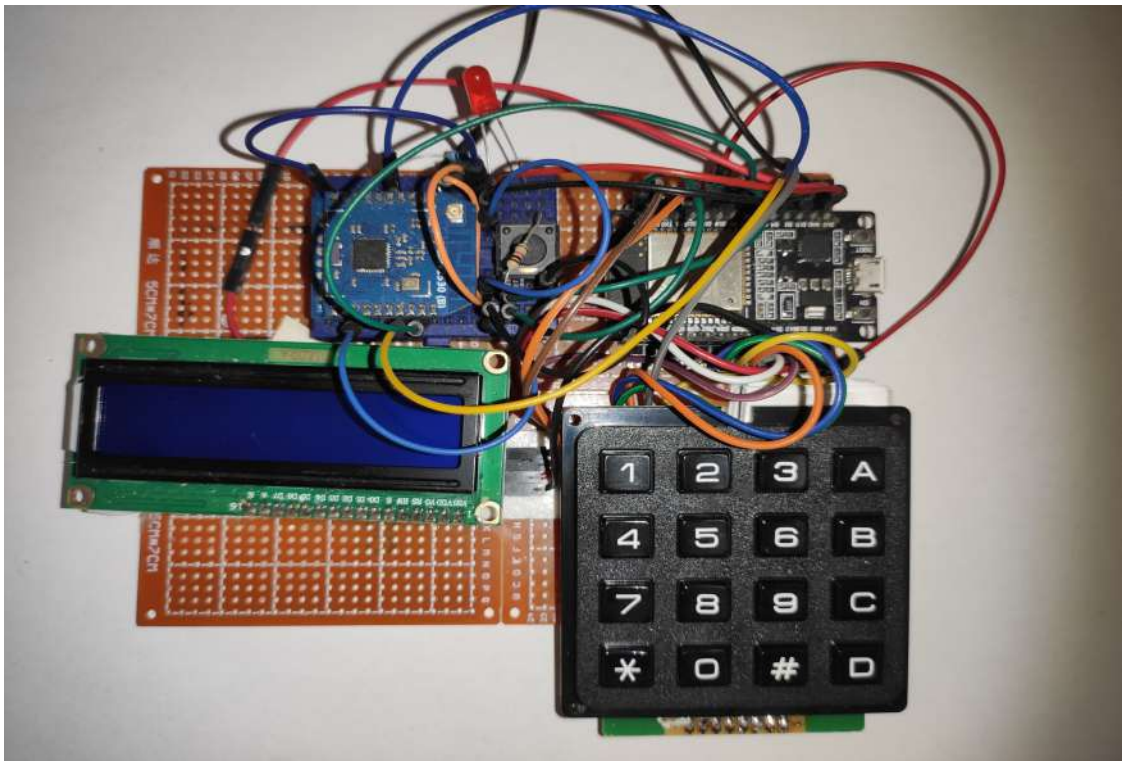
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σύστημα συναγερμού με ασύρματα αισθητήρια zigbee και ESP32 για ασύρματη σύνδεση με το router »



Του φοιτητή
Παλάζη Ορέστη
Αρ. Μητρώου: 515109

Επιβλέπων
Ονοματεπώνυμο Γιακουμής
Άγγελος
Βαθμίδα Λέκτορας

Ημερομηνία 15/09/2021

Τίτλος Δ.Ε:Σύστημα συναγερμού με ασύρματα αισθητήρια zigbee και ESP32 για ασύρματη σύνδεση με το router

Κωδικός Δ.Ε. 19164

Ονοματεπώνυμο φοιτητή: Παλάζης Ορέστης

Ονοματεπώνυμο εισηγητή: Γιακουμής Άγγελος

Ημερομηνία ανάληψης Δ.Ε. 18/11/2019

Ημερομηνία περάτωσης Δ.Ε. 15/09/2021

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

Πρόλογος

Καθώς η εγκληματικότητα αυξάνεται κάθε χρόνο το ζήτημα της ασφάλειας ενός χώρου παραμένει πάντα επίκαιρο. Τα τεχνολογικά μέσα μπορούν να προσφέρουν λύσεις που αφορούν την πρόληψη, την αντιμετώπιση και την συλλογή στοιχείων που αφορούν παράγοντες που θέτουν σε κίνδυνο μια εγκατάσταση. Ένα από τα βασικά τεχνολογικά μέσα ποστασίας ενός χώρου είναι ο συναγερμός. Η ζήτηση των συναγερμών αυξάνεται όσο αυξάνεται και η εγκληματικότητα οδηγώντας πολλά άτομα να αναζητήσουν οικονομικές και αξιόπιστες λύσεις. Ένας ασύρματος συναγερμός αποτελεί την ιδανική λύση για κάποιον που δεν έχει εξειδικευμένες γνώσεις εγκατάστασης συναγερμών ή δεν θέλει να χαλάσει την αισθητική του κτηρίου με καλώδια. Ένα πρότυπο που προσφέρει λύσεις σε ασύρματα δίκτυα αισθητήρων είναι το Zigbee. Το Zigbee αποτελεί αναγνωρισμένο πρότυπο για οικονομικές και αξιόπιστες λύσεις θέτοντας το ιδανικό για ασύρματους συναγερμούς.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στον τρόπο υλοποίησης ενός ασύρματου συναγερμού Zigbee. Αρχικά αναφέρονται μερικοί ορισμοί και στατιστικά στοιχεία για την εγκληματικότητα και την ασφάλεια των εγκαταστάσεων προκειμένου να κατανοήσουμε την ανάγκη και την γενική εικόνα προστασίας ενός χώρου. Στην συνέχεια επικεντρώνεται στον ρόλο των συναγερμών για την επίτευξη των μέτρων ασφαλείας, τα τμήματα που συνθέτουν έναν συναγερμό και πώς επιλέγουμε τον κατάλληλο συναγερμό. Το βασικό κομμάτι της πτυχιακής είναι το Zigbee. Πρώτα γίνεται μια σύγκριση του Zigbee με άλλα παρόμοια πρωτόκολλα επικοινωνίας και στην συνέχεια αναλύονται διάφορα χαρακτηριστικά του. Τέλος ακολουθεί η κατασκευή του συναγερμού όπου ορίζονται τα χαρακτηριστικά του επιθυμητού τελικού συστήματος. Αφού δοθεί το αρχικό σχέδιο υλοποίησης αναλύονται τα βήματα που έγιναν για την επίτευξη της κατασκευής.

«Alarm system with wireless Zigbee sensors and ESP32 for wireless conection with router »

«Palazis Orestis»

Abstract

This dissertation focuses on how to implement a Zigbee wireless alarm. First, some definitions and statistics for the validity and safety of the facilities are mentioned in order to understand the need and the general picture of protection of a building. It then focuses on the role of alarms in achieving security measures, the components that make up an alarm, and how we select the appropriate alarm. The main part of this dissertation is Zigbee. Zigbee is first compared to other similar communication protocols and then its various features are analyzed. Finally follows the construction of the alarm where the characteristics of the desired final system are defined. After the initial implementation plan is given, the steps taken to achieve the construction are analyzed.

Ευχαριστίες

Θα ήταν σημαντική παράλειψη εάν δεν ανέφερα το όνομα του κύριου Άγγελου Γιακουμή ο οποίος μου έδωσε την αρχική ώθηση ώστε να μπορέσω να μελετήσω κατάλληλη βιβλιογραφία που φάνηκε καθοριστική για την ανάπτυξη της πτυχιακής εργασίας μου. Επιπλέον σημαντική βοήθεια είχα φυσικά και από την οικογένειά μου που με βοήθησε καθόλη την διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: Η έννοια της ασφάλειας μιας εγκατάστασης

1.1 Το αίσθημα ασφάλειας

Το αίσθημα της ασφάλειας στον άνθρωπο αποτελεί βασική προϋπόθεση για να έχει ποιοτική ζωή. Ο άνθρωπος πρέπει να αισθάνεται ασφαλής, γιατί μόνο έτσι μπορεί να αναπτύσσεται, να δημιουργεί, να θέτει στόχους και να προοδεύει στην κοινωνία. Πρέπει να νιώθει ασφάλεια τόσο στην οικία του όσο και στο εξωτερικό περιβάλλον στο οποίο βρίσκεται όπως η εργασία τα εμπορικά καταστήματα οι δημόσιοι χώροι κτλ.

1.2 Η έννοια του κινδύνου

Στην κοινωνία μας ελλοχεύουν πολλοί καθημερινοί κίνδυνοι. Ως κίνδυνος νοείται η πιθανότητα εκδήλωσης μιας κατάστασης η οποία μπορεί να απειλήσει τη ζωή, την υγεία, την ιδιοκτησία ή το περιβάλλον. Οι κίνδυνοι χωρίζονται σε δύο μεγάλες κατηγορίες, τους ανθρωπογενής και τους φυσικούς.

Η ανθρωπογενής απειλή ορίζεται ως κάθε περιστατικό ή κατάσταση που προκαλείται από τον άνθρωπο, εκούσια ή ακούσια, με πράξη ή παράλειψη και έχει ως αποτέλεσμα να επέρχεται κίνδυνος είτε σε ανθρώπους είτε σε ιδιοκτησίες, είτε στο περιβάλλον. Αντιθέτως, οι φυσικές καταστροφές οφείλονται στις φυσικές καταστροφές όπως σεισμοί και πλημμύρες.[2]

1.3 Εγκληματολογικά στατιστικά

Οι εγκληματολογικές στατιστικές είναι η συλλογή στοιχείων που σχετίζονται με την εγκληματικότητα. Είναι ένα βασικό μεθοδολογικό εργαλείο για τη μέτρηση του εγκληματικού φαινομένου. Η μέτρηση και η περιγραφή των εγκλημάτων βοηθά τον ερευνητή να διερευνήσει τους παράγοντες εγκληματογένεσης και να διατυπώσει προτάσεις αντεγκληματικής πολιτικής μέσω της πρόβλεψης και της εξέλιξή της. [2]

Η στατιστική της αστυνομίας τηρείται από το αρχηγείο ελληνικής αστυνομίας και πρόκειται για τη συλλογή δεδομένων για αδικήματα που καταγράφονται από τις αστυνομικές αρχές και βοηθά στην καταγραφή και εντοπισμό των εγκληματογόνων περιοχών, τη σχέση τόπου-χρόνου με την τέλεση συγκεκριμένων εγκλημάτων καθώς και τα χαρακτηριστικά στοιχεία των δραστών ώστε να λαμβάνονται τα κατάλληλα προληπτικά μέτρα για την αποτροπή τέλεσης νέων εγκλημάτων και να επιτυγχάνεται η εξιχνίαση μεγαλύτερου αριθμού τελεσθέντων εγκλημάτων.

Σύμφωνα με τα επίσημα στατιστικά στοιχεία της αστυνομίας διαπιστώνεται αύξηση της εγκληματικότητας που οφείλεται σε λόγους όπως η ακύρωση των όποιων προσπαθειών των διωκτικών αρχών για την πάταξη του εγκλήματος από τις γραφειοκρατικές διαδικασίες και τις παθογένειες του ποινικού μας συστήματος, την αύξηση της λαθρομετανάστευσης και η συμμετοχή των αλλοδαπών στα εγκλήματα, η κυκλοφορία παράνομων όπλων στα χέρια κακοποιών που κάνει την πάταξη της εγκληματικότητας έργο πολυσύνθετο και φυσικά τη διάλυση της οικογένειας με αντίκτυπο τη συμπεριφορά των παιδιών στην κοινωνία.

ΕΠΙΚΡΑΤΕΙΑ	2019					2018				
	ΕΓΚΛΗΜΑΤΑ			ΔΡΑΣΤΕΣ		ΕΓΚΛΗΜΑΤΑ			ΔΡΑΣΤΕΣ	
	τελ/να	απόπειρες	εξιχνιάσεις	ημεδαποί	αλλοδαποί	τελ/να	απόπειρες	εξιχνιάσεις	ημεδαποί	αλλοδαποί
ΚΛΟΠΕΣ - ΔΙΑΡΡΗΞΕΙΣ	**81.734**	5.081	14.985	9.705	4.954	**74.183**	4.703	14.389	9.769	5.424
Κλοπές - Διαρρήξεις από ιχε αυτ/τα	21.769	1.566	2.266	976	629	18.507	1.323	1.958	1.112	696
Κλοπές - Διαρρήξεις ιερών ναών	342	52	224	181	39	328	69	231	169	40
Κλοπές - Διαρρήξεις καταστημάτων	7.916	818	3.497	2.354	1.247	7.808	721	3.412	2.426	1.147
Κλοπές - Διαρρήξεις λοιπές	10.666	406	3.154	3.097	745	10.957	456	3.138	2.849	956
Κλοπές - Διαρρήξεις οικιών	22.107	2.039	4.044	2.523	1.040	20.871	2.007	4.027	2.666	1.146
Κλοπές - Διαρρήξεις σε συγκοινωνιακά μέσα	5.735	35	320	29	322	4.809	17	312	33	328
Κλοπές με αρπαγές τσαντών	906	15	94	65	26	801	9	93	44	31
Κλοπές σε δημόσιο χώρο-μικροκλοπες	12.293	150	1.386	480	906	10.102	101	1.218	470	1.080

Εικόνα 1.1 Στατιστικά στοιχεία διαρρήξεων για τα έτη 2018-2019 [31]

1.4 Ασφάλεια εγκατάστασης

Η ασφάλεια φυσικών εγκαταστάσεων περιλαμβάνει όλα τα απαραίτητα μέτρα προστασίας με ανθρώπινο δυναμικό και τεχνικά μέσα, τα οποία έχουν ως βασικό στόχο την προστασία τόσο των φυσικών εγκαταστάσεων όσο και την ασφάλεια των ατόμων που βρίσκονται σε αυτές, αλλά και την εξασφάλιση ακεραιότητας των λειτουργιών της εγκατάστασης και των περιουσιακών στοιχείων που διαθέτουν. [2]

Επιτυγχάνεται κατόπιν μελέτης ασφάλειας και σύνταξης σχεδίου ασφαλείας. Έτσι τοποθετούνται και τίθενται σε λειτουργία τα τεχνολογικά μέσα και οι απαιτούμενες διαδικασίες οριοθετώντας το επιδιωκόμενο επίπεδο ασφαλείας

Τα λαμβανόμενα μέτρα αποσκοπούν στη μείωση της επικινδυνότητας και της ευπάθειας από εσωτερικές και εξωτερικές απειλές με τον εντοπισμό εισόδου ή παραμονής αναρμόδιων προσώπων, επικίνδυνων υλών και υλικών σε μη εξειδικευμένους χώρους, ανίχνευση κενών ασφαλείας της εγκατάστασης και την πρόληψη των πυρκαγιών.

Τα βασικά μέτρα ασφαλείας μιας εγκατάστασης κατηγοριοποιούνται σε τεχνολογικά μέσα, ανθρώπινο δυναμικό και καθορισμένες διαδικασίες. Αυτά περιλαμβάνουν τους συναγερμούς, access contol, κύκλωμα καταγραφής, περιπολίες, στατικές φυλάξεις, περίφραξη, επαρκής φωτισμός και έλεγχος προσώπων-οχημάτων-αποσκευών.

Κεφάλαιο 2ο: Συστήματα συναγερμών

2.1 Εισαγωγή στους συναγερμούς

Ως σύστημα ασφαλείας μπορεί να χαρακτηριστεί κάθε σύστημα που προσφέρει στο χρήστη προστασία απέναντι σε έναν ή περισσότερους κινδύνους όπως για παράδειγμα μια διάρρηξη, τον εντοπισμό φωτιάς σε έναν χώρο ή την ύπαρξη διαρροής. Η σήμανση συναγερμού γίνεται σε κάθε περίπτωση με διαφορετικό ήχο ώστε να είναι εύκολη η αναγνώριση του συμβάντος. Επιπλέον, στα συστήματα ασφαλείας οι κάμερες δίνουν συνεχή πρόσβαση στον χώρο για την οπτική επιβεβαίωση του συμβάντος.

Η βασική παράμετρος της αποτελεσματικότητάς του είναι η σωστή κατασκευή, αλλά και ο σχεδιασμός της εταιρείας που θα εγκαταστήσει και θα αναλάβει την παρακολούθηση του συστήματος. Η προστασία ενός χώρου γίνεται πιο αποτελεσματική όταν υπάρχει μια καλά σχεδιασμένη μελέτη, η οποία συνυπολογίζει όλα τα ενδεχόμενα και διαθέτει ένα συνδυασμό μέτρων ασφάλειας, ούτως ώστε να μη δημιουργηθεί κάποιο πρόβλημα στην περίπτωση που υπάρξει αστοχία σε έστω και ένα από τα συστήματα ασφαλείας.

Οι εγκαταστάτες από την πλευρά τους θα πρέπει να γνωρίζουν όσο το δυνατόν καλύτερα τα κριτήρια επιλογής ενός συστήματος ασφαλείας και θα πρέπει να είναι σε θέση να δώσουν τις κατάλληλες συμβουλές, ανάλογα με τις ιδιαιτερότητες και τις ανάγκες της κάθε περίπτωσης. Η προστασία μπορεί να είναι περιμετρική ή εσωτερική, ενώ η διασύνδεση του συστήματος μπορεί να είναι ασύρματη ή ενσύρματη.

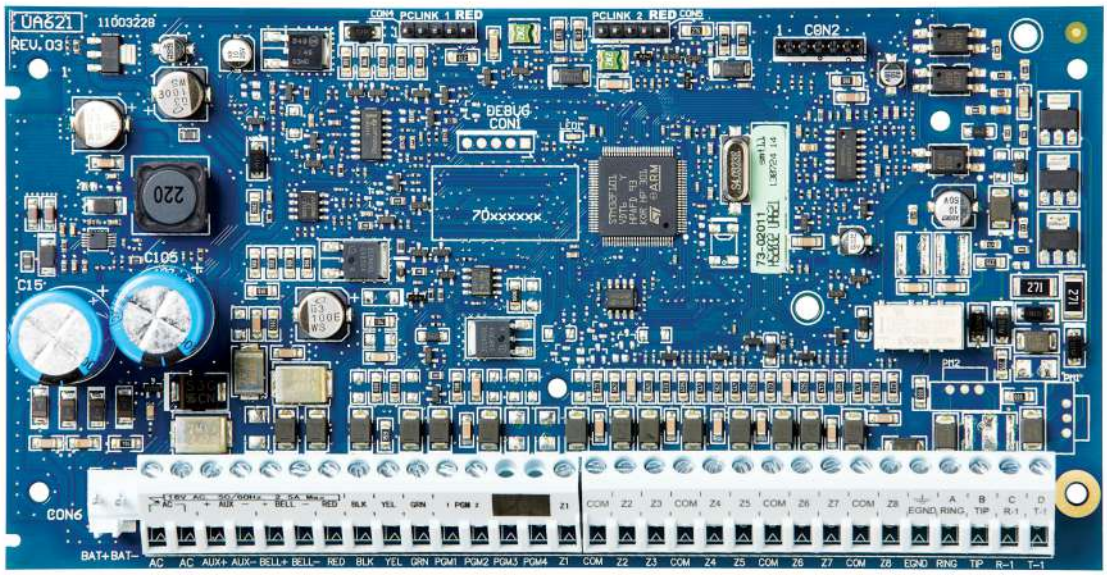
Η επιλογή γίνεται ανάλογα με τις ανάγκες προστασίας του χρήστη και το είδος του συστήματος, και το κόστος αγοράς και εγκατάστασης. Γι’ αυτό, πριν αποφασίσει ο χρήστης θα πρέπει να κάνει μια έρευνα αγοράς και να δει ποιες εναλλακτικές λύσεις έχει, μιας και σήμερα πλέον υπάρχει στην αγορά μεγάλη γκάμα συστημάτων ασφαλείας, για κάθε ανάγκη και για κάθε εφαρμογή. Χρειάζεται λοιπόν να ελέγξει την αξιοπιστία των συσκευών και τις πιστοποιήσεις που διαθέτουν, την τεχνική υποστήριξη που θα πρέπει να συνοδεύει το προϊόν, καθώς και τη δυνατότητα να μπορεί να παρακολουθεί 24 ώρες το 24ωρο το σημείο που ασφαλίζει μέσω κέντρου λήψης σημάτων και πληροφόρησης ακόμα και από μακριά.

2.2 Δομικά στοιχεία ενός συναγερμού

Ένα ολοκληρωμένο ηλεκτρονικό σύστημα συναγερμού αποτελείται από την κεντρική μονάδα ελέγχου, το πληκτρολόγιο, τις εισόδους και τις εξόδους του συστήματος.

Κεντρική μονάδα ελέγχου

Η κεντρική μονάδα είναι συνήθως τοποθετημένη σε ένα μεταλλικό κουτί για προστασία. Επεξεργάζεται τις πληροφορίες που λαμβάνει από τις εισόδους, και ενεργοποιεί τις εξόδους. Σε αυτήν συνδέονται ενσύρματα ή ασύρματα όλα τα εξαρτήματα του συστήματος, όπως αισθητήρες, πληκτρολόγιο, κάρτες επικοινωνίας κτλ και έχει την ευθύνη για τη σωστή λειτουργία του συστήματος ανάλογα με τον προγραμματισμό της. Μια σύγχρονη κεντρική μονάδα πρέπει να υποστηρίζει τουλάχιστον 4 διαφορετικές ζώνες, επίσης πρέπει να υπάρχει μία μπαταρία που αποτελεί την εφεδρική ηλεκτρική παροχή σε περίπτωση διακοπής ρεύματος. Η κεντρική μονάδα μπορεί να έχει τη δυνατότητα να ελέγχει και ηλεκτρικές συσκευές κάτω από ορισμένες συνθήκες. Για παράδειγμα, να ανάβει φώτα της κατοικίας στην περίπτωση συναγερμού. Η τοποθέτηση της σε σωστό σημείο παίζει σημαντικό ρόλο, πρέπει να τοποθετείται σε μέρος που είναι λιγότερο ευάλωτο ώστε να μην εντοπίζεται εύκολα και να υπάρχει ο απαραίτητος χρόνος για να σταλούν τα κατάλληλα σήματα. [34]



Εικόνα 2.1 Κεντρική μοδάδα ελέγχου DSC HS2016

Πληκτρολόγιο

Πρόκειται για μια ειδική είσοδο στο σύστημα. Τα πληκτρολόγια κάνουν πιο απλή και εύκολη τη χρήση των συστημάτων ασφαλείας. Είναι οι μονάδες από τις οποίες ο χρήστης μπορεί να χειριστεί το σύστημα. Προσφέρουν προγραμματιστικές και διαγνωστικές λειτουργίες. Το πάτημα ενός πλήκτρου είναι ικανό να ενεργοποιήσει την περιμετρική ή την ολική προστασία του χώρου και για την απενεργοποίηση του αρκεί ο σχηματισμός του προσωπικού κωδικού. Υπάρχουν τόσο ασύρματα όσο και ενσύρματα πληκτρολόγια. Χρησιμοποιώντας ασύρματα πληκτρολόγια μπορούμε να τα τοποθετήσουμε σε οποιοδήποτε σημείο του χώρου χωρίς να υπάρχει η ανάγκη της καλωδίωσης. [34]

Εικόνα 2.2 Πληκτρολόγιο DSC HS2LCDE6

Είσοδοι του συστήματος

Είναι οι αισθητήρες ή τα κουμπιά έκτακτης ανάγκης του συναγερμού. Ο ρόλος τους είναι να μεταφέρουν πληροφορίες στην κεντρική μονάδα σχετικά με τους κινδύνους, αν για παράδειγμα υπάρχει κάποια κίνηση στο χώρο ή αν άνοιξε κάποιο παράθυρο. Η ποιότητα και η τοποθέτηση τους κατά την εγκατάσταση παίζουν σημαντικό ρόλο και επηρεάζουν τη λειτουργία όλου του συστήματος. Ανιχνευτές χαμηλής ποιότητας ή εγκαταστημένοι με λάθος τρόπο μπορούν να προκαλέσουν αναίτιους συναγερμούς. Οι πιο σημαντικοί αισθητήρες είναι οι μαγνητικές επαφές, ανιχνευτές θραύσης, οι PIR (παθητικοί ανιχνευτές υπέρυθρων), και οι ανιχνευτές καπνού.

Μαγνητικές επαφές συναγερμού:

Η μαγνητική επαφή συναγερμού τοποθετείται σε πόρτες και παράθυρα, και ειδοποιεί την κεντρική μονάδα όταν αυτά είναι ανοιχτά. Λειτουργεί σαν ένας απλός διακόπτης ο οποίος ελέγχεται από ένα εφαρμοσμένο μαγνητικό πεδίο. Όταν τα δύο άκρα της μαγνητικής επαφής βρίσκονται κοντά, τότε ο διακόπτης είναι κλειστός, ενώ όταν απομακρυνθούν ο διακόπτης ανοίγει.

Εικόνα 2.3 Μαγνητική επαφή απλής χρήσης

Αισθητήρες κίνησης PIR:

Αισθητήρας PIR (Passive InfraRed, Παθητικός Υπέρυθρος) είναι ένας αισθητήρας ο οποίος μετράει την υπέρυθρη ακτινοβολία που εκπέμπεται από τα αντικείμενα γύρω του. Ο όρος παθητικός αναφέρεται στο γεγονός ότι ο αισθητήρας δεν παράγει, ούτε ακτινοβολεί κάποια ενέργεια για τη μέτρηση αυτή. Αντίθετα, χρησιμοποιεί την υπέρυθρη ακτινοβολία η οποία εκπέμπεται ή αντανακλάται από τα άλλα αντικείμενα. Η κύρια χρήση των αισθητήρων PIR είναι στους ανιχνευτές κίνησης τύπου PIR ή αλλιώς PID (Passive Infrared Detector, Παθητικός Υπέρυθρος Ανιχνευτής). Σε αυτή την περίπτωση, η αλλαγή στην τιμή της λαμβανόμενης ακτινοβολίας που μετράει ο αισθητήρας ενεργοποιεί την έξοδο του.

Ο ανιχνευτής κίνησης ενεργοποιεί το συναγερμό όταν ανιχνεύσει κίνηση στην περιοχή. Έχουν συνήθως 90 μοίρες βαθμό κάλυψης. Ανιχνεύουν καλύτερα τις κινήσεις που είναι κάθετες στο οπτικό τους πεδίο, έτσι η εγκατάσταση του πρέπει να γίνεται στη γωνία του δωματίου. Μερικοί ανιχνευτές κίνησης είναι δυνατόν να ανιχνεύσουν τα κατοικίδια ζώα και να δώσουν ψευδή συναγερμό, ενώ άλλες έχουν σχεδιαστεί να τα αγνοήσουν, γι' αυτό λοιπόν πρέπει να ελεγχθούν οι προδιαγραφές του ανιχνευτή πριν την αγορά. Σε μια σωστή εγκατάσταση συστήματος συναγερμού θα πρέπει να υπάρχει η δυνατότητα οπλισμού όλων των περιμετρικών ανοιγμάτων (πόρτες, παράθυρα) με αφοπλισμένους τους ανιχνευτές κίνησης. Με τον τρόπο αυτό ακόμα και αν κινείται κανείς στο εσωτερικό της κατοικίας μπορεί να είναι προφυλαγμένος, χωρίς οι ανιχνευτές κίνησης να δίνουν συναγερμό. [35]

Εικόνα 2.4 Αισθητήρας PIR εσωτερικού χώρου

Ενεργητικός υπέρυθρος ανιχνευτής (Infrared Beam):

Αποτελείται από δύο μονάδες, τη μονάδα εκπομπού και τον δέκτη. Ο εκπομπός στέλνει μία παλμική δέσμη υπέρυθρου φωτός στον δέκτη, η οποία αν διακοπεί από κάποιο αντικείμενο ενεργοποιεί την έξοδο του. Χρησιμοποιείται κυρίως σε εξωτερικούς χώρους, λόγω της μεγάλης απόστασης που μπορεί να καλύψει.

Εικόνα 2.5 Infrared Beam της Optex

Ανιχνευτής καπνού ή αερίου:

Οι ανιχνευτές καπνού και αερίου ενεργοποιούν το συναγερμό όταν ανιχνεύσουν καπνό ή αέριο. Οι συσκευές αυτές συνήθως λειτουργούν 24 ώρες ώστε να ειδοποιούν τον ιδιοκτήτη σε περίπτωση πυρκαγιάς. Υπάρχουν δύο κύριοι τύποι ανιχνευτών καπνού:ανιχνευτές ιονισμού και φωτοηλεκτρικοί ανιχνευτές.

Εικόνα 2.6 Ανιχνευτής καπνού της Olympia

Ανιχνευτές Θραύσης:

Οι ανιχνευτές θραύσης κρυστάλλων χωρίζονται σε δύο βασικές κατηγορίες τους ακουστικούς ανιχνευτές θραύσης και τους κραδασμικούς.

Οι ακουστικοί ανιχνευτές θραύσης κρυστάλλων, χρησιμοποιούν συνήθως ένα μικρόφωνο, το οποίο ανιχνεύει κάθε θόρυβο που δημιουργείται μέσα στον επιτηρούμενο χώρο. Τοποθετούνται σε απόσταση από τους υαλοπίνακες διαθέτοντας μια συγκεκριμένη εμβέλεια κάλυψης επιτήρησης χώρου. Εάν οι κραδασμοί υπερβαίνουν ένα ορισμένο όριο ακουστικής έντασης, αυτομάτως

διεγείρεται το υπόλοιπο κύκλωμα του ανιχνευτή, με αποτέλεσμα να αλλάξει κατάσταση η έξοδος ρελέ (από N/C σε N/O) που διαθέτουν, ώστε να αποδοθεί σήμα συναγερμού στον κεντρικό πίνακα. Οι απλούστεροι ανιχνευτές χρησιμοποιούν απλά μικρόφωνα και είναι συντονισμένοι σε τυπικές συχνότητες θραύσης, ώστε να αποδώσουν συναγερμό .Άλλοι ανιχνευτές θραύσης με μικροεπεξεργαστή διαθέτουν πιο σύνθετη λειτουργία, συγκρίνουν τον ήχο που λαμβάνουν με ένα ή περισσότερα προφίλ θραύσης κρυστάλλων.Οι ακουστικοί ανιχνευτές θραύσης τοποθετούνται στον τοίχο ή στην οροφή κοντά στους υαλοπίνακες σε απόσταση σύμφωνα με τις προδιαγραφές που καθορίζονται από τον κατασκευαστή. Κύριο πλεονέκτημα των ακουστικών ανιχνευτών είναι η ευκολία εγκατάστασης, καθώς και το γεγονός ότι μπορούν να επιτηρήσουν παραπάνω από έναν υαλοπίνακες, ανάλογα με τη διαρρύθμιση του χώρου.

Οι κραδασμικοί ανιχνευτές θραύσης είναι η δεύτερη λιγότερο διαδεδομένη κατηγορία. Η λειτουργία τους βασίζεται στο πιεζοηλεκτρικό φαινόμενο και σε αντίθεση με τους προηγούμενους τοποθετούνται πάνω στον υαλοπίνακα. Τοποθετούνται πάντα στις γωνίες των υαλοπινάκων, ενώ το εμβαδόν κάλυψης για μη ανοιγόμενο υαλοπίνακα είναι περίπου τρία μέτρα. Βασικό πλεονέκτημα των ανιχνευτών αυτού του τύπου είναι ότι δεν επηρεάζονται από τη διαρρύθμιση του επιτηρούμενου χώρου ή από τον περιβάλλοντα θόρυβο σε σχέση με τους ακουστικούς. [34]

Εικόνα 2.7 Κραδασμικός ανιχνευτής θραύσης κρυστάλλων της DSC

Έξοδοι του συστήματος

Είναι οι συσκευές που δίνουν την κατάλληλη ενημέρωση στους περαστικούς, τον ιδιοκτήτη ή την εταιρία που επιβλέπει τον συναγερμό ανάλογα τον κίνδυνο που έχει προκύψει. Η ποιο συνηθισμένη έξοδος είναι η σειρήνα. Αυτή έχει κατάλληλη επένδυση για να αντέχει χτυπήματα και τοποθετείται σε όσο το δυνατόν μεγάλο ύψος ώστε να μην μπορεί κάποιος να την ξηλώσει ή έστω να κερδίσει χρόνο ο συναγερμός μέχρι κάποιος να τη φτάσει και να την καταστρέψει. Επιπλέον, τοποθετούνται εσωτερικές σειρήνες οι οποίες διευκολύνουν τον ιδιοκτήτη να αντιληφθεί αμέσως τον συναγερμό εάν βρίσκεται στον χώρο και να δυσκολέψουν την επικοινωνία μεταξύ των εισβολέων. Οι σειρήνες είναι απαραίτητες, όμως σε έναν σύγχρονο συναγερμό δεν είναι αρκετό μόνο η τοπική ενημέρωση που προσφέρει η σειρήνα. Χρειάζεται απομακρυσμένη ενημέρωση που επιτυγχάνεται με τις κάρτες επικοινωνίας GSM/GPRS/IP. Τέλος, μια ακόμη έξοδος με μεγάλα ποσοστά επιτυχίας είναι οι μηχανές ομίχλης. Είναι οι μηχανές οι οποίες όταν ενεργοποιούνται, μέσα σε ελάχιστα δευτερόλεπτα ζεσταίνουν ένα ειδικό υγρό από το οποίο παράγεται πυκνή ομίχλη που δυσκολεύει τους διαρρήκτες να συνεχίσουν την παραβίαση.

Εικόνα 2.8 Τυπική σειρήνα συναγερμού και μηχάνημα παραγωγής ομίχλης

2.3 Κριτίρια επιλογής συναγερμού

Σε κάθε εγκατάσταση συναγερμού γίνεται αξιολόγηση του χώρου από έναν ειδικό για να εντοπιστούν οι ιδιαιτερότητες και οι αδυναμίες του ώστε να γίνει και η επιλογή των υλικών του συναγερμού. Οι ιδιαιτερότητες του χώρου αφορούν συνήθως τη δομή και τη θέση των εισόδων, τα ρεύματα ζεστού ή κρύου αέρα, γυάλινες επιφάνειες, κινούμενα αντικείμενα, πισίνες, δέντρα, κατοικίδια κτλ.

Ενσύρματα-Ασύρματα

Μία από τις βασικές αποφάσεις που παίρνουμε πριν την εγκατάσταση του συναγερμού είναι ο τρόπος επικοινωνίας των στοιχείων του συναγερμού μεταξύ τους, ενσύρματα ή ασύρματα. Συνήθως για να καταλήξουμε λαμβάνουμε υπόψη τις παραμέτρους όπως το κόστος, την εγκατάσταση, την ασφάλεια, την κάλυψη και την επεκτασιμότητα.

Τα ασύρματα συστήματα συναγερμού έχουν μεγαλύτερο κόστος περιφερειακών συσκευών από τα αντίστοιχα ενσύρματα. Από την άλλη, εάν υπολογιστεί το κόστος της εγκατάστασης, της συντήρησης και των καλωδίων, το συνολικό κόστος του ασύρματου μετριάζεται σημαντικά.

Το ασύρματο είναι πολύ πιο βολικό στην εγκατάσταση. Απαιτεί δύο καλώδια προς το δίκτυο ηλεκτροδότησης, σε αντίθεση με τα ενσύρματα που απαιτούν σύνδεση καλωδίων σε όλους τους αισθητήρες και τις συσκευές. Επίσης, μετά από την εγκατάσταση δεν υπάρχουν καλώδια που περνούν μέσα από τους τοίχους, ενώ η αισθητική του κτηρίου παραμένει ίδια με πριν από την εγκατάσταση του συστήματος. Αυτό σημαίνει βέβαια ότι οι μπαταρίες που τοποθετούνται στις συσκευές θα πρέπει να αντικαθίστανται ανά κάποια χρόνια.

Όσον αφορά την ασφάλεια ο ασύρματος συναγερμός μπορεί θεωρητικά να καταστεί μη λειτουργικός από ηλεκτρομαγνητικές παρεμβολές αλλά ακόμα και σε πιο ακραίες περιπτώσεις θα μπορούσε κανείς να κλέψει και κωδικούς που μεταδίδονται μέσω του αέρα. Πρόκειται όμως να καταβάλει κανείς πολύ μεγάλη προσπάθεια και να έχει πολύ εξειδικευμένες δεξιότητες για να το κάνει αυτό. Εάν κάποιος έχει τα προσόντα, το πιο πιθανό είναι ότι αυτά που θα κλέψει, να μην τα χρειάζεται για τα ως προς το ζην. Η δουλειά του συναγερμού είναι να κάνει τη δουλειά του εχθρικού προσώπου όσο πιο δύσκολη γίνεται, σε σημείο που ο χρόνος και το κόστος ενός τέτοιου σεναρίου να το καθιστούν ασύμφορο. Παρόλα αυτά, τα ασύρματα συστήματα οφείλουν να παρουσιάζουν ορθή συμπεριφορά ως συσκευές ηλεκτρομαγνητικής εκπομπής και λήψης δεδομένων. Έτσι και αλλιώς πρέπει να συμμορφώνονται με πρότυπα πάνω και στο ασύρματο κομμάτι.
Οι μπαταρίες, τέλος, πρέπει να αλλάζονται στην ώρα τους, διότι εάν τελειώσουν μπορούν να αφήσουν

απροστάτευτο το συγκεκριμένο χώρο, αλλά ακόμα και εάν είναι σε χαμηλό επίπεδο, ενδέχεται να ανταλλάσσουν ψευδή στοιχεία με τον πίνακα.

Τα ενσύρματα συστήματα έχουν πρακτικά πολύ λίγους περιορισμούς ως προς την ακτίνα κάλυψης, σε σχέση με τα ασύρματα. Από την άλλη, σε κτίρια με πολύ χοντρούς τοίχους, το να φτάσει κανείς ένα καλώδιο σε κάποιους αισθητήρες μπορεί να είναι δύσκολο. Η καλωδίωση ενός υπαίθριου κτιρίου με έναν ενσύρματο συναγερμό, θα είναι επίσης δύσκολη λόγω και της απαιτούμενης καλωδίωσης. Ένα ασύρματο σύστημα συναγερμού μπορεί να μας απαλλάξει από αυτό το πρόβλημα. Μπορεί ο αισθητήρας να τοποθετηθεί σε οποιοδήποτε σημείο, αρκεί να είναι εντός του εύρους του αναμεταδότη, που σημαίνει ότι μπορούν να καλυφθούν περισσότερες περιοχές. Τέλος, σε σημεία με έντονη υγρασία, υψηλές θερμοκρασίες ή καλώδια τροφοδοσίας άλλων συσκευών η καλωδίωση μπορεί να εμφανίσει προβλήματα που θα μας αναγκάσουν σε πολύωρη αναζήτηση του προβλήματος και στην εκ νέου εγκατάσταση καλωδίωσης.

Στο ασύρματο σύστημα είναι πολύ πιο εύκολο να προσθέσει κανείς επιπλέον ανιχνευτές και αισθητήρες, ακόμα και σε μελλοντικό χρόνο από αυτόν της εγκατάστασης. Επίσης, τα είδη των ανιχνεύσεων μπορεί ενδεχομένως να αυξηθούν στο μέλλον. Υπάρχουν πρόσθετα όχι μόνο για τον εντοπισμό εισβολέων, αλλά και ανιχνευτές για άλλες έκτακτες ανάγκες στο σπίτι, όπως η πυρκαγιά και η διαρροή αερίων.

Τέλος, δε θα πρέπει να ξεχνάμε ότι στην αγορά υπάρχουν και υβριδικά συστήματα που μπορούν να συνδυάσουν τα πλεονεκτήματα των ασύρματων και ενσύρματων ανάλογα με τον χώρο που θα καλύψουμε. [35]

Πρότυπο EN 50131

Ένα άλλο κριτήριο που μπορούμε να έχουμε κατα την αγορά συναγερμού είναι η πιστοποίηση που έχει λάβει. Το EN 50131 είναι ένα ευρωπαϊκό πρότυπο που εκδόθηκε το 1997 και αναβαθμίστηκε το 2006 για τις ανάγκες των συναγερμών. Αυτό το πρότυπο περιγράφει τις απαιτήσεις όλων των στοιχείων ενός συναγερμού και τα κατατάσσει σε βαθμίδες ανάλογα με την επίτευξη των απαιτήσεων.

Υπάρχουν τέσσερις βαθμίδες για τους συναγερμούς, από το χαμηλότερο επίπεδο Grade 1 ως το υψηλότερο Grade 4. Συνήθως χρησιμοποιούμε τα Grade 2 και 3. Οι βαθμίδες είναι ένα μέτρο της ανθεκτικότητας σε εξωτερικές επιρροές και σε επιθέσεις από εγκληματίες.

- Grade 1: Οι εισβολείς αναμένεται να έχουν λίγη ή καθόλου εμπειρία.
- Grade 2: Οι εισβολείς αναμένεται να έχουν περισσότερες γνώσεις και κάποιο εξειδικευμένο εξοπλισμό.
- Grade 3: Οι εισβολείς θα διαθέτουν τόσο ολοκληρωμένη γνώση όσο και φορητό ηλεκτρονικό εξοπλισμό.
- Grade 4: οι εισβολείς έχουν εμπειρία, πρόσβαση σε καλό εξοπλισμό και προγραμματισμένη εισβολή.

Ο βαθμός ενός συστήματος καθορίζεται από τη συσκευή με τη χαμηλότερη βαθμίδα. Για παράδειγμα, σε ένα σύστημα με συσκευές Grade 3 αν προσθέσουμε μια συσκευή Grade 2 όλο το σύστημα θα θεωρείται Grade 2. [30]

Κεφάλαιο 3ο: Zigbee

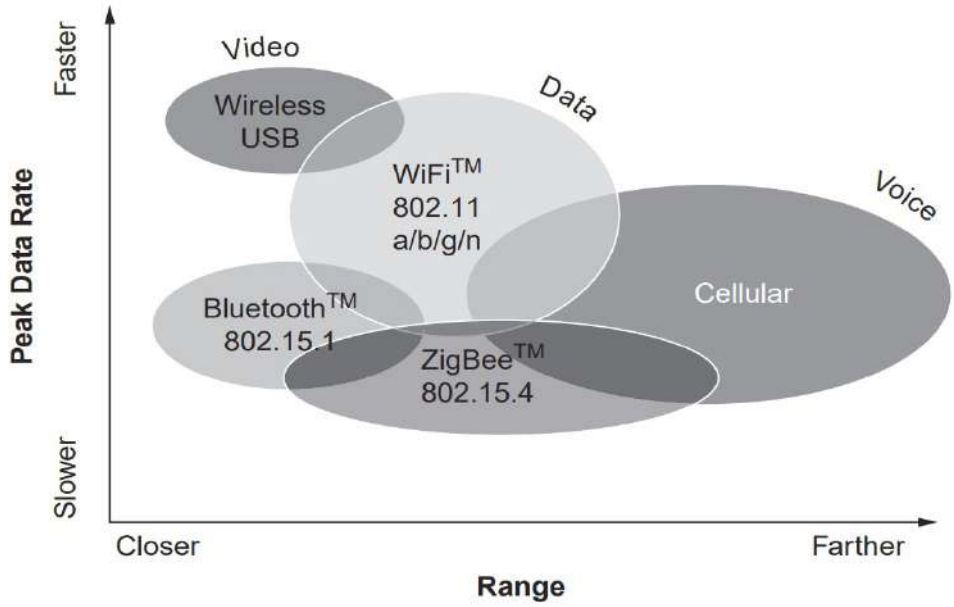
3.1 Εισαγωγή

Το ZigBee είναι ένα πρωτόκολλο επικοινωνίας που έχει σχεδιαστεί για ασύρματες συσκευές δικτύου αισθητήρων. Οι συσκευές Zigbee λειτουργούν στη ζώνη χωρίς άδεια 2,4 GHz, η οποία παρέχει μέγιστο ρυθμό δεδομένων δικτύου ασύρματου αισθητήρα 250 Kbps. Το πρότυπο ZigBee έχει σχεδιαστεί για να παρέχει αξιόπιστη μετάδοση μέτριων ποσοτήτων δεδομένων, ενώ καταναλώνει πολύ λίγη ισχύ. Είναι ένα σύνολο προδιαγραφών που δημιουργήθηκαν πάνω στο πρότυπο 802.12.15 του IEEE.

Τα κύρια πλεονεκτήματά του ZigBee είναι η αξιοπιστία, η οικονομία υλικών, η χαμηλή κατανάλωση ενέργειας και ότι είναι παγκόσμιο πρότυπο. Είναι μια ιδανική λύση για βιομηχανίες ή οικιακές εφαρμογές όπου απαιτείται για την παρακολούθηση ή τον έλεγχο διαφορετικών εργασιών.

3.2 Σύγκριση με άλλα πρωτόκολλα

Υπάρχουν πολλά ασύρματα πρωτόκολλα επικοινωνίας όπως το wifi, το bluetooth, το WiMax, το ασύρματο usb κτλ. Το Zigbee είναι μοναδικό σε σχέση με τα υπόλοιπα διότι καλύπτει έναν τομέα στην αγορά που τα υπόλοιπα πρότυπα δεν στοχεύουν να καλύψουν. Τα υπόλοιπα πρότυπα μπορεί να στοχεύουν στην ταχύτητα μεταδίδοντας μεγάλα δεδομένα ή στη συνεχή προσθήκη καινούργιων δυνατοτήτων ενώ το zigbee στοχεύει σε χαμηλούς ρυθμούς μετάδοσης δεδομένων ελέγχοντας φορτία ή μεταδίδοντας μετρήσεις αισθητήρων, ενώ κρατάει μια απλή μορφή στοίβας που μπορεί να χειριστεί ένας μικροελεγκτής 8-bit με περιορισμένη μνήμη.



Εικόνα 3.1 Σύγκριση ασύρματων τεχνολογιών

Το Bluetooth είναι πρωτόκολλο με υψηλή ασφάλεια που χρησιμοποιείται από ακουστικά, κινητά κτλ και είναι χαμηλού κόστους, όμως λόγο του frequency-hopping η μπαταρία περιορίζεται σημαντικά, σε μερικές ώρες ή μέρες σε αντίθεση με το zigbee που κρατάει μήνες ή χρόνια. Επίσης, ένα δίκτυο bluetooth αποτελείται συνήθως από δύο ή λίγες ακόμη συσκευές ενώ το zigbee μπορεί να έχει εκατοντάδες συσκευές στο ίδιο δίκτυο.

Το Wireless USB είναι μια πολύ οικονομική λύση για την επικοινωνία peer-to-peer που λειτουργεί με μπαταρίες. Μπορεί να υποστηρίξει μέχρι και διπλάσιους ρυθμούς μετάδοσης από το Zigbee με την απόσταση όμως να περιορίζεται σε μερικά μέτρα και να μην υποστηρίζει κάποιο πρότυπο ασφαλείας. Η χρήση του περιορίζεται κυρίως στα περιφερειακά ενός υπολογιστή.

Το Wifi πέρα από τη δικτύωση των υπολογιστών έχει χρησιμοποιηθεί και για δίκτυα αισθητήρων και έλεγχο συσκευών. Είναι λίγο ποιο ακριβό από το Zigbee, χρειάζεται μεγαλύτερους CPU για να τρέξουν ολόκληρο το πρωτόκολλο. Το Wifi υποστηρίζει mesh networking όπως το Zigbee αλλά δεν υπάρχει διαλειτουργικότητα στο τρόπο που οι συσκευές κάνουν mesh networking στο Wifi. Επιπλέον, το Wifi δεν τα πάει καλά με μπαταρίες, φτάνοντας μόνο λίγες ώρες λειτουργίας. [1]

3.3 Χαρακτηριστικά Zigbee

Τα χαρακτηριστικά του ZigBee είναι η αξιοπιστία, η οικονομία υλικών, η χαμηλή κατανάλωση ενέργειας, η υψηλή ασφάλεια και ότι είναι παγκόσμιο πρότυπο.

Αξιοπιστία

Η ασύρματη επικοινωνία γενικά θεωρείται αναξιόπιστη. Όλοι έχουμε αντιμετωπίσει προβλήματα σε κλήσεις τηλεφώνων ή στο σήμα wifi. Είναι αλήθεια ότι τα ραδιοκύματα μπορούν να επηρεαστούν από πολλούς παράγοντες που είτε δεν μπορούμε να προβλέψουμε είτε είναι παράγοντες που μεταβάλλονται με τον χρόνο. Μερικοί από αυτούς τους παράγοντες είναι η απόσταση εκπομπού-λήψης, ο σχεδιασμός των κεραιών, η ισχύς μετάδοσης, οι καιρικές συνθήκες και τα υλικά που παρεμβάλλονται και επηρεάζουν τη συγκεκριμένη συχνότητα όπως μέταλλα, νερό, τσιμέντο κτλ.

Στις περιπτώσεις των κινητών και του wifi συνήθως μετακινούμαστε μέχρι να βρούμε σημείο με καλό σήμα. Το Zigbee έχει άλλους τρόπους για να εξασφαλίζει μόνο του την ασύρματη σύνδεση. Το Zigbee πετυχαίνει υψηλή αξιοπιστία με τους παρακάτω τρόπους:

- IEEE 802.15.4 με O-QPSK και DSSS
- CSMA-CA
- 16-bit CRCs
- Acknowledgments σε κάθε hop
- Mesh networking για την εύρεση της καλύτερης διαδρομής(route)
- End-to-end acknowledgments για την εξασφάλιση ότι τα δεδομένα έφτασαν στον προορισμό τους

Το Zigbee χρησιμοποιεί το Carrier Sense Multiple Access Collision Avoidance (CSMA-CA) για να αυξήσει την αξιοπιστία. Πρίν αρχίσει να μεταδίδει ένα πακέτο ακούει το κανάλι επικοινωνίας ώστε να αρχίσει την μετάδοση μόνο όταν το κανάλι είναι καθαρό.

Επίσης, χρησιμοποιεί 16-bit CRC σε κάθε πακέτο (FCS, Frame Checksum) για να εξασφαλίσει την ακεραιότητα των bits. Κάθε πακέτο ξαναστέλνεται έως τρεις φορές σε περίπτωση αποτυχίας ενώ στην τέταρτη αποτυχία ενημερώνεται ο κόμβος που μεταδίδει το πακέτο ώστε να κάνει τις απαραίτητες ενέργειες.

Ένας άλλος τρόπος που το Zigbee αυξάνει την αξιοπιστία είναι το mesh network. Το mesh network προσφέρει δυνατότητες όπως την αύξηση της εμβέλειας με το multi-hop και την αυτόματη εύρεση της καλύτερης διαδρομής μετάδοσης του πακέτου και την αποκατάσταση της διαδρομής εάν χαθεί(automatic route discovery and self healing). Με το mesh network τα δεδομένα μπορούν να σταλούν ανεξάρτητα της απόστασης εκπομπού-λήψης, αρκεί να υπάρχουν ενδιάμεσα αρκετοί κόμβοι που να κάνουν αναμετάδοση.

Το Zigbee επίσης χρησιμοποιεί τον μηχανισμό end-to-end acknowledgments. Ενεργοποιώντας αυτή τη λειτουργία η εφαρμογή zigbee λαμβάνει πάντα τα δεδομένα χωρίς αλλοιώσεις. [1]

Οικονομική λύση

Μια εφαρμογή Zigbee περιλαμβάνει απαραίτητα την στοίβα Zigbee και τον συνδυασμό MCU με τον εκπομπό. Κοιτάζοντας τις τρέχουσες τιμές από επίσημους διανομείς, οι τιμές για τα SoC είναι μερικά ευρώ. Πέρα όμως από τα φθηνά υλικά το Zigbee αποτελεί μια φθηνή λύση σε ασύρματες εφαρμογές και για άλλους λόγους. Για παράδειγμα, το Zigbee λειτουργεί στη συχνότητα 2.4GHz η οποία είναι παγκοσμίως συχνότητα χωρίς την ανάγκη για ειδική άδεια. Επιπλέον, τα υλικά που απαιτούνται για την ανάπτυξη μιας εφαρμογής Zigbee, όπως τα εργαλεία σε μορφή λογισμικού, οι debbugers, ο compiler, η Zigbee στοίβα, τα απαραίτητα έγγραφα κτλ είτε δεν έχουν μεγάλο κόστος είτε είναι δωρεάν.

Ένας άλλος τρόπος οικονομίας του Zigbee είναι το γεγονός ότι αποτελεί παγκόσμιο πρότυπο. Αυτό οδηγεί πολλές εταιρίες που κατασκευάζουν modules να σχεδιάσουν μεγάλη ποικιλία προϊόντων αυξάνοντας έτσι των ανταγωνισμό και μειώνοντας τις τιμές. Τα modules είναι συνδυασμός MCU με μια πλατφόρμα Zigbee που τα καθιστά έτοιμα να λειτουργήσουν, περιορίζοντας την ανάπτυξη της εφαρμογής μόνο στο κομμάτι του λογισμικού εξασφαλίζοντας χρόνο εργασίας. [1]

Χαμηλή κατανάλωση ενέργειας

Οι συσκευές Zigbee χρησιμοποιούν κατα κύριο λόγο μπαταρίες για την τροφοδοσία τους. Οι μπαταρίες σε μια συσκευή Zigbee μπορούν να διαρκέσουν μερικούς μήνες ή και χρόνια ανάλογα τη χρήση τους. Αυτή τη μικρή κατανάλωση την πετυχαίνει το Zigbee, πέρα από το sleep mode σε μικροελεγκτή και εκπομπό, με το χαμηλό duty cycle. Ένας κόμβος σε ένα δίκτυο Zigbee δεν χρειάζεται να κρατάει σταθερή σύνδεση με το δίκτυο για να παραμένει στο δίκτυο. Στην πραγματικότητα, τα δίκτυα Zigbee την περισσότερη ώρα είναι απολύτως σιωπηλά, δεν ανταλλάσουν ούτε ένα πακέτο. Ένας αισθητήρας συνήθως θα ενημερώσει ανά κάποια διαστήματα για την τιμή που διαβάζει, εκτός αν αλλάξει απότομα η τιμή. Στους συναγερμούς συγκεκριμένα οι κόμβοι-αισθητήρες χρειάζεται να επικοινωνούν μόνο όταν ανοίγει ή κλείνει μια πόρτα ή ένα παράθυρο ή όταν ανιχνεύεται κίνηση στον χώρο. [1]

Υψηλή ασφάλεια

Για την ασφάλεια του δικτύου το Zigbee χρησιμοποιεί το παγκοσμίως αναγνωρισμένο πρότυπο ασφαλείας Advanced Encryption Standard (AES). Είναι ένα πρότυπο που κρυπτογραφεί και αποκρυπτογραφεί πακέτα με τρόπο που είναι αδύνατο να παραβιαστούν. Έχει επιλεγεί από το Zigbee διότι είναι αναγνωρισμένα αξιόπιστο, είναι ελεύθερης πατέντας και υλοποιείται εύκολα στους μικροελεγκτές 8-bit.

Το Zigbee παρέχει τόσο τη λειτουργία της κρυπτογράφησης, που σημαίνει ότι τα πακέτα δε γίνονται κατανοητά από κόμβους που απλώς ακούνε τα πακέτα στον αέρα και δεν έχουν το απαραίτητο κλειδί, όσο και την λειτουργία της αυθεντικοποίησης που σημαίνει ότι ένας κακόβουλος κόμβος δεν μπορεί να εισάγει κάποιο πακέτο στο δίκτυο και το δίκτυο να ανταποκριθεί σε αυτό.

Το Zigbee χρησιμοποιεί ένα 128-bit κλειδί που ονομάζεται network key. Εφόσον μια συσκευή είναι αξιόπιστη της δίνεται το network key και μπορεί να ανταλλάσσει πακέτα με το δίκτυο. Κάποιες εφαρμογές όμως χρειάζονται επιπλέον ασφάλεια εντός του δικτύου. Για παράδειγμα, μπορεί πολλά άτομα να μοιράζονται το ίδιο δίκτυο αλλά ο καθένας να έχει τα προσωπικά του δεδομένα. Σε αυτή την περίπτωση χρησιμοποιείται ένα δεύτερο κλειδί(Link key) που προστατεύει το φορτίο APS. [1]

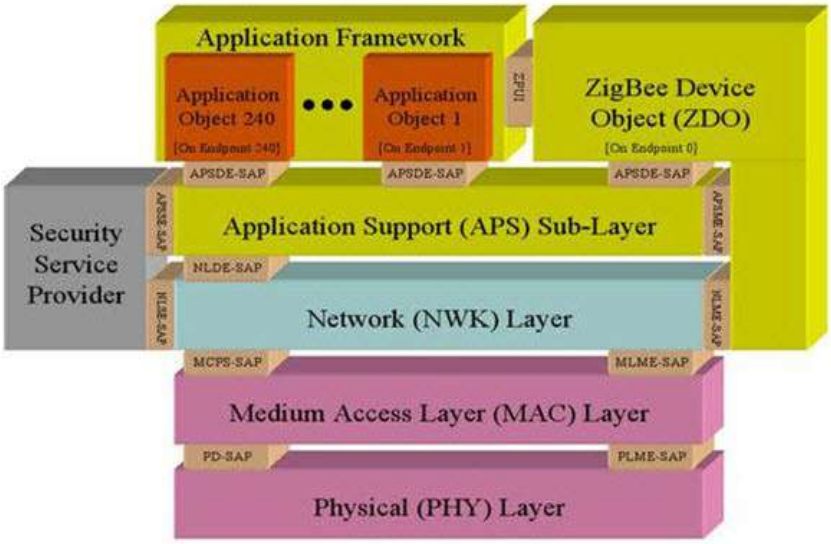
Παγκόσμιο πρότυπο

Το Zigbee είναι παγκόσμιο πρότυπο και το specification document είναι διαθέσιμο δωρεάν στην επίσημη ιστοσελίδα της Zigbee Alliance. Αυτό σημαίνει ότι οι εταιρίες μπορούν να μελετήσουν το πρότυπο που έχει ορίσει η Zigbee και να σχεδιάσουν τη δική τους υλοποίηση. Η Zigbee Alliance έχει καθορίσει τα χαρακτηριστικά που πρέπει να έχει μια στοίβα Zigbee και η συσκευή που θα τρέχει τη στοίβα ώστε να θεωρείται πιστοποιημένη πλατφόρμα(ZigBee Compliant Platform). Επιπλέον, καθορίζει τη συμβατότητα των συσκευών Zigbee σε επίπεδο εφαρμογής μέσω των applications profiles. Το χαρακτηριστικό του παγκόσμιου πρότυπου εξασφαλίζει την διαλειτουργικότητα μεταξύ

συσκευών Zigbee που προέρχονται απο διαφορετικούς κατασκευαστές δίνοντας την δυνατότητα στον τελικό χρήστη να φτιάξει το σύστημα του με τις συσκευές που επιθυμεί. [1]

3.4 Αρχιτεκτονική της στοίβας Zigbee

Το Zigbee όπως και άλλα πρωτόκολλα οργανώνεται σε επίπεδα. Μοιάζει με το μοντέλο OSI 7, έχει τα επίπεδα Physical, MAC, Network αλλά πάνω σε αυτά προστίθενται και τα APS και ZDO. Κάθε επίπεδο διεκπεραιώνει μια συγκεκριμένη ομάδα λειτουργιών χωρίς να γνωρίζει τις λειτουργίες του κάτω επιπέδου.



Εικόνα 3.2 Η αρχιτεκτονική Zigbee

Ανάμεσα στα επίπεδα υπάρχουν τα Service Access Points(SAPs). Τα SAPs παρέχουν ένα API ανάμεσα στα δύο γειτονικά επίπεδα. Όπως και στο IEEE 802.15.4, το Zigbee χρησιμοποιεί δύο SAP σε κάθε επίπεδο, ένα για τα δεδομένα και ένα για τη διαχείριση. Για παράδειγμα, τα δεδομένα του επιπέδου Network περνάνε από το Network Layer Data Entity Service Access Point (NLDE-SAP). Έτσι προκύπτουν και τα primitives που ορίζονται στο specification του Zigbee όπως το APSDE-DATA.request η οποία είναι μια αίτηση που δημιουργήθηκε πάνω από το επίπεδο APS, δίνεται μέσω του data SAP του APS και δίνει την εντολή να σταλούν δεδομένα μέσω του εκπομπού.

Τα δύο κατώτερα επίπεδα, Physical και MAC, καθορίζονται από το IEEE 802.15.4. Το φυσικό επίπεδο είναι υπεύθυνο για τη μετάφραση των πακέτων σε μια ακολουθία bits που μεταδίδονται στον αέρα και το αντίστροφο.Το επίπεδο MAC αναλαμβάνει την peer-to-peer μετάδοση αναλαμβάνοντας κάποιες λειτουργίες όπως αναμεταδόσεις σε περίπτωση αποτυχίας μετάδοσης του πακέτου, per-hop acknowledgments και τον μηχανισμό CSMA-CA.

Το Network επίπεδο αναλαμβάνει τη λειτουργία του mesh networking. Αυτό περιλαμβάνει τη μετάδοση broadcast μηνυμάτων στο δίκτυο, την εύρεση της καλύτερης διαδρομής για την μετάδοση ενός πακέτου και γενικά΄το Network επίπεδο εξασφαλίζει την αξιόπιστη μετάδοση του πακέτου σε οποιονδήποτε κόμβο του δικτύου. Επίσης, αυτό το επίπεδο περιλαμβάνει εντολές ασφαλείας περιλαμβανομένων των secure joining και rejoining. Γενικότερα όλα τα δίκτυα Zigbee παρέχουν προστασία σε αυτό το επίπεδο κρυπτογραφώντας όλο το πλαίσιο που μεταδίδεται.

Την τοπική και over-the-air διαχείριση του δικτύου την αναλαμβάνει το επίπεδο ZDO. Παρέχει λειτουργίες για την εύρεση μιας συσκευής ή κάποιας υπηρεσίας στο δίκτυο και κρατάει την τρέχουσα κατάσταση δικτύου του κόμβου.

Το APS είναι το απαραίτητο επίπεδο για την εφαρμογή. Προσφέρει υπηρεσίες στις εφαρμογές που υλοποιούνται από πάνω, στα endpoints. Φιλτράρει τα μηνύματα που στέλνονται από το NWK επίπεδο, απορρίπτει τα διπλά μηνύματα και κατευθύνει τα μηνύματα στα σωστά endpoints. Επίσης κρατάει τον τοπικό binding table.

Το Application Framework περιέχει την ZigBee Cluster Library(βιβλιοθήκη καθορισμένη από το Zigbee για ένα σύνολο εντολών) και τα endpoints όπου υλοποιούνται οι εφαρμογές. [1]

3.5 Βασικά στοιχεία Zigbee

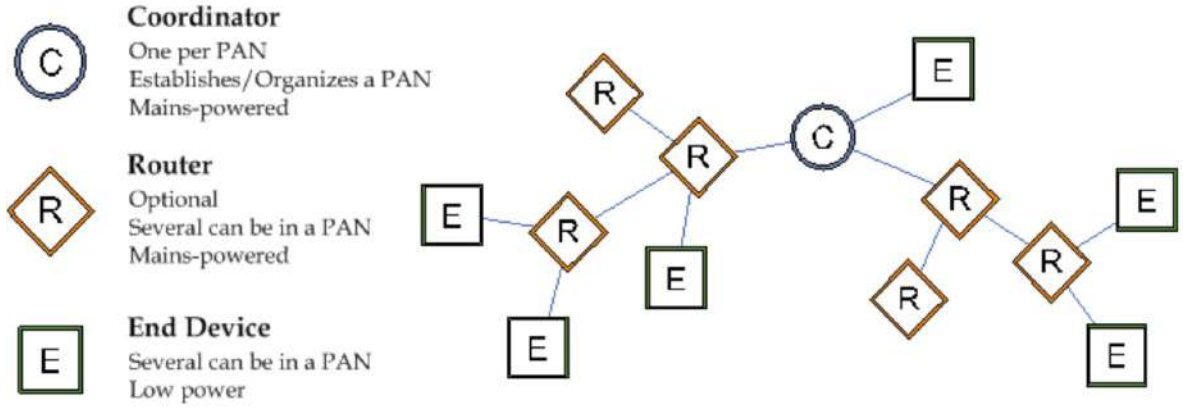
Τα είδη συσκευών ενός δικτύου Zigbee

Το είδος της συσκευής σε ένα δίκτυο zigbee καθορίζεται στη διάρκεια του compile και δεν αλλάζει κατα τη λειτουργία του δικτύου. Μια συσκευή μπορεί να είναι ο coordinator, ένας router ή end-device. Οι συσκευές coordinator και router είναι FFD(Full Function Device) ενώ η end-device είναι RFD(Reduced Function Device).

Coordinator: Ο coordinator είναι μοναδικός σε ένα δίκτυο. Μόνο ο coordinator έχει τη δυνατότητα να δημιουργεί ένα δίκτυο zigbee. Για να δημιουργήσει το δίκτυο σκανάρει την γύρω περιοχή για άλλα δίκτυα, επιλέγει ένα κανάλι λειτουργίας και ένα PAN ID. Μετά τη δημιουργία του δικτύου συμπεριφέρεται σαν ένας απλός router. Στο Zigbee 3 δημιουργείται ένα centralized δίκτυο όπου ο coordinator έχει τον ρόλο του Trust Center. Εκτός της δημιουργίας δικτύου και της δρομολόγησης πακέτων ο coordinator είναι υπεύθυνος για την ασφάλεια του δικτύου και την διακίνηση των κλειδιών για την αποδοχή ή απόρριψη μιας συσκευής στο δίκτυο.

Router: Ένας router χρησιμοποιείται για την επέκταση της εμβέλειας του δικτύου. Αυξάνει τις ικανότητες του mesh networking κάνοντας το δίκτυο ακόμη ποιο αξιόπιστο. Αφού συνδεθεί σε κάποιο δίκτυο οι βασικές του λειτουργίες είναι να δρομολογεί πακέτα και να ανταποκρίνεται σε συσκευές που επιθυμούν να συνδεθούν στο δίκτυο. Επιπλέον, μπορεί να διατηρεί συνδέσεις με “childs” για τα οποία κρατάει πακέτα σε buffer έως ότου είναι διαθέσιμα για μετάδοση. Μια συσκευή router είναι πάντα συνδεδεμένη σε σταθερή πηγή ενέργειας ώστε να είναι συνέχεια διαθέσιμη.

End-device: Οι συσκευές end-devices λειτουργούν με μπαταρίες και δε συμμετέχουν στις λειτουργίες του δικτύου. Είναι οι συσκευές που συνήθως εκτελούν τις επιθυμητές εφαρμογές, όπως μετρήσεις αισθητήρων και έλεγχος φορτίων. Μια end-device συσκευή μπορεί να βρίσκεται σε κατάσταση αδράνειας για την επίτευξη χαμηλής ενέργειας. Με αυτό τον τρόπο τα μηνύματα που στέλνονται προς την end-device συγκεντρώνονται στον κόμβο “parent” της end-device απο όπου θα τα λάβει με την μέθοδο polling όταν η end-device βγεί απο την κατάσταση αδράνειας. Η διάρκεια της αδράνειας μιας end-device μπορεί να παραμετροποιηθεί ή μπορεί να ρυθμιστεί και έτσι ώστε να λαμβάνει αμέσως τα μηνύματα με αντίκτυπο την διάρκεια της μπαταρίας. Σε κάθε περίπτωση η end-device μπορεί να βγαίνει από την κατάσταση αδράνειας για να στείλει τα επιθυμητά μηνύματα. [23]



Εικόνα 3.3 Παράδειγμα δικτύου Zigbee

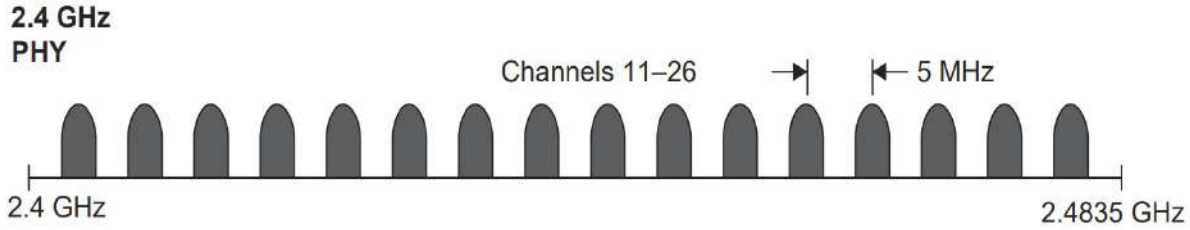
Channels

Το Zigbee χρησιμοποιεί την 2.4GHz μπάντα η οποία είναι παγκοσμίως ελεύθερη για παραγωγή προϊόντων χωρίς ειδική άδεια. Η ίδια μπάντα χρησιμοποιείται από το wifi, το bluetooth και άλλες συσκευές. Όπως καθορίζεται στο 802.18.4, η 2.4GHz μπάντα χωρίζεται στα κανάλια 11 έως 26. Τα κανάλια είναι ένα τμήμα του φάσματος που διαχωρίζονται μεταξύ τους με ένα διάστημα 5MHz. Το 802.15.4 χρησιμοποιεί την μέθοδο Direct Sequence Spread Spectrum (DSSS) για να μεταδώσει τα πακέτα σε σύμβολα και να τα ξανά συναρμολογήσει στο άλλο άκρο επαληθεύοντας τη σωστή αποκωδικοποίηση των δεδομένων με ένα 16-bit CRC. [4]

Οι πομποδέκτες 802.15.4 είναι half-dublex[4], δηλαδή μπορούν να λαμβάνουν και να μεταδίδουν αλλά όχι ταυτόχρονα. Επιπλέον, μπορούν να λαμβάνουν πακέτα μόνο σε ένα κανάλι κάθε φορά. Το Zigbee είναι πρωτόκολλο που δεν κάνει συχνά μεταβάσεις από κανάλι σε κανάλι. Αντί αυτού χρησιμοποιεί τις μεθόδους O-QPSK και DSSS ώστε να μπορεί να σταθεί ακόμη και σε θορυβώδη RF περιβάλλον.

Μια συσκευή Zigbee επιλέγει το κανάλι που θα λειτουργήσει μέσω του Application Profile. Κάποια Application Profiles καθορίζουν σε ποιο κανάλι πρέπει να λειτουργήσει η συσκευή, ενώ στα private profiles η εφαρμογή μπορεί να βρίσκεται σε οποιοδήποτε κανάλι.

Η διαδικασία εύρεσης καναλιού απαιτεί κάποιο χρόνο ο οποίος αυξάνεται αρκετά δευτερόλεπτα εάν προτιμήσουμε να σκανάρουμε όλα τα διαθέσιμα κανάλια. Το Zigbee εκτελεί δύο σαρώσεις, την energy detect scan και το active scan. Η σάρωση energy detect χρησιμοποιείται για να καθοριστούν τα κανάλια που είναι ποιο ήσυχα ενώ στη σάρωση active scan στέλνεται ένα beacon requeset για να εντοπιστούν τυχόν άλλα δίκτυα 802.15.4 στην εμβέλεια του πομποδέκτη. [1]



Εικόνα 3.4 Τα κανάλια IEEE 802.15.4 2.4GHz

PAN ID

Τα PAN ID(Personal Area Network identifiers) χρησιμοποιούνται για να διαχωρίζουν με φυσικό τρόπο μια ομάδα zigbee κόμβων από μια άλλη που βρίσκεται στην ίδια περιοχή ή στο ίδιο κανάλι. Αυτό επιτρέπει σε δύο δίκτυα να συνυπάρχουν σε κοντινή απόσταση χωρίς παρεμβολές μεταξύ τους και να μοιράζονται το ίδιο εύρος ζώνης. Στο Zigbee το PAN ID είναι ένας αριθμός 16-bit με εμβέλεια 0x0000-0x3fff. Γενικά η πιθανότητα σύγκρουσης δύο ίδιων PAN ID είναι απίθανη καθώς και οι στοίβες Zigbee είναι σχεδιασμένες να διαλέγουν τυχαία PAN ID. Η τιμή 0xffff στην περίπτωση δημιουργίας δικτύου σημαίνει οτι το PAN ID θα επιλεγεί τυχαία ενώ στην περίπτωση σύνδεσης μιας συσκευής στο δίκτυο σημαίνει ότι θα επιλέξει ένα δίκτυο με οποιοδήποτε PAN ID. Μια εφαρμογή επιλέγει το PAN ID μέσω του Application Profile. Σε ένα Private Profile μπορεί να χρησιμοποιηθεί οποιοδήποτε PAN ID. [1]

Extended PAN Ids

Το extended PAN ID είναι ανεξάρτητο από το PAN ID και τη διεύθυνση MAC. Είναι ένας αριθμός μήκους 64-bit που βοηθάει τον κόμβο που επιθυμεί να συνδεθεί στο δίκτυο, να επιλέξει το σωστό δίκτυο. Για παράδειγμα, μια εφαρμογή μπορεί να αναζητάει ένα δίκτυο από συγκεκριμένο κατασκευαστή. Οι κατασκευαστές δεσμεύουν μερικά bits του extended PAN ID για αυτό τον σκοπό.

Ενώ σε όλες τις επικοινωνίες στο δίκτυο χρησιμοποιείται το PAN ID, υπάρχει μια περίπτωση που χρησιμοποιείται το 64-bit PAN ID. Αυτή η περίπτωση είναι κατα την σύνδεση του καινούργιου κόμβου που κάνει το beacon request. Στο beacon response που θα λάβει περιέχεται το extended PAN ID το οποίο εξετάζει για να αποφασίσει εάν θα συνδεθεί τελικά στο δίκτυο. [1],[12]

Profiles

Τα profiles ID είναι 16-bit αριθμοί στο διάστημα 0x0000-0x7fff για τα public profiles και στο διάστημα 0xbf00-0xffff για τα manufacturer specific profiles.

Τα profiles προσθέτουν μια σημαντική λειτουργία στο Zigbee, την διαλειτουργικότητα. Είναι ένας τρόπος να ομαδοποιούμε συσκευές με παρόμοια χαρακτηριστικά ανεξάρτητα τον κατασκευαστή. Τα public profiles είναι αυτά που καθορίζονται από την Zigbee Alliance. Το Home Automation είναι ένα από αυτά το οποίο καθορίζει συσκευές που προορίζονται για χρήση στο σπίτι όπως φώτα, διακόπτες, πρίζες, θερμοστάτες κτλ.
Σε ένα δίκτυο μπορεί να υπάρχει απεριόριστος αριθμός profiles, τόσο public όσο και private. Για την ακρίβεια μπορούν να υλοποιούνται πολλά profiles σε κάθε endpoint του εκάστοτε κόμβου. [1],[5]

Device ID

Είναι ένας αριθμός με τιμές στο διάστημα 0x0000-0xffff που δίνει την περιγραφή της φυσικής μορφής της συσκευής. Χρησιμεύει στο να δίνει μια κατανοητή περιγραφή για τον άνθρωπο. Για παράδειγμα, μια λάμπα και μια πρίζα μπορούν να δουλεύουν με τον ίδιο τρόπο χρησιμοποιώντας τον ίδιο cluster OnOff με id 0x0006 αλλά απο την οπτική γωνία του τελικού χρήστη είναι δύο εντελώς διαφορετικές συσκευές. [1]

3.6 Διευθυνσιοδότηση στο Zigbee

Οι συσκευές Zigbee έχουν δύο ειδών διευθύνσεις, τις 64-bit MAC address και τις 16-bit network address. Οι διευθύνσεις MAC δίνονται κατα την κατασκευή της συσκευής από τον κατασκευαστή και είναι παγκοσμίως μοναδικές, ενώ οι network address δίνονται κατα την σύνδεση μιας συσκευής στο δίκτυο και είναι μοναδικές εντός του δικτύου.

Στο Zigbee Pro χρησιμοποιούνται τυχαίες διευθύνσεις. Όποτε συνδέεται μια καινούργια συσκευή στο δίκτυο λαμβάνει με τυχαίο τρόπο την 16-bit διεύθυνση απο τον γονέα-parent. Στην συνέχεια αυτή η καινούργια συσκευή πραγματοποιεί ένα Device Announcement προς το υπόλοιπο δίκτυο για να ανακοινώσει την παρουσία του. Αυτό το πακέτο Device Announce περιλαμβάνει τις διευθύνσεις 16-bit και 64-bit ώστε εάν παρουσιαστεί σύγκρουση στις 16-bit διευθύνσεις μεταξύ δύο συσκευών να προχωρήσουν σε αλλαγή διεύθυνσης. Στις συσκευές end devices, τις περιπτώσεις Address Conflict και Device Announcement τις χειρίζεται ο γονέας. Ο Coordinator έχει πάντα την διεύθυνση 0x0000. Μια συσκευή μπορεί να στείλει μήνυμα σε οποιαδήποτε άλλη συσκευή του δικτύου γνωρίζοντας μόνο την 16-bit διεύθυνση προορισμού, δεν χρειάζεται επιπλέον διαδικασία επικοινωνίας. Αυτός είναι και ο λόγος που η επικοινωνίας των κόμβων με τον coordinator είναι συχνή και άμεση, διότι η 16-bit διεύθυνση είναι πάντα γνωστή. [1]

Groups

Οι ομαδοποίηση(Groups) είναι ένας τρόπος να απευθυνόμαστε σε πολλές συσκευές ταυτόχρονα με μία εντολή. Τα Groups είναι προεραιτικά για την λειτουργία του Zigbee, είναι όμως απαραίτητα σε ορισμένα public profiles όπως το Home Automation Profile αλλά και σε ορισμένες καταστάσεις που απαιτείται ομαδική ενέργεια των συσκευών με μία εντολή. Σε έναν συναγερμό για παράδειγμα μπορούν να δημιουργηθούν partitions. Μπορούμε δηλαδή σε μια ομάδα να έχουμε τα εσωτερικά radar και σε κάποια άλλη ομάδα τα περιφερειακά μαγνητικά. Με αυτόν τον τρόπο μπορούμε να τα οπλίζουμε ανεξάρτητα τα δύο partitions ανάλογα με το αν κυκλοφορούμε στο εσωτερικό του σπιτιού ή βρισκόμαστε μακριά από αυτό.

Σε μια συσκευή μπορεί να υπάρχουν endpoints που συμμετέχουν σε διάφορα groups. Κάθε φορά που ένας κόμβος λαμβάνει ένα groupcast εξετάζει το group ID για να το στείλει στο κατάλληλο endpoint ή να το απορρίψει. [1]

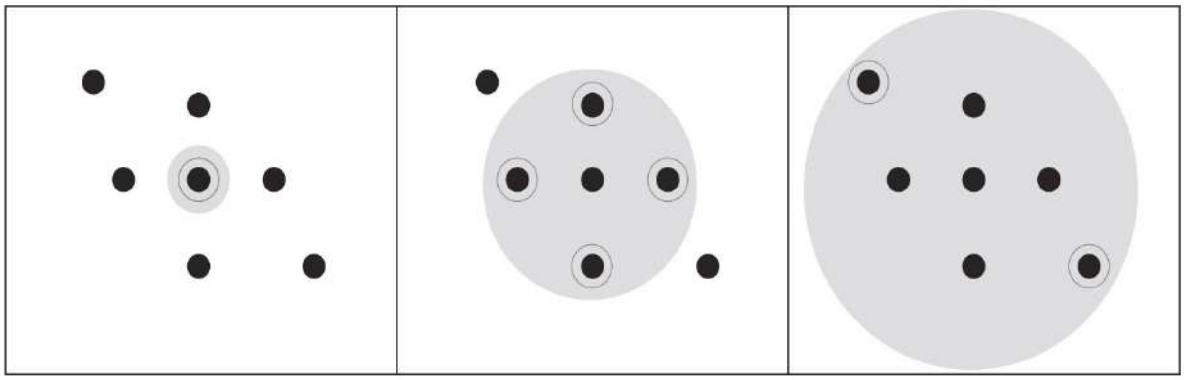
Broadcasts

Ένα μήνυμα broadcast χρησιμοποιείται για να σταλούν δεδομένα σε ολόκληρο το δίκτυο ή σε κάποια καθορισμένη ακτίνα. Υπάρχουν τριών ειδών μηνύματα broadcast:

-0xffff — Broadcast σε όλους τους κόμβους
-0xfffd — Broadcast σε όλους τους κόμβους που δεν είναι σε κατάσταση αδράνειας
-0xfffc — Broadcast μόνο στους ρουτερ
Τα μηνύματα broadcast χρησιμοποιούνται και από μερικές λειτουργίες του δικτύου όπως η εύρεση διαδρομής για την αποστολή πακέτου(route discovery) και την αναζήτηση συγκεκριμένης 16-bit διεύθυνσης(NWK_addr_req).

Οι εφαρμογές θα πρέπει να χρησιμοποιούν τα μηνύματα broadcast με προσοχή. Ένα δίκτυο Zigbee μπορεί να διαχειριστεί συγκεκριμένο αριθμό broadcasts κάθε φορά. Κάθε μήνυμα broadcast παρακολουθείται από τον πίνακα Broadcast Transaction Table (BTT) όπου ο ελάχιστος αριθμός καταχωρίσεων καθορίζεται από το stack profile. Εάν το δίκτυο μεταδίδει συνεχώς broadcast μηνύματα υπάρχει περίπτωση να βρεθεί συσκευή που μεταδίδει broadcast μηνύματα η οποία δεν έχει επιπλέον διαθέσιμες πηγές(RAM) με αποτέλεσμα να χαθεί η αναμετάδοση.

Ένα μήνυμα broadcast με radius 1 ή 2 είναι ένας συνηθισμένος τρόπος επικοινωνίας. Ένα δίκτυο μπορεί να έχει μήκος αρκετές δεκάδες κόμβους. Με τον παραπάνω τρόπο επιτυγχάνεται επικοινωνία μεταξύ γειτονικών κόμβων. [1]



Εικόνα 3.5 Μετάδοση μηνύματος broadcast

3.7 Διευθυνσιοδότηση εντός του κόμβου

Endpoints

Σε έναν κόμβο περιλαμβάνονται τα endpoints. Τα endpoints αριθμούνται απο το 1 έως το 240 και το καθένα καθορίζει την εφαρμογή που βρίσκεται σε αυτά. Κάθε κόμβος δηλαδή μπορεί να υλοποιεί πολλές διαφορετικές εφαρμογές με τη βοήθεια των endpoints. Με αυτό τον τρόπο ένας κόμβος μπορεί να χρησιμοποιεί πολλά διαφορετικά application profiles και να διαχωρίζει διαφορετικές συσκευές που βρίσκονται στον ίδιο κόμβο και μοιράζονται τον ίδιο πομποδέκτη Zigbee.

Για παράδειγμα, ένας διακόπτης μπορεί να πωλείται για οικιακή και εμπορική χρήση υποστηρίζοντας τα Home Automation και Commercial Building Automation profiles σε δυο διαφορετικά endpoints. Άλλο ένα παράδειγμα μπορεί να είναι μια συσκευή διαχείρισης θερμοκρασίας στο σπίτι. Αυτή η συσκευή μπορεί να περιλαμβάνει αισθητήρες και ενεργοποιητές για την αυξομείωση της θερμοκρασίας που κάθε ένα από αυτά αντιμετωπίζεται σαν διαφορετική συσκευή απο το Zigbee σε ξεχωριστό endpoint.

Επιπλέον υπάρχει ένα ειδικό endpoint που καλείται ως broadcast endpoint. Στέλνοντας μηνύματα στο broadcast endpoint απυθευνόμαστε σε όλα τα endpoints.

Κάθε endpoint περιγράφεται λεπτομερώς από μια δομή που ονομάζεται Simple Descriptor η οποία περιλαμβάνει στοιχεία όπως το υποστηριζόμενο profile του endpoint, τις εντολές cluster και το Device ID. [1]

Clusters

Τα Clusters χρησιμοποιούνται κυρίως από τα public profiles, καθορίζονται από την Zigbee Alliance μέσω της ZigBee Cluster Library(ZCL), αναγνωρίζονται από ένα 16-bit ID και αφορούν έννοιες σχετικά με την εφαρμογή. Κοιτάζοντας τα πακέτα που μεταδίδονται στον αέρα, τα ZCL πλαίσια αποτελούν το payload του πλαισίου APS, είναι δηλαδή το ωφέλιμο φορτίο της εκάστοτε εφαρμογής.

Τα Clusters περιλαμβάνουν τόσο εντολές(commands) όσο και δεδομένα(attributes). Οι εντολές προκαλούν ενέργειες ενώ τα δεδομένα κρατάνε πληροφορίες και καταστάσεις για τον συγκεκριμένο cluster. Για παράδειγμα, μπορούμε να σκεφτούμε ένα ρυθμιζόμενο φωτιστικό που λειτουργεί στο Home Automation Profile. Το φωτιστικό μπορεί να υποστηρίζει τα OnOff cluster(0x0006) και LevelControl Cluster(0x0008). Σε αυτά υπάρχουν εντολές όπως η εντολή On, η Off και και η εντολή για το επίπεδο φωτισμού. Επιπλέον, υπάρχουν τα αντίστοιχα attributes που παρακολουθούν τις καταστάσει on,off και επίπεδο φωτισμού.

Τα Clusters έχουν νόημα μόνο μέσα σε συγκεκρριμμένο profile. Για παράδειγμα, σε ένα private profile το cluster 0x0000 μπορεί να ερμηνεύεται οπωσδήποτε, ενώ στο Home Automation profile το cluster 0x0000 αντιστοιχεί στο Basic Cluster.

Τα Clusters έχουν κατεύθυνση που σημαίνει ότι χωρίζονται σε εισόδους και εξόδους. Αυτός ο διαχωρισμός χρησιμοποιείται μόνο για την διαδικασία αναγνώρισης μιας συσκευής από τις υπόλοιπες συσκευές του δικτύου. Για παράδειγμα, ένας διακόπτης που υποστηρίζει τον Cluster OnOff 0x0006 ως έξοδο θα αναζητήσει ένα φωτιστικό με τον ίδιο Cluster ως είσοδο. [1],[6],[12]

3.8 Το επίπεδο ZigBee Device Object-ZDO

Το ZDO είναι μια εφαρμογή που βρίσκεται στο endpoint 0 και χρησιμοποιεί ένα ξεχωριστό profile που ονομάζεται ZigBee Device Profile-ZDP. Αυτή η ZDO εφαρμογή παρακολουθεί την κατάσταση της συσκευής στο δίκτυο και παρέχει μια διεπαφή στο ZigBee Device Profile το οποίο είναι υπεύθυνο για λειτουργίες όπως εύρεση, ρύθμιση και διαχείριση συσκευών και υπηρεσιών στο δίκτυο. [1]

Το ZDO αλληλεπιδρά άμεσα με το network επίπεδο ελέγχοντας την σύνδεση ή την αποσύνδεση της συσκευής στο δίκτυο. Ρυθμίζει επίσης λειτουργίες όπως ο αριθμός επανάληψης προσπάθειας για σύνδεση ή την συνεχόμενη προσπάθεια μέχρι να επιτευχθεί η σύνδεση.

Οι υπηρεσίες του ZDO με το ZDP παρέχονται και over-the-air και μπορούν να χωριστούν στις ακόλουθες κατηγορίες:

-Device discovery services
-Service discovery services
-Binding services
-Management services

Device - Service discovery services

Το ZigBee Device Profile-ZDP περιλαμβάνει εντολές για την διερεύνηση διάφορων πτυχών μιας συσκευής του δικτύου και των υπηρεσιών που προσφέρει. Μέσω αυτών μπορούμε να μάθουμε λεπτομέριες για κάθε συσκευή στο δίκτυο. Όλες αυτές οι εντολές είναι προαιρετικές από την πλευρά της συσκευής που θέτει το ερώτημα(Client) αλλά είναι υποχρεωτικές από τη πλευρά της συσκευής που δέχεται το ερώτημα(Server). Έτσι μια συσκευή μπορεί εάν θέλει να διερευνά άλλες συσκευές στο δίκτυο αλλά είναι υποχρεωμένη να απαντά σε ερωτήματα άλλων συσκευών.

NWK_addr_req: Χρησιμοποιείται όταν γνωρίζουμε τη διεύθυνση MAC μιας συσκευής και θέλουμε να μάθουμε τη 16-bit διεύθυνση. Το αίτημα μπορεί να είναι broadcast ή unicast.

IEEE_addr_req: Είναι η αντίθετη εντολή της προηγούμενης. Γνωρίζοντας τη 16-bit διεύθυνση μας επιστρέφει τη διεύθυνση MAC

ZigBee Descriptors: Το Zigbee χρησιμοποιεί τους Descriptors για να περιγράψει έναν κόμβο και τις λειτουργίες του επιτρέποντας στην εφαρμογή να ανακαλύψει αυτές τις λειτουργίες over-the-air. Αυτοί οι Descriptors περιλαμβάνουν τους node descriptor, power descriptor, complex descriptor, user descriptor, simple descriptor. Κάθε ένας από αυτούς περιέχει πεδία που περιγράφουν τον εκάστοτε κόμβο όπως τα ενεργά endpoints, τα application profiles, το device id και άλλα.

Device Discovery Services	Unicast (U), Broadcast (B) or Either (U,B)	Client Transmission (Request)	Server Processing (Response)
NWK_addr_req	U,B	O	M
IEEE_addr_req	U	O	M
Node_Desc_req	U	O	M
Power_Desc_req	U	O	M
Complex_Desc_req	U	O	O
User_Desc_req	U	O	O
User_Desc_set	U	O	O
Device_annce	B	O	M

Εικόνα 3.6 Device Discovery Services

Service Discovery Services	Client Transmission (Request)	Server Processing (Response)
Simple_Desc_req (unicast)	O	M
Extended_Simple_Desc_req (unicast)	O	O
Active_EP_req (unicast)	O	M
Extended_Active_EP_req (unicast)	O	O
Match_Desc_req (broadcast)	O	M
System_Server_Discover_req	O	O
Find_node_cache_req (broadcast)	O	O
Discovery_Cache_req (unicast)	O	O
Discovery_store_req (unicast)	O	O
Node_Desc_store_req (unicast)	O	O
Power_Desc_store_req (unicast)	O	O
Active_EP_store_req (unicast)	O	O
Simple_Desc_store_req (unicast)	O	O
Remove_node_cache_req (unicast)	O	O

Εικόνα 3.7 Service Discovery Services

Binding services

Η διαδικασία του binding είναι ένας μηχανισμός που επιτρέπει σε ένα endpoint ενός κόμβου να ενωθεί με ένα ή πολλά άλλα endpoints ενός άλλου κόμβου. Με αυτό τον τρόπο κάθε data request μπορεί να χρησιμοποιεί τον τοπικό binding table για την αποστολή δεδομένων στα αναφερόμενα σε αυτόν endpoints. Ο binding table δίνει την ευκολία επικοινωνίας διότι κρατάει ενημερωμένες τις διευθύνσεις 16-bit και 64-bit των καταχωρημένων κόμβων ανεξάρτητα αν κάποια από αυτές χρειαστεί να αλλάξει την διεύθυνση 16-bit.

Management services

Είναι διάφορες προαιρετικές υπηρεσίες για την ανάγνωση ορισμένων πινάκων και την αίτηση για ενέργειες που αφορούν το δίκτυο.

Mgmt_NWK_Disc_req: Μέσω αυτής της εντολής μπορεί μια εφαρμογή να ανιχνεύσει για γειτονικά δίκτυα. Όπως και στην περίπτωση σύνδεσης καινούργιου κόμβου σε δίκτυο, στέλνεται ένα beacon request και στην συνέχεια εξετάζονται τα beacon responses απο τα τυχόν γειτονικά δίκτυα. Στην συνέχεια μπορεί η εφαρμογή να αποφασίσει τις κατάλληλες ενέργειες με βάση τα αποτελέσματα. Για παράδειγμα, θα μπορούσε να αλλάξει το κανάλι λειτουργίας του δικτύου, αν και κάτι τέτοιο δε συνηθίζεται στο Zigbee.

Mgmt_Permit_ Joining _req: Είναι μια εντολή που στέλνεται είτε unicast έιτε broadcast στους ρούτερ του δικτύου έτσι ώστε να επιτρέψουν σε καινούργιες συσκευές να συνδεθούν στο δίκτυο για κάποιο ορισμένο χρονικό διάστημα.

3.9 ZigBee Cluster Library

Η βιβλιοθήκη Zigbee Cluster είναι ένα σύνολο συναρτήσεων που ορίζονται από συγκεκριμένο specification της Zigbee Alliance και βοηθάνε στην εύκολη και γρήγορη ανάπτυξη εφαρμογών Zigbee. Η βιβλιοθήκη χωρίζεται σε τομείς ανάλογα την εφαρμογή όπως, basic cluster, identify cluster,groups cluster, OnOff cluster, alarm cluster, IAS zone cluster και άλλα. [6]

Η χρήση της βιβλιοθήκης δεν είναι υποχρεωτική. Προσφέρει όμως πολλές δυνατότητες, η σημαντικότερη αυτών είναι η διαλειτουργικότητα. Χρησιμοποιείται κυρίως από τα public profiles αλλά μπορεί να χρησιμοποιηθεί και από τα private profiles. Η χρήση της βιβλιοθήκης είναι επιβεβλημένη όταν σχεδιάζουμε ένα προϊόν που θέλουμε να λειτουργεί και με συσκευές άλλων κατασκευαστών. Συνήθως αποφεύγουμε την χρήση της όταν υλοποιούμε μια απλή εφαρμογή που δεν χρειάζεται διαλειτουργικότητα και θέλουμε εξοικονόμηση χώρου διότι η ZCL καταλαμβάνει αρκετό χώρο στις συσκευές 8-bit MCU.

Οι clusters στην ZCL χρησιμοποιούν τις έννοιες του client που ξεκινάει την επικοινωνία και του server που εκτελεί τις εντολές. Για παράδειγμα ένας διακόπτης(Client) δημιουργεί το πακέτο On-Off προς το φωτιστικό(server) που εκτελεί την εντολή On-Off ή στέλνοντας και την κατάστασή του.

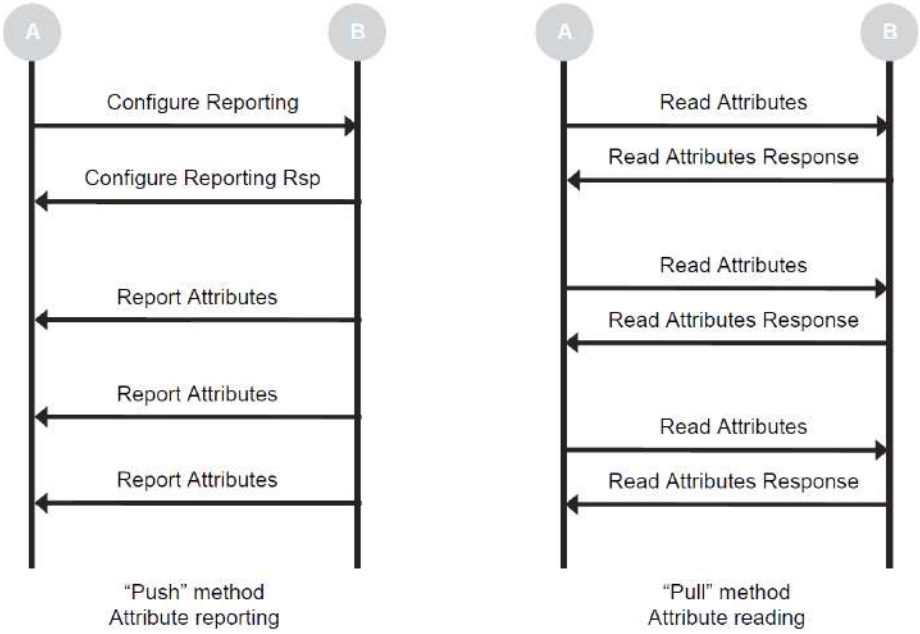
Αρχή λειτουργίας της ZCL

Όπως αναφέρθηκε, η ZCL περιέχει τις έννοιες των data και attributes που αντιστοιχούν στις εντολές που εκτελεί ο κάθε cluster και τα δεδομένα που καθορίζονται από τον cluster. Τα attributes μπορούν να διαβαστούν, να εγγραφούν δεδομένα σε αυτά ή να αναφερθούν σε μια επικοινωνία over-the-air. [1],[6]

Στην ZCL υπάρχουν οι εντολές cross-cluster οι οποίες είναι το θεμέλιο σε όλους τους clusters. Είναι εντολές που εκτελούν τις λειτουργίες όπως την ανάγνωση, την εγγραφή attributes ή την ρύθμιση μιας συσκευής για καθορισμένη αναφορά ενός ή πολλών attributes.

Μέθοδοι push-pull: Η ZCL επιτρέπει τις μεθόδους push-pull για τον καθορισμό της κατάστασης των attributes στα clusters. Στην μέθοδο push οι συσκευές στέλνουν από μόνες τους μια αναφορά με

τα attributes. Αυτή η αναφορά μπορεί να ρυθμιστεί να στέλνεται ανα κάποιο χρονικό διάστημα, με την αλλαγή της τιμής κάποιου attribute ή και τα δυο. Για παράδειγμα, ένας αισθητήρας μπορεί να στέλνει την τιμή του ανα μερικά δευτερόλεπτα ή όταν η τιμή του αλλάξει. Στα παρακάτω σχεδιαγράμματα φαίνεται η ρύθμιση των δύο μεθόδων.



Εικόνα 3.8 Οι μέθοδοι push-pull για την ανάγνωση των attributes

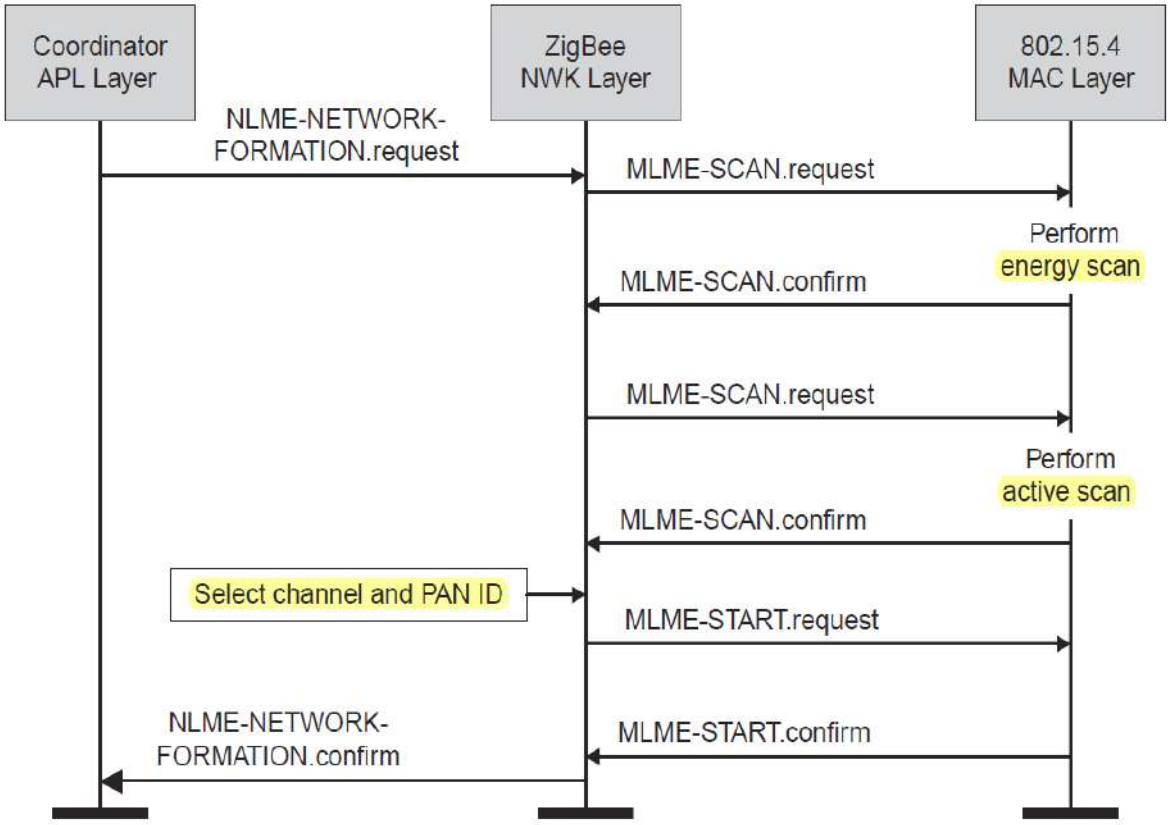
3.10 Επίπεδο Network

Το επίπεδο Network βασίζεται στο 802.15.4 και προσθέτει σε αυτό μερικές επιπρόσθετες λειτουργίες. Είναι υπεύθυνο για τη δημιουργία του δικτύου, τη σύνδεση-αποσύνδεση μιας συσκευής στο δίκτυο, την προσθήκη ασφάλειας στο δίκτυο και τη δρομολόγηση πακέτων(multi hop – mesh network).

Δημιουργία δικτύου

Πριν αρχίσει η επικοινωνία Zigbee θα πρέπει να δημιουργηθεί το δίκτυο και να συνδεθούν συσκευές σε αυτό. Τη δημιουργία του δικτύου την αναλαμβάνει ο coordinator. Η διαδικασία της δημιουργίας περιλαμβάνει δύο βασικά βήματα, την επιλογή του PAN ID και την επιλογή ενός καναλιού. Κατα τη διάρκεια της δημιουργίας του δικτύου στέλνεται ένα πακέτο από τον coordinator σε κάθε κανάλι(11-26). Αν δεν υπάρχει άλλο δίκτυο στην περιοχή τότε αυτό θα είναι το μοναδικό πακέτο που θα ανιχνευθεί από τον sniffer. Ο τρόπος επιλογής του καναλιού, του PAN ID και το πότε θα ξεκινήσει το δίκτυο καθορίζεται απο την εφαρμογή που βρίσκεται στον coordinator. Αυτή η εφαρμογή μπορεί να είναι οτιδήποτε, όπως ένα gateway στο internet, ένας απλός αισθητήρας ή ο κεντρικός πίνακας ενός συναγερμού. Το δίκτυο μπορεί να ξεκινήσει με την εφαρμογή τροφοδοσίας στον coordinator, ή μπορεί να περιμένει μέχρι να εκτελεστεί κάποιο συμβάν(πάτημα ενός κουμπιού). [1]

Η διαδικασία δημιουργίας ενός δικτύου Zigbee ακολουθεί την πορεία του παρακάτω σχεδιαγράμματος.



Εικόνα 3.9 Η διαδικασία δημιουργίας ενός δικτύου Zigbee

Η αίτηση δημιουργίας του δικτύου ξεκινάει από το επίπεδο εφαρμογής και συγκεκριμένα από το ZDO, με το NLME-NETWORK-FORMATION.request να στέλνεται και να επεξεργάζεται στο επίπεδο Network. Στην συνέχεια το Zigbee καλεί το επίπεδο MAC να πραγματοποιήσει δύο σαρώσεις, energy scan και active scan.Το energy scan καθορίζει το ποίο ήσυχο κανάλι απο το σύνολο καναλιών που επιλέχθηκαν στο APS. Κάθε σάρωση καναλιού διαρκεί περίπου μισό δευτερόλεπτο. Στη συνέχεια το active scan ανιχνεύει γειτονικά δίκτυα το οποίο είναι χρήσιμο ώστε να αποφευχθεί η σύγκρουση δύο ίδιων PAN IDs. Το active scan διαρκεί επίσης μερικά δευτερόλεπτα προκειμένου να φτάσουν τα beacon responses από πιθανά δίκτυα στην περιοχή. Έπειτα οι πληροφορίες που συλλέχθηκαν για τα κοντινά δίκτυα και τα κανάλια στέλνονται στο επίπεδο της εφαρμογής για να επιλέξει κατάλληλο κανάλι και PAN ID. Αυτή η επιλογή διαφέρει απο στοίβα σε στοίβα ανάλογα με την υλοποίηση της συνάρτησης επιλογής καναλιού-PAN ID του κάθε κατασκευαστή στοίβας. [1],[3]

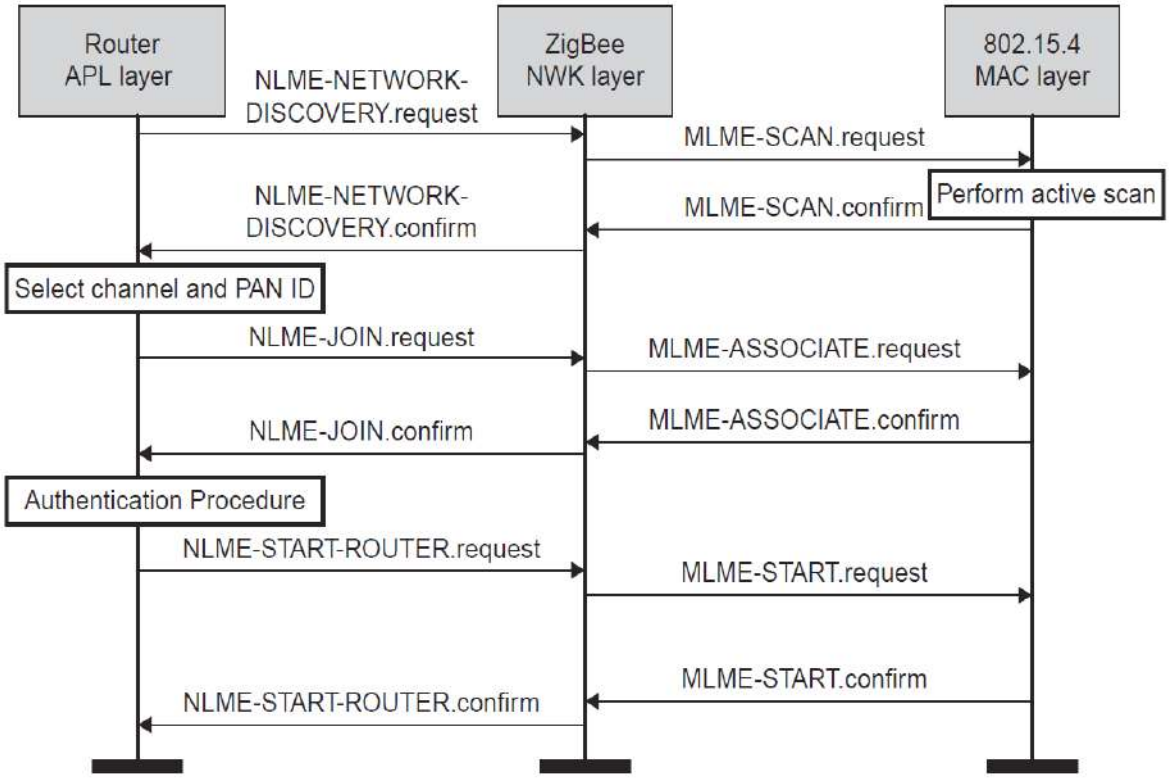
Σύνδεση κόμβου σε δίκτυο Zigbee

Σύνδεση σε δίκτυο πραγματοποιούν οι ρούτερ και οι end-devices. Η διαδικασία της σύνδεσης περιέχει τα βασικά βήματα της ανίχνευσης των δικτύων της περιοχής και την επιλογή ενός από αυτά. Η σύνδεση ξεκινάει απο την συσκευή που επιθυμεί να συνδεθεί στέλνοντας ένα πακέτο beacon request. Στην συνέχεια περιμένει για έναν καθορισμένο χρόνο ώστε να λάβει beacon responses. Τα beacon responses στέλνονται από όλους τους ρούτερ και τον coordinator και περιέχουν χρήσιμες πληροφορίες για το δίκτυο συμπεριλαμβανομένων το PAN ID, το Extended PAN ID, το join enabled και τον διαθέσιμο αριθμό για καινούργιες συσκευές. [1],[3]

Οι συσκευές συνδέονται σε έναν συγκεκριμένο κόμβο του δικτύου και όχι γενικά στο δίκτυο. Η συσκευή που θα συνδεθούν καλείται parent και η συνδεόμενη συσκευή child. Η σχέση parent-child για τις end-devices είναι ιδιαίτερη. Μολονότι όλες οι end-devices μπορούν να επικοινωνήσουν με οποιονδήποτε κόμβο του δικτύου, η επικοινωνία της end-device ξεκινάει πάντα με τον parent που στην συνέχεια θα δρομολογήσει το πακέτο στον τελικό προορισμό με την διαδικασία του mesh

networking. Εάν για κάποιο λόγο η επικοινωνία με τον parent χαθεί τότε η end-device θα αναζητήσει καινούργιο parent, θα ανακοινώσει ότι έχει μετακινηθεί και η επικοινωνία θα συνεχιστεί.[1],[3]

Η ολοκληρωμένη διαδικασία σύνδεσης της συσκευής στο δίκτυο φαίνεται στο παρακάτω σχεδιάγραμμα.



Εικόνα 3.10 Διαδικασία σύνδεσης της συσκευής στο δίκτυο

Αφού πραγματοποιηθούν οι σαρώσεις και επιλεγεί ο κατάλληλος parent η σύνδεση ολοκληρώνεται με τη διαδικασία του Authentication που πραγματοποιεί το trust center του δικτύου που θα αποφασίσει εάν ο κόμβος θα παραμείνει στο δίκτυο ώστε να του δοθεί το network key και να μπορεί να επικοινωνήσει με τους υπόλοιπους κόμβους.

Zigbee packet routing-Mesh routing

Το Zigbee προσθέτει μια πολύ σημαντική λειτουργία στο 802.15.4, αυτή είναι το mesh networking μέσω της οποίας αυξάνεται η αξιοπιστία παράδοσης ενός πακέτου στον προορισμό του. Η λειτουργία του βασίζεται στην εύρεση κατάλληλης διαδρομής(routing) και στη μετάδοση του πακέτου από κόμβο σε κόμβο μέχρι την αποστολή στον τελικό προορισμό. Σε αυτή την λειτουργία συμμετέχουν μόνο οι ρούτερ και ο coordinator, οι end-devices επικοινωνούν μόνο με τον parent.

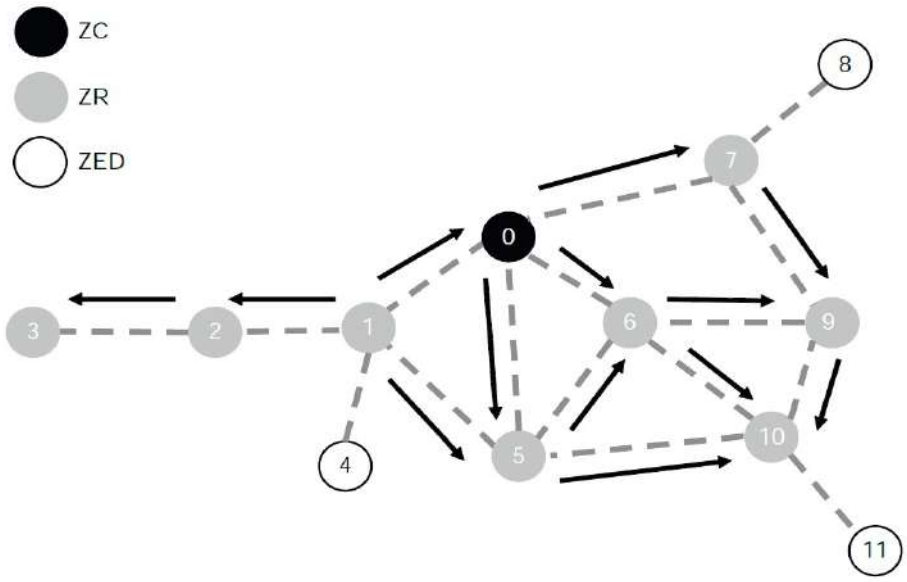
Τα μηνύματα broadcast αποτελούν θεμέλιο για την multi-hop επικοινωνία. Είναι μηνύματα που ξεκινούν από έναν κόμβο και διαδίδονται στο δίκτυο από κόμβο σε κόμβο μέχρι το η παράμετρος radius μηδενιστεί. Κάθε ρούτερ που λαμβάνει ένα broadcast μειώνει κατα ένα την μεταβλητή radius και αν η μεταβλητή έχει τιμή πάνω από ένα τότε αναμεταδίδει το μήνυμα ενώ προσθέτει και το μήνυμα στον πίνακα Broadcast Transaction Table (BTT). Ο πίνακας BTT παρακολουθεί τα μηνύματα broadcast έτσι ώστε να τα αναμεταδίδει και να τα περνάει στο APS μια φορά. Εάν ένας κόμβος λάβει ένα μήνυμα με radius ένα, τότε διαβιβάζει το μήνυμα στο επίπεδο APS και δεν το αναμεταδίδει στον αέρα. Οι εφαρμογές πρέπει να χρησιμοποιούν με προσοχή τα μηνύματα broadcast και πρέπει να

λάβουμε υπόψη ότι και κάποιοι μηχανισμοί του δικτύου χρησιμοποιούν broadcast όπως η λειτουργία route discovery.

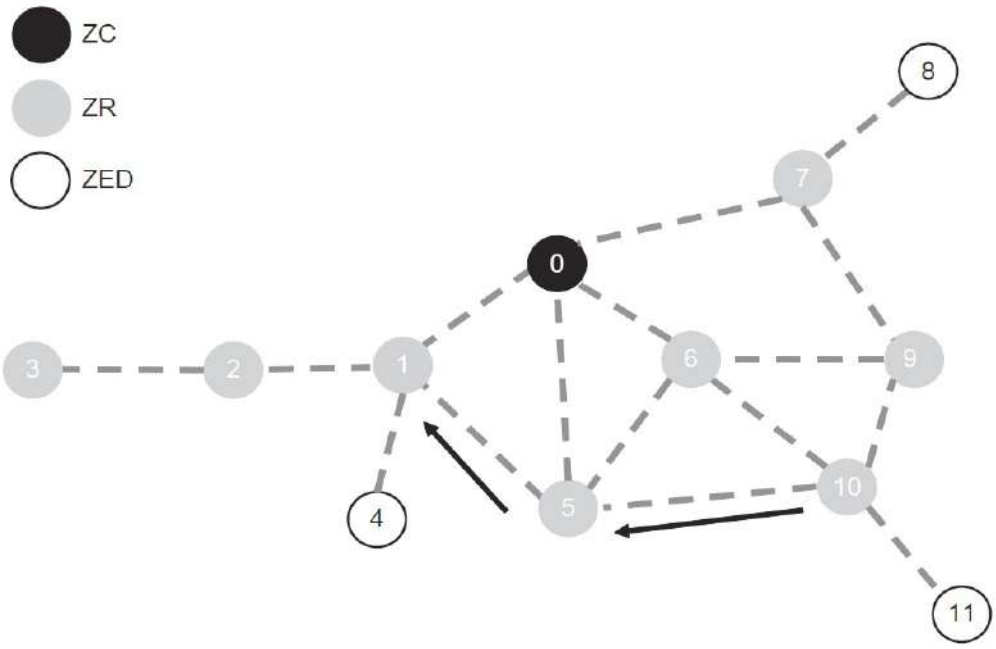
Κάθε κόμβος του δικτύου μπορεί να ξεκινήσει την διαδικασία του route dicovery και όλοι οι ρούτερ είναι προσωρινά υποψήφιο κομμάτι της διαδρομής. Το Zigbee κρατάει έναν πίνακα κατα την διάρκεια του route discovery προκειμένου στο τέλος να επιλεγεί η βέλτιστη διαδρομή. Η βέλτιστη διαδρομή δεν είναι πάντα αυτή με τις λιγότερες αναμεταδόσεις(hops) διότι σε μια διαδρομή μπορεί να υπάρχει και κάποια σύνδεση χαμηλής ποιότητας, λόγω κάποιου πιθανού εμποδίου.

Η διαδικασία route discovery ξεκινάει με ένα μήνυμα broadcast. Αρχικά ο κόμβος που επιθυμεί να στείλει ένα πακέτο δεν γνωρίζει που βρίσκεται ο προορισμός. Έτσι στέλνει ένα broadcast σε όλο το δίκτυο. Καθώς το μήνυμα μεταδίδεται στο δίκτυο κάθε κόμβος προσθέτει έναν αριθμό σε μια μεταβλητή που αντιπροσωπεύει την ποιότητα σύνδεσης per hop.

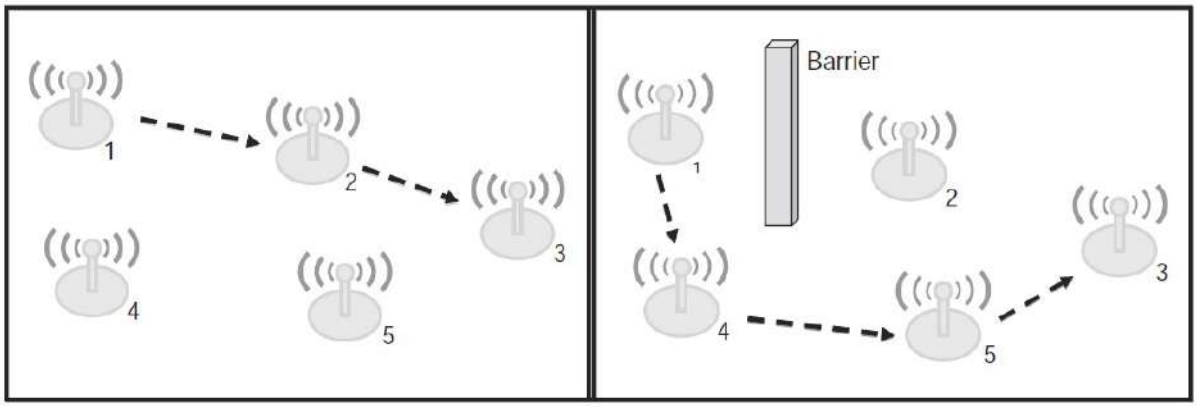
Ένα παράδειγμα της διαδικασίας route discovery φένεται στις παρακάτω εικόνες. Στο παράδειγμα ο κόμβος 1 επικοινωνεί για πρώτη φορά με τον κόμβο 10, έτσι ξεκινάει με την μετάδοση του broadcast. Αν υποθέσουμε πως όλες οι συνδέσεις των κόμβων είναι καλές τότε η βέλτιστη διαδρομή θα είναι 1-5-10. Ο κόμβος 10 περιμένει για κάποιο χρονικό διάστημα μήπως φτάσει κάποια διαδρομή με καλύτερη ποιότητα σύνδεσης. Μια καλύτερη διαδρομή μπορεί να φτάσει αργότερα λόγω της τυχαιότητας του αλγορίθμου του broadcast. Στην συνέχεια ο κόμβος 10 στέλνει μια απάντηση unicast στον κόμβο ένα και στο εξής η διαδρομή 1-5-10 θα χρησιμοποιείται για την επικοινωνία των δύο κόμβων. Εάν στο μέλλον αυτή η διαδρομή δεν λειτουργήσει το Zigbee θα χρησιμοποιήσει τη λειτουργία self-healing και θα ξανακάνει τη διαδικασία route discovery. [1]



Εικόνα 3.11 Μετάδοση broadcast για την εύρεση της βέλτιστης διαδρομής 1-10



Εικόνα 3.12 Μετάδοση unicast απο 10 σε 1 για την επαλήθευση της διαδρομής 1-5-10



Εικόνα 3.13 Η διαδικασία του self-healing

Κεφάλαιο 4ο: Κατασκευή συναγερμού Zigbee

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται η υλοποίηση ασύρματου συναγερμού βασισμένου στο πρότυπο Zigbee. Αρχικά τίθενται οι προδιαγραφές που επιθυμούμε από το τελικό σύστημα, έπειτα μελετούνται τα δομικά στοιχεία που συνθέτουν το σύστημα, ο τρόπος επιλογής των υλικών και των εργαλείων και η περιγραφή τους. Στην συνέχεια αναλύεται η πορεία της κατασκευής και τα σημαντικά σημεία του πηγαίου κώδικα που έχει γραφτεί στις ανάλογες συσκευές, μαζί με τα πακέτα Zigbee που ανταλλάσσονται στον αέρα.

4.2 Προδιαγραφές συστήματος

Στόχος της τελικής υλοποίησης είναι ένας εύχρηστος, οικονομικός, αξιόπιστος και εύκολα επεκτάσιμος ασύρματος συναγερμός.

Ιδιαίτερα σημαντικό για ένα προϊόν είναι ο τελικός αποδέκτης να μπορεί να το χειριστεί με ευκολία, ανεξάρτητα τις γνώσεις που διαθέτει, ειδικά όταν το προϊόν αυτό απευθύνεται σε κοινό που δεν προέρχεται από τον χώρο των ηλεκτρονικών όπως είναι οι κάτοχοι ενός απλού οικιακού συναγερμού. Το σύστημα μας λοιπόν θα πρέπει να έχει τις ελάχιστες δυνατές αρχικές ρυθμίσεις, απλή συντήρηση αλλά και απλές λειτουργίες.

Οι αρχικές ρυθμίσεις και η συντήρηση αφορούν συνήθως τη σύνδεση των ασύρματων αισθητήρων με τον κεντρικό πίνακα, τη σύνδεση του κεντρικού πίνακα με το διαδίκτυο και την αλλαγή μπαταριών στους αισθητήρες. Ενώ αναφερόμενοι στις λειτουργίες ενός συναγερμού εννοούμε συνήθως την διεπαφή που προσφέρει για την εύκολη καθημερινή του χρήση όπως είναι ο οπλισμός και η αφόπλιση του συναγερμού είτε άμεσα είτε απομακρυσμένα αλλά και η ενημέρωση της κατάστασης του συναγερμού και των αισθητήρων-ζωνών τόσο σε κατάσταση οπλισμού όσο και στην κατάσταση αφόπλισης.

Ο τομέας των αυτοματισμών και των οικιακών συναγερμών έχει εξελιχθεί την τελευταία εικοσαετία. Εμπλέκονται από μεγάλες εταιρίες έως και μικρότερους εμπόρους με αποτέλεσμα τη δημιουργία υψηλού ανταγωνισμού. Αυτό θέτει την ανάγκη για την επιλογή φθηνών εργαλείων και υλικών σε κάθε δομική μονάδα του συστήματος μας.

Σε ένα σύστημα συναγερμού πρώτο ρόλο έχει η αξιοπιστία. Η χρήση φθηνών υλικών δε μας περιορίζει σε αυτό. Αρκεί να επιλέξουμε έναν συνδυασμό υλικών και λογισμικού από αναγνωρισμένες και πιστοποιημένες εταιρίες στον χώρο του Zigbee.

Σχεδόν ποτέ ένας συναγερμός, ιδιαίτερα οικιακός, δεν είναι στατικός καθόλη τη διάρκεια αξιοποίησης του. Πολλές φορές, κυρίως για λόγους οικονομίας, ο κάτοχος μπορεί να θελήσει μόνο την απολύτως απαραίτητη κάλυψη του χώρου όπου τοποθετείται αρχικά ο συναγερμός αφήνοντας περιθώρια για μελλοντική επέκταση όπως η περιμετρική κάλυψη ενός κήπου. Επιπλέον, πιθανό σενάριο είναι η μετακίνηση του συναγερμού σε άλλο χώρο με διαφορετικές ανάγκες. Σε αυτές τις περιπτώσεις ο κάτοχος του συναγερμού θα πρέπει να έχει την δυνατότητα για προσθήκη επιπλέον αισθητήρων ή και την κατάργηση κάποιον εξ αυτών ανα πάσα στιγμή.

4.3 Επιλογή υλικών

Στην επίτευξη των επιθυμητών προδιαγραφών του συστήματος τον σημαντικότερο ρόλο έχει η σωστή επιλογή των εργαλείων και των υλικών που θα χρησιμοποιηθούν. Επιπλέον, εάν τα υλικά επιλεχθούν κατάλληλα τότε ο σχεδιασμός του συστήματος μπορεί να απλοποιηθεί σε μεγάλο βαθμό μειώνοντας το φόρτο εργασίας και ελαττώνοντας τα έξοδα αγοράς. Προκειμένου όμως το τελικό προϊόν να μην υστερεί σε κανέναν παράγοντα απο αυτούς που έχουν οριστεί στην προηγούμενη ενότητα θα πρέπει να λάβουμε υπόψη τις παρακάτω ερωτήσεις:

Είναι τα υλικά που επιλέξαμε πιστοποιημένα απο την Zigbee ALLIANCE;

Η Zigbee ALLIANCE πιστοποιεί έναν συνδυασμό υλικών τον οποίο κρίνει ως κατάλληλο συνδυασμό για να αλληλεπιδρά με άλλες συσκευές zigbee και τον ονομάζει ως πλατφόρμα(platform). Αυτός ο συνδυασμός αποτελείται από τον πομποδέκτη zigbee, τον MCU και το λογισμικό της στοίβας zigbee. Εάν επιλεχθεί μια πιστοποιημένη πλατφόρμα είναι σίγουρη η ομαλή λειτουργία των υλικών και σε περίπτωση που χρειαστεί, εξασφαλίζεται η επιτυχημένη αλληλεπίδραση του συστήματος μας με συσκευές zigbee άλλων εταιριών. Ενδέχεται μια πλατφόρμα που δεν έχει πιστοποιηθεί να μην λειτουργεί με συσκευές που χρησιμοποιούν διαφορετική στοίβα zigbee.

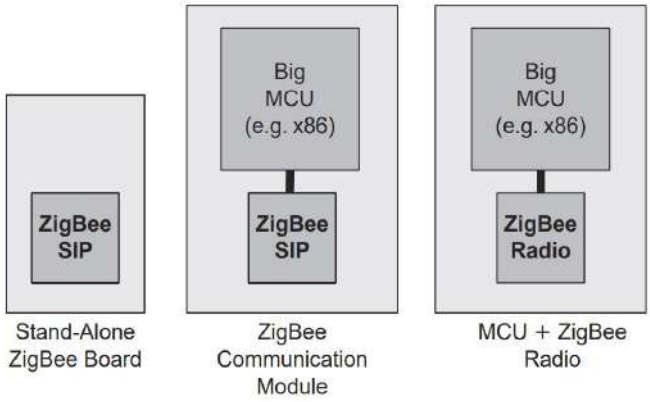
Θα χρησιμοποιηθεί συνδιασμός ολοκληρωμένου MCU/radio ή ξεχωριστού πομποδέκτη zigbee;

Αυτό είναι το σημαντικότερο χαρακτηριστικό που θα κρίνει την δομή των συσκευών του συναγερμού. Από ποια δομικά στοιχεία δηλαδή θα αποτελείται ο κάθε κόμβος στο δίκτυο zigbee και ποια θα είναι η λειτουργία του κάθε στοιχείου.

Στην περίπτωση του ολοκληρωμένου MCU/radio η στοίβα zigbee και η εφαρμογή βρίσκονται στο ίδιο τσιπάκι. Συνήθως βασίζεται σε έναν μικροεπεξεργαστή 8-bit τύπου 8051. Το πλεονέκτημα είναι ότι δεν χρειαζόμαστε επιπλέον υλικά για την εφαρμογή μας.

Στην αντίθετη περίπτωση, του ξεχωριστού πομποδέκτη, έχουμε τη δυνατότητα να επιλέξουμε τον CPU της αρεσκείας μας και να τοποθετήσουμε εκεί την εφαρμογή και τη στοίβα zigbee στέλνοντας έτοιμα πακέτα στον πομποδέκτη 802.15.4. Με αυτόν τον συνδυασμό μπορούμε να επιλέξουμε τον CPU με τον οποίο είμαστε εξοικειωμένοι και να έχουμε μεγαλύτερη άνεση αποθηκευτικού χώρου για την εφαρμογή. Θα πρέπει όμως να ξέρουμε εάν η στοίβα zigbee με την οποία δουλεύουμε λειτουργεί σωστά στον συνδυασμό CPU/radio που επιλέξαμε.

Ένας άλλος τρόπος για να αναπτύξουμε την εφαρμογή μας στον MCU που επιθυμούμε είναι να χρησιμοποιήσουμε ενα ολοκληρωμένο zigbee module, δηλαδή συνδιασμός ενός 802.15.4 radio και ενός μικροεπεξεργαστή 8051. Με αυτόν τον τρόπο όλα τα χαρακτηριστικά του zigbee, στοίβα και μετάδοση-λήψη πακέτων, βρίσκονται στο module αλλά σε αντίθεση με την περίπτωση ολοκληρωμένου MCU/radio η εφαρμογή βρίσκεται στον MCU της επιλογής μας. Η επικοινωνία αυτών των δύο γίνεται μέσω σειριακής θύρας.



Εικόνα 4.1 Zigbee Board Configurations

Τι εργαλεία υπάρχουν για να υποστηρίξουν την κατασκευή;

Τα κατάλληλα εργαλεία μπορούν να διευκολύνουν σε μεγάλο βαθμό τη διαδικασία κατασκευής του τελικού προϊόντος. Εκτός από τα υλικά στο τελικό προϊόν θα πρέπει να γνωρίζουμε τι εργαλεία είναι διαθέσιμα, εάν είμαστε εξοικειωμένοι με αυτά, αν υπάρχουν σε διαθεσιμότητα, αν υπάρχει η κατάλληλη υποστήριξη και φυσικά να τα υπολογίσουμε στο τελικό κόστος κατασκευής σε περίπτωση που χρειαστεί να αγοράσουμε κάποιο από αυτά.

Υπάρχουν πρότυπα εργασιών;

Μπορούμε να γλιτώσουμε πολύ χρόνο και κόπο όταν η εργασία μας βασίζεται σε εργασίες που μας παρέχουν άλλοι. Είναι σημαντικό λοιπόν να ξέρουμε εάν τα υλικά που έχουμε επιλέξει τα υποστηρίζει κατάλληλα ο πάροχος δίνοντας σαφή και λεπτομερή περιγραφή αλλά και μερικά παραδείγματα εφαρμογών. Επιπλέον, μπορούμε να αναρωτηθούμε εάν υπάρχει κάποιο επίσημο φόρουμ στο οποίο θα υπάρχουν απαντήσεις σε συχνά ερωτήματα σχετικά με τα υλικά μας, ιδιαίτερα εάν υπάρχουν πολλά άτομα τα οποία ασχολήθηκαν με τα συγκεκριμένα υλικά.

4.4 Λίστα υλικών

Με βάση τα παραπάνω ακολουθεί η λίστα με τα εργαλεία και υλικά που χρησιμοποιήθηκαν για τον σχεδιασμό και την κατασκευή του συναγερμού:

Υλικά	Ενδεικτική τιμή
Core2530B CC2530Module της waveshare	15€
PIC12F1840	2€
ESP32 DevBoard	8€
LCD16*2	7€
Keypad matrix	4€
Μαγνητικές επαφές	1€

Πίνακας με τα υλικά

Εργαλεία	Ενδεικτική τιμή
CC2531 Sniffer	5€
CC Debugger	15€
Zstack	-
Pickit 3	15€
PL2303 USB to TTL converter	1€
IAR Embedded Workbench IDE	-
ZTool	-
SmartRF Studio 7	-
Visual Studio Code	-
Platform IO IDE	-
MPLAB X IDE v5.45	-
zboss_sniffer	-
Wireshark	-
XCTU	-
MIT App Inventor	-

Πίνακας με τα φυσικά και τα software εργαλεία

Core2530 B CC2530Module

Το Core2530 B είναι της waveshare και ο σχεδιασμός του βασίζεται στο τσιπάκι CC2530F256 της Texas Instruments. Έχει σχεδιαστεί με τέτοιο τρόπο ώστε τον interface να είναι συμβατό με το γνωστό module Xbee της Digi. Επιπλέον, παρέχεται δωρεάν firmware που εγκαθίσταται στο Core2530 B έτσι ώστε ο χρήστης να μπορεί να αλληλεπιδρά με αυτό σαν να ήταν ένα απλό UART module αποφεύγοντας έτσι τις πολλές δύσκολες λεπτομέρειες του πρωτοκόλλου zigbee, όπως συμβαίνει και στο Xbee. Μια επιπλέον δυνατότητα όμως που δίνεται είναι να απομονωθεί το τσιπάκι CC2530 με έναν CC Debbuger. Με αυτόν τον τρόπο μπορούμε να χρησιμοποιήσουμε την στοίβα zigbee της αρεσκείας μας χωρίς να δεσμευόμαστε με το προεγκατεστημένο firmware και software που δίνεται με το Core2530B. [10],[9],[]

Ένα από τα θετικά για τον CC2530 που τον καθιστά ιδανικό για κάποιον που κάνει τα πρώτα βήματα στον χώρο του zigbee είναι η υποστήριξη και η πλούσια βιβλιογραφία που παρέχεται από την TI και του επίσημου φόρουμ.

Core2530 (B) Specifications:

•Onboard chip: CC2530F256RHAR
•Communication distance (open and wide operating environment): up to 130 meters when using PCB antenna
•Frequency range: 2.4GHz
•Wide supply-voltage range (2 V–3.6 V)
•Operating temperature: -40℃ ～85℃
•Serial port baud rate: 38400bps (default), different baud rates are available by software configuration
•Dimension: 26mm x 28mm (PCB)

CC2530F256RHAR Features:
•16 transfer chains, software configurable according to environmental conditions
•Radio baud rate up to 250kbps
•High-Performance and Low-Power 8051 microcontroller core with code prefetch
•2.4-GHz IEEE 802.15.4 compliant RF transceiver
•Watchdog timer, battery monitor, and temperature sensor
•12-Bit ADC with 8 channels and configurable resolution
•2 powerful USARTs with support for several serial protocols
•IR generation circuitry
•General-Purpose timers (One 16-Bit, Two 8-Bit)
•AES security coprocessor
•21 General-Purpose I/O pins (19 × 4 mA, 2 × 20 mA)

ZStack

Πρόκειται για τη στοίβα zigbee της TI που παρέχεται δωρεάν. Αντί λοιπόν να κρατήσουμε το προεπιλεγμένο firmware της waveshare στο module Core2530B έχουμε επιλέξει να το αντικαταστήσουμε με ένα της TI. Αυτό μας δίνει την δυνατότητα να το εξερευνήσουμε περισσότερο και να κατανοήσουμε περισσότερα πράγματα για το zigbee μέσω της πλούσιας βιβλιογραφίας της TI. Η στοίβα Zstack είναι ανοιχτού κώδικα γραμμένη σε γλώσσα C. Είναι άρτια δομημένη σε επίπεδα όπου στο κάθε επίπεδο προσφέρεται το ανάλογο API. Ο προγραμματιστής λοιπόν μπορεί να απομονώσει το επίπεδο το οποίο θέλει να μελετήσει και να κατανοήσει τον τρόπο με τον οποίο υλοποιείται το zigbee. Η Zstack ακολουθεί πιστά το πρότυπο που έχει ορίσει η Zigbee Aliance και σε συνδυασμό με το CC2530 αποτελεί μια πιστοποιημένη πλατφόρμα zigbee. [12],[16]

PIC12F1840

Η χρήση ενός επιπλέον μικροελεγκτή είναι προεραιτική, η παρουσία του όμως απλοποιεί την κατασκευή και δίνει αρκετή ελευθερία στον σχεδιαστή. Πρόκειται για έναν μικροελεγκτή που τον χρειαζόμαστε αποκλειστικά για την εφαρμογή του συναγερμού. Έχουμε επιλέξει δηλαδή να κρατήσουμε όλα τα χαρακτηριστικά του zigbee στο module Core2530 B και με έναν μικροελεκτή PIC να ρυθμίζουμε τη λειτουργία του module μέσω σειριακής επικοινωνίας. Σε αντίθετη περίπτωση, που κρατούσαμε και την εφαρμογή του συναγερμού στο module, θα χρειαζότανε να κατανοήσουμε πλήρως την στοίβα Zstack και το πώς αλληλεπιδρά το application layer με τα κατώτερα επίπεδα της στοίβας. Επιπλέον, θα χρειαζότανε να κάνουμε HAL porting, δηλαδή μικρές αλλαγές σε ένα απο τα χαμηλά επίπεδα της στοίβας που αφορά τους drivers για τα περιφερειακά(LCD,LEDS, Keypad κτλ) διότι η Zstack είναι γραμμένη για τις αναπτυξιακές πλακέτες της TI που περιλαμβάνουν το CC2530.

Ο συγκεκριμένος μικροελεκτής επιλέχθηκε με βάση ορισμένες απαιτήσεις. Εφόσον θα συνδυαστεί με τον CC2530 και θα τοποθετηθεί σε φορητή συσκευή θέλουμε να είναι μιρκός σε μέγεθος, να έχει μικρή κατανάλωση, να δουλεύει στα ίδια επίπεδα τάσης με τον CC2530 (3,3V) για να μη χρειαστούν μετατροπές, να απαιτεί τα ελάχιστα δυνατά εξωτερικά εξαρτήματα, να έχει τουλάχιστον μια σειριακή θύρα για την επικοινωνία με το module, να δουλεύει σε μεγάλη συχνότητα χωρίς εξωτερικό ταλαντωτή ώστε να πετύχουμε υψηλή ταχύτητα μετάδοσης-λήψης μεταξύ PIC-Core2530 χωρίς τον όγκο εξωτερικού ταλαντωτή και φυσικά να υπάρχει διαθεσιμότητα.

4.5 Πορεία κατασκευής

4.5.1 Σχεδιάγραμμα συστήματος

Το σύστημα συναγερμού θα αποτελείται από τον κεντρικό πίνακα ελέγχου, μερικούς ασύρματους αισθητήρες και τουλάχιστον μια συσκευή αναμετάδοσης(router) για μεγαλύτερη εμβέλεια στο δίκτυο zigbee.

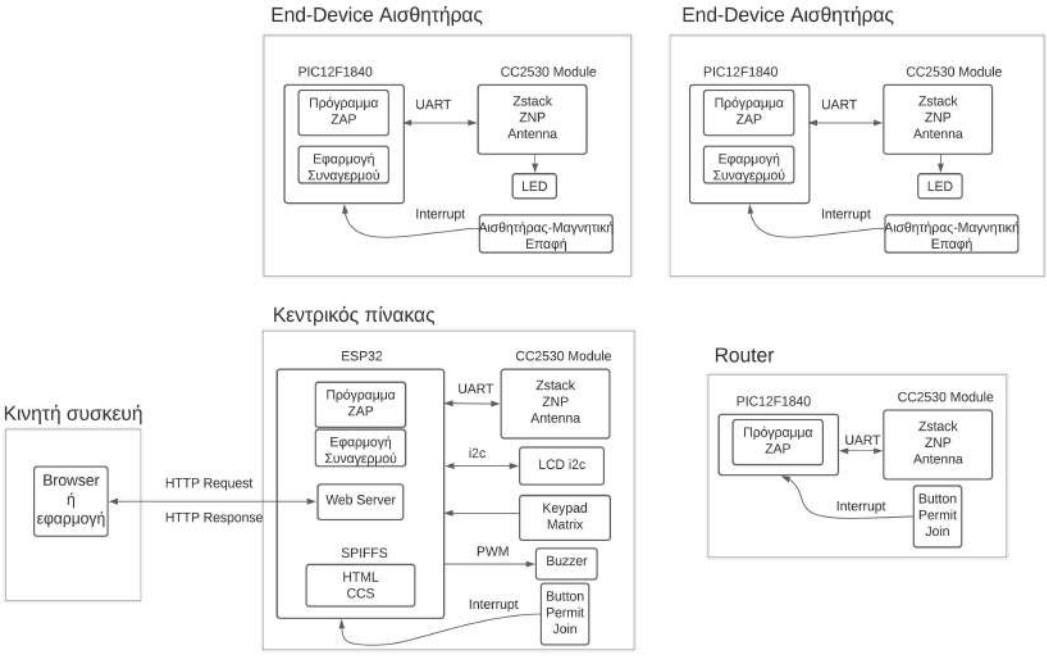
Εφόσον έχουμε επιλέξει ξεχωριστό μικροελεγκτή για την ανάπτυξη της εφαρμογής του συναγερμού και διαφορετικό module για τα χαρακτηριστικά zigbee, κάθε κόμβος του δικτύου θα έχει δύο βασικά δομικά υλικά. Στην περίπτωση των αισθητήρων και του router αυτά θα είναι ο μικροελεκτής PIC12F1840 και το module Core2530B. Από την άλλη πλευρά, ο κεντρικός πίνακας του συναγερμού θα είναι επίσης ένας κόμβος του δικτύου zigbee και θα αποτελείται από το module Core2530 και το ESP32Dev αντί του PIC12F1840.

Κάθε αισθητήρας θα λειτουργεί σαν end device στο δίκτυο. Ο ρόλος τους είναι να εξοικονομούν όσο το δυνατόν περισσότερη ενέργεια και να στέλνουν την κατάσταση του αισθητήρα στον κεντρικό πίνακα όποτε αυτή αλλάζει.

Ο router μπορεί να τοποθετηθεί σε σταθερή πηγή ρεύματος και σε σημείο τέτοιο ώστε να εξασφαλίζει την επιτυχημένη αναμετάδοση των πακέτων zigbee από τους αισθητήρες στον κεντρικό πίνακα.

Ο κεντρικός πίνακας έχει διάφορες λειτουργίες. Πρώτα απ όλα θα πρέπει να ρυθμιστεί έτσι ώστε να δουλεύει σαν coordinator του δικτύου. Εφόσον ρυθμιστεί θα πρέπει να δέχεται και να σχηματίζει δίκτυο με τους αισθητήρες και τον router. Αυτή η σύνδεση θα πρέπει να είναι ελεγχόμενη με κάποιο τρόπο ώστε να αποφεύγονται συνδέσεις με κακόβουλες συσκευές στο δίκτυο. Έπειτα θα τρέχει κατάλληλο πρόγραμμα ώστε να αναγνωρίζει τα πακέτα που στέλνονται απο τους αισθητήρες και να κάνει τις κατάλληλες ενέργειες ώστε να αναγνωρίζει και να σημάνει συναγερμό. Επιπλέον, θα πρέπει να παρέχει ένα εύκολο σύστημα διεπαφής με τον χρήστη τόσο κατά την διάρκεια της εγκατάστασης του συναγερμού στο σπίτι όσο και στις καθημερινές του λειτουργίες όπως είναι ο οπλισμός, η αφόπλιση και η ενημέρωση μέσω συναγερμού. Τέλος, ο κεντρικός πίνακας με τη βοήθεια του ESP32 θα λειτουργεί σαν gateway του δικτύου zigbee προς το internet ώστε ο χρήστης να έχει πρόσβαση και ενημέρωση για την κατάσταση του συναγερμού όσο θα βρίσκεται μακριά από το σπίτι.

Με βάση τα παραπάνω μπορούμε να σχεδιάσουμε μια πρώτη άποψη του συστήματος:

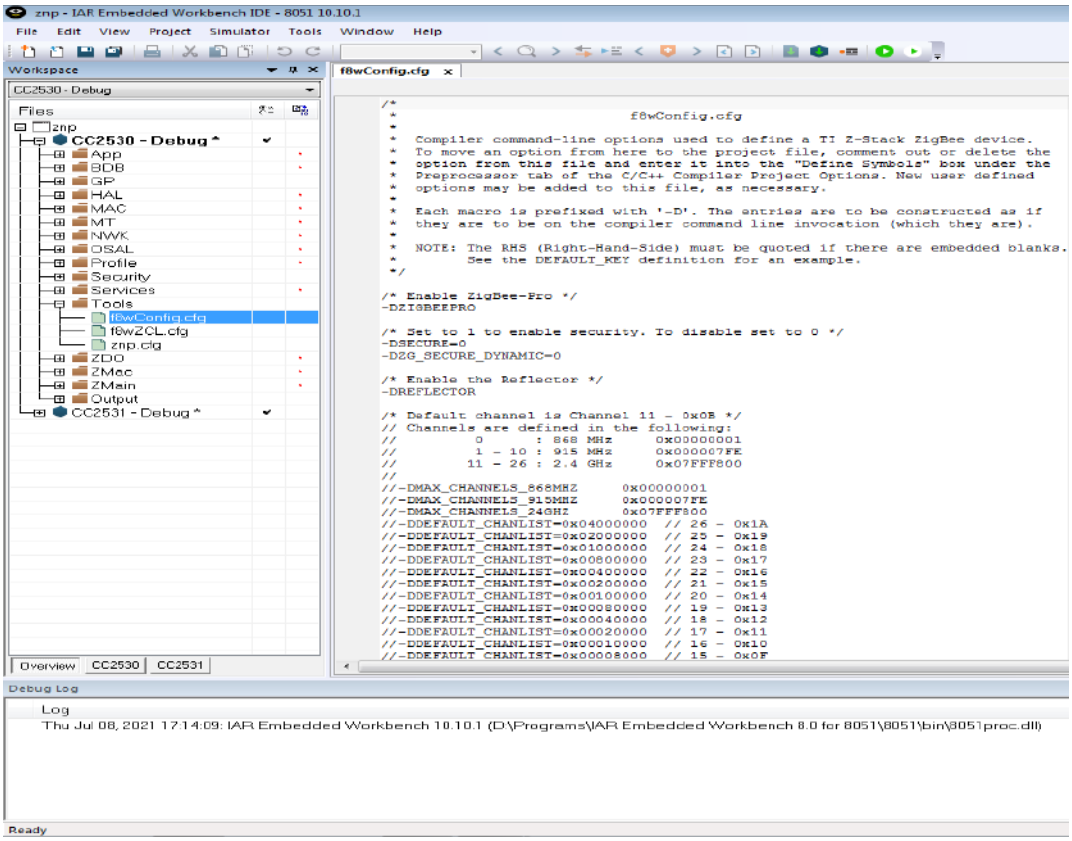


Εικόνα 4.2 Σχεδιάγραμμα συστήματος

4.5.2 Αντικατάσταση firmaware στο module Core2530B

Η TI προσφέρει έτοιμα firmware με τη στοίβα zigbee σε μορφή hex αλλά και σε πηγαίο κώδικα για το CC2530. Ένα από αυτά είναι σχεδιασμένο έτσι ώστε το CC2530 να λειτουργεί σαν ZNP(Zigbee Network Proccesor). Εκτός από τη στοίβα δηλαδή παρέχει και μια σειριακή διεπαφή ώστε να επικοινωνεί με έναν εξωτερικό μικροελεγκτή, στην περίπτωση μας τον PIC12F1840.[7],[15]

Προτού φορτώσουμε το συγκεκριμένο λογισμικό στο module μπορούμε να επεξεργαστούμε τον πηγαίο κώδικα σύμφωνα με τις ανάγκες μας και να παράγουμε το επιθυμητό firmware. Γι αυτό τον σκοπό θα επεξεργαστούμε το project ZNP της TI με το εργαλείο IAR Embedded Workbench IDE.[24]



Εικόνα 4.3 **IAR Embedded Workbench IDE με το project ZNP**

Ανοίγοντας το project znp έχουμε πρόσβαση σε όλους τους φακέλους των επιπέδων της στοίβας. Ένα από τα σημαντικότερα αρχεία που μπορούμε να επεξεργαστούμε είναι το f8wConfig.cfg στον φάκελο tools. Πρόκειται για ένα αρχείο που περιλαμβάνει compile options που ρυθμίζουν διάφορες παραμέτρους της στοίβας.

Κάποιες από τις σημαντικότερες είναι εάν το δίκτυο που θα δημιουργηθεί από τον coordinator θα χρησιμοποιεί κρυπτογραφημένα μηνύματα, ποιο θα είναι το κλειδί του δικτύου, εάν ο coordinator θα σκανάρει όλα τα διαθέσιμα κανάλια ή θα περιορίζεται σε μερικά από αυτά, αν θα χρησιμοποιείται συγκεκριμένο PAN ID και κάποιο extend PAN ID. Ενώ υπάρχουν και μερικές ακόμη παράμετροι όπως ο χρόνος poll rate που οι end-devices θα ελέγχουν για πακέτα σε αναμονή από τον parent, οι προσπάθειες που θα γίνονται από την end device πριν τη σήμανση της απώλειας σύνδεσης με τον parent, το μέγεθος του broadcast table και άλλα. [12],[16]

Στη στοίβα από προεπιλογή χρησιμοποιείται flow control για την επικοινωνία uart. Για να την απενεργοποιήσουμε και να εξοικονομήσουμε ελεύθερους ακροδέκτες στο module και στον PIC12F1840 θα πρέπει να επεξεργαστούμε τη συνάρτηση npInit() στο αρχείο znp_app.c. [27]

```
static void npInit(void)
{
  if (ZNP_CFG1_UART == znpCfg1)
  {
    halUARTCfg_t uartConfig;

    uartConfig.configured           = TRUE;
    uartConfig.baudRate             = ZNP_UART_BAUD;
#ifdef ZNP_ALT
    uartConfig.flowControl          = FALSE;
#else
    uartConfig.flowControl          = FALSE;
#endif
    uartConfig.flowControlThreshold = HAL_UART_FLOW_THRESHOLD;
    uartConfig.rx.maxBufSize        = HAL_UART_RX_BUF_SIZE;
    uartConfig.tx.maxBufSize        = HAL_UART_TX_BUF_SIZE;
    uartConfig.idleTimeout          = HAL_UART_IDLE_TIMEOUT;
    uartConfig.intEnable            = TRUE;
    uartConfig.callBackFunc         = npUartCback;
    HalUARTOpen(HAL_UART_PORT, &uartConfig);
    MT_UartRegisterTaskID(znpTaskId);

#ifdef HAL_PA_LNA_CC2592
   ZMacSetTransmitPower(TX_PWR_PLUS_19);
#else
   ZMacSetTransmitPower(TX_PWR_PLUS_4);
#endif
  }
  else
  {
    /* npSpiInit() is called by hal_spi.c: HalSpiInit().*/
  }
```

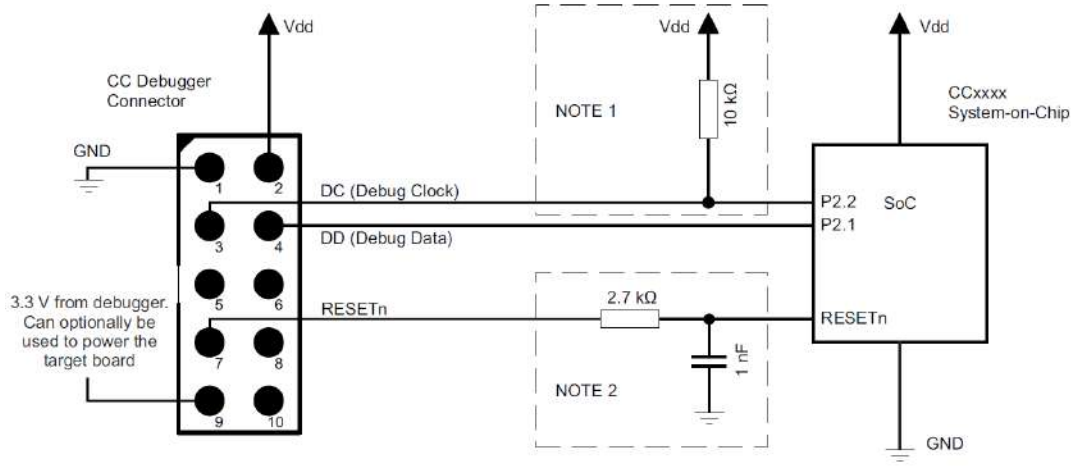
Εικόνα 4.4 Απενεργοποίηση flow control απο την συναρτηση npInit()

Αφού ολοκληρώσουμε τις επιθυμητές τροποποιήσεις κάνουμε compile το project και αποθηκεύουμε το παραγόμενο hex αρχείο.

4.5.2.1 Σύνδεση CC Debbugger με το Core2530B

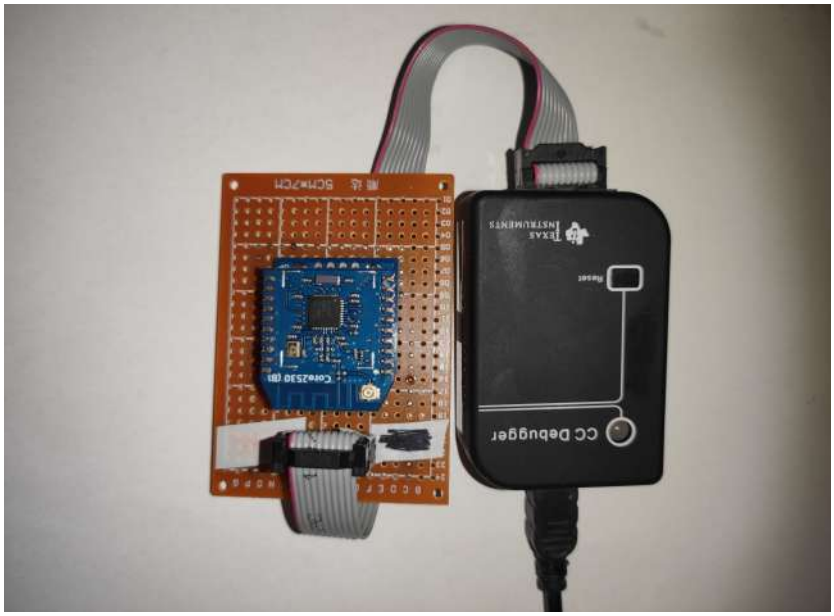
Ο CC Debugger χρησιμοποιείται για τον προγραμματισμό και το debbuging λογισμικού που τρέχει στις Ccxxxx SoC συσκευές τις TI. Το λογισμικό που διατίθεται στον υπολογιστή για να συνεργαστεί με τον CC Debugger είναι το SmartRF Flash Progremmer. Μέσω αυτού μπορούμε να ελέγξουμε απευθείας τις SoC συσκευές και να τις προγραμματίσουμε.[]

Αφού εγκαταστήσουμε τους απαραίτητους drivers θα πρέπει να κάνουμε τη σύνδεση CC debugger με το CC2530 προτού το συνδέσουμε στον υπολογιστή.



Εικόνα 4.5 Σύνδεση CC Debugger με 8051 SoC

Οι ελάχιστες συνδέσεις που πρέπει να γίνουν είναι τα δύο σήματα, Debug Clock, Debug Data, το σήμα reset και να επιλέξουμε τροφοδοσία για το CC2530, είτε από τον CC debbuger είτε από εξωτερική τροφοδοσία. Ο ακροδέκτης του reset είναι επιρρεπής σε θόρυβο, σε περίπτωση που έχουμε reset σε τυχαίες στιγμές θα πρέπει να προσθέσουμε το κατάλληλο RC φίλτρο. Η pull up αντίσταση 10K δε χρειάζεται διότι υπάρχει εσωτερική pull up αντίσταση στον ακροδέκτη P2.2 του CC2530.

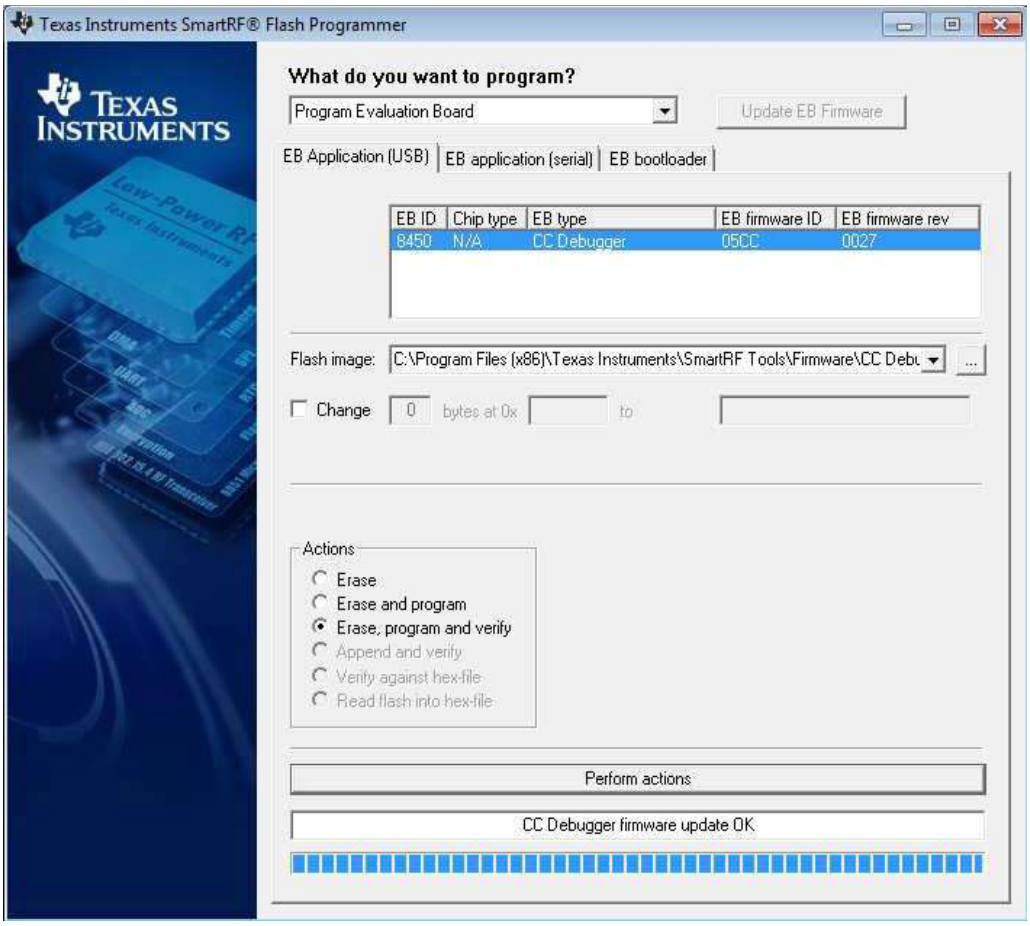


Εικόνα 4.6 Σύνδεση CC Debugger με Core2530B

4.5.2.2 Upload του firmware στον CC2530

Τέλος, έχοντας συνδέσει τον CC2530 στον CC debbugger συνδέουμε τον debbuger στον υπολογιστή.

Για την εγκατάσταση χρησιμοποιούμε το πρόγραμμα SmartRF Flash Proggrammer. Επιλέγουμε System-on-Chip, στη λίστα εμφανίζεται η συσκευή που έχουμε συνδέσει, στο πεδίο Flash image επιλέγουμε το αρχείο hex που δημιουργήσαμε προηγουμένως, στο πεδίο Actions επιλέγουμε Erase, program and verify και τέλος πατάμε Perform actions για να ολοκληρώσουμε τη διαδικασία αλλαγής του προεγκατεστημένου firmware.



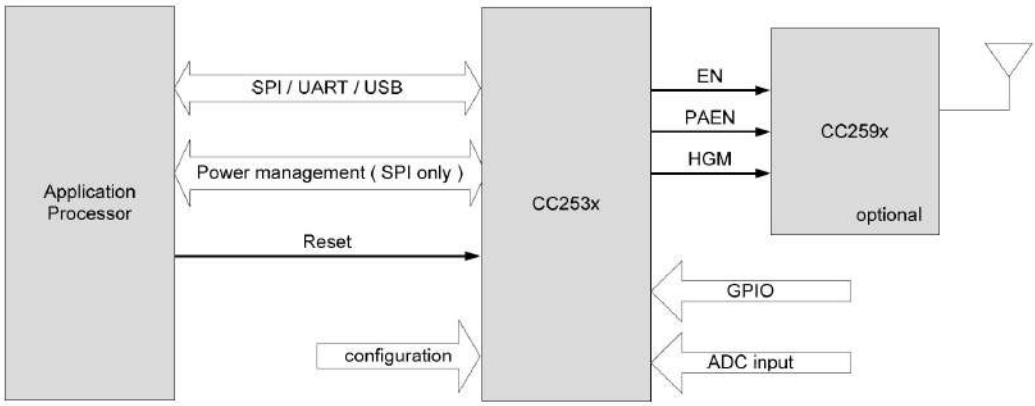
Εικόνα 4.7 Smart flash programmer με το αρχείο ZNP hex

4.5.3 Ανάλυση χαρακτηριστικών του ZNP

Η Zstack ZNP ελέγχει όλες τις διαδικασίες του zigbee και αφήνει τον έλεγχο της εφαρμογής σε οποιονδήποτε εξωτερικό μικροελεγκτή. Η Zstack ZNP αλληλεπιδρά με τον εξωτερικό μικροελεγκτή μέσω σειριακής επικοινωνίας. [7]

4.5.3.1 Το φυσικό επίπεδο του ZNP

Οι ελάχιστες συνδέσεις που απαιτούνται για την επικοινωνία του ZNP με τον μικροελεγκτή της εφαρμογής(ZAP-Zigbee Application Processor) φαίνονται στην παρακάτω εικόνα.



Εικόνα 4.8 Σύνδεση ZNP-ZAP

UART: Περιλαμβάνει τα σήματα TX(trasmit data), RX(Receive data),CT(Clear to send),RT(Ready to send). Επειδή έχουμε καταργήσει το flow control από το firmware του ZNP τα σήματα CT και RT παραλείπονται.
RESET: Αυτό το σήμα χρησιμοποιείται από τον Application Processor για την επανεκκίνηση του CC2530
CFG0,CFG1: Είναι τα δύο configuration signals που διαβάζει ο ZNP κατά τη διάρκεια την εκκίνησης για να παραμετροποιήσει μερικές παραμέτρους.

Το CFG0 είναι για να καθορίζει την παρουσία ή απουσία ενός εξωτερικού κρυστάλλου 32kHz. Ένας τέτοιος κρύσταλλος μπορεί να χρησιμοποιηθεί για τη διάρκεια που ο ZNP βρίσκεται σε κατάσταση sleep mode. Η χρήση εξωτερικού κρυστάλλου αντί του εσωτερικού RC κρυστάλλου βοηθάει σε γρηγορότερη έξοδο του ZNP από το sleep mode και σε χαμηλότερη κατανάλωση σε αυτή τη χρονική διάρκεια.

Απο προεπιλογή ο ZNP χρησιμοποιεί το σειριακό πρότυπο SPI αντί του UART. Εκτός λοιπόν απο τις αλλαγές που έγιναν στο firware, για να μεταβούμε απο το SPI στο UART πρέπει να έχουμε το CFG1 σε κατάσταση low.

GPIO0-3: Στον ZNP περισσεύουν 4 ακροδέκτες γενικής χρήσης, αυτός ήταν και ο λόγος που δεν απαιτούσαμε πολλούς ελεύθερους ακροδέκτες στην επιλογή του PIC12F1840. Η πρόσβαση σε αυτούς τους ακροδέκτες από τον ZAP γίνεται πάλι μέσω του διαύλου UART.

Οι ακροδέκτες του CC2530 μπορούν να ρυθμιστούν με δύο τρόπους, ως main pin configuration και ως alternative pin configuration. Στην κατασκευή μας επιλέξαμε την εναλλακτική ρύθμιση όπως φένεται παρακάτω. [7]

CC2530-ZNP signal	CC2530 PIN	CC2530 NAME	*Direction (on C2530)*
CT	15	P0_4	*In*
RT	14	P0_5	*In / Out*
TX	16	P0_3	*In / Out*
RX	17	P0_2	*Out / In*
RESET	20	RESET_N	*In*
PAEN	9	P1_1	*Out*
EN	6	P1_4	*Out*
HGM	12	P0_7	*Out*
CFG0	8	P1_2	*In*
CFG1	36	P2_0	*In*
GPIO0/AIN0	19	P0_0	*Configurable*
GPIO1/AIN1	18	P0_1	*Configurable*
GPIO2	13	P0_6	*Configurable*
GPIO3	*11*	*P1_0*	*Configurable*

Εικόνα 4.9 ZNP alternative pin configuration

4.5.3.2 UART πλαίσια

Η επικοινωνία του UART έχει ρυθμιστεί ως εξής:
Baud rate: 115200
Flow Control: FALSE
8-N-1 η μορφή κάθε byte

Κάθε ακολουθία bytes οργανώνεται σε μορφή πλαισίων(frames) για τη σωστή επικοινωνία ZNP-ZAP. Τα πλαίσια αυτά έχουμε μια συγκεκριμένη δομή που ορίζεται όπως φαίνεται στην παρακάτω εικόνα.[7]

Bytes: 1	3-253	1
SOF	General format frame	FCS

Εικόνα 4.10 Η δομή ενός πλαισίου

SOF: Κάθε πλαίσιο ξεκινάει πάντα με ένα συγκεκριμένο byte που σηματοδοτεί την έναρξη του πλαισίου. Αυτό είναι πάντα το byte 0xFE.
General format frame: Είναι το ωφέλιμο φορτίο του πλαισίου. Περιέχει όλες τις πληροφορίες, τις εντολές και τα δεδομένα.
FCS: Frame-check sequence. Το τελευταίο byte του πλαισίου είναι το αποτέλεσμα της λογικής πράξης XOR μεταξύ όλων των bytes στο general format frame. Χρησιμεύει για την επαλήθευση της επιτυχημένης σωστής μετάδοσης του πλαισίου.
Η δομή του General frame φαίνεται στην παρακάτω εικόνα:

Bytes: 1	2	0-250
Length	Command	Data

Εικόνα 4.11 Η δομή του General frame

Lenght: Το μέγεθος του πεδίου data που ακολουθεί
Command: Η κατηγορία του πλαισίου. Ποιά εντολή εκτελεί το συγκεκριμένο πλαίσιο
Data: Εξαρτάται από το command field. Περιλαμβάνει τα δεδομένα που μεταφέρει κάθε εντολή.
Το command field αποτελείται από 2 bytes, Αυτά τα 2 bytes σχηματίζονται σύμφωνα με την παρακάτω εικόνα.

Cmd0		Cmd1
Bits: 7-5	4-0	7-0
Type	Subsystem	ID

Εικόνα 4.12 Command field

Type: Έχει μια από τις ακόλουθες τιμές

0:POLL. Αυτή η εντολή εφαρμόζεται στην περίπτωση του SPI για δεδομένα που βρίσκονται σε ουρά

1: SREQ. Ένα synchronous request που απαιτεί μια άμεση ανταπόκριση. Μια συνάρτηση για παράδειγμα που περιμένει μια τιμή επιστροφής.

2: AREQ. Ένα asynchronous request. Για παράδειγμα, ένα callback event ή μια συνάρτηση χωρίς τιμή επιστροφής.

3: SRSP. A synchronous response. Στέλνεται σε ανταπόκριση της εντολής SREQ.

4-7: Reserved

Subsystem: Οι τιμές αυτού του πεδίου είναι στον ακόλουθο πίνακα.

Subsystem Value	Subsystem Name
0	RPC Error interface
1	SYS interface
2	Reserved
3	Reserved
4	AF interface
5	ZDO interface
6	Simple API interface
7	UTIL interface
8	Reserved
9	APP Interface
10-31	Reserved

Εικόνα 4.13 Οι τιμές του πεδίου Subsystem

ID: Καθορίζει ένα συγκεκριμένο μήνυμα. Παίρνει τιμές 0-255

4.5.3.3 Πλαίσια εντολών για τον CC2530-ZNP

Τα πλαίσια που ανταλλάσσονται μεταξύ ZNP και ZAP χωρίζονται στις παρακάτω υποκατηγορίες:

Οι εντολές τύπου SYS παρέχουν τη δυνατότητα στον ZAP να αλληλεπιδρά με τα χαμηλότερα επίπεδα hardware και software του ZNP. Όπως για παράδειγμα είναι η non-volitele μνήμη του CC2530 και οι drivers που είναι γραμμένοι στο επίπεδο HAL για την χρήση των περιφερειακών που υπάρχουν στο CC2530, εάν χρησιμοποιούνται, ή για τον έλεγχο των GPIO.

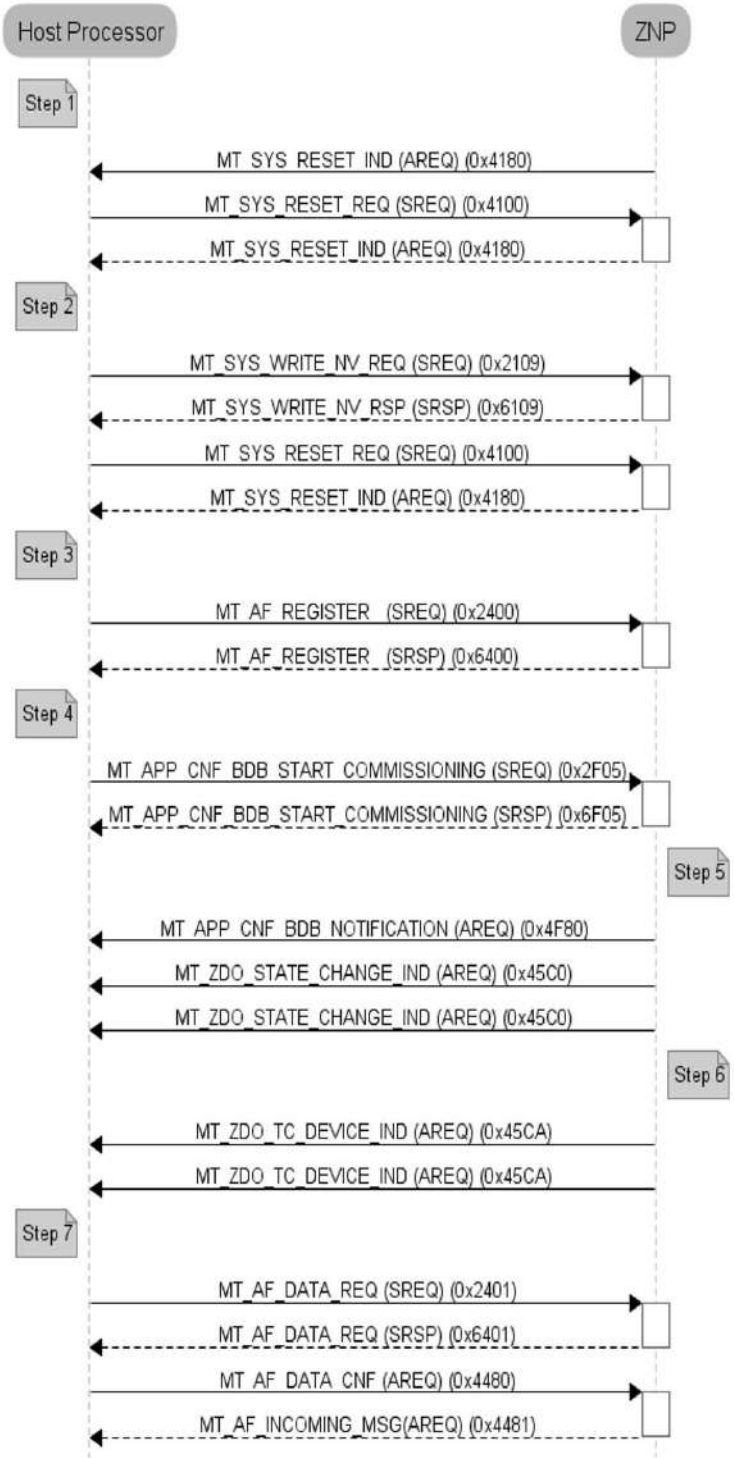
Οι εντολές simple API είναι απλοποιημένες εντολές με λίγες παραμέτρους με τις οποίες έχουμε πρόσβαση στις βασικές λειτουργίες zigbee στο επίπεδο εφαρμογής. Με αυτό το απλοποιημένο interface υπάρχουν μερικοί περιορισμοί για την εφαρμογή όπως είναι η δυνατότητα χρήσης μόνο ενός endpoint.

Οι εντολές τύπου AF και ZDO δίνουν πρόσβαση σε ολόκληρη τη στοίβα zigbee χωρίς κανέναν περιορισμό.

4.5.3.4 Διαδικασία αρχικής ρύθμισης του ZNP από τον ZAP

Μετά την εφαρμογή της τροφοδοσίας σε ZNP και ZAP θα πρέπει να ανταλλάξουν μεταξύ τους συγκεκριμένα πλαίσια για την αρχική ρύθμιση του ZNP πρώτου ξεκινήσει η επικοινωνία zigbee over-the-air. Εάν αυτή η διαδικασία δεν ολοκληρωθεί με επιτυχία μπορεί να προκαλέσει απρόοπτη συμπεριφορά του CC2530. [7]Οι προτεινόμενες αρχικές ρυθμίσεις που κάνει ο ZAP ακολουθούν τα παρακάτω βήματα:

1. Κρατάει τον CC2530 σε κατάσταση reset έχοντας το αντίστοιχο pin σε κατάσταση low.
2. Ρυθμίζει τις παραμέτρους UART σύμφωνα με αυτές που έχουν οριστεί στο firmware του ZNP και στη συνέχεια καθορίζει τις τάσεις στα configuration pin CFG0,CFG1 εάν δε βρίσκονται σε σταθερή τάση.
3. Επαναφέρει το σήμα reset, ο CC2530 ξεκινάει τη λειτουργία του και ο ZAP αναμένει μέχρι ο CC2530 να ενημερώσει με το πλαίσιο SYS_RESET_IND(reset indication)
4. Στη συνέχεια ο ZAP θα πρέπει να χρησιμοποιήσει την εντολή ZB_WRITE_CONFIGURATION για να γράψει στην μνήμη του CC2530 και να ρυθμίσει τον ρόλο(ZCD_NV_LOGICAL_TYPE) που θα έχει η συσκευή στο δίκτυο.
5. Προαιρετικά μπορούν να ρυθμιστούν το κανάλι λειτουργίας και το PAN ID εάν δεν έγινε κατα την διάρκεια compile του firmware.
6. Έπειτα πρέπει να σταλεί η εντολή AF_REGISTER για να γίνει η εγγραφή των endpoints που θα χρησιμοποιηθούν απο την εφαρμογή
7. Τέλος, μπορούν να χρησιμοποιηθούν εντολές API και ZDO για τη δημιουργία και σύνδεση στο δίκτυο και στη συνέχεια να ακολουθήσει η ανταλλαγή μηνυμάτων.



Εικόνα 4.14 Παράδειγμα ανταλλαγής πακέτων ZNP-ZAP για την ρύθμιση του ZNP

4.5.3.5 Ανάλυση μερικών εντολών

SYS_RESET_REQ

Ένας τρόπος για να πραγματοποιηθεί η λειτουργία reset του CC2530 είναι μέσω αυτής της εντολής. Ωστόσο προτείνεται το hardware reset, με την αλλαγή κατάσταση στον ακροδέκτη reset διότι είναι ποιο γρήγορος και αξιόπιστος στην εκτέλεση.[18]

SREQ:

1	1	1	1
Length = 0x01	Cmd0 = 0x41	Cmd1 = 0x00	Type

Type: Με την τιμή 0 εκτελείται hardwawre reset και με την τιμή 1 εκτελείται soft reest(μετάβαση στο reset vector).

SYS_PING
Εντολή για να διαπιστωθεί εάν η συσκευή είναι ενεργή.
SREQ:

1	1	1
Length = 0x00	Cmd0 = 0x21	Cmd1 = 0x01

SYS_OSAL_NV_WRITE
Χρησιμοποιείται για την εγγραφή διάφορων τιμών στη non-volatile μνήμη του CC2530.[18]
SREQ:

1	1	1	2	1	1	1-246
Length = 0x04-0xFA	Cmd0 = 0x21	Cmd1 = 0x09	Id	Offset	Len	Value

Παράμετροι:
ID: Το id του NV αντικειμένου
Offset: Ο αριθμός bytes offset απο την έναρξη της τιμής
Len: Το μέγεθος του NV αντικειμένου
Value: Η τιμή του αντικειμένου NV

SYS_GPIO
Εντολή για τον έλεγχο των 4 GPIO του CC2530-ZNP

1	1	1	1	1
Length = 0x02	Cmd0 = 0x21	Cmd1 = 0x0E	Operation	Value

Παράμετροι:
Operation: Set direction(0x00)
Set input mode(0x01)
Set(0x02)
Clear(0x03)
Toggle(0x04)
Read(0x05)
Value: Η τιμή του εξαρτάται απο το operation

UTIL_SET_PANID
Αποθηκεύει την τιμή του επιθυμητού panid στη non-volatile μνήμη[18]
SREQ:

1	1	1	2
Length = 0x02	Cmd0 = 0x27	Cmd1 = 0x02	PanId

Παράμετροι:
Panid: Η τιμή PANID που θα αποθηκευτεί

UTIL_SET_CHANNELS
Αποθηκεύει μια bit-mask στη non-volatile μνήμη που αντιπροσωπεύει τα κανάλια που θα σκανάρει ο coordinator.[18]
SREQ:

1	1	1	4
Length = 0x04	Cmd0 = 0x27	Cmd1 = 0x03	Channels

Παράμετροι:

Channels:Η τιμή του bit-mask

ZDO_NWK_ADDR_REQ

Στέλνει ένα broadcast μήνυμα στο δίκτυο προκειμένου να λάβουμε τη 16bit διεύθυνση μιας συσκευής με γνωστή τη διεύθυνση 64bit IEEE[18]

SREQ:

Byte: 1	1	1	8	1	1
Length = 0x0A	Cmd0 = 0x25	Cmd1 = 0x00	IEEEAddress	ReqType	StartIndex

Παράμετροι:

IEEEAddress: Η γνωστή 64bit διεύθυνση

ReqType: Παίρνει την τιμή 0x00 για απλό respose

StartIndex: χρησιμοποιείται εάν θέλουμε να λάβουμε λίστα με τις συνδεδεμένες συσκευές στη συσκευή που τίθεται το ερώτημα

ZDO_NWK_IEEE_REQ

Η αντίστοιχη εντολή με την προηγούμενη όταν γνωρίζουμε τη διεύθυνση 16bit

ZDO_SIMPLE_DESC_REQ

Εντολή για τη λήψη του simple descriptor ενός endpoint μίας συσκευής στο δίκτυο[18]

SREQ:

Byte: 1	1	1	2	2	1
Length = 0x05	Cmd0 = 0x25	Cmd1 = 0x04	DstAddr	NWKAddrOfInterest	Endpoint

Παράμετροι:

DstAddr: Η 16bit διεύθυνση του κόμβου που δέχεται την ερώτηση

NWRAddrOfInterest: Η 16bit διεύθυνση του κόμβου που κάνει την ερώτηση

Endpoint: Το endpoint που επιθυμούμε να μάθουμε τον simple descriptor

ZDO_ACTIVE_EP_REQ

Εντολή για να λάβουμε τη λίστα με τα ενεργά endpoints μίας συσκευής στο δίκτυο.[18]

SREQ:

Byte: 1	1	1	2	2
Length = 0x04	Cmd0 = 0x25	Cmd1 = 0x05	DstAddr	NWKAddrOfInterest

Παράμετροι:

DstAddr: Η 16bit διεύθυνση του κόμβου που δέχεται την ερώτηση

NWRAddrOfInterest: Η 16bit διεύθυνση του κόμβου που κάνει την ερώτηση

ZDO_END_DEVICE_ANNCE

Μέσω αυτής της εντολής στέλνεται ένα μήνυμα broadcast στο δίκτυο όταν συνδέεται σε αυτό μια συσκευή end device.[18]

SREQ:

1	1	1	2	8	1
Length = 0x0B	Cmd0 = 0x25	Cmd1 = 0x0A	NwkAddr	IEEEAddr	Capabilites

Παράμετροι:

NwkAddr: Η 16bit διεύθυνση της συσκευής

IEEEAdr: Η 64bit διεύθυνση της συσκευής

Capabilities: 1 – Device type: 1- ZigBee Router; 0 – End Device

2 – Power Source: 1 Main powered

3 – Receiver on when Idle

4 – Reserved

5 – Reserved

6 – Security capability

ZDO_MGMT_PERMIT_JOIN_REQ

Στέλνεται στον coordinator και στους routers για να επιτραπεί η σύνδεση συσκευών στο δίκτυο για ένα συγκεκριμένο χρονικό διάστημα.[18]
SREQ:

1	1	1	1	2	1	1
Length = 0x05	Cmd0 = 0x25	Cmd1 = 0x36	AddrMode	DstAddr	Duration	TCSignificance

Παράμετροι:

AddrMode: 0x02 για 16bit Address mode και 0xFF για Broadcast
DstAddr: Η διεύθυνση της συσκευής που θα επιτρέψει τη σύνδεση άλλων συσκευών
Duration: Ο χρόνος που θα επιτρέπεται η σύνδεση. 0 = join disabled. 0xff = join enabled. 0x01-0xfe = Ο χρόνος σε δευτερόλεπτα.
TCSignificance: Trust Center Significance.

AF_REGISTER
Αυτή η εντολή μας επιτρέπει να κάνουμε εγγραφή ενός endpoint και να δώσουμε την περιγραφή του. [18]
SREQ:

1	1	1	1	2	2
Length = 0x09-0x49	Cmd0 = 0x24	Cmd1 = 0x00	EndPoint	AppProfId	AppDeviceId

1	1	1	0-32	1	0-32
AppDevVer	LatencyReq	AppNumInClusters	AppInClusterList	AppNumOutClusters	AppOutClusterList

Παράμετροι:

EndPoint: Καθορίζει το endpoint προς εγγραφή
AppProfId: Καθορίζει το profile id της εφαρμογής
AppDeviceId: Καθορίζει το device description id του endpoint
AppDevVer: Η έκδοση της συσκευής
LatencyReq: 0x00-No latency
0x01-fast beacons
0x02-slow beacons
AppNumInClusters: Ο αριθμός των Input cluster Id's που ακολουθούν στη λίστα
AppInClusterList: Η λίστα των Input cluster Id's
AppNumOutClusters: Ο αριθμός των Output cluster Id's που ακολουθούν στη λίστα
AppOutClusterList: Η λίστα των Output cluster Id's

AF_DATA_REQUEST
Με αυτή την εντολή δημιουργείται και στέλνεται ένα μήνυμα μέσω του AF layer.[18]
SREQ:

Byte: 1	1	1	2	1
Length = 0x0A-0x8A	Cmd0 = 0x24	Cmd1 = 0x01	DstAddr	DstEndpoint

Byte: 1	2	1	1	1	1	0-128
SrcEndpoint	ClusterId	TransId	Options	Radius	Len	Data

Παράμετροι:

DstAddr: NWK Address της συσκευής προορισμού
DstEndpoint: Endpoint της συσκευής προορισμού
SrcEndpoint: Endpoint της συσκευής που παράγει το μήνυμα
ClusterId: Καθορίζει το Cluster id
TransId: Ορίζει το transaction sequence number του μηνύματος
Options:Bit mask για επιλογές μετάδοσης σύμφωνα με το αρχείο AF.h:
bit 1: sets 'Wildcard Profile ID'
bit 4: turns on/off 'APS ACK'
bit 5 sets 'discover route'
bit 6 sets 'APS security';
bit 7 sets 'skip routing'.

Radius: Ο αριθμός των hops που επιτρέπεται μέχρι να απορριφθεί το μήνυμα εάν δεν έχει φτάσει στον προορισμό του

Len: Το μέγεθος των data

Data: 0-128 bytes data

Callbacks:

AF_INCOMING_MSG

Δημιουργείται όταν υπάρχουν δεδομένα σε οποιοδήποτε endpoint.[18]

SREQ:

1	1	1	2	2	2	1
Length = 0x11-0x91	Cmd0 = 0x44	Cmd1 = 0x81	GroupId	ClusterId	SrcAddr	SrcEndpoint

1	1	1	1	4	1	1	0-128
DstEndpoint	WasBroadcast	LinkQuality	SecurityUse	Timestamp	TransSeqNumber	Len	Data

Παράμετροι:

GroupId: Το Group Id της συσκευής

ClusterId: Το Cluster Id

SrcAddr: Καθορίζει την 16-bit διεύθυνση της συσκευής που έστειλε το μήνυμα

SrcEndpoint: Το endpoint προέλευσης του μηνύματος

DstEndpoint: Το endpoin προορισμού

WasBroadcast: Καθορίζει αν το μήνυμα ήταν Broadcast ή όχι

LinkQuality: Δείχνει την ποιότητα link quality που μετρήθηκε κατα τη λήψη

SecurityUse: Καθορίζει αν χρησιμοποιήθηκε Security

TimeStamp: Καθορίζει το timestamp του μηνύματος

TransSeqNumber: Το Transaction sequence number του μηνύματος

Len: Το μήκος των δεδομένων

Data: Τα δεδομένα

ZDO_SIMPLE_DESC_RSP

Δημιουργείται ως απάντηση στο μήνυμα ZDO Simple Descriptor Request.[18]

SREQ:

1	1	1	2	1	2	1
Length = 0x06-4E	Cmd0 = 0x45	Cmd1 = 0x84	SrcAddr	Status	NwkAddr	Len

1	2	2	1
Endpoint	ProfileId	DeviceId	DeviceVersion

1	0-32	1	0-32
NumInClusters	InClusterList	NumOutClusters	OutClusterList

Παράμετροι:

SrcAddr: Καθορίζει τη 16-bit διεύθυνση της συσκευής που έστειλε το μήνυμα

Status: Δείχνει αν έφτασε το μήνυμα με επιτυχία SUCCESS ή FAILURE

NWKAddr: Η 16-bit διεύθυνση της συσκευής που περιγράφεται σε αυτό το response

Len: Το μήκος του simple descriptor

Endpoint: Το endpoint που περιγράφεται

ProfileId: Το Profile Id του endpoint

DeviceId: Το Device Id του endpoint

DeviceVersion:0-version 1.00

NumInClusters: Ο αριθμός των input clusters στη λίστα InClusterList

InClusterList: Λίστα των cluster id που υποστηρίζονται στο endpoint

NumOutClusters: Ο αριθμός των output clusters στη λίστα OutClusterList

OutClusterList: Λίστα των cluster id που υποστηρίζονται στο endpoint

ZDO_ACTIVE_EP_RSP.

Δημιουργείται ως απάντηση στο μήνυμα ZDO Active Endpoint Request.[18]
SREQ:

1	1	1	2	1	2	1
Length = 0x06-0x53	Cmd0 = 0x45	Cmd1 = 0x85	SrcAddr	Status	NwkAddr	ActiveEPCount

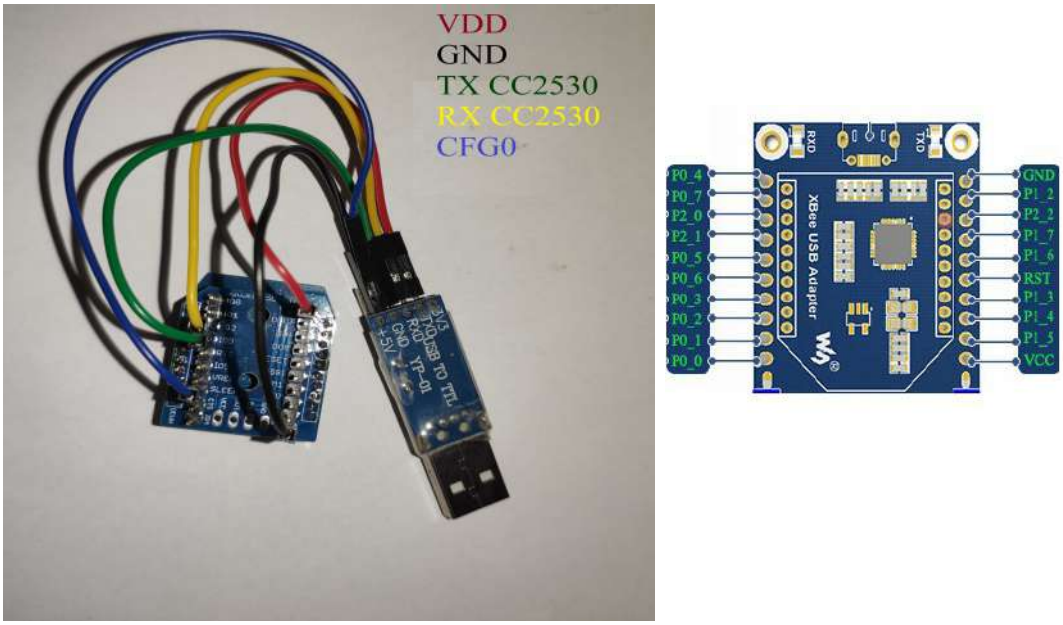
0-77
ActiveEPList

Παράμετροι:
SrcAddr: Καθορίζει τη 16-bit διεύθυνση της συσκευής που έστειλε το μήνυμα
Status: Δίχνει αν έφτασε το μήνυμα με επιτυχία SUCCESS ή FAILURE
NWKAddr: Η 16-bit διεύθυνση της συσκευής που περιγράφεται σε αυτό το response
ActiveEPCount: Ο αριθμός των ενεργών endpoint στη λίστα
ActiveEPList: Πίνακας με τα ενεργά endpoints στη συσκευή

4.5.4 Πρώτο δίκτυο zigbee με το ztool

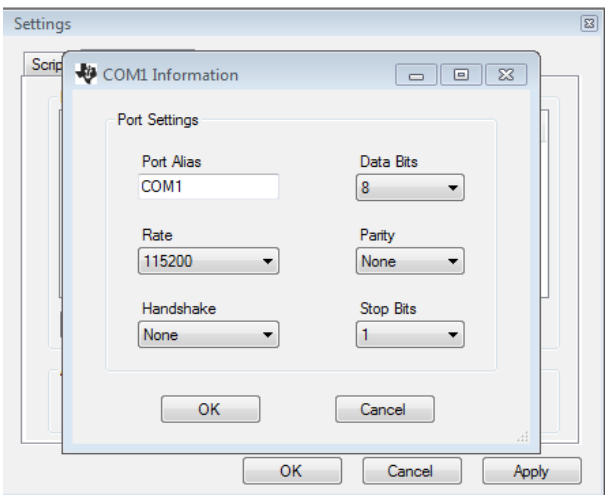
Το Ztool είναι ένα εργαλείο σε μορφή λογισμικού που προσφέρεται από την TI για τη διαδικασία του monitoring. Μοιάζει σαν ένα πρόγραμμα terminal όπου γράφουμε και στέλνουμε δεδομένα μέσω σειριακής σύνδεσης. Προτού αρχίσουμε να γράφουμε κατάλληλο κώδικα στον ZAP για να ρυθμίσουμε τον ZNP και να δημιουργήσουμε το δίκτυο zigbee, μπορούμε να κάνουμε τις δοκιμές μας με το Ztool αντί του ZAP.[11],[18]

Αρχικά θα πρέπει να ετοιμάσουμε το CC2530B module ώστε να συνδεθεί με τον υπολογιστή σαν μια σειριακή συσκευή. Γι αυτόν τον σκοπό θα χρειαστούμε έναν μετατροπέα USB to TTL Οι συνδέσεις που πρέπει να κάνουμε είναι οι δίαυλοι επικοινωνίας TX και RX, τα configuration pins CFG0,CFG1 και η τροφοδοσία του module.



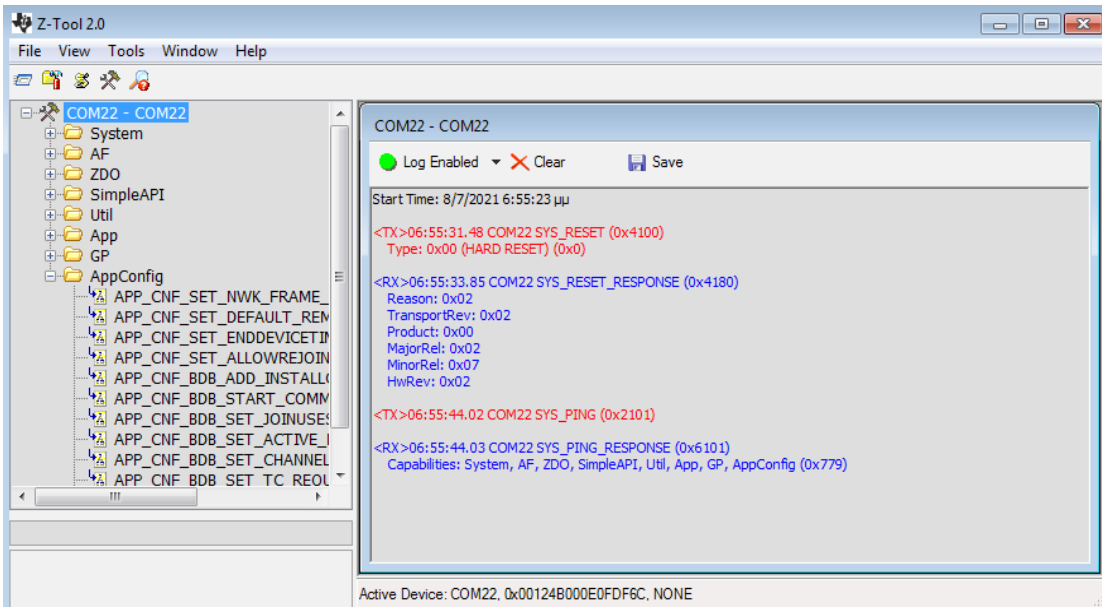
Εικόνα 4.15 Σύνδεση USB-TTL adapter με Core2530B

Για να αναγνωριστεί η συσκευή από το Ztool θα πρέπει να ρυθμίσουμε το Ztool στις κατάλληλες επιλογές UART.



Εικόνα 4.16 Ρυθμίσεις UART για το ZTool

Μετά την επιτυχημένη σύνδεση θα λάβουμε στο Ztool το μήνυμα reset indication από τον CC2530. Τώρα είμαστε έτοιμοι να στείλουμε εντολές στον CC2530 μέσω του Ztool.



Εικόνα 4.17 Σύνδεση Core2530B με ZTool και ανταλλαγή μηνυμάτων

Η πρώτη δοκιμή που μπορούμε να κάνουμε είναι να ρυθμίσουμε τον CC2530 σαν coordinator σύμφωνα με τη διαδικασία ρύθμισης στην ενότητα 4.5.3.4 και στη συνέχεια να δημιουργήσουμε το δίκτυο. Ακολουθούν τα πλαίσια που ανταλλάχθηκαν μεταξύ Ztool και CC2530, με έντονα γράμματα είναι να πλαίσια που στάλθηκαν από το Ztool και τα υπόλοιπα είναι οι ανταποκρίσεις.[28]

COM1 SYS_OSAL_NV_WRITE (0x2109) -->Write startup option to clear NV when reset
Id: 0x0003
Offset: 0x00
Len: 0x01
Value: . (0x03)

COM1 SYS_OSAL_NV_WRITE_SRSP (0x6109)
Status: SUCCESS (0x0)

COM1 SYS_RESET (0x4100)-->Do reset to clear NV
Type: 0x00 (HARD RESET) (0x0)

COM1 SYS_RESET_RESPONSE (0x4180)
Reason: 0x02
TransportRev: 0x02
Product: 0x00
MajorRel: 0x02
MinorRel: 0x07
HwRev: 0x00

COM1 SYS_OSAL_NV_WRITE (0x2109)-->Write ZCD_NV_LOGICAL_TYPE to 0 which means coordinator
Id: 0x0087

Offset: 0x00
Len: 0x01
Value: . (0x00)

COM1 SYS_OSAL_NV_WRITE_SRSP (0x6109)
Status: SUCCESS (0x0)

10:38:32.55 COM1 APP_CNF_BDB_SET_CHANNEL (0x2F08)---> Set Primary channel mask to channel 13 only
isPrimary: TRUE (0x1)
Channel: CHNL_0x00002000 (0x2000)

COM1 APP_CNF_BDB_SET_CHANNEL_SRSP (0x6F08)
Status: SUCCESS (0x0)

COM1 APP_CNF_BDB_SET_CHANNEL (0x2F08)--> Set Secondary channel to 0x0 to disable secondary channel mask
isPrimary: FALSE (0x0)
Channel: NONE (0x0)

COM1 APP_CNF_BDB_SET_CHANNEL_SRSP (0x6F08)
Status: SUCCESS (0x0)

COM1 APP_CNF_BDB_START_COMMISSIONING (0x2F05)-->Start commissioning using network formation as parameter to start coordinator
CommissioningMode: (0x04) Network Formation (0x4)

COM1 APP_CNF_BDB_START_COMMISSIONING_SRSP (0x6F05)
Status: SUCCESS (0x0)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 APP_CNF_BDB_COMMISSIONING_NOTIFICATION (0x4F80)
Status: 1 (0x1)
Commissioning Mode: 0x02 (Formation) (0x2)
Commissioning Mode: 254 (0xFE)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 8 (0x8)

COM1 ZDO_STATE_CHANGE_IND (0x45C0)
State: 9 (0x9)
COM1 APP_CNF_BDB_COMMISSIONING_NOTIFICATION (0x4F80)
Status: 0x00 (Success) (0x0)
Commissioning Mode: 0x02 (Formation) (0x2)
Commissioning Mode: 69 (0x45)

COM1 UTIL_GET_DEVICE_INFO (0x2700)-->Get device info to confirm coordinator is setup correctly

COM1 UTIL_GET_DEVICE_INFO_RESPONSE (0x6700)
Status: SUCCESS (0x0)
IEEEAddr: 0x00124B0001025822
ShortAddress: 0x0000
DeviceType: COORDINATOR, ROUTER, END_DEVICE (0x7)
DeviceState: DEV_ZB_COORD (0x9)
NumAssocDevices: 0x00
AssocDevicesList

COM1 SYS_OSAL_NV_WRITE (0x2109)-->Write ZCD_NV_ZDO_DIRECT_CB to 1 to receive ZDO related messages
Id: 0x008F
Offset: 0x00
Len: 0x01
Value: . (0x01)

COM1 SYS_OSAL_NV_WRITE_SRSP (0x6109)
Status: SUCCESS (0x0)

COM1 APP_CNF_BDB_START_COMMISSIONING (0x2F05)-->Start commissioning using network steering as parameter to be ready for device to join
CommissioningMode: (0x02) Network Steering (0x2)

COM1 APP_CNF_BDB_START_COMMISSIONING_SRSP (0x6F05)
Status: SUCCESS (0x0)

COM1 ZDO_MGMT_PERMIT_JOIN_RSP (0x45B6)
SrcAddr: 0x0000
Status: ZDP_SUCCESS (0x0)

COM1 APP_CNF_BDB_COMMISSIONING_NOTIFICATION (0x4F80)
Status: 0x00 (Success) (0x0)
Commissioning Mode: 0x01 (Network Steering) (0x1)
Commissioning Mode: 67 (0x43)

Start up option

Γενικά στη συσκευή ZNP υπάρχουν 2 ειδών πληροφορίες αποθηκευμένες στη μνήμη non-volatile. Οι configuration parameters όπως είναι το Start up option και οι network state information όπως είναι το PANID του δικτύου, το κανάλι λειτουργίας κτλ.

Οι configuration parameters ρυθμίζονται από τον χρήστη πριν ξεκινήσει η εκτέλεση του zigbee.

Οι network state information συλλέγονται από τη συσκευή αφού συνδεθεί σε κάποιο δίκτυο, δε ρυθμίζονται απο τον ZAP. Αυτές οι πληροφορίες αποθηκεύονται ώστε αν υπάρξει κάποιο τυχαίο reset ή αλλαγή μπαταρίας στις φορητές συσκευές να μην χρειαστεί ξανά η διαδικασία σύνδεσης στο δίκτυο.

Εάν ο ZAP επιθυμεί να μη συνεχίσει τη λειτουργία στο υπάρχον δίκτυο θα πρέπει να δώσει εντολή στον ZNP να καθαρίσει τις πληροφορίες network state από τη μνήμη και να ξανά ξεκινήσει με βάση τις παραμέτρους configuration

Η start up option είναι μια επιλογή τύπου configuration μήκους ενός byte με id 0x03 που καταγράφουμε στη non-volatile μνήμη του CC2530. Κάθε φορά που τροφοδοτείται ο CC2530 διαβάζει αυτή την τιμή.[12]

Item ID	Size	Default Value
0x0003	1 Byte	0x00

Εικόνα 4.18 Start up option

Οι τιμές που μπορεί να πάρει είναι οι εξής:

- ZCD_STARTOPT_DEFAULT_CONFIG_STATE (0x01)
- ZCD_STARTOPT_DEFAULT_NETWORK_STATE (0x02)
- ZCD_STARTOPT_AUTO_START (0x04)
- ZCD_STARTOPT_CLEAR_CONFIG (ZCD_STARTOPT_DEFAULT_CONFIG_STATE)
- ZCD_STARTOPT_CLEAR_STATE (ZCD_STARTOPT_DEFAULT_NETWORK_STATE)
- ZCD_STARTOPT_CLEAR_NWK_FRAME_COUNTER (0x80)

Bit Position	Description
7	ZCD_STARTOPT_CLEAR_NWK_FRAME_COUNTER
6-2	Reserved
1	ZCD_STARTOPT_CLEAR_STATE
0	ZCD_STARTOPT_CLEAR_CONFIG

Εικόνα 4.19 Start up option values

ZCD_STARTOPT_CLEAR_CONFIG:

Εάν αυτό το bit είναι σε κατάσταση set τότε η συσκευή θα γράψει όλες τις default τιμές στις παραμέτρους οδηγώντας έτσι τη συσκευή σε μια γνωστή εργοστασιακή κατάσταση.

ZCD_STARTOPT_CLEAR_STATE:

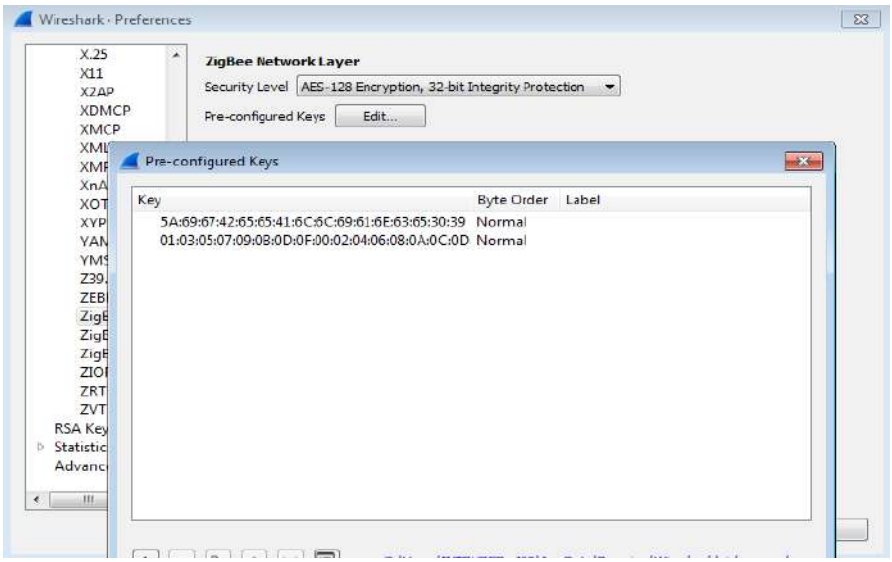
Εάν αυτό το bit είναι σε κατάσταση set τότε η συσκευή θα διαγράψει όλες τις πληροφορίες του δικτύου στο οποίο βρισκόταν πριν το reset, εάν είχε συνδεθεί σε κάποιο δίκτυο. Είναι χρήσιμο κατα τη διάρκεια ανάπτυξης της κατασκευής όπου το δίκτυο δεν παίρνει την τελική του μορφή έως ότου

ολοκληρώσουμε τις δοκιμές μας. Κατα τη διάρκεια λειτουργίας του τελικού συναγερμού αυτό θα πρέπει να έχει την τιμή 0.

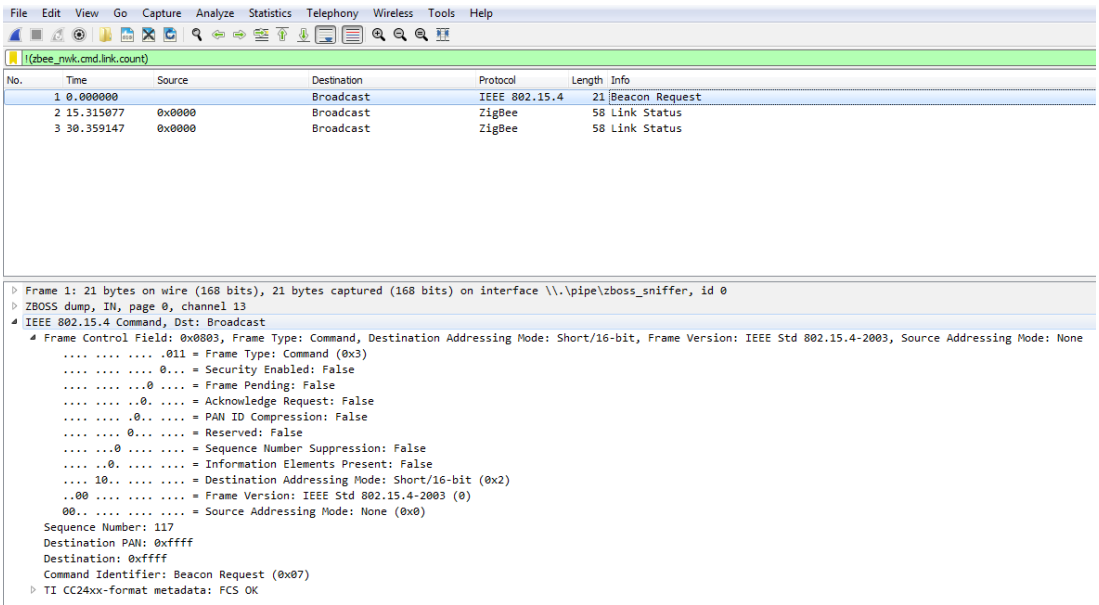
CC2531 Sniffer

Ένα από τα χρησιμότερα εργαλεία για την ανάπτυξη ασύρματων δικτύων είναι ο sniffer. Με τη βοήθεια του μπορούμε να εξετάσουμε όλα τα πακέτα που ανταλλάσσονται στον αέρα. Ένας κατάλληλος sniffer για το zigbee είναι αυτός της TI σε μορφή dongle που βασίζεται στον CC2531. Όπως και με το module Core2530B έτσι και το CC2531 εργοστασιακά έχει το δικό του firmware που είναι κατάλληλο για προγράμματα καταγραφής πακέτων και συγκεκριμένα για το Packet sniffer της TI. Ακολουθώντας όμως παρόμοια διαδικασία με αυτή που κάναμε στον CC2530 μπορούμε να αντικαταστήσουμε το firmware με αυτό της επιλογής μας. Το firmware που επιλέχθηκε λέγεται ZBOSS. Η επιλογή του έγινε διότι είναι συμβατό με το πρόγραμμα Wireshark.

Με την έναρξη του ZBOSS και του Wireshark χρειάζεται να κάνουμε λίγες ρυθμίσεις. Αρχικά θα πρέπει να επιλέξουμε το κανάλι στο οποίο θα ακούει για μηνύματα zigbee, αυτός είναι και ο λόγος για τον οποίο στις δοκιμές είχαμε σταθερό κανάλι λειτουργίας καθώς επίσης με αυτόν τον τρόπο γλυτώνουμε και την καθυστέρηση του energy scan. Έπειτα το wireshark θα αρχίσει να συλλέγει μηνύματα τα οποία όμως είναι κρυπτογραφημένα. Για να αναγνωρίζονται πλήρως ο Sniffer θα πρέπει να γνωρίζει δύο κλειδιά, το Trust Center link key, το οποίο είναι ίδιο σχεδόν σε όλα τα zigbee δίκτυα(5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39) και το network encryption key που βάλαμε στο firmware της Zstack στον CC2530. Στο παράθυρο Edit -> Preferences -> Protocols -> ZigBee επιλέγουμε Security Level το *AES-128 Encryption, 32-bit Integrity Protection* και εισάγουμε τα δυο κλειδιά.



Εικόνα 4.20 Ρύθμιση του wireshark



Αφού έχουμε ρυθμίσει το Wireshark μπορούμε να δοκιμάσουμε το δίκτυο το οποίο κάναμε στην προηγούμενη ενότητα με τη βοήθεια του Ztool. Αυτό που περιμένουμε να δούμε στο Wireshark είναι μόνο ένα πακέτο, το πακέτο beacon που έστειλε ο coordinator για την εύρεση τυχόν άλλων δικτύων zigbee στην περιοχή.

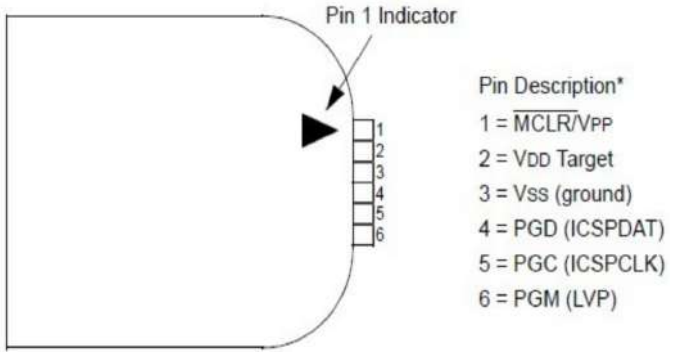
Εικόνα 4.21 Wireshark capture απο την δημιουργία του Zigbee δικτύου

Τέλος, στο Wireshark χρησιμοποιήθηκε το φίλτρο !(zbee_nwk.cmd.link.count). Κάθε router και ο coordinator ανα μερικά δευτερόλεπτα στέλνουν ένα μήνυμα broadcast με radius=1. Αυτό εξυπηρετεί ώστε να αξιολογείται η σύνδεση γειτονικών routers.

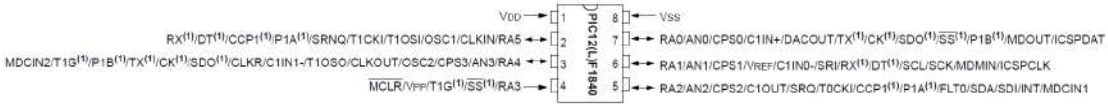
4.5.5 Κώδικας για τον ZAP router

Έχοντας ρυθμίσει έναν κόμβο με το Ztool να συμπεριφέρεται σαν Coordinator μπορούμε να συνδέσουμε συσκευές και να δημιουργήσουμε δίκτυο. Η ποιο απλή συσκευή από άποψη σχεδιασμού εφαρμογής είναι ο router. Αυτό που χρειάζεται είναι να ρυθμίσουμε μερικά configuration bits και να τον συνδέσουμε στον coordinator. Το πρόγραμμα που θα κάνει αυτή τη δουλειά, αντί για το Ztool θα είναι στον PIC12F1840.

Για τον προγραμματισμό του PIC χρησιμοποιήθηκαν τα εργαλεία Pickit 3 και το Mplab X με τον compiler CCS C λόγω οικειότητας. Αφού εγκαταστήσουμε τον compiler της επιλογής μας στο Mplab κάνουμε και τη σύνδεση του PIC με το pickit 3 σύμφωνα με τα παρακάτω σχεδιαγράμματα.



Εικόνα 4.22 Pickit 3 pinout



Εικόνα 4.23 PIC12F1840 Pinout

Δημιουργία Project και απαραίτητων headers αρχείων

Πρώτα δημιουργούμε ένα καινούργιο project με το wizard του Mplab και εισάγουμε τα headers files για τον συγκεκριμένο μικροελεγκτή κάνοντας τις απαραίτητες τροποποιήσεις που αφορούν την πηγή ταλάντωσης, τη συχνότητα ταλάντωσης, την διεύθυνση του reset και του interrupt vector και μερικά fuses. Σύμφωνα με τον CCS Compiler το αρχείο header main.h θα πρέπει να περιέχει τις εξής γραμμές κώδικα:

```
#include <12F1840.h>
#fuses NOMCLR, INTRC_IO, NOWDT, BROWNOUT, NOPROTECT
#FUSES MCLR
#use delay(INTERNAL=32MHz)
#build (reset=0x000, interrupt=0x008)
```

Για την επικοινωνία UART που θα χρειαστούμε γράφουμε την εντολή pre-processor:

```
#use rs232(baud=115200,parity=N,xmit=PIN_A4,rcv=PIN_A5,bits=8)
```

Η οποία ορίζει το baud rate σε 115200, που έχουμε ρυθμίσει τον CC2530, με δομή πλαισίου 8-N-1 και ακροδέκτες TX και RX αντίστοιχα με PIN A4 και PIN A5.

Η συνάρτηση main

Είναι η συνάρτηση που θα εκτελεστεί αρχικά από τον PIC και θα πρέπει να τη διατηρήσουμε απλή. Στην main() εκτελούνται δυο συναρτήσεις και στη συνέχεια μπαίνει σε έναν άδειο ατέρμον βρόχο διότι η δουλειά του router σε επίπεδο εφαρμογής ολοκληρώνεται με τη ρύθμιση του CC2530.

Οι δύο συναρτήσεις στη main() είναι η init() για την αρχικοποίηση μερικών καταχωρητών στον PIC και η συνάρτηση init_router() η οποία στέλνει τα απαραίτητα πλαίσια για την ρύθμιση του CC2530.

Ο αριθμός των πλαισίων που θα ανταλλαχθούν μεταξύ ZAP-ZNP δεν είναι μεγάλος και τα πλαίσια δε χρειάζεται να παραμετροποιούνται, θα είναι σταθερά. Γι αυτό τον λόγο μπορούμε να τα αποθηκεύσουμε σε μορφή σταθερών πινάκων σε ένα αρχείο headers. Μερικά από αυτά από το αρχείο ConstantFrames.h φαίνονται παρακάτω:

```
unsigned int SYS_OSAL_NV_WRITE_START_UP_OPTION[10] = {0xFE,0x05,0x21,0x09,0x03,0x00,0x00,0x01,0x03,0x2C};//NV clear after reset
unsigned int SYS_RESET[6] = {0xFE,0x01,0x41,0x00,0x00,0x40};    //reset cc2530
unsigned int SYS_OSAL_NV_WRITE[10] = {0xFE,0x05,0x21,0x09,0x87,0x00,0x00,0x01,0x00,0xAB};   //build cc2530 for coordinator
unsigned int SYS_OSAL_NV_WRITE_R[10] = {0xFE,0x05,0x21,0x09,0x87,0x00,0x00,0x01,0x01,0xAA};   //build cc2530 for router
```

Στη συνάρτηση init() ρυθμίζουμε τα pins του PIC ως εισόδους ή εξόδους αναλόγως και τα interrupts που θα χρειαστούμε παρακάτω.

```
void init(void){

    //init MsgRe elements to 0. Array for incoming frames
    for( int i=0; i<40; i++){
    MsgRe[i]=0;
   }

  enable_interrupts(INT_RDA);
  enable_interrupts(INT_RA2);
  enable_interrupts(GLOBAL);
  set_tris_a(0xBB); //0b10111011 MSB ... LSB
  output_low(PIN_A0);
}
```

Η έξοδος PIN_A0 χρησιμοποιείται για να κρατάει τον CC2530 σε κατάσταση reset μέχρι την ολοκλήρωση της ρύθμισης του PIC όπως ορίζει το πρότυπο ρύθμισης ZNP στην ενότητα 4.5.3.4.

Η συνάρτηση init_router() που εκτελείται αμέσως μετά επικοινωνεί με τον CC2530 μέσω UART. Τα πλαίσια για τη ρύθμιση του router είναι με τη σειρά τα ακόλουθα:

1. SYS_OSAL_NV_WRITE_START_UP_OPTION
2. SYS_RESET
3. SYS_OSAL_NV_WRITE_R
4. APP_CNF_BDB_SET_CHANNEL_secontary
5. APP_CNF_BDB_SET_CHANNEL_primary
6. APP_CNF_BDB_START_COMMISSIONING_steering
7. ZB_APP_REGISTER_REQUEST

Για τη μετάδοση τους στο καλώδιο χρησιμοποιήθηκε η in-build συνάρτηση του compiler putc(). Για παράδειγμα:

```
putc(APP_CNF_BDB_SET_CHANNEL_primary[i]);
```

Η λογική της συνάρτησης init_router() είναι να γραφεί με τη σειρά τα πλαίσια και ενδιάμεσα να περιμένει την σωστή ανταπόκριση από τον CC2530, διαφορετικά να ξανά προσπαθεί. Στην αρχή λοιπόν όταν ο CC2530 απελευθερωθεί από το reset θα στείλει ένα πλαίσιο που θα ενημερώνει τον ZAP ότι συναίβει reset. Μόλις ο ZAP λάβει το reset indication θα ξεκινήσει την αποστολή του πρώτου πλασίου που είναι το SYS_OSAL_NV_WRITE_START_UP_OPTION και θα περιμένει την ανταπόκριση για να συνεχίσει με το επόμενο.

```
int i;
    error_retry:
    while(idframe != 0){     //wait for reset IND from cc2530. It takes 1 min
        if(idframe==100){
            error();
            goto error_retry;
        }
    }
    //Do something when reset occurred
    //printf("reset occurred");
    //Frame 1 Frame SYS_OSAL_NV_WRITE   START_UP_OPTION
    idframe=10; //reset idframe
    for( i=0; i<=9; i++){
        putc(SYS_OSAL_NV_WRITE_START_UP_OPTION[i]);
    }
    //=============================================================================
```

Η idframe είναι μια μεταβλητή που χρησιμοποιήθηκε για την αναγνώριση των πλαισίων που στέλνονται από τον CC2530 μέσω της συνάρτησης recognizeMsg().

```
//0 idframe
const unsigned int cmd0_SRSP_SYS_RESET = 0x41;
const unsigned int cmd1_SRSP_SYS_RESET = 0x80;
//1 idframe
const unsigned int cmd0_SRSP_SYS_NV_OSAL_WRITE = 0x61;
const unsigned int cmd1_SRSP_SYS_NV_OSAL_WRITE = 0x09;
```

Η συνάρτηση recognizeMsg() υλοποιεί τη λειτουργία switch case ελέγχοντας το command field του εισερχόμενου πλαισίου.

```
void recognizeMsg(void){

switch (MsgRe[2]){

    case cmd0_SRSP_SYS_RESET:
        if(MsgRe[3] == cmd1_SRSP_SYS_RESET){
        idframe=0;   }else{idframe=100;}
    break;

    case cmd0_SRSP_SYS_NV_OSAL_WRITE:
        if(MsgRe[3] == cmd1_SRSP_SYS_NV_OSAL_WRITE){
        idframe=1;   }else{idframe=100;}
    break;
```

Η συνάρτηση init_router() συνεχίζει έως ότου ολοκληρωθεί η αποστολή όλων των πλαισίων και στη συνέχεια επιστρέφει στη main().

Ανάγνωση εισερχόμενων πλαισίων στον ZAP

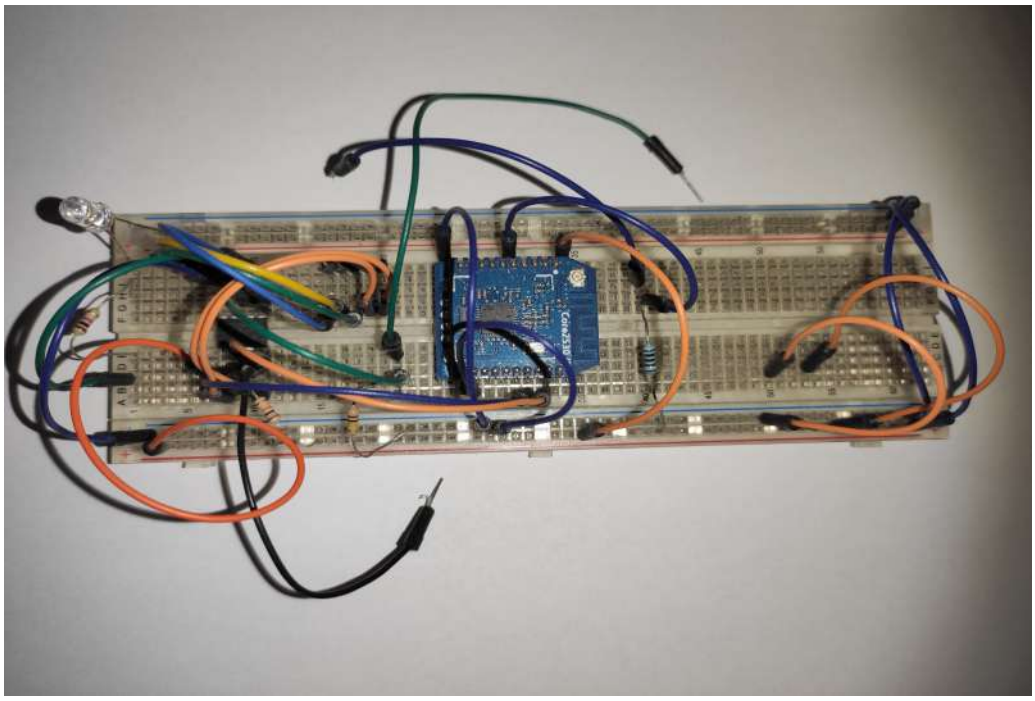
Για να διαβάζει ο ZAP τα πλαίσια που στέλνει ο CC2530 χρησιμοποιήθηκε η μέθοδος του interrupt ώστε να βγαίνουμε από τον βρόγχο της main όποτε υπάρχουν δεδομένα προς ανάγνωση. Όποτε υπάρχουν διαθέσιμα δεδομένα στον buffer του UART το πρόγραμμα διακλαδίζεται στη συνάρτηση που ακολουθεί την εντολή #INT_RDA (RS232 receive data available). Η συνάρτηση αυτή είναι η RDA_isr() που χρησιμοποιεί την in-built συνάρτηση getc() για να διαβάζει κάθε byte του πλαισίου.

```
MsgRe[0] = getc();
if(MsgRe[0] == SOF){//First byte must be 0xFE
MsgRe[1] = getc();  //len
MsgRe[2] = getc();  //cmd0
MsgRe[3] = getc();  //cmd1
```

Το πρώτο byte που φτάνει πάντα στον buffer είναι το 0xFE. Εάν δεν είναι αυτό το πλαίσιο απορρίπτεται. Το επόμενο byte είναι το length, το μήκος των δεδομένων που διαφοροποιείται ανάλογα με τον τύπο του πλαισίου. Γνωρίζοντας το μέγεθος των δεδομένων μπορούμε να διαβάσουμε το General frame.

```
for(int i=4; i<=MsgRe[1]+3; i++){
    MsgRe[i] = getc();
}
MsgRe[MsgRe[1]+4] = getc();  //FCS
```

Έχοντας ολοκληρώσει τον κώδικα του ZAP για τον router μπορούμε να κάνουμε compile, να προγραμματίσουμε τον PIC και να κάνουμε τις δοκιμές.



Εικόνα 4.24 Υλοποίηση του router στο ράστερ

4.5.6 Κώδικας για τον ZAP end-device

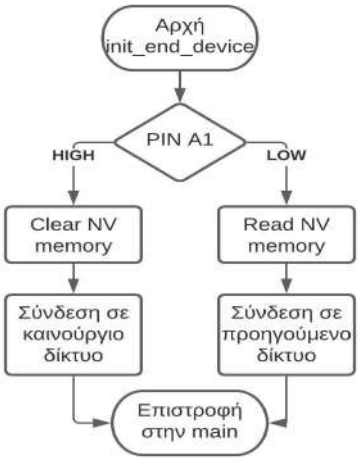
Οι συσκευές end devices θα αποτελούν τους ασύρματους αισθητήρες του συναγερμού. Θα επικοινωνούν είτε απευθείας με τον κεντρικό πίνακα είτε μέσω του router. Θα χρησιμοποιούν τον ίδιο μικροελεγκτή με τον router, τον PIC12F1840. Ο σχεδιασμός του κώδικα ZAP δε διαφέρει πολύ από τον router. Χρησιμοποιούν τις ίδιες συναρτήσεις init(), init_end_device(), RDA_isr() με τη διαφορά ότι μετά την εκτέλεση της main ο PIC θα εκτελεί την εφαρμογή του συνεγερμού. Θα πρέπει δηλαδή να ρυθμιστεί μια είσοδος που θα ελέγχεται για την κατάσταση του αισθητήρα και στη συνέχεια θα στέλνεται κατάλληλο μήνυμα στον Coordinator(διεύθυνση 16bit 0x00). Δε θα πρέπει επίσης να

ξεχάσουμε τον start option να το αλλάξουμε έτσι ώστε να μη χάνει τις ρυθμίσεις του δικτύου με την αλλαγή μπαταριών αλλά να μπορούμε οταν θέλουμε να μεταβούμε σε άλλο δίκτυο.

Έτσι λοιπόν στην συνάρτηση init_end_device() θα πρέπει να σταλούν τα εξής πλαίσια:

1. SYS_OSAL_NV_WRITE_START_UP_OPTION
2. SYS_RESET
3. SYS_GPIO_direction
4. SYS_OSAL_NV_WRITE_E
5. APP_CNF_BDB_SET_CHANNEL_secontary
6. APP_CNF_BDB_SET_CHANNEL_primary
7. SYS_OSAL_NV_WRITE_ZDO
8. APP_CNF_BDB_START_COMMISSIONING_steering
9. ZB_APP_REGISTER_REQUEST

Στην αρχή της συνάρτησης init_end_device() υπάρχει ένας έλεγχος του ακροδέκτη A1 του PIC. Με αυτόν τον τρόπο ο χρήστης μπορεί να αποφασίσει κατα την τροφοδοσία της συσκευής εάν ο αισθητήρας θα συνδεθεί στο δίκτυο που ήταν πριν το reset ή θα αναζητήσει καινούργιο δίκτυο. Είναι ένα κουμπί το οποίο όταν ο χρήστης το έχει πατημένο κατα την εφαρμογή τροφοδοσίας ο αισθητήρας διαγραφεί από τη μνήμη τα στοιχεία του δικτύου που βρισκόταν, διαφορετικά συνεχίζει.



Εικόνα 4.25 Μπλοκ διάγραμμα για την επιλογή δικτύου

Επιπλέον, στο τέλος της συνάρτησης init_end_device() γίνεται χρήση ενός GPIO pin του CC2530 μέσω UART διότι δεν υπάρχουν άλλα διαθέσιμα ελεύθερα pins στον PIC. Αυτό το GPIO χρησιμοποιείται με ένα led σαν ένδειξη στον χρήστη ότι η διαδικασία σύνδεσης ολοκληρώθηκε επιτυχώς.

```
for( i=0; i<=6; i++){
    putc(SYS_GPIO_led_off[i]);
}
```

Για την εφαρμογή του αισθητήρα θα πρέπει να ελέγχεται η κατάσταση του αισθητήρα και να στέλνετε κατάλληλο μήνυμα στον coordinator. Ο έλεγχος του αισθητήρα γίνεται και πάλι με τη μέθοδο του interrupt. Αυτή τη φορά η ρουτίνα interrupt εκτελείται με την αλλαγά κατάστασης στον ακροδέκτη A2 όπου είναι συνδεδεμένος ο αισθητήρας.

```
#INT_RA
void RA_isr(void){
        disable_interrupts(INT_RA2);
        int i;

        if(input(PIN_A2))
        {//kleisto magnitiko
            for(i=0; i<=21; i++){
                putc(ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_close[i]);
            }
        }
        else
        {//anoixto magnitiko
            for(i=0; i<=21; i++){
                putc(ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_open[i]);
            }
        }
        enable_interrupts(INT_RA2);
        clear_interrupt(INT_RA2);    //********************
}
```

Εντολές Cluster στο ZNP

Οι αισθητήρες για να ενημερώσουν τον κεντρικό πίνακα για την κατάσταση του αισθητήρα παράγουν ένα μήνυμα επιπέδου εφαρμογής και το στέλνουν με προορισμός τη διεύθυνση του coordinatator την 0x00. Για την αποστολή μηνύματος μέσω του επιπέδου Aplication layer της Zstack ο ZAP του αισθητήρα χρησιμοποιεί την εντολή ZB_SEND_DATA_REQUEST:

SREQ:

Byte: 1	1	1	2	2	1
Length = 0x08-0x5C	Cmd0 = 0x26	Cmd1 = 0x03	Destination	CommandId	Handle

1	1	1	0-84
Ack	Radius	Len	Data

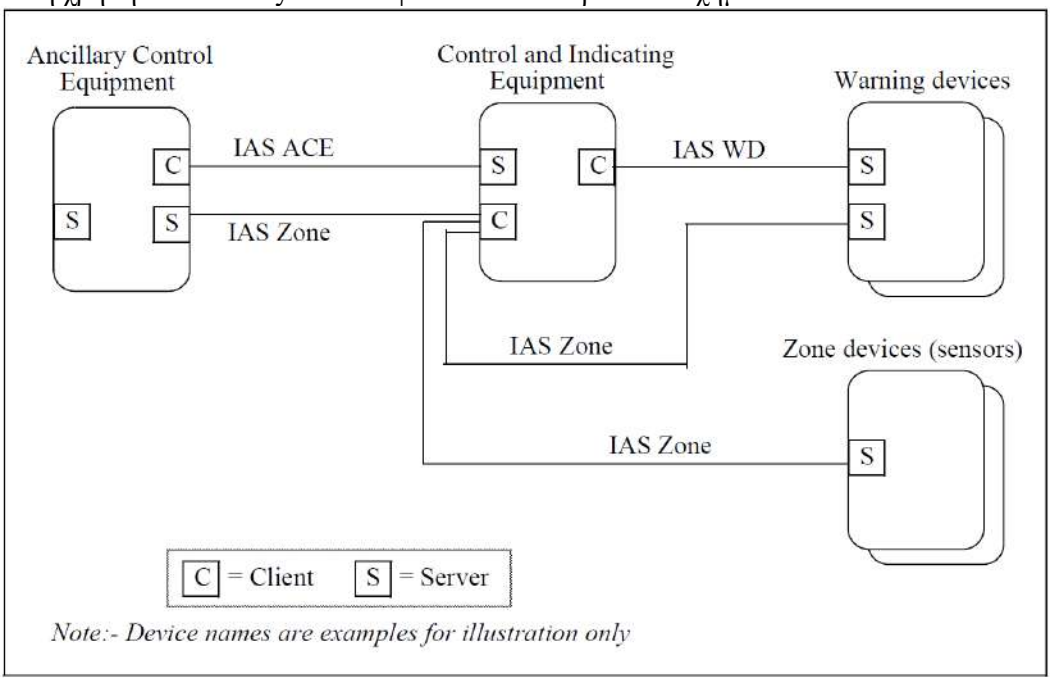
Με τη μέθοδο του ZNP οι εντολές Cluster δε σχηματίζονται στη στοίβα Zstack που τρέχει ο CC2530. Αυτό επιτυγχάνεται από τον ZAP. Συγκεκριμένα για να χρησιμοποιήσουμε εντολές cluster που θα είναι συμφωνα με τα πρότυπα που έχει ορίσει η Zigbee Aliace, οι εντολές ενσωματώνονται στο πεδίο data της εντολής ZB_SEND_DATA_REQUEST.

Στο επίσημο έγγραφο για το πρότυπο Cluster[6] υπάρχει η κατηγορία για τις εφαρμογές σχετικά με το security και safety, Clusters που χρησιμοποιούνται από τα ασύρματα συστήματα zigbee Intruder Alarm Systems (IAS). Τα συστήματα IAS περιλαμβάνουν συναρτήσεις για την ανίχνευση παραβίασης, ειδοποίησης και επεξεργασίας πληροφοριών. Η λίστα με τα clusters που έχουν οριστεί φαίνεται παρακάτω:

Cluster Name	Description
IAS Zone	Attributes and commands for IAS security zone devices.
IAS ACE	Attributes and commands for IAS Ancillary Control Equipment.
IAS WD	Attributes and commands for IAS Warning Devices

Εικόνα 4.26 Λίστα cluster που αφορούν τους τομείς security και safety

Μια τυπική χρήση των security clusters φαίνεται στο παρακάτω σχήμα:



Εικόνα 4.27 Παράδειγμα χρήσης των clusters security και safety

Στον συναγερμό μας θα υλοποιήσουμε τους αισθητήρες και το Control and Indicating Equipment(Coordinator). Όπως φαίνεται παραπάνω οι αισθητήρες υλοποιούν την πλευρά του server του IAS zone Cluster και ο coordinator την πλευρά του cllient.

Τα attributes του Cluster IAS zone οργανώνονται ως εξής:

Attribute Set Identifier	Description
0x000	Zone information
0x001	Zone settings
0x002 – 0xfff	Reserved

Εικόνα 4.28 Attribute Sets για τον Cluster IAS Zone

Η συλλογή Zone Information attribute περιλαμβάνει τα εξής:

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x0000	ZoneState	8-bit Enumeration	All	Read only	0x00	M
0x0001	ZoneType	16-bit Enumeration	All	Read only	-	M
0x0002	ZoneStatus	16-bit bitmap	All	Read only	0x00	M

Εικόνα 4.29 Attributes για το Zone Information Attribute Set

Το ZoneState attribute περιλαμβάνει πληροφορίες όπως είναι η κατάσταση εγγραφής του αισθητήρα στον συναγερμό.

ZoneState Value	Meaning
0x00	Not enrolled
0x01	Enrolled (the server will react to Zone State Change Notification commands from the client)
0x02-0xff	Reserved

Εικόνα 4.30 Τιμές ZoneState attribute

Το ZoneType attribute περιλαμβάνει πληροφορίες όπως είναι η κατηγορία στην οποία ανήκει ο αισθητήρας.

ZoneType attribute value	Zone Type	Alarm1	Alarm2
0x0000	Standard CIE	System Alarm	-
0x000d	Motion sensor	Intrusion indication	Presence indication
0x0015	Contact switch	1st portal Open-Close	2nd portal Open-Close
0x0028	Fire sensor	Fire indication	-
0x002a	Water sensor	Water overflow indication	-
0x002b	Gas sensor	CO indication	Cooking indication
0x002c	Personal emergency device	Fall / Concussion	Emergency button
0x002d	Vibration / Movement sensor	Movement indication	Vibration
0x010f	Remote Control	Panic	Emergency
0x0115	Key fob	Panic	Emergency
0x021d	Keypad	Panic	Emergency
0x0225	Standard Warning Device (see [B2] part 4)	-	-
Other values < 0x7fff	Reserved	-	-
0x8000-0xfffe	Reserved for manufacturer specific types	-	-
0xffff	Invalid Zone Type	-	-

Εικόνα 4.31 Τιμές ZoneType attribute

Το ZoneStatus attribute είναι ένα bit-map που μας ενδιαφέρει περισσότερο στον αισθητήρα, είναι αυτό δηλαδή που έχει να κάνει με την κατάσταση του αισθητήρα.

ZoneStatus Attribute Bit Number	Meaning	Values
0	Alarm1	1 – opened or alarmed 0 – closed or not alarmed
1	Alarm2	1 – opened or alarmed 0 – closed or not alarmed
2	Tamper	1 – Tampered 0 – Not tampered
3	Battery	1 – Low battery 0 – Battery OK
4	Supervision reports (Note 1)	1 – Reports 0 – Does not report
5	Restore reports (Note 2)	1 – Reports restore 0 – Does not report restore
6	Trouble	1 – Trouble/Failure 0 – OK
7	AC (mains)	1 – AC/Mains fault 0 – AC/Mains OK
8-15	Reserved	-

Εικόνα 4.32 Τιμές ZoneStatus attribute

Για να υλοποιηθεί πλήρως η λειτουργία του client και server θα πρέπει με την ολοκλήρωση της σύνδεσης του αισθητήρα στο δίκτυο ο coordinator να ρυθμίσει τον αισθητήρα για το πότε θα στέλνει την αναφορά του(response) με το κατάλληλο attribute και να γραφτούν οι πληροφορίες σε όλα τα παραπάνω attributes. Κάτι τέτοιο όμως απαιτεί πολύ χρόνο και μεγάλη έκταση πηγαίου κώδικα στον ZAP τόσο σε πλευρά αισθητήρα όσο και σε πλευρά coordinator. Γι αυτό τον λόγο αποφασίστηκε στην κατασκευή του συναγερμού, με τη σύνδεση του αισθητήρα στο δίκτυο να μην χρειάζεται κάποια ρύθμιση. Αντιθέτως, ο αισθητήρας στέλνει αυτομάτως ένα read attribute response προς τον coordinator. Αυτό το read attribute response περιλαμβάνει το bit-map του ZoneStatus με την κατάσταση του αισθητήρα. Το response στέλνεται κάθε φορά που η κατάσταση του αισθητήρα αλλάζει. Ο coordinator από την πλευρά του λαμβάνει ένα μήνυμα indication επιπέδου εφαρμογής και αποθηκεύει τοπικά την κατάσταση κάθε αισθητήρα.

Κάθε read attribute response, προτού ενσωματωθεί στο πεδίο data της εντολής ZB_SEND_DATA_REQUEST, θα πρέπει να έχει μια συγκεκριμένη δομή:

Octets: Variable	**Variable**	**Variable**	**…**	**Variable**
ZCL header	Read attribute status record 1	Read attribute status record 2	…	Read attribute status record *n*

Εικόνα 4.33 Η δομή ενός Read Attributes Response

Κάθε read attribute status record θα πρέπει να έχει την μορφή:

Octets: 2	**1**	**0 / 1**	**0 / Variable**
Attribute identifier	Status	Attribute data type	Attribute data

Εικόνα 4.34 Η δομή ενός Read Attributes Status Record Field

ZCL Header Fields:

Παίρνει την τιμή 0 για οποιαδήποτε εντολή Cluster που περιλαμβάνεται στο ZCL και την τιμή 1 για manufacturer specific attributes.

Attribute Identifier Field:

Η τιμή του είναι 16bits μήκους και περιέχει τον identifier της τιμής attribute που διαβάστηκε.

Status Field:

Είναι τιμή μήκους 8bits και δείχνει αν η ανάγνωση του attribute ήταν επιτυχής, διαφορετικά επιστρέφει έναν κωδικό σφάλματος.

Attribute Data Type Field:

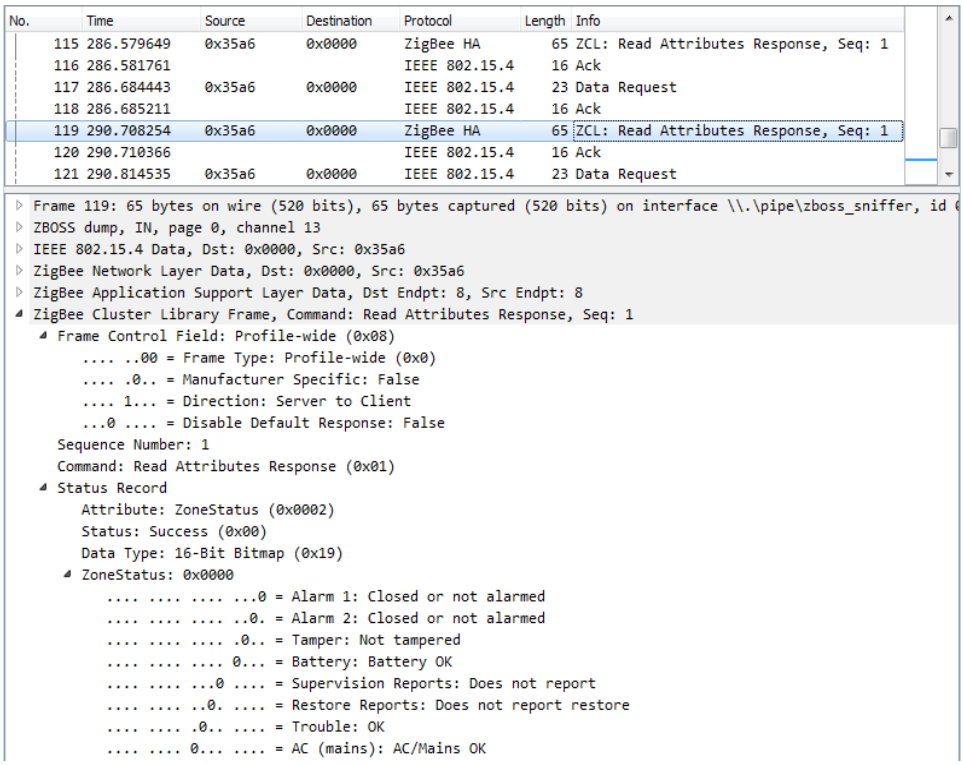
Περιλαμβάνει τον τύπο της Attribute που διαβάστηκε.

Attribute Data Field:

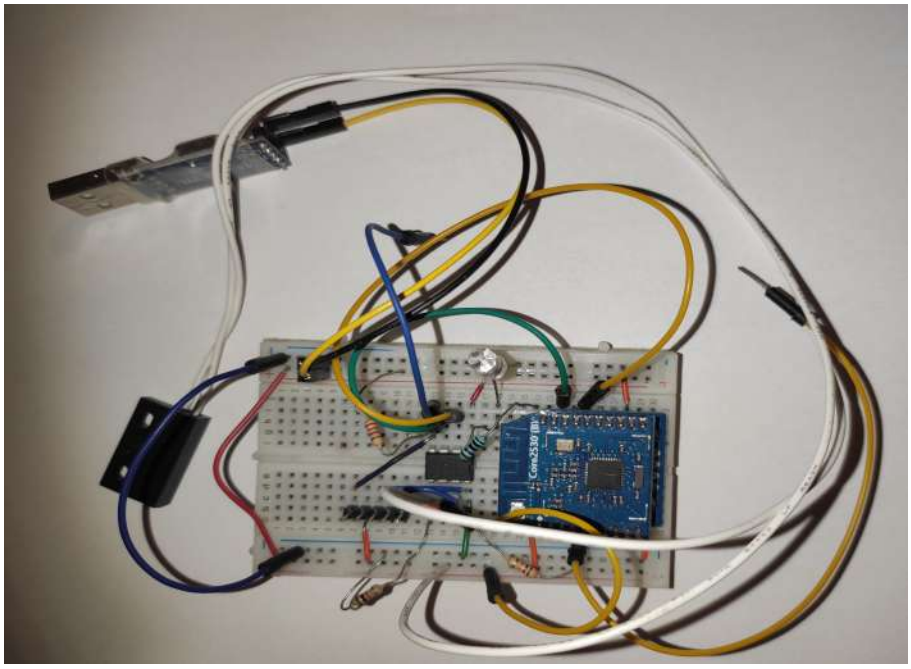
Έχει την τωρινή τιμή της Attribute

Με αυτά υπόψη μπορούμε να καθορίσουμε τα read attribute responses που θα στέλνει ο αισθητήρας με την εντολή ZB_SEND_DATA_REQUEST:

```
unsigned int ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_close[22] = {0xFE,0x11,0x26,0x03,0x00,0x00,0x00,0x05,
0x00,0x00,0x01,0x09,0x08,0x01,0x01,0x02,0x00,0x00,0x19,0x00,0x00,0x2A};  //Send read RSP with ZoneStatus attribute(closed) to coordinator
unsigned int ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_open[22] = {0xFE,0x11,0x26,0x03,0x00,0x00,0x00,0x05,
0x00,0x00,0x01,0x09,0x08,0x01,0x01,0x02,0x00,0x00,0x19,0x01,0x00,0x2B};  //Send read RSP with ZoneStatus attribute(opened) to coordinator
```



Εικόνα 4.35 Read Attribute Response με το ZoneStatus



Εικόνα 4.36 Υλοποίηση end device στο ράστερ

4.5.7 Σχεδιασμός κεντρικού πίνακα

Τα βασικά υλικά του κεντρικού πίνακα είναι το ESP32 και το Module CC2530. Η χρήση του PIC12F1840 είναι περριτή εφόσον το ESP32 εκτός των άλλων λειτουργιών μπορεί μέσω μιας θύρας UART να αναλάβει τον ρόλο του ZAP και να ρυθμίσει το CC2530 ως coordinator.

Ένας βασικός ρόλος του κεντρικού πίνακα είναι η εφαρμογή του συναγερμού. Θα πρέπει να σχεδιαστεί κατάλληλος πηγαίος κώδικας που θα κάνει τις απαραίτητες ενέργειες σύμφωνα με τα πακέτα zigbee που ανταλλάσσονται στο δίκτυο και τις εντολές που έχουν δοθεί από τον χρήστη.

Εκτός από τη λειτουργία του ως coordinator και την εφαρμογή του συναγερμού, ο κεντρικός πίνακας παρέχει ένα user interface για την εύκολη χρήση του. Αυτό επιτυγχάνεται με τα κατάλληλα περιφερειακά όπως είναι το keypad, η οθόνη lcd, ακουστικές ειδοποιήσεις(buzzer), κατάλληλο πρόγραμμα μενού και διεπαφή με κινητό τηλέφωνο.

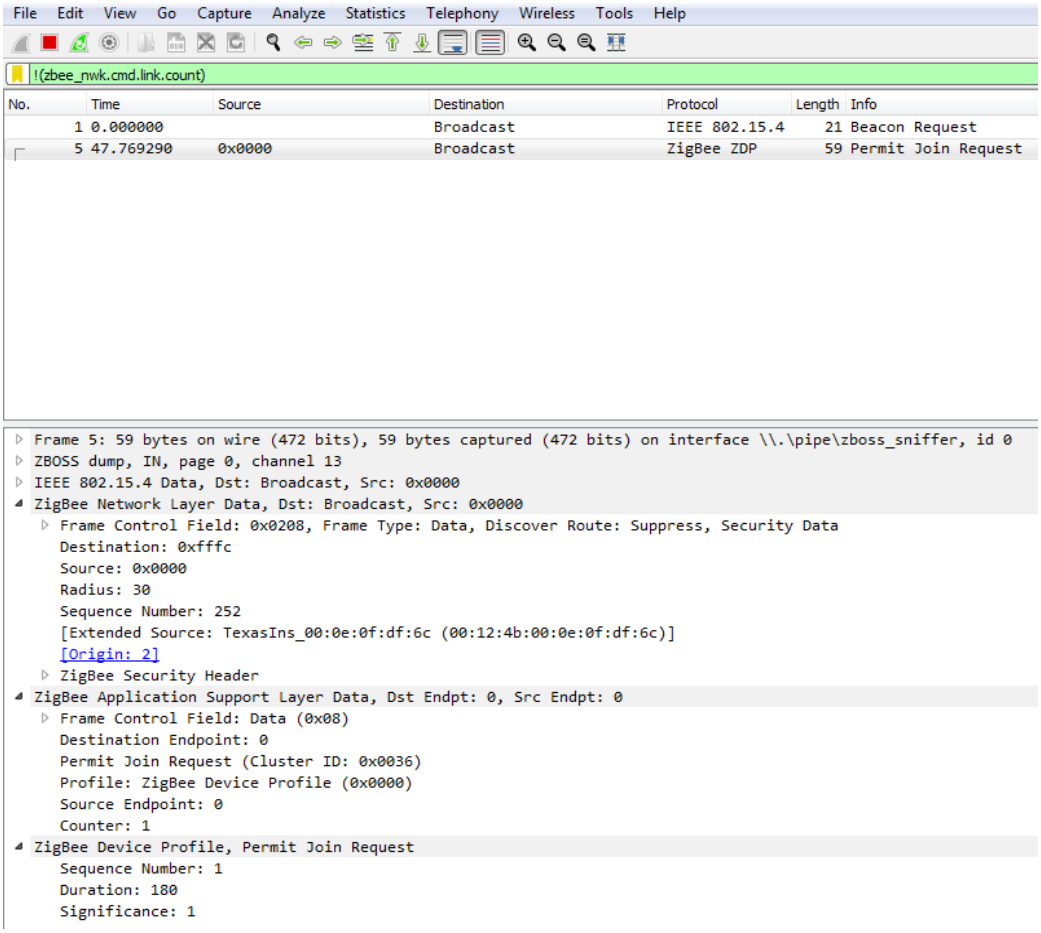
Τέλος, για την κατάλληλη επικοινωνία του συναγερμού με το κινητό, το οποίο θα βρίσκεται είτε εντός είτε εκτός σπιτιού, το ESP32 θα πρέπει να διατηρεί έναν server που θα παρέχει τις υπηρεσίες του μέσω διαδικτύου με ένα κατάλληλο πρωτόκολλο επικοινωνίας.

4.5.7.1 Ο κεντρικός πίνακας στον ρόλο του coordinator

Η ρύθμιση του coordinator δε διαφέρει πολύ από αυτή του router και του end-device. Η βασική διαφορά είναι πως η ανταλλαγή πακέτων σειριακής επικοινωνίας γίνεται μεταξύ CC2530 και ESP32 αντί του PIC12F1840. Ο προγραμματισμός του ESP32 έγινε με τα εργαλεία Visual studio και το πακέτο platform IO.

Οι βασικές συναρτήσεις που χρησιμοποιήθηκαν σε router και end-device όπως είναι οι init_router(), RDA(), recognizeMsg() παραμένουν ίδιες, ωστόσο το framework που χρησιμοποιήθηκε για το ESP32 δε δίνει άμεσα και αξιόπιστα τη δυνατότητα για interrupt από εισερχόμενα πακέτα UART. Έτσι η συνάρτηση RDA() χρησιμοποιεί τη μέθοδο polling κάθε φορά που εκτελείται η συνάρτηση loop(). Αυτό δεν μας δημιουργεί πρόβλημα στον buffer του ESP32 διότι η επικοινωνία CC2530-ESP32 δεν είναι συχνή και η συνάρτηση loop() πρόκειται να είναι γρήγορη και απλή.

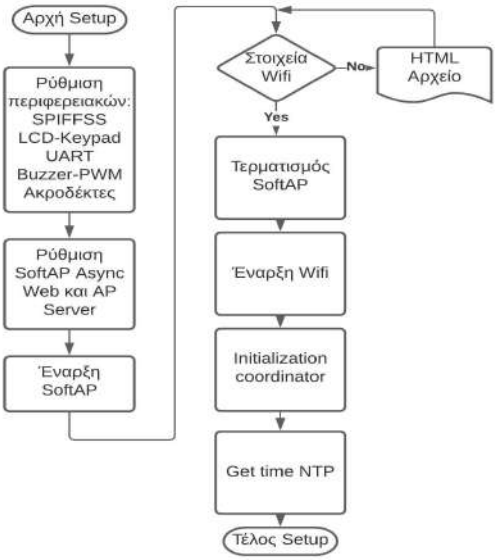
Αφού ολοκληρωθεί η δημιουργία του δικτύου θα πρέπει να δίνεται ένας τρόπος να συνδεθούν οι επιθυμητές συσκευές. Για αυτή τη λειτουργία επιλέχθηκε η εντολή ZDO_MGMT_PERMIT_JOIN_REQ. Ενώ το δίκτυο θα είναι κλειδωμένο για οποιαδήποτε συσκευή, μέσω αυτής της εντολής θα δίνεται ένα χρονικό περιθώριο στο οποίο θα μπορούν να συνδεθούν οι αισθητήρες. Αυτή η εντολή θα δημιουργείται στον coordinator και θα στέλνεται προς όλους τους routers. Ο χρήστης κάθε φορά που θα θέλει να συνδέσει έναν καινούργιο αισθητήρα θα πρέπει να ενεργοποιήσει τον αισθητήρα και να πατήσει ένα κουμπί στον κεντρικό πίνακα που θα εκτελεί την παραπάνω εντολή. Αυτό το κουμπί θα εκτελείται με τη μέθοδο του interrupt σε έναν ακροδέκτη του ESP32.



Εικόνα 4.37 Permit join request

4.5.7.2 Η συνάρτηση Setup

Πρόκειται για την συνάρτηση που θα εκτελεστεί μια φορά στην αρχή του προγράμματος του ESP32 για τις πρώτες απαραίτητες ρυθμίσεις.



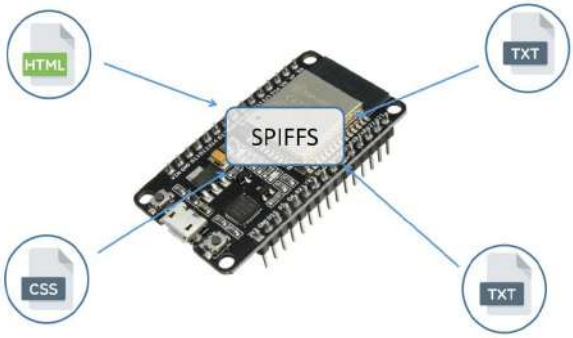
Εικόνα 4.38 Μπλοκ διάγραμμα Setup

Αρχικά γίνεται η ρύθμιση των περιφερειακών, της σειριακής επικοινωνίας, του συστήματος SPIFFS και των ακροδεκτών του ESP32. Για τα περιφερειακά χρησιμοποιούμε μια lcd 2x16 με πρωτόκολλο I2C για εξοικονόμηση ακροδεκτών, ένα keypad 4x5 τύπου matrix και ένα buzzer. Για τον γρήγορο προγραμματισμό τους χρησιμοποιούμε τις έτοιμες βιβλιοθήκες Keypad.h και LiquidCrystal_I2C.h έχοντας υπόψη μόνο τον τρόπο που υλοποιούνται οι συναρτήσεις για τη λειτουργία τους έτσι ώστε να μην δημιουργήσουν πρόβλημα στο υπόλοιπο πρόγραμμα.

Για τη ρύθμιση των σειριακών διεπαφών χρειάζεται να ορίσουμε το baud rate. Το ESP32 διαθέτει δύο διαύλους UART, ένα για την επικοινωνία με το CC2530 και ο δεύτερος χρησιμοποιήθηκε για την διαδικασία του debugging.

Το buzzer για να ακουστεί θα εφαρμόζεται σήμα pwm στον ακροδέκτη 18.

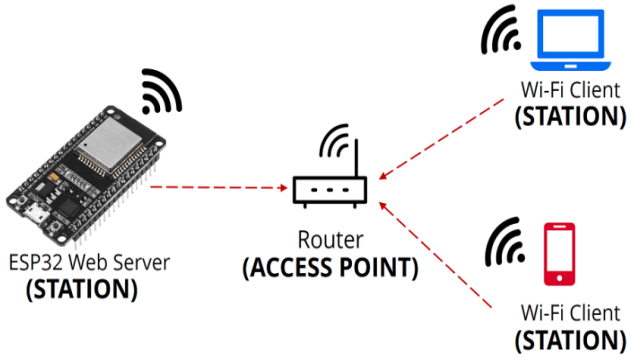
Το ESP32 περιλαμβάνει ένα σύστημα για την προσπέλαση της μνήμης τύπου flash που είναι συνδεδεμένη με δίαυλο SPI, Serial Peripheral Interface Flash File System(SPIFFS). Αυτό το σύστημα μας επιτρέπει να αποθηκεύουμε ολόκληρα αρχεία στο ESP32 και να τα χρησιμοποιούμε όποτε είναι αναγκαίο αντί να έχουμε γραμμένα δεδομένα διάσπαρτα στον πηγαίο κώδικα. Τα αρχεία που θα αποθηκεύσουμε θα είναι τύπου txt, HTML και CCS. Αυτά τα αρχεία είναι απαραίτητα για τον asyncronous server που περιγράφεται παρακάτω.[25]



Εικόνα 4.39 Σύστημα SPIFFS για την αποθήκευση αρχείων στο ESP32

Το ESP32 έχει τη δυνατότητα να λειτουργεί με δύο διαφορετικούς τρόπους όσον αφορά το wifi, σαν station και σαν SoftAP. Η λειτουργία του station είναι η ποιο συνηθισμένη, να συνδέουμε δηλαδή το esp32 σε ένα υπάρχον δίκτυο wifi και να λειτουργεί σαν μια οποιαδήποτε άλλη συσκευή παρέχοντας τις υπηρεσίες του, όπως για παράδειγμα έναν web server. Από την άλλη πλευρά όμως μπορεί να δημιουργήσει το δικό του δίκτυο wifi λειτουργώντας σαν Access point επιτρέποντας σε άλλες συσκευές να συνδεθούν σε αυτό.

Εικόνα 4.40 ESP32 σε λειτουργία access point



Εικόνα 4.41 ESP32 σε λειτουργία wifi station(web server)

Ο συναγερμός κατα την εγκατάσταση στο σπίτι θα πρέπει να βρει και να συνδεθεί στο κατάλληλο δίκτυο wifi. Για να εισάγουμε τα στοιχεία του επιθυμητού wifi υπάρχουν τρεις τρόποι, κατα την διάρκεια του compile, μέσω διεπαφής χρήστη με πληκτρολόγιο και οθόνης και μέσω κινητού. Επιλέχθηκε ο τρίτος τρόπος γιατί είναι φιλικός προς τον χρήστη και είναι μια μέθοδος που χρησιμοποιείται συχνά σε συσκευές iot και οι χρήστες έχουν τη σχετική εμπειρία.

Για να γίνει η σύνδεση ESP32-κινητού το ESP32 θα πρέπει αρχικά να λειτουργήσει σαν Access point ώστε το κινητό να συνδεθεί και να στείλει τα στοιχεία για το επιθυμητό wifi. Για να σταλούν τα στοιχεία χρησιμοποιείται και πάλι asynchronous server που θα περιγραφεί αργότερα.

Εικόνα 4.42 Εισαγωγή στοιχείων wifi στο ESP32 Access point

Στη συνέχεια απενεργοποιείται το access point και το esp32 συνδέεται στο wifi. Η διαδικασία του setup ολοκληρώνεται με την ρύθμιση του coordinator και του δικτύου zigbee. Σημαντική παράμετρος σε έναν συναγερμό είναι η ώρα έτσι ώστε να γνωρίζουμε την ακριβή ώρα ενός συμβάν συναγερμού. Γι αυτό τον λόγο χρησιμοποιείται ένας NTP Client(Network Time Protocol) για την ενημέρωση της ώρας από το ίντερνετ.

4.5.7.3 Η συνάρτηση loop

Είναι η συνάρτηση που θα εκτελείται συνεχώς. Κάνει συνεχώς ελέγχους μέσω των οποίων καλούνται οι κατάλληλες συναρτήσεις για την υλοποίηση της εφαρμογής του συναγερμού. Οι λειτουργίες που υλοποιεί η loop είναι η ανάγνωση των πλαισίων από τον CC2530, ο έλεγχος για σύνδεση καινούργιας συσκευής στο δίκτυο zigbee, η ενημέρωση της κατάστασης του κάθε αισθητήρα όποτε προκύπτει trigger σε αισθητήρα, η ενημέρωση σχετικού μηνύματος στην οθόνη lcd και ο έλεγχος του keypad για την εισαγωγή στο ανάλογο μενού.

Έλεγχος για σύνδεση καινούργιας συσκευής

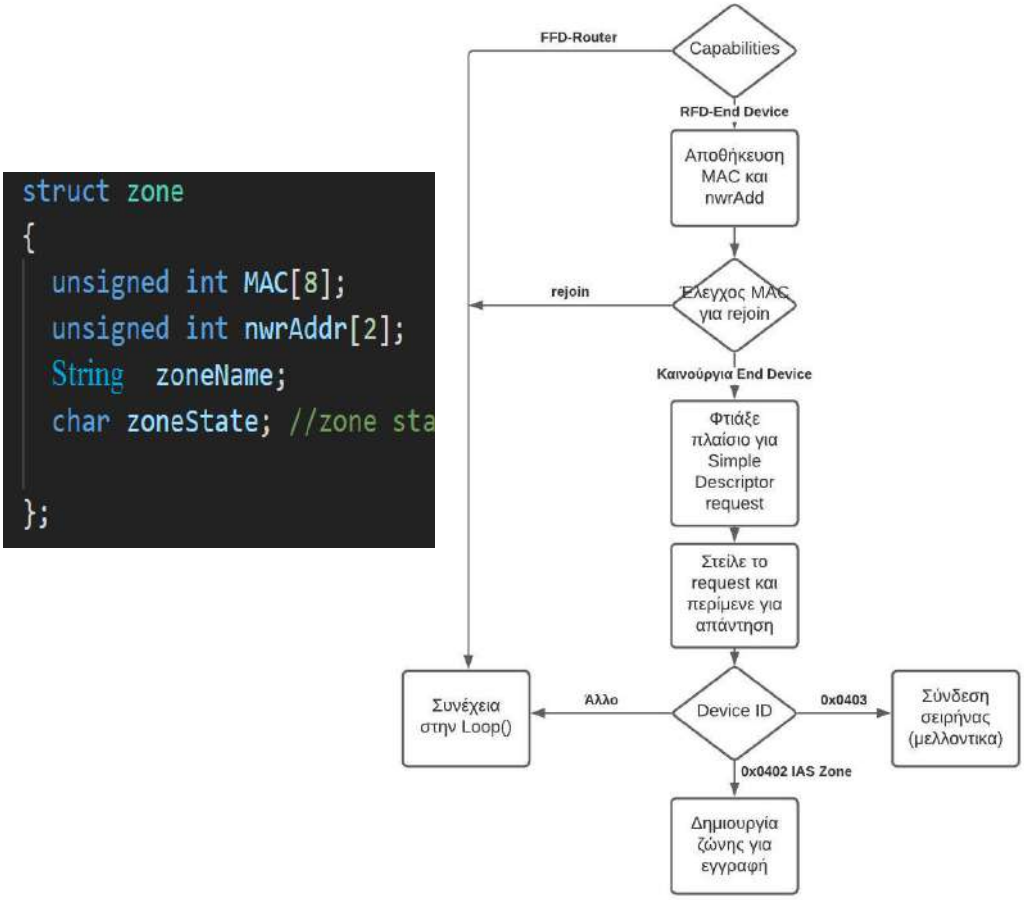
Ο έλεγχος για σύνδεση καινούργιας συσκευής γίνεται έτσι ώστε να ξεχωρίσουμε το είδος κάθε συσκευής στο δίκτυο(αισθητήρα, router, ασύρματη σειρήνα κτλ) και να δημιουργήσουμε τις σωστές ζώνες αποφεύγοντας τη δημιουργία καινούργιων ζωνών στο ενδεχόμενο του rejoin κάποιας συσκευής. Για την υλοποίηση αυτής της λειτουργίας εκμεταλλευόμαστε το device announcment που στέλνει κάθε συσκευή broadcast σε κάθε σύνδεση στο δίκτυο. Αυτό το πακέτο zigbee περιλαμβάνει τη διεύθυνση της συσκευής και τις δυνατότητες που περιλαμβάνει η συσκευή. Για να ξεχωρίσουμε τους αισθητήρες απο τους router μπορούμε να ελέγξουμε εάν η συσκευή είναι RFD(Reduce Function Device) ή FFD(Full Function Device). Στην περίπτωση του RFD, δηλαδή μια συσκευή end device, προχωράμε σε περαιτέρω έλεγχο. Αποθηκεύουμε προσωρινά τη διεύθυνση MAC και nwrAddr και με βάση αυτών μπορούμε να ελέγξουμε εάν η συσκευή έχει συνδεθεί ξανά στο δίκτυο. Στο ενδεχόμενο της καινούργιας συσκευής θα πρέπει να δημιουργηθεί και να σταλεί κατάλληλο πακέτο για τη λεπτομερέστερη διερεύνηση της συσκευής. Πρόκειται για το πακέτο Simple Descriptor που θα μας δώσει την δυνατότητα να μάθουμε για το είδος της συσκευής. Το πακέτο Simple Descriptor ακολουθεί την δομή που περιγράφηκε στην ενότητα 4.5.3.5, όπου στις διευθύνσεις mac και nwraddr τοποθετούνται οι διευθύνσεις που αποθηκεύτηκαν προηγουμένως ενώ για τον υπολογισμό του checksum χρησιμοποιήθηκε η συνάρτηση checksum_xor(unsigned int[], int). Η συνάρτηση checksum_xor δέχεται το πακέτο που πρέπει να υπολογιστεί το checksum και την παράμετρο idframe για την αναγνώριση του είδους του πακέτου που θα χρησιμοποιηθεί.

```
void checksum_xor(unsigned int array[], int id){
unsigned int c; //checksum
//ypologismos checksum tou array
    c = array[1] ^ array[2];
  for (int i=3; i<=array[1]+3; i++){
    c = c ^ array[i];
    }

//apo8ikeusi tou checksum sto katallilo frame
switch (id)
{
case 8:
  ZDO_SIMPLE_DESC_REQ[9] = c;
  break;

case 9:
  ZB_SEND_DATA_REQUEST_Identify[17] = c;
  break;

default:idframe=10;
  break;
}
idframe=10;
}
```

```
void createzones(unsigned int mac[], unsigned int nwr[]){
  //kaleitai otan ginete announcment apo end device sensor
  cntnumzone++;  //auksi8ikan oi zones ana mia
  statezones[cntnumzone]=1; //state of current zone joined -->
  int currZone;
  currZone=cntnumzone;


 // zone unenrolled[currZone] = {}; //create new empty object
String num=String(currZone);  //metatropi ari8mou zonis se Stri

  String name= String("unenrolled_" + num);
  unenrolled[currZone].zoneName=name;

  for (int i = 0; i <= 7; i++)
  {
    unenrolled[currZone].MAC[i]= mac[7-i];
  }
   for (int i = 0; i <= 1; i++)
  {
    unenrolled[currZone].nwrAddr[i]= nwr[i];
  }

  unenrolled[currZone].zoneState = '-'; //unused in unenrolled
```

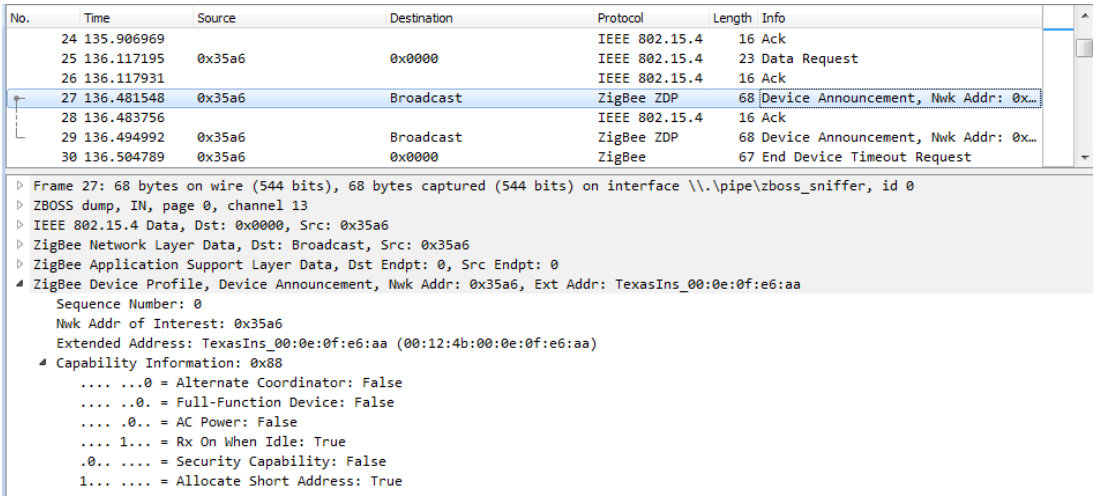
Αφού συνδεθεί το πακέτο simple descritor και σταλεί στη συνέχεια περιμένουμε για την απάντηση από την οποία θα ξεχωρίσουμε το Device ID. Το Device ID παίρνει τιμές ανάλογα με το είδος της συσκευής στον εκάστοτε cluster(στην περίπτωση μας τον cluster IAS Zone με cluster id 0x0500). Οι αισθητήρες που συνδέουμε είναι τύπου IAS Zone με Device ID 0x0402, ενώ μελλοντικά μπορούν να προστεθούν συσκευές όπως IAS Warning Devices με id 0x0403 η IAS Ancillary Control Equipment με id 0x0401. Εάν η συσκευή είναι τύπου IAS Zone τότε θα κληθεί η συνάρτηση createzones(MAC, nwrAdd) όπου θα δημιουργηθεί η ζώνη με τα κατάλληλα χαρακτηριστικά.

Αφού εντοπιστεί καινούργια συσκευή τύπου IAS Zone καλείται η συνάρτηση ceatezones(MAC, nwrAdd) ώστε να γίνει η δημιουργία της ζώνης στον συναγερμό. Για τη δημιουργία μιας ζώνης χρησιμοποιείται μια δομή struct τύπου zone όπου ορίζεται στο αρχείο Globals.h. Η δομή τύπου zone μας δίνει την ευκολία να αναφερόμαστε στα χαρακτηριστικά οποιασδήποτε ζώνης. Στην δομή περιλαμβάνονται τα χαρακτηριστικά όπως είναι οι διευθύνσεις MAC,nwrAdd, το όνομα που θα δοθεί από τον χρήστη κατα τη διάρκεια της εγγραφής και η κατάσταση του αισθητήρα, ανοιχτός ή κλειστός.

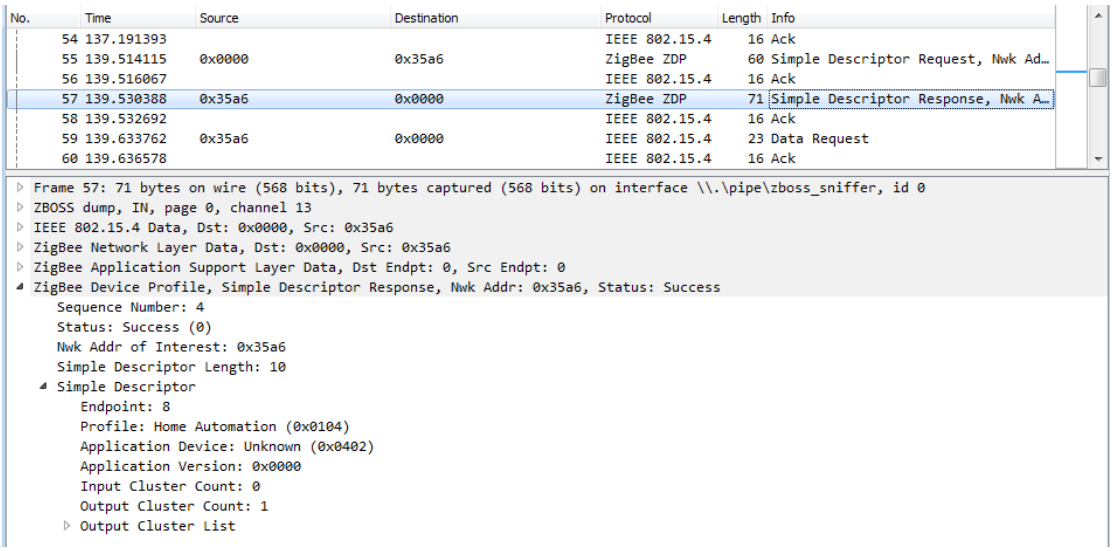


Εικόνα 4.43 Μποκ διάγραμμα Device announcment και η δομή zone

Η συνάρτηση ceatezones(MAC, nwrAdd) αποθηκεύει τις διευθύνσεις, ένα προσωρινό όνομα, τον χαρακτήρα ΄-΄ για την κατάσταση του αισθητήρα και ανανεώνει μια μεταβλητή που εποπτεύει τον αριθμό των ζωνών.



Εικόνα 4.44 Device announcment



Εικόνα 4.45 Simple Descriptor Response

Ενημέρωση της κατάστασης του κάθε αισθητήρα

Η συνάρτηση Loop περιλαμβάνει επίσης τη συνάρτηση IndicationReadRsp(). Η συνάρτηση IndicationReadRsp() καλείται κάθε φορά που στέλνεται ένα μήνυμα τύπου Attribute Read Response απο έναν αισθητήρα στον κεντρικό πίνακα προκειμένου να ανανεωθεί η κατάσταση ενός αισθητήρα. Αρχικά ελέγχει εάν το μήνυμα που λήφθηκε είναι τύπου Attribute Read Response και στην συνέχεια ελέγχει όλες τις εγγεγραμμένες ζώνες μέσω της διεύθυνσης nwrAdd για τον εντοπισμό της ζώνης που έστειλε το μήνυμα. Έπειτα απομονώνει το byte του μηνύματος που περιλαμβάνει το ZoneStatus που αναφέρθηκε στην ενότητα 4.5.6 ώστε να αποθηκεύσει την κατάσταση του αισθητήρα στη μεταβλητή zoneState η οποία παίρνει τις τιμές 'C'(για κλειστό αισθητήρα) και ΄Ο΄(για ανοιχτό αισθητήρα).

```
void IndicationReadRsp(void){
//INDICATION READ RSP ZONESTATUS
if((data_avalable == true)&&(idframe == 5)&&(MsgRe[7] == 0x05)&&(MaxEnrolled>0)){
  idframe = 10;   //reset idframe
  data_avalable = false;

unsigned int nwrAdd[2];
  nwrAdd[0] = MsgRe[5];
  nwrAdd[1] = MsgRe[4];

  for (int i = 1; i <= MaxEnrolled; i++)
  {
    if ((nwrAdd[0]==enrolled[i].nwrAddr[0])&&(nwrAdd[1]==enrolled[i].nwrAddr[1]))
    {
        switch (MsgRe[17])
        {
        case 0x00:
          enrolled[i].zoneState = 'C';
          Serial.print(enrolled[i].zoneName);
          Serial.print(enrolled[i].zoneState);
          break;

        case 0x01:
          enrolled[i].zoneState = 'O';
          Serial.print(enrolled[i].zoneName);
          Serial.print(enrolled[i].zoneState);
          break;

        default:enrolled[i].zoneState = '-';
          break;
        }
    }
  }
 }
}
```

Εικόνα 4.46 Μπλοκ διάγραμμα και κώδικας της IndicationReadRsp()

Ενημέρωση οθόνης lcd και εισαγωγή στο μενού

Η συνάρτηση Loop ολοκληρώνεται με τις συναρτήσεις printLocalTime, updateOpenZones και τον έλεγχο του keypad για εισαγωγή σε κάποιο απο τα διαθέσιμα μενού. Η συνάρτηση printLocalTime είναι υπεύθυνη για την ανανέωση της ώρας στην οθόνη lcd ενώ η συνάρτηση updateOpenZones εμφανίζει στην οθόνη lcd τις ζώνες που είναι ανοιχτές σε πραγματικό χρόνο.

```
void updateOpenZones(void){
int i;
int refresh=0;

  for (i = 0; i <= MaxEnrolled; i++)
  {
    if (enrolled[i].zoneState=='O'){
      if(statenrolledzones[i] == 0){refresh=1;}
      statenrolledzones[i] = 1;
    }
    if (enrolled[i].zoneState=='C'){
      if(statenrolledzones[i] == 1){refresh=1;}
      statenrolledzones[i] = 0;
    }
  }

  if (refresh == 1)
  {
    LcdAlwaysOn();
  }
}
```

```
void LcdAlwaysOn(void){
int i;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(myhour);lcd.print(":");lcd.print(mymin);
  lcd.setCursor(0,1);
  lcd.print("Zones:");
  for (i = 1; i <= MaxEnrolled; i++)
  {
    if(statenrolledzones[i] == 1){
      lcd.print(enrolled[i].zoneName);
    }
  }
}
```

4.5.7.4 Το μενού του κεντρικού πίνακα

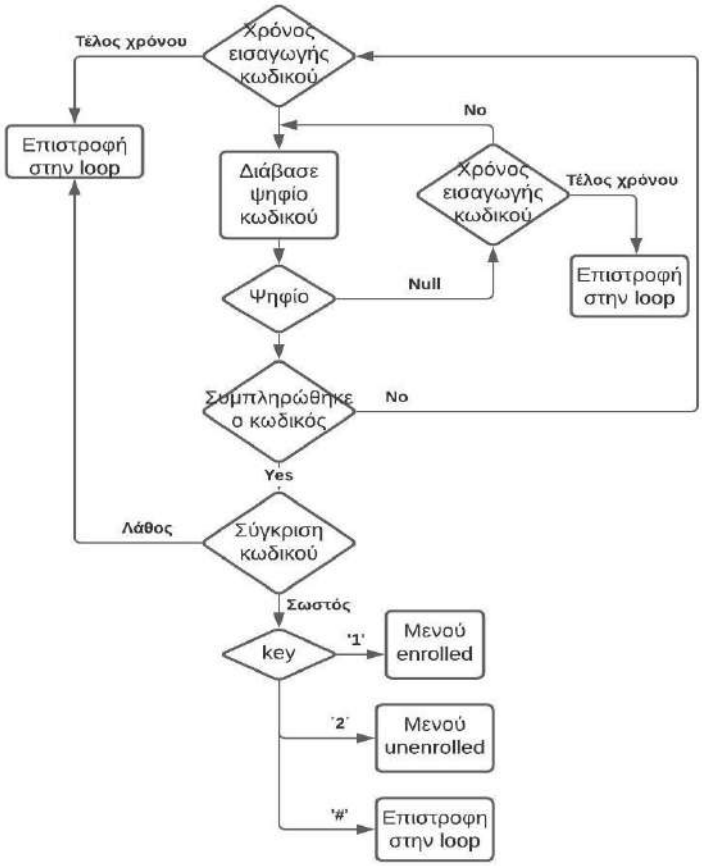
Οι διαθέσιμες καρτέλες μενού είναι το installer μενού και το μενού του οπλισμού. Για την εισαγωγή σε κάποιο μενού χρησιμοποιείται ο έλεγχος Switch case στην συνάρτηση loop.

```
key = keypad.getKey();
switch (key)
{
case '*'://ENTER ISTALLER MENU
  installerMenu();
  break;

case 'D'://ENTER ARM MENU
  arm();
  break;

default:
  break;
}
```

Installer μενού

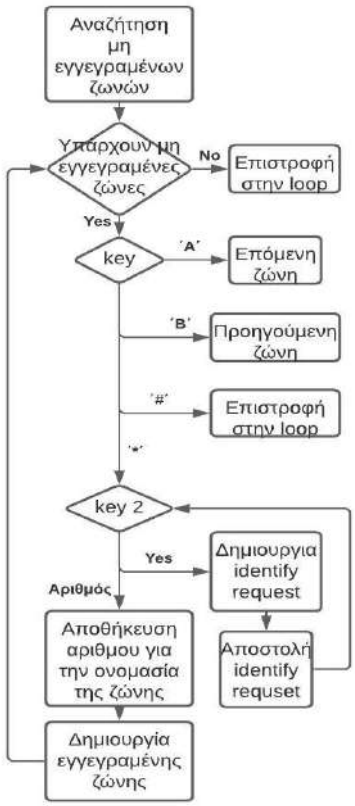
Κατα την εγκατάσταση του συναγερμού για να δοθούν ονομασίες στις ζώνες και να είναι έτοιμες προς χρήση ο εγκαταστάτης θα πρέπει να εισέλθει στο installerMenu μέσω της συνάρτησης loop πατώντας το πλήκτρο ΄*΄ στο keypad. Απο εκεί έχει πρόσβαση στις ζώνες που έχουν συνδεθεί στο δίκτυο zigbee και είναι ακόμα σε κατάσταση unenrolled ή να ελέγξει ποιες ζώνες βρίσκονται σε κατάσταση enrolled. Επιλέγοντας μια ζώνη μπορούμε να απευθυνθούμε σε αυτή στέλνοντας οποιοδήποτε zigbee πακέτο επιθυμούμε.

Για την αποφυγή εγκατάστασης ανεπιθύμητων αισθητήρων στον συναγερμό από αγνώστους, τα installer μενού στους συναγερμούς έχουν έναν κωδικό. Έτσι πριν αποκτήσει πρόσβαση ο χρήστης στις ζώνες προς εγκατάσταση εισάγει τον κωδικό installer εντός ενός χρονικού ορίου.



Εικόνα 4.47 Μπλοκ διάγραμμα installer μενού

Εάν ο χρήστης επιλέξει το μενού unenrolled θα μπορεί να περιηγηθεί στις ζώνες προς εγγραφή. Από εκεί μπορεί να επιλέξει οποιαδήποτε ζώνη και να στείλει ένα πακέτο zigbee ή να δώσει το επιθυμητό όνομα ζώνης για να ολοκληρωθεί η εγγραφή. Προς το παρόν στην υλοποίηση του συναγερμού υπάρχει η δυνατότητα για το πακέτο zigbee identify request. Το identify request είναι μια πολύ χρήσιμη εντολή cluster που μπορεί να απευθύνει μια συσκευή zigbee σε μια άλλη συσκευή του δικτύου. Λαμβάνοντας ένα identify request μια συσκευή θα πρέπει να προσδιορίσει τον εαυτό της με έναν τρόπο όπως είναι η ενεργοποίηση ενός buzzer ή ενός led. Η διάρκεια του identification καθορίζεται από το attribute IdentifyTime. Έτσι ο χρήστης μπορεί να γνωρίζει σε ποιά συσκευή απευθύνεται ανα πάσα στιγμή. Αφού εντοπίσει τις επιθυμητές ζώνες και τις προσδώσει το κατάλληλο όνομα οι ζώνες παύουν να υπάρχουν στο μενού unenrolled και εμφανίζονται στην καρτέλα enrolled. Σε μελλοντική αναβάθμιση του συναγερμού ο χρήστης θα μπορεί να ανταλλάσσει παραπάνω πακέτα zigbee με τις ζώνες έτσι ώστε να υλοποιούνται όλα τα attributes στον cluster IAS Zone και ορισμένες άλλες χρήσιμες εντολές όπως η εντολή leave network για τη διαγραφή μιας ζώνης.

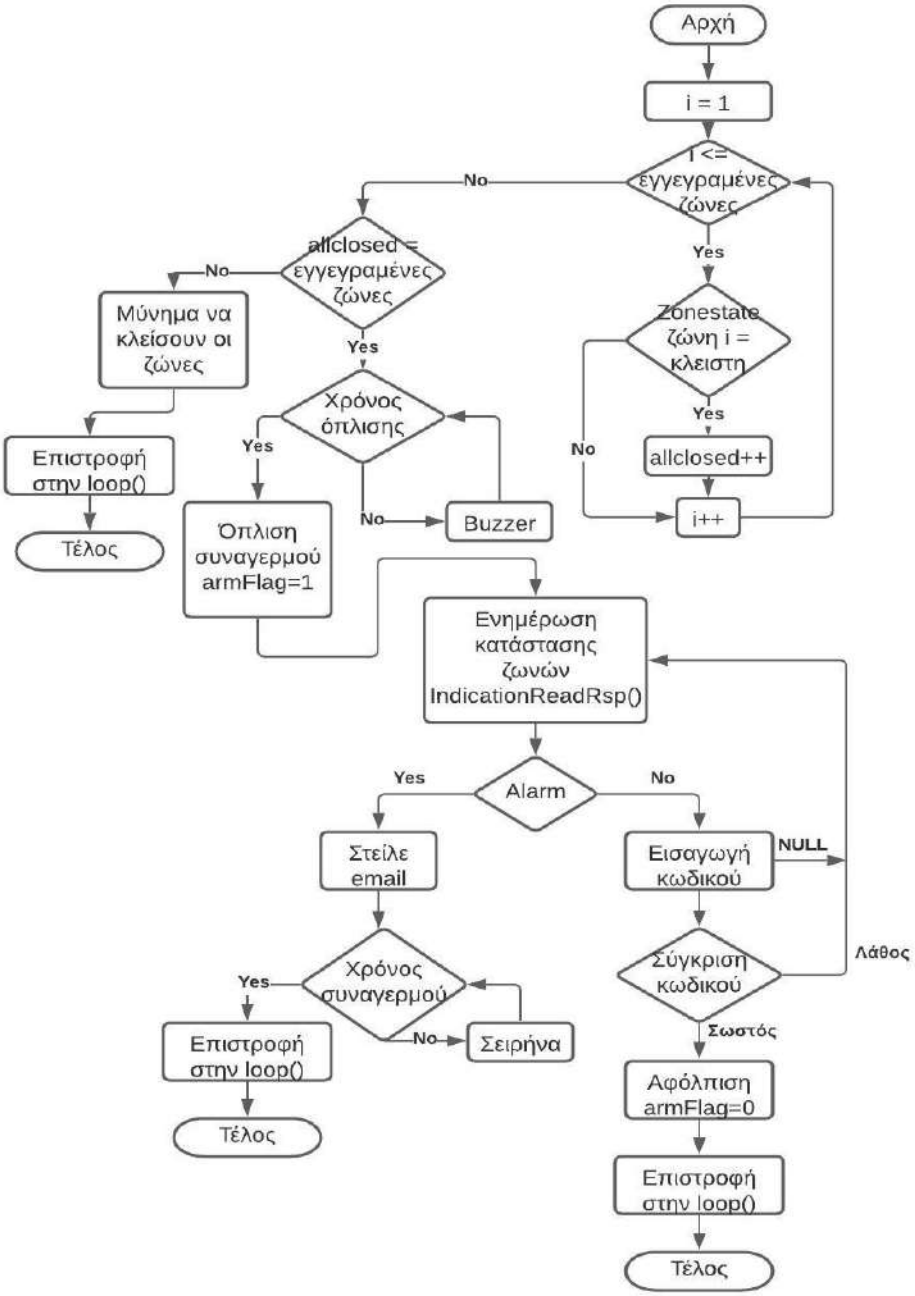


Εικόνα 4.48 Μλοκ διάγραμμα απο την περιήγηση στο installer μενού

Μενού οπλισμού

Ο οπλισμός του συναγερμού γίνεται χωρίς κωδικό, με το πλήκτρο ΄D΄ του keypad. Με την όπλιση του συναγερμού το πρόγραμμα διακλαδίζεται από τη συνάρτηση loop στην συνάρτηση arm όπου αρχίζει η διαδικασία της όπλισης. Αρχικά γίνεται έλεγχος στις ζώνες ώστε να είναι όλες κλειστές. Αυτό σημαίνει οτι ο χρήστης θα πρέπει να κλείσει όλες τις πόρτες και τα παράθυρα προτού αποχωρήσει από το σπίτι, διαφορετικά ο συναγερμός δεν θα οπλίσει. Αφού αρχίσει η όπλιση δίνεται ένας χρόνος ώστε να αποχωρήσει ο χρήστης από το σπίτι. Αυτός ο χρόνος είναι παραμετροποιήσιμος και συνοδεύεται με ηχητική ένδειξη ενός buzzer.

Αφού ολοκληρωθεί η όπλιση ο συναγερμός μπαίνει σε έναν βρόγχο όπου ο κεντρικός πίνακας εξετάζει εάν έχει ληφθεί κάποιο μήνυμα από αισθητήρα μέσω της συνάρτησης IndicationReadRsp που αναλύθηκε ποιο πάνω ή εάν έχει εισάγει ο χρήστης σωστά τον κωδικό αφόπλισης ώστε να επιστρέψει στην συνάρτηση loop.



Εικόνα 4.49 Μπλοκ διάγραμμα μενού οπλισμού

4.5.7.5 ESP32 Gateway

Το ESP32 εκτός της διαχείρισης του δικτύου zigbee έχει τη δυνατότητα σύνδεσης με το ίντερνετ ώστε ο χρήστης να έχει πρόσβαση στα χαρακτηριστικά του συναγερμού όπου και αν βρίσκεται. Κατα τη διαδικασία εγκατάστασης του συναγερμού επιλέγουμε το δίκτυο στο οποίο θα συνδεθεί ο συναγερμός και στην συνέχεια το esp32 μας διαθέτει ένα interface με το πρωτόκολλο http ώστε να μπορούμε να ενημερωνόμαστε και να ρυθμίζουμε παραμέτρους του συναγερμού.

Το http είναι πρωτόκολλο που βασίζεται στη λειτουργία των requests και responses απο clients και servers αντίστοιχα. Το esp32 θα πρέπει να ανταποκρίνεται σε http requests στέλνοντας το κατάλληλο html αρχείο τόσο κατα την ρύθμιση του esp32 για την σύνδεση στο router, όσο και για τη λειτουργία του συναγερμού. Αυτά τα html αρχεία θα τα δημιουργήσουμε και θα τα αποθηκεύσουμε στη μνήμη flash του esp32 με το σύστημα SPIFFS όπως αναφέρθηκε νωρίτερα.

Σύνδεση με το wifi

Αρχικά για τη ρύθμιση του esp32 ώστε να συνδεθεί στο κατάλληλο δίκτυο wifi, θα πρέπει να λειτουργεί σαν access point στο οποίο θα συνδεθεί μια συσκευή που θα εισάγει το ssid και password του δικτύου wifi. Τα στοιχεία σύνδεσης στο access point αναγράφονται στην οθόνη lcd. Με τη σύνδεση της συσκευής, για παράδειγμα ενός κινητού, θα πρέπει μέσω ενός browser να στείλουμε ένα request στη διεύθυνση του esp32 ώστε να λάβουμε το response με το αρχείο html που θα μας επιτρέψει να εισάγουμε τα στοιχεία του wifi.

Το HTML για την εισαγωγή των στοιχείων wifi περιέχει δύο πεδία εισαγωγής μεταβλητών, ένα για το ssid και ένα για το password. Καθε html αρχείο αποτελείται από elements και attributes. Συγκεκριμένα για τα στοιχεία του wifi χρησιμοποιούμε τα elements form και input για τον καθορισμό του html και και την εισαγωγή μεταβλητών αντίστοιχα. Ένα element input μπορεί να έχει διάφορες μορφές, στην περίπτωση μας καθορίζεται ως κείμενο απο το attribute type=”text” με την ονομασία που του δίνει το attribute name=”ssid”. Πατώντας το κουμπί submit η μεταβλητή στέλνεται στο esp32 με την μορφή GET request.

```
<form action="/get">
  Ender SSID: <input type="text" name="ssid">
  <input type="submit" value="Submit">
</form><br>

<form action="/get">
  Enter Pass: <input type="text" name="password">
  <input type="submit" value="Submit">
</form><br>
```

Για τον χειρισμό των requests το esp32 χρησιμοποιεί τη βιβλιοθήκη ESPAsyncWebServer ώστε να τα διαχειρίζεται ασύγχρονα. Οπότε φτάνει ένα request σε κάποιο συγκεκριμένο url στο esp32, αυτό ανταποκρίνεται στέλνοντας πίσω στον client το κατάλληλο αρχείο. Η βιβλιοθήκη ESPAsyncWebServer μας επιτρέπει να ρυθμίσουμε τον server να ακούει για requests σε συγκεκριμένες διαδρομές εκτελώντας τις κατάλληλες συναρτήσεις ως ανταπόκριση. Για αυτό τον σκοπό χρησιμοποιείται η μέθοδος on() στο αντικείμενο server. Για παράδειγμα, εάν μεταβούμε στην αρχική διεύθυνση “/” το αρχείο που θα μας επιστραφεί είναι το indexAP.html από τη flash μνήμη του esp32

```
serverAP.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    //request->send(200, "text/plain", "Hello World");
  request->send(SPIFFS, "/indexAP.html", String(), false);
});
```

Όταν εισάγουμε το όνομα ssid και πατάμε submit το request που δημιουργείται είναι της μορφής /get?input=ssid. Οπότε πρέπει να ορίσουμε τις ενέργειες του server στα requests με διεύθυνση “/get”. Θέλουμε ο server να αποθηκεύει τις τιμές του ssid και password. Αναζητάει στο request την ύπαρξη κάποιας τιμής και αν υπάρχει την αποθηκεύει στη μεταβλητή inputMessage που στη συνέχεια θα χρησιμοποιηθεί για τη σύνδεση στο wifi.

```
serverAP.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
  String inputMessage;
  String inputParam;
   if (request->hasParam(PARAM_INPUT_3)) {
   inputMessage = request->getParam(PARAM_INPUT_3)->value();
   inputParam = PARAM_INPUT_3;
   ssid = inputMessage;
   setSsid = 1;
 }
   if (request->hasParam(PARAM_INPUT_4)) {
   inputMessage = request->getParam(PARAM_INPUT_4)->value();
   inputParam = PARAM_INPUT_4;
   password = inputMessage;
   setPass = 1;
 }
 request->send(SPIFFS, "/indexAP.html", String(), false);
```

Χειρισμός του συναγερμού μέσω internet

Εκτός από τη διεπαφή μέσω keypad και lcd, ο συναγερμός δίνει τη δυνατότητα χειρισμού από κινητή συσκευή. Οι δυνατότητες που δίνονται είναι να ελέγχουμε την κατάσταση του συναγερμού, εάν είναι όλες οι ζώνες κλειστές, εάν είναι οπλισμένος, να ρυθμίζουμε παραμέτρους όπως ο χρόνος όπλισης και ο χρόνος συναγερμού και η όπλιση-αφόπλιση του συστήματος.

Έχοντας εισάγει τα στοιχεία του wifi απο την προηγούμενη ενότητα το esp32 παύει να λειτουργεί ως access point και συνδέεται στο internet προσφέροντας τον web server που διατηρεί στις συσκευές client. Όπως και στη ρύθμιση του wifi το esp32 έχει αποθηκευμένο ένα αρχείο html στην flash μνήμη με τα απαραίτητα στοιχεία. Με βάση τα παραπάνω, τα στοιχεία του html θα πρέπει να είναι δύο τρόποι για είσοδο τιμής για τις μεταβλητές του χρόνου όπλισης και συναγερμού, ένα κουμπί για την όπλιση-αφόπλιση και δύο placeholders για την κατάσταση των ζωνών και του συναγερμού.

Η εισαγωγή των χρόνων όπλισης και συναγερμού ακολουθεί την ίδια λογική με την εισαγωγή του ssid και password. Τα placeholder τοποθετούνται ανάμεσα σε δύο σύμβολα % στο αρχείο html και η τιμή τους δίνεται από το πρόγραμμα του esp32 ανάλογα την κατάσταση τους. Για την λειτουργία της όπλισης-αφόπλισης, με το πάτημα του κουμπιού εκτελείται ένα script. Αυτό το script διαβάζει την κατάσταση του συναγερμού από το placeholder ώστε να οπλίσει αν ο συναγερμός είναι αφοπλισμένος ή να αφοπλίσει αν ο συναγερμός είναι οπλισμένος. Η όπλιση και αφόπλιση γίνεται με ένα request στις διευθύνσεις “/arm” και “/disarm” αντίστοιχα όπου το esp32 τις διαχειρίζεται αναλόγως.

```
<p>Alarm state: <strong id="state"> %STATE%</strong></p>
<p>zones state: <strong> %ZONES%</strong></p>
<button class="button" onclick="myFunction()">%BUTTON_NAME%</button><br><br>
```

```
  <script>
    function myFunction() {

var x = document.getElementById("state").innerHTML;

if(x.trim() === "DISARM")
{
  document.location = '/arm';
}else{
  document.location = '/disarm';
}
    }
    </script>
```

Εικόνα 4.50 Interface του συναγερμού απο το κινητό

DDNS

Μέχρι στιγμής έχουμε δημιουργήσει έναν web server στο esp32 που παρέχει υπηρεσίες σχετικά με τον συναγερμό στο τοπικό δίκτυο(Wifi). Η πρόσβαση στον συναγερμό μέσω wifi είναι βολική όμως τις περισσότερες φορές χρειαζόμαστε τον χειρισμό του συναγερμού από οποιοδήποτε σημείο βρισκόμαστε. Για την πρόσβαση στις υπηρεσίες του web server από διαφορετικό δίκτυο wifi χρειάζεται να έχουμε μια στατική διεύθυνση ip. Επειδή όμως υπάρχει χρέωση για την απόκτηση της από τον πάροχο επιλέχθηκε η μέθοδος με το ddns(dynamic dns).

DDNS είναι μια υπηρεσία η οποία διατηρεί και «μεταφράζει» τα domain names σε δυναμικές διευθύνσεις IP. Μέσω μιας τέτοιας υπηρεσίας, μπορούμε να έχουμε πρόσβαση στον υπολογιστή ή στα αρχεία μας που βρίσκονται στο σπίτι μας, από οπουδήποτε στον κόσμο. Μας επιτρέπει να έχουμε πρόσβαση στον web server καλώντας ένα συγκεκριμένο domain name αντί για την διεύθυνση ip η οποία αλλάζει σε τακτά χρονικά διαστήματα. Σε αντίθεση με το DNS, το οποίο δουλεύει μόνο με στατικές IP, το DDNS είναι σχεδιασμένο να λειτουργεί με δυναμικές IP, οι οποίες χρησιμοποιούνται στις περισσότερες σπιτικές συνδέσεις internet.

Για να χρησιμοποιήσουμε μια τέτοιου τύπου υπηρεσία, αρχικά θα πρέπει να επιλέξουμε μια, όπως είναι η NO-IP, και στη συνέχεια θα πρέπει να δημιουργήσουμε έναν λογαριασμό. Υστέρα ανάλογα με την υπηρεσία που θα επιλέξουμε, θα υπάρχουν αναλυτικές οδηγίες για το πως μπορούμε να ρυθμίσουμε την υπηρεσία. Συνήθως υπάρχουν software προγράμματα που πρέπει να κατεβάσουμε και να εγκαταστήσουμε στον server. Επίσης, σε μερικές από αυτές τις υπηρεσίες, είναι δυνατή η ρύθμιση απευθείας επάνω στο router, έτσι ώστε να μη χρειάζεται να τρέχουμε συνεχώς το software.

Ο τρόπος με τον οποίο δουλεύει το software είναι απλός. Αφού αρχίσει να εκτελείται, ελέγχει συνεχώς για το αν έχει μεταβληθεί η δημόσια IP της σύνδεσης, και όταν αντιληφθεί ότι άλλαξε, επικοινωνεί με την DDNS υπηρεσία, ενημερώνοντας της και ανανεώνοντας τα records της.

Με αυτόν τον τρόπο, όσο υπάρχει ενεργή σύνδεση, και όσο το πρόγραμμα εκτελείται, είναι σίγουρο ότι όποιος καλεί το domain που έχετε δημιουργήσει, θα δρομολογείται στο σωστό μέρος.

Αποστολή email

Σε κάθε σύγχρονο συναγερμό δεν αρκεί να ενεργοποιούνται διάφορες συσκευές όπως μια σειρήνα κατα τη διάρκεια μιας διάρρηξης, θα πρέπει να υπάρχει ενημέρωση προς τον χρήστη. Ένας τρόπος είναι η αποστολή ενός email.

Η αποστολή ενός email γίνεται με έναν SMTP server. Πρόκειται για ένα πρότυπο αποστολής email. Η βιβλιοθήκη που χρησιμοποιήθηκε γι αυτό τον σκοπό είναι η ESP-Mail-Client. Αρχικά δημιουργούμε ένα account που θα χρησιμοποιεί το esp32 και στη συνέχεια εισάγουμε στον κώδικα τα στοιχεία του smtp server.

```
void sendEmail(void){
    // Set the SMTP Server Email host, port, account and password
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);

  // For library version 1.2.0 and later which STARTTLS protocol was supported,the STARTTLS will be
  // enabled automatically when port 587 was used, or enable it manually using setSTARTTLS function.
  //smtpData.setSTARTTLS(true);

  // Set the sender name and Email
  smtpData.setSender("ESP32", emailSenderAccount);

  // Set Email priority or importance High, Normal, Low or 1 to 5 (1 is highest)
  smtpData.setPriority("High");

  // Set the subject
  smtpData.setSubject(emailSubject);

  // Set the message with HTML format
  smtpData.setMessage("<div style=\"color:#2f4468;\"><h1>Alarm Triggered</h1><p>- Sent from ESP32 board</p></div>", true);
  // Set the email message in text format (raw)
  //smtpData.setMessage("Hello World! - Sent from ESP32 board", false);

  // Add recipients, you can add more than one recipient
  smtpData.addRecipient(emailRecipient);
  //smtpData.addRecipient("YOUR_OTHER_RECIPIENT_EMAIL_ADDRESS@EXAMPLE.com");

  smtpData.setSendCallback(sendCallback);

  //Start sending Email, can be set callback function to track the status
  if (!MailClient.sendMail(smtpData))
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  //Clear all data from Email object to free memory
  smtpData.empty();
}
```

4.6 Συμπεράσματα-μελλοντικές βελτιώσεις

Σε αυτό το κεφάλαιο αναλύθηκε η υλοποίηση ενός ασύρματου συναγερμού Zigbee. Το σύστημα είναι πλήρως λειτουργικό με όλα τα στοιχεία τέθηκαν ως στόχοι στην αρχή του κεφαλαίου. Ωστόσο, υπάρχουν λειτουργίες απαραίτητες για την ανάπτυξη και την ολοκλήρωση του συναγερμού για μελλοντική εξέλιξη.

Ο συναγερμός δε διαθέτει partitions. Αυτό σημαίνει πως όλοι οι αισθητήρες οπλίζουν και αφοπλίζουν μαζί. Κάτι τέτοιο δεν είναι εύχρηστο όταν έχουμε αισθητήρες κίνησης και θέλουμε να κινούμαστε στο εσωτερικό της εγκατάστασης με την περιμετρική ασφάλεια ενεργοποιημένη. Αυτή η δυνατότητα μπορεί εύκολα να προστεθεί χάρης την λειτουργία των groups του Zigbee. Οι συσκευές θα πρέπει να υλοποιούν τον cluster groups και ο χρήστης να ομαδοποιεί τις συσκευές είτε από το μενού installer του πληκτρολογίου είτε από την εφαρμογή του κινητού.

Ένα σημαντικό ελάττωμα που υπάρχει στην υλοποίηση του παραπάνω συναγερμού είναι πως το πληκτρολόγιο βρίσκεται στον κεντρικό πίνακα. Τα πληκτρολόγια τοποθετούνται δίπλα στις εξώπορτες ώστε να γίνεται η όπλιση-αφόπλιση κατα την είσοδο-έξοδο στον χώρο. Αντίθετα, ο κεντρικός πίνακας πρέπει να προστατεύεται σε κρυφό χώρο για να μην εντοπίζεται απο τους κλέφτες ή έστω να εντοπίζεται μετά από κάποιο χρόνο που ο κεντρικός πίνακας θα έχει προλάβει να στείλει τα απαραίτητα σήματα. Μια λύση σε αυτό είναι απλώς η επέκταση καλωδίων που θα απομακρύνουν το πληκτρολόγιο από τον πίνακα. Εναλλακτικά μπορεί να υλοποιηθεί ασύρματο πληκτρολόγιο Zigbee που θα δώσει και τη δυνατότητα προσθήκης περισσότερων πληκτρολογίων.

Ένα παρόμοιο πρόβλημα με το πληκτρολόγιο παρουσιάζεται και με τη σειρήνα η οποία είναι ενσύρματη. Κάτι τέτοιο είναι απολύτως αποδεκτό, παρόλα αυτά μπορεί να υλοποιηθεί ασύρματη σειρήνα με τον cluster On-Off.

Σημαντική προσθήκη στο συναγερμό είναι ένα εφεδρικό σύστημα επικοινωνίας. Ο συναγερμός επικοινωνεί μόνο με wifi. Αυτό σημαίνει ότι σε μια περίπτωση κακής σύνδεσης με τον πάροχο internet ή το κόψιμο καλωδίου του παρόχου από τους κλέφτες, το σύστημα δεν θα μας ειδοποιήσει για την παραβίαση του χώρου. Οι συναγερμοί εκτός απο την επικοινωνία IP διαθέτουν και κάρτες επικοινωνίας GSM και PSTN. Ο συνδυασμός IP-GSM-PSTN εξασφαλίζει την ενημέρωση για παραβίαση σε κάθε περίπτωση.

Βιβλιογραφία

Βιβλία

[1] Drew Gislason, Zigbee Wireless Networking, September 2007
[2] Κ.Ε.Κ. Πυθαγόρας Εγχειρίδιο Για Το Προσωπικό Ιδιωτικής Ασφάλειας

Specifications

[3] ZigBee Specification, December 1 2006
[4] IEEE Std 802.15.4, October 1 2003
[5] Zigbee Home Automation Public Application Profile Specification October 25, 2007
[6] Zigbee Cluster Library Specification October 19 2007
[7] TI CC2530ZNP Interface Specification

Application Note

[8] TI ZTool Application Note AN045

Data Sheets-Manuals

[9] CC2530 Datasheet
[10] Waveshare Core2530 user manual
[11] TI ZigBee Sensor Monitor User Guide
[12] TI Z-Stack 3.0 Developer's Guide
[13] TI Z-Stack 3.0 Sample Application User's Guide
[14] TI Z-Stack Sample Applications
[15] TI Z-Stack User's Guide For CC2530 ZigBee-PRO Network Processor Sample Applications
[16] TI Z-Stack User's Guide - CC2530DB
[17] TI Z-Stack ZCL API
[18] TI Z-Stack Monitor and Test API
[19] TI Z-Stack Commissioning Cluster API
[20] TI OSAL API
[21] TI HAL Driver API
[22] TI 802.15.4 MAC API
[23] XBee/XBee-PRO S2C Zigbee User Manual
[24] IAR IDE User Manual Rev. 1.2

Internet Site

[25] ESP32 Projects https://randomnerdtutorials.com/
[26] ESP-IDF Programming Guide https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
[27] Texas Instrrument Zigbee Forum https://e2e.ti.com/support/wireless-connectivity/zigbee-thread-group/zigbee-and-thread/f/zigbee-thread-forum
[28] Use ZTool and Z-Stack 3.0 ZNP to set up a basic Zigbee 3.0 network https://sunmaysky.blogspot.com/2017/02/use-ztool-z-stack-30-znp-to-set-up.html?m=1
[29] How to handle end node announcment, active endpoint respose and simple descriptor response in Z-Stack for CC2530 https://sunmaysky.blogspot.com/2016/08/how-to-handle-end-node-announcement.html
[30] Grades In Intruder Alarm Systems https://eldesalarms.com/articles/en-50131-grades-in-intruder-alarm-systems/
[31] Ελληνική Αστυνομία Υπουργείο Προστασίας Του Πολίτη http://www.astynomia.gr/index.php?option=ozo_content&perform=view&id=81&Itemid=73&lang

Paper in Conference Proceedings

[32] Creating a ZigBee Smart Energy Device with the MSP430F54xx and the CC2530-ZNP, Zin Kyaw

[33] The Research And Design Of Zigbee Wireless Networking,Jian-Ming Liao,Xue-Qin, Guo-Ming, SI-YU Zhan
[34] Εκπαιδευτικός Οδηγός Εγκαταστάσεων Συστημάτων Ασφάλειας-Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης, Πτυχιακή των Δελήτσικα Δημητρίου, Μυτηλιναίου Φανής, Αατή Σέργιου
[35] Τοποθέτηση Συστήματος Ασφαλείας Στον Χώρο Των Εργαστηριακών Μηχανών Πτυχιακή Εργασία Των Χαράλαμπος Γεωργιλάς και Πέτροσ Ζιώτης

Journal Articles

[36] Ποια πρέπει να είναι τα κριτήρια επιλογής συστήματος ασφάλειας Security Manager 24 Ιουλίου 2015
[37] Ενσύρματα & ασύρματα συστήματα συναγερμού: Ποιοτικά Χαρακτηριστικά και κριτήρια επιλογής Security Manager 20 Ιουλίου 2013

ΠΑΡΑΡΤΗΜΑ Α: Πηγαίος Κώδικας Κεντρικού Πίνακα

```
//////////////////////////////////////////////////////
/////      main.cpp          //////////////////////
//////////////////////////////////////////////////////

#include <Arduino.h>
#include <Globals.h>
#include <command_field.h>
#include <ConstantFrames.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <WiFi.h>
#include "time.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"
#include "ESP32_MailClient.h"

//////////////////dilosi sunartisevn////////////////
void RDA(void); //read data available. Polling
void showMsg(void);
void recognizeMsg(void);
void init_coor(void);
void createzones(unsigned int [], unsigned int []);
void checksum_xor(unsigned int [], int);
void error(void);
void installerMenu(void);
void LcdAlwaysOn(void);
void ShowUnenrolled(void);
void ShowEnrolled(void);
void printLocalTime();
void updateOpenZones(void);
void arm(void);
void IndicationReadRsp(void);
int AlarmIdication(void);
void AlarmMenu(void);
void sendCallback(SendStatus info);// Callback function to get the Email sending status
void sendEmail(void);
///////////////dilosi matavliton//////////////////
boolean led_state = false;
char key;
char webKey;

///////////interrupt routines here//////////////////
void IRAM_ATTR button_permit_isr() {
delay(100);//debounch
      for(int i=0; i<=7; i++){
         Serial2.write(ZB_PERMIT_JOINING_REQUEST[i]);
      }
}
```

```
//////////LCD INITIALAZATION//////////
// set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
// set LCD address, number of columns and rows
// if you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

//////////////////KEYPAD INITIALAZATION//////////////////
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {26, 25, 33, 32};  //temp to 25 8a ginei 25  kalvdio
byte colPins[COLS] = {13, 5, 14, 27}; // temp to 35 8a ginei 12 gri kalvdio
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Create AsyncWebServer object on port 301 for web site and 80 for softAP
AsyncWebServer server(301);
AsyncWebServer serverAP(80);

// The Email Sending data object contains config and data to send
SMTPData smtpData;

///////// PLACEHOLDER FOR WEB SERVER ////////////
// Replaces placeholder with alarm state
String processor(const String& var){
  String alarmState;
  if(var == "STATE"){
    if(armFlag == "ARM"){
      alarmState = "ARM";
    }
    else{
      alarmState = "DISARM";
    }
    //Serial.print(alarmState);
    return alarmState;
  }else if (var == "ZONES")
  {
    if (AlarmIdication())// 1 an yparxei esto mia zoni anoixti
    {
      return "CLOSE_ALL_ZONES";
    }else
    {
      return "ALL_CLOSED";
    }
  }else if (var == "BUTTON_NAME")
  {
    if (armFlag == "ARM")
```

```
    {
      return "DISARM";
    }else
    {
      return "ARM";
    }
  }
  return String();
}

//////////////////SETUP//////////////////////
void setup() {
  ////// SETUP PIN MODE //////////
pinMode(4, OUTPUT);  // reset cc2530
pinMode(23, OUTPUT);  //toogle led
digitalWrite(4, LOW);
digitalWrite(23, LOW);
pinMode(19, INPUT_PULLUP);  //button gia permit join
//////buzzer//////
pinMode(15, OUTPUT);  //buzzer
ledcSetup(ledChannel, freq, resolution);
ledcAttachPin(18, ledChannel);

////////// Serial begin & LCD INIT //////////
Serial.begin(115200);
Serial2.begin(115200);
lcd.init(); // initialize LCD
lcd.backlight();  // turn on LCD backlight

//////// SETUP SOFT AP //////////
 WiFi.softAP(ssidAp.c_str(), passwordAp.c_str());

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Pass:");
  lcd.print(passwordAp);
  lcd.setCursor(0,1);
  lcd.print(WiFi.softAPIP());

  //////// SPIFFS BEGIN //////////
  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

/////////////// Async WEB SERVER //////////////////////
// Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){

    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  // Route to load style.css file
```

```
  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/style.css", "text/css");
  });

  // Route to arm the alarm
  server.on("/arm", HTTP_GET, [](AsyncWebServerRequest *request){
    if (armFlag == "DISARM")
    {
      webKey = 'D';
    }
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  // Route to set GPIO to LOW
  server.on("/disarm", HTTP_GET, [](AsyncWebServerRequest *request){
    if (armFlag == "ARM")
    {
      tempDisarmCode[0] = '1';
      tempDisarmCode[1] = '2';
      tempDisarmCode[2] = '3';
      tempDisarmCode[3] = '4';
    }
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

    // Route to take armtime from web
  server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
    String inputMessage;
    String inputParam;
      if (request->hasParam(PARAM_INPUT_1)) {
      inputMessage = request->getParam(PARAM_INPUT_1)->value();
      inputParam = PARAM_INPUT_1;
      armtime = inputMessage.toInt();
      armtime = armtime*1000; // sec to mlsec
    }
      if (request->hasParam(PARAM_INPUT_2)) {
      inputMessage = request->getParam(PARAM_INPUT_2)->value();
      inputParam = PARAM_INPUT_2;
      alarmtime = inputMessage.toInt();
      alarmtime = alarmtime*1000; // sec to mlsec
    }
    request->send(SPIFFS, "/index.html", String(), false, processor);

  });

////////////////// Async SERVER AP ///////////////////////
  serverAP.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
       //request->send(200, "text/plain", "Hello World");
    request->send(SPIFFS, "/indexAP.html", String(), false);
  });

    // Route to take armtime from web
  serverAP.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
```

```
    String inputMessage;
    String inputParam;
     if (request->hasParam(PARAM_INPUT_3)) {
     inputMessage = request->getParam(PARAM_INPUT_3)->value();
     inputParam = PARAM_INPUT_3;
     ssid = inputMessage;
     setSsid = 1;
    }
     if (request->hasParam(PARAM_INPUT_4)) {
     inputMessage = request->getParam(PARAM_INPUT_4)->value();
     inputParam = PARAM_INPUT_4;
     password = inputMessage;
     setPass = 1;
    }
    request->send(SPIFFS, "/indexAP.html", String(), false);

  });
  serverAP.begin();

 while ((setSsid==0) || (setPass==0))
 {
    Serial.print(setSsid);
 delay(1000);
  if ((setSsid==1) && (setPass==1))
  {
    setSsid=2;
    setPass=2;
      //////Connect to Wi-Fi//////////
  Serial.print("Connecting to ");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Connecting to");
  lcd.setCursor(0,1);
  lcd.print("Wifi...");
  Serial.println(ssid);
  WiFi.begin(ssid.c_str(), password.c_str());
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
 ///////

   ////// Start web server ///////
  server.begin();
  ////// End softAP ////////
  serverAP.end();
  WiFi.softAPdisconnect(true);
 /////////

 /////// INIT COORDINATOR //////////////
 digitalWrite(4, HIGH); //reset cc2530
```

```
Serial.printf("Start");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Starting...");
init_coor();
lcd.clear();
lcd.setCursor(0,0);
attachInterrupt(19, button_permit_isr, FALLING);
Serial.printf("coordinator initialazation complete");

  ///////// CONFIGURE TIME ///////////
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  printLocalTime();

LcdAlwaysOn();
for (int i = 0; i <= 29; i++)//declaration in globals.h
{
  statezones[i]=0;
}

  }

}

}

//////////////////LOOP//////////////////
void loop() {

//delay(1000); //from other code
RDA();

//INDICATION TOOGLE
if((data_avalable == true)&&(idframe == 5)&&(MsgRe[12] == 0x02)){
      //showMsg();
    idframe = 10;   //reset idframe
       //toggle

      switch (led_state)
      {
      case false:
        led_state=true;
        digitalWrite(23, HIGH);
        break;

      case true:
        led_state=false;
        digitalWrite(23, LOW);
        break;
      default:
        break;
      }
```

```
  data_avalable = false;
}

//INDICATION READ RSP ZONESTATUS
IndicationReadRsp();

//DEVICE_ANNOUNCEMENT
if((data_avalable == true)&&(idframe == 6)&&(MsgRe[16] == 0x88)){//An einai RFD, diladi END
DEVICE
int i;
int y;
int cmacen=0;
int cmacun=0;
idframe = 10;   //reset idframe
data_avalable = false;
Serial.printf("DEVICE_ANNOUNCEMENT");
//svse proxeira nwrAddress kai MAC tou sensor h sirinas
unsigned int nwrAdd[2];
unsigned int MAC[8];
  nwrAdd[0] = MsgRe[5];
  nwrAdd[1] = MsgRe[4];
       for(int i=0; i<=7; i++){
           MAC[i] = MsgRe[i+8];
       }

////elenxos gia rejoin
for (i = 0; i <= MaxEnrolled; i++)
{
  for(y = 0; y <= 3; y++){
    if (MAC[i]==enrolled[i].MAC[7-i])
    {
      cmacen++;
    }
  }
}

for (i = 0; i <= cntnumzone; i++)
{
  for(y = 0; y <= 3; y++){
    if (MAC[i]==unenrolled[i].MAC[7-i])
    {
      cmacun++;
    }
  }
}

if ((cmacen<4)&&(cmacun<4))
{
//ftiakse to frame tou ZDO_SIMPLE_DESC_REQ gia tin svsti dieu8insi kai to svsto checksun
ZDO_SIMPLE_DESC_REQ[4] = nwrAdd[1]; //documantation error
ZDO_SIMPLE_DESC_REQ[5] = nwrAdd[0]; //documantation error

ZDO_SIMPLE_DESC_REQ[6] = nwrAdd[1];
```

```
ZDO_SIMPLE_DESC_REQ[7] = nwrAdd[0];

idframe=8;
checksum_xor(ZDO_SIMPLE_DESC_REQ,idframe);
delay(3000);
//rota gia simple descriptor
      for(int i=0; i<=9; i++){
         Serial2.write(ZDO_SIMPLE_DESC_REQ[i]);
      }
//Serial.printf("stal8ike");
//perimene gia apantisi simple descriptor
      while(idframe != 7){    //wait for ZDO_SIMPLE_DESC_RSP from sensor-alarm
      RDA();

      }
//Serial.printf("apanti8ike");
//diavase to Device ID apo ton simple descriptor
unsigned int DeviceId[2];
 DeviceId[0] = MsgRe[14];
 DeviceId[1] = MsgRe[13];
 if (DeviceId[0] == 0x04 && DeviceId[1] == 0x02)//An einai IAS Zone
 {
//    Serial.printf("creating zone...");
   createzones( MAC, nwrAdd);
 }

}
}

// MENU
 key = keypad.getKey();
 switch (key)
 {
 case '*'://ENTER ISTALLER MENU
   installerMenu();
   break;

 case 'D'://ENTER ARM MENU
   arm();
   break;

 default:
   break;
 }
//MENU FROM WEB SERVER
 switch (webKey)
 {
 case 'D'://ENTER ARM MENU
   arm();
   break;

 default:webKey='-';
   break;
```

```
  }webKey='-';

//ALWAYS ON LCD MESSAGE
printLocalTime(); //update time and call LcdAlwaysOn if any change
updateOpenZones(); //update state of zones(C or O) and call LcdAlwaysOn if any change

}

void RDA(){

  if(Serial2.available() && (Serial2.read() == 0xFE)){
    MsgRe[0] = 0xFE;
    MsgRe[1] = Serial2.read(); //len

    for(int i=2; i<=(MsgRe[1]+4); i++){
      MsgRe[i] = Serial2.read();
  }
  recognizeMsg();
  //showMsg();//temp
     data_avalable = true;
  }
}

void showMsg(void){
for(int i=0; i<=(MsgRe[1]+4); i++){
  Serial.write(MsgRe[i]);
  }
}

void recognizeMsg(void){

switch (MsgRe[2]){

    case cmd0_SRSP_SYS_RESET:
       if(MsgRe[3] == cmd1_SRSP_SYS_RESET){
       idframe=0;   }else{idframe=100;}
    break;

    case cmd0_SRSP_SYS_NV_OSAL_WRITE:
       if(MsgRe[3] == cmd1_SRSP_SYS_NV_OSAL_WRITE){
       idframe=1;   }else{idframe=100;}
    break;

    case cmd0_SRSP_APP_CNF_BDB_SET_CHANNEL:
       if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_SET_CHANNEL){idframe=2;}
       else if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_START_COMMISSIONING) {idframe=3;}
       else {idframe=100;}
    break;

    case cmd0_SRSP_ZB_APP_REGISTER_REQUEST:
       if(MsgRe[3] == cmd1_SRSP_ZB_APP_REGISTER_REQUEST){
       idframe=4;   }else{idframe=100;}
    break;
```

```
   case cmd0_ZB_RECEIVE_DATA_INDICATION:
      if(MsgRe[3] == cmd1_ZB_RECEIVE_DATA_INDICATION){
      idframe=5;   }else{idframe=100;}
   break;

   case cmd0_ZDO_END_DEVICE_ANNCE_IND:
      if(MsgRe[3] == cmd1_ZDO_END_DEVICE_ANNCE_IND){idframe=6;}
      else if(MsgRe[3] == cmd1_ZDO_SIMPLE_DESC_RSP) {idframe=7;}
      else {idframe=100;}
   break;

default:idframe=100;    //unknown frame,with start byte FE. Call error() or move forward

}
}

void error(void){
   //printf("error");
   idframe=10; //reset idframe
}

void init_coor(void){

   int i;

      //error_retry:
      while(idframe != 0){    //wait for reset IND from cc2530. It takes 1 minute

      RDA();
      }
      //Do something when reset occurred
      //printf("reset occurred");
      //Frame 1 Frame SYS_OSAL_NV_WRITE   START_UP_OPTION
      idframe=10; //reset idframe
      for( i=0; i<=9; i++){
         Serial2.write(SYS_OSAL_NV_WRITE_START_UP_OPTION[i]);
      }
      //=========================================================================

      while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
      RDA();

      }
      idframe=10; //reset idframe
      //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
      //Frame 2 Frame SYS_RESET   Reset cc2530 to clear NV
      for( i=0; i<=5; i++){
      Serial2.write(SYS_RESET[i]);
      }
      //=========================================================================

      while(idframe != 0){    //wait for reset from cc2530
```

```
RDA();

}
idframe=10; //reset idframe
//Do something when SRSP_SYS_NV_OSAL_WRITE arrive
//Frame 3 Frame SYS_OSAL_NV_WRITE  build cc2530 as coordinator
for( i=0; i<=9; i++){
Serial2.write(SYS_OSAL_NV_WRITE[i]);
}
//========================================================================

while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
RDA();

}
idframe=10; //reset idframe
//Do something when SRSP_SYS_NV_OSAL_WRITE arrive
//Frame 4   set channel to 13
for( i=0; i<=9; i++){
Serial2.write(APP_CNF_BDB_SET_CHANNEL_secontary[i]);
}
//========================================================================

   while(idframe != 2){     //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_secontary from
cc2530
RDA();

}
idframe=10; //reset idframe
//Do something when SRSP_UTIL_SET_CHANNELS arrive
//Frame 5   set PAN ID to 0A
for( i=0; i<=9; i++){
Serial2.write(APP_CNF_BDB_SET_CHANNEL_primary[i]);
}
//========================================================================

    while(idframe != 2){     //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_primary from
cc2530
RDA();

}
idframe=10; //reset idframe
//Do something when SRSP_UTIL_SET_PANID arrive
//Frame 6   Register app to endpoint 8
for( i=0; i<=5; i++){
   Serial2.write(APP_CNF_BDB_START_COMMISSIONING[i]);
}
//========================================================================

   while(idframe != 3){     //wait for SRSP_APP_CNF_BDB_START_COMMISSIONING from
cc2530
RDA();
```

```
    }
    delay(1500); //perimene na er8oun ta adiafora frames
    for( i=0; i<=15; i++){  //diavase merika adiafora frame SRSP
    RDA();
    }data_avalable = false;

    idframe=10; //reset idframe
    //Do something when SRSP_ZB_APP_REGISTER_REQUEST arrive
    //Frame 7   start Zigbee stack
    for( i=0; i<=9; i++){
       Serial2.write(SYS_OSAL_NV_WRITE_ZDO[i]);
    }
    //=========================================================================
    while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
    RDA();

    }
    idframe=10;
    for( i=0; i<=5; i++){
       Serial2.write(APP_CNF_BDB_START_COMMISSIONING_steering[i]);
    }
    while(idframe != 3){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
    RDA();

    }
    delay(1500); //perimene na er8oun ta adiafora frames
    for( i=0; i<=15; i++){  //diavase merika adiafora frame SRSP
    RDA();
    }
    data_avalable = false;
     idframe=10;

    for( i=0; i<=15; i++){
       Serial2.write(ZB_APP_REGISTER_REQUEST[i]);
    }
    delay(1500); //perimene na er8oun ta adiafora frames
    for( i=0; i<=15; i++){  //diavase merika adiafora frame SRSP
    RDA();
    }
    data_avalable = false;
     idframe=10;
    //Coordinator initialization completed. Go to main
}

void createzones(unsigned int mac[], unsigned int nwr[]){//kaleitai otan ginete announcment apo end
device sensor

  cntnumzone++;  //auksi8ikan oi zones ana mia
  statezones[cntnumzone]=1; //state of current zone joined --> unenrolled
  int currZone;
  currZone=cntnumzone;
```

```
 // zone unenrolled[currZone] = {}; //create new empty object
String num=String(currZone);  //metatropi ari8mou zonis se String gia xrisi sto onoma

  String name= String("unenrolled_" + num);
  unenrolled[currZone].zoneName=name;

  for (int i = 0; i <= 7; i++)
  {
    unenrolled[currZone].MAC[i]= mac[7-i];
  }
   for (int i = 0; i <= 1; i++)
  {
    unenrolled[currZone].nwrAddr[i]= nwr[i];
  }

  unenrolled[currZone].zoneState = '-'; //unused in unenrolled

Serial.print(unenrolled[currZone].zoneName);
  //tupvse tis plirofories tou sensor
/*

  Serial.print(unenrolled[currZone].zoneName);

    for (int i = 0; i <= 1; i++)
  {
    Serial.write(unenrolled[currZone].nwrAddr[i]);
  }
*/
}

void checksum_xor(unsigned int array[], int id){
unsigned int c; //checksum
//ypologismos checksum tou array
    c = array[1] ^ array[2];
  for (int i=3; i<=array[1]+3; i++){
    c = c ^ array[i];
    }

//apo8ikeusi tou checksum sto katallilo frame
switch (id)
{
case 8:
  ZDO_SIMPLE_DESC_REQ[9] = c;
  break;

case 9:
  ZB_SEND_DATA_REQUEST_Identify[17] = c;
  break;

default:idframe=10;
  break;
}
idframe=10;
```

```
}

void installerMenu(void){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Installer Menu");
  lcd.setCursor(0,1);
  lcd.print("Enter Code...");

  char correctcode[4] = {'9','9','9','9'};
  int tempcode[4] = {99,99,99,99};
  char tempkey;
  int comparearray=0;
  int timeup = 0;
  unsigned long installerTime = 10000 + millis(); //gia 10 deuterolepta
  unsigned long installerTimer = millis(); //current time

//diavase 4-psifio kvdiko tou installer gia 10 deuterolepta
for(int i=0; i<=3; i++){

        if(timeup == 1){
          //timeup=0;
           break;
         }
       while (tempcode[i]==99)
       {
         installerTimer = millis();
          tempkey = keypad.getKey();
         if(!(tempkey == NULL)){
           tempcode[i] =tempkey;
         }

         if(installerTimer >= installerTime){
           timeup=1;
           break;
         }

       }
  }

//compare tempcode with correctcode
for (int i = 0; i <= 3; i++)
{
  if(tempcode[i] != correctcode[i]){
    comparearray = 0;
  }
  else
  {
    comparearray++;
  }
}

if(comparearray == 4){  //an isagoume sosta ton kodiko
```

```
Serial.println("Correct code");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("installer menu");
lcd.setCursor(0,1);
lcd.print("1:->En,  2:Un");
//installer menu
int brk=0;
   while (1)
   {
    tempkey = keypad.getKey();
       switch (tempkey)
       {
       case '1': //patontas 1 vlepoume tis enrolled zones
         ShowEnrolled();
         lcd.clear();
         lcd.setCursor(0,0);
         lcd.print("installer menu");
         lcd.setCursor(0,1);
         lcd.print("1:->En,  2:Un");
         break;

       case '2': //patontas 2 vlepoume tis unenrolled zones
         ShowUnenrolled();
         lcd.clear();
         lcd.setCursor(0,0);
         lcd.print("installer menu");
         lcd.setCursor(0,1);
         lcd.print("1:->En,  2:Un");
         break;

       case '#': //vgenoume apo to installer menu
         brk=1;
         break;

       default:tempkey='@';
       }
   if(brk==1){LcdAlwaysOn();break;}//vges apo tin while
   }
}

else if (timeup == 1)
{
 Serial.println("time out");
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("time out");
 delay(2000);
 timeup = 0;
 LcdAlwaysOn();
}
else
{
```

```
  Serial.println("Wrong code");
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("Wrong code");
  delay(2000);
  LcdAlwaysOn();
}
}

void LcdAlwaysOn(void){
int i;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(myhour);lcd.print(":");lcd.print(mymin);
  lcd.setCursor(0,1);
  lcd.print("Zones:");
  for (i = 1; i <= MaxEnrolled; i++)
  {
    if(statenrolledzones[i] == 1){
      lcd.print(enrolled[i].zoneName);
    }
  }
}

void ShowUnenrolled(void){
  int CurrNumList=1;
  int MaxNumList = 0;
  int FirstNumList = 0;
  char tempkey;
  int existun=0; //existun=0 --> No Unenrolled // existun=1 --> exist Unenrolled

  lcd.clear();
  lcd.setCursor(0,0);

for (int i = 1; i <= cntnumzone; i++) //elenxos an yparxei Unenrolled
{
  if (statezones[i]==1)
  {
    existun=1;
  }

}
  if(existun == 0){
    lcd.print("No Unenrolled");
    delay(5000);
    lcd.clear();
    lcd.setCursor(0,0);
  }
  else{
   int brk=0; //gia na vgo apo while(1)
   int update=1; //gia ananeosi apo velakia

       //find first unenrolled
```

```
   for (int i = 1; i <= cntnumzone; i++) //elenxos gia to proto Unenrolled
      {
        if (statezones[i]==1)
        {
         CurrNumList=i;
         break;
        }
        if (i==cntnumzone)
        {
         existun=0;//den uparxoun pia alla unenrolled
        }
      }
   //find last unenrolled
   for (int i = 1; i <= cntnumzone; i++) //elenxos gia to teletaio Unenrolled
      {
        if (statezones[i]==1)
        {
         MaxNumList=i;
        }
        if ((i==cntnumzone) && (MaxNumList==0))
        {
         existun=0;//den uparxoun pia alla unenrolled
        }
      }

while (existun==1)//oso yparxoun unenrolled
  {

   if (update==0)
   {
   //find first unenrolled
   for (int i = 1; i <= cntnumzone; i++) //elenxos gia to proto Unenrolled
      {
        if (statezones[i]==1)
        {
         FirstNumList=i;
         break;
        }
        if (i==cntnumzone)
        {
         existun=0;//den uparxoun pia alla unenrolled
        }
      }
   //find last unenrolled
   for (int i = 1; i <= cntnumzone; i++) //elenxos gia to teletaio Unenrolled
      {
        if (statezones[i]==1)
        {
         MaxNumList=i;
        }
        if ((i==cntnumzone) && (MaxNumList==0))
        {
         existun=0;//den uparxoun pia alla unenrolled
```

```
        }
      }
    }

    tempkey = keypad.getKey();

    switch (update){
     case 1:
     {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(unenrolled[CurrNumList].zoneName); //11 xaraktires
      update=0;
      break;
     }
     case 2:
     {
      tempkey='@';
         while (tempkey!='R')
         {
          tempkey = keypad.getKey();
          if (tempkey=='D')
          {
           //send indication sto unenrolled[CurrNumList].nwrAddr gia 2 deuterolepta
           ZB_SEND_DATA_REQUEST_Identify[5] = unenrolled[CurrNumList].nwrAddr[0];
           ZB_SEND_DATA_REQUEST_Identify[4] = unenrolled[CurrNumList].nwrAddr[1];
           idframe=9;
           checksum_xor(ZB_SEND_DATA_REQUEST_Identify,idframe);
            for(int i=0; i<=17; i++){
             Serial2.write(ZB_SEND_DATA_REQUEST_Identify[i]);
              }

           tempkey='@';//reset, mporei axristo
         }
          if (tempkey=='1'||tempkey=='2'||tempkey=='3'||tempkey=='4'||tempkey=='5'||tempkey=='6'||
tempkey=='7'||tempkey=='8'||tempkey=='9')
         {
          String num;
          num=tempkey;
          tempkey='@';
          while (tempkey!='*')
          {
           tempkey = keypad.getKey();
                        if  (tempkey=='1'||tempkey=='2'||tempkey=='3'||tempkey=='4'||tempkey=='5'||
tempkey=='6'||tempkey=='7'||tempkey=='8'||tempkey=='9'){
             num=String(num + tempkey);
           }
           /* String name= String("unenrolled_" + num); */
          }
         tempkey='R';
         /////ftia3e tin kanoniki enrolled zoni me ton ari8mo pou elave o xristis "num"///////
         String name= String("Zone " + num);
         countenrl++;
```

```
            enrolled[countenrl].zoneName = name;
            for (int i = 0; i <= 7; i++)
            {
              enrolled[countenrl].MAC[i] = unenrolled[CurrNumList].MAC[i];
            }
            for (int i = 0; i <= 1; i++)
            {
              enrolled[countenrl].nwrAddr[i] = unenrolled[CurrNumList].nwrAddr[i];
            }
            enrolled[countenrl].zoneState = '-'; //default until receive indication from zone
            MaxEnrolled++;
            Serial.print(enrolled[countenrl].zoneName);
            }
           }

          //apo8ikeuse ari8mo gia auti tin zoni
          statezones[CurrNumList]=2; // ala3e to state se enrolled
          update=0;
          break;
      }
      default:update=0;
     }
      switch (tempkey)
        {
        case 'A': //patontas A > next
          while(CurrNumList<MaxNumList){
          CurrNumList++;
             if(statezones[CurrNumList]==1){update=1;break;}
          }
          break;

        case 'B': //patontas B < back
          while(CurrNumList>FirstNumList){
          CurrNumList--;
             if(statezones[CurrNumList]==1){update=1;break;}
          }
          break;

        case '*': //gia isagogi ari8mou zonis kai enroll
          update=2;
          break;

        case '#': //vgenoume apo to ShowUnenrolled
          brk=1;
          break;//vges apo to switch

        default:tempkey='@';

        }
   if(brk==1){existun=0; break;}//vges apo tin while
    }

}
```

```
}

void ShowEnrolled(void){
  int CurrNumList=1;
  int MaxNumList = 0;
  int FirstNumList = 0;
  char tempkey;
  int existen=0; //existun=0 --> No enrolled // existun=1 --> exist Enrolled

  lcd.clear();
  lcd.setCursor(0,0);

  for (int i = 1; i <= cntnumzone; i++) //elenxos an yparxei enrolled
{
  if (statezones[i]==2)
  {
    existen=1;
  }
}

  if(existen == 0){
    lcd.print("No Enrolled");
    delay(5000);
    lcd.clear();
    lcd.setCursor(0,0);
  }
  else
  {
    int brk=0;
    int update=1;

        //find first enrolled
        for (int i = 1; i <= cntnumzone; i++) //elenxos gia to proto enrolled
            {
              if (statezones[i]==2)
              {
                CurrNumList=i;
                break;
              }
              if (i==cntnumzone)
              {
                existen=0;//den uparxoun pia alla enrolled
              }
            }
        //find last enrolled
        for (int i = 1; i <= cntnumzone; i++) //elenxos gia to teletaio enrolled
            {
              if (statezones[i]==2)
              {
                MaxNumList=i;
              }
              if ((i==cntnumzone) && (MaxNumList==0))
              {
```

```
          existen=0;//den uparxoun pia alla enrolled
        }

    }
while (existen==1)//oso yparxoun Enrolled
{

 if (update==0)
 {
 //find first enrolled
 for (int i = 1; i <= cntnumzone; i++) //elenxos gia to proto enrolled
    {
       if (statezones[i]==2)
       {
        FirstNumList=i;
        break;
       }
       if (i==cntnumzone)
       {
        existen=0;//den uparxoun pia alla enrolled
       }

    }
 //find last enrolled
 for (int i = 1; i <= cntnumzone; i++) //elenxos gia to teletaio enrolled
    {
       if (statezones[i]==2)
       {
        MaxNumList=i;
       }
       if ((i==cntnumzone) && (MaxNumList==0))
       {
        existen=0;//den uparxoun pia alla enrolled
       }
    }
 }

 tempkey = keypad.getKey();

  if (update==1)
  {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print(enrolled[CurrNumList].zoneName); // xaraktires
   update=0;
  }
 switch (tempkey)
    {
    case 'A': //patontas A > next
      while(CurrNumList<MaxNumList){
      CurrNumList++;
         if(statezones[CurrNumList]==2){update=1;break;}
      }
```

```
            break;

          case 'B': //patontas B < back
            while(CurrNumList>FirstNumList){
            CurrNumList--;
               if(statezones[CurrNumList]==2){update=1;break;}
            }
            break;

          case '#': //vgenoume apo to ShowEnrolled
            brk=1;
            break;//vges apo to switch

          case 'D': //stelnoume Identify gia tin zoni poy apikonizetai
                ZB_SEND_DATA_REQUEST_Identify[5] = enrolled[CurrNumList].nwrAddr[0];
                ZB_SEND_DATA_REQUEST_Identify[4] = enrolled[CurrNumList].nwrAddr[1];
                idframe=9;
                checksum_xor(ZB_SEND_DATA_REQUEST_Identify,idframe);
                 for(int i=0; i<=17; i++){
                  Serial2.write(ZB_SEND_DATA_REQUEST_Identify[i]);
                    }

                tempkey='@';//reset, mporei axristo
            break;

          default:tempkey='@';

          }
        if(brk==1){existen=0; break;}//vges apo tin while
      }
    }
}

void printLocalTime(){
  struct tm timeinfo; // panto sto Globals.h
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  myhour = timeinfo.tm_hour;
  mymin = timeinfo.tm_min;

  if ((lasthour != myhour) || (lastmin != mymin))
  {
  Serial.print(myhour);
  Serial.print(":");
  Serial.print(mymin);
  Serial.println();
  LcdAlwaysOn();
  }

lasthour = myhour;
lastmin = mymin;
```

```
}

void updateOpenZones(void){
int i;
int refresh=0;

  for (i = 0; i <= MaxEnrolled; i++)
  {
    if (enrolled[i].zoneState=='O'){
      if(statenrolledzones[i] == 0){refresh=1;}
      statenrolledzones[i] = 1;
    }
    if (enrolled[i].zoneState=='C'){
      if(statenrolledzones[i] == 1){refresh=1;}
      statenrolledzones[i] = 0;
    }
  }

  if (refresh == 1)
  {
    LcdAlwaysOn();
  }

}

void arm(void){
  int i;
  int allclosed=0;
  unsigned long currtime=0;
  unsigned long thresholdtime;
  char tempkey;
  int comparearray=0;
  for ( i = 0; i <= 3; i++)
  {
    tempDisarmCode[i] = 99;
  }

//Elenxos an oles oi zones einai kleistes
  for (i = 1; i <= MaxEnrolled; i++)
  {
    if(statenrolledzones[i] == 0){
      allclosed++;//if agter for allclosed = MaxEnrolled then all zones are closed
    }
  }


  if ((allclosed == MaxEnrolled)&&(MaxEnrolled>0))//an oles oi zones einai kleistes meine stin arm
  {

    ////buzzer mexri na oplisei////
    currtime = millis();
    thresholdtime = currtime + armtime; //armtime sto Globals.h
    while (currtime < thresholdtime)
```

```
    {
    ledcWrite(ledChannel, 50);
    delay(500);
    ledcWrite(ledChannel, 0);
    delay(500);
     currtime = millis();
    }ledcWrite(ledChannel, 0);
    ////////////////////////////////
    armFlag = "ARM";
////////////////////////// KATASTASI OPLISMENOU SUNAGERMOU ////////////////////
      while (alarmflag==0)
      {
       if(AlarmIdication()&&(alarmflag==0)){AlarmMenu();break;}
       RDA();
       IndicationReadRsp();
       printLocalTime(); //update time and call LcdAlwaysOn if any change
       updateOpenZones(); //update state of zones(C or O) and call LcdAlwaysOn if any change
            //diavase 4-psifio kvdiko tou installer gia 10 deuterolepta
       for(int i=0; i<=3; i++){

            while (tempDisarmCode[i]==99)
            {
              if(AlarmIdication()&&(alarmflag==0)){AlarmMenu();break;}
              RDA();
              IndicationReadRsp();
              printLocalTime(); //update time and call LcdAlwaysOn if any change
              updateOpenZones(); //update state of zones(C or O) and call LcdAlwaysOn if any change
             tempkey = keypad.getKey();
             if(!(tempkey == NULL)){
              tempDisarmCode[i] =tempkey;
             }
            }
            if(alarmflag==1){break;}
        }

       //compare tempcode with correctcode
       for (int i = 0; i <= 3; i++)
       {
        if(tempDisarmCode[i] != disarmcode[i]){
          comparearray = 0;
        }
        else
        {
          comparearray++;
        }
       }

       if((comparearray == 4)&&alarmflag==0){
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Disarmed");
        delay(3000);
        LcdAlwaysOn();
```

```
        armFlag = "DISARM";
        break;
       }
       else if ((comparearray < 4)&&(alarmflag==0))
       {
        for (int i = 0; i <= 3; i++)//reset tempcode
        {
         tempDisarmCode[i] = 99;
        }
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Wrong Code");
        delay(3000);
        LcdAlwaysOn();
       }
      }

////////////////////////////////////////////////////////////////////
  }else if ((MaxEnrolled == 0)&&(cntnumzone == 0))
  {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("no zone enrolled");
   delay(3000);
   LcdAlwaysOn();
  }
  else//an yparxoun anoixtes zones tipose munima kai vges apo arm()
  {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("close all zones");
   delay(3000);
   LcdAlwaysOn();
  }

alarmflag = 0; //clear alarm
}

void IndicationReadRsp(void){

//INDICATION READ RSP ZONESTATUS
if((data_avalable == true)&&(idframe == 5)&&(MsgRe[7] == 0x05)&&(MaxEnrolled>0))
{//kainourgio paketo,indicatio,cluster ID 0x0500,yparxoun enrolled zones
  //showMsg();
  idframe = 10;   //reset idframe
  data_avalable = false;

unsigned int nwrAdd[2];
  nwrAdd[0] = MsgRe[5];
  nwrAdd[1] = MsgRe[4];

  for (int i = 1; i <= MaxEnrolled; i++)
  {
```

```
   if ((nwrAdd[0]==enrolled[i].nwrAddr[0])&&(nwrAdd[1]==enrolled[i].nwrAddr[1]))
   {
      switch (MsgRe[17])
      {
      case 0x00:
        enrolled[i].zoneState = 'C';
        Serial.print(enrolled[i].zoneName);
        Serial.print(enrolled[i].zoneState);
        break;

      case 0x01:
        enrolled[i].zoneState = 'O';
        Serial.print(enrolled[i].zoneName);
        Serial.print(enrolled[i].zoneState);
        break;

      default:enrolled[i].zoneState = '-';
        break;
      }
   }
  }
}
}

int AlarmIdication(void){
  int cnt=0;
  //An einai esto mia zoni anoixti---gia eidopoieisi alarm
  for (int i = 1; i <= MaxEnrolled; i++)
  {
    if(statenrolledzones[i] == 1){
      cnt++;
    }
  }

  if (cnt>0)
  {
    alarmindication=1;
  }
  else
  {
    alarmindication=0;
  }

  return alarmindication;
}

void AlarmMenu(void){
  armFlag = "DISARM";
  alarmflag=1;//set alarm
  unsigned long currtime=0;
  unsigned long thresholdtime;

  ledcWrite(ledChannel, 50);
```

```
  currtime = millis();
  thresholdtime = currtime + alarmtime;
  while (currtime < thresholdtime)
  {
    currtime = millis();
  }ledcWrite(ledChannel, 0);

  sendEmail();
}

void sendCallback(SendStatus msg) {// Callback function to get the Email sending status
  // Print the current status
  Serial.println(msg.info());

  // Do something when complete
  if (msg.success()) {
    Serial.println("----------------");
  }
}

void sendEmail(void){
    // Set the SMTP Server Email host, port, account and password
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);

  // For library version 1.2.0 and later which STARTTLS protocol was supported,the STARTTLS will
be
    // enabled automatically when port 587 was used, or enable it manually using setSTARTTLS
function.
  //smtpData.setSTARTTLS(true);

  // Set the sender name and Email
  smtpData.setSender("ESP32", emailSenderAccount);

  // Set Email priority or importance High, Normal, Low or 1 to 5 (1 is highest)
  smtpData.setPriority("High");

  // Set the subject
  smtpData.setSubject(emailSubject);

  // Set the message with HTML format
    smtpData.setMessage("<div style=\"color:#2f4468;\"><h1>Alarm Triggered</h1><p>- Sent from
ESP32 board</p></div>", true);
  // Set the email message in text format (raw)
  //smtpData.setMessage("Hello World! - Sent from ESP32 board", false);

  // Add recipients, you can add more than one recipient
  smtpData.addRecipient(emailRecipient);
  //smtpData.addRecipient("YOUR_OTHER_RECIPIENT_EMAIL_ADDRESS@EXAMPLE.com");

  smtpData.setSendCallback(sendCallback);

  //Start sending Email, can be set callback function to track the status
  if (!MailClient.sendMail(smtpData))
```

```
   Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  //Clear all data from Email object to free memory
  smtpData.empty();
}

////////////////////////////////////////////////////////
/////       command_field.h          ////////////////////
////////////////////////////////////////////////////////
//0 idframe
const unsigned int cmd0_SRSP_SYS_RESET = 0x41;
const unsigned int cmd1_SRSP_SYS_RESET = 0x80;
//1 idframe
const unsigned int cmd0_SRSP_SYS_NV_OSAL_WRITE = 0x61;
const unsigned int cmd1_SRSP_SYS_NV_OSAL_WRITE = 0x09;
//2 idframe
const unsigned int cmd0_SRSP_APP_CNF_BDB_SET_CHANNEL = 0x6F;
const unsigned int cmd1_SRSP_APP_CNF_BDB_SET_CHANNEL = 0x08;
//3 idframe
const unsigned int cmd0_SRSP_APP_CNF_BDB_START_COMMISSIONING = 0x6F;
const unsigned int cmd1_SRSP_APP_CNF_BDB_START_COMMISSIONING = 0x05;
//4 idframe
const unsigned int cmd0_SRSP_ZB_APP_REGISTER_REQUEST = 0x66;
const unsigned int cmd1_SRSP_ZB_APP_REGISTER_REQUEST = 0x0A;
//5 idframe
const unsigned int cmd0_ZB_RECEIVE_DATA_INDICATION = 0x46;
const unsigned int cmd1_ZB_RECEIVE_DATA_INDICATION = 0x87;
//6 idframe
const unsigned int cmd0_ZDO_END_DEVICE_ANNCE_IND = 0x45;
const unsigned int cmd1_ZDO_END_DEVICE_ANNCE_IND = 0xC1;
//7 idframe
const unsigned int cmd0_ZDO_SIMPLE_DESC_RSP = 0x45;
const unsigned int cmd1_ZDO_SIMPLE_DESC_RSP = 0x84;

//8 idframe ZDO_SIMPLE_DESC_REQ
//9 idframe ZB_SEND_DATA_REQUEST_Identify

////////////////////////////////////////////////////////
/////       ConstantFrames.h         ////////////////////
////////////////////////////////////////////////////////
unsigned int SYS_OSAL_NV_WRITE_START_UP_OPTION[10] =
{0xFE,0x05,0x21,0x09,0x03,0x00,0x00,0x01,0x03,0x2C};   //NV clear after reset
unsigned int SYS_RESET[6] = {0xFE,0x01,0x41,0x00,0x00,0x40};    //reset cc2530
unsigned int SYS_OSAL_NV_WRITE[10] =
{0xFE,0x05,0x21,0x09,0x87,0x00,0x00,0x01,0x00,0xAB};   //build cc2530 for coordinator
unsigned int SYS_OSAL_NV_WRITE_R[10] =
{0xFE,0x05,0x21,0x09,0x87,0x00,0x00,0x01,0x01,0xAA};   //build cc2530 for router

unsigned int APP_CNF_BDB_SET_CHANNEL_secontary[10] =
{0xFE,0x05,0x2F,0x08,0x00,0x00,0x00,0x00,0x00,0x22};
unsigned int APP_CNF_BDB_SET_CHANNEL_primary[10] =
{0xFE,0x05,0x2F,0x08,0x01,0x00,0x20,0x00,0x00,0x03};
unsigned int APP_CNF_BDB_START_COMMISSIONING[6] = {0xFE,0x01,0x2F,0x05,0x04,0x2F};
```

```
unsigned int SYS_OSAL_NV_WRITE_ZDO[10] =
{0xFE,0x05,0x21,0x09,0x8F,0x00,0x00,0x01,0x01,0xA2}; // receive ZDO
unsigned int APP_CNF_BDB_START_COMMISSIONING_steering[6] =
{0xFE,0x01,0x2F,0x05,0x02,0x29};

unsigned int ZB_SEND_DATA_REQUEST_off[16] =
{0xFE,0x0B,0x26,0x03,0xFF,0xFF,0x06,0x00,0x00,0x00,0x01,0x03,0x01,0x01,0x00,0x2A};  //Send
command off to broadcast
unsigned int ZB_SEND_DATA_REQUEST_toogle[16] =
{0xFE,0x0B,0x26,0x03,0xFF,0xFF,0x06,0x00,0x00,0x00,0x01,0x03,0x01,0x01,0x02,0x28};  //Send
toggle to broadcast
unsigned int ZB_SEND_DATA_REQUEST_Identify[18] =
{0xFE,0x0D,0x26,0x03,0xFF,0xFF,0x03,0x00,0x00,0x00,0x01,0x05,0x01,0x01,0x00,0x05,0x00,0x2
A};  //Send Identify to broadcast
unsigned int ZB_PERMIT_JOINING_REQUEST[8] =
{0xFE,0x03,0x26,0x08,0xFF,0xFF,0x78,0x55};
unsigned int ZDO_SIMPLE_DESC_REQ[10] =
{0xFE,0x05,0x25,0x04,0xFF,0xFF,0xFF,0xFF,0x08,0xFF};//Request simple descriptor se endpoint 8
kai nwrAdrr(Allazei apo to vasiko programma mazi kai to checksum)

//////////////////////////////////////////////////////
/////       Globals.h          ////////////////////
//////////////////////////////////////////////////////
//dilosi metavliton
bool data_avalable = false;
unsigned int MsgRe[40]; //max mege8os frame 40 bytes
int idframe=10; //default 10. error 100
int statezones[30]; //init in setup //truck state of zones exist:
                                    //0 no zone yet
                                    //1 unenrolled
                                    //2 enrolled
int statenrolledzones[30]; //init in setup //truck state of zones exist:
                                    //0 close
                                    //1 open

int cntnumzone=0; //truck number of zones __ oses sundeonte
int countenrl=0; //truck number of enrolled zones__oses kanoun enroll apo ton xristi
int MaxEnrolled=0;
int alarmindication=0; // enimeronete stin updateOpenZones()
                    // 0 an oles oi zones einai kleistes
                   // 1 an yparxei esto mia zoni anoixti
int alarmflag=0; // 0 clear alarm
          // 1 set alarm
String armFlag = "DISARM";

int armtime=10000;//xronos se millseconds gia oplisi tou sunagermou
int alarmtime=10000;

/////// Soft AP ///////////
String ssidAp = "MyESP32AP";
String passwordAp = "123456789";
int setSsid = 0;
int setPass = 0;
```

```
const char* PARAM_INPUT_3 = "ssid";
const char* PARAM_INPUT_4 = "password";
//////////WIFI//////////
String ssid     = "";
String password = "";
/////////WEB SERVER//////////
const char* PARAM_INPUT_1 = "armtime";
const char* PARAM_INPUT_2 = "alarmtime";

//////////NTP TIME//////////
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 7200;
const int   daylightOffset_sec = 3600;
int lasthour=100;
int lastmin=100;
int myhour;
int mymin;

//////// ESP32 EMAIL //////////
// To send Email using Gmail use port 465 (SSL) and SMTP Server smtp.gmail.com
// YOU MUST ENABLE less secure app option https://myaccount.google.com/lesssecureapps?pli=1
#define emailSenderAccount    "mymail.jd7.esp32@gmail.com"
#define emailSenderPassword   "753159789Po@"
#define emailRecipient        "mymail.jd7@gmail.com"
#define smtpServer            "smtp.gmail.com"
#define smtpServerPort        465
#define emailSubject          "ESP32 Alarm"

//////////PWM gia BUZZER//////////
int freq = 5000;
int ledChannel = 0;
int resolution = 8;


//////// kodikos afoplisis ////////
char disarmcode[4] = {'1','2','3','4'};
int tempDisarmCode[4] = {99,99,99,99};

//gia ka8e zoni apo8ikeuete i MAC kai nwrAddress kai stin sunexeia antistixite me filiki onomasia "zone 1"
  struct zone
  {
    unsigned int MAC[8];
    unsigned int nwrAddr[2];
    String zoneName;
       char zoneState; //zone state for enrolled zones: 'C' for closed, 'O' for open, '-' for unused to unenrolled

  };

    zone unenrolled[30] = { //pinakas 30 objects gia unenrolled zones__ To unenrolled[0] den xrisimopieitai
//zone(),zone()
```

```
  };

  zone enrolled[30] = { //pinakas 30 objects gia enrolled zones__ To enrolled[0] den xrisimopieitai
//zone(),zone()
  };


//////////////////////////////////////////////////////
/////       index    ////////////////////
//////////////////////////////////////////////////////

<!DOCTYPE html>
<html>

<head>
  <title>%Alarm System%</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
  <h1>ESP32 Home Alarm System</h1>

  <p>Alarm state: <strong id="state"> %STATE%</strong></p>
  <p>zones state: <strong> %ZONES%</strong></p>
  <button class="button" onclick="myFunction()">%BUTTON_NAME%</button><br><br>

  <form action="/get">
    Time to arm (in sec): <input type="text" name="armtime">
    <input type="submit" value="Submit">
  </form><br>

  <form action="/get">
    Alarm time (in sec): <input type="text" name="alarmtime">
    <input type="submit" value="Submit">
  </form><br>

  <script>
    function myFunction() {

var x = document.getElementById("state").innerHTML;

if(x.trim() === "DISARM")
{
  document.location = '/arm';
}else{
  document.location = '/disarm';
}

    }
```

```
   </script>
</body>
</html>

//////////////////////////////////////////////////////
/////       indexAP           //////////////////////
//////////////////////////////////////////////////////
<!DOCTYPE html>
<html>

<head>
  <title>settings</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
</head>

<body>
  <h1>ESP32 Settings</h1>

  <form action="/get">
    Ender SSID: <input type="text" name="ssid">
    <input type="submit" value="Submit">
  </form><br>

  <form action="/get">
    Enter Pass: <input type="text" name="password">
    <input type="submit" value="Submit">
  </form><br>

</body>
</html>
```

ΠΑΡΑΡΤΗΜΑ Β: Πηγαίος Κώδικας END DEVICE

```
#include <main.h>
#include "ConstantFrames.h"
#include "Globals.h"
#include "command_field.h"

#use rs232(baud=115200,parity=N,xmit=PIN_A4,rcv=PIN_A5,bits=8)

//dilvsi sunartisevn
void init(void);
void init_end_device(void);
void RA_isr(void);
void RDA_isr(void);
void read_General_format_frame(void);
void showMsg(void);
void recognizeMsg(void);
void error(void);
```

```
void main(void){
   int i;
   init();
   init_end_device();  //Sends initialization frames for router
   enable_interrupts(INT_RA2);

 while(1)
   {
      //INDICATION
      if((idframe == 5)&&(MsgRe[6] == 0x03)&&(MsgRe[7] == 0x00&&(MsgRe[12] == 0x00))){
         long timeid = MsgRe[13];
         timeid = timeid*0x03E8;
         for( i=0; i<=6; i++){
         putc(SYS_GPIO_led_on[i]);
         }
      delay_ms(timeid);
            //set led on until setup cc2530 finish
         for( i=0; i<=6; i++){
         putc(SYS_GPIO_led_off[i]);
         }
         idframe=100;
      }
   }
}

#INT_RDA
void RDA_isr(void){
      disable_interrupts(GLOBAL);

   MsgRe[0] = getc();
   if(MsgRe[0] == SOF){    //Frame with no FE for start byte reject
   MsgRe[1] = getc();  //len
   MsgRe[2] = getc();  //cmd0
   MsgRe[3] = getc();  //cmd1

   read_General_format_frame();
   MsgRe[MsgRe[1]+4] = getc();  //FCS
   //showMsg();
   recognizeMsg();
   enable_interrupts(GLOBAL);
   }
   else{
   enable_interrupts(GLOBAL);
   }
}

#INT_RA
void RA_isr(void){
      disable_interrupts(INT_RA2);
      int i;

      if(input(PIN_A2))
      {//kleisto magnitiko
```

```
            //send close atrribute
            //output_low(PIN_A0);//local optical idication
            for(i=0; i<=21; i++){
               putc(ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_close[i]);
            }
         }
         else
         {//anoixto magnitiko
            //send open attribute
            //output_high(PIN_A0);//local optical idication
            for(i=0; i<=21; i++){
               putc(ZB_SEND_DATA_REQUEST_READ_ATTR_RSP_open[i]);
            }
         }
         enable_interrupts(INT_RA2);
         clear_interrupt(INT_RA2);   //*******************
}

void read_General_format_frame(void){

   for(int i=4; i<=MsgRe[1]+3; i++){
      MsgRe[i] = getc();
   }
}

void showMsg(void){
   for(int i=0; i<=MsgRe[1]+4; i++){
      putc(MsgRe[i]);
   }
}

void init(void){

         //init MsgRe elements to 0. Array for incoming frames
         for( int i=0; i<40; i++){
         MsgRe[i]=0;
        }

   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);
   set_tris_a(0xBB); //0b10111011 MSB ... LSB

   //reset cc2530
   output_low(PIN_A0);
   delay_ms(1000);
   output_high(PIN_A0);
   delay_ms(3000);
   idframe=10;
}

void init_end_device(void){
int i;
    if(input(PIN_A1)){
```

```
error_retry:
while(idframe != 0){    //wait for reset IND from cc2530. It takes 1 minute
   if(idframe==100){
      error();
      goto error_retry;
   }
}

//Do something when reset occurred
//printf("reset occurred");
//Frame 1 Frame SYS_OSAL_NV_WRITE   START_UP_OPTION
idframe=10; //reset idframe
for( i=0; i<=9; i++){
   putc(SYS_OSAL_NV_WRITE_START_UP_OPTION[i]);
}
//=====================================================================
while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
   if(idframe==100){
      error();
      goto error_retry;
   }
}
idframe=10; //reset idframe
//Do something when SRSP_SYS_NV_OSAL_WRITE arrive
//Frame 2 Frame SYS_RESET   Reset cc2530 to clear NV
for( i=0; i<=5; i++){
putc(SYS_RESET[i]);
}

//=====================================================================
while(idframe != 0){    //wait for reset from cc2530
   if(idframe==100){
      error();
      goto error_retry;
   }
}
idframe=10; //reset idframe
      //=====================================================================
//setup indication led
for( i=0; i<=6; i++){
   putc(SYS_GPIO_direction[i]);
}

delay_ms(500);
idframe=10;

//Frame 3 Frame SYS_OSAL_NV_WRITE  build cc2530 as End_Device
for( i=0; i<=9; i++){
putc(SYS_OSAL_NV_WRITE_E[i]);
}

while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
   if(idframe==100){
```

```
            error();
            goto error_retry;
          }
        }
        idframe=10; //reset idframe
        //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
        //Frame 4   set channel to 13
        for( i=0; i<=9; i++){
        putc(APP_CNF_BDB_SET_CHANNEL_secontary[i]);
        }
        //=========================================================================
          while(idframe != 2){     //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_secontary from
cc2530
        }
        idframe=10; //reset idframe
        //Do something when SRSP_APP_CNF_BDB_SET_CHANNEL_secontary arrive
        //Frame 5   set PAN ID to 0A
        for( i=0; i<=9; i++){
        putc(APP_CNF_BDB_SET_CHANNEL_primary[i]);
        }

        //=========================================================================
      while(idframe != 2){    //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_secontary from cc2530

        }
        idframe=10;

        for( i=0; i<=9; i++){
          putc(SYS_OSAL_NV_WRITE_ZDO[i]);
        }
        //=========================================================================
        while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530


        }
        idframe=10;

        for( i=0; i<=5; i++){
          putc(APP_CNF_BDB_START_COMMISSIONING_steering[i]);
        }
        delay_ms(1500); //perimene na er8oun ta adiafora frames
        idframe=10;

        for( i=0; i<=15; i++){
          putc(ZB_APP_REGISTER_REQUEST[i]);
        }
        delay_ms(1500); //perimene na er8oun ta adiafora frames
        idframe=10;
        //Do something when ZB_START_CONFIRM arrive
        //Coordinator initialization completed. Go to main

        for( i=0; i<=9; i++){
          putc(SYS_OSAL_NV_WRITE_START_UP_OPTION_OxOO[i]);
```

```
  }
  //=========================================================================

  while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
    if(idframe==100){
      error();
      goto error_retry;
    }
  }
  idframe=10; //reset idframe

  //INDICATION INIT FINISHED
  for( i=0; i<=6; i++){
    putc(SYS_GPIO_led_on[i]);
  }
  delay_ms(3000);
      //set led on until setup cc2530 finish
  for( i=0; i<=6; i++){
    putc(SYS_GPIO_led_off[i]);
  }
  delay_ms(20);idframe=10;
}
else{

  while(idframe != 0){    //wait for reset IND from cc2530. It takes 1 minute
    if(idframe==100){
      error();
      goto error_retry;
    }
  }
  //Do something when reset occurred
  //printf("reset occurred");
  //Frame 1 Frame SYS_OSAL_NV_WRITE   START_UP_OPTION
  idframe=10; //reset idframe
  for( i=0; i<=9; i++){
    putc(SYS_OSAL_NV_WRITE_START_UP_OPTION_OxOO[i]);
  }
  //=========================================================================
  while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
    if(idframe==100){
      error();
      goto error_retry;
    }
  }
  idframe=10; //reset idframe
  //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
  //Frame 2 Frame SYS_RESET   Reset cc2530 to clear NV
  for( i=0; i<=5; i++){
  putc(SYS_RESET[i]);
  }

  //=========================================================================
```

```
  while(idframe != 0){    //wait for reset from cc2530
    if(idframe==100){
      error();
      goto error_retry;
    }
  }
  idframe=10; //reset idframe
      //=========================================================================
  //setup indication led
  for( i=0; i<=6; i++){
    putc(SYS_GPIO_direction[i]);
  }

  delay_ms(500);
  idframe=10;

  for( i=0; i<=5; i++){
    putc(APP_CNF_BDB_START_COMMISSIONING_steering[i]);
  }
  delay_ms(1500); //perimene na er8oun ta adiafora frames
  idframe=10;

  for( i=0; i<=15; i++){
    putc(ZB_APP_REGISTER_REQUEST[i]);
  }
  delay_ms(1500); //perimene na er8oun ta adiafora frames
  idframe=10;
  //Do something when ZB_START_CONFIRM arrive
  //Coordinator initialization completed. Go to main

  for( i=0; i<=9; i++){
    putc(SYS_OSAL_NV_WRITE_START_UP_OPTION_OxOO[i]);
  }
  //=========================================================================
  while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
    if(idframe==100){
      error();
      goto error_retry;
    }
  }
  idframe=10; //reset idframe

  //INDICATION INIT FINISHED
  for( i=0; i<=6; i++){
    putc(SYS_GPIO_led_on[i]);
  }
  delay_ms(3000);
      //set led on until setup cc2530 finish
  for( i=0; i<=6; i++){
    putc(SYS_GPIO_led_off[i]);
  }
  delay_ms(20);idframe=10;
}
```

```
}

void recognizeMsg(void){

switch (MsgRe[2]){

   case cmd0_SRSP_SYS_RESET:
      if(MsgRe[3] == cmd1_SRSP_SYS_RESET){
      idframe=0;   }else{idframe=100;}
   break;

   case cmd0_SRSP_SYS_NV_OSAL_WRITE:
      if(MsgRe[3] == cmd1_SRSP_SYS_NV_OSAL_WRITE){
      idframe=1;   }else{idframe=100;}
   break;

   case cmd0_SRSP_APP_CNF_BDB_SET_CHANNEL:
      if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_SET_CHANNEL){idframe=2;}
      else if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_START_COMMISSIONING) {idframe=3;}
      else {idframe=100;}
   break;

   case cmd0_SRSP_ZB_APP_REGISTER_REQUEST:
      if(MsgRe[3] == cmd1_SRSP_ZB_APP_REGISTER_REQUEST){
      idframe=4;   }else{idframe=100;}
   break;

   case cmd0_ZB_RECEIVE_DATA_INDICATION:
      if(MsgRe[3] == cmd1_ZB_RECEIVE_DATA_INDICATION){
      idframe=5;   }else{idframe=100;}
   break;

default:idframe=100;    //unknown frame,with start byte FE. Call error() or move forward

}
}

void error(void){
   printf("error");
   idframe=10; //reset idframe
}
```

ΠΑΡΑΡΤΗΜΑ Γ: Πηγαίος Κώδικας ROUTER

```
#include <main.h>
#include "ConstantFrames.h"
#include "Globals.h"
#include "command_field.h"

#use rs232(baud=115200,parity=N,xmit=PIN_A4,rcv=PIN_A5,bits=8)

//dilvsi sunartisevn
void init(void);
void init_router(void);
void RA_isr(void);
void RDA_isr(void);
void read_General_format_frame(void);
void showMsg(void);
void recognizeMsg(void);
void error(void);

int1 test=0;//bit

void main(void){

   init();
   init_router();  //Sends initialization frames for router

   while(1){

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

      for(int i=0; i<=15; i++){
      putc(ZB_SEND_DATA_REQUEST_toogle[i]);
      }

     output_high(PIN_A0);
     delay_ms(3000);
     output_low(PIN_A0);
     enable_interrupts(INT_RA2);
   }
   }
}

#INT_RDA
void RDA_isr(void){

      disable_interrupts(GLOBAL);

   MsgRe[0] = getc();
   if(MsgRe[0] == SOF){    //Frame with no FE for start byte reject
   MsgRe[1] = getc();  //len
   MsgRe[2] = getc();  //cmd0
   MsgRe[3] = getc();  //cmd1
```

```
   read_General_format_frame();
   MsgRe[MsgRe[1]+4] = getc();  //FCS
   //showMsg();
   recognizeMsg();
   enable_interrupts(GLOBAL);
   }
   else{
   enable_interrupts(GLOBAL);
   }
}

#INT_RA
void RA_isr(void){

   test=1;
   disable_interrupts(INT_RA2);
   clear_interrupt(INT_RA2);   //********************
}
void read_General_format_frame(void){

   for(int i=4; i<=MsgRe[1]+3; i++){
      MsgRe[i] = getc();
   }
}

void showMsg(void){

   for(int i=0; i<=MsgRe[1]+4; i++){
      putc(MsgRe[i]);
   }
}

void init(void){
      //init MsgRe elements to 0. Array for incoming frames
      for( int i=0; i<40; i++){
      MsgRe[i]=0;
      }

   enable_interrupts(INT_RDA);
   enable_interrupts(INT_RA2);
   enable_interrupts(GLOBAL);
   set_tris_a(0xBB); //0b10111011 MSB ... LSB
   output_low(PIN_A0);
}

void init_router(void){
  int i;
      error_retry:
      while(idframe != 0){    //wait for reset IND from cc2530. It takes 1 minute
         if(idframe==100){
            error();
            goto error_retry;
         }
```

```
    }
    //Do something when reset occurred
    //printf("reset occurred");
    //Frame 1 Frame SYS_OSAL_NV_WRITE   START_UP_OPTION
    idframe=10; //reset idframe
    for( i=0; i<=9; i++){
       putc(SYS_OSAL_NV_WRITE_START_UP_OPTION[i]);
    }
    //=========================================================================
    while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
       if(idframe==100){
          error();
          goto error_retry;
       }
    }
    idframe=10; //reset idframe
    //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
    //Frame 2 Frame SYS_RESET   Reset cc2530 to clear NV
    for( i=0; i<=5; i++){
    putc(SYS_RESET[i]);
    }
    //=========================================================================
    while(idframe != 0){    //wait for reset from cc2530
       if(idframe==100){
          error();
          goto error_retry;
       }
    }
    idframe=10; //reset idframe
    //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
    //Frame 3 Frame SYS_OSAL_NV_WRITE  build cc2530 as router
    for( i=0; i<=9; i++){
    putc(SYS_OSAL_NV_WRITE_R[i]);
    }
    //=========================================================================
    while(idframe != 1){    //wait for SRSP_SYS_NV_OSAL_WRITE from cc2530
       if(idframe==100){
          error();
          goto error_retry;
       }
    }
    idframe=10; //reset idframe
    //Do something when SRSP_SYS_NV_OSAL_WRITE arrive
    //Frame 4   set channel to 13
    for( i=0; i<=9; i++){
    putc(APP_CNF_BDB_SET_CHANNEL_secontary[i]);
    }
    //=========================================================================
       while(idframe != 2){     //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_secontary from
cc2530
    }
    idframe=10; //reset idframe
    //Do something when SRSP_APP_CNF_BDB_SET_CHANNEL_secontary arrive
```

```
      //Frame 5   set PAN ID to 0A
      for( i=0; i<=9; i++){
      putc(APP_CNF_BDB_SET_CHANNEL_primary[i]);
      }

      //=========================================================================
  while(idframe != 2){    //wait for SRSP_APP_CNF_BDB_SET_CHANNEL_secontary from cc2530
      }
      idframe=10;


      for( i=0; i<=5; i++){
         putc(APP_CNF_BDB_START_COMMISSIONING_steering[i]);
      }
      delay_ms(1500); //perimene na er8oun ta adiafora frames
      idframe=10;

      for( i=0; i<=15; i++){
         putc(ZB_APP_REGISTER_REQUEST[i]);
      }
      delay_ms(1500); //perimene na er8oun ta adiafora frames
      idframe=10;
      //Do something when ZB_START_CONFIRM arrive
      //Coordinator initialization completed. Go to main
}

void recognizeMsg(void){

switch (MsgRe[2]){

   case cmd0_SRSP_SYS_RESET:
      if(MsgRe[3] == cmd1_SRSP_SYS_RESET){
      idframe=0;   }else{idframe=100;}
   break;

   case cmd0_SRSP_SYS_NV_OSAL_WRITE:
      if(MsgRe[3] == cmd1_SRSP_SYS_NV_OSAL_WRITE){
      idframe=1;   }else{idframe=100;}
   break;

   case cmd0_SRSP_APP_CNF_BDB_SET_CHANNEL:
      if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_SET_CHANNEL){idframe=2;}
      else if(MsgRe[3] == cmd1_SRSP_APP_CNF_BDB_START_COMMISSIONING) {idframe=3;}
      else {idframe=100;}
   break;

   case cmd0_SRSP_ZB_APP_REGISTER_REQUEST:
      if(MsgRe[3] == cmd1_SRSP_ZB_APP_REGISTER_REQUEST){
      idframe=4;   }else{idframe=100;}
   break;

   case cmd0_ZB_RECEIVE_DATA_INDICATION:
      if(MsgRe[3] == cmd1_ZB_RECEIVE_DATA_INDICATION){
```

```
        idframe=5;   }else{idframe=100;}
    break;

default:idframe=100;    //unknown frame,with start byte FE. Call error() or move forward

}
}

void error(void){

    printf("error");
    idframe=10; //reset idframe

}
```