Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τομέας Ηλεκτρονικής και Υπολογιστών

ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΙΑΚΑ

Προαιρετική Εργασία 2019

Έξυπνο Θερμόμετρο με Arduino

GROUP B - ΟΜΑΔΑ 4

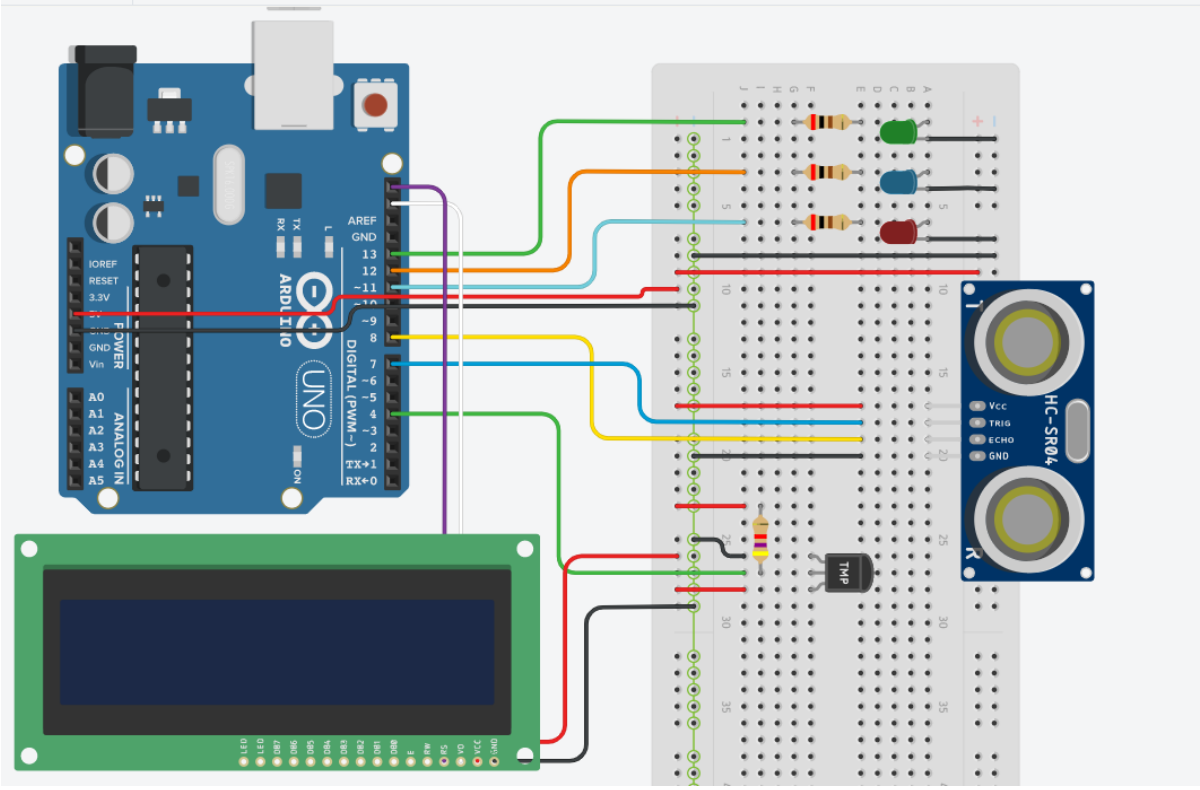
Λέτρος Κωνσταντίνος (ΑΕΜ 8851)

Μανουσαρίδης Ιωάννης (ΑΕΜ 8855)

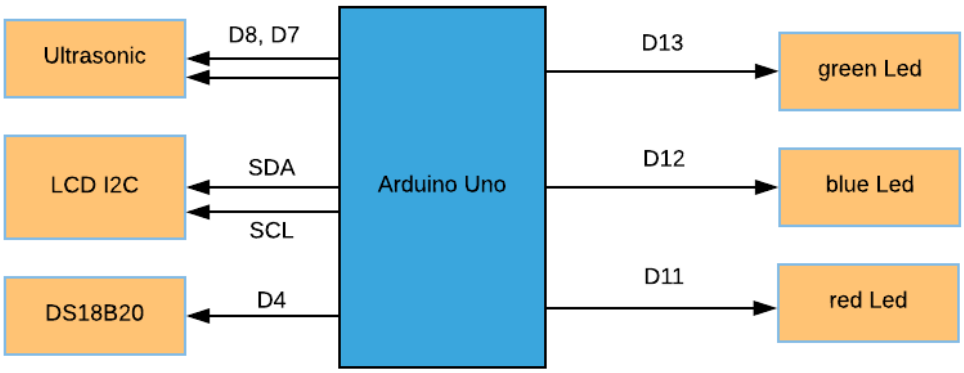
Περιεχόμενα

1. Κύκλωμα εφαρμογής

1.1 Hardware design



1.2 Diagram design



** Αντί για άσπρο LED χρησιμοποιήθηκε πράσινο.

2. Κώδικας
2.1 Ψευδοκώδικας

Include libraries

Initialize Timer1 and variables

Void_loop

If index < 24 // θέλουμε 24 μετρήσεις θερμοκρασίας ανά 2 λεπτά

    get_temperature and save it

    check if a LED needs to turn on according to the temperature value

    for 10 loop reps // 10 επαναλήψεις * 500ms = 5 sec

        sleep for 450ms

        check if distance from sonar less than 10cm // awake for 50ms

        if true, display on LCD the last temperature and the last mean temperature

        index = index + 1

else_if index == 24

    calculate the mean value and print it on the screen

    index = 0 // clear for the next 2 minutes

end_if

end_void_loop

2.2 Ανάλυση Κώδικα

Ο κώδικας ακολουθεί τη λογική ροή που αναφέρεται στην παράγραφο 2.1 με τον ψευδοκώδικα. Αρχικά γίνονται όλες οι απαραίτητες αρχικοποιήσεις καθώς και η συμπερίληψη των βιβλιοθηκών που θα χρειαστούν.

Κατόπιν, στη void_setup() τίθενται αρχικά τα pin των LED σε λογικό μηδέν, ενεργοποιείται η LCD οθόνη (χωρίς κάποια ένδειξη) και τέλος αρχικοποιείται ο Timer1.

Για την ορθή λειτουργεία του Timer1 ενεργοποιήθηκε η υπερχείλιση μέσω της εντολής TIMSK1=1<<TOIE1, ενώ η τιμή του Prescaler που επιλέχθηκε είναι η 1024 ώστε ο κάθε κύκλος ρολογιού να διαρκεί 64μsec. Ο Timer1 κάνει υπερχείλιση μόλις φτάσει στην τιμή 65535. Στην αρχή, έχει την τιμή 58504 και σε κάθε κύκλο ρολογιού αυξάνει κατά ένα. Οι κύκλοι που θα ολοκληρώσει μέχρι να γίνει υπερχείλιση είναι : 65535 – 58504 = 7031 κύκλοι. Επομένως για να γίνει υπερχείλιση του χρονιστή 1 θα περάσουν 7031 cycles · 64 μsec/cycle = 450 msec. Η λειτουργία του Timer1 στο πρόγραμμα αφορά κυρίως την αφύπνιση του Arduino, κάτι που θα αναλυθεί παρακάτω εκτενέστερα.

Στη συνέχεια, ο κώδικας μεταβαίνει στη void_loop(), όπου διακρίνονται δύο λειτουργίες ανάλογα με το αν ο μετρητής index έχει τιμή μικρότερη ή ίση με 24.

Αν η τιμή της είναι μικρότερη από 24, τότε το πρόγραμμα ακολουθεί την πρώτη λειτουργία.

Συγκεκριμένα, μέσω της getTemp() λαμβάνεται η τρέχουσα τιμή της θερμοκρασίας σε βαθμούς ^{o}C . Αν η θερμοκρασία που μετράει ο αισθητήρας είναι μεγαλύτερη από 26^{o}C, ανάβουν το κόκκινο (υπερθέρμανση) και το πράσινο LED (εκκίνηση ανεμιστήρα), ενώ αν είναι μικρότερη των 20^{o}C, ανάβει το μπλε LED (χαμηλή θερμοκρασία). Οι παραπάνω τιμές επιλέχθηκαν για ευκολότερη επίδειξη και έλεγχο λειτουργίας. Επιπλέον, η ληφθείσα τιμή αποθηκεύεται στον πίνακα tempArr[].

Έπειτα, αυξάνονται η τιμή του index και η τιμή του powerOffLCDcounter κατά ένα. Η πρώτη χρησιμοποιείται για την απαρίθμηση των 24 μετρήσεων που πρέπει να ληφθούν εντός 2 λεπτών, ενώ η δεύτερη χρησιμοποιείται για το σβήσιμο της LCD οθόνης όταν παρέλθει το απαιτούμενο χρονικό διάστημα. Η LCD οθόνη σβήνει έπειτα από 10 δευτερόλεπτα, ανεξάρτητα από το αν άνοιξε εξαιτίας του ultrasonic, είτε λόγω του ότι πέρασαν 2 λεπτά και παρουσιάστηκε ο νέος μέσος όρος των τελευταίων 24 μετρήσεων.

Στη συνέχεια ακολουθεί μία επαναληπτική διαδικασία με 10 επαναλήψεις. Η κάθε επανάληψη διαρκεί περίπου 5 δευτερόλεπτα, το οποίο είναι το χρονικό διάστημα που απαιτείται για τη λήψη της επόμενης τιμής της θερμοκρασίας. Μέσα σε αυτήν την επαναληπτική διαδικασία, το Arduino εισέρχεται σε λειτουργία ύπνου και συγκεκριμένα στη λειτουργία IDLE. Σε αυτή τη λειτουργία, ο μικροεπεξεργαστής χρησιμοποιεί κανονικά τους Timers, την επικοινωνία μέσω UART, SPI, I2C και τον ADC. Ωστόσο παύει να χρησιμοποιεί τα CPU και Flash clock για να εξοικονομήσει ενέργεια.

Στο πρόγραμμα, ο μικροεπεξεργαστής εισέρχεται στη λειτουργία IDLE μέσω της συνάρτησης enterSleep(). Επιπλέον, για ακόμα μεγαλύτερη εξοικονόμηση ενέργειας απενεργοποιούνται οι επικοινωνίες μέσω UART, SPI, I2C, ο ADC και οι Timers 0 και 2 χειροκίνητα. Επομένως, ο μικροεπεξεργαστής αναμένει ένα σήμα διακοπής από τον Timer1 για να εξέλθει από την κατάσταση IDLE.

Όπως αναφέρθηκε προηγουμένως, μόλις περάσουν 450 msec, θα γίνει υπερχείλιση του Timer1. Αυτός με τη σειρά του θα προκαλέσει μία διακοπή, η οποία έχει ως αποτέλεσμα να εξέλθει το Arduino από τη λειτουργία IDLE στην οποία βρισκόταν, και να συνεχίσει κανονικά τη λειτουργία του κύριου προγράμματος.

Επιπλέον, να τονιστεί ότι μόλις γίνει η διακοπή εξαιτίας της υπερχείλισης του Timer1 και κληθεί η ISR (Interrupt Service Routine), ο Timer1 αρχικοποιείται στην τιμή 58504 εκ νέου έτσι ώστε να καθίσταται δυνατή η επαναλαμβανόμενη “κοίμηση” του Arduino για 450 msec και η “αφύπνισή” του.

Συνεχίζοντας, μόλις το Arduino εξέλθει από την κατάσταση IDLE, συνεχίζει κανονικά τη λειτουργία του και ελέγχει την απόσταση με το HC-SR04. Αν αυτή είναι μικρότερη των 10cm, καλείται η showTemperatureNow() για να προβάλλει στην LCD οθόνη την τελευταία μέση τιμή της θερμοκρασίας και την τελευταία θερμοκρασία που καταγράφηκε. Μετά από αυτόν τον έλεγχο, ξεκινάει η επόμενη επανάληψη.

Συνοψίζοντας, σε κάθε επανάληψη ο μικροεπεξεργαστής Arduino εισέρχεται σε κατάσταση χαμηλής κατανάλωσης ενέργειας για περίπου 450 msec και ξυπνάει λόγω της διακοπής που προκαλείται από την υπερχείλιση του Timer1. Κατόπιν, ελέγχει την απόσταση μέσω του ultrasonic, μία διαδικασία που διαρκεί περίπου 50 msec. Συνεπώς, κάθε επανάληψη έχει διάρκεια 450 + 50 = 500 msec. Επειδή έχουμε 10 επαναλήψεις, η συνολική καθυστέρηση θα

είναι 5 δευτερόλεπτα. Με το πέρας αυτών, η πρώτη λειτουργία τερματίζει και το πρόγραμμα επανέρχεται στην αρχή της void loop().

Μόλις η τιμή του index γίνει 24, το πρόγραμμα ακολουθεί τη δεύτερη λειτουργία. Σε αυτή τη λειτουργία, υπολογίζεται η μέση τιμή των θερμοκρασιών που λήφθηκαν τα τελευταία 2 λεπτά και είναι αποθηκευμένες στον πίνακα tempArr[]. Τέλος, καλείται η printLCD_every2min() για να προβάλλει στην LCD οθόνη τη μέση αυτή τιμή που υπολογίστηκε. Οι μεταβλητές index και powerOffLCDcounter γίνονται ξανά μηδέν και η όλη διαδικασία του προγράμματος ξεκινά από την αρχή.

2.3 Προβλήματα κατά την ανάπτυξη του κώδικα και τη σύνδεση των περιφερειακών

Αρχικά πρέπει να σημειωθεί ότι η προαιρετική υλοποίηση με την αποστολή email δεν πραγματοποιήθηκε, ενώ χρησιμοποιήθηκε η οθόνη LCD με το ολοκληρωμένο I2C στην πίσω όψη. Μία δυσκολία που αντιμετωπίσαμε αφορούσε την σωστή ρύθμιση του Timer1 ώστε το Arduino να εισέρχεται και εξέρχεται από τη λειτουργία ύπνου στον επιθυμητό χρόνο. Επίσης, αφιερώσαμε αρκετό χρόνο στο να βρούμε έναν τρόπο να λαμβάνουμε μετρήσεις με το ultrasonic χωρίς την αφύπνιση του Arduino, πράγμα που δεν καταφέραμε, επομένως στραφήκαμε στη λύση των λήψεων δειγμάτων από τον αισθητήρα ultrasonic σε τακτά χρονικά διαστήματα. Τέλος, αφιερώσαμε αρκετό χρόνο στο να καταφέρουμε να πετύχουμε τη βέλτιστη δυνατή εξοικονόμηση ενέργειας (απενεργοποίηση όλων των μη χρησιμοποιούμενων περιφερειακών) προσπαθώντας να προσεγγίσουμε περισσότερο ένα πραγματικό σύστημα έξυπνου θερμομέτρου.

2.4 Μετρήσεις και ανάλυση του συστήματος

Παρακάτω παρατίθενται δύο σύνδεσμοι, που οδηγούν σε video demo της εργασίας, με το κύκλωμα εν λειτουργία σε διάφορες καταστάσεις.

Παράδειγμα 1

Στο παράδειγμα αυτό αρχικά η απόσταση που μετράει το Ultrasonic γίνεται μικρότερη των 10cm οπότε εμφανίζεται η τελευταία τιμή της θερμοκρασίας που μετρήθηκε και η τελευταία μέση τιμή που υπολογίστηκε. Στη συνέχεια η θερμοκρασία λαμβάνει τιμή πάνω από 26^{o}C και ανάβουν τα δύο LED, πράσινο (ανεμιστήρας) και κόκκινο. Επιπλέον, η τιμή της θερμοκρασίας είναι πράγματι πάνω από 26 βαθμούς, γεγονός που επιβεβαιώνεται μόλις τοποθετήσουμε το χέρι μας κοντά στο sonar και παρατηρήσουμε το αποτέλεσμα στην οθόνη LCD.

https://drive.google.com/file/d/1lKqoEmVZKzf-UKLgz7fsJpBH1w_DY-Cr/view?usp=sharing

Παράδειγμα 2

Το παράδειγμα 2 δείχνει τη λειτουργία του κυκλώματος σε πλήρη μορφή. Εδώ φαίνεται ξεκάθαρα η λειτουργία του αισθητήρα HC-SR04, του πως ενεργοποιούνται τα LED με βάση τη θερμοκρασία καθώς και η απεικόνιση της μέσης τιμής μόλις περάσουν τα 2 λεπτά.

https://drive.google.com/file/d/1KP14Yv33usrxn_0RGIuUNsrDFcpkZt9P/view?usp=sharing

2.5 GitHub

Ο κώδικας του κυκλώματος βρίσκεται στο παρακάτω repo στο GitHub:

https://github.com/imanousar/AUTh/tree/master/Microprocessors%20and%20peripherals/smartThermometer%20Arduino