# LCD και 8 segment display

*«βλέποντας το ρεύμα»*

# Θ Ε Ω Ρ Ι Α

## Εισαγωγή

Όλοι πλέον έχουμε κατανοήσει το τι ακριβώς είναι το Arduino και το πόσα πολλά πράγματα και εφαρμογές μπορούμε να υλοποιήσουμε με αυτή την σχετικά μικρή πλακέτα μικροελεγκτή.

Ένα όμως από τα παλιότερα προβλήματα στο χώρο των ηλεκτρονικών και της πληροφορικής παραμένει ακόμα και αυτό δεν είναι άλλο από την επικοινωνία Ανθρώπου – Μηχανής. Γιατί ναι μεν η κάθε κατασκευή μπορεί να κάνει σχεδόν τα πάντα, ανάλογα με τον προγραμματισμό του, αλλά ο άνθρωπος αδυνατεί να «δει» το ηλεκτρικό ρεύμα, το μόνο που βλέπει είναι τα αποτελέσματά του. Πολλές δε άλλες φορές υπάρχει η μέγιστη ανάγκη της διάδρασης μεταξύ ανθρώπου-μηχανής για να μπορέσει να πάρει τις ζητούμενες πληροφορίες, ή τελικά και απαραίτητες αποφάσεις.

Τον ρόλο αυτό λοιπόν, δηλαδή να μπορεί να μας δίνει «εικόνα» μία μηχανή (στην περίπτωσή μας ο μικροελεγκτής) το τι συμβαίνει στον κόσμο του τον αναλαμβάνουν ειδικές συσκευές οι οποίες είναι κατασκευασμένες έτσι ώστε από την μία πλευρά να μπορούν να «μιλάνε» μέσω ειδικών κυκλωμάτων (interfaces) με τον μικροελεγκτή και από την άλλη μέσω ειδικών διατάξεων (display) να παρουσιάζουν τις πληροφορίες αυτές σε κείμενο ή εικόνα κατανοητό για τον άνθρωπο.

Αναλόγως του τρόπου με τον οποίον παρουσιάζουν τα δεδομένα και τις πληροφορίες έχουν κατηγοριοποιηθεί π.χ. Led Display, LCD, MONITOR κλπ.

Εμείς στην παρούσα φάση θα ασχοληθούμε με δυο κατηγορίες από τις παραπάνω:

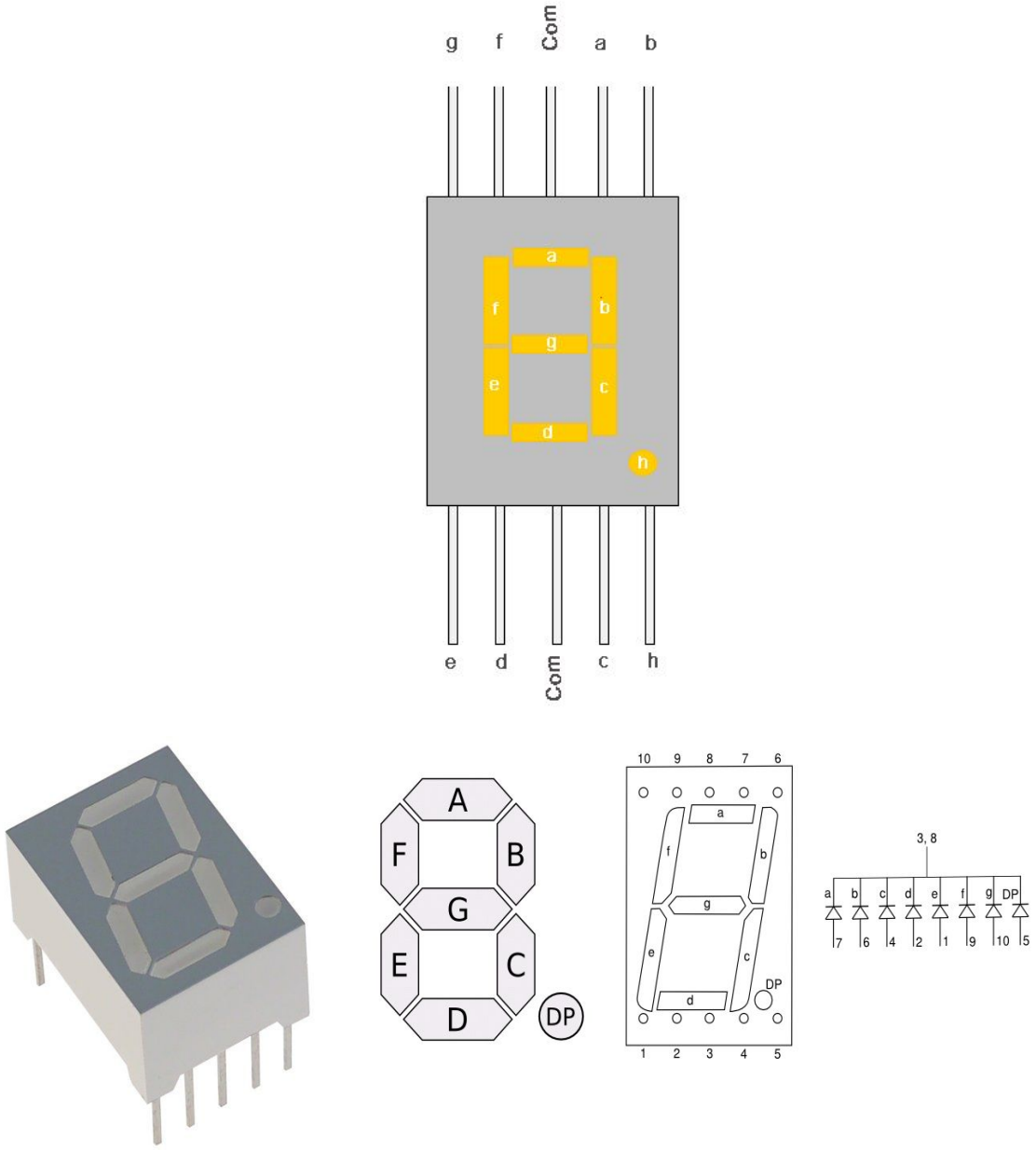
Α. Με τα **LED Display** και την υποκατηγορία τους τα **dot Matrix LED Display** και

Β. Με τις οθόνες υγρών κρυστάλλων τις κοινώς λεγόμενες **LCD**

## 7-segment-display

Τα 7-segment Display χρησιμοποιούνται ευρέως σε ηλεκτρονικές συσκευές συνήθως για την εμφάνιση αριθμών. Στην παρακάτω εικόνα

βλέπουμε ένα συνηθισμένο 7-segment Led Display κοινής καθόδου και δεξιά το ηλεκτρικό ισοδύναμό του.




Όπως παρατηρείτε η συσκευή διαθέτει 10 ακίδες που είναι διατεταγμένες σε 2 σειρές με 5 ακίδες στο επάνω και 5 το κάτω μέρος της οθόνης. Η μεσαία ακίδα κάθε σειράς είναι το σημείο που συνδέουμε την αρνητική τάση (GND). Σε όποια ακίδα και από τις δύο να συνδέσουμε είναι ακριβώς το ίδιο. Αυτό το **LED DISPLAY** λόγω αυτής της κατασκευής του ονομάζεται **LED display κοινής καθόδου**. Αυτό ουσιαστικά για εμάς σημαίνει ότι υπάρχει μια κοινή κάθοδος, δηλαδή η γείωση. Για να ανάψει ένα τμήμα αυτής της συσκευής αρκεί μόνον να συνδέσουμε με τον θετικό πόλο

οποιαδήποτε από τις ακίδες του. Για να πάψει να ανάβει απλώς «αποσυνδέουμε» αυτή την ακίδα. Αν το δούμε από την πλευρά του προγραμματιστή, αυτό μας λέει ότι αν στείλουμε σε οποιαδήποτε ακίδα λογικό HIGH τότε αυτή θα ανάψει ενώ όταν θέλουμε να την απενεργοποιήσουμε αρκεί να στείλουμε σε αυτή την ακίδα λογικό LOW. (Παρατηρείστε ότι «ανάβει» όταν έχουμε λογικό HIGH και όχι όταν έχουμε λογικό LOW. Αυτό οφείλετε στο ότι έχουμε LED Display κοινής καθόδου, το αντίστροφο συμβαίνει σε LED Display κοινής ανόδου.

Πατήστε εδώ καθώς επίσης και εδώ για να δείτε δύο διαφωτιστικά Video σχετικά με τα 7-segment LED Display και τις λεπτομέρειες αυτών.

## Έλεγχος ενός 7-segment display

Ένα 7-segment display δεν είναι τίποτε άλλο παρά ένα μικρό κύκλωμα με 8 LED. Επτά (7) LED είναι τα κύρια τμήματά του και το 8ο είναι η τελεία-κουκίδα. Έτσι έχουμε 7 ακίδες εισόδου για τις κύριες λυχνίες LED, μία ακίδα εισόδου για την κουκίδα και οι άλλες δύο για κοινή κάθοδο ή άνοδο.

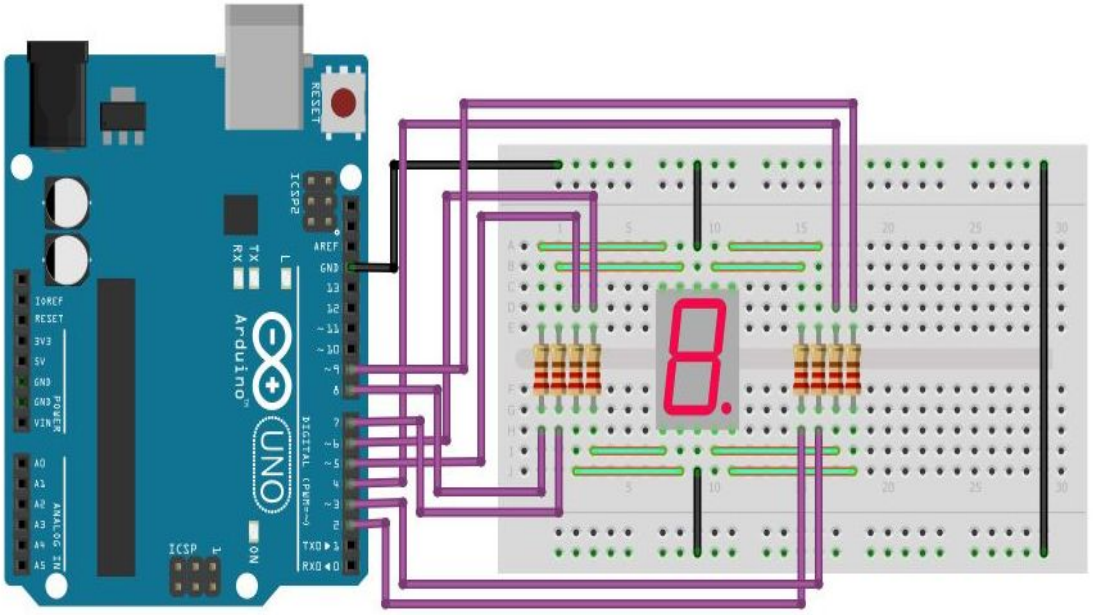
Δεν είναι καθόλου δύσκολο να χρησιμοποιήσουμε μια οθόνη 7 τμημάτων. Χρειάζεται μόνο μερικές λεπτομέρειες που θα πρέπει να γνωρίζουμε. Όπως αναφέραμε υπάρχουν δύο τύποι οθονών - **με κοινή κάθοδο ή κοινή άνοδο**. Αυτή την πληροφορία συνήθως την παίρνουμε από τον κατασκευαστή και συχνά αποτυπώνεται στον κωδικό-ονομασία του προϊόντος. Από τα στοιχεία **SAxxx** αναγνωρίζουμε ότι μια οθόνη είναι με κοινή άνοδο και με τα στοιχεία **SCxxx** για οθόνες με κοινή κάθοδο. (Προσοχή…!!! Δυστυχώς δεν ακολουθούν όλοι οι κατασκευαστές αυτή την τυποποίηση)

## Σύνδεση οθόνης 7 τμημάτων με κοινή κάθοδο χρησιμοποιώντας 8 αντιστάσεις

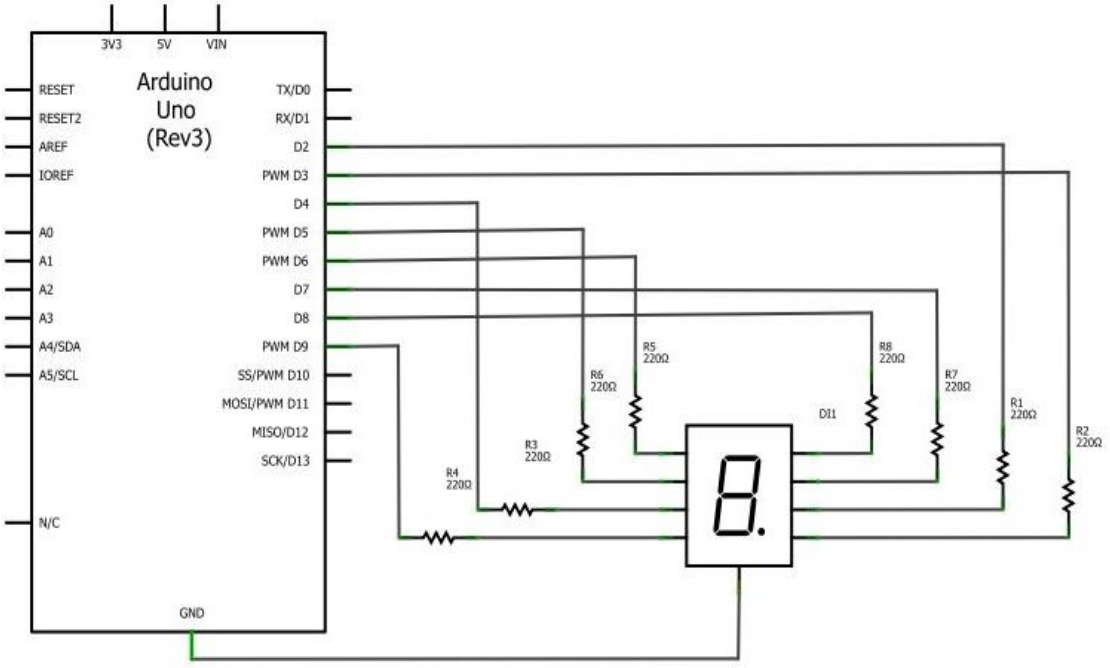
**Απαιτούμενα υλικά:**

- SC56-11GMA (common cathode, green) ή κάποιο αντίστοιχο κοινής καθόδου
- 8x Αντιστάσεις (Περίπου 150-330 ohm).

Τώρα, για να λειτουργήσουμε το κύκλωμά μας χρειαζόμαστε αρκετά καλώδια! Οκτώ (8) καλώδια για τις λυχνίες LED και κάθε λυχνία LED χρειάζεται αντίσταση (όπως ένα LED) για τον **περιορισμό του ρεύματος** και άλλα οκτώ (8) καλώδια επιπλέον για την πλευρά του Arduino και δύο ακόμη για την κοινή άνοδο ή κάθοδο. Με την χρήση όμως ενός motherboard τα πράγματα απλοποιούνται λίγο.

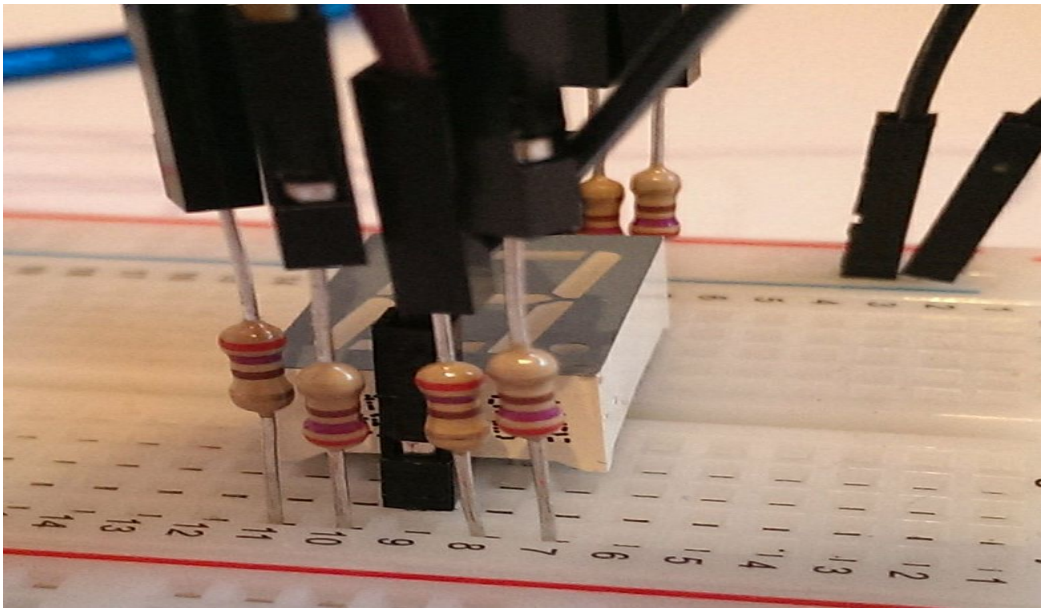


Η επόμενη εικόνα δείχνει το ηλεκτρικό ισοδύναμο της παραπάνω κατασκευής



Κατανοώ ότι τα καλώδια είναι ήδη αρκετά και μπορούν να σας μπερδεύουν για το λόγο αυτό θα σας δείξω ένα μικρό τέχνασμα για το πώς θα τα περιορίσετε αρκετά.

Κανονικά, χρησιμοποιείτε ένα καλώδιο-σύρμα από το Arduino στην αντίσταση στο breadboard. Μια άλλη από την αντίσταση στην οθόνη LED 7 τμημάτων. Πρόκειται για δύο καλώδια (θηλυκά - θηλυκά). Αν έχετε θηλυκά-αρσενικά καλώδια, χρησιμοποιήστε το και συνδέστε την αντίσταση στον αρσενικό ακροδέκτη σύρμα άμεσα. Η άλλη πλευρά του αντιστάτη είναι το θηλυκό τμήμα σύρματος για να συνδεθεί στο breadboard. Κοιτάξτε την παρακάτω εικόνα και θα καταλάβετε



**Προσοχή..!!!**

Η χρήση του 7-segment display χωρίς αντιστάσεις και η σύνδεσή του απευθείας με την πλακέτα του Arduino μπορεί να απλοποιεί σε μεγάλο βαθμό την κατασκευή μας, αλλά εξαρτάται πάντα από τον τύπο της οθόνης που χρησιμοποιούμε, δηλαδή πόσο ρεύμα καταναλώνει έκαστο LED κατά την λειτουργία του και αυτό το ορίζει μόνον ο κατασκευαστής. Πρέπει να είμαστε αρκετά προσεκτικοί διότι μπορεί να οδηγήσουμε την πλακέτα του μικροελεγκτή μας σε **υπερθέρμανση λόγω υπερβολικού συνολικού ρεύματος** και να καταλήξει μοιραίο.

## Σύνδεση οθόνης 7 τμημάτων με κοινή κάθοδο χρησιμοποιώντας 1 αντίσταση

Αντί να βάλουμε 8 αντιστάσεις στα pins 1,2,4,5,6,7,9,10 μπορούμε να βάλουμε μόνο μία αντίσταση (330 Ω είναι μία ασφαλής τιμή αντίστασης) στο pin 3 ή 8 της οθόνης

## Κώδικας δοκιμής 7-segment Display

Ο παρακάτω κώδικας ανάβει με τη σειρά ένα ένα τα segments του Display

```
1  //test 7 segment display
2
3  int pins[] = {2,3,4,5,6,7,8,9};
4
5  void setup () {
6      for(int i=0; i<=7; i++)
7      {
8          pinMode(pins[i], OUTPUT);
9          digitalWrite(i, LOW);
10     }
11 }
12
13 void loop() {
14     for(int i=0; i<=7; i++) {
15         digitalWrite(pins[i], HIGH);
16         delay( 1000);
17         digitalWrite(pins[i], LOW);
18     }
19 }
```

## Ενδεικτικός κώδικας οδήγησης 7-segment Display

```
1  // 7-segment-display
2
3  byte digitArray[11][7] = { { 1,1,1,1,1,1,0 }, // = 0
4                              { 0,1,1,0,0,0,0 }, // = 1
5                              { 1,1,0,1,1,0,1 }, //=2
6                              { 1,1,1,1,0,0,1 }, //=3
7                              { 0,1,1,0,0,1,1 }, //=4
8                              { 1,0,1,1,0,1,1 }, //=5
9                              { 1,0,1,1,1,1,1 }, //=6
10                             { 1,1,1,0,0,0,0 }, //=7
11                             { 1,1,1,1,1,1,1 }, //=8
12                             { 1,1,1,0,0,1,1 }, //=9
13                             { 0,0,0,0,0,0,0 } //=off
14                           };
15 int pins[] = {2,3,4,5,6,7,8,9};
16
17 void setup () {
```

```
19      for(int i=0; i < 7; i++) {
20          pinMode(pins[i], OUTPUT);
21      }
27  }
27
28  // Καθορισμός συνάρτησης ενός ψηφίου
29  void setDigit(byte digit) {
31    for (int i = 0; i < 7; ++i) {
32      digitalWrite(pins[i], digitArray[digit][i]);
34    }
35  }
36
37  void loop() {
38    setDigit(10);  // display off
39    delay(500);
40
41    for (byte count = 0; count < 10; ++count) {
42      setDigit(count);
43      delay(700);
44    }
45  }
```

## Ανάλυση του παραπάνω κώδικα

Στον παραπάνω ενδεικτικό κώδικα από το παράδειγμά μας παρατηρήστε ότι κατ 'αρχάς, ορίζουμε τις μεταβλητές μας. Επίσης παρατηρήστε ότι για να εμφανίσουμε ένα ψηφίο, πρέπει να ελέγξουμε επτά (7) ακίδες (εδώ δεν χρησιμοποιούμε την "κουκίδα" ).

**Παράδειγμα:**

Για να εμφανίσουμε το ψηφίο "1", πρέπει να ορίσουμε το "b" και το "c" σε λογικό HIGH (pin 3 και pin 4) και όλα τα άλλα υπόλοιπα στοιχεία πρέπει να τα έχουμε ορίσει σε λογικό LOW.

Για να γίνει ο κώδικάς μας εύκολος ορίζουμε το μοτίβο των pin σε έναν πίνακα. Ορίζουμε έντεκα σειρές και κάθε σειρά έχει επτά στήλες. Σε κάθε σειρά αποθηκεύουμε το μοτίβο των pin ενός ψηφίου. Αρχίζουμε με την τιμή "**a**", μετά με την τιμή "**b**" και τελειώνουμε με την τιμή "**g**". Π.χ. η σωστή σειρά για το ψηφίο "4" είναι: **0 1 1 0 0 1 1.**

**void setDigit() { ... }**

```
void setDigit(byte digit) {
  for (int i = 0; i < 7; ++i) {
    digitalWrite(pins[i], digitArray[digit][i]);
```

```
  }
}
```

Χρησιμοποιούμε (καθορίζουμε) μια λειτουργία για να ορίσουμε ένα ψηφίο. Ξεκινάμε τη λειτουργία με το ψηφίο, το οποίο θέλουμε να δούμε. Ο αριθμός ψηφίων και ο αριθμός γραμμών στον πίνακα μας είναι ίδιοι. Ο βρόχος συλλέγει κάθε τιμή από τη συστοιχία και στέλνει τη σωστή τιμή με το **digitalWrite** στο σχετικό pin.

**void loop() {...}**

**setDigit (10):** Σημαίνει, θέλουμε να δούμε τη σειρά 10 από τη συστοιχία μας. Η σειρά 10 είναι "απενεργοποιημένη", κάθε pin έχει ρυθμιστεί σε λογικό LOW.

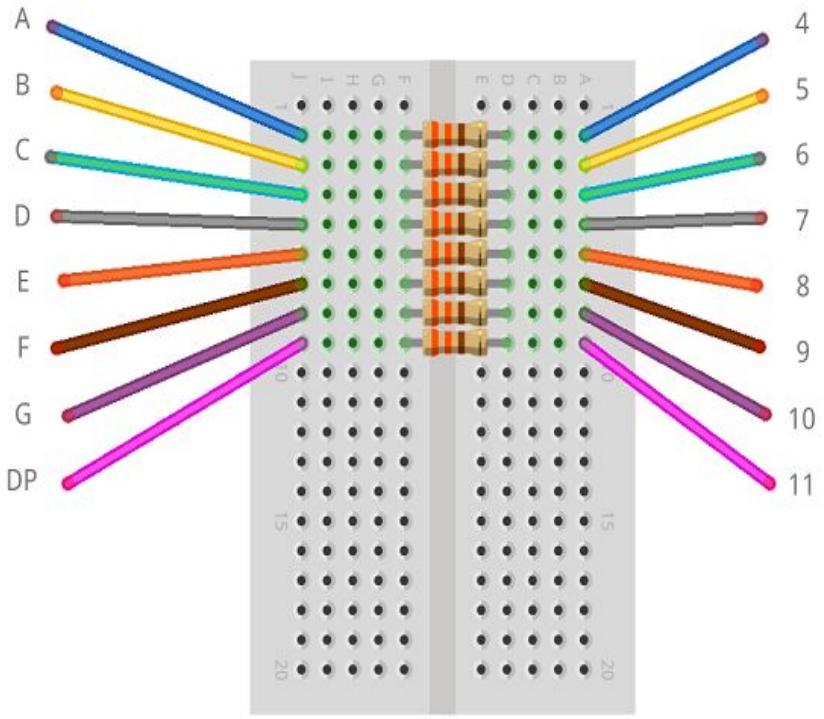
### Δοκιμάζοντας τις συνδέσεις

Αν επιθυμούμε να ελέγξουμε αν έχουμε συνδέσει σωστά ένα 7-segment display τότε σε ένα breadboard κατασκευάζουμε το κύκλωμα με τις αντιστάσεις των 220 Ohm.

Τα γράμματα που χρησιμοποιούνται για την ετικέτα κάθε τμήματος είναι στάνταρ. Θα βοηθήσει όταν χρησιμοποιούμε αναφορές για εμφάνιση χαρακτήρων αν χρησιμοποιούμε την ίδια σειρά.

Συνδέστε έναν από τους ακροδέκτες με την ένδειξη 5V στον ακροδέκτη 5V του Arduino.

Τοποθετήστε τις αντιστάσεις σε ένα ξεχωριστό μέρος του breadboard σας, που εκτείνεται από το μεσαίο τμήμα και συνδέστε τις όπως φαίνεται στο παρακάτω διάγραμμα,




## Ενδεικτικός κώδικας ελέγχου ενός 7-segment Display κοινής ανόδου

```
/* Ο παρακάτω κώδικας μπορεί να χρησιμοποιηθεί για τη δοκιμή των συνδέσεων.
   Χρησιμοποιείται ένας πίνακας για την αναγνώριση των καθόδων που συνδέονται
   με τις ψηφιακές ακίδες.
   Η διαδικασία ρύθμισης οδηγεί κάθε μία από αυτές σε κατάσταση HIGH,
   έτσι ώστε ο κώδικας να ξεκινάμε όλες τις λυχνίες LED να είναι αναμμένες.
   Ο παρακάτω βρόχος οδηγεί μία-μία από τις καθόδους σε κατάσταση LOW
   με τη σειρά, έτσι ώστε να μπορέσουμε να ελέγξουμε ότι οι συνδέσεις είναι
   με την ίδια σειρά, που καθορίζεται στα διαγράμματα της προηγούμενης εικόνας. */

int ledpins[] = {4,5,6,7,8,9,10,11};

void setup()
{
  for (int i =0;i<8;i++)
 {
      pinMode(ledpins[i],OUTPUT);
  digitalWrite(ledpins[i], HIGH);
 }
}

void loop()
{
   for (int i =0;i<8;i++)
```

```
 {
  digitalWrite(ledpins[i], LOW);
  delay(500);
  digitalWrite(ledpins[i], HIGH);
  delay(500);
 }
}
```

## Αξιοποίηση αποκωδικοποιητή για μείωση των απαιτούμενων εξόδων

Για απλοποίηση του κυκλώματος και μείωση των απαιτούμενων εξόδων του Arduino μπορεί να χρησιμοποιηθεί ένας BCD σε 7-segment Display Decoder όπως είναι ο 74LS47. Με τον παραπάνω decoder απαιτούνται μόνο 4 έξοδοι σε αντίθεση με τις 8 που απαιτεί η τυπική συνδεσμολογία.

Για τον ίδιο σκοπό μπορεί να χρησιμοποιηθεί και ο 74HC595 που είναι καταχωρητής ολίσθησης 8-bit.

## Χρήση της SevSeg βιβλιοθήκης για απλοποίηση του κώδικα

Η συμπερίληψη γίνεται από το μενού «Σχέδιο» => «Συμπερίληψη βιβλιοθήκης» και επιλογή SevSeg.

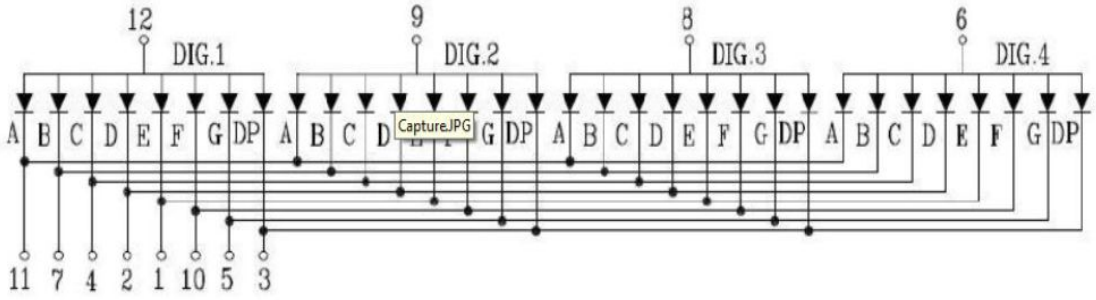
```
// Χρήση SevSeg βιβλιοθήκης

#include <SevSeg.h>

SevSeg sevseg;   //Ορισμός 7-segment display
int n=0;
void setup() {
    byte numDigits = 1;
    byte digitPins[] = {1};
    byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
    byte hardwareConfig = COMMON_CATHODE;
    bool updateWithDelays = true;
    bool leadingZeros = false;
    //Υπάρχει αντίσταση μόνο στο ποδαράκι 4 ή 8
    bool resistorsOnSegments=false;

   sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins,
resistorsOnSegments, updateWithDelays, leadingZeros);
     sevseg.setBrightness(90);
}

void loop() {
     sevseg.setNumber(n, 1);
     if( ++n >= 10)
```

```
24              n = 0;
25          for(int j=1; j <= 30; j++) {
26              sevseg.refreshDisplay();
27              delay( 10);
28          }
29  }
```

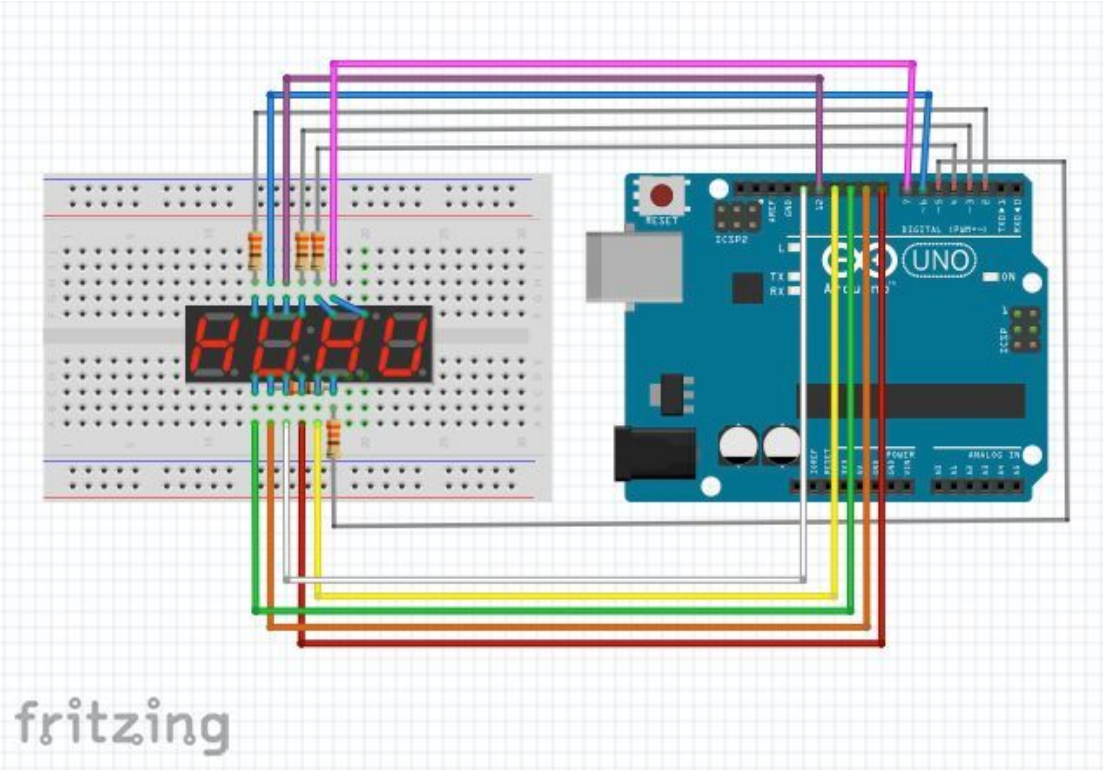
# 4-ψήφιο 7 segment Display

## Εξωτερική μορφή


## Ηλεκτρικό ισοδύναμο



## Ενδεικτικός κώδικας

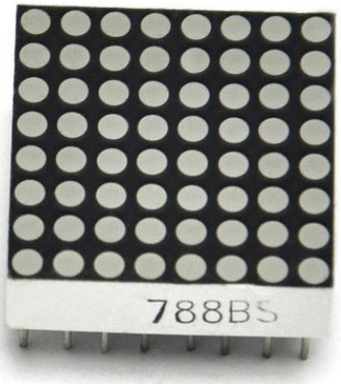



```
// Χρήση SevSeg βιβλιοθήκης
#include <SevSeg.h>

SevSeg sevseg;   //Ορισμός 7-segment display
int n=0;
void setup() {
    byte numDigits = 4;
    byte digitPins[] = {2, 3, 4, 5};
    byte segmentPins[]={6,7,8,9,10,12,11,13};
    byte hardwareConfig = COMMON CATHODE;
    bool updateWithDelays = true;
    bool leadingZeros = false;
    bool resistorsOnSegments=false;

     sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins,
resistorsOnSegments, updateWithDelays, leadingZeros);
     sevseg.setBrightness(90);
}

void loop() {
     sevseg.setNumber(n, 0);
     if( ++n > 9999)
          n = 0;
     for(int j=1; j <= 30; j++) {
          sevseg.refreshDisplay();
          delay( 10);
     }
}
```

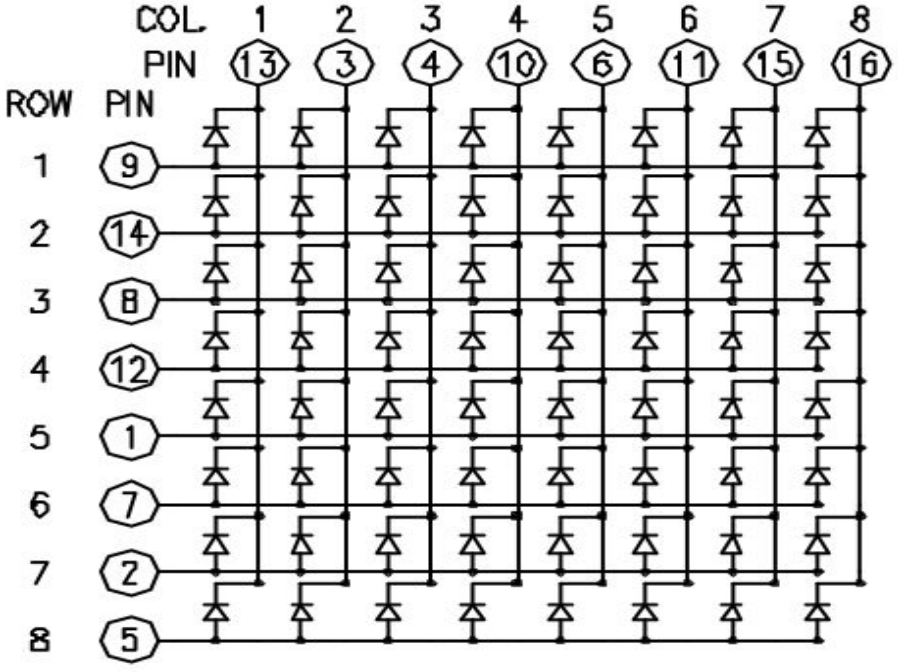
# Dot Matrix Display 8x8

Οι οθόνες LED συχνά συσκευάζονται ως **πίνακες LED** που τα αποκαλούμε **dot matrix Display** και οι οποίες είναι διατεταγμένες σε σειρές **κοινών ανόδων και στηλών κοινών καθόδων ή αντίστροφα**.

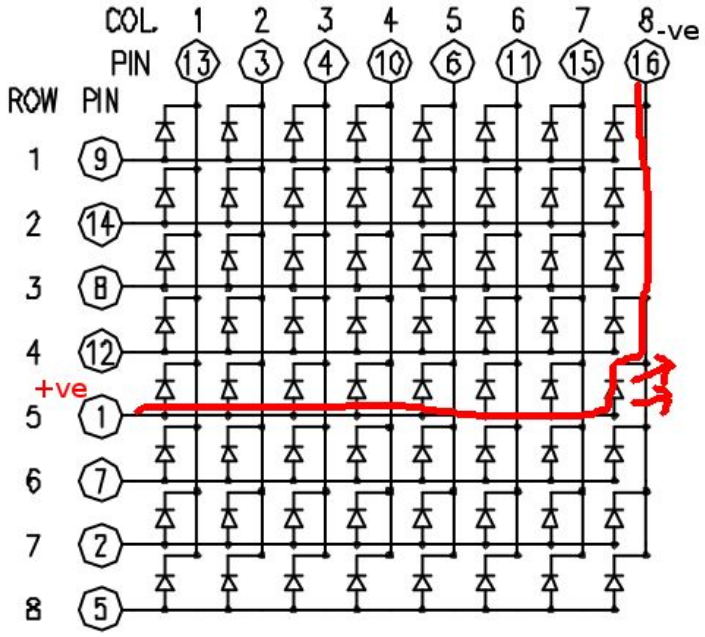
Ο τετραγωνικός πίνακας που πρόκειται να χρησιμοποιήσουμε σε αυτή την εβδομάδα δεν είναι τίποτε άλλο παρά ένας πίνακας που συχνά αποκαλείται **8×8.** Αυτό σημαίνει ότι έχει **8 στήλες** και **8 σειρές**, περιέχοντας συνολικά 64 μικρά ενσωματωμένα **LED**.




Η επόμενη εικόνα δείχνει το ηλεκτρικό του ισοδύναμό του.



Αυτές οι οθόνες είναι πολύ χρήσιμες και έχουν ένα αρκετά ευρύ πεδίων εφαρμογών. Για να ελέγξουμε μια τέτοια μήτρα, συνδέουμε τις σειρές (ROW) και τις στήλες (COL) σε ένα μικροελεγκτή (Arduino). **Οι στήλες συνδέονται με τις κάθοδοι LED, οπότε μια στήλη πρέπει να είναι χαμηλή για να ανάψει οποιοδήποτε από τα LED της στήλης αυτής. Οι σειρές συνδέονται με τις άνοδοι των LED, επομένως η σειρά πρέπει να είναι ΥΨΗΛΗ για να ανάψει ένα μεμονωμένο LED. Εάν η σειρά και η στήλη είναι και οι δύο σε λογικό HIGH είτε και οι δύο σε λογικό LOW, δεν ρέει τάση μέσω της LED και έτσι δεν ανάβει το αντίστοιχο LED.**




Δηλαδή για να ελέγξουμε ένα μεμονωμένο LED, ρυθμίζουμε τη στήλη LOW και τη σειρά HIGH. Για να ελέγξουμε πολλαπλές λυχνίες LED στη σειρά, ρυθμίζουμε ολόκληρη τη σειρά που μας ενδιαφέρει σε HIGH και στη συνέχεια πάμε σε όλες τις στήλες και τις ρυθμίζουμε LOW, οπότε έτσι θα «ανάψει» μία ολόκληρη σειρά (γραμμή). Ανάλογα με την περίπτωση μπορούμε να ανάψουμε τις μισές αυτής της σειράς οι και μία πάρα μία ή ό,τι άλλο επιθυμούμε.

Παρόλο που στο εμπόριο υπάρχουν προκαθορισμένες μήτρες LED, συνήθως των 8x8 μπορούμε επίσης και εμείς να δημιουργήσουμε το δικό μας πίνακα από 64 LED, χρησιμοποιώντας το σχηματικό σχήμα που φαίνεται στην παραπάνω εικόνα.

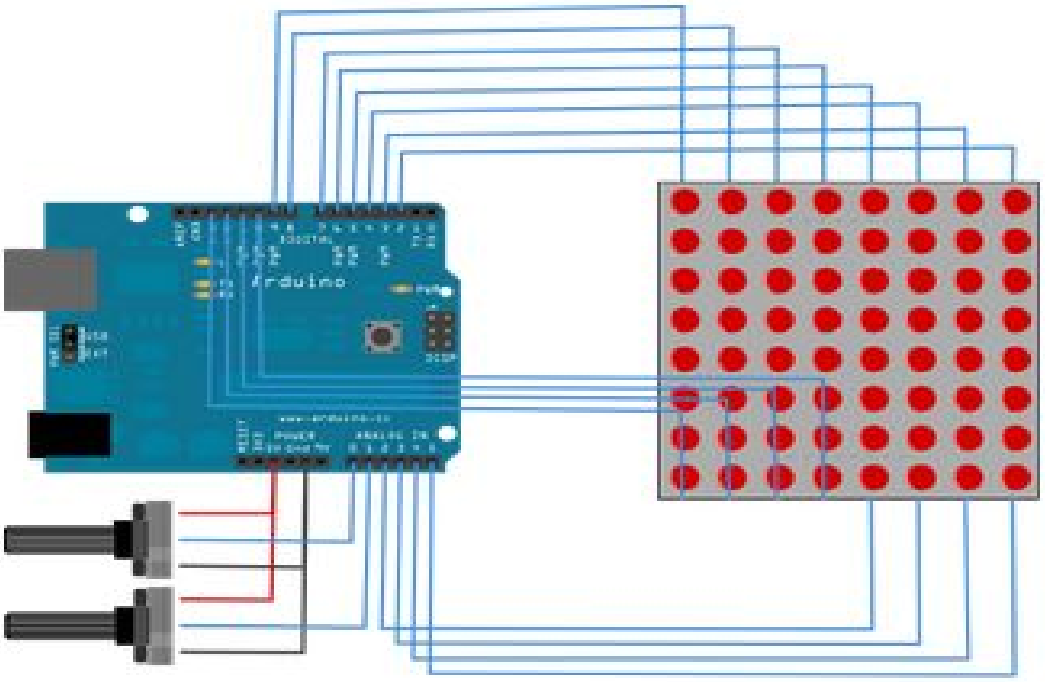
Δεν έχει σημασία ποιες ακίδες του μικροελεγκτή θα συνδέσουμε στις σειρές και στις στήλες, επειδή μπορούμε να αντιστοιχίσουμε αυτές τις καταστάσεις στο λογισμικό. Αυτό που έχει σημασία είναι να συνδέσουμε τα pin με έναν τέτοιο τρόπο και διάταξη που κάνει την καλωδίωση ευκολότερη. Εδώ σας δίνεται ένας πίνακας των συνδέσεων των ακίδων, με βάση το παραπάνω διάγραμμα:

| Matrix pin no. | Row | Column | Arduino pin number |
|---|---|---|---|
| 1 | 5 | | 13 |
| 2 | 7 | | 12 |
| 3 | | 2 | 11 |
| 4 | | 3 | 10 |
| 5 | 8 | | 16 (analog pin 2) |
| 6 | | 5 | 17 (analog pin 2) |
| 7 | 6 | | 18 (analog pin 2) |
| 8 | 3 | | 19 (analog pin 2) |
| 9 | 1 | | 2 |
| 10 | | 4 | 3 |
| 11 | | 6 | 4 |
| 12 | 4 | | 5 |
| 13 | | 1 | 6 |
| 14 | 2 | | 7 |
| 15 | | 7 | 8 |
| 16 | | 8 | 9 |

## Σύνδεση ενός dot matrix Display 8x8 με το Arduino

**Υλικά που θα χρειαστούμε:**

- Πλακέτα Arduino ή Genuino
- 8 x 8 LED dot Matrix Display
- 2 x 10k Ohm ποτενσιόμετρα
- Καλώδια-αγωγοί σύνδεσης
- breadboard



## Ενδεικτικός κώδικας σάρωσης ενός 8x8 dot matrix Display

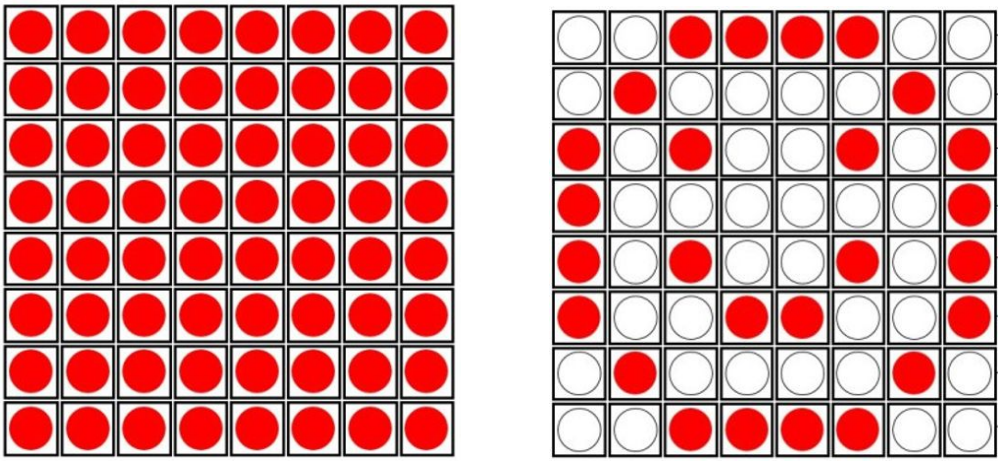
```
/*
  Σάρωση γραμμή-στήλη σε ένα  8x8 LED matrix με είσοδο X-Y συντεταγμένες
  Αυτό το παράδειγμα ελέγχει ένα 8x8 LED matrix χρησιμοποιώντας 2 αναλογικές εισόδους
  Ο κώδικας «δουλεύει» άψογα με τον τύπο Lumex LDM-24488NI Matrix ή και τα συμβατά αυτού.
  Για το συγκεκριμένο τύπο dot matrix μπορείτε να δείτε περισσότερες λεπτομέρειες εδώ σχετικά με τις συνδέσεις των ακροδεκτών του.
  Για άλλο τύπο LED cathode column matrixes, το μόνο που χρειάζεται είναι να αλλάξετε τα pin
  των αριθμών στα row[] και στα column[] των πινάκων.

// Πίνακας με τα pin των γραμμών:
const int row[8] = { 2, 7, 19, 5, 13, 18, 12, 16};
// Πίνακας με τα pin των στηλών:
const int col[8] = { 6, 11, 10, 3, 17, 4, 8, 9};
// Πίνακας δύο διαστάσεων των εικονοστοιχείων (pixels):
int pixels[8][8];

// Θέση του cursor:
int x = 5;
int y = 5;

void setup() {
  for (int thisPin = 0; thisPin < 8; thisPin++) {
        // καθορισμός των pins εξόδου (outputs):
        pinMode(col[thisPin], OUTPUT);
        pinMode(row[thisPin], OUTPUT);
        //Θέτει τα pins της στήλης σε λογικό HIGH
```

```
        digitalWrite(col[thisPin], HIGH);
    }

    //Καθορισμός του εικονοστοιχείου (pixel) του matrix:
    for (int x = 0; x < 8; x++) {
        for (int y = 0; y < 8; y++) {
            pixels[x][y] = HIGH;
        }
     }
}

void loop() {
    // Διαβάζει την είσοδο:
    readSensors();

    // Σχεδιάζει την οθόνη:
    refreshScreen( pixels);
}

void readSensors() {
    //Απενεργοποιεί (turn off) την τελευταία θέση:
    pixels[x][y] = HIGH;
    //Διαβάζει τους αισθητήρες  για τις τιμές των  X και Y:
    x = 7 - map(analogRead(A0), 0, 1023, 0, 7);
    y = map(analogRead(A1), 0, 1023, 0, 7);
    //Θέτει την θέση του νέου pixel σε λογικό LOW έτσι ώστε το LED θα
γίνει ON στο επόμενο refesh
    pixels[x][y] = LOW;
}

void refreshScreen(int data[8][8]) {
    // Θέτει πάλι τις γραμμές (anodes):
    for (int thisRow = 0; thisRow < 8; thisRow++) {
    // Θέτει το pin της γραμμής (άνοδος) σε κατάσταση HIGH:
       digitalWrite(row[thisRow], HIGH);
       //Θέτει πάλι τις στήλες (cathodes):
       for (int thisCol = 0; thisCol < 8; thisCol++) {
           // «εισάγει» την κατάσταση του τρέχοντος εικονοστοιχείου:
           int thisPixel =data[thisRow][thisCol];
           // όταν η γραμμή είναι HIGH και η στήλη είναι LOW,
           // τότε τα LED τα οποία «συναντά» γίνονται ON:
           digitalWrite(col[thisCol], thisPixel);
           // σβήνει (turn off) το εικονοστοιχείο (pixel):
           if (thisPixel == LOW) {
               digitalWrite(col[thisCol], HIGH);
           }
        }
       // Θέτει το pin της γραμμής LOW για να σβήσειόλη η γραμμή
       digitalWrite(row[thisRow], LOW);
    }
}
```

**Σημείωση**

Εάν θέλουμε να εμφανίσουμε κάτι στον πίνακα-μήτρα των παραπάνω LEDs, απλά για να καταλήξουμε στο τελικό «σχέδιο-γράφημα» πρέπει να γνωρίζουμε αν σε μια καθορισμένη σειρά ή στήλη, τα LED που θα πρέπει να είναι ενεργοποιημένα ή απενεργοποιημένα.

Για παράδειγμα, αν επιθυμούσαμε να εμφανίσουμε στον πίνακα dot matrix ένα χαρούμενο πρόσωπο, δείτε την παρακάτω εικόνα στην αριστερά και δεξιά πλευρά της για κατανοήσετε το ΠΡΙΝ και το ΜΕΤΑ



Στον προγραμματισμό ορίζουμε ένα πίνακα 8Χ8 για κάθε σχέδιο και στον πίνακα αυτό υπάρχει μηδέν όπου το αντίστοιχο led θέλουμε να παραμείνει σβηστό και 1 όπου θέλουμε να φωτίζει. Στη συνέχεια περνάμε στη refresh screen σαν παράμετρο τον πίνακα αυτό. Αν π.χ. θέλουμε 4 τέτοια σχέδια χρειαζόμαστε 4 τέτοιους πίνακες.

# LCD Display 2 γραμμών και 16 χαρακτήρων

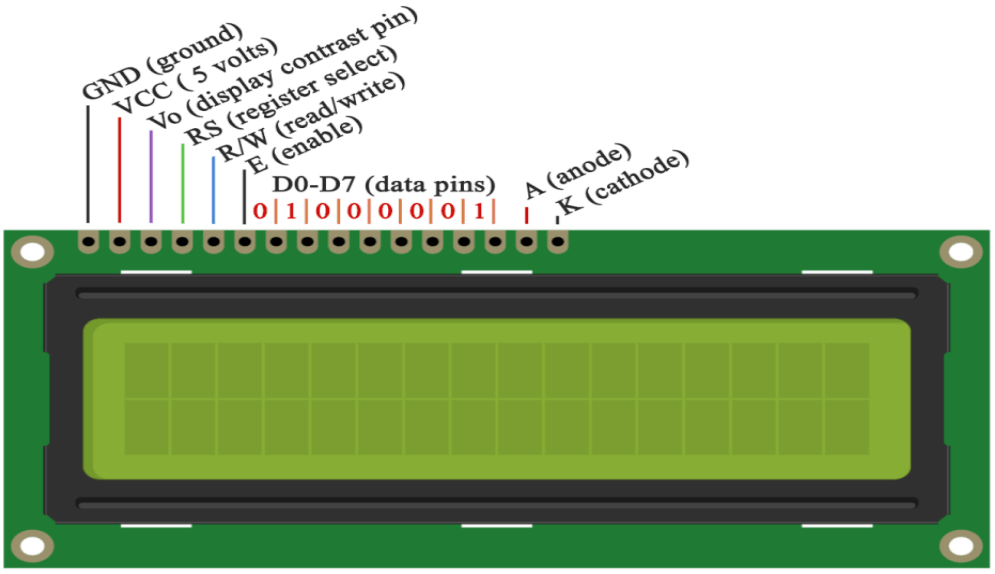
Τα τύπου LCDs είναι πάρα πολύ δημοφιλή και βασικά χρησιμοποιούνται στις ηλεκτρονικές κατασκευές εκεί που απαιτείται η εμφάνιση πληροφοριών-δεδομένων από την συσκευή προς τον άνθρωπο. Επίσης έχουν χαμηλό κόστος αγοράς.

Εδώ θα μελετήσουμε πως μπορούμε να συνδέσουμε και επικοινωνήσουμε με ένα τύπου LCD (**L**iquid **C**rystal **D**isplay) τύπου *Megatronic 1602*, των 2 γραμμών και 16 χαρακτήρων ανα γραμμή, με την πλακέτα του Arduino.

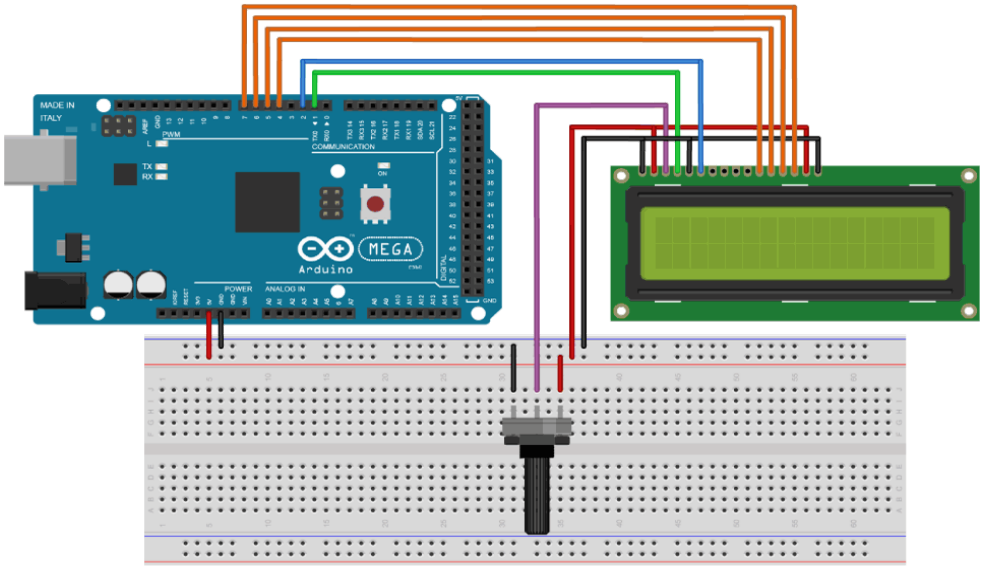
## LCD Pinout

Στην παρακάτω εικόνα βλέπετε την διάταξη των ακροδεκτών μιας αρκετά διαδεδομένης οθόνης τύπου LCD δύο γραμμών και 16 χαρακτήρων ανά γραμμή. Η συγκεκριμένη επικοινωνεί με τον μικροελεγκτή μέσω 16 ακροδεκτών.

**Pin layout LCD Display 1602 Megatronics**




Για την διασύνδεση της παραπάνω πλακέτας συνήθως χρησιμοποιούμε 6 ψηφιακές εξόδους από την πλακέτα του Arduino. Οι ακροδέκτες **D4-D7** συνδέονται με τις ψηφιακές πόρτες **4-7 του Arduino**. Τα **pin RS και E** συνδέονται με το **pin 2 και 3** του Arduino. Το **R/W** pin συνδέεται στην γείωση (Ground) και το **Vo pin** θα το συνδέσουμε στο ποτενσιόμετρο. Δείτε μία τέτοια τυπική διάταξη στην παρακάτω εικόνα.



Το πρώτο πράγμα που πρέπει να κάνουμε, για να επικοινωνήσει ο μικροελεγκτής με το LCD, είναι να εισάγουμε την ανάλογη βιβλιοθήκη υποστήριξης υγρών κρυστάλλων. Αυτό μπορούμε να το κάνουμε ως εξής: **Σχέδιο → Συμπερίληψη Βιβλιοθήκης → LiquidCrystal**. Στη συνέχεια, πρέπει να δημιουργήσουμε ένα αντικείμενο LC. Λάβετε υπόψη σας ότι οι παράμετροι αυτού του αντικειμένου πρέπει να είναι οι αριθμοί των ακίδων ψηφιακής εισόδου του πίνακα Arduino σε αντιστοιχία με τις ακίδες της οθόνης LCD ως εξής: **(RS, Enable, D4, D5, D6, D7)**. Στη ρύθμιση πρέπει να αρχικοποιήσουμε τη διεπαφή στην οθόνη LCD και να καθορίσουμε τις διαστάσεις της οθόνης χρησιμοποιώντας τη λειτουργία **begin().**

Παρατηρήστε ότι μέσα στον βρόχο γράφουμε το κύριο πρόγραμμα μας. Χρησιμοποιώντας τη συνάρτηση **print()** μπορούμε να «εκτυπώσουμε» στην οθόνη LCD. Η συνάρτηση **setCursor()** χρησιμοποιείται για τη ρύθμιση της θέσης στην οποία θα εμφανιστεί το επόμενο κείμενο που έχει γραφτεί στην οθόνη LCD. Η συνάρτηση **blink()** χρησιμοποιείται για την εμφάνιση ενός δρομέα που αναβοσβήνει και της **noBlink()** για απενεργοποίηση αυτού. Η συνάρτηση **cursor()** χρησιμοποιείται για την εμφάνιση του δρομέα υπογράμμισης και η **noCursor()** για απενεργοποίησή του. Χρησιμοποιώντας την **clear()**, μπορούμε να καθαρίσουμε την οθόνη LCD.

## Ενδεικτικός κώδικας οδήγησης ενός LCD Display

```
1  // Κώδικας οδήγησης LCD Display
2  //  Τύπος LCD "Mechatronics 1602"
```

```
3  #include <LiquidCrystal.h> // Συμπεριέλαβε βιβλιοθήκες LiquidCrystal
4  // Δημιουργία ενός LCD object. Παράμετροι: (rs, enable, d4, d5, d6, d7)
5  LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
6
7  void setup() {
8      // Αρχικοποίηση του  interface LCD , και καθορισμός διαστάσεων (πλάτος και
   ύψος )
9      lcd.begin(16,2);
10 }
11
12 void loop() {
13     lcd.print("Arduino"); // Εκτύπωσε – Εμφάνισε "Arduino" στο LCD
14     delay(3000); // 3 seconds delay
15      // Καθορισμός της θέσης που θα αναγραφεί το κείμενο στην LCD οθόνη
15     lcd.setCursor(2,1);
16     lcd.print("LCD TEST");
17     delay(3000);
18     lcd.clear(); // Καθάρισε το display
19     lcd.blink(); //Εμφάνισε το blinking LCD cursor
20     delay(4000);
21     lcd.setCursor(7,1);
22     delay(3000);
23     lcd.noBlink(); // Turns off the blinking LCD cursor
24     // Εμφάνισε ένα underscore (line) στην θέση που θα γράψεις τον επόμενο
   χαρακτήρα
25     lcd.cursor();
26     delay(4000);
27     lcd.noCursor(); // Απόκρυψη LCD cursor
28     lcd.clear(); // Καθαρισμός LCD screen
29 }
```

## Προσαρμοζόμενοι χαρακτήρες

Επίσης έχουμε την δυνατότητα να «γράψουμε» τους δικούς μας χαρακτήρες στο LCD. Υποστηρίζει μέχρι 8 χαρακτήρες των **5x8** εικονοστοιχείων (pixels). Μπορούμε να καθορίσουμε την εμφάνιση του κάθε χαρακτήρα με την βοήθεια ενός πίνακα από 8 bytes.

Στον παρακάτω ενδεικτικό κώδικα μπορούμε να παρατηρήσουμε πώς μπορούμε να καθορίσουμε την εμφάνιση ενός χαρακτήρα μεταβάλλοντας το 0 σε 1 που αντιπροσωπεύει τα **5×8** εικονοστοιχεία (pixels).

Για να το επιτύχουμε αυτό θα πρέπει να δημιουργήσουμε τον προσαρμοσμένο χαρακτήρα χρησιμοποιώντας τη συνάρτηση **createChar()**. Η πρώτη παράμετρος σε αυτή τη λειτουργία είναι ένας αριθμός μεταξύ 0 και 7 ή πρέπει να έχουμε κρατήσει έναν από τους 8 υποστηριζόμενους προσαρμοσμένους χαρακτήρες. Η δεύτερη παράμετρος είναι το όνομα του πίνακα των bytes. Γράφουμε τον προσαρμοσμένο χαρακτήρα στην οθόνη χρησιμοποιώντας τη συνάρτηση **write()** και ως παράμετρος χρησιμοποιούμε τον αριθμό του χαρακτήρα.

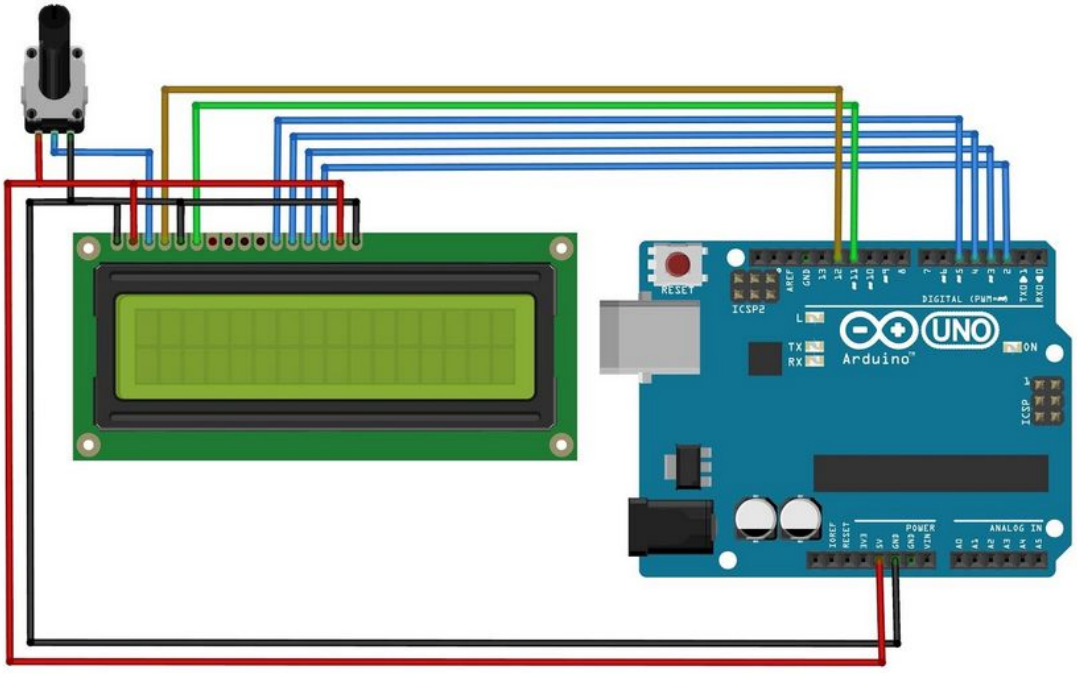
In the source code below we can notice how we can specify the appearance of the character by changing the 0 into 1 which represents the 5×8 pixels. In the setup we have to create the custom character using the **createChar()** function. The first parameter in this function is a number between 0 and 7, or we have to reserve one of the 8 supported custom characters. The second parameter is the name of the array of bytes. We write the custom character to the display using the **write()** function and as a parameter we use the number of the character.

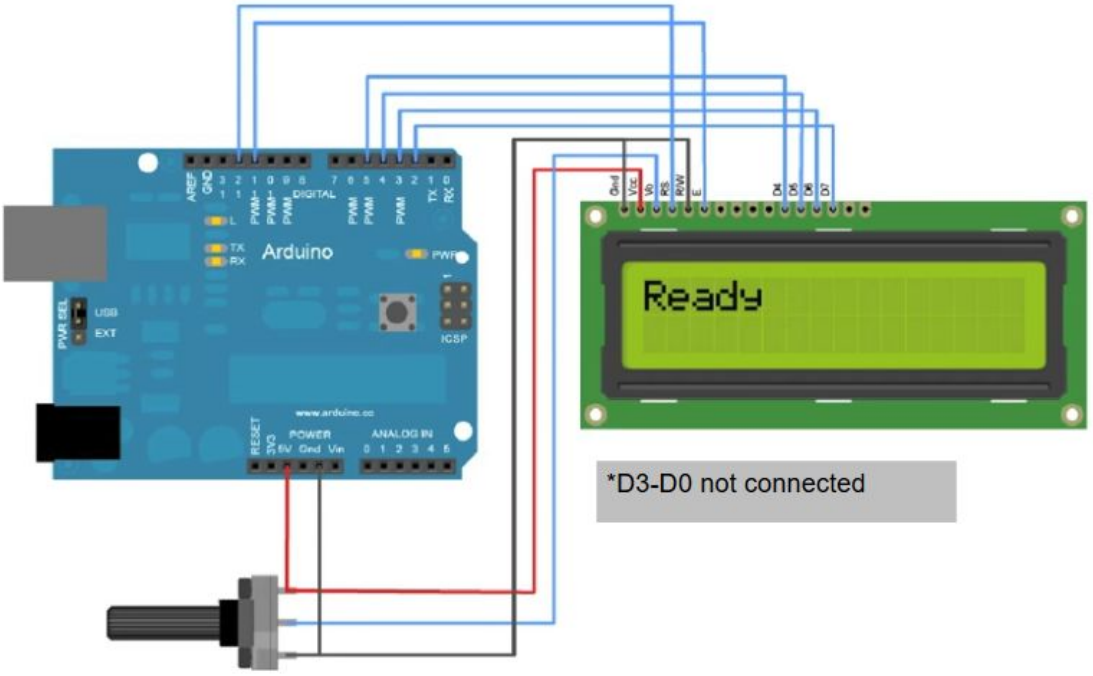
### Ενδεικτικός κώδικας δημιουργίας προσαρμοσμένων χαρακτήρων

```
#include <LiquidCrystal.h>

byte slash[8]= { // Array of bytes
    B00001, // B stands for binary formatter and the 5 numbers are the pixels
    B00010,
    B00100,
    B01000,
    B10000,
    B00000,
    B00000,
    B00000,
};
//Δημιουργία lcd αντικειμένου. Παράμετροι: (rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(1, 2, 4, 5, 6, 7);

void setup() {
    //Ορισμός διάστασης οθόνης
    lcd.begin(16,2);
    //Δημιουργία χαρακτήρα 5X8 σχεδιασμένο από το χρήστη.
```

```
20      //Επιτρέπονται μέχρι 7 τέτοιοι χαρακτήρες
21      lcd.createChar(7, slash);
22  }
23
24  void loop() {
25      for(int i=0;i<=15;i++) {
26          //Ορισμός των συντεταγμένες X,Y στις οποίες θα γράψει
27          lcd.setCursor(i,0);
28          lcd.write(7); // Writes a character to the LCD
29          delay(1000); // 1 second delay
30          lcd.clear(); // Write a character to the LCD
31      }
32  }
```

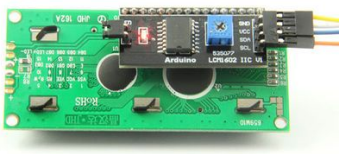
Στις δύο παρακάτω εικόνες μπορούμε να παρατηρήσουμε μια τυπική συνδεσμολογία ενός LCD Display (Megatronics 1602). Αναφερόμαστε και αναλύουμε αυτή την συνδεσμολογία, καθώς επίσης και τον τρόπο πως θα ελέγξουμε μέσω του Arduino αυτό το LCD Display, διότι αυτός ο τύπος οθόνης είναι ένας αρκετά διαδεδομένος λόγω του χαμηλού κόστους και της αρκετά καλής ποιότητας του (Contrast). Επίσης εμπεριέχεται και στο κίτ το οποίον έχετε προμηθευτεί. Για ακόμη περισσότερες διευκρινήσεις και πληροφορίες δείτε το ενημερωτικό video εδώ.






# Η Lcd οθόνη και το I2C

Για οικονομία στις θύρες της πλακέτας Arduino μπορεί να χρησιμοποιηθεί το I2C module με το οποίο συνδέεται η οθόνη με το Arduino μόνο με δυο καλώδια και φυσικά τα Vcc και GND!

4-Pin I2C LCD Display

Το i2c module μπορεί να κολληθεί επάνω στην οθόνη ή απλά να συνδεθεί (καλώδια ή breadboard). Διαθέτει ποτενσιόμετρο για την ρύθμιση της αντίθεσης της οθόνης. Περισσότερες πληροφορίες εδώ και στο video.

# Δ Ρ Α Σ Τ Η Ρ Ι Ο Τ Η Τ Ε Σ

## 1η Δραστηριότητα: Ηλεκτρονική κλήρωση από 0-9 με 7-segment display

Κατασκευάστε ένα απλό κύκλωμα μετρητών 0 έως 9 με δύο διακόπτες. Όταν πατάμε τον 1ο διακόπτη να τρέχουν τα νούμερα από το 0 έως το 9 με ρυθμό ένα 1 ψηφίο ανά 30 msec και όταν πατάμε το δεύτερο πλήκτρο θα σταματάει στον αριθμό που που βρισκόταν εκείνη τη στιγμή.

## 2η Δραστηριότητα: Θερμόμετρο με 4-ψήφιο 7 segment display

Ζητείται να συνδέσετε κατάλληλα στο κύκλωμα ένα 4-ψήφιο 7 segment Display και να γράψτε τον ανάλογο κώδικα έτσι ώστε στην ένδειξη να εμφανίζεται η θερμοκρασία με ακρίβεια 1 δεκαδικού ψηφίου (χρήση πλέον και της κουκκίδας). Πλησιάζοντας τον αισθητήρα κοντά στον υπολογιστή ή με τη χρήση ενός σεσουάρ θα πρέπει η θερμοκρασία να μεταβάλλεται.

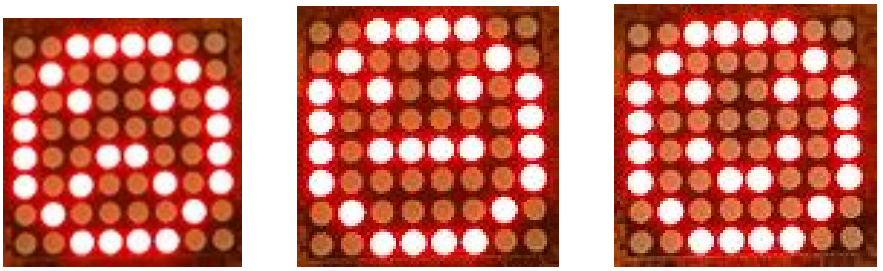
**Αλγόριθμος για τη διόρθωση «λανθασμένων τιμών».**

Ενδεχομένως κάποιες τιμές να είναι λανθασμένες οπότε μπορείτε να κάνετε μία μέτρηση κάθε 1 msec και να κρατάτε σε μία μεταβλητή την τιμή της θερμοκρασίας απεικόνισης. Αν η θερμοκρασία από το μετρητή είναι μεγαλύτερη από τη θερμοκρασία που δείχνει η οθόνη τότε η θερμοκρασία που θα δείξει θα πρέπει να αυξηθεί κατά 0,05 ενώ αν είναι μικρότερη θα πρέπει να μειωθεί κατά 0,05

## 3η Δραστηριότητα: Σχεδιάζοντας γραφικά σε 8X8 dot matrix display

Ζητείται αφού διασυνδέσετε κατάλληλα ένα 8x8 dot matrix display με την πλακέτα του Arduino να γράψετε τον κατάλληλο κώδικα έτσι ώστε να εμφανίζει τρεις τύπους προσώπων: Ένα **θλιμμένο πρόσωπο**, ένα **ουδέτερο πρόσωπο** και ένα **χαρούμενο πρόσωπο**. Οι εναλλαγές φροντίστε να

γίνονται με αργό ρυθμό για να είναι ορατές στο ανθρώπινο μάτι. Επίσης να είναι συνεχόμενες. Δείτε τις τρεις αυτές εναλλαγές στην παρακάτω εικόνα.

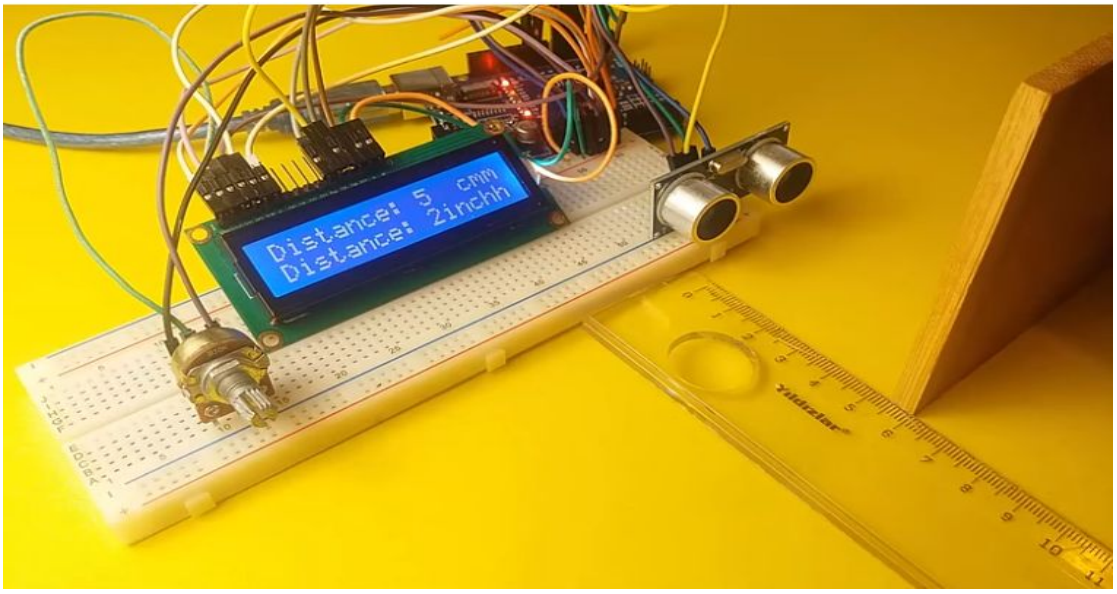


## 4η Δραστηριότητα: Μετρητής απόστασης με LCD

Χρησιμοποιώντας τον αισθητήρα ultrasonic που διαθέτετε κατασκευάστε το κύκλωμα της παρακάτω εικόνας και γράψτε κώδικα έτσι ώστε στην οθόνη του LCD Display να φαίνεται η απόσταση που παρεμβάλλεται προς την κατεύθυνση εκπομπής του ανάμεσα από τον αισθητήρα υπερήχων και από ένα αντικείμενο. Μία ίντσα αντιστοιχεί με 2,54 εκατοστά.

**Υλικά που θα χρειαστείτε**

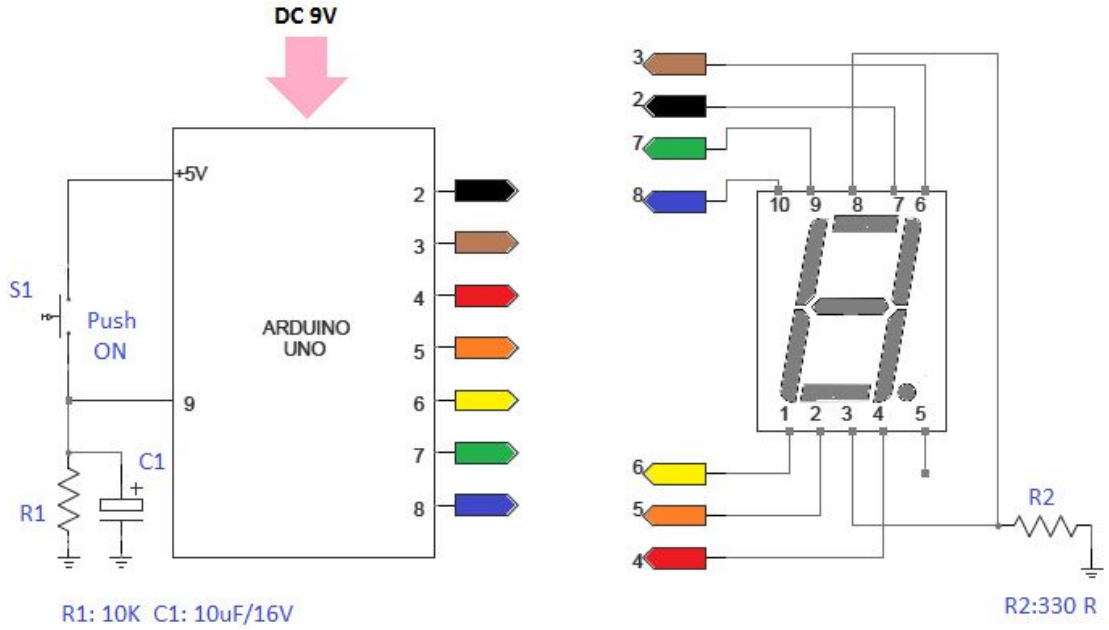
- Πλακέτα Arduino UNO & Genuino UNO
- Ποτενσιόμετρο 10k ohms
- LCD Display 16 χαρακτήρων και 2 γραμμών (Megatron 1602)
- Ultrasonic Sensor (HC-SR04)
- Αγωγοί-καλώδια σύνδεσης
- Breadboard



# 5η Δραστηριότητα: Μετρητής από 0-9 με 7-segment display

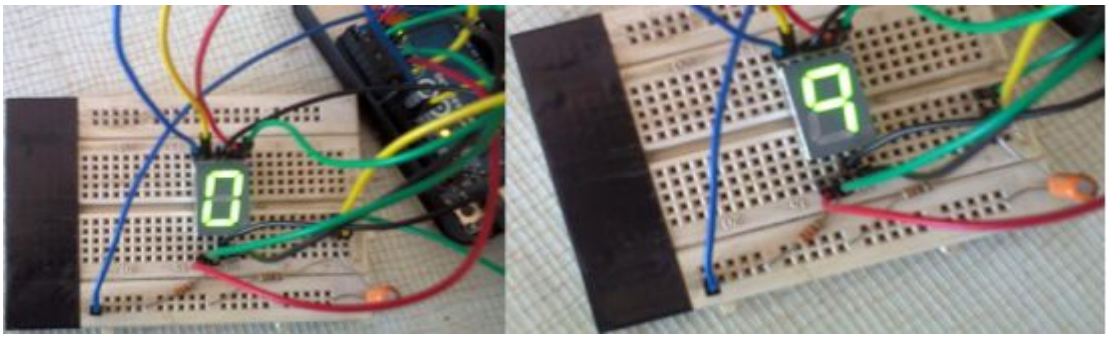
Κατασκευάστε ένα απλό κύκλωμα μετρητών 0 έως 9. Για να το κάνετε αυτό χρησιμοποιήστε μια οθόνη LED κοινής καθόδου (7-segment display) που θα συνδέεται με το Arduino για την εμφάνιση των ψηφίων. Ο κώδικας που θα γράψετε θα επιτρέπει την αύξηση του αριθμού από το 0 έως το 9 με το πάτημα ενός διακόπτη S1 (push button).

Δείτε λεπτομέρειες από το ηλεκτρικό ισοδύναμο στην παρακάτω εικόνα:




Επίσης στον πίνακα που ακολουθεί μπορείτε να δείτε την αντιστοιχία των pins Arduino και μιας οθόνης LED κοινής καθόδου (7-segment Display).

| Arduino Pin | Display Connection |
|---|---|
| 2 | 7 (A) |
| 3 | 6 (B) |
| 4 | 4 (C) |
| 5 | 2 (D) |
| 6 | 1 (E) |
| 7 | 9 (F) |
| 8 | 10 (G) |
| N/C | DP |



**Επισήμανση**

Η σύνδεση των ακροδεκτών ενός 7-segment Display απευθείας με τους ακροδέκτες I / O του Arduino **δεν αποτελεί καλή πρακτική και δεν είναι ασφαλές για την ακεραιότητα του μικροελεγκτή.** Στο παραπάνω κύκλωμα για δοκιμαστικό σκοπό έχουμε παρεμβάλει μόνο μια αντίσταση 330 Ohm (R2) μεταξύ της γειωμένης ράγας (0V) του breadboard και των κοινών ακίδων καθόδου (3 & 8). Ασφαλέστερο είναι να συνδέσετε απευθείας τους ακροδέκτες 3 & 8 της οθόνης στη γείωση, δηλαδή να αφαιρέσετε την αντίσταση R2 και να παρεμβάλετε μια αντίσταση 220 έως 330 Ohm μεταξύ κάθε μιας από τις άλλες συνδέσεις με το Arduino. (Απαιτούνται συνολικά 8 αντιστάσεις).

## 6η Δραστηριότητα: Μετρητής από 0-9 με 7-segment display με overflow

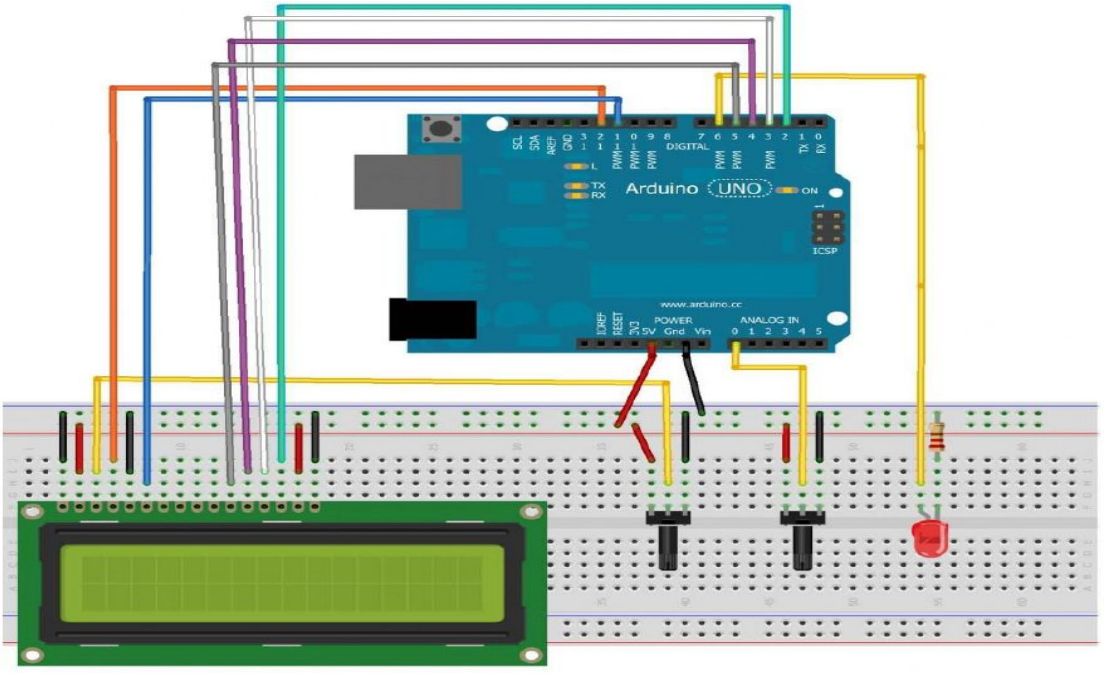
Προσθέστε στο παραπάνω κύκλωμα και δεύτερο διακόπτη **S2** (push button) και ονομάστε αυτόν reset. Κατόπιν κάνετε τις απαραίτητες αλλαγές στον κώδικα της δραστηριότητας 1 έτσι ώστε όταν φτάσει στον μέγιστο αριθμό που μπορεί να απεικονισθεί με ένα μόνο 7-segment display να μας δείχνει την ένδειξη F που για μας σημαίνει overflow. Η μέτρηση από εδώ και κάτω κλειδώνει (σταματάει) και συνεχίζει ΜΟΝΟΝ αν πατηθεί το πλήκτρο reset.

# 7η Δραστηριότητα: Ένδειξη φωτεινότητας LED σε LCD Display 1602

Σε αυτή την δραστηριότητα πρόκειται να εμφανίσουμε με την μορφή «μπάρας» την μεταβολή της φωτεινότητα ενός LED σε ένα LCD Display 16x2.

**Απαιτούμενα υλικά**

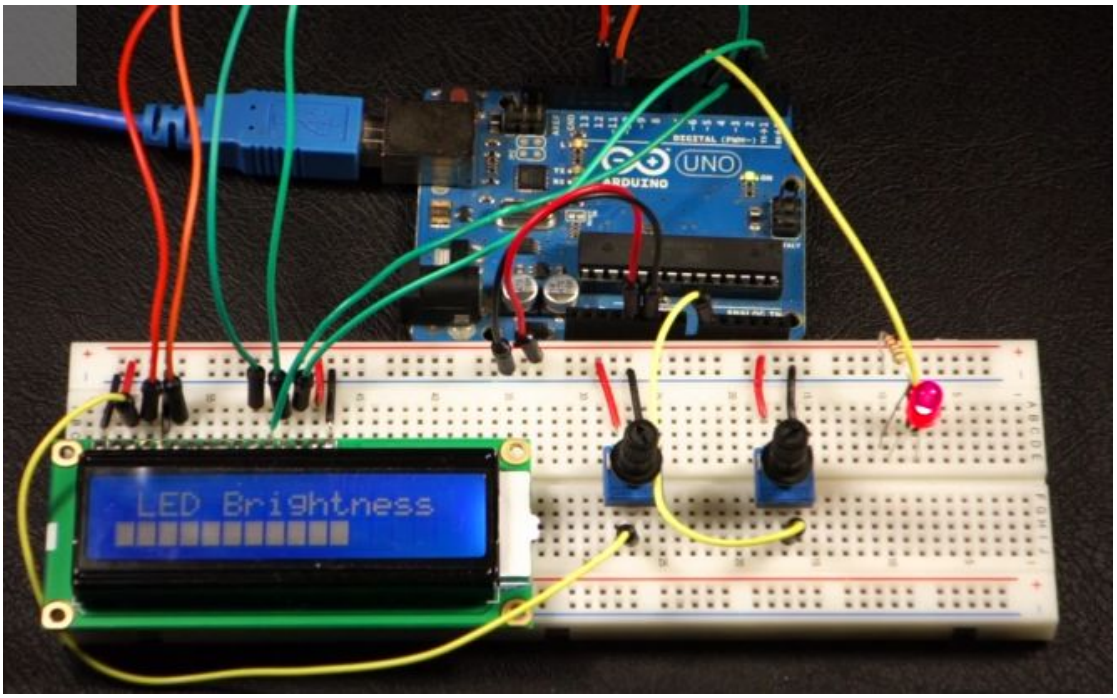
- 1x Arduino
- 1x Breadboard
- 1x LCD 16x2
- 2x 10k Ohm Potentiometers
- 1x 5mm LED
- 1x 220 Ohm Resistor
- Jumper Cables




## LCD Pinout

INTERFACE PIN CONNECTIONS

| Pin No. | Symbol | Level | Description |
|---|---|---|---|
| 1 | VSS | --- | Ground for Logic ( 0V) |
| 2 | VDD | --- | Power supply for Logic (+5.0V) |
| 3 | V0 | --- | Power supply for LCD drive |
| 4 | RS | H/L | Register selection (H:Data register,L:Instruction register) |
| 5 | R/W | H/L | Read/Write selection (H:read,L:Write) |
| 6 | E | H/L⟶L | Enable signal for LCM |
| 7~14 | DB0~DB7 | H/L | Data Bus Lines |
| 15 | LEDA | --- | Power supply for backlight(+5.0V) |
| 16 | LEDK | --- | Power supply for backlight(-) |


LED Brightness
UNO
ARDUINO