ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΪΑΤΡΙΚΗΣ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΨΗΦΙΑΚΟΥ ΜΕΤΡΗΤΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΚΑΙ ΥΓΡΑΣΙΑΣ

ΙΩΑΝΝΗΣ ΞΕΚΑΡΔΑΚΗΣ
Αριθμός Μητρώου: 14068

Επιβλέπων Καθηγητής
Ιωάννης Βαλαής, Καθηγητής

Αθήνα, Οκτώβριος 2022

Η Τριμελής Εξεταστική Επιτροπή

Ο Επιβλέπων Καθηγητής		
Ιωάννης Βαλαής	Γεώργιος Φούντος	Χρήστος Μιχαήλ
Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής
[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]	[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]	[ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ]
Ioannis Valais Digitally signed by Ioannis Valais Date: 2022.10.13 21:34:59 +03'00'	Georgios Fountos Digitally signed by Georgios Fountos Date: 2022.10.14 13:45:14 +03'00'	Christos Michail Digitally signed by Christos Michail Date: 2022.10.14 08:31:55 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο υπογράφων Ξεκαρδάκης Ιωάννης του Λεωνίδας, με αριθμό μητρώου 14068 φοιτητής του Τμήματος Βιοϊατρικής της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου».

Ημερομηνία

13/10/22

Ο Δηλών

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και κατασκευή ενός ψηφιακού θερμομέτρου – υγρασιομέτρου το οποίο βασίζεται στην πλατφόρμα Arduino. Η συσκευή διαθέτει δύο αισθητήρες (ο μεν πρώτος χρησιμοποιείται για τη μέτρηση της θερμοκρασίας και υγρασίας ενός εσωτερικού χώρου, ο δε δεύτερος χρησιμοποιείται για τη μέτρηση της θερμοκρασίας του εξωτερικού χώρου) όπως επίσης και μία οθόνη LCD για την επικοινωνία με τον χρήστη. Ακόμα, υπάρχει η δυνατότητα καταγραφής των μέγιστων ημερήσιων τιμών σε μέσο μόνιμης αποθήκευσης (κάρτα micro-sd). Για τη διατήρηση της ημερομηνίας και της ώρας χρησιμοποιείται ένα ρολόι πραγματικού χρόνου. Τέλος, η συσκευή θα ειδοποιεί ηχητικά τον χρήστη μέσω ενός βομβητή και οπτικά μέσω μηνύματος στην οθόνη LCD σε περίπτωση όπου οι τιμές της εσωτερικής θερμοκρασίας, της υγρασίας ή της διαφοράς των θερμοκρασιών ξεπεράσει το όριο που έχει καθορίσει ο χρήστης.

Λέξεις-κλειδιά: Arduino, θερμοκρασία, υγρασία, οθόνη LCD, κάρτα micro-sd, ρολόι πραγματικού χρόνου

ABSTRACT

The object of this dissertation is the design and construction of a digital thermometer - hygrometer based on the Arduino platform. The device has two sensors (the first is used to measure the temperature and humidity of an indoor space, and the second is used to measure the temperature of the outdoor area) as well as an LCD screen for communication with the user. Also, it is possible to record the maximum daily values on a permanent storage medium (micro-sd card). A real-time clock is used to keep the date and time. Finally, the device will alert the user audibly via a buzzer and visually via a message on the LCD screen in case the values of internal temperature, humidity or temperature difference exceed the limit set by the user.

Λέξεις-κλειδιά: Arduino, temperature, humidity, LCD screen, micro-sd card, real time clock

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΕΙΣΑΓΩΓΗ

Στη σημερινή εποχή τα συστήματα αισθητήρων έχουν εγκατασταθεί πλήρως σε κάθε έκφανση της ζωής μας. Μία πολύ σημαντική χρήση τους είναι στον έλεγχο των συνθηκών που επικρατούν σε έναν χώρο, στη συγκεκριμένη περίπτωση στη μέτρηση της θερμοκρασίας και της υγρασίας του χώρου. Παράλληλα, η ευρεία διάδοση της πλατφόρμας Arduino έχει καταστήσει προσιτή, τόσο οικονομικά όσο και τεχνικά, την ανάπτυξη ηλεκτρονικών συστημάτων τα οποία να περιλαμβάνουν αισθητήρες διαφόρων ειδών. Σε αυτή τη βάση δημιουργήθηκε μία συσκευή η οποία περιλαμβάνει αισθητήρες για τον έλεγχο της θερμοκρασίας και της υγρασίας με επιπλέον δυνατότητες για τον έλεγχο και την προειδοποίηση για υπέρβαση των οριακών τιμών καθώς επίσης και την καταγραφή των μέγιστων τιμών.

Η ανάπτυξη της συσκευής περιλάμβανε τρία στάδια. Στο πρώτο στάδιο έγινε η σχεδίαση και η συναρμολόγηση του υλικού μέρους της συσκευής. Απαραίτητη ήταν η σωστή συνδεσμολογία όλων των απαραίτητων εξαρτημάτων για την ορθή ολοκλήρωση του δεύτερου σταδίου. Στο δεύτερο στάδιο πραγματοποιήθηκε το λογισμικό μέρος του συστήματος με την ανάπτυξη κώδικα για την πλατφόρμα Arduino. Στο τρίτο και τελευταίο στάδιο έλαβαν χώρα δοκιμές οι οποίες πιστοποίησαν την ορθή λειτουργία της συσκευής.

Το κείμενο διαρθρώνεται ως εξής:

Στο πρώτο κεφάλαιο γίνεται μια ανασκόπηση των διαφορετικών τύπων αισθητήρων θερμοκρασίας και υγρασίας που χρησιμοποιούνται στα διάφορα ηλεκτρονικά κυκλώματα παρακολούθησης των δύο αυτών φυσικών μεγεθών. Επίσης, παρουσιάζεται συνοπτικά η πλατφόρμα Arduino.

Στο δεύτερο κεφάλαιο γίνεται μία ανάλυση του Arduino Nano και των εξαρτημάτων που θα χρησιμοποιηθούν στη δημιουργία της συσκευής. Συγκεκριμένα, περιγράφεται ο τρόπος λειτουργίας και διασύνδεσής τους με το Arduino. Επίσης, αναλύονται τα διάφορα πρωτόκολλα που χρησιμοποιούνται για την επικοινωνία μεταξύ αυτών των διαφόρων μερών του συστήματος.

Στο τρίτο κεφάλαιο αναλύεται η μεθοδολογία που εφαρμόστηκε για τη δημιουργία της συσκευής. Αρχικά, γίνεται μια αναφορά στη σχεδίαση της συνδεσμολογίας του κυκλώματος. Έπειτα, παρουσιάζεται η πορεία συναρμολόγησης του τελικού κυκλώματος.

Στο τέταρτο κεφάλαιο αρχικά γίνεται μία αναφορά στο περιβάλλον προγραμματισμού και τις δυνατότητες που παρέχει η πλατφόρμα του Arduino. Έπειτα, αναφέρονται οι βιβλιοθήκες που είναι απαραίτητες για το πρόγραμμα του θερμομέτρου. Στη συνέχεια περιγράφεται η διαδικασία προγραμματισμού της πλακέτας. Τέλος, αναλύονται κάποια βασικά μέρη του προγράμματος της συσκευής.

Στο πέμπτο κεφάλαιο γίνεται έλεγχος λειτουργίας της συσκευής. Αρχικά ελέγχουμε ότι η απόκλιση των τιμών που λαμβάνουν οι δύο αισθητήρες θερμοκρασίας δεν επηρεάζουν την αξιοπιστία της συσκευής. Έπειτα, κάνουμε μία καταγραφή των τιμών που λαμβάνουν οι αισθητήρες για συγκεκριμένο χρονικό διάστημα.

Στο έκτο κεφάλαιο παρατίθενται οδηγίες ρύθμισης και χρήσης της συσκευής.

Στο έβδομο κεφάλαιο αναλύονται τα συμπεράσματα της εργασίας και πιθανές μελλοντικές βελτιώσεις.

ΚΕΦΑΛΑΙΟ 1^{ο} – ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο κεφάλαιο αυτό γίνεται μια ανασκόπηση των διαφορετικών τύπων αισθητήρων θερμοκρασίας και υγρασίας που χρησιμοποιούνται στα διάφορα ηλεκτρονικά κυκλώματα παρακολούθησης των δύο αυτών φυσικών μεγεθών. Επίσης, παρουσιάζεται η πλατφόρμα Arduino και αναφέρονται ενδεικτικά κάποιες από τις πλακέτες αυτής της πλατφόρμας καθώς επίσης και κάποια από τα shields[1] που υπάρχουν για τις πλακέτες αυτές και επεκτείνουν τις δυνατότητές τους.

1.1 Τύποι αισθητήρων θερμοκρασίας και υγρασίας

ΑΙΣΘΗΤΗΡΕΣ ΘΕΡΜΟΚΡΑΣΙΑΣ

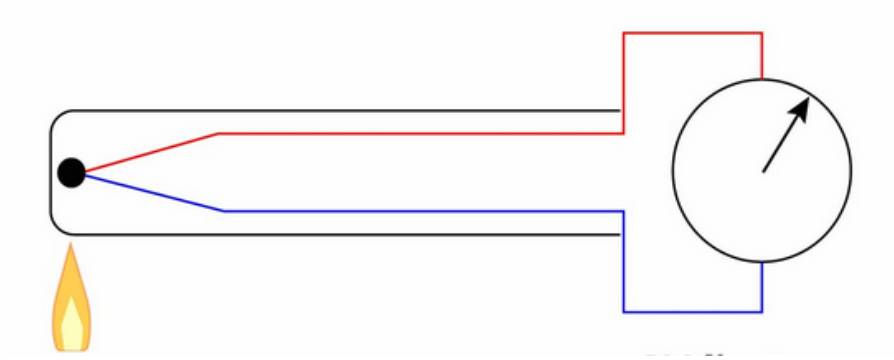
Οι αισθητήρες θερμοκρασίας είναι συσκευές οι οποίες μετράνε τη θερμοκρασία του περιβάλλοντος ή ενός σώματος και τη μετατρέπουν σε ηλεκτρικό σήμα[1], έτσι ώστε να μπορεί να είναι δυνατή η παρακολούθησή τους από κάποιο υπολογιστικό σύστημα. Οι τρεις πιο κοινοί τύποι αισθητήρων θερμοκρασίας είναι τα θερμοζεύγη, τα RTDs και τα thermistors.

Θερμοζεύγος

Το θερμοζεύγος είναι ένας μετατροπέας ενέργειας που μετατρέπει τη θερμική ενέργεια σε ηλεκτρική και αποτελείται από δύο μεταλλικά καλώδια τα οποία είναι κατασκευασμένα από δύο διαφορετικού είδους μέταλλα τα οποία στο ένα άκρο τους δημιουργούν μία ένωση. Όταν η θερμοκρασία μεταβληθεί στο σημείο της ένωσης αυτής, τότε στα δύο καλώδια αυτά παράγεται μία τάση.

Η ιδέα πίσω από το θερμοζεύγος βασίζεται στο φαινόμενο Seebeck[2], κατά το οποίο αν δύο διαφορετικά μέταλλα σχηματίσουν μία ένωση, τότε θα δημιουργήσουν μία μικρή αλλά μετρήσιμη τάση, όταν η θερμοκρασία στην ένωση αυτή μεταβληθεί. Το μέγεθος της τάσης εξαρτάται από τη θερμοκρασιακή μεταβολή και τα χαρακτηριστικά των μετάλλων.

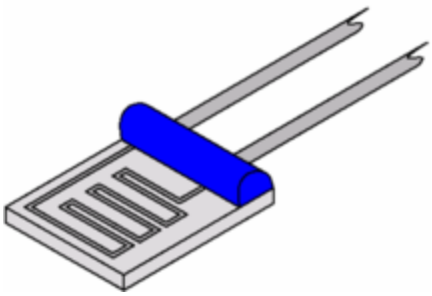
[1] Με τον όρο shield ονομάζεται μία πλακέτα η οποία συνδέεται με κάποια πλακέτα του Arduino με σκοπό να επεκτείνει τη λειτουργικότητά της δεύτερης



Εικόνα 1.1: Αρχή λειτουργίας θερμοζεύγους
(https://www.iqsdirectory.com/articles/thermocouple.html)

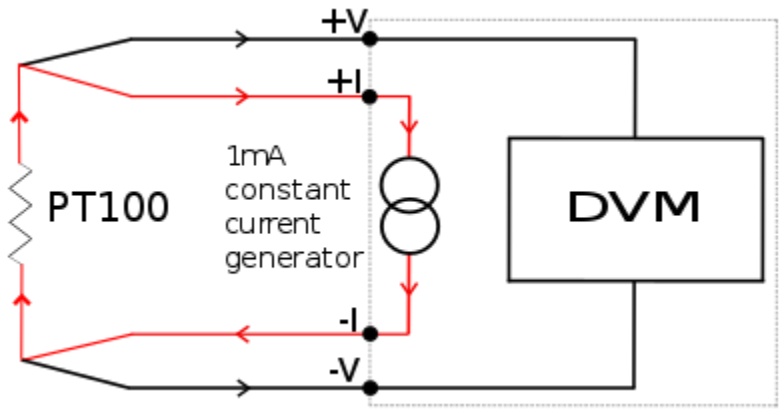
Αντιστατικός Αιθητήρας Θερμοκρασίας (Resistance Temperature Detector - RTD)

Το RTD είναι αισθητήρας θερμοκρασίας που αποτελείται από ένα λεπτό σύρμα τυλιγμένο γύρω από έναν κεραμικό ή γυάλινο πυρήνα[4]. Είναι επί της ουσίας ένα ωμόμετρο, το οποίο μετράει την αντίσταση του αισθητήρα. Για να επιτευχθεί αυτό, εφαρμόζει μία μικρή τάση στον αισθητήρα και μετράει την παραγόμενη τάση στα άκρα του αισθητήρα. Η αντίσταση του RTD μεταβάλλεται γραμμικά με τη μεταβολή της θερμοκρασίας.



Εικόνα 1.2: Κατασκευή RTD
(https://en.wikipedia.org/wiki/Resistance_thermometer)

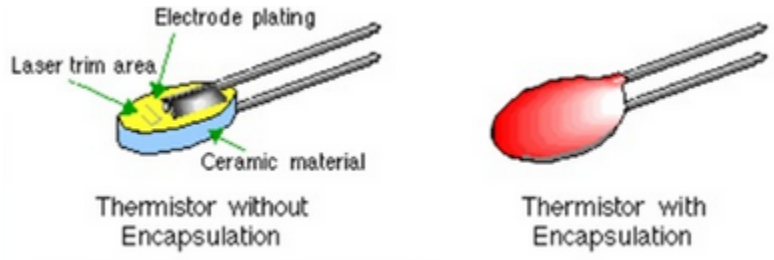
Συνήθως χρησιμοποιείται μία συνδεσμολογία με 4 καλώδια έτσι ώστε να ελαχιστοποιηθούν τα σφάλματα μέτρησης, 2 καλώδια είναι συνδεδεμένα με μια πηγή ρεύματος και διαρρέουν ρεύμα στον αισθητήρα και 2 για μέτρηση της τάσης που παράγεται στα άκρα του αισθητήρα.



Εικόνα 1.3: Κύκλωμα RTD
(https://www.wikiwand.com/en/Resistance_thermometer)

Θερμίστορ (Thermistor)

Το therimstor (thermal resistor) είναι μία αντίσταση της οποίας η τιμή μεταβάλλεται με τη μεταβολή της θερμοκρασίας[3]. Η μεταβολή της τιμής της αντίστασης δεν είναι γραμμική όπως στην περίπτωση του RTD. Τα thermistors είναι κατασκευασμένα από ημιαγώγιμα υλικά περικλεισμένα σε ένα γυάλινο ή εποξειδικό περίβλημα.



Εικόνα 1.4: Κατασκευή thermistor
(https://www.eng-tips.com/viewthread.cfm?qid=427486)

Υπάρχουν δύο τύποι thermistors, τα NTC και τα PTC. Στα NTC (Negative Temperature Coefficient) η τιμή της αντίστασης μειώνεται με την αύξηση της θερμοκρασίας. Στα PTC (Positive Temperature Coefficient) η τιμή της αντίστασης αυξάνεται με την αύξηση της θερμοκρασίας.

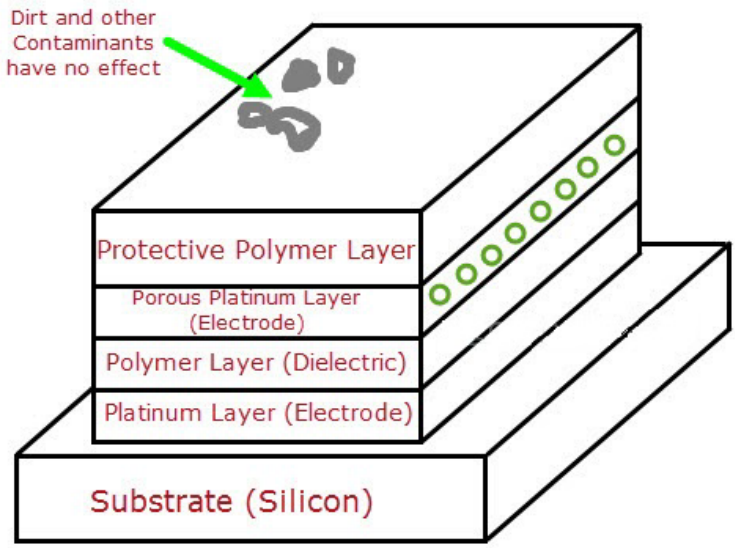
ΑΙΣΘΗΤΗΡΕΣ ΥΓΡΑΣΙΑΣ

Οι αισθητήρες υγρασίας είναι συσκευές οι οποίες μετράνε την υγρασία του περιβάλλοντος και μετατρέπουν τις μετρήσεις αυτές σε ένα ηλεκτρικό σήμα. Διακρίνονται σε δύο κατηγορίες όπου κάθε κατηγορία χρησιμοποιεί διαφορετική μέθοδο υπολογισμού της υγρασίας: τους αισθητήρες σχετικής υγρασίας και τους αισθητήρες απόλυτης θερμοκρασίας. Οι τρεις πιο κοινοί τύποι αισθητήρων είναι αυτοί που βασίζονται στη χωρητικότητα,

αυτοί που βασίζονται στην αντίσταση και οι θερμικοί (thermal)[5][6].

Χωρητικοί αισθητήρες (Capacitive sensors)

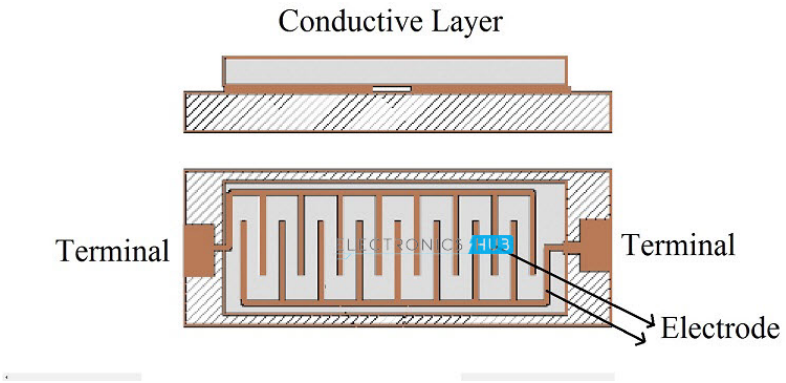
Οι χωρητικοί αισθητήρες χρησιμοποιούν δύο ηλεκτρόδια ανάμεσα από τα οποία υπάρχει τοποθετημένο ένα διηλεκτρικό στοιχείο[6]. Η διηλεκτρική σταθερά αυτής της πυκνωτικής διάταξης αυξάνεται ή μειώνεται με έναν ρυθμό ανάλογο της μεταβολής της υγρασίας στο περιβάλλον του αισθητήρα. Οι χωρητικοί αισθητήρες μεταβάλλουν τις τιμές τους γραμμικά με τη σχετική υγρασία.



Εικόνα 1.5: Κατασκευή χωρητικού αισθητήρα
(https://www.electronicshub.org/humidity-sensor-types-working-principle/)

Αισθητήρες αντίστασης

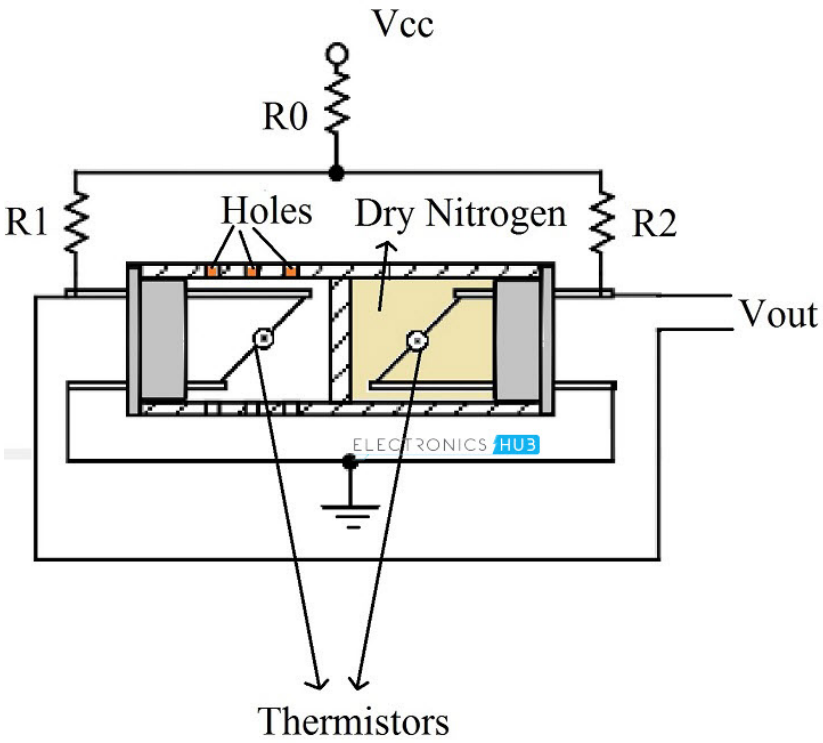
Οι αισθητήρες αντίστασης είναι άλλη μια σημαντική κατηγορία αισθητήρων θερμοκρασίας οι οποίοι μετράνε τη αντίσταση ή την ηλεκτρική αγωγιμότητα. Η αρχή στην οποία στηρίζουν τη λειτουργία τους είναι το γεγονός ότι η αγωγιμότητα των μη μεταλλικών αγωγών εξαρτάται από το περιεχόμενό τους σε νερό. Αυτοί αποτελούνται συνήθως από υλικά με σχετικά χαμηλή ειδική αντίσταση και η ειδική τους αντίσταση μεταβάλλεται σημαντικά με τη μεταβολή της υγρασίας. Η σχέση μεταξύ αντίστασης και σχετικής υγρασίας είναι αντιστρόφως ανάλογη[7]. Το υλικό αυτό χαμηλής ειδικής αντίστασης εναποτίθεται επάνω σε δύο ηλεκτρόδια. Η αντίσταση μεταξύ των δύο ηλεκτροδίων μεταβάλλεται όταν το άνω στρώμα της διάταξης αυτής απορροφά νερό και η μεταβολή αυτή μπορεί να μετρηθεί με τη βοήθεια ενός απλού ηλεκτρικού κυκλώματος.



Εικόνα 1.6: Κατασκευή αισθητήρα αντίστασης
(https://www.electronicshub.org/humidity-sensor-types-working-principle/)

Θερμικοί αισθητήρες

Οι θερμικοί αισθητήρες, οι οποίοι ονομάζονται και αισθητήρες απόλυτης υγρασίας μετράνε τη θερμική αγωγιμότητα του ξηρού αέρα καθώς επίσης και του αέρα ο οποίος περιέχει υγρασία[8]. Γνωρίζοντας τη διαφορά μεταξύ των δύο αυτών αγωγιμοτήτων υπολογίζεται η απόλυτη υγρασία. Οι αισθητήρες αυτοί αποτελούνται από δύο thermistor. Το ένα είναι κλεισμένο σε ξηρό άζωτο ενώ το άλλο είναι εκτεθειμένο στο περιβάλλον. Όταν ο αισθητήρας ενεργοποιείται, υπολογίζεται η αντίσταση των δύο στοιχείων και η διαφορά μεταξύ τους είναι ανάλογη της απόλυτης υγρασίας.



Εικόνα 1.7: Κατασκευή θερμικού αισθητήρα
(https://www.electronicshub.org/humidity-sensor-types-working-principle/)

1.2 Η πλατφόρμα Arduino

Το Arduino είναι μια πλατφόρμα ανοιχτού κώδικα, κύριο χαρακτηριστικό της οποίας είναι η ευκολία που παρέχει στο να δημιουργήσει κάποιος ένα κύκλωμα και να το προγραμματίσει. Οι πλακέτες αυτές επινοήθηκαν για γρήγορη ανάπτυξη πρωτοτύπων και για τη χρήση τους από άτομα τα οποία δεν έχουν ιδιαίτερο υπόβαθρο στα ηλεκτρονικά και τον προγραμματισμό.

Κάθε Arduino αποτελείται από μία πλακέτα ανάπτυξης, στην καρδιά της οποίας βρίσκεται ένας μικροελεγκτής. Επίσης, οι πλακέτες αυτές διαθέτουν ακροδέκτες ψηφιακής επικοινωνίας, αναλογικής επικοινωνίας, εισόδους για παροχή ρεύματος, κυκλώματα μνήμης και διάφορα άλλα ηλεκτρονικά εξαρτήματα τα οποία όλα μαζί συνθέτουν ένα σύνολο μιας πλακέτας που είναι κατά κάποιον τρόπο αντίστοιχη της μητρικής πλακέτας ενός υπολογιστή.

Οι πλακέτες Arduino μπορούν να χρησιμοποιηθούν σε μια πληθώρα εφαρμογών όπως για παράδειγμα στη ρομποτική, στο Διαδίκτυο Των Πραγμάτων (Internet Of Things) και στην καταγραφή δεδομένων από αισθητήρων. Μερικά από τα πλεονεκτήματα που παρέχουν είναι τα εξής:

- Οι πλακέτες Arduino είναι οικονομικά προσιτές και μπορεί κάποιος διαθέτοντας ένα μικρό χρηματικό ποσό να δημιουργήσει τα δικά του κυκλώματα.

- Το περιβάλλον ανάπτυξης προγραμμάτων για το Arduino (Arduino IDE) διαθέτει εκδόσεις για όλα τα κύρια λειτουργικά συστήματα (Windows, Linux, Mac OS)[9].

- Το Arduino IDE είναι πολύ φιλικό ακόμα και για άτομα που δεν έχουν εμπειρία στον προγραμματισμό, συνεπώς το να ξεκινήσει κάποιος τον προγραμματισμό της πλακέτας δεν απαιτεί πολύ χρόνο για την εκ των προτέρων εκμάθηση του περιβάλλοντος.

- Η γλώσσα προγραμματισμού του Arduino είναι ένα «παρακλάδι» της C++, που όμως διαθέτει διάφορα επίπεδα αφαίρεσης (abstraction levels) τα οποία κάνουν πιο εύκολη την εκμάθηση της. Επίσης, είναι δυνατή η συμπερίληψη διάφορων βιβλιοθηκών γραμμένων σε C++ που βοηθούν την επέκταση των προγραμμάτων. Συνεπώς ο προγραμματισμός της πλακέτας γίνεται πολύ πιο εύκολος και προσιτός σε σχέση με τον προγραμματισμό ενός μικροελεγκτή με γλώσσα Assembly.

- Η κατασκευή των πλακετών Arduino είναι ανοιχτού κώδικα, συνεπώς τα σχέδια τους είναι ελεύθερα και μπορεί όποιος επιθυμεί να δημιουργήσει τη δική του εκδοχή της πλακέτας, τόσο για

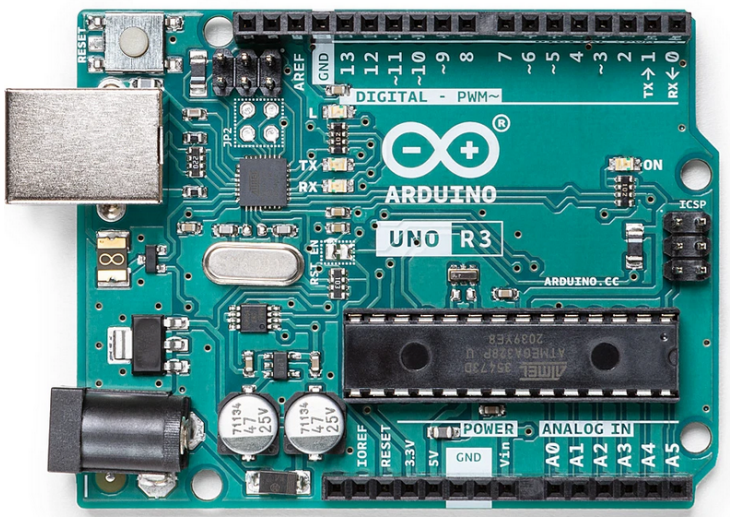
πειραματισμό όσο και για βελτίωση ή επέκταση της αρχικής εκδοχής[10].

- Κάθε πλακέτα διαθέτει αποθηκευμένο στην μνήμη της έναν bootloader, ο οποίος σε συνδυασμό με το IDE επιτρέπει τη φόρτωση της μνήμης του Arduino με το πρόγραμμα που γράψαμε. Η διαδικασία αυτή γίνεται με τη σύνδεση του Arduino στον υπολογιστή με USB. Με αυτόν τον τρόπο, δεν απαιτείται να έχουμε κάποια εξειδικευμένη συσκευή (memory programmer) προκειμένου να πραγματοποιήσουμε τη μεταφόρτωση αυτή.

1.3 Είδη πλακετών Arduino

Στην ενότητα αυτή παρουσιάζονται διάφορα είδη πλακετών της σειράς Arduino[9]. Οι πλακέτες αυτές δημιουργήθηκαν με σκοπό να εξυπηρετήσουν με τον καλύτερο δυνατό τρόπο τις κατασκευές ή λειτουργίες για τις οποίες θα χρησιμοποιηθούν. Οι διαφορές μεταξύ τους εστιάζονται κυρίως στον μικροελεγκτή που διαθέτουν, στη χωρητικότητα της μνήμης τους και στο πλήθος των ακροδεκτών επικοινωνίας τους. Επίσης, κάποια μοντέλα διαθέτουν επιπλέον χαρακτηριστικά σε σχέση με κάποια άλλα, γεγονός που τα καθιστά καταλληλότερα για πιο πολύπλοκα κυκλώματα. (Σημείωση: Το μοντέλο που θα χρησιμοποιηθεί στο κύκλωμα της εργασίας είναι το Arduino Nano και θα γίνει λεπτομερής τεχνική ανάλυση για αυτό στο επόμενο κεφάλαιο) Τα σημαντικότερα μοντέλα είναι τα εξής:

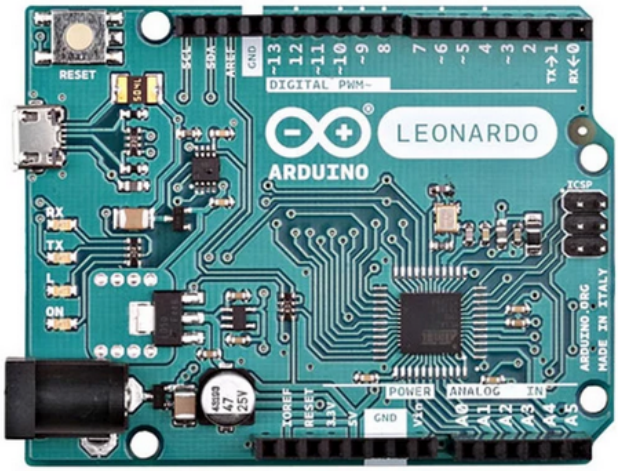
<u>Uno</u>



Εικόνα 1.8: Arduino Uno
(https://www.arduino.cc/en/main/products)

Το Uno διαθέτει μικροελεγκτή Atmega 328p, Flash Memory 32KB, SRAM 2KB, EEPROM 1KB και 20 ακροδέκτες επικοινωνίας (14 ψηφιακής και 6 αναλογικής).

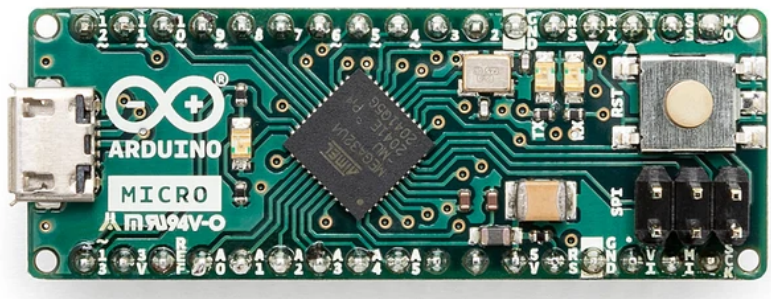
Leonardo



Εικόνα 1.9: Arduino Leonardo
(https://www.arduino.cc/en/main/products)

Το Leonardo διαθέτει μικροελεγκτή Atmega 328u4, Flash Memory 32KB, SRAM 2.5KB, EEPROM 1KB και 20 ακροδέκτες επικοινωνίας από τους οποίους οι 12 μπορούν να χρησιμοποιηθούν για αναλογική επικοινωνία.

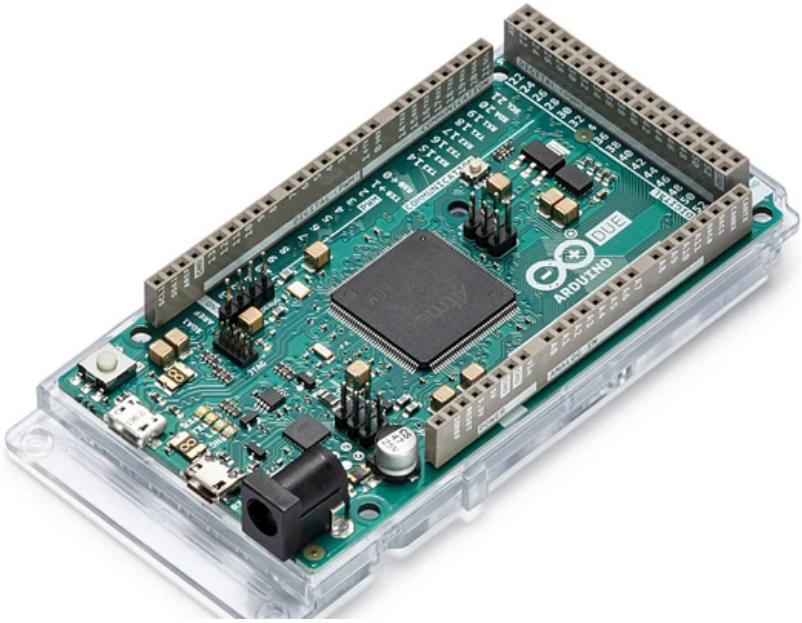
Micro



Εικόνα 1.10: Arduino Micro
(https://www.arduino.cc/en/main/products)

Το Micro διαθέτει μικροελεγκτή Atmega 328u4, Flash Memory 32KB, SRAM 2.5KB, EEPROM 1KB και 20 ακροδέκτες επικοινωνίας από τους οποίους οι 12 μπορούν να χρησιμοποιηθούν για αναλογική επικοινωνία.

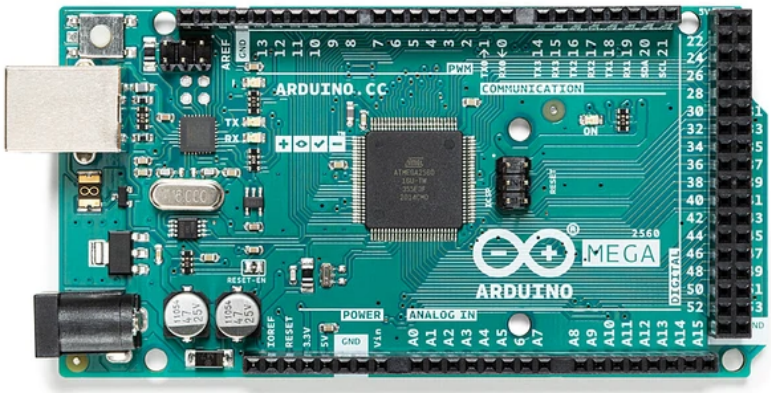
Due



Εικόνα 1.11: Arduino Due
(https://www.arduino.cc/en/main/products)

Το Due διαθέτει μικροελεγκτή Atmel SAM3X8E ARM Cortex-M3 CPU, Flash Memory 512KB, SRAM 94KB, 54 ακροδέκτες ψηφιακής επικοινωνίας και 12 αναλογικής επικοινωνίας.

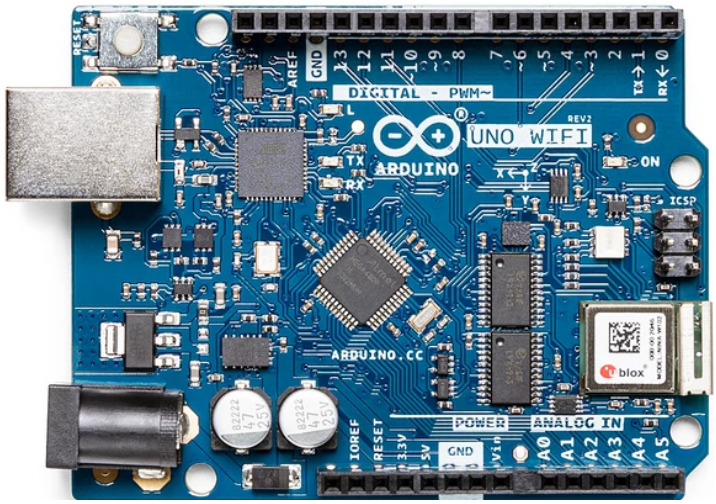
Mega 2560



Εικόνα 1.12: Arduino Mega 2560
(https://www.arduino.cc/en/main/products)

Το Mega 2560 διαθέτει μικροελεγκτή ATmega 2560, Flash Memory 256KB, SRAM 8KB, EEPROM 4KB, 54 ακροδέκτες ψηφιακής επικοινωνίας και 16 αναλογικής επικοινωνίας.

Uno WiFi

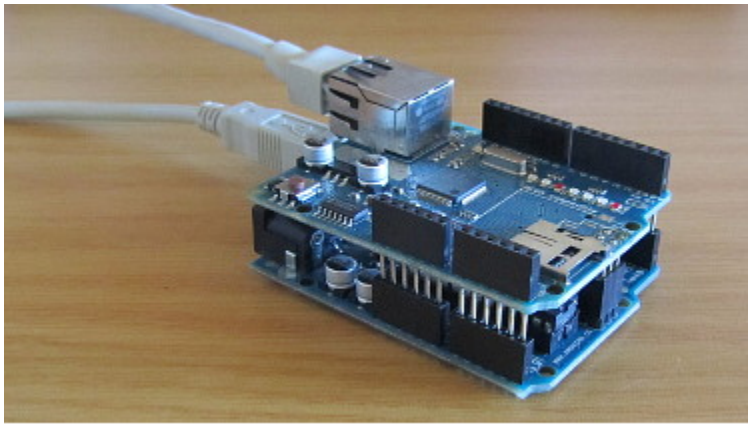


Εικόνα 1.13: Arduino Uno WiFi
(https://www.arduino.cc/en/main/products)

Το Uno WiFi διαθέτει μικροελεγκτή ATmega 4809, Flash Memory 48KB, SRAM 6144 Bytes, EEPROM 256 Bytes και 14 ακροδέκτες ψηφιακής επικοινωνίας και 6 αναλογικής. Επίσης διαθέτει δυνατότητα επικοινωνίας μέσω WiFi και Bluetooth.

1.4 Είδη Arduino shields

Όπως έχει ήδη αναφερθεί, η πλειοψηφία των διαφορετικών πλακετών Arduino παρέχει τη βασική λειτουργικότητα ενός κυκλώματος, δηλαδή ένα μικροελεγκτή, μία σειρά από μνήμες και ακροδέκτες επικοινωνίας με τα υπόλοιπα λειτουργικά μέρη του κυκλώματος, πχ αισθητήρες. Προκειμένου να δώσουμε στο Arduino κάποιες επιπλέον λειτουργικότητες τις οποίες δε διαθέτει η βασική πλακέτα (για παράδειγμα συνδεσιμότητα σε δίκτυο Ethernet) και παράλληλα να μην αυξήσουμε σε μεγάλο βαθμό τον όγκο του κυκλώματος, μπορούν να χρησιμοποιηθούν κάποιες επιπρόσθετες πλακέτες οι οποίες έχουν τις ίδιες διαστάσεις με τις βασικές πλακέτες του Arduino και «στοιβάζονται» πάνω τους, οι οποίες είναι ευρύτερα γνωστές με τον όρο «shields». Με αυτόν τον τρόπο δημιουργείται σύνδεση μεταξύ των απαραίτητων ακροδεκτών και παράλληλα επεκτείνεται η λειτουργικότητα της πλακέτας.

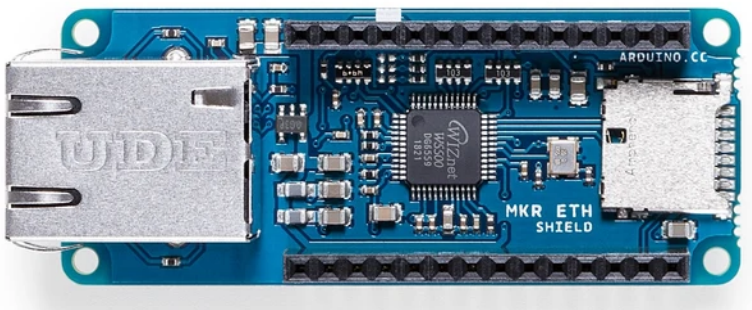


Εικόνα 1.14: Σύνθεση πλακέτας Arduino με shield
(https://www.arduino.cc/en/main/products)

Για τον σκοπό αυτό έχει δημιουργηθεί μια πληθώρα από shields οι οποίες παρέχουν πολλές και διαφορετικές λειτουργικότητες. Μερικές από τις πιο συνηθισμένες είναι οι εξής:

Ethernet Shield

Το shield αυτό χρησιμοποιείται όταν επιθυμούμε να προσθέσουμε στο κύκλωμά μας τη δυνατότητα να συνδέεται σε κάποιο ενσύρματο δίκτυο μέσω ενός καλωδίου Ethernet. Το shield παρέχει τη συνδεσιμότητα σε φυσικό επίπεδο και μέσω της πλακέτα Arduino μπορούμε να υλοποιήσουμε προγράμματα που υλοποιούν κάποια σύνδεση σε τοπικό δίκτυο ακόμα και στο Internet.

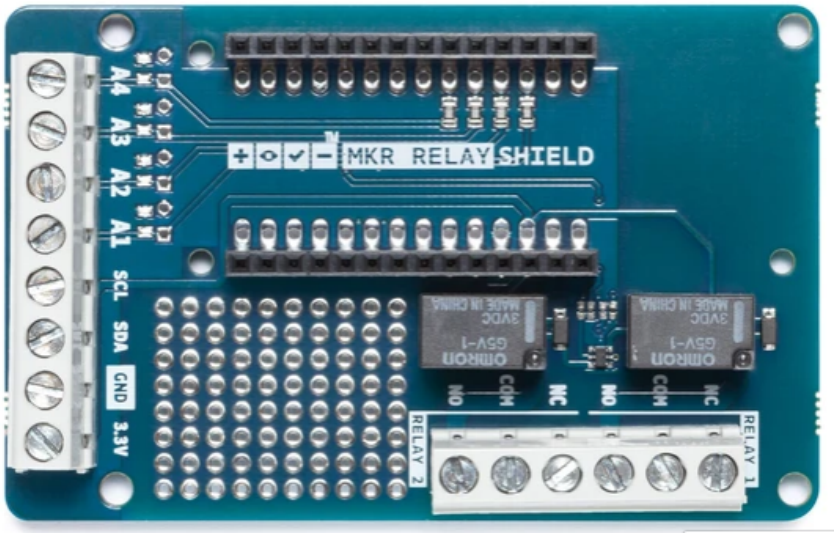


Εικόνα 1.15: Ethernet shield
(https://www.arduino.cc/en/main/products)

Relay Proto Shield

Στο shield αυτό περιέχονται δύο ρελέ τύπου SPDT (Single Pole Double Throw) τα οποία μπορούν να χρησιμοποιηθούν με ευκολία στα κυκλώματα, ελέγχοντας τα με τη χρήση ενός ακροδέκτη αντίστοιχα. Επίσης, στο συγκεκριμένο shield παρέχεται ελεύθερος χώρος για τη σύνδεση επιπλέον

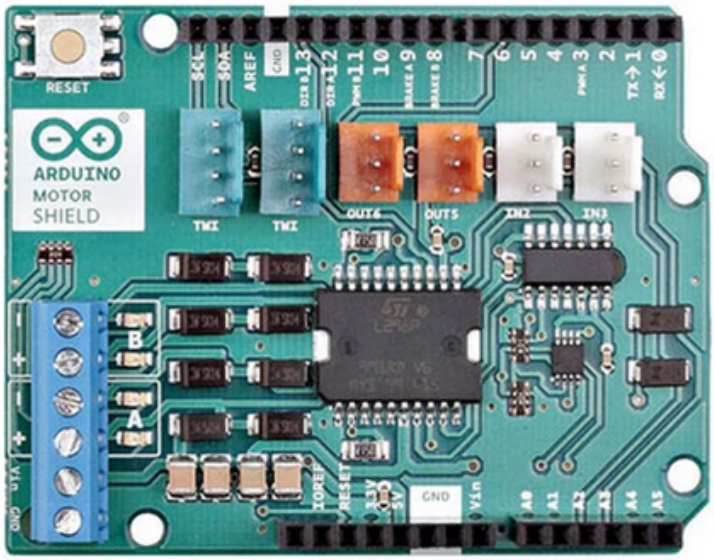
εξαρτημάτων τόσο με το shield όσο και με την πλακέτα του Arduino.



Εικόνα 1.16: Relay Proto Shield
(https://www.arduino.cc/en/main/products)

Motor Shield

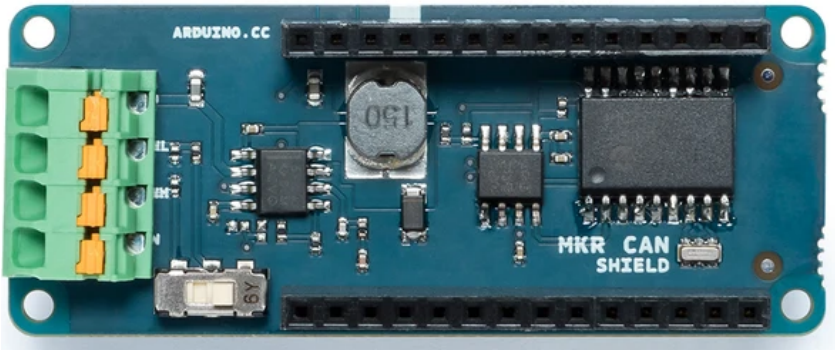
Το shield αυτό στηρίζει τη λειτουργία του στο ολοκληρωμένο L298, το οποίο χρησιμοποιείται για την οδήγηση επαγωγικών φορτίων όπως ρελέ, ηλεκτροβάνες και κινητήρες συνεχούς ρεύματος (DC). Το συγκεκριμένο shield επιτρέπει την οδήγηση δύο DC κινητήρων ταυτόχρονα, προσφέροντας τη δυνατότητα ελέγχου τόσο της ταχύτητάς τους όσο και της κατεύθυνσης τους.



Εικόνα 1.17: Motor Shield
(https://www.arduino.cc/en/main/products)

MRK 485 Shield

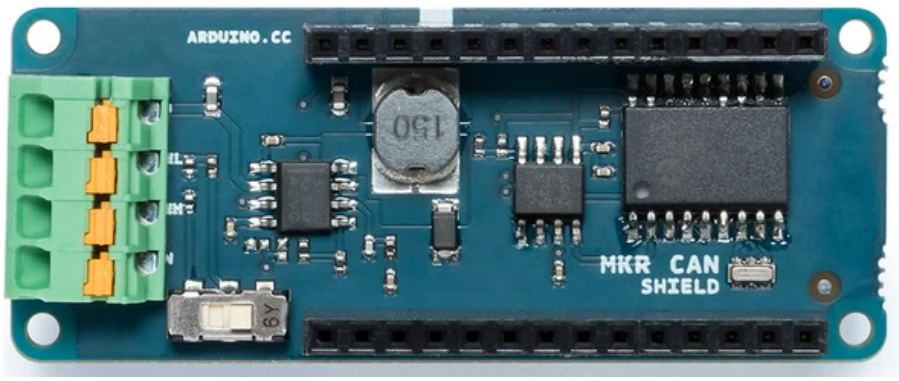
Το shield αυτό προσφέρει τη δυνατότητα σύνδεσης του Arduino με συσκευές οι οποίες επικοινωνούν μέσω του πρωτοκόλλου RS485. Τέτοιες συσκευές είναι συνήθως βιομηχανικού επιπέδου και αφορούν συστήματα κλιματισμού, μηχανολογικό εξοπλισμό και προγραμματιζόμενους λογικούς ελεγκτές (PLC).



Εικόνα 1.18: MKR 485 Shield
(https://www.arduino.cc/en/main/products)

CAN Shield

Το shield αυτό προσφέρει τη δυνατότητα σύνδεσης του Arduino με συσκευές που χρησιμοποιούν το πρωτόκολλο επικοινωνίας CAN (Controller Area Network). Το πρωτόκολλο αυτό συναντάται στην επικοινωνία με συσκευές βιομηχανικού επιπέδου καθώς επίσης και με οχήματα.



Εικόνα 1.19: CAN Shield
(https://www.arduino.cc/en/main/products)

ΚΕΦΑΛΑΙΟ 2^{ο} – ΕΠΙΛΟΓΗ ΥΛΙΚΩΝ ΚΑΤΑΣΚΕΥΗΣ

Στο κεφάλαιο αυτό γίνεται μία ανάλυση του Arduino Nano και των εξαρτημάτων που θα χρησιμοποιηθούν στη δημιουργία της κατασκευής. Συγκεκριμένα, περιγράφεται ο τρόπος λειτουργίας και διασύνδεσής τους με το Arduino. Επίσης, αναλύονται τα διάφορα πρωτόκολλα που χρησιμοποιούνται για την επικοινωνία μεταξύ αυτών των διαφόρων μερών του συστήματος.

2.1 Τεχνικά χαρακτηριστικά της πλακέτας Arduino Nano

Το Arduino Nano είναι ένα low end μέλος της σειράς των πλακετών Arduino. Η λειτουργία του βασίζεται στον 8-bit μικροελεγκτή Atmega328p. Στην πλακέτα περιλαμβάνονται 14 ακροδέκτες ψηφιακής επικοινωνίας, 8 αναλογικής, 2 ακροδέκτες reset και 6 ισχύος. Επίσης, περιλαμβάνονται μία θύρα Micro USB Type-B για την τροφοδοσία της πλακέτας και τον προγραμματισμό της και 6 ακροδέκτες με τους οποίους δίνεται η δυνατότητα για εναλλακτικό τρόπο προγραμματισμού της πλακέτας (ICSP – In Circuit Serial Programming).

Η τάση λειτουργίας του Arduino Nano είναι 5V μέσω της θύρας USB, αλλά μπορεί να τροφοδοτηθεί μέσω κάποιου από τους ακροδέκτες ισχύος του με τάση από 7 έως 12V. Κάθε ακροδέκτης εισόδου/εξόδου μπορεί να παρέχει ρεύμα έως 40mA στο φορτίο με το οποίο συνδέεται.

Καθένας από τους ακροδέκτες επικοινωνίας μπορούν να λειτουργήσουν είτε σαν είσοδος, όταν θέλουμε να συνδέσουμε κάποιον αισθητήρα σε αυτούς, είτε σαν έξοδος όταν θέλουμε να παρέχουμε σήματα σε κάποια άλλη συσκευή ή να οδηγήσουμε κάποιο φορτίο. Οι ακροδέκτες αναλογικής επικοινωνίας παρέχουν διακριτική ικανότητα 10 bits (δηλαδή 1024 διαφορετικές τιμές). Το προκαθορισμένο διάστημα τάσης για τις μετρήσεις που λαμβάνουν οι ακροδέκτες αναλογικής επικοινωνίας είναι από 0 έως 5V, ωστόσο το άνω όριο μπορεί να διαφοροποιηθεί με τη χρήση του ακροδέκτη AREF.

Το Arduino Nano μπορεί να υποστηρίξει τις ακόλουθες μορφές επικοινωνίας:

- Serial: Χρησιμοποιεί τους ακροδέκτες 0 (RX) και 1 (TX). Χρησιμοποιείται για να λάβει και να αποστείλει TTL (Transistor – Transistor Logic) σειριακά δεδομένα. Οι ακροδέκτες αυτοί είναι συνδεδεμένοι στους αντίστοιχους του ολοκληρωμένου FTDI το οποίο μετατρέπει τα σήματα TTL σε σήματα επικοινωνίας μέσω USB.
- SPI: Για αυτό το πρωτόκολλο επικοινωνίας χρησιμοποιούνται οι ακροδέκτες (με τα αντίστοιχα σήματα σε παρένθεση) 10(SS),

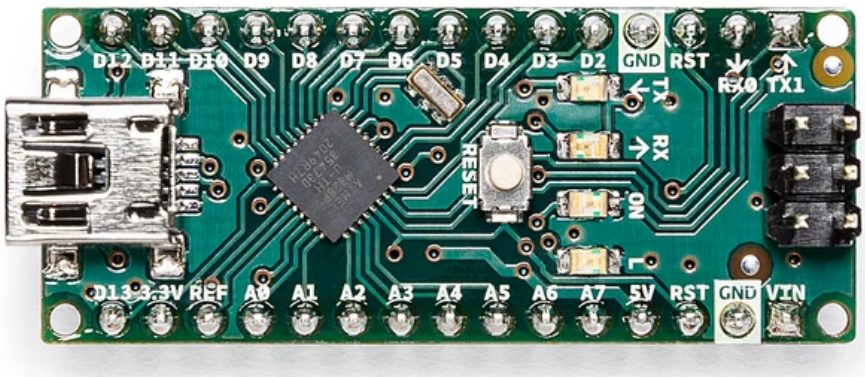
11(MOSI), 12(MISO) και 13(SCK).

- I2C: Για αυτό το πρωτόκολλο επικοινωνίας χρησιμοποιούνται οι ακροδέκτες (με τα αντίστοιχα σήματα σε παρένθεση) A4(SDA) και A5(SDL).
- PWM (Pulse Width Modulation): Χρησιμοποιούνται οι ακροδέκτες 3, 5, 6, 9, 10 και 11 οι οποίοι μπορούν να παρέχουν τετραγωνικούς παλμούς των οποίων η διαμόρφωση μπορεί να έχει 256 διαφορετικές στάθμες (8 bit resolution).
- External Interrupts: Χρησιμοποιούνται οι ακροδέκτες 2 και 3 για τη λήψη σημάτων που θα ενεργοποιήσουν κάποιου είδους διακοπή.

Οι μνήμες που διαθέτει είναι οι εξής:

- Flash Memory μεγέθους 32KB, η οποία χρησιμοποιείται για την αποθήκευση του κώδικα του Arduino. Τα 2 KB της flash memory χρησιμοποιείται για την αποθήκευση του bootloader της πλακέτας.
- SRAM (Static Random Access Memory) μεγέθους 2KB, η οποία χρησιμοποιείται από το πρόγραμμα του Arduino κατά την εκτέλεσή του.
- EEPROM (Electronically Erasable Programmable Read Only Memory) μεγέθους 1KB, το οποίο μπορεί να χρησιμοποιηθεί ως ένας μικρός αποθηκευτικός χώρος.

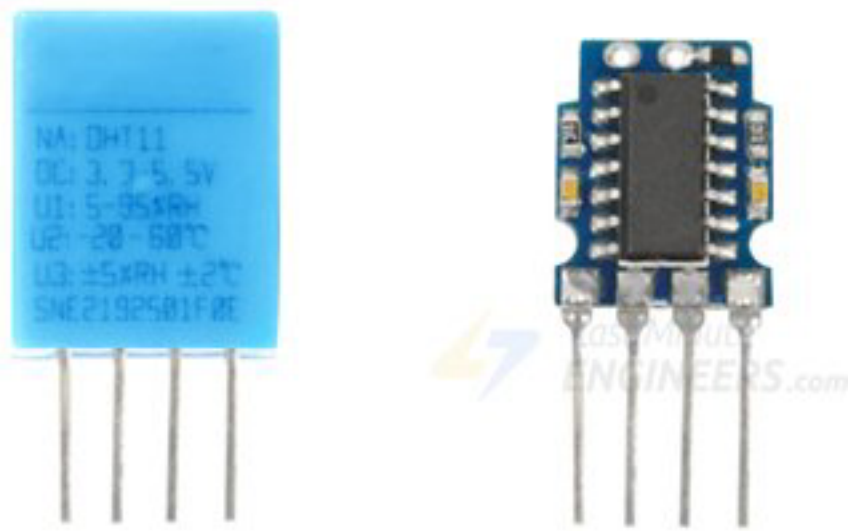
Όπως είναι φυσικό, οι μνήμες Flash και EEPROM είναι μη πτητικές, δηλαδή τα δεδομένα που υπάρχουν σε αυτές δε χάνονται με την απουσία ισχύος. Αντιθέτως, η SRAM είναι πτητική.[11]



Εικόνα 2.1: Ακροδέκτες του Arduino Nano
(https://docs.arduino.cc/hardware/nano)

2.2 Διάταξη (module)[2] αισθητήρα DHT11

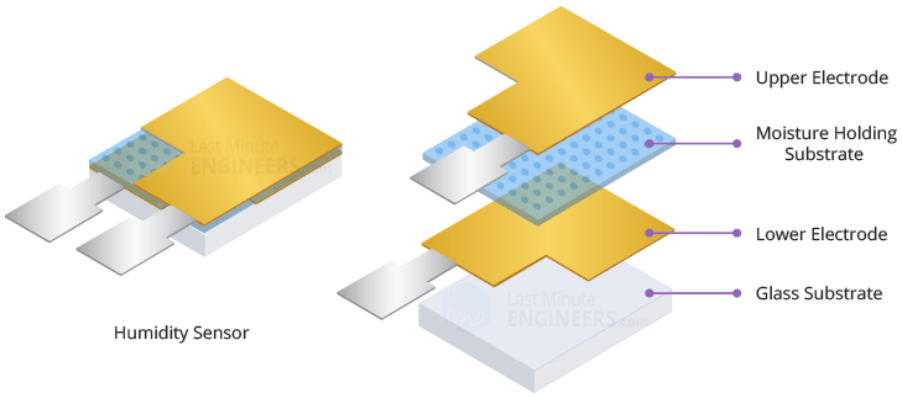
Το module του αισθητήρα DHT11 είναι ένα σύνθετο εξάρτημα το οποίο αποτελείται από τρία μέρη: ένα στοιχείο υπολογισμού της υγρασίας, ένα thermistor και ένα κύκλωμα υποστήριξης του module.[12]



Εικόνα 2.2: Συστατικά μέρη του DHT11
(https://lastminuteengineers.com/dht11-module-arduino-tutorial/)

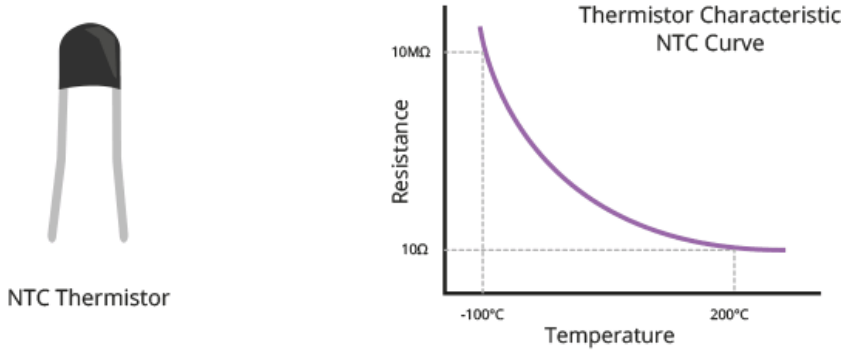
Το στοιχείο υπολογισμού της υγρασίας περιέχει 2 ηλεκτρόδια, ανάμεσα από τα οποία βρίσκεται ένα υπόστρωμα το οποίο απορροφά υγρασία. Καθώς το υπόστρωμα απορροφά υγρασία από τον χώρο στον οποίο είναι τοποθετημένο, απελευθερώνονται ιόντα και αυξάνεται η αγωγιμότητα μεταξύ των δύο ηλεκτροδίων. Η αλλαγή στην αντίσταση μεταξύ των δύο ηλεκτροδίων είναι ανάλογη της σχετικής υγρασίας. Με άλλα λόγια, υψηλότερη σχετική υγρασία μειώνει την αντίσταση μεταξύ των ηλεκτροδίων, ενώ χαμηλότερη σχετική υγρασία αυξάνει την αντίσταση μεταξύ των ηλεκτροδίων.

[2] Το module (ή διάταξη) αποτελεί αυτοτελή και ανεξάρτητη λειτουργική μονάδα η οποία επιτελεί κάποια συγκεκριμένη λειτουργία στο κύκλωμα



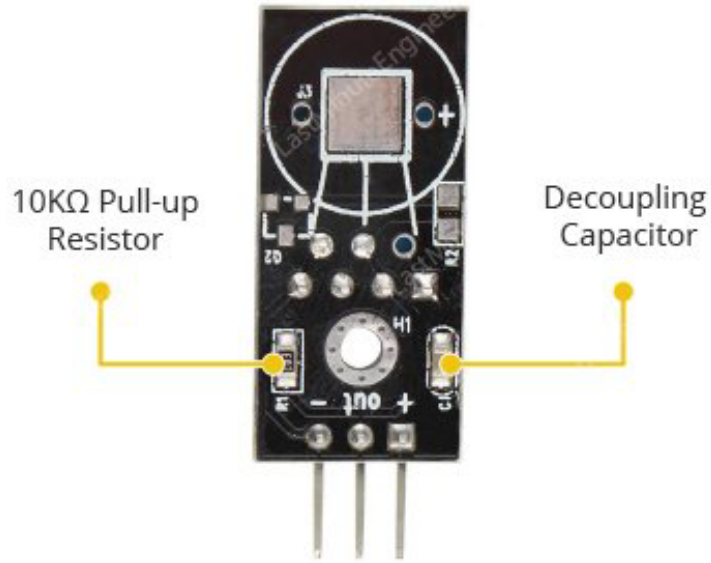
Εικόνα 2.3: Συστατικά μέρη στοιχείου υπολογισμού της υγρασίας
(https://lastminuteengineers.com/dht11-module-arduino-tutorial/)

Το thermistor που περιλαμβάνεται στον αισθητήρα είναι μία θερμική αντίσταση, η οποία αλλάζει τιμή ανάλογα με τη θερμοκρασία. Το συγκεκριμένο thermistor χαρακτηρίζεται και από τον όρο "Negative Temperature Coefficient" (NTC) (βλ. παράγραφο 1.1).



Εικόνα 2.4: Thermistor και χαρακτηριστική καμπύλη αντίστασης-θερμοκρασίας
(https://lastminuteengineers.com/dht11-module-arduino-tutorial/)

Τελευταίο συστατικό μέρος του module είναι το κύκλωμα υποστήριξης, το οποίο είναι υπεύθυνο για την ορθή λειτουργία του (επιτρέποντας το να λειτουργήσει χωρίς την προσθήκη επιπλέον ηλεκτρονικών εξαρτημάτων) και την επικοινωνία με το Arduino. Αποτελείται από ένα 8-bit SOIC-14 ολοκληρωμένο, το οποίο λαμβάνει και επεξεργάζεται τα αναλογικά σήματα του thermistor και του στοιχείου υπολογισμού της υγρασίας, πραγματοποιεί τη μετατροπή των αναλογικών σημάτων σε ψηφιακά και μεταδίδει τα ψηφιακά αυτά σήματα. Επίσης, στο ίδιο κύκλωμα περιλαμβάνονται μία αντίσταση 10ΚΩ, η οποία είναι απαραίτητη για την ορθή επικοινωνία του αισθητήρα με το Arduino και ένας πυκνωτής για το φιλτράρισμα των ηλεκτρικών θορύβων στην τροφοδοσία του αισθητήρα.



Εικόνα 2.5: Κύκλωμα υποστήριξης DTH11
(https://lastminuteengineers.com/dht11-module-arduino-tutorial/)

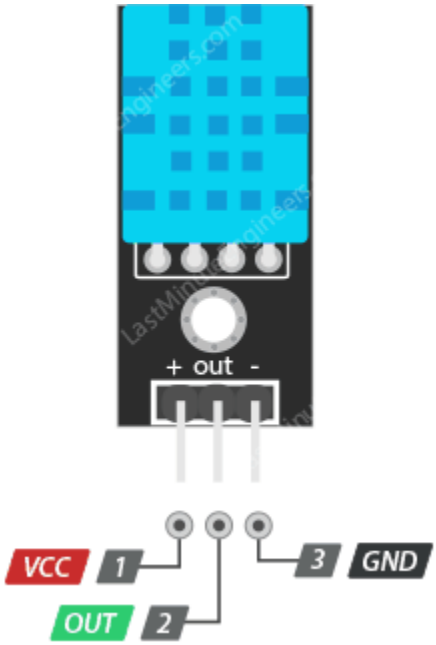
Τα όρια μέτρησης του αισθητήρα για τη μεν θερμοκρασία είναι από 0^{O}C έως 50^{O}C με ακρίβεια ±2^{O}C, για τη δε σχετική υγρασία είναι από 20% έως 80% με ακρίβεια ±5%. Ο ρυθμός δειγματοληψίας του module είναι 1Hz, δηλαδή είναι δυνατή η λήψη νέων δεδομένων από τον αισθητήρα κάθε 1 δευτερόλεπτο.

Το module διαθέτει τρεις ακροδέκτες:

1) Vcc: Ο ακροδέκτης τροφοδοσίας του module. Είναι δυνατή η παροχή τάσης από 3,3V έως 5V. Όταν τροφοδοτείται με 5V, ο αισθητήρας μπορεί να τοποθετηθεί μέχρι και σε απόσταση 20m. Στην περίπτωση της ελάχιστης επιτρεπόμενης τροφοδοσίας (3,3V), η απόσταση δεν μπορεί να ξεπεράσει το 1m, διότι η πτώση τάσης στα καλώδια σύνδεσης θα οδηγήσει σε λάθη στις μετρήσεις.

2) Gnd: Ο ακροδέκτης γείωσης του module. Συνδέεται με τον αντίστοιχο ακροδέκτη γείωσης (Ground) του Arduino.

3) Out: Ο ακροδέκτης επικοινωνίας. Συνδέεται με κάποιον από τους ακροδέκτες ψηφιακής επικοινωνίας του Arduino. [12]



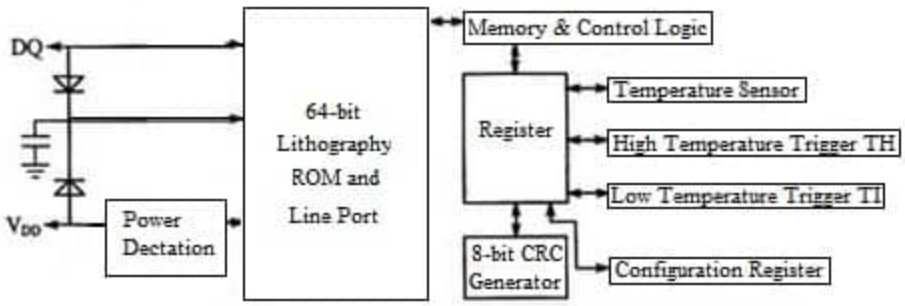
Εικόνα 2.6: Διάγραμμα ακροδεκτών Arduino
(https://lastminuteengineers.com/dht11-module-arduino-tutorial/)

2.3 Αισθητήρας DS18B20

Ο αισθητήρας DS18B20 χρησιμοποιείται για τη μέτρηση της θερμοκρασίας. Για την επικοινωνία με το Arduino χρησιμοποιεί το πρωτόκολλο 1-wire. Περιλαμβάνει δύο τύπους, ο ένας έχει τη μορφή και το μέγεθος ενός τυπικού transistor και ο άλλος αποτελείται από ένα αδιάβροχο μεταλλικό probe, το οποίο σε συνδυασμό με τα τρία μονωμένα καλώδια που διαθέτει, επιτρέπει τη μέτρηση της θερμοκρασίας σε εξωτερικό χώρο ακόμα και υπόγεια ή υποθαλάσσια. [14]

Εικόνα 2.7: Τύποι DS18B20
(https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html)

Ο αισθητήρας αποτελείται από τρία μέρη: μία μνήμη ROM μεγέθους 64 bits, ένα κύκλωμα υπολογισμού της θερμοκρασίας και ένα κύκλωμα μνήμης

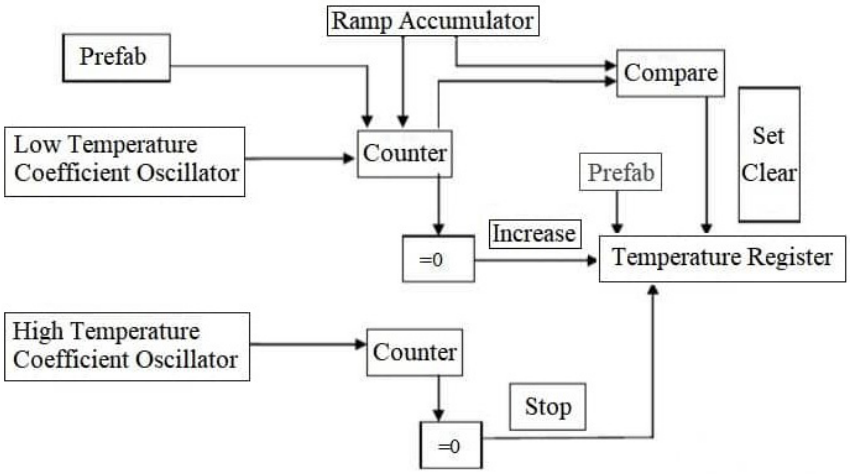


RAM.

Εικόνα 2.8: Block διάγραμμα DS18B20
(https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html)

Η μνήμη ROM περιέχει έναν σειριακό αριθμό μεγέθους 64 bits ο οποίος καταχωρείται από κατασκευής στον αισθητήρα. Ο αριθμός αυτός μπορεί να θεωρηθεί ότι αποτελεί τον κωδικό μοναδικής αναγνώρισης - διευθυνσιοδότησης του αισθητήρα. Με αυτόν τον τρόπο είναι δυνατή η χρήση παραπάνω του ενός αισθητήρα στο ίδιο κύκλωμα.

Το κύκλωμα υπολογισμού της θερμοκρασίας αποτελείται από δύο ταλαντωτές οι οποίοι επηρεάζονται διαφορετικά από τη μεταβολή της θερμοκρασίας. Ανάλογα με τη μεταβολή που υφίστανται στην ταλάντωση τους και βάσει τη διαφορά που δημιουργείται μεταξύ του χρονισμού της παλμοσειράς κάθε ταλαντωτή, υπολογίζεται η τιμή της θερμοκρασίας.



Εικόνα 2.9: Block διάγραμμα κυκλώματος υπολογισμού της θερμοκρασίας
(https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html)

Η μνήμη RAM χρησιμοποιείται στην επικοινωνία του αισθητήρα με το Arduino καθώς επίσης και στον καθορισμό της ανάλυσης (resolution) της μετρήσεων που παρέχονται. Συγκεκριμένα, ο αισθητήρας μπορεί να παρέχει τιμές που το μέγεθός τους να καταλαμβάνει 9, 10, 11 ή 12 bits. Η προεπιλεγμένη ανάλυση είναι 12 bits.

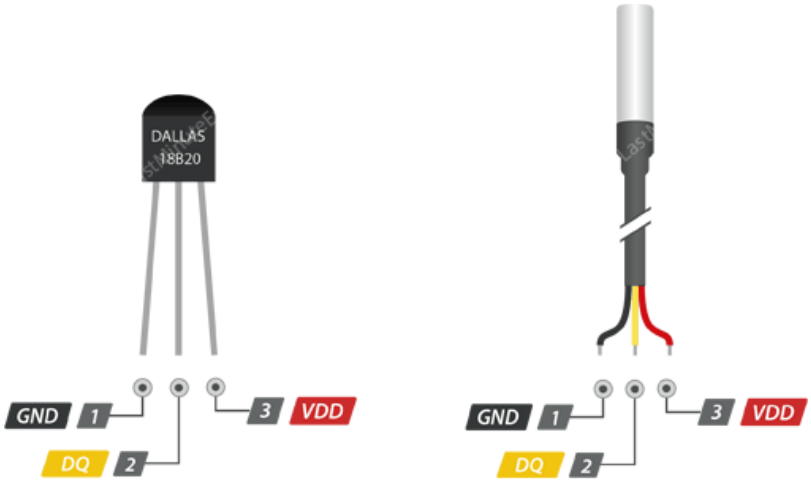
Τα όρια μέτρησης του αισθητήρα είναι από -55°C έως +125°C. Η ακρίβειά του αισθητήρα είναι ± 0,5°C στο πεδίο θερμοκρασιών μεταξύ -10°C έως +85°C και ± 2°C για τις υπόλοιπες θερμοκρασίες.

Ο αισθητήρας αποτελείται από τρεις ακροδέκτες:

1) V_{DD}: Ο ακροδέκτης τροφοδοσίας του module. Είναι δυνατή η παροχή τάσης από 3,3V έως 5V.

2) D_Q: Ο ακροδέκτης επικοινωνίας. Συνδέεται με κάποιον από τους ακροδέκτες ψηφιακής επικοινωνίας του Arduino.

3) Gnd: Ο ακροδέκτης γείωσης του module. Συνδέεται με τον αντίστοιχο ακροδέκτη γείωσης (Ground) του Arduino.

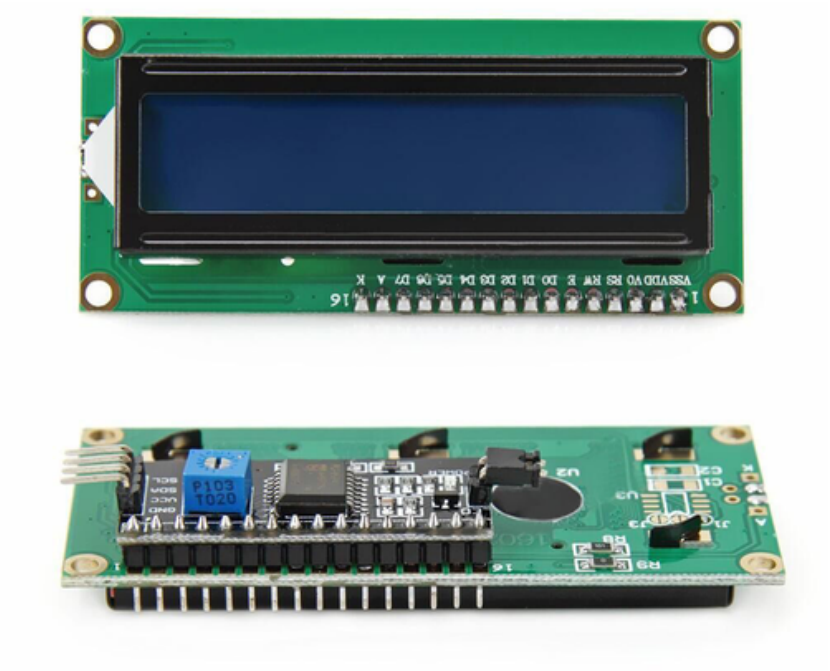


Εικόνα 2.10: Διάγραμμα ακροδεκτών Arduino
(https://lastminuteengineers.com/ds18b20-arduino-tutorial/)

Για τη σύνδεση του αισθητήρα με το Arduino, απαιτείται και η σύνδεση μίας αντίστασης 4,7ΚΩ η οποία εξυπηρετεί στη σταθεροποίηση της γραμμής δεδομένων.

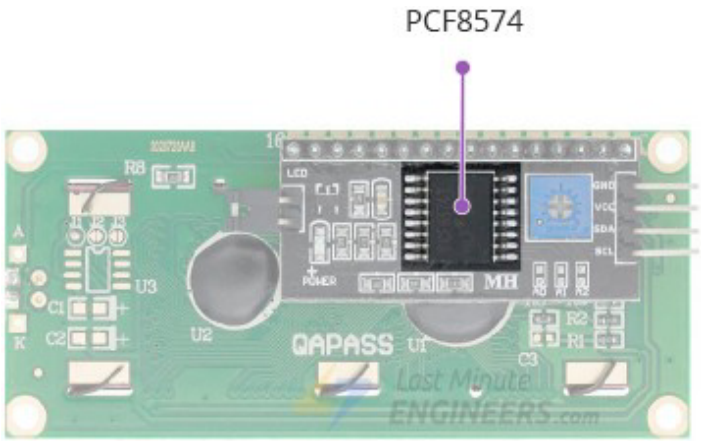
2.4 Οθόνη LCD 16x2 I2C

Για την επικοινωνία του χρήστη με τη συσκευή θα χρησιμοποιηθεί μία οθόνη LCD μεγέθους 16x2. Το συγκεκριμένο μέγεθος υποδηλώνει ότι η οθόνη μπορεί να εμφανίσει 2 σειρές χαρακτήρων, όπου κάθε σειρά μπορεί να διαθέτει μέχρι 16 ASCII χαρακτήρες. Επίσης, επειδή το να συνδεθεί απευθείας η οθόνη με το Arduino απαιτεί τη δέσμευση είτε 6 είτε 10 ψηφιακών ακροδεκτών της πλακέτας (αναλόγως του επιλεγμένου τρόπου λειτουργίας, δηλαδή 4-bit ή 8-bit mode), θα χρησιμοποιηθεί το module το οποίο εκτός από την οθόνη διαθέτει κύκλωμα επικοινωνίας με το Arduino με τη χρήση του I2C πρωτοκόλλου[15]. Αναλυτικότερα:



Εικόνα 2.11: Όψεις της οθόνης LCD 16x2 I2C
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

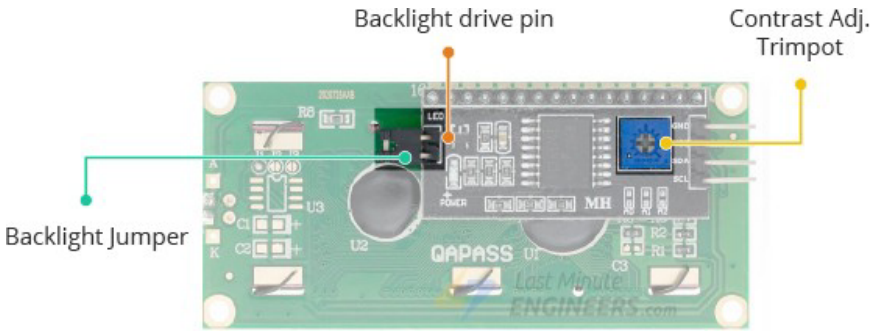
Το module διαθέτει ένα ολοκληρωμένο PCF8574 το οποίο λαμβάνει τα δεδομένα από το Arduino μέσω του πρωτοκόλλου I2C και τα μεταδίδει στην οθόνη με τη χρήση παράλληλης επικοινωνίας 8 bit.



Εικόνα 2.12: Θέση του PCF8574 στην οθόνη LCD
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

Στο ίδιο κύκλωμα υπάρχει ένα ποτενσιόμετρο για τη ρύθμιση της αντίθεσης της οθόνης. Επίσης, για την παροχή ισχύος στο backlight από τη κύρια τροφοδοσία της οθόνης, υπάρχει ένα jumper το οποίο μπορεί να αφαιρεθεί προκειμένου να τροφοδοτηθεί το backlight από ανεξάρτητη

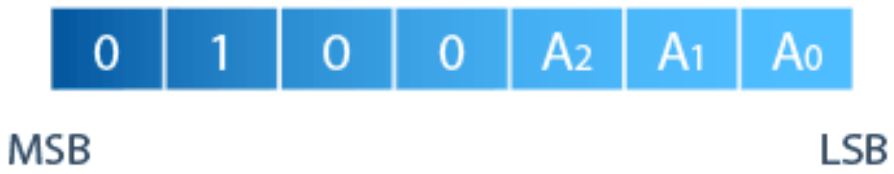
παροχή τάσης.



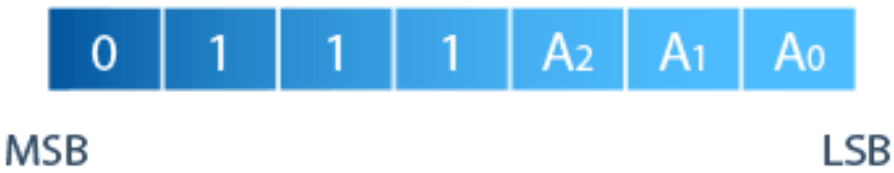
Εικόνα 2.12: Θέση εξαρτημάτων ρύθμισης φωτεινότητας
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

Τέλος, όπως προαναφέρθηκε το module αυτό επικοινωνεί με το Arduino με τη χρήση του πρωτοκόλλου I2C. Αυτό σημαίνει ότι το module διαθέτει μία διεύθυνση προκειμένου να επιτευχθεί η επικοινωνία με το συγκεκριμένο πρωτόκολλο. Οι διευθύνσεις αυτές είναι μήκους 7 bits και υπάρχει η δυνατότητα να καθοριστούν τα 3 λιγότερο σημαντικά bits μέσω hardware, έτσι ώστε αν στο ίδιο κύκλωμα χρησιμοποιούνται παραπάνω από ένα ίδια modules, να μπορεί να διαφοροποιηθεί η διεύθυνσή τους προκειμένου να είναι εφικτή η επικοινωνία με το Arduino.

Πιο συγκεκριμένα, αναλόγως της εταιρεία κατασκευής του ολοκληρωμένου PCF8574, η διεύθυνση θα είναι μία από τις δύο απεικονιζόμενες παρακάτω:



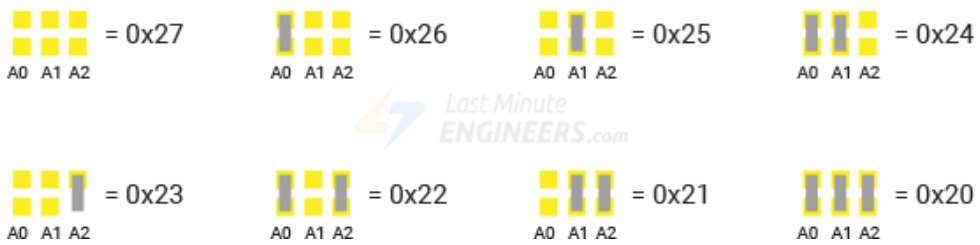
Εικόνα 2.13: Μορφή διεύθυνσης PCF8574 της Texas Instruments
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)



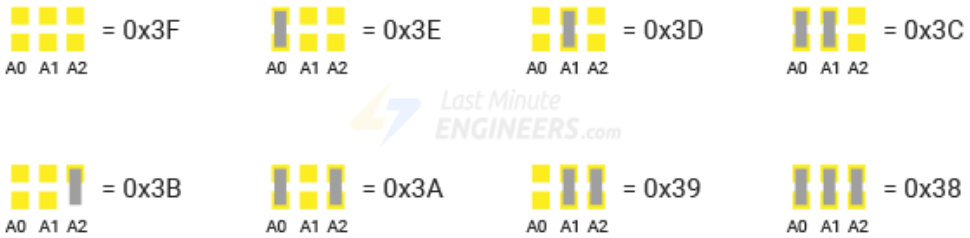
Εικόνα 2.14: Μορφή διεύθυνσης PCF8574 της NXP Semiconductors
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

Για τον καθορισμό των A0, A1 και A2 υπάρχουν στην πλακέτα τρία ζευγάρια επαφών τα οποία όταν δεν είναι βραχυκυκλωμένα θέτουν στο αντίστοιχο

bit την τιμή 1. Όταν βραχυκυκλωθούν με τη χρήση κάποιου jumper, τότε θέτουν την τιμή 0. Ανάλογα με την εταιρεία, οι πιθανές διευθύνσεις που μπορεί να λάβει το module φαίνονται στις παρακάτω φωτογραφίες:



Εικόνα 2.15: Εύρος διευθύνσεων PCF8574 της Texas Instruments
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)



Εικόνα 2.16: Εύρος διευθύνσεων PCF8574 της NXP Semiconductors
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

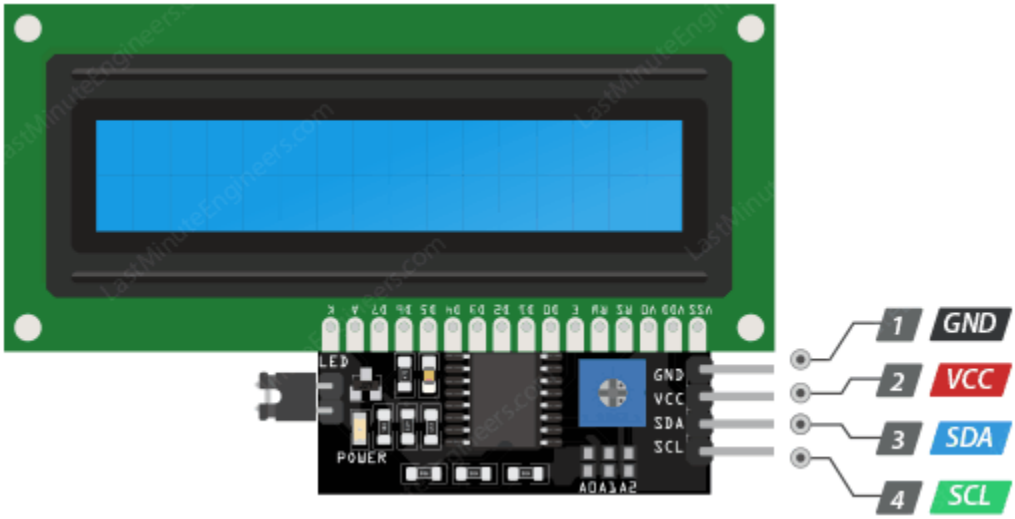
Το module διαθέτει τέσσερις ακροδέκτες:

4) Vcc: Ο ακροδέκτης τροφοδοσίας του module. Συνδέεται στην έξοδο των 5V του Arduino.

5) Gnd: Ο ακροδέκτης γείωσης του module. Συνδέεται με τον αντίστοιχο ακροδέκτη γείωσης (Ground) του Arduino.

6) SDA: Ο ακροδέκτης Serial Data του πρωτοκόλλου I2C. Στη γραμμή αυτή γίνεται η αμφίδρομη μετάδοση δεδομένων. Συνδέεται στον αντίστοιχο ακροδέκτη SDA του Arduino.

7) SCL: Ο ακροδέκτης Serial Clock του πρωτοκόλλου I2C. Μέσω αυτού μεταδίδονται τα σήματα χρονισμού που παρέχει ο master. Συνδέεται στον αντίστοιχο ακροδέκτη SCL του Arduino.

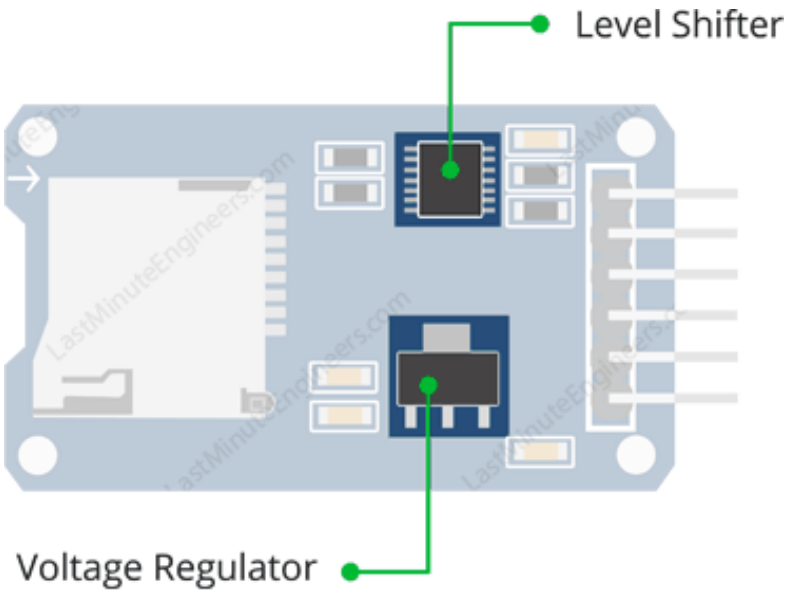


Εικόνα 2.17: Διάγραμμα ακροδεκτών LCD I2C
(https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/)

2.5 Module SD κάρτας

Ένας από τους πιο προσιτούς και εύκολα υλοποιήσιμους τρόπους για την αποθήκευση δεδομένων από αισθητήρες είναι η χρήση καρτών SD. Οι συγκεκριμένες κάρτες είναι ευρέως διαδεδομένες και χρησιμοποιούνται από μια πληθώρα, αν όχι το σύνολο, των διαφορετικών ειδών ηλεκτρονικών συσκευών που απαιτούν αποθήκευση δεδομένων.

Όσον αφορά τις κατασκευές με Arduino, μπορεί να χρησιμοποιηθεί στην κατασκευή το module που διαθέτει υποδοχή SD κάρτας. Η επικοινωνία του module αυτού με το Arduino επιτυγχάνεται με τη χρήση του πρωτοκόλλου SPI.Το συγκεκριμένο module διαθέτει δύο κύρια εξαρτήματα: έναν σταθεροποιητή τάσης (voltage regulator) και ένα ολοκληρωμένο μετατόπισης λογικού επιπέδου (level shifter).[16]



Εικόνα 2.18: Κύρια εξαρτήματα της συσκευής επικοινωνίας κάρτας SD
(https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/)

Το voltage regulator ρυθμίζει την τάση στα 3,3V, διότι η τάση λειτουργίας των καρτών SD είναι 3,3V. Γι' αυτό τον λόγο δε θα μπορούσαμε να συνδέσουμε το module χωρίς τη χρήση του σταθεροποιητή σε κυκλώματα που παρέχουν μεγαλύτερη τάση στα modules. Αν η τάση ξεπεράσει τα 3,6V, τότε η κάρτα SD καταστρέφεται μόνιμα. Έτσι με τη χρήση του σταθεροποιητή τάσης διασφαλίζεται ότι η τάση λειτουργίας που παρέχεται στην κάρτα SD ρυθμίζεται και σταθεροποιείται στα 3,3V. Βέβαια, η μέγιστη τάση λειτουργίας στην οποία μπορεί να συνδεθεί το module είναι 5V.

Το level shifter είναι ένα κύκλωμα το οποίο μετατρέπει την τάση του υψηλού λογικού σήματος (λογικό «1») στο επίπεδο των 3.3V. Γενικότερα, διάφορα modules και μικροελεγκτές μπορεί να χρησιμοποιούν διαφορετικές τάσεις για την αναπαράσταση αυτού του ψηφιακού σήματος, όπως για παράδειγμα το Arduino χρησιμοποιεί 5V. Με τη χρήση του level shifter, το module μετατοπίζει την τάση αυτή στα 3,3V ούτως ώστε να μπορεί να μπορεί να επικοινωνήσει με το Arduino.

Το module διαθέτει έξι ακροδέκτες:

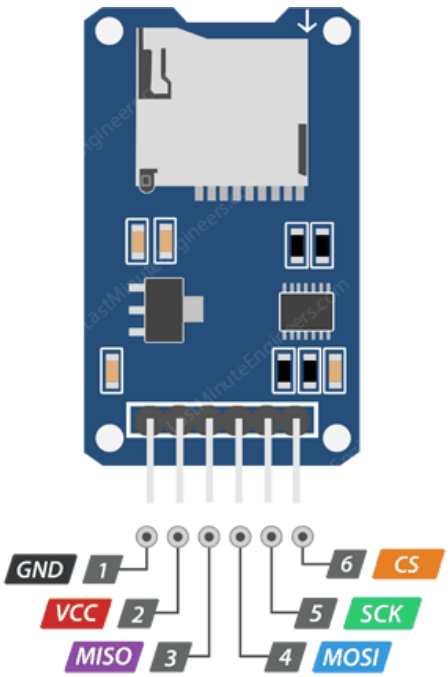
1) Vcc: Ο ακροδέκτης τροφοδοσίας του module. Συνδέεται στην έξοδο των 5V του Arduino.

2) Gnd: Ο ακροδέκτης γείωσης του module. Συνδέεται με τον αντίστοιχο ακροδέκτη γείωσης (Ground) του Arduino.

3) MISO (Master In Slave Out): Ο ακροδέκτης εξόδου του πρωτοκόλλου

SPI. Συνδέεται στον αντίστοιχο ακροδέκτη του Aruino (είναι διαφορετικός σε κάποια μοντέλα).

4) MOSI (Master Out Slave In): Ο ακροδέκτης εισόδου του πρωτοκόλλου SPI. Συνδέεται στον αντίστοιχο ακροδέκτη του Aruino (είναι διαφορετικός σε κάποια μοντέλα).

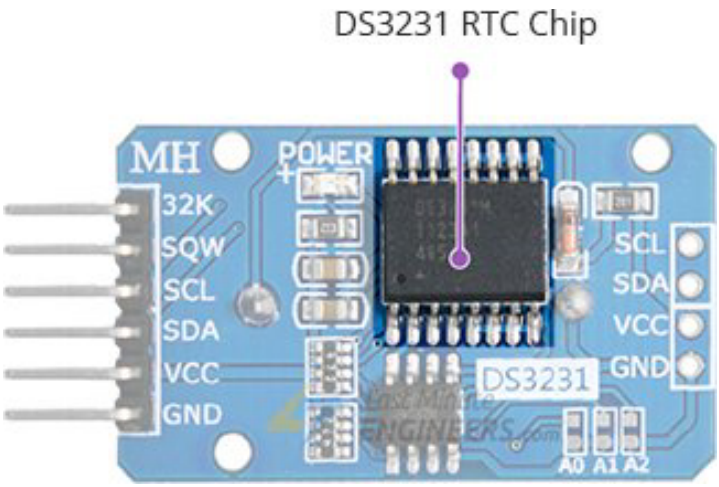


Εικόνα 2.19: Διάγραμμα ακροδεκτών SD module
(https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/)

2.6 Ρολόι πραγματικού χρόνου DS3231

Επειδή η πλακέτα του Arduino δε διαθέτει κύκλωμα μέτρησης - διατήρησης του πραγματικού χρόνου (όπως υπάρχει, για παράδειγμα σε έναν ηλεκτρονικό υπολογιστή) θα χρειαστεί να γίνει η χρήση ενός module που θα επιτελεί τη λειτουργία αυτή. Στη συγκεκριμένη κατασκευή επιλέχθηκε το DS3231[17].

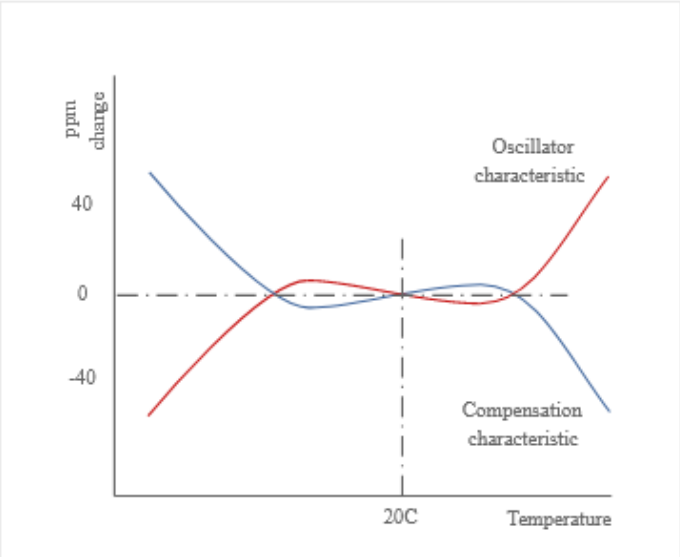
Το συγκεκριμένο module διαθέτει το ολοκληρωμένο DS3231 που επιτελεί τη λειτουργία του ρολογιού πραγματικού χρόνου και υποστηρίζει την επικοινωνία μέσω του πρωτοκόλλου I2C. Το ολοκληρωμένο αυτό διατηρεί πληροφορίες για δευτερόλεπτα, λεπτά, ώρες, ημέρα της εβδομάδας, ημέρα, μήνα και ώρα και υπολογίζει ορθά τις ημερομηνίες των δίσεκτων ετών. Επίσης, μπορεί να παρέχει την ώρα είτε σε μορφή 12ώρου είτε 24ώρου. Μία άλλη δυνατότητα (η οποία δεν είναι αναγκαία για την κατασκευή μας, οπότε δε θα χρησιμοποιηθεί) είναι η δυνατότητα να παράγει τετραγωνικούς παλμούς συχνότητας 1Hz, 4kHz, 8kHz ή 32kHz.



Εικόνα 2.20: Θέση RTC Chip στο module
(https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/)

Τα περισσότερα modules για τη διατήρηση του πραγματικού χρόνου χρησιμοποιούν έναν εξωτερικό κρύσταλλο 32kHz. Το πρόβλημα αυτών των κρυστάλλων είναι ότι η εξωτερική θερμοκρασία επηρεάζει τη συχνότητα ταλάντωσής τους. Η διαφορά που προκύπτει στη συχνότητα είναι αμελητέα, ωστόσο σταδιακά αυτή η διαφορά λειτουργεί αθροιστικά. Για παράδειγμα, ένα διαφορετικό module, το DS1307, λόγω αυτού του φαινομένου παρουσιάζει διαφορά 5 λεπτών από τον πραγματικό χρόνο ανά μήνα.

Στην περίπτωση του DS3231 αυτό δε συμβαίνει, διότι διαθέτει έναν κρύσταλλο TCXO (Temperature Compensated Xtal Oscillator) ο οποίος κάνει αντιστάθμιση της διαφοράς που προκύπτει στη συχνότητα ταλάντωσης λόγω της θερμοκρασίας και έτσι παραμένει εξαιρετικά ακριβής, παρουσιάζοντας διαφορά περίπου 2 λεπτών ανά έτος.



Εικόνα 2.21: Διάγραμμα αντιστάθμισης ταλάντωσης – θερμοκρασίας
(https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/)

Επίσης, το module διαθέτει ενσωματωμένη υποδοχή για μπαταρία 3V. Το κύκλωμα αντιλαμβάνεται την έλλειψη παροχής ισχύος από τον ακροδέκτη τροφοδοσίας και αυτόματα λαμβάνει ισχύ από την μπαταρία. Με αυτόν τον τρόπο, η παροχή ισχύος και κατ' επέκταση η διατήρηση του χρόνου γίνεται αδιάλειπτα.

Εικόνα 2.22: Θέση μπαταρίας στο module
(https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/)

Το module διαθέτει τους εξής ακροδέκτες:

1) Vcc: Ο ακροδέκτης τροφοδοσίας του module. Είναι δυνατή η παροχή τάσης από 3,3V έως 5V.

2) Gnd: Ο ακροδέκτης γείωσης του module. Συνδέεται με τον

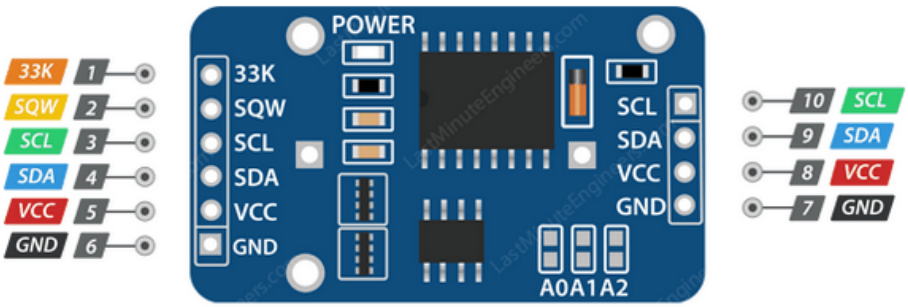
αντίστοιχο ακροδέκτη γείωσης (Ground) του Arduino.

3) SDA: Ο ακροδέκτης Serial Data του πρωτοκόλλου I2C. Στη γραμμή αυτή γίνεται η αμφίδρομη μετάδοση δεδομένων. Συνδέεται στον αντίστοιχο ακροδέκτη SDA του Arduino.

4) SCL: Ο ακροδέκτης Serial Clock του πρωτοκόλλου I2C. Μέσω αυτού μεταδίδονται τα σήματα χρονισμού που παρέχει ο master. Συνδέεται στον αντίστοιχο ακροδέκτη SCL του Arduino.

5) 32K: Ο ακροδέκτης αυτός παρέχει το αντισταθμισμένο σήμα που παράγει ο TCXO κρύσταλλος.

6) SQW: Ο ακροδέκτης αυτός παρέχει τους τετραγωνικούς παλμούς που παράγει το module όπως αναφέρθηκε παραπάνω.



Εικόνα 2.23: Διάγραμμα ακροδεκτών module
(https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/)

2.7 Υπόλοιπα εξαρτήματα συνδεσμολογίας

Για την ολοκλήρωση του κυκλώματος, απαραίτητα στοιχεία ήταν ένας βομβητής (buzzer) για την ηχητική προειδοποίηση του συναγερμού και στιγμιαίοι διακόπτες για την πλοήγηση στο μενού του προγράμματος και την επιλογή των ρυθμίσεων.

Buzzer: Το buzzer χρησιμοποιείται για να παράγει ήχο. Υπάρχουν δύο τύποι: τα ενεργητικά και τα παθητικά. Τα ενεργητικά παράγουν ήχο απλά παρέχοντας τους τροφοδοσία και έχουν δύο ακροδέκτες. Στα παθητικά δεν αρκεί να δοθεί μόνο τροφοδοσία, πρέπει να λάβουν και σήμα από κάποιον μικροελεγκτή (ή το Arduino) για να λειτουργήσουν συνεπώς διαθέτουν τρεις ακροδέκτες.

Εικόνα 2.24: Buzzer module
(https://electropeak.com/learn/interfacing-active-buzzer-with-arduino/)

Στιγμιαίος διακόπτης: Οι στιγμιαίοι διακόπτες είναι διακόπτες που περιέχουν μηχανισμό επαναφοράς και ενεργοποιούνται μόνο κατά τη διάρκεια που παραμένουν πατημένοι. Οι επαφές τους είναι ανά δύο συνδεδεμένες οπότε για τη σύνδεση στο κύκλωμα απαιτείται η σύνδεση μιας επαφής από κάθε ζεύγος επαφών.

Εικόνα 2.25: Καταστάσεις στιγμιαίου διακόπτη
(https://arduinogetstarted.com/)

2.8 Πρώτοκολλα επικοινωνίας

Καθένα από τα προαναφερθέντα εξαρτήματα των προηγούμενων παραγράφων επικοινωνούν με την πλακέτα Arduino με κάποια συγκεκριμένη μορφή επικοινωνίας (στην επιστήμη των υπολογιστών οι μορφές επικοινωνίας είναι γνωστές και ως πρωτόκολλα). Στην παράγραφο αυτή γίνεται αναφορά στα πρωτόκολλα που χρησιμοποιούνται για την επικοινωνία των παραπάνω εξαρτημάτων.

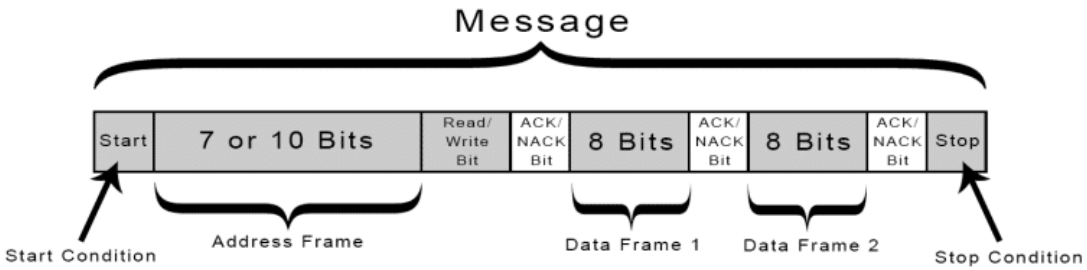
2.8.1 Inter-Integrated Circuit (I2C)

Το πρωτόκολλο Inter-Integrated Circuit (ή I2C ή IIC) χρησιμοποιείται για τη σειριακή επικοινωνία σε αρχιτεκτονικές master – slave. Συγκεκριμένα, μπορεί να χρησιμοποιηθεί είτε σε περιπτώσεις ενός master και ενός ή πολλαπλών slaves, είτε πολλαπλών masters και ενός slave. Αυτό είναι πολύ χρήσιμο όταν, για παράδειγμα, απαιτείται η λήψη δεδομένων από modules αισθητήρων που χρησιμοποιούν το πρωτόκολλο αυτό και η εμφάνισή τους σε κάποια οθόνη χρησιμοποιώντας ένα Arduino στον ρόλο του master και τα υπόλοιπα έχοντας ρόλο slave.[18]

Το I2C είναι πρωτόκολλο σειριακής επικοινωνίας, το οποίο χρησιμοποιεί δύο γραμμές επικοινωνίας:

- Η γραμμή SDA (Serial Data) χρησιμοποιείται για τη μετάδοση δεδομένων.
- Η γραμμή SCL (Serial Clock) χρησιμοποιείται για τη μετάδοση του σήματος χρονισμού.

Το πρωτόκολλο αυτό είναι σύγχρονο, δηλαδή η μετάδοση των bits συγχρονίζεται με την έξοδο του ρολογιού, την οποία μοιράζονται τόσο ο master όσο και ο slave, τον έλεγχο όμως του ρολογιού έχει μόνο ο master. Τα δεδομένα μεταδίδονται σε ομάδες bits που ονομάζονται μηνύματα (messages). Τα μηνύματα χωρίζονται επιπλέον σε πλαίσια δεδομένων (frames).[19]



Εικόνα 2.26: Μορφή μηνύματος I2C
(https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/)

Κάθε μήνυμα έχει ένα πλαίσιο διεύθυνσης που περιέχει τη δυαδική διεύθυνση του slave και ένα ή περισσότερα πλαίσια δεδομένων που περιέχουν τα δεδομένα προς μετάδοση. Επίσης, το μήνυμα περιέχει τα σήματα έναρξης (Start Condition) και τερματισμού (Stop Condition) του μηνύματος, Read/Write bits και ACK/NACK bits μεταξύ κάθε πλαισίου δεδομένων, τα οποία χρησιμοποιούνται ως εξής:

Start Condition: Για την έναρξη μετάδοσης μηνύματος, ο master θέτει τη γραμμή SDA από το υψηλό επίπεδο τάσης στο χαμηλό, πριν θέσει τη γραμμή SCL από το υψηλό επίπεδο τάσης στο χαμηλό.

Stop Condition: Για τον τερματισμό μετάδοσης μηνύματος, ο master θέτει τη γραμμή SCL από το χαμηλό επίπεδο τάσης στο υψηλό, πριν θέσει τη γραμμή SDA από το χαμηλό επίπεδο τάσης στο υψηλό.

Πλαίσιο διεύθυνσης(Address Frame): Αποτελείται από μία ακολουθία 7 έως 10 bits τα οποία αντιπροσωπεύουν τη μοναδική δυαδική διεύθυνση ενός slave, με τον οποίο ο master θέλει να ανταλλάξει δεδομένα.

Read/Write Bit: Προκειμένου ο master να στείλει δεδομένα στον slave, θέτει

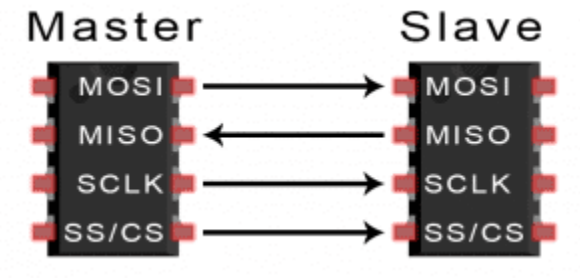
το bit αυτό σε χαμηλό επίπεδο τάσης, ενώ στην περίπτωση που ζητάει δεδομένα, το θέτει σε υψηλό επίπεδο τάσης.

ACK/NACK Bit: Μετά τη μετάδοση ενός πλαισίου ακολουθεί η μετάδοση από τον παραλήπτη ενός bit, το οποίο υποδηλώνει αν το πλαίσιο ελήφθη σωστά. Έτσι, αν ο παραλήπτης έλαβε σωστά το πλαίσιο δεδομένων, αποστέλλει ένα σήμα ACK.

Στο κύκλωμα του θερμομέτρου το πρωτόκολλο αυτό χρησιμοποιείται στην επικοινωνία μεταξύ της πλακέτας Arduino, της οθόνης LCD I2C και του ρολογιού πραγματικού χρόνου DS3231.

2.8.2 Σειριακός Δίαυλος Επικοινωνίας SPI (Serial Peripheral Interface)

Το πρωτόκολλο Serial Peripheral Interface (ή SPI) χρησιμοποιείται για τη σειριακή επικοινωνία σε αρχιτεκτονικές master – slave. Συγκεκριμένα, μπορεί να χρησιμοποιηθεί σε περιπτώσεις όπου υπάρχει ένας master και ένας ή περισσότεροι slaves.[20]



Εικόνα 2.27: Διασύνδεση master – slave
(https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/)

Η μεταφορά δεδομένων με το πρωτόκολλο αυτό γίνεται με συνεχή ροή, δηλαδή δεν αποτελείται από διακριτά πακέτα συγκεκριμένου μεγέθους. Το SPI είναι πρωτόκολλο σειριακής επικοινωνίας, το οποίο χρησιμοποιεί τέσσερις γραμμές επικοινωνίας:

Η γραμμή MOSI (Master Output/Slave Input) χρησιμοποιείται για τη μετάδοση δεδομένων από τον master στον slave.

Η γραμμή MISO (Master Input/Slave Output) χρησιμοποιείται για τη μετάδοση δεδομένων από τον slave στον master.

Η γραμμή SCLK (Serial Clock) χρησιμοποιείται για τη μετάδοση του σήματος χρονισμού.

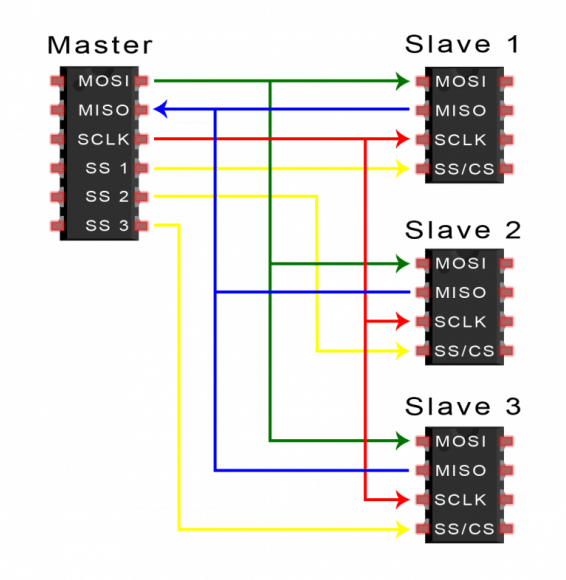
Η γραμμή SS/CS (Slave Select/Chip Select) χρησιμοποιείται για την επιλογή του slave με τον οποίο επιθυμεί να επικοινωνήσει ο master.

Το σήμα χρονισμού χρησιμοποιείται για τον συγχρονισμό της μετάδοσης των bits είτε από τον master στον slave είτε αντίστροφα. Σε κάθε κύκλο του ρολογιού μεταφέρεται ένα bit δεδομένων, συνεπώς η ταχύτητα μεταφοράς δεδομένων καθορίζεται από τη συχνότητα του σήματος χρονισμού. Το σήμα χρονισμού καθορίζεται και παράγεται από τον master.[19]

Η επιλογή του slave με τον οποίο θα επικοινωνήσει ο master επιτυγχάνεται με το σήμα SS/CS. Συγκεκριμένα, για να γίνει η επιλογή ο master θέτει την αντίστοιχη γραμμή σε κατάσταση χαμηλής τάσης. Κατά τη χρονική διάρκεια όπου δεν υπάρχει ανάγκη επικοινωνίας, η γραμμή αυτή βρίσκεται

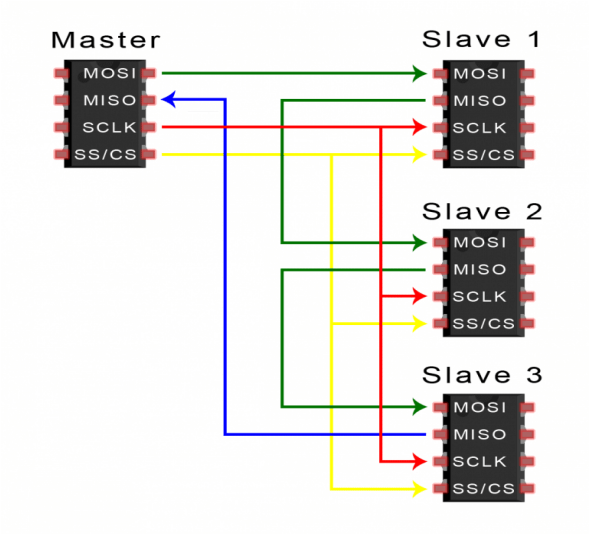
σε κατάσταση υψηλής τάσης. Όταν έχουμε έναν master και πολλαπλούς slaves, προκειμένου να μπορεί να γίνει εφικτή η επικοινωνία με όλους ταυτόχρονα, διακρίνεται η εξής περιπτωσιολογία:

1) Ο master διαθέτει παραπάνω από μία γραμμές τύπου SS/CS: Ο master διαθέτει σε κάθε slave μία διαφορετική γραμμή SS/CS και οι υπόλοιπες είναι συνδεδεμένες παράλληλα στους slaves.



Εικόνα 2.28: Σύνδεση με πολλές γραμμές SS/CS
(https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/)

2) Ο master διαθέτει μόνο μία γραμμή τύπου SS/CS: Ο master διαθέτει στους slaves παράλληλα μόνο τη γραμμή χρονισμού. Η γραμμή MOSI από τον master συνδέεται στον πρώτο slave, η γραμμή MISO του πρώτου slave συνδέεται στον ακροδέκτη MOSI του δεύτερου slave, η γραμμή MISO του δεύτερου slave συνδέεται στον ακροδέκτη MOSI του τρίτου slave κοκ. Τέλος, η γραμμή MISO του τελευταίου slave, συνδέεται στη γραμμή MISO του master.



Εικόνα 2.29: Σύνδεση με μία γραμμή SS/CS
(https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/)

Για τη μετάδοση των δεδομένων χρησιμοποιούνται οι γραμμές MISO και MOSI. Ο master αποστέλλει και ο slave λαμβάνει δεδομένα στη γραμμή MOSI. Συνήθως τα δεδομένα σε αυτήν τη γραμμή αποστέλλονται από το πιο σημαντικό στο λιγότερο σημαντικό bit. Στην αντίθετη περίπτωση, όπου ο slave αποστέλλει και ο master λαμβάνει δεδομένα, χρησιμοποιείται η γραμμή MISO. Τα δεδομένα σε αυτήν τη γραμμή αποστέλλονται συνήθως από το λιγότερο σημαντικό στο πιο σημαντικό bit.

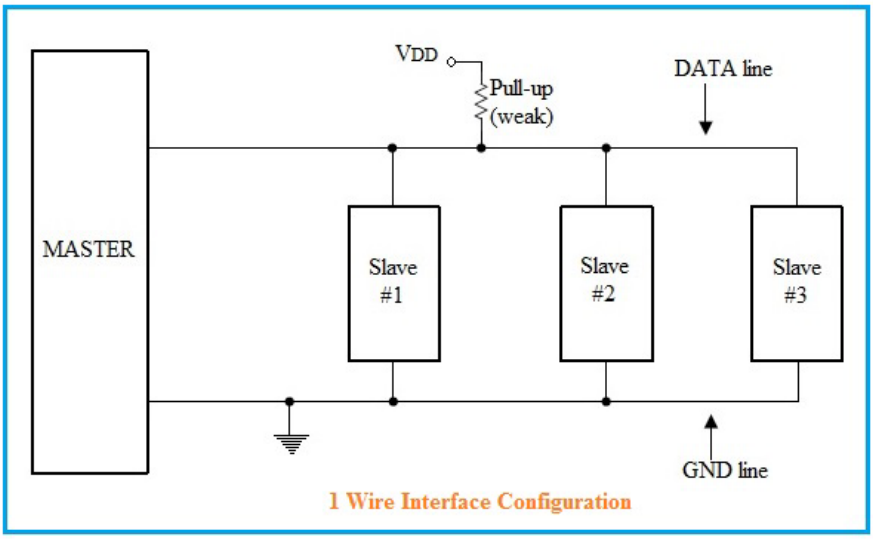
Στο κύκλωμα του θερμομέτρου χρησιμοποιείται το πρωτόκολλο SPI στην επικοινωνία μεταξύ της πλακέτας Arduino και του module της SD κάρτας.

2.8.3 Πρωτόκολλο Σειριακής Επικοινωνίας 1-Wire

Το πρωτόκολλο Serial 1-wire χρησιμοποιείται για τη σειριακή επικοινωνία σε αρχιτεκτονικές master – slave[21]. Συγκεκριμένα, μπορεί να χρησιμοποιηθεί σε περιπτώσεις όπου υπάρχει ένας master και ένας ή περισσότεροι slaves. Το είδος αυτό της επικοινωνίας χαρακτηρίζεται από χαμηλές απαιτήσεις σε ισχύ και χαμηλές ταχύτητες μετάδοσης. Χρησιμοποιείται κυρίως στην επικοινωνία αισθητήρων με μικροελεγκτές.

Το πρωτόκολλο αυτό χρησιμοποιεί δύο γραμμές επικοινωνίας, δηλαδή τη γραμμή data του αντίστοιχου αισθητήρα και το ground. Δεν απαιτείται κάποια γραμμή σήματος χρονισμού, όπως με τα πρωτόκολλα I2C και SPI,

διότι οι slaves κάνουν χρήση του δικού τους εσωτερικού ρολογιού το οποίο συγχρονίζεται με αυτό του master έπειτα από σήμα που θα λάβουν από αυτόν. Επισημαίνεται ότι στη γραμμή data θα πρέπει να συνδεθεί μία αντίσταση η οποία θα κρατάει τη γραμμή σε επίπεδο υψηλής τάσης όταν η γραμμή δε χρησιμοποιείται.



Εικόνα 2.30: Σύνδεση ενός master με πολλούς slaves
(https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html)

Εκ κατασκευής, οι συσκευές που μπορούν να λάβουν τον ρόλο του slave στην επικοινωνία αυτή έχουν τη δυνατότητα να λειτουργήσουν με δύο τρόπους:

Parasitic mode: Προκειμένου να λειτουργήσουν οι συσκευές τροφοδοτούνται χρησιμοποιώντας τη γραμμή data. Συγκεκριμένα, όταν η γραμμή δε χρησιμοποιείται (και συνεπώς βρίσκεται σε επίπεδο υψηλής τάσης) τότε ένας πυκνωτής χωρητικότητας 800pF φορτίζεται και παρέχει ισχύ στη συσκευή.

External mode: Προκειμένου να λειτουργήσουν οι συσκευές τροφοδοτούνται με εξωτερική ισχύ και όχι μέσω της γραμμής data.

Επίσης, στις συσκευές αυτές υπάρχει ένας μοναδικός 64-bit κωδικός ο οποίος αποτελεί το αναγνωριστικό της συσκευής. Χρησιμοποιώντας, αυτόν τον κωδικό ο master μπορεί να δώσει εντολή σε συγκεκριμένη συσκευή, όταν είναι συνδεδεμένες παραπάνω από μία συσκευές[22].

Στο κύκλωμα του θερμομέτρου το πρωτόκολλο αυτό χρησιμοποιείται στην επικοινωνία μεταξύ της πλακέτας Arduino και του αισθητήρα DS18B20.

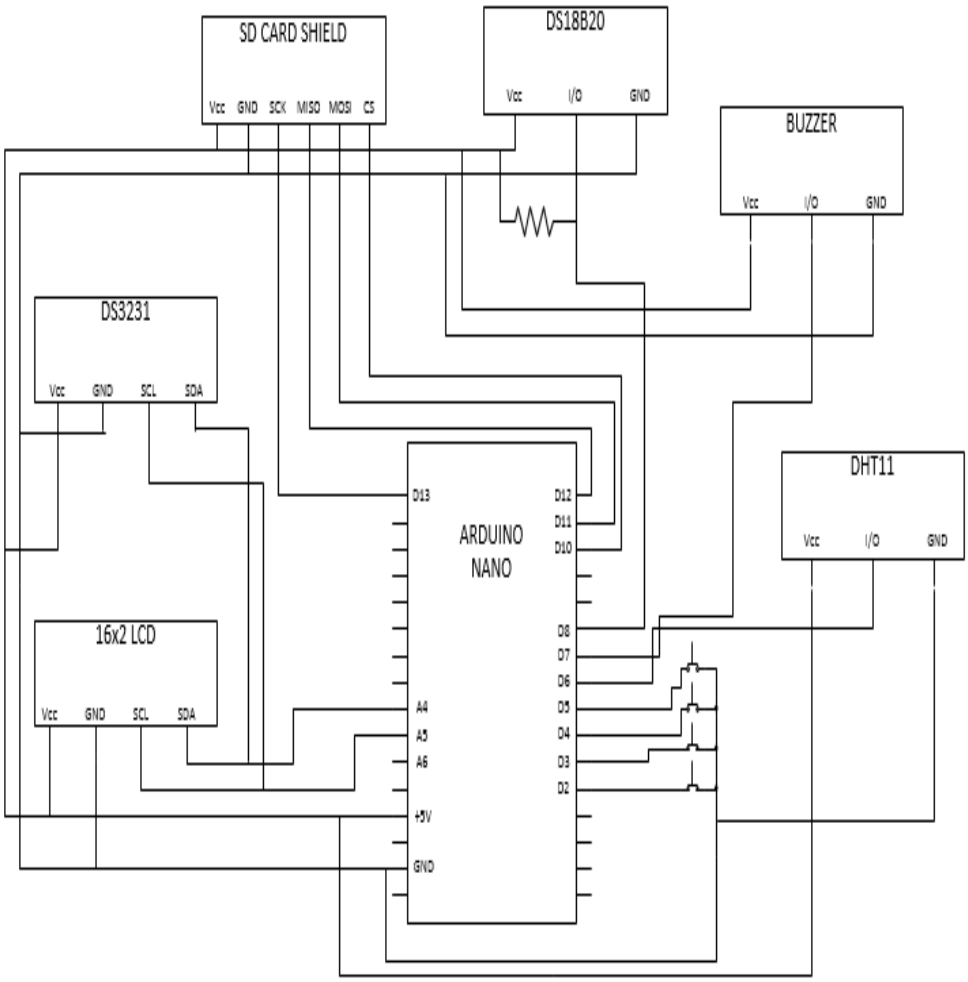
ΚΕΦΑΛΑΙΟ 3 - ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΑΙ ΚΑΤΑΣΚΕΥΗ

Στο κεφάλαιο αυτό αναλύεται η μεθοδολογία που εφαρμόστηκε για τη δημιουργίας της συσκευής. Αρχικά, γίνεται μια αναφορά στη σχεδίαση της συνδεσμολογίας του κυκλώματος. Έπειτα, παρουσιάζεται η πορεία συναρμολόγησης του τελικού κυκλώματος.

3.1 Σχεδίαση συνδεσμολογίας κυκλώματος

Προτού δημιουργηθεί η συνδεσμολογία του κυκλώματος σε φυσικό επίπεδο, ήταν απαραίτητο να δημιουργηθεί ένα σχέδιο της συνδεσμολογίας, ούτως ώστε αφενός μεν να υπάρχει ένα πλάνο καθοδήγησης για τη δημιουργία του αφετέρου δε να είναι πιο εύκολη η απομόνωση κάποιου σφάλματος που τυχόν θα προκύψει κατά τη διαδικασία της σύνδεσης των εξαρτημάτων. Ειδικότερα, στην περίπτωση των κυκλωμάτων που περιέχουν κάποια πλακέτα Arduino, αυτό αποκτά ακόμα μεγαλύτερη σημασία. Ο λόγος είναι ότι οι διάφοροι ακροδέκτες επικοινωνίας του Arduino αναλαμβάνουν τον ρόλο είτε της εισόδου (Input) είτε της εξόδου (Output) μέσα από τον προγραμματισμό της πλακέτας. Στην περίπτωση που ένας τέτοιος ακροδέκτης συνδεθεί από λάθος σε κάποιο διαφορετικό εξάρτημα από αυτόν που έχουμε σχεδιάσει και κατ' επέκταση προγραμματίσει, αυτό μπορεί να προκαλέσει σφάλμα που η έκτασή του μπορεί να κυμανθεί από μία ανεπιθύμητη συμπεριφορά του κυκλώματος μέχρι βλάβη σε κάποιο από τα εξαρτήματα ακόμα και στην πλακέτα του Arduino.

Για τη σχεδίαση του δημιουργήθηκε ένα block διάγραμμα όπου απεικονίζεται το σύνολο του κυκλώματος, το οποίο έχει διαιρεθεί στα διάφορα modules από τα οποία αποτελείται. Σκοπός του διαγράμματος αυτού είναι η ανάδειξη των συνδέσεων μεταξύ των modules. Το block διάγραμμα φαίνεται στην παρακάτω εικόνα.



Εικόνα 3.1 Block διάγραμμα συσκευής

Τα modules που θα χρησιμοποιηθούν στο κύκλωμα αναλύθηκαν λεπτομερώς στο Κεφάλαιο 2. Για την πληρότητα της διαδικασίας σχεδίασης παρακάτω παρατίθενται πίνακας με το ποιες συνδέσεις έγιναν μεταξύ των ακροδεκτών των modules και του Arduino Nano.

ΠΙΝΑΚΑΣ 1. ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΑΚΡΟΔΕΚΤΩΝ

	ΑΚΡΟΔΕΚΤΕΣ MODULE	ΑΚΡΟΔΕΚΤΕΣ ARDUINO
DHT 11	VCC	5V
	OUT	D6
	GND	GND
LCD I2C	VCC	5V
	GND	GND
	SDA	A4
	SCL	A5
BUZZER	VCC	5V
	I/O	D7
	GND	GND
DS3231 I2C	VCC	5V
	GND	GND
	SDA	A4
	SCL	A5
	VCC	5V
SD CARD	GND	GND
	MISO	D12
	MOSI	D11
	SCK	D13
	CS	D10
	VCC	5V
DS18B20	I/O	D8
	GND	GND
BUTTON 1	A	D2
	B	GND
BUTTON 2	A	D3
	B	GND
BUTTON 3	A	D4
	B	GND
BUTTON 4	A	D5
	B	GND
RESISTOR 4,7K	R.1	5V
	R.2	D8

Παρατηρήσεις:

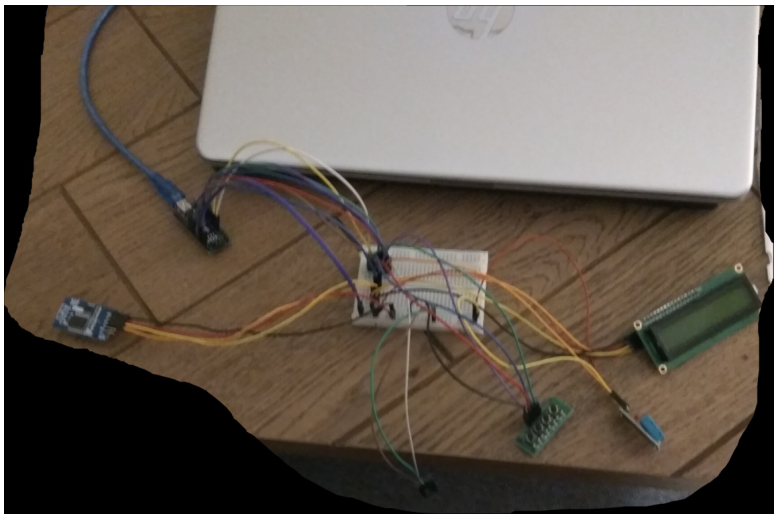
- Παρά το γεγονός ότι κάποια modules είχαν τη δυνατότητα να λειτουργήσουν και με τάση τροφοδοσίας 3.3V, επιλέχθηκε να συνδεθούν όλα στην τάση τροφοδοσίας 5V για την όσο το δυνατόν μεγαλύτερη απλοποίηση των γραμμών σύνδεσης του κυκλώματος.
- Τα τέσσερα κουμπιά (BUTTON 1 – 4) είναι τοποθετημένα σε δική

τους πλακέτα και είναι συνδεδεμένα στο κύκλωμα με συνδεσμολογία κοινής καθόδου. Σε αυτήν τη συνδεσμολογία (η οποία συνεισφέρει στην απλοποίηση του κυκλώματος) ο ακροδέκτης Β κάθε κουμπιού είναι κοινά συνδεδεμένος στο GROUND. Οι ακροδέκτες Α των κουμπιών συνδέονται απευθείας με ακροδέκτες ψηφιακής επικοινωνίας του Arduino οι οποίοι έχουν οριστεί ως είσοδοι συνδεδεμένοι με εσωτερική pullup αντίσταση του Arduino (input pull-up mode).

- Όπως έχει αναφερθεί και στο Κεφάλαιο 2 στα χαρακτηριστικά του πρωτοκόλλου επικοινωνίας I2C, το Arduino μπορεί να λειτουργήσει ως master έχοντας συνδεδεμένες παράλληλα διάφορους slaves. Έτσι, στη συνδεσμολογία μας συνδέονται παράλληλα οι γραμμές SDA και SCL αντίστοιχα των LCD I2C module και του DS3231 I2C module χωρίς να δημιουργείται conflict.

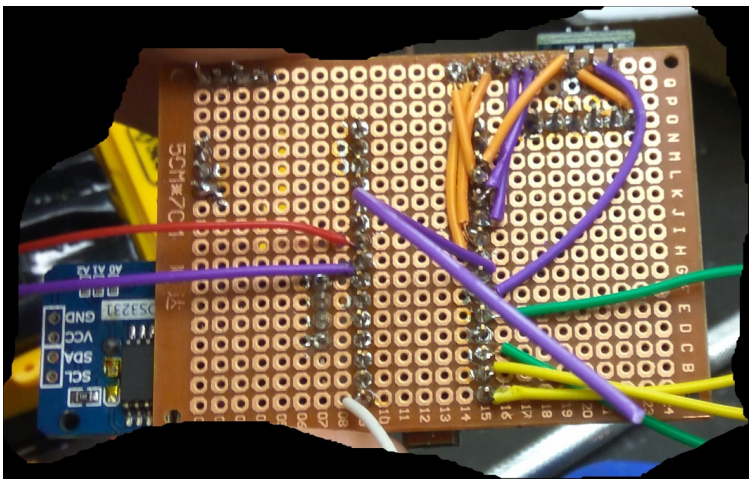
3.2 Συναρμολόγηση κυκλώματος

Η συναρμολόγηση του τελικού κυκλώματος έλαβε χώρα σε δύο στάδια. Στο πρώτο στάδιο τα διάφορα εξαρτήματα συνδέθηκαν με το Arduino Nano με τη χρήση breadboard. Το breadboard είναι μία διάτρητη πλακέτα που διαθέτει προεγκατεστημένες μεταλλικές επαφές και έτσι επιτρέπει τη σύνδεση των εξαρτημάτων με καλώδια που διαθέτουν το αντίστοιχο βύσμα, χωρίς την ανάγκη να γίνουν κολλήσεις μεταξύ τους. Ο τρόπος αυτός συναρμολόγησης είναι πολύ κοινός όταν θέλουμε να επιβεβαιώσουμε την ορθή λειτουργία μίας συνδεσμολογίας καθώς επίσης και να οριστικοποιήσουμε τις οριστικές συνδέσεις, προτού προβούμε στην κόλληση των εξαρτημάτων μεταξύ τους, μετά από την οποία οποιεσδήποτε διορθωτικές ενέργειες είναι πιο δύσκολες και ενέχουν τον κίνδυνο πρόκλησης βλάβης σε κάποιο εξάρτημα αν δεν υπάρχει σχετική εμπειρία στην κόλληση ηλεκτρονικών εξαρτημάτων.

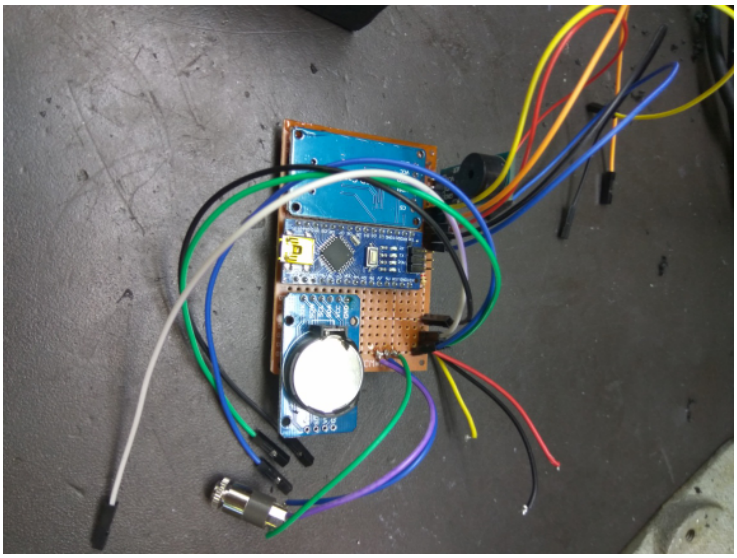


Εικόνα 3.2: Σύνδεση κυκλώματος με χρήση breadboard

Στο δεύτερο στάδιο, τα διάφορα εξαρτήματα συνδέθηκαν με το Arduino Nano με χρήση διάτρητης πλακέτας (PCB) η οποία διαθέτει επαφές (ευρύτερα γνωστές και ως “πίστες”) στις οποίες τοποθετούνται τα εξαρτήματα και παράλληλα μπορούν να κολληθούν, τόσο για τη στήριξή τους όσο και για τη δημιουργία ηλεκτρικών διαδρομών. Στις παρακάτω εικόνες φαίνονται διάφορα στιγμιότυπα από τη συναρμολόγηση στο δεύτερο αυτό στάδιο.



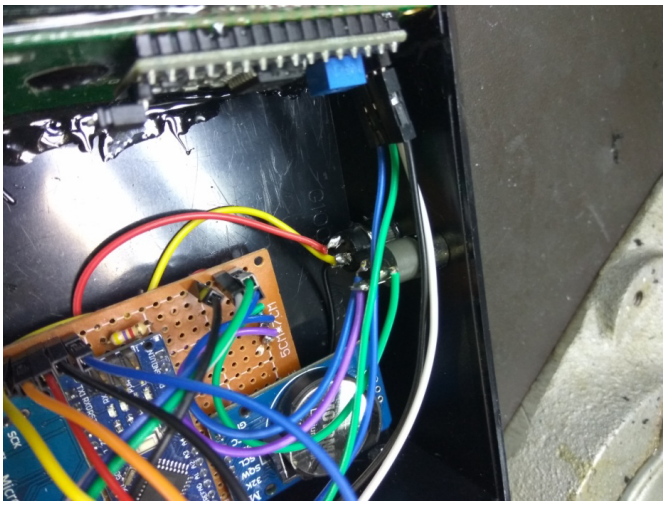
Εικόνα 3.3: Στάδια σύνδεσης κυκλώματος σε PCB (1)



Εικόνα 3.4: Στάδια σύνδεσης κυκλώματος σε PCB (2)

Το τελικό κύκλωμα τοποθετήθηκε μέσα σε ένα πλαστικό κουτί στο οποίο δημιουργήθηκαν οπές για την τοποθέτηση της οθόνης, των κουμπιών, τη δημιουργία πρόσβασης στη θύρα USB της πλακέτας Arduino, στον αισθητήρα εξωτερικής θερμοκρασίας και στην κάρτα microSD.

Εικόνα 3.5: Τοποθέτηση οθόνης και κουμπιών



Εικόνα 3.6: Τοποθέτηση πλακέτας μέσα στο κουτί

Εικόνα 3.7: Αισθητήρας DHT11 σε ξεχωριστό αποσπώμενο κουτί

Εικόνα 3.8: Εξωτερική εμφάνιση ολοκληρωμένης κατασκευής

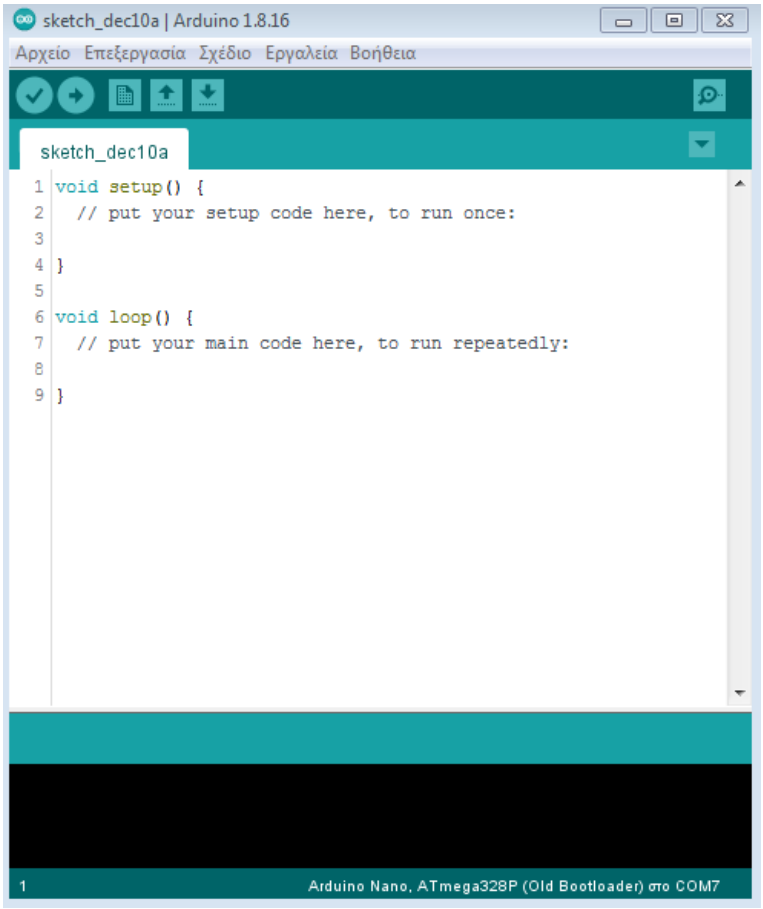
ΚΕΦΑΛΑΙΟ 4 - ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΑΝΑΛΥΣΗ ΛΟΓΙΣΜΙΚΟΥ

Στο κεφάλαιο αυτό αρχικά γίνεται μία αναφορά στο περιβάλλον προγραμματισμού και τις δυνατότητες που παρέχει η πλατφόρμα του Arduino. Έπειτα, αναφέρονται οι βιβλιοθήκες που είναι απαραίτητες για το πρόγραμμα του θερμομέτρου. Στη συνέχεια περιγράφεται η διαδικασία προγραμματισμού της πλακέτας. Τέλος, αναλύονται κάποια βασικά μέρη του προγράμματος της συσκευής.

4.1 Το περιβάλλον Arduino IDE

Όπως έχει αναφερθεί και στο Κεφάλαιο 1, ένα από τα κύρια χαρακτηριστικά της πλατφόρμα Arduino είναι η ευκολία με την οποία μπορεί να προγραμματιστεί. Ένας από τους πιο προσφιλείς τρόπους προγραμματισμού της είναι μέσω του Arduino IDE. Το περιβάλλον αυτό χρησιμοποιείται για τη συγγραφή προγραμμάτων, τη μεταγλώττισή τους και τη μεταφόρτωσή τους στην πλακέτα Arduino. Το περιβάλλον αυτό αποτελεί επίσημο λογισμικό της εταιρείας που παράγει τις πλακέτες Arduino και είναι ανοιχτού κώδικα. Επίσης είναι διαθέσιμο για τα λειτουργικά συστήματα Windows, Linux και Mac OS.[9]

Το περιβάλλον υποστηρίζει τη συγγραφή προγραμμάτων σε γλώσσες C και C++, πάντα όμως υπό τη δομή που επιβάλλει η διάλεκτος του Arduino. Συγκεκριμένα, ο κώδικας που θα γραφτεί, απαιτείται να περιλαμβάνει δύο συναρτήσεις, την setup() η οποία εκτελείται κάθε φορά που επανεκκινείται η πλακέτα Arduino και η loop() η οποία εκτελείται σε έναν άπειρο βρόχο (infinite loop) μέχρι να διακοπεί η τροφοδοσία τάσης στην πλακέτα.



Εικόνα 4.1: Γραφικό περιβάλλον Arduino IDE

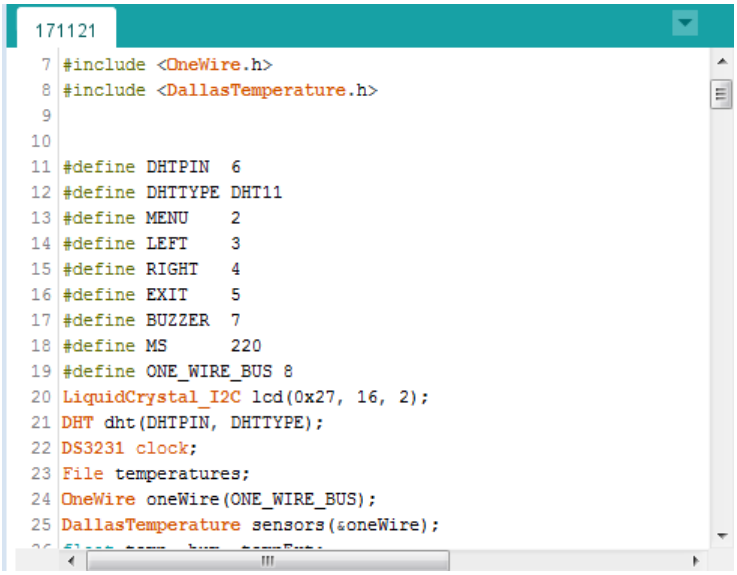
Εντός της συνάρτησης setup θα προστεθεί κώδικας ο οποίος είναι απαραίτητος για την αρχικοποίηση της πλακέτας (δημιουργία αντικειμένων, αρχικοποίηση μεταβλητών κλπ) και εντός της συνάρτησης loop θα προστεθεί ο κώδικας για τις λειτουργίες που θα επιτελεί η πλακέτα. Προφανώς, είναι δυνατή και η συγγραφή επιπλέον συναρτήσεων οι οποίες θα καλούνται εντός των προαναφερθέντων.

Τα προγράμματα που γράφονται στο Arduino IDE ονομάζονται sketches και αποθηκεύονται σαν αρχεία με την κατάληξη .ino. Επίσης, πριν την έκδοση 1.0 τα αποθηκευμένα sketches αποθηκεύονταν με την κατάληξη .pde.[10]

4.2 Συστατικά μέρη Arduino IDE

Κάποια από τα βασικά συστατικά μέρη του Arduino IDE είναι τα εξής:

Επεξεργαστής κειμένου



Εικόνα 4.2: Επεξεργαστής κειμένου

Ο επεξεργαστής κειμένου περιλαμβάνει την περιοχή συγγραφής του προγράμματος και συνδυάζεται με δυνατότητες που περιλαμβάνονται γενικότερα σε επεξεργαστές κειμένου σε άλλα περιβάλλοντα προγραμματισμού. Πιο συγκεκριμένα, κάποιες από τις πιο χρήσιμες δυνατότητες είναι οι εξής:[10]

- Αναίρεση / Επανάληψη
- Αποκοπή / Επικόλληση
- Αντιγραφή / Αντιγραφή για φόρουμ / Αντιγραφή ως HTML
- Σχολιασμός / Αποσχολιασμός
- Αύξηση / Μείωση εσοχής

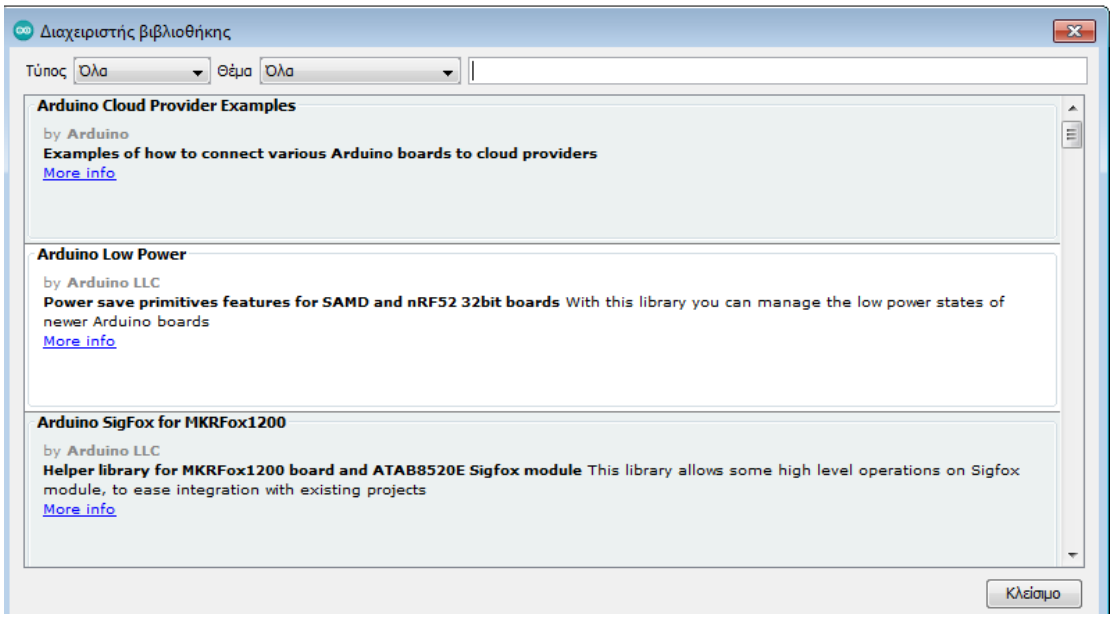
Επίσης, στο περιβάλλον υπάρχουν και οι λειτουργίες επί των αρχείων δηλαδή:

- Δημιουργία / Άνοιγμα
- Αποθήκευση / Αποθήκευση ως…

Διαχειριστής Βιβλιοθηκών

Οι βιβλιοθήκες είναι κομμάτια κώδικα οι οποίες επεκτείνουν τη λειτουργικότητα του κώδικα και διευκολύνουν προγραμματιστικά τη σύνδεση μεταξύ της πλακέτας Arduino και των διάφορων αισθητήρων. Για

παράδειγμα, για τη σύνδεση με τον αισθητήρα DHT11 υπάρχει η αντίστοιχη βιβλιοθήκη, η οποία θα αναφερθεί παρακάτω, και παρέχει τις απαραίτητες συναρτήσεις για την επικοινωνία της πλακέτας με τον αισθητήρα. Προκειμένου να εγκατασταθούν οι βιβλιοθήκες στο Arduino IDE, ο πιο ενδεδειγμένος τρόπος είναι η χρήση του Διαχειριστή Βιβλιοθηκών.[10]



Εικόνα 4.3: Διαχειριστής βιβλιοθηκών

Ο διαχειριστής βιβλιοθηκών επιτρέπει την εγκατάσταση και χρήση μιας πληθώρας από διαθέσιμες βιβλιοθήκες στο IDE. Υπάρχει η δυνατότητα επισκόπησης των διαθέσιμων προς εγκατάσταση βιβλιοθηκών βάσει της κατηγορίας τους δηλαδή:

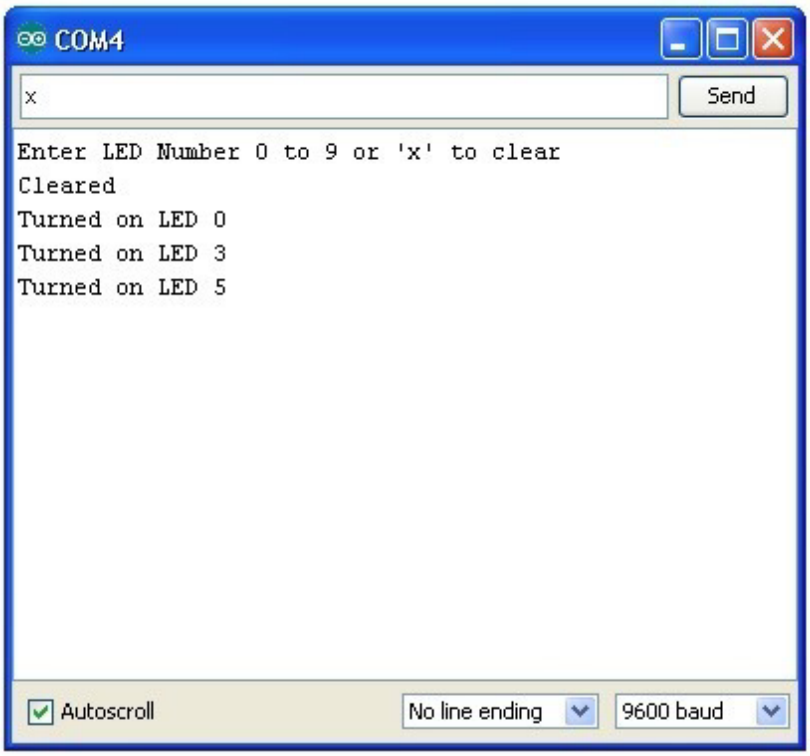
- Επικοινωνία
- Επεξεργασία δεδομένων
- Αποθήκευση δεδομένων
- Έλεγχος συσκευής
- Απεικόνιση
- Αισθητήρες
- Είσοδος / Έξοδος σήματος
- Χρονισμός

Στα θετικά της χρήσης του Διαχειριστή Βιβλιοθηκών συγκαταλέγεται το

γεγονός, ότι εντοπίζει τις ενημερώσεις των βιβλιοθηκών που υπάρχουν εγκατεστημένες στο IDE και προτρέπει τον χρήστη να τις εγκαταστήσει. Σε διαφορετική περίπτωση, ο χρήστης θα έπρεπε να ελέγχει το αποθετήριο κάθε εγκατεστημένης βιβλιοθήκης στο IDE για τυχόν ενημερώσεις και να τις εγκαθιστά.

Παρακολούθηση σειριακής επικοινωνίας

Όπως έχει αναφερθεί οι πλακέτες Arduino έχουν τη δυνατότητα σειριακής επικοινωνίας. Αυτή τη δυνατότητα εκμεταλλεύεται το Arduino IDE και παρέχει στον προγραμματιστή τη δυνατότητα παρακολούθησης της σειριακής αυτής επικοινωνίας. Συγκεκριμένα, η λειτουργία αυτή επιτρέπει την αποστολή καθώς επίσης και τη λήψη μηνυμάτων μεταξύ της πλακέτας Arduino και του IDE.



Εικόνα 4.4: Παρακολούθηση σειριακής επικοινωνίας

Η λειτουργία αυτή είναι πολύ σημαντική γιατί πρώτον επιτρέπει την αλληλεπίδραση μεταξύ χρήστη και πλακέτας (και κατ' επέκταση με το υπόλοιπα κύκλωμα) χωρίς να απαιτούνται επιπλέον διεπαφές, όπως για παράδειγμα οθόνες και πληκτρολόγια τα οποία όχι μόνο θα προσέθεταν επιπλέον όγκο στο συνολικό κύκλωμα, αλλά θα επιβάρυναν τη χρήση της πλακέτας.

Δεύτερον, η χρήση της λειτουργίας αυτής επιτρέπει την πιο εύκολη αποσφαλμάτωση (debugging) του κώδικα. Πιο συγκεκριμένα, από το IDE αυτό απουσιάζει η λειτουργία κάποιου debugger το οποίο θα διευκόλυνε την εύρεση των σφαλμάτων στον κώδικα. Επίσης, είναι κοινή πρακτική (αν και όχι αποδοτική εν γένει) κατά τη διαδικασία αποσφαλμάτωσης να χρησιμοποιούνται διάφορα μηνύματα εντός του κώδικα τα οποία να παρέχουν ενδείξεις για τη ροή εκτέλεσης του κώδικα. Στη λογική αυτής της πρακτικής μπορεί να χρησιμοποιηθεί η παρακολούθηση της σειριακής επικοινωνίας, με την οποία μπορούμε να ελέγξουμε σε πρωταρχικό στάδιο την ορθή επικοινωνία όλων των απαραίτητων συστατικών του κυκλώματος με την πλακέτα καθώς και την ορθή λειτουργία της ίδιας της πλακέτα όπως επίσης και τη ροή εκτέλεσης του προγράμματος για τυχόν λογικά ή άλλου είδους σφάλματα.

AVRDUDE

Το AVRDUDE (AVR Downloader/UploaDEr) είναι ένα εργαλείο το οποίο χρησιμοποιείται από το IDE και εγκαθίσταται ως μέρος της εργαλειοθήκης του τελευταίου και επιτελεί τον προγραμματισμό του Arduino χρησιμοποιώντας την τεχνική in-system programming (ISP)[17]. Εκτός από τη Flash μνήμη του Arduino (στην οποία αποθηκεύεται το πρόγραμμα) μπορεί να εκτελέσει εγγραφή και στην αντίστοιχη EEPROM.[23]

Παρόλο που το πρόγραμμα αυτό μπορεί να χρησιμοποιηθεί και ξεχωριστά από κάποιον χρήστη για τον προγραμματισμό πλακετών και διάφορων μικροελεγκτών κάνοντας χρήση κάποιας συσκευής programmer, ένας τυπικός τρόπος προγραμματισμού του Arduino είναι μέσω του IDE, όπου εκτελείται το συγκεκριμένο εργαλείο και επιτελεί τη λειτουργία αυτή. Το εργαλείο αυτό κατά την εκτέλεσή του παρέχει διάφορα μηνύματα για τα διάφορα στάδια εκτέλεσής του. Έτσι σε περίπτωση όπου δεν έχει εκτελεστεί με επιτυχία ο προγραμματισμός του Arduino, ο χρήστης είναι σε θέση να κατατοπιστεί για το είδος του σφάλματος που προέκυψε.[23]

Παραδείγματα

Όπως έχει ήδη αναφερθεί, το Arduno IDE είναι πολύ φιλικό ακόμα και για όσους δεν έχουν μεγάλη εμπειρία στον προγραμματισμό. Ένα ακόμα στοιχείο που επιτείνει το χαρακτηριστικό αυτό είναι το ότι παρέχει έτοιμα παραδείγματα κώδικα τα οποία μπορεί κάποιος να μεταφορτώσει απευθείας στην πλακέτα Arduino που διαθέτει. Με αυτόν τον τρόπο μπορεί αφενός μεν να προγραμματίσει την πλακέτα του με έτοιμα βασικής λειτουργικότητας προγράμματα χωρίς να γράψει καθόλου κώδικα,

αφετέρου δε να διαπιστώσει τον τρόπο λειτουργίας των προγραμμάτων αυτών στην πλακέτα. Το IDE διαθέτει διάφορες βασικές κατηγορίες από παραδείγματα προγραμμάτων αλλά και κάθε βιβλιοθήκη που εγκαθίσταται σε αυτό προσθέτει τη δική του κατηγορία προγραμμάτων.

4.3 Απαραίτητες βιβλιοθήκες

Οι βιβλιοθήκες οι οποίες είναι απαραίτητες για τη λειτουργία του προγράμματος του θερμομέτρου είναι οι εξής:

1) DHT Sensor Library (https://github.com/adafruit/DHT-sensor-library): Επιτρέπει την επικοινωνία του Arduino με τον αισθητήρα DHT11

2) Adafruit Unified Sensor Driver (https://github.com/adafruit/Adafruit_Sensor). Βιβλιοθήκη που περιέχει εξαρτήσεις για τη βιβλιοθήκη της §1.

3) DS3231 (https://github.com/NorthernWidget/DS3231): Επιτρέπει την επικοινωνία του DS3231 RTC με το Arduino.

4) LiquidCrystal I2C (https://github.com/johnrickman/LiquidCrystal_I2C): Επιτρέπει την επικοινωνία του Arduino με την οθόνη LCD μέσω του πρωτοκόλλου I2C.

5) SD (Built-in by Arduino): Επιτρέπει την επικοινωνία του Arduino με το module της κάρτας SD.

6) SPI (Built-in by Arduino): Επιτρέπει την επικοινωνία του Arduino με συσκευές που χρησιμοποιούν το πρωτόκολλο SPI (στην περίπτωσή μας το module της κάρτας SD). Απαιτείται συνδυαστικά με τη βιβλιοθήκη της §5.

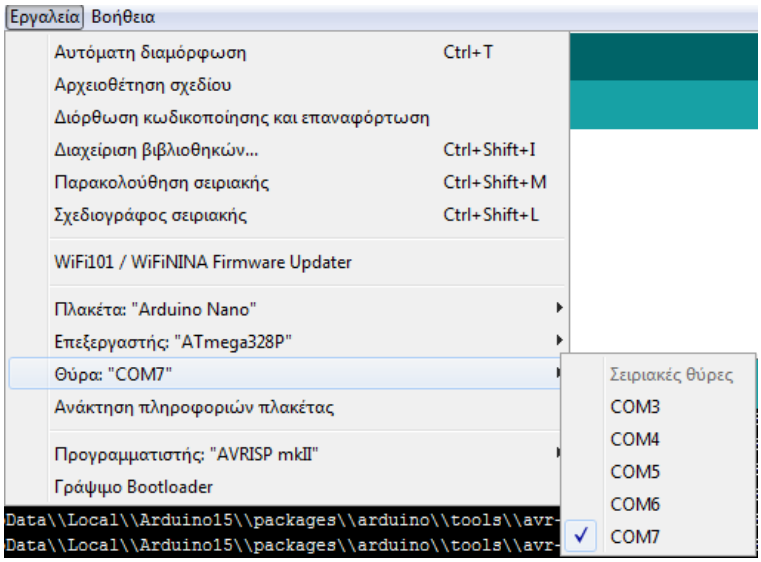
7) OneWire (https://www.pjrc.com/teensy/td_libs_OneWire.html): Επιτρέπει την επικοινωνία με συσκευές που χρησιμοποιούν το πρωτόκολλο επικοινωνίας OneWire (στην περίπτωσή μας ο αισθητήρας DS18B20). Απαιτείται συνδυαστικά με τη βιβλιοθήκη της §8.

8) Dallas Temperature (https://github.com/milesburton/Arduino-Temperature-Control-Library): Επιτρέπει την επικοινωνία με τον αισθητήρα DS18B20.

4.4 Μεταγλώττιση προγράμματος και μεταφόρτωση στην πλακέτα Arduino Nano

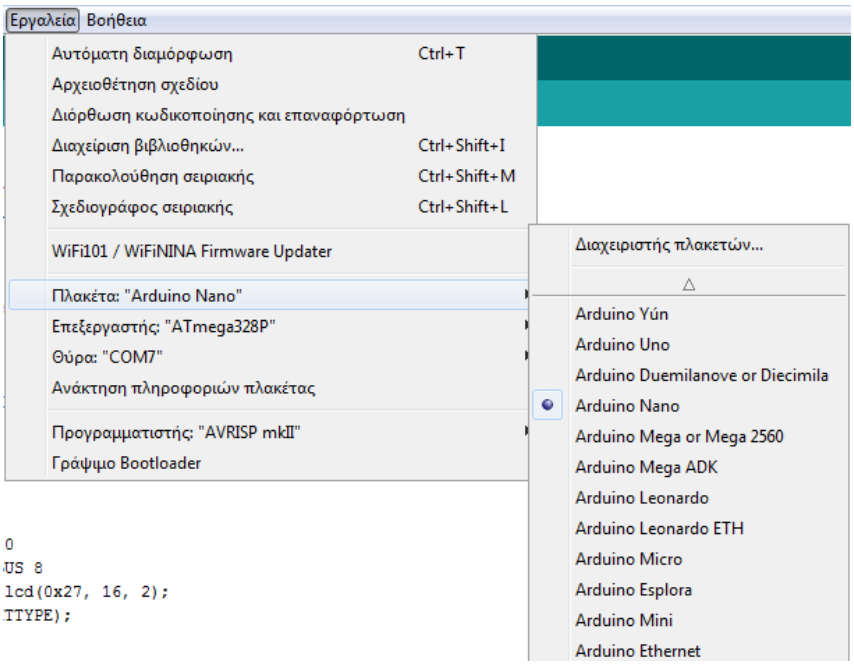
Αφού ολοκληρωθεί η συγγραφή του κώδικα, τα επόμενα βήματα είναι η

μεταγλώττιση του κώδικα και ο προγραμματισμός του μικροελεγκτή στην πλακέτα. Πριν προβούμε στα βήματα αυτά θα πρέπει να ελέγξουμε ότι έχουμε επιλέξει τη σωστή θύρα επικοινωνίας με την οποία συνδέεται η πλακέτα με τον υπολογιστή και κατ' επέκταση με το Arduino IDE[24].



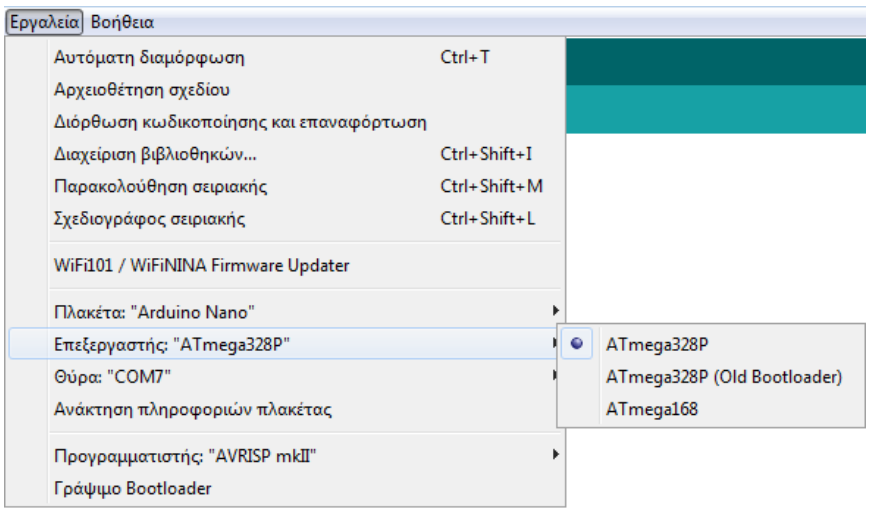
Εικόνα 4.5: Επιλογή θύρας επικοινωνίας

Έπειτα, πρέπει να επιλέξουμε το σωστό μοντέλο της πλακέτας που θέλουμε να προγραμματίσουμε, στην περίπτωση μας τη Nano.



Εικόνα 4.6: Επιλογή πλακέτας

Για την πλακέτα Nano θα πρέπει να επιλέξουμε, τέλος, το σωστό είδος επεξεργαστή το οποίο διαθέτει η συγκεκριμένη πλακέτα που έχουμε στην κατοχή μας.



Εικόνα 4.7: Επιλογή επεξεργαστή πλακέτας

Για τα βήματα της μεταγλώττισης και της επικύρωσης το IDE προσφέρει τις αντίστοιχες λειτουργίες για τις οποίες υπάρχουν οι αντίστοιχες συντομεύσεις.

Εικόνα 4.8: Συντομεύσεις επικύρωσης και ανεβάσματος

Με την επιλογή της επικύρωσης, η οποία έχει το αντίστοιχο κουμπί συντόμευσης (✔), ελέγχεται αρχικά ο κώδικας για συντακτικά και άλλου είδους σφάλματα τα οποία δε θα επιτρέψουν στον κώδικα να μεταγλωττιστεί. Αν ο κώδικας δεν περιέχει σφάλματα, τότε μεταγλωττίζεται και στο τέλος της μεταγλώττισης εμφανίζεται μήνυμα που δείχνει τον συνολικό χώρο που θα καταλάβει το πρόγραμμα και οι καθολικές μεταβλητές του στη μνήμη της πλακέτας.

Με την επιλογή του ανεβάσματος, η οποία έχει το αντίστοιχο κουμπί συντόμευσης (→), εκτελείται αρχικά η λειτουργία της επικύρωσης που είδαμε παραπάνω και έπειτα γίνεται το ανέβασμα του αρχείου μεταγλώττισης σην πλακέτα. Για το ανέβασμα χρησιμοποιείται το εργαλείο AVRDUDE όπως αναφέρθηκε στην παράγραφο 4.2.

Θα μπορούσε κάποιος να αναρωτηθεί ποια είναι η σημασία της επιλογής της επικύρωσης από τη στιγμή που η λειτουργία αυτή εκτελείται στο πρώτο στάδιο της λειτουργίας του ανεβάσματος. Είναι μια καλή πρακτική

να χρησιμοποιείται η λειτουργία της επικύρωσης πριν την εκτέλεση του βήματος του ανεβάσματος ακόμα και σε στιγμές ανεξάρτητες από αυτό, προκειμένου να ελεγξουμε γρηγορότερα ότι το πρόγραμμα δε διαθέτει σφάλματα.

4.5 Βασικά μέρη προγράμματος του συστήματος

Στο υποκεφάλαιο αυτό θα γίνει αναφορά και θα αναλυθούν κάποια μέρη του κώδικα. Επισημαίνεται ότι το σύνολο του κώδικα παρατίθεται στο Παράρτημα.

Συνάρτηση setup

Η συνάρτηση setup, η οποία αποτελεί βασική συνάρτηση για κάθε πρόγραμμα Arduino, εκτελείται μία φορά όταν γίνεται εκκίνηση της πλακέτας (επί της ουσίας εφαρμόζεται τάση στην πλακέτα) και περιέχει εντολές οι οποίες καθορίζουν τη λειτουργία των ακροδεκτών επικοινωνίας (δηλαδή αν είναι Εισόδου ή Εξόδου) και εκκινούν τη διαδικασία επικοινωνίας με τα διάφορα modules που είναι συνδεδεμένα στο Arduino. Στη δική μας περίπτωση, γίνεται εκκίνηση της επικοινωνίας με τον αισθητήρα DHT11, την οθόνη LCD, τον αισθητήρα DS18B20 και το module της κάρτας SD. Επίσης, επειδή οι τιμές των ρυθμίσεων για τους διάφορους συναγερμούς είναι αποθηκευμένες στην EEPROM του Arduino, στη συνάρτηση αυτή γίνεται μία ανάθεση των τιμών αυτών σε τοπικές μεταβλητές του προγράμματος για να μπορούν να χρησιμοποιηθούν πιο γρήγορα από το πρόγραμμα κατά τον έλεγχο των ορίων. Σημειώνεται ότι η EEPROM του συγκεκριμένου Arduino προφορτώθηκε με τιμές στις θέσεις μνήμης στις οποίες το πρόγραμμα θα αναζητά τις τιμές των ρυθμίσεων με διαφορετικό sketch, πριν τον προγραμματισμό του Arduino με το τελικό του πρόγραμμα.

Συνάρτηση loop

Η συνάρτηση loop, η οποία αποτελεί τη δεύτερη βασική συνάρτηση για κάθε πρόγραμμα Arduino, εκτελείται σε αέναο βρόχο κατά τη διάρκεια λειτουργίας της πλακέτας και εντός αυτής εκτελείται η βασική λειτουργικότητα του προγράμματος. Συγκεκριμένα, γίνεται λήψη των τιμών από τους αισθητήρες, τυπώνονται τα διάφορα μηνύματα στην οθόνη LCD και ελέγχονται τα όρια των τιμών των συναγερμών προκειμένου να κληθούν οι ανάλογες συναρτήσεις συναγερμού. Επίσης, τα μεσάνυχτα κάθε ημέρας, οι μέγιστες τιμές των θερμοκρασιών και της υγρασίας καταγράφονται στην κάρτα SD.

Συνάρτηση adjust

Η συνάρτηση adjust χρησιμοποιείται για τις περιπτώσεις όπου ο χρήστης θέλει να ορίσει είτε νέα ημερομηνία είτε νέα ώρα. Συγκεκριμένα, ανάλογα

με τις παραμέτρους που λαμβάνει ορίζει είτε την ημερομηνία είτε την ώρα.

Ορισμοί τιμών συναγερμού

Οι συναρτήσεις setAlarmTemp, setAlarmHum και setAlarmDiff χρησιμοποιούνται για τον ορισμό των τιμών του συναγερμού. Κάθε φορά που ορίζεται νέο όριο συναγερμού, αυτό φορτώνεται και αντικαθιστά το προηγούμενο στην EEPROM, ούτως ώστε να διατηρηθεί η τιμή αυτή σε περίπτωση που διακοπεί η παροχή ρεύματος στην πλακέτα. Ταυτόχρονα, αποθηκεύεται και στην αντίστοιχη τοπική μεταβλητή του προγράμματος όπως αναφέρθηκε και παραπάνω στη συνάρτηση setup.

Συναγερμοί

Οι συναρτήσεις thermoalarm, hygroalarm και diffalarm χρησιμοποιούνται για την εκκίνηση των ενεργειών συναγερμού, δηλαδή την ενεργοποίηση του buzzer και την εμφάνιση των αντίστοιχων μηνυμάτων στην οθόνη LCD.

ΚΕΦΑΛΑΙΟ 5^{ο} – ΔΟΚΙΜΗ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΜΕΤΡΗΣΕΙΣ

Στο κεφάλαιο αυτό γίνεται έλεγχος λειτουργίας της συσκευής. Αρχικά ελέγχουμε ότι η απόκλιση των τιμών που λαμβάνουν οι δύο αισθητήρες θερμοκρασίας δεν επηρεάζουν την αξιοπιστία της συσκευής. Έπειτα, κάνουμε μία καταγραφή των τιμών που λαμβάνουν οι αισθητήρες για συγκεκριμένο χρονικό διάστημα.

5.1 Έλεγχος αποκλίσεων μεταξύ των μετρήσεων των αισθητήρων

Στο θερμόμετρό μας για τον έλεγχο της εσωτερικής θερμοκρασίας έχουμε χρησιμοποιήσει έναν αισθητήρα DHT11. Για τον έλεγχο της εξωτερικής θερμοκρασίας έχουμε χρησιμοποιήσει έναν αισθητήρα DS18B20. Όπως είναι αναμενόμενο και πολύ φυσιολογικό κάθε αισθητήριο παρουσιάζει διαφορετικές αποκλίσεις στις μετρήσεις. Ακόμα και με τη χρήση ίδιου τύπου αισθητήρων πάλι θα υπήρχαν αποκλίσεις στη μέτρηση ίδιων συνθηκών θερμοκρασίας.

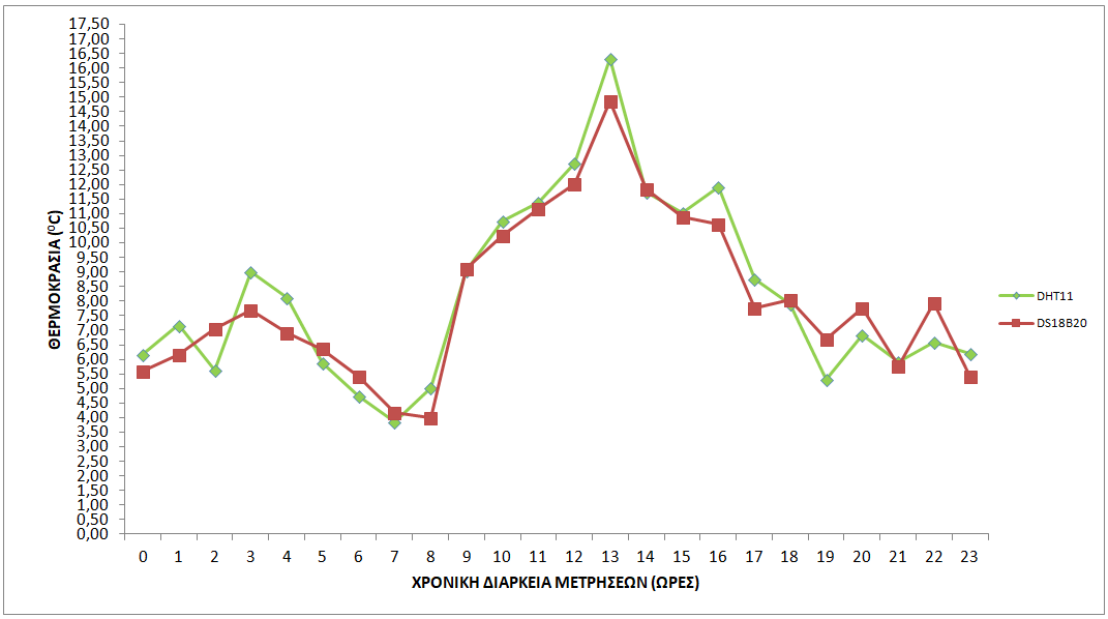
Επειδή μία από τις λειτουργίες του θερμομέτρου είναι η μέτρηση της διαφοράς μεταξύ της εσωτερικής και της εξωτερικής θερμοκρασίας και η ενεργοποίηση του αντίστοιχου συναγερμού, θα πρέπει να γίνει ένας έλεγχος ότι η απόκλιση που παρουσιάζουν στις μετρήσεις είναι εντός ενός αποδεκτού ορίου.

Για τον έλεγχο των αποκλίσεων, λήφθηκαν και καταγράφηκαν θερμοκρασίες από τα δύο αισθητήρια κάθε μία ώρα στη διάρκεια ενός 24ώρου, οι τιμές των οποίων φαίνονται στον Πίνακα 2.

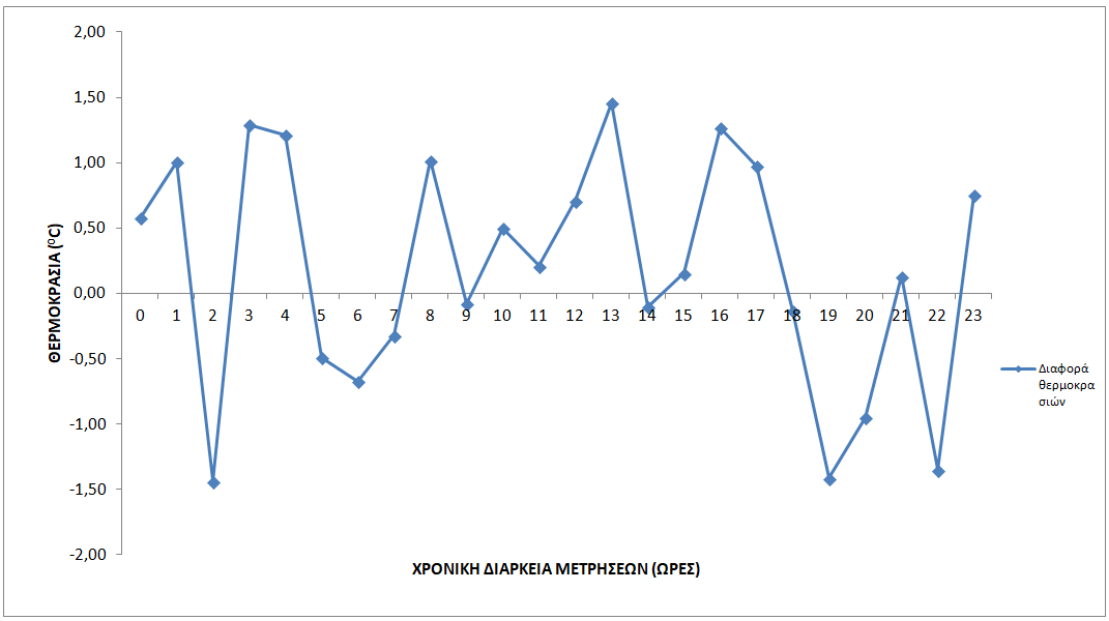
ΠΙΝΑΚΑΣ 2. Καταγραφή θερμοκρασιών από τους δύο αισθητήρες κατά τη διάρκεια 24ωρου (οι τιμές ελήφθησαν την 12/1/2022)

ΩΡΑ	DHT11	DS18B20	ΔΙΑΦΟΡΑ ΘΕΡΜΟΚΡΑΣΙΩΝ
0:00	6,18	5,60	0,58
1:00	7,18	6,17	1,01
2:00	5,61	7,05	-1,44
3:00	8,99	7,70	1,29
4:00	8,13	6,92	1,21
5:00	5,87	6,36	-0,49
6:00	4,73	5,41	-0,67
7:00	3,87	4,19	-0,32
8:00	5,01	3,99	1,02
9:00	9,05	9,13	-0,08
10:00	10,74	10,24	0,50
11:00	11,40	11,19	0,21
12:00	12,74	12,03	0,71
13:00	16,33	14,87	1,46
14:00	11,74	11,84	-0,10
15:00	11,04	10,89	0,15
16:00	11,92	10,65	1,27
17:00	8,75	7,78	0,97
18:00	7,91	8,04	-0,13
19:00	5,29	6,71	-1,41
20:00	6,83	7,78	-0,95
21:00	5,92	5,78	0,13
22:00	6,58	7,93	-1,35
23:00	6,19	5,43	0,76

Από τον πίνακα αυτόν προκύπτουν οι παρακάτω γραφικές παραστάσεις όπου στην πρώτη αναπαρίστανται οι καταγραφόμενες θερμοκρασίες με την πάροδο του χρόνου για τα δύο αισθητήρια και στη δεύτερη καταγράφονται οι αποκλίσεις μεταξύ των θερμοκρασιών που καταγράφηκαν με την πάροδο του χρόνου.



Σχήμα 1: Τιμές θερμοκρασιών 24ώρου από τους δύο αισθητήρες



Σχήμα 2 : Διαφορές θερμοκρασιών δύο αισθητήρων 24ώρου

Για τις καταγραφείσες τιμές ισχύουν τα ακόλουθα:

- Η μέση τιμή της διαφοράς θερμοκρασιών είναι 0,18^{O}C.
- Η μέγιστη τιμή της διαφοράς θερμοκρασιών είναι 1,46^{O}C.

- Η ελάχιστη τιμή της διαφοράς θερμοκρασιών είναι -1,44 ^{O}C.
- Η τυπική απόκλιση της διαφοράς θερμοκρασιών είναι 0,88^{O}C.

Συμπερασματικά, καταλήγουμε ότι η μέτρηση των θερμοκρασιών από τα δύο αισθητήρια παρουσιάζει διαφορά η οποία βρίσκεται εντός ενός αποδεκτού πλαισίου όσον αφορά τη λειτουργία του θερμομέτρου για την παρακολούθηση της διαφοράς των δύο θερμοκρασιών.

5.2 Έλεγχος μετρήσεων συσκευής σε διαφορετικούς χώρους

Εφόσον διαπιστώθηκε ότι η μέτρηση της θερμοκρασίας από τα αισθητήρια της συσκευής για τις ίδιες θερμοκρασίες αναφοράς (στον ίδιο χώρο δηλαδή) παρουσιάζει αξιοπιστία, μένει να διαπιστωθεί η λειτουργικότητα της συσκευής υπό την έννοια ότι η μέτρηση των θερμοκρασιών σε δύο διαφορετικούς χώρους λειτουργεί κανονικά. Για τον σκοπό αυτό, ο αισθητήρας DHT11 τοποθετήθηκε σε χώρο εντός οικίας και ο αισθητήρας DS18B20 εκτός αυτής. Έπειτα, λήφθηκαν και καταγράφηκαν οι τιμές από τους δύο αισθητήρες κάθε μία ώρα κατά τη διάρκεια ενός 24ώρου.

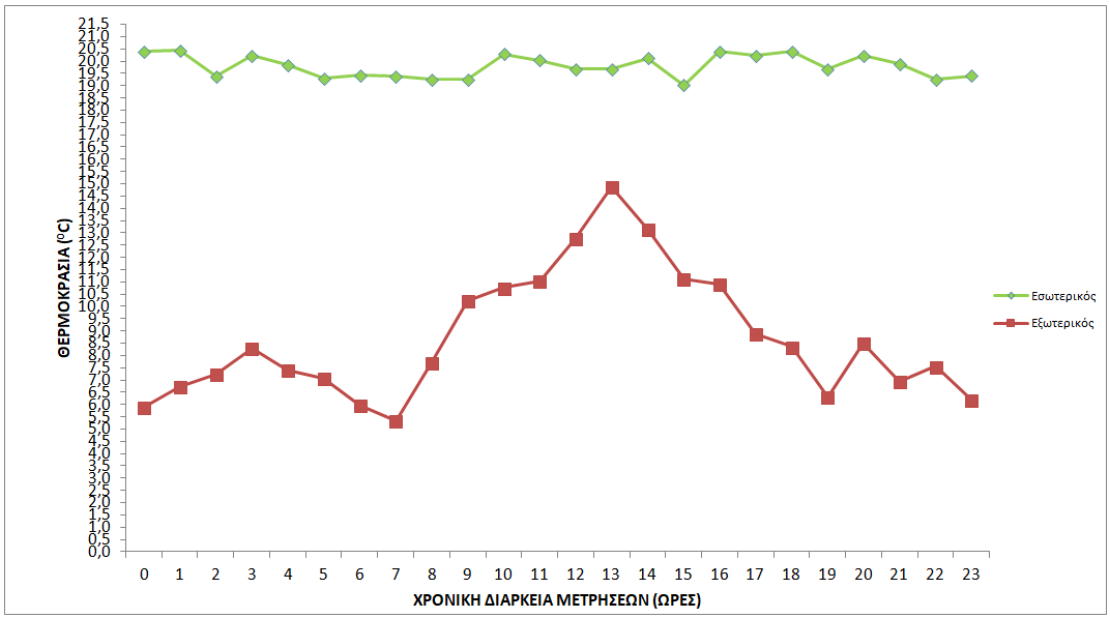
Οι τιμές της θερμοκρασίας που καταγράφηκαν από τους δύο χώρους φαίνονται στον παρακάτω Πίνακα 3:

ΠΙΝΑΚΑΣ 3. Καταγραφή θερμοκρασιών εσωτερικά και εξωτερικά οικίας κατά τη διάρκεια 24ωρου (οι τιμές ελήφθησαν την 23/1/2022)

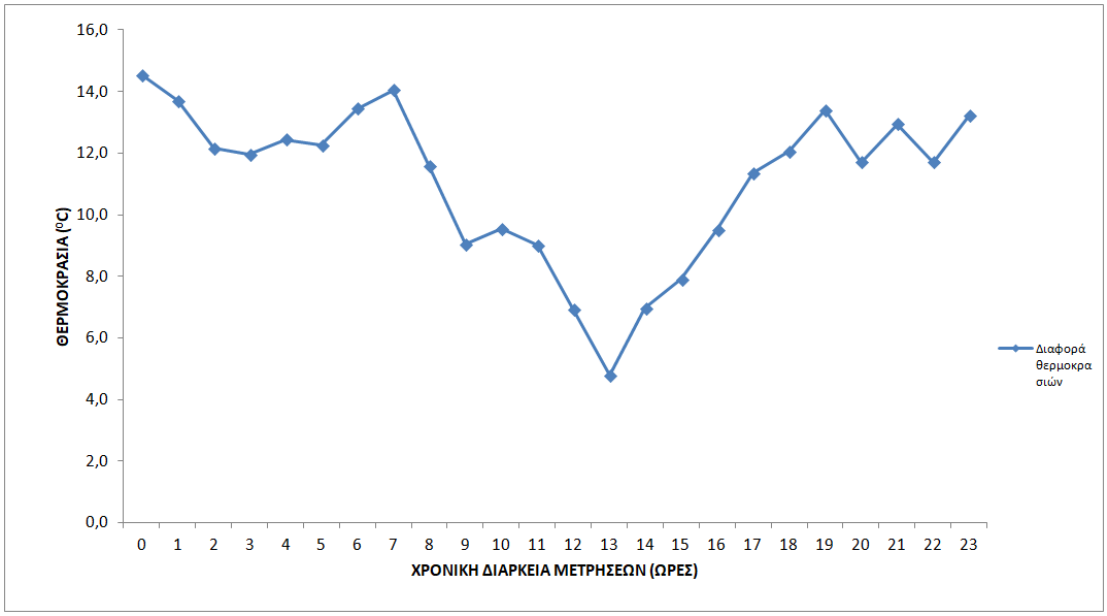
ΩΡΑ	ΕΣΩΤΕΡΙΚΟΣ ΧΩΡΟΣ	ΕΞΩΤΕΡΙΚΟΣ ΧΩΡΟΣ	ΔΙΑΦΟΡΑ ΘΕΡΜΟΚΡΑΣΙΩΝ
0:00	20,4	5,9	14,5
1:00	20,4	6,7	13,7
2:00	19,4	7,2	12,2
3:00	20,2	8,3	12,0
4:00	19,8	7,4	12,4
5:00	19,3	7,1	12,2
6:00	19,4	6,0	13,4
7:00	19,4	5,3	14,1
8:00	19,3	7,7	11,6
9:00	19,3	10,2	9,0
10:00	20,3	10,8	9,5
11:00	20,0	11,0	9,0
12:00	19,7	12,8	6,9
13:00	19,7	14,9	4,8
14:00	20,1	13,2	7,0
15:00	19,0	11,1	7,9
16:00	20,4	10,9	9,5
17:00	20,2	8,9	11,4
18:00	20,4	8,4	12,0

19:00	19,7	6,3	13,4
20:00	20,2	8,5	11,7
21:00	19,9	6,9	12,9
22:00	19,3	7,6	11,7
23:00	19,4	6,2	13,2

Από τον παραπάνω πίνακα προκύπτουν οι εξής γραφικές παραστάσεις:



Σχήμα 3: Τιμές θερμοκρασιών 24ώρου σε διαφορετικούς χώρους



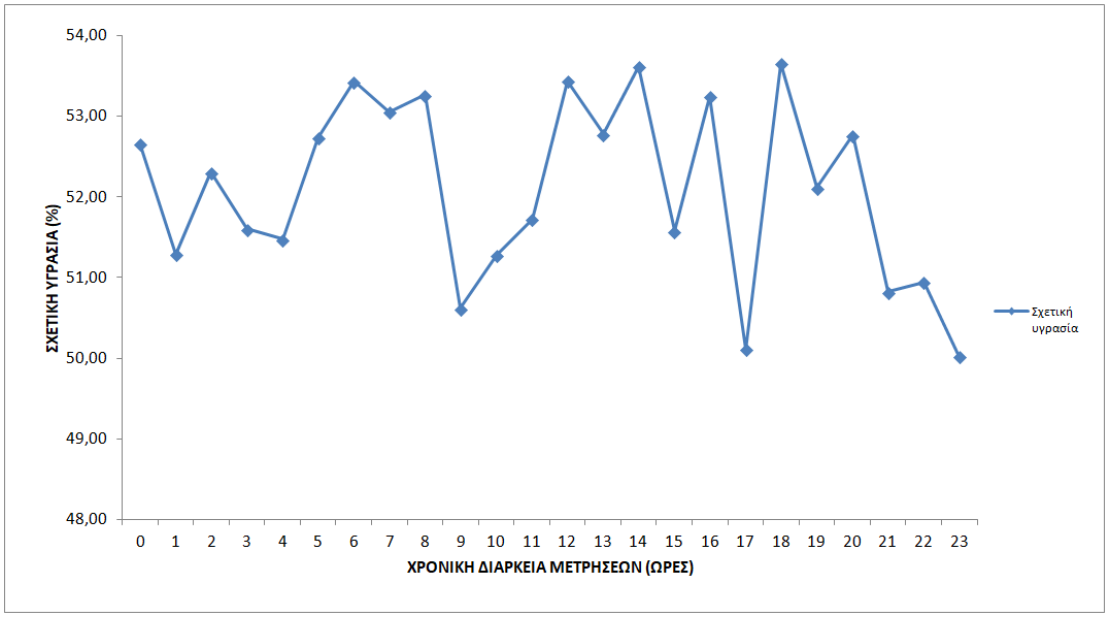
Σχήμα 4: Διαφορές θερμοκρασιών εσωτερικού - εξωτερικού χώρου 24ώρου

Οι τιμές της υγρασίας που καταγράφηκαν από τον εσωτερικό χώρο φαίνονται στον παρακάτω Πίνακα 4:

ΠΙΝΑΚΑΣ 4. Καταγραφή σχετικής υγρασίας εντός οικίας κατά τη διάρκεια 24ωρου (οι τιμές ελήφθησαν την 23/1/2022)

ΩΡΑ	ΣΧΕΤΙΚΗ ΥΓΡΑΣΙΑ
0:00	52,66
1:00	51,29
2:00	52,31
3:00	51,60
4:00	51,47
5:00	52,74
6:00	53,43
7:00	53,05
8:00	53,26
9:00	50,62
10:00	51,28
11:00	51,72
12:00	53,44
13:00	52,78
14:00	53,62
15:00	51,58
16:00	53,24
17:00	50,12
18:00	53,66
19:00	52,11
20:00	52,76
21:00	50,82
22:00	50,95
23:00	50,02

Η γραφική παράσταση που προκύπτει από τον πίνακα αυτό είναι η εξής:



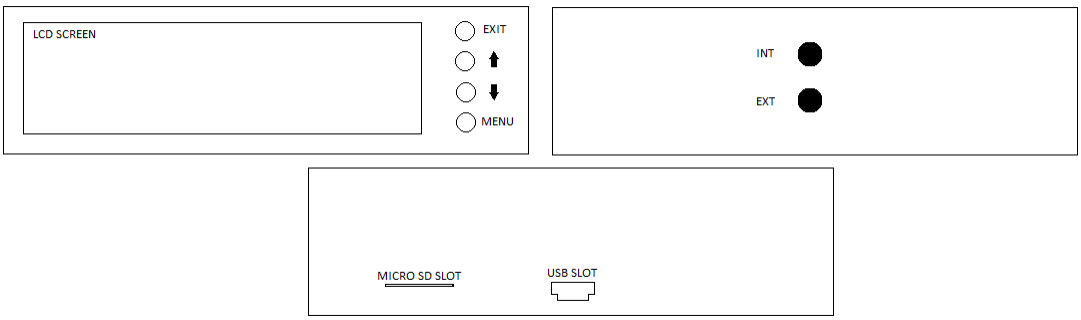
Σχήμα 5: Τιμές υγρασίας 24ώρου από τον εσωτερικό χώρο

Από τα παραπάνω προκύπτει το συμπέρασμα ότι η συσκευή λαμβάνει σωστές μετρήσεις για τη θερμοκρασία και την υγρασία. Το συμπέρασμα αυτό προκύπτει από το γεγονός ότι η εσωτερική θερμοκρασία της οικίας παραμένει σχεδόν σταθερή (σε ένα εύρος τιμών μεταξύ 19^{O}C και 20,4^{O}C) όπως επίσης και η σχετική υγρασία (σε ένα εύρος τιμών μεταξύ 50,02% και 53,66%) ενώ όσον αφορά την εξωτερική θερμοκρασία οι τιμές που ελήφθησαν είναι παρόμοιες με αυτές ενός χειμωνιάτικου 24ώρου.

ΚΕΦΑΛΑΙΟ 6^{ο} – ΟΔΗΓΙΕΣ ΧΡΗΣΗ ΚΑΙ ΡΥΘΜΙΣΗΣ

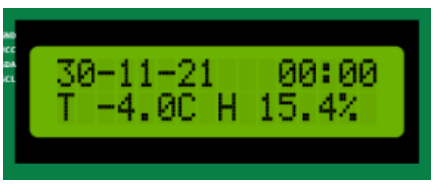
Ο ψηφιακός μετρητής θερμοκρασίας – υγρασίας τροφοδοτείται μέσω θύρας Mini-B USB. Επίσης, για τη διατήρηση της ώρας και της ημερομηνίας της συσκευής σε περίπτωση που αφαιρεθεί από αυτήν η παροχή ρεύματος, απαιτείται η τοποθέτηση μίας μπαταρίας CR2032. Τέλος, για την καταγραφή των μέγιστων τιμών του 24ώρου απαιτείται η τοποθέτηση μίας κάρτας microSD με διαμόρφωση είτε FAT16 είτε FAT32.

Η συσκευή στην πρόσοψη διαθέτει μία οθόνη LCD δύο γραμμών και τέσσερα κουμπιά σε κατακόρυφη διάταξη τα οποία έχουν τους εξής συμβολισμούς αντίστοιχα (από πάνω προς τα κάτω): EXIT, ↑ , ↓ και MENU. Στην πίσω όψη της συσκευής υπάρχει η υποδοχή για το καλώδιο Mini USB και η υποδοχή της κάρτας microSD. Στην αριστερή πλάγια όψη βρίσκονται οι υποδοχές στις οποίες συνδέονται τα καλώδια του εσωτερικού και του εξωτερικού αισθητήρα.



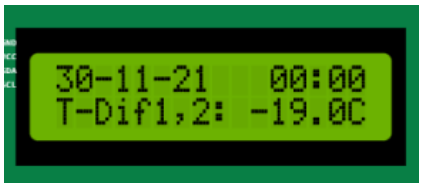
Εικόνα 6.1: Όψεις συσκευής

Κατά την τροφοδοσία της συσκευής με ρεύμα εμφανίζεται η οθόνη ενδείξεων, η οποία στην πάνω γραμμή εμφανίζει την ημερομηνία και την ώρα και στην κάτω γραμμή εναλλάσσει ανά διάστημα 1,5s: α) την τιμή της θερμοκρασίας και της υγρασίας του εσωτερικού αισθητήρα, β) την τιμή της θερμοκρασίας του εξωτερικού αισθητήρα και γ) τη διαφορά θερμοκρασίας μεταξύ εσωτερικού και εξωτερικού αισθητήρα.



Εικόνα 6.5: Εσωτερική θερμοκρασία και υγρασία

Εικόνα 6.6: Εξωτερική θερμοκρασία



Εικόνα 6.7: Διαφορά θερμοκρασιών

Για την εφαρμογή διάφορων ρυθμίσεων στη συσκευή υπάρχει ένα διαθέσιμο μενού επιλογών από όπου ο χρήστης μπορεί να επιλέξει:

1) Ρύθμιση ώρας (Παράγραφος 6.1)
2) Ρύθμιση ημερομηνίας (Παράγραφος 6.2)
3) Ρύθμιση συναγερμού εσωτερικής θερμοκρασίας (Παράγραφος 6.3)
4) Ρύθμιση συναγερμού υγρασίας (Παράγραφος 6.4)
5) Ρύθμιση συναγερμού διαφοράς θερμοκρασιών (Παράγραφος 6.5)

Για ευκρινέστερη εμφάνιση, κάθε επιλογή του μενού εμφανίζεται μόνη της στην οθόνη. Τα ονόματα των διαθέσιμων επιλογών σε αντιστοιχία με την ανωτέρω λίστα είναι οι εξής:

1) SET TIME
2) SET DATE
3) SET ALARM TEMP
4) SET ALARM HUM
5) SET ALARM DIFF

6.1 Ρύθμιση ώρας

Για τη ρύθμιση της ώρας ακολουθούμε τα εξής βήματα:

1. Από την οθόνη ενδείξεων πιέζουμε το πλήκτρο MENU
2. Στην οθόνη πλοήγησης του μενού επιλογών πιέζουμε είτε το

πλήκτρο ↑ είτε το πλήκτρο ↓ για να βρούμε την επιλογή “SET TIME”.

Εικόνα 6.8: Επιλογή αλλαγής ώρας

3. Στην επιλογή “SET TIME” πιέζουμε το πλήκτρο MENU και θα βρεθούμε στην οθόνη αλλαγής της ώρας.

Εικόνα 6.9: Αλλαγή ώρας

4. Όταν εμφανιστεί η οθόνη αλλαγής της ώρας πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ και αλλάζουμε την ώρα.
5. Για να αλλάξουμε τα λεπτά, πιέζουμε το πλήκτρο ξανά MENU και μετά πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓.
6. Για να αποθηκεύσουμε τη ρύθμιση της ώρας πιέζουμε ξανά το MENU και αυτόματα μεταφερόμαστε στην οθόνη του μενού επιλογών.
7. Αν πιέσουμε το πλήκτρο EXIT οποιαδήποτε στιγμή που βρισκόμαστε στην λειτουργία ρύθμισης της ώρας, επιστρέφουμε στο μενού επιλογών χωρίς να αποθηκευτεί η ρύθμιση της ώρας που έχουμε κάνει.

6.2 Ρύθμιση ημερομηνίας

Για τη ρύθμιση της ημερομηνίας ακολουθούμε τα εξής βήματα:

1. Από την οθόνη ενδείξεων πιέζουμε το πλήκτρο MENU
2. Στην οθόνη πλοήγησης του μενού επιλογών πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ για να βρούμε την επιλογή “SET DATE”.

Εικόνα 6.10: Επιλογή αλλαγής ημερομηνίας

3. Στην επιλογή “SET DATE” πιέζουμε το πλήκτρο MENU και θα βρεθούμε στην οθόνη αλλαγής της ημερομηνίας. Η ημερομηνία έχει τη μορφή HH-MM-ΕΕ (Η: Ημέρα, Μ: Μήνας, Ε: Έτος)

Εικόνα 6.11: Αλλαγή ημερομηνίας

4. Όταν εμφανιστεί η οθόνη αλλαγής της ημερομηνίας πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ και αλλάζουμε την ημέρα.
5. Για να αλλάξουμε τον μήνα, πιέζουμε ξανά το πλήκτρο MENU και μετά πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓.
6. Για να αλλάξουμε το έτος, πιέζουμε ξανά το πλήκτρο MENU και μετά πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓.
7. Για να αποθηκεύσουμε τη ρύθμιση της ημερομηνίας πιέζουμε ξανά το MENU και αυτόματα μεταφερόμαστε στην οθόνη του μενού επιλογών.
8. Αν πιέσουμε το πλήκτρο EXIT οποιαδήποτε στιγμή που βρισκόμαστε στην λειτουργία ρύθμισης της ημερομηνίας, επιστρέφουμε στο μενού επιλογών χωρίς να αποθηκευτεί η ρύθμιση της ώρας που έχουμε κάνει.

6.3 Ρύθμιση συναγερμού εσωτερικής θερμοκρασίας

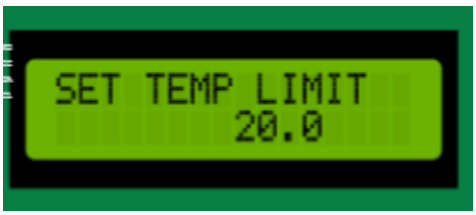
Για τη ρύθμιση της τιμής του συναγερμού εσωτερικής θερμοκρασίας ακολουθούμε τα εξής βήματα:

1. Από την οθόνη ενδείξεων πιέζουμε το πλήκτρο MENU

2. Στην οθόνη πλοήγησης του μενού επιλογών πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ για να βρούμε την επιλογή “SET ALARM TEMP”.

Εικόνα 6.12: Επιλογή ρύθμισης συναγερμού θερμοκρασίας

3. Στην επιλογή “SET ALARM TEMP” πιέζουμε το πλήκτρο MENU και θα βρεθούμε στην οθόνη αλλαγής της τιμής της θερμοκρασίας πάνω από την οποία θα ενεργοποιείται ο συναγερμός. Η τιμή αυτή μπορεί να κυμαίνεται από τους 0^{O}C έως τους 50^{O}C (μόνο ακέραιες τιμές).



Εικόνα 6.13: Τιμή θερμοκρασίας συναγερμού

4. Όταν εμφανιστεί η οθόνη αλλαγής του ορίου θερμοκρασίας πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ και αλλάζουμε το όριο.
5. Για να αποθηκεύσουμε τη ρύθμιση της θερμοκρασίας πιέζουμε ξανά το MENU και αυτόματα μεταφερόμαστε στην οθόνη του μενού επιλογών.
6. Αν πιέσουμε το πλήκτρο EXIT, επιστρέφουμε στο μενού επιλογών χωρίς να αποθηκευτεί η ρύθμιση της θερμοκρασίας που έχουμε κάνει.

6.4 Ρύθμιση συναγερμού υγρασίας

Για τη ρύθμιση της τιμής του συναγερμού θερμοκρασίας ακολουθούμε τα εξής βήματα:

1. Από την οθόνη ενδείξεων πιέζουμε το πλήκτρο MENU
2. Στην οθόνη πλοήγησης του μενού επιλογών πιέζουμε είτε το

πλήκτρο ↑ είτε το πλήκτρο ↓ για να βρούμε την επιλογή “SET ALARM HUM”.

Εικόνα 6.14: Επιλογή ρύθμισης συναγερμού υγρασίας

3. Στην επιλογή “SET ALARM HUM” πιέζουμε το πλήκτρο MENU και θα βρεθούμε στην οθόνη αλλαγής της τιμής της υγρασίας πάνω από την οποία θα ενεργοποιείται ο συναγερμός. Η τιμή αυτή μπορεί να κυμαίνεται μεταξύ των ποσοστών 20% έως 90% (μόνο ακέραιες τιμές).

Εικόνα 6.15: Τιμή υγρασίας συναγερμού

4. Όταν εμφανιστεί η οθόνη αλλαγής του ορίου υγρασίας πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ και αλλάζουμε το όριο.
5. Για να αποθηκεύσουμε τη ρύθμιση της υγρασίας πιέζουμε ξανά το MENU και αυτόματα μεταφερόμαστε στην οθόνη του μενού επιλογών.
6. Αν πιέσουμε το πλήκτρο EXIT, επιστρέφουμε στο μενού επιλογών χωρίς να αποθηκευτεί η ρύθμιση της υγρασίας που έχουμε κάνει.

6.5 Ρύθμιση συναγερμού διαφοράς θερμοκρασιών

Για τη ρύθμιση της τιμής του συναγερμού διαφοράς θερμοκρασιών ακολουθούμε τα εξής βήματα:

7. Από την οθόνη ενδείξεων πιέζουμε το πλήκτρο MENU
8. Στην οθόνη πλοήγησης του μενού επιλογών πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ για να βρούμε την επιλογή “SET ALARM

DIFF”.

Εικόνα 6.16: Επιλογή ρύθμισης συναγερμού διαφοράς

9. Στην επιλογή “SET ALARM DIFF” πιέζουμε το πλήκτρο MENU και θα βρεθούμε στην οθόνη αλλαγής της τιμής της θερμοκρασίας πάνω από την οποία θα ενεργοποιείται ο συναγερμός. Η τιμή αυτή μπορεί να κυμαίνεται από τους -20^{O}C έως τους 50^{O}C (μόνο ακέραιες τιμές).

Εικόνα 6.17: Διαφορά θερμοκρασίας συναγερμού

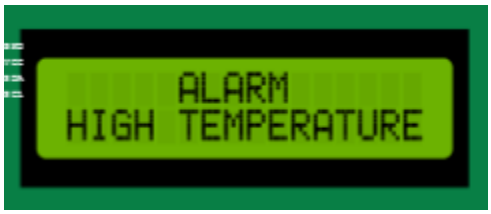
10. Όταν εμφανιστεί η οθόνη αλλαγής του ορίου θερμοκρασίας πιέζουμε είτε το πλήκτρο ↑ είτε το πλήκτρο ↓ και αλλάζουμε το όριο.

11. Για να αποθηκεύσουμε τη ρύθμιση της θερμοκρασίας πιέζουμε ξανά το MENU και αυτόματα μεταφερόμαστε στην οθόνη του μενού επιλογών.

12. Αν πιέσουμε το πλήκτρο EXIT, επιστρέφουμε στο μενού επιλογών χωρίς να αποθηκευτεί η ρύθμιση της θερμοκρασίας που έχουμε κάνει.

6.6 Ενεργοποίηση συναγερμών

Όταν η τιμή της εσωτερικής θερμοκρασίας ή της υγρασίας ή της διαφοράς θερμοκρασιών ξεπεράσει το όριο που έχει καθοριστεί, τότε ενεργοποιείται η λειτουργία του συναγερμού, δηλαδή ηχεί το buzzer και εμφανίζεται το αντίστοιχο προειδοποιητικό μήνυμα στην οθόνη για την ενεργοποίηση του συναγερμού θερμοκρασίας ή υγρασίας ή διαφοράς θερμοκρασιών.



Εικόνα 6.18: Συναγερμός θερμοκρασίας

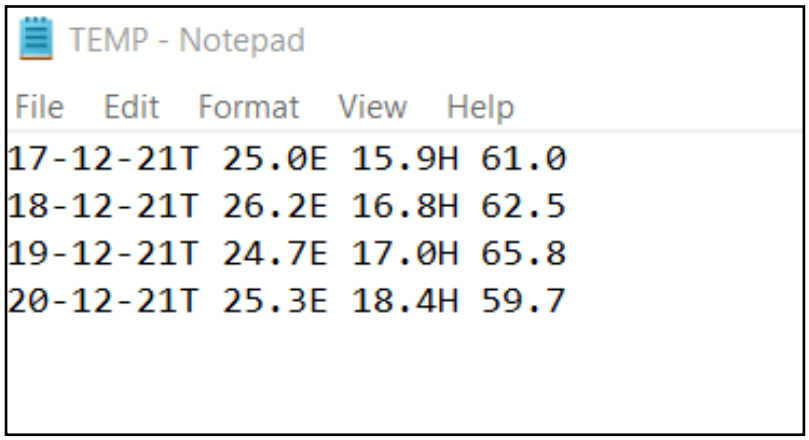
Εικόνα 6.19: Συναγερμός υγρασίας

Εικόνα 6.20: Συναγερμός διαφοράς θερμοκρασιών

Για την απενεργοποίηση του συναγερμού πιέζουμε το πλήκτρο MENU, οπότε εμφανίζεται η οθόνη ρύθμισης του ορίου για το συναγερμό εσωτερικής θερμοκρασίας ή υγρασίας ή διαφοράς θερμοκρασίας αντίστοιχα(βήμα 4 παραγράφων 6.3 ή 6.4 ή 6.5 αντίστοιχα).

6.7 Προβολή μέγιστων τιμών προηγούμενων ημερών

Η συσκευή αποθηκεύει στην κάρτα μνήμης τις μέγιστες τιμές της θερμοκρασίας (εσωτερικής και εξωτερικής) και της υγρασίας για κάθε ημέρα. Η αποθήκευση των τιμών γίνεται σε ένα αρχείο τύπου txt με όνομα TEMP. Ένα στιγμιότυπο αυτού του αρχείου φαίνεται στην παρακάτω εικόνα.



Εικόνα 6.21: Στιγμιότυπο αρχείου μέγιστων τιμών

Η μορφή κάθε εγγραφής είναι η εξής:

(Ημερομηνία)**T**(Εσωτερική Θερμοκρασία)**E**(Εξωτερική Θερμοκρασία)**H**(Υγρασία)

Για να προβάλλουμε τις τιμές αυτές αφαιρούμε την κάρτα μνήμης από την υποδοχή της συσκευής και με τη χρήση ενός card reader την τοποθετούμε σε υπολογιστή όπου ανοίγουμε το αρχείο TEMP.txt.

Επίσης, επειδή οι εγγραφές αποθηκεύονται στο αρχείο με συγκεκριμένη μορφή, είναι εύκολο το αρχείο αυτό να τεθεί ως είσοδος σε κάποιο άλλο πρόγραμμα το οποίο θα μπορέσει να το χρησιμοποιήσει για να δημιουργήσει, για παράδειγμα ένα διάγραμμα των μέγιστων αυτών τιμών σε σχέση με τη θερμοκρασία. Ένα δείγμα ψευδοκώδικα για τη διαδικασία αυτή θα μπορούσε να είναι ο εξής:

```
file = readFile("TEMP.txt")
time[], int_temp[],ext_temp[],hum[]
while (file.hasNextLine()){
    line = readNextLine()
    list_of_measurements = line.split(delimiters("T","E","H"))
    time.push(list_of_measurements(0)
    int_temp.push(list_of_measurements(1)
    ext_temp.push(list_of_measurements(2)
    hum.push(list_of_measurements(3)
}
plot(int_temp, time)
plot(ext_temp, time)
plot(hum, time)
```

Η δημιουργία ενός τέτοιου προγράμματος δεν ανήκει στο αντικείμενο της παρούσας εργασίας.

6.8 Τεχνικά χαρακτηριστικά συσκευής

Τάση λειτουργίας	5V (τροφοδοσία μέσω mini-B USB) Μπαταρία CR2032 για το ρολόι πραγματικού χρόνο
Βάρος	150gr
Διαστάσεις	Κύρια συσκευή: 100mmx100mmx40mm Αισθητήρας εσωτερικού χώρου: 45mmx15mmx30mm
Εύρος θερμοκρασίας αισθητήρα εσωτερικού χώρου	0°C έως 50°C
Εύρος σχετικής υγρασίας αισθητήρα υγρασίας	20% έως 90%
Εύρος θερμοκρασίας αισθητήρα εξωτερικού χώρου	-10°C έως 85°
Εύρος τιμών συναγερμού θερμοκρασίας	0°C έως 50°C
Εύρος τιμών συναγερμού υγρασίας	20% έως 90%
Εύρος τιμών συναγερμού διαφοράς θερμοκρασιών	-20°C έως 50°
Μέγιστη απόκλιση αισθητήρων θερμοκρασίας	$1,46^{O}C$

ΚΕΦΑΛΑΙΟ 7^{ο} – ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ

Η παρούσα διπλωματική πραγματεύτηκε το σχεδιασμό και τη δημιουργία μίας συσκευής μέτρησης της θερμοκρασίας και της υγρασίας. Για τις συνθήκες θερμοκρασίας τοποθετήθηκαν δύο αισθητήρες, ο ένας για τον έλεγχο της θερμοκρασίας ενός εσωτερικού χώρου και ο άλλος για τον έλεγχο της θερμοκρασίας ενός εξωτερικού χώρου. Παράλληλα το σύστημα έχει τη δυνατότητα οπτικής και ηχητικής ειδοποίησης σε περίπτωση όπου οι τιμές είτε της θερμοκρασίας είτε της υγρασίας ξεπεράσουν κάποιο όριο. Επίσης, υπάρχει η δυνατότητα καταγραφής των μέγιστων ημερήσιων τιμών σε κάρτα microSD.

Για τον έλεγχο ορθής λειτουργίας της συσκευής πραγματοποιήθηκαν μετρήσεις σε δύο φάσεις. Στην πρώτη φάση οι αισθητήρες τοποθετήθηκαν στον ίδιο χώρο και λάμβαναν μετρήσεις για ένα 24ωρο. Αυτό είχε ως σκοπό να διαπιστωθεί ότι οι τιμές που λάμβαναν είχαν ανεκτές αποκλίσεις μεταξύ τους λόγω του γεγονότος ότι οι αισθητήρες ήταν διαφορετικού τύπου. Στη δεύτερη φάση τοποθετήθηκαν εκεί όπου προορίζεται η χρήση τους, δηλαδή ο ένας σε εσωτερικό χώρο και ο άλλος σε εξωτερικό και λάμβαναν τιμές για ένα 24ωρο. Τα αποτελέσματα των μετρήσεων έδειξαν ότι αφενός δεν παρατηρούνται σημαντικές αποκλίσεις μεταξύ τους αφετέρου παρουσιάζουν πιστότητα στις μετρήσεις.

Όσον αφορά τις δυσκολίες που παρουσιάστηκαν, αυτές σχετίζονται με τους περιορισμούς που τέθηκαν για τη συσκευή. Με άλλα λόγια, έγινε μια προσπάθεια η συσκευή να έχει όσο το δυνατόν μικρότερες διαστάσεις. Αυτό σε συνδυασμό με το πλήθος των εξαρτημάτων της συσκευής και κατ’ επέκταση των απαιτούμενων συνδέσεων συγκέντρωσε έναν αρκετά μεγάλο όγκο αγωγών οι οποίοι έπρεπε να συνδεθούν και να προσαρμοστούν στη συσκευή χωρίς να δημιουργηθεί κάποιο σφάλμα λόγω ακούσιου βραχυκυκλώματος ή ενδεχόμενης ασυνέχειας λόγω λυγίσματος κάποιου αγωγού.

Μια άλλη δυσκολία ήταν ότι το Arduino Nano έχει περιορισμένη μνήμη Flash συγκριτικά με άλλα μοντέλα. Αυτό σε συνδυασμό με το πλήθος των εξαρτημάτων της συσκευής και την ανάγκη επικοινωνίας μεταξύ τους απαιτούσε τη δέσμευση ενός μεγάλου χώρου της μνήμης για την εγκατάσταση βιβλιοθηκών στο Arduino που ήταν απαραίτητες για την προαναφερθείσα επικοινωνία. Ενδεικτικά αναφέρεται ότι το συνολικό πρόγραμμα (μαζί με τις προαναφερθείσες βιβλιοθήκες) κατέλαβε το 85% της διαθέσιμης μνήμης, με το Arduino IDE να εμφανίζει μήνυμα «Λίγη διαθέσιμη μνήμη, μπορεί να προκύψουν προβλήματα ευστάθειας» κατά την επικύρωση του προγράμματος. Τέτοιου είδους προβλήματα δεν προέκυψαν, ωστόσο δεν κατέστη δυνατή η ανάπτυξη λειτουργίας

ανάγνωσης των περιεχομένων της κάρτας microSD από την οθόνη της συσκευής.

Μελλοντικά, μια βελτίωση της συσκευής θα μπορούσε να είναι η δημιουργία ενός τυπωμένου κυκλώματος στο οποίο θα υπάρχουν οι συνδέσεις μεταξύ των εξαρτημάτων και έτσι θα περιοριστεί σημαντικά ο αριθμός των εξωτερικών αγωγών.

Επίσης, θα μπορούσε να χρησιμοποιηθεί μία πλακέτα με μεγαλύτερη μνήμη αλλά και επιπλέον δυνατότητες. Δηλαδή, μια επέκταση του συστήματος θα μπορούσε να είναι μια πλακέτα η οποία εκτός από τη μεγαλύτερη μνήμη να διαθέτει και δυνατότητα δικτύωσης (είτε ενσύρματα είτε ασύρματα) ώστε να δημιουργηθεί μια ιστοσελίδα στην οποία να προβάλλονται ζωντανά οι μετρήσεις της συσκευής αλλά και να ρυθμίζεται η συσκευή.

Τέλος, το κουτί της συσκευής θα μπορούσε να παραχθεί από έναν εκτυπωτή 3 Διαστάσεων (3D) έτσι ώστε το συνολικό κύκλωμα να συναρμόζει με τον καταλληλότερο τρόπο και να προϋπάρχουν οι απαραίτητες υποδοχές και εσοχές που είναι απαραίτητες για τα διάφορα εξαρτήματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Temperature sensor probes, Encardio (online). Διαθέσιμο στο: https://www.encardio.com/blog/temperature-sensor-probe-types-how-it-works-applications/, Τελευταία επίσκεψη 15/11/21

[2] Thermal Sensors based on the Seebeck effect A. W. Van Herwaarden and P. M. Sarro Electrical Engineering Department, Delft University of Technology, P.o. Box 5031, 2600 GA Delft (The Netherlands) (Received June 12,1986;accepted August 28,1986)

[3] A. Dziedzic and L. J. Golonka and J. Kozlowski and B. W. Licznerski, K. Nitsch, "Thick-film resistive temperature sensors", https://doi.org/10.1088/0957-0233/8/1/011

[4] D.T. Kenny. Basic Sensor Technology. In: J.S Wilson, ed. Sensor Technology Handbook. 2005

[5] I. Toyoda and C. Denso. Capacitive humidity sensor. 2003. U.S. Patent 6,647,782

[6] B. M. Kulwicki (1991), Humidity Sensors. Journal of the American Ceramic Society, 74: 697-708. https://doi.org/10.1111/j.1151-2916.1991.tb06911.x

[7] Humidity Sensors, Electornics Hub (online). Διαθέσιμο στο: https://www.electronicshub.org/humidity-sensor-types-working-principle/, Τελευταία επίσκεψη: 10/11/21

[8] Basic usage of humidity sensors, Electronics Forum (online). Διαθέσιμο στο: https://www.electronicsforu.com/tech-zone/electronics-components/humidity-sensor-basic-usage-parameter, Τελευταία επίσκεψη: 10/11/21

[9] Arduino products, Arduino (online). Διαθέσιμο στο: https://www.arduino.cc/en/main/products, Τελευταία επίσκεψη: 20/11/21

[10] Arduino documentation, Arduino (online). Διαθέσιμο στο: https://docs.arduino.cc/, Τελευταία επίσκεψη: 6/12/21

[11] Arduino Nano, Arduino (online), Διαθέσιμο στο: https://docs.arduino.cc/hardware/nano, Τελευταία επίσκεψη 16/11/21

[12] DHT11 module tutorial, Last Minute Engineers (online), Διαθέσιμο στο: https://lastminuteengineers.com/dht11-module-arduino-tutorial/, Τελευταία επίσκεψη: 5/10/21

[13] K. P. Kuria, O. O. Robinson, M. M. Gabriel, "Monitoring Temperature and

Humidity using Arduino Nano and Module-DHT11 Sensor with Real Time DS3231 Data Logger and LCD Display" in International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol.9 Issue 12, December-2020

[14] DS18B20 tutorial, Last Minute Engineers (online), Διαθέσιμο στο: https://lastminuteengineers.com/ds18b20-arduino-tutorial/, Τελευταία επίσκεψη: 14/10/21

[15] LCD I2C tutorial, Last Minute Engineers (online), Διαθέσιμο στο: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/, Τελευταία επίσκεψη: 7/10/21

[16] SD card module tutorial, Last Minute Engineers (online), Διαθέσιμο στο: https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/, Τελευταία πρόσβαση: 10/10/21

[17] DS3231 tutorial, Last Minute Engineers (online), Διαθέσιμο στο: https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/,Τελευταία επίσκεψη: 17/10/21

[18] Basics of I2C communication, Circuit Basics (online). Διαθέσιμο στο: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/, 7/11/21

[19] F. Leens, "An introduction to I2C and SPI protocols," in IEEE Instrumentation & Measurement Magazine, vol. 12, no. 1, pp. 8-13, February 2009, doi: 10.1109/MIM.2009.4762946.

[20] Basics of SPI communication, Circuit Basics (online). Διαθέσιμο στο: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/, Τελευταία επίσκεψη: 1/12/21

[21] Guide to 1-Wire Communication, Maxim Integrated (online). Διαθέσιμο στο: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html, Τελευταία επίσκεψη: 7/11/21

[22] 1-Wire Communication through Software, Maxim Integrated (online). Διαθέσιμο στο: https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html, Τελευταία επίσκεψη: 7/11/21

[23] AVR DUDE Documentation, NonGuru (online), Διαθέσιμο στο: https://www.nongnu.org/avrdude/, Τελευταία επίσκεψη: 17/12/21

[24] C. Amariei. Arduino Development Cookbook, Pactk Publishing Ltd., 2005, ISBN 978 1-78-398294 -3

ΠΑΡΑΡΤΗΜΑ – Κώδικας συσκευής

```
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS3231.h>
#include <SPI.h>
#include <SD.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <EEPROM.h>
#define DHTPIN  6
#define DHTTYPE DHT11
#define MENU    2
#define LEFT    3
#define RIGHT   4
#define EXIT    5
#define BUZZER  7
#define MS      220
#define ONE_WIRE_BUS 8
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(DHTPIN, DHTTYPE);
DS3231 clock;
File temperatures;
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float temp, hum, tempExt, diff;
char tempstr[7];
char humstr[7];
float templimit;
        if (digitalRead(EXIT) == LOW)
        {
        delay(MS);
        break;
        }
        if (digitalRead(RIGHT) == LOW)
        {
        delay(MS);
        if (temp_templimit == 100)
        {
        continue;
        }
        temp_templimit = temp_templimit + 1;
        change = 1;
        }
        if (digitalRead(LEFT) == LOW)
        {
        delay(MS);
        if (temp_templimit == -20)
        {
        continue;
        }
        temp_templimit = temp_templimit - 1;
        change = 1;
        }
        if (change == 1) {
        flt2str(temp_templimit, tempstr);
```

```
float humlimit;
float dlimit;
char* menu[5] = {"SET TIME", "SET DATE", "SET ALARM TEMP", "SET ALARM HUM", "SET ALARM DIFF"};
byte curMenu = 0;
byte change = 0;
bool h12Flag = false;
bool pmFlag = false;
bool century = false;
float highest_ext = -99.9;
float highest_temp = -99.9;
float highest_hum = 0.0;
char timestamp[9];
char entry[27];
char result[3][21];
byte dates[3];
byte times[2];
byte t_u[] = {23, 59};
byte t_l[] = {0, 0};
byte d_u[] = {31, 12, 999};
byte d_l[] = {1, 1, 0};
byte size;
byte ind;
byte display = 0;
byte today;

void showMenu()
{

lcd.clear();
lcd.print(menu[curMenu]);
change = 0;
lcd.setCursor(7, 1);
lcd.print("      ");
lcd.setCursor(7, 1);
lcd.print(tempstr);

}
}
lcd.clear();
lcd.print(menu[curMenu]);
return;
}

void setAlarmHum()
{
delay(500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("SET HUM LIMIT");
lcd.setCursor(7, 1);
float temp_humlimit = humlimit;
flt2str(temp_humlimit, humstr);
lcd.print(humstr);
while (true)
{
if (digitalRead(MENU) == LOW)
{
delay(MS);
humlimit = temp_humlimit;
EEPROM.put(sizeof(float),
```

```
57. while (digitalRead(EXIT) != LOW)
58. {
59. delay(MS);

60. if (digitalRead(MENU) == LOW)
61. {

62. switch (curMenu)
63. {
64. case 0:
65. adjust('t', 0);
66. break;
67. case 1:
68. adjust('d', 0);
69. break;
70. case 2:
71. setAlarmTemp();
72. break;
73. case 3:
74. setAlarmHum();
75. break;
76. case 4:
77. setAlarmDiff();
78. break;
79. }
80. continue;
81. }
82. if (digitalRead(RIGHT) == LOW)
83. {
84. curMenu = (curMenu + 1) % 5;
85. lcd.clear();
```

```
    humlimit);
373.        break;
374.        }
375.        if (digitalRead(EXIT) == LOW)
376.        {
377.        delay(MS);
378.        break;
379.        }
380.        if (digitalRead(RIGHT) == LOW)
381.        {
382.        delay(MS);
383.        if (temp_humlimit == 100)
384.        {
385.        continue;
386.        }
387.        temp_humlimit =
    temp_humlimit + 1;
388.        change = 1;
389.        }
390.        if (digitalRead(LEFT) == LOW)
391.        {
392.        delay(MS);
393.        if (temp_humlimit == 0)
394.        {
395.        continue;
396.        }
397.        temp_humlimit =
    temp_humlimit - 1;
398.        change = 1;
399.        }
400.        if (change == 1) {
```

```
86. lcd.print(menu[curMenu]);
87. continue;
88. }
89. if (digitalRead(LEFT) == LOW)
90. {
91. curMenu = (curMenu - 1 + 5) % 5;
92. lcd.clear();
93. lcd.print(menu[curMenu]);
94. continue;
95. }
96. }
97. lcd.clear();
98. }
99. void adjust(char a, byte element)
100.        {
101.        byte* handle;
102.        byte* handle_u;
103.        byte* handle_l;
104.        if (a == 't')
105.        {
106.        if (element == 0)
107.        {
108.        delay(MS);
109.        lcd.clear();
110.        lcd.setCursor(2, 0);
111.        lcd.print("CURRENT TIME");
112.        ind = 6;
113.        lcd.setCursor(ind, 1);
114.        size = 2;
115.        times[0] =
    clock.getHour(h12Flag, pmFlag);
401.        flt2str(temp_humlimit,
    humstr);
402.        change = 0;
403.        lcd.setCursor(7, 1);
404.        lcd.print("      ");
405.        lcd.setCursor(7, 1);
406.        lcd.print(humstr);
407.        }
408.        }
409.        lcd.clear();
410.        lcd.print(menu[curMenu]);
411.        return;
412.        }
413.        void flt2str(float src, char*
    dest)
414.        {
415.        memset(dest, 0, 7);
416.        sprintf(dest+strlen(dest),
    "%3d", (int) src);
417.        byte n = strlen(dest);
418.        dest[n] = '.';
419.        dest[n + 1] = '0' + abs(((int)(src
    * 10)) % 10);
420.        }
421.        void thermoalarm()
422.        {
423.        lcd.clear();
424.        lcd.setCursor(5, 0);
425.        lcd.print("ALARM");
426.        lcd.setCursor(0, 1);
```

```
116.        times[1] =
        clock.getMinute();
117.        if (times[0] < 10)
118.        {
119.        lcd.print('0');
120.        }
121.        lcd.print(times[0]);
122.        lcd.print(':');
123.        if (times[1] < 10)
124.        {
125.        lcd.print('0');
126.        }
127.        lcd.print(times[1]);
128.        }
129.        handle = times;
130.        handle_u = t_u;
131.        handle_l = t_l;
132.        }
133.        else
134.        {
135.        if (element == 0)
136.        {
137.        delay(MS);
138.        dates[0] = clock.getDate();
139.        dates[1] =
        clock.getMonth(century);
140.        dates[2] = clock.getYear();
141.        size = 3;
142.        lcd.clear();
143.        lcd.setCursor(2, 0);
144.        lcd.print("CURRENT
```

```
427.        lcd.print("HIGH
        TEMPERATURE");
428.        while (digitalRead(MENU) !=
        LOW);
429.        digitalWrite(BUZZER, HIGH);
430.        delay(300);
431.        setAlarmTemp();
432.        lcd.clear();
433.        return;
434.        }

435.        void hygroalarm()
436.        {
437.        lcd.clear();
438.        lcd.setCursor(5, 0);
439.        lcd.print("ALARM");
440.        lcd.setCursor(0, 1);
441.        lcd.print("HIGH HUMIDITY");
442.        while (digitalRead(MENU) !=
        LOW);
443.        digitalWrite(BUZZER, HIGH);
444.        delay(300);
445.        setAlarmHum();
446.        lcd.clear();
447.        return;
448.        }

449.        void diffalarm()
450.        {
451.        lcd.clear();
452.        lcd.setCursor(5, 0);
453.        lcd.print("ALARM");
```

```
DATE");
145.        ind = 4;
146.        lcd.setCursor(ind, 1);
147.        if (dates[0] < 10)
148.        {
149.        lcd.print('0');
150.        }
151.        lcd.print(dates[0]);
152.        lcd.print('-');
153.        if (dates[1] < 10)
154.        {
155.        lcd.print('0');
156.        }
157.        lcd.print(dates[1]);
158.        lcd.print('-');
159.        if (dates[2] < 10)
160.        {
161.        lcd.print('0');
162.        }
163.        lcd.print(dates[2]);
164.        byte day[] = {31, 29, 31, 30,
    31, 30, 31, 31, 30, 31, 30, 31};
165.        d_u[0] = day[dates[1] - 1];
166.        }
167.        handle = dates;
168.        handle_u = d_u;
169.        handle_l = d_l;
170.        }
171.        while(true)
172.        {
173.        if (digitalRead(EXIT) ==
454.        lcd.setCursor(0, 1);
455.        lcd.print("TEMP DIFFERENCE");
456.        while (digitalRead(MENU) !=
    LOW);
457.        digitalWrite(BUZZER, HIGH);
458.        delay(300);
459.        setAlarmDiff();
460.        lcd.clear();
461.        return;
462.        }

463.        void setup()
464.        {
465.        Serial.begin(9600);
466.        pinMode(MENU,
    INPUT_PULLUP);
467.        pinMode(LEFT,
    INPUT_PULLUP);
468.        pinMode(RIGHT,
    INPUT_PULLUP);
469.        pinMode(EXIT,
    INPUT_PULLUP);
470.        pinMode(BUZZER, OUTPUT);
471.        digitalWrite(BUZZER, HIGH);
472.        dht.begin();
473.        lcd.init();
474.        lcd.backlight();
475.        clock.setClockMode(false);
476.        sensors.begin();
477.        if(clock.getHour(h12Flag,
    pmFlag) < 0 || clock.getHour(h12Flag,
    pmFlag) > 23)
478.        {
479.        clock.setYear(21);
```

```
    LOW)
174.        {
175.        delay(MS);
176.        lcd.clear();
177.        lcd.print(menu[curMenu]);
178.        return;
179.        }
180.        if (digitalRead(RIGHT) ==
    LOW)
181.        {
182.        delay(MS);
183.        if (*(handle + element) ==
    *(handle_u + element))
184.        {
185.        continue;
186.        }
187.        *(handle + element) += 1;
188.        byte temp = *(handle +
    element);
189.        lcd.setCursor(ind, 1);
190.        if (*(handle + element) <
    10)
191.        {
192.        lcd.print('0');
193.        }
194.        lcd.print(*(handle +
    element));
195.        }
196.        if (digitalRead(LEFT) ==
    LOW)
197.        {
198.        delay(MS);
199.        if (*(handle + element) ==
480.        clock.setMonth(10);
481.        clock.setDate(12);
482.        clock.setHour(0);
483.        clock.setMinute(0);
484.        clock.setSecond(1);
485.        }
486.        memset(timestamp, 0, 9);
487.        today = clock.getDate();
488.        SD.begin(10);
489.        if (!SD.exists("temp.txt"))
490.        {
491.        temperatures =
    SD.open("temp.txt", FILE_WRITE);
492.        temperatures.close();
493.        }
494.        byte address = 0;
495.        EEPROM.get(address,
    templimit);
496.        address += sizeof(float);
497.        EEPROM.get(address,
    humlimit);
498.        address += sizeof(float);
499.        EEPROM.get(address, dlimit);
500.        }
501.        void loop()
502.        {
503.        if (today != clock.getDate())
504.        {
505.        sprintf(timestamp, "%02d
    -%02d-%02d", today,
    clock.getMonth(century),
    clock.getYear());
```

```
    *(handle_l + element))
200.        {
201.        continue;
202.        }
203.        *(handle + element) -= 1;
204.        lcd.setCursor(ind, 1);
205.        if (*(handle + element) <
    10)
206.        {
207.        lcd.print('0');
208.        }
209.        lcd.print(*(handle +
    element));
210.        byte temp = *(handle +
    element);
211.        }
212.        if (digitalRead(MENU) ==
    LOW)
213.        {
214.        delay(MS);
215.        if (element == size - 1)
216.        {
217.        if (a == 't')
218.        {
219.        clock.setHour(*(handle));
220.        clock.setMinute(*(handle +
    1));
221.        }
222.        else
223.        {
224.        clock.setDate(dates[0]);
225.        clock.setMonth(dates[1]);
506.        memset(entry, 0, 27);
507.        sprintf(entry, timestamp);
508.        entry[strlen(entry)] = 'T';
509.        flt2str(highest_temp, tempstr);
510.        sprintf(entry + strlen(entry),
    tempstr);
511.        entry[strlen(entry)] = 'E';
512.        flt2str(highest_ext, tempstr);
513.        sprintf(entry + strlen(entry),
    tempstr);
514.        entry[strlen(entry)] = 'H';
515.        flt2str(highest_hum, humstr);
516.        sprintf(entry + strlen(entry),
    humstr);
517.        Serial.print(entry);
518.        temperatures =
    SD.open("temp.txt", FILE_WRITE);
519.        temperatures.println(entry);
520.        temperatures.close();
521.        highest_temp = -99.9;
522.        highest_ext = -99.9;
523.        highest_hum = 0.0;
524.        today = clock.getDate();
525.        delay(1000);
526.        }
527.        sensors.requestTemperatures(
    );
528.        temp = dht.readTemperature();
529.        hum = dht.readHumidity();
530.        tempExt =
    sensors.getTempCByIndex(0);
531.        diff = temp - tempExt;
532.        lcd.setCursor(0, 0);
```

```
226.        clock.setYear(dates[2]);
227.        today = dates[0];
228.        }
229.        lcd.clear();
230.        lcd.print(menu[curMenu]);
231.        return;
232.        }
233.        else
234.        {
235.        ind = ind + 3;
236.        adjust(a, element + 1);
237.        return;
238.        }
239.        }
240.        }
241.        }

242.        void setAlarmDiff()
243.        {
244.        delay(500);
245.        lcd.clear();
246.        lcd.setCursor(0, 0);
247.        lcd.print("SET DIFF LIMIT");
248.        lcd.setCursor(7, 1);
249.        float temp_dlimit = dlimit;
250.        flt2str(temp_dlimit,
    tempstr);
251.        lcd.print(tempstr);
252.        while (true)
253.        {
254.        if (digitalRead(MENU) ==
```

```
533.        if(clock.getDate() < 10)
534.        {
535.        lcd.print("0");
536.        }
537.        lcd.print(clock.getDate());
538.        lcd.print("-");
539.        if(clock.getMonth(century) <
    10)
540.        {
541.        lcd.print("0");
542.        }
543.        lcd.print(clock.getMonth(centu
    ry));
544.        lcd.print("-");
545.        if(clock.getYear() < 10)
546.        {
547.        lcd.print("0");
548.        }
549.        lcd.print(clock.getYear());
550.        lcd.print("    ");
551.        if(clock.getHour(h12Flag,
    pmFlag) < 10)
552.        {
553.        lcd.print("0");
554.        }
555.        lcd.print(clock.getHour(h12Fla
    g, pmFlag));
556.        lcd.print(":");
557.        if(clock.getMinute() < 10)
558.        {
559.        lcd.print("0");
560.        }
```

```
    LOW)
255.        {
256.        delay(MS);
257.        dlimit = temp_dlimit;
258.        EEPROM.put(2*
    sizeof(float), dlimit);
259.        break;
260.        }
261.        if (digitalRead(EXIT) ==
    LOW)
262.        {
263.        delay(MS);
264.        break;
265.        }
266.        if (digitalRead(RIGHT) ==
    LOW)
267.        {
268.        delay(MS);
269.        if (temp_dlimit == 50)
270.        {
271.        continue;
272.        }
273.        temp_dlimit = temp_dlimit
    + 1;
274.        change = 1;
275.        }
276.        if (digitalRead(LEFT) ==
    LOW)
277.        {
278.        delay(MS);
279.        if (temp_dlimit == -20)
280.        {
```

```
561.        lcd.print(clock.getMinute());
562.        lcd.setCursor(0, 1);
563.        lcd.print("              ");
564.        lcd.setCursor(0, 1);
565.        switch(display)
566.        {
567.        case 0:
568.        flt2str(temp, tempstr);
569.        flt2str(hum, humstr);
570.        lcd.print("T");
571.        lcd.print(tempstr);
572.        lcd.print("C");
573.        lcd.setCursor(8, 1);
574.        lcd.print("H");
575.        lcd.print(humstr);
576.        lcd.print("%");
577.        break;
578.        case 1:
579.        flt2str(tempExt, tempstr);
580.        lcd.print("2nd Pr: ");
581.        lcd.print(tempstr);
582.        lcd.print("C");
583.        break;
584.        case 2:
585.        flt2str(diff, tempstr);
586.        lcd.print("T-Dif1,2: ");
587.        lcd.print(tempstr);
588.        lcd.print("C");
589.        Serial.println(diff);
590.        break;
```

```
281.            continue;
282.            }
283.            temp_dlimit = temp_dlimit -
    1;
284.            change = 1;
285.            }
286.            if (change == 1) {
287.            flt2str(temp_dlimit,
    tempstr);
288.            change = 0;
289.            lcd.setCursor(7, 1);
290.            lcd.print("      ");
291.            lcd.setCursor(7, 1);
292.            lcd.print(tempstr);

293.            }
294.            }
295.            lcd.clear();
296.            lcd.print(menu[curMenu]);
297.            return;
298.            }

299.            void setAlarmTemp()
300.            {
301.            delay(500);
302.            lcd.clear();
303.            lcd.setCursor(0, 0);
304.            lcd.print("SET TEMP
    LIMIT");
305.            lcd.setCursor(7, 1);
306.            float temp_templimit =
    templimit;
307.            flt2str(temp_templimit,
```

```
591.            }

592.            delay(1500);
593.            if (tempExt > highest_ext)
594.            {
595.            highest_ext = tempExt;
596.            }
597.            if (temp > highest_temp)
598.            {
599.            highest_temp = temp;
600.            }
601.            if (hum > highest_hum)
602.            {
603.            digitalWrite(BUZZER, HIGH);
                highest_hum = hum;
604.            }
605.            if (temp > templimit)
606.            {
607.            digitalWrite(BUZZER, HIGH);
                thermoalarm();
608.            }

609.            if (hum > humlimit)
610.            {
611.            hygroalarm();
612.            }
613.            if (diff > dlimit)
614.            {
615.            digitalWrite(BUZZER, HIGH);
        diffalarm();
616.            }
617.            if (digitalRead(MENU) == LOW)
```

```
tempstr);
308.        lcd.print(tempstr);
309.        while (true)
310.        {
311.        if (digitalRead(MENU) ==
LOW)
312.        {
313.        delay(MS);
314.        templimit =
temp_templimit;
315.        EEPROM.put(0, templimit);
316.        break;
317.        }
618.        {
619.        showMenu();
620.        }
621.        display = (display + 1) % 3;
622.        }
```