Α.Τ.Ε.Ι. Θεσσαλονίκης
Τμήμα Ηλεκτρονικών Μηχανικών

Επιβλέπων Φοιτητής: Καλοπαναγιώτης Βασίλειος
ΚΑΣ : 501806

Επιβλέπων Καθηγητής: Καζακόπουλος Αριστοτέλης

Σύστημα εμφάνισης μηνυμάτων σε πινακίδα LED

Message display System on LED matrix

Ημερομηνία ανάληψης: 24-1-2014

Ημερομηνία περάτωσης: 3-9-2014

Κωδικός πτυχιακής εργασίας:

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της πτυχιακής εργασίας είναι η μελέτη και κατασκευή συστήματος εμφάνισης μηνυμάτων σε πινακίδα LED. Το σύστημα θα αποτελείτε από ένα LED matrix 6 γραμμών και 24 στηλών (24X6) το οποίο θα ελέγχετε από το ardruino Uno Rev.3. Το Arduino Uno είναι μία πλατφόρμα προγραμματισμού και ελέγχου κυκλωμάτων με βάση τον μικροελεγκτή ATmega328 το οποίο παρέχει ότι χρειάζεται ο μικροελεγκτής και ac/dc μετασχηματιστή για παροχή τροφοδοσίας στην πτυχιακή εργασία . Το arduino παρέχει σύνδεση με τον H/Y μέσω USB και βασίζεται σε πλατφόρμα προγραμματισμού σε C/C++ , το περιβάλλον της εφαρμογής προγραμματισμού είναι γραμμένο σε java και βασίζεται σε avr-gcc ανοικτού κώδικα προγράμματα. Το LED matrix οδηγείτε απο κύκλωμα βασισμένο σε shift registers και έναν δεκαδικό μετρητή , και το κύκλωμα ελέγχετε από την πλακέτα του μικροελεγκτή. Έχει την δυνατότητα εμφάνισης μηνυμάτων κινούμενων, οι χαρακτήρες ορίζονται από το πρόγραμμα του μικροελεγκτή.

Summary

The subject of the thesis is the design and manufacture of a message display system on a LED matrix. The system consists of a LED matrix 6 rows and 24 columns (24X6) which will be controlled from the ardruino Uno Rev.3. The Ardruino Uno is a programming platform and control circuits based on the ATmega328 microcontroller which provides all that the microcontroller needs and ac / dc adapter to provide power the circuits. The ardruino provides connection to the H / Y via USB and programming platform based on C / C ++, the application interface program is written in java and based on avr-gcc open source programs. The LED matrix is driven by a circuit board based on shift registers and a decimal counter, and checks the circuit board microcontroller. And it has the capability of a scrolling message display, the characters defined by the program of the microcontroller.

ΠΕΡΙΕΧΟΜΕΝΑ

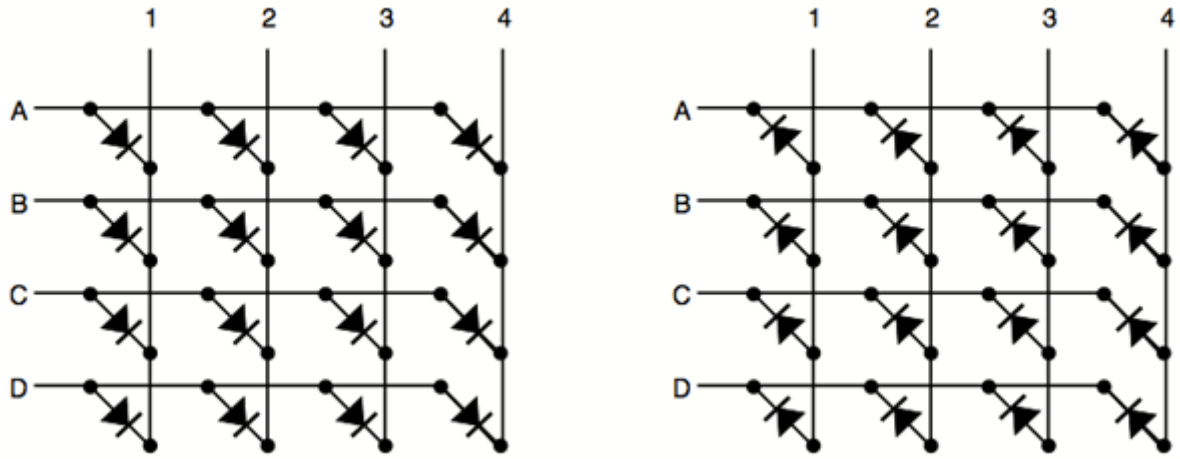
ΣΕΛΙΔΑ

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή στα LED matrix

Δομή των Led Matrix

Σε ένα matrix, είναι η μορφή LEDs διατεταγμένα σε σειρές και στήλες. Μπορείτε επίσης να σκεφτείτε τους ως x και y συντεταγμένες. Ας υποθέσουμε ότι έχουμε 4 × 4 πίνακα. Γραμμές θα πρέπει να επισημαίνονται από το A έως D και στήλες από 1 έως 4 Τώρα μπορούμε να αντιμετωπίσουμε κάθε LED από τη γραμμή και στήλη. Επάνω αριστερά με επικεφαλής θα είναι (A, 1). Κάτω κάτω οδηγήσει θα είναι (Δ, 4).

Led πίνακες έρχονται σε δύο κατηγορίες. Κοινό-σειρά ανόδου (αριστερά) και της κοινής γραμμής καθόδου (δεξιά).

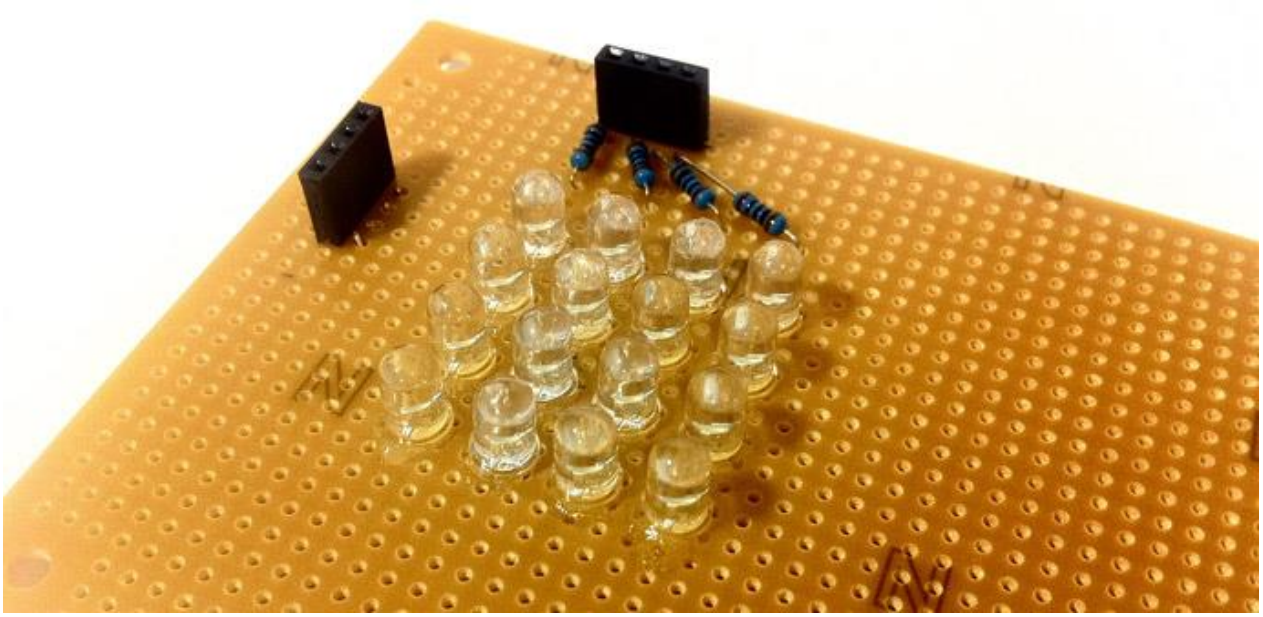


Παραπάνω σχήμα δείχνει τις διαφορετικές διαμορφώσεις. Η διαφορά μεταξύ αυτών των δύο διαμορφώσεις είναι το πώς θα ανάβει μια λυχνία. Με κοινή-σειρά ανόδου πηγές ρεύματος (θετική τάση) είναι συνδεδεμένα με σειρές A..D και η κάθοδος (αρνητική τάση, γείωση) με στήλες 1..4. Με κοινή-σειρά καθόδου που συνδέονται με σειρές A..D και ρεύματα πηγές σε στήλες 1..4.

Για παράδειγμα. Προς το φως κάτω του matrix (D, 4) της κοινής καθόδου θα τροφοδοτήσει θετική τάση στην στήλη 4 και συνδέστε σειρά D στο έδαφος. Με την ίδια λογική λειτουργούν μεγαλύτερα matrix και RGB led Matrix που έχουν 3 LEDs με κοινή γείωση ή κοινή τάση.

Κτίζοντας ένα Led Matrix

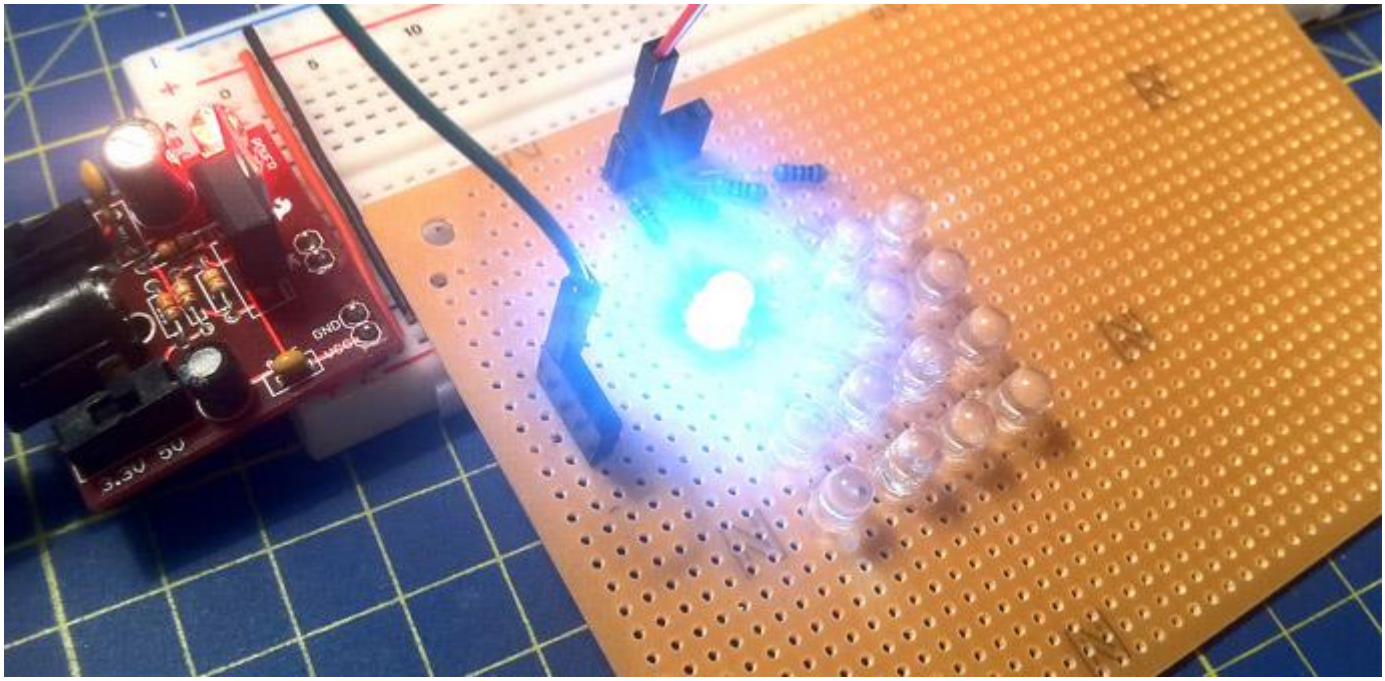
Σε ένα ράστερ ή πλακέτα τοποθετούμε και ευθυγραμμίζουμε τα LED με τον επιθυμητό αριθμό. 4X4 8X6 κτλ. Αναλόγως συνδέουμε τις γειώσεις σε της γραμμής ή την τάση αν έχουμε κοινής τάσης LED, και βάζουμε αντιστάσεις για τη κάθε στήλη LED. Η τιμή της αντίστασής είναι ανάλογη με τι προτείνει ο κατασκευαστής.



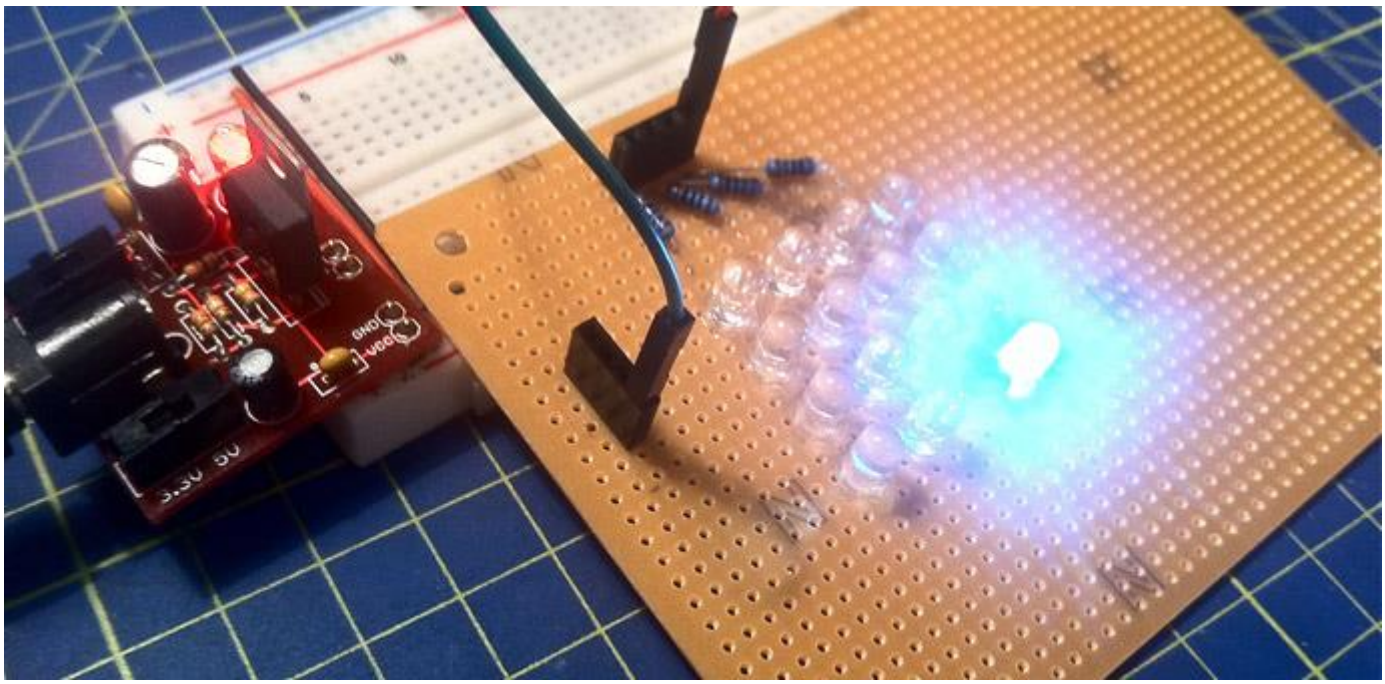
Ανάβοντας 1 LED

Δίνοντας τάση και γείωση στην αντίστοιχη στήλη και σειρά με βάση τις συντεταγμένες του matrix ανάβει ένα led που θέλουμε.

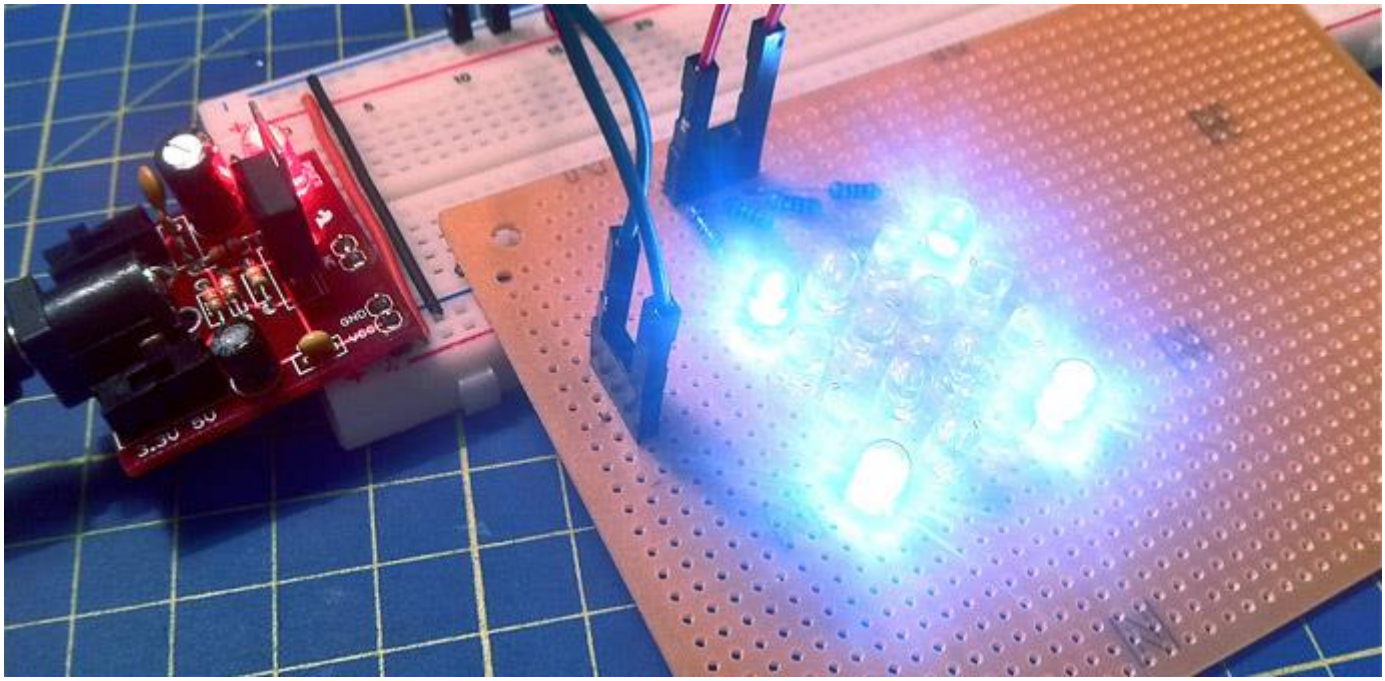
Σύνδεση γείωσης με τη σειρά A και θετική τάση στην στήλη 1 θα ανάψει το πάνω δεξιά LED (A, 1).



Σύνδεση γείωσης με τη σειρά D και θετική τάση στην στήλη 4 θα ανάψει τη βάση προς τα κάτω LED (Δ, 4).



ΠΧ διαλέγουμε τον φωτισμό του τόσο (Α, 1) και (Δ, 4) την ίδια στιγμή είναι απλά συνδέει όλα τα τέσσερα καλώδια. Αυτό δεν είναι η περίπτωση. Υπάρχουν τέσσερις λυχνίες LED που είναι αναμμένη Αυτό συμβαίνει επειδή το ρεύμα επίσης ρέει διαμέσου (Α, 4) και (Δ, 1).



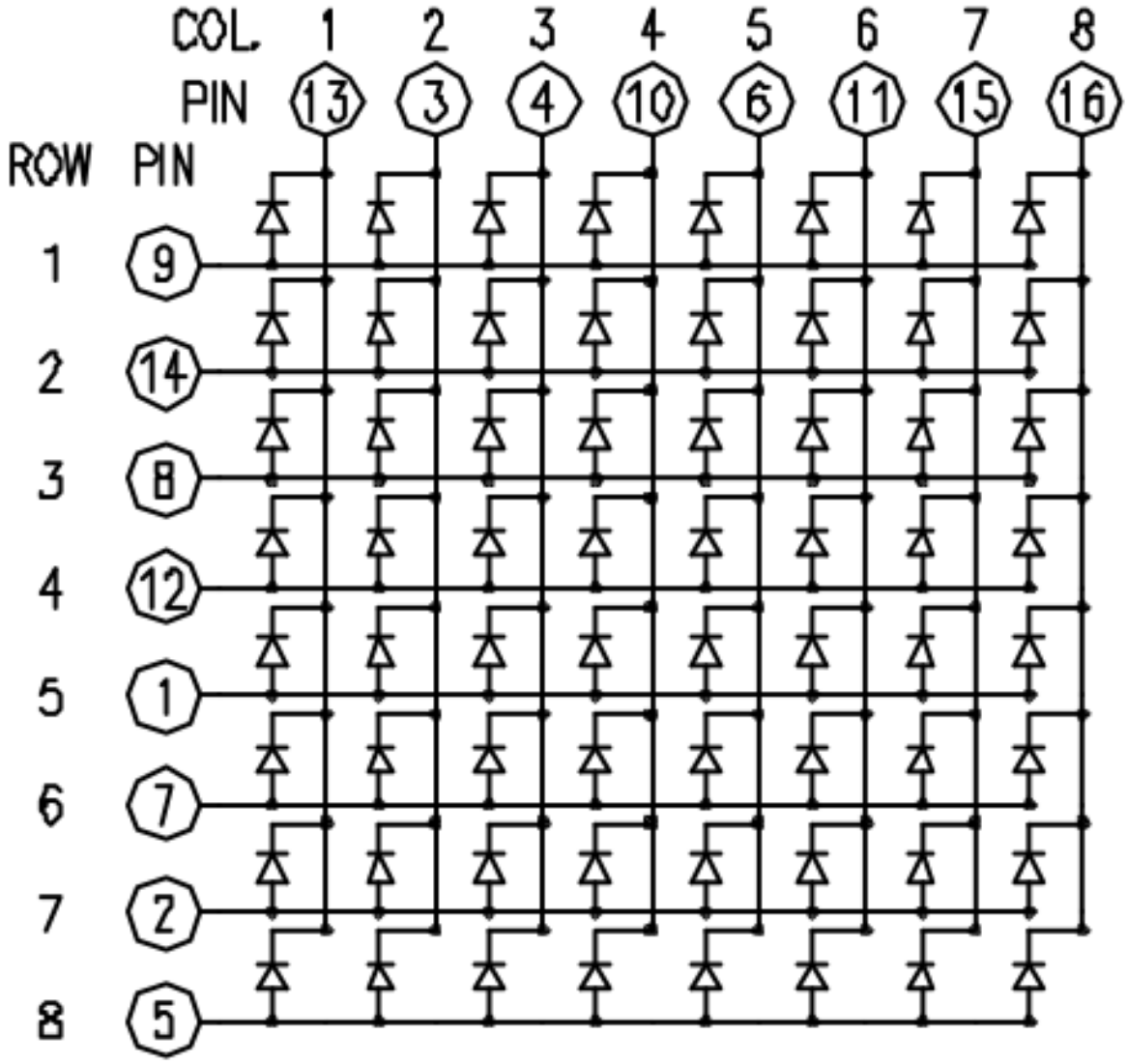
Πολυπλεξία

Πολυπλεξία μπορεί να χρησιμοποιηθεί για την εμφάνιση αυθαίρετες μοτίβα με LED matrix, μερικές φορές ονομάζεται επίσης σάρωση. Σαρώνει τις σειρές (συνήθως από πάνω προς τα κάτω) και τα φώτα που απαιτούνται leds μόνο σε μία γραμμή στο χρόνο του ρολογιού. Και αναλόγως με την ταχύτητα που γίνετε φαίνονται τα σχήματα, γιατί το ανθρώπινο μάτι δεν έχει την ικανότητα να παρατηρήσει την πολύ γρήγορη εναλλαγή από ανοικτό σε κλειστό των ψηφίων σε ένα LED matrix.

Πως γίνετε η σειρά-στήλη σάρωσης για τον έλεγχο ενός 8x8 LED Matrix με τον Arduino

Οι οθόνες LED συνήθως συσκευάζονται ως LED matrix που είναι τοποθετημένα σε σειρές των κοινών ανόδους και τις στήλες των κοινών καθόδων, ή το αντίστροφο. Εδώ είναι ένα χαρακτηριστικό παράδειγμα, και σχηματική του:

Σχήμα 1



Για να ελέγξετε ένα led matrix σαν το παραπάνω συνδέουμε τις σειρές και τις στήλες στο μικροελεγκτή (σχηματικό θα το δείτε παρακάτω) . Οι στήλες συνδέονται με τις καθόδους των LEDs (βλέπε Σχήμα 1) . Οι σειρές που συνδέονται με τις ανόδους των LEDs. Εάν η γραμμή και η στήλη είναι υψηλή ή είναι πολύ χαμηλή η τάση, τα LED και δεν ενεργοποιούνται. Χρειάζεται οι σειρές να έχουν υψηλή τάση και οι στήλες χαμηλή για να ανάψουν τα LED.

Για να ελέγχει ένα μεμονωμένο LED, μπορούμε να ορίσουμε στήλη της υψηλής και της χαμηλής γραμμής. Για τον έλεγχο πολλαπλών LEDs σε μια σειρά, μπορείτε να ορίσετε τις γραμμές υψηλής, στη συνέχεια να λάβει τη στήλη υψηλή, τότε που η σειρά χαμηλά ή υψηλά ανάλογα με την περίπτωση, μια μικρή σειρά θα γίνει η αντίστοιχη λυχνία LED και μια μεγάλη σειρά θα την απενεργοποιήσετε.

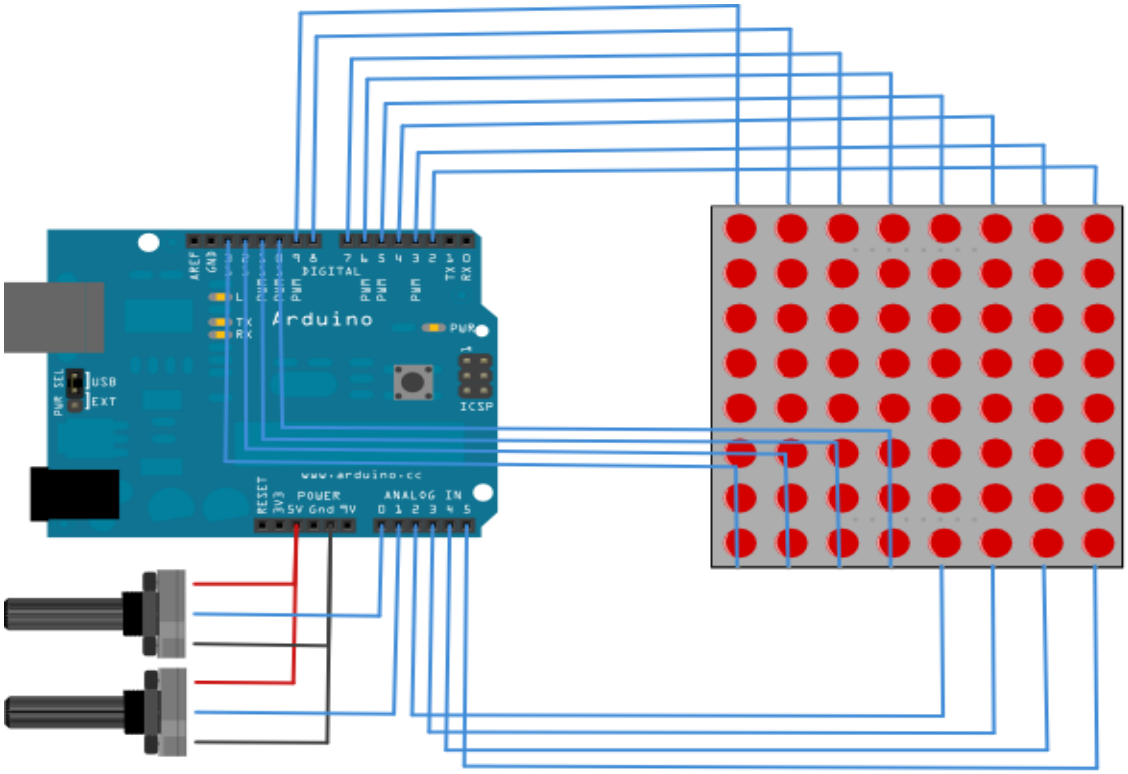
Δεν έχει σημασία ποια ακίδες του μικροελεγκτή να συνδέσετε τις γραμμές και τις στήλες, επειδή μπορούμε να το διαλέξουμε μέσω του προγράμματος του arduino.Συνήθως συνδέουμε τις ακίδες με τρόπο που καθιστά ευκολότερη καλωδίωση. Μία τυπική διάταξη παρουσιάζεται παρακάτω σχήμα 2.

Εδώ είναι οι συνδέσεις του led matrix στις ακίδες του arduino, με βάση το παραπάνω διάγραμμα, σχήμα 3:

Οι 16 ακίδες του matrix συνδέονται τα 16 pins του Arduino. Τέσσερις από τις αναλογικές χρησιμοποιούνται ως ψηφιακές είσοδοι και τα pins 16 έως 19 εκχωρούνται. Η σειρά των pins ορίζονται σε δύο συστοιχίες του κώδικα.

Δύο ποτενσιόμετρα, που συνδέεται με την αναλογική είσοδό pins 0 και 1, ελέγχουν την κίνηση των αναμμένων LED στο matrix.

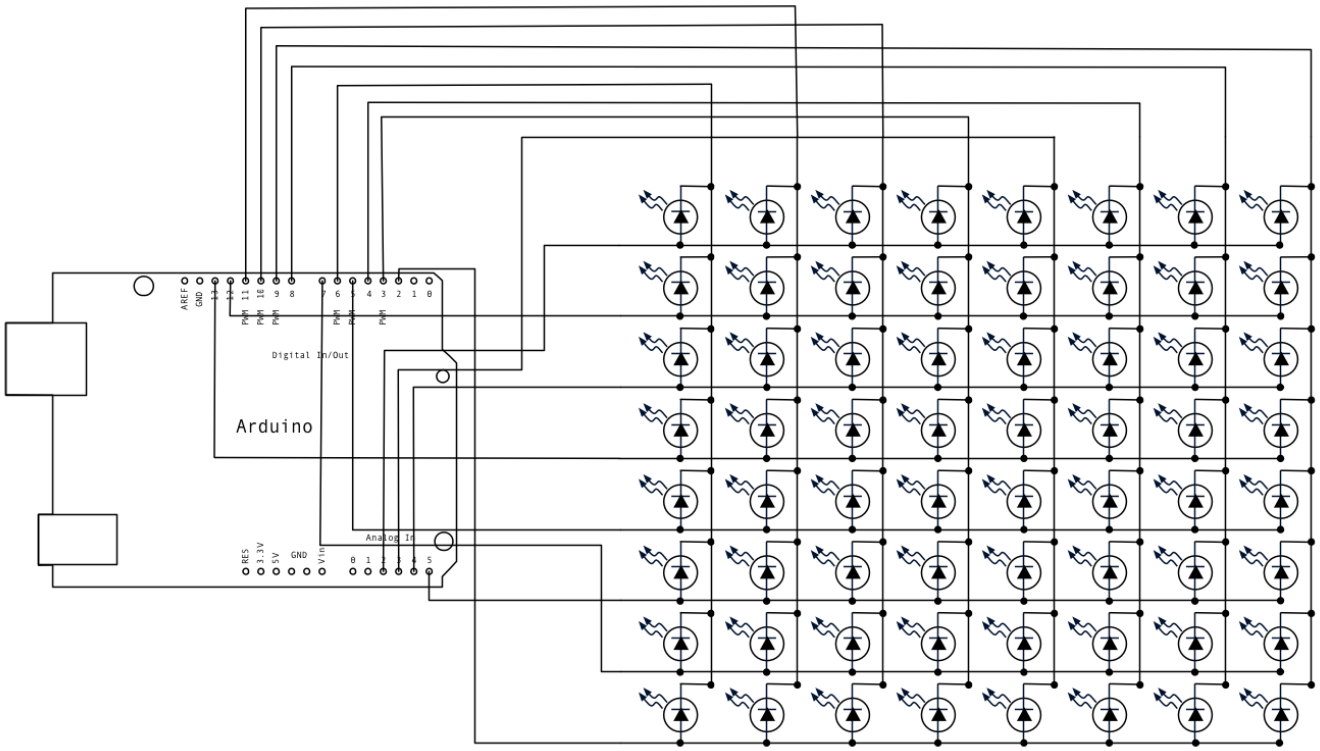
Σχήμα 2



Σχήμα 3

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

Το σχηματικό της σύνδεσης του controller με τα led.



Ο ΚΩΔΙΚΑΣ

```
/* Σειρά-στήλη σάρωση ενός 8X8 LED matrix με X-Y εισόδους.

 Αυτό το παράδειγμα δείχνει πώς ελέγχουμε ένα led matrix χρησιμοποιώντας
2 αναλογικές εισόδους.

Οι σειρές είναι η άνοδος
Οι στήλες είναι η κάθοδος

 ---------

 Arduino Pin numbers:
 Matrix:
 * Ψηφιακά pins 2 με 13,
 * Αναλογικά pins 2 με 5, χρησιμοποιούνται σαν ψηφιακές 16 με 19
 Ποτενσιόμετρα:
 * Οι κεντρικές ακίδες συνδέονται στις αναλογικές εισόδους pins 0 και 1,
αντιστοίχως
 * Οι πλαϊνές ακίδες συνδέονται στην +5V τάση και στην γείωση,
αντιστοίχως.
  */
```

```
// 2-D διάταξη των σειρών στα pin του arduino: το [8] δηλώνει τον αριθμό
// των pin που δηλώνουμε με την εντολή const int για σειρές των led:

const int row[8] = {
  2,7,19,5,13,18,12,16 };

// 2-D διάταξη των στηλών στα pin του arduino: το [8] δηλώνει τον αριθμό
// των pin που δηλώνουμε με την εντολή const int για τις στήλες των led:

const int col[8] = {
  6,11,10,3,17,4,8,9  };

// 2-D διάταξη των πίξελ:
int pixels[8][8];

// Η θέση του κέρσορα:
int x = 5;
int y = 5;

// εδώ κάνουμε το setup

void setup() {

  // προετοιμάζει τα I/O pins σαν εξόδους
  // επαναλαμβάνει πάνω στα pins:

  for (int thisPin = 0; thisPin < 8; thisPin++) {

// προετοιμάζει τα pins εξόδου: με την εντολή pinMode, πριν δηλώσαμε
// thisPin είναι το 0:

    pinMode(col[thisPin], OUTPUT);
    pinMode(row[thisPin], OUTPUT);

    // πάρε τα pins των στηλών (πχ. τις καθόδους) στο HIGH για να
    // εξασφαλίσουμε ότι τα LEDs είναι OFF, με την εντολή digitalWrite:

    digitalWrite(col[thisPin], HIGH);
  }

  // προετοίμασε τα πίξελ του matrix:

  for (int x = 0; x < 8; x++) {
    for (int y = 0; y < 8; y++) {
      pixels[x][y] = HIGH;
    }
  }
}
// είναι ο βασικός βρόγχος του προγράμματος

void loop() {
```

```
  // διάβασε τη είσοδο:
  readSensors();

  // σχεδίασε την οθόνη:
  refreshScreen();
}

void readSensors() {
  // σβήσε την τελευταία θέση:
  pixels[x][y] = HIGH;

  // διάβασε τους σενσορές για τιμές των X και Y:

  x = 7 - map(analogRead(A0), 0, 1023, 0, 7);
  y = map(analogRead(A1), 0, 1023, 0, 7);

  // θέσε την νέα θέση των πίξελ για να ανάψει το LED
  // στην επομένη ανανέωση της οθόνης:
  pixels[x][y] = LOW;

}

void refreshScreen() {
  // επαναλαμβάνουμε για τις σειρές (άνοδοι):
  // δηλώνουμε το thisRow = 0
  for (int thisRow = 0; thisRow < 8; thisRow++) {

    // πάρε το Pin της σειράς (άνοδος) σε high:
    digitalWrite(row[thisRow], HIGH);

    // επαναλαμβάνουμε για τις στήλες (κάθοδοι):
    // δηλώνουμε το thisCol = 0
    for (int thisCol = 0; thisCol < 8; thisCol++) {

      // πάρε την κατάσταση του παρόντος πίξελ;
  int thisPixel = pixels[thisRow][thisCol];

      // όταν η σειρά είναι HIGH η στήλη είναι LOW:
      // τα LED που οι τιμές πληρούν τις προϋποθέσεις ανάβουν:
  digitalWrite(col[thisCol], thisPixel);

      // σβήνει τα πίξελ:
      if (thisPixel == LOW) {
        digitalWrite(col[thisCol], HIGH);
      }
    }

  // πάρε τα pin των σειρών στο LOW και σβήσε όλη την σειρά:
  digitalWrite(row[thisRow], LOW);
  }
```

ΚΕΦΑΛΑΙΟ 2: ARDUINO

Γενικά περί του arduino

Το Arduino είναι μια πλατφόρμα προγραμματισμού μικροελεγκτών, κυρίως της ATMEL AVRs, και προσφέρει πολλές ευκολίες για την ανάπτυξη πρωτότυπων κυκλωμάτων και κατασκευών. Στην πλακέτα έχει όλα τα απαραίτητα κυκλώματα για την σωστή λειτουργία του μικροελεγκτή και την παροχή εξτρά λειτουργιών με την προσθήκη modules αλλά και ευέλικτος προγραμματισμός μέσω USB και μίας ανοικτού κώδικα εύχρηστης εφαρμογής προγραμματισμού.

Τα είδη του Arduino

Τα boards ποικίλουν σε δυνατότητες και βασικό στο σχεδιασμό τους είναι ο αρθρωτός τους χαρακτήρας.

Βασικά χαρακτηρίστηκα είναι η σύνδεση USB και ο voltage regulator με είσοδο για μετασχηματιστή AC/DC για να δώσει την σωστή τάση στον μικροελεγκτή. Το μέγεθος τους ποικίλει αναλόγως τι χρειάζεται κα ξεχωριστά για ρομποτικές εφαρμογές και επίσης πολλά εξειδικευμένα Boards.

- Μικροεπεξεργαστές 8bit και 32 bit, ανάλογα στο μοντέλο
- USB ή micro USB θύρα
- Ενσωματωμένη θύρα Ethernet, Wi-Fi, micro SD

Υπάρχουν και πολλά modules που προστίθενται σε πολλά boards άμα θέλουμε ή χρειάζεται.

- Το Arduino GSM shield συνδέει Arduino σας στο διαδίκτυο χρησιμοποιώντας το ασύρματο δίκτυο GPRS. Συνδέστε μια κάρτα SIM από ένα κάλυψη GPRS επιχείρηση δίνει την δυνατότητα για φωνητικές κλήσεις (θα χρειαστεί ένα εξωτερικό κύκλωμα ηχείο και μικρόφωνο) και την αποστολή / λήψη μηνυμάτων SMS.
- Το Arduino Ethernet Shield συνδέει το Arduino στο διαδίκτυο σε λίγα λεπτά. Συνδέει το arduino στο δίκτυό με ένα καλώδιο RJ45.
- Το Arduino Wi-Fi Shield δίνει Wi-Fi δυνατότητες στο arduino.
- Το Arduino USB Host Shield επιτρέπει να συνδεθεί μια συσκευή USB στην πλακέτα Arduino σας. Το Arduino USB Host Shield βασίζεται στην MAX3421E (τεχνικό φυλλάδιο), το οποίο είναι ένας ελεγκτής USB περιφερειακών / υποδοχής που περιέχει την ψηφιακή λογική και αναλογικό κύκλωμα που απαιτούνται για την εφαρμογή ενός full-speed USB περιφερειακών ή πλήρους υποδοχής / χαμηλής ταχύτητας συμβατές με USB προδιαγραφές rev 2.0. Η ασπίδα είναι TinkerKit συμβατό, το οποίο σημαίνει ότι μπορείτε να δημιουργήσετε γρήγορα τα έργα, συνδέοντας μονάδες TinkerKit πάνω στην πλακέτα.

  Οι ακόλουθες κατηγορίες συσκευών που υποστηρίζονται από την ασπίδα:

  HID συσκευές: πληκτρολόγια, ποντίκια, χειριστήρια, κ.λπ.

Ελεγκτές παιχνιδιών: Sony PS3, Nintendo Wii, Xbox360.

USB σε σειριακή μετατροπείς: FTDI, PL-2303, ACM, καθώς και ορισμένα κινητά τηλέφωνα και δέκτες GPS.

ADK-ικανό Android τηλέφωνα και τραπέζια.

Ψηφιακές φωτογραφικές μηχανές: Canon EOS, Powershot, Nikon φωτογραφικές μηχανές DSLR και P & S, καθώς και γενικές PTP.

Μαζικής συσκευές αποθήκευσης: USB sticks, συσκευές ανάγνωσης καρτών μνήμης, εξωτερικούς σκληρούς δίσκους, κλπ

Dongles Bluetooth.

- Επιπλέον παρά πολλά περιφερειακά όπως οθόνες TFT κα.

Arduino UNO Rev.3

Για την πτυχιακή εργασία θα χρησιμοποιήσω το Arduino UNO Rev.3

Είναι ένα board με βάση τον ATmega 328 έχει 14 ψηφιακά Pins εισόδου / εξόδου (εκ των οποίων 6 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), 6 αναλογικές εισόδους, ένα 16 MHz κεραμικό κρύσταλλο, μια σύνδεση USB, μια υποδοχή ρεύματος, μια επικεφαλίδα ICSP, και ένα κουμπί reset. Περιέχει όλα όσα χρειάζονται για να υποστηρίξουν την μικροελεγκτή. Το Uno Atmega16U2 που έχει προγραμματιστεί ως μετατροπέας USB-to-serial.

Σύνοψη

Μικροελεγκτής: ATmega328

Τάση λειτουργίας: 5V

Τάση εισόδου (συνιστώμενη) 7-12V

Τάση εισόδου (όρια) 6-20V

Ψηφιακά I/O Pins 14 (τα 6 έχουν δυνατότητα PWM έξοδό)

Αναλογικές είσοδοι Pins 6

DC ρεύμα ανά I/O Pin 40 mA

DC ρεύμα for 3.3V Pin 50 mA

Flash μνήμη 32 KB (ATmega328) από τα οποία 0.5 KB για τον bootloader

SRAM 2 KB (ATmega328)

EEPROM 1 KB (ATmega328)

Clock Speed 16 MHz

Πάνω όψη

Πίσω όψη

Είσοδοι και έξοδοι

Κάθε ένα από τα 14 ψηφιακά pins για το Uno μπορεί να χρησιμοποιηθεί ως είσοδος ή έξοδος, χρησιμοποιώντας pinMode (), digitalWrite (), και digitalRead () λειτουργίες. Λειτουργούν σε 5 βολτ. Κάθε pin μπορεί να παρέχει ή να λάβει ένα μέγιστο των 40 mA και έχει μια εσωτερική pull-up αντίσταση (αποσυνδεθεί από προεπιλογή) 20-50 kOhms. Επιπλέον, ορισμένοι ακίδες έχουν εξειδικευμένες λειτουργίες:

Serial: 0 (RX) και 1 (TX). Χρησιμοποιείται για τη λήψη (RX) και να μεταδίδει σειριακά δεδομένα (TX) TTL. Αυτές οι ακίδες συνδέονται με τις αντίστοιχες ακίδες του ATmega8U2 USB-to-TTL Serial τσιπ.

Εξωτερικό Διακοπές: 2 και 3 τα Pins μπορούν να διαμορφωθούν για να προκαλέσει μια διακοπή σε μια χαμηλή τιμή, μια άνοδο ή την πτώση άκρη, ή μια αλλαγή στην αξία.

PWM: 3, 5, 6, 9, 10, και 11 Παροχή 8-bit εξόδου PWM με τη λειτουργία analogWrite ().

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Αυτές οι καρφίτσες υποστηρίζουν την επικοινωνία SPI, χρησιμοποιώντας τη βιβλιοθήκη SPI.

LED: 13 Υπάρχει ένα ενσωματωμένο LED που συνδέονται με ψηφιακά pin 13 Όταν η ακίδα είναι HIGH αξία, η ενδεικτική λυχνία είναι αναμμένη, όταν το pin είναι LOW, είναι μακριά.

Το Uno έχει 6 αναλογικές εισόδους, επισημαίνονται A0 μέσω A5, καθένα από τα οποία παρέχει 10 μπιτ του ψηφίσματος (δηλαδή 1024 διαφορετικές τιμές). Από προεπιλογή μετρούν από το έδαφος έως 5 βολτ, αν και είναι δυνατόν να αλλάξει το άνω άκρο του εύρους τους, χρησιμοποιώντας το pin AREF και τη λειτουργία analogReference (). Επιπλέον, μερικές καρφίτσες έχουν εξειδικευμένες λειτουργίες:

TWI: A4 ή SDA pin και A5 ή SCL pin. Υποστήριξη επικοινωνίας TWI χρησιμοποιώντας τη βιβλιοθήκη Wire.

Υπάρχουν μια-δυο άλλα pin στον πίνακα:

AREF. Τάση αναφοράς για τις αναλογικές εισόδους. Χρησιμοποιείται με analogReference ().

Επαναφορά. Φέρτε αυτήν την γραμμή ΧΑΜΗΛΗ για να επαναφέρετε τον μικροελεγκτή. Συνήθως χρησιμοποιείται για να προταθεί ένα κουμπί reset για ασπίδες που μπλοκάρουν το board.

Επικοινωνία

Το Arduino Uno διαθέτει μια σειρά από εγκαταστάσεις για την επικοινωνία με έναν υπολογιστή, ένα άλλο Arduino, ή άλλους μικροελεγκτές. Η ATmega328 παρέχει UART TTL (5V), σειριακή επικοινωνία, η οποία είναι διαθέσιμη σε ψηφιακά pin0 (RX) και 1 (TX). Μια ATmega16U2 στο board αυτό σειριακή επικοινωνία μέσω USB και εμφανίζεται ως μια εικονική θύρα COM στο λογισμικό στον υπολογιστή. Η «firmware 16U2 χρησιμοποιεί τα τυπικά προγράμματα οδήγησης USB COM, και καμία εξωτερική χρειάζεται πρόγραμμα οδήγησης. Ωστόσο, στα Windows, απαιτείται .inf αρχείο. Το λογισμικό Arduino περιλαμβάνει μια σειριακή οθόνη η οποία επιτρέπει την απλή δεδομένα κειμένου που αποστέλλονται από και προς την πλακέτα Arduino. Οι RX και TX LEDs στον πίνακα θα αναβοσβήνει όταν γίνεται μετάδοση δεδομένων μέσω του τσιπ και USB σύνδεσης USB-to-serial στον υπολογιστή (όχι όμως για σειριακή επικοινωνία στις ακίδες 0 και 1).

Μια SoftwareSerial βιβλιοθήκη επιτρέπει την σειριακή επικοινωνία για οποιαδήποτε ψηφιακά Pin του Uno.

Η ATmega328 υποστηρίζει επίσης I2C (TWI) και την επικοινωνία SPI. Το λογισμικό Arduino περιλαμβάνει μια βιβλιοθήκη καλωδίων για να απλοποιήσει τη χρήση του διαύλου I2C? δείτε την τεκμηρίωση για λεπτομέρειες. Για την επικοινωνία SPI, χρησιμοποιήστε τη βιβλιοθήκη SPI.

Προγραμματισμός

Το Arduino Uno μπορεί να προγραμματιστεί με το λογισμικό από το Arduino foundation. Οι ATmega328 στο Arduino Uno έρχεται preburned με ένα bootloader που σας επιτρέπει για προγραμματισμό κωδικού χωρίς τη χρήση ενός εξωτερικού προγραμματιστή υλικού. Επικοινωνεί χρησιμοποιώντας το αρχικό STK500 πρωτόκολλο (αναφορά, αρχεία κεφαλίδας C).

Μπορείτε επίσης να παρακαμφθεί ο bootloader και να προγραμματιστεί το μικροελεγκτή μέσω του ICSP (In-Circuit Serial Programming) header? δείτε αυτές τις οδηγίες για τις λεπτομέρειες.

Η ATmega16U2 / 8U2 είναι φορτωμένο με ένα bootloader DFU, το οποίο μπορεί να ενεργοποιηθεί από:

Σε Rev1 πίνακες: συνδέει το jumper κολλήσεις στο πίσω μέρος του πίνακα (κοντά στο χάρτη της Ιταλίας) και στη συνέχεια επαναφορά του 8U2.

Σε Rev2 ή αργότερα πίνακες: υπάρχει μια αντίσταση που τραβώντας το 8U2 / 16U2 HWB γραμμή με τη γείωση, πράγμα που καθιστά ευκολότερο να τεθεί σε λειτουργία DFU.

Στη συνέχεια μπορείτε να χρησιμοποιηθεί το λογισμικό της Atmel FLIP (Windows) ή τον προγραμματιστή DFU (Mac OS X και Linux) για να φορτωθεί ένα νέο firmware.

ΚΕΦΑΛΑΙΟ 3: Shift Registers και η χρήση του με το arduino

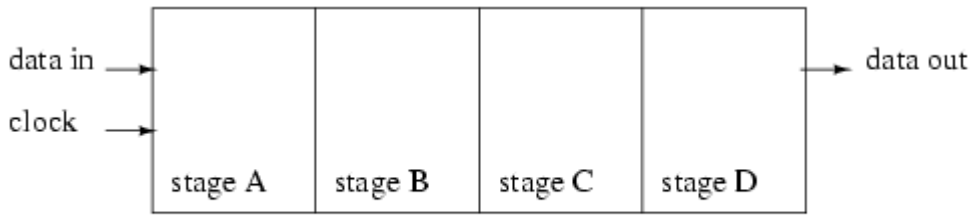
Γενικά

Στα ψηφιακά κυκλώματα, ένας καταχωρητής μετατόπισης (shift register) είναι πολλά flip-flops, τα οποία μοιράζονται το ίδιο ρολόι (clock), στο οποίο η έξοδος κάθε flip-flop συνδέεται με τα "δεδομένα" εισόδου του επόμενου flip-flop στην αλυσίδα, με αποτέλεσμα σε ένα κύκλωμα h μετατοπίσεις κατά μια θέση η «συστοιχία μπιτ» αποθηκεύονται σε αυτό, μετατοπίζοντας τα παρούσα δεδομένα στην είσοδο του και μετατόπιση του τελευταίου bit στη συστοιχία, σε κάθε μετάβαση της εισόδου ρολογιού.

Γενικότερα, ένας καταχωρητής μετατόπισης μπορεί να είναι πολυδιάστατος, έτσι ώστε τα δεδομένα στης εισόδους και εξόδους να είναι οι ίδιες συστοιχίες bit: αυτό απλά με την λειτουργία πολλών καταχωρητών μετατόπισης του ίδιου λίγο-μήκους παράλληλα.

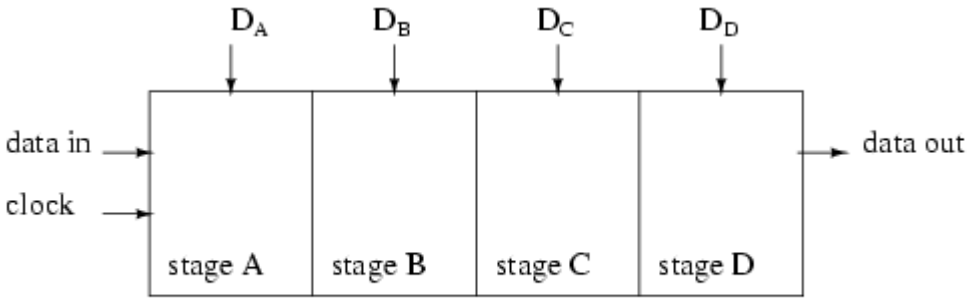
Οι shift registers μπορεί να έχει παράλληλες και σειριακές εισόδους και εξόδους. Αυτά είναι συνήθως ρυθμιστεί ως «serial-in, parrarel-out» (SIPO) ή ως «parrarel-in, serial-out» (PISO). Υπάρχουν επίσης είδη που έχουν τόσο σειριακή και παράλληλη είσοδο και είδη με σειριακή και παράλληλη έξοδο. Υπάρχουν, επίσης, «αμφίδρομοι» καταχωρητές ολίσθησης που επιτρέπουν τη μετατόπιση προς τις δύο κατευθύνσεις: L → R ή R → L. Η σειριακή είσοδο και την τελευταία έξοδο του καταχωρητή ολίσθησης μπορεί επίσης να συνδέεται με τη δημιουργία ενός «καταχωρητή κυκλικής ολίσθησης».

SISO σειριακή είσοδό σειριακή έξοδο



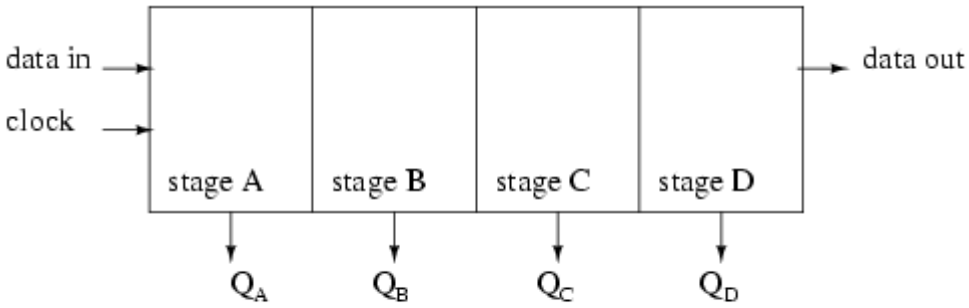
Serial-in, serial-out shift register with 4-stages

PISO παραλλήλη είσοδο σειριακή εξοδό



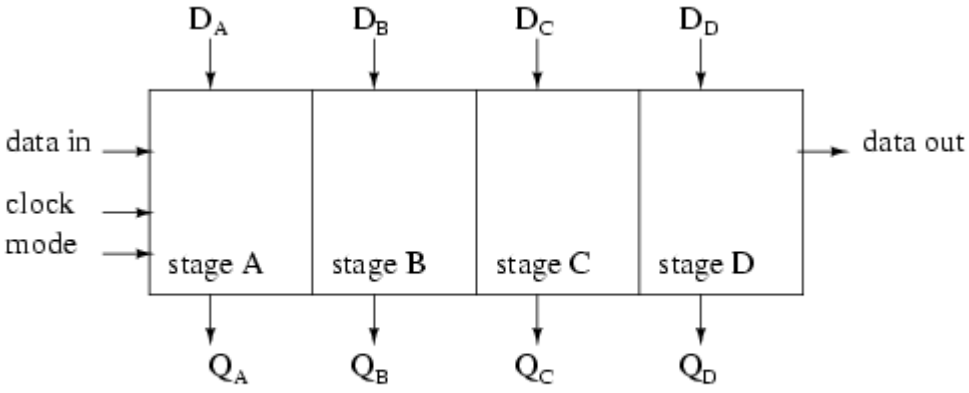
Parallel-in, serial-out shift register with 4-stages

SIPO σειριακή είσοδο παράλληλη έξοδο



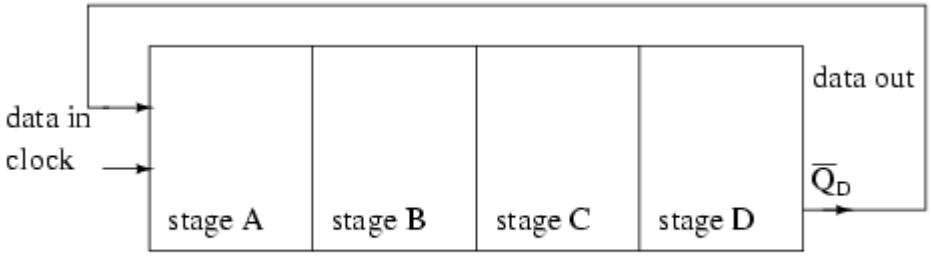
Serial-in, parallel-out shift register with 4-stages

PIPO παράλληλη είσόδο παράλληλη έξοδο



Parallel-in, parallel-out shift register with 4-stages

Καταχωρητής κυκλικης ολίσθησης

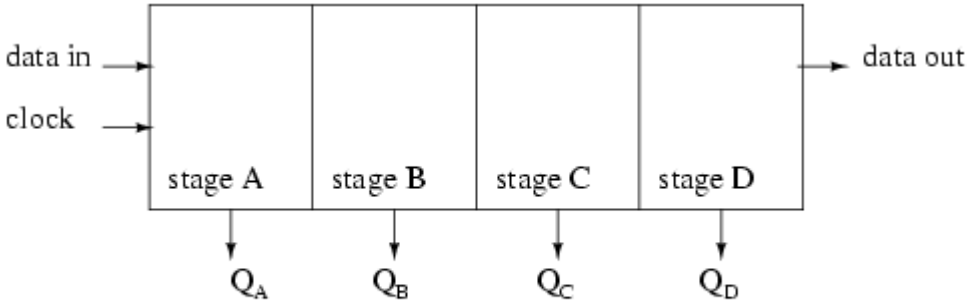


Ring Counter, shift register output fed back to input

Στην πτυχιάκη εργασία χρησιμοποιώ SIPO λογική, σειριακή είσοδος (τα δεδομένα τα παιρνουμέ από το arduino) και παραλλήλη έξοδος στης στήλες του led matrix. Και θα αναφερθώ μονο για αυτού του τύπου καταχωρητών ολήσθησης.

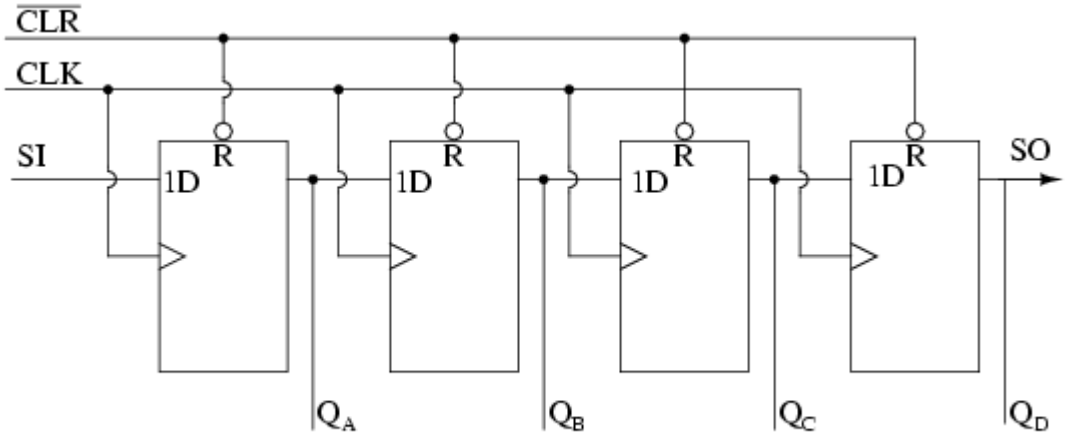
<u>SIPO σειριακή είσοδός – παράλληλη έξοδος σε καταχωρητές ολήσθησης</u>

Ένα serial-in/parallel-out καταχωρητής ολήσθητης, ολησθένει τα δεδομένα σε εσωτερικά στοιχεία αποθήκευσης δεδομένων και μετατοπιζει δεδομένα από τον serial-out, τα data-out, pin. Στο SIPO κάνει όλα τα εσωτερικά στάδια που διατίθενται ως έξοδοι. Ως εκ τούτου, ένα serial-in/parallel-out shift register μετατρέπει τα δεδομένα από τη σειριακή μορφή σε παράλληλη μορφή. Αν τα τέσσερα bits δεδομένων μετατοπίζονται σε τέσσερις παλμούς ρολογιού μέσω ενός μόνο καλωδίου σε δεδομένα-in, παρακάτω, είναι διαθέσιμο ταυτόχρονα σε τέσσερις εξόδους QA να QD μετά το τέταρτο παλμό ρολογιού τα δεδομένα.



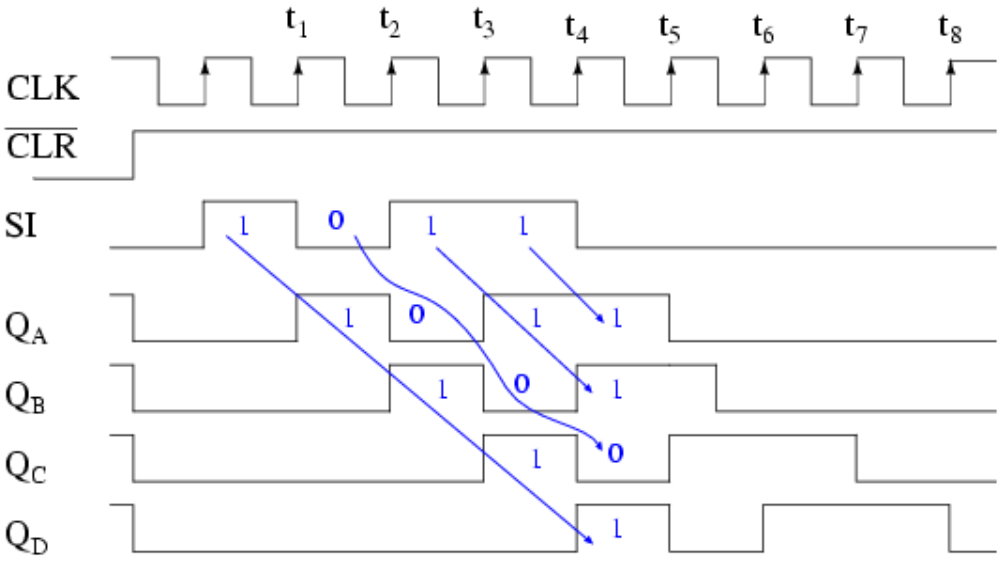
Serial-in, parallel-out shift register with 4-stages

Η πρακτική εφαρμογή των SIPO καταχωρητών ολίσθησης είναι το να μετατρέπουν τα δεδομένα από σειριακή μορφή από ένα και μόνο καλώδιο σε παράλληλη μορφή σε πολλαπλά καλώδια. Πολύ χρήσιμο για την έλεγχο μεγάλων LED matrix που δεν μπορεί να γίνει απ' ευθείας από τα Pins του μικροελεγκτή με τις εξόδους του shift register (QA QB QC QD).



Serial-in/ Parallel out shift register details

Τα παραπάνω στοιχεία του SIPO είναι αρκετά απλή. Μοιάζει με ένα serial-in/ serial-out shift register αλλά παιρνουμέ της εξόδους καθέ σταδίου της ολίσθησης. Τα σειριακά δεδομένα μετατοπίζοται μέσα σε SI (Serial Input). Μετά από μια σειρά από ρολόγια ίσο με τον αριθμό των σταδίων, το πρώτο bit δεδομένων σε εμφανίζεται στο SO (QD) στο παραπάνω σχήμα. Σε γενικές γραμμές, δεν υπάρχει SO pin. Το τελευταίο στάδιο (QD παραπάνω) χρησιμεύει ως SO και κλιμακώνεται προς το επόμενο πακέτο αν υπάρχει. Αυτό είναι χρησιμό για να δουλεύουμε πολλά μαζι στη σείρα για μεγαλύτερα bits δεδόμένων να μεταφράζονται από την σειρακή μορφή σε παράλληλη.

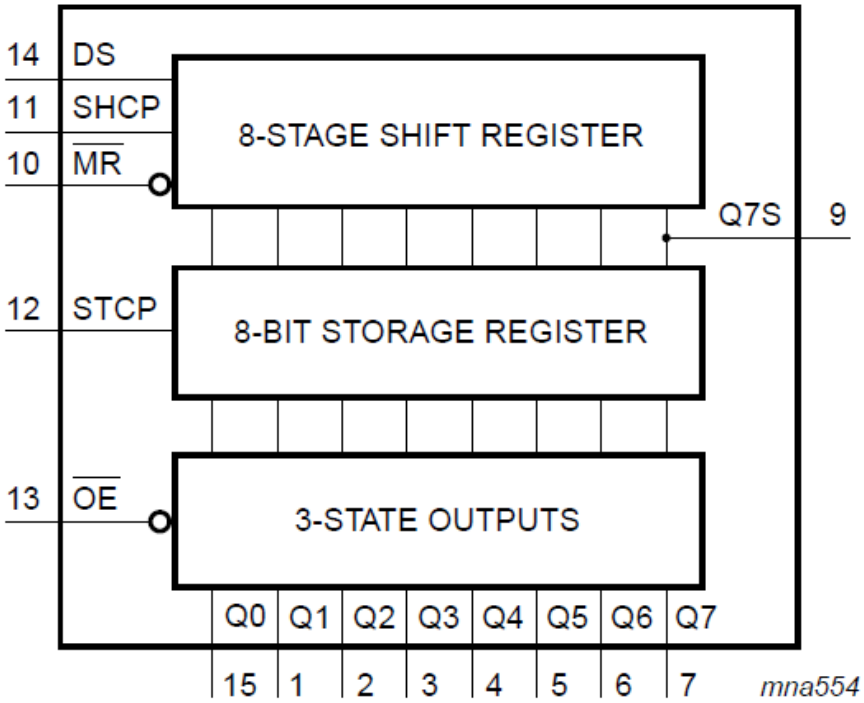


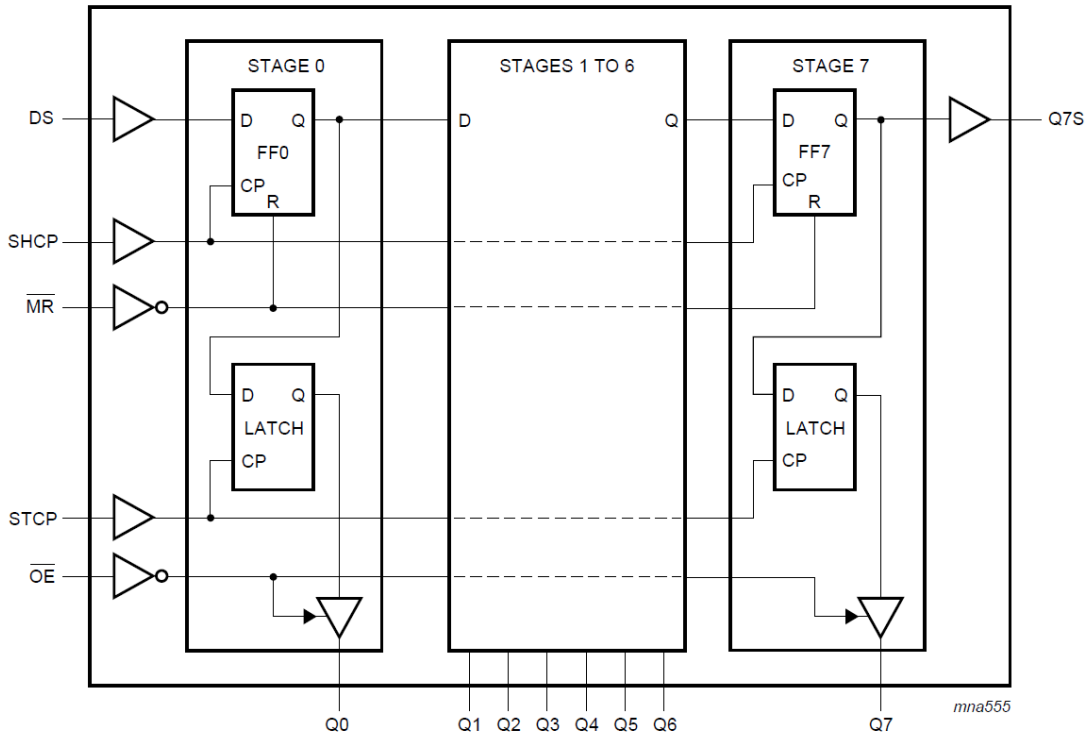
Serial-in/ parallel-out shift register waveforms

Ο καταχωρητής ολίσθησης έχει πρότινος καθαριστεί από τα δεδομένα από το CLR, ένα ενεργό low σήμα, το οποίο καθαρίζει όλα το τυπού D flip-flops μεσά στον καταχωρητή ολίσθησης. Σημείωστε ότι τα σειριακά δεδομένα 1011 τα οποία είναι παρόν στη SI είσοδο. Αυτά τα δεδομένα συγχρονίζονται με το ρολοι (CLOCK) CLK. Στον 1^o κτύπο το 1^o bit πάει στο $\mathbf{Q_A}$ και στο 2^o το 2^o bit στο $\mathbf{Q_A}$,ενώ το το 1^o πάει στο $\mathbf{Q_B}$ και ούτω καθεξής μέχρι όλα τα bit ολισθήσουν. Στην PO λογική μπορούμε να πάρουμε την έξοδο για κάθε στάδιο της ολίσθησης και έτσι να διευθύνουμε στο LED Matrix πως θα ανάβουν τα LED ανά χτύπο ρολογιού.

74HC595 shift register

Στο κύκλωμα έλεγχου του LED matrix της πτυχιακής χρησιμοποιώ τον 74hc595 8bit serial-in parrarel out, για να δίνω τα δεδομένα σε κάθε στήλη του matrix.



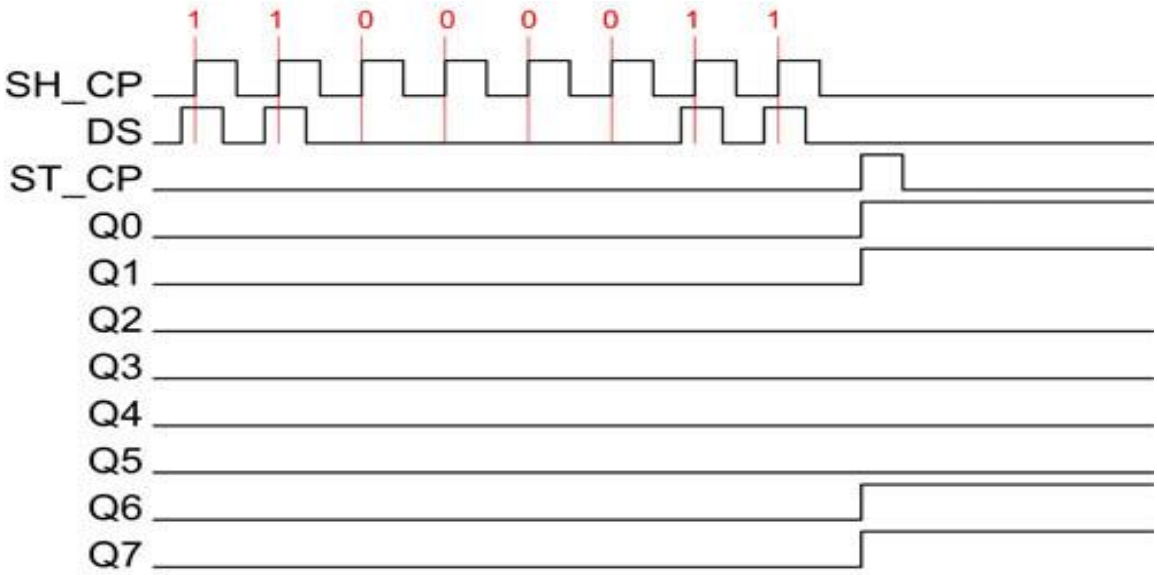


Τα παραπάνω διαγράματα δείχνουν την λειτουργεία του 595 το λείτουργικο και λογικό διαγραμμά είναι 8bit

Τα δεδομένα μπαίνουν από το 14 pin DS, όταν το Pin 11 SHCP πάει από HIGH σε LOW η τιμή το DS αποθηκεύετε στον shift register και οι υπάρχουσες τιμές του καταχωρητή ολίσθενουν για να κάνουν χώρο για το νέο μπιτ. Το Pin 12 STCP παραμένει LOW όταν τα δεδομένα γράφονται στον

shift register. Όταν πάει στο HIGH οι τιμές του shift register ασφαλίζονται στους καταχωρητές αποθύκευσης τα οποία μετά εξάγονται στα Pins Q0 – Q7.

Το παρακάτω διαγραμμά δείχνει πως παιρνουμέ σαν εξοδό στα Qo-Q7 1100011 υποθέτοντας ότι οι αρχικες τιμές είναι 00000000. Όταν συνδέετε το Pin 9 Q7S είναι σειριακή εξοδό, και με την χρησή του στην είσοδο DS ενός 2 595 για να εχουμε παραπάνω στήλες στο matrix και με κάθε shift register που βάζουμε, στο 24X8 θέλουμε 3 για μεγαλύτερα απλά βαζουμε παραπάνω και κάθε ένα προσθέτει μεχρί 8 εξόδους.



<u>Σειριακή σε Παράλληλη Shifting-Out με ένα 74HC595 με το arduino</u>

Shifting-Out και το τσιπ 595

Σε κάποια στιγμή ή την άλλη μπορεί να ξεμείνουμε από εξόδους στο Arduino και μπορούμε να το επεκτείνουμε με καταχωρητές ολίσθησης. Αυτό το παράδειγμα βασίζεται στην 74HC595. Μπορεί να το χρησιμοποιηθεί για να ελέγχει 8 εξόδους σε ένα χρόνο, ενώ χρησιμοποιούμε λίγα Pins στο μικροελεγκτή. Μπορείτε να συνδέσετε πολλαπλούς καταχωρητές ολίσθησης μαζί για να επεκτείνουν έξοδο σας ακόμα περισσότερο.

Αυτό που χρησιμοποιούνται στα σχέδια που είναι μέσα, ονομάζεται «σύγχρονη σειριακή επικοινωνία», δηλαδή μπορείτε να πάλλεται ένα pin up και κάτω επικοινωνεί με αυτόν τον τρόπο ένα byte δεδομένων στον καταχωρητή λίγο-λίγο. Είναι από πάλλεται δεύτερο pin, το pin του ρολογιού, ώστε να οριοθετηθούν μεταξύ bits. Αυτό είναι σε αντίθεση με τη χρήση της "ασύγχρονης σειριακής επικοινωνίας» της συνάρτησης Serial.begin (), η οποία στηρίζεται επί του αποστολέα και του δέκτη για να ρυθμιστεί ανεξάρτητα για μια συμφωνηθεί καθορισμένο ρυθμό δεδομένων. Μόλις το σύνολο byte μεταδίδεται στον καταχωρητή τα HIGH ή LOW μηνύματα που πραγματοποιήθηκαν κρατιούνται και διαμοιράζονται σε καθένα από τα επιμέρους ακροδέκτες εξόδου. Αυτό είναι το μέρος "παράλληλη έξοδο», με όλα τα pins να κάνουν ό, τι θέλετε με τη μία.

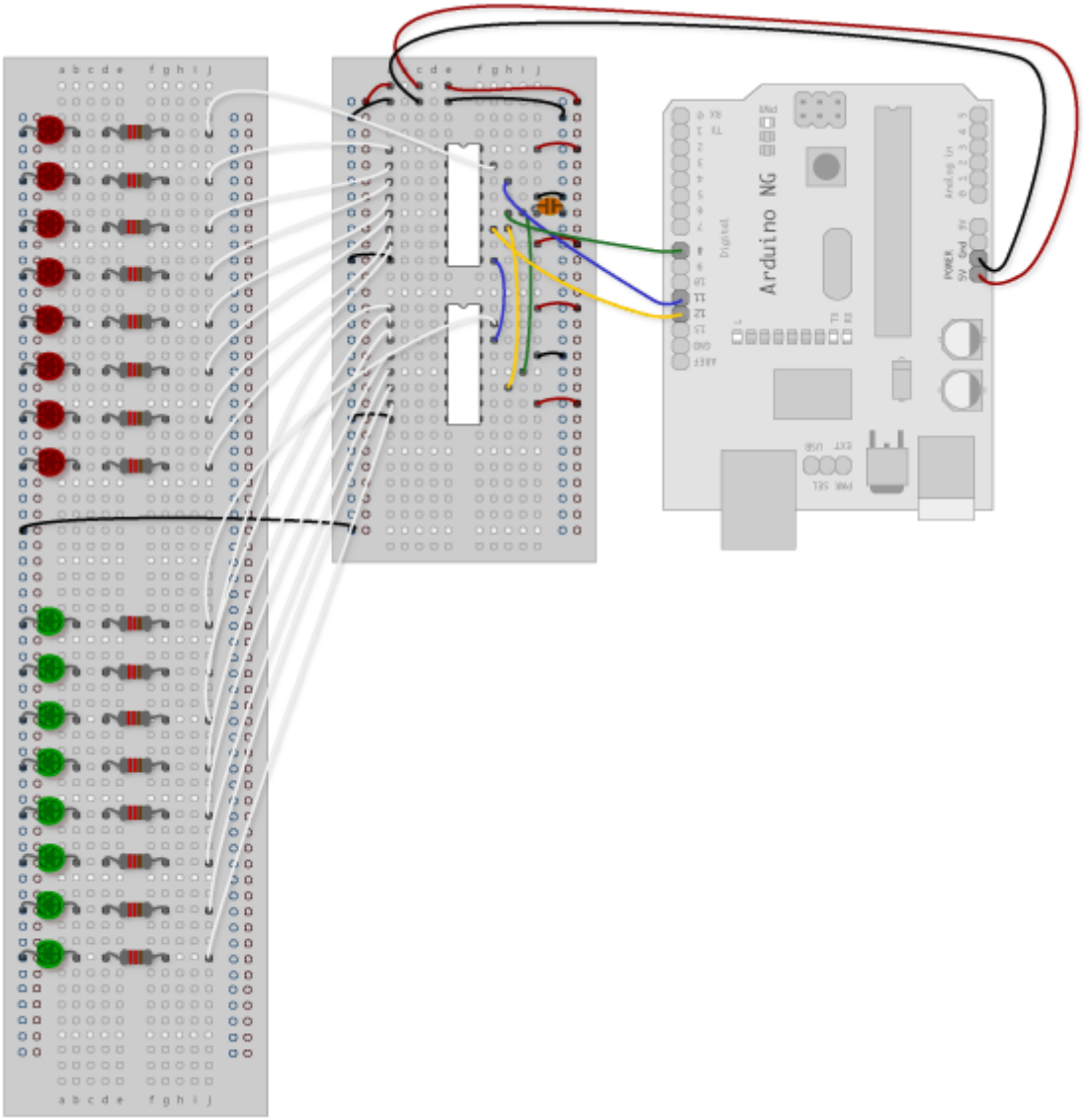
Το μέρος αυτού του στοιχείου «σειριακή έξοδο» προέρχεται από επιπλέον pin της, η οποία μπορεί να περάσει το σειριακό πληροφορίες που έλαβε από τον μικροελεγκτή και πάλι αμετάβλητα. Αυτό

σημαίνει ότι μπορείτε να μεταδίδει 16 bits σε μια σειρά (2 bytes) και το πρώτο 8 θα ρέει μέσω του πρώτου καταχωρητή στο δεύτερο καταχωρητή και να εκφράζονται εκεί.

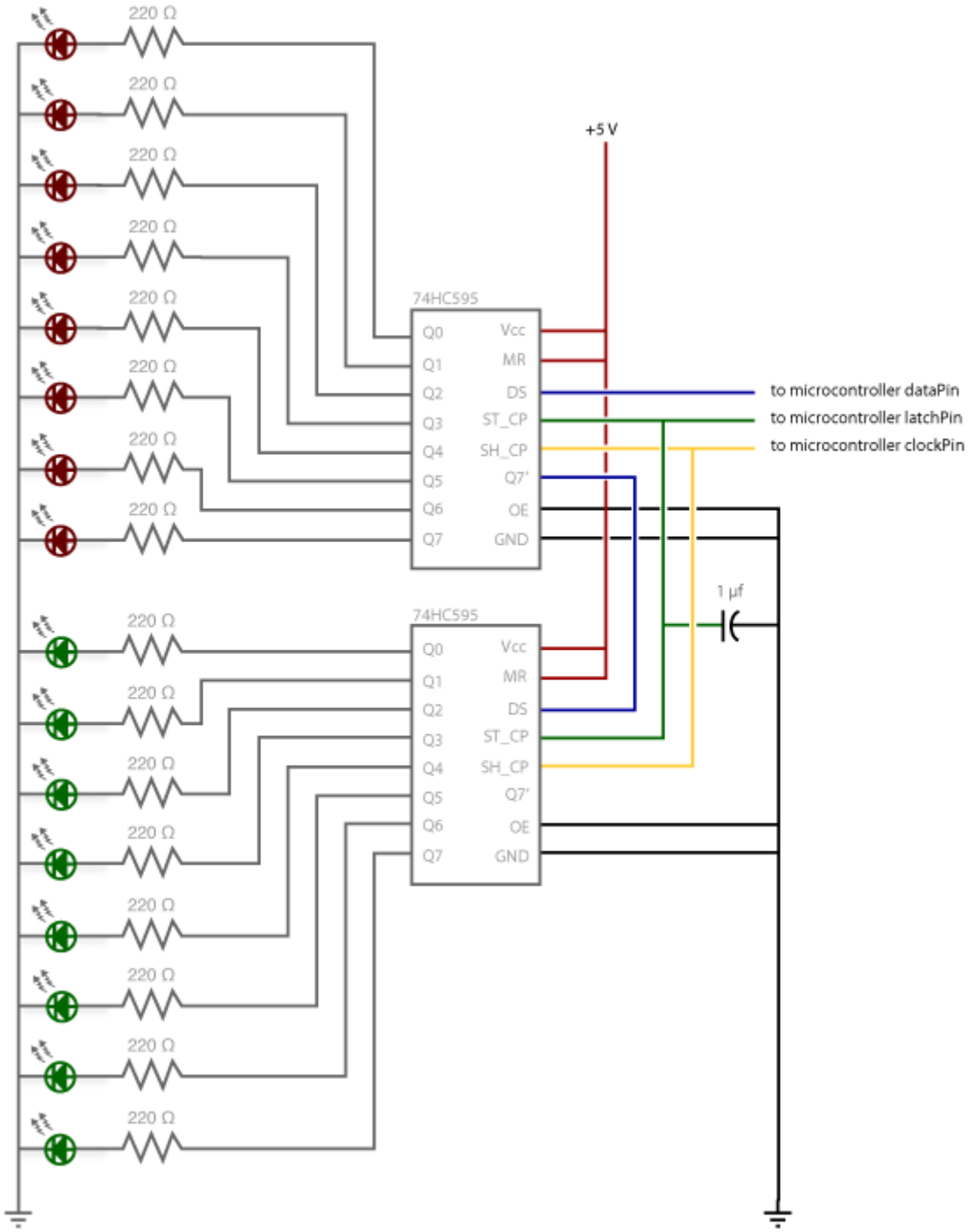
«3 states» αναφέρεται στο γεγονός ότι μπορείτε να ορίσετε τις ακίδες εξόδου είτε ως υψηλή, χαμηλή ή «υψηλή αντίσταση." .. Σε αντίθεση με τα HIGH και LOW καταστάσεις, μπορείτε να "t σετ pins σε κατάσταση υψηλής αντίστασης τους χωριστά. Μπορούμε να ρυθμίσουμε μόνο το σύνολο τσιπ μαζί Αυτό είναι ένα πολύ εξειδικευμένο πράγμα που πρέπει να κάνετε – πχ. μια συστοιχία LED που μπορεί να χρειαστεί να ελέγχονται από εντελώς διαφορετικές μικροελεγκτές, ανάλογα με μια συγκεκριμένη ρύθμιση λειτουργίας ενσωματωμένο στο σχέδιο.

Παράδειγμα με 2 shift registers και τον arduino

Στο breadboard



Το κύκλωμα



Το κυκλώμα του παραδείγματος, έχουμε 2 shift registers 74HC595 και τον μικροέλεγκτη board arduino στις εξόδους των 595 συνδέουμε leds. Αρχικά συνδέουμε τον 1^{ο} 595 στο datapin (DS) του arduino και τα STCP,SHCP στο latchpin και clockpin του arduino, σημειωσή τα pins τα ορίζουμε στο προγραμμά και δεν έχει σημασία ποια είναι αλλά εδώ χρησιμοποιουμε τα 8,11 και 12 του arduino. Βάζουμε της αντιστάσεις στα LED και 1μF στο latchpin και την γείωση για να αποφύγουμε το τρεμοσβημά όταν παλετέ το latchpin, Το MR το βαζουμε στην τάση και το OE στην γείωση για να απενεργοποιησουμέ, και φυσικά το Pin 16 στην τασή και 8 στην γείωση που προσφέρει ο arduino. Οι καθοδοί των led γειώνονται.

Παράδειγμά διπλός δυαδικός μετρητής

```
//  Κώδικας χρησιμοποιώντας 74HC595 Shift Register          //
//          για να μετράει από το 0 έως το 255                  //

//Pin  που συνδέετε στο ST_CP του 74HC595
int latchPin = 8;
//Pin που συνδέετε στο SH_CP του 74HC595
int clockPin = 12;
////Pin που συνδέετε στο DS του 74HC595
int dataPin = 11;



void setup() {
  //Start Serial for debugging purposes
  Serial.begin(9600);
  //θέτει τα pins στην έξοδο γιατί απευθύνονται και βασικό βρόγχο
  pinMode(latchPin, OUTPUT);

}

void loop() {
  //ρουτίνα μετρητή προς τα πάνω
  for (int j = 0; j < 256; j++) {

 //γειώνει το latchPin και το κρατάει LOW όσο μεταδίδουμε
    digitalWrite(latchPin, 0);

 //μετρητής προς τα πάνω στα πράσινα LEDs
    shiftOut(dataPin, clockPin, j);
    //μετρητής προς τα κάτω στα κόκκινα LEDs
    shiftOut(dataPin, clockPin, 255-j);

    //επέστρεψε το latch pin σε high για να δώσουμε σήμα στο τσιπ για μην διαβάζει την πληροφορία
    digitalWrite(latchPin, 1);
    delay(1000);
  }
}

void shiftOut(int myDataPin, int myClockPin, byte myDataOut) {
  // Αυτό ολισθύνει 8 bits out MSB first,
  // σχετικά με την ανερχόμενη ακμή του ρολογιού,
  // ρολόι idles low

..//εσωτερική λειτουργία setup
  int i=0;
  int pinState;
  pinMode(myClockPin, OUTPUT);
  pinMode(myDataPin, OUTPUT);
```

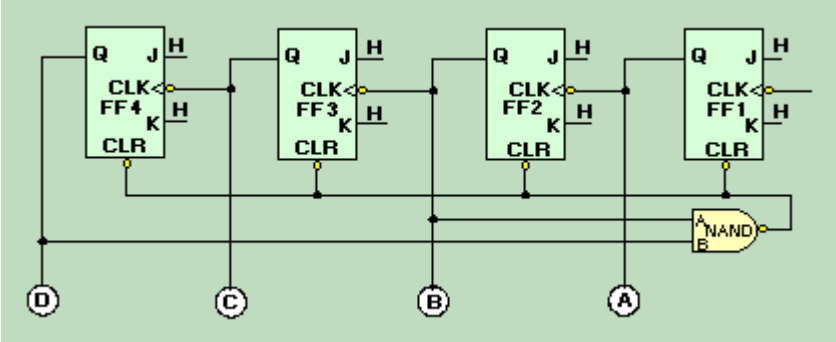
```
. //καθάρισε τα πάντα
. //προετοίμασε τον shift register για bit shifting
  digitalWrite(myDataPin, 0);
  digitalWrite(myClockPin, 0);

  //για κάθε μπιτ στο byte myDataOut
  //Μετράμε προς τα κάτω στον βρόγχο
  //αυτό σημαίνει ότι %00000001 or "1" will go through such
  //αυτό θα είναι το pin Q0 το όποιο ανάβει.
  for (i=7; i>=0; i--)  {
    digitalWrite(myClockPin, 0);

    // αν η τιμή περάσει το myDataOut και ένα αποτέλεσμα δυαδικής μάσκας
    // αληθινό τότε... οπότε αν είμαστε σε i=6 και η τιμή μας είναι
    // %11010100 θα ήταν ο κωδικός συγκρίνεται με %01000000
    // και  προχωρεί για να ρυθμίσετε pinState to 1.
    if ( myDataOut & (1<<i) ) {
      pinState= 1;
    }
    else {
      pinState= 0;
    }

    //Θέτει toPin σε HIGH ή LOW εξαρτώμενο στο pinState
    digitalWrite(myDataPin, pinState);
    //register shifts bits on upstroke of clock pin
    digitalWrite(myClockPin, 1);
    //μηδένισε τα δεδομένα στο data pin μετά την ολίσθηση για να προλαμβάνει το bleed through
    digitalWrite(myDataPin, 0);
  }

  //σταματάει την ολίσθηση
  digitalWrite(myClockPin, 0);
}
```

ΚΕΦΑΛΑΙΟ 4: Decade counter

Γενικα

Ένας δεκαδικός μετρητης είναι ένα δυαδικός μετρητός ο οποιός είναι σχεδιασμένος να μετράει μέχρι 1010. Ένας συνηθησμένος 4-stage μετρητής μπορεί ευκόλα να γίνει ενάς μετρητής προσθετοντας μια NAND πύλη οπώς φαίνετε στο σχήμα 1. Προσεξτέ ότι τα FF2 και FF4 δίνουν τις εισόδους για την NAND πύλη. Η έξοδος της NAND πύλης συνδεετε με την CLR εισοδο κάθε ένα από τα flip-flops.

Σχήμα 1. - Decade counter.



Ο μετρητής λειτουργεί σαν κανονικός μετρητής μεχρι να φτάσει στο 10 δηλαδή 1010 στο δυαδικό. Σε αυτό το σημείο, και οι δύο είσοδοι της NAND πύλης είναι HIGH, και η έξοδος παεί LOW. Αυτό το LOW εφαρμόζετε στα CLR των FFs και τα αναγκάζει να πανε στο 0. Όταν τα FFs επανεκινήσουν το μέτρημα μπορεί να ξαναξεκινήσει .

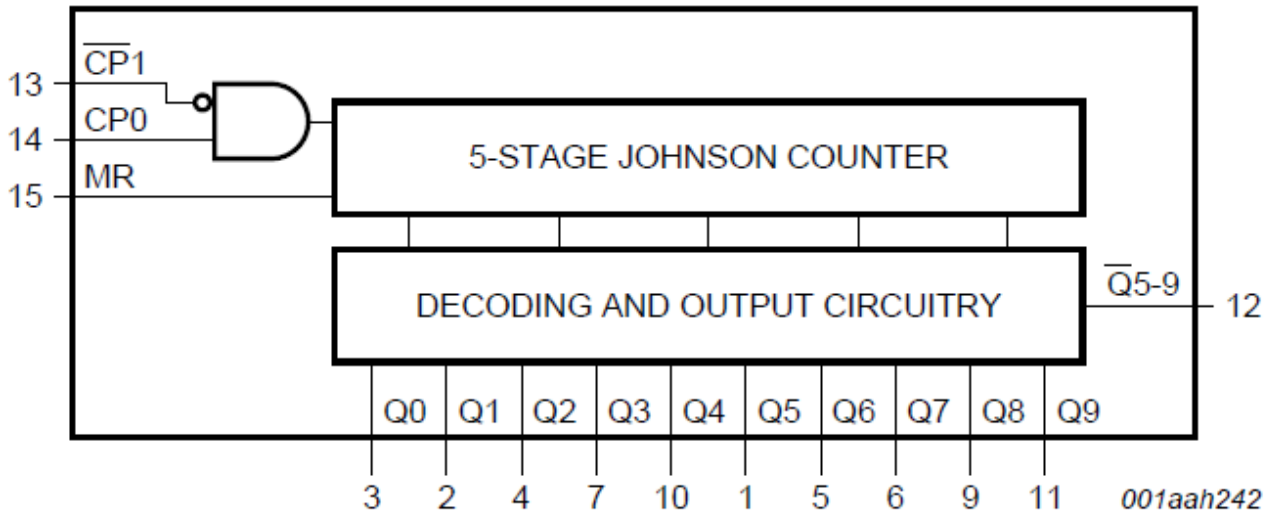
Ο παρακάτω πίνακας δέιχνεί την δυαδική μέτρηση και της είσοδους και εξόδο της NAND:

BINARY COUNT	NAND GATE INPUTS		NAND GATE OUTPUT
*******	A	B	*******
0000	0	0	1
0001	0	0	1
0010	1	0	1
0011	1	0	1
0100	0	0	1
0101	0	0	1
0110	1	0	1
0111	1	0	1
1000	0	1	1
1001	0	1	1

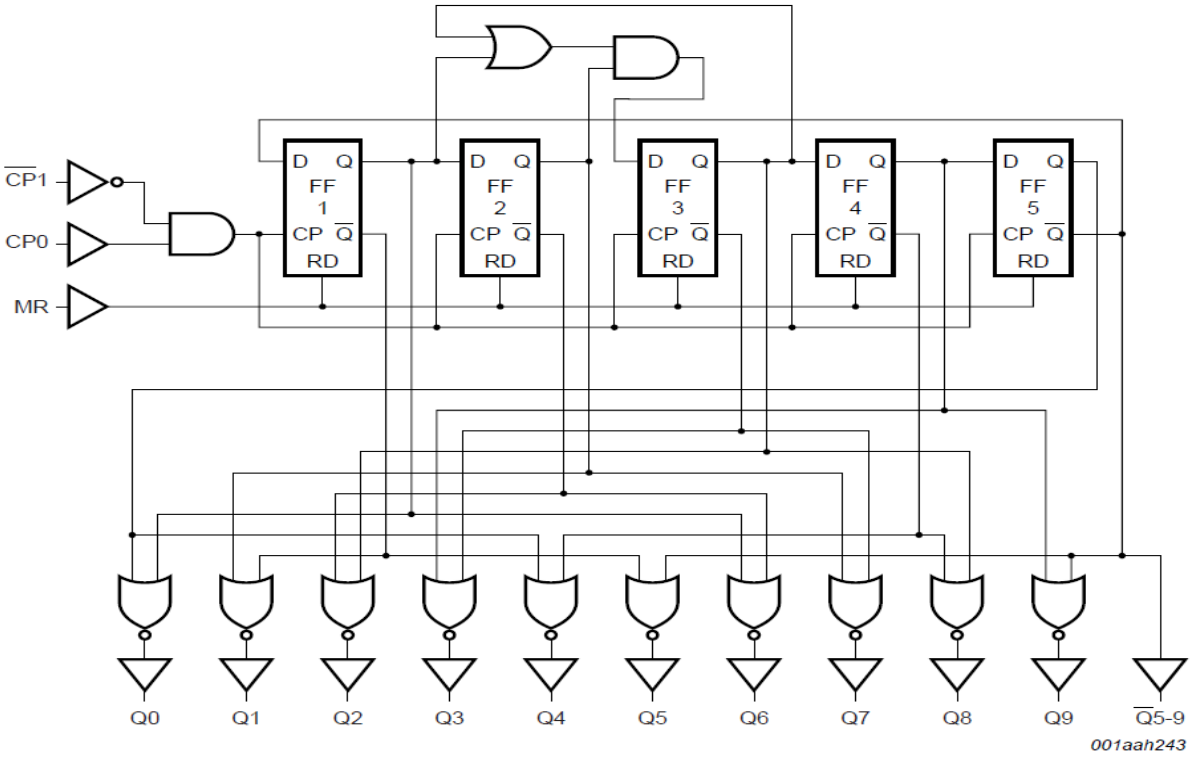
Ο 4017 δεκαδικός μετρητής

Το IC 4017 είναι ένα ευέλικτο IC της οικογένειας CMOS το οποίο έχει ένα ευρύ φάσμα εφαρμογών. Εσωτερικά αποτελείται από ένα 10 stages δεκαδικού μετρητή / διαιρέτη. Όταν ένας παλμός του ρολογιού εφαρμόζεται εξωτερικά, τα αποτελέσματα της να γίνει η λογική «hi» και διαδοχικά «lo» (το ένα μετά το άλλο).

Το λειτουργικό διαγραμμά



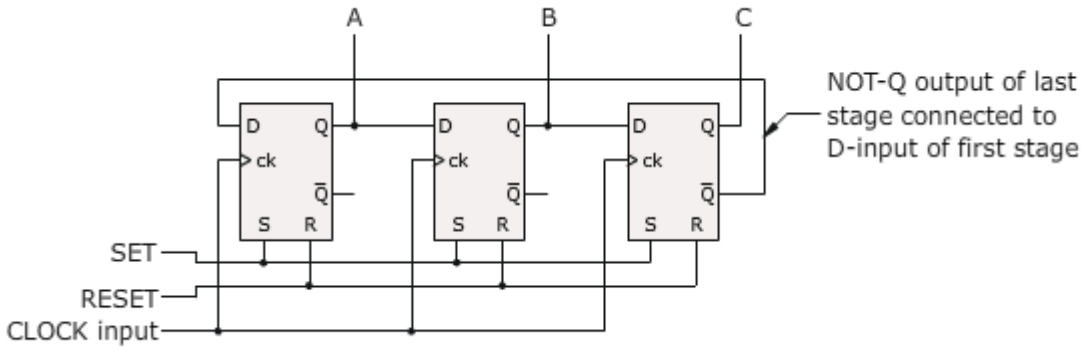
Το λογίκο διάγραμμα



Οπώς φαίνεται και στα διαγράμματα ο 4017 διαθέτει ένα 5-stage Johnson Counter και κυκλώμα αποκωδικοποιησής σε 10 εξόδους Q0 εως Q9. Έχει 2 είσοδους clock οι οποίοι είναι συνεδεμένη σε μια AND πύλη, το CP1 έχει και μια Not πριν τη πύλη και ένα MR master reset. Και με την έξοδο του πιο σημαντικου FFs.

Johnson counter

Ένας μετρητής Johnson είναι ένας τύπος walking ring counter χρησιμοποιώντας ένα κύκλωμα καταχωρητή μετατόπισης στην οποία το NOT-Q, ή αντίστροφος παραγωγή του τελικού σταδίου συνδέεται με τη σειριακή είσοδο του πρώτου σταδίου. Χρειάζεται ένα διάγραμμα για να σας βοηθήσει να καταλάβετε αυτό



Decoder Stage

Ένα στάδιο αποκωδικοποιητής είναι επίσης αναγκαία. Αυτό χρησιμοποιεί 2-input NOR πύλες για να προσδιορίσει μοναδικά το καθένα από τα 6 κράτη στην ακολουθία απαρίθμησης.

Θυμηθείτε τον πίνακα αλήθειας της μια πύλη NOR:

input B	*input A*	*output*
0	0	1
0	1	0
1	0	0
1	1	0

Όπως φαίνεται εξοδό έχω μονό όταν και οι 2 εισοδοί είναι 0

3-stage counter sequence:

clock pulses	*D input*	*output A*	*output B*	*output C*
0	1	0	0	0
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	1	1
5	0	0	0	1
6	1	0	0	0
sequence repeats..				

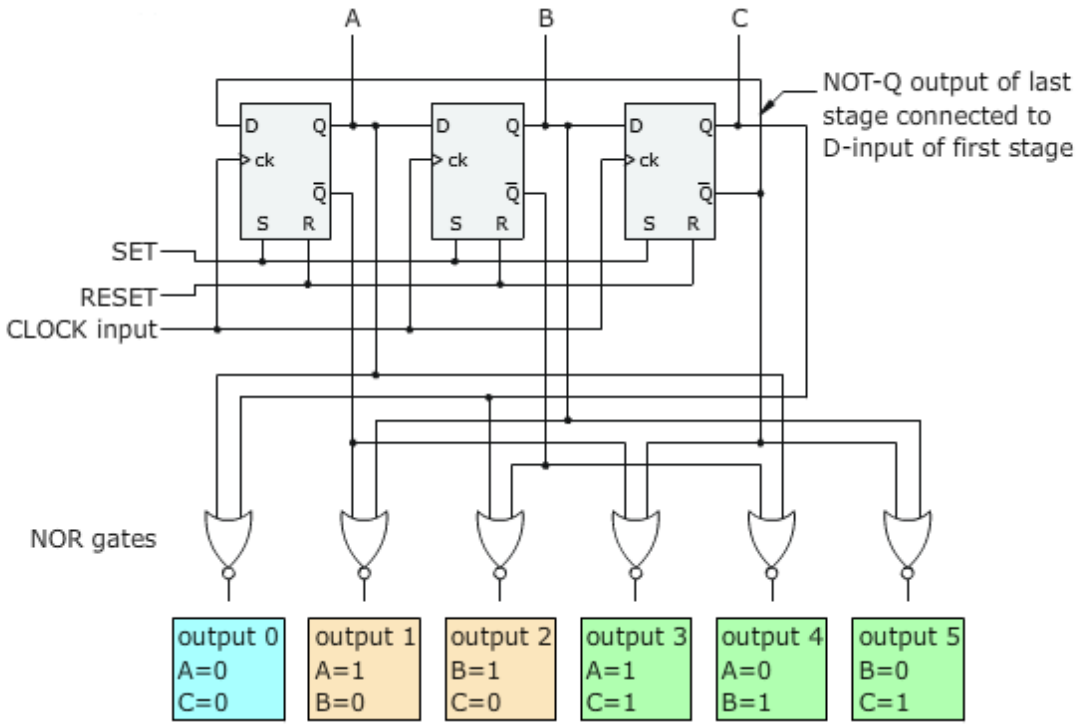
Η πρώτη γραμμή του πίνακα μπορεί να αποκωδικοποιηθεί με μοναδικό τρόπο από τη σύνδεση A και C με τις εισόδους μιας πύλης NOR. Αυτό είναι η μόνη κατάσταση στην ακολουθία για τους οποίους A = 0 και C = 0.

Καθώς ο μετρητής χρονίζεται, η λογική κατάσταση στην είσοδο D μεταφέρεται κατά μήκος από ένα flip-flop στο επόμενο. Η δεύτερη γραμμή δείχνει την κατάσταση μόνο στην αλληλουχία για την οποία A = 1 και B = 0, όπως υποδεικνύεται από τη σκίαση. Αυτή η γραμμή μπορεί να αποκωδικοποιηθεί μοναδικά συνδέοντας NOT-A και B με τις εισόδους μιας πύλης NOR.

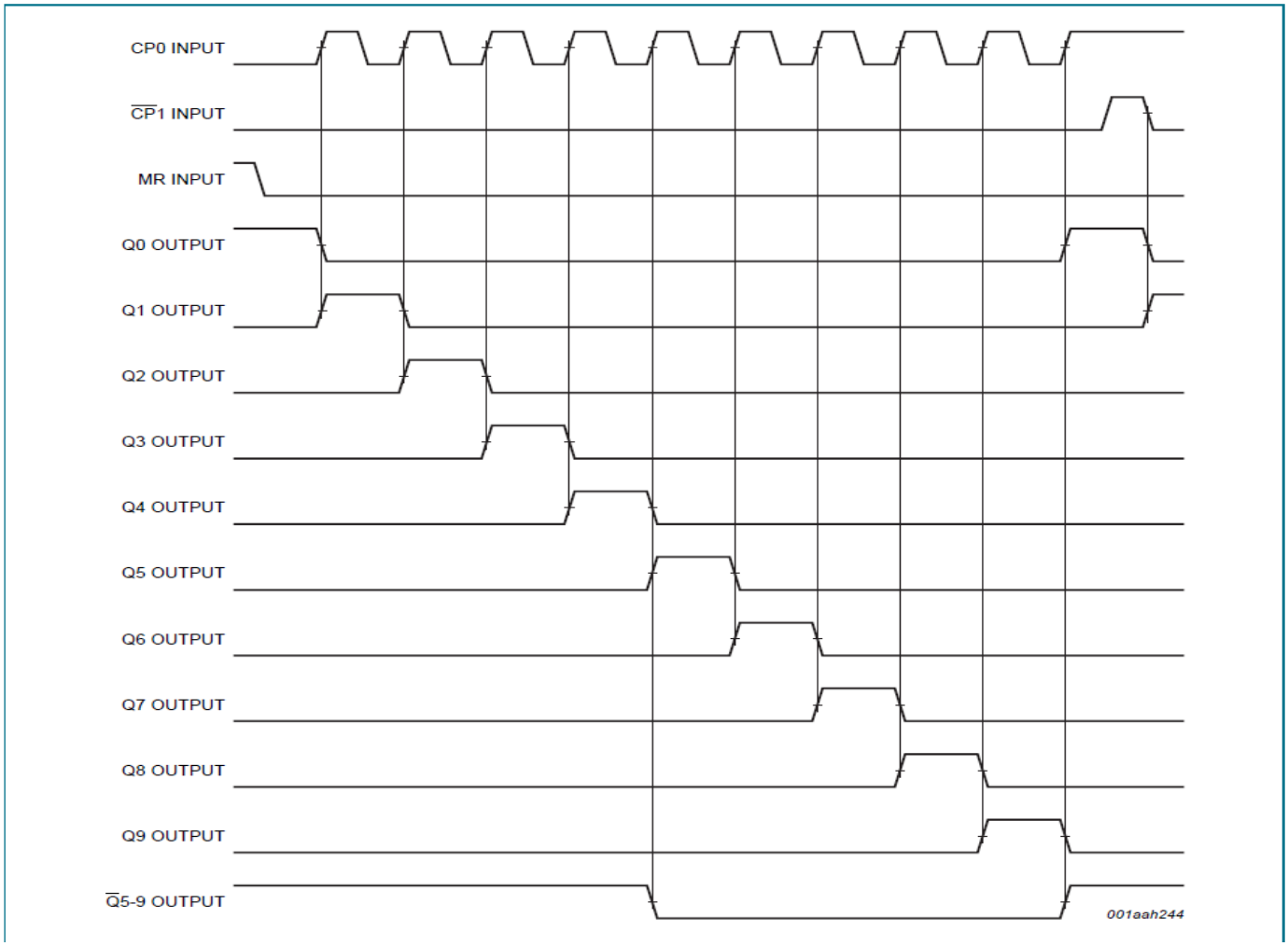
Η τρίτη γραμμή είναι η μόνη κατάσταση στην αλληλουχία για την οποία B = 1 και C = 0, όπως υποδεικνύεται από τη σκίαση. Αυτή η γραμμή μπορεί να αποκωδικοποιηθεί με μοναδικό τρόπο από τη σύνδεση NOT-B και C με τις εισόδους μιας πύλης NOR.

Οι υπόλοιπες γραμμές μπορούν να αποκωδικοποιηθούν με παρόμοιο τρόπο με ανίχνευση των ζεύγη τιμών σκίαση στον πίνακα. Σημειώστε ότι η είσοδος D είναι η ίδια όπως NOT-C.

Το διάγραμμα δείχνει πώς οι εξοδοί του Johnson Counter μπορεί να αποκωδικοποιηθεί για να δώσει 1 από 6 εξόδου:



Ο 5-Stage Johnson Counter χρησιμοποίειτε από τον 4017, έχει 5 FFs και μετράει μέχρι το 10 και παρακάτω θα δείτε το timing διαγραμμά:



ΚΕΦΑΛΑΙΟ 4: *Το LED Matrix της πτυχιακής και το control board*

Το LED Matrix της πτυχιακής εργασίας είναι βασισμένο σε RGB LEDs κοινής ανόδου, (στη επομένη σελίδα).

Είναι μεγέθους 24Χ6 και ακολουθεί την εξής συνδεσμολογία στην επόμενη σελίδα, σειρές είναι 6 όπως φαίνετε στο σχήμα και ενώνονται οι κοινές άνοδοι από το κάθε RGB LED της σειράς και οι στήλες είναι κάθετα στο σχήμα και ενώνονται οι κάθοδοι για το κάθε χρώμα ξεχωριστά. Και με κονέκτορες συνδέονται στο κύκλωμα του ελεγκτή του LED MATRIX. Για τις σειρές είναι ένας που συνδέονται 1-6. Αλλά για τις στήλες και λόγο ότι είναι πολλά χρώματα η συνδεσμολογία είναι χωρισμένη σε 2 δωδεκάδες, 1 έως 12 για το κάθε χρώμα του RGB LED και μετά το ίδιο για τις στήλες 13-24.

Το control board

Το control board συνδέετε με το arduino board, τα pins που χρησιμοποιούνται είναι 8 και 9 για τον δεκαδικό μετρητή 4017 που σαρώνει τις σειρές και 10, 11, 13 που συνδέονται με τους καταχωρητές ολίσθησης που κάνουν την σάρωση των στηλών.

Το 10 Pin συνδέετε με το 12 pin όλων των shift register, STCP

Το 13 Pin συνδέετε με το 11 pin όλων των shift register, SHCP

Το 11 Pin συνδέετε με το ποδαράκι 14 του 1^{o} shift register για τα σειριακά δεδομένα

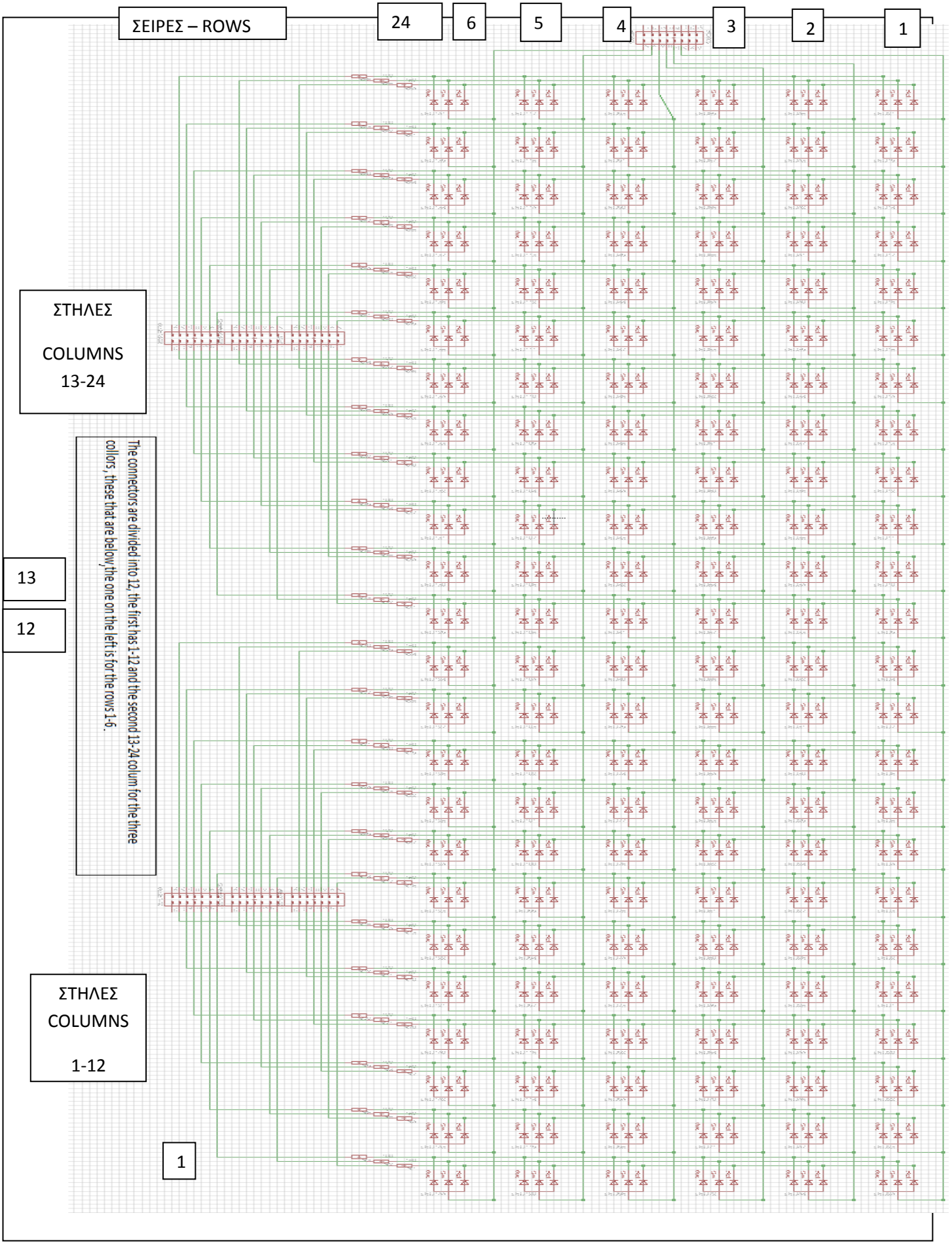
Το 9 Pin συνδέετε με το ρολόι του 4017

Το 8 Pin συνδέετε με το ποδαράκι 15 του 4017, master reset

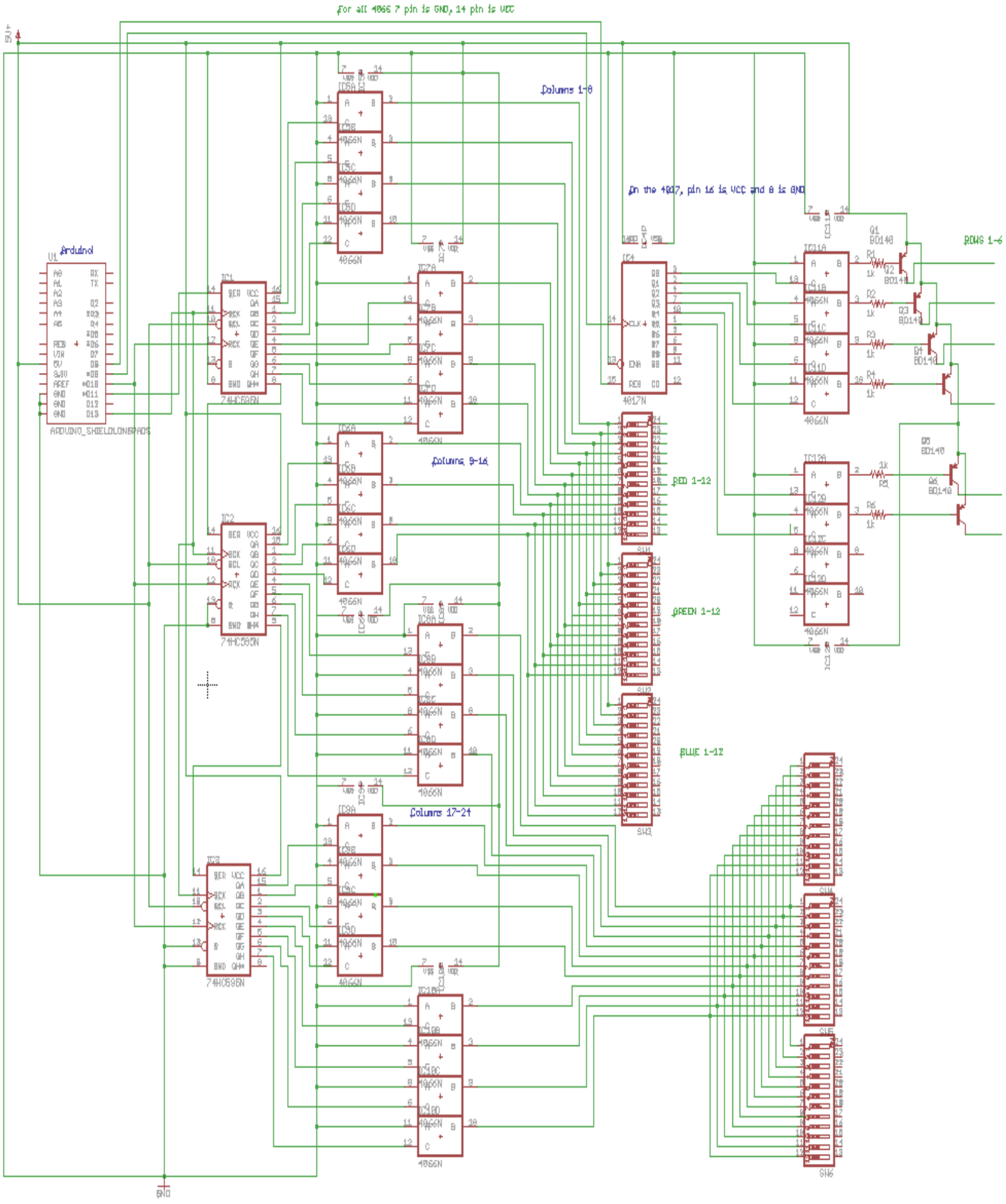
Από τους shift register έχω συνολικά 24 εξόδους που για να μπορούν να γειώσουν τα RBG LED χρησιμοποιώ Quad single-pole single-throw analog switch 74HC4066. Την έξοδο των shift register την συνδέω με το enable των 4066 και στον διακόπτες που βρίσκονται εσωτερικά στον 4066 συνδέω μια άκρη στη γείωση και η άλλη βγαίνει σαν την τελική έξοδο για να οδηγήσει τα LEDs.

Για να έχω την επιλογή του χρώματος που θα δουλεύει το LED matric χρησιμοποιώ dip switches και μετά αυτά συνδέονται με το ανάλογό connector, συνολικά είναι 6 dip switches των 12 και συνδέουν τις στήλες 1 έως 24 και έχω τον έλεγχο για κάθε στήλη.

Στις εξόδους του 4017 πάλι χρησιμοποιώ τον 4066 για να μπορώ να χρησιμοποιήσω PNP transistor σαν διακόπτη για να οδηγήσουν την κοινή τάση των RGB led. Στα enable των 4066 συνδέω τις εξόδους του 4017, και στον διακόπτη την μία άκρη στην γείωση και την άλλη στην βάση των PNP τρανζίστορ, στα οποία ο εκπομπός όλων των τρανζίστορ συνδέετε στην τάση και ο συλλέκτης συνδέετε με τα rows του led matrix.


ΣΕΙΡΕΣ – ROWS
24
6
5
4
3
2
1
ΣΤΗΛΕΣ
COLUMNS
13-24
The connectors are divided into 12, the first has 1-12 and the second 13-24 colum for the three collors, these that are below, the one on the left is for the rows 1-6.
13
12
ΣΤΗΛΕΣ
COLUMNS
1-12
1

Το κύκλωμα του control board



ΚΕΦΑΛΑΙΟ 6: Το πρόγραμμα της πτυχιακής

Το πρόγραμμα που χρησιμοποιώ στον arduino για να στέλνει τα δεδομένα στο control board/

Με το #define δηλώνουμε τον δυαδικό mapping για τους χαρακτήρες των μνημάτων που θα αναγράφονται στο LED matrix, σε ποιο εύχρηστες μεταβλητές. Το mapping είναι 6 σειρές και 5 στήλες και με την ίδια λογική μπορούμε να κάνουμε ότι χαρακτήρα θέλουμε. Σε αυτό το πρόγραμμα είναι δηλωμένα όλο το λατινικό αλφάβητο, (L)πεζά και (B)κεφαλαία επίσης σημεία στίξης και αριθμητικούς χαρακτήρες.

```
#define BA {B01110000,B10001000,B10001000,B11111000,B10001000,B10001000}
#define BB {B11110000,B10001000,B10001000,B11110000,B10001000,B11111000}
#define BC {B11111000,B10000000,B10000000,B10000000,B10000000,B11111000}
#define BD {B11110000,B10001000,B10001000,B10001000,B10001000,B11110000}
#define BE {B11111000,B10000000,B10000000,B11110000,B10000000,B11111000}
#define BF {B11111000,B10000000,B10000000,B11110000,B10000000,B10000000}
#define BG {B01110000,B10001000,B10000000,B10011000,B10001000,B01110000}
#define BH {B10001000,B10001000,B11111000,B10001000,B10001000,B10001000}
#define BI {B11111000,B00100000,B00100000,B00100000,B00100000,B11111000}
#define BJ {B00111000,B00010000,B00010000,B00010000,B10010000,B01100000}
#define BM {B10001000,B11011000,B10101000,B10101000,B10001000,B10001000}
#define BN {B10001000,B11001000,B10101000,B10101000,B10011000,B10001000}
#define BL {B10000000,B10000000,B10000000,B10000000,B10000000,B11111000}
#define BO {B01110000,B10001000,B10001000,B10001000,B10001000,B01110000}
#define BP {B11110000,B10001000,B10001000,B11110000,B10000000,B10000000}
#define BQ {B01110000,B10001000,B10101000,B10011000,B01111000,B00001000}
#define BR {B11110000,B10001000,B10001000,B11110000,B10001000,B10001000}
#define BS {B01110000,B10001000,B01100000,B00010000,B10001000,B01110000}
#define BK {B10001000,B10010000,B11100000,B11100000,B10010000,B10001000}
#define BT {B11111000,B00100000,B00100000,B00100000,B00100000,B00100000}
#define BU {B10001000,B10001000,B10001000,B10001000,B10001000,B01110000}
```

```
#define BV {B10001000,B10001000,B10001000,B10001000,B01010000,B00100000}
#define BW {B10001000,B10001000,B10101000,B10101000,B10101000,B01010000}
#define BX {B10001000,B01010000,B00100000,B00100000,B01010000,B10001000}
#define BY {B10001000,B01010000,B00100000,B00100000,B00100000,B00100000}
#define BZ {B11111000,B00001000,B00110000,B01100000,B10000000,B11111000}
#define LA{B00000000,B01110000,B00001000,B01111000,B10001000,B01111000}
#define LB{B10000000,B10000000,B10110000,B11001000,B10001000,B11110000}
#define LC{B00000000,B01110000,B10000000,B10000000,B10001000,B01110000}
#define LD{B00001000,B00001000,B01111000,B10001000,B10001000,B01111000}
#define LE{B00000000,B01110000,B10001000,B11111000,B10000000,B01110000}
#define LF{B00110000,B01001000,B01000000,B11100000,B01000000,B01000000}
#define LG{B00000000,B01111000,B10001000,B01111000,B00001000,B01110000}
#define LH{B10000000,B10000000,B10110000,B11001000,B10001000,B10001000}
#define LI{B00100000,B00000000,B01100000,B00100000,B00100000,B01111000}
#define LJ{B00010000,B00000000,B00111000,B00010000,B10010000,B01100000}
#define LK{B10000000,B10010000,B10100000,B11000000,B10100000,B10010000}
#define LL{B01100000,B00100000,B00100000,B00100000,B00100000,B01111000}
#define LM{B00000000,B00000000,B11010000,B10101000,B10101000,B10001000}
#define LN{B00000000,B00000000,B10110000,B11001000,B10001000,B10001000}
#define LO{B00000000,B01110000,B10001000,B10001000,B10001000,B01110000}
#define LP{B00000000,B11110000,B10001000,B11110000,B10000000,B10000000}
#define LQ{B00000000,B01101000,B10011000,B01111000,B00001000,B00001000}
#define LR{B00000000,B00000000,B10110000,B11001000,B10000000,B10000000}
#define LS{B00000000,B01110000,B10000000,B01110000,B00001000,B11110000}
#define LT{B01000000,B01000000,B11100000,B01000000,B01001000,B00110000}
#define LU{B00000000,B00000000,B10001000,B10001000,B10011000,B01101000}
#define LV{B00000000,B00000000,B10001000,B10001000,B01010000,B00100000}
#define LW{B00000000,B00000000,B10001000,B10101000,B10101000,B01010000}
```

```
#define LX{B00000000,B10001000,B01010000,B00100000,B01010000,B10001000}
#define LY{B00000000,B10001000,B10001000,B01111000,B00001000,B01110000}
#define LZ{B00000000,B11111000,B00010000,B00100000,B01000000,B11111000}
#define SPACE{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000}
#define NUM0{B01110000,B10011000,B10101000,B10101000,B11001000,B01110000}
#define NUM1{B00100000,B01100000,B10100000,B00100000,B00100000,B01110000}
#define NUM2{B01110000,B10001000,B00001000,B01110000,B10000000,B11111000}
#define NUM3{B11110000,B00001000,B00001000,B01111000,B00001000,B11110000}
#define NUM4{B10001000,B10001000,B10001000,B11111000,B00001000,B00001000}
#define NUM5{B11111000,B10000000,B11110000,B00001000,B10001000,B01110000}
#define NUM6{B11111000,B10000000,B11111000,B10001000,B10001000,B11111000}
#define NUM7{B11111000,B00001000,B00001000,B01111000,B00001000,B00001000}
#define NUM8{B11111000,B10001000,B11111000,B10001000,B10001000,B11111000}
#define NUM9{B11111000,B10001000,B11111000,B00001000,B00001000,B11111000}
#define DEVIDE{B00001000,B00010000,B00100000,B00100000,B01000000,B10000000}
#define TWODOTS{B01100000,B01100000,B00000000,B00000000,B01100000,B01100000}
#define DOT{B00000000,B00000000,B00000000,B00000000,B01100000,B01100000}
#define COMA{B00000000,B00000000,B00000000,B00110000,B00110000,B01100000}
#define LINE{B00000000,B00000000,B11111000,B11111000,B00000000,B00000000}
#define QUASTION{B01110000,B10001000,B00010000,B00100000,B00000000,B00100000}
#define MARK{B00100000,B01110000,B01110000,B00100000,B00000000,B00100000}
// Εδώ δηλώνουμε τα pins του μικροελεγκτή
int latchPin = 10;
int clockPin = 13;
int dataPin = 11;
int clock = 9;
int Reset = 8;
int latchPinPORTB = latchPin - 8;
```

```
int clockPinPORTB = clockPin - 8;

int dataPinPORTB = dataPin - 8;

int i = 0;

long scrolling_word[6]; // εδώ δηλώνουμε των αριθμό των σειρών

int array_turn=0;

byte your_text[8][6]={BH,BI,SPACE,BW,BO,BR,BL,BD}; // εδώ δηλώνουμε τους χαρακτήρες, που //έχουμε δηλώσει πριν με τα #define τα [8][6] είναι ο αριθμός των χαρακτήρων και ο αριθμός των //σειρών του μυνυματος


//Ο βασικός βρόγχος του setup, δηλώνει τα pins σαν εξόδους με τις αντίστοιχες εντολές // Pinmode, και με την εντολή digitalWrite κάνει reset στον δεκαδικό μετρητή.

void setup(){

  Serial.begin(9600);

  pinMode(dataPin,OUTPUT);

  pinMode(clockPin,OUTPUT);

  pinMode(latchPin,OUTPUT);

  pinMode(clock,OUTPUT);

  pinMode(Reset,OUTPUT);

  digitalWrite(Reset,HIGH);

  digitalWrite(Reset,LOW);

  setupSPI();

}

//ο βρόγχος του βασικού προγράμματος

void display_word(int loops,byte word_print[][6],int num_patterns,int delay_langth){//αυτή η εντολή //απεικονίζει τα σύμβολα μας

 i = 0;// resets τον μετρητή για το 4017

  for(int g=0;g<6;g++)//resets the the long int where your word goes

    scrolling_word[g] = 0;

  for(int x=0;x<num_patterns;x++){//ο βασικός βρόγχος πηγαίνει πέρα από τα σύμβολα μας
```

```
  for(int r=0;r<6;r++)// βάζει τα buildes στο πρώτο σύμβολο
    scrolling_word[r] |= word_print[x][r];
   for (int z=0;z<6;z++){//η δράση της κύλισης
     for(int p=0;p<6;p++)
       scrolling_word[p] = scrolling_word[p] << 1;
// end of the scrolling funcion
    for(int t=0;t<delay_langth;t++){// λειτουργία καθυστέρησης, να επαναλαμβάνεται ακριβώς πάνω
//στην ίδια οθόνη
     for(int y=0;y<6;y++){// scaning την οθόνη
      if(i == 6){// μετρώντας έως 6 με το 4017
       digitalWrite(Reset,HIGH);
       digitalWrite(Reset,LOW);
       i = 0;
      }
      latchOff();
      spi_transfer(make_word(0x01000000,y));// στέλνει τα δεδομένα
      spi_transfer(make_word(0x00010000,y));
      spi_transfer(make_word(0x00000100,y));
      latchOn();
      delayMicroseconds(800);//καθυστέρηση
      latchOff();
      spi_transfer(0);// καθαρίζει τα δεδομένα
      spi_transfer(0);
      spi_transfer(0);
      latchOn();
      digitalWrite(clock,HIGH);// μετρώντας πάνω μει το 4017
      digitalWrite(clock,LOW);
```

```
        i++;
       }
      }
     }
    }
  finish_scroll(delay_langth);
}
void finish_scroll(int delay_scroll){// Αυτή η λειτουργία είναι η ίδια όπως η λειτουργία ανωτέρω,
//απλώς τελειώνει την κύλιση
 for (int n=0;n<24;n++){
       for(int h=0;h<6;h++)
         scrolling_word[h] = scrolling_word[h] << 1;
     for(int w=0;w<delay_scroll;w++){
       for(int k=0;k<6;k++){
         if(i == 6){
           digitalWrite(Reset,HIGH);
           digitalWrite(Reset,LOW);
           i = 0;
         }
         latchOff();
         spi_transfer(make_word(0x01000000,k));
         spi_transfer(make_word(0x00010000,k));
         spi_transfer(make_word(0x00000100,k));
         latchOn();
         delayMicroseconds(800);
         latchOff();
         spi_transfer(0);
         spi_transfer(0);
```

```
          spi_transfer(0);
          latchOn();
          digitalWrite(clock,HIGH);
          digitalWrite(clock,LOW);
          i++;
        }
      }
    }
}

byte make_word (long posistion,byte turn){
  byte dummy_word = 0;
  for(int q=0;q<8;q++){
    if(scrolling_word[turn] & (posistion<<q))
      dummy_word |= 0x01<<q;
  }
  return dummy_word;
}

 void loop() {

       display_word(1,your_text,8,15);// ζητεί την display_pattern λειτουργία και λέει ότι βρόχο int = 15 (αν το κάνετε πιο βρόχο το πρότυπο whould scrole πιο αργή).

      }
```

```
void latchOn(){
  bitSet(PORTB,latchPinPORTB);
}

void latchOff(){
  bitClear(PORTB,latchPinPORTB);
}

void setupSPI(){
  byte clr;
  SPCR |= ( (1<<SPE) | (1<<MSTR) ); // ενεργοποιεί τον SPI σαν master
  //SPCR |= ( (1<<SPR1) | (1<<SPR0) ); // κάνει σετ τα  prescaler bits
  SPCR &= ~( (1<<SPR1) | (1<<SPR0) ); // καθαρίζει τα prescaler bits
  clr=SPSR; // clear SPI status reg
  clr=SPDR; // clear SPI data reg
  SPSR |= (1<<SPI2X); // κάνει σετ τα prescaler bits
  //SPSR &= ~(1<<SPI2X); // καθαρίζει prescaler bits

  delay(10);
}
byte spi_transfer(byte data)
{
  SPDR = data;                    // αρχίζει την μετάδοση
  while (!(SPSR & (1<<SPIF)))     // Περίμενε μέχρι το τέλος της μετάδοσης
  {
  };
  return SPDR;                    // επιστρέφει το λαμβανόμενο byte
}
```

Επιλόγος

Με την χρήση του arduino και το LED MATRIX απεικονίζει 24X6 ψηφία για την εμφάνιση μηνυμάτων σε πινακίδα LED, και με την ικανότητα να δίνει ρεύμα μέχρι 1 Ampere (για το UNO rev3) το κύκλωμα του control board αλλά και το RGB LED MATRIX μπορεί να δουλέψει και με μια μπαταρία 9V, (για 1 χρώμα όχι παραπάνω, αλλιώς θα χρειαστεί εξωτερική τροφοδοσία 2Amprete 9V)

Βιβλιογραφία

Programming Arduino: Getting Started With Sketches , Monk Simon
Exploring Arduino: Tools and Techniques for Engineering Wizardry, Jeremy Blum

Η σελίδα του arduino με τις επεξηγήσεις και παραδείγματα

Arduino Cookbook , Michael Margolis

Arduino Booklet , Massimo Banzi

Arduino Programming Notebook - , Brian Evans