

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/328842270>

Προγραμματίζοντας με τον μικροελεγκτή Arduino

Book · January 2015

CITATIONS

0

READS

257

1 author:



Emmanouil Poulakis

University of Thessaly

6 PUBLICATIONS 0 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

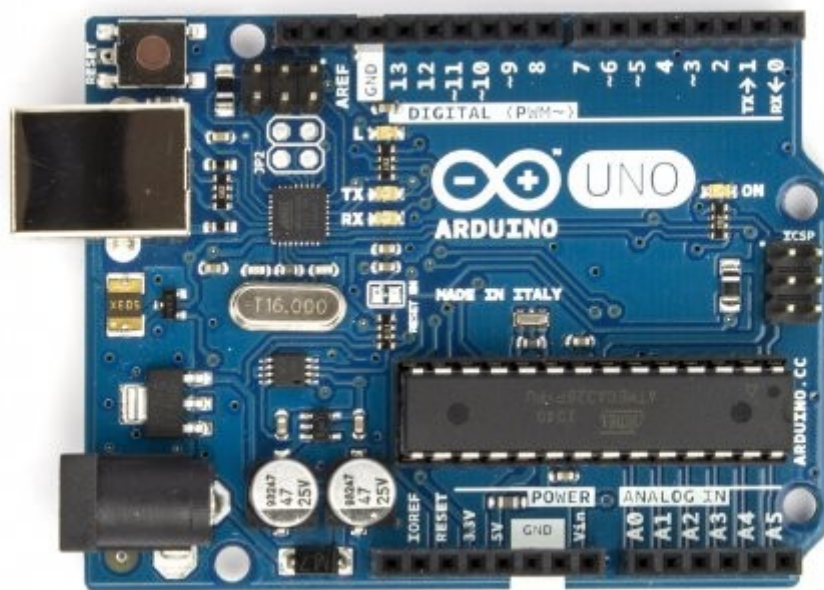


Η Ευαίσθητη Διστακτικότητα των Γυναικών στην Εκπαίδευση [View project](#)



Computational Thinking [View project](#)

Εμμανουήλ Πουλάκης



Προγραμματίζοντας με τον μικροελεγκτή Arduino

Ηράκλειο
Ιανουάριος 2015

Έκδοση 1η
Ηράκλειο, Ιανουάριος 2015

ISBN 978-960-93-6760-8



Αυτό το υλικό διατίθεται με άδεια Creative Commons Αναφορά Δημιουργού - Παρόμοια Διανομή 4.0 (<http://creativecommons.org/licenses/by-sa/4.0/>).
Η αναφορά σε αυτό θα πρέπει να γίνεται ως εξής:

Πουλάκης , Ε. (2015). *Προγραμματίζοντας με τον μικροελεγκτή Arduino*.
Ε. Πουλάκης: Ηράκλειο.

Ευχαριστώ τους/τις συναδέλφους
Βασίλη Βασιλάκη (Χίος),
Ρωξάνη Γραφανάκη (Ηράκλειο)
και **Γιώργο Μπουκέα** (Χίος),
εκπαιδευτικούς Πληροφορικής δευτεροβάθμιας εκπαίδευσης,
για την υποστήριξη και τις πολύ χρήσιμες ιδέες και παρατηρήσεις τους.

Η εικόνα του εξώφυλλου (Arduino UNO R3)
προέρχεται από τον επίσημο ιστότοπο του
Arduino (<http://www.arduino.cc>).

Οι εικόνες των εξαρτημάτων προέρχονται από το λογισμικό
ανοικτού κώδικα (open source) *Fritzing* (<http://www.fritzing.org>).
Το *Fritzing* χρησιμοποιήθηκε επίσης για την παραγωγή
όλων των σχεδίων των κυκλωμάτων στα φύλλα εργασίας.

Εμμανουήλ Πουλάκης
Εκπαιδευτικός Πληροφορικής ΠΕ19, M.Ed.
web: <http://users.sch.gr/manpoul>
email: manpoul@sch.gr

Περιεχόμενα

Εισαγωγή.....	7
1. Μικροελεγκτής και περιβάλλοντα προγραμματισμού Arduino.....	7
2. Προμήθεια απαραίτητων υλικών.....	8
3. Εγκατάσταση περιβάλλοντος Arduino IDE.....	9
4. Ρεύμα λειτουργίας.....	10
5. Θύρες εισόδου/εξόδου (Pins).....	10
6. Προγραμματισμός - Βασικές λειτουργίες.....	11
6.1. Δηλώσεις μεταβλητών.....	11
6.2. Σχόλια.....	12
6.3. Συναρτήσεις διαχείρισης θυρών εισόδου – εξόδου (Pins).....	12
6.4. Συναρτήσεις εισόδου - εξόδου ρεύματος.....	12
6.4.1. Ψηφιακή έξοδος.....	13
6.4.2. Ψηφιακή είσοδος.....	13
6.4.3. Αναλογική έξοδος (PWM pins).....	13
6.4.4. Αναλογική είσοδος.....	14
6.5. Συναρτήσεις χρόνου.....	14
6.5.1. Συνάρτηση καθυστέρησης – delay().....	14
6.5.2. Συνάρτηση καθυστέρησης - delayMicroseconds().....	15
6.5.3. Συνάρτηση καταγραφής χρόνου - millis().....	15
6.6. Συνάρτηση αντιστοίχισης τιμών - map().....	15
6.7. Η σειριακή θύρα επικοινωνίας (Serial).....	15
6.8. Δομή επιλογής.....	16
6.9. Δομή επανάληψης (For).....	17
7. Χρήση breadboard για συνδέσεις και βραχυκυκλώσεις.....	17
8. Μεταφόρτωση προγράμματος στη μονάδα μας.....	18
9. Βασική συνδεσμολογία για τα κυριότερα υλικά που θα χρησιμοποιήσουμε.....	19
9.1. Φωτοдиодοι - Leds.....	19
9.1.1. Έγχρωμοι φωτοдиодοι - RGB Leds.....	20
9.2. Κουμπιά - Buttons.....	21
9.3. Ποτενσιόμετρο - Potentiometer.....	22
9.4. Ηχείο – Sounder.....	22
9.5. Φωτοευαίσθητη αντίσταση – Photoresistor (LDR).....	23
9.6. Σέρβος - Servos.....	23
9.7. Κινητήρας συνεχούς – DC Motor.....	24
9.7.1. Χρήση δύο κινητήρων.....	24
9.8. Αισθητήρας υπερήχων – Ultrasonic sensor.....	25
9.9. Ασπίδες - Shields.....	26
10. Φύλλα εργασίας.....	26
10.1. Φύλλο εργασίας 1 – Το led που αναβοσβήνει.....	27
10.2. Φύλλο εργασίας 2 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out).....	28

10.3. Φύλλο εργασίας 3 – Αυξηση και μείωση φωτεινότητας (Fade in-Fade out), Αναβόσβημα στις άκρες (Blink at peaks).....	29
10.4. Φύλλο εργασίας 4 – Σταδιακή αύξηση και μείωση φωτεινότητας με ποτενσιόμετρο.....	30
10.5. Φύλλο εργασίας 5 - Χρησιμοποιώντας τη σειριακή οθόνη.....	31
10.6. Φύλλο εργασίας 6 – Παίζοντας με τα χρώματα (RGB Led).....	32
10.7. Φύλλο εργασίας 7 – Κατασκευάζοντας ένα RGB Led.....	33
10.8. Φύλλο εργασίας 8 – Τα φανάρια κυκλοφορίας.....	34
10.9. Φύλλο εργασίας 9 – Τα φανάρια κυκλοφορίας με φανάρι πεζών.....	35
10.10. Φύλλο εργασίας 10 – Τα φανάρια κυκλοφορίας με φανάρι πεζών και κουμπί διακοπής.....	36
10.11. Φύλλο εργασίας 11 – Εφέ, κυνηγώντας τη λάμψη (Led chase effect).....	37
10.12. Φύλλο εργασίας 12 – Εφέ, κυνηγώντας τη λάμψη (Led chase effect) με ποτενσιόμετρο.....	38
10.13. Φύλλο εργασίας 13 – Εφέ, ανιχνεύοντας το φως (Led effect + light sensor).....	39
10.14. Φύλλο εργασίας 14 – Χρησιμοποιώντας τον ήχο (Sounder).....	40
10.15. Φύλλο εργασίας 15 – Ήχος & ανίχνευση φωτός (Sounder & Light Sensor).....	41
10.16. Φύλλο εργασίας 16 – Κινώντας άξονες (Servos).....	42
10.17. Φύλλο εργασίας 17 – Κινώντας άξονες (Servos) με ποτενσιόμετρο.....	43
10.18. Φύλλο εργασίας 18 – Ελέγχοντας κινητήρες (DC Motor).....	44
10.19. Φύλλο εργασίας 19 – Ελέγχοντας 2 ή περισσότερους κινητήρες.....	45
10.20. Φύλλο εργασίας 20 - Χρησιμοποιώντας υπερήχους για τη μέτρηση μιας απόστασης.....	46
10.21. Φύλλο εργασίας 21 – Σύνθετη κατασκευή.....	47
11. Αναφορές.....	48
11.1. Βιβλιογραφία.....	48
11.2. Ιστότοποι με υλικό.....	48
Παράρτημα – Ενδεικτικά προγράμματα για τα φύλλα εργασίας.....	49
Φύλλο εργασίας 1 – Το Led που αναβοσβήνει.....	49
Φύλλο εργασίας 2 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out).....	49
Φύλλο εργασίας 3 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out), Αναβόσβημα στις άκρες (Blink at peaks).....	49
Φύλλο εργασίας 4 – Σταδιακή αύξηση και μείωση φωτεινότητας με ποτενσιόμετρο.....	50
Φύλλο εργασίας 5 – Χρησιμοποιώντας τη σειριακή οθόνη.....	50
Φύλλο εργασίας 6 – Παίζοντας με τα χρώματα (RGB Led).....	51
Φύλλο εργασίας 7 – Κατασκευάζοντας ένα RGB Led.....	51
Φύλλο εργασίας 8 – Τα φανάρια κυκλοφορίας.....	52
Φύλλο εργασίας 9 – Τα φανάρια κυκλοφορίας με φανάρι πεζών.....	53
Φύλλο εργασίας 10 – Τα φανάρια κυκλοφορίας με φανάρι πεζών και κουμπί διακοπής.....	54
Φύλλο εργασίας 11 – Εφέ, κυνηγώντας τη λάμψη (Led chase effect).....	55
Φύλλο εργασίας 12 – Εφέ, κυνηγώντας τη λάμψη (Led chase effect) με ποτενσιόμετρο.....	56
Φύλλο εργασίας 13 – Εφέ, ανιχνεύοντας το φως (Led effect + light sensor).....	57
Φύλλο εργασίας 14 – Χρησιμοποιώντας τον ήχο (Sounder).....	57
Φύλλο εργασίας 15 – Ήχος & ανίχνευση φωτός (Sounder & Light Sensor).....	58
Φύλλο εργασίας 16 – Κινώντας άξονες (Servos).....	58
Φύλλο εργασίας 17 – Κινώντας άξονες (Servos) με ποτενσιόμετρο.....	59
Φύλλο εργασίας 18 – Ελέγχοντας κινητήρες (DC Motor).....	59
Φύλλο εργασίας 19 – Ελέγχοντας 2 ή περισσότερους κινητήρες.....	59
Φύλλο εργασίας 20 – Χρησιμοποιώντας υπερήχους για τη μέτρηση μιας απόστασης.....	61

Εισαγωγή

Το παρόν εγχειρίδιο γράφτηκε με σκοπό να βοηθήσει μαθητές και εκπαιδευτικούς στο να εξερευνήσουν τις βασικές αρχές στον προγραμματισμό του μικροελεγκτή Arduino, χρησιμοποιώντας την πλατφόρμα Arduino UNO R3. Η φιλοσοφία που ακολουθείται είναι να δοθεί περισσότερο βάρος στον προγραμματισμό και τον κώδικα που γράφουν οι μαθητές και λιγότερο στο σχεδιασμό των κυκλωμάτων. Έτσι, σε όλα τα φύλλα εργασίας υπάρχουν τα σχέδια των κυκλωμάτων ή οδηγίες συνδεσμολογίας, ενώ ο μαθητής καλείται να γράψει τον αντίστοιχο κώδικα που θα κάνει να δουλέψει το κύκλωμα που του δίνεται, και που υλοποίησε στην τάξη.

Στα πρώτα κεφάλαια του εγχειριδίου αυτού παρουσιάζονται βασικές έννοιες ώστε να εισαχθεί κάποιος αρχάριος στον κόσμο του Arduino, των κυκλωμάτων και του προγραμματισμού του. Έγινε προσπάθεια να παρουσιαστούν πολύ συνοπτικά κάποια βασικά εξαρτήματα και η συνδεσμολογία τους, καθώς και κάποιες πολύ βασικές εντολές για τον προγραμματισμό. Η γλώσσα που χρησιμοποιεί το Arduino άλλωστε έχει βασιστεί στη C/C++, αρκετά διαδεδομένο περιβάλλον προγραμματισμού, για όποιον θέλει να αναζητήσει περισσότερες πληροφορίες. Με μια απλή αναζήτηση στο διαδίκτυο θα βρείτε επίσης πολλές πληροφορίες για το Arduino, τα εξαρτήματα και τον προγραμματισμό τους – λογικό είναι άλλωστε όταν μιλάμε για ένα περιβάλλον ανοικτού κώδικα να έχει δημιουργηθεί ήδη μια αρκετά μεγάλη κοινότητα. Στο τέλος του εγχειριδίου αναφέρονται σελίδες και άλλα εγχειρίδια, τα οποία με τη σειρά μας συμβουλευτήκαμε κατά τη σύνταξη του παρόντος.

Προτείνεται να γίνει μια απλή ανάγνωση στα κεφάλαια της θεωρίας, πριν τα φύλλα εργασίας, και να χρησιμοποιηθούν ως εγχειρίδιο αναφοράς, στα οποία θα ανατρέχει ο αναγνώστης κατά τη διάρκεια της εργασίας του με τα φύλλα. Ειδικότερα, **προτείνεται τα κεφάλαια 5, 6 και 9 να διαβαστούν ταυτόχρονα με τα αντίστοιχα φύλλα εργασίας** τα οποία εισάγουν τις έννοιες. Τα φύλλα εργασίας έχουν προσεχθεί ώστε να είναι μονοσέλιδα, παρέχοντας τη δυνατότητα της εύκολης εκτύπωσης και αναπαραγωγής. Για όλα τα φύλλα εργασίας υπάρχει ενδεικτικός κώδικας στο παράρτημα στο τέλος του εγχειριδίου, που υλοποιεί το αντίστοιχο πρόγραμμα στο περιβάλλον προγραμματισμού Arduino. Τέλος, σε κάθε φύλλο εργασίας υπάρχει αναφορά στα αντίστοιχα κεφάλαια της θεωρίας, με διάκριση σε αυτά που εισάγονται για πρώτη φορά.

Το παρόν εγχειρίδιο μπορεί να χρησιμοποιηθεί σε μια ερευνητική εργασία στο λύκειο ή μια βιωματική δράση στο γυμνάσιο, όπου οι μαθητές θα μπορούν να ψάχνουν περισσότερες πληροφορίες σε κάθε μάθημα σχετικά με αυτά που καλούνται να υλοποιήσουν και τελικά να προτείνουν μόνοι τους λύσεις στα προβλήματα και κυκλώματα με τα οποία ασχολούνται. Επίσης, με τη γνώση και εμπειρία που θα προκύψει από τα φύλλα εργασίας μπορεί μετά να υλοποιηθεί μια μεγαλύτερη κατασκευή που θα συγκεντρώνει αρκετά από τα επιμέρους τμήματα που έχουν ήδη τύχει επεξεργασίας και προγραμματισμού από την τάξη στην οποία απευθυνόμαστε. Για το λόγο αυτό προτείνεται στο τελευταίο φύλλο εργασίας μια σύνθετη κατασκευή ανάλογα με την εμπειρία κάθε μαθησιακής ομάδας (π.χ. αυτοκίνητο που κινείται μόνο του, ελέγχοντας για εμπόδια).

Τελικός σκοπός είναι να εισαχθεί ο μαθητής με ευχάριστο τρόπο στα ψηφιακά κυκλώματα και τον προγραμματισμό. Ευχαριστώ τους μαθητές του Γενικού Λυκείου Τζερμιάδων, και ειδικότερα τη Β' τάξη, οι οποίοι αποτέλεσαν την αρχική ομάδα μαθητών με την οποία δουλέψαμε το υλικό που ακολουθεί, στα πλαίσια της ερευνητικής εργασίας “Ηλεκτρονικά κυκλώματα και Προγραμματισμός με χρήση ανοικτής πλατφόρμας ανάπτυξης Arduino”, του σχ. έτους 2014-15.

1. Μικροελεγκτής και περιβάλλοντα προγραμματισμού Arduino

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοικτού κώδικα και σχεδιασμού, που βασίζεται σε

εύελικο και εύκολο στη χρήση υλικό και λογισμικό. Προορίζεται για καλλιτέχνες, σχεδιαστές, υλοποίηση χόμπι και δραστηριοτήτων, και γενικότερα για οποιονδήποτε ενδιαφέρεται να δημιουργήσει αλληλεπιδραστικά αντικείμενα ή περιβάλλοντα.

Για να μιλήσουμε λίγο πιο τεχνικά, υπάρχει ένα κύκλωμα που χρησιμοποιεί μικροελεγκτή, το οποίο μας δίνει ένα αριθμό πυλών οι οποίες μπορεί να λειτουργήσουν είτε ως εισόδοι είτε ως εξόδοι στα κυκλώματά μας. Αυτές τις εισόδους ή εξόδους μπορούμε να τις διαχειριστούμε γράφοντας κώδικα στο περιβάλλον προγραμματισμού Arduino IDE που έχει βασιστεί στη γλώσσα C/C++.

Στην επίσημη σελίδα του Arduino (<http://arduino.cc/>) μπορείτε να βρείτε πολλές πληροφορίες για αυτό, και να κατεβάσετε το περιβάλλον προγραμματισμού από την αντίστοιχη σελίδα (<http://arduino.cc/en/Main/Software>).

Εκτός από τη βασική έκδοση του περιβάλλοντος Arduino IDE, υπάρχει και μια παραλλαγμένη έκδοση του Scratch*, η οποία μπορεί να χρησιμοποιηθεί για να γράψουμε προγράμματα για το Arduino, η S4A - Scratch For Arduino*, η οποία επίσης είναι ανοικτού κώδικα και δωρεάν. Το πλεονέκτημα της έκδοσης αυτής είναι ο οπτικός προγραμματισμός (blocks όπως στο Scratch) σε σχέση με το γράψιμο εντολών στο κλασσικό περιβάλλον. Παρόμοιας λογικής είναι και το ArduBlock*, το οποίο επίσης χρησιμοποιεί οπτικό προγραμματισμό μέσω έτοιμων blocks για τον προγραμματισμό του. Ακόμα, υπάρχουν οπτικές εκδόσεις στο διαδίκτυο (web περιβάλλοντα), όπως το BlocklyDuino* ή το ArduinoMio*. Μπορείτε να επισκεφθείτε τα περιβάλλοντα αυτά από τις αντίστοιχες ιστοσελίδες τους*.

Στις επόμενες ενότητες θα δούμε αναλυτικά τη λειτουργία του μικροελεγκτή και τον βασικό προγραμματισμό του, μέσα από την κλασσική πλατφόρμα του Arduino IDE, ώστε να εισαχθεί ο αναγνώστης στο βασικό περιβάλλον και αφού αποκτήσει μια πρώτη ευχέρεια και κατανόηση των βασικών αρχών στον προγραμματισμό, να δοκιμάσει μόνος ή με βοήθεια και τις υπόλοιπες προσφερόμενες λύσεις.

2. Προμήθεια απαραίτητων υλικών

Το κύκλωμα των μονάδων του Arduino είναι ανοικτό, δηλαδή ο σχεδιασμός και τα μέρη του είναι γνωστά και δίνονται από τους κατασκευαστές του, με αποτέλεσμα όποιος θελήσει να μπορεί να το υλοποιήσει. Έτσι, υπάρχει υλικό με την ονομασία Arduino που προέρχεται από τους δημιουργούς και επίσημους κατασκευαστές του στην Ιταλία, ενώ μπορείτε να βρείτε πάρα πολλές ακόμα υλοποιήσεις μονάδων του, απόλυτα συμβατές με τα προγράμματα και κυκλώματα που ενδεχομένως ήδη υπάρχουν και δουλεύουν με τις επίσημες μονάδες Arduino. Η μοναδική δέσμευση που ζήτησαν οι δημιουργοί του Arduino, είναι να αναφέρονται με άλλη ονομασία οι κατασκευές τρίτων, κρατώντας την ονομασία Arduino για αυτούς. Η κοινότητα το σεβάστηκε κι έτσι θα βρείτε να κυκλοφορούν πολλές άλλες εκδόσεις οι οποίες συνήθως έχουν ονόματα που καταλήγουν σε -ino, όπως μια από τις κινέζικες εκδόσεις του το Funduino.

Οι επίσημοι δημιουργοί πουλάνε υλικό μέσω της ιστοσελίδας τους (<http://store.arduino.cc/>), ενώ υπάρχουν και επίσημοι συνεργάτες για την Ελλάδα (<http://arduino.cc/en/Main/Buy>).

Αυτό που θα χρειαστείτε για αρχή είναι:

- μια μονάδα, όπως είναι το Arduino Uno R3 το οποίο χρησιμοποιούμε και στα παραδείγματα εδώ, το οποίο με 14 ψηφιακές εισόδους/εξόδους και 6 αναλογικές εισόδους (Pins) είναι υπεραρκετό για τις πρώτες σας εφαρμογές (και το αντίστοιχο usb καλώδιο για να συνδεθεί στον υπολογιστή σας),
- καλώδια για να συνδέετε τα pins με ότι διαχειρίζεστε,

* Οι διευθύνσεις στο διαδίκτυο για τα περιβάλλοντα αυτά δίνονται στην παράγραφο 11.2. Ιστότοποι με Υλικό).

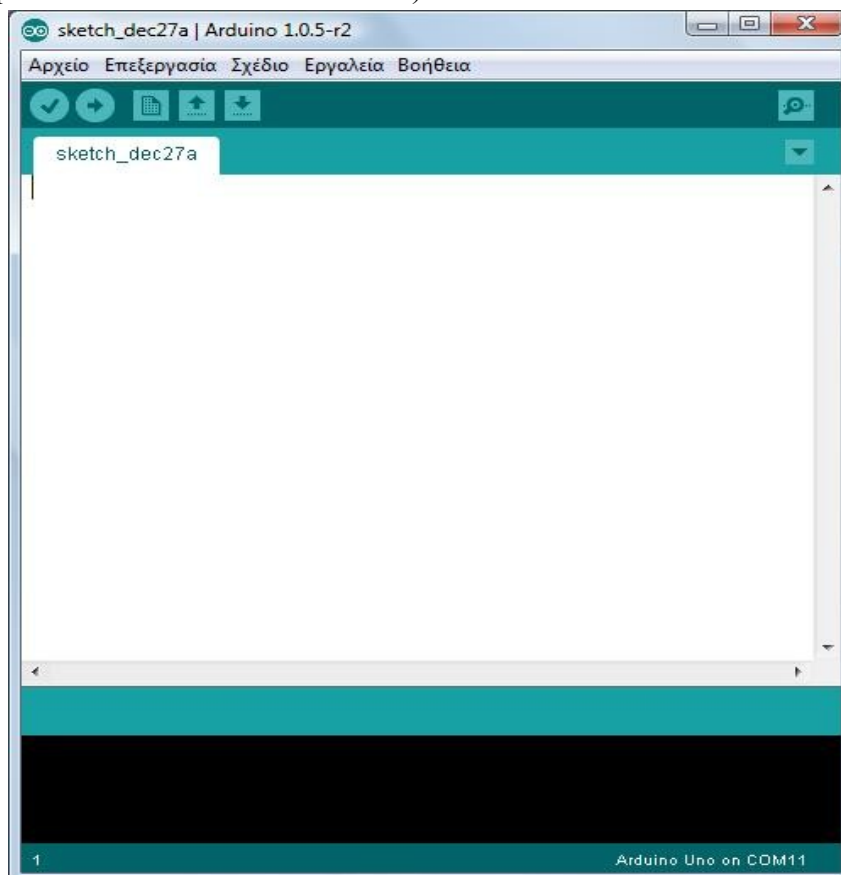
- μια βάση όπου μπορείτε να συνδέετε και να βραχυκυκλώνετε αυτά που χρησιμοποιείτε (breadboard) και τέλος
- μερικά leds, αντιστάσεις, πυκνωτές, ποτενσιόμετρα, αισθητήρες κτλ για τις πρώτες σας υλοποιήσεις.

Γενικά, προτείνουμε να βρείτε ένα **κιτ αρχαρίων (starter kit)** που περιλαμβάνει συνήθως μονάδα, usb καλώδιο, breadboard, καλώδια με σκληρό βύσμα στην άκρη (θηλυκά και αρσενικά), καθώς και αρκετά leds, αντιστάσεις, πυκνωτές, ποτενσιόμετρο, φωτοευαίσθητες αντιστάσεις, κουμπιά κτλ, όπως π.χ. το Arduino Starter Kit (<http://store.arduino.cc/product/K000007>).

Είναι θεμιτό γενικά να υποστηρίζονται οι κατασκευαστές και δημιουργοί του Arduino, στο σημείο αυτό όμως θα πρέπει να παρατηρήσουμε ότι αντίστοιχες υλοποιήσεις, υλικά και κιτ του Arduino, ειδικά αυτά που προέρχονται από την Κίνα αγοράζονται με πολύ μικρότερο κόστος, συμπεριλαμβανομένων και των μεταφορικών τους, που μπορεί να φτάσει και στο 1/3 της αντίστοιχης αξίας.

3. Εγκατάσταση περιβάλλοντος Arduino IDE

Για να προγραμματίσετε τη μονάδα σας θα χρειαστείτε το περιβάλλον προγραμματισμού Arduino IDE (εικόνα 1). Στο περιβάλλον αυτό γράφετε κώδικα (βασίζεται στη γλώσσα C/C++) τον οποίο μετά μεταγλωττίζετε και μεταφορτώνετε στη μονάδα σας. Το Arduino IDE υπάρχει σε εκδόσεις για Windows, Mac και Linux και μπορείτε να το κατεβάσετε εντελώς δωρεάν από την επίσημη ιστοσελίδα (<http://arduino.cc/en/Main/Software>).



Εικόνα 1 – Περιβάλλον προγραμματισμού Arduino IDE

Το περιβάλλον αυτό έχει εξελληνισμένο μενού, καθώς και αρκετά έτοιμα παραδείγματα χρήσης βασικών λειτουργιών (Αρχείο => Παραδείγματα).

4. Ρεύμα λειτουργίας

Το Arduino μπορεί να δουλέψει με ρεύμα από τη USB θύρα του υπολογιστή σας ή με αυτόνομη παροχή ρεύματος από μπαταρία. Η μονάδα παρέχει σταθερά τάση 5V στις εξόδους της.

Για παροχή ρεύματος στη μονάδα από εξωτερική πηγή δέχεται τροφοδοσία από εξωτερικό βύσμα - συνιστώμενη παρεχόμενη τάση λειτουργίας είναι στα 7V έως 12V, ώστε να μπορεί να λειτουργήσει και να δώσει σταθερά τα 5V στην έξοδο (βλ. <http://arduino.cc/en/Main/arduinoBoardUno>). Μπορείτε να συνδέσετε την παροχή ρεύματος απευθείας στα pins που προορίζονται για αυτό το σκοπό: (+) στο Pin VCC IN και (-) στο Gnd δίπλα του.

Στην περίπτωση που είναι συνδεδεμένη η μονάδα σας μόνιμα με θύρα USB τότε δουλεύει χωρίς πρόβλημα με τα 5V που παρέχει η USB θύρα.

5. Θύρες εισόδου/εξόδου (Pins)

Το Arduino Uno R3 έχει 14 ψηφιακές θύρες εισόδου ή εξόδου (digital input/output pins) και έξι αναλογικές εισόδους (analog input pins). Οι 14 ψηφιακές θύρες ονομάζονται με νούμερα από το 0 έως το 13, ενώ οι έξι αναλογικές με το γράμμα A ακολουθούμενο από ένα νούμερο από 0 μέχρι το 5 (π.χ. A3). Στην έξοδο τα pins μπορούν να δώσουν 0 έως και 5V τάση. Από τις 14 ψηφιακές θύρες οι έξι, και ειδικότερα οι 3, 5, 6, 9, 10, 11, είναι και PWM θύρες (Pulse Width Modulation), δηλαδή μπορούν να προσομοιώσουν αναλογικές εξόδους.

Έτσι, συνοπτικά για την είσοδο και έξοδο έχουμε:

- Για **ψηφιακή είσοδο**, χρησιμοποιούμε τις 14 ψηφιακές 0..13. Όταν δουλεύουν ψηφιακά, η είσοδος μπορεί να είναι ή 0 ή 5V, με τον χαρακτηρισμό LOW ή HIGH όπως θα δούμε παρακάτω.
- Για **ψηφιακή έξοδο**, χρησιμοποιούμε τις 14 ψηφιακές 0..13. Όταν δουλεύουν ψηφιακά, η έξοδος μπορεί να είναι 0 ή 5V, με τον χαρακτηρισμό LOW ή HIGH όπως θα δούμε παρακάτω.
- Για **αναλογική είσοδο**, δηλαδή να διαβάσουμε τιμές ρεύματος στο διάστημα 0 έως 5V, χρησιμοποιούμε τις έξι αναλογικές θύρες A0..A5.
- Για **αναλογική έξοδο**, μπορούμε να χρησιμοποιήσουμε τις έξι PWM ψηφιακές θύρες (3, 5, 6, 9, 10, 11), οι οποίες θα μας δώσουν ρεύμα εξόδου όποιας τιμή θέλουμε στο διάστημα από 0 έως 5V.

Γράφοντας κώδικα θα πρέπει να αρχικοποιήσουμε τις θύρες που χρησιμοποιούμε με τη συνάρτηση `pinMode()`, δηλαδή να δίνουμε την πληροφορία για όποιες χρησιμοποιήσουμε αν θα είναι για είσοδο ή για έξοδο. Η συνάρτηση αυτή αναλύεται στην επόμενη ενότητα.

Όταν χρησιμοποιείται η σειριακή οθόνη παρακολούθησης της επικοινωνίας με τον υπολογιστή, χρησιμοποιούνται τα pins 0 και 1 για αυτό, οπότε προτείνουμε να μην τα χρησιμοποιείτε στις εφαρμογές σας, εκτός αν αυτό είναι απαραίτητο (π.χ. δεν μας φτάνουν τα υπόλοιπα 12 pins για την εφαρμογή μας).

Επίσης, στη θύρα 13 υπάρχει συνήθως συνδεδεμένο ήδη ένα Led πάνω στην πλακέτα Arduino Uno, κι έτσι μπορούμε να το χρησιμοποιούμε για σχετικές λειτουργίες.

6. Προγραμματισμός - Βασικές λειτουργίες

Μετά την εγκατάσταση του Arduino IDE μπορούμε να γράψουμε τα πρώτα μας τμήματα κώδικα.

Η λογική του Arduino είναι πολύ απλή - στην ουσία υπάρχουν δύο βασικές συναρτήσεις, η `setup()` και η `loop()` οι οποίες δουλεύουν ως εξής:

- **setup()** - εδώ βάζουμε όλες τις εντολές που πρέπει να τρέξουν μία φορά, όταν ενεργοποιείται η μονάδα μας (όταν δηλαδή δίνουμε ρεύμα ή όταν πατηθεί το πλήκτρο reset που υπάρχει). Συνήθως μπαίνουν αρχικοποιήσεις τιμών μεταβλητών και οπωσδήποτε ο χαρακτηρισμός των εισόδων/εξόδων που θα χρησιμοποιήσουμε (αν δηλαδή ένα συγκεκριμένο Pin θα είναι είσοδος ή έξοδος).
- **loop()** - εδώ γράφουμε το πρόγραμμά μας. Οι εντολές που υπάρχουν θα τρέξουν κι όταν φτάσει στο τέλος θα ενεργοποιηθεί ξανά η `loop()`, συνεχίζοντας από την αρχή της, και ξανά. Αυτό θα συμβαίνει συνεχώς, όσο έχει ρεύμα το Arduino ή μέχρι να πατηθεί το πλήκτρο reset.

Έτσι, η βασική λειτουργία του Arduino είναι ότι τρέχει η συνάρτηση `setup()` μία φορά στην αρχή και ακολούθως η `loop()` ξανά και ξανά μέχρι να το κλείσουμε (να μην τροφοδοτείται με ρεύμα) ή να πατήσουμε το πλήκτρο reset.

Στην περίπτωση του Reset ξανατρέχει η συνάρτηση `setup()` μία φορά και ακολούθως η `loop()` ξανά και ξανά, όπως δηλαδή ακριβώς και όταν αρχικά ενεργοποιείται με ρεύμα ο μικροελεγκτής. Στην περίπτωση που έχουμε κάνει αλλαγές στο πρόγραμμά μας και το φορτώσουμε στον μικροελεγκτή (θα δούμε παρακάτω τη διαδικασία αυτή) αρκεί να πατήσουμε το πλήκτρο Reset ώστε να φορτώσει το πρόγραμμά μας από την αρχή με τον τρόπο που περιγράφηκε.

Ένα τυπικό πρόγραμμα έχει την παρακάτω δομή:

```
void setup() {  
    /* οι εντολές εδώ θα τρέξουν μόνο στην ενεργοποίηση ή μετά από Reset */  
}  
void loop() {  
    /* οι εντολές εδώ θα τρέχουν ξανά και ξανά,  
    μέχρι να απενεργοποιηθεί ή να πατηθεί το Reset */  
}
```

6.1. Δηλώσεις μεταβλητών

Όπως σε όλες τις γλώσσες προγραμματισμού, μπορώ να δηλώσω ονόματα μεταβλητών. Οι τύποι μεταβλητών που υποστηρίζονται στο Arduino είναι αρκετοί. Για έναν αρχάριο χρήστη οι παρακάτω τύποι θα είναι αρκετοί:

- **boolean**, με τιμές το 0 και 1 (ή True – False)
- **byte**, με τιμές από 0 έως και 255
- **int**, ακέραιος με δυνατές τιμές από -32768 έως και 32767
- **long**, ακέραιος με δυνατές τιμές από -2147483648 έως και 2147483647
- **float**, δεκαδικοί αριθμοί
- **char**, ένας χαρακτήρας (μέγεθος ένα Byte)
- **string**, πίνακας χαρακτήρων

Ένα παράδειγμα δήλωσης μεταβλητών δίνεται παρακάτω:

```
int ledPin = 13; // ορίζω ακέραια μεταβλητή ledPin και αρχικοποιώ την τιμή της σε 13
float SinVal; // ορίζω πραγματική μεταβλητή SinVal
```

6.2. Σχόλια

Όπως σε όλες τις γλώσσες προγραμματισμού, μπορώ να έχω σχόλια για την ευκολότερη κατανόηση και συντήρηση του κώδικα που γράφω. Μπορώ να χρησιμοποιήσω τις δύο κάθετες // για σχόλιο σε μία γραμμή (ότι ακολουθεί τις // αγνοείται), ή τα /* */ που περικλείουν τα σχόλια που γράφονται σε περισσότερες γραμμές (ότι υπάρχει ανάμεσα στο /* και στο */ αγνοείται).

Για παράδειγμα:

```
int ledPin = 13; // ορίζω τον αριθμό του Pin για το LED
/* Στον κώδικα που ακολουθεί θα προγραμματίσουμε ένα
LED να αναβοσβήνει ανά 1 sec */
```

6.3. Συναρτήσεις διαχείρισης θυρών εισόδου – εξόδου (Pins)

Όπως αναφέρθηκε, η κύρια λειτουργία του μικροελεγκτή βασίζεται στο να ελέγχει τις θύρες που διαθέτει και είτε να δίνει ρεύμα είτε να παίρνει ρεύμα από αυτές. Στην αρχικοποίηση κάθε προγράμματος (μέσα στη συνάρτηση setup) θα χρειαστεί να χαρακτηρίσουμε τα Pins που χρησιμοποιούμε ως είσοδο ή ως έξοδο.

Η συνάρτηση pinMode(Pin, Mode) χρησιμοποιείται με το όνομά της και ορίσματα α) τον αριθμό Pin και β) την κατάσταση λειτουργίας που χαρακτηρίζεται με τη λέξη INPUT (είσοδος) ή OUTPUT(έξοδος).

Όπως έχουμε αναφέρει έχουμε 14 ψηφιακά Pins, 6 εκ των οποίων είναι PWM, με ονόματα 0..13 και έξι αναλογικά με ονόματα A0..A5.

Για παράδειγμα:

```
pinMode(12, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(A2, INPUT);
```

6.4. Συναρτήσεις εισόδου - εξόδου ρεύματος

Για να μπορέσουμε να δώσουμε ρεύμα προς τα έξω μέσω μιας θύρας (pin) θα πρέπει πρώτα να έχει αυτή οριστεί ως εξόδου, όπως είδαμε στην προηγούμενη παράγραφο. Ακολουθώντας, με χρήση της κατάλληλης εντολής μπορούμε να δώσουμε κάθε φορά την επιθυμητή τάση προς τα έξω.

Αντίστοιχα, για να “διαβάσουμε” από μια είσοδο, θα πρέπει αρχικά να την ορίσουμε ως είσοδο και με χρήση της κατάλληλης κάθε φοράς συνάρτησης να διαβάζουμε την αντίστοιχη τιμή.

6.4.1. Ψηφιακή έξοδος

Και τα 14 pins του Arduino μπορούν να δουλεύουν ως ψηφιακές εξόδους, δηλαδή δίνουν έξοδο 0 ή 5V. Αυτό γίνεται με χρήση της συνάρτησης **digitalWrite(Pin, Value)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε τάση εξόδου, ενώ η τάση εξόδου μπορεί να είναι 0 V ή 5 V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην παράμετρο value

- LOW : θα δώσει 0 V στην έξοδο (pin)
- HIGH : θα δώσει 5 V στην έξοδο (pin)

Για παράδειγμα:

```
digitalWrite(ledPin, HIGH);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

6.4.2. Ψηφιακή είσοδος

Και τα 14 ψηφιακά pins του Arduino μπορούν να δουλεύουν ως ψηφιακές εισόδους, δηλαδή να “διαβάσουν” ως είσοδο τάση με τιμή είτε 0 είτε 5V. Αυτό γίνεται με χρήση της συνάρτησης **digitalRead(Pin)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τάση εισόδου μπορεί να είναι 0V ή 5V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην τιμή που διαβάζουμε:

- LOW : όταν λάβει τάση 0 V στην είσοδο (pin)
- HIGH : όταν λάβει τάση 5 V στην είσοδο (pin)

Για παράδειγμα:

```
Val = digitalRead(ledPin);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, INPUT);
```

6.4.3. Αναλογική έξοδος (PWM pins)

Κάποια από τα 14 Pins του Arduino έχουν την ένδειξη PWM, δηλαδή μπορούν να προσομοιώνουν την αναλογική έξοδο μέσω παλμοκοδικής διαμόρφωσης.

Έτσι, με τιμές από το 0 μέχρι το 255 προσομοιώνουμε (αναλογικά) το διάστημα από 0 έως 5V. Αυτό γίνεται με χρήση της συνάρτησης **analogWrite(Pin, Value)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε ρεύμα εξόδου, ενώ η τάση εξόδου κυμαίνεται από 0 V μέχρι και 5 V, οι οποίες τιμές της τάσης αναλογικά αναπαρίστανται με τιμές στη μεταβλητή value. Τιμή 0 δίνει 0V στην έξοδο (pin), τιμή 255 δίνει τάση 5V στην έξοδο (pin), ενώ

αναλογικά μπορούμε να δώσουμε ενδιάμεσες τάσεις (π.χ. 122 για τάση 2,5V).

Για παράδειγμα:

```
analogWrite(ledPin, 122);
```

Υπενθύμιση: Τη λειτουργία αυτή μπορούν να υποστηρίξουν μόνο τα PWM pins κι όχι όλα τα ψηφιακά. Τα PWM pins είναι τα 3, 5, 6, 9, 10, 11.

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

6.4.4. Αναλογική είσοδος

Το Arduino έχει 6 αναλογικές εισόδους, οι οποίες χαρακτηρίζονται με τα σύμβολα A0, A1, A2, A3, A4, A5. Μπορούμε να συνδέσουμε κάποιο αναλογικό εξάρτημα (π.χ. ένα ποτενσιόμετρο) και να το διαβάσουμε ως είσοδο. Αυτό γίνεται με χρήση της συνάρτησης **analogRead(Pin)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τιμή εισόδου κυμαίνεται από 0 μέχρι και 1023. Συνήθως χρησιμοποιούμε μια μεταβλητή για να καταχωρήσουμε την τιμή.

Για παράδειγμα:

```
int r = analogRead(A1);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(A1, INPUT);
```

6.5. Συναρτήσεις χρόνου

Στα περισσότερα προγράμματά μας θα χρειαστεί να διαχειριστούμε ή να καταγράψουμε τον χρόνο. Για το σκοπό αυτό υπάρχουν αντίστοιχες συναρτήσεις που μας βοηθούν.

6.5.1. Συνάρτηση καθυστέρησης – delay()

Στο πρόγραμμά μας μπορούμε να ορίσουμε μια καθυστέρηση ώστε να διαρκέσει για το χρόνο που εμείς ορίζουμε ένα γεγονός. Αυτό το επιτυγχάνουμε με χρήση της συνάρτησης delay(time) όπου στη θέση time δίνουμε το χρόνο σε ms (1/1000 sec). Η εντολή delay(time) σημαίνει ότι σταματά στο σημείο αυτό η εκτέλεση του προγράμματός μας για το χρόνο time.

Για παράδειγμα:

```
delay(1000); //σταματά την εκτέλεση του προγράμματος για 1000 ms = 1 sec  
delay(500); //σταματά την εκτέλεση στο σημείο αυτό για 500 ms = 0.5 sec
```

6.5.2. Συνάρτηση καθυστέρησης - delayMicroseconds()

Ακριβώς αντίστοιχα με την παραπάνω συνάρτηση delay, μπορούμε να χρησιμοποιήσουμε την delayMicroseconds(), όπου ο χρόνος καθυστέρησης δίνεται πλέον σε microseconds ($1/10^6$ sec).

Για παράδειγμα:

```
delayMicroseconds(10); // σταματά την εκτέλεση για 10μsec = 1/100000 sec
```

6.5.3. Συνάρτηση καταγραφής χρόνου - millis()

Το Arduino έχει ενσωματωμένο ρολόι, το οποίο μετράει το χρόνο από τη στιγμή που ενεργοποιείται (ή του γίνεται reset). Η πληροφορία αυτή μας είναι διαθέσιμη σε κάθε σημείο με κλήση της συνάρτησης millis(), η οποία μας επιστρέφει το χρόνο σε milliseconds ($1/1000$ sec) που έχει περάσει από την ενεργοποίηση της μονάδας μας. Αυτό μας βοηθά να μετράμε το χρόνο στα προγράμματά μας, ειδικά στις περιπτώσεις που θέλουμε να “θυμόμαστε” πράγματα.

Για παράδειγμα, αν θέλουμε να ελέγξουμε αν έχει περάσει συγκεκριμένο διάστημα από τότε που έγινε κάτι (π.χ. πατήθηκε τελευταία φορά ένα πλήκτρο), μπορούμε να καταγράφουμε το γεγονός σε μια μεταβλητή χρόνου και τον χρόνο αυτό να τον αφαιρούμε από τον επόμενο κτλ.

Για παράδειγμα:

```
lastPress = millis();  
if (lastPress - millis() > 1000) {...}
```

6.6. Συνάρτηση αντιστοίχισης τιμών - map()

Πολλές φορές θα χρειαστεί να αντιστοιχίσουμε μια τιμή που ανήκει σε ένα πεδίο τιμών σε μια άλλη τιμή που ανήκει σε ένα άλλο πεδίο τιμών. Η μαθηματική πράξη αυτή είναι σχετικά απλή, αλλά το Arduino μας παρέχει μια συνάρτηση για να το κάνει αυτό, την

```
map(<τιμή>, <κάτω_όριοA>, <πάνω_όριοA>, <κάτω_όριοB>, <πάνω_όριοB>)
```

όπου

<τιμή> είναι η τιμή που θέλουμε να μετατρέψουμε

<κάτω_όριοA> είναι το κάτω όριο του διαστήματος της αρχικής τιμής

<πάνω_όριοA> είναι το πάνω όριο του διαστήματος της αρχικής τιμής

<κάτω_όριοB> είναι το κάτω όριο του διαστήματος της τελικής τιμής (που θέλω να μετατραπεί)

<πάνω_όριοB> είναι το πάνω όριο του διαστήματος της τελικής τιμής (που θέλω να μετατραπεί)

Για παράδειγμα:

```
val = map(val, 0, 1023, 0, 179); /* μετατρέπω μια αναλογική τιμή που διάβασα από  
ένα ποτενσιόμετρο (0 έως 1023) σε μια τιμή για ένα σέρβο (0 έως 179) */
```

6.7. Η σειριακή θύρα επικοινωνίας (Serial)

Το Arduino παρέχει μια σειριακή θύρα επικοινωνίας μεταξύ της πλακέτας και του υπολογιστή ή κάποιας συσκευής που θέλουμε. Για το σκοπό αυτό χρησιμοποιείται η σύνδεση με καλώδιο USB (όταν πρόκειται για τον υπολογιστή) ή τα pins 0 και 1 όταν θέλουμε κάποια πιο εξειδικευμένη σύνδεση (π.χ. με κάποια άλλη συσκευή). Για το λόγο αυτό προτείνεται, αν δεν είναι απαραίτητο

στις εφαρμογές μας, να μην χρησιμοποιούνται τα pins αυτά.

Για να ενεργοποιήσουμε τη σειριακή θύρα επικοινωνίας αρκεί να δώσουμε στη διαδικασία setup() την εντολή Serial.begin(BaudRate), όπου το BaudRate εκφράζει το ρυθμό με τον οποίο θα μεταδίδονται τα bits (μια τιμή στα 9600 είναι συνήθως αρκετή).

Για παράδειγμα:

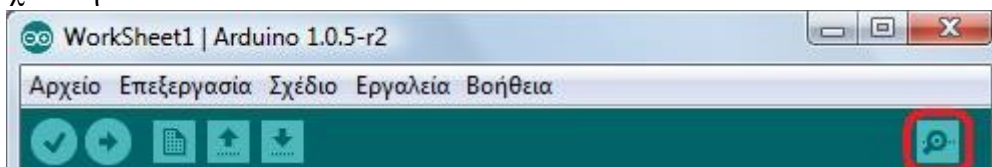
```
Serial.begin(9600);
```

Μπορούμε να χρησιμοποιήσουμε τη σειριακή θύρα στις εφαρμογές για αμφίδρομη επικοινωνία, δηλαδή να στείλουμε και να λάβουμε δεδομένα. Μία απλή περίπτωση χρήση της επικοινωνίας αυτής είναι για εκσφαλμάτωση (debugging) των προγραμμάτων μας, να μπορούμε δηλαδή να δούμε τι τιμές μας δίνουν μετρητές και τι τιμές έχουν οι μεταβλητές μας μέσω της οθόνης σειριακής επικοινωνίας. Μια εντολή που μας βοηθάει σε αυτό είναι η print(), που εκτυπώνει ένα μήνυμα ή τιμές ή η println() που λειτουργεί ακριβώς το ίδιο αλλά εκτυπώνοντας με αλλαγή γραμμής κάθε φορά.

Για παράδειγμα:

```
Serial.print("Η epikoinwnia ksekinhse"); /* Θα εμφανίσει το  
μήνυμα αυτό στην οθόνη χωρίς να αλλάξει γραμμή μετά */  
  
Serial.println(distance); /* Θα εμφανίσει την τιμή της μεταβλητής  
distance σε μια γραμμή */
```

Όταν έχετε συνδέσει το Arduino σας με τη θύρα USB στον υπολογιστή, η σειριακή οθόνη ενεργοποιείται από το εικονίδιο πάνω δεξιά “Σειριακή Οθόνη” (εικόνα 2), και στο παράθυρο που ανοίγει μπορείτε να βλέπετε όλα τα μηνύματα που στέλνονται από τον κώδικα που έχει φορτωθεί ήδη και τρέχει στην πλακέτα.



Εικόνα 2 – Σειριακή οθόνη

6.8. Δομή επιλογής

Στον προγραμματισμό πολλές φορές θα χρειαστεί να ελέγξουμε κάποια συνθήκη για να αποφασίσουμε αν θα εκτελεστεί ένα τμήμα κώδικα ή αν θα εκτελεστεί κάποιο άλλο αντί για αυτό στη θέση του. Αυτό το επιτυγχάνουμε με τη χρήση της δομής επιλογής, η οποία συντάσσεται

```
if <συνθήκη> { <εντολές 1> }  
else { <εντολές 2> }
```

όπου, στη <συνθήκη> έχουμε τον έλεγχο που θέλουμε να γίνει, συνήθως χρησιμοποιώντας τους τελεστές σύγκρισης (>, <, =, <=>, >=, <=), π.χ. potVal > 500. Η συνθήκη μπορεί να είναι και πιο σύνθετη, χρησιμοποιώντας τους λογικούς τελεστές (|| για το Η', && για το ΚΑΙ), π.χ. (potVal > 500) && (timePass >= 1000) .

Στα μπλοκ { <εντολές> } εκτελούνται αντίστοιχα οι εντολές που θέλουμε σε κάθε περίπτωση. Αν ισχύει η <συνθήκη> θα εκτελεστούν οι <εντολές 1>, αν δεν ισχύει οι <εντολές 2>. Σε κάθε περίπτωση, το τελευταίο κομμάτι else { <εντολές 2>} δεν είναι απαραίτητο να υπάρχει.

Τέλος, υπάρχουν πιο σύνθετες μορφές της εντολής επιλογής, οι οποίες ξεφεύγουν από το σκοπό του παρόντος εγχειριδίου – μπορείτε να τις συζητήσετε με τον καθηγητή σας όταν αυτές χρειαστούν.

6.9. Δομή επανάληψης (For)

Πολλές φορές θα χρειαστεί να επαναλάβουμε κάποια διαδικασία αρκετές φορές. Στην περίπτωση αυτή έχουμε εντολές οι οποίες επαναλαμβάνουν ένα σύνολο εντολών όσες φορές θέλουμε, είτε μετρώντας τις επαναλήψεις είτε ελέγχοντας κάθε φορά μία συνθήκη. Η συχνότερη μορφή που συναντάμε σε μια επανάληψη είναι αυτή με τον προκαθορισμένο αριθμό βημάτων. Η σύνταξη της εντολής αυτής είναι η εξής:

```
for (<αρχική τιμή>; <συνθήκη_τερματισμού>; <βήμα>)  
{ <εντολές> }
```

όπου χρησιμοποιείται μια μεταβλητή ελέγχου ως εξής

<αρχική τιμή> δίνουμε την αρχική τιμή, π.χ. $i = 0$

<βήμα> δίνουμε την αλλαγή κάθε επανάληψης, π.χ. $i+5$ (το $i++$ που θα δείτε σημαίνει $i+1$)

<συνθήκη_τερματισμού> η συνθήκη για να τελειώσει η επανάληψη, π.χ. $i < 10$ (όσο ισχύει αυτή θα τρέχει)

π.χ.

```
for (i=1; i<10; i=i+1) {  
    brightness = brightness + 5;  
    analogWrite(ledPin, brightness);  
};
```

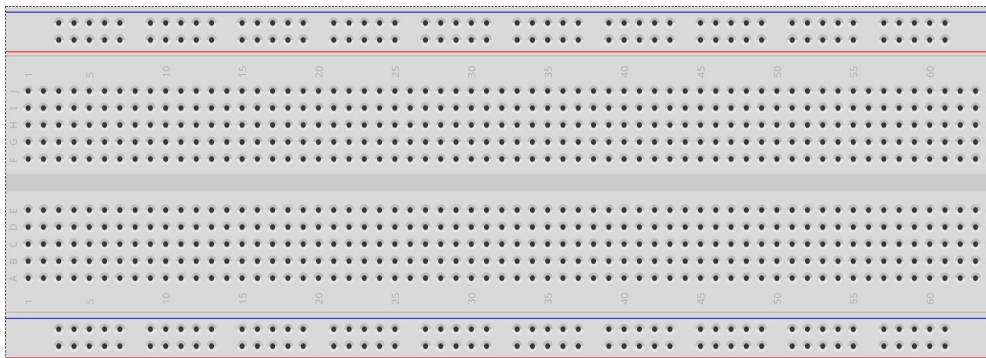
Υπάρχουν εντολές επανάληψης που δεν έχουν προκαθορισμένο αριθμό βημάτων, αλλά συνεχίζουν επ' αόριστο ελέγχοντας μια συνθήκη. Στα φύλλα εργασίας που ακολουθούν δεν θα χρειαστεί να χρησιμοποιήσετε μια τέτοια εντολή, αλλά για λόγους αναφοράς έχουμε τις παρακάτω εντολές:

- while <συνθήκη> { <εντολές> } //όσο ισχύει η <συνθήκη> τρέχουν οι εντολές
- repeat {<εντολές>} until <συνθήκη> // οι <εντολές> τρέχουν όσο δεν ισχύει η συνθήκη

7. Χρήση breadboard για συνδέσεις και βραχυκυκλώσεις

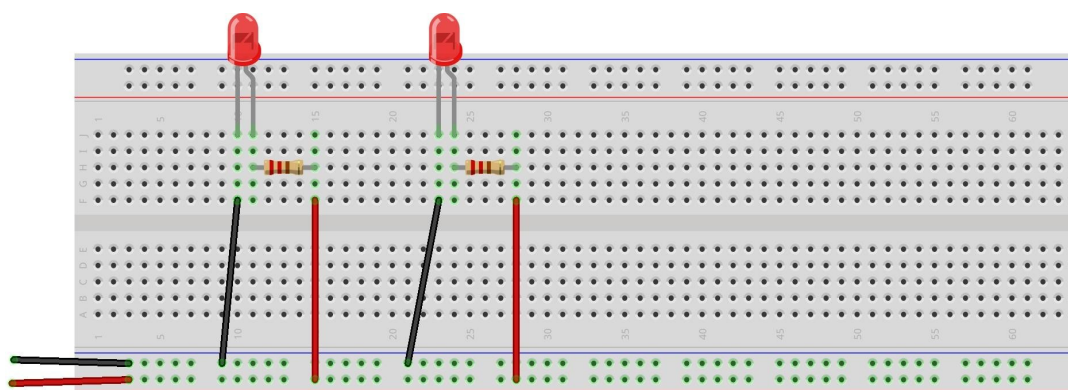
Επειδή τα κυκλώματα συνήθως περιλαμβάνουν αρκετά στοιχεία (καλώδια, leds, αισθητήρες, αντιστάσεις κτλ) προτείνεται να χρησιμοποιείται κάποια breadboard ώστε άνετα να συνδέονται μέσω αυτής τα στοιχεία. Επίσης, λειτουργικά η breadboard μπορεί να βραχυκυκλώνει μεταξύ τους καλώδια, κάτι που είναι πολύ χρήσιμο όταν έχουμε πολλά καλώδια για βραχυκύκλωση – διαφορετικά άλλωστε δεν υπάρχουν τόσες ελεύθερες είσοδοι ή έξοδοι στην πλακέτα μας. Στην εικόνα 3 βλέπετε μια τυπική breadboard.

Προσέξτε ότι οι οριζόντιες γραμμές + και - σε κάθε μεριά (πάνω και κάτω, με κόκκινο και μπλε χρώμα όπως την βλέπουμε) είναι βραχυκυκλωμένες μεταξύ τους, ενώ στις στήλες (που είναι συνήθως αριθμημένες από το 1 μέχρι το 30) είναι βραχυκυκλωμένες οι πέντε κάθετες υποδοχές (συνήθως με γράμματα a, b, c, d, e καθώς και f, g, h, i, j) μεταξύ τους σε κάθε στήλη όπως κοιτάμε. Έτσι, για παράδειγμα μπορούμε να συνδέσουμε στο - το GND του Arduino και στο breadboard να συνδέουμε όλες τις επιστροφές GND των κυκλωμάτων μας.



Εικόνα 3 – Breadboard

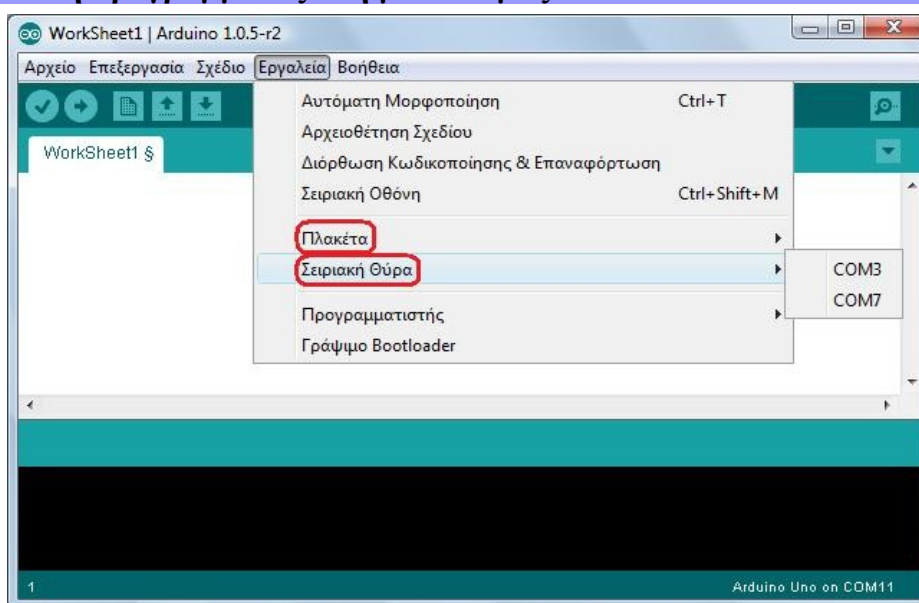
Επίσης, μέσω των βραχυκυκλωμένων γραμμών (5 σε κάθε στήλη όπως κοιτάμε), μπορούμε να συνδέουμε τα μέρη του κυκλώματός μας. Στην εικόνα 4 βλέπετε ένα απλό παράδειγμα συνδεσμολογίας.



Εικόνα 4 – Παράδειγμα χρήσης Breadboard

Αναλυτικότερη πληροφόρηση μπορείτε να βρείτε και στον ιστότοπο του SparkFun.Com, στα αγγλικά (<https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard>).

8. Μεταφόρτωση προγράμματος στη μονάδα μας

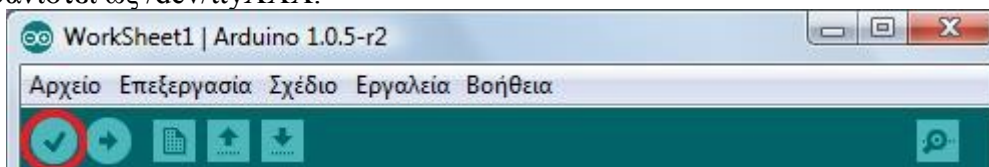


Εικόνα 5 – Επιλογή πλακέτας και σειριακής θύρας

Για να μεταφορτώσουμε το πρόγραμμά μας στη μονάδα θα πρέπει να τη συνδέσουμε με ένα USB καλώδιο στον υπολογιστή. Ο υπολογιστής μας θα αναγνωρίσει τη μονάδα Arduino μας ως σειριακή θύρα, κάτι που μπορείτε να επιβεβαιώσετε και από τον πίνακα ελέγχου του H/Y σας.

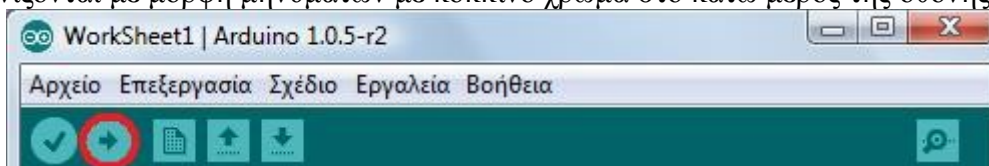
Από το μενού Εργαλεία του Arduino (εικόνα 5) επιλέγουμε δύο πράγματα:

- **Πλακέτα** - διαλέγουμε τον τύπο της μονάδας μας. π.χ. Arduino Uno
- **Σειριακή θύρα** - είναι η σειριακή θύρα που έχει αντιστοιχίσει το λειτουργικό σας στην πλακέτα Arduino που συνδέεται μέσω του USB καλωδίου. Αν χρησιμοποιείτε Windows αυτή θα είναι της μορφής COMX (π.χ. COM3, COM11), ενώ στο Linux η θύρα θα εμφανιστεί ως /dev/ttyXXX.



Εικόνα 6 – Μεταγλώττιση

Ακολούθως, πατάμε το πλήκτρο της μεταγλώττισης (εικόνα 6) το οποίο θα ελέγξει το πρόγραμμά μας για λάθη και θα το προετοιμάσει για τη μεταφόρτωση στην πλακέτα. Αν τυχόν υπάρξουν λάθη, αυτά εμφανίζονται με μορφή μηνυμάτων με κόκκινο χρώμα στο κάτω μέρος της οθόνης.



Εικόνα 7 – Μεταφόρτωση

Τέλος, εφόσον έχουμε επιτυχώς εκτελέσει όλα τα παραπάνω, δηλαδή έχουμε συνδέσει τη μονάδα μας, έχουμε επιλέξει τον τύπο της και τη θύρα που είναι συνδεδεμένη, έχουμε γράψει κάποιο πρόγραμμα και το έχουμε μεταγλωττίσει χωρίς λάθη, μπορούμε πατώντας το πλήκτρο της φόρτωσης (εικόνα 7) να μεταφορτώσουμε το πρόγραμμα πλέον στη μονάδα και αυτό να αρχίσει να τρέχει πλέον σε πραγματικό περιβάλλον. Η διαδικασία της μεταγλώττισης επαναλαμβάνεται αυτόματα στο βήμα αυτό, όπως θα δείτε.

9. Βασική συνδεσμολογία για τα κυριότερα υλικά που θα χρησιμοποιήσουμε

Στο σημείο αυτό θα περιγράψουμε σύντομα το πώς λειτουργούν τα κυριότερα από τα υλικά που θα χρειαστούμε στα φύλλα εργασίας που ακολουθούν στην επόμενη ενότητα.

Προτείνεται η ενότητα αυτή να χρησιμοποιηθεί ως εγχειρίδιο αναφοράς, δηλαδή να ανατρέχει ο αναγνώστης εδώ στην αντίστοιχη υποενότητα, ανάλογα με το φύλλο εργασίας που καλείται να υλοποιήσει κι όχι να προσπαθήσει να αποστηθίσει όλη την πληροφορία που παρουσιάζεται εδώ.

Ο αναγνώστης άλλωστε εισάγεται σταδιακά στα διαθέσιμα υλικά, μέσω των φύλλων εργασίας που προσεγγίζουν επίσης βήμα βήμα διάφορες δυνατές λειτουργίες. Η ύπαρξη του θεωρητικού πλαισίου σε ένα μέρος διευκολύνει την αναφορά σε αυτά από τα διάφορα φύλλα εργασίας που τα χρησιμοποιούν.

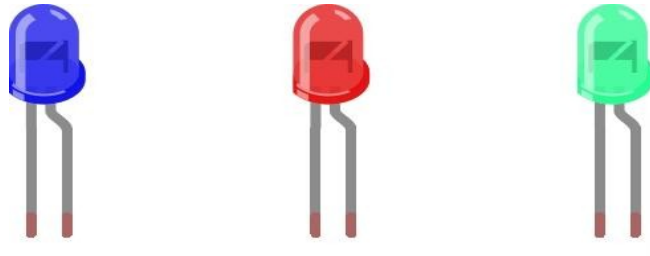
9.1. Φωτοдиодοι - Leds

Οι φωτοдиодοι, ή τα leds όπως έχουν κυριαρχήσει, υπάρχουν σε διάφορα χρώματα και τάσεις λειτουργίας (εικ. 8). Ανάλογα με την προβλεπόμενη τάση λειτουργίας του led που έχουμε πρέπει να χρησιμοποιήσουμε μαζί και την κατάλληλη αντίσταση, ώστε να αποφύγουμε κάποια καταστροφή

από υπέρταση. Για να βρούμε την αντίσταση που χρειαζόμαστε αρκεί να θυμηθούμε τον τύπο

$$R = (V_{\pi} - V_{\lambda}) / I$$

όπου R είναι η αντίσταση που χρειαζόμαστε, V_{π} η παρεχόμενη τάση από την πηγή μας (5V από το Arduino), V_{λ} η τάση λειτουργίας του led και I το ρεύμα λειτουργίας του led. Τυπικά, μια αντίσταση γύρω στα 220 Ω καλύπτει τα περισσότερα led που έχουμε χρησιμοποιήσει.



Εικόνα 8 – Leds σε διάφορα χρώματα

Επίσης, τα leds έχουν πολικότητα, δηλαδή δουλεύουν μόνο αν συνδεθούν στην κατάλληλη φορά ρεύματος. Συνήθως το πόδι που πρέπει να συνδεθεί στη θετική κατεύθυνση (+) είναι πιο μακρύ από το αντίστοιχο για την αρνητική φορά (-). Επίσης, το λαμπάκι από την αρνητική μεριά (-) είναι συνήθως επίπεδο κι όχι στρογγυλό όπως είναι από το άλλο πόδι (+). Μην ανησυχείτε όμως, μια ανάποδη σύνδεση δεν θα το καταστρέψει, απλά δεν θα ανάψει στην ανάποδη φορά (εκτός από την ιδιαίτερη περίπτωση να έχετε δώσει πολύ μεγάλη τάση).

9.1.1. Έγχρωμοι φωτοдиодοι - RGB Leds

Εκτός από τα απλά leds, υπάρχουν και τα RGB leds (εικόνα 9), τα οποία μπορούν να εμφανίσουν οποιοδήποτε χρώμα, βασισμένα στο σύστημα RGB. Μπορούμε να τα σκεφτούμε ως τρία led (κόκκινο - Red, πράσινο - Green, μπλε - Blue) σε ένα, με τα οποία δίνοντας χωριστά φωτεινότητα σε κάθε ένα μπορούμε να συνδυάσουμε χρώματα και να παράγουμε οποιοδήποτε χρωματισμό, όπως ακριβώς συμβαίνει στις ψηφιακές οθόνες.



Εικόνα 9 – RGB led

Τα RGB leds έχουν τέσσερα σημεία σύνδεσης – ένα για κάθε χρώμα (τρία σύνολο) κι ένα για την άνοδο ή κάθοδο. Υπάρχουν δύο τύπου, ανόδου (+) και καθόδου (-), ανάλογα με τη σύνδεση που ακολουθούν. Τα καθόδου, που χρησιμοποιούμε στα παραδείγματά μας, συνδέονται με την κάθοδο (-), δηλαδή τη γείωση (GND). Έτσι, συνδέουμε από ένα Pin που θα ελέγχει την τάση που θα δοθεί σε κάθε χρώμα και τη γείωση του Arduino, καταφέρνοντας να έχουμε απόλυτο έλεγχο σε όλους τους δυνατούς χρωματισμούς.

Προσοχή: Θα πρέπει να συνδέσουμε αντιστάσεις σε κάθε χρώμα, καθώς δουλεύουν με μικρότερη τάση από αυτή που δίνει ως έξοδο το Arduino, όπως φαίνεται και στο σχήμα. Για τον υπολογισμό τους ανατρέξτε στην προηγούμενη ενότητα. Θα πρέπει να γνωρίζετε τί είδους led χρησιμοποιείτε - τυπικά μια αντίσταση 150-180Ω για το κόκκινο και 75-100Ω για το πράσινο και μπλε θα είναι αρκετές. Αν δεν έχετε ακριβώς τις τιμές αυτές για τις αντιστάσεις, χρησιμοποιήστε μεγαλύτερες

(αυτό ισχύει σε όλες τις περιπτώσεις που χρησιμοποιούμε αντίσταση για την προστασία από υπέρταση – προτιμούμε να έχουμε μικρότερη τάση παρά μια καταστροφή λόγω υπέρτασης).

Προσέξτε για τη συνδεσμολογία ότι η γείωση συνδέεται στο πιο μακρύ πόδι του led (2ο από αριστερά στην εικόνα 9), το pin για το κόκκινο στο διπλανό εξωτερικό του (1ο από αριστερά), το pin για το πράσινο στο διπλανό εσωτερικό του (3ο από αριστερά) και το pin για το μπλε στο άλλο εξωτερικό του (4ο από αριστερά).

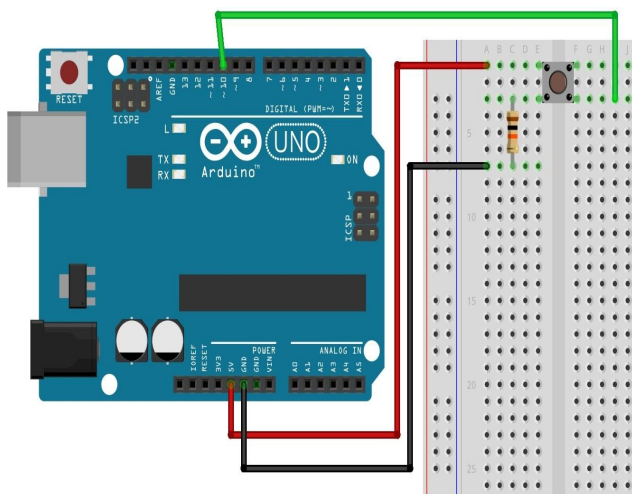
9.2. Κουμπιά - Buttons

Στα κυκλώματά μας θα χρειαστεί πολλές φορές να χρησιμοποιήσουμε κουμπιά (εικόνα 10) ώστε να μπορούμε να παρέμβουμε στο κύκλωμα όταν εμείς το θέλουμε, π.χ. για να δώσουμε ρεύμα σε ένα τμήμα του ή να διακόψουμε την τροφοδότηση με ρεύμα ενός άλλου τμήματος.

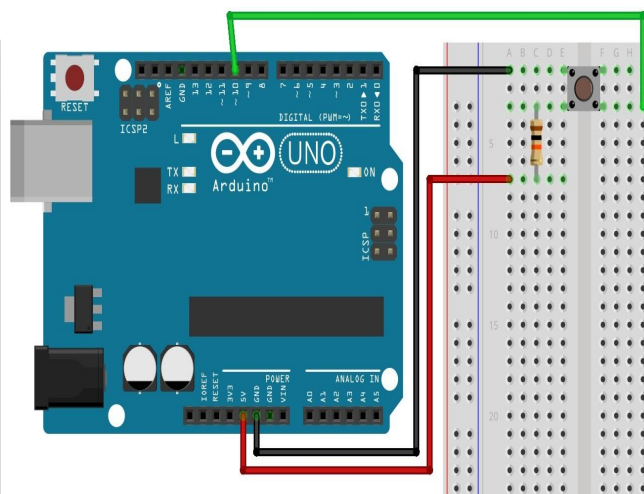


Εικόνα 10 – Κουμπί (button)

Στα σχήματα που ακολουθούν μπορείτε να δείτε δύο κυκλώματα με κουμπιά. Το ένα αφήνει το ρεύμα να περνάει όταν πατηθεί το κουμπί (δηλαδή είναι πάντα στο LOW το input pin κι όταν πατηθεί το πλήκτρο γίνεται HIGH) – αυτό ονομάζεται pull-down resistor και το βλέπουμε στην εικόνα 11.



Εικόνα 11 – Pull-down resistor



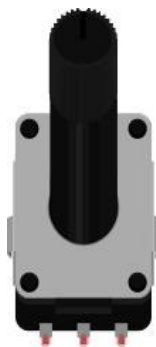
Εικόνα 12 – Pull-up resistor

Το άλλο διακόπτει τη ροή του ρεύματος όταν πατηθεί (δηλαδή είναι πάντα στο HIGH το input pin κι όταν πατηθεί το πλήκτρο γίνεται LOW – αυτό ονομάζεται pull-up resistor και το βλέπουμε στην εικόνα 12 αντίστοιχα.

Υπάρχει ένας κανόνας που λέει ότι το ρεύμα θα ακολουθήσει τη διαδρομή με τη μικρότερη αντίσταση. Το κουμπί έχει μια μικρή εσωτερική αντίσταση μέσα του. Έτσι, στο κύκλωμα της εικόνας 12 (pull-up) όταν πατηθεί το κουμπί και κλείσει το κύκλωμα, έχουμε ροή από τα 5V προς τη γείωση (μικρότερη αντίσταση), οπότε διακοπή του αρχικού κυκλώματος (5V – Input). Στο κύκλωμα της εικόνας 11 (pull-down), όταν πατηθεί το κουμπί, τότε έχουμε ροή από την πηγή προς την είσοδο (μικρότερη αντίσταση), δηλαδή ενεργοποίηση του κυκλώματος (5V – Input). Ανάλογα λοιπόν τι θέλουμε να κάνουμε στα κυκλώματά μας, ακολουθούμε την αντίστοιχη συνδεσμολογία.

9.3. Ποτενσιόμετρο - Potentiometer

Ένα ποτενσιόμετρο μπορεί να περιγραφεί ως μια διάταξη με κύκλωμα μεταβλητής αντίστασης. Γυρίζοντας το χειριστήριο που έχει, αυξάνει ή μειώνει την αντίστασή του, οπότε περνάει λιγότερο ή περισσότερο ρεύμα από αυτό (εικόνα 13).



Εικόνα 13 – Ποτενσιόμετρο (potentiometer)

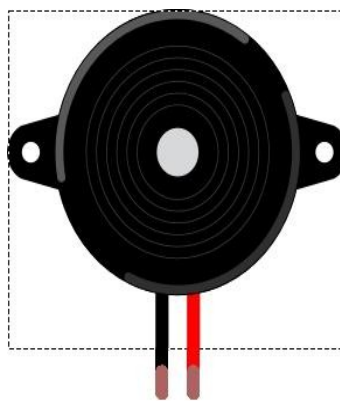
Έχει τρία σημεία σύνδεσης – ένα πόδι συνδέεται στην πηγή (5V), ένα στη γείωση (GND) και το μεσαίο στο κύκλωμα που θέλουμε να πάρει τη μεταβλητή τιμή ρεύματος (αναλογικό pin στο Arduino). Το ποτενσιόμετρο δηλαδή δουλεύει ως ένας διαιρέτης τάσης, ο οποίος όμως μεταβάλλεται κάθε φορά που γυρίζουμε το χειριστήριό του. Ποτενσιόμετρα χρησιμοποιούνται σε πολλές εφαρμογές, όπως στην αυξομείωση του ήχου στα στερεοφωνικά ή την αυξομείωση της έντασης μιας λάμπας.

9.4. Ηχείο – Sounder

Ένα επίσης απλό εξάρτημα που μπορούμε να συνδέσουμε με το Arduino μας είναι ένα ηχείο. Δουλεύει ακριβώς όπως ένα led, με δύο καλώδια, ένα για την πηγή (5V) κι ένα για τη γείωση (GND), όπως βλέπετε και στην εικόνα 14. Αν του δώσουμε σταθερή τάση (π.χ. 5V) θα μας δώσει σταθερό ήχο, ενώ μεταβλητή τάση (π.χ. αντί στην πηγή (5V), σύνδεσή του με ένα PWM pin στο Arduino και αυξομείωση της τάσης που του παρέχεται) θα μας δώσει μεταβλητό ήχο, δηλαδή ηχητικό εφέ.



Εικόνα 14 – Ηχείο (απλό)



Εικόνα 15 – Πιεζοηλεκτρικό ηχείο

Εκτός από τα απλά ηχεία υπάρχουν και διατάξεις ηχείων που αλλάζουν τις ιδιότητές τους ανάλογα με την πίεση που τους ασκείται (εικόνα 15). Η συνδεσμολογία τους είναι ίδια και μπορούμε να τα χρησιμοποιήσουμε για να ανιχνεύσουμε ασκούμενη δύναμη πάνω τους.

9.5. Φωτοευαίσθητη αντίσταση – Photoresistor (LDR)

Υπάρχουν αντιστάσεις για τις οποίες αλλάζουν οι ιδιότητές τους ανάλογα με το πόσο φως πέφτει πάνω τους (εικόνα 16). Η αντίσταση τους μικραίνει με το φως (όταν βρίσκονται σε περιβάλλον μεγαλύτερης φωτεινότητας) και μεγαλώνει όσο πλησιάζουν σε πιο σκοτεινό περιβάλλον. Έτσι, αν συνδεθούν σε ένα κύκλωμα με σταθερή τάση (π.χ. 5V – GND του Arduino) το ρεύμα που τις διαρρέει θα είναι μεταβαλλόμενο αν είναι μεταβλητή και η φωτεινότητα που πέφτει πάνω τους.



Εικόνα 16 – Φωτοευαίσθητη αντίσταση (LDR)

Ένα κύκλωμα που βασίζεται σε μια τέτοια αντίσταση μπορεί να “διαβάσει” τη φωτεινότητα και να δουλέψει αντίστοιχα με τη μεταβολή της (π.χ. ανάβει σταδιακά ένα led όσο πέφτει η φωτεινότητα). Φωτοευαίσθητες αντιστάσεις χρησιμοποιούνται στα κυκλώματα που ανιχνεύουν την κίνηση σε κάποιο χώρο και ανάβουν αυτόματα κάποιο φως.

9.6. Σέρβος - Servos

Ένα ενδιαφέρον εξάρτημα που μπορούμε να διαχειριστούμε μέσω του Arduino είναι το σέρβο (εικόνα 17), και ειδικότερα θα χρησιμοποιήσουμε το TowerPro SG90. Πρόκειται για μια διάταξη που μπορεί να γυρίζει έναν άξονα από τις 0 μέχρι τις 180 μοίρες. Η διάταξη αυτή έχει εφαρμογή σε κατασκευές που θέλουμε να κινείται κάποιο μέρος ελεγχόμενο. Αν σταθεροποιήσουμε κάπου τη βάση του, μπορούμε να χρησιμοποιήσουμε το σέρβο για να εισάγουμε κίνηση σε εύρος 180 μοιρών στην κατασκευή μας. Το σέρβο συνδέεται με ένα καλώδιο στην πηγή, ένα στη γείωση και το τρίτο σε ένα Pin του Arduino ώστε να μπορούμε να δίνουμε εντολές για το πώς και πόσο θα στρίψει.



Εικόνα 17 – Σέρβο (servo)

Χρειάζεται να εισάγουμε την αντίστοιχη βιβλιοθήκη στο πρόγραμμά μας για να δουλέψει (`#include <Servo.h>`), να ορίσουμε μια μεταβλητή τύπου servo (`servo myservo;`) και το pin μέσω του οποίου θα το χειριστούμε.

```
#include <Servo.h>
int servoPin = 13;
Servo myservo;
```

Αντί να ορίσουμε το servoPin ως εξόδου στη συνάρτηση setup, το συσχετίζουμε με το σέρβο μας στη συνάρτηση setup() με την εντολή

```
myservo.attach(servoPin);
```

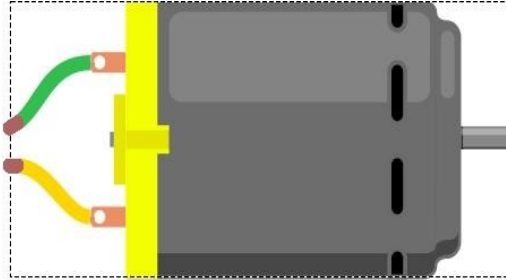
Το σέρβο έχει τρία καλώδια – ένα (πορτοκαλί) συνδέεται στην πηγή (5V), ένα (καφέ) στη γείωση (GND) κι ένα (κίτρινο) στο pin μέσω του οποίου του στέλνουμε είσοδο (εντολές), χρησιμοποιώντας την εντολή


```
servo.write(num);
```

όπου το num είναι ένας αριθμός από 0 έως 179, αντιπροσωπεύοντας τις 180 μοίρες στρέψης του.

9.7. Κινητήρας συνεχούς – DC Motor

Ένα χρήσιμο εξάρτημα για τις κατασκευές μας είναι οι κινητήρες που δουλεύουν με συνεχές ρεύμα (DC motors). Θα τους βρείτε σε διάφορες μορφές και χαρακτηριστικά (εικόνα 18).

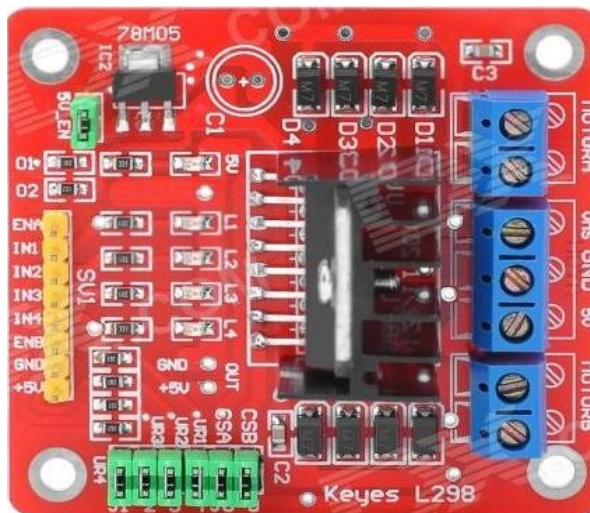


Εικόνα 18 – Κινητήρας συνεχούς (DC motor)

Η συνδεσμολογία τους είναι απλή, χρειάζονται δύο καλώδια, ένα για την πηγή και ένα για τη γείωση, ενώ ανάλογα το πώς θα συνδεθούν θα είναι και η στρέψη που θα δίνουν. Οπότε με μια δοκιμή θα βρούμε την επιθυμητή φορά, ενώ δεν κινδυνεύουμε να καεί κάτι από λάθος συνδεσμολογία.

9.7.1. Χρήση δύο κινητήρων

Σε πολλές εφαρμογές θέλουμε να χρησιμοποιήσουμε δύο κινητήρες ταυτόχρονα. Για να τους διαχειριστούμε χρησιμοποιούμε H-γέφυρες (H-bridges) και μία αρκετά διαδεδομένη και χρήσιμη διάταξη είναι το L298N (εικόνα 19), η οποία χρησιμοποιεί δύο H-γέφυρες. Μια πολλή καλή ανάλυση μπορείτε να βρείτε στο <http://adhocnode.com/motor-control/>.



Εικόνα 19 – Διάταξη L298N (δύο κινητήρων)

Για τη σύνδεση των κινητήρων χρειαζόμαστε τρία pins για τον καθένα, συνολικά έξι δηλαδή. Έχουμε τα pin ENA, ENB (ενεργοποίηση του A και B κινητήρα αντίστοιχα), IN1, IN2 για το χειρισμό του A κινητήρα και IN3, IN4 για το χειρισμό του B κινητήρα. Επίσης, υπάρχουν pins για την πηγή (5V) και τη γείωση (GND), ενώ η διάταξη υποστηρίζει εξωτερική πηγή ρεύματος

(προτείνεται) την οποία συνδέουμε στο GND και VMS στην απέναντι πλευρά. Τέλος, δίπλα στην εξωτερική παροχή ρεύματος, έχουμε δύο υποδοχές για κάθε κινητήρα, τις MOTOR A και MOTOR B αντίστοιχα. Εκεί συνδέουμε τους κινητήρες μας – αν θέλουμε μεγαλύτερη δύναμη μπορούμε να βραχυκυκλώσουμε εκεί μέσα δύο κινητήρες αριστερούς και δύο κινητήρες δεξιούς, αντί για έναν κι έναν.

Η φορά της κίνησης γίνεται από τα pin του κάθε κινητήρα κι ο έλεγχος (κι η ταχύτητα) με τα EN.

- IN1 (ή IN3) <-- HIGH, IN2 (ή IN4) <-- LOW κίνηση προς τα εμπρός
- IN1 (ή IN3) <-- LOW, IN2 (ή IN4) <-- HIGH κίνηση προς τα πίσω
- IN1 (ή IN3) <-- LOW, IN2 (ή IN4) <-- LOW σταμάτημα κίνησης
- ENA <-- HIGH για να μπορώ να ελέγξω τα pins IN1, IN2
- ENB <-- HIGH για να μπορώ να ελέγξω τα pins IN3, IN4

9.8. Αισθητήρας υπερήχων – Ultrasonic sensor

Ο αισθητήρας υπερήχων χρησιμοποιεί δύο διατάξεις, μία για να στείλει ένα υπερηχητικό σήμα κι έναν για να το λάβει. Έτσι, από το χρόνο που μεσολαβεί από το να στείλει μέχρι να λάβει το σήμα πίσω μπορούμε να υπολογίσουμε την απόσταση στην οποία μπροστά μας βρίσκεται κάποιο αντικείμενο. Στην εικόνα 20 βλέπουμε έναν κλασσικό τέτοιο αισθητήρα, τον HC-SR04 Ultrasonic Distance Sensor.



Εικόνα 20 – Αισθητήρας υπερήχων HC-SR04 (ultrasonic distance sensor)

Για τη συνδεσμολογία του χρησιμοποιούμε ένα καλώδιο για την πηγή (Vcc – 5V), ένα για τη γείωση (GND), ένα pin για να στέλνουμε σήμα (trigger) κι ένα pin για να λαμβάνουμε το σήμα που επιστρέφει (Echo). Το Pin για το trigger θα οριστεί ως εξόδου, ενώ το Pin για το echo ως εισόδου στη συνάρτηση setup(). Η αποστολή σήματος γίνεται δίνοντας τάση στο trigger

```
digitalWrite(trigPin, HIGH); //δίνω τάση, αρχίζω να στέλνω σήμα  
delayMicroseconds(10); //αφήνω λίγο χρόνο για την αποστολή  
digitalWrite(trigPin, LOW); //μηδενίζω την τάση, σταματά η αποστολή σήματος
```

ενώ παίρνουμε το χρόνο που πέρασε μέχρι να επιστρέψει το σήμα στο echo με την εντολή

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

Ο υπολογισμός της απόστασης (σε cm) γίνεται βασιζόμενοι στην ταχύτητα του ήχου, δηλαδή

```
distance = duration/58.138;            ή            distance = duration/58.2;
```

Η τιμή 58.138 έρχεται από τον κατασκευαστή και προκύπτει από την ταχύτητα του ήχου. Επειδή οι μετρήσεις είναι εμπειρικές, λεπτομερής παρατήρηση θα μας δώσει ίσως λεπτομερέστερες τιμές – για παράδειγμα άλλη εφαρμογή με τον ίδιο αισθητήρα προτείνει τη διαίρεση με 58.2 (στην πρώτη περίπτωση έχουμε trigger (αποστολή) για 2μs και στη δεύτερη για 10μs. Γενικότερα, ο τύπος είναι

$$\text{distance} = ((\text{διάρκεια παλμού HIGH}) * (\text{ταχύτητα ήχου: } 340\text{m/s})) / 2$$

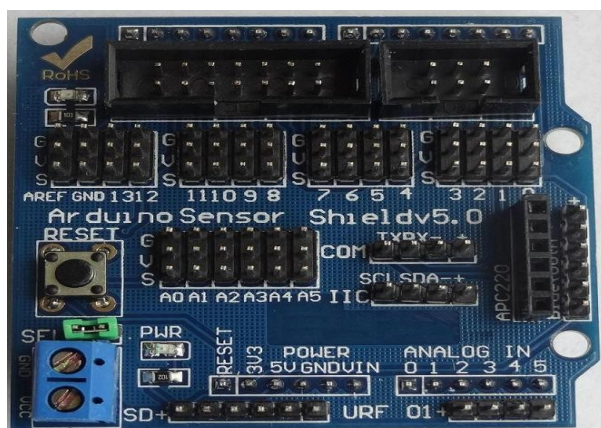
Διαιρούμε με το 2 επειδή το σήμα μας πήγε και ήρθε, δηλαδή έχει διανύσει διπλάσια απόσταση. Η διάρκεια του παλμού έχει δοθεί από την pulseIn σε microseconds (10^{-6} sec). Όπως αναφέρεται και στο <http://arduino.cc/en/Reference/pulseIn> όλες αυτές οι μετρήσεις είναι εμπειρικές, οπότε έχουμε αποκλίσεις.

Μια κλασσική διάταξη είναι να τοποθετήσουμε τον αισθητήρα υπερήχων πάνω σε ένα σέρβο, έτσι ώστε να μπορούμε με τις κατάλληλες εντολές να τον κινούμε δεξιά (σέρβο στις 0 μοίρες), αριστερά (σέρβο στις 180 μοίρες) και ευθεία (σέρβο στις 90 μοίρες).

9.9. Ασπίδες - Shields

Το Arduino μπορεί να επεκταθεί με πλακέτες που ονομάζουμε ασπίδες (shields) και οι οποίες έχουν ενσωματωμένα κυκλώματα ώστε να επεκτείνουν με τα υπάρχοντα pins τη λειτουργία του. Για παράδειγμα, υπάρχει WiFi Shield, BlueTooth Shield κτλ, δίνοντας στο Arduino τη λειτουργικότητα που λέει και το όνομά τους. Τις ασπίδες αυτές μπορούμε να χρησιμοποιήσουμε σε επόμενο στάδιο, έχοντας δηλαδή κατανοήσει τις βασικές λειτουργίες και έχοντας φτάσει σε ένα ικανοποιητικό στάδιο χειρισμού και προγραμματισμού, οπότε δεν θα τις αναλύσουμε εδώ περισσότερο.

Η μόνη ασπίδα με την οποία θα ασχοληθούμε εδώ, είναι η ArduinoSensor Shield (εικόνα 21) η οποία μας επεκτείνει τα pin που ήδη υπάρχουν σε τριάδες της μορφής (Ground, Voltage, Pin) ώστε να μπορώ για κάθε pin (είτε αναλογικό είτε ψηφιακό) να έχω μαζί και την πηγή (5V) και τη γείωση (GND).



Εικόνα 21 – Arduino Sensor Shield v.5.0

10. Φύλλα εργασίας

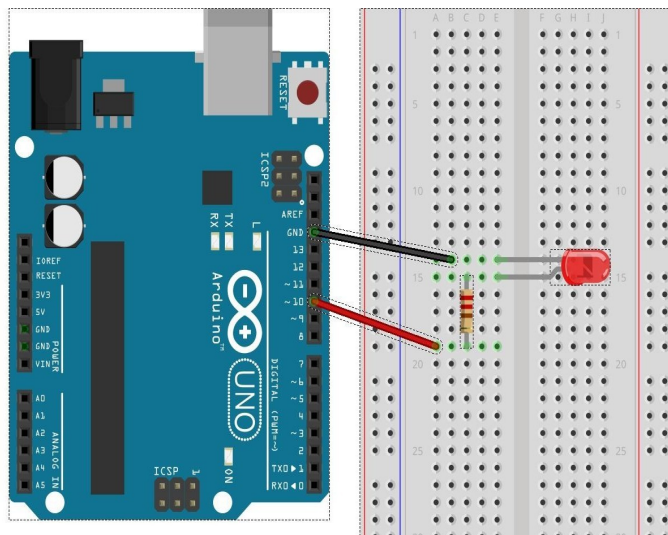
Ακολουθούν φύλλα εργασίας που υλοποιούν βασικές διατάξεις στο περιβάλλον αυτό, με σκοπό οι μαθητές σταδιακά να εξοικειωθούν με την πλατφόρμα και τα υλικά, ώστε από τα απλά να μπορούν να προχωρήσουν σε πιο σύνθετες κατασκευές. Σταδιακά εισάγονται νέες έννοιες στα φύλλα εργασίας, ώστε η γνώση να οικοδομείται πάνω στην προϋπάρχουσα κι ο μαθητής χρησιμοποιεί όλο και περισσότερες διατάξεις. Τελικά, προτείνεται η υλοποίηση κάποιας σύνθετης διάταξης κι οι μαθητές κατανοούν ότι όσο σύνθετη και να είναι μια υλοποίηση, αυτή αποτελείται πάντα από απλούστερες υλοποιήσεις – τμήματα, όπως ακριβώς συμβαίνει και στον προγραμματισμό – όσο σύνθετο κι αν είναι ένα πρόβλημα, αναλύεται σε μικρότερα προβλήματα και τελικά η λύση των επιμέρους τμημάτων μας δίνει και λύση στο συνολικό μας πρόβλημα – το τελικό μας πρόγραμμα.

10.1. Φύλλο εργασίας 1 – Το led που αναβοσβήνει

Σε αυτή την πρώτη άσκηση θα προσπαθήσουμε να κάνουμε ένα led να αναβοσβήνει ανά ένα δευτερόλεπτο, δηλαδή να δέχεται ρεύμα για ένα δευτερόλεπτο, να μην δέχεται ρεύμα για ένα δευτερόλεπτο κι αυτό να επαναλαμβάνεται επ' αόριστο.

Περιγραφή κυκλώματος

Για το κύκλωμα θα χρειαστούμε ένα Led και μια αντίσταση ώστε το ρεύμα των 5V που θα περνάει να αντιστοιχεί στην τάση λειτουργίας του Led που χρησιμοποιούμε και να μην έχουμε κάποιο πρόβλημα (π.χ. καταστροφή του Led από υπέρταση). Μπορείτε να δείτε το κύκλωμα στην εικόνα 22 που ακολουθεί.



Εικόνα 22 – Κύκλωμα φύλλου εργασίας 1

Η έξοδος του ρεύματος στο συγκεκριμένο σχήμα είναι το Pin 10, ενώ αν προσέξουμε η αντίσταση συνδέεται ανάμεσα στην έξοδο και στο Led ώστε να μην περάσει μεγάλη ποσότητα ρεύματος από αυτό. Ακολουθώς συνδέεται το Led. Συνήθως τα led έχουν ένα μακρύτερο πόδι που μπαίνει από τη φορά +, δηλαδή το Pin 10 στην περίπτωση μας. Τέλος, το κύκλωμα κλείνει με το άλλο πόδι του led στην υποδοχή GND που συμβολίζει τη φορά – για το κύκλωμά μας.

Προγραμματισμός κυκλώματος

Το μόνο που μένει είναι να προγραμματίσουμε το κύκλωμά μας, σύμφωνα με αυτά που έχουμε ήδη αναφέρει.

Για ευκολία, σας δίνονται οδηγίες με τη μορφή βημάτων:

1. Δηλώστε μια μεταβλητή που θα αντιπροσωπεύει το Pin σας και δώστε της τιμή 10.
2. Θα πρέπει να ορίσετε το Pin10 ως Εξόδο.
3. Θα πρέπει να δώσουμε τάση 5V στο Pin10.
4. Θα πρέπει να περιμένουμε 1 δευτερόλεπτο κατά τη διάρκεια του οποίου προφανώς ανάβει το led.
5. Θα πρέπει να διακόψουμε την τροφοδοσία με τάση (τάση 0V) στο Pin10.
6. Θα πρέπει να περιμένουμε 1 δευτερόλεπτο κατά τη διάρκεια του οποίου προφανώς είναι σβηστό το led.

Τα βήματα 1,2 θα πρέπει να γίνουν μόνο μία φορά. Τα βήματα 3-6 θα πρέπει να επαναλαμβάνονται συνεχώς.

Μαθαίνω:

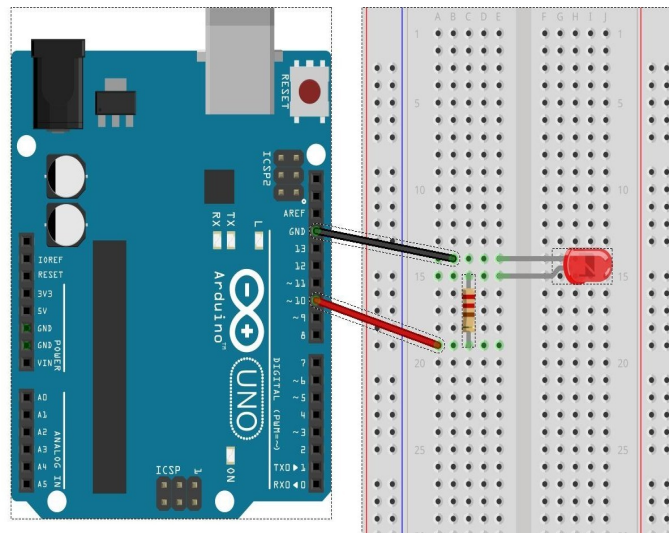
Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

10.2. Φύλλο εργασίας 2 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out)

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε ένα led το οποίο όμως δεν θα ανάβει με το μέγιστο δυνατό ρεύμα όπως στις προηγούμενες ασκήσεις, αλλά η έντασή του θα είναι αυξομειούμενη. Θα ξεκινά δηλαδή να ανάβει αχνά και σταδιακά θα αυξάνει η ένταση, μέχρι να φτάσει στο μέγιστο. Τότε, αντίστροφα θα ξεκινά η ένταση να μειώνεται σταδιακά μέχρι να φτάσει στο χαμηλότερο επίπεδο.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα led και μια αντίσταση, όπως έχει περιγραφεί στα προηγούμενα φύλλα εργασίας και φαίνεται στην εικόνα 23.



Εικόνα 23 – Κύκλωμα φύλλον εργασίας 2

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρειαστεί να συνδέσουμε το led μας σε κάποια από τις PWM θύρες του Arduino, οι οποίες δουλεύουν κι ως αναλογικές εξόδους. Οι δυνατές τιμές που μπορούν να δώσουν είναι από το 0 έως το 255, προσομοιώνοντας τις δυνατές τάσεις από 0 έως 5V. Έτσι, θα χρειαστεί να προγραμματίσετε τη θύρα σας ως εξόδο και ακολούθως με χρήση της συνάρτησης `analogWrite(Pin, Brightness)` να δίνετε στο Pin σιγά σιγά τις επιθυμητές τιμές, από 0 μέχρι 255. Προσοχή να μην ξεπεράσετε τα όρια αυτά!

Όπως ήδη γνωρίζετε, θα πρέπει να:

1. Ορίσετε μια μεταβλητή για τη θύρα σας και αρχικοποιήστε τη θύρα ως εξόδο.
2. Ορίσετε μια μεταβλητή για τη φωτεινότητα με αρχική τιμή το 0.
3. Σε κάθε επανάληψη σταδιακά (π.χ. ανά 5) αυξάνετε τη φωτεινότητα, μέχρι το 255.
4. Όταν φτάνει στο 255 θα πρέπει να μηδενίζετε τη φωτεινότητα και να ξεκινάτε πάλι από την αρχή.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

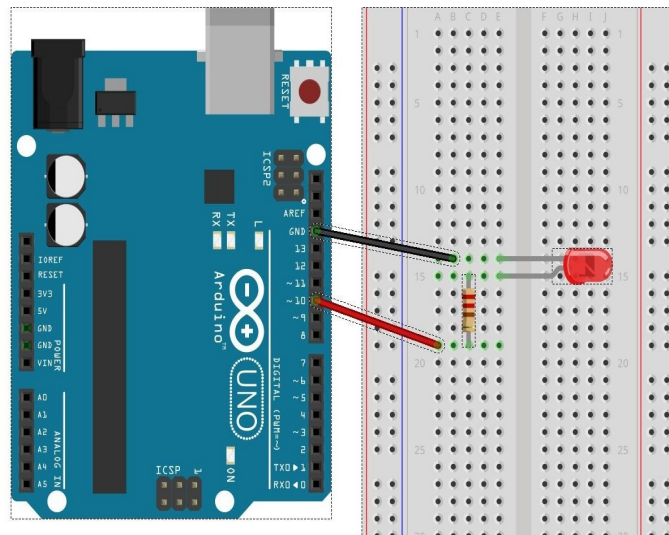
Μαθαίνω: Αναλογική έξοδος (6.4.3.)

10.3. Φύλλο εργασίας 3 – Αυξηση και μείωση φωτεινότητας (Fade in-Fade out), Αναβόσβημα στις άκρες (Blink at peaks)

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε το προηγούμενο κύκλωμά μας, αλλά θα αλλάξουμε τον τρόπο που αυτό αναβοσβήνει. Η ένταση θα ανεβαίνει σταδιακά, κι όταν φτάνει στο μέγιστο θα κάνουμε το led να αναβοσβήσει δύο φορές (blink) κι έπειτα να μειώνεται η ένταση σιγά σιγά. Όταν φτάνει να σβήσει, θα αναβοσβήνει πάλι δύο φορές και θα ξεκινά η ίδια διαδικασία.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα led και μια αντίσταση, όπως έχει περιγραφεί στα προηγούμενα φύλλα εργασίας και φαίνεται στην εικόνα 24.



Εικόνα 24 – Κύκλωμα φύλλου εργασίας 3

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρησιμοποιήσουμε το ίδιο ακριβώς κύκλωμα των προηγούμενων φύλλων εργασίας και θα χρειαστεί μόνο να αλλάξουμε λίγο τον προγραμματισμό του, ώστε να αναβοσβήνει όταν φτάσει στα όρια. Συγκεκριμένα:

1. Κάνετε τις κατάλληλες δηλώσεις και αρχικοποιήσεις (μεταβλητές, τιμές, θύρες).
2. Ξεκινήστε από φωτεινότητα 0 και σταδιακά (π.χ. ανά 5) να φτάνετε στο 255.
3. Όταν φτάσετε στη μέγιστη φωτεινότητα (255), θα κάνετε το led να αναβοσβήσει δύο φορές (blink).
4. Ξεκινήστε σταδιακά να μειώνετε τη φωτεινότητα σιγά σιγά (π.χ. ανά -5) μέχρι να σβήσει, δηλαδή να φτάσει στο 0.
5. Όταν φτάσετε στην ελάχιστη φωτεινότητα (0), θα αναβοσβήσετε δύο φορές το led (blink) και θα ξεκινάει πάλι από την αρχή η διαδικασία.

Χρησιμοποιοῦ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδο (6.4.3.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

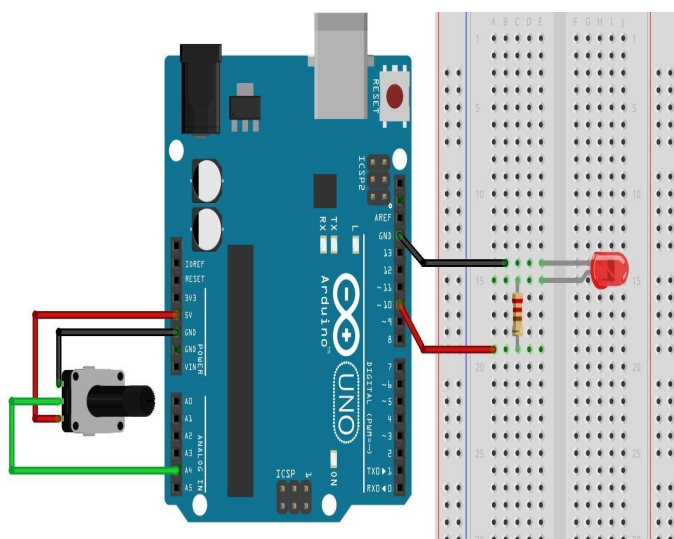
Μαθαίνω: Δομή επιλογής (6.8)

10.4. Φύλλο εργασίας 4 – Σταδιακή αύξηση και μείωση φωτεινότητας με ποτενσιόμετρο

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε ένα ποτενσιόμετρο για να ρυθμίζουμε την ένταση με την οποία θα ανάψει το led μας. Για να γίνει αυτό θα πρέπει να συνδέσουμε ένα led και μια αντίσταση σε ένα ψηφιακό pin, ένα ποτενσιόμετρο σε ένα αναλογικό pin, και ανάλογα με την τιμή που μας δίνει (διαβάζουμε από) το ποτενσιόμετρο θα δίνουμε αντίστοιχη ένταση στο led.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα led, και μια αντίσταση συνδεδεμένα σε ένα ψηφιακό pin (PWM). Επίσης, θα πρέπει να συνδέσουμε ένα ποτενσιόμετρο σε μια αναλογική είσοδο (υπάρχουν έξι όπως έχουμε δει στο Arduino Uno) από την οποία θα διαβάζουμε τις τιμές του ποτενσιόμετρου. Για το ποτενσιόμετρο η συνδεσμολογία είναι απλή, όπως φαίνεται στην εικόνα 25. Χρειαζόμαστε πηγή (5V), γείωση (GND) και το μεσαίο πόδι συνδεδεμένο σε μια αναλογική είσοδο. Οι τιμές που θα έρθουν από εκεί θα κυμαίνονται από το 0 έως το 1023, οπότε θα πρέπει να κάνουμε την αντίστοιχη μετατροπή για να τις δώσουμε ως ένταση του Pin (π.χ. να δίνουμε την είσοδο διά του 4). Το ποτενσιόμετρο τοποθετείται και απευθείας πάνω στο breadboard, αν έχει τα κατάλληλα άκρα.



Εικόνα 25 – Κύκλωμα φύλλου εργασίας 4

Προγραμματισμός κυκλώματος

Προσέξτε στην άσκηση αυτή ότι πρέπει να αρχικοποιήσετε τις θύρες κατάλληλα, δηλαδή αυτή με το led ως εξόδου (OUTPUT) κι αυτή με το ποτενσιόμετρο ως είσοδο (INPUT). Χρησιμοποιήστε μια μεταβλητή για την καταχώρηση της τιμής του ποτενσιόμετρου, κι όπως αναφέρθηκε ήδη, επειδή η τιμή αυτή θα κυμαίνεται από 0 έως και 1023, για να την προσαρμόσετε ως έξοδο προς το led, διαιρέστε τη με το 4. Οι εντολές που θα χρειαστείτε είναι η `AnalogWrite(PIN, Value)` και η `AnalogRead(Pin)`. Προσέξτε ότι τα ονόματα των αναλογικών Pin είναι τα A0, A1, ... , A5, ενώ των ψηφιακών τα 0, 1, 2,...13.

1. Κάνετε τις απαραίτητες δηλώσεις και αρχικοποιήσεις (είσοδος-έξοδος).
2. Σε κάθε επανάληψη θα διαβάζετε την τιμή από το ποτενσιόμετρο και θα τη δίνετε ως είσοδο (με την κατάλληλη προσαρμογή) στο led σας.
3. Χρησιμοποιήστε μια μικρή καθυστέρηση για να προλαβαίνει να είναι διακριτή η διαφορά στη φωτεινότητα στο led (π.χ. 50 msec).

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδος (6.4.3.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

Μαθαίνω: Αναλογική είσοδος (6.4.4), Ποτενσιόμετρο (9.3)

10.5. Φύλλο εργασίας 5 - Χρησιμοποιώντας τη σειριακή οθόνη

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε τη σειριακή οθόνη, στέλνοντας μηνύματα και τιμές μεταβλητών από ένα πρόγραμμα που έχουμε φορτώσει στη μονάδα μας.

Περιγραφή κυκλώματος

Για τη σειριακή οθόνη και επικοινωνία της μονάδας με τον υπολογιστή μας δεν χρειαζόμαστε κάποια ιδιαίτερη συνδεσμολογία, εκτός από τη σύνδεση με καλώδιο USB.

Προγραμματισμός κυκλώματος

Στο φύλλο εργασίας αυτό θα γράψετε ένα απλό πρόγραμμα που θα ενεργοποιεί τη σειριακή θύρα και θα στέλνει μηνύματα στην οθόνη για εμφάνιση, με σκοπό να εξοικειωθείτε με τη σειριακή οθόνη.

1. Ενεργοποιήστε τη σειριακή θύρα στη συνάρτηση `setup()` με ρυθμό δεδομένων 9600.
2. Στη συνάρτηση `loop()`, δηλαδή σε κάθε επανάληψη στείλτε ένα μήνυμα της μορφής “Η επικοινωνία ksekinhse:” χωρίς αλλαγή γραμμής.
3. Με μια επαναληπτική διαδικασία δώστε τιμές σε μια μεταβλητή από το 1 μέχρι το 20 και τις τιμές αυτές στείλτε τις στη σειριακή οθόνη να εκτυπωθούν σε μια γραμμή η κάθε μία.
4. Φτιάξτε ένα πιο σύνθετο μήνυμα χρησιμοποιώντας κείμενο και τιμή μεταβλητής, χρησιμοποιώντας εκτύπωση με και χωρίς αλλαγή γραμμής όταν πρέπει, π.χ. “Τωρα vlerw thn epanalhprsh : X”, όπου στο X θα εμφανίζεται η αντίστοιχη τιμή της μεταβλητής σε κάθε επανάληψη.
5. Συνδέστε τη μονάδα σας και αφού επιλέξετε τα απαραίτητα (τύπο μονάδας και θύρα), φορτώστε το πρόγραμμά σας.
6. Ακολουθώντας, ενεργοποιήστε τη σειριακή οθόνη και παρακολουθήστε τα μηνύματα που βγαίνουν σε αυτή.
7. Προσέξτε ότι δυστυχώς τα ελληνικά δεν εμφανίζονται σωστά.

Προτροπή: Η σειριακή οθόνη μπορεί να χρησιμοποιηθεί για να δούμε τί συμβαίνει κατά τη διάρκεια εκτέλεσης του κώδικά μας (debugging). Για παράδειγμα, μπορείτε να ανατρέξετε στο κύκλωμα και το πρόγραμμα του προηγούμενου φύλλου εργασίας και αφού συνδέσετε τη σειριακή οθόνη:

1. Όταν διαβάσετε την τιμή από το ποτενσιόμετρο να την εκτυπώνετε και στη σειριακή οθόνη χωρίς αλλαγή γραμμής.
2. Όταν προσαρμόζετε την τιμή του ποτενσιόμετρου για την δώσετε ως είσοδο να την εκτυπώνετε και στη σειριακή οθόνη (προσέξτε το κενό για να μην κολλήσει με την προηγούμενη εκτύπωση) με αλλαγή γραμμής.
3. Έτσι, σε κάθε γραμμή θα βλέπετε τις δύο τιμές!

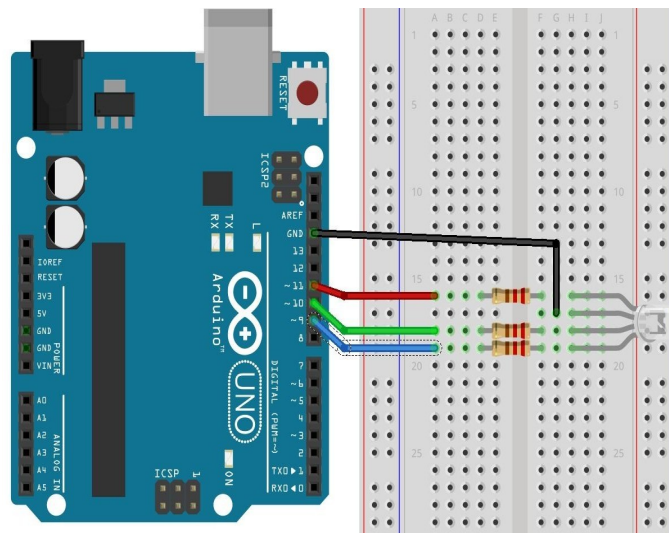
Μαθαίνω: Σειριακή οθόνη (6.7)

10.6. Φύλλο εργασίας 6 – Παίζοντας με τα χρώματα (RGB Led)

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε ένα RGB led το οποίο θα προγραμματίσουμε ώστε να παίρνει μία γκάμα χρωμάτων.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα RGB led και τρεις αντιστάσεις, μία για κάθε χρώμα. Στην εικόνα 26 βλέπετε τη συνδεσμολογία. Προσέξτε τις τιμές των αντιστάσεων, η τιμή για την κόκκινη είναι λίγο διαφορετική από αυτές για το πράσινο και μπλε. Επίσης, προσέξτε ποιο pin θα συνδέσετε με ποιο χρώμα. Το μακρύτερο (2ο) πόδι του led πάει στη γείωση (GND), το δίπλα του εξωτερικό (1ο) στο κόκκινο, το δίπλα του εσωτερικό (3ο) στο πράσινο και το τελευταίο εξωτερικό (4ο) στο μπλε. Επίσης, για όλα τα χρώματα θα χρειαστούμε PWM θύρες κι όχι απλές ψηφιακές.



Εικόνα 26 – Κύκλωμα φύλλον εργασίας 6

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρειαστεί να συνδέσουμε το led μας σε κάποια από τις PWM θύρες του Arduino, οι οποίες δουλεύουν όπως έχουμε δει και ως αναλογικές έξοδοι. Συγκεκριμένα:

1. Κάνετε όποιες αρχικοποιήσεις χρειάζονται (μεταβλητές, τιμές, θύρες).
2. Με μια επαναληπτική διαδικασία θα δίνετε σταδιακά τιμές σε κάθε led. Με τη βοήθεια του καθηγητή σας προσπαθήστε να γράψετε μια εντολή επανάληψης που με διάφορους τρόπους θα παραλλάζει τις τιμές που χρησιμοποιείτε στα led, π.χ.

```
for(i=0;i<=255;i=i+5) {  
    analogWrite(redPin, i);  
    analogWrite(greenPin, i);  
    analogWrite(bluePin, i);  
    delay(200);  
}
```

3. Προσπαθήστε να παραλλάξετε τον κώδικά σας και να πετύχετε όλους τους δυνατούς συνδυασμούς χρωμάτων!

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδος (6.4.3.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8)

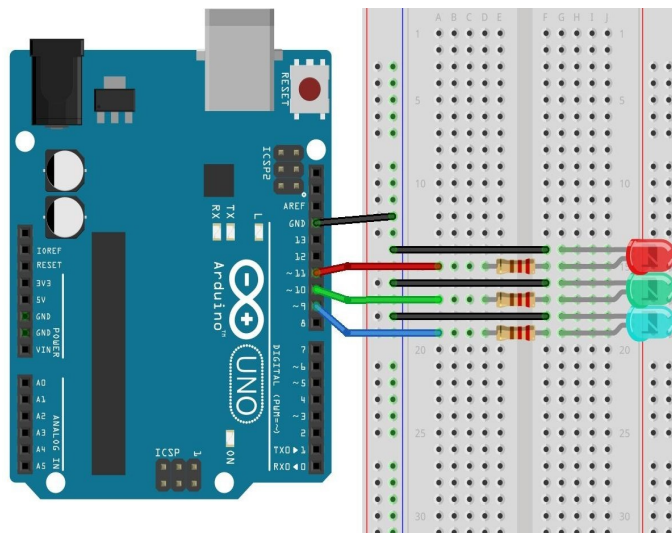
Μαθαίνω: RGB led (9.1.1), Δομή επανάληψης (6.9)

10.7. Φύλλο εργασίας 7 – Κατασκευάζοντας ένα RGB Led

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε τρία led, ένα κόκκινο, ένα πράσινο κι ένα μπλε, προσπαθώντας έτσι να προσομοιώσουμε ένα RGB led και ακολούθως θα προγραμματίσουμε τη διάταξη αυτή ώστε να παίρνει μία γκάμα χρωμάτων.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε τρία έγχρωμα led (κόκκινο, πράσινο, μπλε) και τρεις αντιστάσεις, μία για κάθε χρώμα. Η συνδεσμολογία φαίνεται στην εικόνα 27. Προσέξτε ποιο pin θα συνδέσετε με ποιο χρώμα. Για όλα τα χρώματα θα χρειαστούμε PWM θύρες κι όχι απλές ψηφιακές. Επίσης, τοποθετήστε όσο πιο κοντά μπορείτε τα led, ώστε να μπορέσουμε να τα καλύψουμε μετά με έναν κυλινδρικό σωλήνα για να δούμε συνολικά τι χρώμα βγαίνει!



Εικόνα 27 – Κύκλωμα φύλλου εργασίας 7

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρειαστεί να συνδέσουμε τα leds μας σε PWM θύρες. Η προσέγγισή μας είναι ίδια με αυτή στο προηγούμενο φύλλο, και τα ίδια βήματα θα πρέπει να ακολουθήσετε κι εδώ. Η διαφορά μας είναι ότι φτιάξατε μόνοι σας το RGB led! Για να φανεί η μίξη των χρωμάτων, θα χρησιμοποιήσετε ένα κυλινδρικό άσπρο σωλήνα, ανοικτό και από τις δύο άκρες, ώστε να καπακώσετε τα Led και να μπορείτε από την πάνω άκρη του να βλέπετε τι συμβαίνει.

1. Κάνετε τις απαραίτητες αρχικοποιήσεις (δηλώσεις μεταβλητών, τιμές, είσοδος/έξοδος).
2. Με μια επαναληπτική διαδικασία θα δίνετε σταδιακά τιμές σε κάθε led, όπως στο προηγούμενο φύλλο εργασίας.
3. Προσπαθήστε να παραλλάξετε τον κώδικά σας και να πετύχετε όλους τους δυνατούς συνδυασμούς χρωμάτων!

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδος (6.4.3.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

Μαθαίνω: RGB led (9.1.1), Δομή επανάληψης (6.9)

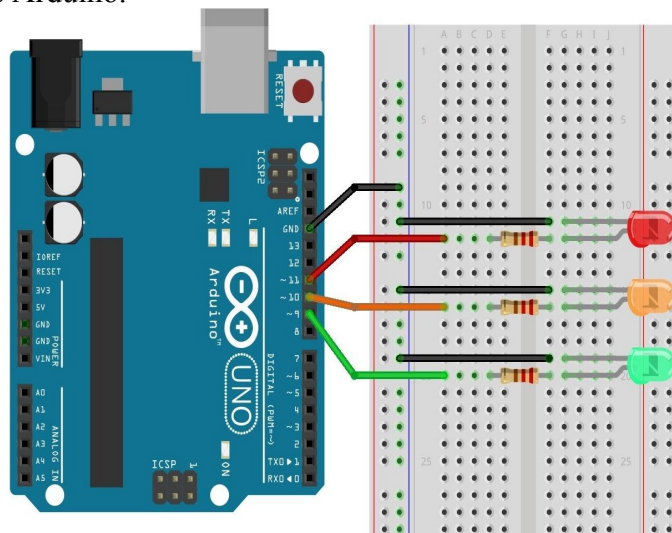
10.8. Φύλλο εργασίας 8 – Τα φανάρια κυκλοφορίας

Στο φύλλο εργασίας αυτό θα προσομοιώσουμε τη λειτουργία των φαναριών. Έτσι, δίνοντας κατάλληλα χρονικά διαστήματα θα κρατήσουμε ανοιχτό το κάθε χρώμα για κάποιο χρόνο και ακολούθως η διαδικασία θα επαναλαμβάνεται από την αρχή. Τα φανάρια θα αναβοσβήνουν όπως στην Ελλάδα*, δηλαδή Πράσινο, Πορτοκαλί, Κόκκινο, Πράσινο, κ.ό.κ.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε τρία led, ένα κόκκινο, ένα πράσινο κι ένα πορτοκαλί (ή κίτρινο) – ένα για κάθε φανάρι, όπως επίσης και τις αντίστοιχες αντιστάσεις.

Μπορείτε να δείτε το κύκλωμα στην εικόνα 28 που ακολουθεί. Προσέξτε το γεγονός ότι επειδή δεν υπάρχουν πολλές υποδοχές GND, όλες βραχυκυκλώνονται πάνω στο breadboard (-) και εκεί συνδέουμε το GND του Arduino.



Εικόνα 28 – Κύκλωμα φύλλου εργασίας 8

Προγραμματισμός κυκλώματος

Το μόνο που μένει είναι να προγραμματίσουμε το κύκλωμά μας, σύμφωνα με αυτά που έχουμε ήδη αναφέρει. Δηλαδή, θα κάνετε τις απαραίτητες αρχικοποιήσεις και ακολούθως:

1. Θα πρέπει να δίνετε τάση 5V στο led που θέλετε να ανάβει και 0V στα υπόλοιπα.
2. Θα πρέπει να περιμένουμε 1 δευτερόλεπτο κατά τη διάρκεια του οποίου προφανώς ανάβει κάποιο led και είναι σβηστά τα υπόλοιπα.
3. Τα βήματα 1, 2 θα τα κάνετε εναλλάξ για το κόκκινο, πορτοκαλί και πράσινο Led.
4. Δοκιμάστε να προσαρμόσετε τους χρόνους. Το πορτοκαλί συνήθως ανάβει για πολύ λιγότερο χρόνο από ότι τα άλλα δύο χρώματα.

* Αναφέρουμε όπως στην Ελλάδα, καθώς σε άλλες χώρες έχουμε διαφοροποίηση στη σειρά των χρωμάτων. Αν έχουμε κόκκινο ανάβει πρώτα πορτοκαλί (για προετοιμασία) και μετά πράσινο. Μετά πάλι πορτοκαλί και μετά κόκκινο.

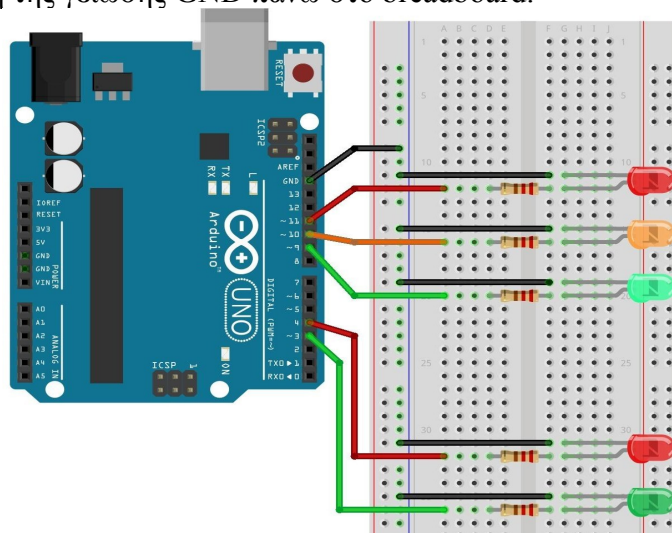
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1.), Καθυστερήση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

10.9. Φύλλο εργασίας 9 – Τα φανάρια κυκλοφορίας με φανάρι πεζών

Στο φύλλο εργασίας αυτό θα εμπλουτίσουμε τη λειτουργία των φαναριών, όπως υλοποιήθηκε στο φύλλο εργασίας 8, με δύο ακόμα φανάρια για τους πεζούς. Έτσι, δίνοντας κατάλληλα χρονικά διαστήματα θα κρατήσουμε ανοιχτό το κάθε χρώμα κυκλοφορίας για κάποιο χρόνο, ενώ όταν υπάρχει κίνηση θα ανάβει το κόκκινο των πεζών, ενώ όταν υπάρχει κόκκινο στην κυκλοφορία θα ανάβει το πράσινο των πεζών. Ακολούθως η διαδικασία θα επαναλαμβάνεται από την αρχή.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε, επιπλέον από αυτά του φύλλου εργασίας 8, δύο led, ένα κόκκινο κι ένα πράσινο για να προσομοιώσουμε τη λειτουργία των φαναριών των πεζών και τις αντίστοιχες αντιστάσεις. Μπορείτε να δείτε το κύκλωμα στην εικόνα 29 που ακολουθεί. Προσέξτε ξανά τη βραχυκύκλωση της γείωσης GND πάνω στο breadboard.



Εικόνα 29 – Κύκλωμα φύλλου εργασίας 9

Προγραμματισμός κυκλώματος

Το μόνο που μένει είναι να προγραμματίσουμε το κύκλωμά μας, σύμφωνα με αυτά που έχουμε ήδη αναφέρει. Θα πρέπει να καθορίσετε κατάλληλα τους χρόνους και τους συνδυασμούς των φαναριών πεζών και κυκλοφορίας.

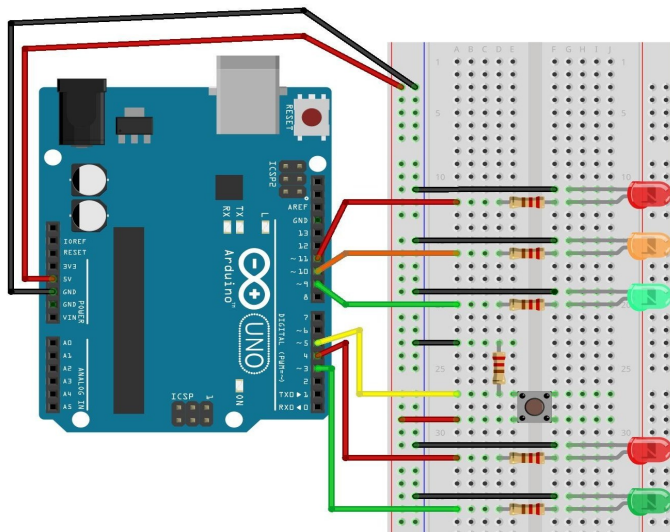
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1.), Καθυστερήση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

10.10. Φύλλο εργασίας 10 – Τα φανάρια κυκλοφορίας με φανάρι πεζών και κουμπί διακοπής

Στο φύλλο εργασίας αυτό θα εμπλουτίσουμε ακόμα περισσότερο τη λειτουργία των φαναριών, όπως υλοποιήθηκε στα προηγούμενα φύλλα εργασίας (φύλλα 8 και 9), με την εισαγωγή ενός κουμπιού, το οποίο θα καθορίζει τη λειτουργία των φαναριών ως εξής: κανονικά θα λειτουργεί το φανάρι κυκλοφορίας μόνιμα ανοικτό (πράσινο), εκτός αν πατηθεί το πλήκτρο, το οποίο θα σημαίνει ότι κάποιος πεζός ζητάει να περάσει. Στην περίπτωση αυτή θα πρέπει να διακόπτεται η κυκλοφορία (πορτοκαλί και κόκκινο) και ακολούθως να ανάβει το πράσινο των πεζών για καθορισμένο χρόνο. Τέλος, θα γίνεται πάλι κόκκινο το φανάρι των πεζών και θα ξεκινά πάλι η κυκλοφορία των αυτοκινήτων (πράσινο). Ακολούθως η διαδικασία θα επαναλαμβάνεται από την αρχή.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε, επιπλέον από τα υλικά των φύλλων εργασίας 8 και 9, ένα κουμπί για να προσομοιώσουμε τη λειτουργία του κουμπιού των φαναριών των πεζών. Θα χρειαστούμε επίσης μια αντίσταση και για το κουμπί. Μπορείτε να δείτε το κύκλωμα στην εικόνα 30 που ακολουθεί.



Εικόνα 30 – Κύκλωμα φύλλον εργασίας 10

Προγραμματισμός κυκλώματος

Το μόνο που μένει είναι να προγραμματίσουμε το κύκλωμά μας. Για ευκολία, σας δίνονται οδηγίες με τη μορφή βημάτων:

1. Κάνετε τις απαραίτητες αρχικοποιήσεις (μεταβλητές, τιμές, είσοδος/έξοδος).
2. Καθορίστε τους χρόνους και τους συνδυασμούς φαναριών πεζών και κυκλοφορίας.
3. Θα πρέπει να διαβάσετε την είσοδο από την αντίστοιχη για το κουμπί είσοδο (pin) κι όταν ανιχνεύετε είσοδο να προβαίνετε στις αντίστοιχες ενέργειες που περιγράφηκαν παραπάνω. Για το βήμα αυτό, προτείνεται να χρησιμοποιήσετε καταγραφή του χρόνου για να γνωρίζετε πότε πατήθηκε τελευταία φορά το κουμπί, με χρήση της συνάρτησης millis(). Ορίστε επίσης ένα ελάχιστο διάστημα κατά το οποίο δεν θα ανάβει το πράσινο των πεζών.

Προσέξτε ποια βήματα θα πρέπει γίνονται μόνο μία φορά και ποια θα επαναλαμβάνονται συνεχώς.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1)

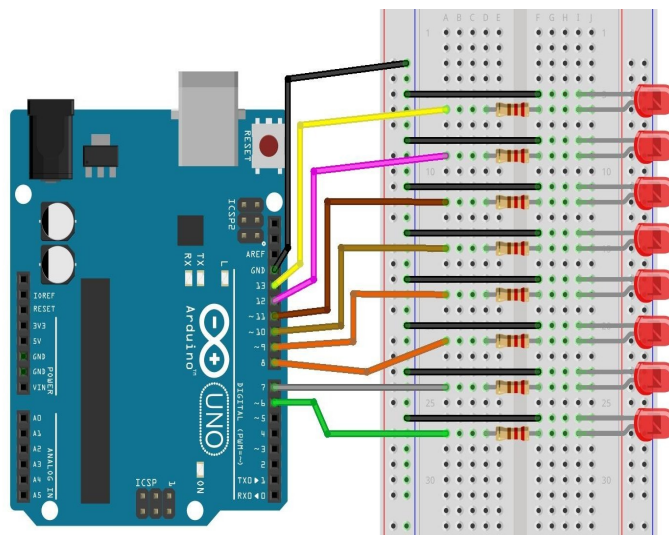
Μαθαίνω: Κουμπιά (9.2), Καταγραφή χρόνου (6.5.3)

10.11. Φύλλο εργασίας 11 – Εφέ, κνηγώντας τη λάμψη (Led chase effect)

Στο φύλλο εργασίας αυτό θα χρησιμοποιηθεί ένας ικανοποιητικός αριθμός led, ανάλογα με τη διαθεσιμότητα αυτών στο εργαστήριό σας, π.χ. 8. Θα τα συνδέσετε με τον γνωστό σας τρόπο σε αντίστοιχα pins και στόχος είναι να ανάβει το ένα αμέσως μετά το άλλο, ενώ σβήνει το προηγούμενο, ώστε να φτιάξουμε ένα όμορφο εφέ. Όταν φτάνει στο τελευταίο led το κύμα θα πρέπει να επιστρέφει προς τα πίσω.

Περιγραφή κυκλώματος

Θα χρειαστούμε έναν ικανό αριθμό led και αντίστοιχων αντιστάσεων. Ακολουθούμε την κλασσική σύνδεση για τα led και αντιστάσεις, όπως φαίνεται στην εικόνα 31.



Εικόνα 31 – Κύκλωμα φύλλου εργασίας 11

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρειαστεί να ενεργοποιούμε και να απενεργοποιούμε με αύξουσα σειρά τα led μας μέχρι το τελευταίο και αντίστροφα, δηλαδή να τα ενεργοποιούμε και απενεργοποιούμε με φθίνουσα σειρά. Αυτό γίνεται με αρκετές εντολές ή, προτιμότερα, με μια επαναληπτική διαδικασία κι ένα πίνακα με τιμές τους αριθμούς των led, ώστε να έχουμε πιο συνοπτικό κι αποδοτικό κώδικα.

1. Αρχικά χρησιμοποιήστε μεταβλητές για κάθε led και διαχειριστείτε τις με τον γνωστό τρόπο (αρχικοποίηση, γράψιμο τιμής με την `digitalWrite(led, value)`).
2. Ολοκληρώστε τον παραπάνω κώδικα και αποθηκεύστε τον.
3. Σε ένα νέο πρόγραμμα (ή αντίγραφο του προηγούμενου προγράμματος), προσπαθήστε να χρησιμοποιήσετε την εντολή επανάληψης για να ορίσετε τα led ως έξοδο, να τα ανάβετε, να τα σβήνετε. Για παράδειγμα θα μπορούσατε να ανάβετε τα led με τις εντολές:

```
for (i=0;i<=7;i++) {  
    digitalWrite(Led[i], HIGH);  
    delay(200);  
    digitalWrite(Led[i], LOW);  
}
```

4. Προσπαθήστε με αντίστοιχη διαδικασία να αρχικοποιήσετε τα Leds ως έξοδο. Παρατηρήστε ότι χρησιμοποιείται η δομή του πίνακα για να ομαδοποιήσουμε παρόμοια πράγματα.

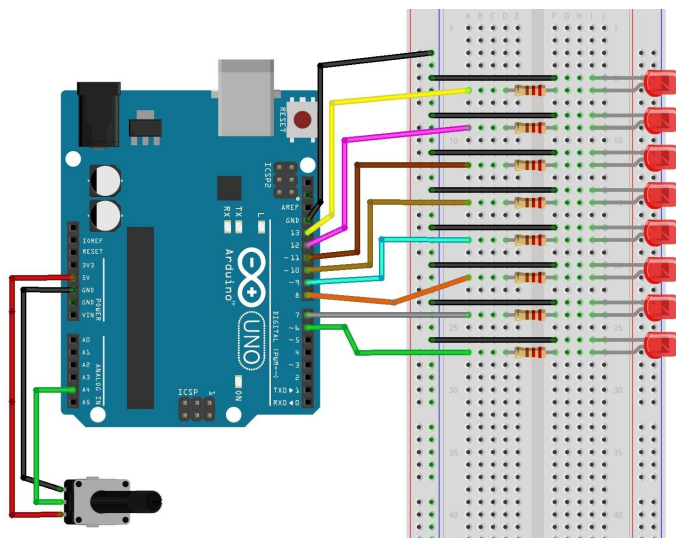
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1.), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Led (9.1), Δομή επανάληψης (6.9)

10.12. Φύλλο εργασίας 12 – Εφέ, κουνιγώντας τη λάμψη (Led chase effect) με ποτενσιόμετρο

Στο φύλλο εργασίας αυτό θα προσαρμόσουμε το κύκλωμα και το πρόγραμμα του προηγούμενου φύλλου εργασίας, ώστε η ταχύτητα που μεταδίδεται το εφέ μας να μην είναι σταθερή. Στο προηγούμενο παράδειγμα δίναμε ένα delay(500) ώστε να μένει το κάθε led αναμμένο για μισό δευτερόλεπτο πριν σβήσει και προχωρήσουμε στο επόμενο. Τώρα θα χρησιμοποιήσουμε ένα ποτενσιόμετρο ώστε βάσει αυτού (της τιμής που μας δίνει) να αυξάνουμε ή να μειώνουμε την ταχύτητα διάδοσης του εφέ, δηλαδή το χρόνο του delay.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ότι και στο προηγούμενο φύλλο εργασίας, leds και αντιστάσεις, καθώς και ένα ποτενσιόμετρο. Το ποτενσιόμετρο το έχουμε χρησιμοποιήσει ξανά σε προηγούμενο φύλλο εργασίας, οπότε ο αναγνώστης είναι εξοικειωμένος με αυτό. Συνδέεται σε μια αναλογική είσοδο από την οποία θα διαβάζουμε τις τιμές του. Η συνδεσμολογία είναι απλή, όπως φαίνεται στην εικόνα 32. Χρειαζόμαστε πηγή (5V), γείωση (GND) και το μεσαίο πόδι συνδεδεμένο σε μια αναλογική είσοδο. Οι τιμές που θα έρθουν από εκεί θα κυμαίνονται από το 0 έως το 1023, οπότε θα κάνουμε την αντίστοιχη μετατροπή για να τις δώσουμε ως χρονοκαθυστέρηση. Το ποτενσιόμετρο τοποθετείται και απευθείας πάνω στο breadboard, αν έχει τα κατάλληλα άκρα.



Εικόνα 32 – Κύκλωμα φύλλου εργασίας 12

Προγραμματισμός κυκλώματος

Θα πρέπει να προσαρμόσουμε τον κώδικα που φτιάξαμε στο προηγούμενο φύλλο εργασίας ώστε να δίνει την καθυστέρηση ανάλογα με την τιμή που διαβάζουμε από το ποτενσιόμετρο. Προσέξτε το γεγονός ότι μια μεγάλη τιμή στο ποτενσιόμετρο θα σημαίνει μεγάλη ταχύτητα εναλλαγής, οπότε μικρό χρόνο στο delay. Η τιμή δηλαδή που θα δίνεται θα πρέπει να είναι αντιστρόφως ανάλογη της τιμής που διαβάζουμε!

Υπάρχει βέβαια και η περίπτωση να συνδέσουμε το ποτενσιόμετρό μας αντίθετα δηλαδή πρώτα τη γείωση και μετά την πηγή – στην περίπτωση αυτή όσο γυρίζουμε το χειριστήριό του περισσότερο, τόσο μειώνουμε την τιμή που διαβάζουμε. Στην περίπτωση αυτή αρκεί να χρησιμοποιήσουμε ανάλογο μέγεθος με την τιμή του ποτενσιόμετρου στην καθυστέρηση που δίνουμε.

Πειραματιστείτε λοιπόν και καταγράψτε τις παρατηρήσεις σας!

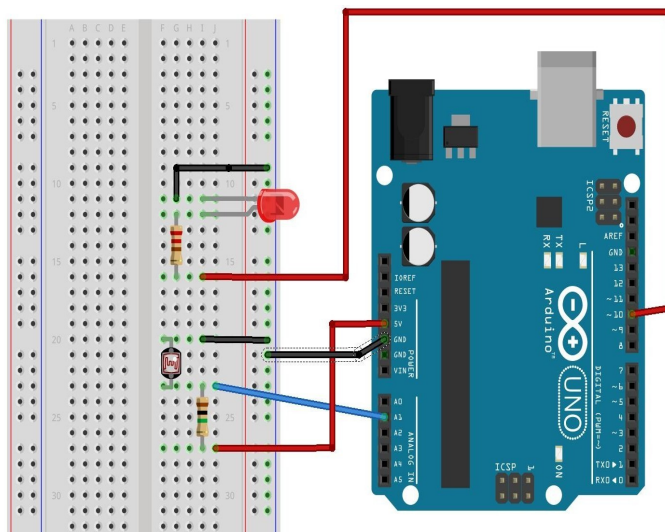
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Ψηφιακή έξοδος (6.4.1.), Αναλογική είσοδος (6.4.4), Καθυστέρηση (6.5.1), Δομή επανάληψης (6.9), Breadboard(7), Φόρτωση (8), Led (9.1), Ποτενσιόμετρο (9.3)

10.13. Φύλλο εργασίας 13 – Εφέ, ανιχνεύοντας το φως (Led effect + light sensor)

Στο φύλλο εργασίας αυτό θα δημιουργήσουμε μια διάταξη που αντιδρά στα οπτικά ερεθίσματα. Θα αποτελείται από ένα led το οποίο θα αυξομειώνει την έντασή του ανάλογα με το πόσο φως προσπίπτει σε μια φωτοευαίσθητη αντίσταση.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα led και μια αντίσταση συνδεδεμένα στο Arduino μας. Για να αναγνωρίσουμε τη φωτεινότητα χρησιμοποιούμε μια φωτοευαίσθητη αντίσταση (photoresistor), η οποία αυξάνει ή μειώνει το μέγεθός της ανάλογα με τη φωτεινότητα του χώρου στον οποίο βρισκόμαστε. Η αντίσταση αυτή θα τοποθετηθεί σε σειρά μαζί με μια κανονική αντίσταση (περίπου 1MΩ), κι έτσι έχουμε μια διάταξη με τρία άκρα: ένα στην πηγή, ένα (αυτό που συνδέονται οι δύο αντιστάσεις) σε αναλογική είσοδο και ένα στη γείωση (εικόνα 33). Η διάταξη προσομοιώνει ένα ποτενσιόμετρο (θυμηθείτε το αντίστοιχο φύλλο εργασίας), όπου το ρόλο του διακόπτη του ποτενσιόμετρου (μεταβολή τάσης που διαβάζουμε) παίζει η φωτοευαίσθητη αντίσταση.



Εικόνα 33 – Κύκλωμα φύλλου εργασίας 13

Προγραμματισμός κυκλώματος

Ο προγραμματισμός του κυκλώματος αυτού είναι εντελώς παρόμοιος με αυτόν του φύλλου με το ποτενσιόμετρο. Διαβάζουμε μια τιμή από την αναλογική είσοδο κι αυτή την τιμή την προσαρμόζουμε κατάλληλα ώστε να τη δώσουμε ως αναλογική έξοδο στο pin που συνδέσαμε με το led μας. Θυμηθείτε να προσαρμόσετε κατάλληλα την είσοδό σας (τιμές 0-1023) στην έξοδο προς το led (τιμές 0-255). Για να το επιτύχετε αυτό μπορείτε να κάνετε κάποια διαίρεση (π.χ. τιμή εισόδου με το 4) ή να χρησιμοποιήσετε τη συνάρτηση map που κάνει αυτό ακριβώς!

Πειραματισμός 1: Πειραματιστείτε με τη φωτοευαίσθητη αντίσταση. Συνδέστε αρχικά το ένα πόδι της στην πηγή και το πόδι της απλής αντίστασης στη γείωση. Τρέξτε το πρόγραμμά σας. Ακολουθώντας, συνδέστε το πόδι της φωτοευαίσθητης αντίστασης στη γείωση και το πόδι της απλής στην πηγή. Τι παρατηρείτε να συμβαίνει;

Πειραματισμός 2: Συνδέστε τα πόδια της φωτοευαίσθητης αντίστασης με μακριά καλώδια ώστε να κινηθείτε στο χώρο με αυτή, κι ανιχνεύστε έτσι τις αλλαγές φωτεινότητας!

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Καθυστέρηση (6.5.1), Αναλογική έξοδος (6.4.3.), Αναλογική είσοδος (6.4.4), Breadboard(7), Φόρτωση (8), Led (9.1)

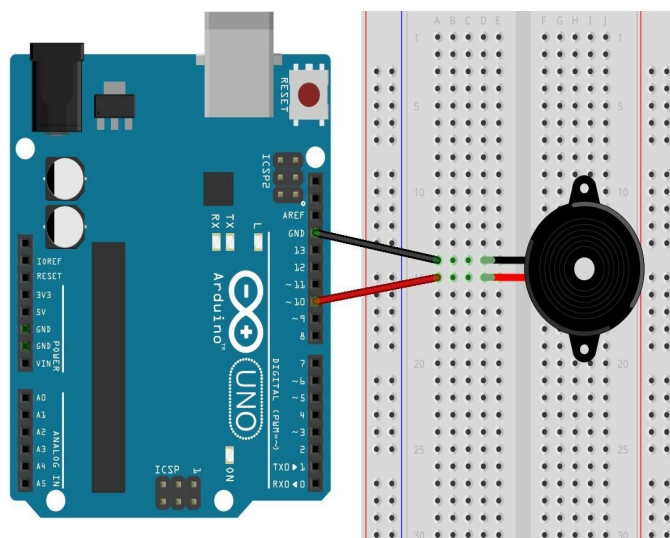
Μαθαίνω: Συνάρτηση αντιστοίχισης τιμών map (6.6), Φωτοευαίσθητη αντίσταση (9.5)

10.14. Φύλλο εργασίας 14 – Χρησιμοποιώντας τον ήχο (Sounder)

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε μια απλή διάταξη με ήχο. Υπάρχουν εξαρτήματα τα οποία όταν δεχθούν ρεύμα παράγουν ήχους (sounders) και είναι σε διάφορους τύπους. Στην άσκηση αυτή θα συνδέσουμε ένα τέτοιο ηχείο και παίζοντας με το ρεύμα θα προσπαθήσουμε να παράγουμε διάφορους ήχους.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα ηχείο. Τα ηχεία αυτά δουλεύουν με συνεχές ρεύμα (πηγή - γείωση), οπότε έχουμε μια διάταξη όπου το ένα άκρο του συνδέεται στην πηγή (5V) και η άλλη στη γείωση (GND). Αυτό θα έδινε σταθερή ροή ρεύματος και συνεπώς σταθερό ήχο. Στην εικόνα 34 που ακολουθεί, αντί την πηγή χρησιμοποιούμε μια (PWM) θύρα του Arduino, ώστε να έχουμε τη δυνατότητα να αυξομειώσουμε την ένταση του ρεύματος και συνεπώς τον παραγόμενο ήχο.



Εικόνα 34 – Κύκλωμα φύλλου εργασίας 14

Προγραμματισμός κυκλώματος

Το κύκλωμα αυτό είναι αρκετά απλό. Αρκεί να αρχικοποιήσουμε το pin του Arduino κι ακολούθως να δώσουμε τις κατάλληλες εντολές ώστε να δημιουργήσουμε διαφορετικά επίπεδα ήχων.

1. Κάνετε τις κατάλληλες αρχικοποιήσεις.
2. Σε μια επαναληπτική διαδικασία δώστε σταδιακά τιμές στο pin ώστε να ακούσετε μια γκάμα ήχων. Ξεκινήστε από το 0 και σταδιακά αυξήστε την ένταση μέχρι το 255, δίνοντας και κατάλληλο ενδιάμεσο χρόνο (π.χ. 500msec) ώστε να ακούτε τον κάθε ήχο.
3. Μπορείτε να δημιουργήσετε μετά και την αντίστροφη διαδικασία, δηλαδή να μειώνετε σταδιακά την ένταση του ήχου μέχρι αυτή να φτάσει στο 0.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδος (6.4.3), Καθυστέρηση (6.5.1), Δομή επανάληψης (6.9), Breadboard(7), Φόρτωση (8)

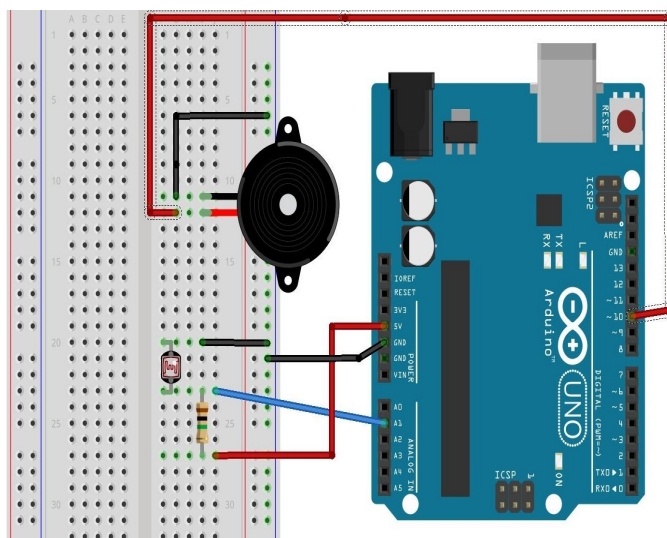
Μαθαίνω: Ηχείο (9.4)

10.15. Φύλλο εργασίας 15 – Ήχος & ανίχνευση φωτός (Sounder & Light Sensor)

Στο φύλλο εργασίας αυτό θα δημιουργήσουμε μια διάταξη που αντιδρά στα οπτικά ερεθίσματα. Αποτελείται από ένα ηχείο το οποίο αλλάζει τον ήχο του ανάλογα με το πόσο φως προσπίπτει σε μια φωτοευαίσθητη αντίσταση. Το φύλλο εργασίας είναι αντίστοιχο με αυτό που υλοποιήσατε με τη φωτοευαίσθητη αντίσταση και το led (φύλλο εργασίας 13).

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα ηχείο συνδεδεμένο σε ένα pin. Για να αναγνωρίσουμε τη φωτεινότητα θα χρησιμοποιήσουμε μια φωτοευαίσθητη αντίσταση (photoresistor). Η αντίσταση αυτή θα τοποθετηθεί σε σειρά μαζί με μια κανονική αντίσταση (περίπου 1MΩ), κι έτσι θα έχουμε μια διάταξη με τρία άκρα: ένα στην πηγή, ένα (αυτό που συνδέονται οι δύο αντιστάσεις) σε αναλογική είσοδο του Arduino (π.χ. A1) και ένα στη γείωση. Μπορείτε να δείτε τη διάταξη στην εικόνα 35 που ακολουθεί. Παρατηρήστε ότι η διάταξη προσομοιώνει ένα ποτενσιόμετρο (θυμηθείτε το αντίστοιχο φύλλο εργασίας), όπου το ρόλο του διακόπτη του ποτενσιόμετρου για τη μεταβολή της τάσης που διαβάζουμε θα παίζει η αντίσταση που μεταβάλλει τα χαρακτηριστικά της με το φως.



Εικόνα 35 – Κύκλωμα φύλλου εργασίας 15

Προγραμματισμός κυκλώματος

Ο προγραμματισμός του κυκλώματος αυτού είναι εντελώς παρόμοιος με αυτόν του φύλλου εργασίας 13, καθώς και αυτού με το ποτενσιόμετρο. Διαβάζουμε μια τιμή από την αναλογική είσοδο κι αυτή την τιμή την δίνουμε (προσαρμοσμένη) ως αναλογική έξοδο στο led μας. Προσαρμόστε κατάλληλα την είσοδό σας (τιμές 0-1023) στην έξοδο προς το led (τιμές 0-255), χρησιμοποιώντας την αντίστοιχη συνάρτηση.

Πειραματισμός 1: Πειραματιστείτε με τη φωτοευαίσθητη αντίσταση. Συνδέστε αρχικά το ένα πόδι της στην πηγή και το πόδι της απλής αντίστασης στη γείωση. Τρέξτε το πρόγραμμά σας. Ακολουθώντας, συνδέστε το πόδι της φωτοευαίσθητης αντίστασης στη γείωση και το πόδι της απλής στην πηγή. Τι παρατηρείτε να συμβαίνει;

Πειραματισμός 2: Συνδέστε τα πόδια της φωτοευαίσθητης αντίστασης με μακριά καλώδια ώστε να κινηθείτε στο χώρο με αυτή, κι ανιχνεύστε έτσι τις αλλαγές φωτεινότητας!

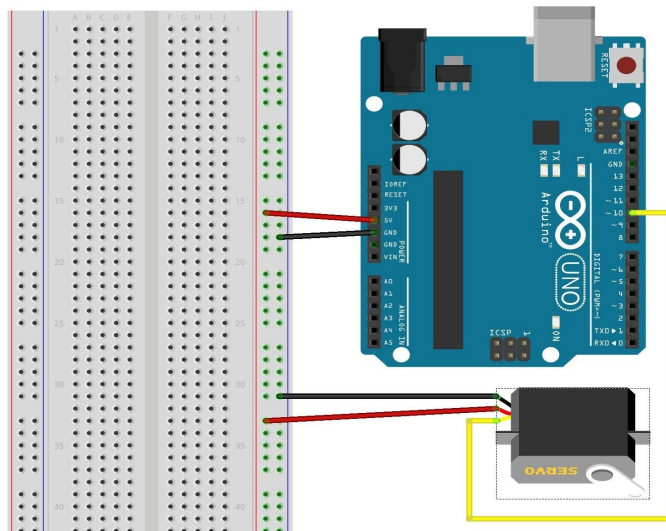
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική έξοδος (6.4.3.), Αναλογική είσοδος (6.4.4), Καθυστέρηση (6.5.1), Συνάρτηση αντιστοίχισης τιμών map (6.6), Breadboard(7), Φόρτωση (8), Ηχείο (9.4), Φωτοευαίσθητη αντίσταση (9.5)

10.16. Φύλλο εργασίας 16 – Κινώντας άξονες (Servos)

Ένας πολύ χρήσιμος μηχανισμός που μπορούμε να συνδέσουμε και να ελέγξουμε είναι το σέρβο, μια διάταξη που μπορεί να κινήσει έναν άξονα σε εύρος 180 μοιρών. Στην άσκηση αυτή θα δούμε τη βασική συνδεσμολογία και προγραμματισμό ενός σέρβο.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα σέρβο. Το σέρβο αποτελείται από τρία καλώδια: ένα για την πηγή (5V), ένα για τη γείωση κι ένα για τον έλεγχο του σέρβο μέσω ενός pin του Arduino. Τα χρώματα θα μας καθοδηγήσουν για τη συνδεσμολογία. Στο TowerPro SG90 έχουμε τρία καλώδια – ένα (πορτοκαλί) συνδέεται στην πηγή (5V), ένα (καφέ) στη γείωση (GND) κι ένα (κίτρινο) στο pin ελέγχου του. Μπορείτε να δείτε το κύκλωμα και τη συνδεσμολογία στην εικόνα 36 που ακολουθεί.



Εικόνα 36 – Κύκλωμα φύλλου εργασίας 16

Προγραμματισμός κυκλώματος

Για να μπορέσουμε να χρησιμοποιήσουμε ένα σέρβο, θα πρέπει να συμπεριλάβουμε στο πρόγραμμά μας την αντίστοιχη βιβλιοθήκη <Servo.h>. Επίσης, χρειάζεται να χρησιμοποιήσουμε συγκεκριμένες συναρτήσεις από τη βιβλιοθήκη αυτή. Αναζητήστε παρόμοιες πληροφορίες στο διαδίκτυο.

1. Δηλώστε μια μεταβλητή για το pin που συνδέσατε το servo
2. Συμπεριλάβετε στο πρόγραμμά σας την κατάλληλη βιβλιοθήκη για το σέρβο.
3. Δηλώστε μια μεταβλητή τύπου servo ώστε να αποκτήσετε πρόσβαση στο μηχανισμό του.
4. Κάνετε την κατάλληλη αρχικοποίηση για τη θύρα που χρησιμοποιείτε για το σέρβο. Για αυτή τη θύρα δεν δηλώνουμε στη συνάρτηση setup() ότι είναι έξοδος, αλλά την δεσμεύουμε με χρήση της εντολής attach(Pin).
5. Χρησιμοποιώντας της εντολή write() δώστε τις κατάλληλες εντολές στο servo σας, ώστε:
 - να ξεκινά στις 90 μοίρες
 - να στρίβει στις 180 μοίρες
 - να επιστρέφει στις 90 μοίρες
 - να στρίβει στις 0 μοίρες,λαμβάνοντας υπόψη και τους χρόνους που θα χρειαστούν για τις στροφές.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8)

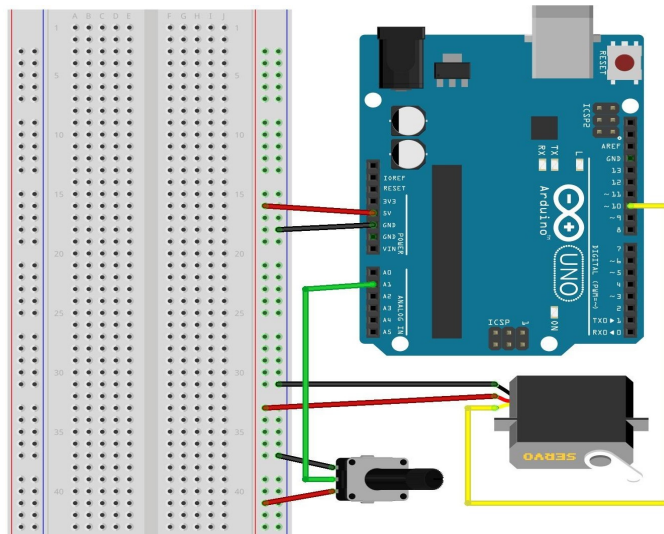
Μαθαίνω: Σέρβο (9.6)

10.17. Φύλλο εργασίας 17 – Κινώντας άξονες (Servos) με ποτενσιόμετρο

Στο φύλλο εργασίας αυτό θα επεκτείνουμε το κύκλωμα του προηγούμενου φύλλου εργασίας με ένα ποτενσιόμετρο. Η διάταξη που θα χρησιμοποιήσουμε για αυτό είναι παρόμοια με αυτή που χρησιμοποιήσαμε για το ποτενσιόμετρο σε προηγούμενα φύλλα εργασίας. Η διάταξή μας θα δουλεύει ως εξής: γυρίζοντας το χειριστήριο του ποτενσιόμετρου θα γυρνάει αντίστοιχα το σέρβο μας.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε ένα ποτενσιόμετρο και ένα σέρβο. Η συνδεσμολογία και για τα δύο αυτά έχει αναλυθεί σε προηγούμενα φύλλα εργασίας, οπότε ο αναγνώστης μπορεί να ανατρέξει σε αυτά για λεπτομέρειες. Στην εικόνα 37 μπορείτε να δείτε τη συνδεσμολογία. Το ποτενσιόμετρο τοποθετείται και απευθείας πάνω στο breadboard, αν έχει τα κατάλληλα άκρα.



Εικόνα 37 – Κύκλωμα φύλλου εργασίας 17

Προγραμματισμός κυκλώματος

Για την εργασία αυτή θα χρειαστεί να γίνουν όλες οι δηλώσεις μεταβλητών και αρχικοποιήσεις που είδαμε για το ποτενσιόμετρο και για το σέρβο. Ειδικότερα:

1. Δηλώστε μεταβλητές και την απαραίτητη βιβλιοθήκη που χρειαζόμαστε για το σέρβο, καθώς και κάντε όποιες αρχικοποιήσεις χρειάζονται.
2. Στις επαναλήψεις θα διαβάζετε την τιμή του ποτενσιόμετρου και, χρησιμοποιώντας την τιμή αυτή, θα δίνετε κατάλληλη εντολή στο σέρβο να κινηθεί.

Προσοχή: Η τιμή που παίρνουμε από το ποτενσιόμετρο είναι από 0 έως 1023 και πρέπει να την προσαρμόσουμε κατάλληλα για το σέρβο.

3. Δώστε την κατάλληλη χρονοκαυστήρηση ώστε να προλαβαίνει να κινηθεί το σέρβο.

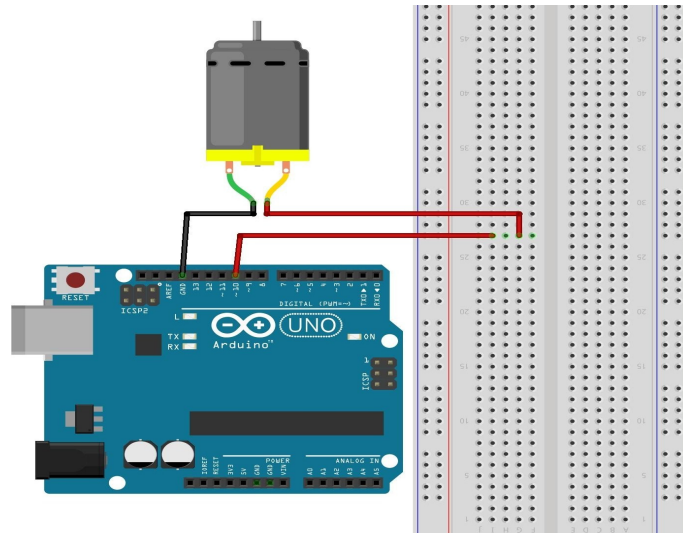
Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Αναλογική είσοδος (6.4.4.), Καθυστέρηση (6.5.1), Συνάρτηση αντιστοίχισης τιμών map (6.6), Breadboard(7), Φόρτωση (8), Ποτενσιόμετρο (9.3), Σέρβο (9.6)

10.18. Φύλλο εργασίας 18 – Ελέγχοντας κινητήρες (DC Motor)

Στο φύλλο εργασίας αυτό θα δούμε τη βασική συνδεσμολογία και έλεγχο κινητήρα συνεχούς ρεύματος (DC motor). Στο φύλλο εργασίας αυτό θα συνδέσουμε ένα κινητήρα και στο πρόγραμμά μας θα τον ενεργοποιήσουμε σταδιακά, δίνοντας αναλογικά τιμές στο pin στο οποίο τον έχουμε συνδέσει. Η σταδιακή αυτή ενεργοποίηση μπορεί να συγκριθεί με την αύξηση ταχύτητας.

Περιγραφή κυκλώματος

Για την εργασία αυτή θα χρειαστούμε έναν κινητήρα. Η πολικότητα στον κινητήρα δεν είναι σημαντική σε αυτό το στάδιο – ανάλογα πως συνδεθούν τα πόδια του θα έχουμε φορά δεξιόστροφη ή αριστερόστροφη του άξονά του. Μπορείτε να δείτε τη συνδεσμολογία στην εικόνα 38.



Εικόνα 38 – Κύκλωμα φύλλου εργασίας 18

Προγραμματισμός κυκλώματος

Ο προγραμματισμός του κυκλώματος αυτού είναι αρκετά απλός στο σημείο αυτό.

1. Ενεργοποιήστε το pin που συνδέσατε τον κινητήρα σας σύμφωνα με όσα γνωρίζετε, ως έξοδο.
2. Δώστε τάση στην έξοδό σας (pin κινητήρα) με χρήση της κατάλληλης συνάρτησης.
3. Δώστε αναλογικά στην έξοδό σας (pin κινητήρα) σταδιακά αυξανόμενες τιμές (0 – 255), με χρήση της κατάλληλης συνάρτησης.
4. Χρησιμοποιήστε μια λογική χρονοκαθυστέρηση ανάμεσα στις εντολές ώστε να μπορείτε να αντιληφθείτε τη διαφορά στην ταχύτητα περιστροφής.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8)

Μαθαίνω: Κινητήρας συνεχούς (9.7)

10.19. Φύλλο εργασίας 19 – Ελέγχοντας 2 ή περισσότερους κινητήρες

Για να ελέγξουμε ταυτόχρονα δύο ή περισσότερους κινητήρες συνήθως χρησιμοποιήσουμε μια Η-γέφυρα (H-bridge). Ένα κλασικό chip που μπορούμε να χρησιμοποιήσουμε για να ελέγξουμε δύο μοτέρ είναι το L298N. Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε δύο κινητήρες οι οποίοι θα ελεγχθούν από το chip αυτό. Οι δύο κινητήρες μπορούν άνετα να κατευθύνουν ένα αυτοκινητάκι ή ένα δίκτροχο*.

Περιγραφή κυκλώματος

Ανατρέξτε στην αντίστοιχη παράγραφο της θεωρίας για τη συνδεσμολογία του L298N. Πρέπει να προσέξουμε τις πολικότητες των κινητήρων, γιατί αν συνδεθούν ανάποδα θα γυρίζουν και με την αντίθετη φορά που θέλουμε.

Προγραμματισμός κυκλώματος

Αφού συνδέσετε τους κινητήρες και τη γέφυρα με τον τρόπο που περιγράφηκε παραπάνω, ξεκινήστε να γράψετε ένα πρόγραμμα που θα υλοποιεί βασικές λειτουργίες πορείας. Συγκεκριμένα:

1. Δημιουργήστε τις κατάλληλες μεταβλητές και αρχικοποιήσεις για τα pin σας.
2. Γράψτε τον κώδικα που θα κάνει τους κινητήρες σας να κινηθούν προς τα μπροστά.
3. Γράψτε τον κώδικα που θα κάνει τους κινητήρες σας να σταματήσουν εντελώς.
4. Γράψτε τον κώδικα που θα κάνει τους κινητήρες σας να κινηθούν προς τα πίσω.
5. Γράψτε τον κώδικα που θα κάνει το όχημά σας να στρίψει δεξιά. Για να γίνει αυτό θα πρέπει ο δεξιός κινητήρας να κινηθεί προς τα εμπρός ενώ ο αριστερός να μείνει ακίνητος.
6. Γράψτε τον κώδικα που θα κάνει το όχημά σας να στρίψει αριστερά. Για να γίνει αυτό θα πρέπει ο αριστερός κινητήρας να κινηθεί προς τα εμπρός ενώ ο δεξιός να μείνει ακίνητος.
7. Με τη βοήθεια του καθηγητή σας προσπαθήστε να δημιουργήσετε νέες συναρτήσεις με τις λειτουργίες που μόλις φτιάξατε, ώστε να τις καλείτε όταν εσείς θέλετε.
8. Πώς μπορείτε να φτιάξετε μια βιβλιοθήκη με αυτές; Τί άλλο θα έπρεπε να περιλαμβάνει η βιβλιοθήκη σας;

*Υπάρχει για 4 κινητήρες το L293D, το οποίο όμως δίνει μικρή ένταση ρεύματος και δεν συστήνεται, εκτός αν συντρέχει ειδικός λόγος να χρησιμοποιηθεί.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών (6.3), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8), Κινητήρας συνεχούς (9.7)

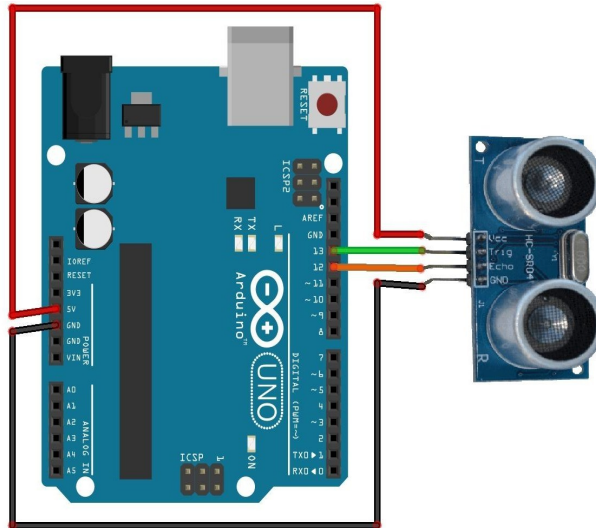
Μαθαίνω: Χρήση δύο κινητήρων (9.7.1)

10.20. Φύλλο εργασίας 20 - Χρησιμοποιώντας υπερήχους για τη μέτρηση μιας απόστασης

Στο φύλλο εργασίας αυτό θα χρησιμοποιήσουμε μια διάταξη υπερήχων, μετρώντας με τη βοήθειά της την απόσταση. Για το λόγο αυτό θα χρησιμοποιήσουμε τη διάταξη HC-SR04, η οποία παρουσιάστηκε στο αντίστοιχο κεφάλαιο της θεωρίας.

Περιγραφή κυκλώματος

Η διάταξη έχει τέσσερα σημεία σύνδεσης (βλ. εικόνα 39). Συνδέουμε την πηγή (5V), τη γείωση (GND) και δύο pin, ένα για την αποστολή του σήματος (trigger) κι ένα για τη λήψη (echo).



Εικόνα 39 – Κύκλωμα φύλλον εργασίας 20

Προγραμματισμός κυκλώματος

Συνδέστε κατάλληλα τους ακροδέκτες τις διάταξης. Για τον προγραμματισμό, θα χρειαστεί να στέλνουμε ένα σήμα και αφού περιμένουμε ελάχιστο χρόνο να το διαβάσουμε πίσω.

1. Ορίστε κατάλληλα τα pins ως εισόδου ή εξόδου.
2. Ενεργοποιήστε τη σειριακή οθόνη με ρυθμό 9600.
3. Ορίστε δύο τιμές για τα όρια της απόστασης που διαβάζετε (σε cm), π.χ. 0 και 200, για να ελέγχετε ότι διαβάζετε τιμές από 0 έως 2 μέτρα.
4. Μηδενίστε τον παλμό στο trigger για να είστε σίγουροι ότι δεν στέλνετε τίποτα αρχικά, με χρήση της συνάρτησης `digitalWrite()` και περιμένετε 2 μ s.
5. Στείλτε έναν παλμό στο trigger και περιμένετε 10 μ s.
6. Μηδενίστε τον παλμό, και διαβάστε από το echo τη διάρκεια του παλμού με χρήση της συνάρτησης `pulseIn(echoPin, HIGH)`;
7. Υπολογίστε την απόσταση που βρίσκεται μπροστά σας το εμπόδιο σύμφωνα με τον τύπο που αναφέραμε ($distance = duration/58.2$;)
8. Ελέγξτε αν η απόσταση βρίσκεται στα όρια που θέσατε.
9. Στείλτε την απόσταση στη σειριακή οθόνη ώστε να μπορέσετε να πειραματιστείτε.
10. Βάλτε μια καθυστέρηση 50 ms πριν ξεκινήσει ξανά να ελέγχει το πρόγραμμά σας.

Χρησιμοποιώ: Θύρες(5), Μεταβλητές (6.1), Διαχείριση Θυρών(6.3), Καθυστέρηση (6.5.1), Breadboard(7), Φόρτωση (8)

Μαθαίνω: Αισθητήρας υπερήχων (9.8), Καθυστέρηση (6.5.2)

10.21. Φύλλο εργασίας 21 – Σύνθετη κατασκευή

Στο φύλλο εργασίας αυτό θα συνδυάσουμε χαρακτηριστικά που έχουμε δουλέψει στα προηγούμενα φύλλα, με σκοπό να φτιάξουμε μία πιο σύνθετη κατασκευή. Οι μαθητές μπορούν να δουλέψουν σε ομάδες και να σκεφτούν πώς μπορούν να συνδυάσουν και να αξιοποιήσουν αυτά τα οποία αντιμετώπισαν στα προηγούμενα φύλλα εργασίας.

Παραδείγματα υλοποιήσεων:

- μπορείτε να υλοποιήσετε μια αυτοκινούμενη κατασκευή (όχημα με κινητήρες), η οποία ανιχνεύει αν υπάρχει εμπόδιο μπροστά της με μια διάταξη υπερήχων και σταματά ή αλλάζει πορεία.
- μπορείτε να υλοποιήσετε μια διάταξη που κινείται προς τη μεγαλύτερη φωτεινότητα, χρησιμοποιώντας ένα σέρβο για την περιστροφή και μια φωτοευαίσθητη αντίσταση για να μετράτε τη φωτεινότητα.

Αισθητήρες

Με τις γνώσεις που έχετε μέχρι στιγμής μπορείτε να αναζητήσετε περισσότερες πληροφορίες (με τη βοήθεια του καθηγητή σας) για περισσότερους αισθητήρες που μπορείτε να χρησιμοποιήσετε με το Arduino σας, ώστε να κάνετε πιο σύνθετες κατασκευές. Όπως θα δείτε, υπάρχει μεγάλη ποικιλία αισθητήρων που μπορείτε να χρησιμοποιήσετε. Ανάλογα με τη διαθεσιμότητά τους στο δικό σας εργαστήριο, σχεδιάστε μια σύνθετη κατασκευή και ξεκινήστε την υλοποίησή της. Καλή επιτυχία!

11. Αναφορές

11.1. Βιβλιογραφία

- Banzi, M. (2011). *Getting Started with Arduino (2nd Edition)*. USA: Make:Books(O' Reilly)
- Borchers, J. (2013). *Arduino In a Nutshell (v.1.8)*. Ανακτήθηκε 27/12/2014 από <http://hci.rwth-aachen.de/arduino>
- McRoberts, M. (2013). *Beginning Arduino (2nd Edition)*. USA: Apress
- SparkFun Electronics (2012). *Introduction to Arduino Educational Material*. Ανακτήθηκε 27/12/2014 από https://dlnmh9ip6v2uc.cloudfront.net/learn/materials/1/Arduino_final_handout.pdf

11.2. Ιστότοποι με υλικό

- Ad Hoc Node: <http://adhocnode.com/category/arduino/>
- ArduBlock: <http://blog.ardublock.com/>
- Arduino: <http://www.arduino.cc>
- ArduinoMio: <http://www.mastrohora.it/arduiniomio/en/xuino.php>
- BlocklyDuino: <http://www.gasolin.idv.tw/public/blockly/demos/blocklyduino/index.html>
- Fritzing: <http://fritzing.org>
- Scratch: <http://scratch.mit.edu>
- Scratch4Arduino: <http://s4a.cat/>
- SparkFun: <https://learn.sparkfun.com/>
- Tolabaki: http://wiki.tolabaki.gr/w/To_Labaki>About

Παράρτημα – Ενδεικτικά προγράμματα για τα φύλλα εργασίας

Φύλλο εργασίας 1 – Το Led που αναβοσβήνει

```
int ledPin = 10;

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

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

Φύλλο εργασίας 2 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out)

```
int ledPin = 10;
int brightness = 0;

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

void loop() {
  if (brightness == 255) {
    brightness = 0;
  };
  analogWrite(ledPin, brightness);
  brightness = brightness + 5;
  delay(100);
}
```

Φύλλο εργασίας 3 – Σταδιακή αύξηση και μείωση φωτεινότητας (Fade in-Fade out), Αναβόσβημα στις άκρες (Blink at peaks)

```
int ledPin = 10;
int brightness = 255;
int interval = 5; // διάστημα που θα ανεβαίνει ή κατεβαίνει η φωτεινότητα

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

```

void loop() {
    if (brightness == 255 || brightness == 0) {
        // αναβοσβήνει (blink) δυο φορές
        digitalWrite(ledPin, HIGH);
        delay(100);
        digitalWrite(ledPin, LOW);
        delay(100);
        digitalWrite(ledPin, HIGH);
        delay(100);
        digitalWrite(ledPin, LOW);
        delay(100);
        /* αλλάζω πρόσημο στο διάστημα, ώστε αν ανέβαινε μέχρι τώρα(θετικό) να αρχίσει να
        κατεβαίνει(αρνητικό) κι αντίστροφα */
        interval = (-1)*interval;
    };
    analogWrite(ledPin, brightness);
    brightness = brightness + interval;
    delay(100);
}

```

Φύλλο εργασίας 4 – Σταδιακή αύξηση και μείωση φωτεινότητας με ποτενσιόμετρο

```

int ledPin = 10;
int potPin = A4;
int potValue = 0; // μεταβλητή για την τιμή που θα διαβάσω από το ποτενσιόμετρο

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

void loop() {
    potValue = analogRead(potPin);
    analogWrite(ledPin, potValue/4);
    delay(50);
}

```

Φύλλο εργασίας 5 – Χρησιμοποιώντας τη σειριακή οθόνη

```

int i;

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

void loop() {
    Serial.print("Η epikoinwnia ksekinhse");
    for(i=1;i<=20;i++) {

```

```

        Serial.println(i);
        delay(100);
    }
    for(i=1;i<=20;i++) {
        Serial.print("Twra vlepw thn epanalhpsh: ");
        Serial.println(i);
        delay(200);
    }
}

```

Φύλλο εργασίας 6 – Παίζοντας με τα χρώματα (RGB Led)

```

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

void setup() {
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop() {
    // άναψε το κόκκινο
    analogWrite(redPin, 255);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 0);
    delay(2000);

    // άναψε το πράσινο
    analogWrite(redPin, 0);
    analogWrite(greenPin, 255);
    analogWrite(bluePin, 0);
    delay(2000);

    // άναψε το μπλε
    analogWrite(redPin, 0);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 255);
    delay(2000);
}

```

Φύλλο εργασίας 7 – Κατασκευάζοντας ένα RGB Led

```

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

```

```

int i;

void setup() {
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop() {
    // άναψε το κόκκινο
    analogWrite(redPin, 255);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 0);
    delay(2000);

    // άναψε το πράσινο
    analogWrite(redPin, 0);
    analogWrite(greenPin, 255);
    analogWrite(bluePin, 0);
    delay(2000);

    // άναψε το μπλε
    analogWrite(redPin, 0);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 255);
    delay(2000);
}

```

Φύλλο εργασίας 8 – Τα φανάρια κυκλοφορίας

```

int ledRed = 11;
int ledOrange = 10;
int ledGreen = 9;

void setup() {
    pinMode(ledRed, OUTPUT);
    pinMode(ledOrange, OUTPUT);
    pinMode(ledGreen, OUTPUT);
}

void loop() {
    // κόκκινο για 3 δευτερόλεπτα
    digitalWrite(ledRed, HIGH);
    digitalWrite(ledOrange, LOW);
    digitalWrite(ledGreen, LOW);
    delay(3000);

    // πράσινο για 5 δευτερόλεπτα
    digitalWrite(ledRed, LOW);

```

```

digitalWrite(ledOrange, LOW);
digitalWrite(ledGreen, HIGH);
delay(5000);

// πορτοκαλί για 1 δευτερόλεπτο
digitalWrite(ledRed, LOW);
digitalWrite(ledOrange, HIGH);
digitalWrite(ledGreen, LOW);
delay(1000);
}

```

Φύλλο εργασίας 9 – Τα φανάρια κυκλοφορίας με φανάρι πεζών

```

int ledRed = 11;
int ledOrange = 10;
int ledGreen = 9;
int pedRed = 4;
int pedGreen = 3;

void setup() {
  pinMode(ledRed, OUTPUT);
  pinMode(ledOrange, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
}

void loop() {
  // κόκκινο για 3 δευτερόλεπτα, πράσινο στους πεζούς
  digitalWrite(ledRed, HIGH);
  digitalWrite(ledOrange, LOW);
  digitalWrite(ledGreen, LOW);
  digitalWrite(pedRed, LOW);
  digitalWrite(pedGreen, HIGH);
  delay(3000);

  // κόκκινο στους πεζούς, περιμένω για 1 δευτερόλεπτο πριν δώσω πράσινο στα αμάξια
  digitalWrite(pedRed, HIGH);
  digitalWrite(pedGreen, LOW);
  delay(1000);

  // πράσινο κυκλοφορίας για 5 δευτερόλεπτα
  digitalWrite(ledRed, LOW);
  digitalWrite(ledOrange, LOW);
  digitalWrite(ledGreen, HIGH);
  delay(5000);

  // πορτοκαλί για 1 δευτερόλεπτο
  digitalWrite(ledRed, LOW);

```



```

digitalWrite(ledOrange, HIGH);
digitalWrite(ledGreen, LOW);
delay(1000);
}

```

Φύλλο εργασίας 10 – Τα φανάρια κυκλοφορίας με φανάρι πεζών και κουμπί διακοπής

```

int ledRed = 11;
int ledOrange = 10;
int ledGreen = 9;
int pedRed = 4;
int pedGreen = 3;
int btn = 5; //συνδέω το κουμπί στο pin 5
int lastPedPass = 0; //πότε διακόπηκε τελευταία φορά η κυκλοφορία για να περάσει πεζός
int btnState; //για να διαβάζουμε αν πατήθηκε το κουμπί μας

void setup() {
    pinMode(ledRed, OUTPUT);
    pinMode(ledOrange, OUTPUT);
    pinMode(ledGreen, OUTPUT);
    pinMode(pedRed, OUTPUT);
    pinMode(pedGreen, OUTPUT);
    pinMode(btn, INPUT);

    // κόκκινο στους πεζούς και πράσινο στα αυτοκίνητα αρχικά
    digitalWrite(pedRed, HIGH);
    digitalWrite(pedGreen, LOW);
    digitalWrite(ledRed, LOW);
    digitalWrite(ledOrange, LOW);
    digitalWrite(ledGreen, HIGH);
}

void loop() {
    int btnState = digitalRead(btn);
    if (btnState == HIGH && (millis() - lastPedPass) > 5000) {
        // αν πατήθηκε το κουμπί και ο χρόνος από την τελευταία αλλαγή είναι πάνω από 5 sec

        // πορτοκαλί για 1 δευτερόλεπτο
        digitalWrite(ledRed, LOW);
        digitalWrite(ledOrange, HIGH);
        digitalWrite(ledGreen, LOW);
        delay(1000);

        // κόκκινο στα αμάξια για 5 δευτερόλεπτα, περιμένω 1 sec και δίνω πράσινο στους πεζούς
        digitalWrite(ledRed, HIGH);
        digitalWrite(ledOrange, LOW);
        digitalWrite(ledGreen, LOW);
        delay(1000);
        digitalWrite(pedRed, LOW);
    }
}

```

```

        digitalWrite(pedGreen, HIGH);
        delay(3000);

        // κόκκινο στους πεζούς, περιμένω για 1 δευτερόλεπτο πριν δώσω πράσινο στα αμάξια
        digitalWrite(pedRed, HIGH);
        digitalWrite(pedGreen, LOW);
        delay(1000);

        // πράσινο κυκλοφορίας για 5 δευτερόλεπτα
        digitalWrite(ledRed, LOW);
        digitalWrite(ledOrange, LOW);
        digitalWrite(ledGreen, HIGH);
        lastPedPass = millis(); // κρατάω το χρόνο της τελευταίας αλλαγής φαναριών
    }
}

```

Φύλλο εργασίας 11 – Εφέ, κνηγώντας τη λάμψη (Led chase effect)

// Πρόγραμμα με μεταβλητές, χωρίς τη χρήση πίνακα

```

int lPin1 = 6;
int lPin2 = 7;
int lPin3 = 8;
int lPin4 = 9;
int lPin5 = 10;
int lPin6 = 11;
int lPin7 = 12;
int lPin8 = 13;

void setup() {
    pinMode(lPin1, OUTPUT);
    pinMode(lPin2, OUTPUT);
    pinMode(lPin3, OUTPUT);
    pinMode(lPin4, OUTPUT);
    pinMode(lPin5, OUTPUT);
    pinMode(lPin6, OUTPUT);
    pinMode(lPin7, OUTPUT);
    pinMode(lPin8, OUTPUT);
}

void loop() {
    digitalWrite(lPin1, HIGH);
    delay(200);
    digitalWrite(lPin1, LOW);
    digitalWrite(lPin2, HIGH);
    delay(200);
    digitalWrite(lPin2, LOW);
    digitalWrite(lPin3, HIGH);
    delay(200);
}

```

```

digitalWrite(IPin3, LOW);
digitalWrite(IPin4, HIGH);
delay(200);
digitalWrite(IPin4, LOW);
digitalWrite(IPin5, HIGH);
delay(200);
digitalWrite(IPin5, LOW);
digitalWrite(IPin6, HIGH);
delay(200);
digitalWrite(IPin6, LOW);
digitalWrite(IPin7, HIGH);
delay(200);
digitalWrite(IPin7, LOW);
digitalWrite(IPin8, HIGH);
delay(200);
digitalWrite(IPin8, LOW);
}

```

// Πρόγραμμα με τη χρήση πίνακα

```

int ledPin[8]; // δημιουργώ πίνακα 8 θέσεων
int i;

void setup() {
    for (i=0;i<8;i++) {
        ledPin[i] = i+6; // βάζω ως τιμές τα Pins από 6 έως 13
        pinMode(ledPin[i], OUTPUT);
    }
}

void loop() {
    for (i=0;i<=7;i++) {
        digitalWrite(ledPin[i], HIGH);
        delay(200);
        digitalWrite(ledPin[i], LOW);
    }
}

```

Φύλλο εργασίας 12 – Εφέ, κυνηγώντας τη λάμψη (Led chase effect) με ποτενσιόμετρο

```

int ledPin[8]; // δημιουργώ πίνακα 8 θέσεων
int potPin = A4; // το pin του ποτενσιόμετρου
int interval = 200; // αρχική τιμή 200 στην αλλαγή
int i;

void setup() {
    for (i=0;i<8;i++) {
        ledPin[i] = i+6; // βάζω ως τιμές τα Pins από 6 έως 13
    }
}

```

```

        pinMode(ledPin[i], OUTPUT);
    };
    pinMode(potPin, INPUT);
}

void loop() {
    for (i=0;i<=7;i++) {
        interval = analogRead(potPin);
        digitalWrite(ledPin[i], HIGH);
        delay(interval);
        digitalWrite(ledPin[i], LOW);
    }
}

```

Φύλλο εργασίας 13 – Εφέ, ανιχνεύοντας το φως (Led effect + light sensor)

```

int ledPin = 10;
int resPin = A1;
int resVal = 0; // εδώ θα διαβάσουμε την τιμή της αντίστασης
int ledVal = 0;

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

void loop() {
    resVal = analogRead(resPin);
    ledVal = map(resVal, 0, 1023, 0, 255);
    analogWrite(ledPin, ledVal);
    // αν δεν χρησιμοποιούσα την map, θα έγραφα analogWrite(ledPin, resVal / 4);
    delay(50);
}

```

Φύλλο εργασίας 14 – Χρησιμοποιώντας τον ήχο (Sounder)

```

int sndPin = 10;
int i;

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

void loop() {
    for (i=0;i<=255;i=i+5) {
        analogWrite(sndPin, i);
    }
}

```

```

        delay(50);
    }
}

```

Φύλλο εργασίας 15 – Ήχος & αντίχρεση φωτός (Sounder & Light Sensor)

```

int sndPin = 10;
int resPin = A1;
int resVal = 0; // εδώ θα διαβάζουμε την τιμή της αντίστασης
int ledVal = 0;

void setup() {
    pinMode(sndPin, OUTPUT);
    pinMode(resPin, INPUT);
}

void loop() {
    resVal = analogRead(resPin);
    ledVal = map(resVal, 0, 1023, 0, 255);
    analogWrite(sndPin, ledVal);
    // αν δεν χρησιμοποιούσα την map, θα έγραφα analogWrite(sndPin, resVal / 4);
    delay(50);
}

```

Φύλλο εργασίας 16 – Κινώντας άξονες (Servos)

```

#include <Servo.h>

int sPin = 10;
Servo s;

void setup() {
    s.attach(sPin);
}

void loop() {
    s.write(90);
    delay(50);
    s.write(179);
    delay(50);
    s.write(90);
    delay(50);
    s.write(0);
    delay(50);
}

```

Φύλλο εργασίας 17 – Κινώντας άξονες (Servos) με ποτενσιόμετρο

```
#include <Servo.h>

int sPin = 10;
int potPin = A1; // το pin του ποτενσιόμετρου

Servo s;
int potVal = 0; // η τιμή του ποτενσιόμετρου
int sVal = 0;

void setup() {
    s.attach(sPin);
    pinMode(potPin, INPUT);
}

void loop() {
    potVal = analogRead(potPin);
    sVal = map(potVal, 0, 1023, 0, 180);
    s.write(sVal);
    delay(50);
}
```

Φύλλο εργασίας 18 – Ελέγχοντας κινητήρες (DC Motor)

```
int mtrPin = 10;
int i;

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

void loop() {
    digitalWrite(mtrPin, HIGH);
    delay(3000);
    digitalWrite(mtrPin, LOW);
    for (i=0; i<=255; i=i+5) {
        analogWrite(mtrPin, i);
        delay(100);
    }
}
```

Φύλλο εργασίας 19 – Ελέγχοντας 2 ή περισσότερους κινητήρες

```
//Κινητήρας Α
int dir1PinA = 7;
int dir2PinA = 5;
```



```

int speedPinA = 6;

//Κινητήρας B
int dir1PinB = 4;
int dir2PinB = 2;
int speedPinB = 3;

int speed = 255;
int time = 5000;

void setup() {
    pinMode (dir1PinA, OUTPUT);
    pinMode (dir2PinA, OUTPUT);
    pinMode (speedPinA, OUTPUT);
    pinMode (dir1PinB, OUTPUT);
    pinMode (dir2PinB, OUTPUT);
    pinMode (speedPinB, OUTPUT);
}

void loop() {
    // όλοι οι κινητήρες προς τα μπροστά
    analogWrite (speedPinA, speed);
    analogWrite (speedPinB, speed);
    digitalWrite (dir1PinA , HIGH);
    digitalWrite (dir2PinA, LOW);
    digitalWrite (dir1PinB, HIGH);
    digitalWrite (dir2PinB, LOW);
    delay(time);

    // όλοι οι κινητήρες προς τα πίσω
    analogWrite (speedPinA, speed);
    analogWrite (speedPinB, speed);
    digitalWrite (dir1PinA, LOW);
    digitalWrite (dir2PinA, HIGH);
    digitalWrite (dir1PinB, LOW);
    digitalWrite (dir2PinB, HIGH);
    delay(time);

    // σταμάτημα όλων των κινητήρων
    digitalWrite (speedPinA, LOW);
    digitalWrite (speedPinB, LOW);
    digitalWrite (dir1PinA , LOW);
    digitalWrite (dir2PinA, LOW);
    digitalWrite (dir1PinB, LOW);
    digitalWrite (dir2PinB, LOW);
    delay(time);

    // ο δεξιός κινητήρας μπροστά
    analogWrite (speedPinA, speed);
    analogWrite (speedPinB, LOW);

```

```

digitalWrite (dir1PinA , HIGH);
digitalWrite (dir2PinA, LOW);
digitalWrite (dir1PinB, HIGH);
digitalWrite (dir2PinB, LOW);
delay(time);

// ο αριστερός κινητήρας μπροστά
analogWrite (speedPinA, LOW);
analogWrite (speedPinB, speed);
digitalWrite (dir1PinA , HIGH);
digitalWrite (dir2PinA, LOW);
digitalWrite (dir1PinB, HIGH);
digitalWrite (dir2PinB, LOW);
delay(time);

// σταμάτημα όλων των κινητήρων
digitalWrite (speedPinA, 255);
digitalWrite (speedPinB, 255);
digitalWrite (dir1PinA , LOW);
digitalWrite (dir2PinA, LOW);
digitalWrite (dir1PinB, LOW);
digitalWrite (dir2PinB, LOW);
delay(time);
}

```

Φύλλο εργασίας 20 – Χρησιμοποιώντας υπερήχους για τη μέτρηση μιας απόστασης

```

int echoPin = 12; // Echo Pin
int trigPin = 13; // Trigger Pin

int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
}

void loop() {
    /* Στέλνουμε κύματα ήχου και τα παίρνουμε πίσω, προσπαθώντας έτσι να μετρήσουμε την
    απόσταση από την ταχύτητα που έχουν. */
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
}

```

```

//Υπολογισμός απόστασης (σε cm) βασιζόμενοι στην ταχύτητα του ήχου.
distance = duration/58.2;
if (distance >= maximumRange || distance <= minimumRange){
    /* εκτός ορίων */
    Serial.println("Εκτός ορίων");
    delay(100);
}
else {
    /* επιτυχής ανάγνωση */
    Serial.println(distance);
    delay(50);
};

//Καθυστέρηση 50ms πριν την επόμενη ανάγνωση
delay(50);
}

```




ISBN 978-960-93-6760-8
ΕΜΜ. ΠΟΥΛΑΚΗΣ