ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΑ
ARDUINO PROJECT

Ονοματεπώνυμο: Παπαγγέλης Άρης Ελευθέριος
ΑΕΜ:8883
Ονοματεπώνυμο: Παπαδάμ Στέφανος
ΑΕΜ:8885

4/7/2019

Περιγραφή προβλήματος

Το ζητούμενο σε αυτό το project είναι η ανάγνωση της θερμοκρασίας του χώρου μέσω ενός αισθητήρα θερμότητας, καθώς και η εμφάνιση των αποτελεσμάτων σε μία οθόνη LCD.

Συγκεκριμένα, ο αισθητήρας θερμότητας θα παίρνει μετρήσεις ανά 5 δευτερόλεπτα για 2 λεπτά δηλαδή 24 μετρήσεις και μετά το πέρας του δίλεπτου θα εμφανίζει τη μέση τιμή στην οθόνη για 10 δευτερόλεπτα και θα ξεκινούν νέες μετρήσεις. Σε περίπτωση που η θερμοκρασία ανέβει πάνω από μια συγκεκριμένη τιμή θα ανάβει ένα κόκκινο LED και σε περίπτωση που πέσει κάτω από μία συγκεκριμένη τιμή θα ανάβει ένα μπλε LED καθώς επίσης θα εμφανίζεται και στην οθόνη αντίστοιχο προειδοποιητικό μήνυμα. Επίσης, αν η θερμοκρασία ανέβει πάνω από μία δεδομένη τιμή θα κλείνει ένας διακόπτης και θα ανοίγει όταν πέσει κάτω από αυτή την τιμή. Τέλος, μέσω ενός αισθητήρα εγγύτητας ανιχνεύεται αν κάποιος έχει πλησιάσει την συσκευή κοντύτερα από μια απόσταση και εμφανίζεται η προηγούμενη μέτρηση της θερμοκρασίας όπως επίσης και η τελευταία μέση τιμή.

Η ανάπτυξη του κώδικα πραγματοποιήθηκε στην αναπτυξιακή πλακέτα Arduino Mega 2560.

Περιγραφή του κώδικα του project και σχεδιαστικές αποφάσεις

Αρχικά κάνουμε include στο πρόγραμμα μας τις βιβλιοθήκες timer, OneWire, DallasTemperature, NewPing, LiquidCrystal έτσι ώστε να μπορούμε να χρησιμοποιήσουμε τα διάφορα αντικείμενα για τη διαχείριση του αισθητήρα θερμότητας, του αισθητήρα εγγύτητας, της οθόνης lcd και του timer που χρησιμοποιούμε για την μέτρηση των χρονικών αποστάσεων.

Στη συνέχεια ορίζουμε κάποιες σταθερές (pins, θερμοκρασίες, αποστάσεις, κλπ) τις οποίες χρησιμοποιούμε έπειτα στο πρόγραμμα μας και αρχικοποιούμε τα εξής αντικείμενα:

- **NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE)**: αρχικοποιούμε τον αισθητήρα εγγύτητας με ένα αντικείμενο τύπου NewPing το οποίο έχει σαν ορίσματα τα TRIGGER_PIN, ECHO_PIN , MAX_DISTANCE τα οποία ορίστηκαν παραπάνω σαν σταθερές.
- **LiquidCrystal lcd(9,8,5,4,3,2)**: αρχικοποιούμε την οθόνη με ένα αντικείμενο τύπου LiquidCrystal όπου οι αριθμοί που περνάμε σαν ορίσματα αντιστοιχούν στα pins rs, enable, d4, d5, d6, d7 της οθόνης.
- **OneWire oneWire(ONE_WIRE_BUS):** Δημιουργούμε ένα αντικείμενο τύπου oneWire για να επικοινωνούμε με τις OneWire συσκευές.
- **DallasTemperature sensors(&oneWire):** Δημιουργούμε ένα αντικείμενο τύπου DallasTemperature και του δίνουμε σαν όρισμα την αναφορά του OneWire αντικειμένου έτσι ώστε να μπορούμε να χρησιμοποιήσουμε τον αισθητήρα θερμοκρασίας.
- **auto timer = timer_create_default():** Δημιουργούμε ένα αντικείμενο τύπου timer, μιας ταυτόχρονης διεργασίας με χρονική ανάλυση σε millis. Θα χρησιμοποιηθεί παρακάτω για την κλήση της συνάρτησης getTemp() κάθε 5 δευτερόλεπτα.

Παρακάτω ορίζουμε κάποιες καθολικές μεταβλητές που θα χρειαστούμε:

- **volatile int i:** Η μεταβλητή i χρησιμοποιείται σαν μετρητής για να ξέρουμε πότε έχουμε φτάσει τις 24 μετρήσεις άρα και τη συμπλήρωση του δίλεπτου.
- **volatile float temp[24]:** Ο πίνακας temp χρησιμοποιείται για να αποθηκεύσουμε τις 24 μετρήσεις θερμοκρασίας εντός του δίλεπτου.
- **volatile float meanTemp:** Στη μεταβλητή αυτή αποθηκεύουμε τη μέση τιμή των θερμοκρασιών που παίρνουμε.
- **volatile float prevTemp:** Στη μεταβλητή αυτή αποθηκεύουμε την προηγούμενη τιμή της θερμοκρασίας.
- **volatile unsigned long screenTime:** Η μεταβλητή αυτή χρησιμοποιείται για να αποθηκεύσουμε τη στιγμή που ξεκινάει να εμφανίζεται η μέση τιμή στην οθόνη έτσι ώστε να μετρήσουμε το πέρασμα των 10 δευτερολέπτων.

- **volatile bool flag:** Η μεταβλητή αυτή είναι τύπου boolean και τη χρησιμοποιούμε για να δούμε αν η θερμοκρασία επανέλθει σε φυσιολογικά πλαίσια, οπότε και θα καθαρίσουμε την οθόνη από την ειδοποίηση υψηλής ή χαμηλής θερμοκρασίας.

Έπειτα ακολουθούν οι εξής συναρτήσεις:

- **void setup():** Η συνάρτηση αυτή αρχικοποιεί τις μεταβλητές μας και καλείται μία φορά στο πρόγραμμα μας. Με τις εντολές Serial.begin(9600) , lcd.begin(NUM_COLS,NUM_ROWS) , sensors.begin() αρχίζουμε την επικοινωνία με τον υπολογιστή μας ορίζοντας το baud σε 9600, με την οθόνη lcd ορίζοντας τις γραμμές και τις στήλες της lcd οθόνης μας και με τον αισθητήρα θερμοκρασίας. Αρχικοποιούμε τις μεταβλητές i, meanTemp, screenTime, flag με κατάλληλες τιμές.

  Με τις εντολές pinMode που ακολουθούν ορίζουμε τα αντίστοιχα pins σε OUTPUT και με τις εντολές digitalWrite δίνουμε στα αντίστοιχα pins την αρχική ψηφιακή τιμή LOW. Καλούμε μια φορά τη συνάρτηση getTemp, ώστε να έχουμε εξ αρχής τιμές για τη θερμοκρασία μας.

  Ύστερα, με την εντολή timer.every(5000, getTemp), ορίζουμε να καλείται η συνάρτηση getTemp που αποθηκεύει την τωρινή θερμοκρασία κάθε 5000 ms, δηλαδή κάθε 5 δευτερόλεπτα από εδώ και στο εξής. Αυτό θα γίνεται σαν software interrupt μέσα στη συνάρτηση loop().

  Η παραπάνω συνάρτηση setup(), θα εκτελεστεί μόνο *μία* φορά, με το που αρχίσει η λειτουργία του κυκλώματός μας.

- **void loop():** Αρχικά, έχουμε την εντολή timer.tick(), η οποία ουσιαστικά προχωρά τον timer και ελέγχει αν έχει παρέλθει το διάστημα των 5 δευτερολέπτων ώστε να κληθεί η συνάρτηση getTemp(). Δηλαδή, η συνάρτηση getTemp() θα κληθεί στην αρχή της loop(), αν έχουν περάσει 5 δευτερόλεπτα από την προηγούμενη κλήση της.

  Στη συνέχεια, ελέγχουμε αν η καθολική μεταβλητή i είναι μεγαλύτερη από το 23, οπότε και θα έχουν ήδη αποθηκευτεί 24 μετρήσεις (2 λεπτά μετρήσεων) και θα πρέπει να καλέσουμε την συνάρτηση showAverage() για να εμφανίσουμε τον μέσο όρο και να προετοιμαστούμε για το επόμενο δίλεπτο.

  Ελέγχουμε επίσης αν το screenTime είναι διάφορο του μηδέν (δηλαδή έχει ληφθεί η πρώτη μέση τιμή), όπου σε αυτή την περίπτωση, ελέγχουμε αν έχουν περάσει 10 δευτερόλεπτα (millis()-screenTime>=10*1000) και καθαρίζουμε την οθόνη, σε περίπτωση που αυτό έχει συμβεί, ξανακάνοντας το screenTime ίσο με 0.

Ύστερα, ελέγχουμε αν η προηγούμενη θερμοκρασία έχει υπερβεί την επικίνδυνη τιμή (σταθερά DANGEROUS_TEMP), οπότε σε αυτή την περίπτωση θα κλείσει ο διακόπτης/ρελέ, θέτοντας το pin του ίσο με HIGH. Διαφορετικά, το pin του διακόπτη/ρελέ τίθεται ίσο με LOW, δηλαδή θα ανοίξει σε περίπτωση που ήταν κλειστός, μόλις η τιμή της θερμοκρασίας πέσει κάτω από την επικίνδυνη τιμή.

Στη συνέχεια ακολουθεί ο κώδικας για τον χειρισμό της υψηλής ή χαμηλής θερμοκρασίας. Αν η προηγούμενη θερμοκρασία, prevTemp, είναι μεγαλύτερη από την υψηλή οριακή τιμή (σταθερά HIGH_TEMP), τότε θέτουμε την καθολική μεταβλητή flag ίση με true (περιγράφεται παραπάνω η λειτουργία της), θέτουμε HIGH το pin που ελέγχει το LED υψηλής θερμοκρασίας και γράφουμε στην οθόνη μήνυμα ειδοποίησης ότι η θερμοκρασία έχει υπερβεί το μέγιστο όριο.

Αν, πάλι, η προηγούμενη θερμοκρασία είναι μικρότερη από την χαμηλή οριακή τιμή (σταθερά LOW_TEMP), τότε θέτουμε την καθολική μεταβλητή flag ίση με true, θέτουμε HIGH το pin που ελέγχει το LED χαμηλής θερμοκρασίας και γράφουμε στην οθόνη μήνυμα ειδοποίησης ότι η θερμοκρασία έχει πέσει κάτω από το ελάχιστο όριο.

Η τρίτη περίπτωση είναι η θερμοκρασία να βρίσκεται εντός των φυσιολογικών ορίων. Σε αυτή την περίπτωση, σίγουρα θα θέσουμε LOW τα pins που ελεγχουν και τα δύο LED. Επιπλέον, αν η μεταβλητή flag είναι ίση με true, σημαίνει ότι στην προηγούμενη εκτέλεση του loop η θερμοκρασία ήταν εκτός φυσιολογικών ορίων, αφου εκτελέστηκαν μία από τις δύο παραπάνω περιπτώσεις. Όμως, τώρα που επανήλθαμε ξανά εντός φυσιολογικών ορίων θερμοκρασίας, θα καθαρίσουμε την οθόνη ώστε να μην βγάζει πια την ειδοποίηση υψηλής/χαμηλής θερμοκρασίας, και θα θέσουμε ξανά το flag ίσο με false ώστε να είμαστε έτοιμοι για την επόμενη εκτέλεση του loop.

Τέλος, ακολουθεί ο κώδικας ελέγχου του αισθητήρα εγγύτητας. Αρχικά, λαμβάνουμε μια μέτρηση της απόστασης που εντοπίζει εκείνη τη στιγμή ο αισθητήρας εγγύτητας, με την εντολή:
unsigned int distance = sonar.ping_cm();
Τώρα, αν η distance είναι μικρότερη της απόστασης ενεργοποίησης (σταθερά ACTIVATION_DISTANCE), θα εμφανίσουμε στην πρώτη γραμμή της οθόνης την προηγούμενη θερμοκρασία (5 δευτερολέπτων), ενώ στην δεύτερη γραμμή θα εμφανίσουμε την μέση θερμοκρασία του προηγούμενου δίλεπτου. Επομένως, οι ενδείξεις της οθόνης θα ανανεώνονται κάθε φορά που ο αισθητήρας ανιχνεύει ηχητική ανάκλαση σε απόσταση μικρότερη του ACTIVATION_DISTANCE.
Να σημειωθεί πως για το πρώτο δίλεπτο, που δε θα υπάρχει ακόμα μέση τιμή δίλεπτου, η θερμοκρασία που θα εμφανίζεται στη δεύτερη γραμμή θα είναι μηδενική.

Η παραπάνω συνάρτηση loop(), θα εκτελείται *επ' αόριστον* για όσο θα συνεχίσει η λειτουργία του κυκλώματός μας.

- **bool getTemp():** Στην συνάρτηση αυτή αρχικά με την εντολή sensors.requestTemperatures() στέλνουμε εντολή για να αρχίσουμε να λαμβάνουμε θερμοκρασίες από τον αισθητήρα**.** Λαμβάνουμε με την εντολή sensors.getTempCByIndex(0) την θερμοκρασία σε βαθμούς Κελσίου εκείνη τη χρονική στιγμή και την αποθηκεύουμε στη θέση που δείχνει ο δείκτης i (καθολική μεταβλητή) του πίνακα temp.

  Με τις εντολές Serial.print("Celsius temperature: "), Serial.println(temp[i]) εμφανίζεται στην οθόνη του υπολογιστή μας η τιμή της θερμοκρασίας, για debugging.

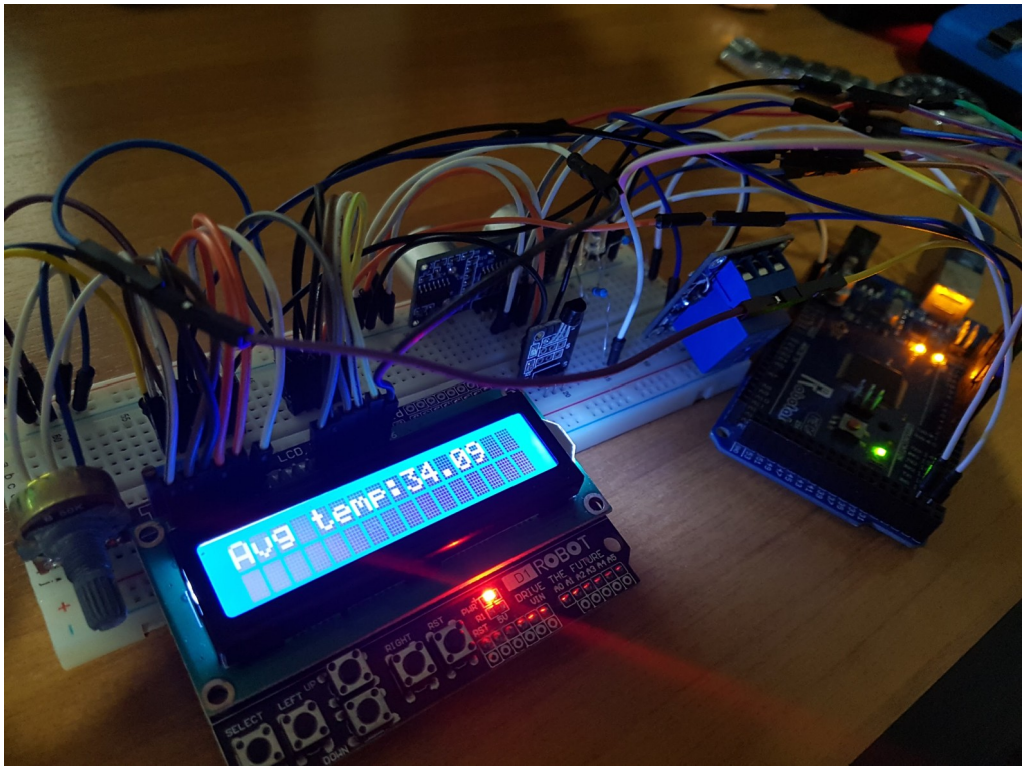
  Στη συνέχεια αποθηκεύουμε στη μεταβλητή prevTemp που αντιστοιχεί στην προηγούμενη θερμοκρασία την θερμοκρασία που έχουμε λάβει και αυξάνουμε το δείκτη i κατά 1 για να πάρουμε την επόμενη μέτρηση, μετά από 5 δευτερόλεπτα. Η συνάρτηση επιστρέφει true, που σημαίνει πως θέλουμε να συνεχιστεί η περιοδική εκτέλεσή της.

- **void showAverage():** Η συνάρτηση αυτή υπολογίζει και εμφανίζει το μέσο όρο της θερμοκρασίας. Με μία for loop υπολογίζουμε το άθροισμα των στοιχείων του πίνακα temp και στη συνέχεια το διαιρούμε με το 24 για να υπολογίσουμε το μέσο όρο, που αποθηκεύεται στην καθολική μεταβλητή meanTemp.

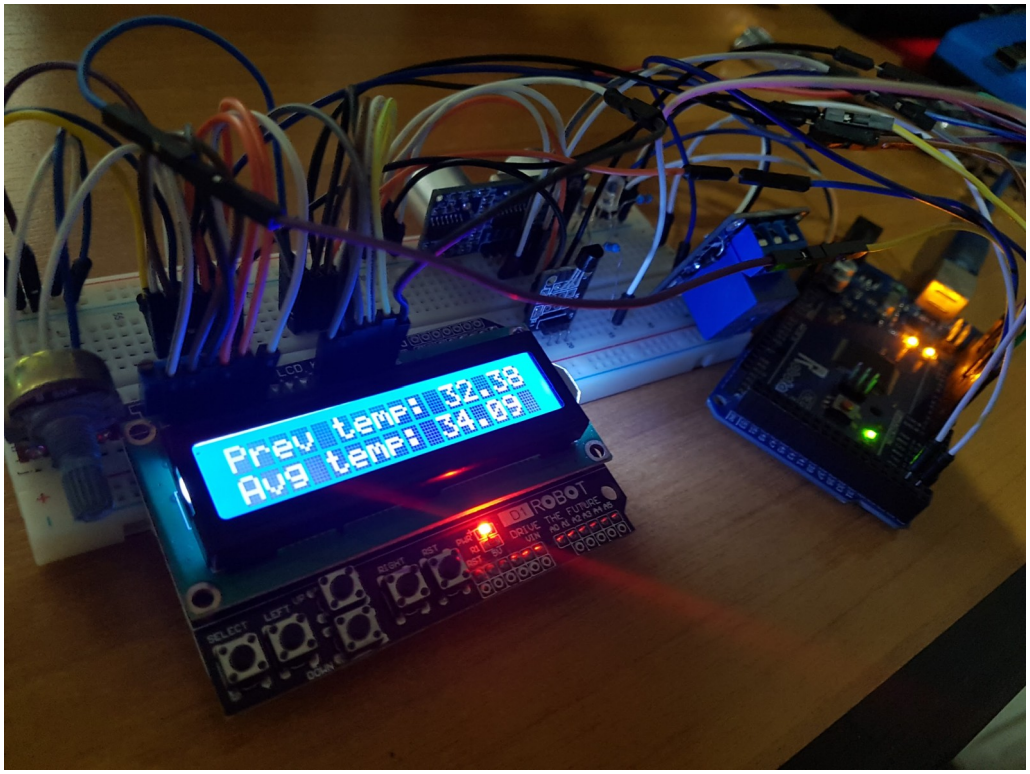
  Μηδενίζουμε το δείκτη i εφόσον έχει περάσει το δίλεπτο και θέλουμε να λάβουμε εκ νέου 24 μετρήσεις και στη συνέχεια με τις εντολές lcd.clear(), lcd.setCursor(0,0), lcd.print("Avg temp:"), lcd.print(meanTemp,2) καθαρίζουμε την lcd οθόνη, θέτουμε τον κέρσορα στην αρχή της οθόνης και εμφανίζουμε τη μέση τιμή της θερμοκρασίας.

  Τελικά, με την εντολή screenTime = millis() αποθηκεύουμε στη μεταβλητή screenTime την τωρινή χρονική στιγμή έτσι ώστε μέσα στη συνάρτηση loop να μπορέσουμε να καταλάβουμε αν έχουν περάσει τα 10 δευτερόλεπτα ώστε να καθαρίσουμε την οθόνη, όπως μας λέει η εκφώνηση.

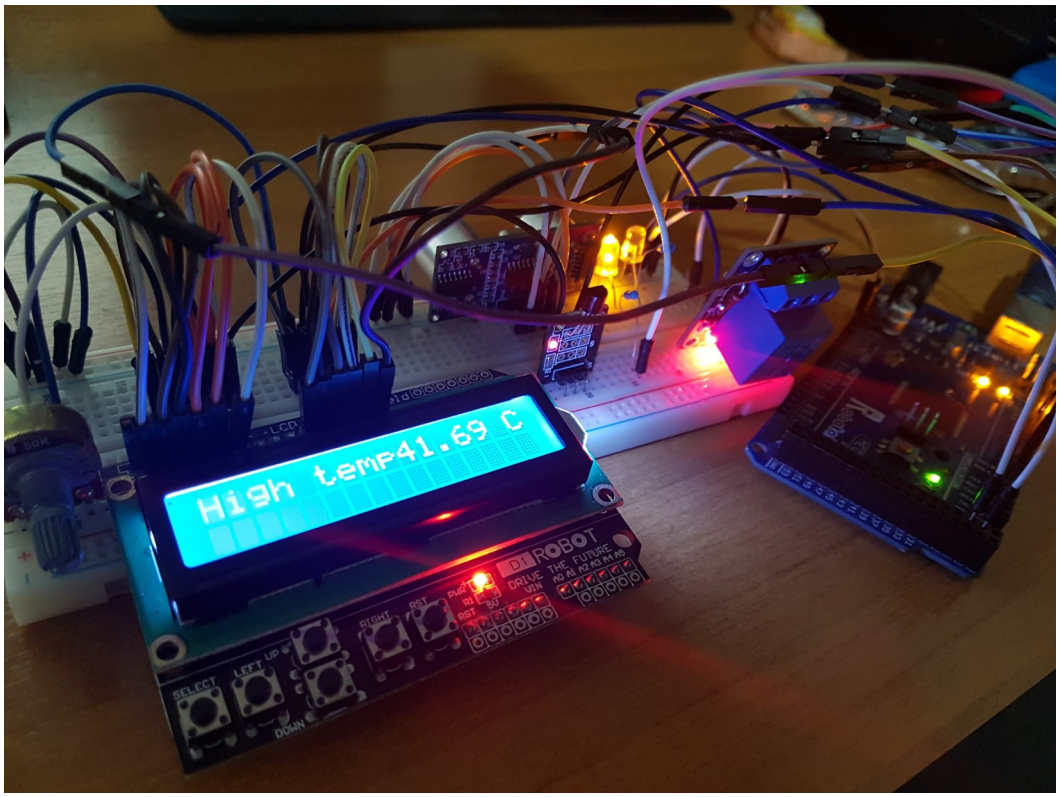
Ενδεικτικές φωτογραφίες του κυκλώματος



Εικόνα 1: Η ένδειξη της οθόνης αφού παρέλθει ένα δίλεπτο μετρήσεων



Εικόνα 2: Η ένδειξη της οθόνης αφού ενεργοποιηθεί ο αισθητήρας εγγύτητας



Εικόνα 3: Η ένδειξη της οθόνης αφού εντοπιστεί υψηλή θερμοκρασία. Παρατηρούμε επίσης πως ανάβει το πορτοκαλί LED, ενώ έχει ενεργοποιηθεί και ο διακόπτης/ρελέ.

Προβλήματα που αντιμετωπίστηκαν

1. Η οθόνη LCD δεν έκανε καλή επαφή με τα female wires που χρησιμοποιήσαμε και μας δυσκόλεψε στην εμφάνιση των αποτελεσμάτων και στη μετακίνηση όλου του κυκλώματος.

2. Ο αισθητήρας θερμοκρασίας λόγω κινέζικης προέλευσης είχε τοποθετημένα ανάποδα τα pins του ground με την είσοδο όποτε δεν μπορούσαμε αρχικά να πάρουμε σωστές μετρήσεις λόγω λανθασμένης συνδεσμολογίας.

3. Πρόβλημα στο ανέβασμα και στο κατέβασμα της θερμοκρασίας για να τεστάρουμε το κύκλωμα. Για το ανέβασμα της θερμοκρασίας χρησιμοποιήθηκε αναπτήρας ενώ δεν χρησιμοποιήσαμε κάποιο μέσο για το κατέβασμα.

4. Το κύκλωμα έγινε πολύπλοκο με τα καλώδια και όλες τις συνδέσεις τους και ήταν δύσκολο να εντοπιστούν πιθανές κακές συνδέσεις μεταξύ των καλωδίων.

5. Ο αισθητήρας εγγύτητας δεν ήταν αρκετά ακριβής και μερικές φορές εντόπιζε εμπόδιο χωρίς να υπάρχει, πιθανόν από λανθασμένες ανακλάσεις.

6. Σε περίπτωση που η θερμοκρασία ανέβει ή κατέβει από μία δεδομένη τιμή τότε εμφανίζεται στην οθόνη αντίστοιχο μήνυμα το οποίο παρατηρήσαμε στην οθόνη ότι δεν είναι σταθερό αλλά τρεμοπαίζει διότι εμφανίζεται για όλη τη διάρκεια των 5 δευτερολέπτων μέχρι να πάρει καινούργια τιμή. Αυτό οφείλεται στο γεγονός ότι το πρόγραμμα στέλνει συνέχεια το μήνυμα στην οθόνη για τα 5 δευτερόλεπτα με αποτέλεσμα να είναι ασταθές.

7. Η αρχική υλοποίηση του project έγινε σε πλατφόρμα Arduino Nano αλλά ήταν αδύνατο να γίνει upload το πρόγραμμα στην πλακέτα και για αυτό προτιμήσαμε Arduino Mega.

8. Δεν διαθέταμε μικροελεγκτή Arduino με Wi-Fi ή Wi-Fi module, επομένως δεν μπορούσαμε να υλοποιήσουμε τη λειτουργικότητα της αποστολής email.

Κώδικας

Ο κώδικας επισυνάπτεται στο αρχείο .zip που βρίσκεται και η αναφορά. Συγκεκριμένα, εντός του .zip βρίσκεται ο φάκελος του project, μέσα στον οποίο βρίσκεται το αρχείο Mikro2_project.ino. Επομένως, για την εκτέλεση του προγράμματος, το μόνο που έχει να κάνει κάποιος είναι να ανοίξει το αρχείο Mikro2_project.ino μέσω του Arduino IDE, και να το περάσει στην εσωτερική μνήμη της πλακέτας Arduino. (έχοντας κάνει πρώτα την κατάλληλη συνδεσμολογία)

Δεν υπάρχει compilation πολλών αρχείων, επομένως κρίνεται μή – απαραίτητη η προσθήκη Readme αρχείου.