές μπορεί να πραγματοποιήσει μία πληθώρα γενικών εργασιών.

Σε αντίθεση ο μικροελεγκτής είναι σχεδιασμένος για πιο εξειδικευμένες εργασίες, έχει πολύ μικρότερες δυνατότητες συνεργασίας με τα εξωτερικά περιφερειακά, αφού υστερεί κατά πολύ σε υπολογιστική ισχύ. Στον σχεδιασμό των μικροελεγκτών δόθηκε περισσότερη έμφαση στο να απαιτούν πολύ μικρότερο αριθμό ολοκληρωμένων κυκλωμάτων για τη λειτουργία μίας συσκευής, το χαμηλό κοστολόγιο κατασκευής τους και τον εύκολο προγραμματισμό εξειδικευμένων εργασιών.

Μικροελεγκτής (microcontroller) είναι η υλοποίηση ενός γενικής χρήσεως επεξεργαστή σε ένα ολοκληρωμένο κύκλωμα με κύριο γνώμονα την αυτοτέλεια. Ουσιαστικά είναι μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Ένας μικροελεγκτής είναι συχνά ένα ενσωματωμένο τσιπ (ολοκληρωμένο κύκλωμα) και αποτελεί συνήθως μέρος ενός συστήματος. Περιλαμβάνει CPU, RAM, ROM, θύρες εισόδου/εξόδου και timers σαν έναν τυπικό υπολογιστή, αλλά επειδή τα επιμέρους συστατικά του είναι σχεδιασμένα να εκτελούν μόνο μία συγκεκριμένη εργασία για τον έλεγχο ενός απλού συστήματος, είναι πολύ μικρότερα και απλούστερα σχεδιασμένα, έτσι ώστε να μπορούν να περιλαμβάνουν όλες τις λειτουργίες που απαιτούνται σε ένα μόνο ολοκληρωμένο κύκλωμα.

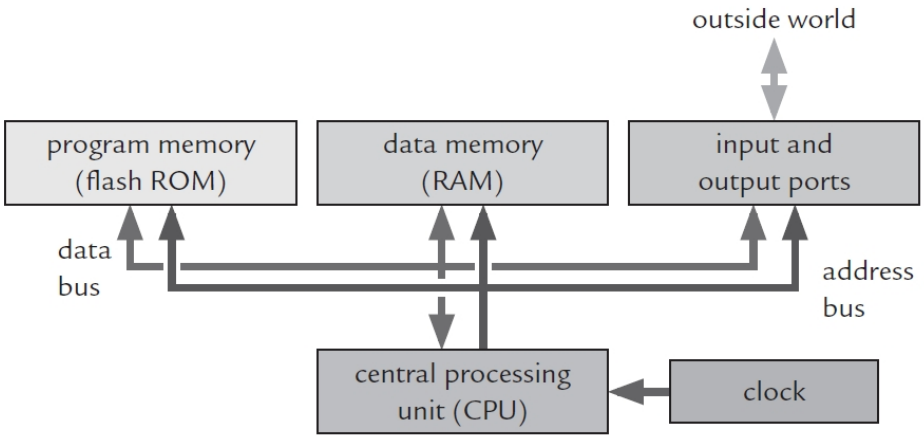
Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου, χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα. Επειδή σχεδόν όλοι οι μικροελεγκτές χρησιμοποιούνται σε ενσωματωμένες εφαρμογές, παρέχουν στήριξη σε επίπεδο υλικού αλλά και σε επίπεδο εντολών για πρωταρχικές συνήθως λειτουργίες όπως η μέτρηση χρόνου (με χρονιστές – timers), η μέτρηση συμβάντων (με μετρητές – counters), η σύγκριση

αναλογικού σήματος με κάποιο προγραμματιζόμενο κατώφλι (analog signal comparators), η υποστήριξη επικοινωνίας μέσω τυποποιημένου σειριακού πρωτοκόλλου (όπως RS 232, USB, κλπ.), καθώς και υποστήριξη δομών που αυξάνουν την αξιοπιστία (π.χ. με μετρητές επανεκκίνησης - watchdog timers).

1.1.2 Πλεονεκτήματα των μικροελεγκτών

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιώντας το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλές ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσης τους για τη σύνδεση εξωτερικών περιφερειακών συσκευών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.
- Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους συναντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (π.χ. οι σειρές από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου φον Νόιμαν.

1.1.3 Βασικά χαρακτηριστικά και συνήθη υποσυστήματα



Εικόνα 1: Ανατομία ενός μικροελεγκτή.

Τα βασικά χαρακτηριστικά ενός μικροελεγκτή είναι τα ακόλουθα:

- **Κεντρική μονάδα επεξεργασίας (CPU)**
  Βασικά της στοιχεία είναι:
  - Arithmetic Logic Unit (ALU), η οποία κάνει τους υπολογισμούς.
  - Καταχωρητές οι οποίοι χρειάζονται για την βασική λειτουργία της CPU, όπως είναι ο program counter (PC), ο stack pointer (SP), και ο status register (SR).
  - Επιπρόσθετους καταχωρητές, για την αποθήκευση των αποτελεσμάτων.
  - Instruction decoder και άλλες λογικές, για τον έλεγχο της CPU, των resets, των interrupts κτλ.
- **Μνήμη για το πρόγραμμα:** Non-volatile μνήμη (ROM), που σημαίνει ότι κρατά τα δεδομένα της όταν αποσυνδεθεί η τροφοδοσία.
- **Μνήμη για τα δεδομένα:** Γνωστή και ως μνήμη τυχαίας προσπέλασης (RAM) και συνήθως είναι volatile.
- **Θύρες εισόδου/εξόδου (I/O Ports):** Για την παροχή ψηφιακής επικοινωνίας με τον «έξω κόσμο».
- **Address και data buses:** Για τη διασύνδεση αυτών των υποσυστημάτων ώστε να μεταφράζονται τα δεδομένα και οι εντολές.
- **Ρολόι (Clock):** Για να κρατά χρονισμένο το όλο σύστημα. Μπορεί να παράγεται εσωτερικά ή να παρέχεται από ένα κρύσταλλο εξωτερικά.

Είναι μάλλον απίθανο κάποιος μικροελεγκτής να μη διαθέτει αυτά τα χαρακτηριστικά, αν και η υλοποίησή τους μπορεί να διαφέρει σημαντικά. Οι μεγάλες και ουσιαστικές διαφορές εμφανίζονται όταν πάμε στο πεδίο των περιφερειακών που εμπεριέχουν. Αρχικά αυτές οι λειτουργίες χρειάζονταν εντελώς ξεχωριστό εξοπλισμό αλλά με τη βελτίωση της τεχνολογίας και τα όλο μικρότερα μεγέθη μπόρεσαν να ενσωματωθούν στα chips των μικροελεγκτών. Πλέον τα περισσότερα περιφερειακά είναι στο ίδιο ολοκληρωμένο κύκλωμα με τον ελεγκτή. Παρακάτω αναφέρονται τα πιο **συνήθη περιφερειακά**.

Timers: Οι περισσότεροι μικροελεγκτές έχουν τουλάχιστον ένα timer λόγω της μεγάλης γκάμας λειτουργιών που παρέχουν. Ενδεικτικά:

- Ο χρόνος στον οποίο γίνονται οι μεταβάσεις σε μία είσοδο μπορούν να καταγραφούν.
- Οι έξοδοι μπορούν να γίνονται on ή off αυτόματα σε συγκεκριμένες συχνότητα. Αυτή ή λειτουργία χρησιμοποιείται πολύ συχνά για ένα PWM σήμα, το οποίο για παράδειγμα ελέγχει την φωτεινότητα ενός LED.
- Παρέχουν τη δυνατότητα χρήσης για προγραμματισμένες διαδικασίες, όπως είναι ο έλεγχος της θερμοκρασίας ενός δωματίου.

Watchdog timer: Είναι ένας timer ασφαλείας, ο οποίος κάνει reset στον ελεγκτή αν το πρόγραμμα εμπλακεί σε ατέρμονα βρόγχο.

Διεπαφές επικοινωνίας: Υπάρχει διαθέσιμη μεγάλη γκάμα τέτοιων διεπαφών για την ανταλλαγή πληροφοριών με κάποιο άλλο IC ή σύστημα. Κάποια από αυτά είναι:

- SPI (serial peripheral interface)
- Inter-integrated circuit (I2C ή απλά IIC)
- Ασύγχρονες διεπαφές, όπως RS-232, USB (universal serial bus), Ethernet, CAN (controller area network), κ.ά.

Analog-to-digital converter: Χρησιμοποιείται ευρέως αφού οι ποσότητες που διαβάζουμε από το εξωτερικό περιβάλλον μεταβάλλονται συνεχώς και μη-ψηφιακά.

Digital-to-analog converter: Χρησιμοποιούνται λιγότερο αφού μπορούμε να προσομοιάσουμε τις περισσότερες αναλογικές εξόδους χρησιμοποιώντας ένα PWM σήμα.

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Έτσι, υπάρχουν:

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως ROM-less. Αυτοί παρέχουν πάντοτε μια παράλληλη αρτηρία (bus) δεδομένων, πάνω στην οποία συνδέονται εξωτερικές μνήμες προγράμματος και RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM, η οποία κατασκευάζεται με το λογισμικό της (Mask ROM) ή γράφεται μόνο μια φορά (One Time Programmable, OTP). Παρέχουν τη δυνατότητα πολύ χαμηλού κόστους, όταν αγοράζονται σε πολύ μεγάλες ποσότητες.
- Μικροελεγκτές με μνήμη FLASH, οι οποία μπορούν συνήθως να προγραμματιστεί πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming, ISP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από το ειδικό τζαμάκι).

1.1.4 Πρόσθετες λειτουργίες

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (πχ I2C, SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικό-λογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC,ISDN, ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες

χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.

- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS-232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.
- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, π.χ. κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

1.1.5 Κατηγορίες μικροελεγκτών

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται οι εξής κυρίως κατηγορίες:

- **Μικροελεγκτές (4-bit, 8-bit (σύνηθες))** πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα για να μην μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως για παράδειγμα οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κ.α.).
- **Μικροελεγκτές (8-bit (σύνηθες) αλλά και 16 ή 32-bit)** χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα),

συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.

- **Μικροελεγκτές (κυρίως 32-bit)** μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Για παράδειγμα μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί φυσικά να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).
- **Μικροελεγκτές εξειδικευμένων εφαρμογών,** οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

1.1.6 Εργαλεία ανάπτυξης, γλώσσες προγραμματισμού μικροελεγκτών και κατασκευαστές

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), τη δυνατότητα προγραμματισμού της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές τα εργαλεία αυτά, δεν αποτελούνται ποτέ μόνο από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας μεταξύ τους. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες.

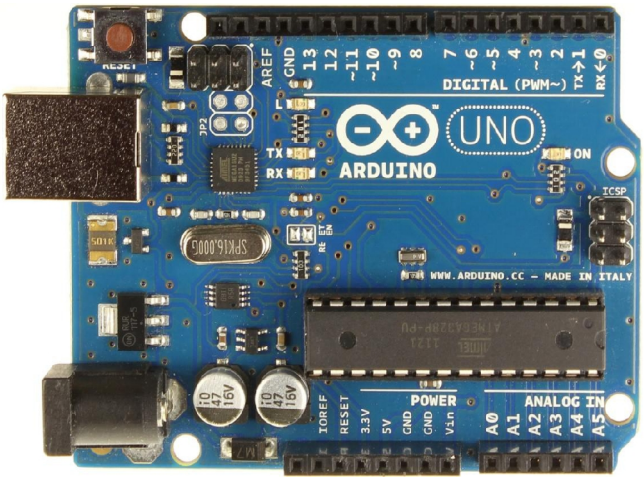
Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η **C**, η **C++** και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται μεγαλύτερη ταχύτητα ή μικρότερο μέγεθος χρησιμοποιούμενης μνήμης μπορεί να χρησιμοποιηθεί η **Assembly**. Όμως οι μεγαλύτερες δυνατότητες σε λειτουργικότητα και η ευκολία προγραμματισμού σε C -έναντι της assembly- σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την assembly από τις περισσότερες εφαρμογές.

Οι σημαντικότεροι κατασκευαστές μικροελεγκτών είναι η ARM η οποία δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα, η Atmel όπου και θα ασχοληθούμε εκτενέστερα στη συνέχεια, η Epson, η Freescale Semiconductor (πρώην Motorola), η Hitachi, η Maxim (μετά την εξαγορά της Dallas), η Microchip, η NEC, η Toshiba και η Texas Instrument.

1.2 Η Ιστορία του Arduino

Το 2005 στην Ίβρεα της Ιταλίας κατασκευάζεται μία συσκευή η οποία θα είχε την δυνατότητα να ελέγχει και να αλληλεπιδρά σύμφωνα με το περιβάλλον. Σκοπός των κατασκευαστών ήταν αυτή η συσκευή να κοστίζει λιγότερο σε σχέση με άλλες παρόμοιων δυνατοτήτων. Η ομάδα αποτελούνταν από τους Massimo Banzi, David Cuartielles, Tom Igoe, David Mellis και Gianluca Martino. Το όνομα της συσκευής έχει τις ρίζες του από τον Arduin of Ivrea, έναν βασιλιά της Ιταλίας του ενάτου αιώνα όπου κατοικούσε στην ίδια πόλη. Η συσκευή ονομάστηκε “Arduino” που αντιστοιχεί σε ένα ιταλικό ανδρικό όνομα και σημαίνει «ισχυρός φίλος».

Το arduino αναπτύχθηκε σύμφωνα με την πλατφόρμα Wiring, μία πτυχιακή εργασία του Hernando Barragan από το Interaction Design Institute Ivrea. Είχε ως στόχο να είναι μία ηλεκτρονική εκδοχή της Processing που θα χρησιμοποιούσε ένα περιβάλλον προγραμματισμού δικό της αλλά θα έμοιαζε σχεδιαστικά και συντακτικά με αυτό της Processing.



Εικόνα 2: Πλατφόρμα Arduino UNO REVision 3.

1.2.1 Τί είναι το arduino;

Όπως το περιγράφει ο δημιουργός του, το Arduino είναι μία open-source (ανοικτού κώδικα) πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών κυκλωμάτων βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software. Πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στο μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια, καθώς και το software που χρειάζεται για τη λειτουργία του, διανέμονται ελεύθερα και δωρεάν, ώστε να μπορεί να κατασκευαστεί από τον καθένα ο οποίος έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα. Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει το μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Το Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για τη δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Όμως το κύριο πλεονέκτημά

του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση.

Με δυο λόγια το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή πλακέτα που παρέχει μια σειρά από εισόδους (inputs) και εξόδους (outputs), η οποία υλοποιεί την γλώσσα προγραμματισμού Processing/Wiring που μας παρέχει την δυνατότητα εισαγωγής ρουτινών λειτουργίας. Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων αντικειμένων -διάδρασης του φυσικού με τον ψηφιακό χώρο- αλλά και συνδεδεμένων με λογισμικό, το οποίο τρέχει σε υπολογιστή.

1.2.2 Γιατί να επιλέξω το Arduino;

Υπάρχει πληθώρα άλλων μικροελεγκτών και αναπτυξιακών στο εμπόριο όπως ο Basic Stamp της Parallax, ο BX-24 της Netmedia, το Handyboard του MIT και πολλοί άλλοι όμοιας λειτουργικότητας. Όλα αυτά τα εργαλεία που προαναφέραμε είναι απλά για τον αρχάριο χρήστη καθώς "κρύβουν" τις δύσκολες λεπτομέρειες της αρχιτεκτονικής και επιτρέπουν τον άμεσο προγραμματισμό του μικροελεγκτή, προσφέροντας τα πάντα σε ένα "πακέτο" έτοιμο για χρήση. Το Arduino απλοποιεί τη χρήση ενός μικροελεγκτή προσφέροντας κάποια πλεονεκτήματα που το καθιστούν προσιτό για χρήση και είναι τα παρακάτω:

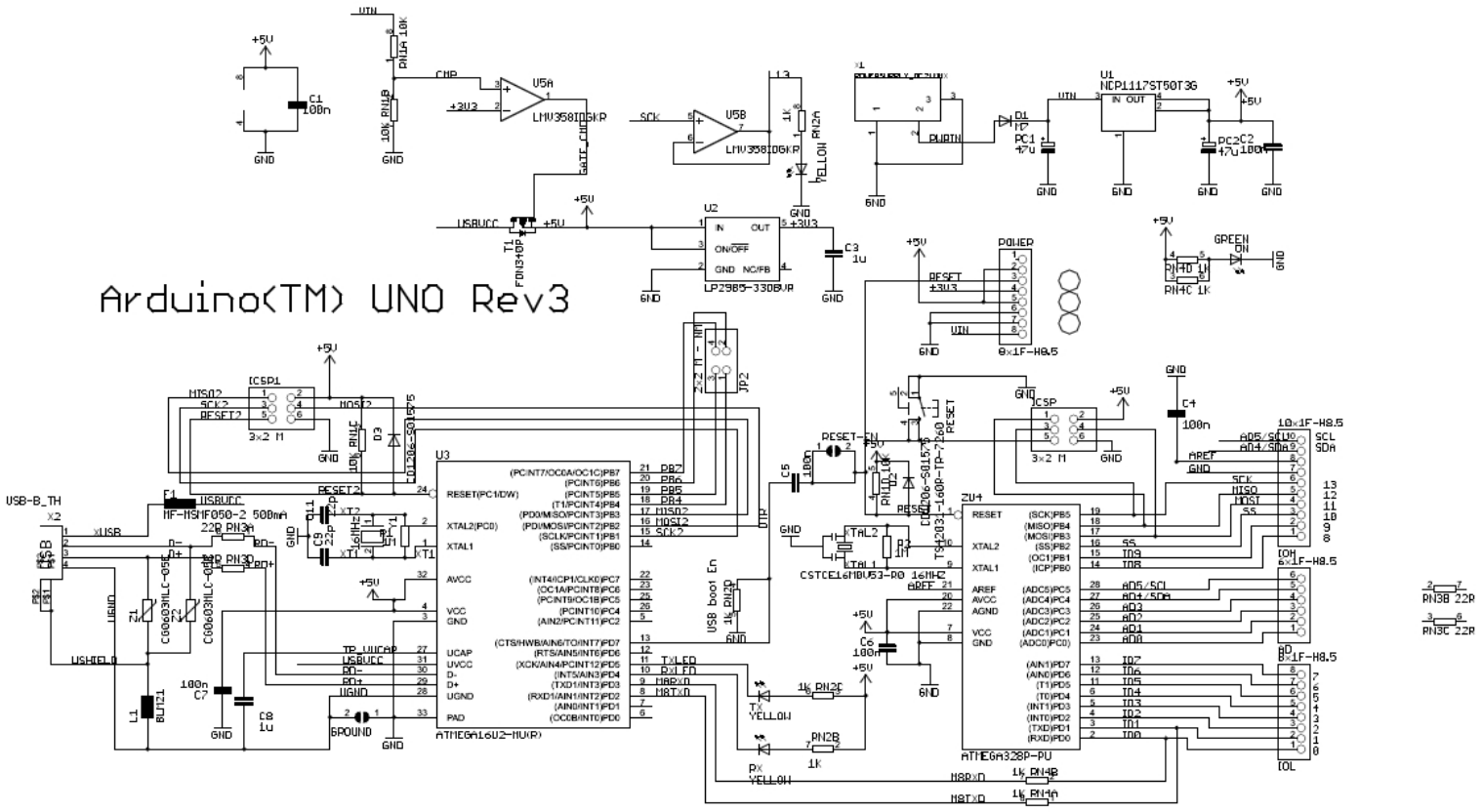
- **Φθηνός.** Οι πλακέτες του Arduino είναι εξαιρετικά φθηνές σε σχέση με άλλες πλατφόρμες μικροελεγκτών. Ειδικά δε, μπορεί με τα σχηματικά που κυκλοφορούν στο Internet να κατασκευάσει κάποιος την φθηνότερη εκδοχή ενός Arduino. Ωστόσο ακόμα και αν προμηθευτεί την έτοιμη (μονταρισμένη πλακέτα) αυτή θα ξεκινήσει από 10 Ευρώ.
- **Τρέχει σε διάφορα Λειτουργικά Συστήματα.** Οι μηχανικοί λογισμικού, ανέπτυξαν το περιβάλλον προγραμματισμού του Arduino για Windows, Machinstoh OSX και για λειτουργικά συστήματα Linux. Τα περισσότερα συστήματα ανάπτυξης Μικροελεγκτών περιορίζονται στα Windows.
- **Απλό, λιτό και απέριττο προγραμματιστικό περιβάλλον.** Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχάριους, αλλά είναι ταυτόχρονα ευέλικτο και για προχωρημένους χρήστες.
- **Ανοιχτού λογισμικού** που επεκτείνεται και παραμετροποιείται. Το software του Arduino διανέμεται με την μορφή εργαλείων ανοιχτού λογισμικού και είναι διαθέσιμο προς επέκταση από έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού του, μπορεί να επεκταθεί διαμέσου των βιβλιοθηκών της C++ και όποιος ενδιαφέρεται και θέλει να ασχοληθεί περισσότερο με τους μικροελεγκτές μπορεί να μεταβεί από το λογισμικό του Arduino στην AVR-C που προσφέρεται για προγραμματισμό των Atmel Μικροελεγκτών και είναι η γλώσσα στην οποία βασίστηκε το λογισμικό του Arduino. Ομοίως κάποιος μπορεί να προσθέσει κώδικα της AVR-C στο πρόγραμμα που έχει γράψει για το Arduino του.

- **Ανοιχτού Υλικού** ή αλλιώς ευέλικτης προσαρμογής υλικού (open-source hardware) το οποίο μπορεί να επεκταθεί. Ο Arduino βασίζεται στους μικροελεγκτές της Atmel. Τα σχηματικά για τα αναπτυξιακά είναι κάτω από την άδεια της Creative Commons, επιτρέποντας σε έμπειρους σχεδιαστές να κατασκευάσουν το δικό τους αναπτυξιακό, εξελίσσοντας το ήδη υπάρχον χωρίς να έχουν νομικά προβλήματα ή ακόμη καλύτερα χρήστες με μικρότερη εμπειρία μπορούν να επιδιώξουν την αντιγραφή και κατασκευή της πλακέτας σε ράστερ για να καταλάβουν την λειτουργία του.

1.3 Τα μέρη από τα οποία αποτελείται το Arduino

Το Arduino αποτελείται από τρία κύρια μέρη:

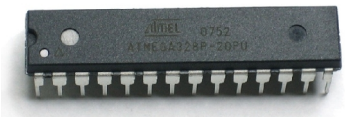
- Την πλακέτα Arduino που αποτελεί το hardware πάνω στο οποίο εργάζεται ο κάθε κατασκευαστής όταν πραγματοποιεί μια κατασκευή.
- Το Arduino IDE, το περιβάλλον του λογισμικού που τρέχει στον υπολογιστή. Το IDE χρησιμοποιείται για να δημιουργηθεί το sketch (ένα μικρό πρόγραμμα στον υπολογιστή) που φορτώνεται στον μικροελεγκτή.
- Μία φιλοσοφία και μία κοινότητα που του δίνουν πνοή. Στο site του Arduino (http://arduino.cc/) εργάζεται μια μεγάλη κοινότητα με πληροφορίες όσον αφορά τις εκδόσεις, την αγορά και τον προγραμματισμό της πλακέτας.



Σχήμα 1: Διάγραμμα Πλατφόρμας Arduino UNO REVision 3.

1.3.1 Μικροελεγκτής ATmega 328

Η δημοφιλέστερη και πιο καινούρια έκδοση του arduino είναι η UNO, που βασίζεται στο ολοκληρωμένο ATmega328P, έναν 8-bit RISC μικροελεγκτή, ο οποίος χρονίζει στα 16MHz. Είναι της εταιρείας ATMEL και είναι βασισμένος στην αρχιτεκτονική AVR. Η αρχιτεκτονική AVR βασίζεται σε μία τροποποίηση της αρχιτεκτονικής Harvard των 8 bit, RISC (Reduced Instruction Set Computing). Με βάση την αρχιτεκτονική Harvard το πρόγραμμα και τα δεδομένα είναι αποθηκευμένα σε διακριτά φυσικά συστήματα μνήμης τα οποία παρουσιάζονται σε διαφορετικές φυσικές διευθύνσεις μνήμης. Συνολικά διαθέτει 131 εντολές μήκους 16 ή 32 bit και 32 καταχωρητές των 8 bit.



Εικόνα 3: Μικροελεγκτής ATmega328.

1.3.1.1 Μνήμη του Μικροελεγκτή ATmega 328

Το ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- **Flash memory**

Η μνήμη flash έχει χωρητικότητα 32Kb, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware στην ορολογία του arduino ονομάζεται **bootloader** είναι αναγκαίο για την αναγνώριση και επικοινωνία του ATmega328 με το περιβάλλον IDE δηλαδή για την εγκατάσταση των προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών των **προγραμμάτων του χρήστη**, αφού πρώτα μεταγλωττιστούν από τον compiler στον υπολογιστή. Η μνήμη Flash δεν χάνει τα περιεχόμενά της με την απώλεια της τροφοδοσίας ή κάνοντας reset το μικροελεγκτή. Επίσης, ενώ η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime, μέσα από τα προγράμματα λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), έχει σχεδιαστεί μία βιβλιοθήκη που επιτρέπει την χρήση runtime στον χώρο που περισσεύει από την αποθήκευση των sketch (30Kb μείον το μέγεθος του προγράμματος σε μεταγλωττισμένη μορφή).

- **SRAM memory**

Η μνήμη SRAM (static random access memory) είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν **μεταβλητές, πίνακες** κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη είναι πτητική, χάνει τα δεδομένα της όταν η παροχή ρεύματος στο arduino σταματήσει ή αν γίνει reset. Στο ATmega328 η SRAM μνήμη καταλαμβάνει χώρο 2048 bytes κατά την διάρκεια μίας κανονικής λειτουργίας και όλες οι μεταβλητές φορτώνονται σε αυτή καθ' όλη την διάρκεια της λειτουργίας του microcontroller.

- **EEPROM memory**

Το τελευταίο μέρος της μνήμης είναι η EEPROM και καταλαμβάνει 1024 bytes, αρκετά μικρή για μνήμη που χρησιμοποιείται μόνο για ανάγνωση (**read-only**), για την αποθήκευση μακροχρόνιων πληροφοριών. Η EEPROM έχει όριο ζωής καθώς δε μπορεί να επαναπρογραμματιστεί για περισσότερες από 100.000 φορές. Είναι byte addressable μνήμη, γεγονός που καθιστά λίγο δυσκολότερο να τεθεί σε χρήση αφού απαιτείται ειδική βιβλιοθήκη ώστε να μπορέσει κάποιος να έχει πρόσβαση σε αυτή.

1.3.2 FTDI ολοκληρωμένο

Εκτός όμως από το ATMega328 το arduino χρησιμοποιεί και ένα FDTI ολοκληρωμένο κύκλωμα (τσιπ). Οι μικροελεγκτές ATMega προγραμματίζονται χρησιμοποιώντας σειριακή επικοινωνία με τους υπολογιστές, έτσι το FDTI αναλαμβάνει την εργασία της μετατροπής της σειριακής θύρας σε USB. Η φόρτωση του sketch πραγματοποιείται μέσω της USB θύρας που διαθέτει η πλακέτα arduino. Έτσι οι πληροφορίες που προέρχονται από την USB θύρα του υπολογιστή εισέρχονται στην USB θύρα του arduino και στην συνέχεια οδηγούνται στο FDTI ολοκληρωμένο για να διαμορφωθούν σε κατάλληλη μορφή ώστε ο μικροελεγκτής να μπορέσει να τις διαβάσει.

1.3.3 Σειριακή επικοινωνία

Το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, η οποία προωθείται μέσα από τον 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.

1.3.4 Θύρες (PINS) πλατφόρμας Arduino

Η πλακέτα arduino UNO διαθέτει:

- **14 Ψηφιακές I/O θύρες (εισόδου & εξόδου)**

Σύμφωνα με το πρόγραμμα που θα φορτωθεί στον μικροελεγκτή αυτές οι θύρες μπορούν να εργαστούν ως είσοδοι ή έξοδοι ψηφιακών σημάτων. Λειτουργούν στα 5 Volts και έχουν την δυνατότητα να παρέχουν ή να καταβυθίζουν ένταση ρεύματος της τάξεως των 40mA (προτεινόμενο 20mA). Σε κάθε θύρα υπάρχει εσωτερικά ένας αντιστάτης στα 20-50ΚΩ (αποσυνδεδεμένος εξ' ορισμού) που ενεργοποιείται από το λογισμικό.

Οι ψηφιακές θύρες **3, 5, 6, 9, 10 και 11** μπορούν να λειτουργήσουν και ως ψευδοαναλογικές θύρες εξόδου με το σύστημα **PWM** (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των

ανεμιστήρων. Παρέχεται οχτάμπιτη (8bit) παραγωγή PWM, το οποίο παίρνει ένα εύρος τιμών από το 0 έως το 255. Δεν είναι πραγματικά αναλογικό σύστημα, έτσι θέτοντας στην έξοδο την τιμή 127, δεν σημαίνει ότι η έξοδος θα παρέχει 2.5V αντί της κανονικής τιμής των 5V, αλλά ότι θα δίνει έναν παλμό που η τάση του θα εναλλάσσεται με μεγάλη συχνότητα και για ίσα χρονικά διαστήματα μεταξύ των τιμών 0V και 5V με σκοπό η μέση τιμή να ισούται με 2,5V. Είναι μια τεχνική που ακολουθείται για την παραγωγή αναλογικών σημάτων με ψηφιακά μέσα.

Οι θύρες **0 και 1** (σειριακή επικοινωνία) χρησιμοποιούνται επίσης για να λαμβάνουν (**RX**) και να μεταδίδουν (**TX**) TTL **σειριακά δεδομένα**. Έτσι, όταν για παράδειγμα το πρόγραμμα στέλνει δεδομένα σειριακά, τότε αυτά προωθούνται στην θύρα USB μέσω του ελεγκτή Serial-Over-USB όπως επίσης και στο pin 0 για να τα διαβάσει ενδεχομένως μία άλλη συσκευή (π.χ. ένα δεύτερο arduino στη δική του θύρα 1). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμα ενεργοποιηθεί το σειριακό interface, καταλαμβάνονται δύο ψηφιακές θύρες εισόδου/εξόδου.

Οι θύρες **2 και 3** λειτουργούν και ως **εξωτερικά interrupts** (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορούν να ρυθμιστούν μέσα από το πρόγραμμα ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές τάσης, η κανονική ροή του προγράμματος να σταματάει άμεσα και να εκτελείται μία συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

Οι θύρες **10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)** μπορούν να υποστηρίξουν και επικοινωνία **SPI**.

- **6 Αναλογικές I/O θύρες (εισόδου & εξόδου)**

Οι αναλογικές θύρες είναι αριθμημένες από το 0 έως το 5. Η κάθε μία από αυτές λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter). Για παράδειγμα, αν τροφοδοτηθεί ένα από αυτά τα pin, με μία τάση η οποία μπορεί να κυμανθεί με ένα ποτενσιόμετρο από 0V ως μία τάση αναφοράς Vref (η οποία αν δεν γίνει κάποια αλλαγή είναι προ-ρυθμισμένη στα 5V), τότε μέσα από το πρόγραμμα μπορεί να «διαβαστεί» η τιμή της θύρας ως ένας ακέραιος αριθμός χωρητικότητας 10-bit, από το 0 (όταν η τάση στο pin είναι 0V) μέχρι το 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μία εντολή μέσω software.

Υποστήριξη **I2C/TWI** επικοινωνίας με χρήση της Wire Library γίνεται με τους ακροδέκτες: **A4** (SDA) (data line), **A5** (SCL) (clock line).

Η εξ' ορισμού τάση αναφοράς είναι τα 5V. Η τάση αναφοράς μπορεί να αλλάξει χρησιμοποιώντας τον ακροδέκτη AREF. Έτσι, τροφοδοτώντας με μία εξωτερική τάση αναφοράς (π.χ. 3.3V) τη θύρα με την σήμανση **AREF** και στη συνέχεια εκτελεστεί η εντολή να διαβαστεί κάποιο pin αναλογικής εισόδου στο οποίο εφαρμόζεται τάση (π.χ. 1.65V), το Arduino θα επιστρέψει την ανάλογη τιμή (512).

1.3.5 Θύρες τροφοδοσίας

Η πλακέτα μπορεί να τροφοδοτηθεί μέσω σύνδεσης USB ή με εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm (θετικός πόλος στο κέντρο). Η πηγή επιλέγεται αυτόματα. Η εξωτερική πηγή μπορεί να είναι είτε AC/DC τροφοδοτικό, είτε μπαταρίες. Η εξωτερική τάση μπορεί να κυμαίνεται από 6 ως 20V. Αν είναι μικρότερη από 7V, ο ακροδέκτης των 5V μπορεί να παρέχει λιγότερη τάση από την ονομαστική και έτσι να προκληθούν αστάθειες. Αν χρησιμοποιηθούν περισσότερα από 12V, ο σταθεροποιητής τάσης μπορεί να υπερθερμανθεί και να καταστραφεί η πλακέτα. Το ενδεδειγμένο είναι από 7 ως 12V. Οι ακροδέκτες τροφοδοσίας βρίσκονται δίπλα από τις θύρες αναλογικής εισόδου. Εκεί υπάρχει μία ακόμα συστοιχία από 6 ακροδέκτες με την σήμανση **POWER**. Η λειτουργία του καθενός ακροδέκτη έχει ως εξής:

- Ο πρώτος, με την ένδειξη **RESET**, όταν γειωθεί (με οποιοδήποτε από τα 3 pin με την ένδειξη GND που υπάρχουν στο arduino) έχει ως αποτέλεσμα την επανεκκίνηση του arduino.
- Ο δεύτερος με την ένδειξη **3.3V**, μπορεί να τροφοδοτήσει διατάξεις, συσκευές ή αισθητήρες 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.
- Η τρίτη θύρα με την ένδειξη **5V**, μπορεί να χρησιμοποιηθεί και αυτή για την τροφοδότηση διαφόρων εξαρτημάτων, συσκευών ή αισθητήρων 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 παρέχει τάση 5V), είτε από την εξωτερική τροφοδοσία μέσω του VIN αφού αυτή περάσει από το ρυθμιστή τάσης για να την «σταθεροποιήσει» στα 5V.
- Ο τέταρτος και ο πέμπτος ακροδέκτης με την ένδειξη **GND** είναι οι γειώσεις.
- Ο έκτος και τελευταίος ακροδέκτης, 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 βολεύει να χρησιμοποιηθεί η υποδοχή του φις των 2.1mm. Αν όμως υπάρχει ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, τότε μπορεί να χρησιμοποιηθεί αυτό το pin για να τροφοδοτήσει εξαρτήματα και συσκευές με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

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 (**ON**) 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** με τη χρήση του οποίου γίνεται η βασική δοκιμή λειτουργίας του Arduino. Αυτό μπορεί να γίνει από την πρώτη στιγμή, χωρίς να συνδεθεί τίποτα πάνω στο Arduino αφού οι κατασκευαστές του σκέφτηκαν να το ενσωματώσουν στην πλακέτα πάνω στη ψηφιακή θύρα **13**. Ανάβει αναθέτοντάς του μέσα από το πρόγραμμά την τιμή HIGH.

1.4 Προστασία από υπερτάση

Το arduino διαθέτει μια ασφάλεια που προκαλεί reset και προστατεύει τις θύρες USB του Η/Υ από τα βραχυκυκλώματα και τις υψηλές τιμές τάσεις. Αν και οι περισσότεροι υπολογιστές παρέχουν τέτοια προστασία εσωτερικά, η ασφάλεια αυτή παρέχει ένα επιπλέον επίπεδο προστασίας. Σε περίπτωση που περισσότερο από 500mA εφαρμοστούν στη θύρα USB, η ασφάλεια θα διακόψει αυτόματα τη σύνδεση μέχρι το βραχυκύκλωμα ή η υπερτάση να σταματήσει.

1.5 Άλλες εκδόσεις Arduino

Σήμερα εκτός από την έκδοση arduino UNO υπάρχουν άλλες 15 επίσημες διαφορετικές εκδόσεις arduino. Μερικές από αυτές είναι:

- Nano: Είναι μία μικρότερη έκδοση του arduino η οποία συνδέεται στον υπολογιστή μέσω καλωδίου mini USB B.
- Bluetooth: Ο ελεγκτής arduino BT περιέχει μία Bluetooth πλακέτα η οποία επιτρέπει την ασύρματη επικοινωνία και προγραμματισμό του μέσω του υπολογιστή.
- LilyPad: Αυτή η έκδοση είναι σχεδιασμένη για να χρησιμοποιείται στα ρούχα. Έχει μοβ χρώμα και μπορεί να ραφτεί εύκολα πάνω σε ύφασμα.
- Pro: Η συγκεκριμένη πλακέτα είναι σχεδιασμένη για προχωρημένους χρήστες που έχουν σκοπό να τη χρησιμοποιήσουν κάπου μόνιμα. Είναι φθηνότερη από την UNO και συνδέεται εύκολα με μπαταρίες αλλά απαιτούνται επιπλέον ηλεκτρονικές διατάξεις για την χρήση της.
- Pro-mini: Όπως και στην Pro έκδοση έτσι και σε αυτήν σχεδιάστηκε για προχωρημένους χρήστες με την διαφορά ότι η πλακέτα καταλαμβάνει μικρότερο χώρο. Η Pro-mini είναι και αυτή φθηνότερη έκδοση αλλά χρειάζεται επίσης περαιτέρω εργασία για την χρησιμοποίηση της.
- Serial: Αυτή είναι η βασική έκδοση arduino που χρησιμοποιεί το πρωτόκολλο RS232 για την επικοινωνία και τον προγραμματισμό του. Το πλεονέκτημα της είναι ότι μπορεί εύκολα να κατασκευαστεί από έναν χρήστη.
- Serial Single Sided: Η έκδοση αυτή σχεδιάστηκε με σκοπό να κατασκευαστεί στο χέρι. Είναι λίγο μεγαλύτερο από τα προηγούμενα arduino, παρ' όλα αυτά παραμένει συμβατή με τις περισσότερες κατασκευές που σχεδιάστηκαν για να προεκτείνουν της δυνατότητες της UNO.

- Mega: Το Arduino mega όπως και το UNO, συνδέεται μέσω USB και έχει μεγαλύτερο μέγεθος. Οι διαφορές τους είναι ότι διαθέτει μεγαλύτερη μνήμη και έχει περισσότερες θύρες εισόδου/εξόδου.
- Leonardo, Fio, Esplora, Ethernet, Micro, Mini, Due κλπ.

Εκτός όμως από το επίσημο arduino υπάρχουν και πολλοί άλλοι κλώνοι που λειτουργούν με παρόμοιο τρόπο (Compatible Arduino) όπως το: Freeduino, Roboduino, AVR.duino, Sanguino, LEDuino, Miduino κ.λ.π.

1.6 Arduino Shields (κάρτες επέκτασης)

Πέραν όμως της μεγάλης ποικιλίας των εκδόσεων arduino, υπάρχει και μεγάλη ποικιλία από πλακέτες οι οποίες μπορούν να “κουμπώσουν” και να συνδεθούν με την πλατφόρμα arduino, με σκοπό την προέκταση των δυνατοτήτων της. Κάποιες από αυτές είναι:

- Motor Controller: Οι ελεγκτές κινητήρων είναι πλακέτες οι οποίες χρησιμοποιούνται με σκοπό τον έλεγχο των κινητήρων. Εξαιτίας του ότι η πλακέτα arduino δεν έχει την δυνατότητα να τροφοδοτήσει με την απαιτούμενη ισχύ τους κινητήρες, παρουσιάστηκε η ανάγκη κατασκευής κυκλώματος όπου θα «οδηγεί» τους κινητήρες αυτούς.
- XBee Shield: Το XBee είναι επέκταση η οποία επιτρέπει σε ένα arduino να επικοινωνήσει ασύρματα με έναν υπολογιστή (με άλλο arduino ή τηλέφωνο android) σε απόσταση έως και 100 μέτρων. Στην πραγματικότητα η ασύρματη αυτή επικοινωνία επιτυγχάνεται από δύο πομποδέκτες. Ο κάθε πομποδέκτης αποτελείται από μία πλακέτα XBee Explorer USB η οποία είναι ο μετατροπέας της USB θύρας σε σειριακή και ένα XBee Antenna. Το XBee Antenna είναι υπεύθυνο για να εκπέμπει και να λαμβάνει σειριακά ηλεκτρικά σήματα τα οποία έχουν διαμορφωθεί σε ηλεκτρομαγνητικά στην συχνότητα των 2.4GHz.
- Voice Recognition Shield: Είναι μία Shield η οποία μαζί με το κατάλληλο software έχει την δυνατότητα να αναγνωρίσει μία ποικιλία φωνητικών εντολών που δίνονται από κάποιον χρήστη και να τις προωθήσει για να πραγματοποιήσει το arduino συγκεκριμένες ενέργειες.
- LCD Shield: Το arduino σε συνδυασμό με μία LCD shield έχει την δυνατότητα να εμφανίσει διάφορα μενού ή μηνύματα σε μία οθόνη. Για παράδειγμα, οι χρήστες μπορούν μέσω της LCD να ενημερώνονται για τα αποτελέσματα που λαμβάνει ένα arduino από τους αισθητήρες που είναι συνδεμένοι. Έτσι δε χρειάζεται να συνδεθεί το arduino με τον υπολογιστή για να διαβαστούν τα αποτελέσματα από το serial monitor.
- GPS Shield: Η συγκεκριμένη πλακέτα επικοινωνεί με τουλάχιστον 3 δορυφόρους και επιστρέφει στο arduino έναν αριθμό μεταβλητών που αντιστοιχούν σε συντεταγμένες.
- Wave Shield: Είναι και αυτή μία πολύ ενδιαφέρουσα κατασκευή που δίνει την δυνατότητα στο arduino να αναπαράγει μουσικά αρχεία μορφοποιημένα σε WAVE (.wav).

Τα shields είναι σχεδιασμένα ώστε αφού κουμπωθούν πάνω στο Arduino να προωθούν τις υποδοχές του, ώστε να μπορούν να συνδεθούν επιπλέον εξαρτήματα ή shields. Το κάθε shield χρησιμοποιεί ορισμένους από τους πόρους συνδεσιμότητας του Arduino και έτσι δεν μπορούν να συνδεθούν απεριόριστα shields. Μάλιστα κάποια shields μπορεί να μην είναι συμβατά μεταξύ τους γιατί χρησιμοποιούν τα ίδια pin του Arduino για επικοινωνία με αυτό. Επίσης, επειδή κάποια από αυτά δεν προωθούν τις συνδέσεις του Arduino (όπως π.χ. οι οθόνες οι οποίες δεν έχουν νόημα αν καλυφτούν από πάνω με ένα επόμενο shield), υπάρχουν ειδικά extender shields που κουμπώνουν στο Arduino και δίνουν την δυνατότητα σε δύο άλλα shields να κουμπώσουν πάνω τους, λειτουργώντας σαν πολύπριζα.

Όπως και για το ίδιο το Arduino, το βασικό πλεονέκτημα των shields δεν είναι τόσο το προφανές πλεονέκτημα του έτοιμου hardware, όσο το ότι συνοδεύονται συνήθως από έτοιμες βιβλιοθήκες που επιτρέπουν τον προγραμματισμό των sketch σε high level γλώσσα.

Η λίστα των arduino shields είναι πραγματικά πολύ μεγάλη και μπορεί να καλύψει τις πιο απαιτητικές ανάγκες των χρηστών για την πραγματοποίηση νέων ιδεών. Παρόλα αυτά συνεχίζεται η ανάπτυξή τους με γοργό ρυθμό.

1.7 Το software για τον προγραμματισμό του Arduino

Το arduino IDE (Integrated Development Environment - Ολοκληρωμένο Περιβάλλον Ανάπτυξης) είναι πρόγραμμα βασισμένο σε Java που εκτελείται στον υπολογιστή και επιτρέπει να δημιουργηθούν τα sketches που θα φορτωθούν στην πλατφόρμα arduino.

Η φόρτωση των προγραμμάτων πραγματοποιείται μέσω της USB σύνδεσης. Έτσι οι πληροφορίες που προέρχονται από την USB θύρα του υπολογιστή εισέρχονται στην USB θύρα του arduino και στην συνέχεια οδηγούνται στο FDTI ολοκληρωμένο για να διαμορφωθούν σε κατάλληλη μορφή ώστε ο μικροελεγκτής να μπορέσει να τις διαβάσει.

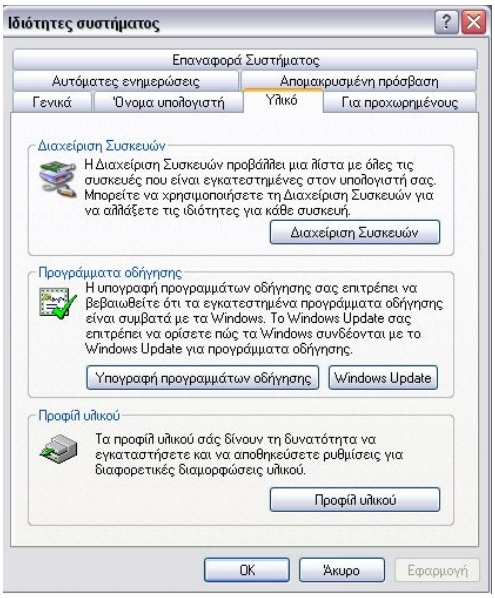
Τα βήματα με τα οποία προγραμματίζεται το arduino UNO είναι:

1. Συνδέεται η πλακέτα με τον υπολογιστή μέσω της USB θύρας.
2. Γράφεται ο κώδικας του sketch ο οποίος θα προγραμματίσει τον μικροελεγκτή ώστε να εκτελέσει τις επιθυμητές εργασίες.
3. Απαιτούνται μερικά δευτερόλεπτα έως ότου να τελειώσει ο έλεγχος (debugging) και η μεταγλώττιση (compile) του προγράμματος.
4. Φορτώνεται το εκτελέσιμο στη μνήμη του μικροελεγκτή μέσω της USB θύρας. Στη συνέχεια το arduino επανεκκινεί (αυτόματα) ώστε να διαβάσει τον καινούριο κώδικα.
5. Ο μικροελεγκτής εκτελεί τις εντολές του κώδικα που του αναθέσαμε.

1.7.1 Εγκατάσταση του Arduino IDE

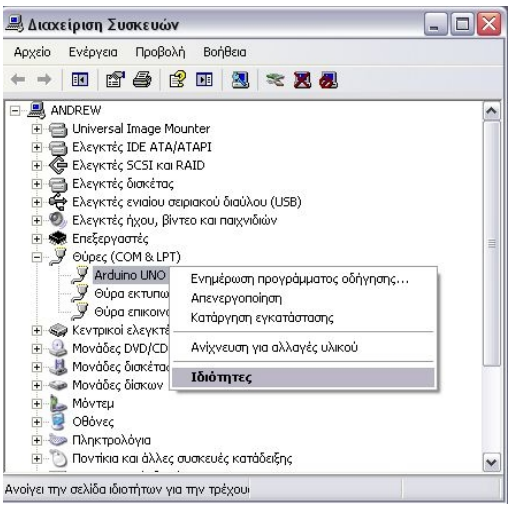
Για την Εγκατάσταση του arduino IDE (Integrated Development Environment - Ολοκληρωμένο Περιβάλλον Ανάπτυξης) οδηγούμαστε στο site του Arduino http://arduino.cc/en/Main/Software, από όπου θα κάνουμε λήψη του επίσημου λογισμικού που απαιτείται για να λειτουργήσει η συσκευή. Έχουμε τη δυνατότητα εγκατάστασης σε 3 διαφορετικά λειτουργικά συστήματα Windows, Mac OS X και Linux. Η διάθεση του παρέχεται δωρεάν από την εταιρεία παραγωγής.

Κατεβάζοντας το αρχείο παρατηρούμε ότι δεν υπάρχει το συνηθισμένο αρχείο εγκατάστασης setup.exe. Απλά αποσυμπιέζουμε το φάκελο και είμαστε έτοιμοι να ξεκινήσουμε. Πριν όμως τρέξουμε τον πρώτο μας κώδικα απαιτείται μια μικρή παραμετροποίηση από την διαχείριση συσκευών. Αρχικά, θα πάμε με δεξί κλικ στις «Ιδιότητες» του «Ο υπολογιστής μου». Από το παράθυρο που θα εμφανιστεί πηγαίνουμε στην καρτέλα «Υλικό» και επιλέγουμε «Διαχείριση συσκευών» όπως φαίνεται στην Εικόνα.

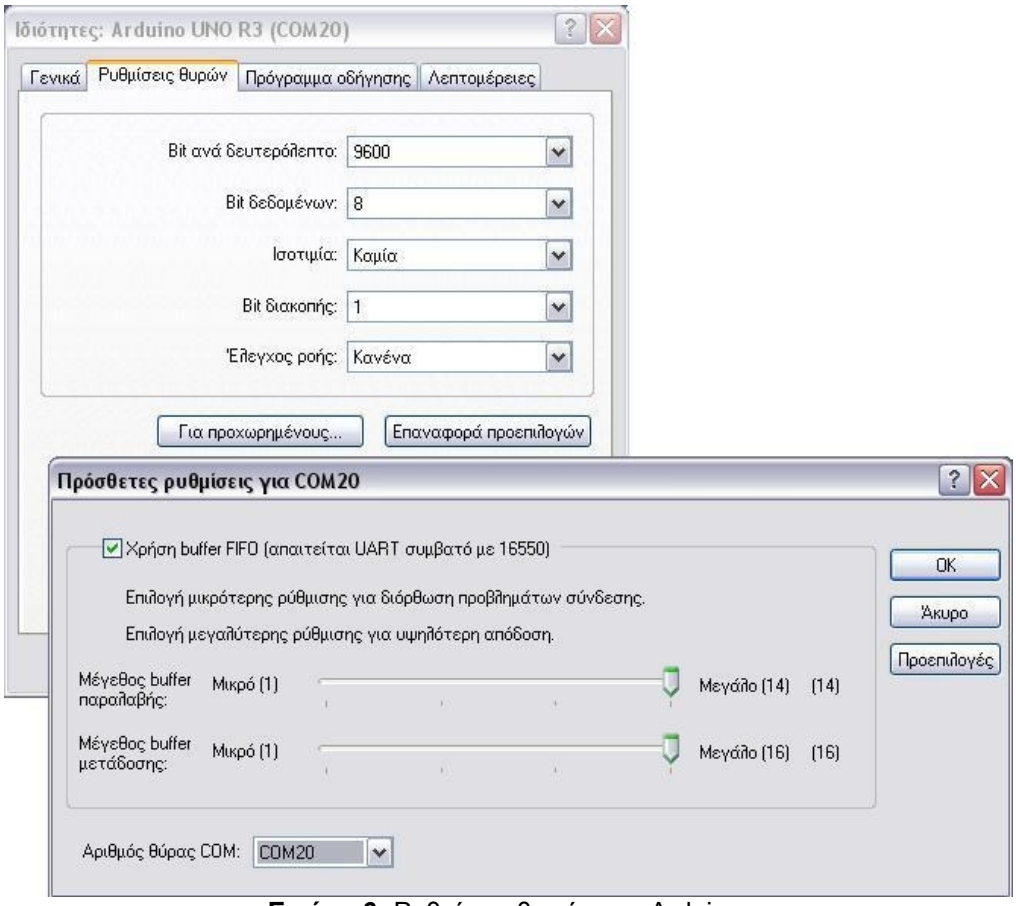


Εικόνα 4: Ο Υπολογιστής μου -> Ιδιότητες -> Υλικό

Οι θύρες COM1 και COM2 είναι κατειλημμένες από το σύστημα. Συνηθισμένες θύρες κατά την εγκατάσταση είναι οι COM3 και COM4. Στην περίπτωσή μας ο Arduino βρίσκεται στην COM20 θύρα. Στη συνέχεια με δεξί κλικ μπαίνουμε στις «Ιδιότητες» του Arduino για να δούμε τις ρυθμίσεις των θυρών. Οι τιμές των ρυθμίσεων που θα πρέπει να έχουμε φαίνονται στην Εικόνα.

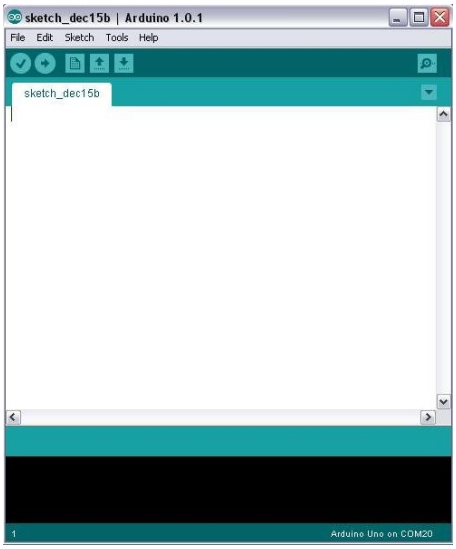


Εικόνα 5: Διαχείριση συσκευών



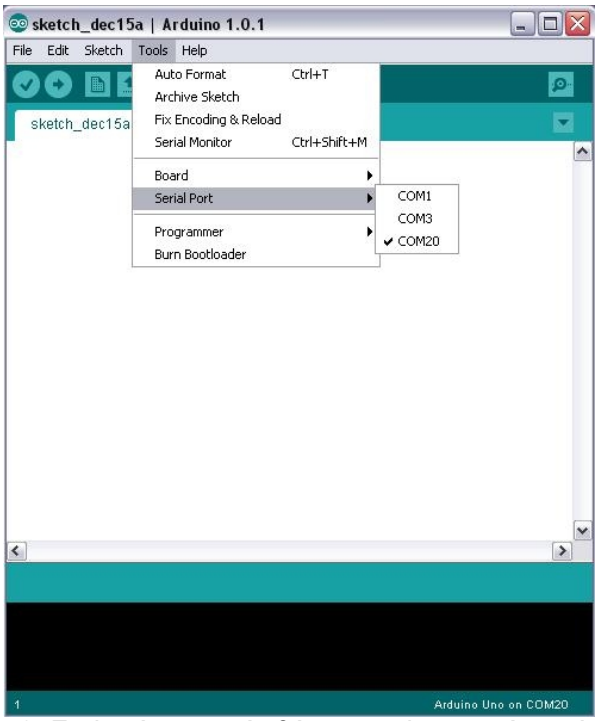
Εικόνα 6: Ρυθμίσεις θυρών του Arduino.

Μετά την παραμετροποίηση της συσκευής μας, τρέχουμε το **Περιβάλλον Ανάπτυξης** arduino.exe.



Εικόνα 7: Ο Compiler του κώδικα.

Πριν περάσουμε το πρώτο μας πρόγραμμα στον arduino πρέπει να κάνουμε παραμετροποίηση του προγράμματος ώστε να επικοινωνήσει στην σωστή θύρα που έχει εγκατασταθεί. Πηγαίνουμε λοιπόν στην επιλογή «Tools», από τη μπάρα εργασιών και στη συνέχεια διαλέγουμε τη σειριακή θύρα που αντιστοιχεί στη συσκευή μας, δηλαδή εδώ επιλέγουμε COM20 όπως φαίνεται και στην Εικόνα.



Εικόνα 8: Επιλογή σειριακής θύρας στο λογισμικό του Arduino.

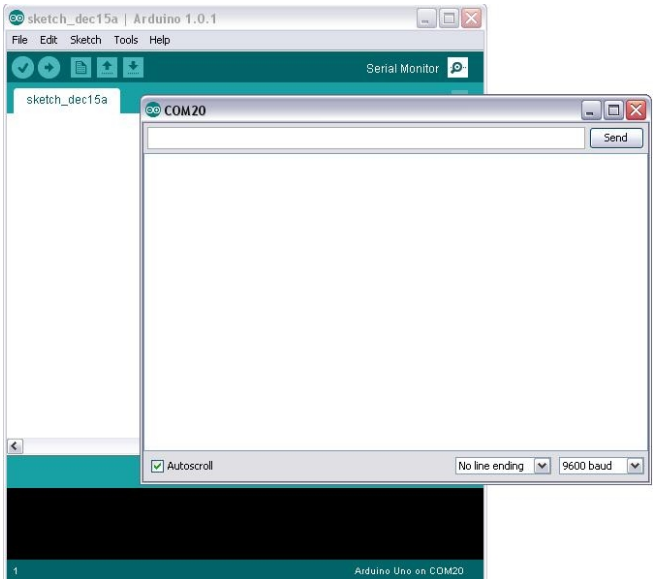
Τώρα είμαστε σε θέση να τρέξουμε τον κώδικα για να βρούμε αρχικά, πιθανά λάθη του, κάνοντας κλικ στο εικονίδιο «Verify» που βρίσκεται ακριβώς από κάτω από το «File».

Εικόνα 9: Έλεγχος ορθότητας κώδικα.

Εφόσον βεβαιωθούμε για την ορθότητα του κώδικα μπορούμε πλέον να τον τρέξουμε. Κάνουμε κλικ στο εικονίδιο «Upload» που απεικονίζεται από ένα δεξί βέλος και βρίσκεται ακριβώς από κάτω από το «Edit».

Εικόνα 10: Φόρτωση κώδικα.

Για να εμφανίσουμε τη σειριακή οθόνη τότε στη μπάρα εργασιών του προγράμματος και από την καρτέλα «Tools» επιλέγουμε «Serial Monitor».

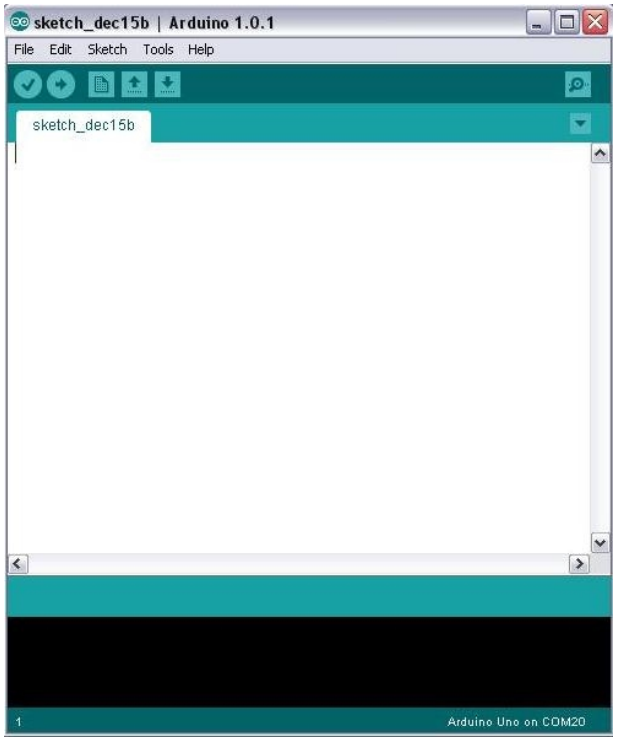


Εικόνα 11: Επιλογή σειριακής οθόνης στο λογισμικό του Arduino.

ΚΕΦΑΛΑΙΟ 2^{ο} : ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ ARDUINO

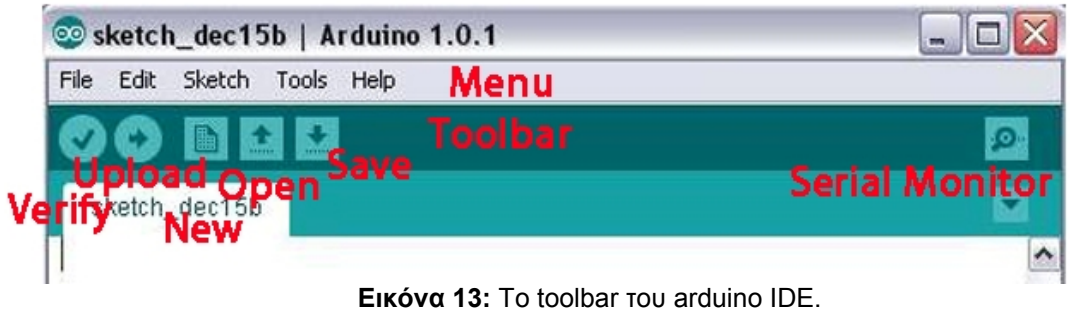
2.1 Το περιβάλλον Arduino IDE (integrated development environment)

Το Arduino IDE είναι αρκετά απλό σε εμφάνιση και σε λειτουργίες σε αντίθεση με άλλα περιβάλλοντα ανάπτυξης λογισμικού αλλά συνάμα παραμένει λειτουργικό και εύχρηστο. Αποτελείται κυρίως από έναν editor (κειμενογράφο), έναν compiler, ένα loader και ένα serial monitor. Δεν περιέχει προχωρημένες λειτουργίες όπως debugging ή code completion, δίνει μόνο τη δυνατότητα για λιγοστές ρυθμίσεις στα «preferences».



Εικόνα 12: Το περιβάλλον προγραμματισμού arduino.

Στην συνέχεια ακολουθεί η εικόνα του toolbar του arduino IDE που δίνει άμεση πρόσβαση στις λειτουργίες για την ανάπτυξη των εφαρμογών που επιθυμεί να αναπτύξει ο χρήστης.



Εικόνα 13: Το toolbar του arduino IDE.

Το κουμπί:

- **Verify** ελέγχει τον κώδικα για συντακτικά λάθη, και μεταγλωττίζει (compile) τον κώδικα που βρίσκεται εκείνη την στιγμή στον editor. Το κουμπί verify εκτός από τον συντακτικό έλεγχο, αφού μεταγλωττιστεί ο κώδικας στη συνέχεια τον μετατρέπει σε μορφή κατάλληλη για να «φορτωθεί» στον μικροελεγκτή του arduino.
- **New** δημιουργεί ένα καινούριο πρόγραμμα διαγράφοντας οτιδήποτε υπάρχει στον editor. Πριν όμως πραγματοποιήσει αυτήν την ενέργεια, δίνει στον χρήστη την ευκαιρία να αποθηκεύσει το υπάρχον sketch (πρόγραμμα) στην περίπτωση που έχει κάνει κάποιες αλλαγές σε αυτό.
- **Open** με το κουμπί αυτό μπορεί ο χρήστης να ανοίξει ένα υπάρχον πρόγραμμα από το σύστημα του.
- **Save** αποθηκεύει τις αλλαγές που έχουν γίνει στο πρόγραμμα που επεξεργάστηκε στον Editor.
- **Upload** όπως και το κουμπί Verify μεταγλωττίζει τον υπάρχων κώδικα στον editor. Με τη διαφορά όμως ότι αφού ελέγξει για τυχόν συντακτικά λάθη και το μετατρέψει σε μορφή κατάλληλη για το arduino, στη συνέχεια θα το προωθήσει στην θύρα που έχει επιλέξει ο προγραμματιστής από το μενού Tools > Serial Port ώστε να «φορτωθεί» στον μικροελεγκτή.
- **Serial Monitor** ανοίγει ένα serial monitor παράθυρο όπου επιτρέπει να παρακολουθεί ο προγραμματιστής τα δεδομένα που στέλνονται προς το arduino αλλά και τα δεδομένα που στέλνονται από το arduino προς τον υπολογιστή. Ο υπολογιστής επικοινωνεί με το arduino μέσω σειριακής σύνδεσης.

Επιλέγοντας από το μενού:

➢Το **Sketch:** πέρα από τη λειτουργία που αναφέρθηκε για την εντολή Verify/Compile, εμφανίζονται και κάποιες άλλες ενδιαφέρουσες λειτουργίες όπως:

- **Show Sketch folder:** είναι μία συντόμευση η οποία ανοίγει σε ένα παράθυρο την διεύθυνση στην οποία το λειτουργικό σύστημα αποθηκεύει τα αρχεία των εφαρμογών.
- **Add File:** επιτρέπει στον χρήση να ανοίξει ένα αρχείο το οποίο μπορεί να βρίσκεται οπουδήποτε μέσα στο σύστημα και να το αποθηκεύσει στον ίδιο φάκελο όπου ανήκει και η εφαρμογή.
- **Import Library:** ο προγραμματιστής έχει την δυνατότητα να εισάγει στο πρόγραμμα που αναπτύσσει οποιαδήποτε βιβλιοθήκη, είτε αυτή είναι του arduino είτε κάποια που δημιούργησε ο ίδιος.

➢Το **Tools:** ο προγραμματιστής έχει την δυνατότητα να επιλέξει την θύρα (serial port) με την οποία θα επικοινωνήσει ο υπολογιστής με το arduino, καθώς και την έκδοση arduino που διαθέτει (board). Εκτός από αυτές τις δύο βασικές λειτουργίες υπάρχουν και άλλες όπως:

- **Auto Format:** μορφοποιεί τον κώδικα που βρίσκεται στον editor κατάλληλα για να διαβάζεται ευκολότερα.

- **Archive Sketch:** μετατρέπει την εφαρμογή που αναπτύσσει ο προγραμματιστής σε ένα αρχείο zip και το αποθηκεύει. Ακόμα σε μερικές περιπτώσεις όπου ένα αρχείο φορτωθεί στο arduino IDE υπάρχει η περίπτωση να περιέχει χαρακτήρες οι οποίοι δεν είναι ASCII, επιλέγοντας το Fix Encoding & Reload ο κώδικας θα ανανεωθεί αντικαθιστώντας τους περίεργους χαρακτήρες σε χαρακτήρες UTF-8 έκδοση.
- **Burning Bootloader:** η επιλογή αυτή δίνει τη δυνατότητα να «καεί» ο Bootloader στην EEPROM μνήμη ενός ATmega328.

2.2 Γλώσσα προγραμματισμού Arduino

Η γλώσσα του Arduino είναι η 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 χρησιμοποιείται η AVR libc της C.

Λόγω της καταγωγής της από τη C, στη γλώσσα του Arduino μπορούμε να χρησιμοποιήσουμε ουσιαστικά τις ίδιες βασικές εντολές και συναρτήσεις (με την ίδια σύνταξη), τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές όπως και στη C. Πέρα από αυτά όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν στη διαχείριση του hardware του Arduino.

Το arduino IDE (Ολοκληρωμένο Περιβάλλον Ανάπτυξης) είναι πρόγραμμα γραμμένο σε Java και βασίζεται πάνω στο περιβάλλον της processing. Τα προγράμματα που γράφονται στην Processing (όπως και στη Wiring) λέγονται sketches-σκίτσα. Τα σκίτσα γράφονται στον επεξεργαστή κειμένου και εκτελούνται - οπτικοποιούνται τρέχοντας τα με το κουμπί Play-Run της εργαλειοθήκης. Κάθε σκίτσο έχει τον δικό του φάκελο με το ίδιο όνομα που έχει ονομαστεί το αρχείο. Επίσης ο φάκελος μπορεί να περιέχει αρχεία και κώδικα που χρησιμοποιείται για την δημιουργία του σκίτσου.

Η processing είναι μία γλώσσα προγραμματισμού ανοικτού κώδικα και παράλληλα ένα περιβάλλον προγραμματισμού για άτομα που θέλουν να δημιουργήσουν εικόνες (στάσιμες ή κινούμενες) και αλληλεπιδράσεις. Αρχικά αναπτύχθηκε για να χρησιμεύσει ως ένα sketchbook λογισμικού και να διδάξει βασικά στοιχεία του προγραμματισμού σε ένα οπτικό πλαίσιο.

Τα προγράμματα του Arduino διαιρούνται σε τρία μέρη: δομή (structure), τιμές (values) και συναρτήσεις (functions), όπως φαίνεται στον παρακάτω πίνακα:

Πίνακας 1: Βασική δομή, τιμές και συναρτήσεις της γλώσσας Wiring

ΔΟΜΗ	ΤΙΜΕΣ	ΣΥΝΑΡΤΗΣΕΙΣ
setup()	Σταθερές	Ψηφιακές I/O
loop()	HIGH - LOW	pinMode()
Έλεγχος	INPUT - OUTPUT	digitalWire()
if	true - false	digitalRead()
if...else	integer constants	Αναλογικές I/O
for	floating point constants	analogReference()
switch case	Τύποι δεδομένων	analogRead()
while	void	analogWrite()
do...while	boolean	Ειδικές I/O
break	(unsigned) char	tone(), noTone()
continue	byte	shiftOut()
return	(unsigned) int	pulseIn()
goto	word	Χρόνος
Σύνταξη	(unsigned) long	millis(), micros()
;	float	delay(), dalayMicroseconds()
{}	double	Μαθηματικές
// ή /* */ (σχόλια)	string (πίνακας char)	min(), max(), abs(), constrain(), map(),
#define	String (αντικείμενο)	pow(), sqrt(), sin(), cos(), tan()
#include	array	randomSeed(), ramdon()
Αριθμητικοί τελεστές	Μετατροπή	Bits/Bytes
= (ανάθεση)	char(), byte(), int(), word(),	lowByte(), highByte() bitRead(), bitWrite()
+ , -, *, /, %	long(), float()	bitSet(), bitClear(), bit()
Σύγκριση	Πεδίο μεταβλητών	Interrupts
== , !=	variable scope, static,	attachInterrupt(), detachInterrupt()
< , > , <= , >=	volatile, const	interrupts(), noInterrupts()
Λογικοί τελεστές	Εργαλεία	Επικοινωνία
&& (και) , || (ή) , ! (όχι)	sizeof()	Serial

2.3 Εισαγωγή στον Προγραμματισμό (Blinking Led)

Μία κοινή τακτική στα βιβλία προγραμματισμού για αρχάριους είναι να δείξουν σε ένα από τα εισαγωγικά κεφάλαια στους αναγνώστες τους πως να εμφανίσουν στην οθόνη τους το μήνυμα «hello world!». Δυστυχώς όμως στον προγραμματισμό των μικροελεγκτών δεν είναι και τόσο εύκολο σε αντίθεση με τις κοινές γλώσσες προγραμματισμού. Έτσι στους μικροελεγκτές συνηθίζεται να δημιουργούν ένα πρόγραμμα το οποίο αναβοσβήνει ένα LED (blinking a LED). Αυτός είναι ο τρόπος με τον οποίο οι μικροελεγκτές λένε «hello world!».

Η πλακέτα arduino έχει ενσωματωμένο ένα LED το οποίο είναι συνδεμένο μεταξύ της γείωσης και της θύρας 13. Έτσι χρησιμοποιώντας την συγκεκριμένη θύρα ο προγραμματιστής δε

χρειάζεται να συνδέσει τις απαραίτητες ηλεκτρονικές διατάξεις (LED, αντιστάσεις, καλώδια) για να πραγματοποιήσει το ακόλουθο παράδειγμα.

Εκτελώντας το arduino IDE, σε περίπτωση που ο editor δεν είναι καθαρός θα πρέπει να δημιουργηθεί ένα καινούριο αρχείο. Χρησιμοποιώντας το κουμπί New από το toolbar είτε μπορεί να γίνει επιλέγοντας το New από το μενού. Στην συνέχεια αναπτύσσεται και αποθηκεύεται το sketch που βρίσκεται ακριβώς από κάτω, δίνοντας του το όνομα «blinking led».

```
// παράδειγμα sketch: Blinking LED
const int LED = 13; //το LED συνδέεται στο ψηφιακή θύρα 13
void setup()
{
        pinMode(LED, OUTPUT); // η θύρα 13 δηλώνετε ως θύρα έξοδου
}
void loop() {
        digitalWrite(LED, HIGH);// Η θύρα LED πηγαίνει σε κατάσταση high(+5Volt)
        delay(1000); // Το πρόγραμμα περιμένει για ένα δευτερόλεπτο
        digitalWrite(LED, LOW);//Η θύρα LED πηγαίνει σε κατάσταση low(0Volt)
        delay(1000); // Το πρόγραμμα περιμένει για ένα δευτερόλεπτο
}
```

Από την στιγμή που ο κώδικας έχει γράφει στον editor, το επόμενο βήμα είναι να γίνει ο έλεγχος του πατώντας το κουμπί Verify. Στην περίπτωση που όλα είναι σωστά και δεν υπάρχουν συντακτικά λάθη τότε εμφανίζεται στο κάτω μέρος του Arduino IDE το μήνυμα «Done compiling». Το μήνυμα που εμφανίζεται σημαίνει ότι το Arduino IDE τροποποίησε τον κώδικα σε ένα εκτελέσιμο αρχείο το οποίο μπορεί να «τρέξει» στο arduino, κάτι αντίστοιχο με τα .exe αρχεία για τα Windows και τα .app αρχεία για τα Mac Os. Στην περίπτωση όμως που υπάρχουν λάθη στον κώδικα, τότε στη θέση του μηνύματος «Done compiling» εμφανίζεται ένα αντίστοιχο μήνυμα σφάλματος.

Αφού έχει μεταγλωττιστεί σωστά ο κώδικας, μπορεί πλέον πατώντας το κουμπί Upload να «φορτωθεί» στην πλακέτα arduino. Αυτό θα επαναφέρει το arduino στις αρχικές του ρυθμίσεις, αναγκάζοντας το να σταματήσει οποιαδήποτε ενέργεια πραγματοποιούσε μέχρι εκείνη την στιγμή και να «ακούσει» τις οδηγίες που θα λάβει από την USB θύρα. Μόλις ο μικροελεγκτής σταματήσει να λειτουργεί, το arduino IDE στέλνει το sketch που έχει δημιουργήσει ο προγραμματιστής προς την πλατφόρμα, το οποίο θα αποθηκευτεί στην μνήμη του μικροελεγκτή και τελικά θα εκτελεστεί. Στη συνέχεια θα εμφανιστούν μερικά μηνύματα στην μαύρη περιοχή που βρίσκεται στο κάτω μέρος του arduino IDE και ακριβώς από πάνω το μήνυμα «Done uploading» το οποίο ενημερώνει τον προγραμματιστή ότι η διαδικασία τελείωσε με επιτυχία. Τα δύο LED RX και TX στην πλακέτα αναβοσβήνουν κάθε φορά που ένα byte λαμβάνεται ή στέλνεται από το arduino. Έτσι κατά την διάρκεια της διαδικασίας της «φόρτωσης» του sketch προς τον μικροελεγκτή τα δύο αυτά LED θα πρέπει αναβοσβήνουν συνεχώς.

Εάν δεν παρατηρηθεί να αναβοσβήνουν τα LED ή εμφανιστεί ένα μήνυμα σφάλματος αντί για το «Done uploading» στην μαύρη περιοχή, τότε υπάρχει πρόβλημα επικοινωνίας μεταξύ του υπολογιστή και του arduino. Ο προγραμματιστής θα πρέπει να σιγουρευτεί από το μενού ότι έχει

επιλέξει τις σωστές ρυθμίσεις της πλακέτας (tools > boards) και της θύρας (tools > serial port) διαλέγοντας την σωστή έκδοση του arduino και την σωστή θύρα που είναι συνδεμένο το arduino με τον υπολογιστή.

Από την στιγμή που ο κώδικας «φορτώθηκε» στον μικροελεγκτή τότε αυτός παραμένει εκεί έως ότου να αντικατασταθεί από ένα άλλο sketch. Το υπάρχον sketch διατηρείται στον μικροελεγκτή του arduino μετά από reset (επαναφορά αρχικών ρυθμίσεων) ή μετά από τη διακοπή της παροχής ρεύματος (turned off). Με την προϋπόθεση ότι όλα έγιναν με επιτυχία τότε το LED «L» που είναι ενσωματωμένο στο Arduino θα πρέπει να αναβοσβήνει ανά ένα δευτερόλεπτο σύμφωνα με τον κώδικα. Αυτό που μόλις δημιουργήθηκε είναι ένα μικρό πρόγραμμα στον υπολογιστή ή sketch.

2.3.1 Επεξήγηση του Sketch

```
// παράδειγμα sketch: Blinking LED
```

Τα σχόλια είναι απαραίτητα όταν αναπτύσσονται εφαρμογές, ιδιαίτερα όταν αυτές αποτελούνται από πολλές γραμμές κώδικα. Ο μεταγλωττιστής όταν συναντάει κείμενο το οποίο στα αριστερά του ξεκινάει με "//" το αγνοεί. Αφήνοντας σχόλια στα κυριότερα σημεία του κώδικα, γίνετε ευκολότερος ο τρόπος για να διαβαστεί από έναν προγραμματιστή που τον ανοίγει για πρώτη φορά αλλά και για τον ίδιο τον προγραμματιστή που τον έγραψε όταν τον ξαναδιαβάσει μετά από ένα χρονικό διάστημα.

```
const int LED = 13;
```

const int σημαίνει ότι η LED είναι μία integer μεταβλητή η οποία δεν πρόκειται να αλλάξει ποτέ η τιμή της και θα αντιστοιχεί στον αριθμό 13. Έτσι ενημερώνεται το arduino όπου βλέπει την λέξη LED να την διαβάζει σαν τον ακέραιο αριθμό 13.

```
void setup()
```

Στην γραμμή αυτή δηλώνουμε στο arduino ότι το block που ακολουθεί ονομάζεται setup().

```
{
```

Με αυτήν την ανοιχτή αγκύλη, ένα block κώδικα ξεκινάει.

```
pinMode(LED, OUTPUT); //set the digital pin as output
```

Το pinMode λέει στο arduino πως θα διαμορφώσει μία καθορισμένη θύρα (pin).

Μία ψηφιακή θύρα μπορεί να χρησιμοποιηθεί ως είσοδος (INPUT) ή ως έξοδος (OUTPUT). Στην συγκεκριμένη περίπτωση θέλουμε μία θύρα εξόδου για να ελέγξουμε το LED, έτσι τοποθετούμε τον αριθμό της θύρας και την κατάσταση της θύρας μέσα στην παρένθεση. Η pinMode είναι μία συνάρτηση και οι λέξεις (ή οι αριθμοί) μέσα στην παρένθεση είναι τα ορίσματα. Η INPUT και η OUTPUT είναι σταθερά ορίσματα της γλώσσας του arduino.

```
}
```

Αυτή η αγκύλη δηλώνει το τέλος της συνάρτησης setup().

```
void loop() {
```
Η συνάρτηση loop() είναι υπεύθυνη για την διαδραστική λειτουργία του μικροελεγκτή. Αυτή θα επαναλαμβάνεται ξανά και ξανά έως ότου να σταματήσει να παρέχεται ρεύμα στην πλακέτα.

```
digitalWrite(LED, HIGH); //turns the LED on
```
Η συνάρτηση ενεργοποιεί ή απενεργοποιεί την θύρα η οποία έχει οριστεί ως θύρα έξοδου. Το πρώτο όρισμα (στην συγκεκριμένη περίπτωση LED) ορίζει σε πιο pin (θύρα) του arduino θα πραγματοποιηθεί κάποια ενέργεια (άρα εδώ που το όρισμα είναι το LED απευθύνεται στην θύρα 13 όπως δηλώθηκε στην αρχή του κώδικα). Το δεύτερο όρισμα δηλώνει αν η θύρα θα πάει σε κατάσταση HIGH (+5 volt) ή σε κατάσταση LOW (0 volt). Σε αυτή την περίπτωση η θύρα θα πάει σε κατάσταση HIGH και το LED που βρίσκεται στο pin 13 θα ανάψει.

```
delay(1000); // waits for a second
```
Η delay(argument) είναι μία συνάρτηση η οποία λέει στο arduino να μη κάνει απολύτως τίποτα και να περιμένει έως ότου να περάσουν όσα milliseconds αντιστοιχούν στο όρισμα (argument). Έτσι το led που βρίσκεται συνδεμένο στην θύρα 13 παραμένει αναμμένο έως ότου η κατάσταση του pin αλλάξει από HIGH σε LOW.

```
digitalWrite(LED, LOW); // turns the led off
```
Εδώ χρησιμοποιείται η ίδια συνάρτηση με πριν, με την διαφορά ότι το δεύτερο όρισμα είναι το LOW. Το αποτέλεσμα θα είναι η θύρα 13 να τεθεί σε κατάσταση LOW. Άρα στο LED δεν θα παρέχονται πλέον τα +5volt με αποτέλεσμα να σβήσει.

```
delay(1000); // waits for a second
```
Η συνάρτηση delay καλείται για ακόμη μία φορά και το led θα παραμείνει σβηστό για 1000 milliseconds (1 second).

```
}
```
Η κλειστή αγκύλη δηλώνει το τέλος της συνάρτησης loop() και ο κώδικας που βρίσκεται μέσα σε αυτήν θα επαναλαμβάνεται από την αρχή, έως ότου να σταματήσει να παρέχεται ρεύμα στον ελεγκτή.

2.4 Μεταβλητές και τύποι δεδομένων

Όταν δηλώνουμε μία καινούρια μεταβλητή τότε χρειαζόμαστε τουλάχιστον δύο είδη πληροφορίας: τον τύπο και το όνομα της μεταβλητής. Αυτή η ενέργεια ονομάζεται ορισμός (declaring) μίας μεταβλητής όπου δεσμεύει έναν χώρο στην μνήμη του μικροελεγκτή με σκοπό την αποθήκευση πληροφοριών.

2.4.1 Ονομασία μεταβλητών

Για την ονομασία των μεταβλητών υπάρχουν κάποιοι κανόνες που θα πρέπει να τηρηθούν. Πρώτα απ' όλα το όνομα μίας μεταβλητής ή συνάρτησης μπορεί να περιέχει γράμματα, αριθμούς, το σύμβολο του δολαρίου και underscores. Κενά και περίεργους χαρακτήρες όπως για παράδειγμα

@ ή & δεν επιτρέπονται, όπως επίσης δεν επιτρέπεται και να ξεκινάει με αριθμό το όνομα μίας μεταβλητής. Παράδειγμα επιτρεπτών ονομάτων από μεταβλητές είναι myVariable, mYvAlUe, DELAY_TIME, ενώ τα ακόλουθα ονόματα 1stValue, delay time, και time&money δεν επιτρέπονται. Τα ονόματα των μεταβλητών συνήθως αντιπροσωπεύουν τον λόγο της ύπαρξης τους με σκοπό να είναι πιο κατανοητή η χρήση τους.

2.4.2 Τύποι δεδομένων των μεταβλητών

Κάθε δεδομένο το οποίο αποθηκεύεται στο arduino πρέπει να αντιστοιχεί σε ένα τύπο. Ανάλογα τις ανάγκες των δεδομένων ο τύπος μπορεί να είναι ένας από τους παρακάτω.

- **Boolean:** οι τιμές αυτών των τύπων καταλαμβάνουν μόνο ένα byte στην μνήμη και μπορεί να αντιστοιχεί σε true ή false.
- **Char**: οι μεταβλητές αυτές καταλαμβάνουν ένα byte στην μνήμη και αποθηκεύουν αριθμούς από το -128 έως το 127. Τις περισσότερες φορές αυτοί οι αριθμοί αντιπροσωπεύουν χαρακτήρες κωδικοποιημένους σε ASCII. Στο παράδειγμα που ακολουθεί, οι μεταβλητές c1 και c2 έχουν τις ίδιες τιμές.

  Char c1 = ‘A’;

  Char c2 = 65;
- **Byte**: είναι μεταβλητές που καταλαμβάνουν ένα byte στην μνήμη και παίρνουν τιμές από το 0 έως το 255.
- **Int**: οι μεταβλητές αυτές καταλαμβάνουν δύο byte χώρο στην μνήμη και αποθηκεύουν τιμές από το -32,768 έως το 32,767.
- Για μεγαλύτερους αριθμούς χρησιμοποιείται η **long**. Καταλαμβάνει 4 byte στην μνήμη και αποθηκεύει τιμές από το -2,147,483,648 έως το 2,147,483,647.
- Η **float** και **double** μεταβλητές είναι ίδιες και μπορούν να χρησιμοποιηθούν για να δηλώσουν αριθμούς κινητής υποδιαστολής. Και οι δύο τύποι καταλαμβάνουν 4 byte στην μνήμη και αποθηκεύουν τιμές από το -3.4028235E+38 έως το 3.4028235E+38.

2.5 Πίνακες (Array)

Ο ορισμός ενός πίνακα είναι παρόμοιος με τον ορισμό μίας μεταβλητής. Η απλούστερη σύνταξη για τον ορισμό ενός πίνακα είναι:

«τύπος δεδομένων» «όνομα πίνακα» [«αριθμός στοιχείων»];

Ο τύπος δεδομένων αντιστοιχεί στον τύπο των δεδομένων που θα αποθηκευτούν στον πίνακα ενώ μέσα στις αγκύλες δηλώνεται το πλήθος των δεδομένων που θα αποθηκευτούν μέσα στον πίνακα. Για παράδειγμα η εντολή int myArray[3]; θα δημιουργήσει 3 νέα στοιχεία με το όνομα myArray[0], myArray[1] και myArray[2]. Το πρώτο στοιχείο σε έναν πίνακα βρίσκεται πάντα στην θέση 0.

Για να εκχωρηθούν οι τιμές στον πίνακα υπάρχουν δύο τρόποι. Ο πρώτος τρόπος σύνταξης είναι:

«όνομα πίνακα»[«θέση στοιχείου»] = «τιμή στοιχείου»

Η τιμή που βρίσκεται στα δεξιά του τελεστή εκχώρησης αποθηκεύεται στην θέση του πίνακα που ορίζεται μέσα στις αγκύλες. Ο δεύτερος τρόπος σύνταξης είναι:

«τύπος δεδομένων» «όνομα πίνακα»[] = {«τιμή 1ου στοιχείου, τιμή 2ου στοιχείου ... »};

Ο τρόπος αυτής της εκχώρησης χρησιμοποιείται μόνο κατά τον ορισμό του πίνακα. Μέσα στις αγκύλες {} περιέχονται τα στοιχεία χωρισμένα με κόμμα που πρόκειται να αποθηκευτούν μέσα στον πίνακα, το πρόγραμμα μετράει τα στοιχεία και γνωρίζει τον αριθμό των θέσεων του πίνακα.

Όταν καλείται το όνομα του πίνακα και μέσα στις αγκύλες [] βρίσκεται ο αριθμός μίας θέσης του, τότε επιστρέφει η τιμή που είναι αποθηκευμένη στην συγκεκριμένη θέση. Ένα string είναι ένας πίνακας από char μεταβλητές. Η σύνταξη ενός char πίνακα μπορεί να διαφέρει λίγο. Στο παράδειγμα που ακολουθεί όλοι οι πίνακες έχουν το ίδιο περιεχόμενο.

```
char string1[8] = { 'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0' };
char string2[] = "Arduino";
char string3[8] = "Arduino";
char string4[] = { 65, 114, 100, 117, 105, 110, 111, 0 };
```

Τα string πρέπει πάντα να τερματίζονται με ένα μηδενικό byte. Όταν χρησιμοποιούνται double quotes για την δημιουργία ενός string τότε το μηδενικό byte προστίθεται αυτόματα. Αυτός είναι και ο λόγος που πρέπει να προσθέτουμε ακόμη μία μονάδα στο μέγεθος των περιεχομένων του πίνακα.

2.6 Qualifiers μεταβλητών

Όταν δηλωθεί μία μεταβλητή μπορούν να χρησιμοποιηθούν τα variable qualifiers. Σκοπός τους είναι η παραλλαγή της συμπεριφοράς μίας μεταβλητής για να αλλάξουν κάποιες λειτουργίες στον τρόπο που χρησιμοποιείται.

Η **const ή constant** μετατρέπει μία μεταβλητή σε κατάσταση read-only (μόνο ανάγνωση) και δεν μπορεί να αλλάξουν οι πληροφορίες της από την στιγμή που οριστεί. Για παράδειγμα:

```
const int red = 9;
```

Η μεταβλητή red θα περιέχει τον ακέραιο αριθμό 9 και δε θα μπορεί να αλλάξει την τιμή αυτή ποτέ ξανά.

Για να μετατραπεί το εύρος των τιμών που παίρνουν οι μεταβλητές ενός τύπου δεδομένων χρησιμοποιείται το unsigned. Το **unsigned** είναι και αυτό ένα variable qualifier το οποίο επιτρέπει στις μεταβλητές να παίρνουν μόνο θετικές τιμές. Έτσι αφού η μεταβλητή δεν περιμένει ποτέ να πάρει αρνητικό πρόσημο, διπλασιάζει το εύρος των θετικών τιμών. Για παράδειγμα:

```
unsigned int maxThreshold = 1024;
```

Πιο συγκεκριμένα τοποθετώντας τη λέξη unsigned μπροστά από μία integer μεταβλητή κατά τον ορισμό της, τότε αυτή η μεταβλητή θα παίρνει μόνο θετικές τιμές μεταξύ του 0 και του 65,535.

2.7 Προκαθορισμένες μεταβλητές

Όπως ο χρήστης δηλώνει κάποιες μεταβλητές μόνο για ανάγνωση (constant) έτσι και το περιβάλλον ανάπτυξης arduino έχει κάποιες δικές του μεταβλητές με προκαθορισμένες τιμές. Οι πρώτες από αυτές τις προκαθορισμένες μεταβλητές είναι η **TRUE** και η **FALSE** βασισμένες στη boolean λογική και τους λογικούς πίνακες αληθείας. Η κατάσταση false εύκολα μπορεί να οριστεί ως μηδέν ή μερικές φορές ως «off». Από την άλλη πλευρά η κατάσταση true είναι οτιδήποτε άλλο εκτός του false όπως για παράδειγμα 1, «on» ή κάποιος άλλος αριθμός έκτος του μηδενός.

Ένα άλλο ζευγάρι προκαθορισμένων μεταβλητών είναι το **INPUT** και το **OUTPUT.** Αυτές οι δύο μεταβλητές καθαρίζουν τον τρόπο λειτουργίας μίας ψηφιακής θύρας (digital pin) χρησιμοποιώντας την συνάρτηση pinMode.

Επίσης υπάρχουν ακόμη δύο προκαθορισμένες μεταβλητές η **HIGH** και η **LOW** όπου κατά προσέγγιση αντιπροσωπεύουν τις τιμές +5 και 0 volt ή on και off. Χρησιμοποιούνται με σκοπό να δηλώσουν την κατάσταση μίας θύρας.

2.8 Εμβέλεια μεταβλητών

Στις προηγούμενες παραγράφους περιγράφηκε ο τρόπος δήλωσης μίας μεταβλητής, ένα άλλο επίσης σημαντικό είναι το μέρος στο οποίο θα δηλωθεί αυτή. Η τοποθεσία που θα οριστεί μία μεταβλητή δηλώνει και την εμβέλεια της. Αυτό είναι γνωστό και ως variable scope.

Οι μεταβλητές στο παράδειγμα blinking led δηλώθηκαν έξω από τις συναρτήσεις setup() και loop() στην αρχή του κώδικα. Αυτές που δηλώνονται εκτός κάποιας συνάρτησης είναι γνωστές και ως global μεταβλητές (variables) και μπορούν να χρησιμοποιηθούν σε όλο το sketch. Σε αντίθεση, αν κάποια μεταβλητή έχει δηλωθεί μέσα σε μία συνάρτηση για παράδειγμα μέσα στην setup(), τότε αυτή θα είναι διαθέσιμη μόνο μέσα στο block που δηλώθηκε και δε θα μπορούσε να χρησιμοποιηθεί έξω από αυτό. Αυτές είναι γνωστές ως local μεταβλητές.

2.9 Εκχώρηση τιμών

Για να γίνει η εκχώρηση τιμών στις μεταβλητές, χρησιμοποιείται ο τελεστής εκχώρησης «=». Αυτός είναι το μαθηματικό σύμβολο της ισότητας αλλά δεν λειτουργεί με τον ίδιο ακριβώς τρόπο όπως στα μαθηματικά. Ο τελεστής λέει στον compiler να πάρει τις τιμές που βρίσκονται στα δεξιά του και να τις εκχωρήσει στην μεταβλητή που βρίσκεται στα αριστερά του.

2.10 Μαθηματικοί τελεστές

Υπάρχουν πέντε βασικοί τελεστές που χρησιμοποιούνται για να κάνουν τις βασικές αριθμητικές πράξεις.

Πρώτος τελεστής είναι αυτός της πρόσθεσης «+». Έτσι αν επιθυμεί ο προγραμματιστής να προσθέσει δύο στοιχεία και να εκχωρήσει σε μία μεταβλητή το αποτέλεσμα τους, θα το κάνει κάπως έτσι:

```
myValue = 4 + 38;
```
Ξεκινώντας από τα δεξιά το arduino θα προσθέσει το 4 με το 38 και θα εκχωρήσει το άθροισμα τους στην μεταβλητή myValue που βρίσκεται στα αριστερά του τελεστή εκχώρησης.

```
myValue = 255;
newValue = myValue - 128;
```
Ο τελεστής της αφαίρεσης «-» στο παραπάνω παράδειγμα θα αφαιρέσει από την μεταβλητή myValue η οποία ισούται με το 255 τον αριθμό 128 και το αποτέλεσμα θα το εκχωρήσει στη μεταβλητή newValue.

```
float pi = 3.14159;
int diameter = 5;
float C;
C = pi * diameter;
```
Στην συγκεκριμένη περίπτωση ο τελεστής του πολλαπλασιασμού «*» αγνοεί τον τύπο της μεταβλητής diameter που είναι integer και πολλαπλασιάζει τις δύο μεταβλητές μεταξύ τους σαν να είναι float. Στην συνέχεια εκχωρεί το αποτέλεσμα στην μεταβλητή C.

```
int myValue;
myValue = 9 / 5;
```
Ο τελεστής διαίρεσης «/» πραγματοποιεί μία διαίρεση μεταξύ των τελεστών και επιστρέφει το αποτέλεσμα. Στο παραπάνω παράδειγμα ο τελεστής διαίρεσης θα κάνει την διαίρεση και θα επιστρέψει την τιμή 1, εξαιτίας του ότι η αριθμητική πράξη γίνεται μεταξύ ακέραιων μεταβλητών. Έτσι η μεταβλητή myValue θα περιέχει την τιμή 1 αντί για 1.8 στην περίπτωση που οι μεταβλητές ήταν αριθμοί κινητής υποδιαστολής.

Τέλος ο τελεστής υπολοίπου **«%»** επιστρέφει το υπόλοιπο μίας διαίρεσης μεταξύ ακέραιων τελεστών. Η έκφραση δηλαδή 9%5 θα επέστρεφε την τιμή 0.8 επειδή είναι το υπόλοιπο της διαίρεσης.

2.11 Σύνθετοι τελεστές

2.11.1 Τελεστές προσαύξησης και μείωσης

Μία συνηθισμένη εργασία είναι να προστίθεται ή να αφαιρείται το 1 από μία ακέραια μεταβλητή.

```
myValue = myValue +1;
```
Το παραπάνω παράδειγμα θα μπορούσε να αντικατασταθεί χρησιμοποιώντας έναν σύνθετο τελεστή. Στην περίπτωση δηλαδή που θέλουμε να αυξήσουμε μία μεταβλητή κατά ένα, για συντομία θα μπορούσε να γραφεί το όνομα της μεταβλητής και ο τελεστής ++. Η γραμμή κώδικα που ακολουθεί θα είχε ακριβώς το ίδιο αποτέλεσμα με την παραπάνω.

```
myValue++;
```
Στην περίπτωση που η μεταβλητή έπρεπε να μειωθεί κατά 1, θα χρησιμοποιούνταν ο αντίστοιχος τελεστής --. Η σύνταξη της εντολής θα ήταν:

```
myValue--;
```

Αυτοί οι δύο τελεστές χρησιμοποιούνται μόνο για να αυξήσουν ή να μειώσουν μία μεταβλητή κατά μία μονάδα. Αν ο προγραμματιστής επιθυμεί να αυξήσει ή να μειώσει την μεταβλητή με αριθμό μεγαλύτερο από το 1 τότε μπορεί να χρησιμοποιήσει τους σύνθετους τελεστές εκχώρησης **+=, -=, *=, /=**. Για παράδειγμα αν θέλει να αυξήσει μία μεταβλητή κατά 10 μπορεί να εκτελέσει την ακόλουθη εντολή:

```
myValue +=10;
```

Η δήλωση αυτή είναι η ίδια με την myValue = myValue + 10;. Δηλαδή προσθέτει την μεταβλητή myValue με το 10 και στη συνέχεια εκχωρεί το αποτέλεσμα της πρόσθεσης στην ίδια μεταβλητή.

Επίσης μπορεί η μεταβλητή να πολλαπλασιαστεί με έναν αριθμό και το αποτέλεσμα να αποθηκευτεί στην ίδια. Για παράδειγμα:

```
myValue *=1,5;
```

Η μεταβλητή myValue πολλαπλασιάζεται με το 1.5 και το αποτέλεσμα εκχωρείται πάλι στην myValue, έτσι αν η αρχική τιμή της ήταν 10, τώρα θα ισούται με το 15.
Παρόμοια λειτουργούν και οι υπόλοιποι σύνθετοι τελεστές εκχώρησης -=, /=.

2.11.2 Συσχετιστικοί τελεστές

Το arduino μπορεί να χρησιμοποιηθεί για την λήψη αποφάσεων, έτσι αν ο προγραμματιστής θέλει να ελέγξει κατά πόσο ή όχι μία μεταβλητή ανέκτησε την επιθυμητή τιμή, θα πρέπει να χρησιμοποιήσει μία έκφραση σαν αυτή που ακολουθεί.

```
myValue == 5
```

Στο παράδειγμα αυτό χρησιμοποιείται ο τελεστής συσχέτισης «**==**», ο οποίος ελέγχει εάν μία μεταβλητή ή τιμή που βρίσκεται στο αριστερό μέρος είναι ίση με μία μεταβλητή ή τιμή που βρίσκεται στο δεξιό μέρος. Στην περίπτωση του παραδείγματος η έκφραση θα πρέπει να επιστρέψει true εάν η μεταβλητή myValue περιέχει την τιμή 5, ενώ false αν περιέχει διαφορετική τιμή.

Ένας παρόμοιος τελεστής σύγκρισης που επιστρέφει ακριβώς τα αντίθετα αποτελέσματα με τον προηγούμενο είναι το «**!=**». Σύμφωνα με αυτόν τον τελεστή όταν το αριστερό στοιχείο είναι διαφορετικό από το δεξιό τότε επιστρέφει true , ενώ όταν έχουν την ίδια τιμή επιστρέφει false.

Άλλοι συσχετιστικοί τελεστές είναι το **>** και το **<**. Ο τελεστής > επιστρέφει true όταν συγκρίνει δύο στοιχεία και το αριστερό είναι μεγαλύτερο από το δεξιό στοιχείο. Αντίθετα ο τελεστής < επιστρέφει true όταν το αριστερό στοιχείο είναι μικρότερο από το δεξιό. Στις περιπτώσεις που τα στοιχεία είναι ίσα μεταξύ τους τότε επιστρέφεται η τιμή false.

Τέλος υπάρχει ακόμη ένα ζευγάρι τελεστών σύγκρισης. Ο τελεστής **>=** ο οποίος επιστρέφει true στην περίπτωση που το αριστερό στοιχείο είναι μεγαλύτερο ή ίσο με το δεξιό και false εάν το αριστερό είναι μικρότερο. Ο τελεστής **<=** επιστρέφει την τιμή true εάν το αριστερό στοιχείο είναι μικρότερο ή ίσο με το δεξιό στοιχείο, σε διαφορετική περίπτωση θα επιστρέψει false.

2.11.3 Λογικοί τελεστές

Οι συγκριτικοί τελεστές δουλεύουν ιδανικά όταν θέλουμε να συγκρίνουμε δύο τιμές μεταξύ τους, υπάρχουν όμως περιπτώσεις που απαιτούν πιο πολύπλοκες συγκρίσεις. Σε τέτοιες περιπτώσεις χρησιμοποιούνται οι λογικοί τελεστές. Για παράδειγμα όταν υπάρχουν δύο ξεχωριστές συγκρίσεις και αυτές θα πρέπει να κάνουν κάτι μόνο στην περίπτωση που είναι και οι δύο αληθείς. Τότε η έκφραση θα πρέπει να γραφεί κάπως έτσι:

```
myValue >= 10 && myValue <=20
```

Η παραπάνω έκφραση χρησιμοποιεί το λογικό **AND** «**&&**» και επιστρέφει true μόνο όταν οι δύο εκφράσεις, στα αριστερά και στα δεξιά του λογικού τελεστή επιστρέψουν true. Άρα για να συμβεί αυτό, η τιμή της μεταβλητής myValue θα πρέπει να είναι μεγαλύτερη του 9 και μικρότερη του 21. Μόνο όταν η τιμή της μεταβλητής βρίσκεται μέσα στο εύρος τιμών 10 έως 20 θα επιστρέψει true.

Όταν ο προγραμματιστής όμως επιθυμεί να επιστραφεί η τιμή true στην περίπτωση που τουλάχιστον μία από τις δύο συνθήκες είναι true τότε θα πρέπει να χρησιμοποιήσει το λογικό **OR** «**||**». Η έκφραση αυτή θα επιστρέψει true όταν μία ή και οι δύο από τις εκφράσεις που βρίσκονται αριστερά και δεξιά του λογικού τελεστή επιστρέψουν true. Για παράδειγμα:

```
myValue < 10 || myValue > 20
```

Τέλος υπάρχει και ο λογικός τελεστής **NOT** «**!**». Ο τελεστής αυτός αντιστρέφει την τιμή μίας έκφρασης boolean. Δηλαδή στο επόμενο παράδειγμα:

```
!myValue
```

θα επιστρέψει true μόνο όταν η μεταβλητή myValue είναι false. Στην περίπτωση που η μεταβλητή είναι true τότε η έκφραση θα επιστρέψει false.

2.12 Σειρά προτεραιότητας των τελεστών

Όταν οι πράξεις γίνονται πιο πολύπλοκες τότε ακολουθείται μία σειρά προτεραιότητας για τις πράξεις. Πιο αναλυτικά, πρώτα πραγματοποιούνται οι πράξεις οι οποίες βρίσκονται μέσα στις παρενθέσεις ή αγκύλες «[], ()». Στην συνέχεια πραγματοποιούνται οι πράξεις με τους τελεστές μοναδιαίας αύξησης ++ και μείωσης --. Αμέσως μετά, την προτεραιότητα έχουν οι πράξεις του πολλαπλασιασμού, της διαίρεσης και του υπολοίπου (*, /, %) ενώ μετά σειρά έχουν οι προσθέσεις και οι αφαιρέσεις (+, -). Αφού τελειώσουν οι αριθμητικές πράξεις πραγματοποιούνται οι συγκριτικές εντολές και στην συνέχεια οι λογικές. Τελευταίοι σε προτεραιότητα είναι οι τελεστές εκχώρησης δηλαδή το «=» καθώς και οι σύνθετοι (+=, -=,*=, /=).

2.13 Εντολές ελέγχου

Υπάρχουν δύο ειδών εντολές διαθέσιμες στο arduino που ελέγχουν την ροή του προγράμματος. Οι εντολές υπό συνθήκη (conditional statement) οι οποίες παίρνουν μία απόφαση ανάλογα με το εάν ή όχι μία συνθήκη επιστρέφει true και οι επαναληπτικές καταστάσεις (iterative statement) οι οποίες επαναλαμβάνουν ένα block εντολών αρκετές φορές έως ότου μία συνθήκη

επιστρέψει false. Στις καταστάσεις υπό συνθήκη περιλαμβάνονται η if – else και η switch εντολές ελέγχου, οι οποίες επιτρέπουν την εκτέλεση ορισμένων ενεργειών σύμφωνα με την κατάσταση της συνθήκης. Στις επαναλαμβανόμενες καταστάσεις περιλαμβάνονται η for και η while, οι οποίες συχνά αποκαλούνται και loops (βρόχοι) επειδή όπως και η συνάρτηση loop() επαναλαμβάνουν ότι βρίσκεται μέσα στο βρόχο έως ότου να επιστρέψει false η υπεύθυνη συνθήκη.

Εντολή if else

Η εντολή if είναι η απλούστερη μορφή ελέγχου και από τις σημαντικότερες εντολές για την λήψη αποφάσεων. Η βασική σύνταξη είναι:

```
If (συνθήκη) {
εντολές
}
```

Η λέξη if ακολουθείται από ένα ζευγάρι παρενθέσεων (), και μέσα σε αυτές βρίσκεται η συνθήκη. Για παράδειγμα εάν το πρόγραμμα πρέπει να τρέξει μία εντολή όταν η μεταβλητή myValue ισούται με το 5, η εντολή θα είναι: if (myValue == 5).

Στην περίπτωση που η συνθήκη επιστρέψει true τότε θα εκτελεστεί η αμέσως επόμενη εντολή ή το block που ακολουθεί μέσα στις αγκύλες {}. Μία κοινή χρήση της συνθήκης if είναι ο έλεγχος μίας ψηφιακής θύρας του arduino και η εκτέλεση μιας εντολής σύμφωνα με την συνθήκη που βρίσκεται.

Όταν όμως η συνθήκη if επιστρέψει false τότε η εντολή ή το block που ακολουθεί θα παρακαμφθούν. Στην περίπτωση αυτή ίσως ο προγραμματιστής επιθυμεί να εκτελέσει μία άλλη εντολή, τότε μπορεί να χρησιμοποιηθεί προαιρετικά η εντολή else.

Παρακάτω φαίνεται ο τρόπος σύνταξης της if μαζί με την else.

```
If (συνθήκη) {
        εντολή
}
else {
        εντολή
}
```

Έτσι στην περίπτωση που η συνθήκη της if είναι μη αληθής τότε θα εκτελεστεί η έκφραση ή το block που βρίσκεται ακριβώς μετά την else. Αντίστοιχα όταν η συνθήκη της if είναι αληθής τότε η εντολή else αγνοείται.

Εντολή switch

Στη περίπτωση που χρησιμοποιηθούν πολλές συνθήκες if με σκοπό να καλυφθεί ένα εύρος τιμών ελέγχου μίας μεταβλητής, τότε το πρόγραμμα γίνεται πολύπλοκο. Η εντολή switch μπορεί να απλοποιήσει αλλά και να επιταχύνει την διαδικασία εκτέλεσης των εντολών. Η σύνταξη της είναι:

```
switch(μεταβλητή)
{
```

```
        case «περίπτωση τιμής μεταβλητής»:
                εντολή 1;
        case «περίπτωση τιμής μεταβλητής 2»:
                εντολή 2;
        case «περίπτωση τιμής μεταβλητής 3»:
                εντολή 3;
        ...
        ...
        default :
                εντολή χ;
}
```

Μετά την λέξη switch αναφέρεται το όνομα της μεταβλητής που θα ελεγχθεί μέσα στις παρένθεσεις (). Στην συνέχεια μέσα στο block καταγράφονται οι περιπτώσεις (cases) της μεταβλητής, έτσι όταν η τιμή της μεταβλητής αντιστοιχήσει με μία περίπτωση τότε εκτελείται η αντίστοιχη εντολή. Όταν όμως η μεταβλητή δεν αντιστοιχίσει σε κάποια περίπτωση, τότε εκτελείται η default εντολή. Η default μπορεί να παραληφθεί.

Μετά το τέλος των εντολών κάθε περίπτωσης μπορεί να προστεθεί η εντολή break;

```
switch(μεταβλητή){
        case « περίπτωση τιμής μεταβλητής»:
                εντολή 1;
        break;
        …
}
```

Η συγκεκριμένη εντολή προκαλεί άμεση έξοδο από την switch. Έτσι αν μετά την εκτέλεση μίας περίπτωσης δεν υπάρχει λόγος να συνεχίσει να συγκρίνει την μεταβλητή με άλλες περιπτώσεις, τότε χρησιμοποιώντας την εντολή break; μπορεί το πρόγραμμα να διαφύγει από το block της switch.

2.14 Βρόχοι

For

Ο βρόχος for επιτρέπει στο arduino να επαναλάβει εκτελέσιμες γραμμές κώδικα που βρίσκονται μέσα σε block για συγκεκριμένο αριθμό. Αυτό που κάνει την εντολή for μοναδική είναι ότι βασίζεται σε έναν μετρητή ο οποίος αυξάνεται κάθε φορά που ο βρόχος επαναλαμβάνεται. Ο μετρητής έχει την δυνατότητα να χρησιμοποιηθεί και μέσα στον βρόχο όπως μία τοπική μεταβλητή. Η βασική σύνταξη του βρόχου for είναι:

```
for (ορισμός μεταβλητής ; έλεγχος μεταβλητής ; αύξηση μεταβλητής)
{
        εντολές
}
```

Η αρχή του βρόχου αποτελείται από τρία μέρη, τον ορισμό μίας μεταβλητής, την συνθήκη που θα ελέγξει τη μεταβλητή και τον τρόπο με τον οποίο θα αυξάνεται η μεταβλητή. Στο πρώτο μέρος ορίζουμε ή αρχικοποιούμε την μεταβλητή, η εντολή αυτή εκτελείται μόνο την πρώτη φορά. Στο δεύτερο μέρος έχουμε μία λογική έκφραση η οποία επιστρέφει μία τιμή boolean. Αν ο έλεγχος

επιστρέψει true τότε συνεχίζεται η εκτέλεση του βρόχου, διαφορετικά σταματάει. Στο τρίτο μέρος αυξάνει την τιμή της μεταβλητής κάθε φορά που ολοκληρώνεται η εκτέλεση των εντολών του βρόχου. Ένα παράδειγμα του βρόχου for είναι:

```
for(int i = 0; i <5; i++){
……
}
```

Στο πρώτο μέρος ορίζουμε μία καινούρια integer μεταβλητή και της εκχωρούμε την τιμή 0 (int i = 0;). Στην συνέχεια ελέγχουμε αν η μεταβλητή «i» είναι μικρότερη από το 5 για να συνεχιστεί η εκτέλεση του βρόχου (i < 5;). Ενώ στο τρίτο μέρος αυξάνουμε την τιμή της μεταβλητής κατά μία μονάδα, κάθε φορά που ο βρόχος επαναλαμβάνεται (i++). Έτσι το εσωτερικό που βρόχου θα εκτελεστεί 5 φορές εάν δεν υπάρξει κάποια προσαύξηση ή μείωση της μεταβλητής «i» μέσα στο block.

While

Ο βρόχος while χρησιμοποιείται για να επαναλάβει ένα block για όσο χρόνο μία συγκεκριμένη συνθήκη είναι true. Ακολουθεί η σύνταξη του βρόχου while:

```
while(συνθήκη) {
        εντολές;
}
```

Το προηγούμενο παράδειγμα θα μπορούσε να ξαναγραφεί χρησιμοποιώντας τη while:

```
int i = 0;
while( i < 5 ){
        ...
        i++;
}
```

Στην πρώτη γραμμή ορίζεται η integer μεταβλητή i = 0. Στην επόμενη γραμμή δημιουργείται ο βρόχος while και συγκρίνει αν η μεταβλητή είναι μικρότερη του 5. Στην περίπτωση που η συνθήκη επιστρέψει true θα εκτελεστεί ο κώδικας μέσα στο block, ενώ στην τελευταία γραμμή του block θα αυξηθεί η τιμή της μεταβλητής κατά μία μονάδα. Τέλος θα επιστρέψει στην αρχή του βρόχου while, θα επαναλάβει τον έλεγχο της συνθήκης. Μέχρι αυτή να επιστρέψει false θα επαναλαμβάνεται η εκτέλεση των εντολών μέσα στο block.

Do

Οι βρόχοι for και while αξιολογούν μία συνθήκη πριν εκτελέσουν τις εντολές που βρίσκονται μέσα στο block. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη τότε το block δε θα τρέξει ποτέ. Μερικές φορές είναι χρήσιμο ο κώδικας που βρίσκεται μέσα στο block να εκτελείται τουλάχιστον μία φορά. Αυτό επιτυγχάνεται χρησιμοποιώντας το βρόχο do, ο οποίος ενεργεί σαν μία while με την διαφορά ότι πρώτα εκτελεί τον κώδικα που βρίσκεται μέσα στο block και στην

συνέχεια ελέγχει αν η συνθήκη επιστρέφει true με σκοπό να τον επαναλάβει. Η βασική της σύνταξη είναι:

```
do {
        εντολές;
} while( συνθήκη);
```

Continue

Η εντολή continue; χρησιμοποιείται μέσα σε βρόχο με σκοπό την έναρξη της επόμενης επανάληψης του. Δηλαδή όταν το πρόγραμμα διαβάσει την εντολή continue; Θα σταματήσει την εκτέλεση των εντολών που ακολουθούν μέσα στο βρόχο και θα επιστρέψει στην αρχή. Εκεί θα ελέγξει αν η συνθήκη συνεχίσει να επιστρέφει true ώστε να επαναλάβει τον κώδικα μέσα στο βρόχο ακόμη μία φορά.

2.15 Συναρτήσεις

2.15.1 Βασικές συνάρτησης setup και loop

Τα προγράμματα που αναπτύσσονται για να εφαρμοστούν στο arduino εκτός της βασικής βιβλιοθήκης arduino.h (που πλέον δεν χρειάζεται δήλωση) απαιτούν και δύο συναρτήσεις, την setup() και την loop(). Κάθε sketch πρέπει να περιλαμβάνει αυτές τις δύο συναρτήσεις ώστε να μπορέσει να τρέξει η εφαρμογή, έστω και αν είναι άδειες. Το επόμενο παράδειγμα είναι τεχνικά ένα ολοκληρωμένο sketch που τρέχει κανονικά, αν και δεν κάνει απολύτως καμία ενέργεια.

```
void setup(){
}
void loop(){
}
```

Η **setup()** καλείται μόνο μία φορά κατά την έναρξη του sketch και χρησιμοποιείται με σκοπό να πραγματοποιήσει διάφορες εντολές, όπως είναι η έναρξη μίας σειριακής επικοινωνίας με μία σειριακή συσκευή, η εκχώρηση αρχικών τιμών σε αισθητήρες ή να εκτελέσει ορισμένες εργασίες που απαιτούνται για την ομαλή λειτουργία του προγράμματος.

Η **loop()** δίνει την δυνατότητα στο sketch να τρέχει επαναλαμβανόμενα έως ότου να σταματήσει η παροχή του ρεύματος στο arduino. Η έναρξη της σηματοδοτείται με το που εκτελέσει όλες τις εντολές τις η setup(). Με το που θα εκτελεστεί η τελευταία εντολή της συνάρτησης loop(), τότε το πρόγραμμα θα ξεκινήσει και πάλι να εκτελεί τις εντολές της loop() από την πρώτη γραμμή του block.

2.15.2 Δημιουργία νέων συναρτήσεων

Σε ένα sketch, ο προγραμματιστής εκτός από τις ήδη υπάρχουσες συναρτήσεις που βρίσκονται στις βιβλιοθήκες, του δίνεται η δυνατότητα να χρησιμοποιήσει και δικές του. Αν στο πρόγραμμα επαναλαμβάνονται κάποιες γραμμές κώδικα, τότε ο προγραμματιστής θα μπορούσε

να δημιουργήσει μία νέα συνάρτηση και να συμπεριλάβει μέσα σε αυτήν τον επαναλαμβανόμενο κώδικα. Έτσι κάθε φορά που απαιτούνται οι συγκεκριμένες γραμμές κώδικα, αντί να τις γράφει επανειλημμένα μπορεί απλά και εύκολα να καλεί την συνάρτηση που δημιούργησε.

Για την δημιουργία μίας συνάρτησης απαιτείται να δηλωθούν ο τύπος της τιμής που θα επιστρέφει, το όνομα της συνάρτησης και το σύνολο των παραμέτρων που απαιτούνται για την εκτέλεση των εντολών μέσα στη συνάρτηση. Ο ορισμός μίας συνάρτησης τηρεί το ακόλουθο σχήμα:

```
«τύπος επιστρεφόμενης τιμής» «όνομα συνάρτησης» («λίστα παραμέτρων»)
{
        «εντολές συνάρτησης»
}
```

Η κάθε συνάρτηση που προστίθεται στο sketch πρέπει να δημιουργείται έξω από τις υπόλοιπες συναρτήσεις που υπάρχουν. Τις περισσότερες φορές δεν παίζει κάποιο ρόλο σε πιο ακριβώς σημείο θα τοποθετηθούν αλλά συνηθίζεται οι καινούριες συναρτήσεις να μπαίνουν αμέσως μετά το τέλος της συνάρτησης loop().

2.15.3 Καλώντας τις συναρτήσεις

Μία συνάρτηση καλείται όταν ο προγραμματιστής θέλει να εκτελέσει ένα σύνολο ομαδοποιημένων εντολών που βρίσκονται μέσα σε αυτήν. Η συνάρτηση καλείται με την ακόλουθη μορφή:

```
«όνομα συνάρτησης» («παράμετροι»);
```

Όταν καλείται η συνάρτηση τότε το πρόγραμμα πηδάει από το σημείο που καλείται στο σημείο που δημιουργήθηκε για να εκτελέσει της εντολές που βρίσκονται μέσα σε αυτήν. Μόλις το πρόγραμμα τελειώσει όλες τις εντολές που βρίσκονται μέσα στην συνάρτηση, τότε θα συνεχίσει από το σημείο που συνάντησε το κάλεσμα της συνάρτηση.

2.15.4 Επιστροφές τιμών συνάρτησης

Σε αρκετές περιπτώσεις ο σκοπός της συνάρτησης είναι να εκτελέσει διάφορες πράξεις και να επιστρέψει ένα αποτέλεσμα σε μία κατάλληλη μορφή ώστε το πρόγραμμα να την χρησιμοποιήσει κάπου αλλού. Η ενέργεια του καλέσματος μίας συνάρτησης και η επιστροφή μίας τιμής από την ίδια συνάρτηση είναι γνωστή ως **function return**.

Στην δήλωση της συνάρτησης το πρώτο πράγμα που χρειάζεται να δηλωθεί είναι ο τύπος της τιμής που θα επιστραφεί. Επίσης μέσα στην συνάρτηση θα χρησιμοποιηθεί η εντολή return «όνομα μεταβλητής»; η οποία επιστρέφει την τιμή της μεταβλητής στο σημείο που καλέστηκε η συνάρτηση. Ο τύπος της συνάρτησης θα πρέπει να είναι ο ίδιος με τον τύπο της μεταβλητής που επιστρέφεται. Στην περίπτωση που η συνάρτηση δεν επιστρέφει καμία τιμή τότε η συνάρτηση δηλώνεται ως void.

2.15.5 Παράμετροι συνάρτησης

Οι παράμετροι συνάρτησης είναι ένα είδος σαν τις μεταβλητές αλλά δηλώνονται μέσα στις παρενθέσεις όπου ορίζεται η συνάρτηση και χωρίζονται με κόμμα. Αν απαιτούνται κάποιες μεταβλητές για τις πράξεις μέσα στην συνάρτηση, των οποίων η εμβέλεια τους δεν επιτρέπει τη χρήση τους, μπορούν να δηλωθούν ως παράμετροι συνάρτησης. Έτσι κατά το κάλεσμα της συνάρτησης μέσα στις παρενθέσεις μπορούν να εισαχθούν τα απαιτούμενα στοιχεία και στη συνέχεια με το που διαβάσει το πρόγραμμα το όρισμα της συνάρτησης, αντιστοιχεί τα στοιχεία με νέες μεταβλητές τοπικής εμβέλειας.

Ένα παράδειγμα δημιουργίας και κάλεσμα συνάρτησης:

```
void loop(){
        int a = 5;
        int b = 4;
        int c;
        c = add(a, b);
}
int add(int num1, int num2){
        return num1+num2;
}
```

Στην πρώτη σειρά εκτελείται επανειλημμένα η συνάρτηση loop. Μέσα στο block της loop, ορίζονται τρεις τοπικές μεταβλητές a b c και εκχωρούνται οι τιμές 5, 4 στις δύο πρώτες. Αφού οριστούν οι μεταβλητές μετά καλείται η συνάρτηση add με παραμέτρους τις μεταβλητές a, b. Έτσι η τιμή που θα επιστραφεί από την συνάρτηση add() θα εκχωρηθεί στην μεταβλητή c. Όταν το πρόγραμμα συναντήσει το κάλεσμα της συνάρτησης αμέσως πηγαίνει στο σημείο όπου ορίζεται.

Η συνάρτηση που καλείται ορίζεται ως int επειδή θα επιστρέψει έναν ακέραιο αριθμό. Το όνομα της είναι add και μέσα στην παρένθεση θα δημιουργήσει δύο νέες μεταβλητές τοπικής εμβέλειας. Στις μεταβλητές αυτές (num1 και num2) θα καταχωρηθούν οι τιμές που αντιστοιχούν στις παραμέτρους που βρίσκονται στο κάλεσμα της συνάρτησης. Στην τελευταία σειρά επιστρέφεται η τιμή που αντιστοιχεί στο άθροισμα των δύο τοπικών μεταβλητών. Έτσι η συνάρτηση add επιστρέφει μία τιμή που στην συνέχεια εκχωρείται στην μεταβλητή c.

2.16 Ψηφιακές συναρτήσεις

pinMode()

Πριν χρησιμοποιηθεί μία ψηφιακή θύρα (digital pin) θα πρέπει πρώτα το arduino να ενημερωθεί με ποιον τρόπο, δηλαδή εάν η θύρα θα είναι εισόδου ή εξόδου. Για να γίνει αυτό καλείται η συνάρτηση pinMode() η οποία ενημερώνει τον μικροελεγκτή για τον αριθμό της θύρας που θα χρησιμοποιηθεί καθώς και την κατάσταση της (είσοδος/έξοδος). Η σύνταξη είναι αρκετά απλή:

```
pinMode(«αριθμός θύρας», «κατάσταση»);
```

Μέσα στις παρενθέσεις υπάρχουν δύο παράμετροι οι οποίες χωρίζονται με κόμμα. Η πρώτη αντιστοιχεί στον αριθμό της θύρας που ο προγραμματιστής επιθυμεί να χρησιμοποιήσει. Η τιμή αυτή μπορεί να κυμαίνεται από το 0 έως το 13 ή από το A0 μέχρι το A5 στην περίπτωση που επιθυμεί να χρησιμοποιήσει της αναλογικές εισόδους ως ψηφιακές εισόδους/εξόδους. Η δεύτερη τιμή ορίζει την κατάσταση που θα δουλεύει στο κύκλωμα η συγκεκριμένη θύρα. Αυτό επιτυγχάνεται χρησιμοποιώντας τις δύο προκαθορισμένες από το arduino σταθερές, την INPUT και την OUTPUT. Όταν η θύρα ορίζεται ως INPUT, τότε δέχεται εισερχόμενα σήματα και καταλαβαίνει μόνο δύο καταστάσεις, την HIGH και την LOW. Αντίστοιχα όταν η θύρα ορίζεται ως OUTPUT τότε έχει την δυνατότητα να παρέχει στην έξοδο έως και 40 milliamperes σε ένα άλλο κύκλωμα, ισχύς που είναι αρκετή για να ανάψει ένα LED.

digitalRead()

Όταν μία ψηφιακή θύρα ρυθμιστεί ως INPUT (εισόδου) τότε για να διαβαστεί η είσοδος της χρησιμοποιείται η συνάρτηση digitalRead(). Η σύνταξη της είναι:

digitalRead(«αριθμός θύρας»);

Η συγκεκριμένη συνάρτηση χρησιμοποιεί μόνο μία παράμετρο που αντιστοιχεί στον αριθμό της θύρας εισόδου. Η συνάρτηση σύμφωνα με την είσοδο της, επιστρέφει μία τιμή HIGH ή LOW. Η τιμή αυτή μπορεί να αποθηκευτεί ή να συγκριθεί με μία άλλη μεταβλητή.

digitalWrite()

Αν μία θύρα έχει οριστεί ως OUTPUT τότε η ρύθμιση της εξόδου της σε on (+5 Volt) ή off (0 Volt) κατάσταση πραγματοποιείται χρησιμοποιώντας την συνάρτηση digitalWrite(). Η σύνταξη της είναι:

digitalWrite(«αριθμός θύρας», «κατάσταση θύρας»);

Εδώ όπως και στην pinMode() υπάρχουν δύο παράμετροι που απαιτούνται για την συνάρτηση. Η πρώτη δηλώνει τον αριθμό της θύρας που παίρνει τιμές από 0 έως 13 ή από A0 έως A5. Η δεύτερη τιμή αντιστοιχεί στις δύο καταστάσεις που μπορεί να ρυθμιστεί η έξοδος χρησιμοποιώντας δύο προκαθορισμένες μεταβλητές του arduino, την HIGH και την LOW. Η κατάσταση HIGH παρέχει στην έξοδο την +5VDC τάση της πλακέτας ενώ η LOW συνδέει το κύκλωμα που βρίσκεται στην έξοδο της θύρας με την γείωση (0 Volt).

2.17 Αναλογικές συναρτήσεις

Οι αναλογικές συναρτήσεις σε αντίθεση με τις ψηφιακές, δε χρειάζεται το arduino να ενημερωθεί για τον αριθμό της αναλογικής θύρας που πρόκειται να χρησιμοποιηθεί όπως συμβαίνει με τη συνάρτηση pinMode(). Επιπλέον αντί για τιμές HIGH και LOW χρησιμοποιούν ακέραιους αριθμούς εύρους από το 0 έως το 1023 για τις τιμές εισόδου, και από 0 έως 255 για τις τιμές εξόδου.

analogRead()

Η analogRead() λειτουργεί παρόμοια με την digitalRead. Η σύνταξη της συνάρτησης είναι:

analogRead(«αριθμός θύρας»);

Όταν καλείται η παραπάνω συνάρτηση, απαιτείται να προσδιοριστεί ο αριθμός της αναλογικής θύρας από όπου αναμένεται να διαβαστούν τα δεδομένα. Η παράμετρος της analogRead παίρνει τιμές από το A0 έως το A5. Το γράμμα A μπαίνει μπροστά από τον αριθμό της θύρας με σκοπό να υπενθυμίσει τον προγραμματιστή ότι πρόκειται για Analog θύρα. Αφού η συνάρτηση διαβάσει το εισερχόμενο σήμα τότε επιστρέφει μία τιμή χωρητικότητας 10 bit που αντιπροσωπεύει έναν ακέραιο αριθμό εύρους από το 0 έως το 1023. Η τιμή που επιστρέφεται αντιστοιχεί στην στάθμη του αναλογικού σήματος εισόδου της θύρας.

analogWrite()

Η συνάρτηση analogWrite() επιτρέπει την πρόσβαση στο pulse width modulation (PWM) hardware του μικροελεγκτή στο arduino. Η βασική σύνταξη της συνάρτησης είναι:

analogWrite(«αριθμός θύρας», «κατάσταση παλμού»);

Χρησιμοποιώντας αυτή την συνάρτηση απαιτούνται δύο παράμετροι. Η πρώτη παράμετρος ορίζει τον αριθμό της αναλογικής θύρας εξόδου. Για το arduino UNO η πρώτη παράμετρος μπορεί να πάρει μόνο τις ακόλουθες τιμές 3, 5, 6, 9, 10 και 11 οι οποίες αντιστοιχούν σε θύρες που έχουν την ένδειξη PWM πάνω στην πλακέτα. Άλλες εκδόσεις arduino μπορεί να έχουν περισσότερες ή λιγότερες PWM θύρες. Η δεύτερη παράμετρος ορίζει την κατάσταση του παλμού η οποία εκφράζεται με έναν integer αριθμό εύρους από το 0 έως το 255 καταλαμβάνοντας χωρητικότητα 8 bit. Όταν η τιμή ισούται με το 0 τότε ο παλμός βρίσκεται στη κατάσταση off ή 0% ή 0 volt καθ' όλη την διάρκεια της περιόδου. Ανάλογα όταν η τιμή ισούται με το 255 τότε σε αυτήν την περίπτωση ο παλμός βρίσκεται στη κατάσταση on ή 100% ή 5 volt καθ΄ όλη την διάρκεια της περιόδου. Αν όμως η τιμή βρίσκεται στο μέσω μεταξύ των τιμών 0 και 255 τότε σε κάθε περίοδο ο παλμός θα βρίσκεται στην κατάσταση off κατά την μισή διάρκεια της περιόδου και στην κατάσταση on στην υπόλοιπη διάρκεια.

analogReference()

Χρησιμοποιώντας την συνάρτηση analogRead(), το arduino είναι προετοιμασμένο να δεχτεί ένα εύρος τάσης από 0 έως +5 volt. Όμως όλο και περισσότεροι αισθητήρες χρησιμοποιούν χαμηλότερη τάση λειτουργίας. Έτσι όταν ένας αισθητήρας στέλνει μέγιστη τιμή μικρότερη των 5 volt τότε το arduino (που περιμένει μέγιστη τιμή 5 volt) θα μεταφράσει λανθασμένα την πληροφορία του αισθητήρα.

Ο μικροελεγκτής της ATMEL έχει την δυνατότητα να αλλάξει την τάση αναφοράς καλώντας την συνάρτηση analogReference(). Η σύνταξη της συνάρτησης είναι:

analogReference(«επιλογή αναφοράς»);

Μέσα στις παρενθέσεις ορίζεται το σημείο αναφοράς που θα έχει ο μικροελεγκτής. Υπάρχουν τρεις επιλογές τιμής αναφοράς οι οποίες είναι:

- **DEFAULT**. Επιλέγοντας την προκαθορισμένη τιμή, το arduino χρησιμοποιεί σαν τάση αναφοράς τα 5 volt (ή τα 3.3 volt ανάλογα τον τύπο του arduino).
- **EXTERNAL**. Η πλακέτα του arduino διαθέτει ένα pin διπλά από το pin 13 που ονομάζεται AREF. Έτσι όταν καλείται η συνάρτηση analogReference(EXTERNAL); η θύρα AREF χρησιμοποιείται με σκοπό ο μικροελεγκτής να διαβάσει μία εξωτερική τάση αναφοράς διαφορετική από την προκαθορισμένη που είναι τα 5 volt. Και πάντα μέσα στο όριο 0 - 5 volt.
- **INTERNAL**. Δηλώνοντας την παράμετρο INTERNAL, το arduino χρησιμοποιεί ως τάση αναφοράς τα 1,1 volt.

Η συνάρτηση αυτή εκτελείται μόνο μία φορά στο sketch, πριν την χρήση της συνάρτησης analogRead(). Συνηθίζεται να καλείται μέσα στην συνάρτηση setup().

map()

Το εύρος της τιμής που επιστρέφει μία αναλογική θύρα είναι αρκετά μεγάλο (0 έως 1023). Το γεγονός αυτό πολλές φορές δυσκολεύει τον έλεγχο της τιμής εισόδου, έτσι απαιτείται η μείωση του εύρους της με σκοπό την διευκόλυνση της επεξεργασία της. Ένας άλλος λόγος μείωσης του εύρους είναι η εξοικονόμηση αποθηκευτικού χώρου που απαιτεί μία μεταβλητή από την μνήμη. Η συνάρτηση map() επιστρέφει μία νέα τιμή σύμφωνα με το επιθυμητό εύρος. Η σύνταξη της είναι:

map(«αρχική τιμή», «ελ. τιμή αρχικού εύρους», «μεγ. τιμή αρχικού εύρους», «ελ. τιμή νέου εύρους», «μεγ. τιμή νέου εύρους»);

Η map() όταν καλείται απαιτεί 5 παραμέτρους. Η πρώτη αντιστοιχεί σε μία τιμή που ανήκει στο αρχικό επιτρεπτό εύρος τιμών και επιθυμούμε να μετατραπεί σε μία νέα τιμή σύμφωνα με ένα νέο εύρος. Η δεύτερη και η τρίτη παράμετρος ορίζει την ελάχιστη και την μεγίστη επιτρεπτή τιμή του αρχικού εύρους. Η τέταρτη και η πέμπτη παράμετρος ορίζει την ελάχιστη και τη μέγιστη τιμή του νέου εύρους τιμών. Η map() στην ουσία επιστρέφει μία νέα τιμή η οποία ανήκει σε ένα νέο εύρος τιμών και είναι αντίστοιχη της αρχικής τιμής (πρώτη παράμετρος) που ανήκει στο αρχικό εύρος τιμών.

constrain()

Η συνάρτηση constrain() ελέγχει μία τιμή εάν βρίσκεται μέσα στα όρια ενός εύρους τιμών. Στην περίπτωση που η τιμή δεν ανήκει μέσα στα όρια των επιτρεπτών τιμών τότε επιστρέφεται η ελάχιστη ή η μεγίστη επιτρεπτή τιμή, ανάλογα εάν η τιμή είναι μικρότερη ή μεγαλύτερη από τα επιθυμητά όρια. Όταν όμως η τιμή ανήκει μέσα στα όρια τότε επιστρέφεται χωρίς κάποια αλλαγή. Η σύνταξη της είναι:

constrain(«τιμή», «ελαχ. επιτρεπτή τιμή», «μεγ. επιτρεπτή τιμή»);

Η συνάρτηση χρησιμοποιεί τρεις παραμέτρους. Η πρώτη παράμετρος είναι η τιμή που θα ελεγχθεί. Η δεύτερη είναι το κατώτατο και η τρίτη το ανώτατο όριο της επιτρεπόμενης τιμής. Η συνάρτηση επιστρέφει πάντα μία τιμή που ανήκει μέσα στα όρια που ορίστηκαν στη συνάρτηση.

2.18 Προχωρημένες συναρτήσεις

Μέχρι τώρα εκτός από τη δομή και την σύνταξη ενός προγράμματος αναφέρθηκαν και οι βασικές συναρτήσεις που χρησιμοποιούνται για την ανάγνωση και εγγραφή ψηφιακών και αναλογικών δεδομένων εισόδου και εξόδου. Εκτός όμως από αυτές τις συναρτήσεις, η βιβλιοθήκη του arduino περιλαμβάνει και άλλες εξίσου σημαντικές και χρήσιμες οι οποίες αναλύονται στη συνέχεια.

2.18.1 Συναρτήσεις Timing

Επειδή ο μικροελεγκτής του arduino μπορεί να «κινηθεί» πιο γρήγορα οπ' όσο τουλάχιστον μπορεί να αντιληφθεί ο άνθρωπος, είναι αρκετές φορές σκόπιμο να επιβραδυνθούν οι ενέργειες που πραγματοποιεί. Αυτό επιτυγχάνεται καλώντας τις κατάλληλες συναρτήσεις.

delay()

Η συνάρτηση delay δημιουργεί μία μικρή παύση στην ροής του προγράμματος. Η σύνταξη της είναι:

delay(«χρόνος»);

Η delay έχει μία παράμετρο η οποία αντιστοιχεί στον χρόνο που θα «παγώσει» την ροή του προγράμματος. Η μονάδα μέτρησης του χρόνου που χρησιμοποιείται είναι τα milliseconds. Έτσι όταν το πρόγραμμα καλέσει την εντολή delay(1000); τότε η ροή του θα σταματήσει για 1 δευτερόλεπτο, αφού 1 second ισούται με 1000 milliseconds.

delayMicroseconds()

Στην περίπτωση που η delay() κάνει μεγάλης διάρκειας παύσεις, τότε μπορεί να χρησιμοποιηθεί η συνάρτηση delayMicroseconds(). Όπως και στην delay() έτσι και η delayMicroseconds() συνάρτηση έχει ακριβώς την ίδια σύνταξη με την διαφορά ότι η παράμετρος του χρόνου αντιστοιχεί σε microseconds. Ένα second ισούται με 1,000,000 microseconds.

millis()

Μέσα στον μικροελεγκτή της πλακέτας arduino υπάρχουν τρεις on-board hardware μετρητές (timers) οι οποίοι εκτελούνται πίσω από το κυρίως πρόγραμμα με σκοπό να χειριστούν επαναλαμβανόμενες εργασίες όπως είναι η αύξηση της τιμής ενός χρονόμετρου. Η συνάρτηση millis() χρησιμοποιεί έναν από τους τρεις hardware timers που περιέχει τον χρόνο από το σημείο που ο microcontroller τέθηκε σε λειτουργία, είτε μετά από διακοπή ρεύματος, είτε μετά από reset. Η σύνταξη της δεν περιέχει κάποια παράμετρο, έτσι καλώντας απλά την συνάρτηση millis(); αυτή θα επιστρέψει μία τιμή σε milliseconds. Η τιμή αυτή μπορεί να εκχωρηθεί σε μία μεταβλητή και να χρησιμοποιηθεί όπως μία οποιαδήποτε άλλη μεταβλητή.

macros()

Όπως η συνάρτηση millis() επιστρέφει τον χρόνο λειτουργίας του μικροελεγκτή σε milliseconds, έτσι και η συνάρτηση macros() κάνει ακριβώς το ίδιο πράγμα με την διαφορά ότι η μονάδα μέτρησης του χρόνου που επιστρέφεται είναι σε microseconds.

2.19 Συναρτήσεις random

Πολλές φορές στο πρόγραμμα απαιτείται η δημιουργία τυχαίων αριθμών, όπως για παράδειγμα για την κατασκευή ενός ηλεκτρονικού ζαριού. Το arduino παρέχει την δυνατότητα χρησιμοποίησης ψευδο-τυχαίων (semi-random) αριθμών καλώντας τις απαιτούμενες συναρτήσεις.

random()

Η συνάρτηση random() επιστρέφει μία ψευδο-τυχαία (semi-random) τιμή, σύμφωνα με τις παραμέτρους. Αν δεν υπάρχουν παράμετροι τότε θα επιστρέψει μία τιμή τύπου long εύρους από το -2,147,483,648 έως το 2,147,483,647. Η κανονική σύνταξη της είναι: random(«ελάχιστη τιμή», «μεγίστη τιμή»);

Η πρώτη παράμετρος ορίζει την ελάχιστη επιτρεπτή τιμή που μπορεί να επιστρέψει, ενώ αντίστοιχα η δεύτερη παράμετρος την μέγιστη. Στην περίπτωση που η ελάχιστη τιμή είναι το μηδέν τότε η πρώτη παράμετρος μπορεί να παραληφθεί. Έτσι το πρόγραμμα όταν δει μόνο μία παράμετρο, καταλαβαίνει ότι η ελάχιστη επιτρεπτή τιμή είναι το μηδέν ενώ η μεγίστη ισούται με την παράμετρο της συνάρτησης.

randomSeed()

Όπως αναφέρθηκε προηγουμένως οι τιμές που επιστρέφει η random() δεν είναι ακριβώς τυχαίες. Το γεγονός αυτό οφείλεται στον σχεδιασμό του arduino που είναι ένα αιτιατό σύστημα. Έτσι όταν χρησιμοποιείται η συνάρτηση, ο αλγόριθμος που παράγει τους τυχαίους αριθμούς επαναλαμβάνεται κάθε φορά που εκτελείται. Με αποτέλεσμα οι τιμές να είναι προβλεπόμενες αφού στην πραγματικότητα δεν είναι τυχαίες. Για να αποφευχθεί η ανάκτηση των ίδιων ψευδο-τυχαίων αριθμών χρησιμοποιείται η συνάρτηση randomSeed(), η σύνταξη της είναι:
randomSeed(«τιμή»);

Τροφοδοτώντας την συνάρτηση με μία τιμή, μπορεί να κάνει την παραγωγή ψευδο-τυχαίων αριθμών αρκετά πιο απρόβλεπτη. Η τιμή μπορεί να αντιστοιχεί σε έναν integer ή ένα long αριθμό που ορίζεται μέσα στο πρόγραμμα. Για να δημιουργηθούν ακόμη πιο τυχαίοι αριθμοί, αντί να οριστεί η τιμή, μπορεί κάλλιστα να εισαχθεί από μία αναλογική θύρα (analog pin) χρησιμοποιώντας έναν αισθητήρα. Για παράδειγμα:
randomSeed(analogRead(A0));

2.20 Hardware Διακοπές

Οι λόγοι για να ζητήσουμε από το πρόγραμμα να διακόψει την ροή της εκτέλεσης του είναι πολλοί. Ένας λόγος είναι ότι η συνάρτηση loop() μπορεί να περιέχει πολλές γραμμές κώδικα με

αποτέλεσμα να χρειάζεται αρκετό χρόνο για μία πλήρη επανάληψη των εντολών που περιέχει, έτσι είναι πιθανόν όταν πατηθεί ένα button να μην προλάβει να γίνει αντιληπτή η ενέργεια. Ένα άλλο παράδειγμα είναι όταν χρησιμοποιείται ένας φωτοηλεκτρικός αισθητήρας ο οποίος αντιλαμβάνεται ένα αντικείμενο όταν πλησιάζει. Οι αποφάσεις του arduino για τις εντολές που πρέπει να δώσει στους κινητήρες θα πρέπει να ληφθούν άμεσα και με ακρίβεια.

Η συνάρτηση **attachInterrupt()** ενεργοποιεί μία διακοπή της κανονικής ροής του κώδικα που εκτελεί το arduino με σκοπό να τρέξει μία συγκεκριμένη συνάρτηση. Η συνάρτηση καλείται όταν σκανδαλιστεί ο μικροελεγκτής σε ένα συγκεκριμένο pin. Η σύνταξη της είναι:
attachInterupt(«διακόπτης», «συνάρτηση», «κατάσταση»);

Η πρώτη παράμετρος ορίζει τον διακόπτη που θα σκανδαλίσει (δώσει το ερέθισμα) στον microcontroller. Στο arduino UNO υπάρχει η δυνατότητα για την χρήση δύο διακοπτών, συγκεκριμένα ο αριθμός 0 δηλώνει ότι ο διακόπτης αντιστοιχεί στο pin 2 και ο αριθμός 1 αντιστοιχεί στο pin 3. Η δεύτερη παράμετρος δηλώνει την συνάρτηση που θα εκτελεστεί όταν καλεστεί η attachInterrupt(). Τέλος η τρίτη παράμετρος δηλώνει με πιο τρόπο θα σκανδαλίσει ο διακόπτης τον μικροελεγκτή.
Υπάρχουν τέσσερις πιθανές καταστάσεις.

1.LOW: σκανδαλίζει όταν ο διακόπτης είναι σε κατάσταση LOW (0 volt).

2.CHANGE: σκανδαλίζει όταν ο διακόπτης αλλάζει κατάσταση, είτε από LOW (0 volt) πηγαίνει σε HIGH (+5 volt) κατάσταση, είτε το αντίθετο.

3.RISING: σκανδαλίζει όταν ο διακόπτης από την κατάσταση LOW αλλάζει σε κατάσταση HIGH.

4.FALLING: σκανδαλίζει όταν ο διακόπτης από την κατάσταση HIGH πηγαίνει στη κατάσταση LOW.

Κατά την διάρκεια εκτέλεσης του προγράμματος και ενώ είναι ενεργοποιημένη η συνάρτηση διακοπής, δίνεται η δυνατότητα αλλαγής της κατάστασης με την οποία θα σκανδαλίζεται η διακοπή. Καλώντας την συνάρτηση **detachInterrupt()** θα απενεργοποιηθεί η συνάρτηση διακοπής. Έτσι αφού διακοπεί, η attachInterupt() μπορεί να ξαναεκτελεστεί ορίζοντας τον επιθυμητό τρόπο σκανδαλισμού. Η συνάρτηση detachInterrupt() έχει μία παράμετρο που ορίζει τον διακόπτη που θα απενεργοποιήσει, 0 για το διακόπτη στην θέση pin 2 και 1 για το διακόπτη στη θέση pin 3.

2.21 Βιβλιοθήκες Arduino

2.21.1 Βασικές Βιβλιοθήκες

Η πλατφόρμα Arduino συνοδεύεται μαζί με μερικές χρήσιμες βιβλιοθήκες όπου κάθε μία από αυτές παρέχει κάποια υπηρεσία είτε σε επίπεδο λογισμικού είτε σε επίπεδο υλικού. Επίσης,

στο Διαδίκτυο μπορούν να βρεθούν και αρκετές βιβλιοθήκες, που αναπτύσσονται κυρίως από προγραμματιστές που στηρίζουν το Arduino. Παρακάτω ακολουθούν μερικές βασικές βιβλιοθήκες:

- **EEPROM:** Διαχείριση της μνήμης EEPROM (ανάγνωση και εγγραφή σε μόνιμη αποθήκευση).
- **Ethernet:** Εφαρμόζεται για τη σύνδεση στο διαδίκτυο χρησιμοποιώντας Arduino Ethernet Shield.
- **Firmata:** Χρησιμοποιείται για την επικοινωνία με τις εφαρμογές του υπολογιστή χρησιμοποιώντας ένα standard σειριακό πρωτόκολλο.
- **LiquidCrystal:** Εφαρμόζεται για τον έλεγχο τυπικών οθονών υγρών κρυστάλλων (LCD).
- **SD:** Χρησιμοποιείται για τη διαχείριση, τον έλεγχο, την ανάγνωση και την εγγραφή σε κάρτες SD.
- **Servo:** Χρησιμοποιείται για τον έλεγχο των σερβοκινητήρων.
- **SPI:** Εφαρμόζεται για την επικοινωνία με συσκευές που χρησιμοποιούν το Serial Peripheral Interface (SPI) Bus.
- **SoftwareSerial:** Χρησιμοποιείται για την σειριακή επικοινωνία οποιασδήποτε ψηφιακής εισόδου.
- **Stepper:** Χρησιμοποιείται για τον έλεγχο των βηματικών κινητήρων.
- **Wire:** Η διεπαφή δύο καλωδίων (TWI/I2C) έχει την δυνατότητα αποστολής και λήψης των δεδομένων εκτός των συσκευών και των αισθητήρων.
- Επιπλέον βιβλιοθήκες που χρησιμοποιήθηκαν στην παρούσα πτυχιακή εργασία είναι οι: **DallasTemperature, DS1302, FrequencyTimer2, IRremote, Keypad, OneWire, TimerOne, Ultrasonic.**

2.21.2 Εισαγωγή βιβλιοθήκης

Για να χρησιμοποιηθεί μία βιβλιοθήκη πρέπει να ενημερωθεί το arduino από τον προγραμματιστή ποια είναι αυτή η βιβλιοθήκη. Υπάρχουν δύο τρόποι να γίνει αυτό. Ο πρώτος είναι να χρησιμοποιηθεί η εντολή #include πριν την εκτέλεση του κώδικα στην αρχή του sketch διευκρινίζοντας το όνομα της επιθυμητής βιβλιοθήκης που θα εισαχθεί στο πρόγραμμα. Η σύνταξη για την εισαγωγή μίας βιβλιοθήκης είναι:

```
#include<”name library”.h>
```

Το «name library» δηλώνει το όνομα του αρχείου της βιβλιοθήκης που επιθυμεί ο προγραμματιστής να χρησιμοποιήσει. Ο δεύτερος τρόπος για την εισαγωγή μίας βιβλιοθήκης είναι από την εργαλειοθήκη του arduino IDE. Κάνοντας κλικ την επιλογή sketch από το μενού και επιλέγοντας το Import Library. Ο προγραμματιστής έχει την δυνατότητα να επιλέξει μία από τις στάνταρτ ενσωματωμένες βιβλιοθήκες που περιλαμβάνει το arduino IDE. Στη συνέχεια το περιβάλλον είναι αυτό που αναλαμβάνει από μόνο του να εισάγει την βιβλιοθήκη στο sketch.

2.21.3 Σειριακή βιβλιοθήκη arduino

Το arduino χρησιμοποιείται σε πολλές stand-alone εφαρμογές ελέγχου οι οποίες δεν έχουν μεγάλες απαιτήσεις για την επεξεργασία δεδομένων, έτσι οι δυνατότητες του μικροελεγκτή είναι αρκετές για τον πλήρη έλεγχο. Σε άλλες όμως περιπτώσεις απαιτείται μεγαλύτερη υπολογιστική δύναμη με αποτέλεσμα να χρειάζεται τη βοήθεια ενός υπολογιστή. Για να επιτευχθεί η επικοινωνία του arduino με τον υπολογιστή γίνεται χρήση της σειριακής βιβλιοθήκης. Η συγκεκριμένη βιβλιοθήκη εισάγεται αυτόματα στο sketch μαζί με την κανονική βιβλιοθήκη του arduino, χωρίς να απαιτείται κάποια περαιτέρω εντολή εισαγωγής στο πρόγραμμα όπως συμβαίνει με τις υπόλοιπες βιβλιοθήκες. Όποτε καλείται μία από τις μεθόδους (συναρτήσεις) που ανήκουν στην Serial βιβλιοθήκη, θα προηγείται πάντα το όνομα της (Serial) πριν από τη συνάρτηση.

begin()

Με τη Serial βιβλιοθήκη να περιλαμβάνεται στο sketch το πρώτο πράγμα που χρειάζεται να πραγματοποιηθεί, για να μπορούν να χρησιμοποιηθούν οι Serial συναρτήσεις είναι να καθοριστεί η ταχύτητα μεταφοράς δεδομένων. Για να επιτευχθεί αυτό καλείται η μέθοδος begin(), η σύνταξη της είναι:

Serial.begin(«ρυθμός δεδομένων»);

Η παράμετρος αντιστοιχεί στο μέτρο ταχύτητας μεταφοράς bit ανά δευτερόλεπτο ανάμεσα στο arduino και τον υπολογιστή και μπορεί να πάρει μία από τις ακόλουθες τιμές 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200. Γενικά χρησιμοποιείται η τιμή 9600 που είναι αρκετά κοινή και γρήγορη ταχύτητα μεταξύ συσκευών σειριακής επικοινωνίας για τη μεταφορά δεδομένων.

available()

Η μέθοδος available() δεν έχει κάποια παράμετρο αλλά επιστρέφει τον αριθμό των διαθέσιμων byte που περιμένουν στο σειριακό buffer. Η hardware σειριακή θύρα του arduino περιέχει ένα buffer το οποίο μπορεί να αποθηκεύσει μέχρι και 128 byte πληροφορίες. Στην περίπτωση που καμία πληροφορία δεν περιέχεται στο buffer τότε η μέθοδος επιστρέφει την τιμή 0. Η σύνταξη της είναι:

Serial.available();

read()

Από την στιγμή που το πρόγραμμα γνωρίζει ότι το buffer περιέχει πληροφορίες, χρησιμοποιείται η μέθοδος read() για να τις διαβάσει. Η σύνταξη της είναι:

Serial.read();

Η read() επιστρέφει το πρώτο διαθέσιμο byte του buffer, στην πραγματικότητα διαβάζει μία ASCII τιμή από το 0 έως το 127 που αντιστοιχεί σε έναν ASCII χαρακτήρα. Τα byte λαμβάνονται ένα ένα κάθε φορά μέσω σειριακού hardware.

Η συνάρτηση Serial.read() περιμένει μέχρι να έρθουν τα δεδομένα και λαμβάνονται μόνο για όσο χρόνο καλείται η συνάρτηση. Δεδομένα που λαμβάνονται σε άλλη περίπτωση, χάνονται, αφού το ολοκληρωμένο δεν «ακούει».

print()

Η μέθοδος print() χρησιμοποιείται για να εμφανίσει ASCII χαρακτήρες στην συσκευή που είναι σειριακά συνδεμένη με το arduino, συνήθως μέσω του serial monitor στην οθόνη του υπολογιστή. Η σύνταξη της είναι:

```
Serial.print(«δεδομένα»);
```

Η παράμετρος που περιέχει τα δεδομένα μπορεί να δηλωθεί με διάφορους τρόπους.

println()

Η μέθοδος println() όπως και η print() χρησιμοποιείται για την εμφάνιση χαρακτήρων. Έχουν την ίδια σύνταξη και η μονή τους διαφορά είναι ότι κάθε φορά που εμφανίζει τα δεδομένα η println() στη συνέχεια προσθέτει έναν ειδικό χαρακτήρα νέας σειράς «/n».

2.22 Comments

Τα comments (σχόλια) είναι περιοχές κειμένων που αγνοούνται από το πρόγραμμα αλλά περιέχουν χρήσιμες σημειώσεις που δίνουν κάποιες πληροφορίες για την λειτουργία του sketch με σκοπό να κάνει τον κώδικα πιο κατανοητό από τους ανθρώπους. Τα comments δεν καταλαμβάνουν χώρο μέσα στη μνήμη του μικροελεγκτή, έτσι μπορούν να χρησιμοποιηθούν άφοβα. Υπάρχουν δύο τρόποι σύνταξης comments στον κώδικα του arduino. Ο πρώτος είναι:

```
/*
«comments»
*/
```

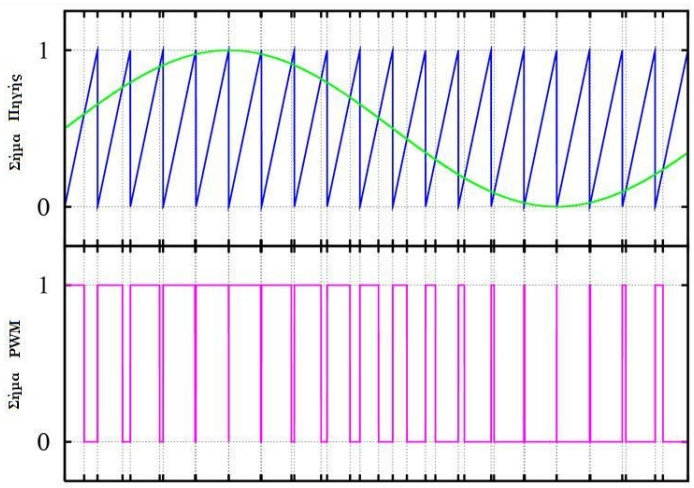
Ότι βρίσκεται ανάμεσα στους ειδικούς χαρακτήρες /* και */ αγνοείται. Τα σχόλια μπορούν να αποτελούνται από όσες γραμμές επιθυμεί ο προγραμματιστής. Ο δεύτερος τρόπος χρησιμοποιεί δύο slash «//». Όταν το πρόγραμμα συναντήσει δύο slash καταλαβαίνει ότι το κείμενο που ακολουθεί στα δεξιά είναι comments και τα αγνοεί. Για παράδειγμα:

```
delay(1000); // παύση προγράμματος για ένα sec
```

Το πρόγραμμα θα εκτελέσει τη συνάρτηση delay() κανονικά και δεν θα ασχοληθεί με το τι υπάρχει δεξιά των slash.

2.23 Διαμόρφωση εύρους παλμών (Pulse Width Modulation) - PWM

Σήματα διαμορφωμένα κατά εύρος παλμών (PWM), ευρύτερα γνωστά με τον όρο ‘συρμοί παλμών’, Pulse Width Modulation ή PWM, είναι μια τεχνική αναπαράστασης ενός αναλογικού μεγέθους σε ψηφιακή μορφή. Μια κυματομορφή PWM βασίζεται σε έναν τετραγωνικό παλμό, ένα σήμα που κυμαίνεται μεταξύ on και off. Η διάρκεια «on time» λέγεται εύρος του παλμού. Για να έχουμε διάφορες αναλογικές τιμές, μεταβάλλουμε το εύρος αυτό.



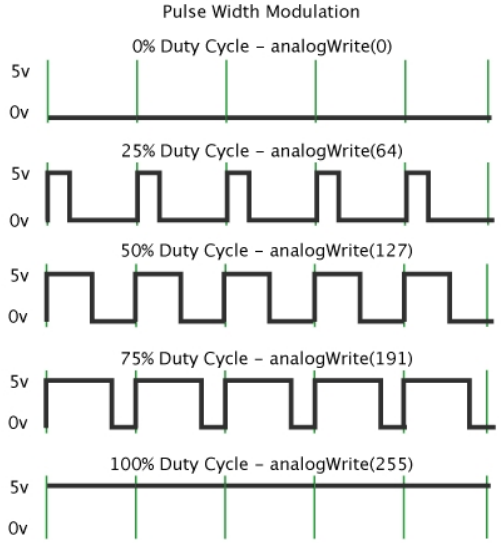
Εικόνα 14: Διαμόρφωση Εύρους Παλμών (PWM), Σήμα Πηγής - Σήμα PWM.

Πλεονεκτήματα Διαμόρφωσης Πλάτους Παλμών

Ένα από τα πλεονεκτήματα του PWM είναι ότι το σήμα παραμένει ψηφιακό σε όλο το κύκλωμα από τον επεξεργαστή έως την έξοδο, χωρίς να είναι απαραίτητη καμία μετατροπή από ψηφιακό σε αναλογικό. Κρατώντας το σήμα ψηφιακό, φαινόμενα θορύβου ελαχιστοποιούνται. Ο θόρυβος μπορεί να επιδράσει σε ένα ψηφιακό σήμα μόνο αν είναι τόσο ισχυρός ώστε να μπορεί να αλλάξει το λογικό 1 σε 0 και το αντίστροφο. Η επίδραση του θορύβου είναι ένας λόγος για το οποίο επιλέγουμε PWM για αναλογικό έλεγχο και είναι ο βασικός λόγος επιλογής του PWM στις επικοινωνίες.

Η συμβολή της συνάρτησης AnalogWrite

Στο Arduino, προκειμένου ένα δείγμα ενός αναλογικού μεγέθους (στην κλίμακα από 0-255) να αναπαρασταθεί σε μορφή PWM γίνεται κλήση της συνάρτησης analogWrite(). Το εύρος του παλμού PWM διαμορφώνεται με βάση την τιμή του αναλογικού δείγματος, όπως φαίνεται στην ακόλουθη εικόνα. Οι πράσινες γραμμές οριοθετούν την (σταθερή) περίοδο του σήματος PWM. Προφανώς, αυτή η διάρκεια είναι το αντίστροφο της συχνότητας PWM και στην περίπτωση του Arduino είναι ίση με 2msec.



Εικόνα 15: Διαμόρφωση παλμών κατά πλάτος (PWM), διάγραμμα τάσης-χρόνου.

ΚΕΦΑΛΑΙΟ 3^{ο} : ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

1ο Εργαστηριακό μάθημα

LED

RGB LED (Πολύχρωμη φωτοδίοδος)

7-Segment LED Display (Οθόνη ένδειξης ψηφίου)

Traffic lights (Φωτεινός σηματοδότης)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός :

Χρωματιστών led.

Πολύχρωμης φωτοδιόδου.

Οθόνης ψηφίου.

Φωτεινού σηματοδότη.

Θεωρητική εισαγωγή

Η δίοδος εκπομπής φωτός (Led, Light Emitting Diode) είναι ένας ημιαγωγός (ειδικά κατασκευασμένη δίοδος) η οποία κατά την ορθή πόλωση, όταν διαρρέεται από ρεύμα, εκπέμπει φωτεινή ακτινοβολία. Η δίοδος φωτοεκπομπής έχει δύο ακροδέκτες την άνοδο και την κάθοδο και πρέπει να πολώνεται πάντα ορθά (ένα συγκεκριμένο άκρο πρέπει να συνδεθεί σε θετικότερη τάση από το άλλο) για να άγει και να εμφανίσει φως. Το θετικότερο άκρο του led ονομάζεται άνοδος και το αρνητικότερο κάθοδος. Όταν το led έχει πολωθεί ορθά τότε άγει, και η πτώση τάσης ανάμεσα στους ακροδέκτες του είναι περίπου 2V. Στην κατάσταση αυτή το ρεύμα που διαρρέει το led κυμαίνεται μεταξύ 10-35mA (η ακριβής τιμή εξαρτάται από τον τύπο του led). Όμως για να αρχίσει να άγει ρεύμα πρέπει η τάση πόλωσης να ξεπεράσει την τάση κατωφλίου (εξαρτάται από την κατασκευή του led).

Η δίοδος πρέπει πάντα να συνδέεται σε σειρά με προστατευτική αντίσταση κατάλληλου μεγέθους ώστε να περιορίζεται το ρεύμα που περνά διαμέσου της, γιατί είναι πολύ ευαίσθητη στην υψηλή τάση. Η φωτεινότητα της διόδου είναι ανάλογη με το ρεύμα που τη διαρρέει. Το χρώμα του φωτός που εκπέμπεται εξαρτάται από το μήκος κύματος της ακτινοβολίας, το οποίο εξαρτάται από το υλικό κατασκευής του led και μπορεί να είναι υπέρυθρο, ορατό φώς ή υπεριώδες.

Η αρχή λειτουργίας των leds βασίζεται στο γεγονός πως στην ορθή πόλωσή τους δημιουργούνται επανασυνδέσεις οπών και ηλεκτρονίων στην επαφή P-N της διόδου. Με τις επανασυνδέσεις οπών και ηλεκτρονίων απελευθερώνεται ενέργεια από τα ηλεκτρόνια με την μορφή ηλεκτρομαγνητικής ακτινοβολίας. Η ένταση της ηλεκτρομαγνητικής ακτινοβολίας είναι ανάλογη με την ένταση του ρεύματος που διαρρέει την δίοδο led. Την μέγιστη ένταση του ρεύματος

την ορίζει ο κατασκευαστής, γι' αυτό και η πόλωση της διόδου γίνεται με τη σύνδεση μιας αντίστασης στο ένα άκρο του led, που η τιμή της υπολογίζεται σε συνάρτηση με την τάση της πηγής σύμφωνα με την σχέση: **R = (V-Vd) / Id**, όπου V η τάση της πηγής, Vd η τάση λειτουργίας του led η οποία δίνεται από τον κατασκευαστή (περίπου 1,8V) και Id το ρεύμα που διαρρέει το led.
Για να υπάρχει ροή ρεύματος μέσω μιας αντίστασης θα πρέπει στα άκρα της αντίστασης να εφαρμόσουμε τάση. Ο νόμος του Ohm συνδέει την τάση (V) με το ρεύμα (I) και την αντίσταση (R). Η σχέση αυτή μπορεί να γραφτεί με τρεις τρόπους:

$$I = \frac{V}{R} \qquad R = \frac{V}{I} \qquad V = I \times R$$

όπου:

V = Τάση σε Volts (V)

I = Ρεύμα σε Amperes (A)

R = Αντίσταση σε Ohm (Ω)

Για τα περισσότερα ηλεκτρονικά κυκλώματα το 1 Ampere είναι πολύ μεγάλο μέγεθος και το 1 Ohm πολύ μικρό, οπότε συνήθως μετράμε το ρεύμα σε milliamperes (mA) και την αντίσταση σε kilohms (ΚΩ). Για να εφαρμοστεί ο νόμος του Ohm ίσως χρειαστεί να μετατρέψουμε το ρεύμα ή την αντίσταση σε πολλαπλάσιο ή υποπολλαπλάσιο τους.
(1mA=0.001A και 1ΚΩ=1000Ω).

Για παράδειγμα, ένα λεντάκι που είναι συνδεδεμένο σε μία μπαταρία 5V πρέπει να διαρρέεται από 18mA ρεύμα σύμφωνα με τον κατασκευαστή του (ελέγχουμε το datasheet), άρα για να υπολογίσουμε την αντίσταση στο κύκλωμα έχουμε:
R=V/I => R=5/0,018(mA) => R=278Ω=~320Ω.

ΑΣΚΗΣΗ 1.1

Σκοπός της Άσκησης:

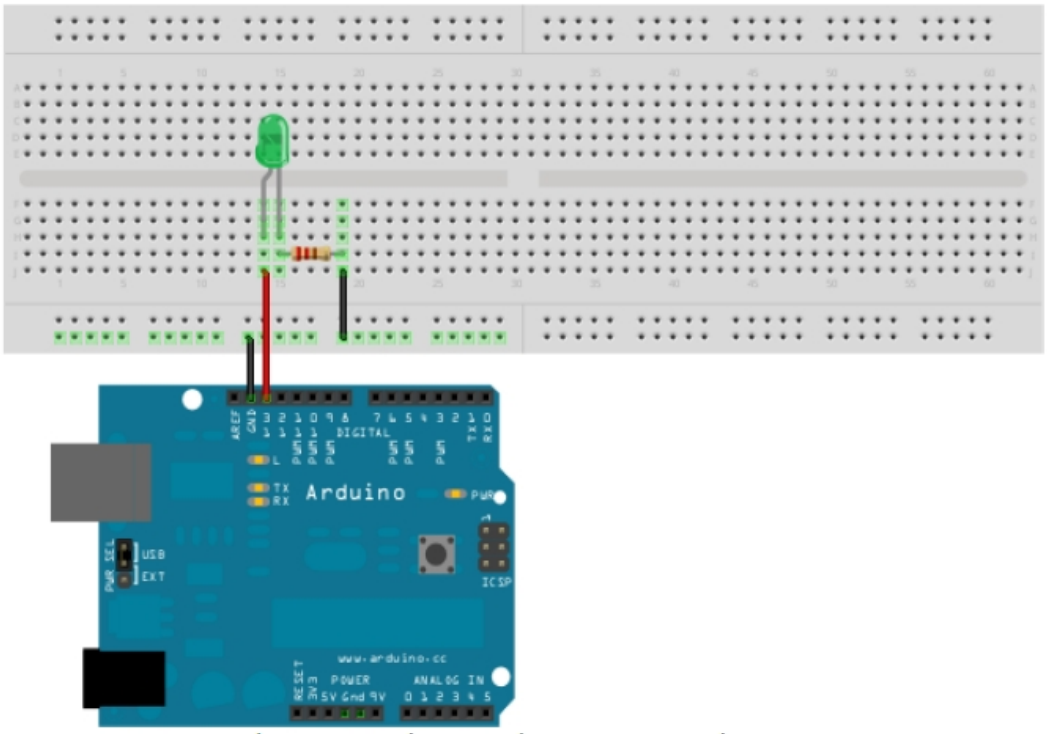
Σκοπός της άσκησης είναι η σύνδεση ενός led στο board και ο προγραμματισμός του, έτσι ώστε να ανάβει και να σβήνει σε χρονικά διαστήματα ενός δευτερολέπτου.

Απαιτούμενα υλικά:

1. Arduino UNO Rev3 
2. Solderless breadboard 
3. 5mm LED
4. 220 Ω 
5. 3x Jumper Wires (Καλώδια σύνδεσης) 

Συνδεσμολογία:

Η συνδεσμολογία που θα κάνουμε φαίνεται στο παρακάτω σχήμα:

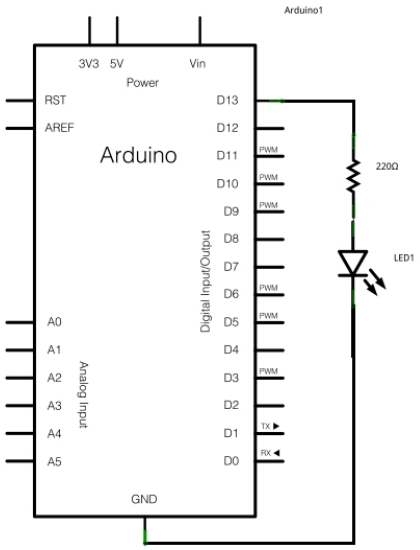


Σχήμα 2: Φυσική διάταξη του κυκλώματος led. [1]

ΠΡΟΣΟΧΗ

Όταν συνδέουμε τα εξαρτήματα πρέπει να αφαιρούμε το μικροελεγκτή από τη USB θύρα ή οποιαδήποτε άλλη πηγή τάσης (μπαταρίες κτλ), αλλιώς υπάρχει κίνδυνος καταστροφής της πλακέτας.

Σχηματικό διάγραμμα:



Σχήμα 3: Συνδεσμολογία οδήγησης κυκλώματος led. [2]

[1] [2] Η αναπαράσταση του κυκλώματος έγινε με τη χρήση του δωρεάν διανεμόμενου προγράμματος Fritzing (http://fritzing.org/).

Πρόγραμμα 1.1

```
int ledPin = 13;

void setup()
{
      pinMode(ledPin, OUTPUT);
}

void loop()
{
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
}
```

Περιγραφή της Άσκησης:

Το Arduino UNO αποτελείται από δεκατέσσερα 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).

Θα δούμε δύο παραδείγματα για να μπορέσουμε να κατανοήσουμε τον προγραμματισμό των ψηφιακών pin του Arduino. Στο πρώτο παράδειγμα βλέπουμε πώς μπορούμε να δηλώσουμε ένα pin του Arduino και να το χρησιμοποιήσουμε σαν έξοδο. Στο δεύτερο θα μελετήσουμε τον τρόπο που μπορούμε να εισάγουμε δεδομένα σε ένα pin. Όπως στην έξοδο, έτσι και στην είσοδο οι τιμές που μπορεί να δεχτεί ένα pin είναι HIGH ή LOW. Την τιμή HIGH μπορούμε να την πάρουμε από το pin 5V του Arduino ενώ την τιμή LOW μπορούμε να την πάρουμε από τα pin GND (ground, 0V). Για την επιλογή της τιμής 0V ή 5V θα μας βοηθήσει η χρήση ενός διακόπτη.

Μελετώντας το πρόγραμμα που φορτώσαμε στο Arduino:

Στη πρώτη γραμμή δηλώνουμε μια ακέραια μεταβλητή την ledPin και της δίνουμε την αρχική τιμή 13, ο λόγος που έγινε αυτή η δήλωση είναι για να μας διευκολύνει παρακάτω στο πρόγραμμα αν δεν θυμόμαστε σε ποιο pin συνδέσαμε το led. Στη συνάρτηση setup() η εντολή pinMode() χρησιμοποιείται για να δηλώσουμε την χρήση του pin 13, δηλαδή αν θα το χρησιμοποιήσουμε σαν είσοδο (input) ή σαν έξοδο (output). Η συνάρτηση loop() χρησιμοποιείται για την επανάληψη του κώδικα, δίνοντάς μας τη ευκαιρία αν θέλουμε να ανανεώσουμε τις τιμές των μεταβλητών και να να κάνουμε το Arduino να ανταποκρίνεται στις αλλαγές αυτές. Η εντολή digitalWrite() είναι υπεύθυνη στο να δώσει τιμές HIGH ή LOW στο pin που δηλώνουμε μέσα στις παρενθέσεις. Η εντολή delay() είναι υπεύθυνη για την παύση του προγράμματός μας για χρόνο που δηλώνουμε μέσα στις παρενθέσεις, το χρόνο αυτό τον καθορίζουμε σε milliseconds (ms) (1000ms = 1sec). Ελέγχουμε τον κώδικά για τυχόν λάθη πατώντας το κουμπί verify, και πατάμε το κουμπί upload για να γίνει η

μεταγλώττιση και το ανέβασμα του εκτελέσιμου αρχείου στο μικροελεγκτή. Αν όλα πήγαν καλά τότε θα δούμε το led να αναβοσβήνει σε διαστήματα ενός δευτερολέπτου.

ΑΣΚΗΣΗ 1.2

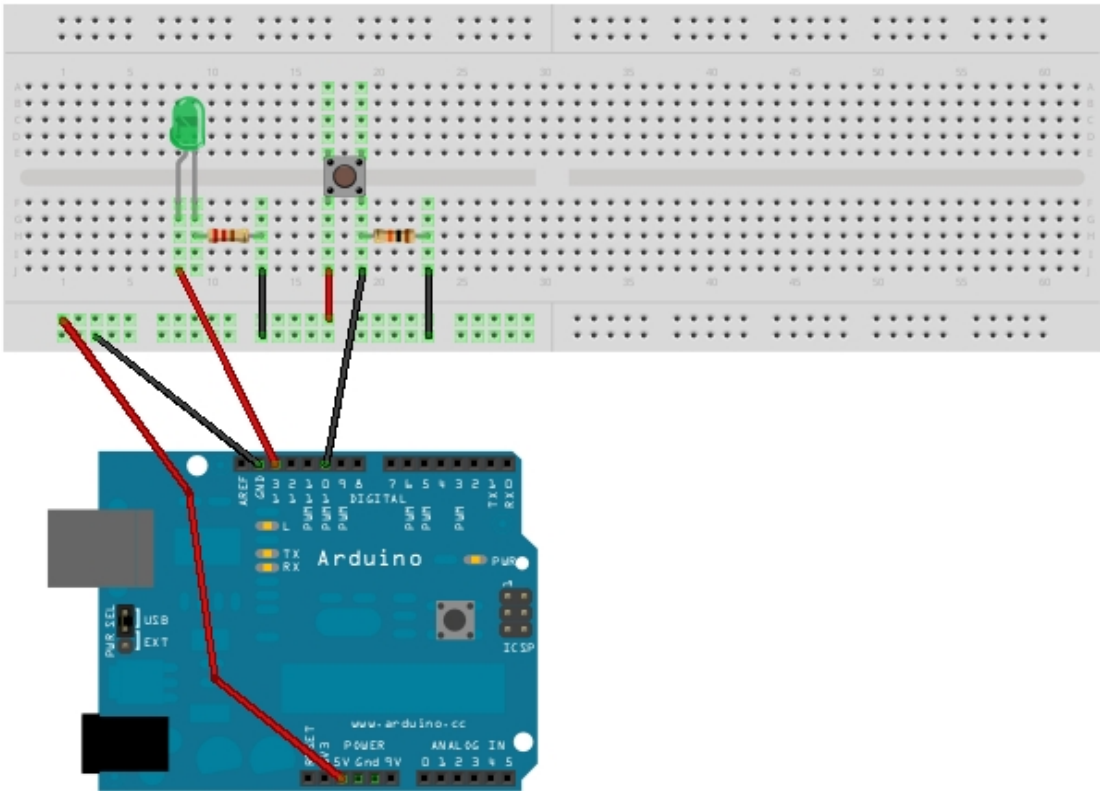
Σκοπός της Άσκησης:

Σκοπός της άσκησης είναι η σύνδεση ενός led στο board και ο προγραμματισμός του, έτσι ώστε με την βοήθεια ενός διακόπτη να ανάβει και να σβήνει για χρονικό διάστημα ενός δευτερολέπτου.

Απαιτούμενα υλικά:

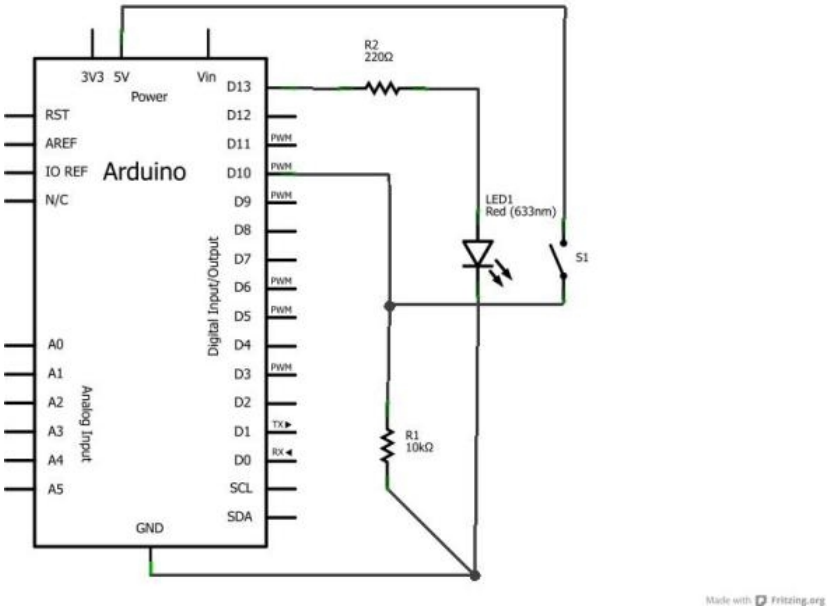
1. Arduino UNO 
2. Breadboard 
3. 5mm LED 
4. 220 ohm Resistor 
5. 10kΩ Resistor 
6. Διακόπτης Pushbutton (tactile switch) 
7. Jumper Wires 

Συνδεσμολογία:



Σχήμα 4: Φυσική διάταξη του κυκλώματος led με χρήση κουμπιού.

Σχηματικό διάγραμμα:



Σχήμα 5: Συνδεσμολογία οδήγησης φωτοδιόδου με χρήση κουμπιού.

Πρόγραμμα 1.2

```
int ledPin = 13;
int inPin = 10;

void setup()
{
      pinMode(ledPin, OUTPUT);
      pinMode(inPin, INPUT);
}

void loop()
{
      if (digitalRead(inPin) == HIGH)
      {
            digitalWrite(ledPin, HIGH);
            delay(1000);
            digitalWrite(ledPin, LOW);
            delay(1000);
     }
}
```

Περιγραφή της Άσκησης

Οι περισσότερες εντολές είναι κοινές με τη προηγούμενη άσκηση.

Μελετώντας τη συνθήκη if (digitalRead(inPin) == HIGH){}, παρατηρούμε ότι η εντολή digitalRead(inPin) διαβάζει τη κατάσταση του pin 10 του Arduino. Αν πιέσουμε το διακόπτη, η θύρα 10 παίρνει την τιμή HIGH, δηλαδή κλείνει το κύκλωμα με 5V. Αν αφήσουμε τον διακόπτη τότε η θύρα γειώνετε (GND) και έχουμε κατάσταση LOW. Όπως βλέπουμε ποιο πάνω, ο κώδικας που βρίσκετε μέσα στην if() θα εκτελεστεί μόλις ανιχνευτούν 5V στη ψηφιακή θύρα.

Traffic lights (Φωτεινός σηματοδότης)

ΑΝΤΙΚΕΙΜΕΝΟ : Προσομοίωση φωτεινού σηματοδότη.

Θεωρητική εισαγωγή

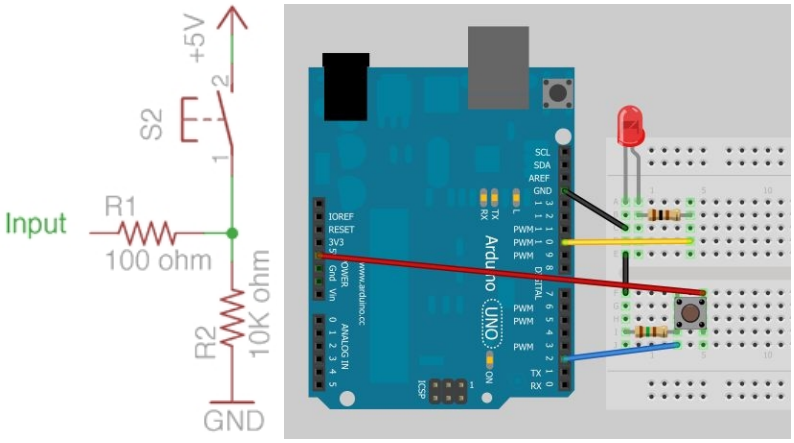
Διακόπτης ή κουμπί (switch-pushbutton) ονομάζεται το ηλεκτρικό στοιχείο που μεταβάλλει τη δυνατότητα διέλευσης του ηλεκτρικού ρεύματος διάμεσο του. Ο διακόπτης μπορεί να χρησιμοποιηθεί, για να απομονώσει μέρος ενός κυκλώματος.

Οι διακόπτες μεταφέρουν τις στοιχειώδεις πληροφορίες 0 ή ψευδής όταν είναι ανοιχτοί και 1 ή αληθής όταν είναι κλειστοί. Η αλλαγή της κατάστασης ενός διακόπτη γίνεται είτε μεταβάλλοντας την αγωγιμότητα ενός μέρους του, που παρεμβάλλεται μεταξύ των ακροδεκτών του, ή αλλάζοντας την απόσταση μεταξύ δύο αγώγιμων μερών του, που ονομάζονται επαφές. Συνήθως ο πρώτος τρόπος χρησιμοποιείται σε αυτόματους διακόπτες, ενώ ο δεύτερος σε χειροκίνητους. Σε αυτήν την περίπτωση μία επαφή είναι σταθερή στη θέση της, ενώ η άλλη μετακινείται μηχανικά.

Οι μηχανικοί διακόπτες έχουν δύο τρόπους σύνδεσης (ενεργοποίησης): Προς την τάση αναφοράς (pull up) και προς τη γείωση (pull down). Όπως και δύο είναι οι ορισμοί της "κανονικής κατάστασης" ανάλογα με τη θέση σε ηρεμία: Κανονικά ανοικτός (N.O.) και Κανονικά κλειστός (N.C.). Όλες οι καταστάσεις είναι ηλεκτρικά ορισμένες είτε στην τάση αναφοράς (5V) είτε στη γείωση (μηδέν).

Για να διέλθει ηλεκτρικό ρεύμα μέσω ενός διακόπτη, πρέπει να είναι κλειστός και να εφαρμοστεί στους ακροδέκτες του διαφορά δυναμικού. Για να μη διέλθει ηλεκτρικό ρεύμα αρκεί να είναι ανοιχτός, αν και είναι πιθανό να είναι κλειστός και να μη διαρρέεται από ηλεκτρικό ρεύμα, γιατί δεν υπάρχει τάση.

Pull-Down Resistors

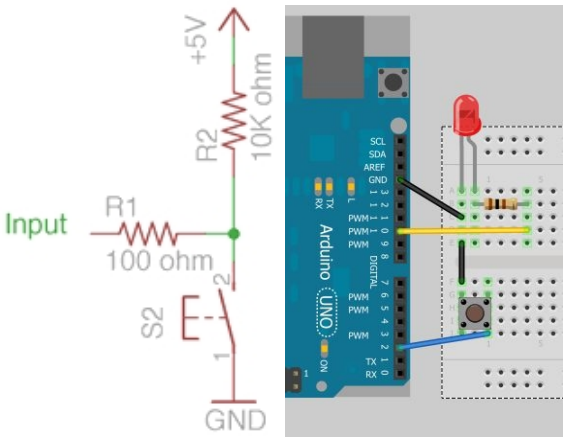


Σχήμα 6: Διάγραμμα και Φυσική διάταξη του παραδείγματος pull-down αντίστασης.

Στο παραπάνω σχήμα κάνουμε χρήση pull-down αντίστασης (πρόσδεσης στη γείωση). Όταν το κουμπί πατηθεί το ρεύμα διαλέγει το μονοπάτι με την μικρότερη αντίσταση και κινείται από

τα 5V προς τη θύρα εισόδου (υπάρχει 100 Ω αντίσταση στη θύρα εισόδου και 10 kΩ στη γείωση). Η αντίσταση σε σειρά προστατεύει την ψηφιακή είσοδο από μεγάλες τιμές του ρεύματος. Όταν το κουμπί δεν είναι πατημένο, η είσοδος συνδέεται με την αντίσταση 10 kΩ και γειώνεται. Χωρίς την αντίσταση R2, η είσοδος δεν θα συνδεόταν πουθενά όταν το κουμπί είναι πατημένο, και το ρεύμα θα κυλούσε από την τάση στη γείωση. Σε αυτό το κύκλωμα η θύρα εισόδου πάντα θα γειώνεται όταν το κουμπί δεν είναι πατημένο και θα οδεύει προς τα 5V όταν το κουμπί πατηθεί. Έτσι εξασφαλίζουμε ότι δε θα δημιουργηθεί κλειστό κύκλωμα.

Pull-Up Resistors



Σχήμα 7: Διάγραμμα και Φυσική διάταξη του παραδείγματος pull-up αντίστασης.

Εδώ η διαφορά με το προηγούμενο κύκλωμα είναι ότι έχουμε αντιστρέψει τις θέσεις της αντίστασης και του κουμπιού. Η αντίσταση τώρα μετατρέπεται σε pull-up (ανύψωσης σε τάση). Όταν το κουμπί δεν είναι πατημένο η θύρα εισόδου συνδέεται με τα 5V και η κατάσταση της θα είναι HIGH. Όταν το κουμπί είναι πατημένο το μονοπάτι που θα ακολουθήσει το ρεύμα (της μικρότερης αντίστασης) είναι προς τη γείωση, έτσι η είσοδος γειώνεται και αποκτάει κατάσταση LOW. Χωρίς την αντίσταση μεταξύ των 5V και της γείωσης, δημιουργείται κλειστό κύκλωμα που θα μπορούσε να καταστρέψει το arduino. Χάρις στην αντίσταση περιορίζεται το ρεύμα.

Η pull-up αντίσταση χρησιμοποιείται συχνότερα σε ψηφιακά κυκλώματα. Με τη χρήση απλών pull-up και pull-down αντιστατών μπορούμε να διασφαλίσουμε την κατάσταση μιας θύρας εισόδου πάντα σε HIGH ή LOW κατάσταση.

ΑΣΚΗΣΗ 1.3

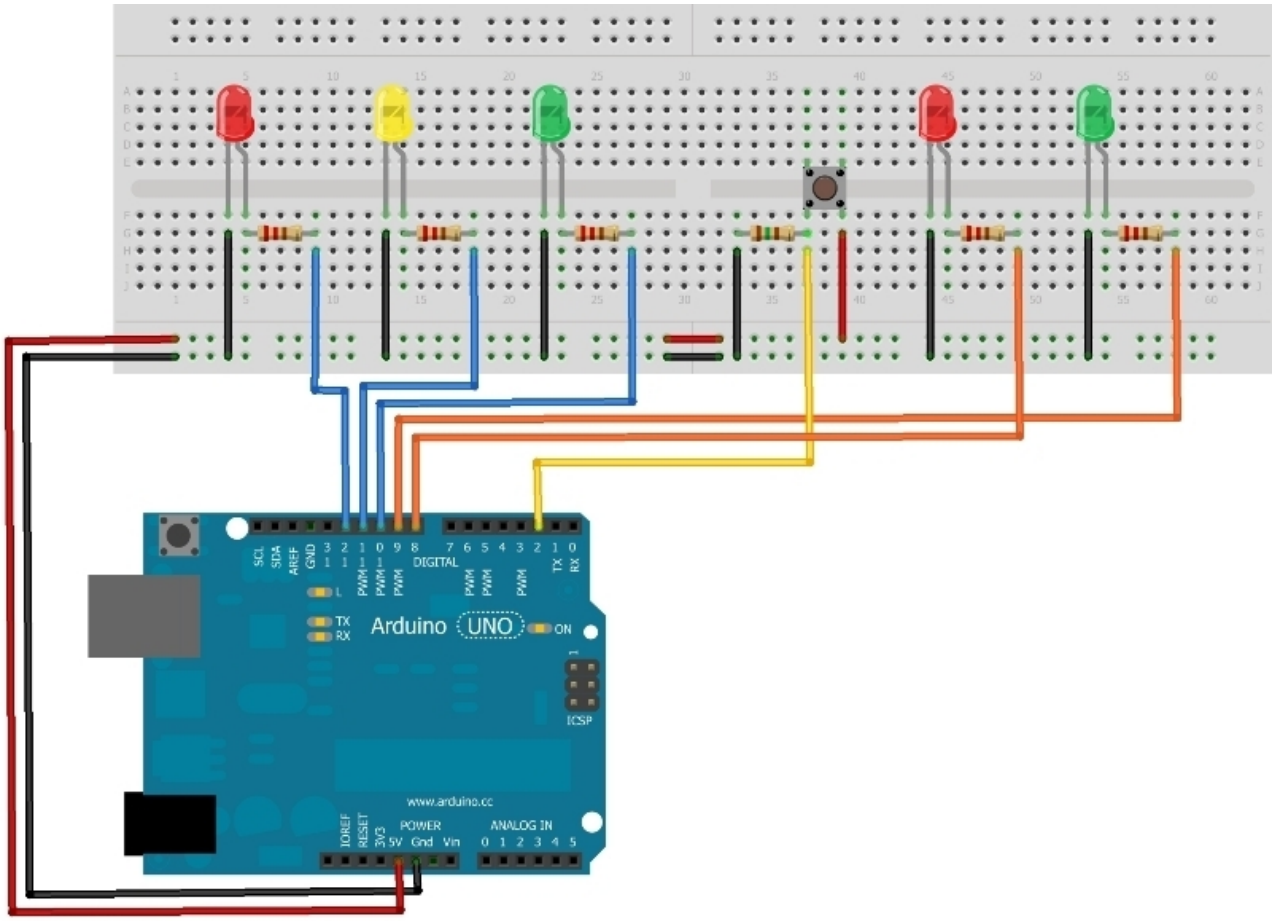
Σκοπός της Άσκησης:

Προσομοίωση φωτεινού σηματοδότη πεζών και αυτοκινήτων. Οι σηματοδότες των αυτοκινήτων εναλλάσσονται μέχρι να πατήσει το κουμπί ο πεζός. Με το πάτημα του κουμπιού ο πεζός ενεργοποιεί το φανάρι προκαλώντας αλλαγή στη κατάσταση των φαναριών που κάνει τα αυτοκίνητα να σταματήσουν και επιτρέπει στους πεζούς να διασχίσουν με ασφάλεια το δρόμο.

Απαιτούμενα Υλικά:

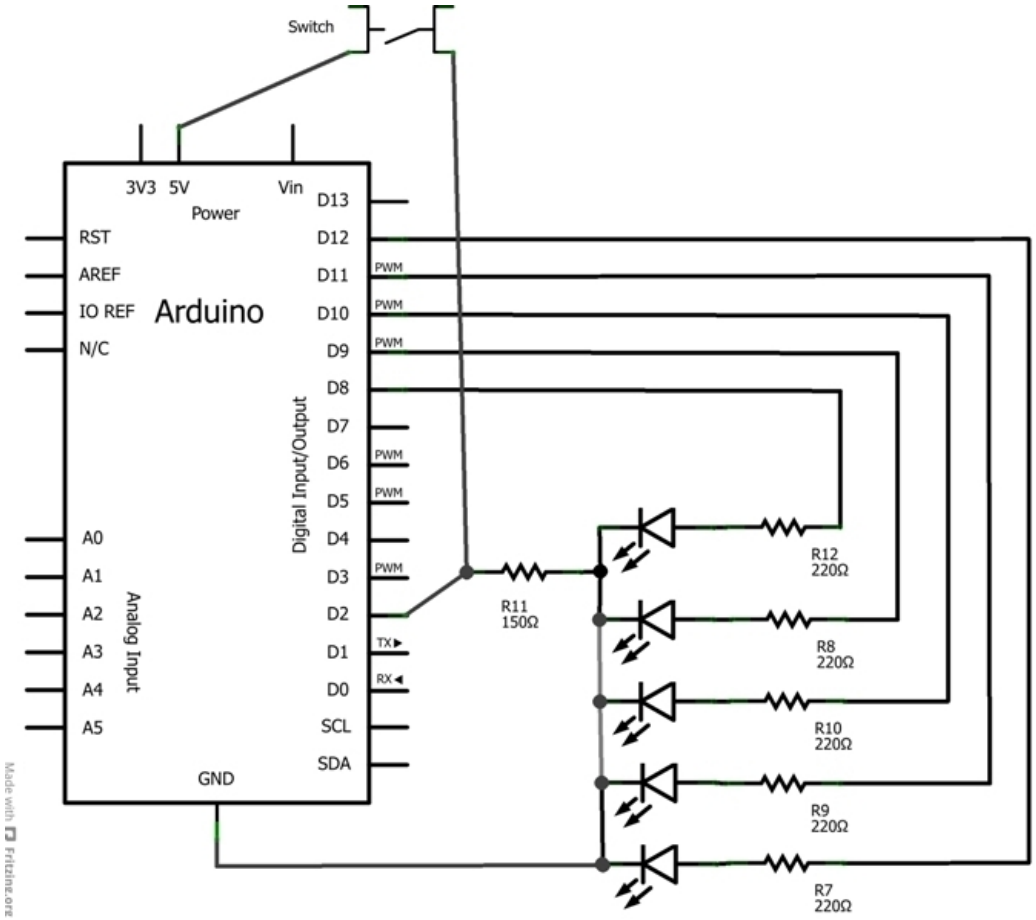
1. Διακόπτης (tactile switch) 
2. 4x 220 Ω 
3. 150 Ω Resistor 
4. 2x Green LED 
5. 2x Red LED 
6. Yellow LED 

Συνδεσμολογία:



Σχήμα 8: Φυσική διάταξη του κυκλώματος «Φωτεινός σηματοδότης».

Σχηματικό διάγραμμα:



Σχήμα 9: Συνδεσμολογία κυκλώματος «Φωτεινός σηματοδότης».

Πρόγραμμα 1.3

```
int carRed = 12; // assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // assign the pedestrian lights
int pedGreen = 8;
int button = 2; // button pin
int crossTime = 5000; // time allowed to cross
unsigned long changeTime; // time since button pressed

void setup() {
      pinMode(carRed, OUTPUT);
      pinMode(carYellow, OUTPUT);
      pinMode(carGreen, OUTPUT);
      pinMode(pedRed, OUTPUT);
      pinMode(pedGreen, OUTPUT);
      pinMode(button, INPUT); // button on pin 2
      // turn on the green light
      digitalWrite(carGreen, HIGH);
      digitalWrite(pedRed, HIGH);
}

void loop() {
      int state = digitalRead(button);
      /* check if button is pressed and it is over 5 seconds since last button
press */
      if (state == HIGH && (millis() - changeTime) > 5000) {
            // Call the function to change the lights
```

```
            changeLights();
      }
}

void changeLights() {
      digitalWrite(carGreen, LOW); // green off
      digitalWrite(carYellow, HIGH); // yellow on
      delay(2000); // wait 2 seconds

      digitalWrite(carYellow, LOW); // yellow off
      digitalWrite(carRed, HIGH); // red on
      delay(1000); // wait 1 second till its safe

      digitalWrite(pedRed, LOW); // ped red off
      digitalWrite(pedGreen, HIGH); // ped green on
      delay(crossTime); // wait for preset time period

      // flash the ped green
      for (int x=0; x<10; x++) {
            digitalWrite(pedGreen, HIGH);
            delay(250);
            digitalWrite(pedGreen, LOW);
            delay(250);
      }
      // turn ped red on
      digitalWrite(pedRed, HIGH);
      delay(500);

      //digitalWrite(carYellow, HIGH); // yellow on
      digitalWrite(carRed, LOW); // red off
      delay(1000);
      digitalWrite(carGreen, HIGH);
      //digitalWrite(carYellow, LOW); // yellow off

      // record the time since last change of lights
      changeTime = millis();
      // then return to the main program loop
}
```

Περιγραφή της Άσκησης - Συμπεράσματα

Το πρόγραμμα ξεκινά με το φανάρι των αυτοκινήτων πράσινο και των πεζών στο κόκκινο. Μόλις πατήσουμε το κουμπί το πρόγραμμα ελέγχει ότι τουλάχιστον 5 δευτερόλεπτα έχουν περάσει από την τελευταία φορά που το φανάρι άλλαξε χρώμα ώστε να μη δημιουργηθεί συμφόρηση στη κυκλοφορία των αυτοκινήτων.

Η changeLights() αλλάζει το φανάρι των αυτοκινήτων από πράσινο σε πορτοκαλί και μετά σε κόκκινο. Στη συνέχεια, το φανάρι των πεζών γίνεται πράσινο, μετά από το χρονικό διάστημα που ορίζεται στην μεταβλητή crossTime() (ώρα αρκετή για να επιτρέψει στους πεζούς να διασχίσουν), το πράσινο φανάρι γίνεται φλας ως προειδοποίηση για τον πεζό να βιαστεί. Τέλος το φανάρι των πεζών γίνεται κόκκινο και των αυτοκινήτων γίνεται από κόκκινο πράσινο.

Στην παραπάνω άσκηση χρησιμοποιούμε μία pull-down αντίσταση εξασφαλίζοντας έτσι ότι η χρήση του διακόπτη θα εκληφθεί σωστά από το arduino. Το κύκλωμά μας περιέχει ένα διακόπτη, ο ένας ακροδέκτης συνδέεται απευθείας στην τάση και ο άλλος στη ψηφιακή θύρα 2 και ταυτόχρονα στη γείωση αλλά μέσω μιας αντίστασης που (λόγω θέσης) είναι γνωστή ως pull-down.

Αυτό σημαίνει ότι όταν ο διακόπτης είναι ανοιχτός, η θύρα εισόδου γειώνεται και αποκτάει κατάσταση LOW. Όταν ο διακόπτης είναι κλειστός, δημιουργείται δυναμικό τάσης 5V και η κατάσταση της εισόδου γίνεται HIGH. Παρατηρώντας αν η είσοδος είναι HIGH ή LOW μπορούμε να ανακαλύψουμε αν ο διακόπτης είναι ανοιχτός ή κλειστός. Η παρουσία της αντίστασης μας διασφαλίζει την εύρυθμη λειτουργία του κυκλώματος.

Οι μικροελεγκτής έχει τη δυνατότητα να προγραμματίζει τις θύρες του, ως εισόδους ή εξόδους και την καθεμία ξεχωριστά. Μπορεί επίσης να καθορίσει τη λογική κατάσταση της κάθε θύρας. Σε περίπτωση που κάποια θύρα είναι έξοδος και καθοριστεί υψηλή ή χαμηλή στάθμη σε αυτή, μπορεί να δώσει ή να γειώσει αντίστοιχα ρεύμα μέχρι 40mA. Σε περίπτωση που είναι είσοδος και καθοριστεί HIGH ή LOW κατάσταση, ενεργοποιείται ή απενεργοποιείται αντίστοιχα η εσωτερική αντίσταση pull-up.

Όταν ένα pin σε έναν μικροελεγκτή είναι ορισμένο ως είσοδος είναι ευαίσθητο σε εξωτερικούς θορύβους (παράσιτα) που κάνει την ανάγνωση της κατάστασής του δύσκολη. Για τον λόγο αυτό υπάρχουν οι λεγόμενες pull-up αντιστάσεις. Οι αντιστάσεις αυτές βρίσκονται εσωτερικά στον μικροελεγκτή. Όταν ενεργοποιηθεί για μία είσοδο, τότε η είσοδος αυτή συνδέεται μέσω της αντίστασης με την τάση και βρίσκεται σε λογική κατάσταση 1. Στη συνέχεια με κάποιο διακόπτη μπορεί να πάει στο GND και να βρεθεί σε λογική κατάσταση 0. Όταν δούμε μέσω του κώδικα μας λοιπόν πως κάποια είσοδος είναι σε λογικό 0 σημαίνει πως κάποιος ενεργοποίησε κάποιο διακόπτη.

Οι ενσωματωμένες pull-up αντιστάσεις του arduino, είναι συνδεδεμένες με τις ψηφιακές αλλά και τις αναλογικές του θύρες. Έχουν τιμή 20 kΩ και για να χρησιμοποιηθούν, πρέπει να ενεργοποιηθούν προγραμματιστικά. Για να ενεργοποιηθεί η εσωτερική pull-up αντίσταση, πρώτα θέτουμε το pinMode() της θύρας σε INPUT και μετά θέτουμε τη λογική κατάσταση HIGH στην ίδια θύρα χρησιμοποιώντας την digitalWrite().

pinMode(pin, INPUT);

digitalWrite(pin, HIGH);

Αν αλλάξουμε το pinMode() της θύρας από INPUT σε OUTPUT, αφού ενεργοποιήσουμε τις εσωτερικές pull-up αντιστάσεις, τότε η θύρα θα παραμείνει στην κατάσταση HIGH. Αυτό ισχύει και αντίστροφα: μια θύρα εξόδου που βρισκόταν σε κατάσταση HIGH και επακόλουθα αλλάζει σε INPUT, θα κρατήσει ενεργοποιημένη την εσωτερική της pull-up αντίσταση.

RGB LED (Πολύχρωμη φωτοδίοδος)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας και ο προγραμματισμός της πολύχρωμη φωτοδιόδου.

Θεωρητική εισαγωγή

Βασικά (ή πρωτογενή) χρώματα είναι αυτά τα οποία όταν συνδυαστούν μεταξύ τους μπορούν να παράγουν όλα τους δυνατούς χρωματισμούς. Τα χρώματα που προκύπτουν από τον συνδυασμό των βασικών λέγονται **συμπληρωματικά** (δευτερογενή).

Η οθόνη της τηλεόρασης και του υπολογιστή λειτουργεί με τρία βασικά χρώματα: Κόκκινο-Πράσινο-Μπλε (Red-Green-Blue: RGB). Με συνδυασμό αυτών των χρωμάτων μπορούμε να δημιουργήσουμε τα δευτερεύοντα ως εξής:

- Κίτρινο: Κόκκινο + Πράσινο
- Γαλάζιο: Πράσινο + Μπλε
- Μοβ(Magenta): Μπλε + Κόκκινο

Από τους συνδυασμούς των πρωτογενών ή των δευτερογενών χρωμάτων μπορούμε να δημιουργήσουμε όλους τους δυνατούς χρωματισμούς. Τα τρία πρωτογενή χρώματα όταν συνδυαστούν δίνουν το λευκό.

- Λευκό: Κόκκινο + Πράσινο + Μπλε

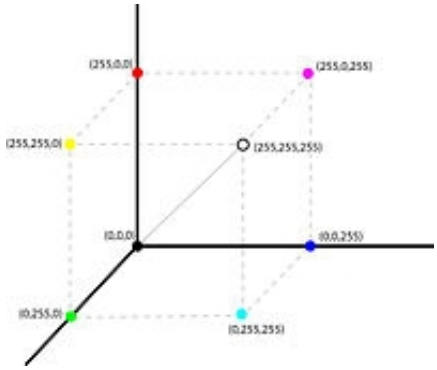
Λευκό επίσης δίνουν και οι συνδυασμοί ενός πρωτογενούς και του αντίθετου του δευτερογενούς.

- Λευκό: Κόκκινο + Γαλάζιο
- Λευκό: Πράσινο + Μοβ(Magenta)
- Λευκό: Μπλε + Κίτρινο

Χρωματικό μοντέλο RGB

Με τα βασικά αυτά χρώματα έχει δημιουργηθεί το χρωματικό μοντέλο RGB με το οποίο μπορεί να γίνει η κωδικοποίηση όλων των χρωμάτων που εμφανίζονται σε μία οθόνη. Στην 8bit έκδοση του χρωματικού αυτού μοντέλου κάθε χρώμα μπορεί να παρασταθεί με μία τριάδα αριθμών από 0 έως 255. Το μοντέλο βασίζεται στο γεγονός ότι όταν μία οθόνη δεν εκπέμπει φως εμφανίζεται μαύρη. Τα υπόλοιπα χρώματα δημιουργούνται με υπέρθεση των τριών βασικών με συγκεκριμένη αναλογία. Τα βασικά, τα δευτερογενή χρώματα και μερικά παραδείγματα δίνονται παρακάτω στην 8bit αυτή έκδοση του μοντέλου:

- Μαύρο: (0,0,0)
- Λευκό: (255,255,255)
- Κόκκινο: (255,0,0)
- Πράσινο: (0,255,0)
- Μπλε: (0,0,255)
- Κίτρινο: (255,255,0)
- Γαλάζιο: (0,255,255)
- Μοβ (Magenta): (255,0,255)
- Πορτοκαλί: (255,102,0)



Σχήμα 10: Γραφική απεικόνιση του 8bit χρωματικού μοντέλου RGB.

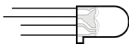
Το μοντέλο αυτό μπορεί να παρασταθεί με έναν κύβο χρωμάτων σε ένα καρτεσιανό σύστημα συντεταγμένων. Στην αρχή των αξόνων είναι η κορυφή του κύβου που αντιστοιχεί στο μαύρο χρώμα, ενώ στις κορυφές του κύβου που βρίσκονται πάνω στους άξονες βρίσκονται τα βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε). Τα δευτερογενή χρώματα βρίσκονται στις τρεις κορυφές του κύβου που βρίσκονται απέναντι από τα αντίστοιχα βασικά χρώματα και στην κορυφή απέναντι από το μαύρο βρίσκεται το λευκό. Κάθε χρώμα στο σύστημα αυτό προσδιορίζεται από ένα σημείο στον κύβο με τρεις συντεταγμένες. Στη διαγώνιο μεταξύ μαύρου και λευκού βρίσκονται όλες οι αποχρώσεις του γκρι.

ΑΣΚΗΣΗ 1.4

Σκοπός της Άσκησης:

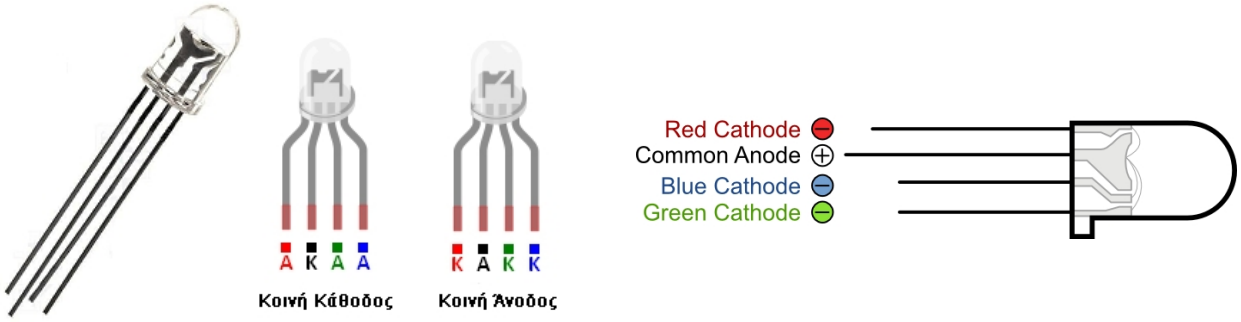
Σύνδεση και προγραμματισμός rgb led με χρήση 3 ποτενσιόμετρων που ελέγχουν κάθε ένα από τα 3 βασικά χρώματα, και δίνουν στην έξοδο όλους τους δυνατούς χρωματικούς συνδυασμούς.

Απαιτούμενα Υλικά:

1. RGB LED ανόδου 
2. 3x Αντιστάσεις 330 Ω 
3. 3x Ποτενσιόμετρα 

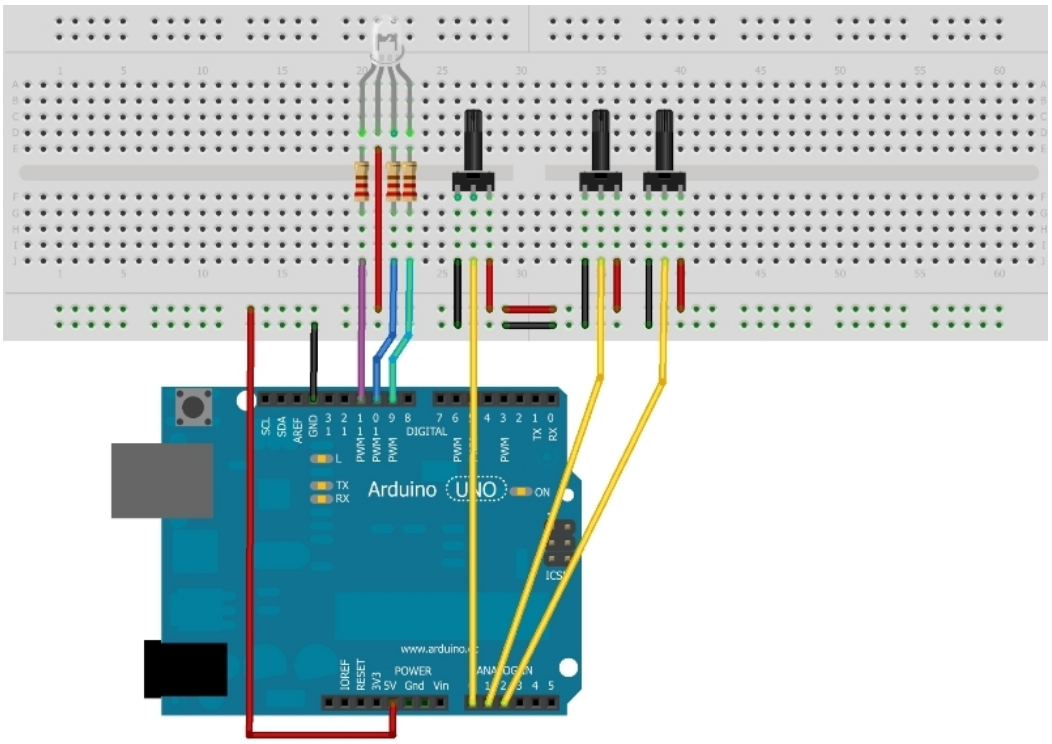
Χαρακτηριστικά RGB LED:

LED diameter: 5mm
LED colour: RGB
Luminosity of red colour: 1800-3000mcd
Luminosity of green colour: 3500-6000mcd
Luminosity of blue colour: 1300-2500mcd
Common electrode: cathode
Viewing angle: 12°
LED lens: transparent
LED current: 20mA
Wavelength of red colour λd: 626nm
Wavelength of green colour λd: 525nm
Wavelength of blue colour λd: 470nm
Mounting: THT



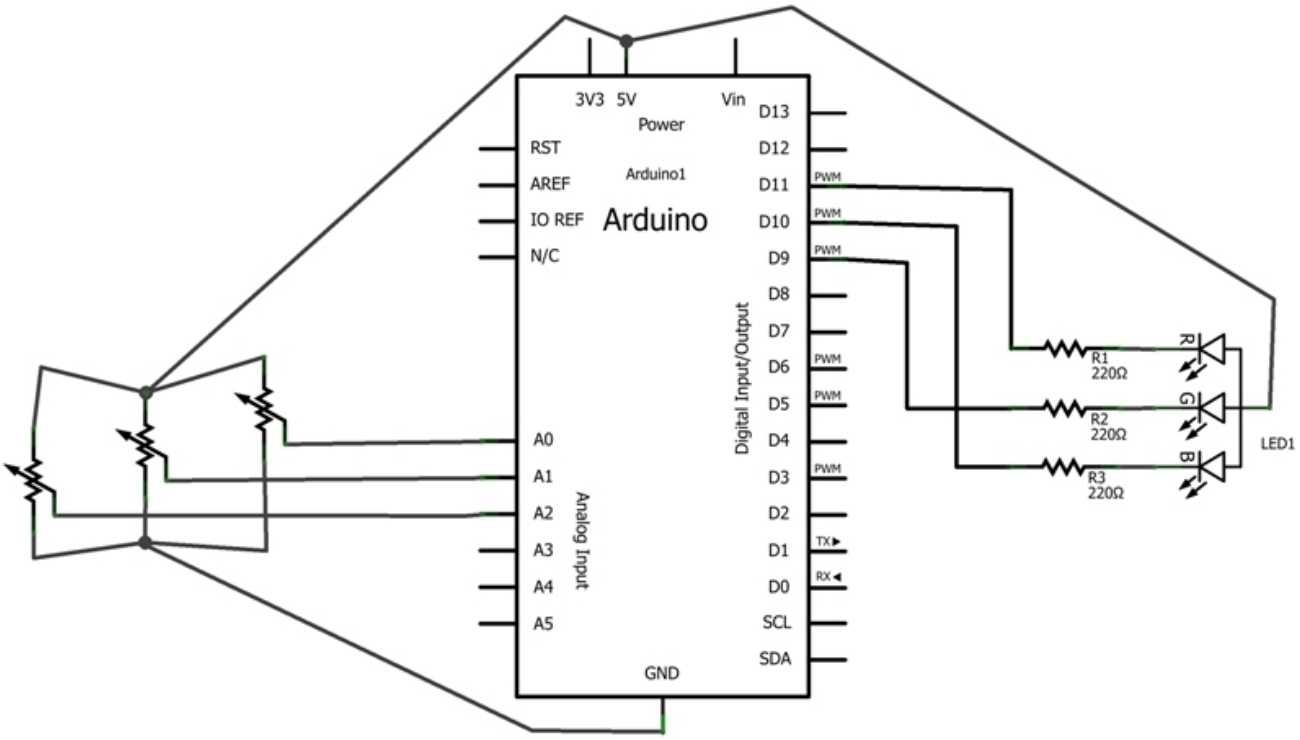
Τα RGB led μπορούν να είναι ανόδου ή καθόδου. Η διαφορά μεταξύ τους έγκειται στη κατασκευή τους που απαιτεί από μέρους μας διαφορετική συνδεσμολογία. Το RGB led έχει 4 ακροδέκτες. Για τα ανοδικά led, αυτός με το μεγαλύτερο μήκος πηγαίνει στη τροφοδοσία 5V ενώ για τα led καθόδου ο μεγαλύτερος ακροδέκτης συνδέεται στη γείωση.

Συνδεσμολογία:



Σχήμα 11: Φυσική διάταξη του κυκλώματος RGB LED ανόδου.

Σχηματικό διάγραμμα:



Σχήμα 12: Συνδεσμολογία κυκλώματος RGB LED ανόδου και 3 ποτενσιόμετρων.

Πρόγραμμα 1.4

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;

int redPot = 0;
int greenPot = 1;
int bluePot = 2;

int redVal;
int greenVal;
int blueVal;

void setup()
{
  // nothing to do here
}

void loop()
{
  redVal = analogRead(redPot);
  greenVal = analogRead(greenPot);
  blueVal = analogRead(bluePot);

  // analogRead returns a value between 0 and 1023
  // analogWrite wants a value between 0 and 255
  // That means we need to map the input range to
  // the correct output range.
  redVal = map(redVal, 0, 1023, 0, 255);
  greenVal = map(greenVal, 0, 1023, 0, 255);
  blueVal = map(blueVal, 0, 1023, 0, 255);
```

```
  analogWrite(redPin, redVal);
  analogWrite(greenPin, greenVal);
  analogWrite(bluePin, blueVal);
}
```

Παρατηρήσεις

Χρησιμοποιούμε 3 ποτενσιόμετρα ένα για κάθε βασικό χρώμα κόκκινο, πράσινο, μπλε, τα οποία είναι συνδεδεμένα με τις αναλογικές θύρες του arduino 0,1 και 2. Έτσι ώστε με τη χρήση τους να μεταβάλλουμε τη **φωτεινότητα** του κάθε χρώματος ξεχωριστά και να πάρουμε στην έξοδο το συνδυασμό που επιθυμούμε. Ελέγχουμε το RGB LED μέσω 3 ψηφιακών θυρών του arduino και παρατηρούμε με τις αλλαγές στα ποτενσιόμετρα, την αλλαγή των χρωμάτων. **Πειραματικά** μπορούμε να διακρίνουμε εύκολα 7 διαφορετικά χρώματα κόκκινο, πράσινο, μπλε, κίτρινο, γαλάζιο, μωβ και λευκό, όπως φαίνεται στον παρακάτω πίνακα.

Πίνακας 2: Χρωματικός πίνακας αληθείας RGB LED

Colour Truth Table			
red	green	blue	
ON	ON	OFF	yellow
OFF	ON	ON	cyan
ON	OFF	ON	magenta
ON	ON	ON	white

Ο ακροδέκτης κοινής ανόδου συνδέεται απευθείας στην τάση. Οι ακροδέκτες καθόδου συνδέονται διαμέσου αντιστατών με τις ψηφιακές θύρες εξόδου του arduino, ώστε να προστατευτεί η δίοδος. Οι θύρες 9,10,11 διαμορφώνουν το σήμα κατά πλάτος μέσω της συνάρτησης analogWrite(). Αυτό μας επιτρέπει να θέσουμε τη φωτεινότητα ξεχωριστά για κάθε χρώμα.

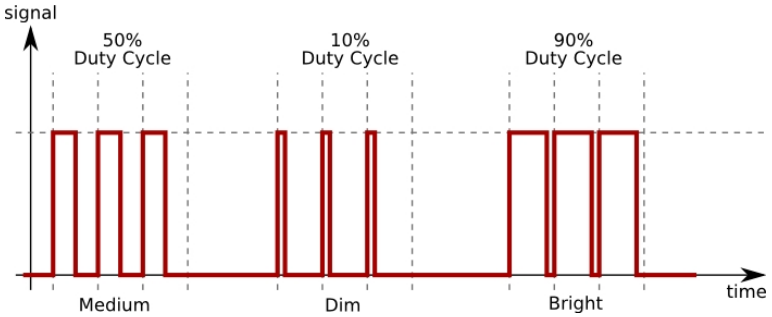
Η φωτεινότητα ενός led εξαρτάται και είναι ανάλογη από το ρεύμα που το διαπερνά. Αλλά θα ήταν αρκετά δύσκολο να χρησιμοποιήσουμε ένα μικροελεγκτή για να ελέγξουμε με ακρίβεια το ρεύμα που ρέει μέσα από το led.

Μολαταύτα, η ανθρώπινη όραση διέπεται από το φαινόμενο της «αισθητήριας μνήμης», με βάση την οποία η «ψευδαίσθηση» της κίνησης προκαλείται από την ταχεία προβολή μιας σειράς ανεξάρτητων εικόνων και οι (μικρές) διαφορές ανάμεσα στις εικόνες γίνονται αντιληπτές σαν κίνηση των στοιχείων της εικόνας. Τα παραπάνω χαρακτηριστικά της ανθρώπινης αντίληψης μέσω του αισθητήριου της όρασης οφείλονται σε δύο φαινόμενα: Το βιολογικό φαινόμενο «**persistence of vision**» το οποίο είναι στην ουσία αποτέλεσμα της μορφής μνήμης η οποία είναι γνωστή ως «αισθητήρια μνήμη», τα ορατά αντικείμενα σύμφωνα με αυτό το φαινόμενο παραμένουν στον αμφιβληστροειδή για ένα μικρό χρονικό διάστημα από τη στιγμή που έχουν ιδωθεί. Καθώς και το

ψυχολογικό φαινόμενο Φι (phi) σύμφωνα με το οποίο ο άνθρωπος τείνει να συμπληρώνει νοητικά μια αντιλαμβανόμενη κίνηση.

Μια εικόνα η οποία έχει ιδωθεί για κλάσματα του δευτερολέπτου θα παραμείνει τυπωμένη στον εγκέφαλο ακόμα και όταν η πραγματική εικόνα έχει εξαφανισθεί. Αυτή είναι η ίδια αρχή που βρίσκεται πίσω από την λειτουργία της τηλεόρασης, όπου μια αστραπιαία αλλαγή της εικόνας δίνει στον εγκέφαλο την ψευδαίσθηση της συνεχόμενης κίνησης. Αναβοσβήνοντας στιγμιαία και συνεχόμενα το led δίνουμε την ψευδαίσθηση στην ανθρώπινη όραση μιας μέσης τιμής φωτεινότητας βασισμένη στον κύκλο διαμόρφωσης κατά πλάτους PWM του σήματος.

Pulse-width Modulation (PWM) ή αλλιώς Διαμόρφωση κατά Πλάτος: είναι η πρακτική της διαμόρφωσης του κύκλου του σήματος, που χρησιμοποιείται για να ελέγξει τη φωτεινότητα του led. Έτσι με τη χρήση ποτενσιόμετρων ελέγχουμε τη φωτεινότητα του κάθε χρώματος του LED και όχι του ρεύματος που το διαπερνά. Στην παρακάτω εικόνα βλέπουμε τρεις κύκλους PWM: 50% ,10% και 90%. Κατά τη διάρκεια του 10% το σήμα βρίσκεται για πολύ λίγο στην λογική κατάσταση HIGH. Κατά τη διάρκεια του 90% το σήμα βρίσκεται τον περισσότερο χρόνο στην λογική κατάσταση HIGH. Αν η συχνότητα του σήματος είναι αρκετά γρήγορη δεν θα υπάρχει εμφανές τρεμόπαιγμα.



Σχήμα 13: Κύκλοι σήματος (PWM) Διαμόρφωσης κατά Πλάτος.

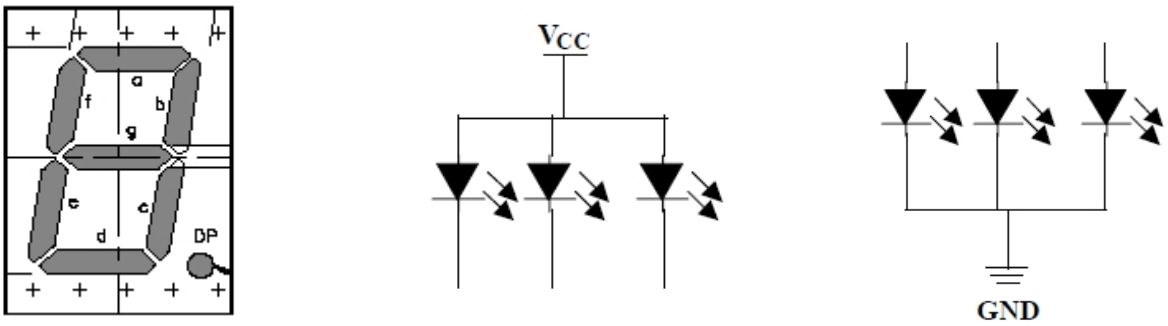
7-Segment LED Display (Επτά-τμηματική μονάδα ένδειξης για φωτεινή απεικόνιση αριθμών)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός οθόνης ψηφίου.

Θεωρητική εισαγωγή

Σε πολλές περιπτώσεις μεμονωμένα LEDs συνδυάζονται σε τμήματα (modules), τα οποία μπορούν να απεικονίσουν έναν χαρακτήρα (αριθμό ή γράμμα). Ο πιο κοινός τύπος ονομάζεται "7-segment-display" και μπορεί να απεικονίσει με 7 LEDs ένα δεκαδικό ψηφίο ή και ορισμένα γράμματα (σχήμα 2α). Τα τμήματα LEDs συναντώνται σε δύο συνδεσμολογίες: κοινής ανόδου (σχήμα 2β), όπου όλα τα LEDs έχουν συνδεμένη την άνοδό τους σε κοινό άκρο, και κοινής καθόδου (σχήμα 2γ), όπου συμβαίνει το αντίστροφο.

Στα τμήματα κοινής ανόδου, ο κοινός ακροδέκτης συνδέεται στην τάση τροφοδοσίας και κάθε κάθοδος οδηγείται από ξεχωριστή λογική έξοδο με αρνητική οδήγηση. Στα τμήματα κοινής καθόδου, ο κοινός ακροδέκτης συνδέεται στη γείωση και κάθε άνοδος συνδέεται σε ξεχωριστή λογική έξοδο με θετική οδήγηση. Και στις δύο περιπτώσεις, θα πρέπει να προστίθεται μια αντίσταση σε σειρά με κάθε ξεχωριστό (όχι στον κοινό) ακροδέκτη των LEDs για τον περιορισμό του ρεύματος.



2 α) **β)** **γ)**

Σχήμα 14: Συνδεσμολογία οδήγησης 7-Segment LED Display κοινής ανόδου (2β), κοινής καθόδου (2γ).

ΑΣΚΗΣΗ 1.5

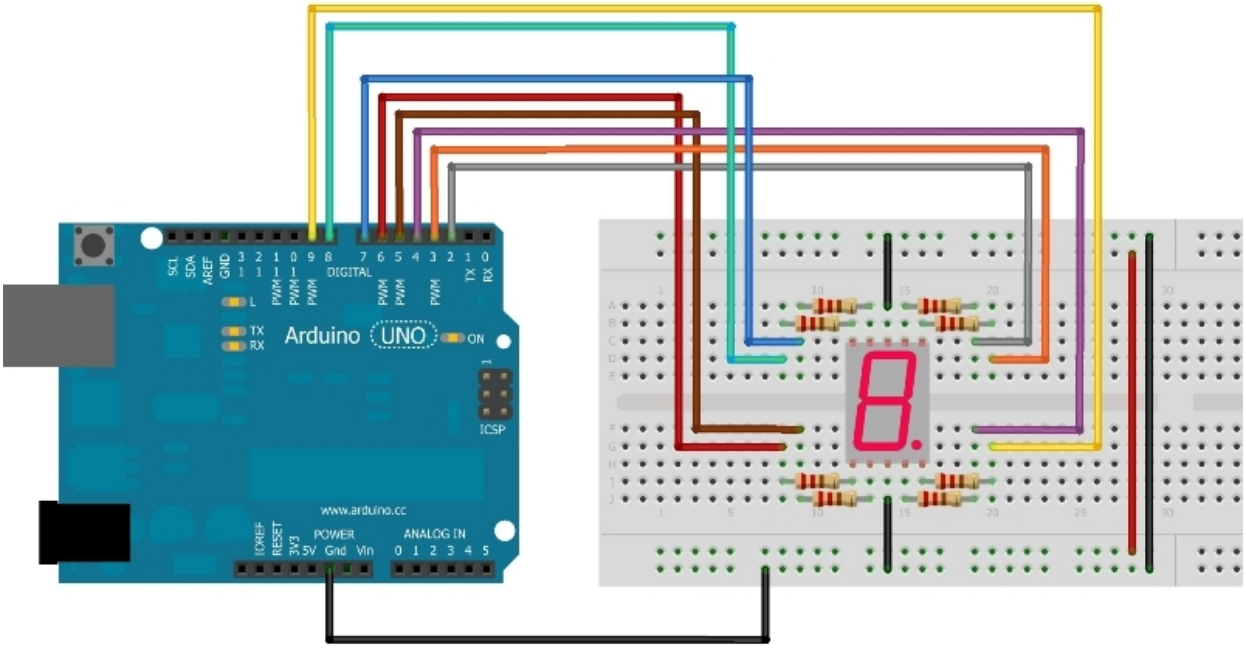
Σκοπός της Άσκησης:

Εμφάνιση αριθμών σε ένα 7-segment display με αντίστροφη μέτρηση από το 9 στο 0.

Απαιτούμενα Υλικά:

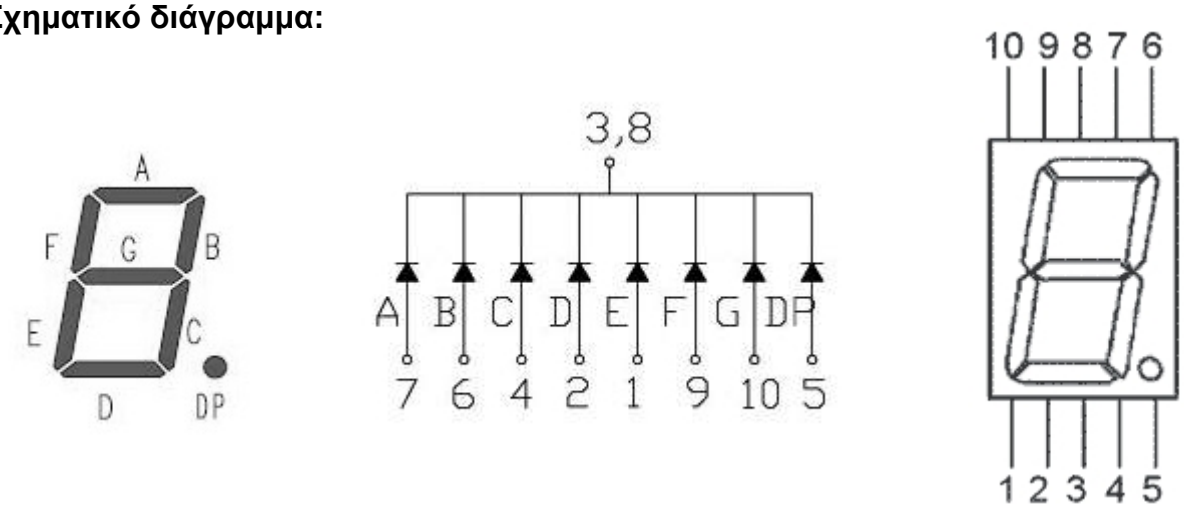
1. 7-Segment LED κοινής καθόδου
2. 8x Αντιστάσεις 470 Ω 

Συνδεσμολογία:



Σχήμα 15: Φυσική διάταξη του κυκλώματος σύνδεσης ψηφίου κοινής καθόδου (common cathode).

Σχηματικό διάγραμμα:

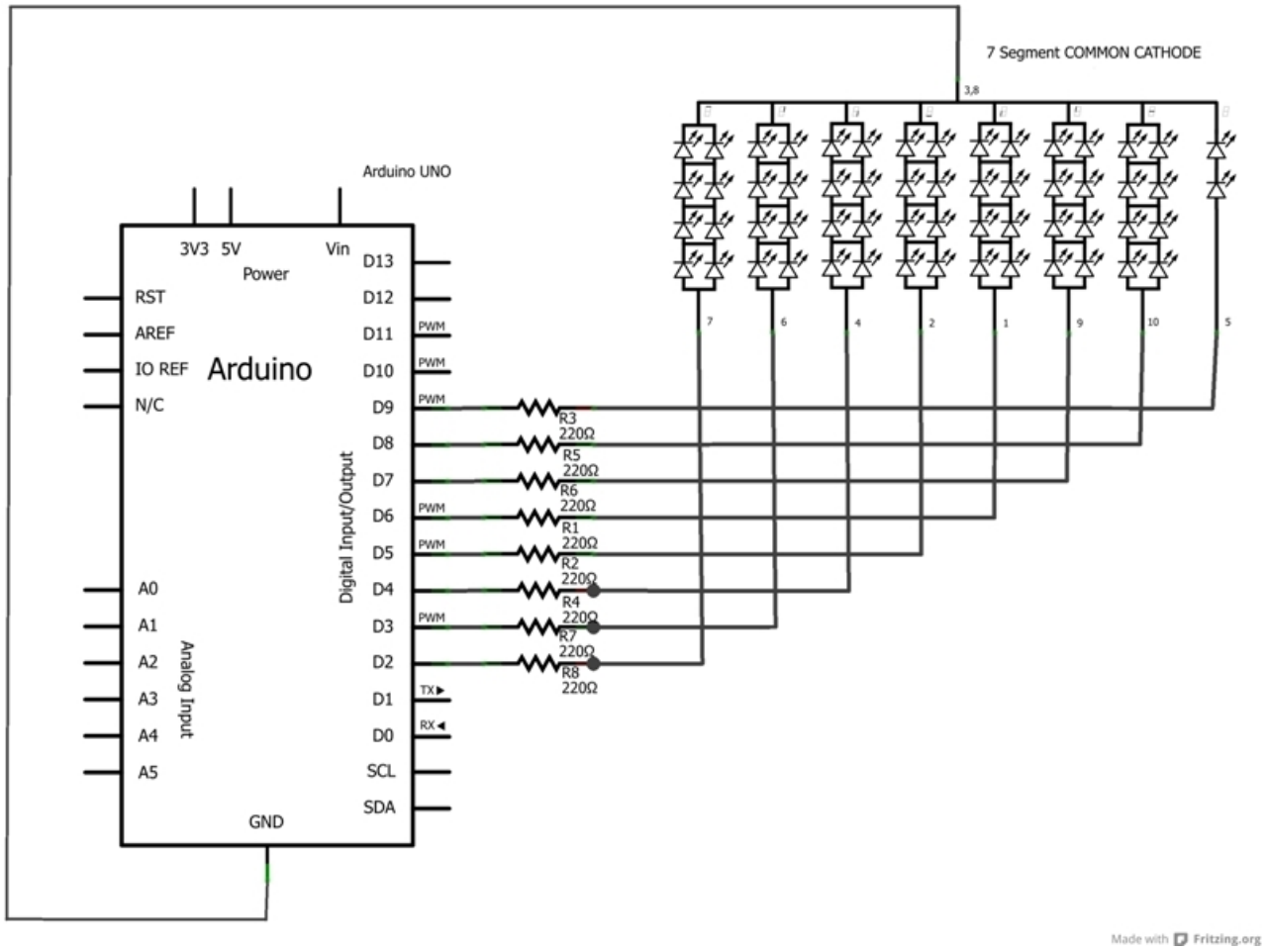


Σχήμα 16: Συνδεσμολογία οδήγησης 7 Segment LED Display κοινής καθόδου.

Πίνακας 3: Αντιστοίχηση θυρών arduino
και οθόνης 7-segment led κοινής καθόδου

Arduino Pin	7 Segment Pin Συνδεσμολογία
2	7 (A)
3	6 (B)
4	4 (C)
5	2 (D)
6	1 (E)
7	9 (F)
8	10 (G)
9	5 (DP)

Στην παρακάτω εικόνα απεικονίζεται ένα 7-seg κοινής καθόδου. Αυτό συμπεραίνεται εύκολα από τη στιγμή που όλες οι κάθοδοι ενώνονται μεταξύ τους και καταλήγουν στη γείωση (στα 7-seg κοινής ανόδου καταλήγουν στην τροφοδοσία 5V). Αντιστάσεις πρέπει να συνδεθούν σε σειρά με τα τμήματα της ανόδου όπως φαίνεται στην εικόνα που ακολουθεί.



Σχήμα 17: Συνδεσμολογία σύνδεσης οθόνης ψηφίου κοινής καθόδου (common cathode).

Πρόγραμμα 1.5

```
// Define the LED digit patterns, from 0 - 9
// Note that these patterns are for common cathode displays
// For common anode displays, change the 1's to 0's and 0's to 1's
// 1 = LED on, 0 = LED off, in this order:

//                                Arduino pin: 2,3,4,5,6,7,8

/*If you were using a common anode LED you would use this section
byte seven_seg_digits[10][7] = { { 0,0,0,0,0,0,1 },  // = 0 x
                                 { 1,0,0,1,1,1,1 },  // = 1 x
                                 { 0,0,1,0,0,1,0 },  // = 2 x
                                 { 0,0,0,0,1,1,0 },  // = 3 x
                                 { 1,0,0,1,1,0,0 },  // = 4 x
                                 { 0,1,0,0,1,0,0 },  // = 5 x
                                 { 0,1,0,0,0,0,0 },  // = 6 x
                                 { 0,0,0,1,1,1,1 },  // = 7 x
                                 { 0,0,0,0,0,0,0 },  // = 8 x
```

```
                                  { 0,0,0,1,1,0,0 }   // = 9 x
                                };
*/
//                                    Arduino pin: 2,3,4,5,6,7,8

 byte seven_seg_digits[10][7] = { { 1,1,1,1,1,1,0 },  // = 0
                                  { 0,1,1,0,0,0,0 },  // = 1
                                  { 1,1,0,1,1,0,1 },  // = 2
                                  { 1,1,1,1,0,0,1 },  // = 3
                                  { 0,1,1,0,0,1,1 },  // = 4
                                  { 1,0,1,1,0,1,1 },  // = 5
                                  { 1,0,1,1,1,1,1 },  // = 6
                                  { 1,1,1,0,0,0,0 },  // = 7
                                  { 1,1,1,1,1,1,1 },  // = 8
                                  { 1,1,1,0,0,1,1 }   // = 9
                                };

// comment list below is LED pin# > Arduino pin# | segment name
void setup() {
  pinMode(2, OUTPUT);  //14 > 2 A
  pinMode(3, OUTPUT);  //13 > 3 B
  pinMode(4, OUTPUT);  // 8 > 4 C
  pinMode(5, OUTPUT);  // 7 > 5 D
  pinMode(6, OUTPUT);  // 6 > 6 E
  pinMode(7, OUTPUT);  // 2 > 7 F
  pinMode(8, OUTPUT);  // 1 > 8 G
  pinMode(9, OUTPUT);  //RDP> 9 decimal point
  writeDot(0);  // start with the "dot" off
}

// displays digits from 9 to 0
void loop() {
  for (byte count = 10; count > 0; --count) {
   delay(1000);
   sevenSegWrite(count - 1);
  }
  delay(2000);
}

void writeDot(byte dot) {
  digitalWrite(9, dot); // decimal point - pin 9 on Arduino
}

void sevenSegWrite(byte digit) {
  byte pin = 2; // Arduino pin where the segment list starts
  for (byte segCount = 0; segCount < 7; ++segCount) {
    digitalWrite(pin, seven_seg_digits[digit][segCount]);
    ++pin;
  }
}
```

Περιγραφή της Άσκησης

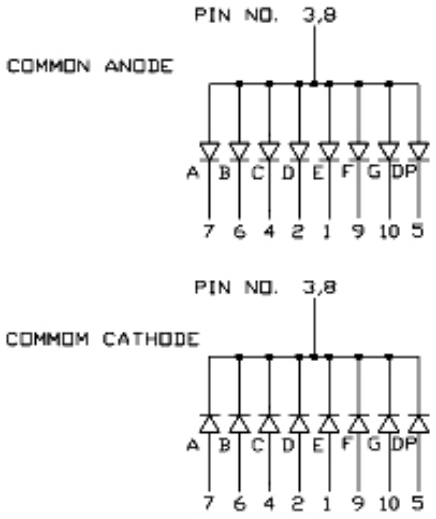
Το 7 Segment LED Display είναι απλά ένα σύμπλεγμα από 8 led σε τμήματα (segments) (το 8^{ο} είναι η τελεία) τα οποία τροφοδοτούνται ξεχωριστά, και είναι ανεξάρτητα το ένα από το άλλο. Το κόκκινο είναι το συνηθέστερο χρώμα. Η έξοδος παράγεται από διαφορετικούς συνδυασμούς των τμημάτων που αντιπροσωπεύουν τα νούμερα. Τροφοδοτώντας όλα τα τμήματα, εμφανίζεται ο

αριθμός 8, τροφοδοτώντας τα a,b,c,d και g εμφανίζεται ο αριθμός 3, το d.p. (decimal point) είναι η τελεία.

Τα 7-segment μπορούν να εμφανίσουν μόνο αριθμούς ή και αλφαριθμητικά. Μπορούν να οδηγηθούν από το Arduino απευθείας είτε μέσω ολοκληρωμένου κυκλώματος. Τα 7-segment χρησιμοποιούνται συχνά σε μικρές συσκευές ελέγχου εξοπλισμού (test equipment) για να εμφανίσουν αριθμητικές ενδείξεις. Οι οθόνες LCD κάνουν παρόμοια δουλειά καταναλώνοντας όμως λιγότερο ρεύμα.

Ο παραπάνω κώδικας εμφανίζει τους αριθμούς μετρώντας αντίστροφα από το 9 ως το 0 σε οθόνη ενός ψηφίου. Χρησιμοποιούμε ένα δυσδιάστατο πίνακα στον οποίον αποθηκεύουμε τα μοτίβα των 8 bit των 10 αριθμών που χρειάζεται το 7 Segment LED Display για να σχηματίσει τους αριθμούς. Διατρέχουμε τον πίνακα με χρήση δύο “for” βρόγχων για την εμφάνιση των αριθμών.

Συνδέουμε τις θύρες 3 και 8 της οθόνης ψηφίου με τη γείωση. Χρησιμοποιούμε από μία αντίσταση σε κάθε άλλη θύρα ειδάλλως αν συνδέσουμε απευθείας την οθόνη με το arduino υπάρχει κίνδυνος σταδιακά να καεί η οθόνη και το arduino. Η συνδεσμολογία εξαρτάται κάθε φορά από το hardware και τον κατασκευαστή του, γι’ αυτό το λόγο ελέγχουμε το φύλλο δεδομένων για το σωστό σχηματικό διάγραμμα. Η οθόνη κοινής ανόδου έχει αντίστροφη συνδεσμολογία όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 18: Διάγραμμα σύνδεσης οθόνης ψηφίου κοινής ανόδου και καθόδου.

2ο Εργαστηριακό μάθημα

ΗΧΟΣ

Audio Output (Sound, Melody) - Knock Sensor

Buzzer, Speaker, Piezo Element

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός :

Βομβητή.

Μεγαφώνου.

Πιεζοηλεκτρικού στοιχείου.

Εικόνα 16: Βομβητής, Μεγάφωνο, Πιεζοηλεκτρικός δίσκος.

Θεωρητική εισαγωγή

Ο ήχος είναι η αίσθηση που προκαλείται λόγω της διέγερσης των αισθητηρίων οργάνων της ακοής από τις μεταβολές πίεσης του ατμοσφαιρικού αέρα. Αυτές οι μεταβολές διαδίδονται με τη μορφή ηχητικών κυμάτων. Ένα ηχητικό κύμα χαρακτηρίζεται από φυσικές ιδιότητες όπως συχνότητα, περίοδος, μήκος κύματος, πλάτος ταλάντωσης, χρόνος και κυματομορφή. Από αυτές τις ιδιότητες πηγάζουν τέσσερα χαρακτηριστικά που αποσκοπούν στην περιγραφή ενός ήχου από μουσικο-ακουστικής προσέγγισης και είναι τα εξής: ύψος, ένταση, διάρκεια και χροιά.

Το Arduino μπορεί να παράγει ήχο μέσω συσκευών εξόδου όπως: μεγάφωνο (speaker), βομβητής (buzzer) και πιεζο-ηλεκτρικός κεραμικός δίσκος (piezo element), μετατρέποντας ηλεκτρικές ταλαντώσεις σε ηχητικά κύματα.

Συχνότητα ονομάζεται γενικά ο αριθμός των επαναλήψεων ενός γεγονότος στη μονάδα του χρόνου. Η συχνότητα (pitch) στον ήχο εκφράζει την ταχύτητα ταλάντωσης και μετράται σε κύκλους ανά δευτερόλεπτο (Hertz, Hz). Γρηγορότερες ταλαντώσεις επιφέρουν υψηλότερους - οξύτερους ήχους, ενώ βραδύτερες ταλαντώσεις επιφέρουν χαμηλότερους - βαρύτερους ήχους.

Η περίοδος είναι μέγεθος που χαρακτηρίζει εκείνα τα φυσικά φαινόμενα, τα οποία έχουν την ιδιότητα να επαναλαμβάνονται κατά τον ίδιο τρόπο μετά την πάροδο ορισμένου χρόνου. Τέτοια φαινόμενα ονομάζονται περιοδικά. Γνωρίζοντας τον χρόνο μίας περιόδου μπορεί να υπολογιστεί η συχνότητα από την σχέση:

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

όπου Τ είναι η περίοδος.

Ο μουσικά εξειδικευμένος όρος 'ύψος' δηλώνει πόσο υψηλός ή χαμηλός είναι ένας ήχος, χαρακτηριστικό που εξαρτάται από την έντονη παρουσία περιοδικών ταλαντώσεων.
Ως ένταση αποκαλείται το πόσο ισχυρή ή ασθενής είναι η ταλάντωση ενός σώματος. Πλατύτερες ταλαντώσεις επιφέρουν ηχητικά κύματα με μεγαλύτερη ένταση, σε σύγκριση με ταλαντώσεις μικρότερου πλάτους των οποίων το προϊόν είναι ήχοι ασθενέστεροι.
Η διάρκεια ορίζει τον συνολικό χρόνο για τον οποίο ένας ήχος γίνεται αντιληπτός. Ένας ήχος είναι μακρύτερος από έναν άλλο, βραχύτερο, όταν η αντιληπτή διάρκεια είναι συγκριτικά μεγαλύτερη.

Στον άνθρωπο η ακοή εκτείνεται για ήχους με συχνότητα μεταξύ 20 Hz και 20.000 Hz. Το εύρος αυτό διαφέρει και σε μεγαλύτερες ηλικίες παρατηρείται μείωση της αντίληψης υψηλών συχνοτήτων. Ήχοι με συχνότητα κάτω ή άνω των ορίων αυτών ονομάζονται υπόηχοι ή υπέρηχοι αντιστοίχως και δεν γίνονται αντιληπτοί από το ανθρώπινο αυτί. Σε άλλους οργανισμούς το φάσμα της ακοής διαφέρει - στον σκύλο το εύρος ακοής εκτείνεται μεταξύ 40 Hz και 60.000 Hz.

ΑΣΚΗΣΗ 2.1

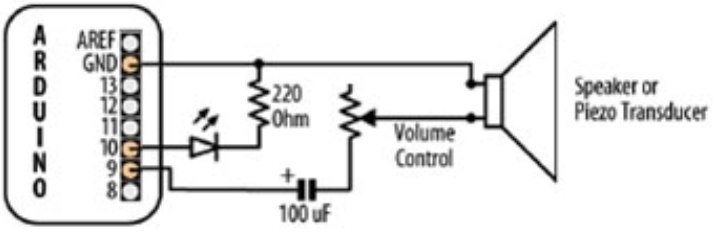
Σκοπός της Άσκησης:

Απόδοση ήχου μέσω ενός βομβητή ή μεγαφώνου και ταυτόχρονα η φωτοδίοδος να ανάβει και να σβήνει ομαλά σε συνάρτηση με τον τόνο που παράγεται.

Απαιτούμενα Υλικά:

1. Βομβητής (buzzer) 
2. Ποτενσιόμετρο 1ΚΩ
3. Αντίσταση 220 Ω 
4. LED 
5. Ηλεκτρολυτικός πυκνωτής 100μF 

Συνδεσμολογία:



Σχήμα 19: Συνδεσμολογία κυκλώματος βομβητή.

Πρόγραμμα 2.1

```
byte speakerPin = 9;
byte ledPin = 10;
```

```
void setup()
{
  pinMode(speakerPin, OUTPUT);
}

void playTone(int period, int duration)
{
// period is one cycle of tone
// duration is how long the pulsing should last in milliseconds
  int pulse = period / 2;
  for (long i = 0; i < duration * 1000L; i += period )
  {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(pulse);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(pulse);
  }
}
```

```
void fadeLED(){
 for (int brightness = 0; brightness < 255; brightness++)
 {
   analogWrite(ledPin, brightness);
   delay(2);
  }
  for (int brightness = 255; brightness >= 0; brightness--) {
    analogWrite(ledPin, brightness);
    delay(2);
  }

}

void loop()
{
  // a note with period of 15289 is deep C (second lowest C note on piano)
  for(int period=15289; period >= 477;  period=period / 2)  // play 6 octaves
  {
     playTone( period, 200); // play tone for 200 milliseconds
  }
  fadeLED();
}
```

Συμπεράσματα - Παρατηρήσεις

Η ένταση του ήχου ελέγχεται από μεταβλητή αντίσταση-ποτενσιόμετρο τιμής 100-1000 ohm, εναλλακτικά θα μπορούσε να χρησιμοποιηθεί μια απλή αντίσταση από 220ohm έως 680ohm. Η τιμή της αντίστασης δεν είναι κριτικής σημασίας και εξαρτάται από τα χαρακτηριστικά της συσκευής εξόδου.

Ο πυκνωτής που χρησιμοποιούμε είναι ηλεκτρολυτικός (έχει πολικότητα) χωρητικότητας 100μF και συνδέεται με το θετικό άκρο στην θύρα του arduino. Το μεγαφωνάκι θα δουλέψει ανεξαρτήτως πολικότητας σε αντίθεση με το πιεζοηλεκτρικό δισκάκι και το μπαζεράκι που έχουν συνήθως πολικότητα και πρέπει να συνδεθεί ο αρνητικός πόλος στη γείωση.

Τα μικρά μεγαφωνάκια, πιεζοηλεκτρικά δισκάκια, πιεζοηλεκτρικά μπάζερ και οι κοινοί βομβητές είναι πολύ φθηνά και μπορούν να βρεθούν εύκολα αφού παράγονται μαζικά. Τα πίεζο βρίσκονται στις μουσικές χριστουγεννιάτικες κάρτες και τα μπάζερ σε κάθε υπολογιστή.

Τα πίεζο είναι πολύ υψηλής αντίστασης και απαιτούν μικρό ρεύμα Ic έτσι συνδέονται απευθείας σε θύρα του arduino. Τα μεγαφωνάκια είναι συνήθως μικρότερης εσωτερικής αντίστασης και χρειάζεται να συνδεθούν σε σειρά με αντίσταση.

ΑΣΚΗΣΗ 2.2

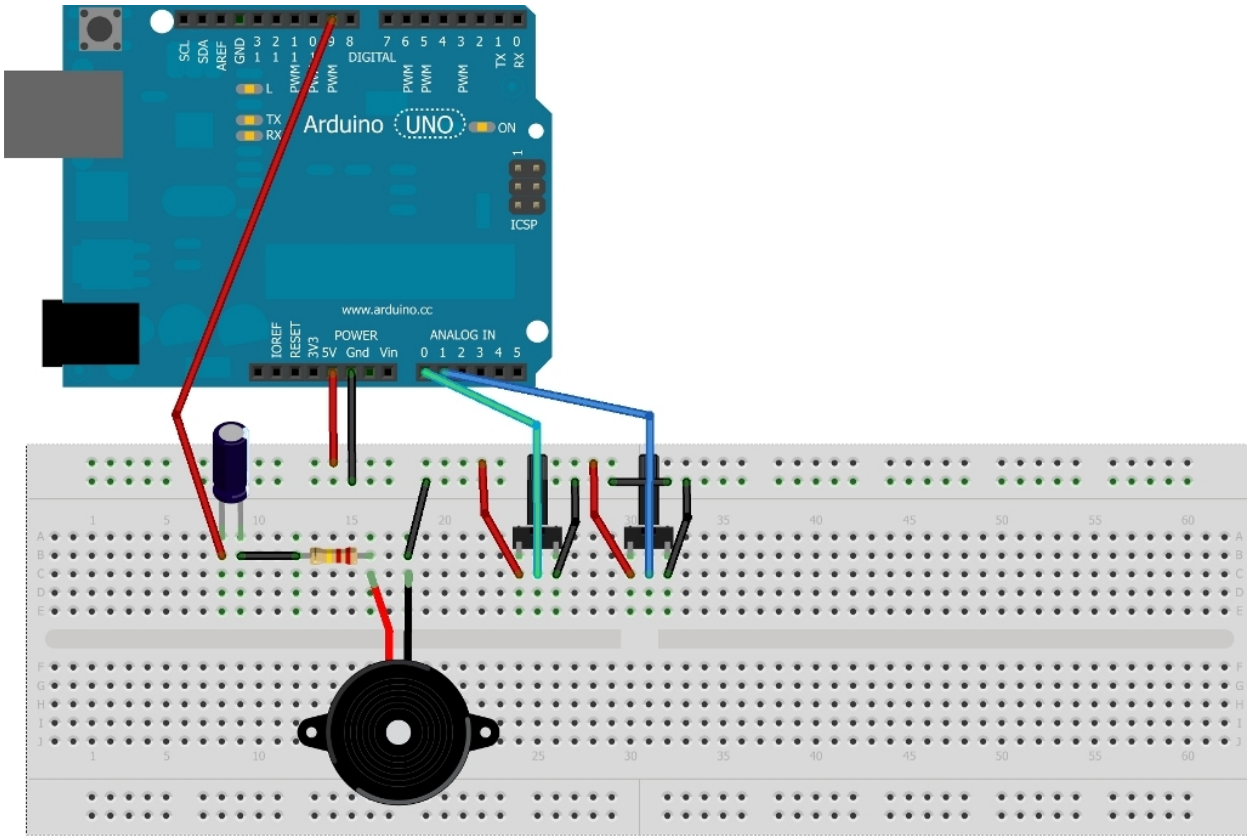
Σκοπός της Άσκησης:

Απόδοση ήχου μέσω ενός μεγάφωνου ή πιεζοηλεκτρικού δίσκου, ελέγχοντας τη συχνότητα και τη διάρκεια του τόνου με χρήση 2 ποτενσιόμετρων.

Απαιτούμενα Υλικά:

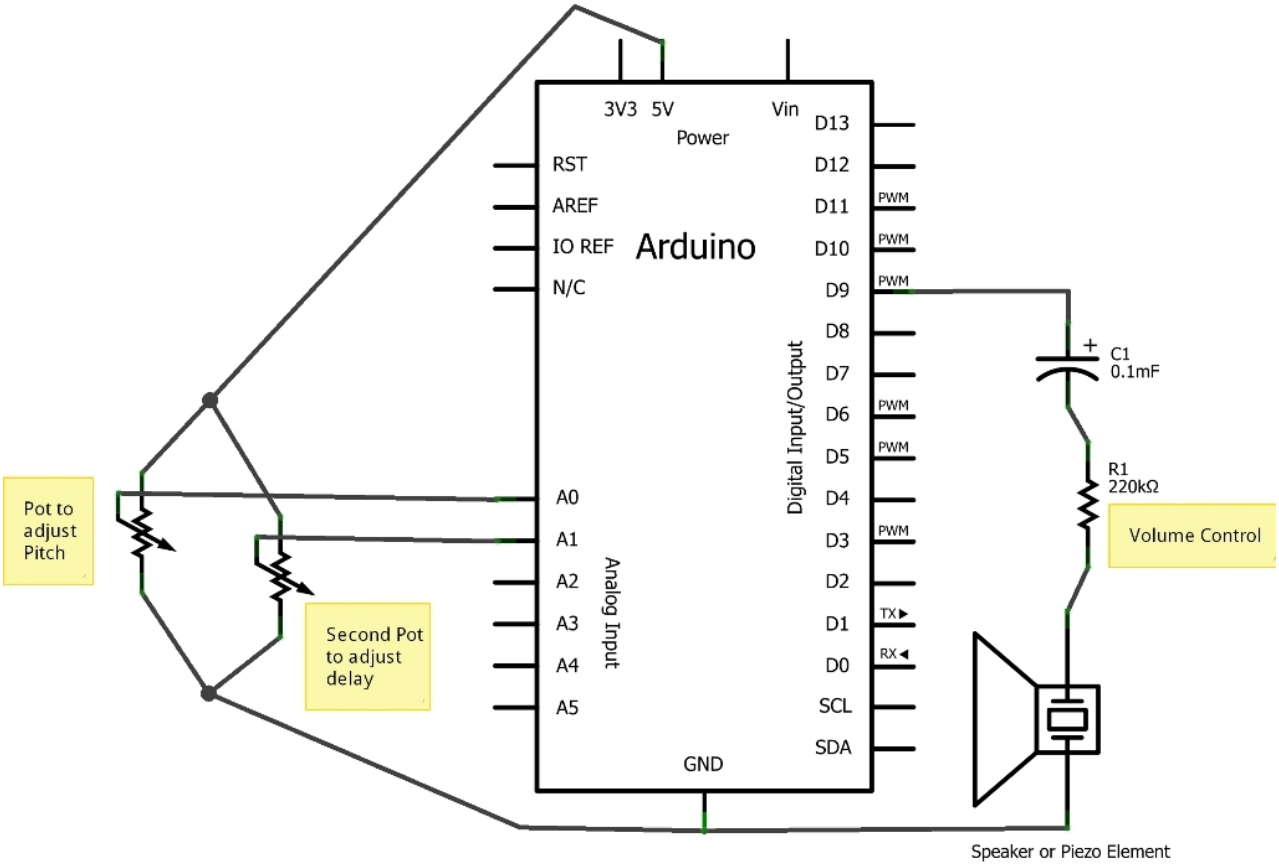
1. Μεγαφωνάκι 8 Ω 
2. 2x Ποτενσιόμετρα 5 ΚΩ 
3. Αντίσταση 220 Ω 
4. Ηλεκτρολυτικός πυκνωτής 100μF 

Συνδεσμολογία:



Σχήμα 20: Φυσική διάταξη του κυκλώματος σύνδεσης μεγαφώνου και 2 ποτενσιόμετρων.

Σχηματικό διάγραμμα:



Σχήμα 21: Συνδεσμολογία μεγαφώνου και 2 ποτενσιόμετρων.

Πρόγραμμα 2.2

```
/*
* Tone sketch
* Plays tones through a speaker on digital pin 9
* frequency determined by values read from analog port
*/
const int speakerPin = 9;     // connect speaker to pin 9
const int pitchPin = 0;       // input that determines frequency of the tone
const int durationPin = 1;    // input that will determine the duration of the
tone

void setup()
{
}

void loop()
{
   int sensor0Reading = analogRead(pitchPin);     // read input for frequency
   int sensor1Reading = analogRead(durationPin); // read input for duration

   // map the analog readings to a meaningful range
   int frequency  = map(sensor0Reading, 0, 1023, 100,5000); // 100Hz to 5kHz
   int duration = map(sensor1Reading, 0, 1023, 100,1000);   // dur 0.1-1 second
   tone(speakerPin, frequency, duration); // play the tone
   delay(duration); //wait for the tone to finish
}
```

Περιγραφή της Άσκησης

Χρησιμοποιούμε την tone συνάρτηση, η οποία βρίσκεται στην ενσωματωμένη βιβλιοθήκη <arduino.h>. Παράγεται ένας τόνος στη συχνότητα που θέτει το ποτενσιόμετρο (ή οποιαδήποτε άλλη μεταβλητή αντίσταση) συνδεδεμένο στη αναλογική θύρα 0. Η διάρκεια του τόνου ελέγχεται από το ποτενσιόμετρο στη θύρα 1.

Η συνάρτηση tone δέχεται 3 ορίσματα: τη θύρα του μεγαφώνου, τη συχνότητα σε Hz και τη διάρκεια σε ms. tone(speakerPin, frequency, duration);

Η διάρκεια είναι προαιρετική, αν παραληφθεί ο τόνος συνεχίζει μέχρι να κληθεί άλλος τόνος ή η συνάρτηση noTone().

Λογικές τιμές για τη συχνότητα (100Hz to 5kHz) που αντιλαμβάνονται από το ανθρώπινο αυτί αλλά και αποδίδονται από τις εκάστοτε συσκευές αντιστοιχίζονται με τη κλήση της map().

int frequency = map(sensor0Reading, 0, 1023, 100, 5000);

Piezo Knock Sensor

Το πίεζο ως αισθητήρας πίεσης, αφής, δονήσεων

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας και ο προγραμματισμός του πιεζοηλεκτρικού δίσκου.

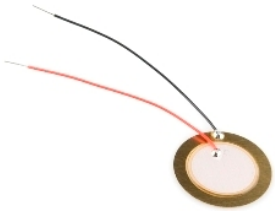
Θεωρητική εισαγωγή

Ο αισθητήρας που χρησιμοποιούμε ανήκει στους πιεζοηλεκτρικούς μετατροπείς. Αρχή λειτουργίας των αισθητήρων αυτού του είδους είναι το πιεζοηλεκτρικό φαινόμενο το οποίο ανακαλύφθηκε από τους Pierre και Jacques Curie το 1880. Σύμφωνα με το φαινόμενο αυτό, η παραμόρφωση ενός κρυστάλλου λόγω μηχανικής πίεσης σε αυτόν προκαλεί επαναπροσανατολισμό του φορτίου και αντίστοιχα την μετακίνηση ίσων θετικών και αρνητικών φορτίων στις αντίθετες πλευρές του κρυστάλλου. Αποτέλεσμα των παραπάνω διεργασιών είναι η ανάπτυξη τάσης στα άκρα του κρυστάλλου ανάλογη της παραμόρφωσης που την προκάλεσε.

Μια σημαντική ιδιότητα του φαινομένου είναι η αντιστρεψιμότητα του, δηλαδή το γεγονός ότι η εφαρμογή τάσης στα άκρα του κρυστάλλου είναι ικανή να προκαλέσει την παραμόρφωση του. Το αντίστροφο αυτό φαινόμενο βρίσκει ευρύτατη εφαρμογή στη σχεδίαση ενεργοποιητών (actuators). Η μέτρηση της παραμόρφωσης του πιεζοηλεκτρικού κρυστάλλου μπορεί να πραγματοποιηθεί μετρώντας είτε το φορτίο που αναπτύσσεται είτε την τάση που προκύπτει στα άκρα του φορτίου. Ωστόσο το φαινόμενο που παρατηρείται (λόγω διαρροής του φορτίου μέσω του κρυστάλλου (αυτοεκφόρτιση) ακόμα και στην περίπτωση που η παραμόρφωση παραμένει σταθερή) η τάση να ακολουθεί την καμπύλη εκφόρτισης και να τείνει ασυμπτωτικά στο μηδέν,

καθιστά τη χρήση των κρυστάλλων κυρίως για τη μέτρηση δυναμικών παραμορφώσεων, πιέσεων και δυνάμεων.

ΑΣΚΗΣΗ 2.3



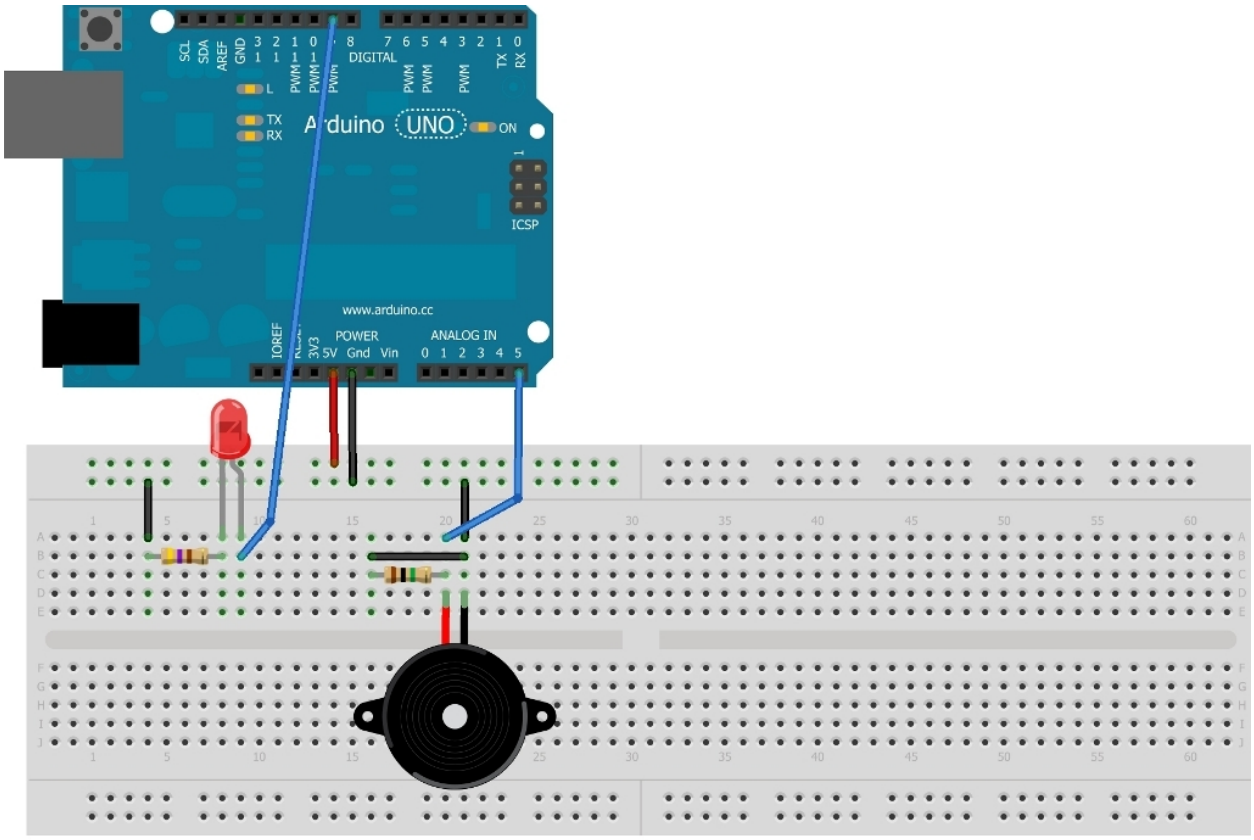
Σκοπός της Άσκησης:

Ανίχνευση δόνησης με χρήση πιεζοηλεκτρικού δίσκου.

Απαιτούμενα Υλικά:

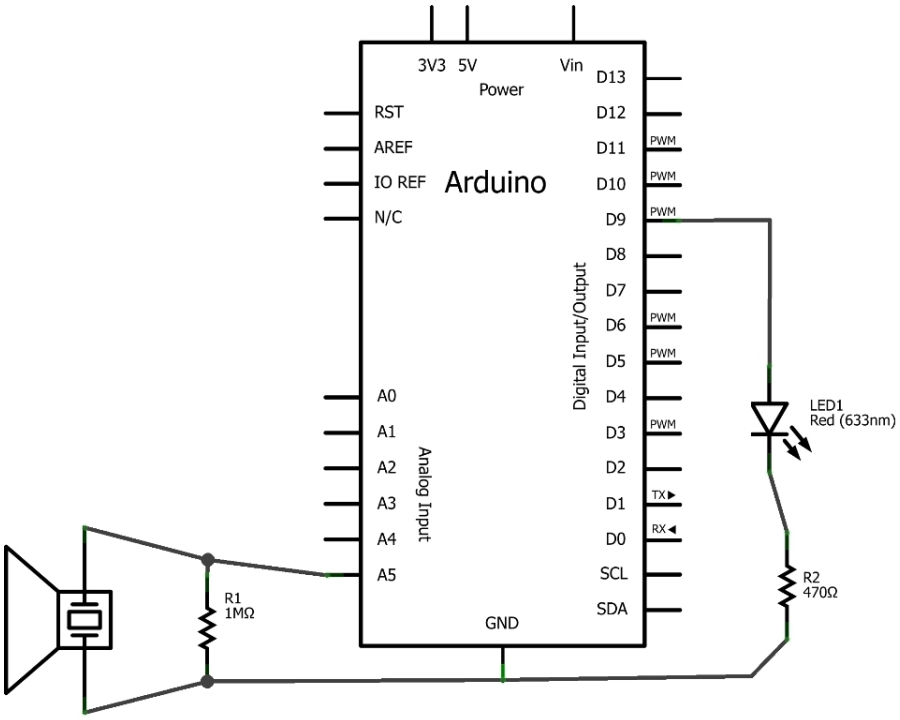
1. Πιεζοηλεκτρικός δίσκος
2. 5mm LED 
3. 1MΩ Resistor 

Συνδεσμολογία:



Σχήμα 22: Φυσική διάταξη του κυκλώματος πιεζοηλεκτρικού δίσκου.

Σχηματικό διάγραμμα:



Σχήμα 23: Συνδεσμολογία κυκλώματος πιεζοηλεκτρικού δίσκου.

Πρόγραμμα 2.3

```
const int ledPin = 9; // LED on Digital Pin 9
const int piezoPin = 5; // Piezo on Analog Pin 5
const int threshold = 60; // The sensor value to reach before activation
int sensorValue = 0; // A variable to store the value read from the sensor
float ledValue = 0; // The brightness of the LED

void setup()
{
      pinMode(ledPin, OUTPUT); // Set the ledPin to an OUTPUT
      // Flash the LED twice to show the program has started
      digitalWrite(ledPin, HIGH);
      delay(150);
      digitalWrite(ledPin, LOW);
      delay(150);
      digitalWrite(ledPin, HIGH);
      delay(150);
      digitalWrite(ledPin, LOW);
      delay(150);
}
void loop()
{
      sensorValue = analogRead(piezoPin); // Read the value from the sensor
      if (sensorValue >= threshold)
      { // If knock detected set brightness to max
            ledValue = 255;
      }
      analogWrite(ledPin, int(ledValue) ); // Write brightness value to LED
      ledValue = ledValue - 0.05; // Dim the LED slowly
      if (ledValue <= 0)
      {
            ledValue = 0;
      } // Make sure value does not go below zero
}
```

Περιγραφή της Άσκησης - Συμπεράσματα:

Το πιεζο-ηλεκτρικό στοιχείο είναι πρακτικό σε περιπτώσεις ανίχνευσης κραδασμών ή χτυπημάτων. Μπορεί να χρησιμοποιηθεί για την ανίχνευση επαφής η χτυπήματος αρκετά εύκολα, απλά μετρώντας με το arduino την αλλαγή στην τάση της εξόδου.

Ο πιεζοηλεκτρικός δίσκος αποτελείται από μια λεπτή ταινία πιεζοηλεκτρικού κρυστάλλου ή αλλιώς πιεζοηλεκτρική κάψα. Όταν διοχετεύουμε ρεύμα μέσα από το κεραμικό υλικό του δίσκου προκαλείται αλλαγή στο σχήμα του που δημιουργεί τον ήχο. Ο δίσκος όμως δουλεύει και αντίστροφα, όταν δέχεται χτύπημα ή πίεση ή δόνηση, ο πιεζοηλεκτρικός κρύσταλλος μετατρέπει την πίεση σε ηλεκτρικό ρεύμα. Δηλαδή οι δονήσεις που συλλαμβάνονται από τον κρύσταλλο, τον διεγείρουν και παράγει αντίστοιχο ηλεκτρικό σήμα.

Μόλις φορτωθεί ο κώδικας το led αναβοσβήνει γρήγορα 2 φορές για να δηλώσει την εκκίνηση του προγράμματος. Χτυπάμε το δισκάκι με το δάκτυλο αφού το τοποθετήσουμε σε επίπεδη επιφάνεια. Κάθε φορά το led ανάβει και σβήνει ομαλά (fade). Η τιμή threshold αντιπροσωπεύει την ευαισθησία στο άγγιγμα και εξαρτάται από το υλικό, τον τύπο και το μέγεθος του πίεζο. Μπορεί να χρειαστεί να τεθεί μεγαλύτερη ή μικρότερη τιμή. Όταν η ευαισθησία ξεπερνά ένα όριο τότε το led παραμένει μονίμως αναμμένο. Μικρότερη τιμή στο threshold βελτιώνει την ευαισθησία και το αντίθετο.

Το πρόγραμμα διαβάζει την αναλογική θύρα του πίεζο και συγκρίνει το αποτέλεσμα με το όριο threshold. Αν το αποτέλεσμα είναι μεγαλύτερο θέτει "knock" στη σειριακή πόρτα και μεταβάλλει την κατάσταση του LED στη θύρα 9.

3ο Εργαστηριακό μάθημα

ΚΙΝΗΤΗΡΕΣ

DC motor (Κινητήρας συνεχούς ρεύματος)

Stepper motor (Βηματικός κινητήρας)

Servo motor (Σερβοκινητήρας)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός :

Κινητήρα συνεχούς ρεύματος.

Βηματικού κινητήρα.

Σερβοκινητήρα.

Θεωρητική εισαγωγή

Ο ηλεκτρικός κινητήρας ή (motor, κοινώς μοτέρ) είναι διάταξη που χρησιμοποιείται για την μετατροπή της ηλεκτρικής ενέργειας σε μηχανική ενέργεια. Οι ηλεκτρικοί κινητήρες αποτελούν μια κατηγορία ηλεκτρικών μηχανών που χρησιμοποιούνται κυρίως στο βιομηχανικό τομέα για να μεταδίδουν κίνηση στα διάφορα μηχανήματα αναλόγως της εφαρμογής τους.

Σ' έναν απλό ηλεκτροκινητήρα συνεχούς ρεύματος, το ηλεκτρικό ρεύμα διαρρέει μια συρμάτινη περιέλιξη (θηλιά), η οποία βρίσκεται ανάμεσα στους πόλους ενός ηλεκτρομαγνήτη. Όμως κάθε ρευματοφόρος αγωγός που βρίσκεται μέσα σε μαγνητικό πεδίο, δέχεται κάποια δύναμη. Στην περίπτωση του ηλεκτρικού κινητήρα συνεχούς ρεύματος, οι δυνάμεις που ασκούνται στην περιέλιξη, η οποία διαρρέεται από ρεύμα και ευρίσκεται μέσα σε μαγνητικό πεδίο, σπρώχνουν τη μια πλευρά της προς τα πάνω και την άλλη προς τα κάτω, με αποτέλεσμα αυτή να περιστρέφεται. Γι' αυτό και η συρμάτινη περιέλιξη λέγεται και «ρότορας». Ο ρότορας καταλήγει σε ένα μεταγωγέα, ο οποίος αντιστρέφει τη φορά του ρεύματος που τροφοδοτείται από τις ψύκτρες, δύο φορές σε κάθε περιστροφή, έτσι ώστε να εξασφαλίζεται σταθερή φορά περιστροφής. Ο ηλεκτρομαγνήτης που παράγει το μαγνητικό πεδίο μέσα στον κινητήρα ονομάζεται «στάτορας».

Οι ηλεκτροκινητήρες διακρίνονται σε «συνεχούς ρεύματος» (DC motors) και σε «εναλλασσόμενου ρεύματος» (AC motors) που διακρίνονται επίσης σε μονοφασικούς και τριφασικούς κινητήρες.

Οι ηλεκτροκινητήρες **εναλλασσόμενου ρεύματος** διακρίνονται επιμέρους στους «σύγχρονους κινητήρες» και στους «ασύγχρονους» ή «επαγωγικούς κινητήρες». Σύγχρονοι κινητήρες είναι οι κινητήρες στους οποίους η μέση ταχύτητα περιστροφής είναι ευθέως ανάλογη της συχνότητας της εφαρμοζόμενης εναλλασσόμενης τάσης. Οι «ασύγχρονοι» ή «επαγωγικοί κινητήρες» διακρίνονται σε μονοφασικούς και τριφασικούς και είναι οι πιο απλοί σε κατασκευή και γενικότερα από τις διατάξεις που εξελίχθηκαν ραγδαία από τις αρχές του περασμένου αιώνα, λόγω και της ευκολίας στη λειτουργία τους. Οι επαγωγικοί κινητήρες ονομάζονται έτσι καθώς η αρχή

λειτουργίας τους στηρίζεται στο φαινόμενο της επαγωγής, ονομάζονται και ασύγχρονοι καθώς περιστρέφονται με την ασύγχρονη ταχύτητα ns, η οποία είναι μικρότερη από τη σύγχρονη ταχύτητα n (ns < n). Οι τόσο απλοί στην κατασκευή τους «ασύγχρονοι» ή «επαγωγικοί κινητήρες» αποτελούνται από δύο κύρια μέρη: (α) το σταθερό μέρος του κινητήρα, ο στάτης ή στάτορας και (β) το κινούμενο μέρος του κινητήρα, ο δρομέας ή ρότορας.

Οι κινητήρες **συνεχούς ρεύματος** (dc motors) είναι μονοφασικοί. Η αρχή λειτουργία τους βασίζεται στον ηλεκτρομαγνητισμό. Οι κινητήρες συνεχούς ρεύματος ή DC Motors βρίσκονται σχεδόν παντού: από τα τηλεκατευθυνόμενα αυτοκινητάκια, τα κασετόφωνα, τους ανεμιστήρες μέχρι τα μηχανάκια κατασκευής φραπέ. Πρόκειται για την πιο οικονομική λύση για όσους σκέφτονται να υλοποιήσουν εφαρμογές όπως ρομπότ, βραχίονες, δαγκάνες και πάρα πολλά άλλα χρησιμοποιώντας το Arduino.

Τα μεγέθη που χαρακτηρίζουν τους ηλεκτροκινητήρες είναι:

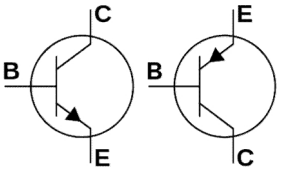
- **Τάση λειτουργίας**, μετράται σε Volts και είναι η τάση που πρέπει να έχει το ηλεκτρικό ρεύμα τροφοδοσίας ώστε ο κινητήρας να λειτουργεί σωστά.
- **Ένταση του ρεύματος**, μετράται σε Amps και είναι η ένταση του ρεύματος που διαρρέει τον κινητήρα όταν αυτός λειτουργεί. Η ένταση αυτή είναι ανάλογη του φορτίου του κινητήρα. Η ελάχιστη τιμή της αντιστοιχεί στην ελεύθερη περιστροφή του κινητήρα. Εάν ο κινητήρας έχει φορτίο η τιμή αυτή αυξάνεται. Για κάποια τιμή του φορτίου ο κινητήρας σταματά να περιστρέφεται, λόγω της μεγάλης αντίστασης, οπότε και η τιμή της έντασης μεγιστοποιείται. Η μέγιστη αυτή τιμή είναι ένα από τα σημαντικά χαρακτηριστικά του κινητήρα που πρέπει να είναι γνωστό για τη σωστή επιλογή της τροφοδοσίας του.
- **Ταχύτητα**, στους κινητήρες συνεχούς ρεύματος μετράται σε στροφές ανά λεπτό, στους σερβοκινητήρες σε μοίρες ανά λεπτό και στους βηματικούς κινητήρες σε βήματα ανά δευτερόλεπτο. Πρόκειται για την ταχύτητα περιστροφής του άξονα του κινητήρα, όταν ο κινητήρας λειτουργεί υπό κανονική ηλεκτρική τάση και με δεδομένο φορτίο.
- **Ροπή**, μετράται σε Nm (Newton meters) και είναι η ροπή που παράγει ο κινητήρας στις διάφορες ταχύτητες περιστροφής του. Η μέγιστη τιμή ροπής κάθε κινητήρα, που ονομάζεται ροπή ακινητοποίησης, είναι η ροπή που παράγει όταν το φορτίο που αντιμετωπίζει είναι τόσο μεγάλο, ώστε να το ακινητοποιεί.

Τρανζίστορ

Το **τρανζίστορ** (transistor), ελλ. **κρυσταλλοτρίοδος** και (παλαιότερα) κρυσταλλολυχνία, είναι διάταξη ημιαγωγών στερεάς κατάστασης, η οποία βρίσκει διάφορες εφαρμογές στην ηλεκτρονική, μερικές εκ των οποίων είναι: η ενίσχυση, η σταθεροποίηση τάσης, η διαμόρφωση συχνότητας, η λειτουργία ως διακόπτης και ως μεταβλητή ωμική αντίσταση. Το τρανζίστορ μπορεί, ανάλογα με την τάση με την οποία πολώνεται, να ρυθμίζει τη ροή του ηλεκτρικού ρεύματος που

απορροφά από συνδεδεμένη πηγή τάσης. Τα τρανζίστορ κατασκευάζονται είτε ως ξεχωριστά ηλεκτρονικά εξαρτήματα είτε ως τμήματα κάποιου ολοκληρωμένου κυκλώματος.

Το τρανζίστορ θεωρείται μία από τις μεγαλύτερες εφευρέσεις του 20ου αιώνα. Είναι το κυριότερο συστατικό όλων σχεδόν των σύγχρονων ηλεκτρονικών κατασκευών. Η πλατιά χρήση του οφείλεται κυρίως στη δυνατότητα παραγωγής του σε τεράστιες ποσότητες που μειώνουν το κόστος ανά μονάδα. Παρόλο που αρκετοί παραγωγοί παράγουν μεμονωμένες συσκευασίες τρανζίστορ, η μεγαλύτερη ποσότητα παράγεται μέσα σε ολοκληρωμένα κυκλώματα (που συχνά αναφέρονται ως τσιπς) μαζί με τις διόδους, αντιστάσεις, πυκνωτές και άλλα ηλεκτρονικά εξαρτήματα.



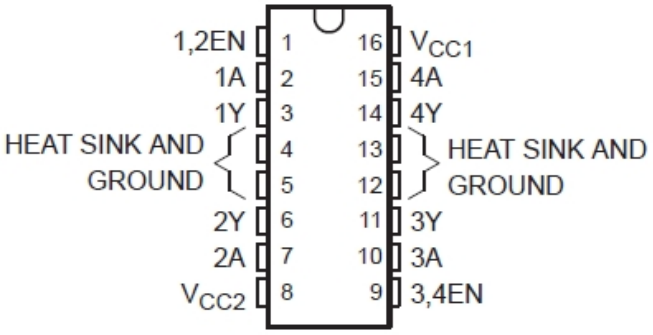
Εικόνα 17: Τρανζίστορ npn και pnp.

Το τρανζίστορ επαφής είναι μία διάταξη η οποία αποτελείται από δύο ημιαγωγούς, έναν τύπου PN και έναν τύπου NP, όπου ο ένας πολώνεται κατά την ορθή φορά και ο άλλος κατά την ανάστροφη. Υπάρχουν δύο κατηγορίες τρανζίστορ επαφής: Το τρανζίστορ PNP και το τρανζίστορ NPN. Σε κάθε τύπο τα δύο ακραία τμήματα έχουν τον ίδιο τύπο ημιαγωγού. Στην παραπάνω εικόνα βλέπουμε τον συμβολισμό ενός τρανζίστορ PNP και ενός τρανζίστορ NPN. Ο ακροδέκτης που έχει το βελάκι είναι ο **εκπομπός**, ο μεσαίος ακροδέκτης είναι η **βάση** και ο τρίτος ακροδέκτης είναι ο **συλλέκτης**.

SN754410 H-Bridge (motor driver)

Ο SN754410 H-Bridge είναι ένας τετραπλός υψηλής τάσης οδηγός (τύπου Η) που είναι σχεδιασμένος να παρέχει αμφίδρομα ρεύματα πάνω από 1A και σε τάσεις από 4.5V ως 36V. Η συσκευή είναι σχεδιασμένη να μεταφέρει επαγωγικά φορτία σε ρελέ και διπολικούς κινητήρες καθώς και άλλα υψηλής τάσης φορτία σε διάφορες εφαρμογές. Όλες οι είσοδοι του είναι συμβατές με TTL (transistor - transistor logic) και χαμηλού επιπέδου λογικής CMOS. Κάθε έξοδος (Y) είναι ένας totem-pole οδηγός με μια αντίσταση Darlington και μια ψευδό-Darlington πηγή. Οι οδηγοί ενεργοποιούνται σε ζευγάρια. Δηλαδή οι οδηγοί 1 και 2 να ενεργοποιούνται με το 1,2 EN και οι οδηγοί 3 και 4 να ενεργοποιούνται με το 3,4 EN. Όταν η είσοδος είναι high οι αντίστοιχοι οδηγοί ενεργοποιούνται και οι έξοδοι τους δουλεύουν συμφασικά με τις εισόδους τους. Όταν η είσοδος είναι low, οι οδηγοί απενεργοποιούνται και μπαίνουν σε κατάσταση υψηλής αντίστασης. Με τις κατάλληλες εισόδους δεδομένων, κάθε ζευγάρι των οδηγών σχηματίζουν ένα πλήρες Η (ή γέφυρα) το οποίο είναι κατάλληλο για την εγκατάσταση κινητήρων. Μια ξεχωριστή παροχή ρεύματος (VCC1) παρέχεται για την λογικές εισόδους και την ελαχιστοποίηση της επαγωγής του ρεύματος. Η παροχή (VCC2) χρησιμοποιείται για τα κυκλώματα εξόδου.

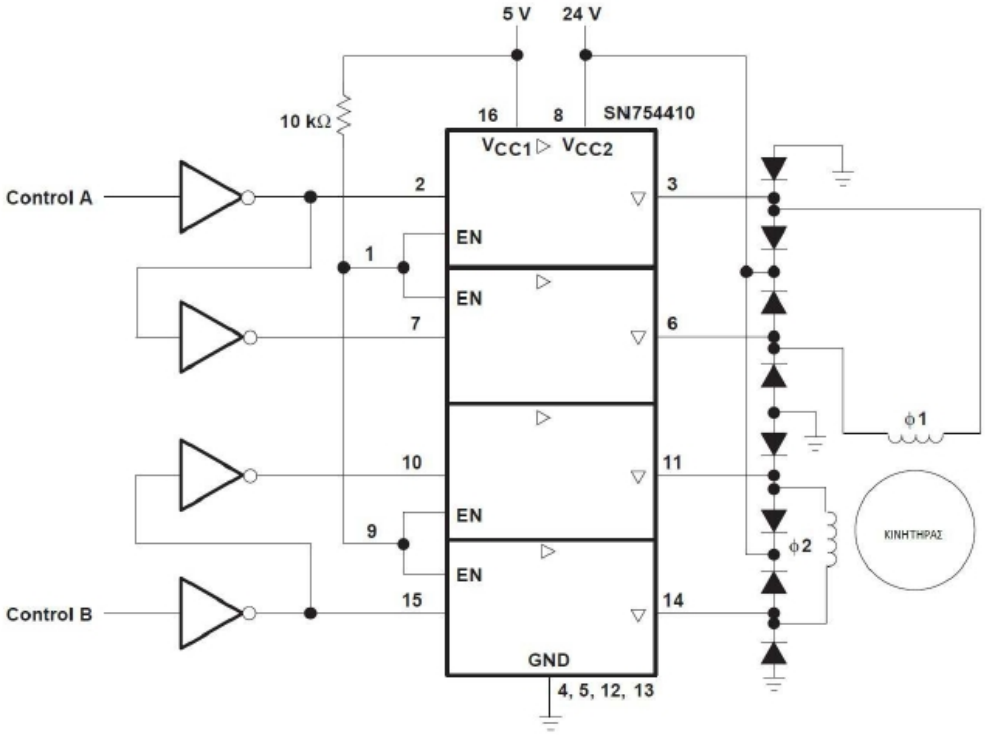
Πίνακας 4: Λειτουργία για κάθε οδηγό του H-Bridge Motor Driver



ΠΙΝΑΚΑΣ ΛΕΙΤΟΥΡΓΙΑΣ ΓΙΑ ΚΑΘΕ ΟΔΗΓΟ

INPUTS		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)



Σχήμα 24: Κύκλωμα λειτουργίας διπλού ελέγχου κινητήρα.

ΑΣΚΗΣΗ 3.1

Σκοπός της Άσκησης:

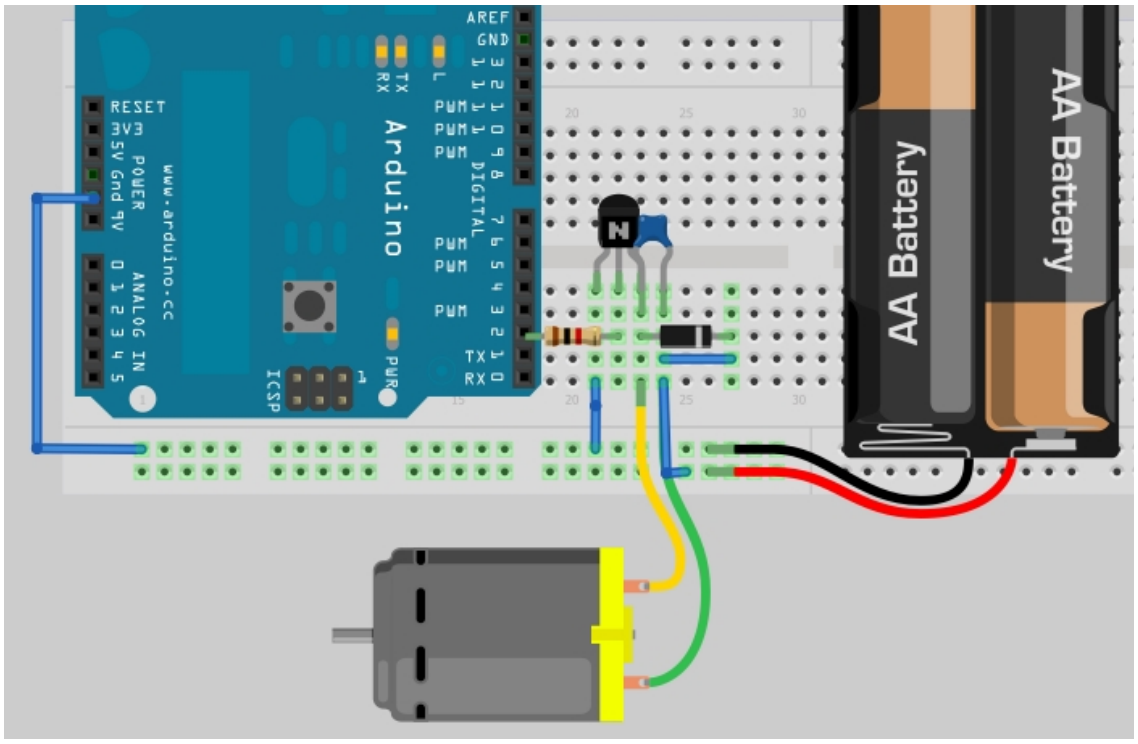
Σύνδεση ενός μικρού κινητήρα στο arduino (brushed dc motor), και έλεγχος μέσα από αυτό της ταχύτητάς του. Ο κινητήρας θα κινείται προς μία μόνο κατεύθυνση.

Απαιτούμενα Υλικά:

1. Brushed DC motor 1,5 - 9 V 
2. Τρανζίστορ οικογενείας TIP102 ή TIP120 
3. Αντίσταση 1 ΚΩ 
4. Δίοδος 1N4001 

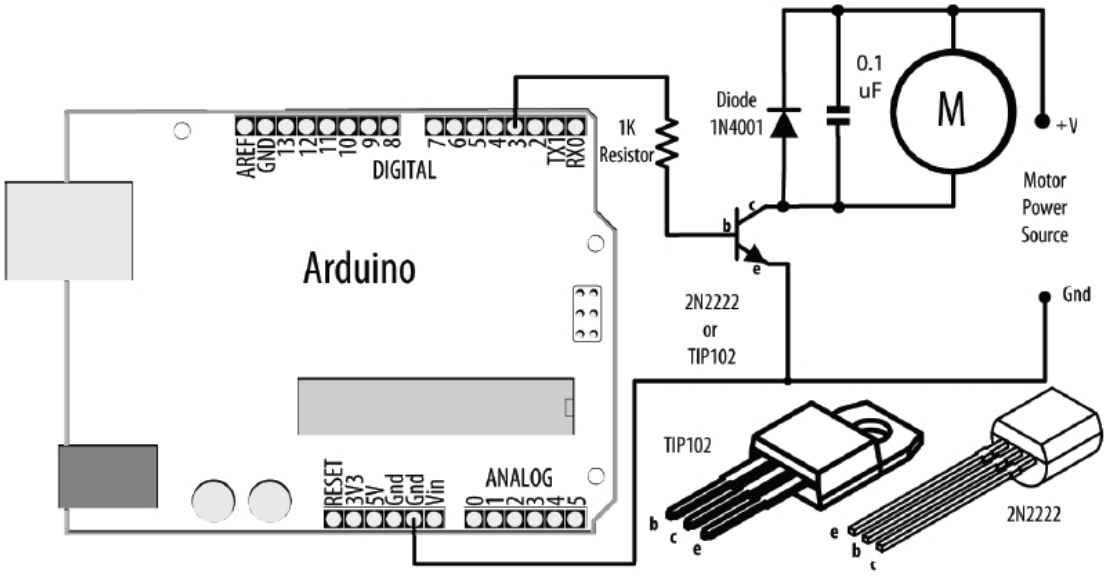
5. Κεραμικός πυκνωτής 0.1μF 
6. Ποτενσιόμετρο 5 ΚΩ 

Συνδεσμολογία:



Σχήμα 25: Φυσική διάταξη του κυκλώματος κινητήρα συνεχούς ρεύματος.

Σχηματικό διάγραμμα:



Σχήμα 26: Συνδεσμολογία κινητήρα συνεχούς ρεύματος.

Πρόγραμμα 3.1

```
/*
 * Sketch for a Simple Brushed Motor
 * commands from serial port and control motor speed
```

```
 * digits '0' through '9' are valid where '0' is off, '9' is max speed
 */

const int motorPin = 3; // motor driver is connected to pin 3

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

void loop()
{
  if ( Serial.available()) {
    char ch = Serial.read();

    if(isDigit(ch))              // is ch a number?
    {
      int speed = map(ch, '0', '9', 0, 255);
      analogWrite(motorPin, speed);
      Serial.println(speed);
    }
    else
    {
      Serial.print("Unexpected character ");
      Serial.println(ch);
    }
  }
}
```

Συμπεράσματα - Παρατηρήσεις

Οι εντολές ελέγχου της ταχύτητας δίνονται μέσω της σειριακής σύνδεσης από τη γραφική διεπαφή του περιβάλλοντος Arduino IDE (integrated development environment). Δίνουμε αριθμούς από το 0 έως το 9 (0 απενεργοποιημένο μοτέρ, 9 μέγιστη ταχύτητα) στο Serial Monitor, για οποιονδήποτε άλλο χαρακτήρα εμφανίζεται ανάλογο μήνυμα.

Η επιλογή του τρανζίστορ εξαρτάται από τα ποσότητα του ρεύματος που απαιτείται για να δουλέψει ο κινητήρας. Ελέγχουμε για αυτό το σκοπό το φύλλο δεδομένων (data sheet). Για ένα μικρό κινητήρα μπορούμε να επιλέξουμε το πολύ κοινό P2N2222A που παρέχει μέχρι 200ma συνεχούς ρεύματος. Για κινητήρες με μεγαλύτερες απαιτήσεις διαλέγουμε τρανζίστορ της οικογενείας TIP102/TIP120 που αποδίδουν μέχρι 5A. Ο σκοπός της διόδου είναι να προστατέψει το κύκλωμα εμποδίζοντας το ανάστροφο ρεύμα EMF (που παράγεται από τον κινητήρα καθώς περιστρέφεται) να εισέλθει πίσω στο τρανζίστορ και να βλάψει το ίδιο ή και το arduino. Η πολικότητα της διόδου είναι φυσικά σημαντική. Ο πυκνωτής απορροφά τις απότομες μεταβολές της τάσης που παράγονται όταν ενεργοποιείται και απενεργοποιείται ο brushed κινητήρας.

Αντίσταση χωρητικότητας 1ΚΩ με 5ΚΩ συνδέει τη ψηφιακή θύρα εξόδου του arduino με τη βάση του τρανζίστορ, η τιμή της δεν είναι κριτικής σημασίας και υπάρχει για να περιορίσει το ρεύμα που θα τραβήξει το τρανζίστορ.

Παρατηρούμε ότι ο κινητήρας τροφοδοτείται από εξωτερική πηγή ενέργειας. Αυτό γίνεται γιατί το arduino προορίζεται για χαμηλής κατανάλωσης κυκλώματα, συγκεκριμένα από τις θύρες εισόδου-εξόδου διαχειρίζεται ρεύματα μέχρι 40ma, ενώ από τη θύρα 5V τροφοδοτεί με 500mA. Το

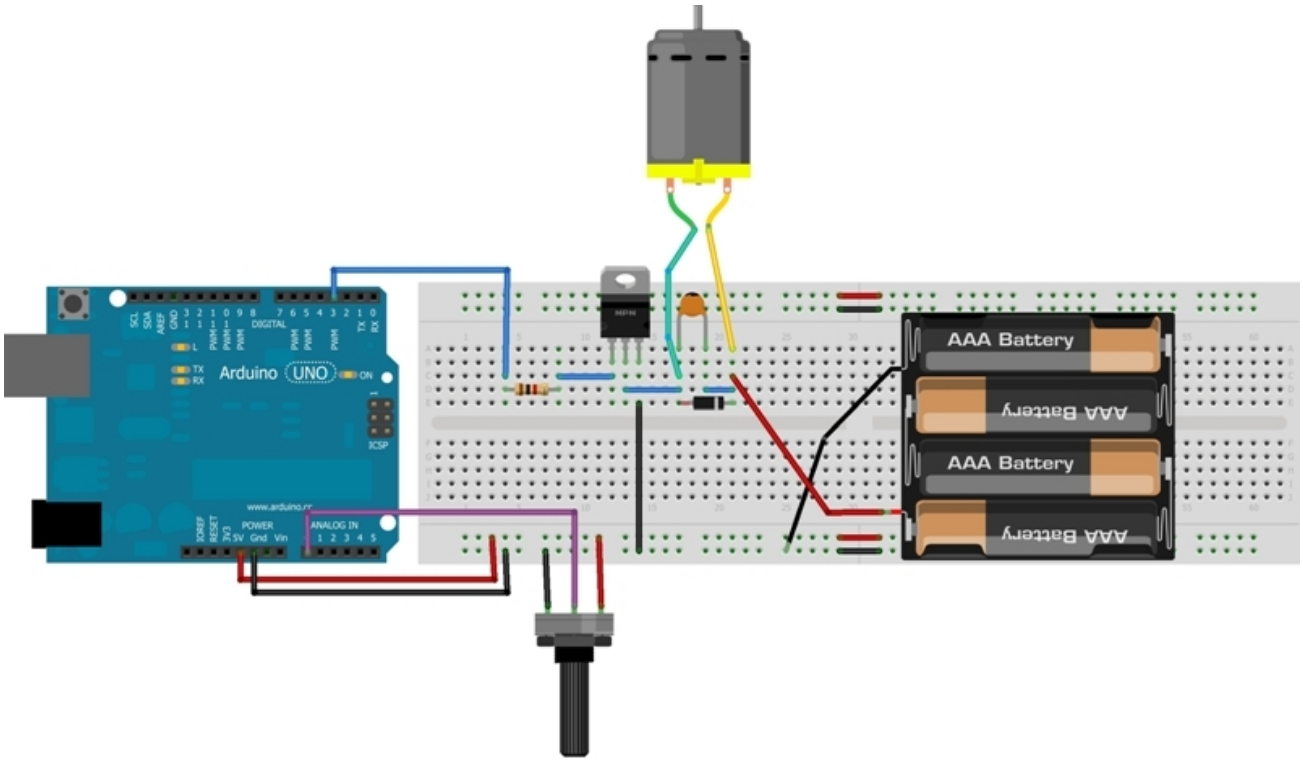
ίδιο το arduino μπορεί να τροφοδοτηθεί από τη USB θύρα ή τροφοδοτικό AC-DC 9-12V ή 9V μπαταρία.

ΑΣΚΗΣΗ 3.2

Σκοπός της Άσκησης:

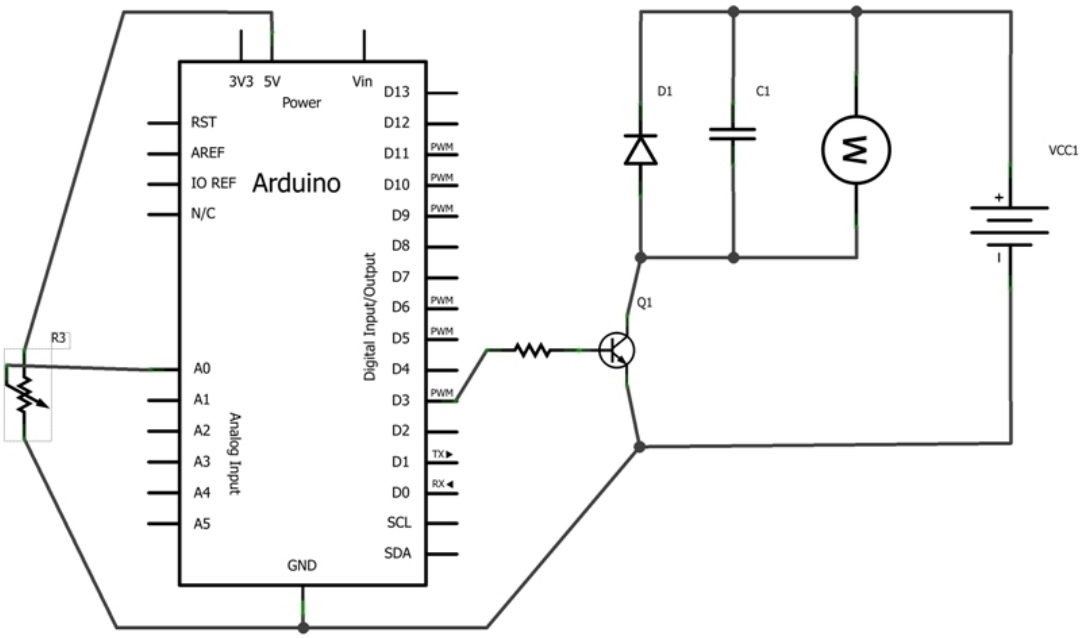
Εδώ πετυχαίνουμε την ίδια λειτουργία με την προηγούμενη άσκηση, κατασκευάζοντας παρόμοιο κύκλωμα αλλά αυτή τη φορά για τον έλεγχο της ταχύτητας χρησιμοποιείται το ποτενσιόμετρο.

Συνδεσμολογία:



Σχήμα 27: Φυσική διάταξη του κυκλώματος
κινητήρα συνεχούς ρεύματος και έλεγχος με το ποτενσιόμετρο.

Σχηματικό διάγραμμα:



Σχήμα 28: Συνδεσμολογία κυκλώματος
κινητήρα συνεχούς ρεύματος με έλεγχο από ποτενσιόμετρο.

Πρόγραμμα 3.2

```
const int potPin = 0; // Analog in 0 connected to the potentiometer
const int transistorPin = 11; // connected to the base of the transistor
int potValue = 0; // value returned from the potentiometer

void setup()
{
      // set the transistor pin as output:
      pinMode(transistorPin, OUTPUT);
}
void loop()
{
      // read the potentiometer, convert it to 0 - 255:
      potValue = analogRead(potPin) / 4;
      // use that to control the transistor:
      analogWrite(transistorPin, potValue);
}
```

ΑΣΚΗΣΗ 3.3

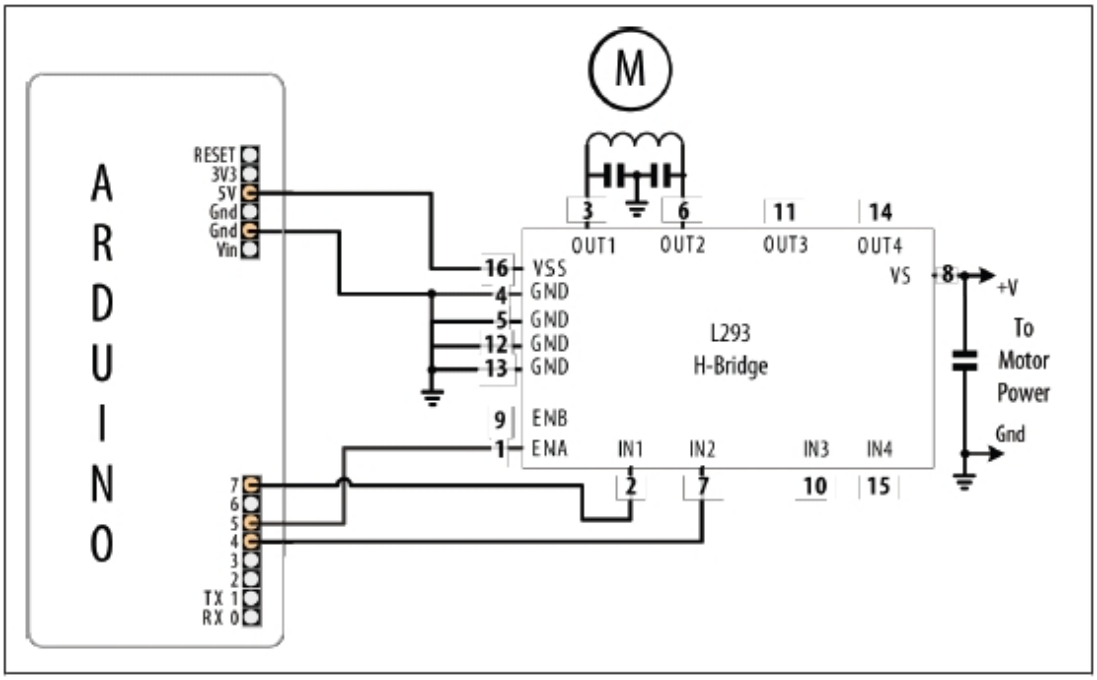
Σκοπός της Άσκησης:

Έλεγχος της ταχύτητας αλλά και της κατεύθυνσης ενός κινητήρα. Αυτό υλοποιείται με τη βοήθεια του ολοκληρωμένου τσιπ SN754410 το οποίο παρέχει περισσότερο ρεύμα (1A) αλλά έχει την ίδια συνδεσμολογία με το L293D (0,6A). Δίνοντας ψηφία από το 1 έως το 9 στο Serial Monitor ελέγχουμε την ταχύτητα, και με το σύμβολο + και - την κατεύθυνση.

Απαιτούμενα Υλικά:

1. Brushed DC motor 6 - 12 V 
2. Ολοκληρωμένο τσιπ SN754410 ή L293D 
3. 3x Κεραμικούς πυκνωτές 0.1μF 

Σχηματικό διάγραμμα:



Σχήμα 29: Συνδεσμολογία οδήγησης κινητήρα συνεχούς ρεύματος με χρήση του ολοκληρωμένου SN754410.

Πρόγραμμα 3.3

```
 /*
 * Brushed H_Bridge sketch
 * commands from serial port control motor speed and direction
 * digits '0' through '9' are valid where '0' is off, '9' is max speed
 * + or - set the direction
 */

const int enPin  = 5;  // H-Bridge enable pin
const int in1Pin = 7;  // H-Bridge input pins
const int in2Pin = 4;

void setup()
{
  Serial.begin(9600);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  Serial.println("Speed (0-9) or + - to set direction");
}

void loop()
{
  if ( Serial.available()) {
    char ch = Serial.read();
```

```
    if(isDigit(ch))              // is ch a number?
    {
      int speed = map(ch, '0', '9', 0, 255);
      analogWrite(enPin, speed);
      Serial.println(speed);
    }
    else if (ch == '+')
    {
      Serial.println("CW");
      digitalWrite(in1Pin,LOW);
      digitalWrite(in2Pin,HIGH);
    }
    else if (ch == '-')
    {
      Serial.println("CCW");
      digitalWrite(in1Pin,HIGH);
      digitalWrite(in2Pin,LOW);
    }
    else
    {
      Serial.print("Unexpected character ");
      Serial.println(ch);
    }
  }
}
```

Περιγραφή της Άσκησης

Ο κλασικός τρόπος μεταβολής της ταχύτητας ενός dc κινητήρα, είναι μέσω μεταβολής της τάσης στα άκρα του. Ο έλεγχος της τάσης στα άκρα ενός dc φορτίου επιτυγχάνεται με την αποστολή ενός παλμού που έχει υποστεί PWM (pulse width modulation), απευθείας από το arduino. Η αλλαγή στην περιστροφή του dc κινητήρα επιτυγχάνεται αντιστρέφοντας την τάση στα άκρα του.

Το ολοκληρωμένο SN754410 χρησιμοποιείται σε εφαρμογές για την κίνηση DC Motor, βηματικών κινητήρων κλπ. Αποτελείται από δύο γέφυρες 'Η' και η κάθε μία αποτελείται από δύο εισόδους (για τη σύνδεση ψηφιακών εξόδων του μικροελεγκτή) και δύο εξόδους (για τη σύνδεση κινητήρα). Η γέφυρα 'Η' παίζει κατά κάποιο τρόπο το ρόλο του διαμεσολαβητή, ώστε ο μικροελεγκτής να έχει τη δυνατότητα να κατευθύνει την κίνηση του κινητήρα και να αποφασίζει τη φορά και το εύρος κίνησής του, καθώς και την ταχύτητα. Παρακάτω φαίνονται οι καταστάσεις των **δύο εξόδων του μικροελεγκτή** προς το chip ούτως ώστε να πετύχουμε την κίνηση του κινητήρα:

InP1	InP2	Κατάσταση κινητήρα
0	0	Παύση λειτουργίας
0	1	Δεξιόστροφη κίνηση
1	0	Αριστερόστροφη κίνηση
1	1	Παύση λειτουργίας

0=LOW

1=HIGH

ΑΣΚΗΣΗ 3.4

Σκοπός της Άσκησης:

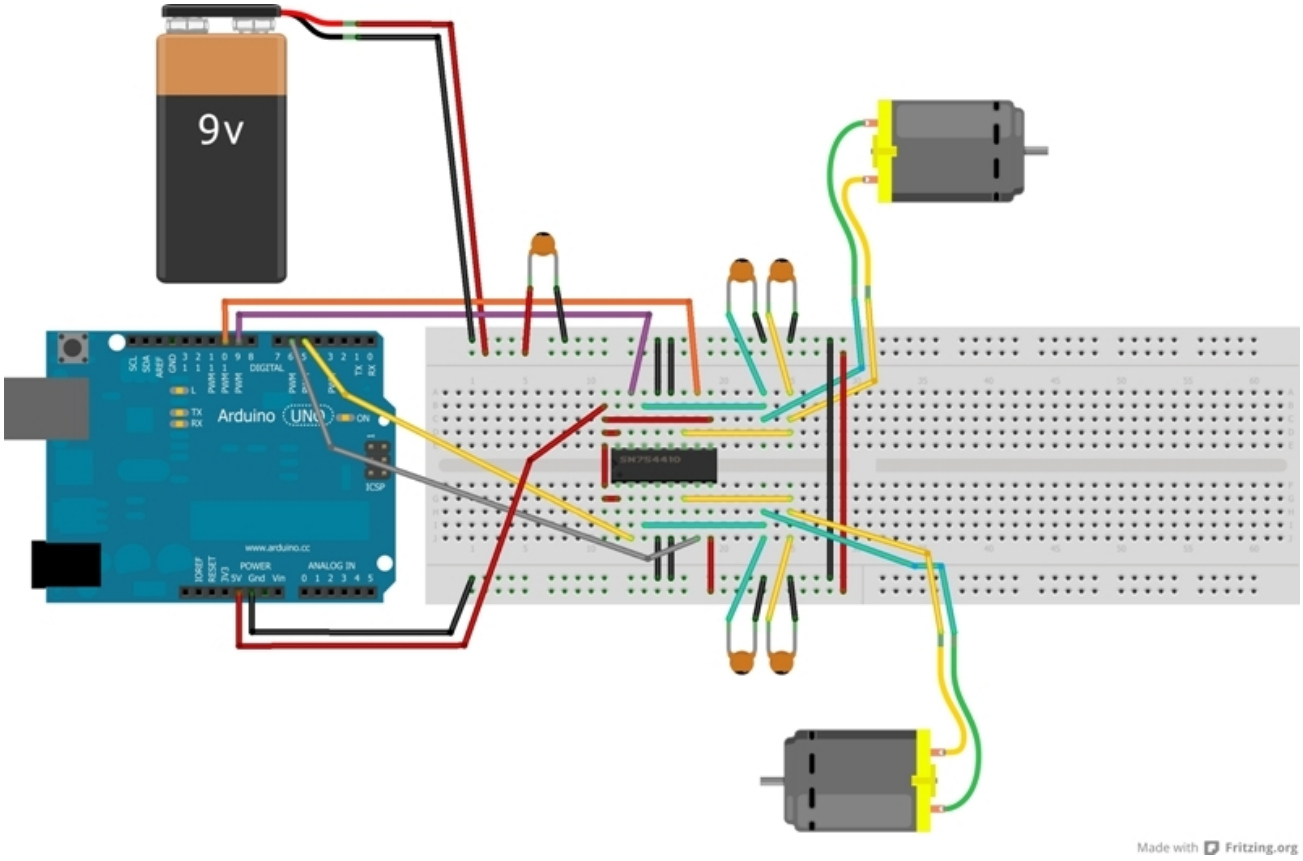
Έλεγχος της κατεύθυνσης δύο κινητήρων με τη βοήθεια του ολοκληρωμένου τσιπ SN754410.

Απαιτούμενα Υλικά:

1. 2x Brushed DC motor 6 - 12 V 
2. Ολοκληρωμένο τσιπ οδήγησης κινητήρα SN754410 ή L293D 

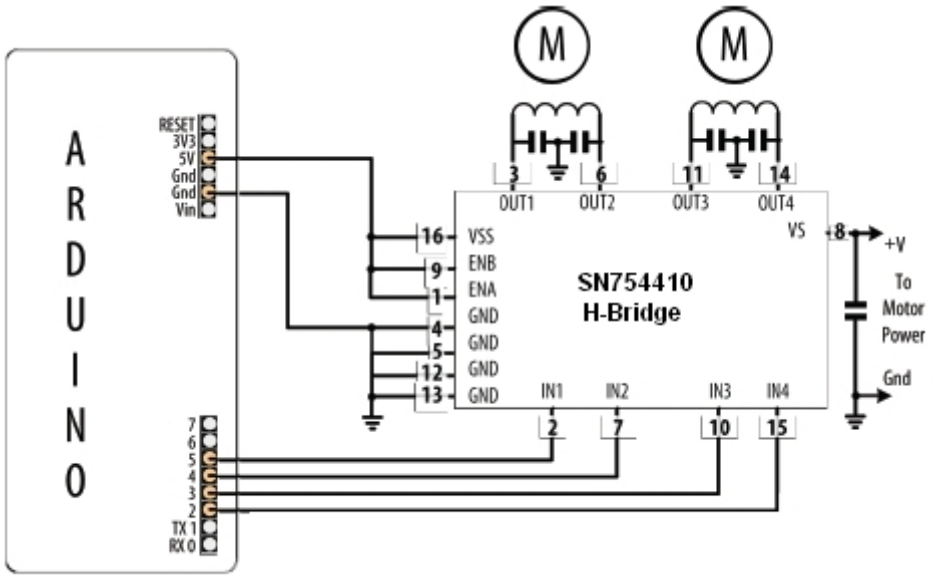
3. 5x Κεραμικούς πυκνωτές 0.1μF 

Συνδεσμολογία:



Σχήμα 30: Φυσική διάταξη του κυκλώματος οδήγησης 2 κινητήρων συνεχούς ρεύματος με χρήση του ολοκληρωμένου SN754410.

Σχηματικό διάγραμμα:



Σχήμα 31: Συνδεσμολογία οδήγησης 2 κινητήρων συνεχούς ρεύματος με χρήση του ολοκληρωμένου SN754410.

Πρόγραμμα 3.4

```
/*
 * Brushed H_Bridge sketch 2
 * commands from serial port control motor direction
 * + or - set the direction, any other key stops the motors
 */

const int in1Pin = 5;  // H-Bridge input pins
const int in2Pin = 4;

const int in3Pin = 3;  // H-Bridge pins for second motor
const int in4Pin = 2;

void setup()
{
  Serial.begin(9600);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
  Serial.println("+ - sets direction of motors, any other key stops motors");
}

void loop()
{
  if ( Serial.available()) {
    char ch = Serial.read();
    if (ch == '+')
    {
      Serial.println("CW");
      // first motor
      digitalWrite(in1Pin,LOW);
      digitalWrite(in2Pin,HIGH);
      //second motor
      digitalWrite(in3Pin,LOW);
      digitalWrite(in4Pin,HIGH);
    }
    else if (ch == '-')
```

```
    {
      Serial.println("CCW");
      digitalWrite(in1Pin,HIGH);
      digitalWrite(in2Pin,LOW);

      digitalWrite(in3Pin,HIGH);
      digitalWrite(in4Pin,LOW);
    }
    else
    {
      Serial.print("Stop motors");
      digitalWrite(in1Pin,LOW);
      digitalWrite(in2Pin,LOW);
      digitalWrite(in3Pin,LOW);
      digitalWrite(in4Pin,LOW);
    }
  }
}
```

Ο παρακάτω πίνακας δείχνει πως οι τιμές στις εισόδους του ολοκληρωμένου επηρεάζουν την κατεύθυνση του κινητήρα. Η είσοδος EN είναι μονίμως HIGH αφού είναι συνδεδεμένη στα +5V. Η κατεύθυνση ελέγχεται από τις εισόδους IN1 και IN2 για το ένα μοτεράκι παρομοίως και IN3, IN4 για το άλλο. CW (ClockWise) κίνηση σύμφωνα με τη φορά ρολογιού επιτυγχάνεται όταν έχουμε IN1=LOW και IN2=HIGH, ανάλογα και για την CCW (CounterClockWise) κίνηση, σύμφωνα με το παρακάτω πίνακα.

Πίνακας 5: Λογικός πίνακας ολοκληρωμένου τσιπ SN754410

EN	IN1	IN2	Function
HIGH	LOW	HIGH	Turn clockwise
HIGH	HIGH	LOW	Turn counterclockwise
HIGH	LOW	LOW	Motor stop
HIGH	HIGH	HIGH	Motor stop
LOW	Ignored	Ignored	Motor stop

Stepper motor (Βηματικός κινητήρας)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός βηματικού κινητήρα.

Θεωρητική εισαγωγή

Ο βηματικός κινητήρας είναι μια ηλεκτρομηχανική διάταξη της οποίας ο άξονας περιστρέφεται σε διακριτά σταθερά βήματα, που ποικίλουν σε αριθμό, παίρνοντας παλμούς τάσης συγκεκριμένης ακολουθίας στους ακροδέκτες εισόδου του. Χρησιμοποιείται σε μία πληθώρα εφαρμογών λόγω του ιδιαίτερου τρόπου περιστροφής του. Οι βηματικοί κινητήρες εξαιτίας του μοναδικού σχεδιασμού τους μπορούν να εκτελέσουν κινήσεις μεγάλης ακριβείας χωρίς κάποιον επιπλέον μηχανισμό ανάδρασης και ελέγχου. Οι βηματικοί κινητήρες χρησιμοποιούν συνδυασμό ηλεκτρικών παλμών για την κίνησή τους, μετατρέποντάς τους σε διακριτά βήματα. Η φορά περιστροφής του κινητήρα σχετίζεται με τη σειρά εφαρμογής των παλμών.

Ο κινητήρας αποτελείται από ένα ρότορα μαλακού σιδήρου με οδοντώσεις και ένα στάτορα ο οποίος αποτελείται συνήθως από τέσσερα ζεύγη ηλεκτρομαγνητών. Για να κινηθεί ο ρότορας εφαρμόζεται διαδοχικά σε κάθε ένα από τα παραπάνω ζεύγη ηλεκτρομαγνητών μια τάση ηλεκτρικού ρεύματος. Όταν το ηλεκτρικό ρεύμα μεταφέρεται από το ένα ζεύγος ηλεκτρομαγνητών στο διπλανό του, ο ρότορας μετατοπίζεται συγκεκριμένες μοίρες λόγω των μαγνητικών δυνάμεων που εφαρμόζονται σε αυτόν. Η γωνία αυτή ονομάζεται βήμα του κινητήρα και διαφέρει από μοντέλο σε μοντέλο.

Αυτού του είδους οι κινητήρες βρίσκουν εφαρμογή σε συστήματα που απαιτούν κινήσεις ακριβείας όπως εκτυπωτές, οδηγοί δισκετών, σκληροί δίσκοι, plotters, scanners, μηχανές fax και διάφορες άλλες συσκευές.

Διακρίνουμε δύο βασικούς τύπους βηματικών κινητήρων τους μονοπολικούς και τους διπολικούς. Είναι σημαντικό να γίνει διαχωρισμός τους καθώς είναι διαφορετική η λειτουργία και ο τρόπος σύνδεσης και οδήγησης του καθενός.

Πλεονεκτήματα βηματικού κινητήρα

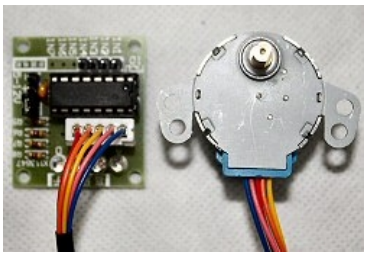
- Σε αντίθεση με τους κινητήρες συνεχούς ρεύματος (dc), δεν χρειάζεται φρένα για να μένει ακίνητος ή για να επιβραδυνθεί.
- Στις μικρές ταχύτητες περιστροφής, αλλά και κατά την εκκίνησή του, παράγει μεγάλες τιμές ροπής.
- Είναι πολύ αξιόπιστος καθώς για τη λειτουργία του δεν απαιτούνται κινούμενες ηλεκτρικές επαφές όπως στον κινητήρα συνεχούς ρεύματος και έτσι η διάρκεια ζωής του εξαρτάται μόνο από την αξιοπιστία του εδράνου κύλισης.
- Δεν απαιτείται χρήση αισθητήρων και κυκλωμάτων ανάδρασης για τον προσδιορισμό της θέσης του άξονα κίνησης.

- Ο βηματικός κινητήρας μπορεί να επιτύχει μεγάλο εύρος ταχυτήτων περιστροφής και πολύ χαμηλές ταχύτητες περιστροφής.

Μειονεκτήματα βηματικού κινητήρα

- Θορυβώδης λειτουργία.
- Αδυναμία περιστροφής σε υψηλές ταχύτητες.
- Κατά τη μετακίνηση φορτίων μεγάλης μάζας μπορεί να μη σταματήσει ακαριαία ο κινητήρας, λόγω της αυξημένης αδράνειας.

ΑΣΚΗΣΗ 3.5



Σκοπός της Άσκησης:

Έλεγχος της ταχύτητας και της κατεύθυνσης της κίνησης του βηματικού κινητήρα με τη χρήση του ολοκληρωμένου ULN2003 Darlington Array και ενός ποτενσιόμετρου.

Απαιτούμενα Υλικά:

1. Μονοπολικός Βηματικός κινητήρας DC 5V Step Motor. 
2. Οδηγός βηματικού κινητήρα ULN2003 Driver Board. 
3. Ποτενσιόμετρο 5ΚΩ 

Χαρακτηριστικά 28YBJ-48 module:

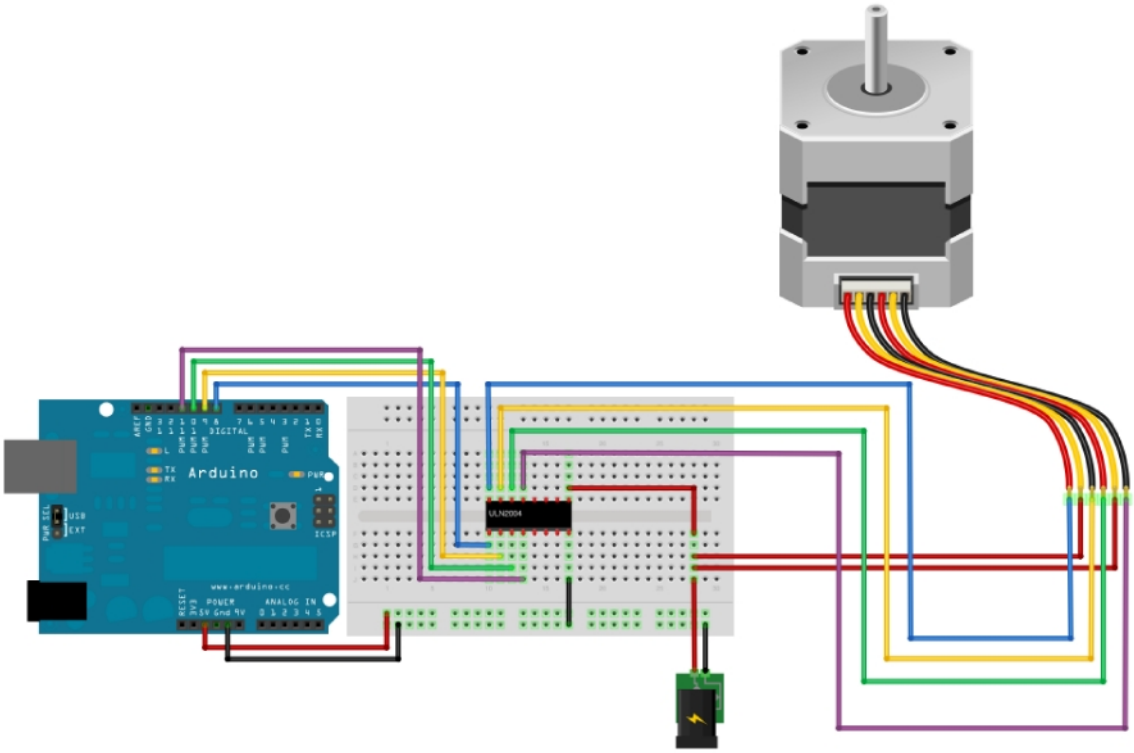
DC 5V Stepper Motor with Gear Reduction and Driver Board ULN2003
(4 Phase 5 Wire Connection)
Phase : 4
Current : 92mA
Resistance : 130 Ω
Voltage : 5V DC
Step Angle : 5.625°
No-Load Pull-Out Frequency : 800pps
No-Load Pull-In Frequency : 500pps
Pull-In Torque : ≥ 78.4mN.m
Wiring Instruction : A (Blue), B (Pink), C (Yellow), D (Orange), E (Red, Mid-Point)
Weight : 30g

Stepper Motor
Rated Voltage: DC5V 4-phase
Reduction Ratio: 1/64
Step Torque Angle: 5.625/64
DC Resistance: 200Ω±7% (25C)
Insulation Resistance: >10MΩ (500V)
Dielectric Strength: 600V AC / 1mA / 1s
Insulation Grade: A
No-load Pull in Frequency: >600Hz
No-load Pull out Frequency: >1000Hz
Pull in Torque: >34.3mN.m(120Hz)
Detent Torque: >34.3mN.m
Temperature Rise: <40K(120Hz)
Noise: <40dB (120Hz, No load, 10cm)
Cable length :23.5CM

ULN2003 Stepper Motor Driver Board

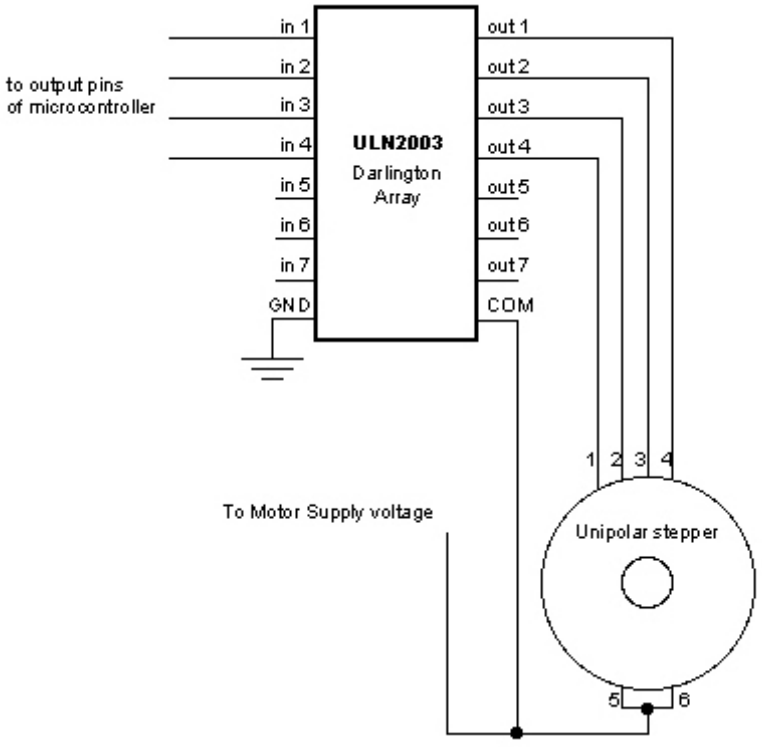
ULN2003 Darlington using high-power stepper motor driver chip.
A, B, C, D four-phase LED indicates the status of the stepper motor work.
Stepper motor with a standard interface, when used directly pluggable.
Drive Module Board Size:35mm x 31mm x 11mm

Συνδεσμολογία:



Σχήμα 32: Φυσική διάταξη του κυκλώματος οδήγησης βηματικού κινητήρα με χρήση του οδηγού ULN2003.

Σχηματικό διάγραμμα:



Σχήμα 33: Συνδεσμολογία οδήγησης βηματικού κινητήρα με χρήση του οδηγού ULN2003.

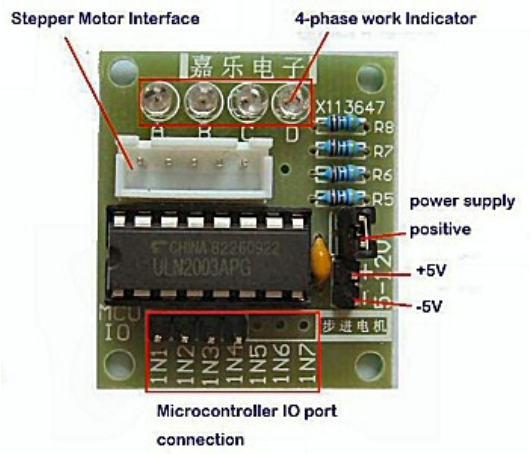
Πρόγραμμα 3.5

```
/*-----( Import needed libraries )-----*/
#include <Stepper.h>

/*-----( Declare Constants, Pin Numbers )-----*/
#define STEPS  2048   //Number of steps per revolution


/*-----( Declare objects )-----*/
// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to

// The pin connections need to be 4 pins connected
// to Motor Driver In1, In2, In3, In4  and then the pins entered
// here in the sequence 1-3-2-4 for proper sequencing
Stepper small_stepper(STEPS, 8, 10, 9, 11);

/*-----( Declare Variables )-----*/
int  Steps2Take;

void setup()
{

  // set the speed of the motor
  small_stepper.setSpeed(2);
}/*--(end setup )---*/

void loop()
{
  small_stepper.setSpeed(1);   // Show the 400 step sequence
  Steps2Take = 400;  // Rotate CW
  small_stepper.step(Steps2Take);
  delay(2000);

  small_stepper.setSpeed(6);
  Steps2Take = 800;  // Rotate CW
  small_stepper.step(Steps2Take);
  delay(2000);

  small_stepper.setSpeed(10);  //is 10 a good max speed??
  Steps2Take = -800;  // Rotate CCW
  small_stepper.step(Steps2Take);
  delay(2000);
}
```

Πρόγραμμα 3.6

```
/*
 * MotorKnob
 * A stepper motor follows the turns of a potentiometer
 * (or other sensor) on analog input 0.
 */

#include <Stepper.h>

// change this to the number of steps on your motor
#define STEPS 2048

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
```

```
// attached to
Stepper mystepper(STEPS, 8, 10, 9, 11);

// the previous reading from the analog input
int previous = 0;

void setup()
{
  // set the speed of the motor to 8? RPMs
  mystepper.setSpeed(8);
}

void loop()
{
  // get the sensor value
  int val = analogRead(0);

  // move a number of steps equal to the change in the
  // sensor reading
  mystepper.step(val - previous);

  // remember the previous value of the sensor
  previous = val;
}
```



Εικόνα 18: Οδηγός βηματικού κινητήρα ULN2003 Darlington Array.

Περιγραφή της Άσκησης

Οι **Μονοπολικοί Βηματικοί** κινητήρες (Unipolar Stepper Motor) αποτελούνται συνήθως από πέντε (ή έξι) pin εισόδου και τέσσερις ηλεκτρομαγνήτες (στην ουσία δύο ηλεκτρομαγνήτες μοιρασμένοι στην μέση από μία κεντρική σύνδεση σε κάθε ηλεκτρομαγνήτη). Οι κεντρικές συνδέσεις των ηλεκτρομαγνητών είναι ενωμένες μαζί και χρησιμοποιούνται σαν τροφοδοσία (common). Για να ελέγξουμε ένα βηματικό κινητήρα πρέπει να εφαρμόσουμε τάση σε κάθε έναν από τους ηλεκτρομαγνήτες με μια συγκεκριμένη αλληλουχία καταστάσεων η οποία είναι η εξής για τον μονοπολικό βηματικό:

STEP	COIL A	COIL B	COIL C	COIL D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Επαναλαμβάνοντας συνεχώς αυτές τις τέσσερις φάσεις (βήματα) τότε θα έχουμε τη συνεχόμενη κίνηση του κινητήρα. Όσο στέλνονται αυτές οι τέσσερις καταστάσεις ο κινητήρας θα περιστρέφεται συνεχώς προς μια κατεύθυνση. Για να πετύχουμε την αντίθετη φορά κίνησης του κινητήρα θα πρέπει να εκτελέσουμε τα τέσσερα βήματα, αρχίζοντας από το τέταρτο βήμα και πηγαίνοντας στο πρώτο (την αντίστροφη διαδικασία δηλαδή).

Τα ULN2003A, ULN2004A είναι ολοκληρωμένα που αποτελούνται από 7 ζεύγη Darlington με τη βοήθεια των οποίων πετυχαίνεται η κίνηση ενός μονοπολικού βηματικού κινητήρα. Έχει τάση εξόδου μέχρι και 50V και ρεύμα 500mA. Πρακτικά το ULN2003A χρησιμεύει για την «μετατροπή» της τάσης εξόδου του μικροελεγκτή (5V) σε ποιο μεγάλες τάσεις, όπως για παράδειγμα 8-12V που είναι η τάση λειτουργίας στους περισσότερους βηματικούς.

Στην άσκησή μας, με το ποτενσιόμετρο (ή άλλο αισθητήρα) ελέγχουμε τη λειτουργία του μονοπολικού κινητήρα με χρήση του ολοκληρωμένου ULN2003 Darlington Array. Σε περίπτωση διπολικού θα έπρεπε να γίνει χρήση διπλής γέφυρας 'H' (SN754410NE ή L293NE) όπως και στους DC κινητήρες. Οι είσοδοι In1, In2, In3, In4 από τον ελεγκτή του βηματικού κινητήρα συνδέονται στα 8, 10, 9, 11 pin του arduino, δηλαδή με 1-3-2-4 σειρά αντίστοιχα. Επιπλέον συνδέουμε το + και – του module του οδηγού στα 5V και γείωση του arduino αντίστοιχα, ή καλύτερα σε εξωτερική πηγή τροφοδοσίας 5-12V και 1A.

Οι **Διπολικοί Βηματικοί** κινητήρες (Bipolar Stepper Motor) αποτελούνται συνήθως από τέσσερα pin εισόδου. Σε σύγκριση με τους μονοπολικούς βηματικούς κινητήρες δεν έχουν κεντρική σύνδεση (common). Αντί για αυτό έχουν δύο ανεξάρτητα σετ από ηλεκτρομαγνήτες. Για να ελέγξουμε αυτού του είδους βηματικό κινητήρα πρέπει να εφαρμόσουμε τάση σε κάθε έναν από τους ηλεκτρομαγνήτες μια μία συγκεκριμένη αλληλουχία η οποία είναι η εξής:

STEP	COIL A	COIL B	COIL C	COIL D
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

Συμπεράσματα - Παρατηρήσεις

Οι τυπικοί βηματικοί έρχονται με γωνία 1,8^{o} έως 7,5^{o} ,για να προσδιορίσουμε πόσα βήματα χρειάζεται ο κινητήρας για μια πλήρη περιστροφή 360^{o} ,διαιρούμε το 360 με τις μοίρες του κάθε βήματος. Από το φύλλο δεδομένων του βηματικού έχουμε:

Speed Variation Ratio : 1/32

Step Torque Angle : 5.625° /32

Στο παράδειγμα μας η γωνία του βήματος είναι 5,625^{o} /32 που αντιστοιχεί σε 2048 βήματα για μια πλήρη περιστροφή. Η γωνία του βήματος είναι 0,17578125° ,άρα για μια πλήρη περιστροφή, δηλαδή σε ένα κύκλο 360^{o} θα κάνει x = 360 / (5.625/32) = 2048 βήματα. Πειραματικά διαπιστώσαμε πως η μέγιστη ταχύτητα είναι περίπου 8 στροφές / λεπτό, δηλαδή 273 βήματα / δευτερόλεπτο. Παρατηρούμε ακόμα πως το σύστημα μετάδοσης της ταχύτητας έχει απώλειες της τάξης των 3^{o} περίπου. Εάν ο κινητήρας μόνο δονείται και δεν γυρίζει τότε έχει πιθανόν συνδεθεί λάθος. Στα παραδείγματα που προηγήθηκαν χρησιμοποιήθηκε η ενσωματωμένη βιβλιοθήκη του Arduino stepper.h ,η οποία αυτοματοποιεί τη διαδικασία χρησιμοποιώντας έτοιμες συναρτήσεις για την κίνηση των βηματικών κινητήρων.

Τι πρέπει να προσέξουμε στους βηματικούς κινητήρες:

Η τάση λειτουργίας του πρέπει να είναι επαρκής (αυτό μπορεί να ελεγχθεί από το φύλλο δεδομένων). Για τον λόγο ότι οι βηματικοί κινητήρες είναι μηχανικές συσκευές, οι χρονισμοί παίζουν σημαντικό ρόλο. Ο κινητήρας πρέπει να ολοκληρώσει ένα βήμα πριν προχωρήσει στο επόμενο. Σε περίπτωση που ο ρυθμός των βημάτων είναι πολύ γρήγορος τότε θα έχουμε κάποιο από τα παρακάτω αποτελέσματα:

- Να μην έχουμε καθόλου κίνηση.
- Να δονείται σε ένα σημείο.
- Να είναι ασταθής η κίνηση του.
- Να κινείται προς την αντίθετη κατεύθυνση.

Servo motor (Σερβοκινητήρας)

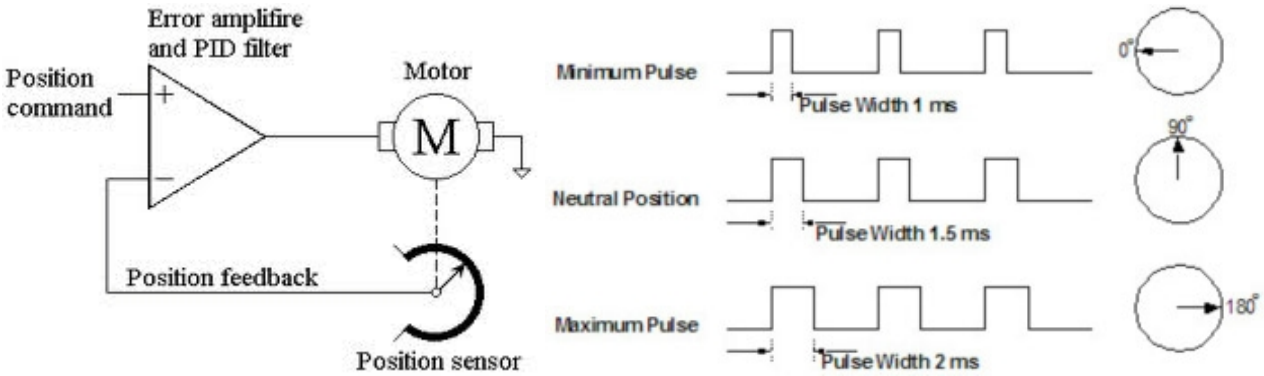
ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός σερβοκινητήρα.

Θεωρητική εισαγωγή

Ένας απλός τρόπος περιγραφής των σερβοκινητήρων θα ήταν να πούμε ότι πρόκειται για DC κινητήρες με ενσωματωμένο ελεγκτή θέσης. Κάθε σέρβο διαθέτει έναν DC κινητήρα και έναν εξειδικευμένο ελεγκτή τύπου PID (ελεγκτής ταχύτητας-θέσης), ο οποίος κατευθύνει τον κινητήρα σύμφωνα με την επιθυμία του χρήστη. Για να το πετύχει αυτό, ο ελεγκτής δέχεται μια σειρά παλμών από το μικροελεγκτή, στον οποίο είναι συνδεδεμένος ο σέρβο. Αυτό που καθορίζει τη γωνία που πρέπει να περιστραφεί ο ρότορας του κινητήρα είναι η αναλογία του «λογικού μηδέν» και του «λογικού ένα» κατά τη διάρκεια του παλμού. Επομένως ο ελεγκτής, με βάση αυτή την αναλογία υπολογίζει τη γωνία που χρειάζεται να περιστρέψει το ρότορα και με ανάδραση της θέσης, καταφέρνει να τοποθετήσει ακριβώς το ρότορα του κινητήρα στην επιθυμητή θέση.

Τα βασικότερα χαρακτηριστικά που μας ενδιαφέρουν σε έναν σέρβο είναι η ροπή του και ο χρόνος που χρειάζεται για να περιστραφεί κατά μία μοίρα. Οι σέρβο στην πλειοψηφία τους δεν κάνουν πλήρη περιστροφή αλλά περιστρέφονται συνήθως από 0 ως 180 μοίρες. Άρα, ενδείκνυνται σε εφαρμογές όπου απαιτείται ακρίβεια για την τελική θέση του κινητήρα.

Ο έλεγχος της κίνησης του servo δηλαδή η θέση του ρότορα, γίνεται με αποστολή παλμών από τον μικροελεγκτή. Η συχνότητα των παλμών που στέλνεται δεν είναι κριτικής σημασίας, αλλά το πλάτος των παλμών είναι που έχει σημασία στον έλεγχο του σέρβο. Το πλάτος των παλμών μετράται σε milliseconds. Αυτός ο παλμός έχει μεταβλητό πλάτος και τον βλέπουμε στην παρακάτω εικόνα:



Σχήμα 34: Διάγραμμα λειτουργίας σέρβο και διαγράμματα παλμοσειρών εντολοδότησης.

Για να κινηθεί ο σερβοκινητήρας προς τα αριστερά (0 μοίρες) πρέπει να του ορίσουμε παλμό πλάτους 1ms. Για να κινηθεί προς τα δεξιά (180 μοίρες) πρέπει να του ορίσουμε παλμό πλάτους 2ms, ενώ για να μείνει στάσιμος (90 μοίρες) ορίζουμε παλμό πλάτους 1.5ms. Ο σερβοκινητήρας περιμένει ένα παλμό συνήθως κάθε 20ms, ο χρόνος αυτός αποτελεί και την

περίοδο του σήματος που στέλνουμε, και αυτό γιατί οι περισσότεροι έχουν συχνότητα λειτουργίας 50Hz. Τυπική συχνότητα για κοινούς σερβοκινητήρες είναι 50 με 400 Hz.

Πλεονεκτήματα

- Χαμηλό κόστος
- Μικρές διαστάσεις και εύχρηστο σχήμα: όλα τα τμήματά ενός σέρβο περιβάλλονται από ένα συμπαγές περίβλημα από το οποίο εξέχει μόνο ο τελικός άξονας κίνησης.
- Παράγουν υψηλές τιμές ροπής
- Δεν απαιτείται χρήση αισθητήρων και κυκλωμάτων ανάδρασης για τον προσδιορισμό της θέσης του άξονα κίνησης.

Μειονεκτήματα

- Αδυναμία εκτέλεσης πλήρους και συνεχούς περιστροφής.

ΑΣΚΗΣΗ 3.6

Σκοπός της Άσκησης:

Σύνδεση σερβοκινητήρα και έλεγχος κίνησης με το ποτενσιόμετρο.

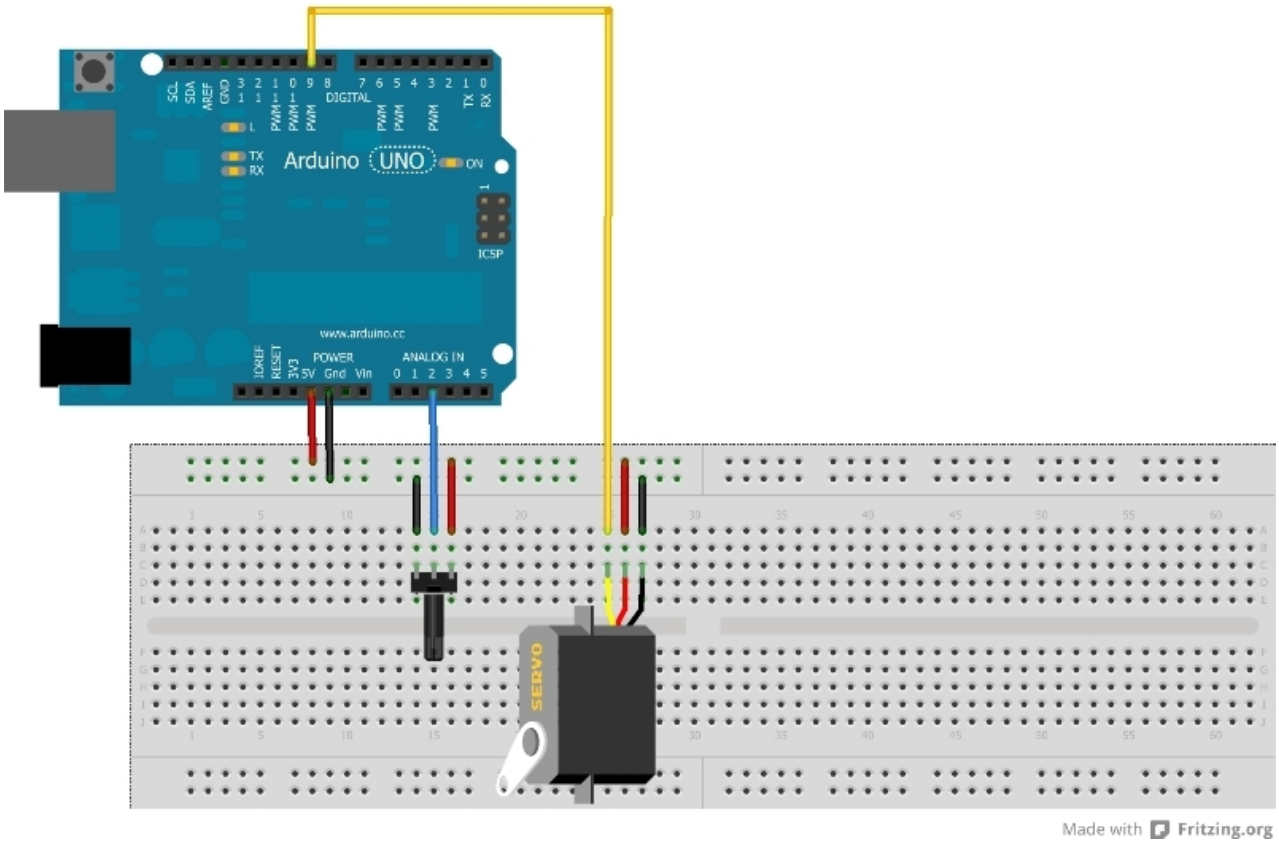
Απαιτούμενα Υλικά:

1. Σερβοκινητήρας 
2. Ποτενσιόμετρο 5ΚΩ 

Χαρακτηριστικά MG995 Tower Pro Servo:

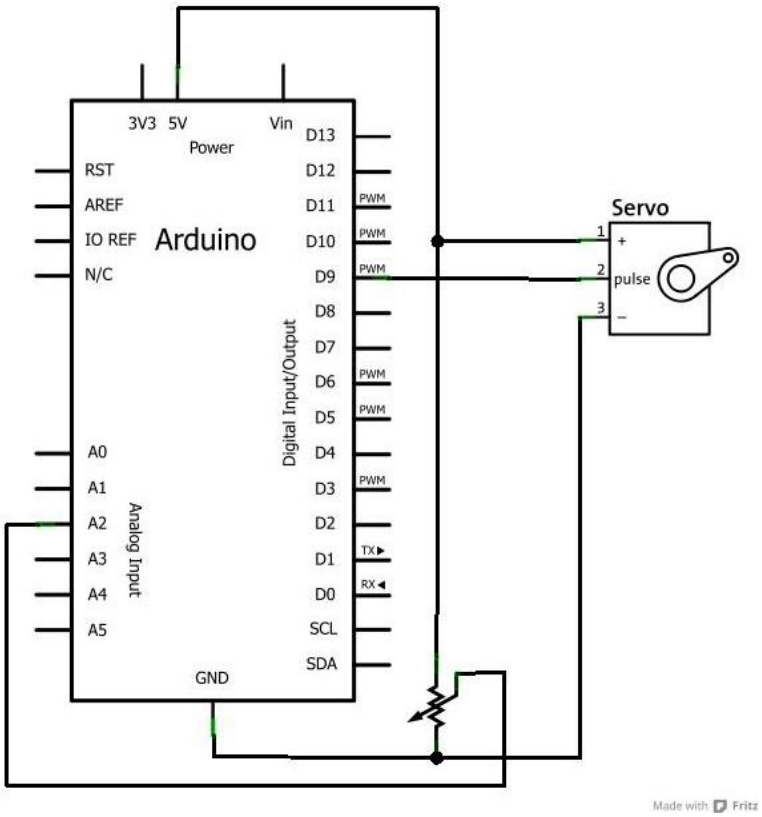
No-load Operation speed : 0.17s/60 degree (4.8V)
0.13s/60 degree (6.0V);
Input Voltage (limits): 6~20V;
Working Torque : 13kg/cm6.0V;
Rotate Speed : 53-62R/M;
Working Temperature : -30~+60'C;
Bad Zone Setting : 4 microseconds;
Connector: Universal JR / FUTABA;
Rotation angle: 180 degree (max.);
Working Voltage : 3.0V~7.2V;
Working current: 100mA;
Structure material: metal gear, coreless or ironless DC motors, dual ball bearings;

Συνδεσμολογία:



Σχήμα 35: Φυσική διάταξη του κυκλώματος σερβοκινητήρα.

Σχηματικό διάγραμμα:



Σχήμα 36: Συνδεσμολογία οδήγησης σερβοκινητήρα.

Πρόγραμμα 3.7

```
#include <Servo.h>
//Χρήση της βιβλιοθήκης servo.h

Servo servo1;
// Δημιουργία ενός αντικειμένου servo για τον έλεγχο του σέρβοκινητήρα
      int potpin = 2;   //analog pin used to connect the potentiometer
      int angle;        //variable to read the value from the analog pin

void setup()
{
      servo1.attach(9);
      //"Επισύναψη" του αντικειμένου σέρβο στη ψηφιακή θύρα 9
}

void loop()
{
      angle = analogRead(potpin);
      //Ανάγνωση αναλογικής τιμής ποτενσιόμετρου
      angle = map(angle, 0, 1023, 0, 172);
      //Μετατροπή τιμής ποτενσιόμετρου σε μοίρες (0 -180)
      servo1.write(angle);
      //Δίνει την εντολή στο σέρβο να κινηθεί προς την κατεύθυνση (μοίρες) που
του ορίζουμε
      delay(15);
      //Καθυστέρηση 15ms ώστε να φτάσει τη θέση που ορίσαμε.
}
```

Περιγραφή της Άσκησης

Οι σερβοκινητήρες είναι κινητήρες ακριβείας και η κίνηση τους εξαρτάται από τον παλμό που λαμβάνουν. Πρέπει να ανατρέξουμε στο φύλλο δεδομένων του σέρβο για το πλάτος των παλμών για διαφορετικές γωνίες. Ωστόσο η Servo.h βιβλιοθήκη παρέχει αυτόματα το απαιτούμενο PWM σήμα ανάλογα με τη τιμή της γωνίας που του στέλνουμε κάθε φορά. Κάνοντας χρήση της βιβλιοθήκης <Servo.h> ο κώδικας απλοποιείται παραμένοντας όμως αποτελεσματικός.[2]

Ωστόσο μπορούμε να προσαρμόσουμε την ελάχιστη και μέγιστη γωνία περιστροφής του άξονα του κινητήρα (ρότορα) ώστε να έχουμε το εύρος κίνησης που επιθυμούμε καλώντας την Servo.attach με προαιρετικά ορίσματα.

myservo.attach(9,1000,2000);

// σύνδεση στη θύρα 9, θέτουμε ως ελάχιστο τα 1000μs, και μέγιστο 2000μs.

Και αυτό διότι οι τυπικοί σερβοκινητήρες ανταποκρίνονται σε παλμούς και όχι σε μοίρες γωνίας.

Τα ορίσματα ενημερώνουν τη βιβλιοθήκη πόσα microseconds αντιστοιχούν όταν ζητούνται από 0 ως 180 μοίρες. Οι standard σερβοκινητήρες δεν ξεπερνάνε σχεδόν ποτέ τις 175^{o} γωνία σε κίνηση, αν και αναφέρουν στα χαρακτηριστικά τους ως μέγιστη τιμή τις 180^{o}. Τυπική γωνία μη

[2] Η στάνταρ βιβλιοθήκη <Servo.h> υποστηρίζει μέχρι 12 σερβοκινητήρες στο arduino UNO και μέχρι 48 στο Arduino Mega. Στο arduino UNO η χρήση της βιβλιοθήκης όμως απενεργοποιεί την δυνατότητα διαμόρφωσης κατά πλάτος (PWM) της analogWrite() στις συγκεκριμένες θύρες 9 και 10, ανεξαρτήτως αν υπάρχει εκεί συνδεδεμένος σέρβο ή όχι.

επαγγελματικών σέρβο στην πράξη είναι 130^{o} με 160^{o}. Αντίθετα οι continuous rotation περιστρέφονται πλήρως κατά 360^{o}. Έτσι ίσως χρειαστεί να πειραματιστούμε με το σέρβο μας ώστε να πάρουμε την επιθυμητή γωνία.

Με τη συνάρτηση writeMicroseconds() της βιβλιοθήκης servo.h μπορούμε να ελέγξουμε τη μέγιστη γωνία του σέρβο μας.

```
myservo.writeMicroseconds(1500);
```

Για τους περισσότερους σέρβο αρκεί τιμή ορίσματος από 550 έως 2500 (μs).

Η συνάρτηση myservo.attach(pin, min, max) δέχεται 3 ορίσματα:

Pin: η θύρα στην οποία συνδέεται ο σερβοκινητήρας (οποιαδήποτε ψηφιακή).

Min: (προαιρετικό) το πλάτος του παλμού σε microseconds για την ελάχιστη γωνία (0 μοίρες) του σέρβο (προεπιλεγμένο το 544).

Max: (προαιρετικό) το πλάτος του παλμού σε microseconds για τη μέγιστη γωνία (180 μοίρες) του σέρβο (προεπιλεγμένο το 2400).

Χρησιμοποιούμε κανονικού ή μικρού μεγέθους σερβοκινητήρα γιατί οι μεγάλοι σε μέγεθος χρειάζονται ξεχωριστή τροφοδοσία. Έτσι, τροφοδοτούμε το σέρβο μας στα 5V από το arduino αφού πρώτα ελέγξουμε τη τάση λειτουργίας και το ρεύμα που απαιτεί χωρίς φορτίο (σύμφωνα με τον κατασκευαστή του είναι 3 - 7,2V και 100ma).

ΑΣΚΗΣΗ 3.7

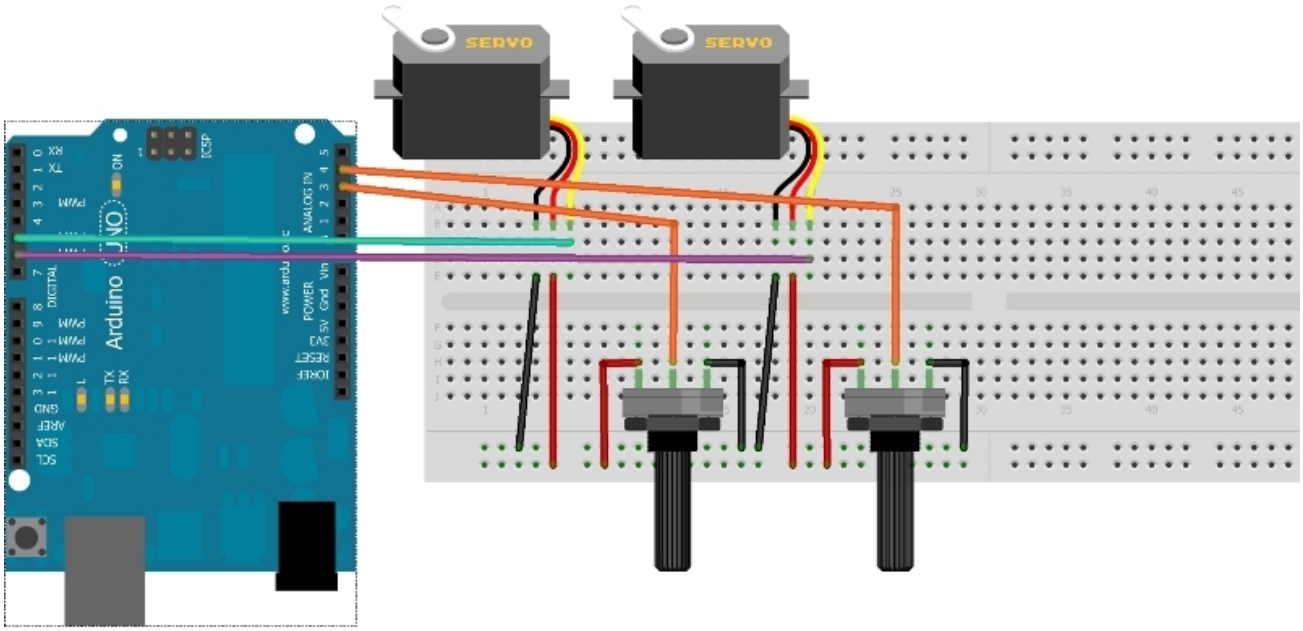
Σκοπός της Άσκησης:

Έλεγχος 2 σερβοκινητήρων με χρήση 2 ποτενσιόμετρων **ή** ενός joystick και τοποθέτηση των 2 κινητήρων κατά τέτοιο τρόπο ώστε να έχουμε περιστροφή οριζόντιου και κάθετου άξονα (pan-tilt).

Απαιτούμενα Υλικά:

1. 2x Σερβοκινητήρες 
2. Joystick ή 2x Ποτενσιόμετρα 5ΚΩ 

Συνδεσμολογία:



Σχήμα 37: Φυσική διάταξη του κυκλώματος σύνδεσης 2 σερβοκινητήρων.

Εικόνα 19: Τοποθέτηση ενός σερβοκινητήρα στο πάνω μέρος του άλλου με τρόπο τέτοιο ώστε να σχηματίζουν συνδεσμολογία pan-tilt.

Πρόγραμμα 3.8

```
#include <Servo.h>
Servo servo1; // Create a servo object
Servo servo2; // Create a second servo object

int pot1, pot2;
void setup()
{
      servo1.attach(5); // Attaches the servo on pin 5 to the servo1 object
      servo2.attach(6); // Attaches the servo on pin 6 to the servo2 object
      servo1.write(90); // Put servo1 at home position
      servo2.write(90); // Put servo2 at home position
}

void loop()
{
      pot1 = analogRead(3); // Read the X-Axis
      pot2 = analogRead(4); // Read the Y-Axis

      pot1 = map(pot1,0,1023,0,170); // scale it to use it with the servo (value
between 0 and 180)
      pot2 = map(pot2,0,1023,0,170);
```

```
      servo1.write(pot1); // sets the servo position according to the scaled
value
      servo2.write(pot2);

      delay(15); // waits for the servo to get there
}
```

Περιγραφή της Άσκησης

Ένα joystick αποτελείται από 2 ποτενσιόμετρα και ένα κουμπί. Ο άξονας των ποτενσιόμετρων είναι συνδεδεμένος στο μοχλό, ο οποίος κινείται μπρος-πίσω, δεξιά-αριστερά αλλά και κυκλικά 360^{o} και κάθε φορά επανέρχεται στη αρχική θέση ισορροπίας χάρις σε ένα σύστημα ελατηρίων.

Οι ακροδέκτες του joystick είναι 5:

Gnd, +5V, VRx, VRy, SW

Ο VRx και VRy ελέγχουν την κίνηση των ποτενσιόμετρων στον άξονα X και Y αντίστοιχα. Το SW είναι το κουμπί (switch).

Εικόνα 20: Τυπικό joystick για DIY εφαρμογές ηλεκτρονικών.

Η τροφοδοσία των κινητήρων είναι προτιμότερο να γίνει με εξωτερική τροφοδοσία όπως dc τροφοδοτικό 5-7Volt ή 4 AA μπαταρίες σε σειρά. Προσοχή πρέπει να δοθεί στο να μην ξεπεραστεί η μέγιστη γωνία που προτείνει ο κατασκευαστής, διότι κάθε φορά που γίνεται αυτό ο κινητήρας είναι σαν να λειτουργεί με το μέγιστο φορτίο που μπορεί να δεχθεί και το αντίστοιχο ρεύμα που καταναλώνει εκτινάσσεται ανάλογα. Ο σερβοκινητήρας μας καταναλώνει χωρίς φορτίο 100mA. Με το μέγιστο φορτίο ή ξεπερνώντας τη μέγιστη γωνία, το ρεύμα που απαιτεί μπορεί να ξεπεράσει το 1A.

4ο Εργαστηριακό μάθημα

LCD (Liquid Crystal Display) Οθόνη Υγρών Κρυστάλλων

LED Matrix Display (Μήτρα LED)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός :

Οθόνης Υγρών Κρυστάλλων.

Οθόνης LED Dot Matrix.

Θεωρητική εισαγωγή

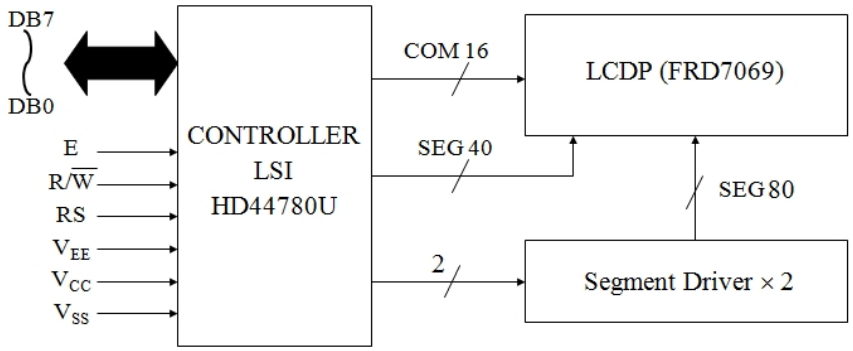
Μία δημοφιλής μέθοδος εμφάνισης κειμένου και εικόνας, είναι μέσω της οθόνης υγρών κρυστάλλων LCD (Liquid Crystal Display). Οι οθόνες LCD χρησιμοποιούνται σε ένα ευρύ φάσμα εφαρμογών όπως είναι οι οθόνες υπολογιστών, οι τηλεοράσεις, τα ρολόγια, οι οθόνες πιλοτηρίων αεροσκαφών αλλά και στη σήμανση οδών.

Η οθόνη αποτελείται από εικονοστοιχεία (pixels), κάθε ένα από τα οποία είναι γεμάτο με υγρά κρύσταλλα. Οι κρύσταλλοι αυτοί περιέχονται μεταξύ δύο διάφανων επιφανειών. Κάθε επιφάνεια περιέχει παράλληλα ηλεκτρόδια, η μία “κοινά” ηλεκτρόδια (common electrodes) και η άλλη ηλεκτρόδια “τμήματος” (segment electrodes). Τα ηλεκτρόδια κάθε επιφάνειας διασταυρώνονται κάθετα με τα ηλεκτρόδια της άλλης και σε κάθε συμβολή υπάρχει ένα ελεγχόμενο pixel. Ανάλογα με την τάση που εφαρμόζεται στα δύο ηλεκτρόδια ελέγχου ενός pixel αλλάζει η πόλωση των υγρών κρυστάλλων στο σημείο αυτό, με αποτέλεσμα να περνάει ή όχι φως και το pixel να φαίνεται σκούρο ή φωτεινό. Για την οδήγηση των pixels ενός LCD πίνακα κάθε ένα “κοινό” ηλεκτρόδιο επιλέγεται διαδοχικά, ενώ την ίδια στιγμή όλα τα ηλεκτρόδια “τμήματος” οδηγούνται με την κατάλληλη τάση. Η τάση αυτή καθορίζει αν θα είναι “αναμμένο” ή “σβηστό” κάθε ένα pixel που ελέγχεται από το “κοινό” ηλεκτρόδιο. Η διαδικασία επαναλαμβάνεται περιοδικά (χρονική πολύπλεξη) για τα επόμενα “κοινά” ηλεκτρόδια.

Μια οθόνη LCD αποτελείται από πλέγμα με pixels τα οποία είναι τοποθετημένα σε μικρότερα δίκτυα που συνθέτουν τους χαρακτήρες. Ένα τυπικό 16x2 μέγεθος θα έχει 16 πλέγματα χαρακτήρων σε δύο γραμμές. Κάθε χαρακτήρας του δικτύου αποτελείται από 5 pixels πλάτος και 8 ύψος.

Πολλές απλές οθόνες ενδείξεων LCD διαθέτουν ενσωματωμένο ελεγκτή και διασυνδέονται εύκολα σε ένα σύστημα ως ολοκληρωμένα τμήματα (modules). Ένα από τα γνωστότερα “βιομηχανικά” interfaces για διασύνδεση LCD modules βασίζεται στον ελεγκτή HD44780 (Hitachi).

Τα δεδομένα για απεικόνιση στην οθόνη μεταφέρονται από τον μικροελεγκτή με παράλληλο δίαυλο μήκους 4 bit. Η τροφοδοσία της μονάδας απεικόνισης γίνεται με 5V. Στο παρακάτω σχήμα φαίνεται το συνοπτικό διάγραμμα της μονάδας απεικόνισης.

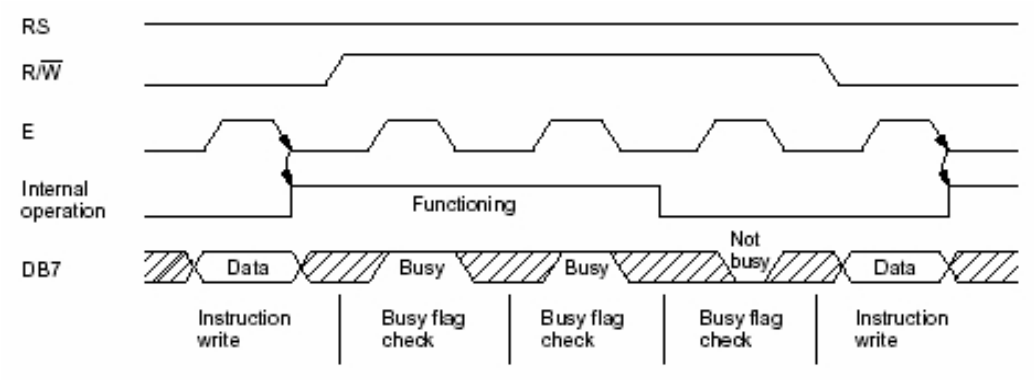


Σχήμα 38: Μονάδα απεικόνισης LCD 16x2 με ελεγκτή HD44780.

Ελεγκτές και Ακροδέκτες LCD

Οι LCD οθόνες διαθέτουν ενσωματωμένους οδηγούς και ενσωματώνονται στο εκάστοτε σύστημα μέσω των ελεγκτών LCD. Ο ελεγκτής LCD είναι ένα ολοκληρωμένο κύκλωμα, το οποίο διασυνδέεται εύκολα με μικροελεγκτές, χρησιμοποιεί μνήμη απεικόνισης όπου τοποθετεί τα προς απεικόνιση δεδομένα. Στη μνήμη αυτή δεν έχει προσπέλαση ο μικροελεγκτής και τα δεδομένα πρέπει να εισαχθούν ως εντολές προς αυτόν. Ο πιο γνωστός τύπος ελεγκτή/οδηγητή που απαιτείται για τον έλεγχό τους είναι το ολοκληρωμένο κύκλωμα HD44780 της εταιρείας Hitachi το οποίο διαθέτει 14 γραμμές διασύνδεσης:

- **Data pins (DB7-DB0):** 8 γραμμές δεδομένων. Από τις γραμμές αυτές εισάγονται εντολές και πληροφορίες απεικόνισης στο LCD module. Κατά την ανάγνωση, το DB7 χρησιμεύει και ως busy flag, για τον έλεγχο ολοκλήρωσης των εντολών προς το LCD. Το interface μπορεί να λειτουργήσει και σε 4-bit mode, ιδιαίτερα χρήσιμο για τη διασύνδεση με μικροελεγκτές με περιορισμένο αριθμό διαθέσιμων ακροδεκτών.
- **Read/Write (R/W):** Επιλογή ανάγνωσης ή εγγραφής.
- **Register select (RS):** Ο ελεγκτής HD44780 διαθέτει δύο καταχωρητές, α) τον καταχωρητή εντολής, όπου τοποθετείται η αιτούμενη εντολή, και β) τον καταχωρητή δεδομένων, όπου τοποθετούνται (λαμβάνονται) τα δεδομένα (αποτελέσματα) της εντολής. Το σήμα RS επιλέγει τον καταχωρητή κατά την προσπέλαση (με 0 επιλέγεται ο καταχωρητής εντολής, με 1 ο καταχωρητής δεδομένων).
- **Enable (E):** Το σήμα αυτό ενεργοποιεί κάθε προσπέλαση. Τα δεδομένα από τον μικροεπεξεργαστή προς τον ελεγκτή αποθηκεύονται στην κατερχόμενη ακμή του E, ενώ τα επιστρεφόμενα δεδομένα τοποθετούνται από τον ελεγκτή στα DB[7:0] λίγο μετά την ανερχόμενη ακμή του E.
- Γραμμές τροφοδοσίας: **VCC, GND, VEE** (Contrast Adjust). Στο σχήμα που ακολουθεί φαίνεται μια τυπική ακολουθία προσπελάσεων και ελέγχου του busy flag (από το φύλλο δεδομένων του HD44780):



Σχήμα 39: Τυπική ακολουθία προσπελάσεων και ελέγχου του busy flag του ελεγκτή HD44780.

ΑΣΚΗΣΗ 4.1

Σκοπός της Άσκησης:

Σύνδεση LCD με τον μικροελεγκτή και παρουσίαση των βασικών συναρτήσεων της LCD βιβλιοθήκης <LiquidCrystal.h>.

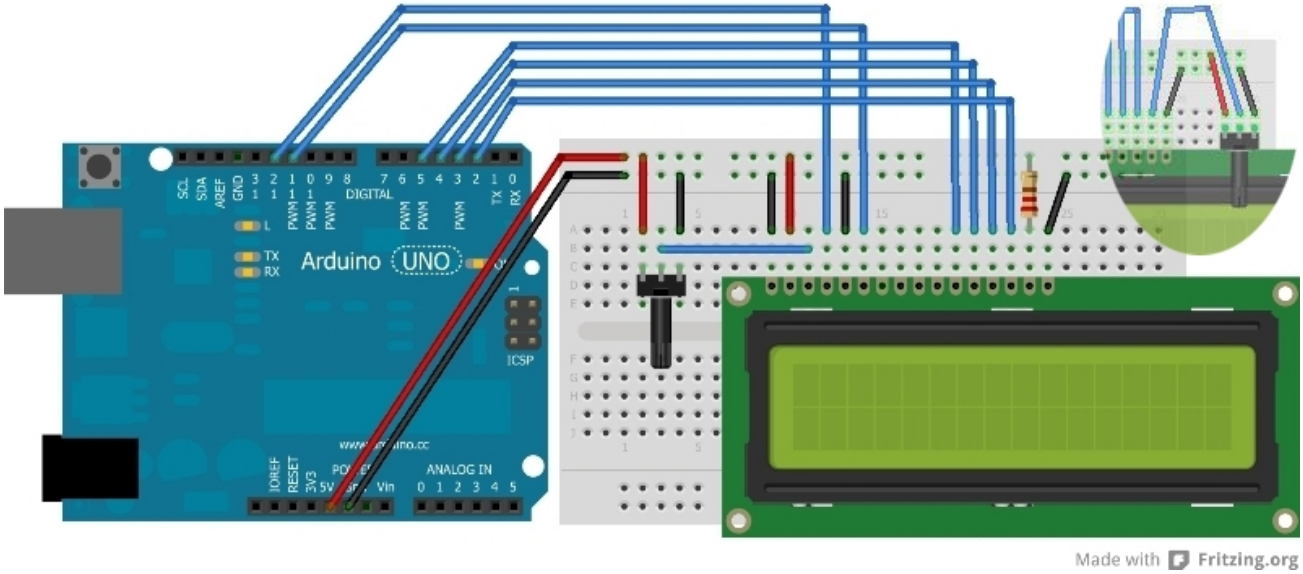
Απαιτούμενα Υλικά:

1. Οθόνη υγρών κρυστάλλων 16x2 συμβατή με τον HD44780 οδηγό. 
2. 2x Ποτενσιόμετρα (5KΩ και 1KΩ) 

ή

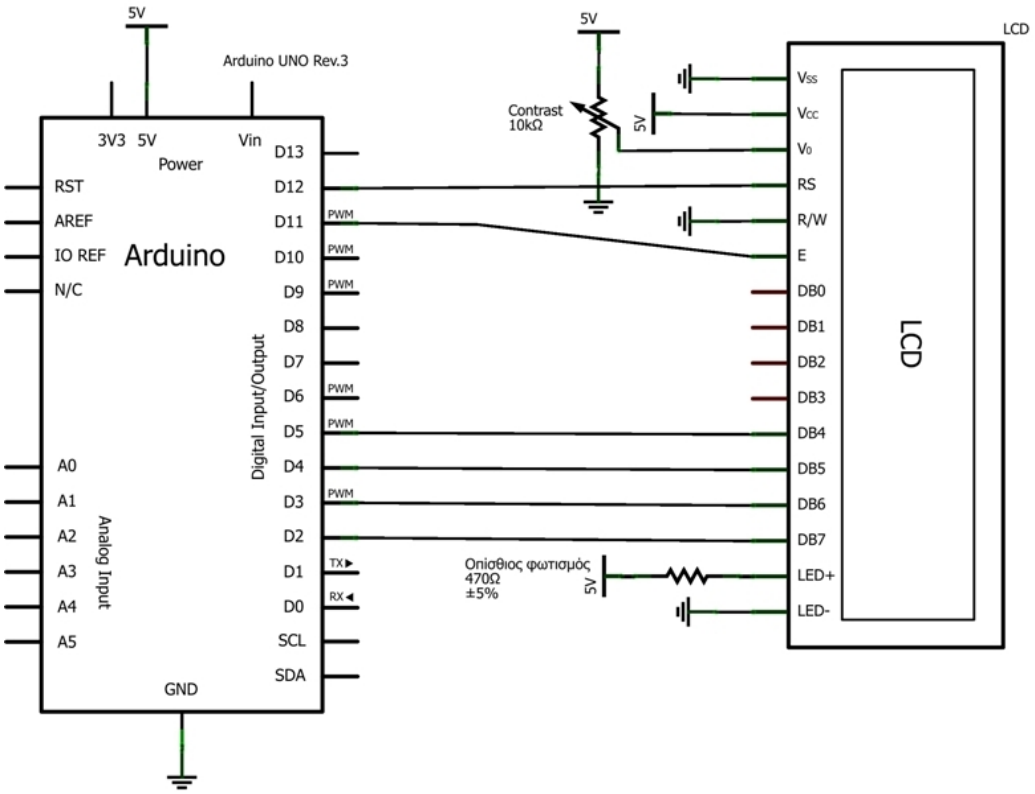
3. 1 Αντίσταση 3 KΩ  και
   1 Αντίσταση 470 Ω  αντίστοιχα.

Συνδεσμολογία:



Σχήμα 40: Φυσική διάταξη του κυκλώματος σύνδεσης Οθόνης Υγρών Κρυστάλλων.

Σχηματικό διάγραμμα:



Σχήμα 41: Συνδεσμολογία κυκλώματος Οθόνης Υγρών Κρυστάλλων.

Πρόγραμμα 4.1

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // create an lcd object and assign the
pins

void setup() {
      lcd.begin(16, 2); // Set the display to 16 columns and 2 rows
}

void loop() {
      // run the 6 routines
      mybasicPrint();
      mydisplayOnOff();
      mysetCursor();
      myscrollLeft();
      myscrollRight();
      mycursor();
}

void mybasicPrint() {
      lcd.clear(); // Clear the display
      lcd.print("Basic Print"); // print some text
      delay(2000);
}

void mydisplayOnOff() {
      lcd.clear(); // Clear the display
      lcd.print("Display On/Off"); // print some text
```

```
      for(int x=0; x < 3; x++) { // loop 3 times
            lcd.noDisplay(); // turn display off
            delay(1000);
            lcd.display(); // turn it back on again
            delay(1000);
    }
}

void mysetCursor() {
      lcd.clear(); // Clear the display
      lcd.print("SetCursor"); // print some text
      delay(1000);
      lcd.clear(); // Clear the display
      lcd.setCursor(5,0); // cursor at column 5 row 0
      lcd.print("5,0");
      delay(2000);
      lcd.setCursor(10,1); // cursor at column 10 row 1
      lcd.print("10,1");
      delay(2000);
      lcd.setCursor(3,1); // cursor at column 3 row 1
      lcd.print("3,1");
      delay(2000);
}

void myscrollLeft() {
      lcd.clear(); // Clear the display
      lcd.print("Scroll Left");
      delay(1000);
      lcd.clear(); // Clear the display
      lcd.setCursor(7,0);
      lcd.print("Beginning");
      lcd.setCursor(9,1);
      lcd.print("Arduino...");
      delay(1000);
      for(int x=0; x<16; x++) {
            lcd.scrollDisplayLeft(); // scroll display left 16 times
            delay(250);
      }
}

void myscrollRight() {
      lcd.clear(); // Clear the display
      lcd.print("Scroll Right");
      delay(1000);
      lcd.clear(); // Clear the display
      lcd.print("I love");
      lcd.setCursor(0,1);
      lcd.print("Arduino!");
      delay(1000);
      for(int x=0; x<16; x++) {
      lcd.scrollDisplayRight(); // scroll display right 16 times
            delay(250);
      }
}

void mycursor() {
      lcd.clear(); // Clear the display
      lcd.cursor(); // Enable cursor visible
      lcd.print("Cursor On");
      delay(3000);
      lcd.clear(); // Clear the display
      lcd.noCursor(); // cursor invisible
      lcd.print("Cursor Off");
      delay(3000);
```

```
      lcd.clear(); // Clear the display
      lcd.cursor(); // cursor visible
      lcd.blink(); // cursor blinking
      lcd.print("Cursor Blink On");
      delay(3000);
      lcd.noCursor(); // cursor invisible
      lcd.noBlink(); // blink off
}
```

Παρατηρήσεις:

Το κύκλωμα διασύνδεσης

Για τη σύνδεση της LCD οθόνης απαιτούνται καλώδια χωρίς άλλα εξαρτήματα. Εξαίρεση αποτελεί ο έλεγχος του κοντράστ όπου χρησιμοποιείται αντίσταση (χωρίς αντίσταση θα είναι απλά δύσκολη η ανάγνωση των πληροφοριών). Χρησιμοποιούμε ποτενσιόμετρο ώστε να υπάρχει δυνατότητα ρύθμισης ανάλογα με τις συνθήκες φωτισμού. Οι ακροδέκτες της LCD οθόνης είναι αριθμημένοι από το 1 έως το 14 με ταυτόχρονη αναγραφή της ετικέτας τους επάνω στην οθόνη. Στην περίπτωση αγοράς LCD με 16 επαφές, η χρήση των ακροδεκτών 1 έως 14 είναι η ίδια, οι 2 τελευταίοι ακροδέκτες A(5V), K(Gnd) προορίζονται για τον background φωτισμό.

Πίνακας 6: Συνδεσμολογία Οθόνης Υγρών Κρυστάλλων

Ακροδέκτης Arduino	**Ακροδέκτης LCD**
12	RS (4)
11	E (6)
GND	RW(5)
5 (Digital)	D4 (11)
4 (Digital)	D5 (12)
3 (Digital)	D6 (13)
2 (Digital)	D7 (14)
GND	Vss (1)
5V	Vcc/Vdd (2)
GND(μέσω 3KΩ ή pot)	Vo (3)
5V (μέσω 470Ω ή pot)	A (15) (background light)
GND	K (16) (background light)

Το ποτενσιόμετρο που ελέγχει τη φωτεινότητα της οθόνης μπορεί να έχει τιμή από 5 έως 10 kΩ. Μπορούμε να προσθέσουμε ένα επιπλέον ποτενσιόμετρο 1 ΚΩ για τον background φωτισμό, αντί της αντίστασης που προηγείται του ακροδέκτη 15 της οθόνης LCD.

Η 16x2 LCD θα «κόψει» οτιδήποτε πέραν του 16^{ου} χαρακτήρα. Ενώ η 20x4 LCD θα «τυλίξει» την πρώτη γραμμή στην 3^{η}, ομοίως την 2^{η} στην 4^{η}. Αυτός είναι ο τρόπος που η μνήμη της LCD διαχειρίζεται το κείμενο. Έτσι πρέπει να είμαστε προσεκτικοί και να μη ξεπερνάμε το όριο των 16 χαρακτήρων σε κάθε γραμμή ή το ανάλογο μήκος για την κάθε lcd οθόνη, μέχρι μία LCD βιβλιοθήκη στο μέλλον να μπορεί να κάνει αυτόματα αυτή τη δουλειά.

LED Matrix Display (Οδήγηση οθόνης LED)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός οθόνης LED.

Θεωρητική εισαγωγή

Το led matrix display είναι μια συστοιχία led ενωμένα σε γραμμές και στήλες, σχηματίζοντας συνήθως ένα τετράγωνο ή ορθογώνιο. Σύνηθες μέγεθος είναι το 8x8, δηλαδή συνολικά 64 leds. Υπάρχουν δίχρωμα ή και τρίχρωμα RGB led matrix.

Οι φωτοδίοδοι σε μία μήτρα led είναι καλωδιωμένοι μεταξύ τους, έτσι ώστε η άνοδος ή η κάθοδος από κάθε led να είναι ενωμένοι σε μια γραμμή. Για παράδειγμα σε ένα led matrix **ανόδου**, όλοι οι άνοδοι μιας γραμμής είναι συνδεδεμένοι μεταξύ τους ομοίως και όλες οι κάθοδοι μιας στήλης. Όπως επίσης ισχύει και το αντίστροφο, σε ένα led matrix **καθόδου** όλες οι κάθοδοι μιας γραμμής είναι ενωμένες.

64 φωτοδίοδοι θα απαιτούσαν συνολικά 65 θύρες (μία για κάθε led και μία για τη τάση ή τη γείωση αντίστοιχα, αναλόγως αν είναι ανόδου ή καθόδου) σε άλλη περίπτωση, όμως σε ένα κοινό 8x8 led matrix ενώνοντας γραμμές και στήλες μεταξύ τους, χρειάζονται μόνο 16 θύρες, ελαχιστοποιώντας έτσι το πλήθος των απαιτούμενων θυρών. Έστω ότι έχουμε μια μήτρα led κοινής ανόδου και θέλουμε να φωτοβολήσουμε το led στη θέση X,Y (X στήλη, Y γραμμή), τότε τροφοδοτούμε τη X γραμμή και γειώνουμε την Y στήλη.

Σε εφαρμογές που πρέπει να χρησιμοποιηθούν πολλά LED, για ελαχιστοποίηση των θυρών που απαιτούνται χρησιμοποιείται η τεχνική της πολύπλεξης. Η **πολύπλεξη** λειτουργεί επιλέγοντας led ενωμένα σε σειρά, χωρισμένα σε ομάδες (γραμμές ή στήλες). Σαρώνοντας τις γραμμές και τις στήλες αρκετά γρήγορα τουλάχιστον 25 φορές το δευτερόλεπτο δίνεται η εντύπωση στο ανθρώπινο μάτι της συνεχόμενης κίνησης. Ενώ στην πραγματικότητα τροφοδοτείται η μία γραμμή μετά την άλλη σε πολύ γρήγορο ρυθμό. Σαν αποτέλεσμα έχει να φαίνεται, ότι τα led παραμένουν σταθερά φωτεινά και όχι να αναβοσβήνουν ή να τρεμοπαίζουν.

Σύμφωνα με την τεχνική της χρονικής πολύπλεξης:

α) κάθε λογική έξοδος οδηγεί τους ατομικούς ακροδέκτες των αντίστοιχων LEDs όλων των τμημάτων,

β) οι κοινοί ακροδέκτες των τμημάτων οδηγούνται διαδοχικά με ρυθμό μερικών kHz έτσι ώστε

γ) ο κοινός ακροδέκτης ενός μόνο τμήματος να έχει την κατάλληλη τάση για ένα συγκεκριμένο μερίδιο χρόνου, και σε συνδυασμό με τις τιμές των λογικών εξόδων εκείνη τη στιγμή να ανάβουν τα επιθυμητά LEDs στο τμήμα αυτό ενώ τα άλλα τμήματα παραμένουν σβηστά.

δ) Στη συνέχεια, απενεργοποιείται το οδηγούμενο τμήμα, οι τιμές απεικόνισης για το επόμενο τμήμα οδηγούνται στις λογικές εξόδους και ενεργοποιείται ο κοινός ακροδέκτης του επόμενου τμήματος. Η διαδικασία επαναλαμβάνεται κυκλικά, έτσι ώστε το ανθρώπινο μάτι να αντιλαμβάνεται όλα τα τμήματα ενεργοποιημένα ταυτόχρονα.

ΑΣΚΗΣΗ 4.2

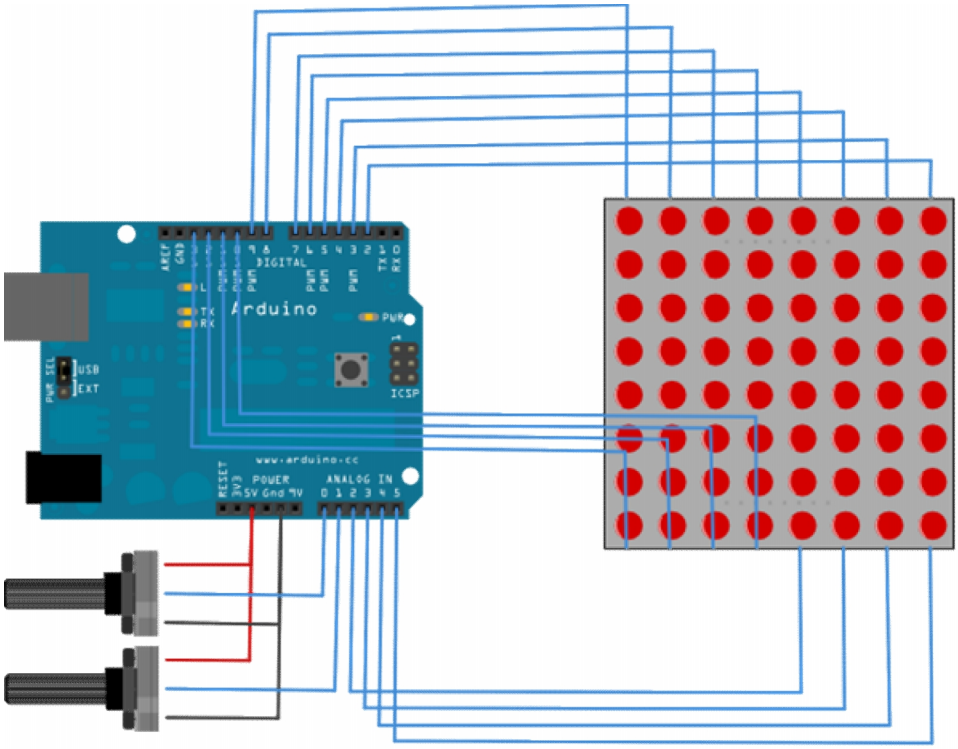
Σκοπός της Άσκησης:

Σύνδεση 8x8 LED Matrix και σάρωση γραμμών-στηλών στον άξονα X-Y με χρήση 2 ποτενσιόμετρων.

Απαιτούμενα Υλικά:

1. Led Matrix Display 8x8 κοινής ανόδου (καθοδικών στηλών). 
2. 8x Αντιστάσεις 470 Ω 
3. 2x Ποτενσιόμετρα 

Συνδεσμολογία:



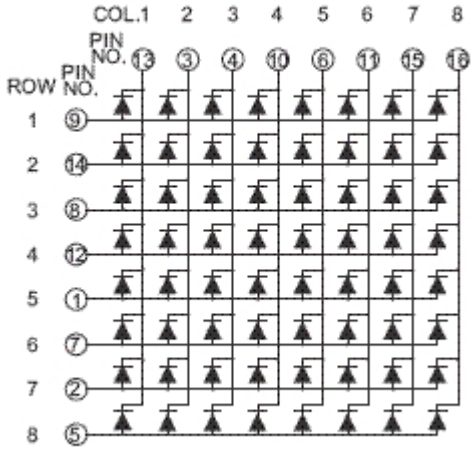
Σχήμα 42: Φυσική διάταξη του κυκλώματος σύνδεσης led matrix και 2 ποτενσιόμετρων με απεικόνιση Fritzing.

Σχηματικό διάγραμμα:

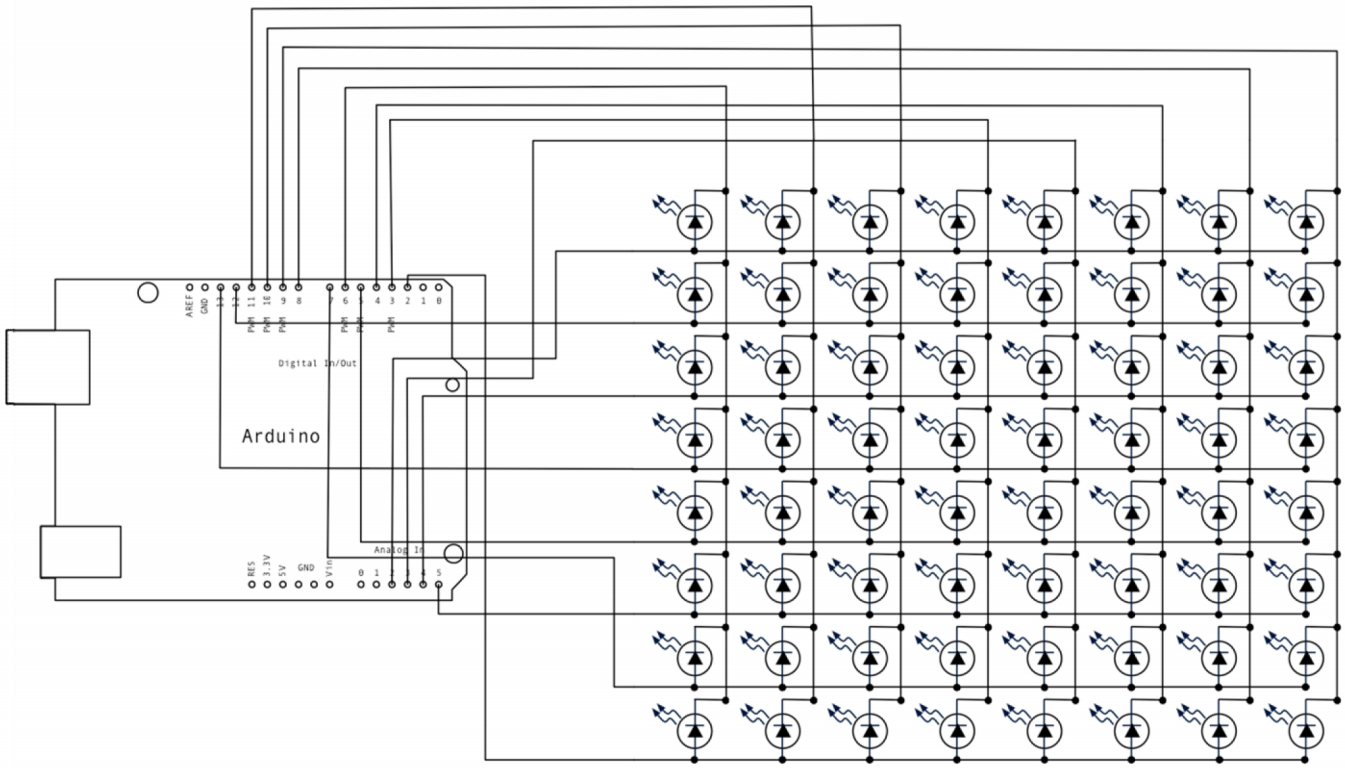
Πίνακας 7: Συνδεσμολογία θυρών μήτρας led 8x8

Θύρες matrix	Γραμμές	Στήλες	Θύρες Arduino
1	5	-	13
2	7	-	12
3	-	2	11

4	-	3	10
5	8	-	16 (analog pin 2)
6	-	5	17 (analog pin 3)
7	6	-	18 (analog pin 4)
8	3	-	19 (analog pin 5)
9	1	-	2
10	-	4	3
11	-	6	4
12	4	-	5
13	-	1	6
14	2	-	7
15	-	7	8
16	-	8	9



Σχήμα 43: Διάγραμμα συνδεσμολογίας μήτρας led 8x8 (κοινής ανόδου).



Σχήμα 44: Συνδεσμολογία μήτρας led 8x8 (καθοδικών στηλών και ανοδικών γραμμών) με το arduino.

Πρόγραμμα 4.2

```
/*
 Row-Column Scanning an 8x8 LED matrix with X-Y input
 This example controls an 8x8 LED matrix using two analog inputs
 For other LED cathode column matrixes, you should only need to change
 the pin numbers in the row[] and column[] arrays

 rows are the anodes
 cols are the cathodes
```

```
 ---------
 Pin numbers:
 Matrix:
 * Digital pins 2 through 13,
 * analog pins 2 through 5 used as digital 16 through 19
 Potentiometers:
 * center pins are attached to analog pins 0 and 1, respectively
 * side pins attached to +5V and ground, respectively.
*/

// 2-dimensional array of row pin numbers:
const int row[8] = { 2,7,19,5,13,18,12,16 };

// 2-dimensional array of column pin numbers:
const int col[8] = { 6,11,10,3,17,4,8,9 };

// 2-dimensional array of pixels:
int pixels[8][8];

// cursor position:
int x = 5;
int y = 5;

void setup() {
  // initialize the I/O pins as outputs
  // iterate over the pins:
  for (int thisPin = 0; thisPin < 8; thisPin++) {
    // initialize the output pins:
    pinMode(col[thisPin], OUTPUT);
    pinMode(row[thisPin], OUTPUT);
    // take the col pins (i.e. the cathodes) high to ensure that
    // the LEDS are off:
    digitalWrite(col[thisPin], HIGH);
  }
  // initialize the pixel matrix:
  for (int x = 0; x < 8; x++) {
    for (int y = 0; y < 8; y++) {
      pixels[x][y] = HIGH;
    }
  }
}

void loop() {
  // read input:
  readSensors();
  // draw the screen:
  refreshScreen();
}

void readSensors() {
  // turn off the last position:
  pixels[x][y] = HIGH;
  // read the sensors for X and Y values:
  x = 7 - map(analogRead(A0), 0, 1023, 0, 7);
  y = map(analogRead(A1), 0, 1023, 0, 7);
  // set the new pixel position low so that the LED will turn on
  // in the next screen refresh:
  pixels[x][y] = LOW;
}

void refreshScreen() {
  // iterate over the rows (anodes):
  for (int thisRow = 0; thisRow < 8; thisRow++) {
    // take the row pin (anode) high:
```

```
    digitalWrite(row[thisRow], HIGH);
    // iterate over the cols (cathodes):
    for (int thisCol = 0; thisCol < 8; thisCol++) {
      // get the state of the current pixel;
      int thisPixel = pixels[thisRow][thisCol];
      // when the row is HIGH and the col is LOW,
      // the LED where they meet turns on:
      digitalWrite(col[thisCol], thisPixel);
      // turn the pixel off:
      if (thisPixel == LOW) {
        digitalWrite(col[thisCol], HIGH);
      }
    }
    // take the row pin low to turn off the whole row:
    digitalWrite(row[thisRow], LOW);
  }
}
```

Παρατηρήσεις:

Οι οθόνες Led Matrix δεν έχουν στάνταρ εξόδους θυρών, έτσι ελέγχουμε προσεκτικά τα διαγράμματα και το φύλλο δεδομένων για τη σωστή συνδεσμολογία. Η χρήση μήτρας LED χωρίς τη χρήση κάποιου αποκωδικοποιητή είναι μια επίπονη διαδικασία. Όταν ο αριθμός των οδηγούμενων τμημάτων LEDs είναι μεγάλος, απαιτείται αντίστοιχα μεγάλος αριθμός λογικών εξόδων οδήγησης. Επειδή ο αριθμός των εξόδων σε ολοκληρωμένα κυκλώματα ελέγχου είναι περιορισμένος, για την οδήγηση πολλών τμημάτων χρησιμοποιείται η τεχνική της **πολύπλεξης**.

Πολυπλεξία είναι η τεχνική της επιλογής μίας σειράς της οθόνης led τη φορά. Αφού τροφοδοτήσουμε με ρεύμα μία - μία όλες τις γραμμές διαδοχικά, δηλαδή ανάβουμε και σβήνουμε τη μία γραμμή μετά την άλλη, επαναλαμβάνουμε τη διαδικασία αυτή για κάθε γραμμή και αν αυτό γίνει πολύ γρήγορα (περισσότερο από 100 Hz ή 100 φορές το δευτερόλεπτο) μας δίνεται η εντύπωση ότι φωτίζεται σταθερά ολόκληρη η οθόνη των Led, (λόγω του φαινομένου της ανθρώπινης όρασης, στο οποίο μια εικόνα εντυπώνεται στο μάτι για περίπου 1/25 του δευτερολέπτου). Το μειονέκτημα της συγκεκριμένης διαδικασίας είναι ότι δε μπορούμε να ανάψουμε ξεχωριστά ένα led, χωρίς να ανάψουν και τα υπόλοιπα στη γραμμή ή στήλη που τα εμπεριέχει.

Κάθε led matrix 8x8 έχει 64 LED, δεν έχει όμως 128 ακροδέκτες. Αντ' αυτού τα led είναι καλωδιωμένα μεταξύ τους σε μορφή πίνακα. Για να ανάψουμε ένα led συνδέουμε τη γραμμή ανόδου που το περιλαμβάνει στα 5V -μέσω κατάλληλης αντίστασης- και τη στήλη καθόδου στη γείωση, αυτό μπορεί να επιτευχθεί και χωρίς τη χρήση μικροελεγκτή. Όταν η κατάσταση μιας στήλης γίνεται LOW και η κατάσταση μιας σειράς HIGH τότε το διασταυρούμενο led ανάβει. Για όλα τα led, όταν ενεργοποιείται η θύρα της στήλης (HIGH 5V) ή απενεργοποιείται η θύρα της γραμμής (LOW 0V), δε περνάει ρεύμα και το led δεν ανάβει.

Μια αντίσταση σε σειρά παρεμβάλλεται μεταξύ κάθε στήλης της μήτρας LED και της ψηφιακής θύρας εισόδου. Η τιμή των αντιστάσεων επιλέγεται έτσι ώστε να διασφαλίσουμε ότι το

μέγιστο ρεύμα Ic που διαπερνά μια θύρα του arduino δεν υπερβαίνει τα 40mA. Και αυτό διότι το ρεύμα για 8 led κάθε στήλης που θα συνδεθεί με μία θύρα του μικροελεγκτή θα πρέπει να είναι το 1/8 των 40mA, δηλαδή 5mA. Κάθε φωτοδίοδος σε ένα τυπικό led matrix απαιτεί τάση περίπου 1.8 - 2.2V. Υπολογίζοντας την αντίσταση μιας στήλης ή γραμμής για τάση 2V μας δίνει αντίσταση τιμής περίπου 470Ω.

Στο πρόγραμμα, ο βρόγχος for σκανάρει κάθε γραμμή και στήλη και ανάβει διαδοχικά τα led μέχρι όλα να φωτιστούνε. Ο βρόγχος ξεκινάει με τη $1^{η}$ στήλη και $1^{η}$ γραμμή και αυξάνει το μετρητή των γραμμών μέχρι όλα τα led σε αυτή τη γραμμή να ανάψουν. Μετά προχωράει στην επόμενη στήλη και συνεχίζει με τον ίδιο τρόπο ανάβοντας το επόμενο led διαδοχικά με κάθε πέρασμα του βρόγχου. Μπορούμε να ελέγξουμε τον αριθμό των φωτιζόμενων led σε συνάρτηση με την τιμή ενός αναλογικού αισθητήρα, όπως το ποτενσιόμετρο.

5ο Εργαστηριακό μάθημα

ΑΙΣΘΗΤΗΡΕΣ

Ultrasonic Distance Sensor (Υπερηχητικός αισθητήρας - Αποστασιόμετρο)

Ir Infrared Receiver (Υπέρυθρος δέκτης)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός :

Υπερηχητικού αισθητήρα.

Δέκτη υπερύθρων.

Θεωρητική εισαγωγή

Οι αισθητήρες απόστασης Ultrasonic χρησιμοποιούν τον ήχο για να μετρήσουν την απόσταση. Αυτοί οι αισθητήρες εκπέμπουν μια υπερηχητική παλμική ακολουθία και στη συνέχεια μετρούν το χρόνο μέχρι να ανακλαστεί ο παλμός. Οι ερασιτεχνικοί αισθητήρες χρησιμοποιούνται για μέτρηση απόστασης που κυμαίνεται από μερικά εκατοστόμετρα μέχρι μερικά μέτρα. Δεδομένου ότι η ταχύτητα του ήχου είναι σταθερή στον αέρα (περίπου 343m/s) είναι εύκολο να υπολογιστεί η απόσταση από το ανακλώμενο αντικείμενο. Η αρχή λειτουργίας τους, η οποία βασίζεται στην εκπομπή και λήψη υπερήχων, απαντάται δε και στη φύση στον ηχοεντοπισμό που χρησιμοποιούν τα δελφίνια και οι νυχτερίδες.

Ο αισθητήρας λειτουργεί στέλνοντας μια ομοβροντία (κατά ριπές) υπερήχων και ακούγοντας την ηχώ τους όταν ανακλάται από ένα αντικείμενο, παράγει παλμούς εξόδου, κάθε ένας από τους οποίους, αντιστοιχεί στον χρόνο που απαιτείται ώστε να επιστρέψει στον αισθητήρα η ηχώ της κάθε ριπής. Μετρώντας το εύρος του παλμού αυτού, η απόσταση από τον στόχο μπορεί εύκολα να υπολογιστεί. Ο αισθητήρας συγκεκριμένα στέλνει 8 ριπές παλμών στα 40kHz και περιμένει να ανιχνεύσει αν υπάρχει αντίλαλος, η κάθε ριπή ταξιδεύει στον αέρα (και σε θερμοκρασία δωματίου 20-25^{o} C) με ταχύτητα 343m/s. Ο ανιχνευτής υπερήχων θα ανιχνεύσει την απόσταση από το πλησιέστερο αντικείμενο μπροστά από τον αισθητήρα (από 2cm έως 5m) με ακρίβεια 0,5cm.

Για την κατανόηση της λειτουργίας εκτελούμε το παρακάτω παράδειγμα, στο οποίο ο μικροελεγκτής σε χρόνο t1 στέλνει ένα σύντομο υπερηχητικό παλμό για να προκαλέσει το ερέθισμα (trigger), που θα ληφθεί πίσω σε χρόνο t2 σαν αντίλαλος (echo). Η διάρκεια του παλμού dt ισούται με το χρόνο που χρειάζεται ο ήχος να ταξιδέψει από τον αισθητήρα στο αντικείμενο και πίσω (dt = t2–t1). Ο ακριβής χρόνος που χρειάζεται ο ήχος να «χτυπήσει» σε εμπόδιο είναι dt/2.

Χρησιμοποιώντας την ταχύτητα του ήχου και με τη βοήθεια του χρόνου (t) θα βρούμε την απόσταση (s).

U=s/t

t1=0msec, t2=5msec

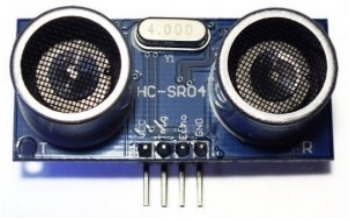
dt=t2-t1 ⇨ **dt=5-0=5msec**

s=u * t=343,2m/s * 0,005s=1,7m

Η ταχύτητα του ήχου είναι: 343.29 m/s = 0.034329 cm/microseconds.

Άρα ο τύπος εύρεσης της πραγματικής απόστασης που ταξιδεύει ο ήχος ή της απόστασης ανάμεσα στον αισθητήρα και το αντικείμενο (με μονάδα μέτρησης cm/microseconds) που θα χρησιμοποιήσουμε στην άσκησή μας είναι: **distance (cm) = (dt/2) * 0.034329**

ΑΣΚΗΣΗ 5.1



Σκοπός της Άσκησης:

Υπολογισμός απόστασης με υπερηχητικό αισθητήρα.

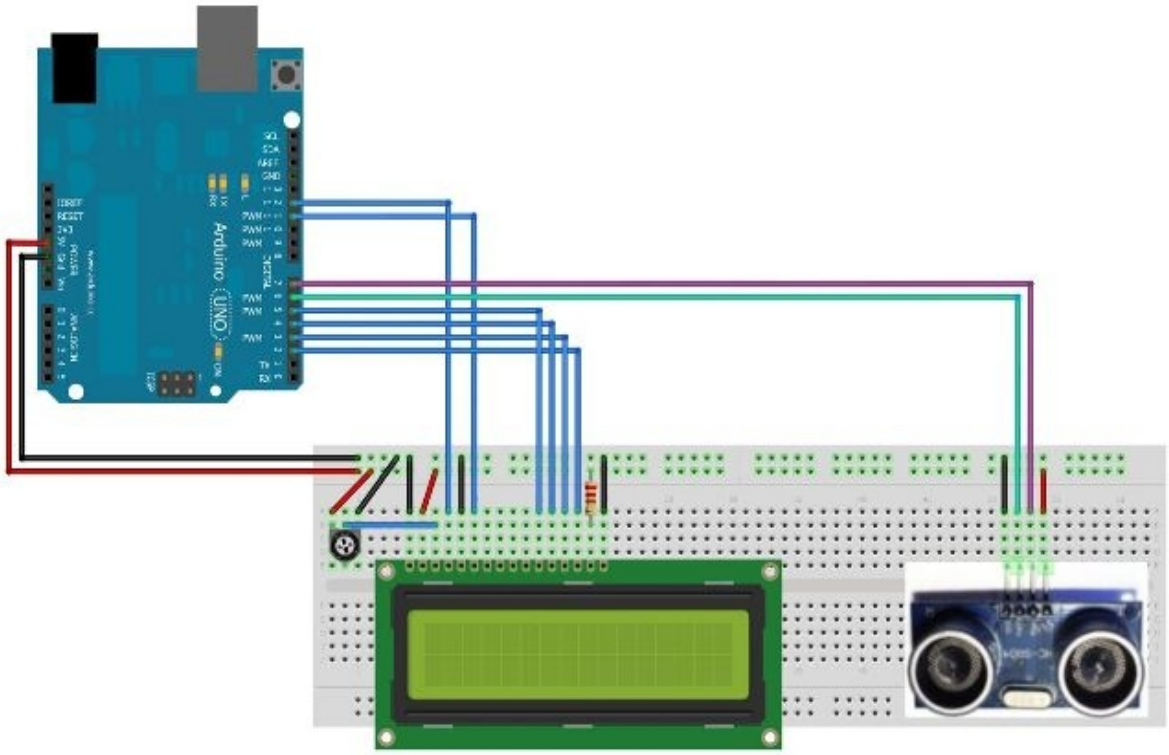
Απαιτούμενα Υλικά:

1. Υπερηχητικός αισθητήρας HC-SR04 
2. Οθόνη υγρών κρυστάλλων 16x2 (HD44780) 
3. Ποτενσιόμετρο 5 ΚΩ 
4. Αντίσταση 470 Ω 

Χαρακτηριστικά λειτουργίας HC-SR04:

Περιλαμβάνει πομπό υπερήχων, δέκτη και το κύκλωμα ελέγχου.
Τάση λειτουργίας: 5V DC
Ρεύμα λειτουργίας: 15 mA
Ρεύμα σε αδράνεια: <2mA
Δραστική γωνία: <15°
Trigger Pulse Input: 10μS TTL pulse
Echo Pulse Output: προτεινόμενος ~60ms παλμός
Ελάχιστη – Μέγιστη Εμβέλεια: 2cm - 500cm
Ακρίβεια μέτρησης: 3mm
Συχνότητα λειτουργίας: 40KHz
Διαστάσεις: 45*20*15mm

Συνδεσμολογία:



Σχήμα 45: Φυσική διάταξη του κυκλώματος συνδεσμολογίας υπερηχητικού αισθητήρα και οθόνης υγρών κρυστάλλων.

Το HC-SR04 αισθητήριο υπερήχων έχει 4 ακροδέκτες, δύο είναι για την τροφοδοσία και τη γείωση και οι άλλοι δύο αποτελούν το ερέθισμα (trigger) και τον αντίλαλο (echo). Με τη σειρά είναι οι: VCC, Trig, Echo, GND οι οποίοι συνδέονται αντίστοιχα με τις εξής θύρες του arduino 5V Supply, Trigger Pulse Input (Pin 9) (7 στο σχήμα), Echo Pulse Output (Pin 8) (6 στο σχήμα), GND. Η γωνία λειτουργίας του, περιορίζεται στις 15 μοίρες.

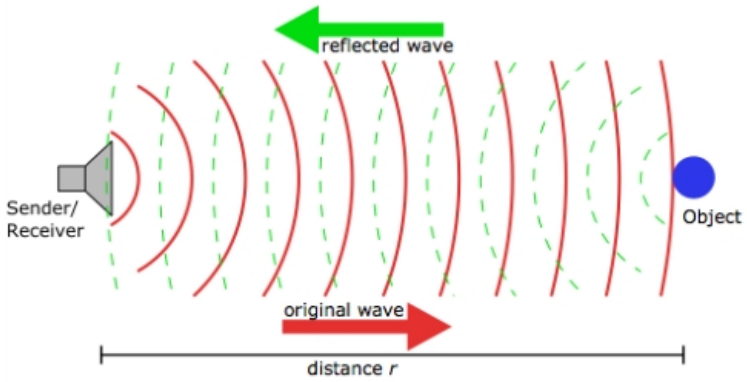
Πρόγραμμα 5.1.1

```
#define trigPin 9
#define echoPin 8
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
      pinMode(trigPin, OUTPUT);
      pinMode(echoPin, INPUT);
      lcd.begin(16, 2);
      lcd.print("Welcome To");
      lcd.setCursor(0, 1);
      lcd.print("Distance Meter");
      delay(2000);
}

void loop()
{
      int timetaken, dist;
      lcd.clear();
```

```
      lcd.print("Distance (cm):");
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);
      timetaken = pulseIn(echoPin, HIGH);
      dist = (timetaken/2) * 0.034049 ;
      if (dist >= 300 || dist <= 0){
         lcd.setCursor(0, 1);
         lcd.print("Out Of Range");
      }
      else
      {
         lcd.setCursor(0, 1);
         lcd.print(dist);
      }
delay(500);
}
```

Πρόγραμμα 5.1.2

```
// Υλοποίηση του ίδιου προγράμματος με χρήση της βιβλιοθήκης <Ultrasonic.h>
// http://ultrasonic-ranging-module-hc-sr04-updates.googlecode.com/files/Ultrasonic.rar

#include "Ultrasonic.h"
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
Ultrasonic ultrasonic(9, 8);

void setup()
{
      lcd.begin(16, 2);
      lcd.print("testing...");
}
void loop()
{
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("UltraSonic");
      lcd.setCursor(0, 1);
      lcd.print("Distance: ");
      lcd.print(ultrasonic.Ranging(CM));
      lcd.print("cm");

  delay(100);
}
```

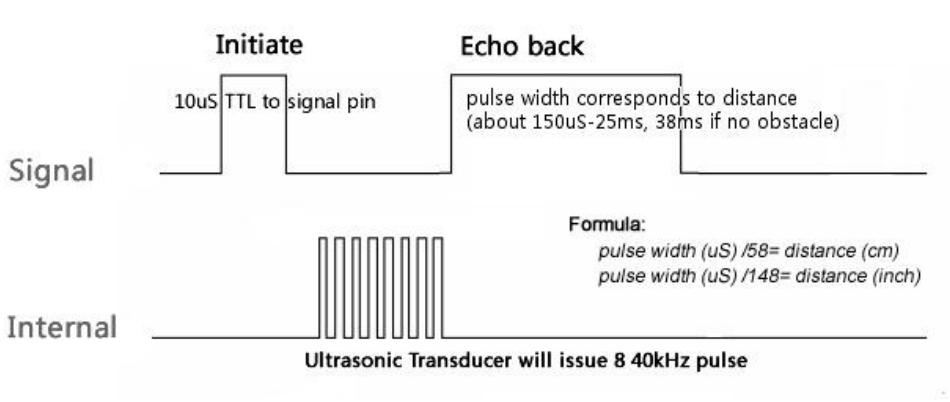


Εικόνα 21: Ανάκλαση υπερηχητικού κύματος.

Συμπεράσματα - Παρατηρήσεις

Το HC-SR04 αισθητήριο υπερήχων αποτελείται από δικό του ολοκληρωμένο κύκλωμα, στο οποίο υπάρχει ενσωματωμένος ελεγκτής, κάποια απαραίτητα φίλτρα και ρολόι χρονισμού. Ο δέκτης και ο πομπός είναι ανεξάρτητοι.

Συνδέουμε τους ακροδέκτες echo και trigger σε δύο ψηφιακές θύρες του Arduino. Με το trigger παράγουμε το ερέθισμα, δημιουργώντας με τον μικροελεγκτή μας έναν παλμό πλάτους 10 microseconds. Αυτός ο παλμός δίνει την εντολή στον ελεγκτή του αισθητηρίου να αρχίσει να παράγει με την βοήθεια του εκπομπού μια ακολουθία υπέρηχων συχνότητας 40kHz. Ο μικροελεγκτής του Arduino, από τη στιγμή που θα στείλει τον παλμό των 10 microseconds, μπαίνει σε έναν ατέρμονα βρόχο και περιμένει μέχρι να λάβει τον επιστρεφόμενο παλμό με τον ακροδέκτη echo. Το arduino υπολογίζει πόσο χρονικό διάστημα διανύθηκε από την ώρα που ξεκίνησε η εκπομπή του σήματος και με δεδομένη την ταχύτητα του ήχου στον αέρα, υπολογίζει την απόσταση του αντικειμένου.



Σχήμα 46: Χρονικό διάγραμμα λειτουργίας υπερηχητικού αισθητήρα HC-SR04.

Το αισθητήριο υπερήχων έχει εφαρμογή και ως εντοπιστής εμποδίων. Έτσι αν μπροστά από το αισθητήριο υπάρχει κάποιο εμπόδιο, τότε μέρος από τον υπέρηχο γυρίζει πίσω, λαμβάνεται από το δέκτη και παράγεται ο παλμός στο echo. Αν όμως δεν παρουσιαστεί εμπόδιο μπροστά από το αισθητήριο, τότε μετά από 60ms ο ελεγκτής του αισθητηρίου δημιουργεί τεχνητά τον παλμό στο echo, για να μην απασχολείται άδικα ο μικροελεγκτής μας.

Στο βρόγχο loop του προγράμματος μας, ξεκινάμε διαβάζοντας τον παλμό από τη θύρα echo και τον αποθηκεύουμε στη μεταβλητή timetaken.

```
timetaken = pulseIn(echoPin, HIGH);
```

Η εντολή pulseIn() είναι σχεδιασμένη να μετράει το μήκος του παλμού σε μs σε μια θύρα. Ο παλμός αναλύεται χρησιμοποιώντας τη pulseIn() συνάρτηση η οποία απαιτεί 2 παραμέτρους. Η πρώτη παράμετρος είναι η θύρα echo και η δεύτερη είναι: είτε HIGH είτε LOW παράμετρος, που προσδιορίζει σε ποια κατάσταση θα ξεκινήσει η εντολή pulseIn(), τη χρονομέτρηση του παλμού. Στην περίπτωσή μας θέτουμε κατάσταση HIGH, ώστε με το που θα γίνει το echoPin HIGH, η pulseIn() να αρχίσει τη χρονομέτρηση του παλμού, ενώ όταν γίνει LOW θα σταματήσει και θα επιστρέψει το χρόνο σε microseconds. Ο χρόνος που μεσολαβεί ανάμεσα στις αλλαγές κατάστασης του echoPin -μετά από επεξεργασία- θα μας δώσει την απόσταση που ψάχνουμε.

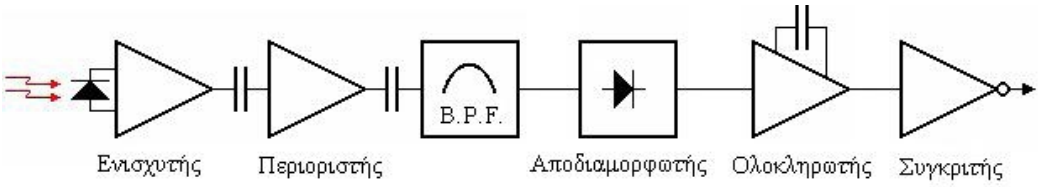
Ir Infrared Receiver (Υπέρυθρος δέκτης)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός του δέκτη υπερύθρων.

Θεωρητική εισαγωγή

Ολοκληρωμένος δέκτης υπερύθρων (αισθητήρας υπερύθρων): Εξαιτίας της μεγάλης τυποποίησης στον τηλεχειρισμό με υπέρυθρες, έχει αναπτυχθεί και δημιουργηθεί ένας μεγάλος αριθμός από έτοιμους ολοκληρωμένους δέκτες υπερύθρων. Σήμερα κατασκευάζονται έτοιμα ολοκληρωμένα κυκλώματα (συνήθως 3 ακροδεκτών) προστατευμένα μέσα σε ένα ειδικό πλαστικό περίβλημα (μαύρου χρώματος), που ταυτόχρονα λειτουργεί και ως φίλτρο ορατού φωτός (δηλαδή επιτρέπει τη διέλευση μόνο της υπέρυθρης ακτινοβολίας). Μέσα σε αυτά τα ολοκληρωμένα ενσωματώνεται ο IR δέκτης (PIN δίοδος) καθώς και το απαραίτητο ηλεκτρονικό κύκλωμα.

Το σημαντικότερο κριτήριο επιλογής ενός τέτοιου δέκτη, είναι η συχνότητα του φέροντος κύματος που χρησιμοποιείται στη διαμόρφωση, η οποία στον δέκτη μεταφράζεται ως η κεντρική συχνότητα ζωνοπερατού φίλτρου. Στο παρακάτω σχήμα παρουσιάζεται το μπλοκ διάγραμμα ενός τυπικού δέκτη.



Σχήμα 47: Το μπλοκ διάγραμμα ενός ολοκληρωμένου δέκτη υπερύθρων.

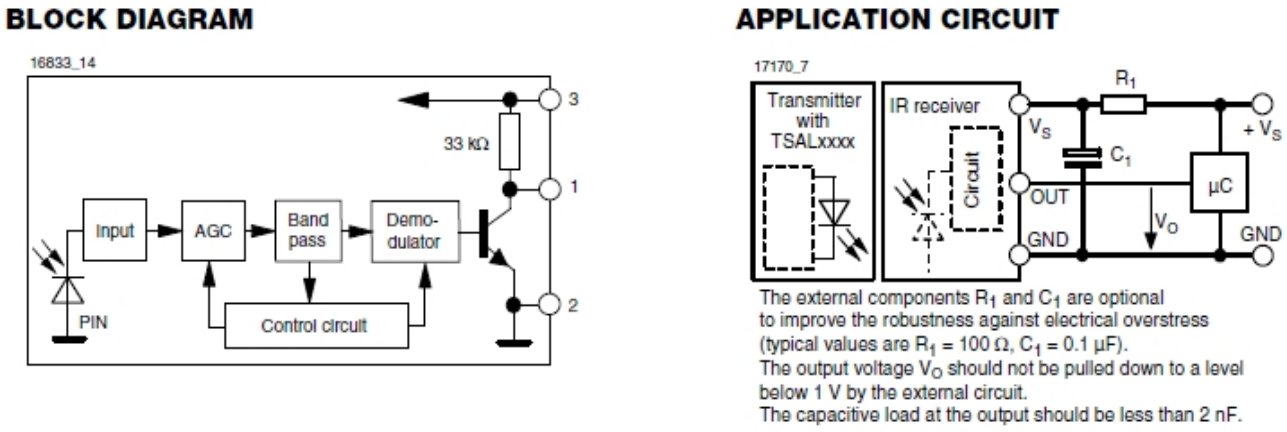
Το σήμα IR λαμβάνεται από τον δέκτη και ενισχύεται από τον ενισχυτή που ακολουθεί. Ο περιοριστής λειτουργεί ως ένα κύκλωμα αυτόματης ρύθμισης της απολαβής, αφού στην έξοδό του λαμβάνεται ένα σήμα σταθερού πλάτους, ανεξάρτητα της απόστασης της συσκευής. Στο ζωνοπερατό φίλτρο οδηγείται μόνο το ac σήμα (εξαιτίας του πυκνωτή σύζευξης). Η κεντρική συχνότητα του φίλτρου είναι ίση με τη συχνότητα του διαμορφωμένου φέροντος του πομπού. Οποιαδήποτε άλλη συχνότητα απορρίπτεται. Οι συνηθισμένες συχνότητες που χρησιμοποιούνται στο εμπόριο κυμαίνονται στην περιοχή από 30KHz έως 60KHz. Οι επόμενες βαθμίδες είναι ο αποδιαμορφωτής, ο ολοκληρωτής και ο συγκριτής. Ο σκοπός τους είναι να ανιχνεύσουν την παρουσία της φέρουσας συχνότητας. Όσο αυτή είναι παρούσα, η έξοδος του συγκριτή μεταβαίνει σε χαμηλή στάθμη (0V). Στην αντίθετη περίπτωση παραμένει σε υψηλή λογική στάθμη (5V).

Χαρακτηριστικά λειτουργίας του IR Receiver Tsop4838:

Ενσωματώνει τον αισθητήρα υπερύθρων και τον προ-ενισχυτή σε ένα πακέτο.
Το πλαστικό που το προστατεύει από διάφορους εξωγενείς παράγοντες, δουλεύει και ως φίλτρο.
Δεν επηρεάζεται από ηλεκτρικές παρεμβολές.
Υποστηρίζει TTL και CMOS.
Έχει χαμηλή ενεργειακή έξοδο.
Καταναλώνει λίγο ρεύμα. Δουλεύει με τροφοδοσία ρεύματος από 2.7V μέχρι 5.5 V.
Δεν περιέχει μόλυβδο στην κατασκευή του.
Δεν επηρεάζεται από τον περιβάλλοντα φωτισμό.

Ο Δέκτης Υπερύθρων Tsop38xx ή Tsop48xx της εταιρίας Vishay είναι μια PIN δίοδος με προ-ενισχυτή και το πλαστικό που την περιβάλλει λειτουργεί ως υπέρυθρο φίλτρο. Έχει την δυνατότητα να λαμβάνει το από-διαμορφωμένο υπέρυθρο σήμα που στέλνει το τηλεκοντρόλ και μέσω του μικροελεγκτή να γίνεται η αποκωδικοποίηση του.

Το κύκλωμα της σειράς Tsop είναι κατασκευασμένο με τέτοιο τρόπο ώστε να αποφεύγονται οι ανεπιθύμητοι παλμοί που παράγονται λόγω θορύβου. Το ζωνοπερατό φίλτρο στο στάδιο ολοκλήρωσης και ο αυτόματος έλεγχος κέρδους (gain control) χρησιμοποιούνται για να αποκόψουν τέτοιες παρεμβολές.



Σχήμα 48: Κύκλωμα εφαρμογής του δέκτη υπερύθρων Tsop4838.

ΑΣΚΗΣΗ 5.2

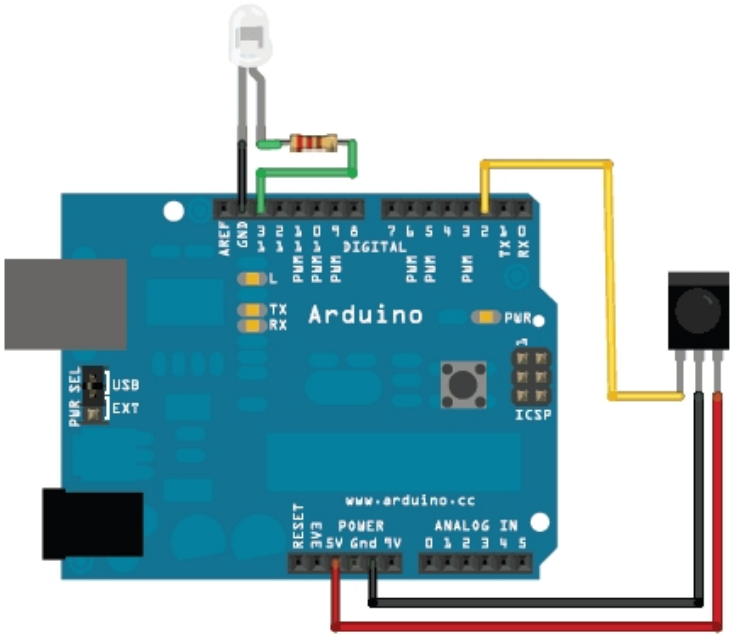
Σκοπός της Άσκησης:

Θέλουμε ο μικροελεγκτής να αντιλαμβάνεται και να αποκρίνεται σε οποιοδήποτε υπέρυθρο τηλεχειριστήριο. Μία φωτοδίοδος ανάβει κάθε φορά που ένα κουμπί υπέρυθρου τηλεχειριστηρίου ενεργοποιείται προς το μέρος του δέκτη.

Απαιτούμενα Υλικά:

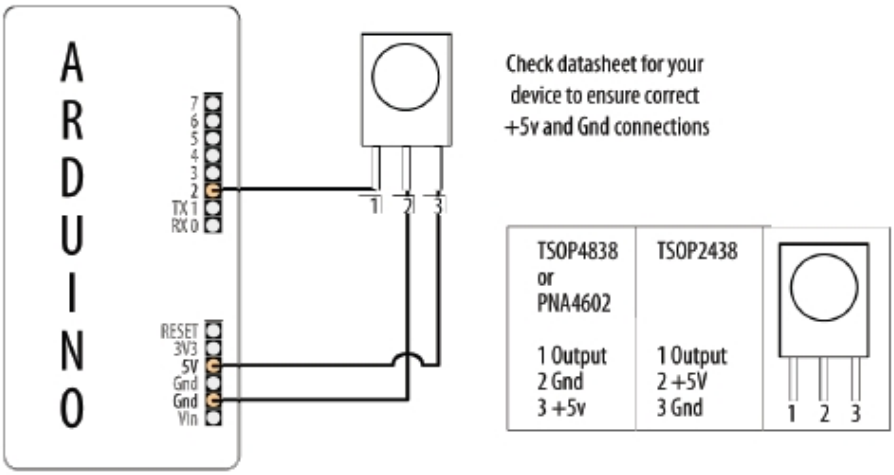
1. Δέκτης Υπερύθρων TSOP4838
2. Κοινό Υπέρυθρο Τηλεχειριστήριο (π.χ. Sony TV)
3. LED 5mm 

Συνδεσμολογία:



Σχήμα 49: Φυσική διάταξη του κυκλώματος σύνδεσης υπέρυθρου δέκτη.

Σχηματικό διάγραμμα:



Σχήμα 50: Συνδεσμολογία κυκλώματος υπέρυθρου δέκτη.

Το Arduino ανταποκρίνεται σε υπέρυθρα σήματα χρησιμοποιώντας το δέκτη υπερύθρων. Μερικοί κοινοί δέκτες είναι TSOPxx38 και PNA4602. Ελέγχουμε το φύλλο δεδομένων για τη σωστή συνδεσμολογία.

Για το TSOP4838 έχουμε 3 ακροδέκτες:

1ος : έξοδος, **2ος :** γείωση, **3ος :** 5V.

Στο παρακάτω πρόγραμμα χρησιμοποιούμε την IRremote library που υπάρχει εδώ:
http://www.arcfn.com/2009/08/multi-protocol-infrared-remote-library.html

Πρόγραμμα 5.2

```
/*
  IR_remote_detector
  An IR remote receiver is connected to pin 2.
  The LED on pin 13 toggles each time a button on the remote is pressed.
 */

#include <IRremote.h>                //adds the library code to the sketch

const int irReceiverPin = 2;         //pin the receiver is connected to
const int ledPin = 13;

IRrecv irrecv(irReceiverPin);        //create an IRrecv object
decode_results decodedSignal;        //stores results from IR detector

void setup()
{
  pinMode(ledPin, OUTPUT);
  irrecv.enableIRIn();               // Start the receiver object
}

boolean lightState = false;      //keep track of whether the LED is on
unsigned long last = millis();   //remember when we last received an IR message

void loop()
{
  if (irrecv.decode(&decodedSignal) == true) //this is true if a message
                                             //has been received
  {
    if (millis() - last > 250) {     //has it been 1/4 sec since last message
      lightState = !lightState;      //toggle the LED
      digitalWrite(ledPin, lightState);
    }
    last = millis();
    irrecv.resume();                 // watch out for another message
  }
}
```

Συμπεράσματα - Παρατηρήσεις

Παρατηρούμε ότι το led ανάβει κάθε φορά που δέχεται σήμα από ένα υπέρυθρο τηλεκοντρόλ, οποιοδήποτε πλήκτρο και αν πατηθεί. Ο δέκτης υπερύθρων μετατρέπει τα IR σήματα σε ψηφιακούς παλμούς (αλληλουχίες από μηδενικά και άσσους) οι οποίοι αντιστοιχούν στα κουμπιά του χειριστηρίου. Η IR βιβλιοθήκη αποκωδικοποιεί αυτούς τους παλμούς και παρέχει μια αριθμητική τιμή για κάθε κουμπί. Οι πραγματικές τιμές που λαμβάνουμε εξαρτώνται κάθε φορά από το συγκεκριμένο τηλεκοντρόλ που χρησιμοποιούμε.

ΑΣΚΗΣΗ 5.3

Σκοπός της Άσκησης:

Αποκωδικοποίηση σημάτων υπέρυθρου τηλεχειριστηρίου. Το arduino αντιλαμβάνεται, αποκρίνεται, αποθηκεύει στη μνήμη του και αναγνωρίζει το πλήκτρο του υπέρυθρου τηλεχειριστήριου που πιέζεται κάθε φορά.

Απαιτούμενα Υλικά:

1. Δέκτης Υπερύθρων TSOP4838
2. Κοινό Υπέρυθρο Τηλεχειριστήριο (π.χ. Sony TV)
3. LED 5mm 

Συνδεσμολογία:

Το σχηματικό διάγραμμα είναι το ίδιο με την προηγούμενη άσκηση 5.2

Πρόγραμμα 5.3

```
/*
 RemoteDecode sketch
 Infrared remote control signals are decoded to control LED brightness
 The values for keys 0 through 4 are detected and stored when the sketch starts
 key 0 turns the LED off, the brightness increases in steps with keys 1 through
4
 */

#include <IRremote.h>          // IR remote control library

const int irReceivePin = 2;    // pin connected to IR detector output
const int ledPin  = 9;         // LED is connected to a PWM pin

const int numberOfKeys = 10;  //  10 keys are learned (0 through 9)
long irKeyCodes[numberOfKeys];// holds the codes for each key

IRrecv irrecv(irReceivePin);  // create the IR library
decode_results results;       // IR data goes here

void setup()
{
  Serial.begin(9600);
  pinMode(irReceivePin, INPUT);
  pinMode(ledPin, OUTPUT);
  irrecv.enableIRIn();        // Start the IR receiver
  learnKeycodes();            // learn remote control key  codes
  Serial.println("Press a remote key");
}

void loop()
{
  long key;
  int  brightness;

  if (irrecv.decode(&results))
  {
    // here if data is received
    irrecv.resume();
    key = convertCodeToKey(results.value);
    if(key >= 0)
    {
      Serial.print("Got key ");
```

```
      Serial.println(key);
      brightness = map(key, 0,numberOfKeys-1, 0, 255);
      analogWrite(ledPin, brightness);
    }
  }
}

/*
 * get remote control codes
 */
void learnKeycodes()
{
  while(irecv.decode(&results))   // empty the buffer
    irecv.resume();

  Serial.println("Ready to learn remote codes");
  long prevValue = -1;
  int i=0;
  while( i < numberOfKeys)
  {
    Serial.print("press remote key ");
    Serial.print(i);
    while(true)
    {
      if( irecv.decode(&results) )
      {
          if(results.value != -1 && results.value != prevValue)
          {
            showReceivedData();
            irKeyCodes[i] = results.value;
            i = i + 1;
            prevValue = results.value;
            irrecv.resume(); // Receive the next value
            break;
          }
        irrecv.resume(); // Receive the next value
      }
    }
  }
  Serial.println("Learning complete");
}

/*
 * converts a remote protocol code to a logical key code
 * (or -1 if no digit received)
 */
int convertCodeToKey(long code)
{
  for( int i=0; i < numberOfKeys; i++)
  {
    if( code == irKeyCodes[i])
    {
      return i; // found the key so return it
    }
  }
  return -1;
}

/*
 * display the protocol type and value
 */
void showReceivedData()
{
  if (results.decode_type == UNKNOWN)
```

```
  {
    Serial.println("-Could not decode message");
  }
  else
  {
    if (results.decode_type == NEC) {
      Serial.print("- decoded NEC: ");
    }
    else if (results.decode_type == SONY) {
      Serial.print("- decoded SONY: ");
    }
    else if (results.decode_type == RC5) {
      Serial.print("- decoded Philips RC5: ");
    }
    else if (results.decode_type == RC6) {
      Serial.print("- decoded Philips RC6: ");
    }
    Serial.print("hex value = ");
    Serial.println( results.value, HEX);
  }
}
```

Περιγραφή της Άσκησης:

Στην άσκηση 5.3 κάθε φορά που επικοινωνεί το τηλεχειριστήριο με το δέκτη, ο μικροελεγκτής αποκωδικοποιεί το υπέρυθρο σήμα και το εμφανίζει στη σειριακή οθόνη. Το πρόγραμμα αρχικά μας ζητάει 10 πλήκτρα (π.χ. από 0 έως 9, channel up, volume κλπ), τα οποία αποθηκεύονται στη μνήμη του arduino. Ταυτόχρονα αποκωδικοποιεί το σήμα σε 16δικη μορφή και το εμφανίζει στην οθόνη. Μόλις γίνει η εισαγωγή των 10 συχνοτήτων το πρόγραμμα ανταποκρίνεται στα ίδια πλήκτρα αυξάνοντας τη φωτεινότητα του led, με το $1^{ο}$ πλήκτρο (π.χ. 0) να αντιστοιχεί σε σβηστό led και το $10^{ο}$ πλήκτρο (π.χ. 9) στη μέγιστη φωτεινότητα, εμφανίζοντας ταυτόχρονα στην οθόνη το νούμερο του πλήκτρου.

6ο Εργαστηριακό μάθημα

Real Time Clock (RTC) Το ρολόι πραγματικού χρόνου

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός ρολογιού πραγματικού χρόνου.

Θεωρητική εισαγωγή

Η Μονάδα Ρολογιού Πραγματικού Χρόνου (Real Time Clock ή RTC) χρησιμοποιείται ως μετρητής χρόνου σε διάφορες εφαρμογές. Το ολοκληρωμένο DS1302 της Dallas Semiconductor (Maxim-IC) έχει τα παρακάτω χαρακτηριστικά:

• τάση τροφοδοσίας από 2V μέχρι 5.5V

• μετράει ώρα, λεπτά, δευτερόλεπτα, ημέρα, μήνα και έτος

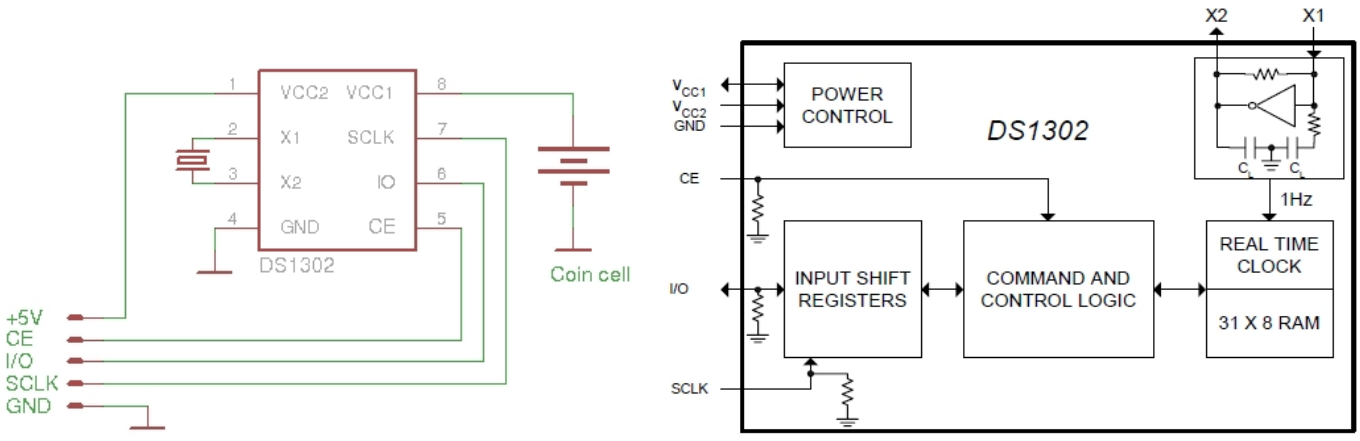
• έχει αυτόματο υπολογισμό δίσεκτων ετών

• απλό σειριακό interface 3-αγωγών (3-wire)

• μικρή κατανάλωση (<300nA)

• ενσωματωμένο κύκλωμα φόρτισης μπαταρίας

Το ρολόι πραγματικού χρόνου πρέπει να λειτουργεί ανεξάρτητα από τα υπόλοιπα κυκλώματα της συσκευής και να τροφοδοτείται και από μπαταρία, ώστε σε περίπτωση μη επαρκούς τροφοδοσίας της συσκευής, ο χρόνος να μη χάνεται. Όταν επανέλθει η τροφοδοσία, η συσκευή θα συνεχίσει να λειτουργεί κανονικά, καταγράφοντας μετρήσεις με σωστό χρονικό στίγμα. Η μπαταρία είναι συνήθως επαναφορτιζόμενη που χρησιμοποιείται για να τροφοδοτεί το DS1302 όταν η κανονική τάση τροφοδοσίας του κυκλώματος (3.3V ή 5V) απουσιάζει. Σε κανονική τάση τροφοδοσίας, η μπαταρία φορτίζεται μέσω του εσωτερικού φορτιστή του DS1302.



Σχήμα 51: Συνδεσμολογία ρολογιού πραγματικού χρόνου DS1302 και Χονδρικό διάγραμμα λειτουργίας DS1302.

Η επικοινωνία του DS1302 με τον μικροελεγκτή γίνεται μέσω των τριών γραμμών επικοινωνίας CE, SCLK και I/O. Η λειτουργία των τριών σημάτων είναι:

• **CE** – chip enable, χρησιμοποιείται από τον μικροελεγκτή για να ενεργοποιήσει το ολοκληρωμένο κύκλωμα.

• **SCLK** – synchronous clock, είσοδος στην οποία παρέχει ο μικροελεγκτής παλμούς χρονισμού.

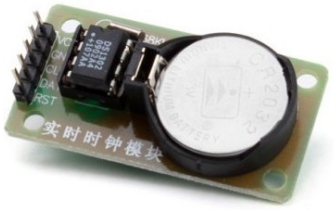
• **IO** – input/output, είσοδος ή έξοδος των σύγχρονων σειριακών δεδομένων χρόνου.

Τα έτοιμα real-time clock module περιλαμβάνουν: α) τον κρύσταλλο ο οποίος συνδέεται στα **X1** και **X2**, έχει συχνότητα ταλάντωσης τα 32.768kHz και χρησιμοποιείται για το χρονισμό του ρολογιού του DS1302 (ο χρόνος εκκίνησης του κρυστάλλου είναι λιγότερο από 1 δευτερόλεπτο). Η ακρίβεια του ρολογιού εξαρτάται από την ακρίβεια του κρυστάλλου αλλά και β) τη μπαταρία εφεδρικής τροφοδοσίας ώστε η πληροφορία να παραμένει σωστή όταν το arduino επανεκκινεί ή κλείνει. Έτσι σε περίπτωση διακοπής της τροφοδοσίας η μικρή μπαταρία 3V που διαθέτει, μπορεί να κρατήσει τον εσωτερικό ταλαντωτή του RTC σε λειτουργία για χρόνια. Το **Vcc2** πρέπει να συνδεθεί στα 5V ή 3.3V. Το **Vcc1** προορίζεται για την προαιρετική μπαταρία εφεδρικής τροφοδοσίας.

Το ολοκληρωμένο κύκλωμα RTC διαθέτει εσωτερική μνήμη SRAM, χωρητικότητας 31 bytes. Το εσωτερικό τμήμα σειριακής επικοινωνίας χρησιμοποιεί μόνο τρεις αγωγούς σύνδεσης με έναν μικροεπεξεργαστή. Οι πληροφορίες που παρέχει είναι δευτερόλεπτα, λεπτά, ώρες, ημέρες, ημερομηνία, μήνες και χρόνια. Επίσης μπορεί να λειτουργήσει σε μορφή πληροφορίας 12-ωρου με ταυτόχρονη ένδειξη μεσημβρινής/απογευματινής (AM/PM) ώρας, ή σε μορφή πληροφορίας 24ώρου.

ΑΣΚΗΣΗ 6

Σκοπός της Άσκησης:



Εμφάνιση πραγματικής ώρας και ημερομηνίας με τη βοήθεια του RTC module DS1302 σε οθόνη υγρών κρυστάλλων LCD.

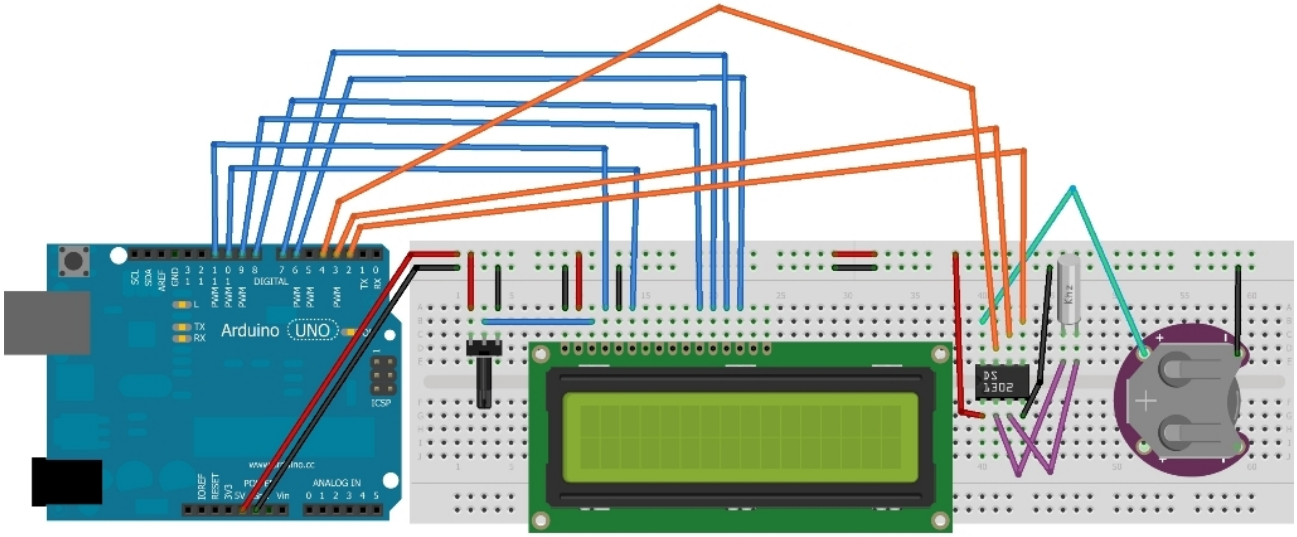
Απαιτούμενα Υλικά:

1. Ρολόι πραγματικού χρόνου DS1302 module.
2. Οθόνη υγρών κρυστάλλων 16x2 (HD44780) 
3. Ποτενσιόμετρο 

Χαρακτηριστικά λειτουργίας DS1302 **Real Time Clock Module** with Battery CR2032:

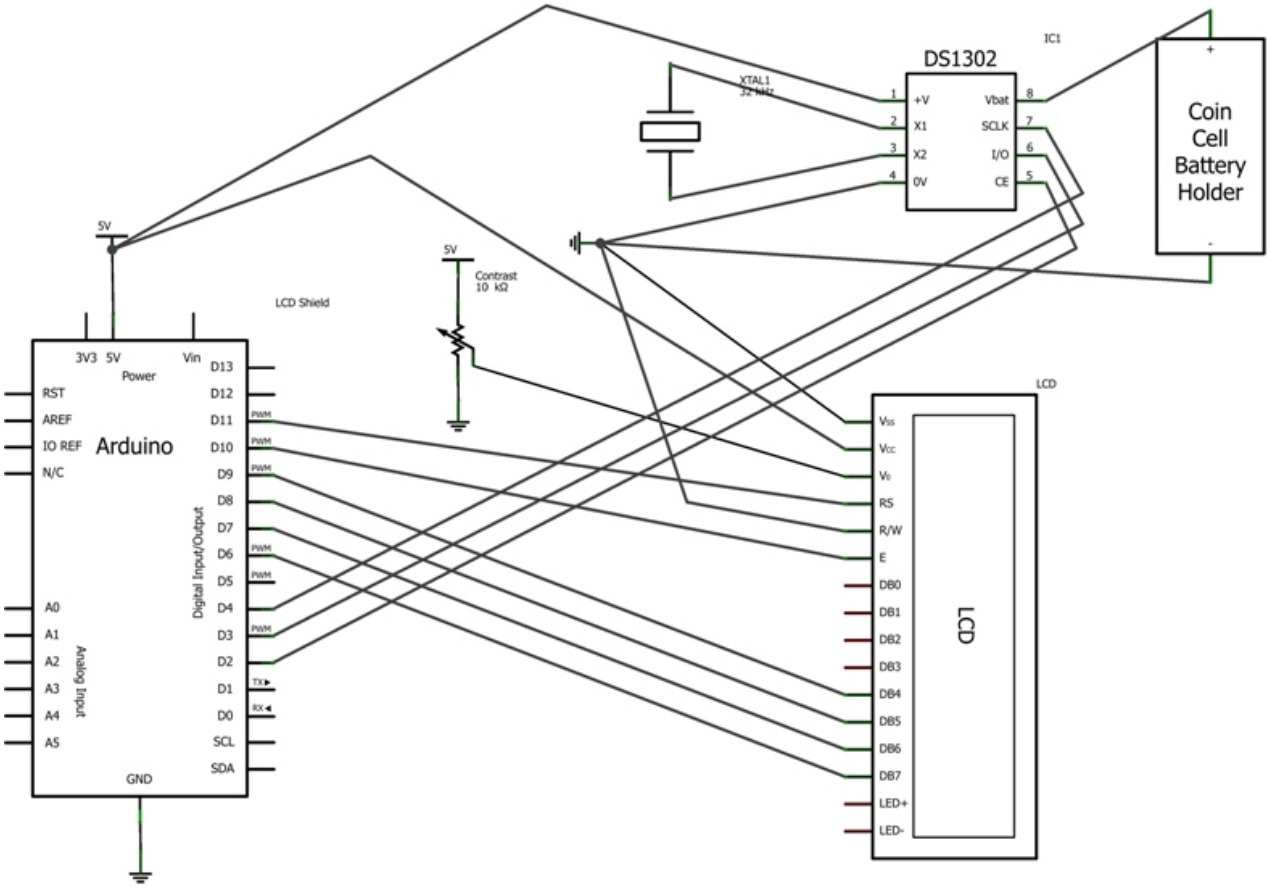
A Real Time Clock Module with battery backup using the easy to use DS1302 chip.
Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100
Serial I/O for Minimum Pin Count
2.0V to 5.5V Full Operation
Uses Less than 300nA at 2.0V
Single-Byte or Multiple-Byte (Burst Mode) Data Transfer for Read or Write of Clock or RAM Data
Board Size: 44mm x 24mm

Συνδεσμολογία:



Σχήμα 52: Φυσική διάταξη του κυκλώματος ρολογιού πραγματικού χρόνου DS1302.

Σχηματικό διάγραμμα:



Σχήμα 53: Συνδεσμολογία κυκλώματος ρολογιού πραγματικού χρόνου DS1302.

Στο παρακάτω πρόγραμμα χρησιμοποιούμε την DS1302 library που υπάρχει εδώ:
http://www.henningkarlsen.com/electronics/library.php?id=5

Πρόγραμμα 6

```
// A scetch of how to use DS1302-library to make a quick
// clock using a DS1302 and a 16x2 LCD.
//
// DS1302:  CE pin    -> Arduino Digital 2
//          I/O pin   -> Arduino Digital 3
//          SCLK pin  -> Arduino Digital 4
// LCD:     DB7       -> Arduino Digital 6
//          DB6       -> Arduino Digital 7
//          DB5       -> Arduino Digital 8
//          DB4       -> Arduino Digital 9
//          E         -> Arduino Digital 10
//          RS        -> Arduino Digital 11

#include <LiquidCrystal.h>
#include <DS1302.h>

DS1302 rtc(2, 3, 4);                        // Init the DS1302

LiquidCrystal lcd(11, 10, 9, 8, 7, 6);     // Init the LCD

void setup()
{
  // Set the clock to run-mode, and disable the write protection
  rtc.halt(false);
  rtc.writeProtect(false);

  // Setup LCD to 16x2 characters
  lcd.begin(16, 2);

  // The following lines can be commented out
  // to use the values already stored in the DS1302
  rtc.setDOW(FRIDAY);        // Set Day-of-Week to FRIDAY
  rtc.setTime(12, 0, 0);     // Set the time to 12:00:00 (24hr format)
  rtc.setDate(6, 8, 2012);   // Set the date to August 6th, 2012
}

void loop()
{
  // Display time centered on the upper line
  lcd.setCursor(4, 0);
  lcd.print(rtc.getTimeStr());

  // Display abbreviated Day-of-Week in the lower left corner
  lcd.setCursor(0, 1);
  lcd.print(rtc.getDOWStr(FORMAT_SHORT));

  // Display date in the lower right corner
  lcd.setCursor(6, 1);
  lcd.print(rtc.getDateStr());

  // Wait one second before repeating
  delay (1000);
}
```

Περιγραφή της Άσκησης

Χρησιμοποιώντας ένα real-time clock (RTC) τσιπάκι μπορούμε με τις κατάλληλες εντολές που του αποστέλλουμε να θέσουμε (γράψουμε) σε αυτό την ώρα και την ημερομηνία και έπειτα να

την διαβάσουμε από αυτό οποιαδήποτε στιγμή και να την εισάγουμε στη εφαρμογή μας. Χρησιμοποιείται συνήθως σε data-logging κατασκευές.

Στη εφαρμογή μας για την μέτρηση του χρόνου χρησιμοποιούμε το ολοκληρωμένο DS1302 της Dallas-Maxim. Κάνουμε χρήση της βιβλιοθήκης <DS1302.h> η οποία απλοποιεί τη χρήση του. Το ολοκληρωμένο τσιπ, μας παρέχει μέτρηση του πραγματικού χρόνου, τον οποίο εμφανίζουμε στη οθόνη LCD 16x2. Το DS1302 έχει τάση λειτουργίας από 2.0V έως 5.5V. Στα 2.0V έχει τη λιγότερο ενεργοβόρα λειτουργία, λιγότερο από 300nA. Άρα θα συνδέουμε την LCD οθόνη **και** το clock σε κοινή τροφοδοσία, στα 3.3V που μας παρέχει το arduino.

Συμπεράσματα - Παρατηρήσεις

Σύμφωνα με το επίσημο φύλλο δεδομένων του DS1302, στις τρεις γραμμές επικοινωνίας CE, SCLK και I/O πρέπει να αποφεύγεται η χρήση pull-up αντιστάσεων για την σωστή λειτουργία του. Το DS1302 έχει εσωτερικές pull-down αντιστάσεις 40ΚΩ σε κάθε αγωγό σύνδεσης και οι CE και SCLK θα είναι χαμηλές σε τάση όταν είναι ανενεργές.

Ωστόσο σε μερικά σχηματικά διαγράμματα από διάφορες πηγές (βιβλία και ιντερνέτ) έχουνε συνδεδεμένες pull-up αντιστάσεις και στις τρεις γραμμές επικοινωνίας καθώς και 2 πυκνωτές σε σειρά με το κρύσταλλο. Αυτό είναι λάθος, καθώς τονίζεται ότι το κύκλωμα του ταλαντωτή, δεν απαιτεί τη χρήση εξωτερικών αντιστάσεων ή πυκνωτών για να λειτουργήσει.

Πίνακας 8: Χαρακτηριστικά εξωτερικού κρυστάλλου DS1302

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
Nominal Frequency	fO		32.768		kHz
Series Resistance	ESR		45		kΩ
Load Capacitance	CL		6		pF

Το DS1302 δεν χρησιμοποιεί το I2C πρωτόκολλο, ούτε το SPI και δεν είναι OneWire συσκευή. Η πιο συχνή του ονομασία είναι "3-wire interface". Σε αντίθεση με το ολοκληρωμένο DS1307 που χρησιμοποιεί το I2C πρωτόκολλο και απαιτούνται pull-up αντιστάσεις όπως αναφέρει ο κατασκευαστής του (χωρίς εξωτερικές αντιστάσεις 2,2kΩ και 10kΩ στο clock και dataline τα αποτελέσματα θα είναι απρόβλεπτα).

Παρόλα αυτά, το παράδοξο στην κατασκευή μας είναι, ότι μετά από αλλεπάλληλες προσπάθειες λειτουργίας του ολοκληρωμένου DS1302, μέσω της διαδικασίας πείραμα-παρατήρηση, καταφέραμε να πάρουμε τις σωστές μετρήσεις και την απρόσκοπτη λειτουργία του για μέρες, μόνον όταν τροφοδοτήθηκε στα 3.3V και με 3 αντιστάτες 470Ω σε σειρά με κάθε αγωγό σύνδεσης CE, SCLK και I/O. Σε κάθε άλλη περίπτωση σύνδεσης στα 5V με ή χωρίς αντιστάτες, ή στα 3.3V με αντιστάτες διαφορετικής τιμής τα αποτελέσματα ήταν τα μη αναμενόμενα. Για παράδειγμα με αντιστάσεις 4.7ΚΩ, 2.2ΚΩ, 1ΚΩ, 680Ω, 220Ω οι αναγνώσεις κυμαίνονταν από 1 στις 100 λανθασμένες μέχρι όλες λανθασμένες.

7ο Εργαστηριακό μάθημα

KEYPAD matrix (Αριθμητικό πληκτρολόγιο)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός αριθμητικού πληκτρολογίου.

Θεωρητική εισαγωγή

Το αριθμητικό πληκτρολόγιο είναι μια συστοιχία από Κανονικά Ανοιχτούς (Normally Open) διακόπτες, οι οποίοι ενώνουν μια γραμμή με μια στήλη όταν πατηθούν. Βρίσκονται συνδεδεμένοι σε σειρά και παράλληλα, σχηματίζοντας πίνακα γραμμών και στηλών. Έτσι ενώ έχουμε 12 εισόδους, χρειαζόμαστε μόνο 7 εξόδους (3 στήλες και 4 γραμμές). Συνηθισμένα μεγέθη για εφαρμογές είναι 3x4 και 4x4. Παρακάτω θα συνδέσουμε ένα 3x4 matrix keypad το οποίο αποτελείται από 12 κουμπιά τακτοποιημένα σε στυλ τηλέφωνου. Κάθε μία από τις 4 γραμμές και 3 στήλες συνδέεται με μία θύρα του μικροελεγκτή.

ΑΣΚΗΣΗ 7

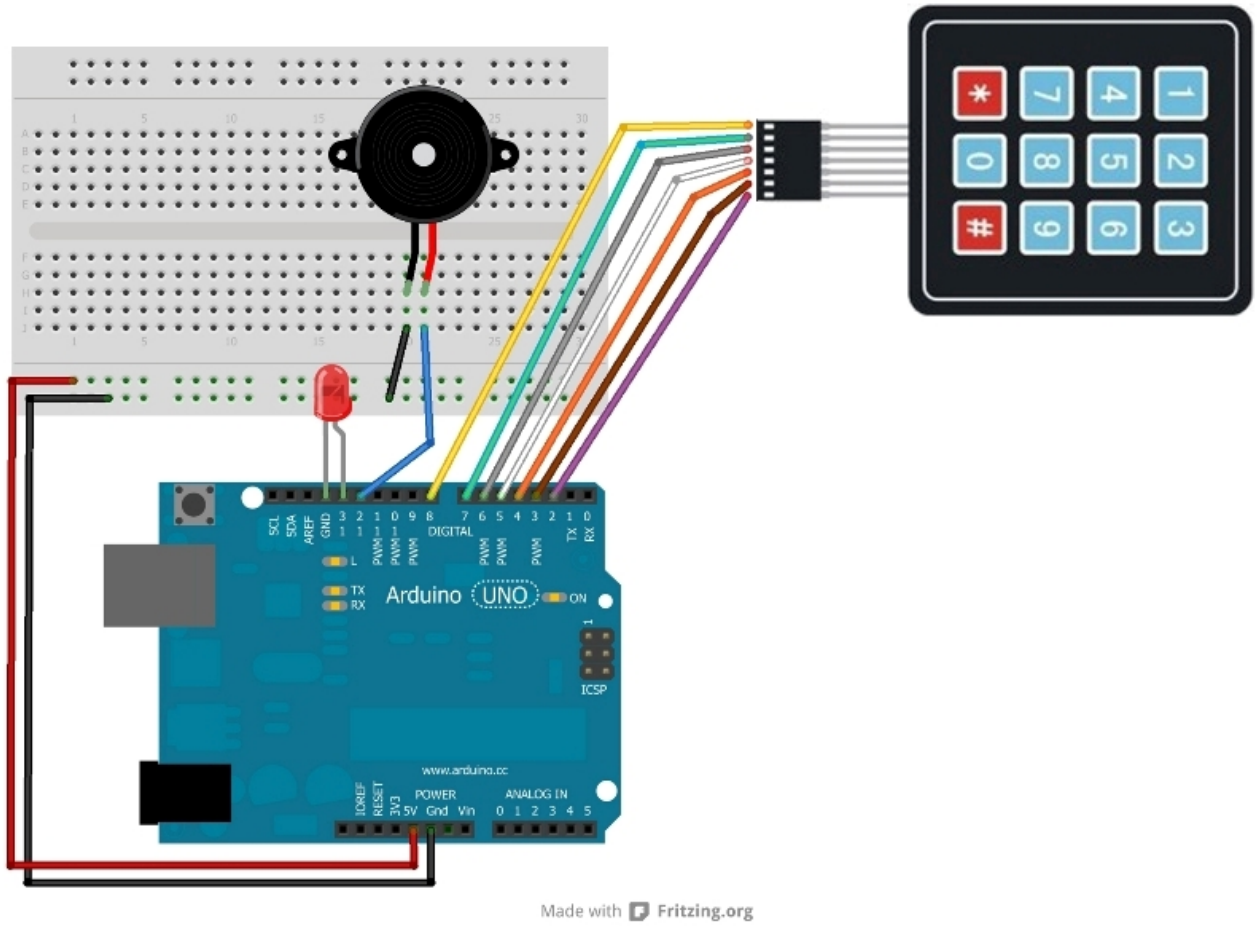
Σκοπός της Άσκησης:

Σύνδεση αριθμητικού πληκτρολογίου στο arduino. Κάθε φορά που πιέζεται ένα πλήκτρο να εμφανίζεται στο serial monitor. Ταυτόχρονα να ανάβει το led και να ηχεί το buzzer. Η # να αλλάζει τη κατάσταση του led (από αναμμένο σε σβησμένο και το αντίθετο) και το * όταν κρατηθεί πατημένο να αναβοσβήνει συνεχόμενα το led.

Απαιτούμενα Υλικά:

1. Αριθμητικό πληκτρολόγιο 3x4
2. Βομβητής (buzzer) 
3. Αντίσταση 470 Ω 
4. LED 

Συνδεσμολογία:



Σχήμα 54: Φυσική διάταξη του κυκλώματος συνδεσμολογίας Αριθμητικού πληκτρολογίου.

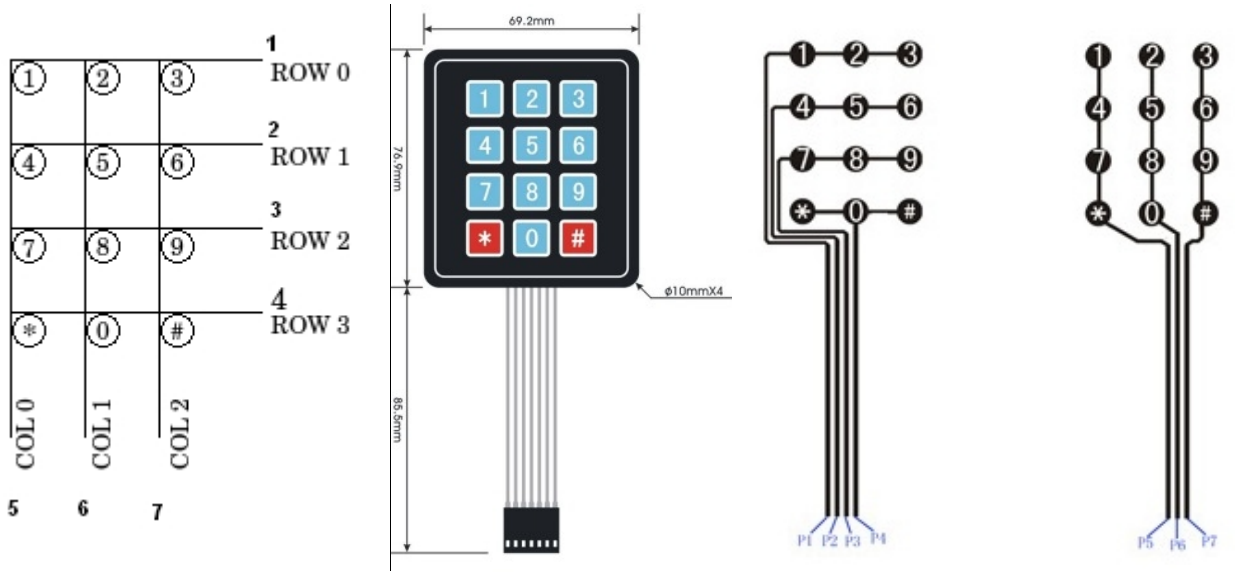
Στο παρακάτω πρόγραμμα χρησιμοποιούμε την Keypad library που υπάρχει εδώ:
http://playground.arduino.cc/code/Keypad

Πρόγραμμα 7

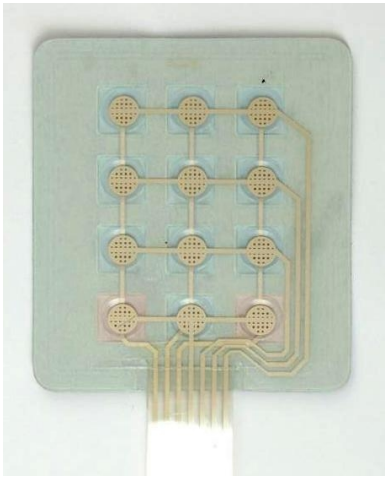
```
#include <Keypad.h>

const byte ROWS = 4; // four rows
const byte COLS = 3; // three columns
char keys[ROWS][COLS] = {
      {'1','2','3'},
      {'4','5','6'},
      {'7','8','9'},
      {'*','0','#'}
      };
byte rowPins[ROWS] = {8, 7, 6, 5};  // connect to the row pinouts of the keypad
byte colPins[COLS] = {4, 3, 2};  // connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
byte ledPin = 13;
int beep = 12;

boolean blink = false;

void setup(){
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);     // sets the digital pin as output
  pinMode(beep, OUTPUT);
```

```
  digitalWrite(ledPin, LOW);            // sets the LED off
  keypad.addEventListener(keypadEvent); // add an event listener for this keypad
}

void loop(){
  char key = keypad.getKey(); // Returns the key that is pressed, if any

  if (key) {                  // if (key != NO_KEY)
    Serial.println(key);
    digitalWrite(beep, HIGH); // set the buzzer on
    delay(100);
    digitalWrite(beep, LOW);  // set the buzzer off
  }
  if (blink){
    digitalWrite(ledPin,!digitalRead(ledPin));
    delay(100);
  }
}
// take care of some special events
void keypadEvent(KeypadEvent key){
  switch (keypad.getState()){ // Returns the current state of any of the keys
    case PRESSED:             // The four states are IDLE PRESSED RELEASED HOLD
      switch (key){
        case '#':
        case '*':
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
          digitalWrite(ledPin,!digitalRead(ledPin));
        break;
      }
    break;
    case RELEASED:
      switch (key){
        case '*':
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
          digitalWrite(ledPin,!digitalRead(ledPin));
          blink = false;
        break;
      }
    break;
    case HOLD:
      switch (key){
        case '*':
          blink = true;
        break;
      }
  break;
  }
}
```

Σχηματικό διάγραμμα:



Σχήμα 55: Συνδεσμολογία αριθμητικού πληκτρολογίου.



Σχήμα 56: Πίσω όψη συνδεσμολογίας αριθμητικού πληκτρολογίου.

Πίνακας 9: Αντιστοίχιση των θυρών του arduino με τις εισόδους του keypad και τις γραμμές και στήλες

Θύρες arduino	Είσοδοι keypad	Γραμμές/Στήλες
2	7	Column 2
3	6	Column 1
4	5	Column 0
5	4	Row 3
6	3	Row 2
7	2	Row 1
8	1	Row 0

Περιγραφή της Άσκησης

Η <keypad.h> βιβλιοθήκη αποτελείται και από συναρτήσεις που είναι non-blocking όπως η getKey() (επιστρέφει την τιμή του κουμπιού που πατήθηκε), που σημαίνει ότι μπορούμε να κρατήσουμε πατημένο οποιοδήποτε κουμπί και το arduino θα συνεχίζει να επεξεργάζεται τον υπόλοιπο κώδικα. Σε αντίθεση με τη waitForKey(), που μπλοκάρει κάθε κώδικα μέχρι να πατηθεί ένα κουμπί. Κάθε delay() όμως που χρησιμοποιούμε παίρνει τον επεξεργαστικό χρόνο μακριά από το πληκτρολόγιο. Ακόμα και μία μικρή καθυστέρηση όπως delay(250) μπορεί να κάνει το πληκτρολόγιο να μην ανταποκρίνεται. Η συνάρτηση getKey() επιστρέφει μια τιμή, τη στιγμή που πατάμε το κουμπί (PRESSED ή HOLD), αλλά δεν επαναλαμβάνεται αυτόματα. Με το που αφήνουμε το κουμπί μπορούμε να καλέσουμε τη συνθήκη RELEASED, όταν χρησιμοποιούμε την addEventListener() δυνατότητα της βιβλιοθήκης. Τέλος η getState(): επιστρέφει την τωρινή κατάσταση του κουμπιού, οι καταστάσεις είναι 4: IDLE, PRESSED, RELEASED and HOLD.

8ο Εργαστηριακό μάθημα

SD

SD CARD module (SD κάρτα επέκτασης)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός Sd κάρτας επέκτασης.

Θεωρητική εισαγωγή

Κάρτα Επέκτασης SD / Κάρτα Μνήμης SD: Οι πλατφόρμες Arduino με την περιορισμένη σε χωρητικότητα μνήμη EEPROM που διαθέτουν δεν μπορούν να χρησιμοποιηθούν σε εφαρμογές στις οποίες είναι επιθυμητή η μόνιμη αποθήκευση μεγάλου όγκου δεδομένων. Έτσι λοιπόν, εάν είναι απαραίτητο να αναπτυχθεί ένα αυτόνομο ενσωματωμένο σύστημα όπου θα πρέπει να διαθέτει μεγάλη αποθηκευτική ικανότητα, μπορεί να χρησιμοποιηθεί κάποια εξωτερική κάρτα επέκτασης για την υποστήριξη καρτών μνήμης (π.χ. SD, microSD) μεγάλης χωρητικότητας.

Η σειριακή επικοινωνία με την κάρτα μνήμης SD πραγματοποιείται μέσω της κάρτας επέκτασης SDcard Reader Module καθώς και της διεπαφής SPI την οποία υποστηρίζει η πλατφόρμα Arduino μέσω των ειδικών επαφών εισόδου και εξόδου που διαθέτει. Το σύστημα αρχείων που χρησιμοποιείται για την κάρτα μνήμης SD είναι το FAT32. Η υποστήριξη του συστήματος αρχείων FAT32 παρέχεται μέσω της βιβλιοθήκης SD.h.

Η κάρτα επέκτασης **Sd card module** διαθέτει τα απαραίτητα ηλεκτρονικά εξαρτήματα για να λειτουργήσει “plug and play” όπως: αντιστάσεις, πυκνωτές, μία μεταλλική υποδοχή για κάρτες μνήμης SD, ένα ενσωματωμένο κύκλωμα για την μετατροπή της τάσης από 5V σε 3.3V για την ορθή λειτουργία των καρτών μνήμης SD.

Στην άσκηση που ακολουθεί θα χρησιμοποιήσουμε και αναλογικούς αισθητήρες όπως η φωτοαντίσταση και το θερμίστορ. Αποτελούν ειδικές κατηγορίες αντιστατών που ανήκουν στις **μη γραμμικές αντιστάσεις:**

Η **Φωτοαντίσταση (LDR)** (light dependent resistor) (αισθητήριο μέτρησης της φωτεινής ακτινοβολίας) είναι μια μεταβλητή αντίσταση της οποίας η τιμή μεταβάλλεται αντίστροφα με την προσπίπτουσα φωτεινή ακτινοβολία δηλαδή μειώνεται με την αύξηση του φωτός που προσπίπτει στην επιφάνεια της. Αν πέσει φως στην συσκευή με αρκετά υψηλή συχνότητα, τότε φωτόνια απορροφούνται από τον ημιαγωγό και δεσμευμένα ηλεκτρόνια αποκτούν αρκετή ενέργεια, ώστε να αποσπαστούν από τα άτομα που τα δεσμεύουν. Τα ελεύθερα ηλεκτρόνια που δημιουργούνται άγουν ρεύμα και αυτό έχει σαν αποτέλεσμα την μείωση της αντίστασης. Η αρχή λειτουργίας της στηρίζεται στο φαινόμενο της φωτοαγωγιμότητας και κατασκευάζεται από ειδικά φωτοαγώγιμα υλικά όπως ενώσεις κάδμιου - σεληνίου. Το υλικό αυτό στηρίζεται πάνω σε μια μικρή μονωτική επιφάνεια και το όλο στοιχείο

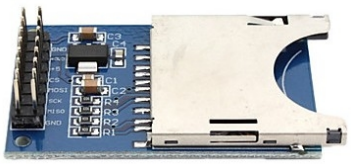
τοποθετείται μέσα σε ένα γυάλινο σωλήνα, διαφανή για το φως που προστατεύει το φωτοαγώγιμο υλικό. Οι φωτοαντιστάσεις χρησιμοποιούνται σε πολλά κυκλώματα αυτοματισμών.

Το **Θερμίστορ** (thermistor) είναι και αυτό αντίσταση, του οποίου η τιμή μεταβάλλεται με τη θερμοκρασία. Τα θερμίστορ χωρίζονται σε δύο κατηγορίες, τα θερμίστορ θετικού συντελεστή θερμοκρασίας (PTC) και τα θερμίστορ αρνητικού συντελεστή θερμοκρασίας (NTC). Στα θερμίστορ θετικού συντελεστή θερμοκρασίας η αντίσταση αυξάνεται με την αύξηση της θερμοκρασίας, ενώ στα θερμίστορ αρνητικού συντελεστή η αντίσταση μειώνεται με την αύξηση της θερμοκρασίας. Τα θερμίστορ χρησιμοποιούνται σε κυκλώματα προστασίας σε ενισχυτές, δέκτες tv, κτλ.

Το **Βαρίστορ (VDR)** προέρχεται από τα αρχικά των λέξεων Voltage Depending Resistance που σημαίνει αντίσταση εξαρτώμενη από την τάση. Σε αυτές τις αντιστάσεις όσο αυξάνεται η τάση στα άκρα της τόσο μειώνεται η ονομαστική της τιμή.

ΑΣΚΗΣΗ 8

Σκοπός της Άσκησης:



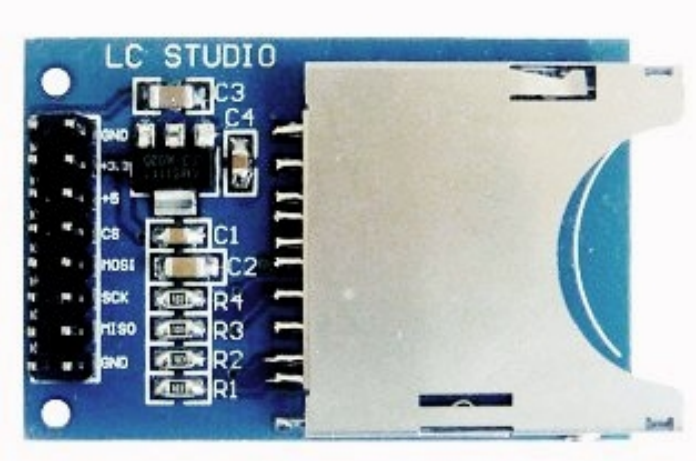
Χρήση εξωτερικής κάρτας επέκτασης SD module με εισαγωγή τυπικής κάρτας μνήμης SD card για την μόνιμη αποθήκευση των δεδομένων, που λαμβάνουμε από 3 αναλογικούς αισθητήρες: ένα ποτενσιόμετρο, έναν αισθητήρα φωτός (photoresistor) και ένα θερμίστορ (thermistor). Κάνοντας χρήση της SD βιβλιοθήκης. Ταυτόχρονα θα γίνεται εμφάνιση όλων των δεδομένων στη σειριακή οθόνη με απεικόνιση της θερμοκρασίας και της ποσότητας του φωτός.

Απαιτούμενα Υλικά:

1. Κάρτα επέκτασης SD 
2. Κάρτα μνήμης SD
3. Ποτενσιόμετρο 10 kΩ 
4. Αισθητήρας φωτός 10 kΩ 
5. Θερμίστορ NTC 10 kΩ 
6. 2x Αντιστάσεις 10 kΩ

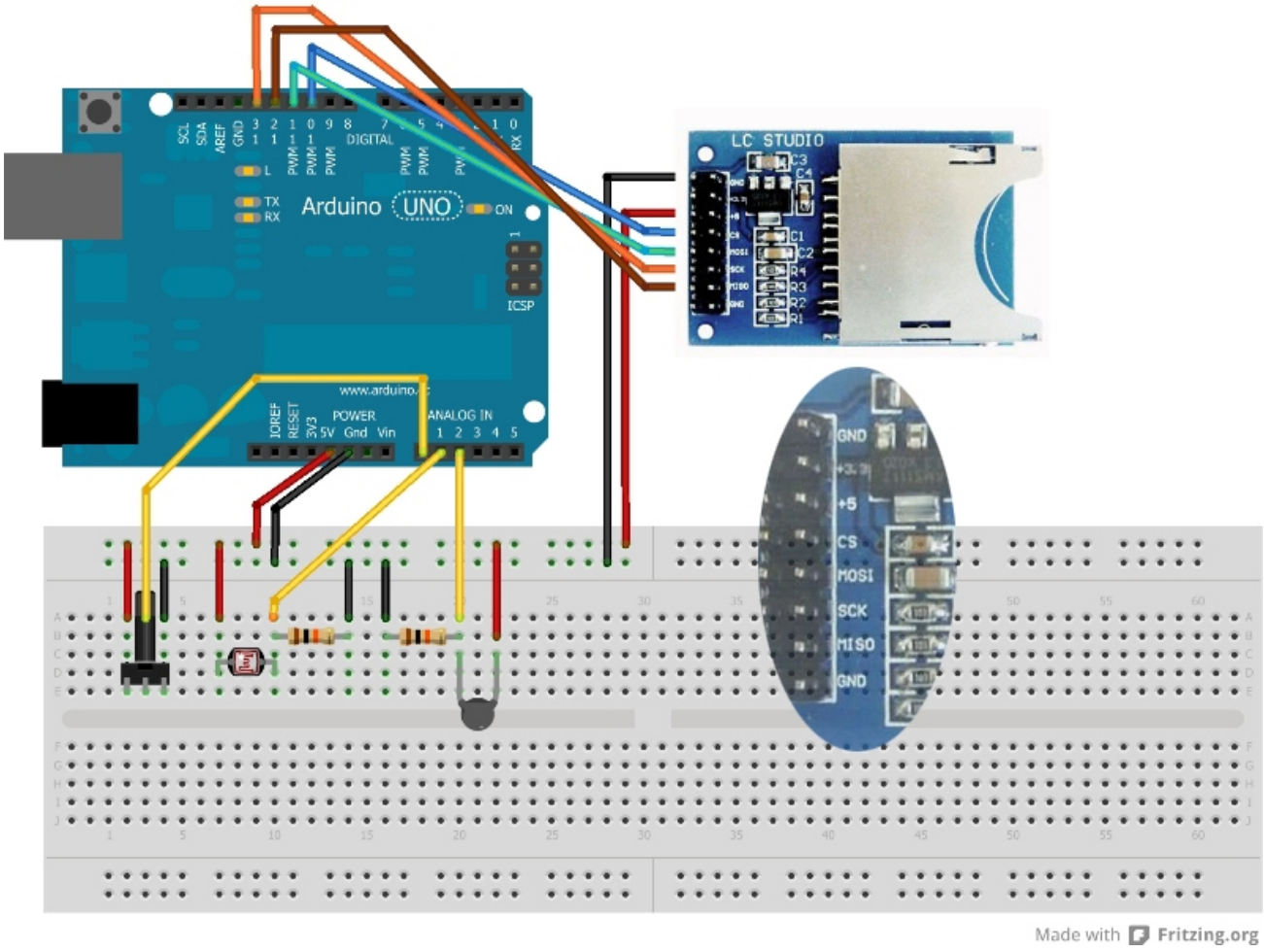
Χαρακτηριστικά λειτουργίας SD Card Socket Reader Module:

Micro SD card memory board SD card reading and writing Supports SDIO and SPI Supports 3.3V /5V power supply Plastic + iron + copper



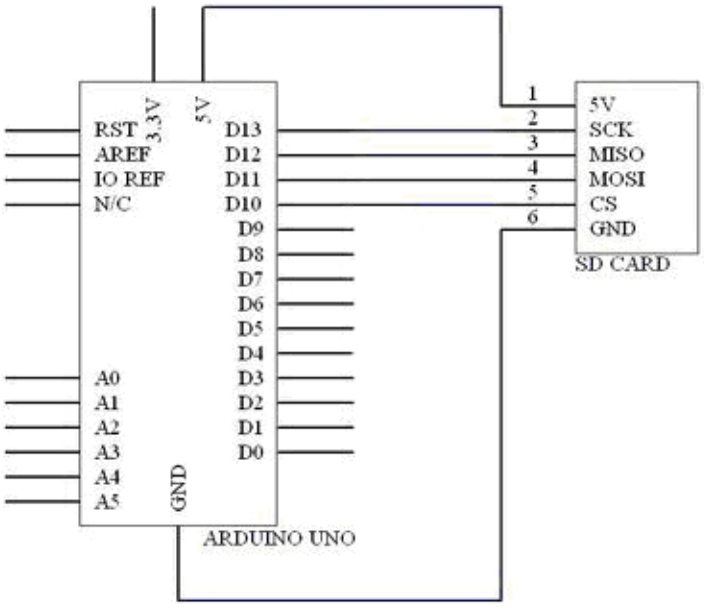
Εικόνα 22: Θύρες κάρτας επέκτασης SD (SD card module).

Συνδεσμολογία:



Σχήμα 57: Φυσική διάταξη του κυκλώματος σύνδεσης.

Σχηματικό διάγραμμα:



Σχήμα 58: Συνδεσμολογία κάρτας επέκτασης SD (SD card module).

Για την υλοποίηση του κυκλώματος συνδέουμε τους αναλογικούς αισθητήρες στις αναλογικές θύρες 0,1 και 2. Η SD κάρτα συνδέεται με τον SPI δίαυλο του μικροελεγκτή όπως στον παρακάτω πίνακα:

SD	Arduino
CS	pin 10
MOSI	pin 11
MISO	pin 12
CLK	pin 13

Πρόγραμμα 8

```
// SD card datalogger sketch.

#include <SD.h>
#include <math.h>

const int chipSelect = 10;

int potPin = 0;           // the potentiometer is connected to A0
int photocellPin = 1;     // the cell and 10K pulldown are connected to A1
int thermistorPin = 2;    // the thermistor is connected to A2

void setup()
{
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.print("Initializing SD card...");

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
```

```
  Serial.println("card initialized.");
}

void loop()
{
  // make a string for assembling the data to log:
  String dataString = "";

  // read three sensors and append to the string:
  for (int analogPin = 0; analogPin < 3; analogPin++) {
    int sensor = analogRead(analogPin);
    dataString += String(sensor);
    if (analogPin < 2) {
      dataString += ",";
    }
  }

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  File dataFile;
   dataFile = SD.open("dat.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    // print to the serial port too:
    Serial.println(dataString);
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("Error opening datalog file...");
  }

  printPot();
  printLdr();
  printTherm();
  delay(1000);
}

void printPot() {
  int potValue;     // the analog reading from the analog resistor divider
  potValue = analogRead(potPin);
  potValue = map(potValue, 0, 1023, 0, 10240);  // map(potValue, 0, 1023, 0,
5120); for a linear 5 Kohm pot
  Serial.print("Resistor is ");
  Serial.print(potValue);
  Serial.println(" Ohm");
}

void printLdr() {
  int photocellReading;     // the analog reading from the analog resistor
divider
  photocellReading = analogRead(photocellPin);

//  Serial.print("Analog reading = ");
//  Serial.print(photocellReading);     // the raw analog reading

  Serial.print("Light is ");
  // We'll have a few threshholds, qualitatively determined
  if (photocellReading < 10) {
    Serial.println("Dark");
  } else if (photocellReading < 200) {
    Serial.println("Dim");
```

```
  } else if (photocellReading < 500) {
    Serial.println("twilight");
  } else if (photocellReading < 800) {
    Serial.println("Bright");
  } else {
    Serial.println("Very bright");
  }
}

void printTherm() {

  double temp;
  temp = Thermister(analogRead(thermistorPin));  // Read sensor on pin
thermistorPin
  Serial.print("Temp is ");
  Serial.print(temp);  // display Celcius
  Serial.println(" C\n");

//  Serial.println(int(Thermister(analogRead(thermistorPin))));  // σε μια
γραμμή κώδικα το ακέραιο μόνο μέρος της θερμοκρασίας
}

double Thermister(int RawADC) {
 double Temp;
 Temp = log(((10240000/RawADC) - 10000));
 Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))*
Temp );
 Temp = Temp - 273.15;              // Convert Kelvin to Celcius
// Temp = (Temp * 9.0)/ 5.0 + 32.0; // Convert Celcius to Fahrenheit
 return Temp;
}
```

Περιγραφή της Άσκησης

Στην Sd card αποθηκεύονται μόνο οι τιμές που επιστρέφουν οι αναλογικές θύρες, τακτοποιημένες σε 3 στήλες. Στο serial monitor εμφανίζεται η αντίσταση του ποτενσιόμετρου, η θερμοκρασία σε Κελσίου ή Φαρενάιτ (από το θερμίστορ) και η ποσότητα του φωτός σε μήνυμα σκοτεινό, ημίφως, λαμπρό, πολύ φωτεινό (από τη φωτοαντίσταση), μαζί με τις τιμές που επιστρέφουν οι αναλογικές θύρες (δηλαδή από 0 έως 1023).

Ενδεικτική εμφάνιση αποτελεσμάτων από το σειριακό monitor:

```
Initializing SD card...card initialized.
645,102,434
Resistor is 6526 Ohm
Light is Dim
Temp is 18.23 C

838,531,480
Resistor is 8388 Ohm
Light is Bright
Temp is 22.26 C
```

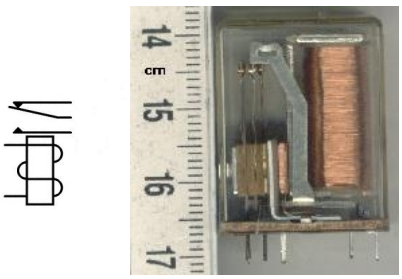
9ο Εργαστηριακό μάθημα

RELAY (Ηλεκτρονόμος, ρελέ)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός ηλεκτρονόμου.

Θεωρητική εισαγωγή

Ο ηλεκτρονόμος, ρελέ (relay) είναι ένας ηλεκτρομηχανικός διακόπτης που είτε ανοίγει, είτε κλείνει ένα ηλεκτρικό κύκλωμα κάτω από τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος. Ιστορικά στην αρχική μορφή του, ένας ηλεκτρομαγνήτης ενεργοποιούσε το διακόπτη, με το άνοιγμα ή το κλείσιμο μιας ή περισσότερων επαφών. Εφευρέθηκε από τον Τζόζεφ Χένρυ το 1835. Επειδή ένας ηλεκτρονόμος είναι ικανός να ελέγχει ένα κύκλωμα εξόδου υψηλότερης ισχύος από το κύκλωμα εισόδου, μπορεί να θεωρηθεί γενικά, μια μορφή ηλεκτρικού ενισχυτή. Σήμερα πέρα από τα ηλεκτρομαγνητικά ρελέ υπάρχουν και τα ηλεκτρονικά.



Εικόνα 23: Το σύμβολο του ρελέ στο λογικό διάγραμμα και μια εικόνα ενός μικρού ηλεκτρονόμου.

Λειτουργία ρελέ

Το ρελέ διαρρέεται από ρεύμα και το μαγνητικό πεδίο που δημιουργείται από το πηνίο του, έλκει τον οπλισμό ο οποίος είναι συνδεδεμένος με μια επαφή, με αποτέλεσμα -η κινούμενη αυτή επαφή- να συνδέεται και να αποσυνδέεται από μια σταθερή επαφή. Αν διακοπεί το ρεύμα τότε υπάρχει ένα ελατήριο το οποίο επαναφέρει τον οπλισμό. Για να λειτουργήσει το πηνίο και να μετακινηθούν οι επαφές χρειαζόμαστε μεγάλη ένταση ηλεκτρικού ρεύματος και όταν κλείσει ο οπλισμός τότε μας αρκεί και το 1/10 της αρχικής.

Οι επαφές του ρελέ μπορεί να είναι τριών ειδών: Κανονικά-Ανοικτή, Κανονικά-Κλειστή και Μεταγωγική.

- Μια επαφή **Κανονικά-Ανοικτή** (NO) συνδέει το κύκλωμα όταν ο ηλεκτρονόμος ενεργοποιείται και το κύκλωμα αποσυνδέεται όταν ο ηλεκτρονόμος είναι ανενεργός.
- Μια επαφή **Κανονικά-Κλειστή** (NC) αποσυνδέει το κύκλωμα όταν ο ηλεκτρονόμος ενεργοποιείται και το κύκλωμα συνδέεται όταν ο ηλεκτρονόμος είναι ανενεργός.
- Μια επαφή **Μεταγωγική** (COM) μπορεί να ελέγχει δύο κυκλώματα. Ισοδυναμεί με μια επαφή κανονικά-ανοικτή και μια επαφή κανονικά-κλειστή που έχουν ένα κοινό ακροδέκτη.

Οι επαφές σε έναν ρελέ χωρίζονται σε **κύριες** και **βοηθητικές**. Οι κύριες διαρρέονται από ισχυρότερα ρεύματα και έτσι είναι αυτές που διακόπτουν το κύριο κύκλωμα και συνήθως είναι Κανονικά-Ανοικτές. Οι βοηθητικές έχουν επικουρικό χαρακτήρα και ο ρόλος τους είναι να βοηθούν στον έλεγχο των αυτοματισμών (που είναι ο κύριος τομέας χρήσης των ηλεκτρονόμων). Για παράδειγμα βοηθούν στην ενεργοποίηση και απενεργοποίηση βοηθητικών κυκλωμάτων, όπως είναι οι ενδεικτικές λυχνίες.

ΑΣΚΗΣΗ 9

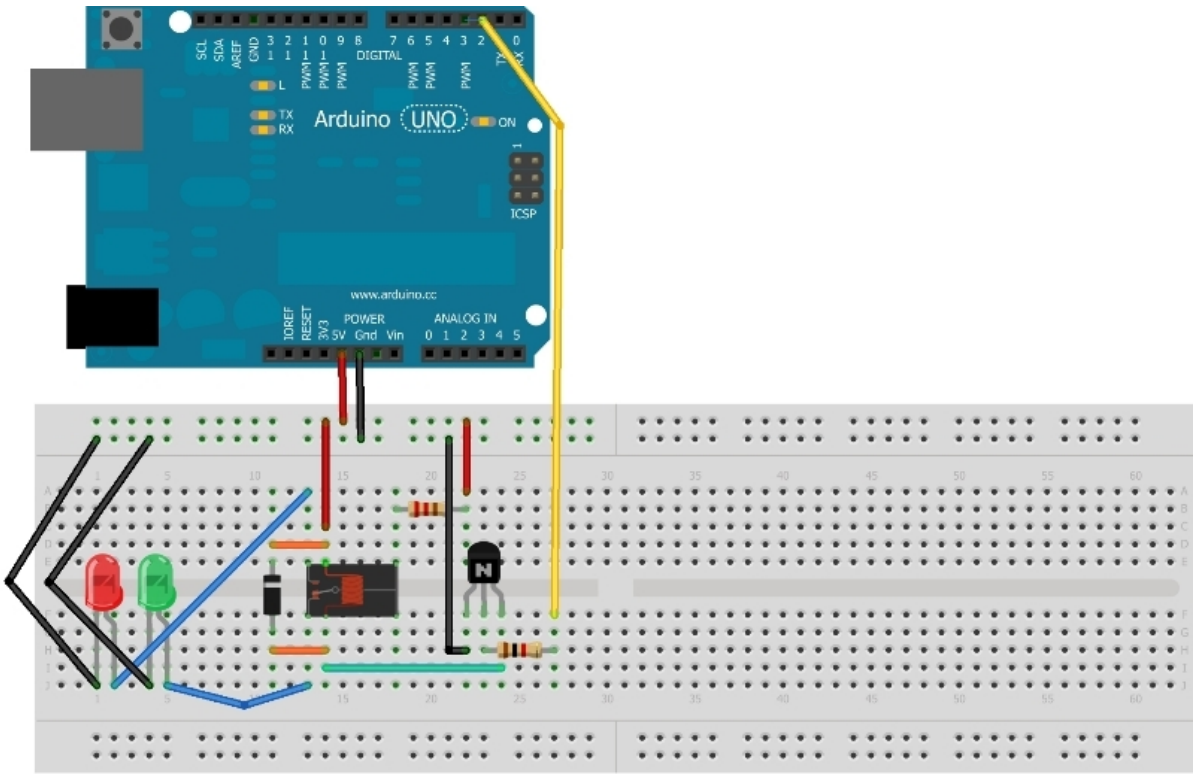
Σκοπός της Άσκησης:

Οδήγηση ηλεκτρονόμου με τη βοήθεια τρανζίστορ και έξοδο με εναλλαγή τροφοδοσίας σε 2 φωτοδιόδους.

Απαιτούμενα Υλικά:

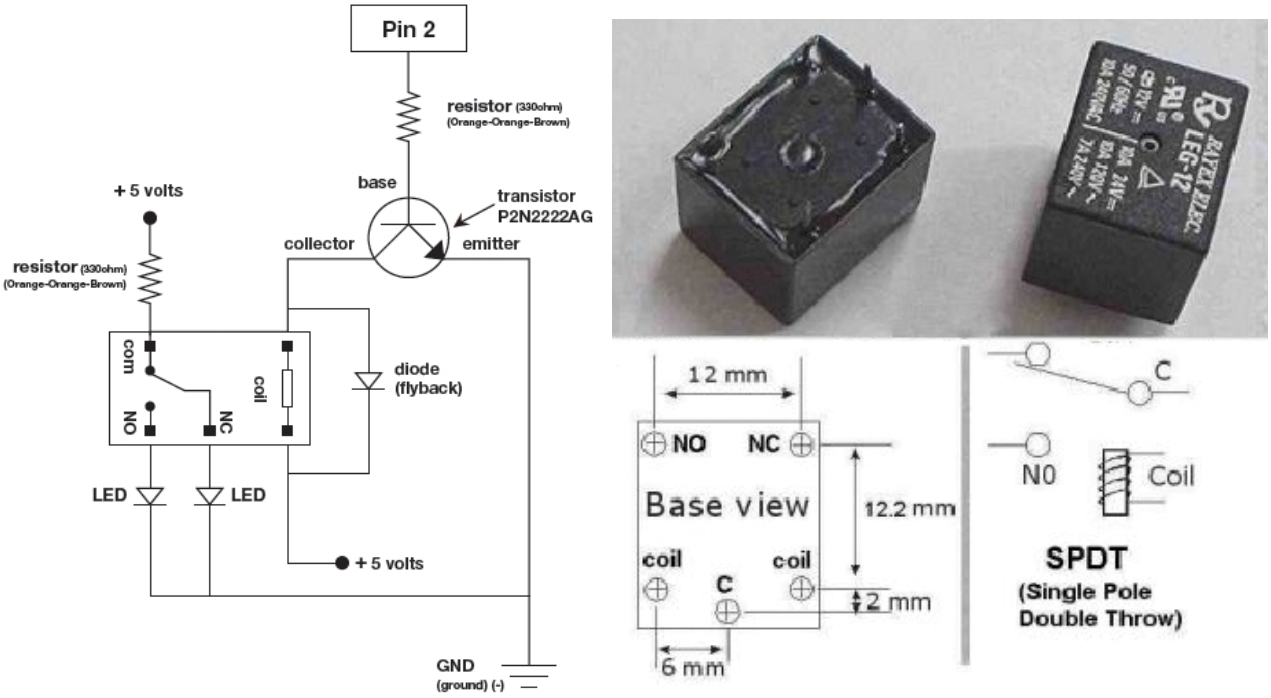
1. Ηλεκτρονόμος 5V
2. Τρανζίστορ P2N2222A
3. Δίοδος 1N4148
4. LED 5mm 
5. 2x 330Ω αντιστάσεις 

Συνδεσμολογία:

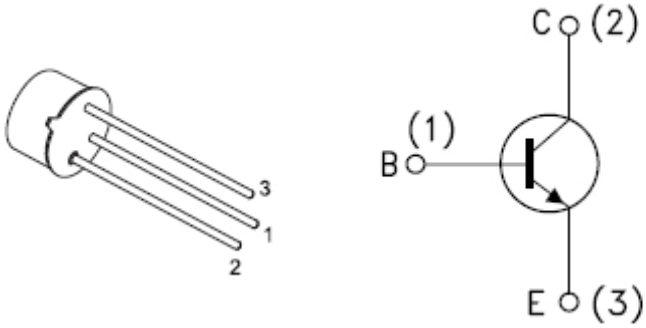


Σχήμα 59: Φυσική διάταξη του κυκλώματος σύνδεσης ρελέ με χρήση τρανζίστορ.

Σχηματικό διάγραμμα:

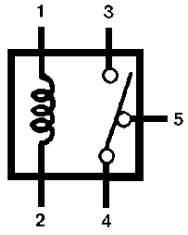


Σχήμα 60: Συνδεσμολογία οδήγησης ηλεκτρονόμου με χρήση τρανζίστορ.



Σχήμα 61: Σχηματικό διάγραμμα τρανζίστορ P2N2222A.

Όταν το ρελέ είναι κλειστό, το COMMON pin είναι συνδεδεμένο με το NC pin και το αντίστροφο, όταν το COMMON pin είναι συνδεδεμένο με το NO pin το ρελέ λειτουργεί.



1,2: coil (πηνίο)
3: NO (Normally Open)
4: NC (Normally Closed)
5: COM (Common)

Πρόγραμμα 9

```
const int relayPin = 2;       // use this pin to drive the transistor
const int timeDelay = 1000;   // delay in ms for on and off phases

// You can make timeDelay shorter, but note that relays, being
// mechanical devices, will wear out quickly if you try to drive
// them too fast.

void setup()
{
  pinMode(relayPin, OUTPUT);  // set pin as an output
}

void loop()
{
  digitalWrite(relayPin, HIGH);  // turn the relay on
  delay(timeDelay);              // wait for one second

  digitalWrite(relayPin, LOW);   // turn the relay off

  delay(timeDelay);              // wait for one second
}
```

Περιγραφή της Άσκησης

Με τη βοήθεια του τρανζίστορ οδηγήσαμε ένα ρελέ, το οποίο δίνει έξοδο σε 2 φωτοδιόδους. Στην κατασκευή μας χρησιμοποιήσαμε τρανζίστορ τύπου PNP το οποίο έχει 3 ακροδέκτες. Ο ακροδέκτης που έχει το βελάκι είναι ο **εκπομπός** (emitter), ο μεσαίος ακροδέκτης είναι η **βάση** (base) και ο τρίτος ακροδέκτης είναι ο **συλλέκτης** (collector). Συνδέσαμε τη βάση του τρανζίστορ μέσω μιας αντίστασης 330Ω στη ψηφιακή θύρα 2 και τον εκπομπό στη γείωση.

Επίσης χρησιμοποιήσαμε μια δίοδο για την προστασία του τρανζίστορ, λόγω των απότομων μεταβολών και διακυμάνσεων της τάσης που δημιουργούνται κατά τη λειτουργία του ρελέ, οι οποίες μπορεί να είναι καταστροφικές για το arduino και το κύκλωμά μας. Συνδέσαμε τη μία πλευρά της διόδου στην τάση 5V του arduino και την άλλη στον ακροδέκτη (συλλέκτη) του τρανζίστορ, κυκλώνοντας έτσι το πηνίο.

Μελετώντας το πρόγραμμα που φορτώσαμε στο Arduino παρατηρούμε ότι:
Όταν ενεργοποιούμε το τρανζίστορ (εκείνη τη στιγμή ο διακόπτης του ρελέ είναι κλειστός, NC), αυτό με τη σειρά του ενεργοποιεί το πηνίο του ρελέ και ανοίγει το διακόπτη (η COM θύρα συνδέεται με τη NO θύρα). Οτιδήποτε έχουμε συνδεδεμένο στην NO θύρα θα λειτουργήσει, όπως το πράσινο λεντάκι μας.

digitalWrite(relayPin, HIGH);

Ενώ όταν το ρελέ είναι κλειστό, η NC θύρα είναι συνδεδεμένη με την COM θύρα. Μπορούμε να χρησιμοποιήσουμε οποιαδήποτε θύρα θέλουμε ανάλογα με το αν θέλουμε HIGH ή LOW στην έξοδο μας. Μπορούμε επιπλέον να χρησιμοποιήσουμε και τις 2 θύρες NO και NC αν θέλουμε να εναλλάξουμε τροφοδοσία σε 2 συσκευές. Χαρακτηριστικό παράδειγμα τα προειδοποιητικά φανάρια σε σιδηροδρομικές γραμμές.

digitalWrite(relayPin, LOW);

Παρατηρήσεις - Συμπεράσματα

Το ρελέ παίρνει το ρόλο του διακόπτη, που αντί να τον «ανοίξουμε» με το χέρι, χρησιμοποιούμε ένα χαμηλής έντασης ηλεκτρικό σήμα. Έτσι όταν το ηλεκτρικό ρεύμα διαρρέει το πηνίο του ηλεκτρονόμου, το παραγόμενο μαγνητικό πεδίο έλκει τον οπλισμό που είναι μηχανικά συνδεδεμένος στην κινούμενη επαφή.

Το ηλεκτρομηχανικό ρελέ διαθέτει μία ή περισσότερες επαφές όπου κάθε μία από αυτές μπορεί να είναι είτε “Κανονικά Ανοικτή” (Normally Open, NO), είτε “Κανονικά Κλειστή” (Normally Closed, NC). Η επαφή “Κανονικά Ανοικτή” συνδέει το κύκλωμα όταν το ρελέ είναι ενεργοποιημένο και αποσυνδέει το κύκλωμα όταν είναι απενεργοποιημένο. Ενώ η επαφή “Κανονικά Κλειστή” συνδέει το κύκλωμα όταν το ρελέ είναι απενεργοποιημένο και αποσυνδέει το κύκλωμα όταν είναι ενεργοποιημένο. Εκτελώντας την παραπάνω κατασκευή μπορούμε να ακούσουμε τις επαφές του ρελέ να κλικάρουν και να δούμε τα 2 led να φωτίζονται εναλλάξ με μεσοδιάστημα 1 δευτερόλεπτου.

10ο Εργαστηριακό μάθημα

Shift Register 74HC595 (Καταχωρητής ολίσθησης)

ΑΝΤΙΚΕΙΜΕΝΟ : Η κατανόηση της αρχής λειτουργίας, η σύνδεση και ο προγραμματισμός του καταχωρητή ολίσθησης 74HC595.

Θεωρητική εισαγωγή

Ένας **καταχωρητής** (register) είναι κύκλωμα που χρησιμοποιείται για την αποθήκευση πληροφοριών. Είναι συνήθως μικρή σε μέγεθος αλλά μεγάλης ταχύτητας αποθηκευτική μονάδα που βρίσκεται μέσα σε μικροεπεξεργαστή.

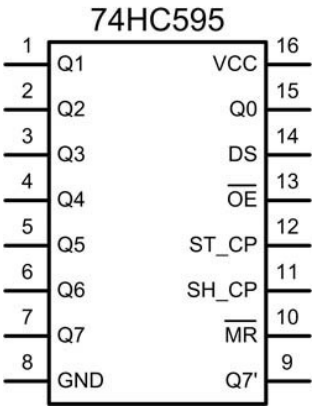
Αποτελείται από ομάδα δυαδικών κυττάρων αποθήκευσης flip-flops (FFs) (κατάλληλα για να κρατάνε δυαδικές πληροφορίες) και από λογικές πύλες που βοηθάνε στη διεκπεραίωση της μεταφοράς των πληροφοριών από και προς τον καταχωρητή, αλλά και στο να εκτελούν διάφορες λειτουργίες επεξεργασίας δεδομένων. Ο απλούστερος δυνατός τύπος καταχωρητή αποτελείται μονάχα από FFs χωρίς εξωτερικές πύλες. Δηλαδή μια ομάδα FFs αποτελεί έναν καταχωρητή, αφού κάθε FF μπορεί να αποθηκεύσει ένα (1) bit πληροφορίας. Επομένως, αν χρησιμοποιηθούν n-FFs μπορούν να αποθηκευτούν n-bit λέξεις. Ένας καταχωρητής των n-bit περιέχει n-FFs και μπορεί να αποθηκεύσει n-bits οποιαδήποτε δυαδικής πληροφορίας.

Η μεταφορά της πληροφορίας από τις εισόδους στις εξόδους του καταχωρητή ονομάζεται **φόρτωση** (loading). Παράλληλη αν γίνεται σε όλα τα bits μαζί. Σειριακή αν γίνεται σε ένα - ένα cell χωριστά.

Ο **καταχωρητής ολίσθησης** (shift register) είναι ένας καταχωρητής, όπου η έξοδος του κάθε FF τροφοδοτεί την είσοδο του γειτονικού του. Έτσι, τα δεδομένα του καταχωρητή ολισθαίνουν (μετακινούνται) κατά μία θέση, από κάθε FF στο γειτονικό του, με κάθε παλμό ρολογιού. Ο καταχωρητής ολίσθησης μπορεί και ολισθαίνει τα περιεχόμενά του και προς τις δύο κατευθύνσεις. Τα δεδομένα μπορούν να φορτωθούν σειριακά. Ένα ψηφιακό σύστημα λειτουργεί σειριακά, όταν οι πληροφορίες μεταφέρονται και επεξεργάζονται το ένα bit μετά το άλλο σε διαδοχικούς χρόνους.

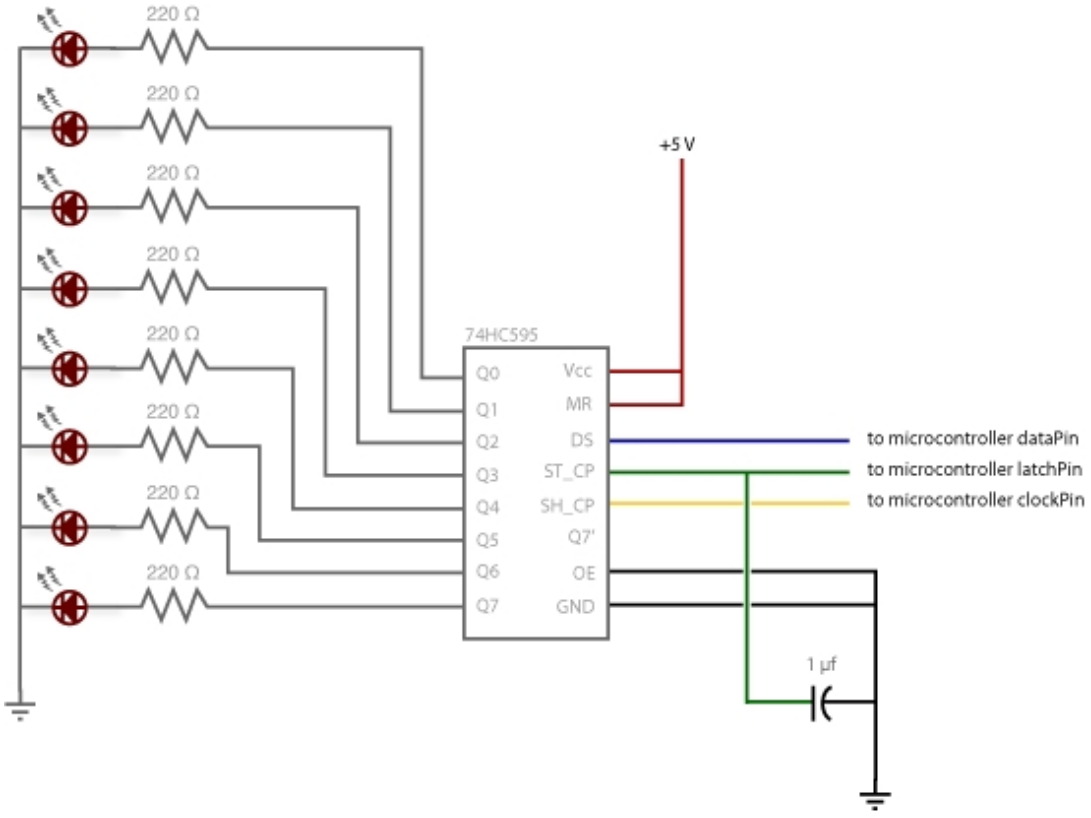
74HC595 Shift Register: Πρόκειται για απλούς και εύχρηστους 8-bit καταχωρητές ολισθητές. Χρησιμοποιούνται για την παραλληλοποίηση σειριακών δεδομένων. Έχουν μία είσοδο για σειριακά δεδομένα (ds) και 8 εξόδους (από Q0 έως Q7) για την παράλληλη εξόρυξη τους. Ακόμα έχουν μία επιπλέον σειριακή έξοδο, την q7 η οποία χρησιμοποιείται για την σύνδεση περισσοτέρων καταχωρητών ολίσθησης σε σειρά.

Πίνακας 10: Θύρες ολοκληρωμένου κυκλώματος 74HC595



PINS 1-7, 15	Qo " Q7	Output Pins
PIN 8	GND	Ground, Vss
PIN 9	Q7"	Serial Out
PIN 10	MR	Master Reclear, active low
PIN 11	SH_CP	Shift register clock pin
PIN 12	ST_CP	Storage register clock pin (latch pin)
PIN 13	OE	Output enable, active low
PIN 14	DS	Serial data input
PIN 16	Vcc	Positive supply voltage

Διαθέτουν δύο ακροδέκτες ωρολογιακών εισόδων (clock inputs). Τον SH_CP και τον ST_CP. Σε κάθε ανοδικό παλμό του ακροδέκτη SH_CP, swift clock, μετατοπίζονται σειριακά τα δεδομένα στους καταχωρητές. Σε κάθε ανοδικό παλμό του ακροδέκτη ST_CP, storage clock, μεταφέρονται στις παράλληλες εξόδους του ολοκληρωμένου κυκλώματος οι τιμές των καταχωρητών. Παρακάτω δίνονται σε μορφή πίνακα οι κυριότεροι ακροδέκτες του και μία σύντομη περιγραφή της λειτουργίας τους.



Σχήμα 62: Συνδεσμολογία οδήγησης 8 led με χρήση καταχωρητή ολίσθησης 74HC595.

Κυριότεροι ακροδέκτες του 74HC595 με σύντομη περιγραφή λειτουργίας τους:

- **DS :** Ονομάζεται Data Serial. Εισάγει σειριακά τα δεδομένα που οι καταχωρητές ολίσθησης θα βγάλουν παράλληλα στις εξόδους τους.
- **OE :** Ονομάζεται Output Enable. όταν έχει την τιμή «0» οι έξοδοι του καταχωρητή ολίσθησης είναι ενεργοί.
- **ST_CP :** Ονομάζεται Storage Clock Pin. Μετάβαση τάσης από λογικό «0» σε «1» έχει ως αποτέλεσμα την μεταφορά των τιμών των καταχωρητών στις εξόδους του ολοκληρωμένου κυκλώματος.
- **SH_CP :** Ονομάζεται Shift Clock Pin. Μετάβαση τάσης από λογικό «0» σε «1» έχει ως αποτέλεσμα την μετατόπιση των σειριακών τιμών στους καταχωρητές.
- **MR :** Ονομάζεται Master Reset. όταν πάρει την λογική τιμή «0» μηδενίζει τα περιεχόμενα των καταχωρητών
- **Qx :** Παράλληλη έξοδος δεδομένων που είναι συνδεμένα στον x καταχωρητή
- **Q7' :** Ονομάζεται Data Serial Out. Εξάγει σειριακά τα δεδομένα που έχουν «περάσει» διαδοχικά από όλους τους καταχωρητές.

Λοιποί ακροδέκτες:

- **1-7 και 15 :** Σε κάθε έξοδο του καταχωρητή συνδέεται μία αντίσταση σε σειρά με ένα LED. Τα LEDs ελέγχονται με αρνητική λογική, δηλαδή η κάθοδος τους είναι συνδεδεμένη με την έξοδο του καταχωρητή και η άνοδος τους είναι πάντα HIGH. έτσι για να κάνουμε ένα LED On , πρέπει στην αντίστοιχη έξοδο του καταχωρητή ολίσθησης να βγάλουμε λογικό '0'. Συνδέονται με τις αντιστάσεις, οι όποιες είναι σε σειρά με τα leds.
- **8, 13 :** Γείωση (Gnd)
- **9 :** μέσω αυτού γίνεται η σύνδεση πολλαπλών καταχωρητών. Συνδέεται στον ακροδέκτη 14 του επόμενου καταχωρητή. Στον ακροδέκτη 9 του τελευταίου καταχωρητή δεν συνδέεται τίποτα.
- **10, 16 :** Τάση Vcc

ΑΣΚΗΣΗ 10

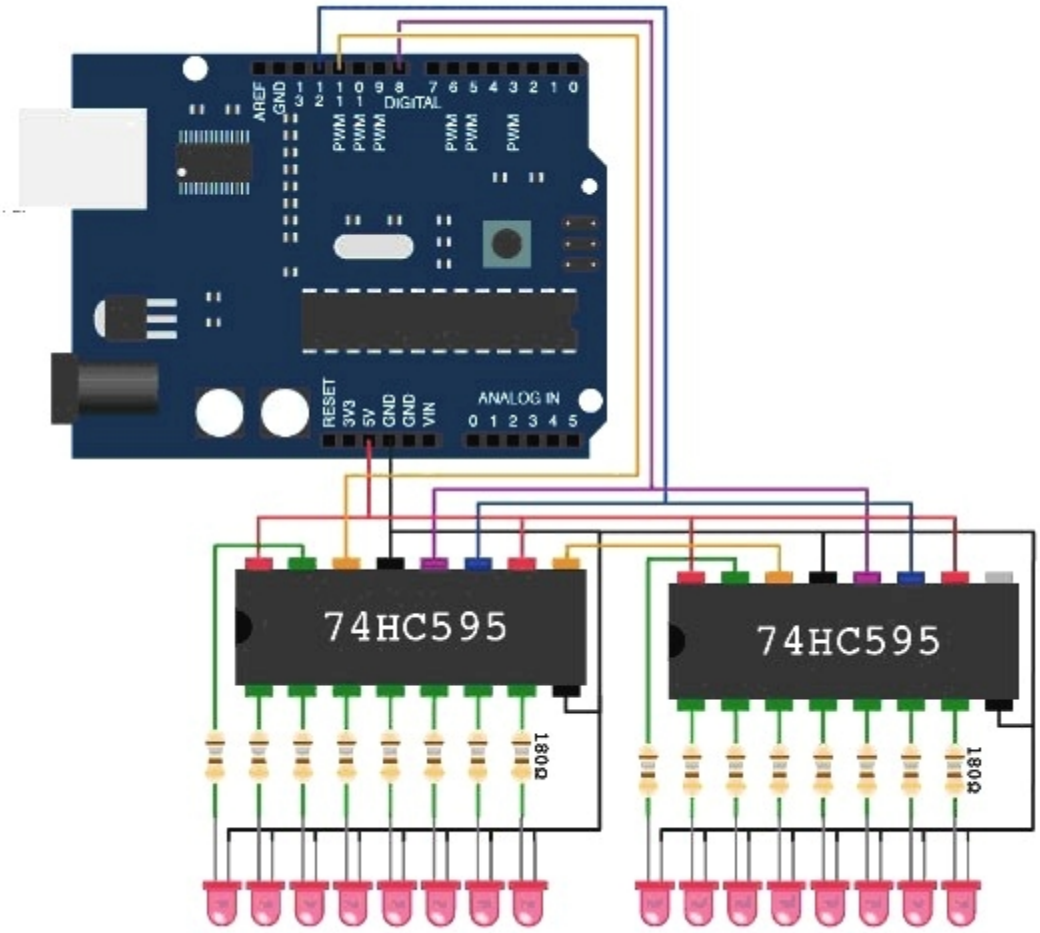
Σκοπός της Άσκησης:

Σύνδεση 16 led με χρήση 2 καταχωρητών ολίσθησης 74HC595 και ρυθμικό αναβόσβημά τους.

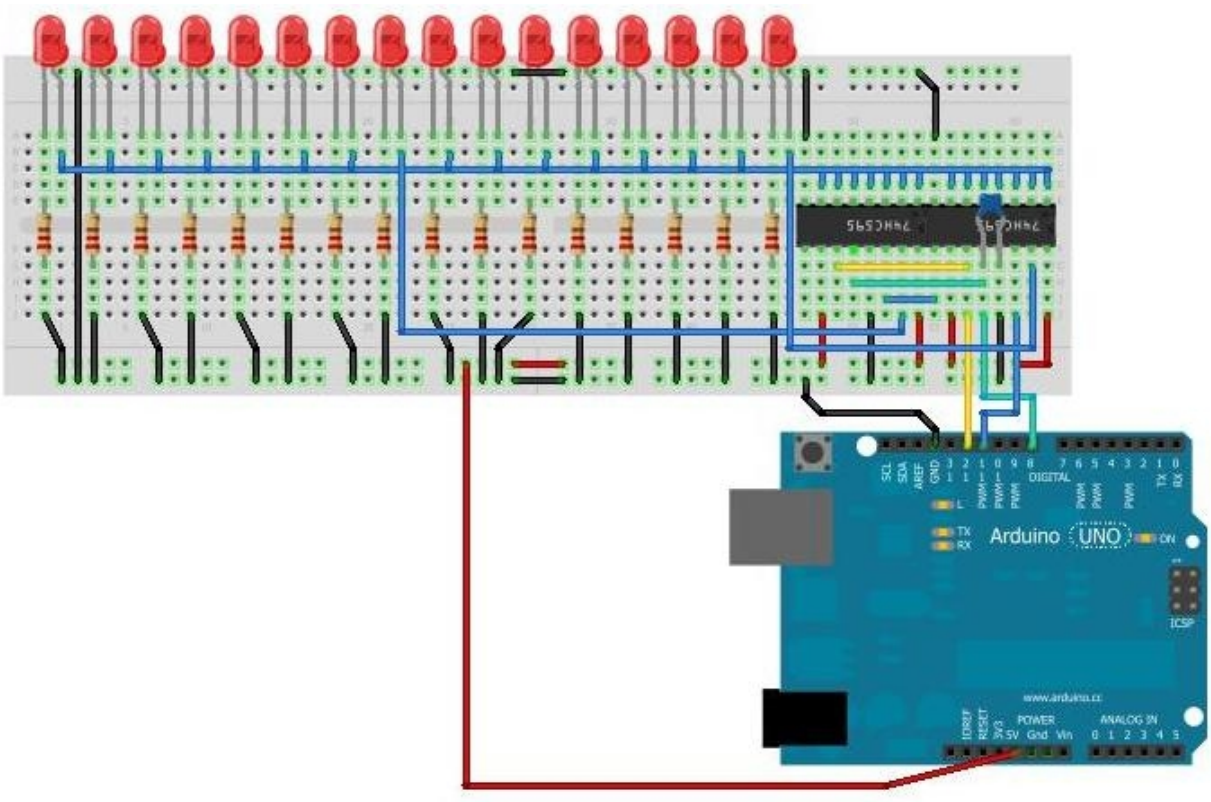
Απαιτούμενα υλικά:

1. 2x Καταχωρητής ολίσθησης 74HC595 
2. 16x Αντιστάσεις 470 Ω 
3. 16x Red LEDs 
4. Κεραμικός πυκνωτής 1μF 

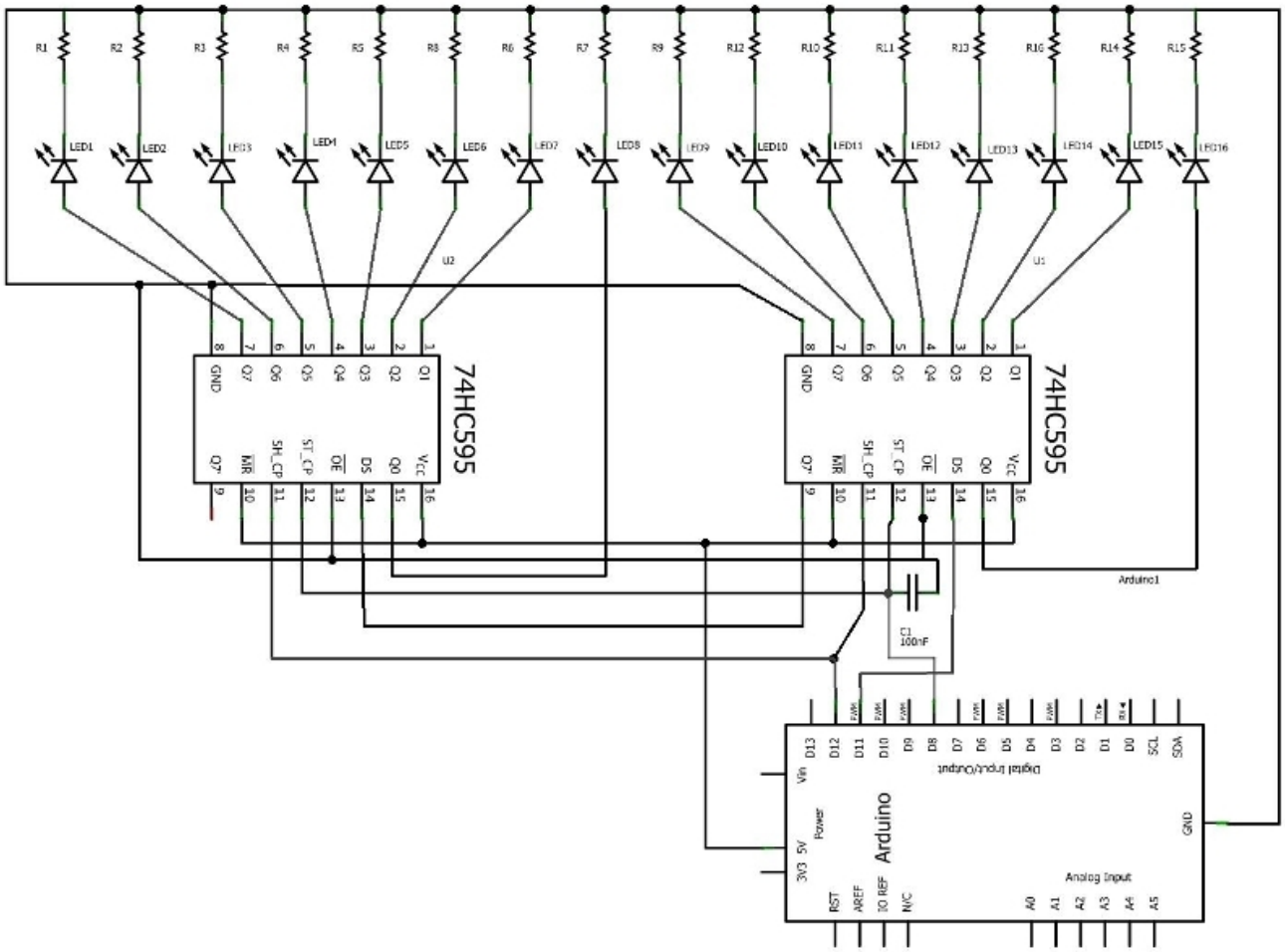
Συνδεσμολογία:



Σχήμα 63: Φυσική διάταξη του κυκλώματος 2 καταχωρητών ολίσθησης 74HC595.



Σχήμα 64: Απεικόνιση σε διάγραμμα Fritzing του κυκλώματος 2 καταχωρητών ολίσθησης 74HC595.



Σχήμα 65: Συνδεσμολογία του κυκλώματος 2 καταχωρητών ολίσθησης 74HC595.

Πρόγραμμα 10

```
int latchPin = 8;              // The Latch Pin to the Shift Register
int dataPin = 11;              // The Serial Data Pin to the Shift Register
int clockPin = 12;             // The Clock Pin to the Shift Register
int value = 0;
int value2 = 0;
int seq[31] =
{0,0,0,0,0,0,0,0,1,2,4,8,16,32,64,128,64,32,16,8,4,2,1,0,0,0,0,0,0,0,0};
int seq2[31]=
{0,0,0,0,0,0,0,0,128,64,32,16,8,4,2,1,2,4,8,16,32,64,128,0,0,0,0,0,0,0,0};

void setup()
{
      pinMode(dataPin, OUTPUT);     // Configure Digital Pins
      pinMode(latchPin, OUTPUT);
      pinMode(clockPin, OUTPUT);
}

void loop()
{
      for (int n=0; n < 31; n++)
      {
              value2 = seq[n];
              value = seq2[n];
```

```
                writeOutput();
                delay(75);
        }
}

void writeOutput()
{
        digitalWrite(latchPin, LOW);                    // Pull latch LOW to send data
        shiftOut(dataPin, clockPin, MSBFIRST, value);       // Send the data byte 1
        shiftOut(dataPin, clockPin, MSBFIRST, value2);      // Send the data byte 2
        digitalWrite(latchPin, HIGH);       // Pull latch HIGH to stop sending data
}
```

Παρατηρήσεις

Στο παραπάνω πρόγραμμα χρησιμοποιούμε 3 ψηφιακές θύρες εξόδου για τον έλεγχο 16 ψηφιακών εξόδων σημάτων απο δύο καταχωρητές. Οχτώ led συνδέθηκαν στις εξόδους κάθε καταχωρητή. Τα συνόλικα σε πλήθος 16 led αναβοσβήνουν ρυθμικά σε στύλ “Knight Rider”.

Χρησιμοποιούμε 2 Shift Registers σε σειρά για να οδηγήσουμε τα 16 LEDs. Το Arduino UNO έχει πεπερασμένο πλήθος θυρών I/O Ports (14 ψηφιακές και 6 αναλογικές). Όταν η πολυπλοκότητα της κατασκευής μας απαιτεί παραπάνω θύρες, ένας από τους τρόπους που μπορεί αυτό να επιτευχτεί είναι με χρήση καταχωρητή ολίσθησης. Άλλοι τρόποι είναι χρησιμοποιώντας τις αναλογικές θύρες ως ψηφιακές ή με πολύπλεξη των εξόδων που έχει όμως ένα βασικό μειονέκτημα, δε μπορούν να οδηγηθούν όλα τα LEDs ταυτόχρονα.

Με χρήση Shift Register στέλνονται τα δεδομένα σειριακά, μόλις περάσουν και τα 16 bits τα οποία καθορίζουν αν κάθε LED θα είναι On ή Off, μέσω της διαδικασίας Latch αυτές οι τιμές των καταχωρητών περνάνε στην έξοδο. Έτσι μπορούμε να ελέγξουμε 16 LEDs χρησιμοποιώντας μόνο τρεις εξόδους του μικροελεγκτή.

Υπάρχουν δυο τρόποι σύνδεσης των Shift Registers με το Arduino. Μπορούμε να τους συνδέσουμε με όποια ψηφιακή θύρα θέλουμε και να ελέγξουμε τη λειτουργία τους από το λογισμικό. Ο ATmega όμως προσφέρει και τη δυνατότητα ο έλεγχος να γίνεται και με λειτουργία SPI (Serial Peripheral Interface). Έτσι ο ATmega συμπεριφέρεται σαν master και οι Shift Registers σαν slaves, που συγχρονίζονται από το ρολόι του ATmega, με αυτόν τον τρόπο κερδίζουμε σε ταχύτητα πολλούς κύκλους μηχανής από ότι αν τους ελέγχαμε μόνο μέσω λογισμικού.

ΠΑΡΑΤΗΡΗΣΕΙΣ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Οι δυσκολίες που συναντήσαμε με το ξεκίνημα της εργασίας ήταν πρώτα απ' όλα η μελέτη και η επιλογή: των κατάλληλων περιφερειακών εξαρτημάτων, μέσα στην πληθώρα συμβατού υλικού (open hardware), καθώς και, η επιλογή των κατάλληλων ασκήσεων, μέσα στην αχανή βιβλιογραφία για το arduino -που καθημερινά συνεχίζει να εμπλουτίζεται- τέτοιες, ώστε το σύνολο από παραδείγματα που θα υλοποιούντο, να αξιοποιούν και να επιδεικνύουν τις δυνατότητες της πλατφόρμας.

Επιπρόσθετα το εγχείρημα απαιτούσε να αναλάβουμε την προμήθεια του απαραίτητου εξοπλισμού όπως: πλατφόρμα Arduino, διάφορα ηλεκτρονικά εξαρτήματα, καλώδια, κινητήρες κ.λ.π. Αφού πραγματοποιήσαμε διεξοδική ανάλυση, καταλήξαμε στο υλικό που θα χρησιμοποιήσουμε. Έπειτα από ενδελεχή έρευνα αγοράς στην Ελλάδα και το εξωτερικό αποφασίσαμε το hardware της πτυχιακής να αγοραστεί από 3 ηλεκτρονικά καταστήματα (e-shop) που εδρεύουν Κίνα (Χονγκ-Κονγκ), Αίγυπτο, Αγγλία καθώς και από ένα φυσικό κατάστημα ηλεκτρονικών της γειτονιάς. Ο χρόνος αναμονής έλευσης των υλικών ήτανε μία από τις παραμέτρους που έδρασαν αρνητικά στη ομαλή διεξαγωγή της εργασίας.

Επιπλέον, η υλοποίηση του πειραματικού μέρους απαιτούσε ικανότητες κατασκευής και διασύνδεσης ηλεκτρονικών κυκλωμάτων (δεξιότητες οι οποίες είχαν αποκτηθεί στο ΤΕΙ). Επίσης, ήταν πολύτιμη η καλή γνώση αγγλικών και ιδιαίτερα τεχνικής ορολογίας για τη μετάφραση των εγχειριδίων χρήσης. Τέλος, για την πτυχιακή εργαστήκαμε 2 άτομα χωρίς όμως να παρουσιαστούν προβλήματα συνεργασίας.

ΕΥΡΕΤΗΡΙΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

[1] Michael Margolis, Arduino Cookbook 2nd Edition, O'Reilly Media 2012, ISBN: 978-1-449-31387-6
[2] Michael McRoberts, Beginning Arduino, Apress 2010, ISBN: 978-1-4302-3241-4
[3] M. McRoberts, A Complete Beginners Guide to the Arduino, Earthshine Design 2009
[4] SparkFun Inventor's Guide, SparkFun Inc. 2012
[5] Joshua Noble, Programming Interactivity (SE), Oreilly 2009, ISBN: 978-1-449-31144-5
[6] Kimmo and Tero Karvinen, Make: Arduino Bots and Gadgets, O'Reilly Media 2011, ISBN: 978-1-449-38971-0
[7] Massimo Banzi, Getting Started with Arduino 2nd Edition, O'Reilly 2011, ISBN: 978-1-449-30987-9
[8] Brian Evans, Beginning Arduino Programming, Apress 2011, ISBN: 978-1-4302-3777-8
[9] Alan Smith, Introduction to Arduino A Piece of Cake, ISBN: 978-1463698348
[10] Donald Wilcher, Learn Electronics with Arduino, Apress Jul 2012, ISBN-13: 978-1-4302-4267-3
[11] Charles Platt, Make Electronics (Learning by Discovery), O'Reilly Media 2009, ISBN: 978-0-596-15374-8
[12] Emily Gertz, Environmental Monitoring Arduino, O'Reilly Media 2012, ISBN: 978-1-449-31056-1
[13] John-David Warren and Josh Adams, Arduino Robotics, Apress 2011, ISBN: 978-1-4302-3184-4
[14] Maik Schmidt, Arduino A Quick-Start Guide, Pragmatic Programmers 2011, ISBN-10: 1-934356-66-2
[15] Μ.Στεφανιδάκης, Διασύνδεση Μικροϋπολογιστικών Συστημάτων 2004-05

Πτυχιακές εργασίες:

[1] Ευστάθιος Χατζηκυριακίδης, Υλοποίηση δικτύου ασύρματης ραδιοεπικοινωνίας μεταξύ δύο ενσωματωμένων κόμβων, Σέρρες 2011
[2] Σιώπης Βασίλειος, Σχεδιασμός και κατασκευή μοντέλου εκπαιδευτικής ρομποτικής με τον μικροελεγκτή arduino, Λάρισα 2012
[3] Άννα Ντρέμπου, Κατασκευή γεννήτριας θορύβου, Τμήμα τεχνολογίας πληροφορικής και τηλεπικοινωνιών τομέας τηλεπικοινωνιών και δικτύων 2010
[4] Σάββας Πιπερίδης, Βηματικοί κινητήρες και σέρβο, Εργαστήριο Ευφυών Συστημάτων & Ρομποτικής, Εργαστηριακά Μαθήματα Ρομποτικής

Πηγές Διαδικτύου: [τελευταία πρόσβαση 12/2012]

Κεφάλαια 1ο, 2ο

http://el.wikipedia.org/wiki/Μικροελεγκτής
http://deltahacker.gr/2009/08/01/arduino-intro
http://arduino.cc
http://wiring.org.co/
http://processing.org
http://www.sparkfun.com
http://www.adafruit.com
http://www.ladyada.net/learn/arduino
http://makeprojects.com
http://www.instructables.com

Κεφάλαιο 3ο : Εργαστηριακές Ασκήσεις

1η

http://arduino.cc/en/Tutorial/ReadASCIIString
http://cloud.github.com/downloads/ghadjikyriacou/Learn-Arduino-GR/1-Εισαγωγή_0001.pdf
http://www.instructables.com/id/Arduino-Examples-1-Make-An-RGB-Led-Randomly-Flash/?ALLSTEPS
http://el.wikipedia.org/wiki/Βασικά_χρώματα
http://www.mbeckler.org/microcontrollers/rgb_led/
http://allaboutee.com/2011/05/16/arduino-tutorial-rgb-led/
http://pjrc.com/teensy/tutorial2.html
http://www.cs.ucy.ac.cy/~nicolast/courses/cs422/lectures/mm17.pdf
http://www.hacktronics.com/Tutorials/arduino-and-7-segment-led.html
http://www.electronicsblog.net/controling-7-segment-led-display-from-arduino-full-code-included/
http://www.proz.com/kudoz/english_to_greek/electronics_elect_eng/3789839-pull_up_resistor.html

2η
http://www.arduino.cc/en/Tutorial/Knock
http://www.arduino.cc/en/Tutorial/KnockSensor

3η
http://www.instructables.com/id/Arduino-Expermentation-Kit-How-to-get-Started-wi/step5/Spin-Motor-Spin-Transistor-amp-Motor-CIR/
http://www.electrojoystick.com/tutorial/?page_id=571
http://www.grobot.gr/index.php?option=com_content&view=article&id=137:pid-controler-dc-&catid=40:2008-04-19-12-23-10&Itemid=80
http://cloud.github.com/downloads/ghadjikyriacou/Learn-Arduino-GR/2-Ηλεκτρικοί_κινητήρες_0001.pdf
http://hackaday.com/2011/07/21/intro-to-dc-motor-control-using-the-sn754410/
http://users.ntua.gr/manias/HLE_ELE_KIN_DC_BIO.htmL
http://itp.nyu.edu/physcomp/Labs/DCMotorControl
http://arduino-info.wikispaces.com/SmallSteppers
http://club.dx.com/forums/Forums.dx/threadid.1183400
http://www.electronics-lab.com/pic-ingreek/samples/stepper/Operation_principle_of_stepper_motor.htm
http://arduino.cc/forum/index.php?topic=108983.0
http://www.jameco.com/Jameco/workshop/howitworks/how-servo-motors-work.html
http://www.engineersgarage.com/insight/how-servo-motor-works

4η
http://learn.adafruit.com/downloads/pdf/character-lcds.pdf
http://www.quinapalus.com/hd44780udg.html
http://arduino.cc/en/Tutorial/RowColumnScanning
http://playground.arduino.cc/Main/DirectDriveLEDMatrix
http://www.ics.forth.gr/~kateveni/120/12f/lab01_swLogic.html
http://www.tigoe.net/pcomp/code/category/arduinowiring/514

5η
http://subashchandra.com/electronics/distance-meter-using-hc-sr04-and-arduino/
http://arduino-info.wikispaces.com/UltraSonicDistance
http://www.electronicsblog.net/arduino-and-ultra-sonic-range-measurement-module-or-how-to-measure-the-pulse-time-with-a-hardware-timer-and-an-interrupt/
http://www.arduino.cc/en/Tutorial/Ping
http://www.ladyada.net/wiki/tutorials/learn/sensors/ir.html
http://www.arcfn.com/2009/08/multi-protocol-infrared-remote-library.html
http://letsmakerobots.com/node/29634

6η
http://playground.arduino.cc/Main/DS1302
http://www.henningkarlsen.com/electronics/library.php?id=5
http://club.dx.com/forums/forums.dx/threadid.1186105?page=2

7η
http://playground.arduino.cc/KeypadTutorial/EventKeypad
http://arduino.cc/playground/Main/KeypadTutorial
http://playground.arduino.cc/code/Keypad

8η
http://arduino.cc/en/Tutorial/Datalogger
http://www.ladyada.net/products/microsd/
http://elm-chan.org/docs/mmc/mmc_e.html
http://arduino-info.wikispaces.com/SD-Cards
http://learn.adafruit.com/photocells/using-a-photocell
http://learn.adafruit.com/thermistor/using-a-thermistor
http://playground.arduino.cc/ComponentLib/Thermistor2
http://www.hacktronics.com/Tutorials/arduino-thermistor-tutorial.html
http://bildr.org/2012/11/thermistor-arduino/
http://www.penguintutor.com/electronics/arduino2
http://learn.sparkfun.com/products/2

9η

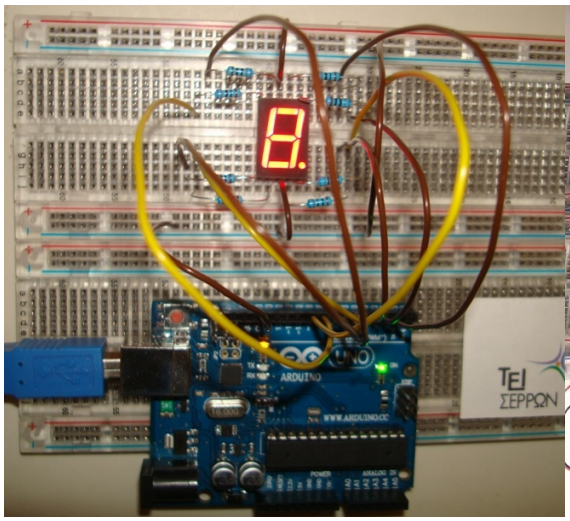
http://greekelectrician.blogspot.gr/2011/01/blog-post.html
http://www.electronics-base.com/index.php/general-description/control/101-relays
http://el.wikipedia.org/wiki/Ηλεκτρονόμος

10η

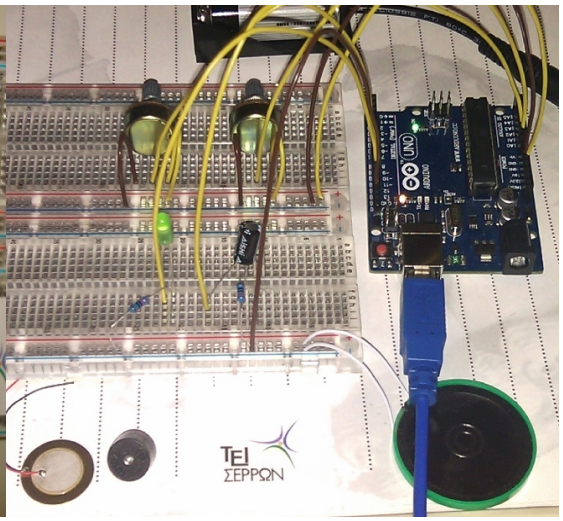
http://bildr.org/2011/02/74hc595/
http://arduino.cc/en/Tutorial/ShiftOut
http://www.dave-auld.net/index.php?option=com_content&view=article&id=94:arduino-digital-out-dual-shift-register&catid=53:arduino-input-output-basics&Itemid=107
http://fritzing.org/projects/nightrider-with-arduino-16-leds-and-two-74hc595/
http://www.arduino.cc/en/Reference/ShiftOut

ΠΑΡΑΡΤΗΜΑ Α

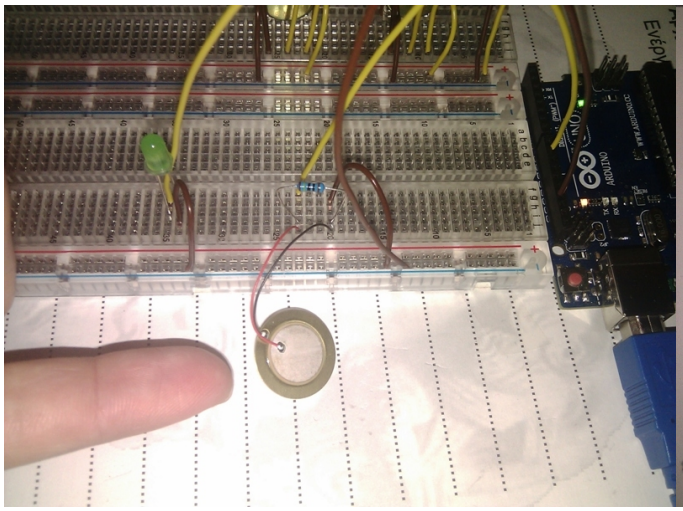
Φωτογραφίες κυκλωμάτων των 10 εργαστηριακών ασκήσεων.



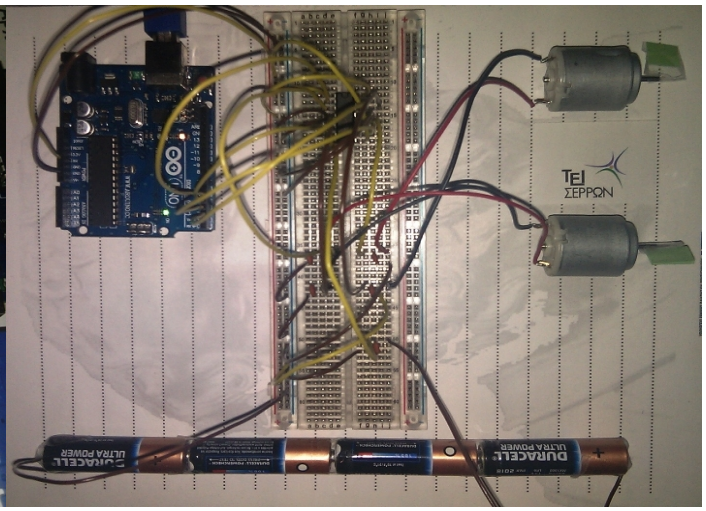
Εικόνα 1: Επτά-τμηματική μονάδα ένδειξης.



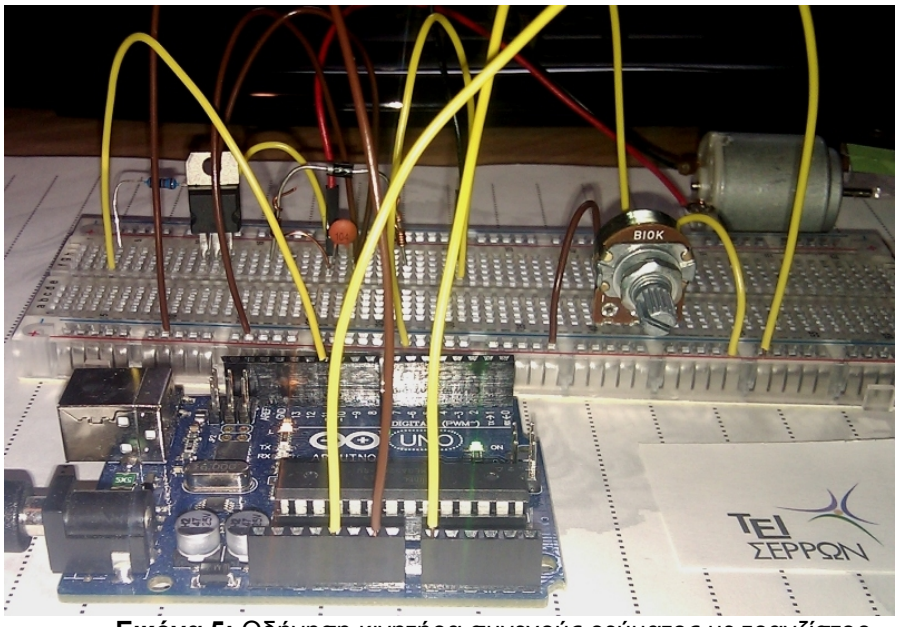
Εικόνα 2: Μεγαφωνάκι, Βομβητής.



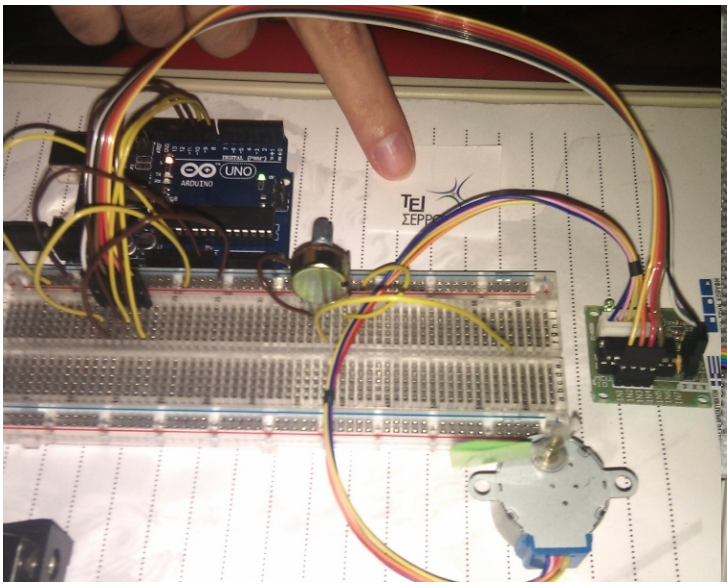
Εικόνα 3: Πιεζοηλεκτρικός δίσκος.



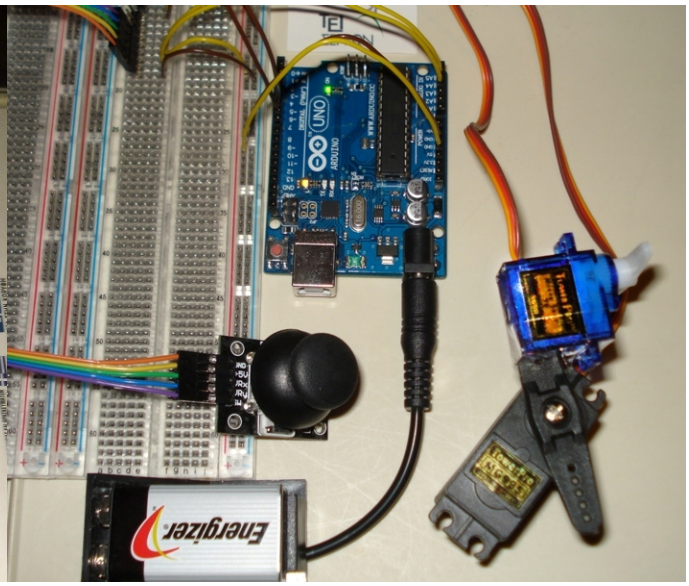
Εικόνα 4: Οδήγηση κινητήρων συνεχούς ρεύματος με χρήση H-Bridge SN754410.



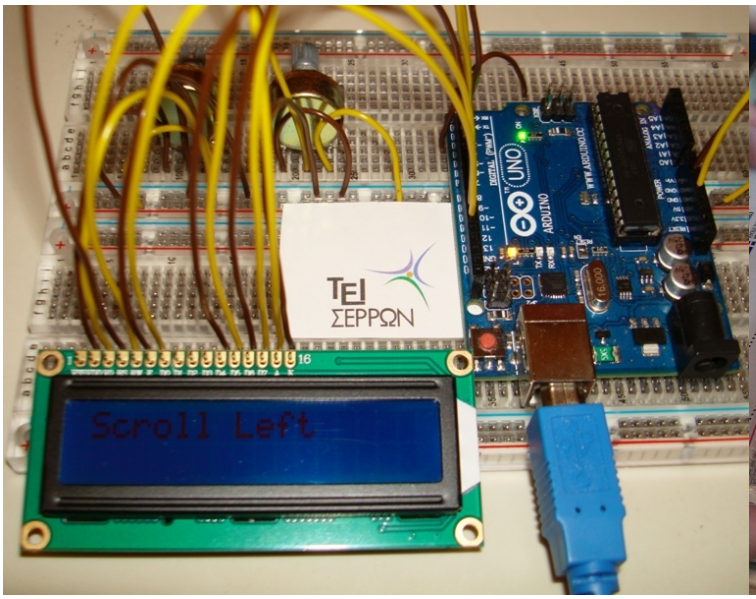
Εικόνα 5: Οδήγηση κινητήρα συνεχούς ρεύματος με τρανζίστορ.



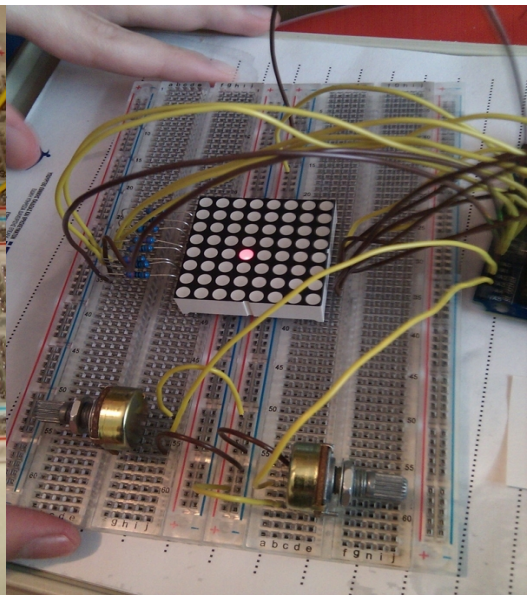
Εικόνα 6: Οδήγηση βηματικού κινητήρα με χρήση U2003 Darlington Array.



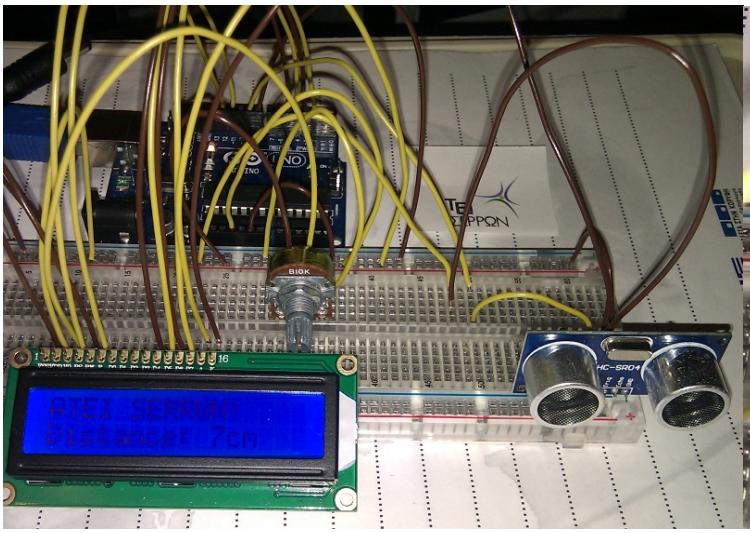
Εικόνα 7: Σερβοκινητήρες.



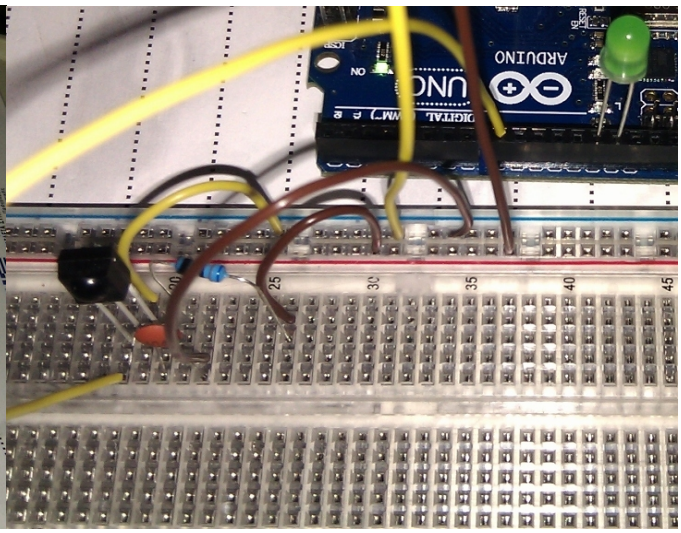
Εικόνα 8: Οθόνη υγρών κρυστάλλων.



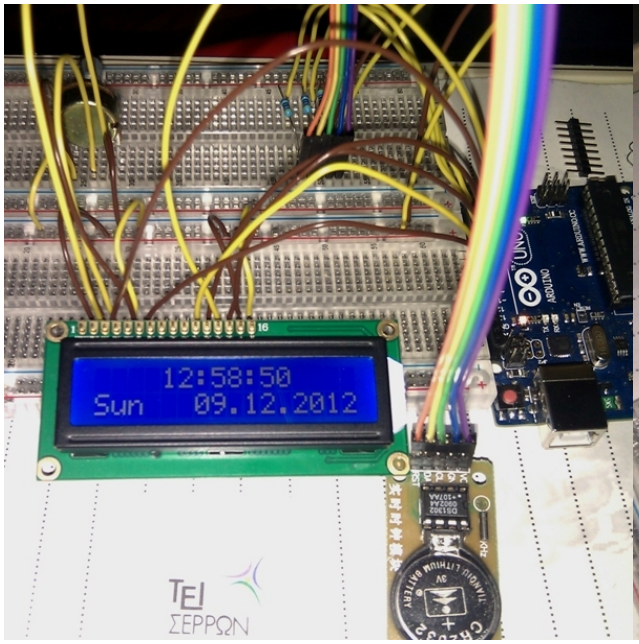
Εικόνα 9: Οθόνη LED Dot Matrix.



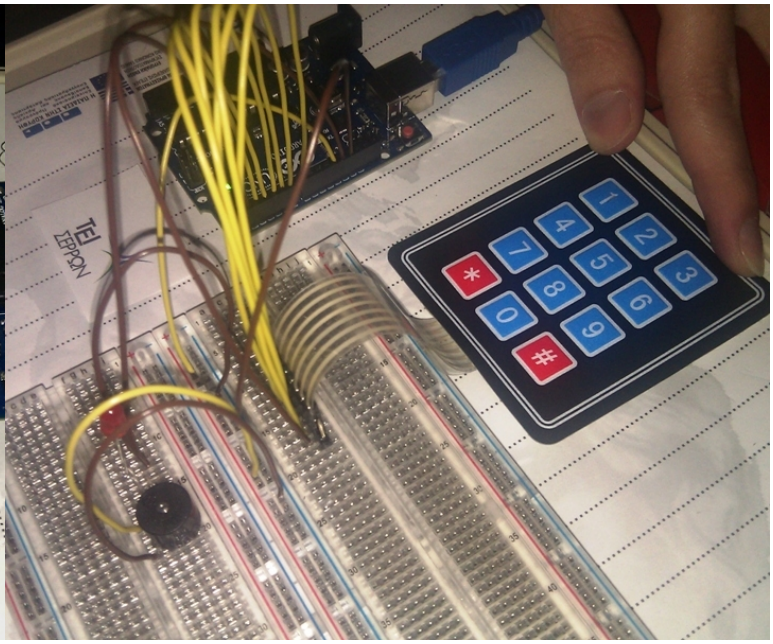
Εικόνα 10: Υπερηχητικός αισθητήρας.



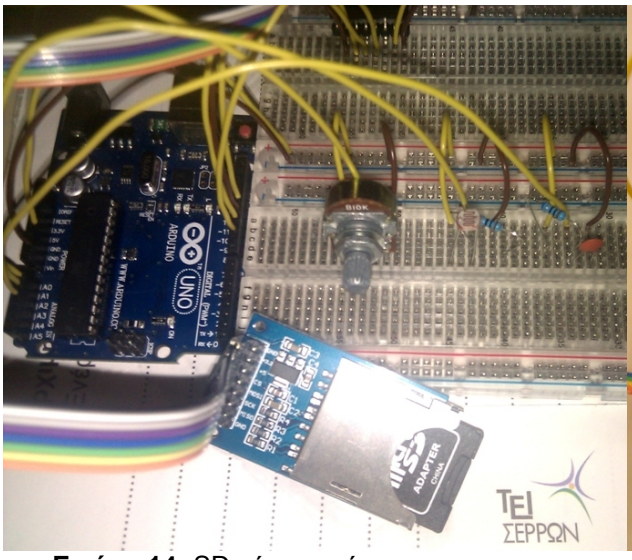
Εικόνα 11: Δέκτης υπερύθρων.



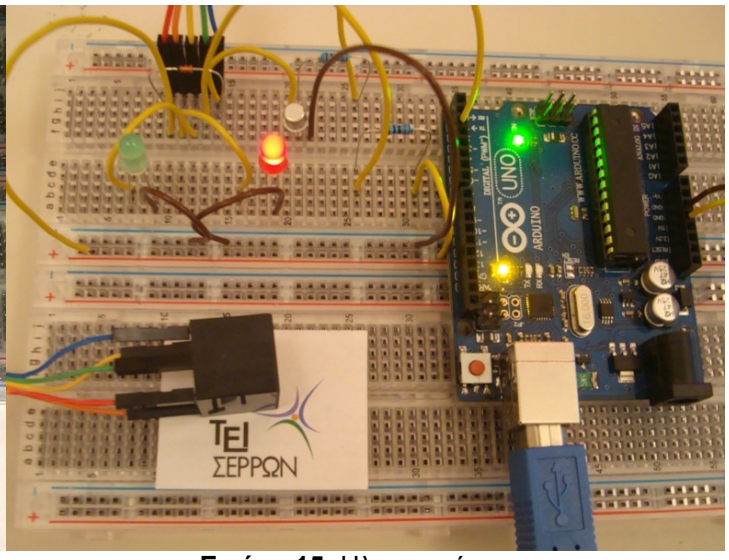
Εικόνα 12: Ρολόι πραγματικού χρόνου.



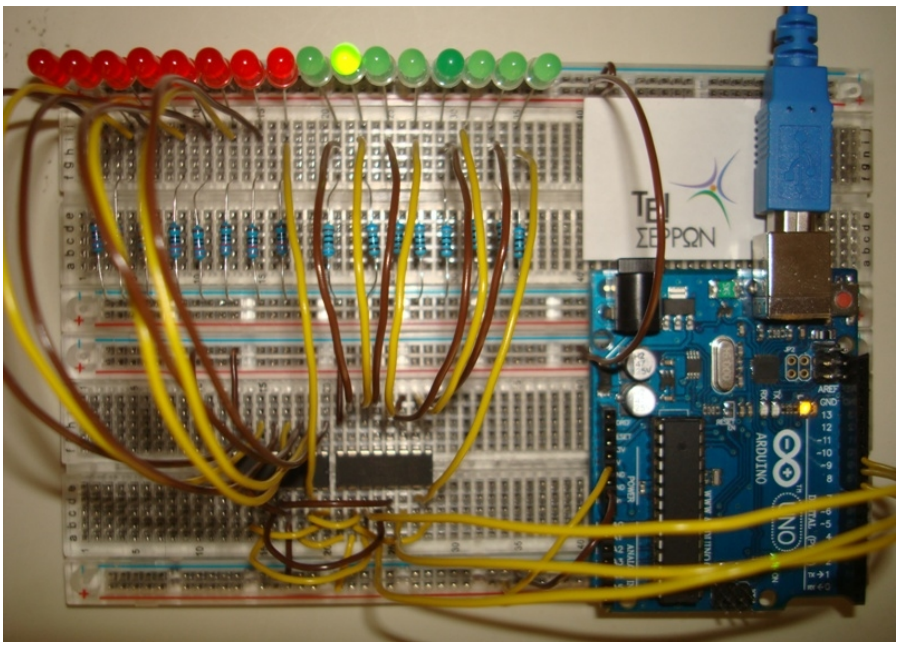
Εικόνα 13: Αριθμητικό πληκτρολόγιο.



Εικόνα 14: SD κάρτα επέκτασης.



Εικόνα 15: Ηλεκτρονόμος.



Εικόνα 16: 74HC595 καταχωρητές ολίσθησης.

ΠΑΡΑΡΤΗΜΑ Β

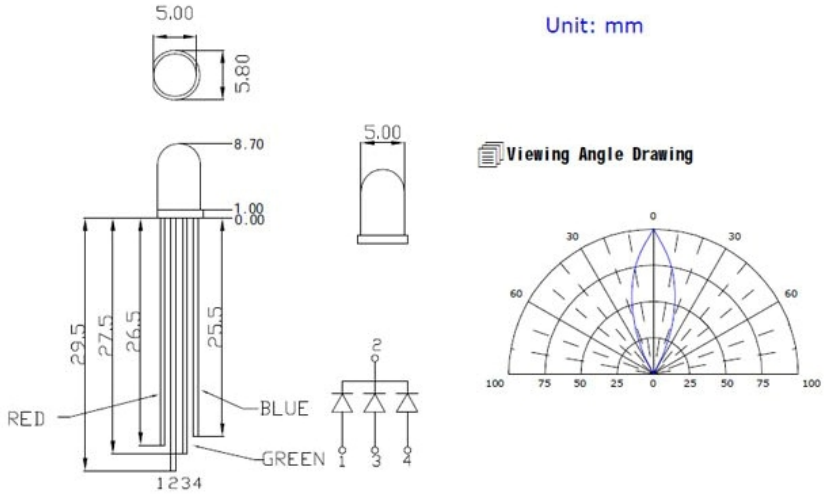
Ενδεικτικός Τιμοκατάλογος (Ιούλιος 2012)

Arduino UNO Rev.3 + USB cable		15,15€	LEG-5 5V Relay SPDT 5v		1,09 €
Solderless Breadboard - Transparent		2x 5,31 €	Darlington NPN TIP121 (Τρανζίστορ)		0,70 €
16x2 Character LCD Display Module with Blue Backlight		4,09 €	RGB LED 5mm (Diffused or Clear)		0,55 €
Breadboard Jumper Cable Wires for Electronic DIY		3,31 €	RED Led Display 0.56" (7-Segment)		2x 0,80 €
Toy Motor (1,5 - 6) V		2x 0,70 €	Linear Potentiometer 5KΩ-10KΩ		3x 1,19 €
SN754410 HBridge (Οδηγός για motor)		2,29 €	IR Receiver TSOP4838 38kHz (Δέκτης υπερύθρων)		1,50 €
SG90 Mini Servo with Gears and Parts (2Kg Torque)		3,48 €	Piezo Element (Πιεζοηλεκτρικός δίσκος)		0,65 €
DS1302 Real Time Clock Module with Battery		3,13 €	Thin Speaker 8 Ohm (Μεγαφωνάκι)		0,86 €
SD Card Module Slot Socket Reader		3,40 €	Female to Female 1-Pin DuPont Cables 40pcs/30cm		4,24 €
HC-SR04 Ultrasonic Sensor Distance Measuring Module		4,27 €	74HC595 shift register (Καταχωρητής)		2x 0,71 €
Matrix 3x4, 12 Key Membrane Switch Keypad Keyboard		3,05 €	Stepper Motor with ULN2003 Driver (Βηματικός Κινητήρας)		4,44 €
1/4W Resistance Metal Film Resistors (400-Piece Pack)		4,37 €	30x LED 5mm		Από 0,05 €
8x8 Dot-Matrix Red LED Display Board		2,50 €	DS18B20 (Αισθητήρας θερμοκρασίας)		2,13 €
Break Away Headers 1x40 Male (Long or Right Angle)		Από 0,48 €	1N4001 Diode Rectifier - 1A και 1N4148 Switching Signal Diode		Από 0,02 €
Photo resistor LDR 5mm (Φωτοαντίσταση)		2x 0,70 €	9V Battery Holder, DC Barrel Jack Plug		2,09 €
Towerpro MG995 Digital Metal Servo with Gears and Parts		8,02 €	Thumb Joystick Module for Electronic DIY - Black		2,73 €
Buzzer 5V (Βομβητής)		0,59 €	Tact switch 6x6mm 7mm 4pins (Κουμπί)		Από 0,10 €
Ceramic Capacitor 50V		Από 0,02 €	Electrolytic Capacitor 25V		Από 0,05 €

ΠΑΡΑΡΤΗΜΑ Γ

Φύλλα Δεδομένων

LED RGB 5mm YSL-R596CR3G4B5C-C10



LED Chip Typical Electircal & Optical Characteristics: (Ta=25℃)

ITEMS	Color	Symbol	Condition	Min.	Typ.	Max.	Unit
Forward Voltage	Red	V_F	I_F=20mA	1.8	2.0	2.2	V
	Green			3.0	3.2	3.4	
	Blue			3.0	3.2	3.4	
Luminous Intensity	Red	I_V	I_F=20mA	– – –	– – –	800	mcd
	Green			– – –	– – –	4000	
	Blue			– – –	– – –	900	
Wavelenength	Red	Δ λ	I_F=20mA	620	623	625	nm
	Green			515	517.5	520	
	Blue			465	466	467.5	
Light Degradation after 1000 hours	Red	-4.68% ~ -8.27%					
	Green	-11.37% ~ -15.30%					
	Blue	-8.23% ~ -16.81%					

8ohm 0.25w Speaker
Parameters:

1	Type	Dynamic speaker
2	Dimension	External diameter 40 mm
3	Rated Input Power	0.25 W
4	Impedance	8 ohm ± 15% at 1500Hz
5	Resonance Frequency (Fo)	440 Hz ± 20% at Fo, 1V
6	Sensitivity (S.P.L.)	85dB(W/m) ± 3 dB at AVE 0.6K,0.8K,1.0K,1.2K(Hz).
		96dB(0.25W/0.1m) ± 3 dB
7	Frequency Range	Fo – 20KHz
8	Distortion	Less than 10 % at 1500Hz 0.25W
9	Max. Input Power	Must be normal at 0.4W white noise for 1 minute.
10	Voice Coil	Diameter 10.8 mm
11	Magnet	Rare earth permanent (Nd-Fe-B) magnet Φ10 x 1.5mm
12	Weight	11g ± 2g

Piezo Element model (7BB-20-6L0)

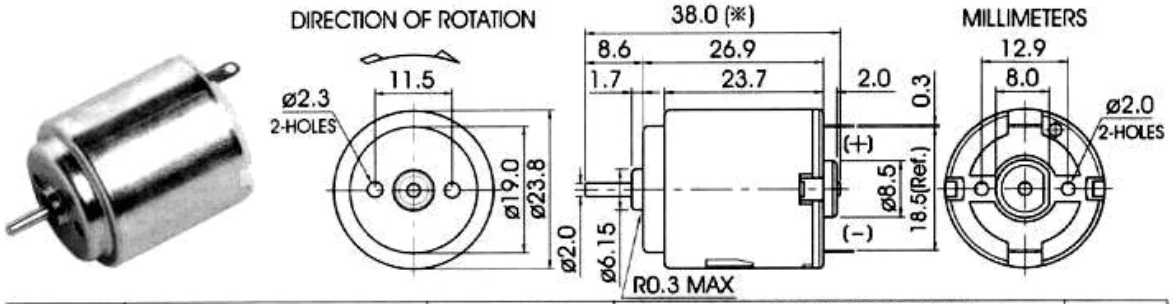
Part Number	Resonant Frequency (kHz)	Resonant Impedance (ohm)	Capacitance (nF)	Plate Size dia. D (mm)	Element Size dia. a (mm)	Electrode Size dia. b (mm)	Thickness T (mm)	Plate Thickness t (mm)	Plate Material
7BB-20-6L0	6.3 ±0.6kHz	1000 max.	10.0 ±30% [1kHz]	20.0	14.0	12.8	0.42	0.20	Brass (with Lead Wire: AWG32 Length 50mm)

Buzzer 5V CEM-1203

SPECIFICATION

No.	Item	Unit	Specification	Condition
1	Rated Voltage	Vo-p	3.5	Vo-p  0V
2	Operating Volt.	Vo-p	3.0~5.0	
3	Mean Current	mA	Max. 35	Applying rated voltage,2048Hz square wave, 1/2duty
4	Coil Resistance	Ω	42.0 ± 6.3	
5	Sound Output	dBA	Min.85 (Typical 95)	Distance at 10cm(A-weight free air). Applying rated voltage 2048Hz,square wave, 1/2duty
6	Rated Frequency	Hz	2048	
7	Operating Temp.	℃	-20 ~ +60	
8	Storage Temp.	℃	-30 ~ +70	
9	Dimension	mm	⌀12.0 × H8.5	See attached drawing.
10	Weight	gram	1.4	
11	Material		PPO(Black)	
12	Terminal		Pin type (Plating Au)	See attached drawing.
13	Environmental Protection Regulation		RoHS	

Hobbymotor (201-A)

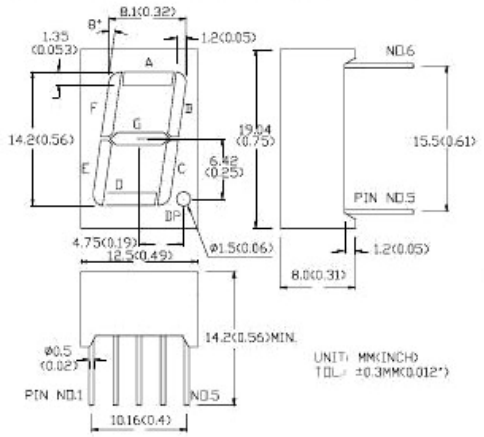


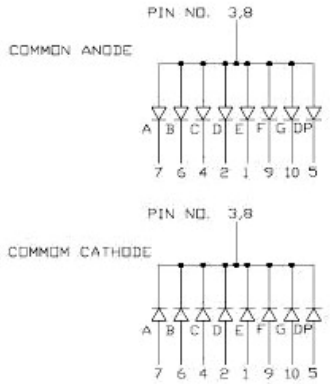
MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL TORQUE
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE	OUTPUT	EFF	
			rpm	A	rpm	A	g.cm	W	%	g.cm
201-A	1.5~4.5	3.0V CONSTANT	12800	0.32	10000	1.28	20	2.05	53.4	80
201-B	1.5~4.5	3.0V CONSTANT	11800	0.30	9350	1.04	17.5	1.67	53	73
201-G	1.5~6.0	3.0V CONSTANT	6750	0.16	5050	0.46	12	0.62	45	42

※38,42,45 WEIGHT: 28g (APPROX) (F.O.B) HKD.

7-seg LED numeric display FD-05611A/B-X

MECHANICAL OUTLINE & CIRCUIT DIAGRAM:





COMMON ANODE: FD-05611A-X

COMMON CATHODE: FD-05611B-X

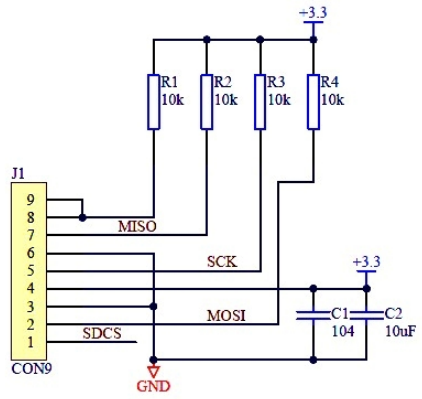
ELECTRO-OPTICAL CHARACTERISTICS (Ta=25℃)

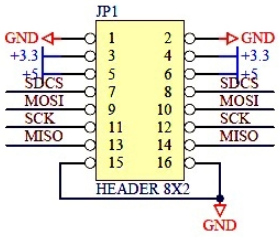
Part No.	Package Description		Emitting Color	Material	λd typ. (nm)	Δλ typ. (nm)	Vf(V)@20mA		Iv(mcd)@10mA	
	Height	Segment Color					Typ.	Max.	Min.	Typ.
FD-05611A/B-R	0.56"		Red	GaP	700	100	2.2	3.0	0.5	0.8
FD-05611A/B-SR	0.56"		Super Red	GaAlAs	660	20	1.9	2.5	2.0	4.0

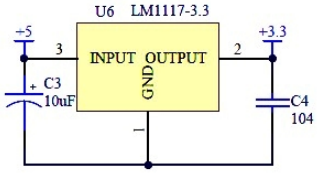
ABSOLUTE MAXIMUM RATINGS (Ta=25°C)

Parermeter	Symbl	Note	Maximum
Power Dissipation /Segment	Pd		100mW
Peak Forward Current / Segment	I_F/Pulse	(Pulse Width=1mS , Duty Ratio=1/10)	200mA
Continuous Forward Current / Segment	I_F/DC		20mA
Reverse Current /Segment	Ir	Reverse Voltage=5V	100μA
Segment to Segment Luminous Intensity Ratio		If=20mA	2:1
Operating Temperature Range	Topr		−20 to +80°C
Storage Temperature Range	Tstg		−25 to +80°C
Solder Temperature		1/16 Inch from body	260℃
Soldering Time (@260°C)			5 sec

SD Module

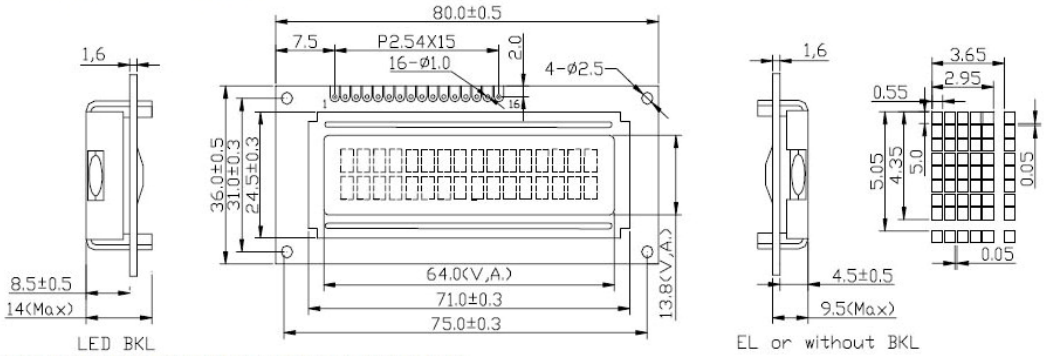




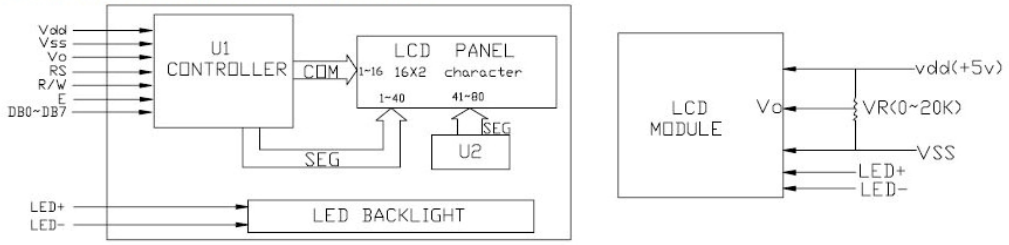


LCD VSTM-1602A

●OUTLINE DIMENSION



●BLOCK DIAGRAM & POWER SUPPLY



●INTERFACE PIN CONNECTIONS

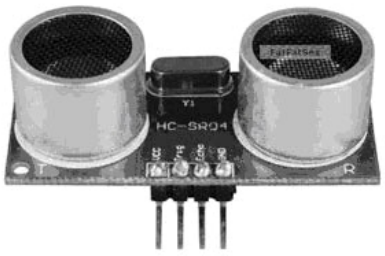
Pin No.	Symbol	Function
1	Vss	Ground for logic
2	Vdd	Power supply for logic
3	Vo	Power supply for LCD
4	RS	Register selection(H:Data,L:Instruction)
5	R/W	Read/Write selection(H:R,L:W)

Pin No.	Symbol	Function
6	E	Enable signal
7~14	DB0~DB7	Data bus lines
15	A	BKL+
16	K	BKL-

●ELECTRICAL CHARACTERISTICS

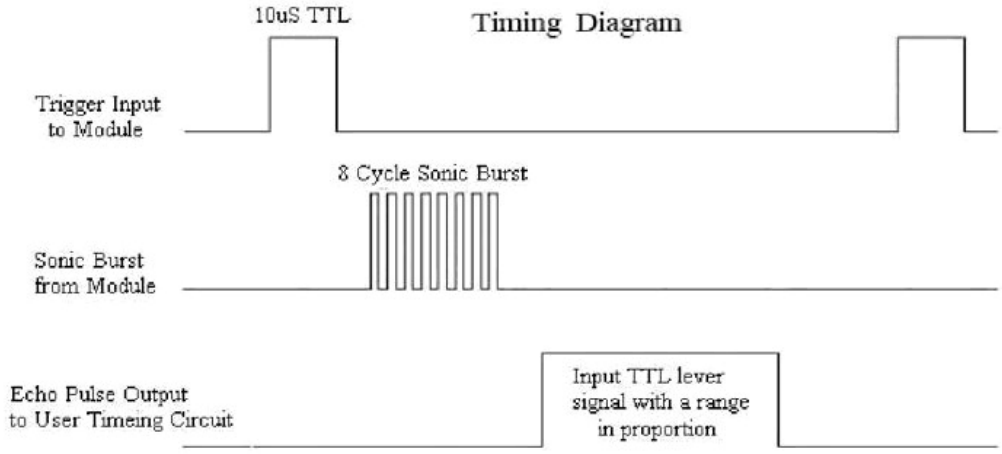
Item	Symbol	Test Condition	Min	Typ.	Max.	Unit
Operating Voltage	Vdd	Ta=25℃	…	5.0	…	V
Operating Voltage for LCD	Vlcd	Ta=25℃	…	4.5	…	V
Supply Current	Idd	Ta=25℃,Vdd=5.0V	…	2.0	3.0	mA
Supply Voltage for LED	Vf	Ta=25℃,R=6.8Ω	…	4.2	…	V
Supply Current for LED	If	Ta=25℃,Vf=4.2V	…	190	…	mA

Ultrasonic Ranging Module HC - SR04



Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



IR Receiver TSOP4838 for Remote Control Systems

ELECTRICAL AND OPTICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)						
PARAMETER	**TEST CONDITION**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
Supply current	E_v = 0, V_S = 5 V	I_{SD}	0.65	0.85	1.05	mA
	E_v = 40 klx, sunlight	I_{SH}		0.95		mA
Supply voltage		V_S	2.7		5.5	V
Transmission distance	E_v = 0, test signal see fig. 1, IR diode TSAL6200, I_F = 400 mA	d		45		m
Output voltage low	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m², test signal see fig. 1	V_{OSL}			100	mV
Minimum irradiance	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig. 1	$E_{e\ min.}$		0.17	0.35	mW/m²
Maximum irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig. 1	$E_{e\ max.}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		± 45		deg

TYPICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

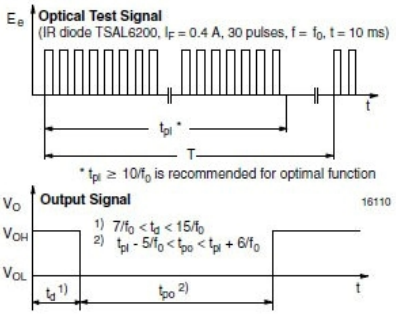


Fig. 1 - Output Active Low

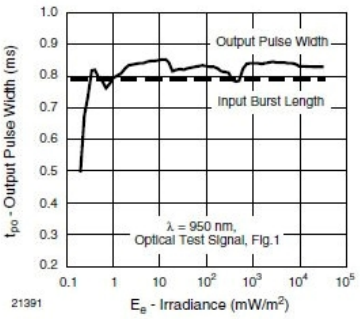
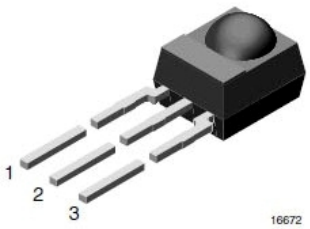


Fig. 2 - Pulse Length and Sensitivity in Dark Ambient



MECHANICAL DATA

Pinning for TSOP44.., TSOP48..:

1 = OUT, 2 = GND, 3 = V_S

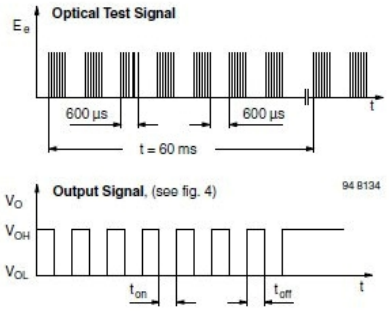


Fig. 3 - Output Function

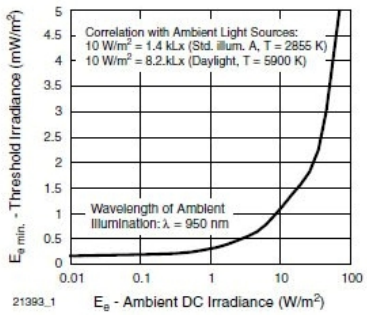
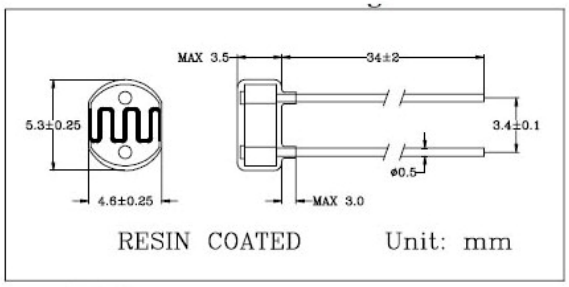


Fig. 6 - Sensitivity in Bright Ambient

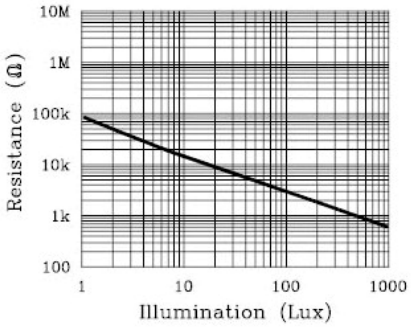
LDR KE-10720



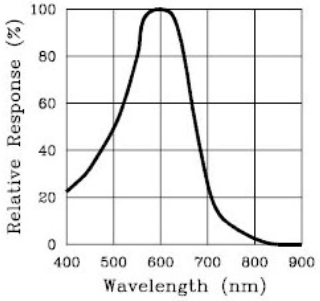
Electrical Characteristics (Ta=25°C)

Descriptions	Symbol	Min.	Typ.	Max.	Unit
Photo Resistance at 10 Lux (Light Source: 2856K)	R_L	10		20	kΩ
Dark Resistance After 10 sec. Removal of 10 Lux	R_D	0.5			MΩ
Gamma Value at 10 ~ 100 Lux	γ_{10}^{100}		0.7		
Maximum Power Dissipation	P_D			35	mW
Maximum Breakdown Voltage	V_{MAX}			100	V_{DC}
Peak Spectral Response	λp	550		650	nm
Rise Response Time at 1 fc	tr		35		ms
Fall Response Time at 1 fc	tf		5		ms
Ambient Temperature	T_A	-30 ~ +60			°C

Resistance vs Illumination

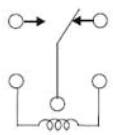


Relative Spectral Response

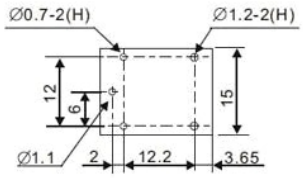


Relay 5V LEG-5

WIRING DIAGRAM (BOTTOM VIEW)



P.C.B LAYOUT (BOTTOM VIEW)



COIL DATA(0.36W, at 25°C)

Coil Nominal Voltage (VDC)	Resistance Tol.±10% (Ohms)	Nominal Current (mA)	Maximum Pick Up Voltage (V)	Minimum Drop Out Voltage (V)
3	25	120	2.1	0.3
5	70	72	3.5	0.5
6	100	60	4.2	0.6
9	225	40	6.3	0.9
12	400	30	8.4	1.2
24	1,600	15	16.8	2.4
48	6,400	7.5	33.6	4.8

Insulation Resistance	100 M Min. (DC 500V)
Dielectric Strength	750 VAC, 50/60Hz between contact. 1,500 VAC, 50/60Hz between all elements.
Contact Material	Silver- Cadmium Oxide as standard.
Contact Resistance	100 milliohms max. (initial value)
Shock Resistance	Malfunction: 10G(11ms) ; Destructive: 100G(6ms)
Vibration Resistance	Malfunction: 10 to 55 Hz. at Double Amplitude of 1.5 mm Destructive: 10 to 55 Hz. at Double Amplitude of 1.5 mm
Operation Time	8 ms max.
Release Time	8 ms max.
Temperature Range	- 25°C ~ + 60°C
Expected Life	With operation rate 30/min. Mechanical - 10,000,000 operations min. Electrical - 100,000 operations min. at rated load.