

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τμήμα Πληροφορικής



Πτυχιακή Εργασία

Αλγόριθμοι Συγχρονισμού και Απόκλιση Ρολογιών
σε Ασύρματα Δίκτυα Αισθητήρων

Νέμανια Γιέβτις Π2017182

Επιβλέπων: Καθηγητής Κωνσταντίνος Οικονόμου

16 Σεπτεμβρίου 2021

Επιβλέπων Καθηγητής

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Χριστόφορος Νταντογιάν, Επίκουρος Καθηγητής,
Ιόνιο Πανεπιστήμιο

Στην μητέρα.

Περίληψη

Τα ασύρματα δίκτυα αισθητήρων (wireless sensor networks, WSN) αποτελούν θερμό πεδίο έρευνας ήδη αρκετά χρόνια. Γεγονός στο οποίο συνέβαλαν η ανάπτυξη της τεχνολογίας παραγωγής μικροεπεξεργαστών και αισθητήρων, με αποτέλεσμα να καθίστανται δυνατές πληθώρα καινούριων εφαρμογών πραγματικού κόσμου. Τα WSN υπόσχονται εφαρμογές που θα βελτιώσουν τις παραγωγές αγροτικών καλλιεργειών, που θα συμβάλουν στην μείωση της ανθρώπινης επίπτωσης στην κλιματική αλλαγή, που θα κάνουν την έρευνα σε άλλα επιστημονικά πεδία ευκολότερη. Τα πρακτικά προβλήματα των εφαρμογών αυτών - μεταξύ των οποίων είναι και το θέμα του συγχρονισμού στα WSN - βρίσκονται μάλλον μακριά από την τελική λύση. Στην παρούσα εργασία, λοιπόν, θα εξεταστεί το θέμα του συγχρονισμού στα WSN, και τις απόκλισης των ρολογιών των κόμβων. Στην συνέχεια, θα γίνει αναφορά σε συγκεκριμένους γνωστούς αλγορίθμους συγχρονισμού και θα παρουσιαστεί η υλοποίηση ενός εκ αυτών που έγινε στα πλαίσια αυτής της προσπάθειας. Τέλος, θα γίνει αναφορά στα αποτελέσματα των πειραμάτων που διεξάχθηκαν και στην συσχέτισή τους με την υπάρχουσα βιβλιογραφία.

Abstract

The wireless sensor networks (WSN) have for a time now been receiving an sizable amount of interest of the academic community. This can be attributed to the rapid advancement of the microchip and sensor technologies, which made possible a vide new array of real-world applications. The WSN are opening the road for applications that will help farmers increase the quality of their produce, that will help with mitigating the impact of human activities on global warming, and that will make the research in other scientific areas easier, amongst other things. However, the practical problems related to the deployment of the WSNs - synchronization problem included - are far from solved for good. This thesis aims to investigate the issue of synchronization in WSNs and the issue of clock skews. Next, some of the most widely discussed synchronization algorithms, as well as the implementation of one of them will be presented. Finally, the results of the experiments conducted will be presented, confirming the established knowledge regarding the relative clock drifts.

Ευχαριστίες

Κανένας τίποτα δεν φτιάχνει μόνος του. Έτσι κι εγώ οφείλω να ευχαριστήσω με την σειρά μου όσους με βοήθησαν στην συγγραφή αυτής της εργασίας. Αρχικά θέλω να ευχαριστήσω τον Καθηγητή Κωνσταντίνο Οικονόμου, για την εμπιστοσύνη και την πολύτιμη βοήθειά του καθ' όλη την διάρκεια αυτής της προσπάθειας. Επίσης θέλω να ευχαριστήσω τον Σπύρο Βεργή και τον Γιώργο Σιδερά, για την βοήθεια τους στο τεχνικό κομμάτι της εργασίας. Ευχαριστώ και την μητέρα μου και την αδερφή μου, για την τεράστια υποστήριξη, αγάπη, και δύναμη που μου δίνουν συνεχώς. Τέλος, οφείλω να ευχαριστήσω και τους φίλους μου, που ήταν πάντοτε δίπλα μου, και δίχως οποίους δεν θα πολυείχε τίποτα νόημα.

Περιεχόμενα

A'	Εισαγωγικά Στοιχεία	1
A'.1	Εισαγωγή	1
A'.2	Συγχρονισμός στα WSN	2
A'.3	Ρολόγια - ακρίβεια και αποκλίσεις	3
A'.4	Συνεισφορά και αποτελέσματα	5
B'	Ο Αλγόριθμος Συγχρονισμού Δεδομένων	6
B'.1	Βασική ιδέα	6
B'.1.1	Ο αλγόριθμος	7
B'.2	Σύγκριση με άλλους αλγορίθμους	10
B'.2.1	TPSN	10
B'.2.2	RBS	10
Γ'	Η Τλοποίηση του Αλγορίθμου	12
Γ'.1	Περιβάλλον υλοποίησης	12
Γ'.1.1	Το ZigBee πρωτόκολλο	12
Γ'.1.2	Οι κόμβοι Arduino	14
Γ'.2	Λογισμικό της εφαρμογής	15
Γ'.2.1	Λειτουργία της εφαρμογής	15
Γ'.3	Μορφή του πακέτου	17
Γ'.3.1	Το πακέτο των δεδομένων	17
Γ'.3.2	Τα υπόλοιπα πακέτα	18
Γ'.4	Απόκλιση από τον αλγόριθμο αναφοράς	19
Γ'.5	Μετρήσεις : Αποκλίσεις ρολογιών σε ιδανικές συνθήκες	19

Περιεχόμενα	v
Δ' Σύνοψη και Συμπεράσματα	21
Δ'.1 Σύνοψη	21
Δ'.2 Συμπεράσματα	21
Βιβλιογραφία	22
Παραρτήματα	30
I Κώδικες	31
I.1 Κύριος κώδικας κόμβων	31
I.2 Κλάση Messenger	35
I.3 Κλάση MessageList	43
I.4 Κώδικας για την καταγραφή αποκλίσεων ρολογιών σε σχέση με ακριβές ρολόι	49

Κατάλογος Σχημάτων

A'.1	Σχετικές αποκλίσεις από τον πραγματικό χρόνο	4
B'.1	Διαδρομή μηνύματος & σχετικές καθυστερήσεις	7
B'.2	Πεδία μηνύματος	8
Γ'.1	Ένας κόμβος Arduino	14
Γ'.2	Καταστάσεις λειτουργίας κόμβων	16
Γ'.3	Μορφή πακέτου δεδομένων υλοποίησης	17
Γ'.4	Αποκλίσεις ρολογιών σε σχέση με τον coordinator	20

Κεφάλαιο Α΄

Εισαγωγικά Στοιχεία

Σ Ε αυτό το κεφάλαιο θα γίνει μία αρχική περιγραφή των ασύρματων δικτύων αισθητήρων και των εφαρμογών τους, τις προκλήσεις που υπάρχουν για την υλοποίηση αλγορίθμων συγχρονισμού στα ασύρματα δίκτυα αισθητήρων, και τέλος θα γίνει μια αναφορά στην λογική της λειτουργίας των ρολογιών και των προβλημάτων που μπορούν να προκαλέσουν όσο αφορά τον συγχρονισμό των κόμβων.

Α΄.1 Εισαγωγή

Τις τελευταίες δύο περίπου δεκαετίες έχει γίνει μεγάλος όγκος έρευνας πάνω στο πεδίο των ασύρματων δικτύων αισθητήρων (wireless sensor networks, WSNs). Τα τελευταία χρόνια ωστόσο το πεδίο έχει εξελιχθεί από ένα πλήρως ερευνητικού ενδιαφέροντος σε ένα πεδίο με ποικίλες σημαντικές εφαρμογές [1, 2, 3]. Η βασική ιδέα στο πεδίο των WSN είναι η ανάπτυξη καταναεμημένων δικτυοκεντρικών συστημάτων συνήθως χαμηλού κόστους και πολύ συγκεκριμένου σκοπού. Η ουσία των κόμβων αυτών των συστημάτων βρίσκεται στους αισθητήρες που κατέχουν οι κόμβοι, τα δεδομένα των οποίων και η γνώση που μπορεί να εξαχθεί από αυτά είναι συνήθως το κίνητρο για ανάπτυξη τέτοιων συστημάτων. Τα κύρια πεδία εφαρμογών είναι αυτά της ακριβούς γεωργίας (precision agriculture), της παρακολούθησης περιβάλλοντος (enviromental monitoring), των έξυπνων πόλεων (smart cities) και το πεδίο των βιομηχανικών εφαρμογών.

Στο πεδίο της έξυπνης γεωργίας ο σκοπός είναι η συλλογή των δεδομένων σχετικών με τις καλλιέργειες όπως η υγρασία εδάφους, θερμοκρασία, εικόνα, με στόχο

την παρακολούθηση μικροκλίματος, την καταπολέμηση εχθρικών εντόμων [4, 5], την ανάπτυξη έξυπνων συστημάτων ποτίσματος [6, 7]. Πέραν από τα άμεσα κέρδη που υπάρχουν για τους ίδιους τους αγρότες από υλοποιήσεις τέτοιων συστημάτων, υπάρχει και η όψη της προώθησης της περιβαλλοντικά - έξυπνης γεωργίας (climate-smart agriculture) [8], με σκοπό την μείωση της επίπτωσης του τομέα της γεωργίας [9] στην κλιματική αλλαγή.

Στο πεδίο της παρακολούθησης περιβάλλοντος βρίσκουμε εφαρμογές όπως αυτές για αποτροπή και παρακολούθηση δασικών πυρκαγιών [10] ή για παρακολούθηση ποιότητας νερού και συνθηκών θαλάσσιων περιοχών [11], και αρκετά άλλα.

Οι εφαρμογές στον τομέα των έξυπνων πόλεων θα μπορούσαν να χωριστούν σε αυτές που προορίζονται να χρησιμοποιηθούν από κάποιο μεμονωμένο νοικοκυριό - έναν ιδιώτη, και εφαρμογές που αφορούν και προορίζονται για ένα μεγαλύτερο σύνολο άγνωστων ανθρώπων. Πολλές από τις εφαρμογές που αφορούν τα έξυπνα σπίτια [12] έχουν σκοπό την διευκόλυνση της ζωής των ανθρώπων που ζουν σε αυτό, ενώ οι εφαρμογές που αφορούν τα έξυπνα αυτοκίνητα [13, 14] αρκετές φορές έχουν σαν σκοπό την βελτίωση ζωής μεγαλύτερου συνόλου ανθρώπων. Επειδή ακριβώς το φάσμα εφαρμογών που αφορούν την ζωή σε μία πόλη είναι τεράστιο, το ερευνητικό ενδιαφέρον που υπάρχει γύρω από το θέμα είναι ανάλογο [15].

Στον τομέα της βιομηχανίας, τα WSN χρησιμοποιούνται συνήθως για την συλλογή κρίσιμων δεδομένων όπως είναι η δόνηση, η θερμοκρασία και η πίεση κυρίως για την εξασφάλιση της ποιότητας υπηρεσιών (quality of service, QoS) [16], αλλά και για την παρακολούθηση της κατάστασης των ίδιων των μηχανημάτων [17].

Α΄.2 Συγχρονισμός στα WSN

Στις εφαρμογές WSN είναι αρκετά συνηθισμένη η ανάγκη συγχρονισμού των ρολογιών των κόμβων του συστήματος. Αυτό συμβαίνει είτε για να επιτευχθεί η λήψη αποφάσεων σε συγκεκριμένες χρονικές στιγμές, είτε για να εγγυάται η έγκυρη συσχέτιση δεδομένων - χρόνου [18].

Τα συγκεκριμένα προβλήματα που αφορούν τον συγχρονισμό στα WSN, είναι οι κακές συνθήκες περιβάλλοντος στις οποίες βρίσκονται οι κόμβοι, το ζήτημα του κόστους του κάθε μεμονωμένου κόμβου, καθώς και το ζήτημα της ενεργειακής τροφοδότησης των κόμβων. Το κάθε ένα από αυτά τα θέματα της υλοποίησης των συστημάτων φέρνει και ένα πρόβλημα στον χώρο του συγχρονισμού.

Το γεγονός που οι κόμβοι διαθέτουν πεπερασμένη ενέργεια σημαίνει πως το κάθε πακέτο που κυκλοφορεί έχει κατά πολύ αυξημένο κόστος σε σχέση με ένα πακέτο σε ένα συμβατικό δίκτυο, δεδομένου πως η χρήση του πομποδέκτη αποτελεί τον μεγαλύτερο παράγοντα κατανάλωσης ενέργειας στους κόμβους [19]. Πράγμα που σημαίνει πως στον προγραμματισμό των εφαρμογών WSN, θα πρέπει να δοθεί μέγιστη δυνατή προσοχή για να επιτευχθεί η μικρότερη δυνατή σπατάλη μηνυμάτων - συμπεριλαμβανομένων των μηνυμάτων που αφορούν τον συγχρονισμό των κόμβων. Το απαιτούμενο χαμηλό κόστος των κόμβων σημαίνει πως δεν είναι αποδεκτός ο εξοπλισμός των κόμβων με επιπλέον εξαρτήματα όπως ακριβέστερα ρολόγια ή άλλο εξοπλισμό ειδικά για τον σκοπό του συγχρονισμού. Οι κόμβοι των εφαρμογών WSN είναι πολύ συχνά εκτεθειμένοι σε κακές περιβαλλοντικές συνθήκες όπως πολύ υψηλές ή πολύ χαμηλές θερμοκρασίες, υψηλή υγρασία και λοιπά. Αυτοί οι παράγοντες - όπως θα συζητηθεί και στο Α'.3 - παίζουν σημαντικό ρόλο στην ακρίβεια που θα έχουν τα ρολόγια των κόμβων, με αποτέλεσμα τα τυχόν πρωτόκολλα συγχρονισμού να πρέπει πιθανόν να λάβουν υπόψιν και ειδικότερα αυτούς τους παράγοντες.

Ένας βασικός διαχωρισμός που πρέπει να γίνει σχετικά με τις εφαρμογές WSN και τον συγχρονισμό είναι στις εφαρμογές που έχουν πρόσβαση στο διαδίκτυο και αυτές που δεν έχουν. Οι εφαρμογές με κόμβους που έχουν άμεση πρόσβαση στο διαδίκτυο και κάνουν χρήση αυτού σε κάθε περίπτωση μπορούν απλώς να συγχρονιστούν με το NTP. Όμοια και οι κόμβοι που για κάποιον λόγο κατέχουν GPS συσκευές. Σε κάθε άλλη περίπτωση η ανάγκη για την ανάπτυξη εξειδικευμένων αλγορίθμων συγχρονισμού για τα WSN παραμένει [20].

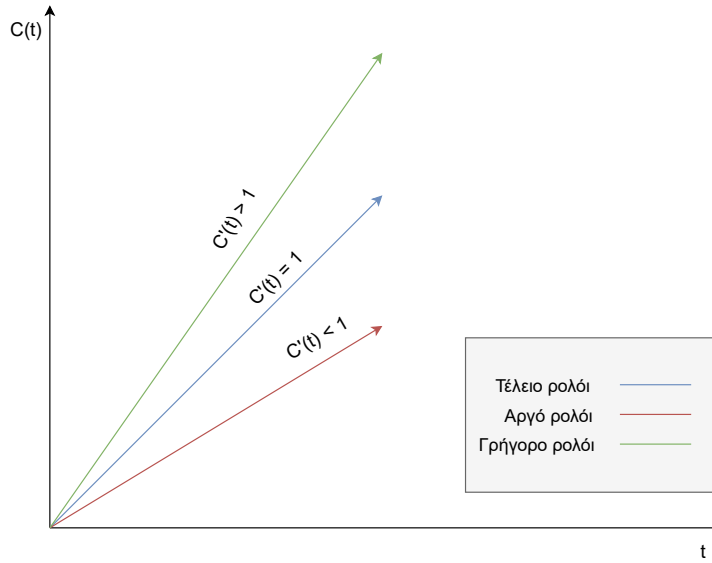
Α'.3 Ρολόγια - ακρίβεια και αποκλίσεις

Η πηγή της μέτρησης χρόνου σε έναν WSN κόμβο είναι κατά κανόνα ένας ενσωματωμένος μετρητής που ανανεώνονται με μία συχνότητα f από έναν κρυσταλλικό ταλαντωτή που βρίσκεται στο κύκλωμα [21]. Οι ταλαντωτές αυτοί ωστόσο, λόγω ατέλειας κατά την παραγωγή, είναι αδύνατο να διατηρούν συνεχώς σταθερή αυτήν την συχνότητα. Αυτή η απόκλιση από την αναμενόμενη συχνότητα f ονομάζεται σχετική απόκλιση συχνότητας (relative frequency drift), συμβολίζεται με δf , και μετριέται σε ppm - parts per million (μέρη ανά εκατομμύριο).

Υπολογίζεται ως :

$$\delta f = \frac{f - F_0}{F_0} \quad (A'.1)$$

Όπου F_0 είναι η πραγματική στιγμιαία συχνότητα του ταλαντωτή. Οι κύριοι παράγοντες που επηρεάζουν την απόκλιση συχνότητας ρολογιού είναι οι ατέλειες κατά την παραγωγή, η θερμοκρασία και γήρανση [21]. Στο σχήμα Α'.1 φαίνεται το αποτέλεσμα των διαφορετικών αποκλίσεων συχνότητας σε δύο κόμβους, που αποτελεί και το κύριο πρόβλημα που καλούνται να λύσουν οι αλγόριθμοι συγχρονισμού. Αν πούμε πως $C_A(t) = t_A$ και $C_B(t) = t_B$ είναι η τιμή των ρολογιών κόμβων Α και Β την στιγμή t , τότε η διαφορά τους, $C_A(t) - C_B(t)$, είναι η σχετική απόκλιση των δύο ρολογιών (clock drift). Όταν, από την άλλη έχουμε διαφορετικές αποκλίσεις συχνότητας ρολογιών τότε εμφανίζεται το φαινόμενο της λόξωσης (clock skew) - δηλαδή τα δύο ρολόγια να προχωράνε σε διαφορετικούς ρυθμούς, το ένα πιο γρήγορο από το άλλο.



Σχήμα Α'.1: Σχετικές αποκλίσεις από τον πραγματικό χρόνο

Σε γενικές γραμμές, είναι δυνατό το κούρδισμα (calibration) των ρολογιών ώστε να απαλειφθεί η αρχική απόκλιση συχνότητας. Ωστόσο, δεδομένου πως - όπως σημειώθηκε - ούτε η απόκλιση συχνότητας είναι σταθερή σε έναν ταλαντωτή, και εξαρτάται

και από εξωτερικούς παράγοντες, η διαδικασία αυτή μάλλον κρίνεται ανούσια στην γενική περίπτωση. Από την άλλη, υπάρχουν προτάσεις για πιο περίπλοκες προσεγγίσεις για το δυναμικό κούρδισμα ρολογιών κατά την διάρκεια της διαδικασίας συγχρονισμού [22].

Όπως θα φανεί και στην συνέχεια, η λόξωση των ρολογιών είναι ουσιαστική και μετρήσιμη, ακόμα και σε διαστήματα λεπτών [23]. Είναι φανερό πως στις εφαρμογές που ο συντονισμός των ρολογιών των κόμβων είναι κρίσιμος αποκλίσεις τέτοιου τύπου δεν είναι αποδεκτές, ειδικά αν ληφθεί υπόψιν πως οι αποκλίσεις αυτές με την πάροδο του χρόνου απλώς αυξάνονται, πράγμα που μπορεί να καταστήσει τις ίδιες τις εφαρμογές μη λειτουργικές.

Α΄.4 Συνεισφορά και αποτελέσματα

Επίκεντρο της προσπάθειας αυτής της εργασίας ήταν η ανάπτυξη εφαρμογής WSN, με την χρήση του αλγορίθμου συγχρονισμού δεδομένων που περιγράφεται στο Β΄.1, και έχει οριστεί στο [24]. Με την ολοκλήρωση αυτής της εργασίας από το πειραματικό μέρος της προκύπτει μια επιβεβαίωση της υπάρχουσας βιβλιογραφίας σχετικά με τις αποκλίσεις των ρολογιών στους κόμβους WSN, που παρουσιάζεται στο Δ΄1. Πέραν απ' αυτό, η ανάπτυξη των κλάσεων MessageList και Messenger αποτελούν μια καλή βάση για την πιθανή μελλοντική επέκταση των λειτουργιών της βιβλιοθήκης 'ξβεε-αρδουινο' [25] που χρησιμοποιήθηκε. Επίσης, σε πλαίσιο μαθήματος προγράμματος σπουδών έχει αναπτυχθεί ένα πρωτότυπο εφαρμογής πραγματικού κόσμου [26], βασισμένη στις ιδέες της παρούσης εργασίας. Τέλος, ο κώδικας και η επεξήγηση της λειτουργίας της εφαρμογής της παρούσης εργασίας μπορεί να αποτελέσει σημείο αναφοράς για όποιον ενδιαφέρεται να αναπτύξει ένα πρώτο WSN.

Κεφάλαιο Β΄

Ο Αλγόριθμος Συγχρονισμού Δεδομένων

ΒΑΣΗ για την προσπάθεια της παρούσας εργασίας αποτέλεσε ο αλγόριθμος συγχρονισμού δεδομένων, που περιγράφεται στην [24]. Σε αυτό το κεφάλαιο θα περιγραφεί η βασική ιδέα του αλγορίθμου, θα γίνει μια σύγκριση με άλλους γνωστούς αλγορίθμους συγχρονισμού και τέλος θα αιτιολογηθεί η επιλογή του συγκεκριμένου αλγορίθμου για την υλοποίηση της εργασίας.

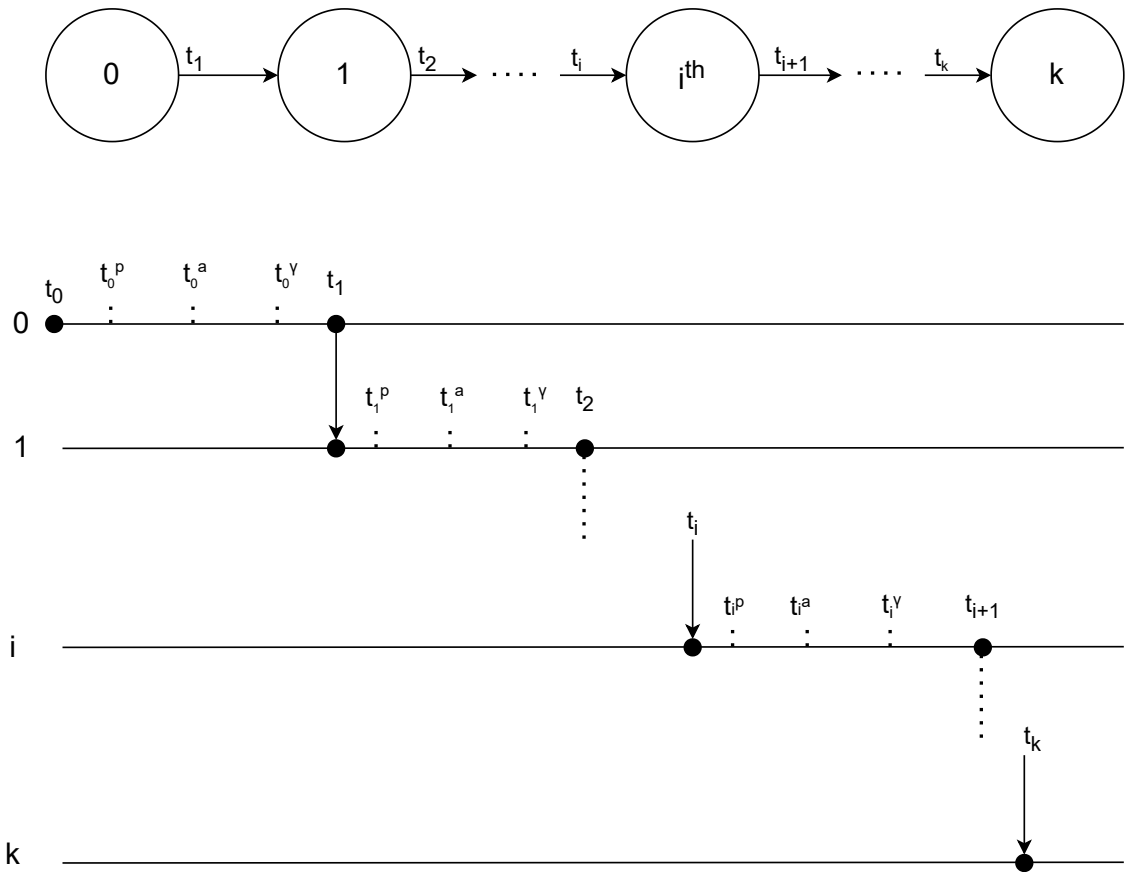
Β΄.1 Βασική ιδέα

Η βασική ιδέα του αλγορίθμου είναι να αντικαταστήσει την προσπάθεια συγχρονισμού του κάθε επιμέρους ρολογιού των κόμβων του δικτύου με μία μέθοδο υπολογισμού διαφοράς ρολογιών βάση της οποίας εν τέλει θα είναι δυνατός ο προσδιορισμός της χρονικής στιγμής που λήφθηκαν τα δεδομένα.

Οι ουσιαστικές περιπτώσεις στις οποίες θα ήταν χρήσιμο να υλοποιηθεί ο αλγόριθμος είναι εφαρμογές IoT - όπως εφαρμογές παρακολούθησης γεωργικών καλλιεργειών, συστήματα δασοπροστασίας, συστήματα έξυπνων σπιτιών, και γενικότερα περιπτώσεις εφαρμογών των οποίων ο σκοπός είναι συλλογή δεδομένων από ένα δίκτυο αισθητήρων και χρήζουν της χρονικής παραμέτρου των δεδομένων που συλλέγονται αλλά ωστόσο δεν ωφελούνται από το να έχουν συγχρονισμένα τα ίδια τα ρολόγια των συσκευών.

B'.1.1 Ο αλγόριθμος

Ας υποθέσουμε ότι έχουμε έναν κόμβο σε ένα WSN. Αυτός ο κόμβος ανά χρονικά διαστήματα - είτε μετά από προτροπή άλλου κόμβου, είτε από δική του πρωτοβουλία - θέλει να στείλει κάποια δεδομένα τα οποία έχει παραγάγει ο ίδιος σε έναν άλλο κόμβο του δικτύου k hops μακριά. Στην κάθε προώθηση του μηνύματος, όπως φαίνεται και στο Σχήμα B'.1, οι καθυστερήσεις είναι ο χρόνος επεξεργασίας μηνύματος t^p , ο χρόνος που απαιτείται μέχρι η συσκευή να ανιχνεύσει ελεύθερο κανάλι επικοινωνίας t^a , και ο χρόνος μετάδοσης μηνύματος t^r . Ο χρόνος διάδοσης μηνύματος θεωρείται μηδαμινός και η συνολική καθυστέρηση τ θεωρείται ίση σε όλους τους κόμβους του δικτύου.



Σχήμα B'.1: Διαδρομή μηνύματος & σχετικές καθυστερήσεις

Συνεπώς για να μπορέσει ο τελικός κόμβος-παραλήπτης να έχει την γνώση της χρονικής στιγμής που παράχθηκαν τα δεδομένα αρκεί να γνωρίζει την διαφορά της τιμής του ρολογιού του και του ρολογιού του κόμβου-παραγωγού. Από το διάγραμμα

προκύπτει πως $\tau = t_{i+1} - t_i$ και $t_i = t_0 + \tau i$.

Ο αλγόριθμος εκμεταλλεύεται αυτήν την ιδιότητα του συστήματος διαχωρίζοντας την αποστολή του πακέτου σε δύο μέρη - το πρώτο στο οποίο θα σταλούν τα preamble bits και το SOF και το οποίο θα βοηθήσει στην συνέχεια τον παραλήπτη να υπολογίσει την διαφορά των ρολογιών, και το δεύτερο στο οποίο αποστέλλονται τα δεδομένα καθώς και η χρονοσφραγίδα της στιγμής της αποστολής.

Preamble	SOF	$c_i(t_{i+1})$	TS_i	Data
----------	-----	----------------	--------	------

Σχήμα Β'.2: Πεδία μηνύματος

Αναλυτικότερα, τα βήματα του αλγορίθμου είναι ως έχουν:

- Αν ο κόμβος διαθέτει καινούρια δεδομένα και δεν είναι ο κύριος κόμβος τότε κρατάει τον χρόνο παραγωγής δεδομένων και μπαίνει στην κατάσταση αποστολής
- Αν ο κόμβος έχει λάβει preamble bits μπαίνει στην κατάσταση λήψης
- Στην κατάσταση αποστολής ο κόμβος στέλνει τα preamble bits και το SOF στον επόμενο κόμβο της διαδρομής και αμέσως κρατάει την τιμή του ρολογιού του. Στην συνέχεια στέλνει το υπόλοιπο του μηνύματος που είναι η τιμή ρολογιού που μόλις κράτησε, η χρονοσφραγίδα παραγωγής των δεδομένων, και τα ίδια τα δεδομένα.
- Στην κατάσταση λήψης ο κόμβος λαμβάνει το SOF και αμέσως μετά κρατάει την τιμή του ρολογιού του. Έπειτα λαμβάνει το υπόλοιπο του πακέτου και μπορεί πλέον να υπολογίσει την διαφορά των ρολογιών (γραμμή 22) και να προσδιορίσει την χρονική στιγμή παραγωγής δεδομένων (γραμμή 23). Τέλος, στην περίπτωση που ο κόμβος δεν είναι ο κύριος κόμβος, ξαναμπαίνει στην κατάσταση αποστολής για να προωθήσει τα δεδομένα με τον ίδιο τρόπο στον επόμενο κόμβο της διαδρομής.

Με αυτόν τον τρόπο ο αλγόριθμος καταφέρνει πολύ κομψά να εκμεταλλευτεί απαλειφθεί από την χρήση πολλαπλών μηνυμάτων που χρησιμοποιούνται μόνο για τους σκοπούς του συγχρονισμού. Βέβαια, όπως θα φανεί στο επόμενο κεφάλαιο, η ακριβής

Algorithm 1: Ο αλγόριθμος συγχρονισμού δεδομένων

```

1 Var  $k$ : Κύριος κόμβος,  $i$ : Κόμβος αναφοράς,  $Data$ : Δεδομένα κόμβου
   αναφοράς,  $TS$ : Πεδίο χρονοσφραγίδας,  $Clock$ : Τοπικός χρόνος,  $Preamble$ :
   Αρχικά bits
2 Loop:
3   if  $Data$  then:
4      $TS := Clock$ 
5     if  $i \neq k$  then SendPacket( $TS$ ,  $Data$ )
6   if receive <Preamble> then ReceivePacket
7   function SendPacket( $TS$ ,  $data$ )
8   Var  $c$ ,  $SOF$ : Αρχή του πακέτου
9     send <Preamble,  $i+1$ >
10    send <SOF,  $i+1$ >
11     $c := Clock$ 
12    send < $c$ ,  $i+1$ >
13    send < $TS$ ,  $i+1$ >
14    send < $data$ ,  $i+1$ >
15   function ReceivePacket
16   Var  $c$ ,  $c_r$ : Πεδίο χρονοσφραγίδας,  $data$ : Πεδίο δεδομένων,  $\Delta C$ : Διαφορά
     ρολογιών,  $SOF$ : Πεδίο αρχής πακέτου,  $TS$ : Χρονοσφραγίδα
17   receive <SOF>
18    $c := Clock$ 
19   receive < $c_r$ >
20   receive < $TS$ >
21   receive < $data$ >
22    $\Delta C := c - c_r$ 
23    $TS := TS + \Delta C$ 
24   if  $i \neq k$  then SendPacket( $TS$ ,  $data$ )

```

υλοποίηση του αλγορίθμου σε πραγματικό σύστημα είναι δύσκολη, ωστόσο είναι δυνατή λήψη κάποιων παραδοχών οι οποίες διευκολύνουν την υλοποίηση και δεν μειώνουν την αποτελεσματικότητα ή την εγχυρότητα του συγχρονισμού.

Β'.2 Σύγκριση με άλλους αλγορίθμους

Σε αυτό το μέρος θα γίνει μια προσπάθεια αναφοράς σε άλλους γνωστούς αλγορίθμους συγχρονισμού για τα WSN.

Β'.2.1 TPSN

Το πρωτόκολλο συγχρονισμού χρονισμού δικτύων αισθητήρων ή timing-sync protocol for sensor networks (TPSN) [27] είναι από τις πρώτες προσπάθειες που έγιναν για να λυθεί το πρόβλημα του συγχρονισμού στα δίκτυα αισθητήρων. Ο TPSN αποσκοπεί να συγχρονίσει τα ρολόγια όλων των κόμβων του δικτύου δημιουργώντας μια ιεραρχική δομή και στην συνέχεια διορθώνοντας τις τιμές των ρολογιών όλων των κόμβων ένα ένα. Αυτή η διαδικασία εισαγάγει έναν πλεονασμό μηνυμάτων στο δίκτυο ο οποίος μπορεί να γίνει εύκολα ασφυκτικός για το σύστημα αν σκεφτούμε τις περιπτώσεις δικτύων με μεγάλο πλήθος κόμβων σε συνδυασμό με εχθρικές συνθήκες περιβάλλοντος όπως υψηλές θερμοκρασίες ή τις υποθαλάσσιες τοποθετήσεις, στις οποίες η ποιότητα των ζεύξεων μειώνεται δραστικά [28, 29] - που μπορεί να επηρεάσει τον αριθμό των χαμένων πακέτων - αλλά έχει αποδειχθεί πως επηρεάζεται και η ολίσθηση των ρολογιών [30] - που θα απαιτήσει τον πιο συχνό συγχρονισμό των κόμβων.

Έχουν γίνει προσπάθειες βελτίωσης του πρωτοκόλλου [31, 32, 33], που επικεντρώνονται κυρίως στο βήμα της δημιουργίας ιεραρχίας. Οι βελτιωμένες εκδοχές καταφέρνουν να μειώσουν τον πλεονασμό μηνυμάτων δημιουργώντας καλύτερα υποσύνολα κόμβων (clusters) και εισάγοντας έναν τύπο αποφυγής συγκρούσεων (collision avoidance). Ωστόσο τα προβλήματα των πρακτικών υλοποιήσεων που περιγράφηκαν παραμένουν και στις καλύτερες εκδοχές του πρωτοκόλλου.

Β'.2.2 RBS

Το πρωτόκολλο συγχρονισμού βάση εκπομπής αναφοράς (reference broadcast synchronization, RBS) [34], βασίζεται στην ιδέα πως είναι δυνατός ο συγχρονισμός των κόμβων βάση της σχετικής απόκλισης που έχουν τα ρολόγια των κόμβων μεταξύ τους, σε αντίθεση με τον συγχρονισμό βάση μιας απόλυτης ώρας δικτύου. Η διαδικασία είναι ως έχει:

- Κόμβος που ξεκινάει την διαδικασία συγχρονισμού στέλνει έναν αριθμό μηνυμάτων συγχρονισμού προς όλους τους κόμβους στην εμβέλειά του.

- Όλοι οι κόμβοι που λαμβάνουν τα μηνύματα καταγράφουν τις τιμές των εσωτερικών τους ρολογιών την ώρα που έλαβαν το εκάστοτε μήνυμα.
- Οι κόμβοι ανταλλάσσουν μεταξύ τους τις τιμές που κατέγραψαν.

Σε αυτό το σημείο όλοι οι κόμβοι-γείτονες του κόμβου που αρχικοποίησε τον συγχρονισμό έχουν τις χρονοσφραγίδες των συμβάντων λήψης μηνυμάτων συγχρονισμού - δεδομένου πως είναι και εκείνοι γείτονες του αρχικού κόμβου. Στην συνέχεια, είναι δυνατό ο κάθε κόμβος να υπολογίσει την σχετική απόκλιση του ρολογιού του με αυτά των γειτόνων του. Η διαδικασία βέβαια πρέπει να επαναληφθεί στην περίπτωση που και ο αρχικός κόμβος χρειάζεται να συγχρονιστεί.

Το πλεονέκτημα της συγκεκριμένης μεθόδου συγχρονισμού είναι ότι διαβεβαιώνεται ο συγχρονισμός όλων των κόμβων του δικτύου αλλά και η μεγάλη ακρίβεια του συγχρονισμού - με την προϋπόθεση πως η διαδικασία θα επαναλαμβάνεται ανά χρονικά διαστήματα. Το μειονέκτημα είναι το πολύ μεγάλο πλήθος απαιτούμενων μηνυμάτων που πρέπει να σταλούν σε κάθε επανάληψη του συγχρονισμού. Για λόγους υπερκατανάλωσης ενέργειας που περιγράφηκαν και στο κεφάλαιο Α', μια τέτοια προσέγγιση μάλλον θα ήταν λιγότερο από ιδανική.

Κεφάλαιο Γ΄

Η Υλοποίηση του Αλγορίθμου

Η προσπάθεια της υλοποίησης του αλγορίθμου σε πραγματικές συνθήκες κρίθηκε απαραίτητη για την εκπόνηση αυτής της εργασίας με σκοπό την μελέτη της συμπεριφοράς του αλγορίθμου σε συνθήκες που είναι εξαιρετικά δύσκολο να προσμετρηθούν στις παραμέτρους ενός περιβάλλοντος προσομοίωσης. Σε αυτό το κεφάλαιο θα περιγραφεί το περιβάλλον της υλοποίησης και οι κόμβοι που χρησιμοποιήθηκαν, το λογισμικό που γράφτηκε και χρησιμοποιήθηκε, η μορφή και οι τύποι μηνυμάτων του συστήματος και τέλος θα γίνει αναφορά στις αποκλίσεις που έγιναν από τον αλγόριθμο αναφοράς.

Γ΄.1 Περιβάλλον υλοποίησης

Τα πειράματα που θα παρουσιαστούν στην συνέχεια έχουν γίνει στο χτίριο ‘Αρεταίος’ του Ιονίου Πανεπιστημίου. Οί κόμβοι είχαν τοποθετηθεί σε διάφορα σημεία του χτιρίου και σε διαφορετικές αποστάσεις. Οι παχείς τοίχοι ήταν τα κύρια φυσικά εμπόδια που υπάρχουν στον χώρο της υλοποίησης, οι οποίοι απαιτούν και την ιδιαίτερα προσεχτική τοποθέτηση των κόμβων για να εξασφαλίζεται η συνεκτικότητα του δικτύου καθώς και η αποφυγή δημιουργίας σχέσεων πατέρα-παιδιού σε ζεύξεις χαμηλής ποιότητας.

Γ΄.1.1 Το ZigBee πρωτόκολλο

Το πρωτόκολλο ZigBee βασίζεται στο πρότυπο IEEE 802.15.4 για το φυσικό επίπεδο (physical layer) και το επίπεδο ελέγχου πρόσβασης (medium access control layer,

Συχνότητα	868MHz	915MHz	2450MHz
Κανάλια	1	10	16
Ρυθμός μετάδοσης	20Kbps	40Kbps	240Kbps
Τόπος Εφαρμογής	Ευρώπη	ΗΠΑ	Παγκόσμια

Πίνακας Γ'.1: Υποστηριζόμενες συχνότητες λειτουργίας

MAC) μέσου και παρέχει ένα δικό του επίπεδο δικτύου network layer και επίπεδο εφαρμογής application layer [35].

Το φυσικό επίπεδο είναι ο πιο κοντινός μεσάζων ενδιάμεσα στο υλικό και τα ανώτερα επίπεδα του πρωτοκόλλου. Παρέχει λειτουργίες για την ενεργοποίηση και απενεργοποίηση του πομποδέκτη, για την λήψη και αποστολή δεδομένων, για την ανίχνευση μέσου, για την επιλογή καναλιού και παρέχει πληροφορίες για την ποιότητα ζεύξεων. Υποστηρίζονται οι συχνότητες των 2.45GHz, με 16 κανάλια επικοινωνίας, 915MHz, με 10 κανάλια επικοινωνίας και 868MHz, με ένα κανάλι επικοινωνίας [36].

Το επίπεδο MAC είναι υπεύθυνο για την σωστή λειτουργία της CSMA/CA μεθόδου για την πρόσβαση στο μέσο, για την σωστή διευθυνσιοδότηση, για την δημιουργία συνδέσεων μεταξύ των MAC επιπέδων δύο διαφορετικών κόμβων, για την αποστολή μηνυμάτων αναγνώρισης (acknowledgment messages) και για την αποστολή μηνυμάτων δεδομένων μέσω του φυσικού επιπέδου.

Το επίπεδο δικτύου είναι υπεύθυνο για την σύνδεση του επιπέδου εφαρμογής με το επίπεδο MAC. Είναι υπεύθυνο για την δημιουργία του δικτύου και για την δρομολόγηση των πακέτων. Επίσης σε αυτό το επίπεδο υλοποιούνται τα τυχόν πρωτόκολλα ασφαλείας του συστήματος.

Το επίπεδο εφαρμογής είναι το υψηλότερο επίπεδο του πρωτοκόλλου που μεταξύ άλλων παρέχει υπηρεσίες για χρήση χαμηλότερων επιπέδων, ορίζει την μορφή των μηνυμάτων και μπορεί να ανιχνεύσει και να αναγνωρίσει διαθέσιμες υπηρεσίες σε άλλες συσκευές του δικτύου.

Όσο αφορά την αρχιτεκτονική του δικτύου, το πρωτόκολλο υποστηρίζει τρία προφίλ κόμβων - κύριου κόμβου (coordinator), δρομολογητή (router) και φύλλου (end device). Οι κύριοι κόμβοι είναι η ρίζα της δεντρικής μορφής του δικτύου και είναι υπεύθυνοι για την αρχικοποίηση και την επιλογή ορισμένων παραμέτρων του δικτύου - όπως της συχνότητας του καναλιού επικοινωνίας. Κάθε ZigBee δίκτυο έχει έναν μό-

νο κύριο κόμβο. Οι δρομολογητές είναι μεσάζουσες συσκευές που χρησιμεύουν στην προώθηση των δεδομένων του δικτύου. Οι δρομολογητές μπορούν ανάλογα με τις ανάγκες της εφαρμογής να παράγουν και οι ίδιοι δεδομένα και να στέλνουν ανεξάρτητα από άλλους κόμβους μηνύματα. Οι κόμβοι φύλλα είναι οι πιο απλές συσκευές του δικτύου, που επικοινωνούν μόνο με έναν κόμβο - πατέρα. Συνήθως έχουν τον ρόλο της συλλογής δεδομένων και απαιτούν την ελάχιστη ενέργεια για την λειτουργία τους, καθώς στέλνουν τα λιγότερα μηνύματα, αλλά έχουν και την δυνατότητα να ενεργοποιήσουν μία κατάσταση αδράνειας (sleep mode) για επιπλέον εξοικονόμηση ενέργειας.

Γ'.1.2 Οι κόμβοι Arduino

Όλοι οι κόμβοι που χρησιμοποιήθηκαν αποτελούνταν από έναν μικροελεγκτή Arduino Uno, μια προέκταση υποδοχής κεραίας και κάρτας SD (arduino wireless SD shield), μια κεραία XBee S2C Pro ZigBee, και μια εξωτερική μπαταρία για παροχή ενέργειας.



Σχήμα Γ'.1: Ένας κόμβος Arduino

Γ'.2 Λογισμικό της εφαρμογής

Το λογισμικό μέρος της υλοποίησης βασίστηκε εκ μέρους στην βιβλιοθήκη 'xbee-arduino' [25] του Andrew Rapp. Η βιβλιοθήκη διευκολύνει αρκετά την χρήση της κεραίας με τις κλάσεις για την χρήση της κεραίας, και κλάσεις πακέτων που υποστηρίζει το ZigBee stack. Επιπλέον χρησιμοποιήθηκε και η βιβλιοθήκη 'SD', για την αλληλεπίδραση με την κάρτα μνήμης που διέθεταν οι κόμβοι για την αποθήκευση των δεδομένων.

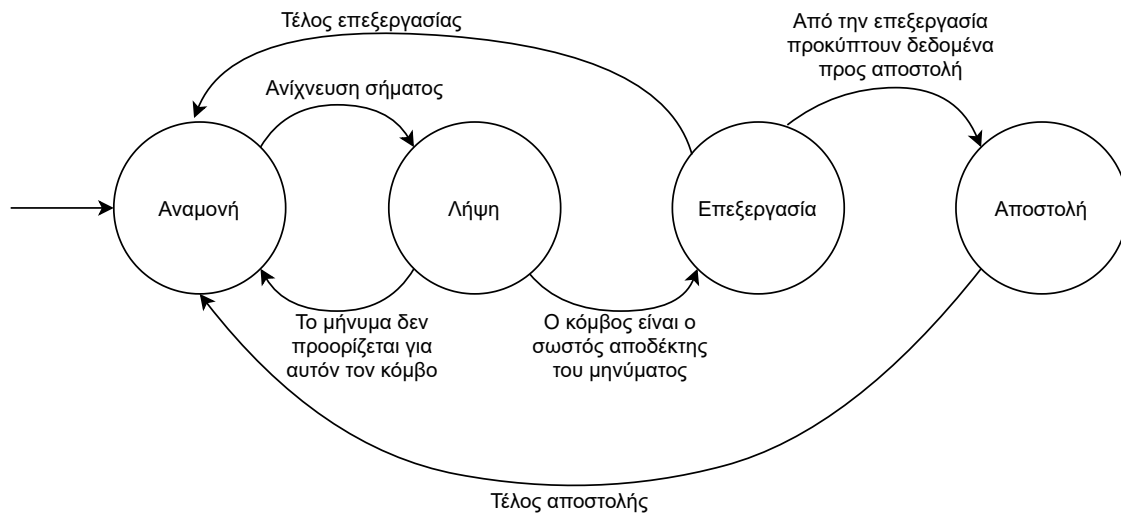
Ειδικά για τους σκοπούς της παρούσης εργασίας γράφτηκε η κλάση MessageList, που υποστηρίζει την δημιουργία και διαχείριση λιστών μηνυμάτων που χρησιμοποιούνται για την αποστολή των δεδομένων, και η Messenger που γράφτηκε κυρίως για την καθαρότητα του κώδικα και λειτουργεί σαν μια πιο αφηρημένη εκδοχή των μερών του 'xbee-arduino' που χρησιμοποιούνται.

Γ'.2.1 Λειτουργία της εφαρμογής

Όλοι οι κόμβοι του δικτύου έχουν δύο καταστάσεις λειτουργίας - την κατάσταση αρχικοποίησης (setup), και την κατάσταση βρόχου (loop), που είναι κοινές γενικότερα σε όλα τα προγράμματα μικροεπεξεργαστών arduino. Στην κατάσταση αρχικοποίησης αρχικοποιούνται οι καθολικές μεταβλητές του προγράμματος και οι τυχόν συνδέσεις σε σειριακές θύρες. Αξίζει να σημειωθεί πως εδώ είναι και το σημείο στο οποίο οι κόμβοι στέλνουν μια εντολή προσοχής (AT command) στις κεραίες τους για να λάβουν γνώση του ρόλου τους στο δίκτυο - αν είναι δηλαδή coordinator ή router. Στην κατάσταση βρόχου εκτελείται στην ουσία η πλειοψηφία της εφαρμογής και εκεί βρίσκεται όλη η λογική του προγράμματος. Για λόγους κατανόησης είναι δυνατός ο διαχωρισμός της κατάστασης βρόχου σε θεωρητικές καταστάσεις αναμονής, λήψης μηνύματος, επεξεργασίας και αποστολής.

Στην κατάσταση αναμονής ο κόμβος απλώς παραμένει αδρανής και ακούει στο κανάλι επικοινωνίας για τυχόν μηνύματα προς αυτόν. Στην κατάσταση λήψης ο κόμβος λαμβάνει το πακέτο που προορίζεται για αυτόν και στην συνέχεια έρχεται η κατάσταση επεξεργασίας στην οποία ο κόμβος αποφασίζει τι πρέπει να γίνει με το μήνυμα που έλαβε. Στην κατάσταση αποστολής ο κόμβος στέλνει το κατάλληλο μήνυμα που προέκυψε από την επεξεργασία.

Κατά την αρχή της διαδικασίας εκτέλεσης δημιουργείται ένα spanning tree του οποίου η ρίζα είναι ο coordinator. Αυτό πετυχαίνεται ουσιαστικά με ένα flooding το



Σχήμα Γ'.2: Καταστάσεις λειτουργίας κόμβων

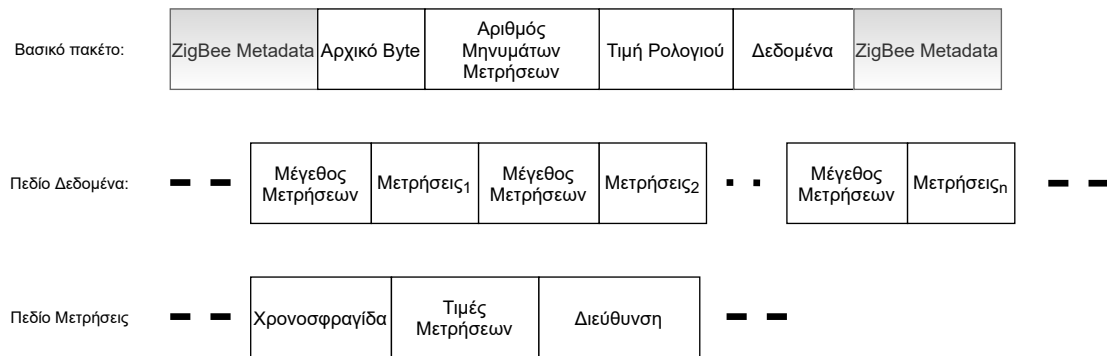
οποίο ξεκινά ο coordinator, κατά το οποίο ο κάθε κόμβος που λαμβάνει για πρώτη φορά το μήνυμα ορίζει ως πατέρα του τον αποστολέα του μηνύματος, και ως παιδιά του τους κόμβους που απαντάνε ρητά - με διαφορετικό μήνυμα - στο broadcast του. Στην συνέχεια οι κόμβοι λειτουργούν ως έχει:

Ο coordinator:

- Ανά χρονικά διαστήματα ξεκινάει ένα flooding με αίτημα την συλλογή των δεδομένων από όλους τους κόμβους.
- Όταν λαμβάνει μήνυμα με δεδομένα κόμβου τα αποθηκεύει στην SD.

Οι υπόλοιποι κόμβοι:

- Περιμένουν να λάβουν κάποιο μήνυμα
- Όταν λάβουν μήνυμα με αίτηση για δεδομένα το προωθούν στην περίπτωση που δεν είναι φύλλα του δέντρου. Εάν είναι φύλλα τότε συλλέγουν τα δεδομένα και τα αποστέλλουν στον πατέρα τους.
- Όταν λάβουν μήνυμα δεδομένων το προωθούν στον πατέρα τους και στην συνέχεια συλλέγουν οι ίδιοι τα δεδομένα τους και τα στέλνουν και αυτά στον πατέρα τους.



Σχήμα Γ'.3: Μορφή πακέτου δεδομένων υλοποίησης

Γ'.3 Μορφή του πακέτου

Βασικό ελάττωμα της συγκεκριμένης υλοποίησης του αλγορίθμου συγχρονισμού δεδομένων είναι η χρήση των ήδη έτοιμων ZigBee πακέτων. Και αυτό είναι ελάττωμα γιατί στο [24] ο αλγόριθμος προϋποθέτει τον προγραμματισμό του αλγορίθμου σε φυσικό επίπεδο, πράγμα που δεν έγινε στην παρούσα προσπάθεια για λόγους ευκολίας. Ωστόσο, ήταν ανάγκη για την λογική του συστήματος να δημιουργηθούν κάποιου είδους 'ψευδο-πακέτα' τα οποία θα εμφιαλωνόντουσαν στο πεδίο δεδομένων (data frame) του βασικού ZigBee πακέτου. Σε αυτήν την ενότητα θα γίνει μία προσπάθεια της περιγραφής τους.

Γ'.3.1 Το πακέτο των δεδομένων

Η μορφή του πακέτου όπως φαίνεται στο παρακάτω διάγραμμα σχεδιάστηκε με γνώμονα τα ζητούμενα της γενικής περίπτωσης. Σε ένα ιδανικό σενάριο η μορφή του πακέτου θα ήταν τέτοια ώστε να είναι εύκολα διαχειρίσιμο, ευέλικτο στις τυχόν αλλαγές της λογικής λειτουργίας του συστήματος και στα περιεχόμενα των δεδομένων αλλά και μη επιβαρυντικό για το σύστημα. Αυτός είναι και ο σκοπός του παρόντος σχεδιασμού.

Στο σχήμα το πακέτο αναλύεται σε τρία στάδια - στο βασικό πακέτο, που είναι η ολόκληρη μορφή του, έτσι όπως κυκλοφορεί στο σύστημα, στο πεδίο των δεδομένων, που περιγράφει την διευθέτηση των μετρήσεων στην περίπτωση αποστολής δεδομένων συλλεγμένων από πολλαπλούς κόμβους, και τέλος στο πεδίο των μετρήσεων που είναι το πεδίο στο οποίο βρίσκονται τα συλλεχθέντα δεδομένα.

Πιο αναλυτικά, όλα τα πεδία του πακέτου είναι ως έχει:

- Το πεδίο αρχικού byte χρησιμεύει στην διάκριση της συγκεκριμένης μορφής πακέτου από υπόλοιπες που χρησιμοποιεί το σύστημα. Στην περίπτωση μηνύματος δεδομένων παίρνει την τιμή 82.
- Στο επόμενο πεδίο περιέχεται η πληροφορία για τον αριθμό των κόμβων από τους οποίους έχουν συλλεχθεί δεδομένα στο συγκεκριμένο πακέτο. Μέγεθος του πεδίου είναι 1 byte.
- Στο πεδίο τιμής ρολογιού βρίσκεται η χρονοσφραγίδα της στιγμής που στάλθηκε το πακέτο. Μέγεθος του πεδίου είναι 4 bytes
- Το πεδίο των δεδομένων αποτελείται από μία σειρά εναλλασσόμενων δυάδων πεδίων μεγέθους μετρήσεων και μετρήσεων. Το τελικό μέγεθος θα εξαρτηθεί από τον τρόπο υλοποίησης του συστήματος.
- Το πεδίο των μετρήσεων παρέχει την χρονοσφραγίδα της στιγμής που συλλέχθηκαν τα δεδομένα, τις τιμές των μετρήσεων και τα 4 σημαντικά bytes της διεύθυνσης του κόμβου που παρήγαγε τα συγκεκριμένα δεδομένα. Το μέγεθος της χρονοσφραγίδας είναι 4 bytes, ενώ το μέγεθος των μετρήσεων διαφέρει με την κάθε υλοποίηση.

Γ'.3.2 Τα υπόλοιπα πακέτα

Οι μορφές των υπόλοιπων μηνυμάτων δεν έχουν κάποια ιδιαίτερη δομή. Είναι απλώς ένα byte δεδομένων που - μαζί με τα υπόλοιπα μεταδεδομένα του ZigBee πακέτου - ανάλογα με την τιμή που παίρνει καθορίζει σε ποιο σημείο της εκτέλεσης θα βρεθεί ο κόμβος. Συνοπτικά, υπάρχουν έξι τύποι μηνυμάτων που αναγνωρίζει το σύστημα. Ένα είναι το 82, που περιγράφηκε αναλυτικά προηγουμένως. Τα υπόλοιπα είναι το 72, που είναι το μήνυμα broadcast, για την ανακάλυψη κόμβων και δημιουργία σχέσεων, το 71, που είναι το μήνυμα αίτησης για συλλογή δεδομένων, το 67, που δηλώνει σχέση παιδιού-πατέρα, και το στέλνει πάντοτε ο κόμβος-παιδί, το 80, που προαναγγέλλει το μήνυμα των δεδομένων, και τέλος το 7, που χρησιμοποιείται για την επανεκκίνηση όλων των κόμβων του δικτύου.

Γ'.4 Απόκλιση από τον αλγόριθμο αναφοράς

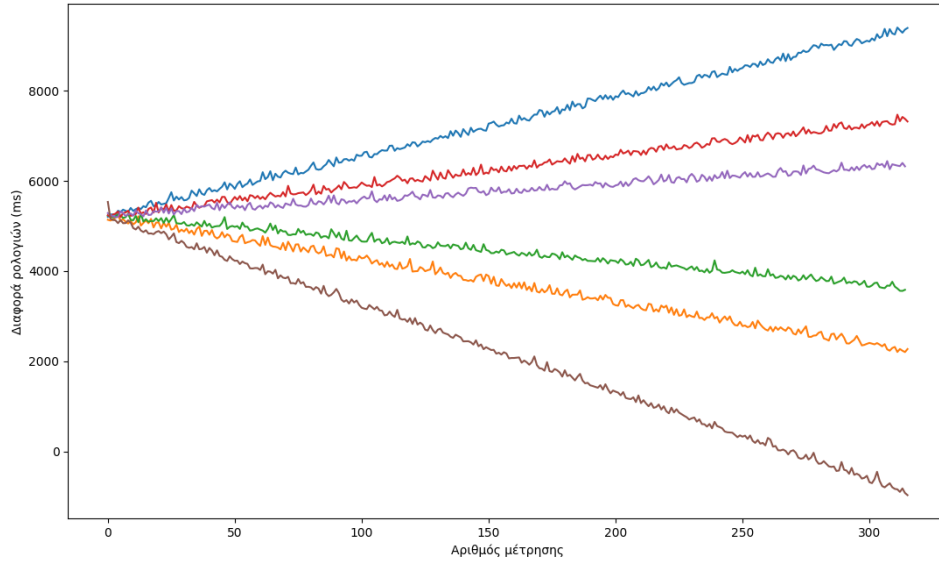
Όπως σημειώθηκε, η κύρια διαφορά μεταξύ του αλγορίθμου αναφοράς και της υλοποίησης είναι η έλλειψη της βελτιστοποίησης της μορφής του πακέτου στο φυσικό επίπεδο. Ο θεωρητικός αλγόριθμος προϋποθέτει την αποστολή μηνυμάτων των δεδομένων σε δύο μέρη, λειτουργικότητα που δεν παρέχεται από την βιβλιοθήκη [25] που χρησιμοποιήθηκε. Ωστόσο, μετά από μετρήσεις που έγιναν για τον έλεγχο της λειτουργίας της εφαρμογής φάνηκε πως στην συγκεκριμένη υλοποίηση αυτή η διαφοροποίηση δεν είχε κάποια επίπτωση. Η μετρήσεις που έγιναν αφορούσαν την ικανότητα του συστήματος να υπολογίζει την διαφορά των τιμών ρολογιών μεταξύ δύο κόμβων αναφοράς. Πιο συγκεκριμένα, γράφτηκε ο κώδικας του I.4 για την καταγραφή της απόκλισης ρολογιών δύο κόμβων από έναν υπολογιστή. Φάνηκε πως η υλοποίηση σημειώνει όμοιες σχετικές απόκλισης σε σχέση με αυτές των αποκλίσεων μεταξύ κόμβων και υπολογιστή και κρίθηκε πως λειτουργεί σωστά.

Αυτό εξηγείται και θεωρητικά καθώς μπορούμε να σκεφτούμε στην πραγματικότητα όλο το πακέτο ZigBee σαν ένα αρκετά μεγάλο preamble, γεγονός το οποίο βάση των υποθέσεων που παίρνει ο αλγόριθμος αναφοράς για τις καθυστερήσεις δεν παρουσιάζει κάποιο πρόβλημα.

Γ'.5 Μετρήσεις : Αποκλίσεις ρολογιών σε ιδανικές συνθήκες

Από τους πρώτους στόχους ήταν η κατανόηση της συμπεριφοράς των ρολογιών σε συνθήκες όπου θα ήταν σίγουρο πως δεν θα υπήρχαν χαμένα μηνύματα, και πως οι συνθήκες περιβάλλοντος θα ήταν σταθερές για όλους τους κόμβους. Για να γίνει αυτό τοποθετήθηκαν 7 κόμβοι σε κοντινή απόσταση μεταξύ τους και σε εσωτερικό περιβάλλον. Η τοπολογία που προκύπτει είναι ουσιαστικά μια τοπολογία αστέρα με κέντρο τον coordinator. Το σύστημα αφέθηκε να λειτουργήσει για μια περίπου ώρα, συλλέγοντας δεδομένα κάθε 10 δευτερόλεπτα. Παρακάτω φαίνεται το γράφημα της απόκλισης ρολογιών όλων των κόμβων σε σχέση με το ρολόι του coordinator.

Σε όλες τις επαναλήψεις του πειράματος προέκυψαν όμοια αποτελέσματα, με τις σχετικές ολισθήσεις ρολογιών να παραμένουν σταθερές μεταξύ τους. Οι συγκεκριμένες μετρήσεις συμπίπτουν με την βιβλιογραφία [23], που σημειώνει πως δεδομένου



Σχήμα Γ'.4: Αποκλίσεις ρολογιών σε σχέση με τον coordinator

σταθερών εξωτερικών συνθηκών, οι σχετικές ολισθήσεις ρολογιών παραμένουν σταθερές.

Κεφάλαιο Δ΄

Σύνοψη και Συμπεράσματα

Το κύριο αντικείμενο αυτής της εργασίας υπήρξαν τα θέματα συγχρονισμού ρολογιών σε ασύρματα δίκτυα αισθητήρων και ειδικότερα η υλοποίηση του αλγορίθμου συγχρονισμού δεδομένων, όπως αυτός περιγράφεται στο [24] σε πραγματικό WSN. Σε αυτό το σημείο θα γίνει μια ανακεφαλαίωση της δουλειάς που έχει γίνει.

Δ΄.1 Σύνοψη

Στο κεφάλαιο Α΄ έγινε μία εισαγωγή στις έννοιες των WSN, και ορίστηκαν τα βασικά προβλήματα που αφορούν τον συγχρονισμό και τις αποκλίσεις ρολογιών. Στο κεφάλαιο Β΄ έγινε μια αναλυτική περιγραφή της λογικής και της λειτουργίας του αλγορίθμου αναφοράς, και συζητήθηκαν δύο από τους πιο γνωστούς από την βιβλιογραφία αλγορίθμους συγχρονισμού για τα WSN. Στο κεφάλαιο Γ΄ ακολούθησε η αναφορά του υλικού που χρησιμοποιήθηκε για την υλοποίηση του συστήματος και περιγράφηκε αναλυτικότερα η λειτουργία του υλοποιημένου συστήματος καθώς και τα οι μορφές των πακέτων που χρησιμοποιήθηκαν. Τέλος, παρουσιάστηκαν τα αποτελέσματα των μετρήσεων που αφορούσαν τις σχετικές αποκλίσεις ρολογιών των κόμβων του δικτύου.

Δ΄.2 Συμπεράσματα

Κλείνοντας, από όσα γράφτηκαν σε αυτήν την εργασία μπορούμε να συμπεράνουμε πως το θέμα του συγχρονισμού στα WSN αξίζει να διερευνηθεί περισσότερο, με εμφάσεις

σε διαφορετικούς τρόπους προσέγγισης της επίλυσης του προβλήματος. Όπως φαίνεται και από τα αποτελέσματα του Γ'.5, είναι πιθανή η δυνατότητα της συμπερίληψης των δεδομένων που αφορούν τα ρολόγια (λ.χ. λόξωση) στα πρωτόκολλα συγχρονισμού, με σκοπό την πρόβλεψη μελλοντικής απόκλισης και εμπρόθεσμης διόρθωσης τιμών ρολογιών, χωρίς την ανάγκη αποστολής μηνυμάτων. Παρόμοιες ιδέες υπάρχουν και στην βιβλιογραφία [37, 38], και συνεχίζουν να παρουσιάζουν ενδιαφέρον πεδίο έρευνας.

Από την ίδια διαδικασία της υλοποίησης φάνηκε η δυσκολία του προγραμματισμού μικροελεγκτών και η γενικότερη έλλειψη εργαλείων και βιβλιοθηκών που θα βοηθούσαν στην γρηγορότερη ανάπτυξη του πεδίου. Συνεπώς, είναι φανερό η ανάγκη ανάπτυξης λογισμικού ανοιχτού κώδικα που θα βοηθούσε στην διαδικασία προγραμματισμού μικροελεγκτών, αλλά και των πομποδεκτών. Όπως αναφέρθηκε προηγουμένως, και ο κώδικας της υλοποίησης αυτής της εργασίας έχει προσπαθήσει να δημιουργήσει μια βάση για την ανάπτυξη μιας πολύ πιο γενικευμένης βιβλιοθήκης για διαχείριση μηνυμάτων, ωστόσο την στιγμή της συγγραφής αυτό αποτελεί μονάχα μια ιδέα.

Βιβλιογραφία

- [1] T. Arampatzis, J. Lygeros, and S. Manesis, “A survey of applications of wireless sensors and wireless sensor networks,” in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, pp. 719–724, IEEE, 2005.
- [2] B. Rashid and M. H. Rehmani, “Applications of wireless sensor networks for urban areas: A survey,” *Journal of network and computer applications*, vol. 60, pp. 192–219, 2016.
- [3] S. J. Ramson and D. J. Moni, “Applications of wireless sensor networks●”a survey,” in *2017 international conference on innovations in electrical, electronics, instrumentation and media technology (ICEEIMT)*, pp. 325–329, IEEE, 2017.
- [4] V. Jeličić, T. Ražov, D. Oletić, M. Kuri, and V. Bilas, “Maslinet: A wireless sensor network based environmental monitoring system,” in *2011 Proceedings of the 34th International Convention MIPRO*, pp. 150–155, IEEE, 2011.
- [5] L. Bencini, D. Di Palma, G. Collodi, G. Manes, A. Manes, and Y. K. Tan, “Wireless sensor networks for on-field agricultural management process,” *Wireless Sensor Networks: application-centric design*, pp. 1–21, 2010.
- [6] C. M. Angelopoulos, S. Nikolettseas, and G. C. Theofanopoulos, “A smart system for garden watering using wireless sensor networks,” in *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pp. 167–170, 2011.
- [7] A. H. Abbas, M. M. Mohammed, G. M. Ahmed, E. A. Ahmed, and R. A. A. A. A. Seoud, “Smart watering system for gardens using wireless sensor

- networks,” in *2014 International Conference on Engineering and Technology (ICET)*, pp. 1–5, IEEE, 2014.
- [8] M. Taylor, “Climate-smart agriculture: what is it good for?,” *The Journal of Peasant Studies*, vol. 45, no. 1, pp. 89–107, 2018.
- [9] S. Lenka, N. Lenka, V. Sejian, and M. Mohanty, “Contribution of agriculture sector to climate change,” in *Climate change impact on livestock: Adaptation and mitigation*, pp. 37–48, Springer, 2015.
- [10] A. Tsipis, A. Papamichail, I. Angelis, G. Koufoudakis, G. Tsoumanis, and K. Oikonomou, “An alertness-adjustable cloud/fog iot solution for timely environmental monitoring based on wildfire risk forecasting,” *Energies*, vol. 13, no. 14, p. 3693, 2020.
- [11] G. Xu, W. Shen, and X. Wang, “Applications of wireless sensor networks in marine environment monitoring: A survey,” *Sensors*, vol. 14, no. 9, pp. 16932–16954, 2014.
- [12] K. Nisar, A. A. A. Ibrahim, L. Wu, A. Adamov, and M. J. Deen, “Smart home for elderly living using wireless sensor networks and an android application,” in *2016 IEEE 10th international conference on application of information and communication technologies (AICT)*, pp. 1–8, IEEE, 2016.
- [13] S. Vergis, V. Komianos, G. Tsoumanis, A. Tsipis, and K. Oikonomou, “A low-cost vehicular traffic monitoring system using fog computing,” *Smart Cities*, vol. 3, no. 1, pp. 138–156, 2020.
- [14] A. Pascale, M. Nicoli, F. Deflorio, B. Dalla Chiara, and U. Spagnolini, “Wireless sensor networks for traffic management and road safety,” *IET Intelligent Transport Systems*, vol. 6, no. 1, pp. 67–77, 2012.
- [15] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafiekhah, and P. Siano, “Iot-based smart cities: A survey,” in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 1–6, IEEE, 2016.

- [16] I. Howitt, W. W. Manges, P. T. Kuruganti, G. Allgood, J. A. Gutierrez, and J. M. Conrad, “Wireless industrial sensor networks: Framework for qos assessment and qos management,” *ISA transactions*, vol. 45, no. 3, pp. 347–359, 2006.
- [17] L. Hou and N. W. Bergmann, “Novel industrial wireless sensor networks for machine condition monitoring and fault diagnosis,” *IEEE transactions on instrumentation and measurement*, vol. 61, no. 10, pp. 2787–2798, 2012.
- [18] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, “Clock synchronization of wireless sensor networks,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, 2010.
- [19] H.-Y. Zhou, D.-Y. Luo, Y. Gao, and D.-C. Zuo, “Modeling of node energy consumption for wireless sensor networks,” *Wireless Sensor Network*, vol. 3, no. 1, p. 18, 2011.
- [20] P. Sommer and R. Wattenhofer, “Symmetric clock synchronization in sensor networks,” in *Proceedings of the workshop on Real-world wireless sensor networks*, pp. 11–15, 2008.
- [21] T. Schmid, Z. Charbiwala, J. Friedman, Y. H. Cho, and M. B. Srivastava, “Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, pp. 97–108, 2008.
- [22] A. Elsts, X. Fafoutis, S. Duquennoy, G. Oikonomou, R. Piechocki, and I. Craddock, “Temperature-resilient time synchronization for the internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2241–2250, 2017.
- [23] S. K. Mani, R. Durairajan, P. Barford, and J. Sommers, “An architecture for iot clock synchronization,” in *Proceedings of the 8th International Conference on the Internet of Things, IOT ’18*, (New York, NY, USA), Association for Computing Machinery, 2018.
- [24] K. Skiadopoulos, A. Tsipis, K. Giannakis, G. Koufoudakis, E. Christopoulou, K. Oikonomou, G. Kormentzas, and I. Stavrakakis, “Synchronization of data

- measurements in wireless sensor networks for iot applications,” *Ad Hoc Networks*, vol. 89, pp. 47–57, 2019.
- [25] A. Rapp, “xbee-arduino.” <https://github.com/andrewrapp/xbee-arduino>, 2015.
- [26] E. Alexiou and N. Jevtic, “field stalker.” <https://github.com/leuteris99/field-stalker-public>, 2020.
- [27] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync protocol for sensor networks,” in *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 138–149, 2003.
- [28] G. Horvat, D. Vinko, and J. Vlaović, “Impact of propagation medium on link quality for underwater and underground sensors,” in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 129–134, IEEE, 2016.
- [29] N. Baccour, A. Koubâa, L. Mottola, M. A. Zuniga, H. Youssef, C. A. Boano, and M. Alves, “Radio link quality estimation in wireless sensor networks: A survey,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, no. 4, pp. 1–33, 2012.
- [30] M. B. Uddin and C. Castelluccia, “Toward clock skew based wireless sensor node services,” in *2010 The 5th Annual ICST Wireless Internet Conference (WICON)*, pp. 1–9, IEEE, 2010.
- [31] P. Gou, F. Li, Y. Wang, and M. Li, “Low energy consumption and high-precision time synchronization algorithm based on improved tpsn in wireless sensor networks,” in *2018 International Conference on Engineering Simulation and Intelligent Control (ESAIC)*, pp. 54–57, IEEE, 2018.
- [32] S. El Khediri, N. Nasri, H. Hedfi, A. Kachouri, and A. Wei, “An improved tpsn algorithm for time synchronization in wireless sensors networks,” in *The International Conference on Information Processing and Wireless Systems (IP-WiS)*, Citeseer, 2013.

- [33] D. Liu, Z. Zheng, Z. Yuan, and W. Li, “An improved tpsn algorithm for time synchronization in wireless sensor network,” in *2012 32nd International Conference on Distributed Computing Systems Workshops*, pp. 279–284, IEEE, 2012.
- [34] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.
- [35] D. Gislason, *Zigbee wireless networking*. Newnes, 2008.
- [36] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, “Study on zigbee technology,” in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, pp. 297–301, IEEE, 2011.
- [37] J. M. Castillo-Secilla, J. M. Palomares, and J. Olivares, “Temperature-compensated clock skew adjustment,” *Sensors*, vol. 13, no. 8, pp. 10981–11006, 2013.
- [38] Z. Yang, L. Cai, Y. Liu, and J. Pan, “Environment-aware clock skew estimation and synchronization for wireless sensor networks,” in *2012 Proceedings IEEE INFOCOM*, pp. 1017–1025, 2012.

Παραρτήματα

Παράρτημα Ι

Κώδικες

I.1 Κύριος κώδικας κόμβων

```
#include <XBee.h>
#include <MessageList.h>
#include <Messenger.h>
#include <SD.h>

//The class to send and recieve messages
Messenger msger = Messenger();
uint8_t payload[] = {'H','i'};
bool isCoordinatorEnabled = false;
unsigned long getting_time;

void setup(){
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    while (1);
  }
}
```

```

    }
    isCoordinatorEnabled = msger.sendATCommand("CE");
    delay(2000);
    if(isCoordinatorEnabled){
        payload[0] = 7; //BELL - RESET
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        delay(10000);
        payload[0] = 72;
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        getting_time = millis();
    }
}

void(* resetFunc)(void)=0;

void loop(){
    if(isCoordinatorEnabled && millis() - getting_time > 10000){
        payload[0] = 71;
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        getting_time = millis();
    }
    msger.changed = false;
    msger.checkForPacketDelivery();
    if(msger.changed && (msger.msg->getMessageValue() == 7) &&
        !isCoordinatorEnabled){
        payload[0] = 7; //BELL - RESET
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        delay(3000);
        resetFunc();
    }
    if(msger.changed && msger.msg->getMessageValue() == 72 && msger.father
        == msger.bcst_address){
        msger.father = XBeeAddress64(msger.msg->getSenderAddressHigh(),
            msger.msg->getSenderAddressLow());
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        payload[0] = 67;
    }
}

```

```
    msger.sendMessage(payload, sizeof(payload), msger.father);
}
if(msger.changed && msger.msg->getMessageValue() == 67){
    msger.setChild();
    Serial.println("No of children:");
    Serial.println(msger.getChildCount());
    msger.clearMsg();
}
else if(msger.changed && msger.msg->getMessageValue() == 71 &&
        msger.father.getLsb() == msger.msg->getSenderAddressLow()){
    if(msger.getChildCount()){
        payload[0] = 71;
        sendPreamble();
        sendMeasurments();
        msger.sendMessage(payload, sizeof(payload), msger.bcst_address);
        msger.clearMsg();
    }
    else{
        sendPreamble();
        sendMeasurments();
        msger.clearMsg();
    }
}
else if(msger.changed && msger.msg->getMessageValue() == 80){
    msger.setClockOnRecieve();
    msger.clearMsg();
}
else if(msger.changed && msger.msg->getMessageValue() == 82){
    msger.setChildDeviation();
    msger.writeClock();
    if(!isCoordinatorEnabled){
        msger.forwardDataToFather();
    }
    else{
        printTheData();
    }
}
```

```
    msger.clearMsg();
}
}

void sendPreamble(){
    payload[0] = 80;
    msger.sendMessage(payload, sizeof(payload), msger.father);
}

void sendMeasurments(){
    unsigned long this_clock;
    int temperature = getTemperatureSensorValue(A0);
    int humidity = getLightSensorValue(A1);
    int light = getMoistySensorValue(A2);
    uint8_t data_size = sizeof(temperature) + sizeof(humidity) +
        sizeof(light) + sizeof(this_clock);
    uint8_t measurments[7 + data_size];
    measurments[0] = 82;
    measurments[1] = 1;
    measurments[6] = data_size;
    this_clock = millis();
    for(byte i = 0; i < 4; i++){
        measurments[i+2] = this_clock >> i*8;
        measurments[i+7] = measurments[i+2];
    }
    measurments[11] = temperature;
    measurments[12] = temperature >> 8;
    measurments[13] = humidity;
    measurments[14] = humidity >> 8;
    measurments[15] = light;
    measurments[16] = light >> 8;

    msger.sendMessage(measurments, sizeof(measurments), msger.father);
}

int getTemperatureSensorValue(int tempPin){
```

```
    return analogRead(tempPin);
}
int getLightSensorValue(int lightPin){
    return analogRead(lightPin);
}
int getMoistySensorValue(int moistyPin){
    return analogRead(moistyPin);
}

void printTheData(){
    Serial.print("da,");
    Serial.print(msger.getTemperatureValue());
    Serial.print(",");
    Serial.print(msger.getLightValue());
    Serial.print(",");
    Serial.print(msger.getMoistyValue());
    Serial.println();

    Serial.println("dv," + String(millis()) + ",0,0");
    Serial.print("da,");
    Serial.print(getTemperatureSensorValue(A0));
    Serial.print(",");
    Serial.print(getLightSensorValue(A1));
    Serial.print(",");
    Serial.print(getMoistySensorValue(A2));
    Serial.println();
}
```

I.2 Κλάση Messenger

\\Messenger.h file

```
#include "Arduino.h"
```

```
#include <XBee.h>
#include<MessageList.h>
#include <SD.h>

#ifndef Messenger_h
#define Messenger_h

class Messenger
{
public:
    Messenger();
    ~Messenger();
    void sendMessage(uint8_t * msg, uint8_t msg_size, XBeeAddress64
        dest_address);
    byte sendATCommand(String command);
    void checkForPacketDelivery();
    void clearMsg();
    void setClockOnRecieve();
    void setChildDeviation();
    void forwardDataToFather();
    void writeClock();
    int16_t getLightValue();
    int16_t getMoistyValue();
    int16_t getTemperatureValue();
    byte setChild();
    byte getChildCount();
    XBeeAddress64 father;
    XBeeAddress64 bcst_address;
    MessageList *msg;
    bool changed;
    File data_file;
private:
    uint8_t *the_data;
    unsigned long clockOnRecieve;
    unsigned long clockOnRecievePreamble;
    uint32_t children[15];
```

```
    int32_t childrenDeviation[15];
    byte childCount;
    XBee xbee;
    ZBTxRequest *zbTx;
    uint8_t payload[2] = {'H', 'i'};
    ZBExplicitRxResponse explicitResponse = ZBExplicitRxResponse();
};
```

```
#endif
```

```
//Messenger.cpp file
```

```
#include "Messenger.h"
#include "Arduino.h"
```

```
Messenger::Messenger(){
    this->xbee = XBee();
    this->father = XBeeAddress64(0x00000000, 0x0000ffff);
    this->bcst_address = XBeeAddress64(0x00000000, 0x0000ffff);
    this->zbTx = new ZBTxRequest(this->bcst_address, this->payload,
        sizeof(this->payload));
    this->msg = new MessageList();
    this->changed = false;
    this->childCount = 0;
    for(byte i = 0; i < 15; i++){
        this->childrenDeviation[i] = 0;
        this->children[i] = 0;
    }
}
```

```
Messenger::~Messenger(){
    delete this->msg;
    delete this->zbtX;
}

void Messenger::sendMessage(uint8_t * msg, uint8_t msg_size,
    XBeeAddress64 dest_address) {
    delete zbtX;
    this->zbtX = new ZBTxRequest(dest_address, msg, msg_size);
    this->zbtX->setBroadcastRadius(0x1);
    this->xbee.send(*this->zbtX);
}

void Messenger::checkForPacketDelivery(){
    this->xbee.readPacket();
    ZBTxStatusResponse txStatus = ZBTxStatusResponse();
    if (this->xbee.getResponse().isAvailable()) {
        this->changed = true;
        // got something
        this->clockOnRecieve = millis();
        //Serial.println(F("Got The Message"));
        if (this->xbee.getResponse().getApiId() ==
            ZB_EXPLICIT_RX_RESPONSE) {
            this->xbee.getResponse().getZBExplicitRxResponse(this->explicitResponse);
            this->msg->newMessage(this->explicitResponse.getRemoteAddress64().getLsb(),
                                this->explicitResponse.getRemoteAddress64().getMsb(),
                                this->explicitResponse.getFrameData(),
                                this->explicitResponse.getDataOffset(),
                                this->explicitResponse.getDataLength());
        }
    }
    else if (this->xbee.getResponse().getApiId() ==
        ZB_TX_STATUS_RESPONSE) {
        this->xbee.getResponse().getZBTxStatusResponse(txStatus);
        //Serial.println(txStatus.getDeliveryStatus(), HEX);
    }
}
```

```

        //get the delivery status, the fifth byte
        if (txStatus.getDeliveryStatus() == SUCCESS) {
            //Serial.print(F("Successful transmtion\n"));
        }
        else {
            // the remote XBee did not receive our packet. is it powered
            // on?
            //Serial.print(F("Unsuccessful transmtion\n"));
        }
    }
    else if (this->xbee.getResponse().getApiId() == ZB_RX_RESPONSE) {
        //Serial.println(F("It Is A ZB_RX_RESPONSE"));
    } else if (this->xbee.getResponse().isError()) {
        //Serial.print(F("Error reading packet. Error code: "));
        //nss.println(xbee.getResponse().getErrorCode());
    }
}

void Messenger::setChildDeviation(){
    for (byte i = 0; i < this->childCount; i++){
        if (this->children[i] == this->msg->getSenderAddressLow()){
            this->childrenDeviation[i] =
                int32_t(int64_t(this->clockOnRecievePreamble) -
                    int64_t(this->msg->getLastClockValue()));
            Serial.print("dv," + String(this->msg->getLastClockValue()) +
                "," + String(this->childrenDeviation[i]) + "," +
                String(this->childrenDeviation[i] - int32_t(32)));
            Serial.println(this->msg->getSenderAddressLow());
            break;
        }
    }
}

void Messenger::forwardDataToFather(){
    for (byte i = 0; i < this->childCount; i++){
        if (this->children[i] == this->msg->getSenderAddressLow()){

```

```
        this->msg->appendMessage(this->childrenDeviation[i]);
        this->the_data = new uint8_t[this->msg->getListSize() + 1];
        this->the_data[0] = 82;
        for(byte j = 1; j< this->msg->getListSize() + 1; j++){
            this->the_data[j] = this->msg->getList()[j-1];
        }
        this->sendMessage(this->the_data, this->msg->getListSize() +
            1, this->father);
        delete this->msg;
        this->msg = new MessageList();
        break;
    }
}

void Messenger::writeClock(){
    for (byte i = 0; i < this->childCount; i++){
        if(this->children[i] == this->msg->getSenderAddressLow()){
            this->data_file = SD.open("datafile.csv", FILE_WRITE);
            this->data_file.print(this->childrenDeviation[i]);
            this->data_file.print(",");
            this->data_file.println(this->msg->getSenderAddressLow());
            this->data_file.close();
            break;
        }
    }
}

void Messenger::setClockOnRecieve(){
    this->clockOnRecievePreamble = this->clockOnRecieve;
}

void Messenger::clearMsg(){
    this->msg->deletePacketData();
}
```

```
byte Messenger::setChild(){
    for(byte i = 0; i < this->childCount; i++)
        if(this->children[i] == this->msg->getSenderAddressLow()) return 0;
    this->children[this->childCount] = this->msg->getSenderAddressLow();
    this->childCount++;
    return 1;
}

byte Messenger::getChildCount(){
    return this->childCount;
}

int Messenger::getTemperatureValue(){
    return (int16_t(this->msg->getData()[12]) << 8) |
        this->msg->getData()[11];
}

int Messenger::getMoistyValue(){
    return (int16_t(this->msg->getData()[14]) << 8) |
        this->msg->getData()[13];
}

int Messenger::getLightValue(){
    return (int16_t(this->msg->getData()[16]) << 8) |
        this->msg->getData()[15];
}

byte Messenger::sendATCommand(String command) {
    //This is the Coordinator Command that we need to read upon the
    //initialization of the
    //network so that we can tell the coordinator only to start looking for
    //neighbours.
    uint8_t CoEnCmd[] = {'C', 'E'};
    //This Command is nessessary for writing the command we previously gave
    //to the XBee module.
    uint8_t writeCmd[] = { 'W', 'R' };
```

```

//Below we have the request for the AT Command
AtCommandRequest atRequest = AtCommandRequest(CoEnCmd);
//And here we have the response of the AT Command Request.
AtCommandResponse atResponse = AtCommandResponse();
if (command == "CE") {
    //Here we tell the request which command to send to the XBee module.
    atRequest.setCommand(CoEnCmd);
} else if (command == "WR") {
    atRequest.setCommand(writeCmd);
}
// send the command
this->xbee.send(atRequest);
// wait up to 5 seconds for the status response
if (this->xbee.readPacket(250)) {
    // got a response!
    // should be an AT command response
    if (this->xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
        this->xbee.getResponse().getAtCommandResponse(atResponse);
        if (atResponse.isOk()) {
            if (atResponse.getValueLength() > 0) {
                for (int i = 0; i < atResponse.getValueLength(); i++) {
                    if (atResponse.getValue()[i] == 0x1) {
                        Serial.println(" Coordinator Is Enabled");
                        return 1;
                    }
                }
            }
        }
        else {
            Serial.print("Command return error code: ");
            Serial.println(atResponse.getStatus(), HEX);
        }
    } else {
        Serial.print("Expected AT response but got ");
        Serial.print(this->xbee.getResponse().getApiId(), HEX);
    }
}

```

```

    } else {
        // at command failed
        if (this->xbee.getResponse().isError()) {
            Serial.print("Error reading packet. Error code: ");
            Serial.println(this->xbee.getResponse().getErrorCode());
        }
        else {
            Serial.print("No response from radio");
        }
    }
    return 0;
}

```

I.3 Κλάση MessageList

```

#include "Arduino.h"
#ifndef MessageList_h
#define MessageList_h

class MessageList
{
public:
    MessageList();
    ~MessageList();
    uint8_t* getList();
    void newMessage(uint32_t addL, uint32_t addH, uint8_t *frameData,
                    int offset, int dataSize);
    void appendMessage(int64_t c);
    uint8_t getMessageValue();
    uint32_t getSenderAddressLow();
    uint32_t getSenderAddressHigh();
    uint8_t* getData();
    int getListSize();
    int getDataSize();

```

```
void deleteBigData();
void deletePacketData();
unsigned long getLastClockValue();
int getMessageCount();
void makeArray(uint8_t* arr, uint32_t address);

private:
    int messageCount;
    void makeLong(unsigned long* l, uint8_t* arr);
    void setLastClockValue();
    int dataSize;
    uint32_t addressLow;
    uint32_t addressHigh;
    uint8_t *packetData;
    uint8_t *bigData;
    int bigDataSize;
    unsigned long clockValue;

};

#endif



---




---


//MessageList.cpp file

#include "MessageList.h"
#include "Arduino.h"

MessageList::MessageList(){
    this->addressLow = 0;
    this->addressHigh = 0;
    this->bigDataSize = 0;
    this->dataSize = 0;
    this->messageCount = 0;
    this->clockValue = 0;
}
```

```
MessageList::~MessageList(){
    this->deleteBigData();
    this->deletePacketData();
}

uint8_t* MessageList::getList(){
    return this->bigData;
}

void MessageList::newMessage(uint32_t addL, uint32_t addH, uint8_t
    *frameData, int offset, int dataSize){
    this->messageCount++;
    this->deletePacketData();
    int j = 0;
    this->packetData = new uint8_t[dataSize];
    for (int i = offset; i < offset + dataSize; i++){
        this->packetData[j] = frameData[i];
        j++;
    }
    this->addressLow = addL;
    this->addressHigh = addH;
    this->dataSize = dataSize;
    if(packetData[0] == 82){
        this->setLastClockValue();
    }
}

void MessageList::appendMessage(int64_t c ){
    /*
    Remove the recieved packet to the bigData array, without the first 6
    bytes
    and add the sender address at the end.
    */
}
```

```

*/
byte address_size = 4;
unsigned long newTimestamp;
uint8_t *add = new uint8_t[address_size];
this->makeArray(add, this->addressLow);
uint8_t *temp = new uint8_t[this->bigDataSize];
for(int i = 0; i < this->bigDataSize; i++){
    temp[i] = this->bigData[i];
}
if (this->bigDataSize){
    delete[] this->bigData;
}
this->bigData = new uint8_t[this->bigDataSize + this->dataSize - 6 +
    address_size];
for(int i = 0; i < this->bigDataSize; i++){
    this->bigData[i] = temp[i];
}
int j = 6;
for(int i = 0; i < this->packetData[1]; i++){
    // c is difference
    makeLong(&newTimestamp, &this->packetData[j + 1]);
    newTimestamp += c;
    makeArray(&this->packetData[j+1], (uint32_t)newTimestamp);
    j += this->packetData[j];
}
j=0;
for(int i = 6; i < this->dataSize; i++){
    this->bigData[this->bigDataSize + j] = this->packetData[i];
    j++;
}
Serial.println("data size is:");
Serial.println(this->getDataSize());
if(this->getDataSize() <= 17){
    for(int i = 0; i < address_size; i++){
        this->bigData[this->bigDataSize + this->dataSize - 6 + i] = add[i];
    }
}

```

```
        this->bigDataSize = this->bigDataSize+ this->dataSize - 6 +
            address_size;
    }
    else{
        this->bigDataSize = this->bigDataSize + this->dataSize - 6;
    }
    if(this->bigDataSize){
        delete[] temp;
    }
    delete[] add;
}

void MessageList::deletePacketData(){
    if (this->dataSize>0){
        delete[] this->packetData;
        this->dataSize = 0;
    }
}

void MessageList::deleteBigData(){
    if(this->bigDataSize){
        delete[] this->bigData;
        this->bigDataSize = 0;
    }
}

void MessageList::makeArray(uint8_t* arr, uint32_t address){
    for(byte i = 0; i < 4; i++){
        arr[i] = address >> i*8;
    }
}

void MessageList::makeLong(unsigned long* l, uint8_t* arr){
```

```
*l = 0;
for(int i = 3; i >= 0; i--){
    *l = (*l) | arr[i];
    if (i != 0){
        *l = (*l) << 8;
    }
}
}

void MessageList::setLastClockValue(){
    this->makeLong(&this->clockValue, &this->packetData[2]);
}

unsigned long MessageList::getLastClockValue(){
    return this->clockValue;
}

uint8_t MessageList::getMessageValue(){
    return this->packetData[0];
}

uint32_t MessageList::getSenderAddressLow(){
    return this->addressLow;
}

uint32_t MessageList::getSenderAddressHigh(){
    return this->addressHigh;
}

uint8_t *MessageList::getData(){
    return this->packetData;
}
```

I.4. Κώδικας για την καταγραφή αποκλίσεων ρολογιών σε σχέση με ακριβές ρολόι

```
int MessageList::getListSize(){
    return this->bigDataSize;
}

int MessageList::getMessageCount(){
    return this->messageCount;
}

int MessageList::getDataSize(){
    return this->dataSize;
}
```

I.4 Κώδικας για την καταγραφή αποκλίσεων ρολογιών σε σχέση με ακριβές ρολόι

Κώδικας για την πλευρά του υπολογιστή:

```
import serial
import time

if __name__ == "__main__":
    arduino = serial.Serial(port='COM5', baudrate=9600, timeout=.1)
    ts = time.time()
    while True:
        data = arduino.readline()
        te = time.time()
        if data != b'':
            dt = te-ts
            ts = time.time()
            print(dt)
```

Κώδικας για την πλευρά του κόμβου:

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("starting");  
}
```

```
void loop() {  
  delay(10000);  
  Serial.println("0");  
}
```
