ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Μελέτη, σχεδίαση και υλοποίηση συστήματος διαχείρισης και εξοικονόμησης ενεργειακών πόρων σε κατοικία με την χρήση μικροελεγκτή 32 bit»

Του φοιτητή
Βασιλείου Κανταρτζή
Αρ. Μητρώου: 52008Μ

Επιβλέπων
Άγγελος Γιακουμής
Επίκουρος Καθηγητής

Ιούνιος 2024

Τίτλος Δ.Ε. :
Μελέτη, σχεδίαση και υλοποίηση συστήματος διαχείρισης και εξοικονόμησης ενεργειακών πόρων σε κατοικία με την χρήση μικροελεγκτή 32 bit
Κωδικός Δ.Ε.: 23103
Ονοματεπώνυμο φοιτητή: Βασίλειος Κανταρτζής
Ονοματεπώνυμο εισηγητή: Άγγελος Γιακουμής
Ημερομηνία ανάληψης Δ.Ε.: 31-01-2023
Ημερομηνία περάτωσης Δ.Ε.: 30-06-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της μεταπτυχιακής 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 Μεταπτυχιακό Πρόγραμμα Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» 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.

«Αφιέρωση»

Αφιερωμένη στους Γονείς μου, την σύζυγό μου και τα παιδιά μου.

Πρόλογος

Η υπερβολική αύξηση του ενεργειακού κόστους τα τελευταία χρόνια σε συνδυασμό με την σπατάλη των ενεργειακών πόρων από τον άνθρωπο ήταν το κίνητρο που με ώθησε στην αναζήτηση τρόπων εξοικονόμησης ενέργειας. Αλλάζοντας τους λαμπτήρες σε λαμπτήρες LED και χρησιμοποιώντας άλλους τρόπους θέρμανσης (φυσικό αέριο αντί για πετρέλαιο) και ψύξης (κλιματιστικά με μεγαλύτερη απόδοση) βοηθάει προς αυτήν την κατεύθυνση αλλά πάντα υπάρχουν περιθώρια βελτίωσης. Σε γενικές γραμμές χρειάζεται ένα κτήριο να είναι καλά μονωμένο για να μην έχει θερμικές απώλειες, τα κουφώματα να είναι νέας τεχνολογίας με διπλά τζάμια, οι οικιακές συσκευές που χρησιμοποιούνται να είναι ενεργειακά πιο αποδοτικές (δείκτης ενεργειακής απόδοσης Α και πάνω) και άλλες παρεμβάσεις που μπορεί να κάνει ο ιδιοκτήτης αλλά με υψηλό κόστος.

Σε αυτήν την μεταπτυχιακή εργασία γίνεται μελέτη ενός τρόπου εξοικονόμησης ενέργειας με την μείωση της κατανάλωσης ενέργειας ή και την διακοπή κατανάλωσης ενέργειας σε χρόνους και σε μέρη της οικίας που δεν χρειάζεται.

Περίληψη

Η παρούσα διπλωματική εργασία μελετά έναν τρόπο εξοικονόμησης ενέργειας μέσα σε ένα σπίτι. Στην αρχή γίνεται αναφορά της ενέργειας που καταναλώνεται από τα νοικοκυριά στην Ευρωπαϊκή Ένωση συμφώνα με την Ευρωπαϊκή Στατιστική Αρχή καθώς και για το ποσοστό εξοικονόμησης με διάφορους τρόπους. Στην συνέχεια γίνεται παρουσίαση του συστήματος εξοικονόμησης ενέργειας και του τρόπου λειτουργίας και ρύθμισης αυτού μέσα από εικόνες για να γίνει πιο εύκολη η κατανόηση της λειτουργίας και της παραμετροποίησης αυτού. Μετά γίνεται ανάλυση της σχεδίασης και κατασκευής του κάθε μέρους του συστήματος δίνοντας έμφαση στα κυριότερα σημεία του κάθε μέρους του συστήματος. Παρουσιάζονται οι επεξεργαστές, οι αισθητήρες, ο πομποδέκτης και όλα τα υπόλοιπα εξαρτήματα που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος. Παρακάτω γίνεται αναφορά στο πρόγραμμα που χρησιμοποιήθηκε για τον 3D σχεδιασμό και την εκτύπωση των κουτιών που χρησιμοποιήθηκαν στην κατασκευή του συστήματος. Στην συνέχεια έγινε αναλυτική παρουσίαση του λογισμικού των μικροελεγκτών που χρησιμοποιήθηκαν. Στο τέλος καταγράφηκαν τα συμπεράσματα καθώς και οι βελτιώσεις που είναι δυνατόν να γίνουν στο μέλλον.

«Study, design and implementation of a system for managing and saving energy resources in a residence using a 32-bit microcontroller»

«VASILEIOS KANTARTZIS»

Abstract

This thesis studies a way to save energy inside a house. At the beginning there is a reference to the energy consumed by households in the European Union according to the Eurostat as well as to the percentage of savings in various ways. Then the energy saving system and its operation and adjustment mode are presented through images to make it easier to understand its operation and configuration. Then the design and construction of each part of the system is analyzed, emphasizing the main points of each part of the system. The processors, sensors, transceiver and all other components used to implement the system are presented. After that there is a reference to the program used for the 3D design and printing of the boxes used in the construction of the system. Then there was a detailed presentation of the software of the microcontrollers used. At the end, the conclusions were documented as well as the improvements that can be made in the future.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένειά μου για την υπομονή και την στήριξη τους σε αυτήν την προσπάθειά μου. Επίσης θα ήθελα να ευχαριστήσω όλους τους καθηγητές του μεταπτυχιακού για τις γνώσεις που μου μετέφεραν και την βοήθειά τους και ένα μεγάλο ευχαριστώ για τον επιβλέπων καθηγητή κ. Άγγελο Γιακουμή, για την βοήθειά και την εμπιστοσύνη που μου έδειξε.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Συντομογραφίες

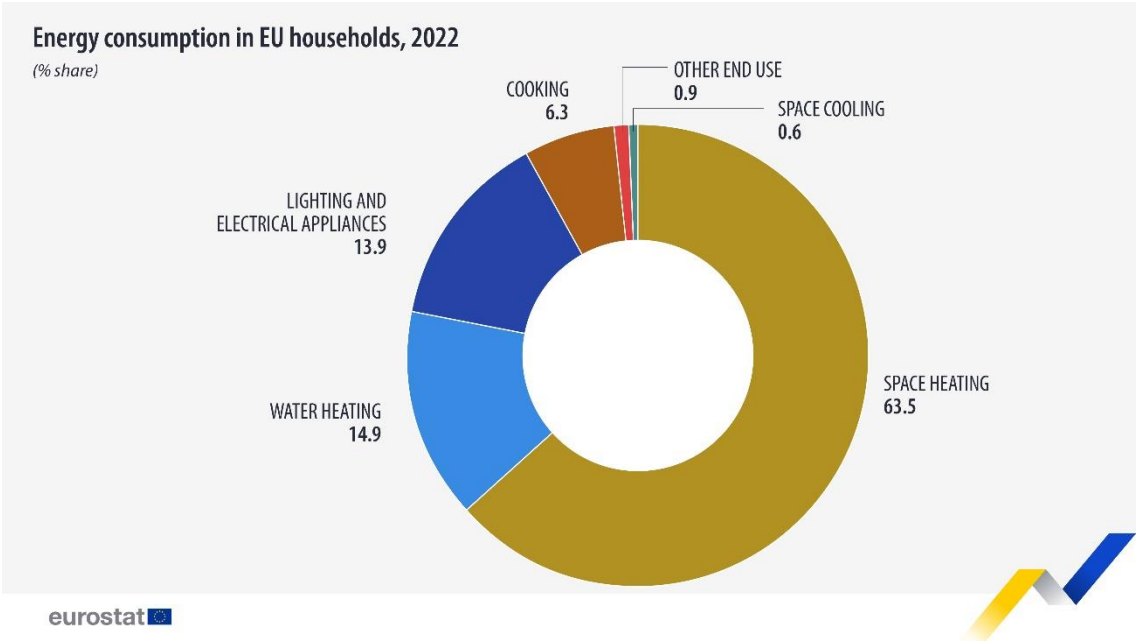
Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Φ.Α.	Φυσικό Αέριο
HMI	Human Machine Interface
RTC	Real Time Clock
Φ.Π.Α.	Φόρος Προστιθέμενης Αξίας
LDO	Low Dropout Voltage
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
κ.ο.κ.	και ούτω καθεξής

Κεφάλαιο 1ο: Ενεργειακοί Πόροι

1.1 Εισαγωγή

Τα τελευταία χρόνια ακούμε συνέχεια συζητήσεις γύρο από την εξοικονόμηση ενεργειακών πόρων όχι μόνο στην Ελλάδα αλλά και σε όλη την Ευρώπη. Βασικότερη αιτία θα μπορούσε να πει κάποιος ότι είναι η απεξάρτηση από την εισαγόμενη ενέργεια (πετρέλαιο και Φ.Α.) από την Ρωσία μετά από την έναρξη του πολέμου με την Ουκρανία. Από τότε μέχρι και σήμερα παρατηρούμε μια συνεχόμενη αυξητική τάση με μικρές διακυμάνσεις προς τα κάτω στις τιμές του πετρελαίου του Φ.Α. και φυσικά της ηλεκτρικής ενέργειας.

Σύμφωνα με την Eurostat η κατανάλωση ενέργειας στην Ευρώπη το 2022 στα νοικοκυριά [1] ήταν κατά 63,5% για θέρμανση χώρων και 13,9% για φωτισμό και διάφορες ηλεκτρικές συσκευές όπως φαίνεται στην Εικόνα 1.1.



Εικόνα 1.1 : Energy consumption in EU households, 2022

1.2 Εξοικονόμηση ενεργειακών πόρων

Δυστυχώς για τις τιμές δεν μπορούμε να κάνουμε και πολλά πράγματα πέρα από το να ψάχνουμε κάθε μήνα για παρόχους με φθηνότερα τιμολόγια ηλεκτρικής ενέργειας και Φ.Α.. Εξοικονόμηση ενεργειακών πόρων μπορεί να γίνει με τους εξής τρόπους:

- Κτηριακές Παρεμβάσεις
- Αλλαγή τρόπου θέρμανσης και ψύξης
- Αλλαγή οικιακών συσκευών
- Ηλεκτρονικό σύστημα διαχείρισης ενέργειας

1.2.1 Κτηριακές Παρεμβάσεις

Η πιο απλή παρέμβαση στο κτήριο είναι η τοποθέτηση τέντας, έτσι ώστε να δημιουργείται σκιά στο σημείο και να μην υπάρχει απευθείας έκθεση στον ήλιο. Μια άλλη παρέμβαση είναι η αντικατάσταση των κουφωμάτων με νέας τεχνολογίας με διπλά τζάμια για καλύτερη θερμομόνωση. Επίσης πολύ σημαντική είναι και η θερμομόνωση του κτηρίου είτε εξωτερικά είτε εσωτερικά αποτρέποντας έτσι την μεταφορά της θερμότητας από μέσα προς τα έξω το χειμώνα και από έξω προς τα μέσα το καλοκαίρι. Επίσης πολύ σημαντική είναι και μόνωση της ταράτσας και του πατώματος όπως και των σωληνώσεων στην εγκατάσταση της θέρμανσης. Αυτές είναι μερικές κτηριακές παρεμβάσεις για την εξοικονόμηση ενεργειακών πόρων αλλά είναι και αρκετά δαπανηρές.

1.2.2 Αλλαγή τρόπου θέρμανσης και ψύξης

Ένας άλλος τρόπος εξοικονόμησης ενέργειας είναι η αλλαγή του καυστήρα θέρμανσης με έναν νέας τεχνολογίας ο οποίος είναι πιο αποδοτικός και κατά συνέπεια έχει μικρότερη κατανάλωση καυσίμου. Ακόμα παραπέρα εάν στην περιοχή υπάρχει Φ.Α. η αντικατάσταση του καυστήρα πετρελαίου με καυστήρα Φ.Α. με συμπύκνωση οι οποίοι είναι οικονομικότεροι και ποιο φιλικοί προς το περιβάλλον. Μια άλλη λύση με μεγάλο αρχικό κόστος είναι οι αντλίες θερμότητας.

Όσο για την ψύξη μπορούν να χρησιμοποιηθούν κλιματιστικά με υψηλή απόδοση ή αντλίες θερμότητας με fan coil.

1.2.3 Αλλαγή οικιακών συσκευών

Η πιο διαδεδομένη μέθοδος εξοικονόμησης ενέργειας τα τελευταία δύο χρόνια και σε πολλές περιπτώσεις με επιδοτήσεις είναι η αλλαγή ενεργοβόρων οικιακών συσκευών με οικιακές συσκευές υψηλής απόδοσης όπως ψυγεία, καταψύκτες κ.α..

1.2.4 Ηλεκτρονικό σύστημα διαχείρισης ενέργειας

Τέλος ένας άλλος τρόπος εξοικονόμησης ενέργειας είναι αυτός ο οποίος μελετάτε σε αυτήν την Δ.Ε.. Με απλά λόγια γίνεται κατανάλωση ενέργειας μόνο εκεί που είναι απαραίτητο και για όσο χρονικό διάστημα χρειάζεται. Αυτό μπορούμε να το πετύχουμε τοποθετώντας αισθητήρια ανίχνευσης ανθρώπινης παρουσίας σε επιλεγμένα σημεία μέσα στο σπίτι. Όταν ανιχνεύει ανθρώπινη παρουσία ακόμα και στατική (δίχως να κινείται καθόλου) τότε όλα τα συνδεδεμένα συστήματα λειτουργούν όπως έχουν προγραμματιστεί να λειτουργήσουν όταν υπάρχει ανθρώπινη παρουσία. Όταν δεν ανιχνεύει ανθρώπινη παρουσία τότε απενεργοποιεί ή αλλάζει τον τρόπο λειτουργίας των συνδεμένων συσκευών και μειώνει κατά έναν βαθμό την θερμοκρασία στον χώρο αυτό ανάλογα με τον προγραμματισμό που έχει γίνει. Επίσης το σύστημα είναι συνδεμένο και με αισθητήρες στα παράθυρα και τις μπαλκονόπορτες έτσι ώστε όταν ανοίγουμε κάποιο παράθυρο ή μπαλκονόπορτα να σταματάει η θέρμανση του συγκεκριμένου χώρου και να μην λειτουργεί άσκοπα η θέρμανση καταναλώνοντας ενέργεια.

Σύμφωνα με τον Διεθνή Οργανισμό Ενέργειας και την Ευρωπαϊκή Επιτροπή [2] η μέση θερμοκρασία για την θέρμανση στα σπίτια στην Ευρωπαϊκή Ένωση είναι ρυθμισμένη πάνω από 22 βαθμούς αλλά θα μπορούσε άνετα να είναι στους 19 με 20 βαθμούς. Στην θέρμανση ρυθμίζοντας την θερμοκρασία κατά έναν βαθμό λιγότερο έχουμε οικονομία της τάξης του 7%, ενώ στην ψύξη ανεβάζοντας την θερμοκρασία κατά έναν βαθμό έχουμε οικονομία της τάξης του 10% στην ηλεκτρική ενέργεια. Έτσι σε χώρους που δεν χρησιμοποιούμαι συχνά μπορούμε να χαμηλώσουμε την θερμοκρασία στην τάξη των 19 με 20 βαθμών.

Η θερμοκρασία δεν είναι ο μόνος παράγοντας που μας δίνει την αίσθηση του ζεστού ή του κρύου και έτσι να ανεβάζουμε συνεχώς την θερμοκρασία του θερμοστάτη όταν αισθανόμαστε κρύο (ή να την κατεβάζουμε όταν αισθανόμαστε ζέστη). Σημαντικό παράγοντα παίζει και η υγρασία του χώρου που βρισκόμαστε την οποία επίσης μπορούμε να ρυθμίσουμε για κάθε χώρο που ελέγχουμε ξεχωριστά.

1.3 Επίλογος

Στο πρώτο κεφάλαιο είδαμε σύμφωνα με την Ευρωπαϊκή στατιστική αρχή ότι στην Ευρώπη τα νοικοκυριά καταναλώνουν το μεγαλύτερο μέρος της ενέργειας για την θέρμανση των χώρων διαβίωσης και ένα αρκετά σημαντικό ποσοστό ενέργειας για τον φωτισμό και διάφορες ηλεκτρικές συσκευές. Είδαμε τρόπους εξοικονόμησης ενεργειακών πόρων με κτηριακές παρεμβάσεις οι οποίες είναι και αρκετά δαπανηρές, με χρήση άλλων ενεργειακών πόρων (Φ.Α. ή ηλεκτρικής ενέργειας αντί για πετρέλαιο) και αλλαγή ενεργοβόρων οικιακών συσκευών με οικιακές συσκευές υψηλής απόδοσης. Στο τέλος είδαμε και έναν τρόπο να μειώσουμε την κατανάλωση ενέργειας όταν δεν την χρειαζόμαστε μέσω ηλεκτρονικού ελέγχου των χώρων μιας οικίας.

Κεφάλαιο 2ο: Λειτουργία Συστήματος Διαχείρισης Ενέργειας

2.1 Εισαγωγή

Η αρχή λειτουργίας του συστήματος διαχείρισης ενέργειας περιγράφεται στην συνέχεια. Πρώτα περνάμε όλες τις ρυθμίσεις που μας ικανοποιούν στον κεντρικό ελεγκτή για τους χώρους που θέλουμε να ελέγχουμε. Στην συνέχεια μετά την ενεργοποίηση των τοπικών ελεγκτών οι ρυθμίσεις περνάνε στους τοπικούς ελεγκτές και ο καθένας ελέγχει στον κάθε χώρο του σπιτιού που βρίσκεται για ανθρώπινη παρουσία, την θερμοκρασία και την υγρασία του χώρου και εάν είναι ανοιχτό κάποιο παράθυρο ή κάποια μπαλκονόπορτα. Αυτές οι πληροφορίες μεταδίδονται ασύρματα μεταξύ των ελεγκτών και των δεκτών και ανάλογα με τα δεδομένα που έχουν εισαχθεί στον κεντρικό ελεγκτή εκτελούνται διάφορες λειτουργίες που έχουν ως σκοπό την εξοικονόμηση ενέργειας.

2.2 Ρύθμιση-Λειτουργία του Συστήματος Διαχείρισης Ενέργειας

Η όλη λειτουργία του συστήματος στηρίζεται στην ανίχνευση ανθρώπινης παρουσίας στον χώρο που ελέγχουμε, την επιθυμητή θερμοκρασία και την σχετική υγρασία που θέλουμε στον χώρο και αν υπάρχει κάποια απώλεια θερμοκρασίας στον χώρο εξαιτίας κάποιου ανοιχτού παραθύρου ή μπαλκονόπορτας.

Αρχικά μέσω του καταλόγου όπως θα δούμε παρακάτω ενεργοποιούμε τον ή τους τοπικούς ελεγκτές που επιθυμούμε. Στην συνέχεια ορίζουμε την επιθυμητή θερμοκρασία και σχετική υγρασία του δωματίου που θα ελέγχει ο τοπικός ελεγκτής. Όταν υπάρχει ανίχνευση ανθρώπινης παρουσίας τότε η θερμοκρασία παραμένει σε αυτήν που έχει αρχικά ρυθμιστεί. Αν δεν υπάρχει ανίχνευση ανθρώπινης παρουσίας τότε η θερμοκρασία ρυθμίζεται αυτόματα κατά έναν βαθμό λιγότερο για να μην υπάρχει σπατάλη ενέργειας και ταυτόχρονα να μην πέσει και πολύ η θερμοκρασία στο δωμάτιο. Το επόμενο βήμα είναι ο ορισμός της ώρας κατά την οποία το σύστημα θα λειτουργεί με νυχτερινές ρυθμίσεις και κατόπιν ο ορισμός της ώρας που θα λειτουργεί με ρυθμίσεις ημέρας. Άλλη μια ρύθμιση που πρέπει να γίνει είναι αυτή του νυχτερινού φωτισμού. Κατά την διάρκεια της λειτουργίας ημέρας ο φωτισμός ανάβει στο μέγιστο της έντασής του με την ανίχνευση ανθρώπινης παρουσίας. Κατά την διάρκεια λειτουργίας νύχτας, όπου είναι ενεργοποιημένος ο νυχτερινός φωτισμός όταν ο ανιχνευτής ανιχνεύσει ανθρώπινη παρουσία τότε ενεργοποιείται ο φωτισμός αλλά με μειωμένη ένταση (dimmer). Όπου όμως δεν είναι ενεργοποιημένος ο νυχτερινός φωτισμός είτε ανιχνευτεί είτε όχι ανθρώπινη παρουσία δεν ενεργοποιείται ο φωτισμός. Η τελευταία ρύθμιση που χρειάζεται να κάνουμε είναι αυτή του χρόνου που θα παραμένει ενεργοποιημένος ο φωτισμός από την στιγμή που δεν ανιχνεύεται ανθρώπινη παρουσία. Με άλλα λόγια για πόση ώρα θα είναι αναμμένα τα φώτα από την στιγμή που θα φύγουμε από το δωμάτιο. Ο αρχικός χρόνος είναι ένα λεπτό αλλά μπορεί να ρυθμιστεί μέχρι πέντε λεπτά με βήμα ενός λεπτού.

Το σύστημα για την σωστή εναλλαγή της νυχτερινής και της ημερήσιας λειτουργίας είναι εφοδιασμένο με ένα αυτόνομο RTC με δική του μπαταρία. Μια ακόμα ρύθμιση είναι αυτή της ημερομηνίας και της ώρας του συστήματος μέσω του καταλόγου.

Αφού έχουμε ρυθμίσει τον κεντρικό ελεγκτή με τις δικές μας προτιμήσεις αυτός με την σειρά του στέλνει ασύρματα μέσω ενός πομποδέκτη τον ‘NRF24L01+’ [3] όλες τις ρυθμίσεις στους τοπικούς ελεγκτές τον έναν μετά τον άλλον. Ο τοπικός ελεγκτής με την σειρά του αποθηκεύει σε καταχωρητές τις ρυθμίσεις και αφού συλλέξει πληροφορίες από τους αισθητήρες θερμοκρασίας, υγρασίας, κίνησης και τον διακόπτη παραθύρου, συγκρίνει τις πληροφορίες με τις ρυθμίσεις και στέλνει τις κατάλληλες

πληροφορίες στους τοπικούς δέκτες για να εκτελέσουν τις αντίστοιχες λειτουργίες που πρέπει, όπως να ανάψουν ή να σβήσουν τα φώτα κ.τ.λ.. Όταν ο τοπικός ελεγκτής τελειώσει την αποστολή δεδομένων σε όλους τους τοπικούς δέκτες που είναι συνδεμένος τότε στέλνει δεδομένα στον κεντρικό ελεγκτή για να ανοίξει ή να κλείσει την θέρμανση στον χώρο που ελέγχει ο τοπικός ελεγκτής και ότι έχει τελειώσει με την αποστολή δεδομένων στους τοπικούς δέκτες για να στείλει τα δεδομένα ο κεντρικός ελεγκτής στον επόμενο τοπικό ελεγκτή.

Η λειτουργία της εκπομπής και της λήψης δεδομένων μεταξύ των επιμέρους συστημάτων γίνεται σειριακά. Αυτό γίνεται για να μην υπάρξει ταυτόχρονη εκπομπή από δύο ή περισσότερους πομποδέκτες και χαθούν έτσι οι πληροφορίες. Ανά τακτά χρονικά διαστήματα πρώτα ο κεντρικός ελεγκτής επικοινωνεί με τον τοπικό ελεγκτή. Ο τοπικός ελεγκτής αφού πάρει τα δεδομένα εκπέμπει πίσω ένα μήνυμα λέγοντας ότι παρέλαβε τα δεδομένα αλλά περίμενε να επικοινωνήσω με τους τοπικούς δέκτες. Μετά ο τοπικός ελεγκτής επικοινωνεί με τους τοπικούς δέκτες τον έναν μετά τον άλλον. Όταν τελειώσει με τους τοπικούς δέκτες στην συνέχεια στέλνει νέο μήνυμα στον κεντρικό ελεγκτή να προχωρήσει με τον επόμενο τοπικό ελεγκτή, Το μειονέκτημα με αυτόν τον τρόπο επικοινωνίας είναι ότι χάνεται χρόνος μέχρι να επικοινωνήσουν όλοι με όλους, αλλά το πλεονέκτημα είναι ότι έτσι δεν θα υπάρξει απώλεια στα δεδομένα λόγο παρεμβολών. Επίσης ο χρόνος επικοινωνίας μεταξύ των μονάδων είναι πολύ μικρός για να δημιουργήσει πρόβλημα στην λειτουργία του συστήματος.

2.3 Σύστημα διαχείρισης ενέργειας

Το σύστημα διαχείρισης ενέργειας αποτελείται από τέσσερα κύρια μέρη:

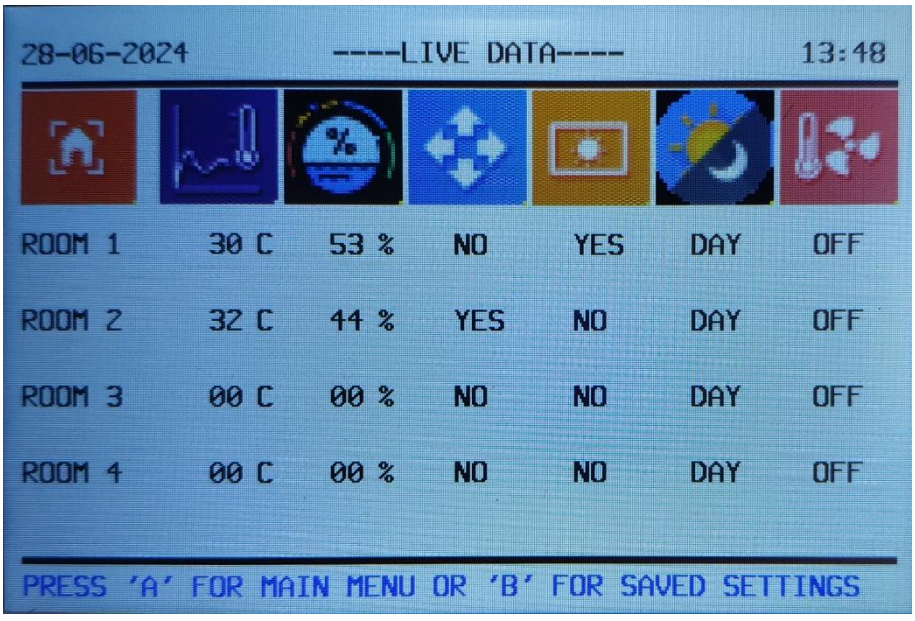
- Τον κεντρικό ελεγκτή
- Τους τοπικούς ελεγκτές
- Τους τοπικούς δέκτες
- Τελική βαθμίδα ελέγχου ηλεκτροθερμικών κεφαλών

2.3.1 Κεντρικός Ελεγκτής

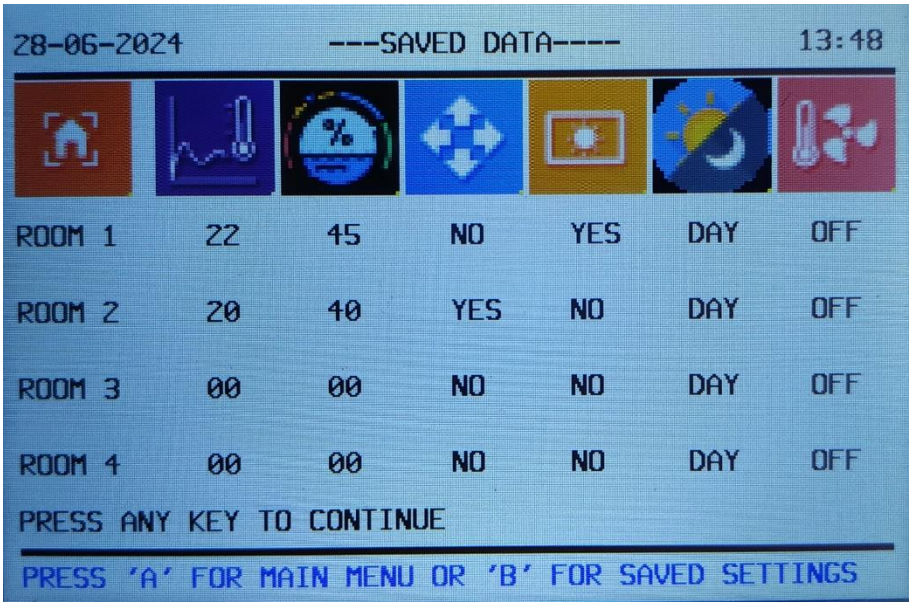
Ο κεντρικός ελεγκτής (Εικόνα 2.1) είναι ένας για κάθε ολοκληρωμένο σύστημα και είναι στην ουσία ένας Human Machine Interface (HMI), ένας πίνακας ελέγχου μέσω του οποίου περνάμε όλες τις ρυθμίσεις που θέλουμε να κάνουμε μέσω ενός πληκτρολογίου 4X4 ενώ ταυτόχρονα μέσω της οθόνης που διαθέτει μπορούμε ανά πάσα στιγμή να βλέπουμε τα δεδομένα που έχουμε περάσει στον ελεγκτή και τα δεδομένα που στέλνονται σε αυτόν από τους τοπικούς ελεγκτές.

Εικόνα 2.1 : Κεντρικός Ελεγκτής

Επίσης μέσω της οθόνης (Εικόνες 2.2 και 2.3) βλέπουμε είτε ζωντανά (LIVE) είτε τις αποθηκευμένες (SAVED) ρυθμίσεις που έχουμε κάνει.



Εικόνα 2.2 : Οθόνη Κεντρικού Ελεγκτή (LIVE DATA)



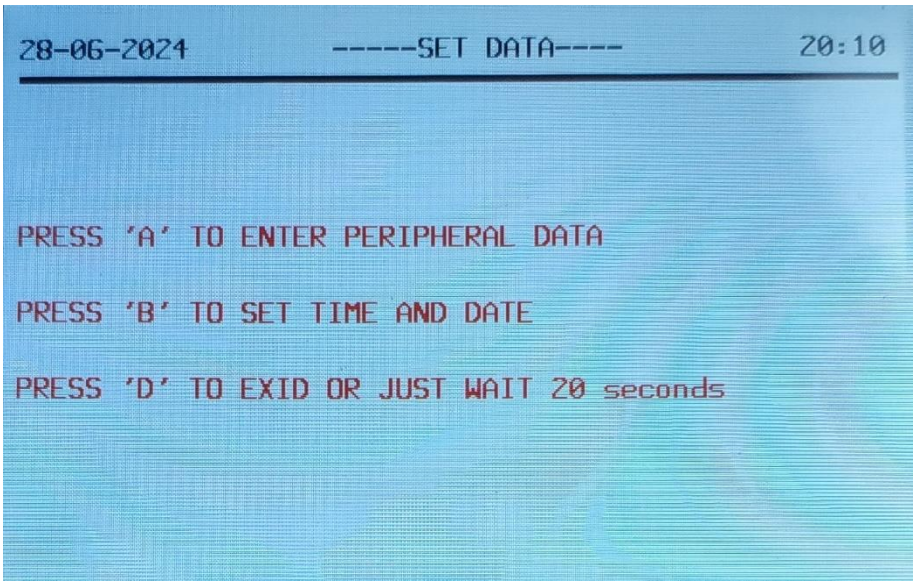
Εικόνα 2.3 : Οθόνη Κεντρικού Ελεγκτή (SAVED DATA)

Όπως φαίνεται από τις Εικόνες 2.2 και 2.3 επάνω αριστερά βλέπουμε την ημερομηνία και επάνω δεξιά την τρέχουσα ώρα. Στο κέντρο επάνω στην Εικόνα 2.2 έχουμε την ένδειξη '**LIVE DATA**' και βλέπουμε τα δεδομένα όπως έρχονται από τους αισθητήρες, ενώ στην Εικόνα 2.3 την ένδειξη '**SAVED DATA**' που είναι τα δεδομένα που περάσαμε εμείς στον κεντρικό ελεγκτή.

Από κάτω έχουμε εφτά εικονίδια που αντιστοιχούν στις εφτά στήλες που υπάρχουν κάτω από τις εικόνες. Το πρώτο εικονίδιο αναφέρετε στα δωμάτια στα οποία έχουμε τοποθετήσει τους τοπικούς ελεγκτές. Ο μέγιστος αριθμός είναι τέσσερις τοπικοί ελεγκτές σε τέσσερα δωμάτια (**ROOM 1, 2, 3, 4**) έναν για κάθε δωμάτιο. Το δεύτερο εικονίδιο αναφέρετε στην θερμοκρασία που υπάρχει σε κάθε δωμάτιο όταν είναι σε '**LIVE DATA**' (Εικόνα 2.2) και την επιθυμητή θερμοκρασία όταν είναι σε '**SAVED DATA**' (Εικόνα 2.3). Το τρίτο εικονίδιο αναφέρετε στην σχετική υγρασία που έχει το κάθε δωμάτιο σε '**LIVE DATA**' και την επιθυμητή σχετική υγρασία όταν είναι σε '**SAVED DATA**'. Το τέταρτο εικονίδιο αναφέρεται στο αν υπάρχει κίνηση σε κάποιο δωμάτιο. Όταν ο αισθητήρας ανιχνεύσει κάποια κίνηση (ανθρώπινη παρουσία) τότε έχουμε την ένδειξη '**YES**' για το αντίστοιχο δωμάτιο ενώ όταν δεν υπάρχει κίνηση τότε έχουμε την ένδειξη '**NO**'. Το επόμενο εικονίδιο μας πληροφορεί για το αν έχουμε ενεργοποιήσει τον νυχτερινό φωτισμό και σε ποιο δωμάτιο. Όπου έχουμε την ένδειξη '**YES**' σε εκείνο το δωμάτιο είναι ενεργοποιημένος ο νυχτερινός φωτισμός ενώ η ένδειξη '**NO**' μας ενημερώνει ότι ο νυχτερινός φωτισμός είναι απενεργοποιημένος. Το έκτο εικονίδιο μας δείχνει αν βρίσκεται το σύστημα σε λειτουργία ημέρας (**DAY**) ή σε λειτουργία νύχτας (**NIGHT**). Το τελευταίο εικονίδιο μας πληροφορεί αν είναι ενεργοποιημένη κάποια από τις ηλεκτροθερμικές κεφαλές και σε ποιο δωμάτιο. Όταν δίνεται η εντολή να ενεργοποιηθεί η ηλεκτροθερμική κεφαλή τότε στην οθόνη έχουμε την ένδειξη '**ON**' ενώ όταν απενεργοποιείται την ένδειξη '**OFF**'.

Όταν βρισκόμαστε στην οθόνη '**SAVED DATA**' κάτω από την ένδειξη '**ROOM 4**' εμφανίζεται η επιλογή '**PRESS ANY KEY TO CONTINUE**' που μας δίνει την δυνατότητα πατώντας οποιοδήποτε πλήκτρο να επιστρέψουμε στην αρχική οθόνη. Επίσης σε οποιοδήποτε σημείο του καταλόγου βρισκόμαστε μπορούμε να επιστέψουμε στην αρχική οθόνη πατώντας το πλήκτρο '**D**'. Στην

αρχική οθόνη στο κάτω μέρος της οθόνης δίνονται δύο επιλογές '**PRESS A FOR MAIN MENU OR B FOR SAVED SETTINGS**' που μας δίνει πρόσβαση στον κεντρικό κατάλογο πατώντας το πλήκτρο '**A**' ή πατώντας το πλήκτρο '**B**' πηγαίνουμε στις αποθηκευμένες ρυθμίσεις. Αφού πατήσουμε το πλήκτρο '**A**' μεταφερόμαστε στον κεντρικό κατάλογο (Εικόνα 2.4).



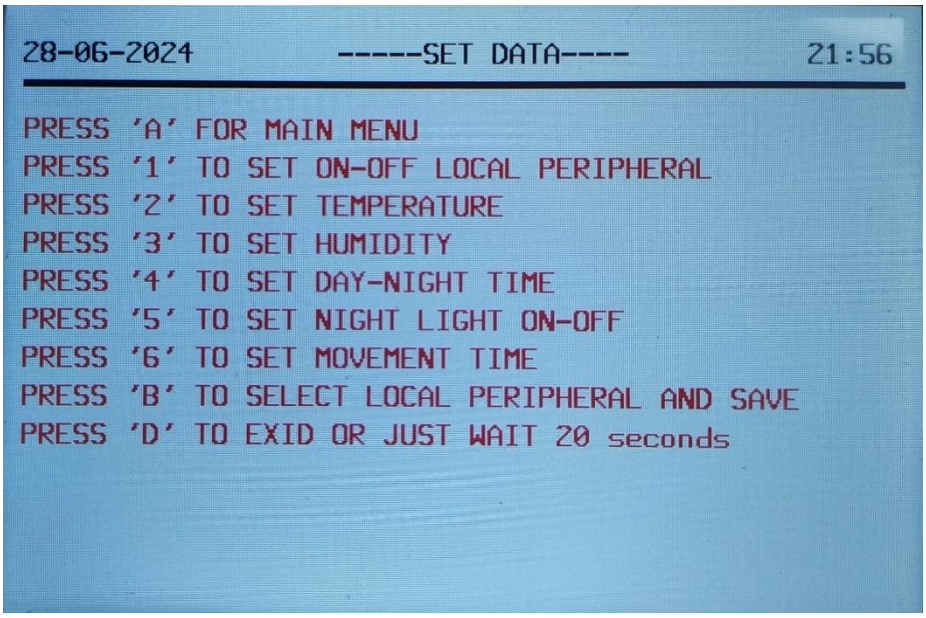
Εικόνα 2.4 : Οθόνη Κεντρικού Ελεγκτή (SET DATA)

Στον κεντρικό κατάλογο έχουμε τρείς επιλογές :

- '**PRESS A TO ENTER PERIPHERAL DATA**' όπου πατώντας το πλήκτρο '**A**' μεταφερόμαστε στον κατάλογο ρύθμισης παραμέτρων των περιφερειακών συσκευών.
- '**PRESS B TO SET TIME AND DATE**' όπου πατώντας το πλήκτρο '**B**' μεταφερόμαστε στον κατάλογο ρύθμισης ημερομηνίας και ώρας.
- '**PRESS D TO EXIT OR JUST WAIT 20 seconds**' όπου πατώντας το πλήκτρο '**D**' ή απλώς περιμένοντας 20 δευτερόλεπτα μεταφερόμαστε στην αρχική οθόνη.

Όταν γίνεται η τρίτη επιλογή ή απλά αν δεν γίνεται καμία επιλογή από τον κατάλογο, γίνεται μεταφορά στην αρχική οθόνη.

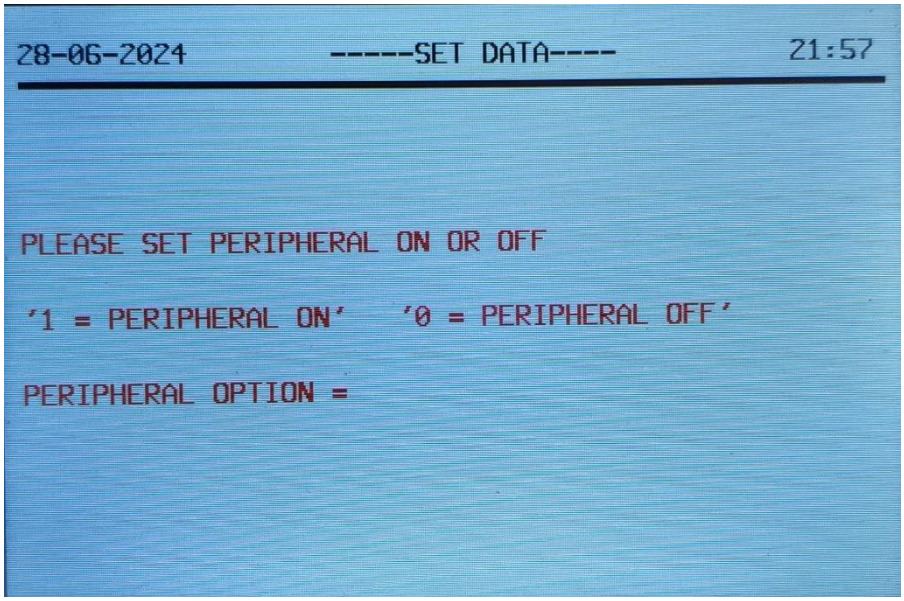
Με την πρώτη επιλογή ο χρήστης μεταφέρεται στον κατάλογο ρύθμισης παραμέτρων των περιφερειακών συσκευών (Εικόνα 2.5).



Εικόνα 2.5 : Κατάλογος ρύθμισης παραμέτρων των περιφερειακών συσκευών

Σε αυτόν τον κατάλογο διαλέγοντας την πρώτη επιλογή '**PRESS A FOR MAIN MENU**' γίνεται μεταφορά στον ακριβώς προηγούμενο κατάλογο, τον κεντρικό κατάλογο (Εικόνα 2.4).

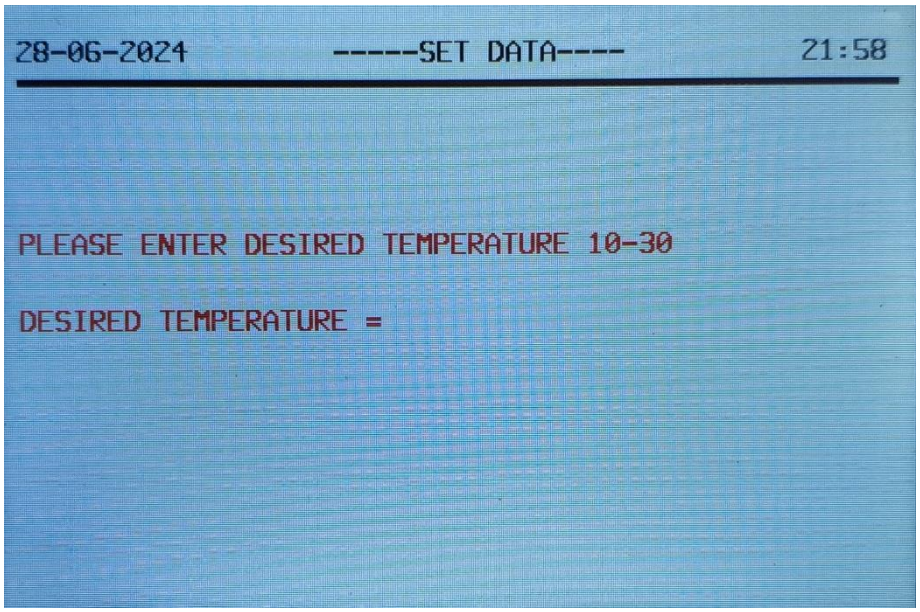
Με την δεύτερη επιλογή '**PRESS 1 TO SET ON-OFF LOCAL PERIPHERAL**' υπάρχει η δυνατότητα ενεργοποίησης ή να απενεργοποίησης του τοπικού ελεγκτή (Εικόνα 2.6).



Εικόνα 2.6 : Επιλογή ενεργοποίησης απενεργοποίησης τοπικού ελεγκτή

Σε αυτόν τον κατάλογο μπορεί να επιλεχθεί η τιμή '**1'** για να ενεργοποίηση του τοπικού ελεγκτή ή η τιμή '**0'** για να την απενεργοποίηση. Μετά την επιλογή πατώντας οποιοδήποτε πλήκτρο η επιλογή αποθηκεύεται στην μνήμη για να επιλεχθεί στο τέλος ποιόν από τους τοπικούς ελεγκτές αφορά.

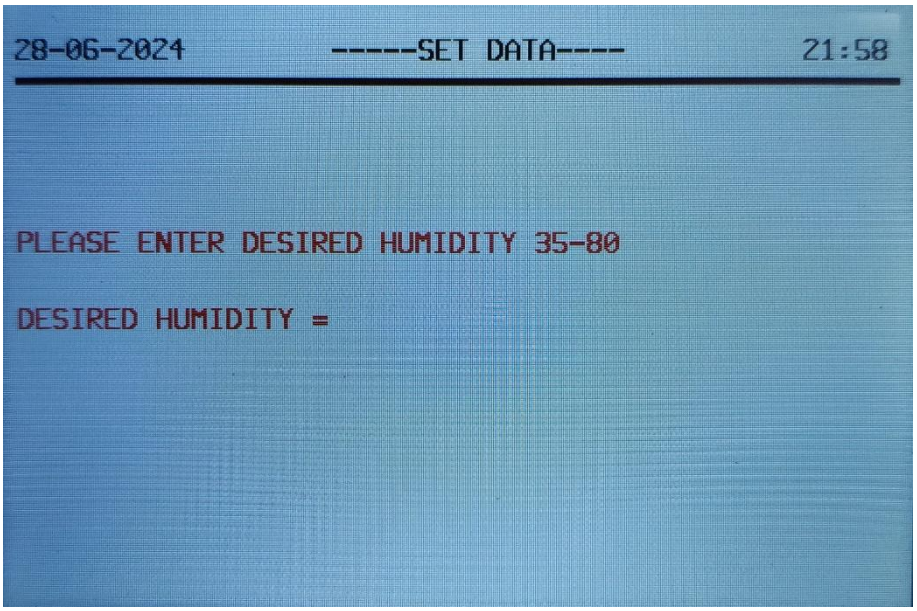
Στην τρίτη επιλογή '**PRESS 2 TO SET TEMPERATURE**' γίνεται εισαγωγή της επιθυμητής θερμοκρασίας του δωματίου (Εικόνα 2.7).



Εικόνα 2.7 : Εισαγωγή επιθυμητής θερμοκρασίας

Εδώ μπορεί να επιλεχθεί θερμοκρασία από 10 – 30 βαθμούς. Αν επιλεχθεί τιμή μικρότερη από 10 τότε θα αποθηκευτή η τιμή 10. Το ίδιο και για την υψηλότερη τιμή, αν επιλεχθεί τιμή μεγαλύτερη από 30 τότε θα αποθηκευτή η τιμή 30. Η θερμοκρασία που αποθηκεύεται εδώ είναι η θερμοκρασία που θα έχει το δωμάτιο όταν υπάρχει κάποιος μέσα σε αυτό. Όταν δεν είναι κανένας μέσα στο δωμάτιο τότε η θερμοκρασία αυτόματα θα είναι κατά έναν βαθμό μικρότερη για εξοικονόμηση ενέργειας. Αφού εισαχθεί η επιθυμητή θερμοκρασία τότε πατώντας οποιοδήποτε πλήκτρο αποθηκεύεται η τιμή στην μνήμη.

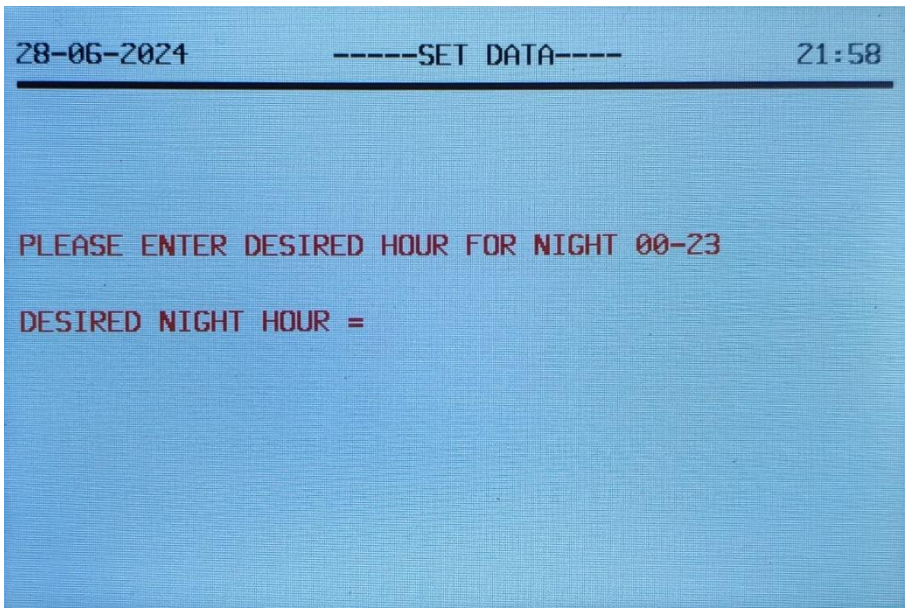
Στην τέταρτη επιλογή '**PRESS 3 TO SET HUMIDITY**' γίνεται εισαγωγή της επιθυμητής σχετικής υγρασίας που θέλουμε να διατηρείται στο δωμάτιο (Εικόνα 2.8).



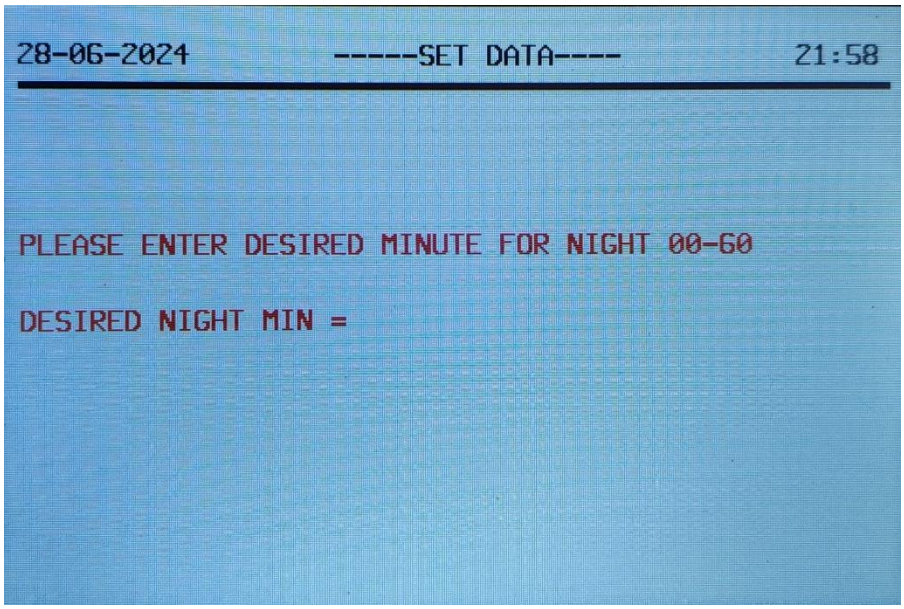
Εικόνα 2.8 : Εισαγωγή επιθυμητής υγρασίας

Η επιθυμητή υγρασία που μπορεί να επιλεχθεί είναι από 35%-80%. Η υστέρηση που υπάρχει είναι 5%. Δηλαδή αν επιλεχθεί υγρασία 50% τότε θα ενεργοποιηθεί στο 55% και θα απενεργοποιηθεί στο 45%. Και εδώ για να αποθηκευτεί η τιμή απλός πληκτρολογείτε οποιοδήποτε πλήκτρο.

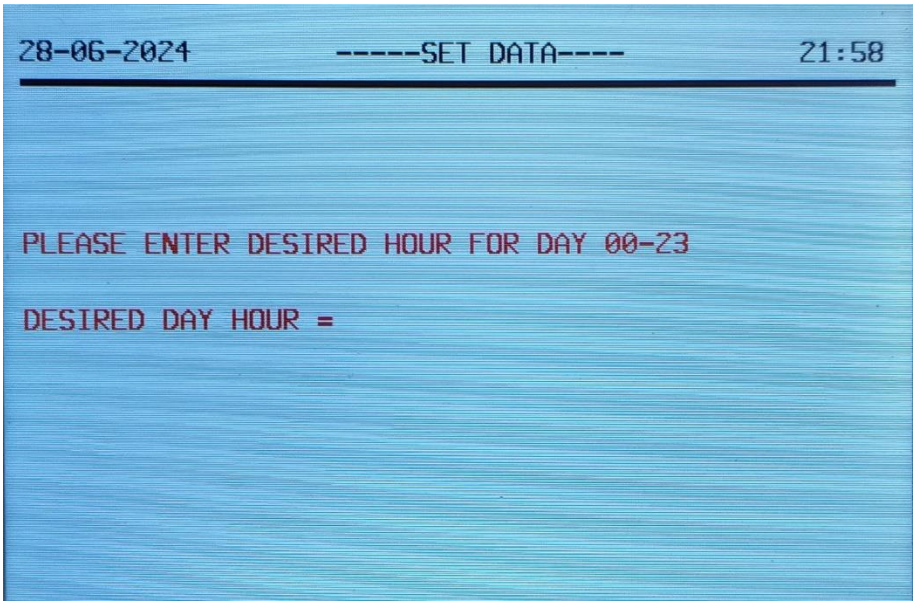
Η πέμπτη επιλογή '**PRESS 4 TO SET DAY-NIGHT TIME**' καθορίζει την ώρα που σταματάει η λειτουργία ημέρας και ξεκινάει η λειτουργία νύχτας καθώς επίσης την ώρα που σταματάει η λειτουργία νύχτας και ξεκινάει η λειτουργία ημέρας (Εικόνες 2.9, 2.10, 2.11, 2.12).



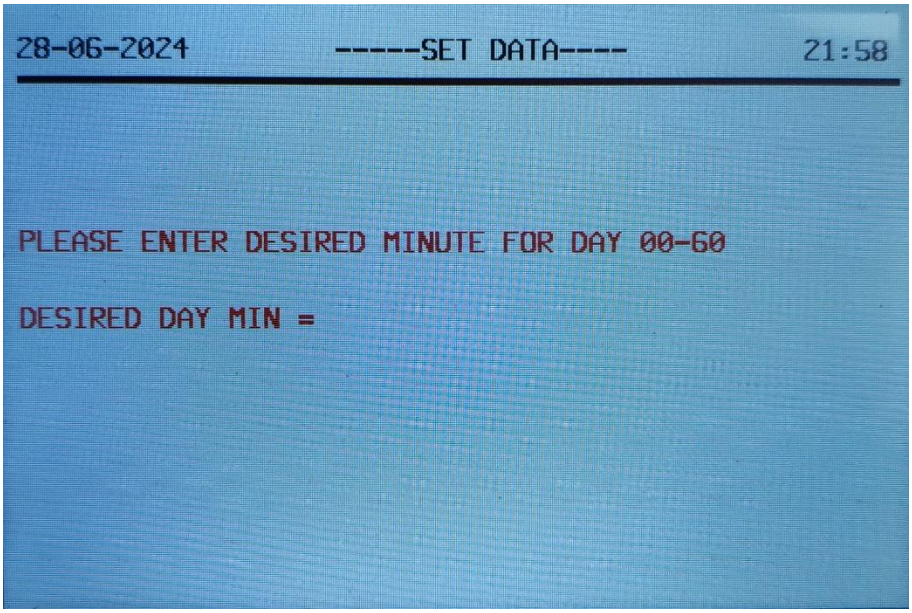
Εικόνα 2.9 : Εισαγωγή ώρας λειτουργίας νύχτας



Εικόνα 2.10 : Εισαγωγή λεπτών λειτουργίας νύχτας



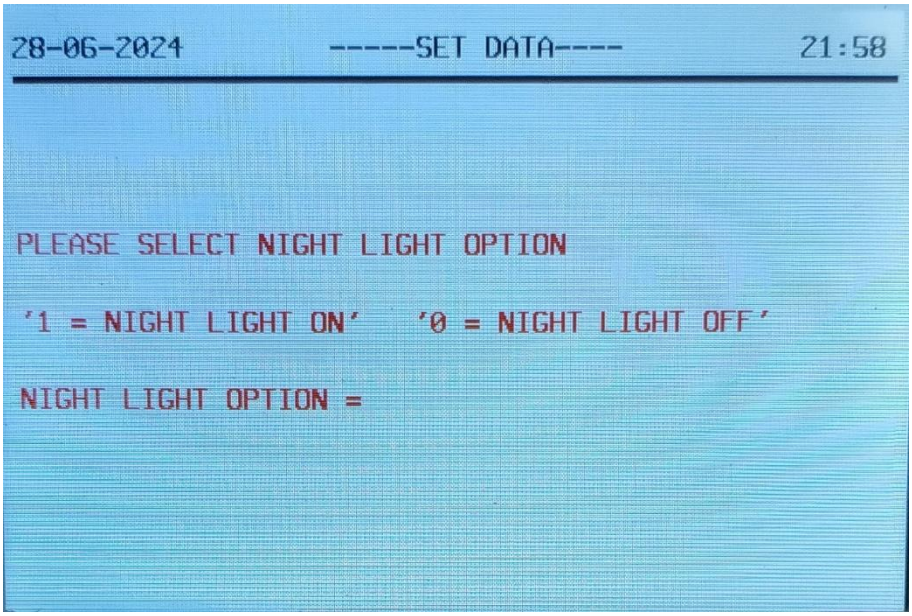
Εικόνα 2.11 : Εισαγωγή ώρας λειτουργίας ημέρας



Εικόνα 2.12 : Εισαγωγή λεπτών λειτουργίας ημέρας

Όπως φαίνεται και από τις εικόνες 2.9 – 2.12 γίνεται εισαγωγή πρώτα της ώρας της αλλαγής από λειτουργίας ημέρας σε λειτουργία νύχτας και μετά των λεπτών. Το ίδιο γίνεται και με την αλλαγή από λειτουργία νύχτας σε λειτουργία ημέρας. Πρώτα εισάγεται η ώρα και μετά τα λεπτά.

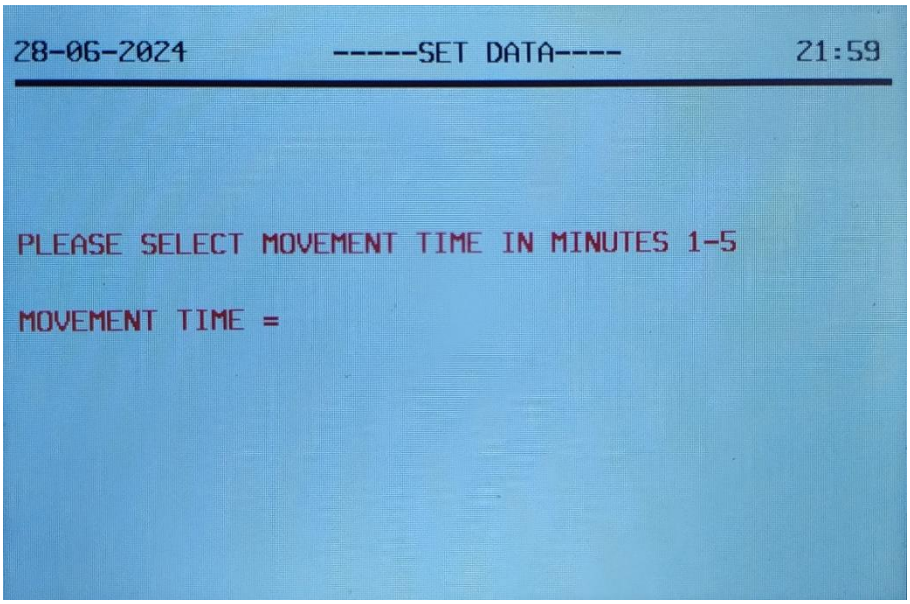
Στην επόμενη επιλογή '**PRESS 5 TO SET NIGHT LIGHT ON-OFF**' γίνεται η ενεργοποίηση ή η απενεργοποίηση του φωτός νύχτας (Εικόνα 2.13).



Εικόνα 2.13 : Επιλογή ενεργοποίησης απενεργοποίησης φωτός νύχτας

Επιλέγοντας την τιμή '**1**' γίνεται ενεργοποίηση της λειτουργίας φωτός νύχτας με αποτέλεσμα κατά την λειτουργία νύχτας αν ενεργοποιηθεί ο αισθητήρας κίνησης το φως θα ανάψει με μειωμένη φωτεινότητα. Επιλέγοντας την τιμή '**0**' γίνεται απενεργοποίηση της λειτουργίας με αποτέλεσμα το φως να μην ανάβει καθόλου ακόμα και όταν ανιχνευτεί κίνηση.

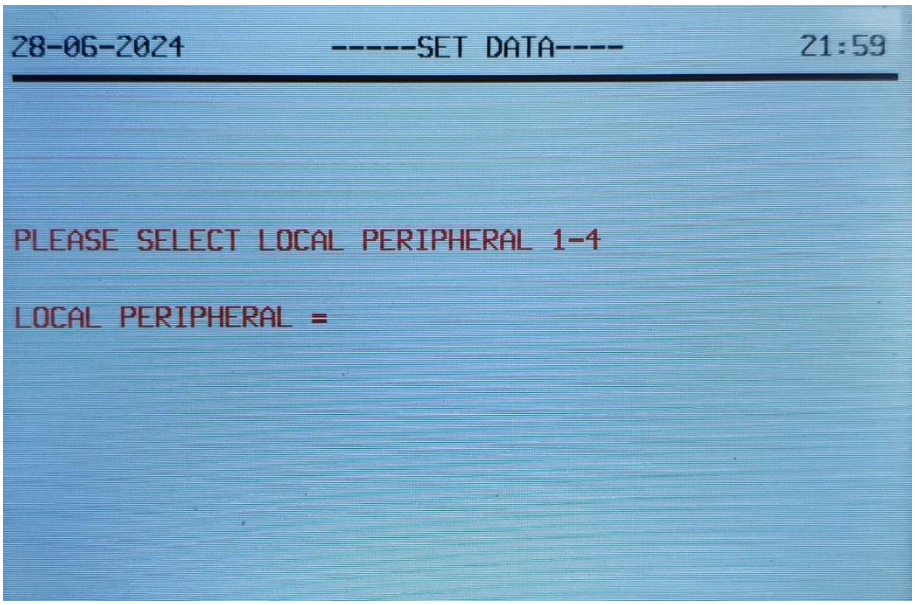
Η έβδομη επιλογή '**PRESS 6 TO SET MOVEMENT TIME**' επιτρέπει την ρύθμιση του χρόνου που μένει το φως αναμμένο μετά από την στιγμή που δεν ανιχνεύεται ανθρώπινη παρουσία (Εικόνα 2.14).



Εικόνα 2.14 : Επιλογή χρόνου ακινησίας

Εδώ γίνεται επιλογή της τιμής από 1 – 5 που αντιστοιχεί σε 1 – 5 λεπτά της ώρας τα οποία είναι ο χρόνος καθυστέρησης απενεργοποίησης του φωτισμού λόγο μη ανίχνευσης ανθρώπινης παρουσίας. Μετά από τον χρόνο αυτό σβήνει το φως.

Αφού έχουν γίνει όλες οι παραπάνω παραμετροποιήσεις απομένει να γίνει επιλογή του τοπικού ελεγκτή που θα είναι αποδέκτης όλων αυτών των ρυθμίσεων. Αυτό γίνεται με την όγδοη επιλογή '**PRESS B TO SELECT LOCAL PERIPHERAL AND SAVE**' (Εικόνα 2.15).



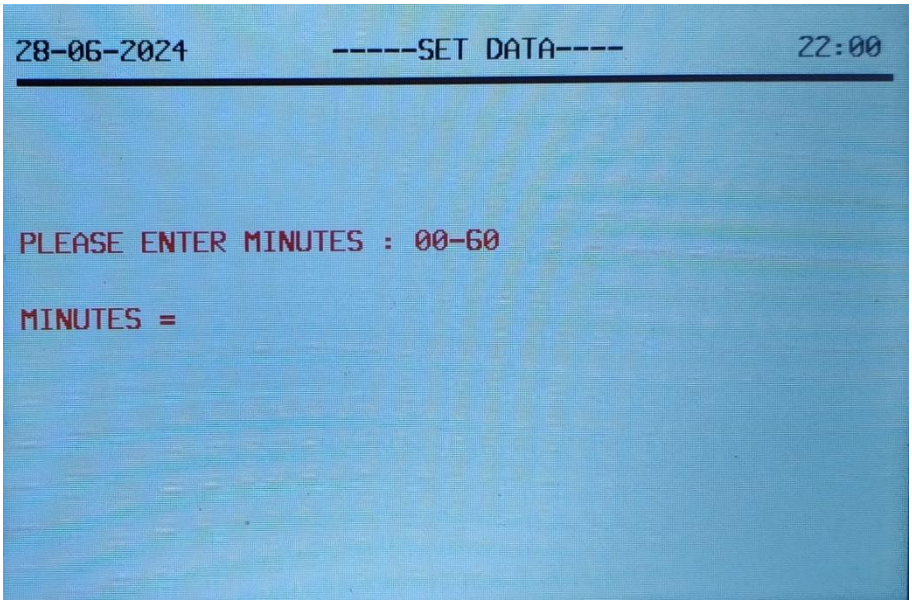
Εικόνα 2.15 : Επιλογή τοπικού ελεγκτή

Όπως φαίνεται και στην Εικόνα 2.15 οι επιλογές είναι 1 – 4. Διαλέγοντας μια από τις τέσσερις επιλογές και πατώντας οποιοδήποτε πλήκτρο μετά όλες οι ρυθμίσεις μεταφέρονται στον τοπικό ελεγκτή που επιλέχθηκε.

Με την τελευταία επιλογή ή αν δεν γίνει καμία επιλογή για ένα συγκεκριμένο χρονικό διάστημα, γίνεται μεταφορά στην αρχική οθόνη.

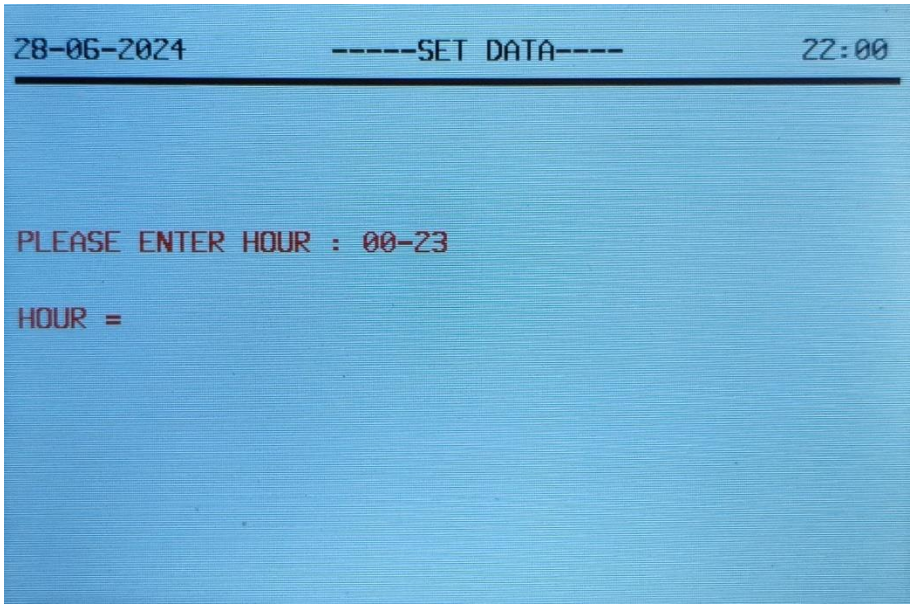
Όλες οι επιλογές και ρυθμίσεις που γίνονται στον κατάλογο ρύθμισης παραμέτρων των περιφερειακών συσκευών παραμένουν στην μνήμη του ελεγκτή μέχρι να αλλαχτούν ή να διακοπή η τροφοδοσία. Έτσι υπάρχει η δυνατότητα οι ίδιες επιλογές να μεταφερθούν και σε άλλους τοπικούς ελεγκτές χωρίς να χρειαστεί να γίνουν όλες ξανά από την αρχή παρά μόνο αυτές που θέλει να αλλάξει ο χρήστης.

Από τον κεντρικό κατάλογο (Εικόνα 2.4) η δεύτερη επιλογή '**PRESS B TO SET TIME AND DATE**' είναι για την ρύθμιση της ημερομηνίας και της ώρας του RTC (Εικόνες 2.16, 2.17, 2.18, 2.19, 2.20).



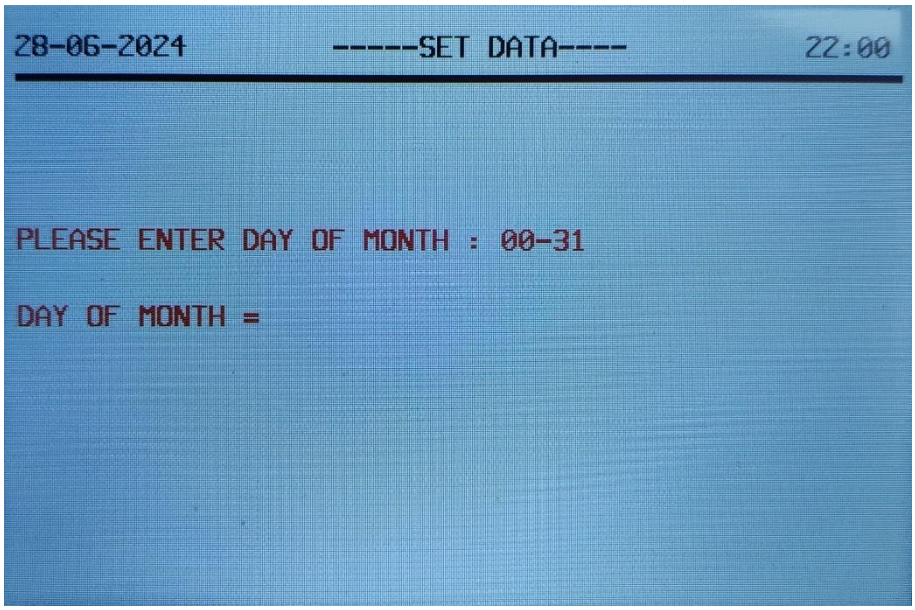
Εικόνα 2.16 : Εισαγωγή λεπτών της ώρας

Αρχικά γίνεται η ρύθμιση των λεπτών της ώρας (Εικόνα 2.16).



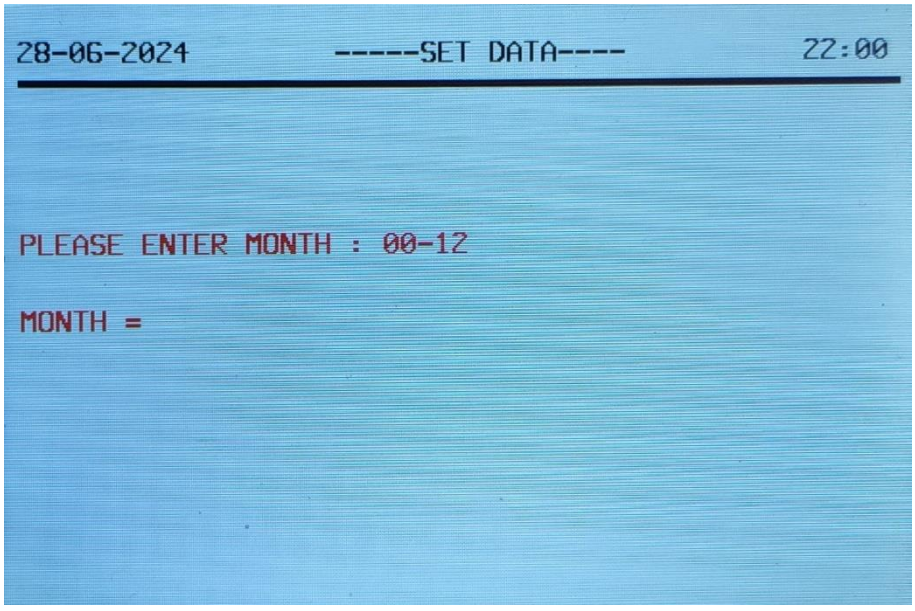
Εικόνα 2.17 : Εισαγωγή ώρας

Στην συνέχεια γίνεται η εισαγωγή της ώρας (Εικόνα 2.17)



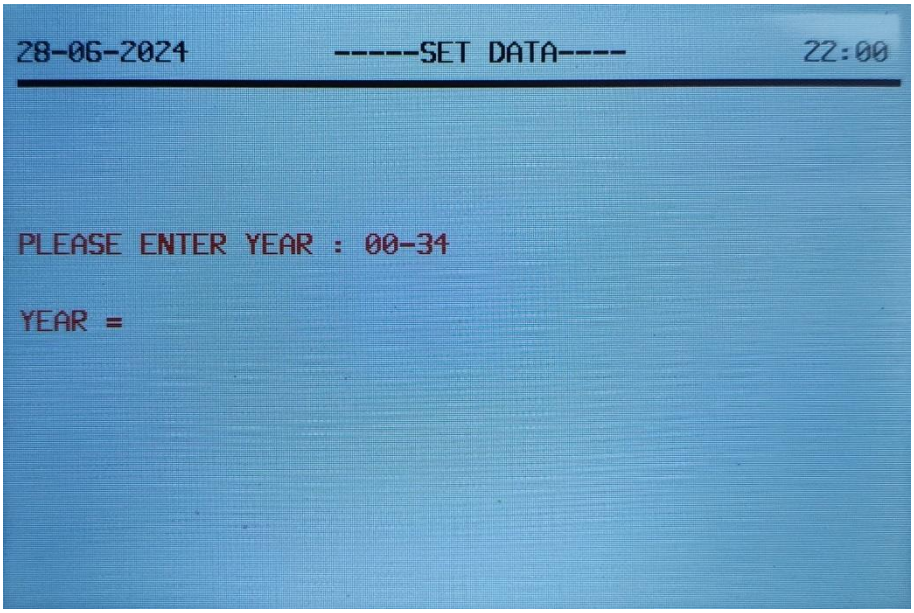
Εικόνα 2.18 : Εισαγωγή ημέρας του μήνα

Επόμενο βήμα είναι η εισαγωγή της ημέρας του μήνα (Εικόνα 2.18).



Εικόνα 2.19 : Εισαγωγή μήνα του έτους

Μετά γίνεται η εισαγωγή του μήνα του έτους (Εικόνα 2.19).



Εικόνα 2.20 : Εισαγωγή έτους

Τελευταία γίνεται η εισαγωγή του έτους (Εικόνα 2.20).

Κάθε φορά που γίνεται μια εισαγωγή (π.χ. μήνας του έτους) για να αποθηκευτή η εισαγωγή και να προχωρήσει στην επόμενη πρέπει να πατηθεί οποιοδήποτε πλήκτρο και αυτό πρέπει να γίνει μέσα σε χρονικό διάστημα 10 δευτερολέπτων. Αλλιώς το πρόγραμμα επανέρχεται στην αρχική οθόνη. Αυτό γίνεται για να μην ξεχάσει ο χρήστης για οποιονδήποτε λόγο και παραμείνει το πρόγραμμα κολλημένο στο συγκεκριμένο σημείο θέτοντάς το ουσιαστικά εκτός λειτουργίας.

Αυτές ήταν όλες οι παραμετροποιήσεις που γίνονται από τον κεντρικό ελεγκτή ο οποίος στην συνέχεια ανά τακτά χρονικά διαστήματα επικοινωνεί με τους τοπικούς ελεγκτές για να στείλει ότι ρυθμίσεις έχουν γίνει ή έχουν αλλαχτεί και να λάβει πληροφορίες από τους τοπικούς ελεγκτές. Από τον κεντρικό ελεγκτή γίνεται και η ρύθμιση της θερμοκρασίας του κάθε χώρου. Ο κεντρικός ελεγκτής παίρνει από κάθε τοπικό ελεγκτή την εντολή για την ενεργοποίηση ή απενεργοποίηση των ηλεκτροθερμικών κεφαλών στους χώρους που ελέγχει ο κάθε τοπικός ελεγκτής ρυθμίζοντας έτσι κατάλληλα και την θερμοκρασία.

2.3.2 Τοπικός Ελεγκτής

Ο τοπικός ελεγκτής είναι αυτός ο οποίος βρίσκεται σε κάθε χώρο που επιθυμεί ο χρήστης να παρακολουθεί. Ελέγχει συνεχώς τον χώρο για ανθρώπινη παρουσία με την χρήση αισθητήρα ανθρώπινης παρουσίας HLK-LD2420 [4] της εταιρίας Hi-Link, την θερμοκρασία και την υγρασία με τον αισθητήρα SHT30 [5] της εταιρίας SENSIRION, και τέλος τα παράθυρα και τις μπαλκονόπορτες μέσω μιας εισόδου του μικροελεγκτή STM32F030C8T6 [6]. Ανά τακτά χρονικά διαστήματα επικοινωνεί τόσο με τον κεντρικό ελεγκτή όσο και με τους τοπικούς δέκτες για ανανέωση των πληροφοριών. Ο τοπικός ελεγκτής (Εικόνα 2.21) αφού πάρει από τον κεντρικό ελεγκτή τις ρυθμίσεις θερμοκρασίας και υγρασίας που εισήγαγε ο χρήστης, τις συγκρίνει με αυτές από τον αισθητήρα και

αφού πάρει υπόψιν του και την παρουσία ανθρώπου ή όχι από τον αισθητήρα ανθρώπινης παρουσίας στέλνει εντολή στον κεντρικό ελεγκτή για ενεργοποίηση ή όχι των ηλεκτροθερμικών κεφαλών ρυθμίζοντας έτσι κατάλληλα την θερμοκρασία και υγρασία του χώρου που ελέγχει. Επίσης αν ανιχνεύσει ότι έχει ανοιχτεί κάποιο παράθυρο ή μπαλκονόπορτα στον χώρο τότε στέλνει εντολή στον κεντρικό ελεγκτή για απενεργοποίηση των ηλεκτροθερμικών κεφαλών έτσι ώστε να σταματήσει η άσκοπη κατανάλωση ενέργειας στον χώρο μέχρι να κλείσει το παράθυρο ή η μπαλκονόπορτα.

Εικόνα 2.21 : Τοπικός Ελεγκτής

2.3.3 Τοπικός Δέκτης

Οι τοπικοί δέκτες είναι αυτοί οι οποίοι εκτελούν τις εντολές του τοπικού και του κεντρικού ελεγκτή. Χωρίζονται σε τρείς μέχρι στιγμής κατηγορίες και είναι:

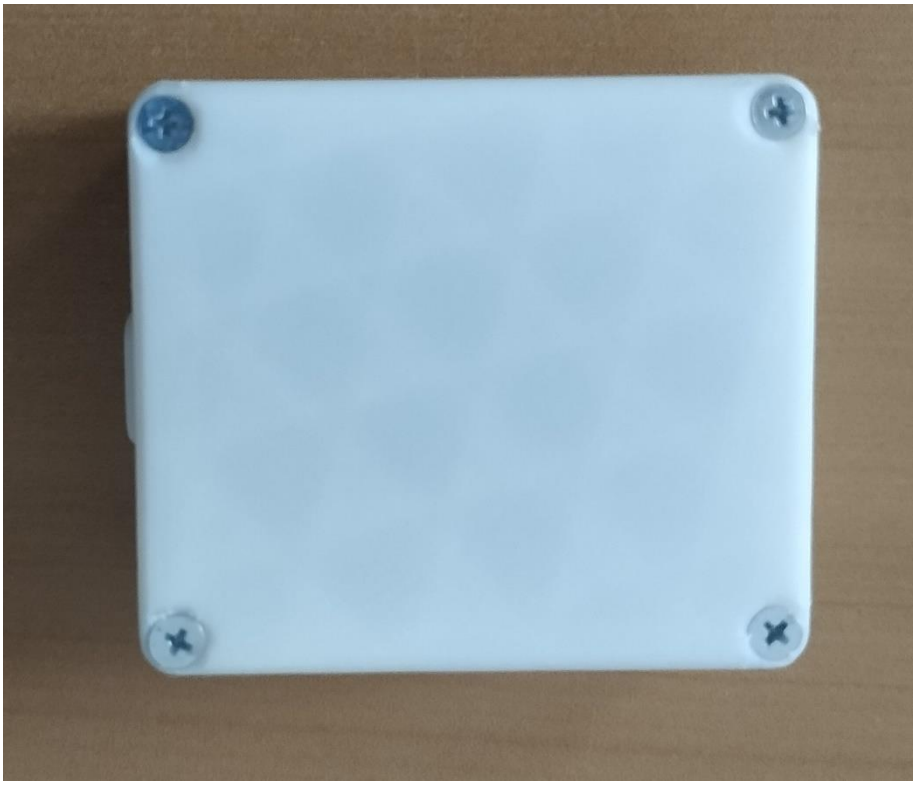
- Δέκτης διακόπτης με χρονοκαθυστέρηση συνήθως για τον φωτισμό του χώρου
- Δέκτης διακόπτης για τον αφυγραντήρα
- Δέκτης Dimmer με τον οποίο μπορούμε μετά από κάποια ώρα να έχουμε τα φώτα χαμηλωμένα

Ο δέκτης διακόπτης με χρονοκαθυστέρηση και ο δέκτης διακόπτης για τον αφυγραντήρα, εξωτερικά (το κουτί) και εσωτερικά (η πλακέτα) δεν διαφέρουν καθόλου. Αλλάζουν μόνο στον προγραμματισμό.

2.3.3.1 Δέκτης διακόπτης με χρονοκαθυστέρηση

Ο δέκτης διακόπτης με χρονοκαθυστέρηση (Εικόνες 2.22, 2.23, 2.24) είναι ένας διακόπτης (με ηλεκτρονόμο) ο οποίος όταν ο τοπικός δέκτης ανιχνεύσει ανθρώπινη παρουσία στέλνει εντολή ενεργοποίησης στον δέκτη και αυτός με την σειρά του ενεργοποιεί έναν ηλεκτρονόμο. Μόλις ο άνθρωπος φύγει από τον ελεγχόμενο χώρο τότε με εντολή του τοπικού δέκτη απενεργοποιεί με χρονοκαθυστέρηση τον ηλεκτρονόμο. Η χρονοκαθυστέρηση είναι ρυθμιζόμενη από τον κεντρικό

ελεγκτή και μπορεί να ρυθμιστεί από ένα έως πέντε λεπτά με βήμα ενός λεπτού. Χρησιμοποιείτε συνήθως για το άναμμα και το σβήσιμο των φώτων. Μπορεί να χρησιμοποιηθεί και για τον έλεγχο οποιασδήποτε συσκευής θέλουμε.



Εικόνα 2.22 : Δέκτης Διακόπτης (πάνω όψη)

Εικόνα 2.23 : Δέκτης Διακόπτης (πλαϊνή όψη)

Εικόνα 2.24 : Δέκτης Διακόπτης (κάτω όψη)

2.3.3.2 Δέκτης διακόπτης για τον αφυγραντήρα

Ο δέκτης διακόπτης για τον αφυγραντήρα ελέγχεται από τον τοπικό ελεγκτή και ανάλογα με την ρύθμιση που έχουμε δώσει στον κεντρικό ελεγκτή ενεργοποιεί ή απενεργοποιεί τον αφυγραντήρα για να πετύχουμε την επιθυμητή υγρασία στον χώρο που ελέγχει ο τοπικός ελεγκτής. Επίσης θα μπορούσε να ελέγχει και κάποιον εξαερισμό αν υπήρχε στον συγκεκριμένο χώρο. Και αυτός στην εμφάνιση (Εικόνες 2.22, 2.23, 2.24) και το hardware μοιάζει με τον δέκτη διακόπτη χωρίς χρονοκαθυστέρηση.

2.3.3.3 Δέκτης Dimmer

Ο δέκτης με την δυνατότητα μείωσης φωτισμού μετά από κάποια ώρα που καθορίζεται μέσω του κεντρικού ελεγκτή, αντί να μην ενεργοποιεί τα φώτα ακόμα και όταν ο τοπικός ελεγκτής ανιχνεύσει την παρουσία ανθρώπου, αυτός τα ενεργοποιεί αλλά χαμηλωμένα. Αυτή η λειτουργία είναι χρήσιμη το βράδυ όταν για παράδειγμα σηκωθούμε από το κρεββάτι και πάμε στην κουζίνα για νερό, αντί να ανάψουν τα φώτα στο μέγιστο να ανάψουν χαμηλωμένα. Εξωτερικά η διαφορά με τους προηγούμενους δέκτες διακόπτες είναι πολύ μικρή (Εικόνα 2.25).

Εικόνα 2.25 : Δέκτης Dimmer

Στο hardware (πλακέτα) όπως και στο πρόγραμμα οι διαφορές είναι πολλές όπως θα δούμε παρακάτω.

2.4 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν τα μέρη από τα οποία αποτελείται το σύστημα εξοικονόμησης ενέργειας και τον τρόπο με τον οποίο λειτουργεί το κάθε ένα από αυτά. Έγινε μια μικρή αναφορά στον τρόπο που επικοινωνούν μεταξύ τους έτσι ώστε να μην υπάρχουν απώλειες δεδομένων με ταυτόχρονη εκπομπή από δύο ή και περισσότερους πομποδέκτες. Επίσης αναλύθηκαν όλες οι παραμετροποιήσεις που μπορούν να γίνουν στον κεντρικό ελεγκτή για την λειτουργία του συστήματος καθώς επίσης και όλες τις ενδείξεις που μπορεί να παρακολουθεί ο χρήστης από την οθόνη του κεντρικού ελεγκτή.

Κεφάλαιο 3ο: Σχεδιασμός και Κατασκευή Συστήματος

3.1 Εισαγωγή

Στόχος του συγκεκριμένου συστήματος είναι η εξοικονόμηση ενεργειακών πόρων σε μια κατοικία. Οι ενεργειακοί πόροι που προσπαθεί να εξοικονομήσει το σύστημα είναι το Φ.Α. που χρησιμοποιείται για την θέρμανση και το ηλεκτρικό ρεύμα που χρησιμοποιείται για τον φωτισμό και όπου αλλού χρειάζεται. Για την θέρμανση ένα 7% περίπου μπορεί να εξοικονομηθεί μειώνοντας την θέρμανση κατά έναν βαθμό. Φυσικά δεν πρόκειται να μειωθεί γενικά ένας βαθμός αλλά μόνο σε χώρους όπου δεν υπάρχει ανθρώπινη παρουσία για όσο χρονικό διάστημα δεν υπάρχει κατά έναν ή και δύο βαθμούς. Επίσης σε αυτούς τους χώρους γίνεται εξοικονόμηση ηλεκτρικής ενέργειας κόβοντας την παροχή σε συσκευές που δεν χρειάζεται να λειτουργούν για όσο χρονικό διάστημα δεν είναι κάποιος εκεί.

Το σύστημα χωρίζεται σε τέσσερα μέρη, τον κεντρικό ελεγκτή για την εισαγωγή και παρακολούθηση των δεδομένων, τον τοπικό ελεγκτή ο οποίος έχει και τους αισθητήρες για όλους τους ελέγχους του χώρου, τους τοπικούς δέκτες που είναι στην ουσία οι διακόπτες για τον έλεγχο των συσκευών και την τελική βαθμίδα ελέγχου των ηλεκτροθερμικών κεφαλών που ελέγχεται από τον κεντρικό ελεγκτή μέσω καλωδίου.

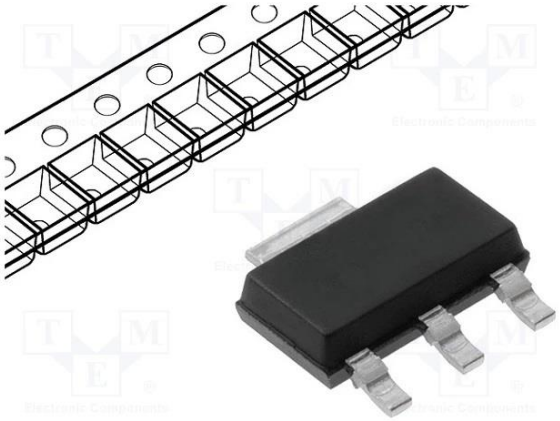
3.2 Ανάλυση Κεντρικού Ελεγκτή

Ο κεντρικός ελεγκτής αποτελείται από :

- Τροφοδοτικό 3,3V 1A
- Οθόνη TFT 3.5” RGB 65K color
- Πομποδέκτης NRF24L01+
- Πληκτρολόγιο 4X4
- RTC DS3231
- Κύκλωμα οδήγησης ηλεκτροθερμικών κεφαλών
- Μικροελεγκτή STM32F401RBT6

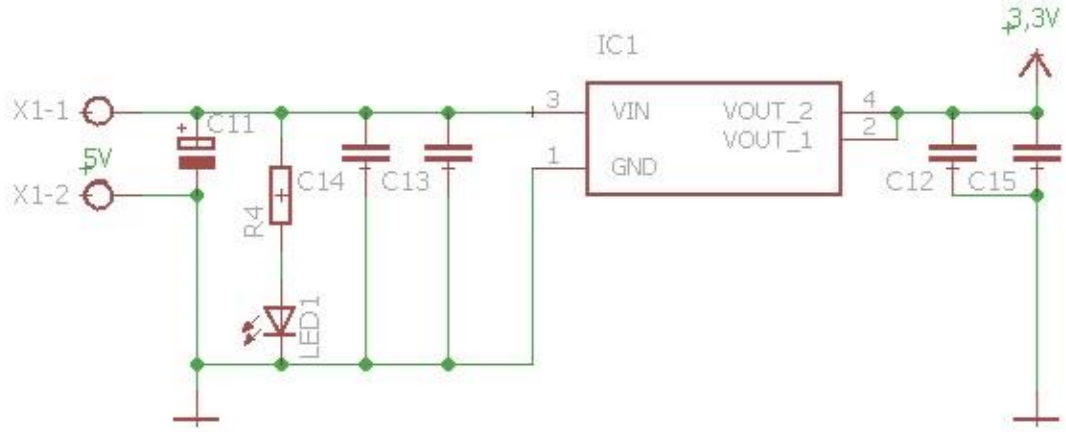
3.2.1 Τροφοδοτικό 3,3V 1A

Η τάση τροφοδοσίας του κυκλώματος είναι 3,3V τα οποία παρέχονται από το ολοκληρωμένο **AP2114H-3.3TRG1 [7]** (Εικόνα 3.1). Είναι ένας σταθεροποιητής τάσης LDO χωρίς ρύθμιση με σταθερή τάση εξόδου 3,3V, 1A ρεύμα εξόδου και περίβλημα SOT223. Για την σταθερή λειτουργία του χρειάζεται έναν πυκνωτή 4,7μF στην είσοδό του και έναν ίδιο στην έξοδό του.



Εικόνα 3.1 : AP2114H-3.3TRG1

Το σχηματικό διάγραμμα του τροφοδοτικού φαίνεται στο Σχήμα 3.1.



Σχήμα 3.1 : Σχηματικό διάγραμμα τροφοδοσίας Κεντρικού Ελεγκτή

Στα σημεία X1-1 και X1-2 έχουμε την είσοδο της τροφοδοσίας που είναι 5V και μπορούν να προέρχονται από φορτιστές κινητών με usb type c καλώδιο ή ακόμα και από κάποιο powerbank ή οποιοδήποτε τροφοδοτικό 5V με καλώδιο usb type c. Οι C12 και C14 είναι οι απαραίτητοι πυκνωτές 4,7μF σύμφωνα με το datasheet του ολοκληρωμένου. Όλα τα εξαρτήματα του κεντρικού ελεγκτή τροφοδοτούνται από 3,3V και ο λόγος που επιλέχθηκε αυτός ο σταθεροποιητής είναι για την χαμηλή τιμή του (0,206 € με Φ.Π.Α.) και για τη χαμηλή πτώση τάσης εισόδου εξόδου (LDO) που είναι 0,45V και μας δίνει την δυνατότητα να τροφοδοτηθεί το κύκλωμα και από επαναφορτιζόμενες μπαταρίες τύπου 18650 που έχουν και μεγάλη χωρητικότητα (Εικόνα 3.2).

Εικόνα 3.2 : Μπαταρία τύπου 18650

3.2.2 Οθόνη TFT 3,5” RGB 65K color

Η απεικόνιση όλων των παραμέτρων που εισάγει ο χρήστης στον κεντρικό ελεγκτή καθώς επίσης και η παρακολούθηση όλων των δεδομένων που στέλνονται στον κεντρικό ελεγκτή γίνεται σε μια οθόνη TFT 3,5” [8] (Εικόνα 3.3).

Εικόνα 3.3 : Οθόνη TFT 3.5” RGB 65K color

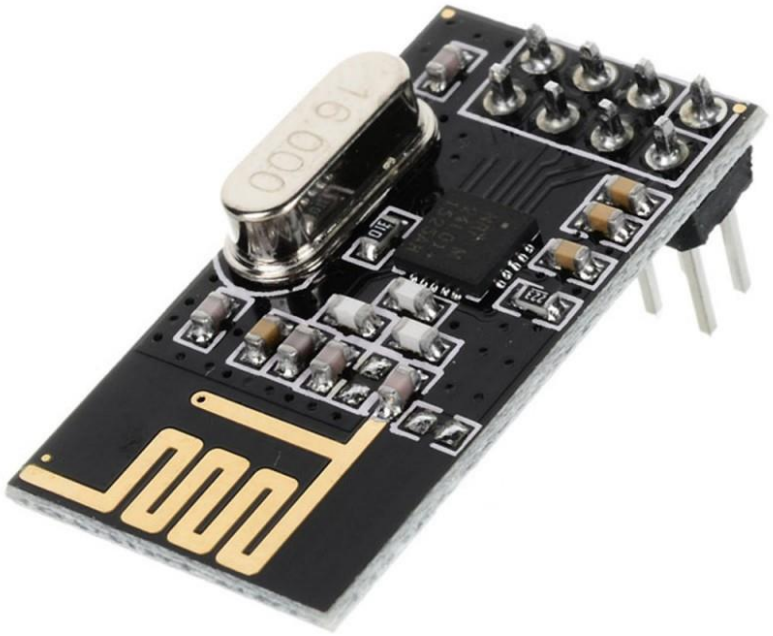
Τα κυριότερα χαρακτηριστικά της οθόνης είναι :

- Μέγεθος : 3,5”
- Τύπος : TFT
- Χρώματα οθόνης : RGB 65K
- Ανάλυση : 480*320
- Ολοκληρωμένο οδήγησης : ILI9488
- Τύπος διασύνδεσης : 4-wire SPI
- Τάση τροφοδοσίας : 3.3V~5V
- Μέγεθος : 56.34x98(mm)
- Κόστος : περίπου 15€

3.2.3 Πομποδέκτης NRF24L01+

Η επικοινωνία μεταξύ όλων των μονάδων του συστήματος γίνεται με τον πομποδέκτη NRF24L01+ [3] της εταιρίας NORDIC SEMICONDUCTOR (Εικόνες 3.4 και 3.5). Το NRF24L01+ είναι ένα ολοκληρωμένο κύκλωμα το οποίο είναι ένας πλήρης πομποδέκτης ο οποίος λειτουργεί στην συχνότητα των 2,4GHz.

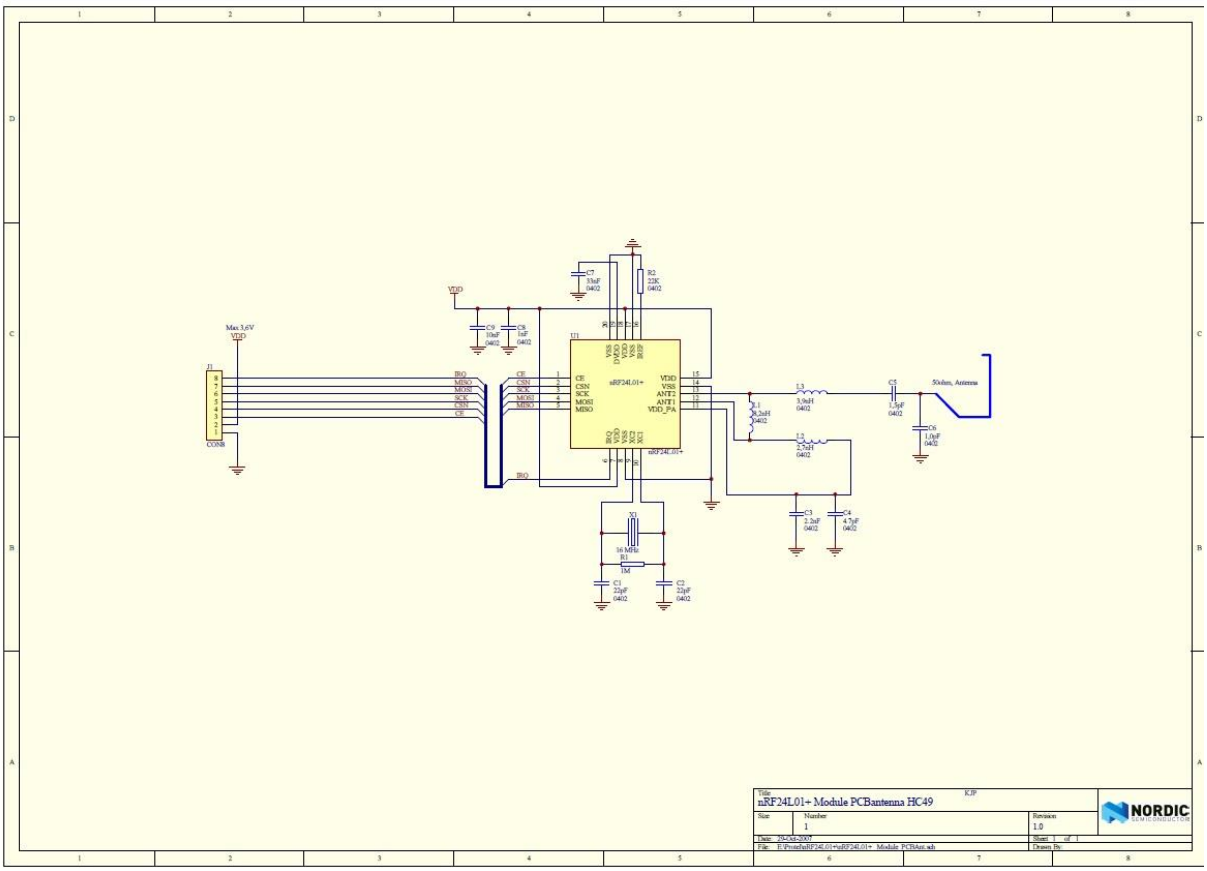
Εικόνα 3.4 : Πομποδέκτης NRF24L01+ με ενισχυτή



Εικόνα 3.5 : Πομποδέκτης NRF24L01+

Όπως φαίνεται και από τις εικόνες 3.4 και 3.5 υπάρχουν δύο μονάδες NRF24L01+, η απλή που φαίνεται στην εικόνα 3.5 και αυτή με ενισχυτή όπως αυτή της εικόνας 3.4. Στον κεντρικό ελεγκτή χρησιμοποιείται η μονάδα με τον ενισχυτή. Η εμβέλεια της μονάδας χωρίς ενισχυτή σε ανοιχτό μέρος και χωρίς παρεμβολές είναι 20 έως 30 μέτρα αλλά μέσα στο σπίτι μετά από δοκιμές διαπιστώθηκε ότι δεν ξεπερνά τα 5 μέτρα. Για τον λόγο αυτόν για τον κεντρικό ελεγκτή όπως και για τους τοπικούς ελεγκτές χρησιμοποιήθηκαν οι μονάδες με ενισχυτή. Κατά την διάρκεια των δοκιμών πολλά δεδομένα δεν φτάνανε στον προορισμό τους. Πιθανότερος λόγος είναι οι παρεμβολές της συχνότητας (2,4GHz) στην οποία λειτουργεί ο πομποδέκτης και κάποιοι τσιμεντένιοι τοίχοι που υπήρχαν ανάμεσα. Οι μονάδες με ενισχυτή έχουν εμβέλεια σε ανοιχτό χώρο και χωρίς παρεμβολές (σύμφωνα με τα δεδομένα που δίνουν οι προμηθευτές) που φτάνει και τα 1000 μέτρα.

Το σχηματικό διάγραμμα που δίνει η εταιρία είναι στο σχήμα 3.2.



Σχήμα 3.2 : Σχηματικό διάγραμμα NRF24L01+

Τα κυριότερα χαρακτηριστικά είναι :

- Είναι και πομπός και δέκτης σε ένα πολύ μικρό ολοκληρωμένο κύκλωμα
- Λειτουργεί στην ελεύθερη συχνότητα 2,4GHz
- 1,9 – 3,6V τάση λειτουργίας
- Χαμηλή κατανάλωση ρεύματος
- Έχει 126 κανάλια επικοινωνίας
- Ταχύτητα μετάδοσης δεδομένων 1 και 2Mbps
- Έως 32 byte δεδομένων μπορεί να στείλει και να λάβει
- 4-pin SPI επικοινωνία με μικροελεγκτή
- Εύκολο στην χρήση και τον προγραμματισμό
- Κόστος NRF24L01+ χωρίς ενισχυτή 2,5€
- Κόστος NRF24L01+ με ενισχυτή 6,8€

3.2.4 Πληκτρολόγιο 4X4

Για την εισαγωγή των παραμέτρων του χρήστη στο σύστημα χρησιμοποιήθηκε ένα πληκτρολόγιο μεμβράνης 4X4 (Εικόνα 3.6).

Εικόνα 3.6 : Πληκτρολόγιο μεμβράνης 4X4

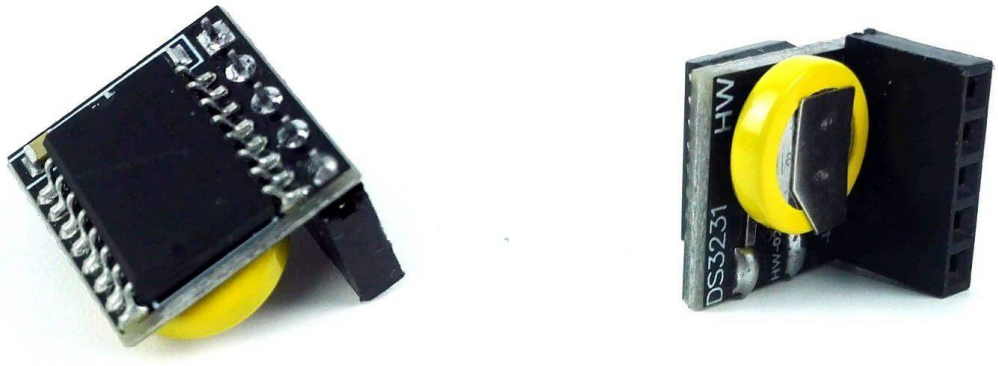
Τα κυριότερα χαρακτηριστικά του πληκτρολογίου είναι:

- Αντίσταση επαφής : 10Ω ~ 500Ω
- Rebound time : 1 (ms)
- Μέγιστη τάση επαφών : 35VDC
- Μέγιστο ρεύμα επαφών : 100mA
- Μήκος καλωδιοταινίας : 85mm
- Αριθμός ακίδων ακροδέκτη : 8 ακίδες
- Απόσταση ακίδων : 2,54mm
- Κόστος : 1,5€

3.2.5 RTC DS3231

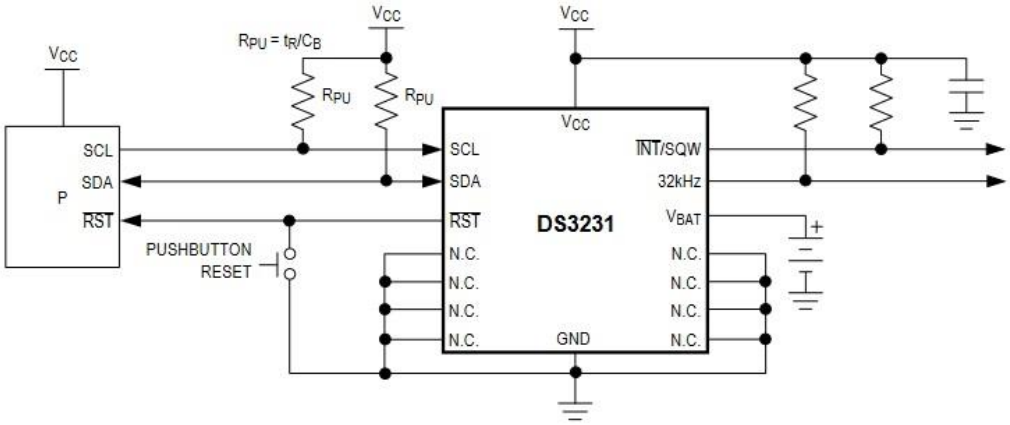
Στην παράμετρο DAY-NIGHT του συστήματος ό χρήστης πρέπει να εισάγει την ώρα που επιθυμεί το σύστημα να λειτουργεί με παραμέτρους νύχτας και την ώρα που επιθυμεί το σύστημα να λειτουργεί με παραμέτρους ημέρας. Για να γίνει αυτό θα πρέπει και το σύστημα να διατηρεί την σωστή ώρα ακόμα και αν διακοπεί η τροφοδοσία του. Αυτό γίνεται με την χρήση RTC κυκλώματος και στο συγκεκριμένο σύστημα γίνεται με το DS3231 [9] της εταιρίας Maxim Integrated.

Το DS3231 (Εικόνα 3.7) είναι ένα υψηλής ακριβείας ρολόι, με ενσωματωμένο κρύσταλλο, χαμηλού κόστους και επικοινωνεί με τον μικροελεγκτή μέσω του διαύλου I2C. Οι παράμετροι που διατηρεί είναι τα δευτερόλεπτα, λεπτά, ώρα, ημέρα της εβδομάδας, ημέρα του μήνα, τον μήνα και τον χρόνο. Επίσης αυτόματα ρυθμίζει την τελευταία μέρα του μήνα ανάλογα τον μήνα και υπάρχει πρόβλεψη για τον δίσεκτο χρόνο. Ακόμα έχει και δύο προγραμματιζόμενες υπενθυμίσεις (ξυπνητήρια). Τέλος έχει ενσωματωμένο αισθητήρα θερμοκρασίας που βοηθάει στην διατήρηση της ακρίβειας του χρόνου και τροφοδοτείται από τάση 3,3V.



Εικόνα 3.7 : RTC DS3231

Το DS3231 για να λειτουργήσει δεν χρειάζεται πολλά περιφερειακά υλικά όπως φαίνεται και από τυπικό σχηματικό διάγραμμα σχήμα 3.3 που δίνει η εταιρία.

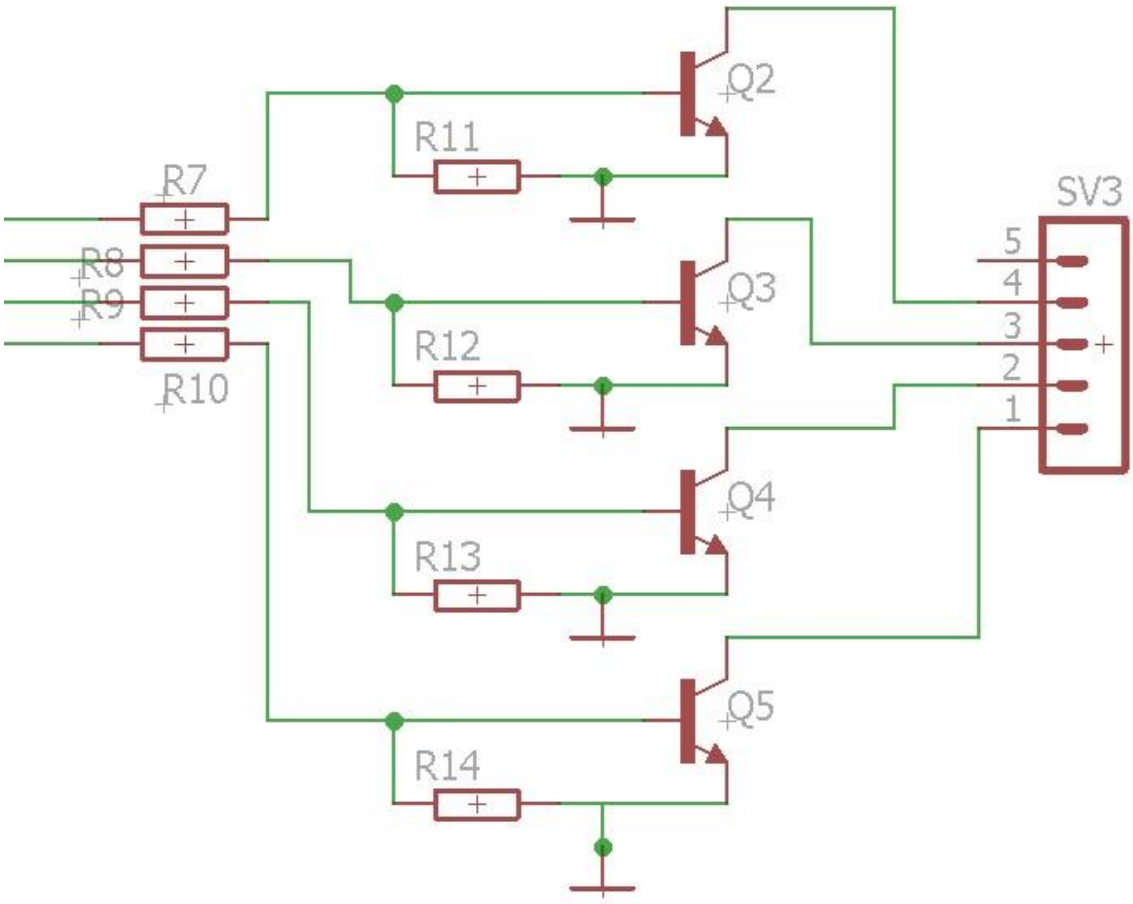


Σχήμα 3.3 : Σχηματικό διάγραμμα DS3231

Πέρα από όλα τα χαρακτηριστικά που αναφερθήκαν ποιο πάνω το κόστος είναι χαμηλό και είναι ίσο με 3,1€.

3.2.6 Κύκλωμα οδήγησης ηλεκτροθερμικών κεφαλών

Ο έλεγχος της θερμοκρασίας του χώρου γίνεται από τους τοπικούς ελεγκτές του κάθε χώρου. Ο τοπικός ελεγκτής συγκρίνει την θερμοκρασία του αισθητήρα που διαθέτει με την θερμοκρασία που εισήγαγε ο χρήστης στο σύστημα και στέλνει το αποτέλεσμα στον κεντρικό ελεγκτή. Αν ο χώρος χρειάζεται να θερμανθεί περισσότερο τότε ο κεντρικός ελεγκτής ενεργοποιεί τον ηλεκτρονόμο, στην τελική βαθμίδα ελέγχου των ηλεκτροθερμικών κεφαλών, που ελέγχει την αντίστοιχη ηλεκτροθερμική κεφαλή του αντίστοιχου σώματος θέρμανσης. Αντίθετα αν ο χώρος έχει φτάσει στην θερμοκρασία που ρυθμίστηκε τότε ο κεντρικός ελεγκτής απενεργοποιεί τον ηλεκτρονόμο, στην τελική βαθμίδα ελέγχου των ηλεκτροθερμικών κεφαλών, που ελέγχει την αντίστοιχη ηλεκτροθερμική κεφαλή του αντίστοιχου σώματος θέρμανσης. Το κύκλωμα οδήγησης των ηλεκτροθερμικών κεφαλών φαίνεται στο σχήμα 3.4.

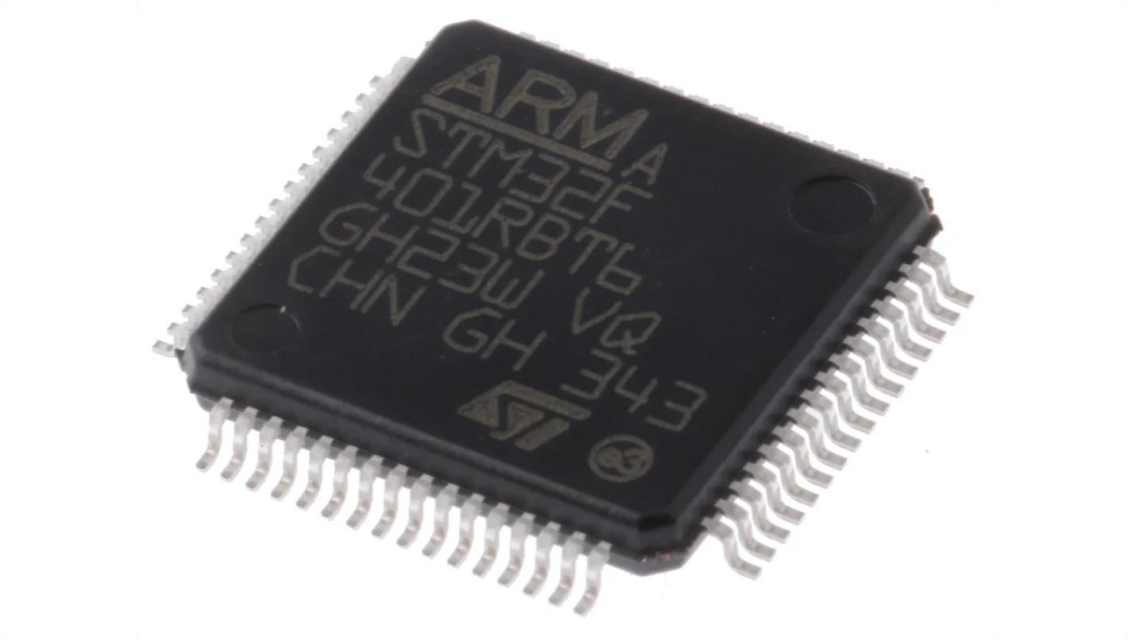


Σχήμα 3.4 : Σχηματικό διάγραμμα κυκλώματος οδήγησης ηλεκτροθερμικών κεφαλών

Οι αντιστάσεις R7 – R10 είναι συνδεμένες σε τέσσερις εξόδους του μικροελεγκτή ελέγχοντας έτσι τα τρανζίστορ Q2 – Q5 που με την σειρά τους ελέγχουν τους ηλεκτρονόμους στην τελική βαθμίδα ελέγχου των ηλεκτροθερμικών κεφαλών.

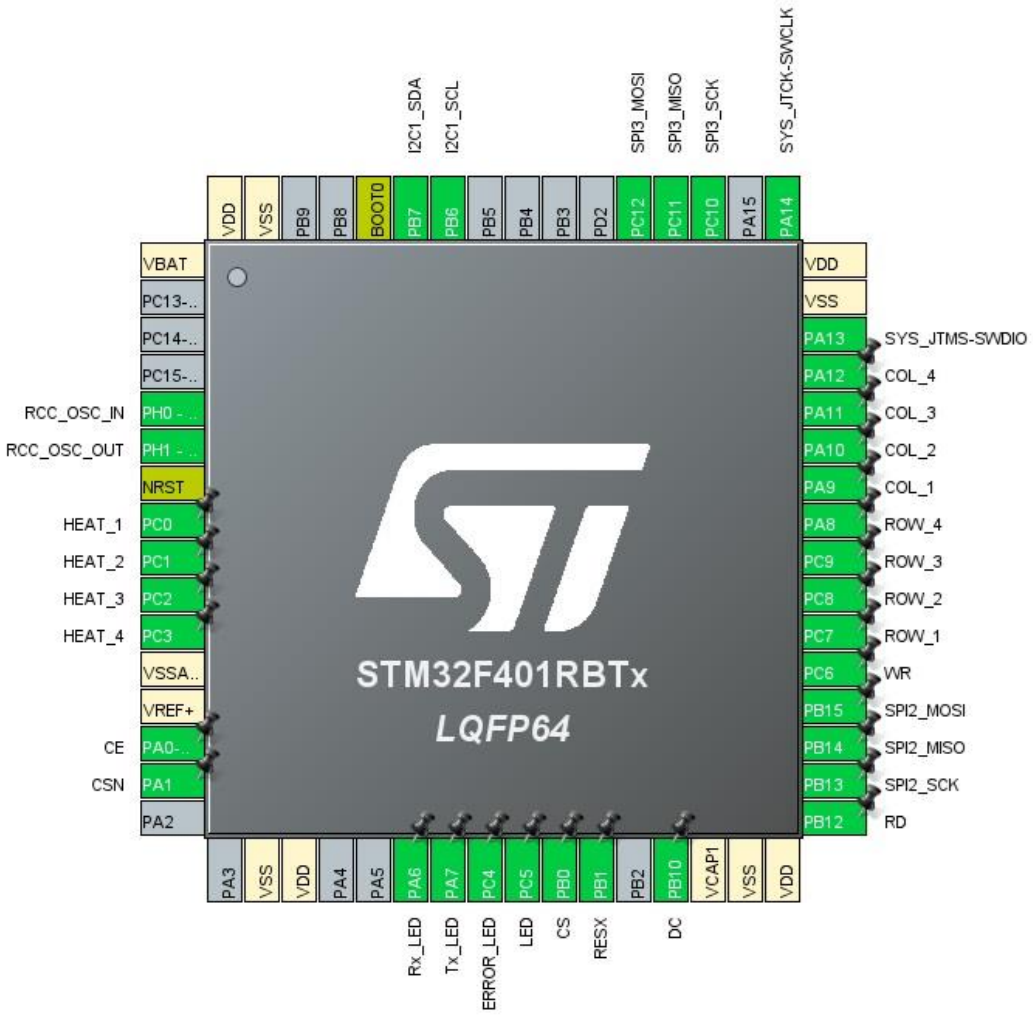
3.2.7 Μικροελεγκτής STM32G401RBT6

Ο έλεγχος όλου του κυκλώματος του κεντρικού ελεγκτή γίνεται από τον μικροεπεξεργαστή της εταιρίας STMicroelectronics STM32F401RBT6 [10] (Εικόνα 3.8).



Εικόνα 3.8 : STM32F401RBT6

Ο STM32F401RBT6 είναι ένας μικροεπεξεργαστής υψηλής απόδοσης, ανήκει στην οικογένεια Arm Cortex-M4, 32-bit RISC core (32-bit πυρήνα με μειωμένο αριθμό εντολών) και λειτουργία πυρήνα με συχνότητα έως 84 MHz. Περιλαμβάνει μνήμη Flash 128 Kbytes και SRAM 64 Kbytes τα οποία είναι απαραίτητα για την αποθήκευση των εικονιδίων της οθόνης. Βρίσκεται σε περίβλημα LQFP64 και τα περισσότερα ποδαράκια είναι σε χρήση όπως φαίνεται στο σχήμα 3.5.

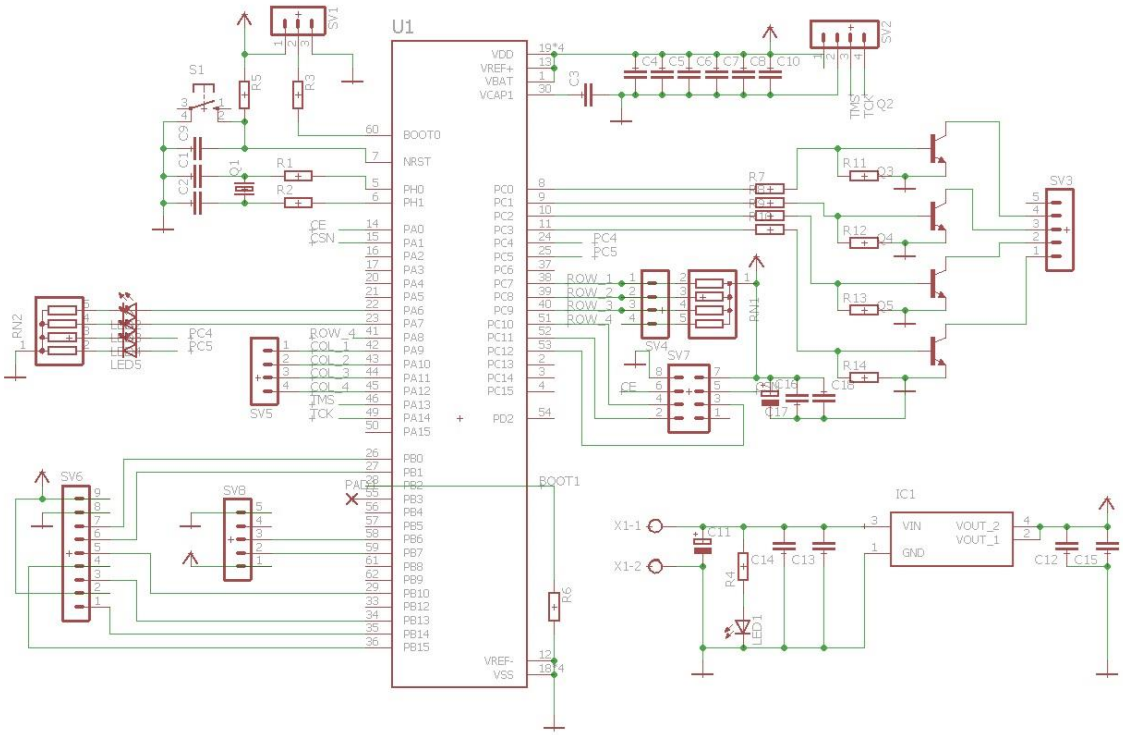


Σχήμα 3.5 : Συνδέσεις STM32F401RBT6

Τα κυριότερα χαρακτηριστικά του μικροελεγκτή STM32F401RBT6 είναι:

- Τροφοδοσία λειτουργίας : 1,7V – 3.6V
- Συχνότητα λειτουργίας : max 84 MHz
- Πυρήνας : Arm® 32-bit Cortex®-M4
- Μνήμη Flash : 128 Kbytes
- Μνήμη SRAM : 64 Kbytes
- Timers : 7 General, 1 Advance
- SPI : 3
- I2C : 3
- USART : 3
- USB : 1
- GPIO : 50
- Κόστος : 7,4€
- Serial wire debug (SWD) & JTAG interfaces

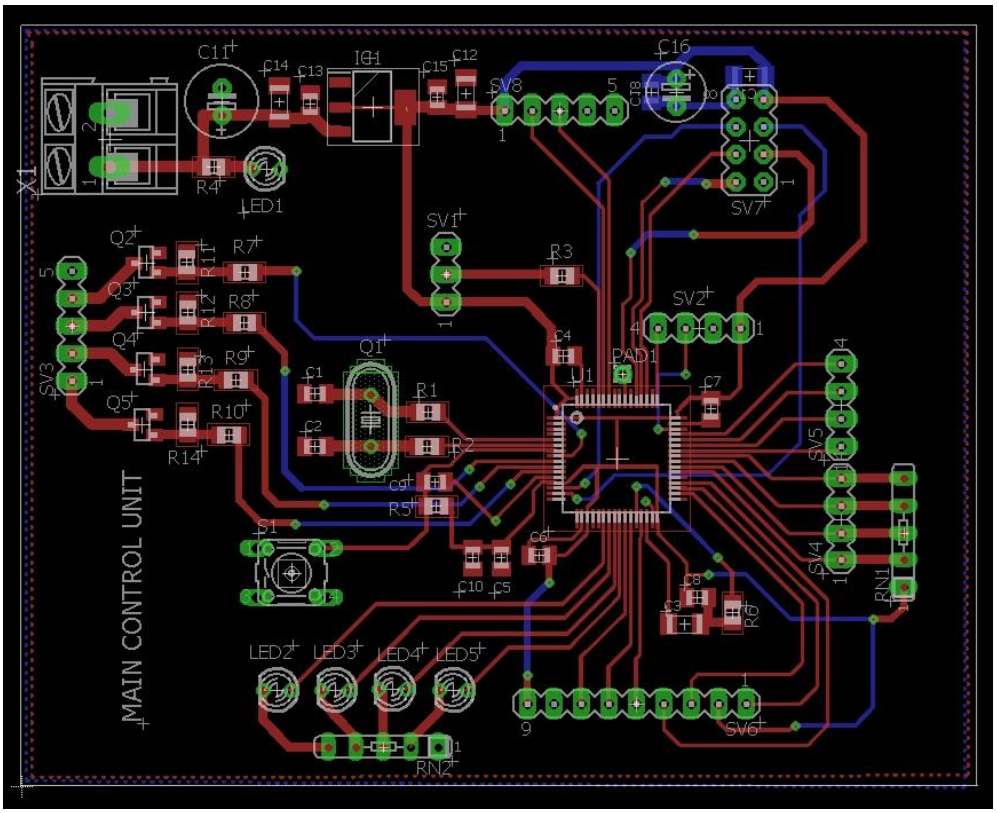
Το σχηματικό διάγραμμα του κυκλώματος του κεντρικού ελεγκτή φαίνεται στο σχήμα 3.6.



Σχήμα 3.6 : Σχηματικό διάγραμμα κεντρικού ελεγκτή

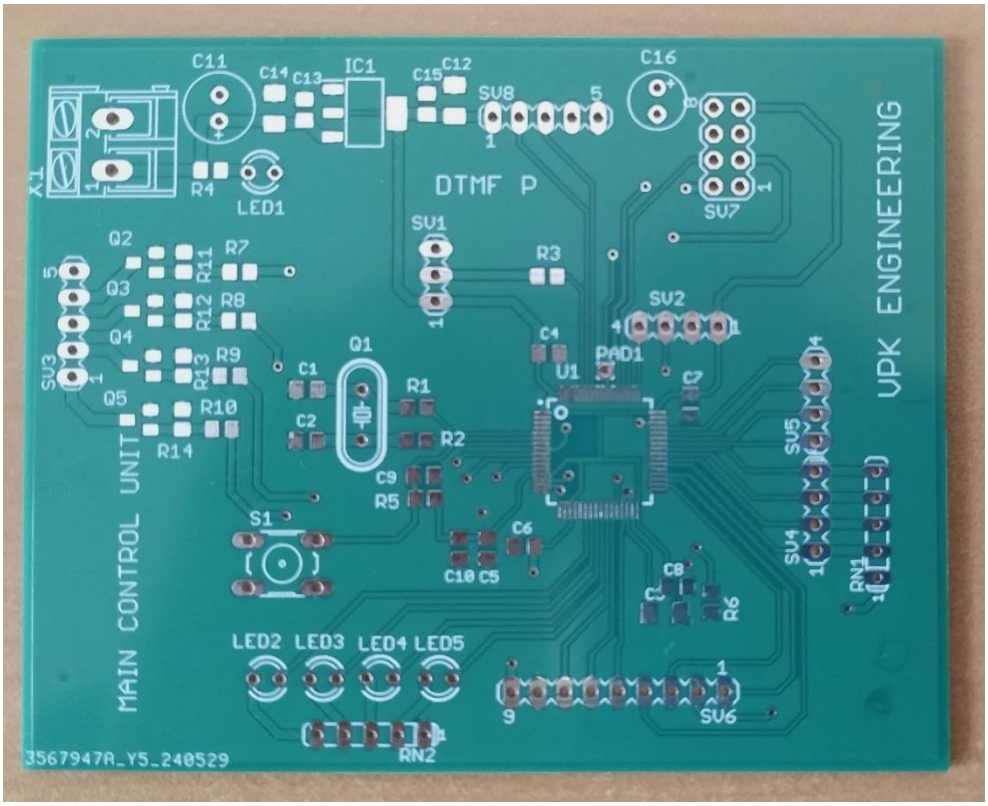
Στο σχηματικό διάγραμμα ξεχωρίζουν ο μικροελεγκτής, το τροφοδοτικό, το κυκλώματος οδήγησης των ηλεκτροθερμικών κεφαλών, ο κρύσταλλος 8 MHz και τα LED για τον έλεγχο λειτουργιών του κυκλώματος. Η οθόνη συνδέεται στον μικροελεγκτή μέσω του ακροδέκτη SV6, ενώ το RTC μέσω του ακροδέκτη SV8. Ο πομποδέκτης NRF24L01+ συνδέεται μέσω του ακροδέκτη SV7 και το πληκτρολόγιο συνδέεται μέσω του ακροδέκτη SV5 και του ακροδέκτη SV4. Ο ακροδέκτης SV2 είναι για τον προγραμματισμό του μικροελεγκτή μέσω του αποσφαλματωτή/προγραμματιστή (debugger/programmer) ST-LINK/V2 (Εικόνα).

Η πλακέτα του κυκλώματος φαίνεται στην εικόνα 3.9 όπως αυτή φαίνεται μαζί με τα εξαρτήματα στο σχεδιαστικό πρόγραμμα.

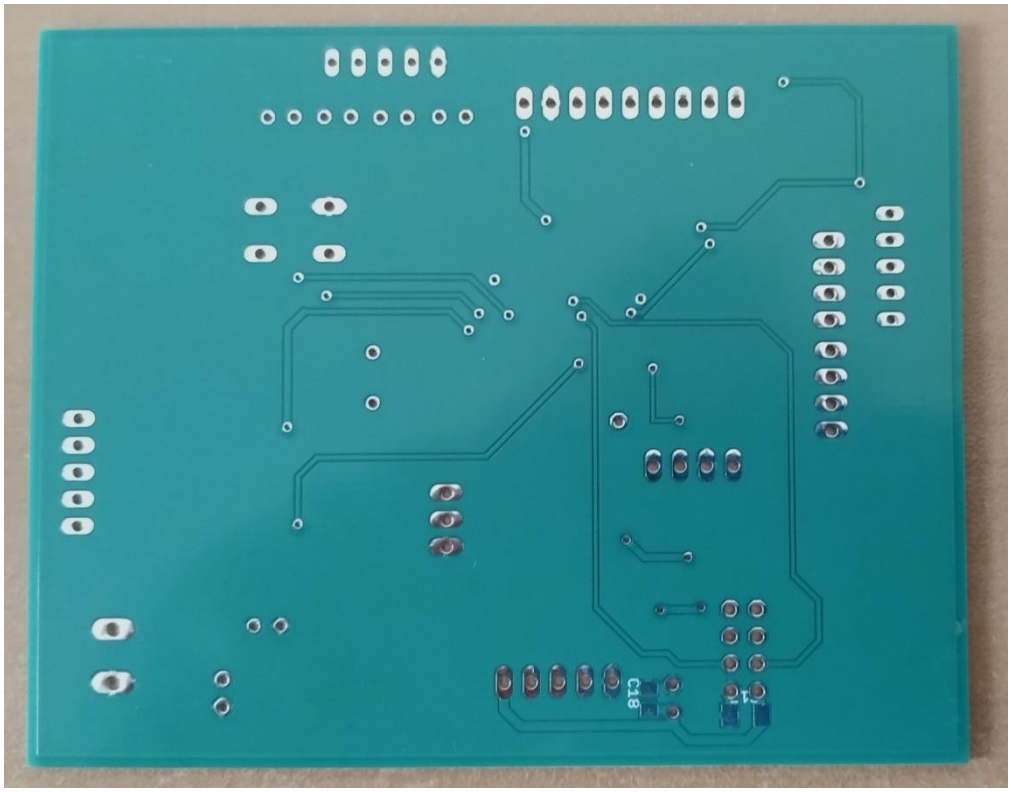


Εικόνα 3.9 : Σχεδίαση πλακέτας κεντρικού ελεγκτή

Η πλακέτα πριν την συναρμολόγηση φαίνεται στις εικόνες 3.10 και 3.11.

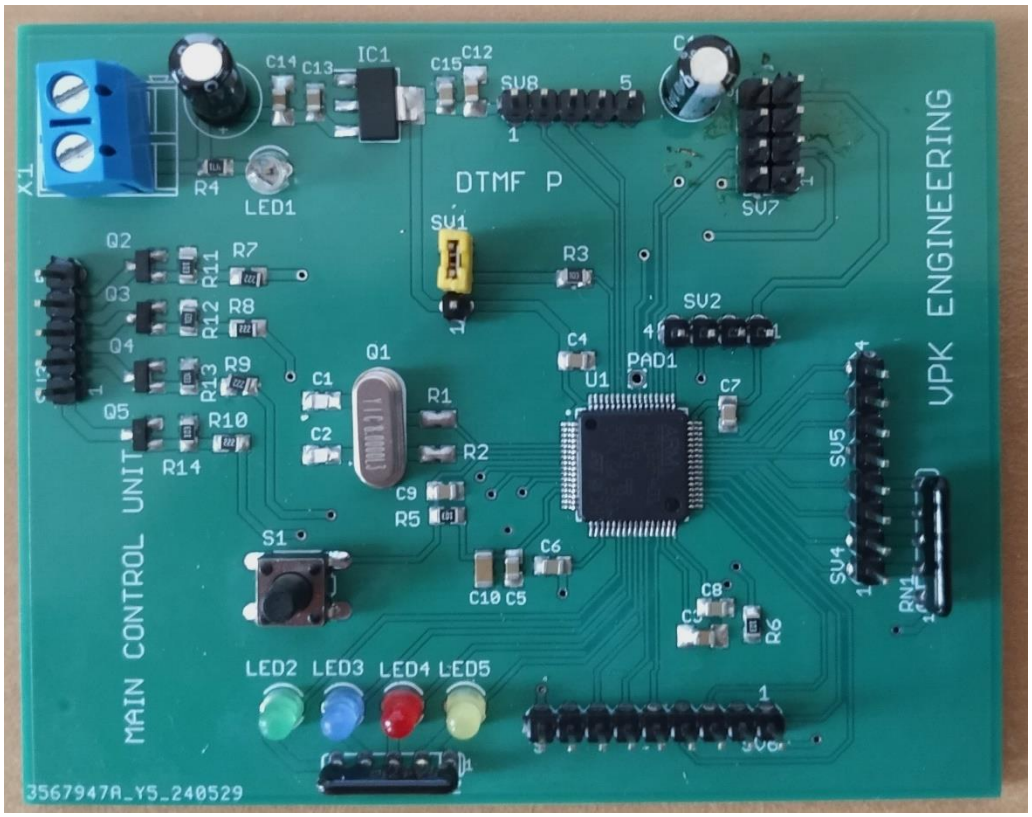


Εικόνα 3.10 : Πάνω όψη της πλακέτας



Εικόνα 3.11 : Κάτω όψη της πλακέτας

Η πλακέτα μετά την συναρμολόγηση φαίνεται στην εικόνα 3.12.



Εικόνα 3.12 : Πλακέτα μετά την συναρμολόγηση

3.3 Ανάλυση Τοπικού Ελεγκτή

Ο τοπικός ελεγκτής αποτελείται από :

- Τροφοδοτικό 3,3V 1A
- Πομποδέκτης NRF24L01+
- Αισθητήρας ανθρώπινης παρουσίας HLK-LD2420
- Αισθητήρα θερμοκρασίας και υγρασίας SHT-30
- Μικροελεγκτή STM32F030C8T6

3.3.1 Τροφοδοτικό 3,3V 1A

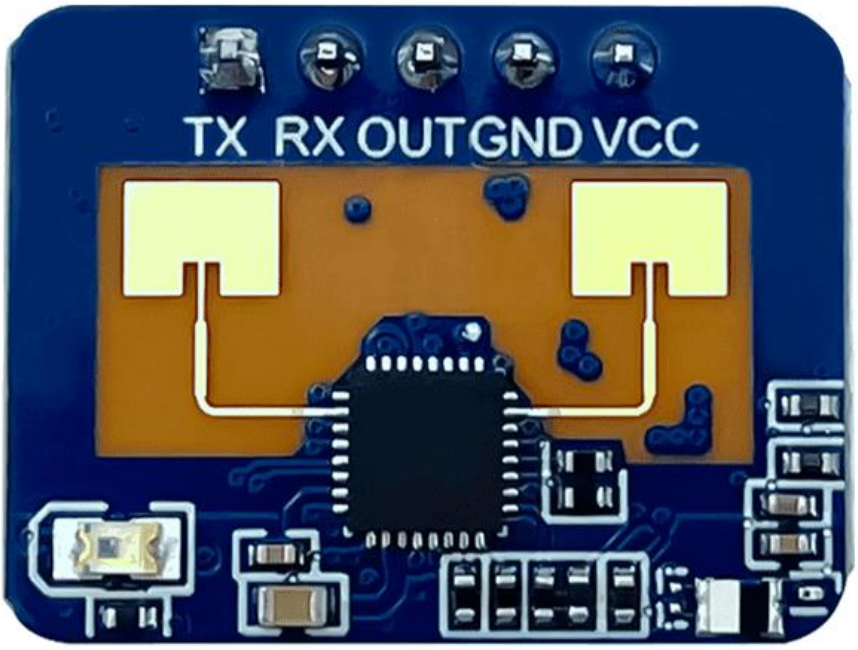
Το τροφοδοτικό του τοπικού ελεγκτή είναι ίδιο με αυτό του κεντρικού ελεγκτή και έχει αναλυθεί εκεί οπότε δεν χρειάζεται περεταίρω ανάλυση.

3.3.2 Πομποδέκτης NRF24L01+

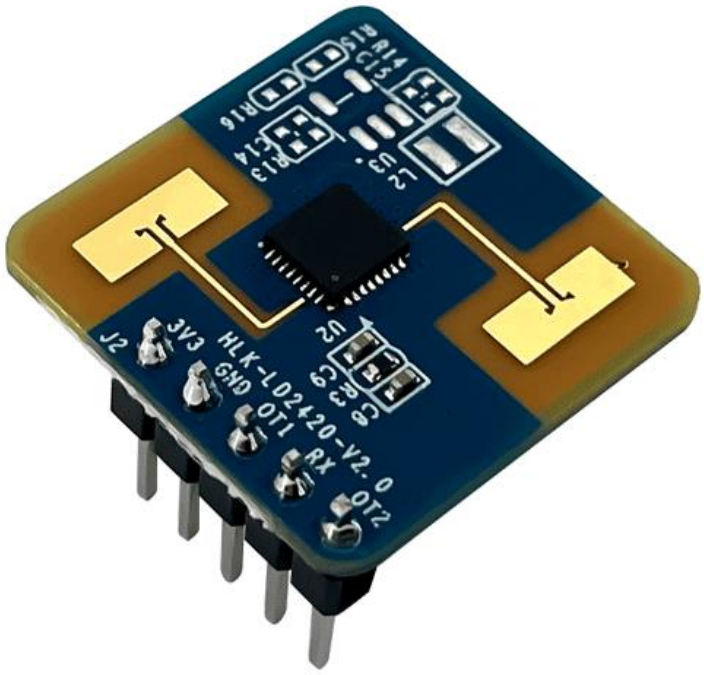
Ο πομποδέκτης NRF24L01+ του τοπικού ελεγκτή είναι και αυτός ίδιος με του κεντρικού ελεγκτή και είναι αυτός με τον ενισχυτή.

3.3.3 Αισθητήρας ανθρώπινης παρουσίας HLK-LD2420

Οι αισθητήρες που δοκιμαστήκανε στον τοπικό ελεγκτή είναι δύο, ο HLK-LD2410C [11] (Εικόνα 3.13) και ο HLK-LD2420 [4] (Εικόνα 3.14). Και οι δύο είναι από την ίδια εταιρία την Hi-Link.



Εικόνα 3.13 : HLK-LD2410C



Εικόνα 3.14 : HLK-LD2420

Τα τεχνικά χαρακτηριστικά του αισθητήρα HLK-LD2420 είναι καλύτερα από του αισθητήρα HLK-LD2410C αλλά ο λόγος που προτιμήθηκε ο πρώτος είναι διότι ο HLK-LD2410C ανεβάζει αρκετή θερμοκρασία σε σχέση με τον HLK-LD2420. Το μόνο πλεονέκτημα του HLK-LD2410C σε σχέση με τον HLK-LD2420 είναι ότι ο πρώτος διαθέτει και Bluetooth για την επικοινωνία με το πρόγραμμα ρύθμισης των παραμέτρων των δύο αισθητήρων.

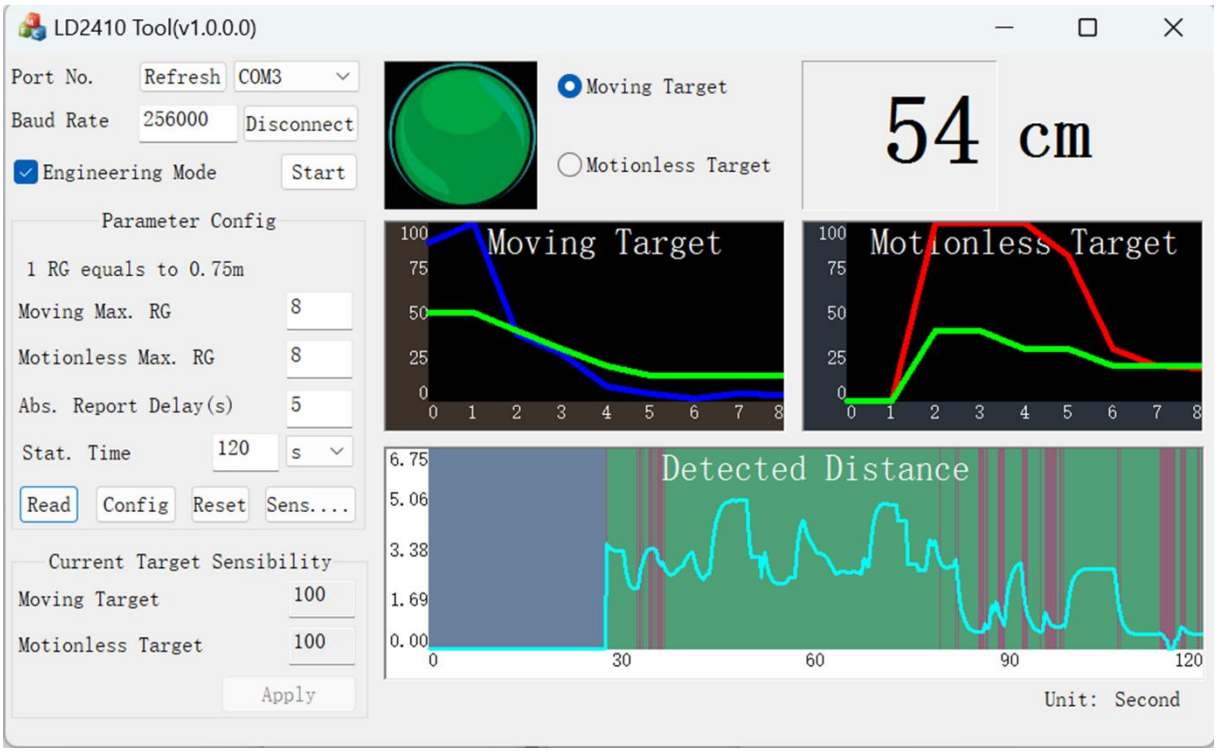
Ο αισθητήρας HLK-LD2420 είναι ένα υψηλής απόδοσης ραντάρ στα 24 GHz με ξεχωριστή κεραία εκπομπής και ξεχωριστή κεραία λήψης. Η ανεπτυγμένη τεχνολογία επεξεργασίας σήματος των ολοκληρωμένων κυκλωμάτων της σειράς S3 πετυχαίνουν με ακρίβεια την κίνηση ή την πολύ μικρή κίνηση του ανθρώπινου σώματος.

Ο αισθητήρας χρησιμοποιείται κυρίως σε εσωτερικούς χώρους για την ανίχνευση κίνησης ή πολύ μικρής κίνησης ανθρώπινου σώματος.

Η μέγιστη απόσταση ανίχνευσης είναι 8 μέτρα και μπορεί να παραμετροποιηθεί μέσω ειδικού προγράμματος που παρέχεται.

Έχει και σειριακή έξοδο αλλά και παρέχει σήμα 0V όταν δεν ανιχνεύει τίποτα και 3V όταν έχει ανιχνεύσει κίνηση.

Το πρόγραμμα ρύθμισης των παραμέτρων φαίνεται στην εικόνα 3.15.



Εικόνα 3.15 : HLK-LD2410, HLK-LD2420 Tool

Αυτό το πρόγραμμα επικοινωνεί με τον αισθητήρα μέσω της UART θύρας και μας επιτρέπει να αλλάξουμε την ευαισθησία και την περιοχή ανίχνευσης του αισθητήρα.

Τα κυριότερα χαρακτηριστικά του αισθητήρα HLK-LD2420 είναι:

- Τάση λειτουργίας : 3 - 3,6V
- Ρεύμα λειτουργίας : 50mA
- Διαστάσεις : 20 mm × 20 mm
- Συχνότητα λειτουργίας : 24 GHz
- Γωνία ανίχνευσης : ±60°
- Μέγιστη απόσταση ανίχνευσης κίνησης : 8 μέτρα
- Μέγιστη απόσταση ανίχνευσης μικρής κίνησης : 6 μέτρα
- Κόστος : 5€

3.3.4 Αισθητήρας θερμοκρασίας και υγρασίας SHT30

Ο αισθητήρας θερμοκρασίας και υγρασίας που επιλέχθηκε για το συγκεκριμένο σύστημα είναι ο SHT30 [5] της εταιρίας Sensirion. Η εικόνα του αισθητήρα φαίνεται στην εικόνα 3.16.

Εικόνα 3.16 : SHT30 Sensirion

Λόγο του πολύ μικρού μεγέθους του αισθητήρα 2.5X2.5mm αγοράστηκε έτοιμο κολλημένο σε μια μικρή πλακέτα (Εικόνα 3.17).

Εικόνα 3.17 : SHT30

Ο αισθητήρας έχει ψηφιακή έξοδο και επικοινωνεί με τον μικροελεγκτή με τον δίαυλο I2C με δύο διευθύνσεις επικοινωνίας και με ταχύτητες έως 1 MHz. Έχει πολύ μεγάλη ακρίβεια στην μέτρηση της θερμοκρασίας στο εύρος που χρειάζεται και επίσης πολύ καλή ακρίβεια στην μέτρηση της σχετικής υγρασίας. Η κατανάλωση είναι πολύ μικρή και το εύρος τροφοδοσίας είναι από 2,15V μέχρι 5,5V.

Τα κυριότερα χαρακτηριστικά του αισθητήρα είναι:

- Τάση λειτουργίας : 2,15V – 5,5V
- Μέγιστο ρεύμα κατανάλωσης : 1,5mA κατά την μέτρηση
- Επικοινωνία : I2C 1MHz max
- Ακρίβεια μέτρησης θερμοκρασίας: ±0.2 °C από 0°C μέχρι 65°C
- Ακρίβεια μέτρησης υγρασίας : ±2% από 20% μέχρι 80%
- Κόστος : 3€

3.3.5 Μικροελεγκτής STM32F030C8T6

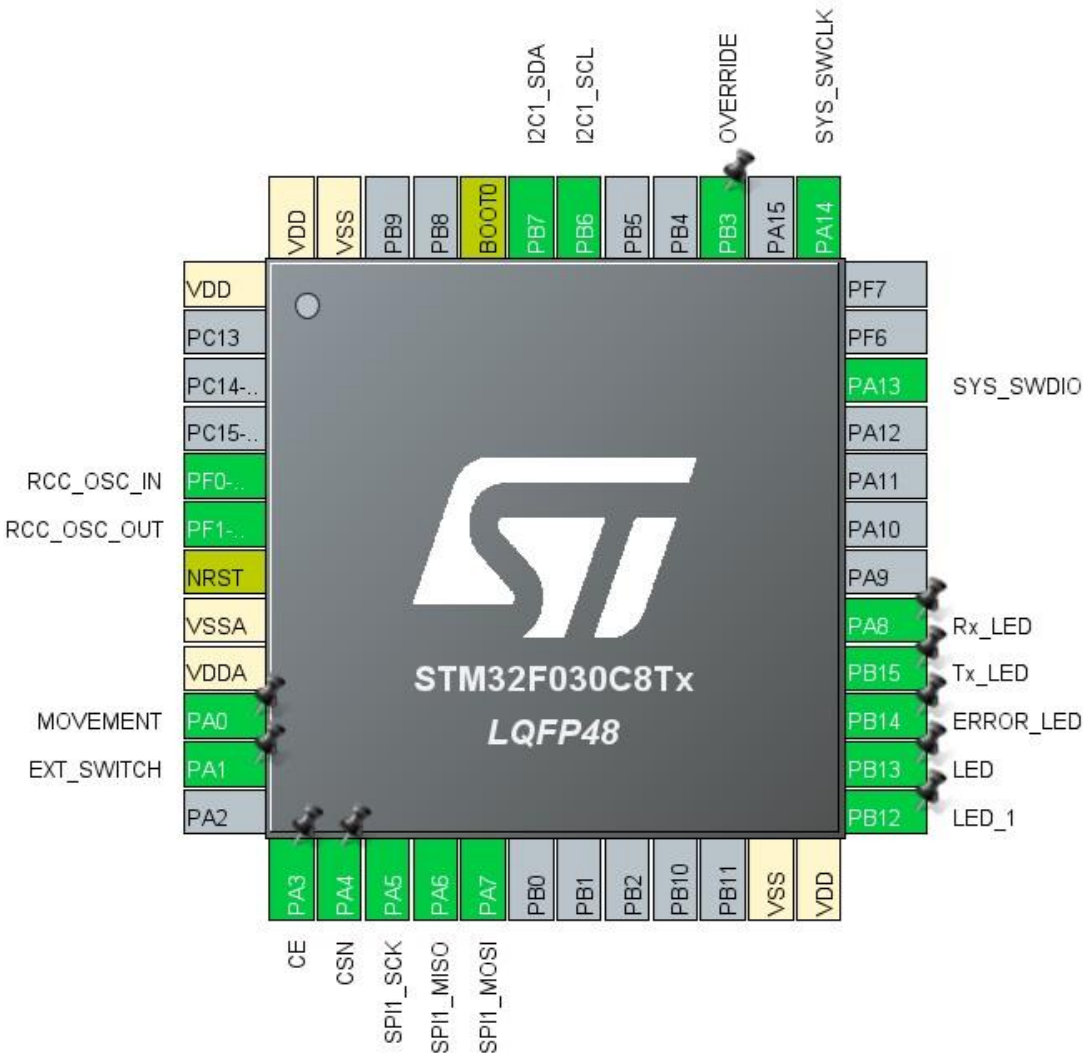
Για την λειτουργία του τοπικού ελεγκτή υπεύθυνος είναι ο μικροελεγκτής STM32F030C8T6 [6] της εταιρίας STMicroelectronics (Εικόνα 3.18).

Εικόνα 3.18 : STM32F030C8T6

Ο STM32F030C8T6 είναι ένας μικροεπεξεργαστής μεσαίας κατηγορίας, ανήκει στην οικογένεια Arm Cortex-M0, 32-bit RISC core (32-bit πυρήνα με μειωμένο αριθμό εντολών) και λειτουργία πυρήνα

με συχνότητα έως 48 MHz. Διατίθεται σε περίβλημα LQFP48 και διαθέτει 39 εισόδους εξόδους. Επιλέχθηκε αυτός ο μικροελεγκτής διότι για την λειτουργία που χρειάζεται είναι αρκετός, ενώ ο προηγούμενος ο STM32F401RBT6 είναι υπερβολικά μεγάλος σε δυνατότητες για αυτήν την χρήση.

Στο σχήμα 3.7 βλέπουμε τις συνδέσεις του μικροελεγκτή.



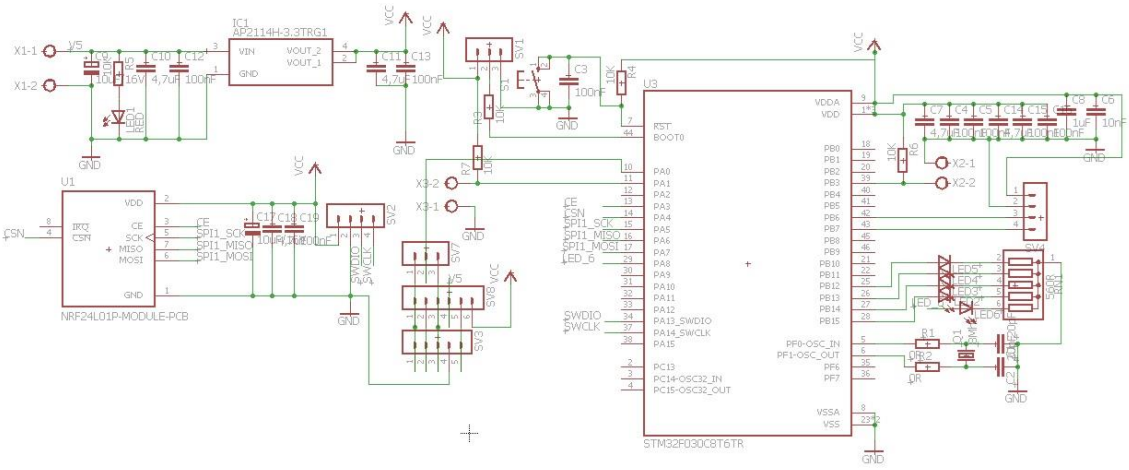
Σχήμα 3.7 : Συνδέσεις STM32F030C8T6

Τα κυριότερα χαρακτηριστικά του μικροελεγκτή STM32F030C8T6 είναι:

- Τροφοδοσία λειτουργίας : 2,4V – 3.6V
- Συχνότητα λειτουργίας : max 48 MHz
- Πυρήνας : Arm® 32-bit Cortex®-M0
- Μνήμη Flash : 64 Kbytes
- Μνήμη SRAM : 8 Kbytes
- Timers : 5 General, 1 Advance, 1 Basic
- SPI : 2

- I2C : 2
- USART : 2
- GPIO : 39
- Κόστος : 2,93€
- Serial wire debug (SWD) & JTAG interfaces

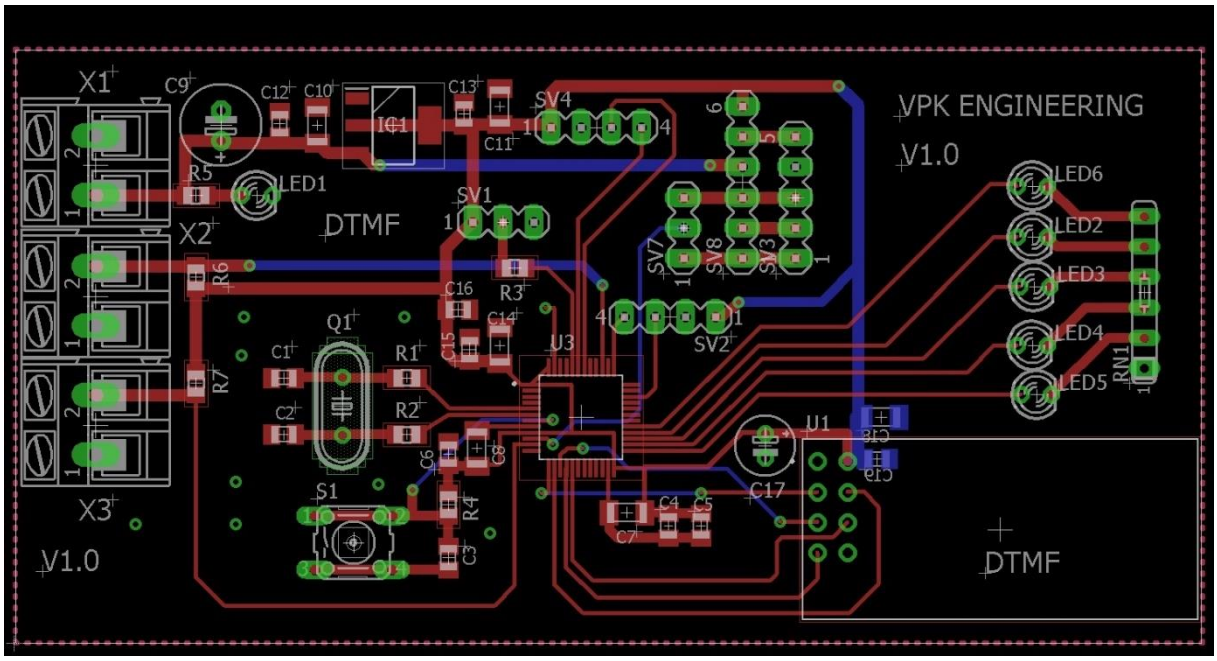
Το σχηματικό διάγραμμα του κυκλώματος του τοπικού ελεγκτή φαίνεται στο σχήμα 3.8.



Σχήμα 3.8 : Σχηματικό διάγραμμα τοπικού ελεγκτή

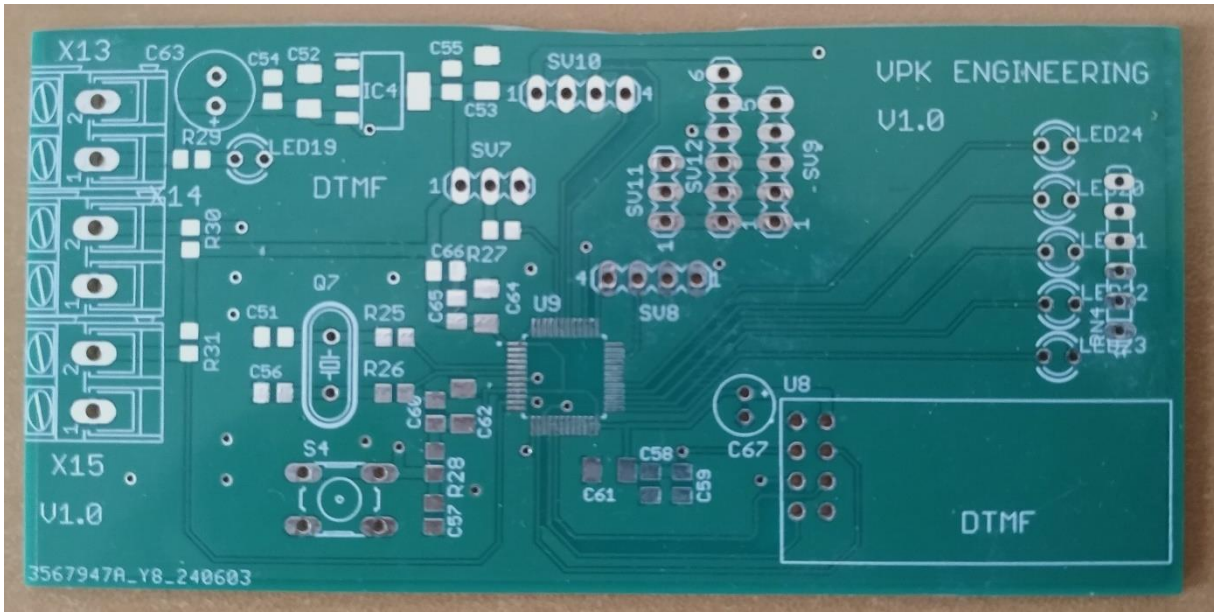
Ο μικροελεγκτής με τον κρύσταλλο και τα LED για τον έλεγχο λειτουργιών του κυκλώματος καθώς επίσης και το τροφοδοτικό φαίνονται με την πρώτη ματιά. Το ίδιο ισχύει και για τον πομποδέκτη NRF24L01+. Ο ακροδέκτης SV2 είναι για τον προγραμματισμό του μικροελεγκτή μέσω του αποσφαλματωτή/προγραμματιστή (debugger/programmer) ST-LINK/V2. Ο ακροδέκτης SV4 είναι για την σύνδεση του αισθητήρα θερμοκρασίας και υγρασίας, ενώ ο SV3 είναι για την σύνδεση του αισθητήρα κίνησης. Με τον ακροδέκτη SV8 μέσω βραχυκυκλωτήρα επιλέγουμε την τάση τροφοδοσίας του αισθητήρα κίνησης. Βραχυκυκλώνοντας τους ακροδέκτες 4 και 5 τροφοδοτούμε με τάση 5V ενώ βραχυκυκλώνοντας τους ακροδέκτες 5 και 6 τροφοδοτούμε με τάση 3,3V. Ο ακροδέκτης SV7 μας δίνει την έξοδο του αισθητήρα κίνησης. Βραχυκυκλώνοντας τους ακροδέκτες 1 και 2 έχουμε έξοδο από το ποδαράκι 1 του αισθητήρα ενώ βραχυκυκλώνοντας τους ακροδέκτες 2 και 3 έχουμε έξοδο από το ποδαράκι 3 του αισθητήρα. Αυτό έγινε για να μπορούν να τοποθετηθούν στην ίδια βάση και οι δύο αισθητήρες κίνησης απλός αλλάζοντας την θέση των βραχυκυκλωτήρων.

Στην εικόνα 3.19 έχουμε την πλακέτα του κυκλώματος όπως αυτή φαίνεται μαζί με τα εξαρτήματα στο σχεδιαστικό πρόγραμμα.

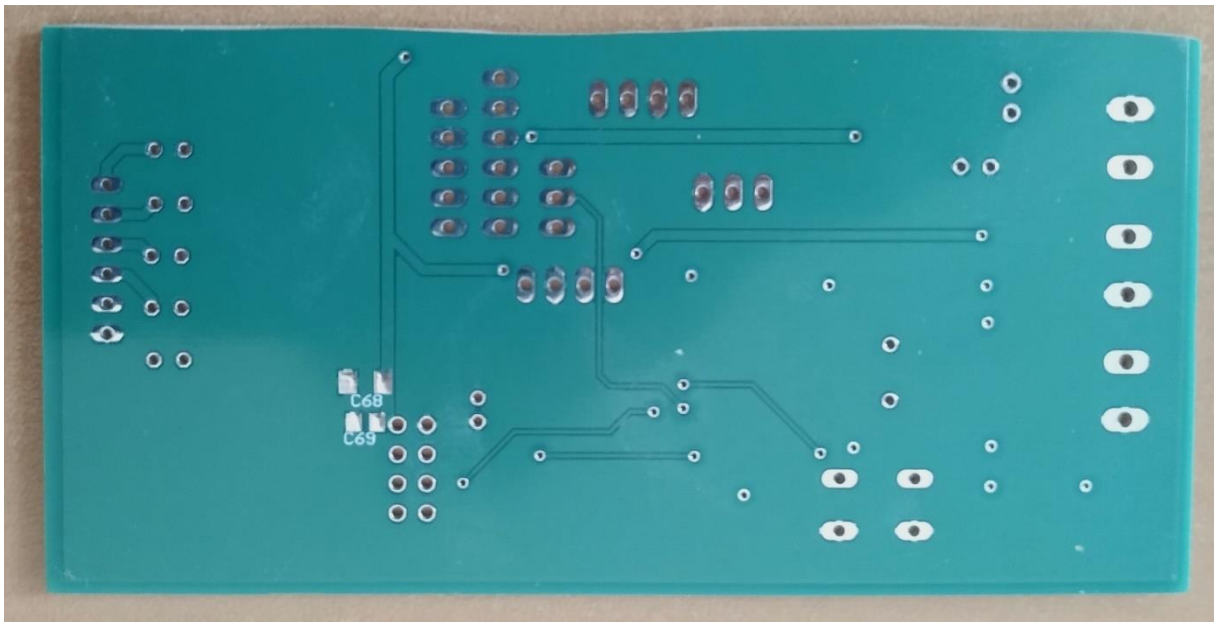


Εικόνα 3.19 : Σχεδίαση πλακέτας τοπικού ελεγκτή

Η πλακέτα πριν την συναρμολόγηση φαίνεται στις εικόνες 3.20 και 3.21.

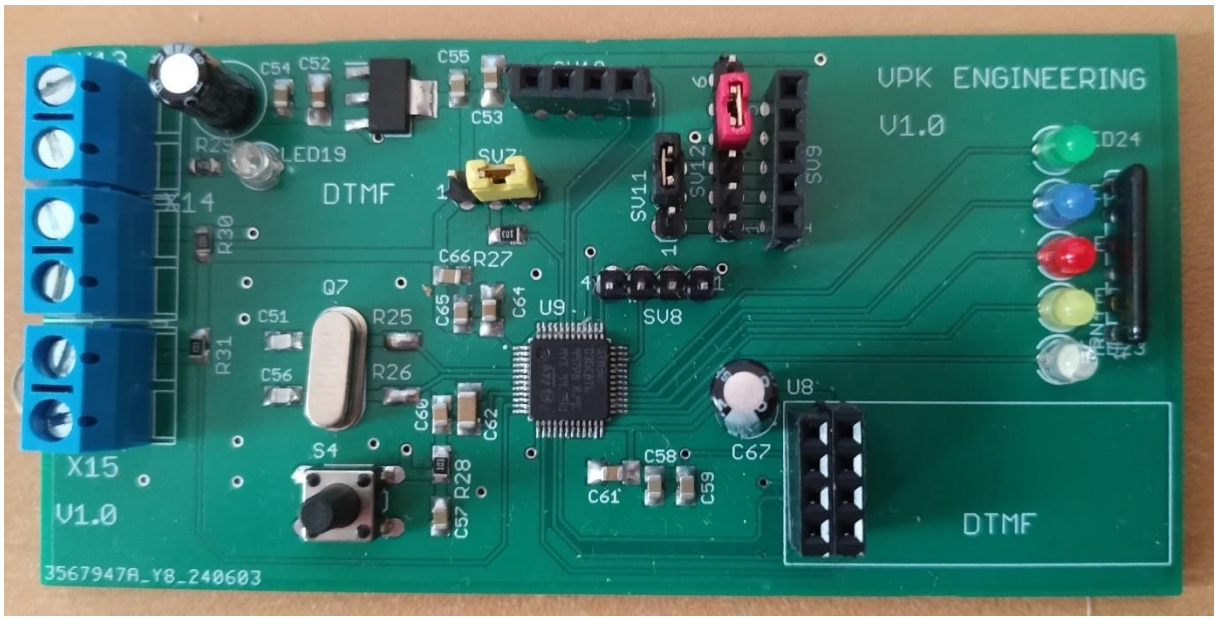


Εικόνα 3.20 : Πάνω όψη της πλακέτας



Εικόνα 3.21 : Κάτω όψη της πλακέτας

Στην εικόνα 3.22 φαίνεται η πλακέτα μετά την συναρμολόγηση



Εικόνα 3.22 : Πλακέτα μετά την συναρμολόγηση

3.4 Ανάλυση Τοπικών Δεκτών

Όπως αναφέρθηκε και στο δεύτερο κεφάλαιο υπάρχουν τρείς διαφορετικοί τοπικοί δέκτες οι οποίοι είναι και στην ουσία οι διακόπτες για την ενεργοποίηση ή την απενεργοποίηση των συσκευών στις οποίες είναι συνδεμένοι.

Ο διακόπτης με χρονοκαθυστέρηση απενεργοποίησης και ο διακόπτης αφυγραντήρα στο υλικό μέρος της κατασκευής είναι ακριβώς ίδιοι. Διαφέρουν μόνο στο πρόγραμμα του μικροελεγκτή. Για αυτό και η περιγραφή του δέκτη διακόπτη αναφέρεται και στους δύο αυτούς διακόπτες.

Ο δέκτης διακόπτης αποτελείται από:

- Τροφοδοτικό 3,3V 1A
- Πομποδέκτης NRF24L01+
- Μικροελεγκτή STM32F030C8T6
- Ηλεκτρονόμος 10A/250VAC

Ο δέκτης Dimmer αποτελείται από:

- Τροφοδοτικό 3,3V 1A
- Πομποδέκτης NRF24L01+
- Μικροελεγκτή STM32F030C8T6
- Κύκλωμα dimmer

Το τροφοδοτικό και ο μικροελεγκτής έχουν περιγραφή στην προηγούμενη ενότητα και δεν χρειάζονται περεταίρω ανάλυση. Ο πομποδέκτης όμως ο οποίος χρησιμοποιείται σε αυτούς τους διακόπτες είναι αυτός της Εικόνας 3.5 δηλαδή χωρίς τον ενισχυτή διότι η μέγιστη απόσταση επικοινωνίας με τον τοπικό ελεγκτή είναι συνήθως μικρότερη των τριών μέτρων. Αυτό έχει σαν αποτέλεσμα και μικρότερο όγκο της κατασκευής.

3.4.1 Ηλεκτρονόμος 10A/250VAC

Στον δέκτη διακόπτη χρησιμοποιήθηκε στην έξοδο ένας ηλεκτρονόμος για να ενεργοποιεί ή απενεργοποιεί οποιαδήποτε συσκευή ανεξάρτητα από την τροφοδοσία της. Μπορεί να τροφοδοτείται είτε από το δίκτυο (τα 230VAC) είτε από κάποια μπαταρία.

Ο ηλεκτρονόμος είναι ο AZ9371T-1A-5D [12] (Εικόνα 3.23) της εταιρίας ZETTLER electronics και τα κυριότερα χαρακτηριστικά του είναι:

- Τάση πηνίου : 5V DC
- Μέγιστο ρεύμα επαφών : 10A/277V AC
- Αντίσταση επαφών : 100mΩ
- Αντίσταση πηνίου : 125Ω
- Διαστάσεις : 20,5X7,2X15,3mm
- Κόστος : 0,99€

Ο λόγος που επιλέχθηκε ο συγκεκριμένος ηλεκτρονόμος ήταν ο συνδυασμός μικρού μεγέθους, αντοχή σε υψηλό ρεύμα (για οικιακές συσκευές), τάση πηνίου και χαμηλή τιμή.

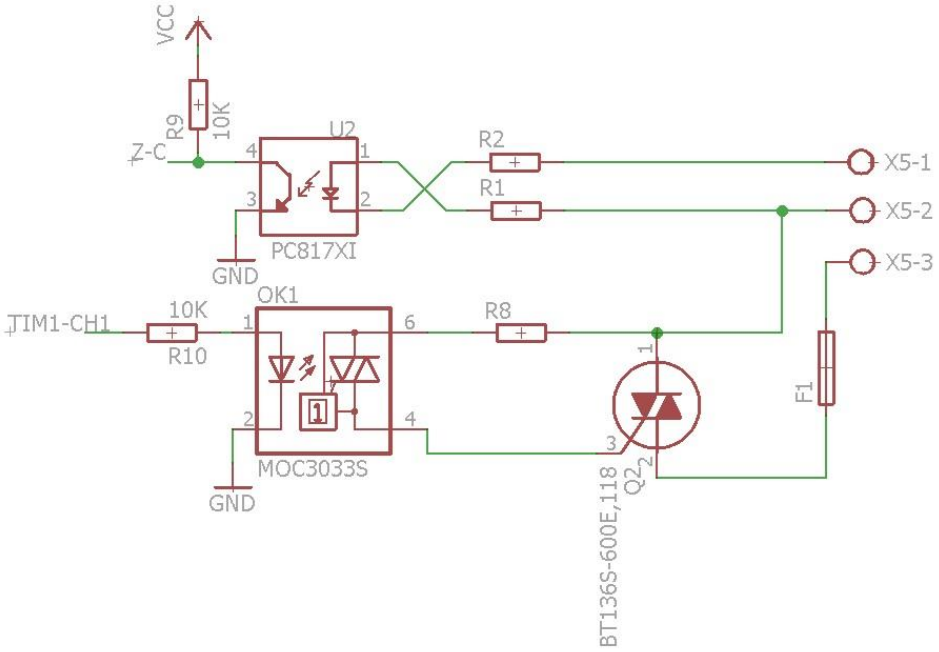
Εικόνα 3.23 : Ηλεκτρονόμος AZ9371T-1A-5D

3.4.2 Κύκλωμα Dimmer

Η διαφορά του δέκτη διακόπτη από τον δέκτη dimmer είναι στην έξοδό τους. Ενώ ο δέκτης διακόπτης έχει έναν ηλεκτρονόμο για έξοδο, ο δέκτης dimmer στην έξοδό του έχει ένα κύκλωμα dimmer για να μπορεί να έχει και χαμηλωμένο φωτισμό όπου χρειάζεται.

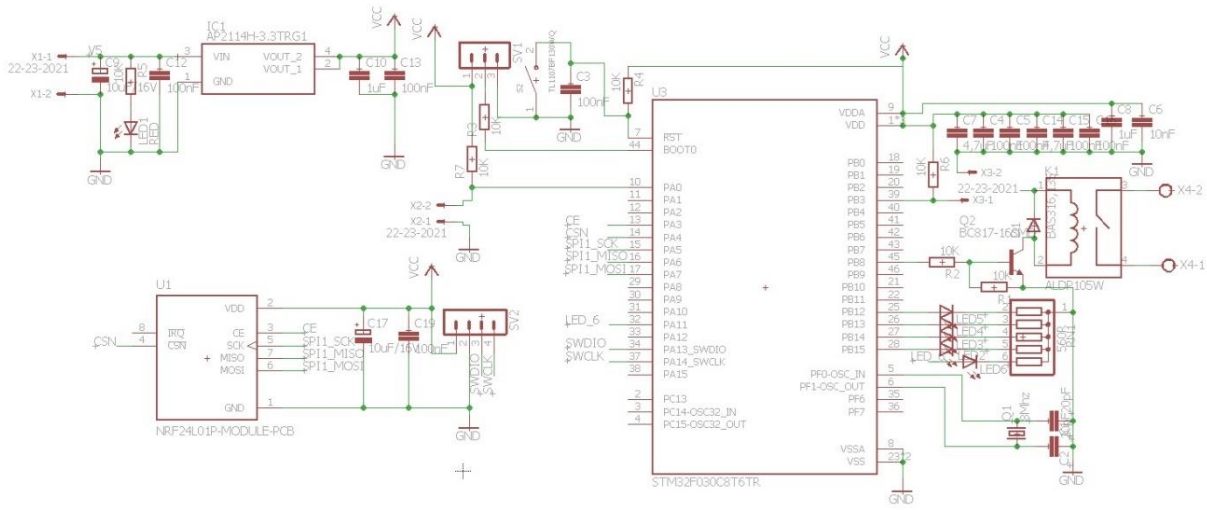
Όπως φαίνεται και από το σχηματικό διάγραμμα στο Σχήμα 3.9 το κύκλωμα αποτελείται από έναν ανιχνευτή zero crossing και από το κύκλωμα εξόδου με το Triac BT136. Αρχικά το zero cross κύκλωμα αποτελούταν από μια γέφυρα ανόρθωσης και το optocoupler PC817. Το optocoupler PC817 έχει εσωτερικά μόνο μια δίοδο LED οπότε για να ενεργοποιηθεί και να μας δώσει σήμα στην έξοδό του έπρεπε να χρησιμοποιηθεί μια γέφυρα ανόρθωσης για να πάρει στην είσοδό του το optocoupler PC817 και τις δύο ημιπεριόδους της εναλλασσόμενης τάσης. Για λόγους όμως εξοικονόμησης χώρου παραλήφθηκε η γέφυρα ανόρθωσης και το PC817 αντικαταστάθηκε από το PC814 το οποίο έχει το ίδιο footprint με το PC817 αλλά εσωτερικά έχει δύο διόδους LED και είναι συνδεμένες η μια αντίθετα από την άλλη, με αποτέλεσμα όταν έρχεται στην είσοδο του optocoupler PC814 είτε η θετική ημιπερίοδος είτε η αρνητική ημιπερίοδος ενεργοποιείται μια το ένα LED και μια το άλλο LED του optocoupler ενεργοποιώντας την έξοδο και στις δύο ημιπεριόδους της τάσης.

Το κύκλωμα εξόδου αποτελείται από το Triac BT136 και τον οδηγό του Triac το MOC3023S.



Σχήμα 3.9 : Σχηματικό διάγραμμα κυκλώματος Dimmer

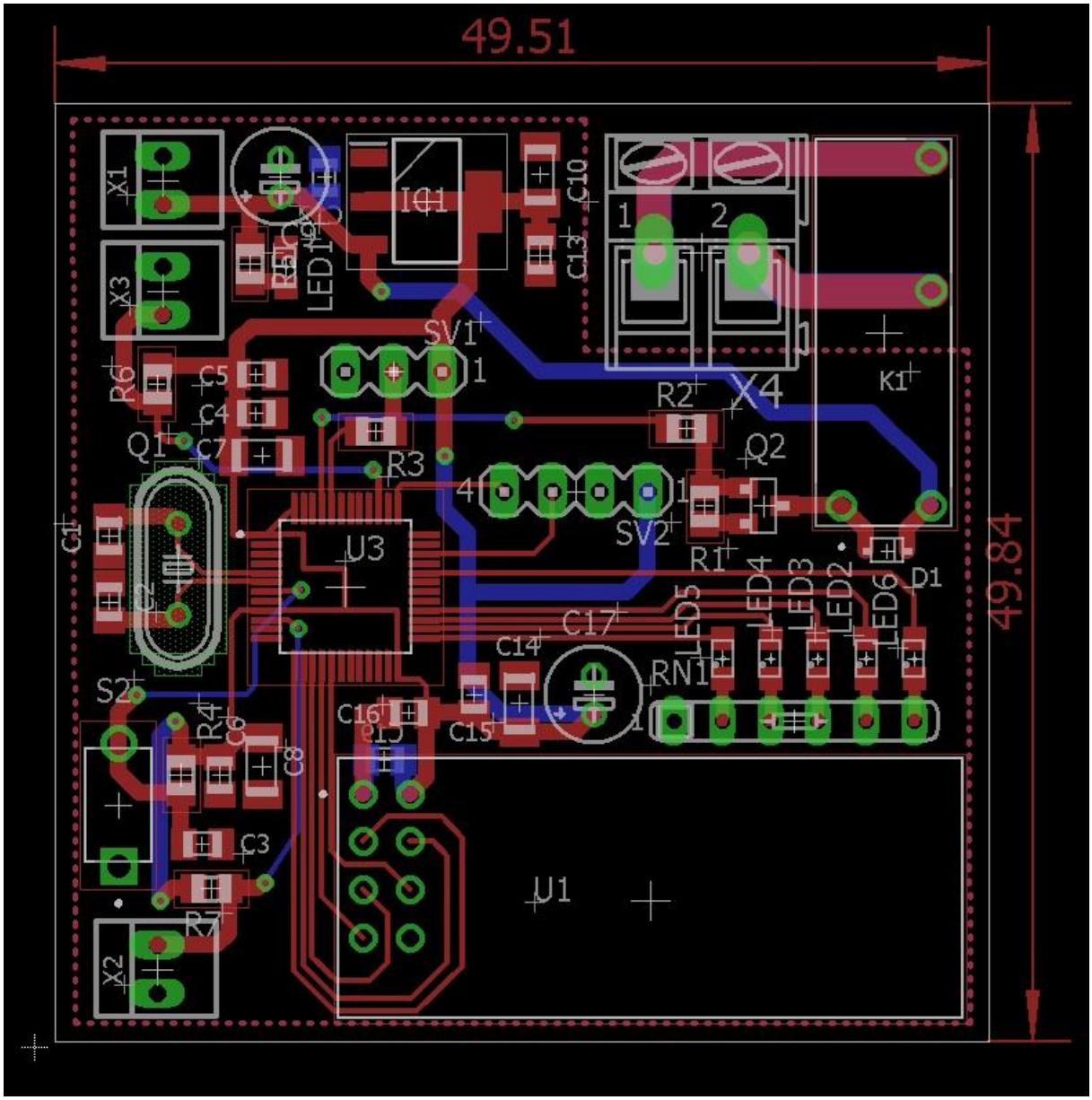
Στο σχήμα 3.10 φαίνεται το σχηματικό διάγραμμα του κυκλώματος του τοπικού δέκτη διακόπτη.



Σχήμα 3.10 : Σχηματικό διάγραμμα κυκλώματος τοπικού δέκτη διακόπτη

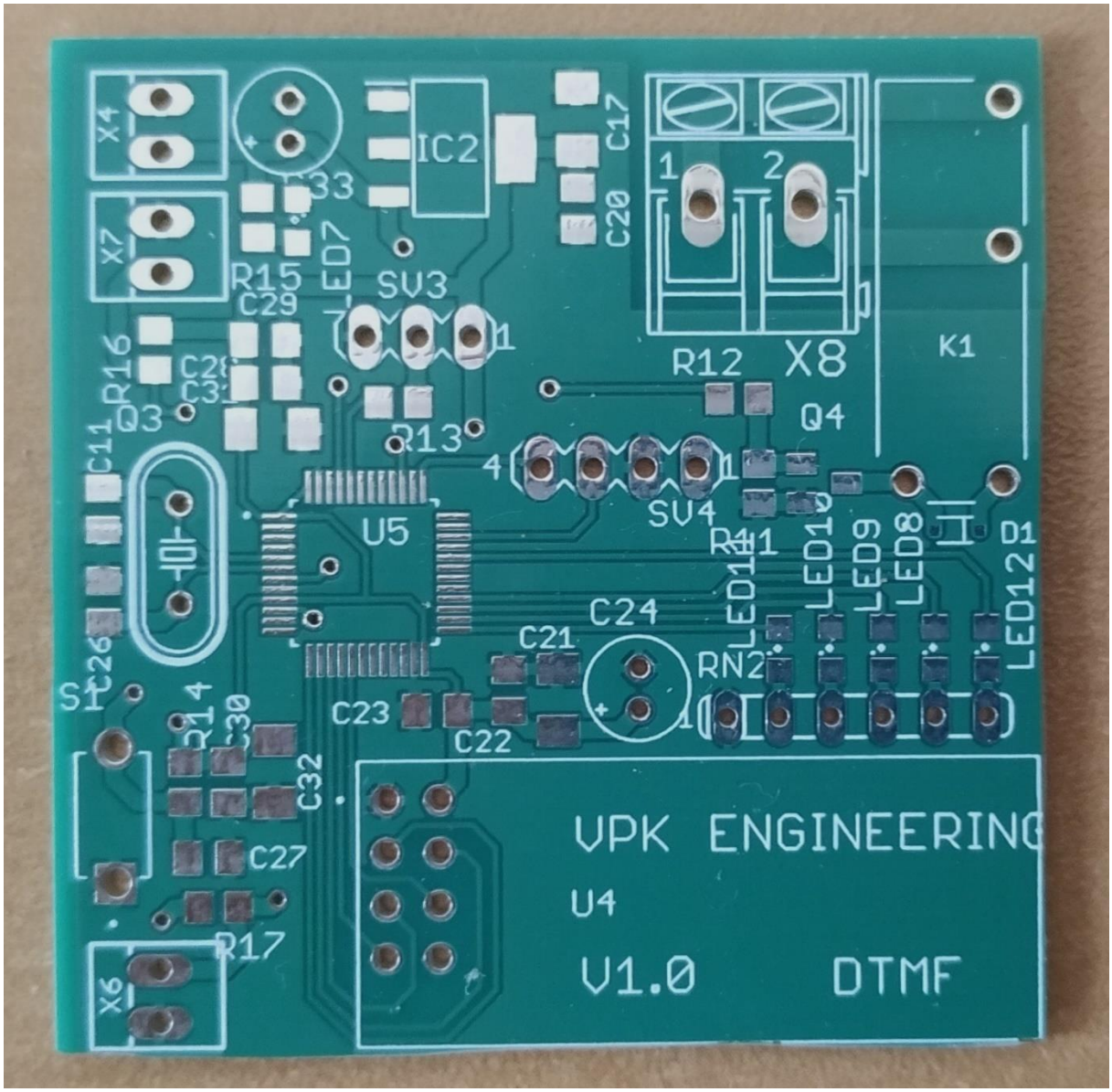
Επάνω αριστερά φαίνεται το τροφοδοτικό του κυκλώματος στα 3,3V και κάτω αριστερά είναι ο πομποδέκτης NRF24L01+. Δεξιά λίγο πιο πάνω από την μέση είναι ο ηλεκτρονόμος εξόδου.

Στην Εικόνα 3.24 φαίνεται η πλακέτα του τοπικού δέκτη διακόπτη μέσα από το σχεδιαστικό πρόγραμμα μαζί με τα εξαρτήματα.

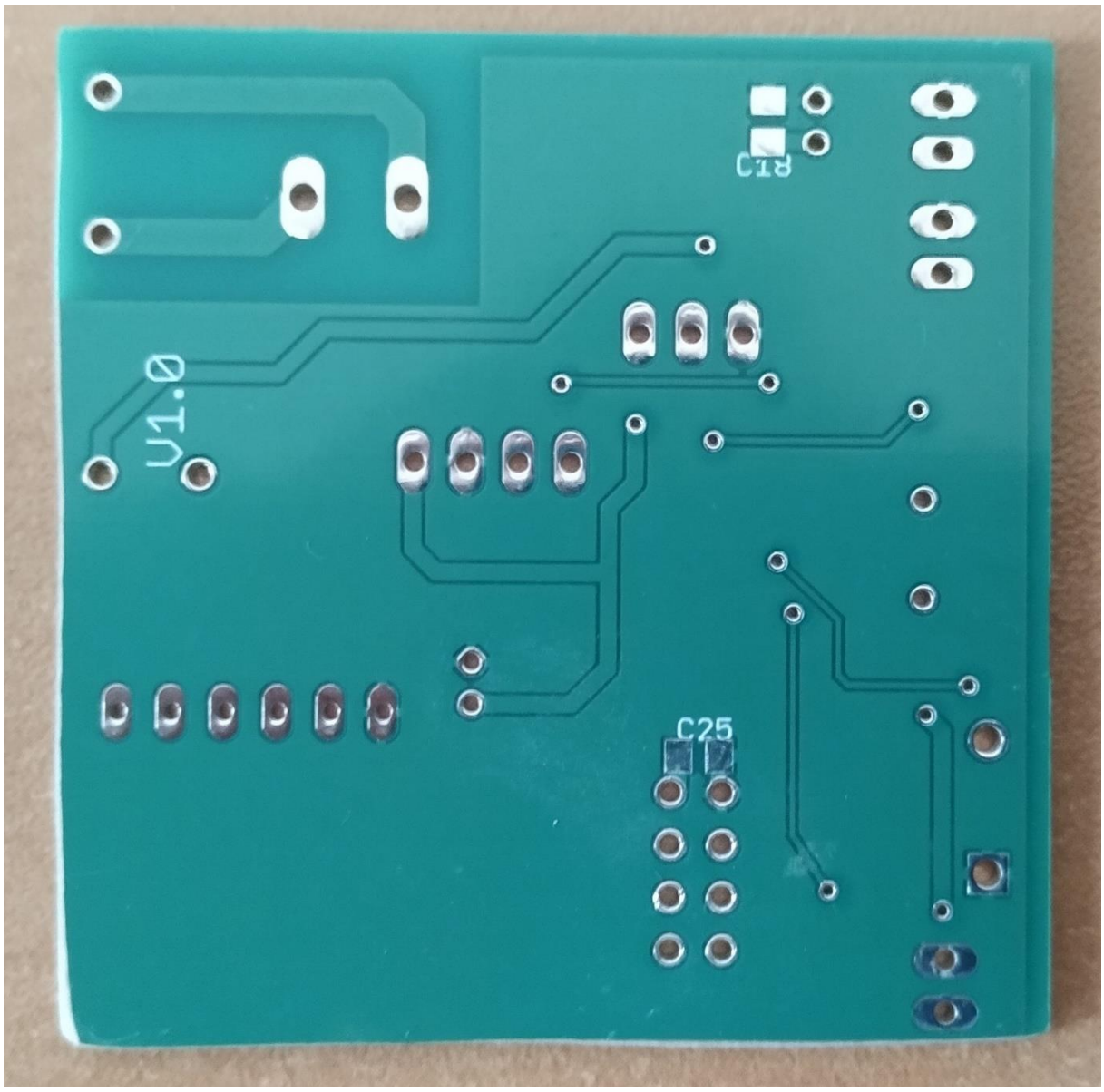


Εικόνα 3.24 : Πλακέτα τοπικού δέκτη διακόπτη

Στην Εικόνα 3.25 έχουμε την πάνω όψη της πλακέτας πριν την συναρμολόγηση και στην Εικόνα 3.26 έχουμε την κάτω όψη της πλακέτας.

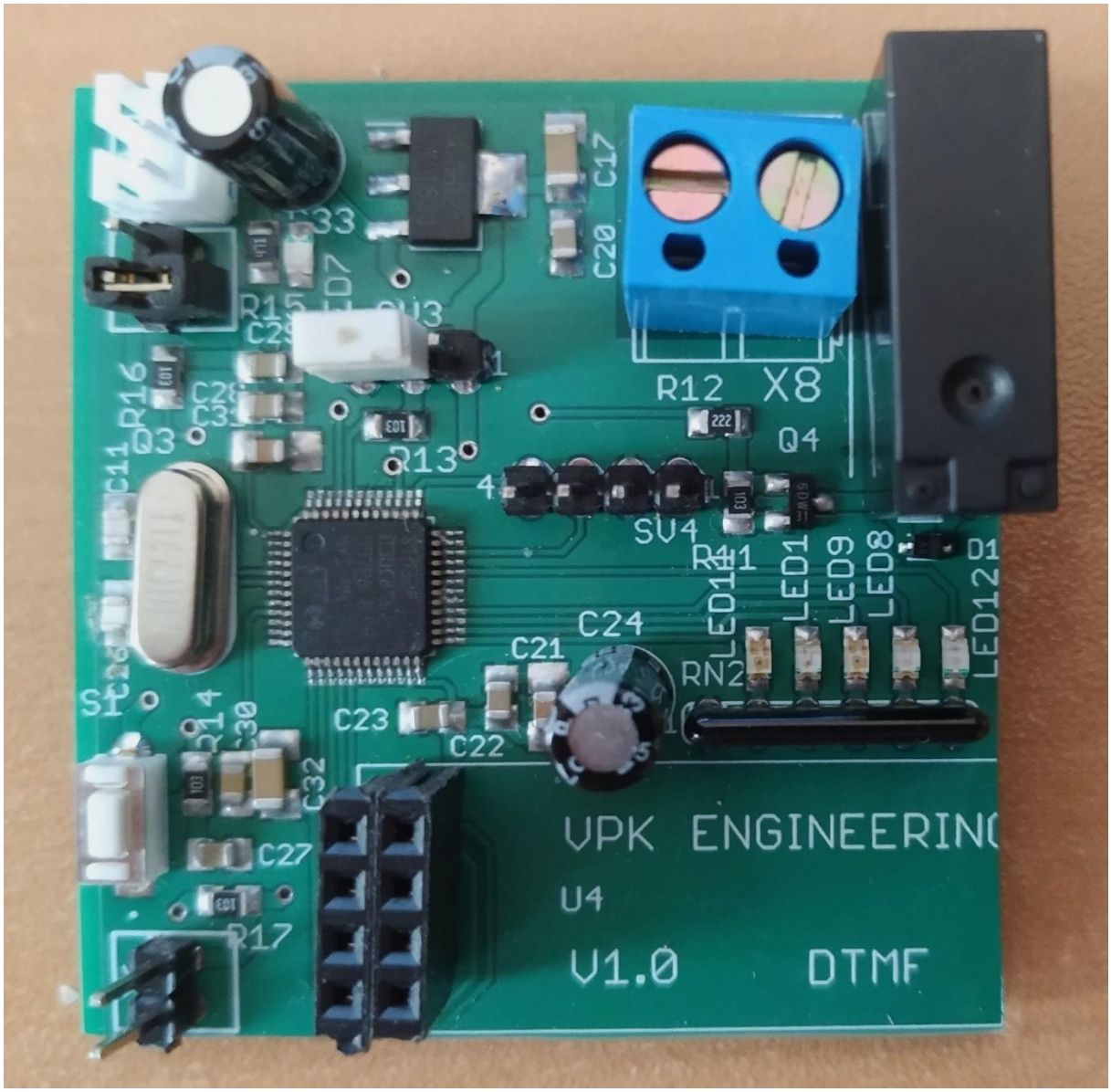


Εικόνα 3.25 : Πάνω όψη πλακέτας τοπικού δέκτη διακόπτη



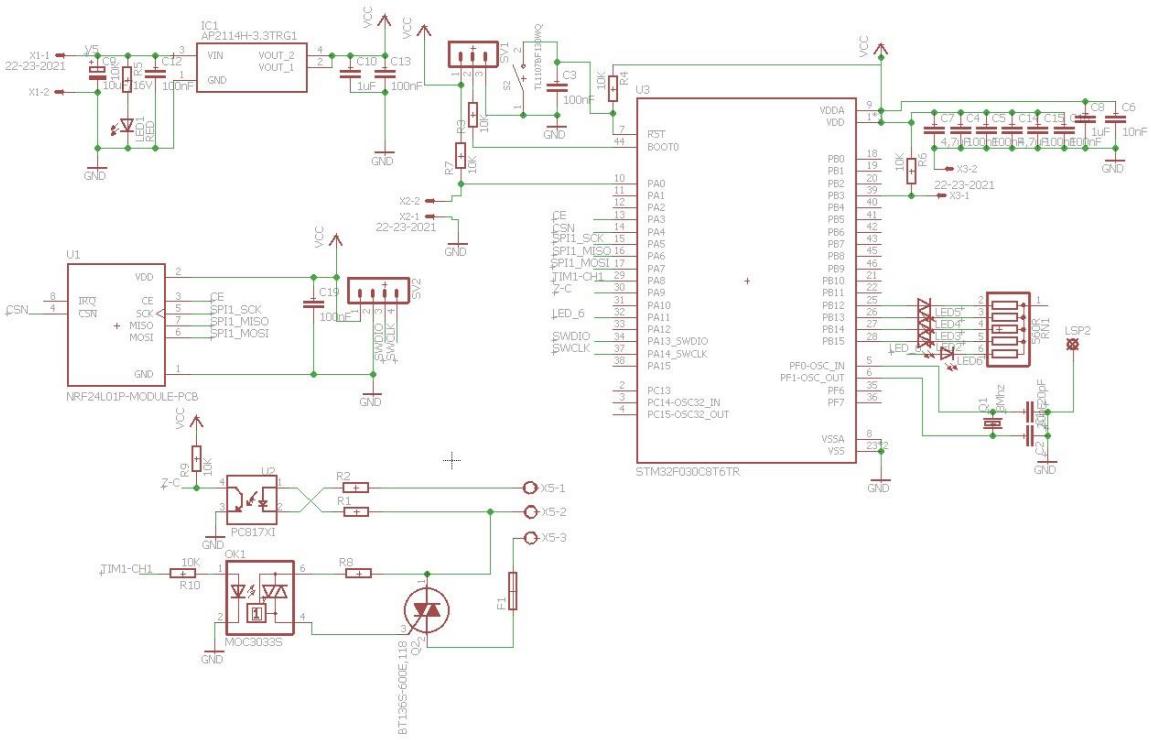
Εικόνα 3.26 : Κάτω όψη πλακέτας τοπικού δέκτη διακόπτη

Στην Εικόνα 3.27 έχουμε την πλακέτα μετά την συναρμολόγηση των υλικών.



Εικόνα 3.27 : Πλακέτα τοπικού δέκτη διακόπτη συναρμολογημένη

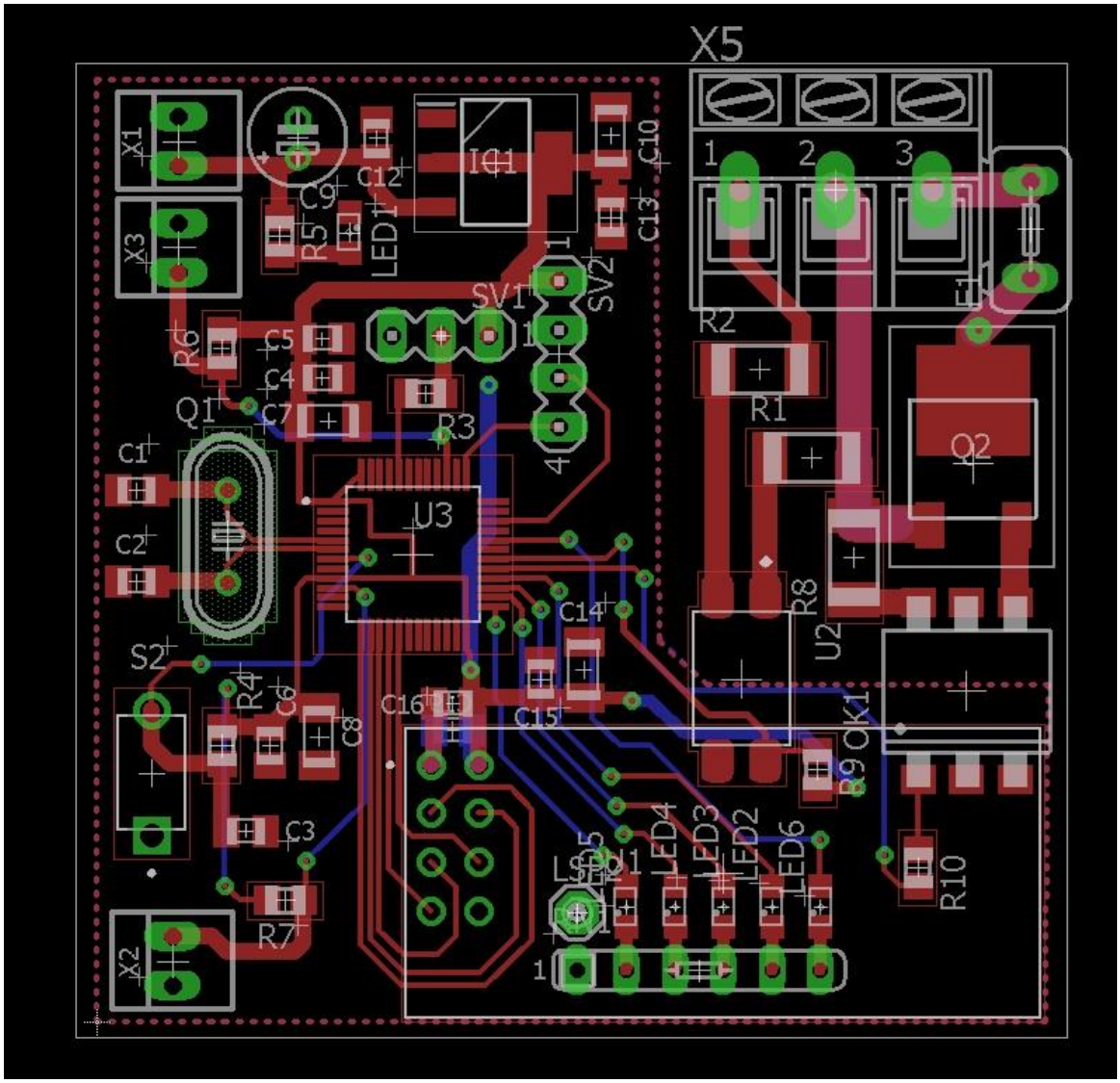
Το σχηματικό διάγραμμα του κυκλώματος του τοπικού δέκτη Dimmer φαίνεται στο σχήμα 3.11.



Σχήμα 3.11 : Σχηματικό διάγραμμα κυκλώματος τοπικού δέκτη Dimmer

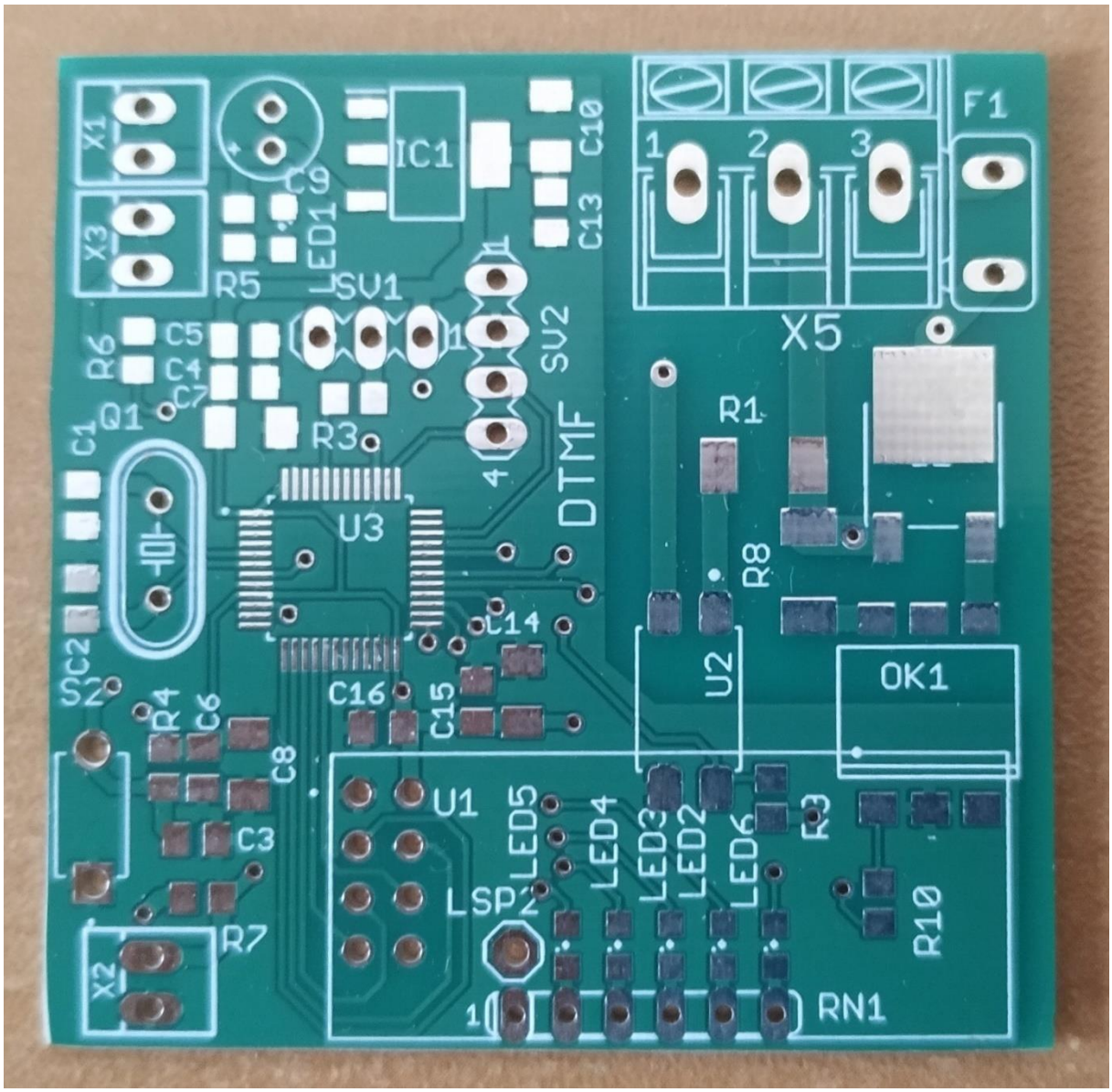
Με μια σύγκριση με το σχηματικό διάγραμμα του τοπικού δέκτη διακόπτη παρατηρούμε ότι απουσιάζει ο ηλεκτρονόμος και στο κάτω αριστερό μέρος του διαγράμματος υπάρχει το κύκλωμα dimmer που αναφέρθηκε λίγο πιο πάνω.

Η πλακέτα του τοπικού δέκτη Dimmer μέσα από το σχεδιαστικό πρόγραμμα μαζί με τα εξαρτήματα φαίνεται στην Εικόνα 3.28.

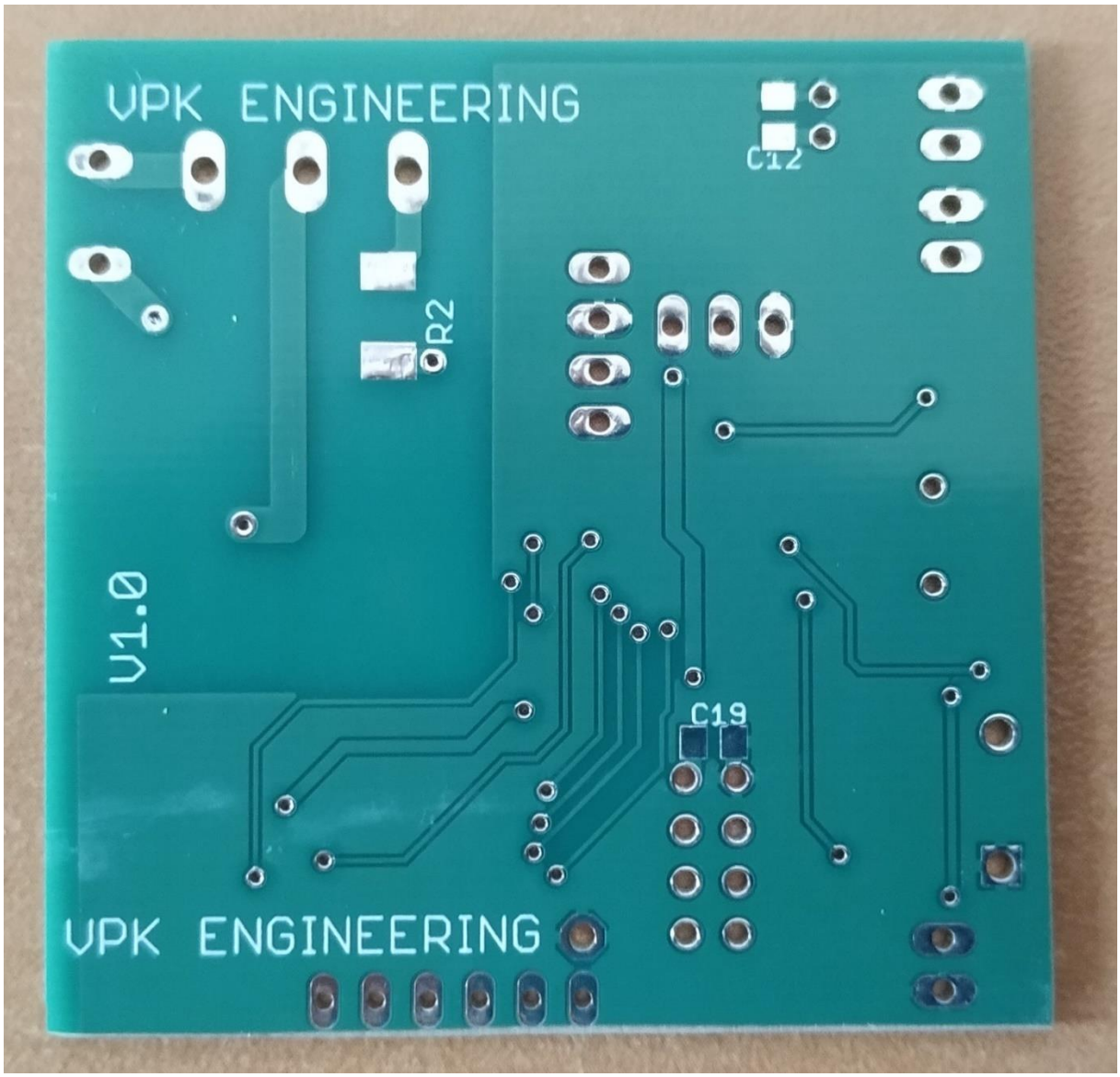


Εικόνα 3.28 : Πλακέτα τοπικού δέκτη Dimmer

Στην Εικόνα 3.29 έχουμε την πάνω όψη της πλακέτας πριν την συναρμολόγηση και στην Εικόνα 3.30 έχουμε την κάτω όψη της πλακέτας.

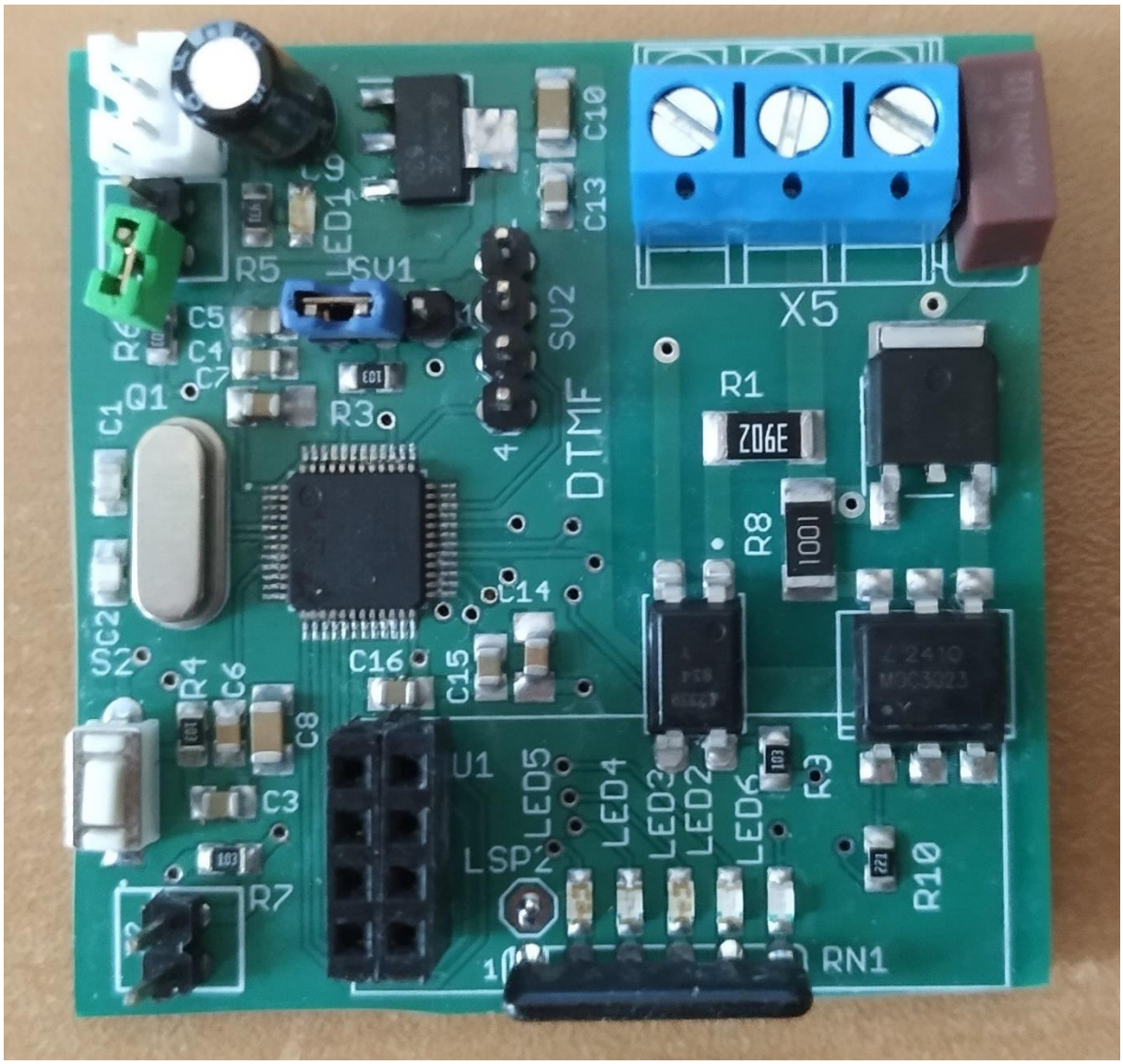


Εικόνα 3.29 : Πάνω όψη πλακέτας τοπικού δέκτη Dimmer



Εικόνα 3.30 : Κάτω όψη πλακέτας τοπικού δέκτη Dimmer

Η πλακέτα μετά την συναρμολόγηση των υλικών φαίνεται στην Εικόνα 3.31.



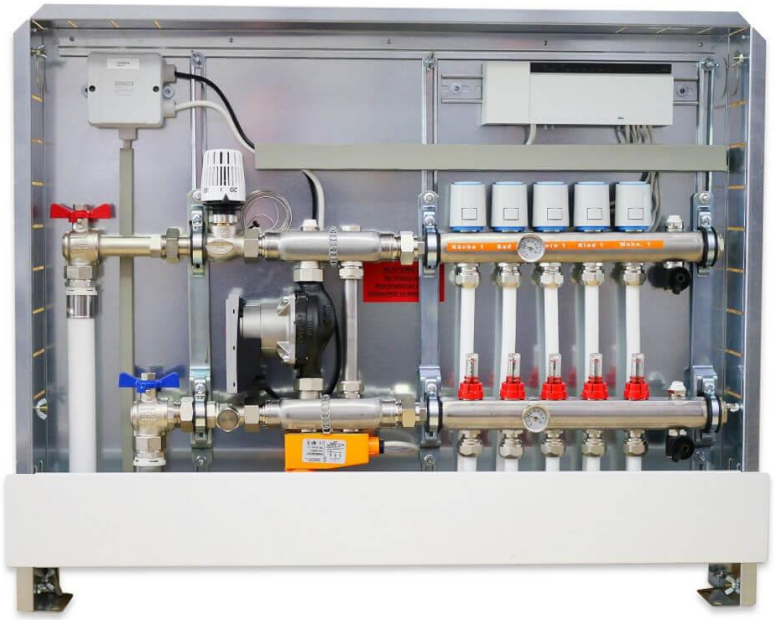
Εικόνα 3.31 : Πλακέτα τοπικού δέκτη Dimmer συναρμολογημένη

3.5 Ανάλυση τελικής βαθμίδας ελέγχου ηλεκτροθερμικών κεφαλών

Το τελευταίο μέρος του συστήματος αποτελείται από την βαθμίδα ελέγχου των ηλεκτροθερμικών κεφαλών. Στην πραγματικότητα είναι η βαθμίδα στην οποία βρίσκονται οι ηλεκτρονόμοι που τροφοδοτούν τις ηλεκτροθερμικές κεφαλές (Εικόνα 3.32) οι οποίες με την σειρά τους ανοίγουν ή κλείνουν βάνες για να περάσει το ζεστό νερό στα σώματα θέρμανσης του σπιτιού. Η βαθμίδα αυτή ελέγχεται από τον κεντρικό ελεγκτή ανάλογα με την ζήτηση ή όχι θέρμανσης του κάθε χώρου.

Εικόνα 3.32 : Ηλεκτροθερμική κεφαλή

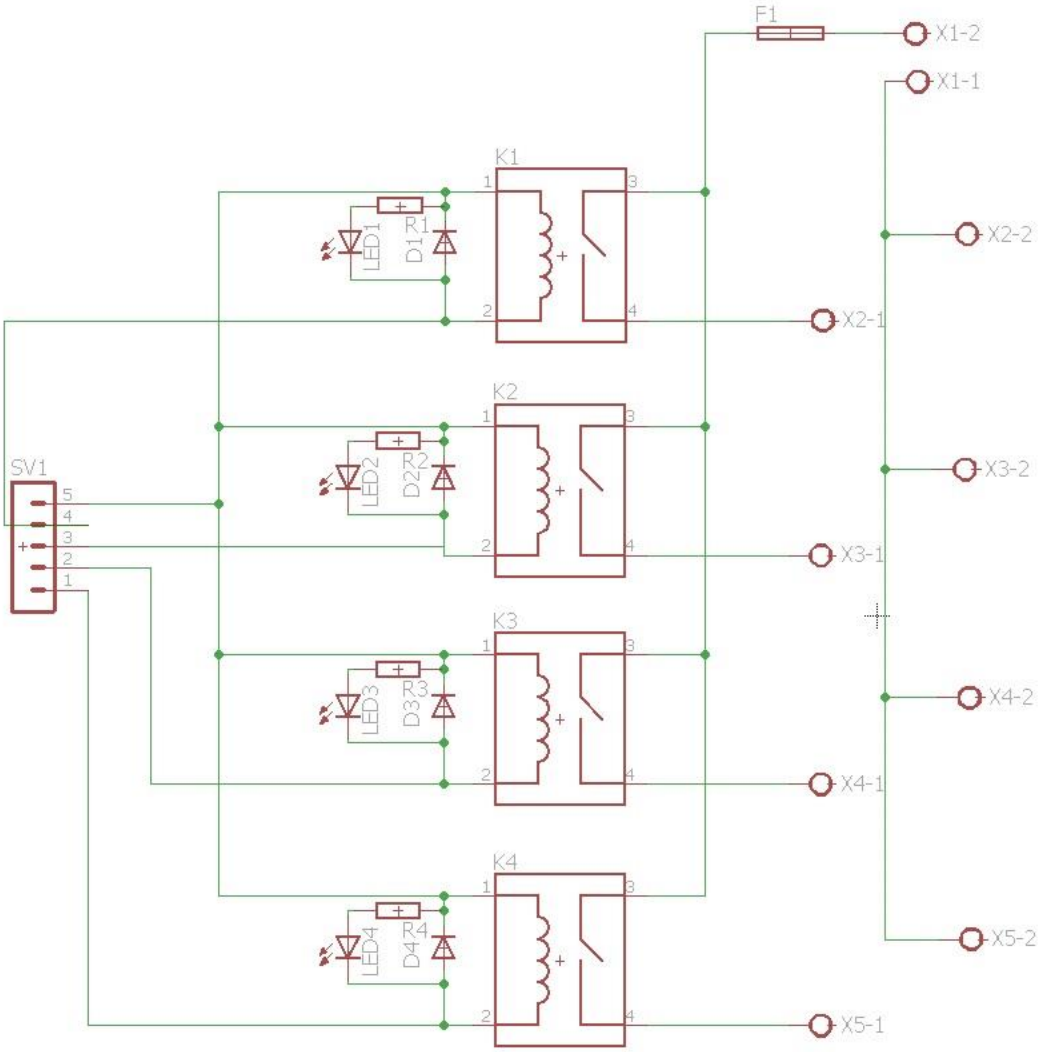
Οι ηλεκτροθερμικές κεφαλές είναι συνδεμένες η κάθε μια σε μια βάνα στον κεντρικό συλλέκτη (Εικόνα 3.33) διανομής ζεστού νερού παρέχοντας ή διακόπτοντας την ροή ζεστού νερού στο αντίστοιχο σώμα θέρμανσης που είναι συνδεμένες.



Εικόνα 3.33 : Σταθμός διανομής νερού θέρμανσης

Στην Εικόνα 3.30 φαίνονται 5 ηλεκτροθερμικές κεφαλές συνδεμένες στον συλλέκτη ζεστού νερού και ακριβώς από πάνω το κουτί με τα κυκλώματα που ελέγχουν τις ηλεκτροθερμικές κεφαλές.

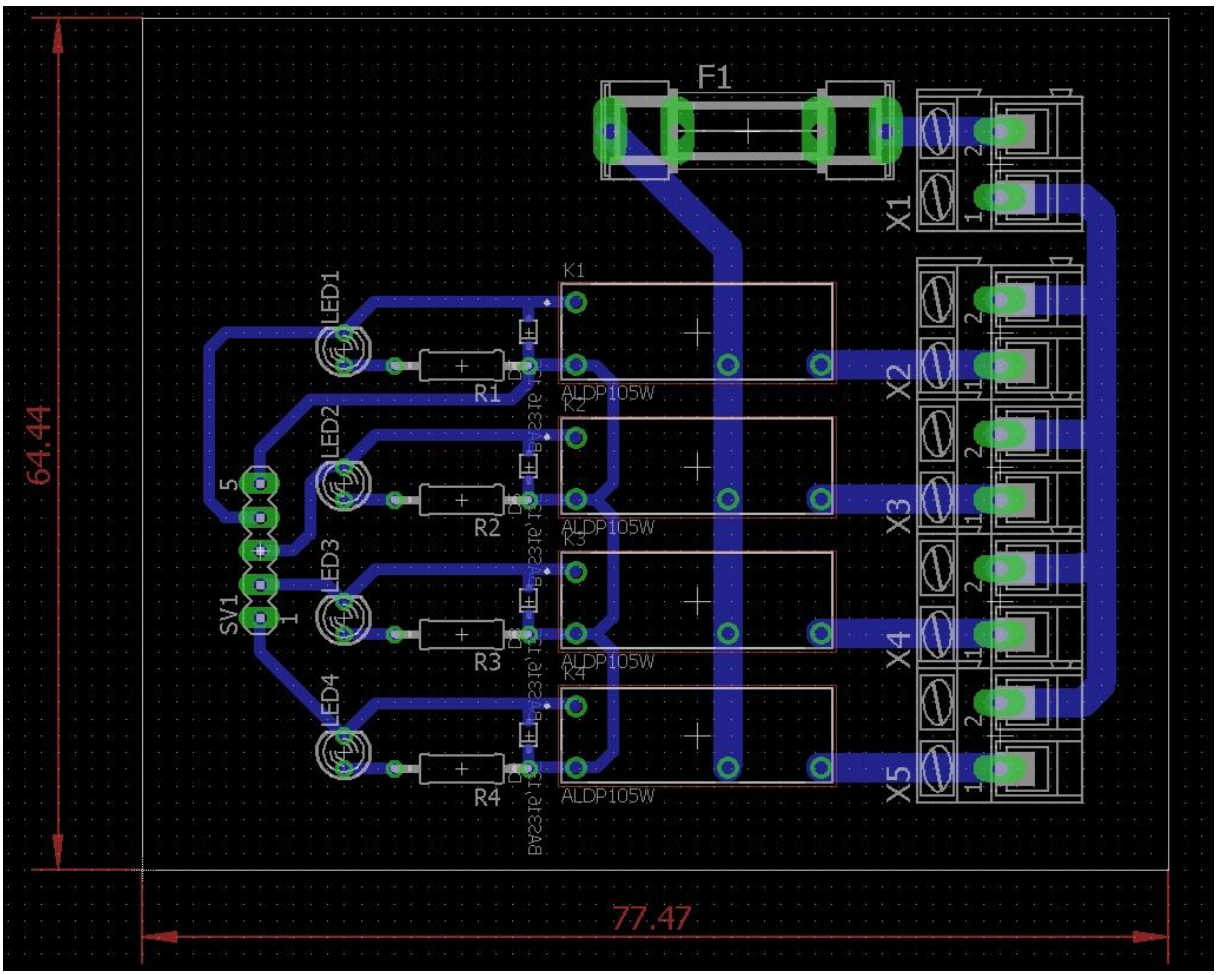
Το σχηματικό διάγραμμα της τελικής βαθμίδας ελέγχου των ηλεκτροθερμικών κεφαλών φαίνεται στο σχήμα 3.12.



Σχήμα 3.12 : Σχηματικό διάγραμμα κυκλώματος τελικής βαθμίδας ελέγχου ηλεκτροθερμικών κεφαλών

Στον ακροδέκτη X1-1 και X1-2 έχουμε την τροφοδοσία των ηλεκτροθερμικών κεφαλών που είναι 230V AC. Υπάρχουν και ηλεκτροθερμικές κεφαλές που τροφοδοτούνται με 24V AC. Στους ακροδέκτες X2, X3, X4, X5 συνδέονται οι τροφοδοσίες των τεσσάρων ηλεκτροθερμικών κεφαλών που ελέγχονται από τους τέσσερις ηλεκτρονόμους K1 – K4. Στην τροφοδοσία κάθε ηλεκτρονόμου υπάρχει και από ένα ενδεικτικό λειτουργίας LED. Ο ακροδέκτης SV1 συνδέεται στον αντίστοιχο ακροδέκτη SV3 του κεντρικού ελεγκτή.

Η πλακέτα της τελικής βαθμίδας ελέγχου των ηλεκτροθερμικών κεφαλών μέσα από το σχεδιαστικό πρόγραμμα μαζί με τα εξαρτήματα φαίνεται στην Εικόνα 3.34.



Εικόνα 3.34 : Πλακέτα τελικής βαθμίδας ελέγχου ηλεκτροθερμικών κεφαλών

3.6 Επίλογος

Για όλα τα σημαντικά εξαρτήματα τα οποία χρησιμοποιήθηκαν σε αυτό το σύστημα έγινε αναφορά σε αυτό το κεφάλαιο. Παρουσιάστηκαν τα πιο σημαντικά χαρακτηριστικά των εξαρτημάτων αυτών καθώς και την λειτουργία αυτών. Επίσης παρουσιάστηκαν τα σχηματικά διαγράμματα όλων των κυκλωμάτων που κατασκευάστηκαν, οι πλακέτες αυτών και η συναρμολόγησή τους.

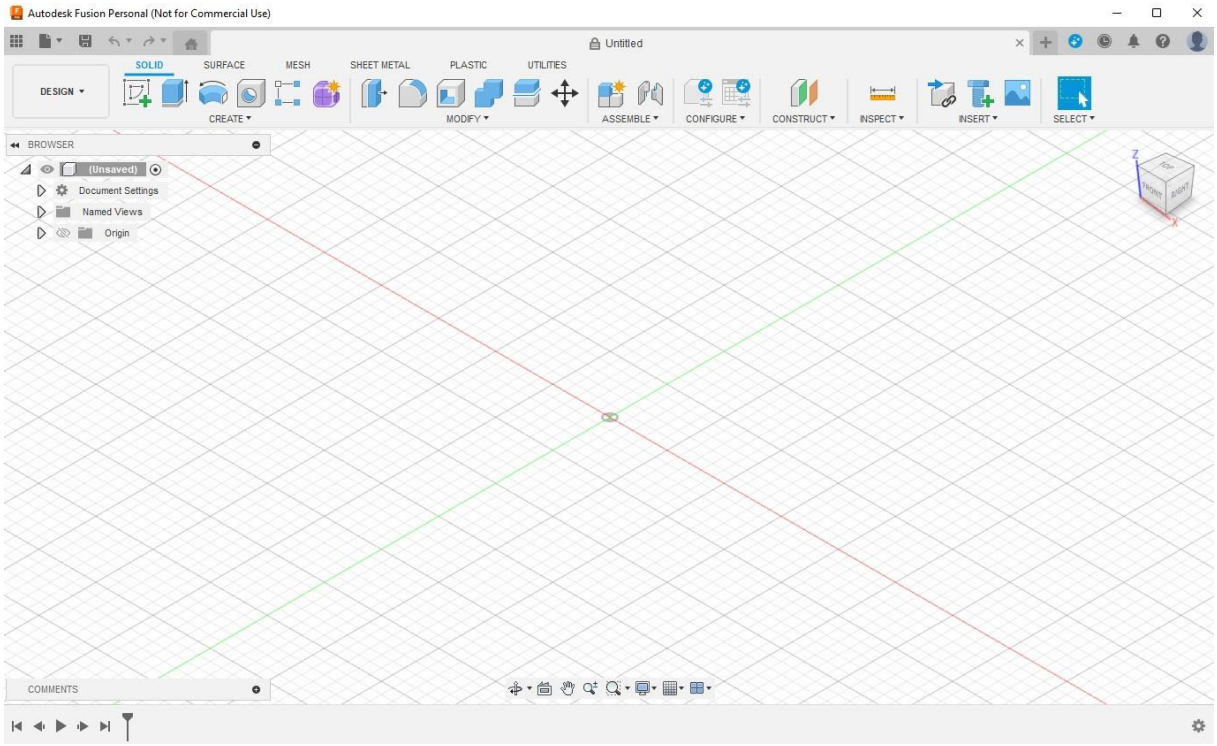
Κεφάλαιο 4ο: Σχεδιασμός και Εκτύπωση Κουτιών

4.1 Εισαγωγή

Μετά την συναρμολόγηση των πλακετών το επόμενο βήμα είναι η τοποθέτηση τους σε κουτιά έτσι ώστε τα εξαρτήματα αλλά και οι επαφές να μην είναι εκτεθειμένες ιδίως επειδή σε κάποιες πλακέτες υπάρχει και επικίνδυνη για τον άνθρωπο τάση.

4.2 Πρόγραμμα σχεδιασμού

Το πρόγραμμα που χρησιμοποιήθηκε για τον σχεδιασμό των κουτιών είναι η δωρεάν έκδοση FUSION 360 το οποίο πλέον μετονομάστηκε σε Autodesk Fusion [13]. Το Autodesk Fusion είναι ένα πρόγραμμα σχεδιασμού τρισδιάστατων μοντέλων τα οποία μέσω εντολής του προγράμματος μπορούν να εκτυπωθούν σε εκτυπωτή 3D. Το περιβάλλον του προγράμματος φαίνεται στην Εικόνα 4.1.

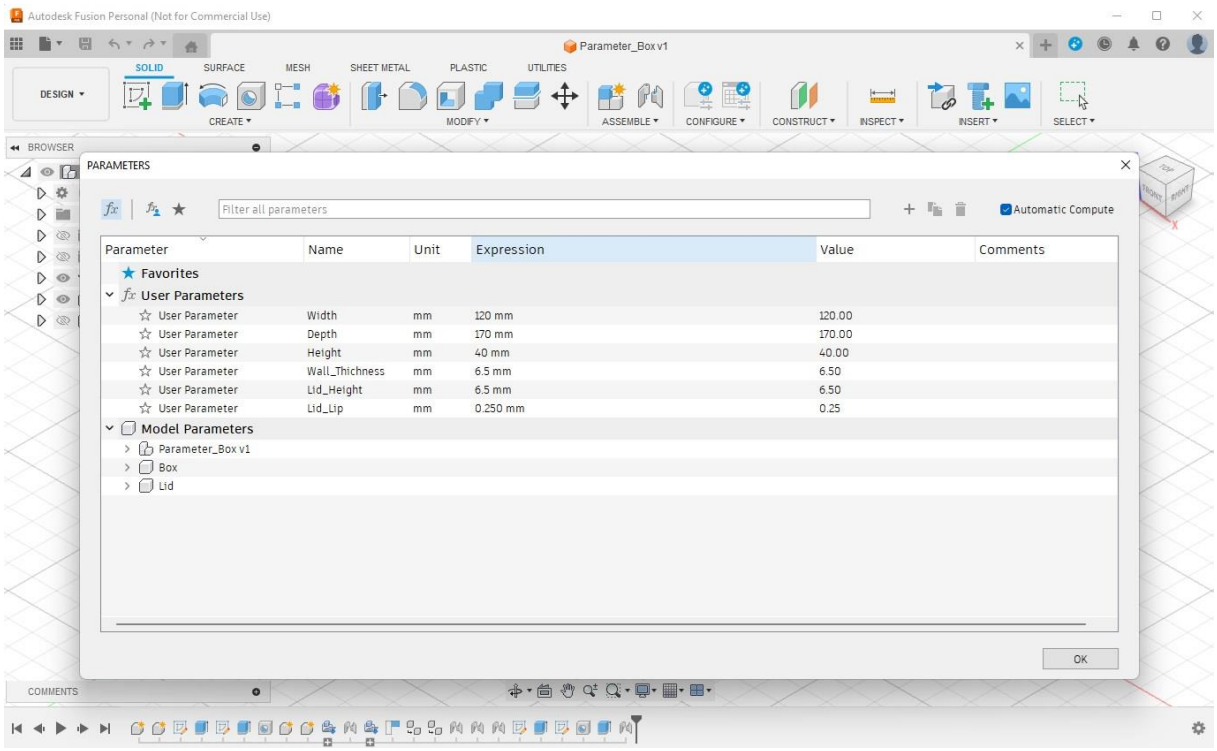


Εικόνα 4.1 : Περιβάλλον προγράμματος Autodesk Fusion

Να σημειωθεί ότι ένα πλεονέκτημα αυτού του προγράμματος είναι ότι κάθε τι που γίνεται καταγράφεται σε μια χρονική γραμμή (time line) στο κάτω μέρος του προγράμματος και έτσι δίνεται η δυνατότητα όποτε χρειαστεί σε όποιο σημείο της σχεδίασης να ανατρέξει ο σχεδιαστής σε εκείνο το χρονικό σημείο και ότι αλλαγές κάνει να ισχύουν από εκεί και πέρα.

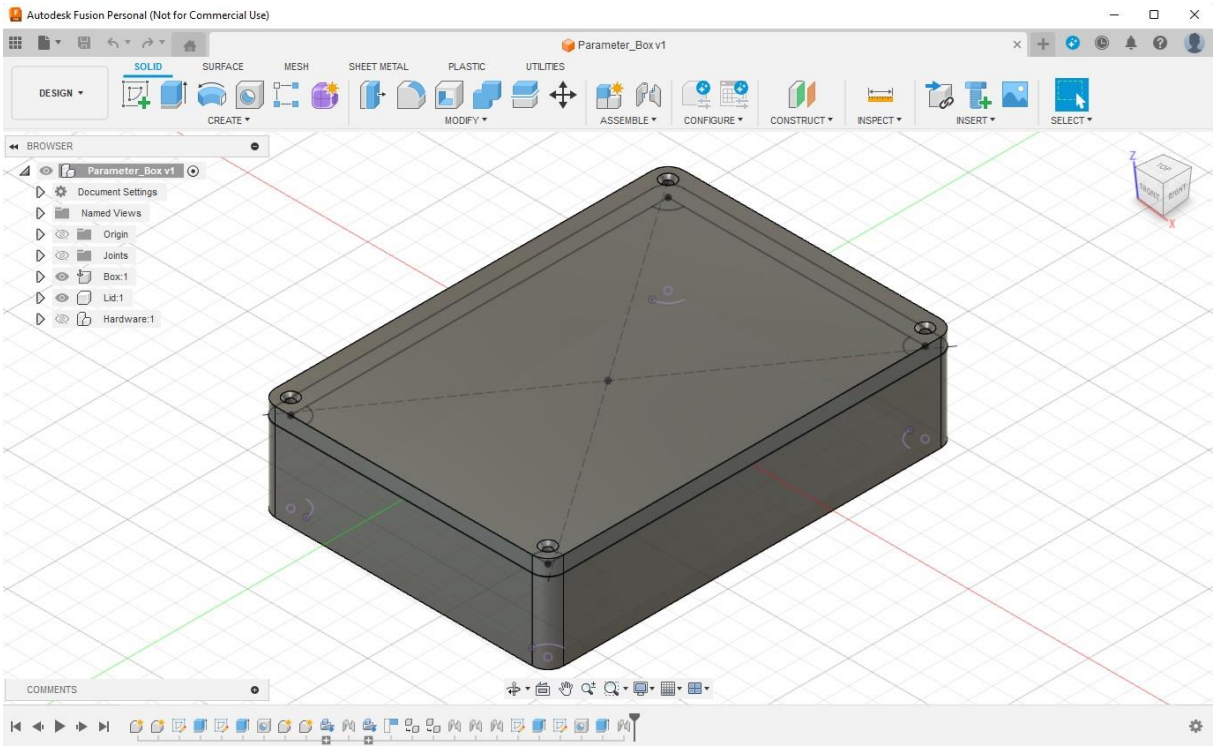
4.3 Σχεδίαση κουτιών

Η σχεδίαση ξεκίνησε με την καταγραφή κάποιων αρχικών παραμέτρων. Η ιδέα ήταν να δημιουργηθεί ένα αρχικό κουτί το οποίο να είναι παραμετροποιήσιμο (Εικόνα 4.2) και στην συνέχεια αλλάζοντας τις αρχικές παραμέτρους να σχεδιάζεται ένα άλλο κουτί με βάση τις ανάγκες που υπάρχουν. Οι παράμετροι είναι το ύψος, το πλάτος, το βάθος, το πάχος του τοιχώματος κ.α..



Εικόνα 4.2 : Παράμετροι κουτιού

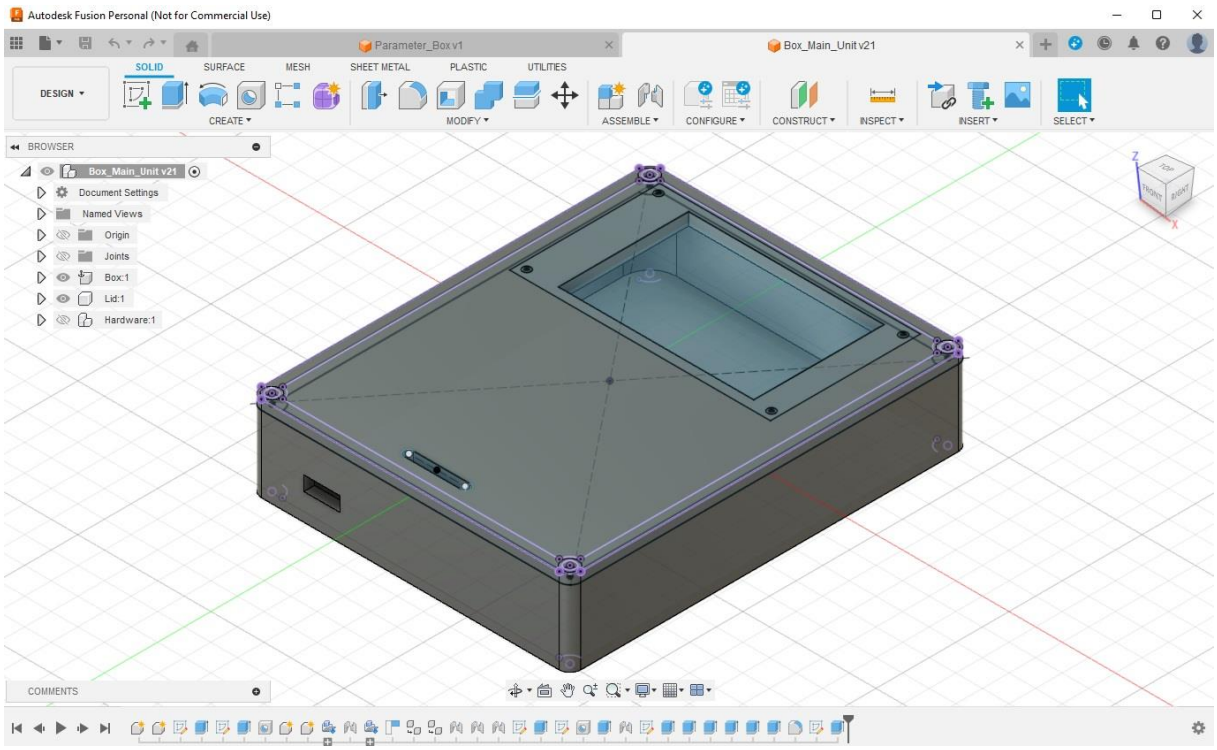
Το αποτέλεσμα είναι ένα απλό κουτί με θέσεις στις άκρες για χάλκινα χωνευτά παξιμάδια διάστασης M3. Όλα τα κουτιά σχεδιάστηκαν με βάση αυτό το πρότυπο κουτί (Εικόνα 4.3).



Εικόνα 4.3 : Πρότυπο κουτί

4.3.1 Κουτί Κεντρικού Ελεγκτή

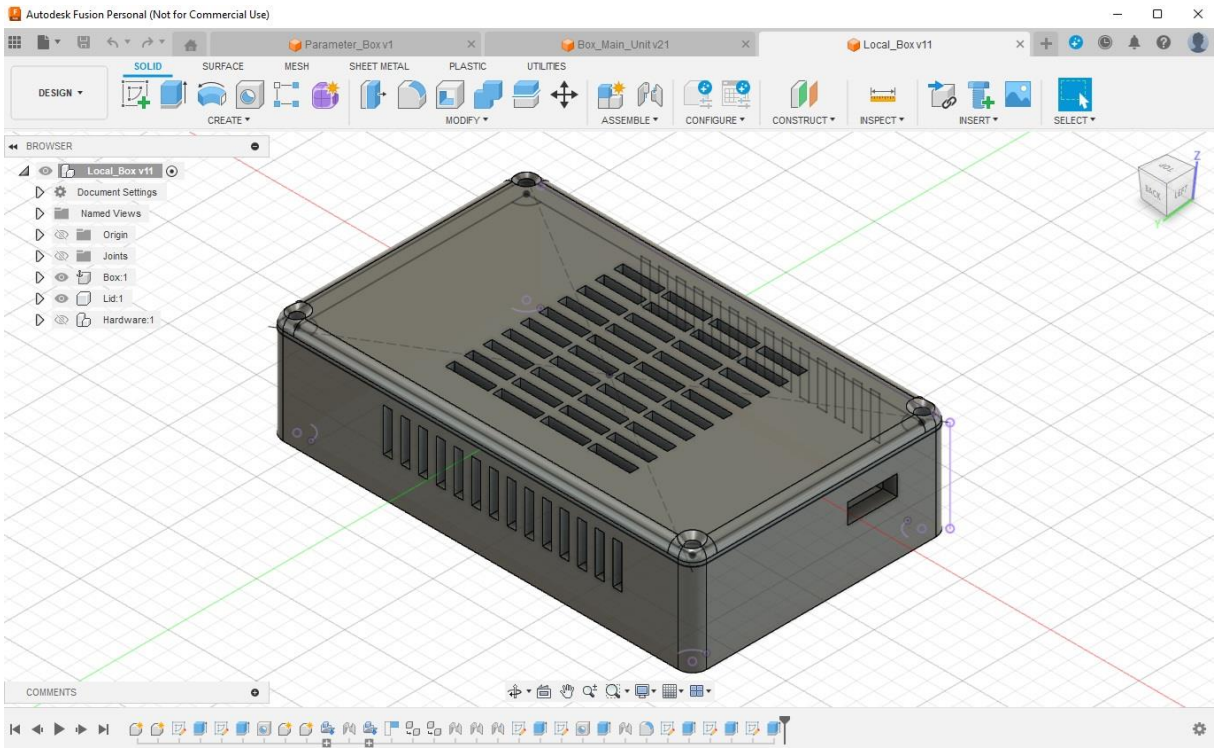
Για την σχεδίαση του κουτιού του κεντρικού ελεγκτή περάστηκαν αρχικά οι απαιτούμενες παράμετροι για να δημιουργηθεί το κουτί στις διαστάσεις που χρειάζεται. Μετά έγιναν οι ανάλογες μετατροπές που χρειαζόταν για την συγκεκριμένη κατασκευή οι οποίες ήταν πρώτα να μετρηθούν οι διαστάσεις του βύσματος τροφοδοσίας (USB type C). Στην συνέχεια έγινε η αντίστοιχη τρύπα στο πλαϊνό μέρος του κουτιού. Στο καπάκι του κουτιού τοποθετήθηκαν η οθόνη και το πληκτρολόγιο και έγινε και εδώ το ίδιο με το βύσμα τροφοδοσίας. Αφού μετρήθηκαν οι διαστάσεις της οθόνης και της καλωδιοταινίας του πληκτρολογίου με μικρόμετρο, έγιναν οι αντίστοιχες τρύπες. Η Εικόνα 4.4 δείχνει το κουτί του κεντρικού ελεγκτή.



Εικόνα 4.4 : Κουτί Κεντρικού Ελεγκτή

4.3.2 Κουτί Τοπικού Ελεγκτή

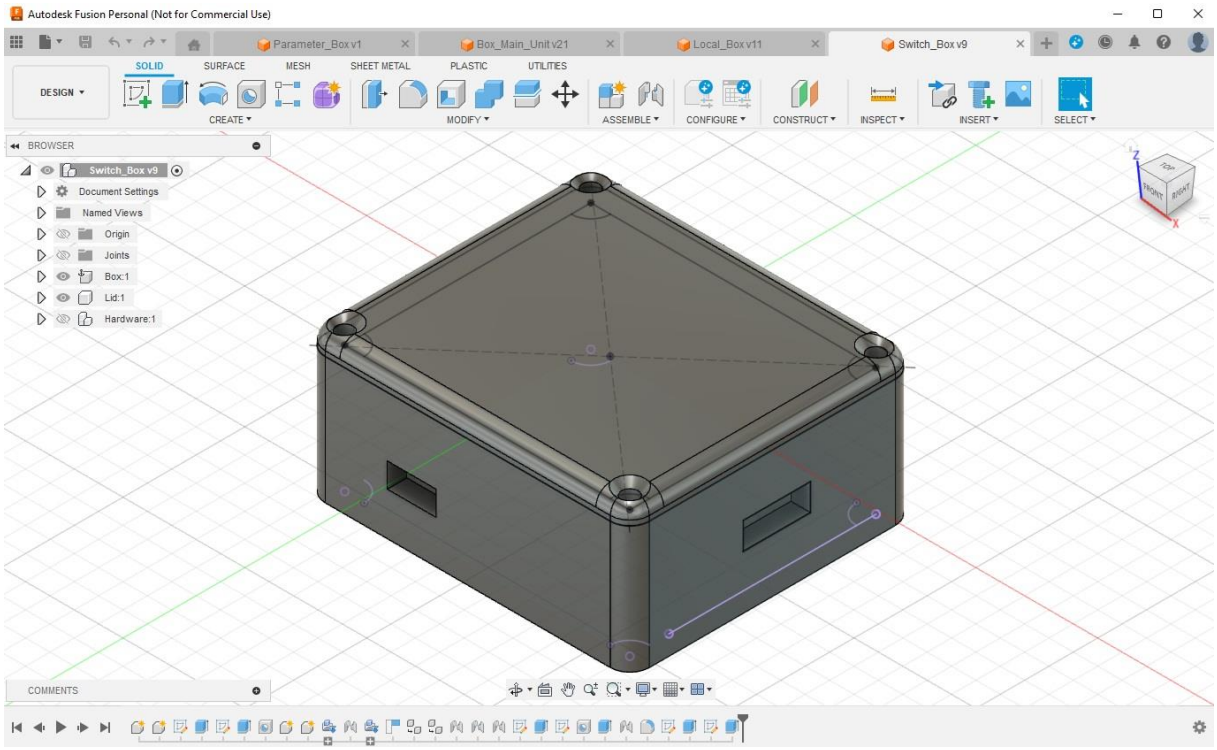
Η ίδια λογική ισχύει και για το κουτί του τοπικού ελεγκτή. Στην αρχή περάστηκαν οι απαιτούμενες διαστάσεις και μετά έγιναν όλες οι απαραίτητες αλλαγές στο κουτί όπως φαίνεται στην Εικόνα 4.5.



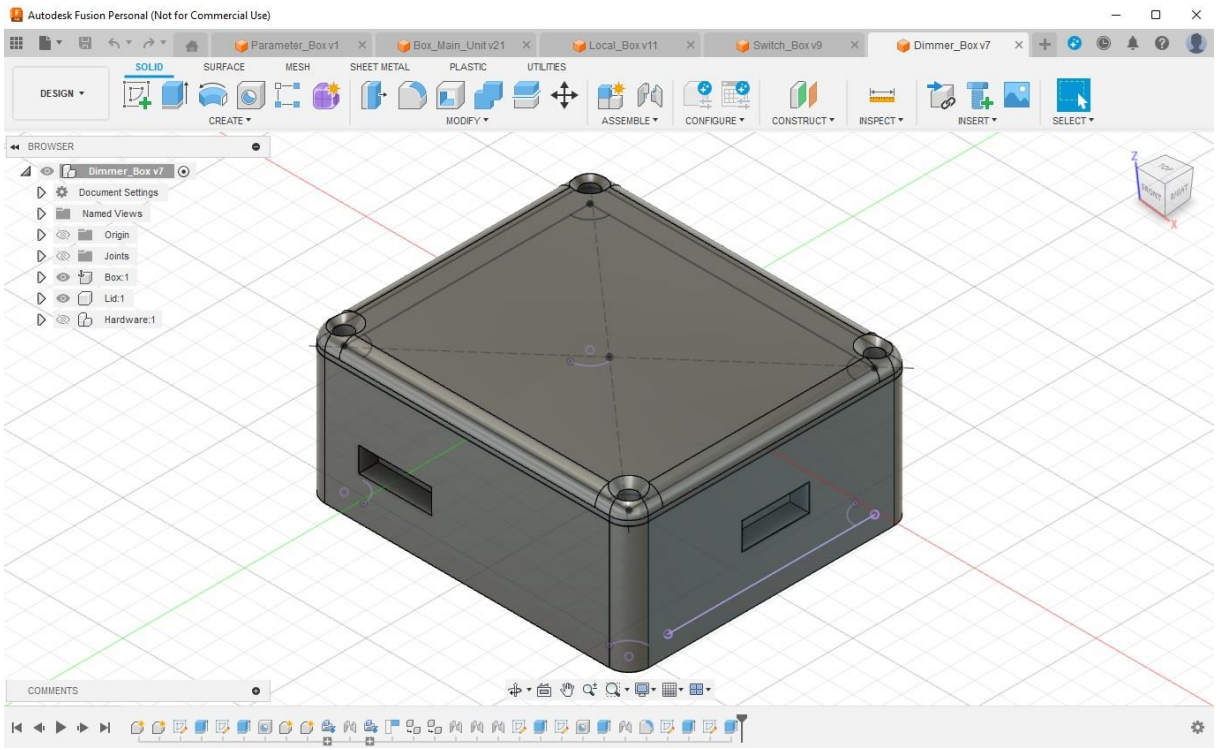
Εικόνα 4.5 : Κουτί Τοπικού Ελεγκτή

4.3.3 Κουτί Τοπικού Δέκτη

Τα κουτιά του τοπικού δέκτη διακόπτη και του τοπικού δέκτη dimmer είναι σχεδόν ίδια. Η μόνη διαφορά είναι ότι η πλαϊνή τρύπα για την σύνδεση των καλωδίων του δέκτη dimmer είναι λίγο μεγαλύτερη για να συνδέονται τρία καλώδια έναντι δύο του δέκτη διακόπτη. Στην Εικόνα 4.6 φαίνεται το κουτί του τοπικού δέκτη διακόπτη και στην Εικόνα 4.7 του τοπικού δέκτη dimmer.



Εικόνα 4.6 : Κουτί Τοπικού Δέκτη Διακόπτη



Εικόνα 4.7 : Κουτί Τοπικού Δέκτη Dimmer

4.4 Επίλογος

Η σχεδίαση ενός παραμετροποιήσιμου κουτιού ως βάση για την σχεδίαση των υπολοίπων κουτιών βοήθησε πολύ ως προς την ταχύτητα και την ευκολία σχεδίασης δίνοντας την δυνατότητα για σχεδίαση πολλών διαφορετικών κουτιών.

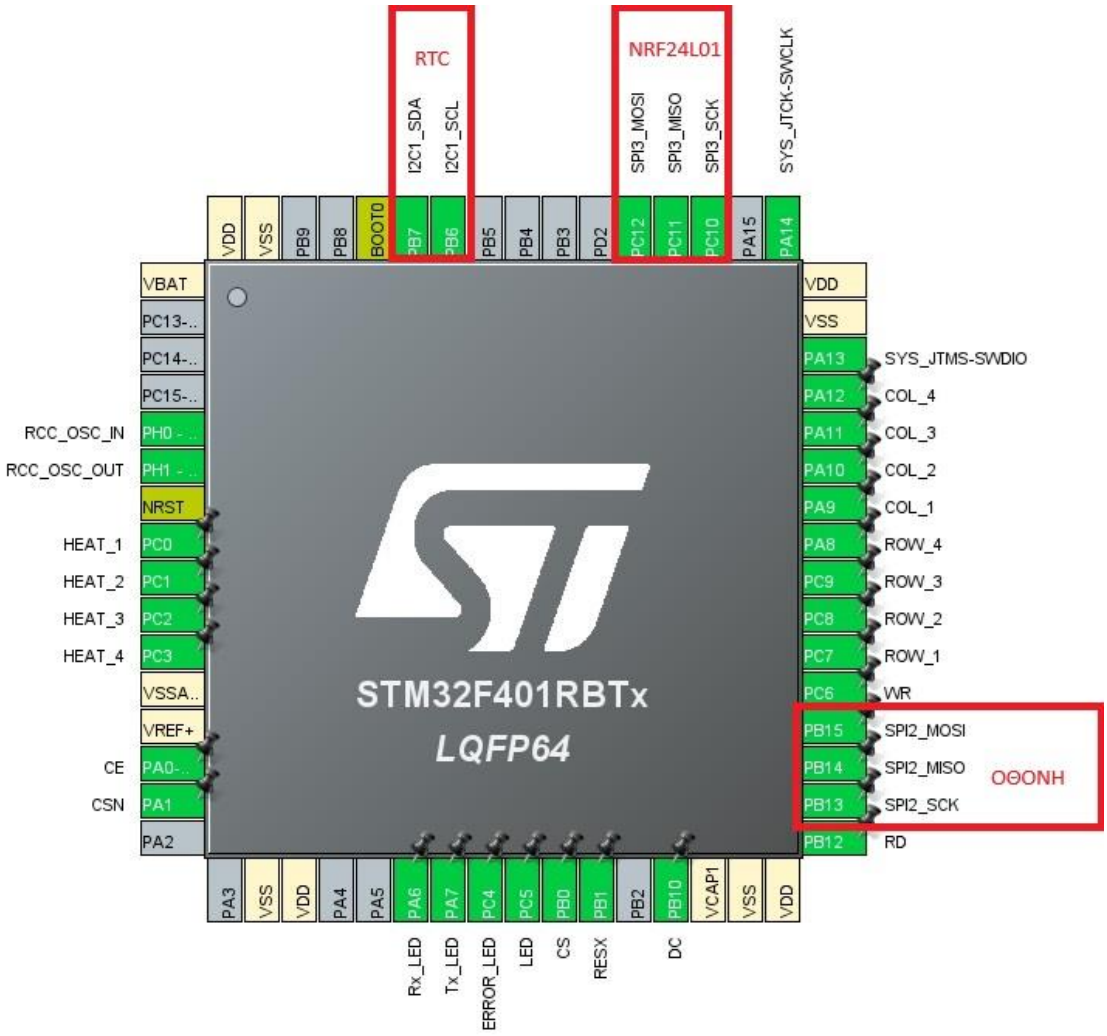
Κεφάλαιο 5ο: Λογισμικό Διαχείρισης Συστήματος

5.1 Εισαγωγή

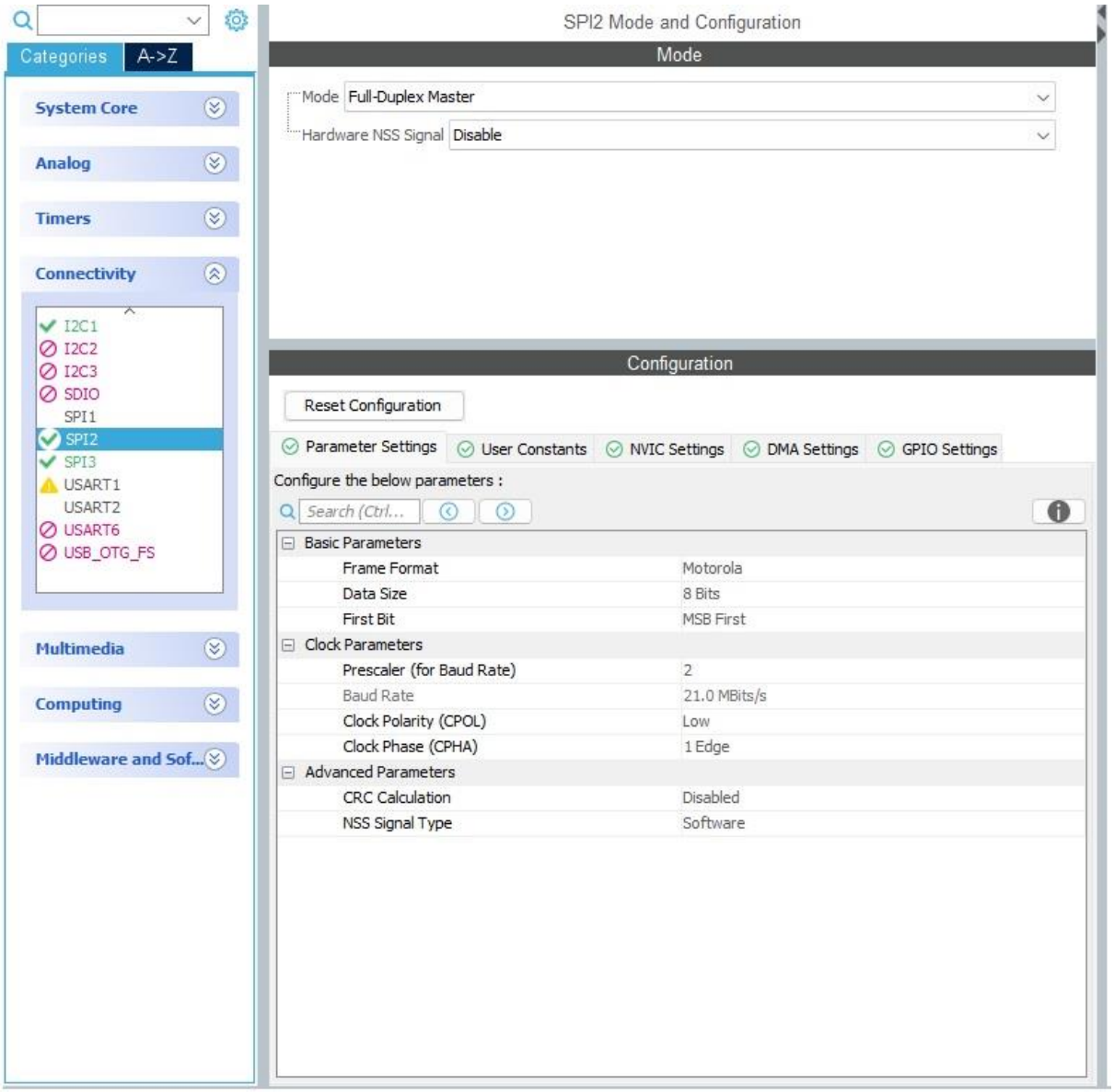
Το λογισμικό του μικροελεγκτή είναι υπεύθυνο για την σωστή λειτουργία της κάθε μονάδας του συστήματος. Παρακάτω θα αναφερθούν τα πιο σημαντικά μέρη του λογισμικού του κάθε μικροελεγκτή σε κάθε κομμάτι του συστήματος.

5.2 Λογισμικό Κεντρικού Ελεγκτή

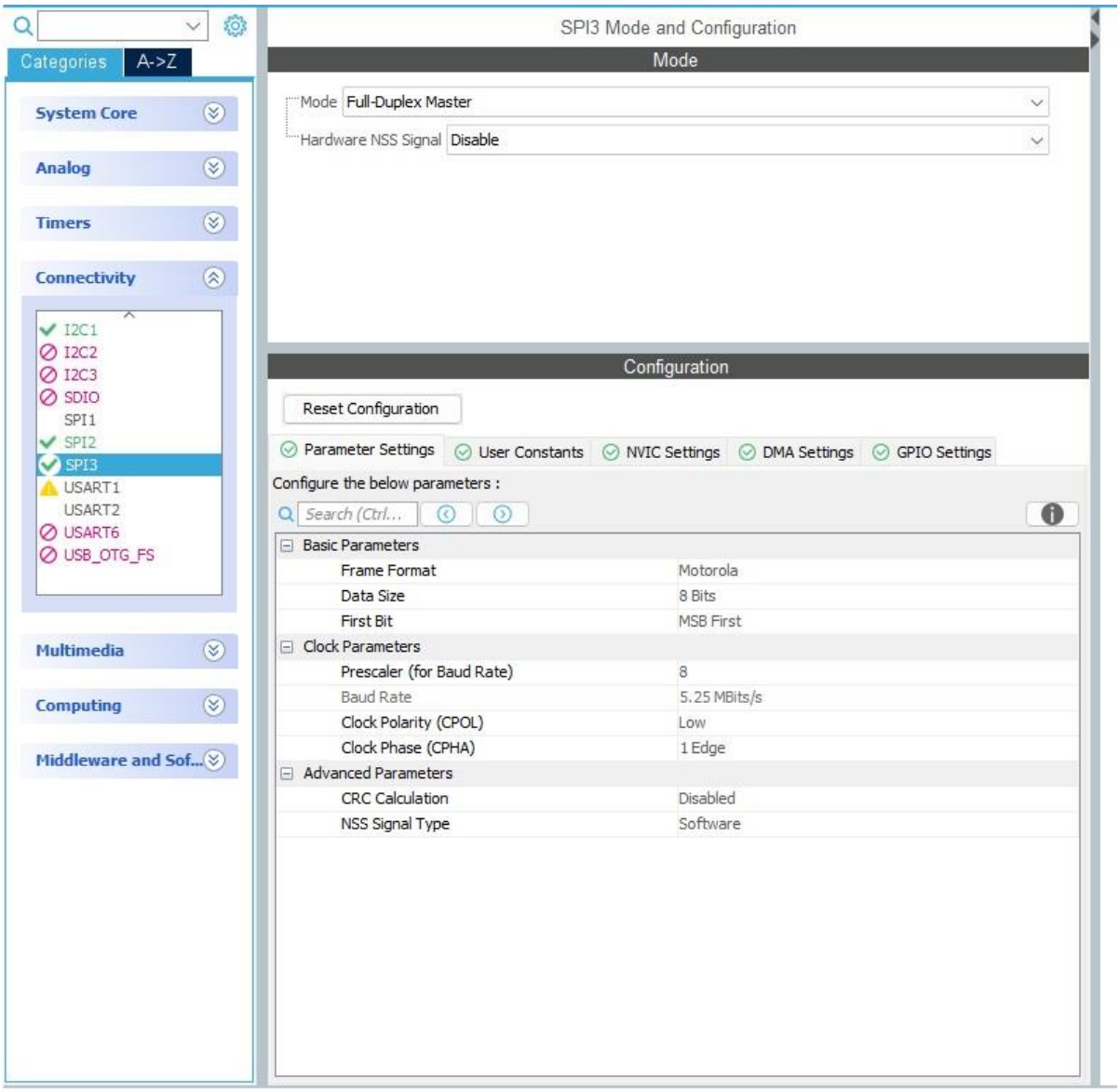
Ο μικροελεγκτής του κεντρικού ελεγκτή είναι ο STM32F401RBT6. Στην Εικόνα 5.1 φαίνονται οι συνδέσεις για την επικοινωνία με τις περιφερειακές μονάδες. Για την επικοινωνία του με τις περιφερειακές μονάδες γίνεται χρήση δύο από τα σειριακά πρωτόκολλα επικοινωνίας που διαθέτει. Αυτά είναι το πρωτόκολλο επικοινωνίας SPI για την επικοινωνία με την οθόνη και τον πομποδέκτη και το πρωτόκολλο επικοινωνίας I2C για την επικοινωνία με τον RTC. Συγκεκριμένα στην Εικόνα 5.2 φαίνονται οι ρυθμίσεις για την επικοινωνία με την οθόνη μέσω του SPI2 του μικροελεγκτή, στην Εικόνα 5.3 οι ρυθμίσεις για την επικοινωνία με τον πομποδέκτη μέσω του SPI3 του μικροελεγκτή και στην Εικόνα 5.4 οι ρυθμίσεις για την επικοινωνία με τον RTC μέσω του I2C1 του μικροελεγκτή.



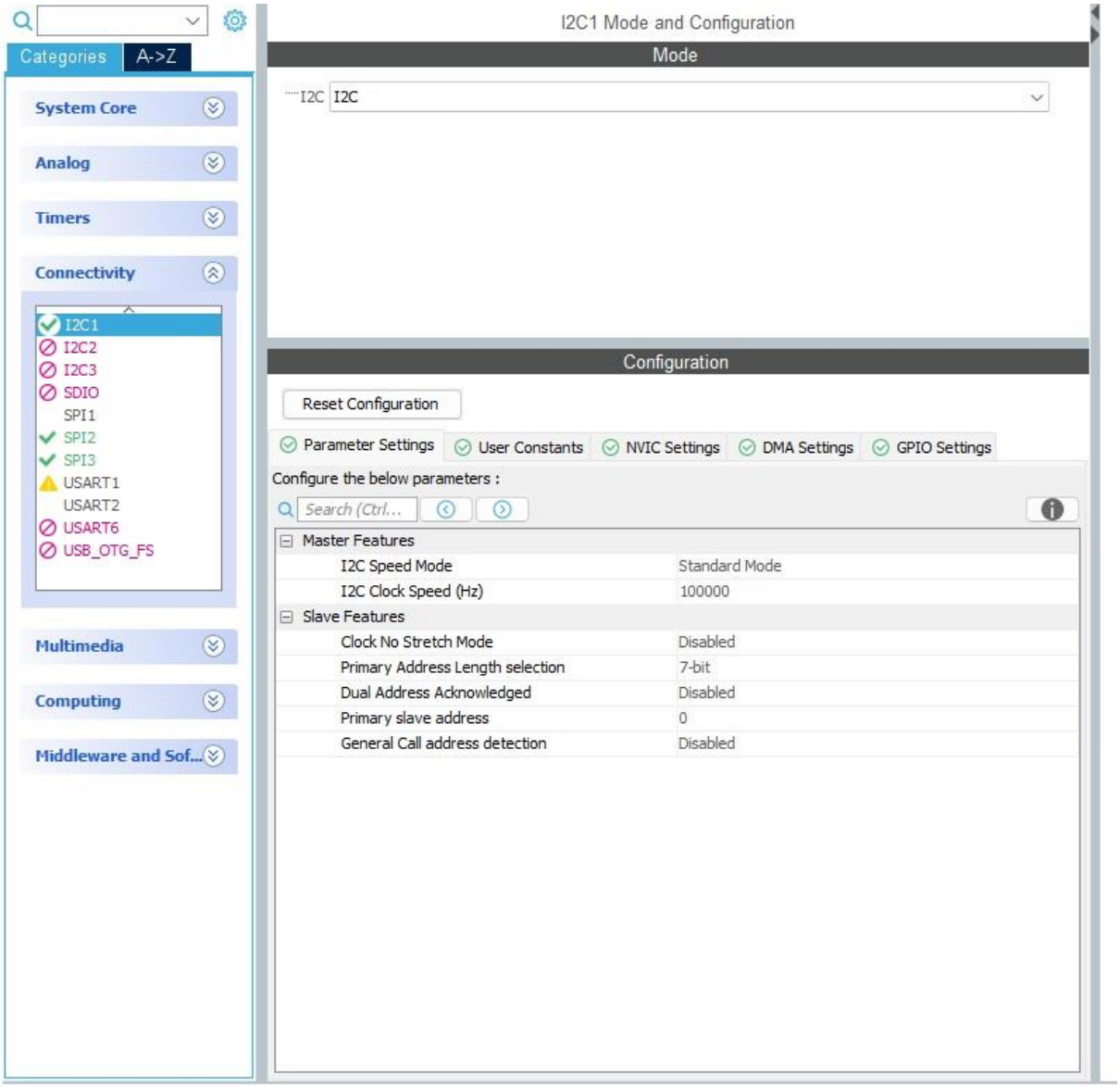
Εικόνα 5.1 : Συνδεσμολογία STM32F401RBT6 Κεντρικού Ελεγκτή



Εικόνα 5.2 : Ρυθμίσεις SPI2 Κεντρικού Ελεγκτή



Εικόνα 5.3 : Ρυθμίσεις SPI3 Κεντρικού Ελεγκτή

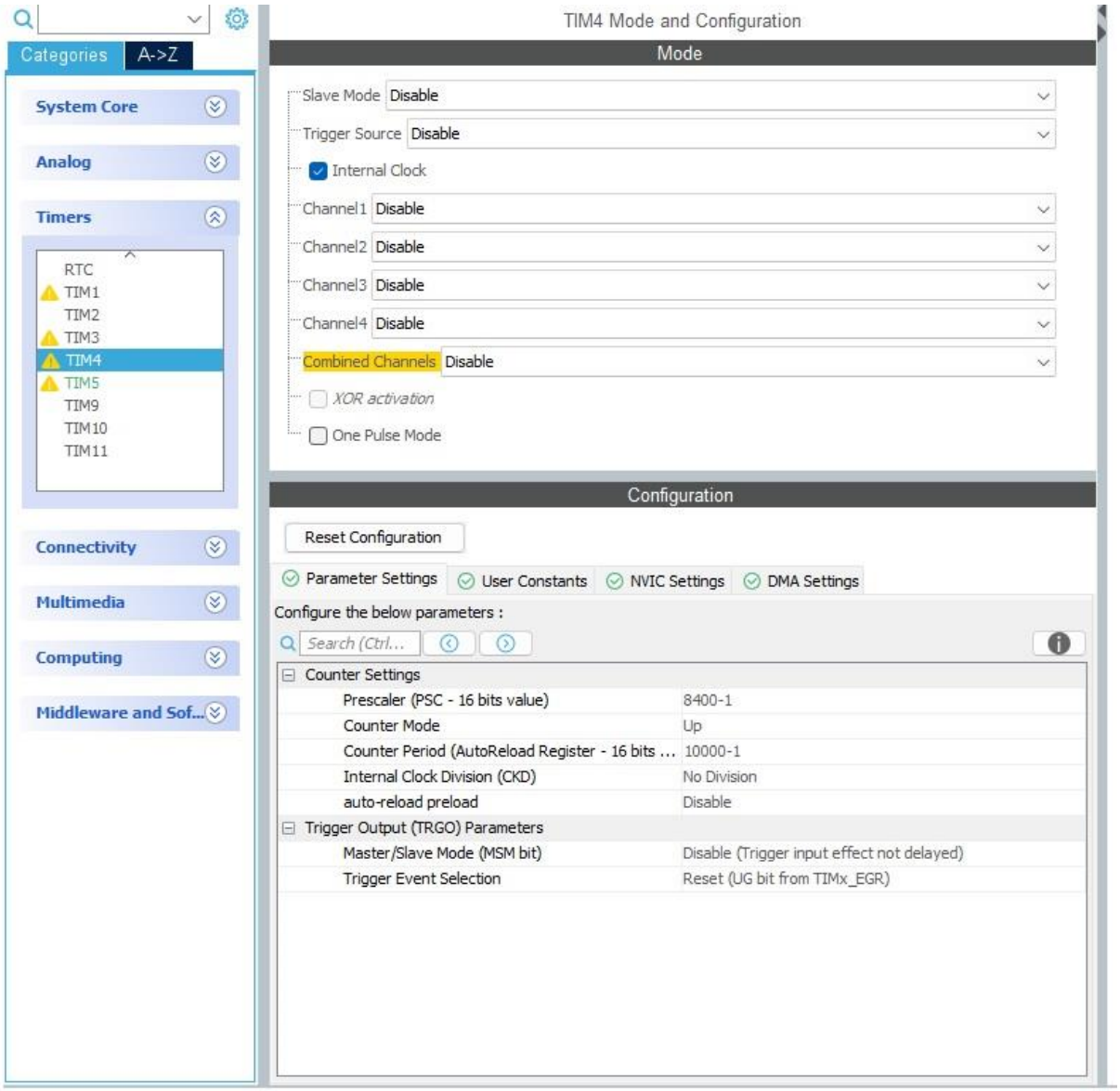


Εικόνα 5.4 : Ρυθμίσεις I2C1 Κεντρικού Ελεγκτή

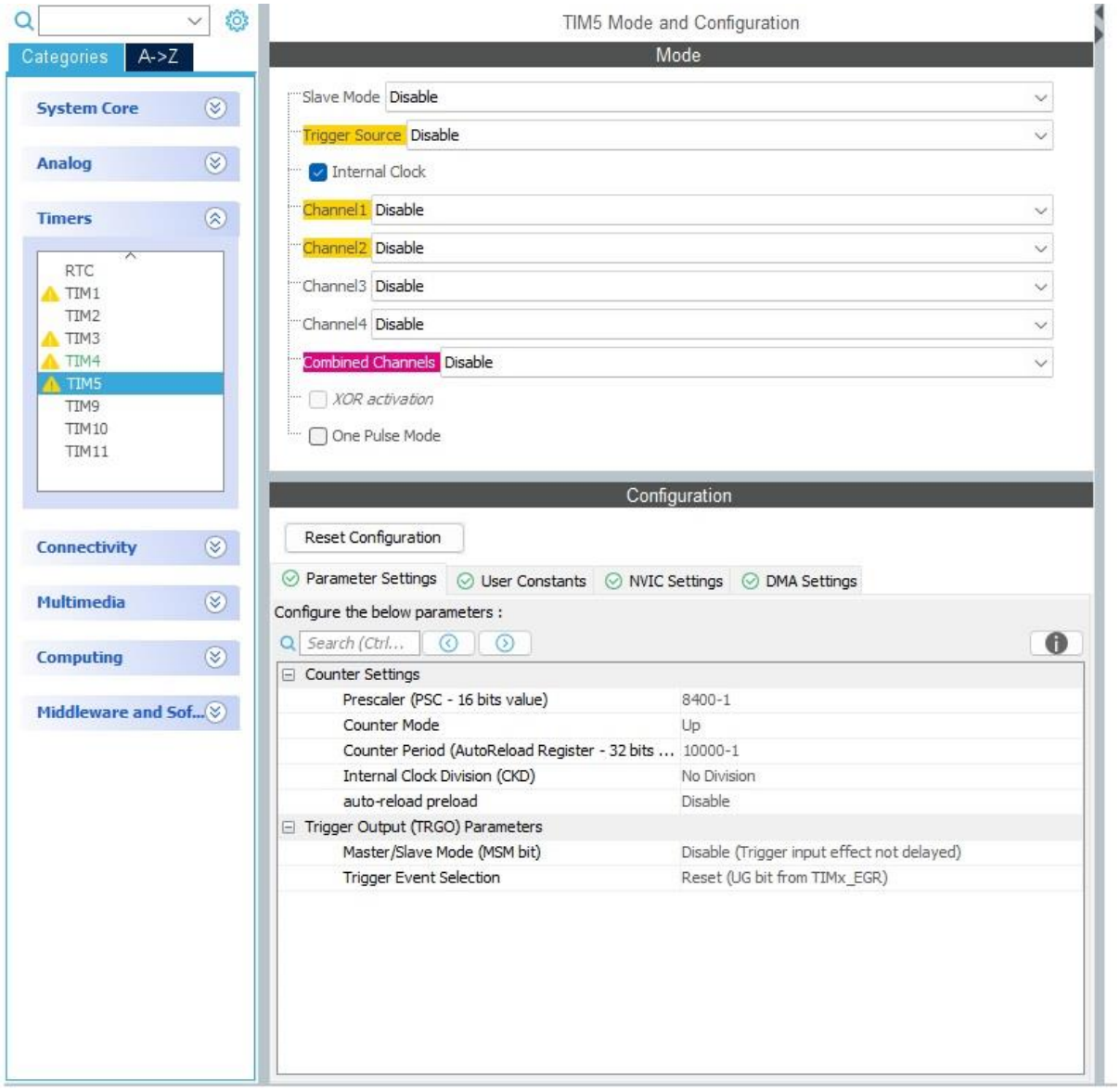
Δύο άλλα σημαντικά χαρακτηριστικά του μικροελεγκτή που χρησιμοποιήθηκαν είναι οι χρονιστές (Timers). Ο συγκεκριμένος μικροελεγκτής έχει 7 γενικής χρήσης Timers και 1 προχωρημένου επιπέδου Timer. Χρησιμοποιήθηκαν συγκεκριμένα οι TIM4 και TIM5 οι οποίοι είναι Timers γενικής χρήσης. Και οι δύο Timer ρυθμίστηκαν έτσι ώστε να δημιουργούν μια διακοπή κάθε ένα δευτερόλεπτο. Ο υπολογισμός έγινε με βάση τον τύπο που δίνει η εταιρία και είναι:

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler+1)*(Period+1)} \qquad (5.1)$$

Το Timer clock του μικροελεγκτή είναι 84MHz, το Update Event που χρειάζεται είναι 1000 ms, οπότε δίνοντας την τιμή 8400-1 για τον Prescaler και λύνοντας ως προς Period το αποτέλεσμα είναι 9999 (10000-1) όπως φαίνεται και στην Εικόνα 5.5 για το TIM4 και στην Εικόνα 5.6 για το TIM5.



Εικόνα 5.5 : Ρυθμίσεις TIM4 Κεντρικού Ελεγκτή



Εικόνα 5.6 : Ρυθμίσεις TIM5 Κεντρικού Ελεγκτή

Όπως φαίνεται από τις Εικόνες 5.5 και 5.6 οι ρυθμίσεις είναι ίδιες και για τους δύο Timers. Η διαφορά τους είναι στον τρόπο χρήσης μέσα στο πρόγραμμα. Και στους δύο Timer είναι ενεργοποιημένο το interrupt (διακοπή) αλλά μόνο ο Tim4 είναι μόνιμα ενεργοποιημένος, ενώ ο TIM5 ενεργοποιείται και απενεργοποιείται μέσα από το πρόγραμμα ανάλογα τις ανάγκες όπως φαίνεται και στην Εικόνα 5.7.

```
182 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
183 {
184 //TIM5 update EVERY 1000ms (1 sec)
185 //if time_set_menu over exit menu
186     if (htim->Instance == TIM5)
187     {
188         update_menu = update_menu + 1;
189         if (update_menu > time_set_menu)
190         {
191             time_end_menu = 1;
192             update_menu = 0;
193             HAL_TIM_Base_Stop_IT(&htim5);
194         }
195         else
196         {
197             time_end_menu = 0;
198         }
199     }
200
201 //TIM4 update EVERY 1000ms (1 sec)
202 //if time_set_check is over then start checking every local peripheral
203         if (htim->Instance == TIM4)
204         {
205             update_check = update_check + 1;
206             if (update_check > time_set_check)
207             {
208                 time_end_check = 1;
209                 update_check = 0;
210             }
211             else
212             {
213                 time_end_check = 0;
214             }
215
216             led_time_update = led_time_update + 1;
217             if (led_time_update >= led_time_off)
218             {
219                 led_time_update = 0;
220                 led_time_flag = 1;
221             }
222
223             wait_time_update = wait_time_update + 1;
224             if (wait_time_update >= wait_time_set)
225             {
226                 wait_time_update = 0;
227                 wait_time_end = 1;
228             }
229         }
230
231 }
```

Εικόνα 5.7 : Interrupt από Timer Κεντρικού Ελεγκτή

Το TIM5 ενεργοποιείται όταν γίνει πρόσβαση σε κάποιο κατάλογο (menu) του προγράμματος και μετά από ένα χρονικό διάστημα εάν δεν πατηθεί κανένα πλήκτρο επιστρέφει στην αρχική οθόνη του προγράμματος. Τότε απενεργοποιεί και τον Timer.

Το TIM4 είναι μόνιμα ενεργοποιημένο και ελέγχει τρείς παραμέτρους:

- Τον χρόνο για να απενεργοποιηθούν τα LED του κυκλώματος
- Τον χρόνο κατά τον οποίο γίνεται περιοδική επικοινωνία με τους τοπικούς ελεγκτές
- Και τον χρόνο που δίνει σε κάθε τοπικό ελεγκτή για να απαντήσει

Μετά τις αρχικοποιήσεις και τις δηλώσεις των καταχωρητών το κυρίως πρόγραμμα χωρίζεται σε δύο κυρίως μέρη. Σε αυτό της καταχώρησης όλων των δεδομένων από τον χρήστη μέσω του πληκτρολογίου και σε αυτό της επικοινωνίας του κεντρικού ελεγκτή με τους τοπικούς ελεγκτές.

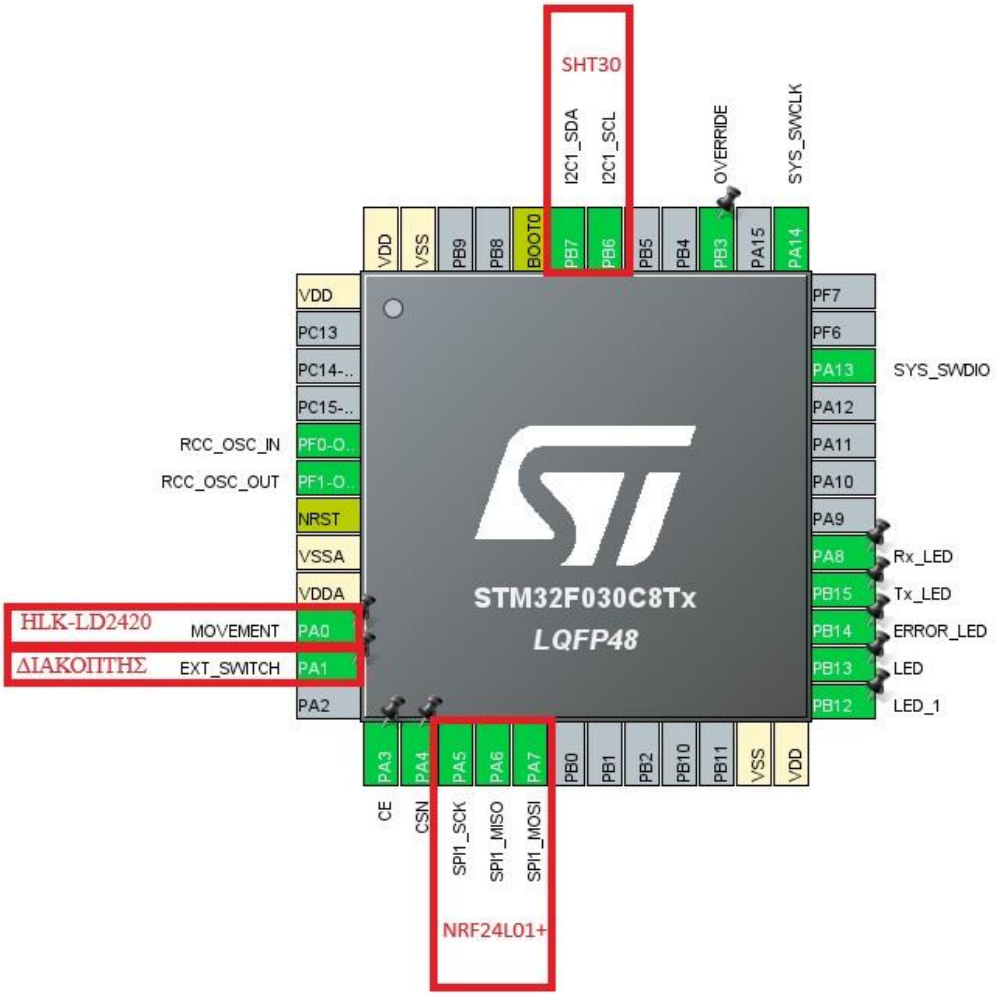
Στο πρώτο μέρος ο μικροελεγκτής εμφανίζει στον χρήστη καταλόγους (menu) μέσα από τους οποίους ο χρήστης διαλέγει την επιλογή που θέλει να αλλάξει και καταχωρεί την τιμή που επιθυμεί. Αυτό είναι και το μεγαλύτερο σε όγκο μέρος του προγράμματος.

Στο δεύτερο μέρος, μετά από ένα συγκεκριμένο χρονικό διάστημα ο μικροελεγκτής στέλνει πληροφορίες σε κάθε έναν από τους τοπικούς ελεγκτές που ελέγχει τον έναν μετά τον άλλον. Ξεκινάει με τον πρώτο. Στέλνει τις πληροφορίες και περιμένει για ένα συγκεκριμένο χρονικό διάστημα απάντηση από τον τοπικό ελεγκτή. Ο τοπικός ελεγκτής με την σειρά του μόλις εκτελέσει τις λειτουργίες που πρέπει να κάνει απαντάει στον κεντρικό ελεγκτή και αυτός επικοινωνεί με τον δεύτερο τοπικό ελεγκτή κ.ο.κ.. Σε περίπτωση που παρέλθει το χρονικό διάστημα αναμονής του κεντρικού ελεγκτή τότε αυτός άμεσα ξεκινάει την επικοινωνία με τον επόμενο τοπικό ελεγκτή.

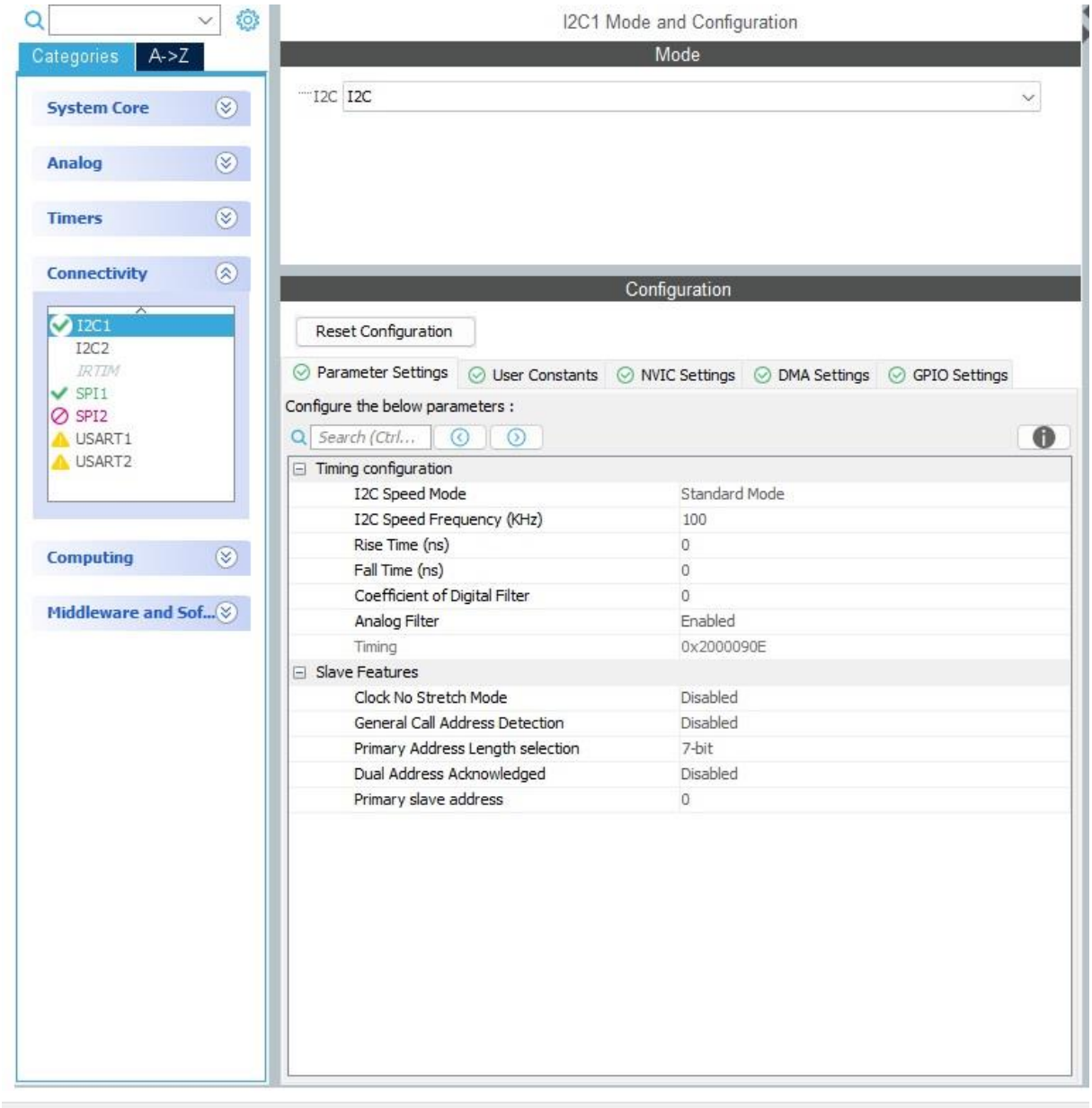
5.3 Λογισμικό Τοπικού Ελεγκτή

Ο τοπικός ελεγκτής ελέγχεται από τον μικροελεγκτή STM32F030C8T6. Οι συνδέσεις με τις περιφερειακές μονάδες φαίνονται στην Εικόνα 5.8. Και εδώ γίνεται χρήση δύο από τα σειριακά πρωτόκολλα επικοινωνίας που διαθέτει. Ο αισθητήρας θερμοκρασίας και υγρασίας χρησιμοποιεί το πρωτόκολλο επικοινωνίας I2C μέσω του I2C1 του μικροελεγκτή με τις ρυθμίσεις να φαίνονται στην Εικόνα 5.9. Ο πομποδέκτης NRF24L01+ χρησιμοποιεί το πρωτόκολλο επικοινωνίας SPI μέσω του SPI1 του μικροελεγκτή και οι ρυθμίσεις φαίνονται στην εικόνα 5.10.

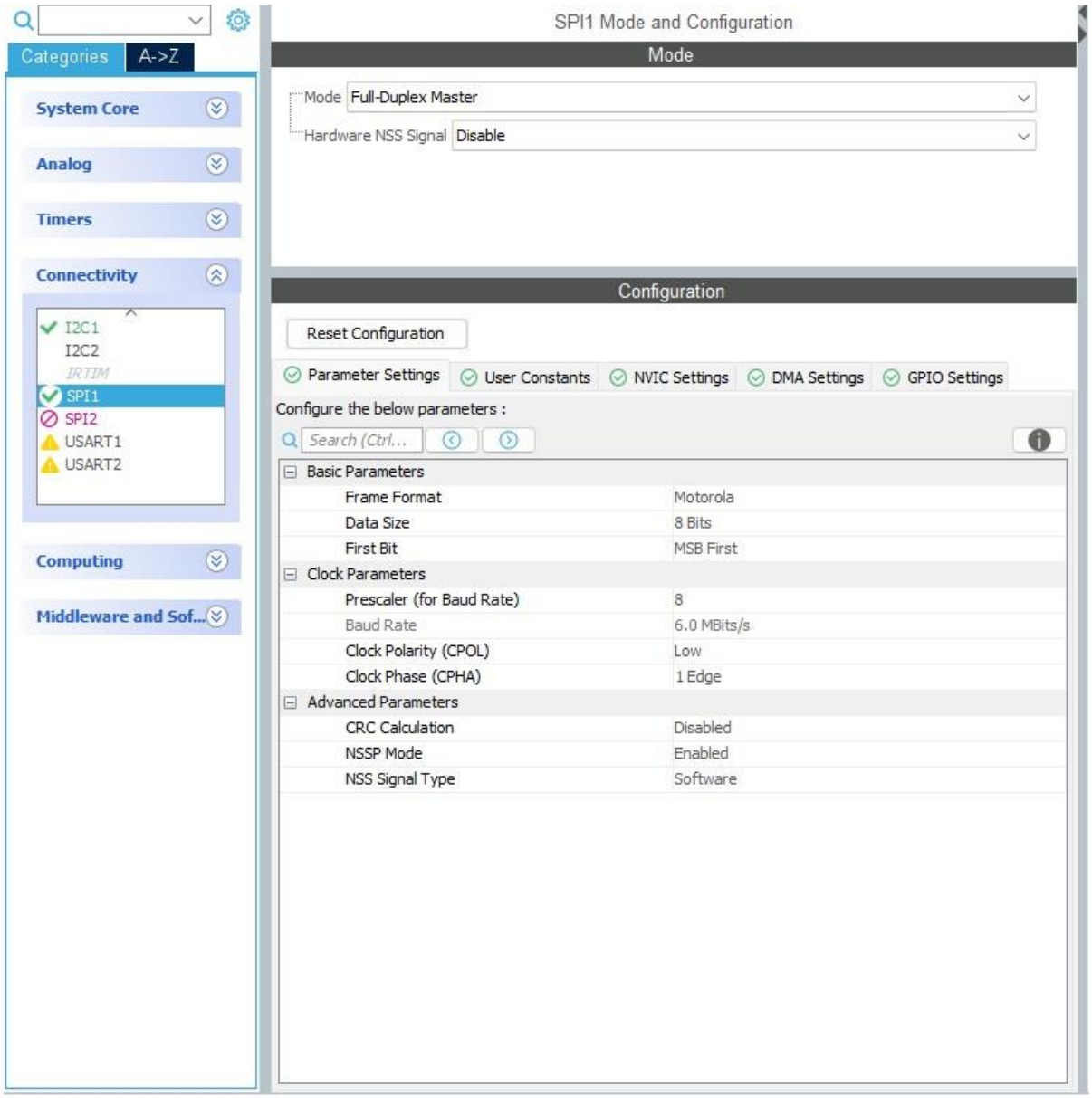
Από την μονάδα HLK-LD2420 χρησιμοποιείται μόνο η έξοδος που γίνεται λογικό 1 όταν ανιχνευτεί ανθρώπινη παρουσία και λογικό 0 όταν δεν ανιχνεύεται ανθρώπινη παρουσία.



Εικόνα 5.8 : Συνδεσμολογία STM32F030C8T6 Τοπικού Ελεγκτή

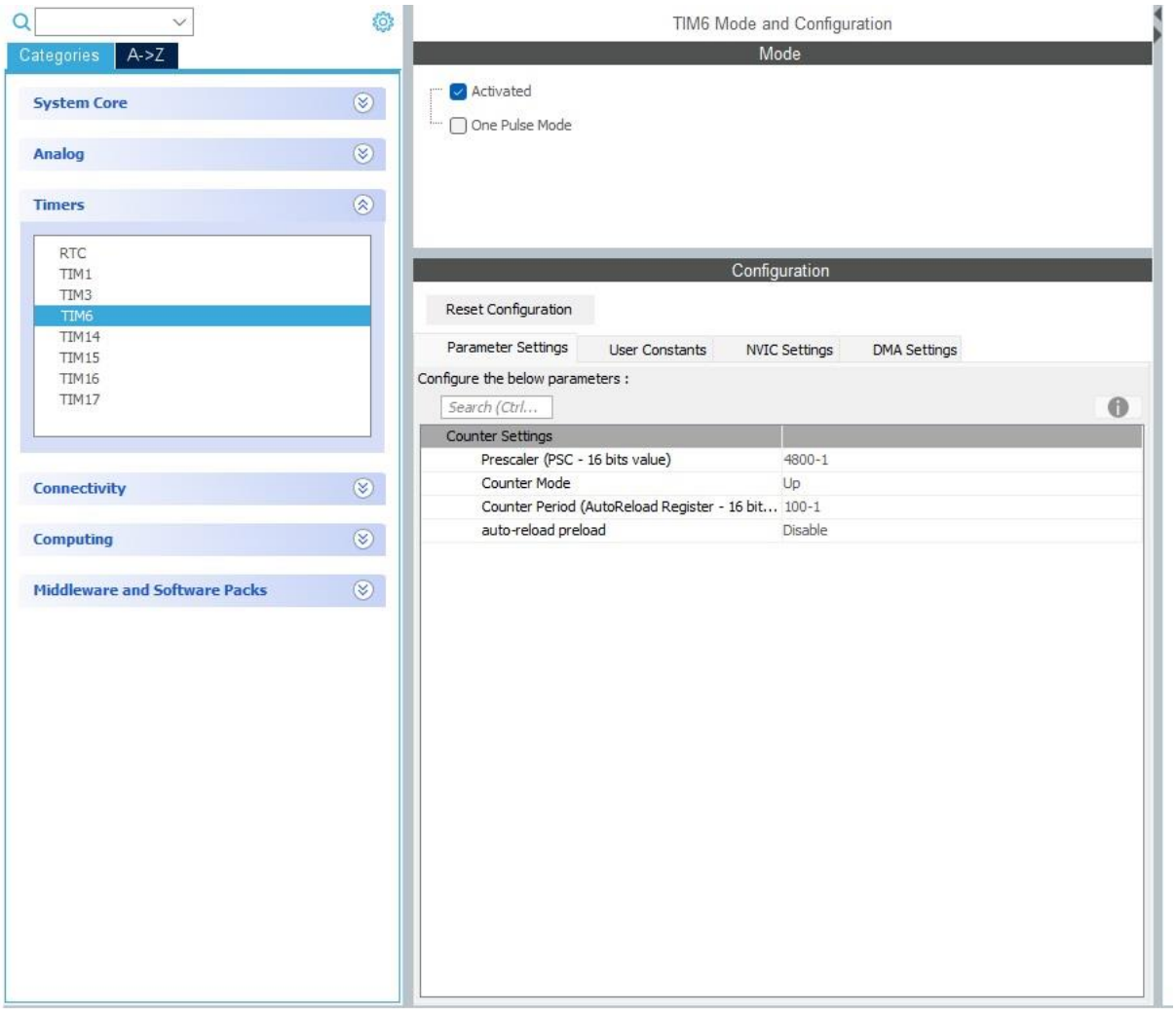


Εικόνα 5.9 : Ρυθμίσεις I2C1 Τοπικού Ελεγκτή



Εικόνα 5.10 : Ρυθμίσεις SPI1 Τοπικού Ελεγκτή

Στον μικροελεγκτή του τοπικού ελεγκτή γίνεται χρήση μόνο ενός από τους επτά Timers του TIM6. Η συχνότητα λειτουργίας του μικροελεγκτή είναι 48MHz και δίνοντας την τιμή 4800-1 για τον Prescaler και λύνοντας ως προς Period το αποτέλεσμα είναι 99 (100-1) και μας δίνει διακοπές κάθε 100ms (Εικόνα 5.11).



Εικόνα 5.11 : Ρυθμίσεις TIM6 Τοπικού Ελεγκτή

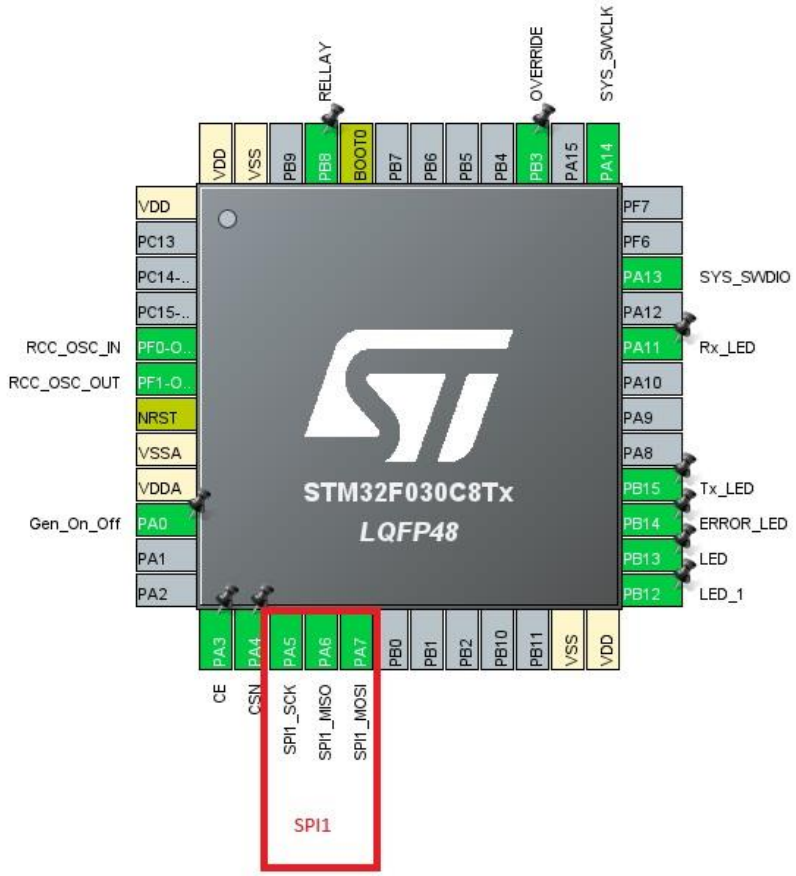
Αφού γίνουν οι δηλώσεις των καταχωρητών και όλες οι αρχικοποιήσεις τότε γίνεται από τον αισθητήρα SHT30 μια μέτρηση της θερμοκρασίας και της υγρασίας και αποθηκεύονται στους αντίστοιχους καταχωρητές. Στην συνέχεια ελέγχει αν υπάρχει κάποια λήψη δεδομένων από τον κεντρικό ελεγκτή. Αν δεν υπάρχει τίποτα τότε ξανά επιστρέφει στην αρχή του προγράμματος. Αν έχει γίνει λήψη δεδομένων τότε πρώτα ελέγχει αν αυτά τα δεδομένα προορίζονται για αυτόν τον τοπικό ελεγκτή. Αν προορίζονται τα δεδομένα για αυτόν τότε στέλνει απάντηση στον κεντρικό ελεγκτή ότι πήρε τα δεδομένα και στέλνει αίτηση αναμονής, δηλαδή να περιμένει ο κεντρικός ελεγκτής μέχρι να επικοινωνήσει με όλους τους τοπικούς δέκτες ο τοπικός ελεγκτής.

Στην συνέχεια αντιγράφει όλα τα δεδομένα που έλαβε στους αντίστοιχους καταχωρητές και ελέγχει αν υπάρχει κίνηση στον χώρο. Μετά συγκρίνει την θερμοκρασία και την υγρασία που μέτρησε με την θερμοκρασία και την υγρασία που έλαβε από τον κεντρικό ελεγκτή και σε συνδυασμό με το αν υπάρχει κίνηση ή όχι στον χώρο καθώς επίσης και αν είναι ανοιχτό κάποιο παράθυρο ή κάποια μπαλκονόπορτα ή όχι τροποποιεί και αποθηκεύει τα δεδομένα που θα στείλει στην συνέχεια.

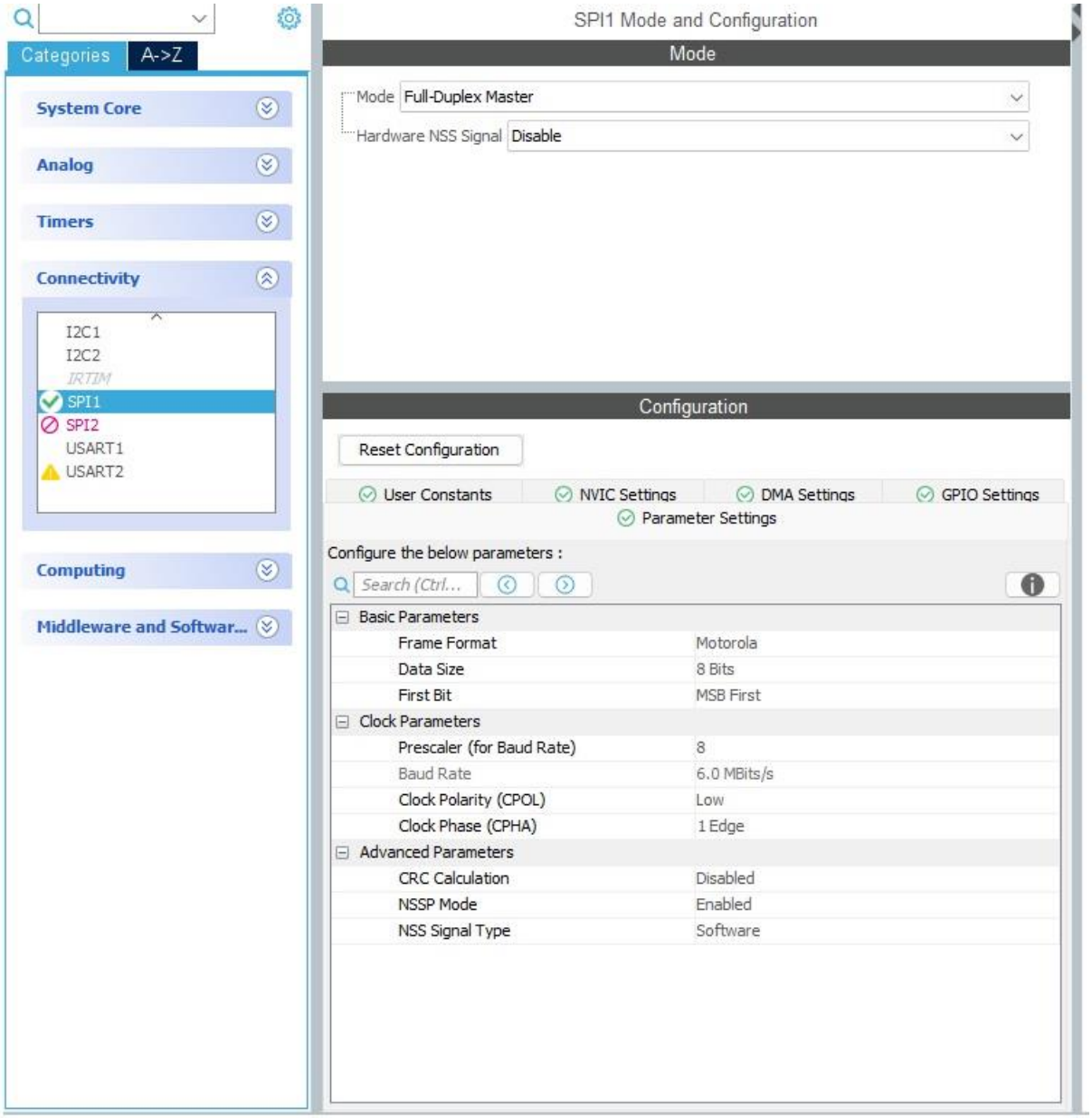
Το επόμενο βήμα είναι να στείλει σε όλους τους τοπικούς δέκτες έναν προς έναν τα δεδομένα που έλαβε και τα δεδομένα που τροποποίησε και να πάρει στην συνέχεια απάντηση από τον κάθε έναν. Αν μετά από ένα συγκεκριμένο χρονικό διάστημα δεν πάρει απάντηση από κάποιον τοπικό δέκτη τότε άμεσα προσπαθεί να επικοινωνήσει με τον επόμενο δέκτη.

5.4 Λογισμικό Τοπικού Δέκτη Διακόπτη

Ο μικροελεγκτής STM32F030C8T6 είναι υπεύθυνος για την λειτουργία του τοπικού δέκτη διακόπτη και η συνδεσμολογία του φαίνεται στην Εικόνα 5.12. Ο πομποδέκτης NRF24L01+ είναι η μόνη περιφερειακή μονάδα που συνδέεται μέσω του σειριακού πρωτόκολλου επικοινωνίας SPI μέσω του SPI1 του μικροελεγκτή και οι ρυθμίσεις φαίνονται στην εικόνα 5.13.

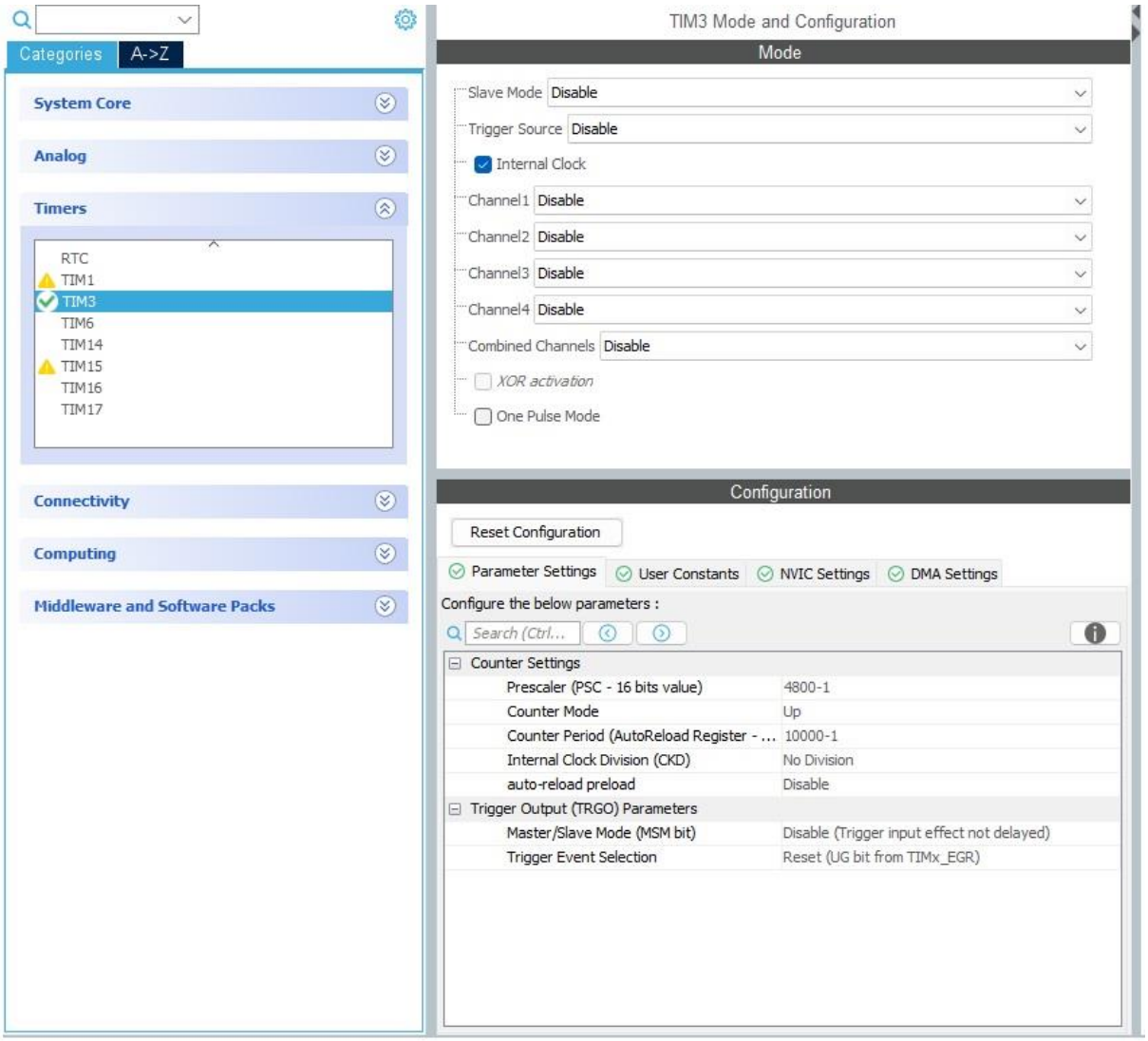


Εικόνα 5.12 : Συνδεσμολογία STM32F030C8T6 Τοπικού Δέκτη Διακόπτη



Εικόνα 5.13 : Ρυθμίσεις SPI1 Τοπικού Δέκτη Διακόπτη

Στον τοπικό δέκτη διακόπτη γίνεται χρήση ενός Timer του TIM3. Ο TIM3 δεν είναι μόνιμα ενεργοποιημένος Αλλά ενεργοποιείται μόνο όταν ξεκινάει η μέτρηση του χρόνου καθυστέρησης απενεργοποίησης του ηλεκτρονόμου. Οι ρυθμίσεις φαίνονται στην Εικόνα 5.14.



Εικόνα 5.14 : Ρυθμίσεις TIM3 Τοπικού Δέκτη Διακόπτη

Μόλις ενεργοποιηθεί ο TIM3 δίνει διακοπές κάθε 1000ms όπως μπορεί να υπολογιστεί μέσω του τύπου 5.1. Η συχνότητα λειτουργίας του μικροελεγκτή είναι 48MHz Prescaler είναι 4800-1 υπολογίζοντας το Period είναι 9999 (10000-1).

Η λειτουργία του προγράμματος είναι αρκετά απλή. Μόλις γίνουν οι δηλώσεις των καταχωρητών και οι αρχικοποιήσεις ο μικροελεγκτής ελέγχει αν έλαβε κάποια δεδομένα. Αν δεν έλαβε περιμένει μέχρι να λάβει. Αν έλαβε τότε ελέγχει αν τα δεδομένα είναι για αυτόν. Αν δεν είναι για αυτόν τότε πηγαίνει ξανά στην αρχή του προγράμματος. Αν είναι για αυτόν τότε στέλνει απάντηση στον τοπικό ελεγκτή ότι έλαβε τα δεδομένα έτσι ώστε να συνεχίσει με τον επόμενο τοπικό δέκτη.

Στην συνέχεια εκτελεί τις εντολές που πρέπει σύμφωνα με τα δεδομένα που παρέλαβε. Αρχικά ενημερώνει τον χρόνο καθυστέρησης απενεργοποίησης του ηλεκτρονόμου. Μετά ελέγχει από τα δεδομένα αν υπάρχει κίνηση ή όχι. Αν υπάρχει κίνηση τότε ελέγχει αν είναι σε λειτουργία ημέρας ή νύχτας. Αν είναι σε λειτουργία ημέρας τότε ενεργοποιεί τον ηλεκτρονόμο. Αν είναι σε λειτουργία νύχτας ελέγχει τον διακόπτη BYPASS αν είναι απενεργοποιημένος άμεσα απενεργοποιεί τον ηλεκτρονόμο. Αν το BYPASS είναι ενεργοποιημένο τότε ενεργοποιεί τον ηλεκτρονόμο.

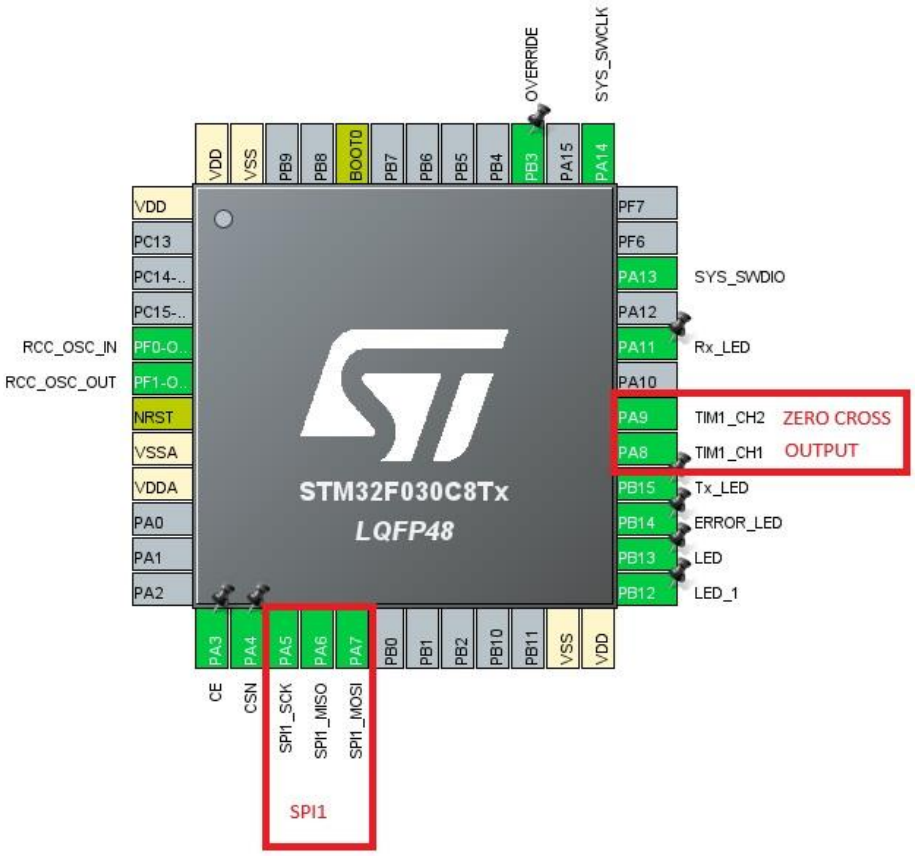
Στην περίπτωση που δεν υπάρχει κίνηση αν ο ηλεκτρονόμος είναι απενεργοποιημένος παραμένει απενεργοποιημένος. Αν όμως ο ηλεκτρονόμος είναι ενεργοποιημένος τότε ενεργοποιεί το TIM3 και αρχίζει να μετρά ο χρόνος καθυστέρησης απενεργοποίησης του ηλεκτρονόμου. Μόλις συμπληρωθεί ο χρόνος αυτός τότε απενεργοποιεί και τον ηλεκτρονόμο.

5.5 Λογισμικό Τοπικού Δέκτη Αφυγραντήρα

Το λογισμικό του τοπικού δέκτη αφυγραντήρα είναι σχεδόν ίδιο με αυτό του τοπικού δέκτη διακόπτη με ορισμένες αλλαγές. Χρησιμοποιεί το ίδιο Hardware (υλικό). Η πρώτη διαφορά είναι ότι δεν κάνει χρήση κανενός Timer διότι η ενεργοποίηση και η απενεργοποίηση του ηλεκτρονόμου γίνεται άμεσα. Επίσης η εντολή ενεργοποίησης ή απενεργοποίησης του ηλεκτρονόμου προέρχεται από διαφορετική θέση στα δεδομένα.

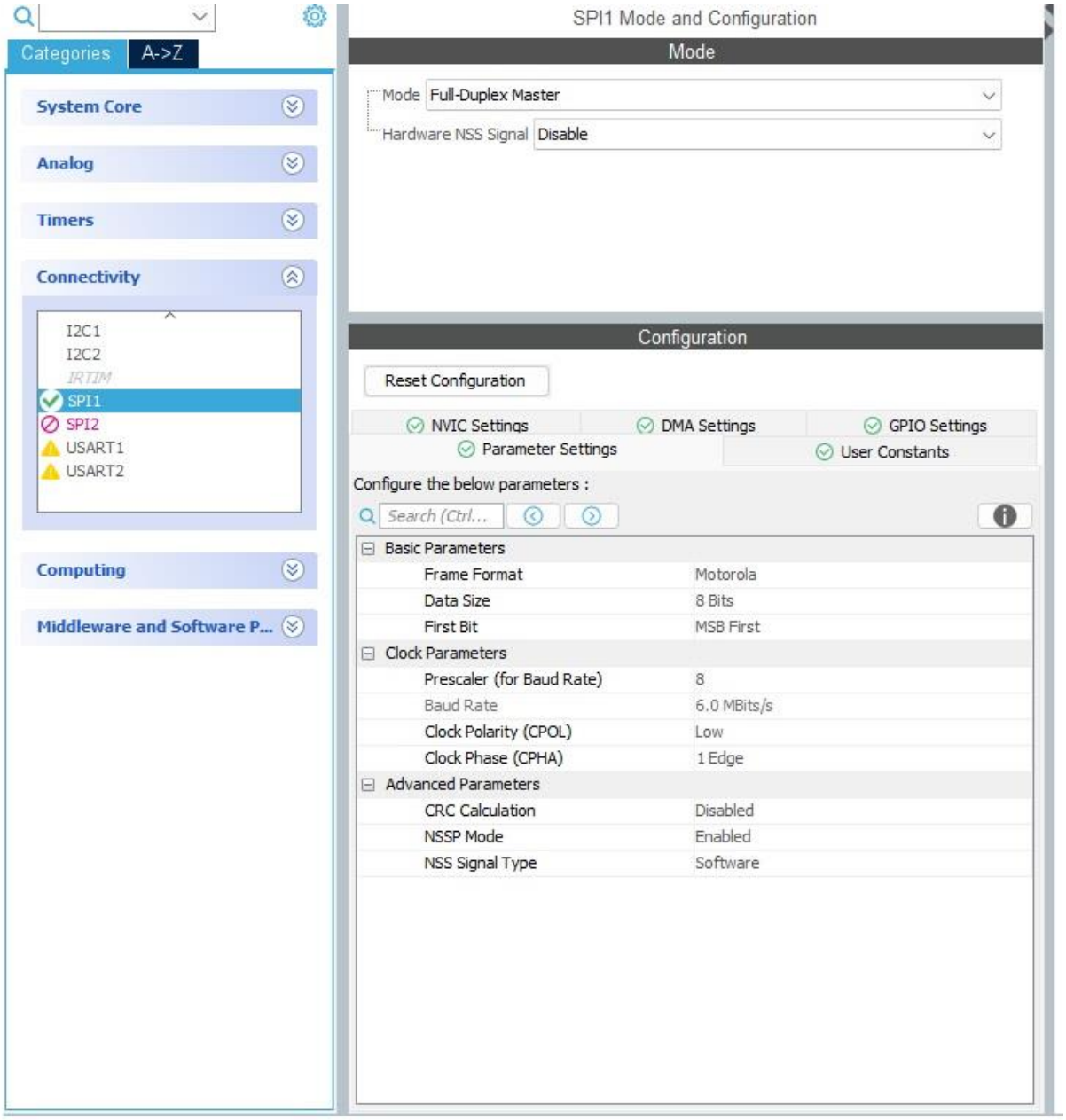
5.6 Λογισμικό Τοπικού Δέκτη Dimmer

Ο μικροελεγκτής του τοπικού δέκτη dimmer είναι ο STM32F030C8T6. Στην Εικόνα 5.15 φαίνεται η συνδεσμολογία του.



Εικόνα 5.15 : Συνδεσμολογία STM32F030C8T6 Τοπικού Δέκτη Dimmer

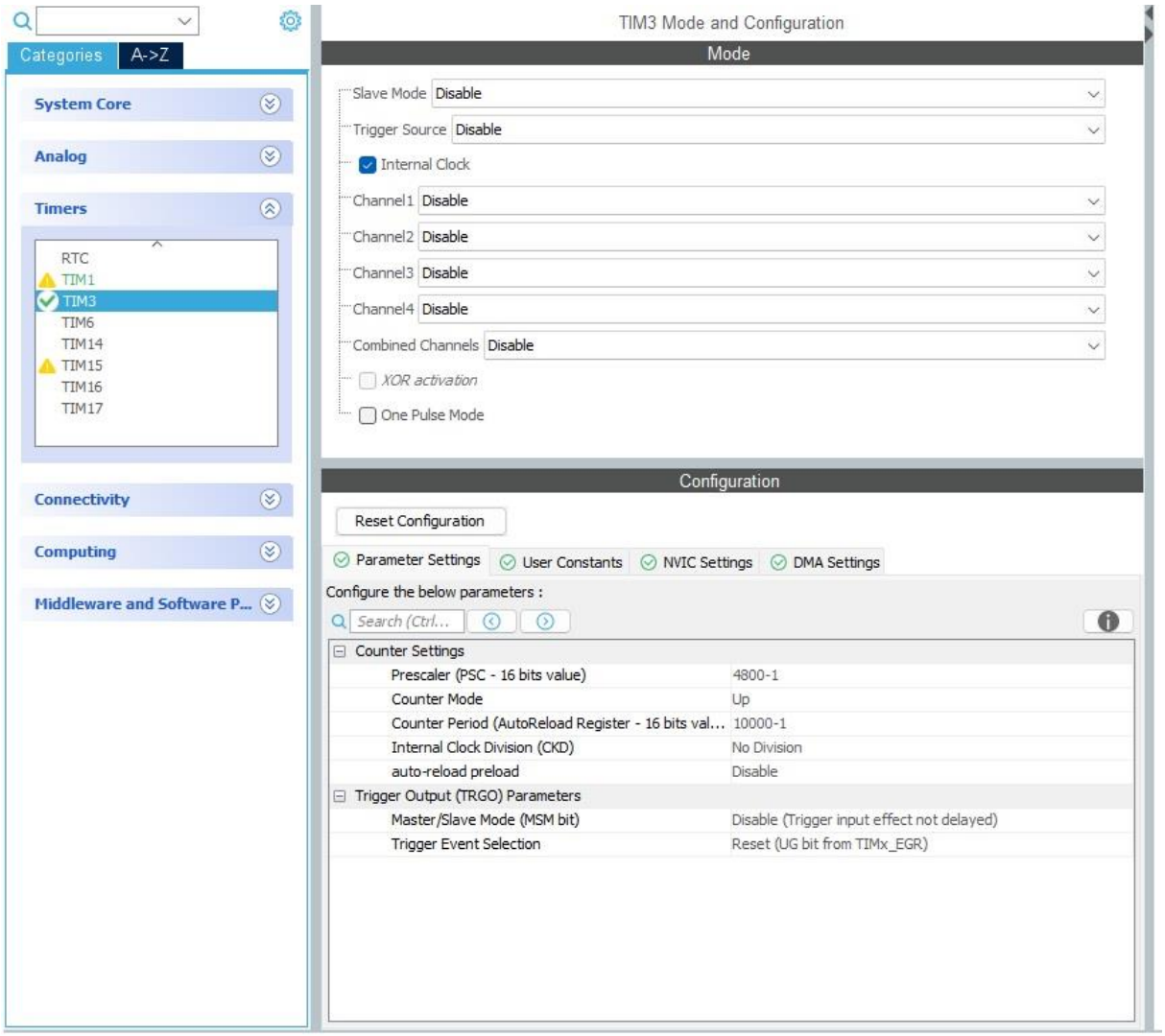
Όπως και στον τοπικό δέκτη διακόπτη έτσι και εδώ ο πομποδέκτης NRF24L01+ είναι η μόνη περιφερειακή μονάδα που συνδέεται μέσω του σειριακού πρωτόκολλου επικοινωνίας SPI μέσω του SPI1 του μικροελεγκτή και οι ρυθμίσεις φαίνονται στην εικόνα 5.16.



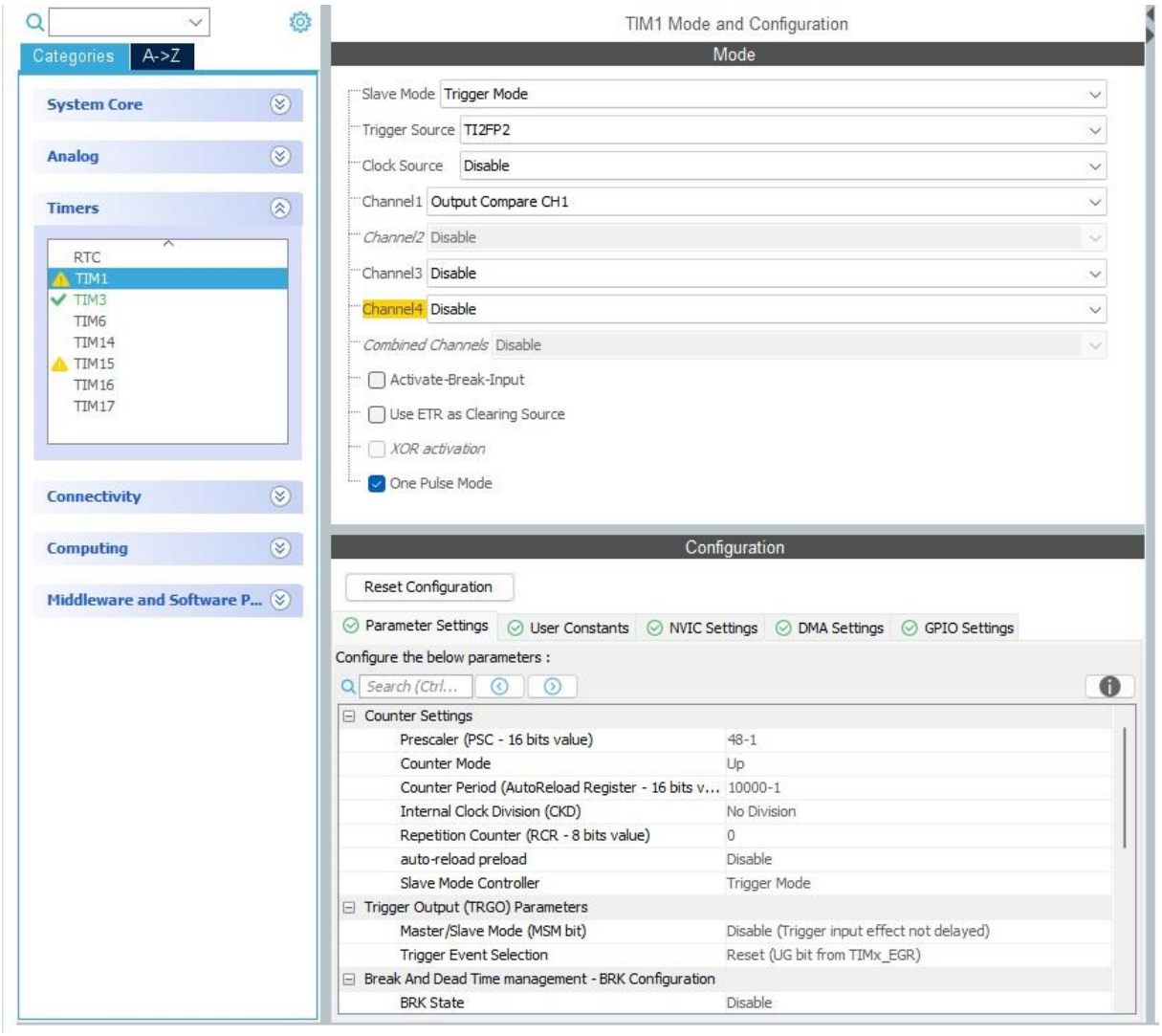
Εικόνα 5.16 : Ρυθμίσεις SPI1 Τοπικού Δέκτη Dimmer

Στον τοπικό δέκτη dimmer είναι ενεργοποιημένα δύο Timer, το TIM3 για την καθυστέρηση απενεργοποίησης της εξόδου, όπως και στον τοπικό δέκτη διακόπτη και τον TIM1 ο οποίος χρησιμοποιείται ως one pulse mode.

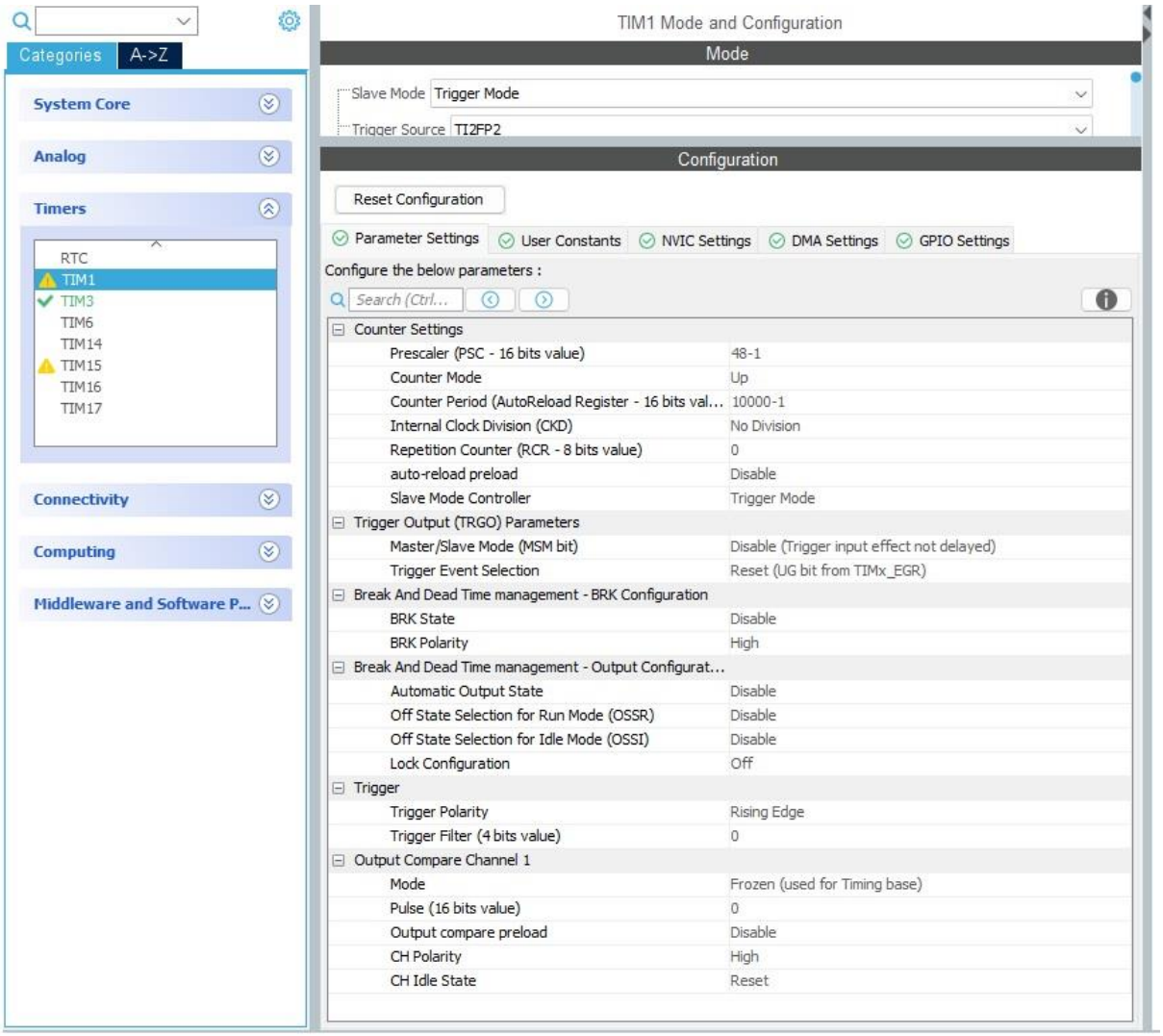
Οι ρυθμίσεις του TIM3 φαίνονται στην Εικόνα 5.17 ενώ οι ρυθμίσεις του TIM1 στις Εικόνες 5.18 και 5.19.



Εικόνα 5.17 : Ρυθμίσεις TIM3 Τοπικού Δέκτη Dimmer



Εικόνα 5.18 : Ρυθμίσεις TIM1 Τοπικού Δέκτη Dimmer



Εικόνα 5.19 : Ρυθμίσεις TIM1 Τοπικού Δέκτη Dimmer

Με την λειτουργεία one pulse mode μόλις έρθει ένας παλμός (trigger) στην είσοδο που έχει οριστεί στον TIM1 τότε στην έξοδο δημιουργείται ένας παλμός συγκεκριμένης διάρκειας με ρυθμιζόμενη καθυστέρηση έναρξης.

Όπως φαίνεται από την Εικόνα 5.18 ο TIM1 είναι ρυθμισμένος ως ‘trigger mode’ με είσοδο trigger το κανάλι 2 το οποίο είναι ο ακροδέκτης PA9 του μικροελεγκτή. Σε αυτήν την είσοδο είναι συνδεμένη η έξοδος του κυκλώματος zero cross και κάθε φορά που μηδενίζεται η τάση στην είσοδο του κυκλώματος zero cross υπάρχει ένας παλμός στον ακροδέκτη PA9 του μικροελεγκτή. Η έξοδος είναι από το κανάλι 1 το οποίο είναι ο ακροδέκτης PA8 του μικροελεγκτή. Επίσης πρέπει να είναι ενεργοποιημένη και η λειτουργία του one pulse mode του TIM1.

Στην Εικόνα 5.19 φαίνεται η μέγιστη διάρκεια του παλμού εξόδου (Counter Period) η οποία είναι 10ms όση δηλαδή και η ημιπερίοδος της τάσης του δικτύου. Η ρύθμιση της τιμής Pulse στο Output Compare Channel 1 δίνει τον χρόνο καθυστέρησης έναρξης του παλμού εξόδου.

Έτσι ρυθμίζοντας τον χρόνο καθυστέρησης έναρξης του παλμού εξόδου μπορούμε να ρυθμίσουμε την έξοδο του dimmer. Βάζοντας την τιμή 0 στην έξοδο υπάρχει ολόκληρη η ημιπερίοδος της τάσης του δικτύου ενώ βάζοντας την τιμή 5000 υπάρχει η μισή, δηλαδή το 50%.

Η λειτουργία του προγράμματος είναι παρόμοια με αυτήν του τοπικού δέκτη διακόπτη. Η διαφορά είναι στο σημείο όπου βρίσκεται σε λειτουργία νύχτας και το BYPASS είναι ενεργοποιημένο. Σε αυτήν την περίπτωση αν είναι ενεργοποιημένη η λειτουργία NIGHT LIGHT τότε αλλάζει η τιμή Pulse στο Output Compare Channel 1 από 0 σε 6000 καθυστερώντας την έναρξη του παλμού εξόδου με συνέπεια το Triac να πάρει παλμό στο Gate με καθυστέρηση και έτσι να έχουμε μειωμένη ένταση στον φωτισμό. Προσοχή οι λάμπες θα πρέπει να είναι dimmable.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

Τα αποτελέσματα της εξοικονόμησης ενέργειας με την χρήση του συστήματος φαίνονται μετά από μια μεγάλη σχετικά περίοδο χρήσης. Αυτό συμβαίνει διότι οι μεγαλύτερες καταναλώσεις ενέργειας σε ένα σπίτι είναι κυρίως αυτή της θέρμανσης του χώρου. Η ηλεκτρική ενέργεια συνήθως σπαταλάτε σε φωτισμό που μένει αναμμένος χωρίς να υπάρχει ανάγκη και σε συσκευές όπως ανεμιστήρες, τηλεοράσεις κ.α. που έχουν όμως μικρή σχετικά κατανάλωση ενέργειας. Και εκεί βέβαια όσο μικρή και αν φαίνεται η σπατάλη ενέργειας σε βάθος χρόνου είναι ένα αξιοσημείωτο ποσοστό ενέργειας που χάνεται και αν υπολογιστεί συνολικά σε μια πόλη για παράδειγμα τα μεγέθη αυξάνονται ραγδαία.

Όσο για την θέρμανση των χώρων ενός σπιτιού η αίσθηση της ζέστης σε είναι διαφορετική για κάθε άνθρωπο. Άλλος μπορεί να κρυώνει, άλλος μπορεί να ζεσταίνεται και άλλος μπορεί να νοιώθει ικανοποιημένος με την ίδια θερμοκρασία. Αυτό όμως δεν εμποδίζει βάζοντας σε λειτουργία το σύστημα εξοικονόμησης ενέργειας να μειώνεται η θερμοκρασία σε χώρους όπου δεν είναι κανένας έστω και για έναν βαθμό εξοικονομώντας έτσι ενέργεια εκεί που δεν χρειάζεται να σπαταλάτε.

Μια βελτίωση στο σύστημα είναι η σύνδεση ενός δέκτη διακόπτη για στόρια εκεί όπου υπάρχουν ηλεκτρικά στόρια έτσι ώστε τον χειμώνα μόλις αισθανθεί ένας αισθητήρας τον ήλιο να ανεβαίνουν και το καλοκαίρι να γίνεται το αντίθετο. Το ίδιο μπορεί να γίνει και με ηλεκτρικές τέντες.

Επίσης μπορεί να καταργηθεί το πληκτρολόγιο και στην θέση του να χρησιμοποιηθεί οθόνη με λειτουργία αφής και όλες οι ρυθμίσεις να γίνονται απευθείας μέσω της οθόνης.

Οι πλακέτες που είναι για τον φωτισμό του χώρου θα μπορούσαν να γίνουν λίγο πιο μικρές για να χωράνε μέσα στα μπουάτ ή πίσω από τους διακόπτες και να μην φαίνονται καθόλου.

Άλλη βελτίωση είναι η κατασκευή ξεχωριστού αισθητήρα θερμοκρασίας και υγρασίας με μικρή οθόνη και μικρή κατανάλωση ενέργειας για λειτουργία με επαναφορτιζόμενη μπαταρία τύπου 18650 έτσι ώστε να γίνει φορητός ο αισθητήρας θερμοκρασίας και υγρασίας του χώρου.

Τέλος η κατασκευή μιας μονάδας με υπέρυθρο πομποδέκτη για τον χειρισμό κλιματιστικών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Energy use in EU households in 2022 lowest since 2016 () [Online]. Διαθέσιμο σε: https://ec.europa.eu/eurostat/web/p roducts-eurostat-news/w/ddn-20240605-2 [Ανακτήθηκε: 25 Ιούν. 2024].

[2] Playing my part () [Online]. Διαθέσιμο σε: https://energy.ec.europa.eu/topics/markets-and-consumers/actions-and-measures-energy-prices/playing-my-part_en [Ανακτήθηκε: 25 Ιούν. 2024].

[3] "Nrf24 series - nordic semiconductor," nRF24 Series - Nordic Semiconductor, https://www.nordicsemi.com/Products/nRF24-series (accessed Jun. 25, 2024).

[4] "HLK-LD2420 24Ghz human body micro motion sensing detection radar sensor module HLK-LD2410 low cost solution_Radar Module_Products_ Shenzhen Hi-Link Electronic Co., Ltd.," *www.hlktech.net*. https://www.hlktech.net/index.php?id=1150 (accessed Jun. 25, 2024).

[5] S. AG, "SHT30A-dis-B," ±3% Digital automotive-grade humidity and temperature sensor, https://sensirion.com/products/catalog/SHT30A-DIS-B (accessed Jun. 25, 2024).

[6] "STM32F030C8," STMicroelectronics, https://www.st.com/en/microcontrollers-microprocessors/stm32f030c8.html (accessed Jun. 25, 2024).

[7] "Linear and low dropout (LDO) regulators," Diodes Incorporated, https://www.diodes.com/part/view/AP2114 (accessed Jun. 25, 2024).

[8] "3.5inch SPI module ILI9488 SKU:MSP3520," 3.5inch SPI Module ILI9488 SKU:MSP3520 - LCD wiki, http://www.lcdwiki.com/3.5inch_SPI_Module_ILI9488_SKU:MSP3520 (accessed Jun. 25, 2024).

[9] "DS3231," DS3231 Datasheet and Product Info | Analog Devices, https://www.analog.com/en/products/ds3231.html (accessed Jun. 25, 2024).

[10] "STM32F401RB," STMicroelectronics, https://www.st.com/en/microcontrollers-microprocessors/stm32f401rb.html (accessed Jun. 25, 2024).

[11] "HLK-LD2410C 24Ghz human presence induction distance detection radar sensor module support BLE/Uart Adjustment paraments_radar module_products_ shenzhen hi-link electronic co., ltd..," Shenzhen Hi-Link Electronic Co., Ltd., https://www.hlktech.net/index.php?id=1095 (accessed Jun. 25, 2024).

[12] "AZ9371 - Power Relay," ZETTLER electronics GmbH - worldwide.competence in components - A ZETTLER GROUP Company, https://www.zettlerelectronics.com/en/products/relays/elektromechanical-relais.html?--zettler_products-elecmechrelais%5B%40package%5D=zettler.products&--zettler_products-elecmechrelais%5B%40controller%5D=elecmechrelais&--zettler_products-elecmechrelais%5B%40action%5D=show&--zettler_products-elecmechrelais%5B%40format%5D=html&--zettler_products-

elecmechrelais%5BelecMechRelais%5D%5B__identity%5D=138fa80a-2716-11ef-8fef-901b0ebd9b15 (accessed Jun. 25, 2024).

[13] “Compare autodesk fusion vs autodesk fusion for personal use,” Autodesk, https://www.autodesk.com/products/fusion-360/personal (accessed Jun. 25, 2024).

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας STM32F401RBT6 Κεντρικού Ελεγκτή

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
***
  * @project name :
      Receiver_Transmitter_Main_Control_Unit_STM32F401RBT6_New_Test
  * @author             :     VPK Engineering
  * @file           :   main.c
  * @brief          :   MAIN CONTROL UNIT WITH ILI9488 DISPLAY, RTC AND
NRF24L01
  * @version            :     V 0.1

******************************************************************************
***
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
***
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------
---*/
#include "main.h"

/* Private includes ----------------------------------------------------------
---*/
/* USER CODE BEGIN Includes */

#include "ILI9488.h"
#include "keypad.h"
#include "bitmaps.h"
#include "NRF24L01.h"
#include "stdio.h"
#include <stdbool.h>
#include <string.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
---*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

```
/* Private define ------------------------------------------------------------
---*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
---*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
---*/
I2C_HandleTypeDef hi2c1;

SPI_HandleTypeDef hspi2;
SPI_HandleTypeDef hspi3;

TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim5;

/* USER CODE BEGIN PV */

uint8_t     RxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t     TxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};

uint8_t     device_channel[4] = {10, 20, 30, 40};
uint8_t     channel = 10;

uint8_t     device_id[4] = {10, 50, 100, 150};
uint8_t     peripheral_id = 10;
uint8_t     id_m = 100;

uint8_t           RxData[32];
uint8_t           TxData[32];
uint8_t     device_Rx[32];
uint8_t     device_Tx[32];
uint8_t     device_Rx_reg;
uint8_t     device_Tx_reg;



char        key_store;
char        disp[8];
char        id;
uint8_t     update_menu = 0;
uint8_t     time_set_menu = 10;
uint8_t           time_end_menu = 0;
uint8_t     update_check = 0;
uint8_t     time_set_check = 15;
uint8_t           time_end_check = 0;
uint8_t     wait_time_update = 0;
uint8_t     wait_time_set = 2;
uint8_t           wait_time_end = 0;
uint8_t     i = 0;
uint8_t     d = 0;
uint16_t    k = 0;
uint16_t    l = 0;
uint16_t    m = 0;
uint8_t     set_time_temp[7];
```

```
uint8_t           image_flag = 1;
uint8_t           wait_reg = 1;

uint8_t           temperature_temp = 0;
uint8_t           humidity_temp = 0;
uint8_t           sleep_light_temp = 0;
uint8_t           day_night_temp = 0;
uint8_t           on_off_temp = 0;
uint8_t           mov_time_temp = 1;
uint8_t     led_time_off = 1;
uint8_t     led_time_update = 0;
uint8_t     led_time_flag = 0;

uint8_t           hour_on = 8;
      //day hour starts
uint8_t           min_on = 0;
      //day min starts
uint16_t    day_time = 0;
uint8_t           hour_off = 23;
      //night hour starts
uint8_t           min_off = 0;
      //night min starts
uint16_t    night_time = 0;
uint16_t    current_time = 0;

uint8_t           temperature_1 = 0;
uint8_t           temperature_1_r = 0;
uint8_t           humidity_1 = 0;
uint8_t           humidity_1_r = 0;
uint8_t           sleep_light_1 = 0;
uint8_t           on_off_1 = 0;
uint8_t           mov_time_1 = 1;
uint8_t           movement_1 = 0;
uint8_t           heat_1 = 0;

uint8_t           temperature_2 = 0;
uint8_t           temperature_2_r = 0;
uint8_t           humidity_2 = 0;
uint8_t           humidity_2_r = 0;
uint8_t           sleep_light_2 = 0;
uint8_t           on_off_2 = 0;
uint8_t           mov_time_2 = 1;
uint8_t           movement_2 = 0;
uint8_t           heat_2 = 0;

uint8_t           temperature_3 = 0;
uint8_t           temperature_3_r = 0;
uint8_t           humidity_3 = 0;
uint8_t           humidity_3_r = 0;
uint8_t           sleep_light_3 = 0;
uint8_t           on_off_3 = 0;
uint8_t           mov_time_3 = 1;
uint8_t           movement_3 = 0;
uint8_t           heat_3 = 0;

uint8_t           temperature_4 = 0;
uint8_t           temperature_4_r = 0;
uint8_t           humidity_4 = 0;
uint8_t           humidity_4_r = 0;
uint8_t           sleep_light_4 = 0;
uint8_t           on_off_4 = 0;
```

```
uint8_t           mov_time_4 = 1;
uint8_t           movement_4 = 0;
uint8_t           heat_4 = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_SPI2_Init(void);
static void MX_SPI3_Init(void);
static void MX_TIM4_Init(void);
static void MX_TIM5_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------
---*/
/* USER CODE BEGIN 0 */

//-----------------------------------------TIMER INTERRUPT CHECK START-----
----------------------------------
//-------------------------------------------------------------------------
----------------------------------

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
//TIM5 update EVERY 1000ms (1 sec)
//if time_set_menu over exit menu
      if (htim->Instance == TIM5)
      {
            update_menu = update_menu + 1;
            if (update_menu > time_set_menu)
            {
                  time_end_menu = 1;
                  update_menu = 0;
                  HAL_TIM_Base_Stop_IT(&htim5);
            }
            else
            {
                  time_end_menu = 0;
            }
      }

//TIM4 update EVERY 1000ms (1 sec)
//if time_set_check is over then start checking every local peripheral
            if (htim->Instance == TIM4)
            {
                  update_check = update_check + 1;
                  if (update_check > time_set_check)
                  {
                        time_end_check = 1;
                        update_check = 0;
                  }
                  else
                  {
                        time_end_check = 0;
                  }
```

```
                led_time_update = led_time_update + 1;
                if (led_time_update >= led_time_off)
                {
                      led_time_update = 0;
                      led_time_flag = 1;
                }

                wait_time_update = wait_time_update + 1;
                if (wait_time_update >= wait_time_set)
                {
                      wait_time_update = 0;
                      wait_time_end = 1;
                }
          }

}

//-------------------------------------------------------------------------
---------------------------------
//-----------------------------------------TIMER INTERRUPT CHECK END-------
---------------------------------



//-------------------------------------------------RTC START----------------
---------------------------------
//-------------------------------------------------------------------------
---------------------------------


#define DS3231_ADDRESS 0xD0

// Convert normal decimal numbers to binary coded decimal
uint8_t decToBcd(int val)
{
  return (uint8_t)( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
int bcdToDec(uint8_t val)
{
  return (int)( (val/16*10) + (val%16) );
}

typedef struct {
      uint8_t seconds;
      uint8_t minutes;
      uint8_t hour;
      uint8_t dayofweek;
      uint8_t dayofmonth;
      uint8_t month;
      uint8_t year;
} TIME;

TIME time;

// function to set time

void Set_Time (uint8_t sec, uint8_t min, uint8_t hour, uint8_t dow, uint8_t
dom, uint8_t month, uint8_t year)
{
```

```
	uint8_t set_time[7];
	set_time[0] = decToBcd(sec);
	set_time[1] = decToBcd(min);
	set_time[2] = decToBcd(hour);
	set_time[3] = decToBcd(dow);
	set_time[4] = decToBcd(dom);
	set_time[5] = decToBcd(month);
	set_time[6] = decToBcd(year);

	HAL_I2C_Mem_Write(&hi2c1, DS3231_ADDRESS, 0x00, 1, set_time, 7,
1000);
}

void Get_Time (void)
{
	uint8_t get_time[7];
	HAL_I2C_Mem_Read(&hi2c1, DS3231_ADDRESS, 0x00, 1, get_time, 7, 1000);
	time.seconds = bcdToDec(get_time[0]);
	time.minutes = bcdToDec(get_time[1]);
	time.hour = bcdToDec(get_time[2]);
	time.dayofweek = bcdToDec(get_time[3]);
	time.dayofmonth = bcdToDec(get_time[4]);
	time.month = bcdToDec(get_time[5]);
	time.year = bcdToDec(get_time[6]);
}

char buffer[15];
char buffer1[15];

//-----------------------------------------------------------------------------
---------------------------------
//-------------------------------------------------RTC END-------------------
---------------------------------



void Background()
{
	if (image_flag == 0)
	{
		fill_rectangle(0, 0, WIDTH, HEIGHT, COLOR_BLACK);
		fill_rectangle(10, 40, 470, 43, COLOR_LIGHTBLUE);
		fill_rectangle(10, 290, 470, 293, COLOR_LIGHTBLUE);

		draw_bitmap(10, 45, 2, &home);
		draw_bitmap(84, 45, 2, &temperature);
		draw_bitmap(149, 45, 2, &humidity);
		draw_bitmap(214, 45, 2, &move);
		draw_bitmap(279, 45, 2, &light);
		draw_bitmap(344, 45, 2, &day);
		draw_bitmap(409, 45, 2, &night);
		draw_string(10, 300, COLOR_YELLOW, 1, "PRESS 'A' FOR MAIN MENU
OR 'B' FOR SAVED SETTINGS");

		draw_string(10, 120, COLOR_LIGHTBLUE, 1, "ROOM 1     ");
		draw_string(10, 160, COLOR_LIGHTBLUE, 1, "ROOM 2     ");
		draw_string(10, 200, COLOR_LIGHTBLUE, 1, "ROOM 3     ");
		draw_string(10, 240, COLOR_LIGHTBLUE, 1, "ROOM 4     ");

		image_flag = 1;
	}
```

```
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_SPI2_Init();
  MX_SPI3_Init();
  MX_TIM4_Init();
  MX_TIM5_Init();
  /* USER CODE BEGIN 2 */

  lcd_init_spi();
  NRF24_Init();

  HAL_TIM_Base_Stop_IT(&htim5);
  HAL_TIM_Base_Start_IT(&htim4);

//    Set_Time(00, 00, 15, 5, 15, 5, 24);
//      draw_string(10, 20, COLOR_LIGHTBLUE, 1, buffer1);

  fill_rectangle(0, 0, WIDTH, HEIGHT, COLOR_BLACK);
  fill_rectangle(10, 40, 470, 43, COLOR_LIGHTBLUE);
  fill_rectangle(10, 290, 470, 293, COLOR_LIGHTBLUE);

  draw_bitmap(10, 45, 2, &home);
  draw_bitmap(84, 45, 2, &temperature);
  draw_bitmap(149, 45, 2, &humidity);
  draw_bitmap(214, 45, 2, &move);
  draw_bitmap(279, 45, 2, &light);
  draw_bitmap(344, 45, 2, &day);
```

```
  draw_bitmap(409, 45, 2, &night);

  draw_string(10, 120, COLOR_LIGHTBLUE, 1, "ROOM 1      ");
  draw_string(10, 160, COLOR_LIGHTBLUE, 1, "ROOM 2      ");
  draw_string(10, 200, COLOR_LIGHTBLUE, 1, "ROOM 3      ");
  draw_string(10, 240, COLOR_LIGHTBLUE, 1, "ROOM 4      ");

  draw_string(10, 300, COLOR_YELLOW, 1, "PRESS 'A' FOR MAIN MENU OR 'B' FOR
SAVED SETTINGS");

  HAL_GPIO_WritePin(GPIOC, ERROR_LED_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOC, LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, Tx_LED_Pin, GPIO_PIN_SET);
  HAL_Delay(2000);
  HAL_GPIO_WritePin(GPIOC, ERROR_LED_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Tx_LED_Pin, GPIO_PIN_RESET);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        Start:


              if (led_time_flag == 1)
              {
                    HAL_GPIO_WritePin(GPIOC, ERROR_LED_Pin,GPIO_PIN_RESET);
                    HAL_GPIO_WritePin(GPIOC, LED_Pin, GPIO_PIN_RESET);
                    HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
                    HAL_GPIO_WritePin(GPIOA, Tx_LED_Pin, GPIO_PIN_RESET);
                    led_time_flag = 0;
              }

              HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
              key_store = Key();



        //-------------------------------------IF 'A' ENTER SETTINGS : IF
'B' DISPLAY SETTINGS----------------------

              if(key_store == 'A')
              {
        Main_menu:
                    image_flag = 0;
                    time_end_menu = 0;
                    HAL_TIM_Base_Start_IT(&htim5);
                    update_menu = 0;
                    fill_rectangle(0, 44, WIDTH, HEIGHT, COLOR_BLACK);
                    draw_string(10, 120, COLOR_CYAN, 1, "PRESS 'A' TO ENTER
PERIPHERAL DATA");
                    draw_string(10, 160, COLOR_CYAN, 1, "PRESS 'B' TO SET
TIME AND DATE");
                    draw_string(10, 200, COLOR_CYAN, 1, "PRESS 'D' TO EXID
OR JUST WAIT 20 seconds");
```

```
                    draw_fast_string(175, 20, COLOR_LIGHTBLUE, COLOR_BLACK,
"-----SET DATA---");
                    HAL_Delay(100);

        //                                      START INPUT
        //Loop1:
                    key_store = Key();
                    while (key_store == KEYPAD_NO_PRESSED)
                        {
                          key_store = Key();
                          if (time_end_menu == 1)
                          {
                                goto Time_out;
                          }
                        }
                    update_menu = 0;


                    if(key_store == 'D')
                    {
                          fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                          goto Time_out;
                    }



                    if(key_store == 'A')
                    {
        Sub_menu:
                          fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                          draw_string(10, 60, COLOR_CYAN, 1, "PRESS 'A' FOR
MAIN MENU");
                          draw_string(10, 80, COLOR_CYAN, 1, "PRESS '1' TO
SET ON-OFF LOCAL PERIPHERAL");
                          draw_string(10, 100, COLOR_CYAN, 1, "PRESS '2' TO
SET TEMPERATURE");
                          draw_string(10, 120, COLOR_CYAN, 1, "PRESS '3' TO
SET HUMIDITY");
                          draw_string(10, 140, COLOR_CYAN, 1, "PRESS '4' TO
SET DAY-NIGHT TIME");
                          draw_string(10, 160, COLOR_CYAN, 1, "PRESS '5' TO
SET NIGHT LIGHT ON-OFF");
                          draw_string(10, 180, COLOR_CYAN, 1, "PRESS '6' TO
SET MOVEMENT TIME");
                          draw_string(10, 200, COLOR_CYAN, 1, "PRESS 'B' TO
SELECT LOCAL PERIPHERAL AND SAVE");
                          draw_string(10, 220, COLOR_CYAN, 1, "PRESS 'D' TO
EXID OR JUST WAIT 20 seconds");

                          key_store = Key();
                          while (key_store == KEYPAD_NO_PRESSED)
                          {
                                key_store = Key();
                                if (time_end_menu == 1)
                                {
                                      goto Time_out;
                                }
                          }
                          update_menu = 0;
```

```
                        if(key_store == 'A')
                        {
                              goto Main_menu;
                        }

                        if(key_store == '1')
                        {
                              goto Menu_on_off;
                        }

                        if(key_store == '2')
                        {
                              goto Menu_temp;
                        }

                        if(key_store == '3')
                        {
                              goto Menu_hum;
                        }

                        if(key_store == '4')
                        {
                              goto Menu_day_night;
                        }

                        if(key_store == '5')
                        {
                              goto Menu_night_light;
                        }

                        if(key_store == '6')
                        {
                              goto Menu_mov;
                        }

                        if(key_store == 'B')
                        {
                              goto Menu_save;
                        }

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store == 'C') || (key_store == '7') ||
(key_store == '8') || (key_store == '9') || (key_store == '0'))
                        {
                              goto Sub_menu;
                        }


        //-----------------------------------------------TEMPERATURE SET---
----------------------------------------

        Menu_temp:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
```

```
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED TEMPERATURE 10-30");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED
TEMPERATURE = ");

        Loop2:          key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop2;
                        }

                        l = key_store -48;

                        draw_string(208, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

        Loop3:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }
```

```
                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop3;
                        }

                        k = key_store -48;

                        draw_string(216, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        temperature_temp = (l * 10) + k;
                        if (temperature_temp > 30)
                        {
                              temperature_temp = 30;
                        }
                        if (temperature_temp < 10)
                        {
                              temperature_temp = 10;
                        }
                        sprintf (disp, "%02i", temperature_temp);
                        draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


        //-----------------------------------------------HUMIDITY SET------
----------------------------------------

        Menu_hum:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED HUMIDITY 35-80");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED
HUMIDITY = ");
```

```
Loop4:              key_store = Key();
                    while (key_store == KEYPAD_NO_PRESSED)
                    {
                          key_store = Key();
                          if (time_end_menu == 1)
                          {
                                goto Time_out;
                          }
                    }
                    update_menu = 0;

                    if(key_store == 'D')
                    {
                          fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                          goto Time_out;
                    }

                    if ((key_store >=58) || (key_store <= 47))
                    {
                          goto Loop4;
                    }

                    l = key_store -48;

                    draw_string(185, 160, COLOR_CYAN, 1, &key_store);

                    key_store = Key();
                    while (key_store != KEYPAD_NO_PRESSED)
                    {
                          key_store = Key();
                          if (time_end_menu == 1)
                          {
                                goto Time_out;
                          }
                    }
                    update_menu = 0;

Loop5:
                    key_store = Key();
                    while (key_store == KEYPAD_NO_PRESSED)
                    {
                          key_store = Key();
                          if (time_end_menu == 1)
                          {
                                goto Time_out;
                          }
                    }
                    update_menu = 0;

                    if(key_store == 'D')
                    {
                          fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                          goto Time_out;
                    }

                    if ((key_store >=58) || (key_store <= 47))
                    {
                          goto Loop5;
                    }
```

```
                        k = key_store -48;

                        draw_string(193, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        humidity_temp = (l * 10) + k;
                        if (humidity_temp > 80)
                        {
                              humidity_temp = 80;
                        }
                        if (humidity_temp < 35)
                        {
                              humidity_temp = 35;
                        }
                        sprintf (disp, "%02i", humidity_temp);
                        draw_fast_string(185, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


        //-----------------------------------------------NIGHT HOUR SET----
-----------------------------------------


        Menu_day_night:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED HOUR FOR NIGHT 00-23");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED
NIGHT HOUR = ");

        Loop6:          key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
```

```
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;

                  if(key_store == 'D')
                  {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                  }

                  if ((key_store >=58) || (key_store <= 47))
                  {
                        goto Loop6;
                  }

                  l = key_store -48;

                  draw_string(208, 160, COLOR_CYAN, 1, &key_store);

                  key_store = Key();
                  while (key_store != KEYPAD_NO_PRESSED)
                  {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;

            Loop7:
                  key_store = Key();
                  while (key_store == KEYPAD_NO_PRESSED)
                  {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;

                  if(key_store == 'D')
                  {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                  }

                  if ((key_store >=58) || (key_store <= 47))
                  {
                        goto Loop7;
                  }

                  k = key_store -48;
                        //convert to dec

                  draw_string(216, 160, COLOR_CYAN, 1, &key_store);
```

```
                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        hour_off = (l * 10) + k;
                        if (hour_off > 23)
                        {
                              hour_off = 23;
                        }

                        sprintf (disp, "%02i", hour_off);
                        draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------NIGHT MINUTE SET--
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED MINUTE FOR NIGHT 00-60");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED
NIGHT MIN = ");

        Loop8:          key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
```

```
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                    }

                    if ((key_store >=58) || (key_store <= 47))
                    {
                        goto Loop8;
                    }

                    l = key_store -48;
                              //convert to dec

                    draw_string(208, 160, COLOR_CYAN, 1, &key_store);

                    key_store = Key();
                    while (key_store != KEYPAD_NO_PRESSED)
                    {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                            goto Time_out;
                        }
                    }
                    update_menu = 0;

        Loop9:
                    key_store = Key();
                    while (key_store == KEYPAD_NO_PRESSED)
                    {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                            goto Time_out;
                        }
                    }
                    update_menu = 0;

                    if(key_store == 'D')
                    {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                    }

                    if ((key_store >=58) || (key_store <= 47))
                    {
                        goto Loop9;
                    }

                    k = key_store -48;
                              //convert to dec

                    draw_string(216, 160, COLOR_CYAN, 1, &key_store);

                    key_store = Key();
                    while (key_store != KEYPAD_NO_PRESSED)
                    {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
```

```
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        min_off = (l * 10) + k;
                        if (min_off > 60)
                        {
                              min_off = 60;
                        }

                        sprintf (disp, "%02i", min_off);
                        draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        sprintf (disp, "%02d:%02d", hour_off, min_off);
                        draw_string(10, 200, COLOR_CYAN, 1, "DESIRED
NIGHT TIME = ");
                        draw_string(208, 200, COLOR_CYAN, 1, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------DAY HOUR SET------
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED HOUR FOR DAY 00-23");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED DAY
HOUR = ");

        Loop10:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;



                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
```

```
                        goto Time_out;
                }

                if ((key_store >=58) || (key_store <= 47))
                {
                        goto Loop10;
                }

                l = key_store -48;
                        //convert to dec

                draw_string(208, 160, COLOR_CYAN, 1, &key_store);

                key_store = Key();
                while (key_store != KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
                }
                update_menu = 0;

        Loop11:
                key_store = Key();
                while (key_store == KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
                }
                update_menu = 0;

                if(key_store == 'D')
                {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                }

                if ((key_store >=58) || (key_store <= 47))
                {
                        goto Loop11;
                }

                k = key_store -48;
                        //convert to dec

                draw_string(216, 160, COLOR_CYAN, 1, &key_store);

                key_store = Key();
                while (key_store != KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
```

```
                        }
                        update_menu = 0;

                        hour_on = (l * 10) + k;
                        if (hour_on > 23)
                        {
                              hour_on = 23;
                        }

                        sprintf (disp, "%02i", hour_on);
                        draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------DAY MINUTE SET----
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DESIRED MINUTE FOR DAY 00-60");
                        draw_string(10, 160, COLOR_CYAN, 1, "DESIRED DAY
MIN = ");

        Loop12:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop12;
                        }
```

```
                        l = key_store -48;
                                //convert to dec

                        draw_string(208, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

      Loop13:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop13;
                        }

                        k = key_store -48;
                                //convert to dec

                        draw_string(216, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        min_on = (l * 10) + k;
                        if (min_on > 60)
                        {
                              min_on = 60;
                        }
```

```
                        sprintf (disp, "%02i", min_on);
                        draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        sprintf (disp, "%02d:%02d", hour_on, min_on);
                        draw_string(10, 200, COLOR_CYAN, 1, "DESIRED DAY
TIME = ");
                        draw_string(208, 200, COLOR_CYAN, 1, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


        //-----------------------------------------------NIGHT LIGHT
OPTION-----------------------------------------

        Menu_night_light:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE
SELECT NIGHT LIGHT OPTION");
                        draw_string(10, 160, COLOR_CYAN, 1, "'1 = NIGHT
LIGHT ON'    '0 = NIGHT LIGHT OFF'");
                        draw_string(10, 200, COLOR_CYAN, 1, "NIGHT LIGHT
OPTION = ");

        Loop14:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >49) || (key_store < 48))
                        {
                              goto Loop14;
                        }
```

```
                        k = key_store -48;
                                //convert to dec

                        draw_string(197, 200, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        sleep_light_temp = k;


        //                sprintf (disp, "%01i", sleep_light_temp);
        //                draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


        //-------------------------------------------------PERIPHERAL ON
OFF---------------------------------------

        Menu_on_off:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE SET
PERIPHERAL ON OR OFF");
                        draw_string(10, 160, COLOR_CYAN, 1, "'1 =
PERIPHERAL ON'   '0 = PERIPHERAL OFF'");
                        draw_string(10, 200, COLOR_CYAN, 1, "PERIPHERAL
OPTION = ");

        Loop15:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
```

```
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >49) || (key_store < 48))
                        {
                              goto Loop15;
                        }

                        k = key_store -48;
                                    //convert to dec

                        draw_string(197, 200, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        on_off_temp = k;


        //                sprintf (disp, "%01i", sleep_light_temp);
        //                draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


        //-----------------------------------------------MOVEMENT TIME SET-
----------------------------------------

        Menu_mov:
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
```

```
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE
SELECT MOVEMENT TIME IN MINUTES 1-5");
                        draw_string(10, 160, COLOR_CYAN, 1, "MOVEMENT
TIME = ");


        Loop26:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >53) || (key_store < 49))
                        {
                              goto Loop26;
                        }

                        k = key_store -48;
                                  //convert to dec

                        draw_string(154, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        mov_time_temp = k;


        //              sprintf (disp, "%01i", sleep_light_temp);
        //              draw_fast_string(208, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
```

```
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;
                  goto Sub_menu;


      //-----------------------------------LOCAL PERIPHERAL SELECT AND
SAVE SETTINGS-----------------------------

      Menu_save:
                  fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                  draw_string(10, 120, COLOR_CYAN, 1, "PLEASE
SELECT LOCAL PERIPHERAL 1-4");
                  draw_string(10, 160, COLOR_CYAN, 1, "LOCAL
PERIPHERAL = ");


      Loop27:
                  key_store = Key();
                  while (key_store == KEYPAD_NO_PRESSED)
                  {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;

                  if(key_store == 'D')
                  {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                  }

                  if ((key_store >52) || (key_store < 49))
                  {
                        goto Loop27;
                  }

                  id = key_store;
                  k = key_store -48;
                        //convert to dec

                  draw_string(181, 160, COLOR_CYAN, 1, &key_store);

                  key_store = Key();
                  while (key_store != KEYPAD_NO_PRESSED)
                  {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                              goto Time_out;
                        }
                  }
                  update_menu = 0;
```

```
                        if (id == '1')
                        {
                              temperature_1 = temperature_temp;
                              humidity_1 = humidity_temp;
                              sleep_light_1 = sleep_light_temp;
                              on_off_1 = on_off_temp;
                              mov_time_1 = mov_time_temp;
                        }

                        if (id == '2')
                        {
                              temperature_2 = temperature_temp;
                              humidity_2 = humidity_temp;
                              sleep_light_2 = sleep_light_temp;
                              on_off_2 = on_off_temp;
                              mov_time_2 = mov_time_temp;
                        }

                        if (id == '3')
                        {
                              temperature_3 = temperature_temp;
                              humidity_3 = humidity_temp;
                              sleep_light_3 = sleep_light_temp;
                              on_off_3 = on_off_temp;
                              mov_time_3 = mov_time_temp;
                        }

                        if (id == '4')
                        {
                              temperature_4 = temperature_temp;
                              humidity_4 = humidity_temp;
                              sleep_light_4 = sleep_light_temp;
                              on_off_4 = on_off_temp;
                              mov_time_4 = mov_time_temp;
                        }

                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO SAVE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        goto Sub_menu;


                  }



      //-----------------------------------------------TIME DATE SET-----
----------------------------------------
      //-----------------------------------------------------------------
----------------------------------------
```

```
                    if(key_store == 'B')
                                  //input date and clock data
                    {

                          fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                          draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
MINUTES : 00-60");
                          draw_string(10, 160, COLOR_CYAN, 1, "MINUTES =
");

        //-----------------------------------------------MINUTE SET--------
----------------------------------------

        Loop16:           key_store = Key();
                          while (key_store == KEYPAD_NO_PRESSED)
                          {
                                key_store = Key();
                                if (time_end_menu == 1)
                                {
                                      goto Time_out;
                                }
                          }
                          update_menu = 0;

                          if(key_store == 'D')
                          {
                                fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                                goto Time_out;
                          }

                          if ((key_store >=58) || (key_store <= 47))
                          {
                                goto Loop16;
                          }

                          l = key_store -48;
                                    //convert to dec

                          draw_string(103, 160, COLOR_CYAN, 1, &key_store);

                          key_store = Key();
                          while (key_store != KEYPAD_NO_PRESSED)
                          {
                                key_store = Key();
                                if (time_end_menu == 1)
                                {
                                      goto Time_out;
                                }
                          }
                          update_menu = 0;

        Loop17:
                          key_store = Key();
                          while (key_store == KEYPAD_NO_PRESSED)
                          {
                                key_store = Key();
                                if (time_end_menu == 1)
                                {
```

```
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop17;
                        }

                        k = key_store -48;
                                    //convert to dec

                        draw_string(111, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        set_time_temp[1] = (l * 10) + k;
                        if (set_time_temp[1] > 60)
                        {
                              set_time_temp[1] = 60;
                        }

                        sprintf (disp, "%02i", set_time_temp[1]);
                        draw_fast_string(103, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------HOUR SET----------
----------------------------------------
```

```
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
HOUR : 00-23");
                        draw_string(10, 160, COLOR_CYAN, 1, "HOUR = ");

        Loop18:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop18;
                        }

                        l = key_store -48;
                                  //convert to dec

                        draw_string(77, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

        Loop19:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
```

```
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop19;
                        }

                        k = key_store -48;
                                    //convert to dec

                        draw_string(85, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        set_time_temp[2] = (l * 10) + k;
                        if (set_time_temp[2] > 23)
                        {
                              set_time_temp[2] = 23;
                        }

                        sprintf (disp, "%02i", set_time_temp[2]);
                        draw_fast_string(77, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------DAY OF MONTH SET--
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
DAY OF MONTH : 00-31");
                        draw_string(10, 160, COLOR_CYAN, 1, "DAY OF MONTH
= ");

        Loop20:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
```

```
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop20;
                        }

                        l = key_store -48;
                                  //convert to dec

                        draw_string(145, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                Loop21:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop21;
                        }
```

```
                        k = key_store -48;
                                //convert to dec

                        draw_string(153, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        set_time_temp[4] = (l * 10) + k;
                        if (set_time_temp[4] > 31)
                        {
                              set_time_temp[4] = 31;
                        }

                        sprintf (disp, "%02i", set_time_temp[4]);
                        draw_fast_string(145, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------MONTH SET---------
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
MONTH : 00-12");
                        draw_string(10, 160, COLOR_CYAN, 1, "MONTH = ");

        Loop22:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
```

```
                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop22;
                        }

                        l = key_store -48;
                                    //convert to dec

                        draw_string(82, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

      Loop23:
                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                              goto Time_out;
                        }

                        if ((key_store >=58) || (key_store <= 47))
                        {
                              goto Loop23;
                        }

                        k = key_store -48;
                                    //convert to dec

                        draw_string(90, 160, COLOR_CYAN, 1, &key_store);

                        key_store = Key();
                        while (key_store != KEYPAD_NO_PRESSED)
```

```
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;

                        set_time_temp[5] = (l * 10) + k;
                        if (set_time_temp[5] > 12)
                        {
                              set_time_temp[5] = 12;
                        }

                        sprintf (disp, "%02i", set_time_temp[5]);
                        draw_fast_string(82, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;


        //-----------------------------------------------YEAR SET----------
----------------------------------------

                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        draw_string(10, 120, COLOR_CYAN, 1, "PLEASE ENTER
YEAR : 00-34");
                        draw_string(10, 160, COLOR_CYAN, 1, "YEAR = ");

        Loop24:         key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;



                        if(key_store == 'D')
                        {
                              fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
```

```
                        goto Time_out;
                }

                if ((key_store >=58) || (key_store <= 47))
                {
                        goto Loop24;
                }

                l = key_store -48;
                        //convert to dec

                draw_string(73, 160, COLOR_CYAN, 1, &key_store);

                key_store = Key();
                while (key_store != KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
                }
                update_menu = 0;

        Loop25:
                key_store = Key();
                while (key_store == KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
                }
                update_menu = 0;

                if(key_store == 'D')
                {
                        fill_rectangle(0, 44, WIDTH, HEIGHT,
COLOR_BLACK);
                        goto Time_out;
                }

                if ((key_store >=58) || (key_store <= 47))
                {
                        goto Loop25;
                }

                k = key_store -48;
                        //convert to dec

                draw_string(81, 160, COLOR_CYAN, 1, &key_store);

                key_store = Key();
                while (key_store != KEYPAD_NO_PRESSED)
                {
                        key_store = Key();
                        if (time_end_menu == 1)
                        {
                                goto Time_out;
                        }
```

```
                        }
                        update_menu = 0;

                        set_time_temp[6] = (l * 10) + k;
                        if (set_time_temp[6] > 34)
                        {
                              set_time_temp[6] = 34;
                        }

                        sprintf (disp, "%02i", set_time_temp[6]);
                        draw_fast_string(73, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                        draw_fast_string(10, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        Set_Time(00, set_time_temp[1], set_time_temp[2],
5, set_time_temp[4], set_time_temp[5], set_time_temp[6]);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                  }



            }


      //-----------------------------------------------SAVED SETTINGS
DISPLAY-------------------------------------
      //-------------------------------------------------------------------------
----------------------------------------

            if (key_store == 'B')
            {
                  time_end_menu = 0;
                  HAL_TIM_Base_Start_IT(&htim5);
                  update_menu = 0;

                  draw_fast_string(175, 20, COLOR_LIGHTBLUE, COLOR_BLACK,
"---SAVED DATA----");
                  fill_rectangle(100, 120, 470, 250, COLOR_BLACK);

                  sprintf (disp, "%02i", temperature_1);
                  draw_fast_string(110, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                  sprintf (disp, "%02i", humidity_1);
                  draw_fast_string(173, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                  if (movement_1 == 0)
                  {
                        draw_fast_string(238, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
```

```
                    }
                    else
                    {
                          draw_fast_string(238, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_1 == 0)
                    {
                          draw_fast_string(301, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(301, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (day_night_temp == 0)
                    {
                          draw_fast_string(354, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                          draw_fast_string(354, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_1 == 0)
                    {
                          draw_fast_string(427, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                          draw_fast_string(427, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



                    sprintf (disp, "%02i", temperature_2);
                    draw_fast_string(110, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    sprintf (disp, "%02i", humidity_2);
                    draw_fast_string(173, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    if (movement_2 == 0)
                    {
                          draw_fast_string(238, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(238, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_2 == 0)
                    {
```

```
                        draw_fast_string(301, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                        draw_fast_string(301, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (day_night_temp == 0)
                    {
                        draw_fast_string(354, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                        draw_fast_string(354, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_2 == 0)
                    {
                        draw_fast_string(427, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                        draw_fast_string(427, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



                    sprintf (disp, "%02i", temperature_3);
                    draw_fast_string(110, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    sprintf (disp, "%02i", humidity_3);
                    draw_fast_string(173, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    if (movement_3 == 0)
                    {
                        draw_fast_string(238, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                        draw_fast_string(238, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_3 == 0)
                    {
                        draw_fast_string(301, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                        draw_fast_string(301, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }
```

```
                    if (day_night_temp == 0)
                    {
                          draw_fast_string(354, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                          draw_fast_string(354, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_3 == 0)
                    {
                          draw_fast_string(427, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                          draw_fast_string(427, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



                    sprintf (disp, "%02i", temperature_4);
                    draw_fast_string(110, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    sprintf (disp, "%02i", humidity_4);
                    draw_fast_string(173, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    if (movement_4 == 0)
                    {
                          draw_fast_string(238, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(238, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_4 == 0)
                    {
                          draw_fast_string(301, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(301, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (day_night_temp == 0)
                    {
                          draw_fast_string(354, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                          draw_fast_string(354, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
```

```
                        }

                        if (heat_4 == 0)
                        {
                              draw_fast_string(427, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                        }
                        else
                        {
                              draw_fast_string(427, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                        }


                        draw_fast_string(10, 270, COLOR_LIGHTBLUE, COLOR_BLACK,
"PRESS ANY KEY TO CONTINUE");
                        HAL_Delay(200);

                        key_store = Key();
                        while (key_store == KEYPAD_NO_PRESSED)
                        {
                              key_store = Key();
                              if (time_end_menu == 1)
                              {
                                    fill_rectangle(8, 260, 470, 280,
COLOR_BLACK);
                                    goto Time_out;
                              }
                        }
                        update_menu = 0;
                        fill_rectangle(8, 260, 470, 280, COLOR_BLACK);
                        goto Start;
                  }

            //-----------------------------------------------END OF SETTINGS---
----------------------------------------
            //-----------------------------------------------------------------
----------------------------------------


            //-----------------------------------------------CHECK DAY NIGHT---
----------------------------------------
            //-----------------------------------------------------------------
----------------------------------------

                  Get_Time();
                  night_time = ((hour_off * 60) + min_off);
                  day_time = ((hour_on * 60) + min_on);
                  current_time = (((time.hour)*60) + (time.minutes));
                  if (night_time > day_time)
                  {
                        if ((night_time > current_time) && (day_time <
current_time))
                        {
                              day_night_temp = 1;
                        }
                        else
                        {
                              day_night_temp = 0;
                        }
                  }
```

```
		if (night_time < day_time)
		{
			if ((night_time < current_time) && (day_time >
current_time))
			{
				day_night_temp = 0;
			}
			else
			{
				day_night_temp = 1;
			}
		}


	//-----------------------------------------------END OF CHECK------
----------------------------------------
	//-----------------------------------------------------------------
----------------------------------------


	//-----------------------------------------------DISPLAY LIVE DATA-
----------------------------------------
	//-----------------------------------------------------------------
----------------------------------------

	Time_out:
			Get_Time();
			sprintf (buffer, "%02d:%02d", time.hour, time.minutes);
			sprintf (buffer1, "%02d-%02d-20%02d", time.dayofmonth,
time.month, time.year);
			draw_fast_string(420, 20, COLOR_LIGHTBLUE, COLOR_BLACK,
buffer);
			draw_fast_string(10, 20, COLOR_LIGHTBLUE, COLOR_BLACK,
buffer1);

			Background();
			draw_fast_string(175, 20, COLOR_LIGHTBLUE, COLOR_BLACK,
"----LIVE DATA----");


			sprintf (disp, "%02i", temperature_1_r);
			draw_fast_string(126, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, " C");
			draw_fast_string(110, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
			sprintf (disp, "%02i", humidity_1_r);
			draw_fast_string(189, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, " %");
			draw_fast_string(173, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
			if (movement_1 == 0)
			{
				draw_fast_string(238, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
			}
			else
			{
				draw_fast_string(238, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
```

```
                    }

                    if (sleep_light_1 == 0)
                    {
                          draw_fast_string(301, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(301, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (day_night_temp == 0)
                    {
                          draw_fast_string(354, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                          draw_fast_string(354, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_1 == 0)
                    {
                          draw_fast_string(427, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                          draw_fast_string(427, 120, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



                    sprintf (disp, "%02i", temperature_2_r);
                    draw_fast_string(126, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, " C");
                    draw_fast_string(110, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    sprintf (disp, "%02i", humidity_2_r);
                    draw_fast_string(189, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, " %");
                    draw_fast_string(173, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    if (movement_2 == 0)
                    {
                          draw_fast_string(238, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                          draw_fast_string(238, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_2 == 0)
                    {
```

```
                        draw_fast_string(301, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                        draw_fast_string(301, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (day_night_temp == 0)
                    {
                        draw_fast_string(354, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                        draw_fast_string(354, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_2 == 0)
                    {
                        draw_fast_string(427, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                        draw_fast_string(427, 160, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



                    sprintf (disp, "%02i", temperature_3_r);
                    draw_fast_string(126, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, " C");
                    draw_fast_string(110, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    sprintf (disp, "%02i", humidity_3_r);
                    draw_fast_string(189, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, " %");
                    draw_fast_string(173, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                    if (movement_3 == 0)
                    {
                        draw_fast_string(238, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
                        draw_fast_string(238, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                    }

                    if (sleep_light_3 == 0)
                    {
                        draw_fast_string(301, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                    }
                    else
                    {
```

```
                        draw_fast_string(301, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                  }

                  if (day_night_temp == 0)
                  {
                        draw_fast_string(354, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                  }
                  else
                  {
                        draw_fast_string(354, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                  }

                  if (heat_3 == 0)
                  {
                        draw_fast_string(427, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                  }
                  else
                  {
                        draw_fast_string(427, 200, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                  }



                  sprintf (disp, "%02i", temperature_4_r);
                  draw_fast_string(126, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, " C");
                  draw_fast_string(110, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                  sprintf (disp, "%02i", humidity_4_r);
                  draw_fast_string(189, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, " %");
                  draw_fast_string(173, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, disp);
                  if (movement_4 == 0)
                  {
                        draw_fast_string(238, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                  }
                  else
                  {
                        draw_fast_string(238, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                  }

                  if (sleep_light_4 == 0)
                  {
                        draw_fast_string(301, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NO ");
                  }
                  else
                  {
                        draw_fast_string(301, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "YES");
                  }

                  if (day_night_temp == 0)
```

```
                    {
                          draw_fast_string(354, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "NIGHT");
                    }
                    else
                    {
                          draw_fast_string(354, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, " DAY ");
                    }

                    if (heat_4 == 0)
                    {
                          draw_fast_string(427, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "OFF");
                    }
                    else
                    {
                          draw_fast_string(427, 240, COLOR_LIGHTBLUE,
COLOR_BLACK, "ON ");
                    }



        //-------------------------------------START COMMUNICATION WITH
LOCAL PERIPHERALS----------------------------
        //-------------------------------------------------------------------
----------------------------------------

                    if (time_end_check == 1)
                    {
                          time_end_check = 0;

                          for(i=0; i<4; i++)
                          {
                                for (m = 0; m < 32; m ++)
                                {
                                      TxData[m] = 0;
                                }

                                TxData[7] = day_night_temp;
                                channel = device_channel[0];
                                peripheral_id = device_id[i];
                                TxData[0] = peripheral_id;

                                if (i == 0)
                                {
                                      TxData[5] = temperature_1;
                                      TxData[6] = humidity_1;
                                      TxData[8] = sleep_light_1;
                                      TxData[1] = on_off_1;
                                      TxData[12] = mov_time_1;
                                }

                                if (i == 1)
                                {
                                      TxData[5] = temperature_2;
                                      TxData[6] = humidity_2;
                                      TxData[8] = sleep_light_2;
                                      TxData[1] = on_off_2;
                                      TxData[12] = mov_time_2;
                                }
```

```
                                    if (i == 2)
                                    {
                                          TxData[5] = temperature_3;
                                          TxData[6] = humidity_3;
                                          TxData[8] = sleep_light_3;
                                          TxData[1] = on_off_3;
                                          TxData[12] = mov_time_3;
                                    }

                                    if (i == 3)
                                    {
                                          TxData[5] = temperature_4;
                                          TxData[6] = humidity_4;
                                          TxData[8] = sleep_light_4;
                                          TxData[1] = on_off_4;
                                          TxData[12] = mov_time_4;
                                    }

                                    for (m = 0; m < 32; m ++)
                                    {
                                          RxData[m] = 0;
                                    }
        //START TRANSMITTING DATA TO PERIPHERAL UNITS

                                          NRF24_TxMode(TxAddress, channel);

      //Go to TX mode
                                          if (NRF24_Transmit(TxData) == 1)

      //If data transmitted
                                          {
                                                NRF24_RxMode(RxAddress,
channel);
      //Go to RX mode
                                                HAL_GPIO_WritePin(GPIOA,
Tx_LED_Pin, GPIO_PIN_SET);
                                                led_time_update = 0;

      //Transmit LED ON

                                                HAL_Delay(20);
                                          }

                                    NRF24_RxMode(RxAddress, channel);
                                                                        //Go to
RX mode
                                    wait_reg = RxData[13];
                                    wait_time_update = 0;
                                    wait_time_end = 0;
                                    while (wait_reg == 0)
                                    {
                                      HAL_Delay(20);
                                      if (isDataAvailable(1) == 1)
                                      {
                                        NRF24_Receive(RxData);
                                        HAL_GPIO_WritePin(GPIOA,
Rx_LED_Pin, GPIO_PIN_SET);
                                        led_time_update = 0;

                                        if (RxData[0] == peripheral_id)
```

```
                                        {
                                              HAL_GPIO_TogglePin(GPIOC,
LED_Pin);
                                              led_time_update = 0;
                                              wait_reg = RxData[13];
                                        }
                                      }
                                      if (wait_time_end == 1)
                                      {
                                        wait_time_end = 0;
                                      goto    Next;
                                      }
                                }


                                if (i == 0)
                                {
                                      temperature_1_r = RxData[3];
                                      humidity_1_r = RxData[4];
                                      movement_1 = RxData[2];
                                      heat_1 = RxData[9];
                                }

                                if (i == 1)
                                {
                                      temperature_2_r = RxData[3];
                                      humidity_2_r = RxData[4];
                                      movement_2 = RxData[2];
                                      heat_2 = RxData[9];
                                }

                                if (i == 2)
                                {
                                      temperature_3_r = RxData[3];
                                      humidity_3_r = RxData[4];
                                      movement_3 = RxData[2];
                                      heat_3 = RxData[9];
                                }

                                if (i == 3)
                                {
                                      temperature_4_r = RxData[3];
                                      humidity_4_r = RxData[4];
                                      movement_4 = RxData[2];
                                      heat_4 = RxData[9];
                                }
Next:
                                HAL_Delay(20);
                          }


                    }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
```

```
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 84;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
```

```
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief SPI2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI2_Init(void)
{

  /* USER CODE BEGIN SPI2_Init 0 */

  /* USER CODE END SPI2_Init 0 */

  /* USER CODE BEGIN SPI2_Init 1 */

  /* USER CODE END SPI2_Init 1 */
  /* SPI2 parameter configuration*/
  hspi2.Instance = SPI2;
  hspi2.Init.Mode = SPI_MODE_MASTER;
  hspi2.Init.Direction = SPI_DIRECTION_2LINES;
  hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi2.Init.NSS = SPI_NSS_SOFT;
  hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
  hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi2.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI2_Init 2 */

  /* USER CODE END SPI2_Init 2 */

}

/**
  * @brief SPI3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI3_Init(void)
```

```
{

  /* USER CODE BEGIN SPI3_Init 0 */

  /* USER CODE END SPI3_Init 0 */

  /* USER CODE BEGIN SPI3_Init 1 */

  /* USER CODE END SPI3_Init 1 */
  /* SPI3 parameter configuration*/
  hspi3.Instance = SPI3;
  hspi3.Init.Mode = SPI_MODE_MASTER;
  hspi3.Init.Direction = SPI_DIRECTION_2LINES;
  hspi3.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi3.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi3.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi3.Init.NSS = SPI_NSS_SOFT;
  hspi3.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi3.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi3.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi3.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi3.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI3_Init 2 */

  /* USER CODE END SPI3_Init 2 */

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 8400-1;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 10000-1;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
```

```
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */

}

/**
  * @brief TIM5 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM5_Init(void)
{

  /* USER CODE BEGIN TIM5_Init 0 */

  /* USER CODE END TIM5_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM5_Init 1 */

  /* USER CODE END TIM5_Init 1 */
  htim5.Instance = TIM5;
  htim5.Init.Prescaler = 8400-1;
  htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim5.Init.Period = 10000-1;
  htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM5_Init 2 */

  /* USER CODE END TIM5_Init 2 */
```

```
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, HEAT_1_Pin|HEAT_2_Pin|HEAT_3_Pin|HEAT_4_Pin
                          |ERROR_LED_Pin|LED_Pin|WR_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, CE_Pin|CSN_Pin|Rx_LED_Pin|Tx_LED_Pin
                          |COL_1_Pin|COL_2_Pin|COL_3_Pin|COL_4_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, CS_Pin|RESX_Pin|DC_Pin|RD_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : HEAT_1_Pin HEAT_2_Pin HEAT_3_Pin HEAT_4_Pin
                           ERROR_LED_Pin LED_Pin WR_Pin */
  GPIO_InitStruct.Pin = HEAT_1_Pin|HEAT_2_Pin|HEAT_3_Pin|HEAT_4_Pin
                          |ERROR_LED_Pin|LED_Pin|WR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : CE_Pin CSN_Pin Rx_LED_Pin Tx_LED_Pin
                           COL_1_Pin COL_2_Pin COL_3_Pin COL_4_Pin */
  GPIO_InitStruct.Pin = CE_Pin|CSN_Pin|Rx_LED_Pin|Tx_LED_Pin
                          |COL_1_Pin|COL_2_Pin|COL_3_Pin|COL_4_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : CS_Pin RESX_Pin DC_Pin RD_Pin */
  GPIO_InitStruct.Pin = CS_Pin|RESX_Pin|DC_Pin|RD_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : ROW_1_Pin ROW_2_Pin ROW_3_Pin */
  GPIO_InitStruct.Pin = ROW_1_Pin|ROW_2_Pin|ROW_3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
  /*Configure GPIO pin : ROW_4_Pin */
  GPIO_InitStruct.Pin = ROW_4_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(ROW_4_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

ΠΑΡΑΡΤΗΜΑ Β : Κώδικας STM32F030C8T6 Τοπικού Ελεγκτή

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
***
  * @project name :
      Receiver_Transmitter_Local_Control_STM32F030C8T6_New_Test
  * @author             :     VPK Engineering
  * @file           :   main.c
  * @brief          :   Local Control Unit with Motion and Temperature-
Humidity Sensor
  * @version            :     V 0.1

******************************************************************************
***
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
***
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------
---*/
#include "main.h"

/* Private includes ----------------------------------------------------------
---*/
/* USER CODE BEGIN Includes */

#include "NRF24L01.h"
#include "stdio.h"
#include "sht3x.h"
#include <stdbool.h>
#include <string.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
---*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

```
/* Private define ------------------------------------------------------------
---*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
---*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
---*/
I2C_HandleTypeDef hi2c1;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim6;

/* USER CODE BEGIN PV */

uint8_t     RxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t     TxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};

//uint8_t   device_channel[5] = {20, 21, 22, 23, 24};
uint8_t     device_channel[5] = {10, 11, 12, 13, 14};
//uint8_t   channel = 20;
uint8_t     channel = 10;

//uint8_t   device_id[5] = {50, 55, 60, 65, 70};
uint8_t     device_id[5] = {10, 15, 20, 25, 30};
//uint8_t   id = 50;
uint8_t     id = 10;

uint8_t           RxData[32];
uint8_t           TxData[32];
uint8_t     device_Rx[32];
uint8_t     device_Tx[32];
uint8_t     device_Rx_reg;
uint8_t     device_Tx_reg;

uint16_t    temperature_raw = 0;
uint16_t    humidity_raw = 0;
float       temperature;
float             humidity;
uint8_t           temperature_int;
uint8_t           humidity_int;
uint8_t           temperature_set;
uint8_t           humidity_set;
uint8_t     light = 0;
uint8_t     dimmer = 0;
uint8_t     ext_switch = 0;
uint8_t     day_night = 0;
uint8_t     sleep = 0;
uint8_t     movement = 0;
uint8_t     On_Off = 0;

uint8_t     peripheral_id = 0;
uint8_t     light_reg = 0;
uint8_t     dimmer_reg = 0;
```

```
uint8_t     ext_switch_reg = 0;
uint8_t     day_night_reg = 0;
uint8_t     movement_reg = 0;
uint8_t           wait_reg = 0;
uint8_t           temper_time_reg = 0;

uint8_t     i = 0;
uint8_t     d = 0;
uint8_t     k = 0;
uint8_t     reg = 0;

uint8_t     TXB_reg = 0;
uint8_t     timer = 0;
uint16_t    update = 0;
uint16_t    time_set = 120;

uint8_t     time_update = 0;
uint8_t     time_end = 20;
uint8_t     wait_time_update = 0;
uint8_t     wait_time_set = 2;
uint8_t           wait_time_end = 0;
uint8_t     temper_time_update = 0;
uint8_t     temper_time_set = 80;
uint8_t           temper_time_end = 0;

bool        start_flag = false;
bool        movement_flag = false;
bool        led_time_flag = false;

uint8_t     pin_read = 0;
uint8_t     pin_status = 0;
uint8_t     On_Off_reg = 0;
uint8_t     night_reg = 0;
uint16_t    pulse_start = 0;
uint16_t    pulse_end = 1000;
uint8_t     off_update = 0;
uint8_t     off_mains = 0;
uint8_t     led_time_off = 1;
uint8_t     led_time_update = 0;
uint8_t           mov_time_temp = 1;


/* USER CODE END PV */

/* Private function prototypes -----------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM6_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------
---*/
/* USER CODE BEGIN 0 */

/*
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

```
{
  if(GPIO_Pin == MOVEMENT_Pin)
  {
        movement_flag = true;
  }
}
*/

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
//TIM6 UPDATE EVERY 100ms //start_flag=true EVERY time_update*time_end
      if (htim->Instance == TIM6)
      {
            led_time_update = led_time_update + 1;
            if (led_time_update >= led_time_off)
            {
                  led_time_update = 0;
                  led_time_flag = false;
            }

            wait_time_update = wait_time_update + 1;
            if (wait_time_update >= wait_time_set)
            {
                  wait_time_update = 0;
                  wait_time_end = 1;
            }

            temper_time_update = temper_time_update + 1;
            if (temper_time_update >= temper_time_set)
            {
                  temper_time_update = 0;
                  temper_time_end = 1;
            }

      }




}


/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
```

```
/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_I2C1_Init();
MX_SPI1_Init();
MX_TIM6_Init();
/* USER CODE BEGIN 2 */

HAL_TIM_Base_Start_IT(&htim6);

NRF24_Init();

sht3x_handle_t handle =
{
            .i2c_handle = &hi2c1,
            .device_address = SHT3X_I2C_DEVICE_ADDRESS_ADDR_PIN_LOW
};
if (!sht3x_init(&handle))
{
      HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin,GPIO_PIN_SET);
      led_time_update = 0;
      HAL_Delay(2000);
}

memset(RxData, 0, sizeof RxData);
                              //clear
memset(TxData, 0, sizeof TxData);
                              //all
memset(device_Rx, 0, sizeof device_Rx);
                        //four
memset(device_Tx, 0, sizeof device_Tx);
                        //arrays

led_time_flag = false;

HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_SET);

HAL_Delay(1000);

HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_RESET);

id = device_id[0];
channel = device_channel[0];
NRF24_RxMode(RxAddress, channel);
```

```
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_12);

//AFTER 5 SEC TURN LED OFF
        if (led_time_flag == false)
        {
              HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_RESET);
              led_time_flag = true;
        }

//RECEIVER MODE CHANNEL 0 TO RECEIVE FROM MAIN CONTROL
        id = device_id[0];
        TxData[0] = id;
        channel = device_channel[0];
                                    //MAIN CONTROL CHANNEL=0
        NRF24_RxMode(RxAddress, channel);
//      HAL_Delay(10);

//BYPASS ACTIVATED?
        pin_read = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3);
                          //READ BYPASS SWITCH
        if (pin_read == 1)
                                    //BYPASS NOT ACTIVATED
        {

//TEMPERATURE AND HUMIDITY READ
//            if (temper_time_end == 1)
//            {
                  sht3x_read_temperature_and_humidity(&handle,
&temperature, &humidity);                  //READ TEMPERATURE AND HUMIDITY
                  humidity_int = (int)humidity;
                  temperature_int = (int)temperature;
                  TxData[3] = temperature_int;
                  TxData[4] = humidity_int;
                  temper_time_update = 0;
                  temper_time_end = 0;
//            }

//RECEIVE DATA
              if (isDataAvailable(1) == 1)
                                    //Did receive data?
              {
                  NRF24_Receive(RxData);
                                    //Write data to RxData register (32
bytes)
                  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
                  led_time_update = 0;
//            }

              //CHECK IF DATA IS FOR ME
```

```
		if (RxData[0] == id)
                                //Is data for me?
		{
                                                //If yes go on
			HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
			led_time_update = 0;
			NRF24_TxMode(TxAddress, channel);
                                //Go to Tx mode
			TxData[13] = 0;
                                //WAIT FOR NEXT TRANSMITTION WITH OK
BYTE (TxData[13] = 1;)
//			HAL_Delay(10);
//			HAL_Delay(1);
//			if (NRF24_Transmit(TxData) == 1)
                                //Send data back for confirmation
//			{
//				HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin,
GPIO_PIN_SET);                  // Send data 2 times
//				led_time_update = 0;
//				HAL_Delay(90);
//				HAL_Delay(100);
//				if (NRF24_Transmit(TxData) == 1)
//				{
//					HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin,
GPIO_PIN_SET);
//					led_time_update = 0;
//				}

//			}
//			else
//			{
//				HAL_GPIO_WritePin(GPIOB,
ERROR_LED_Pin,GPIO_PIN_SET);                                    //ERROR_LED ON
if Tx error
//				led_time_update = 0;
//			}

			for (d = 0; d < 32; d++)
			{
				device_Rx[d] = RxData[d];
				device_Tx[d] = RxData[d];
			}

//		}

//COPY REGISTERS
		On_Off = device_Rx[1];
                                                                //TURN MODULE
ON OR OFF
		day_night = device_Rx[7];
                                                        //DAY OR NIGHT
		sleep = device_Rx[8];
                                                        //SLEEP OR NO SLEEP
		mov_time_temp = device_Rx[12];
		temperature_set = device_Rx[5];
		humidity_set = device_Rx[6];


		if (On_Off == 1)
		{
```

```
//CHECK FOR MOVEMENT
                    reg = HAL_GPIO_ReadPin(GPIOA, MOVEMENT_Pin);
                                          //MOVEMENT CHECK
                    if (reg == 1)

      //MOVEMENT
                    {
                          device_Tx[2] = 1;

      //MOVEMENT TO REGISTER
                    }
                    else

      //NO MOVEMENT
                    {
                          device_Tx[2] = 0;
                                                                  //NO
MOVEMENT TO REGISTER
                    }

//TEMPERATURE AND HUMIDITY READ
//                  sht3x_read_temperature_and_humidity(&handle,
&temperature, &humidity);                 //READ TEMPERATURE AND HUMIDITY
//                  humidity_int = (int)humidity;
//                  temperature_int = (int)temperature;
                    device_Tx[3] = temperature_int;
                    device_Tx[4] = humidity_int;

                    if (device_Tx[2] == 1)
                                                                  //IF WE
HAVE MOVEMENT
                    {
                          if (temperature_int >= (temperature_set + 1))
                                                //IF TEMPERATURE OVER SET
VALUE
                          {
                                device_Tx[9] = 0;
                                                                  //TURN
OFF HEAT
                          }

                          if (temperature_int <= (temperature_set - 1))
                                                //IF TEMPERATURE UNDER SET
VALUE
                          {
                                device_Tx[9] = 1;
                                                                  //TURN ON
HEAT
                          }
                    }
                    else

      //IF WE HAVE NO MOVEMENT
                    {
                          if (temperature_int >= (temperature_set -1))
                                                //SET VALUES LOWER
                          {
                                device_Tx[9] = 0;
                          }

                          if (temperature_int <= (temperature_set - 2))
```

```
                        {
                              device_Tx[9] = 1;
                        }
                  }

                  if (humidity_int >= (humidity_set + 5))
                                          //IF HUMIDITY OVER SET
VALUE
                  {
                        device_Tx[10] = 1;
                                                                  //TURN ON
DEHUMDIFIER
                  }

                  if (humidity_int <= (humidity_set - 5))
                                          //IF HUMIDITY UNDER SET
VALUE
                  {
                        device_Tx[10] = 0;
                                                                  //TURN
OFF DEHUMDIFIER
                  }


//CHECK FOR OPEN WINDOW-DOOR
                  reg = HAL_GPIO_ReadPin(GPIOA, EXT_SWITCH_Pin);
                                          //WINDOW-DOOR SWITCH READ
                  if (reg == 1)
                                                                  //IF OPEN
                  {
                        device_Tx[9] = 0;
                                                                  //TURN
OFF HEAT
                  }


                  for (i = 0; i < 32; i++)
                  {
                        TxData[i] = device_Tx[i];
                  }


//SEND AND RECEIVE DATA TO PERIPHERAL UNITS

//            {
                  for(i=1; i<5; i++)
                  {
                        if (led_time_flag == false)
                        {
                              HAL_GPIO_WritePin(GPIOB,
ERROR_LED_Pin,GPIO_PIN_RESET);
                              HAL_GPIO_WritePin(GPIOB, LED_Pin,
GPIO_PIN_RESET);
                              HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin,
GPIO_PIN_RESET);
                              HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin,
GPIO_PIN_RESET);
                              led_time_flag = true;
                        }
                        channel = device_channel[0];
                        peripheral_id = device_id[i];
```

```
                        TxData[0] = peripheral_id;

//START TRANSMITTING DATA TO PERIPHERAL UNITS

//                      for (k = 0; k < 3; k++)
//                      {
                        NRF24_TxMode(TxAddress, channel);
                                                        //Go to TX mode
                        if (NRF24_Transmit(TxData) == 1)
                                                        //If data
transmitted
                        {
                              NRF24_RxMode(RxAddress, channel);
                                                        //Go to RX mode
                              HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin,
GPIO_PIN_SET);
                              led_time_update = 0;

      //Transmit LED ON
                        }
//                            NRF24_RxMode(RxAddress, channel);
                                                              //Go to
RX mode
//                            HAL_Delay(100);
//                            HAL_Delay(100);
//                            if (isDataAvailable(1) == 1)
//                            {
//                                  NRF24_Receive(RxData);
//                                  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin,
GPIO_PIN_SET);
//                                  led_time_update = 0;

//                                  if (RxData[0] == peripheral_id)
//                                  {
//                                        HAL_GPIO_TogglePin(GPIOB,
LED_Pin);
//                                        led_time_update = 0;
//                                        break;
//                                  }
//                            }
//                      }

                        NRF24_RxMode(RxAddress, channel);
                                                        //Go to RX mode
                        wait_reg = RxData[14];
                        wait_time_update = 0;
                        wait_time_end = 0;
                        while (wait_reg == 0)
                        {
//                              HAL_Delay(100);
                              HAL_Delay(100);
                              if (isDataAvailable(1) == 1)
                              {
                                NRF24_Receive(RxData);
                                HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin,
GPIO_PIN_SET);
                                led_time_update = 0;

                                if (RxData[0] == peripheral_id)
                                {
```

```
                                        HAL_GPIO_TogglePin(GPIOB,
LED_Pin);
                                        led_time_update = 0;
                                        wait_reg = RxData[14];
                                    }
                                }
                                if (wait_time_end == 1)
                                {
                                  wait_time_end = 0;
                                  goto Next;
                                }
                          }

Next:
//                        HAL_Delay(100);
                          HAL_Delay(100);
                    }

                    id = device_id[0];
                    TxData[0] = id;
                    channel = device_channel[0];
                                                //MAIN CONTROL CHANNEL=0
                    TxData[13] = 1;
                    NRF24_TxMode(TxAddress, channel);
                                                //Go to TX mode
                    if (NRF24_Transmit(TxData) == 1)
                                                //If data transmitted
                    {
                          NRF24_RxMode(RxAddress, channel);
                                                //Go to RX mode
                          HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin,
GPIO_PIN_SET);
                          led_time_update = 0;
                                                      //Transmit LED
ON
                    }

                }
                else
                {
                          TxData[9] = 0;
                          TxData[10] = 0;
                }

                }
                }
        }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
```

```
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_I2C1;
  PeriphClkInit.I2c1ClockSelection = RCC_I2C1CLKSOURCE_HSI;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.Timing = 0x2000090E;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
```

```
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Analogue filter
  */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Digital filter
  */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief TIM6 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM6_Init(void)
{

  /* USER CODE BEGIN TIM6_Init 0 */

  /* USER CODE END TIM6_Init 0 */

  /* USER CODE BEGIN TIM6_Init 1 */

  /* USER CODE END TIM6_Init 1 */
  htim6.Instance = TIM6;
  htim6.Init.Prescaler = 4800-1;
  htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim6.Init.Period = 1000-1;
  htim6.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM6_Init 2 */

  /* USER CODE END TIM6_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, CE_Pin|CSN_Pin|Rx_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pins : MOVEMENT_Pin EXT_SWITCH_Pin */
  GPIO_InitStruct.Pin = MOVEMENT_Pin|EXT_SWITCH_Pin;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : CE_Pin CSN_Pin Rx_LED_Pin */
  GPIO_InitStruct.Pin = CE_Pin|CSN_Pin|Rx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : LED_1_Pin LED_Pin ERROR_LED_Pin Tx_LED_Pin */
  GPIO_InitStruct.Pin = LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : OVERRIDE_Pin */
  GPIO_InitStruct.Pin = OVERRIDE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OVERRIDE_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
```

```
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

ΠΑΡΑΡΤΗΜΑ C : Κώδικας STM32F030C8T6 Δέκτη Διακόπτη

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
***
  * @project name :     Receiver_Switch_STM32F030C8T6
  * @author             :     VPK Engineering
  * @file           :   main.c
  * @brief          :   Simple Receiver Switch with relay and override
switch
  * @version            :     V 1.0

******************************************************************************
***
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
***
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------
---*/
#include "main.h"

/* Private includes ----------------------------------------------------------
---*/
/* USER CODE BEGIN Includes */

#include "NRF24L01.h"
#include "stdio.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
---*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------
---*/
```

```
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------
---*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------
---*/
SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim3;

/* USER CODE BEGIN PV */

uint8_t RxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t RxData[32];
uint8_t TxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t Tx_Data[32];

uint8_t device_1_Rx[32];
uint8_t device_1_Tx[32];

uint8_t dev_1_channel = 10;

uint8_t id = 30;
uint8_t i = 0;
uint8_t reg = 0;
uint8_t timer = 0;
uint16_t update = 0;
uint16_t time_set = 60;
uint8_t pin_read = 0;
uint8_t pin_status = 0;
uint8_t On_Off_reg = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

void OFF ()
{
      HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_RESET);
      pin_status = 0;
      update = 0;
      timer = 0;
      HAL_TIM_Base_Stop_IT(&htim3);
}

void ON ()
{
      HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_SET);
      pin_status = 1;
```

```
}

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
---*/
/* USER CODE BEGIN 0 */

      void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
        {
        if (htim->Instance == TIM3)
        {
              update = update + 1;
              HAL_GPIO_TogglePin(GPIOB, LED_1_Pin);
              if (update >= time_set)
              {
                    update = 0;
                    timer = 0;
                    HAL_TIM_Base_Stop_IT(&htim3);
                    OFF();
              }
        }
        }

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
```

```
HAL_TIM_Base_Stop_IT(&htim3);

NRF24_Init();

NRF24_RxMode(RxAddress, dev_1_channel);

HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_SET);

HAL_Delay(1000);

HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_RESET);
timer = 0;

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
      device_1_Rx[14] = 0;
      NRF24_RxMode(RxAddress, dev_1_channel);
      while (isDataAvailable(1) == 0)
      {
            HAL_Delay(1);
      }
      NRF24_Receive(device_1_Rx);
                //Write data to device_1_Rx register (32 bytes)
      HAL_GPIO_TogglePin(GPIOA, Rx_LED_Pin);

      if (device_1_Rx[0] == id)
                      //Is data for me?
      {
                                    //If yes go on
            HAL_GPIO_TogglePin(GPIOB, LED_Pin);

            NRF24_TxMode(TxAddress, dev_1_channel);
          //Go to Tx mode
            device_1_Rx[14] = 1;
            HAL_Delay(100);
            if (NRF24_Transmit(device_1_Rx) == 1)
                //Send data back for confirmation
            {
                  HAL_GPIO_TogglePin(GPIOB, Tx_LED_Pin);
          // Send data
                  HAL_Delay(200);
            }
            else
            {
                  HAL_GPIO_TogglePin(GPIOB, ERROR_LED_Pin);
                //Toggle ERROR_LED if Tx error
            }
```

```
//-----------------------------------------------CHECK REGISTERS-----------
---------------------------------
//-------------------------------------------------------------------------
---------------------------------

              reg = device_1_Rx[12];
              time_set = (reg * 60);

              reg = device_1_Rx[2];
              if (reg == 1)
                  //Movement (1) or NO Movement (0)
              {
                    reg = device_1_Rx[7];
                    if (reg == 1)
                  //DAY = 1    NIGHT = 0
                    {
                          HAL_Delay(10);
                          update = 0;
                          timer = 0;
                          HAL_TIM_Base_Stop_IT(&htim3);
                          ON();
                    }
                    else
                    {
                          pin_read = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3);
            //READ BYPASS SWITCH
                          if (pin_read == 1)
                        //BYPASS OFF
                          {
                                OFF();
                          }
                          else
                          {
                                update = 0;
                                timer = 0;
                                HAL_TIM_Base_Stop_IT(&htim3);
                                ON();
                          }
                    }
              }
              else
              {
                    if (pin_status == 0)
                    {
                          OFF();
                    }
                    else
                    {
                          if (timer == 0)
                          {
                                HAL_TIM_Base_Start_IT(&htim3);
                                timer = 1;
                                update = 1;
                                HAL_Delay(10);
                          }
                          if (update == 0)
                          {
                                OFF();
                          }
                    }
```

```
            }

        }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */
```

```
  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 4800-1;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 10000-1;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, CE_Pin|CSN_Pin|Rx_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin
                          |RELLAY_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Gen_On_Off_Pin */
  GPIO_InitStruct.Pin = Gen_On_Off_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(Gen_On_Off_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : CE_Pin CSN_Pin Rx_LED_Pin */
  GPIO_InitStruct.Pin = CE_Pin|CSN_Pin|Rx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : LED_1_Pin LED_Pin ERROR_LED_Pin Tx_LED_Pin
                           RELLAY_Pin */
  GPIO_InitStruct.Pin = LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin
                          |RELLAY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : OVERRIDE_Pin */
  GPIO_InitStruct.Pin = OVERRIDE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OVERRIDE_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

ΠΑΡΑΡΤΗΜΑ D : Κώδικας STM32F030C8T6 Δέκτη Dimmer

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
  * @project name :     Receiver_Dimmer_STM32F030C8T6
  * @author             :     VPK Engineering
  * @file           :   main.c
  * @brief          :   Simple Receiver Switch with dimmer and override
switch
  * @version            :     V 0.2

******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------
---*/
#include "main.h"

/* Private includes ----------------------------------------------------------
---*/
/* USER CODE BEGIN Includes */

#include "NRF24L01.h"
#include "stdio.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
---*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

```
/* Private define ------------------------------------------------------------
---*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
---*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
---*/
SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim3;

/* USER CODE BEGIN PV */

uint8_t RxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t RxData[32];
uint8_t TxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t Tx_Data[32];

uint8_t device_1_Rx[32];
uint8_t device_1_Tx[32];

uint8_t dev_1_channel = 10;

uint8_t id = 60;
uint8_t i = 0;
uint8_t reg = 0;
uint8_t timer = 0;
uint16_t update = 0;
uint16_t time_set = 60;
uint8_t pin_read = 0;
uint8_t pin_status = 0;
uint8_t On_Off_reg = 0;
uint8_t night_reg = 0;
uint16_t pulse_start = 0;
uint16_t pulse_end = 1000;
uint8_t off_update = 0;
uint8_t off_mains = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

void OFF ()
{
      HAL_TIM_OnePulse_Stop(&htim1, TIM_CHANNEL_1);
      pin_status = 0;
```

```
	update = 0;
	timer = 0;
	HAL_TIM_Base_Stop_IT(&htim3);
	HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_RESET);
}

void ON ()
{
	HAL_TIM_OnePulse_Start(&htim1, TIM_CHANNEL_1);
	pin_status = 1;
	HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_SET);
}

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
---*/
/* USER CODE BEGIN 0 */

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
  {
  if (htim->Instance == TIM3)
  {
	update = update + 1;
//	HAL_GPIO_TogglePin(GPIOB, LED_1_Pin);
	if (update >= time_set)
	{
		update = 0;
		timer = 0;
		HAL_TIM_Base_Stop_IT(&htim3);
		OFF();
	}
  }

/*
  if (htim->Instance == TIM15)
	//Interrupt from TIM$
  {
	pin_read = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_9);	//Read pin A9
(zero crossing signal)
	  if (pin_read == 1)
	//If pin high
	  {
		off_update = off_update + 1;
	//Update off_update register
		if (off_update > 50)
	//If high for 50 times in a row (8ms * 50 = 400ms)
		{
			off_mains = 1;
	//Ac mains NOT present
		}
	  }
	  else
		//If pin low (ac mains present)
	  {
		off_update = 0;
	//off_update register 0
		off_mains = 0;
	//Ac mains present
	  }
  }
```

```
  */

  }

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_TIM1_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */

  HAL_TIM_Base_Stop_IT(&htim3);
//  HAL_TIM_Base_Start_IT(&htim15);
  HAL_TIM_OnePulse_Stop(&htim1, TIM_CHANNEL_1);

  NRF24_Init();

  NRF24_RxMode(RxAddress, dev_1_channel);

  HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_SET);

  HAL_Delay(1000);

  HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_RESET);
```

```
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_RESET);
  timer = 0;

  TIM1->CCR1 = 0;
  TIM1->ARR = 1000;

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        device_1_Rx[14] = 0;
        NRF24_RxMode(RxAddress, dev_1_channel);
        while (isDataAvailable(1) == 0)
        {
              HAL_Delay(1);
        }
        NRF24_Receive(device_1_Rx);
                  //Write data to device_1_Rx register (32 bytes)
        HAL_GPIO_TogglePin(GPIOA, Rx_LED_Pin);

        if (device_1_Rx[0] == id)
                        //Is data for me?
        {
                                      //If yes go on
              HAL_GPIO_TogglePin(GPIOB, LED_Pin);

              NRF24_TxMode(TxAddress, dev_1_channel);
            //Go to Tx mode
              device_1_Rx[14] = 1;
              HAL_Delay(10);
              if (NRF24_Transmit(device_1_Rx) == 1)
                  //Send data back for confirmation
              {
                    HAL_GPIO_TogglePin(GPIOB, Tx_LED_Pin);
            // Send data
                    HAL_Delay(10);
              }
              else
              {
                    HAL_GPIO_TogglePin(GPIOB, ERROR_LED_Pin);
                  //Toggle ERROR_LED if Tx error
              }

//-----------------------------------------------CHECK REGISTERS-----------
----------------------------------
//-------------------------------------------------------------------------
----------------------------------

              reg = device_1_Rx[12];
              time_set = (reg * 60);

              reg = device_1_Rx[2];
              if (reg == 1)
                  //Movement (1) or NO Movement (0)
              {
                    reg = device_1_Rx[7];
                    if (reg == 1)
                  //DAY = 1    NIGHT = 0
```

```
                {
//                      HAL_Delay(10);
                        TIM1->CCR1 = 0;
                        TIM1->ARR = 1000;
                        update = 0;
                        timer = 0;
                        HAL_TIM_Base_Stop_IT(&htim3);
                        ON();
                }
                else
                {
                        pin_read = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3);
            //READ BYPASS SWITCH
                        if (pin_read == 1)
                    //BYPASS OFF
                        {
                              OFF();
                        }
                        else
                        {
                              reg = device_1_Rx[8];
                              if (reg == 1)
                                //SLEEP_LIGHT_ON = 1    SLEEP_LIGHT_OFF = 0
                              {
                                    TIM1->CCR1 = 6000;
                                        //If light ON it is dimmed
                                    TIM1->ARR = 8000;
                              }
                              else
                              {
                                    TIM1->CCR1 = 0;
                                    TIM1->ARR = 1000;
                              }
                              update = 0;
                              timer = 0;
                              HAL_TIM_Base_Stop_IT(&htim3);
                              ON();
                        }
                }
          }
          else
          {
                if (pin_status == 0)
                {
                      OFF();
                }
                else
                {
                      if (timer == 0)
                      {
                            HAL_TIM_Base_Start_IT(&htim3);
                            timer = 1;
                            update = 1;
                            HAL_Delay(10);
                      }
                      if (update == 0)
                      {
                            OFF();
                      }
                }
          }
```

```
        }


    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */
```

```
  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_SlaveConfigTypeDef sSlaveConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 48-1;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 10000-1;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_OC_Init(&htim1) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  if (HAL_TIM_OnePulse_Init(&htim1, TIM_OPMODE_SINGLE) != HAL_OK)
  {
    Error_Handler();
  }
  sSlaveConfig.SlaveMode = TIM_SLAVEMODE_TRIGGER;
  sSlaveConfig.InputTrigger = TIM_TS_TI2FP2;
  sSlaveConfig.TriggerPolarity = TIM_TRIGGERPOLARITY_RISING;
  sSlaveConfig.TriggerFilter = 0;
  if (HAL_TIM_SlaveConfigSynchro(&htim1, &sSlaveConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM2;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_OC_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */
  HAL_TIM_MspPostInit(&htim1);

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{
```

```
  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 4800-1;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 10000-1;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, CE_Pin|CSN_Pin|Rx_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin,
GPIO_PIN_RESET);
```

```
  /*Configure GPIO pins : CE_Pin CSN_Pin Rx_LED_Pin */
  GPIO_InitStruct.Pin = CE_Pin|CSN_Pin|Rx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : LED_1_Pin LED_Pin ERROR_LED_Pin Tx_LED_Pin */
  GPIO_InitStruct.Pin = LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : OVERRIDE_Pin */
  GPIO_InitStruct.Pin = OVERRIDE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OVERRIDE_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
```

```
#endif /* USE_FULL_ASSERT */
```

ΠΑΡΑΡΤΗΜΑ Ε : Κώδικας STM32F030C8T6 Αφυγραντήρα

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
***
  * @project name :     Receiver_Switch_Hum_STM32F030C8T6
  * @author             :     VPK Engineering
  * @file           :   main.c
  * @brief          :   Simple Receiver Switch for Dehumidifier with
Override switch
  * @version            :     V 1.0

******************************************************************************
***
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
***
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------
---*/
#include "main.h"

/* Private includes ----------------------------------------------------------
---*/
/* USER CODE BEGIN Includes */

#include "NRF24L01.h"
#include "stdio.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
---*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */
```

```
/* Private define ------------------------------------------------------------
---*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
---*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
---*/
SPI_HandleTypeDef hspi1;

/* USER CODE BEGIN PV */

uint8_t RxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t RxData[32];
uint8_t TxAddress[] = {0xCD, 0xAB, 0xEF, 0xCD, 0xAB};
uint8_t Tx_Data[32];

uint8_t device_1_Rx[32];
uint8_t device_1_Tx[32];

uint8_t dev_1_channel = 10;

uint8_t id = 70;
uint8_t i = 0;
uint8_t reg = 0;
uint8_t pin_read = 0;
uint8_t On_Off_reg = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
/* USER CODE BEGIN PFP */

void OFF ()
{
      HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_RESET);
}

void ON ()
{
      HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_SET);
}

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
---*/
/* USER CODE BEGIN 0 */
```

```c
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  /* USER CODE BEGIN 2 */


  NRF24_Init();

  NRF24_RxMode(RxAddress, dev_1_channel);

  HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_SET);

  HAL_Delay(1000);

  HAL_GPIO_WritePin(GPIOB, LED_1_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, ERROR_LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, RELLAY_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Rx_LED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, Tx_LED_Pin, GPIO_PIN_RESET);


  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        pin_read = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);           //READ
Gen_On_Off SWITCH
        if (pin_read == 1)
      //Gen_On_Off ON
        {
              device_1_Rx[14] = 0;
              NRF24_RxMode(RxAddress, dev_1_channel);
              while (isDataAvailable(1) == 0)
              {
                    HAL_Delay(1);
              }
              NRF24_Receive(device_1_Rx);
                        //Write data to device_1_Rx register (32 bytes)
              HAL_GPIO_TogglePin(GPIOA, Rx_LED_Pin);

              if (device_1_Rx[0] == id)
                              //Is data for me?
              {
                                          //If yes go on
                    HAL_GPIO_TogglePin(GPIOB, LED_Pin);

                    NRF24_TxMode(TxAddress, dev_1_channel);
                  //Go to Tx mode
                    device_1_Rx[14] = 1;
                    HAL_Delay(10);
                    if (NRF24_Transmit(device_1_Rx) == 1)
                        //Send data back for confirmation
                    {
                          HAL_GPIO_TogglePin(GPIOB, Tx_LED_Pin);
                  // Send data
                          HAL_Delay(10);
                    }
                    else
                    {
                          HAL_GPIO_TogglePin(GPIOB, ERROR_LED_Pin);
                        //Toggle ERROR_LED if Tx error
                    }

//-----------------------------------------------CHECK REGISTERS-----------
---------------------------------
//-------------------------------------------------------------------------
---------------------------------

                    reg = device_1_Rx[10];
                    if (reg == 1)
                        //Dehumidifier ON (1) or Dehumidifier OFF (0)
                    {
                          reg = device_1_Rx[7];
                          if (reg == 1)
                        //DAY = 1    NIGHT = 0
                          {
                                ON();
                          }
                          else
                          {
```

```
                                    pin_read = HAL_GPIO_ReadPin(GPIOB,
GPIO_PIN_3);            //READ BYPASS SWITCH
                                    if (pin_read == 1)
                                  //BYPASS OFF
                                    {
                                          OFF();
                                    }
                                    else
                                    {
                                          ON();
                                    }
                              }
                        }
                        else
                        {
                              OFF();
                        }
                  }
            }
            else
            {
                  OFF();
            }

      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
```

```
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
```

```
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, CE_Pin|CSN_Pin|Rx_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin
                          |RELLAY_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Gen_On_Off_Pin */
  GPIO_InitStruct.Pin = Gen_On_Off_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(Gen_On_Off_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : CE_Pin CSN_Pin Rx_LED_Pin */
  GPIO_InitStruct.Pin = CE_Pin|CSN_Pin|Rx_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : LED_1_Pin LED_Pin ERROR_LED_Pin Tx_LED_Pin
                           RELLAY_Pin */
  GPIO_InitStruct.Pin = LED_1_Pin|LED_Pin|ERROR_LED_Pin|Tx_LED_Pin
                          |RELLAY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : OVERRIDE_Pin */
  GPIO_InitStruct.Pin = OVERRIDE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OVERRIDE_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
```

```c
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```