ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδίαση και κατασκευή σχολικού κουδουνιού»

Του φοιτητή
Ιωάννη Παπαγιαννακέλη
Αρ. Μητρώου: 518234

Επιβλέπων
Άγγελος Γιακουμής
Επίκουρος Καθηγητής

Θεσσαλονίκη Ιανουάριος 2022

Τίτλος Δ.Ε. Σχεδίαση και κατασκευή σχολικού κουδουνιού
Κωδικός Δ.Ε. 21188
Ονοματεπώνυμο φοιτητή: Ιωάννης Παπαγιαννακέλης
Ονοματεπώνυμο επιβλέποντα εισηγητή: Άγγελος Γιακουμής
Ημερομηνία ανάληψης Δ.Ε. 15-03-2021
Ημερομηνία περάτωσης Δ.Ε. 31-01-2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Στη γυναίκα μου Βάσω

Πρόλογος

Με την περάτωση των σπουδών μου στο τμήμα Ηλεκτρονικής, ένα τμήμα με υψηλό επίπεδο σπουδών και κυρίως πρακτικό προσανατολισμό των αντικειμένων που διδάσκονται, έφτασε η στιγμή να συνδεθούν οι προηγούμενες γνώσεις μου σε προγραμματισμό, με τον έλεγχο πραγματικών κυκλωμάτων με συνδυασμό αναλογικών και ψηφιακών υποσυστημάτων. Αυτό το σκοπό εξυπηρετεί η ανάληψη της συγκεκριμένης πτυχιακής εργασίας. Η διασύνδεση ενός συστήματος ισχύος, μέσω ενός αναλογικού συστήματος, που ελέγχεται από την έξοδο ενός ψηφιακού συστήματος , είχε σαν αποτέλεσμα να αντιμετωπίσω μια σειρά προβλημάτων - προκλήσεων με επιτυχία, όπως φάνηκε από το τελικό αποτέλεσμα. Η ενασχόληση με ένα δημοφιλές ενσωματωμένο σύστημα, η σχεδίαση ενός κυκλώματος χρονισμού και ο έλεγχος ενός εξαρτήματος ισχύος, βοήθησαν στην ολοκληρωμένη θεώρηση των σύγχρονων ηλεκτρονικών συστημάτων αυτοματισμού. Επιπλέον, ο σκοπός ήταν να αντιμετωπιστεί ένα πραγματικό πρόβλημα, και γι’ αυτό δόθηκε ιδιαίτερο βάρος στην αξιοπιστία του τελικού συστήματος, όπως και στην ευχρηστία από τον τελικό, μη ειδικό, χρήστη.

Περίληψη

Η παρούσα εργασία έχει ως στόχο την κατασκευή ενός αξιόπιστου συστήματος σχολικού κουδουνιού, με τη βοήθεια λιγότερο αξιόπιστων εξαρτημάτων όπως ένα χαμηλού κόστους ενσωματωμένο σύστημα. Με διάφορες απλές τεχνικές ανάνηψης από προβληματικές καταστάσεις, εξασφαλίζουμε την απρόσκοπτη λειτουργία του συστήματος. Η εργασία οργανώνεται ως εξής: Αφού περιγράφεται η υπάρχουσα κατάσταση σχετικά με τα κουδούνια σχολείων στο κεφάλαιο 1, επεξηγείται / αναλύεται το κύκλωμα διασφάλισης αξιόπιστης λειτουργίας (watchdog) στο κεφάλαιο 2 με ιδιαίτερη αναφορά στην αξιόπιστη λειτουργία των υλικών του. Πραγματοποιούνται έλεγχοι καλής λειτουργίας. Στο κεφάλαιο 3, εξηγούνται οι λόγοι επιλογής του ενσωματωμένου συστήματος, το οποίο προτιμήθηκε για το σύστημα που υλοποιείται. Κατόπιν, περιγράφεται η διαδικασία κατασκευής (κεφάλαιο 4), εγκατάστασης και απομακρυσμένης κυρίως υποστήριξης (κεφάλαιο 5). Προτείνονται αλλαγές για να γίνει το τελικό σύστημα πιο χρήσιμο και ελκυστικό για τον τελικό χρήστη (κεφάλαιο 6). Ιδιαίτερη έμφαση δίνεται στην ευχρηστία του συστήματος και στα μηνύματα τα οποία εμφανίζονται στην οθόνη, έτσι ώστε να μην απογοητεύει ακόμη και μη σχετικούς με τέτοια συστήματα χρήστες.

«Design and construction of a school bell»

Ioannis Papagiannakelis

Abstract

The present work aims to build a reliable school bell system, with the help of less reliable components such as a low cost embedded system. With various simple techniques of recovery from problematic situations, we ensure the smooth operation of the system. The work is organized as follows: After describing the current situation regarding school bells in chapter 1, in chapter 2, the reliable operation assurance circuit (watchdog) is explained / analyzed with special reference to the reliable operation of its materials. Performance checks are performed. Chapter 3 explains the reasons for choosing the embedded system to use with the system being implemented. The process of construction (Chapter 4), installation and remote - mainly - support (Chapter 5) is then described. Upgrades are proposed to make the final system more useful and attractive to the end user (Chapter 6). Particular emphasis is placed on the usability of the system and the messages displayed on the screen, so as not to disappoint users non-familiar to such systems.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους εκείνους που μ βοήθησαν με την άμεση ή έμμεση συμβολή τους στην αποπεράτωση αυτής της πτυχιακής εργασίας.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, Επίκουρο Καθηγητή του τμήματος κ. Άγγελο Γιακουμή για την εμπιστοσύνη την οποία μου έδειξε αναθέτοντάς μου τη συγκεκριμένη εργασία, όπως και για όλη τη συμπαράσταση και τη βοήθειά του. Επιπλέον ευχαριστώ όλους τους φίλους και συναδέλφους για τη βοήθεια που μου προσέφεραν σε διάφορα στάδια της υλοποίησης. Ενδεικτικά αναφέρω τους Μιχάλη Μανούση, που με βοήθησε πολύ για την αρχική ιδέα της εργασίας, την Κατερίνα Κίτσιου και τον Βαγγέλη Κρικέλη.

Θα ήταν παράλειψη να μην ευχαριστήσω τη σύζυγό μου Βάσω Κρικέλη, για την υπομονή της σε όλη τη διάρκεια των σπουδών μου και για τη στάση της που μου έδινε κουράγιο και δύναμη να συνεχίσω.

Περιεχόμενα

Ευρετήριο Εικόνων

Κεφάλαιο 1ο: Οι Σχολικές Μονάδες σήμερα

1.1 Περιγραφή προβλήματος

Ο διδακτικός χρόνος, σε όλα τα εκπαιδευτικά ιδρύματα, χωρίζεται σε συνεχείς περιόδους κατά τις οποίες πραγματοποιείται η διδασκαλία, τις διδακτικές ώρες. Οι περίοδοι αυτές ποικίλουν, ανάλογα με την ηλικία των διδασκομένων και την εκπαιδευτική μέθοδο που χρησιμοποιείται στη διδασκαλία. Σε περιπτώσεις ενηλίκων λόγω ωριμότητας σχετικής με την ηλικία, η διδακτική ώρα είναι 45 λεπτά και σε κάποιες περιπτώσεις 90'. Σε περιπτώσεις ανηλίκων μαθητών η διδακτική ώρα δεν πρέπει να ξεπερνά τα 45'. Σε μικρές ηλικίες η προσοχή των ακροατών διασπάται σχετικά γρήγορα, σε 15-20', και τα παιδιά κουράζονται πολύ νωρίτερα από μεγαλύτερες ηλικίες μαθητών ή ενηλίκων εκπαιδευομένων. Ανάμεσα στις διδακτικές ώρες υπάρχουν τα διαλείμματα.

Τα παραπάνω χρονικά διαστήματα αναφέρονται στην κατά μέτωπο διδασκαλία / εισήγηση, η οποία είναι η πιο κουραστική για τον εκπαιδευόμενο μέθοδος διδασκαλίας. Γι' αυτόν το λόγο ο δάσκαλος-εισηγητής προσπαθεί να κινήσει το ενδιαφέρον του ακροατή και με άλλες μεθόδους. Σύμφωνα με τον Dale [1] ο ακροατής «μαθαίνει» το 20% από αυτά που ακούει, το 50% από αυτά που βλέπει και ακούει και το 90% από αυτά τα οποία δημιουργεί ό ίδιος. Επομένως, είναι στο χέρι του εκπαιδευτικού να εμπλουτίσει τη διδασκαλία του με οπτικά - τουλάχιστον - ερεθίσματα ώστε να παρακινήσει τους εκπαιδευομένους. Ο εμπλουτισμός αυτός είναι αρκετά εύκολος για τα περισσότερα αντικείμενα μάθησης, ακόμη και για τα απολύτως θεωρητικά, με την προβολή πινάκων, σχημάτων, εννοιολογικών χαρτών και άλλων εικόνων που απευθύνονται συναισθηματικά στους ακροατές και απαιτούν τη χρήση λογισμικού παρουσίασης και συστήματος προβολής.

Στην περίπτωση των πρακτικών μαθημάτων και επιστημών, για να εκμεταλλευτούμε τα ευρήματα του Dale, έχουν καθιερωθεί οι πρακτικές ενασχολήσεις με συγκεκριμένα αντικείμενα. Τα μαθήματα αυτά, ευρέως γνωστά ως εργαστηριακά μαθήματα, εναλλάσσουν θεωρητική διδασκαλία και πρακτική ενασχόληση των εκπαιδευομένων και περιλαμβάνονται όχι μόνο στην εκπαίδευση ενηλίκων, π.χ. φοιτητών σε εφαρμοσμένες επιστήμες, αλλά και σε μαθητές μικρής ηλικίας σε μαθήματα φυσικών επιστημών και πληροφορικής. Οι διδακτικές αυτές ώρες μπορούν να είναι μεγαλύτερης διάρκειας από τις θεωρητικές, λόγω του αυξημένου ενδιαφέροντος το οποίο προκαλεί η πρακτική ενασχόληση στον εκπαιδευόμενο και μπορούν να καλύψουν δύο έως τρεις κλασσικές διδακτικές ώρες των 45' χωρίς διάλειμμα. Στην περίπτωση από την άλλη πλευρά, πολύ μικρής ηλικίας μαθητών, λόγω του πολύ μικρού χρόνου δυνατότητας συγκέντρωσης σε θεωρητική διδασκαλία, ο εκπαιδευτικός είναι υποχρεωμένος να εμπλουτίσει την εισήγησή του όχι μόνο με προβολή εικόνων αλλά και με video όπως και με άλλες μεθόδους (ομαδική εργασία, παιχνίδια ρόλων και άλλους τρόπους για να φαίνεται σαν παιχνίδι η μάθηση), που κάνουν το μάθημα πιο ενδιαφέρον και ανεκτό από τους μαθητές. Οι ίδιες οι διδακτικές ώρες προσαρμόζονται ανάλογα με την περίπτωση, π.χ. οι τελευταίες διδακτικές ώρες στο δημοτικό και το Γυμνάσιο όταν οι μαθητές είναι κουρασμένοι, δεν είναι 45λεπτες αλλά μπορούν να γίνουν έως και 35λεπτες, όπως και σε δομές απογευματινών ή βραδινών μαθημάτων σε εργαζόμενους ενήλικες που οι διδακτικές ώρες είναι μικρότερης διάρκειας.

Σε κάθε περίπτωση οι διδακτικές ώρες είναι κεντρικά (με αποφάσεις του Υπουργείου Παιδείας) καθορισμένης διάρκειας, όπως και οι ώρες έναρξης και λήξης του προγράμματος. Στην τριτοβάθμια εκπαίδευση υπάρχει και καθορισμός ελάχιστου χρόνου διδασκαλίας μαθημάτων για να θεωρηθεί ένα μάθημα επαρκώς διδαχθέν (διδακτικές εβδομάδες). Για τη διευκόλυνση εκπαιδευτικών και

εκπαιδευομένων χρησιμοποιείται, τουλάχιστον στην πρωτοβάθμια και τη δευτεροβάθμια εκπαίδευση, ηχητική ειδοποίηση. Το γνωστό σε όλους από τα παιδικά μας χρόνια σχολικό κουδούνι.

Σήμερα τα περισσότερα σχολεία χρησιμοποιούν έτοιμα ενσωματωμένα συστήματα για σχολικά κουδούνια. Οι διατάξεις αυτές παίζουν το ρόλο απλού χρονοδιακόπτη / ξυπνητηριού με πολλαπλά χρονικά σημεία ενεργοποίησης. Συνδέονται με κλασσικά ηλεκτρομαγνητικά κουδούνια, τα οποία ενεργοποιούνται σε καθορισμένες από πριν χρονικές στιγμές, τις ώρες έναρξης και λήξης των διδακτικών ωρών. Τα συστήματα αυτά, είναι αρκετά αξιόπιστα για σταθερή λειτουργία, για την κατάσταση δηλαδή όπου τα σημεία ενεργοποίησης είναι προκαθορισμένα σύμφωνα με τις ισχύουσες Υπουργικές αποφάσεις. Διαθέτουν εσωτερικό ρολόι για να μη χάνεται η ώρα και η ημερομηνία μετά από διακοπές τροφοδοσίας. Ελέγχονται από μικροελεγκτή για να μην καταναλώνουν πολλή ενέργεια. Η ρύθμισή τους γίνεται από ενσωματωμένο πληκτρολόγιο και η έξοδός τους είναι ένα ρελέ στο οποίο συνδέονται τα κουδούνια του κτιρίου, τα οποία λειτουργούν με 12 V DC ή κάποιες φορές με 220 V AC. Διαθέτουν συνήθως πλήκτρο χειροκίνητης ενεργοποίησης του κουδουνιού, τη στιγμή που το πατάμε, και διακόπτη, ο οποίος απομονώνει την αυτόματη λειτουργία σε περιόδους διακοπών, αργιών κ.τ.λ.

Το πρόβλημα παρουσιάζεται όταν χρειάζεται να αλλάξουν οι ώρες κουδουνίσματος. Αυτό γίνεται στις παρακάτω περιπτώσεις:

- Δύο φορές το χρόνο κατά τη μετάβαση από τη θερινή στη χειμερινή ώρα και αντίστροφα. Σ' αυτήν την περίπτωση αλλάζει η ώρα του συστήματος.
- Όταν πρέπει να συμπτυχθεί η διάρκεια των διδακτικών ωρών για να τελειώσει το πρόγραμμα νωρίτερα και να ακολουθήσει κάποιο έκτακτο γεγονός π.χ. μια σχολική εκδήλωση με συμμετοχή των μαθητών, μια συνεδρίαση εκπαιδευτικών με συμμετοχή αντιπροσωπείας των μαθητών ή χωρίς μαθητές.
- Όταν πρέπει να αλλάξουν οι ώρες έναρξης / λήξης του προγράμματος σε περίπτωση εκτάκτων γεγονότων, π.χ. σε περιπτώσεις καθυστέρησης έναρξης λόγω καιρικών συνθηκών (χιονοπτώσεις, καθυστερήσεις δρομολογίων ferry-boat αν το σχολείο φιλοξενεί μαθητές από κοντινά νησιά κ.τ.λ.)

Στις περιπτώσεις αυτές απαιτείται επαναπρογραμματισμός του συστήματος. Παρόλο που η ρύθμιση γίνεται σχετικά εύκολα από κάποιον ο οποίος έχει μια υποτυπώδη σχέση με ηλεκτρονικές συσκευές, υπάρχουν περιπτώσεις σχολείων με ελάχιστο προσωπικό που δεν έχει την παραμικρή σχέση με αυτοματισμούς (π.χ. νηπιαγωγεία, απομακρυσμένα δημοτικά). Επιπλέον, όταν οι λόγοι που επέβαλαν την αλλαγή δεν υπάρχουν πια (π.χ. την επομένη της συνεδρίασης/σχολικής εκδήλωσης ή όταν αποκατασταθούν οι συγκοινωνίες), το σύστημα θα πρέπει να επαναπρογραμματιστεί στην αρχική του κατάσταση. Αυτό αν δεν είναι δύσκολο, είναι τουλάχιστον χρονοβόρο.

Μια λύση η οποία προσφέρει περισσότερη ευελιξία είναι η χρήση λογισμικού, το οποίο αντλεί τα δεδομένα (ώρα, χρόνους και διάρκεια κουδουνισμάτων) απ' ευθείας από το διαδίκτυο [2]. Σε αυτήν την περίπτωση δε πρόκειται για απλό ενσωματωμένο σύστημα, αλλά για πλήρες υπολογιστικό σύστημα, στο οποίο εκτελείται στο παρασκήνιο ένα πρόγραμμα που ενεργοποιεί σε συγκεκριμένες χρονικές στιγμές το κουδούνι, μέσω μιας απλής συσκευής USB ή συχνότερα, αναπαράγει κάποιους ήχους ή προκαθορισμένα μηνύματα στο ηχητικό σύστημα του σχολείου. Μπορούν επίσης να εμφανίζονται κάποια μηνύματα κειμένου σε επιγραφές LED σε διάφορα σημεία του κτιρίου. Στην περίπτωση αυτή, οι απαιτήσεις είναι εκτός από έναν αξιόπιστο υπολογιστή, η απρόσκοπτη σύνδεση στο διαδίκτυο, ηχητική εγκατάσταση σε όλο το κτιριακό συγκρότημα και ενδεχομένως κάποιες επιγραφές LED. Ο υπολογιστής ιδανικά θα έπρεπε να είναι αφιερωμένος σ' αυτή τη δουλειά. Η λύση

αυτή είναι σπάνια λόγω της έλλειψης αξιοπιστίας των υπολογιστών στα σημερινά δημόσια σχολεία κυρίως λόγω της ηλικίας τους, αλλά και των προβληματικών συνδέσεων (υπάρχουν σχολεία που μπορεί να μείνουν χωρίς σύνδεση στο διαδίκτυο για ημέρες).

1.2 Η προτεινόμενη λύση

Στην παρούσα εργασία προτείνεται η χρήση απλού ενσωματωμένου συστήματος για σχολικό κουδούνι με τα παρακάτω χαρακτηριστικά :

- Μικροελεγκτής ο οποίος ελέγχει την ώρα ενεργοποίησης και τη διάρκεια κάθε κουδουνίσματος
- Ρύθμιση των κουδουνισμάτων της σημερινής μέρας (ώρα – διάρκεια) μέσω joystick.
- Ρύθμιση των κουδουνισμάτων κάθε μέρας (ώρα – διάρκεια) μέσω αρχείου κειμένου (απαιτείται σύνδεση με υπολογιστή).
- Οθόνη LCD 20x4 που δείχνει
    - την ώρα την ημερομηνία και το επόμενο κουδούνισμα σε κανονική λειτουργία
    - το επόμενο κουδούνισμα για αλλαγή σε λειτουργία ρύθμισης
- Ηχητικό σήμα μέσω πιεζοηλεκτρικής σειρήνας 12 V
- Τροφοδοσία :
    - Μέσω USB σε κανονική λειτουργία με τη βοήθεια φορτιστή κινητού 5V.
    - Μέσω της USB θύρας του υπολογιστή σε λειτουργία προγραμματισμού.
- Στη συγκεκριμένη σχεδίαση χρησιμοποιείται επιπλέον τροφοδοτικό 12V / 2A για τη λειτουργία της πιεζοηλεκτρικής σειρήνας, η οποία απαιτεί περίπου 300 mA. Ο έλεγχος του κυκλώματος 12 V γίνεται με ένα MOSFET ισχύος.
- Δυνατότητα απομακρυσμένης ρύθμισης (όχι ελέγχου) με μοναδικές απαιτήσεις την ύπαρξη λογισμικού απομακρυσμένου ελέγχου υπολογιστή (π.χ. Teamviewer, Anydesk) και μηδαμινές τεχνικές γνώσεις από τον εκπαιδευτικό χειριστή του κουδουνιού.

1.3 Αξιοπιστία / Στιβαρότητα

Ένα σύστημα σχολικού κουδουνιού θα πρέπει να προσφέρει αξιοπιστία στη λειτουργία του. Η πιο δύσκολη περίπτωση θα ήταν να κολλήσει το σύστημα, με το κουδούνι στην κατάσταση ON. Για την αποφυγή –κυρίως- αυτής της κατάστασης αλλά και οποιασδήποτε άλλης μη φυσιολογικής, το σύστημα διαθέτει τους παρακάτω μηχανισμούς:

- Επανεκκίνηση σε συγκεκριμένη χρονική στιγμή κάθε 24ωρο. Προεπιλεγμένη τιμή είναι κάποια στιγμή λίγο μετά τα μεσάνυχτα. Είναι γνωστό ότι η επανεκκίνηση λύνει ένα πολύ μεγάλο ποσοστό προβλημάτων σε όλα τα υπολογιστικά συστήματα, απλούστερα ή πιο περίπλοκα (υπολογιστές, κινητά, ενσωματωμένα συστήματα). Επιπλέον, η καθημερινή επανεκκίνηση ακυρώνει τυχόν αλλαγές στους χρόνους ενεργοποίησης του κουδουνιού οι οποίες έγιναν την προηγούμενη μέρα.
- Κύκλωμα watchdog το οποίο ελέγχει τη φυσιολογική λειτουργία του συστήματος και επανεκκινεί το μικροελεγκτή όταν αυτός κολλήσει. Η λειτουργία του, η οποία περιγράφεται λεπτομερώς παρακάτω είναι να στέλνει ένα σήμα επανεκκίνησης στην αντίστοιχη είσοδο όταν ο μ/ε σταματήσει να εναλλάσσει το σήμα σε έναν ακροδέκτη εξόδου.
- Έξοδος η οποία αν για οποιοδήποτε λόγο βρεθεί «στον αέρα», δηλαδή σε κατάσταση όπου δεν εφαρμόζεται προκαθορισμένη τάση (π.χ. αν θεωρηθεί είσοδος στο μ/ε και όχι έξοδος μετά

από λανθασμένη παραμετροποίηση), κρατιέται σε κατάσταση OFF μέσω μιας pull-down αντίστασης στην αντίστοιχη πύλη του FET ελέγχου. Στην περίπτωση χρήσης εξωτερικού ρελέ, συνιστάται χρήση κλασσικού ηλεκτρομαγνητικού ρελέ με σύνδεση του κουδουνιού στην Normally Open επαφή. Δεν συνιστάται η χρήση SSR (ρελέ στερεάς κατάστασης), λόγω του γεγονότος ότι οι συγκεκριμένες διατάξεις είναι δυνατό σε περίπτωση βλάβης να βρεθούν σε κατάσταση ON.

Κεφάλαιο 2ο: Κύκλωμα διασφάλισης ομαλής λειτουργίας (watchdog)

2.1 Απαιτήσεις κυκλώματος

2.1.1 Απλότητα

Τα περισσότερα κυκλώματα watchdog βασίζονται σε έναν μετρητή, ο οποίος όταν υπερχειλίσει στέλνει σήμα επανεκκίνησης (reset) στο υπολογιστικό σύστημα. Επομένως, σε τακτά χρονικά διαστήματα θα πρέπει να μηδενίζεται ο μετρητής, όσο το πρόγραμμα εκτελείται φυσιολογικά. Αυτό θα απαιτούσε εξωτερικό ψηφιακό κύκλωμα μέτρησης. Ακόμη, για λόγους αξιοπιστίας, θα απαιτούνταν εξωτερικό κύκλωμα χρονισμού (ταλαντωτής), για να είναι σίγουρο ότι σε περίπτωση δυσλειτουργίας ο μετρητής θα συνεχίζει να λειτουργεί, ακόμη κι αν κολλήσει ο μικροελεγκτής σε μια κατάσταση που πρέπει να αποφευχθεί με κάθε τρόπο, π.χ. αν κολλήσει στην κατάσταση που το κουδούνι χτυπάει.

Σημειώνεται ότι απλό κύκλωμα το οποίο δουλεύει με τον παραπάνω τρόπο περιλαμβάνεται ήδη στη σχεδίαση του ArduinoUno, με χρήση της βιβλιοθήκης Watchdogtimer.

Τα παραπάνω ζητούμενα αυξάνουν την πολυπλοκότητα, άρα μειώνουν την αξιοπιστία του εξωτερικού κυκλώματος watchdog. Έτσι επιλέχθηκε η χρήση απλού κυκλώματος διαφόρισης, που θα παραμένει High όσο ο μικροελεγκτής στέλνει τετραγωνικό παλμό με συχνότητα 50 Hz, δηλαδή αντιστροφή μιας εξόδου κάθε 10 ms (T = 20ms). Το σήμα θα γίνεται Low μετά από κάποιο χρόνο αδράνειας της εξόδου αυτής και θα παράγει ένα σήμα (reset) που θα συνδέεται στην αντίστοιχη είσοδο του μικροελεγκτή. Λόγω της απαίτησης απλότητας, το κύκλωμα διαφόρισης δεν σχεδιάστηκε με τη βοήθεια τελεστικών ενισχυτών και άλλων σύνθετων κυκλωμάτων αλλά μόνο με διόδους, πυκνωτές και αντιστάσεις και με ένα τμήμα διακοπτικής λειτουργίας με MOSFET-Διακόπτες.

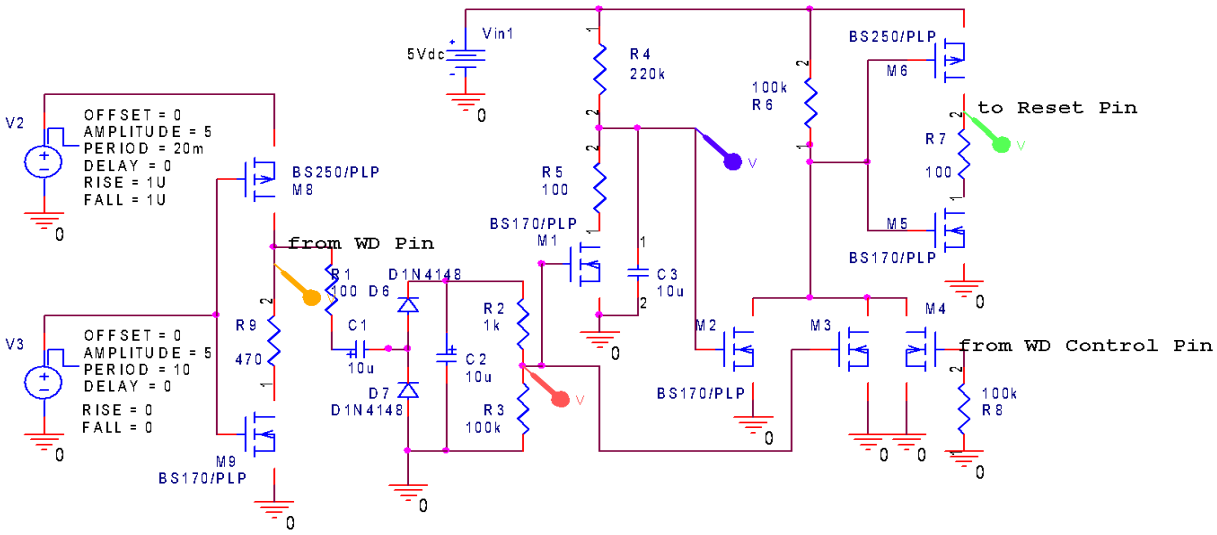
2.1.2 Σήμα reset για περιορισμένο χρόνο

Η σχεδίαση του κυκλώματος θα πρέπει να είναι τέτοια ώστε η έξοδος που συνδέεται στο reset pin του μικροελεγκτή, να παραμένει ενεργή (Low : Reset) για περιορισμένο χρονικό διάστημα, αλλιώς υπάρχει κίνδυνος ο μικροελεγκτής να μην ξαναξεκινήσει. Σε διάφορα manual αναφέρεται ότι το σήμα reset θα πρέπει να παραμείνει LOW για πάνω από 120 milliseconds. Με δεδομένη την ευρεία διάδοση του παραπάνω μικροελεγκτή και το γεγονός ότι οι κλώνοι του δεν είναι σίγουρο ότι θα ακολουθούν πιστά την παραπάνω χρονική συνθήκη, αποφασίστηκε το σήμα reset να παραμένει Low για χρόνο της τάξης του ενός περίπου δευτερολέπτου.

2.1.3 Αρχική έξοδος High

Η έξοδος του κυκλώματος θα πρέπει να βρίσκεται στην κατάσταση High τη στιγμή της εκκίνησης (Power on) ή τουλάχιστον σε σύντομο χρόνο μετά από αυτήν, για να μην εγκλωβιστούμε στην κατάσταση Low, όπου ο μικροελεγκτής δε θα ξεκινά.

Το κύκλωμα που χρησιμοποιήθηκε φαίνεται στην εικόνα 2.1.1



Εικόνα 2.1.1 : Το κύκλωμα watchdog που χρησιμοποιήθηκε

2.2 Επεξήγηση λειτουργίας

Το κύκλωμα Watchdog (κύκλωμα διαφόρισης) στην παραπάνω εικόνα είναι μόνο το κομμάτι από τον αριστερό ενδείκτη τάσης (κίτρινο) έως τον δεύτερο από αριστερά (κόκκινο). Είναι κύκλωμα της λογικής αντλίας φόρτισης AC, παραλλαγή αρκετά συνηθισμένου κυκλώματος ανίχνευσης [3]. Το αριστερό τμήμα χρησιμοποιήθηκε για τον έλεγχο σε προσομοίωση στο Pspice, για παραγωγή διακοπτόμενων παλμών περιόδου 20ms. Το δεξιό τμήμα αναλαμβάνει τη διακοπτική λειτουργία και τη χρονοκαθυστέρηση του παλμού reset.

2.2.1 Κυρίως κύκλωμα Watchdog

Ο πυκνωτής C_1 κόβει τη συνεχή συνιστώσα του παλμού εισόδου (2.5 Volt στην περίπτωση απευθείας εισόδου από το μικροελεγκτή, 0-5 Volt). Στην αρνητική ημιπερίοδο η D_7 διατηρεί το σημείο ανάμεσα στις διόδους ελαφρώς κάτω από το μηδέν (περίπου -0,7 V) . Στη θετική ημιπερίοδο η D_7 είναι ανάστροφα πολωμένη και η D_6 ορθά, με αποτέλεσμα κάποιο τμήμα του φορτίου μεταφέρεται από το C_1 στο C_2 αυξάνοντας την τάση του. Ο C_2 εκφορτίζεται αργά μέσω των αντιστάσεων R_2 και R_3. Μετά από μερικούς κύκλους, η τάση του C_2 θα είναι αρκετά ψηλά. Η έξοδος του κυρίως κυκλώματος (κόκκινος ενδείκτης τάσης) είναι κοντά στην τάση φόρτισης του C_2, αφού $R_2 << R_3$ και παραμένει ψηλά (πάνω από 2.5 V). Έτσι ενεργοποιούνται τα MOSFET που ελέγχει, δηλαδή M_1 και M_3.

Όταν δεν υπάρχει μεταβαλλόμενος παλμός εισόδου (κίτρινος ενδείκτης) , δηλαδή όταν το κύκλωμα κολλήσει, ανεξάρτητα από το αν θα βρεθεί σε κατάσταση 0 η 5 V, ο πυκνωτής C_1 απομονώνει και ο C_2 εκφορτίζεται μέσω των αντιστάσεων R_2 και R_3. Η έξοδος πέφτει εκθετικά προς το μηδέν σε χρόνο που εξαρτάται από τις τιμές των C_2 και R_2, R_3 και την αρχική τάση στα άκρα του C_2 κατά τη διάρκεια του μεταβαλλόμενου παλμού. Το δυναμικό στο σημείο εξόδου (κόκκινος ενδείκτης) θα πέσει εκθετικά στο μηδέν και τα M_1 και M_3 θα απομονώσουν.

2.2.2 Διακοπτικό τμήμα

Όταν η έξοδος του κυκλώματος διαφόρισης (κόκκινος ενδείκτης τάσης) είναι πάνω από 2 V, το FET M_1 άγει και το σημείο ανάμεσα στις αντιστάσεις R_4 και R_5 (μπλε) είναι σε χαμηλό δυναμικό, εφόσον $R_4 >> R_5$ και ο πυκνωτής C_3 είναι αφόρτιστος. Η τάση στο παραπάνω σημείο ελέγχει το FET M_2 το οποίο απομονώνει. Παράλληλα με το M_2 υπάρχουν τα M_3 και M_4 . Θεωρούμε ότι η πύλη του M_4 είναι αρχικά γειωμένη, άρα απομονώνει και αυτό. Όμως η έξοδος του κυκλώματος διαφόρισης ελέγχει και το M_3. Επομένως αφού αυτό άγει, οι πύλες των M_5 και M_6 (συνδεμένα σε συνδεσμολογία μισής γέφυρας) είναι σε δυναμικό 0 με αποτέλεσμα να άγει το M_6 και η έξοδος (πράσινος ενδείκτης τάσης) να είναι σε υψηλό δυναμικό.

Όταν κάποια στιγμή η είσοδος του κυκλώματος διαφόρισης σταματήσει να εναλλάσσεται, η έξοδος του κυρίως κυκλώματος (σημείο που βρίσκεται ο κόκκινος ενδείκτης), γίνεται μετά από λίγο Low και τα M_1 και M_3 απομονώνουν. Η είσοδος του M_2 βρίσκεται - για λίγο - σε δυναμικό 0 λόγω της ύπαρξης του πυκνωτή C_3 , ο οποίος αρχίζει να φορτίζεται μέσω της R_4. Όσο ο C_3 φορτίζεται, τα M_2, M_3 και M_4 είναι όλα σε κατάσταση απομόνωσης, με αποτέλεσμα οι πύλες των M_5 και M_6 να είναι σε ψηλό δυναμικό λόγω της R_6 και να άγει το M_5 γειώνοντας την έξοδο. Αυτό διαρκεί μόνο όσο χρόνο φορτίζεται ο πυκνωτής C_3 και κατόπιν η πύλη του M_2 (μπλε) έρχεται σε υψηλό δυναμικό, γειώνονται οι πύλες των M_5 και M_6 και η έξοδος ξαναγίνεται High.

2.2.3 Λειτουργία του M4

Για λόγους αξιοπιστίας στην αρχικοποίηση του συστήματος, φροντίζουμε η είσοδος του M_4 να είναι High, ελέγχοντάς την αρχικά (από τις πρώτες εντολές) στο πρόγραμμά μας με έναν ακροδέκτη εξόδου του μικροελεγκτή (WD ControlPin). Έτσι διασφαλίζουμε ότι η τελική έξοδος του κυκλώματος θα είναι High, έως ότου το κύκλωμα θα λειτουργεί κανονικά, γεγονός το οποίο απαιτεί μερικές εναλλαγές παλμών στην είσοδο, άρα κάποιο χρόνο. Όταν η λειτουργία των παλμών είναι κανονική, τότε φέρνουμε το WD ControlPin σε δυναμικό 0 (Low) με αποτέλεσμα το M_4 να απομονώνει, αφήνοντας το υπόλοιπο κύκλωμα να λειτουργήσει κανονικά σαν WatchDog. Ο χρόνος αυτός επιλέχθηκε να είναι 0.5 second, χρόνος υπεραρκετός, εφόσον το κύκλωμα έρχεται σε κατάσταση ισορροπίας πολύ νωρίτερα.

2.3 Χρόνοι

Η τάξη μεγέθους των χρόνων ενεργοποίησης και απενεργοποίησης του κυκλώματος υπολογίζεται από τις σταθερές εκφόρτισης των διατάξεων πυκνωτή-αντίστασης σε κάθε περίπτωση.

Χρόνος ενεργοποίησης (Output High ->Low): Για να υπολογίσουμε περίπου τον χρόνο αυτόν θα βρούμε τη σταθερά RC για τον Πυκνωτή C_2 (10μF) και το άθροισμα των αντιστάσεων R_2 και R_3 (101 kΩ) :

$$(R_2 + R_3) \cdot C_2 = (1k\Omega + 100k\Omega) \cdot 10\mu F = 101 \cdot 10^3 \Omega \cdot 10 \cdot 10^{-6} F \approx 1\, s \qquad (2.3\text{-}1)$$

Βρίσκουμε επομένως ότι σε 1 δευτερόλεπτο ο πυκνωτής εκφορτίζεται με εκθετικό τρόπο στο 37% της αρχικής του τάσης.

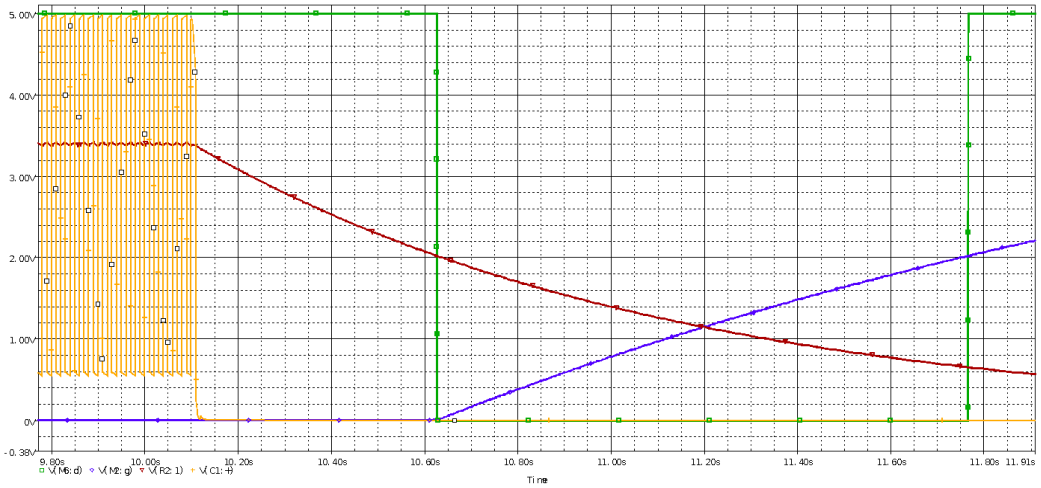
Χρόνος απενεργοποίησης (OutputLow->High): Για τον χρόνο αυτόν θα βρούμε τη σταθερά RC για τον Πυκνωτή C_3 (10μF) και την αντίσταση R_4 (220 kΩ) :

$$R_4 \cdot C_3 = 220k\Omega \cdot 10\mu F = \ 220 \cdot \ 10^3\Omega \ \cdot 10 \cdot 10^{-6}F = 2.2\ s \qquad (2.3\text{-}2)$$

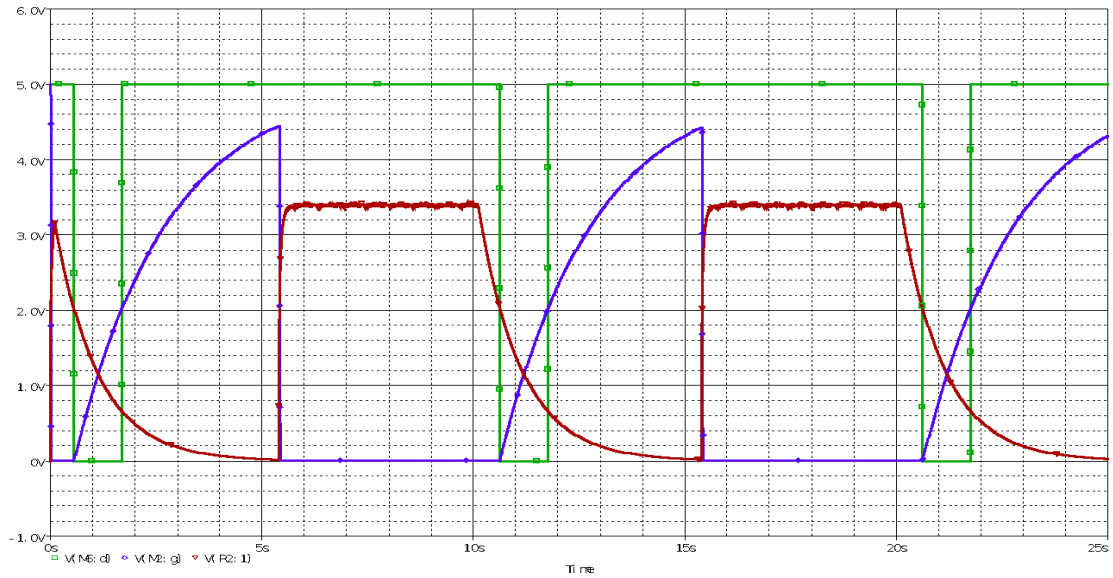
Επομένως σε 2,2 δευτερόλεπτα ο πυκνωτής φορτίζεται εκθετικά στο 63% της μέγιστης τάσης.

Σημείωση: Για να βρούμε τους ακριβείς χρόνους ενεργοποίησης / απενεργοποίησης του κυκλώματος, θα έπρεπε να υπολογίσουμε ακριβώς τη στάθμη φόρτισης / ισορροπίας του πυκνωτή C2 (αρκετά σύνθετο πρόβλημα, το οποίο απαιτεί μεθόδους με ολοκλήρωση) και κατόπιν με αυτό το δεδομένο να υπολογίσουμε εκθετικά το χρόνο ο οποίος απαιτείται για την πτώση αυτής της τάσης στα 2 V ή την άνοδό της από το 0 στα 2V. Με δεδομένο ότι τα συγκεκριμένα FET έχουν τάση ενεργοποίησης περίπου στα 2 V (για ένα από αυτά μετρήθηκε στα 1,92V), και επιπλέον ότι οι χρόνοι ενεργοποίησης απενεργοποίησης μας ενδιαφέρουν μόνο κατά προσέγγιση, ο ακριβής υπολογισμός ξεφεύγει από τα πλαίσια της παρούσας εργασίας, και κυρίως δε χρειάζεται για το συγκεκριμένο πρόβλημα

Όπως φαίνεται στην προσομοίωση (εικόνα 2.3.1), για το κύκλωμα ανίχνευσης παλμών (είσοδος 0,6 έως 5 Volt - πορτοκαλί, έξοδος 3,4 ± 0,02Volt - κόκκινο) ο χρόνος ενεργοποίησης είναι 0.5 second (10.62 s - 10,11s) και ο χρόνος απενεργοποίησης είναι 1.1 second (11.76 s - 10.62 s). Η κανονική λειτουργία φαίνεται και σε πιο μακρόχρονη προσομοίωση (εικόνα 2.3.2), όπου η λειτουργία διακόπτεται και αποκαθίσταται ανά 10 δευτερόλεπτα.

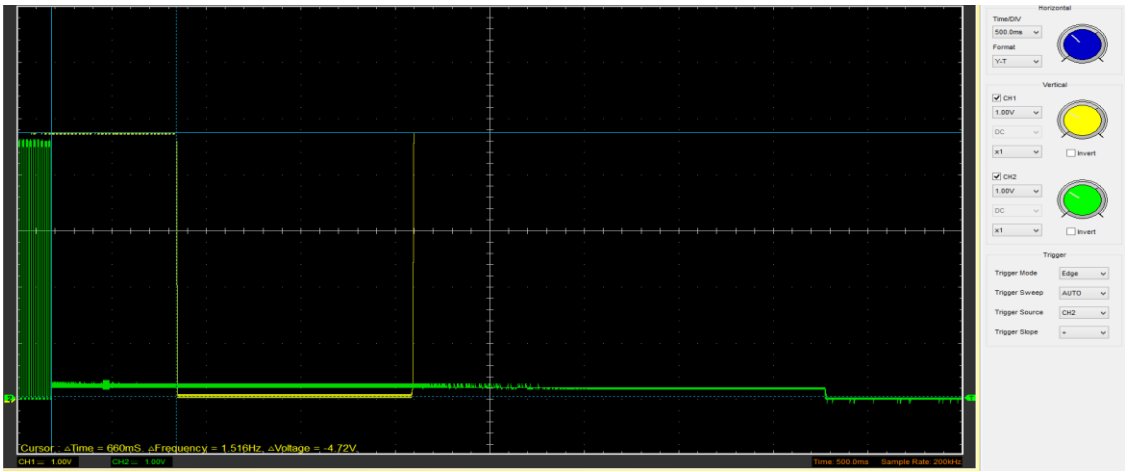


Εικόνα 2.3.1 : Προσομοίωση διακοπής παλμών (2 seconds)

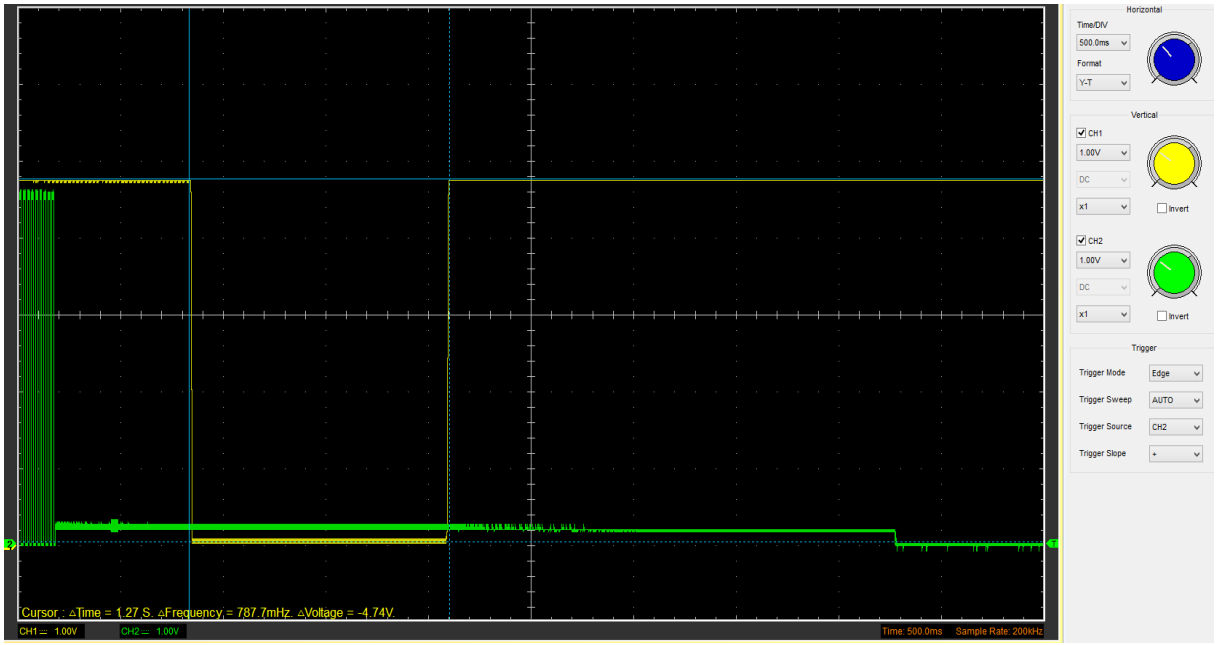


Εικόνα 2.3.2 : Προσομοίωση διακοπής παλμών (25 seconds)

Ανάλογα αποτελέσματα φαίνοονται και στο πραγματικό κύκλωμα, σε στιγμιότυπα σε παλμογράφο USB (Hantek 6022 BE) δύο καναλιών. Στις παρακάτω εικόνες το πράσινο είναι η είσοδος και το κίτρινο η έξοδος του κυκλώματος. Οι μετρήσεις δείχνουν χρόνο ενεργοποίησης (Output High → Low) 660 ms (εικόνα 2.3.3) και χρόνο απενεργοποίησης (Output Low → High) 1.27 s (εικόνα 2.3.4).

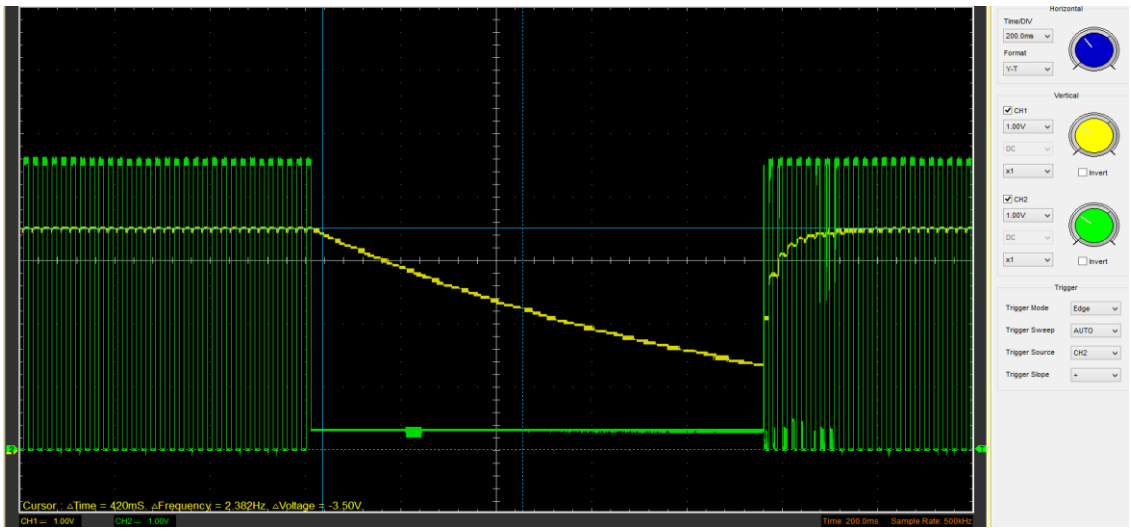


Εικόνα 2.3.3 : Μέτρηση χρόνου ενεργοποίησης Reset μετά τη διακοπή παλμών



Εικόνα 2.3.4 : Μέτρηση χρόνου απενεργοποίησης Reset μετά τη διακοπή παλμών

Η τάση φυσιολογικής λειτουργίας του Watchdog μετρήθηκε στα 3.5 V με πολύ μικρή διακύμανση (εικόνα 2.3.5), πολύ κοντά σ' αυτήν της προσομοίωσης (3.4 V). Στην ίδια εικόνα φαίνεται και ο χρόνος αποκατάστασης να είναι μικρότερος της χρονικής υποδιαίρεσης της οθόνης του παλμογράφου (200 ms). Στη συγκεκριμένη εικόνα το κίτρινο σήμα είναι η έξοδος του κυκλώματος ανίχνευσης (κόκκινο στο αρχικό σχέδιο)



Εικόνα 2.3.5 : Τάση φυσιολογικής λειτουργίας watchdog

Κεφάλαιο 3ο: Επιλογή εξαρτημάτων

Σήμερα υπάρχει πληθώρα εξαρτημάτων τα οποία μπορούν να χρησιμοποιηθούν στη θέση του κεντρικού υπολογιστή/ελεγκτή. Ενδεικτικά κάποιες γνωστές οικογένειες συστημάτων

- Raspbery pi: Πρόκειται για οικογένεια ολοκληρωμένων υπολογιστικών συστημάτων με δυνατότητα πληκτρολογίου, οθόνης, με μνήμη της τάξης των Giga Byte, με δικό του λειτουργικό σύστημα (τύπου Linux). Υπάρχει όμως και απαίτηση ψύξης με ανεμιστήρα πάνω στον επεξεργαστή, όπως στους κλασσικούς υπολογιστές, σταθερούς ή φορητούς. Τα συγκεκριμένα συστήματα αναφέρονται συχνά σαν υπολογιστές σε μέγεθος κάρτας.
- ESP 32: Οικογένεια σύγχρονων συστημάτων από την εταιρεία Espressif με πολλές δυνατότητες σε σχέση με την τιμή τους. Οι περισσότερες λύσεις προσφέρουν συνδεσιμότητα (wifi / Bluetooth)
- Arduino: Οικογένεια συστημάτων με ευρεία διάδοση για εκπαιδευτική χρήση. Χρησιμοποιούνται επίσης από ερασιτέχνες / χομπίστες. Έχουν περιορισμένες δυνατότητες - υπολογιστικές και διασύνδεσης- αλλά υποστηρίζονται από τεράστια παγκόσμια κοινότητα χρηστών. Το περιβάλλον προγραμματισμού τους, γνωστό σαν Arduino IDE, έχει υιοθετηθεί και από την εταιρεία Espressif για τον προγραμματισμό των συστημάτων ESP32.
- STM 32: Οικογένεια σύγχρονων και ισχυρών συστημάτων από την εταιρεία ST. Προορίζονται κυρίως για μαζική / βιομηχανική παραγωγή, γι αυτό και το σύστημα το οποίο προσφέρεται στο εμπόριο σε τυπωμένη πλακέτα και φέρει τον κεντρικό ελεγκτή, έχει περιορισμένες δυνατότητες, επειδή δεν χρησιμοποιεί όλες τις ακίδες του μικροελεγκτή.
- PIC: Οικογένεια παλιότερων μικροελεγκτών από την εταιρεία Microchip. Χρησιμοποιούνται για εκπαιδευτική και σε απλές περιπτώσεις βιομηχανική χρήση.

Στις παραπάνω περιπτώσεις συστημάτων, η τελευταία (PIC) προσφέρει για αγορά μόνο το μικροελεγκτή, και χρειάζεται ξεχωριστή προμήθεια για την πλακέτα ανάπτυξης / προγραμματισμού. Στις υπόλοιπες προμηθευόμαστε ολόκληρο το σύστημα (όχι με πλήρη διασύνδεση στον STM 32 όπως αναφέρθηκε) με διασύνδεση USB για προγραμματισμό από υπολογιστή. Βασικά τεχνικά χαρακτηριστικά αναφέρονται στον πίνακα 1.

Πίνακας 1 : Τεχνικά Χαρακτηριστικά Επεξεργαστών - Μικροελεγκτών [7] - [12]

Σύστημα	Raspbery pi	ESP 32	Arduino	STM 32	PIC
Εταιρεία		ESPRESSIF		ST	Microchip
Μοντέλο	4 B /2019	DevKitC	UNO / 2010	STM32F3 Series	PIC32MZ /2013
Επεξεργαστής	Cortex-A72 (ARM v8) / 4 cores	Tensilica Xtensa LX6 / Dual-Core	ATmega328P	ARM Cortex-M4	PIC32MZ
Αρχιτεκτονική	64-bit	32-bit	8-bit	32-bit	32-bit
Μνήμη	1 GB +	520kB /448kB Flash	2 kB/ 32kB Flash	16 KB / 32 KB Flash	512KB/2 MB Flash
Συχνότητα λειτουργίας	1.5 GHz	160 / 240Mhz	16 MHz	72 MHz	252 MHz
Απαιτήσεις ισχύος	1.25 A @5V	100mA@5V max	100mA@5V	10mA@5V max	2mA@3.3V max
Απαιτήσεις ψύξης	Ανεμιστήρας στη CPU	-	-	-	-
Περιβάλλον / Γλώσσα προγραμματισμού	Linux / Python	Arduino IDE / C	Arduino IDE / C	Λογισμικό Εταιρείας / C	Λογισμικό Εταιρείας / C
Τιμή (ενδεικτική)	35 $	8 $	10 $	5 $	2 $

Τα παραπάνω χαρακτηριστικά και οι τιμές είναι ενδεικτικές και αναφέρονται στα πιο δημοφιλή μοντέλα της κάθε οικογένειας. Λεπτομερέστερα χαρακτηριστικά θα πρέπει να αναζητηθούν στους δικτυακούς τόπους κατασκευαστών και πωλητών, όπου και υπάρχει μεγαλύτερη ποικιλία μοντέλων.

3.1 Γιατί Arduino;

Οι απαιτήσεις του συγκεκριμένου συστήματος είναι υποτυπώδεις. Όπως αναφέρθηκε πρόκειται για έναν απλό χρονοδιακόπτη με πολλαπλά χρονικά σημεία ενεργοποίησης από τον οποίο όμως απαιτούμε αξιοπιστία και στιβαρότητα. Πολλά σχολεία χρησιμοποιούν στη θέση του κεντρικού ελεγκτή ένα απλό και φτηνό PLC, κόστους 100-200 € [4]. Για το λόγο της αξιοπιστίας θα απορρίψουμε τη λύση του υπολογιστή-κάρτας (Raspberry Pi), ο οποίος απαιτεί και σύστημα ψύξης και θα στραφούμε στη χρήση μικροελεγκτή.

Με βάση τα παραπάνω συγκριτικά στοιχεία, ιδανικός μικροελεγκτής ως προς την ευκολία προγραμματισμού και χρήσης είναι ο ESP32. Ο αξιόπιστος STM32 με πολλές πωλήσεις είναι σχετικά δύσχρηστος στην περίπτωση κατά την οποία ο τελικός χρήστης θα θελήσει να αλλάξει κάτι στις ρυθμίσεις και να ξαναπρογραμματίσει το σύστημα. Ο ESP32, από την άλλη, είναι ένα εύκολα προγραμματιζόμενο (και επαναπρογραμματιζόμενο) σύστημα με ασύρματη συνδεσιμότητα, υπεραρκετή μνήμη για τις απαιτήσεις του συγκεκριμένου προβλήματος, μικρός σε όγκο και με αμελητέα κατανάλωση ενέργειας. Το δυνατό του σημείο είναι το περιβάλλον προγραμματισμού το οποίο είναι το ίδιο με του δημοφιλέστατου Arduino και ο εκπαιδευτικός κόσμος το γνωρίζει πολύ καλά.

Εφόσον πάντως, το καθοριστικό κριτήριο για τη χρήση του συστήματος είναι η εξοικείωση των χρηστών με το μικροελεγκτή, ξεκάθαρος νικητής είναι ο Arduino Uno. Ο PIC, αν και παλαιότερος, είναι γνωστός σε σχετικά μικρό αριθμό σχολείων. Είναι κυρίως γνωστός σε τεχνικές σχολές και Επαγγελματικά Λύκεια, στα οποία αποτελούσε ένα από τα διδακτικά αντικείμενα με προγραμματιστικό περιβάλλον αρχικά σε γλώσσα Visual Basic και κατόπιν, σε C. Αντίθετα, ο Arduino Uno είναι σήμερα γνωστός σε όλες τις εκπαιδευτικές βαθμίδες, από το Δημοτικό έως την τριτοβάθμια εκπαίδευση, λόγω της έντονης προώθησης της εκπαιδευτικής ρομποτικής και του προγραμματισμού μικροσυστημάτων με εξόδους LEDs, buzzers και κινητήρες βηματικούς ή συμβατικούς. Η τεράστια διάδοσή του οφείλεται στα εξής :

- Έντονο ενδιαφέρον εκπαιδευτικών, οι οποίοι παρακινούν και το ενδιαφέρον των μαθητών, για προγραμματισμό σε συστήματα με έξοδο διαφορετική από την απεικόνιση σε μια οθόνη. (LEDs, κίνηση, ήχος).
- Πλήθος εκπαιδευτικών σεμιναρίων για δασκάλους και καθηγητές για τη χρήση της συγκεκριμένης πλατφόρμας.
- Εύχρηστη δωρεάν πλατφόρμα προγραμματισμού με ελάχιστες γνώσεις γλώσσας C. Η συγκεκριμένη πλατφόρμα συμπεριλαμβάνει πολλά προγράμματα-παραδείγματα ακόμη και για τον αρχάριο χρήστη.
- Απλούστατη δομή προγράμματος. Χρήση μόνο δύο functions της C: **`Setup()`** που εκτελείται μια φορά στην αρχή του προγράμματος και **`Loop()`**, που εκτελείται συνεχώς μέχρι να διακοπεί η τροφοδοσία του συστήματος. Διαφοροποίηση από την κλασσική δομή της C που χρησιμοποιείται από άλλα συστήματα :

```
void main()
{   . . . . // initialize
    while (true)
    {
          . . . . // do forever
    }
}
```

- Ευκολίες προγραμματισμού για ακόμη πιο αρχάριους χρήστες με τη χρήση block programming (προγραμματισμός με εντολές-τουβλάκια, τα οποία τοποθετούνται στο σωστό σημείο του προγράμματος με Drag and Drop, όπως, στο αγαπητό από τα παιδιά περιβάλλον Scratch)
- Εύκολα προσβάσιμες από το διαδίκτυο όλες οι παραπάνω ευκολίες, με αναλυτικές οδηγίες για ενσωμάτωση στο περιβάλλον προγραμματισμού και χρήση τους.
- Βιβλιοθήκες - οδηγοί για αξιοποίηση πολλών εξαρτημάτων υλικού για εκπαιδευτική χρήση όπως:
  - αισθητήρες πίεσης, θερμοκρασίας, υγρασίας, φωτός, απόστασης, ανίχνευσης αερίων, κ.τ.λ.
  - οδηγοί βηματικών κινητήρων H-brigde, Half-Bridge

  με τη βοήθεια δημοφιλών πρωτοκόλλων επικοινωνίας (I^2C, OneWire, SPI, . . .)
- Ποικιλία εισόδων/εξόδων σε ψηφιακή μορφή και σε αναλογική μορφή. Ο χρήστης για να χρησιμοποιήσει τον 10-bit ADC, δε χρειάζεται να κάνει τίποτε άλλο από το να αντιστοιχίσει την κλίμακα τάσης εισόδου 0 έως 5 Volt σε έναν αριθμό από 0 έως 1023 με απλή αναλογία, και δεν απαιτείται καμία γνώση για την τεχνική PWM που χρησιμοποιείται για την ψευτο-αναλογική έξοδο.
- Τεράστια παγκόσμια κοινότητα υποστήριξης. Το Arduino όπως και το Raspbery Pi δεν υποστηρίζονται από εταιρείες αλλά από κοινότητες συνεργασίας, όπως και τα τύπου LINUX Λειτουργικά Συστήματα. Είναι πρακτικά απίθανο να βρεθεί σήμερα ένα περιφερειακό για το οποίο να μην έχει γραφεί βιβλιοθήκη υποστήριξης για το περιβάλλον Arduino IDE, όπως και αναλυτικές οδηγίες για τη χρήση του.

Επομένως, εφόσον μας ενδιαφέρει η άνεση στη χρήση του μικροελεγκτή από τον τελικό χρήστη, προτιμήθηκε ο Arduino Uno, ο οποίος υπάρχει σε πολλά σχολεία και χρησιμοποιείται ήδη από τους εκπαιδευτικούς. Στο διαδίκτυο, σε πολλές εκπαιδευτικές σελίδες, αλλά και σε σελίδες με κατασκευές από χομπίστες, υπάρχουν πάρα πολλά κυκλώματα - παραδείγματα - ιδέες που μπορούν να αξιοποιηθούν για εκπαίδευση και γνωριμία με το συγκεκριμένο μικροελεγκτή. Στην εικόνα 3.1.1 βλέπουμε μια εργασία υλοποίησης κουδουνιού με κινητήρα και χρήση παραδοσιακού κουδουνιού.

Εικόνα 3.1.1 : Κουδούνι με Arduino [5]

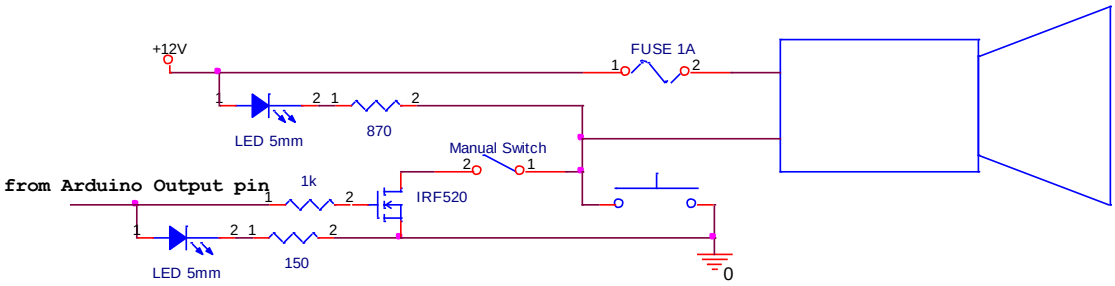
3.2 Τροφοδοσία - Έξοδος

Στην προτεινόμενη σχεδίαση όπως έχει ήδη αναφερθεί χρησιμοποιήθηκε ξεχωριστή τροφοδοσία για τον μικροελεγκτή και το κύκλωμα ελέγχου 5V/2A και ξεχωριστό τροφοδοτικό 12V/2A για το κύκλωμα ισχύος που ελέγχει τη σειρήνα. Συγκεκριμένα:

- Το κύκλωμα μικροελεγκτή / ελέγχου τροφοδοτείται από καλώδιο USB με τραπεζοειδή απόληξη (καλώδιο εκτυπωτών USB), το οποίο είναι το συνηθισμένο καλώδιο τροφοδοσίας και διασύνδεσης του Arduino και συνήθως συνοδεύει τον μικροελεγκτή με την αγορά του. Το καλώδιο αυτό συνδέεται με μια USB θύρα του υπολογιστή για τον επαναπρογραμματισμό του, όταν απαιτείται, ενώ συνδέεται σε φορτιστή ταξιδίου για USB συσκευές στη φυσιολογική του λειτουργία. Οι συγκεκριμένοι φορτιστές είναι αρκετά αξιόπιστοι για τροφοδοσία μικροελεγκτών, με αρκετά ομαλή έξοδο (η κυμάτωση μετρήθηκε και βρέθηκε κάτω από 10 mV).
- Το κύκλωμα ισχύος τροφοδοτείται από τροφοδοτικό 12 V/2A. Η σειρήνα η οποία χρησιμοποιήθηκε (100 dB) χρειάζεται μόλις 120 mA για τη λειτουργία της. Το κύκλωμα είναι προστατευμένο με ασφάλεια 1 A . Σημειώνεται ότι το MOSFET του κυκλώματος (IRF 520) μπορεί να άγει χωρίς ψήκτρα 2-3 A για μερικά δευτερόλεπτα.

Τα δύο κυκλώματα δεν είναι γαλβανικά απομονωμένα, έχουν κοινή γείωση (εικόνα3.2.1). Σε περίπτωση που το σύστημα χρησιμοποιηθεί με συμβατικά κουδούνια σε ένα σχολικό κτίριο, θα πρέπει να συνδεθεί εξωτερικό ρελέ το οποίο απομονώνει το κύκλωμα ισχύος. Συνήθως, για λόγους ασφαλείας, δεν χρησιμοποιούνται επικίνδυνες τάσεις σε τέτοια κυκλώματα και κατά κανόνα λειτουργούν με 12 ή 24V DC. Οφείλει επομένως ο εγκαταστάτης να υπολογίσει την κατανάλωση κάθε κουδουνιού, εφόσον είναι αρκετά για να καλύπτουν όλο το κτίριο, υπολογίζοντας μερικές εκατοντάδες mA για κάθε κουδούνι.

Εάν χρησιμοποιηθεί εξωτερικό ρελέ, η τροφοδοσία των 12 V δε χρειάζεται και μπορεί στο σημείο που εφαρμόζονται τα 12 V στην εικόνα 3.2.1, να συνδεθεί η τροφοδοσία των 5 V. Εννοείται ότι το ρελέ το οποίο θα χρησιμοποιηθεί πρέπει να είναι αναλόγων δυνατοτήτων. Τονίζεται ότι υπάρχει ποικιλία από τέτοια ρελέ που χρησιμοποιούνται ειδικά σε κυκλώματα με τον Arduino και έχουν τη δυνατότητα να ελέγξουν αρκετά Ampere στην έξοδο.



Εικόνα 3.2.1 : Κύκλωμα ελέγχου Πιεζοηλεκτρικής Σειρήνας

Στην παρακάτω εικόνα (3.2.2) φαίνεται το σύστημα με τις συνδέσεις οι οποίες απαιτούνται για τον προγραμματισμό / ρύθμιση (USB → PC) και για τη φυσιολογική λειτουργία (USB → Wall charger).

Εικόνα 3.2.2 : Απαιτούμενες Συνδέσεις

Κεφάλαιο 4ο: Η κατασκευή

Για λόγους ευχρηστίας, οι αρχικές προδιαγραφές/απαιτήσεις μεταβλήθηκαν ελαφρώς προς το καλύτερο. Έτσι το κουδούνι εφοδιάστηκε με οθόνη 20x4 και όχι 16x2 που ήταν η αρχική απαίτηση. Αυτό έγινε επειδή η μικρή οθόνη κατά κανόνα δημιουργεί προβλήματα στους χρήστες. Έτσι σε ένα εργασιακό περιβάλλον όπως το σχολείο, όπου το προσωπικό αλλάζει σχεδόν κάθε σχολική χρονιά, οι μικρές οθόνες κάνουν το κάθε σύστημα δύσχρηστο. Αυτό φαίνεται σε ένα δημοφιλές σύστημα το οποίο υπάρχει σε αρκετά σχολεία, αλλά η οθόνη του περιέχει μόνο 4 ενδείκτες 7 τομέων και μερικά επιπλέον LEDs (εικόνα 4.1).

Εικόνα 4.1 : Κουδούνι γνωστής εταιρίας [6] σε λειτουργία

Τα κουδούνια PLC τα οποία είναι πιο αξιόπιστες κατασκευές, συνήθως αγοράζονται χωρίς οθόνη για λόγους κόστους. Κι εδώ επομένως ο προγραμματισμός είναι δύσκολος και η ρύθμιση απαιτεί εμπειρία η οποία συνήθως δεν υπάρχει στο προσωπικό των περισσότερων σχολείων.

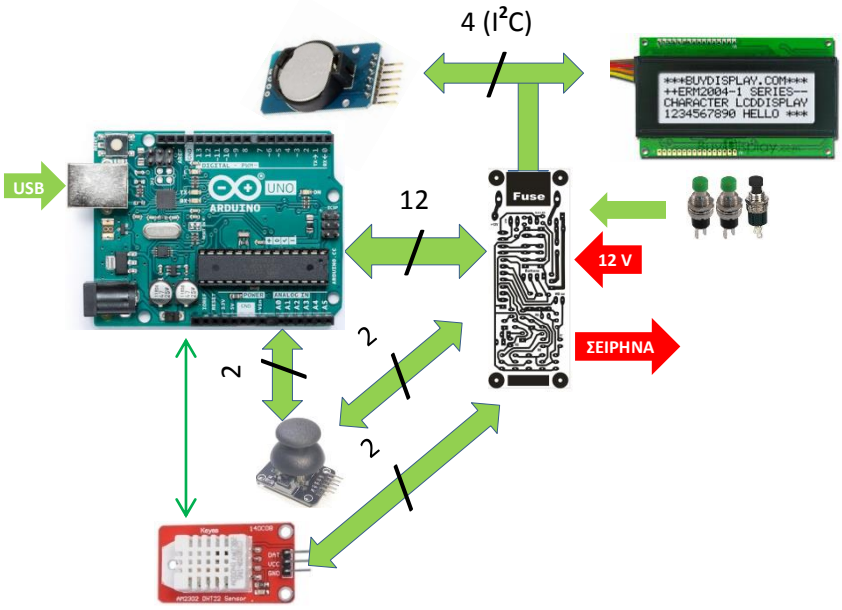
Η μεγαλύτερη οθόνη LCD, η οποία ενσωματώθηκε παρέχει αρκετές πληροφορίες ειδικά κατά το στάδιο του προγραμματισμού, χαρακτηριστικό που κάνει τη ρύθμιση εύκολη, ακόμη και για τον μη εξοικειωμένο χρήστη.

Ενσωματώθηκε επιπλέον αισθητήριο υγρασίας / θερμοκρασίας DHT22. Το αισθητήριο αυτό τοποθετήθηκε πολύ κοντά στο FET ελέγχου σειρήνας, έτσι ώστε τυχόν αύξηση της θερμοκρασίας, να γίνει αντιληπτή από τον επιτηρητή του συστήματος, ακόμη και αν αυτός το ελέγχει από μακριά, μέσω λογισμικού απομακρυσμένου ελέγχου.

Η κατασκευή στο σύνολό της περιλαμβάνει τα εξής :

- Arduino UNO
- LCD 20x4
- Real Time Clock DS3231 με μπαταρία (2032 Button Cell, 3 Volt).
- 3 πλήκτρα για το μενού ρύθμισης
- Joystick μεταβολής ρύθμισης κουδουνισμάτων
- Είσοδο USB για επικοινωνία με υπολογιστή ή τροφοδοσία από φορτιστή ταξιδίου 220V/5V
- Είσοδο κυκλώματος 12V
- Έξοδο 12V για την πιεζοηλεκτρική σειρήνα
- Αισθητήριο υγρασίας / θερμοκρασίας DHT22

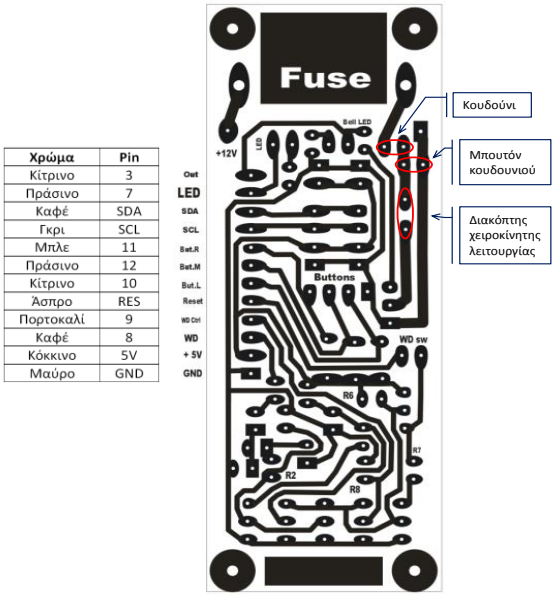
Στην εικόνα 4.2 φαίνονται τα εξαρτήματα και ο αριθμός των αγωγών που απαιτούνται για τη διασύνδεση του καθενός.



Εικόνα 4.2 : Τα μέρη του συστήματος

Η πλακέτα του κυκλώματος (εικόνα 4.3) σχεδιάστηκε έτσι ώστε να περιλαμβάνει :

- Το κύκλωμα watchdog
- 12 Ακροδέκτες για σύνδεση στην πλακέτα του μικροελεγκτή.
- 2x4 Ακροδέκτες για σύνδεση σε δύο περιφερειακά I2C. Χρησιμοποιήθηκαν για την οθόνη LCD 20x4 και για το Real Time Clock DS3231.
- 2 ζεύγη ακροδεκτών 5 Volt - γείωσης. Χρησιμοποιήθηκε το ένα ζεύγος για τη σύνδεση του joystick στα 0 και 5 Volt, και το άλλο για την τροφοδοσία του αισθητηρίου υγρασίας / θερμοκρασίας DHT22.



Χρώμα	Pin
Κίτρινο	3
Πράσινο	7
Καφέ	SDA
Γκρι	SCL
Μπλε	11
Πράσινο	12
Κίτρινο	10
Άσπρο	RES
Πορτοκαλί	9
Καφέ	8
Κόκκινο	5V
Μαύρο	GND

Εικόνα 4.3 : PCB - Χρωματικός κώδικας διασύνδεσης

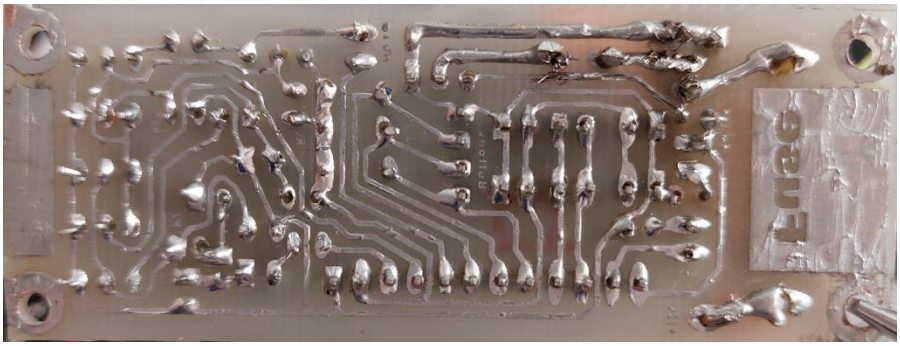
Η σχεδίαση της πλακέτας έγινε με σχεδιαστικό πρόγραμμα γενικής χρήσης (Corel Draw). Εκτυπώθηκε σε διαφάνεια σε inkjet εκτυπωτή. Μετά από φωτισμό σε λάμπα UV για 16 λεπτά και εμφάνιση με υδροξείδιο του Νατρίου, έγινε η αποχάλκωση με αραιό υδροχλωρικό οξύ και

υπεροξείδιο του υδρογόνου. Η πλακέτα επιμεταλλώθηκε ώστε να διασφαλιστεί η μακροζωία της με την απλή μέθοδο (κολλητήρι και σολντερίνη). Το τρύπημα έγινε αρχικά με τρυπάνι Φ 1 mm για τα περισσότερα εξαρτήματα και κάποιες τρύπες μεγάλωσαν εκ των υστέρων, π.χ. οι τρύπες για το FET (1.5 mm), για την ασφαλειοθήκη και τα καλώδια των 12V (2 mm), όπως και οι τρύπες στήριξης (3 mm). Κολλήθηκαν επίσης, για λόγους ασφαλείας, λεπτοί αγωγοί από πολύκλωνο καλώδιο στους φαρδύτερους αγωγούς των 12V, οι οποίοι ενώνουν τους ακροδέκτες Source και Drain του FET. Κατόπιν κολλήθηκαν τα εξαρτήματα του κυκλώματος και τα λεπτά καλώδια των 2 LEDs.

Οι ακροδέκτες που κολλήθηκαν στο τυπωμένο κύκλωμα ήταν male-male βήματος 0,25 mm (1/10 in, εικόνα 4.4 α) ενώ για το μικροελεγκτή χρησιμοποιήθηκαν ίδιου βήματος male-female εύκαμπτοι που χρειάστηκε να καμφθούν στις 90º για οικονομία χώρου (εικόνα 4.4 β). Η σύνδεση έγινε με καλώδια male-to-female

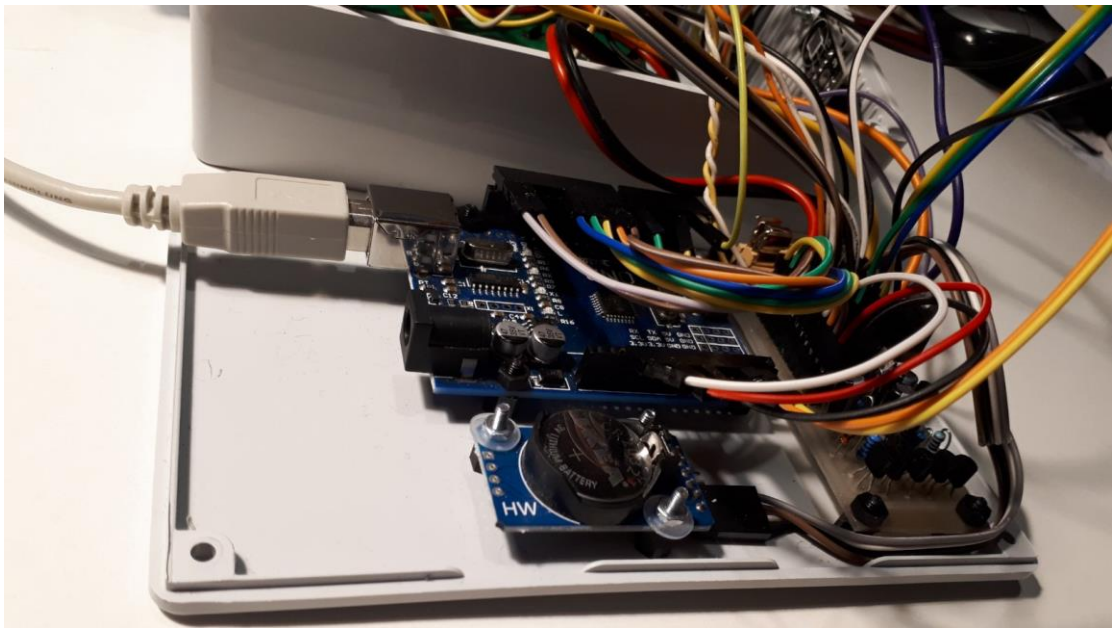
Εικόνα 4.4 : Ακροδέκτες για το PCB (α) και τον Arduino (β)

Η πλακέτα μετά τις κολλήσεις φαίνεται στην εικόνα 4.5

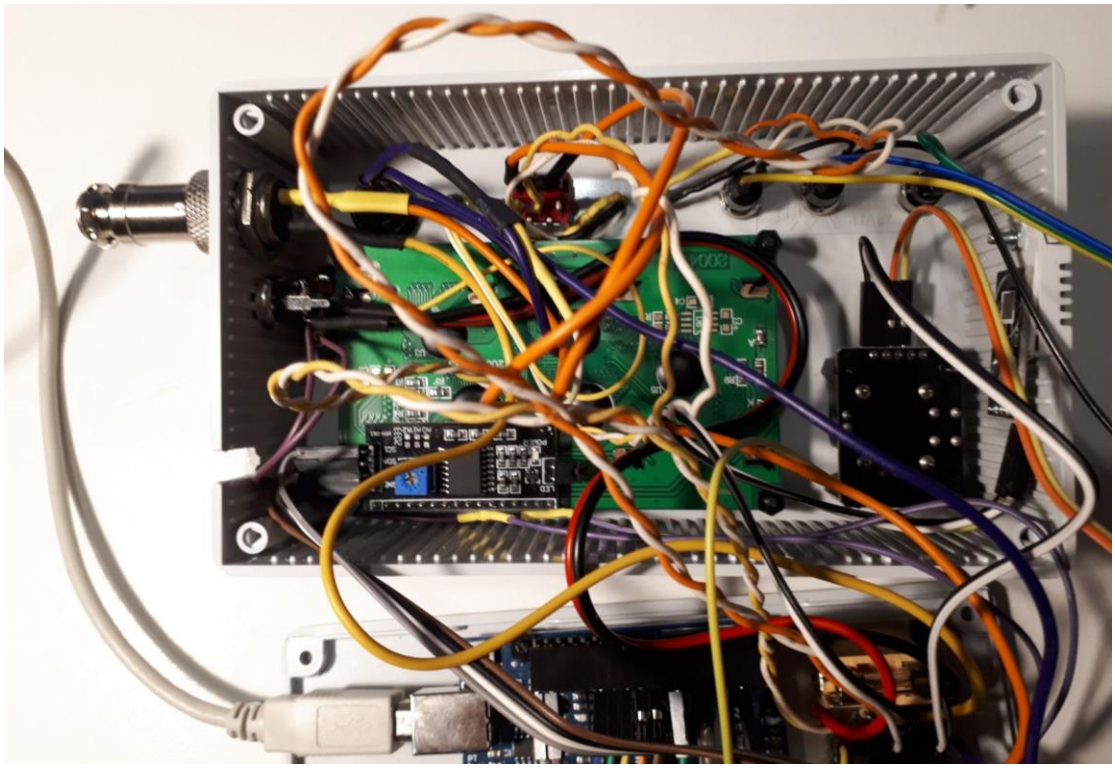


Εικόνα 4.5 : Η πλακέτα μετά τις κολλήσεις

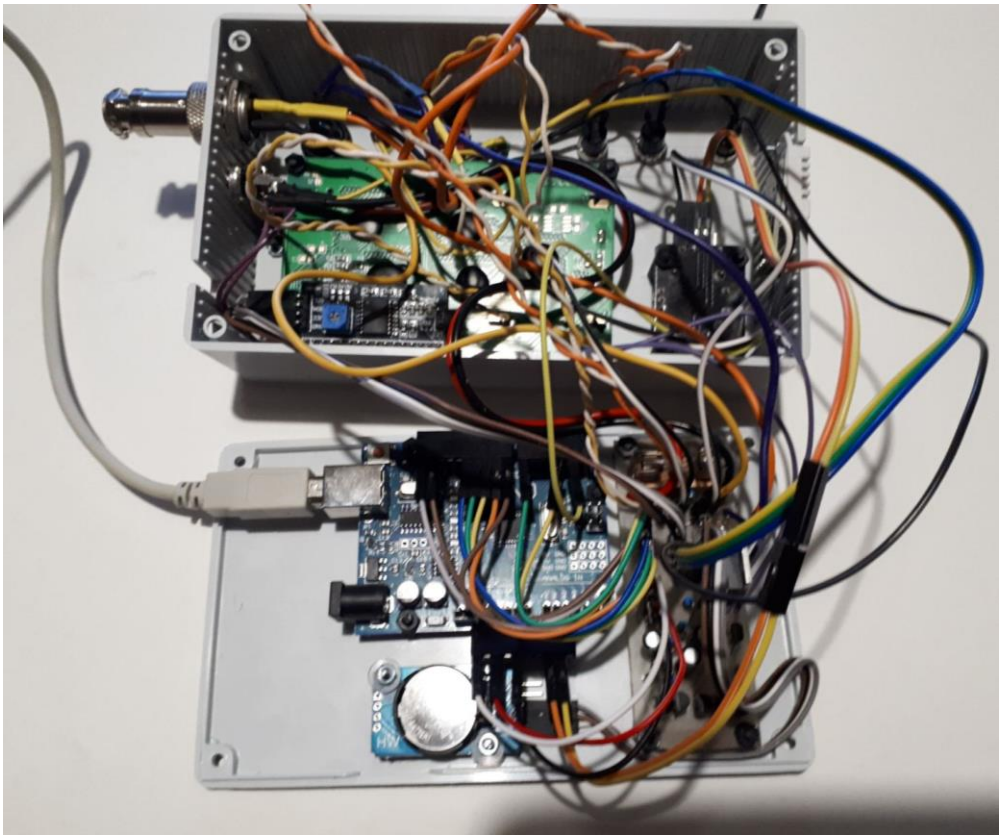
Η συναρμολόγηση έγινε σε κουτί κατασκευών 160x95x50 mm. Στις παρακάτω εικόνες φαίνονται τα σημαντικότερα στάδια.



Εικόνα 4.6 : Κάτω μέρος: Arduino, PCB, RTC.



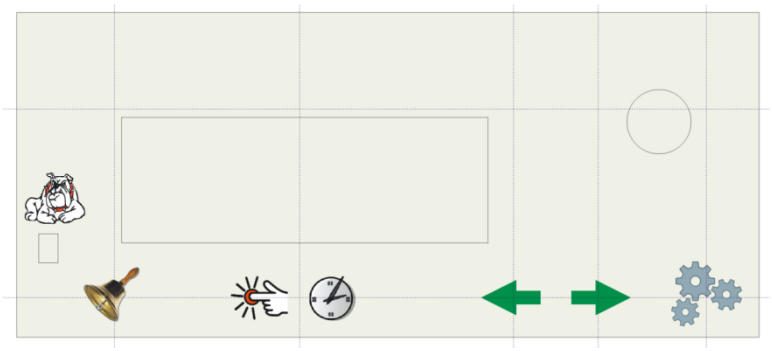
Εικόνα 4.7 : Εσωτερικό του καλύμματος : LCD, Joystick, buttons, DHT22.



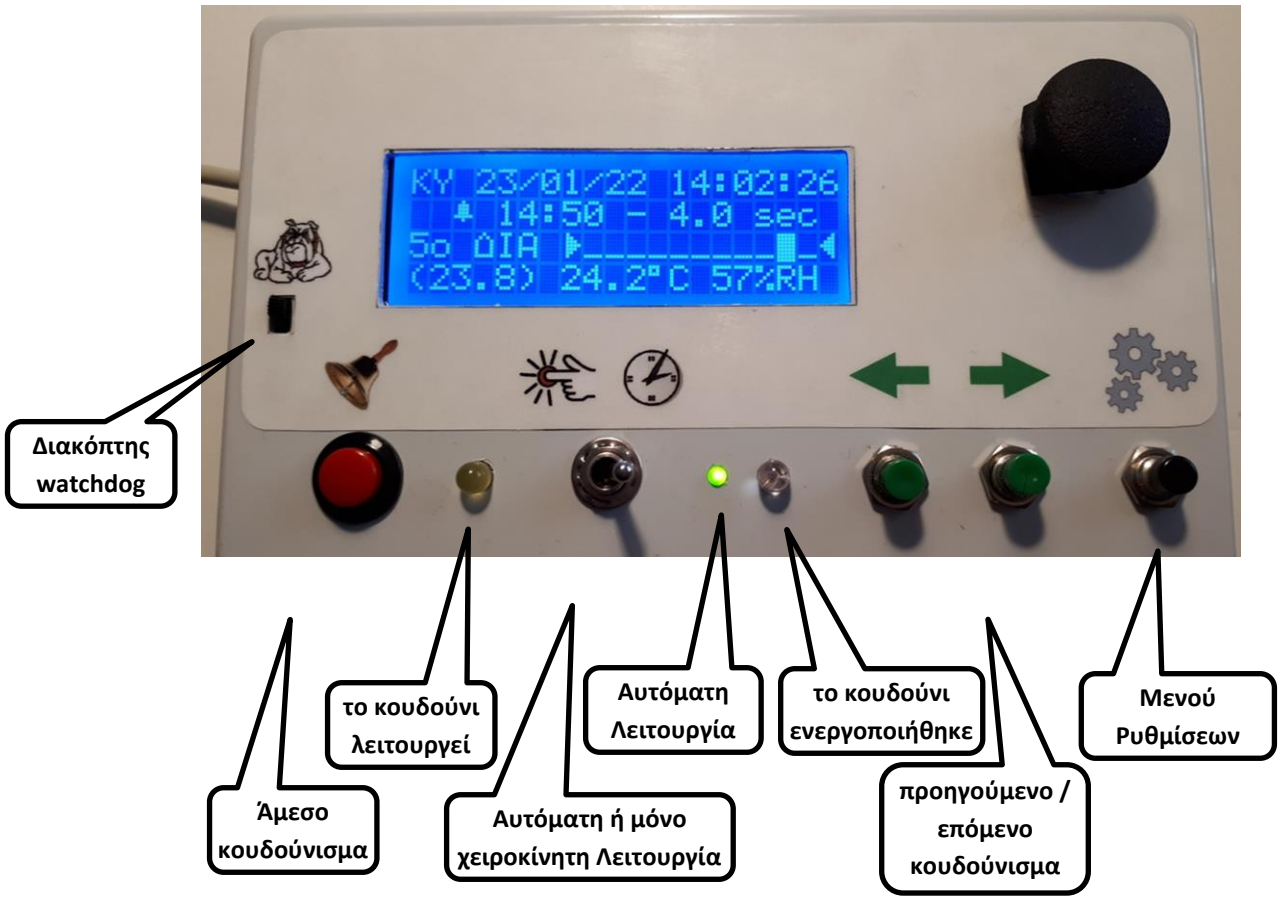
Εικόνα 4.8 : Το κουτί ανοιχτό

Στην κατασκευή ενσωματώθηκαν ένα επιπλέον μπουτόν και 2 διακόπτες. Ένας μικρός για τη λειτουργία του watchdog, ο οποίος διακόπτει αν χρειαστεί την ασφαλή λειτουργία του κυκλώματος και ένας μεγαλύτερος για την επιλογή αυτόματης ή μόνο χειροκίνητης λειτουργίας. Το μπουτόν χειροκίνητης λειτουργίας ενεργοποιεί το κουδούνι βραχυκυκλώνοντας τους ακροδέκτες Source και Drain του FET σε κάθε χρονική στιγμή, ανεξάρτητα της κατάστασης του υπόλοιπου κυκλώματος. Ο μικρός διακόπτης απομονώνει την έξοδο του κυκλώματος του watchdog από την είσοδο reset μικροελεγκτή. Ο μεγαλύτερος διακόπτης απομονώνει την έξοδο του κυκλώματος ενεργοποίησης του κουδουνιού, μετατρέποντας το κύκλωμα σε χειροκίνητο. Επιπλέον, εφόσον ο διακόπτης αυτός είναι διπλός, απομονώνει και το μικρό πράσινο LED φυσιολογικής λειτουργίας, έτσι ώστε να φαίνεται ότι το κύκλωμα αυτόματης ενεργοποίησης του κουδουνιού είναι σε κατάσταση OFF (εικόνα 4.10).

Σχεδιάστηκε και τυπώθηκε αυτοκόλλητη πρόσοψη για το κουτί (εικόνα 4.10), η οποία αφού πλαστικοποιήθηκε κολλήθηκε στην τελική κατασκευή.



Εικόνα 4.9 : Η πρόσοψη της κατασκευής



Εικόνα 4.10 : Η τελική κατασκευή

Εικόνα 4.11 : Αριστερή και δεξιά όψη της κατασκευής

Όπως φαίνεται στην εικόνα 4.11, στην αριστερή πλευρά εκτός από το καλώδιο USB, υπάρχει είσοδος για τα 12 V (στενή υποδοχή) και έξοδος για τη σειρήνα (φαρδύτερη). Δεξιά φαίνεται το αισθητήριο θερμοκρασίας / υγρασίας DHT22.

4.1 Χειρισμός - λειτουργίες

Άμεσο κουδούνισμα - κόκκινο κουμπί: Όταν πιέζεται, το κουδούνι ενεργοποιείται για όση ώρα παραμένει πατημένο, ανεξάρτητα ποια είναι η θέση του διακόπτη αυτόματης / χειροκίνητης λειτουργίας.

Πορτοκαλί LED λειτουργίας κουδουνιού: Ανάβει όταν ενεργοποιείται η σειρήνα, χειροκίνητα ή αυτόματα. Είναι συνδεμένο στο κύκλωμα των 12 V, πολύ κοντά στην έξοδο, μέσω μιας αντίστασης 1 kΩ. Χρησιμεύει, αν δεν έχουμε συνδεμένη τη σειρήνα, για να ελέγξουμε τη λειτουργία του κυκλώματος 12 V.

Διακόπτης αυτόματης ή μόνο χειροκίνητης λειτουργίας: Απομονώνει την έξοδο του κυκλώματος ενεργοποίησης του κουδουνιού η οποία ελέγχεται από τον μικροελεγκτή. Στην κατάσταση αυτόματης λειτουργίας, δεξιά, το κουδούνι ενεργοποιείται στα προκαθορισμένα χρονικά σημεία. Στην κατάσταση χειροκίνητης λειτουργίας, αριστερά, το κουδούνι δεν ενεργοποιείται από τον μικροελεγκτή.

Πράσινο μικρό LED (3 mm): Αναβοσβήνει με περίοδο 2 δευτερόλεπτα σε κατάσταση φυσιολογικής λειτουργίας, αναβοσβήνει με περίοδο 1 δευτερόλεπτο σε κατάσταση λειτουργίας ρύθμισης κουδουνισμάτων και παραμένει σβηστό σε κατάσταση μόνο χειροκίνητης λειτουργίας. Στην κατάσταση αυτή η φυσιολογική λειτουργία του συστήματος φαίνεται μόνο στην οθόνη.

Διαφανές LED λειτουργίας χρονοδιακόπτη: Ανάβει όταν δίνεται σήμα να ενεργοποιηθεί η σειρήνα από τον μικροελεγκτή, ανεξάρτητα αν είναι συνδεμένο το κύκλωμα των 12V. Σε φυσιολογική λειτουργία όταν ενεργοποιείται το κουδούνι από το χρονοδιακόπτη, ανάβουν και τα δύο LED (πορτοκαλί και διαφανές).

Αριστερό - Δεξί πράσινα κουμπιά: Όταν βρισκόμαστε σε κατάσταση λειτουργίας ρύθμισης πηγαίνει στο προηγούμενο / επόμενο κουδούνισμα. Σε φυσιολογική λειτουργία, καμία επίδραση στο σύστημα.

Joystick : Όταν βρισκόμαστε σε κατάσταση λειτουργίας ρύθμισης, αύξηση / μείωση χρόνου / διάρκειας κουδουνίσματος. Πάνω / Κάτω : αύξηση / μείωση. Αριστερά - Δεξιά : επιλογή χρόνου ή διάρκειας κουδουνίσματος. Σε φυσιολογική λειτουργία, καμία επίδραση στο σύστημα.

Μαύρο κουμπί : Είσοδος σε κατάσταση λειτουργίας ρύθμισης / επιστροφή σε φυσιολογική λειτουργία. Εάν για 10 δευτερόλεπτα δεν ενεργοποιηθούν τα πράσινα κουμπιά ή το Joystick, το σύστημα επανέρχεται σε κατάσταση φυσιολογικής λειτουργίας.

4.2 Προβλήματα - Αβλεψίες

Υπήρξε πρόβλημα κατά τη συγκόλληση των καλωδίων του κυκλώματος ισχύος. Χρησιμοποιώντας περισσότερη ισχύ στο κολλητήρι ή μεγαλύτερους χρόνους είχε σαν αποτέλεσμα να καταστραφεί ένας λεπτός αγωγός πολύ κοντά στο τμήμα ισχύος του κυκλώματος, ο οποίος αποκαταστάθηκε με ένα λεπτό αγωγό από πολύκλωνο καλώδιο (εικόνα 4.2.1).

Εικόνα 4.2.1 : Αποκατάσταση κατεστραμμένης λεπτής όδευσης

Οι κολλήσεις στο κύκλωμα των 12 V ήταν γενικά προβληματικές. Κάποιο τμήμα (το πιο κοντό στην παραπάνω εικόνα) χρειάστηκε να αφαιρεθεί τελείως και να κολληθούν τα τρία καλώδια τα οποία κατέληγαν σ' αυτό, εκτός πλακέτας. Ο λόγος ήταν ότι με την κίνηση του πάνω μέρους των καλωδίων κατά το κλείσιμο του κουτιού, δημιουργούνταν βραχυκύκλωμα. Σε μια πιο λεπτομερή σχεδίαση θα ήταν καλό να απομακρυνθούν τα κυκλώματα watchdog και ισχύος και να δοθεί περισσότερος χώρος στους ακροδέκτες ισχύος για τις κολλήσεις.

Το μέγεθος του κουτιού δεν ήταν ικανοποιητικό για την άνετη τακτοποίηση των καλωδίων. Θα μπορούσε να έχει επιλεγεί μεγαλύτερο κουτί.

Η επικοινωνία με τον υπολογιστή ήταν από προβληματική έως ανύπαρκτη με την έξοδο του watchdog συνδεμένη στη είσοδο Reset του μικροελεγκτή. Γι' αυτό το λόγο τοποθετήθηκε και ο διακόπτης απομόνωσης του watchdog.

4.3 Πρόγραμμα - Λειτουργίες

Οι βασικές παραδοχές του προγράμματος είναι:

- Αριθμός κουδουνισμάτων: Πρακτικά απεριόριστος (εξαρτάται από τη μνήμη του μικροελεγκτή)
- Διάρκεια κουδουνίσματος: 0 έως 9.9 seconds
- Χαρακτηρισμός περιόδων: έως 9 Διδακτικές ώρες και 9 διαλείμματα
- Μονάδα χρόνου: 100 ms. Εξηγείται παρακάτω.

4.3.1 Ρύθμιση διαλειμμάτων

Η ρύθμιση των διαλειμμάτων γίνεται μέσα στον κώδικα, στην αρχή του προγράμματος. Τα στοιχεία τα οποία χρειάζονται είναι τα εξής:

- ώρα έναρξης
- λεπτό έναρξης
- διάρκεια κουδουνίσματος σε δέκατα του δευτερολέπτου
- ένας αριθμός ο οποίος αντιπροσωπεύει τον αύξοντα αριθμό της ώρας ή του διαλείμματος. Αν ο αριθμός αυτός είναι από 1 έως 9, π.χ. 4, αντιστοιχεί σε διδακτική ώρα και εμφανίζεται στην οθόνη LCD το μήνυμα **4η ωρα**. Αν είναι πάνω από 10, π.χ. 25, το τελευταίο ψηφίο αντιστοιχεί σε διάλειμμα και εμφανίζεται στην οθόνη LCD το μήνυμα **5ο ΔΙΑ**. Αν ο αριθμός είναι 0, δεν εμφανίζεται κάποιο μήνυμα στην οθόνη LCD.

4.3.2 Ενημέρωση του Real Time Clock

Το ρολόι του συστήματος ενημερώνεται μέσω μιας εντολής define, στην αρχή του προγράμματος.

```
#define read_PC_time 0 /// if 1, set compiling time to RTC
```

Όποτε αλλάζει η ώρα του συστήματος (π.χ. Αντικατάσταση μπαταρίας του RTC, χειμερινή/θερινή ώρα), η τιμή αυτή πρέπει να γίνεται 1. Στην περίπτωση αυτή, το RTC παίρνει την ώρα μεταγλώττισης του προγράμματος, η οποία δε διαφέρει από την ώρα του υπολογιστή πάνω από 10 δευτερόλεπτα, που είναι ο χρόνος ο οποίος μεσολαβεί από την μεταγλώττιση έως τη μεταφόρτωση του προγράμματος. Οι λεπτομέρειες οι οποίες πρέπει να προσεχθούν είναι

- Η ώρα του υπολογιστή να είναι σωστή, το οποίο κατά κανόνα συμβαίνει μέσω του αυτόματου συγχρονισμού ώρας στα windows 10.
- Το πρόγραμμα να ξαναμετταγλωττιστεί με τιμή 0 στην παραπάνω εντολή και να μεταφορτωθεί εκ νέου. Αν δεν γίνει αυτό, στην επόμενη επανεκκίνηση, το σύστημα θα ξαναπάρει την ώρα μεταγλώττισης και θα ενημερώσει εσφαλμένα και πάλι το RTC στην (λανθασμένη) τιμή της προηγούμενης μεταγλώττισης

Το σύστημα δεν ενεργοποιεί το κουδούνι, όταν η ημέρα της εβδομάδας είναι Σάββατο ή Κυριακή ή όταν οι μήνες είναι Ιούλιος ή Αύγουστος. Περίοδοι όπως οι διακοπές των Χριστουγέννων / Πάσχα ή άλλες αργίες δεν περιελήφθησαν στον έλεγχο, παρότι τα Χριστούγεννα αφορούν συγκεκριμένες ημερομηνίες και το Πασχάλιο είναι διαθέσιμο για τα επόμενα 30 χρόνια τουλάχιστον. Ο λόγος είναι η ελαστικοποίηση των ημερομηνιών έναρξης / λήξης μαθημάτων, όπως και η χρήση διαφόρων σχολικών αργιών για αναπληρώσεις μαθημάτων, κάτι που κάνει το μακροπρόθεσμο προγραμματισμό επισφαλή. Για αποτροπή λειτουργίας προτείνεται η μετακίνηση του διακόπτη χειροκίνητης/αυτόματης λειτουργίας σε μόνο χειροκίνητη θέση, η αποσύνδεση της σειρήνας από το κουτί ή η πλήρης απενεργοποίηση του συστήματος.

4.3.3 Ρύθμιση ώρας καθημερινής επανεκκίνησης

Η επανεκκίνηση του προγράμματος γίνεται με τη χρήση του ενσωματωμένου watchdog του Arduino που ενεργοποιείται προσωρινά εφόσον έχει απενεργοποιηθεί στην αρχή του προγράμματος. Η ώρα ρυθμίζεται με τη δήλωση :

```
#define REBOOT_TIME "00:05:" //as a string. Reboot between time+":00"-":59"
```

Όπως έχει ήδη αναφερθεί, η τακτική επανεκκίνηση όλων των υπολογιστικών συστημάτων λύνει ένα μεγάλο ποσοστό από προβλήματα τα οποία δημιουργούνται κατά τη μακρόχρονη λειτουργία τους. Εδώ η επανεκκίνηση επιλέχθηκε να γίνεται κάθε μέρα. Προτιμητέα ώρα κάποια μεταμεσονύχτια, χωρίς αυτό να είναι ιδιαίτερα καθοριστικό.

4.3.4 Μονάδα χρόνου : 100 ms

Κάθε 10 ms, αντιστρέφεται το σήμα το οποίο αποστέλλεται στον ακροδέκτη watchdog. Η περίοδος του σήματος αυτού είναι επομένως 20 ms (f=50 Hz). Κάθε 10 φορές (100 ms) καλείται η αντίστοιχη συνάρτηση που το πρόγραμμα ελέγχει τις εισόδους και ανανεώνει όσες χρειάζεται από τις εξόδους. Μονάδα χρόνου επομένως, είναι τα 100 ms. Κάθε πλήκτρο θα πρέπει να παραμείνει πατημένο τουλάχιστον για αυτόν τον χρόνο για να γίνει αντιληπτό.

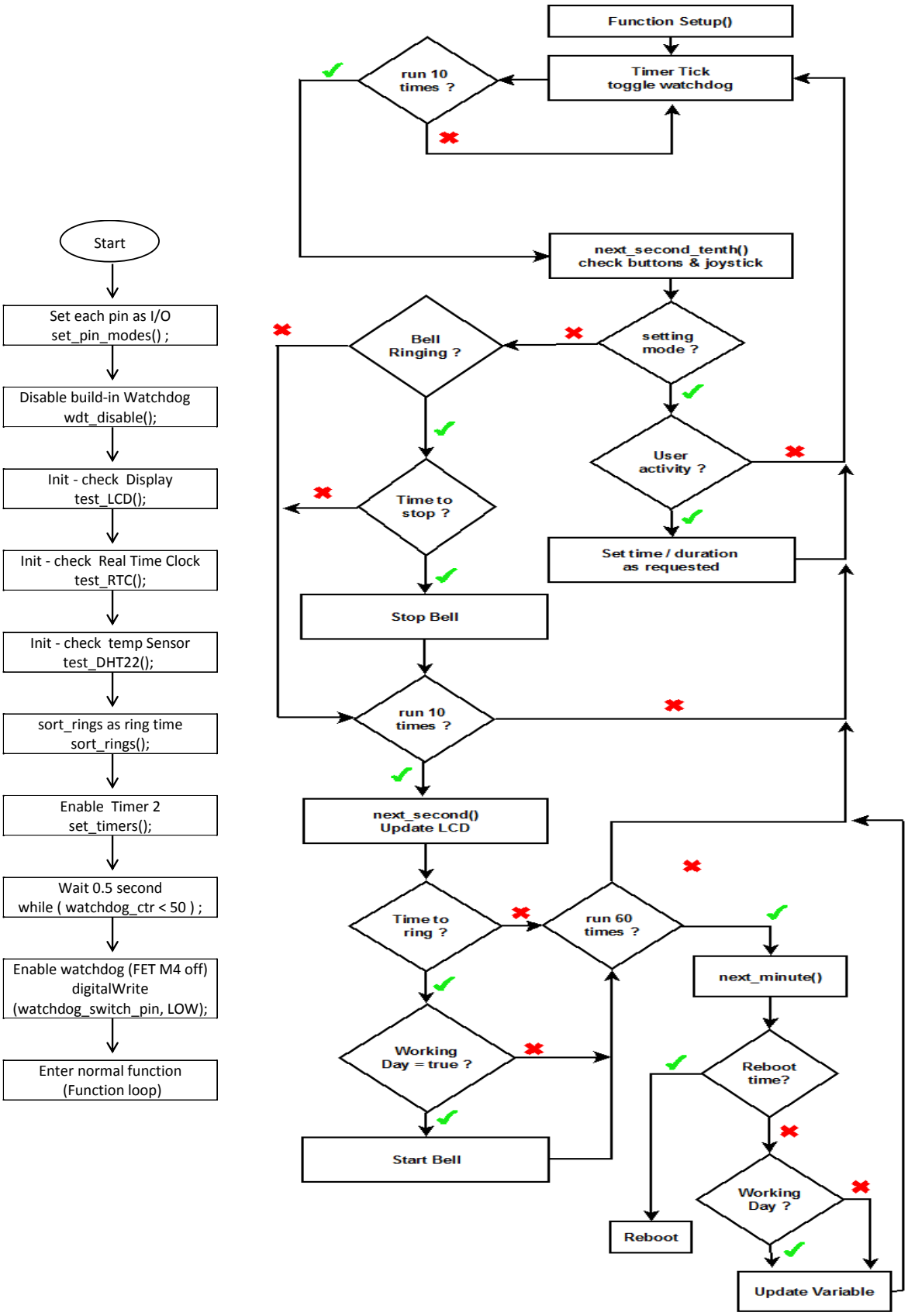
Το πρόγραμμα βασίζει τη λειτουργία του στη συνάρτηση η οποία εξυπηρετεί τον timer2, κάθε 10 ms (f=100 Hz). Η συνάρτηση αυτή, το μόνο που κάνει είναι να αντιστρέφει πάντα το watchdog και να κάνει αληθή μια λογική μεταβλητή (new_second_tenth) κάθε 10 φορές που εκτελείται. Η κύρια συνάρτηση του προγράμματος είναι πολύ απλή:

```
void loop()
{
  if ( new_second_tenth )   { next_second_tenth(); }
}
```

Ουσιαστικά ελέγχει αν έχει γίνει αληθής η λογική μεταβλητή και καλεί τη συνάρτηση εξυπηρέτησής της. Αυτό συμβαίνει κάθε 1/10 του δευτερολέπτου.

4.3.5 Λογικό Διάγραμμα

Το απλοποιημένο λογικό διάγραμμα του προγράμματος παρουσιάζεται παρακάτω, για τη συνάρτηση αρχικοποίησης (Setup, αριστερό τμήμα της εικόνας 4.3.1) και σε φυσιολογική λειτουργία (δεξιό τμήμα). Για τη συνάρτηση Setup παρατίθεται η αντίστοιχη συνάρτηση ή εντολή κάθε φορά. Έχουν παραληφθεί μετρητές και άλλες βοηθητικές μεταβλητές ένδειξης κατάστασης. Επίσης, το λογικό διάγραμμα έχει αρχή αλλά δεν έχει τέλος εφόσον το πρόγραμμα τρέχει συνεχώς, κάτι που συμβαίνει σε όλα τα συστήματα αυτού του τύπου (μικροελεγκτές, PLC κ.τ.λ.).



Εικόνα 4.3.1 : Λογικό Διάγραμμα

4.4 Έξοδος του προγράμματος

Στην οθόνη LCD εμφανίζονται οι εξής πληροφορίες

1η σειρά: Τρέχουσα Ημερομηνία και ώρα, σε πλήρη μορφή : Ημέρα εβδομάδα (2 χαρακτήρες), ημερομηνία (ηη/μμ/εε), ώρα (ωω:λλ:δδ). Ανανεώνεται κάθε δευτερόλεπτο.

2η σειρά: Η ώρα έναρξης (ωω:λλ) και η διάρκεια (δευτερόλεπτα) του επόμενου κουδουνίσματος. Ανανεώνεται κάθε λεπτό.

3η σειρά: Αν υπάρχει αρίθμηση ώρας ή διαλείμματος, εμφανίζεται η αντίστοιχη ένδειξη. Υπενθυμίζεται ότι αριθμός από 1 έως 9 στις ρυθμίσεις αντιστοιχεί σε διδακτική ώρα, ενώ αριθμός πάνω από 10 αντιστοιχεί το τελευταίο ψηφίο του σε διάλειμμα, και ο αριθμός 0 σημαίνει ότι δεν υπάρχει χαρακτηρισμός. Μετά την ένδειξη, αν υπάρχει η αρίθμηση, ενδεικτική μπάρα εμφανίζει κατά προσέγγιση την τωρινή χρονική στιγμή σε σχέση με τη διάρκεια της τρέχουσας χρονικής περιόδου. Αν π.χ. είμαστε περίπου στη μέση του 2ου διαλείμματος η ένδειξη θα είναι : `2ο ΔΙΑ >____▌___<`. Αν δεν υπάρχει αρίθμηση της περιόδου που διανύεται (είναι 0) όλη η σειρά είναι κενή. Ο τύπος που χρησιμοποιείται για τον υπολογισμό της θέσης του χαρακτήρα ένδειξης χρόνου είναι :

$$\text{round}(11 \cdot \frac{\text{now}-\text{prevring}}{\text{nextring}-\text{prevring}}) \qquad 4.4\text{-}1$$

όπου now, prevring και nextring είναι οι χρονικές στιγμές τώρα, του προηγούμενου και του επόμενου κουδουνίσματος. Οι διαφορές είναι σε λεπτά. Το 11 είναι ο χώρος σε χαρακτήρες που διατίθεται (μήκος της μπάρας). Ανανεώνεται κάθε λεπτό.

4η σειρά: Εμφανίζεται η θερμοκρασία στο εσωτερικό του κουτιού, η οποία διαβάζεται από το θερμόμετρο που είναι ενσωματωμένο στο Real Time Clock DS3231 μέσα σε παρενθέσεις. Κατόπιν εμφανίζεται η θερμοκρασία από τον αισθητήρα DHT 22 και η σχετική υγρασία. Οι δυο διαφορετικές θερμοκρασίες που εμφανίζονται, έχουν σκοπό αφενός να δείξουν τυχόν υπερθερμάνσεις μέσα στο κουτί, αφετέρου να δείξουν στιγμιαίες υπερθερμάνσεις του FET ισχύος σε σχέση με το κουτί και το εξωτερικό περιβάλλον. Τονίζεται ότι ο αισθητήρας DHT 22 είναι τοποθετημένος πολύ κοντά στο FET ισχύος. Ανανεώνεται κάθε δευτερόλεπτο.

Στην εικόνα 4.4.1 φαίνεται η έξοδος του συστήματος σε φυσιολογική λειτουργία. Βρισκόμαστε στην αρχή του 1ου διαλείμματος. Το μικρό πράσινο LED αναβοσβήνει με περίοδο 2 δευτερόλεπτα.

Εικόνα 4.4.1 : Φυσιολογική λειτουργία (αρχή 1ου διαλείμματος)

Σε κατάσταση ρύθμισης δεν εμφανίζεται η τελευταία γραμμή (θερμοκρασία / υγρασία) και αλλάζει η 2η γραμμή. Εμφανίζονται :

- Ο αύξων αριθμός του κουδουνίσματος.
- Η ώρα και η διάρκεια του κουδουνίσματος.
- Ο χαρακτηρισμός της περιόδου, αν υπάρχει.

Με κίνηση του joystick δεξιά-αριστερά εναλλασσόμαστε από τη ρύθμιση χρόνου έναρξης στη ρύθμιση διάρκειας κουδουνίσματος. Με κίνηση του joystick πάνω-κάτω αυξάνουμε ή μειώνουμε τους χρόνους έναρξης ή διάρκειας κουδουνίσματος. Με το δεξί ή το αριστερό πράσινο πλήκτρο πηγαίνουμε στο επόμενο ή προηγούμενο κουδούνισμα. Τονίζεται ότι η διάρκεια του κουδουνίσματος μπορεί να είναι από 0 έως 9.9 δευτερόλεπτα.

Το μικρό πράσινο LED αναβοσβήνει με περίοδο 1 δευτερόλεπτο. Οι ενδείκτες (βέλη) του πεδίου που αλλάζει αναβοσβήνουν με την ίδια περίοδο. Αν δεν υπάρξει καμία είσοδος από το χρήστη, το σύστημα μετά από 10 δευτερόλεπτα περνά στην κατάσταση φυσιολογικής λειτουργίας.

Παρακάτω φαίνεται η ρύθμιση του χρόνου έναρξης (εικόνα 4.4.2) και διάρκειας (εικόνα 4.4.3) του 12ου κουδουνίσματος (1ο διάλειμμα).

Εικόνα 4.4.2 : Ρύθμιση ώρας κουδουνίσματος

Εικόνα 4.4.3 : Ρύθμιση διάρκειας κουδουνίσματος

Όταν το κουδούνι ηχεί, στον αυτόματο τρόπο λειτουργίας, ανάβουν και τα δύο μεγάλα LED: Το δεξί - διαφανές, σημαίνει ότι δίνεται σήμα ενεργοποίησης από το χρονόμετρο, και το αριστερό - πορτοκαλί ότι υπάρχει τάση στο κύκλωμα 12V (εικόνα 4.4.4)

Εικόνα 4.4.4 : Κουδούνισμα με αυτόματο τρόπο

Αντίθετα, όταν ενεργοποιούμε το κουδούνι με το χειροκίνητο τρόπο, ενεργοποιείται μόνο το αριστερό μεγάλο LED (πορτοκαλί), το οποίο ελέγχεται από το κύκλωμα 12 V (εικόνα 4.4.5)

Εικόνα 4.4.5 : Κουδούνισμα με χειροκίνητο τρόπο

Κεφάλαιο 5ο: Εγκατάσταση - Παραμετροποίηση

5.1 Εγκατάσταση - Ρύθμιση

Το κύκλωμα θα πρέπει να βρίσκεται στο γραφείο του Διευθυντή ή σε κάποιο άλλο ελεγχόμενο σημείο. Θα πρέπει επίσης να βρίσκεται κοντά σε πολύμπριζο και κοντά σε υπολογιστή, σταθερό - κατά προτίμηση - ή φορητό. Στον υπολογιστή θα βρίσκεται εγκαταστημένο το περιβάλλον προγραμματισμού Arduino IDE, που υπάρχει σε πάρα πολλούς υπολογιστές σήμερα στα σχολεία. Η σειρήνα θα πρέπει να τοποθετηθεί σε τέτοια θέση, ώστε αφενός να μην είναι εύκολα προσβάσιμη από τους μαθητές, αφετέρου να ακούγεται καλά με χρήση εξωτερικού ανακλαστήρα ή ενός τοίχου ή άλλου εμποδίου στη θέση του (οι πιεζοηλεκτρικές σειρήνες στηρίζονται στην ενίσχυση στάσιμων ηχητικών κυμάτων τα οποία δημιουργούνται ανάμεσα στο παλλόμενο πιεζοηλεκτρικό στοιχείο και ένα φυσικό εμπόδιο). Κατόπιν, μέσω του καλωδίου της, των 12V θα πρέπει να συνδεθεί στο σύστημα στην αντίστοιχη υποδοχή - έξοδο των 12V.

Όταν θέλουμε να αλλάξουμε τις ρυθμίσεις του προγράμματος, αποσυνδέουμε τη συσκευή από το φορτιστή των 5V και τη συνδέουμε σε μια θύρα USB του υπολογιστή. Αφού ο υπολογιστής αναγνωρίσει τη θύρα, κάνουμε όλες τις ρυθμίσεις οι οποίες χρειάζεται για μόνιμη αλλαγή των ωρών κουδουνίσματος. Η πιο συνηθισμένη είναι η αλλαγή θερινής / χειμερινής ώρας , που γίνεται με τη μεταφορά της ώρας του υπολογιστή στο Real Time Clock της συσκευής. Όταν η ρύθμιση τελειώσει αποσυνδέουμε το σύστημα από τον υπολογιστή και το ξανασυνδέουμε στο φορτιστή για κανονική λειτουργία.

5.2 Υποστήριξη

Από τη στιγμή που το κύκλωμα θα δουλεύει, η απομακρυσμένη υποστήριξή του είναι αρκετά απλή. Χρειάζεται μόνο κάποιο λογισμικό απομακρυσμένης διαχείρισης στον υπολογιστή που συνδέεται με το σύστημα και, κατά προτίμηση, μια κάμερα.

Οι εκπαιδευτικοί σήμερα έχουν την απαραίτητη εξοικείωση με τέτοια συστήματα. Η ρύθμιση επομένως θα γίνει από το προσωπικό του σχολείου, με ελάχιστη έως καθόλου τη δική μας παρέμβαση. Εάν, παρ' όλα αυτά, δεν υπάρχει κάποιος που να νιώθει άνετα με τέτοια συστήματα, του λέμε απλά να συνδέσει τη συσκευή σε μια θύρα USB του υπολογιστή, να εκτελέσει το λογισμικό απομακρυσμένης διαχείρισης και να μας πάρει τηλέφωνο. Σε σπάνιες περιπτώσεις θα χρειαστεί να κατευθύνει την κάμερα στη συσκευή για να διαπιστώσουμε οι ίδιοι τη λειτουργία της από μακριά. Στις περισσότερες όμως περιπτώσεις, τα μηνύματα τα οποία αποστέλλονται στη σειριακή θύρα είναι αρκετά για να καταλάβουμε τυχόν προβλήματα και δυσλειτουργίες. Γενικά, είναι μάλλον απίθανο να χρειαστεί επίσκεψη και επιτόπιος έλεγχος για να λυθούν προβλήματα.

Κεφάλαιο 6ο: Συμπεράσματα - Προτάσεις

6.1 Προστιθέμενη Εκπαιδευτική Αξία

Η κατασκευή η οποία πραγματοποιήθηκε στην παρούσα εργασία, μου έδωσε την ευκαιρία να έρθω σε επαφή με τη μελέτη, σχεδίαση, προγραμματισμό και υλοποίηση ενός αξιόπιστου ενσωματωμένου συστήματος. Αντιμετώπισα αρκετά και διαφορετικά προβλήματα στη φάση της σχεδίασης του κυκλώματος τα οποία επιλύθηκαν με τον καλύτερο δυνατό τρόπο με γνώμονα την αξιοπιστία του τελικού συστήματος. Για τον προγραμματισμό, απαιτήθηκε ενδελεχής μελέτη του τρόπου λειτουργίας του μικροελεγκτή, για να είναι το πρόγραμμα λειτουργικό και αποδοτικό. Λεπτομέρειες του προγράμματος, όπως π.χ. το γεγονός ότι μια μεταβλητή στη C θα έπρεπε να έχει δηλωθεί ως volatile για να εξασφαλιστεί ότι θα διαβάζεται κάθε φορά από την αρχική της θέση αποθήκευσης [13] και όχι από ένα αντίγραφο σε κάποιον καταχωρητή, όπου το περιβάλλον της γλώσσας το έχει σε πρώτη ζήτηση, διασφαλίζουν την ομαλή λειτουργία του προγράμματος αφενός και αφετέρου αναβαθμίζουν την εμπειρία μου σε τέτοια συστήματα. Χρειάστηκε επισταμένη μελέτη ενημερωτικών ιστοσελίδων [14], όπως και των εξειδικευμένων συγγραμμάτων της σχολής ([15], [16], [17]). Επίσης, ο χωρικός διαχωρισμός κυκλωμάτων ελέγχου και ισχύος σε πλακέτες οι οποίες περιλαμβάνουν και τους δύο τύπους βρέθηκε ότι είναι καθοριστικής σημασίας και θα έπρεπε να λαμβάνεται πολύ σοβαρά υπόψη.

6.2 Μελλοντικές Βελτιώσεις

Μελλοντικές βελτιώσεις στο σύστημα θα μπορούσαν να περιλαμβάνουν καταγραφή των κουδουνισμάτων και των τιμών θερμοκρασίας / υγρασίας ακόμη και κατά τη διάρκεια των περιόδων που το σχολείο δε λειτουργεί (π.χ. Χριστούγεννα, καλοκαίρι) και αποστολή τους σε δικτυακούς τόπους παρουσίασης τέτοιων στοιχείων (π.χ. ThinkgSpeak της MathWorks). Σέ αυτήν την περίπτωση θα ήταν καλό να χρησιμοποιήσουμε ένα πιο ισχυρό σύστημα, όπως π.χ. τον ESP32, με αρκετή μνήμη (η μνήμη του Arduino είναι ελάχιστη), ενσωματωμένη δυνατότητα ασύρματης δικτύωσης και πολύ μικρότερες καταναλώσεις. Το συγκεκριμένο σύστημα προγραμματίζεται στο περιβάλλον Arduino IDE, με αποτέλεσμα να μην ξενίζει τον τελικό χρήστη και είναι και αυτό οικονομικό και αξιόπιστο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://en.wikipedia.org/wiki/Edgar_Dale, Cone of Experience. Accessed : 5/12/2021

[2] https://www.singlewire.com/informacast/use-case/school-bells. Accessed : 19/12/2021

[3] https://electronics.stackexchange.com/questions/212312/ac-watchdog-circuit . Accessed : 9/5/2021

[4] https://new.siemens.com/global/en/products/automation/systems/industrial/plc/logo/logo-basic-modules.html. Accessed : 9/1/2022

[5] https://www.instructables.com/Arduino-XMAS-hitcounter/. Accessed : 10/1/2022

[6] https://www.arapi.gr/automato-koudouni-sxoleiou/#. Accessed : 10/1/2022

[7] https://en.wikipedia.org/wiki/Arduino_Uno, Technical specifications. Accessed : 12/1/2022

[8] https://www.espressif.com/en/products/modules/esp32. Accessed : 12/1/2022

[9] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Accessed : 12/1/2022

[10] https://socialcompare.com/en/comparison/raspberrypi-models-comparison. Accessed : 12/1/2022

[11] https://www.st.com/en/microcontrollers-microprocessors/stm32f301c6.html. Accessed : 12/1/2022

[12] https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32mz-ef. Accessed : 12/1/2022

[13] https://www.geeksforgeeks.org/understanding-volatile-qualifier-in-c/. Accessed : 15/6/2021

[14] http://gammon.com.au/interrupts . Accessed : 11/4/2021

[15] Α. Γιακουμής, *Σημειώσεις Μαθήματος Μικροελεγκτές*. Σίνδος: ΔΙΠΑΕ, 2018

[16] Α. Γιακουμής, *Σημειώσεις Μαθήματος Ενσωματωμένα Συστήματα*. Σίνδος: ΔΙΠΑΕ, 2019

[17] Μ. Σπάσος και Κ. Αμοιρίδης, *Σύγχρονες Εφαρμογές Αναλογικών Ηλεκτρονικών*. Θεσσαλονίκη: Αιβάζης, 2015

Παράρτημα Α : Κώδικας Εφαρμογής

```
//// settings

#define green_pin 7     // pin toggled by timer
//#define yellow_pin 9
#define watchdog_pin 8
#define watchdog_switch_pin 9
#define bell_pin 3
#define button1_pin 11  //  next ring button
#define button2_pin 12  //  insert-exit setting menu
#define button3_pin 10   //  previous ring button
#define pot_x_pin A0
#define pot_y_pin A1
#define Large_LCD true // true LCD 20x4, false 16x2
#define seconds_to_wait 10 /// max time in setting menu without any key pressed

#define REBOOT_TIME "00:05:" // + as a string. Reboot between time+":00" - ":59"

#define read_PC_time 0 /// if 1, set compiling time to RTC


#define maxrings 18
byte ring[maxrings][4]  = {
 // Hour, minute, tenths of seconds , hour / break nr
  // 1 - 9: hour nr,  21+ : break nr

  15, 1, 30, 1,
  17, 10, 30, 2,
  18, 0 , 30, 21,
  18, 10, 30, 3,
  18, 50, 30, 22,
  18, 55, 30, 4,
  19, 35, 30, 23,
  19, 40, 30, 5,
  20, 20, 30, 0,
  0, 6, 30, 6,
  0, 18, 30, 7,
  0, 20, 30, 25,
  14, 50, 40, 28,

  0, 5, 40 , 0

};

//////////

#define LIM1 140
#define LIM2 380
const int very_low = LIM1, low = LIM2, high = 1023-LIM2, very_high = 1023-LIM1;
unsigned int setting_max_time = 10 * seconds_to_wait ; /// max nr of timer ticks;
int pot_x, pot_y;  /// variables holding potentiometer's position (0 - 1023)
unsigned int setting_time_counter ;

#include <string.h>

#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display
// usually 0x3F for a 20 chars and 4 line display
//LiquidCrystal_I2C lcd(0x27, 16, 2);
LiquidCrystal_I2C lcd(0x27, 20, 4);

#include <TimeLib.h>
#include <DS3231.h>  // timer
DS3231  rtc(SDA, SCL);  // Init the DS3231 using the hardware interface
```

```
#include <DHT.h>
#define DHTPIN 2     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)
DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

String wkday [7] = { "DE", "TP", "TE", "PE", "PA", "SA", "KY" } ;

byte  next_ring ,  // array index of next ring
      button_state ; // 01 : button 1 down, 02 : button 2 down
volatile byte bell_counter ; // counter of time ringing (10ths of seconds)
volatile unsigned long watchdog_ctr = 0; /// watchdog absolute counter

volatile boolean watchdog;
boolean    //  watchdog_off, /// true at the beginning of the program
        setting_mode ,  // true when setting ring times / duration
        time_change , // true : setting ring time , false : setting ring duration
        button_1_pressed , button_2_pressed , button_3_pressed ,
        green_on , // true when green led is on
        working_day ; // false at Sat, Sun, July and August

int i, j, second_counter, minute_counter ;

String s,  s1, s2 ;

Time  t;

float temp, dhttemp, hmd;

//interrupt variables

boolean toggle2 = 0, tim2_tick ;

#define ticks_per_second_tenth 10 /// 100 ms / 10 ms

//int timer2_counter = timer2_ticks ;
volatile int i_sec_10 ; // = ticks_per_second;
volatile int second_tenth ;
volatile boolean new_second_tenth  ;
//volatile
boolean wdtog = true ; // watchdog toggle
int sec_cntr;

#include <avr/wdt.h>  /// internal watchdog timer used for reboot

  ///// temp/debug variables functions

volatile unsigned long ms, ms2; /// milliseconds used to count rings duration
int btcntr1, btcntr2, btcntr3 ; // button counter

  //////

void setup()
{   set_pin_modes() ;
    wdt_disable(); // disable built in Watchdog
    green_on = false ;

    Serial.begin(9600);  // Initialize the serial port : speed 9600 baud
    Serial.println("\n\n Beginning ..... \n Watchdog OFF ");

    test_LCD();  // delay (1000);
    test_RTC();
    test_DHT22();

    print_rings_to_serial();

    sort_rings();

    lcd.clear();
```

```
    Serial.println(" Enabling Interrupts ..... ");
    set_timers();
    Serial.print("\n End of Init Function.  Turning on Watcdog ..... ");
    while ( watchdog_ctr < 50 ) ; /// wait 500 ms
    digitalWrite(watchdog_switch_pin, LOW); // enable watchdog
    Serial.println(" Watchdog ON \n");

    new_second_tenth = false ;
    button_1_pressed = button_2_pressed = false;
    sec_cntr = 60;
    bell_counter = 0 ;
    button_state = 0 ; // both buttons up
    i_sec_10 = ticks_per_second_tenth ;
    second_tenth = 10 ;
    setting_time_counter = setting_max_time ;

    setting_mode = true; // show rings
    find_next_ring();
    print_2_lcd (0, 1, set_rings_msg( next_ring ), false ); time_change = true ;


    print_2_lcd (0, 2, period_position(), false );

}   //end setup



void loop()
{
  if ( new_second_tenth )   { next_second_tenth(); }
}

void set_timers()
{ cli();//stop interrupts

  TCCR0A = 0;   // set entire TCCR0A register to 0
  TCCR0B = 0;   // same for TCCR0B
  TCNT0  = 0;   //initialize counter value to 0
  // set compare match register for 100 Hz increments
  OCR0A = 155;  // = (16*10^6) / (100*1024) - 1 (must be <256)
  TCCR0A |= (1 << WGM01); // turn on CTC mode
  TCCR0B |= (1 << CS02) | (1 << CS00);  // Set CS02 and CS00 bits for 1024 prescaler
  TIMSK0 |= (1 << OCIE0A); // enable timer compare interrupt

  sei();//allow interrupts
}

ISR(TIMER0_COMPA_vect)//timer2 interrupt 100 Hz
{   watchdog = not (  watchdog ) ;
    digitalWrite( watchdog_pin, ( watchdog ? HIGH : LOW ) ); // toggle watchdog every 10 ms
    watchdog_ctr ++ ; // inc watchdog_ctr
    if ( -- i_sec_10 ) return; // wait for next tenth of second
    i_sec_10 = ticks_per_second_tenth;
    new_second_tenth = true ;
}

void next_second_tenth()
{   new_second_tenth = false ;

    check_buttons();

    if ( button_2_pressed )
    {   button_2_pressed = false;
        setting_mode =  not ( setting_mode ) ;
        print_2_lcd (0, 3, "                    ", false ); //clear last line
        find_next_ring();
        if ( setting_mode )
```

```
        {   digitalWrite(bell_pin, LOW); bell_counter = 0 ; // bell off
            print_2_lcd (0, 1, set_rings_msg( next_ring ), false );
            time_change = true ;
            setting_time_counter = setting_max_time ; /// input activity
        }
        else  // to normal mode
            {   show_next_ring();  print_rings_to_serial();    }
     }



    if ( ! setting_mode ) // normal function
    {   boolean ringing = ( bell_counter > 0 ) ;
        if ( ringing )  // if bell ringing
        {   bell_counter -- ;  // count down
            if ( bell_counter == 0 ) // time to stop ?
            { digitalWrite(bell_pin, LOW);

              ms2 = watchdog_ctr  ; ///
              Serial.print(" Ring Stop : "); Serial.print(time_str());
              Serial.print(" ( ");Serial.print(String( (float)( ms2 - ms )/100.0 , 3 ));
              Serial.println(" sec ) ");

            } // turn off bell
         }
    }
    else /// setting mode
    {   digitalWrite(bell_pin, LOW); bell_counter = 0 ; // bell off

        if ( second_tenth == 5 ) // green blinks twice a second
            { toggle_green();  update_blinking_pointers(green_on); }
        if ( ! ( -- setting_time_counter ) ) // enough setting time ?
            {  button_2_pressed = true; } // exit settings menu
        if ( button_1_pressed || button_3_pressed ) //left or right button pressed
        {   if ( button_1_pressed ) //left button
            {   button_1_pressed = false;
                next_ring ++ ; if ( next_ring == maxrings ) { next_ring = 0 ; }
            }
            if ( button_3_pressed ) // right button
            {   button_3_pressed = false;
                if ( next_ring == 0  ) { next_ring = maxrings ; } next_ring -- ;
            }
            print_2_lcd (0, 1, set_rings_msg( next_ring ), false );
            time_change = true ;
            setting_time_counter = setting_max_time ; /// input activity
       }
        check_joystick();
    }

    if ( -- second_tenth ) { return ; } // wait for next second
    second_tenth = 10 ;
    print_2_lcd (0, 0,  time_str() , false); //update time
    if ( setting_mode ) { toggle_green(); update_blinking_pointers(green_on);  }
    else {  next_second() ;  }
}

void next_second()
{
    toggle_green();
    if ( time_str().indexOf( ring_str( next_ring , false ) + ":00" ) > 0 ) // time to ring
?
    {
        bell_counter = ring[next_ring][2]; // keep duration
        if ( ( working_day ) && ( bell_counter ) )
        digitalWrite( bell_pin, HIGH ); // bell ringing ....

        ms = watchdog_ctr;
        Serial.print("\n\n Ringing : "); Serial.println(time_str());
```

```
    }
    if ( Large_LCD )
    {  hmd = dht.readHumidity();  dhttemp = dht.readTemperature(); temp = rtc.getTemp() ;
       print_2_lcd (0, 3,
          "(" + String(temp, 1) + ") " +  String(dhttemp, 1) + char(223) + "C "
              + String(hmd, 0) + "%RH", false);
    }
    if  ( -- sec_cntr ) return;  // wait for next minute
    sec_cntr = 60 ;  next_minute() ;
}

void next_minute()
{
    Serial.print(".");
    if ( time_str().indexOf(REBOOT_TIME) > 0 ) { reboot() ; return ; }
    working_day = is_working_day(); /// update boolean variable
    show_next_ring();
    print_2_lcd (0, 2, period_position(), false );
}

int check_pot(int val)
{   if ( ( val >= low ) && ( val <= high ) )  {  return( 0 ) ;  }
    if ( val < very_low )   {   return ( -2 ) ; }
    if ( val < low )        {   return ( -1 ) ; }
    if ( val > very_high )  {   return ( 2 ) ;  }
    if ( val > high )       {   return ( 1 ) ;  }
}

void check_joystick()
{
    pot_x = check_pot( analogRead( pot_x_pin ) );
    pot_y = check_pot( analogRead( pot_y_pin ) );
    if ( ! ( pot_x | pot_y ) ) // both zero
        {
          print_2_lcd (0, 3, "         " , false);
          return ; }
    setting_time_counter = setting_max_time ; /// input activity
    if ( time_change )
    { if ( pot_x == 1 ) {  if ( !( watchdog_ctr & 0x07 ) ) {inc_time();} } //every 8 ticks
(80 ms)
      if ( pot_x == 2 ) {  inc_time();  inc_time(); inc_time();  inc_time();}
      if ( pot_x == -1 ) { if ( !( watchdog_ctr & 0x07 ) ) {dec_time();} } //every 8 ticks
(80 ms)
      if ( pot_x == -2 ) { dec_time(); dec_time(); dec_time(); dec_time(); }
      print_2_lcd (3, 1, ring_str( next_ring, false ), false );
    }
    else // duration change
    { if ( pot_x == 1 )
          {  if ( !( watchdog_ctr & 0x07 ) ) // once every 80 ms
              {  if ( ring[next_ring][2] < 99 ) { ring[next_ring][2] ++ ; } } }
      if ( pot_x == 2 )
          {  if ( ring[next_ring][2] < 99 ) { ring[next_ring][2] ++ ; } }
      if ( pot_x == -1 )
          {  if ( !( watchdog_ctr & 0x07 ) )
              {  if ( ring[next_ring][2] > 0 )  { ring[next_ring][2] -- ; } } }
      if ( pot_x == -2 )
          {  if ( ring[next_ring][2] > 0 )  { ring[next_ring][2] -- ; } }
      print_2_lcd (9, 1, String(ring[next_ring][2]/10.0, 1), false );
    }

    if ( ( pot_y == 2 ) || ( pot_y == -2 ) )
        {  if ( !( watchdog_ctr & 0x07 ) ) {  time_change = not time_change ; } }
}

void check_buttons()
{   byte b = digitalRead(button1_pin);
    if ( ! b ) //  button 1 down
```

```
    {   if ( !( button_state & 0x01 ) ) { button_1_pressed = true; } // 1st time button
pressed
        button_state |= 0x01 ;  // set LSB bit 1
    }
    else  {   button_state &= ~(0x01) ;   } // button 1 up, set LSB bit 0
    b = digitalRead(button2_pin);
    if ( ! b ) //  button 2 down
    {   if ( !( button_state & 0x02 ) ) { button_2_pressed = true; } // 1st time button
pressed
        button_state |= 0x02 ;  // set 2nd LSB bit 1
    }
    else  {   button_state &= ~(0x02) ;   } // button 2 up, set 2nd LSB bit 0
    b = digitalRead(button3_pin);
    if ( ! b ) //  button 3 down
    {   if ( !( button_state & 0x04 ) ) { button_3_pressed = true; } // 1st time button
pressed
        button_state |= 0x04 ;  // set 2nd LSB bit 1
    }
    else  {   button_state &= ~(0x04) ;   } // button 3 up, set 3nd LSB bit 0
}

String set_rings_msg(int i)
{   String msg = nr2digits( i + 1 ) + " " + ring_str( i, false );
    msg += " " + String(ring[i][2]/10.0, 1) + " sec ";
    if ( Large_LCD ) // 20x4 LCD
    {   if ( ring[i][3] )
        {   msg += String( ring[i][3] % 10 );
            msg += ( ring[i][3] > 9 ? char(6) : char(244) );
        }  else { msg += "  " ; }
    }

//Serial.println(">>"+msg+"<<");

    return msg ;
}

void toggle_green()
    { green_on = not(green_on) ; digitalWrite( green_pin, ( green_on ? HIGH : LOW ) );  }

void update_blinking_pointers(boolean show)
{   String left,right;
    left = right = " ";
    right[0] = char(0); left[0] = char(1);
    if ( ! show )
    {   print_2_lcd (2,1," ",false); print_2_lcd (8,1," ",false);
        print_2_lcd (8,1," ",false); print_2_lcd (12,1," ",false);
    }
    else  // pointers on
      if ( time_change ) { print_2_lcd (2,1,right,false);
print_2_lcd (8,1,left,false); }
      else {  print_2_lcd (8,1,right,false); print_2_lcd (12,1,left,false); } // set
duration
}

void set_pin_modes()
{ pinMode(watchdog_switch_pin, OUTPUT);
  digitalWrite(watchdog_switch_pin, HIGH); // bypass watchdog

  pinMode(bell_pin, OUTPUT); digitalWrite(bell_pin, LOW); // turn off bell

  pinMode(pot_x_pin, INPUT);   pinMode(pot_y_pin, INPUT); // Analogs
  pinMode(button1_pin, INPUT); pinMode(button2_pin, INPUT); pinMode(button3_pin, INPUT);

  pinMode(green_pin, OUTPUT);
  pinMode(watchdog_pin, OUTPUT);

  for ( i = 0 ; i < 10 ; i ++ ) { toggle_green();  delay(200); } // 5 blinks
  digitalWrite(green_pin, LOW);
```

```
}

void reboot()
{   Serial.println(" Restarting  \n");
    print_2_lcd (4, 1, "Restarting...  "  , true);
    wdt_enable(WDTO_15MS); MCUSR = 0; //for (;;) ;
}

void show_next_ring()
{   find_next_ring();
    String msg = "  "; msg += char(5); msg += " " + ring_str ( next_ring , true ) + " sec";
    print_2_lcd ( ( Large_LCD ? 0 : 2 ), 1, msg , false );
}

String period_position()
{   String s = "";  int i,p_ring, hr, mins, mins_now ;

    if ( next_ring == 1 ) return("                    ");
    i = next_ring ; hr = 0;
    do
    { if ( i == 0 ) { i = maxrings ; hr += 24 ; }
      else { i -- ; }  // find previous ring
      if ( hr >= 48 ) { return("                    "); } // kenos pinakas
    } while ( ring[i][2] == 0 ) ;
    p_ring = i ; // Previous Ring
    if ( ring[p_ring][3] == 0 ) return("                    ");
    mins = ( ring[next_ring][0]-ring[p_ring][0] ) * 60 + ( ring[next_ring][1]-
ring[p_ring][1] );
        ///  minutes between previous and next ring
    t = rtc.getTime();
    mins_now = ( t.hour - ring[p_ring][0] ) * 60 + ( t.min-ring[p_ring][1] );
        ///  minutes between previous ring and now

    i = ring[p_ring][3] ;
    s = String ( i % 10 ) + "h Wra";
    if ( i > 9 )  { s[1]= 'o' ; s[3] = char(6); s[4] ='I'; s[5] ='A'; }  // o DIA
    else  { s [1] = 'n'; s[3] = char(244); s[4] = char(230); s[5] = char(224); } // n wra
    s += " _____________" ; s[7] = char(0); s[19] = char(1); // left, right arrows

    i = 8 + round( 11 * mins_now / mins );  /// Range : 8 - 18
    s[i] = char(4); // bar

    return (s);
}

void inc_time()
{   ring[next_ring][1] ++ ; // increase minutes
    if ( ring[next_ring][1] == 60 )
    {   ring[next_ring][1] = 0 ; ring[next_ring][0] ++ ; }
    if ( ring[next_ring][0] == 24 ) { ring[next_ring][0] = 0 ; }
}

void dec_time()
{   if ( ring[next_ring][1] ) { ring[next_ring][1] -- ; return; }// if min>0 decrease
minutes
    ring[next_ring][1] = 59 ; // else minutes = 00
    if ( ring[next_ring][0] ) { ring[next_ring][0] -- ; return; }// if hr>0 decrease hours
    ring[next_ring][0] = 23 ;  // else hour = 00
}

void sort_rings()
{   int i,j ;  byte r0, r1, r2, r3 ;
    boolean swap ; /// if no change, then sorted
    for ( i = 1 ; i < maxrings ; i ++ )   /// bubblesort
    {   swap = false ;
          for ( j =  maxrings - 1 ; j >= i ; j -- )
              if ( ring[j-1][0]*100 + ring[j-1][1] > ring[j][0]*100 + ring[j][1] )
              {    /// compare hr * 100 + min
```

```
                swap = true;
                r0 = ring[j-1][0] ; r1 = ring[j-1][1] ; r2 = ring[j-1][2] ; r3 = ring[j-
1][3] ;
                ring[j-1][0] = ring[j][0]; ring[j-1][1] = ring[j][1];
                ring[j-1][2] = ring[j][2]; ring[j-1][3] = ring[j][3];
                ring[j][0] = r0 ; ring[j][1] = r1 ; ring[j][2] = r2 ; ring[j][3] = r3 ;
              }
          if ( ! swap ) return ; // no change : sorted
    }
}

String ring_line_2_serial(int j)
{   String s ;
    s = nr2digits(j) + " : " + nr2digits(ring[j][0]) + ":";
    s += nr2digits(ring[j][1]) + ", " + String ( (float)ring[j][2] / 10 , 1 ) + " sec";
    if ( ring[j][3] )
    {   s += ",  " + String ( ring[j][3] % 10 ) ;
        s += ( ring[j][3] > 9 ? "o Dialeimma" : "n Wra" );
    }
    return (s);
}

void print_rings_to_serial()
{
    for ( i = 0 ; i < maxrings ; i ++ )
    {
        Serial.println( ring_line_2_serial(i) );
    }
    i = next_ring;
    // s = " Next : ";
    //+ nr2digits(i) + " : " + nr2digits(ring[i][0]) + ":";
    //s += nr2digits(ring[i][1]) + ", " + String ( (float)ring[i][2] / 10 , 1 );
    Serial.println(" Next : " + ring_line_2_serial(i));
    Serial.println("---------------");
}

void  find_next_ring()
{ String time_now = rtc.getTimeStr();
  int i = 0;
  sort_rings();
  while ( i < maxrings )
    if ( ring_str( i ++ , false )+":00" > time_now )  { i -- ;  break ;  }
  if ( i == maxrings ) i = 0 ; // ring not found
   //ringing = false ;
  next_ring =  i  ;
}

String ring_str(int nr, bool duration)
{ String rs ;
  rs = nr2digits(ring[nr][0]) + ":" + nr2digits(ring[nr][1]) ;
  if  ( duration ) rs += " - " + String(ring[nr][2]/10.0, 1);
  return (rs);
}

void test_LCD()
{ Serial.println("\n\n Testing LCD ..... ");
  lcd.begin();  lcd.backlight(); lcd.clear();
  create_characters();
  if ( Large_LCD ) { display_characters(); }
}

void test_RTC()
{ Serial.println(" Testing RTC ..... ");
  rtc.begin();
  if ( read_PC_time )
  { Serial.println(" Setting PC time to RTC ..... ");
    set_PC_time_2_rtc();
  }
```

```
  Serial.print(" Reading time from RTC ..... ");
  Serial.println(time_str());
}

void test_DHT22()
{ Serial.print(" Testing DHT 22 Sensor ..... ");
  dhttemp = hmd = 51.1;
  dht.begin();
  hmd = dht.readHumidity();   dhttemp = dht.readTemperature();
  s = String(dhttemp, 2) + "°C, " + String(hmd, 2) + "% RH" ;
  Serial.println(s);
}
boolean is_working_day() /// -- > working_day boolean variable
{
  if ( rtc.getTime().dow > 5 ) { return false; }
  if ( ( rtc.getTime().mon == 7 ) || ( rtc.getTime().mon == 8 ) )
      { return false ; }   // July  - August
  return true ;
}

void set_PC_time_2_rtc()
{ String month_names = "JanFebMarAprMayJunJulAugSepOctNovDec";
  String PCstr;
  int i, j, k;
  PCstr = __TIME__ ; // Time the program was compiled
  i = PCstr.substring( 0, 2 ).toInt(); // hour
  j = PCstr.substring( 3, 5 ).toInt(); // min
  k = PCstr.substring( 6, 8 ).toInt(); // sec
  rtc.setTime(i, j, k);
  PCstr = __DATE__ ; // Date the program was compiled
  j = PCstr.substring( 4, 6 ).toInt(); // day
  k = PCstr.substring( 7, 11 ).toInt(); // year
  PCstr = PCstr.substring( 0, 3 ); // month (string)
  for ( i = 0 ; i < 36 ; i += 3 )
    if ( month_names.substring(i, i + 3) == PCstr )
    {
      i = int(i / 3) + 1;   //month
      break;
    }
  rtc.setDate(j, i, k);
  /*
    rtc.setDate(26,9,2020); // Sabbato
    rtc.setTime(16,29,30);
  */
  rtc.setDOW();  // calculate week day ....
}

String time_str()
{ String s = "";
  t = rtc.getTime();  s += wkday[int(t.dow) - 1];
  if ( Large_LCD ) { s += " "; }
  s += nr2digits(t.date) + "/" + nr2digits(t.mon) ;
  if ( Large_LCD ) { s += "/" + nr2digits( ( t.year % 100 ) ) ; }
  s += " " + nr2digits(t.hour) + ":" + nr2digits(t.min) + ":" + nr2digits( t.sec )  ;
  return ( s ) ;
}

String nr2digits(int n)
{ String s = "" ;
  s += char( int (n / 10) + byte('0') ) ;
  s += char( n % 10 + byte('0') ) ;
  return ( s ) ;
}

void print_2_lcd (int x, int y, String s, bool clr )
  {   if ( clr ) lcd.clear();  lcd.setCursor(x, y);  lcd.print(s);  }

void create_characters()
```

```
{
  // int i;
  //  for (i=0 ; i<8 ; i++ )   lcd.createChar(i, bar[i]);

 /*
  *
  byte ver_bar[5][8] =
  { 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
    0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C,
    0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
    0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F
  };
  for (int i = 0 ; i < 5 ; i++ )   lcd.createChar(i, ver_bar[i]); /// <---
*/

  byte ver_bar[8] =
    { 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F  };
  lcd.createChar(4, ver_bar);

  byte bell [8] = { 0b00100, 0b01110, 0b01110, 0b01110,
                    0b11111, 0b00100, 0b00000, 0b00000
                  };
  lcd.createChar(5, bell);



  //  lcd.createChar(0, heart);
  //  lcd.createChar(1, smiley);
  byte right_arrow[8] =   { 0b01000, 0b01100, 0b01110, 0b01111,
                            0b01110, 0b01100, 0b01000, 0b00000
                          };
  lcd.createChar(0, right_arrow);

  byte left_arrow[8] =   { 0b00010, 0b00110, 0b01110, 0b11110,
                           0b01110, 0b00110, 0b00010, 0b00000
                          };
  lcd.createChar(1, left_arrow);

  byte delta[8] =   { 0b00100, 0b01010, 0b10001, 0b10001,
                      0b10001, 0b10001, 0b11111, 0b00000
                    };
  lcd.createChar(6, delta);

  byte gr_pi[8] =   { 0b11111, 0b10001, 0b10001, 0b10001,
                      0b10001, 0b10001, 0b10001, 0b00000
                    };
  lcd.createChar(7, gr_pi);

  wkday [5][0] = char(246) ; // SA - Greek Sigma
  wkday [0][0] = char(6) ; // De - Greek Delta
  wkday [3][0] = char(7) ; // PE - Greek Pi
  wkday [4][0] = char(7) ; // PA - Greek Pi
}

void display_characters()
{ int i, j;
  lcd.clear();
  j = 140;
  for (i = j ; i < j + 60 ; i++ )
  { lcd.setCursor((i - j) % 20, (i - j) / 20);
    lcd.write(byte(i));
  }
  lcd.setCursor(0, 3); ///lcd.print(" .  ");

  String s = "" ;
  s += char(246) ;// greek Sigma
```

```
  s += char(244); // greek short Omega
  s += char(242); // greek short theta
  s += char(224); // greek small alfa
  s += char(226); // greek small beta
  s += char(227); // greek small epsilon
  s += char(228); // greek small mi
  s += char(229); // greek small sigma
  s += char(230); // greek small ro
  s += char(247); // greek small pi
  s += char(223); // Degrees
  s += char(96); // short upper slash
  s += char(164); // short lower slash
  s += "/\\|";
  for (i = 0; i < 6 ; i++ ) s += char(i);
  lcd.print(s);
  // for (i=0; i<5 ; i++ )  lcd.write(byte(i));

  //delay(5000);
  //  wkday2greek();
  s = "" ;
  for (i = 0 ; i < 7 ; i++ ) s += wkday [i] + " ";
  print_2_lcd(0, 2, s, false);
//  delay(1000);
}
```

Παράρτημα Β : Πρότυπα κατασκευής

Απεικονίζονται τα πατρόν που χρησιμοποιήθηκαν για τη διάτρηση του κουτιού στα σημεία που χρειαζόταν, για τον Arduino, το Real Time Clock, την Οθόνη και τις υπόλοιπες τρύπες στην πρόσοψη Διαστάσεις σε mm.