

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τμήμα Πληροφορικής



Πτυχιακή Εργασία

Υλοποίηση Αλγορίθμων Πιθανοτικής Πλημμυρίδας σε
Ασύρματα Δίκτυα

Ανδρεάνα Στυλίδου

Επιβλέπων: Αν. Καθηγητής Κωνσταντίνος Οικονόμου

26 Σεπτεμβρίου 2019

Επιβλέπων Καθηγητής

Κωνσταντίνος Οικονόμου, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Κωνσταντίνος Οικονόμου, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Δημήτριος Τσουμάκος, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Μιχαήλ Στεφανιδάκης, Επίκουρος Καθηγητής,
Ιόνιο Πανεπιστήμιο

Αφιερώνω την εργασία μου στην οικογένεια μου και στους φίλους μου,
και τους ευχαριστώ για την πολύτιμη υποστήριξη που μου παρείχαν, καθ'
όλη τη διάρκεια των σπουδών μου.

Περίληψη

Η ανάγκη για καθολική διάδοση πληροφορίας στα δίκτυα με αποδοτικούς μηχανισμούς καθίσταται επιτακτική στα σύγχρονα ασύρματα περιβάλλοντα. Οι παραδοσιακοί μέθοδοι και αλγόριθμοι δεν θεωρούνται πλέον κατάλληλοι να αντεπεξέλθουν στις αυξημένες ενεργειακές απαιτήσεις και στον μεγάλο αριθμό των κόμβων σε αυτού του είδους τα δίκτυα.

Συνεπώς σήμερα, μελετούνται συστηματικά και υιοθετούνται παραλλαγές γνωστών αλγορίθμων, ένας από τους οποίους είναι ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας, ο οποίος περιορίζοντας το πλήθος των μεταδιδόμενων μηνυμάτων, προσφέρει μείωση της συμφόρησης του δικτύου, έναντι όμως της αβεβαιότητας για τη πλήρη κάλυψή του. Η μελέτη της Πιθανοτικής Πλημμυρίδας δημιουργεί ενδιαφέροντα ερωτήματα όπως, ποια είναι η βέλτιστη πιθανότητα αποστολής για την επίτευξη της, κατά το δυνατόν, μεγαλύτερης κάλυψης δικτύου και με τη χρήση των λιγότερων δυνατών μεταδόσεων.

Για τους σκοπούς της παρούσας εργασίας, υλοποιείται ένα ασύρματο δίκτυο χαμηλού κόστους το οποίο βασίζεται στην ανοικτού κώδικα πλατφόρμα Arduino και τις συσκευές Xbee του πρωτοκόλλου επικοινωνίας ZigBee, με στόχο την πειραματική προσέγγιση της Πιθανοτικής Πλημμυρίδας, μέσω της πρακτικής υλοποίησης αντίστοιχης εφαρμογής. Ακολουθεί έπειτα η παρουσίαση των αποτελεσμάτων, η ποιοτική τους μελέτη και γίνεται σύγκριση με τη μέχρι τώρα θεωρία.

Abstract

The need for global information dissemination across networks with efficient mechanisms is becoming imperative in modern wireless environments. Traditional methods and algorithms are no longer considered to be able to withstand the increasing energy requirements and the large number of nodes in these types of networks.

Therefore, variants of known algorithms are systematically being studied and adopted, one of which is the Probability Flooding algorithm, which reduces the number of transmitted messages and helps with congestion in the network, at the cost of the uncertainty regarding the network's full coverage. The study of Probabilistic Flooding yields interesting questions such as, what is the optimal forwarding probability so that using the least possible number of transmission the maximum coverage in network can be achieved.

For the purposes of this work, a low-cost wireless network is being implemented, based on the Open Source Platform Arduino and using the Xbee modules of the ZigBee communication protocol with the aim of experimenting with the Probabilistic Flooding, through the implementation of a corresponding application. At the end of this work follows the presentation of the results, as well as their qualitative study and a comparison with the present theory.

Ευχαριστίες

Η υλοποιηθείσα εφαρμογή της παρούσας πτυχιακής εργασίας, υποστηρίχθηκε από το έργο ' Πιλοτικό Σύστημα Ασύρματου Δικτύου Αισθητήρων Συγχρονισμένης Παρακολούθησης Κλιματολογικών και Εδαφολογικών Παραμέτρων σε Ελαιώνες ' (Κωδικός MIS: 5007309), το οποίο χρηματοδοτήθηκε από το ΕΣΠΑ (ΠΕΠ Ιονίων Νήσων 2014-2020).

Περιεχόμενα

A´	Εισαγωγή	1
A´.1	Ασύρματα Δίκτυα	2
A´.2	Τύποι Ασύρματης Μετάδοσης	3
A´.3	Ασύρματα Δίκτυα Προσωπικής Περιοχής και Εφαρμογές	4
A´.3.1	RFID	5
A´.3.2	Κινητά Κατά Περίπτωση Δίκτυα - MANET	5
A´.3.3	Ασύρματα Δίκτυα Αισθητήρων	7
A´.4	Βιβλιογραφική Ανασκόπηση	8
A´.5	Δομή της Πτυχιακής Εργασίας	11
B´	Πλημμυρίδα και Πιθανοτική Πλημμυρίδα	13
B´.1	Πλημμυρίδα	13
B´.1.1	Απόδοση Πλημμυρίδας	15
B´.2	Πιθανοτική Πλημμυρίδα	16
Γ´	Πρωτόκολλα Συστήματος	18
Γ´.1	IEEE 802.15.4	18
Γ´.1.1	Τύποι Συσκευών που Υποστηρίζονται	18
Γ´.1.2	Τοπολογίες	19
Γ´.1.3	Τα Επίπεδα του Προτύπου	19
Γ´.2	Τεχνολογία ZigBee	20
Γ´.2.1	Συσκευές ZigBee	21
Γ´.2.2	Τοπολογίες	23
Γ´.2.3	Επίπεδο Δικτύου	25
Γ´.2.4	Επίπεδο Εφαρμογών	25

Δ΄	Υλοποίηση του Δικτύου	26
Δ΄.1	Εξοπλισμός και Περιβάλλον Υλοποίησης	26
Δ΄.1.1	Μικροελεγκτής Arduino	26
Δ΄.1.2	Digi Xbee Pro S2C - ZigBee RF Module	27
Δ΄.1.3	Λειτουργία Transparent	30
Δ΄.1.4	Λειτουργία API	30
Δ΄.1.5	Παράμετροι Συστήματος (Configurations)	32
Δ΄.2	Ανάπτυξη Εφαρμογής της Πιθανοτικής Πλημμυρίδας	35
Δ΄.2.1	Γλώσσα Arduino	35
Δ΄.2.2	Δομή sketch	36
Δ΄.2.3	Βιβλιοθήκη MyFrame	36
Δ΄.2.4	Περιγραφή Προγράμματος	38
Δ΄.3	Πειράματα	47
Δ΄.3.1	Διαγράμματα	49
Ε΄	Σύνοψη και Συμπεράσματα	57
	Βιβλιογραφία	58
	Παραρτήματα	64
I	Κώδικας Εφαρμογής Πλημμυρίδας	65
II	Κώδικας Βιβλιοθήκης MyFrame.h	83

Κατάλογος Σχημάτων

B'.1	Μετάδοση πληροφορίας με τον αλγόριθμο της Πλημμυρίδας. Τα βέλη του σχήματος απεικονίζουν τις μεταδόσεις, ενώ οι ευθείες τις ζεύξεις μεταξύ των κόμβων.	14
Γ'.1	Στοίβα πρωτοκόλλων ZigBee και 802.15.4	22
Γ'.2	Τοπολογία αστέρα	24
Γ'.3	Τοπολογία δέντρου	24
Γ'.4	Τοπολογία πλέγματος	25
Δ'.1	Arduino Uno	27
Δ'.2	Xbee Pro S2C Module	29
Δ'.3	Μορφή εντολής AT. Η παράμετρος που αναλαμβάνει να τροποποιήσει η εντολή του σχήματος είναι το ID και η νέα τιμή της παραμέτρου το 2015	30
Δ'.4	Σχηματική αναπαράσταση της επικοινωνίας μεταξύ δυο μονάδων Xbee σε λειτουργία API.	31
Δ'.5	Μονάδα Xbee τοποθετημένη επάνω σε Xbee Shield για τη διασύνδεση της μονάδας Xbee με το Arduino Uno	32
Δ'.6	Εμφάνιση των χαρακτηριστικών του κόμβου συντονιστή με Node Identifier 0 μέσω του περιβάλλοντος XCTU (α)	33
Δ'.7	Εμφάνιση των χαρακτηριστικών του κόμβου συντονιστή με Node Identifier 0 μέσω του περιβάλλοντος XCTU (β)	34
Δ'.8	Εμφάνιση των χαρακτηριστικών του κόμβου δρομολογητή με Node Identifier 1 μέσω του περιβάλλοντος XCTU (α)	34
Δ'.9	Εμφάνιση των χαρακτηριστικών του κόμβου δρομολογητή με Node Identifier 1 μέσω του περιβάλλοντος XCTU (α)	35

- Δ'.10 Παράδειγμα προγράμματος σε γλώσσα Arduino με το οποίο αναβοσβήνει περιοδικά μια λάμπα συνδεδεμένη στην πλακέτα. Στο μέρος `setup()` ορίζεται ο ακροδέκτης για μια λάμπα τύπου LED, ενώ στο `loop()` η λειτουργικότητα της λάμπας για όσον η πλακέτα παραμένει σε κατάσταση τροφοδοσίας. 37
- Δ'.11 Η λογική του συγχρονισμού που υλοποιείται. Ο κόμβος Co είναι ο συντονιστής. Πραγματοποιείται η αποστολή του μηνύματος SYNC από τον κόμβο Co στον κόμβο C, με σκοπό ο κόμβος C να αποθηκεύσει τη χρονική διαφορά Dt του ρολογιού του από το ρολόι του Co. Σε μεταγενέστερη φάση, ο κόμβος C λαμβάνει μήνυμα τύπου FLOOD από γειτονικό του κόμβο B. Για να βρει τον χρόνο παραλαβής του FLOOD σύμφωνα με το ρολόι του συντονιστή, αφαιρεί τη χρονική διαφορά Dt από τον χρόνο παραλαβής της πλημμυρίδας. Η πληροφορία αυτή αποστέλλεται με μήνυμα τύπου RESULT, κατευθείαν στον συντονιστή. 44
- Δ'.12 Παράδειγμα μετάδοσης της Πιθανοτικής Πλημμυρίδας σε ένα δίκτυο μεγέθους 5 κόμβων, με αριθμό επαναλήψεων πλημμυρίδας $N = 2$. Οι μεταδόσεις των πλαisiών της κάθε διαφορετικής επανάληψης απεικονίζονται με ξεχωριστό χρώμα. Ο κόμβος 0 είναι ο συντονιστής, ο οποίος εκκινεί τη διαδικασία. Με τα περιγραμμισμένα βέλη απεικονίζονται οι μεταδόσεις των πλαisiών της Πλημμυρίδας. Με τα βέλη χωρίς τα περιγράμματα απεικονίζονται τα πλαίσια απάντησης των κόμβων. Κάθε κόμβος που λαμβάνει μια διαφορετική Πλημμυρίδα, οφείλει να απαντήσει στον συντονιστή. 47
- Δ'.13 Τοπολογία διεξαγωγής πειραμάτων για δίκτυο μεγέθους 10 κόμβων. Ο κόμβος 0 έχει τον ρόλο του συντονιστή του δικτύου. 48
- Δ'.14 Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1$ (Σίγουρο ενδεχόμενο). Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος. 50
- Δ'.15 Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/2$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος. 50

Δ'.16H διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/3$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.	51
Δ'.17H διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/4$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.	51
Δ'.18H διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/5$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.	52
Δ'.19H διάδοση της Πλημμυρίδας για όλες τις τιμές Πιθανοτήτων των πειράματων. Διάγραμμα (α): Κάλυψη του δικτύου συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου συναρτήσει των βημάτων (hops) του πειράματος.	53
Δ'.20Κάλυψη του δικτύου C_m συναρτήσει της πιθανότητας προώθησης p . . .	54
Δ'.21Χρόνοι τερματισμού του πειράματος συναρτήσει της πιθανότητας προώθησης p	55

Κατάλογος Πινάκων

Δ'.1 Χαρακτηριστικά του μικροελεγκτή <i>Arduino Uno</i>	28
Δ'.2 Χαρακτηριστικά του <i>Xbee Pro S2C Module</i>	28
Δ'.3 Πίνακας πλαισίου <i>_frame</i> της κλάσης <i>myFrame</i> . Οι πρώτες τρεις θέσεις του πίνακα περιέχουν τον τύπο του πλαισίου, το αναγνωριστικό και το μήκος των δεδομένων αντίστοιχα. Το μεγαλύτερο δυνατό μέγεθος πλαισίου σε <i>bytes</i> που μπορεί να υποστηρίξει το <i>ZigBee</i> πρωτόκολλο είναι 255 και έτσι το μέγιστο μέγεθος των δεδομένων του πλαισίου <i>myFrame</i> ανέρχεται στα $255 - 3 = 252$ <i>bytes</i>	38
Δ'.4 Τιμές των πιθανοτήτων των πειραμάτων	48
Δ'.5 Μορφή δεδομένων των αποτελεσμάτων πειράματος	49

Κεφάλαιο Α΄

Εισαγωγή

Τα Δίκτυα Υπολογιστών συντέλεσαν σπουδαίο ρόλο στην εξέλιξη της σημερινής εποχής στην Εποχή της Πληροφορίας. Τα έξυπνα κινητά τηλέφωνα καθώς και άλλου είδους φορητές συσκευές επέτρεψαν την αποδέσμευση του ανθρώπου από την καλωδιακή επικοινωνία, και συνεπώς εξαπλώθηκαν με τεράστιους ρυθμούς, σε σημείο η ύπαρξή τους να θεωρείται πλέον δεδομένη. Οι φορητές αυτές συσκευές κάνουν έντονη χρήση των σύγχρονων ασύρματων δικτύων και επομένως παρουσιάζεται η ανάγκη εύρεσης ενεργειακά αποδοτικών αλγορίθμων με σκοπό την εξοικονόμηση πολύτιμων πόρων.

Στην παρούσα εργασία αναλύεται ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας, ο οποίος αποτελεί μια παραλλαγή του αλγορίθμου της Πλημμυρίδας. Σκοπός της Πλημμυρίδας είναι να επιτευχθεί η καθολική κάλυψη ενός δικτύου, η πληροφορία δηλαδή που υπάρχει σε έναν κόμβο να διαδοθεί σε όλους τους υπόλοιπους. Η Πιθανοτική Πλημμυρίδα έρχεται να καλύψει το χάσμα ανάμεσα στη διάχυση πληροφορίας σε περιβάλλοντα όπου δεν υπάρχει επαρκής αρχική γνώση (τοπολογία, αριθμός κόμβων), και στον μεγάλο αριθμό των μεταδιδόμενων μηνυμάτων ανάμεσα στους κόμβους που προκύπτει από τον αλγόριθμο της Πλημμυρίδας.

Για τους σκοπούς της εργασίας ο αλγόριθμος υλοποιείται σε μια πραγματική εφαρμογή. Οι μονάδες Xbee αποτελούν τους κόμβους του δικτύου και ενσωματώνονται σε μικροελεγκτές Arduino, οι οποίοι προγραμματίζονται κατάλληλα και εκτελούν τον συγκεκριμένο αλγόριθμο. Τα αποτελέσματα των πειραμάτων για διαφορετικές τιμές της πιθανότητας αποστολής παρουσιάζονται σε μια προσπάθεια μελέτης τους και σύγκρισης με την ως τώρα θεωρία.

Α΄.1 Ασύρματα Δίκτυα

Τα δίκτυα, ανάλογα με το φυσικό μέσο μετάδοσης χαρακτηρίζονται ως ενσύρματα ή ασύρματα.

Στα ασύρματα δίκτυα η επικοινωνία πραγματοποιείται με τη χρήση ηλεκτρομαγνητικών κυμάτων χαμηλής συχνότητας, και συγκεκριμένα τα ραδιοκύματα, ως φορείς πληροφορίας. Με τον όρο ηλεκτρομαγνητικό κύμα εννοούμε τον συνδυασμό ενός ηλεκτρικού και ενός μαγνητικού πεδίου, που είναι κάθετα μεταξύ τους και διαδίδεται στο κενό και στα μονωτικά σώματα, κάθετα στις διευθύνσεις των πεδίων και με την ταχύτητα του φωτός. Μπορεί να περιγραφεί από τη συχνότητά του f και το μήκος κύματος λ . [1]

Το φάσμα των ηλεκτρομαγνητικών κυμάτων καλύπτει τις συχνότητες από 3 KHz μέχρι 300 GHz. Αποτελείται από περιοχές συχνοτήτων, ή αλλιώς ζώνες, κάθε μία από τις οποίες είναι 10 φορές μεγαλύτερη από την προηγούμενή της. Κατά κύριο λόγο τις χωρίζουμε σε δύο μεγάλες κατηγορίες: τις ζώνες των χαμηλών και τις ζώνες των υψηλών συχνοτήτων. Η πρώτη περιλαμβάνει τα ραδιοκύματα, τα μικροκύματα, την υπέρυθη ακτινοβολία και το ορατό φως, ενώ η δεύτερη την υπεριώδη ακτινοβολία, τις ακτίνες χ και τις ακτίνες γ . Λόγω των μεγάλων ποσών ενέργειας που μεταφέρουν, οι υψηλές συχνότητες των κυμάτων της δεύτερης κατηγορίας αν και θεωρητικά θα ήταν καλύτερες για τη μετάδοση πληροφορίας, ανακλώνται ευκολότερα από φυσικά εμπόδια, και γενικώς δεν χρησιμοποιούνται, καθώς μπορούν να αποβούν εξαιρετικά επιβλαβείς για τους ζωντανούς οργανισμούς. [2]

Τα κύματα που προορίζονται για την επικοινωνία στα Ασύρματα Δίκτυα είναι τα ραδιοκύματα ευρείας εκπομπής, που εκτείνονται περίπου από τα 30 MHz ως το 1 GHz, καθώς και τα μικροκύματα που εκτείνονται από τα 2 GHz ως τα 40 GHz. Μερικά από τα μεγαλύτερα πλεονεκτήματα των ραδιοκυμάτων είναι η ευκολία παραγωγής τους, η δυνατότητα τους να διαδοθούν μέσα από συμπαγή αντικείμενα καθώς και η ικανότητα τους να εκτείνονται σε μια αρκετά μεγάλη περιοχή, από ένα εκατοστό έως δεκάδες και εκατοντάδες μέτρα. [3]

Ειδικές συσκευές που ονομάζονται κεραίες, παράγουν τα ραδιοκύματα μετατρέποντας το υψίσυχνο ρεύμα στον πομπό σε ηλεκτρομαγνητική ακτινοβολία, την οποία εκπέμπουν στο περιβάλλον (ατμόσφαιρα, διάστημα, νερό). Για τη λήψη του σήματος απαιτείται, αντίστοιχα, η μετατροπή της ηλεκτρομαγνητικής ακτινοβολίας που λαμβάνεται από την κεραία σε ρεύμα και η προώθηση της στον δέκτη. Από τη φύση τους τα

κύματα αυτά είναι μη κατευθυνόμενα, διαδίδονται δηλαδή από την πηγή προς όλες τις κατευθύνσεις (omnidirectional), και έτσι δεν χρειάζεται να ευθυγραμμίζεται ο πομπός (transmitter) με τον δέκτη (receiver). Μπορούμε ωστόσο να προσανατολίσουμε τα ραδιοκύματα χρησιμοποιώντας διάφορες τεχνικές, τόσο για την ενίσχυσή τους, όσο και για την αποφυγή της μετάδοσης τους προς ανεπιθύμητα σημεία.

Α΄.2 Τύποι Ασύρματης Μετάδοσης

Η κεραία μπορεί να μεταδώσει ραδιοκύματα, τα οποία διαδίδονται με έναν από τους ακόλουθους τρεις τρόπους[3] :

- Μετάδοση μέσω επιγείων κυμάτων

Τα κύματα ακολουθούν το περίγραμμα του εδάφους της Γης και είναι ικανά να διαδοθούν σε σημαντικές αποστάσεις πέρα από την ευθεία οπτικής επαφής. Το φαινόμενο αυτής της διάδοσης εμφανίζεται κατά κύριο λόγο για συχνότητες έως 2 MHz. Η τάση του ηλεκτρομαγνητικού κύματος να ακολουθήσει την καμπυλότητα της Γης εξηγείται χάρη σε παράγοντες μεταξύ των οποίων είναι: η ηλεκτρομαγνητική ακτινοβολία προκαλεί ένα ρεύμα στην επιφάνεια της Γης, το οποίο έχει σαν αποτέλεσμα την επιβράδυνση του μετώπου κύματος δίπλα στη Γη και το κύμα παίρνει την κλίση της επιφάνειας της Γης, και η περίθλαση του ηλεκτρομαγνητικού κύματος η οποία προκαλείται κατά τη συνάντηση εμποδίων. Τα κύματα του συγκεκριμένου εύρους συχνοτήτων είναι διασκορπισμένα στην ατμόσφαιρα με τέτοιο τρόπο που δεν μπορούν να εισβάλουν στα ανώτερα στρώματά της. Ένα παράδειγμα επικοινωνίας με τη χρήση κυμάτων εδάφους αποτελεί το ράδιο AM.

- Μετάδοση μέσω εναέριων κυμάτων

Χρησιμοποιείται σε επικοινωνίες CB και σε διεθνείς εκπομπές τύπου BBC για συχνότητες κυμάτων από 2 MHz έως 30 MHz. Σε αυτή τη περίπτωση μετάδοσης, το σήμα που εκπέμπεται από κάποια κεραία ανακλάται από το ιονισμένο στρώμα της ανώτερης ατμόσφαιρας και ύστερα από την επιφάνεια της Γης πολλαπλές φορές (χάρη στο φαινόμενο της διάθλασης) μέχρι να φτάσει στον δέκτη. Με αυτό το είδος διάδοσης, είναι δυνατή η κάλυψη αποστάσεων χιλιάδων χιλιομέτρων από

τον πομπό.

- Μετάδοση μέσω Οπτικής Επαφής (LoS)

Χρησιμοποιείται για συχνότητες μεγαλύτερες των 30 MHz. Τα σήματα που ξεπερνούν αυτό το εύρος συχνοτήτων δεν ανακλώνται από την ιονόσφαιρα και έτσι μπορούν να διαδοθούν μεταξύ δορυφόρων και επίγειων σταθμών. Όσον αφορά την επικοινωνία στην επιφάνεια της Γης, οι κεραίες του πομπού και του δέκτη θα πρέπει να βρίσκονται σε ικανοποιητική απόσταση οπτικής επαφής μεταξύ τους, για να είναι δυνατή η επίτευξη μιας αποτελεσματικής επικοινωνίας. Σημειώνεται, πως με τον όρο ικανοποιητική εννοείται η απόσταση κατά την οποία δεν προλαβαίνει να εμφανιστεί η εγγενής ιδιότητα των ραδιοκυμάτων να κλίνουν με την καμπυλότητα της Γης.

Α΄.3 Ασύρματα Δίκτυα Προσωπικής Περιοχής και Εφαρμογές

Ανάλογα με το εύρος της γεωγραφικής κάλυψης, τα δίκτυα χαρακτηρίζονται ως δίκτυα ευρείας περιοχής (WAN), μητροπολιτικά (MAN), τοπικά (LAN) και δίκτυα προσωπικής περιοχής (PAN). Η παρούσα εργασία επικεντρώνεται στα δίκτυα τύπου PAN και συγκεκριμένα στα ασύρματα WPAN.

Τα Ασύρματα Δίκτυα Προσωπικής Περιοχής WPAN γίνονται ολοένα και περισσότερο πεδίο σημαντικού ενδιαφέροντος στην εποχή μας. Η οικογένεια πρωτοκόλλων IEEE 802.15 αποτελείται από αναδυόμενα πρότυπα ασύρματης επικοινωνίας του χώρου αυτού των δικτύων. Πρόκειται για μια ομάδα προτύπων που αναπτύσσονται από την αντίστοιχη ομάδα εργασίας IEEE 802.15 [4] και στοχεύουν στη διασύνδεση συσκευών που βρίσκονται σε αποστάσεις μικρής εμβέλειας μεταξύ τους, έναντι χαμηλού κόστους επικοινωνίας. Γνωστά πρότυπα μεταξύ αυτών είναι το Bluetooth (802.15.1), που στοχεύει στη διασύνδεση φορητών συσκευών, όπως κινητά τηλέφωνα, εκτυπωτές και υπολογιστές, καθώς και το ZigBee (802.15.4) ένα πρότυπο ιδανικό για την υλοποίηση δικτύων αισθητήρων με σκοπό τη διαχείριση οικιακών συστημάτων, όπως κλιματισμού, θέρμανσης και φωτισμού, με σκοπό την αυτοματοποίηση χώρου.

Α'.3.1 RFID

Ένα ενδιαφέρον παράδειγμα δικτύου WPAN είναι η τεχνολογία RFID (Radio Frequency Identification), η οποία επιτρέπει την ασύρματη αυτόματη αναγνώριση ετικετών αντικειμένων ή ακόμα και ανθρώπων. Μπορεί να θεωρηθεί ως η τεχνολογική εξέλιξη των παραδοσιακών ετικετών ραβδωτού κώδικα (barcode).

Ένα RFID σύστημα αποτελείται από τις ετικέτες (μικροτσιπ με κεραίες) και από έναν αναγνώστη RFID Reader με τη δική του κεραία. Όταν η κεραία μιας RFID ετικέτας, λάβει ραδιοκύματα από την κεραία του αναγνώστη, διεγείρεται και στέλνει στον αναγνώστη μία απάντηση. Με τον τρόπο αυτόν, μια RFID ετικέτα μπορεί να αναγνωριστεί από τον αντίστοιχο αναγνώστη του RFID συστήματος.

Η RFID ετικέτα μπορεί να έχει τύπο Read Only, Read - Write ή Write Once - Read Many. Οι ετικέτες πρώτου τύπου εγγράφονται μία φορά στο εργοστάσιο και δεν έχουν τη δυνατότητα επανεγγραφής. Οι ετικέτες δεύτερου τύπου είναι δυνατών να επανεγγραφούν από έναν αναγνώστη που φέρει τα κατάλληλα δικαιώματα. Οι τρίτου τύπου ετικέτες επιτρέπουν την εγγραφή πληροφοριών από τον αναγνώστη για μια μόνο φορά χωρίς να είναι δυνατή η μελλοντική τροποποίηση των πληροφοριών της ετικέτας.

Α'.3.2 Κινητά Κατά Περίπτωση Δίκτυα - MANET

Ένα κατανεμημένο δίκτυο ονομάζεται το δίκτυο του οποίου όλοι οι κόμβοι αποτελούν ομότιμα μέλη. Επομένως, δεν ελέγχεται από κεντρικούς κόμβους-αρχηγούς που συντονίζουν τη συμπεριφορά των υπόλοιπων κόμβων, αλλά αντιθέτως, οι αποφάσεις λαμβάνονται αυτόνομα από όλους. Οι αποφάσεις αυτές, προσδιορίζονται από τον κατανεμημένο αλγόριθμο που εκτελείται στους κόμβους του δικτύου. Ένας τέτοιος αλγόριθμος περιλαμβάνει το σύνολο των εντολών που εκτελούνται από τον κόμβο ο οποίος ξεκινάει τη διαδικασία της επικοινωνίας (initiator), και το σύνολο των εντολών που εκτελούνται από όλους τους ομότιμους κόμβους του δικτύου (συμπεριλαμβανομένου του κόμβου initiator) και επαναληπτικά, μέχρι να τερματίσει ο αλγόριθμος. Οι εντολές αυτές μπορεί να περιλαμβάνουν την αποστολή μηνυμάτων ανάμεσα στους κόμβους καθώς και την αλλαγή των καταστάσεων τους ανάλογα με τα μηνύματα-ερεθίσματα που έχουν δεχθεί. [5]

Σκοπός της αποκεντρωμένης φύσης είναι να μη περιορίζεται η δυναμικότητα του δικτύου λόγω της σχεδίασης του λογισμικού. Έτσι οποιοδήποτε κόμβος θα μπορούσε θεωρητικά να αποσυνδεθεί από το δίκτυο, χωρίς να επηρεάζεται η λειτουργικότητα του

σε επίπεδο εφαρμογής.

Τα δίκτυα MANET είναι ένα χαρακτηριστικό παράδειγμα κατανεμημένων δικτύων. Ο όρος MANET αποτελεί συντομογραφία του Mobile Ad Hoc Networks, δηλαδή των Κινητών Κατά Περίπτωση Δικτύων. Αποτελούνται από κόμβους οι οποίοι έχουν την ιδιαιτερότητα να είναι ελεύθεροι να κινηθούν στον χώρο και δεν απαιτούν την ύπαρξη κάποιας σταθερής υποδομής. Οι κόμβοι ενός δικτύου MANET μπορεί να είναι εξοπλισμένοι με κεραιές μη κατευθυντικές (omnidirectional), κατευθυντικές (point-to-point), ή να συνδυάζουν τους παραπάνω τύπους κεραιών. Λόγω της κινητικότητας των κόμβων του δικτύου, τη μη σταθερή εμβέλεια των πομποδεκτών, καθώς και τις παρεμβολές του περιβάλλοντος οι ζεύξεις μεταξύ των κόμβων των δικτύων αλλάζουν συνεχώς και οι τοπολογίες των δικτύων αυτών είναι τυχαίες για διαφορετικές χρονικές στιγμές και είναι δυνατόν να υποστούν μη-προβλέψιμες και απότομες αλλαγές (ad hoc τοπολογίες). Ένα δίκτυο MANET έχει τη δυνατότητα να λειτουργήσει αυτόνομα, είτε και να είναι μέρος ενός μεγαλύτερου δικτύου, όπως για παράδειγμα να είναι συνδεδεμένο στο διαδίκτυο.

Κάποιες από τις μεγαλύτερες δυσκολίες στον χώρο των Ad Hoc δικτύων, έχουν να κάνουν με τη δρομολόγηση. Η εύρεση του καταλληλότερου μονοπατιού ανάμεσα στους κόμβους, δυσκολεύει, καθώς οι κόμβοι μπορούν να εξέρχονται από το δίκτυο και να προκαλούν αποσυνδέσεις, λόγω της μεταβαλλόμενης δομής του δικτύου. Έτσι απαιτούνται επαναδρομολογήσεις οι οποίες υποβιβάζουν, αντίστοιχα, την ταχύτητα των εφαρμογών.

Για την επίλυση αυτών των ζητημάτων προτείνεται πλήθος πρωτοκόλλων δρομολόγησης αλλά και βελτιώσεων των γνωστών πρωτοκόλλων. Τα πρωτόκολλα αυτά χωρίζονται σε δύο βασικές κατηγορίες reactive και proactive, ενώ κάποιες φορές γίνεται χρήση υβριδικών πρωτοκόλλων (συνδυασμός και των δύο). Το χαρακτηριστικό της πρώτης κατηγορίας είναι πως οι κόμβοι αποθηκεύουν πληροφορίες για το δίκτυο, όπως είναι ο πίνακας δρομολόγησης, για να μπορούν να τις ανακτήσουν μελλοντικά για την επικοινωνία με κάποιον άλλον κόμβο του δικτύου. Αντιθέτως, στη δεύτερη κατηγορία δε διατηρούνται πληροφορίες για το δίκτυο στη μνήμη των κόμβων, με αποτέλεσμα να αναχτούν πληροφορίες για το δίκτυο μόνο σε περίπτωση που υπάρξει κάποιο αίτημα επικοινωνίας.

Α΄.3.3 Ασύρματα Δίκτυα Αισθητήρων

Πρόοδοι στην τεχνολογία των μικρο- ηλεκτρομηχανικών συστημάτων (MEMS), επέτρεψαν την εισαγωγή στην αγορά των ασύρματων δικτύων αισθητήρων. Τα δίκτυα αισθητήρων αποτελούν κατηγορία των κατανεμημένων ad hoc δικτύων και συγκροτούνται κατά κανόνα από μικρού μεγέθους ασύρματες φορητές συσκευές, κάθε μία από τις οποίες αναλαμβάνει τον ρόλο ενός κόμβου του δικτύου. Οι εν λόγω συσκευές κατανέμονται σε συγκεκριμένες περιοχές για την παρακολούθηση περιβαλλοντικών παραμέτρων των υπο εποπτεία περιοχών (πχ. η θερμοκρασία, ο ήχος, η ατμοσφαιρική πίεση και η υγρασία), την επεξεργασία των δεδομένων που συλλέγονται και τη δρομολόγηση τους στους κατάλληλους προορισμούς. Οι εφαρμογές τους μπορεί να είναι στρατιωτικές, υγείας, αυτοματισμού ή ασφάλειας χώρων. [6]

Τα δίκτυα αισθητήρων διαφοροποιούνται από τα παραδοσιακά ad hoc δίκτυα λόγω των ακόλουθων ιδιαιτεροτήτων τους:

- Ο αριθμός των κόμβων του δικτύου μπορεί να είναι αρκετά μεγάλος σε σχέση με τα παραδοσιακά ad hoc .
- Απαιτούν πολύ μεγάλες ποσότητες σε ενέργεια, αφού προορίζονται για μη επιβλεπόμενα από ανθρώπους περιβάλλοντα (πχ. καλλιέργειες) και η αντικατάσταση της μπαταρίας τους ή η επαναφόρτιση της δεν είναι πάντα άμεσα δυνατή.
- Οι περιορισμοί σε ενεργειακό επίπεδο και το μικρό εύρος ζώνης του δικτύου (bandwidth) καθιστούν απαραίτητη την υλοποίηση τεχνικών συσσώρευσης πληροφορίας (aggregation). Πλήθος πακέτων συγχωνεύονται σε ένα ενιαίο πακέτο πριν την αποστολή τους. Κατά αυτόν τον τρόπο μειώνεται ο αριθμός των συνολικών κεφαλίδων των πακέτων, οι οποίες καταναλώνουν περιττό bandwidth, καθώς επίσης μειώνεται και η καθυστέρηση που σχετίζεται με την πρόσβαση στο μέσο μετάδοσης. Επομένως, εκτελούνται διαδικασίες προεπεξεργασίας των δεδομένων στους ενδιάμεσους κόμβους πριν την προώθησή τους προς τον κόμβο προορισμού.
- Οι κόμβοι-αισθητήρες μπορεί να μην έχουν μοναδικές ταυτότητες (identifiers) και έτσι δεν είναι πάντα εφικτή η αποκλειστική διευθυνσιοδότηση [7].

Η υποδομή των Ασύρματων Δικτύων Αισθητήρων στηρίζεται στις αρχές των MANET και δεν απαιτείται η ύπαρξη μιας κεντρικής κεραίας επικοινωνίας. Έτσι τα δίκτυα

αυτά έχουν την ικανότητα να δημιουργούνται με δυναμικό τρόπο και είναι ελεύθερα να κινηθούν στον χώρο.

Τα Ασύρματα Δίκτυα Αισθητήρων αποτελούνται από ένα μεγάλο αριθμό κόμβων χαμηλού κόστους και με περιορισμένες δυνατότητες επεξεργασίας, επικοινωνίας και αποθεμάτων ενέργειας και για αυτόν τον λόγο χρειάζονται αλγόριθμους που προσαρμόζονται στο υλικό τους. Το απλούστερο πρωτόκολλο δρομολόγησης είναι η πλημμυρίδα, που εξετάζεται παρακάτω, η οποία αν και ιδιαίτερα χρήσιμη, στις περισσότερες περιπτώσεις μη αποδοτική. Ένα σημαντικό μειονέκτημα του αλγόριθμου είναι ο μεγάλος αριθμός μηνυμάτων που πλημμυρίζουν το δίκτυο, και για το λόγο αυτό προτιμούνται αλγόριθμοι δρομολόγησης πιθανοτικής πλημμυρίδας. Με αυτό το τρόπο κάθε κόμβος αποφασίζει αν θα στείλει ένα μήνυμα που έλαβε με τυχαίο τρόπο και σύμφωνα με μια πιθανότητα.[8]

Α΄.4 Βιβλιογραφική Ανασκόπηση

Η διάδοση πληροφορίας είναι αναπόσπαστο στοιχείο των βασικών λειτουργιών που εκτελούνται στα δίκτυα (δρομολόγηση, ανακάλυψη δικτύου, εύρεση βέλτιστου μονοπατιού), και ο αλγόριθμος της Πλημμυρίδας, όπως προαναφέρθηκε, αποτελεί έναν από τους κύριους αλγόριθμους διάδοσης πληροφορίας. Αυτό αποδίδεται κυρίως στην απλότητά του και στην απουσία ανάγκης για γνώση, καθώς σε πολλές περιπτώσεις δεν είναι εξ' αρχής γνωστή η τοπολογία του δικτύου και επιπλέον, η εύρεση αυτής αποτελεί ακριβή διαδικασία.

Η μέθοδος της Τυφλής Πλημμυρίδας [9] προτείνεται για χρήση στα δίκτυα στις αρχές του '80, ως ένας από τους πρώτους μηχανισμούς διάχυσης πληροφορίας. Προτείνεται για τα δίκτυα της εποχής, το μέγεθος των οποίων είναι αρκετά μικρότερης κλίμακας από αυτά των σύγχρονων και οι συνδέσεις μεταξύ των οποίων πραγματοποιούνται με ενσύρματο τρόπο. Η μέθοδος αυτή είναι ιδιαίτερα δαπανηρή, λόγω του μεγάλου αριθμού των άσκοπα μεταδιδόμενων μηνυμάτων στο δίκτυο. Στην ουσία, ο κάθε κόμβος που λαμβάνει ένα μήνυμα, το προωθεί στους υπόλοιπους γείτονές του, δημιουργώντας έτσι ένα πλήθος διπλότυπων μηνυμάτων και καταναλώνοντας άσκοπα τους πόρους του δικτύου.

Το προηγούμενο γίνεται γνωστό και ως το Πρόβλημα της Θύελλας [10]. Συγκεκριμένα στο [10], ο Ni et al. (1999) εξετάζουν το πρόβλημα στα δίκτυα MANET και παρουσιάζουν το βαθμό σοβαρότητας του Προβλήματος της Θύελλας μέσω ανάλυσης

και προσομοιώσεων του αλγορίθμου της Πλημμυρίδας για δίκτυα MANET. Αναφέρουν πως στα δίκτυα MANET, η λειτουργία μετάδοσης broadcast χρησιμοποιείται εκτενώς για τον σκοπό της συνεχούς εύρεσης των βέλτιστων διαδρομών ανάμεσα στους κόμβους των δικτύων, τόσο λόγω της κινητικότητας τους, όσο και λόγω έλλειψης συγχρονισμού μεταξύ των κόμβων. Η μετάδοση broadcast επιτυγχάνεται με τη χρήση της Τυφλής Πλημμυρίδας, και με τον όρο Θύελλα περιγράφεται η κατάσταση κατά την οποία: συμβαίνουν περιττές αναμεταδόσεις του μηνύματος, το πλήθος των μηνυμάτων συμβάλλει στον ανταγωνισμό του μέσου μετάδοσης και παράλληλα στις συγκρούσεις των μηνυμάτων. Προτείνουν έτσι τη χρήση εναλλακτικών μεθόδων προώθησης για τον περιορισμό του προβλήματος, όπως την πιθανοτική μέθοδο προώθησης (προώθηση βάσει μια προκαθορισμένη τιμή πιθανότητας), τον έλεγχο προώθησης μέσω της χρήσης ενός απλού μετρητή (παύση της προώθησης όταν το μήνυμα έχει δεχτεί τόσες φορές όσες το επιτρεπτό όριο), τον έλεγχο της προώθησης με βάση την απόσταση (αποφυγή μετάδοσης σε περίπτωση που η απόσταση από τον κόμβο προορισμού είναι μικρή), τον έλεγχο της προώθησης με βάση την τοποθεσία, όπου βάσει της τοποθεσίας των κόμβων οι οποίοι έχουν ήδη στείλει το μήνυμα, υπολογίζεται η επιπλέον κάλυψη που θα μπορούσε να επιτευχθεί κατά την προώθησή του, καθώς και την μέθοδο ελέγχου της προώθησης με την χρήση συστάδων. Η αποτελεσματικότητα των παραπάνω μεθόδων, καθώς και η σύγκριση μεταξύ αυτών, παρουσιάζεται μέσω προσομοιώσεων. Από τα αποτελέσματα φαίνεται, πως ακόμα και η απλή μέθοδος του ελέγχου με χρήση μετρητή, μπορεί να μειώσει σημαντικά τις περιττές αναμεταδόσεις του δικτύου. Προκύπτει ακόμα, πως ανάμεσα από όλες τις παραπάνω μεθόδους, την καλύτερη απόδοση επιφέρει αυτή του ελέγχου βάσει τοποθεσίας, με την προϋπόθεση όμως, πως οι γεωγραφικές θέσεις των κόμβων υπολογίζονται με ακρίβεια (GPS).

Οι αυξημένες απαιτήσεις των σύγχρονων δικτύων μεγάλης κλίμακας, οδηγούν στην ανάγκη αξιοποίησης των παραπάνω εναλλακτικών μεθόδων διάχυσης πληροφορίας. Αν και όχι η βέλτιστη, η πιθανοτική μέθοδος αποτελεί τον κυρίαρχο μηχανισμό διάδοσης σε δίκτυα MANET. Συγκεκριμένα, η παραλλαγή της πιθανοτικής μεθόδου, γνωστή και ως Πιθανοτική Πλημμυρίδα, αποτελεί μια ευρέως γνωστή παραλλαγή, κυρίως λόγω της απλότητάς της.

Ο William et al. (2002)[11], διεξάγουν μια συγκριτική μελέτη διάφορων αλγορίθμων μετάδοσης broadcast, συμπεριλαμβανομένου της Πιθανοτικής Πλημμυρίδας για τα δίκτυα MANET. Δείχνουν πως αυξάνοντας τον αριθμό των κόμβων σε στατικά δίκτυα, μειώνεται η απόδοση της Πιθανοτικής Πλημμυρίδας λόγω του αυξημένου αριθμού α-

ναμεταδόσεων. Αυτό συνεπώς, καθιστά τον αλγόριθμο ακατάλληλο για συμφορημένα δίκτυα.

O Sasson et al. (2003)[12], εξετάζουν την απόδοση του αλγορίθμου με τη βοήθεια τυχαίων γράφων, προσομοιάζοντας τους με πραγματικά MANET δίκτυα. Υποστηρίζουν πως η μελέτη των παραγόντων που επηρεάζουν τη μετάβαση φάσης (φαινόμενο της θεωρίας διήθησης και των τυχαίων γράφων κατά το οποίο παρατηρείται μια απότομη μεταβολή για κάποια ιδιότητα) υπό ιδανικές συνθήκες, συμβάλλει στην κατανόηση της Πιθανοτικής Πλημμυρίδας στα πραγματικά δίκτυα MANET. Τα αποτελέσματα δείχνουν πως η Πιθανοτική Πλημμυρίδα συμβάλλει σημαντικά στην επιτυχημένη μετάδοση πακέτων στα πυκνά δίκτυα MANET, όμως τα διαγράμματα του δείκτη επιτυχίας συναρτήσει της πιθανότητας διαφέρουν ριζικά στα πραγματικά δίκτυα MANET από αυτά των ιδανικών συνθηκών, με τα πρώτα να μην εμφανίζουν καθόλου μετάβαση φάσης.

O Stauffer et al. (2007)[13] προτείνουν τον αλγόριθμο της Ευρετικής Πλημμυρίδας, ο οποίος αντί να χρησιμοποιεί μια προκαθορισμένη τιμή για την πιθανότητα προώθησης, την μεταβάλλει δυναμικά και σύμφωνα με κάποιον ευρετικό κανόνα. Τα αποτελέσματα που προκύπτουν παρουσιάζουν ενδιαφέρον. Γενικά, η Ευρετική Πλημμυρίδα προσφέρει μικρότερο αριθμό μεταδιδόμενων μηνυμάτων σε σχέση με την Πιθανοτική, μειονεκτεί όμως από άποψη χρόνου τερματισμού. Στις περιπτώσεις όμως όπου ο νόμος δύναμης (power law) των τυχαίων γράφων καθορίζεται από σχετικά μικρή παράμετρο, το πλήθος των μηνυμάτων που μειώνεται είναι αναλογικά πολύ μεγαλύτερο σε σχέση με την καθυστέρηση τερματισμού που παρουσιάζεται.

O Oikonomou και ο Stavrakakis (2007)[14], επίσης μελετούν το δίκτυο της Πιθανοτικής Πλημμυρίδας (κόμβοι και σύνδεσμοι που διασχίζει ο αλγόριθμος) υπό το πρίσμα των τυχαίων γράφων, και για πρώτη φορά θέτουν τα όρια για την τιμή της κατάλληλης τιμής προώθησης (δηλαδή, με μεγάλη πιθανότητα να επιτυγχάνεται μεγάλη κάλυψη του δικτύου) μέσω μαθηματικής ανάλυσης. Επιπλέον, θέτουν το άνω όριο του μέσου όρου του συνολικού αριθμού μηνυμάτων που μεταδίδονται κατά τον αλγόριθμο και συγκρίνουν την τιμή του, με την αντίστοιχη της παραδοσιακής Πλημμυρίδας. Η διαφορά που προκύπτει μεταξύ αυτών είναι μεγάλη, ωστόσο εξαιτίας της πιθανοτικής φύσης του αλγορίθμου, αυξάνεται ο χρόνος που χρειάζεται για την κάλυψη του δικτύου και αναιρείται ο ντετερμινιστικός χαρακτήρας της παραδοσιακής εκδοχής (κάποιοι κόμβοι δε λαμβάνουν ποτέ το μήνυμα).

Συνοψίζοντας, η Πιθανοτική Πλημμυρίδα πραγματοποιεί την προώθηση της πληροφορίας στους γείτονες χρησιμοποιώντας μια προκαθορισμένη πιθανότητα μετάδοσης.

Με τον τρόπο αυτόν, μειώνεται ο αριθμός των μηνυμάτων στο δίκτυο, και ανάλογα με την τιμή της εν λόγω πιθανότητας μετάδοσης καθορίζεται το ποσοστό κάλυψης του δικτύου. Χρησιμοποιώντας δηλαδή την πιθανοτική εκδοχή του αλγορίθμου, συμβιβάζομαστε με το γεγονός πως όσο μικρότερη είναι η πιθανότητα προώθησης του μηνύματος, και άρα και το μέγεθος της Θύελλας του δικτύου, τόσο μεγαλύτερος ο αριθμός των κόμβων που δεν καλύπτονται από τον αλγόριθμο.

Α΄.5 Δομή της Πτυχιακής Εργασίας

Τον παρόν κεφάλαιο της εργασίας είχε ως στόχο την εξοικείωση του αναγνώστη με τον σκοπό της παρούσας εργασίας. Παρουσιάστηκαν οι βασικοί κανόνες της λειτουργίας των ασύρματων δικτύων, περιγράφηκαν τα Ασύρματα Προσωπικά Δίκτυα WPAN και η σημαντικότητά τους, όπως επίσης και κάποια χαρακτηριστικά παραδείγματα τέτοιων δικτύων. Το δίκτυο που υλοποιείται στην εργασία ανήκει στην κατηγορία των Ασύρματων Δικτύων Αισθητήρων, η οποία αναφέρεται στο παρόν κεφάλαιο. Στη βιβλιογραφική ανασκόπηση περιγράφεται το ερευνητικό ενδιαφέρον γύρω από τον αλγόριθμο της Πλημμυρίδας και της Πιθανοτικής Πλημμυρίδας που έχει παρατηρηθεί τα τελευταία χρόνια.

Στα κεφάλαιο Β παρουσιάζεται ο αλγόριθμος της Πλημμυρίδας. Δίνεται ο ορισμός του και αναλύεται η πολυπλοκότητά του. Έπτερα γίνεται περιγραφή της παραλλαγής του- της Πιθανοτικής Πλημμυρίδας.

Στο κεφάλαιο Γ γίνεται η ανάλυση των πρωτοκόλλων που διέπουν τη λειτουργία του υλοποιηθέντος δικτύου. Το Φυσικό Επίπεδο και το Επίπεδο MAC ορίζονται από το πρωτόκολλο IEEE 802.15.4, ενώ το Επίπεδο Δικτύου και το Επίπεδο Εφαρμογής ορίζονται μέσω του πρωτοκόλλου ZigBee. Για τα πρωτόκολλα αυτά, περιγράφεται η λειτουργία τους, και οι τύποι συσκευών και τοπολογιών που υποστηρίζονται.

Στο κεφάλαιο Δ εξηγείται αναλυτικά ο τρόπος υλοποίησης του δικτύου με τις μονάδες-κόμβους XBee, η σύνταξη του λογισμικού της εφαρμογής και παρουσιάζονται τα αποτελέσματα.

Στο πρώτο μέρος του κεφαλαίου παρουσιάζεται ο εξοπλισμός που χρησιμοποιήθηκε, ο οποίος αποτελείται από τους μικροελεγκτές Arduino και τις μονάδες XBee. Για αυτά, περιγράφονται οι δυνατότητές τους και οι δύο δυνατές λειτουργίες των μονάδων: Transparent και API. Παρουσιάζεται έπειτα η παραμετροποίηση του συστήματος.

Το δεύτερο μέρος του κεφαλαίου, αφορά στην αναλυτική περιγραφή της ανάπτυξης

της εφαρμογής. Περιγράφεται η γλώσσα της πλατφόρμας Arduino, η δομή που έχουν τα αντίστοιχα sketch, η βοηθητική βιβλιοθήκη MyFrame που δημιουργήθηκε και τέλος σχολιάζονται αναλυτικά οι βασικές δομές, μεταβλητές και συναρτήσεις του προγράμματος, καθώς και οι ενέργειες που πραγματοποιούνται σε κάθε σύνολο εντολών.

Στη συνέχεια, το κεφάλαιο περιλαμβάνει την ανάλυση της διαδικασίας και των συνθηκών διεξαγωγής των πειραμάτων, όπως της τοπολογίας του δικτύου και των τιμών της πιθανότητας προώθησης. Παρουσιάζονται έπειτα στον αναγνώστη και περιγράφονται όλα τα διαγράμματα που προέκυψαν από την διεξαγωγή των πειραμάτων της εργασίας και αφορούν στην κάλυψη του δικτύου και τον χρόνο τερματισμού.

Τέλος, το κεφάλαιο Ε περιλαμβάνει τη σύνοψη και γίνεται αναφορά στα συμπεράσματα της πτυχιακής εργασίας.

Στο παράρτημα της εργασίας, παρατίθεται στο σύνολο του ο κώδικας που αναπτύχθηκε και η βιβλιοθήκη MyFrame.h.

Κεφάλαιο Β΄

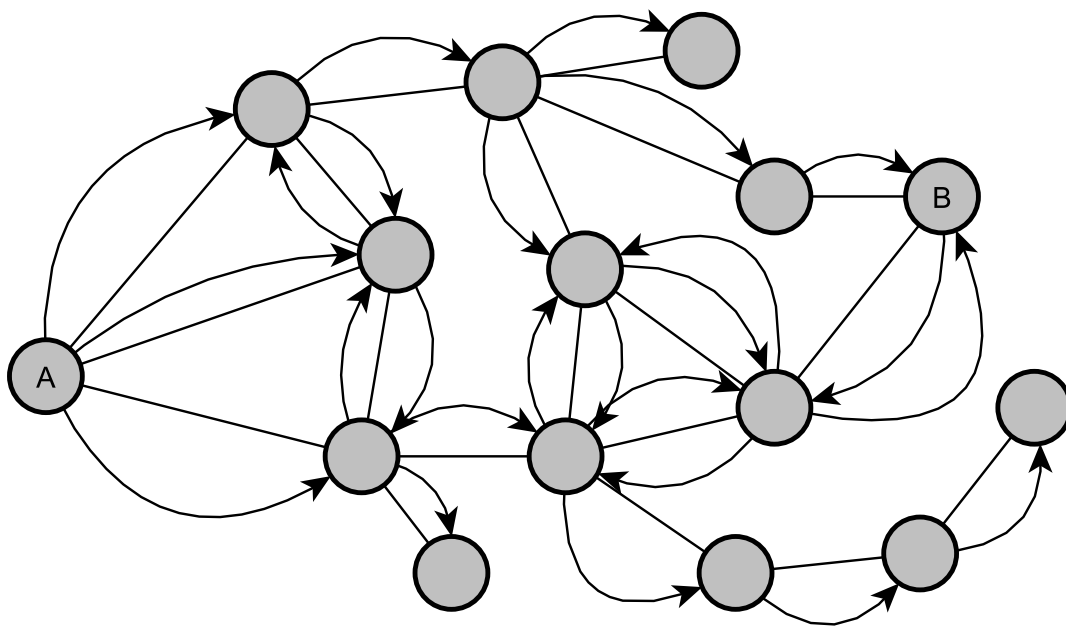
Πλημμυρίδα και Πιθανοτική Πλημμυρίδα

Η αποδοτική διάδοση πληροφορίας στους κόμβους του δικτύου αποτελεί ένα από τα βασικότερα αλγοριθμικά ζητήματα. Διάφοροι αλγόριθμοι αναλαμβάνουν τη δρομολόγηση μηνυμάτων στο δίκτυο με σκοπό την κάλυψή του. Έναν από τους πλέον κλασικούς αποτελεί ο αλγόριθμος της Πλημμυρίδας, με το μειονέκτημά του να είναι ιδιαίτερα κοστοβόρος. Διάφορες παραλλαγές του έχουν προταθεί κατά καιρούς, οι οποίες εισάγουν τον παράγοντα της πιθανότητας με στόχο την εξάλειψη των περιττών αναμεταδόσεων των μηνυμάτων και την αύξηση της συνολικής απόδοσής του. Τέτοιες παραλλαγές είναι ιδιαίτερα πρακτικές για τα μεγάλης κλίμακας σύγχρονα περιβάλλοντα. Κάποιες από αυτές είναι ο αλγόριθμος του Κουτσομπολιού (Gossiping) και ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας (Probabilistic Flooding), ο οποίος και αναλύεται παρακάτω.

Β΄.1 Πλημμυρίδα

Όπως ήδη αναφέρθηκε, αλγόριθμος της Πλημμυρίδας χρησιμοποιείται όταν είναι αναγκαίο να επιτευχθεί η καθολική κάλυψη ενός δικτύου. Αυτή η απαίτηση εμφανίζεται συχνά σε περιπτώσεις που σε έναν κόμβο εμπεριέχεται κάποια σημαντική πληροφορία για τους υπόλοιπους κόμβους του δικτύου. Τότε ο αρχικός κόμβος με τη χρήσιμη πληροφορία ή αλλιώς ο κόμβος εκκινήτης (initiator node), αναλαμβάνει να προωθήσει το μήνυμα-πληροφορία προς τους υπόλοιπους κόμβους του δικτύου. Συγκεκριμένα

ο εκκινητής προωθεί το μήνυμα σε όλους τους γειτονικούς του κόμβους. Οι κόμβοι που λαμβάνουν το μήνυμα με τη σειρά τους το προωθούν και αυτοί στους δικούς τους γειτονικούς κόμβους, πλην του κόμβου από τον οποίο έλαβαν το μήνυμα. Για να είναι δυνατή η διεκπεραίωση του αλγόριθμου της Πλημμυρίδας σε τοπολογίες που περιέχουν κύκλους (μη δεντρικές τοπολογίες), σε περίπτωση που ένας κόμβος δεχτεί το μήνυμα της Πλημμυρίδας για παραπάνω από μια φορά, δεν το θα το επαναπροωθήσει. Η εικόνα Β'.1 παρουσιάζει την προώθηση πληροφορίας από τον αρχικό κόμβο Α (εκκινητή) σε κάποιον κόμβο προορισμού Β. Κάθε βέλος στο σχήμα απεικονίζει μια μετάδοση, ενώ οι ευθείες γραμμές τις ζεύξεις μεταξύ κόμβων. Επειδή το κάθε μήνυμα προωθείται σε όλες τις ζεύξεις του κόμβου, στις πυκνότερες περιοχές του δικτύου παρατηρείται μεγαλύτερος αριθμός αναμεταδόσεων. Επίσης, μπορεί κανείς να παρατηρήσει τον μεγάλο αριθμό περιττών αναμεταδόσεων του μηνύματος στο δίκτυο της εικόνας. Η αποστολή της πληροφορίας από τον κόμβο Α στον κόμβο προορισμού Β, μπορεί να επιτευχθεί με αρκετά μικρότερο αριθμό αναμεταδόσεων.



Σχήμα Β'.1: Μετάδοση πληροφορίας με τον αλγόριθμο της Πλημμυρίδας. Τα βέλη του σχήματος απεικονίζουν τις μεταδόσεις, ενώ οι ευθείες τις ζεύξεις μεταξύ των κόμβων.

B'.1.1 Απόδοση Πλημμυρίδας

Εάν θεωρήσουμε N τον αριθμό των κόμβων του δικτύου και E τον αριθμό των ζεύξεων που υπάρχει στο δίκτυο, τότε ο συνολικός αριθμός των μηνυμάτων που μεταδίδονται κατά τον αλγόριθμο της Πλημμυρίδας ισούται με:

$$M = 2E - N + 1 \quad (B'.1)$$

Ο παραπάνω τύπος γίνεται κατανοητός, εάν αναλογιστεί κανείς πως σε κάθε ζεύξη του δικτύου στέλνονται δύο μηνύματα ($2E$), ένα από τον κόμβο A προς τον κόμβο B , και ένα από τον κόμβο B προς τον κόμβο A . Σύμφωνα με τον ορισμό, όλοι οι κόμβοι (N) δεν μεταδίδουν μήνυμα πίσω στον γείτονα από τον οποίο το έλαβαν, με μόνη εξαίρεση τον εκκινήτη ($N - 1$ μηνύματα) ο οποίος στέλνει πρώτος και σε όλους. Ο αριθμός των συνολικών μηνυμάτων προκύπτει, αφαιρώντας τον αριθμό των μη μεταδιδόμενων μηνυμάτων ($N - 1$), από τον αριθμό των συνολικών ($2E$).

Επομένως, η πολυπλοκότητα του αλγορίθμου της Πλημμυρίδας αναφορικά με τα μηνύματα που μεταδίδονται είναι $O(M)$.

Ο χρόνος εκτέλεσης του αλγορίθμου της Πλημμυρίδας εξαρτάται άμεσα από την απόσταση του αρχικού κόμβου από τον πλέον απομακρυσμένο. Ο αλγόριθμος τερματίζει όταν το μήνυμα φτάσει σε όλους του κόμβους του δικτύου ή αλλιώς, το μήνυμα ληφθεί από τον πιο απομακρυσμένο κόμβο του δικτύου σε σχέση με τον αρχικό. Η απόσταση αυτή μπορεί να μετρηθεί σε αριθμό αλμάτων (hops), όπου κάθε άλμα είναι ίσο με μία χρονική μονάδα και συγκεκριμένα η μέγιστη απόσταση ενός αρχικού κόμβου A από τον πλέον απομακρυσμένο Ω είναι ίση με την εκκεντρότητα του κόμβου A . Για την χειρότερη περίπτωση κατά την οποία η εκκεντρότητα του αρχικού κόμβου είναι ίση με τη διάμετρο D του δικτύου, ο χρόνος που χρειάζεται για να τερματίσει ο αλγόριθμος είναι της τάξης D ή έχει πολυπλοκότητα $O(D)$. Ενδιαφέρον έχει να σημειωθεί, πως ο αλγόριθμος της Πλημμυρίδας εγγυάται την χρήση της συντομότερης διαδρομής, αφού κατά την εκτέλεσή του κάνει χρήση όλων των δυνατών διαδρομών μέσα στο δίκτυο.

Για τις περισσότερες σύγχρονες εφαρμογές ο αλγόριθμος της Πλημμυρίδας δεν χρησιμοποιείται σε μεγάλο βαθμό, καθώς δεν θεωρείται ιδιαίτερα πρακτικός. Υπάρχουν όμως αρκετές περιπτώσεις στις οποίες επιφέρει σημαντικά πλεονεκτήματα, όπως το γεγονός ότι εγγυάται την πλήρη κάλυψη του δικτύου, η ανθεκτικότητά του σε περίπτωση απώλειας ενός μεγάλου αριθμού κόμβων δικτύου (υπό την προϋπόθεση πως το δίκτυο παραμένει συνεκτικό) και η χρήση του για σύγκριση απόδοσης με άλλους αλγορίθμους δρομολόγησης [15].

Από τον παραπάνω αναλυτικό ορισμό που έχει δοθεί για τον αλγόριθμο, είναι προφανές πως κατά την εκτέλεσή του, μεταδίδεται ένας μεγάλος αριθμός μηνυμάτων στο δίκτυο. Μάλιστα, γίνεται αντιληπτό πως η επίτευξη του ζητούμενου (της πλήρης κάλυψης), θα μπορούσε να πραγματοποιηθεί με λιγότερες ανταλλαγές μηνυμάτων ανάμεσα στους κόμβους.

Συγκεκριμένα, μέσα στο δίκτυο ανταλλάσσεται μεγάλος αριθμός αντιγράφων του ίδιου μηνύματος, με αποτέλεσμα κάποιοι κόμβοι να λαμβάνουν το μήνυμα πολλές φορές και από διαφορετικούς γείτονες. Η Πλημμυρίδα δεν υλοποιεί κάποιον μηχανισμό ελέγχου των περιττών μεταδόσεων και κάθε μετάδοση, υπό πραγματικές συνθήκες, μεταφράζεται σε κατανάλωση ενέργειας, γεγονός που καθιστά τον αλγόριθμο σημαντικά κοστοβόρο. Επομένως, για τα σημερινά μεγάλης κλίμακας περιβάλλοντα, γίνονται προσπάθειες μείωσης των μεταδιδόμενων στο δίκτυο μηνυμάτων χρησιμοποιώντας διάφορες παραλλαγές του παραδοσιακού αλγορίθμου της Πλημμυρίδας, μία εκ των οποίων είναι η Πιθανοτική Πλημμυρίδα και αναλύεται παρακάτω.

Β'.2 Πιθανοτική Πλημμυρίδα

Ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας προσπαθεί να περιορίσει τον πλεονάζοντα αριθμό των παραγόμενων μηνυμάτων κατά την κλασική Πλημμυρίδα, ο οποίος οδηγεί σε συμφόρηση του δικτύου και σπαταλά πολύτιμους πόρους.

Για να επιτευχθεί αυτό, γίνεται χρήση μίας προκαθορισμένης τιμής (κατωφλίου), προκειμένου να ληφθεί η απόφαση για τη μετάδοση του μηνύματος προς κάποιον γειτονικό κόμβο. Συγκεκριμένα, ο κάθε κόμβος που λαμβάνει το μήνυμα, εξετάζει, αν η τιμή ενός τυχαία παραγόμενου αριθμού είναι μεγαλύτερη από το κατώφλι, για να πραγματοποιηθεί κατά αυτόν τον τρόπο η μετάδοση του μηνύματος προς κάθε ξεχωριστό γείτονα. Η τιμή αυτή ονομάζεται πιθανότητα προώθησης q , επιλέγεται από τον κόμβο εκκινήτη και μεταδίδεται μαζί με το μήνυμα προς τους υπόλοιπους κόμβους. Όπως και στην κλασική Πλημμυρίδα, σε περίπτωση που κάποιος κόμβος λάβει το μήνυμα για δεύτερη φορά, δεν θα το επαναπροωθήσει σε κανέναν γειτονικό του κόμβο.

Λόγω της πιθανοτικής του φύσης, ο παρόν αλγόριθμος είναι μη ντετερμινιστικός, δεν μπορούμε δηλαδή εξ αρχής να γνωρίζουμε για το αποτέλεσμα του (ποσοστό κάλυψης κόμβων δικτύου). Εύκολα όμως συμπεραίνουμε, πως όσο μεγαλύτερη η πιθανότητα προώθησης, τόσο πιο σίγουροι μπορούμε να είμαστε για την κάλυψη ολόκληρου του δικτύου. Επιπλέον αντιλαμβανόμαστε, πως το ποσοστό των ζεύξεων του δικτύου

επηρεάζει το αποτέλεσμα της κάλυψης. Ένα πυκνό δίκτυο αυξάνει την πιθανότητα ένα μήνυμα να επιζήσει για περισσότερη ώρα μέσα στο δίκτυο, σε σχέση με ένα αραιό, και επομένως συμβάλλει στο αποτέλεσμα της κάλυψης.

Επιπρόσθετα, η Πιθανοτική Πλημμυρίδα μπορεί να αυξήσει σημαντικά τον απαιτούμενο χρόνο για την ολική κάλυψη του δικτύου. Τα μηνύματα που εκπέμπονται από τον κόμβο εκκινήτη, λόγω του γεγονότος ότι δε διατηρούν μνήμη, δεν έχουν δηλαδή σαφή προορισμό, μπορούν να αναμεταδίδονται για αρκετό χρονικό διάστημα ανάμεσα σε κάποιους κόμβους, μέχρι να καλυφθεί τελικά όλο το δίκτυο. Έτσι, η εισαγωγή μιας πιθανότητας μετάδοσης τελικά δυσχεράνει την απόδοση του δικτύου, αφού διάφορες τοπολογίες επιτρέπουν την μετάδοση μέσω εναλλακτικών διαδρομών, με αποτέλεσμα να μην ακολουθείται πάντα η βέλτιστη δυνατή και έτσι χρονικά να απαιτείται περισσότερος χρόνος για τον τερματισμό της διαδικασίας. Για τον λόγο αυτό, κάποιες παραλλαγές χρησιμοποιούν ένα μέγιστο μήκος K διαδοχικών μεταβιβάσεων (hops) που μπορεί να διανύσει το μήνυμα, αφού αναχωρήσει από τον κόμβο εκκινήτη. Με αυτόν τον τρόπο, τόσο ο χρόνος τερματισμού της διαδικασίας, αλλά και το κόστος αυτής μειώνονται, ανάλογα με την τιμή του K .

Ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας φέρνει στο φως ενδιαφέροντα ερευνητικά ζητήματα που αφορούν στην εύρεση τιμών πιθανότητας προώθησης (threshold probability) με τρόπο που θα εξασφαλίζει με μεγάλο ποσοστό επιτυχίας την πλήρη κάλυψη του δικτύου, τον μικρό αριθμό μεταδιδόμενων μηνυμάτων και τη μικρή αύξηση του χρόνου τερματισμού. Αποδεικνύεται πως τα παραπάνω ζητούμενα μπορούν να επιτευχθούν με την κατάλληλη παραμετροποίηση. Στο [14], με τη βοήθεια μαθηματικής ανάλυσης προκύπτει ένα όριο για την κατάλληλη τιμή της πιθανότητας μετάδοσης. Αργότερα, στο [16][Oikonomou et al. (2012)], η κάλυψη του δικτύου από τον αλγόριθμο εκφράζεται ως ένα πολυώνυμο, η μελέτη των ριζών του οποίου θέτει νέα και αυστηρότερα όρια για την τιμή της κατάλληλης πιθανότητας προώθησης, καθώς και επιβεβαιώνει τα προηγούμενα αποτελέσματα της βιβλιογραφίας. Επιπρόσθετα, παρουσιάζονται αποτελέσματα προσομοιώσεων που συμφωνούν με την αναλυτική προσέγγιση και δείχνουν πως τα όρια που προκύπτουν, πράγματι αυξάνουν την κάλυψη του δικτύου με τη χρήση του αλγόριθμου.

Κεφάλαιο Γ΄

Πρωτόκολλα Συστήματος

Το σύστημα που υλοποιήθηκε για την πραγματοποίηση των πειραμάτων, συνίσταται από τα XBee-PRO S2C Zigbee RF Modules της εταιρείας Digi International, τα οποία αποτελούν τους κόμβους του δικτύου. Στη συνέχεια του κεφαλαίου αναλύονται τα πρότυπα 802.15.4 και ZigBee τα οποία διέπουν τη λειτουργία των XBee RF Modules, καθώς και ο μικροελεγκτής Arduino UNO πάνω στον οποίο αυτοί ενσωματώνονται.

Γ΄.1 IEEE 802.15.4

Το IEEE 802.15.4 [4] καθορίζει την ασύρματη επικοινωνία των χαμηλής ταχύτητας (low-rate) ασύρματων δικτύων προσωπικής περιοχής (LR-WPANs). Συγκεκριμένα, εστιάζει στην διαμόρφωση προτύπων τα οποία καθορίζουν το φυσικό επίπεδο (Physical Layer) και το επίπεδο MAC (Medium Access Control). Το πρότυπο επικυρώθηκε τον Μάιο του 2006 από την ομάδα εργασίας του IEEE 802.15. Ο στόχος του 802.15.4 είναι η πολύ χαμηλή κατανάλωση ενέργειας, το χαμηλό κόστος και η απλότητά του, γεγονός που το καθιστά κατάλληλο για εφαρμογές όπως τα δίκτυα αισθητήρων ή έξυπνα σπίτια (smart homes)[17].

Γ΄.1.1 Τύποι Συσκευών που Υποστηρίζονται

Το πρότυπο υποστηρίζει δύο είδη συσκευών: την συσκευή πλήρους λειτουργικότητας (FFD) και την συσκευή μειωμένης λειτουργικότητας (RFD). Μία συσκευή (FFD) εκτελεί όλες τις λειτουργίες που προδιαγράφονται στο MAC επίπεδο και είναι σε θέση να επικοινωνήσει με οποιαδήποτε συσκευή του δικτύου, και να αναλάβει τον ρόλο του

συντονιστή του δικτύου network coordinator ή του δρομολογητή router ανάλογα με την εφαρμογή. Αντιθέτως, οι συσκευές (RFD) έχουν λιγότερες δυνατότητες και κάθε τέτοια συσκευή είναι απαραίτητο να συνδέεται με μία συσκευή πλήρους λειτουργίας που μόνο με αυτήν μπορεί να επικοινωνήσει. Για τον λόγο αυτό, αναλαμβάνουν συνήθως να εκτελέσουν απλές διεργασίες και δεν απαιτούν μεγάλη υπολογιστική ισχύ και ενεργειακή κατανάλωση. Η ύπαρξη τουλάχιστον μίας συσκευής (FFD) σε ένα δίκτυο είναι αναγκαία για να αναλάβει τον μοναδικό ρόλο του συντονιστή και να γίνει δυνατή η δημιουργία του δικτύου. ζιτεβαροντι2007ωιρελεσς

Γ'.1.2 Τοπολογίες

Το πρότυπο υποστηρίζει πολλά είδη τοπολογιών όπως αστέρα (star), δέντρου (tree), δέντρου συστάδων (cluster tree) και την τοπολογία πλέγματος (mesh).

Τα παραπάνω μπορούν να χωριστούν σε δύο κύρια είδη τοπολογιών: την τοπολογία αστέρα και την τοπολογία ομότιμων κόμβων (peer-to-peer).

Στην τοπολογία αστέρα η κάθε συσκευή του δικτύου έχει την δυνατότητα να συνδεθεί μόνο με μία κεντρική συσκευή πλήρους λειτουργικότητας. Οι ζεύξεις μεταξύ των ακριανών συσκευών με την κεντρική ελέγχονται από την δεύτερη που λειτουργεί ως συντονιστής του δικτύου. Για να πραγματοποιηθεί μια ζεύξη, η συσκευή που θέλει να συνδεθεί με το δίκτυο στέλνει αίτημα στον συντονιστή, ο οποίος απαντάει στο αίτημα. Αποτελεί έναν γρήγορο τρόπο επικοινωνίας των περιφερειακών συσκευών με την κεντρική και μπορεί να λειτουργήσει σε περιοχές μικρών εκτάσεων.

Στην τοπολογία ομότιμων κόμβων, όλες οι συσκευές μπορούν να συνδεθούν με οποιαδήποτε άλλη, χωρίς περιορισμό στον αριθμό των συνδέσεων. Είναι χρήσιμη για περιοχές μεγάλων εκτάσεων όπου δεν είναι δυνατή η απευθείας σύνδεση όλων των συσκευών με μία. Έτσι το μήνυμα δρομολογείται ώστε να επιτευχθεί η επικοινωνία όλων των συσκευών με τη χρήση των ενδιάμεσων γειτονικών συνδέσεων.

Γ'.1.3 Τα Επίπεδα του Προτύπου

Αποτελεί συνήθης πρακτική τα πρωτόκολλα των δικτύων να διαχωρίζονται σε συστατικά στοιχεία, τα επίπεδα. Κάθε ένα διαφορετικό επίπεδο εκτελεί ένα σύνολο λογικά σχετιζόμενων μεταξύ τους λειτουργιών και παρέχει υπηρεσίες για λογαριασμό του αμέσως επόμενου επιπέδου.

Στο φυσικό επίπεδο του 802.15.4, όπως και στο αντίστοιχο του OSI, εκτελείται η αποκωδικοποίηση και η κωδικοποίηση των πακέτων από και σε ακολουθίες bytes και η εκπομπή και λήψη των δεδομένων στο φυσικό μέσο μετάδοσης (ζώνες ραδιοσυχνοτήτων). Επομένως, περιλαμβάνει διαδικασίες όπως η διαχείριση του πομποδέκτη (κεραίας), την επιλογή του καναλιού επικοινωνίας, καθώς και τη αποστολή - λήψη πακέτων μέσω του καναλιού. Το πρότυπο ορίζει συνολικά είκοσι επτά κανάλια σε τρεις ζώνες συχνοτήτων (868.0-868.6 MHz, 902-928 MHz και 2400-2483.5 MHz).

Το επίπεδο MAC ελέγχει την πρόσβαση στο κοινόχρηστο φυσικό κανάλι. Κύριες λειτουργίες του επιπέδου είναι η σύνδεση - αποσύνδεση συσκευών από - στο δίκτυο, οι αποστολές των beacons (beacon frames), ο έλεγχος ασφάλειας, η διαχείριση των χρονθυρίδων (timeslots) και η διαχείριση των Acks frames. Βασίζεται στο πρωτόκολλο CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance) το οποίο ανήκει στην ομάδα πρωτοκόλλων ανίχνευσης φέροντος σήματος. Τα πρωτόκολλα αυτά παρακολουθούν το κανάλι επικοινωνίας για να εντοπίσουν ένα φέρον σήμα, ώστε να αποφασίσουν για την πραγματοποίηση της μετάδοσης.

Το πρωτόκολλο IEEE 802.15.4 ορίζει δύο τρόπους πρόσβασης στο κανάλι επικοινωνίας. Ο πρώτος τρόπος χρησιμοποιεί το unslotted CSMA/CA χωρίς σήματα beacons. Στο unslotted CSMA/CA, οι κόμβοι ακούνε το κανάλι και μπορούν να μεταδώσουν σε περίπτωση που αυτό είναι ελεύθερο, αλλιώς περιμένουν ένα back-off period. Ο δεύτερος χρησιμοποιεί τα σήματα beacons. Οι συσκευές του δικτύου μπορούν να πραγματοποιήσουν μεταδόσεις σε προκαθορισμένα χρονικά διαστήματα, τις χρονθυρίδες. Ο συντονιστής καθορίζει τις εξασφαλισμένες χρονθυρίδες (Guaranteed Time Slot - GTS) και κάθε συσκευή του δικτύου μεταδίδει στη δικιά της χρονθυρίδα. Για τον σκοπό αυτό ο συντονιστής εκπέμπει σήματα beacons ανά καθορισμένα χρονικά διαστήματα Beacon Intervals και συγχρονίζει τις συνδεδεμένες με αυτόν συσκευές. Αξίζει να σημειωθεί πως η χρήση beacons είναι ενεργειακά περισσότερο δαπανηρή, καθώς απαιτεί τις συσκευές να πραγματοποιούν συνεχή έλεγχο για σήματα συγχρονισμού. [18]

Γ'.2 Τεχνολογία ZigBee

Το ZigBee αποτελεί ένα ανοιχτό πρότυπο επικοινωνίας για χαμηλής ταχύτητας κατανεμημένα (mesh) ασύρματα δίκτυα προσωπικής περιοχής. Αναπτύχθηκε από την ZigBee Alliance, μία κοινοπραξία μη κερδοσκοπικού χαρακτήρα, η οποία αποτελείται από πλήθος μεγάλων οργανώσεων όπως πανεπιστήμια, εταιρείες παραγωγής ηλεκτρονικού εξο-

πλισμού και εταιρείες παραγωγής ημιαγωγών. Το όνομά του προτύπου προέρχεται από τις ζιγκ ζαγκ κινήσεις που παρατηρούνται κατά τη διάρκεια του χορού των μελισσών (bees), με σκοπό να επικοινωνήσουν στις υπόλοιπες μέλισσες της κυψέλης κάποια πληροφορία. Οι κινήσεις αυτές αντιπροσωπεύουν τις ανταλλαγές των μηνυμάτων μεταξύ των κόμβων σε mesh δίκτυα [19].

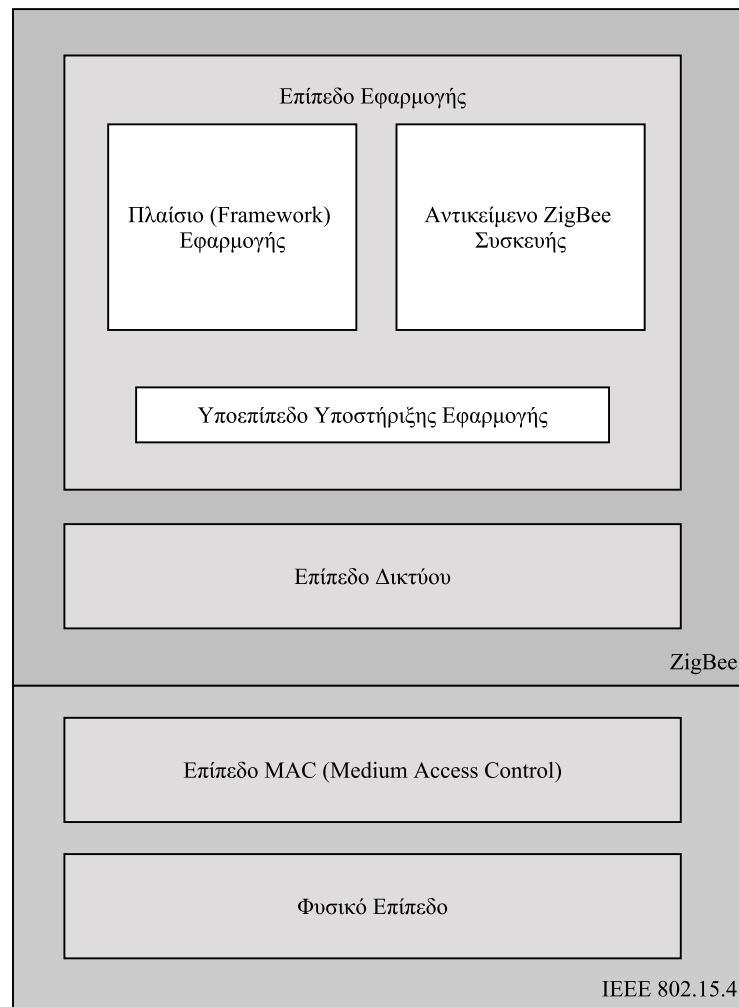
Το πρότυπο ZigBee λειτουργεί στις ζώνες ραδιοσυχνοτήτων ISM (Industrial, Scientific and Medical). Οι ραδιοσυχνότητες της ζώνης αυτής δεν απαιτούν αδειοδότηση και προορίζονται για μη εμπορικές χρήσεις (επιστήμη-εκπαίδευση-ιατρική). Η ακριβής ραδιοσυχνότητα εξαρτάται από τη γεωγραφική τοποθεσία της συσκευής. Κατά κανόνα, η συχνότητα 868 MHz αντιστοιχεί στις συσκευές των περιοχών της Ευρώπης, η 915 MHz των περιοχών των Ηνωμένων Πολιτειών, ενώ η συχνότητα 2.4 GHz χρησιμοποιείται από τις περισσότερες περιοχές του κόσμου. Η ταχύτητα μετάδοσης προσδιορίζεται από τις ζώνες συχνοτήτων, με τη μέγιστη να ανέρχεται στα 250 Kbps, αριθμός αρκετά χαμηλός σε σύγκριση με άλλα ασύρματα πρωτόκολλα. Λόγω αυτών των ταχυτήτων μετάδοσης, θεωρείται ένα πρότυπο χαμηλής κατανάλωσης ενέργειας όπως και χαμηλού κόστους. Επίσης, χαρακτηρίζεται από ισχυρή ασφάλεια, υψηλή αξιοπιστία ως προς τη μετάδοση των δεδομένων και ευελιξία. [20]

Η λειτουργία του ZigBee βασίζεται στο πρότυπο IEEE 802.15.4, το οποίο ενσωματώνεται στα χαμηλότερα επίπεδά του, μιας και η αρχιτεκτονική του στηρίζεται στην διαστρωμάτωση. Συγκεκριμένα, εκτός του φυσικού επιπέδου και του επιπέδου MAC που ορίζονται από το IEEE 802.15.4, το ZigBee προσθέτει το επίπεδο δικτύου (Network Layer) και το επίπεδο εφαρμογών (Application Layer).[17]

Γ'.2.1 Συσκευές ZigBee

Το πρότυπο ZigBee ορίζει τους εξής τρεις τύπους κόμβων δικτύου ο κάθε ένας εκ των οποίων εκτελεί διαφορετικές λειτουργίες [21]:

- Τον κόμβο συντονιστή (Coordinator), ο οποίος είναι μοναδικός σε κάθε διαφορετικό δίκτυο και υπεύθυνος για τη διαμόρφωσή του. Επιλέγει, τόσο το κανάλι της επικοινωνίας, όσο και το αναγνωριστικό του δικτύου (PAN ID), και στη συνέχεια επιτρέπει σε άλλους κόμβους να συνδεθούν στο δίκτυο. Όσον αφορά τη δρομολόγηση, ο coordinator είναι σε θέση να εκτελέσει όλες τις βασικές λειτουργίες των απλών δρομολογητών του δικτύου όπως είναι η αποστολή και η λήψη πακέτων από και προς τον κατάλληλο προορισμό, καθώς και να αποτελέσει



Σχήμα Γ΄.1: Στοιίβα πρωτοκόλλων ZigBee και 802.15.4

ο ίδιος την πηγή ή τον προορισμό ορισμένων πακέτων. Ο συντονιστής του δικτύου χρειάζεται να παραμένει σε συνεχή κατάσταση τροφοδοσίας, ώστε να είναι σε θέση να ανταποκριθεί στις παραπάνω λειτουργίες.

- Τον κόμβο δρομολογητή (Router), ο οποίος αναλαμβάνει τη δρομολόγηση πακέτων από και προς του κατάλληλους προορισμούς με τον βέλτιστο δυνατό τρόπο (συντομότερη διαδρομή) βάσει των πληροφοριών που συλλέγει για το δίκτυο. Ο κάθε δρομολογητής μπορεί να αποτελέσει πηγή ή προορισμό ορισμένων πακέ-

των. Μπορεί ακόμα να επιτρέψει τη σύνδεση άλλων δρομολογητών ή τερματικών συσκευών στο δίκτυο. Για να είναι σε θέση να εκτελέσει τις διαδικασίες δρομολόγησης, ο δρομολογητής θα πρέπει να παραμένει σε συνεχή κατάσταση τροφοδοσίας.

- Την τερματική συσκευή (End device), η οποία αποτελεί μέρος του δικτύου, αλλά δεν πραγματοποιεί δρομολογήσεις πακέτων. Είναι απαραίτητο κάθε τερματική συσκευή να είναι συνδεδεμένη με έναν δρομολογητή ή έναν συντονιστή, ο οποίος αναλαμβάνει να δρομολογήσει πακέτα από και προς την τερματική συσκευή. Η τερματική συσκευή μπορεί περιοδικά να μεταβαίνει σε κατάσταση ύπνου και έτσι τα πακέτα που προορίζονται για αυτήν διατηρούνται στον δρομολογητή με τον οποίον αυτή είναι συνδεδεμένη. Όποτε η συσκευή αφυπνίζεται ο δρομολογητής της μεταδίδει τα αποθηκευμένα πακέτα. Λόγω της χαμηλής κατανάλωσής της σε ενέργεια η συσκευή μπορεί να λειτουργήσει με μπαταρία και η συνεχή τροφοδοσία της με ρεύμα δεν καθίσταται απαραίτητη.

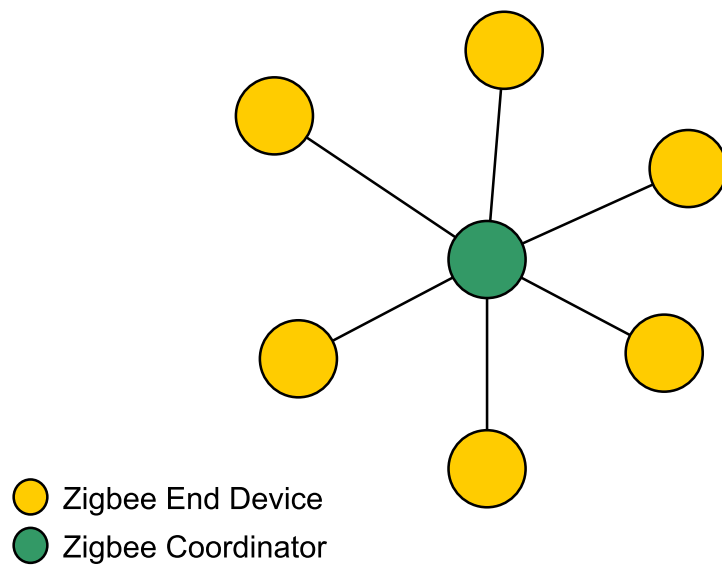
Γ'.2.2 Τοπολογίες

Οι τοπολογίες δικτύωσης διακρίνονται κατά κύριο λόγο σε αστέρα, δέντρο και πλέγμα [22].

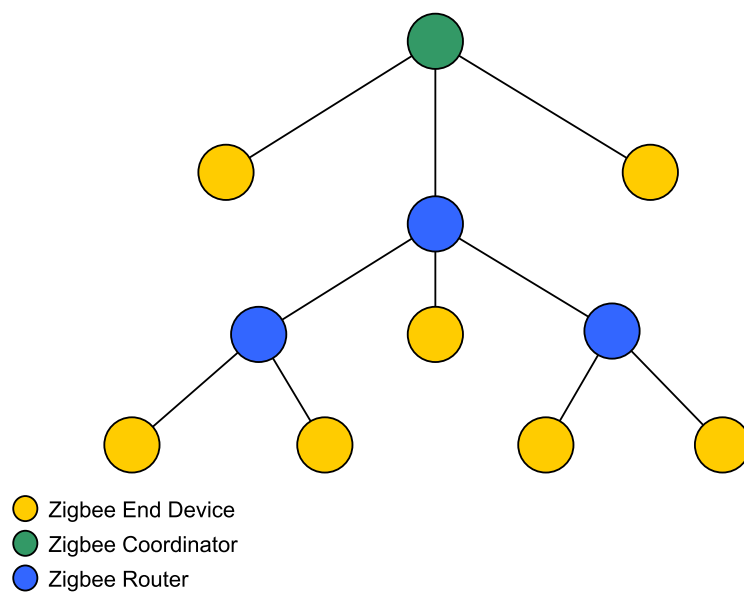
Η **τοπολογία αστέρα** αποτελείται από τον συντονιστή και τις συνδεδεμένες σε αυτόν συσκευές οι οποίες είναι δρομολογητές ή τερματικές συσκευές. Δύο κόμβοι μπορούν να επικοινωνήσουν μεταξύ τους μόνο μέσω του συντονιστή καθώς δεν μπορούν να είναι άμεσα συνδεδεμένες μεταξύ τους.

Στη **δεντρική τοπολογία**, οι συσκευές παιδιά του συντονιστή, μπορούν να συνδεθούν με άλλες συσκευές και να ενεργήσουν αντίστοιχα ως γονείς επεκτείνοντας το δίκτυο με αυτόνομο τρόπο. Όλα τα μηνύματα της συσκευής παιδιού δρομολογούνται μόνο μέσω της γονικής συσκευής.

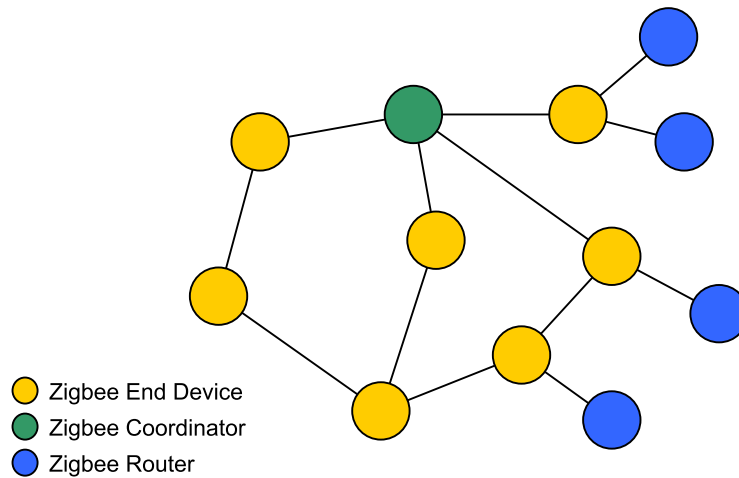
Στη **τοπολογία πλέγματος** ο συντονιστής διαμορφώνει το δίκτυο, όμως κάθε δρομολογητής έχει τη δυνατότητα να το επεκτείνει επιτρέποντας την είσοδο καινούργιων συσκευών. Επιπρόσθετα, δεν υπάρχει ιεραρχία όσον αφορά τη δρομολόγηση και κάθε συσκευή μπορεί να επικοινωνήσει απ' ευθείας με μια άλλη. Λόγω της κατανεμμένης φύσης της τοπολογίας και του πρωτοκόλλου, εάν κάποια συσκευή εξέλθει από το δίκτυο ή τεθεί εκτός λειτουργίας η δρομολόγηση θα πραγματοποιηθεί κανονικά με τη αυτόματη εύρεση εναλλακτικών διαδρομών.



Σχήμα Γ'.2: Τοπολογία αστέρα



Σχήμα Γ'.3: Τοπολογία δέντρου



Σχήμα Γ'.4: Τοπολογία πλέγματος

Γ'.2.3 Επίπεδο Δικτύου

Το επίπεδο δικτύου είναι υπεύθυνο για τον σχηματισμό του δικτύου, την είσοδο ή έξοδο μιας συσκευής από το δίκτυο, την ασφάλεια και τη δρομολόγηση των πακέτων. Μπορεί επίσης να εκτελέσει το Network Discovery, διαδικασία που ανακαλύπτει το δίκτυο σύμφωνα με τον PAN ID και τη συχνότητα του καναλιού, και επιστρέφονται όλες τις συσκευές που συγκροτούν το δίκτυο.

Γ'.2.4 Επίπεδο Εφαρμογών

Αντίστοιχα, το επίπεδο εφαρμογών αποτελεί το ανώτερο στη στοίβα επιπέδων και συμπεριλαμβάνει το υποεπίπεδο υποστήριξης εφαρμογών (Application Support SubLayer), το οποίο παρέχει λειτουργίες διασύνδεσης της εφαρμογής για διαφορετικές συσκευές κάνοντας χρήση των ZigBee αντικειμένων συσκευής (ZDO). Στις λειτουργίες των αντικειμένων αυτών, μεταξύ άλλων, εμπεριέχεται ο προσδιορισμός του ρόλου της κάθε συσκευής του δικτύου και τα δικαιώματά της. Όλα τα επίπεδα του πρωτοκόλλου ZigBee κάνουν χρήση των ειδικών εντολών primitives με σκοπό να χρησιμοποιήσουν τις υπηρεσίες των γειτονικών επιπέδων τους.

Κεφάλαιο Δ΄

Υλοποίηση του Δικτύου

Στα πλαίσια της παρούσας πτυχιακής υλοποιήθηκε ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας για ένα ασύρματο δίκτυο. Το πρόγραμμα εκτελείται με τον εξής τρόπο: η Πιθανοτική Πλημμυρίδα διαδίδεται από τον αρχικό κόμβο προς το δίκτυο ενώ οι κόμβοι οι οποίοι το λαμβάνουν, απαντάνε με αντίστοιχο μήνυμα απάντησης στον αρχικό κόμβο. Το μήνυμα απάντησης εμπεριέχει χρήσιμες πληροφορίες για το δίκτυο: τον χρόνο λήψης του μηνύματος από τον κόμβο, τον αριθμό των κόμβων από τους οποίους έχει περάσει το μήνυμα (hops) τη στιγμή που λήφθηκε και τη τοπολογία του δικτύου. Πραγματοποιείται επίσης ένα είδος συγχρονισμού των κόμβων με τον αρχικό με σκοπό τη διατήρηση της πληροφορίας του πραγματικού χρόνου του πειράματος. Στο κεφάλαιο αυτό, παρουσιάζεται ο εξοπλισμός που χρησιμοποιήθηκε για τα πειράματα, το περιβάλλον υλοποίησης και γίνεται αναλυτική περιγραφή του προγράμματος.

Δ΄.1 Εξοπλισμός και Περιβάλλον Υλοποίησης

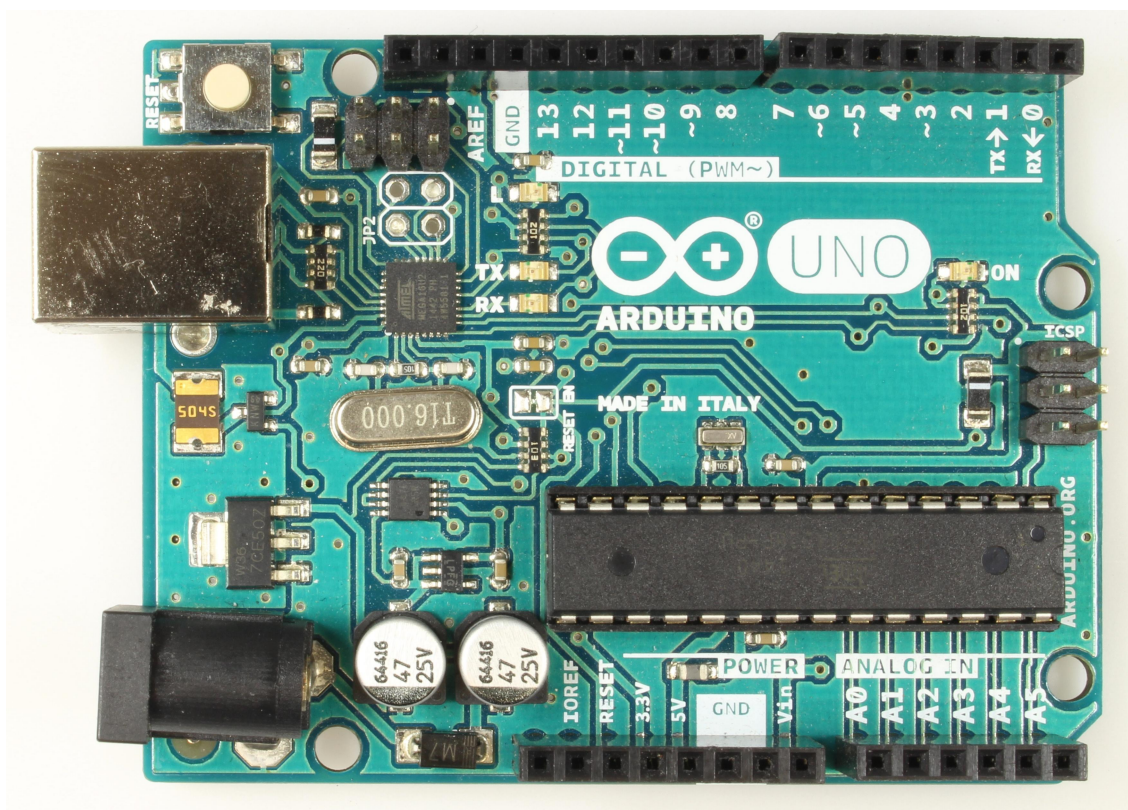
Δ΄.1.1 Μικροελεγκτής Arduino

Το Arduino Uno είναι μία ανοιχτού κώδικα πλακέτα με προσαρμοσμένο μικροελεγκτή για την πλακέτα, της ιταλικής εταιρείας Arduino, το οποίο κάποιος προμηθεύεται έτοιμο προς χρήση. Βασίζεται στον μικροεπεξεργαστή ATmega328 της εταιρείας Atmel, ο οποίος διαθέτει 32KB μνήμη (όπου τα 0.5KB χρησιμοποιούνται από τον φορτωτή εκκίνησης (bootloader), 2KB μνήμη τυχαίας προσπέλασης SRAM και 1KB μνήμη EEPROM. Η πλακέτα διαθέτει 14 ψηφιακούς ακροδέκτες I/O (εκ των οποίων οι 6

μπορούν να χρησιμοποιηθούν και ως έξοδοι PWM), 6 αναλογικές εισόδους, κρύσταλλο χαλαζία 16 MHz, θύρα υποδοχής USB, ενσωματωμένη κεφαλή ICSP και κουμπί επαναφοράς (reset). Η Arduino Uno είναι η πρώτη της σειράς πλακετών Arduino και ενδεχομένως η περισσότερο φημισμένη [23].

Για να λειτουργήσει χρειάζεται να τροφοδοτηθεί με ρεύμα μέσω ενός καλωδίου USB, μίας μπαταρίας ή τη χρήση ενός προσαρμογέα AC/DC.

Τα χαρακτηριστικά του μικροελεγκτή φαίνονται στον πίνακα Δ'.1.



Σχήμα Δ'.1: Arduino Uno

Δ'.1.2 Digi Xbee Pro S2C - ZigBee RF Module

Οι πομποδέκτες Xbee Pro S2C είναι προϊόντα της εταιρείας Digi International που παρέχουν χαμηλού κόστους ασύρματη διασύνδεση μεταξύ διαφόρων ηλεκτρονικών συσκευών βασισμένων στο πρωτόκολλο ZigBee και το 802.15.4. Η συγκεκριμένη σειρά

Specification	Arduino Uno
Μικροελεγκτής	ATmega328
Τάση Λειτουργίας	5V
Συνιστώμενη Τάση Εισόδου	7-12V
Τάση Εξόδου	6-20V
Αριθμός Ψηφιακών Ακροδεκτών	14 (εκ των οποίων 6 PWM)
Αριθμός Αναλογικών Ακροδεκτών	6
Ένταση Ρεύματος Ψηφιακού Ακροδέκτη	40mA
Ένταση Ρεύματος Αναλογικού Ακροδέκτη	50mA
Μνήμη Flash	32KB (εκ των οποίων 0.5KB για bootloader)
Μνήμη SRAM	2KB
Μνήμη EPROM	1KB
Ταχύτητα Ρολογιού	16MHz

Πίνακας Δ'.1: Χαρακτηριστικά του μικροελεγκτή Arduino Uno

πομποδεκτών χρησιμοποιείται ευρέως για οικιακές εφαρμογές καθώς οι μονάδες modules είναι εύκολο να ρυθμιστούν μέσα από την ειδική πλατφόρμα XCTU [24] που παρέχεται από την εταιρεία, μπορεί ωστόσο να παραγραμματιστούν και να χρησιμοποιηθούν για εφαρμογές με ειδικότερες ανάγκες [25].

Τα βασικά χαρακτηριστικά της μονάδας φαίνονται στον πίνακα Δ'.2.

Specification	Xbee Pro S2C
Εσωτερική/Αστική εμβέλεια	Έως 90 m
Εξωτερική εμβέλεια RF Γραμμή Οράσεως	Έως 32000 m
Μέγιστη ισχύς μετάδοσης	63 mW (+18 dBm)
RF Ρυθμός μετάδοσης	250.000 b/s
Ευαισθησία δέκτη	-101 dBm
Περιοχή συχνοτήτων λειτουργίας	ISM 2.4 - 2.5 GHz

Πίνακας Δ'.2: Χαρακτηριστικά του Xbee Pro S2C Module

Ένας πομποδέκτης Xbee υποστηρίζει τόσο την αυτόνομη λειτουργία όσο και την



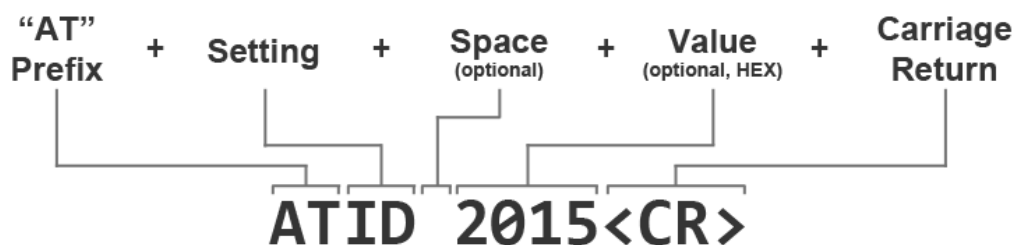
Σχήμα Δ'.2: Xbee Pro S2C Module

σειριακή επικοινωνία με μία έξυπνη συσκευή. Στην πρώτη περίπτωση αρκεί η μονάδα να τροφοδοτείται με ρεύμα, και η μόνη διαδικασία που εκτελείται είναι η δρομολόγηση συγκεκριμένων πακέτων (π.χ. μετρήσεις αισθητήρα) προς κάποιον κεντρικό κόμβο του δικτύου. Όταν όμως ο πομποδέκτης συνδεθεί με κάποια έξυπνη συσκευή, όπως έναν μικροελεγκτή, μπορεί να ανταλλάξει πληροφορία με αυτήν μέσω της σειριακής επικοινωνίας ακόμα και να δεχτεί εντολές από την έξυπνη συσκευή.

Η σειριακή διεπαφή μεταξύ της μονάδας και της έξυπνης συσκευής πραγματοποιείται με τη χρήση κυκλώματος UART (Universal Asynchronous Receiver/Transmitter) και οι μονάδες Xbee υποστηρίζουν δύο τρόπους λειτουργίας ή πρωτοκόλλων διεπαφής: Transparent και API.[20]

Δ'.1.3 Λειτουργία Transparent

Κατά τη λειτουργία Transparent, η μονάδα συμπεριφέρεται με τον ίδιο τρόπο όπως μία ενσύρματη σειριακή σύνδεση. Όλα τα πακέτα που λαμβάνονται από το UART μέσω του ακροδέκτη DIN εισέρχονται στον buffer της μονάδας για μετάδοση από την κεραία Xbee. Αντίθετα, όλα τα πακέτα που λαμβάνονται από την κεραία Xbee διαβάζονται μέσω του ακροδέκτη DOUT από το UART. Για τη ρύθμιση της ασύρματης μονάδας αποστέλλονται σε αυτήν οι ειδικές εντολές AT Commands, κάθε μία εκ των οποίων τροποποιεί κάποια παράμετρο της μονάδας.



Σχήμα Δ'.3: Μορφή εντολής AT. Η παράμετρος που αναλαμβάνει να τροποποιήσει η εντολή του σχήματος είναι το ID και η νέα τιμή της παραμέτρου το 2015

Δ'.1.4 Λειτουργία API

Η λειτουργία API βασίζεται στη χρήση δομημένων πλαισίων frames, τα οποία επεκτείνουν τις δικτυακές δυνατότητες της μονάδας. Τα frames μπορεί να περιλαμβάνουν πληροφορίες που ορίζουν λειτουργίες ή γεγονότα στη μονάδα. Η δρομολόγηση των πλαισίων μπορεί να πραγματοποιηθεί από επίπεδο εφαρμογής, η κάθε εφαρμογή δηλαδή μπορεί να ορίσει στη μονάδα τα δεδομένα προς αποστολή, τη διεύθυνση του παραλήπτη και το μέγεθος και το είδος του πλαισίου γενικότερα. Η μονάδα με τη σειρά της μπορεί να στέλνει πλαίσια προς την εφαρμογή καθορίζοντας έτσι τη ροή της εκτέλεσής της. Επιπρόσθετη χρήσιμη λειτουργία του πρωτοκόλλου API είναι η ύπαρξη αναφορών επιτυχών/ανεπιτυχών μεταδόσεων των πακέτων.

Η επικοινωνία μεταξύ δύο πομποδεκτών μπορεί να περιγραφεί μέσα σε 5 βήματα τα οποία φαίνονται στην εικόνα Δ'.4.

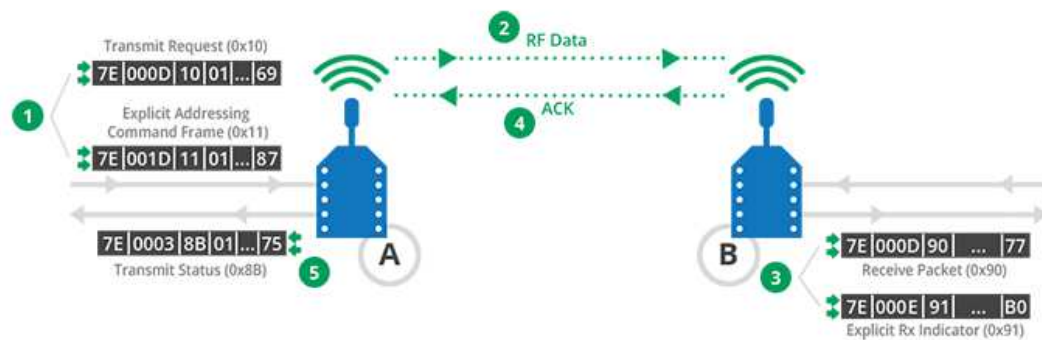
Βήμα 1: Αποστέλλεται το πλαίσιο αιτήματος μετάδοσης από την έξυπνη συσκευή προς τη μονάδα A μέσω της σειριακής εισόδου. Το πλαίσιο περιέχει, μεταξύ άλλων, τη διεύθυνση της ασύρματης συσκευής προορισμού.

Βήμα 2: Η μονάδα μεταδίδει ασύρματα τα δεδομένα στη μονάδα B χρησιμοποιώντας τη διεύθυνση του πλαισίου αιτήματος.

Βήμα 3: Η έξυπνη συσκευή της μονάδας B λαμβάνει τα δεδομένα του πλαισίου λήψης μετάδοσης μέσω της σειριακής εξόδου. Μέσα στο πλαίσιο περιέχεται η διεύθυνση του αποστολέα.

Βήμα 4: Η μονάδα B αποστέλλει πλαίσιο αναφοράς ACK στη μονάδα A.

Βήμα 5: Η μονάδα A λαμβάνει την αναφορά και την προωθεί στην έξυπνη συσκευή μέσω του σειριακού καναλιού επικοινωνίας.



Σχήμα Δ'.4: Σχηματική αναπαράσταση της επικοινωνίας μεταξύ δυο μονάδων Xbee σε λειτουργία API.

Συνοψίζοντας, η λειτουργία Transparent, αν και λιγότερο περίπλοκη, δε χρησιμοποιήθηκε για τις ανάγκες της εργασίας λόγω των περιορισμών που παρουσίαζε σε επίπεδο εφαρμογής και έτσι η εφαρμογή του δικτύου της εργασίας αναπτύχθηκε με βάση το πρωτόκολλο API.

Το δίκτυο ZigBee υλοποιήθηκε με τη χρήση μικροεπεξεργαστών Arduino Uno οι οποίοι αναλαμβάνουν τον ρόλο των κόμβων του δικτύου. Ο κάθε κόμβος αποτελείται από έναν μικροελεγκτή Arduino ο οποίος είναι συνδεδεμένος με μια μονάδα Xbee. Ο έλεγχος του πομποδέκτη επιτυγχάνεται μέσω του προγραμματισμού του μικροελεγ-

κτή (που λειτουργεί ως έξυπνη συσκευή) στο περιβάλλον ανάπτυξης προγραμμάτων Arduino IDE [26]. Η απευθείας σύνδεση του πομποδέκτη με τον μικροελεγκτή, δεν υποστηρίζεται και έτσι χρησιμοποιήθηκε το Xbee Arduino Shield, ένα εξάρτημα προσαρμογής που μεσολαβεί μεταξύ των δύο, και παρέχει την κατάλληλη διασύνδεση (βλ. Δ'.5).



Σχήμα Δ'.5: Μονάδα Xbee τοποθετημένη επάνω σε Xbee Shield για τη διασύνδεση της μονάδας Xbee με το Arduino Uno

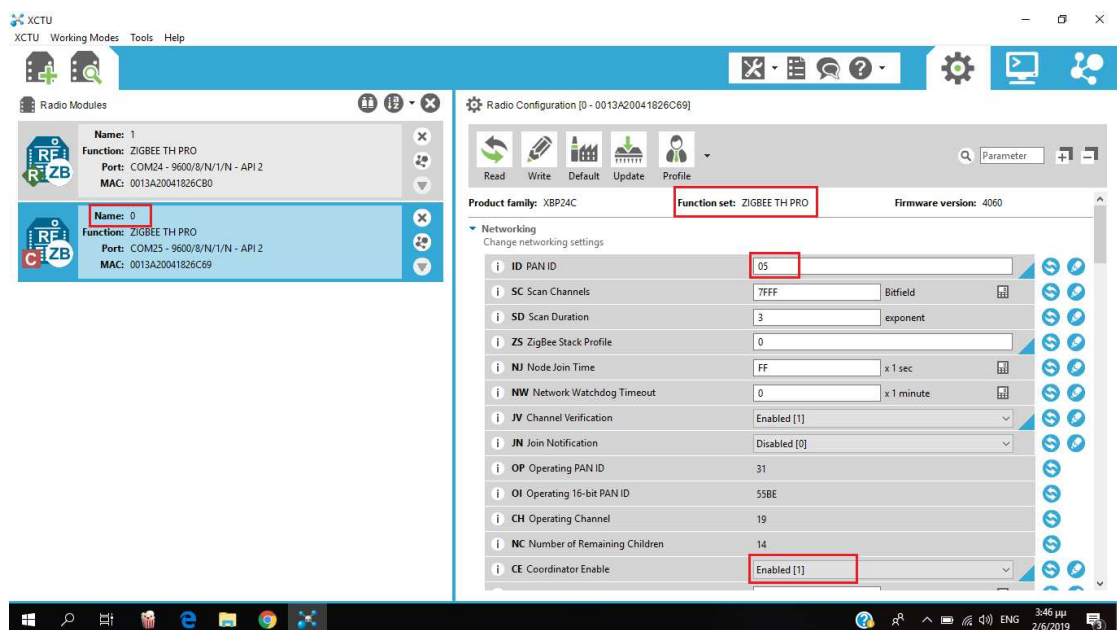
Δ'.1.5 Παράμετροι Συστήματος (Configurations)

Οι παράμετροι που επιλέγονται για τις μονάδες Xbee, όσο και το περιβάλλον της πλατφόρμας XCTU φαίνονται στις παρακάτω εικόνες.

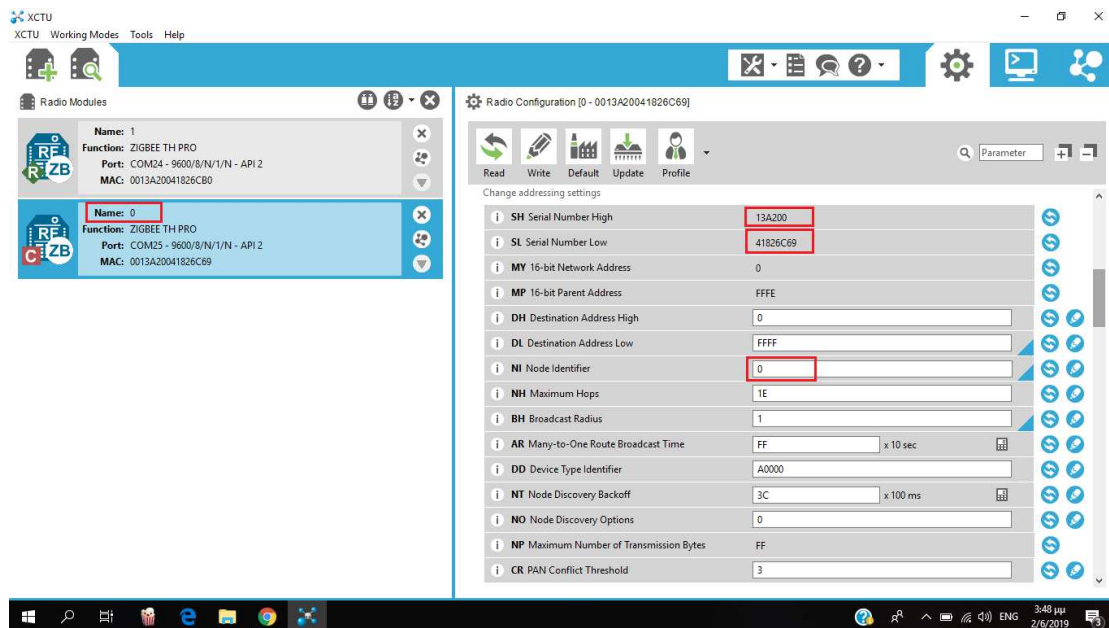
Στο παράδειγμα έχουν συνδεθεί στη πλατφόρμα δύο ενδεικτικές μονάδες. Με κόκκινο χρώμα επισημαίνονται οι κυρίως ρυθμίσεις και τα χαρακτηριστικά των μονάδων. Τέτοια αποτελούν το Function set της συσκευής, το PAN ID στο οποίο λειτουργεί το δίκτυο, τα Serial Number High και Serial Number Low της συσκευής, το αναγνωριστικό του κόμβου Node Identifier και η επιλογή Coordinator Enabled που ανάλογα με την τιμή της θέτει τον κόμβο σε συντονιστή.

Για τη μονάδα του δικτύου η οποία αναλαμβάνει τον ρόλο του Coordinator, τα χαρακτηριστικά φαίνονται στις εικόνες Δ'.6 και Δ'.7.

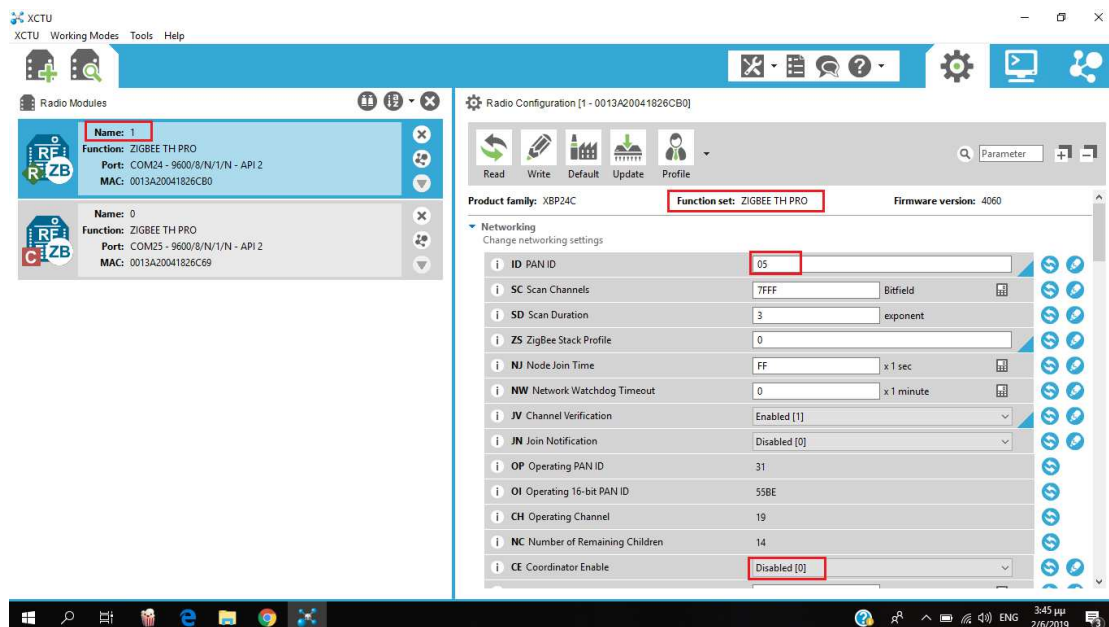
Για τη μονάδα του δικτύου η οποία αναλαμβάνει τον ρόλο του Router, τα χαρακτηριστικά φαίνονται στις εικόνες Δ'.8 και Δ'.9.



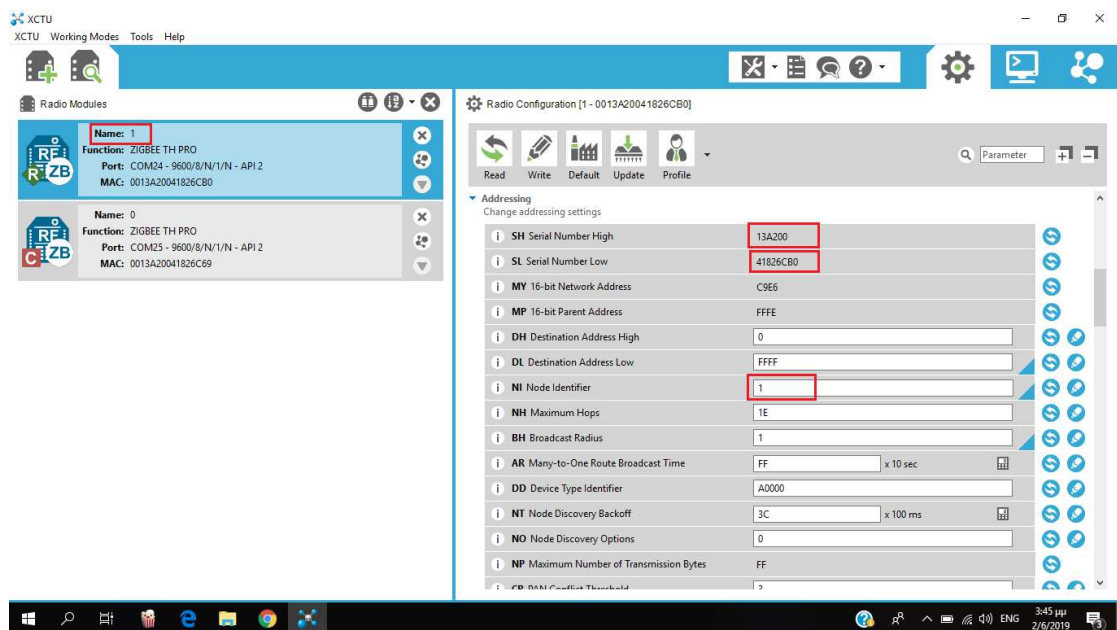
Σχήμα Δ'.6: Εμφάνιση των χαρακτηριστικών του κόμβου συντονιστή με Node Identifier 0 μέσω του περιβάλλοντος XCTU (α)



Σχήμα Δ'.7: Εμφάνιση των χαρακτηριστικών του κόμβου συντονιστή με Node Identifier 0 μέσω του περιβάλλοντος XCTU (β)



Σχήμα Δ'.8: Εμφάνιση των χαρακτηριστικών του κόμβου δρομολογητή με Node Identifier 1 μέσω του περιβάλλοντος XCTU (α)



Σχήμα Δ'.9: Εμφάνιση των χαρακτηριστικών του κόμβου δρομολογητή με Node Identifier 1 μέσω του περιβάλλοντος XCTU (α)

Δ'.2 Ανάπτυξη Εφαρμογής της Πιθανοτικής Πλημμυρίδας

Στην ενότητα αυτή περιγράφεται ο κώδικας της εφαρμογής που συντάχθηκε και φορτώθηκε στις συσκευές Arduino, με σκοπό αυτές να διαχειρίζονται κατάλληλα τις μονάδες Xbee.

Δ'.2.1 Γλώσσα Arduino

Η γλώσσα προγραμματισμού της πλατφόρμας Arduino βασίζεται στη γλώσσα προγραμματισμού C και μεταγλωττίζεται από C/C++ compiler αντίστοιχα. Στην ορολογία του Arduino, τα προγράμματα ονομάζονται sketches και έχουν επέκταση ονόματος αρχείου .ino.

Στο sketch που υλοποιήθηκε χρησιμοποιείται η βιβλιοθήκη xbee-arduino[27] για την διεπαφή των Xbee με τα Arduino σε λειτουργία API 2, η οποία παρέχει χρήσιμες

συναρτήσεις για την διαχείριση πολλών τύπων πακέτων του Xbee Pro S2C όπως είναι τα TX/RX και τα AT Commands. Για την χρήση της βιβλιοθήκης θα πρέπει τα αρχεία της να τοποθετηθούν στον υποφάκελο `libraries` του φακέλου των αρχείων Arduino. Ο φάκελος των αρχείων μπορεί να βρεθεί στις *Προτιμήσεις* του Arduino IDE.

Δ'.2.2 Δομή sketch

Οι δύο βασικές συναρτήσεις ενός sketch του Arduino, το οποίο αποτελεί πρόγραμμα κυκλικής εκτέλεσης, είναι η `setup()` και η `loop()`.

- Η `setup()` εκτελείται κατά την εκκίνηση του προγράμματος και αρχικοποιεί τις ρυθμίσεις και μεταβλητές.
- Η συνάρτηση `loop()` εκτελείται επαναληπτικά μέχρι η πλακέτα να απενεργοποιηθεί και περιλαμβάνει το μεγαλύτερο κομμάτι του κώδικα που εκτελεί το Arduino.

Αυτός ο κυκλικός τρόπος εκτέλεσης είναι ιδιαίτερα εύχρηστος για τη συγγραφή προγραμμάτων καταμετρημένου χαρακτήρα, καθώς το πρόγραμμα θα εκτελεσθεί από όλες τις πλακέτες- κόμβους του δικτύου, συμπεριλαμβανομένου του συντονιστή.

Δ'.2.3 Βιβλιοθήκη MyFrame

Ο βασικός στόχος της εφαρμογής της Πιθανοτικής Πλημμυρίδας είναι η διάδοση του μηνύματος της πλημμυρίδας από τον αρχικό κόμβο προς το όλους τους κόμβους του δικτύου. Προκύπτει έτσι η ανάγκη για την διαφοροποίηση των frames των μεταδιδόμενων μηνυμάτων στο δίκτυο.

Για την ταυτόχρονη αποστολής ενός αριθμού μηνυμάτων τύπου πλημμυρίδας, κρίθηκε απαραίτητος ο λογικός διαχωρισμός κάποιων frames τύπου πλημμυρίδας τους, καθώς το κάθε ένα χρήζει διαφορετικής αντιμετώπισης από τους κόμβους.

Για την διευκόλυνση στον παραπάνω διαχωρισμό των frames υλοποιήθηκε η βιβλιοθήκη **MyFrame.h**, η οποία παρατίθεται στο παράρτημα. Η βιβλιοθήκη ορίζει το frame με τη κλάση `MyFrame()` και αξιοποιεί τη λογική των ετικετών (πεδία τύπου και αναγνωριστικού) οι οποίες περιλαμβάνουν τα δεδομένα του μηνύματος. Το πεδίο του τύπου

```
1  /*
2   * Blink
3   * Turns on an LED on for one second, then off for one second, repeatedly.
4   *
5   * Most Arduinos have an on-board LED you can control. On the Uno and
6   * Leonardo, it is attached to digital pin 13. If you're unsure what
7   * pin the on-board LED is connected to on your Arduino model, check
8   * the documentation at http://arduino.cc
9   *
10  * This example code is in the public domain.
11  *
12  * modified 8 May 2014
13  * by Scott Fitzgerald
14  */
15
16
17 // the setup function runs once when you press reset or power the board
18 void setup() {
19   // initialize digital pin 13 as an output.
20   pinMode(13, OUTPUT);
21 }
22
23 // the loop function runs over and over again forever
24 void loop() {
25   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
26   delay(1000);            // wait for a second
27   digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
28   delay(1000);            // wait for a second
29 }
```

Done uploading.

Σχήμα Δ'.10: Παράδειγμα προγράμματος σε γλώσσα Arduino με το οποίο αναβοσβήνει περιοδικά μια λάμπα συνδεδεμένη στην πλακέτα. Στο μέρος `setup()` ορίζεται ο ακροδέκτης για μια λάμπα τύπου LED, ενώ στο `loop()` η λειτουργικότητα της λάμπας για όσον η πλακέτα παραμένει σε κατάσταση τροφοδοσίας.

της κλάσης `FrameType` αποδίδεται με τη μορφή `enumerator` για να είναι περιγραφι-

κός αλλά ταυτόχρονα να εξασφαλίζεται πως θα μπορεί να παίρνει μόνο συγκεκριμένες τιμές της δομής. Το πεδίο του αναγνωριστικού αποδίδεται με `uint8_t`. Στην κλάση ορίζονται διάφοροι κατασκευαστές, καθώς και μέθοδοι `set()` και `get()` για την ανάγνωση και τροποποίηση των πεδίων της κλάσης. Για τη διαχείριση των μηνυμάτων του προγράμματος χρησιμοποιήθηκαν τρεις τύποι πακέτου:

- SYNC - Μήνυμα συγχρονισμού
- FLOOD - Μήνυμα πλημμυρίδας
- RESULT - Μήνυμα αποτελέσματος πειράματος

Κατά την κλήση της μεθόδου `uint8_t* serialize()` όλο το περιεχόμενο των πεδίων του πλαισίου επιστρέφεται ως πίνακα (`_frame`) με μήκος το μήκος του πλαισίου, η δομή του οποίου φαίνεται στην εικόνα Δ'.3. Αντίστοιχα σε περίπτωση που θέλουμε μεταφράσουμε έναν πίνακα δεδομένων σε αντικείμενο `myFrame`, είναι απαραίτητο να κληθεί η μέθοδος `void deserialize(uint8_t* frame, uint8_t frameLength)` με ορίσματα τον πίνακα και το μήκος του.

Frame Type 1 byte	ID 1 byte	Data Length 1 byte	Data Maximum Length: 252 bytes
----------------------	--------------	-----------------------	-----------------------------------

Πίνακας Δ'.3: Πίνακας πλαισίου `_frame` της κλάσης `myFrame`. Οι πρώτες τρεις θέσεις του πίνακα περιέχουν τον τύπο του πλαισίου, το αναγνωριστικό και το μήκος των δεδομένων αντίστοιχα. Το μεγαλύτερο δυνατό μέγεθος πλαισίου σε bytes που μπορεί να υποστηρίξει το ZigBee πρωτόκολλο είναι 255 και έτσι το μέγιστο μέγεθος των δεδομένων του πλαισίου `myFrame` ανέρχεται στα $255 - 3 = 252$ bytes.

Δ'.2.4 Περιγραφή Προγράμματος

Πριν την ανάλυση της λειτουργικότητας του προγράμματος, αξίζει να αναφερθούν κάποιες από τις βασικές δομές, μεταβλητές και συναρτήσεις που χρησιμοποιούνται σε αυτό.

Σταθερές

Οι σταθερές που καθορίζονται με την οδηγία `define` και ισχύουν για όλους τους κόμβους του δικτύου είναι:

- **NUMNODES**: Ο αριθμός των συνολικών κόμβων του δικτύου
- **PROBABILITY**: Ο παρανομαστής της τιμής $Q = \frac{1}{PROBABILITY}$ όπου Q η πιθανότητα αποστολής του μηνύματος
- **N**: Ο αριθμός των μηνυμάτων της πλημμυρίδας που θα αποσταλούν από τον συντονιστή
- **DATASIZE**: Ο μεγαλύτερος αριθμός των γειτόνων που μπορεί να έχει ένας κόμβος του δικτύου

Ο ορισμός αυτών των αριθμών στην αρχή του προγράμματος καθιστά εύκολη την αλλαγή τους για διαφορετικές επαναλήψεις πειραμάτων.

Μεταβλητές

Οι βασικές μεταβλητές και δομές δεδομένων που χρησιμοποιούνται στο πρόγραμμα είναι:

- Η μεταβλητή **zigbeeAddressHigh** τύπου `uint32_t` που αντιπροσωπεύει τα πρώτα 32 bits της 64 bit MAC διεύθυνσης των μονάδων του δικτύου (είναι ίδια για όλες της συσκευές του δικτύου).
- Η `bool` μεταβλητή **isCoordinatorEnabled** που παίρνει τιμή ανάλογα με το αν η συσκευή στην οποία τρέχει το πρόγραμμα είναι ο συντονιστής του δικτύου ή όχι.
- Το αντικείμενο **xbee** κλάσης **XBee**. Η κλάση `XBee` αποτελεί τη διεπαφή του μικροελεγκτή με τη μονάδα και παρέχει διάφορους μεθόδους αποστολής και λήψης πλαισίων μέσω της σειριακής διεπαφής. Όσον αφορά τα πακέτα που διαβάζονται από τον μικροελεγκτή, η κλάση δημιουργεί έναν πίνακα στον οποίον αποθηκεύεται μόνο ένα πλαίσιο λήψης την φορά. Αυτό πρακτικά σημαίνει, πως μετά από την κάθε κλήση της μεθόδου `readPacket()`, το πλαίσιο του πίνακα επανεγγράφεται σε περίπτωση ύπαρξης διαφορετικού πακέτου στη μονάδα.

Οι βασικές μέθοδοι της κλάσης και αυτές που χρησιμοποιούνται στο πρόγραμμα είναι:

- **void readPacket ()** : διαβάζει τα bytes που υπάρχουν στην σειριακή διεπαφή μέχρι το τέλος ενός πακέτου.
 - **bool readPacket (int timeout)** : Σε περίπτωση που διαβάσει κάποιο πακέτο εντός του χρονικού διαστήματος timeout σε milliseconds, επιστρέφει true.
 - **void send (XBeeRequest &request)** : Αποστέλλει το αίτημα κλάσης XBeeRequest από τη σειριακή διεπαφή στη μονάδα για να το προωθήσει αντίστοιχα.
- Το αντικείμενο **zbTx** της κλάσης **ZBTxRequest**, που αντιπροσωπεύει πακέτο αιτήματος TX. Ο κατασκευαστής της κλάσης που χρησιμοποιείται είναι ο **ZBTxRequest (XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)**, με ορίσματα: τη διεύθυνση της συσκευής για την οποία προορίζεται το αίτημα, τα δεδομένα (με μορφή πίνακα bytes uint8_t) και το μέγεθος των δεδομένων (μέγεθος πίνακα).
 - Το αντικείμενο **addr64** της κλάσης **XBeeAddress64**, για τη διευθυνσιοδότηση μιας μονάδας. Διατηρεί ολόκληρη τη MAC διεύθυνση μιας συσκευής, δηλαδή το πρώτο και το δεύτερο μέρος της 64 bit διεύθυνσής της.
 - Η μεταβλητή **parentAddressLow** τύπου uint32_t που αντιπροσωπεύει το δεύτερο μέρος με τα 32 bits της 64 bit διεύθυνσης του πατέρα. Πατέρας θεωρείται κάθε φορά ο τελευταίος αποστολέας, ο κάθε κόμβος δηλαδή κρατάει προσωρινά για πατέρα τον αποστολέα του τελευταίου πακέτου που έλαβε. Η πληροφορία αυτή χρησιμεύει στην διαδικασία αποστολής μηνυμάτων απάντησης.
 - Πίνακας **uint8_t sData[4]** για τη διατήρηση τεσσάρων bytes δεδομένων. Χρησιμοποιείται για την αντιγραφή δεδομένων μηνύματος στον πίνακα των προς αποστολή πλαισίων.
 - Τα αντικείμενα **receiveFrame** και **sendFrame** της κλάσης **myFrame**, τα οποία διατηρούν τα αντίστοιχα πλαίσια της μονάδας. Συγκεκριμένα, κάθε φορά που διαβάζονται δεδομένα από τη σειριακή διεπαφή, αυτά αντιγράφονται από τον

πίνακα του αντικειμένου xbee στο αντικείμενο receiveFrame, ενώ πριν από κάθε αποστολή τα δεδομένα του sendFrame μεταφέρονται προς το αντικείμενο αιτήματος zbTx της κλάσης ZBTxRequest.

- Ο πίνακας **neighboursLow[DATASIZE]** με τις διευθύνσεις 32 bit των γειτόνων του κόμβου. Παίρνει τις κατάλληλες τιμές για τον κάθε κόμβο ανάλογα με τις απαντήσεις των εντολών AT. Το μέγεθός του (**DATASIZE**) είναι ο μεγαλύτερος αριθμός των γειτόνων που μπορεί να έχει ένας κόμβος του δικτύου.
- Η μεταβλητή **neighboursCount** τύπου `uint8_t` για τον αριθμό των γειτόνων του κόμβου. Παίρνει τις κατάλληλες τιμές για τον κάθε κόμβο ανάλογα με τις απαντήσεις των εντολών AT.
- Για την αποθήκευση των αποτελεσμάτων των πειραμάτων της πλημμυρίδας αρχικά υλοποιήθηκε η δομή **struct Measur** που αποτελείται από τα δύο πεδία `uint8_t TTL` και `unsigned long time` και έπειτα ο πίνακας **Measur measurements[N][NUMNODES]**, ο οποίος κρατάει τα αποτελέσματα των μετρήσεων του κάθε κόμβου και για κάθε διαφορετική πλημμυρίδα.
- Ο πίνακας **myFloodMessage[3]** και η `int8_t` μεταβλητή **myFloodMessageLength = 3** για τα περιεχόμενα ενός μηνύματος πλημμυρίδας. Τα περιεχόμενα του πίνακα είναι το id του μηνύματος πλημμυρίδας στην πρώτη θέση, η μεταβλητή **PROBABILITY** στη δεύτερη θέση και στην τρίτη θέση το **TTL** που αντιπροσωπεύει τον αριθμό των βημάτων που έχει διανύσει το μήνυμα.

Συναρτήσεις

Οι βασικές συναρτήσεις του προγράμματος:

- **void sendZBTxFrame(uint8_t BH, uint32_t addrHigh, uint32_t addrLow)**

Βασική συνάρτηση αποστολής του αντικειμένου xbee από την κεραία της μονάδας προς την κατάλληλη ασύρματη διεύθυνση. Παίρνει σαν παράμετρο το Broadcast Radius που προσδιορίζει τον μέγιστο αριθμό hops της μετάδοσης, την ασύρματη διεύθυνση της συσκευής προορισμού. Εντός της συνάρτησης δημιουργείται αντικείμενο διεύθυνσης και ύστερα αντικείμενο ZBTxRequest στο οποίο μεταβιβάζεται το sendFrame, τίθεται η τιμή των hops της μετάδοσης και αποστέλλεται

το πακέτο μέσω του αντικειμένου xbee. Η τιμή του `sendFrame` στο τέλος της διαδικασίας τίθεται στο `default` της.

- **`void checkForPacketDelivery()`**

Συνάρτηση που πραγματοποιεί τη διαδικασία ελέγχου, για την ύπαρξη κάποιου καινούργιου `frame` εντός της σειριακής διεπαφής. Πιο συγκεκριμένα, προετοιμάζεται το `frame` της λήψης (`receiveFrame`) για την εγγραφή σε αυτό του καινούργιου πακέτου, και ύστερα καλείται η συνάρτηση `xbee.readPacket()`, με την οποία διαβάζονται ένα ένα τα `bytes` της σειριακής διεπαφής, μέχρις ότου να βρεθεί κάποιο πακέτο, να προκύψει κάποιο σφάλμα ή να εξαντληθούν τα `bytes`. Έπειτα, ελέγχεται για το εάν υπάρχει κάποιο `frame` στην απάντηση της παραπάνω διαδικασίας μέσω της εντολής `xbee.getResponse().isAvailable()`, ούτως ώστε να γίνει ο κατάλληλος χειρισμός των νέων δεδομένων. Συνοπτικά, σχετικά με τον χειρισμό των δεδομένων, ακολουθεί η αποθήκευση της διεύθυνσης του αποστολέα του καινούργιου πακέτου, και έπειτα με τη `deserialize()` τα δεδομένα (μαζί με τον τύπο του πλαισίου) εισάγονται στο `receiveFrame`.

- **`void sendATCommand(String command)`**

Η συνάρτηση που αναλαμβάνει την αποστολή των ειδικών εντολών (AT Commands) προς τη μονάδα, που ως σκοπό έχουν να αλλάξουν τις ρυθμίσεις της μονάδας. Για την κάθε μονάδα, υπάρχει ένας αριθμός ρυθμίσεων που μπορούν να αλλάξουν, όπως για παράδειγμα το αναγνωριστικό ID δικτύου που έχει. Αυτές οι ρυθμίσεις μπορούν να αλλάξουν και από το περιβάλλον του XCTU. Η βιβλιοθήκη XBee.h παρέχει κλάσεις ειδικού τύπου για αιτήματα και απαντήσεις εντολών AT. Η συνάρτηση παίρνει ως όρισμα μια συμβολοσειρά που αντιπροσωπεύει την εντολή και την χωρίζει σε δύο μέρη: την ονομασία της παραμέτρου και την τιμή της, εάν αυτή υπάρχει, την οποία μετατρέπει σε αριθμό και τα περνάει ως ορίσματα στο αντικείμενο `atRequest` της κλάσης `AtCommandRequest`, το οποίο στέλνεται στην κεραία (βλ. εικόνα Δ'.3). Η συνάρτηση έπειτα μπαίνει στη διαδικασία της λήψης απάντησης από τη μονάδα με τον ως εξής: με συνεχόμενο τρόπο γίνεται ανάγνωση πακέτου απάντησης (με όρισμα το χρονικό διάστημα 6 milliseconds) μέχρι να βρεθεί αίτημα απάντησης AT εντολής είτε να προκύψει σφάλμα. Σε περίπτωση επιτυχίας και ανάλογα με την εντολή εκτελούνται κάποιες επιρόσθητες διεργασίες.

- **void unicastNeighboursProb(bool sendToParent, uint8_t prob)**

Η συνάρτηση που καλείται για την προώθηση του μηνύματος της πλημμυρίδας. Με το όρισμα `sendToParent` αποφασίζεται εάν η πλημμυρίδα θα προωθηθεί στον πατρικό κόμβο, ενώ το όρισμα `prob` αντιπροσωπεύει την παράμετρο πιθανότητας της πλημμυρίδας. Η συνάρτηση επιχειρεί να πραγματοποιήσει αποστολή στον κάθε γείτονα από τον πίνακα γειτόνων. Για να μεταδοθεί το πακέτο θα πρέπει η συνθήκη `rand() % prob < 1` να αποτιμηθεί σε αληθής. Η πιθανότητα της συνθήκης να αποτιμηθεί σε αληθής είναι ίση με $Q = \frac{1}{prob}$. Σημειώνεται, πως η πραγματική τυχαιότητα της `rand()` επιτυγχάνεται με τη χρήση μιας αναλογικής εισόδου του μικροελεγκτή ως τυχαίου `seed` της συνάρτησης: `srand(analogRead(0))`.

Κυρίως Μέρος

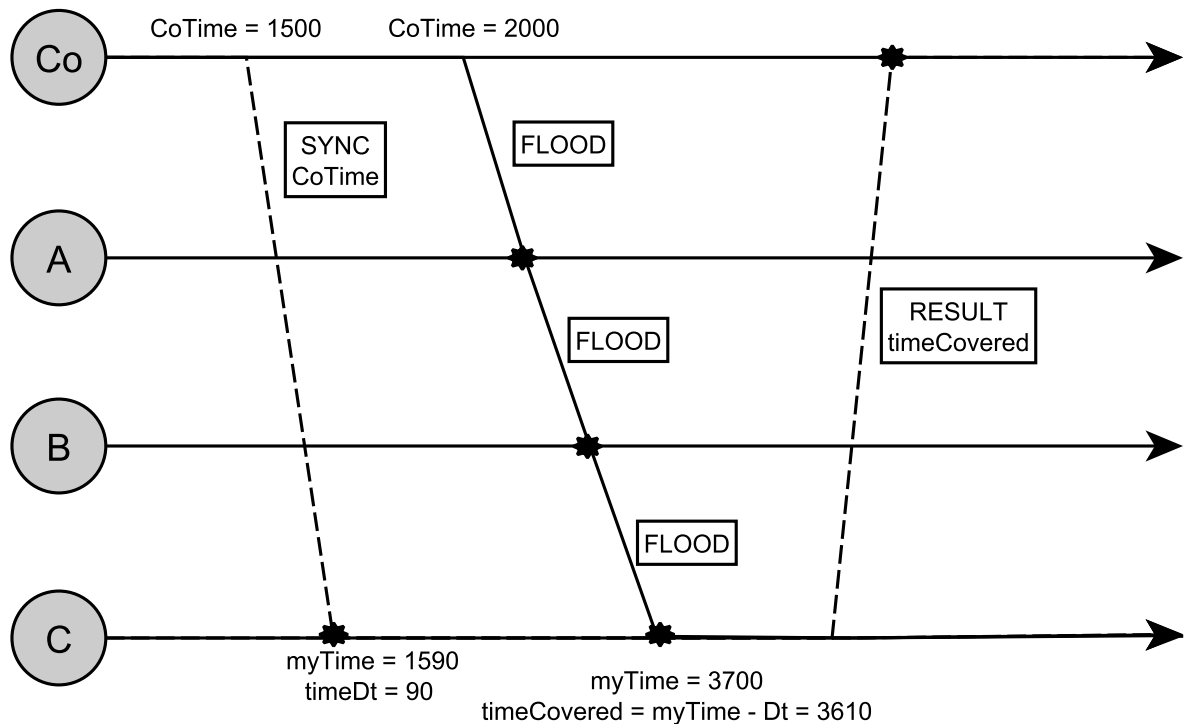
`setup()`

Οι εντολές του μέρους `setup()` του προγράμματος εκτελούνται μία φορά κατά την έναρξη της εκτέλεσής του. Συγκεκριμένα, σε αυτό το μέρος ορίζεται το baud rate της σειριακής επικοινωνίας στα 9600 baud και στέλνονται οι εντολές AT προς τη μονάδα, έτσι ώστε να ρυθμιστούν κάποιες παράμετροι και να αποδοθεί τιμή στη μεταβλητή `isCoordinatorEnabled` και στον πίνακα με τους γείτονες. Η απόδοση τιμής σε αυτή τη μεταβλητή είναι υποχρεωτική για να δοθεί το έναυσμα για τη μετάδοσης της πλημμυρίδας.

Όπως προαναφέρθηκε, πριν την μετάδοση των μηνυμάτων πλημμυρίδας εκτελείται ένα είδος συγχρονισμού μεταξύ των κόμβων του δικτύου. Η λογική είναι οι κόμβοι να κρατήσουν την χρονική διαφορά του ρολογιού τους από το ρολόι του συντονιστή, ώστε στο τέλος η χρονική πληροφορία που θα λάβει ο συντονιστής να είναι προσαρμοσμένη στο ρολόι του συντονιστή. Η λογική αυτή απεικονίζεται σχηματικά στο σχήμα Δ'.11.

Σε περίπτωση που ο κόμβος είναι ο συντονιστής ορίζεται ο τύπος του `sendFrame` σε SYNC και πραγματοποιείται η μέτρηση του ρολογιού του συντονιστή με τη συνάρτηση `micros()`. Τα δεδομένα της μέτρησης αντιγράφονται στον βοηθητικό πίνακα των δεδομένων `sData` και έπειτα στο `sendFrame`. Σκοπός είναι να λάβουν το μήνυμα όλοι οι κόμβοι του δικτύου, στο συντομότερο δυνατό χρονικό διάστημα. Χρησιμοποιείται έτσι η μέθοδος μετάδοσης **broadcast** που υλοποιείται από το πρωτόκολλο ZigBee. Κατά τη μέθοδο αυτή, κάθε συσκευή που λαμβάνει το πακέτο, το αναμεταδίδει τρεις φορές σε όλους τους κόμβους για να επιτευχθεί η κάλυψη. Για να πραγματοποιηθεί μια

μετάδοση broadcast τα 16 τελευταία bits της 64 bit διεύθυνσης του πακέτου τίθενται σε 1. Η διαχείριση αυτών των πακέτων από τους κόμβους περιγράφεται παρακάτω στο μέρος **loop()**.



Σχήμα Δ'.11: Η λογική του συγχρονισμού που υλοποιείται. Ο κόμβος Co είναι ο συντονιστής. Πραγματοποιείται η αποστολή του μηνύματος SYNC από τον κόμβο Co στον κόμβο C, με σκοπό ο κόμβος C να αποθηκεύσει τη χρονική διαφορά Dt του ρολογιού του από το ρολόι του Co. Σε μεταγενέστερη φάση, ο κόμβος C λαμβάνει μήνυμα τύπου FLOOD από γειτονικό του κόμβο B. Για να βρει τον χρόνο παραλαβής του FLOOD σύμφωνα με το ρολόι του συντονιστή, αφαιρεί τη χρονική διαφορά Dt από τον χρόνο παραλαβής της πλημμυρίδας. Η πληροφορία αυτή αποστέλλεται με μήνυμα τύπου RESULT, κατευθείαν στον συντονιστή.

Η τελευταία λειτουργία στο **setup()** που πραγματοποιείται από τον συντονιστή, είναι η έναρξη της διάδοσης της μίας εκ των **N** επαναλήψεων πλημμυρίδας. Στη μεταβλητή

`floodTempTime` κρατείται ο χρόνος της έναρξης της πλημμυρίδας. Για να αποσταλεί η πλημμυρίδα είναι απαραίτητο να οριστεί η πληροφορία που θα εμπεριέχεται στο μήνυμα με τη βοήθεια του πίνακα `myFloodMessage`. Για τον πίνακα αυτόν ορίζεται το `id` της πλημμυρίδας σε 0, το `TTL` του μηνύματος (0 στην αρχή της διάδοσης) και η τιμή `PROBABILITY`. Η αποστολή γίνεται για κάθε γείτονα με τη χρήση της συνάρτησης `unicastNeighboursProb()`, η λειτουργίας της οποίας περιγράφεται παραπάνω, και με πιθανότητα $\frac{1}{PROBABILITY}$. Ο μετρητής των πραγματοποιημένων πλημμυρίδων `floodCounter` παίρνει την τιμή 1 σε αυτό το σημείο.

`loop()`

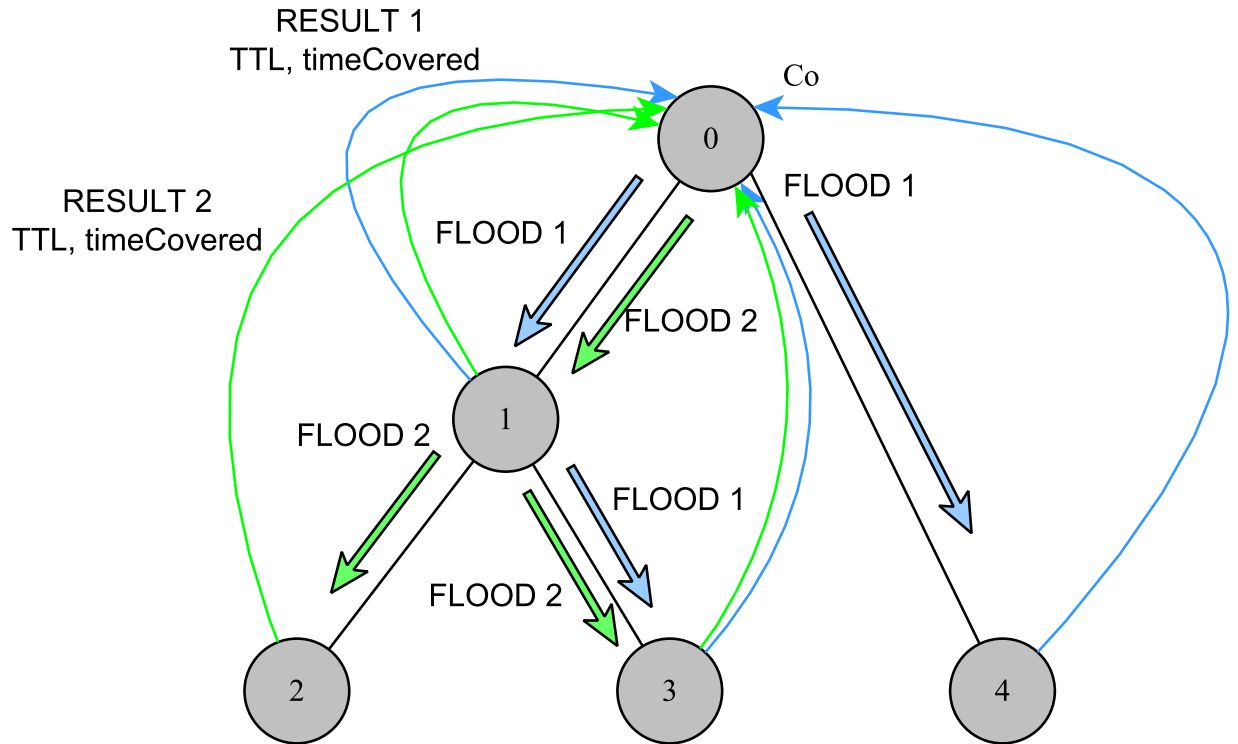
Στο επαναληπτικό μέρος του προγράμματος καλείται κάθε φορά η συνάρτηση `checkForPacketDelivery()` με σκοπό να ανανεωθεί το `receiveFrame`. Ανάλογα με τον τύπο του ληφθέντος `frame` εκτελείται ένα από τα παρακάτω:

- **SYNC**: Ο τύπος που εντοπίστηκε στη `receiveFrame` είναι `SYNC`. Πρόκειται για μήνυμα συγχρονισμού. Σύμφωνα με τον αλγόριθμο συγχρονισμού που υλοποιείται, μόλις ληφθεί ένα τέτοιο μήνυμα από κάποιον κόμβο του δικτύου, αποθηκεύεται η χρονική τιμή του ρολογιού τη στιγμή της λήψης, καθώς και η χρονική διαφορά (`timeDt`) του ρολογιού του συγκεκριμένου κόμβου από το ρολόι του συντονιστή (από το περιεχόμενο του μηνύματος).
- **FLOOD**: Ο τύπος που εντοπίστηκε στη `receiveFrame` είναι `FLOOD`. Πρόκειται για μήνυμα πλημμυρίδας. Κατά τη λήψη του, αποθηκεύεται ο χρόνος λήψης και έπειτα διαβάζεται το περιεχόμενο του μηνύματος. Έπειτα, με τη προϋπόθεση πως η πλημμυρίδα με το συγκεκριμένο `ID` λαμβάνεται για πρώτη φορά, εκτελούνται τα παρακάτω:

Οι μετρήσεις αποθηκεύονται σε κατάλληλο πίνακα του κόμβου. Τις μετρήσεις αυτές, εκτός από την πληροφορία για τη λήψη της πλημμυρίδας, αποτελούν το `TTL` και ο πραγματικός χρόνος κάλυψης του κόμβου σύμφωνα με το ρολόι του συντονιστή, ο οποίος προκύπτει αφαιρώντας τη χρονική διαφορά `timeDt` από το χρόνο κάλυψης του κόμβου. Οι μετρήσεις αυτές μεταβιβάζονται σε πακέτο και αποστέλλονται απευθείας στον συντονιστή. Τέλος, ο κόμβος αναλαμβάνει να συνεχίσει τη προώθηση της πλημμυρίδας στους γειτονικούς του κόμβους με τη χρήση της συνάρτησης `unicastNeighboursProb` (βλ. Συναρτήσεις).

- **RESULT:** Ο τύπος που εντοπίστηκε στη `receiveFrame` είναι **RESULT**. Το συγκεκριμένο μήνυμα μπορεί να ληφθεί μόνο από τον συντονιστή του δικτύου. Κατά τη λήψη του, τα περιεχόμενα του πακέτου αποθηκεύονται σε κατάλληλο πίνακα `measurements` του συντονιστή.

Η τελευταία λειτουργία που παίρνει μέρος στο `loop()` συμβαίνει αποκλειστικά στον κόμβο συντονιστή. Ελέγχεται εάν έχουν περάσει 5.000.000 microseconds από την αρχή της διάδοσης της πλημμυρίδας από τον συντονιστή, και εάν οι συνολικές αποστολές πλημμυρίδων του συντονιστή δε ξεπερνάνε τον απαιτούμενο αριθμό (N). Στη περίπτωση που συμβαίνουν τα παραπάνω πραγματοποιείται η αποστολή της επόμενης πλημμυρίδας με ανανεωμένα περιεχόμενα, αυξάνεται ο μετρητής `floodCounter` και ανανεώνεται ο χρόνος `floodTempTime`. Εάν διαφορετικά έχουν περάσει 5.000.000 microseconds και ο μετρητής `floodCounter` έχει γίνει ίσος με N, η διαδικασία της πλημμυρίδας έχει τελειώσει, συνεπώς εκτυπώνονται τα περιεχόμενα του πίνακα αποτελεσμάτων του συντονιστή. Σημειώνεται, πως το χρονικό περιθώριο των 5.000.000 microseconds, πειραματικά επιβεβαιώθηκε επαρκές για τη λήψη όλων των μηνυμάτων αποτελέσματος **RESULTS** των κόμβων του δικτύου. Σημειώνεται επίσης, πως ο λόγος της αποστολής των μηνυμάτων πλημμυρίδας με τη χρήση του διαστήματος αναμονής, υλοποιήθηκε για το σκοπό της αποφυγής συμφόρησης του δικτύου.

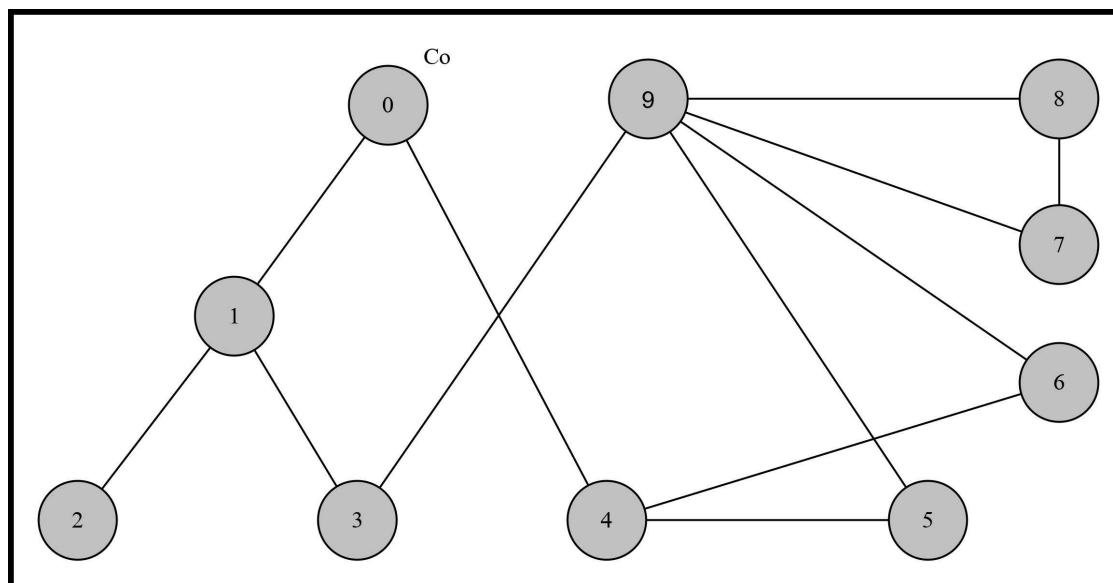


Σχήμα Δ'.12: Παράδειγμα μετάδοσης της Πιθανοτικής Πλημμυρίδας σε ένα δίκτυο μεγέθους 5 κόμβων, με αριθμό επαναλήψεων πλημμυρίδας $N = 2$. Οι μεταδόσεις των πλαισίων της κάθε διαφορετικής επανάληψης απεικονίζονται με ξεχωριστό χρώμα. Ο κόμβος 0 είναι ο συντονιστής, ο οποίος εκκινεί τη διαδικασία. Με τα περιγραμμισμένα βέλη απεικονίζονται οι μεταδόσεις των πλαισίων της Πλημμυρίδας. Με τα βέλη χωρίς τα περιγράμματα απεικονίζονται τα πλαίσια απάντησης των κόμβων. Κάθε κόμβος που λαμβάνει μια διαφορετική Πλημμυρίδα, οφείλει να απαντήσει στον συντονιστή.

Δ'.3 Πειράματα

Για το πρόγραμμα που εξηγήθηκε παραπάνω, η διαδικασία διεξαγωγής πειραμάτων εκτελείται με τον τρόπο που περιγράφεται στην παρούσα ενότητα.

Αρχικά, παρουσιάζεται η τοπολογία που χρησιμοποιήθηκε στα πειράματα:



Σχήμα Δ'.13: Τοπολογία διεξαγωγής πειραμάτων για δίκτυο μεγέθους 10 κόμβων. Ο κόμβος 0 έχει τον ρόλο του συντονιστή του δικτύου.

Το δίκτυο που χρησιμοποιήθηκε αποτελείται από 10 κόμβους.

Οι πιθανότητες που χρησιμοποιήθηκαν για τα πειράματα φαίνονται στον παρακάτω πίνακα:

Πιθανότητα
1
1/2
1/3
1/4
1/5

Πίνακας Δ'.4: Τιμές των πιθανοτήτων των πειραμάτων

Για την κάθε τιμή της πιθανότητας πραγματοποιήθηκαν $N = 20$ διαφορετικές επαναλήψεις. Για την ακρίβεια το πρόγραμμα εκτελέστηκε από το δίκτυο 2 φορές για κάθε διαφορετική τιμή της πιθανότητας με παράμετρο $N = 10$, για την αποφυγή

περαιτέρω συμφόρησης του δικτύου και μείωση του χρονικού διαστήματος αναμονής για το πέρας κάθε πειράματος, έτσι ώστε να διευκολύνονται τυχών δοκιμές και επαναλήψεις λόγω προβλημάτων εκτέλεσης.

Η καταγραφή των αποτελεσμάτων έγινε συλλέγοντας τις τελικές τιμές που εκτυπώνονταν με τη κατάλληλη μορφοποίηση στο Serial Monitor της συσκευής του συντονιστή, και αποθηκεύοντας τες σε αρχεία κειμένου.

Ενδεικτικά, παρακάτω φαίνονται οι τιμές δύο επαναλήψεων ενός κάποιου πειράματος με παράμετρο πιθανότητας $p = 1/2$.

0 0	0 0	0 0	0 0	1 11891360	0 0	2 12253508	4 13125068	0 0	3 12688528
0 0	1 16761488	0 0	0 0	1 16837588	0 0	0 0	0 0	0 0	0 0

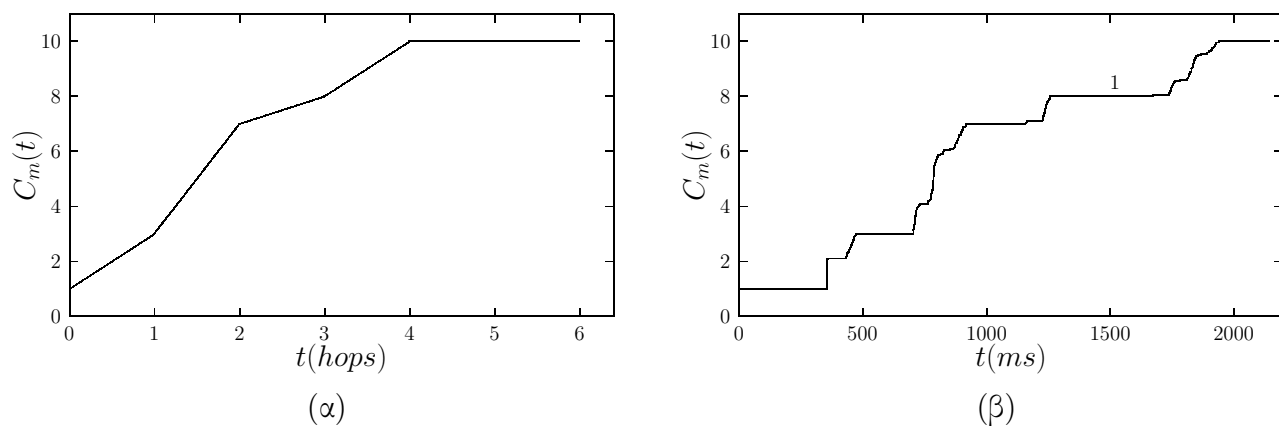
Πίνακας Δ'.5: Μορφή δεδομένων των αποτελεσμάτων πειράματος

Έπειτα ακολούθησε η διαδικασία της αναδιοργάνωσης των δεδομένων, ώστε να είναι δυνατή η οπτικοποίησή τους στο επόμενο στάδιο. Τα δεδομένα οργανώθηκαν σε ξεχωριστά αρχεία κειμένου σύμφωνα με την τιμή της κάθε πιθανότητας, την τιμή του χρόνου κάλυψης και τον συνολικό αριθμό των καλυφθέντων κόμβων.

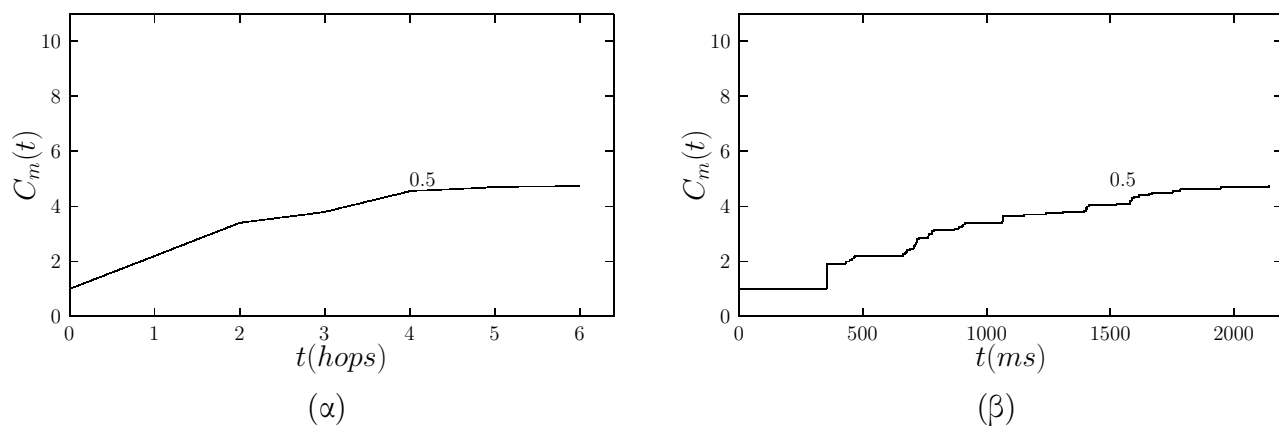
Τέλος, από τα αρχεία αυτά, προέκυψαν τα τελικά διαγράμματα τα οποία παρουσιάζονται στο παρόν κεφάλαιο.

Δ'.3.1 Διαγράμματα

Παρακάτω φαίνονται τα διαγράμματα που προέκυψαν από τα δεδομένα των αποτελεσμάτων και αφορούν την κάθε ξεχωριστή πιθανότητα p .

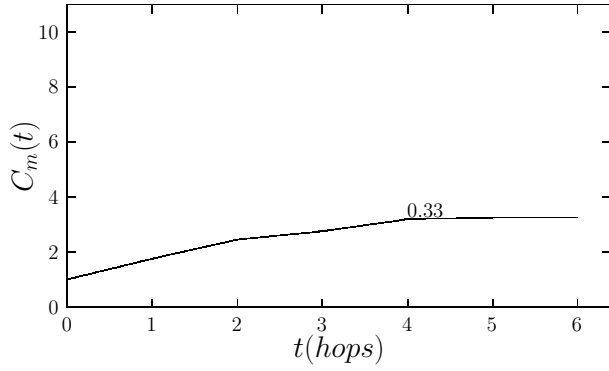
Πιθανότητα $p = 1$ 

Σχήμα Δ'.14: Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1$ (Σίγουρο ενδεχόμενο). Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.

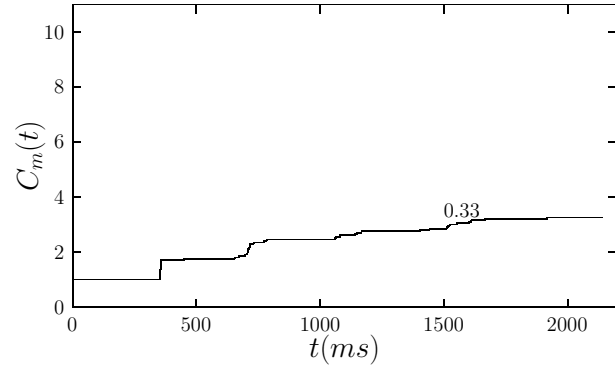
Πιθανότητα $p = 1/2$ 

Σχήμα Δ'.15: Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/2$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.

Πιθανότητα $p = 1/3$



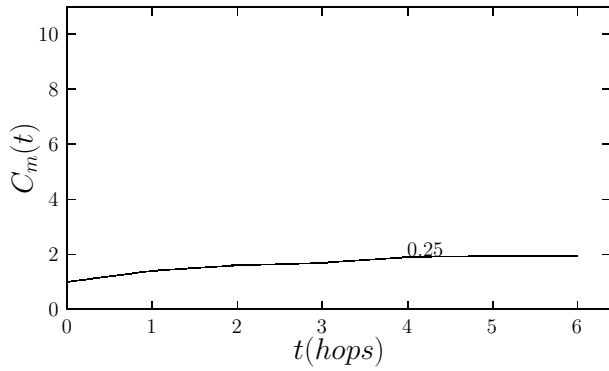
(α)



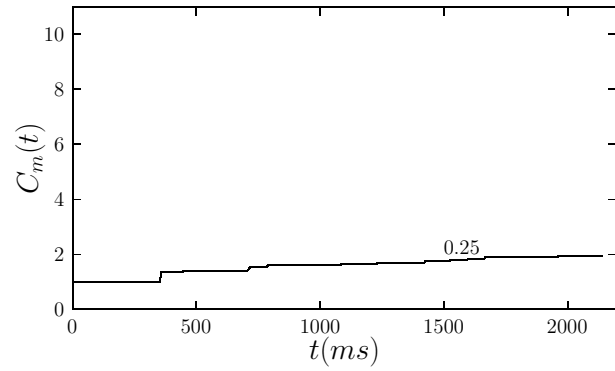
(β)

Σχήμα Δ'.16: Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/3$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.

Πιθανότητα $p = 1/4$



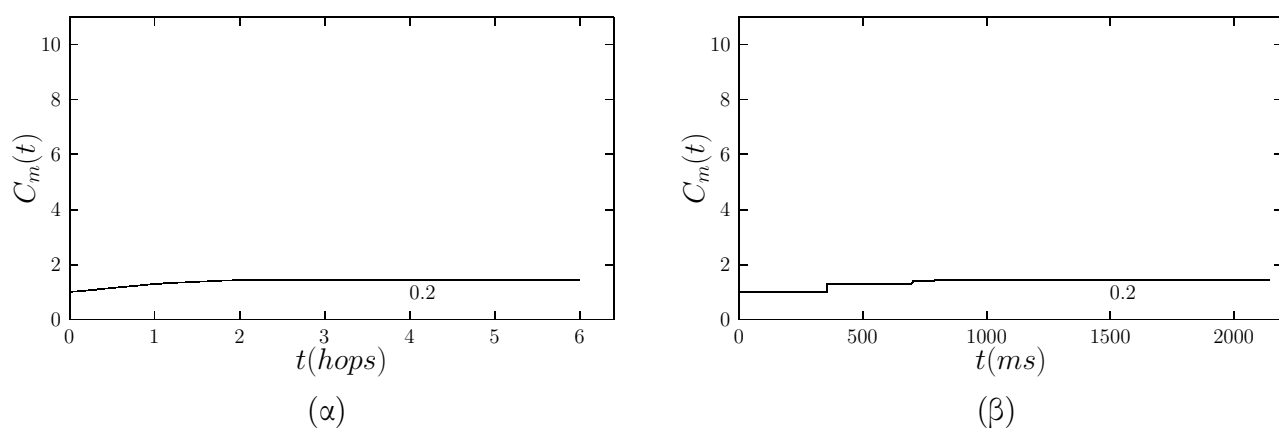
(α)



(β)

Σχήμα Δ'.17: Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/4$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.

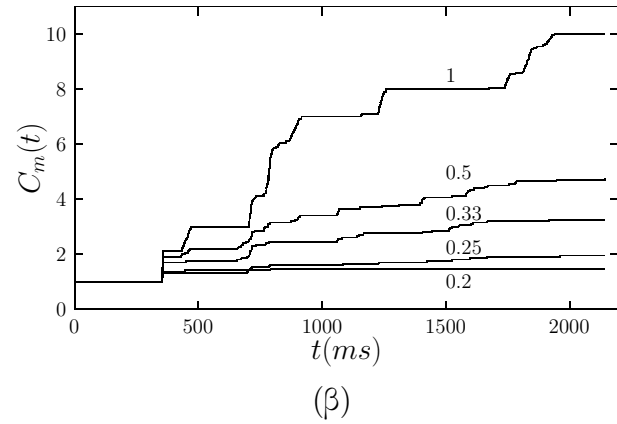
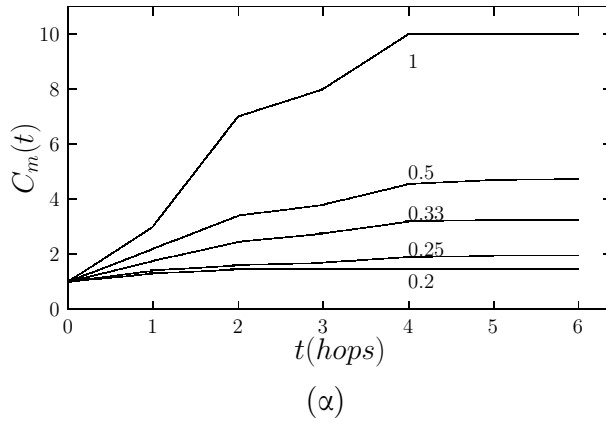
Πιθανότητα $p = 1/5$



Σχήμα Δ.18: Η διάδοση της Πλημμυρίδας για τιμή Πιθανότητας $p = 1/5$. Διάγραμμα (α): Κάλυψη του δικτύου C_m συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου C_m συναρτήσει των βημάτων (hops) του πειράματος.

Σύγκριση των Διαγραμμάτων Κάλυψης

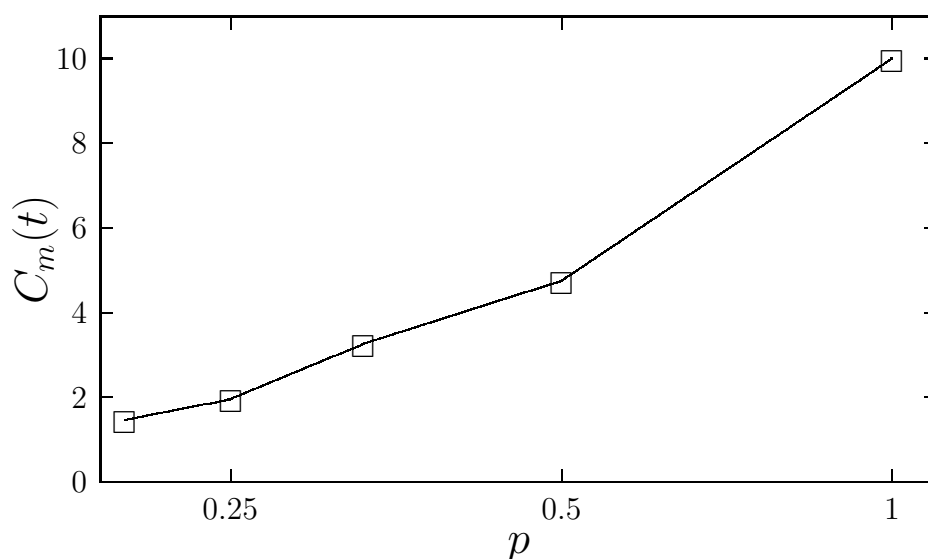
Αφού παρουσιάστηκαν τα διαγράμματα της Κάλυψης του Δικτύου στον χρόνο διεξαγωγής του πειράματος, για την κάθε ξεχωριστή πιθανότητα, έχει ενδιαφέρον να ακολουθήσει η ενσωμάτωσή τους σε ενιαία διαγράμματα με σκοπό την ποιοτική σύγκρισή τους.



Σχήμα Δ'.19: Η διάδοση της Πλημμυρίδας για όλες τις τιμές Πιθανοτήτων των πειραμάτων. Διάγραμμα (α): Κάλυψη του δικτύου συναρτήσει του χρόνου πειράματος (ms). Διάγραμμα (β): Κάλυψη του δικτύου συναρτήσει των βημάτων (hops) του πειράματος.

Ο ταχύτερος ρυθμός κάλυψης φαίνεται να είναι αυτός της περίπτωσης $p = 1$, καθώς η διάδοση γίνεται αξιοποιώντας την κατά περίπτωση βέλτιστη διαδρομή του δικτύου. Με την εισαγωγή του πιθανοτικού τρόπου προώθησης πληροφορίας ο ρυθμός της κάλυψης δικτύου πέφτει και ο αριθμός των καλυφθέντων κόμβων δεν είναι ίδιος με συνολικό αριθμό των κόμβων του δικτύου.

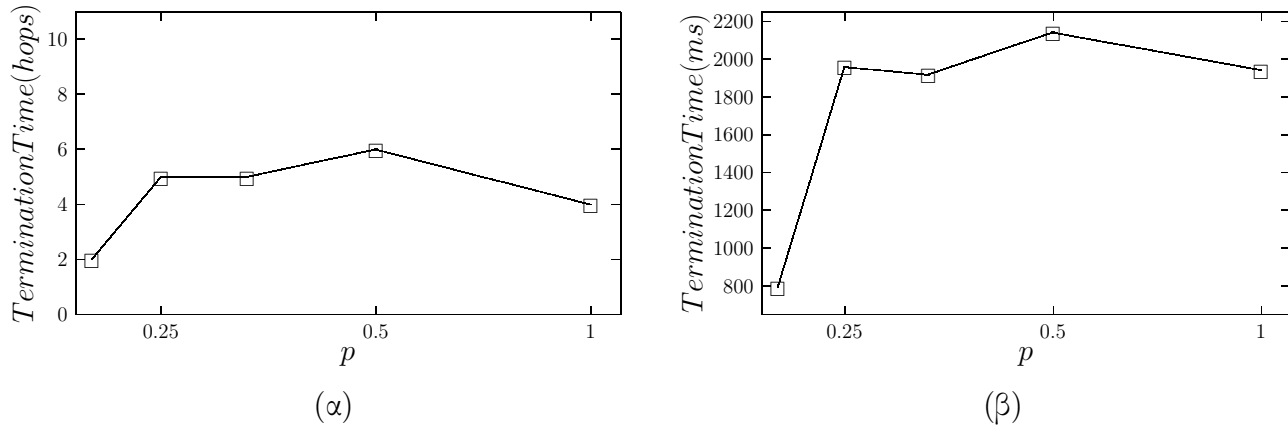
Η Κάλυψη του Δικτύου ανά Πιθανότητα



Σχήμα Δ'.20: Κάλυψη του δικτύου C_m συναρτήσει της πιθανότητας προώθησης p .

Η κάλυψη του δικτύου συναρτήσει της πιθανότητας προώθησης φαίνεται να παρουσιάζει εκθετική αύξηση. Οι μικρές πιθανότητες προώθησης έχουν ως αποτέλεσμα την κάλυψη λιγότερων κόμβων του δικτύου. Για παράδειγμα, για $p = 1/5$ το μήνυμα σπανίως διαφεύγει από τον κόμβο εκκινήτη και έτσι η τιμή C_m (τελικός αριθμός καλυφθέντων κόμβων τη τελευταία χρονική στιγμή του πειράματος) κυμαίνεται μεταξύ 1 και 2 κόμβων. Για $p = 1/2$ καλύπτεται, κατά μέσο όρο, ο μισός αριθμός των κόμβων και, σαφώς, για την περίπτωση της Πλημμυρίδας με πιθανότητα $p = 1$ όλοι οι κόμβοι λαμβάνουν το μήνυμα.

Χρόνοι Τερματισμού



Σχήμα Δ'.21: Χρόνοι τερματισμού του πειράματος συναρτήσει της πιθανότητας προώθησης p .

Ενδιαφέρον έχει η ερμηνεία των γραφικών παραστάσεων του χρόνου τερματισμού, σε συνάρτηση με την πιθανότητα προώθησης. Φαίνεται πως γενικά, ο χρόνος τερματισμού αυξάνεται, καθώς αυξάνεται η πιθανότητα προώθησης. Αυτό αποδίδεται στο γεγονός πως για μικρές τιμές της πιθανότητας προώθησης, το μήνυμα δεν καταφέρνει να επιβιώσει για αρκετό χρόνο μέσα στο δίκτυο και έτσι ο αλγόριθμος τερματίζεται πριν το μήνυμα προλάβει να φτάσει όλους τους κόμβους. Για πιθανότητα προώθησης $p = 1/2$ ωστόσο, ο χρόνος τερματισμού είναι μεγαλύτερος από τον χρόνο τερματισμού για πιθανότητα προώθησης $p = 1$ (Κλασική Πλημμυρίδα).

Και αυτό το φαινόμενο μπορεί να ερμηνευτεί, αν αναλογιστεί κανείς την ανάλυση της Πλημμυρίδας που προηγήθηκε στο δεύτερο κεφάλαιο, όπου αναφέρθηκε πως με τον συγκεκριμένο πιθανοτικό αλγόριθμο μπορεί να αυξηθεί σημαντικά ο χρόνος τερματισμού σε σχέση με την Κλασική Πλημμυρίδα, σε περίπτωση που δε περιοριστεί ο χρόνος παραμονής (διάρκεια ζωής) του μηνύματος στο δίκτυο. Επομένως για $p = 1/2$ (μέγιστη αβεβαιότητα), το μήνυμα διανύει περισσότερα βήματα στο δίκτυο, ακολουθώντας τυχαίες διαδρομές μέχρι να τερματίσει το πείραμα, σε σχέση με την περίπτωση του $p = 1$, όπου από τον εκκινητή προς τον κάθε κόμβο το μήνυμα καταφθάνει ακολουθώντας τη βέλτιστη διαδρομή.

Κεφάλαιο Ε΄

Σύνοψη και Συμπεράσματα

Το αντικείμενο μελέτης και επεξεργασίας της παρούσας πτυχιακής εργασίας υπήρξε η υλοποίηση αλγορίθμων πιθανοτικής πλημμυρίδας σε ασύρματα δίκτυα μέσω αντίστοιχης εφαρμογής και σε πραγματικές συνθήκες δικτύου.

Για τον σκοπό αυτό περιγράφηκαν οι ιδιότητες των ασύρματων δικτύων και ειδικότερα των ασυρμάτων δικτύων προσωπικής περιοχής και των δικτύων MANET, και αναλύθηκαν οι δυνατότητές τους.

Στη συνέχεια, έγινε αναφορά στο ερευνητικό ενδιαφέρον που παρουσιάζεται γύρω από τη διάδοση πληροφορίας με τον αλγόριθμο της παραδοσιακής Πλημμυρίδας, αλλά και λοιπών παραλλαγών του, όπως είναι ο αλγόριθμος της Πιθανοτικής Πλημμυρίδας, και εξηγήθηκε λεπτομερώς η λειτουργία τους.

Αφιερώθηκε, ύστερα, ένα κεφάλαιο στην περιγραφή των πρωτοκόλλων που διέπουν τη λειτουργία των πομποδεκτών XBee Pro, τα οποία χρησιμοποιήθηκαν για την υλοποίηση του συστήματος της εφαρμογής. Η λειτουργία του φυσικού και του επιπέδου πρόσβασης των μονάδων ορίζονται από το πρότυπο 802.15.4, ενώ το επίπεδο δικτύου και εφαρμογής ορίζονται από το πρωτόκολλο ZigBee.

Όλη η διαδικασία της προετοιμασίας και της υλοποίησης του δικτύου της εργασίας παρουσιάστηκε στο επόμενο κεφάλαιο. Περιγράφηκε λεπτομερώς ο εξοπλισμός και το περιβάλλον υλοποίησης, έγινε εξοικείωση με τις δύο διαφορετικές λειτουργίες της μονάδας XBee (Transparent, API) και παρατάθηκαν οι παράμετροι του συστήματος. Έπειτα, αναλύθηκε η λειτουργικότητα του κώδικα της εφαρμογής που συντάχθηκε με την βοήθεια της βιβλιοθήκης arduino-xtbee.

Η διεξαγωγή των πειραμάτων έγινε για 5 τιμές πιθανότητας προώθησης σε ένα

δίκτυο μεγέθους 10 κόμβων και χρησιμοποιώντας έναν απλό συγχρονισμό των μετρήσεων των ρολογιών τους. Σύμφωνα με τα αποτελέσματα, ο ρυθμός κάλυψης του δικτύου μειώνεται ανάλογα με τη μείωση της πιθανότητας προώθησης, η συνολική κάλυψη του δικτύου συναρτήσει της τιμής της πιθανότητας προώθησης παρουσιάζει, όπως ήταν αναμενόμενο, εκθετική αύξηση και οι χρόνοι τερματισμού για μικρές τιμές της πιθανότητας προώθησης προκύπτουν αντίστοιχα μικροί, ενώ για μεγάλες μπορούν, αντιθέτως, να αυξήσουν την διάρκεια του πειράματος σε σχέση με τον κλασικό αλγόριθμο πλημμυρίδας.

Τέλος, η μελέτη της εργασίας εμφανίζει προοπτικές για περαιτέρω ανάπτυξη σε μελλοντικό ερευνητικό στάδιο. Κάποιες από τις βελτιώσεις που μπορούν να πραγματοποιηθούν είναι:

- η αύξηση του αριθμού των κόμβων δικτύου
- η αύξηση του αριθμού των τιμών της πιθανότητας προώθησης
- η μεταφορά του πειράματος σε περισσότερες τοπολογίες δικτύων
- η βελτίωση του αλγόριθμου συσχετισμού των ρολογιών των κόμβων, για επίτευξη αποτελεσμάτων μεγαλύτερης ακρίβειας
- η καταμέτρηση των συνολικών μεταδιδόμενων μηνυμάτων στο δίκτυο για τη χρήση τους ως μετρική απόδοσης του αλγορίθμου

Βιβλιογραφία

- [1] R. A. Serway and J. W. Jewett, *Physics for scientists and engineers with modern physics*. Cengage learning, 2018.
- [2] M. Alonso and E. J. Finn, *Fundamental university physics*, vol. 2. Addison-Wesley Reading, MA, 1967.
- [3] W. Stallings, *Data and computer communications*. Pearson Education India, 2007.
- [4] “Ieee 802.15 wpan task group 4.” <http://www.ieee802.org/15/pub/TG4.html>. Τελευταία Ανακτηση: 25-06-2019.
- [5] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.
- [6] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.
- [7] C. S. R. Murthy and B. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.
- [8] G. Vakulya and G. Simony, “Energy efficient percolation-driven flood routing for large-scale sensor networks,” in *2008 International Multiconference on Computer Science and Information Technology*, pp. 877–883, IEEE, 2008.
- [9] A. Segall, “Distributed network protocols,” *IEEE transactions on Information Theory*, vol. 29, no. 1, pp. 23–35, 1983.
- [10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *Proceedings of the 5th annual ACM/IEEE*

- international conference on Mobile computing and networking*, pp. 151–162, ACM, 1999.
- [11] B. Williams and T. Camp, “Comparison of broadcasting techniques for mobile ad hoc networks,” in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 194–205, ACM, 2002.
 - [12] Y. Sasson, D. Cavin, and A. Schiper, “Probabilistic broadcast for flooding in wireless mobile ad hoc networks,” in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 2, pp. 1124–1130 vol.2, March 2003.
 - [13] A. O. Stauffer and V. C. Barbosa, “Probabilistic heuristics for disseminating information in networks,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 425–435, 2007.
 - [14] K. Oikonomou and I. Stavrakakis, “Performance analysis of probabilistic flooding using random graphs,” in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–6, IEEE, 2007.
 - [15] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Upper Saddle River, NJ, USA: Prentice Hall Press, 5th ed., 2010.
 - [16] K. Oikonomou, G. Koufoudakis, and S. Aissa, “Probabilistic flooding coverage analysis in large scale wireless networks,” in *2012 19th International Conference on Telecommunications (ICT)*, pp. 1–6, April 2012.
 - [17] P. Kinney *et al.*, “Zigbee technology: Wireless control that simply works,” in *Communications design conference*, vol. 2, pp. 1–7, 2003.
 - [18] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, “Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards,” *Computer communications*, vol. 30, no. 7, pp. 1655–1695, 2007.
 - [19] <https://www.zigbee.org/>. Τελευταία Ανακτση: 29-03-2019.
 - [20] “Xbee/xbee-pro s2c zigbee documentation.” <https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf>. Τελευταία Ανακτση: 29-06-2019.

- [21] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, “Study on zigbee technology,” in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, pp. 297–301, IEEE, 2011.
- [22] A. Elahi and A. Gschwender, *ZigBee wireless sensor and control network*. Pearson Education, 2009.
- [23] “Arduino uno.” <https://store.arduino.cc/arduino-uno-rev3>. Τελευταία Ανακτηση: 20-04-2019.
- [24] “Xctu next generation configuration platform for xbee/rf solutions.” <https://www.digi.com/products/iot-platform/xctu>. Τελευταία Ανακτηση: 22-04-2019.
- [25] “Xbee pro s2c - rf module.” <https://www.digi.com/products/embedded-systems/rf-modules/2-4-ghz-modules/xbee-zigbee>. Τελευταία Ανακτηση: 22-04-2019.
- [26] “Arduino ide.” <https://www.arduino.cc/en/Main/ArduinoXbeeShield>. Τελευταία Ανακτηση: 25-04-2019.
- [27] “arduino-xbee library.” <https://github.com/andrewrapp/xbee-arduino>.

Παραρτήματα

Παράρτημα Ι

Κώδικας Εφαρμογής Πλημμυρίδας

```
1 //Xbee Flooding
2 //This is a Series 2 XBee program
3
4 #include <MyFrame.h>
5
6 #include <XBee.h>
7
8 // create the XBee object
9 XBee xbee = XBee();
10
11 #
12 define DATASIZE 5 // the highest number of neighbours a node can have
13 # define ECHOSIZE 50# define PROBABILITY 5 // that is , p = (1/
    PROBABILITY)
14 # define NUMNODES 10# define N 10 // number of different
    probabilistic messages Co will send
15
16 const uint32_t zigbeeAddressHigh = 0x0013A200; // same for all XBees
17 uint8_t neighboursCount = 0;
18 uint8_t lowsFilled = 0;
19 char myID = '0';
20 uint32_t neighboursLow[DATASIZE] = {};
21 char neighboursID[DATASIZE] = {};
22
23 bool isCoordinatorEnabled = false;
24
```

```

25  uint8_t repliesFormExplorer = 0;
26  bool floodReceived[N] = {
27  false
28  };
29  uint32_t parentAddressLow = 0x0;
30
31  uint8_t TTLCovered[N] = {
32  0
33  };
34  unsigned long timeCovered[N] = {
35  0
36  };
37  //sync
38  unsigned long myTime = 0;
39  unsigned long timeDt = 0;
40  unsigned long CoTime = 0;
41
42  unsigned long floodTempTime = 0;
43  uint8_t floodCounter = 0;
44  uint8_t myEchoMessage[ECHOSIZE] = {}; // maximun length of echo
    message +5 more cells to fit status
45  uint8_t myEchoMessageLength = 0;
46  uint8_t myFloodMessage[3] = {};
47  uint8_t myFloodMessageLength = 3;
48  uint8_t resultMessage[7] = {};
49  uint8_t resultMessageLength = 7;
50
51  uint8_t sData[4];
52  uint16_t sDataLength = 0;
53
54  MyFrame receiveFrame = MyFrame();
55  MyFrame sendFrame = MyFrame();
56
57  struct Measur {
58  uint8_t TTL;
59  unsigned long time;
60
61  };
62
63  struct Measur measurements[N][NUMNODES];
64  void setup() {

```

```

65
66 Serial.begin(9600);
67 xbee.setSerial(Serial);
68
69 Serial.print(F("\n\nRX_BUFFER: "));
70 Serial.println(SERIAL_RX_BUFFER_SIZE);
71 Serial.print(F("TX_BUFFER: "));
72 Serial.println(SERIAL_TX_BUFFER_SIZE);
73 //Checking if the node is the coordinator of the network (initializer
   )
74 sendATCommand("CE");
75
76 sendATCommand("AP2");
77
78 //sending one hop away (neighbours)
79 sendATCommand("BH1");
80
81 sendATCommand("ID19");
82
83 sendATCommand("AO0");
84
85 sendATCommand("JV1");
86
87 sendATCommand("PL0");
88
89 sendATCommand("WR");
90
91 sendATCommand("SL");
92 sendATCommand("NI");
93 while (searchforNeighbError() == true) {
94   removeNeighbors();
95   sendATCommand("NI");
96 }
97
98 randomSeed(analogRead(0));
99 srand(analogRead(0));
100 int randomTime = random(1000, 5000);
101
102 //We set a delay after the beginning of the serial connection so that
   the nodes can have enough time to set up the network
103 delay(randomTime);

```

```
104
105  if (isCoordinatorEnabled) {
106    delay(2000);
107
108    //broadcast sync
109    sendFrame.setType(SYNC);
110
111    myTime = micros();
112    Serial.print(F("Sending time: "));
113    Serial.println(myTime);
114    memcpy(sData, & myTime, 4);
115    sendFrame.setData(sData, 4);
116
117    XBeeAddress64 addr64 = XBeeAddress64(0x00000000, 0x0000ffff); //
118    //broadcast
119    ZBTxRequest zbTx = ZBTxRequest(addr64, sendFrame.serialize(),
120    sendFrame.getLength());
121
122    xbee.send(zbTx);
123
124    sendFrame.defaultInit();
125
126    //initialize measurements results to 0
127    for (int i = 0; i < N; i++) {
128      for (int k = 0; k < NUMNODES; k++) {
129        measurements[i][k].TTL = 0;
130        measurements[i][k].time = 0;
131      }
132    }
133    //initiate Flooding
134
135    floodTempTime = micros();
136
137    myFloodMessage[0] = 0; //message id field
138    myFloodMessage[1] = PROBABILITY; // the probability
139    myFloodMessage[2] = 0; //TTL, starting with 0
140
141    sendFrame.setType(FLOOD);
142    sendFrame.setData(myFloodMessage, myFloodMessageLength);
143
144    Serial.print(F("\nFlooding attempt No "));
```

```

143 Serial.print(0);
144 unicastNeighboursProb(false , myFloodMessage[1]); //sending the above
    sendFrame with unicast to all neighbours
145 floodReceived[myFloodMessage[0]] = true; //will not resend
146
147 floodCounter = 1;
148 }
149
150 }
151
152 void loop() {
153
154     checkForPacketDelivery(); // Updates receiveFrame
155     FrameType type = receiveFrame.getType();
156
157     if (type != 0) {
158         Serial.print(F("\nReceived message type: "));
159         Serial.println(type);
160     }
161
162     if (type == SYNC) {
163         Serial.print(F("\nSync message received! "));
164         myTime = micros();
165         memcpy( & CoTime, receiveFrame.getData(), 4);
166         timeDt = myTime - CoTime;
167         Serial.println(myTime);
168         Serial.println(CoTime);
169         Serial.println(timeDt);
170
171     }
172
173     if (type == FLOOD) {
174         unsigned long tmp = micros();
175
176         Serial.print(F("\nFlood message received! Flood ID: "));
177
178         for (uint8_t i = 0; i < receiveFrame.getDataLength(); i++) {
179             myFloodMessage[i] = receiveFrame.getData()[i]; //we save the message's
                content
180         }
181         uint8_t floodID = myFloodMessage[0];

```

```

182 Serial.print(floodID , DEC);
183
184 if (floodReceived[floodID] == false) {
185
186     floodReceived[floodID] = true;
187     resultMessage[0] = floodID;
188     //add one to TTL and save it
189     myFloodMessage[2]++;
190     TTLCovered[floodID] = myFloodMessage[2]; // save it locally
191     resultMessage[1] = myFloodMessage[2]; // also send to Co
192     //save the time the node was covered at
193     timeCovered[floodID] = tmp - timeDt;
194     Serial.print(F("Time covered: "));
195     Serial.println(timeCovered[floodID]);
196     memcpy(resultMessage + 2, (uint8_t * )(timeCovered + floodID), 4); //
197     //saves 4 bytes after result[2]
198     //send results to parent
199     resultMessage[6] = myID - '0';
200     Serial.println(resultMessage[6]);
201     sendFrame.setType(RESULT);
202     sendFrame.setData(resultMessage, resultMessageLength); // not using
203     //the last cell yet
204     XBeeAddress64 addr64 = XBeeAddress64(0, 0);
205     ZBTxRequest zbTx = ZBTxRequest(addr64, sendFrame.serialize(),
206         sendFrame.getLength());
207     Serial.println(F("\nSending Result to Co... "));
208     for (uint8_t i = 0; i < sendFrame.getLength(); i++) {
209         Serial.print(sendFrame.getData()[i]);
210         Serial.print(" ");
211     }
212     Serial.println();
213     xbee.send(zbTx);
214
215     sendFrame.defaultInit();
216     Serial.print(F("\nAttempting to forward... "));
217     sendFrame.setType(FLOOD);
218     sendFrame.setData(myFloodMessage, myFloodMessageLength);
219     unicastNeighboursProb(false, myFloodMessage[1]);
220 }

```

```

220 if (type == RESULT) {
221   Serial.println(F("Result from child received! Data: "));
222
223   for (uint8_t i = 0; i < 7; i++) {
224     resultMessage[i] = receiveFrame.getData()[i]; //we save the message's
           content
225   }
226   unsigned long timeReceived = 0;
227   memcpy( & timeReceived, receiveFrame.getData() + 2, 4);
228   Serial.println(timeReceived);
229
230   Serial.print(resultMessage[0]);
231   Serial.print(", ");
232   Serial.println(resultMessage[6]);
233   measurements[resultMessage[0]][resultMessage[6]].TTL = resultMessage
     [1];
234   measurements[resultMessage[0]][resultMessage[6]].time = timeReceived;
235
236 }
237
238 if (isCoordinatorEnabled) {
239   if (micros() - floodTempTime >= 5000000 && floodCounter < N) {
240     myFloodMessage[0] = floodCounter; //message id field
241     myFloodMessage[1] = PROBABILITY; // the probability
242     myFloodMessage[2] = 0; //TTL, starting with 0
243
244     sendFrame.setType(FLOOD);
245     sendFrame.setData(myFloodMessage, myFloodMessageLength);
246
247     Serial.print(F("\nFlooding attempt No "));
248     Serial.print(floodCounter);
249     unicastNeighboursProb(false, myFloodMessage[1]); //sending the above
           sendFrame with unicast to all neighbours
250     floodReceived[myFloodMessage[0]] = true; //will not resend
251     floodCounter++;
252     floodTempTime = micros();
253   } else if (micros() - floodTempTime >= 5000000 && floodCounter == N)
     {
254     floodCounter++; // to not reprint
255     for (int i = 0; i < N; i++) {
256       for (int k = 0; k < NUMNODES; k++) {

```

```
257 Serial.print(measurements[i][k].TTL);
258 Serial.print(" ");
259 Serial.print(measurements[i][k].time);
260 Serial.print("\t");
261 }
262 Serial.println();
263 }
264 }
265 }
266 }
267 }
268 }
269
270 void sendZBTxFrame(uint8_t BH, uint32_t addrHigh, uint32_t addrLow) {
271
272   XBeeAddress64 addr64 = XBeeAddress64(addrHigh, addrLow);
273
274   //Here we create a frame of type ZBTxRequest
275   ZBTxRequest zbTx = ZBTxRequest(addr64, sendFrame.serialize(),
276     sendFrame.getLength());
277
278   zbTx.setBroadcastRadius(BH);
279
280   xbee.send(zbTx);
281
282   sendFrame.defaultInit();
283   Serial.print("Sent to: ");
284   Serial.println(zbTx.getAddress64().getLsb(), HEX);
285 }
286
287 void checkForPacketDelivery() {
288   if (receiveFrame.getType() != OTHER) {
289     receiveFrame.defaultInit();
290   }
291
292   xbee.readPacket();
293
294   if (xbee.getResponse().isAvailable()) {
295
296     XBeeAddress64 remoteAddress = XBeeAddress64();
```

```

297 Serial.println("\nGot ");
298
299 if (xbee.getResponse().getApiId() == ZB_EXPLICIT_RX_RESPONSE) {
300 ZBExplicitRxResponse explicitResponse = ZBExplicitRxResponse();
301 xbee.getResponse().getZBExplicitRxResponse(explicitResponse);
302 Serial.print(F("From: "));
303 Serial.println((explicitResponse.getRemoteAddress64().getLsb(), HEX))
    ;
304
305 receiveFrame.deserialize(explicitResponse.getData(), explicitResponse
    .getDataLength());
306
307 if (parentAddressLow == 0x0 && isCoordinatorEnabled == false) {
308
309 parentAddressLow = explicitResponse.getRemoteAddress64().getLsb();
310
311 Serial.print("My parent is ");
312 Serial.println(parentAddressLow, HEX);
313 }
314 return;
315
316 } else if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) {
317 ZBTxStatusResponse txStatus = ZBTxStatusResponse();
318 xbee.getResponse().getZBTxStatusResponse(txStatus);
319
320 // get the delivery status, the fifth byte
321 if (txStatus.getDeliveryStatus() == SUCCESS) {
322
323 Serial.print("Success\n");
324 } else {
325 // the remote XBee did not receive our packet. is it powered on?
326 Serial.print("Unsuccess\n");
327
328 }
329 } else if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) {
330 ZBRxResponse rxResponse = ZBRxResponse();
331 xbee.getResponse().getZBRxResponse(rxResponse);
332
333 if (rxResponse.getOption() == ZB_PACKET_ACKNOWLEDGED) {
334 Serial.print("ACK ");
335 } else {

```

```
336 Serial.print("NO ACK ");
337 }
338 Serial.print("\nFROM ");
339 Serial.println(rxResponse.getRemoteAddress64().getLsb(), HEX);
340 uint8_t * incData = rxResponse.getData();
341
342 uint8_t incDataLength = rxResponse.getDataLength();
343
344 Serial.println("Got data: ");
345 for (uint8_t i = 0; i < incDataLength; i++) {
346 Serial.print(incData[i], HEX);
347 }
348 Serial.println();
349
350 receiveFrame.deserialize(incData, incDataLength);
351 if (parentAddressLow == 0x0 && isCoordinatorEnabled == false) {
352 parentAddressLow = rxResponse.getRemoteAddress64().getLsb();
353 Serial.print("Parent: ");
354 Serial.println(parentAddressLow, HEX);
355 Serial.print("Type: ");
356 Serial.println(receiveFrame.getType());
357 }
358 return;
359
360 }
361
362 } else if (xbee.getResponse().isError()) {
363 Serial.print("Error code: ");
364 Serial.println(xbee.getResponse().getErrorCode());
365 }
366 if (receiveFrame.getType() != OTHER) {
367 receiveFrame.setType(OTHER);
368 }
369 }
370
371 uint8_t dBmValue = 0;
372
373 void sendATCommand(String command) {
374
375 uint8_t commandSplit[2];
376 uint8_t commandValue[1] = {
```

```

377 0x0
378 };
379 bool needsValue = false;
380
381 commandSplit[0] = command[0];
382 commandSplit[1] = command[1];
383 Serial.print("\nCommand: ");
384 Serial.print((char) commandSplit[0]);
385 Serial.print((char) commandSplit[1]);
386 Serial.println();
387
388 if (command.length() > 2) {
389     needsValue = true;
390     String tmp = command.substring(2, command.length());
391     commandValue[0] = (uint8_t) tmp.toInt();
392     command = command.substring(0, 2);
393     Serial.print("Command Value: 0x");
394     Serial.print(commandValue[0], HEX);
395     Serial.println();
396 }
397
398 //Below we have the request for the AT Command
399 AtCommandRequest atRequest = AtCommandRequest();
400
401 //And here we have the response of the AT Command Request.
402 AtCommandResponse atResponse = AtCommandResponse();
403
404 //Set atRequest's parameters
405 atRequest.setCommand(commandSplit);
406
407 if (needsValue) {
408
409     atRequest.setCommandValue(commandValue);
410     atRequest.setCommandValueLength(sizeof(commandValue));
411
412 }
413
414 // send the command
415 xbee.send(atRequest);
416

```

```
417 //This while loop will run at most for 5 second from the last
    AT_COMMAND_RESPONSE we get. If the command is Broadcast Hops it
    will take at most 5 seconds for the while to stop.
418 while (1) {
419
420 // wait up to 5 seconds for the status response.
421 if (xbee.readPacket(6000)) {
422 // got a response!
423
424 // should be an AT command response
425 if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
426 Serial.print("got a response!!");
427 xbee.getResponse().getAtCommandResponse(atResponse);
428
429 if (atResponse.isOk()) {
430 Serial.print("Command [");
431 Serial.print((char) atResponse.getCommand()[0]);
432 Serial.print((char) atResponse.getCommand()[1]);
433 Serial.println("] was successful!");
434
435 if (atResponse.getValueLength() > 0) {
436
437 if (command == "CE") {
438
439 if (atResponse.getValue()[0] == 0x1) {
440
441 isCoordinatorEnabled = true;
442 Serial.println("Co");
443
444 } else {
445
446 Serial.println("Ro");
447
448 }
449 break;
450 } else if (command == "NI") {
451 myID = static_cast<char>(atResponse.getValue()[0]);
452 Serial.println("MY ID is: ");
453 Serial.print(myID);
454 if (myID == '0') {
455 neighboursID[0] = '1';
```

```

456 neighboursID[1] = '4';
457 neighboursCount = 2;
458 sendATCommand("ND49");
459 sendATCommand("ND52");
460 } else if (myID == '1') {
461 neighboursID[0] = '0';
462 neighboursID[1] = '2';
463 neighboursID[2] = '3';
464 neighboursCount = 3;
465 sendATCommand("ND48");
466 sendATCommand("ND50");
467 sendATCommand("ND51");
468 } else if (myID == '2') {
469 neighboursID[0] = '1';
470 neighboursCount = 1;
471 sendATCommand("ND49");
472
473 } else if (myID == '3') {
474 neighboursID[0] = '1';
475 neighboursID[1] = '9';
476 neighboursCount = 2;
477 sendATCommand("ND49");
478 sendATCommand("ND57");
479
480 } else if (myID == '4') {
481 neighboursID[0] = '0';
482 neighboursID[1] = '5';
483 neighboursID[2] = '6';
484 neighboursCount = 3;
485 sendATCommand("ND48");
486 sendATCommand("ND53");
487 sendATCommand("ND54");
488 } else if (myID == '5') {
489 neighboursID[0] = '4';
490 neighboursID[1] = '9';
491 neighboursCount = 2;
492 sendATCommand("ND52");
493 sendATCommand("ND57");
494 } else if (myID == '6') {
495 neighboursID[0] = '4';
496 neighboursID[1] = '9';

```

```
497 neighboursCount = 2;
498 sendATCommand("ND52");
499 sendATCommand("ND57");
500 } else if (myID == '7') {
501     neighboursID[0] = '8';
502     neighboursID[1] = '9';
503     neighboursCount = 2;
504     sendATCommand("ND56");
505     sendATCommand("ND57");
506 } else if (myID == '8') {
507     neighboursID[0] = '7';
508     neighboursID[1] = '9';
509     neighboursCount = 2;
510     sendATCommand("ND55");
511     sendATCommand("ND57");
512 } else if (myID == '9') {
513     neighboursID[0] = '3';
514     neighboursID[1] = '5';
515     neighboursID[2] = '6';
516     neighboursID[3] = '7';
517     neighboursID[4] = '8';
518     neighboursCount = 5;
519     sendATCommand("ND51");
520     sendATCommand("ND53");
521     sendATCommand("ND54");
522     sendATCommand("ND55");
523     sendATCommand("ND56");
524 }
525
526 } else if (command == "ND") {
527
528     uint32_t neighLow = 0;
529
530     //extracting the destination bytes
531     for (int i = 6; i < 10; i++) {
532
533         // reverse the bytes
534         neighLow += (uint32_t) atResponse.getValue()[i] << (8 * (9 - i));
535     }
536
537     neighboursLow[lowFilled] = neighLow;
```

```

538     lowsFilled += 1;
539     Serial.print(neighLow, HEX);
540     Serial.print("All neighbours at the moment");
541     for (int i = 0; i < lowsFilled; i++) {
542         Serial.println(neighboursLow[i], HEX);
543         Serial.println(",");
544     }
545
546     break;
547
548     } else if (command == "BH") {
549
550     break;
551
552     } else if (command == "DB") {
553
554     dBmValue = atResponse.getValue()[0];
555
556     break;
557     } else if (command == "SL") {
558
559     break;
560     }
561
562     } else if (atResponse.getValueLength() == 0) {
563     break;
564     }
565     } else {
566     Serial.print(F("AT Error code: "));
567     Serial.println(atResponse.getStatus(), HEX);
568     }
569     } else {
570     Serial.print(F("Expected AT, got "));
571     Serial.print(xbee.getResponse().getApiId(), HEX);
572     }
573     } else {
574     // at command failed
575     if (xbee.getResponse().isError()) {
576     Serial.print(F("Error code: "));
577     Serial.println(xbee.getResponse().getErrorCode());
578     } else {

```

```

579
580 Serial.println(F("No response from radio"));
581 Serial.println(F("MAC Addresses found:"));
582 myEchoMessage[0] = myID;
583 myEchoMessage[1] = 0; //covered 'y' or 'n'
584 myEchoMessage[2] = 0; //ttl
585 myEchoMessage[3] = ':';
586 myEchoMessageLength += 4;
587 for (int i = 0; i < neighboursCount; i++) {
588 myEchoMessage[myEchoMessageLength] = neighboursID[i];
589 myEchoMessage[myEchoMessageLength + 1] = ',';
590 myEchoMessageLength += 2;
591
592 Serial.println(neighboursID[i]);
593 Serial.print(" : ");
594 Serial.println(neighboursLow[i], HEX);
595 }
596 Serial.print("my echo message:");
597 myEchoMessageLength--;
598 for (int i = 0; i < myEchoMessageLength; i++) {
599 Serial.print((char) myEchoMessage[i]);
600 }
601
602 break;
603 }
604
605 }
606
607 }
608
609 }
610
611 void unicastNeighboursProb(bool sendToParent, uint8_t prob) {
612
613 for (uint8_t i = 0; i < neighboursCount; i++) {
614 bool send = false;
615
616 if (rand() % prob < 1) {
617 send = true;
618 }
619 if (sendToParent == false && neighboursLow[i] == parentAddressLow) {

```

```

620     continue;
621 }
622 if (!send) {
623     continue;
624 }
625 Serial.print(F("\nSuccess! Sending to neighbor: "));
626 Serial.print(neighboursID[i]);
627
628 XBeeAddress64 addr64 = XBeeAddress64(zigbeeAddressHigh, neighboursLow
    [i]);
629
630 for (uint8_t i = 0; i < sendFrame.getLength(); i++) {
631     Serial.print(sendFrame.getData()[i]);
632     Serial.print(" ");
633 }
634 Serial.println();
635 ZBTxRequest zbTx = ZBTxRequest(addr64, sendFrame.serialize(),
    sendFrame.getLength());
636 xbee.send(zbTx);
637
638 }
639
640 sendFrame.defaultInit();
641 }
642
643 bool checkIfNeighborExists(uint32_t remoteAddress) {
644
645     for (uint8_t i = 0; i < neighboursCount; i++) {
646         if (neighboursLow[i] == remoteAddress) {
647             return true;
648         }
649     }
650
651     return false;
652
653 }
654
655 void removeNeighbors() {
656
657     for (uint8_t i = 0; i < neighboursCount; i++) {
658

```

```
659 neighboursLow[i] = 0x0;
660
661 }
662 Serial.println("Removing Neighbours...");
663 lowsFilled = 0;
664
665 }
666
667 bool searchforNeighbError() {
668     for (int i = 0; i < lowsFilled; i++) {
669         if (neighboursLow[i] == 0)
670             return true;
671     }
672     return false;
673
674 }
675
```

Παράρτημα II

Κώδικας Βιβλιοθήκης MyFrame.h

```
1  #
2  ifndef MyFrame_h# define MyFrame_h
3
4  #include "Arduino.h"
5
6  #include <inttypes.h>
7
8
9  using namespace std;
10
11  enum FrameType: uint8_t {
12  OTHER = 0,
13  SYNC = 1,
14  FLOOD = 2,
15  RESULT = 3
16  };
17
18  class MyFrame {
19  public:
20  FrameType _frameType;
21  uint8_t _maxLength;
22  uint8_t _dataLength;
23  uint8_t _dataOffset;
24  uint8_t * _frame = NULL;
25  uint8_t _length;
26  uint8_t _id;
```

```
27
28 // Many different constructors
29 MyFrame() {
30     defaultInit();
31 }
32
33 MyFrame(uint8_t fr) {
34     defaultInit();
35     setType(fr);
36 }
37
38 MyFrame(FrameType fr) {
39     defaultInit();
40     setType(fr);
41 }
42
43 MyFrame(FrameType fr, uint8_t * data, uint8_t dataLength) {
44     defaultInit();
45     setType(fr);
46     setData(data, dataLength);
47 }
48
49 MyFrame(uint8_t fr, uint8_t * data, uint8_t dataLength) {
50     defaultInit();
51     setType(fr);
52     setData(data, dataLength);
53 }
54
55 ~MyFrame() {
56     if (_frame != NULL) {
57         delete [] _frame;
58     }
59     _length = 0;
60     _dataLength = 0;
61     _maxDataLength = 0;
62     _frameType = OTHER;
63 }
64
65 uint8_t getDataOffset() {
66     uint8_t offset = sizeof(_frameType) + sizeof(_id) + sizeof(
        _dataLength);
```

```

67     return offset;
68 }
69
70 void defaultInit() {
71     _frameType = OTHER;
72     _id = 0;
73     _dataOffset = getDataOffset();
74     _maxLength = 255 - _dataOffset;
75     _length = _dataOffset; //10 byte o typos, 20 byte datalength
76     setDataLength(0);
77     defaultFrame();
78 }
79
80 void defaultFrame() {
81     if (_frame != NULL) {
82         delete [] _frame;
83     }
84     _frame = new uint8_t[_length];
85     _frame[0] = _frameType;
86     _frame[1] = _id;
87     _frame[2] = _dataLength;
88 }
89
90 void setType(FrameType frameType) {
91     _frameType = frameType;
92 }
93
94 void setType(uint8_t frameType) {
95     FrameType fr;
96     switch (frameType) {
97     case 1:
98         fr = SYNC;
99         break;
100    case 2:
101        fr = FLOOD;
102        break;
103    case 3:
104        fr = RESULT;
105        break;
106    default:
107        fr = OTHER;

```

```
108     break;
109 }
110 _frameType = fr;
111 }
112
113 void setType(String frameType) {
114     FrameType fr;
115     if (frameType == "sync") {
116         fr = SYNC;
117     } else if (frameType == "flood") {
118         fr = FLOOD;
119     } else if (frameType == "result") {
120         fr = RESULT;
121     } else {
122         fr = OTHER;
123     }
124     _frameType = fr;
125 }
126
127 FrameType getType() {
128     return _frameType;
129 }
130
131 void setID(uint8_t id) {
132     _id = id;
133 }
134
135 uint8_t getID() {
136     return _id;
137 }
138
139 void setMaxDataLength(uint8_t len) {
140     _maxDataLength = len;
141 }
142
143 void setDataLength(uint8_t len) {
144     if (len > _maxDataLength) {
145         len = _maxDataLength;
146     }
147     _dataLength = len;
148     _length = _dataOffset + _dataLength;
```

```

149 }
150
151 uint8_t getDataLength() {
152     return _dataLength;
153 }
154
155 void setData(uint8_t * data) {
156     defaultFrame();
157     for (uint8_t i = _dataOffset; i < _length; i++) {
158         _frame[i] = data[i - _dataOffset]; // i=2⇒ _frame[2] = data[0],
159         _frame[3] = data[1], ...
160     }
161 }
162
163 void setData(uint8_t * data, uint8_t len) {
164     setDataLength(len);
165     setData(data);
166 }
167
168 uint8_t * getData() {
169     return _frame + _dataOffset;
170 }
171
172 uint8_t getLength() {
173     return _length;
174 }
175
176 uint8_t * serialize() {
177     _frame[0] = _frameType;
178     _frame[1] = _id;
179     _frame[2] = _dataLength;
180     return _frame;
181 }
182
183 void deserialize(uint8_t * frame, uint8_t frameLength) {
184     defaultInit();
185     setType(frame[0]);
186     setID(frame[1]);
187     setDataLength(frame[2]);
188     if (frameLength > 3) {
189         setData(frame + 3); // frame[3:]

```

```
189     }  
190     }  
191     };  
192  
193     #  
194     endif //MyFrame_h  
195
```