Arduino Project

ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΑ
ΙΟΥΛΙΟΣ 2019

Απόστολος Φανάκης, 8261
Αναστασία Παχνή Τσιτηρίδου, 8485

Περιεχόμενα

Ερώτημα 1

Ως πρώτο ζητούμενο της εργασίας ζητείται η ανάγνωση μιας θερμοκρασίας από τον αισθητήρα θερμοκρασίας και η αποθήκευση αυτής της τιμής. Κάθε 5 δευτερόλεπτα διαβάζεται μία νέα τιμή της θερμοκρασίας και όταν συμπληρωθούν 2 λεπτά (ή αντίστοιχα 24 μετρήσεις), υπολογίζεται η μέση τιμή των δειγμάτων θερμοκρασίας. Η μέση τιμή στη συνέχεια εμφανίζεται στην οθόνη LCD για χρονικό διάστημα 10 δευτερολέπτων.

Ανάγνωση θερμοκρασίας από αισθητήρα

Η δειγματοληψία μίας θερμοκρασίας υλοποιείται στην συνάρτηση getNewTemp. Η συνάρτηση αυτή αρχικά διαβάζει την θερμοκρασία από τον DHT αισθητήρα, ελέγχει αν έγινε σωστή ανάγνωση θερμοκρασίας και στη συνέχεια αποθηκεύει τη τιμή που διάβασε στην πρώτη θέση ενός πίνακα 24 θέσεων. Οι τιμές που υπήρχαν στον πίνακα πριν την ανάγνωση της τελευταίας θερμοκρασίας, μετακινούνται μία θέση δεξιά. Έτσι το index των τιμών αλλάζει ως εξής: $i' = i - 1$, για $i \in (0, 23]$. Τέλος, αυξάνεται ο counter των μετρήσεων (temperatureReadingsCounter) και επιστρέφεται η τιμή της θερμοκρασίας.

Υπολογισμός μέσης θερμοκρασίας

Ο υπολογισμός της μέσης τιμής των 24 θερμοκρασιών γίνεται στη συνάρτηση calcAverageTempAndReset. Η συνάρτηση αυτή, όπως υποδεικνύει και το όνομά της, πέρα από την επιστροφή της μέσης τιμής κάνει reset και τον counter των μετρήσεων.

Χρονισμός

Ο χρονισμός υλοποιείται στην συνάρτηση loop().

Αρχικά ελέγχεται αν έχει περάσει το επιθυμητό χρονικό διάστημα μεταξύ των μετρήσεων ανάμεσα στην τρέχουσα χρονική στιγμή και την χρονική στιγμή της τελευταία μέτρησης. Για τον έλεγχο αυτής της συνθήκης χρησιμοποιείται η συνάρτηση millis που δίνει την τρέχουσα χρονική στιγμή, η μεταβλητή lastReading που έχει αποθηκευμένη την χρονική στιγμή της τελευταίας μέτρησης και η σταθερά SLEEP_INTERVAL που καθορίζεται στο header αρχείο. Από σύμβαση, η μεταβλητή lastReading αρχικοποιείται στην τιμή 0 στο αρχείο header και έτσι λαμβάνεται η πρώτη μέτρηση του θερμοστάτη. Συγκεκριμένα γίνεται ο έλεγχος που ακολουθεί:

```
if (!lastReading || millis() - lastReading > SLEEP_INTERVAL) {
   getNewSamples();
   lastReading = millis();
}
```

Πίνακας 1 Έλεγχος για την λήψη νέας μέτρησης του θερμοστάτη

Συνάρτηση getNewSamples

Η συνάρτηση getNewSamples καλεί την συνάρτηση getNewTemp για την λήψη νέας θερμοκρασίας. Παράλληλα, ελέγχει αν ο counter των μετρήσεων είναι ίσος με 24 και αν η συνθήκη είναι αληθής καλεί την συνάρτηση calcAverageTempAndReset και την συνάρτηση sendEmail που τυπώνει στο console την μέση τιμή της θερμοκρασίας των τελευταίων δύο λεπτών. Ταυτόχρονα, αποθηκεύει στη μεταβλητή displayTimeStart τη χρονική στιγμή που ξεκινά το display της μέσης θερμοκρασίας στην οθόνη LCD. Η displayTimeStart λαμβάνει τιμή από τη συνάρτηση millis.

Μέσα στη συνάρτηση loop() γίνεται και ο παρακάτω έλεγχος για την διάρκεια εμφάνισης της μέσης τιμής στη οθόνη LCD:

```
if ((millis() - displayTimeStart) <   TEMPERATURE_DISPLAY_DURATION) {
    // Should display the average temperature
    sevseg.setNumber(averageTemp, 1);
    sevseg.refreshDisplay();
} else {
    sevseg.blank();
   sevseg.refreshDisplay();
 }
```

Πίνακας 2 Έλεγχος διάρκειας εμφάνισης μέσης τιμής στην οθόνη LCD

Η τιμή TEMPERATURE_DISPLAY_DURATION ορίζεται στο αρχείο header στα 10 δευτερόλεπτα. Όσο η συνθήκη είναι αληθής εμφανίζεται η μέση τιμή στην οθόνη. Σε διαφορετική περίπτωση η οθόνη παραμένει κενή.

Ερώτημα 2

Ως δεύτερο ζητούμενο της εργασίας ζητείται ο προγραμματισμός 2 LED με την ακόλουθη λογική:

- Ένα κόκκινο LED ενεργοποιείται όταν η τελευταία θερμοκρασία που μετρήθηκε από τον αισθητήρα είναι υψηλότερη μίας δεδομένης τιμής. Η τιμή αυτή επιλέγεται από τον προγραμματιστή και αποθηκεύεται στην μεταβλητή *HIGH_TEMP*. Όταν η τιμή πέσει κάτω από το κατώφλι που ορίζει η HIGH_TEMP , το LED απενεργοποιείται.
- Ένα μπλε LED ενεργοποιείται όταν η τελευταία θερμοκρασία που μετρήθηκε από τον αισθητήρα είναι χαμηλότερη μίας δεδομένης τιμής. Η τιμή αυτή επιλέγεται από τον προγραμματιστή και αποθηκεύεται στην μεταβλητή *LOW_TEMP*. Όταν η τιμή πέσει κάτω από το κατώφλι που ορίζει η LOW_TEMP , το LED απενεργοποιείται.

Στις παραπάνω δύο περιπτώσεις, κάθε φορά που ενεργοποιείται κάποιο εκ των δύο LED στέλνετε ένα email.

Έλεγχος κατωφλιών

Ο έλεγχος για την ενεργοποίηση των LEDs γίνεται μέσα στη συνάρτηση getNewSamples. Όπως έχει ήδη αναφερθεί η συνάρτηση αυτή καλεί την συνάρτηση getNewTemp για την λήψη νέας θερμοκρασίας. Η θερμοκρασία που μετριέται, αποθηκεύεται σε μία μεταβλητή και στη συνέχεια περνά μια σειρά ελέγχων με τα κατώφλια *HIGH_TEMP & LOW_TEMP* που ενεργοποιούν ή απενεργοποιούν το κόκκινο και το μπλε LED. Κάθε φορά που ενεργοποιούνται τα LEDs καλείται η συνάρτηση sendEmail.

Τα κατώφλια HIGH_TEMP και LOW_TEMP ορίζονται στο header file.

Συνάρτηση sendEmail(int _case, float averageTemp)

Η συνάρτηση αυτή υλοποιεί εικονικά την αποστολή email. Συγκεκριμένα δέχεται δύο ορίσματα, εκ των οποίων το πρώτο καθορίζει την περίσταση που προκάλεσε αυτή τη δράση (δηλαδή είτε ανάγνωση ακραίας θερμοκρασίας είτε υπολογισμός νέας μέσης τιμής) και το δεύτερο όρισμα είτε είναι μηδέν είτε μεταφέρει τη νέα μέση τιμή για να αποσταλεί στο κείμενο του email.

Τα cases ορίζονται στο header file ως εξής:

```
#define CASE_HIGH_TEMP 0
#define CASE_LOW_TEMP 1
#define CASE_AVERAGE_TEMP 2
```

Πίνακας 3 Ορισμός περιπτώσεων αποστολής email

Αντί για την αποστολή email στην εργασία υλοποιήθηκε η εμφάνιση του αντίστοιχου μηνύματος στο console.

Ερώτημα 3

Ως τρίτο ζητούμενο της εργασίας ζητείται ο προγραμματισμός ενός διακόπτη ή ενός LED με την ακόλουθη λογική:

- Αν η θερμοκρασία ξεπεράσει ένα κατώφλι θερμοκρασίας θα κλείνει ο διακόπτης ή αντίστοιχα θα ανάβει το LED. Η τιμή του κατωφλιού επιλέγεται από τον προγραμματιστή και αποθηκεύεται στην μεταβλητή *HIGH_TEMP_RELAY*.
- Όταν η τιμή πέσει κάτω από το κατώφλι που ορίζει η *HIGH_TEMP_RELAY* , ο διακόπτης θα πρέπει να κλείνει ή αντίστοιχα το LED να απενεργοποιείται.

Στη υλοποίηση μας επιλέξαμε τη χρήση LED. Έτσι, με αντίστοιχη λογική όπως στο ερώτημα 2, ελέγχουμε το κατώφλι για κάθε νέα δειγματοληψία και ενεργοποιούμε ή απενεργοποιούμε το LED αντίστοιχα. Ο έλεγχος του κατωφλιού γίνεται στην συνάρτηση getNewSamples, όπως και προηγουμένως.

Η σταθερά HIGH_TEMP_RELAY ορίζεται στο header file.

Ερώτημα 4

Ως τελευταίο ζητούμενο της εργασίας ζητείται η χρήση ενός αισθητήρα εγγύτητας που θα ανιχνεύει την κίνηση κοντά στο θερμοστάτη. Σε περίπτωση ανίχνευσης κίνησης θα εμφανίζει στην οθόνη LCD την τελευταία μέτρηση της θερμοκρασίας και την τελευταία μέση τιμή που έχει υπολογιστεί.

Ανάγνωση απόστασης από αισθητήρα

Η δειγματοληψία της απόστασης υλοποιείται στην συνάρτηση getDistance. Ο αισθητήρας μετρά έναν παλμό σε κατάσταση HIGH η διάρκεια του οποίου ορίζεται ως η χρονική περίοδος από τη στιγμή που στέλνεται το σήμα μέχρι την λήψη της ανάκλασής του. Στη συνέχεια η διάρκεια του παλμού μετατρέπεται σε απόσταση σύμφωνα με τον τύπο

$$distance = \frac{duration * 0.5}{29.1}$$

που δίνεται από τα χαρακτηριστικά του αισθητήρα.

Πριν τη λήψη της μέτρησης και για να εγγυηθούμε την λήψη ενός καθαρού παλμού γίνεται η ακόλουθη διαδικασία

1) Γράφεται χαμηλή στάθμη παλμού στον αισθητήρα για 5 μsec
2) Γράφεται υψηλή στάθμη παλμού στον αισθητήρα για 10 μsec
3) Γράφεται χαμηλή στάθμη παλμού στον αισθητήρα
4) Το PIN τίθεται ως είσοδος

Χρονισμός

Ο χρονισμός υλοποιείται στην συνάρτηση loop(). Στο εσωτερικό της συνάρτησης καλείται η getDistance και η τιμή της απόστασης που επιστρέφει συγκρίνεται με ένα κατώφλι που ορίζεται από τον προγραμματιστή στο header file και αποθηκεύεται στην μεταβλητή *DIST_THRESHOLD*. Αν η απόσταση που μετρήθηκε είναι μικρότερη από το κατώφλι εμφανίζεται στην οθόνη η τελευταία μέση τιμή.

Ο έλεγχος της απόστασης ενσωματώνεται στον έλεγχο για την εμφάνιση της μέσης τιμής στην οθόνη LCD. Συγκεκριμένα, ο υπάρχον κώδικας τροποποιήθηκε ως εξής για να περιλαμβάνει και τον έλεγχο της απόστασης:

```
if (distance < DIST_THRESHOLD || (millis() - displayTimeStart) <
TEMPERATURE_DISPLAY_DURATION) {
   // Should display the average temperature
   sevseg.setNumber(averageTemp, 1);
   sevseg.refreshDisplay();
  } else {
   sevseg.blank();
```

```
  sevseg.refreshDisplay();
}
```

Πίνακας 4 Έλεγχος χρόνου και απόστασης (τροποποίηση του πίνακα 2)

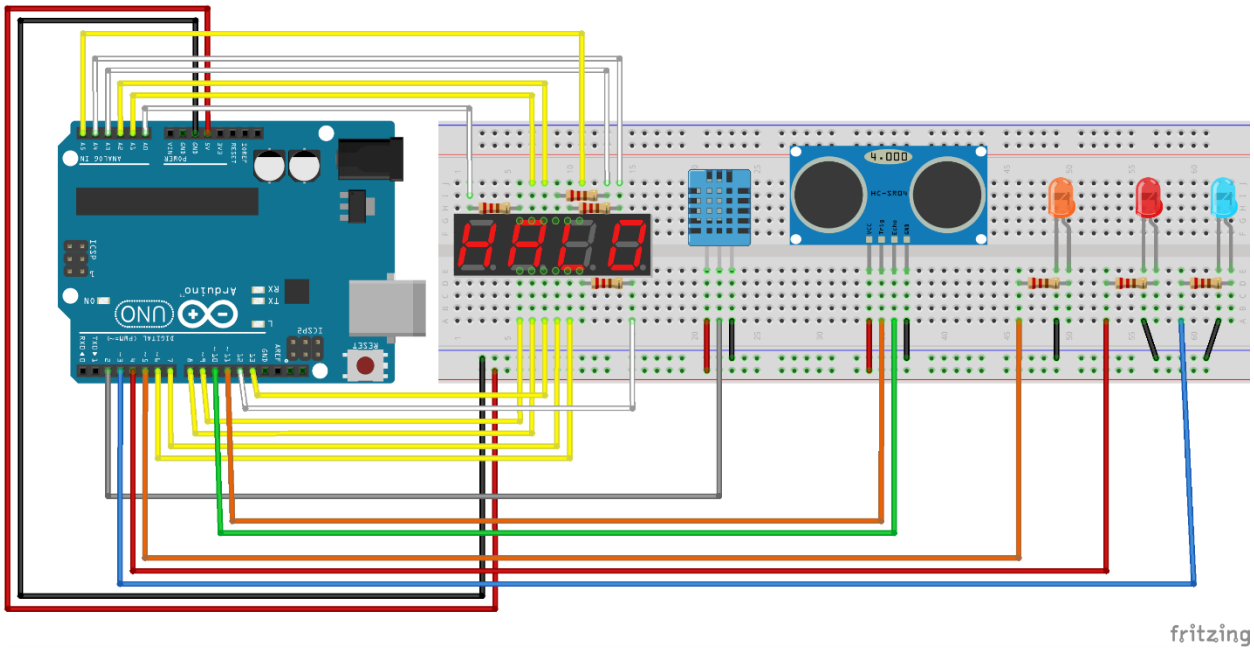
Παραδοχές

Κατά τη διάρκεια ανάπτυξης του κώδικα έγιναν οι ακόλουθες παραδοχές:

1. Θεωρούμε ότι δεν αποτελεί πρόβλημα η συνεχόμενη λειτουργία τους επεξεργαστή, για το λόγο αυτό δεν χρησιμοποιούνται εντολές sleep, αλλά ούτε και delay.
2. Εξαιτίας του διαθέσιμου hardware και συγκεκριμένα της οθόνης LCD (seven segment display) (υπάρχουν εικόνες του εξαρτήματος παρακάτω) στο τέταρτο ερώτημα εμφανίζουμε μόνο τη μέση τιμή και όχι την τελευταία μέτρηση του αισθητήρα θερμοκρασίας.
3. Επιλέχθηκε να μην χρησιμοποιηθούν interrupts καθώς δεν θεωρήθηκε απαραίτητη η χρήση τους για την επίτευξη της ζητούμενης λειτουργικότητας. (Πρόκειται για ένα σύστημα DAQ με γνωστούς χρόνους λήψης μετρήσεων). Έτσι, επιλέξαμε να έχουμε ένα πρόγραμμα με περισσότερους και ίσως πιο πολύπλοκους λογικούς ελέγχους, αλλά πιο κατανοητή και δομημένη ροή.

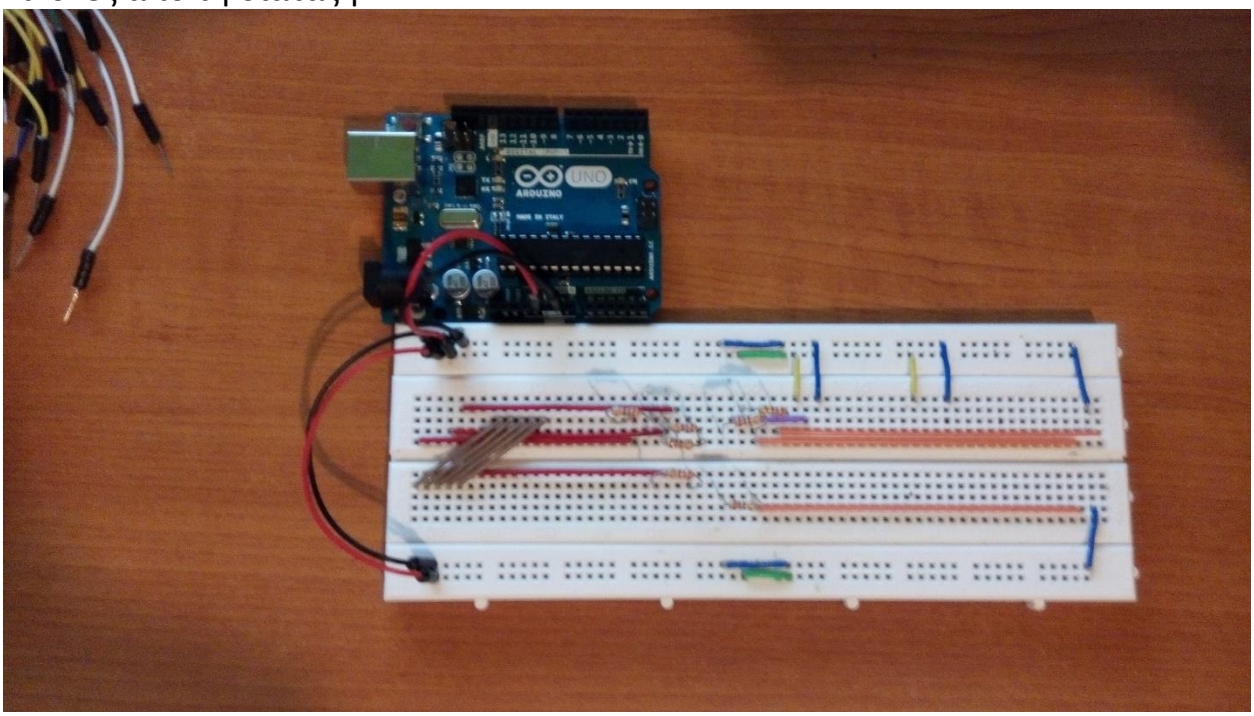
Schematic της διάταξης

Παρατίθεται το διάγραμμα των συνδέων της διάταξης με τη χρήση του προγράμματος fritzing

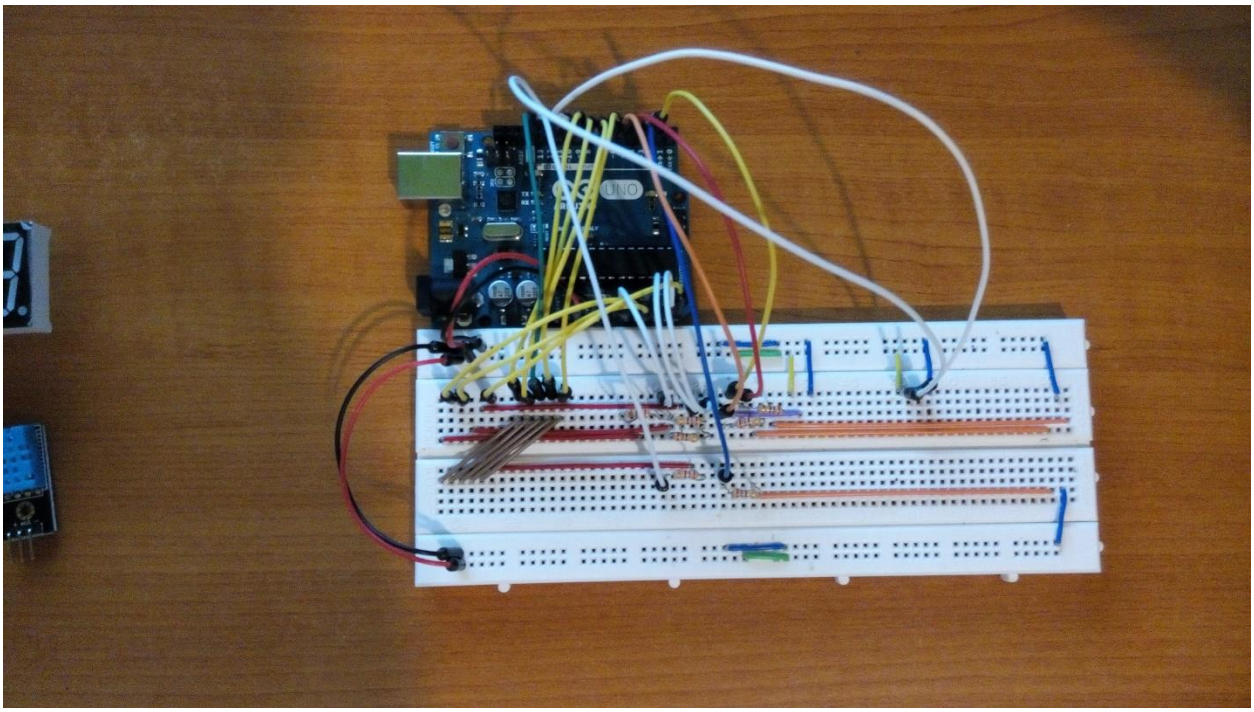


Εικόνα 1 Schematic diagram

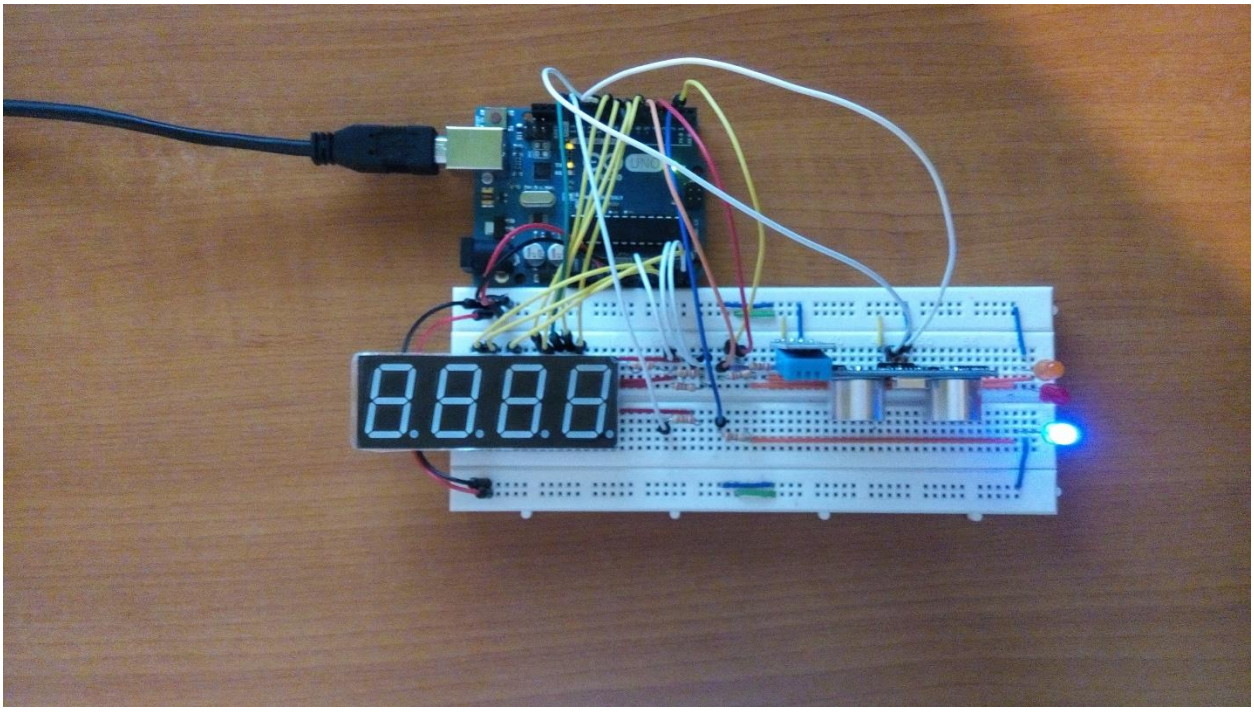
Εικόνες από τη διάταξη



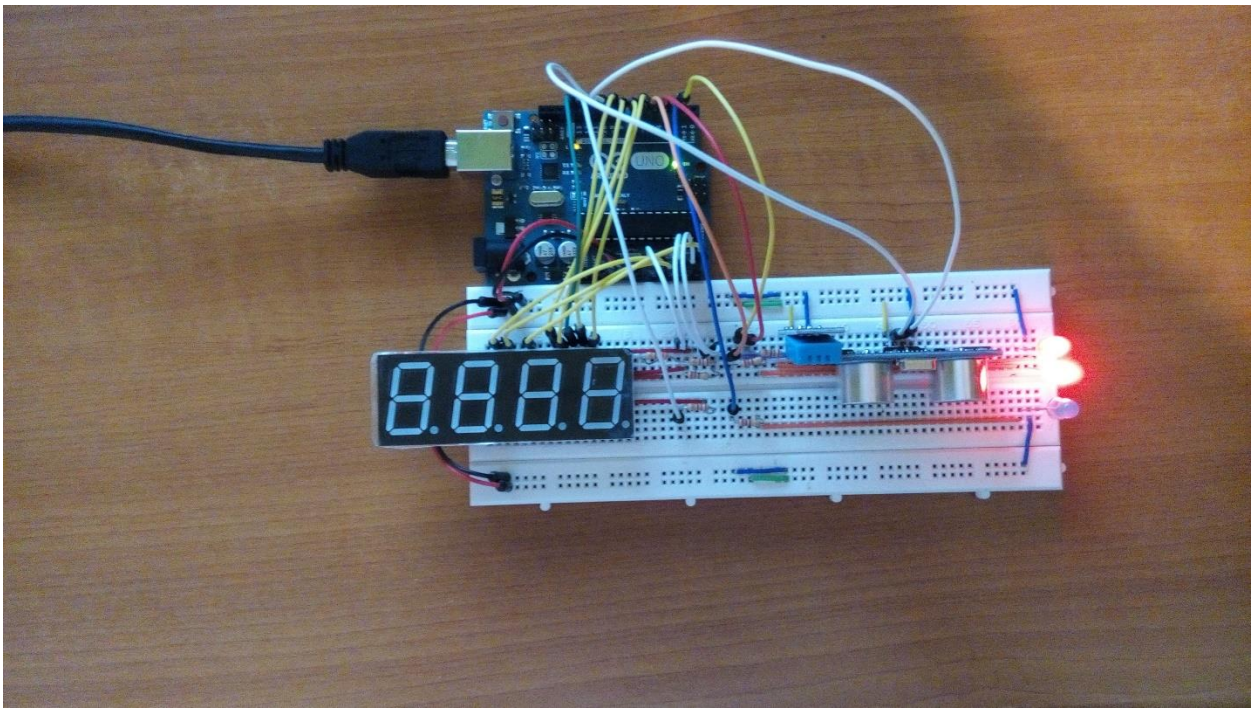
Εικόνα 2 Συνδέσεις Vcc και Ground



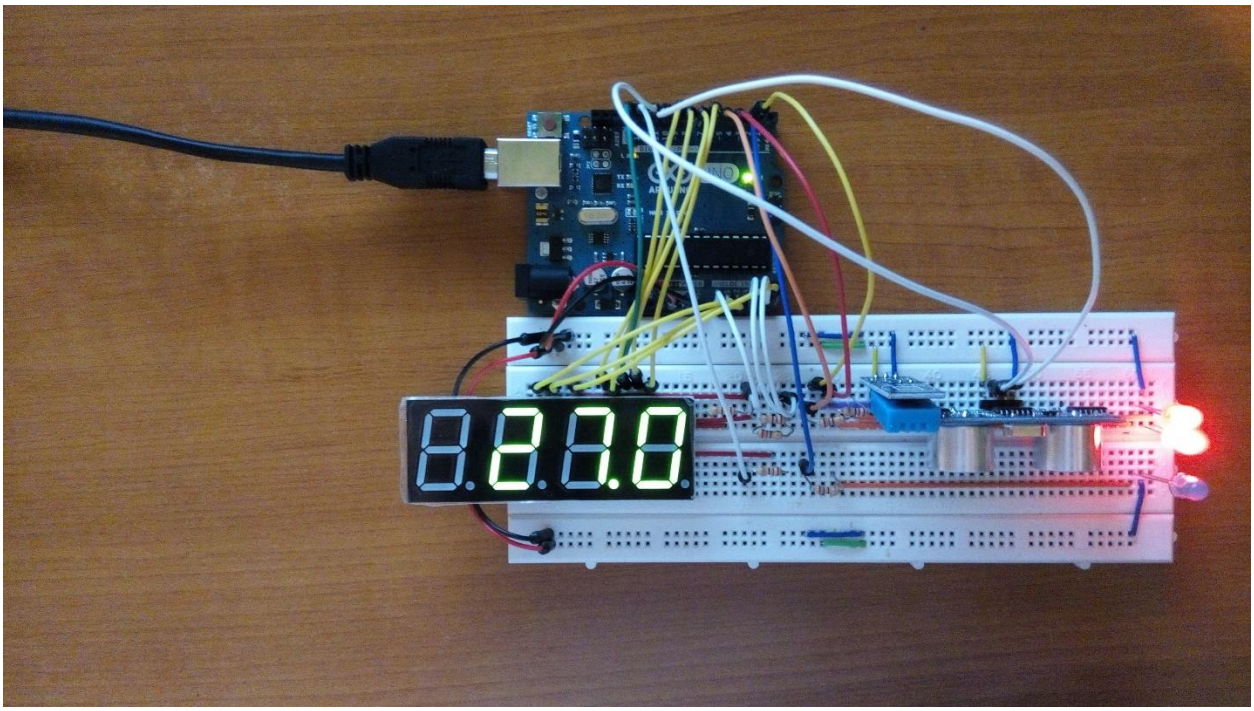
Εικόνα 3 Συνδέσεις των PIN



Εικόνα 4 Ολοκληρωμένη διάταξη με ενεργοποίηση του μπλε LED



Εικόνα 5 Ενεργοποίηση του κόκκινου LED



Εικόνα 6 Εμφάνιση μέση τιμής