

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



– Πτυχιακή Εργασία –

Τλοποίηση Αλγορίθμων Δρομολόγησης
Ελαχιστοποίησης Κατανάλωσης Ενέργειας
σε Ασύρματα Δίκτυα Αισθητήρων

Δημήτριος Σταμάτης
Επιβλέπων: Κωνσταντίνος Οικονόμου

13 Σεπτεμβρίου 2021

Επιβλέπων

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Αυτή η προσπάθεια αφιερώνεται στην οικογένεια και τους φίλους μου.
Ευχαριστώ για όλα.

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας είναι να μελετηθεί το ζήτημα της δρομολόγησης σε ασύρματα δίκτυα αισθητήρων λαμβάνοντας υπόψιν θέματα κατανάλωσης ενέργειας. Η εργασία επικεντρώνεται σε έναν αλγόριθμο που χρησιμοποιεί μια μέθοδο ανίχνευσης της ευπάθειας των διαθέσιμων μονοπατιών για τη μετάδοση πακέτων προς τον κόμβο συλλογής δεδομένων του δικτύου, και προσαρμόζει τη δρομολόγησή τους, προκειμένου να πετύχει την μέγιστη δυνατή διάρκεια ζωής για το δίκτυο. Σε αυτήν την εργασία διερευνάται ο συγκεκριμένος αλγόριθμος σε ένα πραγματικό δίκτυο και τα αποτελέσματά του συγκρίνονται με αυτά των προσομοιώσεων που παρουσιάζονται στη βιβλιογραφία καθώς και με τα αποτελέσματα ενός αντίστοιχου αλγορίθμου εύρεσης ελάχιστης διαδρομής με στατική δρομολόγηση. Η μεθοδολογία που χρησιμοποιήθηκε προκειμένου να γίνει αυτός ο έλεγχος ήταν αρχικά να βρεθεί το κατάλληλο υλικολογισμικό το οποίο μπορεί να υποστηρίξει τη δημιουργία ενός τέτοιου δικτύου σε μικρή ή μεγάλη κλίμακα. Στη συνέχεια υλοποιήθηκαν κάποιοι μη προσαρμοστικοί αλγόριθμοι δρομολόγησης προκειμένου να υπάρχει σχετικό μέτρο σύγκρισης. Τέλος, υλοποιήθηκε ο αλγόριθμος του πηγαιού υλικού της εργασίας, και έγινε μελέτη η οποία εξετάζει τις διαφορές ανάμεσα στα αποτελέσματα των δύο υλοποιήσεων. Τα αποτελέσματα της εργασίας δείχνουν μια σημαντική αύξηση της διάρκειας ζωής του δικτύου όταν γίνεται χρήση του πρωτογενούς αλγορίθμου δρομολόγησης σε σύγκριση με χρήση στατικής δρομολόγησης συντομότερης διαδρομής. Για την υλοποίηση της εργασίας, δημιουργήθηκε ένα ασύρματο δίκτυο με τη χρήση μικροελεγκτών arduino σε συνδυασμό με μονάδες XBee Pro S2C και μπαταρίες λιθίου-πολυμερούς.

Πρόλογος και Ευχαριστίες

Η παρούσα πτυχιακή εργασία με τίτλο "Υλοποίηση Αλγόριθμων Δρομολόγησης Ελαχιστοποίησης Κατανάλωσης Ενέργειας σε Ασύρματα Δίκτυα Αισθητήρων" αποτελεί ένα δείγμα των γνώσεων που αποκτήθηκαν κατά τη διάρκεια των σπουδών μου στο τμήμα πληροφορικής του Ιονίου Πανεπιστημίου, και της εφαρμογής αυτών των γνώσεων σε πρακτικό επίπεδο. Σκοπός της εργασίας είναι να εξεταστεί ένας προσαρμοστικός αλγόριθμος δρομολόγησης ασύρματων δικτύων αισθητήρων από την βιβλιογραφία, σε ένα πραγματικό περιβάλλον χρήσης και η εξαγωγή δεδομένων με τα οποία θα αξιολογηθεί η απόδοσή του. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέπων καθηγητή σε αυτήν την εργασία, κύριο Κωνσταντίνο Οικονόμου, για τον ενεργό ρόλο του στην ανάπτυξη του ενδιαφέροντος και της έλξης μου με τον κλάδο της πληροφορικής, καθώς και την πολύτιμη καθοδήγησή του κατά τη διαδικασία ανάπτυξης της πτυχιακής μου εργασίας. Επίσης, θα ήθελα να ευχαριστήσω το διδακτικό προσωπικό του Ιονίου Πανεπιστημίου καθώς και τους συμφοιτητές μου για το φιλικό και συναδελφικό περιβάλλον που διατήρησαν. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, των οποίων η συνεχής και άνευ όρων υποστήριξη με βοήθησε κατά τη διάρκεια των σπουδών μου.

Περιεχόμενα

A´	ΕΙΣΑΓΩΓΗ	1
A´.1	ΠΡΟΒΛΗΜΑ ΕΝΕΡΓΕΙΑΚΗΣ ΤΡΥΠΑΣ - ENERGY HOLE PROBLEM	2
A´.2	ΣΤΟΧΟΙ ΑΛΓΟΡΙΘΜΩΝ ΔΡΟΜΟΛΟΓΗΣΗΣ ΣΤΑ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗ- ΤΗΡΩΝ	2
A´.3	Ο ΕΞΕΤΑΖΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ	2
A´.3.1	Απαιτήσεις Αλγορίθμου	3
A´.3.2	Εφαρμογή του Αλγορίθμου σε Πραγματικό Δίκτυο	3
A´.4	ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΥΛΙΚΟΥ ΜΕΡΟΥΣ	4
A´.4.1	Μικροελεγκτής Arduino Uno	4
A´.4.2	Μονάδα Ασύρματης Επικοινωνίας Digi Xbee S2C Zigbee	4
A´.4.3	Arduino Wireless SD Shield	5
A´.5	ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ	5
A´.5.1	Arduino Integrated Development Environment (IDE)	5
A´.5.2	X-CTU	5
A´.5.3	Python Programming Language	6
A´.6	ΣΥΝΕΙΣΦΟΡΑ	6
A´.7	ΔΟΜΗ ΕΡΓΑΣΙΑΣ	6
B´	ΥΛΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΟΥ ΔΙΚΤΥΟΥ	7
B´.1	ΜΙΚΡΟΕΛΕΓΚΤΗΣ ARDUINO UNO	8
B´.1.1	Πλατφόρμα και Τρόπος Λειτουργίας	8
B´.1.2	Χρήση Arduino για τη Δημιουργία Ασύρματου Δικτύου Αισθητήρων	9
B´.2	DIGI XBEE S2C RADIO FREQUENCY MODULE	10
B´.2.1	Τεχνικά Χαρακτηριστικά	10

B'.2.2	Παραμετροποίηση για την Υλοποίηση του Δικτύου	10
B'.2.3	Διεπαφή με Arduino	11
B'.3	ARDUINO WIRELESS SD SHIELD	11
B'.3.1	Κάρτα Μνήμης SD	11
B'.3.2	Συνδεσιμότητα	11
B'.4	ΤΡΟΦΟΔΟΣΙΑ ΤΟΥ ΔΙΚΤΥΟΥ	12
B'.4.1	Μπαταρίες Λιθίου - Πολυμερούς 3.7 Volts	13
B'.4.2	Συνδεσιμότητα και Μέτρηση Τάσης	13
B'.4.3	Εναλλακτικές Επιλογές Τροφοδοσίας	13
Γ'	ΑΛΓΟΡΙΘΜΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΟΥ ΔΙΚΤΥΟΥ	15
Γ'.1	ΑΝΑΛΥΣΗ ΒΑΣΙΚΗΣ ΑΛΓΟΡΙΘΜΙΚΗΣ ΡΟΗΣ	16
Γ'.1.1	Βασικά Ζητούμενα του Πειράματος	17
Γ'.1.2	Sink Κόμβος	17
Γ'.1.3	Κόμβοι Αισθητήρων	18
Γ'.2	ΠΕΡΙΟΡΙΣΜΟΙ ΥΔΙΚΟΛΟΓΙΣΜΙΚΟΥ	19
Γ'.2.1	Δυνατότητες Κόμβων	19
Γ'.2.2	Καταγραφή Δεδομένων	20
Γ'.3	ΑΡΧΙΚΗ ΚΑΤΑΣΤΑΣΗ ΔΙΚΤΥΟΥ	20
Γ'.3.1	Τοπολογία Δικτύου	21
Γ'.3.2	Αναγνώριση Κόμβου	21
Γ'.3.3	Εύρεση Γειτονικών Κόμβων	22
Γ'.3.4	Εύρεση Ισχύος Σήματος με Γειτονικούς Κόμβους	22
Γ'.3.5	Πίνακες Γειτνίασης	23
Γ'.3.6	Αλγόριθμος Bellman - Ford	23
Γ'.4	ΒΙΒΛΙΟΘΗΚΕΣ ΚΑΙ ΚΛΑΣΕΙΣ ARDUINO	24
Γ'.4.1	Βιβλιοθήκη xbee-arduino - Xbee.h	24
Γ'.4.2	Βιβλιοθήκη climits - limits.h	24
Γ'.4.3	Πρότυπη Βιβλιοθήκη C - stdint.h	25
Γ'.4.4	Κλάση String() - string.h	25
Γ'.5	PYTHON SCRIPT - SERIALTOCSV.PY	25
Γ'.5.1	Αναγκαιότητα Χρήσης	26
Γ'.5.2	Βιβλιοθήκες Python	26

Γ'.5.3	Ανάλυση Αλγορίθμου	27
Γ'.5.3.1	Κώδικας	27
Γ'.5.4	Εναλλακτικές Επιλογές για την Καταγραφή Δεδομένων	29
Γ'.6	Ο ΚΟΜΒΟΣ SINK	29
Γ'.6.1	Αλγοριθμική Ροή του Sink Κόμβου	29
Γ'.6.2	Ανάλυση του Κώδικα	30
Γ'.6.2.1	Μέθοδος <i>triggerNodes()</i>	30
Γ'.6.2.2	Μέθοδος <i>readIncomingPackets()</i>	31
Γ'.6.3	Θέση Κόμβου στο Δίκτυο	32
Γ'.7	ΚΟΜΒΟΙ ΑΙΣΘΗΤΗΡΩΝ	33
Γ'.7.1	Απαιτήσεις	33
Γ'.7.2	Αλγοριθμική Ροή των Κόμβων Αισθητήρων	33
Γ'.7.3	Μεθοδολογία Επίλυσης Κεντρικών Ζητούμενων	34
Γ'.7.3.1	Υπολογισμός της Ποιότητας Σήματος	34
Γ'.7.3.2	Υπολογισμός της Ελάχιστης Απόστασης από τον <i>sink</i>	35
Γ'.7.3.3	Υπολογισμός <i>Path Vulnerability</i>	35
Γ'.7.3.4	Έλεγχος Κατάστασης Μπαταρίας	36
Γ'.7.3.5	Εύρεση Χρονικής Στιγμής Πρώτου Θανάτου	36
Γ'.7.4	Ανάλυση του Κώδικα	36
Γ'.7.4.1	Μέθοδος <i>waitForTrigger()</i>	36
Γ'.7.4.2	Μέθοδος <i>pushTrigger()</i>	37
Γ'.7.4.3	Μέθοδος <i>findDB()</i>	39
Γ'.7.4.4	Μέθοδος <i>manageDB()</i>	40
Γ'.7.4.5	Μέθοδος <i>getBattVolts()</i>	43
Γ'.7.4.6	Μέθοδος <i>beginTransmitting()</i>	44
Γ'.7.5	Συνθήκη Τερματισμού	46
Γ'.8	ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ	47
Γ'.8.1	Δομή Δεδομένων	47
Γ'.8.2	Μεθοδολογία Σύγκρισης	49
Δ'	ΣΤΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ	51
Δ'.1	ΕΝΕΡΓΕΙΑΚΕΣ ΜΕΤΡΗΣΕΙΣ ΤΟΥ ΔΙΚΤΥΟΥ	51
Δ'.2	ΔΙΑΜΕΤΑΓΩΓΗ (THROUGHPUT) ΤΟΥ ΔΙΚΤΥΟΥ	53

Περιεχόμενα	vi
Δ΄.3 ΣΥΣΧΕΤΙΣΗ ΜΕ ΤΟ ΠΗΓΑΙΟ ΥΛΙΚΟ	54
Ε΄ ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ	55
Ε΄.1 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΑΡΑΤΗΡΗΣΕΙΣ	55
Ε΄.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ	56
Βιβλιογραφικές Αναφορές	56
Συντμήσεις	59
Ευρετήριο	60

Κατάλογος Σχημάτων

B'.1	Ο μικροελεγκτής μονής πλακέτας Arduino Uno R3.	8
B'.2	Το Arduino συνδεδεμένο με το Wireless SD Shield και τη μονάδα XBee. . . .	12
Γ'.3	Διάγραμμα ροής της λειτουργίας των κόμβων.	16
Γ'.4	Η τοπολογία του δικτύου με τις τιμές των RSSI των ακμών.	21
Δ'.5	Διάγραμμα Μπαταρίας/Χρόνου με Στατική Δρομολόγηση	52
Δ'.6	Διάγραμμα Μπαταρίας/Χρόνου με Προσαρμοστική Δρομολόγηση	52

Κατάλογος Πινάκων

<i>B'.1 Βασικά χαρακτηριστικά των Xbee modules.</i>	10
<i>Γ'.1 Δεδομένα στατικής δρομολόγησης από την ίδια χρονική περίοδο.</i>	48
<i>Γ'.2 Δεδομένα προσαρμοστικής δρομολόγησης από την ίδια χρονική περίοδο.</i>	48
<i>Δ'.1 Διάρκεια ζωής του δικτύου (Network Lifetime).</i>	53
<i>Δ'.2 Διαμεταγωγή του δικτύου (Network Throughput).</i>	54

Αποσπάσματα Κώδικα

Γ'.1 Κώδικας serialToCsv.py	27
Γ'.2 Μέθοδος triggerNodes	30
Γ'.3 Μέθοδος readIncomingPackets	31
Γ'.4 Μέθοδος waitForTrigger	36
Γ'.5 Μέθοδος pushTrigger	37
Γ'.6 Μέθοδος findDB	39
Γ'.7 Μέθοδος manageDB - Επικοινωνία	40
Γ'.8 Μέθοδος manageDB - Εύρεση Ελάχιστων Αποστάσεων	41
Γ'.9 Μέθοδος manageDB - Ενημέρωση Γειτονικών Κόμβων	42
Γ'.10 Μέθοδος getBattVolts	43
Γ'.11 Μέθοδος beginTransmitting - Εύρεση των Path Vulnerabilities	44
Γ'.12 Μέθοδος beginTransmitting - Εύρεση Βέλτιστου Μονοπατιού	45
Γ'.13 Μέθοδος beginTransmitting - Επιλογή Μονοπατιού	45

Κεφάλαιο Α΄

ΕΙΣΑΓΩΓΗ

ΤΑ ασύρματα δίκτυα αισθητήρων (Wireless Sensor Networks ή WSNs) μπορούν να οριστούν ως ασύρματα δίκτυα που διαμορφώνονται χωρίς υποδομή και παρακολουθούν φυσικές ή περιβαλλοντικές συνθήκες, όπως η θερμοκρασία, ο ήχος, οι κραδασμοί, η πίεση, η κίνηση ή οι ρύποι τις οποίες μετατρέπουν σε δεδομένα κατάλληλης μορφής και τα μεταφέρουν συλλογικά μέσω του δικτύου στον sink¹ κόμβο. Ο sink κόμβος λειτουργεί σαν μια διεπαφή μεταξύ των χρηστών και του δικτύου. Οι χρήστες μπορούν να ανακτήσουν τις απαιτούμενες πληροφορίες από το δίκτυο με την έγχυση ερωτημάτων και τη συλλογή αποτελεσμάτων από τον κόμβο συλλογής. Συνήθως, ένα ασύρματο δίκτυο αισθητήρων περιέχει εκατοντάδες χιλιάδες κόμβους αισθητήρων οι οποίοι μπορούν να επικοινωνούν μεταξύ τους χρησιμοποιώντας ραδιοκύματα [1]. Τα ασύρματα δίκτυα αισθητήρων θεωρούνται από τους πιο γρήγορα εξελισσόμενους τεχνολογικούς τομείς στην επιστήμη των δικτύων χάρη στα πολυάριθμα οφέλη που παρέχει η χρήση τους. Το φάσμα εφαρμογών τους αυξάνεται ραγδαία όσο η ανάγκη μας για πιο έξυπνα και αποδοτικά συστήματα παρακολούθησης και αξιολόγησης μεγαλώνει [2].

¹Ως sink κόμβος, ή κόμβος συλλογής δεδομένων, ορίζεται μια συσκευή παρόμοια με τους κανονικούς κόμβους αισθητήρων αλλά πιο ισχυρή, η οποία συλλέγει δεδομένα από τους υπόλοιπους κόμβους.

Α'.1 Πρόβλημα Ενεργειακής Τρύπας - Energy Hole Problem

Ένας σημαντικός περιορισμός σε ένα ασύρματο δίκτυο αισθητήρων είναι η διάχυση ενέργειας των κόμβων και η διάρκεια ζωής της μπαταρίας του κόμβου. Η κυκλοφορία των πακέτων σε ένα ασύρματο δίκτυο αισθητήρων ακολουθεί ένα μοτίβο many-to-one, όπου οι κόμβοι που βρίσκονται κοντά στον κόμβο συλλογής μεταφέρουν μεγαλύτερα φορτία δεδομένων. Επομένως, οι κόμβοι που βρίσκονται κοντά στον κόμβο συλλογής και από τους οποίους εξαρτώνται οι περισσότεροι κόμβοι-παιδιά, θα εξαντλήσουν γρηγορότερα την ενέργειά τους, οδηγώντας στο γνωστό φαινόμενο της ενεργειακής τρύπας.

Α'.2 Στόχοι Αλγορίθμων Δρομολόγησης στα Ασύρματα Δίκτυα Αισθητήρων

Τα πρωτόκολλα δρομολόγησης στα ασύρματα δίκτυα αισθητήρων ποικίλουν ανάλογα με την εφαρμογή. Ένας μη αποτελεσματικός αλγόριθμος δρομολόγησης μπορεί να οδηγήσει σε μειωμένη διάρκεια ζωής του δικτύου και αυξημένη κατανάλωση ενέργειας. Έτσι, έχουν αναπτυχθεί διάφορα πρωτόκολλα δρομολόγησης για την ελαχιστοποίηση της κατανάλωσης ενέργειας και τη μεγιστοποίηση της διάρκειας ζωής του δικτύου. Κύριες προκλήσεις -μεταξύ άλλων- στη δρομολόγηση των πακέτων σε ένα ασύρματο δίκτυο αισθητήρων αποτελούν η κατανάλωση ενέργειας, η επεκτασιμότητα του δικτύου, η συνδεσιμότητα, η κάλυψη καθώς και οι επεξεργαστικές και αποθηκευτικές απαιτήσεις που επιφέρει στους κόμβους [3][4].

Α'.3 Ο Εξεταζόμενος Αλγόριθμος

Σε τέτοια δίκτυα, προκύπτουν σημαντικές απαιτήσεις. Αρχικά, οι κόμβοι του δικτύου συνήθως τροφοδοτούνται με μπαταρίες [5]. Επιπλέον, υπάρχει περίπτωση οι κόμβοι ανά πάσα στιγμή να απενεργοποιηθούν λόγω εξωτερικών παραγόντων ή εξάντλησης της ενεργειακής τους πηγής. Χρησιμοποιώντας συμβατικές στατικές τεχνικές δρομολόγησης των πακέτων (shortest-costs και shortest-paths αλγόριθμους), προκύπτουν μεγάλα προβλήματα ενεργειακής τρύπας (energy-hole problems) [6], τα οποία μειώνουν τη συνολική διάρκεια ζωής του δικτύου. Ο αλγόριθμος που υλοποιείται και εξετάζεται σε αυτήν την εργασία, προτείνει μια

μέθοδο ανίχνευσης της ευπάθειας των πιθανών μονοπατιών (Path Vulnerability) τα οποία μπορεί να ακολουθήσει ένα πακέτο από κάποιον απομακρυσμένο κόμβο προς τον κεντρικό κόμβο sink, επιλέγοντας το κατάλληλο μονοπάτι, ώστε να πετύχει τη μέγιστη δυνατή διάρκεια ζωής του δικτύου, καθώς και να παρατείνει όσο το δυνατόν περισσότερο τον πρώτο θάνατο του δικτύου (First Death Time).

Α΄.3.1 Απαιτήσεις Αλγορίθμου

Προκειμένου να εξεταστεί αυτός ο αλγόριθμος χρειάζεται να δημιουργηθεί ένα ασύρματο δίκτυο το οποίο θα λειτουργεί με τις βασικές αρχές των ad hoc δικτύων και το οποίο θα έχει τους αντίστοιχους περιορισμούς. Πιο συγκεκριμένα, οι κόμβοι του δικτύου θα πρέπει να έχουν μηδενική γνώση της θέσης τους στο δίκτυο καθώς και του μεγέθους του δικτύου και να τροφοδοτούνται με μπαταρίες ρεαλιστικών προδιαγραφών και χωρητικότητας, προκειμένου να θεωρούνται οικονομικοί και λογικοί για χρήση. Η επεξεργαστική τους ισχύς καθώς και η χωρητικότητα της μνήμης τους θα πρέπει να είναι εξίσου περιορισμένες εφόσον εξετάζεται ο αλγόριθμος για χρήση σε ασύρματα δίκτυα αισθητήρων, τα οποία τυπικά περιέχουν μικρής επεξεργαστικής ισχύος κόμβους. Τέλος, είναι απαραίτητο να έχουν τη δυνατότητα ασύρματης μετάδοσης και λήψης δεδομένων, δηλαδή να είναι ασύρματοι πομποδέκτες [7]. Εκτός από τους κόμβους αισθητήρων, θα πρέπει να υλοποιηθεί και ο sink κόμβος. Αυτός ο κόμβος είναι υπεύθυνος για τη συλλογή των δεδομένων που μεταδίδουν οι υπόλοιποι κόμβοι καθώς και για τη δημιουργία αιτημάτων προς τους κόμβους του δικτύου. Ο sink κόμβος δεν έχει τους ίδιους περιορισμούς. Πιο συγκεκριμένα, δεν εξαρτάται από μπαταρίες για την τροφοδοσία του. Επιπλέον, η επεξεργαστική του ισχύς μπορεί να είναι ίδια ή μεγαλύτερη από τους υπόλοιπους κόμβους. Τέλος, ο χρήστης του δικτύου συνήθως έχει πρόσβαση στον sink κόμβο καθώς και τη δυνατότητα να τον διαχειρίζεται και να του στέλνει αιτήματα, τα οποία ο sink υπολογίζει και απαντάει ή προωθεί στο υπόλοιπο δίκτυο ώστε να εκτελεστούν.

Α΄.3.2 Εφαρμογή του Αλγορίθμου σε Πραγματικό Δίκτυο

Για την επίτευξη της δημιουργίας ενός ασύρματου δικτύου το οποίο να αντιπροσωπεύει ένα πραγματικό ασύρματο δίκτυο αισθητήρων, χρησιμοποιήθηκαν οι εξής τεχνολογίες: Για το ρόλο των κόμβων ως μονάδων επεξεργασίας πληροφοριών έγινε χρήση μικροελεγκτών ανοικτού κώδικα Arduino Uno. Για την ασύρματη επικοινωνία μεταξύ των κόμβων χρησιμοποιήθηκαν μονάδες ασύρματης επικοινωνίας Digi Xbee S2C, και για τη διεπαφή ανάμεσα στον

μικροελεγκτή και τις μονάδες ΉΞβее, τοποθετήθηκαν Arduino Wireless SD Shields σε κάθε κόμβο. Όσον αφορά την τροφοδοσία τους, χρησιμοποιήθηκαν επαναφορτιζόμενες μπαταρίες λιθίου-πολυμερούς (Li-Po batteries) με μέγιστη τάση 3.7 Volts. Για τον κόμβο συλλογής πληροφοριών (sink) έγινε χρήση των ίδιων τεχνολογιών με τους κόμβους αισθητήρων με την εξαίρεση της μπαταρίας. Ο sink ήταν συνδεδεμένος σε φορητό υπολογιστή συνδεδεμένο σε παροχή ρεύματος, προκειμένου να είναι εύκολη η καταγραφή των δεδομένων και για να μην περιορίζεται χρονικά από πιθανή εξάντληση μπαταρίας.

Α΄.4 Συνοπτική Παρουσίαση του Υλικού Μέρους

Α΄.4.1 Μικροελεγκτής Arduino Uno

Το Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (πανομοιότυπη με τη γλώσσα προγραμματισμού C++, συνοδευόμενη από βιβλιοθήκες συγκεκριμένα γραμμένες για το Arduino). Μια πλακέτα Arduino αποτελείται από έναν μικροελεγκτή Atmel AVR (ATmega328P για το Arduino Uno) και συμπληρωματικά εξαρτήματα για τη διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωσή σου σε άλλα κυκλώματα ². Υπάρχουν πολλές διαφορετικές εκδόσεις των Arduino, η κάθε μια εκ των οποίων με διαφορετικά τεχνικά χαρακτηριστικά, δυνατότητες και λειτουργίες. Η πιο γνωστή έκδοση του οικοσυστήματος των Arduino είναι η έκδοση Uno. Η επιλογή της χρήσης του Arduino Uno έγινε λόγω της μεγαλύτερης γκάμας βοηθητικών εξαρτημάτων και υποστηριζόμενων μονάδων για την επέκτασή του. Για το συγκεκριμένο πείραμα χρειάστηκε να συνδυαστεί το Arduino Uno με δυο επιπλέον εξαρτήματα για να δημιουργηθούν οι κόμβοι του δικτύου, διαδικασίες που αναλύονται στα επόμενα υποκεφάλαια.

Α΄.4.2 Μονάδα Ασύρματης Επικοινωνίας Digi Xbee S2C Zigbee

Οι μονάδες ασύρματης επικοινωνίας Xbee (Xbee RF Modules) καλύπτουν τις ανάγκες του φυσικού επιπέδου του δικτύου μας καθώς και κάποιες βασικές λειτουργίες του επιπέδου ζεύξης δεδομένων (data link layer). Τα Xbee modules διαθέτουν πολλές λειτουργίες για τη δημιουργία και διατήρηση δικτύων πλέγματος (mesh networks), υποστηρίζοντας επικοινωνία από σημείο σε σημείο (point-to-point connection) αλλά και επικοινωνία μεταξύ σημείων και

² Arduino Reference: <https://www.farnell.com/datasheets/1682209.pdf>

πολλαπλών σημείων (multipoint connection)³ [8]. Για το σκοπό και τις ανάγκες του πειράματος της εργασίας, απέκλεισα κάποιες από αυτές τις λειτουργίες των μονάδων όπως η αυτόματη ανίχνευση γειτόνων και η αυτόματη επικοινωνία με πολλαπλά άλματα (multihop transmission), όταν ο ζητούμενος κόμβος βρίσκεται εκτός εμβέλειας. Η λειτουργία αυτή υλοποιείται αργότερα με δικό μου αλγόριθμο.

Α΄.4.3 Arduino Wireless SD Shield

Το Arduino Wireless SD Shield επιτρέπει σε μια πλακέτα Arduino Uno να επικοινωνεί ασύρματα χρησιμοποιώντας μια ασύρματη μονάδα (wireless module). Βασίζεται στις μονάδες Xbee της Digi. Το module περιλαμβάνει υποδοχή κάρτας SD. Ένας ενσωματωμένος διακόπτης επιτρέπει στην ασύρματη μονάδα να επικοινωνεί με τον μετατροπέα USB-to-serial ή με τον μικροελεγκτή, δίνοντας έτσι τη δυνατότητα στον χρήστη να επικοινωνεί απευθείας με την ασύρματη μονάδα ⁴.

Α΄.5 Συνοπτική Παρουσίαση του Λογισμικού

Α΄.5.1 Arduino Integrated Development Environment (IDE)

Αποτελεί το προγραμματιστικό περιβάλλον της πλακέτας Arduino. Περιέχει έναν επεξεργαστή κειμένου για συγγραφή κώδικα, περιοχή μηνυμάτων, γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες και διάφορα μενού. Συνδέεται με Arduino και Genuino για να ανεβάσει προγράμματα και να επικοινωνήσει μαζί τους. Η κατάληξη των αρχείων κώδικα που ανεβάζουν στις πλακέτες Arduino είναι η .ino. Η γλώσσα προγραμματισμού που υποστηρίζεται ονομάζεται Wiring, γλώσσα C++ με μερικές επιπλέον βιβλιοθήκες ενσωματωμένες για την υποστήριξη των πλακετών Arduino ⁵.

Α΄.5.2 X-CTU

Το XCTU είναι μια εφαρμογή που έχει σχεδιαστεί για να επιτρέπει στους προγραμματιστές να αλληλεπιδρούν με μονάδες ασύρματης επικοινωνίας της Digi μέσω μιας απλής γραφικής

³ Xbee Datasheet: <https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf>

⁴ Arduino Wireless SD Shield: <https://store.arduino.cc/arduino-wireless-sd-shield>

⁵ Arduino IDE: <https://www.arduino.cc/en/guide/environment>

διεπαφής. Περιλαμβάνει εργαλεία για τη ρύθμιση, τη διαμόρφωση και τη δοκιμή μονάδων Xbee RF ⁶.

Α΄.5.3 Python Programming Language

Για την αποθήκευση και επεξεργασία των δεδομένων που αντλούσε ο sink κόμβος του δικτύου μας, υλοποιήθηκε ένα πρόγραμμα σε γλώσσα python, το οποίο επικοινωνούσε με τον sink κόμβο μέσω σειριακής θύρας και στη συνέχεια μετέτρεπε τα δεδομένα που δεχόταν σε κατάλληλη μορφή και τα αποθήκευε σε αρχείο .csv (comma-separated values). Ο λόγος για τον οποίο προτιμήθηκε η χρήση εξωτερικού προγράμματος για την αποθήκευση των δεδομένων ήταν για την ελάφρυνση των επεξεργαστικών απαιτήσεων του sink κόμβου. Επιλέχθηκε η γλώσσα προγραμματισμού python, λόγω της ιδιαίτερης ευκολίας της χρήσης της, της εξαιρετικής διαχείρισης δεδομένων που προσφέρει, καθώς και της προσωπικής μου εμπειρία στη συγκεκριμένη γλώσσα.

Α΄.6 Συνεισφορά

Η εργασία αυτή στοχεύει στην ενίσχυση της ήδη υπάρχουσας βιβλιογραφικής μελέτης του προσαρμοστικού αλγορίθμου, με δεδομένα από την εφαρμογή του σε ένα πραγματικό δίκτυο αισθητήρων. Επιπλέον, στοχεύει στην ανάδειξη των αδύναμων σημείων των αλγορίθμων που θα εξεταστούν και αφού ολοκληρωθεί η ανάλυση των παραγόμενων δεδομένων τους, στην πρόταση πιθανών λύσεών τους.

Α΄.7 Δομή Εργασίας

Στα επόμενα κεφάλαια, θα γίνει ανάλυση των υλικών προδιαγραφών του δικτύου που υλοποιήθηκε, παραθέτοντας τα κίνητρα και τις εναλλακτικές επιλογές των εξαρτημάτων που χρησιμοποιήθηκαν. Έπειτα, θα αναλυθεί η δομή, ροή και υλοποίηση των αλγοριθμικών προδιαγραφών του δικτύου και της ανάλυσης των παραγόμενων δεδομένων τους. Τέλος, τα αποτελέσματα της εργασίας θα αναλυθούν και θα εξαχθούν κάποια συμπεράσματα και παρατηρήσεις καθώς και σημεία τα οποία φέρουν ενδιαφέρον για μελλοντική έρευνα.

⁶ XCTU Information Page: <https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu>

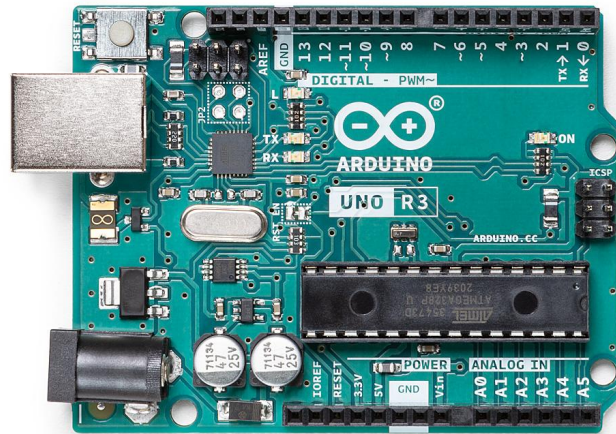
Κεφάλαιο Β΄

ΥΛΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΟΥ ΔΙΚΤΥΟΥ

ΟΙ κόμβοι ενός ασύρματου δικτύου αισθητήρων αποτελούνται συνήθως από ένα συνδυασμό μικροελεγκτών, εξαρτημάτων που προσφέρουν τη δυνατότητα ασύρματης επικοινωνίας καθώς και ένα σύνολο αισθητήρων.

Για τον σκοπό της εργασίας, χρησιμοποιήθηκαν μικροελεγκτές Arduino Uno R3 και Digi XBee S2C RF Modules. Προκειμένου τα δύο εξαρτήματα να έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους τοποθετήθηκε ένα Wireless SD Shield στην πλακέτα Arduino.

Β'.1 Μικροελεγκτής Arduino Uno



Σχήμα Β'.1: Ο μικροελεγκτής μονής πλακέτας Arduino Uno R3.

Β'.1.1 Πλατφόρμα και Τρόπος Λειτουργίας

Η πλατφόρμα Arduino έχει προσεγγίσει πολλούς ανθρώπους που ασχολούνται με ηλεκτρονικά κυκλώματα διότι σε αντίθεση με τις προηγούμενες προγραμματιζόμενες πλακέτες κυκλωμάτων, το Arduino δεν απαιτεί ξεχωριστό κομμάτι υλικού προκειμένου να προγραμματιστεί [9]. Το Arduino επιτρέπει στους χρήστες να δημιουργούν λειτουργικά ηλεκτρονικά πρότζεκτ τα οποία είτε λειτουργούν αυτόνομα είτε αλληλεπιδρούν με κάποιον υπολογιστή. Μπορεί να δεχτεί εισόδους από μια μεγάλη γκάμα αισθητήρων, να ελέγξει ένα ευρύ φάσμα συσκευών εξόδου καθώς και να επικοινωνήσει με λογισμικό το οποίο λειτουργεί σε κάποιον υπολογιστή ή να μεταφέρει πληροφορίες μέσω δικτύου [10]. Μία πλακέτα Arduino μπορεί να ταξινομηθεί σε δύο μέρη, το υλικό και το λογισμικό. Το υλικό της πλακέτας Arduino αποτελείται από πολλά συστατικά που συνδυάζονται για να λειτουργήσει, τα κύρια εκ των οποίων θα αναφέρω συνοπτικά.

- **Σύνδεση USB:** Αυτό είναι το πρώτο μέρος του Arduino επειδή χρησιμοποιείται για τη μεταφόρτωση ενός προγράμματος στον μικροελεγκτή και ταυτόχρονα τροφοδοτεί την πλακέτα με ρυθμιζόμενη ισχύ των 5 Volts.
- **Εξωτερική τροφοδοσία:** Χρησιμοποιείται μόνο για την τροφοδοσία της πλακέτας

και έχει ρυθμιζόμενη τάση 9 έως 12 Volts.

- **Κουμπί επανεκκίνησης:** Επαναφέρει το arduino, διατηρώντας τον κώδικα που έχει μεταφορτώσει ο χρήστης στη μνήμη και τον εκτελεί από την αρχή.
- **Μικροελεγκτής:** Αποτελεί τον "εγκέφαλο" της πλακέτας, δηλαδή η επεξεργαστική μονάδα της πλατφόρμας. Το μοντέλο του μικροελεγκτή εξαρτάται από την έκδοση του Arduino.
- **Αναλογικές ακίδες:** Δέχονται αναλογικό σήμα ως είσοδο και μπορούν να χρησιμοποιηθούν και για ψηφιακά σήματα εξόδου.
- **Ψηφιακές ακίδες εισόδου/εξόδου:** Δέχονται και στέλνουν ψηφιακό σήμα στις συνδεδεμένες συσκευές, μερικές θύρες υποστηρίζουν και Pulse-Width Modulation (PWM) σήμα.
- **Ακίδες τροφοδοσίας:** Για παροχή 3.3 και 5.5 Volts ισχύος.

Β'.1.2 Χρήση Arduino για τη Δημιουργία Ασύρματου Δικτύου Αισθητήρων

Η πλατφόρμα Arduino σε συνδυασμό με τις μονάδες ασύρματης επικοινωνίας Xbee προσφέρουν μια ιδανική επιλογή για την υλοποίηση του ζητούμενου δικτύου, λόγω της έμφυτης ευελιξιμότητας των δύο τεχνολογιών και του υψηλού επιπέδου παραμετροποίησης που διαθέτουν. Το Arduino μας δίνει τη δυνατότητα να γράψουμε κώδικα, ο οποίος θα υλοποιεί όλες τις αλγοριθμικές μας απαιτήσεις, χωρίς να χρειάζεται να βασιστούμε σε έτοιμα πρωτόκολλα επικοινωνίας, τα οποία πιθανόν να διακινδύνευαν την αρτιότητα των αποτελεσμάτων της ανάλυσης των δεδομένων μας [11]. Το Arduino δε διαθέτει από μόνο του δυνατότητες ασύρματης επικοινωνίας, αλλά το κενό αυτό το καλύπτει με ιδιαίτερη αποτελεσματικότητα η μονάδα ασύρματης επικοινωνίας Xbee της Digi. Η χρήση των Arduino σε εφαρμογές ασύρματων δικτύων αισθητήρων γίνεται όλο και πιο διάσημη λόγω του μειωμένου κόστους του συνολικού δικτύου και της πρακτικά απεριόριστης ευελιξίας χρήσης που προσφέρουν [12].

B'.2 Digi Xbee S2C Radio Frequency Module

Οι μονάδες Digi Xbee προσφέρουν ασύρματη συνδεσιμότητα για ηλεκτρονικές συσκευές σε χαμηλό κόστος. Χρησιμοποιούν πρωτόκολλα επικοινωνίας πλέγματος που υλοποιούνται πάνω στο IEEE 802.15.4 πρότυπο Zigbee [13]. Τα Xbee υποστηρίζουν ασύρματες επικοινωνίες δικτύου peer-to-peer καθώς και point to multi-point με ταχύτητα έως 250 kbits/s.

B'.2.1 Τεχνικά Χαρακτηριστικά

Η έκδοση της μονάδας που χρησιμοποιήθηκε για τη δημιουργία του δικτύου είναι το Xbee-PRO S2C Zigbee. Τα κύρια τεχνικά χαρακτηριστικά της μονάδας φαίνονται στο παρακάτω σχήμα.

Εμβέλεια σε εσωτερικό χώρο	Έως 90 μ.
Εξωτερική εμβέλεια με line-of-site ¹	Έως 3200 μ.
Ρυθμός Δεδομένων Ραδιοσυχνοτήτων	63 mW (+18 dBm)
Ευαισθησία Δέκτη	-101 dBm

Πίνακας B'.1: Βασικά χαρακτηριστικά των Xbee modules.

B'.2.2 Παραμετροποίηση για την Υλοποίηση του Δικτύου

Η αποστολή και παραλαβή των πακέτων μεταξύ ενός απομακρυσμένου κόμβου και του sink θα πρέπει να είναι multi-hop. Η μονάδα XBee S2C αναλαμβάνει αυτή τη διαδικασία αυτόματα και επιπλέον χρησιμοποιεί πρωτόκολλα επικοινωνίας, τα οποία επαναλαμβάνουν έως τρεις φορές την αποστολή ενός πακέτου σε περίπτωση μη παράδοσής του ή σύγκρουσης με κάποιο άλλο. Αυτές οι λειτουργίες αυξάνουν την κατανάλωση ενέργειας, διότι μεταδίδουν περισσότερα μηνύματα μεταξύ των κόμβων. Για το δικό μας δίκτυο απενεργοποιήθηκαν οι παραπάνω λειτουργίες προκειμένου να έχουμε καλύτερο έλεγχο της επικοινωνίας και βελτιστοποίηση της ποιότητας των αποτελεσμάτων του πειράματος στο στάδιο αρχικοποίησης των μονάδων XBee, το οποίο παρουσιάζεται αναλυτικά στο Κεφάλαιο Γ.4.1 (σελίδα 24).

¹line-of-site: Ευθεία κατά μήκος της οποίας ο πομπός και ο δέκτης έχουν ανεμπόδιση όραση μεταξύ τους

Β'.2.3 Διεπαφή με Arduino

Για την ενσωμάτωση της μονάδας XBee στην πλακέτα Arduino γίνεται χρήση του Arduino Wireless SD Shield σε επίπεδο υλικού, και της βιβλιοθήκης XBee.h σε επίπεδο λογισμικού.

Β'.3 Arduino Wireless SD Shield

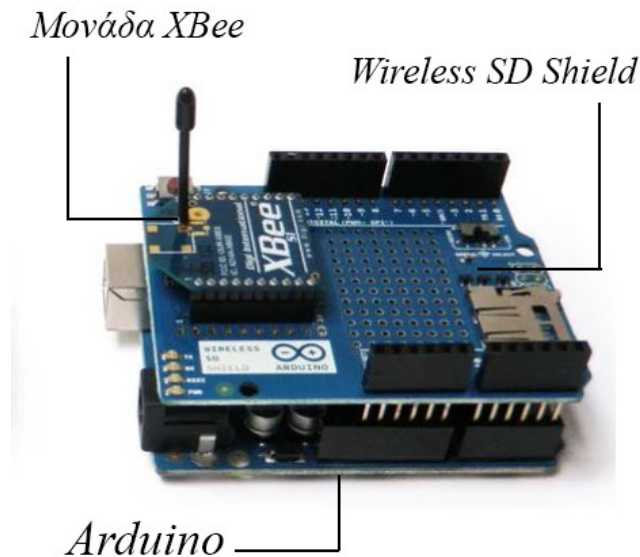
Το Arduino Wireless SD Shield επιτρέπει στην πλακέτα Arduino να επικοινωνεί ασύρματα χρησιμοποιώντας μια ασύρματη μονάδα, όπως η μονάδα Xbee. Βασίζεται στις μονάδες Xbee της Digi, αλλά μπορεί να χρησιμοποιήσει οποιαδήποτε μονάδα η οποία χρησιμοποιεί το ίδιο καλούπι. Ένας ενσωματωμένος διακόπτης επιτρέπει στην ασύρματη μονάδα να επικοινωνεί είτε μέσω της σειριακής θύρας του μικροελεγκτή, είτε απευθείας με τη συσκευή στην οποία ο μικροελεγκτής είναι συνδεδεμένος μέσω USB, παρακάμπτοντας έτσι τον μικροελεγκτή.

Β'.3.1 Κάρτα Μνήμης SD

Μία επιπλέον δυνατότητα που προσφέρει το Arduino Wireless SD Shield είναι η εισαγωγή κάρτας μνήμης τύπου micro SD την οποία μπορεί να διαχειριστεί το Arduino με τη χρήση κατάλληλης βιβλιοθήκης. Αυτή η δυνατότητα είναι σημαντική, διότι μας προσφέρει μια φαινομενικά ιδανική λύση για τη καταγραφή δεδομένων των αισθητήρων ή ακόμη και της τοπογραφίας του δικτύου μας στον κάθε κόμβο ή μόνο στον sink κόμβο. Η καταγραφή των δεδομένων για αυτό το πείραμα έγινε με άλλον τρόπο για λόγους αποδοτικότητας οι οποίοι αναλύονται στο Κεφάλαιο Γ5, σελίδα 25.

Β'.3.2 Συνδεσιμότητα

Η σύνδεση του Wireless SD Shield με την πλακέτα Arduino και τη μονάδα ασύρματης επικοινωνίας γίνεται εύκολα, και το τελικό σύστημα που προκύπτει και αποτελεί τους ασύρματους κόμβους του δικτύου μας φαίνεται στο παρακάτω σχήμα (Β'.2).



Σχήμα Β'.2: Το Arduino συνδεδεμένο με το Wireless SD Shield και τη μονάδα XBee.

Β'.4 Τροφοδοσία του Δικτύου

Η τροφοδοσία των κόμβων σε ένα ασύρματο δίκτυο αισθητήρων αποτελεί μεγάλο θέμα συζήτησης τόσο στην επιστημονική όσο και στην DIY² κοινότητα. Οι προτεινόμενες λύσεις του θέματος της τροφοδοσίας ποικίλουν αλλά ο κύριος παράγοντας που θα πρέπει να λάβει κάποιος υπόψιν όταν δημιουργεί ένα τέτοιο δίκτυο είναι το ίδιο το δίκτυο που θέλει να υλοποιήσει και οι συνιστώμενες απαιτήσεις του. Εφόσον στο δικό μας δίκτυο θέλουμε να παρατηρήσουμε την κατανάλωση της ενέργειας, αποκλείουμε απευθείας τις ανανεώσιμες πηγές ενέργειας, όπως η ηλιακή ενέργεια. Επιπλέον θέλουμε μια πηγή τροφοδοσίας με ρεαλιστικές προδιαγραφές για χρήση στον πραγματικό κόσμο. Τέλος, επειδή θα γίνουν διάφορα πειράματα μέχρι να καταλήξουμε στο τελικό δίκτυο θέλουμε να μπορούμε να αρχικοποιούμε την τροφοδοσία του δικτύου μας στις ίδιες αρχικές τιμές. Επομένως, καταλήξαμε στη χρήση επαναφορτιζόμενων μπαταριών Λιθίου-Πολυμερούς με τάση 3,7 Volts.

²DIY (Do It Yourself): Η διαδικασία κατά την οποία δημιουργείς ή διορθώνεις κάτι μόνος σου αντί να χρησιμοποιήσεις επαγγελματική βοήθεια.

Β'.4.1 Μπαταρίες Λιθίου - Πολυμερούς 3.7 Volts

Οι μπαταρίες που χρησιμοποιήθηκαν ανήκουν στην κατηγορία Li-Po. Τα πιο σημαντικά προτερήματα που προσφέρει η τεχνολογία Li-Po είναι η πολύ καλή αναλογία ενέργειας-μάζας και ενέργειας-όγκου, καθώς και η δυνατότητα επαναφόρτισης τους. Οι επιλεγμένες μπαταρίες έχουν ονομαστική τάση τα 3.7V, τάση ορίου φόρτισης τα 4.2V και τάση διακοπής εκφόρτισης τα 2.75V³. Για την τροφοδοσία του Arduino μπορούμε να προσφέρουμε από 1.8V έως 5.5V⁴ επομένως τα τεχνικά χαρακτηριστικά της μπαταρίας ικανοποιούν τις απαιτήσεις του δικτύου. Για λόγους ακρίβειας και αποφυγής κινδύνου υπολειτουργίας από τον μικροελεγκτή θα θεωρήσουμε πως η μπαταρία έχει αδειάσει όταν φτάνει την τάση των 3.3V.

Β'.4.2 Συνδεσιμότητα και Μέτρηση Τάσης

Προκειμένου να τροφοδοτηθεί ο κόμβος από τη μπαταρία, χρειάστηκε να συνδέσουμε το θετικό άκρο της μπαταρίας στην είσοδο 5V της πλακέτας Arduino (στις εσοχές του Wireless SD Shield), και τη γείωση της μπαταρίας στην εσοχή GND του Arduino. Η διαδικασία για τη μέτρηση της τάσης υλοποιήθηκε με αλγοριθμικό τρόπο χρησιμοποιώντας του καταχωρητές ADMUX του Arduino και αναλύεται στο κεφάλαιο Γ7.5.

Β'.4.3 Εναλλακτικές Επιλογές Τροφοδοσίας

Μερικές εναλλακτικές επιλογές τροφοδοσίας του δικτύου οι οποίες εξετάστηκαν ήταν οι εξής.

- **Χρήση Power Bank:** Η χρήση ενός power bank για τον κάθε κόμβο ήταν η πρώτη μας σκέψη κατά τη διαδικασία εύρεσης λύσης για την τροφοδοσία των κόμβων. Ο λόγος που απορρίφθηκε ήταν το αυξημένο κόστος τους καθώς και η ανάγκη για επιπλέον εξάρτημα για τη μέτρηση της τάσης τους με χρήση κώδικα.
- **Ηλιακοί συλλέκτες:** Η χρήση τους προτείνεται γενικά σε ασύρματα δίκτυα αισθητήρων με χαμηλές ενεργειακές απαιτήσεις. Ωστόσο, για το συγκεκριμένο πείραμα χρειάζεται τροφοδοσία πεπερασμένης διάρκειας προκειμένου να ελεγχθεί η αποτελεσματικότητα του αλγορίθμου δρομολόγησης.

³Πηγή: *Cellevia Batteries Specification Approval Sheet*

⁴Πηγή: *ATmega328P Datasheet*

- **Μπαταρίες μεγαλύτερης χωρητικότητας:** Για ένα αυτόνομο δίκτυο αισθητήρων μεγάλης διάρκειας θα γινόταν χρήση τόσο μπαταριών όσο και ηλιακών συλλεκτών συνδυαστικά προκειμένου να αυξηθεί η διάρκεια ζωής του δικτύου όσο το δυνατόν περισσότερο. Στο συγκεκριμένο πείραμα όμως δε χρειαζόμαστε το δίκτυο να μείνει ζωντανό για μεγάλη διάρκεια, αλλά χρειαζόμαστε ακρίβεια στη μέτρηση της διαφοράς ανάμεσα στη χρήση ενός παραδοσιακού αλγορίθμου δρομολόγησης και του προτεινόμενου. Επομένως δε χρειάστηκε να ενισχύσουμε την αποθηκευτική ικανότητα των μπαταριών ή την αυτονομία της πηγής τροφοδοσίας.

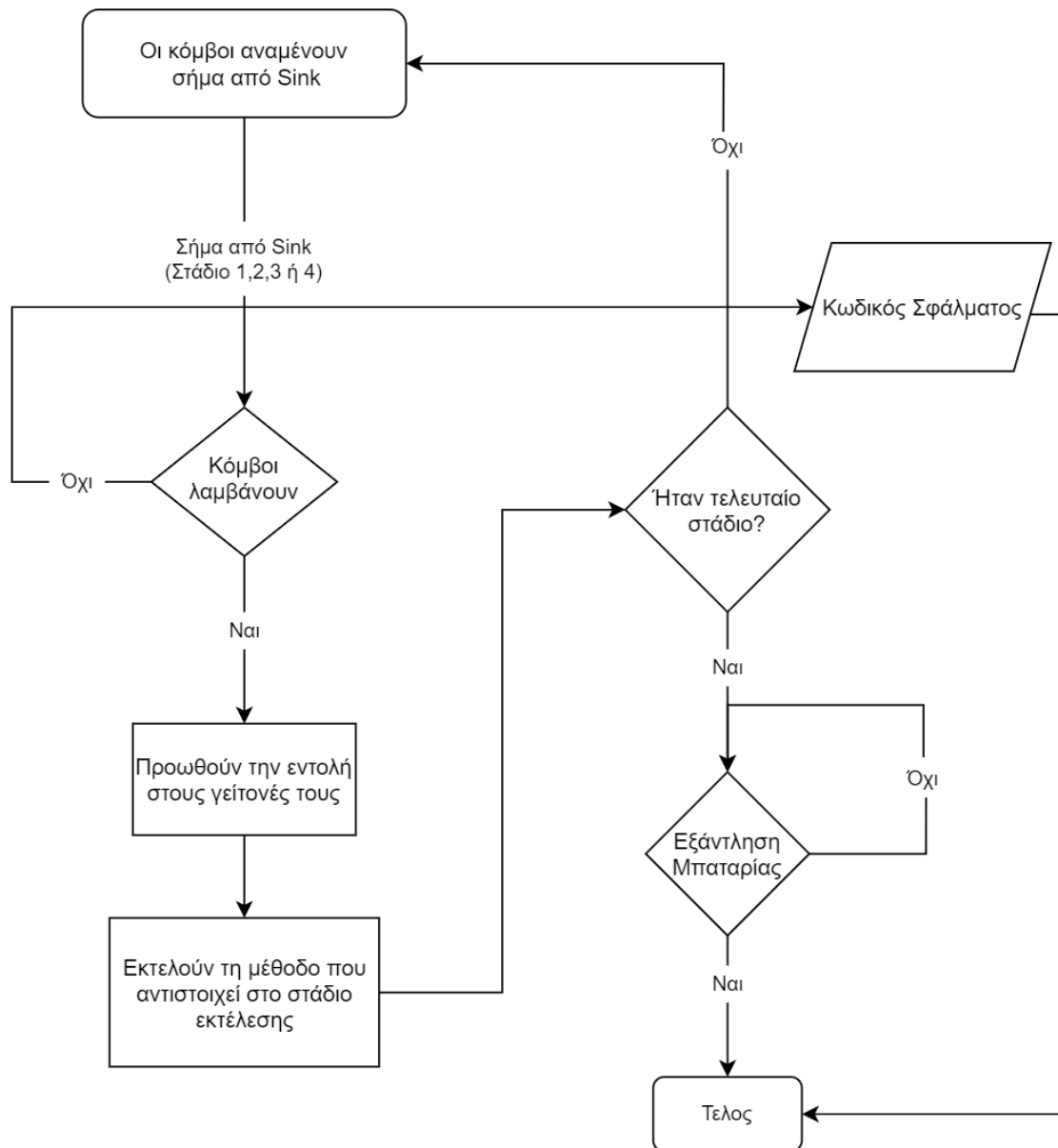
Κεφάλαιο Γ΄

ΑΛΓΟΡΙΘΜΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΟΥ ΔΙΚΤΥΟΥ

ΠΡΟΚΕΙΜΕΝΟΥ να λειτουργήσει το δίκτυό μας, και τα συνθετικά του μέρη να επικοινωνούν, υλοποιήθηκαν κάποιοι αλγόριθμοι και προγράμματα, τα οποία αποτελούν το αλγοριθμικό μέρος της εργασίας. Το αλγοριθμικό μέρος της εργασίας χωρίζεται σε τρία διακριτά μέρη, το μέρος της αλγοριθμικής σκέψης και οργάνωσης της εργασίας, το μέρος της προγραμματιστικής και αλγοριθμικής υλοποίησής του μέσω της ανάπτυξης λειτουργικών προγραμμάτων σε κατάλληλο προγραμματιστικό περιβάλλον και με παραμέτρους οι οποίοι συμφωνούν με τα ζητούμενα της εργασίας. Τέλος, το μέρος της ανάπτυξης και χρήσης λογισμικού το οποίο καθιστά δυνατή την ορθή επεξεργασία και ανάλυση των δεδομένων που προκύπτουν.

Ακολουθεί μια συνοπτική παρουσίαση των αλγοριθμικών μερών της εργασίας, ακολουθούμενη από την αναλυτική περιγραφή των κύριων τμημάτων της, καταλήγοντας στην ανάλυση της μεθοδολογίας που εφαρμόστηκε για την επεξεργασία των παραγόμενων δεδομένων.

Γ'.1 Ανάλυση Βασικής Αλγοριθμικής Ροής



Σχήμα Γ'.3: Διάγραμμα ροής της λειτουργίας των κόμβων.

Γ'.1.1 Βασικά Ζητούμενα του Πειράματος

Οι αλγοριθμικοί στόχοι του πειράματος είναι να δημιουργήσουμε ένα καταναμημένο δίκτυο ασύρματων αισθητήρων. Ο κάθε κόμβος τους δικτύου θα τρέχει τον ίδιο κώδικα και στο κέντρο του δικτύου θα υπάρχει ο sink κόμβος, ο οποίος θα τρέχει έναν διαφορετικό κώδικα, με σκοπό την οργάνωση του δικτύου, τη σηματοδότηση των διαφορετικών σταδίων του πειράματος στους κόμβους του δικτύου, καθώς και τη συλλογή των δεδομένων από τους υπόλοιπους κόμβους. Ο κόμβος που ανιχνεύει την περάτωση του πειράματος ή αλλιώς τον πρώτο θάνατο του δικτύου, είναι ο sink κόμβος. Πιο συγκεκριμένα, οι αλγόριθμοι που αναπτύχθηκαν δίνουν τη δυνατότητα στους κόμβους να αναγνωρίζουν τη δική τους ταυτότητα, τους γείτονές τους (μόνο αυτούς που βρίσκονται σε απόσταση 1-hop και επομένως μπορούν να επικοινωνήσουν απευθείας μαζί τους), καθώς και την απόστασή τους από τον sink κόμβο (στην περίπτωση του ίδιου του sink, η απόσταση είναι 0). Σε επόμενο στάδιο, κάθε κόμβος έχει γνώση των αποστάσεων των γειτόνων του από τον sink κόμβο, και ο κάθε κόμβος υπολογίζει τα path vulnerabilities των πιθανών μονοπατιών που μπορεί να χρησιμοποιήσει για την αποστολή των πακέτων του, τα οποία είναι αναγκαία προκειμένου να επιλεγεται αποδοτικά το κατάλληλο μονοπάτι. Τέλος, καταγράφονται δεδομένα που αφορούν τα μονοπάτια που χρησιμοποιήθηκαν από τον κάθε κόμβο για να προωθήσει τα πακέτα του στον sink, καθώς και τα ενεργειακά επίπεδα των μπαταριών του κάθε κόμβου στο δίκτυο καθόλη τη διάρκεια του πειράματος.

Γ'.1.2 Sink Κόμβος

Ο sink κόμβος, αφού λάβει το σήμα εκκίνησης από τη σειριακή διεπαφή (το σήμα εκκίνησης είναι το string "start") εκτελεί τα παρακάτω βήματα:

1. Βρίσκει τους γειτονικούς του κόμβους.
2. Στέλνει με τη χρήση της τεχνικής flooding στο υπόλοιπο δίκτυο σήμα, ώστε να ελέγξει ο κάθε κόμβος για τους δικούς του γείτονες.
3. Μετά από μία συγκεκριμένη χρονική στιγμή, σηματοδοτεί με τον ίδιο τρόπο τους κόμβους να ελέγξουν την ποιότητα του σήματος που έχουν με τους γείτονες τους.
4. Σηματοδοτεί τους κόμβους να ξεκινήσουν τη διαδικασία manageDB η οποία αναλύεται στο κεφάλαιο Γ (λογική: Γ7.3.2, σελ.35, κώδικας: Γ7.4.4, σελ.40). Μέσω αυτής

της διαδικασίας ο κάθε κόμβος βρίσκει ένα σύνολο από eligible parents¹ οι οποίοι διαθέτουν μικρότερη απόσταση από τον sink σε σύγκριση με τον ίδιο, κι έτσι μπορούν να χρησιμοποιηθούν για την προώθηση των πακέτων.

5. Στέλνει το τελευταίο έναυσμα στους κόμβους, το οποίο τους ενεργοποιεί και ξεκινάει τη μετάδοση δεδομένων προς τον sink μέχρι να αδειάσει η μπαταρία τους.
6. Ο sink πλέον "ακούει" για πακέτα και καταγράφει τα δεδομένα με μια διαδικασία που αναλύεται στο κεφάλαιο Γ6.2.2, σελίδα 31.
7. Όταν δε λάβει εισερχόμενα σήματα για πάνω από δύο λεπτά, τερματίζει τη διαδικασία.

Γ'.1.3 Κόμβοι Αισθητήρων

Οι κόμβοι αισθητήρων αφού ενεργοποιηθούν εκτελούν τα παρακάτω βήματα:

1. Αρχικοποιούν τις μεταβλητές τους.
2. Αναμένουν για ένα συγκεκριμένο πακέτο ενεργοποίησης το οποίο λαμβάνουν από τον sink κόμβο με την τεχνική flooding.
3. Όταν ένας κόμβος λάβει το πρώτο σήμα ενεργοποίησης, ανιχνεύει τους γείτονές του, και το προωθεί σε αυτούς. Με αυτόν τον τρόπο όλο το δίκτυο λαμβάνει το σήμα. Για την αποφυγή ατέρμονα βρόχου, όταν ο κόμβος εκτελέσει αυτό το βήμα, αν λάβει εκ νέου το ίδιο σήμα από άλλον γείτονα, το αδειάζει από το buffer και το αγνοεί. Αφού ολοκληρώσει αυτό το βήμα, περιμένει για το επόμενο σήμα από τον sink.
4. Αφού λάβει το επόμενο σήμα από τον sink, το οποίο σηματοδοτεί το επόμενο στάδιο του αλγορίθμου, προωθεί το σήμα στους γείτονές του και ξεκινάει τη διαδικασία findDB. Με αυτήν τη διαδικασία, βρίσκει την ισχύ σήματος που έχει με τον κάθε γειτονικό του κόμβο. Αυτή η πληροφορία είναι χρήσιμη για την εύρεση του βέλτιστου μονοπατιού σε επόμενο στάδιο. Όταν ολοκληρώσει τη διαδικασία αυτή, αναμένει για το επόμενο σήμα από τον sink.
5. Αφού λάβει το επόμενο σήμα από τον sink, το προωθεί στους γείτονές του και ξεκινάει τη διαδικασία manageDB. Αυτή η σύνθετη διαδικασία τρέχει για ένα πεπερασμένο χρονικό

¹ Eligible Parents: Οι γειτονικοί κόμβοι οι οποίοι βρίσκονται πιο κοντά στον sink κόμβο. Έτσι αποφεύγονται οι βρόγχοι.

διάστημα, ανανεώνοντας έναν πίνακα που περιέχει πληροφορίες με τις αποστάσεις που έχουν οι γειτονικοί κόμβοι από τον sink καθώς και ο ίδιος ο κόμβος από τον sink. Αφού ο κόμβος έχει λάβει τιμές από όλους τους γείτονές του και έχει ενημερώσει με τις δικές του τιμές όλους τους γείτονές του με επιτυχία, η διαδικασία ολοκληρώνεται, και ο κόμβος αναμένει για το επόμενο σήμα από τον sink.

6. Αφού ο κόμβος λάβει το επόμενο -και τελευταίο- σήμα από τον sink, το προωθεί στους γείτονές του και ξεκινάει τη διαδικασία `beginTransmitting`. Αυτή η διαδικασία είναι ο κορμός του αλγορίθμου και τρέχει μέχρι να τελειώσει η μπαταρία του κόμβου (πιο συγκεκριμένα όταν η τάση της μπαταρίας πέσει κάτω από 3.3Volts). Η διαδικασία αυτή αναλύεται σε επόμενα κεφάλαια.
7. Αφού εξαντληθεί η μπαταρία του κόμβου ή ο κόμβος δε μπορεί να βρει μονοπάτι για τον sink κόμβο, ο αλγόριθμος τερματίζει.

Γ'.2 Περιορισμοί Υλικολογισμικού

Το δίκτυο υλοποιείται με τη χρήση μικροελεγκτών Arduino σε συνδυασμό με τις ασύρματες μονάδες Xbee. Το σύστημα αυτό έχει κάποιους περιορισμούς όσον αφορά την επεξεργαστική ισχύ των κόμβων, την αποθηκευτική ικανότητά τους αλλά και την εμβέλεια της ασύρματης επικοινωνίας τους.

Γ'.2.1 Δυνατότητες Κόμβων

Οι κόμβοι του δικτύου μπορούν να υποστηρίξουν τον ζητούμενο αλγόριθμο, αρκεί να υπάρχουν κάποιοι χρονικοί περιορισμοί ανάμεσα στις μεταδόσεις των πακέτων μεταξύ τους και την προσεκτική χρήση του αποθηκευτικού τους χώρου. Για τον λόγο αυτόν, στο στάδιο της αποστολής πακέτων προς τον sink κόμβο, γίνονται διάφοροι έλεγχοι που αφορούν τον χρόνο που έχει περάσει από την τελευταία μετάδοση αλλά και παραλαβή πακέτων. Είναι πολύ σημαντικό οι μικροελεγκτές να προλαβαίνουν να διαβάσουν τα πακέτα που βρίσκονται στο buffer της σειριακής διεπαφής, διότι, όποτε ένα καινούργιο πλαίσιο καταφθάνει στην ασύρματη μονάδα, καταστρέφει το προηγούμενο. Ο κάθε κύκλος ελέγχου της διαδικασίας `beginTransmitting()` πρέπει να διαρκεί ελάχιστα `millisecond` ώστε να μη χάνονται πλαίσια. Ο τρόπος με τον οποίο επιλύθηκε το παραπάνω ζήτημα ήταν να γίνουν δοκιμές οι οποίες βοήθησαν να βρεθούν τα

κατάλληλα χρονικά διαστήματα παύσης μετάδοσης του κόμβου που χρειάζονται ώστε η επεξεργαστική ισχύς του μικροελεγκτή να είναι αρκετή για την έγκαιρη απορρόφηση των πλαισίων στο buffer.

Γ'.2.2 Καταγραφή Δεδομένων

Τη μεγαλύτερη πίεση τη λαμβάνει ο sink κόμβος ο οποίος εκτός από την έγκαιρη και ταχεία απορρόφηση των πλαισίων του serial buffer του, είναι υπεύθυνος και για την αποθήκευση των δεδομένων σε εξωτερικό μέσο, διαδικασία η οποία κοστίζει πολύ χρονικά και επεξεργαστικά. Η μια λύση που δοκιμάστηκε ήταν να μειωθεί ο ρυθμός μετάδοσης όλων των κόμβων του δικτύου, ώστε ο sink να έχει τη δυνατότητα να απορροφά τα δεδομένα από τον buffer και να τα γράφει μετά σε κάρτα μνήμης micro SD. Η δεύτερη λύση, η οποία και χρησιμοποιήθηκε στο τελικό πείραμα, ήταν ο sink κόμβος να μην ασχολείται καθόλου με την αποθήκευση των δεδομένων. Την αποθήκευση των δεδομένων την ανέλαβε ο υπολογιστής ο οποίος τροφοδοτούσε τον sink κόμβο. Υλοποιήθηκε ένα απλό πρόγραμμα σε γλώσσα Python το οποίο "έτρεχε" στον υπολογιστή, "άκουγε" τη σειριακή θύρα και ήταν υπεύθυνο για την αποθήκευση των δεδομένων σε αρχείο τύπου .csv με κατάλληλη μορφοποίηση. Χάρη σε αυτό, ο sink κόμβος μπορούσε να απορροφήσει τα πλαίσια από το buffer με ιδιαίτερη ταχύτητα, οπότε δε χρειάστηκε να περιοριστούν περεταίρω οι ρυθμοί μετάδοσης των απομακρυσμένων κόμβων.

Γ'.3 Αρχική Κατάσταση Δικτύου

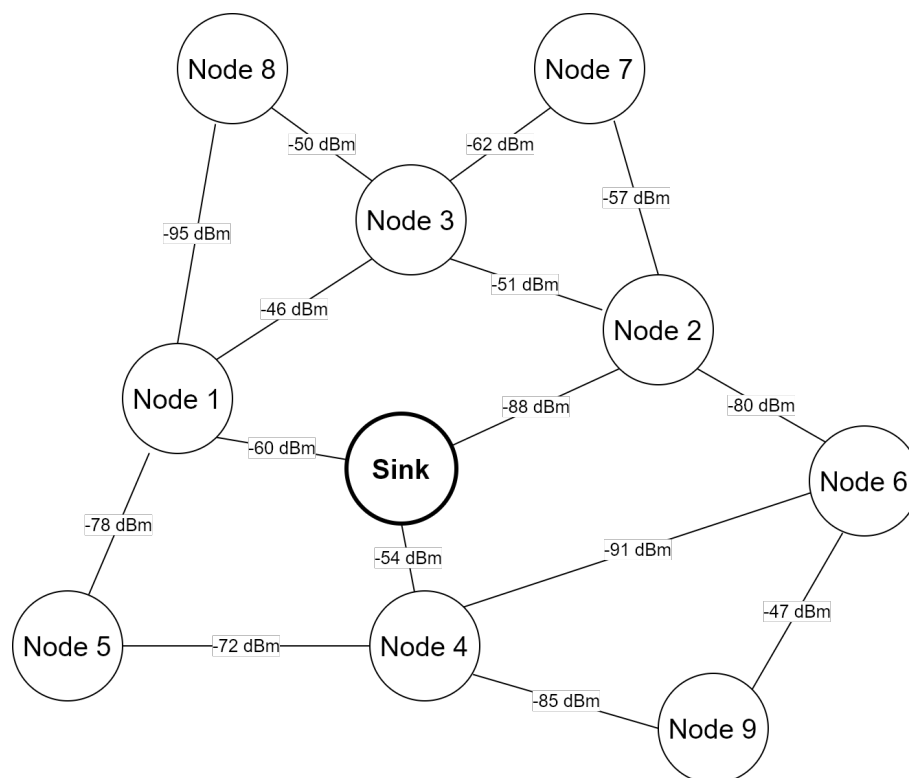
Από προεπιλογή, οι ασύρματες μονάδες XBee όταν στέλνουν ένα πακέτο προς μια άλλη μονάδα XBee, εκτελούν αυτόματα multi-hop επικοινωνία και φτάνει το πακέτο στον προορισμό του. Για τις ανάγκες του πειράματος, η λειτουργία αυτή έχει απενεργοποιηθεί από τις ρυθμίσεις της βιβλιοθήκης του Arduino. Έτσι, οι κόμβοι μας μπορούν να μεταδώσουν πακέτα μόνο σε γείτονες που απέχουν ένα άλμα.

Πριν ξεκινήσει η διαδικασία αποστολής και λήψης πακέτων στο δίκτυο, εκτελούνται κάποιες λειτουργίες στον κάθε κόμβο του δικτύου. Ο κάθε απομακρυσμένος κόμβος αρχικά αναγνωρίζει τον εαυτό του αντλώντας δεδομένα από την ασύρματη μονάδα του, όπως την διεύθυνσή του στο δίκτυο καθώς και το μοναδικό αναγνωριστικό του. Έπειτα, αναγνωρίζει τους γείτονές του, ελέγχει την ποιότητα του καναλιού επικοινωνίας του με αυτούς, δημιουργεί έναν πίνακα γειτνίασης και δρομολόγησης που περιέχει πληροφορίες για τους υπόλοιπους κόμβους

στο δίκτυο, και τέλος, υπολογίζει ποιοι γειτονικοί του κόμβοι είναι ικανοί να χρησιμοποιηθούν για την μετάδοση των πακέτων του προς τον sink, επιλέγοντας αρχικά αυτόν που προσφέρει το μονοπάτι με το μικρότερο ενεργειακό κόστος.

Γ'.3.1 Τοπολογία Δικτύου

Η τοπολογία του δικτύου έχει τον sink κόμβο στο κέντρο και φαίνεται στον παρακάτω γράφο (Γ'.4).



Σχήμα Γ'.4: Η τοπολογία του δικτύου με τις τιμές των RSSI των ακμών.

Γ'.3.2 Αναγνώριση Κόμβου

Η πρώτη ενέργεια που εκτελούν όλοι οι κόμβοι του δικτύου, όπως και ο sink, είναι η εύρεση των στοιχείων του ίδιου του κόμβου. Αυτό επιτυγχάνεται με κλήση εντολής προς την ασύρματη μονάδα επικοινωνίας του κόμβου, η οποία επιστρέφει τη μοναδική του διεύθυνση στο δίκτυο. Η διαδικασία αυτή υλοποιείται στη συνάρτηση `getMyAddress()` των κόμβων, και εκμεταλλεύεται

τα AT Commands της μονάδας XBee. Αφού σταλεί το αίτημα προς τη μονάδα XBee, ο μικροελεγκτής λαμβάνει πίσω ένα πακέτο που περιέχει τη διεύθυνση της ασύρματης μονάδας σε δεκαεξαδική μορφή, και η πληροφορία αυτή αποθηκεύεται τοπικά στη μνήμη του Arduino.

Γ'.3.3 Εύρεση Γειτονικών Κόμβων

Η εύρεση των γειτονικών κόμβων με απόσταση ενός άλματος επιτυγχάνεται με τη χρήση ενός ακόμα AT Command που προσφέρουν οι μονάδες XBee. Αυτό είναι το δεύτερο βήμα που εκτελούν όλοι οι κόμβοι του δικτύου και αποθηκεύουν τις διευθύνσεις των γειτόνων τους σε πίνακες. Το AT Command που χρησιμοποιείται είναι το FN (Find Neighbours). Η ασύρματη μονάδα στέλνει ένα unicast πακέτο το οποίο, όταν το λαμβάνουν οι ασύρματες μονάδες που βρίσκονται εντός της εμβέλειας, στέλνουν πίσω ένα πακέτο που περιέχει τη διεύθυνση του απομακρυσμένου κόμβου. Έστερα η ασύρματη μονάδα προωθεί τα πακέτα στο σειριακό buffer του Arduino, τα οποία διαβάζει με τη σειρά του ο αλγόριθμος και τα αποθηκεύει σε έναν πίνακα γειτνίασης.

Γ'.3.4 Εύρεση Ισχύος Σήματος με Γειτονικούς Κόμβους

Η εύρεση της ισχύος του σήματος ανάμεσα σε έναν κόμβο και έναν γείτονά του επιτυγχάνεται με την εξής διαδικασία. Έστω ο κόμβος A, ο οποίος θέλει να βρει την ισχύ του σήματος όταν επικοινωνεί με τον κόμβο B:

1. Ο κόμβος A στέλνει ένα Remote AT Command στον κόμβο B. Το Remote AT Command που χρησιμοποιείται είναι το "CH", το οποίο ζητάει από τον κόμβο B να του επιστρέψει το κανάλι στο οποίο λειτουργεί (Operating Channel).
2. Ο κόμβος B στέλνει πίσω στον κόμβο A τον αριθμό του Operating Channel στο οποίο λειτουργεί.
3. Αφού λάβει την απάντηση ο κόμβος A, εκτελεί την εντολή 'DB' τοπικά, η οποία επιστρέφει τον δείκτη ισχύος λαμβανόμενου σήματος (RSSI - Received Signal Strength Indication) του τελευταίου σήματος που έλαβε, δηλαδή της απάντησης του κόμβου B από το Remote AT Command που αιτήθηκε.

Προκειμένου να είναι ακριβής αυτή η τιμή, η διαδικασία επαναλαμβάνεται πολλαπλές φορές για κάθε ζεύγος γειτονικών κόμβων στο δίκτυο.

Γ'.3.5 Πίνακες Γειτνίασης

Σε κάθε κόμβο, κατά τη διαδικασία αρχικοποίησης του δικτύου, αναπτύσσεται ένας πίνακας γειτνίασης (adjacency table). Σε αυτόν τον πίνακα αποθηκεύονται οι διευθύνσεις των γειτονικών κόμβων του, και σε ξεχωριστούς παράλληλους πίνακες (parallel arrays) αποθηκεύονται δεδομένα για τους ίδιους κόμβους όπως η ισχύς σήματος, η κατάσταση της μπαταρίας τους, το path vulnerability που περιέχει το μονοπάτι προς τον sink, αν επιλεγεί ο κόμβος για προώθηση, καθώς και άλλοι προσωρινοί πίνακες που χρησιμοποιούνται περιστασιακά τοπικά από κάποιες μεθόδους. Ο πίνακας γειτνίασης του κάθε κόμβου προσαρμόζεται κατά τη διάρκεια λειτουργίας του δικτύου, προσθέτοντας και αφαιρώντας κόμβους οι οποίοι δεν είναι πλέον γείτονές του ή κόμβους οι οποίοι εισέρχονται εντός της εμβέλειάς του.

Γ'.3.6 Αλγόριθμος Bellman - Ford

Για την εύρεση των shortest paths προς τον sink κόμβο, υλοποιήθηκε μια προσαρμοσμένη έκδοση του γνωστού αλγορίθμου εύρεσης συντομότερης διαδρομής "Bellman - Ford" [14]. Ο αλγόριθμος Bellman - Ford εφαρμόζεται με χαλάρωση (relaxation), στην οποία οι εκτιμήσεις στη σωστή απόσταση αντικαθίστανται από καλύτερες, μέχρι να φτάσουμε τελικά στη λύση. Ο κάθε κόμβος αρχικοποιεί, στη θεωρία, την απόστασή του από τον sink στη τιμή του άπειρου. Στην υλοποίησή μας αρχικοποιείται στη μέγιστη τιμή που μπορεί να αποθηκεύσει ένας ακέραιος αριθμός χωρίς πρόσημο (unsigned integer), την οποία βρίσκουμε χρησιμοποιώντας τη βιβλιοθήκη limits.h. Στη συνέχεια, επικοινωνεί με κάθε γείτονά του, ενημερώνει για την απόστασή του από τον sink και ενημερώνεται για τις αποστάσεις των γειτόνων του από τον sink. Όταν ένας κόμβος, ο οποίος δεν έχει γείτονά του τον sink, ή η απόστασή του από τον sink είναι μεγαλύτερη από το άθροισμα της απόστασής του από κάποιον γείτονα και της απόστασης του γείτονά του από τον sink, τότε αυτός ο γείτονας επιλέγεται για προώθηση των σημάτων του προς τον sink, και ενημερώνει εκ νέου τους υπόλοιπους γείτονές του για την καινούργια απόστασή του από τον sink. Αυτή η διαδικασία προκαλεί το φαινόμενο της χαλάρωσης για την εύρεση της λύσης, και επαναλαμβάνεται έως ότου κανένας κόμβος δεν έχει καινούργια ελάχιστη απόσταση προς τον sink για να ενημερώσει τους γείτονές του, επομένως όταν δε ληφθεί κάποιο σήμα από γειτονικό κόμβο για αρκετό χρονικό διάστημα.

Γ'.4 Βιβλιοθήκες και Κλάσεις Arduino

Για την υλοποίηση των αλγορίθμων που αναπτύχθηκαν για τους κόμβους του δικτύου, χρησιμοποιήθηκαν τέσσερις βιβλιοθήκες. Ακολουθεί μια σύντομη ανάλυση της κάθε βιβλιοθήκης μαζί με τον σκοπό που εξυπηρέτησαν για τους συγκεκριμένους αλγορίθμους.

Γ'.4.1 Βιβλιοθήκη xbee-arduino - Xbee.h

Υλοποιημένη από τον Andrew Rapp, η βιβλιοθήκη xbee-arduino περιέχει μεθόδους που επιτρέπουν την επικοινωνία ανάμεσα στο Arduino και τη μονάδα ασύρματης επικοινωνίας XBee. Υποστηρίζει τους περισσότερους τύπους πακέτων που προσφέρουν οι μονάδες XBee όπως: TX/RX, AT Commands, Remote AT, I/O Samples και Modem Status. Για τους σκοπούς της εργασίας, έγινε χρήση ορισμένων μεθόδων που εξυπηρετούν στη μετάδοση σημάτων (TX), τη λήψη σημάτων (RX), την επικοινωνία με τη μονάδα XBee τοπικά (AT Commands), την επικοινωνία με μια απομακρυσμένη μονάδα XBee (Remote AT) καθώς και την αναγνώριση του τύπου εισερχόμενων σημάτων ή την ανίχνευση της κατάστασης αποστολής ενός σήματος (Modem Status). Επιπλέον, στον πηγαίο κώδικα του αρχείου κεφαλίδας (header file) της βιβλιοθήκης, έγιναν κάποιες αλλαγές οι οποίες αφορούσαν την αυτόματη μετάδοση σημάτων με χρήση multi-hop επικοινωνίας, η οποία απενεργοποιήθηκε, καθώς και την αυτόματη ανα μετάδοση σημάτων και αυτόματη αναγνώριση σημάτων, οι οποίες επίσης απενεργοποιήθηκαν.

Γ'.4.2 Βιβλιοθήκη climits - limits.h

Η βιβλιοθήκη climits καθορίζει διάφορες ιδιότητες των διαφόρων τύπων μεταβλητών της C. Οι μακροεντολές που ορίζονται σε αυτήν την κεφαλίδα, περιορίζουν τις τιμές διαφόρων τύπων μεταβλητών όπως char, int και long. Αυτά τα όρια καθορίζουν ότι μια μεταβλητή δε μπορεί να αποθηκεύσει καμία τιμή πέρα από αυτά τα όρια. Για παράδειγμα ένας χαρακτήρας χωρίς πρόσημα μπορεί να αποθηκεύσει μια τιμή με μέγιστο το 255. Η μόνη χρήση της βιβλιοθήκης βρίσκεται στην αρχικοποίηση της μεταβλητής που κρατάει την απόσταση των κόμβων από τον κόμβο sink, η οποία αρχικοποιείται στη μέγιστη δυνατή τιμή της μεταβλητής, προσομοιώνοντας έτσι την αρχικοποίηση στο άπειρο που ορίζει ο αλγόριθμος Bellman - Ford.

Γ'.4.3 Πρότυπη Βιβλιοθήκη C - stdint.h

Το `stdint.h` είναι ένα header file το οποίο ανήκει στην πρότυπη βιβλιοθήκη C (C Standard Library) το οποίο επιτρέπει την υλοποίηση πιο φορητού κώδικα, παρέχοντας ένα σύνολο typedefs που καθορίζουν ακέραιους τύπους ακριβούς πλάτους (exact-width integer types), μαζί με το καθορισμένο ελάχιστο και μέγιστο επιτρεπόμενο όριο τιμών του κάθε τύπου, χρησιμοποιώντας μακροεντολές.

Αυτή η κεφαλίδα είναι ιδιαίτερα χρήσιμη για ενσωματωμένο προγραμματισμό ο οποίος συχνά απαιτεί σημαντική χειραγώγηση των καταχωρητών I/O του υλικού, που απαιτούν ακέραια δεδομένα ακριβούς πλάτους και ακριβείς ευθυγραμμίσεις. Συγκεκριμένα για τα `arduino`, ορίζουν compiler-specific ορια των ακεραίων μεταβλητών τους. Για τη συγκεκριμένη εργασία, έγινε χρήση των τύπων μεταβλητών `uint8_t` για τα πακέτα και άλλες μεταβλητές των κόμβων και `uint32_t` για τις διευθύνσεις των κόμβων, με σκοπό την άψογη συνεργασία του `arduino` με την ασύρματη μονάδα XBee, η οποία χρησιμοποιεί τους ίδιους τύπους μεταβλητών.

Γ'.4.4 Κλάση String() - string.h

Η κλάση `String`, που ανήκει στις βασικές κλάσεις του `Arduino`, επιτρέπει τη χρήση και επεξεργασία συμβολοσειρών κειμένου (text strings) με πιο πολύπλοκους τρόπους από αυτούς που προσφέρουν οι πίνακες χαρακτήρων. Περιέχει έτοιμες μεθόδους που εκτελούν όλες τις βασικές και σύνθετες ενέργειες που αφορούν τις συμβολοσειρές κειμένου. Η χρήση της κλάσης περιορίστηκε στο κομμάτι του εντοπισμού σφαλμάτων (debugging process) της ανάπτυξης του αλγορίθμου για τη γρήγορη και εύκολη διαχείριση μηνυμάτων σφάλματος.

Γ'.5 Python Script - serialToCsv.py

Η καταγραφή των δεδομένων πραγματοποιήθηκε με την υλοποίηση και χρήση ενός Python προγράμματος το οποίο έτρεχε στον φορητό υπολογιστή στον οποίο ήταν συνδεδεμένος ο sink κόμβος. Τα βήματα που ακολουθεί αυτός ο αλγόριθμος είναι τα εξής:

1. Αρχικοποίηση μεταβλητών και σειριακού διαύλου επικοινωνίας με το `arduino`.
2. Αποστολή στη σειριακή θύρα του σήματος εκκίνησης προς τον κόμβο sink.

3. Αναμονή για εισερχόμενο σήμα στη σειριακή θύρα στην οποία είναι συνδεδεμένος ο sink κόμβος.
4. Μορφοποίηση και αποθήκευση των εισερχομένων δεδομένων σε αρχεία τύπου .csv (Comma Separated Values).
5. Επανάληψη του βήματος 3 και 4 έως ότου ανιχνευτεί ο θάνατος του δικτύου.

Γ'.5.1 Αναγκαιότητα Χρήσης

Η χρήση εξωτερικού προγράμματος για τη διαχείριση και αποθήκευση των δεδομένων που παράγει το πείραμα αφαιρεί σημαντικό επεξεργαστικό φορτίο από τον επεξεργαστικό κύκλο του sink κόμβου, επιτρέποντας έτσι σε πιο γρήγορη και risk-free εκτέλεση του πειράματος. Η γλώσσα Python είναι γνωστή για την ιδανική αναλογία πολυπλοκότητας/χρηστικότητας των προγραμμάτων, τα οποία αναπτύσσονται σε αυτήν, και υποστηρίζεται από μια τεράστια γκάμα βιβλιοθηκών και επεκτάσεων για όλες τις απλές και σύνθετες προγραμματιστικές απαιτήσεις. Ειδικά όσον αφορά τη διαχείριση δεδομένων εύκολα και αποδοτικά, η επιλογή της Python για την ανάπτυξη του προγράμματος ήταν λογική και αποδοτική.

Γ'.5.2 Βιβλιοθήκες Python

Για το serialToCsv.py script έγινε χρήση δύο βιβλιοθηκών της Python.

- **serial module - PySerial:** Πρόκειται για μια βιβλιοθήκη η οποία περιέχει απαραίτητες μεθόδους για την εύκολη αναγνώριση σειριακών θυρών στο σύστημα και τη λήψη ή αποστολή σημάτων μέσω αυτών. Λειτουργεί σε όλα τα βασικά λειτουργικά συστήματα (Windows, MAC OS, Linux, BSD), και κατά την κλήση της αναγνωρίζει αυτόματα το κατάλληλο backend. [23]
- **csv module - Πρότυπη Βιβλιοθήκη Python:** Η βιβλιοθήκη csv περιέχει κλάσεις οι οποίες επιτρέπουν την εύκολη ανάγνωση και εγγραφή δεδομένων σε αρχεία με τη χρήση των πρότυπων της μορφής .csv.
- **time module - Πρότυπη Βιβλιοθήκη Python:** Η βιβλιοθήκη time προσφέρει δυνατότητες χρονομέτρησης που χρειάζονται για την καταγραφή των δεδομένων. Επιπλέον, με τη χρήση της μπορούμε να προσθέτουμε timestamps στα δεδομένα που λαμβάνουμε.

Γ'.5.3 Ανάλυση Αλγορίθμου

Πρόκειται για ένα αρκετά απλό πρόγραμμα το οποίο "ακούει" για εισερχόμενα πακέτα στη σειριακή διεπαφή και αφού τα μετατρέψει, τα αποθηκεύει σε αρχεία τύπου .csv.

Γ'.5.3.1 Κώδικας

```

1 import csv
2 import datetime
3 from datetime import datetime as dt
4 import serial
5
6 addressT = {'013A2041826C72': 'node1', '013A2041826CCE': 'node2', '013
    A2041826CEC': 'node3', '013A2041826831' : 'node4', '013A20418268A3' : '
    node5', '013A2041826CBF' : 'node6', '013A2041826CAE' : 'node7', '013
    A2041826C6C' : 'node8', '013A2041826CA2' : 'node9'}
7 nodeInfo = {'node1': {'battery':100, 'lastRec': None}, 'node2': {'battery'
    :100, 'lastRec': None}, 'node3': {'battery':100, 'lastRec': None}, '
    node4': {'battery':100, 'lastRec': None}, 'node5': {'battery':100, '
    lastRec': None}, 'node6': {'battery':100, 'lastRec': None}, 'node7': {'
    battery':100, 'lastRec': None}, 'node8': {'battery':100, 'lastRec':
    None}, 'node9': {'battery':100, 'lastRec': None}}
8 arduino = serial.Serial(port = 'COM2', baudrate=115200)
9 fRaw = open('RAW_Static.csv', 'w')
10 fLog = open('AdjustableRouting.csv', 'w')
11 fLog.write('Time,Node 1,Node 2,Node 3,Node 4,Node 5,Node 6,Node 7,Node 8,
    Node 9\n')
12 arduino.write(bytes('start','utf-8'))
13 logInterval = dt.strptime('2','%S')
14 nextLog = datetime.now()
15 timeOut = dt.strptime('120','%S')
16 while True:
17     existsPacket = arduino.inWaiting()
18     currentTime = dt.now()
19     if existsPacket:
20         data = arduino.readline()
21         if('Got Packet!' in data):
22             fRaw.write('DATA - '+dt.strftime('%H:%M:%S:%f',currentTime)[:3]+'\\n'
                )
23             continue
24     else:

```

```

25     savedData = data.split(',')
26     nodesInPacket = []
27     for node in savedData:
28         tempBat = node[0]
29         tempNode = addressT[node[1:]]
30         nodeInfo[tempHex]['battery'] = tempBat
31         nodeInfo[tempHex]['lastRec'] = currentTime
32         nodesInPacket.append(tempNode)
33     for item in nodesInPacket:
34         fRaw.write(str(nodeInfo[item]['battery'])+','+item)
35 if currentTime > nextLog:
36     nextLog = currentTime + logInterval
37     fLog.write(dt.strftime('%H:%M:%S:%f',currentTime)[:3])
38     for node in nodeInfo.keys():
39         fLog.write(','+nodeInfo[node]['battery'])
40     fLog.write('\n')
41     for node in nodeInfo.keys():
42         if (((currentTime - nodeInfo[node]['lastRec']) > timeOut) and
43             nodeInfo[node]['battery'] != 0):

```

Κώδικας C.1: Κώδικας *serialToCsv.py*

Στον κώδικα έχουμε αποθηκευμένες τις διευθύνσεις των κόμβων του δικτύου αντιστοιχησμένες με ένα φιλικό όνομα (γραμμή 6). Έχουμε και ένα δεύτερο λεξικό στο οποίο για κάθε κόμβο αποθηκεύουμε το επίπεδο της μπαταρίας του και την τελευταία χρονική στιγμή που μετέδωσε με επιτυχία στον sink κόμβο (γραμμή 7). Αρχικοποιείται η σειριακή διεπαφή στη θύρα που είναι συνδεδεμένο το arduino και οι υπόλοιπες μεταβλητές (γραμμές 8-15). Ελέγχουμε αν υπάρχει διαθέσιμο πακέτο (γραμμή 17), και αποθηκεύουμε τα δεδομένα σε μεταβλητή (γραμμή 20). Ύστερα γίνεται ο έλεγχος αν το εισερχόμενο μήνυμα είναι το διαχωριστικό ανάμεσα στις μεταδόσεις (γραμμή 21), κι αν είναι, γράφει τη διαχωριστική γραμμή στο Raw αρχείο. Αν δεν πρόκειται για το διαχωριστικό μήνυμα, χωρίζει τα δεδομένα σε τμήματα ανά κόμβο (γραμμή 25) και αποθηκεύει τα απαραίτητα δεδομένα για τον κάθε κόμβο και γράφει τα δεδομένα κατάλληλα μορφοποιημένα στο Raw αρχείο (γραμμές 27-32). Επιπλέον, κάθε 2 δευτερόλεπτα (γραμμή 13), καταγράφει γενικά δεδομένα στο αρχείο log με κατάλληλη μορφοποίηση (γραμμές 35-40) και ελέγχει αν κάποιος κόμβος δε μπορεί πλέον να μεταδώσει δεδομένα προς τον sink, ώστε να τερματίσει την εκτέλεσή του.

Γ'.5.4 Εναλλακτικές Επιλογές για την Καταγραφή Δεδομένων

Η πρώτη και πιο προφανής επίλυση του θέματος της καταγραφής των δεδομένων ήταν η χρήση κάρτας microSD στον κόμβο sink. Ο τρόπος λειτουργίας αυτής της μεθόδου θα ήταν η εγγραφή μιας σειράς δεδομένων για κάθε εισερχόμενο σήμα από τους κόμβους του δικτύου. Μετά το πέρας του πειράματος, τα δεδομένα της κάρτα μνήμης θα αποσαφηνίζονταν με τη χρήση ενός άλλου προγράμματος είτε θα μεταφέρονταν απευθείας σε κάποιο πρόγραμμα επεξεργασίας υπολογιστικών φύλλων. Αυτή η διαδικασία θα αύξανε σημαντικά τον φόρτο εργασίας στον sink κόμβο, το οποίο θα απειλούσε το έγκαιρο άδειασμα του receive buffer της ασύρματης του μονάδας. Αυτό είτε θα μείωνε την ακρίβεια των δεδομένων είτε θα σήμαινε μεγαλύτερα χρονικά διαστήματα ανάμεσα στις αποστολές σημάτων από τους κόμβους, με σκοπό την ελάφρυνση του φόρτου εργασίας του sink. Μία ακόμη εναλλακτική λύση ήταν ο κάθε κόμβος να περιέχει μια κάρτα μνήμης και να καταγράφει την κάθε χρονική στιγμή που στέλνει κάποιο σήμα στο δίκτυο, ώστε ο φόρτος εργασίας της εγγραφής σε εξωτερικό μέσο να μειωθεί δραματικά για τον sink κόμβο και να βρίσκεται σε αποδεκτά όρια για τον κάθε απομακρυσμένο κόμβο. Κατά τον τερματισμό του πειράματος, τα δεδομένα που έχει καταγράψει ο κάθε κόμβος συνδυάζονται και ταξινομούνται αναλόγως με τις καταγεγραμμένες χρονικές στιγμές της κάθε αποστολής ή λήψης. Αυτή η τεχνική, όμως, επιφέρει μεγάλο ρίσκο ανακρίβειας των δεδομένων, λόγω προβλημάτων συγχρονισμού και απόκλισης των ρολογιών του κάθε κόμβου, καθώς και αυξημένο κόστος ανά κόμβο, λόγω των επιπλέον καρτών μνήμης. Λόγω των προαναφερθέντων, αποφύγαμε την υλοποίηση κάποιας κατανεμημένης λύσης.

Γ'.6 Ο Κόμβος Sink

Ο ρόλος του sink κόμβου είναι ο συντονισμός των απομακρυσμένων κόμβων του δικτύου, η συλλογή δεδομένων από αυτούς καθώς και ο τερματισμός του αλγορίθμου, όταν δε μεταφέρονται πλέον πακέτα στο δίκτυο.

Γ'.6.1 Αλγοριθμική Ροή του Sink Κόμβου

Ο sink κόμβος πρέπει να εκτελεί τις εξής λειτουργίες. Αρχικά, αναμένει σήμα από τη σειριακή διεπαφή το οποίο θα σηματοδοτεί την εκκίνηση του πειράματος κι επομένως του δικτύου μας. Αφού λάβει το σήμα εκκίνησης, ανιχνεύει τους γειτονικούς του κόμβους στο δίκτυο. Εφόσον βρει τουλάχιστον έναν κόμβο συνεχίζει στο επόμενο βήμα, αλλιώς τερματίζει την εκτέλεση

του αλγορίθμου. Αφού έχει βρει τους γείτονές του και υπάρχει τουλάχιστον ένας, στέλνει ένα trigger payload σε κάθε του γείτονα, το οποίο αντιπροσωπεύει την έναρξη της πρώτης από τις τέσσερις διακριτές φάσεις του αλγορίθμου. Έστερα από ένα λεπτό, το οποίο είναι αρκετά μεγάλο διάστημα, ώστε να έχουν λάβει όλοι οι κόμβοι στο δίκτυο το σήμα και να αναμένουν το επόμενο, μεταδίδει το επόμενο trigger payload. Αυτή η διαδικασία επαναλαμβάνεται άλλες δύο φορές, κι εφόσον όλα τα πακέτα σηματοδότησης έχουν αποσταλεί στους γείτονες, αναμένει συνεχώς για εισερχόμενα σήματα από τους κόμβους. Κάθε φορά που ένα σήμα φτάνει στη μονάδα ασύρματης επικοινωνίας, το καταναλώνει και το μεταφέρει στο σειριακό δίαυλο επικοινωνίας. Όταν δε λάβει μήνυμα για πάνω από ένα λεπτό, θεωρεί πως το δίκτυο έχει πεθάνει και τερματίζει. Τον πρώτο θάνατο κόμβου του δικτύου τον ανιχνεύει ο αλγόριθμος που τρέχει στον υπολογιστή, ο οποίος είναι υπεύθυνος για την ανάλυση και αποθήκευση των δεδομένων.

Γ'.6.2 Ανάλυση του Κώδικα

Ακολουθεί συνοπτική περιγραφή των κύριων μεθόδων του αλγορίθμου που εκτελείται από τον sink κόμβο. Παραλείπονται ορισμένες αρχικοποιήσεις μεταβλητών και αντικειμένων.

Γ'.6.2.1 Μέθοδος triggerNodes()

```

1 bool triggerNodes(int indexToSend){
2     int retryCount = 0;
3     bool finished = false;
4     uint8_t triggerPayload[] = {stagePayloads[indexToSend]};
5     ZBTxRequest zbtxRequest = ZBTxRequest();
6     ZBTxStatusResponse txStatus = ZBTxStatusResponse();
7     for(int i=0;i<numOfNeighbours;i++){
8         retryCount = 0;
9         finished = false;
10        while(retryCount!=RETRY_LIMIT&&!finished){
11            addr64 = XBeeAddress64(*(netNodesHigh+i),*(netNodesLow+i));
12            zbtxRequest = ZBTxRequest(addr64,triggerPayload,sizeof(triggerPayload));
13            xbee.send(zbtxRequest);
14            if(xbee.readPacket(200)){
15                if(xbee.getResponse().isAvailable()){
16                    if(xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE){

```

```

17         xbee.getResponse().getZBTxStatusResponse(txStatus);
18         if(txStatus.getDeliveryStatus() == SUCCESS){
19             finished = true;
20         }
21     }else{
22         retryCount++;
23     }
24 }
25 }
26 }
27 }
28 return true;
29 }

```

Κώδικας C.2: Μέθοδος *triggerNodes*

Για κάθε γειτονικό κόμβο (γραμμή 7), όσο δεν έχει γίνει επιτυχημένη αποστολή και οι προσπάθειες δεν ξεπερνάνε το όριο (γραμμή 10), στέλνει στον γειτονικό κόμβο *i* το πακέτο του σταδίου που επιδεικνύει το όρισμα της μεθόδου. Αν η αποστολή είναι επιτυχής (γραμμή 18), τότε επαναλαμβάνεται η διαδικασία για τον επόμενο γειτονικό κόμβο. Αυτή η διαδικασία εκτελείτε έως ότου έχει σταλεί το πακέτο σε όλους τους γειτονικούς κόμβους.

Γ'.6.2.2 Μέθοδος *readIncomingPackets()*

```

1 void readIncomingPackets(){
2     unsigned long time = millis();
3     unsigned long timeSinceLastRec = time;
4     unsigned long waitLimit = 60000;
5     int incomingLength = 0;
6     int counter = 0;
7     do{
8         time = millis();
9         xbee.readPacket();
10        if(xbee.getResponse().isAvailable()){
11            Serial.println("\nGot Packet,");
12            if(xbee.getResponse().getApiId() == ZB_RX_RESPONSE){
13                counter=0;
14                xbee.getResponse().getZBRxResponse(rxResponse);
15                incomingLength = (int)rxResponse.getDataLength();
16                if(incomingLength%9==0){

```

```

17         timeSinceLastRec = time;
18         for(int i=0;i<incomingLength;i++){
19             if(counter==9){
20                 counter=0;
21                 Serial.print(",");
22             }
23             if(counter==0){
24                 Serial.print(rxResponse.getData()[i],DEC);
25                 Serial.print(",");
26                 counter++;
27             }else{
28                 Serial.print(rxResponse.getData()[i],HEX);
29                 counter++;
30             }
31         }
32     }
33 }
34 }
35 }while(time-timeSinceLastRec < waitLimit);
36 }

```

Κώδικας C.3: Μέθοδος *readIncomingPackets*

Μέσα σε βρόγχο, όσο έρχονται πακέτα στον sink χωρίς να απέχουν μεταξύ τους χρονικό διάστημα μεγαλύτερο από ένα λεπτό (γραμμή 176 και 207), ελέγχει τον buffer για εισερχόμενο πακέτο (γραμμές 181-182). Αν ληφθεί πακέτο, το προωθεί στη σειριακή διεπαφή μορφοποιημένο κατάλληλα για χρήση σε αρχείο τύπου .csv (γραμμές 193, 196-197, 200). Ο έλεγχος της γραμμής 188 γίνεται έτσι ώστε να καταλάβει αν πρόκειται για σήμα το οποίο περιέχει πληροφορίες για τις μπαταρίες κόμβων του δικτύου, διότι το άθροισμα του μήκους μιας διεύθυνσης ενός κόμβου και του ποσοστού της μπαταρίας του είναι εννέα. Υπάρχει η δυνατότητα ένα σήμα να περιέχει πληροφορίες για πολλούς κόμβους, για αυτό γίνεται η επαναρχικοποίηση του μετρητή κάθε φορά που λαμβάνει πληροφορίες μήκους 9 (γραμμή 192).

Γ'.6.3 Θέση Κόμβου στο Δίκτυο

Η θέση του sink κόμβου στο δίκτυο επηρεάζει τη συνολική απόδοση του δικτύου όπως φαίνεται και από την πληθώρα βιβλιογραφίας η οποία εξετάζει το φαινόμενο. Για το συγκεκριμένο πείραμα, ο sink κόμβος βρίσκεται στο κέντρο του δικτύου. Με αυτήν τη θέση του sink

μπορούμε να παρατηρήσουμε την ενεργειακή 'τρύπα' που δημιουργείται γύρω από τον sink, καθώς και να εξετάσουμε την αποδοτικότητα της προσαρμοστικής προώθησης των πακέτων στο δίκτυο με τη χρήση ελέγχου των path vulnerabilities.

Γ'.7 Κόμβοι Αισθητήρων

Οι υπόλοιποι κόμβοι στο δίκτυο κατά την ενεργοποίηση μπαίνουν σε κατάσταση αναμονής έως ότου λάβουν το πρώτο σήμα ενεργοποίησης. Έστερα εκτελούν λειτουργίες για το κάθε στάδιο του αλγορίθμου. Ξεκινούν από την ανίχνευση των γειτόνων τους, συνεχίζουν με την εύρεση της ισχύος του καναλιού επικοινωνίας με τους γειτονικούς τους κόμβους, την εύρεση των αποστάσεών τους από τον sink κόμβο και των κατάλληλων γειτονικών κόμβων για την προώθηση των πακέτων τους, σε περίπτωση που δεν είναι γείτονες με τον sink ή δεν έχουν καλή ισχύ σήματος μαζί του, και τέλος ολοκληρώνουν, με τη διαδικασία αποστολής δεδομένων προς τον sink. Η μέθοδος που χρησιμοποιείται για την τυφλή προώθηση των trigger payloads σε όλο το δίκτυο είναι η τεχνική flooding. Σε περίπτωση που το flag "beSmart" έχει αρχικοποιηθεί σε αληθές, τότε εκτελούν κάποιες επιπλέον λειτουργίες κατά τη διαδικασία της αποστολής των δεδομένων σύμφωνα με τον εξεταζόμενο αλγόριθμο, με τις οποίες υπολογίζουν τα path vulnerabilities για τον κάθε πιθανό κόμβο που μπορούν να επιλέξουν για την προώθηση των πακέτων τους και επιλέγουν το πιο ασφαλές μονοπάτι. Αν το flag έχει αρχικοποιηθεί σε ψευδές, τότε χρησιμοποιείται στατική δρομολόγηση πακέτων και, συγκεκριμένα, επιλέγοντας τον κόμβο με την μικρότερη απόσταση προς τον sink.

Γ'.7.1 Απαιτήσεις

Οι απαιτήσεις του αλγορίθμου για τους κόμβους του δικτύου είναι η γρήγορη και αποτελεσματική ροή εκτέλεσης προκειμένου να μην υπάρχει πιθανότητα απώλειας εισερχομένων σημάτων από τον buffer. Ο κάθε κόμβος διατηρεί αρκετούς δυναμικά δεσμευμένους πίνακες, οι οποίοι κρατούν σημαντικές πληροφορίες για τους γείτονες και την κατάστασή τους, προκειμένου να είναι εφικτή η σωστή δρομολόγηση των πακέτων στο δίκτυο.

Γ'.7.2 Αλγοριθμική Ροή των Κόμβων Αισθητήρων

Ο κάθε κόμβος εκτελεί τον ίδιο κώδικα, ο οποίος εκτελεί τις παρακάτω λειτουργίες. Αρχικά, αφού αρχικοποιήσει τις μεταβλητές του, αναμένει για το σήμα εκκίνησης του αλγορίθμου και,

αφού το λάβει, βρίσκει τους γειτονικούς του κόμβους και τους προωθεί το σήμα εκκίνησης. Κόμβοι οι οποίοι έχουν ήδη λάβει το σήμα εκκίνησης το αγνοούν και αναμένουν για το σήμα του επόμενου σταδίου. Έτσι καταφέρνει ο αλγόριθμος να μεταφέρει το σήμα εκκίνησης σε όλους τους κόμβους του δικτύου. Όταν λάβει το σήμα του δεύτερου σταδίου, το προωθεί απευθείας και εκτελεί τη διαδικασία ανίχνευσης της ποιότητας του καναλιού επικοινωνίας με τον κάθε γείτονά του. Έπειτα αναμένει για το επόμενο σήμα, το οποίο προωθεί απευθείας στους γείτονές του και σε αυτό το στάδιο του αλγορίθμου, οι κόμβοι του δικτύου (εκτός του sink) εκτελούν μια διαδικασία κατά την οποία ανταλλάσσουν πληροφορίες για την απόστασή τους από τον sink και ανανεώνουν τους κατάλληλους πίνακες συνεχώς, έως ότου δεν υπάρχουν άλλες ενημερώσεις να πραγματοποιηθούν. Αυτή η διαδικασία πραγματοποιείται με την τεχνική που προτείνει ο αλγόριθμος Bellman - Ford. Αφού ολοκληρωθεί η διαδικασία αυτή, αναμένει για το επόμενο -και τελευταίο- σήμα από τον sink και αφού το λάβει και το προωθήσει, εκκινεί τη βασική διαδικασία του αλγορίθμου. Σε αυτήν τη διαδικασία, κάθε κόμβος προωθεί πακέτα προς τον sink κόμβο, τα οποία περιέχουν την ισχύ της μπαταρίας του, επιλέγοντας είτε το πιο σύντομο μονοπάτι, είτε προσαρμόζοντας το επιλεγόμενο μονοπάτι ανάλογα με το path vulnerability που έχει υπολογίσει. Όταν η τάση της μπαταρίας ενός κόμβου πέσει στα 3.3 Volts, σταματάει τη διαδικασία και τερματίζει.

Γ'.7.3 Μεθοδολογία Επίλυσης Κεντρικών Ζητούμενων

Σύμφωνα με τις απαιτήσεις και τη λογική του αλγορίθμου, τα παρακάτω ζητούμενα επιλύθηκαν με αλγοριθμική λογική. Έπειτα το κάθε τμήμα μετατράπηκε σε μέθοδο στον κώδικα ή τμήμα μεθόδου του κώδικα.

Γ'.7.3.1 Υπολογισμός της Ποιότητας Σήματος

Για τον υπολογισμό της ποιότητας του καναλιού επικοινωνίας με γειτονικούς κόμβους, ο κάθε κόμβος στέλνει ένα dummy πακέτο στους γείτονές του και στη συνέχεια, με τη χρήση της εντολής DB (Last Packet RSSI), που υλοποιεί το XBee module, ελέγχει την ισχύ του σήματος της μετάδοσης. Αυτή η τιμή χρησιμοποιείται σα δείκτης της απόστασης ενός κόμβου από τον άλλον. Αξίζει να σημειωθεί πως, ακόμα κι αν δύο κόμβοι είναι πιο κοντά τοποθετημένοι μεταξύ τους, στην περίπτωση που υπάρχουν εμπόδια ανάμεσά τους, αυτή η τιμή θα είναι μικρότερη από την τιμή ανάμεσα σε δύο κόμβους χωρίς εμπόδια αλλά με μεγαλύτερη απόσταση. Επομένως, αυτός ο δείκτης αντιπροσωπεύει, εκτός από φυσική απόσταση, και την ενεργειακή κατανάλωση

της μετάδοσης. Η τιμή που επιστρέφεται από την εντολή DB μπορεί να είναι από -40dBm έως -110dBm (όσο μικρότερος ο αριθμός τόσο πιο αδύναμο το σήμα). Οι τιμές που επιστρέφονται είναι στο δεκαεξαδικό σύστημα (0x28 - 0x6E). Αυτή η διαδικασία υλοποιείται στη μέθοδο findDB().

Γ'.7.3.2 Υπολογισμός της Ελάχιστης Απόστασης από τον sink

Για την εύρεση της ελάχιστης απόστασης του κάθε κόμβου από τον sink, βασιζόμαστε στη λογική του αλγορίθμου εύρεσης μικρότερου μονοπατιού Bellman - Ford. Ο αλγόριθμος αυτός έχει κάποιες απαιτήσεις. Αρχικά, οι κόμβοι πρέπει, θεωρητικά, να αρχικοποιούν την απόστασή τους στο $+\infty$. Αυτό προγραμματιστικά δε γίνεται, επομένως χρησιμοποιούμε τη μέγιστη τιμή που μπορεί να λάβει ο τύπος δεδομένων της μεταβλητής που κρατάει την απόσταση του κόμβου από τον sink². Έπειτα, κατά τη διαδικασία της "χαλάρωσης" προς τη λύση, πρέπει να γίνεται ο έλεγχος για το αν ο κόμβος που έχει στείλει την τιμή της απόστασής του από τον sink είναι eligible για να του προωθηθούν τα πακέτα. Αν ο γειτονικός κόμβος που ελέγχεται έχει μικρότερη απόσταση από τον sink, τότε είναι eligible. Σε αντίθετη περίπτωση, δεν είναι. Για τον υπολογισμό της απόστασης ενός απομακρυσμένου κόμβου από τον sink, γίνεται πρόσθεση των αποστάσεων των eligible γειτόνων του και της απόστασης του ίδιου από αυτόν. Αφού ολοκληρωθεί η ο αλγόριθμος Bellman-Ford, κάθε κόμβος γνωρίζει πώς να δρομολογήσει τα πακέτα του για να ακολουθήσει το μικρότερο δυνατό μονοπάτι προς τον sink. Επιπλέον, κάθε κόμβος γνωρίζει ποιοι γείτονές τους είναι eligible για την προώθηση των πακέτων του σε περίπτωση αλλαγής μονοπατιού όταν λειτουργεί η μέθοδος δρομολόγησης με path vulnerabilities. Η διαδικασία αυτή υλοποιείται στη μέθοδο manageDB().

Γ'.7.3.3 Υπολογισμός Path Vulnerability

Για τον υπολογισμό της ευπάθειας ενός μονοπατιού (Path Vulnerability), κάθε κόμβος πρέπει να έχει γνώση των ευπαθειών όλων των eligible parent κόμβων του. Αφού έχει λάβει αυτές τις πληροφορίες μπορεί να υπολογίσει τη δική του ευπάθεια και να προωθήσει τις πληροφορίες αυτές σε όλους τους non-eligible parent κόμβους του. Έτσι, όλοι οι κόμβοι στο δίκτυο αποκτούν τις απαραίτητες γνώσεις, ώστε να μπορούν να ακολουθήσουν το μονοπάτι με το μικρότερο path vulnerability. Ως δείκτης ευπάθειας ενός μονοπατιού ορίζεται η χαμηλότερη τιμή της κατάστασης της μπαταρίας ενός κόμβου που ανήκει σε αυτό το μονοπάτι. Η διαδικασία

² Η μέγιστη τιμή του τύπου δεδομένων με χρήση σταθεράς από τη βιβλιοθήκη limits.h

υπολογισμού των path vulnerabilities επαναλαμβάνεται ανά τακτά χρονικά διαστήματα προκειμένου να ανανεώνονται οι τιμές που παράγουν και έτσι οι κόμβοι να μπορούν να επιλέγουν το κατάλληλο και ενεργειακά αποδοτικότερο μονοπάτι.

Γ'.7.3.4 Έλεγχος Κατάστασης Μπαταρίας

Η διαδικασία ελέγχου της κατάστασης της μπαταρίας ενός κόμβου ακολουθεί την εξής λογική. Αρχικά, στην αρχικοποίηση των μεταβλητών, ανιχνεύουμε την τάση αναφοράς (internal reference voltage) του μοντέλου Arduino που χρησιμοποιούμε. Αυτό γίνεται τρέχοντας έναν συγκεκριμένο κώδικα στην πλακέτα ο οποίος επιστρέφει την τιμή. Για τη δική μας πλακέτα η τιμή ήταν 1.08. Αφού έχουμε βρει αυτήν τη τιμή, με χρήση του καταχωρητή ADCSRA (ADC Control and Status Register A) του Arduino, μπορούμε να βρούμε την πραγματική τάση της μπαταρίας σε Volts, και μετά να την μετατρέψουμε σε ποσοστό. Αν η τάση είναι κάτω από 3.3 Volts θεωρείται ότι είναι 0%.

Γ'.7.3.5 Εύρεση Χρονικής Στιγμής Πρώτου Θανάτου

Η χρονική στιγμή πρώτου θανάτου του δικτύου (First Death Time) ορίζεται από τη χρονική στιγμή κατά την οποία ένας κόμβος στο δίκτυο δε μεταδίδει πλέον δεδομένα λόγω εξάντλησης της μπαταρίας του. Τη χρονική στιγμή αυτήν την ανιχνεύει ο αλγόριθμος που εκτελείται στον υπολογιστή (serialToCsv.py).

Γ'.7.4 Ανάλυση του Κώδικα

Ακολουθεί η ανάλυση του κώδικα των κύριων μεθόδων του αλγορίθμου που εκτελείται από τους κόμβους του δικτύου. Παραλείπονται ορισμένες αρχικοποιήσεις και κομμάτια κώδικα τα οποία είναι αρκετά προφανή.

Γ'.7.4.1 Μέθοδος waitForTrigger()

```
1 bool waitForTrigger(int indexToExpect){
2     while(1){
3         xbee.readPacket();
4         if(xbee.getResponse().isAvailable()){
5             if(xbee.getResponse().getApiId() == ZB_RX_RESPONSE){
```

```

6      xbee.getResponse().getZBRxResponse(rxResponse);
7      uint8_t incoming = rxResponse.getData()[0];
8      if((incoming == expectedIncomingPayload[indexToExpect]+0x01&&
indexToExpect==0)|| (incoming == expectedIncomingPayload[indexToExpect])
9      ){
10         addressMSB = rxResponse.getRemoteAddress64().getMsb();
11         addressLSB = rxResponse.getRemoteAddress64().getLsb();
12         if(indexToExpect == 0&&(incoming == (expectedIncomingPayload[
indexToExpect]))){
13             sinkIsNeighbour = true;
14             sinkIndex = 0;
15             *(nTableMSB+numOfNeighbours) = addressMSB;
16             *(nTableLSB+numOfNeighbours) = addressLSB;
17             *(nTableIsEligible+numOfNeighbours) = true;
18             numOfNeighbours++;
19             nTableMSB = realloc(nTableMSB,(numOfNeighbours+1)*sizeof(
uint32_t));
20             nTableLSB = realloc(nTableLSB,(numOfNeighbours+1)*sizeof(
uint32_t));
21             nTableIsEligible = realloc(nTableIsEligible,(numOfNeighbours+1)
* sizeof(uint8_t));
22         }
23         return true;

```

Κώδικας C.4: Μέθοδος *waitForTrigger*

Εκτελεί έναν ατέρμων βρόγχο έως ότου λάβει το πακέτο σταδίου που αναμένεται (γραμμές 2 και 8). Αν πρόκειται για το πρώτο στάδιο, γίνεται έλεγχος για το αν ο sink είναι γείτονας (γραμμή 11), κι αν είναι τον προσθέτει στον πίνακα γειτνίασης και στον πίνακα των eligible κόμβων (γραμμές 13-20). Ο ατέρμων βρόγχος διακόπτεται με την επιστροφή της μεθόδου (γραμμή 22).

Γ'.7.4.2 Μέθοδος *pushTrigger()*

```

1 bool pushTrigger(int indexToSend){
2     bool atLeastOne = false;
3     int failed = 0;
4     int retryLimit = 5;
5     uint8_t triggerPayload[] = {expectedIncomingPayload[indexToSend]};
6     if(indexToSend==0&&sinkIsNeighbour){

```

```

7     triggerPayload[0] += 0x01;
8 }
9 for(int i=0; i<numOfNeighbours; i++){
10    if((addressLSB != (*(nTableLSB+i))) || (addressMSB != (*(nTableMSB+i)))){
11        atLeastOne = true;
12        addr64 = XBeeAddress64(*(nTableMSB+i), *(nTableLSB+i));
13        zbtxRequest = ZBTxRequest(addr64, triggerPayload, sizeof(triggerPayload));
14    };
15    xbee.send(zbtxRequest);
16    xbee.readPacket(200);
17    if(xbee.getResponse().isAvailable()){
18        if(xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE){
19            xbee.getResponse().getZBTxStatusResponse(zbtxStatus);
20            if(zbtxStatus.getDeliveryStatus() != SUCCESS && retryLimit >= failed)
21            {
22                failed++;
23                i--;
24            }else{
25                failed = 0;
26            }
27        }else if(failed <= retryLimit){
28            failed++;
29            i--;
30        }else{
31            failed = 0;
32        }
33    }
34    return atLeastOne;
35 }

```

Κώδικας C.5: Μέθοδος *pushTrigger*

Αρχικά, εάν έχει λάβει το trigger από τον sink κόμβο, προκειμένου να το προωθήσει σε κάποιον άλλον κόμβο, πρώτα το τροποποιεί ώστε να μη θεωρήσει ο επόμενος κόμβος ότι το έλαβε από τον sink (γραμμές 6-8). Έπειτα, για κάθε γειτονικό κόμβο (γραμμή 9) εκτός από τον κόμβο από τον οποίο έλαβε το σήμα (γραμμή 10), δημιουργεί και μεταδίδει ένα πακέτο το οποίο περιέχει το trigger (γραμμές 12-14). Τέλος, ελέγχει αν ήταν επιτυχημένη η μετάδοση προτού συνεχίσει με τον επόμενο κόμβο (γραμμές 15-19).

Γ'.7.4.3 Μέθοδος findDB()

```

1  for(int i=0;i<numOfNeighbours;i++){
2      do{
3          time = millis();
4          passed1 = false;
5          passed2 = false;
6          if(time>=waitTime){
7              if(!passed1){
8                  addr64 = XBeeAddress64(*(nTableMSB+i),*(nTableLSB+i));
9                  ratcomreq = RemoteAtCommandRequest(addr64,dummyPayload);
10                 xbee.readPacket();
11                 xbee.send(ratcomreq);
12                 xbee.readPacket(100);
13                 if(xbee.getResponse().isAvailable()){
14                     if(xbee.getResponse().getApiId() == REMOTE_AT_COMMAND_RESPONSE)
15                 {
16                     passed1 = true;
17                 }
18             }
19             if(passed1&&!passed2){
20                 atcomreq = AtCommandRequest(DBPayload);
21                 xbee.readPacket();
22                 xbee.send(atcomreq);
23                 xbee.readPacket(100);
24                 if(xbee.getResponse().isAvailable()){
25                     if(xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE){
26                         xbee.getResponse().getAtCommandResponse(atcomres);
27                         if(atcomres.isOk()){
28                             passed2 = true;
29                             uint8_t tempHex = 0x00;
30                             tempHex = atcomres.getValue()[0];
31                             if(i==sinkIndex){
32                                 dts.distToSink32 = 0x00;
33                                 dts.distToSink8[0] = tempHex;
34                                 sinkDistChange = true;
35                             }
36                             *(nTableDB+i) = 0x00;
37                             *(nTableDB+i) = tempHex;
38                             nTableDB = realloc(nTableDB,(i+1)*sizeof(uint32_t));
39                         }

```

```

40         }else{
41             passed1 = false;
42             waitTime = time + 200;
43         }
44     }
45 }
46 }
47 }while ((!passed1 || !passed2) && time - startingTime <= tempTimeout);
48 }
49 }

```

Κώδικας C.6: Μέθοδος *findDB*

Για κάθε γειτονικό κόμβο τρέχει έναν βρόγχο (γραμμές 1-2). Όσο δεν έχουν ολοκληρωθεί τα δύο απαραίτητα βήματα της ανίχνευσης της ισχύος του σήματος με τον γειτονικό κόμβο, η διαδικασία επαναλαμβάνεται, εκτός αν η χρονική διάρκεια από τη στιγμή που ξεκίνησε η μέθοδος ξεπεράσει το ορισμένο όριο (γραμμή 47). Το πρώτο στάδιο αποτελείται από την επιτυχή αποστολή ενός dummy payload (γραμμές 7-18). Το δεύτερο στάδιο αποτελείται από την αίτηση προς την ασύρματη μονάδα του κόμβου για την ισχύ του σήματος που στάλθηκε (γραμμές 19-27) και την αποθήκευση της τιμής αυτής, η οποία αντιπροσωπεύει την τιμή RSSI και επομένως, την απόσταση των δύο κόμβων σύμφωνα με τις προδιαγραφές του αλγορίθμου που ορίσαμε. Ταυτόχρονα, αν ο κόμβος που ελέγχθηκε ήταν ο sink, ενημερώνεται η μεταβλητή που κρατάει την απόσταση του κόμβου από τον sink (γραμμές 31-34). Πριν συνεχίσει ο βρόγχος με τον επόμενο γειτονικό κόμβο, αποθηκεύει την τιμή RSSI στον αντίστοιχο δυναμικό πίνακα, του οποίου ύστερα αυξάνει το δεσμευμένο χώρο (γραμμές 36-38).

Γ'.7.4.4 Μέθοδος *manageDB()*

Η μέθοδος αυτή υλοποιεί τη λογική του αλγορίθμου Bellman - Ford, προσαρμοσμένη στις παραμέτρους της εργασίας. Είναι μια εκτενής μέθοδος οπότε αναλύεται σε τρία μέρη.

```

1  do{
2      time = millis();
3      xbee.readPacket();
4      if(xbee.getResponse().isAvailable()){
5          isNeighbour = false;
6          if(xbee.getResponse().getApiId() == ZB_RX_RESPONSE){
7              xbee.getResponse().getZBRxResponse(rxResponse);
8              if(rxResponse.getDataLength() == 4){

```

```

9      addressMSB = rxResponse.getRemoteAddress64().getMsb();
10     addressLSB = rxResponse.getRemoteAddress64().getLsb();
11     for(int l=0;l<4;l++){
12         indts.incomingDist8[l] = rxResponse.getData()[l];
13     }
14     for(int j=0;j<numOfNeighbours;j++){
15         if((addressLSB == (*(nTableLSB+j)))&&(addressMSB == (*(
nTableMSB+j)))){
16             isNeighbour = true;
17             addr64 = XBeeAddress64(*(nTableMSB+j),*(nTableLSB+j));
18             zbtxRequest = ZBTxRequest(addr64,tempPackage,sizeof(
tempPackage));
19             xbee.send(zbtxRequest);
20             gotFromNeighbour[j] = true;

```

Κώδικας C.7: Μέθοδος *manageDB* - Επικοινωνία

Μέσα σε βρόγχο, όταν λαμβάνει εισερχόμενο πακέτο αποθηκεύει προσωρινά τη διεύθυνση του αποστολέα καθώς και το περιεχόμενο το οποίο αντιπροσωπεύει την απόσταση του αποστολέα από τον sink κόμβο (γραμμές 1-12). Έπειτα ελέγχει αν ο αποστολέας ανήκει στον πίνακα γειτνίασης (γραμμές 14-15). Αν ανήκει, του απαντάει με ένα προκαθορισμένο πακέτο το οποίο λειτουργεί σαν acknowledgement (ACK) (γραμμές 17-19), και αποθηκεύει το γεγονός ότι έχει λάβει ενημέρωση από τον συγκεκριμένο κόμβο (γραμμή 20).

```

1      if((unsigned int)(indts.incomingDist32)<(unsigned int)(dts.
distToSink32)){
2          *(nTableIsEligible+j) = true;
3          uint32_t tempDistToNode = 0x00;
4          tempDistToNode += *(nTableDB+j);
5          if(((unsigned int)indts.incomingDist32+(unsigned int)
tempDistToNode)<(unsigned int)dts.distToSink32){
6              dts.distToSink32 = indts.incomingDist32+tempDistToNode;
7              sinkIndex = j;
8              newSinkDistChange = true;
9              sinkPathChanged = true;
10         }
11     }else{
12         *(nTableIsEligible+j) = false;
13     }

```

```

14         }
15     }
16     if(!isNeighbour){
17         *(nTableMSB+numOfNeighbours) = addressMSB;
18         *(nTableLSB+numOfNeighbours) = addressLSB;
19         *(nTableIsEligible+numOfNeighbours) = false;
20         numOfNeighbours++;
21         nTableMSB = realloc(nTableMSB,(numOfNeighbours+1)*sizeof(
uint32_t));
22         nTableLSB = realloc(nTableLSB,(numOfNeighbours+1)*sizeof(
uint32_t));
23         nTableIsEligible = realloc(nTableIsEligible,(numOfNeighbours+1)
*sizeof(uint8_t));
24     }
25     }else{
26         addressMSB = rxResponse.getRemoteAddress64().getMsb();
27         addressLSB = rxResponse.getRemoteAddress64().getLsb();
28         for(int j=0;j<numOfNeighbours;j++){
29             if((addressLSB == *(nTableLSB+j))&&(addressMSB == (*(
nTableMSB+j)))){
30                 uint8_t incoming = rxResponse.getData()[0];
31                 if((char)incoming==(char)tempPackage[0]){
32                     successfulTr[j]=true;

```

Κώδικας C.8: Μέθοδος *manageDB* - Εύρεση Ελάχιστων Αποστάσεων

Συνεχίζοντας, ελέγχει αν η απόσταση του αποστολέα από τον sink είναι μικρότερη από τη δική του και εαν είναι τον προσθέτει στους eligible κόμβους (γραμμές 1-2). Έπτερα ελέγχει αν το άθροισμα των αποστάσεων ανάμεσά τους, και της απόστασης του αποστολέα από τον sink είναι μικρότερη από την τωρινή απόσταση από τον sink. Αν είναι, ο αποστολέας επιλέγεται για την προώθηση των πακέτων προς τον sink εφόσον προσφέρει το γρηγορότερο μονοπάτι (γραμμές 5-9). Εάν ο αποστολέας δεν ανήκει στον πίνακα γειτνίασης, τον προσθέτει και τον ορίζει σαν eligible (γραμμές 16-24). Αν το πακέτο που ελήφθη δεν είναι μήκους 4, σημαίνει πως είναι κάποιο ACK από γειτονικό κόμβο (γραμμή 25). Σε αυτήν την περίπτωση, ενημερώνεται ο πίνακας successfulTr (γραμμές 26-32).

```

1         if(i>=numOfNeighbours&&!sentToAll&&time-timeSinceLastTransmit>
transmitLimit){

```

```

2      sentToAll = true;
3      for(int l=0;l<numOfNeighbours;l++){
4          if(!successfulTr[l]){
5              sentToAll = false;
6              addr64 = XBeeAddress64(*(nTableMSB+l),*(nTableLSB+l));
7              zbtxRequest = ZBTxRequest(addr64,dts.distToSink8,sizeof(dts.
distToSink8));
8              xbee.send(zbtxRequest);
9          }
10     }
11     timeSinceLastTransmit = millis();
12 }

```

Κώδικας C.9: Μέθοδος *manageDB* - Ενημέρωση Γειτονικών Κόμβων

Αυτό είναι το τμήμα του κώδικα το οποίο εκτελεί τις μεταδόσεις προς τους κόμβους οι οποίοι δεν έχουν λάβει ενημέρωση. Ελέγχει αν δεν έχει στείλει σε κάποιους και αν είναι η ώρα να στείλει (γραμμή 1). Σε περίπτωση που η απόσταση ενός κόμβου από τον sink αλλάξει, αυτή η διαδικασία επαναλαμβάνεται από τη μεριά του κόμβου που υπέστη την αλλαγή. Έτσι επιτυγχάνουμε την εύρεση των shortest paths με κύκλους "χαλάρωσης" των ακμών (relax edges).

Γ'.7.4.5 Μέθοδος *getBattVolts()*

```

1 int getBattVolts() { // Returns actual value of Vcc (x 100)
2     ADCSRA |= _BV( ADSC ); // Start a conversion
3     while( ( (ADCSRA & (1<<ADSC)) != 0 ) ); // Wait for it to complete
4     int results = (((InternalReferenceVoltage * 1024L) / ADC) + 100L) / 10L;
5     // Scale the value
6     return results;
7 }

```

Κώδικας C.10: Μέθοδος *getBattVolts*

Χρησιμοποιώντας τον καταχωρητή ADCSRA και θέτοντας το bit ADSC στην τιμή 1, εκκινεί τη διαδικασία μετατροπής AD (γραμμή 2). Περιμένει να ολοκληρωθεί η μετατροπή (γραμμή 3). Έπειτα κλιμακώνει το αποτέλεσμα και επιστρέφει την πραγματική τάση της εισόδου Vcc, δηλαδή της μπαταρίας (γραμμές 4-5).

Γ'.7.4.6 Μέθοδος beginTransmitting()

Η μέθοδος αυτή υλοποιεί τον αλγοριθμικό κορμό της εργασίας. Λόγω του μεγέθους της, αναλύεται παρακάτω μόνο ο τρόπος με τον οποίο διαχειρίζεται τα Path Vulnerabilities.

Εύρεση των Path Vulnerabilities

```

1      if(incoming!=expectedIncomingPayload[3]){
2          addressMSB = rxResponse.getRemoteAddress64().getMsb();
3          addressLSB = rxResponse.getRemoteAddress64().getLsb();
4          for(int i=0;i<numOfNeighbours;i++){
5              if((*nTableMSB+i)==addressMSB&&(*nTableLSB+i)==addressLSB)
6          ){
7              nTablePathVulnerability[i][0] = incoming+0x00;
8              lastSeen[i] = time;
9          }
10         }
11     }else if(incomingLength==numOfNodes&&time-startingTime>
PATH_UPDATE_INTERVAL){
12         addressMSB = rxResponse.getRemoteAddress64().getMsb();
13         addressLSB = rxResponse.getRemoteAddress64().getLsb();
14         uint8_t incoming;
15         for(int i=0;i<numOfNeighbours;i++){
16             if((*nTableMSB+i)==addressMSB&&(*nTableLSB+i)==addressLSB)){
17                 if((*nTableIsEligible+i)){
18                     for(int j=0;j<numOfNodes;j++){
19                         incoming = rxResponse.getData()[j];
20                         nTablePathVulnerability[i][j] = incoming+0x00;
21                     }
22                     lastSeen[i] = time;
23                 }
24             }
25         }
26         pathVulnerabilityUpdate = true;

```

Κώδικας C.11: Μέθοδος *beginTransmitting* - Εύρεση των Path Vulnerabilities

Αν λάβει σήμα το οποίο δεν αντιστοιχεί στο αναμενόμενο, σημαίνει πως ο αποστολέας ενημερώνει για την κατάσταση της μπαταρίας του (γραμμές 1 και 6). Διαφορετικά, το εισερχόμενο

μήνυμα περιέχει πληροφορίες για όλους τους γειτονικούς κόμβους του αποστολέα, οι οποίες είναι απαραίτητες για τον υπολογισμό των path vulnerabilities (γραμμή 11). Σε αυτήν την περίπτωση, ελέγχει με διπλό βρόγχο και ενημερώνει τον πίνακα ο οποίος περιέχει την κατάσταση της μπαταρίας όλων των γειτονικών κόμβων του αποστολέα (γραμμές 15-20).

Εύρεση Βέλτιστου Μονοπατιού

```

1      for(int i=0;i<numOfNeighbours;i++){
2          if(*(nTableIsEligible+i)&&stillNeighbour[i]){
3              for(int j=0;j<numOfNodes;j++){
4                  if(nTablePathVulnerability[i][j]+0x00>=max&&
nTablePathVulnerability[i][j]+0x00!=0xff){
5                      tempPosMax = i;
6                      max = nTablePathVulnerability[i][j]+0x00;
7                  }
8                  if(nTablePathVulnerability[i][j]+0x00<min){
9                      tempPosMin = i;
10                     min = nTablePathVulnerability[i][j];
11                 }
12             }
13         }
14     }

```

Κώδικας C.12: Μέθοδος *beginTransmitting* - Εύρεση Βέλτιστου Μονοπατιού

Με χρήση απλής τεχνικής εύρεσης μέγιστου και ελάχιστου σε δισδιάστατο πίνακα, βρίσκει τον κόμβο που προσφέρει το βέλτιστο μονοπάτι με το μικρότερο path vulnerability (γραμμές 4-7) καθώς και το χειρότερο (γραμμές 8-11).

Εξαγωγή Συμπερασμάτων και Επιλογή Μονοπατιού

```

1      if(tempPosMax<0){
2          return;
3      }else if(tempPosMax!=sinkIndex){
4          sinkIndex = tempPosMax;
5          sinkPathChanged = true;
6      }
7      if(tempPosMin<0){

```

```

8      }else if(tempPosMin==sinkIndex){
9          max = 0x00;
10         for(int i=0;i<numOfNodes;i++){
11             if(nTablePathVulnerability[tempPosMin][i]+0x00>=max&&
nTablePathVulnerability[tempPosMin][i]+0x00!=0xff){
12                 max = nTablePathVulnerability[tempPosMin][i]+0x00;
13             }
14         }
15         for(int i=0;i<numOfNeighbours;i++){
16             if(i==tempPosMin){i++;}
17             for(int j=0;j<numOfNodes;j++){
18                 if(*(nTableIsEligible+i)&&stillNeighbour[i]){
19                     if(nTablePathVulnerability[i][j]+0x00>=max&&
nTablePathVulnerability[i][j]+0x00!=0xff){
20                         tempPosMax = i;
21                         max = nTablePathVulnerability[i][j]+0x00;
22                     }
23                 }
24             }
25         }
26         if(tempPosMin!=tempPosMax){
27             sinkIndex = tempPosMax;
28             sinkPathChanged = true;
29         }
30     }
31     maximumVulnerabilityPos = tempPosMin;
32 }

```

Κώδικας C.13: Μέθοδος *beginTransmitting* - Επιλογή Μονοπατιού

Αν δεν υπάρχει μονοπάτι προς τον sink, τερματίζει η εκτέλεση (γραμμές 1-2). Εάν το βέλτιστο μονοπάτι δεν είναι το ήδη επιλεγμένο (γραμμή 3), τότε ο νέος γείτονας στον οποίο προωθεί τα πακέτα του ο κόμβος αλλάζει (γραμμές 4-5). Αν ο κόμβος χρησιμοποιεί το πιο ευάλωτο μονοπάτι (γραμμή 8), τότε ψάχνει για ένα πιο ασφαλές (γραμμές 9-12). Αν βρει πιο ασφαλές, μεταφέρεται σε αυτό (γραμμές 15-29), αλλιώς παραμένει στο πιο ευάλωτο.

Γ'.7.5 Συνθήκη Τερματισμού

Ο τερματισμός του αλγορίθμου ορίζεται από τη χρονική στιγμή κατά την οποία δημιουργείται κενό στο δίκτυο και κάποιοι κόμβοι δε μπορούν να βρουν μονοπάτι προς τον sink. Το

φαινόμενο αυτό το ανιχνεύει το πρόγραμμα που επεξεργάζεται τα δεδομένα (serialToCsv.py), χρησιμοποιώντας μια μέθοδο για τον υπολογισμό του ρυθμού λήψης δεδομένων. Αν αυτός ο ρυθμός υποστεί απότομη μείωση, σηματοδοτεί το τέλος της ζωής του δικτύου (network lifespan) και παύει να λαμβάνει υπόψη επόμενα σήματα.

Γ'.8 Αποθήκευση και Ανάλυση Δεδομένων

Η επεξεργασία και αποθήκευση των δεδομένων υλοποιείται με τη χρήση ενός python script στον υπολογιστή που τροφοδοτεί τον κόμβο sink, όπως περιγράφεται στο κεφάλαιο Γ5 (σελίδα 25). Τα δεδομένα τα λαμβάνει ο υπολογιστής από τη σειριακή διεπαφή και περιέχουν ανεπεξέργαστα τα σήματα που λαμβάνει ο sink κόμβος. Από την εκτέλεση του αλγορίθμου, προκύπτουν τα δύο αρχεία καταγραφής δεδομένων AdjustableRouting.csv και StaticRouting.csv. Το πρώτο προκύπτει από την εκτέλεση του αλγορίθμου με χρήση προσαρμογής μονοπατιού βάσει των path vulnerabilities και το δεύτερο από την εκτέλεσή του με στατική δρομολόγηση. Επιπλέον, προκύπτουν δύο αρχεία που περιέχουν τα ανεπεξέργαστα σήματα που έλαβε ο sink με ονόματα RAW_Adjustable.csv και RAW_Static.csv. Τα αρχεία αυτά περιέχουν πληροφορίες που αφορούν τις επιλογές δρομολόγησης που ακολούθησε ο κάθε κόμβος. Ακολουθούν δείγματα του κάθε αρχείου.

Γ'.8.1 Δομή Δεδομένων

Στους πίνακες Γ'.1α' και Γ'.2α' βλέπουμε τα δεδομένα που παράγει ο sink ανεπεξέργαστα, τα οποία καταγράφει ο υπολογιστής σε αρχείο .csv. Τα δεδομένα αυτά έχουν την εξής δομή. Η πρώτη σειρά ξεκινάει με τη συμβολοσειρά "DATA" ακολουθούμενη από μια παύλα και την χρονική στιγμή που καταγράφει ο υπολογιστής. Έπειτα, ακολουθούν σειρές δεδομένων με δομή: κατάσταση μπαταρίας - κόμβος. Ο τελευταίος κόμβος του κάθε κομματιού δεδομένων είναι γειτονικός κόμβος του sink, και οι κόμβοι πάνω από αυτόν, τον έχουν χρησιμοποιήσει για την προώθηση των δεδομένων τους. Στους πίνακες Γ'.1β' και Γ'.2β' βλέπουμε τα δεδομένα που παράγει το serialToCsv.py script από την ανάλυση των raw δεδομένων σύμφωνα με τη λογική που αναλύει το κεφάλαιο Γ'.5 (σελίδα 25). Στη στήλη "Time" καταγράφεται η χρονική στιγμή του ελέγχου. Στις στήλες N1...N9 καταγράφονται οι τιμές της κατάστασης των μπαταριών των κόμβων 1 έως 9 αντίστοιχα.

DATA - 12:41:54:513	
58	node8
75	node3
52	node1
DATA - 12:41:54:603	
71	node7
51	node2
DATA - 12:41:55:343	
66	node5
61	node9
59	node6
46	node4

(α') Δείγμα από *RAW_Static.csv*.

Πίνακας Γ'.1: Δεδομένα στατικής δρομολόγησης από την ίδια χρονική περίοδο.

DATA - 12:37:48:713	
72	node7
52	node2
DATA - 12:37:49:073	
66	node5
63	node9
61	node6
52	node4
DATA - 12:37:49:503	
76	node3
66	node5
60	node8
52	node1

(α') Δείγμα από *RAW_Adjustable.csv*.

Πίνακας Γ'.2: Δεδομένα προσαρμοστικής δρομολόγησης από την ίδια χρονική περίοδο.

Time	N1	N2	N3	N4	N5	N6	N7	N8	N9
12:41:52	52	51	75	46	66	59	71	58	61
12:41:54	52	51	75	46	66	59	71	58	61
12:41:56	52	51	75	46	66	59	71	58	61

(β') Δείγμα από *StaticRouting.csv*.

Time	N1	N2	N3	N4	N5	N6	N7	N8	N9
12:37:47	52	52	76	52	66	61	72	60	63
12:37:49	52	52	76	52	66	61	72	60	63
12:37:51	52	52	76	52	66	61	72	60	63

(β') Δείγμα από *AdjustableRouting.csv*.

Γ'.8.2 Μεθοδολογία Σύγκρισης

Για τη σύγκριση των σετ δεδομένων που προκύπτουν από την εκτέλεση του αλγορίθμου εξάγουμε κάποιες βασικές μετρήσεις, τη διάρκεια ζωής του δικτύου, τον πρώτο θάνατο κόμβου του δικτύου, το ρυθμό μετάδοσης δεδομένων προς τον sink, καθώς και πληροφορίες για τα μονοπάτια που ακολούθησε ο κάθε κόμβος. Έστερα, συγκρίνουμε τις μετρήσεις και καταλήγουμε σε κάποια συμπεράσματα.

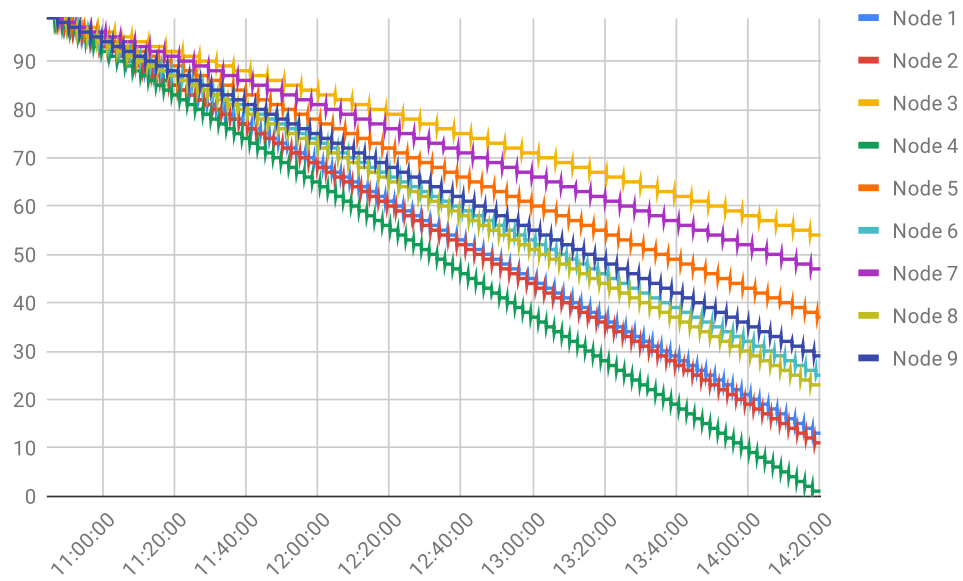
Κεφάλαιο Δ΄

ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ

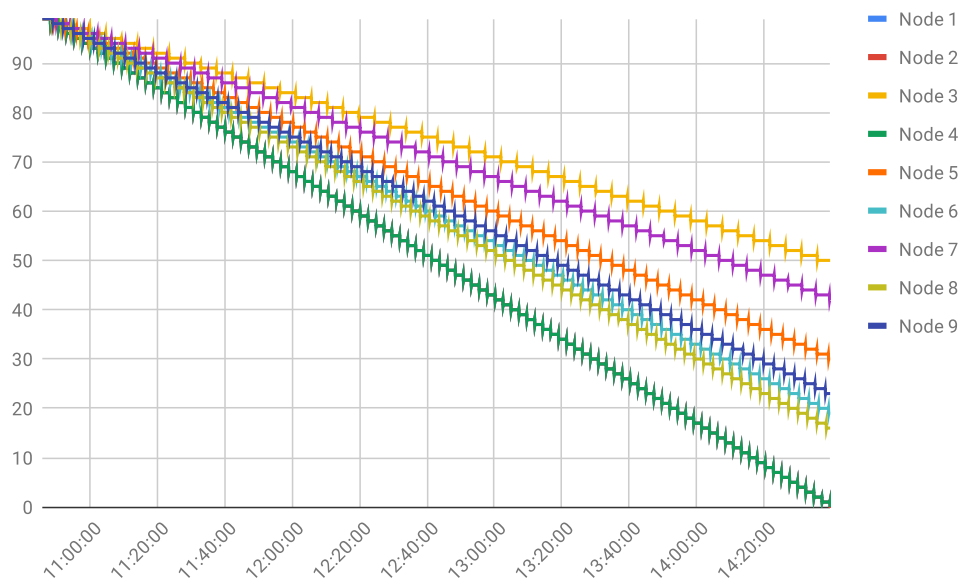
Η διαδικασία σύγκρισης των αποτελεσμάτων βασίζεται στην ύπαρξη καλά δομημένων δεδομένων. Εφόσον έχουμε τα αρχεία .csv και έχουμε αναλύσει την πληροφορία που μας δίνουν, καταλήγουμε σε κάποιες μετρήσεις οι οποίες περιγράφουν την απόδοση του δικτύου. Αφού παραθέσουμε τις μετρήσεις του δικτύου, θα τις συγκρίνουμε με τις μετρήσεις των προσομοιώσεων του πηγαιού άρθρου[15] ώστε να αξιολογήσουμε τον εξεταζόμενο αλγόριθμο. Στα επόμενα κεφάλαια αναλύεται ο τρόπος υπολογισμού των μετρήσεων και παρουσιάζονται τα αποτελέσματά τους.

Δ΄.1 Ενεργειακές Μετρήσεις του Δικτύου

Η υλοποίηση του προτεινόμενου αλγορίθμου δρομολόγησης δείχνει ότι υπάρχει προφανής συσχέτιση ανάμεσα στα αποτελέσματα των προσομοιώσεων και την πρακτική εφαρμογή του. Όπως φαίνεται στα διαγράμματα Δ΄.5 και Δ΄.6, χρησιμοποιώντας προσαρμοστική λογική στη δρομολόγηση των πακέτων, οι κόμβοι με το περισσότερο φόρτο (Node 1, Node 2 και Node 4) μοιράστηκαν πιο αποδοτικά τον μεγάλο όγκο πακέτων.



Σχήμα Δ'.5: Διάγραμμα Μπαταρίας/Χρόνου με Στατική Δρομολόγηση



Σχήμα Δ'.6: Διάγραμμα Μπαταρίας/Χρόνου με Προσαρμοστική Δρομολόγηση

Από τα διαγράμματα μπορούμε να εξάγουμε δύο πολύ σημαντικές πληροφορίες. Αρχικά, το

δίκτυο έζησε για είκοσι λεπτά περισσότερο όταν εκμεταλλεύτηκε τις μετρικές των path vulnerabilities. Επιπλέον, οι κόμβοι 1 και 2, μοιράστηκαν με πλήρη αποδοτικότητα το φόρτο τους με τον κόμβο 4, ο οποίος αποτελούσε τον πρώτο θάνατο του δικτύου στην εκτέλεση με στατική δρομολόγηση. Οι μπαταρίες τους εξαντλήθηκαν σχεδόν ταυτόχρονα, σε αντίθεση με τη διαφορά της τάξης του 10% που είχαν με τη στατική δρομολόγηση. Σύμφωνα με την τοπολογία του δικτύου (Σχήμα Γ'.4, σελίδα 21), παρατηρούμε ότι την ίδια στιγμή που ένας γειτονικός κόμβος του sink πεθαίνει, το ίδιο το δίκτυο πεθαίνει, διότι ένα μεγάλο ποσοστό των κόμβων δεν έχουν διαθέσιμο μονοπάτι προς τον sink. Με τη χρήση προσαρμοστικής δρομολόγησης όμως, οι κόμβοι επιλέγουν εναλλακτικά μονοπάτια και έτσι το δίκτυο πεθαίνει όταν τουλάχιστον δύο από τους τρεις γειτονικούς κόμβους του sink πεθάνουν.

Πολιτική Δρομολόγησης	Διάρκεια Ζωής Δικτύου
Προσαρμοστική	3:59:00
Στατική	3:40:00

Πίνακας Δ'.1: Διάρκεια ζωής του δικτύου (*Network Lifetime*).

Δ'.2 Διαμεταγωγή (Throughput) του Δικτύου

Για τον υπολογισμό του throughput του Δικτύου, υπολογίστηκε το άθροισμα των σειρών δεδομένων που προέκυψαν από τα RAW_Adjustable.csv και RAW_Static.csv αρχεία (Data Lines - DL), αφαιρώντας τις σειρές που περιείχαν τη συμβολοσειρά "DATA" διότι ήταν παραγόμενη από τον sink και δεν αποτελούσε μέρος των δεδομένων που μεταφέρθηκαν μέσω του δικτύου (False Lines - FL). Έπειτα το άθροισμα πολλαπλασιάστηκε με το μήκος της κάθε σειράς δεδομένων (Payload Length - PL) επί το μέγεθος της μεταβλητής χαρακτήρα (Char Size - CS) (1 char = 8 bits)¹ και στο τέλος διαιρέθηκε με τη συνολική διάρκεια ζωής του δικτύου (Life Time - LT) σε δευτερόλεπτα².

$$Throughput = \frac{(DL - FL) \times PL \times CS}{LT}$$

¹Πηγή: https://en.wikipedia.org/wiki/C_data_types

²Πηγή τρόπου υπολογισμού throughput: Jackson Lewis, How to Calculate Network Throughput, <https://www.techwalla.com/>

Πολιτική Δρομολόγησης	Διαμεταγωγή
Προσαρμοστική	624.61 bps ³
Στατική	629.04 bps

Πίνακας Δ'.2: Διαμεταγωγή του δικτύου (*Network Throughput*).

Παρατηρούμε ότι το throughput του δικτύου μειώνεται ελαφρώς με τη χρήση του εξεταζόμενου αλγορίθμου. Ο λόγος για τον οποίο υπάρχει αυτή η μείωση είναι επειδή κατά τη διάρκεια μετάδοσης δεδομένων, οι κόμβοι ανά τακτικά χρονικά διαστήματα ελέγχουν τις τιμές των path vulnerabilities για τους eligible γειτονικούς τους κόμβους. Η διαδικασία αυτή καταναλώνει πόρους του δικτύου και απασχολεί τις ζεύξεις των κόμβων με δεδομένα τα οποία δεν καταλήγουν στον sink και δεν είναι ωφέλιμα για την εκάστοτε εφαρμογή του δικτύου. Σε περίπτωση εφαρμογής χαμηλών απαιτήσεων ταχύτητας του δικτύου όμως, αυτή η διαφορά γίνεται πρακτικά αμελητέα.

Δ'.3 Συσχέτιση με το Πηγαίο Υλικό

Η προβλεπόμενη συμπεριφορά του δικτύου με τη χρήση του εξεταζόμενου αλγορίθμου σύμφωνα με τις προσομοιώσεις της βιβλιογραφίας, επιβεβαιώνεται και στην πράξη. Πιο συγκεκριμένα, οι προσομοιώσεις δείχνουν καλύτερη απόδοση στις ενεργειακές μετρήσεις όπως ο πρώτος θάνατος και η διάρκεια ζωής του δικτύου. Αυτό επιβεβαιώνεται και από τα αποτελέσματα της παρούσας εργασίας. Όσον αφορά την απόδοση του δικτύου σε μετρήσεις που αφορούν την ταχύτητα μετάδοσης και την διαμεταγωγή του δικτύου, ελέγχοντας τη διαμεταγωγή παρατηρείται ότι η απόδοση του δικτύου μειώνεται, γεγονός το οποίο επιβεβαιώνει ξανά τις προβλέψεις των προσομοιώσεων της βιβλιογραφίας.

³bps: bits per seconds

Κεφάλαιο Ε΄

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Σ Ε αυτή την εργασία, εξετάστηκε ένας αλγόριθμος δρομολόγησης ελαχιστοποίησης κατανάλωσης ενέργειας για ασύρματα δίκτυα αισθητήρων που βρίσκεται στη βιβλιογραφία[15]. Για τον έλεγχο του αλγορίθμου σε πραγματικό περιβάλλον χρήσης, χρησιμοποιήθηκαν μικροελεγκτές Arduino Uno R3, σε συνδυασμό με Wireless SD Shields και ασύρματες μονάδες επικοινωνίας Xbee S2C Pro. Το δίκτυο "έτρεξε" χρησιμοποιώντας στατική δρομολόγηση και στη συνέχεια με τη χρήση προσαρμοστικής δρομολόγησης σύμφωνα με τις προδιαγραφές του εξεταζόμενου αλγορίθμου. Τα δεδομένα αναλύθηκαν και έγινε σύγκριση με τα δεδομένα των προσομοιώσεων της βιβλιογραφίας. Στο τελευταίο κεφάλαιο θα γίνει μια συνοπτική ανάλυση των αποτελεσμάτων της εργασίας καθώς και πιθανές προτάσεις για μελλοντική έρευνα.

Ε΄.1 Συμπεράσματα και Παρατηρήσεις

Από την ανάλυση των δεδομένων που παράγει η εκτέλεση των αλγορίθμων συμπεραίνουμε ότι υπάρχει σίγουρη συσχέτιση ανάμεσα στις προβλέψεις των προσομοιώσεων και τα αποτελέσματα της εκτέλεσης σε πραγματικό περιβάλλον χρήσης. Η διάρκεια ζωής του δικτύου αυξήθηκε, καθώς και η χρονική στιγμή του πρώτου θανάτου του δικτύου. Επιπλέον, η απόδοση του δικτύου μειώθηκε κατά παρόμοια κλίμακα στο κομμάτι της μετάδοσης δεδομένων, και συγκεκριμένα στην τιμή του throughput του δικτύου. Αυτά τα αποτελέσματα συμβαδίζουν με

την υπάρχουσα βιβλιογραφία. Η χρησιμότητα του αλγορίθμου μεγιστοποιείται όταν ζητείται η υλοποίηση ασύρματων δικτύων αισθητήρων με ιδιαίτερα υψηλές ενεργειακές απαιτήσεις και ταυτόχρονα χαμηλές απαιτήσεις μετάδοσης δεδομένων ή όταν το δίκτυο στο οποίο εφαρμόζεται απειλείται από το φαινόμενο της ενεργειακής τρύπας. Παρατηρείται ωστόσο, ότι ο αλγόριθμος φαίνεται να σπαταλάει αρκετή ενέργεια και αποθηκευτικό χώρο για τον υπολογισμό των path vulnerabilities, χαρακτηριστικό το οποίο σε δίκτυα μεγαλύτερης κλίμακας πιθανόν να μειώσει περαιτέρω την απόδοσή του.

Ε'.2 Μελλοντική Έρευνα

Πιθανά σημεία ενδιαφέροντος για μελλοντική έρευνα είναι η επεκτασιμότητα του αλγορίθμου (scalability), δηλαδή, η διερεύνηση του μεγέθους του δικτύου για το οποίο, το ενεργειακό κόστος του υπολογισμού των path vulnerabilities εξισορροπείται από την ενέργεια που εξοικονομεί το δίκτυο χάρη σε αυτόν τον υπολογισμό. Σε αυτό το θεωρητικό σημείο, η διάρκεια ζωής του δικτύου πλέον παύει να επηρεάζεται από τον εξεταζόμενο αλγόριθμο. Ένας επιπλέον παράγοντας που φέρει ενδιαφέρον έρευνας είναι η θέση του sink κόμβου στο δίκτυο και ο βαθμός με τον οποίο η τοποθέτησή του επηρεάζει τις αποδόσεις του αλγορίθμου. Επιπρόσθετα, όσον αφορά το υλικό κομμάτι της εργασίας, θα μπορούσαν να εξεταστούν άλλες επιλογές για τη δημιουργία των ασύρματων κόμβων αισθητήρων καθώς και η εξέταση του αλγορίθμου σε κόμβους οι οποίοι χρησιμοποιούν μπαταρίες σαν συμπληρωματικές πηγές τροφοδοσίας, με ανανεώσιμες πηγές ενέργειας ως κύρια τροφοδοσία.

Βιβλιογραφία

- [1] C.S. Raghavendra, K.M. Sivalingam, and T. Znati. *Wireless Sensor Networks*. English. Ercoftac Series. Springer US, 2006. ISBN: 9781402078842. URL: <https://books.google.gr/books?id=4zyDBwAAQBAJ>.
- [2] Dionisis Kandris et al. “Applications of Wireless Sensor Networks: An Up-to-Date Survey”. English. In: *Applied System Innovation* 3.1 (2020). ISSN: 2571-5577. DOI: 10.3390/asi3010014. URL: <https://www.mdpi.com/2571-5577/3/1/14>.
- [3] J.N. Al-Karaki and A.E. Kamal. “Routing techniques in wireless sensor networks: a survey”. English. In: *IEEE Wireless Communications* 11.6 (2004), pp. 6–28. DOI: 10.1109/MWC.2004.1368893.
- [4] Amit Sarkar and T. Senthil Murugan. “Routing protocols for wireless sensor networks: What the literature says?” English. In: *Alexandria Engineering Journal* 55.4 (2016), pp. 3173–3183. ISSN: 1110-0168. DOI: <https://doi.org/10.1016/j.aej.2016.08.003>. URL: <https://www.sciencedirect.com/science/article/pii/S1110016816302125>.
- [5] Dipak K. Sah and Tarachand Amgoth. “Renewable energy harvesting schemes in wireless sensor networks: A Survey”. English. In: *Information Fusion* 63 (2020), pp. 223–247. ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2020.07.005>. URL: <https://www.sciencedirect.com/science/article/pii/S156625352030316X>.
- [6] Τσουμάνης Γεώργιος. ‘Βελτιστοποίηση Κατανάλωσης Ενέργειας σε Ασύρματα Δίκτυα Αισθητήρων’. Διδακτορική διατρ. Ιόνιο Πανεπιστήμιο, 2018.
- [7] M.A. Matin and M.M. Islam. “Overview of Wireless Sensor Network”. English. In: *Wireless Sensor Networks*. Ed. by Mohammad A. Matin. Rijeka: IntechOpen, 2012. Chap. 1. DOI: 10.5772/49376. URL: <https://doi.org/10.5772/49376>.

- [8] T. Ozaki, Jaime Bae Kim, and T. Suda. “Bandwidth-efficient multicast routing for multihop, ad-hoc wireless networks”. English. In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*. Vol. 2. 2001, 1182–1191 vol.2. DOI: 10.1109/INFCOM.2001.916313.
- [9] Yusuf Abdullahi Badamasi. “The working principle of an Arduino”. English. In: *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*. 2014, pp. 1–4. DOI: 10.1109/ICECCO.2014.6997578.
- [10] David Mellis et al. “Arduino: An open electronic prototyping platform”. English. In: *Proc. Chi*. Vol. 2007. 2007, pp. 1–11.
- [11] Jay P Sipani et al. “Wireless Sensor Network for Monitoring & Control of Environmental Factors using Arduino.” English. In: *International Journal of Interactive Mobile Technologies* 12.2 (2018).
- [12] Alejandro Cama-Pinto et al. “Received strength signal intensity performance analysis in wireless sensor network using Arduino platform and XBee wireless modules”. English. In: *International Journal of Distributed Sensor Networks* 13.7 (2017), p. 1550147717722691. DOI: 10.1177/1550147717722691. eprint: <https://doi.org/10.1177/1550147717722691>. URL: <https://doi.org/10.1177/1550147717722691>.
- [13] Robert Faludi. *Building wireless sensor networks: with ZigBee, XBee, arduino, and processing*. English. ” O’Reilly Media, Inc.”, 2010.
- [14] Mudasar Basha et al. “Approach to Find Shortest Path Using Ant Colony Algorithm”. English. In: *Intelligent Communication, Control and Devices*. Springer, 2018, pp. 1243–1252.
- [15] Apostolos Demertzis, Konstantinos Oikonomou, and Ioannis Stavrakakis. “An Adjustable Forwarding Policy Exploiting Path Vulnerability in Wireless Sensor Networks”. English. In: *Second International Balkan Conference on Communications and Networking 2018 (BalkanCom’18)*. 2018.

Συντμήσεις

RSSI	Received Signal Strength Indicator
WSN	Wireless Sensor Network
Li-Po	Lithium Polymer
RF	Radio Frequency
USB	Universal Serial Bus
IDE	Integrated Development Environment
CSV	Comma Separated Values
PWM	Pulse-Width Modulation
IEEE	Institute of Electrical and Electronics Engineers
BSD	Berkeley Software Distribution
ACK	Acknowledgement

Ευρετήριο

Arduino Wireless SD Shield, 11

First death time, 36

Path Vulnerability, 3, 35, 44

XBee RF Module, 4, 10

Αλγόριθμος Bellman - Ford, 23, 40

Διαμεταγωγή - Throughput, 53

Κόμβοι Αισθητήρων, 18, 33

Κόμβος Sink, 17, 29

Μικροελεγκτής Arduino, 4, 8

Τροφοδοσία Δικτύου, 12