ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
Τμήμα Ηλεκτρονικών Μηχανικών Τ.Ε.

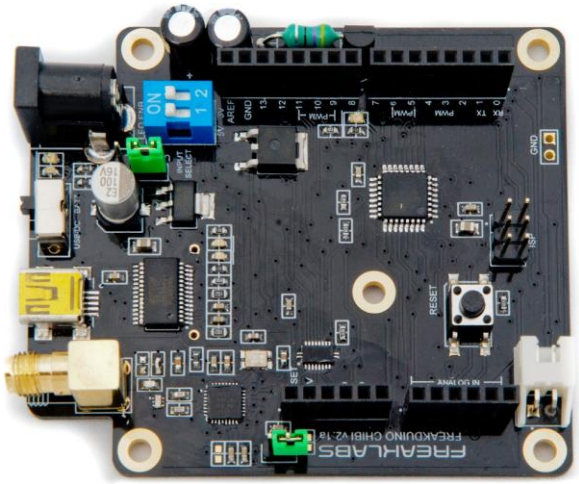
PIRAEUS UNIVERSITY of APPLIED SCIENCES
FACULTY OF ENGINEERING
Department of Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in
Internetworked Electronic Systems

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Δικτύου Ασύρματων Αισθητήρων με Ενσωματωμένη Τεχνολογία Ευφυών Κεραιών



Μεταπτυχιακός Φοιτητής: Νιτσιούλης Ευάγγελος, Α.Μ. 0005

Επιβλέπων: Μυτιληναίος Στέλιος, Επίκουρος Καθηγητής

ΑΙΓΑΛΕΩ, ΣΕΠΤΕΜΒΡΙΟΣ 2017

ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
Τμήμα Ηλεκτρονικών Μηχανικών Τ.Ε.

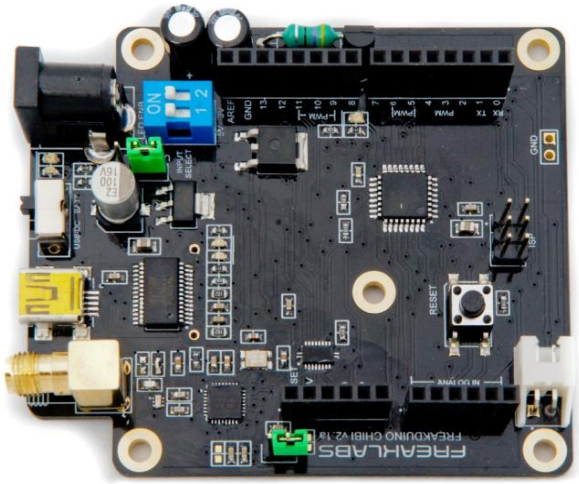
PIRAEUS UNIVERSITY of APPLIED SCIENCES
FACULTY OF ENGINEERING
Department of Electronics Engineering

Πρόγραμμα Μεταπτυχιακών Σπουδών
Διαδικτυωμένα Ηλεκτρονικά Συστήματα

Master of Science in
Internetworked Electronic Systems

MSc Thesis

Wireless Sensor Network Development with Embedded Intelligent Antennas Technology



Student: Nitsioulis Evangelos, Reg. Nr. 0005

MSc Thesis Supervisor: Mitilineos Stelios, Assistant Professor

ATHENS-EGALEO, SEPTEMBER 2017

ΠΕΡΙΛΗΨΗ

Τα ασύρματα δίκτυα αισθητήρων (WSNs) έχουν προσελκύσει μεγάλο ερευνητικό ενδιαφέρον τα τελευταία χρόνια, με πιθανές εφαρμογές, καθιστώντας τα ιδανικά για την ανάπτυξη του οραματισμένου κόσμου της πανταχού παρούσας και διαδεδομένης πληροφορικής. Οι περιορισμοί της ενεργειακής και της υπολογιστικής απόδοσης είναι τα κύρια βασικά ζητήματα κατά την αντιμετώπιση αυτού του τύπου δικτύων. Η κύρια ερευνητική προσπάθεια έχει επικεντρωθεί στη δρομολόγηση και την κατανεμημένη επεξεργασία προκειμένου να επιτευχθεί καλύτερη ποιότητα επικοινωνίας (QoS), χαμηλότερες παρεμβολές και ρυθμός κατανάλωσης ενέργειας ενώ πραγματοποιείται η διάδοση των δεδομένων. Η ενσωμάτωση έξυπνων κεραιών σε ασύρματους κόμβους αισθητήρων προτείνεται εδώ ως εναλλακτική και καινοτόμος προσέγγιση στο φυσικό στρώμα με λύσεις σε προβλήματα που παρουσιάζονται σε παραδοσιακά προβλήματα που αντιμετωπίζουν οι τρέχουσες αρχιτεκτονικές WSN. Μελετώντας τη συμπεριφορά των WSNs που αποτελούνται από διαφορετικούς τύπους κεραιών (έξυπνες και πολυκατευθυντικές), παρουσιάστηκαν ευνοϊκά αποτελέσματα που βελτιώνουν τη λειτουργία των δικτύων αυτού του τύπου.

Παρατίθεται η μελέτη και η πιλοτική ανάπτυξη δικτύων 3 και 5 κόμβων ασύρματων αισθητήρων, με πολυκατευθυντικές κεραίες, και παρουσιάζονται επίσης τεχνικές δρομολόγησης πακέτων πληροφορίας μεταξύ των κόμβων. Η προβολή δεδομένων για την ποιότητα του δικτύου ασύρματων αισθητήρων με χρήση έξυπνων κεραιών (QoS) είναι βασικό εργαλείο για την επίβλεψη του δικτύου και είναι διαθέσιμο προς τον χρήστη. Αναπτύσσεται επίσης η δυνατότητα οδήγησης συγκεκριμένων λοβών ακτινοβολίας των κεραιών στην περίπτωση που μελλοντικά τοποθετηθούν έξυπνες κεραίες.

Λέξεις-Κλειδιά: Ασύρματα Δίκτυα Αισθητήρων, Έξυπνες Κεραίες, Λοβός Ακτινοβολίας, Ποιότητα Επικοινωνίας, Arduino

ABSTRACT

Wireless Sensor Networks (WSNs) have attracted a great deal of research interest over recent years, with potential applications, making them ideal for the development of the envisioned world of ubiquitous and widespread computing. The limitations of energy and computing performance are the main key issues in dealing with this type of network. The main research effort has focused on routing and distributed processing to achieve better QoS, lower interference and energy consumption while data is being disseminated. The integration of smart antennas into wireless sensor nodes is proposed here as an alternative and innovative approach to the physical layer with solutions to problems encountered in traditional current WSN architectures. By studying the behavior of WSNs consisting of different types of antennas (smart and multi-directional), favorable results have been shown to improve the operation of such networks.

The study and pilot development of a 3- and a 5-nodes network of wireless sensors with multi-directional antennas are also presented. Packet information routing techniques between the nodes, along with QoS parameters, are used in this study in order to obtain better network surveillance. The capability of driving specific lobes of radiation of the antennas is also analysed, for the case of future deployment of networks with smart antennas.

Keywords: Wireless Sensor Networks, Smart Antennas, Radiation Lobes, Quality of Communication, Arduino

Περιεχόμενα

ΕΙΣΑΓΩΓΗ:

Αντικείμενο, ερευνητικά ερωτήματα και διάρθρωση της εργασίας

Εισαγωγικά στοιχεία για τα Ασύρματα Δίκτυα Αισθητήρων

Ένα δίκτυο ασύρματων αισθητήρων μπορεί να οριστεί ως ένα δίκτυο συσκευών, που χαρακτηρίζονται ως κόμβοι, που μπορούν να ανιχνεύσουν το περιβάλλον και να μεταδώσουν τις πληροφορίες που συλλέγονται από το υπό παρακολούθηση πεδίο (π.χ. περιοχή) μέσω ασύρματων συνδέσεων [1], [2], [4]. Τα δεδομένα προωθούνται, ενδεχομένως μέσω πολλαπλών αναπηδήσεων (hops) δηλαδή μέσω πολλαπλών επόμενων παραληπτών, σε ένα τελικό κόμβο (sink node), που μπορεί να τα χρησιμοποιήσει τοπικά ή είναι συνδεδεμένο με άλλα δίκτυα (π.χ. το Διαδίκτυο) μέσω μιας πύλης. Οι κόμβοι μπορούν να σταθούν ή να κινηθούν. Μπορούν να γνωρίζουν την τοποθεσία τους ή όχι. Μπορούν να είναι ομοιογενείς ή όχι. Η ποικιλία πιθανών εφαρμογών των WSN στον πραγματικό κόσμο είναι ουσιαστικά απεριόριστη, από την περιβαλλοντική παρακολούθηση , την υγειονομική περίθαλψη, την τοποθέτηση και τον εντοπισμό, την υλικοτεχνική υποστήριξη, τον εντοπισμό κλπ.

Είναι σημαντικό να υπογραμμιστεί ότι η εφαρμογή επηρεάζει σημαντικά την επιλογή της ασύρματης τεχνολογίας που πρόκειται να χρησιμοποιηθεί. Μόλις καθοριστούν οι απαιτήσεις εφαρμογής, ο σχεδιαστής πρέπει να επιλέξει την τεχνολογία που επιτρέπει την ικανοποίηση αυτών των απαιτήσεων. Για το σκοπό αυτό, η γνώση των χαρακτηριστικών, των πλεονεκτημάτων και των μειονεκτημάτων των διαφόρων τεχνολογιών είναι θεμελιώδης.

Λόγω της μεγάλης ποικιλίας πιθανών εφαρμογών των WSN, οι απαιτήσεις του συστήματος θα μπορούσαν να αλλάξουν σημαντικά. Για παράδειγμα, σε περίπτωση εφαρμογών περιβαλλοντικής παρακολούθησης, συνήθως κυριαρχούν οι ακόλουθες απαιτήσεις:

- Ενεργειακή απόδοση: κόμβοι που τροφοδοτούνται με μπαταρία ή έχουν περιορισμένη παροχή ρεύματος,
- Χαμηλό ποσοστό δεδομένων: συνήθως η ποσότητα δεδομένων που πρέπει να ανιχνευθεί είναι περιορισμένη,

- Επικοινωνία μονής κατεύθυνσης: οι κόμβοι λειτουργούν μόνο ως αισθητήρες και ως εκ τούτου η ροή δεδομένων είναι από κόμβους με αισθητήρες (περιφερειακούς) προς κόμβους ελέγχου (κεντρικότερους) και όχι αντίστροφα,
- Ασύρματη συνδεσιμότητα: συνήθως σε εφαρμογές περιβαλλοντικής παρακολούθησης δεν υπάρχουν διαθέσιμες ενσύρματες συνδέσεις για τη σύνδεση ενός κόμβου ελέγχου στο σταθερό δίκτυο.

Σημαντικά διαφορετικές είναι οι απαιτήσεις μιας τυπικής βιομηχανικής εφαρμογής, όπου οι ασύρματοι κόμβοι χρησιμοποιούνται σε αντικατάσταση καλωδιακής σύνδεσης:

- Αξιοπιστία: η επικοινωνία πρέπει να είναι ανθεκτική σε αστοχία και παρεμβολές.
- Ασφάλεια: η επικοινωνία πρέπει να είναι ισχυρή σε σκόπιμες επιθέσεις.
- Υψηλό ποσοστό δεδομένων: η διαδικασία που θα παρακολουθείται συνήθως μεταφέρει ένα μεγάλο όγκο δεδομένων.
- Αμφίδρομη επικοινωνία: οι περιφερειακοί κόμβοι με τους αισθητήρες στην περίπτωση των βιομηχανικών εφαρμογών λειτουργούν συνήθως και ως ενεργοποιητές (actuators) και ως εκ τούτου η επικοινωνία μεταξύ του κόμβου ελέγχου και των περιφερειακών κόμβων πρέπει να είναι εγγυημένη και αμφίδρομη.
- Ενσύρματη συνδεσιμότητα: οι τελικοί κόμβοι ελέγχου μπορούν να συνδεθούν απευθείας στο σταθερό δίκτυο χρησιμοποιώντας ενσύρματες συνδέσεις.

Ακόμη και αν οι απαιτήσεις εξαρτώνται έντονα από την εφαρμογή, ένα από τα σημαντικότερα ζητήματα στο σχεδιασμό των WSNs, ειδικά σε σενάρια όπου η διαθεσιμότητα ισχύος είναι περιορισμένη, είναι η ***ενεργειακή αποδοτικότητα***. Υψηλή ενεργειακή απόδοση σημαίνει μεγάλη διάρκεια ζωής του δικτύου και περιορισμένο κόστος εγκατάστασης και συντήρησης του δικτύου.

Η ενεργειακή αποδοτικότητα μπορεί να επιτευχθεί σε διαφορετικά επίπεδα ξεκινώντας από το επίπεδο της τεχνολογίας (π.χ. υιοθετώντας συστατικά υλικού (hardware) χαμηλής κατανάλωσης) και προχωρώντας στην επιλογή πρωτοκόλλων φυσικού επιπέδου, επιπέδου MAC, επιπέδου δρομολόγησης (routing) μέχρι και το επίπεδο της εφαρμογής (application).

Για παράδειγμα, σε φυσικό επίπεδο και σε επίπεδο MAC, οι κόμβοι θα μπορούσαν να λειτουργούν με χαμηλό κύκλο λειτουργίας (duty cycle), ξοδεύοντας το μεγαλύτερο μέρος του χρόνου τους σε λειτουργία ύπνου (sleep mode), για εξοικονόμηση ενέργειας. Αυτό δημιουργεί

νέα προβλήματα, όπως π.χ. ότι οι κόμβοι μπορεί να μην «ξυπνούν» ταυτόχρονα, λόγω των παρασυρόμενων τοπικών ρολογιών, καθιστώντας έτσι αδύνατη την επικοινωνία. Κατάλληλα συστήματα συγχρονισμού δικτύου είναι υποχρεωτικά στην περίπτωση αυτή, [5].

Τα κύρια χαρακτηριστικά των WSNs, όπως θα μπορούσε να συναχθεί από τη γενική περιγραφή που δόθηκε προηγουμένως, είναι: η δυνατότητα επέκτασης σε σχέση με τον αριθμό των κόμβων στο δίκτυο, η αυτο-οργάνωση, η αυτο-θεραπεία, η ενεργειακή αποδοτικότητα, ο επαρκής βαθμός συνδεσιμότητας μεταξύ κόμβων, η χαμηλή πολυπλοκότητα, το χαμηλό κόστος και το μέγεθος των κόμβων. Η κατάλληλη επιλογή συνδυασμού δικτυακής *αρχιτεκτονικής* και δικτυακού *πρωτοκόλλου* συνιστά μία *τεχνική λύση* για την ανάπτυξη του WSN. Οι τεχνικές λύσεις που παρέχουν τέτοιες δυνατότητες, δηλαδή εξασφαλίζουν τα προαναφερθέντα κύρια χαρακτηριστικά, μπορούν να θεωρηθούν ως ένα πιθανό πλαίσιο για την υλοποίηση ενός WSN στην πράξη. Δυστυχώς, η επιλογή μιας τέτοιας αρχιτεκτονικής και πρωτοκόλλου, δηλαδή μίας συγκεκριμένης τεχνικής λύσης προς υλοποίηση, δεν είναι καθόλου απλή διαδικασία, γι΄αυτό και η έρευνα στο πεδίο αυτό είναι ακόμα σήμερα σε πλήρη εξέλιξη διεθνώς, [3].

Η δομή της εργασίας

Η παρούσα διπλωματική εργασία ασχολείται με τα Ασύρματα Δίκτυα Αισθητήρων και προτείνει την αξιοποίηση έξυπνων κεραιών για τη βελτίωση διάφορων χαρακτηριστικών λειτουργίας τους. Η μελέτη ολοκληρώνεται με πιλοτική υλοποίηση δύο ασυρμάτων δικτύων με απλούς κόμβους κατηγορίας mini-PCs για την επιβεβαίωση των αποτελεσμάτων.

Το κύριο ερώτημα που διερευνά η εργασία αφορά την ικανότητα των ευφυών κεραιών να βελτιώσουν την ποιότητα επικοινωνίας και να μειώσουν την παρεμβολή σε ένα ασύρματο δίκτυο αισθητήρων, διατηρώντας παράλληλα την ενεργειακή αυτονομία των κόμβων.

Η δομή και τα περιεχόμενα της εργασίας είναι τα εξής:

Στο πρώτο κεφάλαιο γίνεται επισκόπηση των Ασυρμάτων Δικτύων Αισθητήρων και των τεχνολογιών που χρησιμοποιούνται σήμερα για την κατασκευή τους.

Στο δεύτερο κεφάλαιο θα περιγραφεί η ανάπτυξη ενός δικτύου ασύρματων κόμβων και η λειτουργία αυτών. Θα επεξηγηθεί αναλυτικά η συνδεσιμότητα των κόμβων και πώς τελικά επιτυγχάνεται η επικοινωνία μεταξύ τους.

Στο τρίτο κεφάλαιο θα γίνει αναφορά για την δρομολόγηση πακέτων πληροφορίας με συγκεκριμένα μοτίβα και θα περιγραφεί ο τρόπος που επιλέγεται κατάλληλος λοβός επικοινωνίας σύμφωνα με την θέση του κάθε κόμβου.

Στο τέταρτο κεφάλαιο θα πραγματοποιηθεί επίδειξη ενός Δικτύου Ασύρματων Αισθητήρων 3 κόμβων και εν συνεχεία 5 κόμβων με αναλυτική επεξήγηση στην λειτουργία του και σε όλες τις διαδικασίες δρομολόγησης αλλά και επιλογής λοβού ακτινοβολίας,

Τέλος η εργασία κλείνει με τα Συμπεράσματα και τις προοπτικές περαιτέρω έρευνας του θέματος.

ΚΕΦΑΛΑΙΟ 1:

Ασύρματα Δίκτυα Αισθητήρων

1.1 Εισαγωγικά στοιχεία

Τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks, WSNs) είναι μια κατηγορία κατανεμημένων υπολογιστικών και επικοινωνιακών συστημάτων που αποτελούν μέρος του φυσικού χώρου που τοποθετούνται. Αν και διαθέτουν περιορισμένη υπολογιστική ισχύ και χαμηλούς ενεργειακούς πόρους, το πιο ενδιαφέρον χαρακτηριστικό τους είναι η συνεργατικότητα και η αλληλεπίδρασή τους με τον κόσμο που τους περιβάλλει. Οι πρόσφατες εξελίξεις στο πεδίο αυτό πυροδότησαν την ανάπτυξη των WSNs των οποίων η λειτουργικότητα βασίζεται στη συνεργατική προσπάθεια ενός μεγάλου αριθμού μικρών, χαμηλού κόστους, χαμηλής ισχύος, πολυλειτουργικών κόμβων αισθητήρων που είναι σε θέση να επικοινωνούν ελεύθερα σε μικρές αποστάσεις. Επιπλέον, μηχανισμοί που υπολογίζουν την ακριβή θέση των κόμβων δεν κρίνονται απαραίτητοι. Αυτό συνεπάγεται τυχαία ανάπτυξη σε εχθρικά περιβάλλοντα ή σε επιχειρήσεις αντιμετώπισης καταστροφών, ένα μοναδικό χαρακτηριστικό για τα δίκτυα αυτού του τύπου. Τα έξυπνα περιβάλλοντα αντιπροσωπεύουν το επόμενο βήμα εξέλιξης στην κατασκευή, τις επιχειρήσεις κοινής ωφέλειας, τη βιομηχανία, το σπίτι, το πλοίο και τα συστήματα αυτοματοποίησης των μεταφορών. Αυτή η γέφυρα προς τον φυσικό κόσμο επέτρεψε την ανάπτυξη μιας πλειάδας δυνητικών υπηρεσιών, από την υγεία μέχρι τις στρατιωτικές επιχειρήσεις και την ασφάλεια, όπως ο έλεγχος του περιβάλλοντος, η παρακολούθηση των φυσικών περιβάλλοντων, η παρακολούθηση οχημάτων και κυκλοφορίας, παρακολούθηση υγείας ασθενών, παρακολούθηση και επίβλεψη κτηρίων και βιομηχανικών εγκαταστάσεων κ.ά.

Από την άλλη πλευρά, τα WSNs εμφανίζουν κάποια ιδιαίτερα χαρακτηριστικά, όπως περιορισμούς ισχύος, συχνά μεταβαλλόμενη τοπολογία, ευαισθησία και χαμηλή μνήμη, ενώ η αρχιτεκτονική τους απαιτεί πρωτόκολλα και αλγόριθμους με αυτο-οργανωτικές δυνατότητες. Επιπλέον, στις περισσότερες περιπτώσεις ένα WSN αποτελείται από έναν μεγάλο αριθμό πυκνοτοποθετημένων κόμβων, πράγμα που σημαίνει ότι οι γειτονικοί κόμβοι μπορεί να είναι

πολύ κοντά ο ένας στον άλλο. Η επικοινωνία τους αναμένεται να παράγει υψηλά επίπεδα παρεμβολών και κατανάλωσης ενέργειας. Ένας από τους σημαντικότερους περιορισμούς των κόμβων αισθητήρων προέρχεται από τις περιορισμένες πηγές ενέργειας. Επομένως, ενώ τα παραδοσιακά δίκτυα στοχεύουν στην επίτευξη υψηλών προδιαγραφών ποιότητας υπηρεσιών (QoS), τα πρωτόκολλα WSN πρέπει να επικεντρώνονται κυρίως στην ελαχιστοποίηση της καταναλισκόμενης ισχύος. Σε αυτό το πλαίσιο, ένα βασικό ζήτημα σχεδίασης σχετίζεται με τη διερεύνηση παραμέτρων του συστήματος, όπως το μέγεθος του δικτύου και η πυκνότητα των κόμβων, όσον αφορά τις μετρήσεις του συστήματος, συμπεριλαμβανομένης της χωρικής κάλυψης, της απόδοσης, της καθυστέρησης, της διάρκειας ζωής του δικτύου, της ενεργειακής απόδοσης και της αξιοπιστίας, που αναφέρθηκαν.

Οι περισσότερες προσεγγίσεις στα παραπάνω θέματα επικεντρώνονται στη βελτιστοποίηση δρομολόγησης και το σχεδιασμό πρωτοκόλλου. Πολλοί ερευνητές έχουν αναπτύξει σχέδια που πληρούν τις απαιτήσεις που περιγράφονται παραπάνω, προτείνοντας πρωτόκολλα και αλγόριθμους για WSNs. Η ποιότητα της υπηρεσίας (QoS) μπορεί να μετρηθεί από την άποψη της ενεργειακής απόδοσης ή του βέλτιστου αριθμού των αισθητήρων που στέλνουν πληροφορίες ανά πάσα στιγμή. Σε αυτή τη περίπτωση, μηχανισμοί ελέγχου QoS έχουν υποβληθεί για να προσαρμόσουν την ανάλυση QoS, επεκτείνοντας έτσι τη διάρκεια ζωής του δικτύου και τη διαχείριση της εξάντλησης της ενέργειας.

Η ελαχιστοποίηση της παρεμβολής των κόμβων είναι μία από τις κύριες προκλήσεις των WSNs. Η υψηλή παρεμβολή αυξάνει την πιθανότητα σύγκρουσης πακέτων, η οποία με τη σειρά της επηρεάζει την απόδοση και την κατανάλωση ενέργειας. Διάφοροι μηχανισμοί ελέγχου τοπολογίας έχουν αναπτυχθεί για την ελαχιστοποίηση της καταναλισκόμενης ενέργειας σε WSNs αλλά χωρίς να είναι σε θέση να εγγυηθούν ρητά χαμηλές παρεμβολές. Από την άλλη πλευρά, οι έξυπνες κεραίες έχουν χρησιμοποιηθεί εκτενώς στη βιβλιογραφία για συμβατικά συστήματα επικοινωνιών και η χρήση τους επεκτείνεται διαρκώς λόγω της αποδεδειγμένης ευεργετικής επίδρασής τους στις επιδόσεις των ασύρματων επικοινωνιών. Οι ευφυείς κεραίες έχουν προταθεί προκειμένου να ικανοποιηθεί η ζήτηση για υψηλά ποσοστά δεδομένων σε ορισμένους χρήστες, διατηρώντας παράλληλα ένα υψηλό επίπεδο QoS για τους συμβατικούς χρήστες. Έχουν χρησιμοποιηθεί επίσης για να αμβλύνουν την παρεμβολή και την καθυστέρηση της εξάπλωσης, να αυξήσουν την χωρητικότητα του συστήματος και τη φασματική απόδοση κ.λπ. Επιπλέον, έχουν προταθεί ποικίλα μοντέλα διάδοσης και πολλαπλής πρόσβασης, π.χ. με

επαναχρησιμοποίηση της θέσης, εκτίμηση της κατεύθυνσης άφιξης σήματος, εντοπισμό θέσης κλπ.

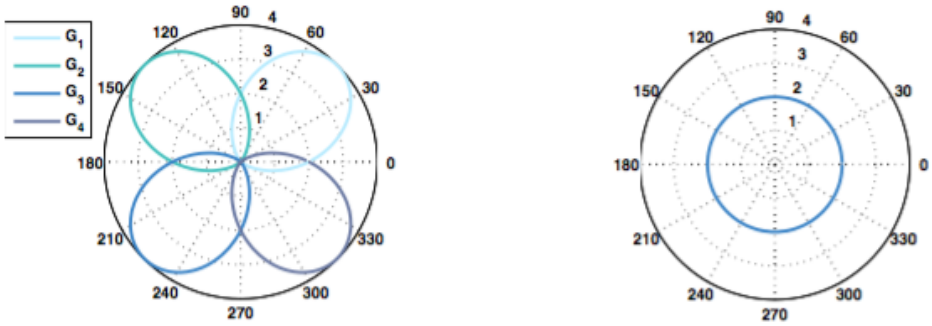
Μια διαφορετική προτεινόμενη προσέγγιση συνιστάται στο να εξοπλιστούν οι κόμβοι WSN με έξυπνες κεραίες, εφαρμόζοντας ορισμένες μικρές αλλαγές στις διαδικασίες επιλογής και δρομολόγησης κόμβων, προκειμένου να αντιμετωπιστούν τα εγγενή μειονεκτήματα τέτοιων δικτύων με ένα αποτελεσματικό εργαλείο στο φυσικό στρώμα.

1.2 Μοντέλα Ασύρματων δικτύων αισθητήρων

Αυτή η ενότητα παρέχει μια επισκόπηση της προτεινόμενης προσέγγισης και του γενικού πλαισίου επί του οποίου βασίζεται η προσομοίωση, μαζί με τις κύριες παραδοχές που σχετίζονται με τη μοντελοποίηση μας. Επιπλέον, εισάγονται οι βασικοί μηχανισμοί που καθορίζουν τη λειτουργία του δικτύου.

1.2.1 Ορισμός Δικτύου

Οι έξυπνες κεραίες μπορούν να αντικαταστήσουν τις πολυκατευθυντικές κεραίες στους κόμβους WSN, καθώς μπορούν να μεταδίδουν εντός της συνολικής περιοχής κάλυψης μίας πολυκατευθυντικής, αλλά με ένα κέρδος κατεύθυνσης που εξαρτάται από τη δέσμη που ενεργοποιείται σε κάθε χρονική στιγμή [6]. Αυτό σημαίνει ότι η σχετική γωνιακή θέση του ζεύγους κόμβων προέλευσης - προορισμού θα καθορίσει τη δέσμη που θα ενεργοποιηθεί για κάθε κόμβο, εξυπηρετώντας είτε τις εκπομπές είτε τις λήψεις. Εν ολίγοις, η επικοινωνία 2 κόμβων εξαρτάται από την γωνία μεταξύ τους. Ανάλογα με αυτή, θα επιλεχθεί ο λοβός ακτινοβολίας εκείνος που παρατηρείται το μεγαλύτερο κέρδος για τη συγκεκριμένη μετάδοση. Αυτό δεν μπορεί να συμβεί στην περίπτωση της ομοικατευθυντικής κεραίας η οποία εκμπέμπει με σταθερό κέρδος προς όλες τις κατευθύνσεις. Εφόσον βέβαια χρησιμοποιείται κάποιος λοβός για εκπομπή ή λήψη δεδομένων, δεν μπορεί να χρησιμοποιηθεί ταυτόχρονα για άλλη λήψη ή εκπομπή μιας και θεωρείται ήδη κατειλημμένος. Με άλλα λόγια, η χρησιμοποίηση των λοβών ακτινοβολίας μπορεί να μειώσει την παρεμβολή και να περιορίσει τις περιοχές σύγκρουσης σε πιο πυκνά δίκτυα αισθητήρων πράγμα που δεν συμβαίνει στην περίπτωση των πολυκατευθυντικών κεραιών (Εικόνα 1.1). Ωστόσο, το κέρδος θα διαφέρει με τη γωνία, αφού γενικά εξαρτάται από την πραγματική γωνία μεταξύ κόμβων προέλευσης και προορισμού.



(a) 4-beam Smart Antennas (b) Omnidirectional Antennas

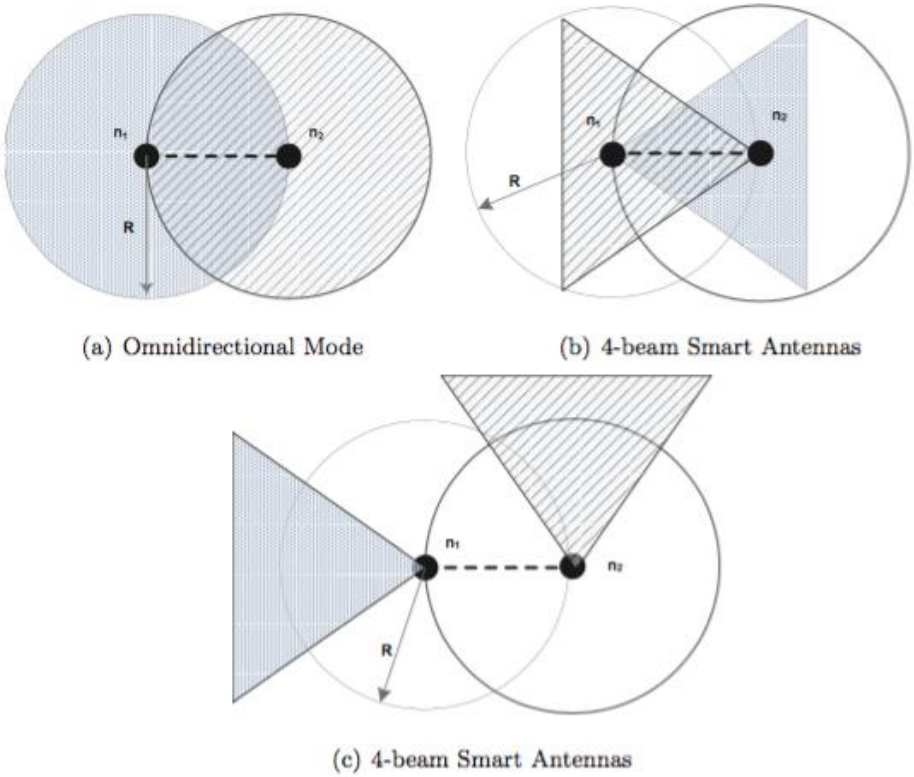
Εικόνα 1.1: Διαγράμματα ακτινοβολίας έξυπνης κεραίας 4 λοβών ακτινοβολίας και πολυκατευθυντικής. Στην περίπτωση (a) φαίνονται οι διαφορετικοί λοβοί ανάλογα με τη γωνία ενώ στη (b) δεν υπάρχει αυτή η δυνατότητα.

Στην παραπάνω Εικόνα 1.1 φαίνεται στην περίπτωση των έξυπνων κεραιών πως το κέρδος μεταβάλλεται από τη γωνία που σχηματίζεται μεταξύ των κόμβων, καθώς ο λοβός που θα επιλεχθεί ποικίλλει.

1.2.2 Τρόποι λειτουργίας δικτύου

Οι τρόποι λειτουργίας που θα αναφερθούν θα είναι δύο: (a) Με κόμβους που διαθέτουν πανκατευθυντικές κεραίες και (b) με κόμβους που διαθέτουν έξυπνες κεραίες, με δυνατότητα δηλαδή επιλογής λοβού ακτινοβολίας για τις ανάγκες λήψης και εκπομπής [6].

- **Τρόπος Λειτουργίας με Πανκατευθυντική Κεραία** (Omnidirectional Antennas): Οι κόμβοι είναι εξοπλισμένοι με πολυκατευθυντικές κεραίες. Σε αυτόν τον τρόπο λειτουργίας, το κέρδος του πομπού και του δέκτη είναι ίσο το ένα με το άλλο, δηλαδή G1 = G2. Έτσι, κάθε σύνδεση γίνεται αμοιβαία.
- **Τρόπος Λειτουργίας με Έξυπνη Κεραία** (Smart Antenna): Οι κόμβοι είναι εξοπλισμένοι με έξυπνες κεραίες. Σε αυτόν τον τρόπο, το κέρδος του πομπού και του δέκτη δεν είναι το ίδιο. Προκειμένου να πραγματοποιηθεί μετάδοση, πρέπει να βρούμε τη γωνία μεταξύ δύο κόμβων που πρόκειται να επικοινωνήσουν, έστω n_1 και n_2. Στη συνέχεια, για τη γωνία αυτή, βρίσκουμε την ακτίνα που πρόκειται να ενεργοποιηθεί για κάθε κόμβο, όταν n_1 είναι ο πομπός και n_2 ο δέκτης, έτσι ώστε να υπολογιστεί ανάλογα το κέρδος μετάδοσης και λήψης. Τέλος, εξετάζουμε αν η ισχύς λήψης είναι πάνω ή κάτω από το όριο (ευαισθησία του δέκτη) και θεωρούμε επιτυχημένη τη ζεύξη.



Εικόνα 1.2: Παρεμβολές μεταξύ ενεργών κόμβων.

Η κύρια διαφορά μεταξύ αυτών των δύο τρόπων λειτουργίας έγκειται στη διαμόρφωση των περιοχών σύγκρουσης, δηλαδή τομείς που καταλαμβάνουν οι ενεργές μεταδόσεις.

Η Εικόνα 1.2 δείχνει τον τρόπο με τον οποίο οι κόμβοι αλληλεπιδρούν μεταξύ τους και σχηματίζουν περιοχές κάλυψης κατά την προσπάθεια μετάδοσης. Στην Εικόνα 1.2(α), η κοινή περιοχή κάλυψης ορίζεται ως η τομή των κυκλικών λοβών, ενώ η ένωση τους ορίζει την περιοχή εμπλοκής, στην οποία δεν είναι δυνατή η μετάδοση από κάθε άλλο κόμβο. Η Εικόνα 1.2(b) δείχνει την συνολική περιοχή κάλυψης κατά τη διάρκεια μίας συνεχιζόμενης μετάδοσης μεταξύ των κόμβων n_1 και n_2 όταν χρησιμοποιούν έξυπνες κεραίες και συνεπώς ενεργοποιείται μόνο ένας από τους 4 λοβούς. Η Εικόνα 1.2(c) δείχνει ένα διαφορετικό παράδειγμα, όπου ο κόμβος n_1 δεν μεταδίδει προς το n_2, αλλά ανταλλάσσει πληροφορίες με έναν άλλο κόμβο εντός του τομέα που καλύπτεται από την ενεργοποιημένη δέσμη του (κατευθυνόμενη προς τα αριστερά). Από το σχήμα είναι φανερό πως αν και οι κόμβοι n_1 και n_2 μπορούν να επικοινωνήσουν κατευθείαν μεταξύ τους, η δυνατότητα αυτή δεν επισκιάζεται στην περίπτωση που μιλούν και με άλλους κόμβους την ίδια χρονική στιγμή. Η σχετική γωνία είναι

διαφορετική σε αυτή τη περίπτωση για κάθε κόμβο και έτσι δεν υπάρχει αλληλοκάλυψη στα διαγράμματα ακτινοβολίας στην περίπτωση των έξυπνων κεραιών. Στην Εικόνα 1.2(α), το n_2 παραμένει πάντοτε εντός της περιοχής του n_1 και επομένως αποκλείεται από τη μετάδοση υπό την προϋπόθεση ότι ο n_1 αποστέλλει προς το παρόν δεδομένα σε έναν τρίτο κόμβο, εφόσον το κανάλι θεωρείται απασχολημένο.

1.2.3 Βασικές παράμετροι δικτύου

Η μέτρηση της αποδοτικότητας του δικτύου στηρίζεται στα ακόλουθα δεδομένα:

1. Ποιότητα Υπηρεσίας (QoS): ο αριθμός των πακέτων που μεταδόθηκαν στο δίκτυο προς τον συνολικό αριθμό πακέτων που δημιουργήθηκαν (συνολικό φορτίο δικτύου).
2. Αποδοτικότητα: ο αριθμός των μηνυμάτων που παραδόθηκαν επιτυχώς από την πηγή στον προορισμό προς το συνολικό φορτίο δικτύου που δημιουργήθηκε κατά τη διάρκεια της προσομοίωσης.
3. Ποσοστό ενεργών κόμβων, A (%): ο μέσος αριθμός κόμβων που επιτρέπεται να εκπέμπουν μέσα στην ίδια χρονική περίοδο χωρίς να παρεμποδίζονται λόγω παρεμβολών που προκαλούνται από συνεχή κίνηση στο δίκτυο. Ένα μικρό ποσοστό των ενεργών κόμβων αντιστοιχεί σε περισσότερες συγκρούσεις, γεγονός που μειώνει την απόδοση του δικτύου.

Οι προηγούμενοι όροι εκφράζονται καλύτερα σύμφωνα με τα παρακάτω ποσοτικά μεγέθη:

1. Πυκνότητα κόμβου D (κόμβοι ανά τετραγωνικό μέτρο)
2. Παράμετρος λ της στατιστικής κατανομής Poisson
3. Ισχύς μετάδοσης Pt

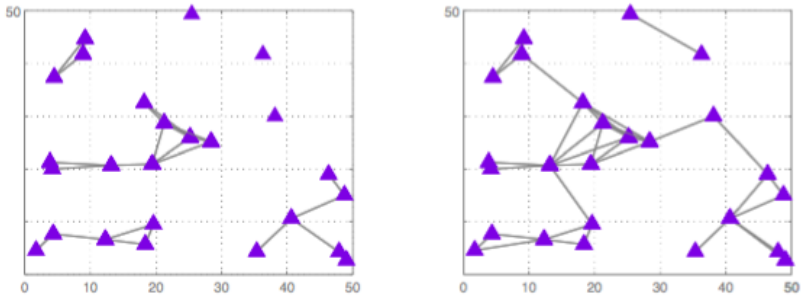
Όσον αφορά την πυκνότητα D των κόμβων, είναι εξ ορισμού ο αριθμός των κόμβων που αναπτύσσονται εντός της περιοχής του δικτύου προς την συνολική περιοχή ανάπτυξης. Η παράμετρος λ καθορίζει το ρυθμό με τον οποίο παράγονται τα πακέτα κατά τη διάρκεια κάθε χρονικής περιόδου [6].

Θα πρέπει να προσέξουμε τη μέγιστη τιμή που μπορεί να πάρει αυτή η παράμετρος, δεδομένου ότι αν αφήσουμε τον αριθμό των πακέτων να αυξηθεί αυθαίρετα και με υψηλό ρυθμό, τα αποτελέσματα δεν θα είναι αντιπροσωπευτικά της απόδοσης του δικτύου. Τέλος, με τον όρο *ισχύς μετάδοσης* αναφερόμαστε στην ισχύ μετάδοσης κάθε κόμβου, η οποία ισχύει για

ολόκληρο το δίκτυο, δεδομένου ότι υποθέτουμε ότι το δίκτυο είναι ομοιογενές, δηλαδή κάθε κόμβος παρουσιάζει ταυτόσημα χαρακτηριστικά.

1.2.4 Τοπολογία Δικτύου

Η τοπολογία του δικτύου και ο τρόπος διαφοροποίησης των τρόπων λειτουργίας μπορεί να εξηγηθεί καλύτερα μέσω γραφικού παραδείγματος (Εικόνα 1.3). Ένα μοντέλο δικτύου και η κατάσταση του μετά την προσθήκη των ζεύξεων παρουσιάζεται, έχοντας πρώτα τους κόμβους που μεταδίδουν πολυκατευθυντικώς (Εικόνα 1.3(a)) και στη συνέχεια χρησιμοποιώντας έξυπνες κεραίες (Εικόνα 1.3(b)). Η διαφορά τους έγκειται στον αριθμό των συνδέσεων που προστέθηκαν, καθιστώντας τους κόμβους εξοπλισμένους με έξυπνες κεραίες ικανούς να επικοινωνούν σε μεγαλύτερες αποστάσεις, καθώς το κέρδος είναι υψηλότερο προς κάθε κατεύθυνση. Αυτό εξηγεί γιατί το γράφημα γίνεται πυκνότερο, σε σχέση με το σύνολο των ακμών του, όταν βρίσκεται στη λειτουργία έξυπνων κεραιών.



(a) Omnidirectional Mode (b) Smart Antennas

Εικόνα 1.3: Τοπολογία δικτύου (a) με πολυκατευθυντικές κεραίες και (b) με έξυπνες κεραίες.

1.3 Αξιολόγηση μετρήσεων με βάση τις παραμέτρους δικτύου

Η ανάλυσή των δικτύων πραγματοποιείται σε σχέση με την πυκνότητα κόμβων και την παράμετρο λ, λαμβάνοντας παράλληλα υπόψη την Ποιότητα Υπηρεσίας (QoS), την απόδοση και το ποσοστό ενεργών κόμβων.

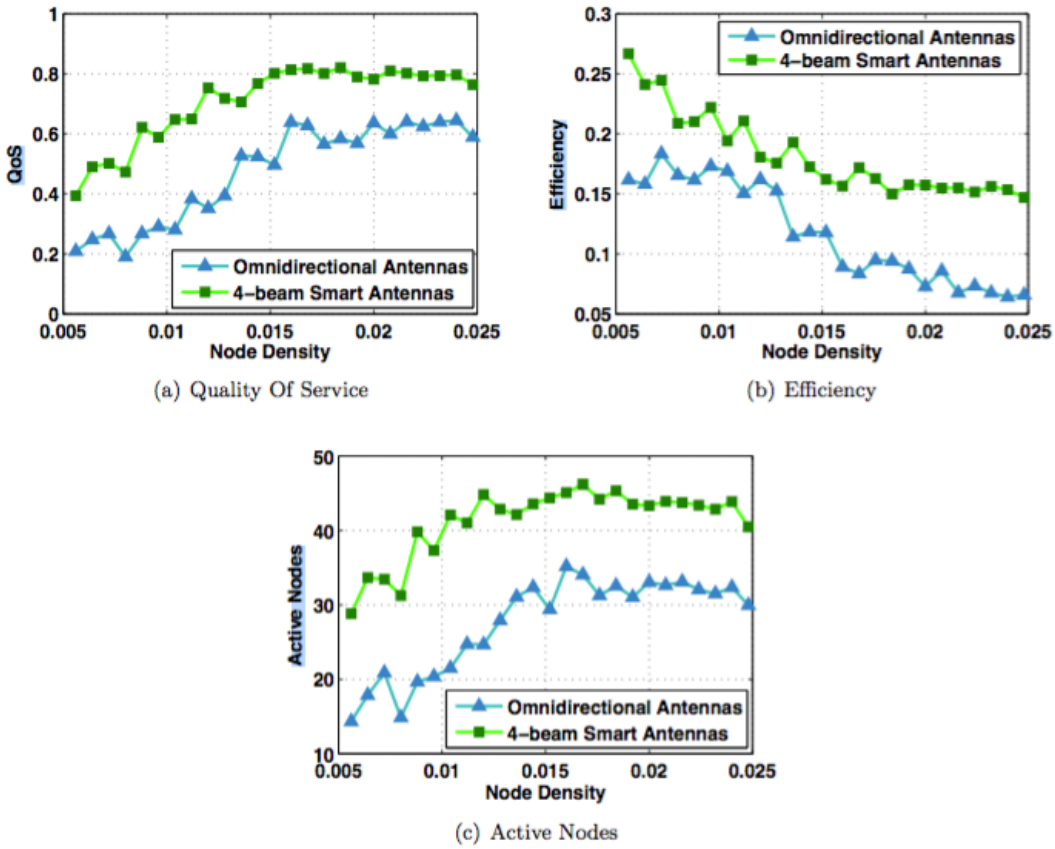
1.3.1 Πυκνότητα Κόμβων δικτύου

Η πυκνότητα κόμβων D διαδραματίζει ζωτικό ρόλο στην αποτελεσματικότητα του δικτύου, καθώς αποτελεί καθοριστικό παράγοντα τόσο για τις περιοχές σύγκρουσης όσο και για τις συντομότερες διαδρομές που χρησιμοποιούνται στη διάδοση πληροφοριών. Όσο η πυκνότητα κόμβων αυξάνεται, περισσότεροι κόμβοι βρίσκονται στους ίδιους τομείς και είναι

απενεργοποιημένοι λόγω συνεχών μεταδόσεων. Όπως θα αποδεικνύεται, τα αραιά δίκτυα δεν παρουσιάζουν σημαντική βελτίωση μετά την εγκατάσταση έξυπνων κεραιών, ενώ το αντίθετο φαινόμενο παρατηρείται για πυκνότερα δίκτυα. Οι συγκρούσεις που εντοπίζονται είναι λιγότερες, αλλά καθώς αυξάνεται το μέγεθος του δικτύου, η πυκνότητα συνδέσεων αυξάνεται με υψηλό ρυθμό, εμποδίζοντας έτσι επιτυχείς μεταδόσεις χωρίς συγκρούσεις. Το πρόβλημα επιδεινώνεται όταν οι κόμβοι χρησιμοποιούν πολυκατευθυντικές κεραίες.

Αρχικά, υποθέτουμε ότι το δίκτυο καλύπτει μια τετράγωνη περιοχή της οποίας η έκταση ισούται με 50 x 50 τετραγωνικά μέτρα. Σε αυτόν τον τομέα, αναπτύσσουμε έναν σταθερό αριθμό κόμβων και έτσι ξεκινάμε από την τοποθέτηση ενός κόμβου κάθε 15 μέτρα, ο οποίος αντιστοιχεί σε πυκνότητα κόμβου 0.005 κόμβων ανά τετραγωνικό μέτρο. Στη συνέχεια αυξάνουμε σταδιακά τον αριθμό των κόμβων που αναπτύσσονται έως ότου υπάρχει περίπου ένας κόμβος κάθε 6 μέτρα (στην περίπτωση αυτή η πυκνότητα είναι 0,025). Δείχνουμε ότι το QoS βελτιώνεται καθώς το δίκτυο γίνεται πιο πυκνό. Αυτό οφείλεται κυρίως στην έλλειψη σύνδεσης που εμφανίζεται σε πιο αραιά δίκτυα. Ωστόσο, αυτή η παράμετρος τείνει να συγκλίνει σε μια σταθερή τιμή καθώς η πυκνότητα του κόμβου ανέρχεται πάνω από 0.015 κόμβους ανά τετραγωνικό μέτρο. Η βελτίωση του QoS με έξυπνες κεραίες σε σχέση με τις πολυκατευθυντικές είναι περίπου 20%, σχεδόν σε κάθε τιμή πυκνότητας, η οποία είναι σημαντική διαφορά, καθώς ενεργοποιούνται περισσότερες μεταδόσεις εντός της ίδιας χρονικής περιόδου.

Επιπλέον, συζητούμε πώς η απόδοση επηρεάζεται από την πυκνότητα των κόμβων.



(a) Quality Of Service (b) Efficiency

(c) Active Nodes

Εικόνα 1.4: Απόδοση δικτύου με βάση την πυκνότητα των κόμβων.

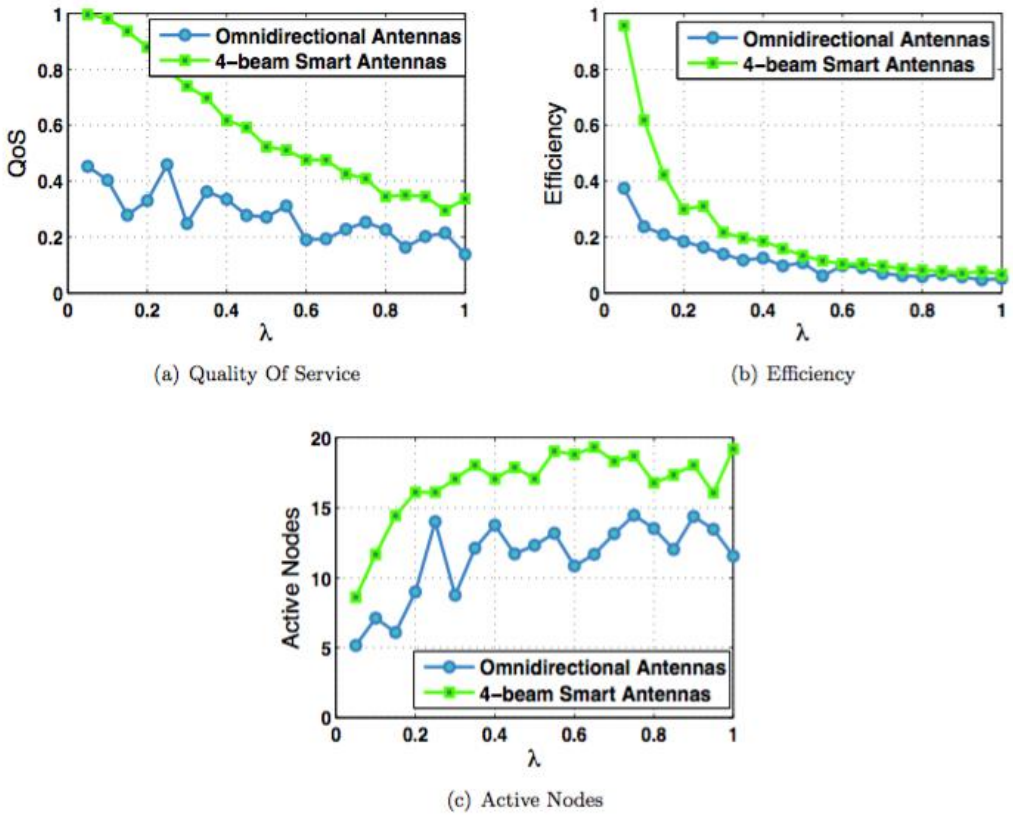
Αξίζει να σημειωθεί ότι, όταν χρησιμοποιούνται πολυκατευθυντικές κεραίες, η αποδοτικότητα του δικτύου πέφτει τόσο σημαντικά όσο και γρήγορα καθώς αυξάνεται η πυκνότητα των κόμβων. Αρχικά, όσο το δίκτυο είναι αραιό, η αποτελεσματικότητα που επιτυγχάνεται είναι υψηλή. Ωστόσο, τα αραιά δίκτυα - που προκαλούν χαμηλή πυκνότητα κόμβων και, συνεπώς, χαμηλή παρεμβολή - δεν ανήκουν στις περιοχές ενδιαφέροντος μας, δεδομένου ότι πρέπει να συνδεθούν με συμβατικά WSNs. Με αυτό, εννοείται ότι το φαινόμενο της ύπαρξης απομονωμένων κόμβων πρέπει να εξαλειφθεί. Αυτό εξασφαλίζεται για τιμές πυκνότητας πάνω από 0,015. Από την άλλη πλευρά, τα δίκτυα που χρησιμοποιούν έξυπνες κεραίες αποκλίνουν από αυτή τη συμπεριφορά και δείχνουν την τάση να διατηρούν τα ποσοστά απόδοσης στα ίδια επίπεδα. Με άλλα λόγια, εγγυώνται ότι τα περισσότερα πακέτα θα παραδοθούν στον προορισμό με επιτυχία, κυρίως λόγω των μειωμένων επιπέδων παρεμβολών. Τέλος, υπάρχει βελτίωση της απόδοσης σε σχέση με τους ενεργούς κόμβους, όπως φαίνεται στην Εικόνα 1.4(c), αφού το ποσοστό των ενεργών κόμβων είναι πάντα υψηλότερο σε σύγκριση με τον περίπτωση των

πολυκατευθυντικών κεραιών. Η διαφορά αυτή κυμαίνεται από 10% και ανέρχεται στο 20% του συνολικού αριθμού κόμβων N. Αυτό το ποσοστό αντιστοιχεί σε 5-10 περισσότερους ενεργούς κόμβους όταν N = 50, 10-20 όταν N = 100 κ.λπ.

1.3.2 Παράμετρος λ της κατανομής Poisson

Επιπλέον, εξετάζεται η απόδοση κάθε τρόπου λειτουργίας σε σχέση με την παράμετρο λ της κατανομής Poisson. Η παράμετρος λ ουσιαστικά αντικατοπτρίζει την «κίνηση» στο δίκτυο. Ένα απασχολημένο δίκτυο, για παράδειγμα, όπου η πληροφορία ρέει συνεχόμενα, εμφανίζει συνεχώς μεγάλη τιμή λ και τείνει να εμφανίζει μια ανεπιθύμητη συμπεριφορά κατά την υπερχείλιση των ουρών πληροφορίας. Από την άλλη πλευρά, τα δίκτυα στα οποία οι πληροφορίες ρέουν σταθερά και σε χαμηλότερους ρυθμούς, τείνουν να παρέχουν καλύτερη ποιότητα υπηρεσιών και να είναι πολύ πιο αποτελεσματικά σε σύγκριση με την προηγούμενη περίπτωση. Έτσι, πρέπει να μελετήσουμε την απόδοση του δικτύου υπό διαφορετικές συνθήκες κυκλοφορίας δικτύου.

Η Εικόνα 1.5 δείχνει τον τρόπο με τον οποίο η παράμετρος λ επηρεάζει την απόδοση του δικτύου και για τις δύο λειτουργίες. Καθώς αυτή η παράμετρος αυξάνεται, η απόδοση επιδεινώνεται και για τις δύο λειτουργίες, ενώ οι καμπύλες που αφορούν τις έξυπνες κεραίες (πράσινες) παρουσιάζουν μεγαλύτερη μείωση από εκείνες που αφορούν τις πολυ-κατευθυντικές (μπλε), κυρίως λόγω των υψηλών τιμών που επιτυγχάνονται με τις έξυπνες κεραίες όταν οι πληροφορίες παράγονται με χαμηλότερους ρυθμούς. Παρόλα αυτά, η απόδοση δικτύων WSN με έξυπνες κεραίες βελτιώνεται σε σχέση με τα δίκτυα WSN που χρησιμοποιούν πολυκατευθυντικές κεραίες σε όλες τις περιπτώσεις. Πιο συγκεκριμένα, η ποιότητα της υπηρεσίας και η αποδοτικότητα λαμβάνουν την τιμή «1» εφόσον η κυκλοφορία δικτύου παραμένει χαμηλή ενώ μειώνεται σημαντικά καθώς τα πακέτα παράγονται με υψηλότερους ρυθμούς.



Εικόνα 1.5: Απόδοση δικτύου με βάση την παράμετρο λ της κατανομής Poisson.

1.4 Κατανάλωση ενέργειας

Αυτή η ενότητα είναι αφιερωμένη σε έναν από τους σημαντικότερους παράγοντες των WSNs, την κατανάλωση ενέργειας. Η κατανάλωση ενέργειας διαδραματίζει βασικό ρόλο στη λειτουργία του δικτύου και πρέπει να λαμβάνεται υπόψη κατά το σχεδιασμό και την κατασκευή ασύρματων κόμβων και αισθητήρων. Με την προϋπόθεση ότι κάθε κόμβος έχει την ίδια ενεργειακή ικανότητα, δηλαδή τις ίδιες ενεργειακές δεξαμενές, ισχύ μετάδοση, χρόνο εξάντλησης ενέργειας, ρυθμό κατανάλωσης ενέργειας κλπ., προκύπτει ότι η μείωση της ισχύος μετάδοσης μπορεί να επηρεάσει όλους τους υπόλοιπους καθοριστικούς παράγοντες ενέργειας. Αυτή η μείωση επιτυγχάνεται με την αύξηση της κατευθυντικότητας μέσω της χρήσης έξυπνων κεραιών. Δεδομένου ότι οι αποστάσεις μεταξύ κάθε ζεύγους κόμβων είναι γνωστές εκ των προτέρων, οι κόμβοι μπορούν να προσαρμόσουν ανάλογα την ισχύ εκπομπής τους. Έτσι, αντί να αυξήσουμε τον αριθμό των συνδέσεων που παράγονται στο δίκτυο διατηρώντας την ισχύ εκπομπής σταθερή, τροποποιούμε το σχέδιο προσομοίωσής μας μειώνοντας την ισχύ μετάδοσης των κόμβων που είναι εφοδιασμένοι με έξυπνες κεραίες. Αυτή η προσέγγιση

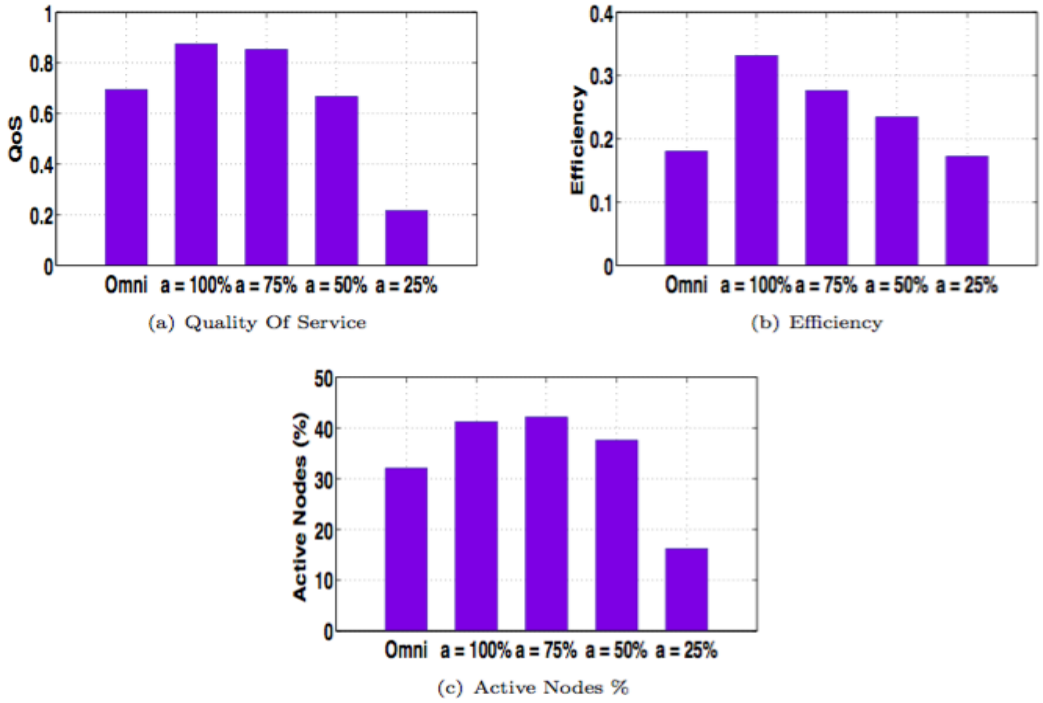
θεωρείται πιο «δίκαιη» όταν συγκρίνουμε κόμβους με έξυπνες κεραίες με κόμβους που διαθέτουν πολυκατευθυντικές κεραίες. Αργότερα σε αυτή την ενότητα εξετάεται η δυνατότητα ρύθμισης της ισχύος μετάδοσης κάθε κόμβου σε σχέση με την καθολική κατώτατη τιμή. Παρά το γεγονός ότι είναι μια δαπανηρή λύση, μπορεί να βελτιώσει την αποδοτικότητα του δικτύου μειώνοντας ταυτόχρονα την παρεμβολή και την κατανάλωση ενέργειας [6].

Δεδομένου ότι η ισχύς μετάδοσης για τα δίκτυα που λειτουργούν με τις πολυκατευθυντικές κεραίες είναι σταθερή, μελετάμε τη συμπεριφορά των δικτύων με έξυπνες κεραίες και εξετάζουμε τις χαμηλότερες τιμές της ισχύος μετάδοσης για διαφορετικές τοπολογίες δικτύου. Ως εκ τούτου, μπορούμε να καταλήξουμε σε συμπεράσματα σχετικά με το σημείο στο οποίο εξομαλύνονται οι διεργασίες και το ποσό της εξοικονόμησης ενέργειας μετά την περίοδο αξιολόγησης. Ας επεξεργαστούμε την Εικόνα 1.6, όπου η παράμετρος α αντιστοιχεί σε ένα κλάσμα της αρχικής ισχύος μετάδοσης, της οποίας η τιμή είναι καθολική στο δίκτυο (δεδομένου ότι το δίκτυο θεωρείται ομοιογενές, κάθε κόμβος έχει την ίδια ικανότητα μετάδοσης). Η ισχύς εκπομπής στην περίπτωση της πολυκατεύθυνσης είναι 10mW και οι περιπτώσεις που αξιολογούνται περιλαμβάνουν δίκτυα που χρησιμοποιούν έξυπνες κεραίες με μειωμένη ισχύ (που κυμαίνονται από 0,01W όπου α = 100% έως 0,0025W αντιστοιχούν σε α = 25%) που αντανακλάται από την παράμετρο α. Όπως αναμενόταν, το πιο αποδοτικό δίκτυο αντιστοιχεί σε α = 100%. Ωστόσο, το σημείο όπου το QoS με κόμβους έξυπνων κεραιών παραμένει στα ίδια επίπεδα σε σύγκριση με τους κόμβους πολυκατευθυντικής μετάδοσης, αντιστοιχεί σε πολύ χαμηλότερη μεταδιδόμενη ισχύ, η οποία ισούται με το 75% των αρχικών τιμών(δηλαδή με των κόμβων με πολυκατεύθυνση). Αυτό είναι κάπως αναμενόμενο, αλλά η εξοικονόμηση ενέργειας είναι θεαματική. Λόγω των υψηλότερων κερδών κάθε δέσμης των έξυπνων κεραιών, η υψηλή ισχύς μετάδοσης οδηγεί σε μεγαλύτερο αριθμό συνδέσεων και κατά συνέπεια σε υψηλότερα επίπεδα παρεμβολών. Με τη μείωση της ισχύος εκπομπής επιτυγχάνουμε τα εξής:

Το QoS είναι περίπου ίσο με το επίπεδο που επιτυγχάνεται όταν οι κόμβοι μεταδίδουν με τη μέγιστη ισχύ. Έτσι, περίπου ο ίδιος αριθμός πακέτων εξυπηρετείται στην ίδια χρονική περίοδο.

Το ποσοστό των ενεργών κόμβων στο ίδιο χρονικό διάστημα είναι μεγαλύτερο κατά σχεδόν 2% σε σύγκριση με την περίπτωση του α = 100% Pt και σχεδόν διπλασιάζεται σε σχέση με τον τρόπο πολυκατεύθυνσης.

Στον αντίποδα, αυτό το δίκτυο δεν είναι τόσο αποτελεσματικό όσο το πρώτο. Η βαριά κυκλοφορία προκαλεί τις περισσότερες ουρές να διατηρούν πακέτα για μεγαλύτερες χρονικές περιόδους και αυτό πιθανόν να οφείλεται στα χαμηλότερα ποσοστά αποδοτικότητας.

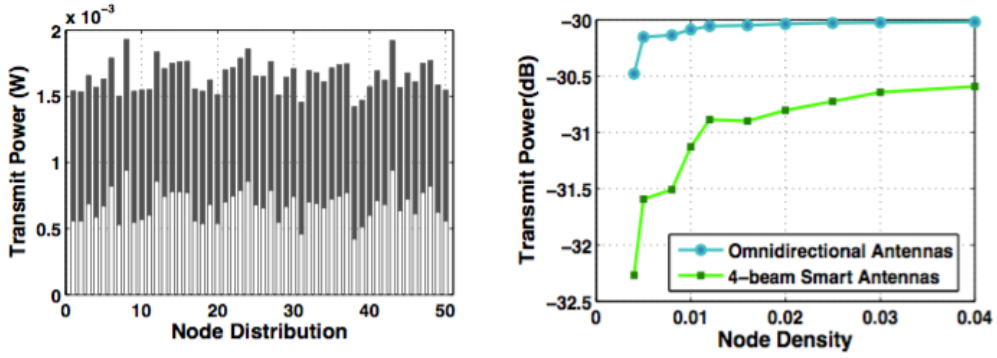


Εικόνα 1.6: Κατανάλωση ενέργειας. Απόδοση δικτύου με βάση την ισχύ εκπομπής.

Η διαδικασία που ακολουθείται σε αυτή την ενότητα διαφέρει από τις προηγούμενες προσεγγίσεις. Σε αυτή την προσπάθεια, ο στόχος μας είναι να διατηρήσουμε τον αριθμό των συνδέσμων ανεπηρέαστο. Παρόλο που χτίζουμε τον πίνακα γειτονικών κόμβων με τον ίδιο ακριβώς τρόπο, τροποποιούμε την ισχύ μετάδοσης για κάθε μεμονωμένο κόμβο για να καθορίσουμε την επιτυχημένη ελάχιστη ισχύ που απαιτείται για τη μετάδοση μεταξύ κάθε γειτονικού ζεύγους κόμβων. Αυτή η τιμή ισχύος είναι σημαντικά χαμηλότερη σε σύγκριση με αυτή που χρησιμοποιείται με πολυκατευθυντικές κεραίες. Για παράδειγμα, ένα πακέτο λαμβάνεται κάτω από – 70dB ενώ το κατώφλι που ορίζεται από τον δέκτη ισούται με – 100dB. η μετάδοση κόμβου θα μπορούσε να εξοικονομήσει πολύτιμη ενέργεια (περίπου 10 – 15dBs σε αυτή την περίπτωση) με τη μείωση της ισχύος μετάδοσης. Η τροποποίηση της ισχύος εκπομπής επιτρέπεται μόνο εάν όλες οι μεταδόσεις για αυτόν τον κόμβο μπορούν να πραγματοποιηθούν με επιτυχία μετά από αυτή την τροποποίηση, η οποία εξασφαλίζεται με ρύθμιση της ισχύος

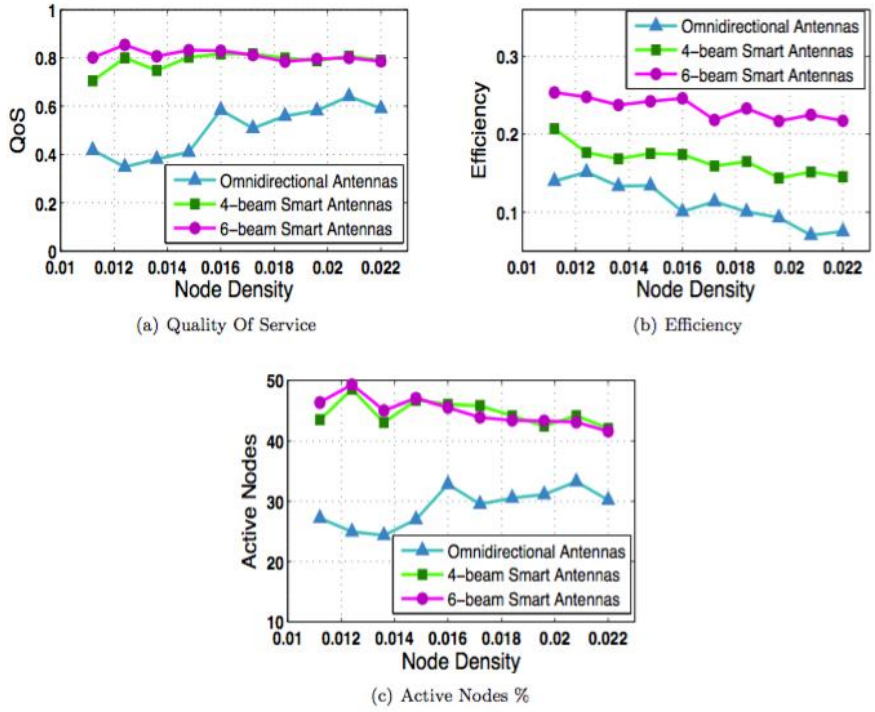
μετάδοσης ίση με την ελάχιστη ισχύ που απαιτείται για κάθε υπάρχοντα σύνδεσμο του κόμβου. Στη συνέχεια, συγκρίνουμε την απαιτούμενη ισχύ μετάδοσης όταν οι κεραίες μεταδίδουν πολυκατευθυντικά έναντι της απαιτούμενης ισχύος όταν οι κόμβοι είναι εξοπλισμένοι με έξυπνες κεραίες που τους επιτρέπουν να μεταδίδουν προς διαφορετικές κατευθύνσεις που εξαρτώνται από τον στόχο για διάφορες τιμές πυκνότητας κόμβων. Υποθέτουμε ότι το δίκτυο είναι ένα αραιό δίκτυο αισθητήρων που αναπτύσσει N = 50 κόμβους τοποθετημένους σε απόσταση περίπου 10 μέτρων από τους γείτονές τους. Η Εικόνα 1.6(a) είναι ένα χαρακτηριστικό παράδειγμα ενός μοναδικού δικτύου όπου κάθε κόμβος έχει διαφορετική ισχύ εκπομπής, προσδιορισμένη όπως εξηγείται παραπάνω. Αυτή η κατανομή εμφανίστηκε από σχεδόν κάθε δίκτυο πανομοιότυπης πυκνότητας κόμβων. Κάθε κόμβος απαιτεί μεγαλύτερη ισχύ εκπομπής στον παραδοσιακό τρόπο λειτουργίας, ο οποίος ξεπερνά το 40% της ισχύος που χρειάζονται οι έξυπνες κεραίες. Αυτή η διαφορά είναι κοντά στα 2 dB, δηλαδή η διαφορά κέρδους μεταξύ των τρόπων λειτουργίας.

Τέλος, εισάγονται δύο ακόμη διαγράμματα στην Εικόνα 1.7. Το αριστερό διάγραμμα (Εικόνα 1.7(a)) δείχνει την απαραίτητη ισχύ ώστε τα σήματα να λαμβάνονται με την απαιτούμενη ελάχιστη ισχύ, ενώ το δεξί διάγραμμα (Εικόνα 1.7(b)) απεικονίζει τη σύγκριση της μέσης ισχύος μετάδοσης σε dB, σε σχέση με την πυκνότητα των κόμβων D στον οριζόντιο άξονα. Οι πολυκατευθυντικές κεραίες παρουσιάζουν μια μέση τιμή κοντά στην ισχύ εκπομπής (– 30dB), ενώ οι έξυπνες κεραίες απαιτούν λιγότερη ενέργεια για να δημιουργήσουν τους ίδιους συνδέσμους στο δίκτυο, με ποσοστό βελτίωσης να κυμαίνεται από 15-30%.



(a) Node Distribution: Transmit Power (b) Average Transmit Power to Node Density

Εικόνα 1.7: Ισχύς εκπομπής – μετάδοσης.



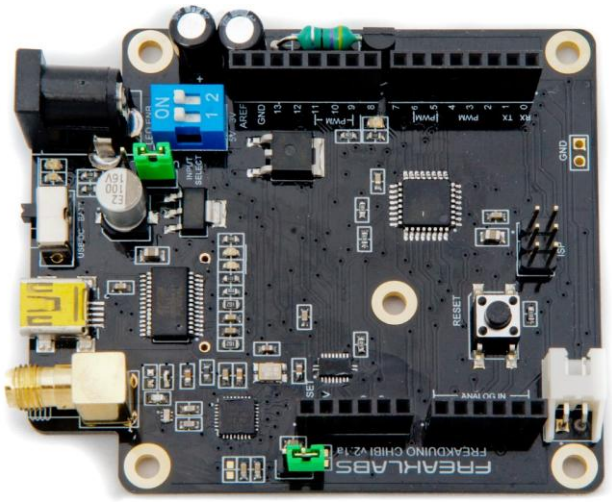
Εικόνα 1.8: Έξυπνες κεραίες: Απόδοση δικτύου με βάση την πυκνότητα κόμβων του δικτύου.

ΚΕΦΑΛΑΙΟ 2:

Πειραματικό μέρος: Ανάπτυξη δικτύου – Κόμβοι – Λειτουργία

2.1 Η βασική μονάδα επεξεργασίας του κόμβου

Για την πειραματική ανάπτυξη πρωτότυπου WSN επιλέχθηκε ως βασική μονάδα κάθε κόμβου το miniPC Freakduino της εταιρίας FreakLabs. Πρόκειται για ένα ευέλικτο και ιδιαίτερα εύχρηστο board, που φαίνεται στην Εικόνα 2.1 με τα βασικά τεχνικά χαρακτηριστικά του.



MCU: ATMega328P
Memory: 32 kB Flash/2 kB RAM
Communications: 802.15.4 wireless, USB
Expansion: Arduino-compatible shield connector
Power: Ext 5VDC, USB, Battery (optional)
Current Consumption: Sleep: ~250 uA, Active: ~40 mA
Options: Ruggedized enclosure, battery regulation circuit, standalone battery case

Εικόνα 2.1: Το Freaklabs Freakduino board και τα κυριότερα τεχνικά χαρακτηριστικά του.

Το Freakduino board της FreakLabs έχει σχεδιαστεί για χαμηλού κόστους και ταχεία ανάπτυξη, αξιολόγηση και αξιοποίηση διαφόρων ασύρματων συσκευών. Συνδυάζει την ευκολία στη χρήση του Arduino IDE (Integrated Developer's Environment) και των διαθέσιμων εργαλείων για την πλατφόρμα του Arduino, συμβατότητα με μια πλούσια ποικιλία από περιφερειακά και τέλος ενσωματωμένο wireless radio module προδιαγραφών πρωτοκόλλου ασύρματης επικοινωνίας IEEE 802.15.4. Το βασικό board έχει όλες τις λειτουργίες του συστήματος

Arduino με ενσωματωμένο wireless radio και είναι ένας ανέξοδος τρόπος για να αρχίσει κανείς να πειραματίζεται με Arduino designs αλλά και φυσικά designs ασύρματων επικοινωνιών.

Διάφορες βελτιώσεις σε σχέση με το απλό Arduino board έχουν προστεθεί σε αυτό το board για την αύξηση της λειτουργικότητας Μερικές από αυτές αναφέρονται παρακάτω [7].

Η προσθήκη ενός ασύρματου δικτύου με βάση το πρωτόκολλο 802.15.4 το οποίο είναι πλέον ενσωματωμένο στην πλακέτα του Freakduino board εκ κατασκευής (ίδιο radio protocol όπως το Xbee) σε φυσικό επίπεδο, επιτρέπει τον ασύρματο έλεγχο συσκευών ή τη συλλογή δεδομένων από ασύρματους αισθητήρες.

Επίσης, προστέθηκε ένα κύκλωμα μπαταρίας έτσι ώστε να μπορεί να λειτουργήσει ως ασύρματη συσκευή χωρίς κανένα καλώδιο ή εξωτερική τροφοδοσία, διευρύνοντας την αυτονομία του.

Ο σχεδιασμός του board έχει επίσης βελτιστοποιηθεί για χαμηλή κατανάλωση ισχύος. Καταναλώνει περίπου 200 μA σε λειτουργία Power Down / sleep στα 3.0V που είναι η τάση που παρέχεται από 2 αλκαλικές μπαταρίες AA, ενώ καταναλώνει περίπου 300 μA στα 2.4V που είναι η τάση που παρέχεται από 2 AA NiMH επαναφορτιζόμενες μπαταρίες. Με την κατάλληλη διαχείριση ενέργειας είναι δυνατόν να έχουμε τη συσκευή ζωντανή για μήνες χρησιμοποιώντας μονάχα ένα ζευγάρι μπαταρίες.

2.2 Επικοινωνία των κόμβων στις ραδιοσυχνότητες (Radio)

Η μονάδα ραδιοεπικοινωνίας (radio module) του κόμβου λειτουργεί στα 2.4 GHz. Διαθέτει έναν RP-SMA connector που είναι μια τυπική υποδοχή κεραίας που συναντάται συνήθως σε Wi-Fi routers. Το πρόγραμμα οδήγησης (driver software) καθώς και το protocol stack είναι πλήρως open source και διατίθεται ως βιβλιοθήκη Arduino. Η έκδοση του protocol stack είναι πλήρως απλουστευμένη σε τρεις κύριες λειτουργίες, οι οποίες είναι: Αρχικοποίηση, αποστολή και λήψη. Έτσι καθιστάται πιο εύκολο να χρησιμοποιηθεί το radio module ώστε οποιαδήποτε συσκευή να μπορεί να «μιλήσει» με οποιαδήποτε άλλη συσκευή εντός της εμβέλειας ακρόασης (η εμβέλεια δίνεται από τα χαρακτηριστικά της συσκευής).

Από την άλλη, η χρήση του πρωτοκόλλου IEEE 802.15.4 προσφέρει διαλειτουργικότητα στην αποστολή, λήψη και γενικότερα διαμεταγωγή πακέτων δεδομένων (data frames). Στην παρούσα έκδοση έχουν προστεθεί οι ακόλουθες χρήσιμες δυνατότητες:

- η αυτόματη επιβεβαίωση (automatic acknowledge) αποστολής πακέτου
- αυτόματη απόρριψη των data frames (auto discard) που δεν ταιριάζουν με τη συγκεκριμένη διεύθυνση δικτύου ή τη διεύθυνση του κόμβου ή περιέχουν κατεστραμένες πληροφορίες
- αυτόματη επανάληψη (auto retry): αν ο δέκτης δεν αναγνωρίζει ότι έλαβε ένα πακέτο ο πομπός θα ξαναδοκιμάσει να στείλει το πακέτο μέχρι κάποιο μέγιστο αριθμό που θα δηλώσει ο χρήστης
- αυτόματο έλεγχο των διευθύνσεων (auto address checking): το radio module από μόνο του πραγματοποιεί ελέγχους για να εξασφαλίσει ότι το πακέτο φτάνει ανέπαφο στον αποδέκτη.

Το chipset που χρησιμοποιεί το συγκεκριμένο board διαθέτει επίσης έναν επιταχυντή υλικού (hardware accelerator) για κρυπτογράφηση AES-128. Το AES-128 είναι ένα ισχυρό πρότυπο κρυπτογράφησης που χρησιμοποιείται από πολλούς κρατικούς φορείς για την εξασφάλιση των επικοινωνιών, ιδιαίτερα όσο αφορά τις ασύρματες επικοινωνίες [7].

2.3 Τροφοδοσία - Ισχύς (Power)

Υπάρχουν τρεις επιλογές για την παροχή ρεύματος στο board. Η πιο κοινή είναι μέσω του USB, ωστόσο μπορεί επίσης να τροφοδοτείται μέσω ενός εξωτερικού τροφοδοτικού που συνδέεται στην υποδοχή DC. Είναι ιδιαίτερα χρήσιμο όταν χρειάζεται περισσότερη ισχύς από όση μπορεί να παρέχει η θύρα USB. Τέλος, όταν δεν υπάρχει διαθέσιμη εξωτερική τροφοδοσία, μπορεί να λειτουργεί με μπαταρίες.

Η θύρα USB παρέχει έως και 500 mA ρεύμα στα 5V και μπορεί να τροφοδοτήσει άμεσα το board για τις περισσότερες εφαρμογές. Είναι βολικό όταν η συσκευή είναι συνδεδεμένη με έναν υπολογιστή, δεδομένου ότι η USB σε σειριακό μετατροπέα επιτρέπει επίσης την επικοινωνία με τον υπολογιστή.

Για σχέδια με υψηλότερες ρευματικές απαιτήσεις, όπως οι κινητήρες οδήγησης ή LED υψηλής ισχύος, μπορεί να χρησιμοποιηθεί ένας εξωτερικός τροφοδότης DC. Το συγκεκριμένο τροφοδοτικό θα πρέπει να είναι τουλάχιστον 6V και κατ' ανώτατο όριο τα 14V. Το board έχει επίσης δύο ξεχωριστές υποδοχές για μπαταρίες στην κάτω πλευρά του.

Η πλακέτα διαθέτει επίσης ένα κύκλωμα ρύθμισης μπαταρίας (battery regulation circuit). Αυτό είναι απαραίτητο γιατί η τάση της μπαταρίας ποικίλει ανάλογα με τον τύπο της μπαταρίας και

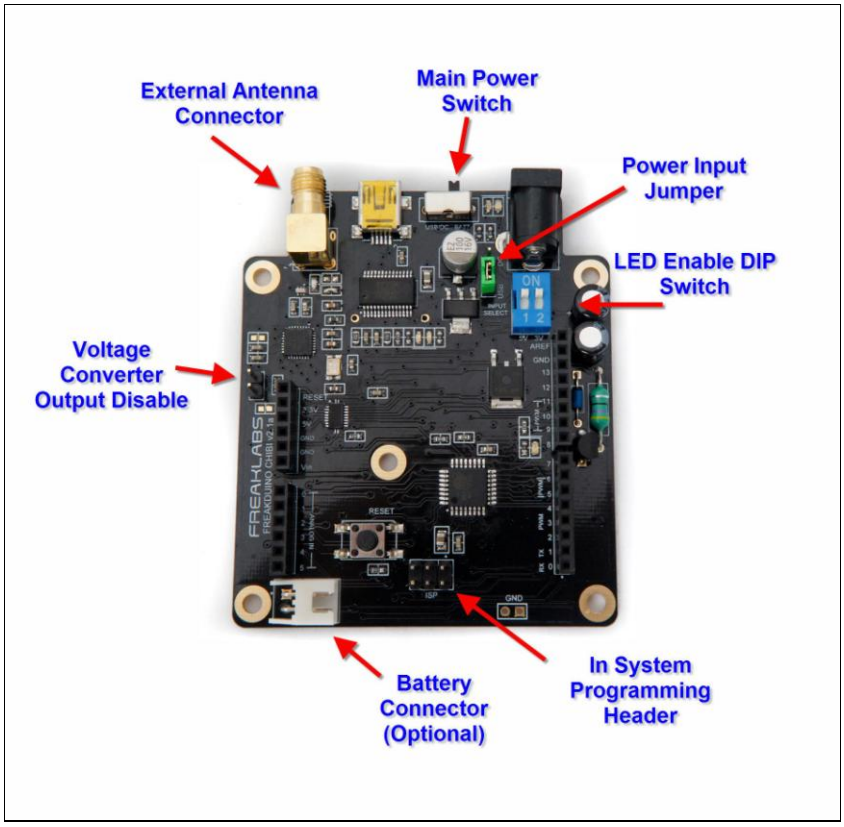
την ποσότητα του φορτίου που έχει μείνει από την φόρτιση. Η ρύθμιση της τάσης δίνει τη δυνατότητα ανεξάρτητα από κάθε τύπο μπαταριών NiMH (αλκαλικών ή επαναφορτιζόμενων) που συνδεθεί με το board, ώστε να εξασφαλιστεί μια σταθερή έξοδος 5V.

Το κύκλωμα ρύθμισης της τάσης εξόδου μπορεί να παρακολουθεί την τάση της μπαταρίας μέσω ενός από τους αναλογικούς ακροδέκτες εισόδου στην MCU, και είναι δυνατόν να αποστείλει ασύρματα μια έγκαιρη προειδοποίηση όταν η μπαταρία αρχίζει να εξαντλείται [7].

2.4 Διασυνδεσιμότητα και επιλογείς (Connectors, Jumpers and Switches)

Η πλακέτα περιέχει μια σειρά από υποδοχές, jumpers και διακόπτες (Εικόνα 2.2):

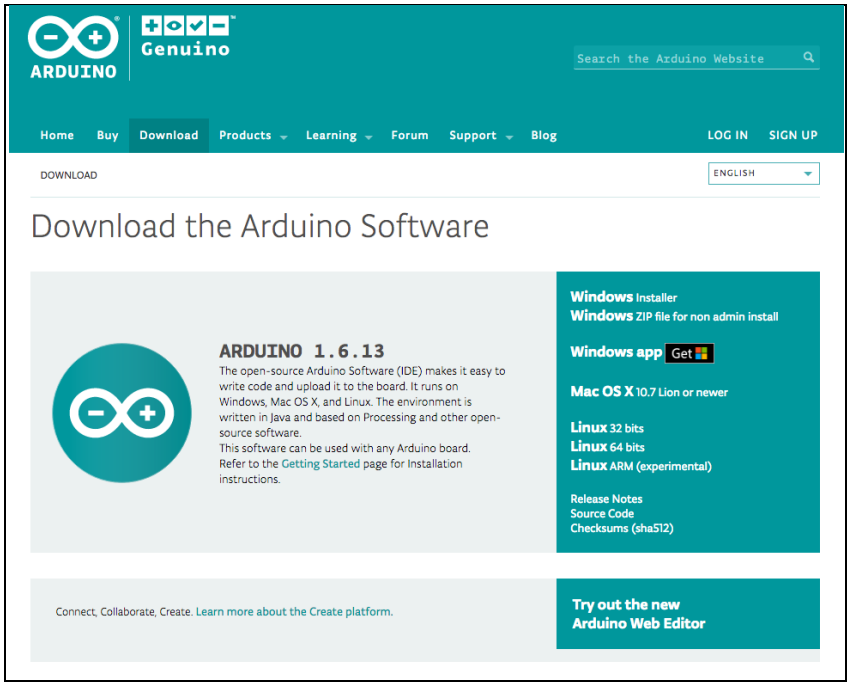
- **Power Input Jumper.** Χρησιμοποιείται για την επιλογή μεταξύ της εξωτερικής τροφοδοσίας DC ή USB
- **Main Power Switch.** Ο κύριος διακόπτης που ορίζει την χρήση τροφοδοσίας DC.
- **LED Enable Switch.** Ο συγκεκριμένος μικροδιακόπτης χρησιμοποιείται για ενεργοποίηση ή απενεργοποίηση των main power LEDs ώστε να αποφευχθεί περιττή κατανάλωση ισχύος στην περίπτωση τροφοδοσίας από μπαταρίες.
- **Υποδοχή Εξωτερικής Κεραίας.** Συνδετήρας τύπου RP-SMA που συνδέεται με μια συμβατή κεραία.
- **Υποδοχή μπαταρίας.**



Εικόνα 2.2: Υποδοχές και διακόπτες πλακέτας Freaklabs Freakduino.

2.5 Εγκαθιστώντας το Arduino IDE

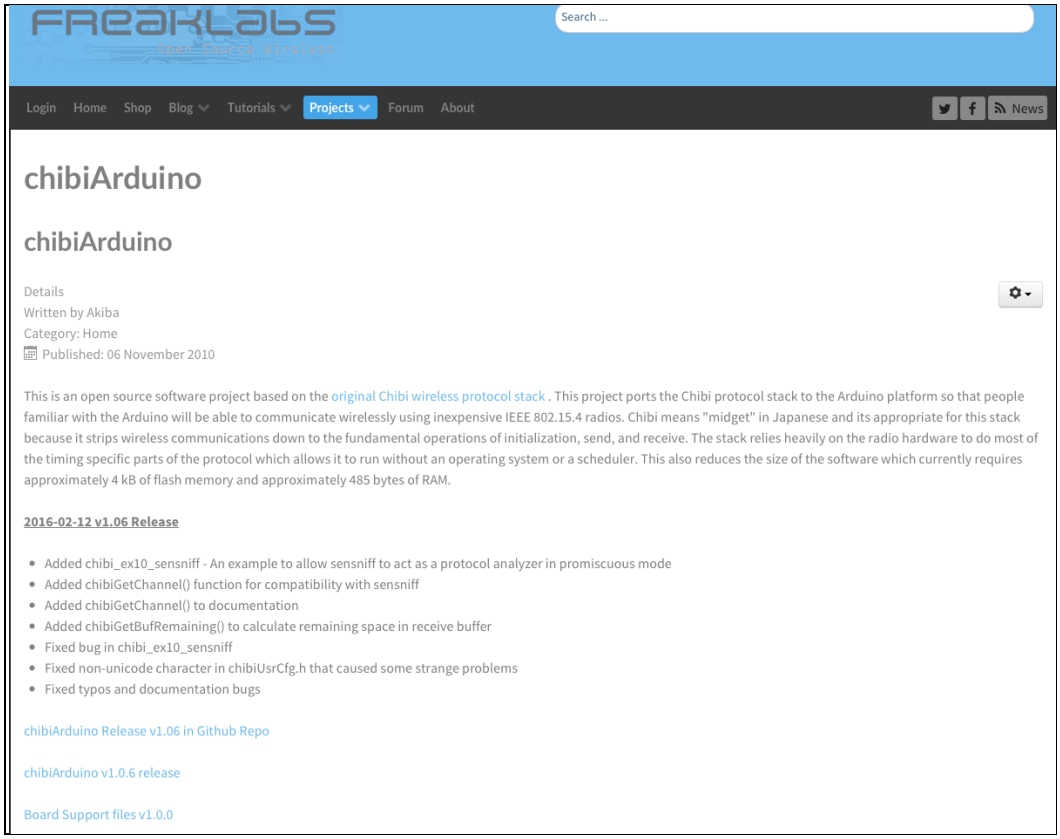
Το Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE) για την οικογένεια των Arduino boards είναι διαθέσιμο στην ηλεκτρονική διεύθυνση https://www.arduino.cc/en/Main/Software (Εικόνα 2.3). Πραγματοποιούμε εγκατάστασή του στον Η/Υ μας, επιλέγοντας τον Windows Installer.



Εικόνα 2.3: Πρόσβαση στο Arduino Software (IDE).

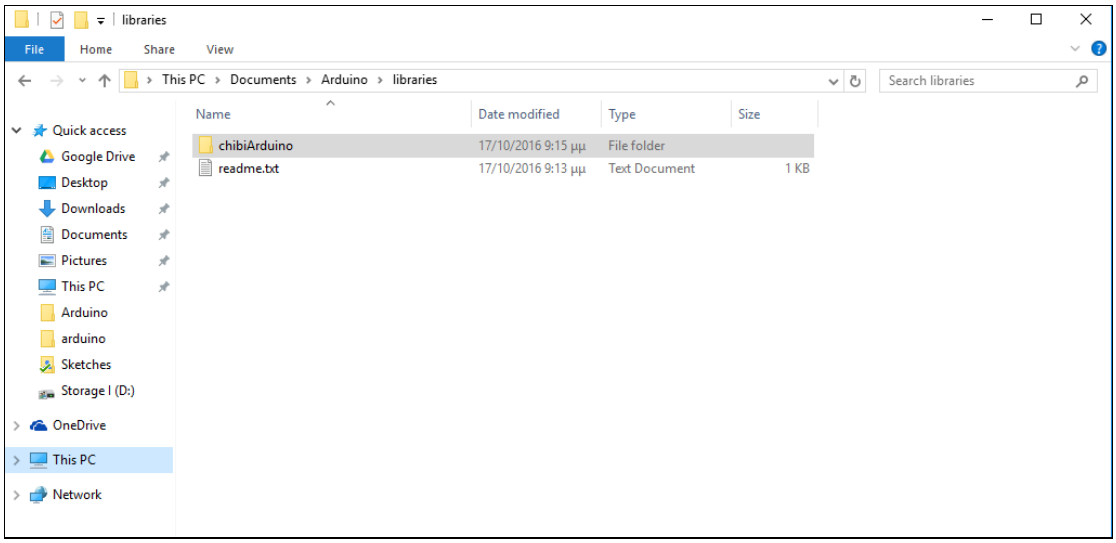
Δημιουργούμε νέο φάκελο με ονομασία «chibiArduino» στον φάκελο εγκατάστασης του Arduino IDE και συγκεκριμένα μέσα στον φάκελο libraries. Μεταφορτώνουμε το 2016-02-12_chibiarduino_v1.06.zip αρχείο στον υπολογιστή. Περιέχει τα απαραίτητα αρχεία για να ξεκινήσουμε το prototyping με το board, κατάλληλα examples για να πειραματιστούμε καθώς και την απαραίτητη βιβλιοθήκη για τις εντολές που θα χρησιμοποιήσουμε (chibi.h). Το αρχείο αυτό μπορεί να βρεθεί στον παρακάτω σύνδεσμο: http://www.freaklabs.org/chibi/2016-02-12_chibiarduino_v1.06.zip

Στην Εικόνα 2.4 φαίνεται η βιβλιοθήκη chibiarduino (http://www.freaklabs.org/index php/chibiarduino.html (Διαθέσιμες Εκδόσεις) [9].



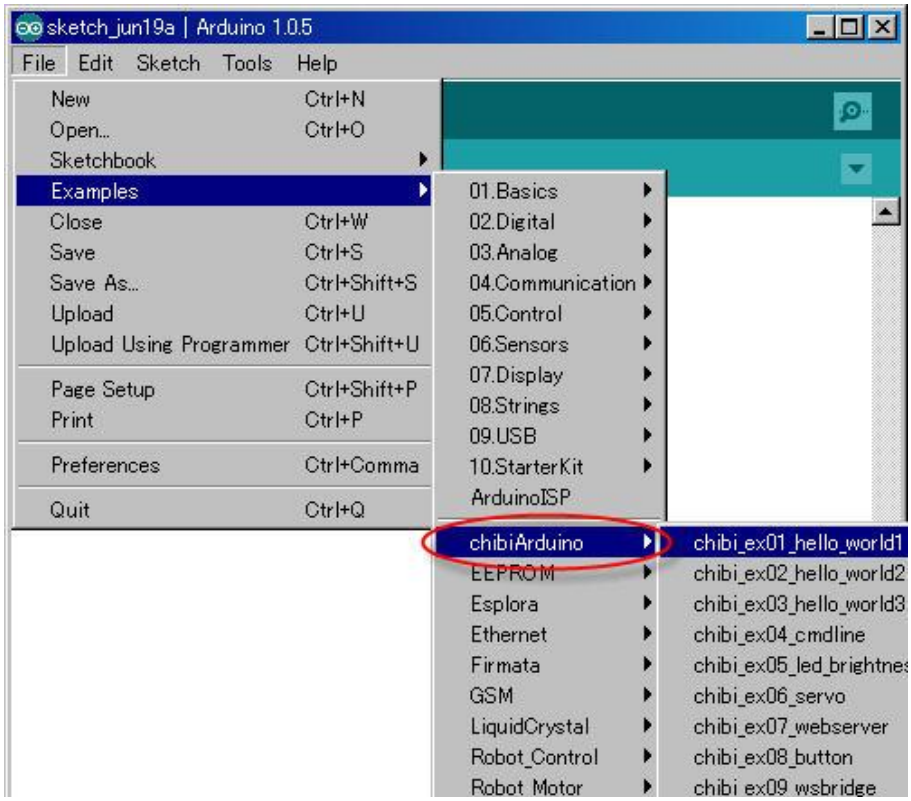
Εικόνα 2.4: Η Βιβλιοθήκη chibiArduino.

Αποσυμπιέζουμε το αρχείο 2016-02-12_chibiarduino_v1.06.zip στον φάκελο «chibiArduino» που δημιουργήσαμε σε προηγούμενο βήμα (Εικόνα 2.5).



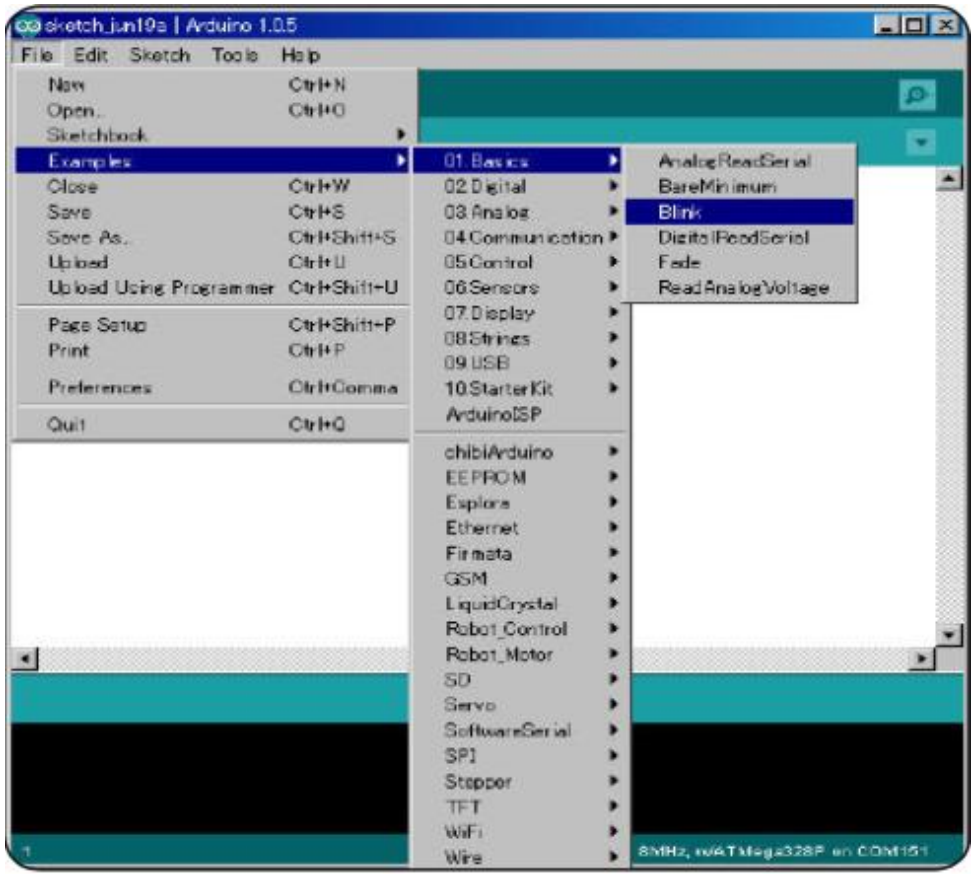
Εικόνα 2.5: Φάκελος εγκατάστασης Arduino IDE.

Η βιβλιοθήκη πρέπει να έχει εγκατασταθεί επιτυχώς. Στην ουσία δεν χρειάζεται κάποια διαδικασία εγκατάστασης, αρκεί η αποσυμπίεση του αρχείου .zip μέσα στον φάκελο chibiArduino που δημιουργήσαμε. Επιβεβαιώνοντας πως έχει γίνει σωστά η αποσυμπίεση, εκκινούμε το Arduino IDE (με διπλό κλικ του εικονιδίου του, που θα έχει εμφανιστεί στην οθόνη μας με το πέρας της εγκατάστασης). Πηγαίνουμε στο μενού File/Examples όπου, εφόσον όλα έχουν γίνει σωστά, θα πρέπει στην επιλογή examples να φαίνεται ο φάκελος chibiArduino με ορισμένα παραδείγματα (Εικόνα 2.6).



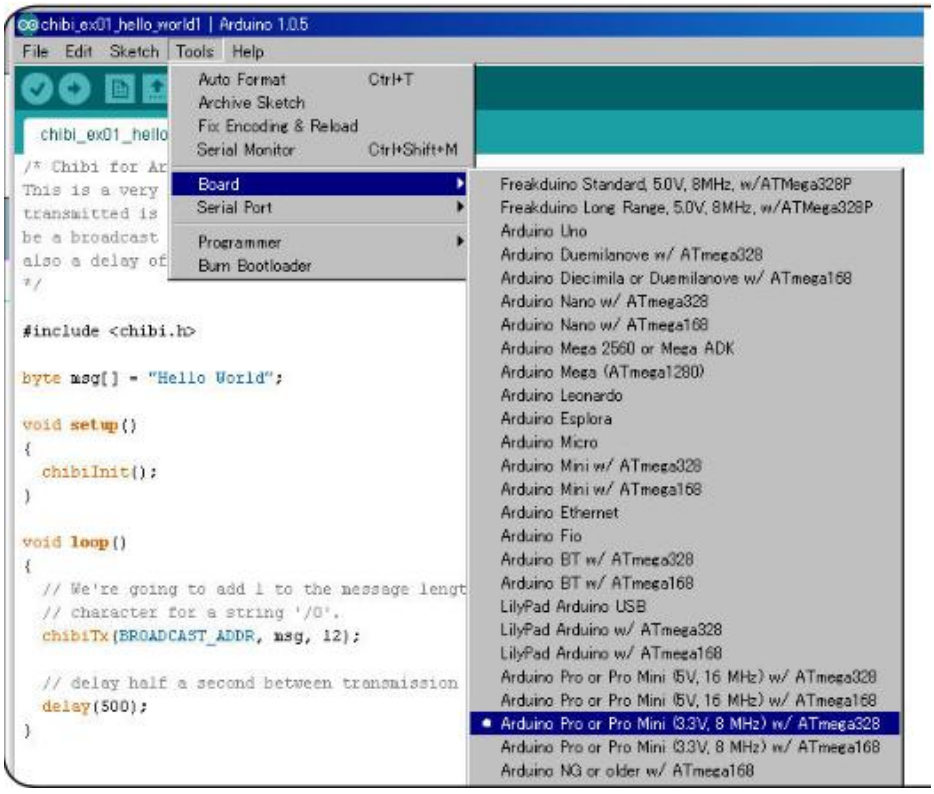
Εικόνα 2.6: Περιβάλλον προγράμματος Arduino IDE.

Ανοίγουμε ένα προϋπάρχον παράδειγμα στον editor του Arduino IDE. Αφού πρώτα έχουμε συνδέσει το chibiArduino board μέσω θύρας USB, πηγαίνουμε στο μενού File/Examples/Basics menu και επιλέγουμε το “Blink” (Εικόνα 2.7).



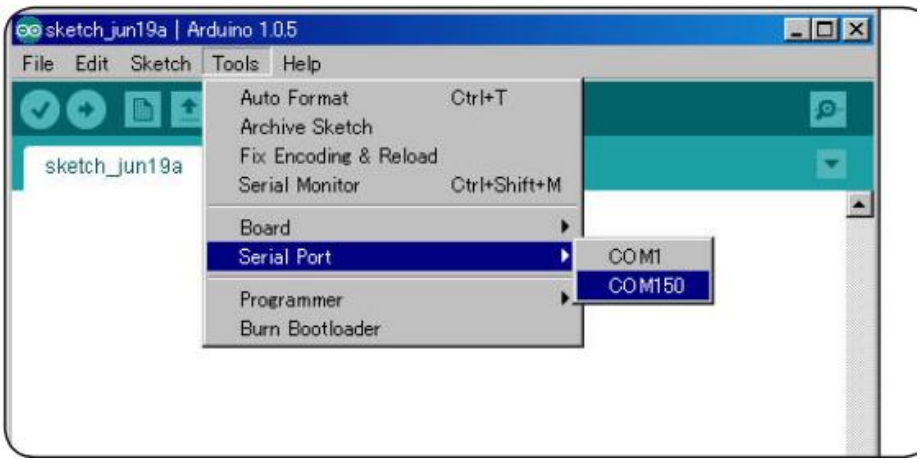
Εικόνα 2.7: Φόρτωση προγράμματος blink.

Ο κώδικας του συγκεκριμένου παραδείγματος εμφανίζεται στην οθόνη μας και συγκεκριμένα μέσα στον κειμενογράφο του Arduino IDE. Ύστερα επιλέγουμε το σωστό board. Πηγαίνουμε στο μενού Tools/Board και επιλέγουμε Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATMega328. Ετσι έχουμε δηλώσει το σωστό board και είμαστε έτοιμοι να περάσουμε τον κώδικα στο Freakduino (Εικόνα 2.8).



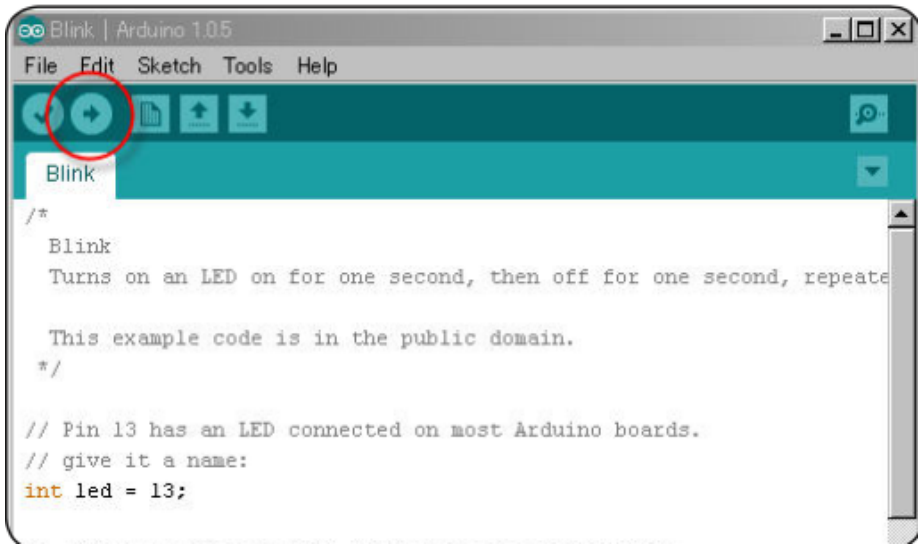
Εικόνα 2.8: Επιλογή κατάλληλου board.

Επιλέγουμε Serial Port. Εφόσον έχουμε συνδέσει ήδη μέσω θύρας USB το Freakduino board, επιλέγουμε σε ποια θύρα (σειριακή επικοινωνία) το συνδέσαμε. Το COM1 θα φαίνεται πιθανότατα εκεί (δεσμευμένη θύρα από το σύστημα, αγνοήστε την), οπότε επιλέγουμε την αμέσως επόμενη διαθέσιμη (Εικόνα 2.9). Προσοχή, αν έχουμε συνδεδεμένα περισσότερα από ένα boards θα φαίνονται εκεί οι σειριακές θύρες που είναι ενεργές, επομένως τόσες όσες και τα board μας.



Εικόνα 2.9: Επιλογή θύρας σειριακής επικοινωνίας.

Το board έχει συνδεθεί επιτυχώς στον υπολογιστή και είμαστε έτοιμοι να περάσουμε τον επιθυμητό κώδικα. Πατάμε το κουμπί Upload. Ο κώδικάς μας θα πρέπει να γίνει επιτυχώς compiled για τυχόν λάθη και να περαστεί επιτυχώς στο Freakduino board (Εικόνα 2.10).



Εικόνα 2.10: Upload κώδικα στο board.

Παρατηρούμε το board με παλλόμενα LEDs και συμπεραίνουμε πως το πρόγραμμα περάστηκε επιτυχώς και τρέχει κανονικά. Να σημειωθεί πως μετά από επιτυχές compilation του κώδικα και upload αυτού, ο κώδικας πλέον «τρέχει» κανονικά χωρίς να χρειάζεται να κάνουμε καμία άλλη ενέργεια.

2.6 Εγκαθιστώντας το πακέτο υποστήριξης για τα διαθέσιμα boards

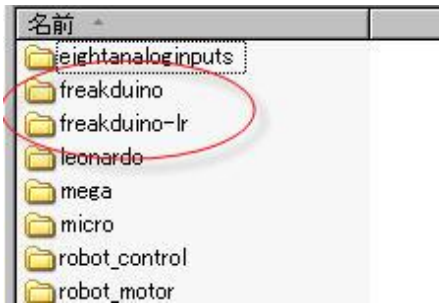
Το συγκεκριμένο βήμα είναι απαραίτητο για την ορθή λειτουργία του πιλοτικού δικτύου 5 κόμβων που υλοποιείται στο πλαίσιο της εργασίας.

Μαζί με την εγκατάσταση των βιβλιοθηκών για το Arduino, συστήνεται να εγκατασταθεί και το πακέτο υποστήριξης για τα διάφορα boards. Αυτό το επιπλέον βήμα καιστά δυνατό το πρόγραμμα να μπορεί να αναγνωρίσει το οποιοδήποτε board συνδεθεί, το οποίο είναι πολύ χρήσιμο από τη στιγμή που υπάρχουν διάφορα Arduino boards με διαφορετικές δυνατότητες.

Αρχικά μεταφορτώνουμε το συμπιεσμένο αρχείο με τις πληροφορίες για τα boards και με όνομα αρχείου Board Support files v1.0.0., το οποίο βρίσκεται στον σύνδεσμο

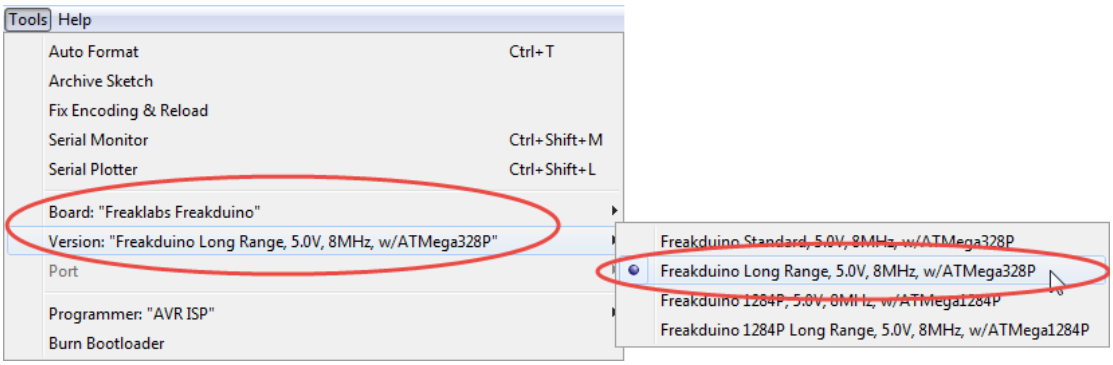
https://archive.freaklabs.org/index.php/home/chibiarduino.html [9]

Μετά την αποσυμπίεση του αρχείου, κατευθυνόμαστε στον φάκελο εγκατάστασης του Arduino. Εκεί θα βρίσκεται ένας φάκελος με το όνομα hardware. Μέσα στον φάκελο μεταφέρουμε τον φάκελο που προέκυψε από την αποσυμπίεση του αρχείου Board Support files v1.0.0. Πλέον, μέσα στον φάκελο πρέπει να υπάρχουν πληροφορίες για τα boards Freakduino και Freakduino long range που θα επιτρέψουν στο Arduino IDE να εισάγει περισσότερες επιλογές στην επιλογή boards από το Tools → Board menu. (Εικόνα 2.11)



Εικόνα 2.11: Φάκελοι με πληροφορίες για boards Freakduino and Freakduino long range.

Πλέον, μέσα από το πρόγραμμα Arduino IDE μπορούμε να επιλέξουμε την κατάλληλη πλακέτα που μας ενδιαφέρει, ενεργοποιόντας παράλληλα όλες τις δυνατότητες αυτής (Εικόνα 2.12).



Εικόνα 2.12: Επιλογή πρωτότυπου board μέσα από το πρόγραμμα Arduino IDE.

2.7 Βασικοί Καταχωρητές (Basic Registers)

Με τον παρακάτω κώδικα είμαστε σε θέση να δούμε τις τιμές που περιέχουν οι βασικοί καταχωρητές του board (Εικόνα 2.13). Ο συγκεκριμένος κώδικας διαθέτει περισσότερο ενημερωτικό χαρακτήρα μιας και οι τιμές των καταχωρητών δηλώνουν π.χ. διευθύνσεις στο δίκτυο, κανάλι ασύρματης επικοινωνίας, κατάσταση συσκευής κλπ.

Ο κώδικας βρίσκεται στο αρχείο
paper/Sketches/transmitter_081216_manual/transmitter_081216_manual.ino

```
#include <chibi.h>
void setup() {
  Serial.begin(57600); //ταχύτητα σειριακής επικοινωνίας
  chibiInit();
 for (int i = 1; i < 48; i++) { //εντολή επανάληψης για 48 φορές
 byte data = chibiRegRead(i);
  Serial.print("Register "); //εντολή εμφάνισης στην κονσόλα
  Serial.print(i); //εντολή εμφάνισης στην κονσόλα
  Serial.print(": "); //εντολή εμφάνισης στην κονσόλα
  Serial.println(data); //εντολή εμφάνισης στην κονσόλα
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Εικόνα 2.13: Κώδικας εμφάνισης περιεχομένου βασικών καταχωρητών.

Δίνονται εδώ λίγες επεξηγήσεις σε σημεία του κώδικα που στην συνέχεια θα ξανασυναντήσουμε. Αρχικά, με το #include <chibi.h> δηλώνουμε ποιά βιβλιοθήκη θα συμπεριληφθεί. Εχουμε ήδη αναφέρει πως σ' αυτή τη βιβλιοθήκη περιέχονται πολλές βασικές εντολές για να λειτουργήσει επιθυμητά το board μας.

Η void setup() έχει χαρακτήρα ρυθμιστικό: είναι μια εντολή η οποία απλά σηματοδοτεί πως ό,τι κώδικας εμπεριέχεται σε αυτή τη συνάρτηση θα εκτελεστεί μόνο μια φορά. Επομένως σε αυτό το κομμάτι «ρυθμίζουμε» την συμπεριφορά του board.

Αντίθετα, η συνάρτηση void loop() έπεται της void setup() και σηματοδοτεί την αέναη επανάληψη του κώδικα που περιέχει. Στο συγκεκριμένο παράδειγμα η συνάρτηση void loop() είναι κενή.

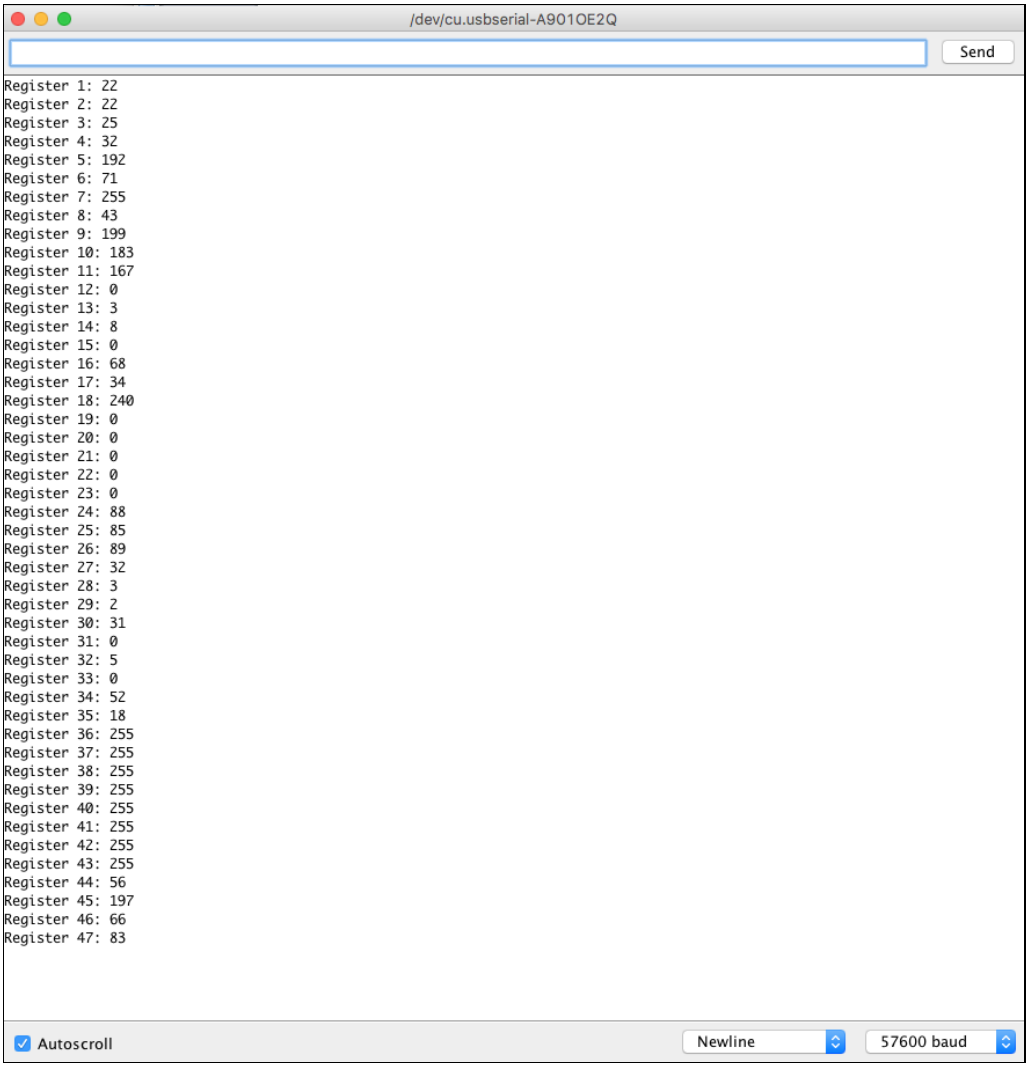
Η πρώτη εντολή εντός της void setup() είναι η εντολή Serial.begin (57600). Με αυτόν τον τρόπο αρχικοποιούμε την σειριακή μας επικοινωνία και συγκεκριμένα το ρυθμό μετάδοσης. Εδώ έχουμε επιλέξει τα 57.600 bits per seconds το οποίο είναι και το maximum rate. Αυτή τη ρύθμιση την χρειάζονται οι συσκευές που επικοινωνούν, (ο υπολογιστής και το board) για το κανάλι επικοινωνίας που πρόκειται να αναπτύξουν. Προφανώς δεν είναι η μόνη επιλογή, αν και

προτείνεται να χρησιμοποιούνται ταχύτητες από αυτήν και κάτω για την ορθότερη διαμεταγωγή δεδομένων. Ποικίλλει ανάλογα με τις εφαρμογές αλλά και φυσικά με τον τύπο του board.

Η εντολή chibiInit() είναι μια επίσης βασική εντολή της βιβλιοθήκης που εγκαταστήσαμε και ουσιαστικά σημαίνει την αρχικοποίηση του Freakduino board.

Στην συνέχεια, παρουσιάζουμε τις τιμές που είναι αποθηκευμένες στους καταχωρητές 1 μέχρι και 47, οι οποίοι είναι μοναδικοί και τους αντιστοιχεί σταθερή θέση στη μνήμη EEPROM. Με την εντολή επανάληψης (εκτελείται 47 φορές, όσες και οι καταχωρητές) μπορούμε κάθε φορά να αποτυπώνουμε το περιεχόμενο του καθενός. Οι εντολές με το χαρακτηριστικό serial.print() ουσιαστικά εκτυπώνουν στην κονσόλα μας. Οι τιμές στους παραπάνω καταχωρητές έχουν δοθεί είτε από το χρήστη (με προγενέστερο prototyping) για συγκεκριμένους λόγους (π.χ συγκεκριμένο mode λειτουργίας) είτε εργοστασιακά από την κατασκευή του board για ακόμα πιο εύκολη και γρήγορη χρήση (να μας επιτραπεί ο χαρακτηρισμός «plug and play»!).

Ύστερα από compile και upload του συγκεκριμένου κώδικα, στην κονσόλα του board μας μπορούμε τα δούμε τις τιμές των καταχωρητών (Εικόνα 2.14):



Εικόνα 2.14: Εμφάνιση περιεχομένων βασικών καταχωρητών.

Επειδή η ένδειξη οποιουδήποτε register με τον αριθμό που περιέχει (π.χ Register 32: 5) προφανώς δεν αποτελεί μια πληροφορία που μπορεί να φανεί χρήσιμη, ακολουθεί η επεξηγηματική λίστα αντιστοίχισης των καταχωρητών όπως αυτή βρέθηκε στην βασική βιβλιοθήκη chb_drvr.h (Εικόνα 2.15) [8]. Περισσότερες πληροφορίες στο παρακάτω link. Στη λίστα αυτή αναφέρεται το νούμερο του κάθε καταχωρητή σε δεκαεξαδική μορφή (1 εως 47) και ένα shortname αυτού σύμφωνα με το ρόλο του. Η λίστα που εξάγουμε καθώς και η λίστα που περιέχεται στην προαναφερθείσα βιβλιοθήκη chb_drvr.h σε συνδυασμό και αντιστοίχιση μας δίνουν αρκετά σημαντική, ενημερωτικού χαρακτήρα, πληροφορία.

https://github.com/freaklabs/chibiArduino/blob/master/src/chb_drvr.h

```
enum
        {
            TRX_STATUS      = 0x01,
            TRX_STATE       = 0x02,
            TRX_CTRL0       = 0x03,
            TRX_CTRL1       = 0x04,
            PHY_TX_PWR      = 0x05,
            PHY_RSSI        = 0x06,
            PHY_ED_LEVEL    = 0x07,
            PHY_CC_CCA      = 0x08,
            CCA_THRES       = 0x09,
            RX_CTRL         = 0x0a,
            SFD_VALUE       = 0x0b,
            TRX_CTRL_2      = 0x0c,
            ANT_DIV         = 0x0d,
            IRQ_MASK        = 0x0e,
            IRQ_STATUS      = 0x0f,
            VREG_CTRL       = 0x10,
            BATMON          = 0x11,
            XOSC_CTRL       = 0x12,
            CC_CTRL_0       = 0x13,
            CC_CTRL_1       = 0x14,
            RX_SYN          = 0x15,
            RF_CTRL_0       = 0x16,
            XAH_CTRL_1      = 0x17,
            FTN_CTRL        = 0x18,
            RF_CTRL_1       = 0x19,
            PLL_CF          = 0x1a,
            PLL_DCU         = 0x1b,
            PART_NUM        = 0x1c,
            VERSION_NUM     = 0x1d,
            MAN_ID_0        = 0x1e,
            MAN_ID_1        = 0x1f,
            SHORT_ADDR_0    = 0x20,
            SHORT_ADDR_1    = 0x21,
            PAN_ID_0        = 0x22,
            PAN_ID_1        = 0x23,
            IEEE_ADDR_0     = 0x24,
            IEEE_ADDR_1     = 0x25,
            IEEE_ADDR_2     = 0x26,
            IEEE_ADDR_3     = 0x27,
            IEEE_ADDR_4     = 0x28,
            IEEE_ADDR_5     = 0x29,
            IEEE_ADDR_6     = 0x2a,
            IEEE_ADDR_7     = 0x2b,
            XAH_CTRL_0      = 0x2c,
            CSMA_SEED_0     = 0x2d,
            CSMA_SEED_1     = 0x2e,
            CSMA_BE         = 0x2f
        };
```

Εικόνα 2.15: Λίστα επεξήγησης καταχωρητών.

Συγκεκριμένα, θα μπορούσαμε να αποφανθούμε για την συμπεριφορά του board αντλώντας σημαντικές πληροφορίες από την επίσημη σελίδα του κατασκευαστή για τον ρόλο που παίζουν οι παραπάνω καταχωρητές. Πιο συγκεκριμένα, στο ηλεκτρονικό έγγραφο

http://www.atmel.com/images/doc5131.pdf και περίπου στη σελίδα 27 ξεκινά η επεξήγηση των καταχωρητών αυτών.

2.8 Αμφίδρομη επικοινωνία (Transmitter – Receiver modes)

Τα Freakduino boards που έχουμε προμηθευτεί είναι πανομοιότυπα. Η επιλογή ποιό από αυτά θα παίξει τον ρόλο του αποστολέα ή του παραλήπτη είναι τελείως τυχαία. Επομένως, ο κώδικας που υλοποιήσαμε και θα χρησιμοποιήσουμε θα δώσει χαρακτήρα παραλήπτη ή αποδέκτη στον κάθε κόμβο. Προφανώς τα συγκεκριμένα boards με κατάλληλο κώδικα μπορούν ταυτόχρονα να υποστηρίζουν και τα δύο modes, receiver και transmitter, κάτι που θα αναλυθεί στη συνέχεια. Προς το παρόν ακολουθούν οι κώδικες για αποστολέα και παραλήπτη ενός data frame αντίστοιχα (Εικόνες 2.16 και 2.17). Στην συγκεκριμένη περίπτωση θα αποστείλουμε το κλασσικό γραπτό μήνυμα «Hello World».

2.8.1 Κώδικας αποστολέα

```
#include <chibi.h>
#define DEST_ADDR 9 //ορισμός δικτυακής διεύθυνσης αποδέκτη
void setup()
{
  Serial.begin(57600); //ταχύτητα σειριακής επικοινωνίας
  chibiInit(); //αρχικοποίηση
  chibiSetShortAddr(1); //δήλωση δικτυακής διεύθυνσης αποστολέα
  chibiSetChannel(15); //δήλωση καναλιού ασύρματης επικοινωνίας
}

void loop()
{
  byte msg[100]; //δήλωση μεταβλητής αποθήκευσης data frame
  strcpy((char *)msg, "Hello World"); //μήνυμα προς αποστολή
  chibiTx(DEST_ADDR, msg, 12); //εντολή αποστολής μηνύματος
}
```

Εικόνα 2.16: Κώδικας αποστολέα (transmitter).

Με την #define DEST_ADDR 9 εντολή καταχωρούμε με ένα γρήγορο τρόπο (διαθέσιμη εντολή βιβλιοθήκης) ποιά θα είναι η δικτυακή διεύθυνση του παραλήπτη μας. Με τις εντολές chibiSetShortAddr(); και chibiSetChannel(); ορίζουμε διεύθυνση και κανάλι επικοινωνίας για τον αποστολέα μας (το συγκεκριμένο board). Η byte msg[100] δημιουργεί απλά μια μεταβλητή

εκχώρησης δεδομένων τύπου character με το όνομα msg μήκους 100 χρακτήρων (δηλώθηκε αυθαίρετα). Στη συνέχεια, η εντολή strcpy((char *)msg, "Hello World") πραγματοποιεί μια καταχώρηση του string «Hello World» στην μεταβλητή msg που δημιουργήσαμε προηγουμένως. Τέλος η εντολή chibiTx(DEST_ADDR, msg, 12) με 3 ορίσματα στην παρένθεση εκτελεί τα εξής: Να σταλεί μήνυμα μέσω radio communication, στην διεύθυνση παραλήπτη (DEST_ADDR) συγκεκριμένου μηνύματος που περιέχεται στην μεταβλητή msg, η οποία έχει μήκος 12 χαρακτήρων (11 χαρακτήρων, στην πραγματικότητα, αλλά χρειάζεται εξ ορισμού ακόμα ένα χαρακτήρα κενό #null για να σηματοδοτηθεί το τέλος του μηνύματος προς αποστολή). Το συγκεκριμένο παράδειγμα είναι ένα string μήκους 12 χαρακτήρων, που δεν είναι απαραίτητο. Να σημειωθεί πως η συγκεκριμένη σύνταξη κώδικα θα στέλνει ασταμάτητα μηνύματα στον αποδέκτη. Εφόσον θέλουμε να μη συμβαίνει κάτι τέτοιο, μπορούμε να διακόψουμε την αέναη επανάληψη με την χρήση της while() καθώς και ένα μετρητή για πεπερασμένο αριθμό επαναλήψεων.

2.8.2 Κώδικας αποδέκτη

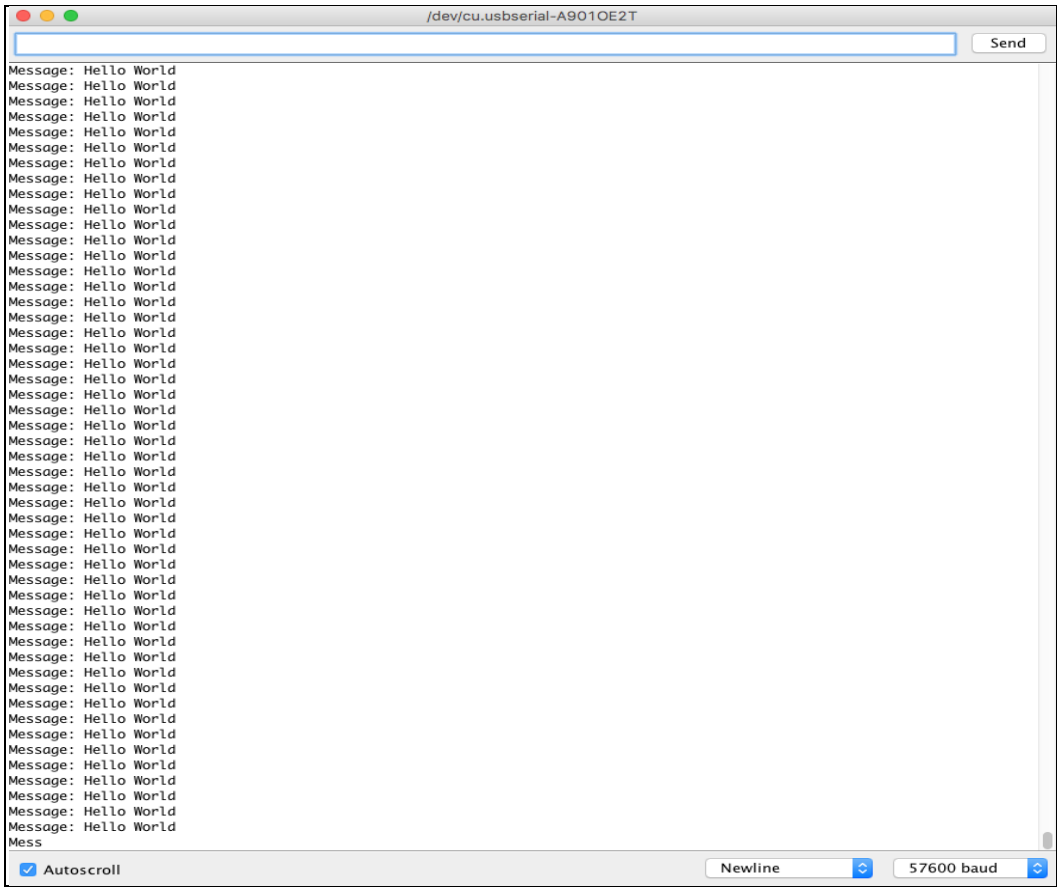
```
#include <chibi.h>
void setup()
{
  Serial.begin(57600); //ταχύτητα σειριακής επικοινωνίας
  chibiInit(); //αρχικοποίηση
  chibiSetShortAddr(9); //ορισμός δικτυακής διεύθυνσης παραλήπτη
  chibiSetChannel(15); //δήλωση καναλιού ασύρματης επικοινωνίας
}

void loop()
{
if (chibiDataRcvd() == true) //Αν ληφθεί μήνυμα από το δίκτυο, να το διαχειριστεί κατάλληλα
  {
    byte len, buf[100];

    len = chibiGetData(buf); //αποθήκευση data frame σε μεταβλητη
    if (len == 0) return; //αν η μεταβλητή είναι κενή, τότε κενό μήνημα, να παραληφθεί από τη διαδικασία

    Serial.print("Message: "); //εντολή εμφάνισης string στην κονσόλα
```

```
    Serial.println((char *)buf); //εντολή εμφάνισης περιεχομένου μεταβλητής
buf στην κονσόλα
  }
}
```

Εικόνα 2.17: Κώδικας παραλήπτη (receiver).

Η μεταβλητή len παίρνει το μήκος του buf. Αν το μήκος είναι 0 τότε η συνάρτηση δεν κάνει τίποτε. Αν όχι τότε τυπώνει το περιεχόμενο του buf στην κονσόλα.

Στην Εικόνα 2.18 εμφανίζεται η επιβεβαίωση της παραλαβής του συγκεκριμένου μηνύματος.



Εικόνα 2.18: Κονσόλα καταγραφής συμβάντων όπου εμφανίζεται η επιβεβαίωση για τη λήψη του μηνύματος.

2.9 Message loopback and data manipulation

Με βάση τους παραπάνω κώδικες για την λειτουργία αποστολέα ή παραλήπτη, μπορούμε εύκολα να μετατρέψουμε έναν ή και περισσότερους κόμβους σε λειτουργία αναμεταδότη (παραλήπτη και αποστολέα ταυτόχρονα). Ετσι θα έχουμε καταφέρει την διπλή ταυτότητα σε κάθε μας κόμβο η οποία παίζει σημαντικό ρόλο στο στήσιμο ενός δικτύου ασύρματων κόμβων.

Η πληροφορία δεν θα σταματά απαραίτητα σε κάποιον κόμβο αν αυτός δεν έχει ρυθμιστεί σωστά ή αν ο χρήστης απλά δεν το έχει επιλέξει ή δεν το επιθυμεί για συγκεκριμένο λόγο. Με κάποιες παραλλαγές στον κώδικα αλλά και κάποιες προσθέσεις μπορούμε να καταφέρουμε την αποστολή ενός μηνύματος (data frame) από τον αρχικό μας αποστολέα στον παραλήπτη και ξανά πίσω στον αποστολέα. Το μήνυμα θα έχει διαγράψει έτσι ένα πλήρη κύκλο. Αυτό θα μας βοηθήσει μετέπειτα στην αποστολή μηνυμάτων με πιο σύνθετο τρόπο εφόσον θα εχουμε την δυνατότητα να επέμβουμε στους κανόνες δρομολόγησης μηνυμάτων και να χρησιμοποιούμε συγκεκριμένα μοτίβα για την πορεία της πληροφορίας σύμφωνα πάντα με τις ανάγκες του δικτύου που θα έχουμε δημιουργήσει.

Ακολουθεί ξανά κώδικας αποστολέα και παραλήπτη (Εικόνες 2.19 και 2.20), αυτή τη φορά με προσθήκες και τροποποιήσεις με σκοπό τον διττό χαρακτήρα τους. Για λόγους ευκολίας και πρακτικότητας, παρατίθενται σχόλια στον κώδικα με σκοπό την απλούστερη επεξήγηση του. Οι κώδικες του αποστολέα και του παραλήπτη βρίσκονται στα αρχεία, αντίστοιχα,

paper/Sketches/transmitter_081216_manual/transmitter_081216_manual.ino &

paper/Sketches/receiver_211116_cmd/receiver_211116_cmd.ino.

2.9.1 Κώδικας αποστολέα

```
#include <chibi.h>
#define DEST_ADDR 9 //δήλωση δικτυακής διεύθυνσης παραλήπτη
int i = 1; //αρχικοποίηση μετρητή αποστελλούμενων πακέτων

void setup()
{
  Serial.begin(57600); //ταχύτητα σειριακής επικοινωνίας
  chibiInit(); //αρχικοποίηση
  chibiSetShortAddr(1); //δήλωση δικτυακής διεύθυνσης
  Serial.println();
  delay(3000); //καθυστερηση 3 δευτερολέπτων
}

void loop()
{
  sendPacket(); //κάλεσμα συνάρτησης sendPacket
  delay(2000); //καθυστέρηση 2 δευτερολέπτων
  i++; //προσθήκη ενός ακόμα πακέτου στον μετρητή αποστελλούμενων πακέτων
```

```
  if (chibiDataRcvd() == true) { //λήψη πακέτων δεδομένων ή όχι
    Serial.println("Received message back. Displaying:"); //εκτύπωση στην
κονσόλα
    receivePacket(); //κάλεσμα συνάρτησης receivePacket
  }

//while(1){} Η while βρίσκεται σε σχόλια. Απλά μπορεί να προστεθεί εδώ για
πεπερασμένο αριθμό επαναλήψεων.
}

void receivePacket() { //συνάρτηση receivePacket
    byte len, buf2[100]; // δήλωση μεταβλητών
    len = chibiGetData(buf2); //καταχώρηση ληφθέντων δεδομένων απο radio


void sendPacket() { //συνάρτηση sendPacket
  Serial.print("Sending packet ");
  Serial.println(i); //εκτύπωση μετρητή αποστελλούμενου πακέτου
  byte dataBuf[100]; //δήλωση – δημιουργία μεταβλητής dataBuf
  strcpy((char *)dataBuf, "This is a dummy message");
  chibiTx(DEST_ADDR, dataBuf, strlen((char *)dataBuf) + 1); //εντολή
αποστολής δεδομένων με κατάλληλες παραμέτρους

}

//if (len == 0) return; //συνθήκη εγκυροποίησης
    Serial.print("Message: "); //εκτύπωση
    Serial.println((char *)buf2); //εκτύπωση εισερχομένου μηνύματος
(περιεχόμενο μεταβλητής buf2)
}
```

Εικόνα 2.19: Κώδικας αποστολέα (transmitter).

2.9.2 Κώδικας αποδέκτη

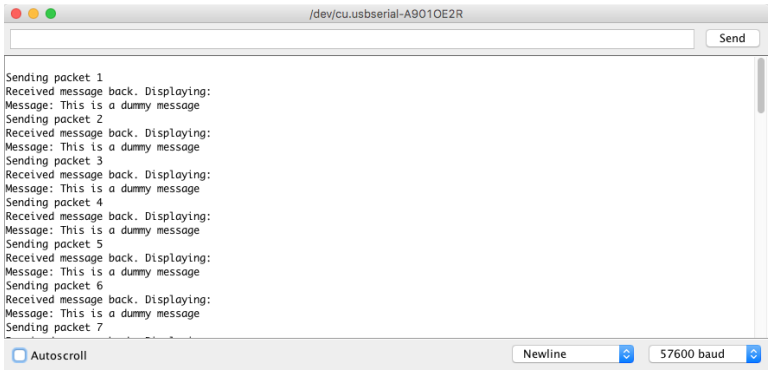
```
#include <chibi.h>
#define DEST_ADDR 1 //δήλωση δικτυακής διεύθυνσης παραλήπτη
int i=0; //αρχικοποίηση μετρητή ληφθέντων πακέτων
```

```
bool sendPacket = false; //βοηθητική μεταβλητή ενεργοποίησης - κλήσης συνάρτητης
byte buf[100]; //δήλωση μεταβλητής
int len, rssi, src_addr;

void setup()
{
  Serial.begin(57600); //ταχύτητα σειριακής επικοινωνίας
  chibiInit(); //αρχικοποίηση
  chibiSetShortAddr(9); //δήλωση δικτυακής διεύθυνσης
  Serial.println("Starting..."); //μήνυμα σηματοδότησης έναρξης
}

void loop()
{
 if (chibiDataRcvd() == true) //Συνθήκη αν ελήφθη μήνυμα από radio
  {
    byte len;
    len = chibiGetData(buf);
    if (len == 0) return; // if no len, discard (συνθήκη κενού ή όχι μηνύματος)
    i++; //Αύξηση μετρητή ληφθέντων μηνυμάτων
    rssi = chibiGetRSSI(); //μεταβλητή αναφοράς ισχύος σήματος λήψης πακέτου (received signal strength indicator)
    src_addr = chibiGetSrcAddr(); //μεταβλητή αναφοράς δικτυακής διεύθυνσης αποστολέα μηνύματος (source address)
    sendPacket = true;
    Serial.print("Message received from node "); //εκτύπωση δεδομένων στην κοσνόλα
    Serial.print(src_addr, DEC);
    Serial.print(": ");
    Serial.print((char *)buf);
    Serial.print(", RSSI = ");
    Serial.print(rssi, DEC);
    Serial.print(", msg Nr: ");
    Serial.println(i);
   }
```

```
  if (sendPacket == true) {
    sendPacket = false;
    chibiTx(DEST_ADDR, buf, strlen((char *)buf) + 1); //εντολή επαναποστολής
ήδη ληφθέντος μηνύματος
  }

}
```

Εικόνα 2.20: Κώδικας παραλήπτη (receiver).

Τέλος στην επόμενη Εικόνα 2.21 εμφανίζεται η οθόνη (κονσόλα) με το ληφθέν μήνυμα, επιβεβαιώνοντας έτσι την επιτυχή μετάδοση.



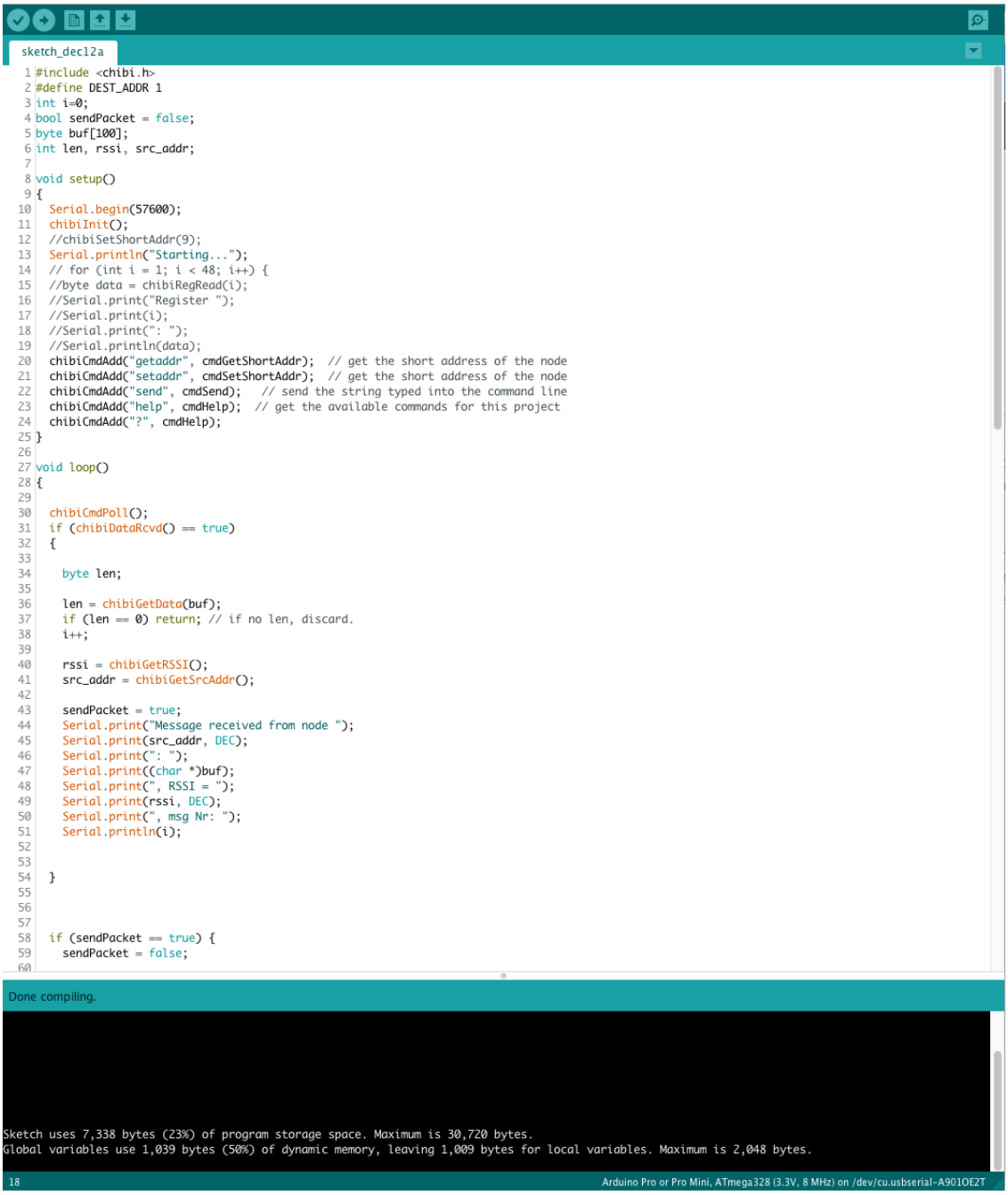
Εικόνα 2.21: Κονσόλα εμφάνισης λήψης μηνύματος.

Σύνοψη της προηγούμενης διαδικασίας:

Στην ουσία αρχικοποιούμε τον αποστολέα να στέλνει αδιάκοπα μηνύματα (άκρως παραμετροποιήσιμο από χρήστη) προς μια συγκεκριμένη δικτυακή διεύθυνση. Ο μετρητής αναφέρει πόσα μηνύματα αποστέλλονται σε πραγματικό χρόνο και συγκεκριμένα την ώρα της αποστολής του κάθε συγκεκριμένου μηνύματος. Έπειτα, ο παραλήπτης λαμβάνει το μήνυμα και το επεξεργάζεται. Εξάγει πληροφορίες όσο αφορά το μήνημα αυτό, όπως το ποιο είναι (εκτύπωση μηνύματος στην οθόνη), με ποια τιμή ισχύος απολαβής περελήφθη το μήμυμα (received signal strength indicator σε dB), από πού παραλήφθηκε (διεύθυνση αποστολέα) και τέλος τον αύξοντα αριθμό του μηνύματος που παραλήφθηκε (με σκοπό να γίνει αντιληπτός ο επιτυχής αριθμός των αποστελούμενων και παραληφθέντων πακέτων, packet loss). Στη συνέχεια, κάθε ένα από αυτά τα μηνύματα αποστέλλεται πίσω μόνο που ο παραλήπτης πια είναι ο αρχικός αποστολέας.

2.10 Δυνατότητα διαχείρισης και παραμετροποίησης μέσω γραμμής εντολών (Command Line Prompting)

Η βιβλιοθήκη του chibiArduino περιέχει εντολές που δίνουν την δυνατότητα να ελέγχεται και να παραμετροποιείται το board από τον χρήστη μέσω απλού περιβάλλοντος γραμμής εντολών. Έτσι, ο χρήστης, έχοντας μια διαδραστική γραμμή εντολών πραγματοποιεί διαδικασίες όπως παραμετροποίηση δικτυακών διευθύνσεων για κάθε κόμβο, αλλαγή καναλιού επικοινωνίας κ.α. πολύ απλά και γρήγορα. Δίνει, επίσης, τον έλεγχο στο χρήστη να αποστέλλει δεδομένα κατά βούληση σε όποια διεύθυνση αυτός επιθυμεί. Με την επιτυχή μεταφόρτωση του επιθυμητού κώδικα στο Freakduino board, πλέον ο κώδικας «τρέχει» σε πραγματικό χρόνο και μπορούμε ανοίγοντας μια κονσόλα στον υπολογιστή μας (με ρυθμίσεις listening στην συγκεκριμένη πόρτα που είναι συνδεδεμένο το board) να δούμε ένα διαδραστικό περιβάλλον γραμμής εντολών (command prompt) (Εικόνα 2.22).



Εικόνα 2.22: Εμφάνιση του περιβάλλοντος Arduino IDE με δυνατότητα κονσόλας.

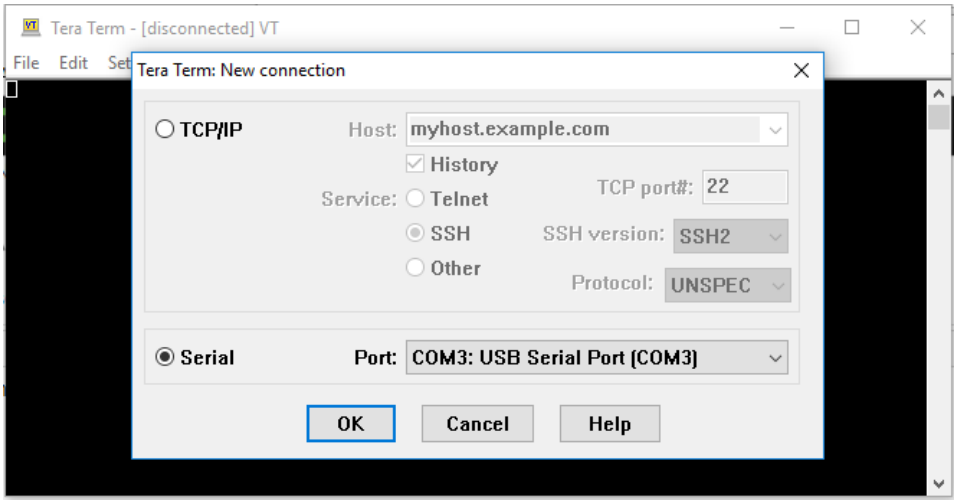
Παρατηρούμε στην επάνω δεξιά γωνία του κειμενογράφου κώδικα Arduino IDE το εικονίδιο με τον μεγεθυντικό φακό, το οποίο ενεργοποιεί την κονσόλα που διαθέτει το ίδιο το Arduino IDE, οι δυνατότητες της οποίας κρίνονται σχετικά περιορισμένες. Για τον λόγο αυτό, προτείνουμε για περιβάλλον Windows εναλλακτικά κάποιο άλλο πρόγραμμα που εξομοιώνει άριστα την κονσόλα και δίνει την δυνατότητα του prompting. Αυτό είναι το Teraterm το οποίο διατίθεται προς χρήση εντελώς δωρεάν.

Πληκτρολογούμε την ηλεκτρονική διεύθυνση:

https://osdn.net/projects/ttssh2/downloads/66795/teraterm-4.93.exe/ .

Μεταφορτώνουμε το αρχείο teraterm-4.93.exe στον Η/Υ και το εκτελούμε με διπλό κλικ. Ακολουθούμε τις οδηγίες του οδηγού εγκατάστασης και ολοκληρώνουμε την εγκατάσταση σε φάκελο της αρεσκείας μας ή στον προεπιλεγμένο (default). Εκκινούμε το πρόγραμμα.

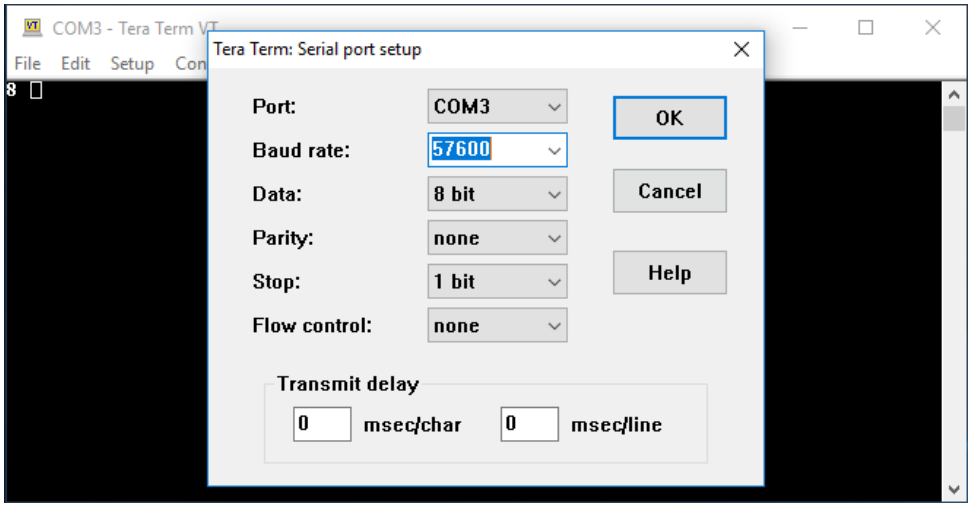
Από το menu File→ New Connection εμφανίζεται το παρακάτω παράθυρο



Εικόνα 2.23: Αρχικοποίηση προγράμματος Teraterm.

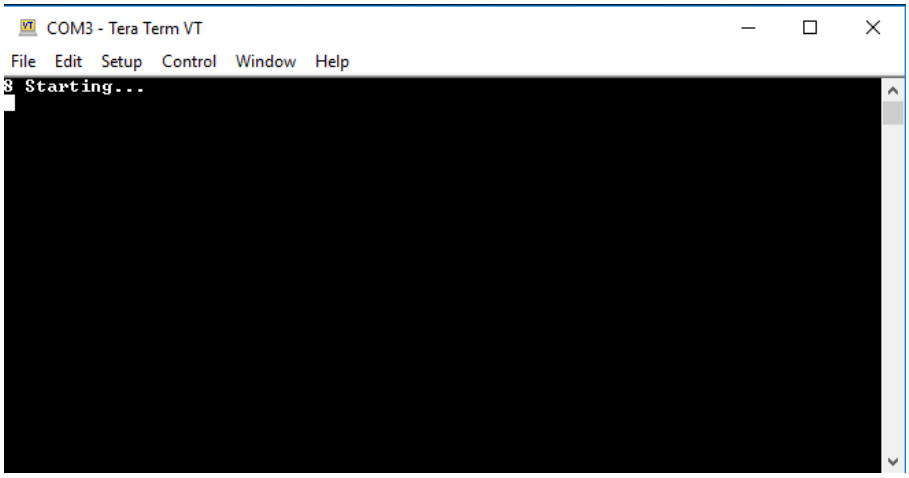
Επιλέγουμε το Serial και εκεί πλέον μας εμφανίζει τις διαθέσιμες σειριακές θύρες που είναι ενεργές (συνήθως εκτός της COM1, που είναι δεσμευμένη από το σύστημα). Επιλέγουμε την σειριακή πόρτα που είναι συνδεδεμένο το Freakduino board, στην συγκεκριμένη περίπτωση την COM3.

Ύστερα, πλοηγούμαστε στο menu Setup → Serial Port και μπορούμε να επιλέξουμε ρυθμίσεις για την σειριακή θύρα που έχουμε συνδέσει το Freakduino board. Επιλέγουμε Baud rate: 57.600 (όμοιο data rate που έχει δηλωθεί και στον κώδικά μας) και πατάμε OK. Οι υπόλοιπες ρυθμίσεις της σειριακής επικοινωνίας είναι όπως στο παράδειγμα της Εικόνας 2.24.



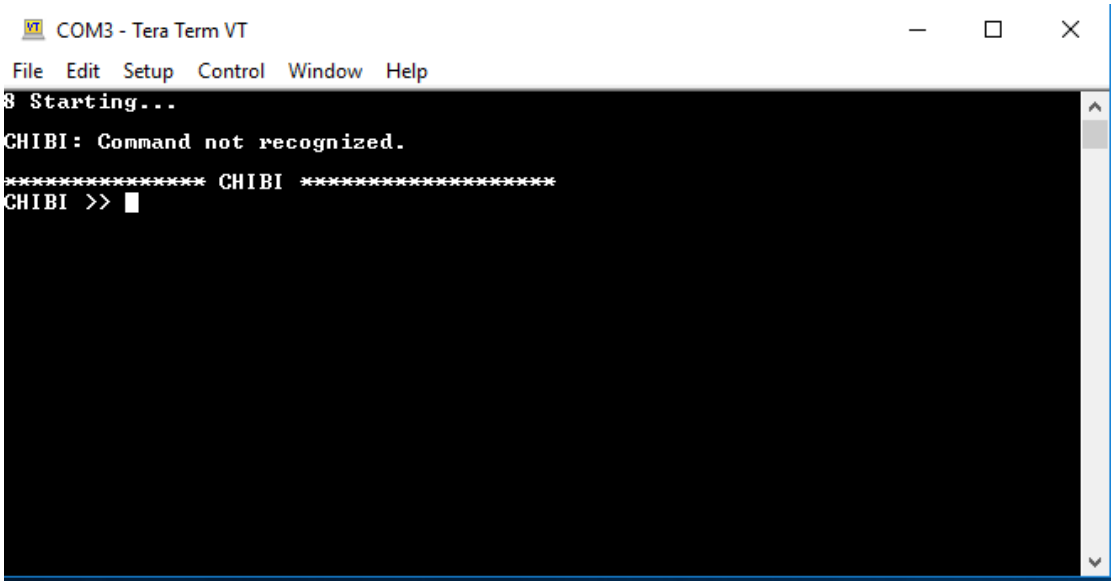
Εικόνα 2.24: Επιλογές θύρας σειριακής επικοινωνίας Teraterm.

Πλέον έχουμε ρυθμίσει επιτυχώς την κονσόλα σειριακής επικοινωνίας. Πατάμε το RESET button στο Freakduino board ώστε να «ξανατρέξει» ο κώδικας με σωστές πλέον ρυθμίσεις από την κονσόλα ώστε να έχουμε απεικόνιση στην οθόνη. Θα φανεί η λέξη Starting… (είναι σημείο στον κώδικά μας να φαίνεται η συγκεκριμένη λέξη με το που εκκινεί το Freakduino board) όπου μας επιβεβαιώνει πως οι ρυθμίσεις μας έχουν γίνει σωστά.



Εικόνα 2.25: Εναρξη επικοινωνίας με Freakduino board.

Πατώντας το Enter ενεργοποιούμε την cmd βιβλιοθήκη και το διαδραστικό κομμάτι εντολών από το Freakduino board είναι πλέον πραγματικότητα.



Εικόνα 2.26: Γραμμή εντολών στο Freakduino board.

Πλέον είμαστε έτοιμοι να εκτελέσουμε εντολές από την διαδραστική κονσόλα του Freakduino board. Ο κέρσορας βρίσκεται σε αναμονή πληκτρολόγησης εντολών ελέγχου.

Στην επόμενη Εικόνα 2.27 παρατίθεται ο κώδικας του αποδέκτη που στη συνέχεια γίνεται αποστολέας με ενεργοποιημένη την δυνατότητα του cmd prompting. Για ευνόητους λόγους δεν θα σχολιαστεί περαιτέρω ο ήδη σχολιασμένος στην παράγραφο 2.9 κώδικας αλλά μόνο οι προσθήκες για τη λειτουργία cmd. (Τοποθεσία αρχείου κώδικα: paper/Sketches/receiver_211116_cmd/receiver_211116_cmd.ino).

```
#include <chibi.h>
#define DEST_ADDR 1
int i=0;
bool sendPacket = false;
byte buf[100];
int len, rssi, src_addr;

void setup()
{
  Serial.begin(57600);
  chibiInit();
  Serial.println("Starting...");
  chibiCmdAdd("getaddr", cmdGetShortAddr);   // get the short address of the
node
```

```
  chibiCmdAdd("setaddr", cmdSetShortAddr);  // set the short address of the
node
  chibiCmdAdd("send", cmdSend);    // send the string typed into the command
line
  chibiCmdAdd("help", cmdHelp);    // get the available commands for this
project
  chibiCmdAdd("?", cmdHelp);
}

void loop()
{
  chibiCmdPoll(); //διαδικασία για ενεργοποίηση γραμμής εντολών την στιγμή
που ο χρήστης εισάγει κάτι από κονσόλα.
  if (chibiDataRcvd() == true)
  {
    byte len;
     len = chibiGetData(buf);
    if (len == 0) return; // if no len, discard.
    i++;
    rssi = chibiGetRSSI();
    src_addr = chibiGetSrcAddr();
    sendPacket = true;
    Serial.print("Message received from node ");
    Serial.print(src_addr, DEC);
    Serial.print(": ");
    Serial.print((char *)buf);
    Serial.print(", RSSI = ");
    Serial.print(rssi, DEC);
    Serial.print(", msg Nr: ");
    Serial.println(i);
   }



  if (sendPacket == true) {
    sendPacket = false;
    chibiTx(DEST_ADDR, buf, strlen((char *)buf) + 1);
  }
```

```
}
  // USER FUNCTIONS (εδω περιγράφεται η δήλωση και η λειτουργία εντολών με
μορφή συναρτήσεων)
/*********************************************************************/

/*********************************************************************/
/*!
    Get short address of device from EEPROM
    Usage: getaddr
*/
/*********************************************************************/
void cmdGetShortAddr(int arg_cnt, char **args)
{
  int val;

  val = chibiGetShortAddr();
  Serial.print("Short Address: "); Serial.println(val, DEC);
}

/*********************************************************************/
/*!
    Write short address of device to EEPROM
    Usage: setaddr <addr>
*/
/*********************************************************************/
void cmdSetShortAddr(int arg_cnt, char **args)
{
  int val;

  val = chibiCmdStr2Num(args[1], 16);
   chibiSetShortAddr(val);
}
/*********************************************************************/
/*!
    Transmit data to another node wirelessly using Chibi stack. Currently
    only handles ASCII string payload
    Usage: send <addr> <string...>
*/
/*********************************************************************/
```

```
void cmdSend(int arg_cnt, char **args)
{
    byte data[100];
    int addr, len;

    // convert cmd line string to integer with specified base
    addr = chibiCmdStr2Num(args[1], 16);

    // concatenate strings typed into the command line and send it to the
specified address
    len = strCat((char *)data, 2, arg_cnt, args);
    chibiTx(addr, data,len
}
/*************************************************************************/
/*!
    Concatenate multiple strings from the command line starting from the
    given index into one long string separated by spaces.
*/
/*************************************************************************/
int strCat(char *buf, unsigned char index, char arg_cnt, char **args)
{
    uint8_t i, len;
    char *data_ptr;

    data_ptr = buf;
    for (i=0; i<arg_cnt - index; i++)
    {
        len = strlen(args[i+index]);
        strcpy((char *)data_ptr, (char *)args[i+index]);
        data_ptr += len;
        *data_ptr++ = ' ';
    }
    *data_ptr++ = '\0';

    return data_ptr - buf;
/*************************************************************************/
/*!
    help instructions for available commands.. Just type "help" or "?" and
enjoy!! Feel free to use!
```

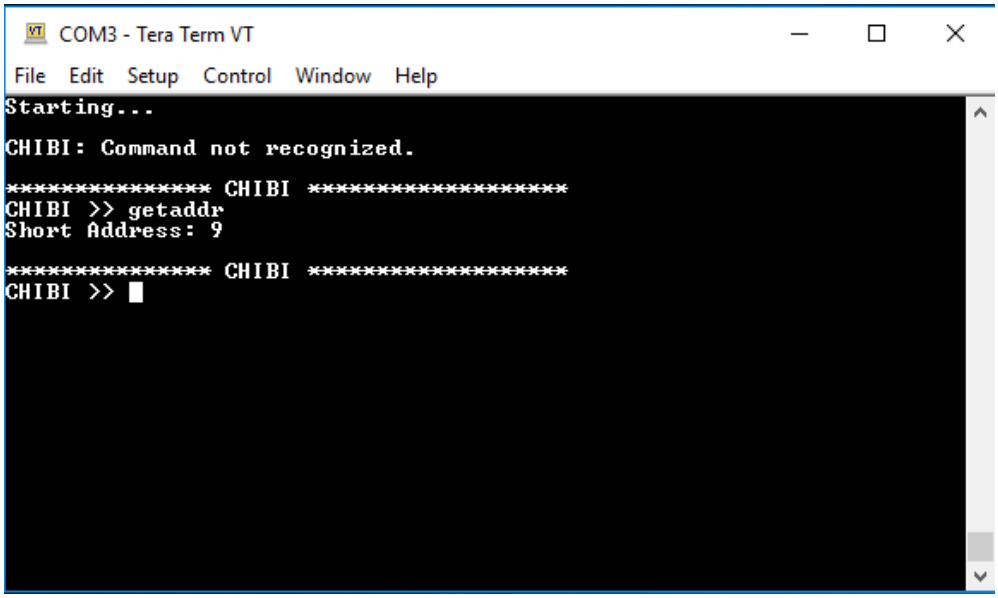
```
*/
/*************************************************************************/
void cmdHelp()
{
Serial.println("getaddr --- Usage: [getaddr] --- Gets short address of device
from EEPROM");
Serial.println("setaddr --- Usage: [setaddr <addr>] --- Writes short address
of device to EEPROM");
Serial.println("send  ---  Usage:  [send  <addr>  <string...>]  ---  Transmits
specific data (string) to another specific node");
```

Εικόνα 2.27: Κώδικας αποστολέα – παραλήπτη, με δυνατότητα γραμμής εντολών από χρήστη.

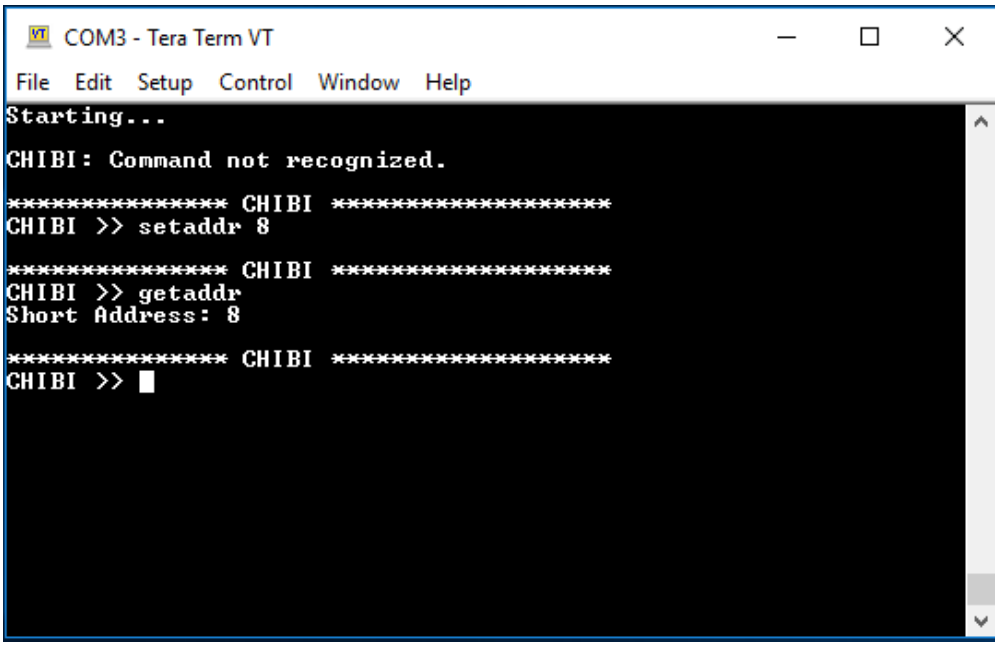
Με τον παραπάνω κώδικα έχουμε δημιουργήσει τρεις (3) εντολές ελέγχου και μία (1) εντολή ενημέρωσης. Οι εντολές αυτές είναι οι εξής:

1. Η getaddr επιστρέφει την δικτυακή διεύθυνση του κόμβου στον οποίο είμαστε συνδεδεμένοι. Ουσιαστικά ζητά την τιμή που έχει αποθηκευτεί στην θέση μνήμης που έχει δεσμευτεί για την δικτυακή διεύθυνση.
2. Η setaddr μπορεί να αποθηκεύσει εκ νέου την δικτυακή διεύθυνση του κόμβου που είμαστε συνδεδεμένοι. Ο χρήστης δηλώνει ποιά διεύθυνση θέλει να έχει πλέον ο κόμβος και γίνεται rewrite της ήδη υπάρχουσας τιμής. Ο κόμβος πλέον θα χρησιμοποιεί αυτή τη διεύθυνση.
3. Η send μπορεί να στείλει συγκεκριμένο μήνυμα σε συγκεκριμένο αποδέκτη. Η φιλοσοφία της λειτουργίας της είναι απλή. Μετατρέπει το μήνυμα που θέλει να αποστείλει ο χρήστης από διάφορες λέξεις που μπορεί να χωρίζονται με κενά σε ένα ενιαίο string και το αποθηκεύει σε μια προσωρινή μεταβλητή τύπου string. Αυτή τη χρησιμοποιεί μετά με την συνάρτηση αποστολής μηνύματος που έχει επεξηγηθεί νωρίτερα. Η διεύθυνση του αποδέκτη από δεκαδικό νούμερο που δηλώνει ο χρήστης μετατρέπεται σε δεκαεξαδικό (hexadecimal) και στη συνέχεια καταχωρείται σε συγκεκριμένη θέση μνήμης (βλέπε αναφορά στους καταχωρητές συστήματος).
4. Η help, ή ?, έχει ενημερωτικό χαρακτήρα και στην ουσία βοηθά το χρήστη να καταλάβει ποιές εντολές είναι διαθέσιμες προς χρήση και μάλιστα την απαραίτητη σύνταξη για την επιτυχή εκτέλεση των διαθέσιμων εντολών.

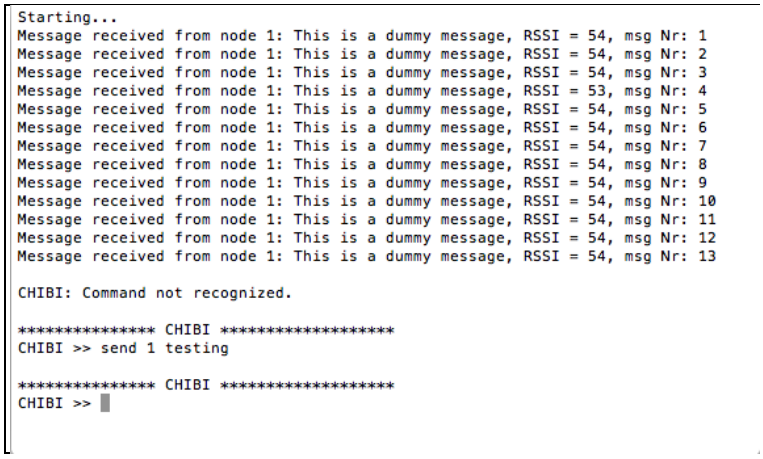
Ακολουθούν εικονικά παραδείγματα εκτέλεσης εντολών.



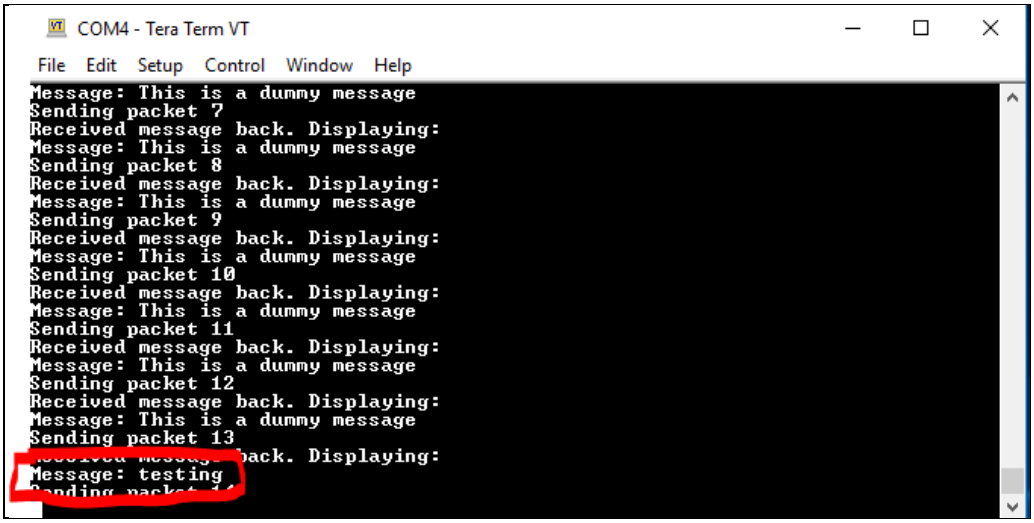
Εικόνα 2.28: Χρήση εντολής getaddr.



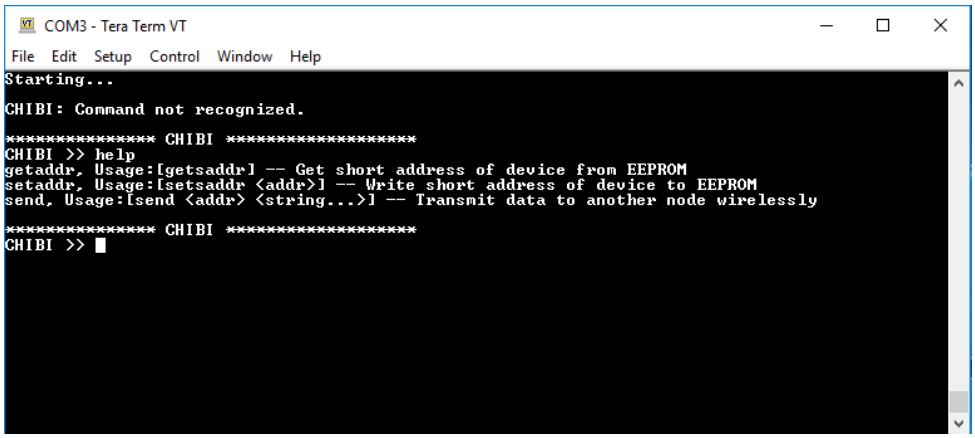
Εικόνα 2.29: Χρήση εντολής setaddr.



Εικόνα 2.30: Χρήση εντολής send.



Εικόνα 2.31: Παραλαβή μηνύματος μέσω εντολής send (παραλαβή από τον «άλλο» κόμβο).



Εικόνα 2.31: Χρήση εντολής help.

ΚΕΦΑΛΑΙΟ 3:

Δρομολόγηση πακέτων πληροφορίας και διαδικασία επιλογής λοβού ακτινοβολίας κεραίας

Όπως αναφέρθηκε ήδη στην Εισαγωγή, η παρούσα διπλωματική εργασία αφορά την σχεδίαση και ανάπτυξη λογισμικού για WSNs με στόχο εφαρμογές IoT, των οποίων οι κόμβοι είναι εξοπλισμένοι με ευφυείς κεραίες μεταγωγής λοβού. Οι εξοπλισμένοι κόμβοι με την βοήθεια του λογισμικού θα προωθούν το φορτίο τους ενεργοποιώντας κάθε φορά εκείνο τον λοβό της κεραίας που είναι κατάλληλότερος για μετάδοση και λήψη. Το λογισμικό θα περιλαμβάνει διεπαφή με τον χρήστη του δικτύου για την δημιουργία φορτίου, έλεγχο του δικτύου και καταγραφή της επίδοσής του.

Στις επόμενες παραγάφους περιγράφεται αναλυτικά η ανάπτυξη του λογισμικού των κόμβων για ένα δίκτυο WSN στα 2.4 GHz με 3 κόμβους και ένα δίκτυο WSN στα 915 MHz με 5 κόμβους. Ο τρόπος ανάπτυξης των πινάκων δρομολόγησης και των πινάκων επιλογής λοβού γίνεται με τρόπο ο οποίος είναι άμεσα επεκτάσιμος σε δίκτυα με περισσότερους κόμβους. Επίσης, αναφέρεται αναλυτικά ο κώδικας λειτουργίας των κόμβωνκαι οι διάφορες τεχνικές που χρησιμοποιήθηκαν, ενώ ελέγχεται η λειτουργία τους σε διάφορα σενάρια με φυσικά εμπόδια στην επικοινωνία αυτών και γενικότερα εξομοιώνεται η λειτουργία τους όσο πιο κοντά μπορεί να υπάρξει σε πραγματικές συνθήκες. Τέλος, παρέχεται η δυνατότητα στον χρήστη να μπορεί να παραμετροποιήσει το δίκτυο από γραμμή εντολών σε κάθε κόμβο και να λάβει στατιστικά δεδομένα για την λειτουργία του.

Παρακάτω αναλύονται οι τρόποι και διαδικασίες που ακολουθήθηκαν για την δημιουργία των δικτύων αυτών.

3.1 Unicode (ενιαίος κώδικας για κάθε κόμβο)

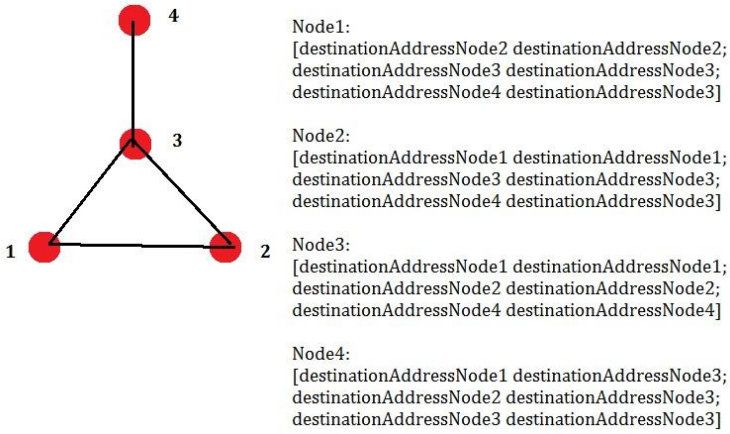
Για τις ανάγκες της συγκεκριμένης διάταξης κόμβων, έχει κριθεί αναγκαίο να μπορεί κάθε ένας από αυτούς να λειτουργεί με έναν κώδικα ο οποίος θα είναι κοινός για όλους τους κόμβους, αποφεύγοντας έτσι την περαιτέρω αλλαγή του σε περίπτωση που χρειάζεται κόμβος

αποστολέα και αποδέκτη ή μόνο αποστολέα ή μόνο αποδέκτη. Ο κώδικας πρέπει να είναι ελαφρύς και ευέλικτος. Επίσης, αυτό που πρέπει να προσφέρει είναι διαδραστικότητα καθώς και λειτουργικότητα. Έτσι λοιπόν, αλλάχτηκε για τις παραπάνω ανάγκες ο υπάρχων κώδικας και πλέον ο χρήστης μπορεί, μέσω της κονσόλας επικοινωνίας με τον εκάστοτε κόμβο, να διευθυνσιοδοτήσει, να αποστείλει μηνύματα και να πάρει χρήσιμες πληροφορίες για την συμπεριφορά του δικτύου καθώς και να διαχειριστεί κανάλια επικοινωνίας και άλλες σημαντικές ρυθμίσεις. Επομένως, ανεξαρτήτως του αριθμού των κόμβων, ο κώδικας είναι κοινός να προσφέρει μεγάλη κλίμακα παραμετροποίησης και ευελιξία.

3.2 Πίνακας Δρομολόγησης (Routing table)

Ο **πίνακας δρομολόγησης** είναι ένας μικρός διδιάστατος πίνακας ο οποίος βρίσκεται αποθηκευμένος μέσα σε κάθε κόμβο του δικτύου. Είναι ενιαίος και περιλαμβάνει για κάθε κόμβο τις πληροφορίες επόμενου αποδέκτη βάσει του επιθυμητού τελικού αποδέκτη. Άρα, όλοι οι κόμβοι γνωρίζουν την συμπεριφορά κάθε κόμβου μέσα στο δίκτου. Ο πίνακας δρομολόγησης προκύπτει ως αποτέλεσμα της εκτέλεσης ενός αλγόριθμου δρομολόγησης, ο οποίος δίνει ως αποτέλεσμα τον επόμενο σταθμό στον οποίο πρέπει να μεταφερθεί ένα πακέτο πληροφορίας. Ο πίνακας δρομολόγησης αποθηκεύει όλες τις διαδρομές προς άλλους σταθμούς του δικτύου.

Στην συγκεκριμένη περίπτωση του δικτύου αισθητήρων οι διαδρομές αυτές θεωρούνται στατικές και ο επόμενος σταθμός είναι προκαθορισμένος από εμάς.



Εικόνα 3.1: Μοντέλο 4 κόμβων και πίνακα δρομολόγησης αυτών.

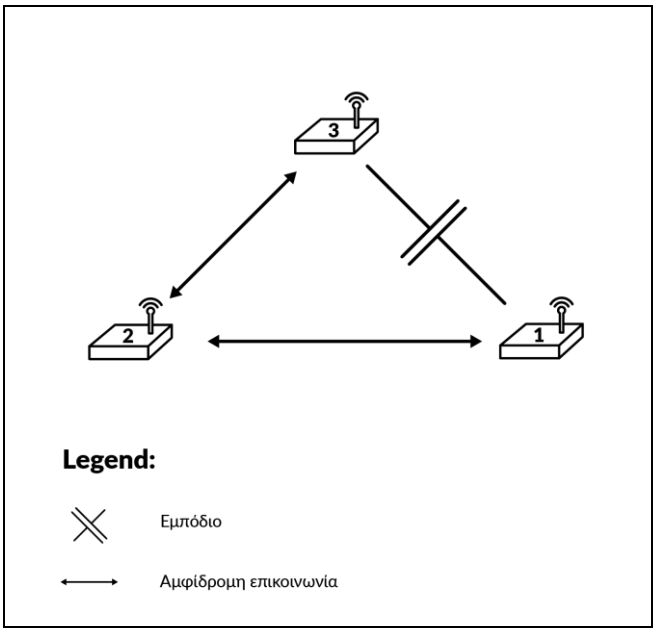
Οι κόμβοι θεωρούνται ταυτόχρονα και αποστολείς αλλά και παραλήπτες (tranceivers). Ακολουθώντας πιστά τον πίνακα δρομολόγησης που ο χρήστης έχει δηλώσει προκύπτει ένα

συγκεκριμένο μοντέλο αποστολής πακέτων και διασύνδεσης κάθε κόμβου στο συγκεκριμένο δίκτυο. Ορίζοντας μέσα από τον πίνακα δρομολόγησης τον επόμενο σταθμό για τον τελικό παραλήπτη, στην ουσία έχει δηλωθεί – πραγματοποιηθεί ο αλγόριθμος δρομολόγησης.

Στην περίπτωση του συγκεκριμένου δικτύου αισθητήρων, ο πίνακας δρομολόγησης κρίνεται αναγκαίος εφόσον θέλουμε το πακέτο να ακολουθεί συγκεκριμένη διαδρομή για κάποιο λόγο. Παραδείγματος χάριν, στην περίπτωση που κάποιος κόμβος παρεμποδίζεται από φυσικά εμπόδια, κάποιος κόμβος έχει βγει εκτός λειτουργίας ή ακόμη κάποιος κόμβος χρειάζεται να στείλει την πληροφορία του σε κάποιον άλλο κόμβο αναγκαστικά όπως για παράδειγμα την μαζική συλλογή πληροφορίας (sink node), ο πίνακας δρομολόγησης παίζει τον ρόλο του καθοδηγητή. Είναι αυτός που δηλώνει στην ουσία τον δρόμο που θα ακολουθήσει το πακέτο πληροφορίας, δηλαδή ουσιαστικά «χαρτογραφεί» το δίκτυο αισθητήρων.

3.2.1 Ασύρματο δίκτυο τριών (3) κόμβων

Για να γίνει ευκολα κατανοητή η προσέγγισή μας στην ανάπτυξη του πίνακα δρομολόγησης στην παρούσα εργασία, στην Εικόνα 3.2 παρουσιάζεται ένα παράδειγμα δικτύου 3 κόμβων, με ένα φυσικό εμπόδιο μεταξύ των κόμβων 1 και 3. Η ύπαρξη του εμποδίου απαγορεύει την απευθείας επικοινωνία μεταξύ των κόμβων αυτών. Το εμπόδιο μπορεί να είναι απτό, όπως π.χ. ένα ογκώδες μεταλλικό αντικείμενο, ή νοητό, όπως π.χ. το ότι οι κόμβοι 1 και 3 βρίσκονται εκτός εμβέλειας μεταξύ τους.



Εικόνα 3.2: Ασύρματο δίκτυο 3 κόμβων με εμπόδιο επικοινωνίας.

Στην περίπτωση αυτή, ο πίνακας δρομολόγησης θα είναι όπως στην Εικόνα 3.3:

```
static int routes[3][3] = {
     {0,2,2},
     {1,0,3},
     {2,2,0}
     };
```

Εικόνα 3.3: Πίνακας δρομολόγησης τριών κόμβων

Όπως βλέπουμε, ο πίνακας δρομολόγησης αποτελείται από 3 γραμμές και 3 στήλες, όσοι δηλαδή είναι και οι κόμβοι του δικτύου. Η πρώτη γραμμή περιλαμβάνει τους επόμενους κόμβους που πρεπει να ακολουθήσει η πληροφορία. Συγκεκριμένα, ο κόμβος 1 ($1^{η}$ γραμμή) για να στείλει στον κόμβο 1 ($1^{η}$ στηλη) αποστέλλει στον κόμβο 0 όπου δεν υφίσταται εφόσον είναι ο ευτός του. Προχωρώντας στην $2^{η}$ στηλη, δηλαδή τελικός αποδέκτης ο κόμβος 2, ο επόμενος αποδέκτης είναι ο κόμβος 2. Αρα ο κόμβος 1 για να στείλει στον κόμβο 2, ο επόμενος αποδέκτης είναι ο 2 που σημαίνει πως στέλνει κατευθείαν εκεί. Τέλος, $1^{η}$ γραμμή, $3^{η}$ στήλη (δηλαδή ο κόμβος 1 για να στείλει στον κόμβο 3), η πληροφορία πρεπει να περάσει από τον επόμενο αποδέκτη που είναι ο κόμβος 2. Εν συνεχεία, $2^{η}$ γραμμή αφορά τον κόμβο 2 σαν αποστολέα. Κάθε στήλη είναι ο τελικός αποδέκτης. $2^{η}$ γραμμή, $1^{η}$ στήλη (στοιχείο 2,1) αποστολέας κόμβος 2, τελικός αποδέκτης κόμβος 1, ο επόμενος αποδέκτης είναι ο κόμβος 1. Επίσης, $2^{η}$ γραμμή, $3^{η}$ στήλη (στοιχείο 2,3) αποστολέας κόμβος 2, τελικός αποδέκτης κόμβος 3, επόμενος αποδέκτης είναι ο κόμβος 3. Τέλος, $3^{η}$ γραμμή $1^{η}$ στήλη (στοιχείο 3,1) αποστολέας κόμβος 3, τελικός αποδέκτης κόμβος 1, ο επόμενος παραλήπτης θα είναι ο κόμβος 2. Αντίστοιχα, $3^{η}$ γραμμή $2^{η}$ στήλη (στοιχείο 3,2) αποστολέας κόμβος 3 τελικός αποδέκτης κόμβος 2, ο επόμενος παραλήπτης θα είναι ο κόμβος 2 που θα αναδρομολογήσει το πακέτο. Στην περίπτωση $3^{ης}$ γραμμής, $3^{ης}$ στήλης (στοιχείο 3,3) αποστολέας κόμβος 3 τελικός αποδέκτης κόμβος 3 ο επόμενος αποδέκτης είναι ο 0 επειδή δεν θα αποσταλλεί κανένα πακέτο επειδή ο αποστολέας και ο παραλήπτης ταυτίζονται.

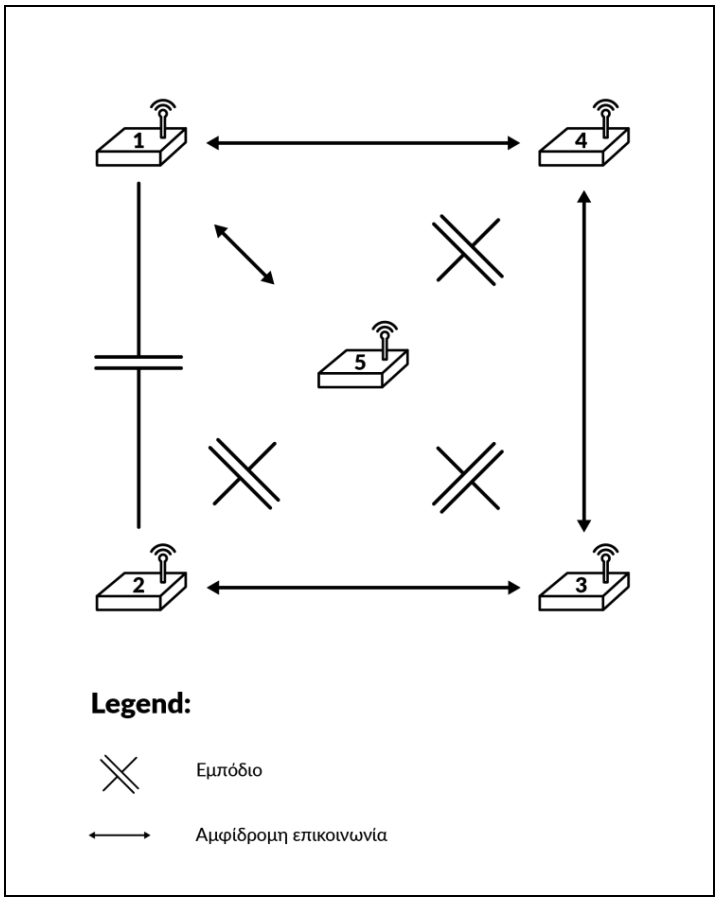
Εν κατακλείδι, το στοιχείο {m, n} του πίνακα περιέχει την τιμή του επόμενου αποδέκτη, αν ο αποστολέας είναι ο κόμβος m και τελικός αποδέκτης ο κόμβος n.

Στην πραγματικότητα το κανάλι επικοινωνίας μεταξύ κόμβων 2 και 3 δεν είναι αμφίδρομο αλλά μονόδρομο, δηλαδή ο κόμβος 3 δεν έχει την δυνατότητα να αποστέλλει πακέτα πληροφορίας παρά μόνο να λαμβάνει. Αυτό προκύπτει από κατασκευαστική αδυναμία ή κάποιας βλάβης που υπέστη κατά την μεταφορά, και είμαστε σε επικοινωνία με τον

κατασκευαστή για αντικατάστασή του. Εν ολίγοις, για οποιαδήποτε αποστολή πακέτου, ο κόμβος 2 είναι ενδιάμεσος και θα επαναδρομολογήσει πακέτο αν δεν είναι ο ίδιος ο τελικός αποδέκτης.

3.2.2 Ασύρματο δίκτυο πέντε (5) κόμβων

Στην συνέχεια παρουσιάζεται η περίπτωση ενός δικτύου 5 κόμβων, όπως φαίνεται στην Εικόνα 3.4.



Εικόνα 3.4: Ασύρματο δίκτυο 5 κόμβων με εμπόδια επικοινωνίας.

Η συγκεκριμένη τοπολογία δικτύου περιέχει 5 κόμβους με αρκετά εμπόδια επικοινωνίας. Λαμβάνοντας υπόψιν και τον πίνακα δρομολόγησης που ακολουθεί, μπορεί να παρατηρηθεί πως για να επικοινωνήσει ο κόμβος 1 με τον κόμβο 2 πρεπει το πακέτο επικοινωνίας να περάσει από τον κόμβο 4, να επαναδρομολογηθεί στον κόμβο 3 και στην συνέχεια να φτάσει στον κόμβο 2 που είναι και ο τελικός αποδέκτης. Αντίστοιχα για να στείλει ο κόμβος 2 στον κόμβο 5 το μονοπάτι είναι 2→3→4→1→5. Το σενάριο επικοινωνίας επιλέχθηκε τυχαία. Ακολουθεί ο πίνακας δρομολόγησης στο Σχήμα 3.5.

```
static int routes[5][5] = {
     {0,4,4,4,5},
     {3,0,3,3,3},
     {4,2,0,4,4},
     {1,3,3,0,1},
     {1,1,1,1,0}
     };
```

Εικόνα 3.5: Πίνακας δρομολόγησης σε δίκτυο πέντε (5) κόμβων.

Όπως και προηγουμένως, κάθε γραμμή αποτελεί τον κόμβο που βρίσκεται το πακέτο πληροφορίας και κάθε στήλη τον τελικό αποδέκτη. Ισχύει η ίδια λογική του επόμενου παραλήπτη με το στοιχείο {m, n}. Για παράδειγμα, 3η γραμμή, 2η στήλη (στοιχείο 3, 2) αποστολέας κόμβος 3 τελικός αποδέκτης κόμβος 2, ο επόμενος παραλήπτης είναι ο κόμβος 2. Επίσης, 1η γραμμή, 2η στήλη (στοιχείο 1, 2) αποστολέας κόμβος 1, τελικός αποδέκτης κόμβος 2, ο επόμενος παραλήπτης είναι ο κόμβος 4. Ο πίνακας δρομολόγησης έχει προκαθοριστεί σύμφωνα με τα εμπόδια ή τους λόγους που θέλουμε να κατευθύνεται η πληροφορία μέσα στο δίκτυο.

3.3 Επόμενος αποδέκτης (Next hop)

Όπως αναφέρθηκε και προηγουμένως, κάθε μήνυμα ταξιδεύει μέσα από προκαθορισμένο μονοπάτι όπως ορίζεται από τον πίνακα δρομολόγησης. Κάθε μήνυμα ξεκινά από τον αρχικό αποστολέα και συνεχίζει μέσα στο δίκτυο φέροντας πάντα την πληροφορία του τελικού αποδέκτη. Αυτό είναι αναγκαίο προφανώς για την ορθή μεταφορά του υπακούοντας τους κανόνες δρομολόγησης αλλά και για την σωστή καταμέτρηση στατιστικών όσο αφορά την αποστολή και την λήψη των πακέτων. Πρέπει να είναι ξεκάθαρο πόσα μηνύματα έστειλε κάθε κόμβος και πόσα τελικά έφτασαν στον προορισμό τους. Έτσι έχουμε ένα σωστό overview της λειτουργίας του δικτύου και μπορεί ο χρήστης να αποφανθεί από τα δεδομένα αυτά για ελλείψεις ή και προβλήματα στο δίκτυο με αποτέλεσμα την καλυτέρευση της ποιότητας επικοινωνίας μεταξύ των κόμβων αλλά και να ξεπεράσει διάφορα προβλήματα που μπορεί να προκύψουν (για παράδειγμα κόμβος εκτός λειτουργίας).

Ακολουθεί κώδικας που χρησιμοποιείται για τον προσδιορισμό του επόμενου παραλήπτη (next hop), Εικόνα 3.6. (Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
int findNextHop(int destination) { //συνάρτηση εύρεσης επόμενου κόμβου
     int routes[3][3] = { //πίνακας δρομολόγησης για 3 κόμβους
     {0,2,2},
     {1,0,3},
     {2,2,0}
     };

     int addr = chibiCmdStr2Num(destination, 16); // μετατροπή string από γραμμή εντολών σε ακέραιο με συγκεκριμένη βάση
     int x = destination -1; //υπολογισμός κόμβου προορισμού (το -1 υπολογίζεται διότι η αρίθμιση των κόμβων ξεκινά από το ¨0¨)
     int nextHop = routes[chibiGetShortAddr() -1][x]; //εύρεση συντεταγμένων από τον πίνακα δρομολόγησης και επομένως του επόμενου κόμβου
return nextHop;
}
```

Εικόνα 3.6: Κώδικας για τον υπολογισμό του next hop.

Με την συνάρτηση findNextHop υπολογίζεται από τον πίνακα δρομολόγησης ο επόμενος κόμβος. Με την μεταβλητή chibiGetShortAddr() μας απαντά ο κόμβος με την ορισμένη από τον χρήστη διεύθυνσή του και έτσι είναι εύκολο σε ενιαίο κώδικα να υπολογιστεί ο επόμενος κόμβος – παραλήπτης εφόσον γνωστοποιείται η διεύθυνση του κάθε κόμβου αλλά και η διεύθυνση του τελικού παραλήπτη όπως την δηλώνει ο χρήστης στην γραμμή εντολών.

3.3.1 Συνθήκη ελέγχου αν ο επόμενος κόμβος (next hop) είναι και ο τελικός παραλήπτης

Κάθε κόμβος του δικτύου αποθηκεύει τον κοινό πίνακα δρομολόγησης. Ο πίνακας φορτώνεται με την έναρξη (τροφοδοσία) του κάθε κόμβου και δηλώνει το μονοπάτι της πληροφορίας. Το πακέτο, έχοντας την πληροφορία του αποστολέα και του τελικού παραλήπτη (final destination), αποστέλλεται στον επόμενο παραλήπτη (next hop). Ο επόμενος παραλήπτης (next hop) που θα λάβει το πακέτο, θα συμβουλευτεί τον πίνακα δρομολόγησης και ελέγχοντας αν ο ίδιος είναι ή όχι και ο τελικός παραλήπτης θα σταματήσει την πορεία του πακέτου πληροφορίας εκεί ή θα επαναδρομολογήσει το πακέτο στον επόμενο κόμβο. Χωρίς αυτή τη συνθήκη ελέγχου, η πληροφορία θα χανόταν μέσα στο δίκτυο. (Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
if (finalDestination != chibiGetShortAddr()) { // συνθήκη ελέγχου τελικού παραλήπτη με διεύθυνση κόμβου που έλαβε το μήνυμα. Αν δεν υπάρχει ταύτιση
```

```
διευθύνσεων (είναι δηλαδή απλά ένας next hop και όχι ο final destination ο κόμβος που παρέλαβε),
int nextHop = findNextHop(finalDestination); // τότε ο επόμενος παραλήπτης αλλάζει σύμφωνα με τον πίνακα δρομολόγησης από την συνάρτηση findNextHop
chibiTx(nextHop, buf, strlen((char *)buf) + 1); // και το μήνυμα που μόλις έλαβε, επαναδρομολογείται μέχρι η συγκεκριμένη συνθήκη να μην δώσει αποτέλεσμα true.
}
```

Εικόνα 3.7: Συνθήκη ελέγχου next hop ως τελικός παραλήπτης.

3.4 Τελικός παραλήπτης (Final Destination)

Γενικότερα, ο τελικός παραλήπτης για την αποστολή ενός πακέτου ή ακόμα και μιας συστοιχίας πακέτων πληροφορίας, ορίζεται από τον χρήστη ή από μια διαδικασία παραγωγής ψευδοτυχαίας διεύθυνσης τελικού παραλήπτη για να εξομοιωθεί μια πιο τυχαία και φυσική αναπαράσταση ενός δικτύου.

Επομένως, όπως είναι λογικό, είτε στην μία είτε στην άλλη περίπτωση (εισαγωγή τελικού παραλήπτη από γραμμή εντολών από τον χρήστη ή παραγωγή τελικού παραλήπτη με τυχαίο τρόπο) χρειάζεται ο κώδικας του κάθε κόμβου να μπορεί να εξάγει τη πληροφορία που ο χρήστης εισάγει ή που παράγεται τυχαία.

Ακολουθεί κώδικας για τις συγκεκριμένες διαδικασίες.

3.4.1 Ο τελικός παραλήπτης δηλώνεται από το χρήστη:

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
void cmdSend(int arg_cnt, char **args)
{
    byte data[150];
    int addr, len;
    int x;

    int routes[3][3] = { // πίνακας δρομολόγησης για 3 κόμβους
{0,2,2},
{1,0,3},
{2,2,0}
```

```
};

addr = chibiCmdStr2Num(args[1], 16); // μετατροπή string από γραμμή εντολών καθώς και διεύθυνση παραλήπτη που γράφει ο χρήστης σε integer συγκεκριμένης βάσης (σημειώνεται πως η εντολή από τη γραμμή εντολών συντάσσεται: send 2 test, όπου 2 ο τελικός παραλήπτης και test η λέξη που στέλνει ο χρήστης στο δίκτυο)

len = strCat((char *)data, 1, arg_cnt, args); // συνένωση των δοσμένων strings από το χρήστη σε ένα ενιαίο που περιλαμβάνει στην πρώτη θέση την διεύθυνση του παραλήπτη και στην δεύτερη και μετά την λέξη-μήνυμα.
    x = addr -1; // αφαίρεση 1 από τον τελικό παραλήπτη που έχει δηλώνει ο χρήστης. Η αρίθμηση, υπενθυμίζεται, ξεκινά από το ¨0¨
    int nextHop = routes[chibiGetShortAddr() -1][x]; // μεταβλητή που περιλαμβάνει τον επόμενο παραλήπτη-κόμβο
    chibiTx(nextHop,data,len); // εντολή αποστολής μηνύματος στον επόμενο κόμβο με το συγκεκριμένο μήνυμα που εισήγαγε ο χρήστης

}
```

Εικόνα 3.8: Ο κώδικας δήλωσης τελικού παραλήπτη από τον χρήστη.

Είναι εμφανές πως στην συγκεκριμένη ρουτίνα η διεύθυνση τελικού παραλήπτη δηλώνεται από τον χρήστη στην γραμμή εντολών και έτσι είναι γνωστή σε όλη τη διαδικασία αποστολής του μηνύματος.

3.4.2 Ο τελικός παραλήπτης υπολογίζεται με τυχαίο τρόπο:

Για την συγκεκριμένη διαδικασία θα χρησιμοποιηθεί η γεννήτρια τυχαίων αριθμών, όπως περιγράφεται στο επόμενο υποκεφάλαιο 3.5.

3.5 Δημιουργία τυχαίας τιμής τελικού παραλήπτη

Όπως προαναφέρθηκε, η τυχαιότητα σε ένα δίκτυο και συγκεκριμένα στην συμπεριφορά του είναι σημαντικός παράγοντας. Συγκεκριμένα ο τυχαίος τελικός αποδέκτης, ο τυχαίος χρόνος που μεσολαβεί στην αποστολή πακέτων, το τυχαίο πακέτο πληροφορίας (με μέγεθος και σε τύπο, π.χ. μετήσεις από διάφορους αισθητήρες, όπως θερμοκρασίας, υγρασίας κλπ.) αποτελούν ίσως και μια πρόκληση για το δίκτυο. Με τον τρόπο αυτό, εσκεμμένα, θα μπορούσε ο δημιουργός του δικτύου να γνωρίσει τα φυσικά του όρια. Θα μπορούσε επίσης να υποβάλει το δίκτυο αυτό σε σκληρές δοκιμασίες που συναντούν τα κατασκευαστικά όρια των εκάστοτε

κόμβων και να μελετηθεί περαιτέρω η συμπεριφορά του. Έτσι προκύπτουν και οι ανάγκες για ανάπτυξη του δικτύου αλλά και οι περιπτώσεις για τις οποίες το δίκτυο δεν θα συμπεριφερθεί αξιόπιστα αλλά και οι περιπτώσεις αυτές που το δίκτυο θα σφάλλει. Αξίζει να σημειωθεί πως ο παράγοντας της τυχαιότητας στα δυο δίκτυα τριών και πέντε αισθητήρων αντίστοιχα που μελετήθηκαν στη συγκεκριμένη εργασία, ανέδειξε διάφορα ενδιαφέροντα αποτελέσματα τόσο στις μετρήσεις όσο και στην γενικότερη ιδέα για την κατασκευή αυτών των δικτύων. Συγκεκριμένα, ανέδειξε τα φυσικά όρια στην ταυτόχρονη αποστολή πακέτων από κάθε κόμβο, ανέδειξε την ανάγκη για έναν ενιαίο κώδικα και άκρως παραμετροποιήσιμο.

Ακολουθεί ο κώδικας της συνάρτησης παραγωγής τυχαίων αριθμών στο Εικόνα 3.9:

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
int random_number(int min_num, int max_num) // δήλωση συνάρτησης που σαν ορίσματα παίρνει το ελάχιστο και το μέγιστο όριο εύρους από τον χρήστη
{
    int result = 0, low_num = 0, hi_num = 0; // αρχικοποίηση αποτελέσματος τυχαίου αριθμού, ελάχιστου και μέγιστου εύρους

    if (min_num < max_num) // συνθήκη ελάχιστου και μέγιστου ορίου. Αν το ελάχιστο όριο είναι μικρότερο από το μέγιστο που έχει δηλώσει ο χρήστης τότε ορίζεται σαν low_num το ελάχιστο όριο και σαν hi_num το μέγιστο αυξημένο κατά 1 ώστε να μπορεί να χρησιμοποιηθεί και το ανώτατο όριο σαν τιμή στην έξοδο της συνάρτησης.
    {
        low_num = min_num;
        hi_num = max_num + 1;
    } else { // αλλιώς, σαν low_num ορίζεται το μέγιστο όριο αυξημένο κατά 1 (επεξηγήθηκε προηγουμένως γιατί) και σαν hi_num ορίζεται το κατώτατο όριο (αντίστροφα δηλαδή)
        low_num = max_num + 1;
        hi_num = min_num;
    }
```

```
    result = (rand() % (hi_num - low_num)) + low_num; // Τέλος, το τυχαίο αποτέλεσμα μεταξύ των ορίων που έχει δηλώσει ο χρήστης (ακέραιο φυσικά) είναι το υπόλοιπο της διαίρεσης της παραγωγής ενός τυχαίου αριθμού με τη διαφορά του ελαχίστου και του μεγίστου ορίου που έχει δηλώσει ο χρήστης, όπου προστίθεται το ελάχιστο αυτό όριο.
    return result;

}
```

Εικόνα 3.9: Συνάρτηση παραγωγής τυχαίων αριθμών.

Με την παραπάνω διαδικασία, παράγονται τυχαίοι ακέραιοι αριθμοί, για την ακρίβεια είναι ψευδοτυχαίοι, με την δυνατότητα να ορίσει ο χρήστης το εύρος τιμών και έτσι μπορεί να έχει μια τυχαία επιλογή τελικού αποδέκτη, τυχαία επιλογή χρόνου καθυστέρησης μεταξύ αποστελούμενων πακέτων πληροφορίας και άλλων πολλών ¨τυχαίων¨ τιμών για να μπορέσει να εξομοιώσει στο έπακρο και όσο πιο κοντά σε ένα πραγματικό περιβάλλον την συμπεριφορά του εκάστοτε δικτύου ασύρματων αισθητήρων.

Θα αναφερθούν παρακάτω, τόσο για το δίκτυο 3 κόμβων όσο και για το δίκτυο 5 κόμβων, οι περιπτώσεις για τις οποίες επιλέχθηκαν τυχαίοι ακέραιοι και ποιός σκοπός επιτεύχθηκε.

3.5.1 Παραγωγή τυχαίου τελικού αποδέκτη (Δίκτυο 3 κόμβων)

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
random1 = random_number (1,3); // συνάρτηση παραγωγής τυχαίων αριθμών με εύρος από 1 μέχρι και 3, όσοι και οι κόμβοι του δικτύου
int chibiId = chibiGetShortAddr(); // ορισμός μεταβλητής chibiId ως η διεύθυνση του ίδιου του κόμβου
  while (random1 == chibiId) { // συνθήκη ελέγχου αν ο παραχθείς τυχαίος αριθμός είναι ο ίδιος κόμβος, να ξανατρέχει η συνάρτηση παραγωγής. Δεν έχει ιδιαίτερη σημασία ο τυχαίος αριθμός που παράγεται δηλαδή ο κόμβος τελικού παραλήπτη, να είναι ο ίδιος ο κόμβος που βρισκόμαστε. Το μήνυμα δεν διαδίδεται, ο αποστολέας και ο παραλήπτης είναι ο ίδιος κόμβος.
    random1 = random_number (1,3); // επανάληψη παραγωγής τυχαίου αριθμού
  }

int addr= random1;
```

```
    char text[]=" test"; //δήλωση μηνύματος που θα αποσταλεί. Στη συγκεκριμένη περίπτωση η λέξη ¨test¨
    char strtemp[30]; // δήλωση μεταβλητής strtemp τύπου character (char)
    sprintf(strtemp,"%d", addr); // τοποθέτηση της παραχθείσας διεύθυνσης στη μεταβλητή strtemp
    strcat(strtemp, text); //ενσωμάτωση της λέξης test επίσης στη μεταβλητή strtemp

    x = addr -1;
    int nextHop = routes[chibiGetShortAddr() -1][x]; // υπολογισμός επόμενου κόμβου
    chibiTx(nextHop,strtemp,7); // αποστολή στην τυχαία διεύθυνση κόμβου (συγκεκριμένα στον επόμενο κόμβο με τελικό παραλήπτη την τυχαία διεύθυνση κόμβου) της επιθυμητής λέξης test
```

Εικόνα 3.10: Κώδικας τυχαίου τελικού αποδέκτη (δίκτυο 3 κόμβων).

3.5.2 Παραγωγή τυχαίου χρόνου καθυστέρησης στην αποστολή διαδοχικών πακέτων (Δίκτυο 3 κόμβων)

```
random2 = random_number (1,4); // επιλογή τυχαίου αριθμού από εύρους 1 μέχρι 4
delay(random2*100); // καθυστέρηση που ορίζεται σε ms. Εύκολα προκύπτει πως η καθυστέτρηση κυμαίνεται από 100ms μέχρι και 400ms.
```

Εικόνα 3.11: Κώδικας τυχαίας καθυστέρησης (δίκτυο 3 κόμβων).

3.5.3 Παραγωγή τυχαίου τελικού αποδέκτη (Δίκτυο 5 κόμβων)

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_5x5paper/Unicode_with_random_and_lobes_leds_v2107_5x5paper.ino)

```
random1 = random_number (1,5); // συνάρτηση παραγωγής τυχαίων αριθμών με εύρος από 1 μέχρι και 5, όσοι και οι κόμβοι του δικτύου
int chibiId = chibiGetShortAddr(); // ορισμός μεταβλητής chibiId ως η διεύθυνση του ίδιου του κόμβου
  while (random1 == chibiId) { // συνθήκη ελέγχου αν ο παραχθείς τυχαίος αριθμός είναι ο ίδιος κόμβος, να ξανατρέχει η συνάρτηση παραγωγής. Δεν έχει ιδιαίτερη σημασία ο τυχαίος αριθμός που παράγεται δηλαδή ο κόμβος τελικού παραλήπτη, να είναι ο ίδιος ο κόμβος που βρισκόμαστε. Το μήνυμα δεν διαδίδεται, ο αποστολέας και ο παραλήπτης είναι ο ίδιος κόμβος.
```

```
    random1 = random_number (1,5); // επανάληψη παραγωγής τυχαίου αριθμού
  }

int addr= random1;
    char text[]=" test"; //δήλωση μηνύματος που θα αποσταλεί. Στη συγκεκριμένη περίπτωση η λέξη ¨test¨
    char strtemp[30]; // δήλωση μεταβλητής strtemp τύπου character (char)
    sprintf(strtemp,"%d", addr); // τοποθέτηση της παραχθείσας διεύθυνσης στη μεταβλητή strtemp
    strcat(strtemp, text); //ενσωμάτωση της λέξης test επίσης στη μεταβλητή strtemp

    x = addr -1;
    int nextHop = routes[chibiGetShortAddr() -1][x]; // υπολογισμός επόμενου κόμβου
    chibiTx(nextHop,strtemp,7); // αποστολή στην τυχαία διεύθυνση κόμβου (συγκεκριμένα στον επόμενο κόμβο με τελικό παραλήπτη την τυχαία διεύθυνση κόμβου) της επιθυμητής λέξης test
```

Εικόνα 3.12: Κώδικας τυχαίου τελικού αποδέκτη (δίκτυο 5 κόμβων).

Με την συγκεκριμένη τεχνική της τυχαίας αποστολής πακέτων πληροφορίας όσο αφορά τον τελικό παραλήπτη μπορεί να ελεγχθεί η ορθή λειτουργία ενός δικτύου και να βελτιστοποιηθεί η συμπεριφορά του.

3.5.4 Παραγωγή τυχαίου χρόνου καθυστέρησης στην αποστολή διαδοχικών πακέτων (Δίκτυο 5 κόμβων)

```
random2 = random_number (1,4); // επιλογή τυχαίου αριθμού από εύρους 1 μέχρι 4
delay(random2*100); // καθυστέρηση που ορίζεται σε ms. Εύκολα προκύπτει πως η καθυστέρηση κυμαίνεται από 100ms μέχρι και 400ms.
```

Εικόνα 3.13: Κώδικας τυχαίας καθυστέρησης (δίκτυο 5 κόμβων).

Ακολουθείται η ίδια διαδικασία και με το δίκτυο των 3 κόμβων.

3.6 Αποστολή πολλαπλών πακέτων πληροφορίας (Mass packet sent)

Για τις ανάγκες του δικτύου, κρίθηκε αναγκαίο να υπάρξει η δυνατότητα να μπορεί να στείλει ένας κόμβος στο δίκτυο παραπάνω από ένα πακέτο πληροφορίας. Ως γνωστόν, σε ένα πραγματικό δίκτυο αισθητήρων, οι πληροφορίες που αποστέλλονται δεν είναι ποτέ μοναδικές.

Ένας αισθητήρας θερμοκρασίας, για παράδειγμα, δεν στέλνει ποτέ μια μόνο τιμή. Ρυθμίζεται έτσι ώστε να στέλνει σε τακτά χρονικά διαστήματα την θερμοκρασία έτσι ώστε να γίνεται πιο σωστά η επίβλεψη αυτής χωρίς να δίνει λανθασμένη εντύπωση σε περίπτωση για παράδειγμα πυρκαγιάς όπου και οι μεταβολές θα είναι ραγδαίες. Έτσι λοιπόν, με την παρακάτω συνάρτηση, μπορεί ο χρήστης να παραμετροποιήσει τον αριθμό των αποστελλούμενων πακέτων απλά και γρήγορα εισάγοντας βέβαια την όποια καθυστέρηση επιθυμεί μεταξύ των πακέτων. Στην Εικόνα 3.14 ακολουθεί ο σχετικός κώδικας:

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_5x5paper/Unicode_with_random_and_lobes_leds_v2107_5x5paper.ino)

```
void cmdSendMore(int arg_cnt, char **args) // συνάρτηση πολλαπλών πακέτων cmdSendMore
{
    byte data[150]; // Δήλωση και αρχικοποίηση μεταβλητών
    int addr, len;
    int x;
    int a=0;

while (a<30){ // Συνθήκη επανάληψης. Στην συγκεκριμένη περίπτωση δηλώνεται 30 επαναλήψεις που σηματοδοτεί την αποστολή 30 διαφορετικών πακέτων.
    // convert cmd line string to integer with specified base
    addr = chibiCmdStr2Num(args[1], 16);

    // concatenate strings typed into the command line and send it to
    // the specified address
    len = strCat((char *)data, 1, arg_cnt, args);
    x = addr -1;
    int nextHop = routes[chibiGetShortAddr() -1][x]; // εύρεση επόμενου παραλήπτη (next hop)
    //Serial.println(addr); // εμφάνιση στην οθόνη διαφόρων μεταβλητών για debug
    //Serial.println(nextHop);

    chibiTx(nextHop,data,len); //Αποστολή του πακέτου στον επόμενο παραλήπτη
    delay(1); // καθυστέρηση 1ms από το επόμενο πακέτο
```

```
    a++; // Μετρητής επαναλήψεων ώστε να μπορεί να ελέγχει η συνθήκη πότε θα
αγγίξει το όριο των 30 επαναλήψεων.
    }
```

Εικόνα 3.14: Κώδικας αποστολής πολλαπλών πακέτων πληροφορίας.

3.7 Λοβοί ακτινοβολίας

Η πρόταση της συγκεκριμένης εργασίας, όπως ήδη αναφέρθηκε στην Εισαγωγή, είναι η αξιοποίηση έξυπνων κεραιών για τη διασύνδεση των κόμβων του WSN. Αυτό σημαίνει πολύ απλά ότι κάθε μήνυμα θα μεταδίδεται στον επόμενο κόμβο και τελικά στον τελευταίο μέσω συγκεκριμένου λοβού ακτινοβολίας ο οποίος κάθε φορά θα ορίζεται από την σχετική θέση του εκάστοτε αποστολέα και ενδιάμεσου παραλήπτη. Ο λοβός αυτός θα επιλέγεται αυτόματα από τον κώδικα του κόμβου που αποστέλλει το μήνυμα σύμφωνα με διάφορους παράγοντες. Ένας από αυτούς και ο κυριότερος, είναι το μέγιστο δυνατό σήμα που προσφέρει. Σε ένα δίκτυο που έχει σταθερή τοπολογία, η επιλογή αυτή θα είναι σταθερή για κάθε ζεύγος αποστολέα-παραλήπτη. Ακόμα και σε ένα δίκτυο που δομή του αλλάζει, κρίνεται σχεδόν αναγκαίο σε επίπεδο ποιότητας επικοινωνίας να υπάρχει επιλογή για το καλύτερο δυνατό σήμα επικοινωνίας μεταξύ των κόμβων.

Στην παρούσα εργασία, ο λοβός που θα επιλέγεται θα υποδεικνύεται από μια φωτεινή οπτική ένδειξη LED διαφορετικού χρώματος. Οι λοβοί θα είναι 4 και σε κάθε αποστολή μηνύματος προς συγκεκριμένο κόμβο θα επιλέγεται αποκλειστικά ένας με την αντίστοιχη ένδειξη φωτεινής LED. Οι λαμπτήρες LED θα είναι συνδεδεμένοι σε ψηφιακές εξόδους των boards, στις οποίες αντίστοιχα θα συνδεθούν οι λογικές είσοδοι των κυκλωμάτων ελέγχου των κεραιών μεταγωγής λοβού, όταν αυτά είναι έτοιμα μελλοντικά.

Για να πραγματοποιηθεί αυτό, όπως είναι λογικό, γίνεται για μια ακόμη φορά η χρήση του πίνακα δρομολόγησης. Κοινοποιείται ο επόμενος παραλήπτης και με τη βοήθεια ενός 2^{ου} πίνακα που έχουμε ορίσει από πρίν, τον πίνακα επιλογής λοβού ακτινοβολίας, ορίζεται σε κάθε περίπτωση αποστολής μηνύματος ένας συγκεκριμένος λοβός. Έτσι στέλνεται το μήνυμα επιτυχώς από τον επιθυμητό λοβό. Να σημειωθεί πως ο πίνακας επιλογής λοβού ακτινοβολίας βρίσκεται στον κώδικα κάθε κόμβου και φορτώνεται με την έναρξή του. Όπως και στην περίπτωση του πίνακα δρομολόγησης, ο πίνακας επιλογής λοβού είναι κοινός για όλους τους κόμβους, δηλαδή κάθεκόμβος γνωρίζει τις επιλογές λοβού για όλους τους υπόλοιπους κόμβους.

Ακολουθεί ο συγκεκριμένος πίνακας του δικτύου 3 κόμβων όπως φαίνεται στην Εικόνα 3.15.

```
static int lobe_select[3][3] = {
  {0,1,4},
  {2,0,3},
  {4,4,0}
  };
```

Εικόνα 3.15: Πίνακας επιλογής λοβού ακτινοβολίας (δίκτυο 3 κόμβων).

Επίσης για την περίπτωση του δικτύου 5 κόμβων παρατίθεται ο πίνακας λοβού ακτινοβολίας με σχηματική απεικόνιση στην Εικόνα 3.16.

```
static int lobe_select[5][5] = {
  {0,4,4,4,3},
  {2,0,2,2,2},
  {2,4,0,2,2},
  {2,4,4,0,2},
  {2,2,2,2,0}
  };
```

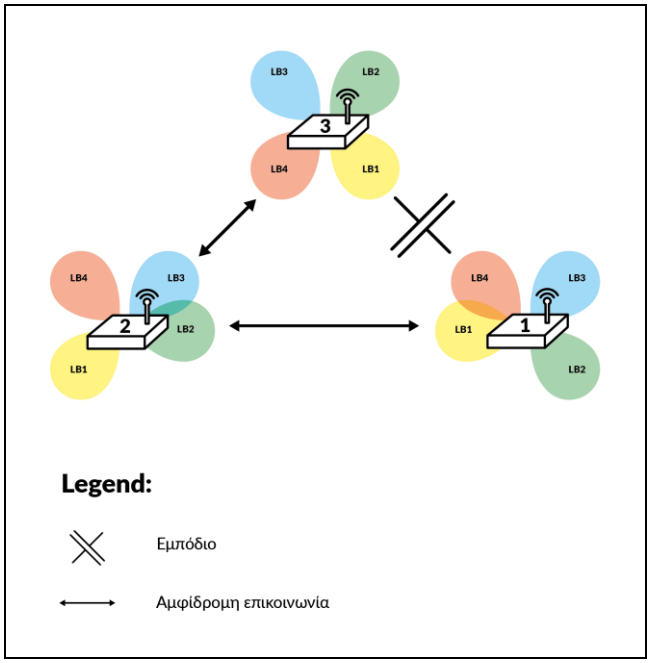
Εικόνα 3.16: Πίνακας επιλογής λοβού ακτινοβολίας (δίκτυο 5 κόμβων).

Η λογική για τη δημιουργία των παραπάνω πινάκων είναι η ίδια που ακολουθήθηκε και για τους πίνακες δρομολόγησης. Για παράδειγμα, στο δίκτυο 3 κόμβων, ο κόμβος 1 για να στείλει στον κόμβο 2 θα ενεργοποιήσει τον λοβό 1 και ο κόμβος 2 για να στείλει στον 3 θα ενεργοποιήσει τον λοβό 3.

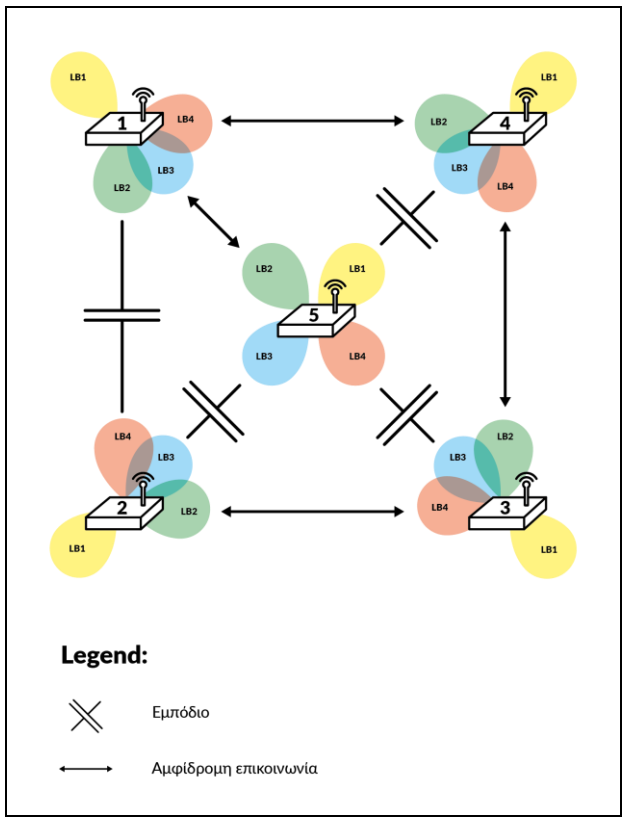
Αντίστοιχη λογική με το δίκτυο 5 κόμβων. Για παράδειγμα, ο κόμβος 1 για να στείλει στον κόμβο 5 θα ενεργοποιήσει τον λοβό 3 ενώ ο κόμβος 3 για να στείλει στον κόμβο 2 θα ενεργοποιήσει τον λοβό 4.

Η λογική είναι η ίδια με το στοιχείο {m, n}. Κάθε κόμβος που αποστέλει πληροφορία είναι η αντίστοιχη γραμμή του πίνακα ενώ ο τελικός αποδέκτης είναι πάντα η κατάλληλη στήλη του πίνακα. $4^{η}$ γραμμή $2^{η}$ στήλη (στοιχείο 4, 2) αποστολέας κόμβος 4 και τελικός αποδέκτης κόμβος κόμβος 2, θα ενεργοποιηθεί ο λοβός 4 όπως φαίνεται από το παραπάνω σχήμα. Παρακάτω, στη σχηματική απεικόνιση, φαίνονται οι λοβοί ακτινοβολίας με διαφορετικά χρώματα για πιο εύκολη κατανόηση. Κάθε λοβός έχει το δικό του χρώμα και κοιτά προς μια συγκεκριμένη κατεύθυνση.

Διαγράμματα δικτύων με εντυπωμένους λοβούς ακτινοβολίας:



Εικόνα 3.17: Δίκτυο 3 κόμβων με λοβούς ακτινοβολίας κάθε κόμβου (χρωματική σήμανση).



Εικόνα 3.18: Δίκτυο 5 κόμβων με λοβούς ακτινοβολίας κάθε κόμβου (χρωματική σήμανση).

Παρατίθεται ο κώδικας επιλογής λοβού ακτινοβολίας.

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

```
pinMode(7, OUTPUT); // δήλωση συγκεκριμένων pin ώς έξοδοι προς τροφοδοσία LED
pinMode(8, OUTPUT);
pinMode(4, OUTPUT);
pinMode(13, OUTPUT);

int led = routes[chibiGetShortAddr() -1][random1 -1]; // Εδώ επιλέγεται το ζευγάρι συντεταγμένων που θα χρησιμοποιηθεί για την ενεργοποίηση του LED. Στην ουσία προκύπτει από τον πίνακα δρομολόγησης και το ζεύγος τιμών είναι του κόμβου που είναι να στείλει μήνυμα και του τελικού παραλήπτη ώς ψευδοτυχαία επιλογή. Δηλαδή είναι η τιμή του επόμενου παραλήπτη (next hop)

Serial.print("The random address is: "); // εμφάνιση στην κονσόλα του ψευδοτυχαίου τελικού παραλήπτη
Serial.println(random1);
Serial.print("The random delay is: "); // εμφάνιση στην κοσνόλα του ψευδοτυχαίου αριθμού για την καθυστέρηση που πολ/ζεται επί 100 και προκύπτει η καθυστέρηση σε ms.
Serial.println(random2);
Serial.print("route for led drive with lobes: "); // εμφάνιση της τιμής led στην κονσόλα, υπολογισμένη προηγουμένως
Serial.println(led);
int lobe = lobe_select[chibiGetShortAddr() -1][led - 1]; // Υπολογισμός λοβού ακτινοβολίας. Προκύπτει από τον πίνακα lobe select που έχει δηλωθεί στην αρχή του κώδικα και είναι ο πίνακας επιλογής λοβού ακτινοβολίας.Ο πρώτος αριθμός του ζεύγους είναι η διεύθυνση του ίδιου του κόμβου. Ο δεύτερος είναι ο επόμενος παραλήπτης (next hop). Το ζευγάρι τιμών αντιστοιχεί στον λοβό δηλαδή LED που θα ενεργοποιηθεί.
Serial.print("The lobe is: "); // εμφάνιση στην κονσόλα του λοβού αντινοβολίας
Serial.println(lobe);

if (lobe == 1) { // Διαθέσιμοι 4 λοβοί ακτινοβολίας. Αν επιλεχθεί ο 1, ενεργοποιείται η έξοδος (pin) 7 στο board, τροφοδοτεί με τάση LED συγκεκριμένου χρώματος για οπτικλη διευκόλυνση
  digitalWrite(7, HIGH);   // turn the LED on (HIGH is the voltage level)
```

```
  delay(1000);                       // wait for a second
  digitalWrite(7, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
else if (lobe == 2) { // Αν επιλεχθεί ο 2, ενεργοποιείται η έξοδος (pin) 8
στο board, τροφοδοτεί με τάση LED συγκεκριμένου χρώματος για οπτική
διευκόλυνση
  digitalWrite(8, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(8, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
else if (lobe == 3) { // Αν επιλεχθεί ο 3, ενεργοποιείται η έξοδος (pin) 4
στο board, τροφοδοτεί με τάση LED συγκεκριμένου χρώματος για οπτική
διευκόλυνση
  digitalWrite(4, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(4, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
else if (lobe == 4) { // Αν επιλεχθεί ο 4, ενεργοποιείται η έξοδος (pin) 13
στο board, τροφοδοτεί με τάση LED συγκεκριμένου χρώματος για οπτική
διευκόλυνση
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
```

Εικόνα 3.19: Κώδικας επιλογής λοβού ακτινοβολίας.

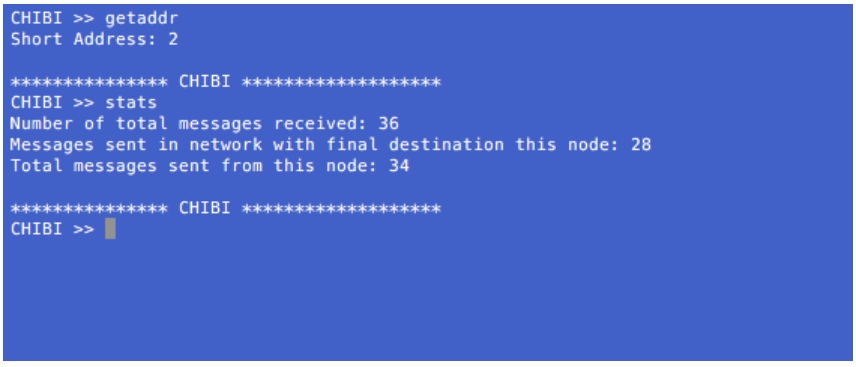
3.8 Στατιστικά δεδομένα δικτύου (statistical data)

Μία από τις πιο σίγουρες μεθόδους για επίβλεψη ενός δικτύου όσο αφορά την αποστολή και την λήψη πακέτων πληροφορίας είναι η καταγραφή αυτών. Κάθε κόμβος πρέπει να κρατά στατιστικά στοιχεία για τα πακέτα όχι μόνο που έστειλε αλλά και που έλαβε. Αν αυτό μπορεί να συμβαίνει σε κάθε κόμβο, τότε ο χρήστης μπορεί εύκολα να έχει μια ολοκληρωμένη εικόνα για την ποιότητα του δικτύου. Σε περίπτωση που τα στατιστικά στοιχεία δείξουν κάποια

ανωμαλία στο δίκτυο και συγκεκριμένα σε κάποιον κόμβο, τότε μπορεί να αντιμετωπίσει οποιοδήποτε πρόβλημα προκύψει πολύ πιο εστιασμένα.

Για τις ανάγκες των δικτύων 3 αλλά και 5 κόμβων, δημιουργήθηκαν τα παρακάτω στατιστικά στοιχεία:

- **Συνολικός αριθμός μηνυμάτων που ελήφθησαν στον συγκεκριμένο κόμβο. (Number of total messages received):** Ο κόμβος είναι ένα σημείο του δικτύου. Μηνύματα περνούν από αυτόν και είτε επαναδρομολογούνται από τον ίδιο μέχρι να φτάσουν στον τελικό αποδέκτη είτε ο ίδιος είναι ο τελικός αποδέκτης επομένως το ταξίδι του μηνύματος σταματά εκεί. Με αυτό το νούμερο ξέρουμε συνολικά πόσα μηνύματα έχουν περάσει τον κόμβο και είτε συνέχισαν είτε όχι.
- **Συνολικός αριθμός μηνυμάτων που ελήφθησαν στον συγκεκριμένο κόμβο με αυτόν ως τελικό παραλήπτη. (Messages sent in network with final destination this node):** Με αυτό το στατιστικό στοιχείο καταγράφονται τα μηνύματα που κατεφθασαν σε αυτόν τον κόμβο μονάχα αν αυτός ο κόμβος είναι και ο τελικός παραλήπτης. Σημαντική μέτρηση, εφόσον παίζει μεγάλο ρόλο στην αποστολή μηνυμάτων πόσα μηνύματα εφτασαν τελικά στον τελικό προορισμό τους.
- **Συνολικός αριθμός μηνυμάτων που έστειλε ο συγκεκριμένος κόμβος. (Total messages sent from this node):** Εδώ καταγράφεται ο αρθμός των μηνυμάτων που έφυγαν από τον συγκεκριμένο κόμβο. Δεν πρόκειτε για τα μηνύματα που επαναδρομολόγησε αλλά μονάχα για αυτά που γέννησε.



Εικόνα 3.20: Παράδειγμα εμφάνισης εντολής στατιστικών δεδομένων (stats).

Παραπάνω απεικονίζονται τα στατιστικά δεδομένα του κόμβου 2, έπειτα από μια δοκιμή αποστολής και λήψης πακέτων πληροφορίας. Σε κάποιο πειραματικό στάδιο μελέτης της συμπεριφοράς του δικτύου, ο κόμβος έλαβε συνολικά 36 μηνύματα εκ των οποίων στα 28 όπως

φαίνεται ήταν και ο τελικός αποδέκτης. Ο συγκεκριμένος κόμβος γέννησε συνολικά 34 μηνύματα. Με έναν εύκολο και γρήγορο υπολογισμό, εφόσον τα στατιστικά αυτά είναι διαθέσιμα σε κάθε κόμβο, μπορεί να είναι γνωστό πόσα μηνύματα έστειλε κάποιος προηγούμενος κόμβος με τον συγκεκριμένο σαν τελικό αποδέκτη. Εφόσον είναι γνωστό πόσα μηνύματα έλαβε ο ίδιος ο κόμβος ως τελικός αποδέκτης, προκύπτει εύκολα ο αριθμός των μηνυμάτων που χάθηκαν στην πορεία - σημαντική πληροφορία για την ομαλή λειτουργία του δικτύου.

3.9 Απώλεια πακέτων πληροφορίας και καθυστέρηση (Packet loss and delay)

Εχοντας πλέον διαθέσιμα τα παραπάνω στατιστικά μπορούμε να υπολογίσουμε εύκολα τυχούσες ελλείψεις που παρατηρούνται στο δίκτυο. Μπορεί κάποιος κόμβος να έστειλε κάποιο αριθμό πακέτων με τελικό παραλήπτη κάποιο άλλο κόμβο και να μην έφτασαν ποτέ όλα. Αυτό σημαίνει πως κάποιο πακέτο χάθηκε στην πορεία ή δεν αναδρομολογήθηκε ποτέ από τον ενδιάμεσο κόμβο στο δίκτυο. Αυτό μπορεί να συμβεί για πολλούς λόγους αλλά δύο είναι οι κυριότεροι: είτε δεν είχε επαρκές σήμα για να καταφέρει να αποστείλει και ο επόμενος κόμβος να παραλάβει είτε έγινε τόσο γρήγορη η επόμενη αποστολή πακέτου που δεν πρόλαβε το χαμένο πακέτο να παραληφθεί. Επίσης μπορεί την δεδομένη χρονική στιγμή να έστελναν πακέτα πληροφορίας διάφοροι άλλοι κόμβοι και να παρατηρήθηκε υπερφόρτωση στο δίκτυο. Έτσι λοιπόν, εισήχθη και μια νέα παράμετρος στην τυχαία αποστολή πακέτων στο δίκτυο, η καθυστέρηση. Η καθυστέρηση που εισάγεται είναι απο οριακά χαμηλή μέχρι αρκετά υψηλή έτσι ώστε να μπορούν να παρατηρηθούν φαινόμενα κορεσμού του καναλιού επικοινωνίας στο δίκτυο. Εκεί εχει σημασία να παρατηρηθούν τα στατιστικά στοιχεία πως μεταβάλλονται αλλά και πόσο μεγάλο ρόλο παίζει τελικά η καθυστέρηση στο δίκτυο. Δίνει τον επιθυμητό χρόνο σε κάθε κόμβο να διαχειριστεί τα πακέτα πληροφορίας τόσο αυτά που λαμβάνει όσο και αυτά που επαναδρομολογεί.

3.10 Επίδειξη λειτουργίας δικτύου (demo trial)

Για τις ανάγκες της παρουσίασης λειτουργίας τόσο του δικτύου 3 κόμβων όσο και για του δικτύου 5 κόμβων, χρησιμοποιείται μια συνάρτηση η οποία ονομάζεται SendMoreTrial. Στην συγκεκριμένη συνάρτηση έχουν ενσωματωθεί οι παράγοντες τυχαιότητας στην επιλογή τελικού αποδέκτη και τυχαιότητα στην χρονική καθυστέρηση μεταξύ των μηνυμάτων που παράγονται και αποστέλλονται. Είναι σε πλήρη ισχύ ο πίνακας δρομολόγησης για 3 ή 5 κόμβους όπως και ο

πίνακας επιλογής λοβού ακτινοβολίας για 3 ή 5 κόμβους. Επίσης για την επίδειξη έχει προοριστεί ο συνολικός αριθμός πακέτων πληροφορίας που θα αποσταλεί στο δίκτυο (συνολικά 10, για να μπορεί να υπάρξει και οπτική επιβεβαίωση με τα LED). Παρακάτω παρατίθεται συνολικά για μια πιο εμπεριστατωμένη άποψη ο σχετικός κώδικας της συνάρτησης SendMoreTrial όπου περιέχει κομμάτια κώδικα που έχει ήδη σχολιαστεί και επεξηγηθεί.

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_3x3paper/Unicode_with_random_and_lobes_leds_v2107_3x3paper.ino)

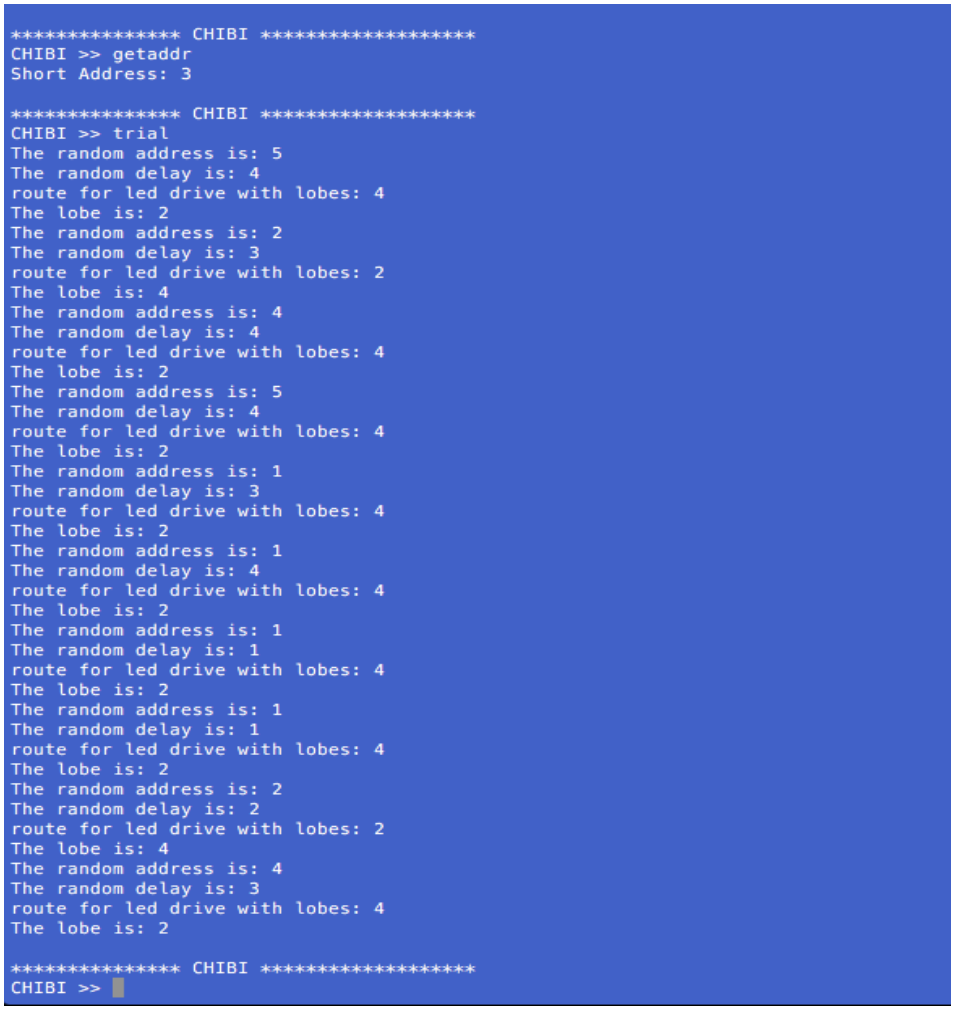
```
void cmdSendMoreTrial() // δήλωση συνάρτησης SendMoreTrial που μπορεί να εκτελεστεί από κονσόλα
{
    byte data[150];
    char str1[30];
    int len;
    int x, random1, random2;
    int a=0;


while (a<10){ // σε αυτό το σημείο, ο χρήστης μπορεί να δηλώσει αριθμό επαναλήψεων

pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(4, OUTPUT);
pinMode(13, OUTPUT);

    random1 = random_number (1,5); // για δίκτυο 5 κόμβων
int chibiId = chibiGetShortAddr();
  while (random1 == chibiId) {
    random1 = random_number (1,5);
  }

//concat addr and message to be sent
    int addr= random1;
    char text[]=" test";
    char strtemp[30];
    sprintf(strtemp,"%d", addr); //concat addr to strtemp
```

```
    strcat(strtemp, text); //combine text to addr

random2 = random_number (1,4); //for delay



    x = addr -1;
    int nextHop = routes[chibiGetShortAddr() -1][x];
    chibiTx(nextHop,strtemp,7);
    delay(random2*100);
    a++;

int led = routes[chibiGetShortAddr() -1][random1 -1];

Serial.print("The random address is: ");
Serial.println(random1);
Serial.print("The random delay is: ");
Serial.println(random2);
Serial.print("route for led drive with lobes: ");
Serial.println(led);
int lobe = lobe_select[chibiGetShortAddr() -1][led - 1];
Serial.print("The lobe is: ");
Serial.println(lobe);

if (lobe == 1) {
  digitalWrite(7, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(7, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
else if (lobe == 2) {
  digitalWrite(8, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(8, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}
else if (lobe == 3) {
  digitalWrite(4, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(4, LOW);    // turn the LED off by making the voltage LOW
```

```
  delay(50);
}
else if (lobe == 4) {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(50);
}

}
counter++;
temp2=counter*a;
}
```

Εικόνα 3.21: Κώδικας επίδειξης λειτουργίας (συνάρτησης SendMoreTrial).

Στο συγκεκριμένο σημείο ακολουθεί στιγμιότυπο εκτέλεσης της συνάρτησης από κονσόλα γραφοντας την λέξη trial.



Εικόνα 3.22: Επίδειξη εκτέλεσης εντολής trial από γραμμή εντολών από κόμβο 3.

Ουσιαστικά, για λόγους debug εμφανίζονται οι παραπάνω πληροφορίες αλλά κρίθηκε χρήσιμο να αναφέρονται σε κάθε κόμβο που τρεχει την trial για να επιβεβαιώνεται πως το δίκτυο λειτουργεί σωστά και επίσης να κοινοποιείται ο τελικός αποδέκτης και η τυχαία καθυστέρηση που εισάγεται. Επίσης βοηθά στο αν οδηγείται ο σωστός λοβός ακτινοβολίας. Φαίνονται ξεκάθαρα οι πληροφορίες για καθένα από τα 10 μηνύματα που παράγονται και ύστερα διοχετεύονται στο δίκτυο.

3.11 Επιλογή ισχύος εκπομπής κάθε κόμβου (Transmit Power)

Κατασκευαστικά, οι κόμβοι όπως έχει προαναφερθεί και στο πρώτο μέρος της εργασίας, στο εγχειρίδιο χρήσης, μέσω διαφόρων καταχωρητών μπορεί ο χρήστης να επηρρεάσει την λειτουργία των κόμβων σε διάφορα επίπεδα. Για παράδειγμα, να αλλάξει το κανάλι επικοινωνίας, να επιλέξει την ισχύ εξόδου για την εκπομπή σε κάθε κόμβο ξεχωριστά, να

επιλέξει από διάφορες διαμορφώσεις την επιθυμητή που χρησιμοποιείται για την αποστολή πακέτων πληροφορίας κ.α. Για θέματα αποστολής και λήψης πακέτων πληροφορίας, ιδιαίτερο ενδιαφέρον φαίνεται να παρουσιάζει η επιλογή της ισχύος εκπομπής. Στο δίκτυο 3 και 5 κόμβων, ανάλογα την γεωγραφική τοποθεσία κάθε κόμβου, μπορεί να χρειαστεί να αυξηθεί η ισχύς εκπομπής, αν παρατηρούνται αρκετά λάθη ή ελλείψεις στην διαμεταγωγή δεδομένων (ποιότητα σύνδεσης κόμβου με κόμβο) ή και να μειωθεί αυτή, σε περίπτωση που το κανάλι επικοινωνίας ποιοτικά χαρακτηρίζεται άριστο και ο χρήστης θέλει να περιορίσει στο ελάχιστο την κατανάλωση ενέργειας και ισχύος.

Ενδεικτικά θα αναφερθεί η δυνατότητα επιλογής της ισχύος εκπομπής κάθε κόμβου.

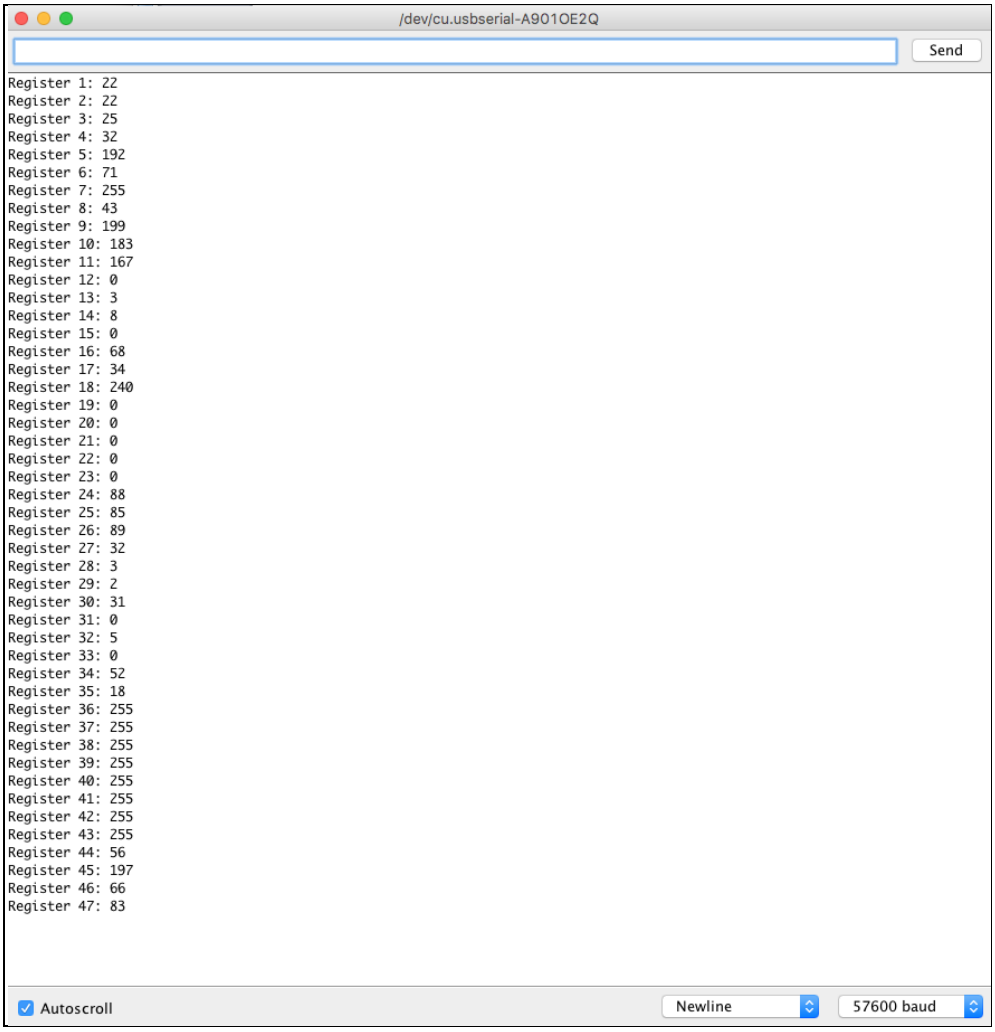
Ήδη από το 1^{ο} κεφάλαιο έχει περιγραφεί πώς μπορεί ο χρήστης να εξάγει τα δεδομένα βασικών καταχωρητών που επηρρεάζουν την λειτουργία των κόμβων. Δημιουργήθηκε συνάρτηση cmdShowRegisters η οποία μπορεί από γραμμή εντολών, εφόσον την καλέσει ο χρήστης, να επιστρέψει το περιεχόμενο των βασικών αυτών καταχωρητών. Ακολουθεί σχετικός κώδικας στην Εικόνα 3.23.

(Ο κώδικας βρίσκεται στο αρχείο: paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_5x5paper/Unicode_with_random_and_lobes_leds_v2107_5x5paper.ino)

```
void cmdShowRegisters() // συνάρτηση εμφάνισης στην κονσόλα βασικών καταχωρητών
{
 for (int i = 1; i < 48; i++) { // οι καταχωρητές είναι 47, μετρητής για 47 καταχωρήσεις
  byte data = chibiRegRead(i); // καταχώρηση τιμής κάθε καταχωρητή στην μεταβλητή data
  Serial.print("Register "); // εμφάνιση στην κονσόλα κάθε ένα καταχωρητή
  Serial.print(i);
  Serial.print(": ");
  Serial.println(data);
  }
 }
```

Εικόνα 3.23: Κώδικας εμφάνισης βασικών καταχωρητών.

Το αποτέλεσμα που εμφανίζεται στην κονσόλα του χρήστη θα είναι:



Εικόνα 3.24: Εμφάνιση βασικών καταχωρητών στην κονσόλα του χρήστη.

Συμβουλευόμενοι το manual του κατασκευαστή

(http://www.atmel.com/images/doc5131.pdf), εύκολα προκύπτει πως ο καταχωρητής για την επιλογή ισχύος εκπομπής είναι ο **Register 0x05 (PHY_TX_PWR)**. Οι διαθέσιμες τιμές που μπορεί να λάβει ο συγκεκριμένος καταχωρητής περιγράφονται από τον ακόλουθο πίνακα:

ΠΙΝΑΚΑΣ 3.1: Διαθέσιμες τιμές Καταχωρητή ισχύος εκπομπής.

Register Bits	Value [3:0]	Output Power [dBm]
TX_PWR	.0x0	+3.0
	0x1	+2.6
	0x2	+2.1
	0x3	+1.6
	0x4	+1.1
	0x5	+0.5

	0x6	-0.2
	0x7	-1.2
	0x8	-2.2
	0x9	-3.2
	0xA	-4.2
	0xB	-5.2
	0xC	-7.2
	0xD	-9.2
	0xE	-12.2
	0xF	-17.2

Καταχωρώντας την τιμή 0x0 στον καταχωρητή 0x05, ο χρήστης θα επιλέξει την ισχύ εξόδου στα +3dBm. Γι' αυτόν ακριβώς το λόγο δημιουργήθηκε η συνάρτηση cmdChangePower. Καλώντας την ο χρήστης επιλέγει την ισχύ εξόδου στη μέγιστη τιμή όπου είναι η 0x0. Η συνάρτηση περιέχει μονάχα μια εντολή η οποία απλά γράφει στον καταχωρητή την επιθυμητή τιμή.

(Ο κώδικας βρίσκεται στο αρχείο:

paper/Sketches/Unicode_with_random_and_lobes_leds_v2107_5x5paper/Unicode_with_random_and_lobes_leds_v2107_5x5paper.ino)

```
void cmdChangePower() // συνάρτηση επιλογής ισχύος εκπομπής
 {
  chibiRegWrite (0x05, 0x0); // εντολή καταγραφής στον καταχωρητή 0x05 την τιμή 0x0
 }
```

Εικόνα 3.25: Κώδικας καταγραφής επιθυμητής τιμής ισχύος.

ΚΕΦΑΛΑΙΟ 4:

Πιλοτική Εφαρμογή Δικτύων και Αποτελέσματα

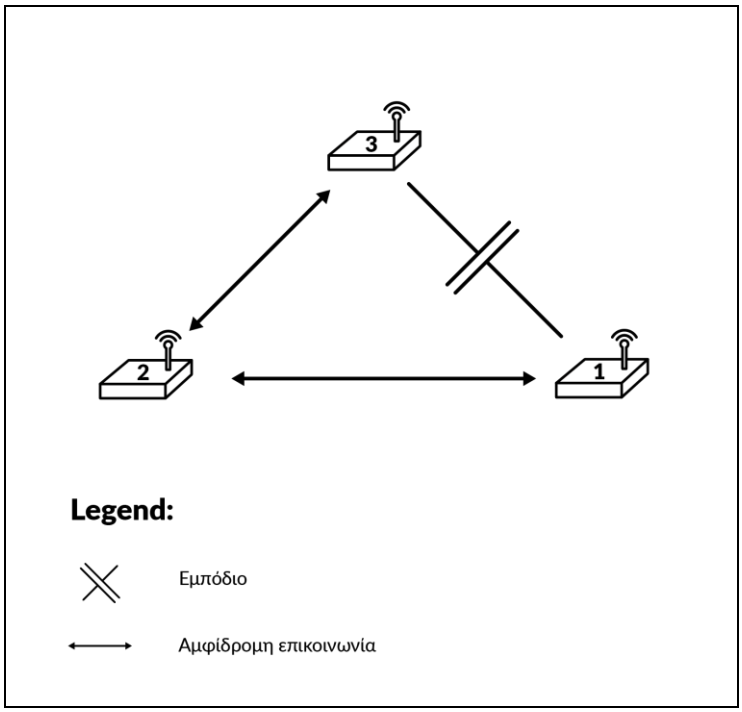
Στο παρόν κεφάλαιο θα αναλυθεί η ανάπτυξη και υλοπίηση του πιλοτικού δικτύου και θα επιδειχθεί η σωστή λειτουργία του μέσω αποτελεσμάτων. Τα αποτελέσματα αυτά λαμβάνονται μέσω της κονσόλας του χρήστη – που είναι και ο μόνος τρόπος επικοινωνίας με οποιονδήποτε κόμβο. Σκοπός είναι η καλύτερη κατανόηση των συναρτήσεων καθώς και εμφάνιση στην πράξη των βασικών λειτουργιών των δικτύων, όπως είναι π.χ. η απλή αποστολή πακέτων πληροφορίας, η δρομολόγηση των πακέτων αυτών, η ενεργοποίηση λοβών εκπομπής, η συγκέντρωση στατιστικών δεδομένων και η γενικότερη παρατήρηση του δικτύου.

4.1 Δίκτυο 3 κόμβων

Στην Εικόνα 4.1 εμφανίζεται το πιλοτικό δίκτυο 3 κόμβων, όπως αυτό υλοποιήθηκε και τέθηκε σε λειτουργία, για τις ανάγκες της συγκεκριμένης εργασίας.

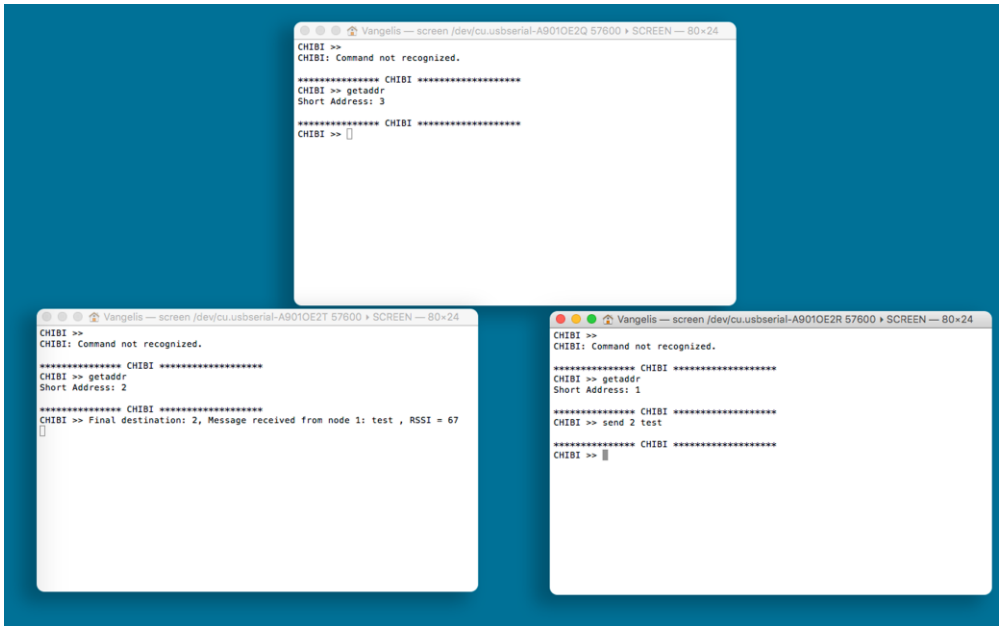
Εικόνα 4.1: Δίκτυο 3 κόμβων υλοποιημένο και σε λειτουργία.

Το δίκτυο βρίσκεται σε κατάσταση αναμονής. Ο χρήστης πλέον μπορεί να αρχίσει να βλέπει τη συμπεριφορά του στέλνοντας πακέτα πληροφορίας μέσα στο δίκτυο. Ακολουθεί απλή αποστολή πακέτου από κάποιο κόμβο σε ένα άλλο. Αναφορικά, οι κονσόλες καταγραφής συμβάντων έχουν τοποθετηθεί όπως είναι και η διάταξη στην Εικόνα 4.2 που ακολουθεί.



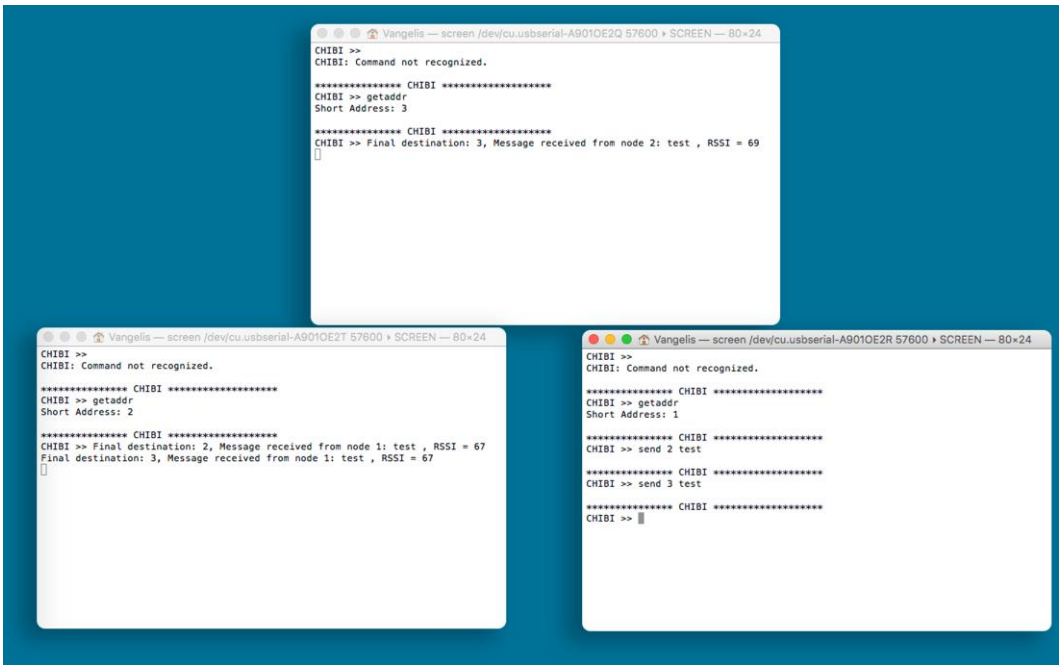
Εικόνα 4.2: Διάταξη δικτύου 3 κόμβων.

Στο σημείο αυτό, κάθε κόμβος αναγράφει την διεύθυνση του στην κοσνόλα του. Ο χρήστης στέλνει ένα πακέτο πληροφορίας, συγκεκριμένα τη λέξη test από τον κόμβο 1 στον κόμβο 2. Ο κόμβος 2 λαμβάνει το συγκεκριμένο μήνυμα και αναφέρονται στη κοσνόλα του κόμβου που έλαβε διάφορες πληροφορίες όπως ο τελικός παραλήπτης του συγκεκριμένου μηνύματος, ποιός κόμβος έστειλε το συγκεκριμένο μήνυμα, τί μήνυμα έστειλε και το RSSI (Received Signal Strength Indicator) του μηνύματος όπως φαίνεται στην Εικόνα 4.3.



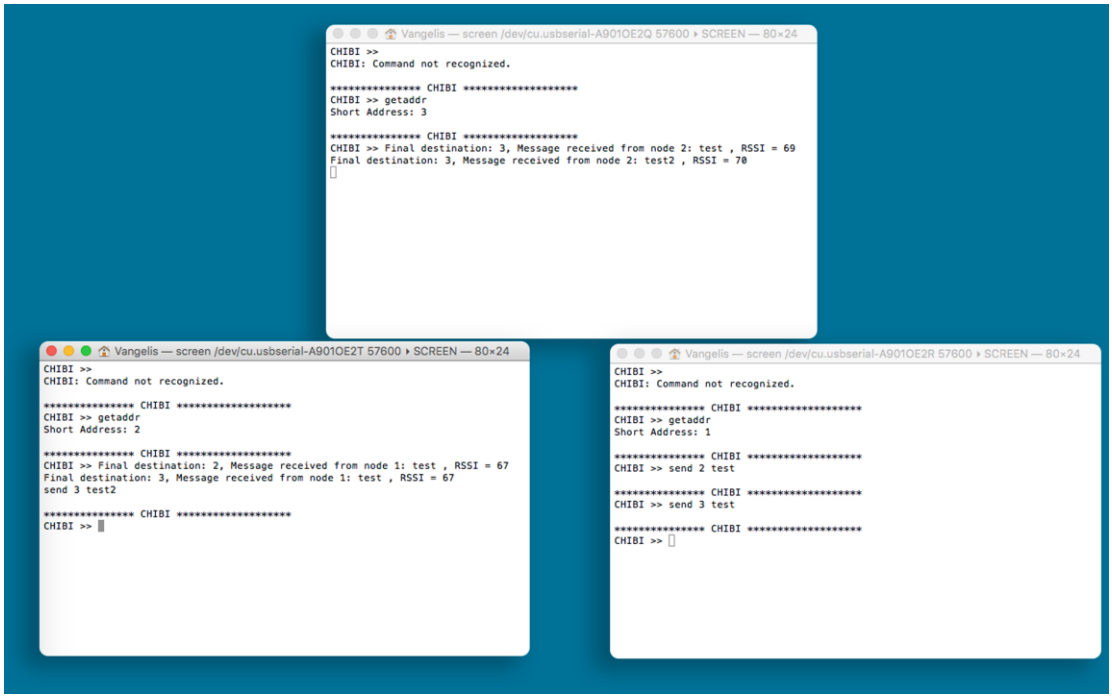
Εικόνα 4.3: Εντολή send και παραλαβή μηνύματος από τον επόμενο κόμβο.

Στην συνέχεια ο κόμβος 1 στέλνει πάλι ένα μήνυμα, αυτή τη φορά όμως αποδέκτης είναι ο κόμβος 3. Εδώ θα φανεί για $1^{η}$ φορά ο κανόνας δρομολόγησης. Σύμφωνα με αυτόν, το μήνυμα θα περάσει πρώτα από τον κόμβο 2 (next hop), ο οποίος θα επαναδρομολογήσει το μήνυμα προς τον κόμβο 3. Το εγχείρημα αυτό φαίνεται στην Εικόνα 4.4.



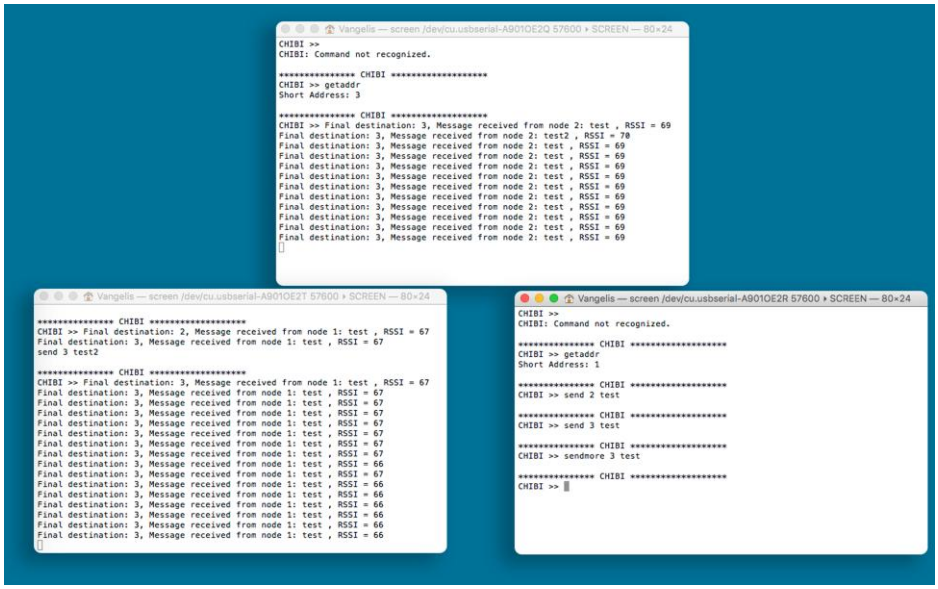
Εικόνα 4.4: Αποστολή μηνύματος σύμφωνα με τον κανόνα δρομολόγησης.

Όπως παρατηρούμε, το μήνυμα ελήφθη επιτυχώς από τον κόμβο 2 και στην συνέχεια το επαναδρομολόγησε στον κόμβο 3. Ο κόμβος 3 έλαβε με τη σειρά του το μήνυμα με τις αναγκαίες πληροφορίες που το ακολουθούν. Προς ολοκλήρωση της επίδειξης απλού μηνύματος, ο κόμβος 2 θα στείλει τώρα στον κόμβο 3. Ο πίνακας δρομολόγησης ορίζει πως ο επόμενος κόμβος για το μήνυμα αυτό είναι ο ίδιος ο κόμβος 3, όπως φαίνεται στην Εικόνα 4.5.



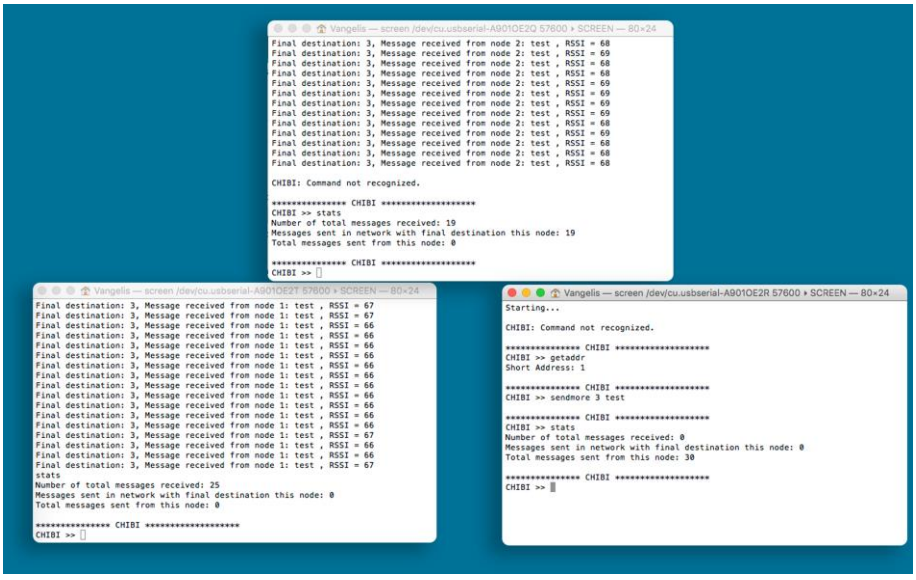
Εικόνα 4.5: Ο κόμβος 2 στέλνει επιτυχώς στον κόμβο 3.

Στη συνέχεια θα γίνει χρήση της εντολής sendmore όπου θα στείλει ο κόμβος 1 προς τον κόμβο 3 συνολικά ένα πακέτο 30 μηνυμάτων. Όπως έγινε και στην απλή αποστολή ενός πακέτου, περιμένουμε τα 30 μηνύματα να ληφθούν από τον κόμβο 2 και στη συνέχεια να προωθηθούν στον κόμβο 3. Το εγχείρημα αυτό φαίνεται στην Εικόνα 4.6.



Εικόνα 4.6: Ομαδική αποστολή πακέτων με δρομολόγηση.

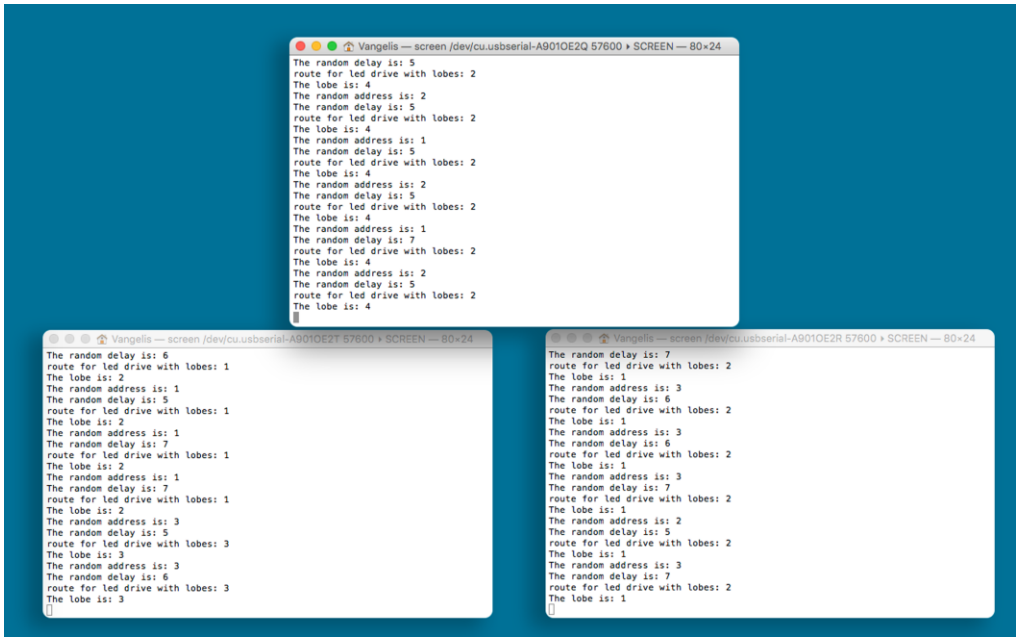
Πρεπει στο συγκεκριμένο σημείο να αναφερθεί πως η ομαδική αποστολή 30 μηνυμάτων πραγματοποιήθηκε χωρίς καθόλου καθυστέρηση μεταξύ των μηνυμάτων (συγκεκριμένα 1 ms). Όπως φαίνεται από την κονσόλα του κόμβου 2, έλαβε 16 μηνύματα και στη συνέχεια ο κόμβος 3 έλαβε 12 εξ αυτών ενώ ο καθένας έπρεπε να λάβει από 30 για μηδενικές απώλειες στο δίκτυο. Η επόμενη εικόνα, εικόνα 4.7, θα πραγματοποιηθεί ξανά το ίδιο πείραμα, αυτή τη φορά όμως με καθυστέρηση μεταξύ μηνυμάτων μισού δευτερολέπτου (500 ms).



Εικόνα 4.7: Ομαδική αποστολή πακέτων με δρομολόγηση και καθυστέρηση.

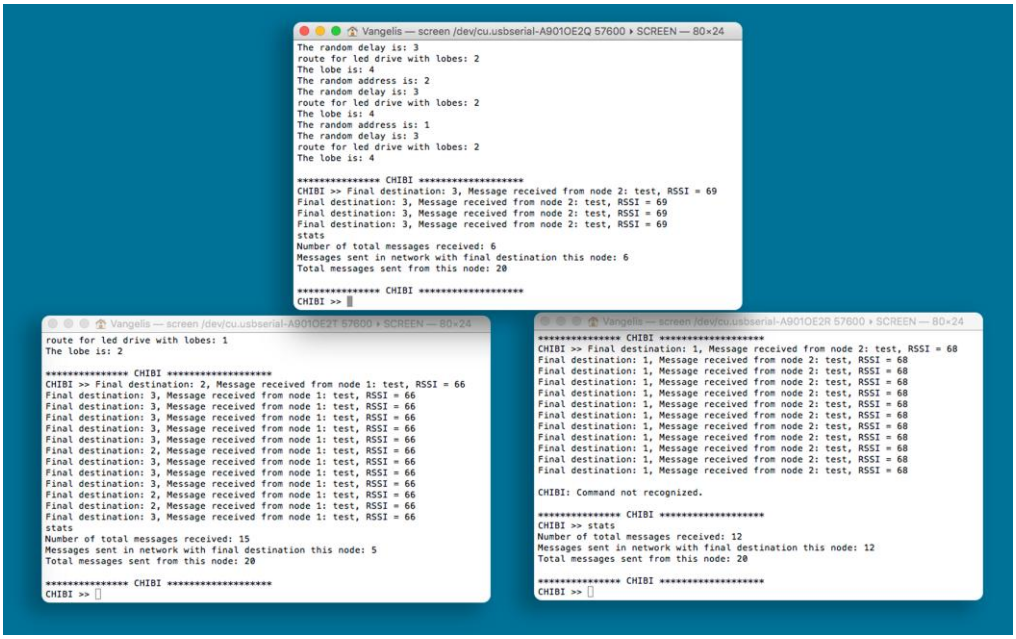
Όπως φαίνεται πλέον από την κονσόλα καταγραφής, ο κόμβος 2 έχει λάβει πλέον (με χρήση της εντολής στατιστικών δεδομένων stats) 25 πακέτα από τα 30 εν αντιθέση με προηγουμένως που έλαβε σχεδόν τα μισά (16 μηνύματα). Ο κόμβος 3 έλαβε στην συνέχεια τα μηνύματα που απέστειλε ο κόμβος 2 (25 συνολικά μηνύματα)και όπως φαίνεται από τη χρήση της εντολής stats έλαβε τα 19. Αυτό είναι προφανές, δεν προστέθηκε καθυστέρηση μεταξύ των μηνυμάτων που αποστέλει ο κόμβος 2 παρά μόνο στα μηνύματα του κόμβου 1.

Στο σημείο αυτό θα παρουσιαστεί η χρήση της εντολής trial όπου και θα φανεί στην πράξη η ενεργοποίηση των λοβών επικοινωνίας με την φωτεινή ενδειξη των LED όπως φαίνεται στην Εικόνα 4.8.



Εικόνα 4.8: Χρήση εντολής trial.

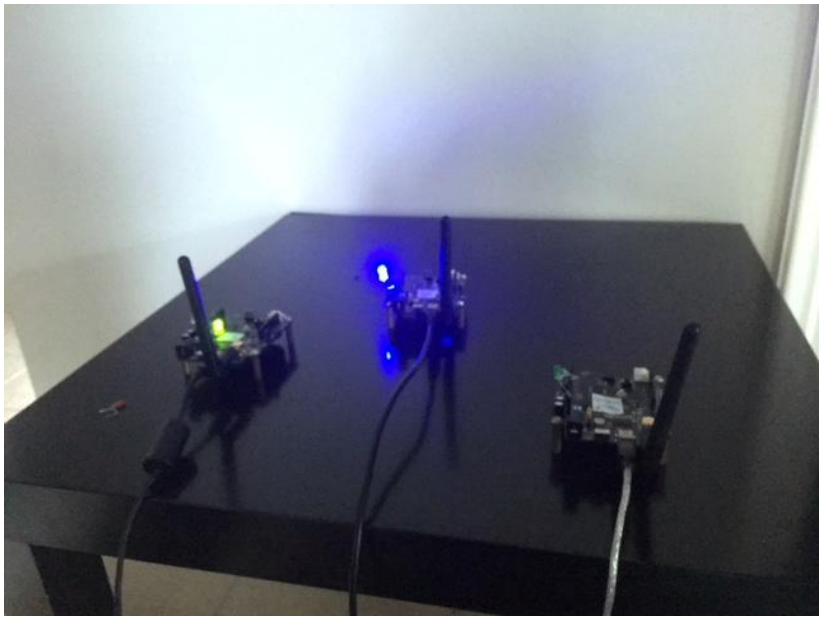
Όπως είναι φανερό, και οι 3 κόμβοι στέλνουν ακαταπαυστα μηνύματα (συνολικά 20 κάθε κόμβος), χρησιμοποιώντας συγκεκριμένους λοβούς με τυχαία καυστέρηση. Οι κανόνες δρομολόγησης ισχύουν φυσικά και οι κονσόλες καταγραφής και για τους 3 κόμβους αναφέρουν πληροφορίες σε κάθε μήνυμα που στέλνουν όπως ο τυχαίος παραλήπτης, η τυχαία καθυστέρηση που εισάγεται καθώς και ο λοβός που χρησιμοποιείται. Έπονται τα στατιστικά δεδομένα του κάθε κόμβου μετά την εντολή trial στην Εικόνα 4.9.



Εικόνα 4.9: Στατιστικά δεδομένα ύστερα από εντολή trial.

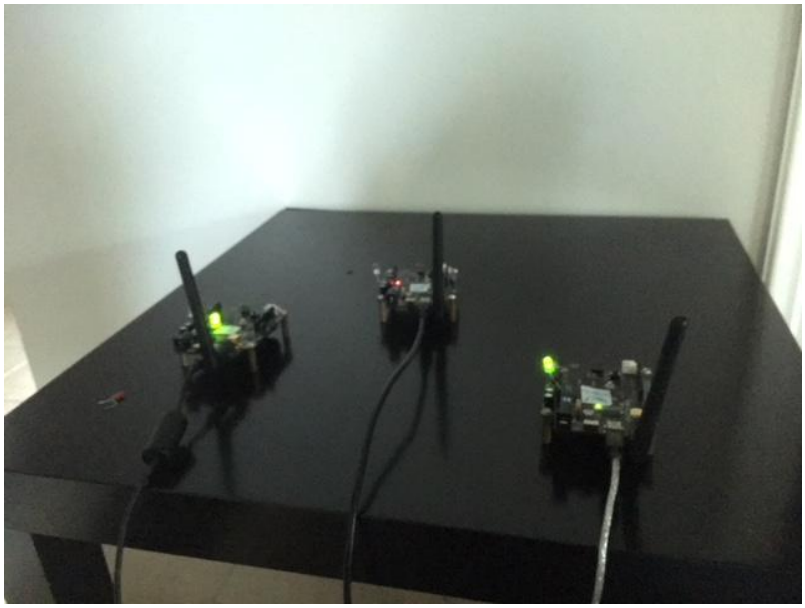
Στα δεδομένα κάθε κόμβου φαίνεται ο αριθμός μηνυμάτων που έλαβε και αν τελικά ήταν ο ίδιος ο τελικός παραλήπτης. Έτσι προκύπτει και ο αριθμός μηνυμάτων που επαναδρομολόγησε. Ο συνολικός αριθμός μηνυμάτων κάθε κόμβου δεν είναι αυτός που έστειλε ο προηγούμενος και αυτό οφείλεται και στην καθυστέρηση (συνήθως όταν είναι μικρή) αλλά και στην επιβάρυνση του δικτύου που κατακλύζεται με μηνύματα. Κάθε κόμβος στέλνει ασταμάτητα και επίσης είναι υπεύθυνος για τυχούσες επαναδρομολογήσεις. Είναι φυσικό κάποια πακέτα να χαθούν και να μην καταγραφούν.

Τέλος, στις παρακάτω Εικόνες 4.10 – 4.13 φαίνονται οι κόμβοι με ενεργούς λοβούς την ώρα της αποστολής μηνυμάτων. Κάποιοι εκπέμπουν ταυτόχρονα με κάποιους άλλους όπως είναι λογικό, μιας και η trial ετρεξε ταυτόχρονα σε όλους.



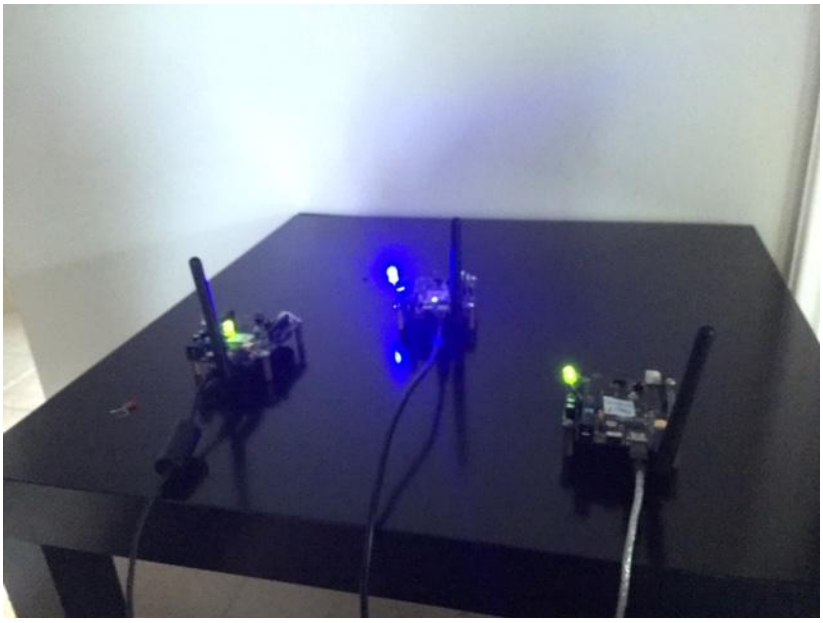
Εικόνα 4.10: Ενεργοί λοβοί σε κόμβους 2 και 3.

Στη συγκεκριμένη περίπτωση βλέπουμε πως οι κόμβοι 2 και 3 εκπέμπουν από συγκεκριμένους λοβούς. Ο κόμβος 1 στην συγκεκριμένη χρονική στιγμή δεν εκπέμπει κάποιο μήνυμα αλλά σίγουρα λαμβάνει από τα μηνύματα που στέλνουν οι κόμβοι 2 και 3. Αυτό βέβαια δεν αποτρέπει τους κόμβους 2 και 3 ενώ ταυτόχρονα εκπέμπουν, να λαμβάνουν επίσης. Υπό συγκεκριμένες χρονικές στιγμές, όλοι οι κόμβοι παίζουν τον ρόλο και του αποδέκτη αλλά και του αποστολέα.



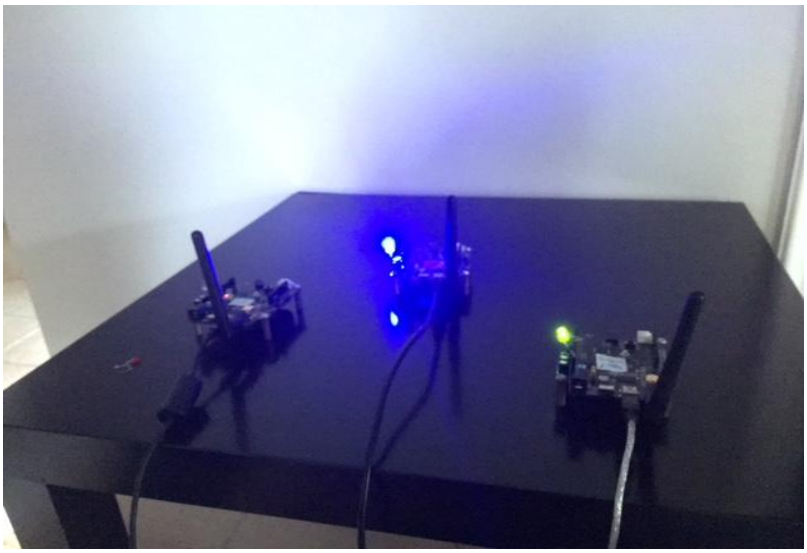
Εικόνα 4.11: Ενεργοί λοβοί σε κόμβους 1 και 2.

Σε αυτό το στιγμιότυπο, ο κόμβος 1 και 2 βρίσκονται σε κατάσταση αποστολής μηνυμάτων, γι' αυτό και παρατηρούμε ενεργούς λοβούς ακτινοβολίας. Ο κόμβος 3 δεν εκπέμπει αλλά όπως και πριν σίγουρα λαμβάνει. Σε επόμενο στιμιότυπο αυτό μπορεί να αλλάξει μιας και οι αποδέκτες εντελώς τυχαίοι.



Εικόνα 4.12: Ενεργοί λοβοί σε κόμβους 1, 2 και 3.

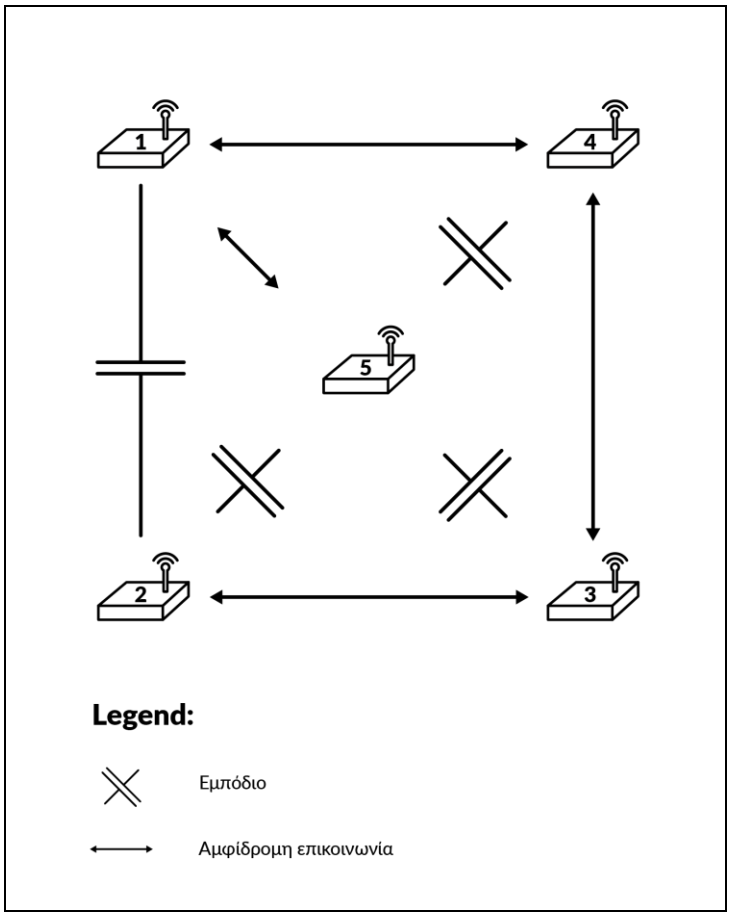
Και οι 3 κόμβοι βρίσκονται σε κατάσταση αποστολής ταυτόχρονα βέβαια με τη λήψη. Και οι 3 κόμβοι έχουν ενεργούς λοβούς ακτινοβολίας όπως φαίνεται στην Εικόνα 4.13.



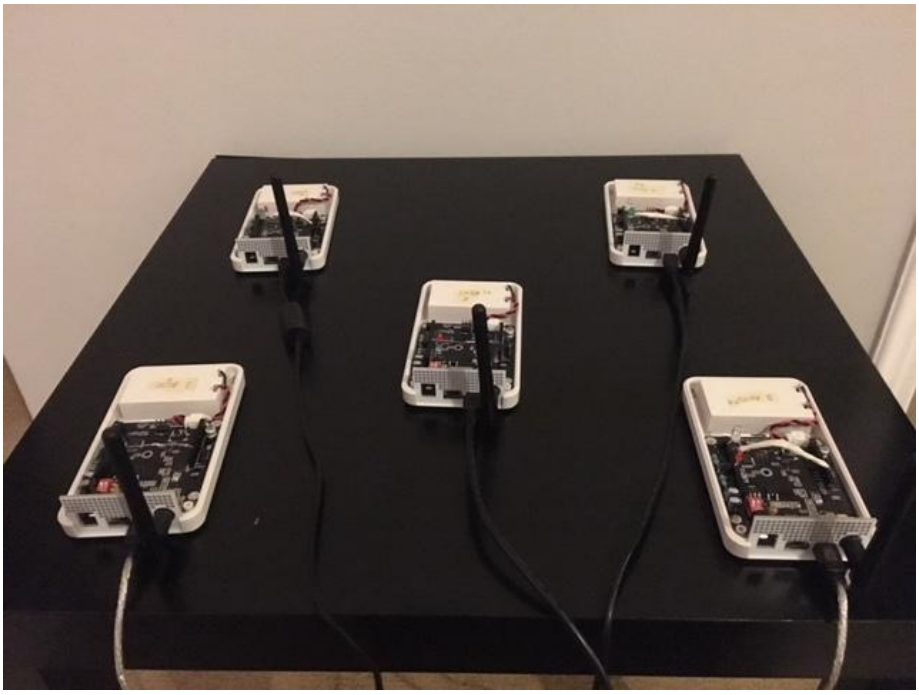
Εικόνα 4.13: Ενεργοί λοβοί σε κόμβους 1 και 3.

4.2 Δίκτυο 5 κόμβων

Στην Εικόνα 4.14 εμφανίζεται η διάταξη του δικτύου 5 κόμβων όπως αυτό υλοποιήθηκε πιλοτικά. Ακολουθεί επεξήγηση και επίδειξη του δικτύου αυτού με τις βασικές εντολές λειτουργίας του από το περιβάλλον εντολών (κονσόλα) και καταγραφή της συμπεριφοράς του. Το δίκτυο απεικονίζεται στην Εικόνα 4.14 με πιστή αντιγραφή στην τοποθεσία κάθε κόμβου από τη σχηματική διάταξη.

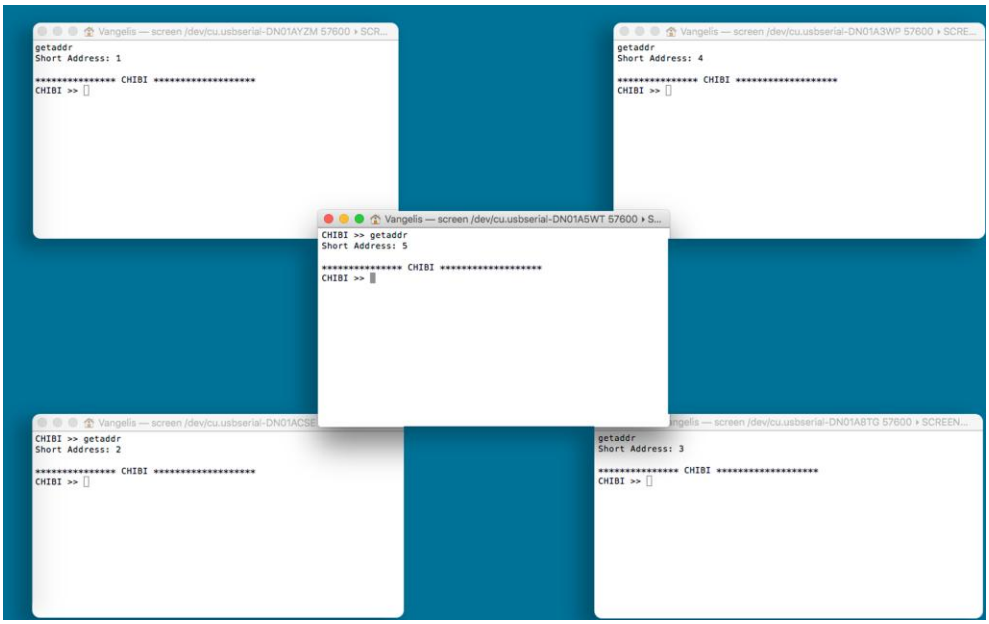


Εικόνα 4.14: Διάταξη δικτύου 5 κόμβων.



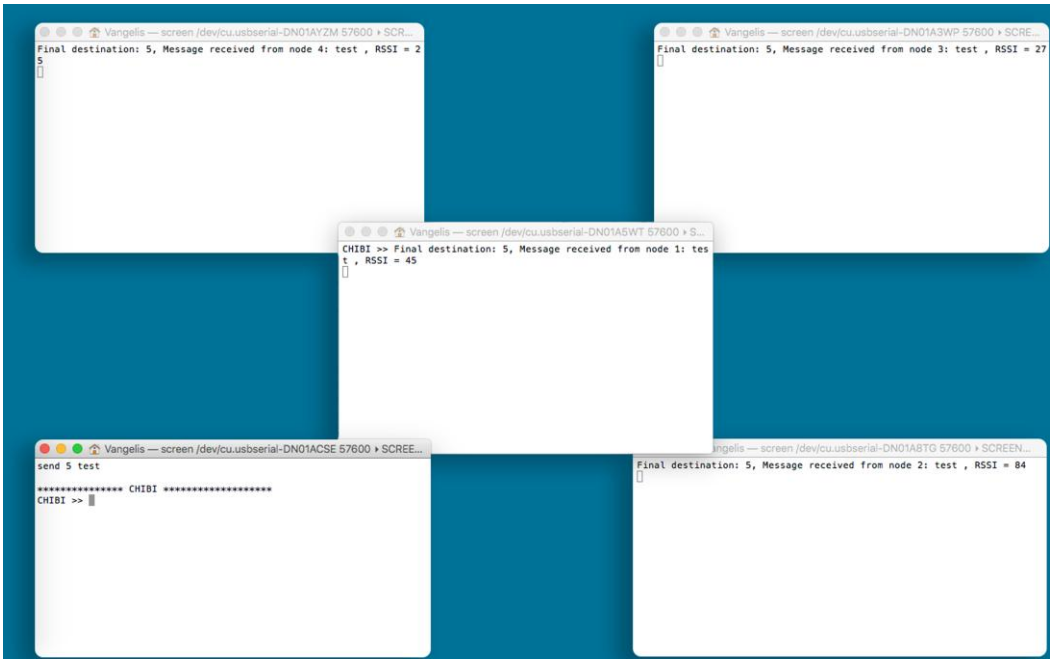
Εικόνα 4.15: Το δίκτυο 5 κόμβων όπως αυτό υλοποιήθηκε.

Στην συνέχεια θα γίνει αναφορά στο δίκτυο από τις κονσόλες καταγραφής συμβάντων κάθε κόμβου. Για μεγαλύτερη οπτική ευκολία, οι κονσόλες έχουν τοποθετηθεί και αυτές χωροταξία όμοια με τη φυσική διάταξη των κόμβων, όπως φαίνεται στην εικόνα 4.16.



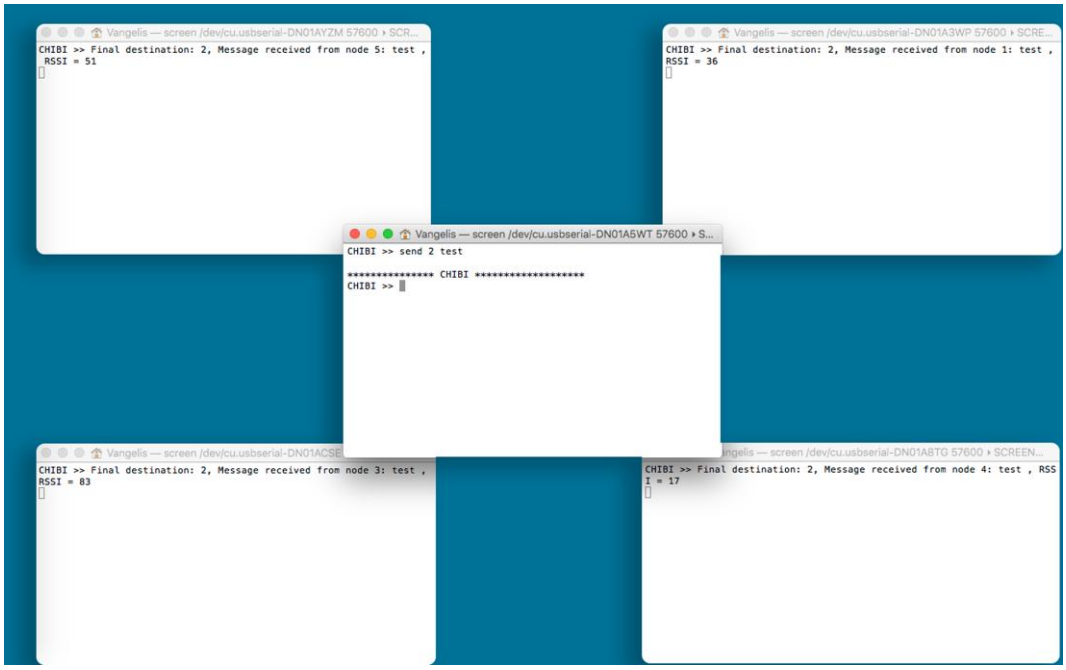
Εικόνα 4.16: Κονσόλες καταγραφής συμβάντων κάθε κόμβου.

Αναλύεται στη συνέχεια η απλή αποστολή ενός μηνύματος μεταξύ των κόμβων, μέσω ενός παραδείγματος. Η αποστολή μηνύματος του παραδείγματος θα είναι από τον κόμβο 2 προς τον κόμβο 5. Ο λόγος που επιλέχτηκαν οι συγκεκριμένοι κόμβοι είναι η διαδρομή του μηνύματος. Σύμφωνα με τον πίνακα δρομολόγησης, η διαδρομή είναι η εξής: κόμβος 2 → κόμβος 3 → κόμβος 4 → κόμβος 1 → κόμβος 5. Ετσι θα μπορούμε εύκολα να διαπιστώσουμε αν οι κανόνες δρομολόγησης τηρούνται. Ακολουθεί η σχετική εικόνα 4.17.



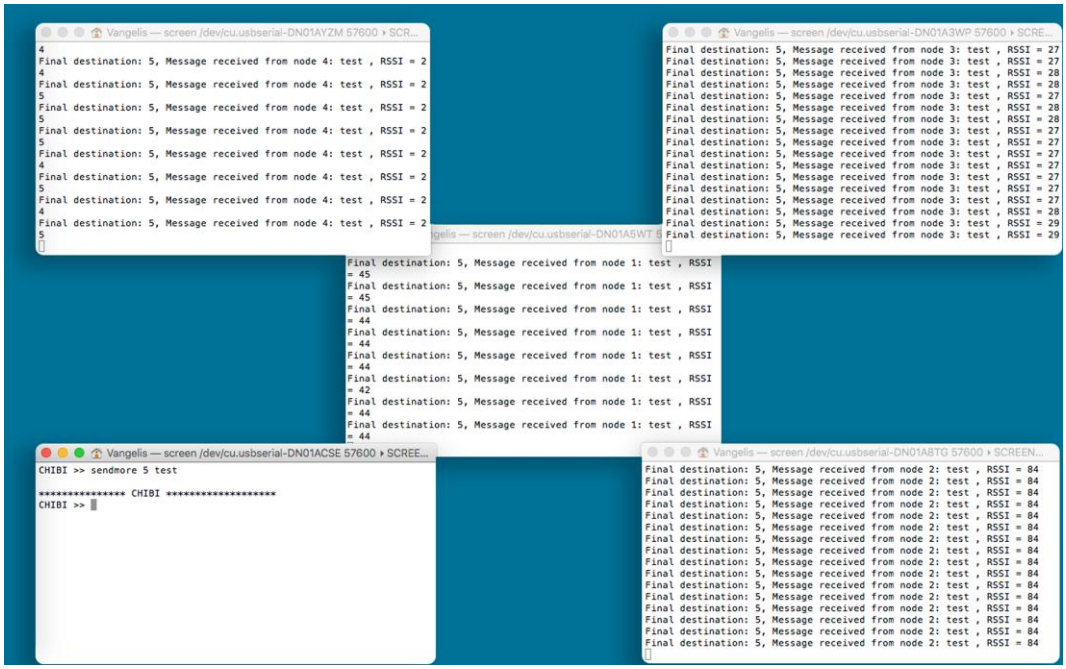
Εικόνα 4.17: Χρήση εντολής send για αποστολή μηνύματος από κόμβο 2 σε κόμβο 5.

Στην συνέχεια θα παρουσιαστεί το ίδιο πείραμα αντίστροφα, δηλαδή, ο κόμβος 5 θα στείλει μήνυμα προς τον κόμβο 2. Με την αντίστροφη διαδικασία, ολοκληρώνουμε τον έλεγχο για την σωστή δρομολόγηση, εφόσον έχουμε επιλέξει τη μεγαλύτερη διαδρομή μέσα στο δίκτυό μας και αυτή είναι μεταξύ των κόμβων 2 και 5. Ακολουθεί απεικόνιση στην εικόνα 4.18.



Εικόνα 4.18: Χρήση εντολής send για αποστολή μηνύματος από κόμβο 5 στον κόμβο 2.

Στη συνέχεια, το δίκτυο θα δοκιμαστεί σε πολλαπλή αποστολή μηνυμάτων με χρήση της εντολής sendmore. Έτσι θα καταγραφεί η επίδοσή του στην ομαδική αποστολή μηνυμάτων πληροφορίας και θα ελεγχθεί η ποιότητα επικοινωνίας καθώς, μετά την πολλαπλή αποστολή, με χρήση της εντολής stats, θα μπορέσουν να καταγραφούν στατιστικά δεδομένα από την εκτέλεση της εντολής sendmore. Ακολουθούν οι εικόνες 4.19 και 4.20 με τη χρήση εντολών sendmore και stats αντίστοιχα.



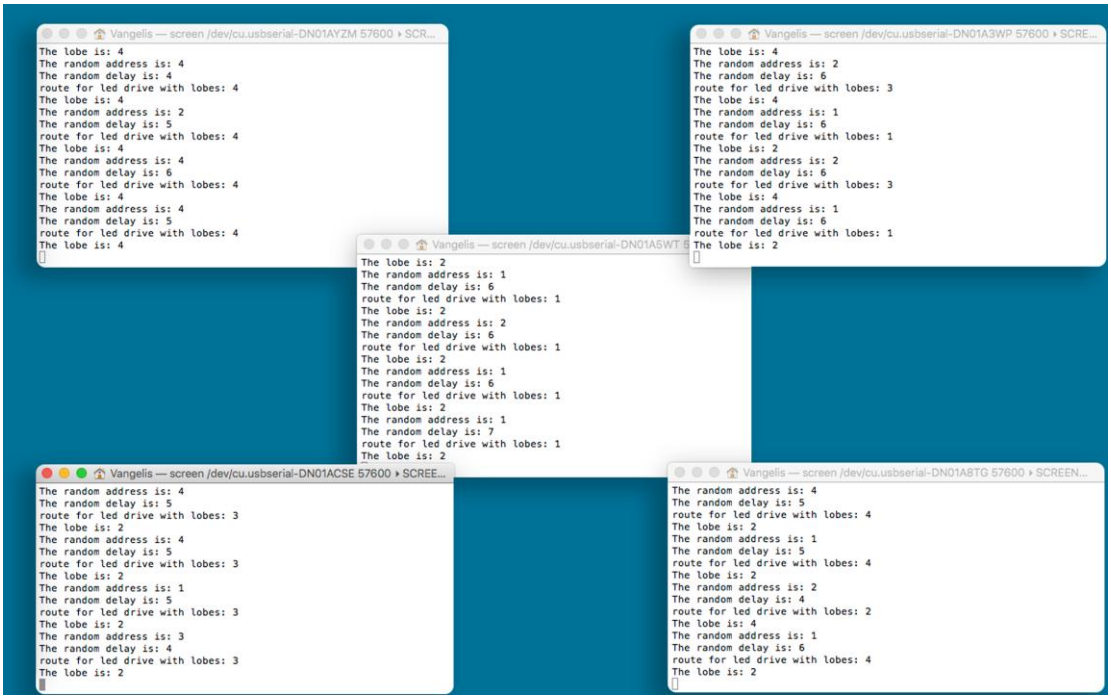
Εικόνα 4.19: Πολλαπλή αποστολή μηνυμάτων από κόμβο 2 στον κόμβο 5.

Όπως φαίνεται από τα παραπάνω στιγμιότυπα, τόσο η χρήση της εντολής send (απλή αποστολή μοναδικού μηνύματος) όσο και η χρήση της sendmore (πολλαπλή αποστολή μηνυμάτων) πλαισιώνονται από τους κανόνες δρομολόγησης. Τα μηνύματα ακολουθούν την προκαθορισμένη οδό που ορίζει ο πίνακας δρομολόγησης. Τα μηνύματα δρομολογούνται στον επόμενο παραλήπτη (next hop) μέχρι να φτάσουν τον τελικό τους παραλήπτη.

Εικόνα 4.20: Προβολή στατιστικών δεδομένων ύστερα από πολλαπλή αποστολή μηνυμάτων.

Συνολικά η εντολή sendmore στέλνει 30 μηνύματα. Από τον κόμβο 2 στον κόμβο 3, όπως φαίνεται στα στατιστικά δεδομένα κάθε κόμβου, χάθηκαν και δεν καταγράφτηκαν 5 μηνύματα. Στη συνέχεια, από τον κόμβο 3 προς τον τελικό κόμβο 5, δεν χάθηκε κανένα. Κάθε κόμβος μετά τον 2 έχει λάβει από 25 μηνύματα. Αυτό μπορεί να συνέβη κατά τη διάρκεια μιας στιγμιαίας παρεμβολής στο κανάλι ή ακόμα και λόγω καθυστέρησης, οπότε 5 μηνύματα μπορεί να έφθασαν γρηγορότερα απ' ό,τι χρειαζόταν ο αποδέκτης για την καταγραφή τους.

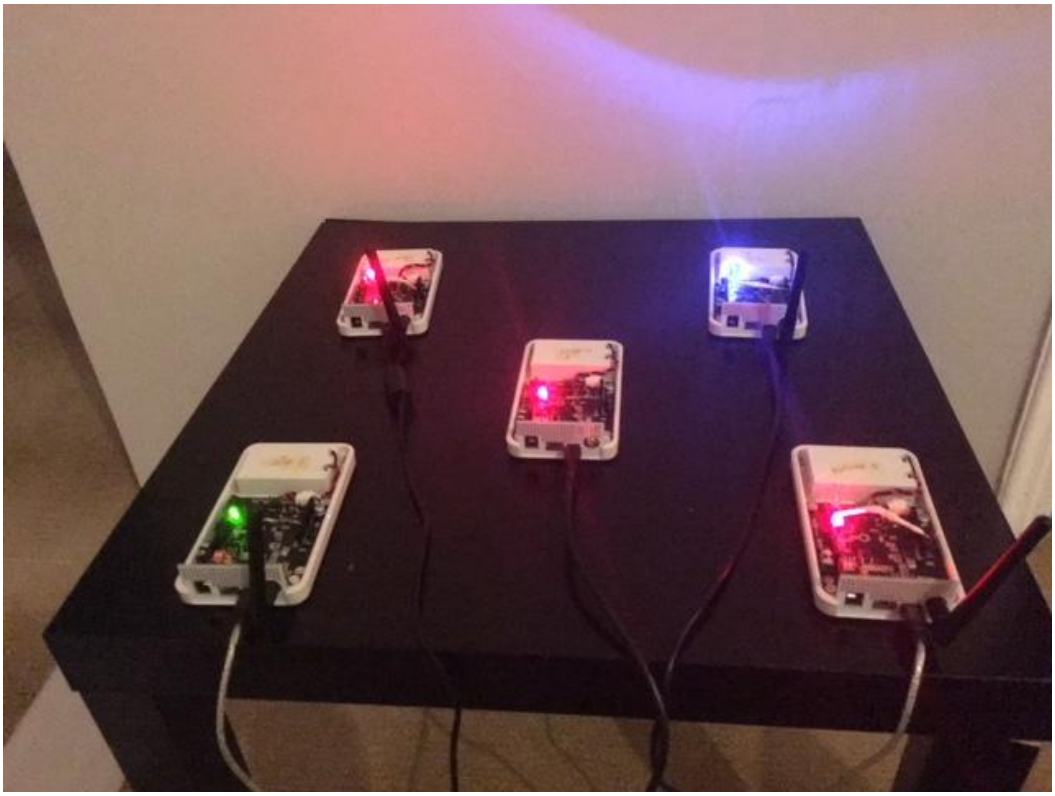
Ακολουθεί η επίδειξη της εντολής trial. Εδώ θα γίνει ταυτόχρονη αποστολή μηνυμάτων από κάθε κόμβο σε κάθε κόμβο και επίσης θα γίνει ενεργοποίηση συγκεκριμένων λοβών ακτινοβολίας. Κατά την αποστολή των μηνυμάτων. Η συγκεκριμένη διαδικασία φαίνεται στην εικόνα 4.21.



Εικόνα 4.21: Χρήση εντολής trial – Πολλαπλή αποστολή μηνυμάτων και ενεργοποίηση λοβών ακτινοβολίας.

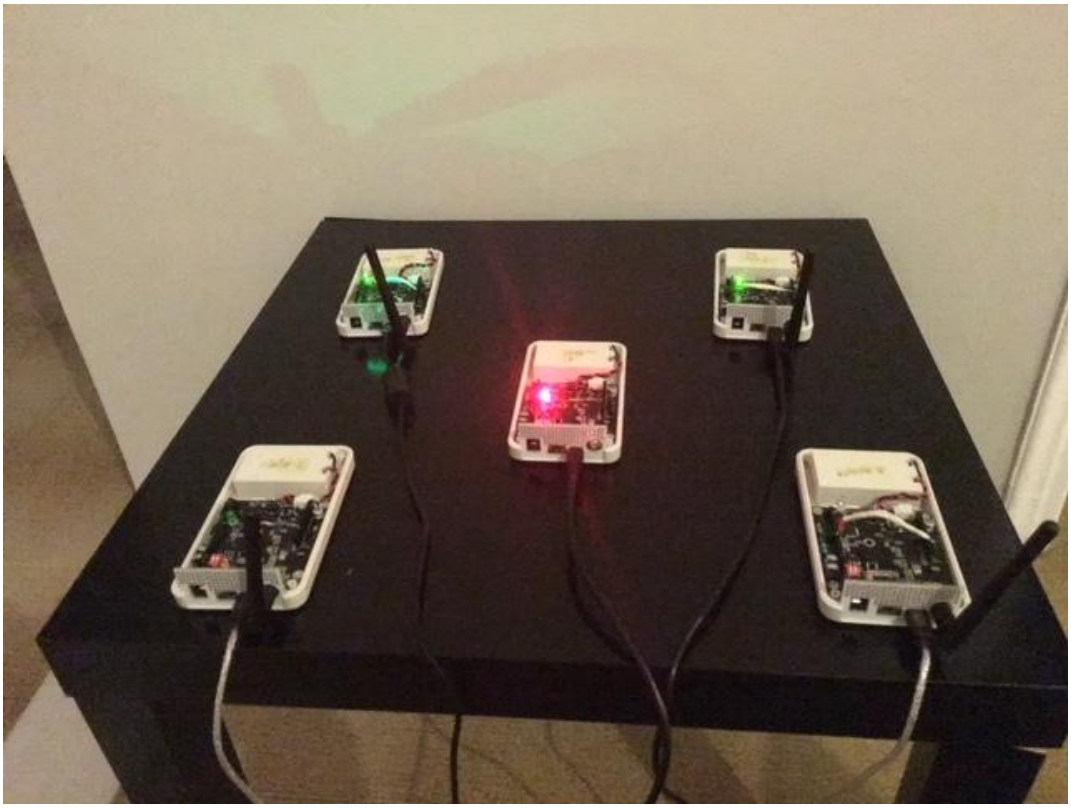
Όπως προαναφέρθηκε και στο δίκτυο 3 κόμβων, στις κονσόλες καταγραφής φαίνεται κάθε μήνυμα που διοχετεύεται στο δίκτυο με πληροφορίες όπως ο τελικός κόμβος, η καθυστέρηση που παίρνει κάθε μήνυμα από το επόμενο και ο λοβός ακτινοβολίας που ενεργοποιείται. Όπως

αναμενόταν, υπάρχει απώλεια μηνυμάτων που οφείλεται στη υπερφόρτωση του δικτύου το οποίο και συναντά στο παράδειγμα αυτό τα όρια του. Ακολουθεί απεικόνιση των κόμβων με ενεργούς τους λοβούς ακτινοβολίας κατά την αποστολή μηνυμάτων στην Εικόνα 4.22.



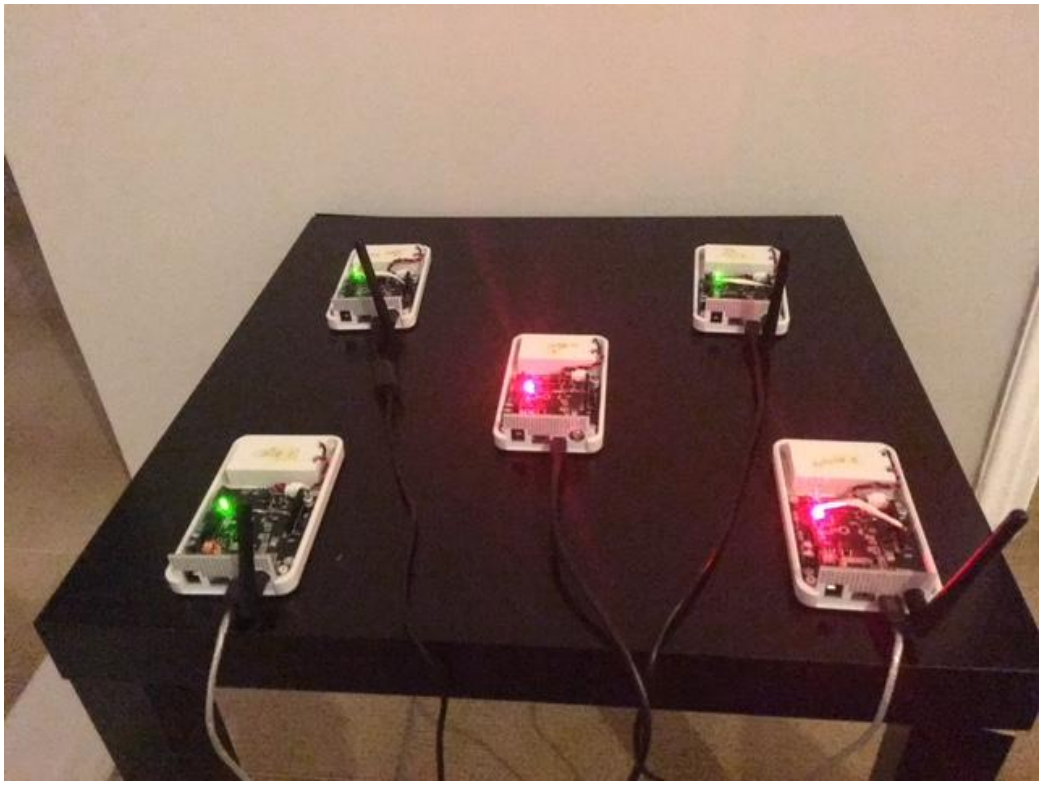
Εικόνα 4.22: Κόμβοι με ενεργούς λοβούς ακτινοβολίας.

Στην Εικόνα 4.22 όλοι οι κόμβοι εκπέμπουν μηνύματα προς όλες τις κατευθύνσεις. (πολλαπλή αποστολή δεδομένων με τυχαίους τελικούς παραλήπτες) Ο κάθε κόμβος ενεργοποιεί τον λοβό ακτινοβολίας που προβλέπεται από τον κατάλληλο πίνακα (πίνακας ενεργοποίησης λοβών ακτινοβολίας) προκειμένου να εκπέμψει μήνυμα.



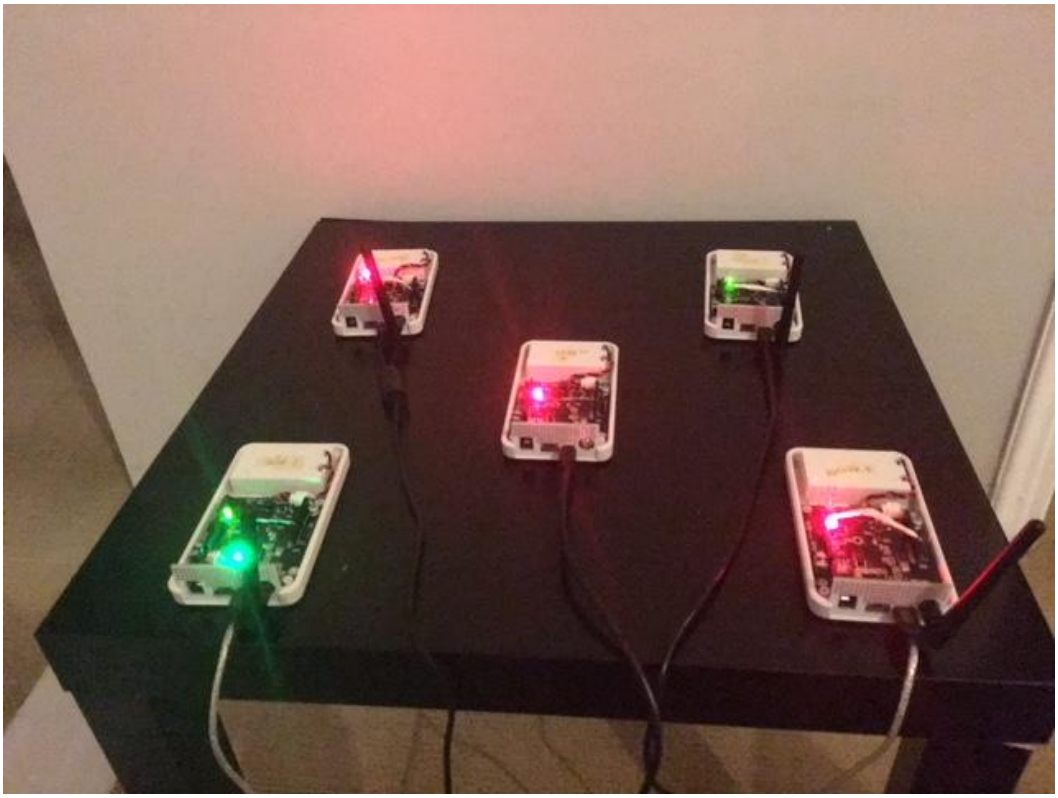
Εικόνα 4.23: Κόμβοι που εκπέμπουν από συγκεκριμένους λοβούς. Ενεργοί κόμβοι 1,4 και 5

Αν και όλοι οι κόμβοι λαμβάνουν μηνύματα, στην παρούσα φάση, οι κόμβοι 2 και 3 δεν εκπέμπουν.



Εικόνα 4.24: Ολοι οι κόμβοι εκπέμπουν, από συγκεκριμένο λοβό ο καθένας

Παρατηρώντας την Εικόνα 4.22 γίνεται φανερό πως οι κόμβοι 1 και 4 εκπέμπουν από διαφορετικό λοβό ακτινοβολίας. Αυτό συμβαίνει επειδή στέλνουν μήνυμα σε διαφορετικό αποδέκτη και χρησιμοποιούν διαφορετικό λοβό. Σε αυτό το σημείο βρίσκεται και η ουσία ενός WSN με έξυπνες κεραίες. Αυτή η δυνατότητα και μόνο, να μπορεί δηλαδή κάθε κόμβος να εκπέμψει από διαφορετικό λοβό ακτινοβολίας που εξαρτάται καθαρά από την θέση του επόμενου κόμβου, προσφέρει μοναδικά οφέλη στο δίκτυο, όπως αυτά έχουν προαναφερθεί στο κεφάλαιο Α. Ένα από τα κυριότερα και πιο σημαντικά οφέλη είναι η εξοικονόμηση ενέργειας – στοιχείο ζωτικής σημασίας σε ένα δίκτυο ασύρματων αισθητήρων.



Εικόνα 4.25: Κόμβοι απαντες σε κατάσταση αποστολή μηνύματος. Να σημειωθεί πως σε σχέση με την εικόνα 4.24, ο κόμβος 1 χρησιμοποιεί διαφορετικό λοβό για να εκπέμψει.

ΚΕΦΑΛΑΙΟ 5:

Συμπεράσματα – Προτάσεις

Είναι γεγονός πως τα δίκτυα ασύρματων αισθητήρων που είναι εξοπλισμένα με έξυπνες κεραίες επιδεικνύουν βελτιωμένα χαρακτηριστικά ακόμη και στην περίπτωση που αυτές οι κεραίες είναι εγκατεστημένες μόνο σε ένα ποσοστό των κόμβων και όχι στο συνολο αυτών (μοντέλο υβριδικού δικτύου). Ήταν εντυπωσιακό το γεγονός που επιβεβαιώθηκε πειραματικά, ότι η απόδοση των δικτύων WSN των εξοπλισμένων με έξυπνες κεραίες διπλασιάζεται σε σχέση με την απόδοση των δικτύων WSN με τις πολυκατευθυντικές κεραίες, ενώ η αύξηση του αριθμού των λοβών οδηγεί σε ακόμη υψηλότερες επιδόσεις. Επιπλέον, διαπιστώθηκε ότι, γενικά, η απόδοση του δικτύου είναι ανεξάρτητη από το μέγεθος («διαστάσεις») του δικτύου, γεγονός που εγγυάται την επεκτασιμότητά του.

Η προτεινόμενη προσέγγιση αποκαλύπτει τη σημασία της ενσωμάτωσης έξυπνων κεραιών σε ασύρματα δίκτυα αισθητήρων: μπορεί να εξασφαλίσει τα επιθυμητά αποτελέσματα χωρίς τροποποίηση των κύριων χαρακτηριστικών ενός WSN δικτύου (αυτο-οργάνωση, περιορισμένη περιοχή μετάδοσης, κλπ). Έχει αποδειχθεί ότι οι έξυπνες κεραίες μπορούν να σχεδιαστούν έτσι ώστε να ταιριάζουν σε ένα ευρύτερο φάσμα εφαρμογών, εξασφαλίζοντας μεγαλύτερη απόδοση και βελτιωμένη ποιότητα σχεδόν χωρίς κόστος. Θεωρείται ότι αυτή η εναλλακτική λύση θα μπορούσε να ανοίξει νέες κατευθύνσεις στην έρευνα, παρέχοντας κίνητρα για καινοτόμες ιδέες καθώς και για περαιτέρω βελτιώσεις και αλλαγές σε υπάρχοντα έργα.

Επίσης, αξίζει να σημειωθεί πως οι πληροφορίες τοποθεσίας σχετικά με έναν ασύρματο αισθητήρα είναι κρίσιμης σημασίας για την επιτυχή συγκέντρωση δεδομένων και φυσικά την εξαγωγή συμπερασμάτων σχετικά με το περιβάλλον. Επιπλέον, η ενσωμάτωση και ενεργοποίηση της δυνατότητας *εντοπισμού* ανθρώπων ή/και άψυχων αντικειμένων από το δίκτυο ασύρματων αισθητήρων θεωρείται ότι είναι ένας οικονομικός τρόπος προσθήκης αξίας σε ένα τέτοιο δίκτυο. Οι πληροφορίες τοποθεσίας μπορούν να επεκταθούν και σε άλλους τομείς εφαρμογών, όπως προστασία από απάτες, διαχείριση στόλου, πρόσβαση δικτύου, παρακολούθηση προσώπων / περιουσιακών στοιχείων, πλοήγηση κλπ.

Διάφορες τεχνικές εντοπισμού έχουν μελετηθεί. Το Παγκόσμιο Σύστημα Εντοπισμού Θέσης (GPS) παρέχει παγκόσμια κάλυψη, ωστόσο απαιτείται ενεργειακά αποδοτικός και εξειδικευμένος εξοπλισμός. Επιπλέον, το GPS δεν είναι σε θέση να λειτουργεί σε εσωτερικούς χώρους, λόγω της μεγάλης εξασθένησης που εισάγεται από τους τοίχους και τις οροφές των κτιρίων. Συνεπώς δεν αποτελεί μια γενική μέθοδο εντοπισμού. Οι τεχνικές ραδιοσυχνότητας έχουν προταθεί ως εναλλακτικό μέσο εντοπισμού ασύρματων μονάδων. Τα συστήματα εντοπισμού θέσης ραδιοσυχνοτήτων (GPRS) ταξινομούνται σε δύο ευρείες κατηγορίες, [10]:

- Συστήματα Εύρεσης Κατεύθυνσης (Direction Finding, DF) και
- Συστήματα Βασισμένα στην Απόσταση (Range Based, RB).

Τα συστήματα DF χρησιμοποιούν σειρές κεραιών και τεχνικές εκτίμησης κατεύθυνσης (Direction of Arrival, DoA) για τον εντοπισμό του κινητού σταθμού (Mobile Station, MS) και χρησιμοποιούνται κυρίως σε περιοχές με περιορισμένη τυχαιότητα. Από την άλλη πλευρά, τα συστήματα RB μετρούν την απόσταση μεταξύ του MS και ενός αριθμού Σταθμών Βάσης (Base Station, BS), και στη συνέχεια η θέση της MS προκύπτει ως το σημείο τομής των αντίστοιχων καμπυλών. Το εύρος της MS υπολογίζεται με τη χρήση είτε μετρήσεων των χρόνων αφίξεων (Time-of-Arrival, ToA) είτε μετρήσεων αντοχής σήματος λήψης (Received Signal Strength, RSS).

Οι μετρήσεις DoA και ToA δεν είναι διαθέσιμες στα οικονομικότερα συστήματα εντοπισμού, [11], λόγω της ανάγκης για συστοιχία κεραιών ή για συγχρονισμό, αντίστοιχα. Αντίθετα, ο εξοπλισμός με δυνατότητα ένδειξης RSS είναι ευρέως διαθέσιμος και παρέχει ένα οικονομικό μέσο για τον εντοπισμό της θέσης. Η ποιότητα εκτίμησης της απόστασης βάσει RSS μπορεί να υποβαθμιστεί σοβαρά λόγω «σκίασης» και πολλαπλής διαδρομής, αλλά η εξασθένιση μικρής κλίμακας μπορεί να εξομαλυνθεί με τον μέσο όρο του χρόνου ή με την αντίστοιχη ζώνη συχνοτήτων, [12].

Είναι περισσότερο από προφανές πως με την χρήση της τεχνολογίας έξυπνων κεραιών και τη παρεχόμενη πληροφορία RSS από οποιονδήποτε κόμβο, ο εντοπισμός κάθε κόμβου στο δίκτυο γίνεται ακόμα πιο εύκολος. Η ακρίβεια βελτιώνεται σημαντικά και τα αποτελέσματα διακίνησης πληροφορίας μέσα σε ένα δίκτυο είναι ακόμα πιο στοχευμένα, δίνοντας στον χρήστη περισσότερες σημαντικές πληροφορίες. Επιπροσθέτως, με την δυνατότητα των έξυπνων κεραιών καθώς και τον εντοπισμό κόμβων μέσω RSS, η ανάγκη υλοποίησης και χρήσης

αλγορίθμων που να λαμβάνουν υπόψιν την τοποθεσία κάθε κόμβου και να δρομολογούν τις όποιες πληροφορίες στο δίκτυο (location aware routing), αυξάνεται σημαντικά.

ΒΙΒΛΙΟΓΡΑΦΙΑ – ΔΙΑΔΙΚΤΥΑΚΕΣ ΠΗΓΕΣ

[1] Akyildiz I., Su W., Sankarasubramaniam Y., Cayirci E., *A survey on sensor networks*. IEEE Commun. Mag. 2002; 40:102–114.

[2] Tubaishat M., Madria S., *Sensor networks: an overview*. IEEE Potentials. 2003; 22:20–30.

[3] Sohrabi K., Gao J., Ailawadhi V., Pottie G., *Protocols for self-organization of a wireless sensor network*. IEEE Personal Commun. 2000; 7:16–27.

[4] Culler D., Estrin D., Srivastava M., *Overview of sensor networks*. IEEE Comput. 2004; 37:41–49.

[5] Verdone R., Dardari D., Mazzini G., Conti A., *Wireless Sensor and Actuator Networks*. London, UK: Elsevier, 2008.

[6] E. Skiani, S. Mitilineos & S. Thomopoulos, *Study of the Performance of Wireless Sensor Networks Operating with Smart Antennas*. IEEE Antennas and Propagation Magazine. 2012; 54(3): 50-67.

[7] *Freakduino Chibi v2.1a Datasheet* [On-line source]. Available: https://freaklabsstore.com/pub/FREAKDUINO-CHIBI%20v2.1A-1%20Datasheet.pdf. [Access: 5-12-2016].

[8] *Freaklabs Chibi Arduino headers* [On-line source]. Available: https://github.com/freaklabs/chibiArduino/blob/master/src/chb_drvr.h. [Access: 15-01-2017].

[9] *Freaklabs Freakduino board* [On-line source]. Available: https://freaklabs.org/chibiarduino.html. [Access 09-12-2016].

[10] Liberti J. & Rappaport T.S., *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Application*. New Jersey: Prentice Hall PTR, 1999.

[11] Caffery J.L. & Stüber G.L., Overview of radiolocation in CDMA cellular systems. IEEE Communications Magazine. 1998; 36(4):38–45.

[12] Liu B.C., Lin K.H. & Wu J.C., *Analysis of hyperbolic and circular positioning algorithms using stationary signal-strength difference measurements in wireless communications*. IEEE Trans. on Vehicular Technology. 2006; 55(2):499–509.