LEDs - Λυχνίες

Θεωρητικό μέρος

1.1 Εισαγωγή

Σκοπός μας είναι να ασχοληθούμε με λυχνίες (LEDs) χρησιμοποιώντας το Arduino, να φτιάξουμε απλά προγράμματα για να ελέγξουμε τη λειτουργία τους.

LED *(**L**ight **E**mitting **D**iode)* ή **Δίοδος Εκπομπής Φωτός** αποκαλείται ένας ημιαγωγός, ο οποίος εκπέμπει φωτεινή ακτινοβολία στενού φάσματος όταν του παρέχεται μία ηλεκτρική τάση κατά τη φορά ορθής πόλωσης (forward-biased). Περισσότερες πληροφορίες εδώ.

Οι διαφορετικές λυχνίες LED (μεγέθους και χρώματος) απαιτούν συνήθως διαφορετικές αντιστάσεις. Τα LEDs που έρχονται μαζί με τα περισσότερα starter kit είναι **5mm** και απαιτούν μια αντίσταση περίπου 330 Ohm για να εξασφαλίσουν το σωστό είδος φωτεινότητας.

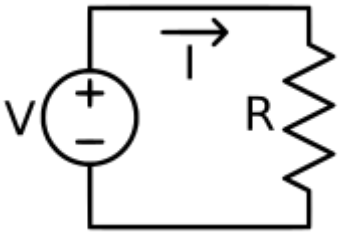
Οι προδιαγραφές για κάθε λυχνία LED καθορίζουν ακριβώς το μέγιστο και ελάχιστο ρεύμα καθώς και την τάση που απαιτείται για την τροφοδοσία της. Για να υπολογίσουμε την αντίσταση που χρειάζεται το LED αρκεί να εφαρμόσουμε το νόμο του Ohm.

Νόμος του Ohm: το ρεύμα το οποίο διαρρέει έναν αντιστάτη είναι ανάλογο της τάσης που εφαρμόζεται στα άκρα του και αντιστρόφως ανάλογο της τιμής της αντίστασης.

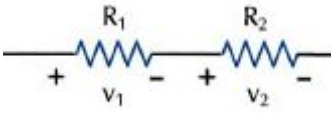
$$I = V / R$$

όπου

R είναι η τιμή της αντίστασης σε μονάδες Ohm (Ω).
V είναι η διαφορά δυναμικού στα άκρα της αντίστασης, σε μονάδες Volt (V).
I είναι το ρεύμα το οποίο διαρρέει την αντίσταση μετρημένο σε Ampère (A).



Επίσης χρήσιμο είναι να θυμηθούμε τη σύνδεση αντιστατών σε σειρά και παράλληλα:
Δύο αντιστάτες λέμε ότι συνδέονται σε σειρά όταν διαρρέονται από το ίδιο ρεύμα.



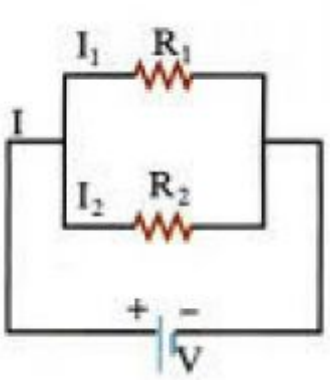
Η τιμή της αντίστασης του ισοδύναμου αντιστάτη δίνεται από την εξίσωση

$$R_{o\lambda} = R_1 + R_2$$

και

$$V = V_1 + V_2$$

Δύο αντιστάτες λέμε ότι συνδέονται παράλληλα όταν έχουν ίδια τάση ή πιο απλά όταν οι δύο αντιστάτες έχουν κοινά άκρα.



Ο ισοδύναμος αντιστάτης έχει αντίσταση

$$\frac{1}{R_{o\lambda}} = \frac{1}{R_1} + \frac{1}{R_2}$$

ή

$$R_{o\lambda} = \frac{R_1 \cdot R_2}{R_1 + R_2}$$

Στην παράλληλη σύνδεση το ηλεκτρικό ρεύμα διακλαδίζεται:

$$I = I_1 + I_2$$

Τα «πακέτα» LEDs του εμπορίου συνήθως έρχονται με τις σωστές αντιστάσεις τους.

Για την υλοποίηση των εργασιών θα χρειαστείτε:

- ✓ Arduino kit
- ✓ LEDs
- ✓ Αντιστάσεις 270 - 470 Ohm Αντίσταση (παθητική) ¼ watt. 220Ω ή 330Ω είναι εντάξει.
- ✓ Καλώδια σύνδεσης

Σημείωση: οι παρακάτω λέξεις που θα συναντήσετε στο κείμενο χρησιμοποιούνται με τον ίδιο τρόπο:

- ακίδες, ακροδέκτες, pins
- πρόγραμμα, κώδικας, sketch

1.2 Arduino IDE - Ρύθμιση θύρας

Αν έχετε ήδη τρέξει πρόγραμμα στο περιβάλλον Arduino IDE του υπολογιστή σας μπορείτε να παραλείψετε αυτό το βήμα.

Ό,τι χρειάζεστε για την διαχείριση του Arduino από τον υπολογιστή σας το παρέχει το Arduino IDE, την τελευταία έκδοση του οποίου μπορείτε να κατεβάσετε από το επίσημο site εδώ (για λειτουργικό σύστημα Linux, Mac και Windows).

ΜΕΤΑ την εγκατάσταση του λογισμικού συνδέστε την πλακέτα του Arduino με τον Η/Υ σας μέσω του συνοδευτικού καλωδίου USB (τύπου AB).

Με το Arduino συνδεδεμένο, πρέπει να βεβαιωθείτε ότι το λογισμικό Arduino μπορεί να «βλέπει» την πλακέτα, δηλαδή να υπάρχει σωστή επικοινωνία μεταξύ Η/Υ και πλακέτας Arduino.

Ακολουθήστε τα παρακάτω τρία βήματα:

1. Πατήστε στο μενού ***Εργαλεία*** → ***Πλακέτα: "Arduino/Genuino Uno"*** και επιλέξτε την έκδοση του Arduino που διαθέτετε (π.χ. Arduino Uno).
2. Πατήστε πάλι στο μενού ***Εργαλεία*** → ***Θύρα*** και επιλέξτε την **σωστή θύρα επικοινωνίας** του Η/Υ με την πλακέτα του Arduino. Πρέπει να βεβαιωθείτε ότι το λογισμικό γνωρίζει σε ποια θύρα έχει συνδεθεί το Arduino. Εάν υπάρχουν περισσότερες από μία θύρες COM, το Arduino σας είναι κανονικά συνδεδεμένο σε αυτή με το μεγαλύτερο αριθμό - οι άλλες θύρες θα χρησιμοποιούνται πιθανώς για ένα ποντίκι ή ένα πληκτρολόγιο ανάλογα με το ποια περιφερειακά έχετε συνδέσει.

1.3 Πρόγραμμα Blink

Η πλακέτα Arduino διαθέτει ένα ενσωματωμένο LED στη θύρα 13 που μπορούμε να χρησιμοποιήσουμε.

Ας ξεκινήσουμε εκτελώντας το πρόγραμμα Blink που υπάρχει στα έτοιμα παραδείγματα του περιβάλλοντος Arduino IDE. Το πρόγραμμα Blink δεν απαιτεί άλλα εξαρτήματα εκτός από την πλακέτα Arduino και τρέχοντάς το επιβεβαιώνουμε ότι η πλακέτα λειτουργεί σωστά.

Το πρόγραμμα αυτό είναι βοηθητικό να χρησιμοποιείται όποτε έχετε αμφιβολία αν λειτουργεί ή όχι η πλακέτα και η σύνδεσή της με τον υπολογιστή.

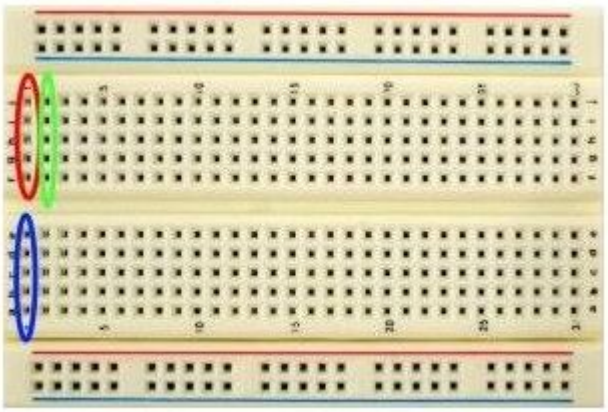
Κάθε πρόγραμμα στο Arduino λέγεται αλλιώς **Sketch**.

Για να τρέξει το πρόγραμμα Blink κάνουμε τα ακόλουθα:

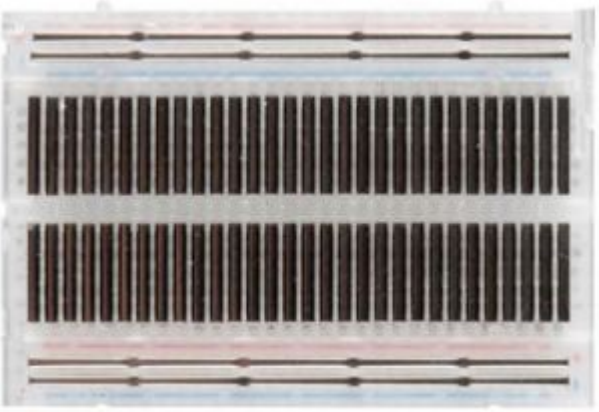
1. Επιλέγουμε από το μενού ***Αρχείο*** → ***Παραδείγματα*** → ***01.Basics*** και μετά ***Blink.*** Το πρόγραμμα εμφανίζεται στην οθόνη μας.
2. Επιλέγουμε  ώστε να γίνει μεταγλώττιση του προγράμματος.
3. Επιλέγουμε  ώστε το πρόγραμμα να ανέβει στην πλακέτα του Arduino.
4. Λογικά στην πλακέτα θα δούμε το ενσωματωμένο LED της θύρας 13 να αναβοσβήνει με περίοδο 2 δευτερολέπτων.

Αξίζει να σημειωθεί ότι όταν έχει φορτωθεί ένα πρόγραμμα στην πλακέτα, αυτό θα εκτελείται αυτόματα μόλις το Arduino τροφοδοτηθεί με ρεύμα. Δηλαδή, κάποια άλλη στιγμή που θα συνδέσουμε το Arduino με τον υπολογιστή θα δούμε ότι τρέχει το τελευταίο πρόγραμμα που φορτώσαμε στην πλακέτα.

1.4 Πλακέτα δοκιμών - breadboard



Εικόνα 1α

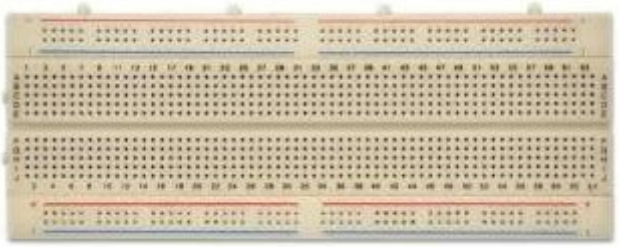


Εικόνα 1β

Η πλακέτα δοκιμών ή αλλιώς breadboard έχει τρύπες που είναι συνδεδεμένες ανά ομάδες και χρησιμοποιείται για να φτιάξουμε ηλεκτρονικά κυκλώματα χωρίς κολλήσεις.

Στο παραπάνω breadboard (Εικόνα 1α) υπάρχουν οι ακόλουθες ομάδες ξεκινώντας από πάνω: η πρώτη οριζόντια σειρά, η 2η οριζόντια σειρά, 30 κάθετες πεντάδες, πιο κάτω άλλες 30 κάθετες πεντάδες, μία οριζόντια σειρά κάτω και άλλη μία οριζόντια σειρά κάτω. Οι ομάδες φαίνονται στην εικόνα 1β. Η έννοια της ομάδας είναι ότι αν βάλουμε κάτι σε μία τρύπα και κάτι άλλο σε άλλη τρύπα της ίδια ομάδας είναι σαν να τα έχουμε συνδέσει σε σειρά.

Αν έχουμε πιο μεγάλο breadboard κάθε οριζόντια σειρά χωρίζεται και σε δύο ομάδες (Εικόνα 2).

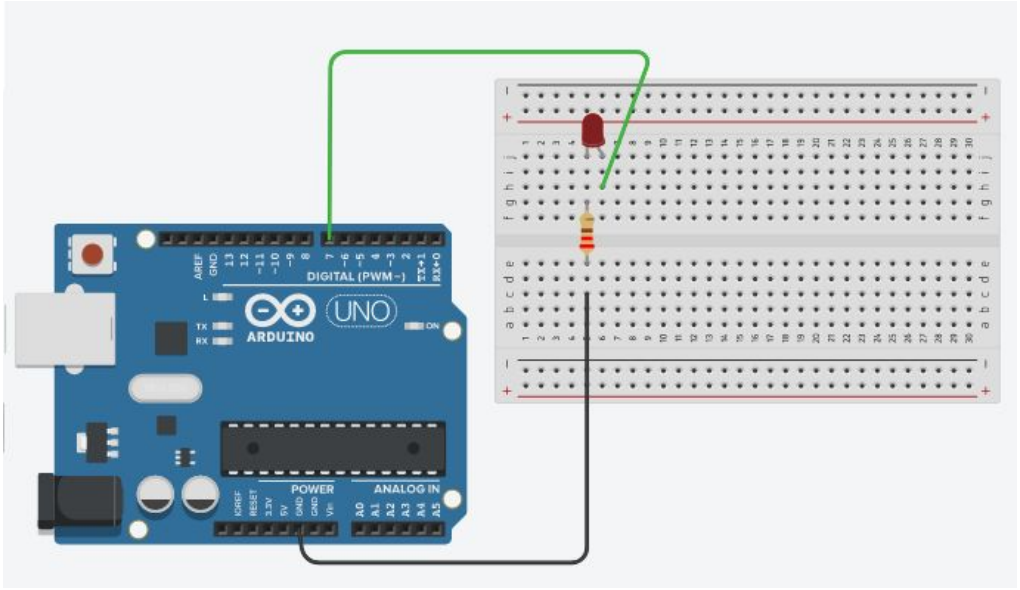


Εικόνα 2

Είναι καλή τακτική, κυρίως σε πιο πολύπλοκα κυκλώματα να χρησιμοποιούμε:
-την οριζόντια σειρά με το + για να συνδέσουμε κάτι σε 5V στο Arduino (καλύτερα με **κόκκινο** καλώδιο) και
-την οριζόντια σειρά με το - για σύνδεση σε GND - γείωση (επιθυμητό με **μαύρο** καλώδιο).

1.5 Εφαρμογές με ένα LED

1.5.1 Κύκλωμα με ένα LED



Εικόνα 3

Μία από τις ακίδες του LED είναι μεγαλύτερη από την άλλη. Αυτή είναι η άνοδος ή το θετικό ποδαράκι. Θα συνδέσουμε αυτήν την ακίδα στην πλακέτα του Arduino στην θύρα 7 (πράσινο καλώδιο στην Εικόνα 3). Οι λυχνίες LEDs πρέπει να τοποθετούνται πάντα με την σωστή πολικότητα, αλλιώς δεν λειτουργούν. (Συνήθως δεν καταστρέφονται αν συνδεθούν με αντίστροφη πολικότητα σε ένα κύκλωμα - η μόνη παρενέργειά τους είναι ότι δεν εκπέμπουν φως, δηλαδή ΔΕΝ ΑΝΑΒΟΥΝ).

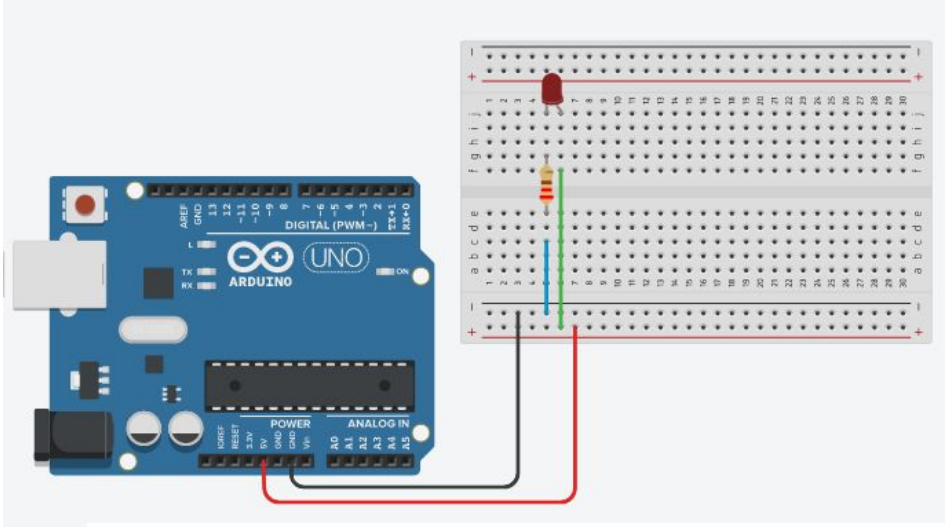
Το άλλο άκρο του LED (αρνητικό ποδαράκι - κάθοδος) θα συνδεθεί με την αντίσταση και αυτή με τη γείωση GND του Arduino (στο Arduino uno υπάρχουν 3 θύρες για γείωση - GND).

Έχοντας ολοκληρώσει το κύκλωμα συνδέουμε το συνοδευτικό καλώδιο USB στο Arduino και το άλλο άκρο του σε μια υποδοχή USB στον υπολογιστή μας.

Σημείωση: Είναι ισοδύναμο η αντίσταση να μη συνδεθεί στο μικρό, αλλά στο μεγάλο ποδαράκι του pin, δηλαδή:

μακρύ ποδαράκι - αντίσταση - pin	και	**μικρό ποδαράκι-γείωση**	ή αλλιώς
μακρύ ποδαράκι - pin	και	**μικρό ποδαράκι-αντίσταση-γείωση**	

Το παρακάτω κύκλωμα χρησιμοποιηθεί τις οριζόντιες γραμμές στο breadboard για γείωση και 5V και αν υλοποιηθεί έχει σαν αποτέλεσμα να μένει μόνιμα αναμμένο το LED.

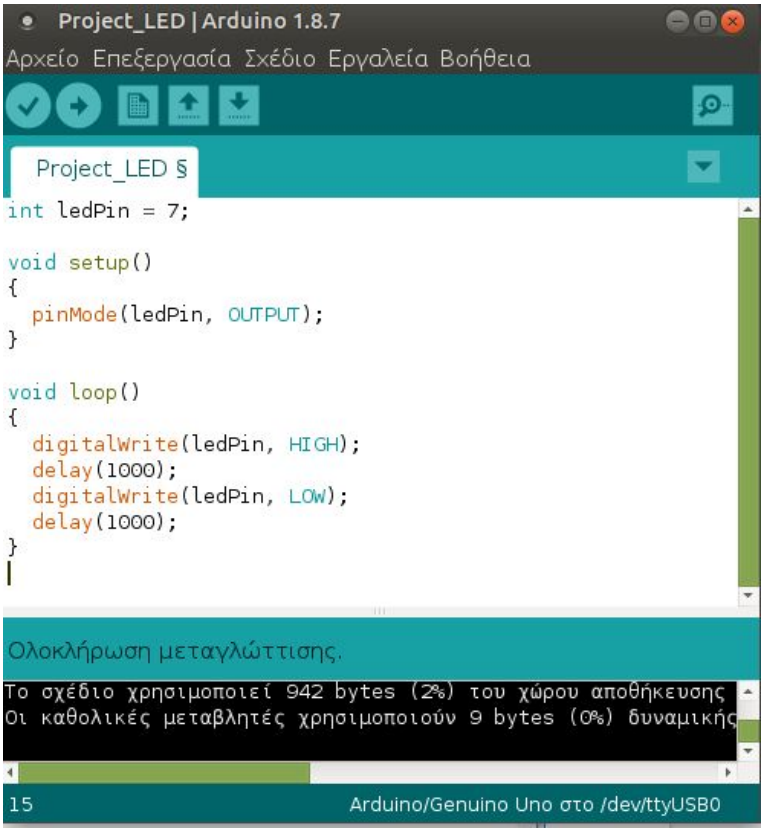


Εικόνα 4

1.5.2 Προγραμματίζοντας το Arduino να αναβοσβήνει ένα LED

Ας σβήσουμε τον κώδικα Blink και ας επικολλήσουμε τον παρακάτω κώδικα μέσα στο παράθυρο του IDE.

```
int ledPin = 7;

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

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



Πατώντας το κουμπί  γίνεται μεταγλώττιση του προγράμματος. Αν υπάρχουν λάθη εμφανίζονται σχετικά μηνύματα στο κάτω μέρος του παραθύρου. Όταν εμφανιστεί μήνυμα ότι η μεταγλώττιση ολοκληρώθηκε επιλέγουμε το  για να ανεβάσουμε (upload) τον κώδικα στην πλακέτα. Η λυχνία LED πρέπει να ανάψει. Εάν δεν ανάψει κάπου έχουμε κάνει λάθος (στην καλωδίωση, στο πρόγραμμα, έχει καεί το LED...).

Κατανοώντας τον κώδικα

Η διαδικασία **setup()** εκτελείται μία φορά όταν φορτωθεί το πρόγραμμα στην πλακέτα.
Η εντολή **pinMode**(ledPin, OUTPUT); ορίζει τον **ακροδέκτη 7** του Arduino σα θύρα **εξόδου** (**OUTPUT**).

Η διαδικασία **loop()**, όπως υποδηλώνει και το όνομά της, εκτελείται συνεχώς το τμήμα κώδικα που περιέχει. Όταν το τσιπ Arduino εκτελέσει όλες τις οδηγίες στη διαδικασία **loop**, επιστρέφει στην πρώτη εντολή του βρόχου και ξεκινά πάλι.

Η εντολή **digitalWrite**(ledPin, HIGH); δίνει στο pin 7 υψηλή τάση (**HIGH**) ώστε να ανάψει το LED.

Η εντολή **digitalWrite**(ledPin, LOW); δίνει στο pin 7 χαμηλή τάση (LOW) οπότε το LED σβήνει.

Η εντολή **delay**(1000) προκαλεί καθυστέρηση 1 δευτερολέπτου. Ο αριθμός στις αγκύλες είναι ο αριθμός των χιλιοστών του δευτερολέπτου που θέλουμε να περιμένει το Arduino πριν εκτελέσει την επόμενη εντολή.

1.5.3 Προγραμματίζοντας το Arduino να αναβοσβήνει ένα LED συγκεκριμένες φορές

Σε αυτό το παράδειγμα το LED αναβοσβήνει μόνο 6 φορές.

Η μεταβλητή **d** (global variable **d**) παρακολουθεί πόσες φορές το LED έχει ενεργοποιηθεί και απενεργοποιηθεί. Αρχικά έχει τιμή 0.

Στο βρόγχο (loop) η μεταβλητή d αυξάνεται κατά 1 σε κάθε επανάληψη. Γίνεται έλεγχος και εάν δεν έχει υπερβεί το 6, η ενδεικτική λυχνία συνεχίζει να αναβοσβήνει με τον ρυθμό που ορίζουν οι δηλώσεις **delay**. Μόλις η μεταβλητή ξεπεράσει την οριζόμενη τιμή του 6 το πρόγραμμα τερματίζεται.

```
int ledPin = 7;
long d = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  d++;
  if (d<=6)   {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
  }
}
```

1.5.4 Προγραμματίζοντας το Arduino ώστε να μεταβάλλεται η φωτεινότητα ενός LED

Χρησιμοποιώντας την εντολή digitalWrite(ledPin, HIGH); ανάβουμε το αντίστοιχο LED.

Πώς θα μπορούσαμε όμως στο Arduino να αλλάξουμε τη **φωτεινότητα** ενός LED; Θα μας χρειαζόταν αντί ψηφιακή έξοδος αναλογική. Στην πραγματικότητα το Arduino δεν έχει την δυνατότητα να μας δώσει αναλογική έξοδο. Για να το πετύχουμε χρησιμοποιούμε το τρικ της «*διαμόρφωσης πλάτους παλμού*».

Διαμόρφωση πλάτους παλμού (PWM) σημαίνει ότι ποικίλλει ο χρόνος που το HIGH και LOW αποστέλλονται μέσω του ακροδέκτη-pin και κάθε φορά εμείς παίρνουμε το μέσο όρο αυτής της τάσης. Μικρότερη τάση στα άκρα του LED σημαίνει και μικρότερη φωτεινότητά του και το αντίθετο.

Οι **έξοδοι που κατασκευαστικά μπορούν να υποστηρίξουν PWM είναι οι 3, 5, 6, 9, 10, και 11.**

Τα παραπάνω υλοποιούνται με τη συνάρτηση analogWrite(). Π.χ. Η analogWrite στέλνει ένα σήμα-παλμό (PWM) σε ένα pin πετυχαίνοντας την γρήγορη εναλλαγή μεταξύ HIGH/LOW. Το led αναβοσβήνει πολύ γρήγορα δίνοντας μας έτσι την αίσθηση ότι "ξεθωριάζει" χάνοντας την φωτεινότητα του. Έτσι το συγκεκριμένο ψηφιακό pin είναι σε κατάσταση ψευδοαναλογικής εξόδου (PWM).

analogWrite(led, brightness); Η μεταβλητή brightness μπορεί να πάρει τιμές από **0 ως 255.**

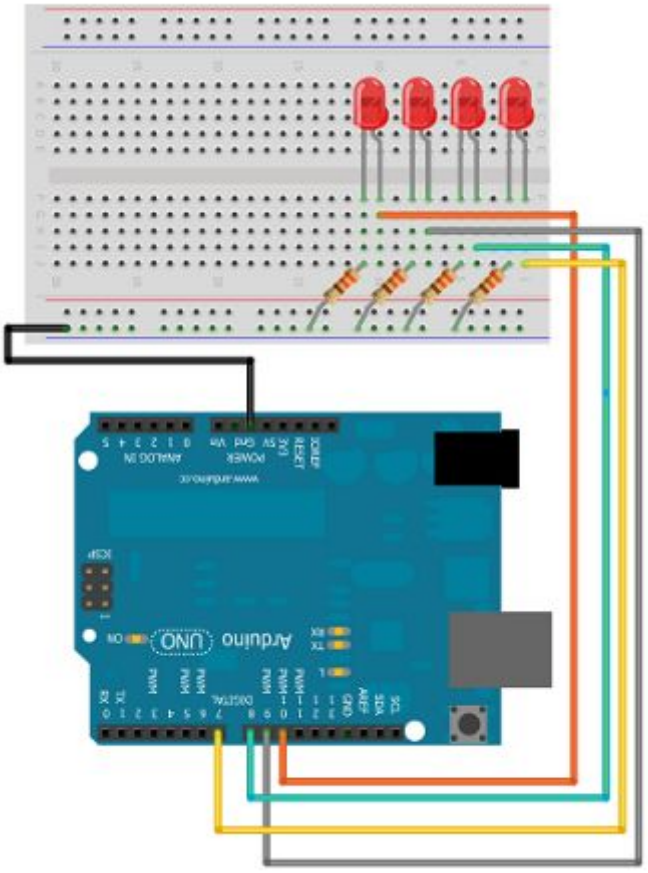
Περισσότερες πληροφορίες για τη μεταβολή της φωτεινότητας μπορείτε να βρείτε εδώ.

```
    int led = 3;
int brightness = 0;
int fadeAmount = 5;

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

void loop()  {
  analogWrite(led, brightness);
  brightness = brightness + fadeAmount;

  if (brightness == 0 || brightness ==
255) {
    fadeAmount = -fadeAmount ;
  }
   delay(30);
}
```

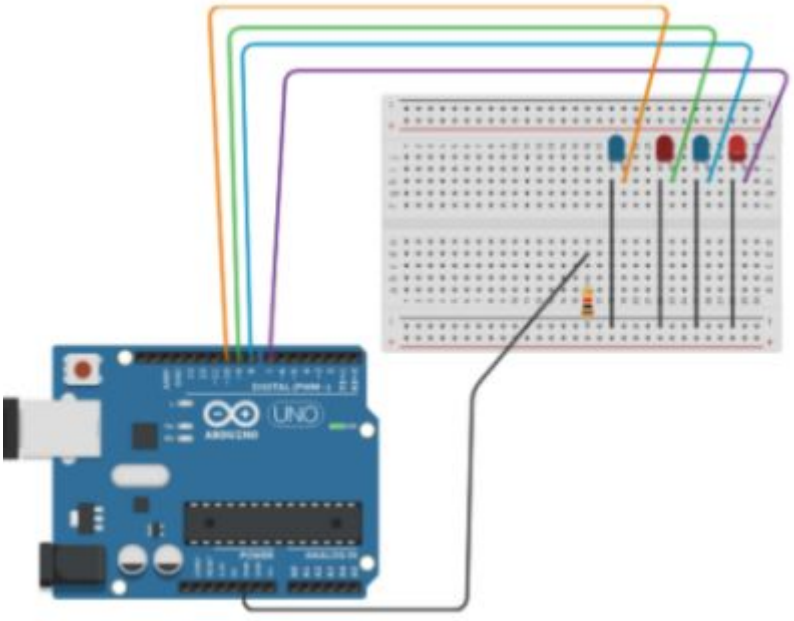
1.6 Εφαρμογές με τέσσερα LEDs

1.6.1 Κύκλωμα με τέσσερα LEDs

Ας φτιάξουμε κύκλωμα με 4 LEDs - με τις αντίστοιχες αντιστάσεις τους - το ένα LED δίπλα στο άλλο σε σειρά. Η κατασκευή μας θα πρέπει να μοιάζει με την παρακάτω εικόνα:



Σε ειδικές μόνο περιπτώσεις μπορούμε να κάνουμε οικονομία στις αντιστάσεις όταν έχουμε πολλά LEDs. Τέτοια περίπτωση είναι όταν κάθε φορά ανάβει μόνο ένα LED, όπου θα μπορούσαμε να χρησιμοποιήσουμε μία μόνο αντίσταση, όπως φαίνεται στο παρακάτω κύκλωμα:



Υπενθύμιση: το μακρύτερο ποδαράκι σε ένα LED συνδέεται στο pin του Arduino και το μικρότερο στη **γείωση (GND).**

1.6.2 Προγραμματίζοντας το Arduino για έλεγχο των συνδέσεων με 4 LEDs

Επειδή το κύκλωμα απέκτησε κάποια σχετική πολυπλοκότητα θα ήταν χρήσιμο πριν κάνουμε οτιδήποτε να ελέγξουμε αν αυτό είναι καλωδιωμένο σωστά. Κάτι τέτοιο θα μας γλίτωνε από αρκετό ψάξιμο και κόπο αργότερα.

Το παρακάτω πρόγραμμα ελέγχει ότι όλες οι συνδέσεις είναι σωστές.

```
int ledPin[] = {7,8,9,10};

void setup() {
  for (int i =0;i<4;i++)   {
    pinMode(ledPin[i], OUTPUT);
  }
}

void loop() {
  for (int i =0;i<4;i++)   {
    digitalWrite(ledPin[i], HIGH);
  }
}
```

Στο πρόγραμμα αυτό ορίζουμε αρχικά έναν πίνακα για να αποθηκεύσουμε τη λίστα των pins Arduino που έχουμε συνδέσει με τις λυχνίες LED.

Στη συνάρτηση setup() χρησιμοποιούμε ένα βρόχο (loop) για να ορίσουμε οτι καθένα από τα pins είναι **εξόδου**.

Στη συνάρτηση loop() χρησιμοποιούμε έναν άλλο βρόχο (loop) για να γυρίσουμε όλα τα pins σε **HIGH** (on).

Αν αντιγράψουμε και εκτελέσουμε αυτόν τον κώδικα μπορούμε να ελέγξουμε αν όλα τα LEDs ανάβουν.

Στη συνέχεια, αλλάζοντας το HIGH σε LOW, ελέγχουμε αν τα LEDs σβήνουν.

Εάν κάποια από τις λυχνίες LED δούμε ότι δεν ανάβει, ελέγχουμε τα ακόλουθα:

1. είναι το θετικό (μακρύτερο ποδαράκι) του LED συνδεδεμένο με καλώδιο στο κατάλληλο pin του Arduino;
2. είναι το μικρότερο ποδαράκι του LED συνδεδεμένο με αντίσταση που καταλήγει στην οριζόντια ομάδα στο breadboard, από όπου ξεκινά καλώδιο για το GND του Arduino;
3. έχουμε χρησιμοποιήσει σωστές αντιστάσεις (π.χ. 220Ω, 330Ω);
4. μήπως κάποιο καλώδιο δεν κάνει επαφή;
5. μήπως να αλλάξουμε pin στο Arduino;
6. μήπως να χρησιμοποιήσουμε άλλα καλώδια, ή άλλο led;

1.6.3 Προγραμματίζοντας το Arduino να παρουσιάζει δυαδικούς αριθμούς με 4 LEDs

Για να μπορέσουμε να κατανοήσουμε αυτό το βήμα θα πρέπει να έχουμε κάποιες θεωρητικές γνώσεις σχετικά με τους δυαδικούς αριθμούς.

Στον παρακάτω πίνακα παριστάνονται οι πρώτοι 16 δυαδικοί αριθμοί καθώς και η αντιστοιχία τους με το δεκαδικό αριθμητικό σύστημα:

2^3	2^2	2^1	2^0	
Binary digit **3**	Binary digit **2**	Binary digit **1**	Binary digit **0**	Decimal
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**

1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**10**
1	0	1	1	**11**
1	1	0	0	**12**
1	1	0	1	**13**
1	1	1	0	**14**
1	1	1	1	**15**

Περισσότερες πληροφορίες σχετικά με τους δυαδικούς αριθμούς και τις μετατροπές τους μπορείτε να βρείτε εδώ.

Θα χρησιμοποιήσουμε τα ενδεικτικά LEDs με την εξής λογική:
Όταν το LED είναι αναμμένο (on) θεωρούμε ότι συμβολίζει το λογικό: **1**
Όταν το LED είναι σβηστό (off) θεωρούμε ότι συμβολίζει το λογικό: **0**
Με αυτή την παραδοχή θα μπορούσαμε με το κατάλληλο πρόγραμμα-κώδικα να αναπαραστήσουμε σύμφωνα με τον παραπάνω πίνακα αριθμούς από το 0 έως το 15 (*Αιτιολογήστε γιατί μόνον αυτούς;*)

Ας κάνουμε τις αντιστοιχίες ανάμεσα σε leds, pins και δυαδικά ψηφία

Λυχνία στο breadboard	Pin (ακροδέκτης) στο Arduino	Δυαδικό ψηφίο
1ο led από δεξιά	7	**Binary digit 0** (μονάδες)
2ο led από δεξιά	8	**Binary digit 1**
3ο led από δεξιά	9	**Binary digit 2**
4ο led από δεξιά	10	**Binary digit 3**

Δηλαδή αν στα τέσσερα LEDs δούμε: σβηστό, αναμμένο, αναμμένο, σβηστό, αυτό θα σημαίνεο 0110, δηλαδή τον αριθμό 6.

Στον παρακάτω κώδικα, έχουμε μια νέα διαδικασία (displayBinary) για την εμφάνιση του δυαδικού αριθμού που θέλουμε να δείξουμε.
Οι αριθμοί στη γλώσσα προγραμματισμού αποθηκεύονται στην πραγματικότητα σε δυαδική μορφή ακόμη και όταν ορίζουμε τις τιμές τους χρησιμοποιώντας decimal. Στον κώδικα αυτό, ένας βρόχος χρησιμοποιείται για να διαβάσει τις τιμές των πρώτων τεσσάρων bits του δεκαδικού αριθμού που αποστέλλεται στη διαδικασία. Η συνάρτηση **bitRead** { if (bitRead(numToShow, i)==1)} «λέει» αν υπάρχει 1 ή 0 σε μια δεδομένη τιμή θέσης. Εάν βρεθεί ένα 1, τότε ο ακροδέκτης-pin του Arduino μεταβαίνει σε κατάσταση HIGH δηλαδή ανάβει η λυχνία LED που είναι συνδεδεμένη στο αντίστοιχο pin, αν βρεθεί 0 τότε σβήνει η λυχνία LED που είναι συνδεδεμένη σε αυτό το pin για τη θέση αυτή.

Ας δοκιμάσουμε αυτόν τον κώδικα με όλους τους διαφορετικούς δεκαδικούς αριθμούς από το 0 έως και το 15.

```
int ledPin[] = {7,8,9,10};

void setup() {
  for (int i =0;i<4;i++)   {
    pinMode(ledPin[i], OUTPUT);
  }
}

void loop() {
  displayBinary(2);
}
```

```
void displayBinary(byte numToShow) {
  for (int i =0;i<4;i++)   {
    if (bitRead(numToShow, i)==1) {
      digitalWrite(ledPin[i], HIGH);
    }
    else {
      digitalWrite(ledPin[i], LOW);
    }
  }
}
```

1.6.2 Προγραμματίζοντας το Arduino να μετράει προς τα πάνω με 4 LEDs

Θα χρησιμοποιήσουμε τη διαδικασία **displayBinary** που γράψαμε παραπάνω για να μετρήσει προς τα πάνω (από το 0 προς το 15) στο δυαδικό σύστημα.

Η χρονοκαθυστέρηση μεταξύ της εμφάνισης των αριθμών (600 ms) μας επιτρέπει να προλάβουμε να δούμε με σαφήνεια πως έγινε η αλλαγή για να εμφανισθεί ο επόμενος αριθμός στην ακολουθία.

```
    int ledPin[] = {7,8,9,10};

void setup() {
  for (int i =0;i<4;i++)   {
    pinMode(ledPin[i], OUTPUT);
  }
}

void loop() {
  for (byte counter =0;counter<=15; counter++) {
    displayBinary(counter);
    delay(600);
  }
}

void displayBinary(byte numToShow) {
  for (int i =0;i<4;i++)   {
    if (bitRead(numToShow, i)==1)     {
      digitalWrite(ledPin[i], HIGH);
    }
    else {
      digitalWrite(ledPin[i], LOW);
    }
  }
}
```

Με ανάλογο τρόπο μπορούμε να φτιάξουμε πρόγραμμα που να μετράει προς τα κάτω, δηλαδή να ξεκινάει από το 15 και να φτάνει στο 0.

1.6.4 Προγραμματίζοντας το Arduino - Μοτίβα με 4 LEDs

Ας φτιάξουμε αριθμητικά μοτίβα χρησιμοποιώντας τη διαδικασία που εμφανίζει αριθμούς από το 0 έως το 15 σε δυαδική μορφή.

Θα ορίσουμε σε έναν πίνακα ποιοι αριθμοί θέλουμε να εμφανίζονται με τη σειρά και θα ανάβουν τα αντίστοιχα τέσσερα LEDs.

Το αποτέλεσμα είναι να φαίνεται το φως να αναπηδά από το ένα LED στο άλλο.

```
    int ledPin[] = {7,8,9,10};

void setup() {
  for (int i =0;i<4;i++) {
    pinMode(ledPin[i], OUTPUT);
  }
}

void loop() {
  byte nums[] = {0, 1, 3, 6, 4, 12, 8, 12, 4, 6, 3, 1, 0};
  for (byte i = 0; i<13;i++)   {
    displayBinary(nums[i]);
    delay(25);
  }
}

void displayBinary(byte numToShow) {
  for (int i =0;i<4;i++)   {
    if (bitRead(numToShow, i)==1) {
      digitalWrite(ledPin[i], HIGH);
    }
    else {
      digitalWrite(ledPin[i], LOW);
    }
  }
}
```

Δραστηριότητες

1η δραστηριότητα: Διαδοχικό άναμμα των LEDs στις μονές και ζυγές θέσεις

Χρησιμοποιώντας κύκλωμα με 4 LEDs, γράψτε τον κώδικα προγράμματος που να επαναλαμβάνει συνεχώς τα ακόλουθα: πρώτα θα ανάβουν **ΟΛΑ ΜΑΖΙ τα LEDs που βρίσκονται στις ΜΟΝΕΣ** θέσεις με τα υπόλοιπα σβηστά και μετά θα ανάβουν όλα μαζί **ΟΛΑ ΜΑΖΙ τα LEDs που είναι στις ΖΥΓΕΣ** θέσεις με τα υπόλοιπα σβηστά.

2η δραστηριότητα: Άναμμα 6 λυχνιών LEDs με διαδοχική σειρά

Φτιάξτε κύκλωμα με 6 LEDs στη σειρά (χρησιμοποιήστε όσα διαφορετικά χρώματα έχετε).

Γράψτε κώδικα ώστε να ανάβει κάθε LED με τη σειρά του. (Δηλαδή το ένα μετά το άλλο, σαν να αναπηδάει το φως από το ένα LED στο διπλανό του και πάλι από τη αρχή).

Στο πρόγραμμα θα πρέπει να χρησιμοποιηθεί ένας πίνακας 6 θέσεων που να περιέχει τα ανάλογα 6 pins που έχετε χρησιμοποιήσει για να συνδέσετε τα LEDs με το Arduino.

3η δραστηριότητα: Ενδεικτική λυχνία LED με μεταβλητή φωτεινότητα

Σε κύκλωμα με ένα LED γράψτε πρόγραμμα που επαναλαμβάνει τα ακόλουθα:

Το LED αρχικά είναι αναμμένο. Σταδιακά μειώνεται η φωτεινότητα του μέχρι που σβήνει τελείως. Μένει σβηστό για 3 δευτερόλεπτα και στη συνέχεια κατευθείαν (και όχι σταδιακά) παίρνει τη μέγιστη φωτεινότητά του, τη μειώνει σταδιακά κτλ.

Η μεταβολή της φωτεινότητας θα πρέπει να γίνεται με βήμα 3.

4η δραστηριότητα: Οπτικός Κώδικας Mors με LED

Χρησιμοποιώντας κύκλωμα με ένα LED, γράψτε πρόγραμμα που να υλοποιεί τον οπτικό κώδικα Morse για τα 3 πρώτα γράμματα από το επώνυμό σας. Θα χρησιμοποιήσετε ένα συνδυασμό μακρών και σύντομων καθυστερήσεων ώστε να κάνετε το LED να αναβοσβήσει το επώνυμό σας.

Βάλτε μια μεγαλύτερη καθυστέρηση στο τέλος του μηνύματος και στη συνέχεια αφήστε τον κώδικα να επαναλαμβάνει το επώνυμό σας ξανά και ξανά.

Παρακάτω βλέπετε την αντιστοιχία κάθε γράμματος με τον κώδικα Morse.

Γράμμα	Κώδικας Morse
Α	·—
Β	—···
Γ	——·
Δ	—··
Ε	·
Ζ	——··
Η	····
Θ	—·—·
Ι	··
Κ	—·—
Λ	·—··
Μ	——

Γράμμα	Κώδικας Morse
Ν	—·
Ξ	—··—
Ο	———
Π	·——·
Ρ	·—·
Σ	···
Τ	—
Υ	—·——
Φ	··—·
Χ	————
Ψ	··—
Ω	·——

Βήματα και παραδοχές για την υλοποίηση της δραστηριότητας

✓ Τελεία (**dot)** θεωρούμε το «*μικρό χρόνο*» που παραμένει αναμμένο το LED. Ορίστε το χρόνο που θα αφήσετε αναμμένη την ενδεικτική λυχνία LED για την τελεία (**dot)**. Αυτό είναι το μήκος μίας τελείας (**dot length**).

✓ Παύλα (**dash**) θεωρούμε τον «*μεγάλο χρόνο*» που παραμένει αναμμένο το LED. Το μήκος της παύλας (**dash length**) πρέπει να είναι 3 φορές το μήκος της τελείας.

✓ Ανάμεσα σε κάθε τελεία ή παύλα πρέπει να υπάρχει ένα διάκενο για να ξεχωρίζουμε το ένα στοιχείο από το άλλο (δηλαδή διάκριση μεταξύ τελείας και παύλας) με ένα μήκος χρόνου (όπου η ενδεικτική λυχνία θα είναι σβηστή).

✓ Ανάμεσα σε κάθε πλήρες γράμμα, (δηλαδή διάκριση μεταξύ γραμμάτων) πρέπει να υπάρχει κενό ίσο με το μήκος μιας παύλας (όπου η ενδεικτική λυχνία θα είναι σβηστή).

✓ Μεταξύ κάθε πλήρους λέξης, θα πρέπει να υπάρχει ένα κενό ίσο με το 7 φορές το μήκος της μιας τελείας (όπου η ενδεικτική λυχνία θα είναι σβηστή).

Για το πρόγραμμα αυτό πρέπει να χρησιμοποιηθούν συναρτήσεις.

Η παρακάτω συνάρτηση dot υλοποιεί την τελεία.

```
void dot() {
  digitalWrite(pin, HIGH);
  delay(400); // 400 θα μπορούσε να είναι ο χρόνος για την τελεία.
  digitalWrite(pin, LOW);
  delay(400);
}
```

Όμοια θα φτιάξετε τη συνάρτηση dash() για την παύλα, καθώς και συναρτήσεις για τα γράμματα που σας ενδιαφέρουν. Π.χ. κάποιος που λέγεται Χ. ΑΡΝΤΟΥΙΝΟΠΟΥΛΟΣ πρέπει να υλοποιήσει συναρτήσεις για το Α, το Ρ και το Ν.

Π.χ. η συνάρτηση για το Ω θα ήταν:

```
void omega () {
  dot();
  dash();
  dash();
}
```