Παράρτημα Β΄

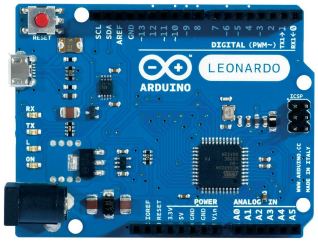
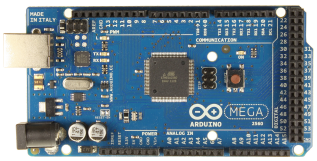
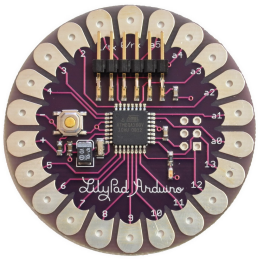
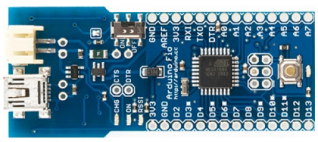
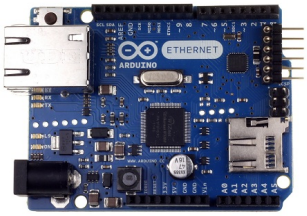
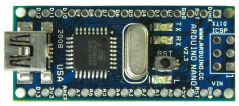
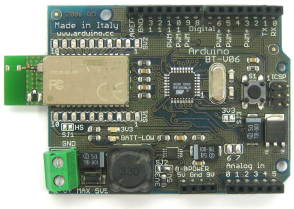
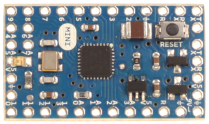
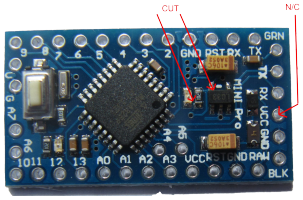
Σχεδίαση με μικρο-ελεγκτή 8bit

Αν και οι επεξεργαστές που χρησιμοποιούνται στους τυπικούς σταθμούς εργασίας ή στους διακομιστές ή στα έξυπνα τηλέφωνα, είναι αρχιτεκτονικής 32 bit ή 64 bit, στα ενσωματωμένα συστήματα χρησιμοποιούνται αρχιτεκτονικές από 8 bit έως και 64bit. Ένας αρκετά δημοφιλής, αξιόπιστος και οικονομικός επεξεργαστής 8 bit είναι ο ATmega328P, που αποτελεί την καρδιά της αναπτυξιακής πλατφόρμας Arduino UNO. Σε αυτό το παράρτημα, θα γίνει η παρουσίαση του Arduino UNO και στη συνέχεια θα ακολουθήσουν οι ασκήσεις εμβάθυνσης σε αυτή την αρχιτεκτονική.

Β΄.1 Η αναπτυξιακή πλατφόρμα ARDUINO

Στην ενότητα αυτή, παρουσιάζεται ο μικροελεγκτής Arduino που χρησιμοποιήθηκε, τα χαρακτηριστικά του και μερικά από τα διαθέσιμα εξαρτήματα που μπορούν να συνδεθούν απ' ευθείας με αυτό. Το Arduino [98] είναι μία ανοιχτού λογισμικού πλατφόρμα πρωτοτύπων ηλεκτρονικών συσκευών που βασίζονται στην ευελιξία και στην ευκολία χρήσης υλικού και λογισμικού. Το Arduino μπορεί να αλληλεπιδρά με το περιβάλλον κάνοντας λήψη σημάτων μέσα από μια ποικιλία αισθητήρων. Το arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη διαλογικών λειτουργιών, με είσοδο από μια πληθώρα πηγών (διακόπτες, αισθητήρες,..) και έλεγχο φυσικών αντικειμένων (φώτα, κινητήρες,..). Το arduino μπορεί να είναι αυτόνομο ή να επικοινωνεί με άλλα arduino ή υπολογιστικά συστήματα Τα έργα που βασίζονται σε αυτόν τον μικροελεγκτή, μπορούν να είναι αυτόνομα ή μπορούν να επικοινωνούν με το λογισμικό που τρέχει σε έναν υπολογιστή (π.χ. Flash, Processing, MaxMSP). Το arduino είναι ένα εργαλείο που μας επιτρέπει να κατασκευάσουμε υπολογιστικά συστήματα που μπορούν να αισθανθούν και να ελέγξουν το φυσικό κόσμο πολύ πιο εύκολα από ότι αν χρησιμοποιούσαμε έναν τυπικό υπολογιστή γραφείου. Είναι μια αρχιτεκτονική που

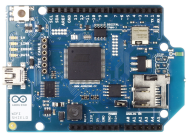
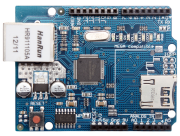
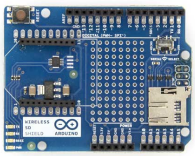
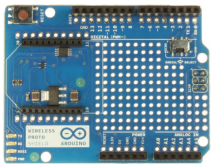
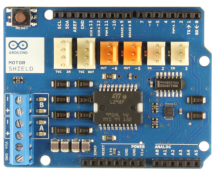
βασίζεται σε ανοιχτό κώδικα, μια πλακέτα μικρο-επεξεργαστή και ένα αναπτυξιακό περιβάλλον για τη συγγραφή προγράμματος για την πλακέτα. Η οικογένεια Arduino αποτελείται από πολλές αναπτυξιακές πλακέτες, διαφορετικών χαρακτηριστικών (Εικόνα Β΄.1). Η πιο δημοφιλής αναπτυξιακή πλακέτα είναι η Arduino UNO (Εικόνα Β΄.1).

Arduino Leonardo	Arduino Mega	Arduino LilyPad
		
Arduino Fio	**Arduino Ethernet**	**Arduino Nano**
		
Arduino BT	**Arduino Mini**	**Arduino Pro Mini**
		

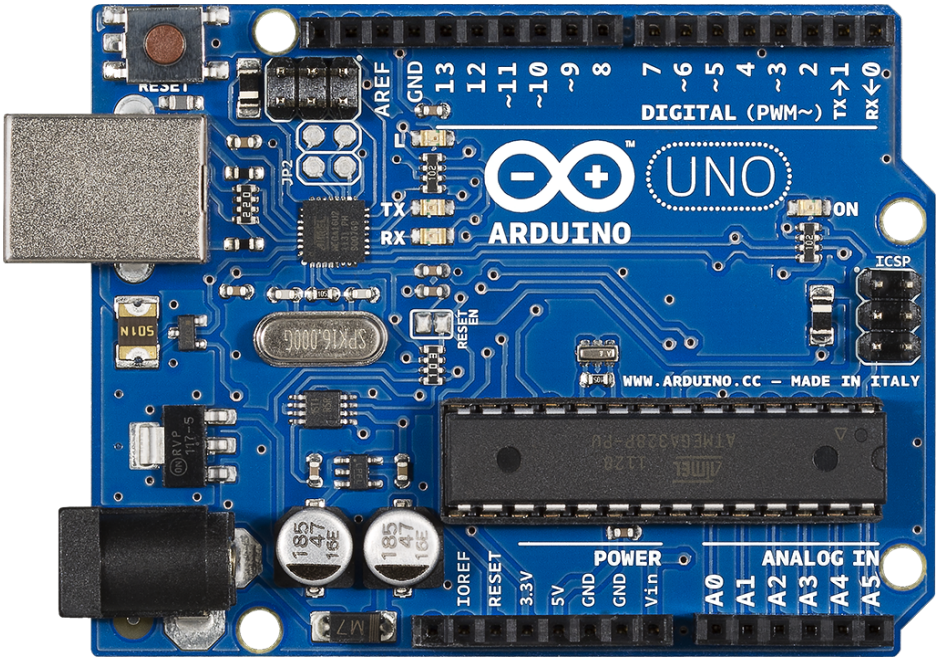
Πίνακας Β΄.1: Η οικογένεια αναπτυξιακών πλακετών Arduino

Η ευκολία σχεδίασης και χρήσης ενσωματωμένων συστημάτων που βασίζονται στην οικογένεια arduino, οφείλεται κατά μεγάλο ποσοστό στις δυνατότητες επέκτασης που παρέχονται με τη χρήση πλακετών επέκτασης (που ονομάζονται ως ‘shields’ στην αγγλική γλώσσα). Shields είναι τα εξαρτήματα που συνδέονται απευθείας με όλα τα pin του arduino. Μερικά από αυτά φαίνονται στην Εικόνα Β΄.2.

Στις εργαστηριακές ασκήσεις που βρίσκονται σε αυτό το παράρτημα, χρησιμοποιείται η αναπτυξιακή πλακέτα Arduino UNO. Τα χαρακτηριστικά αυτής

Arduino Wifi Shield		Το Arduino WiFi Shield συνδέει το Arduino στο διαδίκτυο ασύρματα.
Arduino Ethernet Shield		Το Arduino Ethernet Shield συνδέει το Arduino στο διαδίκτυο με ένα RJ45 καλώδιο.
Wireless SD Shield		Το Wireless SD Shield επιτρέπει σε μια πλακέτα Arduino να επικοινωνεί ασύρματα με μια ασύρματη μονάδα. Η μονάδα μπορεί να επικοινωνήσει έως και 100 πόδια σε εσωτερικούς χώρους ή σε εξωτερικούς χώρους ως 300 πόδια. Η μονάδα περιλαμβάνει μια θύρα υποδοχής SD
Wireless Proto Shield		Το Wireless Proto Shield επιτρέπει στο Arduino να επικοινωνεί ασύρματα με μια ασύρματη μονάδα. Η μονάδα μπορεί να επικοινωνήσει έως και 100 πόδια σε εσωτερικούς χώρους ή σε εξωτερικούς χώρους ως 300 πόδια. Η μονάδα δεν περιλαμβάνει θύρα υποδοχής SD.
Arduino Motor Shield		Το Arduino Motor Shield επιτρέπει την οδήγηση δύο DC κινητήρων από την ίδια συσκευή, ελέγχοντας την ταχύτητα και την κατεύθυνση του καθενός ξεχωριστά.

Πίνακας Β΄.2: Τυπικές πλακέτες επέκτασης (shields) arduino



Σχήμα Β΄.1: Η αναπτυξιακή πλατφόρμα Arduino αποτελείται από τον μικροεπεξεργαστή 8bit ATMEGA328P. Στην εικόνα φαίνεται η έκδοση UNO.

της πλακέτας εμφανίζονται στον Πίνακα Β΄.3.

Το Arduino UNO μπορεί να τροφοδοτηθεί με DC ρεύμα είτε από τον υπολογιστή μέσω της σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm που βρίσκεται στην κάτω αριστερή γωνία. Για την αποφυγή προβλημάτων, η εξωτερική τροφοδοσία θα πρέπει να είναι από 7 ως 12V. Το Σχήμα Β΄.2 παρουσιάζει τις εισόδους και εξόδους τροφοδοσίας του Arduino UNO.

Οι ακροδέκτες τροφοδοσίας είναι οι ακόλουθοι:

Vin : Η τάση εισόδου της πλακέτας όταν χρησιμοποιεί εξωτερική πηγή ενέργειας. Η τροφοδοσία τάσης γίνεται μέσω αυτού του ακροδέκτη.

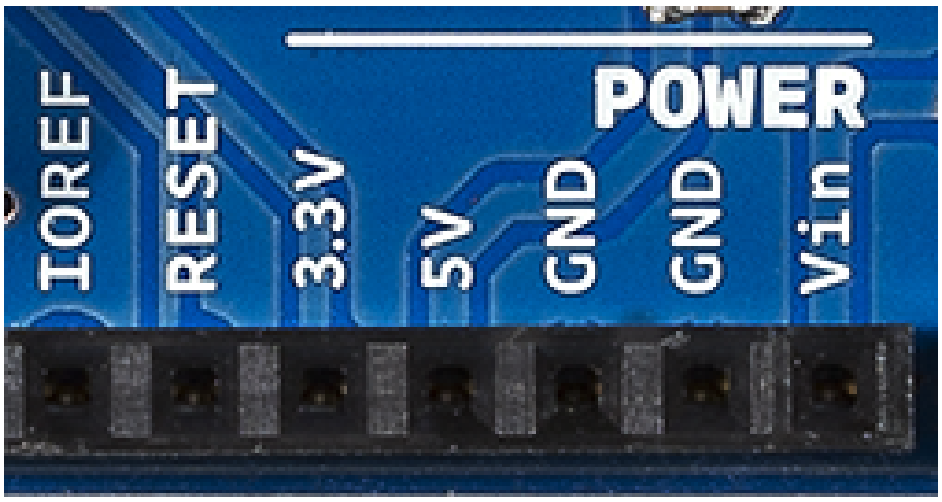
5V : Η τάση που χρησιμοποιείται από τα διάφορα μέρη της πλακέτας και το μικροελεγκτή είναι 5V. Η τάση αυτή, την οποία δίνει αυτός ο ακροδέκτης, είναι είτε η τάση 5V που δίνει η σύνδεση με USB, είτε η ρυθμισμένη τάση που δίνεται μέσω του Vin.

GND : Είσοδοι γείωσης.

Ο μικροεπεξεργαστής ATmega328 έχει τρεις ομάδες μνήμης. Διαθέτει flash memory, στην οποία αποθηκεύονται τα Arduino sketch, SRAM (static random

Πίνακας Β΄.3: Χαρακτηριστικά της αναπτυξιακής πλακέτας Arduino UNO R3

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



Σχήμα Β΄.2: Είσοδοι/Εξοδοι Τροφοδοσίας Arduino UNO.

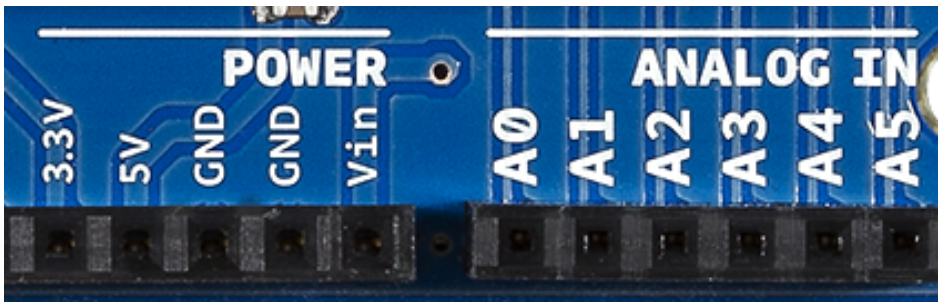
access memory), στην οποία δημιουργείται το sketch και χρησιμοποιεί τις μεταβλητές όταν τρέχει, και EEPROM, η οποία χρησιμοποιείται από τους προγραμματιστές για την αποθήκευση μακροχρόνιων πληροφοριών. **2KB μνήμης SRAM**: Η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κλπ Η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή πατηθεί το κουμπί επανεκκίνησης. **1KB μνήμης EEPROM**: Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τα προγράμματα. Σε αντίθεση με την SRAM, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης. **32KB μνήμης Flash**: 2 KB χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware είναι αναγκαίο για την εγκατάσταση προγραμμάτων στο μικροελεγκτή μέσω της θύρας USB. Τα υπόλοιπα 30KB της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

Η αναπτυξιακή πλακέτα Arduino UNO έχει 14 ψηφιακούς ακροδέκτες (Σχήμα Β΄.3).

Σχήμα Β΄.3: Οι ψηφιακοί ακροδέκτες του Arduino UNO.

Όλοι οι ψηφιακοί ακροδέκτες μπορεί να χρησιμοποιηθούν για είσοδο και έξοδο ψηφιακών τιμών. Το Arduino UNO χρησιμοποιεί 5V τάση στους ακροδέκτες, οπότε αν ένας ακροδέκτης εισόδου φέρει τάση 5V διαβάζεται ως '1' ενώ διαφορετικά διαβάζεται ως 0. Αντίστοιχα, ο ακροδέκτης εξόδου γράφει το λογικό '1' ως τάση +5V, ενώ το '0' αντιστοιχεί στη γείωση. Εκτός από τη γενική λειτουργία των ακροδεκτών εισόδου εξόδου, κάποιοι ακροδέκτες έχουν επιπρόσθετες λειτουργίες. Οι ακροδέκτες αυτοί περιγράφονται στη συνέχεια. **Ακροδέκτες 0 και 1**: λειτουργούν ως RX και TX της σειριακής θύρας όταν το πρόγραμμά ενεργοποιεί τη σειριακή θύρα. Έτσι, όταν το πρόγραμμά στέλνει δεδομένα στη σειριακή θύρα, αυτά προωθούνται και στη θύρα USB μέσω του ελεγκτή Serial-Over-USB, αλλά και στον ακροδέκτη 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή. Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά ενεργοποιήσει το σειριακό interface, χάνει 2 ψηφιακές εισόδους/εξόδους η πλατφόρμα. **Ακροδέκτες 2 και 3**: λειτουργούν και ως εξωτερικά interrupt (interrupt

0 και 1 αντίστοιχα). Ρυθμίζονται μέσα από το πρόγραμμά ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει άμεσα και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας. **Ακροδέκτες 3, 5, 6, 9, 10 και 11**: μπορούν να λειτουργήσουν και ως ψευδό- αναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation).



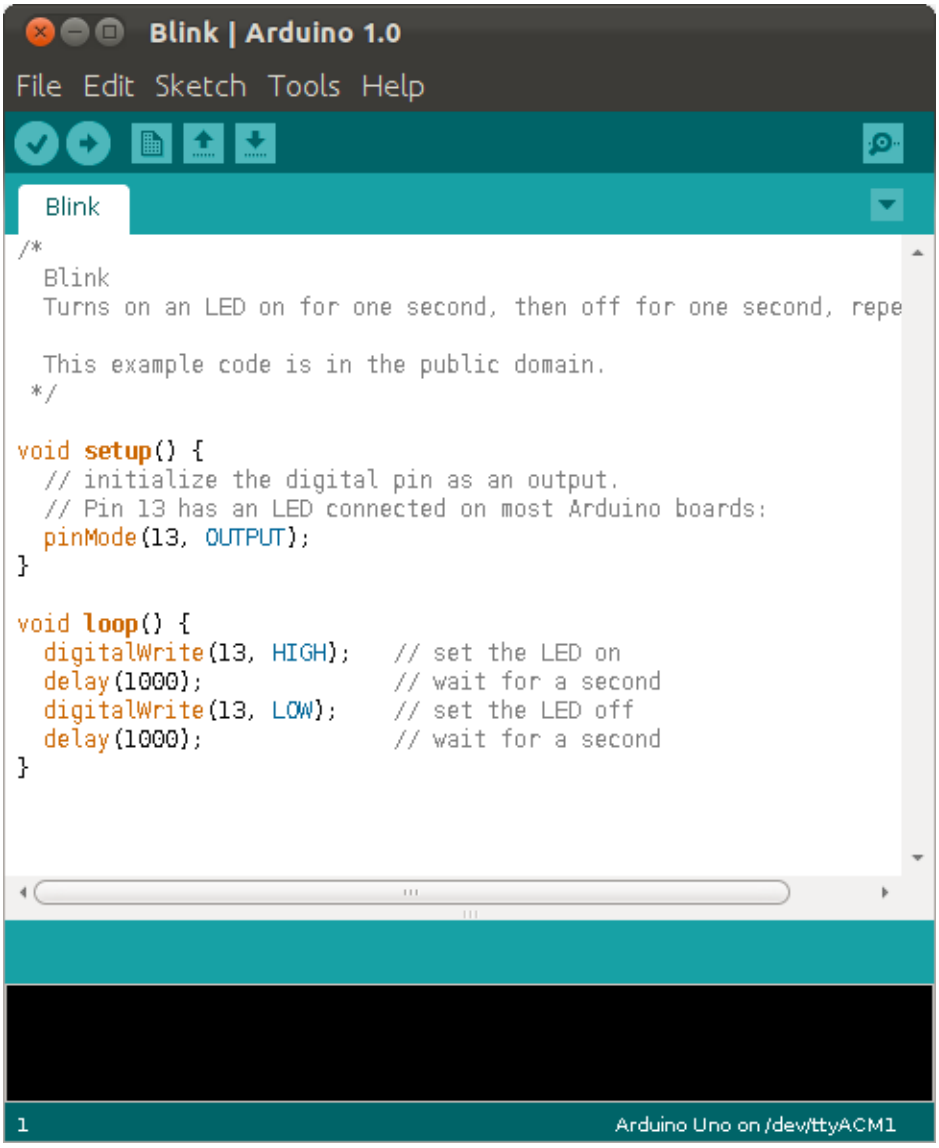
Σχήμα Β΄.4: Οι αναλογικοί ακροδέκτες του Arduino UNO.

Η αναπτυξιακή πλατφόρμα Arduino UNO μπορεί να χρησιμοποιηθεί και για τη ανάγνωση αναλογικών σημάτων. Στην κάτω πλευρά του Arduino, με τη σήμανση ANALOG IN όπως φαίνεται και στο Σχήμα Β΄.4, υπάρχει μια ακόμη σειρά από 6 pin, αριθμημένα από το 0 ως το 5. Κάθε κανάλι εισόδου μπορεί να χρησιμοποιηθεί ανεξάρτητα. Το κάθε κανάλι έχει διακριτική ικανότητα 10bit (1024 τιμές), δηλαδή διαιρείται η τάση αναφοράς σε 1024 εύρη. Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή (analogReference())στο 1.1V, ενώ η προεπιλογή είναι στα 5V. Επίσης, μπορεί να χρησιμοποιηθεί η τάση που εφαρμόζεται στο pin με τη σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτηθεί ο ακροδέκτης AREF με 3.3V και στη συνέχεια διαβάσει κάποιον ακροδέκτη αναλογικής εισόδου στο οποίο εφαρμόζεται τάση 1.65V, το Arduino θα επιστρέψει την τιμή 512 (γιατί, (1,65V/3,3V) * 1024 = 512.

Η ανάπτυξη προγραμμάτων στην οικογένεια Arduino, γίνεται μέσω του προγράμματος Arduino IDE [99].

Το περιβάλλον ανάπτυξης Arduino περιέχει ένα πρόγραμμα επεξεργασίας κειμένου, για τη σύνταξη του κώδικα, μια περιοχή στην οποία εμφανίζονται μηνύματα, μία κονσόλα κειμένου και μια γραμμή εργαλείων υπό μορφή κουμπιών. Συνδέεται με το hardware μέρος του arduino για να φορτώσει προγράμματα και να επικοινωνεί μαζί τους. Ο κώδικας που έχει γραφεί για το Arduino ονομάζεται sketch. Στον Πίνακα Β΄.4 παρουσιάζονται τα εργαλεία του περιβάλλοντος ανάπτυξης, υπό μορφή κουμπιών και στο Σχήμα Β΄.5 το ίδιο το περιβάλλον.

Το Arduino IDE είναι βασισμένο σε Java και συγκεκριμένα παρέχει:



Σχήμα Β΄.5: Το ολοκληρωμένο πρόγραμμα ανάπτυξης προγραμμάτων σε Arduino.

- Ένα πρακτικό περιβάλλον για τη συγγραφή των προγραμμάτων, με συντακτική χρωματική σήμανση.
- Μερικές έτοιμες βιβλιοθήκες για προέκταση της.
- Τον compiler για τη μεταγλώττιση των sketch.
- Μία σειριακή οθόνη (serial monitor) που παρακολουθεί τις επικοινωνίες της σειριακής (USB), αναλαμβάνει να στείλει αλφαριθμητικά στο Arduino μέσω αυτής και είναι ιδιαίτερα χρήσιμο για την αποσφαλμάτωση των sketch.
- Την επιλογή για ανέβασμα των μεταγλωττισμένων sketch στο Arduino.

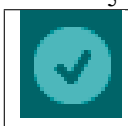
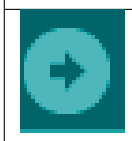
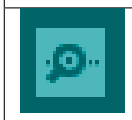
Γλώσσα Προγραμματισμού: Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring [100] μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc. Λόγω της καταγωγής της από τη C, στη γλώσσα του Arduino, μπορούν να χρησιμοποιηθούν ουσιαστικά οι ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές όπως και στη C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για τη διαχείριση του ειδικού hardware του Arduino. Τα προγράμματα του Arduino διαιρούνται σε τρία μέρη: δομή (structure), τιμές (values) και συναρτήσεις (functions).

Τα προγράμματα του Arduino διαιρούνται σε τρία μέρη: δομή (structure), τιμές (values) και συναρτήσεις (functions).

Ασφαλώς, σημαντικό στοιχείο της διαδικασίας συγγραφής προγραμμάτων στο Arduino, αποτελεί η αποσφαλμάτωση του συστήματος, είτε του λογισμικού (software) είτε του υλικού (hardware). Πολλές φορές θέλουμε να αποσφαλματώσουμε το κύκλωμά μας ή να επιβεβαιώσουμε ότι ένα κομμάτι του λειτουργεί σωστά. Για να το πετύχουμε αυτό, χρησιμοποιούμε τη σειριακή επικοινωνία σε συνδυασμό με εντολές εκτύπωσης στο σειριακό τερματικό.

1. Εμφανίζουμε τη σειριακή οθόνη, πατώντας το εικονίδιο με το μεγεθυντικό φακό στη εργαλειοθήκη με τα εικονίδια (τέρμα δεξιά).
2. Ρυθμίζουμε την ταχύτητα σε 9600 bps
3. Στο sketch του arduino στο setup() τοποθετούμε τη γραμμή Serial.begin(9600);

Πίνακας Β΄.4: Εργαλεία υπό μορφή κουμπιών στο Arduino IDE

	Verify	Ελέγχει για συντακτικά λάθη στον κώδικα.
	Upload	Μεταγλωττίζει τον κώδικα και τον φορτώνει στο Arduino.
	New	Δημιουργεί ένα νέο sketch.
	Open	Παραθέτει ένα μενού με όλα τα sketch για άνοιγμα σε νέο παράθυρο.
	Save	Αποθηκεύει ένα sketch.
	Serial Monitor	Ανοίγει την σειριακή οθόνη.

Σχήμα Β΄.6: Η τυπική δομή των προγραμμάτων Arduino.

```
<Δήλωση μεταβλητών>
void setup(){
        <Δήλωση λειτουργιών που ισχύουν για όλο το πρόγραμμα>
}
void loop(){
        <Δήλωση των λειτουργιών  που επαναλαμβάνονται κατά
        τη λειτουργία του προγράμματος>
}
```

4. Στο σημείο που θέλουμε να εμφανίσουμε κάποια τιμή, δίνουμε Serial.println(value); ή Serial.print(value) (το ln κάνει και αλλαγή γραμμής).

Με αυτόν τον τρόπο μπορούμε να αποσφαλματώσουμε βήμα-προς-βήμα όλο το κύκλωμα. Εκτυπώνουμε τις τιμές από τις εισόδους, και τις τιμές που στέλνουμε στις εξόδους και έτσι προσδιορίζουμε το πρόβλημα.

Β΄.2 Κανόνες Εργαστηρίου

Η εκπαίδευση πάνω στην ανάπτυξη εφαρμογών στο arduino γίνεται συνήθως σε κάποιο εργαστήριο. Παρακάτω παραθέτονται οι κανόνες που διέπουν το εργαστήριο του Ψηφιακών Συστημάτων και Αρχιτεκτονικής του Πανεπιστημίου Δυτικής Μακεδονίας.

1. Πριν χρησιμοποιήσετε οποιοδήποτε εξοπλισμό, θα πρέπει να ακουμπήσετε μια γειωμένη επιφάνεια (π.χ. το κουτί ενός υπολογιστή) για να απομακρυνθεί ο στατικός ηλεκτρισμός. Αυτό θα πρέπει να το επαναλαμβάνετε κατά τακτά χρονικά διαστήματα (π.χ. κάθε 20-30 λεπτά).

2. Απαγορεύεται η απομάκρυνση οποιουδήποτε εξοπλισμού από το εργαστήριο.

3. Να αναφέρετε αμέσως οποιοδήποτε πρόβλημα (χαμένο εξάρτημα, δυσλειτουργία εξαρτήματος) στους υπευθύνους.

4. Απαγορεύετε να σημειώνετε ή να τροποποιείτε μόνιμα τα εξαρτήματα.

5. Κατά την είσοδο στο εργαστήριο:

   (α΄) Ενημερώνετε τον υπεύθυνο για τον εξοπλισμό που θα χρησιμοποιήσετε.

   (β΄) Υπογράφετε την παραλαβή του εξοπλισμού και των εξαρτημάτων που το συνοδεύουν.

6. Κατά την έξοδο από το εργαστήριο:

   (α΄) θα πρέπει να παραδώσετε τακτοποιημένο τον εξοπλισμό (μέσα στις πλαστικές σακούλες/κουτιά), ακριβώς όπως σας παραδόθηκε.

   (β΄) Ο πάγκος να είναι καθαρός, η οθόνη, ο υπολογιστής, το πληκτρολόγιο και το mouse να είναι τακτοποιημένα.

   (γ΄) Ο υπεύθυνος ελέγχει τον εξοπλισμό που του παραδίδετε και υπογράφει το φύλλο παραλαβής.

7. Πάντα να έχετε κλειστή την τροφοδοσία κατά τη σύνδεση ή αποσύνδεση εξαρτημάτων από μια πλακέτα.

8. Απαγορεύονται χυμοί, νερά, καφέδες, τρόφιμα στο εργαστήριο. Μπορείτε να τα αφήνετε έξω από το εργαστήριο.

9. Μην ασκείτε υπερβολική πίεση κατά τη συναρμολόγηση ενός κυκλώματος. Μπορείτε να χρησιμοποιείτε γειτονικές επαφές/connection points αν δείτε ότι ένα εξάρτημα δεν τοποθετείται χωρίς πίεση.

10. Κάποια εξαρτήματα, όπως η μεμβράνη πίεσης ή το ποτενσιόμετρο επαφής, είναι πολύ ευαίσθητα. Για να τα τοποθετήστε στο breadboard πρέπει να τα πιάσετε από πολύ χαμηλά (δίπλα στις επαφές).

11. Οι αντιστάσεις και οι δίοδοι που έχουν λυγισμένες επαφές να τις αφήνετε σε αυτή τη μορφή και να μην τις ισιώνετε, διαφορετικά υπάρχει κίνδυνος να κοπούν.

12. Σε περίπτωση που δεν ακολουθήσετε τις υποδείξεις ασφαλείας και προστασίας τόσο του εαυτού σας όσο και του εξοπλισμού, ή αν δεν είστε προσεκτικοί θα υπάρχουν κυρώσεις.

13. Ο υπεύθυνος του εργαστηρίου έχει τον τελευταίο λόγο. Οι υποδείξεις του θα πρέπει να λαμβάνονται σοβαρά υπόψιν.

Οι εργαστηριακές ασκήσεις που παρουσιάζονται σε αυτό το Παράρτημα, βασίζονται και χρησιμοποιούν ηλεκτρονικά στοιχεία που βρίσκονται στο δημοφιλές ολοκληρωμένο αναπτυξιακό κιτ της Sparkfun (SparkFun Inventor's Kit). Ο Πίνακας Β΄.5 απαριθμεί τα εξαρτήματα που έχει το κιτ και απαιτούνται για τις ασκήσεις. Οι ενδιαφερόμενοι μπορούν είτε να προμηθευτούν το συγκεκριμένο kit είτε να προμηθευτούν ξεχωριστά από κατάλληλους προμηθευτές.

Β΄.3 Εργαστηριακή Άσκηση 1

Τα LEDs (δίοδοι εκπομπής φωτός) χρησιμοποιούνται σε πάρα πολλές εφαρμογές. Αυτός είναι και ο λόγος που το πρώτο εργαστήριο χρησιμοποιεί LEDs και έχουν συμπεριληφθεί στο SparkFun Inventor's Kit. Σκοπός της άσκησης είναι να κάνουμε ένα LED να ενεργοποιείται και να απενεργοποιείται κατ' επανάληψη.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 15 λεπτά

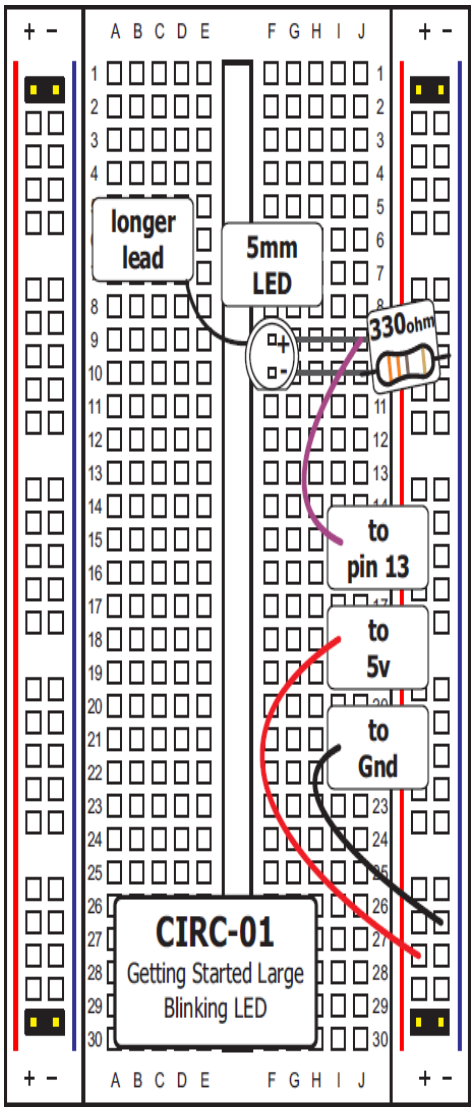
Για την εκπόνηση του κυκλώματος CIRC-01 απαιτούνται τα εξής μέρη:

- 1 κίτρινο LED 5mm
- 1 αντίσταση 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 3 καλώδια

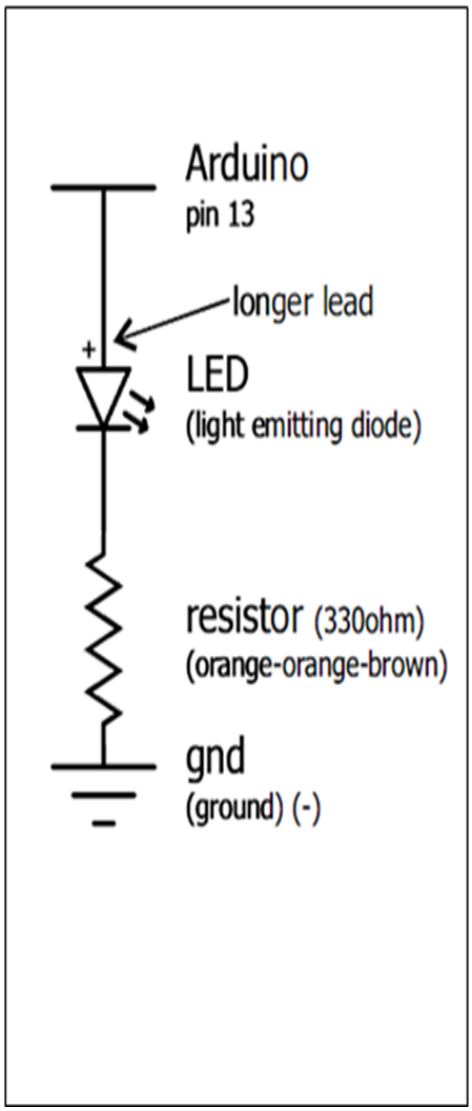
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.8. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.7. Συνδέστε το θετικό ακροδέκτη του LED στο 13ο pin του Arduino και μια αντίσταση στον αρνητικό ακροδέκτη, που να πηγαίνει στη γείωση. Επισημαίνεται ότι ο θετικός ακροδέκτης του LED είναι αυτός με το μεγαλύτερο μήκος. Τέλος συνδέστε, στις κατάλληλες θέσεις του breadboard, ένα καλώδιο στο pin τροφοδοσίας 5V του Arduino και ένα καλώδιο σε ένα από τα τρία pin γείωσης (Gnd) που βρίσκονται πάνω στο Arduino.

Β΄.3.1 Επαλήθευση παραμέτρων στο προγραμματιστικό περιβάλλον του Arduino

Συνδέστε το Arduino Uno και ανοίξτε το προγραμματιστικό περιβάλλον του. Επιβεβαιώστε ότι η επιλεγμένη σειριακή θύρα είναι η θύρα στην οποία είναι συνδεδεμένο το Arduino. Πηγαίνετε στις Ιδιότητες Υπολογιστή>Διαχείριση Συσκευών>Θύρες(COM & LPT) και συγκρίνετε την αναγραφόμενη θύρα με αυτή που είναι επιλεγμένη στο μενού Εργαλεία > Σειριακή θύρα, του προγραμματιστικού περιβάλλοντος του Arduino. Βεβαιωθείτε ότι η επιλεγμένη πλακέτα είναι το Arduino Uno Πηγαίνετε στο μενού Εργαλεία > Πλακέτα και επιλέξτε το

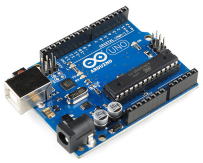
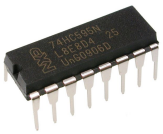


Σχήμα Β΄.7: Προτεινόμενη υλοποίηση του CIRC01

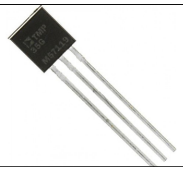
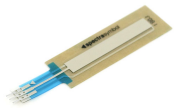


Σχήμα Β΄.8: Σχηματικό Διάγραμμα του CIRC01

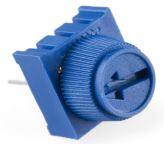
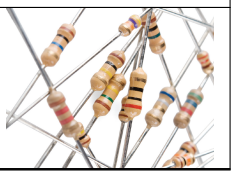
Πίνακας Β΄.5: Εξαρτήματα που απαιτούνται στις εργαστηριακές ασκήσεις

Ποσότητα	Είδος (Περιγραφή)	Εικόνα
1	Arduino Uno R3	
1	Breadboard	
1	Arduino and Breadboard Holder	
1	74HC595 Shift Register (16 pins)	
2	2N2222 transistors	
2	Diode Small Signal	
1	DC Motor with wires	
		Συνεχίζεται...

Πίνακας Β΄.5: Συνεχίζεται

Ποσότητα	Είδος (Περιγραφή)	Εικόνα
1	Small Servo	
1	5V Relay SPDT Sealed	
1	TMP36 - Temperature Sensor	
1	SoftPot MembranePotentiometer - 50mm	
30	Jumper Wires Standard 7” M/M Pack of 30	
1	Mini Photocell	
1	LED - RGB 3 color	
		Συνεχίζεται...

Πίνακας Β΄.5: Συνεχίζεται

Ποσότητα	Είδος (Περιγραφή)	Εικόνα
1	LED 5mm	
1	Trimpot 10K	
1	Buzzer 12mm 2.048kHz	
2	Momentary Push Button 12mm	
50	Resistors 330 Ohm & 10 KOhm 1/6 Watt	

Arduino Uno. Βεβαιωθείτε ότι ο επιλεγμένος προγραμματιστής είναι ο AVRISP mkII. Πηγαίνετε στο μενού Εργαλεία > Προγραμματιστής και επιλέξτε τον AVRISP mkII.

Β΄.3.2 Προγραμματισμός του κυκλώματος CIRC-01

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. (Εναλλακτικά κατεβάστε τον από (http://ardx.org/CODE01).

```
1 /*
2  * Blink
3  * Turns on an LED on for one second,
4  * then off for one second, repeatedly
```

```
5   * The circuit:
6   * LED connected from digital pin 13 to ground
7   * Note: On most Arduino boards, there is already
8   * an LED on the board
9   * connected to pin 13, so you don't need any extra
10  * components for this example
11  *Created 1 June 2005
12  *By David Cuartielles
13  *http://arduino.cc/en/Tutorial/Blink
14  *based on an orginal by H. Barragan
15  * for the Wiring i/o board
16 */
17 int ledPin = 13; // LED connected to digital pin 13
18 // The setup() method runs once, when it starts
19 void setup(){
20  // initialize the digital pin as an output:
21  pinMode(ledPin, OUTPUT);
22 }
23 // the loop() method runs over and over again,
24 // as long as the Arduino has power
25 void loop()
26 {
27  digitalWrite(ledPin, HIGH); // set the LED on
28  delay(1000); // wait for a second
29  digitalWrite(ledPin, LOW); // set the LED off
30  delay(1000); // wait for a second
31 }
```

Πατήστε Αποθήκευση (Ctrl+s). Θα σας ζητηθεί να δώστε όνομα φακέλου (Sketch Folder). Δώστε 'CIRC-01' και πατήστε 'Αποθήκευση'. Το πρόγραμμα θα δημιουργήσει τον φάκελο CIRC-01 μέσα στον οποίο θα περιέχεται το αρχείο CIRC-01.ino με τον κώδικα που γράψατε παραπάνω. Στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα πατώντας Φόρτωση (Ctrl+u). Μόλις ολοκληρωθεί η διαδικασία φόρτωσης το πρόγραμμα θα εκτελεστεί στην πλακέτα, κάνοντας το LED του κυκλώματος και της πλακέτας να αναβοσβήνει. Σε περίπτωση που δε συμβαίνει τίποτα στο κύκλωμα, ελέγξτε την πολικότητα του LED, καθώς και την ορθότητα των παραμέτρων που περιγράφηκαν στην προηγούμενη ενότητα.

Β΄.3.3 Παραμετροποίηση του κυκλώματος CIRC-01

Αλλαγή του pin σύνδεσης του LED: Το LED είναι συνδεδεμένο στο pin 13, αλλά μπορούμε να το συνδέσουμε σε οποιοδήποτε από τα pins του Arduino. Για αλλάξετε το pin αφαιρέστε το καλώδιο από το pin 13 και συνδέστε το σε ένα άλλο pin της επιλογής σας (από 0 ως 13 για τα ψηφιακά ή A0 ως A5 για τα αναλογικά pins). Ταυτόχρονα αλλάξτε τη γραμμή 15 του κώδικα:

int ledPin = 13; // LED connected to digital pin 13

Δίνοντας τον αλλαγμένο αριθμό pin (από 0 ως 13 για τα ψηφιακά ή 14 ως 19 για τα αναλογικά pins). Πατήστε Αποθήκευση ως (Ctrl+Shift+s), και μεταβείτε έξω από το φάκελο του project CIRC_01 αν είστε μέσα σ αυτόν (θα πρέπει να βλέπετε το φάκελο CIRC_01). Δώστε όνομα φακέλου CIRC_01_d1 για να απο-

θηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος.

Αλλαγή της χρονικής διάρκειας on/off του LED: Το LED είναι ρυθμισμένο να αναβοσβήνει ανά 1sec (1000ms), κάτι το οποίο μπορούμε να αλλάξουμε. Αλλάξτε τις τιμές '1000' στο παρακάτω τμήμα του κώδικα:

```
digitalWrite(ledPin, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(ledPin, LOW); // set the LED off
delay(1000); // wait for a second}
```

Πατήστε Αποθήκευση ως (Ctrl+Shift+s), μεταβείτε έξω από το φάκελο του ήδη αποθηκευμένου project, αν είστε μέσα σ αυτόν, (όμοια με πριν θα πρέπει να βλέπετε τους φακέλους CIRC_01, CIRC_01_d1) και δώστε όνομα φακέλου CIRC_01_d2 για να αποθηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε ότι ο χρόνος on/off έχει μεταβληθεί.

Αλλαγή της φωτεινότητας του LED: Εκτός από τον ψηφιακό (on/off) έλεγχο, το Arduino μπορεί να παρέχει έλεγχο που μοιάζει με τον αναλογικό (ενδιάμεσες καταστάσεις) σε ορισμένα pins. Αλλάξτε την τιμή του pin σε 9.

int ledPin = 9;

Προσαρμόστε κατάλληλα και το καλώδιο στο κύκλωμα. Στη συνέχεια αντικαταστήστε τον κώδικα μέσα στο loop() με τον παρακάτω κώδικα:

analogWrite(ledPin, 20);

Όπου 20 μπορείτε να δώσετε οποιαδήποτε τιμή από 0 ως 255. Πατήστε Αποθήκευση ως (Ctrl+Shift+s), μεταβείτε έξω από το φάκελο του ήδη αποθηκευμένου project (CIRC-01-d2) και δώστε όνομα φακέλου CIRC_01_d3 για να αποθηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος.

Ομαλή μετάβαση μεταξύ των καταστάσεων on/off του LED (Fading): Προσαρμόζοντας τον κώδικα του προηγούμενου ερωτήματος σε δύο διαδοχικές επαναλήψεις από 0 ως 255 η πρώτη, και από 254 ως 1 η δεύτερη, μπορούμε να δώσουμε όλες τις ενδιάμεσες τιμές φωτεινότητας στο LED, δημιουργώντας έτσι, ένα αποτέλεσμα ομαλής μετάβασης μεταξύ των καταστάσεων on/off. Αντικαταστήστε τον κώδικα μέσα στο loop() με τον παρακάτω κώδικα:

```
for(int  i=0; i<=255; i++)
{
 analogWrite(ledPin, i);
 delay(10);
}

for(int  i=254; i>=1; i--)
{
 analogWrite(ledPin, i);
 delay(10);
}
```

Πατήστε Αποθήκευση ως (Ctrl+Shift+s), μεταβείτε έξω από το φάκελο του ήδη αποθηκευμένου project (CIRC-01-d3) και δώστε όνομα φακέλου CIRC_01_d4 για να αποθηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος. Δείτε επίσης τον κώδικα στο Αρχείο>Παραδείγματα>01.Basics>Fade που περιέχει μια διαφορετική υλοποίηση της ίδιας λειτουργίας.

Β΄.4 Εργαστηριακή Άσκηση 2

Στην προηγούμενη εργαστηριακή άσκηση υλοποιήθηκε ένα κύκλωμα που ενεργοποιούσε και απενεργοποιούσε ένα LED. Σ' αυτή την εργαστηριακή άσκηση θα υλοποιηθεί ένα κύκλωμα που εφαρμόζει την ίδια λειτουργία σε οκτώ LEDs. Θα υλοποιηθούν διάφορα LED animations και θα κατανοηθούν καλύτερα οι βασικές αρχές λειτουργίας και προγραμματισμού του Arduino, αφού θα χρησιμοποιηθούν πίνακες για ευκολότερη διαχείριση των μεταβλητών και η for() για επαναλήψεις.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 25 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

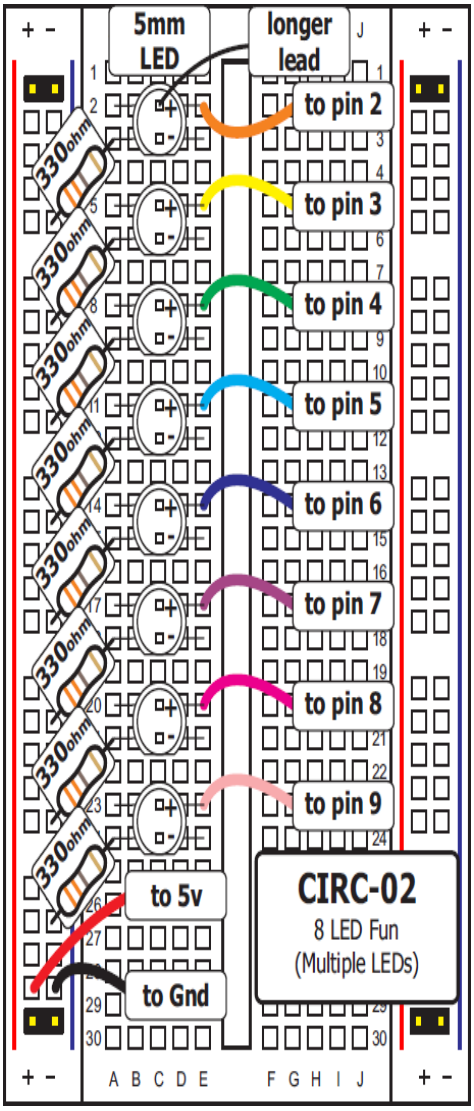
- 8 κίτρινο LED των 5mm
- 8 αντιστάσεις των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 10 καλώδια

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.10. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.9.

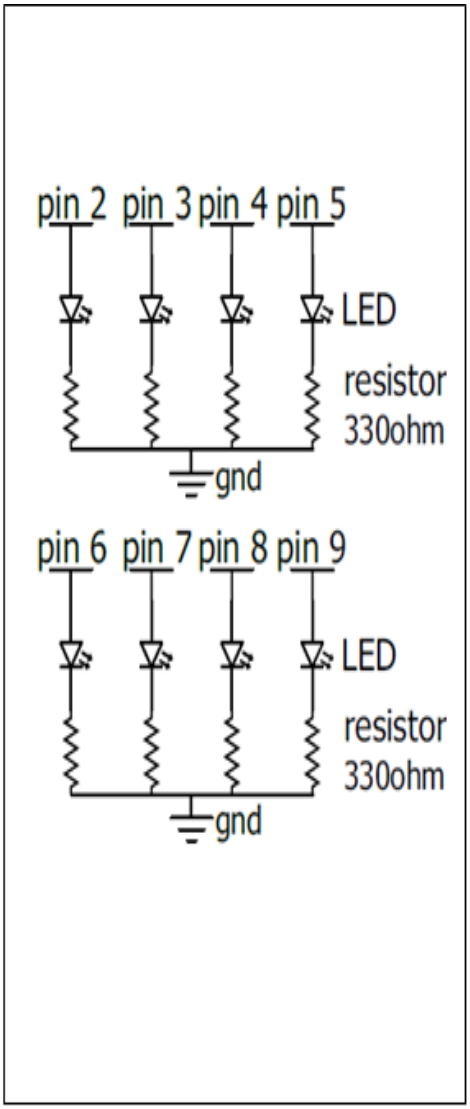
Συνδέστε το θετικό ακροδέκτη του LED στο στο αναγραφόμενο pin του Arduino και μια αντίσταση 330 Ohm στον κάθε αρνητικό ακροδέκτη, που να πηγαίνει στη γείωση. Επισημαίνεται ότι ο θετικός ακροδέκτης του LED είναι αυτός με το μεγαλύτερο μήκος. Τέλος συνδέστε, στις κατάλληλες θέσεις του breadboard, ένα καλώδιο στο pin τροφοδοσίας 5V του Arduino και ένα σε κάποιο pin γείωσης (Gnd).

Β΄.4.1 Επαλήθευση παραμέτρων στο προγραμματιστικό περιβάλλον του Arduino

Συνδέστε το Arduino Uno και ανοίξτε το προγραμματιστικό περιβάλλον του. Επιβεβαιώστε ότι η επιλεγμένη σειριακή θύρα είναι η θύρα στην οποία είναι συνδεδεμένο το Arduino. Βεβαιωθείτε ότι η επιλεγμένη πλακέτα είναι το Arduino Uno. Βεβαιωθείτε ότι ο επιλεγμένος προγραμματιστής είναι ο AVRISP



Σχήμα Β΄.9: Προτεινόμενη υλοποίηση του CIRC02



Σχήμα Β΄.10: Σχηματικό Διάγραμμα του CIRC02

mkII. Όλα τα παραπάνω έχουν περιγραφεί αναλυτικά στην Ενότητα Β΄.3.1 του CIRC-01.

Β΄.5 Προγραμματισμός του κυκλώματος CIRC-02

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. (Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE02)

```
/*        ------------------------------------------------------
 * | Arduino Experimentation Kit Example Code|
 * | CIRC-02 .: 8 LED Fun :. (Multiple LEDs)|
 *        ------------------------------------------------------
 *
 * A few Simple LED animations
 *
 * For more information on this circuit http://tinyurl.com/d2hrud
 *
 */
 //LED Pin Variables
int ledPins[] = {2,3,4,5,6,7,8,9};
    //An array to hold the pin each LED is connected to
    //i.e. LED #0 is connected to pin 2, LED #1, 3 and so on
    //to address an array use ledPins[0] this would equal 2
    //and ledPins[7] would equal 9

/*
 * setup() - this function runs once when you turn your Arduino on
 * We the three control pins to outputs
 */
void setup()
{
  //Set each pin connected to an LED to output mode
  //(pulling high (on) or low (off)
  for(int i = 0; i < 8; i++){ //this is a loop and will repeat eight times
   pinMode(ledPins[i],OUTPUT); //we use this to set each LED pin to output
  } //the code this replaces is below
 /* (commented code will not run)
  * these are the lines replaced by the for loop above they do exactly the
  * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
  pinMode(ledPins[4],OUTPUT);
  pinMode(ledPins[5],OUTPUT);
  pinMode(ledPins[6],OUTPUT);
  pinMode(ledPins[7],OUTPUT);
  (end of commented code)*/
}
 /*
 * loop() - this function will start after setup finishes and then repeat
 * we call a function called oneAfterAnother().
 * if you would like a different behaviour
 * uncomment (delete the two slashes) one of the other lines
 */
void loop() // run over and over again
{
    //this will turn on each LED one by one then turn each off
```

```
51      oneAfterAnotherNoLoop();
52
53      //does the same as oneAfterAnotherNoLoop but with less typing
54      //oneAfterAnotherLoop();
55
56      //oneOnAtATime();
57      //this will turn one LED on then turn the next one
58      //on turning the
59      //former off (one LED will look like it is scrolling
60      //along the lin//inAndOut();
61      //lights the two middle LEDs then moves them out then back
62      //in again
63  }
64  /*
65   * oneAfterAnotherNoLoop() - Will light one LED then delay for delayTime
66   * then light the next LED until all LEDs are on it will then turn
67   * them off one after another
68   *
69   * this does it without using a loop which makes for a lot of typing.
70   * oneOnAtATimeLoop() does exactly the same thing with less typing
71   */
72  void oneAfterAnotherNoLoop(){
73    int delayTime = 100; //the time (in milliseconds) to pause between LEDs
74      //make smaller for quicker switching and larger for slower
75    digitalWrite(ledPins[0], HIGH); //Turns on LED #0 (connected to pin 2 )
76    delay(delayTime); //waits delayTime milliseconds
77    digitalWrite(ledPins[1], HIGH); //Turns on LED #1 (connected to pin 3 )
78    delay(delayTime); //waits delayTime milliseconds
79    digitalWrite(ledPins[2], HIGH); //Turns on LED #2 (connected to pin 4 )
80    delay(delayTime); //waits delayTime milliseconds
81    digitalWrite(ledPins[3], HIGH); //Turns on LED #3 (connected to pin 5 )
82    delay(delayTime); //waits delayTime milliseconds
83    digitalWrite(ledPins[4], HIGH); //Turns on LED #4 (connected to pin 6 )
84    delay(delayTime); //waits delayTime milliseconds
85    digitalWrite(ledPins[5], HIGH); //Turns on LED #5 (connected to pin 7 )
86    delay(delayTime); //waits delayTime milliseconds
87    digitalWrite(ledPins[6], HIGH); //Turns on LED #6 (connected to pin 8 )
88    delay(delayTime); //waits delayTime milliseconds
89    digitalWrite(ledPins[7], HIGH); //Turns on LED #7 (connected to pin 9 )
90    delay(delayTime); //waits delayTime milliseconds
91     //Turns Each LED Off
92    digitalWrite(ledPins[7], LOW); //Turns on LED #0 (connected to pin 2 )
93    delay(delayTime); //waits delayTime milliseconds
94    digitalWrite(ledPins[6], LOW); //Turns on LED #1 (connected to pin 3 )
95    delay(delayTime); //waits delayTime milliseconds
96    digitalWrite(ledPins[5], LOW); //Turns on LED #2 (connected to pin 4 )
97    delay(delayTime); //waits delayTime milliseconds
98    digitalWrite(ledPins[4], LOW); //Turns on LED #3 (connected to pin 5 )
99    delay(delayTime); //waits delayTime milliseconds
100   digitalWrite(ledPins[3], LOW); //Turns on LED #4 (connected to pin 6 )
101   delay(delayTime); //waits delayTime milliseconds
102   digitalWrite(ledPins[2], LOW); //Turns on LED #5 (connected to pin 7 )
103   delay(delayTime); //waits delayTime milliseconds
104   digitalWrite(ledPins[1], LOW); //Turns on LED #6 (connected to pin 8 )
105   delay(delayTime); //waits delayTime milliseconds
106   digitalWrite(ledPins[0], LOW); //Turns on LED #7 (connected to pin 9 )
107   delay(delayTime); //waits delayTime milliseconds
108 }
109 /*
110  * oneAfterAnotherLoop() - Will light one LED then delay for delayTime then light
111  * the next LED until all LEDs are on it will then turn them
112  * off one after another
```

```
113  * this does it using a loop which makes for a lot less typing.
114  * than oneOnAtATimeNoLoop() does exactly the same thing with less typing
115  */
116 void oneAfterAnotherLoop(){
117   int delayTime = 100; //the time (in milliseconds) to pause between LEDs
118    //make smaller for quicker switching and larger for slower
119    //Turn Each LED on one after another
120   for(int i = 0; i <= 7; i++){
121 digitalWrite(ledPins[i], HIGH); //Turns on LED #i each time this runs i
122 delay(delayTime); //gets one added to it so this will repeat
123 } //8 times the first time i will = 0 the final
124   //time i will equal 7;
125
126 //Turn Each LED off one after another
127   for(int i = 7; i >= 0; i--){
128    //same as above but rather than starting at 0 and counting up
129    //we start at seven and count down
130    digitalWrite(ledPins[i], LOW); //Turns off LED #i each time this runs i
131    delay(delayTime); //gets one subtracted from it so this will repeat
132 } //8 times the first time i will = 7 the final
133   //time it will equal 0
134 }
135
136 /*
137  * oneOnAtATime() - Will light one LED then the next turning off all the others
138  */
139 void oneOnAtATime(){
140   int delayTime = 100; //the time (in milliseconds) to pause between LEDs
141    //make smaller for quicker switching and larger for slower
142   for(int i = 0; i <= 7; i++){
143   int offLED = i - 1; //Calculate which LED was turned on last time through
144   if(i == 0) { //for i = 1 to 7 this is i minus 1 (i.e. if i = 2 we will
145   offLED = 7; //turn on LED 2 and off LED 1)
146   } //however if i = 0 we don't want to turn of led -1 (doesn't exist)
147    //instead we turn off LED 7, (looping around)
148   digitalWrite(ledPins[i], HIGH); //turn on LED #i
149   digitalWrite(ledPins[offLED], LOW); //turn off the LED we turned on last time
150   delay(delayTime);
151  }
152 }
153
154 /*
155  * inAndOut() - This will turn on the two middle LEDs then the next two out
156  * making an in and out look
157  */
158 void inAndOut(){
159   int delayTime = 100; //the time (in milliseconds) to pause between LEDs
160    //make smaller for quicker switching and larger for slower
161
162 //runs the LEDs out from the middle
163  for(int i = 0; i <= 3; i++){
164   int offLED = i - 1; //Calculate which LED was turned on last time through
165   if(i == 0) { //for i = 1 to 7 this is i minus 1 (i.e. if i = 2 we will
166   offLED = 3; //turn on LED 2 and off LED 1)
167   } //however if i = 0 we don't want to turn of led -1 (doesn't exist)
168    //instead we turn off LED 7, (looping around)
169   int onLED1 = 3 - i; //this is the first LED to go on
170                       //ie. LED #3 when i = 0 and LED #0 when i = 3
171   int onLED2 = 4 + i; //this is the first LED to go on
172                       //ie. LED #4 when i = 0 and LED #7 when i = 3
173   int offLED1 = 3 - offLED; //turns off the LED we turned on last time
174   int offLED2 = 4 + offLED; //turns off the LED we turned on last time
```

```
175
176   digitalWrite(ledPins[onLED1], HIGH);
177   digitalWrite(ledPins[onLED2], HIGH);
178   digitalWrite(ledPins[offLED1], LOW);
179   digitalWrite(ledPins[offLED2], LOW);
180   delay(delayTime);
181  }
182
183  //runs the LEDs into the middle
184   for(int  i  = 3; i  >= 0; i--){
185     int offLED = i + 1; //Calculate which LED was turned on last time through
186     if(i  == 3) { //for i = 1 to 7 this is i minus 1 (i.e.  if  i  = 2 we will
187     offLED = 0; //turn on LED 2 and off LED 1)
188     } //however if i = 0 we don't want to turn of led -1 (doesn't exist)
189  //instead we turn off LED 7, (looping around)
190  int onLED1 = 3 –I;  //this is the first  LED to go on
191         //ie. LED #3 when i = 0 and LED #0 when i = 3
192  int onLED2 = 4 + i; //this is the first LED to go on
193         //ie. LED #4 when i = 0 and LED #7 when i = 3
194  int offLED1 = 3 - offLED; //turns off the LED we turned on last time
195  int offLED2 = 4 + offLED; //turns off the LED we turned on last time
196
197  digitalWrite(ledPins[onLED1], HIGH);
198  digitalWrite(ledPins[onLED2], HIGH);
199  digitalWrite(ledPins[offLED1], LOW);
200  digitalWrite(ledPins[offLED2], LOW);
201  delay(delayTime);
202   }
203  }
```

Πατήστε Αποθήκευση (Ctrl+s). Θα σας ζητηθεί να δώστε όνομα φακέλου (Sketch Folder). Δώστε 'CIRC-02' και πατήστε 'Αποθήκευση'. Το πρόγραμμα θα δημιουργήσει τον φάκελο CIRC-02 μέσα στον οποίο θα περιέχεται το αρχείο CIRC-02.ino με τον κώδικα που γράψατε παραπάνω. Στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα πατώντας Φόρτωση (Ctrl+u). Μόλις ολοκληρωθεί η διαδικασία φόρτωσης το πρόγραμμα θα εκτελεστεί στην πλακέτα, κάνοντας το LED του κυκλώματος και της πλακέτας να αναβοσβήνει. Σε περίπτωση που κάποια LEDs δεν ανάβουν, ελέγξτε την πολικότητα τους. Αν κάποια LEDs ανάβουν εκτός της ακολουθίας που περιγράφεται στη συνάρτηση oneAfterAnotherNoLoop() ελέγξτε μήπως τα καλώδια τους έχουν τοποθετηθεί στη λάθος θύρα.

Β΄.5.1 Παραμετροποίηση του κυκλώματος CIRC-02

Αλλαγή συνάρτησης ελέγχου των LEDs: Στη συνάρτηση loop() υπάρχουν τέσσερεις συναρτήσεις και οι τελευταίες τρεις ξεκινούν με '//'. Αυτό σημαίνει ότι η γραμμή αντιμετωπίζεται ως σχόλιο και δεν εκτελείται. Για να αλλάξετε τη συνάρτηση ελέγχου των LEDs από την oneAfterAnotherNoLoop(), σε μία από τις άλλες τρεις, βάλτε σχόλιο στην oneAfterAnotherNoLoop(), και σβήστε τα από μια άλλη, όπως παρακάτω. Πατήστε Αποθήκευση ως (Ctrl+Shift+s), και μεταβείτε έξω από το φάκελο του project CIRC_02 αν είστε μέσα σ' αυ-

τόν. Δώστε όνομα φακέλου CIRC_02_d1 για να αποθηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος. (Σημείωση: Οι oneAfterAnotherNoLoop() και oneAfterAnotherLoop() έχουν την ίδια λειτουργία, απλά διαφέρουν ως προς την υλοποίηση)

Δημιουργία μιας δικιάς μας συνάρτησης για τον έλεγχο των LEDs: Στο τέλος του προγράμματος, μετά τη συνάρτηση inAndOut(), ορίστε μια συνάρτηση με όνομα my_animation() η οποία θα υλοποιεί ένα animation που θα γράψετε εσείς. Ενδεικτικά η δομή μιας συνάρτησης είναι:

```
1 void function_name()
2 {
3   variable_decleration;
4   commands;
5 }
```

Μόλις ολοκληρώσετε τη συγγραφή της συνάρτησης, πηγαίνετε στη συνάρτηση loop() και καλέστε την γράφοντας my_animation();. Το animation που πρέπει να υλοποιήσετε είναι το animation που υπάρχει στο video http://www.youtube.com/watch?v=WxE2xWZNfOc. Θα πρέπει να χρησιμοποιήσετε τουλάχιστον ένα for-loop. Μην παραλείψετε να κάνετε σχόλια όλες τις άλλες κλήσεις συναρτήσεων που περιέχονται στην loop(). Πατήστε Αποθήκευση ως (Ctrl+Shift+s), μεταβείτε έξω από το φάκελο του ήδη αποθηκευμένου project, αν είστε μέσα σ αυτόν, (όμοια με πριν θα πρέπει να βλέπετε τους φακέλους CIRC_02, CIRC_02_d1) και δώστε όνομα φακέλου CIRC_02_d2 για να αποθηκεύσετε το πρόγραμμα. Στη συνέχεια, φορτώστε το στην πλακέτα (Ctrl+u). Επιβεβαιώστε ότι η συνάρτηση σας δουλεύει όπως θα έπρεπε.

Β΄.6 3η εργαστηριακή άσκηση

Τα pins του Arduino είναι κατάλληλα για να ελέγχουν μικρά ηλεκτρονικά στοιχεία όπως τα LEDs, όμως όταν έχουμε να κάνουμε με μεγαλύτερα στοιχεία, όπως ένα μοτέρ, απαιτείται η χρήση εξωτερικού τρανζίστορ. Τα τρανζίστορ είναι πολύ χρήσιμα, γιατί μας επιτρέπουν να ρυθμίζουμε μεγάλης έντασης ρεύματα, χρησιμοποιώντας ρεύματα μικρότερης έντασης. Το τρανζίστορ έχει τρία pins. Για ένα NPN τρανζίστορ όπως αυτό που θα χρησιμοποιήσουμε σ' αυτό το εργαστήριο, συνδέετε το φορτίο στο συλλέκτη, και τον εκπομπό στη γείωση. Όταν ένα μικρής έντασης ρεύμα περνάει από τη βάση στον εκπομπό, θα ελεγχθεί ένα μεγαλύτερης έντασης ρεύμα και θα κινηθεί το μοτέρ. Το τρανζίστορ που θα χρησιμοποιηθεί είναι ένα P2N2222AG, το οποίο είναι ένα σύνηθες τρανζίστορ γενικής χρήσης. Τα σημαντικά στοιχεία του, είναι η μέγιστη τάση (40 V) και το μέγιστο ρεύμα (200 mA) τα οποία είναι αρκετά υψηλά για να θέσουν σε

λειτουργία το μοτέρ μας. Όσον αφορά τη δίοδο 1N4001 μπορείτε να ανατρέξετε στο σχετικό λήμμα της Wikipedia για να δείτε για ποιο λόγο χρησιμοποιείται.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 30 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

- 1 Τρανζίστορ P2N2222AG (TO92)
- 1 Μοτέρ DC
- 1x Αντίσταση 10 kOhm (Καφέ–Μαύρο–Πορτοκαλί)
- 8 καλώδια

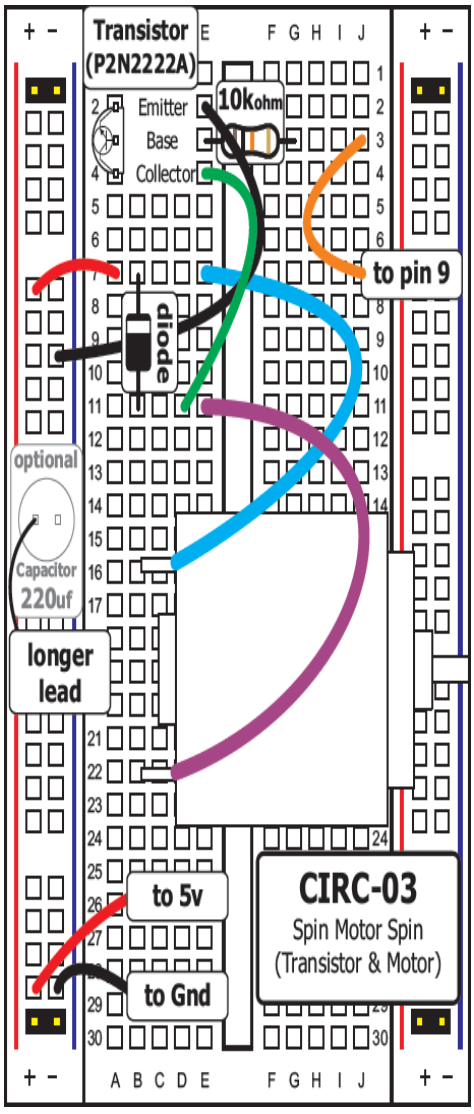
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.12. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.11.

Συνδέστε τη βάση του τρανζίστορ με μία αντίσταση 10kOhm και στη συνέχεια συνδέστε την άλλη άκρη της αντίστασης με το pin 9 του Arduino. Ο εκπομπός του τρανζίστορ συνδέεται στη γείωση, ενώ ο συλλέκτης ακριβώς πριν τη δίοδο, όπως φαίνεται στο Σχήμα Β΄.12. Τέλος στη δίοδο συνδέεται και το μοτέρ (το κόκκινο καλώδιο στην πλευρά που συνδέεται με την τάση +5 V, ενώ το μαύρο στην πλευρά που έχουμε συνδέσει τον συλλέκτη του τρανζίστορ). Ο πυκνωτής που φαίνεται στο Σχήμα Β΄.11 δεν είναι απαραίτητος για τη λειτουργία του κυκλώματος και μπορεί να παραληφθεί. Η τοποθέτηση του χρειάζεται μόνο στην περίπτωση που παρατηρήσετε ότι το Arduino κάνει επανεκκίνηση μόνο του. Τέλος συνδέστε, στις κατάλληλες θέσεις του breadboard, ένα καλώδιο στο pin τροφοδοσίας 5V του Arduino και ένα καλώδιο σε ένα από τα τρία pin γείωσης (Gnd) που βρίσκονται πάνω στο Arduino.

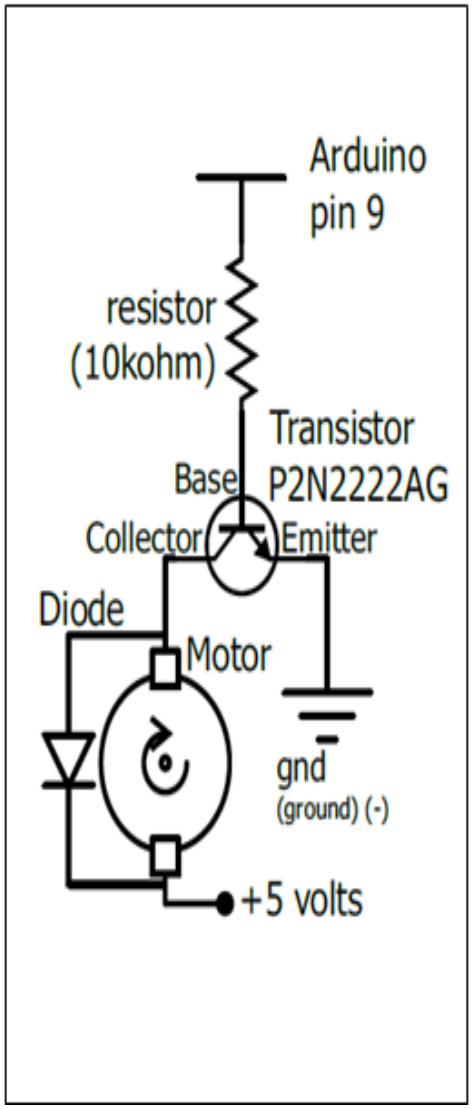
Β΄.7 Προγραμματισμός του κυκλώματος CIRC-03

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE03.

```
1  /*       ------------------------------------------------------------
2   *|  Arduino Experimentation Kit Example Code|
3   *|  CIRC-03 .: Spin Motor Spin :. (Transistor and Motor) |
4          *------------------------------------------------------------
5   *
6   * The Arduinos pins are great for driving LEDs however if you hook
7   * up something that requires more power you will quickly break them.
8   * To control bigger items we need the help of a transistor.
9   * Here we will use a transistor to control a small toy motor
10  *
11  * http://tinyurl.com/d4wht7
12  *
13  */
14
```



Σχήμα Β΄.11: Προτεινόμενη υλοποίηση του CIRC03



Σχήμα Β΄.12: Σχηματικό Διάγραμμα του CIRC03

```
int motorPin = 9; // define the pin the motor is connected to
// (if you use pin 9,10,11 or 3you can also control speed)

/*
 * setup() - this function runs once when you turn your Arduino on
 * We set the motors pin to be an output (turning the pin high (+5v)
 * or low (ground) (-))
 * rather than an input (checking whether a pin is high or low)
 */
void setup()
{
 pinMode(motorPin, OUTPUT);
}


/*
 * loop() - this function will start after setup finishes and then repeat
 * we call a function called motorOnThenOff()
 */

void loop() // run over and over again
{
 motorOnThenOff();
 //motorOnThenOffWithSpeed();
 //motorAcceleration();
}

/*
 * motorOnThenOff() - turns motor on then off
 * (notice this code is identical to the code we used for
 * the blinking LED)
 */
void motorOnThenOff(){
   int onTime = 2500; //the number of milliseconds for the motor to turn on for
   int offTime = 1000; //the number of milliseconds for the motor to turn off for
   digitalWrite(motorPin, HIGH); // turns the motor On
   delay(onTime); // waits for onTime milliseconds
   digitalWrite(motorPin, LOW); // turns the motor Off
   delay(offTime); // waits for offTime milliseconds
}

/*
 * motorOnThenOffWithSpeed() - turns motor on then off but uses speed values
 * (notice this code is identical to the code we used for
 * the blinking LED)
 */
void motorOnThenOffWithSpeed(){
   int onSpeed = 200; // a number between 0 (stopped) and 255 (full speed
   int onTime = 2500; //the number of milliseconds for the motor to turn on for
   int offSpeed = 50; // a number between 0 (stopped) and 255 (full speed)
   int offTime = 1000; //the number of milliseconds for the motor to turn off for
   analogWrite(motorPin, onSpeed); // turns the motor On
   delay(onTime); // waits for onTime milliseconds
   analogWrite(motorPin, offSpeed); // turns the motor Off
   delay(offTime); // waits for offTime milliseconds
}

/*
 * motorAcceleration() - accelerates the motor to full speed then
 * back down to zero
*/
void motorAcceleration(){
```

```
77    int delayTime = 50; //milliseconds between each speed step
78    //Accelerates the motor
79    for(int i = 0; i < 256; i++){ //goes through each speed from 0 to 255
80     analogWrite(motorPin, i); //sets the new speed
81     delay(delayTime); // waits for delayTime milliseconds
82    }
83
84  //Decelerates the motor
85    for(int i = 255; i >= 0; i--){ //goes through each speed from 255 to 0
86     analogWrite(motorPin, i); //sets the new speed
87     delay(delayTime); // waits for delayTime milliseconds
88    }
89  }
```

Αποθηκεύστε το ως CIRC_03 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Αν το μοτέρ δεν δουλεύει, ελέγξτε ξανά το κύκλωμα σας, καθώς και τις παραμέτρους του προγραμματιστικού περιβάλλοντος του Arduino (όπως έχουν περιγραφεί στο CIRC_01, Ενότητα Β΄.3.1). Ελέγξτε επίσης αν τα τρανζίστορ είναι τα αναγραφόμενα P2N2222AG (TO92) και αν δεν είναι, ελέγξτε αν τα pins τους έχουν την ίδια λειτουργία με αυτά του P2N2222AG (TO92), από το site του κατασκευαστή (σε ορισμένα τρανζίστορ ο εκπομπός και ο συλλέκτης είναι ανάποδα). Τέλος, αν το Arduino κάνει επανεκκίνηση μόνο του, πρέπει να τοποθετήσουμε στο κύκλωμα τον πυκνωτή που αναφέρθηκε στην αρχή της άσκησης.

Β΄.8 Παραμετροποίηση του κυκλώματος CIRC-03

Αλλαγή της συνάρτησης ελέγχου του μοτέρ (Έλεγχος της ταχύτητας): Σε προηγούμενο εργαστήριο είδαμε την ικανότητα του Arduino να ελέγχει τη φωτεινότητα ενός LED. Θα χρησιμοποιήσουμε την ίδια ικανότητα για να ελέγξουμε την ταχύτητα του μοτέρ μας. Το Arduino το πετυχαίνει αυτό χρησιμοποιώντας διαμόρφωση πλάτους παλμού (PWM), μόνο που αντί να μεταβάλλει απευθείας την τάση που προέρχεται από το pin, μεταβαίνει μεταξύ των δύο καταστάσεων πολύ γρήγορα. Για παράδειγμα, εάν το Arduino διαμορφώνει το πλάτος του παλμού στο 50επειδή τα μάτια μας δεν είναι αρκετά γρήγορα για να το δουν να ανάβει και να σβήνει. Το ίδιο χαρακτηριστικό λειτουργεί και με τρανζίστορ. Στη συνάρτηση loop() υπάρχουν τρεις συναρτήσεις, δύο εκ των οποίων είναι σχόλια. Αφού διαβάσετε και κατανοήσετε τι λειτουργία έχει η κάθε συνάρτηση, αλλάξτε τη συνάρτηση ελέγχου του μοτέρ. Στη συνέχεια αποθηκεύστε και τρέξτε το πρόγραμμα για καθεμιά από τις συναρτήσεις.

Β΄.9 4η εργαστηριακή άσκηση

Σε εφαρμογές όπου χρειάζεται μεγαλύτερος έλεγχος και ακρίβεια στην κίνηση από αυτή που μας παρέχει ένα μοτέρ, χρησιμοποιούμε Servos. Στο εσωτερικό ενός servo, υπάρχει ένα μικρό κιβώτιο ταχυτήτων (για να κάνει πιο ισχυρή την κίνηση) και μερικά ηλεκτρονικά στοιχεία (μοτέρ, ποτενσιόμετρο κτλ). Ένα τυπικό servo μπορεί να κινείται μεταξύ των 0 και 180 μοιρών. Η κίνηση ελέγχεται μέσω ενός χρονισμένου παλμού, μεταξύ 1.25 ms (0 μοίρες) και 1.75 ms (180 μοίρες). Ο χρόνος ενδέχεται να διαφέρει μεταξύ διαφορετικών κατασκευαστών. Αν ο παλμός στέλνεται μεταξύ κάθε 25 - 50 ms το servo λειτουργεί ομαλά. Το Arduino έχει μια βιβλιοθήκη η οποία δίνει τη δυνατότητα να ελέγχετε 2 servos μαζί (συνδεδεμένα στα pins 9, 10) με μόνο μια γραμμή κώδικα.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 30 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

- 1 3pin Header
- 1 Mini Servo
- 5 καλώδια

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.14. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.13.

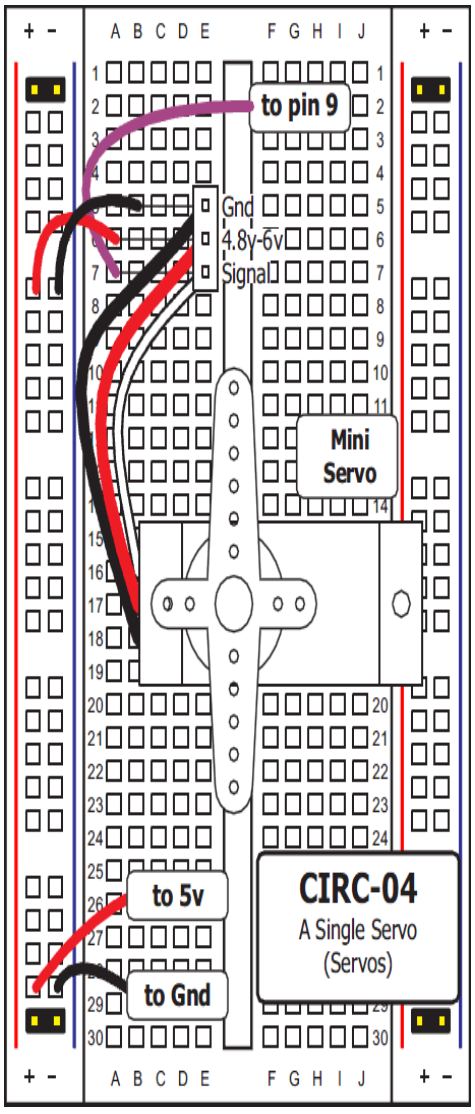
Συνδέστε το Servo με το 3pin Header και στη συνέχεια τοποθετήστε το στο breadboard όπως φαίνεται στο Σχήμα Β΄.13. Συνδέστε το άσπρο καλώδιο του Servo με το pin 9 του Arduino, το κόκκινο καλώδιο στην τάση +5 V, ενώ το μαύρο στη γείωση. Τέλος συνδέστε τα +5V και Gnd στο Arduino. Στο servo τοποθετήστε το εξάρτημα που έχει μια ακμή (‘ένα δόντι’) μόνο για να μπορείτε να παρακολουθήσετε καλύτερα την μετακίνηση.

*** **Προσοχή: Ενδέχεται το header pin να μη μπορεί να σταθεροποιηθεί πάνω στο breadboard. Σε αυτή την περίπτωση, αφού προγραμματίσετε το arduino θα ασκείτε ελαφριά πίεση με το δάχτυλο στο συνδετήρα για να λειτουργεί.** ***

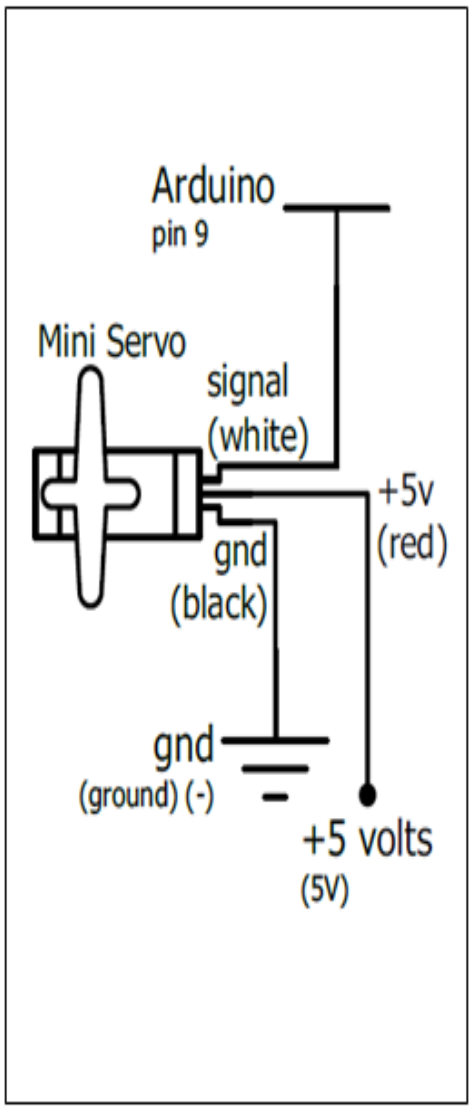
Β΄.10 Προγραμματισμός του κυκλώματος CIRC-04

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE04.

```
1 // Sweep
2 // by BARRAGAN <http://barraganstudio.com>
3
4 #include <Servo.h>
```



Σχήμα Β΄.13: Προτεινόμενη υλοποίηση του CIRC04



Σχήμα Β΄.14: Σχηματικό Διάγραμμα του CIRC04

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

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

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


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

***** Προσοχή: Το servo δέχεται εντολές τοποθέτησης από 0 έως 180 μοίρες. Μη δοκιμάσετε να στείλετε τιμή μεγαλύτερη από 180 γιατί θα υπερθερμανθεί και μπορεί να καεί. *****

Αποθηκεύστε το ως CIRC_04 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Αν το Servo δεν περιστρέφεται, ελέγξτε μήπως τυχόν το έχετε τοποθετήσει ανάποδα. Ελέγξτε επίσης αν έχετε συνδέσει σωστά την πηγή. Αν το Servo ξεκινά να περιστρέφεται αλλά καταλήγει να κάνει σπασμωδικές κινήσεις, και υπάρχει ένα λαμπάκι πάνω στην πλακέτα arduino που αναβοσβήνει, τότε η τροφοδοσία του USB δεν είναι αρκετή και θα πρέπει να χρησιμοποιηθεί ένα εξωτερικό καλώδιο τροφοδοσίας.

Β΄.11 Παραμετροποίηση του κυκλώματος CIRC-04

Μπορείτε να προγραμματίσετε το Servo χωρίς να χρησιμοποιήσετε τις έτοιμες βιβλιοθήκες του Arduino. Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino:

```
int servoPin = 9;
void setup(){
  pinMode(servoPin,OUTPUT); }

void loop() {
  int pulseTime = 2100; //(the number of microseconds
   //to pause for (1500 90 degrees
   // 900 0 degrees 2100 180 degrees)
 digitalWrite(servoPin, HIGH);
 delayMicroseconds(pulseTime);
 digitalWrite(servoPin, LOW);
```

```
12  delay(25);
13  }
```

Τροποποιήστε τον (αν δε λειτουργεί κανονικά), ώστε να περιστρέφεται για 180ο. Στη συνέχεια, αποθηκεύστε το ως CIRC_04_c1 και φορτώστε το πρόγραμμα στο Arduino.

Β΄.11.1 Ενδιαφέρουσες εφαρμογές

Μπορείτε να δείτε μερικές ενδιαφέρουσες εφαρμογές με Servos στους παρακάτω συνδέσμους:

- Xmas Hit Counter (http://tinkerlog.com/2007/12/04/arduino-xmas-hitcounter/)
- Open Source Robotic Arm (http://www.thingiverse.com/thing:387)
- Servo Walker (http://www.instructables.com/id/simpleWalker-4-legged-2-servo-walking-robot/)

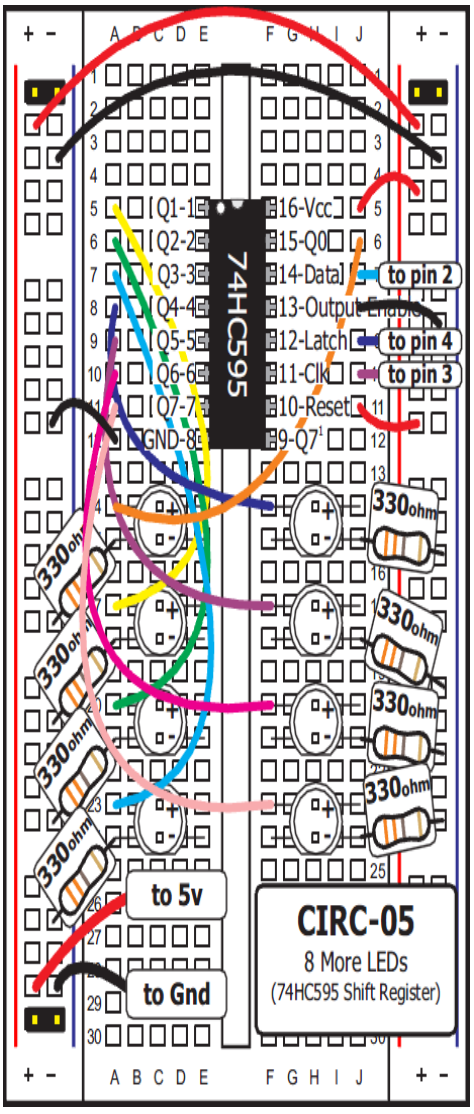
Β΄.12 5η εργαστηριακή άσκηση

Ο καταχωρητής μετατόπισης είναι ένα ολοκληρωμένο κύκλωμα, το οποίο δίνει 8 πρόσθετες εξόδους (για τον έλεγχο ηλεκτρονικών στοιχείων) χρησιμοποιώντας μόνο τρία pins του Arduino. Επίσης μπορούν να συνδεθούν πολλοί καταχωρητές μετατόπισης μαζί, ώστε να έχουμε επιπρόσθετες εξόδους χρησιμοποιώντας πάλι τα ίδια pins. Για να το χρησιμοποιήσετε θέτετε το data pin σε HIGH ή LOW, δίνετε έναν παλμό στο ρολόι, και επαναλαμβάνετε μέχρι να έχετε 8 bits δεδομένων. Τότε δίνετε έναν παλμό στο latch και μεταφέρονται τα 8 bits δεδομένων στα αντίστοιχα pins του καταχωρητή μετατόπισης. Για βαθύτερη μελέτη της λειτουργίας του ανατρέξτε στο σχετικό λήμμα της Wikipedia (http://en.wikipedia.org/wiki/Shift_register).

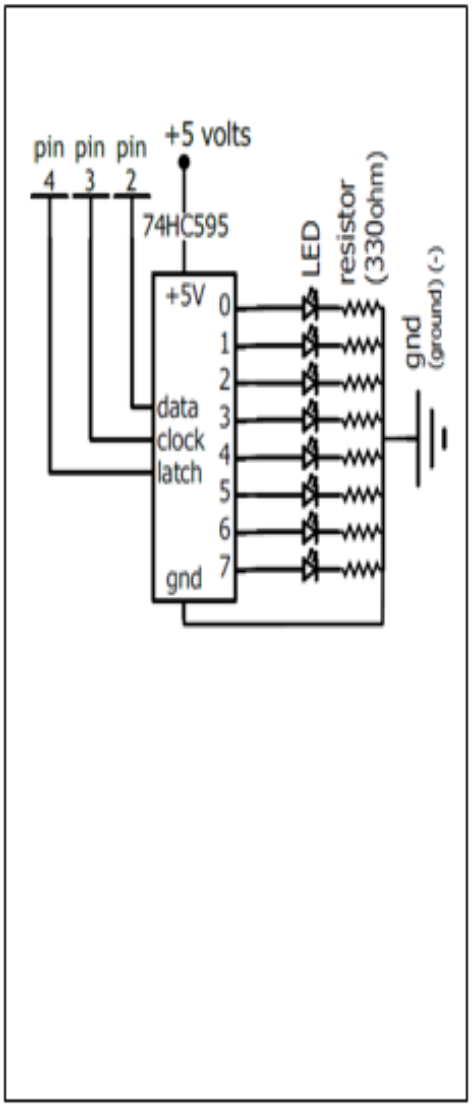
Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 45 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-05 απαιτούνται τα εξής μέρη:

- 8 κόκκινα LED των 5mm
- 8 αντιστάσεις των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 19 καλώδια
- 1 Καταχωρητής ολίσθησης 74HC595



Σχήμα Β΄.15: Προτεινόμενη υλοποίηση του CIRC05



Σχήμα Β΄.16: Σχηματικό Διάγραμμα του CIRC05

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.16. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.15.

Τοποθετήστε τον καταχωρητή ολίσθησης στο breadboard. Η ημικυκλική εγκοπή του, θα πρέπει να είναι σύμφωνη με τη φορά του Σχήματος Β΄.15. Συνδέστε το θετικό ακροδέκτη του κάθε LED στο αναγραφόμενο pin του καταχωρητή ολίσθησης, και μια αντίσταση 330 Ohm στον κάθε αρνητικό ακροδέκτη, που να πηγαίνει στη γείωση. Συνδέστε το 8ο και το 13ο pin του καταχωρητή ολίσθησης στη γείωση καθώς και το 10ο και 16ο στο +5 V. Συνδέστε τα pin 11, 12 και 14 του καταχωρητή ολίσθησης στα αναγραφόμενα pins του Arduino. *** **Προσοχή: Αν συνδέσετε ανάποδα το IC του καταχωρητή ολίσθησης μπορεί να καταστρέψει το ολοκληρωμένο κύκλωμα. Διαβάστε το datasheet και στη συνέχεια ακολουθήστε πιστά τις οδηγίες σύνδεσης.** *** Τέλος συνδέστε μεταξύ τους τις γειώσεις και τα +5 V του breadboard. Στο σημείο αυτό θα πρέπει να γνωρίζετε κάποιες βασικές πληροφορίες για τον καταχωρητή μετατόπισης (Serial to parallel converter): Διαβάστε το σχετικό λήμμα στη Wikipedia (http://en.wikipedia.org/wiki/Shift_register). Στο παρόν κύκλωμα η χρήση του καταχωρητή μετατόπισης θα μας δώσει 8 επιπλέον εξόδους για τον έλεγχο των LEDs, χρησιμοποιώντας μόνο 3 pins του Arduino.

Β΄.13 Προγραμματισμός του κυκλώματος CIRC-05

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE05.

```
/*       ------------------------------------------------------------
 *|   Arduino Experimentation Kit Example Code |
 *|   CIRC-05 .: 8 More LEDs :. (74HC595 Shift Register) |
          *------------------------------------------------------------
 *
 *We have already controlled 8 LEDs however this does it in a slightly
 *different  manner. Rather than using 8 pins we will use just three
 *and an additional chip.
 *
 */
//Pin Definitions
//Pin Definitions
//The 74HC595 uses a serial communication
//link which has three pins
int data = 2;
int clock = 3;
int latch = 4;

//Used for single LED manipulation
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;

/*
 * setup() - this function runs once when you turn your Arduino on
 * We set the three control pins to outputs
```

```
 */
void setup()
{
  pinMode(data, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT);
}


/*
 * loop() - this function will start after setup finishes and then repeat
 * we set which LEDs we want on then call a routine which sends to the 74HC595
 */
void loop()  // run over and over again
{
 int delayTime = 100; //the number of milliseconds to delay between LED updates
 for(int i = 0; i < 256; i++){
  updateLEDs(i);
  delay(delayTime);
 }
}

/*
 *updateLEDs() - sends the LED states set in ledStates to the 74HC595
 *sequence
 */
void updateLEDs(int value){
 digitalWrite(latch, LOW); //Pulls the chips latch low
 shiftOut(data, clock, MSBFIRST, value); //Shifts out the 8 bits to the
    //shift register
  digitalWrite(latch, HIGH); //Pulls the latch high displaying the data
}

/*
 * updateLEDsLong() - sends the LED states set in ledStates to the 74HC595
 * sequence. Same as updateLEDs except the shifting out is done in software
 * so you can see what is happening.
 */
void updateLEDsLong(int value){
  digitalWrite(latch, LOW); //Pulls the chips latch low
  for(int i = 0; i < 8; i++){ //Will repeat 8 times (once for each bit)
  int bit = value & B10000000; //We use a "bitmask" to select only the eighth
   //bit in our number (the one we are addressing this time through
  value = value << 1; //we move our number up one bit value
   //so next time bit 7 will be
   //bit 8 and we will do our math on it
  if(bit == 128){digitalWrite(data, HIGH);}
//if bit 8 is set then set our data pin high
  else{digitalWrite(data, LOW);} //if bit 8 is unset then set the data pin low
  digitalWrite(clock, HIGH); //the next three lines pulse the clock pin
  delay(1);
  digitalWrite(clock, LOW);
  }
  digitalWrite(latch, HIGH);
//pulls the latch high shifting our data into being displayed
}

//These are used in the bitwise math that we use to change individual LEDs
//For more details http://en.wikipedia.org/wiki/Bitwise_operation
int bits[] = {B00000001, B00000010, B00000100, B00001000, B00010000,\
 B00100000, B01000000, B10000000};
int masks[] = {B11111110, B11111101, B11111011, B11110111, B11101111,\
```

```
 B11011111, B10111111, B01111111};
/*
 * changeLED(int led, int state) - changes an individual LED
 * LEDs are 0 to 7 and state is either 0 - OFF or 1 - ON
 */
void changeLED(int led, int state){
  //clears ledState of the bit we are addressing
  ledState = ledState & masks[led];
  if(state == ON){ledState = ledState | bits[led];}
  //if the bit is on we will add it to ledState
  updateLEDs(ledState); //send the new LED state to the shift register
 }
```

Αποθηκεύστε το ως CIRC_05 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Αν το Arduino σβήνει, ελέγξτε μήπως έχετε τοποθετήσει ανάποδα τον καταχωρητή ολίσθησης.

Β΄.14 Παραμετροποίηση του κυκλώματος CIRC-05

Υλοποίηση του προγράμματος δίχως τη χρήση της έτοιμης συνάρτησης shiftOut(): Αλλάξτε στη loop() τη γραμμή updateLEDs(i); σε updateLEDsLong(i); και φορτώστε το πρόγραμμα στην πλακέτα. Παρατηρήστε ότι ενώ δεν αλλάζει κάτι στη λειτουργία του κυκλώματος, ο κώδικας, μας επιτρέπει πλέον να 'επικοινωνούμε' με το chip 1 bit τη φορά. Για περισσότερες λεπτομέρειες διαβάστε τα σχόλια του κώδικα της συνάρτησης updateLEDsLong() και για μεγαλύτερη εμβάθυνση μπορείτε να ανατρέξτε στο σχετικό λήμμα της Wikipedia (http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus).

Έλεγχος μεμονωμένων LEDs: Μπορείτε επίσης να ελέγξετε μεμονωμένα LEDs, όπως είχατε κάνει στο CIRC-02, χρησιμοποιώντας την changeLED(<LED>,<ON/OFF>); Για παράδειγμα στη loop(), αντικαταστήστε τον υπάρχοντα κώδικα με τον παρακάτω, αποθηκεύστε το ως CIRC_05_c1 και φορτώστε το στο Arduino:

```
//the number of milliseconds to delay between LED updates
int delayTime = 100;
 for(int i = 0; i < 8; i++){
  changeLED(i,ON);
  delay(delayTime);
 }
 for(int i = 0; i < 8; i++){
  changeLED(i,OFF);
  delay(delayTime);
 }
```

Υλοποίηση ενός δικού σας LED animation: Ορίστε μια δική σας συνάρτηση με όνομα my_animation(), η οποία θα υλοποιεί ένα LED animation και καλέστε την, όπως έχει περιγραφεί στο CIRC-02. Μπορείτε αν θέλετε να χρη-

σιμοποιήσετε την ίδια συνάρτηση που είχατε υλοποιήσει στο CIRC-02, τροποποιώντας την κατάλληλα, ώστε να κάνει compile χωρίς σφάλματα (αλλαγή της digitalWrite() σε changeLED() κτλ). Αποθηκεύστε το ως CIRC_05_c2 και φορτώστε το στο Arduino.

Β΄.15 6η εργαστηριακή άσκηση

Το Arduino μπορεί να χρησιμοποιηθεί και για τον έλεγχο συσκευών που παράγουν ήχο, παρόλο που ο ήχος είναι αναλογικό μέγεθος. Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε ένα πιεζοηλεκτρικό στοιχείο buzzer. Το πιεζοηλεκτρικό στοιχείο παράγει έναν ήχο κάθε φορά που το διαπερνά ένας παλμός ρεύματος. Αν πάλλουμε το πιεζοηλεκτρικό στοιχείο στην κατάλληλη συχνότητα οι παραγόμενοι ήχοι θα παράγουν μουσικές νότες (π.χ. 440Hz για να παραχθεί η νότα λα).

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 20 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-06 απαιτούνται τα εξής μέρη:

- 1 Πιεζοηλεκτρικό στοιχείο
- 4 Καλώδια

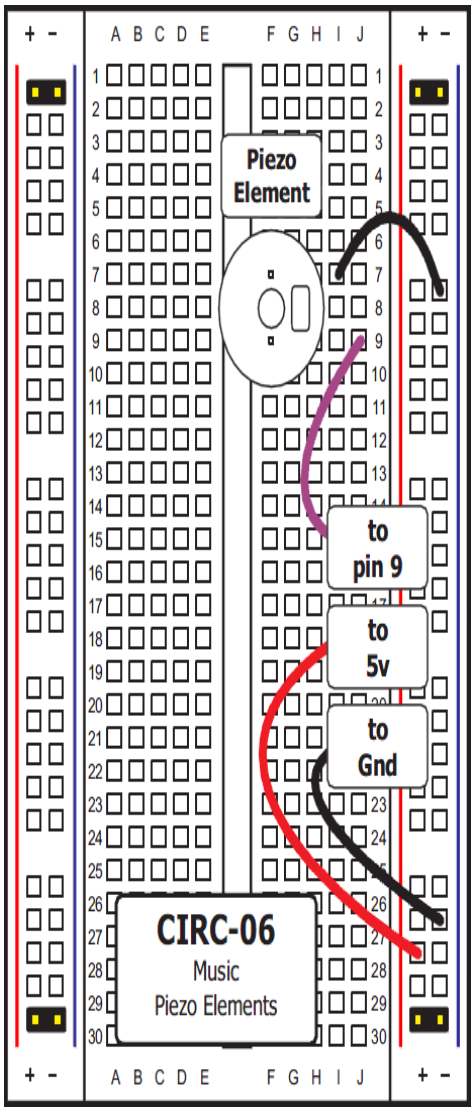
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.18. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.17.

Συνδέστε το θετικό ακροδέκτη του πιεζοηλεκτρικού στοιχείου στο pin 9 του Arduino, και τον αρνητικό ακροδέκτη στη γείωση.

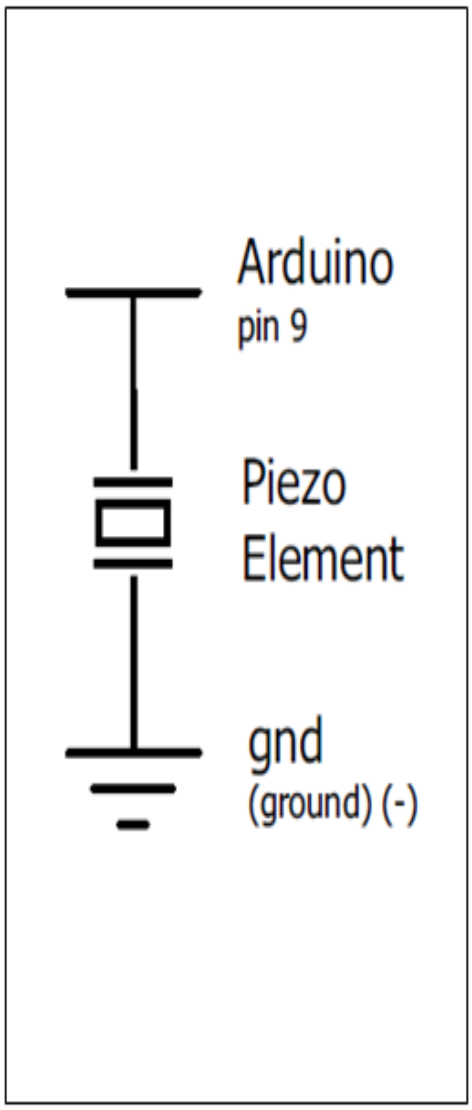
Β΄.16 Προγραμματισμός του κυκλώματος CIRC-06

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE06.

```
1
2  /* Melody
3   * (cleft) 2005 D. Cuartielles for K3
4   *
5   * This example uses a piezo speaker to play melodies. It sends
6   * a square wave of the appropriate frequency to the piezo, generating
7   * the corresponding tone.
8   *
9   * The calculation of the tones is made following the mathematical
10  * operation:
11  *
12  *  timeHigh = period / 2 = 1 / (2 * toneFrequency)
13  *
14  * where the different tones are described as in the table:
15  *
```



Σχήμα Β΄.17: Προτεινόμενη υλοποίηση του CIRC06



Σχήμα Β΄.18: Σχηματικό Διάγραμμα του CIRC06

```
16 *note frequency period  timeHigh
17 * c    261 Hz  3830  1915
18 * d    294 Hz  3400  1700
19 * e    329 Hz  3038  1519
20 * f    349 Hz  2864  1432
21 * g    392 Hz  2550  1275
22 * a    440 Hz  2272  1136
23 * b    493 Hz  2028  1014
24 * C    523 Hz      1912  956
25 *
26 * http://www.arduino.cc/en/Tutorial/Melody
27 */
28
29 int speakerPin = 9;
30
31 int length = 15; // the number of notes
32 // a space represents a rest
33 char notes[] = "ccggaagffeeddc ";
34 int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
35 int tempo = 300;
36
37 void playTone(int tone, int duration) {
38  for (long i = 0; i < duration * 1000L; i += tone * 2) {
39   digitalWrite(speakerPin, HIGH);
40   delayMicroseconds(tone);
41   digitalWrite(speakerPin, LOW);
42   delayMicroseconds(tone);
43  }
44 }
45
46 void playNote(char note, int duration) {
47  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
48  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
49
50  // play the tone corresponding to the note name
51  for (int i = 0; i < 8; i++) {
52   if (names[i] == note) {
53   playTone(tones[i], duration);
54 }
55 }
56 }
57
58 void setup() {
59   pinMode(speakerPin, OUTPUT);
60 }
61
62 void loop() {
63  for (int i = 0; i < length; i++){
64   if (notes[i] == ' '){
65   delay(beats[i] * tempo); // rest
66   } else {
67   playNote(notes[i], beats[i] * tempo);
68  }
69
70  // pause between notes
71   delay(tempo / 2);
72  }
73 }
```

Αποθηκεύστε το ως CIRC_06 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Αν το Arduino σβήνει, ελέγξτε μήπως έχετε τοποθετήσει ανά-

ποδα τον καταχηρητή ολίσθησης.

Β΄.17 Παραμετροποίηση του κυκλώματος CIRC-06

Ρύθμιση της ταχύτητας της μελωδίας: Η ταχύτητα της μελωδίας καθορίζεται από την τιμή της μεταβλητής tempo. Μεγαλύτερες τιμές, δίνουν γρηγορότερη εκτέλεση της μελωδίας, ενώ μικρότερες πιο αργή. Δοκιμάστε να αλλάξετε την τιμή της στο παρακάτω τμήμα του κώδικα: int tempo = 300; Στη συνέχεια, αποθηκεύστε και φορτώστε το πρόγραμμα στο Arduino ώστε να διαπιστώσετε την αλλαγή στη λειτουργία του κυκλώματος.

Έλεγχος τονικότητας κάθε νότας: Μπορείτε επίσης να αλλάξετε την τονικότητα κάθε νότας, αλλάζοντας τις τιμές του πίνακα tones[ ]. int tones[] = 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 ; Κάθε τιμή του πίνακα είναι η αντίστοιχη νότα στον πίνακα names[ ].

Σύνθεση μιας δικιάς σας μελωδίας: Το πρόγραμμα είναι ρυθμισμένο να παίζει το παιδικό τραγούδι 'Twinkle Twinkle Little Star', αλλά η αλλαγή του είναι απλή υπόθεση. Αλλάξτε το παρακάτω τμήμα του κώδικα:

```
1 int length = 15; // the number of notes
2 char notes[] = "ccggaagffeeddc "; // a space represents a rest
3 int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

με αυτό

```
1 int length = 13; // the number of notes
2 char notes[] = "ccdcfeccdcgf "; // a space represents a rest
3 int beats[] = { 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 4 };
```

Αφού το δοκιμάσετε, προσπαθήστε να συνθέσετε τη δικιά σας μελωδία και παραδώστε τον κώδικα. Αποθηκεύστε το ως CIRC_06_c1 και φορτώστε το στο Arduino.

Β΄.18 7η εργαστηριακή άσκηση

Το Arduino μπορεί να χρησιμοποιηθεί και για να διαβάσει δεδομένα από συσκευές εισόδου. Στο παρόν εργαστήριο θα χρησιμοποιήσουμε κουμπιά ως συσκευές εισόδου. Το Arduino ελέγχει την τιμή της τάσης στο pin που έχουμε συνδεδεμένο το κουμπί και με βάση την τιμή αυτή θέτει την τιμή σε HIGH ή LOW. Το κουμπί είναι ρυθμισμένο να ανεβάζει την τιμή LOW όταν το πατήσουμε, όμως αν το κουμπί δεν έχει πατηθεί η τιμή της τάσης μπορεί να είναι κυμαινόμενη και να προκαλεί σφάλματα στη λειτουργία του κυκλώματος. Για το λόγο αυτό χρησιμοποιούνται αντιστάσεις pull up στα κουμπιά, ώστε όταν δεν έχουν πατηθεί, το pin να παίρνει την τιμή HIGH.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 35 λεπτά
Για την εκπόνηση του κυκλώματος CIRC-07 απαιτούνται τα εξής μέρη:

- 1 κόκκινο LED των 5mm
- 1 αντίσταση των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 7 καλώδια
- 1 Καταχωρητής ολίσθησης 74HC595
- 2 Αντιστάσεις 10k Ohm (Καφέ – Μαύρο – Πορτοκαλί)

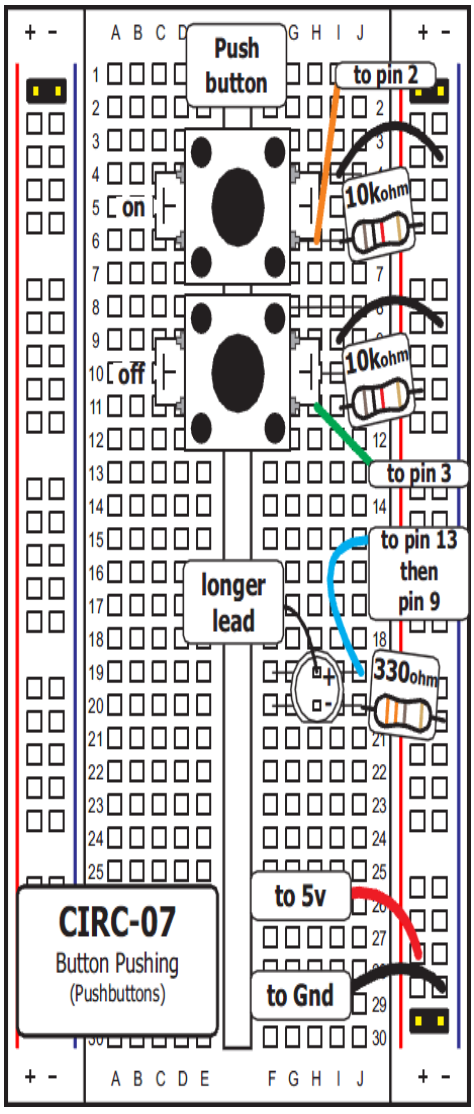
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.20. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.19.

Συνδέστε το θετικό ακροδέκτη του LED στο pin 13 του Arduino (το pin 9 που αναγράφεται στο Σχήμα Β΄.20, θα χρησιμοποιηθεί σε παρακάτω ερώτημα) και τον αρνητικό ακροδέκτη σε μια αντίσταση 330 Ohm, η άλλη άκρη της οποίας συνδέεται στη γείωση. Τοποθετήστε στο breadboard τα κουμπιά και συνδέστε το ένα pin στη γείωση και το άλλο με μια αντίσταση 10k Ohm, η οποία συνδέεται στα +5 V

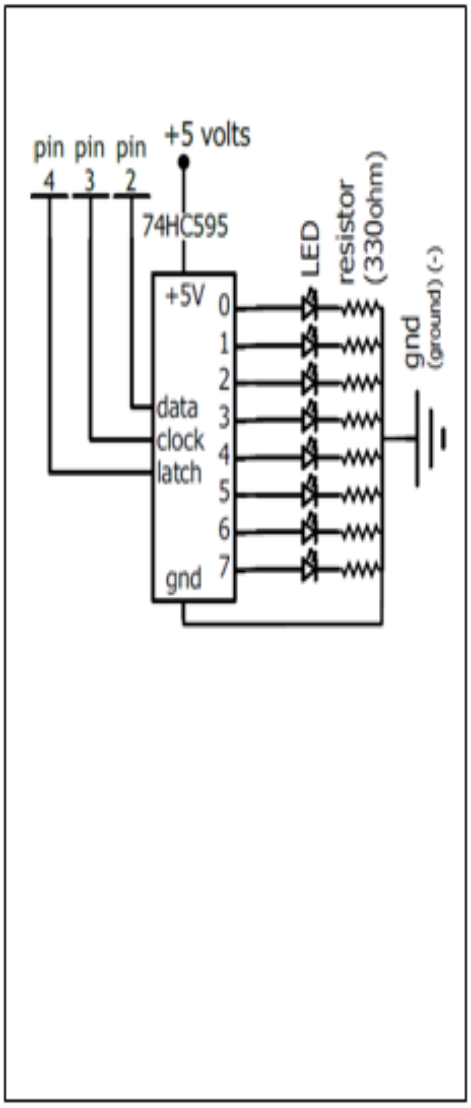
Β΄.19 Προγραμματισμός του κυκλώματος CIRC-07

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE07.

```
1  /*
2  Button: Turns on and off a light emitting diode(LED) connected to digital
3  pin 13, when pressing a pushbutton attached to pin 7.
4
5  The circuit:
6  * LED attached from pin 13 to ground
7  * pushbutton attached to pin 2 from +5V
8  * 10K resistor attached to pin 2 from ground
9
10 * Note: on most Arduinos there is already an LED on the board
11 attached to pin 13.
12
13 created 2005
14 by DojoDave <http://www.0j0.org>
15 modified 17 Jun 2009
16 by Tom Igoe
17
18 http://www.arduino.cc/en/Tutorial/Button
19 */
20
21 // constants won't change. They're used here to
22 // set pin numbers:
23 const int buttonPin = 2; // the number of the pushbutton pin
```



Σχήμα Β΄.19: Προτεινόμενη υλοποίηση του CIRC07



Σχήμα Β΄.20: Σχηματικό Διάγραμμα του CIRC07

```
const int ledPin = 13; // the number of the LED pin

// variables will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
 // read the state of the pushbutton value:
 buttonState = digitalRead(buttonPin);

// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
 if (buttonState == HIGH){
  // turn LED on:
  digitalWrite(ledPin, HIGH);
}
 else {
 // turn LED off:
  digitalWrite(ledPin, LOW);
 }
}
```

Αποθηκεύστε το ως CIRC_07 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα.

Β΄.20 Παραμετροποίηση του κυκλώματος CIRC-07

Χρήση των κουμπιών: Δημιουργήστε ένα νέο sketch και αντιγράψτε τον παρακάτω κώδικα:

```
int ledPin = 13; // choose the pin for the LED
int inputPin1 = 3; // button 1
int inputPin2 = 2; // button 2

void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin1, INPUT); // make button 1 an input
pinMode(inputPin2, INPUT); // make button 2 an input
}
void loop(){
 if (digitalRead(inputPin1) == LOW) {
digitalWrite(ledPin, LOW); // turn LED OFF
}

 if (digitalRead(inputPin2) == LOW) {
  digitalWrite(ledPin, HIGH); // turn LED ON
}
}
```

Στη συνέχεια, αποθηκεύστε το ως CIRC_07_c1 και φορτώστε το πρόγραμμα στο Arduino. Επιβεβαιώστε την αλλαγή στη λειτουργία του κυκλώματος.

LED fading: Μπορείτε επίσης να χρησιμοποιήσετε τα κουμπιά για να ελέγξετε αναλογικό σήμα. Στο σημείο αυτό θα χρειαστεί να αλλάξετε το καλώδιο στο θετικό pin του LED, από το pin 13 του Arduino, στο pin 9. Επίσης κάντε την αντίστοιχη αλλαγή και στον κώδικα (του CIRC_07_c1).

const int ledPin = 13; ⟶ const int ledPin = 9;

Τέλος αλλάξτε τη loop() με τον παρακάτω κώδικα:

```
int value = 0;
void loop(){
  if (digitalRead(inputPin1) == LOW) { value--; }
  else if (digitalRead(inputPin2) == LOW) { value++; }
  value = constrain(value, 0, 255);
  analogWrite(ledPin, value);
  delay(10);
}
```

Αποθηκεύστε το ως CIRC_07_c2 και φορτώστε το πρόγραμμα στο Arduino. Επιβεβαιώστε την αλλαγή στη λειτουργία του κυκλώματος.

Ρύθμιση της ταχύτητας του fading: Αλλάζετε την τιμή στο παρακάτω τμήμα του κώδικα CIRC_07_c2: delay(10);

Β΄.21 8η εργαστηριακή άσκηση

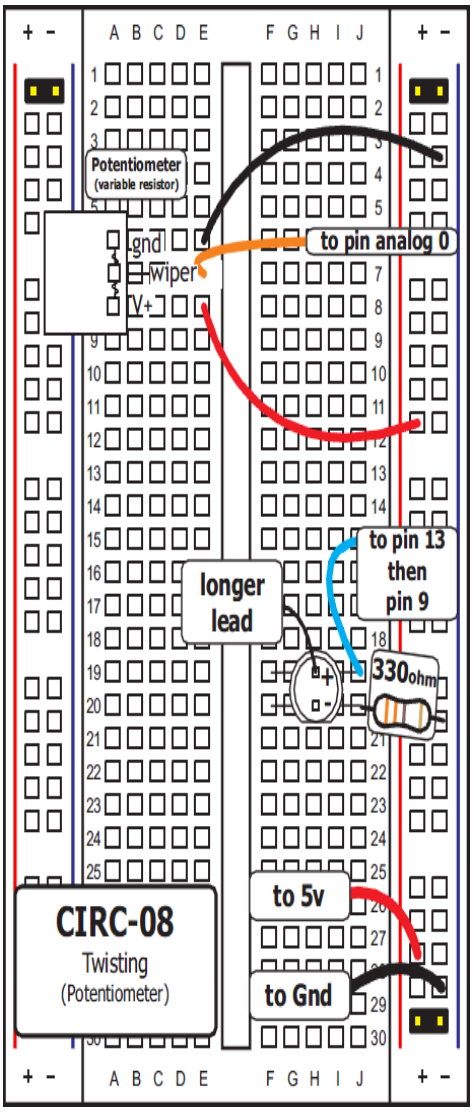
Πέρα από τα ψηφιακά pins, το Arduino έχει επίσης 6 pins τα οποία μπορούν να χρησιμοποιηθούν για αναλογικές εισόδους. Αυτές οι είσοδοι δέχονται μια τιμή τάσης (μεταξύ 0 και 5 volts DC) και τη μετατρέπουν σε έναν αριθμό από 0 ως 1024 (ανάλυση 10 bit). Μια χρήσιμη συσκευή που εκμεταλλεύεται τις εισόδους αυτές είναι το ποτενσιόμετρο (μεταβλητή αντίσταση). Περιστρέφοντας τον διακόπτη του μπορούμε να μεταβάλλουμε την τιμή της τάσης στο pin εισόδου και να χρησιμοποιήσουμε τις παραγόμενες τιμές σαν μεταβλητές στο πρόγραμμά μας.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 25 λεπτά

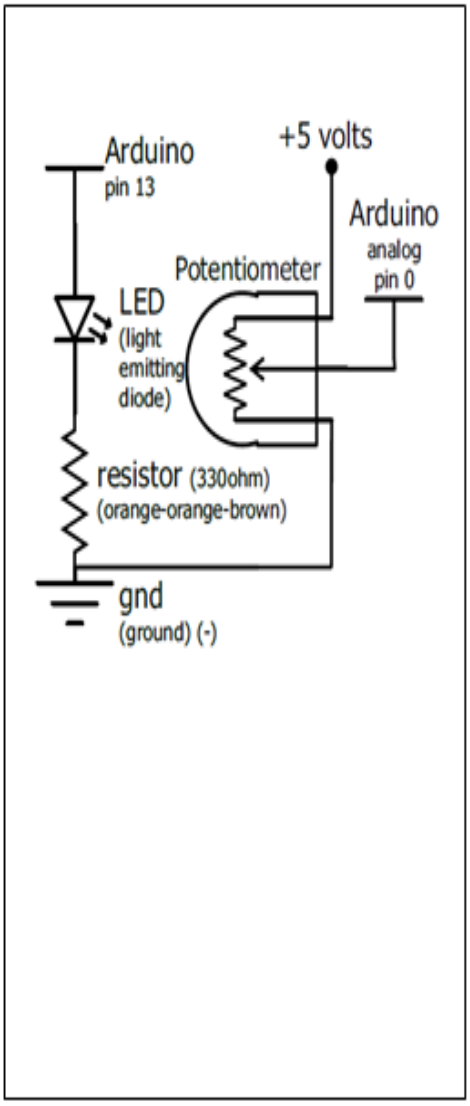
Για την εκπόνηση του κυκλώματος CIRC-08 απαιτούνται τα εξής μέρη:

- 1 κίτρινο LED των 5mm
- 1 αντίσταση των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 4 καλώδια
- 1 Ποτενσιόμετρο 10kOhm

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.22. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.21.



Σχήμα Β΄.21: Προτεινόμενη υλοποίηση του CIRC08



Σχήμα Β΄.22: Σχηματικό Διάγραμμα του CIRC08

Συνδέστε το θετικό ακροδέκτη του LED στο pin 13 του Arduino (το pin 9 που αναγράφεται στο Σχήμα Β΄.21, θα χρησιμοποιηθεί σε παρακάτω ερώτημα) και τον αρνητικό ακροδέκτη σε μια αντίσταση 330 Ohm, η άλλη άκρη της οποίας συνδέεται στη γείωση. Τοποθετήστε στο breadboard το ποτενσιόμετρο και συνδέστε το ένα pin στη γείωση, το μεσαίο στο αναλογικό pin 0 του Arduino και το τρίτο στα +5 V.

Β΄.22 Προγραμματισμός του κυκλώματος CIRC-08

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE08.

```
/*
Analog Input
Demonstrates analog input by reading an analog sensor on analog pin 0 and
turning on and off a light emitting diode(LED) connected to digital pin 13.
The amount of time the LED will be on and off depends on
the value obtained by analogRead().

The circuit:
*Potentiometer attached to analog input 0
*center pin of the potentiometer to the analog pin
*one side pin (either one) to ground
*the other side pin to +5V
*LED anode (long leg) attached to digital output 13
*LED cathode (short leg) attached to ground

*Note: because most Arduinos have a built-in LED attached
to pin 13 on the board, the LED is optional.


Created by David Cuartielles
Modified 16 Jun 2009
By Tom Igoe

http://arduino.cc/en/Tutorial/AnalogInput

*/

int sensorPin = 0; // select the input pin for the potentiometer
int ledPin = 13; // select the pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor

void setup(){
 // declare the ledPin as an OUTPUT:
 pinMode(ledPin, OUTPUT);
}

void loop(){
// read the value from the sensor:
 sensorValue = analogRead(sensorPin);
 // turn the ledPin on
 digitalWrite(ledPin, HIGH);
 // stop the program for <sensorValue> milliseconds:
 delay(sensorValue);
 // turn the ledPin off:
```

```
45  digitalWrite(ledPin, LOW);
46  // stop the program for for <sensorValue> milliseconds:
47  delay(sensorValue);
48 }
```

Αποθηκεύστε το ως CIRC_08 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Σε περίπτωση που το κύκλωμα δε λειτουργεί, ελέγξτε αν έχετε συνδέσει το ποτενσιόμετρο με το αναλογικό pin 0 (A0 στο Arduino).

Β΄.23 Παραμετροποίηση του κυκλώματος CIRC-04

Ενεργοποίηση με τη χρήση τιμής - κατωφλίου: Αντικαταστήστε την loop() του CIRC_08 με την παρακάτω:

```
1 void loop() {
2   int threshold = 512;
3   if(analogRead(sensorPin) > threshold){ digitalWrite(ledPin, HIGH);}
4   else{ digitalWrite(ledPin, LOW);}
5 }
```

Στη συνέχεια, αποθηκεύστε το ως CIRC_08_c1 και φορτώστε το πρόγραμμα στο Arduino. Όταν η τιμή της αντίστασης ξεπεράσει τα 512 Ohm που έχουμε θέσει ως κατώφλι, τότε το LED ανάβει.

Έλεγχος φωτεινότητας: Μπορείτε επίσης να χρησιμοποιήσετε το ποτενσιόμετρο για να ελέγξετε τη φωτεινότητα του LED. Στο σημείο αυτό θα χρειαστεί να αλλάξετε το καλώδιο στο θετικό pin του LED, από το pin 13 του Arduino, στο pin 9. Επίσης κάντε την αντίστοιχη αλλαγή και στον κώδικα (του CIRC_08_c1).

const int ledPin = 13; ⟶ const int ledPin = 9;

Τέλος αλλάξτε ολόκληρη τη loop() με τον παρακάτω κώδικα:

```
1 void loop() {
2    int value = analogRead(sensorPin) / 4;
3    analogWrite(ledPin, value);
4 }
```

Αποθηκεύστε το ως CIRC_08_c2 και φορτώστε το πρόγραμμα στο Arduino. Επιβεβαιώστε την αλλαγή στη λειτουργία του κυκλώματος.

Έλεγχος Servo: Αντικαταστήστε το LED του παραπάνω κυκλώματος με ένα Servo. Συνδέστε το όπως στο κύκλωμα CIRC_04 (στο pin 9) και ανοίξτε το παράδειγμα που βρίσκεται στο: Αρχείο>Παραδείγματα>Servo>Knob Αλλάξτε τη γραμμή: int sensorpin = 0; σε: int sensorpin = 2; Αποθηκεύστε το ως CIRC_08_c3 και φορτώστε το πρόγραμμα στο Arduino. Επιβεβαιώστε την αλλαγή στη λειτουργία του κυκλώματος.

Β΄.24 9η εργαστηριακή άσκηση

Χρησιμοποιώντας την ίδια βασική αρχή λειτουργίας που περιγράφηκε στην εργαστηριακή άσκηση 8 μπορούμε να χρησιμοποιήσουμε έναν φωτοαντιστάτη σαν συσκευή εισόδου. Επειδή το Arduino δε μπορεί να μετρήσει την τιμή της αντίστασης, αλλά μόνο αυτή της τάσης, χρησιμοποιείται διαιρέτης τάσης. Ο φωτοαντιστάτης επιστρέφει υψηλή τιμή όταν βρίσκεται σε συνθήκες χαμηλού φωτισμού και χαμηλή τιμή όταν βρίσκεται σε καλά φωτισμένο χώρο.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 30 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-09 απαιτούνται τα εξής μέρη:

- 1 κίτρινο LED των 5mm
- 1 αντίσταση των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 1 αντίσταση 10 kOhm (Καφέ – Μαύρο – Πορτοκαλί)
- 6 καλώδια
- 1 Φωτοαντιστάτης
- 1 Servo

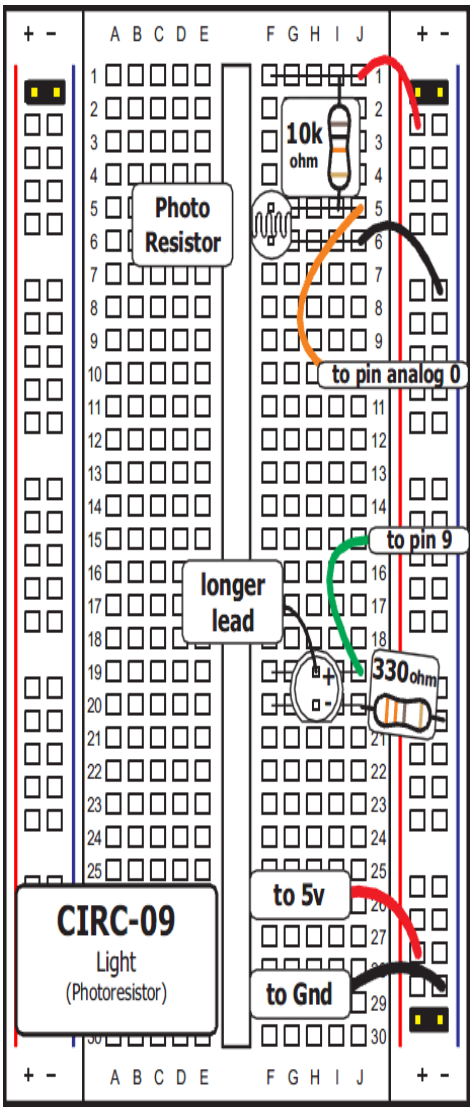
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.24. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.23.

Συνδέστε το θετικό ακροδέκτη του LED στο pin 9 του Arduino και τον αρνητικό ακροδέκτη σε μια αντίσταση 330 Ohm, η άλλη άκρη της οποίας συνδέεται στη γείωση. Τοποθετήστε στο breadboard το φωτοαντιστάτη και συνδέστε το ένα pin στη γείωση και το άλλο σε μια αντίσταση 10k Ohm που καταλήγει στα +5V καθώς και στο αναλογικό pin 0 ταυτόχρονα.

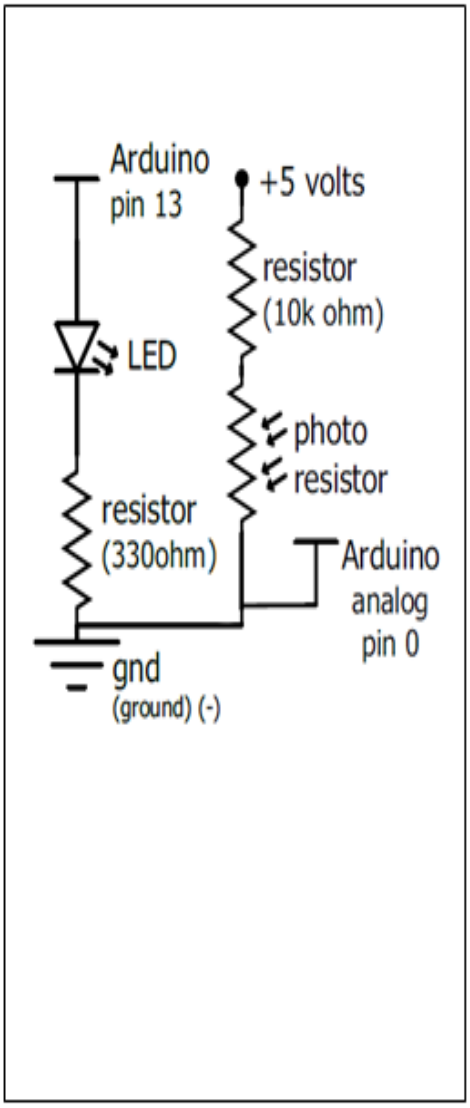
Β΄.25 Προγραμματισμός του κυκλώματος CIRC-09

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE09.

```
1  /*
2   Analog Input
3  Demonstrates analog input by reading an analog sensor on analog pin 0 and
4  turning on and off a light emitting diode(LED) connected to digital pin 13.
5  The amount of time the LED will be on and off depends on
6  the value obtained by analogRead().
7
8  The circuit:
9  *Potentiometer attached to analog input 0
10 *center pin of the potentiometer to the analog pin
```



Σχήμα Β΄.23: Προτεινόμενη υλοποίηση του CIRC09



Σχήμα Β΄.24: Σχηματικό Διάγραμμα του CIRC09

```
*one side pin (either one) to ground
*the other side pin to +5V
*LED anode (long leg) attached to digital output 13
*LED cathode (short leg) attached to ground

*Note: because most Arduinos have a built-in LED attached
to pin 13 on the board, the LED is optional.


Created by David Cuartielles
Modified 16 Jun 2009
By Tom Igoe

http://arduino.cc/en/Tutorial/AnalogInput

*/

int sensorPin = 0; // select the input pin for the potentiometer
int ledPin = 13; // select the pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {
 // declare the ledPin as an OUTPUT:
 pinMode(ledPin, OUTPUT);
}

void loop(){
 // read the value from the sensor:
 sensorValue = analogRead(sensorPin);
 // turn the ledPin on
 digitalWrite(ledPin, HIGH);
 // stop the program for <sensorValue> milliseconds:
 delay(sensorValue);
 // turn the ledPin off:
 digitalWrite(ledPin, LOW);
 // stop the program for for <sensorValue> milliseconds:
 delay(sensorValue);
}
```

Αφού επιβεβαιώστε τη λειτουργία, αντικαταστήσετε το loop με: int lightPin=sensorPin;

```
void loop()
{
 int lightLevel = analogRead(lightPin); //Read the lightlevel
 lightLevel = map(lightLevel, 0, 900, 0, 255); //adjust the value 0 to 900 to
   //span 0 to 255

 lightLevel = constrain(lightLevel, 0, 255);//make sure the
   //value is betwween
   //0 and 255
 analogWrite(ledPin, lightLevel); //write the value
}
```

Αποθηκεύστε το ως CIRC_09 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα.

Αν το LED δεν ανταποκρίνεται στις διακυμάνσεις φωτός και έχετε ελέγξει το κύκλωμά σας για πιθανά λάθη (πολικότητα LED, σωστά pins στο Arduino),

είναι πιθανό να φταίνε οι συνθήκες φωτισμού του εργαστηρίου. Δοκιμάστε μια πιο δυνατή πηγή φωτός κοντά στη φωτοαντίσταση (πχ φακός) και δείτε αν αυτό έχει αποτέλεσμα.

Β΄.26 Παραμετροποίηση του κυκλώματος CIRC-09

Αντιστροφή της λειτουργίας: Για να κάνετε το κύκλωμα να έχει την αντίστροφη απόκριση στις μεταβολές του φωτισμού αλλάξτε το παρακάτω τμήμα του κώδικα:

analogWrite(ledPin, lightLevel); με αυτό: analogWrite(ledPin, lightLevel); Στη συνέχεια, αποθηκεύστε το ως CIRC_09_c1 και φορτώστε το πρόγραμμα στο Arduino.

Ενεργοποίηση με χρήση τιμής - κατωφλίου: Αντικαταστήστε την loop() του CIRC_09 με την παρακάτω:

```
void loop(){
 int threshold = 300;
 if(analogRead(lightPin) > threshold){
  digitalWrite(ledPin, HIGH);
 }else{
  digitalWrite(ledPin, LOW);
 }
}
```

Τέλος αλλάξτε ολόκληρη τη loop() με τον παρακάτω κώδικα:

```
void loop() {
    int value = analogRead(sensorPin) / 4;
    analogWrite(ledPin, value);
}
```

Στη συνέχεια, αποθηκεύστε το ως CIRC_09_c2 και φορτώστε το πρόγραμμα στο Arduino.

Έλεγχος ενός Servo: Αντικαταστήστε το LED με ένα Servo το οποίο θα συνδέσετε όπως στο κύκλωμα CIRC_04 (στο pin 9). Επιβεβαιώστε ότι με τις διακυμάνσεις του φωτός στο φωτοαντιστάτη μπορείτε να ελέγξετε το σέρβο. Να τοποθετήσετε κατάλληλους ελέγχους, ώστε να στέλνονται τιμές από 0 (για σκοτάδι) έως 180 (μέγιστη φωτεινότητα) στο σέρβο. Στη συνέχεια ανοίξτε το παράδειγμα που βρίσκεται στο: Αρχείο>Παραδείγματα>Servo>Knob και φορτώστε το στο Arduino χωρίς αλλαγές. Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος.

Β΄.27 10η εργαστηριακή άσκηση

Στο παρόν εργαστήριο θα χρησιμοποιηθεί ένας αισθητήρας θερμοκρασίας ως συσκευή εισόδου. Έχει τρία pins, γείωση, σήμα και +5V και δίνει έξοδο 10

mV ανά μονάδα κελσίου στο pin σήματος. Η μετατροπή αυτής της εξόδου σε μονάδες μέτρησης θερμοκρασίας γίνεται με τη χρήση μαθηματικών συναρτήσεων. Στη συνέχεια για να εμφανίσουμε τις τιμές θα χρησιμοποιηθεί η Σειριακή Οθόνη του προγραμματιστικού περιβάλλοντος του Arduino.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 15 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-10 απαιτούνται τα εξής μέρη:

- 1 αισθητήρας θερμοκρασίας
- 5 καλώδια

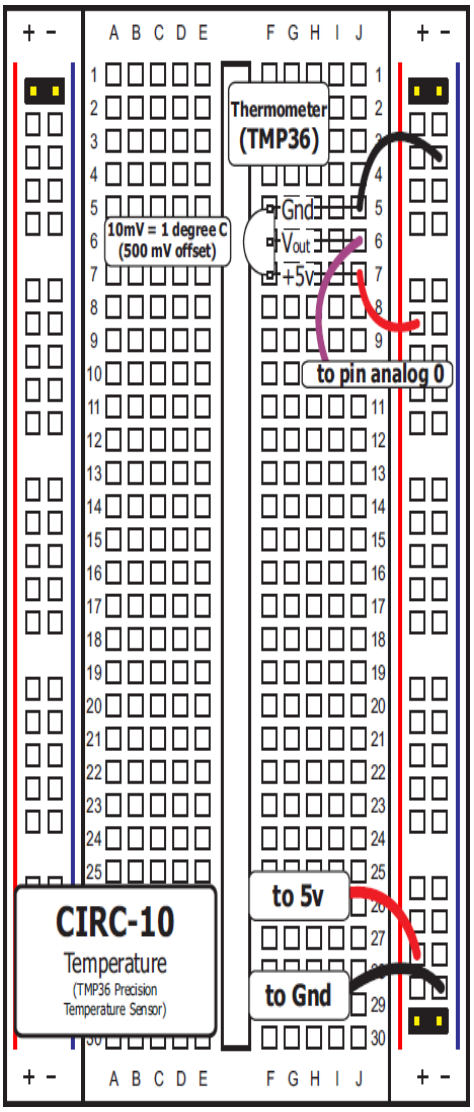
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.26. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.25.

Τοποθετήστε τον αισθητήρα θερμοκρασίας στο breadboard όπως υποδεικνύεται στο Σχήμα Β΄.25 και στη συνέχεια συνδέστε το ένα του άκρο στα +5V, το μεσαίο pin στο A0 του Arduino και τέλος το άλλο του άκρο στη γείωση.

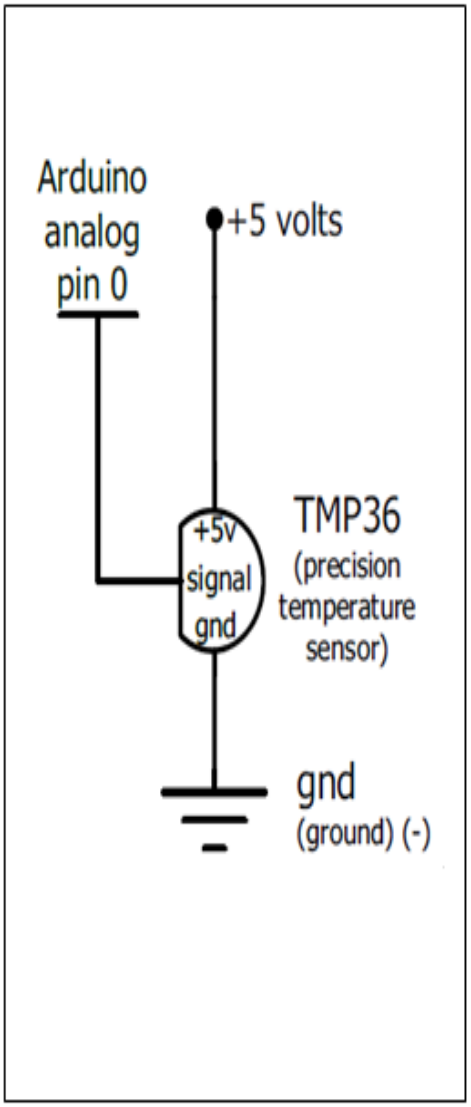
Β΄.28 Προγραμματισμός του κυκλώματος CIRC-03

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE10.

```
/*       ------------------------------------------------------
| Arduino Experimentation Kit Example Code |
*| CIRC-10 .: Temperature :. (TMP36 Temperature Sensor) |
*       ------------------------------------------------------
*
* A simple program to output the current temperature to the IDE's debug window
*
* For more details on this circuit: http://tinyurl.com/c89tvd
*/

//TMP36 Pin Variables
//the analog pin the TMP36's Vout (sense) pin is connected to
int temperaturePin = 0;
    //the resolution is 10 mV / degree centigrade
    //(500 mV offset) to make negative temperatures an option

/*
* setup() - this function runs once when you turn your Arduino on
* We initialize the serial connection with the computer
*/
void setup()
{
  Serial.begin(9600); //Start the serial connection with the computer
//to view the result open the serial monitor
//last button beneath the file bar (looks like a box with an antenae)
}

void loop() // run over and over again
{
//getting the voltage reading from the temperature sensor
```



Σχήμα Β΄.25: Προτεινόμενη υλοποίηση του CIRC10



Σχήμα Β΄.26: Σχηματικό Διάγραμμα του CIRC10

```
31   float temperature = getVoltage(temperaturePin);
32   //converting from 10 mv per degree wit 500 mV offset
33   temperature = (temperature - .5) * 100;
34  //to degrees ((volatge - 500mV) times 100)
35   Serial.println(temperature); //printing the result
36   delay(1000); //waiting a second
37  }
38
39  /*
40  * getVoltage() - returns the voltage on the analog input defined by
41  * pin
42  */
43  float getVoltage(int pin){
44  //converting from a 0 to 1023 digital range
45   return (analogRead(pin) * .004882814);
46  // to 0 to 5 volts (each 1 reading equals ~ 5 millivolts
47  }
```

Αποθηκεύστε το ως CIRC_10 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Για να δείτε τις μετρήσεις του αισθητήρα πρέπει να μεταβείτε στο προγραμματιστικό περιβάλλον του Arduino στο μενού Εργαλεία>Σειριακή οθόνη (Ctrl+Shift+M). Θα πρέπει να σας εμφανίζονται οι μετρήσεις με χρόνο ανανέωσης 1 sec. Αν εμφανίζονται "ασυναρτησίες", πρέπει να αλλάξετε την ταχύτητα σε 9600 baud.

Β΄.29 Παραμετροποίηση του κυκλώματος CIRC-10

Μορφοποίηση - μετατροπή των δεδομένων εξόδου: Για να μορφοποιήσετε την έξοδο τροποποιήστε το παρακάτω σημείο του κώδικα: Serial.println(temperature); σε: Serial.print(temperature);Serial.println("Deegrees centigrade"); ή οποιαδήποτε άλλη πληροφορία θα θέλατε να εκτυπώνεται στην έξοδο. Η μετατροπή των δεδομένων αντίστοιχα, γίνεται με μαθηματικό τρόπο.

Για να εμφανίσουμε τις μετρήσεις σε βαθμούς Fahrenheit αλλάξουμε την παρακάτω γραμμή στον κώδικά μας: temperature = (temperature - .5) * 100; με: temperature = (((temperature - .5) * 100)*1.8) + 32; Με παρόμοιο τρόπο μπορούμε να εμφανίσουμε τις μετρήσεις σε volts, αν διαγράψουμε τη γραμμή temperature = (temperature - .5) * 100; Αποθηκεύστε ως CIRC_10_c1, CIRC_10_c2, CIRC_10_c3, για καθεμιά από τις περιπτώσεις και φορτώστε τα στο Arduino για να διαπιστώσετε τις αλλαγές στη λειτουργία του προγράμματος.

Αλλαγή στο ρυθμό μετάδοσης συμβόλων (baud rate): Αν ποτέ χρειαστεί να έχετε σαν έξοδο μεγάλο όγκο δεδομένων θα πρέπει να αυξήσετε το baud rate. Στο πρόγραμμά μας είναι 9600, αλλά είναι επιτεύξιμες και πολύ μεγαλύτερες ταχύτητες. Αλλάξτε την παρακάτω γραμμή: Serial.begin(9600); σε: Serial.begin(115200); και μεταβείτε στη σειριακή οθόνη (Ctrl+Shift+M) όπου θα

αλλάξετε επίσης από 9600 σε 115200. Πλέον μπορείτε να μεταδίδετε δεδομένα 12 φορές γρηγορότερα.

Β΄.30 11η εργαστηριακή άσκηση

Στο παρόν εργαστήριο θα ελέγξουμε ένα ρελέ. Το ρελέ είναι ένας μαγνητικά ελεγχόμενος μηχανικός διακόπτης. Περιέχει έναν ηλεκτρομαγνήτη, ο οποίος όταν ενεργοποιείται μετακινεί έναν ενσωματωμένο διακόπτη. Χρησιμοποιείται ξανά το τρανζίστορ P2N2222AG και η δίοδος 1N4001 για τους λόγους που αναλύθηκαν στο εργαστήριο 3.

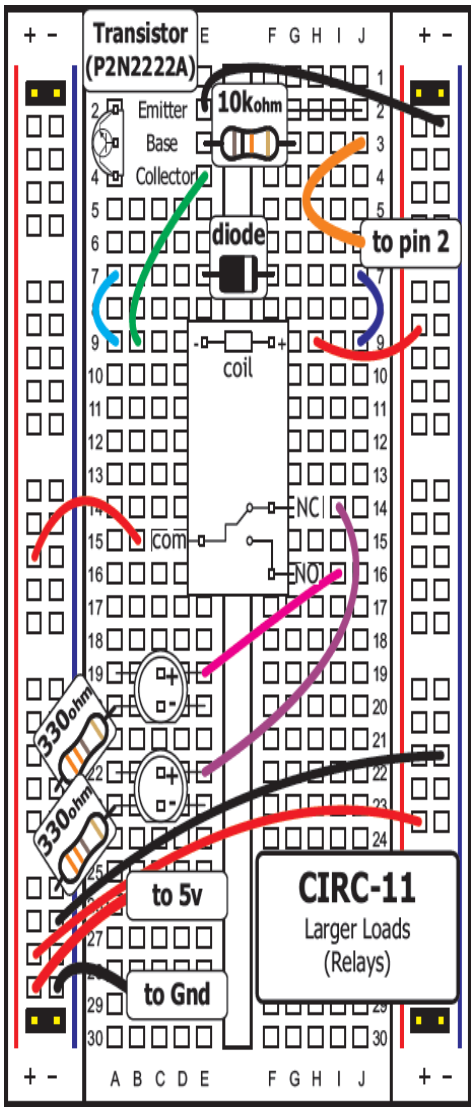
Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 30 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

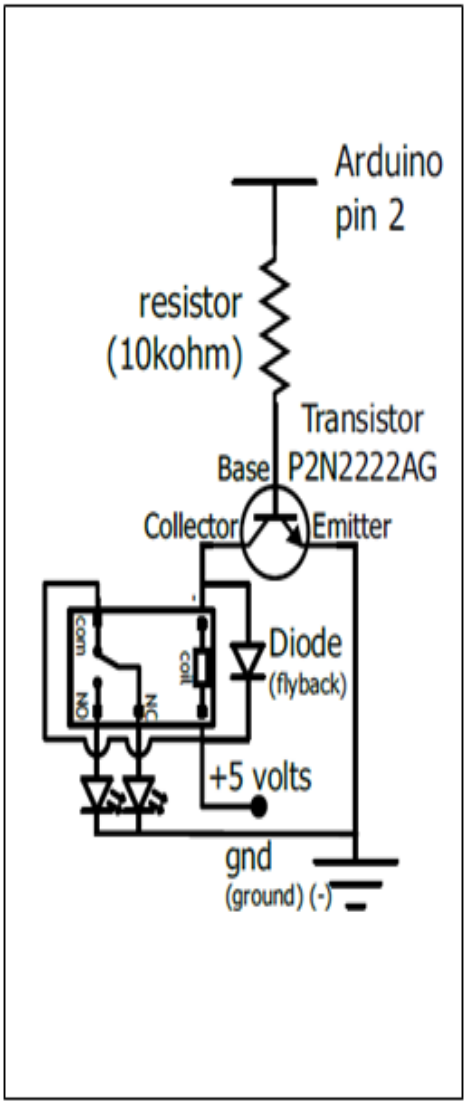
- 1 κίτρινο LED των 5mm
- 13 Καλώδια
- 2 Αντιστάσεις 330 Ohm (Πορτοκαλί – Πορτοκαλί – Καφέ)
- 1 Αντίσταση 10k Ohm (Καφέ – Μαύρο – Πορτοκαλί)
- 1 Ρελέ (SPDT)
- 1 Μοτέρ
- 1 κόκκινο LED
- 1 Τρανζίστορ P2N2222AG (TO92)
- 1 Δίοδος (1N4001)

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.28. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.27.

Τοποθετείστε το τρανζίστορ πάνω αριστερά, κατά τη φορά του Σχήματος Β΄.27, τη δίοδο και το ρελέ, επίσης κατά τη φορά του Σχήματος Β΄.27. Τοποθετήστε το κίτρινο LED με μία αντίσταση 330 Ohm στο αρνητικό του άκρο που να καταλήγει στη γείωση και όμοια από κάτω του το κόκκινο LED (δεν θα υπάρξει διαφορά στη λειτουργία αν τοποθετηθούν τα LED με την αντίθετη σειρά, απλά χρησιμοποιούμε αυτή τη διάταξη για να υπάρχει ένα σημείο αναφοράς στις μετέπειτα αλλαγές). Συνδέστε τον εκπομπό του τρανζίστορ στη γείωση, τη βάση με μία αντίσταση 10 kOhm της οποίας το άλλο άκρο πηγαίνει στο pin 2 του Arduino και το συλλέκτη με το αρνητικό άκρο του πηνίου του ρελέ (Σχήμα Β΄.27). Στη συνέχεια συνδέστε το αρνητικό άκρο του πηνίου με τη δίοδο, και στη



Σχήμα Β΄.27: Προτεινόμενη υλοποίηση του CIRC11



Σχήμα Β΄.28: Σχηματικό Διάγραμμα του CIRC11

συνέχεια το άλλο άκρο της διόδου με τη θετική πλευρά του πηνίου. Συνδέστε την επαφή NC του ρελέ με το θετικό άκρο του κόκκινου LED, την επαφή NO με το θετικό άκρο του κίτρινου LED και την επαφή com του ρελέ με τα +5V. Τέλος συνδέστε μεταξύ τους τις γειώσεις και τα +5V του breadboard.

Β΄.31 Προγραμματισμός του κυκλώματος CIRC-11

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE11.

```
1
2  /*
3   Blink
4   Turns on an LED on for one second, then off for one second, repeatedly.
5   The circuit:
6  * LED connected from digital pin 13 to ground.
7   * Note: On most Arduino boards, there is already an LED on the board
8  connected to pin 13, so you don't need any extra components for this example.
9
10 Created 1 June 2005
11 By David Cuartielles
12 http://arduino.cc/en/Tutorial/Blink
13 based on an orginal by H. Barragan for the Wiring i/o board
14 */
15
16 int ledPin = 2;
17 // Relay connected to digital pin 2 <-----Change this to pin 2
18
19 // The setup() method runs once, when the sketch starts
20
21 void setup(){
22   // initialize the digital pin as an output:
23   pinMode(ledPin, OUTPUT);
24 }
25
26 // the loop() method runs over and over again,
27 // as long as the Arduino has power
28
29 void loop()
30 {
31   digitalWrite(ledPin, HIGH); // set the LED on
32   delay(1000); // wait for a second
33   digitalWrite(ledPin, LOW); // set the LED off
34   delay(1000); // wait for a second
35 }
```

Αποθηκεύστε το ως CIRC_11 και στη συνέχεια φορτώστε το πρόγραμμα στην πλακέτα. Σε περίπτωση που δε δουλεύει όπως θα έπρεπε ελέγξτε ξανά αν έχουν τοποθετηθεί σωστά όλα τα μέρη του κυκλώματος. Επίσης ενδέχεται τα ρελέ να μην κάνουν καλή επαφή, καθώς δεν έχουν σχεδιαστεί για χρήση με το breadboard και γι αυτό το λόγο μπορεί να χρειάζονται λίγη επιπλέον πίεση προς τα κάτω (και ίσως να πετάγονται ξανά προς τα έξω περιστασιακά).

Β΄.32 Παραμετροποίηση του κυκλώματος CIRC-11

Παρατήρησης του επαγωγικού ρεύματος: Η μεταβολή της μαγνητικής ροής στο πηνίο γεννά ένα ηλεκτρικό ρεύμα, το λεγόμενο "επαγωγικό", που έχει αντίθετη φορά από εκείνο που παρέχεται στο πηνίο. Για να το παρατηρήσετε, αφού αποσυνδέσετε το breadboard από την πηγή ή το Arduino από την τροφοδοσία του (USB), αφαιρέστε τη δίοδο και στη θέση της τοποθετήστε ένα LED, το θετικό άκρο του οποίου να συνδέεται με το αρνητικό άκρο του πηνίου (αν το τοποθετήσετε ανάποδα το κύκλωμα δε θα δουλεύει σωστά) Αφού τοποθετήσετε το LED συνδέστε ξανά την πηγή και παρατηρήστε το να ανάβει στιγμιαία, λίγο αφότου σβήσει το κίτρινο LED που προϋπήρχε στο κύκλωμα.

Έλεγχος μοτέρ: Στο αρχικό κύκλωμα (πριν την αλλαγή της διόδου με το LED) και αφού αποσυνδέσετε το breadboard από την πηγή ή το Arduino από την τροφοδοσία του (USB), αφαιρέστε το κόκκινο LED μαζί με την αντίσταση 330 Ohm που συνδέεται στο αρνητικό του άκρο, και στη θέση του τοποθετήστε ένα μοτέρ, με το κόκκινο καλώδιο να συνδέεται στην NC και το μαύρο στη γείωση. Συνδέστε ξανά την πηγή και επιβεβαιώστε την ορθή λειτουργία του κυκλώματος.

Β΄.33 12η εργαστηριακή άσκηση

Εκτός από τα LEDs που έχουμε δει ως τώρα, τα οποία είναι μονόχρωμα, υπάρχουν και LEDs τα οποία μπορούν να παράγουν πολλά χρώματα. Ονομάζονται RGB LEDs και είναι τυπικά τρία LEDs (κόκκινο, πράσινο και μπλε) σε ένα. Όταν ενεργοποιούνται ταυτόχρονα και τα τρία LEDs τα φώτα τους ενώνονται και προκύπτουν διάφορα χρώματα. Τα χρώματα που προκύπτουν εξαρτώνται από τη φωτεινότητα κάθε LED ξεχωριστά. Η φωτεινότητα ελέγχεται με διαμόρφωση πλάτους παλμού (PWM), η λειτουργία της οποίας έχει αναλυθεί στην εργαστηριακή άσκηση 3.

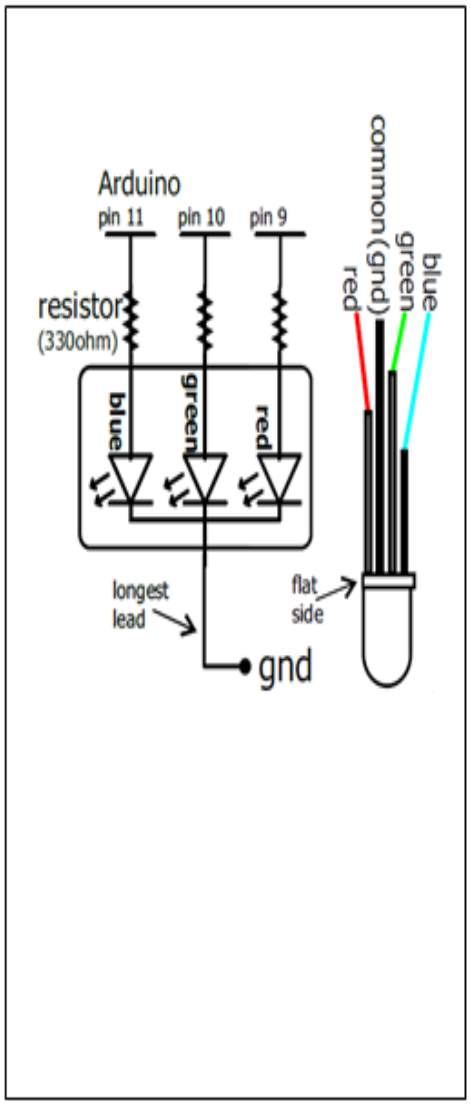
Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 30 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

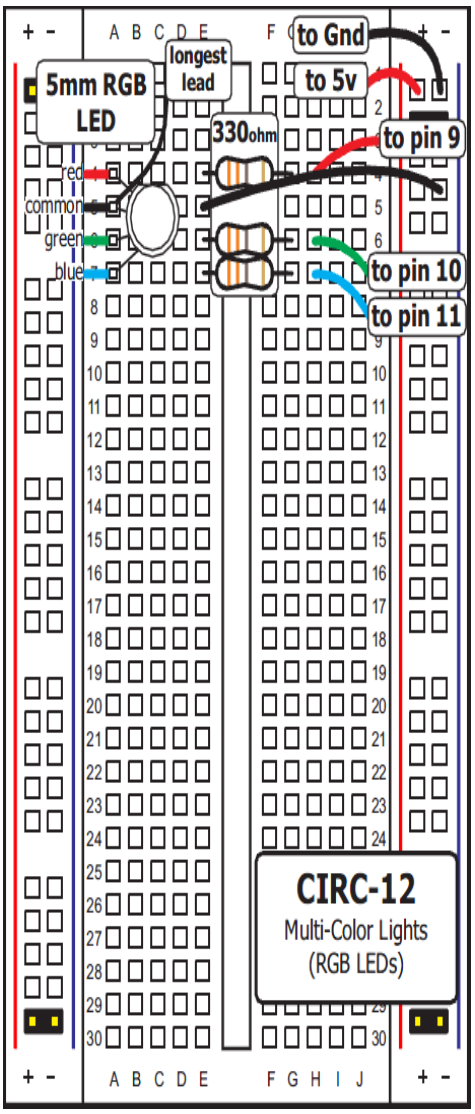
- 1 RGB LED των 5mm
- 3 αντιστάσεις των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 6 καλώδια

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.30. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.29.

Τοποθετήστε το RGB LED όπως φαίνεται στο Σχήμα Β΄.29. (Τα pins του LED αναλύονται στο Σχήμα Β΄.30). Στη συνέχεια συνδέστε από μία αντίσταση 330



Σχήμα Β΄.29: Προτεινόμενη υλοποίηση του CIRC12



Σχήμα Β΄.30: Σχηματικό Διάγραμμα του CIRC12

Ohm σε κάθε pin που αντιστοιχεί σε κάποιο χρώμα, και έπειτα κάθε αντίσταση με το ανάλογο pin του Arduino. Επίσης συνδέστε και το pin της γείωσης.

Β΄.34 Προγραμματισμός του κυκλώματος CIRC-03

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE12S.

```
/*

RGB_LED_Color_Fade_Cycle.pde

Cycles through the colors of a RGB LED

Written for SparkFun Arduino Inventor's Kit CIRC-RGB

*/

// LED leads connected to PWM pins
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;

// Used to store the current intensity level of the individual LEDs
int redIntensity = 0;
int greenIntensity = 0;
int blueIntensity = 0;

// Length of time we spend showing each color
const int DISPLAY_TIME = 100; // In milliseconds


void setup() {
 // No setup required.
}

void loop() {
// Cycle color from red through to green
// (In this loop we move from 100% red, 0% green to 0% red, 100% green)
 for (greenIntensity = 0; greenIntensity <= 255; greenIntensity+=5){
  redIntensity = 255-greenIntensity;
  analogWrite(GREEN_LED_PIN, greenIntensity);
  analogWrite(RED_LED_PIN, redIntensity);
  delay(DISPLAY_TIME);
 }

// Cycle color from green through to blue
// (In this loop we move from 100% green, 0% blue to 0% green, 100% blue)
 for (blueIntensity = 0; blueIntensity <= 255; blueIntensity+=5){
  greenIntensity = 255-blueIntensity;
  analogWrite(BLUE_LED_PIN, blueIntensity);
  analogWrite(GREEN_LED_PIN, greenIntensity);
  delay(DISPLAY_TIME);
}

// Cycle cycle from blue through to red
  // (In this loop we move from 100% blue, 0% red to 0% blue, 100% red)
  for (redIntensity = 0; redIntensity <= 255; redIntensity+=5) {
```

```
  blueIntensity = 255-redIntensity;
  analogWrite(RED_LED_PIN, redIntensity);
  analogWrite(BLUE_LED_PIN, blueIntensity);
  delay(DISPLAY_TIME);
}
}
```

Αποθηκεύστε το ως CIRC_12 και στη συνέχεια φορτώστε το πρόγραμμα στο Arduino. Σε περίπτωση που το LED δεν ανάβει, ή δείχνει λάθος χρώματα, ελέγξτε αν το κύκλωμά σας είναι σωστό.

Β΄.35 Παραμετροποίηση του κυκλώματος CIRC-12

Μείωση της υπερβολικής παρουσίας του κόκκινου χρώματος: Ενδέχεται το κόκκινο χρώμα του LED να είναι εντονότερο από τα άλλα και αυτό να έχει ως συνέπεια τα χρώματα που προκύπτουν να μην είναι τόσο ισορροπημένα. Αν συμβαίνει κάτι τέτοιο, μπορείτε να το διορθώσετε είτε βάζοντας μια αντίσταση περισσότερων Ohm στο pin του κόκκινου χρώματος, είτε αλλάζοντας την παρακάτω γραμμή στο πρόγραμμά σας, από: analogWrite(RED_LED_PIN, redIntensity); σε: analogWrite(RED_LED_PIN, redIntensity/3); Αποθηκεύστε (Ctrl+s) και φορτώστε ξανά το πρόγραμμα στο Arduino, για να επιβεβαιώσετε ότι το πρόβλημα λύθηκε.

Χρήση δεκαεξαδικών τριάδων για την απεικόνιση των χρωμάτων: Εάν έχετε δουλέψει με HTML ή CSS, πιθανότατα θα σας βολεύει περισσότερο να χρησιμοποιείτε δεκαεξαδικούς για να καθορίσετε το χρώμα που επιθυμείτε. Σε κάθε περίπτωση πάντως μπορείτε να επισκεφθείτε το σχετικό λήμμα στη Wikipedia (http://en.wikipedia.org/wiki/Web_colors#Hex_triplet) για περισσότερες πληροφορίες. Για να μπορείτε να χρησιμοποιήσετε κι εδώ το ίδιο σύστημα, δημιουργήστε ένα καινούριο Arduino Sketch και αντιγράψτε τον παρακάτω κώδικα (Εναλλακτικά κατεβάστε τον από http://ardx.org/RGBMB)

```
/*
Set_RGB_Color.pde
Shows how to use HTML-style hex triplet color codes to specify RGB LED colors
Written for SparkFun Arduino Inventor's Kit CIRC-RGB
*/

// LED leads connected to PWM pins
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;

// HTML-style hex triplet color code values from Wikipedia
const unsigned long ORANGE = 0xFF7F00;
const unsigned long TEAL = 0x008080;
const unsigned long AUBERGINE = 0x614051;

// Length of time we spend showing each color
const int DISPLAY_TIME = 2000; // In milliseconds
```

```
19
20  void setup() {
21  // No setup required.
22  }
23
24  void loop() {
25  // Cycle through our awesome colors
26
27  setColor(ORANGE);
28  delay(DISPLAY_TIME);
29
30  setColor(TEAL);
31  delay(DISPLAY_TIME);
32
33  setColor(AUBERGINE);
34  delay(DISPLAY_TIME);
35  }
36
37  void setColor(unsigned long color) {
38  /*
39  Sets the color of the RGB LED to the color specified by the
40  HTML-style hex triplet color value supplied to the function.
41
42  The color value supplied should be an unsigned long number of the form:
43
44  0xRRGGBB
45
46  e.g. 0xFF7F00 is orange
47
48  where:
49   0x specifies the value is a hexadecimal number
50   RR is a byte specifying the red intensity
51   GG is a byte specifying the green intensity
52   BB is a byte specifying the blue intensity
53
54   How it works (you don't need to understand this to use the function):
55
56   The supplied value of the form 0xRRGGBB is an unsigned long which can
57   store four bytes. If we view the number in bit form the bits of the
58   unsigned long look like this:
59
60   iiiiiiiirrrrrrrrggggggggbbbbbbbb
61
62  where:
63
64   i are bits that are ignored (because we only need three bytes for the value)
65   r are bits specifying the red intensity
66   g are bits specifying the green intensity
67   b are bits specifying the blue intensity
68
69   We use bit shifting and masking to extract the individual values.
70
71   Bit shifting moves the bits in a number along in one direction to produce
72   a new number. For example ">> n" means shift the bits n positions to the
73   right.
74
75   e.g. iiiiiiiirrrrrrrrggggggggbbbbbbbb >> 8 gives:
76
77   00000000iiiiiiiirrrrrrrrgggggggg
78
79   The number 0xFF represents the bit mask:
80
```

```
81  00000000000000000000000011111111
82
83  The "&" character means the two values are "and"ed together--for each bit
84  in the two numbers being "and"ed together the resulting bit is only set in
85  the result if the same bits are set in both the input numbers.
86
87  So, continuing our example:
88
89  00000000iiiiiiiirrrrrrrrggggggggg && 0xFF gives:
90
91  00000000iiiiiiiirrrrrrrrggggggggg &&
92  00000000000000000000000011111111 =
93  000000000000000000000000ggggggggg
94
95 And we now have extracted only the green bits from our hex triplet.
96
97 More details on bit math can be found here:
98 <http://www.arduino.cc/playground/Code/BitMath>
99
100 */
101
102 // Extract the intensity values from the hex triplet
103 byte redIntensity = (color >> 16) & 0xFF;
104 byte greenIntensity = (color >> 8) & 0xFF;
105 byte blueIntensity = color & 0xFF;
106
107 // Display the requested color
108  analogWrite(RED_LED_PIN, redIntensity);
109  analogWrite(GREEN_LED_PIN, greenIntensity);
110  analogWrite(BLUE_LED_PIN, blueIntensity);
111 }
```

Εκτός από τα παραπάνω, χρησιμοποιήστε τη συνάρτηση δημιουργίας τυχαίων αριθμών για την παραγωγή τυχαίων χρωμάτων. Ορίστε μια νέα μεταβλητή: long randNumber; Αρχικοποιήστε την συνάρτηση randomSeed(analogRead(0)); και λάβετε μια τυχαία τιμή: randNumber = random(1300000); την οποία θα την εμφανίσετε με setColor();

Στη συνέχεια αποθηκεύστε το ως CIRC_12_c1 και φορτώστε το στο Arduino για να επιβεβαιώσετε την ορθή λειτουργία του. Εφόσον λειτουργεί σωστά μπορείτε να πειραματιστείτε δίνοντας τις δικές σας τιμές για τα χρώματα με έναν, ίσως, πιο γνώριμο για εσάς τρόπο.

Β΄.36 13η εργαστηριακή άσκηση

Στην παρούσα εργαστηριακή άσκηση θα χρησιμοποιηθεί ένας αισθητήρας κάμψης σαν συσκευή εισόδου. Ο αισθητήρας κάμψης χρησιμοποιεί άνθρακα πάνω σε μία λωρίδα πλαστικό, για να λειτουργεί σαν ποτενσιόμετρο, μόνο που τώρα η αντίσταση δεν μεταβάλλεται γυρίζοντας κάποιον διακόπτη αλλά λυγίζοντας τον αισθητήρα. Χρησιμοποιείται ξανά διαιρέτης τάσης (όπως στα εργαστήρια 8 και 9) προκειμένου να ανιχνευθεί η αλλαγή στην αντίσταση από το Arduino. Ο αισθητήρας λυγίζει προς μια κατεύθυνση (η 'ριγέ' πλευρά να

βρίσκεται στο εξωτερικό της καμπύλης) και όσο περισσότερο λυγίζει, τόσο μεγαλώνει η αντίσταση. Η αντίσταση κυμαίνεται από 10 KOhm μέχρι 35 KOhm. Στο κύκλωμά μας θα χρησιμοποιήσουμε τον αισθητήρα κάμψης για να ελέγξουμε την κίνηση ενός Servo.

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 25 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

- 1 3pin Header
- 1 Mini Servo
- 1 Αντίσταση 10 kOhm (Καφέ – Μαύρο – Καφέ)
- 1 Αισθητήρας κάμψης
- 8 καλώδια

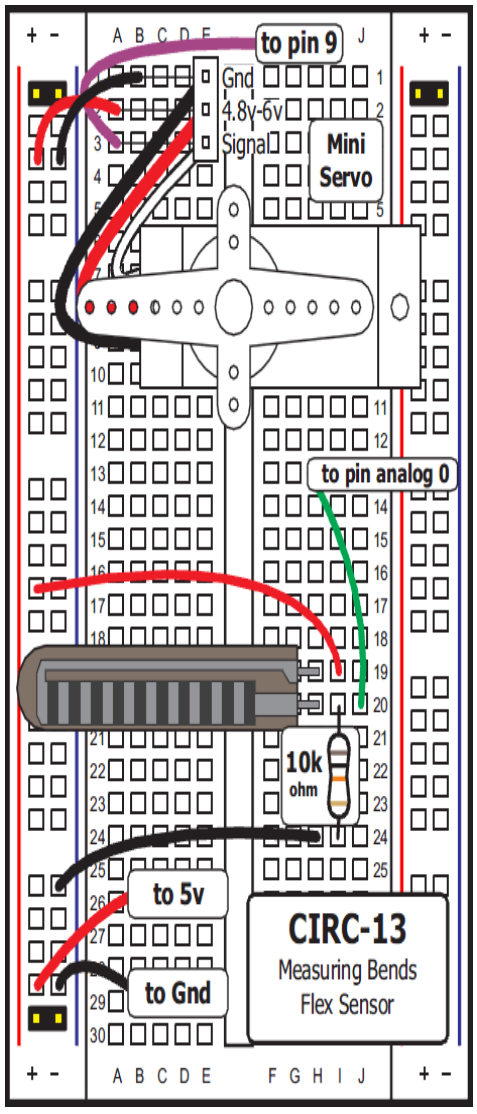
Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.32. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.31.

Τοποθετήστε το 3pin Header όπως φαίνεται στο Σχήμα Β΄.31. Συνδέστε κάθε καλώδιο του Servo με το 3pin Header και στη συνέχεια με τρία καλώδια συνδέστε το άσπρο καλώδιο του Servo με το pin 9 του Arduino, το κόκκινο καλώδιο στην τάση +5V, ενώ το μαύρο στη γείωση. Τοποθετήστε στη συνέχεια τον αισθητήρα κάμψης στο breadboard και συνδέστε το ένα άκρο του στα +5V και το άλλο σε μία αντίσταση 10 kOhm το άλλο άκρο της οποίας πηγαίνει στη γείωση, καθώς και με το αναλογικό pin A0 του Arduino. Τέλος συνδέστε τα +5V και Gnd στο Arduino. **ΠΡΟΣΟΧΗ1**: Ο αισθητήρας κάμψης πρέπει να λυγίζει από τη μέση προς την άκρη, όχι στο σημείο που είναι οι συνδετήρες. **ΠΡΟΣΟΧΗ2**: Ο αισθητήρας κάμψης πρέπει να λυγίζει ως 90 μοίρες όχι περισσότερο. Επίσης, να αποφεύγονται οι παραπάνω από μια κάμψεις.

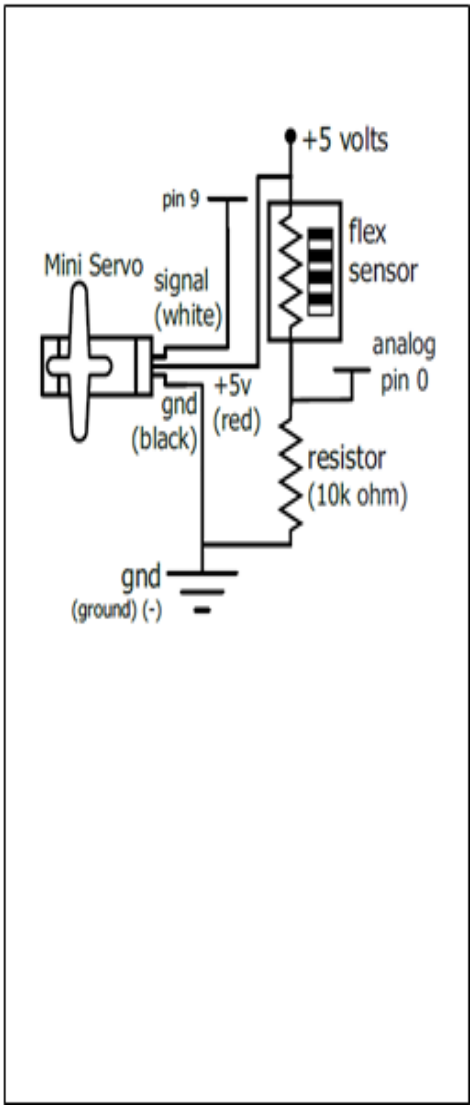
Β΄.37 Προγραμματισμός του κυκλώματος CIRC-13

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE13S.

```
1
2  //Based on File>Examples>Servo>Knob
3  //Controlling a servo position using a potentiometer (variable resistor)
4  //by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>
5
6  #include <Servo.h>
7
8  Servo myservo; //create servo object to control a servo
9
10 int potpin = 0; // analog pin used to connect the potentiometer
```



Σχήμα Β΄.31: Προτεινόμενη υλοποίηση του CIRC13



Σχήμα Β΄.32: Σχηματικό Διάγραμμα του CIRC13

```
11 int val; // variable to read the value from the analog pin
12
13 void setup()
14 {
15  Serial.begin(9600);
16  myservo.attach(9);//attaches the servo on pin 9 to the servo object
17 }
18
19 void loop()
20
21 {
22  val=analogRead(potpin);
23 //reads the value of the potentiometer (value between 0 and 1023)
24  Serial.println(val);
25  val=map(val,50,300,0,179);
26 //scale it to use it with the servo (value between 0 and 180)
27  myservo.write(val);
28 //sets the servo position according to the scaled value
29  delay(15);//waits for the servo to get there
30 }
```

Αποθηκεύστε το ως CIRC_13 και στη συνέχεια φορτώστε το πρόγραμμα στο Arduino. Σε περίπτωση που το Servo δεν κινείται ελέγξτε μήπως έχετε συνδέσει κάτι λάθος. Αν το Servo δεν κινείται όπως θα περιμένατε ίσως λυγίζετε τον αισθητήρα κάμψης προς τη λάθος πλευρά (ενδέχεται να λειτουργεί μόνο προς μια κατεύθυνση). Δοκιμάστε να τον λυγίσετε με τη "ριγέ" πλευρά να βρίσκεται στο εξωτερικό της καμπύλης. Τέλος, αν το Servo κινείται μόνο μια φορά θα πρέπει να βαθμονομηθεί σωστά (θα εξηγηθεί παρακάτω).

Β΄.38 Παραμετροποίηση του κυκλώματος CIRC-13

Βαθμονόμηση (Calibration): Ανεξάρτητα απ' το αν το Servo κινείται σωστά ή όχι, πιθανότατα το εύρος τιμών που παίρνουμε από τον αισθητήρα είναι ανακριβές. Για να προσαρμόσουμε το εύρος στις σωστές του τιμές θα ακολουθήσουμε την εξής διαδικασία:

1. Ανοίγουμε το μενού Εργαλεία>Σειριακή οθόνη (Ctrl+Shift+M).
2. Παρατηρούμε την τιμή που μας επιστρέφει ο αισθητήρας άκαμπτος.
3. Αλλάζουμε την τιμή fromLow στη συνάρτηση map(value, fromLow, fromHigh, toLow, toHigh) από 50 στην τιμή που βρήκαμε στο βήμα 2. Αν η τιμή που βρέθηκε στο βήμα 2 είναι 50 δε χρειάζεται να προβούμε σε καμία αλλαγή.
4. Επιστρέφουμε στη σειριακή οθόνη.
5. Λυγίζουμε εντελώς τον αισθητήρα (90 μοίρες) και σημειώνουμε την τιμή που μας επιστρέφεται.

6. Αλλάζουμε την τιμή της fromHigh στη συνάρτηση map(value, fromLow, fromHigh, toLow, toHigh) από 300 στην τιμή που βρήκαμε στο βήμα 5. Αν η τιμή που βρέθηκε στο βήμα 5 είναι 300 δε χρειάζεται να προβούμε σε καμία αλλαγή. Αποθηκεύστε το πρόγραμμα ως CIRC_13_c1 και φορτώστε το στο Arduino. Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος.

Ενδιαφέρουσες εφαρμογές: Μπορείτε να δείτε μερικές ενδιαφέρουσες εφαρμογές του αισθητήρα κάμψης στις παρακάτω ιστοσελίδες:

- One player Rock Paper Scissors Glove (http://grathio.com/2010/03/rock_paper_scissors_training_glove/)
- Electronic Plant Brace (http://www.flickr.com/photos/mleak/322577143/in/set-72157601527068224/)

Β΄.39 14η εργαστηριακή άσκηση

Το ποτενσιόμετρο μεμβράνης λειτουργεί ακριβώς όπως ένα κοινό ποτενσιόμετρο, αλλά είναι επίπεδο, λεπτό ευλύγιστο και δεν έχει περιστρεφόμενο διακόπτη. Η αντίσταση του μεταβάλλεται απλά ασκώντας πίεση σε διαφορετικά σημεία της επιφάνειάς του. Η αντίσταση κυμαίνεται από 100 μέχρι 10K Ohm και η τιμή που επιστρέφεται μπορεί να χρησιμοποιηθεί για να υπολογιστεί η θέση που πιέστηκε στην επιφάνειά του. Στο παρόν κύκλωμα θα χρησιμοποιηθεί για να ελέγξουμε τα χρώματα ενός RGB LED.

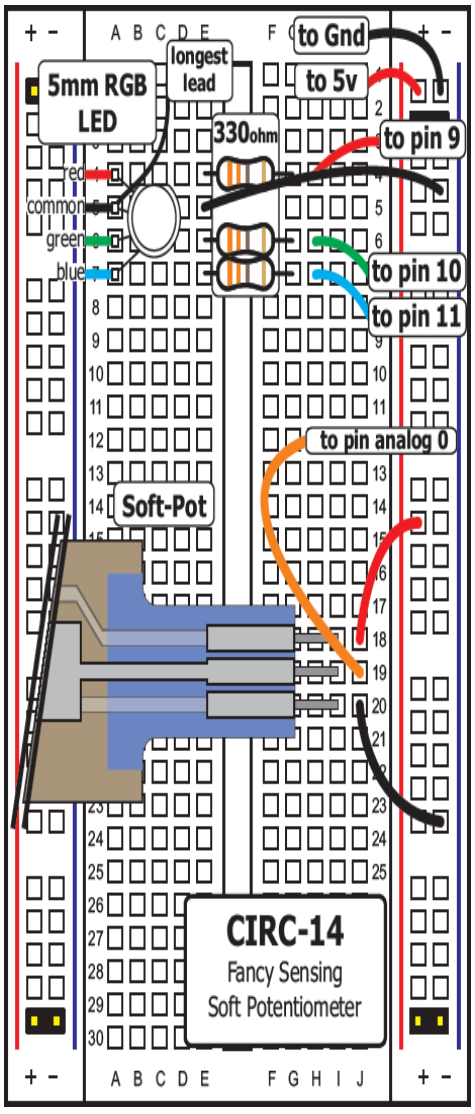
Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 25 λεπτά

Για την εκπόνηση του κυκλώματος CIRC-02 απαιτούνται τα εξής μέρη:

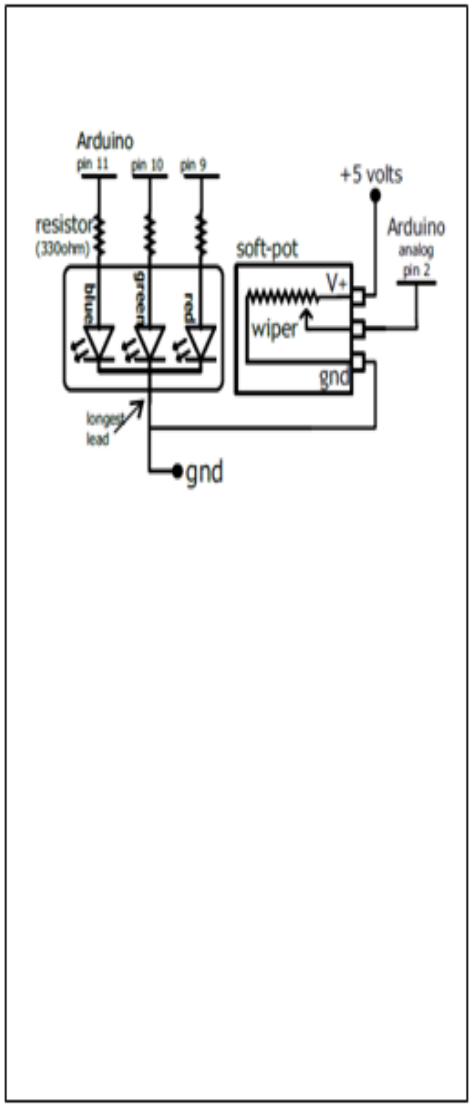
- 1 RGB LED
- 3 αντιστάσεις των 330 Ohm (πορτοκαλί-πορτοκαλί-καφέ)
- 9 καλώδια
- 1 Ποτενσιόμετρο Μεμβράνης

Κατασκευάστε το κύκλωμα, σύμφωνα με τη σχηματική αναπαράσταση του Σχήματος Β΄.34. Μια προτεινόμενη υλοποίηση εικονίζεται στο Σχήμα Β΄.33.

Τοποθετήστε το RGB LED όπως φαίνεται στο Σχήμα Β΄.33. (Τα pins του LED αναλύονται στο Σχήμα Β΄.34). Στη συνέχεια συνδέστε από μία αντίσταση 330 Ohm σε κάθε pin που αντιστοιχεί σε κάποιο χρώμα, και έπειτα κάθε αντίσταση με το ανάλογο pin του Arduino. Επίσης συνδέστε και το pin της γείωσης. Τέλος τοποθετήστε το ποτενσιόμετρο μεμβράνης στο breadboard και συνδέστε το μεσαίο pin στο A0 του Arduino, και τα άλλα δυο στα +5V και τη γείωση αντίστοιχα. ΠΡΟΣΟΧΗ: το ποτενσιόμετρο μεμβράνης ΔΕΝ πρέπει να λυγίζει. Με το δάχτυλο σας το ακουμπάτε σε διάφορα σημεία ελαφρά.



Σχήμα Β΄.33: Προτεινόμενη υλοποίηση του CIRC14



Σχήμα Β΄.34: Σχηματικό Διάγραμμα του CIRC14

Β΄.40 Προγραμματισμός του κυκλώματος CIRC-14

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino. Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE14S.

```
/*       ------------------------------------------------------
 *| Experimentation Kit for Arduino Example Code|
 *| CIRC-14: Fancy Sensing:(Soft Potentiometer) |
         *------------------------------------------------------
 *
 * Will fade an RGB LED from Red-Green-Blue in relation to the
 * soft pot value
 *
 */

// LED leads connected to PWM pins

const int RED_LED_PIN=9; //Red LED Pin
const int GREEN_LED_PIN=10; //Green LED Pin
const int BLUE_LED_PIN=11; //Blue LED Pin

void setup(){
//no need for any code here
}

void loop(){
 int sensorValue=analogRead(0); //read the Soft Pot

 int redValue=constrain(map(sensorValue,0,512,255,0),0,255);
//calculate the red Value (255-0 over the range 0-512)

 int greenValue=constrain(map(sensorValue,0,512,0,255),0,255)\
          -constrain(map(sensorValue,512,1023,0,255),0,255);
//calculate the green value (0-255 over 0-512 & 255-0 over 512-1023)

 int blueValue=constrain(map(sensorValue,512,1023,0,255),0,255);
//calculate the blue value 0-255 over 512-1023

 // Display the requested color

 analogWrite(RED_LED_PIN, redValue);
 analogWrite(GREEN_LED_PIN, greenValue);
 analogWrite(BLUE_LED_PIN, blueValue);
}
```

Αποθηκεύστε το ως CIRC_14 και στη συνέχεια φορτώστε το πρόγραμμα στο Arduino. Σε περίπτωση που το LED δεν ανάβει, ή δείχνει λάθος χρώματα, ελέγξτε αν το κύκλωμά σας είναι σωστό. Αν λαμβάνετε περίεργα αποτελέσματα, ίσως πιέζετε το ποτενσιόμετρο σε περισσότερες από μία θέσεις. Αυτό είναι φυσιολογικό και μπορεί να χρησιμοποιηθεί για την επίτευξη ιδιαίτερων αποτελεσμάτων.

Β΄.41 Παραμετροποίηση του κυκλώματος CIRC-14

Μείωση της υπερβολικής παρουσίας του κόκκινου χρώματος: Ενδέχεται το κόκκινο χρώμα του LED να είναι εντονότερο από τα άλλα και αυτό να έχει ως συνέπεια τα χρώματα που προκύπτουν να μην είναι τόσο ισορροπημένα. Αν συμβαίνει κάτι τέτοιο, μπορείτε να το διορθώσετε είτε βάζοντας μια αντίσταση περισσότερων Ohm στο pin του κόκκινου χρώματος, είτε αλλάζοντας την παρακάτω γραμμή στο πρόγραμμά σας, από: analogWrite(RED_LED_PIN, redIntensity); σε: analogWrite(RED_LED_PIN, redIntensity/3); Αποθηκεύστε (Ctrl+s) και φορτώστε ξανά το πρόγραμμα στο Arduino, για να επιβεβαιώσετε ότι το πρόβλημα λύθηκε.

Χρήση HSB (Hue, Saturation & Brightness) για την απεικόνιση των χρωμάτων: Ένας εναλλακτικός τρόπος απεικόνισης των χρωμάτων πέρα από το RGB είναι το HSB. Επισκεφθείτε το σχετικό λήμμα στη Wikipedia (http://en.wikipedia.org/wiki/HSV_color_space) για περισσότερες πληροφορίες. Για να μπορείτε να χρησιμοποιήσετε κι εδώ το ίδιο σύστημα, δημιουργήστε ένα καινούριο Arduino Sketch και αντιγράψτε τον παρακάτω κώδικα (Εναλλακτικά κατεβάστε τον από http://ardx.org/CODE14MB)

```
1  //LED leads connected to PWM pins
2  const int RED_LED_PIN=9;
3  const int GREEN_LED_PIN=10;
4  const int BLUE_LED_PIN=11;
5
6  //http://www.kasperkamperman.com/blog/arduino/arduino-programming-hsb-to-rgb/
7
8  void getRGB(int hue, int sat, int val, int colors[3]){
9   /*convert hue, saturation and brightness ( HSB/HSV ) to RGB
10    The dim_curve is used only on brightness/value and on saturation
11 (inverted).
12  This looks the most natural.
13  */
14
15 // val=dim_curve[val];
16 // sat=255-dim_curve[255-sat];
17
18  int r;
19  int g;
20  int b;
21  int base;
22
23 if (sat==0){//Acromatic color (gray).Hue doesn't mind.
24  colors[0]=val;
25  colors[1]=val;
26  colors[2]=val;
27 }else{
28
29  base=((255-sat)*val)>>8;
30
31  switch(hue/60) {
32  case 0:
33    r = val;
34    g = (((val-base)*hue)/60)+base;
```

```
35    b = base;
36   break;
37
38   case 1:
39    r = (((val-base)*(60-(hue%60)))/60)+base;
40    g = val;
41    b = base;
42   break;
43
44   case 2:
45    r = base;
46    g = val;
47    b = (((val-base)*(hue%60))/60)+base;
48   break;
49
50   case 3:
51    r = base;
52    g = (((val-base)*(60-(hue%60)))/60)+base;
53    b = val;
54   break;
55
56   case 4:
57    r = (((val-base)*(hue%60))/60)+base;
58    g = base;
59    b = val;
60   break;
61
62   case 5:
63    r = val;
64    g = base;
65    b = (((val-base)*(60-(hue%60)))/60)+base;
66   break;
67  }
68   colors[0]=r;
69   colors[1]=g;
70   colors[2]=b;
71  }
72  }
73
74  int rgb_colors[3];
75  ///  ----------------
76
77  int redIntensity;
78  int greenIntensity;
79  int blueIntensity;
80
81  void setup() {
82
83  }
84
85  void loop() {
86    int sensorValue = analogRead(0);
87    getRGB(map(sensorValue, 0, 1023, 0, 255), 0xff, 0xff, rgb_colors);
88    redIntensity = rgb_colors[0];
89    greenIntensity = rgb_colors[1];
90    blueIntensity = rgb_colors[2];
91
92   // Display the requested color
93    analogWrite(RED_LED_PIN, 255-redIntensity);
94    analogWrite(GREEN_LED_PIN, 255-greenIntensity);
95    analogWrite(BLUE_LED_PIN, 255-blueIntensity);
96
```

```
97 }
```

Στη συνέχεια αποθηκεύστε το ως CIRC_14_c1 και φορτώστε το στο Arduino για να επιβεβαιώσετε την ορθή λειτουργία του.

Προσαρμοσμένα κουμπιά: Το ποτενσιόμετρο μεμβράνης, μπορεί να χρησιμοποιηθεί για να δημιουργήσετε προσαρμοζόμενα κουμπιά. Για να το πετύχετε αυτό θα πρέπει να ορίσετε ζώνες τιμών στις οποίες θα αντιστοιχεί μία συγκεκριμένη λειτουργία. Χρησιμοποιήστε το παρακάτω ενδεικτικό κομμάτι κώδικα και τη Σειριακή οθόνη (Ctrl+Shift+M) (ανατρέξτε στον κώδικα του εργαστηρίου 10 για να θυμηθείτε πως εκτυπώνεται κάτι ως έξοδος στη σειριακή οθόνη) προκειμένου να προσδιορίσετε τις επιθυμητές τιμές για 2 ή περισσότερα κουμπιά.

```
1  if(analogRead(0)>minValue && analogRead(0)<maxValue){
2  analogWrite(RED_LED_PIN, YourValue);
3  analogWrite(GREEN_LED_PIN, YourValue);
4  analogWrite(BLUE_LED_PIN, YourValue); }
```

Τροποποιήστε σχετικά τον κώδικα για δημιουργία 3 ζωνών. Κάθε ζώνη να αντιστοιχηθεί με ένα χρώμα της επιλογής σας. Στη συνέχεια αποθηκεύστε το ως CIRC_14_c2 και φορτώστε το στο Arduino για να επιβεβαιώσετε την ορθή λειτουργία του.

Β΄.42 15η εργαστηριακή άσκηση

Εκτιμώμενος Χρόνος Υλοποίησης Εργαστηρίου: 120 λεπτά
Για την εκπόνηση του κυκλώματος CIRC-15 απαιτούνται τα εξής μέρη:

- 14 καλώδια
- 1 beeper
- 1 οθόνη NOKIA 5110 LCD
- 1 ultrasonic module

Η εργασία χωρίζεται σε δύο φάσεις: στην πρώτη, υλοποιείται το κύκλωμα του συστήματος και στη δεύτερη, γράφεται ο κώδικας που θα πραγματοποιεί τις κατάλληλες ενέργειες. Το τελικό αποτέλεσμα καλείται να εκτελεί τις παρακάτω ενέργειες: το σύστημα μέσω του ultrasonic Module να μετράει την απόσταση και όταν αυτή είναι μικρότερη των σαράντα cm (40cm) να ενεργοποιείται ο βομβητής εκπέμποντας επαναλαμβανόμενο ήχο που θα αυξάνεται ανάλογα με την απόσταση. Παράλληλα, στην οθόνη θα εμφανίζονται τα cm που απομένουν αν η απόσταση είναι μικρότερη των σαράντα cm, αλλιώς θα εμφανίζεται το μήνυμα “Hello Student XXXX (όπου XXX είναι ο αριθμός μητρώου του φοιτητή)”.

Το σύστημα, θα υλοποιηθεί στον μικροελεγκτή Arduino. Οι συσκευές που θα επικοινωνούν με το Arduino, είναι μία οθόνη NOKIA 5110LCD μαζί με έξι καλώδια σύνδεσης, ο αισθητήρας ultrasonic SDM-IO και τέσσερα καλώδια, το beeper με δύο καλώδια και δύο επιπλέον καλώδια σύνδεσης του Pin (5V) και του (GND) από το Arduino στο breadboard. Η συγγραφή του κώδικα και το Compile, μπορούν να επιτευχθούν με το πρόγραμμα Arduino IDE. ΠΡΟΣΟΧΗ!! Το καλώδιο σύνδεσης USB του Arduino με τον Η/Υ πρέπει να συνδεθεί μόνο όταν ο κώδικας δεν έχει λάθη και ΜΟΝΟ ΑΝ η σύνδεση του κυκλώματος είναι σωστή.

Β΄.42.1 Σύνδεση Arduino με breadboard

Αρχικά πρέπει να δοθεί ρεύμα στο breadboard όπου και θα γίνουν όλες οι συνδέσεις. Με ένα καλώδιο συνδέστε την τάση 5V του Arduino με μια υποδοχή με την ένδειξη (+) στο breadboard. Η τάση 5V βρίσκεται στην πλευρά του Arduino που βρίσκονται και οι αναλογικές θύρες (Analog IN). Κάντε το ίδιο και για τη γείωση (GND του Arduino) με το (-).

Β΄.42.2 Σύνδεση του beeper

Ο βομβητής έχει δύο pins. Το (+) συνδέεται με ένα από τα digital pins του Arduino από το 2 έως το 13 και το (-) με τη γείωση. (Α1) Ποια είναι η συνάρτηση καθυστέρησης του arduino και τι παραμέτρους δέχεται; (Β1) Βρείτε τμήμα κώδικα από τον ιστοχώρο Arduino Playground http://arduino.cc ώστε το beeper να ενεργοποιείται και να εκπέμπει ήχο για ένα δευτερόλεπτο. Τροποποιήστε τον κώδικα στη συνάρτηση beep() Στο X (της beeper) αντικαταστήστε τον αριθμό του pin σύνδεσης με το Arduino.

```
1  int beeper = X;
2
3  void setup(){
4      Serial.begin(9600);
5      pinMode(beeper, OUTPUT); //pin is output
6  }
7  void loop() {
8      beep();
9      delay(100);
10 }
11
12 void beep(){
13
14 }
```

Αφού συμπληρώσετε τα τμήματα του κώδικα συνδέστε το καλώδιο και φορτώστε το πρόγραμμα. Βεβαιωθείτε για την ορθή λειτουργία.

Β΄.42.3 Σύνδεση του ultrasonic module

Ο αισθητήρας αυτός έχει τέσσερα pins σύνδεσης. Αυτά συνδέονται με το Arduino με τον παρακάτω τρόπο:

1. VCC ⟶ στην τάση 5V
2. TRIG ⟶ σε ένα από τα ελεύθερα pin του Arduino
3. ECHO ⟶ σε ένα από τα ελεύθερα pin του Arduino
4. GND ⟶ GND (Arduino)

(A2) Βρείτε τι κάνει κάθε ένα από τα τέσσερα pins του ultrasonic module. Συμπληρώστε τον κώδικα ορίζοντας τα εξής:

```
#define SDM_IO_TIMEOUT 1000
int TrigPin = XX;
```

(B2) Συμπληρώστε τον αριθμό TrigPin.

```
int EchoPin = XX;
```

(B3) Συμπληρώστε τον αριθμό EchoPin.

```
unsigned long ultrasoundDuration;
int timeout;
unsigned long tStartPing = 0;
float sensorValue = 0;
```

Στην setup() συμπληρώστε επίσης pinMode(XXXXPin, OUTPUT); (B4) Συμπληρώστε το σωστό Pin pinMode(XXXXPin, INPUT); (B5) Συμπληρώστε το σωστό Pin Στην loop() συμπληρώστε το εξής: sensorValue = read_sdm_io_range();

Δίνεται ακόμα η παρακάτω ημιτελής συνάρτηση

```
//SDM-IO Ultrasonic Range Sensor distance function
float read_sdm_io_range(){
 unsigned char pin = 0;
 unsigned int time_flag = 0;

 digitalWrite(TrigPin, HIGH);
 delayMicroseconds(2);
 digitalWrite(TrigPin, LOW);
 delayMicroseconds(10);
 digitalWrite(TrigPin, HIGH);

 tStartPing = micros();
 timeout = 0;
 pin = digitalRead(EchoPin);
 while(pin) {
  pin = digitalRead(EchoPin);
  time_flag++;
  if(time_flag>SDM_IO_TIMEOUT){
```

```
19    timeout = 1;
20    break;
21   }
22  }
23   ultrasoundDuration=micros()-tStartPing;
```

(Β6) Εκτυπώστε στη θυριακή οθόνη σε δεκαδική μορφή το μήνυμα «ultrasoundDuration us,» και αφήστε μια νέα γραμμή αν πολλαπλασιάσουμε με 0,017 το ultrasoundDuration το αποτέλεσμα μετατρέπεται σε cm. Κάντε το και εκτυπώστε το και αυτό στη θυριακή οθόνη.

```
1   if (timeout)
2    return 999;
3   else
4    return ultrasoundDuration*0.017; //result in cm
5  }
```

Αφού συμπληρώσετε τα τμήματα του κώδικα συνδέστε το καλώδιο και φορτώστε το πρόγραμμα. Βεβαιωθείτε για την ορθή λειτουργία.

Σύνδεση της οθόνης: Η οθόνη έχει οχτώ pins σύνδεσης. Αυτά συνδέονται με το Arduino με τον παρακάτω τρόπο:

1. VCC $\longrightarrow$ στην τάση 3.3V (Arduino)
2. GND $\longrightarrow$ (Κενό)
3. SCE $\longrightarrow$ σε ένα από τα ελεύθερα pin του Arduino
4. RST $\longrightarrow$ σε ένα από τα ελεύθερα pin του Arduino
5. D/C $\longrightarrow$ σε ένα από τα ελεύθερα pin του Arduino
6. DN(MOSI) $\longrightarrow$ σε ένα από τα ελεύθερα pin του Arduino
7. SCLK $\longrightarrow$ σε ένα από τα ελεύθερα pin του Arduino
8. LED $\longrightarrow$ (Κενό)

(Α3) Βρείτε τι κάνει κάθε ένα από τα οχτώ pins. Συμπληρώστε τον κώδικα ορίζοντας τα εξής:

```
1  //The pins to use on the arduino
2  #define PIN_SCE XX
3  #define PIN_RESET XX
4  #define PIN_DC XX
5  #define PIN_SDIN XX
6  #define PIN_SCLK XX
```

Όπου XX οι αριθμοί των Pin που συνδέσατε στο Arduino. ΠΡΟΣΟΧΗ!! Υπενθυμίζουμε πως στα Pin 0 και 1 δεν πρέπει να συνδεθεί τίποτα. Στην setup() συμπληρώστε τις LcdInitialise();, LcdClear();.

Από τη διεύθυνση (http://blog.stuartlewis.com/2011/02/12/scrolling-text-with-an-arduino-and-nokia-5110-screen/) θα βρείτε τους ορισμούς των παραπάνω συναρτήσεων και άλλων που είναι απαραίτητοι για να εκτυπωθεί μήνυμα στη οθόνη. (Β7) Βρείτε και τοποθετείστε στον κώδικα ότι είναι απαραίτητο για να γίνει εκτύπωση και στη συνάρτηση loop() εμφανίστε το μήνυμα "Hello Student XXXX (αριθμός μητρώου)" Αφού συμπληρώσετε τα τμήματα του κώδικα συνδέστε το καλώδιο και φορτώστε το πρόγραμμα. Βεβαιωθείτε για την ορθή λειτουργία.

Σύνδεση ultrasonic + beeper: Συνδέστε μαζί στο Arduino τον αισθητήρα ultrasonic και ρυθμίστε τη συνάρτηση beep() να ενεργοποιεί το beeper όταν η απόσταση είναι μικρότερη από 40 cm. (Α4) Εξηγείστε πως ο αισθητήρας ultrasonic μετρά την απόσταση. (Β8) Υλοποιείστε τον κώδικα που πραγματοποιεί την παραπάνω διαδικασία.

Σύνδεση ultrasonic + LCD: Συνδέστε μαζί στο Arduino τον αισθητήρα ultrasonic και την οθόνη LCD και Τροποποιήστε τον κώδικα ώστε στην οθόνη να εμφανίζεται η απόσταση σε cm όταν αυτή είναι μικρότερη των 40 cm, αλλιώς να εμφανίζεται το μήνυμα "Hello Student XXXX (αριθμός μητρώου)" (Β9) Υλοποιείστε τον κώδικα που πραγματοποιεί την παραπάνω διαδικασία. (Α5) Εξηγείστε τον τρόπο με τον οποίο εμφανίζονται τα γράμματα και οι χαρακτήρες στην οθόνη.

Σύνδεση όλων των περιφερειακών: Αφού όλα τα υλικά έχουν συνδεθεί, μένει να τα κάνουμε να αλληλεπιδρούν. (Β10) Στη συνάρτηση loop() ορίστε το εξής: Αν η επιστρεφόμενη τιμή είναι μικρότερη από 40 cm να εκτυπώνεται στην οθόνη η επιστρεφόμενη τιμή και να ενεργοποιείται το beeper.. Αλλιώς το μήνυμα "Hello Student XXXX (αριθμός μητρώου)" και το beeper να σταματάει. (Β11) Επεξεργαστείτε τη συνάρτηση beep(), ώστε ο ήχος που εκπέμπεται να επαναλαμβάνεται γρηγορότερα καθώς τα cm πέφτουν σε 30, 20 και 10.