ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΞΥΠΝΟ ΣΠΙΤΙ ΚΑΙ ΑΣΥΡΜΑΤΗ ΕΠΙΚΟΙΝΩΝΙΑ

Αικατερίνη Νικολάκη

Εισηγητές: Βελώνη Αναστασία

ΑΙΓΑΛΕΩ

2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έξυπνο Σπίτι και Ασύρματη Επικοινωνία

Αικατερίνη Νικολάκη
ΑΜ: 41757

Εισηγητής:

Βελώνη Αναστασία, Καθηγήτρια

Εξεταστική Επιτροπή:

Ιωάννης Έλληνας, Υπεύθυνος Τμήματος

Ημερομηνία Εξέτασης:

__/__/20__

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Νικολάκη Αικατερίνη του Δημητρίου, με αριθμό μητρώου 41757 φοιτήτρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Ολοκληρώνοντας τον κύκλο σπουδών μου με την παρούσα πτυχιακή εργασία, θα ήθελα να εκφράσω τις ευχαριστίες μου σε συγκεκριμένα άτομα που με τον δικό τους τρόπο, με βοήθησαν και με στήριξαν.

Πρώτα και κυριότερα θα ήθελα να ευχαριστήσω την οικογένειά μου, που με παρότρυνε να επιμείνω σε ένα δύσκολο κομμάτι, όπως αυτό της πτυχιακής εργασίας και εν συνεχεία να τα καταφέρω.

Το συμφοιτητή μου Βασίλη Λασκαρίδη για την πολύτιμη βοήθειά του στην εκπλήρωση του πρακτικού μέρους και για τις γνώσεις του στο αντικείμενο.

Την καθηγήτριά μου, κ. Αναστασία Βελώνη, για την άριστη συνεργασία μας και την επίβλεψη της καθ' όλη τη διάρκεια της εκπόνησης της εργασίας.

Ακόμα, θα ήθελα να εκφράσω ευχαριστίες προς εκείνους τους καθηγητές του τμήματος που μου προσέφεραν τις γνώσεις και συνεπώς τις ικανότητες που έχω σήμερα στον τομέα της πληροφορικής και της μηχανικής υπολογιστών.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή παρουσιάζεται η εφαρμογή ενός δικτύου για Έξυπνο Σπίτι (Smart Home). Τα έξυπνα συστήματα (Smart Systems) μπορούν και ελέγχουν ηλεκτρολογικές εγκαταστάσεις καθώς και συσκευές μέσω ενός ενοποιημένου συστήματος. Το σύστημα που δημιουργήθηκε στην εργασία αυτή χρησιμοποιεί ηλεκτρονικά εξαρτήματα ραδιοσυχνοτήτων γνωστά ως RF και πιο συγκεκριμένα χρησιμοποιούνται nRF24L01. Οι βασικές εφαρμογές που θα προβληθούν είναι:

- Άνοιγμα και κλείσιμο διακόπτη λάμπας, εφόσον υπάρχει κίνηση στο χώρο, ανάλογα με τη φωτεινότητα που υπάρχει στο δωμάτιο.
- Χρήση του συναγερμού όταν ο ιδιοκτήτης βρίσκεται εκτός του σπιτιού.
- Μέτρηση θερμοκρασίας και υγρασίας εντός και εκτός του σπιτιού.
- Αυτόματη ενεργοποίηση του κλιματισμού όταν η θερμοκρασία αποκλίνει από τις επιθυμητές τιμές.
- Μέτρηση κατανάλωσης ρεύματος.

Αρχικά, στην εργασία αυτή γίνεται μια εισαγωγή στα Smart Homes και παρουσιάζονται τα πλεονεκτήματά τους καθώς και η άμεση ένταξή τους στην καθημερινότητά μας. Έπειτα, παρουσιάζονται πληροφορίες και βασικά στοιχεία για την πλατφόρμα Arduino και για τον πομποδέκτη nRF24 καθώς και για την τοπολογία δικτύου που χρησιμοποιήθηκε.

Επιστημονική Περιοχή:

Οικιακοί Αυτοματισμοί

Λέξεις Κλειδιά:

Έξυπνο Σπίτι, Οικιακοί Αυτοματισμοί, Ασύρματος έλεγχος, Arduino, ATmega328

ABSTRACT

In the present thesis it is presented the implementation of a Smart Home network. Smart Systems can control electrical systems and devices through a unified system. The system, which was created for this thesis, uses electronic RF components and specifically uses nRF24L01. The main applications that will be projected are the following:

- Turning on and off the lights, whenever there is movement in the room, according to the brightness inside it.
- Use of the alarm system, when the owner is outside of the house.
- Temperature and humidity measurement, inside and outside of the house.
- Automatic activation of the air conditioning whenever temperature exceeds the desired values.
- Power consumption measurement.

Initially, in this thesis there will be an introduction to Smart Homes as also as their advantages and their direct integration into our lives. Then, there will be a presentation of information and main data for Arduino platform, the transceiver nRF24 and the network topology used.

Science Field:

Home Automation

Keywords:

Smart Home, Home Automations, Wireless control, Arduino Uno, ATmega328

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

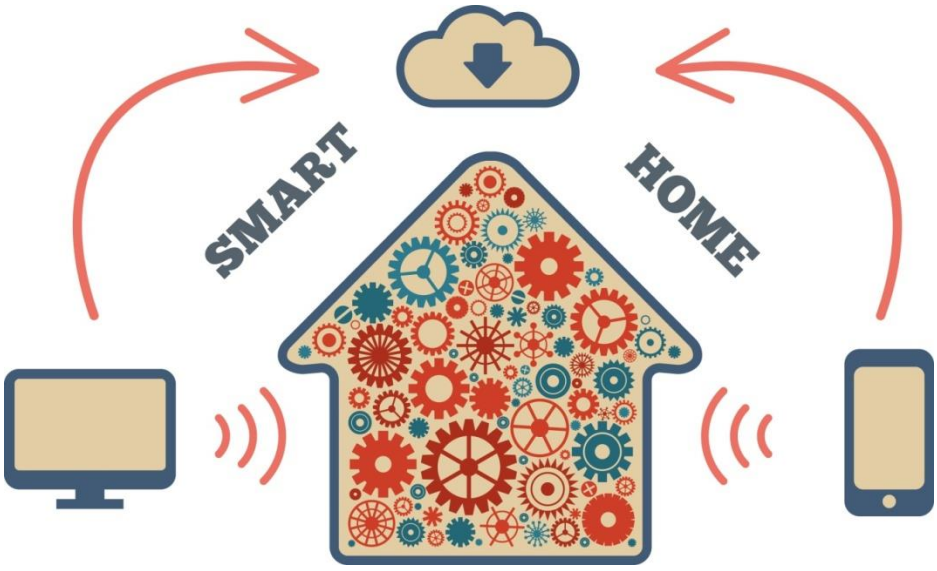
ACK	Acknowledgement
ALU	Arithmetic Logic Unit
AREF	Analog Reference
CE	Chip Enable
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
CSN	Chip Select Not
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First In First Out
FTDI	Future Technology Devices International
GFSK	Gaussian Frequency Shift Keying
GND	Ground
ICSP	In Circuit Serial Programming
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
MCU	Microcontroller Unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
PWM	Pulse Width Modulated
PWR_DWN	Power Down
PWR_UP	Power Up
RX	Receive
SCK	Serial Clock
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TX	Transmit
USB	Universal Serial Bus

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Οι σημερινοί ρυθμοί ζωής και οι συνεχώς αυξανόμενες απαιτήσεις σε διάφορους τομείς της καθημερινότητάς μας, έχουν περιορίσει κατά πολύ τον ελεύθερό μας χρόνο. Η πίεση και το άγχος που δέχεται ο σύγχρονος άνθρωπος, είναι αυτά που τον κάνουν να αναζητά άνεση, ασφάλεια και λειτουργικότητα όταν θα βρίσκεται στο δικό του προσωπικό χώρο, δηλαδή την κατοικία του.

Για τους λόγους αυτούς απαιτείται από την τεχνολογία να έρθει σε συνεργασία με τον χώρο αυτό και να δημιουργήσει ένα ζωντανό οργανισμό ο οποίος θα «σκέφτεται», θα παίρνει πρωτοβουλίες και θα αντιδρά με τους κατοίκους του, είτε αυτοί βρίσκονται εντός είτε εκτός αυτού. Κάπως έτσι γεννήθηκε η ανάγκη να βρεθεί ένα πλαίσιο διαδικασιών, το οποίο θα εξυπηρετεί τις ανάγκες των κατοίκων αυτών αλλά και θα μεγιστοποιεί τις δυνατότητές τους. Στο πλαίσιο αυτό φτιάχτηκαν τα «έξυπνα» συστήματα και κατ' επέκταση τα «Έξυπνα» Σπίτια.



Εικόνα 1.1: Το Smart Home σαν ζωντανός οργανισμός που λειτουργεί όπως εμείς του ορίσουμε

1.1. Ιστορική Αναδρομή σε Έξυπνα Συστήματα

Η λέξη «έξυπνο» χρησιμοποιήθηκε επίσημα αναφορικά με τεχνολογικές κατασκευές κατά τη δεκαετία του 70, όταν ξεκίνησαν να κατασκευάζονται στρατιωτικά προϊόντα, όπως βόμβες και πύραυλοι οι οποίοι έχοντας έναν στόχο, κατευθυνόντουσαν από μόνοι τους προς αυτόν. Αργότερα, κατά τη δεκαετία του 80 η λέξη «έξυπνο» άρχισε να χρησιμοποιείται και για συσκευές οι οποίες αποτελούνταν από μικροελεγκτές, όπως οι υπολογιστές ή άλλες προηγμένες συσκευές.

Φτάνοντας στο σήμερα έχουμε πάψει πλέον να αποκαλούμε ένα υπολογιστή έξυπνο, ακόμα και εάν οι δυνατότητες τον σύγχρονων υπολογιστών είναι κατά πολύ μεγαλύτερες εκείνων του ’80. Σήμερα τα έξυπνα συστήματα μπορούν και ελέγχουν ηλεκτρολογικές και μηχανολογικές εγκαταστάσεις καθώς και συσκευές και εφαρμογές μέσω ενός ενοποιημένου συστήματος με σκοπό τη βελτίωση του επιπέδου της άνεσης, της ασφάλειας, της επικοινωνίας και της ενεργειακής διαχείρισης.

Το Έξυπνο Σπίτι είναι ένα τέτοια σύστημα καθώς μέσω αυτοματισμών που ελέγχουν τις εγκαταστάσεις και τον οικιακό εξοπλισμό κάνει τη ζωή και τη διαμονή των κατοίκων του πολύ πιο άνετη και τους παρέχει δυνατότητες που δεν θα μπορούσαν να έχουν.

1.2. Τι είναι το Έξυπνο Σπίτι;

Ένας ορισμός για το έξυπνο σπίτι είναι ο εξής: «Μία κατοικία η οποία διαθέτει συσκευές, φωτισμό, θέρμανση, κλιματισμό, υπολογιστές, συστήματα ασφαλείας, συστήματα ψυχαγωγίας κοκ, τα οποία μπορούν και επικοινωνούν μεταξύ τους αλλά και να ελέγχονται μέσω ενός δικτύου (ασύρματου ή ενσύρματου), θεωρείται έξυπνη». Πιο συγκεκριμένα, το Smart Home είναι το σπίτι εκείνο που όντας εξοπλισμένο με διάφορες τεχνολογικές συσκευές και εργαλεία, αυτοματοποιεί τη ζωή των κατοίκων του αναλόγως με τις προτιμήσεις και τις ανάγκες τους.

Εικόνα 1.2: Ένα Έξυπνο Σπίτι φτιαγμένο στην πλατφόρμα Qivicon

Σαν έξυπνο σύστημα που είναι, συνδυάζει όλες τις συσκευές τηλεπικοινωνιών, τα ηχοσυστήματα, τις οθόνες και κυριότερα τους αισθητήρες και μετρητές(κίνησης, φωτός, ρεύματος, θερμοκρασίας, υγρασίας, καπνού, κοκ) και τις αυτοματοποιεί λαμβάνοντας πρωτοβουλίες.

1.3. Τα πλεονεκτήματα των Smart Homes

Τα πλεονεκτήματα που προκύπτουν από το συντονισμό των συστημάτων αφορούν τη διευκόλυνση της καθημερινότητας των κατοίκων. Εφόσον υπάρχει σωστός προγραμματισμός του συστήματος, βελτιώνεται η ποιότητα ζωής καθώς επίσης εξοικονομείται ενέργεια και συνεπώς μειώνονται τα έξοδα.

Υπάρχουν πολλοί λόγοι που χρησιμοποιούμε τα έξυπνα συστήματα και οι πιο σημαντικοί είναι:

1. Άνεση

   Αυτόματη διαχείριση των εξής:

   - Φωτισμού
   - Θερμοκρασίας
   - Παραθύρων και τεντών
   - Θερμοσίφωνα
   - Κουζίνας
   - Πριζών

2. Επικοινωνία
   - Αυτόματος συγχρονισμός με κινητό για άμεση προβολή της κατάστασης του σπιτιού
   - Αυτόματος συγχρονισμός των συσκευών πολυμέσων

3. Ασφάλεια
   - Επιτήρηση της οικίας μέσω κάμερας
   - Συναγερμός

4. Εξοικονόμηση Ενέργειας
   - Μέτρηση της κατανάλωσης ρεύματος
   - Ενεργοποίηση εγκαταστάσεων και συσκευών σε συγκεκριμένες χρονικές στιγμές
   - Απενεργοποίηση εγκαταστάσεων και συσκευών μετά από συγκεκριμένα χρονικά διαστήματα

ΚΕΦΑΛΑΙΟ 2

Η ΥΠΟΛΟΓΙΣΤΙΚΗ ΠΛΑΤΦΟΡΜΑ ARDUINO

Εικόνα 2.1: Το λογότυπο του Arduino

2.1. Γενικές Πληροφορίες σχετικά με το Arduino

Το Arduino, είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια μητρική πλακέτα, με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για Τη C++ με κάποιες μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή. Στην ενότητα αυτή, παρουσιάζεται η πλατφόρμα Arduino που χρησιμοποιήθηκε για την πραγματοποίηση της εργασίας αυτής και τα χαρακτηριστικά της.

Το Arduino είναι μία ανοιχτού λογισμικού πλατφόρμα πρωτοτύπων ηλεκτρονικών συσκευών που βασίζονται στην ευελιξία και στην ευκολία χρήσης υλικού και λογισμικού. Το Arduino μπορεί να αλληλεπιδρά με το περιβάλλον, κάνοντας λήψη σημάτων μέσα από μια ποικιλία αισθητήρων. Τα έργα που βασίζονται σε αυτόν τον μικροελεγκτή, μπορούν να είναι αυτόνομα ή μπορούν να επικοινωνούν με το λογισμικό που τρέχει σε έναν υπολογιστή (π.χ. Flash, Processing).[1]

2.1.1. Τεχνικά χαρακτηριστικά του Arduino Uno

Για την παρούσα διπλωματική εργασία, χρησιμοποιήθηκε σαν μικροελεγκτής το Arduino Uno. Μία πλακέτα Arduino Uno αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz. Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής.[1]

Ο ακόλουθος πίνακας αναλύει τα χαρακτηριστικά του.

Μικροελεγκτής	ATMEGA328
Τάση λειτουργίας	5V
Τάση εισόδου	7-12V
Όρια τάσης εισόδου	6-20V
Ψηφιακοί ακροδέκτες I/O	14, (6 PWM έξοδοι)
Αναλογικοί ακροδέκτες εισόδου	6
Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη	40mA
Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V	50mA
Μνήμη flash	32KB (ATMEGA328)
Μνήμη SRAM	2KB (ATMEGA328)
Μνήμη EEPROM	1KB (ATMEGA328)
Ταχύτητα ρολογιού	16MHz

Πίνακας 2.1: Χαρακτηριστικά του Arduino Uno

2.1.2. Τροφοδοσία

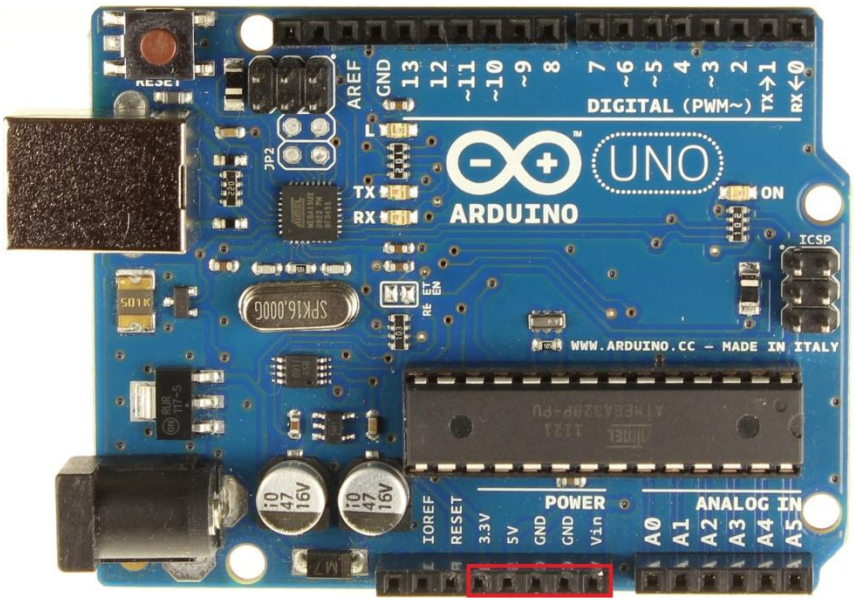
Το Arduino, πρέπει να τροφοδοτηθεί με ρεύμα, είτε από τον υπολογιστή μέσω της σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm που βρίσκεται στην κάτω αριστερή γωνία. Για την αποφυγή προβλημάτων, η εξωτερική τροφοδοσία θα πρέπει να είναι από 7 ως 12V. Οι ακροδέκτες τροφοδοσίας είναι οι ακόλουθοι:

- Vin: Η τάση εισόδου της πλακέτας, όταν χρησιμοποιεί εξωτερική πηγή ενέργειας. Η τροφοδοσία τάσης γίνεται μέσω αυτού του ακροδέκτη.

- 5V: Η τάση που χρησιμοποιείται από τα διάφορα μέρη της πλακέτας και το μικροελεγκτή είναι 5V. Η τάση αυτή, την οποία δίνει αυτός ο ακροδέκτης, είναι είτε η τάση 5V που δίνει η σύνδεση με USB, είτε η ρυθμισμένη τάση που δίνεται μέσω του Vin.

- 3.3V: Η τάση αυτή παράγεται από το ολοκληρωμένο FTDI. Το όριο άντλησης ρεύματος είναι 50mA.

- GND: Είσοδοι γείωσης.



Εικόνα 2.2 : Οι ακροδέκτες τροφοδοσίας του Arduino Uno

2.1.3. Μνήμη Arduino

Ο μικροεπεξεργαστής ATmega328, έχει τρεις ομάδες μνήμης. Διαθέτει flash memory, στην οποία αποθηκεύονται τα Arduino sketch, SRAM (Static Random Access Memory), στην οποία δημιουργείται το sketch και χρησιμοποιεί τις μεταβλητές όταν τρέχει, και EEPROM, η οποία χρησιμοποιείται από τους προγραμματιστές για την αποθήκευση μακροχρόνιων πληροφοριών. Πιο συγκεκριμένα, η μνήμη του ATmega328 αποτελείται από:

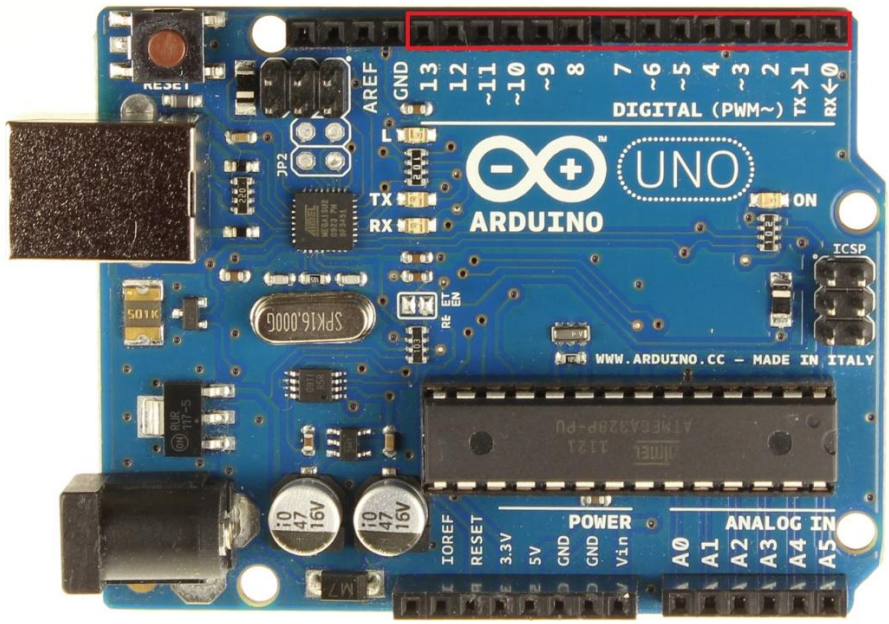
- 2KB μνήμης SRAM: Η ωφέλιμη μνήμη, που μπορεί να χρησιμοποιήσει ο κώδικας για να αποθηκεύει μεταβλητές, πίνακες κ.λπ. Η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή πατηθεί το κουμπί επανεκκίνησης(reset).

- 1KB μνήμης EEPROM: Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τον κώδικα. Σε αντίθεση με την SRAM, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

- 32KB μνήμης Flash: Από τα 32, τα 2 KB χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware, είναι αναγκαίο για την εγκατάσταση κώδικα στο μικροελεγκτή μέσω της θύρας USB, καθώς ο επεξεργαστής χωρίς αυτό μπορεί να προγραμματιστεί μόνο μέσω πρωτοκόλλουSPI. Τα υπόλοιπα 30KB της μνήμης Flash, χρησιμοποιούνται για την αποθήκευση αυτού ακριβώς του κώδικα, αφού πρώτα μεταγλωττιστεί στον υπολογιστή. Η μνήμη Flash, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

2.1.4. Ακροδέκτες Arduino Uno

Κάθε μικροελεγκτής Arduino διαθέτει εισόδους και εξόδους για την αλληλεπίδραση με το περιβάλλον του και τα εξαρτήματα. Κάθε ακροδέκτης (pin) είναι τόσο εισόδου όσο και εξόδου. Το Arduino Uno διαθέτει 20 ακροδέκτες, από τους οποίους 14 είναι ψηφιακοί και 6 είναι αναλογικοί.



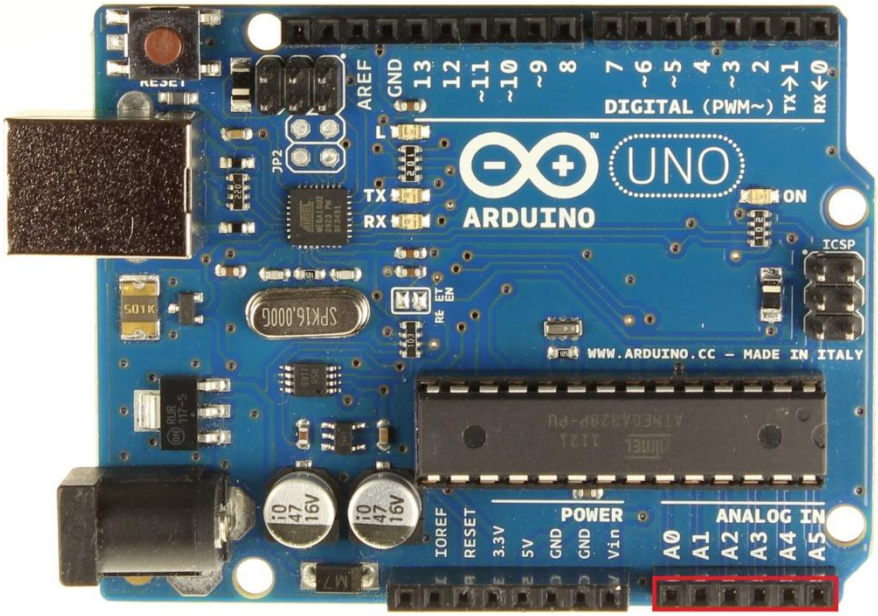
Εικόνα 2.3 : Οι ψηφιακοί ακροδέκτες του Arduino Uno

- Ακροδέκτες 0 και 1: Λειτουργούν ως RX και TX της σειριακής θύρας, όταν το πρόγραμμά ενεργοποιεί τη σειριακή θύρα. Έτσι, όταν το πρόγραμμά στέλνει δεδομένα στη σειριακή θύρα, αυτά προωθούνται και στη θύρα USB μέσω του ελεγκτή Serial-Over-USB, αλλά και στον ακροδέκτη 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή. Αυτό φυσικά σημαίνει, ότι αν στο πρόγραμμά ενεργοποιήσει το σειριακό interface, χάνει 2 ψηφιακές εισόδους/εξόδους η πλατφόρμα.

- Ακροδέκτες 2 και 3: Λειτουργούν και ως εξωτερικά interrupts (interrupt 0 και 1 αντίστοιχα). Ρυθμίζονται μέσα από το πρόγραμμά, ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι, στις οποίες όταν συμβαίνουν συγκεκριμένες

αλλαγές, η κανονική ροή του προγράμματος σταματάει άμεσα και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupts είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

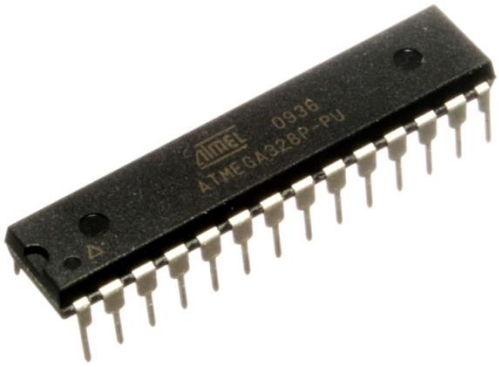
- Ακροδέκτες 3, 5, 6, 9, 10 και 11: Μπορούν να λειτουργήσουν και ως ψευδό-αναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation).



Εικόνα 2.4 :Οι αναλογικοί ακροδέκτες του Arduino Uno

Στην άλλη πλευρά του Arduino, με τη σήμανση ANALOG IN υπάρχει μια ακόμη σειρά από 6 pin, αριθμημένα από το 0 ως το 5. Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V (μεταξύ 2 και 5V) τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με τη σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτηθεί ο ακροδέκτης AREF με 3.3V και στη συνέχεια διαβάσει κάποιον ακροδέκτη αναλογικής εισόδου στο οποίο εφαρμόζεται τάση 1.65V, το Arduino θα επιστρέψει την τιμή 512.

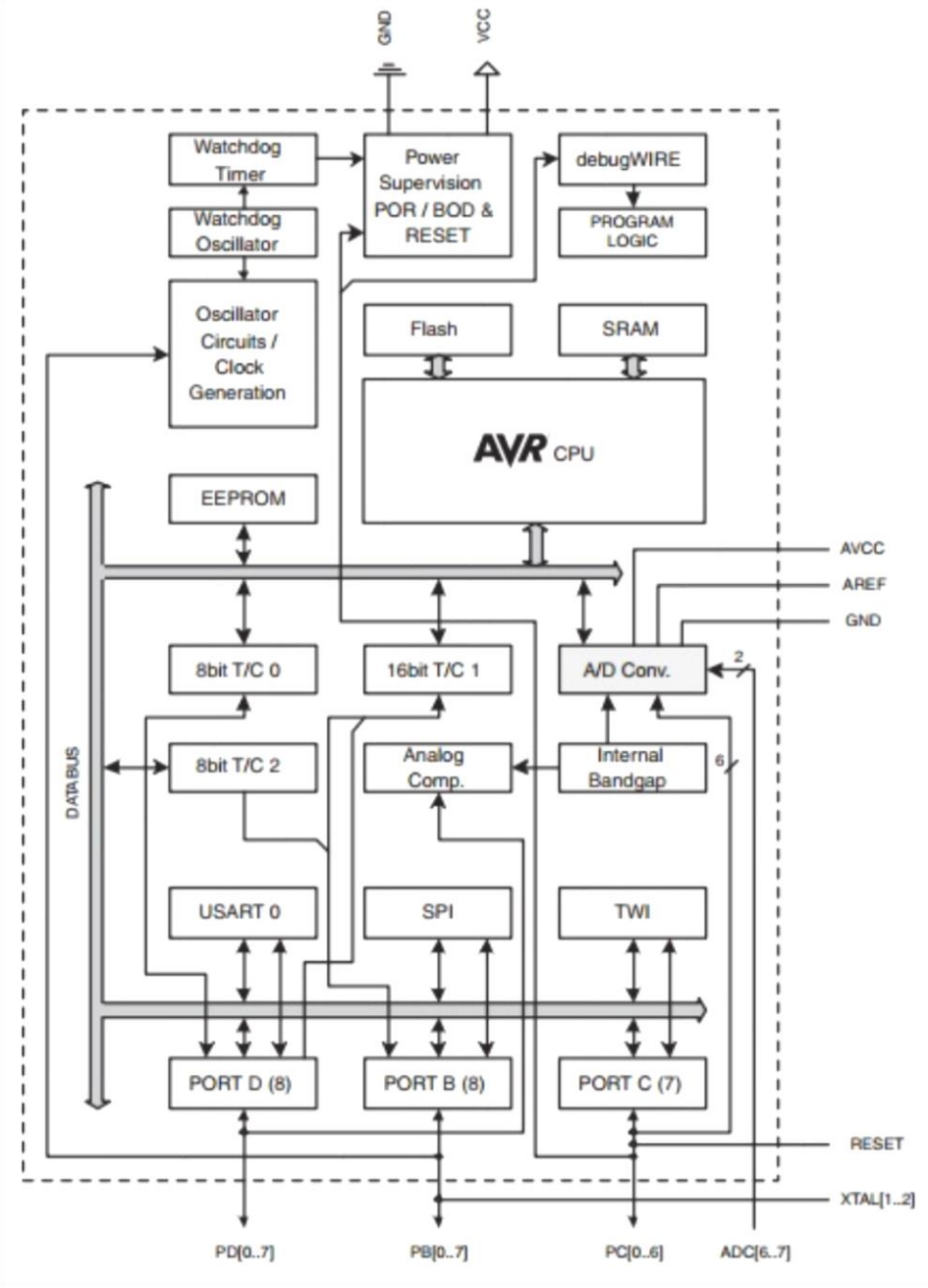
2.2. Ο Μικροεπεξεργαστής ATmega328 [2]



Εικόνα 2.5: Ο μικροεπεξεργαστής ATmega328

Ο ATmega328 είναι ένας μικροελεγκτής CMOS χαμηλής ισχύος 8-bit ο οποίος βασίζεται στην AVR αρχιτεκτονική – μία πιο ενισχυμένη αρχιτεκτονική RISC. Εκτελώντας πανίσχυρες εντολές σε μόνο ένα κύκλο ρολογιού, ο ATmega328 επιτυγχάνει τη διακίνησή τους, φτάνοντας το 1 MIPS ανά MHz, πράγμα που επιτρέπει στο σχεδιαστή του συστήματος να βελτιστοποιήσει την κατανάλωση ενέργειας σε σχέση με την ταχύτητα επεξεργασίας.

Εικόνα 2.6: Ο μικροεπεξεργαστής ATmega328 σε τύπο SMD



Εικόνα 2.7 : Το διάγραμμα Block του ATmega328

Ο πυρήνας AVR συνδυάζει ένα πλούσιο σετ εντολών με 32 καταχωρητές γενικής χρήσης. Όλοι οι καταχωρητές είναι άμεσα συνδεδεμένοι στην Αριθμητική Λογική Μονάδα (Arithmetic Logic Unit – ALU), επιτρέποντας σε δύο ανεξάρτητους καταχωρητές να είναι προσβάσιμοι σε μία εντολή που εκτελείται σε έναν κύκλο ρολογιού. Η αρχιτεκτονική που προκύπτει είναι πιο αποδοτική στον κώδικα καθώς επιτυγχάνει τη μεταφορά εντολών έως και δέκα φορές γρηγορότερα από τους CISC μικροελεγκτές.

2.3. Περιβάλλον Ανάπτυξης Λογισμικού

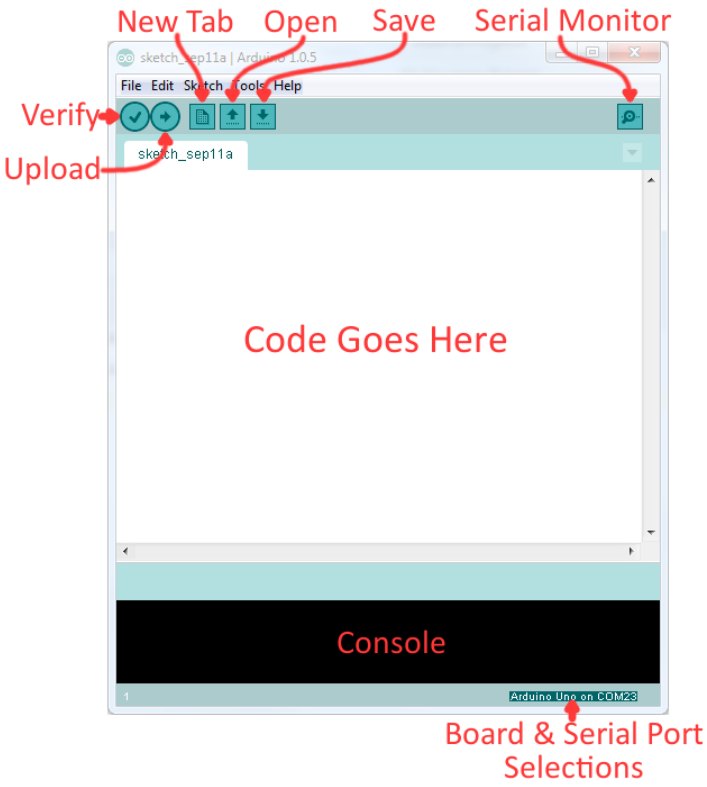
Βασικό στοιχείο της πλατφόρμας του Arduino είναι το Arduino IDE, το οποίο περιέχει το απαραίτητο λογισμικό ώστε να μπορεί ένας υπολογιστής να επικοινωνήσει ή και να ανεβάσει ένα sketch στο σύστημα. Το Arduino IDE είναι μία εφαρμογή με γραφικό περιβάλλον που προσφέρει δυνατότητες επεξεργασίας κώδικα (code editor), μετάφρασης κώδικα (compiler), μεταφορά κώδικα στο Arduino (code uploader) όπως και δυνατότητες διαχείρισης βιβλιοθηκών και αρχείων κώδικα. Είναι γραμμένο σε JAVA, με αποτέλεσμα να μπορεί να «τρέξει» σε Windows, Linux, MAC, ακόμα και σε Android συσκευές.

Υπάρχουν δύο τρόποι εγγραφής κώδικα στο Arduino IDE. Τη γλώσσα του Arduino, μία γλώσσα βασισμένη σε Processing, που υποστηρίζει όλες τις βιβλιοθήκες που έχουν διαμορφωθεί από την κοινότητα του Arduino, δίνοντας τη δυνατότητα ακόμα και σε έναν άπειρο χρήστη να ελέγξει περίπλοκες συσκευές και λειτουργίες. Σε C χαμηλού επιπέδου, δίνοντας τη δυνατότητα πιο γρήγορης εκτέλεσης του προγράμματος (εκτέλεση σε λιγότερους κύκλους μηχανής), αλλά και απευθείας χειρισμού των καταχωρητών του επεξεργαστή, ακόμα και σε λειτουργίες που δεν υποστηρίζει το Arduino IDE (όπως η input capture).

2.3.1. Διαδικασία Uploading

Η διαδικασία του ανεβάσματος ενός sketch στο board, χρησιμοποιώντας το Arduino IDE, περιέχει τα εξής βήματα: Ο κώδικας μετατρέπεται σε C χαμηλού επιπέδου και μετά μεταγλωττίζεται μέσω του avr-gcc (ενός ελευθέρου και ανοιχτού λογισμικού compiler, βασισμένου στον GnuCCompiler – gcc , ειδικά σχεδιασμένο για AVR μικροελεγκτές). Ο παραγόμενος δυαδικός κώδικας αποθηκεύεται στην μνήμη του επεξεργαστή από όπου και να εκτελείται μέσω της USB σύνδεσης. Η σύνδεση αυτή γίνεται μέσω του avrdude, ένα πρόγραμμα που υλοποιεί το πρωτόκολλο επικοινωνίας για την μεταφορά και αποθήκευση του κώδικα στην μνήμη του Arduino.

Ο ATmega328P παρέχεται preburned με ένα bootloader που μας επιτρέπει να φορτώσουμε το νέο κώδικα χωρίς την χρήση εξωτερικού προγραμματιστή υλικού (programmer). Επικοινωνεί χρησιμοποιώντας το πρωτόκολλο STK 500. Μπορεί επίσης να παρακάμψει τον bootloader και το πρόγραμμα του μικροελεγκτή μέσω της ICSP (In-Circuit Serial Programming).



Εικόνα 2.8 : Προβολή του προγράμματος Arduino IDE και των επιλογών του

ΚΕΦΑΛΑΙΟ 3

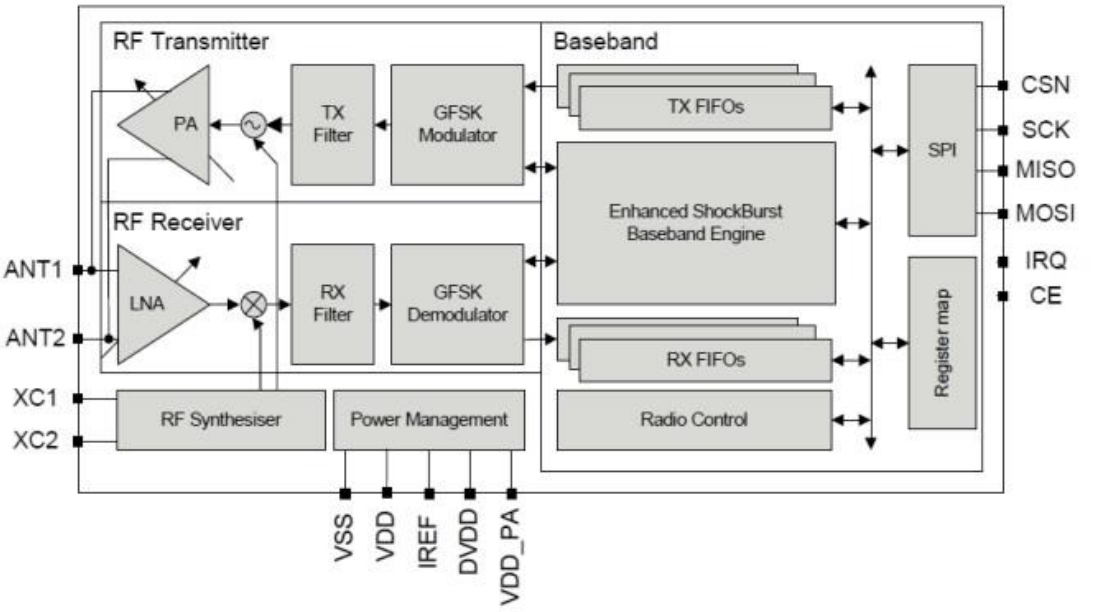
Ο ΑΣΥΡΜΑΤΟΣ ΠΟΜΠΟΔΕΚΤΗΣ NRF24L01[3]

3.1. Γενικές πληροφορίες με τον πομποδέκτη nRF24L01

Το ολοκληρωμένο nRF24L01 είναι ένας πομποδέκτης που λειτουργεί στο εύρος συχνοτήτων 2.400 – 2.4835GHz και διαθέτει ενσωματωμένο πρωτόκολλο βασικής ζώνης, σχεδιασμένο για ασύρματες επικοινωνίες χαμηλής ισχύος. Ο nRF24L01 μπορεί να προγραμματιστεί από ένα μικροεπεξεργαστή μέσω περιφερειακού σειριακής επικοινωνίας (SPI).

Το πρωτόκολλο βασικής ζώνης Enhanced Shock Burst ™ βασίζεται σε επικοινωνία με πακέτα και υποστηρίζει προηγμένους τρόπους λειτουργίας. Εσωτερικές ουρές FIFO επιτρέπουν ομαλή μεταφορά δεδομένων μεταξύ του πομποδέκτη και του μικροϋπολογιστικού συστήματος.

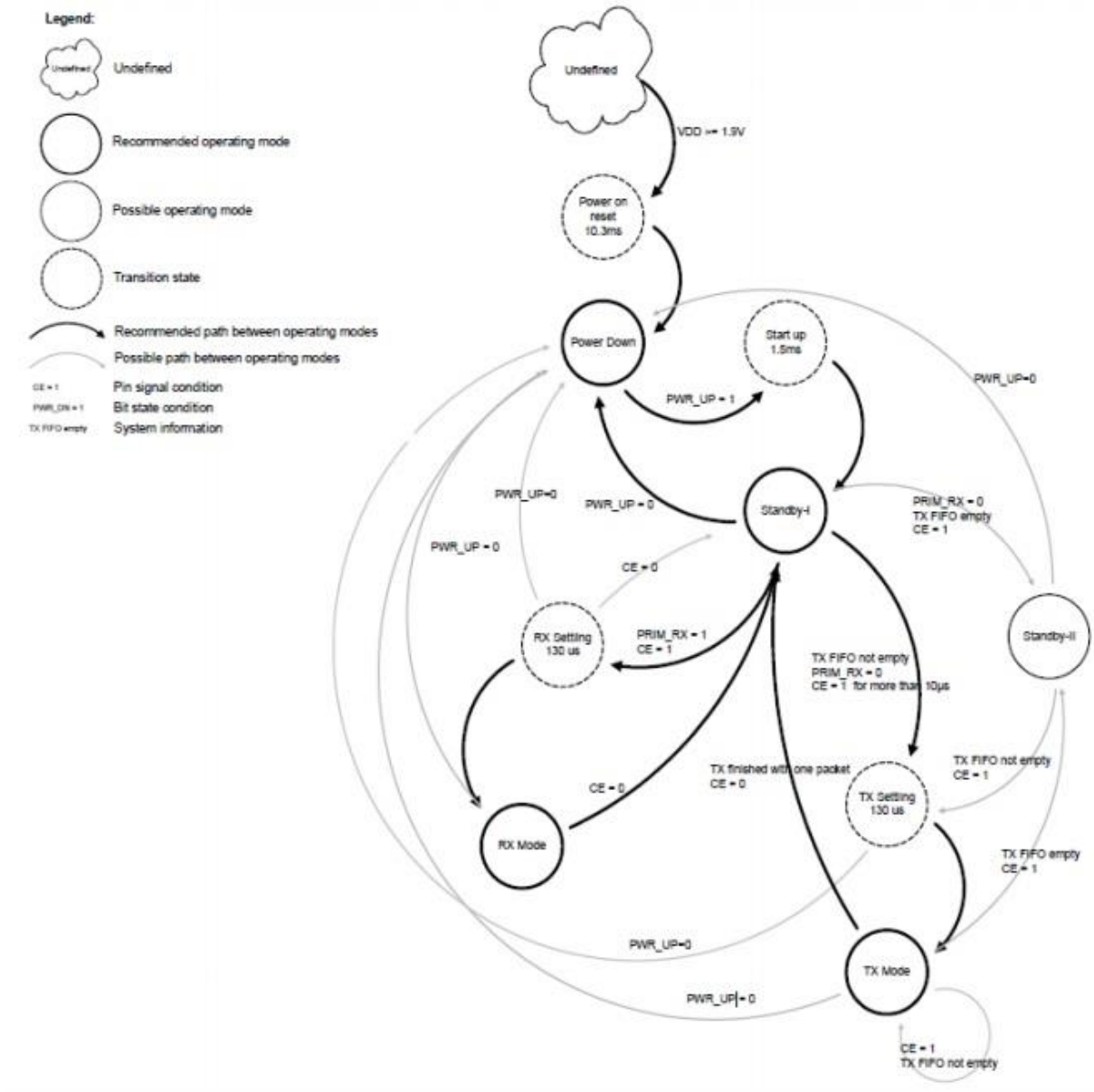
Η διαμόρφωση που χρησιμοποιεί ο nRF24L01 είναι GFSK ενώ δίνεται η δυνατότητα στο χρήστη να αλλάξει παραμέτρους όπως κανάλι συχνότητας, ισχύ εξόδου και ρυθμό μετάδοσης. Ο ρυθμός μετάδοσης είναι παραμετροποιήσιμος μέχρι 2Mbps και συνδυάζεται με δύο τρόπους λειτουργίας χαμηλής κατανάλωσης.



Εικόνα 3.1: Ο πομποδέκτης nRF24l01 και τα βασικά του στοιχεία

3.2. Τρόποι λειτουργίας

Ο nRF24L01 διαθέτει τέσσερις τρόπους λειτουργίας: powerdown, standby, receiver, transmitter.



Εικόνα 3.2: Σχηματική προβολή της λειτουργίας του πομποδέκτηnRF24l01

3.2.1. Κατάσταση απενεργοποίησης – Powerdown

Σε αυτή την κατάσταση ο πομποδέκτης είναι απενεργοποιημένος με ελάχιστη κατανάλωση ενώ οι τιμές των καταχωρητών διατηρούνται. Ο χρόνος ενεργοποίησης είναι 150μs χρησιμοποιώντας εξωτερικό ταλαντωτή. Η μετάβαση σε αυτή την κατάσταση γίνεται θέτοντας το PWR_UP bit του CONFIG καταχωρητή σε χαμηλή στάθμη.

3.2.2. Κατάσταση αναμονής – Standby

Θέτοντας το PWR_UP bit του CONFIG καταχωρητή σε υψηλή στάθμη η συσκευή μεταβαίνει σε κατάσταση standby-I, όπου διατηρείται η χαμηλή κατανάλωση και ο χρόνος εκκίνησης είναι μικρός. Σε αυτή την κατάσταση επιστρέφει από τη λειτουργία TX ή RX όταν το σήμα CE είναι σε χαμηλή στάθμη. Στην κατάσταση standby-II η κατανάλωση είναι αυξημένη, ενώ βρίσκεται σε αυτή όταν η συσκευή λειτουργεί ως πομπός και το σήμα CE είναι σε υψηλή στάθμη. Έτσι, αν η ουρά εκπομπής είναι άδεια και μεταφερθεί ένα πακέτο, θα ξεκινήσει αμέσως η εκπομπή του.

3.2.3. Κατάσταση λήψης – Receiver (RX)

Στο τρόπο λειτουργίας RX ο nRF24L01 λειτουργεί ως δέκτης όταν το PWR_UP και το PRIM_RX bit του CONFIG καταχωρητή καθώς και το CE είναι σε υψηλή στάθμη. Σε αυτή την κατάσταση αποδιαμορφώνει τα σήματα από το κανάλι ενώ το πρωτόκολλο βασικής ζώνης συνεχώς αναζητά έγκυρο πακέτο (αντίστοιχη διεύθυνση και έγκυρο CRC). Αν βρεθεί έγκυρο, τα δεδομένα τοποθετούνται σε μια κενή θέση στην ουρά 47 λήψης. Αν δεν βρεθεί, το πακέτο απορρίπτεται. Επιπλέον, είναι δυνατή η ανίχνευση φέροντος, όπου αν υπάρχει σήμα για τουλάχιστον 128μs, τότε το CD bit τίθεται σε υψηλή στάθμη.

3.2.4. Κατάσταση μετάδοσης – Transmitter (TX)

Στο τρόπο λειτουργίας TX ο nRF24L01 λειτουργεί ως πομπός όταν το PWR_UP είναι σε υψηλή στάθμη και το PRIM_RX bit σε χαμηλή, ενώ υπάρχουν δεδομένα στην ουρά εκπομπής και εφαρμοστεί παλμός στο CE διάρκειας 10μs. Μετά το τέλος της εκπομπής επιστρέφει σε standby-I, αν CE=0. Στην περίπτωση που CE=1, αν υπάρχουν δεδομένα για αποστολή θα παραμείνει σε κατάσταση TX, αλλιώς θα επιστρέψει σε κατάσταση standby-II.

Mode	PWR_UP register	PRIM_RX register	CE	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO. Will empty all levels in TX FIFO[a].
TX mode	1	0	minimum 10μs high pulse	Data in TX FIFO.Will empty one level in TX FIFO[b].
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

Πίνακας 3.1: Πίνακας καταστάσεων του πομποδέκτη nRF24l01

3.3. Διεπαφές δεδομένων και ελέγχου

Η συμπεριφορά του nRF24L01 προγραμματίζεται μέσω της διεπαφής SPI παρέχοντας έτσι πρόσβαση σε όλους τους 29 καταχωρητές που ρυθμίζουν και ελέγχουν τη λειτουργία του. Ο μέγιστος ρυθμός δεδομένων που υποστηρίζει η διεπαφή SPI του πομποδέκτη είναι 8Mbps με τη μεταφορά δεδομένων και σημάτων ελέγχου μεταξύ του πομποδέκτη και της MCU να γίνεται μέσω έξι σημάτων: τέσσερα που αφορούν την διεπαφή SPI (CSN, SCK, MOSI, MISO), το σήμα CE για επιλογή μεταξύ RX και TX λειτουργίας και το σήμα διακοπής IRQ που ελέγχεται από τρεις πηγές διακοπών. Η ρύθμιση και η εξαγωγή των τιμών των παραμέτρων που καθορίζουν τη λειτουργία του πομποδέκτη γίνεται μέσω εντολών. Το σύνολο εντολών αποτελείται από λέξεις μήκους οκτώ bit και κάθε νέα εντολή ξεκινά με μία μετάβαση από υψηλή σε χαμηλή στάθμη στην ακίδα CSN. Παράλληλα με τη μεταφορά της εντολής μέσω του σήματος στην ακίδα MOSI, ο πομποδέκτης απαντά με τα περιεχόμενα του STATUS καταχωρητή στην ακίδα MISO. Η μετάδοση των εντολών γίνεται από το περισσότερο σημαντικό bit προς το λιγότερο σημαντικό, και τα δεδομένα μεταδίδονται από το λιγότερο σημαντικό byte προς το πιο σημαντικό και κάθε byte με πρώτο το σημαντικότερο bit. Το σήμα διακοπής μεταβαίνει από υψηλή σε χαμηλή στάθμη (active low) όταν μία από τρεις διακοπές TX_DS (αποστολή δεδομένων), RX_DR (λήψη δεδομένων) ή MAX_RT (μέγιστος αριθμός επανεκπομπών) ενεργοποιηθεί από τη μηχανή κατάστασης στον STATUS καταχωρητή. Αντίθετα, απενεργοποιείται γράφοντας ‘1’ στο αντίστοιχο bit του 48 καταχωρητή STATUS. Η ενεργοποίηση των διακοπών ελέγχεται από μία μάσκα στον καταχωρητή CONFIG – θέτοντας ‘1’ η αντίστοιχη διακοπή απενεργοποιείται.

3.4. Σύνολο εντολών SPI

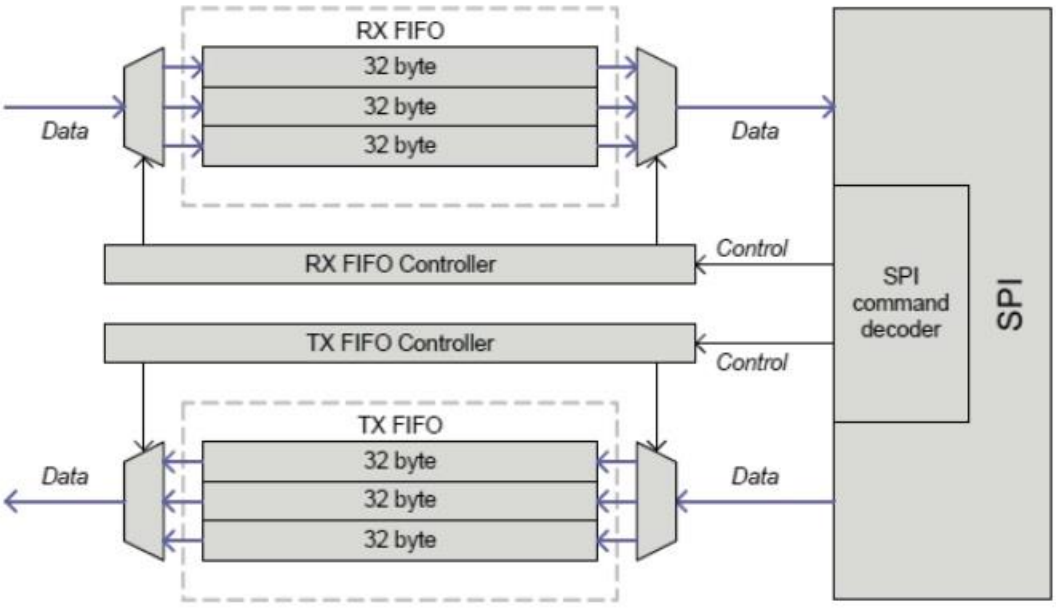
Το σύνολο εντολών του nRF24L01 αποτελείται από δώδεκα εντολές, οι οποίες χρησιμοποιούνται τόσο για την ρύθμιση των παραμέτρων λειτουργίας όσο και για τη μεταφορά δεδομένο από και προς τον πομποδέκτη. Παρακάτω ακολουθεί μια συνοπτική παρουσίαση των εντολών:

Όνομα Εντολής	Κωδική λέξη	Πλήθος δεδομένων	Λειτουργία
R_REGISTER	000A AAAA	1 έως 5 bytes	Ανάγνωση περιεχομένων του καταχωρητή στη διεύθυνση AAAAA.
W_REGISTER	001A AAAA	1 έως 5 bytes	Εγγραφή στον καταχωρητή με διεύθυνση AAAAA.
R_RX_PAYLOAD	0110 0001	1 έως 32 bytes	Ανάγνωση περιεχομένων φορτίου εισερχόμενου πακέτου από ουρά λήψης. Μετά την ανάγνωση το φορτίο διαγράφεται από την ουρά λήψης.
W_TX_PAYLOAD	1010 0000	1 έως 32 bytes	Εγγραφή περιεχομένων φορτίου εξερχόμενου πακέτου στην ουρά αποστολής.
FLUSH_TX	1110 0001	0	Άδειασμα ουράς αποστολής
FLUSH_RX	1110 0010	0	Άδειασμα ουράς λήψης. Δεν πρέπει να εκτελείται όταν μεταδίδεται πακέτο επιβεβαίωσης.
REUSE_TX_PL	1110 0011	0	Συνεχόμενη εκπομπή του πιο πρόσφατου πακέτου όσο το σήμα CE είναι σε υψηλή στάθμη. Η λειτουργία απενεργοποιείται με την εντολή W_TX_PAYLOAD ή FLUSH_TX.
ACTIVATE	0101 0000	1	Ακολουθούμενη από δεδομένο τιμής 0x73 ενεργοποιεί τις εξής λειτουργίες: ▪ R_RX_PL_WID ▪ W_ACK_PAYLOAD ▪ W_TX_PAYLOAD_NOACK Νέα αποστολή της ίδιας εντολής με τα ίδια δεδομένα απενεργοποιεί τις παραπάνω λειτουργίες.
R_RX_PL_WID	0110 0000	0	Εξαγωγή μήκους φορτίου του εισερχόμενου πακέτου στην κορυφή της ουράς λήψης
W_ACK_PAYLOAD	1010 1PPP	1 έως 32 bytes	Εγγραφή φορτίου που θα μεταδοθεί μαζί με την επόμενη επιβεβαίωση στο κανάλι (pipe) PPP.
W_TX_PAYLOAD_NOACK	1011 0000	1 έως 32 bytes	Αποστολή αυτού του πακέτου χωρίς λειτουργία αυτόματης επιβεβαίωσης.
NOP	1111 1111	0	Καμία λειτουργία. Μπορεί να χρησιμοποιηθεί για την ανάγνωση του STATUS καταχωρητή.

Πίνακας 3.2: Το σύνολο των εντολών που χρησιμοποιεί ο πομποδέκτης nRF24l01

3.5. Ουρές δεδομένων

Οι ουρές δεδομένων είναι τύπου FIFO (First In – First Out) και χρησιμοποιούνται από τον nRF24L01 για την αποθήκευση των φορτίων που πρέπει να αποσταλούν (Tx FIFO) ή έχουν ληφθεί (Rx FIFO). Τόσο η ροή της λήψης όσο και η ροή της αποστολής αποτελείται από μία ουρά τριών θέσεων για την αποθήκευση φορτίων μέγιστου μήκους 32 bytes. Κάθε ουρά έχει ένα ξεχωριστό ελεγκτή, ενώ είναι προσβάσιμη μέσω της διεπαφής SPI χρησιμοποιώντας τις κατάλληλες εντολές. Όταν η συσκευή λειτουργεί ως δέκτης η ουρά αποστολής μπορεί να αποθηκεύσει φορτία για ACK πακέτα προς τρεις διαφορετικές συσκευές που λειτουργούν ως πομποί. Όμως, η ουρά αποστολής μπορεί να μπλοκάρει αν όλα τα πακέτα έχουν προορισμό συσκευή με την οποία η σύνδεση έχει χαθεί. Σε αυτή την περίπτωση, η μονάδα επεξεργασίας (MCU) μπορεί να αδειάσει την ουρά αποστολής με την εντολή FLUSH_TX. Εγγραφή στην ουρά αποστολής μπορεί να γίνει με μία από τις εντολές W_TX_PAYLOAD ή W_TX_PAYLOAD_NO_ACK στην λειτουργία ως πομπός και την εντολή W_ACK_PAYLOAD στην λειτουργία ως δέκτης. Αντίστοιχα, η ανάγνωση της ουράς λήψης γίνεται με την εντολή R_RX_PAYLOAD. Επίσης, το φορτίο της ουράς αποστολής σε έναν πομπό δεν αφαιρείται αν έχει επιτευχθεί ο μέγιστος αριθμός επαναλήψεων μετάδοσης (δηλαδή συμβεί διακοπή MAX_RT).



Εικόνα 3.3: Τρόπος μετάδοσης δεδομένων του πομποδέκτη nRF24l01

Η κατάσταση των δύο ουρών δεδομένων είναι προσβάσιμη μέσω του καταχωρητή κατάστασης FIFO_STATUS. Χρησιμοποιώντας την εντολή R_REGISTER μπορούν να εξαχθούν τα περιεχόμενα του καταχωρητή ελέγχοντας έτσι αν οι ουρές δεδομένων περιέχουν φορτία ή όχι.

3.6. Φυσικό κανάλι

Ο πομποδέκτης nRF24L01 επιτρέπει τη ρύθμιση παραμέτρων του φυσικού καναλιού, όπως το ρυθμό δεδομένων, τη συχνότητα του καναλιού και την ισχύ εκπομπής.

3.7. Ρυθμός δεδομένων

Ο ρυθμός δεδομένων στον αέρα (air data rate) είναι ο ρυθμός του διαμορφωμένου σήματος κατά τη μετάδοση και τη λήψη. Μπορεί να είναι είτε στα 1Mbps ή 2Mbps, με 3db μεγαλύτερη ευαισθησία όταν χρησιμοποιείται ρυθμός 1Mbps. Χρησιμοποιώντας υψηλό ρυθμό δεδομένων μειώνονται οι συγκρούσεις πάνω από το ασύρματο κανάλι καθώς και η κατανάλωση ισχύος. Ο ρυθμός δεδομένων επιλέγεται από το RF_DR bit του RF_SETUP καταχωρητή. Για να επικοινωνήσουν μεταξύ τους ένας πομπός και ένας δέκτης θα πρέπει να προγραμματιστούν με τον ίδιο ρυθμό δεδομένων.

3.8. Συχνότητα του καναλιού

Η ραδιοσυχνότητα του καναλιού καθορίζει το κέντρο του καναλιού που χρησιμοποιεί ο nRF24L01. Το εύρος του καναλιού είναι 1MHz για ρυθμό δεδομένων 1Mbps και 2MHz για ρυθμό 2Mbps. Ο πομποδέκτης μπορεί να λειτουργήσει σε συχνότητες από 2,4GHz μέχρι 2,525GHz, ενώ η ακρίβεια της επιλογής του καναλιού είναι 1MHz. Στην περίπτωση που ο ρυθμός δεδομένων είναι 2Mbps το κανάλι δεσμεύει μεγαλύτερο εύρος ζώνης από την ακρίβεια της

ρύθμισης· για να μην υπάρχουν επικαλυπτόμενα κανάλια, η απόσταση μεταξύ τους θα πρέπει να είναι μεγαλύτερη από 2MHz. Η ρύθμιση της συχνότητας του χρησιμοποιούμενου καναλιού γίνεται από τον RF_CH καταχωρητή σύμφωνα με τη σχέση: $F0 = 2400 + RF\_CH$ $[MHz]$. Για να υπάρξει επικοινωνία μεταξύ πομπού και δέκτη θα πρέπει να έχουν την ίδια συχνότητα καναλιού.

3.9. Ενισχυτής ισχύος

Η ρύθμιση του ενισχυτή ισχύος (power amplifier -PA) καθορίζει την ισχύ εξόδου του nRF24L01. Όταν βρίσκεται σε λειτουργία εκπομπής, η ισχύς μπορεί να ρυθμιστεί μεταξύ τεσσάρων επιλογών, όπως φαίνεται στην παρακάτω εικόνα. Η ρύθμιση της ισχύος γίνεται από τα αντίστοιχα RF_PWR bits του RF_SETUP καταχωρητή.

SPI RF-SETUP (RF_PWR)	RF output power	DC current consumption
11	0dBm	11.3mA
10	-6dBm	9.0mA
01	-12dBm	7.5mA
00	-18dBm	7.0mA

Πίνακας 3.3: Ρύθμιση εκπεμπόμενης ισχύος

3.10. Πρωτόκολλο Enhanced Shock Burst ™

Το Enhanced Shock Burst ™ είναι πρωτόκολλο επιπέδου ζεύξης που βασίζεται σε πακέτα. Προσφέρει αυτόματο σχηματισμό πακέτων και χρονισμούς καθώς και αυτόματη μετάδοση πακέτων επιβεβαίωσης (acknowledgement packets) και αναμεταδόσεις πακέτων (re-transmissions). Τα κύρια χαρακτηριστικά του είναι δυναμικό μήκος ωφέλιμου φορτίου (payload) 1-32 bytes, η αυτόματη διαχείριση των συναλλαγών των πακέτων και λειτουργία ως multiceiver σε τοπολογία αστέρα 1:6.

Μία συναλλαγή πακέτων είναι μία ανταλλαγή πακέτων μεταξύ δύο πομποδεκτών όπου ο ένας λειτουργεί ως κύριος δέκτης (Primary Receiver – PRX) και ο άλλος ως κύριος πομπός (Primary Transmitter – PTX). Η συναλλαγή ξεκινά πάντα με τη μετάδοση πακέτου από τον PTX και ολοκληρώνεται όταν ο PTX λάβει πακέτο επιβεβαίωσης (ACK packet) από τον PRX. Αφού ο PTX στείλει πακέτο στον PRX, ο PTX τίθεται σε λειτουργία δέκτη (receive mode) και περιμένει για πακέτο επιβεβαίωσης. Αν το πακέτο ληφθεί από τον PRX, το πρωτόκολλο μεταδίδει ένα πακέτο επιβεβαίωσης στον PTX και επιστρέφει σε λειτουργία λήψης. Αν ο PTX δεν λάβει το ACK πακέτο μέσα σε ένα χρονικό παράθυρο, αυτόματα αναμεταδίδει τα δεδομένα στον PRX και περιμένει για ACK πακέτο. Η συσκευή που λειτουργεί ως PRX μπορεί μαζί με το ACK πακέτο να στέλνει και δεδομένα δημιουργώντας μια δικατευθυντήρια ζεύξη δεδομένων. Επιπλέον, μπορεί να καθοριστούν από το χρήστη παράμετροι όπως μέγιστος αριθμός αναμεταδόσεων και καθυστέρηση μεταξύ των αναμεταδόσεων.

3.11. Μορφότυπο πλαισίων

Κάθε πακέτο του πρωτοκόλλου περιλαμβάνει ένα προοίμιο, ένα πεδίο διεύθυνσης, ένα πεδίο ελέγχου, το ωφέλιμο φορτίο και πεδίο κυκλικού ελέγχου πλεονασμού (CRC).

Εικόνα 3.4: Η δομή της πληροφορίας που στέλνεται μέσω του πομποδέκτη nRF24l01

Το προοίμιο είναι μια ακολουθία ενός byte που χρησιμοποιείται για την ανίχνευση της στάθμης των 0 και 1. Αν η διεύθυνση ξεκινάει με 0, ως προοίμιο τοποθετείται το 01010101 αλλιώς το 10101010. Η διεύθυνση είναι η διεύθυνση του δέκτη, μπορεί να έχει μήκος από 3 έως 5 bytes και προγραμματίζεται από τον SETUP_AW καταχωρητή. Το πεδίο ελέγχου αποτελείται από 6 bits που δείχνουν τον αριθμό των bytes στο ωφέλιμο φορτίο (000000 για 0 byte και 100000 για 32 bytes), 2 bits που αντιστοιχούν στην ταυτότητα του πακέτου (packet identification – PID) που 53 χρησιμοποιούνται για να αναγνωρίζει ο δέκτης αν το πακέτο είναι καινούριο ή έχει αναμεταδοθεί και μία σημαία ενός bit που όταν είναι 1 δηλώνει ότι το πακέτο δεν θα επιβεβαιωθεί (NO_ACK flag). Το περιεχόμενο του ωφέλιμου φορτίου καθορίζεται από το χρήστη και το μήκος του μπορεί να είναι από 0 μέχρι 32 bytes. Το πεδίο CRC αποτελείται από 1 ή δύο bytes και υπολογίζεται από τα πεδία διεύθυνσης, ελέγχου και ωφέλιμου φορτίου. Για 1 byte χρησιμοποιείται το πολυώνυμο X 8+X 2+X+1 με αρχική τιμή 0xFF, ενώ για 2 bytes το X 16 + X 12 + X 5 + 1 με αρχική τιμή 0xFFFF. Κανένα πακέτο δεν γίνεται δεκτό αν ο έλεγχος αποτύχει.

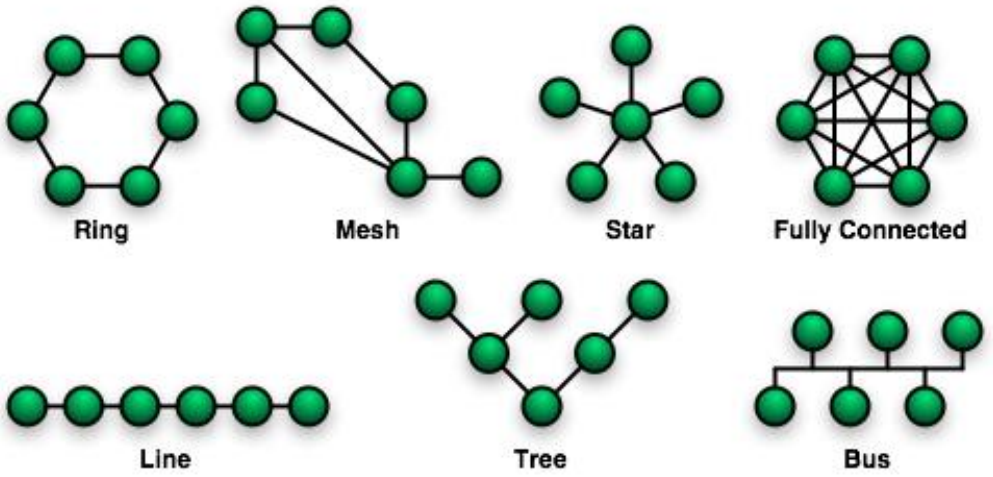
ΚΕΦΑΛΑΙΟ 4

ΤΟΠΟΛΟΓΙΑ ΔΙΚΤΥΟΥ

4.1. Τοπολογίες δικτύων

Τοπολογία δικτύου ονομάζεται ο τρόπος που συνδέονται μεταξύ τους οι κόμβοι ενός δικτύου. Οι τοπολογίες είναι είτε φυσικές, είτε λογικές και τα κυριότερα είδη που υπάρχουν είναι τα εξής:

- Γραμμική
- Τύπου διαύλου
- Δακτυλίου
- Αστέρα
- Πλέγματος
- Πλήρους πλέγματος
- Τύπου δέντρου



Εικόνα 4.1: Διαγράμματα μερικών από τις πιο γνωστές τοπολογίες

4.2. Τοπολογία Πλήρους ή Μερικού Πλέγματος (Mesh Topology)

Σε ένα δίκτυο πλήρους πλέγματος, ή πλήρους διασύνδεσης, ή πλήρους δικτύωσης, για κάθε ζεύγος κόμβων που υπάρχουν στο δίκτυο, υπάρχει και ένα κανάλι που τους συνδέει. Κάθε κόμβος, δηλαδή, έχει άμεση σύνδεση με κάθε άλλο κόμβο.

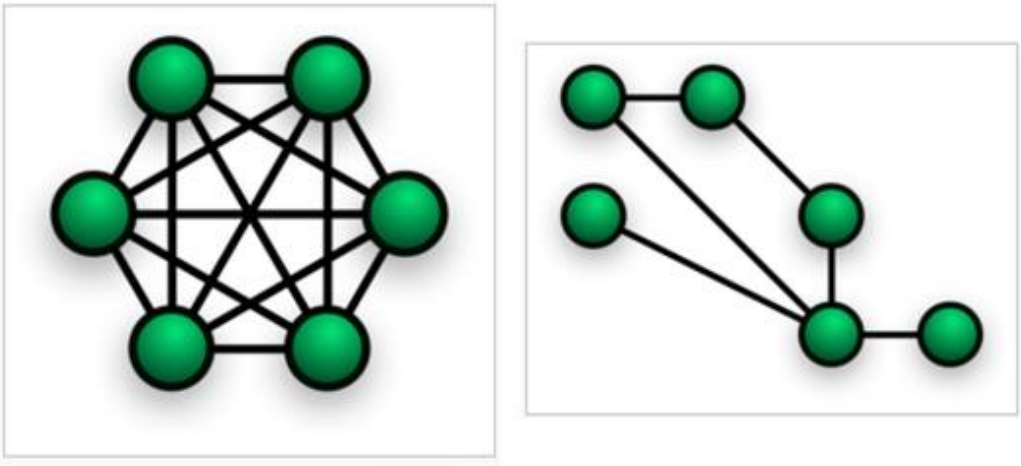
Πιο συγκεκριμένα, εάν υπάρχουν χκόμβοι στο δίκτυό μας, ο κάθε κόμβος θα έχει x-1 συνδέσεις, ενώ ο αριθμός των συνολικών καναλιών θα είναι x*(x-1)/2. Παρακάτω παραθέτονται τα πλεονεκτήματα και τα μειονεκτήματα του δικτύου πλήρους διασύνδεσης [4]:

- Πλεονεκτήματα:
  - Τα προβλήματα κυκλοφορίας δεδομένων είναι ελάχιστα έως και μηδαμινά
  - Παρέχεται μέγιστη ασφάλεια
  - Σε περίπτωση που μία γραμμή αχρηστευθεί, δεν αχρηστεύεται όλο το σύστημα
  - Γίνεται εύκολη ανίχνευση και απομόνωση σφαλμάτων
  - Προκειμένου να αποφευχθούν οι αχρηστεμένες γραμμές υπάρχει δυνατότητα επαναδρομολόγησης

- Μειονεκτήματα:
  - Το κόστος υλοποίησης είναι υψηλό έως και απαγορευτικό
  - Υπάρχουν δυσκολίες κατά την εγκατάσταση του συστήματος λόγω των πολλών καλωδιώσεων και της στενότητας χώρων

Ο συνδυασμός των τοπολογιών πλήρους πλέγματος και αστέρα, δημιουργεί την τοπολογία μερικού πλέγματος η οποία μειώνει τον αριθμών των καναλιών. Πλέον, οι παραπάνω τύποι υπολογισμού καναλιών δε θα μπορούν να ισχύουν καθώς ο κάθε κόμβος δεν είναι υποχρεωτικό να συνδέεται άμεσα με όλους τους υπόλοιπους.

Μία τοπολογία πλέγματος μπορεί να είναι πλήρης ή μερική. Στην συγκεκριμένη περίπτωση χρησιμοποιείται μία μορφή μερικού πλέγματος τοπολογίας η οποία αναλύεται στο επόμενο υποκεφάλαιο.



Εικόνα 4.2: Τοπολογία πλέγματος (Αριστερά πλήρους διασύνδεσης, Δεξιά μερικής διασύνδεσης)

4.3. Δικτύωση Πλέγματος για το ραδιοπομπό RF24 [7]

Η βιβλιοθήκη RF24Network παρέχει ένα σύστημα κατανομής διευθύνσεων και δρομολόγησης για τους ραδιοπομπούς RF24 που επιτρέπει να κατασκευαστούν μεγάλα δίκτυα ασύρματων αισθητήρων. Το δίκτυο πλέγματος RF24 παρέχει αναβαθμισμένες λειτουργίες, συμπεριλαμβανομένων και της αυτόματης κατανομής διευθύνσεων και της δυναμικής διαμόρφωσης των ασύρματων αισθητήρων.

Ο κόμβος master παρακολουθεί τις IDs των υπολοίπων κόμβων και τις διευθύνσεις που έχουν κατανεμηθεί από το RF24 δίκτυο. Όταν ένας κόμβος κινηθεί, ή όταν απλά χάσει τη σύνδεσή του με το δίκτυο, μπορεί να ρυθμιστεί ώστε να επανέλθει αυτόματα στο πλέγμα και να αναδιαμορφωθεί εντός του δικτύου.

Σε κάθε κόμβο έχει εκχωρηθεί ένας μοναδικός αριθμός από το 1 έως το 255, και οτιδήποτε άλλο διαχειρίζεται η βιβλιοθήκη (όπως η διεύθυνση, η δρομολόγηση, κλπ.)

Στη διαμόρφωση πλέγματος, οι απλοί κόμβοι μπορούν να κινούνται μακριά από τον master, χρησιμοποιώντας τους άλλους κόμβους για τη δρομολόγηση σε μακρινές αποστάσεις. Η κατανομή διευθύνσεων και η τοπολογία μπορούν να αναδιαμορφωθούν ενώ οι συνδέσεις έχουν κοπεί και επανεγκατασταθεί μέσα σε διαφορετικές περιοχές του δικτύου. Επιπλέον, οι κατανεμηθείσες διευθύνσεις δεν αποθηκεύονται. Εάν ο κύριος κόμβος για κάποιο λόγο βγει εκτός δικτύου, όλοι οι κόμβοι πρέπει να επανασυνδεθούν στο πλέγμα ή να επαννεκινηθούν προκειμένου να αποφευχθεί η «σύγκρουσή» τους

Σήμερα, χρησιμοποιείται πολύ η βασική λειτουργία πλέγματος σε συνδυασμό με το Arduino σαν το master κόμβο, ο οποίος λαμβάνει και στέλνει δεδομένα από και προς τους κόμβους.

ΚΕΦΑΛΑΙΟ 5

ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΣΙΑΣ – ΕΞΑΡΤΗΜΑΤΑ ΚΑΙ ΑΙΣΘΗΤΗΡΕΣ

Στα παρακάτω υποκεφάλαια παρουσιάζονται τα εξαρτήματα και οι αισθητήρες που χρησιμοποιήθηκαν για να υλοποιηθεί το πρακτικό μέρος της εργασίας:

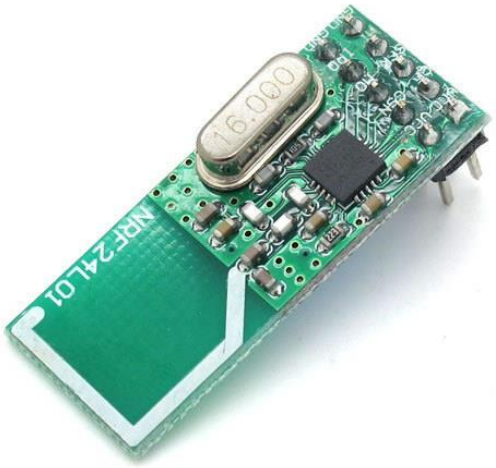
5.1. Arduino Uno

ToArduino Uno είναι η υπολογιστική πλατφόρμα, που όπως αναφέρθηκε και στο Κεφάλαιο 2, αποτελείται από τον μικροελεγκτή ATmega328.

Εικόνα 5.1: Η πλακέτα του Arduino Uno

5.2. Πομποδέκτης nRF24L01

Το ολοκληρωμένο nRF24L01 που περιγράφηκε αναλυτικά στο Κεφάλαιο 3.

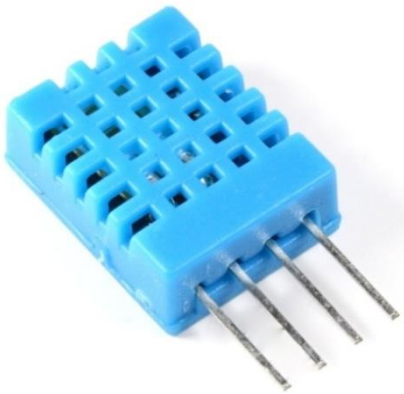


Εικόνα 5.2: Η πλακέτα του πομποδέκτη nRF24L01

5.3. Αισθητήρας υγρασίας και θερμοκρασίας DHT11

Ψηφιακός αισθητήρας θερμοκρασίας και υγρασίας. Ανιχνεύει την θερμοκρασία και την υγρασία του περιβάλλοντα χώρου και τα αποτελέσματα τα βγάζει σε ψηφιακή έξοδο. Τα δυο πρώτα pinstou είναι η παροχή ηλεκτρικού ρεύματος και η γείωση και είναι αυτά που τροφοδοτούν τον αισθητήρα, ενώ το τρίτο είναι το σήμα εξόδου.

Το πολύ μικρό του μέγεθος και το μικρό του βάρος το κάνουν ιδανική επιλογή για υλοποίηση μικρών ρομπότ ή και για συστήματα παρακολούθησης και μελέτης του περιβάλλοντος.



Εικόνα 5.3: Αισθητήρας θερμοκρασίας και υγρασίας DHT11

5.4. Αισθητήρας θερμοκρασίας lm35

Ο LM35 είναι ένας αισθητήρας θερμοκρασίας που μετρά θερμοκρασίες από 0 έως 100^{o}C. Το πρώτο του pinσυνδέεται με τάση 5V και το τρίτο με τη γείωση, ενώ το μεσαίο δίνει το σήμα εξόδου του. Χρησιμοποιήθηκε σε κάποιους κόμβους αντί του DHT11 λόγω της σχεδίασης της πλακέτας.

Εικόνα 5.4 : Ο αισθητήρας θερμοκρασίας LM35

5.5. Αισθητήρας κίνησης

Αισθητήρας κίνησης. Ο αισθητήρας αυτός είναι κατασκευασμένος από ένα πυροηλεκτρικό αισθητήρα, ο οποίος μπορεί να ανιχνεύσει επίπεδα υπέρυθρης ακτινοβολίας (τα πάντα εκπέμπουν υπέρυθρη ακτινοβολία και όσο πιο θερμό είναι ένα σώμα, τόσο περισσότερη ακτινοβολία εκπέμπει) και βγάζει τα αποτελέσματα σε ψηφιακή έξοδο.

Ανιχνεύουν κίνηση από περίπου 600 εκατοστά. Έχουν ρυθμιζόμενη καθυστέρηση πριν δώσουν το θετικό σήμα κίνησης και ρυθμιζόμενη ευαισθησία. Λειτουργεί με τροφοδοσία 5V και το ψηφιακό σήμα εξόδου του είναι 3.3V. Τέλος, η εμβέλειά του φτάνει περίπου τα 7 μέτρα (120^{o}).

Εικόνα 5.5: Αισθητήρας κίνησης PIR

5.6. Αισθητήρας φωτός LDR

Αισθητήρας φωτός. Στην πραγματικότητα είναι ένας μετρητής φωτός καθώς «βλέπει» την ποσότητα του φωτός που υπάρχει στο χώρο. Τα αποτελέσματά του βγαίνουν σε αναλογική έξοδο.

Εικόνα 5.6: LDR Αισθητήρας φωτός

5.7. Πληκτρολόγιο 4x4

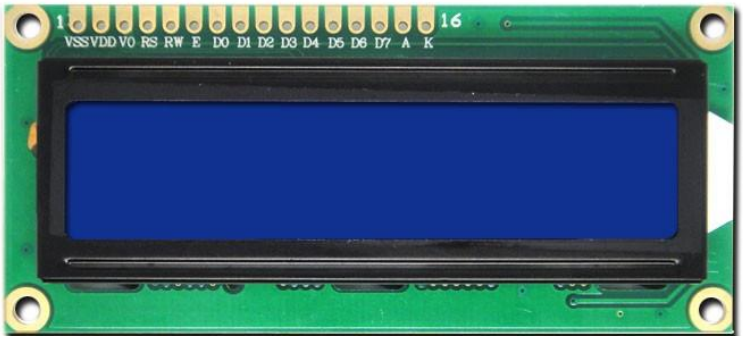
Πληκτρολόγιο 16 πλήκτρων:

- 10 αριθμών – 0 έως 9
- 4 γραμμάτων – A, B, CκαιD
- 2 συμβόλων – * και #

Εικόνα 5.7: Πληκτρολόγιο 4x4

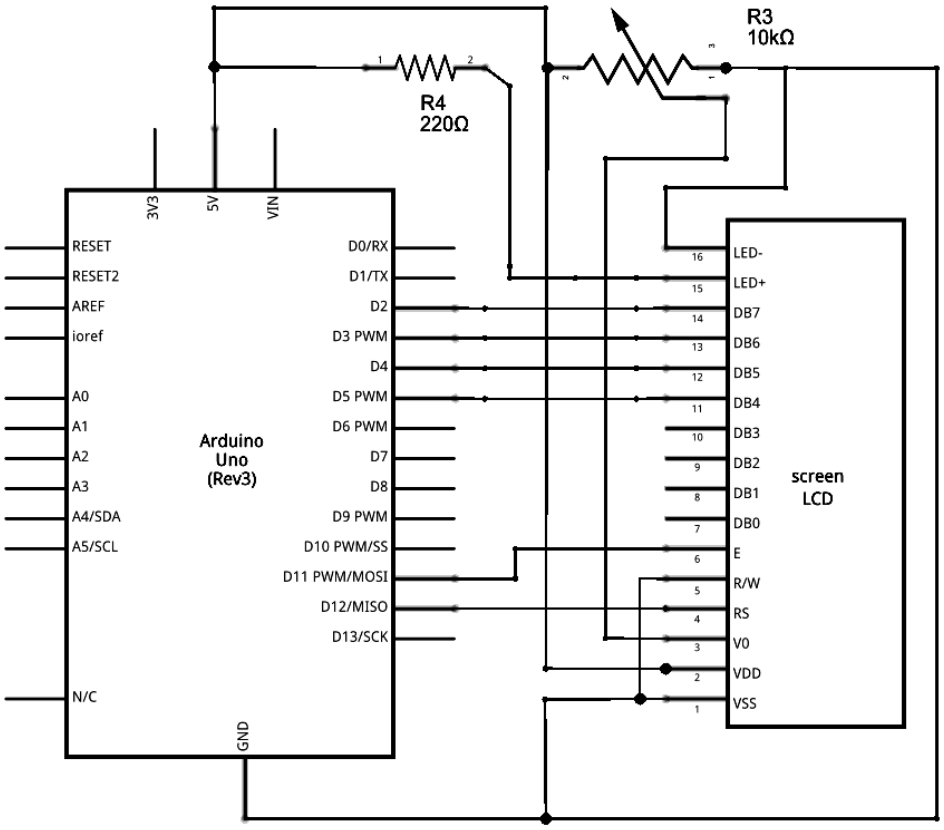
5.8. LCD οθόνη 16x2

Οθόνη δύο γραμμών και 16 χαρακτήρων ανά γραμμή. Έχει μπλε φόντο και λευκούς χαρακτήρες και στην παρούσα πτυχιακή θα αναγράφει την κατάσταση του συναγερμού καθώς και τις καταστάσεις κάποιων λειτουργιών που γίνονται στο σπίτι.



Εικόνα 5.8 : Οθόνη LCD 16x2

Έχει 16 pins τα οποία συνδέονται όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 5.9: Συνδεσμολογία LCD οθόνης 16x2 με το Arduino Uno

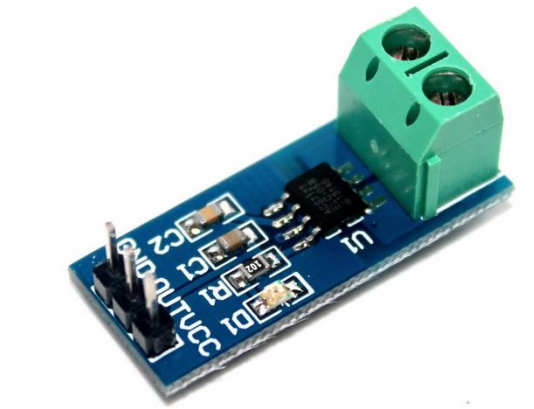
5.9. Πιεζοηλεκτρικός βομβητής (Buzzer)

Ο βομβητής είναι μια συσκευή ήχου σηματοδότησης. Περιλαμβάνει ένα ενσωματωμένο κύκλωμα ταλαντωτή και χρησιμοποιεί την αντίστροφη πιεζοηλεκτρική αρχή προκειμένου να μετατρέψει τον ηλεκτρισμό σε ήχο.

Εικόνα 5.10: Πιεζοηλεκτρικός βομβητής

5.10. Αισθητήρας ρεύματος

Ένας αισθητήρας ρεύματος ανιχνεύει το ηλεκτρικό ρεύμα σε ένα καλώδιο και παράγει σήμα ανάλογο προς αυτό. Από το σήμα που παράγεται φαίνεται η κατανάλωση ρεύματος τη δεδομένη στιγμή.



Εικόνα 5.11: Αισθητήρας ρεύματος

ΚΕΦΑΛΑΙΟ 6

ΠΑΡΑΡΤΗΜΑ Α'– ΚΩΔΙΚΑΣ ΓΙΑ ΤΟΝ ΚΥΡΙΟ ΚΟΜΒΟ

```
//----------------------------------Libraries---------------------------------//
#include "RF24Network.h"
#include "RF24.h"
#include "RF24Mesh.h"
#include <SPI.h>
#include <LiquidCrystal.h>
#include "OnewireKeypad.h"
//----------------------------------Libraries---------------------------------//

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
char KEYS[] =   // Define keys values of Keypad
{
  '1', '2', '3', 'A',
  '4', '5', '6', 'B',
  '7', '8', '9', 'C',
  '*', '0', '#', 'D'
};

OnewireKeypad <Print, 16 > Keypad(Serial, KEYS, 4, 4, A0, 4700, 1000);

//----------------------------Define Master Settings---------------------------//
#define SendFailTry 15
#define NodesNumber 4
//----------------------------Define Master Settings---------------------------//

//----------------------------Define Radio Settings----------------------------//
RF24 radio(9, 10);
RF24Network network(radio);
```

```
RF24Mesh mesh(radio, network);
//------------------------------Define Radio Settings-----------------------------//

//---------------------------------Rule Variable---------------------------------//
byte LightLimit = 70;
byte TempLimit = 31;

byte pressflag = 0;
byte AlarmMode = 0;
int AlarmRingTime = 5000;
int AlarmPin = A3;
unsigned long int StartAlarmRing;
String PressPassword;
String Password = "1234";
//---------------------------------Rule Variable---------------------------------//

//-------------------------Variables that are sent to Node-------------------------//
struct SendToNode_t {
  byte Trans1;
  byte Trans2;
  byte Trans3;
  byte Triac1;
  byte Triac2;
};
//-------------------------Variables that are sent to Node-------------------------//

//---------------------Variables that are receiver from Node---------------------//
struct ReceiveFromNode_t {
  byte node;
  byte Light;
  byte Temp;
  byte Motion;
  int Analog1;
```

```
  int Analog2;
  byte Button1;
  byte Button2;
  byte FigureDistance;
  byte FigureDirection;
};
//----------------------Variables that are receiver from Node---------------------//

//-----------------------------Array with Node Data-----------------------------//
int nodedata [10][10] = {
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
};
//-----------------------------Array with Node Data-----------------------------//

//-------------------------Array with Data sent to Node-------------------------//
int nodesend [10][10] = {
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
};
//-------------------------Array with Data sent to Node-------------------------//
```

```
void setup() {
  Serial.begin(250000);
  lcd.begin(16, 2);
  mesh.setNodeID(0);
  pinMode(AlarmPin, OUTPUT);
  Serial.println("Automation Home Mesh Network...");
  Serial.println("Master Is Starting With ID 0");
  Serial.println("Software Version: Master_V0.9");
  mesh.begin();
  PrintNodeData();
}


void loop() {
  mesh.update();// Network update
  mesh.DHCP();
  if (network.available()) { //Check if there are data for Master to read
    RF24NetworkHeader header;
    ReceiveFromNode_t ReceiveFromNode;
    network.read(header, &ReceiveFromNode, sizeof(ReceiveFromNode)); //Read
the message
    if (ReceiveFromNode.node >= 1 && ReceiveFromNode.node <=
NodesNumber) { //Put the variables in the nodeData array
      nodedata[1][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Light;
      nodedata[2][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Temp;
      nodedata[3][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Motion;
      nodedata[4][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Analog1;
      nodedata[5][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Analog2;
      nodedata[6][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Button1;
      nodedata[7][(ReceiveFromNode.node) - 1] = ReceiveFromNode.Button2;
      nodedata[8][(ReceiveFromNode.node) - 1] =
ReceiveFromNode.FigureDistance;
```

```
    nodedata[9][(ReceiveFromNode.node) - 1] = ReceiveFromNode.FigureDirection;
    PrintNodeData();
    PrintNodesend();
   }
  }


  //-------------------------------------Rules-------------------------------------//

  //Node 1 - The light turns on when there is movement and while there isn't enough lightness inside the room//
  if (nodedata[3][0] == 1 && nodesend[2][0] == 0 && nodedata[1][0] < LightLimit) {
   nodesend[2][0] = 1;
   SendToNode(1);
  }
  if (nodedata[3][0] == 0 && nodesend[2][0] == 1) {
   nodesend[2][0] = 0;
   SendToNode(1);
  }

  //Node 3 - Same as 1st - light///
  if (nodedata[3][2] == 1 && nodesend[2][2] == 0 && nodedata[1][2] < LightLimit) {
   nodesend[2][2] = 1;
   SendToNode(3);
  }
  if (nodedata[3][2] == 0 && nodesend[2][2] == 1) {
   nodesend[2][2] = 0;
   SendToNode(3);
  }

  //Node 3 - The airconditioning turns on when temperature is over 40 degrees//
  if (nodedata[2][2] >= TempLimit && nodesend[1][2] == 0) {
```

```
    nodesend[1][2] = 1; //fan
    nodesend[0][2] = 1;
    SendToNode(3);
  }
  if (nodedata[2][2] < TempLimit && nodesend[1][2] == 1) {
    nodesend[1][2] = 0;
    nodesend[0][2] = 0;
    SendToNode(3);
  }

  //Node 2 - Button
  if (nodedata[5][1] > 100 && pressflag == 0) {
    if (nodesend[0][1] == 0) {
      nodesend[0][1] = 1;
      nodesend[1][1] = 1;
      nodesend[2][1] = 1;
    }
    else if (nodesend[0][1] == 1) {
      nodesend[0][1] = 0;
      nodesend[1][1] = 0;
      nodesend[2][1] = 0;
    }
    SendToNode(2);
    pressflag = 1;
  }
  if (nodedata[5][1] < 100 && pressflag == 1) {
    pressflag = 0;
  }

  //Master Node - Alarm screen//
  if (AlarmMode == 1 && (nodedata[3][0] == 1 || nodedata[3][1] == 1 ||
nodedata[3][2] == 1))StartAlarmRing = millis();
  if ((StartAlarmRing + AlarmRingTime >= millis()) && AlarmMode == 1 && millis()
```

```
> AlarmRingTime)digitalWrite(AlarmPin, HIGH);
  else digitalWrite(AlarmPin, LOW);
  //---------------------------------Rules-------------------------------------//



  //---------------------------------Lcd---------------------------------------//

  lcd.setCursor(14, 1);
  if (AlarmMode == 1)lcd.print("Lo");
  else lcd.print("Un");

   lcd.setCursor(0, 0);
  lcd.print("C:");
  double Voltage = (nodedata[4][1] / 1024.0) * 5000; // Gets you mV
  double Amps = ((Voltage - 2500) / 185);
  //double Amps=((nodedata[4][1]*0.0049)-2.5)/0.185;
  if(Amps>0)Amps=0;
  lcd.print(Amps*-5,1);
  lcd.setCursor(5, 0);
  lcd.print(" ");

  lcd.setCursor(7, 0);
  lcd.print("T:");
  lcd.print(nodedata[2][3]);

  lcd.setCursor(12, 0);
  lcd.print("H:");
  lcd.print(nodedata[4][3]);

  lcd.setCursor(0, 1);
  lcd.print("Pass:");

  if ((Keypad.Key_State() == 1))     // not pressed = 0, pressed = 1, released = 2,
```

```
held = 3
  {
    char keypress = Keypad.Getkey();  // put value of key pressed in variable 'keypress'
    while ((Keypad.Key_State())) {} // Stay here while Key is held down
    if (keypress >= '0' && keypress <= '9' && PressPassword.length() <= 3)PressPassword = PressPassword + keypress;
    if (keypress == 'B')PressPassword = "";
    if (keypress == 'A' && PressPassword == Password) {
      PressPassword = "";
      if (AlarmMode == 1)AlarmMode = 0;
      else if (AlarmMode == 0)AlarmMode = 1;
    }
    lcd.setCursor(5, 1);
  lcd.print("    ");
  lcd.setCursor(5, 1);
  lcd.print(PressPassword);
  }
  //-----------------------------------Lcd---------------------------------------//

}

//-------------------------Printing the Nodesend Array-------------------------//
void PrintNodesend() {
  Serial.println("////////////////////////////////////////////////////////");
  for (int j = 0; j <= 4; j++) {
    for (int i = 0; i <= 3; i++) {
      Serial.print(nodesend[j][i]);
      Serial.print("    ");
    }
    Serial.println();
    Serial.println();
  }
```

```
  Serial.println("////////////////////////////////////////////////////");
}
//------------------------Printing the Nodesend Array-------------------------//


//------------------------Printing the NodeData Array-------------------------//
void PrintNodeData() {
  Serial.println("////////////////////////////////////////////////////");
  for (int j = 0; j <= 9; j++) {
    for (int i = 0; i <= 3; i++) {
      Serial.print(nodedata[j][i]);
      Serial.print("    ");
    }
    Serial.println();
    Serial.println();
  }
  Serial.println("////////////////////////////////////////////////////");
}
//------------------------Printing the NodeData Array-------------------------//

//-------------------------------Sending to Node-------------------------------//
void SendToNode(byte NodeID) {
  int check = 0; //Variable to check if the message is sent
  SendToNode_t SendToNode = {nodesend[0][NodeID - 1], nodesend[1][NodeID -
1], nodesend[2][NodeID - 1], nodesend[3][NodeID - 1], nodesend[4][NodeID - 1]};
//Fill the struct to send
  for (int i = 0; i < mesh.addrListTop; i++) { //Search for nodeAddress and send
message
    if (mesh.addrList[i].nodeID == NodeID) {
      do {
        RF24NetworkHeader header(mesh.addrList[i].address, OCT); //Constructing
a header
        if (network.write(header, &SendToNode, sizeof(SendToNode)))check = 1;
```

```
      } while (check == 0); //If the message is not sent then repeat
      break;
    }
  }
}
//------------------------------Sending to Node-------------------------------//
```

ΚΕΦΑΛΑΙΟ 7

ΠΑΡΑΡΤΗΜΑ Β' – ΚΩΔΙΚΑΣ ΓΙΑ ΤΟΥΣ ΑΠΛΟΥΣ ΚΟΜΒΟΥΣ

```
//----------------------------------Libraries-----------------------------------//
#include "RF24.h"
#include "RF24Network.h"
#include "RF24Mesh.h"
#include <SPI.h>
#include <EEPROM.h>
#include <OneWire.h>
#include <DallasTemperature.h>
//----------------------------------Libraries-----------------------------------//

//-----------------------------Define Radio-----------------------------//
RF24 radio(9, 10);
RF24Network network(radio);
RF24Mesh mesh(radio, network);
//-----------------------------Define Radio-----------------------------//

//---------------------------Define Node Settings---------------------------//
#define nodeID 2
#define VariableSample 5
#define LightSensor 1
#define TempSensor 0
#define MotionSensor 1
#define Analog1Sensor  1
#define Analog2Sensor  0
#define Button1Sensor 0
#define Button2Sensor  0
#define FigureDistanceSensor  0
#define FigureDirectionSensor 0
```

```
//-----------------------------Define Node Settings-----------------------------//

//-----------------------------Temperature Definitions--------------------------//
#define DS18B20 A1
OneWire ourWire(DS18B20);
DallasTemperature sensors(&ourWire);
DeviceAddress tempDeviceAddress;
int  resolution = 8;
//-----------------------------Temperature Definitions--------------------------//

//-------------------------------Pin Definitions--------------------------------//
const int  Button1Pin = A6;
const int  Button2Pin = A7;
const int MotionPin = 8;
const int LightPin = A0;
const int Analog1Pin = A2;
const int Analog2Pin = A3;

int Trans1Pin = 5;
int Trans2Pin = 4;
int Trans3Pin = 3;
int Triac1Pin = 6;
int Triac2Pin = 7;
//-------------------------------Pin Definitions--------------------------------//

//-----------------------------Node Variables---------------------------------//
int LightState = 0;
int LastLightState = 0;
unsigned long int LightStateSample;
int LightStateSampleCounter = 0;

int TempState = 0;
int LastTempState = 0;
```

```
unsigned long int TempStateSample;
int TempStateSampleCounter = 0;

int Analog1State = 0;
int LastAnalog1State = 0;
unsigned long int Analog1StateSample;
int Analog1StateSampleCounter = 0;

int Analog2State = 0;
int LastAnalog2State = 0;
unsigned long int Analog2StateSample;
int Analog2StateSampleCounter = 0;

int Button1State = 0;
int LastButton1State = 0;
int Button2State = 0;
int LastButton2State = 0;
int MotionState = 0;
int LastMotionState = 0;
int FigureDistanceState = 0;
int LastFigureDistanceState = 0;
int FigureDirectionState = 0;
int LastFigureDirectionState = 0;

byte Trans1State, Trans2State, Trans3State;
byte Triac1State, Triac2State;
//------------------------------Node Variables---------------------------------//

//-------------------------Struct for sending to Master-----------------------//
struct SendToServer_t {
  byte node;
  byte Light;
  byte Temp;
```

```
  byte Motion;
  int Analog1;
  int Analog2;
  byte Button1;
  byte Button2;
  byte FigureDistance;
  byte FigureDirection;
};
//-------------------------Struct for sending to Master----------------------//

//-------------------------Struct for receiving from Master----------------------//
struct ReceiveFromServer_t {
  byte Trans1;
  byte Trans2;
  byte Trans3;
  byte Triac1;
  byte Triac2;
};
//-------------------------Struct for receiving from Master----------------------//

void setup() {
  delay(2000);
  //pin types
  pinMode(Button1Pin, INPUT);
  pinMode(Button2Pin, INPUT);
  pinMode(MotionPin, INPUT);
  pinMode(LightPin, INPUT);
  pinMode(Analog1Pin, INPUT);
  pinMode(Analog2Pin, INPUT);
  pinMode(Trans1Pin, OUTPUT);
  pinMode(Trans2Pin, OUTPUT);
  pinMode(Trans3Pin, OUTPUT);
  pinMode(Triac1Pin, OUTPUT);
```

```
  pinMode(Triac2Pin, OUTPUT);
  if (TempSensor == 1) { //temperature sensor connection
    sensors.begin();
    sensors.getAddress(tempDeviceAddress, 0);
    sensors.setResolution(tempDeviceAddress, resolution);
  }
  mesh.setNodeID(nodeID); //network connection
  mesh.begin();
}

void loop() {
  mesh.update(); //update network
  if ( ! mesh.checkConnection() ) { //if connection is lost the take a new address
    mesh.renewAddress();
  }

  if (LightSensor == 1) { //calculate light sensor output
    if (LightStateSampleCounter < VariableSample) {
      LightStateSample = LightStateSample + map(analogRead(LightPin), 0, 1023,
0, 100);;
      LightStateSampleCounter++;
    }
    if (LightStateSampleCounter >= VariableSample) {
      LightState = LightStateSample / VariableSample;
      LightStateSample = 0;
      LightStateSampleCounter = 0;
    }
  } else {
    LightState = 0;
    LightStateSampleCounter = 0;
    LightStateSample = 0;
  }
```

```
if (TempSensor == 1) { //calculate temperature sensor output
  if (TempStateSampleCounter < VariableSample) {
    TempStateSample = TempStateSample + sensors.getTempCByIndex(0);;
    TempStateSampleCounter++;
  }
  if (TempStateSampleCounter >= VariableSample) {
    TempState = TempStateSample / VariableSample;
    TempStateSample = 0;
    TempStateSampleCounter = 0;
  }
  sensors.setResolution(tempDeviceAddress, resolution);
  sensors.requestTemperatures();
} else {
  TempState = 0;
  TempStateSampleCounter = 0;
  TempStateSample = 0;
}

if (Analog1Sensor == 1) { //calculate analog1 output
  if (Analog1StateSampleCounter < VariableSample) {
    Analog1StateSample = Analog1StateSample + analogRead(Analog1Pin);
    Analog1StateSampleCounter++;
  }
  if (Analog1StateSampleCounter >= VariableSample) {
    Analog1State = Analog1StateSample / VariableSample;
    Analog1StateSample = 0;
    Analog1StateSampleCounter = 0;
  }
} else {
  Analog1State = 0;
  Analog1StateSampleCounter = 0;
  Analog1StateSample = 0;
}
```

```
  if (Analog2Sensor == 1) { //calculate analog2 output
    if (Analog2StateSampleCounter < VariableSample) {
      Analog2StateSample = Analog2StateSample + analogRead(Analog2Pin);
      Analog2StateSampleCounter++;
    }
    if (Analog2StateSampleCounter >= VariableSample) {
      Analog2State = Analog2StateSample / VariableSample;
      Analog2StateSample = 0;
      Analog2StateSampleCounter = 0;
    }
  } else {
    Analog2State = 0;
    Analog2StateSampleCounter = 0;
    Analog2StateSample = 0;
  }

  if (Button1Sensor == 1)Button1State = !digitalRead(Button1Pin); //Check Button1
state
  if (Button2Sensor == 1)Button2State = !digitalRead(Button2Pin); //Check Button2
state
  if (MotionSensor == 1)MotionState = digitalRead(MotionPin); //Check motion
sensor state
  if (FigureDistanceSensor == 1);
  if (FigureDirectionSensor == 1);

  //if one variable changes then all the variables are sent to Master
  if (Button1State != LastButton1State || Button2State != LastButton2State ||
MotionState != LastMotionState || LightState != LastLightState || TempState !=
LastTempState || Analog1State != LastAnalog1State || Analog2State !=
LastAnalog2State || FigureDistanceState != LastFigureDistanceState ||
FigureDirectionState != LastFigureDirectionState) {
    SendToServer_t SendToServer = {nodeID, LightState, TempState,
```

```
MotionState, Analog1State, Analog2State, Button1State, Button2State,
FigureDistanceState, FigureDirectionState}; //data for the master
    RF24NetworkHeader header(0);
    if (network.write(header, &SendToServer, sizeof(SendToServer))) {
      LastButton1State = Button1State;
      LastButton2State = Button2State;
      LastMotionState = MotionState;
      LastLightState = LightState;
      LastTempState = TempState;
      LastAnalog2State = Analog2State;
      LastAnalog1State = Analog1State;
      LastFigureDistanceState = FigureDistanceState;
      LastFigureDirectionState = FigureDirectionState;
    }
  }

  while (network.available()) { //Check if there are Data from Master
    RF24NetworkHeader header;
    ReceiveFromServer_t ReceiveFromServer;
    network.read(header, &ReceiveFromServer, sizeof(ReceiveFromServer));
    Trans1State = ReceiveFromServer.Trans1;
    Trans2State = ReceiveFromServer.Trans2;
    Trans3State = ReceiveFromServer.Trans3;
    Triac1State = ReceiveFromServer.Triac1;
    Triac2State = ReceiveFromServer.Triac2;
    //set Pins due to Master Commands
    digitalWrite(Trans1Pin,Trans1State);
    digitalWrite(Trans2Pin,Trans2State);
    digitalWrite(Trans3Pin,Trans3State);
    digitalWrite(Triac1Pin,Triac1State);
    digitalWrite(Triac2Pin,Triac2State);
  }
}
```

ΚΕΦΑΛΑΙΟ 8

ΠΑΡΑΡΤΗΜΑ Γ' – ΚΩΔΙΚΑΣ ΣΧΕΔΙΑΣΗΣ ΚΟΥΤΙΟΥ ΟΘΟΝΗΣ

```
// [Box dimensions]
// Length
Length      = 40;
// Width
Width       = 90;
// Height
Height      = 130;
// Wall thickness
Thick       = 2;//[2:5]

// [Box options]
//Filet diameter
Filet       = 2;//[0.1:12]
//Filet smoothness
Resolution  = 50;//[1:100]
//Tolerance (Panel/rails gap)
m           = 0.9;
//Pieds PCB - PCB feet (x4)
PCBFeet     = 0;// [0:No, 1:Yes]
//Decorations to ventilation holes
Vent        = 1;// [0:No, 1:Yes]
//Holes width (in mm)
Vent_width  = 1.5;

//Low left corner X position
PCBPosX     = 7;
//Low left corner Y position
PCBPosY     = 6;
```

```
//PCB Length
PCBLength       = 70;
//PCB Width
PCBWidth        = 50;
//Feet height
FootHeight      = 10;
//Foot diameter
FootDia         = 8;
//Hole diameter
FootHole        = 3;

// [STL element to export]
//Top shell
TShell          = 0;// [0:No, 1:Yes]
//Bottom shell
BShell          = 0;// [0:No, 1:Yes]
//Front panel
FPanL           = 1;// [0:No, 1:Yes]
//Back panel
BPanL           = 0;// [0:No, 1:Yes]

// [Hidden]
//Shell color
Couleur1        = "Orange";
//Panels color
Couleur2        = "OrangeRed";
//Making decorations thicker if it is a vent to make sure they go through shell
Dec_Thick       = Vent ? Thick * 2 : Thick;
//Depth decoration
Dec_size        = Vent ? Thick * 2 : 0.8;
```

```
//////////Generic rounded box/////////

module RoundBox($a = Length, $b = Width, $c = Height) { // Cube bords arrondis
  $fn = Resolution;
  translate([0, Filet, Filet]) {
    minkowski () {
      cube ([$a - (Length / 2), $b - (2 * Filet), $c - (2 * Filet)], center = false);
      rotate([0, 90, 0]) {
        cylinder(r = Filet, h = Length / 2, center = false);
      }
    }
  }
}

//////////////////////////////Module Shell////////////////////////////////

module Coque() {
  Thick = Thick * 2;
  difference() {
    difference() { //sides decoration
      union() {
        difference() {//Substraction Fileted box
          difference() { //Median cube slicer
            union() {
              difference() {
                RoundBox();
                translate([Thick / 2, Thick / 2, Thick / 2]) {
                  RoundBox($a = Length - Thick, $b = Width - Thick, $c = Height -
Thick);
                }
              }
              difference() { //largeur Rails
                translate([Thick + m, Thick / 2, Thick / 2]) { // Rails
```

```
            RoundBox($a = Length - ((2 * Thick) + (2 * m)), $b = Width - Thick, $c
= Height - (Thick * 2));
           }//fin Rails
           translate([((Thick + m / 2) * 1.55), Thick / 2, Thick / 2 + 0.1]) { // +0.1
added to avoid the artefact
            RoundBox($a = Length - ((Thick * 3) + 2 * m), $b = Width - Thick, $c =
Height - Thick);
           }
          }
         }
         translate([-Thick, -Thick, Height / 2]) {
          cube ([Length + 100, Width + 100, Height], center = false);
         }
        }//End Median cube slicer
        translate([-Thick / 2, Thick, Thick]) {
         RoundBox($a = Length + Thick, $b = Width - Thick * 2, $c = Height -
Thick);
        }
       }

       difference() { // wall fixation box legs
        union() {
         translate([3 * Thick + 5, Thick, Height / 2]) {
          rotate([90, 0, 0]) {
           $fn = 6;
           cylinder(d = 16, Thick / 2);
          }
         }

         translate([Length - ((3 * Thick) + 5), Thick, Height / 2]) {
          rotate([90, 0, 0]) {
           $fn = 6;
           cylinder(d = 16, Thick / 2);
```

```
          }
        }
      }
      translate([4, Thick + Filet, Height / 2 - 57]) {
        rotate([45, 0, 0]) {
          cube([Length, 40, 40]);
        }
      }
      translate([0, -(Thick * 1.46), Height / 2]) {
        cube([Length, Thick * 2, 10]);
      }
    } //Fin fixation box legs
  }

  union() { // outbox sides decorations

    for (i = [0:Thick:Length / 4]) {

      translate([10 + i, -Dec_Thick + Dec_size, 1]) {
        cube([Vent_width, Dec_Thick, Height / 4]);
      }
      translate([(Length - 10) - i, -Dec_Thick + Dec_size, 1]) {
        cube([Vent_width, Dec_Thick, Height / 4]);
      }
      translate([(Length - 10) - i, Width - Dec_size, 1]) {
        cube([Vent_width, Dec_Thick, Height / 4]);
      }
      translate([10 + i, Width - Dec_size, 1]) {
        cube([Vent_width, Dec_Thick, Height / 4]);
      }
    }
  }
}
```

```
    union() { //sides holes
      $fn = 50;
      translate([3 * Thick + 5, 20, Height / 2 + 4]) {
        rotate([90, 0, 0]) {
          cylinder(d = 2, 20);
        }
      }
      translate([Length - ((3 * Thick) + 5), 20, Height / 2 + 4]) {
        rotate([90, 0, 0]) {
          cylinder(d = 2, 20);
        }
      }
      translate([3 * Thick + 5, Width + 5, Height / 2 - 4]) {
        rotate([90, 0, 0]) {
          cylinder(d = 2, 20);
        }
      }
      translate([Length - ((3 * Thick) + 5), Width + 5, Height / 2 - 4]) {
        rotate([90, 0, 0]) {
          cylinder(d = 2, 20);
        }
      }
    }
  }
}

////////////////////////////// - Experiment - ////////////////////////////////////////

////////////////////// - Foot with base filet - ////////////////////////////
module foot(FootDia, FootHole, FootHeight) {
  Filet = 2;
  color(Couleur1)
  translate([0, 0, Filet - 1.5])
```

```
  difference() {
    difference() {
      cylinder(d = FootDia + Filet, FootHeight - Thick, $fn = 100);
      rotate_extrude($fn = 100) {
        translate([(FootDia + Filet * 2) / 2, Filet, 0]) {
          minkowski() {
            square(10);
            circle(Filet, $fn = 100);
          }
        }
      }
    }
    cylinder(d = FootHole, FootHeight + 1, $fn = 100);
  }
}

module Feet() {
  ////////////////////// - PCB only visible in the preview mode - //////////////////////
  translate([3 * Thick + 2, Thick + 5, FootHeight + (Thick / 2) - 0.5]) {
    % square ([PCBLength + 10, PCBWidth + 10]);
    translate([PCBLength / 2, PCBWidth / 2, 0.5]) {
      color("Olive")
      % text("PCB", halign = "center", valign = "center", font = "Arial black");
    }
  }

  ////////////////////////////// - 4 Feet - ////////////////////////////////////////
  translate([3 * Thick + 7, Thick + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
  translate([(3 * Thick) + PCBLength + 7, Thick + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
```

```
  translate([(3 * Thick) + PCBLength + 7, (Thick) + PCBWidth + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
  translate([3 * Thick + 7, (Thick) + PCBWidth + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
}

////////////////////////////////////////////////////////////////////
////////////////////// <- Holes Panel Manager -> //////////////////////
////////////////////////////////////////////////////////////////////

//                        <- Panel ->
module Panel(Length, Width, Thick, Filet) {
  scale([0.5, 1, 1])

  minkowski() {

    cube([Thick, Width - (Thick * 2 + Filet * 2 + m), Height - (Thick * 2 + Filet * 2 +
m)]);
    translate([0, Filet, Filet])
    rotate([0, 90, 0])
    cylinder(r = Filet, h = Thick, $fn = 100);
  }
}

//                        <- Circle hole ->
// Cx=Cylinder X position | Cy=Cylinder Y position | Cdia= Cylinder dia |
Cheight=Cyl height
module CylinderHole(OnOff, Cx, Cy, Cdia) {
  if (OnOff == 1)
    translate([Cx, Cy, -1])
    cylinder(d = Cdia, 10, $fn = 50);
```

```
}

//                      <- Square hole ->
// Sx=Square X position | Sy=Square Y position | Sl= Square Length | Sw=Square
Width | Filet = Round corner
module SquareHole(OnOff, Sx, Sy, Sl, Sw, Filet) {
  if (OnOff == 1)
    minkowski() {
    translate([Sx + Filet / 2, Sy + Filet / 2, -1])
    cube([Sl - Filet, Sw - Filet, 10]);
    cylinder(d = Filet, h = 10, $fn = 100);
  }
}

//                    <- Linear text panel ->
module LText(OnOff, Tx, Ty, Font, Size, Content) {
  if (OnOff == 1)
    translate([Tx, Ty, Thick + .5])
    linear_extrude(height = 0.5) {
    text(Content, size = Size, font = Font);
  }
}

//                    <- Circular text panel->
module CText(OnOff, Tx, Ty, Font, Size, TxtRadius, Angl, Turn, Content) {
  if (OnOff == 1) {
    Angle = -Angl / len(Content);
    translate([Tx, Ty, Thick + .5])
    for (i = [0:len(Content) - 1] ) {
      rotate([0, 0, i * Angle + 90 + Turn])
      translate([0, TxtRadius, 0]) {
        linear_extrude(height = 0.5) {
          text(Content[i], font = Font, size = Size,  valign = "baseline", halign =
```

```
"center");
        }
      }
    }
  }
}

////////////////////// <- New module Panel -> //////////////////////
module FPanL() {
  difference() {
    color(Couleur2)
    Panel(Length, Width, Thick, Filet);
    rotate([90, 0, 90]) {
      color(Couleur2) {
        //Start Back
        //CylinderHole(1,30,25,16);
        //CylinderHole(1,55,25,16);
        //CylinderHole(1,80,25,16);
        //CylinderHole(1,105,25,16);
        //CylinderHole(1,150,25,16);
        //CylinderHole(1,150,70,12.7);
        //End Back

        //Start Front
        SquareHole  (1, 11, 95, 64, 14, 4);    //(On/Off,Xpos,Ypos,Length,Width,Filet)
auto
        SquareHole  (1, 25.5, 15, 35, 8, 3);
        //End Front
        //                        <- To here ->
      }
    }
  }
```

```
  color(Couleur1) {
    translate ([-.5, 0, 0])
    rotate([90, 0, 90]) {

      //Start Front
      LText(1, 12, 117, "Arial Black", 3.6, "NIKOLAKI AIKATERINI");
      LText(1, 15, 4, "Arial Black", 3, "AEI PEIRAIA T.T.  M.H.Y.S");
      //End Front
      //LText(1,120,83,"Arial Black",4,"KeyPad");
      //CText(1,93,29,"Arial Black",4,10,180,0,"1 . 2 . 3 . 4 . 5 . 6");//(On/Off, Xpos,
Ypos, "Font", Size, Diameter, Arc(Deg), Starting Angle(Deg),"Text")
    }
  }
}

//////////////////////////// <- Main part -> ////////////////////////
if (TShell == 1)
  //Top Shell
  color( Couleur1, 1) {
  translate([0, Width, Height + 0.2]) {
    rotate([0, 180, 180]) {
      Coque();
    }
  }
}

if (BShell == 1)
  //Bottom shell
  color(Couleur1) {
  Coque();
}

//PCB feet
```

```
if (PCBFeet == 1)
  translate([PCBPosX, PCBPosY, 0]) {
  Feet();
}

//Front panel
if (FPanL == 1)
  translate([Length-(Thick * 2 + m / 2), Thick + m / 2, Thick + m / 2])
  FPanL();

//Back panel
if (BPanL == 1)
  color(Couleur2)
  translate([Thick + m / 2, Thick + m / 2, Thick + m / 2])
  Panel(Length, Width, Thick, Filet);
```

ΚΕΦΑΛΑΙΟ 9

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

Ολοκληρώνοντας την υλοποίηση της παρούσας πτυχιακής εργασίας, γίνονται κατανοητές οι δυσκολίες μίας τόσο απαιτητικής κατασκευής, αλλά και τα προτερήματα αυτής. Τα υλικά και τα εξαρτήματα είναι χαμηλού κόστους, γεγονός που είναι δελεαστικό για να πραγματοποιηθεί μία τέτοια εγκατάσταση οικιακού αυτοματισμού. Επίσης, το διαδίκτυο προσφέρει απεριόριστες πληροφορίες και κομμάτι αυτών, αποτελούν και οι κώδικες που αφορούν το κάθε εξάρτημα του «Έξυπνου Σπιτιού». Σε αντίφαση με τα παραπάνω παρουσιάζεται να είναι η ένωση των προγραμμάτων αυτών και η σύμπτυξή τους σε έναν κώδικα που θα αποτελεί τον κύριο μηχανισμό του εκάστοτε κόμβου. Πολλές φορές, οι κώδικες είναι δύσκολο να συνδυαστούν λόγω των πολλών μεταβλητών τους είτε λόγω της μη πλήρης κατανόησης της λειτουργίας του εξαρτήματος από τον προγραμματιστή.

Εντούτοις, ένας ικανός προγραμματιστής και γνώστης ηλεκτρολογίας και ηλεκτρονικών εξαρτημάτων, είναι σε θέση να υλοποιήσει ένα σύστημα όπως αυτό του «Έξυπνου Σπιτιού». Η διαμόρφωση του οικιακού αυτοματισμού ανάλογα με τις ανάγκες του χρήστη, αποτελεί ένα σίγουρα θετικό μέρος του. Οι προοπτικές, όμως, τις οποίες έχει με βάση την εξέλιξη της τεχνολογίας και την παροχή δυνατοτήτων που παλιότερα δεν είχε ο σύγχρονος άνθρωπος, είναι ατελείωτες.

ΚΕΦΑΛΑΙΟ 10

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Σιάτης Χρήστος, «Μελέτη και υλοποίηση συστήματος ελέγχου βηματικού κινητήρα με χρήση ARDUINO», Πτυχιακή Εργασία, Τμήμα Μηχανικών Πληροφορικής Τ.Ε., ΤΕΙ Ηπείρου, 2014

[2]ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES - IN-SYSTEM PROGRAMMABLE FLASH DATASHEET, ATMEL, Νοέμβριος 2015

[3]nRF24L01Single Chip 2.4GHz Transceiver - Product Specification, Nordic Semiconductor, Ιούλιος 2007

[4] Τεχνολογίες Διαδικτύου – Τοπολογίες και Ταξινομήσεις Δικτύων, Καθ. Χρήστος Δουληγέρης, Εύη Κοπανάκη

[5] https://el.wikipedia.org/wiki/Τοπολογία_δικτύου

[6] https://www.arduino.cc/en/Main/ArduinoBoardUno

[7] RF24Mesh V1.0.3b - A user friendly mesh overlay for sensor networks using RF24Network and nRF24L01 radio modules

Πηγή: http://tmrh20.github.io/RF24Mesh/