ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ: Μηχανικών Πληροφορικής (Τ.Ε.)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΙΚΤΥΟ ΑΙΣΘΗΤΗΡΩΝ ΓΙΑ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΑΤΜΟΣΦΑΙΡΙΚΩΝ ΣΥΝΘΗΚΩΝ

Ράλλης Σπυρίδων

Επιβλέπων Καθηγητής:

Δρ. Παναγιωτάκης Σπυρίδων

2021

HELLENIC MEDITERRANEAN UNIVERSITY

SCHOOL OF ENGINEERING

Department Of Electrical & Computer Engineering

ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ: Computer Engineering (Τ.Ε.)

THESIS

SENSOR NETWORK FOR MONITORING ENVIRONMENTAL CONDITIONS

Rallis Spyridon

Advising Professor:

Dr. Panagiotakis Spyridon

2021

Ευχαριστίες

Μέσα από τις γραμμές αυτού του κειμένου θα ήθελα να εκφράσω τις θερμές και ειλικρινείς ευχαριστίες μου στους ανθρώπους που συνέβαλαν στην περάτωση της παρούσας διπλωματικής εργασίας.

Αρχικά, θα ήθελα να ευχαριστήσω και να αναγνωρίσω τον καθοριστικό ρόλο που διαδραμάτισε ο Επιβλέπων Καθηγητής, Δρ. Παναγιωτάκης Σπυρίδων, εισηγητής της διπλωματικής μου εργασίας, ο οποίος με την αμέριστη εμπιστοσύνη και υποστήριξη, την αγόγγυστη υπομονή αλλά και την ουσιώδη καθοδήγηση που μου προσέφερε σε όλα αυτά τα χρόνια, συνέτεινε τα μέγιστα στην μέχρι τώρα ακαδημαϊκή μου πορεία.

Εν συνεχεία, οφείλω να ευχαριστήσω το σύνολο του εκπαιδευτικού προσωπικού του ιδρύματος για την πλειάδα των γνωστικών εφοδίων και προκλήσεων που μου παρείχαν κατά τη διάρκεια των σπουδών μου, ενώ ιδιαίτερη μνεία θα ήθελα να κάνω στον εκλιπόντα καθηγητή, Δρ. Βλησίδη Ανδρέα, για τις εκπαιδευτικές ευκαιρίες που μου προσέφερε στο εργαστήριο του οποίου ήταν ιθύνων.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, για την ηθική αλλά βεβαίως και για την έμπρακτη υποστήριξη που μου παρείχαν σε όλη τη διάρκεια της μέχρι τώρα πορείας μου, στοιχεία που αποτέλεσαν θεμέλιο ανάπτυξης για την ακαδημαϊκή εξέλιξή μου αλλά ταυτόχρονα και πολύτιμο εφόδιο για τη μετέπειτα ζωή μου.

Abstract

The aim of this thesis is the design and deployment of a wireless sensor network targeted at measuring and presenting environmental and air-quality data, like temperature or particulate matter. Taking advantage of an implementation consisting of a long-range (LoRa) backbone and a multitude of sensor nodes connected in a mesh topology, the system will be able to serve varying geomorphological formations.

Σύνοψη

Στόχος της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και ανάπτυξη ενός ασύρματου δικτύου αισθητήρων που έχει σαν στόχο τη μέτρηση και παρουσίαση περιβαλλοντικών συνθηκών καθώς και συνθηκών ποιότητας αέρα, όπως η θερμοκρασία και τα αιωρούμενα σωματίδια. Αξιοποιώντας μία υλοποίηση που αποτελείται από ένα δίκτυο κορμού μεγάλης εμβέλειας (LoRa) καθώς και μία πληθώρα αισθητήριων κόμβων συνδεδεμένων σε τοπολογία πλέγματος, το σύστημα θα είναι ικανό να εξυπηρετήσει ποικίλους γεωμορφολογικούς σχηματισμούς.

Περιεχόμενα

1.1 **Κίνητρα για την Παρούσα Εργασία**

Η ολοένα και αυξανόμενη πυκνότητα κατοίκησης στις μεγαλουπόλεις, οι ταχεία εξελισσόμενοι ρυθμοί ζωής, αλλά και οι εκτεταμένες ανθρώπινες δραστηριότητες και επεμβάσεις στη φύση, έχουν αλλάξει δραστικά τις ανάγκες αλλά και τα χαρακτηριστικά τόσο των σύγχρονων κοινωνιών, όσο και των οικοσυστημάτων. Νέοι –ελάσσονος μέχρι πρότινος σημασίας- κίνδυνοι, όπως η αυξανόμενη επιβάρυνση της ατμόσφαιρας με μολυσματικούς παράγοντες (όπως λ.χ. τα αιωρούμενα σωματίδια), η αύξηση των ποσοστών του διοξειδίου του άνθρακα στον ατμοσφαιρικό αέρα, αποτελούν στοιχεία που επιβαρύνουν τόσο την ανθρώπινη υγεία, αλλά αποτελούν και δυνητικά καταστρεπτικές για τον πλανήτη δυνάμεις.

Τα ασύρματα δίκτυα αισθητήρων (wireless sensor networks – WSNs), –τομέας εδραιωμένος μεν, αλλά ταχύτατα αναπτυσσόμενος στον κλάδο των ασύρματων επικοινωνιών- αποτελούν πολύτιμο εφόδιο για την παρακολούθηση, καταγραφή και μελέτη τόσο του κλίματος, όσο και των άμεσα μεταβαλλόμενων περιβαλλοντικών συνθηκών, επιτρέποντας την έγκαιρη λήψη αποφάσεων για διορθωτικές επεμβάσεις, ή ακόμα και την πρόληψη δυσμενών καταστάσεων.

1.2 **Στόχος και Περιγραφή της Εργασίας**

Στόχος μέσα από την παρούσα διπλωματική εργασία είναι να σχεδιαστεί και να κατασκευαστεί ένα ασύρματο δίκτυο αισθητήρων μέτρησης συνθηκών ποιότητας αέρα το οποίο θα συλλέγει και θα αποστέλλει δεδομένα σε μία διαδικτυακή υποδομή για την αποθήκευση, πιθανώς περεταίρω επεξεργασία αλλά και παρουσίασή τους.

Η πλατφόρμα θα βασιστεί στην ανάπτυξη φορητών ασύρματων κόμβων σε τοπολογία πλέγματος (mesh) στην ελεύθερη μπάντα των 433 MHz με διαμόρφωση FSK με κάνοντας χρήση πομποδεκτών της οικογένειας “RFM69”. Το δεδομένα από το δίκτυο πλέγματος θα συγκεντρώνονται σε μία πύλη (gateway– coordinator), από την οποία και θα γίνεται η αποστολή τους στον εξυπηρετητή διαμέσου μίας ασύρματης ζεύξης μεγάλης εμβέλειας βασιζόμενης σε διαμόρφωση LoRa.

1.3 **Δομή της Εργασίας**

Τα πρώτα τρία κεφάλαια της εργασίας αποτελούν εισαγωγή, παρέχοντας τόσο την περιγραφή (απλουστευμένη, αλλά και αναλυτική), όσο και το απαραίτητο τεχνολογικό υπόβαθρο που απαιτήθηκε για τη συγγραφή, αλλά και για την υλοποίηση της εργασίας.

Στο τέταρτο, πέμπτο και έκτο κατά σειρά κεφάλαιο, γίνεται ανάλυση του ηλεκτρονικού εξοπλισμού που χρησιμοποιήθηκε, του απαιτηθέντος λογισμικού, τόσο στο κομμάτι των ηλεκτρονικών (ενσωματωμένο λογισμικό), όσο και στο κομμάτι της δικτυακής υποδομής.

Το έβδομο κεφάλαιο, εστιάζει στην υλοποίηση του λογισμικού που απαιτήθηκε για τη λειτουργία του δικτύου, ενώ αναλύονται οι αλγόριθμοι που χρησιμοποιούνται σε κάθε υπομονάδα του συστήματος.

Τέλος, το όγδοο και τελευταίο κεφάλαιο, καταλήγει σε συμπεράσματα τα οποία παράχθηκαν από την εκπόνηση της εργασίας, ενώ προτείνει και πιθανές επεκτάσεις του δικτύου.

2.1 Περί Δικτύων Mesh

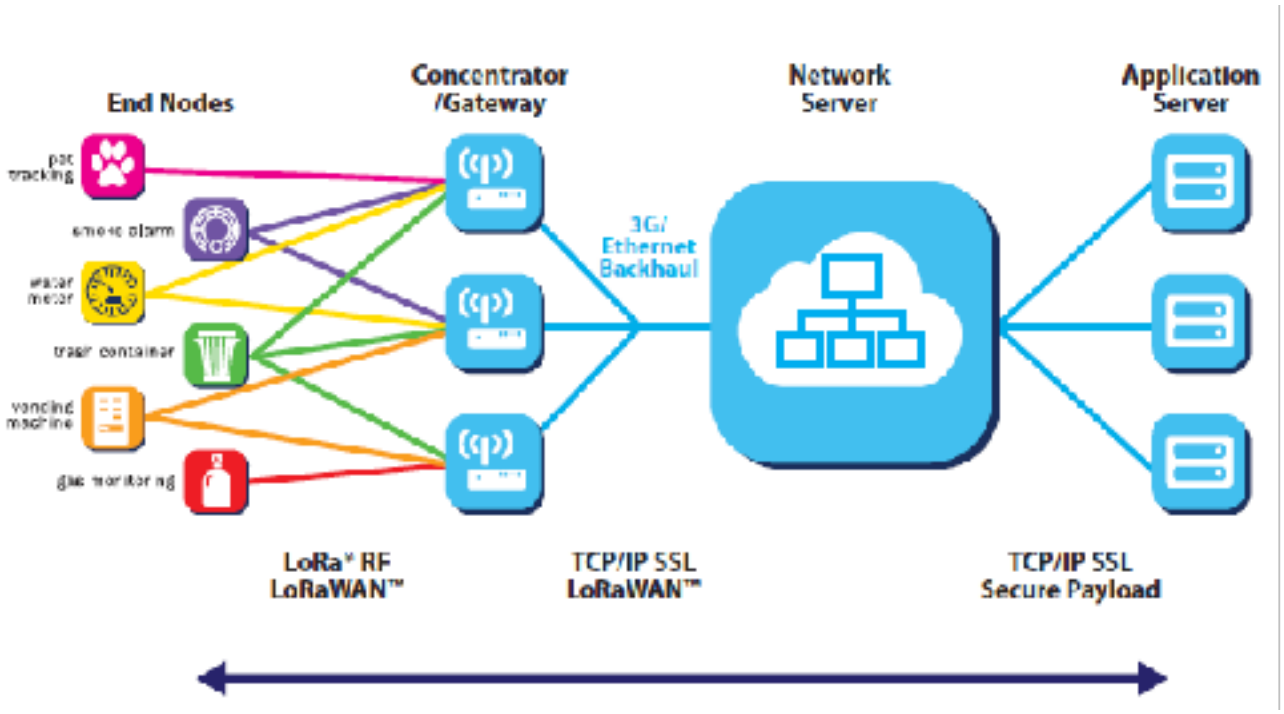
Σε ένα ασύρματο δίκτυο σε τοπολογία πλέγματος ένας κόμβος είναι –τυπικά- συνδεδεμένος με τους γειτονικούς του με δύο ή περισσότερες διαδρομές, ενώ συνηθέστερα, οι διαδρομές αυτές σχηματίζουν βρόγχους (και γι' αυτό το λόγο, είθισται να ονομάζονται και δίκτυα βρόγχου)[1]. Οι πολλαπλές διαδρομές για την δρομολόγηση των μηνυμάτων, προσφέρουν τη δυνατότητα εναλλακτικής δρομολόγησης σε περίπτωση αποτυχίας μίας σύνδεσης.

2.2 Περί LoRa

Αναπτυχθέν από τη Cycleo στη Γρενόβλη (Γαλλία), το LoRa αποτελεί μία τεχνική διαμόρφωσης ραδιοσυχνοτήτων που βασίζεται στη μέθοδο διασποράς φάσματος chirp spread spectrum (CSS)[2]. Πρόκειται για τεχνολογία εμπορικού χαρακτήρα, ενώ αργότερα εξαγοράστηκε από την εταιρεία Semtech (Καλιφόρνια, ΗΠΑ).

Ένα δίκτυο αρχιτεκτονικής LoRa, τυπικά αναπτύσσεται σε τοπολογία "αστέρα-αστέρων" (star of stars), όπου οι κόμβοι (nodes), οι πύλες (gateways), αλλά και οι εξυπηρετητές δικτύου και εφαρμογής επικοινωνούν μεταξύ τους όπως στο (Σχ. 1)[3].

Σχ.1

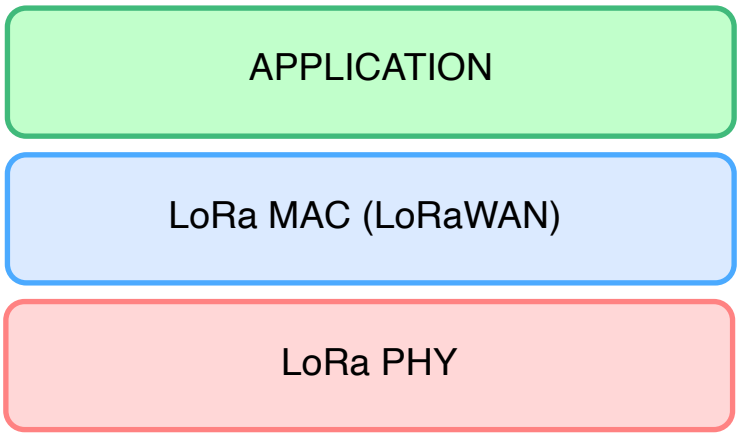


τυπική αρχιτεκτονική ενός δικτύου βασισμένου σε LoRa

Ένας κόμβος συλλέγει τιμές αισθητήρων, ενώ τις αποστέλλει στη δικτυακή πύλη χρησιμοποιώντας μία single-hop ασύρματη ζεύξη που κάνει χρήση της διαμόρφωσης LoRa. Η πύλη επικοινωνεί με έναν κεντρικό εξυπηρετητή μέσω τυπικών ζεύξεων IP και αποτελεί ουσιαστικά "γέφυρα" ανάμεσα στον κόμβο και τον εξυπηρετητή, διερμηνεύοντας τα ραδιοκύματα σε πακέτα IP, και το ανάποδο.

Με τον όρο LoRa είθισται να γίνεται αναφορά τόσο στο κατώτερο, φυσικό επίπεδο (LoRa PHY), όσο και στο επίπεδο MAC, το οποίο ονομάζεται LoRaWAN. Στο (Σχ. 2) παρουσιάζεται η προαναφερθείσα διαστρωμάτωση, ενώ στο εξής, θα γίνεται αναφορά και στα δύο επίπεδα (PHY και MAC) με τον όρο "LoRa" και όπου απαιτείται θα γίνεται διάκριση ανάμεσά τους.

Σχ. 2



διαστρωμάτωση των επιπέδων του LoRa

2.2.1 **LoRa PHY**

Το LoRa κάνει χρήση ραδιοσυχνοτήτων στις ελεύθερες μπάντες των 433 και 863–870 MHz (Ευρώπη), των 915–928 MHz (Αυστραλία), των 902–928 MHz (Βόρεια Αμερική), των 865–867 MHz (Ινδία) αλλά και των 2.4 GHz, σύμφωνα με τους κανονισμούς της κάθε γεωγραφικής περιοχής[4]. Βασίζεται στη μέθοδο διασποράς φάσματος "chirp spread spectrum" (CSS), βάση της οποίας ένα chirp αναπαριστά ένα ημιτονοειδές σήμα της αύξησης ή μείωσης συχνότητας κατά το πέρασμα του χρόνου[5].

Το LoRa έχει αξιοσημείωτη ανοχή στον εξωτερικό θόρυβο, μιας και καταλαμβάνει το σύνολο του διαθέσιμου εύρους ζώνης (bandwidth) του καναλιού εκπομπής[6]. Επιπλέον,

χάρη στη συχνοτική μεταπήδηση που πραγματοποιείται σε κάθε μετάδοση, η ανοχή στο θόρυβο αυξάνεται– ακόμα περισσότερο[7].

Με δεδομένη ισχύ, το LoRa επιτυχγάνει μεγαλύτερη εμβέλεια εν συγκρίσει με λοιπά τηλεπικοινωνιακά συστήματα που λειτουργούν με μεγαλύτερες συχνότητες φορέα. Ταυτόχρονα όμως, η χρήση χαμηλών συχνοτήτων (< 1 GHz) οδηγεί σε μικρότερη ταχύτητα μεταφοράς δεδομένων. Ο χαμηλότερος ρυθμός μετάδοσης δεδομένων πάντως μειώνει τις απαιτήσεις ευαισθησίας στον δέκτη, μιας και παρέχει πλεονεκτήματα στον ρυθμό σφαλμάτων (error rate), ενώ επιπλέον, η χαμηλή συχνότητα αυξάνει και την πιθανότητα μετάδοσης του σήματος γύρω από εμπόδια (φαινόμενο περίθλασης) ακόμα και όταν η οπτική επαφή δεν είναι εγγυημένη[8].

Το LoRa αξιοποιεί 6 παράγοντες διασποράς (spreading factors – από SF 7 έως και SF 12), αλλά και 3 εύρη ζώνης (στα 125, 250 ή 500 KHz)[9]. Χάρη στην τεχνική διασποράς φάσματος, μηνύματα με διαφορετικούς ρυθμούς μετάδοσης μπορούν να θεωρηθούν ως διαφορετικά “κανάλια”. Έτσι, η χωρητικότητα της πύλης δύναται να αυξηθεί.

Για την ανάλυση της απόδοσης της διαμόρφωσης LoRa, είναι σημαντικό να εξεταστούν οι παράγοντες που επηρεάζουν την ευαισθησία του δέκτη. Η Εξ. 1 αναπαριστά την ευαισθησία του δέκτη στους 25°C[10].

Εξ. 1

$$S = -174 + 10\log_{10} BW + NF + SNR$$

Στην Εξ.1, ο πρώτος όρος αναπαριστά το θερμικό θόρυβο σε 1 Hz συχνότητας, ενώ εξαρτάται από την θερμοκρασία του δέκτη. Οι όροι BW και NF, αναπαριστούν το εύρος ζώνης στα του δέκτη, αλλά και το θόρυβο στον δέκτη, αντίστοιχα. Τέλος, το SNR αναπαριστά τον ελάχιστο λόγο σήματος προς θόρυβο ώστε το σήμα να αποδιαμορφωθεί με επιτυχία. Το εύρος ζώνης στα και ο λόγος σήματος προς θόρυβο είναι οι κατά κύριο λόγος παράγοντες που επηρεάζουν την απόδοση του LoRa[11].

Όσο μειώνεται το εύρος ζώνης, αυξάνεται η ευαισθησία του δέκτη, πράγμα που σημαίνει ότι τα δεδομένα μπορούν να αποσταλούν μακρύτερα με την ίδια ισχύ. Με άλλα λόγια, η απαιτούμενη ισχύ εκπομπής μειώνεται για μετάδοση στην ίδια (χιλιομετρική) απόσταση. Παρόλα αυτά, εξαιτίας του ότι το εύρος ζώνης στα είναι ανάλογο του ρυθμού

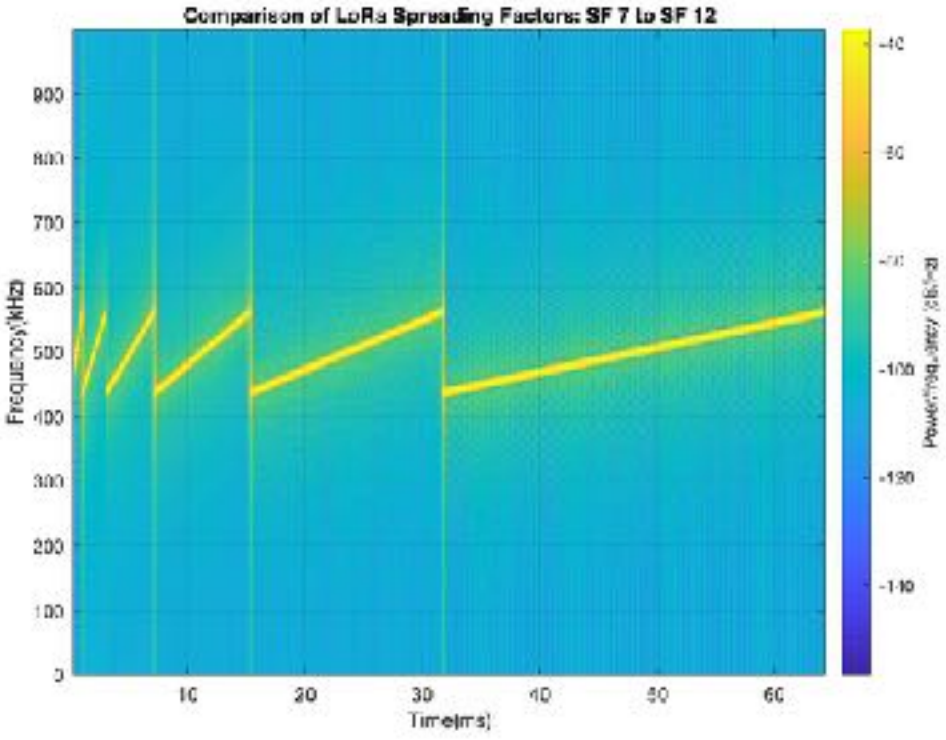
μετάδοσης των chirp, η χρήση μικρότερου εύρους ζώνης μεθερμηνεύεται σε σε αύξηση του χρόνου μετάδοσης. Κατά τρόπο παρόμοιο, μία αύξηση στο λόγο σήματος προς θόρυβο, βελτιώνει την απόδοση του LoRa.

Οι δύο παράμετροι που καθορίζουν το λόγο σήματος προς θόρυβο είναι ο παράγοντας διασποράς (SF), αλλά και το Coding rate (CR). Ο παράγοντας διασποράς είναι παράμετρος που χρησιμοποιείται κατά την διαμόρφωση, ενώ ένα μοναδικό bit διαμορφώνεται σε 2^{SF} chirp(s)[13].

Γενικεύοντας, όσο υψηλότερος είναι ο παράγοντας διασποράς, τόσο χαμηλότερος είναι ο λόγος σήματος - προς - θόρυβο που απαιτείται για την αποδιαμόρφωση. Παρόλα αυτά, ένας υψηλότερος παράγοντας διασποράς (λ.χ. SF 12) απαιτεί περισσότερα chirps για την αποδιαμόρφωση, πράγμα που αυξάνει τον χρόνο μετάδοσης (time-on-air – ToA)[14].

Το Σχ. 3 επιχειρεί να συγκρίνει τα χαρακτηριστικά διαφορετικών spreading factor(s).

Σχ. 3



2.2.2 **LoRaWAN**

Το LoRaWAN είναι ένα πρωτόκολλο που αναπτύσσεται πάνω στο LoRa, κάνοντας χρήση αυτού και αντιμετωπίζοντας τις ιδιαιτερότητές του. Ουσιαστικά, το LoRaWAN υλοποιεί τον έλεγχο προσπέλασης μέσου (medium access control – MAC) για το LoRa και κατά συνέπεια ανήκει στο data link layer (επίπεδο 2 του μοντέλου OSI)[15]. Υποστηρίζεται από το 2015 από το LoRa Alliance, μέλη του οποίου αποτελούν καταξιωμένες επιχειρήσεις, μερικές εκ των οποίων είναι οι IBM, Microchip και Cisco.

Με δεδομένο ότι σε ένα δίκτυο LoRaWAN οι κόμβοι πιθανώς να εξυπηρετούν διαφορετικές ανάγκες (π.χ. κόμβοι που είτε συμπεριλαμβάνουν επενεργητές ή όχι, καθώς και κόμβοι με αισθητήρες με διαφορετικό ρυθμό ανάγνωσης) υπάρχει πρόβλεψη για διαφορετικές κλάσεις συσκευών, με ποικίλα χαρακτηριστικά[16].

Έτσι, οι τερματικές συσκευές – κόμβοι, κατατάσσονται σε μία εκ των κλάσεων A, B, ή C, οι οποίες χαρακτηρίζονται από διαφορετικές ενεργειακές απαιτήσεις, αλλά και διαφορετικά χαρακτηριστικά αποστολής – λήψης. Πρακτικά, μία συσκευή Class-A έχει τη μικρότερη ενεργειακή κατανάλωση, αλλά ταυτόχρονα έχει ανοιχτό παράθυρο λήψης δεδομένων (downlink) μόνο ύστερα από μία επιτυχημένη αποστολή (uplink).

Σε ένα δίκτυο LoRaWAN ένας κόμβος δεν “αντιστοιχίζεται” σε κάποια συγκεκριμένη πύλη, αλλά αντιθέτως τα δεδομένα που αποστέλλει μπορούν να αναληφθούν από πληθώρα πυλών που βρίσκονται σε εγγύτητα[17]. Κάθε μία από τις πύλες θα προωθήσει τα δεδομένα προς τον εξυπηρετητή δικτύου μέσω κάποιας δικτυακής ζεύξης (λ.χ. Ethernet, ή Wi-Fi) ο οποίος και αναλαμβάνει τη διαχείριση του δικτύου (έλεγχοι ασφαλείας, χρονικός προγραμματισμός επιβεβαιώσεων, κ.λπ).

2.2.3 **Εξυπηρετητής Δικτύου και Εφαρμογής**

Το κομμάτι του εξυπηρετητή δικτύου (network server) είναι υπεύθυνο για τη διαχείριση της κυκλοφορίας μέσα στο δίκτυο, υλοποιώντας ουσιαστικά το πρωτόκολλο LoRaWAN. Οι κύριοι στόχοι του είναι η εγγύηση της ασφάλειας, της επεκτασιμότητας και της αξιοπιστίας στη δρομολόγησης των δεδομένων.

Από την άλλη, ο εξυπηρετητής εφαρμογής (application server) διαχειρίζεται το στρώμα εφαρμογής. Αυτό συμπεριλαμβάνει την αποκωδικοποίηση και αποκρυπτογράφηση

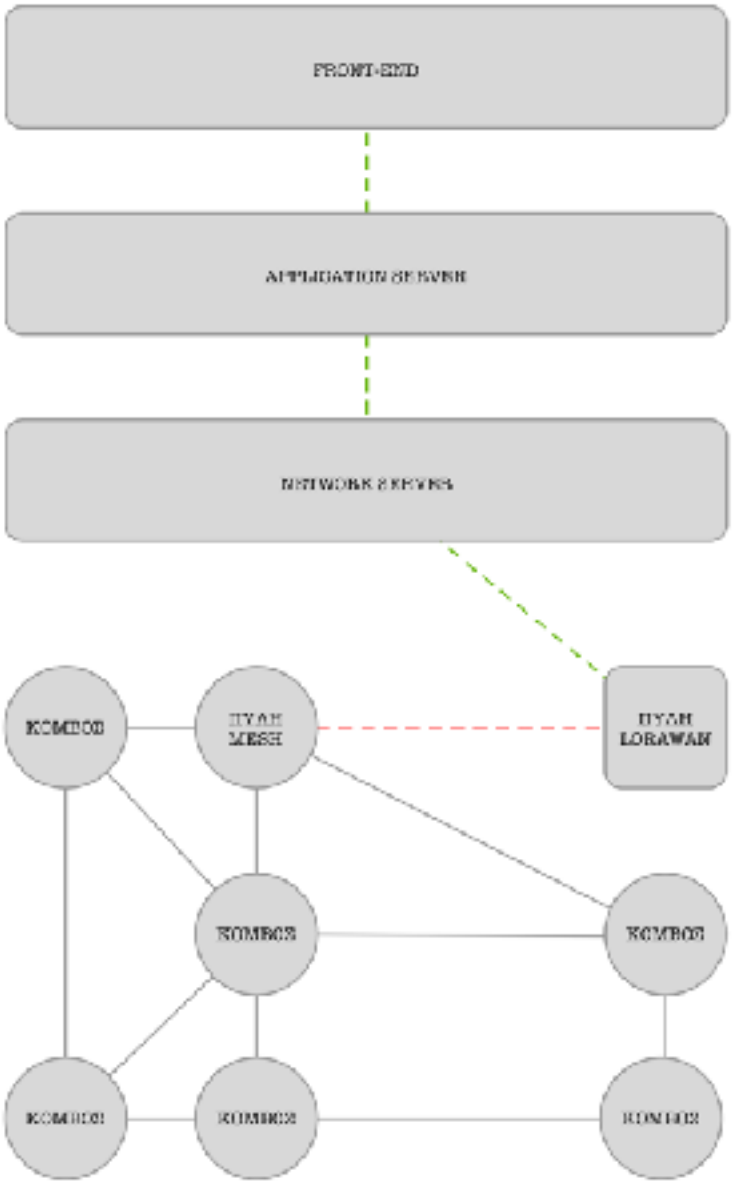
των ανερχόμενων δεδομένων, την κωδικοποίηση και κρυπτογράφηση των κατερχόμενων δεδομένων, αλλά και την διαχείρισης της ουράς.

Τυπικά, ένας εξυπηρετητής εφαρμογής αναλαμβάνει και την ενσωμάτωση τρίτων υπηρεσιών (λ.χ. Amazon Web Services, Microsoft Azure, κ.λπ) είτε για την αποθήκευση των δεδομένων, ή για την προώθηση και παρουσίασή τους.

3.1 **Επισκόπηση**

Το σύστημα αποτελείται από ένα δίκτυο κόμβων που φιλοξενούν αισθητήριες μονάδες σε τοπολογία πλέγματος (mesh), οι οποίοι και αποστέλλουν δεδομένα σε έναν κεντρικό κόμβο (πύλη– gateway) για την μεταφορά τους στο δίκτυο κορμού, και πιο συγκεκριμένα, στον network server. Από εκεί, και ενώ η ζεύξη έχει ήδη μεταβεί σε TCP/IP, τα δεδομένα περνάνε στον εξυπηρετητή εφαρμογής, πριν τη διαχείρισή τους από την εφαρμογή του χρήστη (front-end).

Στο Σχ. 4 παρουσιάζεται η αρχιτεκτονική του συστήματος, καθώς και η διαστρωμάτωσή του:



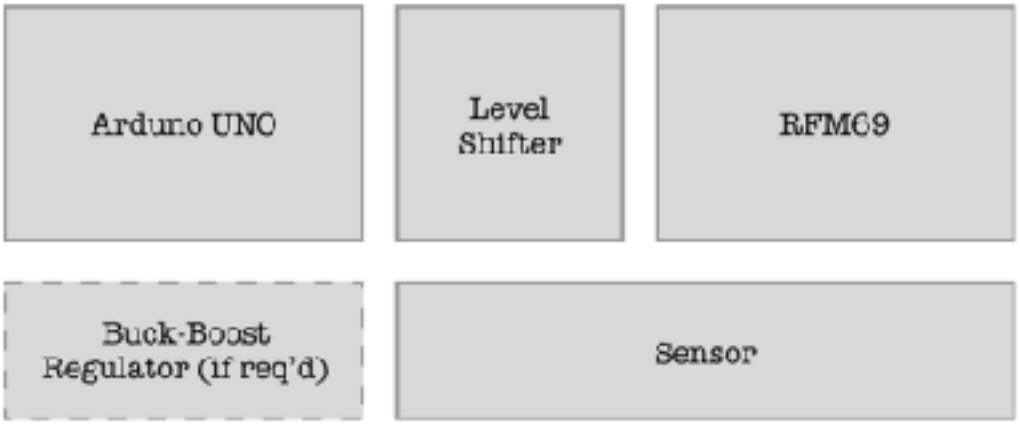
Σχ. 4

3.2 **Υποδομή Υλικού**

Στην παρούσα ενότητα παρουσιάζεται η υλική υποδομή τόσο των κόμβων, όσο και της πύλης. Η πύλη, στην ουσία αποτελεί και αυτή κόμβο του συστήματος, με ειδοποιό διαφορά το ρόλο της, ο οποίος εν αντιθέσει με τους λοιπούς κόμβους, δεν είναι η καταγραφή τιμών από τα αισθητήρια, αλλά η αποστολή και η λήψη μηνυμάτων προς και από το LoRaWAN.

Στο Σχ. 5, παρουσιάζεται η γενική υλοποίηση της πύλης, ενώ στο Σχ. 6 η υλοποίηση των κόμβων:

Σχ. 5



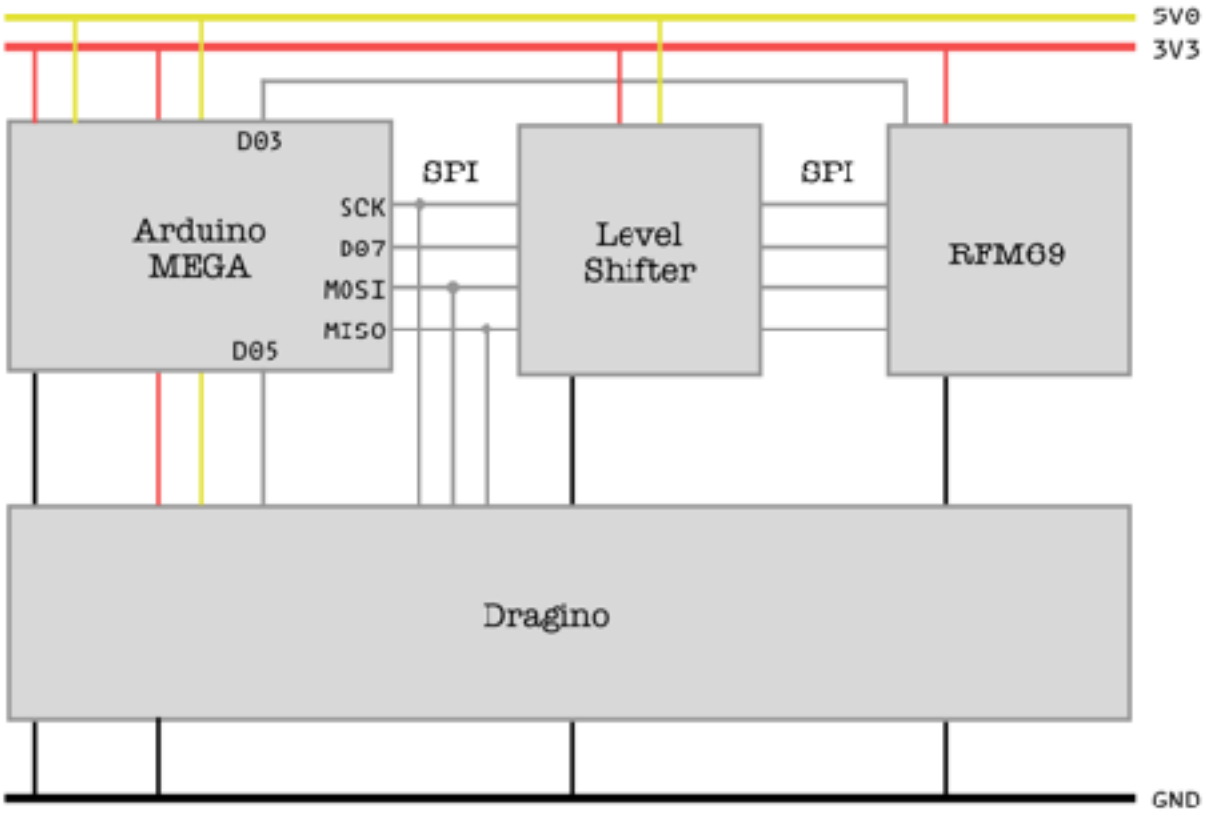
Σχ. 6

3.2.1 **Υποδομή Πύλης**

Η πύλη αποτελείται από μία μονάδα Arduino MEGA, καθώς και τις απαραίτητες υπομονάδες επικοινωνίας RFM69 (mesh) και Dragino (LoRa). Για την επικοινωνία με το RFM69 απαιτείται μονάδα μετάφρασης τάσεων (level shifter) από τα 5,0V στα 3,3V και αντίστροφα, ενώ από την άλλη στο Dragino, η μονάδα αυτή είναι ενσωματωμένη.

Στην πύλη δεν ενσωματώνεται κάποιο αισθητήριο, ενώ αντιθέτως, τοποθετείται η κεραία του Dragino (868 MHz).

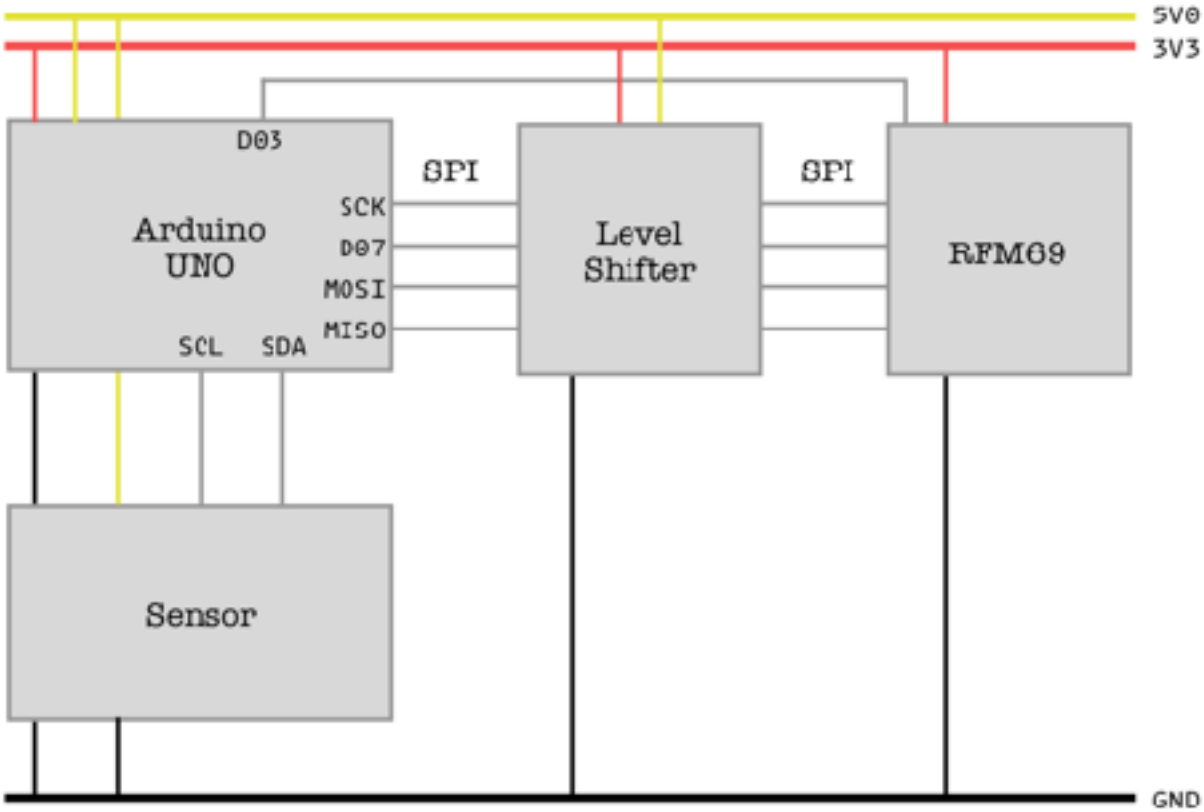
Στο Σχ. 7 αποτυπώνονται οι συνδέσεις μεταξύ των τμημάτων που αποτελούν την πύλη.



Σχ. 7

3.2.2 **Υποδομή Κόμβων**

Οι κόμβοι που φιλοξενούν τα αισθητήρια, αποτελούνται από μία μονάδα Arduino UNO, καθώς και την απαραίτητη υπομονάδα επικοινωνίας RFM69 (mesh). Όπως και στην περίπτωση της πύλης, για την επικοινωνία ανάμεσα στο Arduino (5,0V) και το RFM69 (3,3V) απαιτείται μονάδα μετάφρασης τάσεων (level shifter).



Σχ. 8

Και τα τρία (3) αισθητήρια που επιλέχθηκαν επικοινωνούν με το Arduino UNO μέσω σειριακού πρωτοκόλλου I2C, ενώ τόσο η τάση λειτουργίας, όσο και η «λογική» τους είναι στα 5,0V. Κατ' αυτόν τον τρόπο, ανάμεσα στο Arduino UNO και τα αισθητήρια, δεν απαιτείται level shifter.

Οι αντιστάσεις pull-up που απαιτούνται από το πρωτόκολλο I2C, βρίσκονται –εκ κατασκευής- πάνω στην πλακέτα του Arduino UNO.

3.3 **Υποδομή Λογισμικού**

Η ενότητα αυτή εστιάζει στην υποδομή του λογισμικού του συστήματος, η οποία και αποτελείται από τρία (3) κύρια τμήματα.

3.3.1 **Εξυπηρετητής Δικτύου**

Ο εξυπηρετητής δικτύου που αξιοποιείται στην πλατφόρμα, ονομάζεται The Things Stack, ενώ πρόκειται για εμπορικού χαρακτήρα προϊόν της The Things Industries (TTI). Υπάρχει, επιπροσθέτως, έκδοση προσβάσιμη στην κοινότητα (The Things Stack – "Community Edition") η οποία και αξιοποιείται για το σύστημα υπό κατασκευή.

3.3.2 **Εξυπηρετητής Εφαρμογής**

Ο εξυπηρετητής εφαρμογής που περιλαμβάνεται στο TTS παρέχει τη δυνατότητα διασύνδεσης με εξυπηρετητές MQTT ή HTTP, ενώ ενσωματώνει επιπλέον την υποστήριξη υπηρεσιών όπως οι Amazon Web Services (AWS), Microsoft Azure, αλλά και Google Cloud.

3.3.3 **Διεπαφή Χρήστη**

Το τρίτο κατά σειρά και τελευταίο κομμάτι του συστήματος που αφορά τη διεπαφή του χρήστη με το σύστημα, υλοποιείται με χρήση της πλατφόρμας Node-RED. Αυτή, αποτελεί ένα εύχρηστο εργαλείο οπτικού προγραμματισμού, αρχικώς αναπτυχθέν από την IBM, με στόχο την ενοποίηση υλικών συσκευών, διαδικτυακών υπηρεσιών και API(s) ως μέρος του Διαδικτύου των Αντικειμένων (Internet of Things - IoT).

Το Node-RED παρέχει ένα εργαλείο επεξεργασίας ροών που «φιλοξενείται» μέσα στον φυλλομετρητή (browser) του χρήστη, καθότι κάνοντας χρήση της πλατφόρμας Node.JS, είναι κατασκευασμένο σε JavaScript. Οι ροές αποθηκεύονται σε μορφή JSON, ενώ και αυτό, υποστηρίζει με τη σειρά του πληθώρα εξωτερικών συνδέσεων και υπηρεσιών, όπως MQTT, The Things Stack, κ.λπ.

4.1 **Πύλη LoRaWAN**

Για την επικοινωνία ανάμεσα στον τερματικό κόμβο και τον εξυπηρετητή της εφαρμογής, απαιτείται η ύπαρξη μίας πύλης (gateway) που θα ζεύγνει τα δύο μέρη, δεδομένου ότι ο τερματικός κόμβος διαθέτει –μόνον- διεπαφή LoRa, ενώ ο εξυπηρετητής προϋποθέτει επικοινωνία σε επίπεδο IP.

Το ρόλο αυτό αναλαμβάνει η συσκευή “The Things Indoor Gateway” (συντομογραφικά TTIG) (Εικ. 1), κατασκευασθείσα από την The Things Industries. Βασισμένη στο ολοκληρωμένο SX1308 (Semtech) και με δυνατότητα σύνδεσης με το διαδίκτυο μέσω ασύρματου δικτύου (WiFi), αποτελεί μία από τις πιο δημοφιλείς και χαμηλού κόστους συσκευές της κατηγορίας.

Εικ. 1

4.2 **Αναπτυξιακή Πλατφόρμα Arduino**

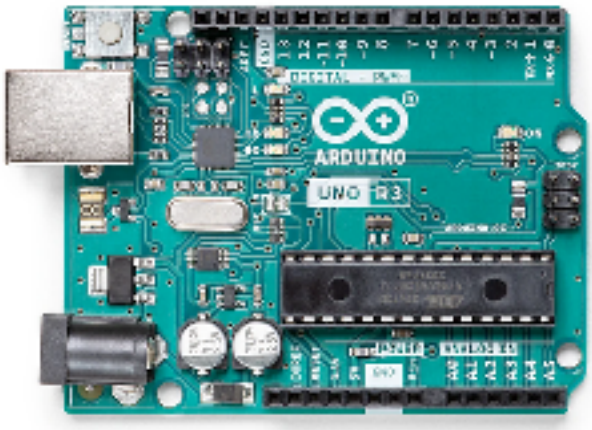
Υπό τον όρο Arduino, στην πραγματικότητα καλύπτεται ένα ευρύ φάσμα συσκευών αλλά και λογισμικού. Πρόκειται για μία ολοκληρωμένη πλατφόρμα σχεδιασμού και υλοποίησης πρωτοτύπων (prototypes) δημιουργηθείσα από τους Massimo Banzi και David Cuartielles [2005], η οποία χάρη στην εξαιρετική ευκολία χρήσης αλλά και το χαμηλό κόστος, αποτελεί την πιο διαδεδομένη πλατφόρμα του είδους.

Εστιάζοντας στο υλικό της πλατφόρμας, βρίσκουμε αναπτυξιακές πλακέτες βασιζόμενες –συνηθέστερα- σε μικροελεγκτές της οικογένειας AVR. Οι προγραμματιζόμενοι αυτοί ελεγκτές παρέχουν πληθώρα περιφερειακών (μνήμες, σειριακά πρωτόκολλα,

αναλογικοί μετατροπείς, κ.λπ) με υψηλό επίπεδο ολοκλήρωσης, διευκολύνοντας έτσι τις υλοποιήσεις πολλαπλών τύπων εφαρμογών.

Επί παραδείγματι, η αναπτυξιακή πλακέτα Arduino UNO (Εικ. 2), χρήση της οποίας γίνεται στους κόμβους του υπό ανάπτυξη συστήματος, αποτελείται από τον μικροελεγκτή ATMEGA328P, καθώς και τα απαραίτητα περιφερειακά για τη λειτουργία (π.χ. κρύσταλλο– ταλαντωτή) αλλά και τον προγραμματισμό του (π.χ. κύκλωμα ISP). Περιλαμβάνονται επιπλέον ορισμένα βοηθητικά, αλλά όχι ζωτικής σημασίας περιφερειακά (π.χ. ενδεικτικά LED, πλήκτρο reset, κ.λπ).

Εικ. 2



Λοιπές αναπτυξιακές πλακέτες της πλατφόρμας, όπως π.χ. το Arduino MEGA (χρήση του οποίου γίνεται στην πύλη– gateway του συστήματος) προσφέρουν περισσότερα, ή διαφορετικά χαρακτηριστικά. Ειδικότερα, το Arduino MEGA προσφέρει τετραπλάσια μνήμη RAM (8KB έναντι 2KB), οκτώ φορές τη μνήμη προγραμμάτων (256KB έναντι 32KB), περισσότερα περιφερειακά (4 έναντι 1 θυρών UART), αλλά και μεγαλύτερες δυνατότητες για τη διασύνδεση περιφερειακών (54 έναντι 14 GPIO).

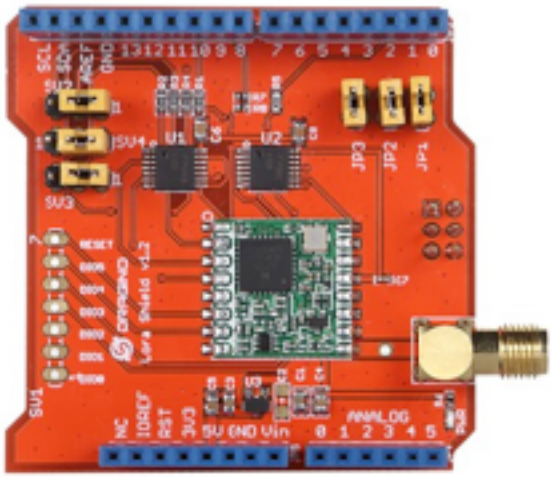
4.3 **Μονάδες Ασύρματης Επικοινωνίας**

Για την ευκολότερη υλοποίηση του συστήματος, επιλέχθηκε η χρήση εμπορικά διαθέσιμων μονάδων επικοινωνίας για κάθε ένα από τα πρωτόκολλα που αξιοποιήθηκαν.

4.3.1 **Μονάδα Dragino (LoRa)**

Το Dragino Shield (Κίνα) (Εικ. 3) αποτελεί μία εύχρηστη πλακέτα επέκτασης για τα Arduino UNO και MEGA, ενώ τοποθετείται επικαθήμενο στην αναπτυξιακή πλακέτα. Αξιοποιώντας τον δίαυλο SPI (Serial Peripheral Interface) που παρέχει ο μικροελεγκτής και των ολοκληρωμένων SX1276/SX1278 της Semtech (Καλιφόρνια, ΗΠΑ), προσθέτει τη δυνατότητα επικοινωνίας στο Arduino με διαμόρφωση LoRa. Η σχεδίαση του, παρόλο που βασίζεται σε υλικά χαμηλού κόστους, επιτυγχάνει αξιοσημείωτη ευαισθησία λήψης (μεγαλύτερη των -148 dBm) και μεγάλη ισχύ εκπομπής (+20 dBm).

Εικ. 3



4.3.2 **Μονάδα RFM69**

Αποτελώντας μία οικογένεια πομποδεκτών της εταιρείας HopeRF (Κίνα) με υψηλό επίπεδο ολοκλήρωσης, τα RFM69 (Εικ. 4) κάνουν χρήση των ελεύθερων συχνοτήτων στα 433, 868 και 915 MHz. Υποστηρίζοντας διαμορφώσεις FSK με GFSK με αλλά και OOK με διαφορετικούς δυνατούς ρυθμούς μετάδοσης δεδομένων, αποτελούν ευέλικτες και προσιτές προτάσεις στο χώρο των ασύρματων δικτύων αισθητήρων, όντας ιδιαίτερα διαδεδομένοι.

Επικοινωνούν με τις αναπτυξιακές πλακέτες του Arduino μέσω σειριακού πρωτοκόλλου SPI, ενώ αξιοποιούν τη δυνατότητα του μικροελεγκτή για αναγνώριση εξωτερικών διακοπών (interrupts) έτσι ώστε να παρέχουν χαμηλότερη ενεργειακή κατανάλωση.

Εικ. 4

4.4 **Αισθητήρες**

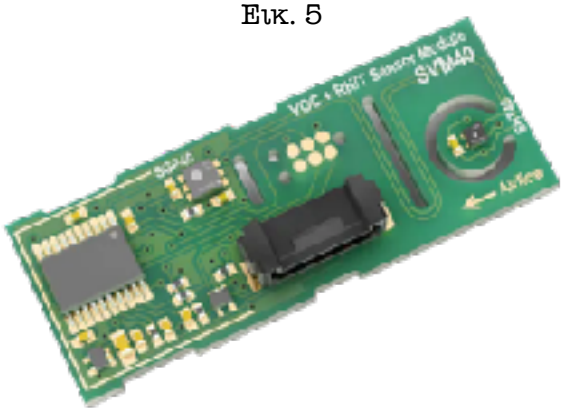
Για την υλοποίηση του συστήματος επιλέχθηκαν τρεις (3) αισθητήρες της εταιρείας Sensirion (Ελβετία), καταξιωμένη στο χώρο των περιβαλλοντικών αισθητηρίων. Κοινό τους χαρακτηριστικό αλλά και παράγοντας επιλογής τους είναι η δυνατότητα για διεπαφή μέσω σειριακού πρωτοκόλλου I2C, το οποίο και διευκόλυνε την διασύνδεσή τους με την αναπτυξιακή πλατφόρμα.

4.4.1 **Αισθητήρας Υγρασίας, Θερμοκρασίας και Π.Ο.Ε. (SVM40)**

Η πλακέτα SVM40 (Εικ. 5) στην πραγματικότητα φιλοξενεί δύο (2) αισθητήρια. Αφενός, έναν εξαιρετικής ακρίβειας αισθητήρα θερμοκρασίας και υγρασίας (SHT40), ικανό για τυπική ακρίβεια ± 0.2°C (στο εύρος 0-65°C) αλλά και για ± 1.8% (στο εύρος 30-70%

RH)., ενώ επιπλέον φιλοξενεί έναν αισθητήρα πτητικών οργανικών ενώσεων (SGP40). Και οι δύο αισθητήρες επικοινωνούν με τον ελεγκτή μέσω σειριακού πρωτοκόλλου I2C (υπό διαφορετικές διευθύνσεις), ενώ παρέχεται και η δυνατότητα επιλογής του πρωτοκόλλου UART.

Εικ. 5

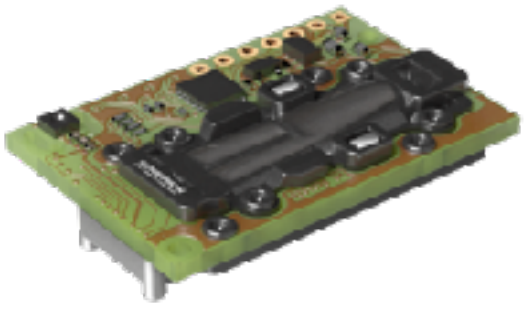


4.4.2 **Αισθητήρας Διοξειδίου του Άνθρακα (SCD30)**

Αξιοποιώντας δύο διαφορετικές τεχνολογίες (υπέρυθρη ανίχνευση, καθώς και μη-διαχεόμενη υπέρυθρη ανίχνευση) για την ποσοτικοποίηση του διοξειδίου του άνθρακα στον αέρα, ο αισθητήρας SCD30 (Εικ. 6) συνδυάζει την ακρίβεια με το χαμηλό κόστος κτήσης. Ικανός να λειτουργήσει στο εύρος 0 - 40.000 ppm με τυπική ακρίβεια ± 30 ppm (στους 25°C) αλλά και με εξαιρετικό χρόνο απόκρισης (20 δευτερόλεπτα), αποτελεί καίριο κομμάτι ενός συστήματος μέτρησης ποιότητας αέρα.

Υποστηρίζει επικοινωνία με τον ελεγκτή είτε μέσω πρωτοκόλλου I2C ή UART, ενώ είναι πλήρως γραμμικός και βαθμονομημένος από την κατασκευάστρια εταιρεία, κάτι που απλοποιεί εξαιρετικά μία κατά τ' άλλα περίπλοκη μέτρηση.

Εικ. 6



4.4.3 **Αισθητήρας Αιωρούμενων Σωματιδίων (SPS30)**

Ο αισθητήρας αιωρούμενων σωματιδίων (ικανός για μέτρηση PM1.0, PM2.5, PM4 και PM10) της Sensirion (Εικ. 7) κάνει χρήση της τεχνικής βασιζόμενης στη διασπορά μιας δέσμης laser πάνω στα αιωρούμενα σωματίδια. Έχει χρόνο ζωής μεγαλύτερο των 10 ετών, χαρακτηριστικό σημαντικό για τη φύση του αισθητηρίου.

Εικ. 7

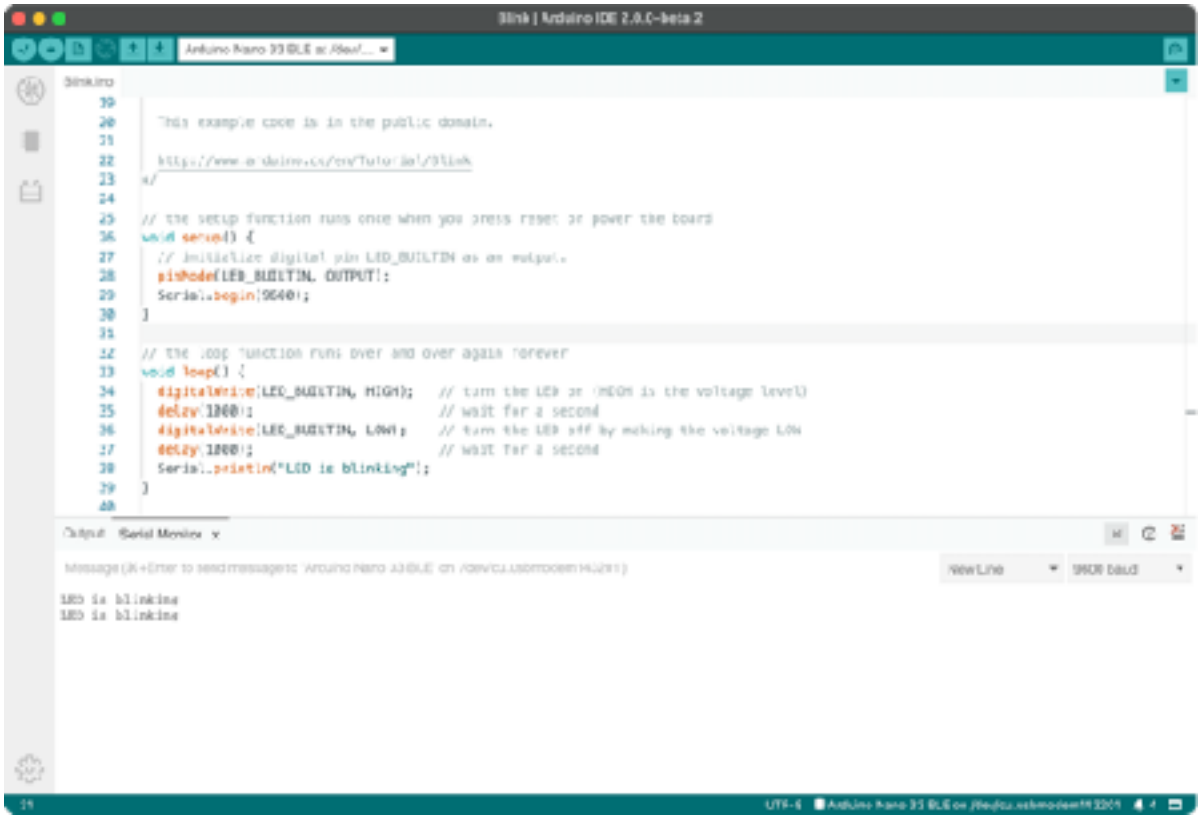
5.1. **Wiring και Arduino IDE**

Η ανοιχτού κώδικα πλατφόρμα Wiring, η οποία βασίζεται στη C++ αλλά χωρίς να υλοποιεί το σύνολο των δυνατοτήτων της, αποτέλεσε τον προπομπό του Arduino. Έτσι σήμερα το Arduino κάνει χρήση της Wiring ως γλώσσας προγραμματισμού των αναπτυξιακών της πλατφόρμας, απορροφώντας την.

Από την άλλη, το Arduino IDE (Εικ. 8) αποτελώντας αναπόσπαστο κομμάτι της πλατφόρμας, αναλαμβάνει το κομμάτι του προγραμματισμού των αναπτυξιακών. Αν και πλέον έχουν κάνει την εμφάνισή τους λύσεις με πολυπληθέστερες δυνατότητες (λ.χ. PlatformIO), εντούτις, το Arduino IDE παραμένει το πιο διαδεδομένο περιβάλλον ανάπτυξης ενσωματωμένου λογισμικού.

Το Arduino IDE παρέχει τη δυνατότητα μεταγλώττισης των προγραμμάτων (μέσω του προγράμματος avrdude το οποίο και ενσωματώνει) αλλά και μεταφόρτωσης στις αναπτυξιακές πλακέτες. Πλέον, φτάνοντας στην δεύτερη έκδοση (2.0) προστίθενται δυνατότητες αποσφαλμάτωσης (debugging) αλλά και ελέγχου εκδόσεων (version control), χαρακτηριστικά που αναβαθμίζουν ουσιαστικά το ρόλο του στην πλατφόρμα.

Εικ. 8



5.2 **Βιβλιοθήκη LMIC (LoRa)**

Η βιβλιοθήκη LMIC, αρχικώς αναπτύχθηκε από την IBM Research GmbH (Ζυρίχη, Ελβετία) και εν συνεχεία τροποποιήθηκε από τους Matthijs Kooijman (Ολλανδία) και Thomas Telkamp – οι οποίοι και ανέπτυξαν τη δυνατότητα για χρήση με το Arduino. Σήμερα, υποστηρίζεται από την εταιρεία MCCI (Νέα Υόρκη, ΗΠΑ), η οποία και συνεχίζει την εξέλιξη της.

Αποτελεί τη δημοφιλέστερη βιβλιοθήκη για την προσθήκη δυνατοτήτων χειρισμού των πομποδεκτών SX1276 μέσα από την πλατφόρμα του Arduino, ενώ αναλαμβάνοντας τον χειρισμό των πομποδεκτών, απελευθερώνει την κύρια εφαρμογή από αυτή την εργασία και εγγυάται τη συμμόρφωση με το πρωτόκολλο.

5.2.1 **Χαρακτηριστικά της LMIC**

Η βιβλιοθήκη υλοποιεί το μεγαλύτερο μέρος της Class A (baseline), ενώ υποστηρίζει τις μπάντες EU868, IN868, US915/AU915, KR920 και AS923. Υποστηρίζει το πρωτόκολλο LoRaWAN έως και την έκδοση 1.0.3 (δεν υποστηρίζει το LoRaWAN 1.1 – 2017). Η βιβλιοθήκη αναλαμβάνει τη διαχείριση όλων των καταστάσεων MAC, αλλά και των χρονικών περιορισμών του LoRaWAN[18].

5.2.2 **Μοντέλο Προγραμματισμού της LMIC**

Η LMIC παρέχει ένα μοντέλο προγραμματισμού βασιζόμενο σε γεγονότα. Ενσωματώνει επίσης εργαλεία για τη διαχείριση εργασιών (job management) και του χρονοπρογραμματισμού (run-time scheduler), αλλά και μία δομή (lmic_t) που χρησιμοποιείται για την ανταλλαγή πληροφοριών ανάμεσα στον κώδικα του χρήστη και το API.

Η εφαρμογή του χρήστη οφείλει να εκκινήσει το περιβάλλον της LMIC, δημιουργώντας μία αρχική εργασία και κάνοντας χρήση της μεθόδου os_init(). Στη συνέχεια και ανά τακτά διαστήματα, θα πρέπει να καλείται ο χρονοπρογραμματιστής εργασιών, με τη μέθοδο os_runloop_once(). Ο τυπικός τρόπος εκκίνησης της LMIC, παρουσιάζεται παρακάτω:

```
osjob_t initjob;

void setup()
{
      os_init();
      os_setCallback(&initjob, initfunc);
}

void loop()
{
      os_runloop_once();
}

static void initfunc (osjob_t* j)
{
      LMIC_reset();
      LMIC_startJoining();
}
```

Η συνάρτηση initfunc() εκκινεί την LMIC και αιτείται την συμμετοχή του κόμβου στο δίκτυο. Η συνάρτηση θα επιστρέψει αμέσως χωρίς να αναμένει απάντηση από τον εξυπηρετητή LoRaWAN, ενώ με οποιαδήποτε έκβαση του αιτήματος θα εμφανιστεί ένα από τα γεγονότα EV_JOINING, EV_JOINED ή EV_JOIN_FAILED.

5.2.3 **Η Δομή Δεδομένων της LMIC**

Αντί της συνεχούς ανταλλαγής δεδομένων ανάμεσα στο API της LMIC και τις κληθείσες μεθόδους, η βιβλιοθήκη παρέχει μία καθολικής εμβέλειας δομή η οποία και μπορεί να προσπελαστεί με σκοπό την πρόσβαση σε πληροφορία σχετικά με την κατάσταση του πρωτοκόλλου.

```
struct lmic_t
{
      u1_t frame[MAX_LEN_FRAME];
      u1_t dataLen;
      u1_t dataBeg;
      u1_t txCnt;
      u1_t txrxFlags;

      …

}
```

Με δεδομένο ότι τα περισσότερα πεδία της δομής βρίσκουν χρήση μόνον εσωτερικά της βιβλιοθήκης, η δομή δεν παρουσιάζεται, ούτε και αναλύεται διεξοδικά. Παρόλα αυτά, τα πεδία που παρουσιάζονται παραπάνω αποτελούν και τα σημαντικότερα πεδία αυτής. Λόγου χάριν, το πεδίο frame[] εμπεριέχει τη χρήσιμη πληροφορία που ανακτήθηκε από τον εξυπηρετητή LoRaWAN, ενώ το πεδίο dataLen, το μήκος αυτής.

5.2.4 **Το API της LMIC**

Παρέχεται πληθώρα μεθόδων για την διαχείριση της LMIC, τον έλεγχο της κατάστασης του πρωτοκόλλου, αλλά και την έναυση διεργασιών πρωτοκόλλου. Οι σημαντικότερες από αυτές, παρουσιάζονται στον παρακάτω πίνακα:

void	**LMIC_reset()**
Επανεκκινεί το επίπεδο MAC. Προηγούμενα δεδομένα συνεδρίας αλλά και εκκρεμείς αποστολές απορρίπτονται.	
bit_t	**LMIC_startJoining()**
Εκκινεί τη διαδικασία συμμετοχής στο δίκτυο. Η μέθοδος αυτή ενδέχεται να κληθεί αυτόματα σε περίπτωση κλήσης άλλης μεθόδου που απαιτεί ενεργή συνεδρία.	
void	**LMIC_setTxData2()**
Προετοιμάζει το πρωτόκολλο για αποστολή δεδομένων στην αμέσως επόμενη διαθέσιμη στιγμή.	
void	**LMIC_clrTxData()**
Απορρίπτει τις εκκρεμείς αποστολές.	
void	**LMIC_registerEventCb()**
Καταχωρεί μία μέθοδο (χρήστη) προς κλήση σε περίπτωση που αφιχθεί ένα ορισμένο γεγονός.	

5.2.5 **Λήψη Δεδομένων από τον Εξυπηρετητή LoRaWAN**

Κατά την άφιξη ενός γεγονότος EV_TXCOMPLETE ή EV_RXCOMPLETE, το πρόγραμμα του χρήστη θα πρέπει να ελέγξει για τυχόν νεοαφιχθέντα ωφέλιμα δεδομένα (payload). Αυτό γίνεται αξιοποιώντας τη δομή της LMIC, μιας και όπως προαναφέρθηκε, τα δεδομένα εμπεριέχονται ήδη σε αυτήν.

Παρακάτω παρουσιάζεται ο τυπικός τρόπος ανάκτησης δεδομένων που προήλθαν από τον εξυπηρετητή LoRaWAN. Η συνάρτηση receiveMessage() είναι ορισμένη από τον χρήστη για τη διαχείριση του μηνύματος που ανακτήθηκε:

```
if (LMIC.dataLen != 0)
{
      u1_t bPort = 0;
      if (LMIC.txrxFlags & TXRX_PORT)
            bPort = LMIC.frame[LMIC.dataBeg - 1];
      receiveMessage(bPort, LMIC.frame + LMIC.dataBeg, LMIC.dataLen);
}
```

5.3 **Βιβλιοθήκη RadioHead (RFM69)**

Η βιβλιοθήκη RadioHead, αναπτύχθηκε και συντηρείται από τον Mike McCauley βασιζόμενη σε προγενέστερες, μικρότερης κλίμακας υλοποιήσεις του ιδίου, ενώ αποτελεί μία πλήρη και αντικειμενοστρεφή λύση για την επικοινωνία μικροελεγκτών και επεξεργαστών με υποστήριξη πληθώρας μονάδων ασύρματης συνδεσιμότητας.

5.3.1 **Η Δομή της RadioHead**

Η βιβλιοθήκη χωρίζεται σε δύο (2) ομάδες κλάσεων. Οι κλάσεις της πρώτης ομάδας, ονόματι “Managers”, παρέχουν διευθυνσιοδοτούμενη αποστολή και λήψη μηνυμάτων με προεραιτική τη δυνατότητα αξιόπιστης επικοινωνίας (επιβεβαιώσεις λήψεων – acknowledgements) αλλά και την τοπολογία πλέγματος (mesh). Από την άλλη, οι κλάσεις της δεύτερης ομάδας που ονομάζεται Drivers, παρέχουν χαμηλού επιπέδου πρόσβαση σε πληθώρα μονάδων ασύρματης επικοινωνίας, μεταξύ των οποίων οι RFM69 και RFM22 (HopeRF), nRF24 και nRF51 (Nordic Semiconductor), SX1276 (Semtech) αλλά και RFM95/96 (HopeRF) αποτελώντας ουσιαστικά, ένα αφαιρετικό στρώμα ανάμεσα στο υλικό και το ανώτερο επίπεδο.

Στον παρακάτω πίνακα αναλύονται οι δυνατότητες καθεμίας από τις τέσσερεις (4) Manager κλάσεις:

`RHDatagram`

Μεταβλητού μήκους μηνύματα χωρίς μηχανισμούς αξιοπιστίας, με προαιρετική δυνατότητα broadcast.

`RHReliableDatagram`

Μεταβλητού μήκους μηνύματα με μηχανισμούς αξιοπιστίας, όπως acknowledgements και επαναποστολές.

`RHRouter`

Μεταβλητού μήκους μηνύματα με μηχανισμούς αξιοπιστίας και αναμετάδοση μεταξύ 0 ή περισσότερων ενδιάμεσων κόμβων, με προ-προγραμματισμένες διαδρομές μεταξύ αυτών.

`RHMesh`

Μεταβλητού μήκους μηνύματα με μηχανισμούς αξιοπιστίας και αναμετάδοση μεταξύ 0 ή περισσότερων ενδιάμεσων κόμβων, με αυτόματη δρομολόγηση και επαναδρομολόγηση.

5.3.2 **Η Κλάση RH_RF69**

Η κλάση παρέχει τη βασική λειτουργικότητα αποστολής και λήψης μηνυμάτων κάνοντας χρήση των πομποδεκτών RFM69, χωρίς όμως την υποστήριξη διευθυνσιοδότησης ή μηχανισμών αξιοπιστίας. Τα επιπλέον αυτά χαρακτηριστικά μπορούν να προστεθούν με τη χρήση των κατάλληλων“Manager” κλάσεων.

Όπως είναι φυσικό, για να επικοινωνήσουν μεταξύ τους δύο πομποδέκτες, πρέπει μεταξύ άλλων να κάνουν χρήση ίδιων συχνοτήτων, αλλά και ίδιας διαμόρφωσης. Η κλάση υποστηρίζει διαμορφώσεις FSK, GFSK αλλά και OOK με πολλούς διαθέσιμους ρυθμούς μετάδοσης (data rates) και ευρών ζώνης με τη δυνατότητα για ρύθμιση του πομποδέκτη να δίνεται μέσα από τη μέθοδο init() με χρήση του σχετικού enumerator (Πίνακας 1):

Πίνακας 1

ENUMERATOR	**Διαμόρφωση**	**Ρυθμός Αποστολής (kbit/s)**	**Εύρος Ζώνης (KHz)**
FSK_Rb2Fd5	FSK	2	5
FSK_Rb2_4Fd4_8	FSK	2.4	4.8
FSK_Rb4_8Fd9_6	FSK	4.8	9.6

FSK_Rb9_6Fd19_2	FSK	9.6	19.2
FSK_Rb19_2Fd38_4	FSK	19.2	38.4
FSK_Rb38_4Fd76_8	FSK	38.4	76.8
FSK_Rb57_6Fd120	FSK	57.6	120
FSK_Rb125Fd125	FSK	125	125
FSK_Rb250Fd250	FSK	250	250
FSK_Rb55555Fd50	FSK	55555	50
GFSK_Rb2Fd5	GFSK	2	5
GFSK_Rb2_4Fd4_8	GFSK	2.4	4.8
GFSK_Rb4_8Fd9_6	GFSK	4.8	9.6
GFSK_Rb9_6Fd19_2	GFSK	9.6	19.2
GFSK_Rb19_2Fd38_4	GFSK	19.2	38.4
GFSK_Rb38_4Fd76_8	GFSK	38.4	76.8
GFSK_Rb57_6Fd120	GFSK	57.6	120
GFSK_Rb125Fd125	GFSK	125	125
GFSK_Rb250Fd250	GFSK	250	250
GFSK_Rb55555Fd50	GFSK	55555	50
OOK_Rb1Bw1	OOK	1	1
OOK_Rb1_2Bw75	OOK	1.2	75
OOK_Rb2_4Bw4_8	OOK	2.4	4.8
OOK_Rb4_8Bw9_6	OOK	4.8	9.6
OOK_Rb9_6Bw19_2	OOK	9.6	19.2
OOK_Rb19_2Bw38_4	OOK	19.2	38.4
OOK_Rb32Bw64	OOK	32	64

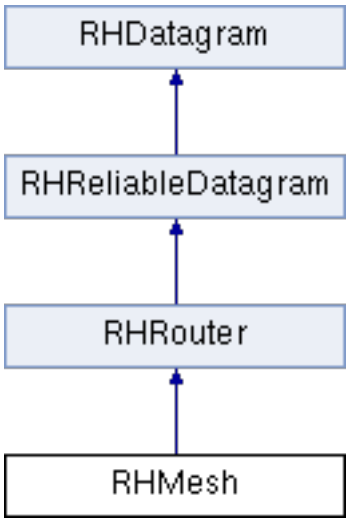
Η χρήση του driver, προϋποθέτει τη δυνατότητα διαχείρισης εξωτερικών διακοπών (interrupts) από τον μικροελεγκτή, μιας και αξιοποιούνται από τον πομποδέκτη κατά την άφιξη νέων μηνυμάτων. Αυτό καθίσταται σαφές και από τον κατασκευαστή (constructor) που παρέχεται από την κλάση:

```
RH_RF69::RH_RF69(uint8_t slaveSelectPin = SS, uint8_t interruptPin = 2,
RHGenericSPI & spi = hardware_spi);
```

5.3.3 **Η Κλάση RHMesh**

Η κλάση RHMesh, κληρονομεί (κατά σειρά) τις RHRouter, RHReliableDatagram και RHDatagram, χτίζοντας επάνω σε αυτές τη δυνατότητα για δικτύωση τοπολογίας πλέγματος. Επιτυγχάνει αυτόματη ανακάλυψη εναλλακτικών διαδρομών με πολλαπλές αναμεταδόσεις, όπου παρίσταται ανάγκη, ενώ χάρη στα χαρακτηριστικά της, δίνει τη δυνατότητα ανάπτυξης δικτύων με άγνωστη, αβέβαιη ή αναξιόπιστη τοπολογία (σε αντίθεση με την RHRouter, η οποία και απαιτεί τις διαδρομές αναμετάδοσης μηνυμάτων να είναι εκ των προτέρων γνωστές).

Σχ. 9



Εξαιτίας της περιπλοκότητάς της, η κλάση RHMesh δεσμεύει σημαντική ποσότητα μνήμης από τον μικροελεγκτή, που ορισμένες φορές αγγίζει τα 2 KB. Αυτό πρακτικά σημαίνει πως ορισμένοι –μικρότερης τάξης- μικροελεγκτές δε μπορούν να κάνουν χρήση αυτής, ενώ ακόμα και το Arduino UNO – η αναπτυξιακή πλακέτα του συστήματος, έρχεται κοντά στα όρια των δυνατοτήτων του.

5.3.3.1 **Εύρεση Διαδρομών προς Άλλους Κόμβους**

Όταν ένας κόμβος βασιζόμενος στην RHMesh εκκινείται, δεν «γνωρίζει» οποιαδήποτε διαδρομή προς άλλους, γειτονικούς κόμβους. Έτσι, ο λεγόμενος πίνακας δρομολόγησης (routing table) είναι άδειος.

Κατά την πραγματοποίηση μίας αποστολής προς οποιοδήποτε κόμβο, τερματικό ή μη, πραγματοποιείται πρώτα έλεγχος στον πίνακα δρομολόγησης ώστε να ανακτηθεί η τελευταία γνωστή διαδρομή προς τον κόμβο προορισμού. Εάν κάτι τέτοιο δεν καταστεί εφικτό, ο κόμβος θα εκπέμψει προς κάθε γειτονικό κόμβο (broadcast) ένα μήνυμα τύπου RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_REQUEST. Οποιοσδήποτε γειτονικός κόμβος που θα λάβει το μήνυμα, ελέγχει πρωτίστως την περίπτωση να είναι ο εαυτός του ο κόμβος προορισμού, όπου σε αυτή την περίπτωση επιστρέφει ένα μήνυμα RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_RESPONSE. Σε αντίθετη περίπτωση, επανεκπέμπει το μήνυμα, αφού πρώτα εγγράψει τον εαυτό του στη λίστα των κόμβων που έχει προσπελάσει το συγκεκριμένο μήνυμα. Εάν η εγγραφή αυτή υπάρχει ήδη, τότε το μήνυμα δεν επανεκπέμπεται, αλλά αγνοείται. Έτσι αποφεύγεται μία κατάσταση καταιγισμού από εκπομπές broadcast δυνητικά καταστρεπτική για το δίκτυο.

Κάνοντας χρήση των περιεχομένων του μηνύματος RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_REQUEST, ο οποιοσδήποτε κόμβος μπορεί να εξάγει τη διαδρομή προς τον κόμβο αφετηρίας, μίας και το μήνυμα περιέχει κάθε κόμβο που έχει «επισκεφτεί» κατά τη διάρκεια αυτής της διαδικασίας. Αυτό σημαίνει πως όταν το μήνυμα φτάσει στον κόμβο προορισμού, θα περιέχει τις διαδρομές για κάθε ενδιάμεσο κόμβο από τον οποίο πέρασε. Έτσι, όταν ο κόμβος προορισμού απαντήσει με το κατάλληλο μήνυμα RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_RESPONSE που σηματοδοτεί τη λήξη της αναζήτησης, θα ακολουθηθεί η ίδια διαδικασία μέχρις ώτου να φτάσει το μήνυμα στον κόμβο αφετηρίας, ο οποίος και ξεκίνησε τη διαδικασία.

Κατά συνέπεια, η εύρεση διαδρομών από έναν κόμβο σε οποιονδήποτε άλλο πραγματοποιείται αξιοποιώντας το συνδυασμό μηνυμάτων RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_REQUEST και RH_MESH_MESSAGE_TYPE_ROUTE_DISCOVERY_RESPONSE. Με την επιτυχή ολοκλήρωση αυτής της διαδικασίας, τόσο οι κόμβοι αφετηρίας και προορισμού, όσο και οι ενδιάμεσοι, θα έχουν εκπαιδευτεί σχετικά με τις πιθανές διαδρομές ανάμεσά τους.

5.3.3.2 **Ενδεχόμενη Αποτυχία Διαδρομής**

Οι επιβεβαιώσεις παράδοσης που επιστρέφονται σε κάθε μετάδοση από την RHMesh, δεν αφορούν την επιτυχημένη παράδοση στον κόμβο προορισμού, αλλά μόνο την παράδοση σε κάθε επόμενο κόμβο (hop-to-hop acknowledgement, όχι end-to-end acknowledgement). Έτσι, δεν υπάρχει η πληροφορία εάν το μήνυμα έφτασε (ή ακόμα και εάν μπορεί πράγματι να φτάσει) στον κόμβο προορισμού.

Στην περίπτωση κατά την οποία ένας κόμβος αποτύχει να μεταδώσει κάποιο μήνυμα στον επόμενο, αποστέλλει ένα μήνυμα τύπου RH_MESH_MESSAGE_TYPE_ROUTE_FAILURE πίσω στον αποστολέα. Οι ενδιάμεσοι κόμβοι, αναγνωρίζουν το μήνυμα αυτό και σε απάντηση, διαγράφουν τη συγκεκριμένη διαδρομή από τον πίνακα δρομολόγησης που έχουν κατασκευάσει. Αυτό σημαίνει ότι σε επόμενη μετάδοση μηνύματος προς τον μη-προσπελάσιμο κόμβο θα επανεκτελεστεί η διαδικασία εύρεσης διαδρομής, αλλά και το ότι το συγκεκριμένο μήνυμα έχει χαθεί.

5.3.1.3 **Το API της RHMesh**

Η βιβλιοθήκη παρέχει ένα απλό, αλλά πλήρες API για τον χειρισμό της, αλλά και την αποστολή- λήψη δεδομένων. Στην περίπτωση της RHMesh, το δημόσιο API αποτελείται πό τρεις (3) μεθόδους, οι οποίες και παρουσιάζονται στον πίνακα που ακολουθεί:

`uint8_t`	**`sendtoWait()`**
Αποστέλλει ένα μήνυμα στον κόμβο προορισμού. Αναζητάει μέσω κλήσης της εσωτερικής μεθόδου route() μία διαδρομή προς τον κόμβο προορισμού στον πίνακα δρομολόγησης, ενώ σε περίπτωση που αυτή δεν υπάρχει, εκκινεί τη σχετική διαδικασία. Επιστρέφει επιβεβαίωση παράδοσης για τον επόμενο στη διαδρομή κόμβο, με μία εκ των τιμών RH_ROUTER_ERROR_NONE (επιτυχημένη παράδοση), RH_ROUTER_ERROR_NO_ROUTE (μη-ύπαρξη διαδρομής στον πίνακα δρομολόγησης) ή RH_ROUTER_ERROR_UNABLE_TO_DELIVER (αποτυχία παράδοσης).	
`bool`	**`recvfromAck()`**

Εκκινεί τον δέκτη (εάν δεν είναι ήδη ενεργός), επεξεργάζεται και προωθεί μηνύματα που έχουν σαν αποδέκτη τρίτους κόμβους, ενώ προσπελαύνει τυχόν ληφθέντα μηνύματα που έχουν σαν αποδέκτη τον εαυτό κόμβο. Στη συνέχεια επιστρέφει επιβεβαίωση παράδοσης στον κόμβο που έστειλε το μήνυμα (όχι αναγκαστικά στον κόμβο αφετηρίας) πλην της περίπτωσης που το μήνυμα είναι broadcast (σε αυτή την περίπτωση δε θα επιστραφεί επιβεβαίωση).

`bool` **`recvfromAckTimeout()`**

Παρόμοια με την recvfromAck(), με κύρια διαφορά το ότι δεσμεύει την εκτέλεση (blocking) του προγράμματος, έως ότου είτε ληφθεί κάποιο μήνυμα, ή λήξει ένα ορισμένο χρονικό διάστημα.

5.4 Βιβλιοθήκες Αισθητήρων

Μικρότερης έκτασης και περιπλοκότητας από τις άνωθεν αναφερθείσες βιβλιοθήκες ασύρματης επικοινωνίας, οι βιβλιοθήκες αισθητήρων αξιοποιούν το σειριακό πρωτόκολλο I2C (ή UART) για την επικοινωνία με τις αισθητήριες μονάδες που αναφέρθηκαν στο κεφάλαιο {4.3}.

5.4.1 Βιβλιοθήκη Αισθητήρα SVM40

Αναπτυχθείσα από την Sensirion, η βιβλιοθήκη του αισθητήρα SVM40 (arduino-i2c-svm40) παρέχει ένα απλό και αντικειμενοστρεφές API για τη λήψη μετρήσεων από αυτόν. Επιστρέφει τιμές μήκους 16- bit, ενώ περιλαμβάνει επιπλέον δυνατότητες, όπως π.χ. τροποποίηση των ρυθμίσεων του αισθητήρα, λήψη πληροφοριών γι' αυτόν, ή επανεκκίνησή του. Το API της βιβλιοθήκης παρουσιάζεται εν συντομία στον παρακάτω πίνακα:

`uint16_t` **`startContinuousMeasurement()`**

Εκκινεί τον αισθητήρια σε λειτουργία συνεχούς προσπέλασης (polling mode). Επιστρέφει μηδέν σε επιτυχημένη ενεργοποίηση, ή κωδικό σφάλματος σε άλλη περίπτωση.

`uint16_t` **`stopMeasurement()`**

Διακόπτη τη λειτουργία του αισθητήρα, ο οποίος και εισέρχεται σε κατάσταση αδράνειας (idle mode). Επιστρέφει μηδέν σε επιτυχημένη ενεργοποίηση, ή κωδικό σφάλματος σε άλλη περίπτωση.

`uint16_t` **`readMeasuredValuesAsIntegers()`**

Επιστρέφει τις τιμές που έχει καταγράψει ο αισθητήρας ως ακέραιους αριθμούς. Οι τιμές ανανεώνονται κάθε ένα (1) δευτερόλεπτο, ενώ τυχόν συχνότερη προσπέλασή τους, θα επιστρέψει την τελευταία διαθέσιμη τιμή. Λόγω της φύσης των επιστρεφόμενων τιμών (ακέραιοι) πραγματοποιείται κλιμάκωσή τους (ειδικότερα, η τιμή Π.Ο.Ε. ανάγεται με κλίμακα 10, η τιμή της υγρασίας του αέρα με κλίμακα 100, ενώ η τιμή της θερμοκρασίας σε βαθμούς Κελσίου, με κλίμακα 200).

`uint16_t` **`deviceReset()`**

Επανεκκινεί τον αισθητήρα.

5.4.2 Βιβλιοθήκη Αισθητήρα SCD30

Αναπτυχθείσα από την εταιρεία SparkFun, η βιβλιοθήκη για τον αισθητήρα διοξειδίου του άνθρακα (SparkFun SCD30 Arduino Library) παρέχει έναν εξαιρετικά απλουστευμένο τρόπο διασύνδεσης και ανάγνωσης του αισθητήρα, ο οποίος και συνοψίζεται στο παρακάτω API:

`bool` **`beginMeasuring()`**

Εκκινεί τον αισθητήρια σε λειτουργία συνεχούς προσπέλασης (polling mode). Η κατάσταση αυτή αποθηκεύετα στην ενσωματωμένη, μη-πτητική μνήμη του αισθητήρα. Έτσι, σε επόμενη εκκίνησή του (εάν π.χ. χαθεί η τροφοδοσία) ο αισθητήρας θα συνεχίσει να δίνει τιμές χωρίς να απαιτηθεί εκ νέου ενεργοποίησή του.

`bool` **`stopMeasurement()`**

Διακόπτει τη λειτουργία συνεχούς προσπέλασης του αισθητήρα, αποθηκεύοντας την κατάσταση αυτή στην ενσωματωμένη, μη-πτητική μνήμη του αισθητήρα.

`uint16_t` **`getCO2()`**

Επιστρέφει την τιμή του διοξειδίου του άνθρακα που μετρήθηκε από τον αισθητήρα. Σε περίπτωση που η τιμή έχει ήδη προσπελαστεί, εκκινείται μία νέα μέτρηση.

`bool` **`dataAvailable()`**

Επιστρέφει true σε περίπτωση που ο αισθητήρας έχει πραγματοποιήσει μία νέα μέτρηση η οποία δεν έχει προσπελαστεί.

5.4.2 **Βιβλιοθήκη Αισθητήρα SPS30**

Η βιβλιοθήκη παρέχεται από τη Sensirion (arduino-sps). Το σύντομο API της παρέχει τις βασικές δυνατότητες για την αξιοποίηση του αισθητήρα SPS30, ενώ οι βασικότερες δυνατότητές του, περιγράφονται στην πίνακα που ακολουθεί:

`int16_t` **`sps30_probe()`**

Ελέγχει το σύστημα για πιθανή παρουσία του SPS30 στο δίαυλο I2C και εφόσον βρεθεί, τον αρχικοποιεί.

`int16_t` **`sps30_start_measurement()`**

Εκκινεί τη διαδικασία μετρήσεων του αισθητήρα. Στη συνέχεια, οι τιμές μπορούν να προσπελαστούν κάθε ένα (1) δευτερόλεπτο.

`int16_t` **`sps30_stop_measurement()`**

Σταματάει τη διαδικασία μετρήσεων του αισθητήρα και τον εισάγει σε λειτουργία αναμονής, μειώνοντας την ενεργειακή κατανάλωση.

`int16_t` **`sps30_data_ready()`**

Διαβάζει τη «σημαία» (flag) data-ready η οποία και υποδεικνύει την ύπαρξη νέων δεδομένων στα οποία δεν έχει πραγματοποιηθεί προσπέλαση.

`int16_t` **`sps30_read_measurement()`**

Λήψη τιμών της τελευταίας μέτρησης. Οι τιμές της μέτρησης γίνονται διαθέσιμες μέσα από μία δομή τύπου sps30_measurement.

`int16_t` **`sps30_read_reset()`**

Επανεκκινεί τον αισθητήρα.

Έπειτα της επιτυχημένης ανάγνωσης των τιμών του αισθητήρα, αυτές τοποθετούνται σε μία δομή δεδομένων τύπου sps30_measurement. Αυτή, αποτελώντας ουσιαστικά ένα σύνολο από float τιμές, αντιπροσωπεύει τη συγκέντρωση αιωρούμενων σωματιδίων στον αέρα για τα διάφορα μεγέθη (π.χ. PM2.5, PM10, κ.λπ) ενώ στη δομή εμπεριέχεται και η μέτρηση για το τυπικό μέγεθος σωματιδίων που ανιχνεύθηκε.

Το σύστημα θα υλοποιηθεί κάνοντας χρήση τεχνολογιών κυρίως ανοιχτής, προσβάσιμης φύσης. Παρόλα αυτά, για λόγους απλούστευσης της υλοποίησης επιλέχθηκε η αξιοποίηση καταξιωμένων παρόχων εικονικών μηχανών έναντι της χρήσης ενός «τοπικού» μηχανήματος. Μεταξύ άλλων, επιλέχθηκε η εταιρεία Vultr (ΗΠΑ), καθότι προσέφερε λύσεις χαμηλού κόστους, συμβατές με τις ανάγκες του συστήματος.

Η διαδικασία δημιουργίας και αρχικών ρυθμίσεων μίας «εικόνας» λειτουργικού συστήματος στο περιβάλλον της Vultr δεν θα αναλυθεί, αφενός διότι ξεφεύγει από το σκοπό της παρούσας διπλωματικής, αφετέρου διότι διαφέρει σημαντικά για κάθε πάροχο.

6.1 **Χαρακτηριστικά Εικονικού Μηχανήματος**

Ο εικονικός εξυπηρετητής (cloud server) εφοδιάζεται με 1 vCore (πυρήνα επεξεργαστή), μνήμη RAM της τάξεως των 1024 MB, αποθηκευτικό χώρο (τεχνολογίας SSD) της τάξεως των 25 GB, καθώς και λειτουργικό σύστημα GNU/Linux (συγκεκριμένα, Ubuntu Server 20.04 x64 LTS). Η IP του μηχανήματος που αποδόθηκε από την Vultr, είναι `108.61.189.206`, αλλά καθίσταται σαφές πως η διεύθυνση αυτή μπορεί να πάψει να είναι έγκυρη σε οποιαδήποτε στιγμή. Για το λόγο αυτό, οπουδήποτε υπάρχει ανάγκη χρήσης της διεύθυνσης IP, αυτή θα αναγράφεται ως `<IP>`.

Για τις ανάγκες της διπλωματικής εργασίας έχει κατοχυρωθεί το όνομα χώρου (domain name) “meteoswarm.com”. Το όνομα χώρου θα παραμείνει στην αποκλειστική κυριότητα, νομή και κατοχή του συντάκτη κατ’ ελάχιστον έως το 2025.

Σαν ενδιάμεσος εξυπηρετητής ονομάτων χώρου (DNS) θα χρησιμοποιηθούν οι υπηρεσίες της CloudFlare (ΗΠΑ). Η προσθήκη του ενδιάμεσου αυτού “επιπέδου” στην απόδοση του ονόματος χώρου σε IP διεύθυνση, παρέχει τη δυνατότητα για ταχύτερες αλλαγές στη διεύθυνση του τελικού εξυπηρετητή, εάν και εφόσον παραστεί τέτοια ανάγκη.

6.2 **Παραμετροποίηση Λειτουργικού Συστήματος**

Για τη διαχείριση του εικονικού εξυπηρετητή, είναι απαραίτητη η εγκαθίδρυση μίας συνεδρίας SSH για την επικοινωνία με αυτόν. Για συστήματα Windows, απαιτείται λογισμικό ικανό να συνδεθεί σε εξυπηρετητές μέσω SSH (π.χ. PuTTY), ενώ για *NIX-based συστήματα (GNU/Linux, MacOS, κ.λπ) αρκεί η απόδοση της παρακάτω εντολής σε ένα τερματικό:

```
ssh root@<IP>
```

Είναι πολύ πιθανό, αν και εφόσον το μηχάνημα από το οποίο επιχειρείται η εγκαθίδρυση της συνεδρίας δεν έχει συνδεθεί στο παρελθόν με τη συγκεκριμένη διεύθυνση, να απαιτηθεί επιβεβαίωση για την σύνδεση.

Στη συνέχεια, θα ζητηθεί το συνθηματικό του χρήστη root, στοιχείο που έχει αποδωθεί αυτόματα από τη Vultr κατά τη δημιουργία του εικονικού μηχανήματος. Το περιβάλλον της Vultr επιτρέπει την εύκολη αντιγραφή – επικόλληση του συνθηματικού για το εικονικό μηχάνημα.

Για λόγους ασφαλείας, είναι σκόπιμο να δημιουργηθεί ένας νέος χρήστης (με όνομα meteoswarm) με περιορισμένα δικαιώματα, για τις ανάγκες της πλατφόρμας. Αυτό, γίνεται εύκολα με την εξής εντολή στο τερματικό:

```
sudo useradd -m meteoswarm
```

Προκειμένου ο νεοεγκαθιδρυθείς χρήστης να έχει τη δυνατότητα σύνδεσης με το εικονικό μηχάνημα, είναι απαραίτητη η δημιουργία ενός νέου συνθηματικού, αποκλειστικά για αυτό τον χρήστη. Στην παρούσα εκτέλεση της υλοποίησης, θα χρησιμοποιηθεί το ίδιο συνθηματικό με τον χρήστη root, κάτι που γενικά δε συστήνεται.

Η εντολή για τη δημιουργία του συνθηματικού στο λογαριασμό του χρήστη meteoswarm, δίνεται ως εξής, ενώ θα ζητηθεί η εισαγωγή του συνθηματικού για δύο (2) φορές, όπου η δεύτερη είναι για επιβεβαίωση:

```
sudo passwd meteoswarm
```

Τέλος, για να μπορέσει ο χρήστης meteoswarm να χρησιμοποιήσει εργαλεία που απαιτούν την εντολή `sudo` (με αυξημένα δικαιώματα), θα πρέπει να προστεθεί στο group “sudo”, ως εξής:

```
usermod -aG sudo meteoswarm
```

Με την επιτυχή ολοκλήρωση των εντολών αυτών, ο χρήστης meteoswarm θα μπορεί να εγκαταστήσει νέες εφαρμογές, καθώς και να τροποποιήσει ρυθμίσεις του συστήματος όπου προηγουμένως θα απαιτούσαν λογαριασμό υπερ-χρήστη (root).

Πλέον, μπορεί να γίνει αποσύνδεση από το λογαριασμό root, με την εντολή `exit` και στη συνέχεια επανασύνδεση στο νέο λογαριασμό, ως εξής:

```
ssh meteoswarm@<IP>
```

6.3 **Περί NGINX**

Αρχικώς δημιουργηθέν από τον Igor Sysoev το 2004, το NGINX είναι ένα πακέτο λογισμικού ανοιχτού κώδικα (άδεια BSD) το οποίο —μεταξύ άλλων- μπορεί να λειτουργήσει σαν εξυπηρετητής HTTP, ή σαν διακομιστής μεσολάβησης (reverse proxy). Στην περίπτωση του υπό ανάπτυξη συστήματος, θα χρησιμοποιηθεί ως διακομιστής μεσολάβησης, ανακτώντας πόρους για λογαριασμό του χρήστη από λοιπούς εξυπηρετητές.

6.3.1 **Εγκατάσταση του NGINX**

Πριν την εγκατάσταση του πακέτου, θεωρείται καλή πρακτική να «ανανεωθούν» τα αποθετήρια λογισμικού της διανομής, έτσι ώστε να περιλαμβάνονται οι νεότερες εκδόσεις των υπό εγκατάσταση προγραμμάτων. Αυτό, πραγματοποιείται με την εντολή:

```
sudo apt update
```

Με το πέρας της εκτέλεσης της εντολής, πραγματοποιείται η εγκατάσταση του πακέτου nginx, με την εντολή:

```
sudo apt install nginx
```

Κατά πάσα πιθανότητα το apt (Διαχειριστής Πακέτων του Debian) θα ζητήσει άδεια για την εγκατάσταση του πακέτου, για την οποία και αρκεί η πληκτρολόγηση του χαρακτήρα `y`.

6.3.2 **Παραμετροποίηση του NGINGX**

Για τον ορισμό των subdomain(s) που θα κληθεί να εξυπηρετήσει ο nginx, γίνεται ορισμός αρχείων κειμένου μέσα στον κατάλογο /etc/nginx/sites-available/, ενώ στη συνέχεια αυτά συνδέονται με τον κατάλογο /etc/nginx/sites-enabled/.

Για τον ορισμό του τομέα —meteoswarm.com- που θα φιλοξενήσει τη διεπαφή για την αλληλεπίδραση με το χρήστη, δημιουργείται ένα νέο αρχείο κειμένου, με τα εξής περιεχόμενα:

```
server
{
  listen 80;

  server_name meteoswarm.com;

  location /
  {
      proxy_pass "http://127.0.0.1:1880/";
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
  }
}
```

6.4 **Περί Node.JS**

Το Node.JS είναι είναι μία πλατφόρμα ανάπτυξης λογισμικού χτισμένη σε JavaScript. Δημιουργήθηκε από τον Ryan Dahl το 2009 ο οποίος και παραμένει υπεύθυνος του έργου μέχρι και σήμερα. Βασίζεται στη μηχανή εκτέλεσης JavaScript “V8” της Google, που κυκλοφόρησε κατά την ίδια περίοδο, ενώ πλέον αποτελεί κραταιά πλατφόρμα ανάπτυξης ασύγχρονων διαδικτυακών εφαρμογών.

6.4.1 **Εγκατάσταση του Node.JS**

Η έκδοση του Node.JS που θα εγκατασταθεί στο σύστημα είναι η 12.x, καθότι συνεργάζεται βέλτιστα με τα λοιπά προαπαιτούμενα προγράμματα.

Με την παρακάτω εντολή, μεταφορτώνεται ένα αρχείο που ορίζει τα αποθετήρια του Node.JS, και ταυτόχρονα προωθείται προς εκτέλεση στο κέλυφος με δικαιώματα υπερχρήστη. Αυτός είναι ένας εύκολος τρόπος να "ενημερωθούν" οι πηγές αποθετηρίων του λειτουργικού συστήματος για την ύπαρξη του Node.JS, δίνοντας στο apt τη δυνατότητα εγκατάστασης του δεύτερου, ενώ με τη δεύτερη εντολή πραγματοποιείται η εγκατάσταση του:

```
curl -fsSL https://deb.nodesource.com/setup_12.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs
```

6.5 **Περί Node-RED**

Το Node-RED αποτελεί ένα εύχρηστο εργαλείο οπτικού προγραμματισμού, αρχικώς αναπτυχθέν από την IBM, με στόχο την ενοποίηση υλικών συσκευών, διαδικτυακών υπηρεσιών και API(s) ως μέρος του Διαδικτύου των Αντικειμένων (Internet of Things - IoT).

Παρέχει ένα εργαλείο επεξεργασίας ροών που «φιλοξενείται» μέσα στον φυλλομετρητή του χρήστη, καθότι κάνοντας χρήση της πλατφόρμας Node.JS, είναι κατασκευασμένο σε JavaScript. Οι ροές αποθηκεύονται σε μορφή JSON, ενώ υποστηρίζει πληθώρα εξωτερικών συνδέσεων και υπηρεσιών, όπως MQTT, The Things Stack, κ.λπ.

6.5.1 **Εγκατάσταση του Node-RED**

Με την πρότερη εγκατάσταση του Node.JS, έχει εγκατασταθεί στο σύστημα και ο npm, πρόγραμμα που δρα ως διαχειριστής πακέτων (παρόμοιος του apt). Αυτό μας επιτρέπει την εύκολη εγκατάσταση του Node-RED, ως εξής:

```
sudo npm install -g --unsafe-perm node-red
```

Τέλος, για την λειτουργία της περιβάλλοντος προγραμματισμού μέσα από τον υποκατάλογο `/backend` (αντί του `/`) αλλά και τη λειτουργία του τελικού dashboard από τον κύριο κατάλογο (`/` αντί για `/ui`), είναι απαραίτητη η επεξεργασία του αρχείο settings.js στον κρυφό κατάλογο `~/.node-red/`. Πιο συγκεκριμένα, οι ρυθμίσεις που πρέπει να τροποποιηθούν, είναι οι εξής:

```
module.exports = { httpAdminRoot: ‘/backend’, … },
ui: { path: “/“ },
…
```

Έτσι, στην επόμενη εκκίνηση του Node-RED, αυτό θα είναι προσβάσιμο από το http://meteoswarm.com/backend (περιβάλλον προγραμματισμού) αλλά και http://meteoswarm.com (διεπαφή χρήστη).

6.5.2 **Αυτόματη Εκκίνηση του Node-RED**

Για την αυτόματη εκκίνηση του Node-RED με την εκκίνηση του συστήματος, δίνεται η εντολή:

```
sudo nano /etc/systemd/system/nodered.service
```

Αυτό θα δημιουργήσει ένα αρχείο με όνομα “nodered.service” στο φάκελο υπηρεσιών του “systemd”. Τα περιεχόμενα του αρχείου οφείλουν να τροποποιηθούν ως εξής:

```
[Unit]
Description=Node-RED
After=syslog.target network.target

[Service]
ExecStart=/usr/bin/node-red --max-old-space-size=256 -v
Restart=on-failure
KillSignal=SIGINT

SyslogIdentifier=node-red
StandardOutput=syslog

WorkingDirectory=/root/
User=root
Group=root

[Install]
WantedBy=multi-user.target
```

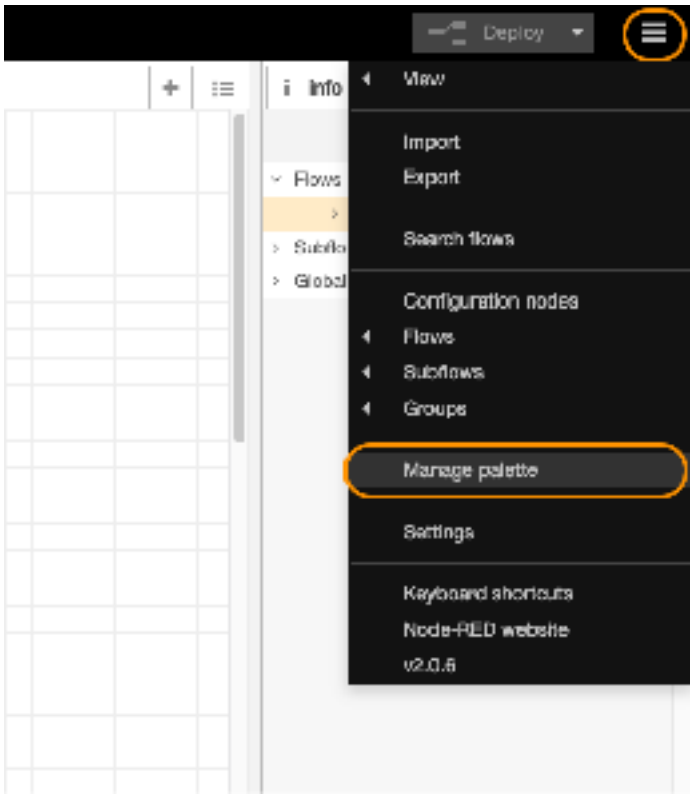
Δίνοντας Ctrl+O και Ctrl+X στο τερματικό, επιτυγχάνεται η αποθήκευση των περιεχομένων του αρχείου και η μετέπειτα έξοδος από το nano (πρόγραμμα επεξεργασίας κειμένου).

Τέλος, για την εγκαθίδρυση της αυτόματης εκκίνησης του Node-RED, δίνεται η παρακάτω εντολή, ενώ με επανεκκίνηση του συστήματος εφαρμόζονται οι αλλαγές.

```
sudo systemctl enable nodered.service
```

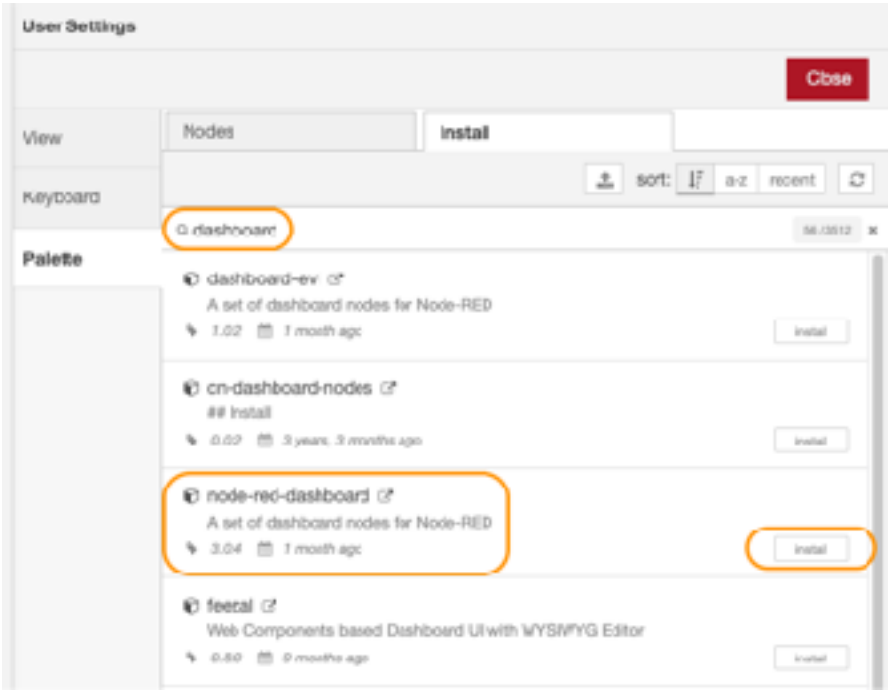
6.5.3 **Εγκατάσταση του Dashboard**

Με το Node-RED ήδη ενεργό, από τη διεύθυνση http://meteoswarm.com/backend γίνεται η εγκατάσταση των επιπλέον στοιχείων που απαιτούνται. Πιο συγκεκριμένα, από το μενού του Node-RED, με την επιλογή “Manage palette” όπως στην (Εικ. 9) γίνεται η διαχείριση των επιπλέον κόμβων.



Εικ. 9

Ο κόμβος που πρέπει να προστεθεί ονομάζεται node-red-dashboard ενώ με το πλήκτρο install, από το αναδυόμενο παράθυρο, πραγματοποιείται η εγκατάστασή του (βλ. Εικ. 10).



Εικ. 10

6.6 **Παραμετροποίηση Τείχους Προστασίας**

Εξ' ορισμού, το τείχος προστασίας στη διανομή της οποίας γίνεται χρήση, είναι ανενεργό. Κρίνεται όμως σκόπιμη η ενεργοποίησή του για λόγους ασφαλείας, μιας και το εικονικό μηχάνημα του συστήματος βρίσκεται στον ιστό. Η διαδικασία παραμετροποίησης του firewall (ονόματι UFW) βασίζεται στο ορισμό κανόνων για την παροχή ή την απαγόρευση πρόσβασης σε κάποια πόρτα ή υπηρεσία.

Οι υπηρεσίες που πρέπει να επιτραπούν, είναι το OpenSSH (πόρτα 22), το HTTP (80), το Node-RED (1880), αλλά και ο NGINX. Αυτές θα επιτραπούν με τις παρακάτω εντολές, οι οποίες όμως πρέπει να δωθούν με δικαιώματα υπερχρήστη:

```
sudo ufw allow 'OpenSSH'
sudo ufw allow 'Nginx Full'
sudo ufw allow 1880
sudo ufw allow 80
```

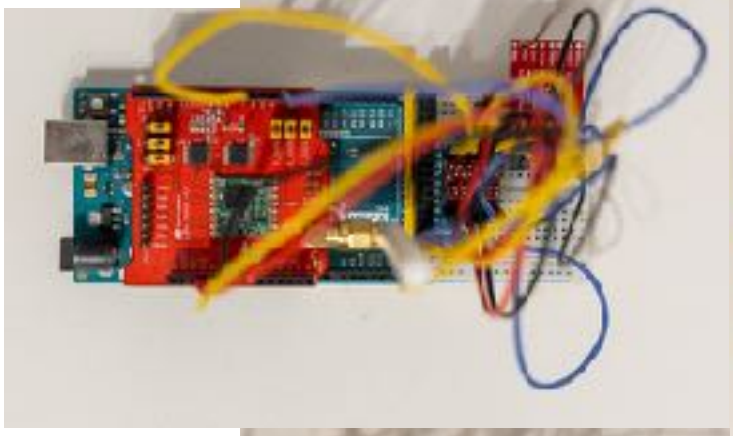
Τέλος, η ενεργοποίηση του τείχους προστασίας γίνεται με την εντολή:

```
sudo ufw enable
```

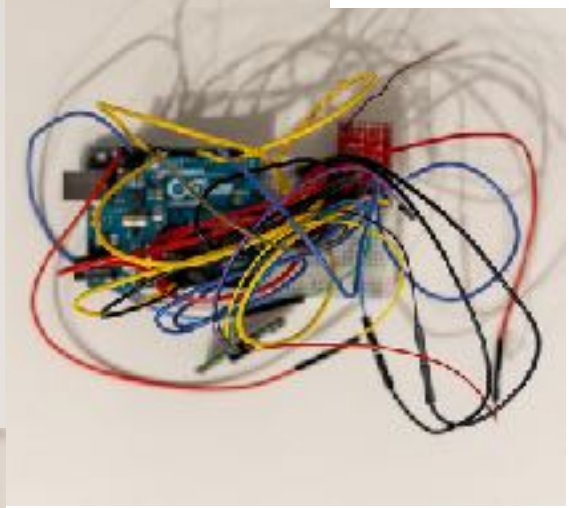
7.1 Κυκλωματική Υλοποίηση

Η (Εικ. 11) παρουσιάζει την κυκλωματική υλοποίηση του κόμβου, ενώ οι (Εικ. 12, 13, 14) την υλοποίηση (κατ' αντιστοιχία) των κόμβων SVM4, SCD30 και SPS30.

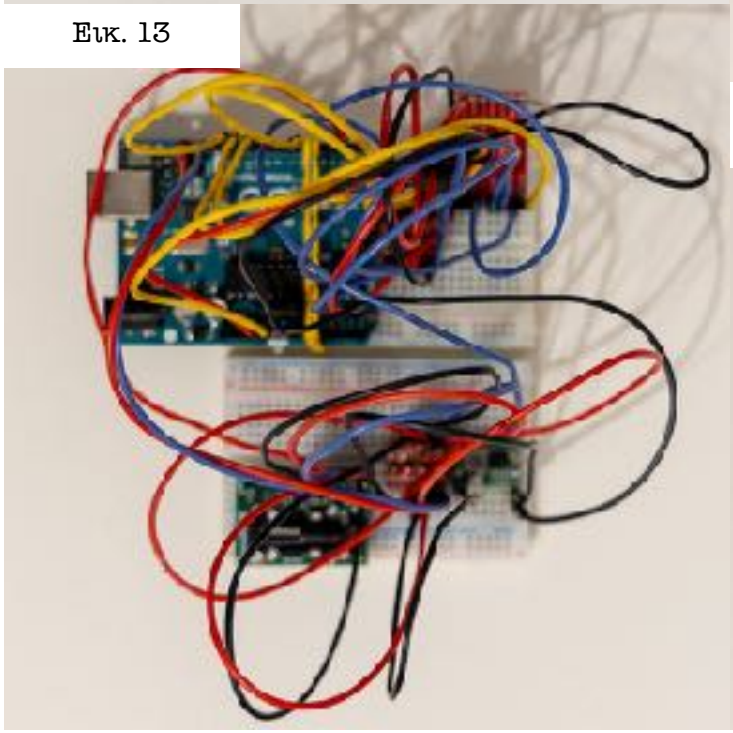
Εικ. 11



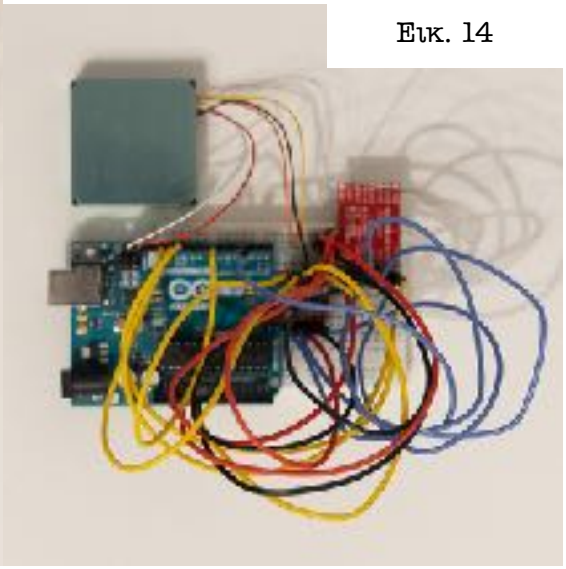
Εικ. 12



Εικ. 13



Εικ. 14



7.2 Πρωτόκολλο Επικοινωνίας

Για την υλοποίηση του συστήματος έχει αναπτυχθεί ένα απλό πρωτόκολλο επικοινωνίας, το οποίο βασιζόμενο σε αλφαριθμητικά, μεταφέρει πληροφορία τόσο για τον κόμβο αφετηρίας ή προορισμού, όσο και για το είδος των μεταφερόμενων δεδομένων.

Πίνακας 2

ENUMERATOR	Είδος Δεδομένων
56_T_20.0	Μέτρηση θερμοκρασίας (20.0°C) από τον κόμβο 56
78_P_120	Μέτρηση αιωρούμενων σωματιδίων (120 PPM) από τον κόμβο 78
12_A_1	Ενεργοποίηση του επενεργητή στον κόμβο 12
17_A_0	Απενεργοποίηση του επενεργητή στον κόμβο 17
34_H_67	Μέτρηση σχετικής υγρασίας αέρα (67%) από τον κόμβο 34
15_C_310	Μέτρηση ποσότητας διοξειδίου του άνθρακα (310 PPM) από τον κόμβο 15
19_V_134	Μέτρηση δείκτη Π.Ο.Ε. (134) από τον κόμβο 19

Το πρωτόκολλο βασίζεται σε τρία (3) πεδία, τα οποία και διαχωρίζονται μεταξύ τους με underscore. Αυτά, αναπαριστούν κατά σειρά τον κόμβο (αφετηρίας ή προορισμού, ανάλογα τη φορά), το είδος των δεδομένων, καθώς και τα ίδια τα δεδομένα.

Η διεύθυνση του κόμβου είναι στο διάστημα [1, 255], ενώ το είδος των δεδομένων παρουσιάζεται στον (Πίνακα 2). Ο κόμβος [0] είναι η πύλη, συνεπώς και αποτελεί δεσμευμένη διεύθυνση.

Στον (Πίνακα 3) παρουσιάζονται παραδείγματα του πρωτοκόλλου:

Πίνακας 3

ENUMERATOR	Είδος Δεδομένων
A	Εντολή προς τον επενεργητή του κόμβου
C	Μέτρηση ποσότητας διοξειδίου του άνθρακα
H	Μέτρηση σχετικής υγρασίας αέρα (RH%)
P	Μέτρηση αιωρούμενων σωματιδίων (PM2.5)
S	Μέτρηση τυπικού μεγέθους αιωρούμενων σωματιδίων
T	Μέτρηση θερμοκρασίας (°C)
V	Μέτρηση δείκτη Π.Ο.Ε.

7.3 Υλοποίηση και Αλγόριθμοι στην Πύλη

Η πύλη του δικτύου, ούσα υπεύθυνη για τη μεταφορά των μετρήσεων από τους κόμβους προς το LoRaWAN, υλοποιεί μέρος από το προαναφερθέν πρωτόκολλο, ενώ αναλαμβάνει και τη διαχείριση των επικοινωνιών (τόσο με το LoRa, όσο και με το υποδίκτυο mesh). Ο πλήρης κώδικας, παρουσιάζεται στο παράρτημα (Π. 1).

7.3.1 Εναλλαγή Μονάδων Επικοινωνίας

Η πύλη οφείλει κάθε φορά που λαμβάνει δεδομένα από το υποδίκτυο mesh, να τα προωθήσει στο LoRaWAN, αλλά και το αντίστροφο. Για να γίνει αυτό, πρέπει κάθε φορά να γίνεται μεταγωγή της επικοινωνίας από τη μονάδα διαμορφωτή – αποδιαμορφωτή (modem) του mesh, στην αντίστοιχη του LoRa, ή αντίστροφα.

Ενώ και οι δύο συσκευές είναι συνδεδεμένες στο δίαυλο SPI του Arduino MEGA, μόνο μία (1) μπορεί να επικοινωνεί κάθε φορά με τον ελεγκτή. Ο εναλλαγή των δύο συσκευών πραγματοποιείται με τον έλεγχο του ακροδέκτη “CS” (chip select) της κάθε μίας από αυτές. Πιο συγκεκριμένα, όταν ο ακροδέκτης κάποιας από τις δύο περιφερειακές συσκευές είναι σε υψηλό δυναμικό (5,0V) αυτή δεν μπορεί να επικοινωνήσει με τον ελεγκτή, και το αντίστροφο.

Κατά συνέπεια, για τον έλεγχο των δύο modem, απαιτούνται δύο (2) ακροδέκτες GPIO. Αυτοί μπορούν να είναι οποιοιδήποτε από αυτούς που προσφέρει το Arduino MEGA, αλλά για λόγους συμβατότητας έχουν επιλεχθεί οι D5 (LoRa) και D7 (RFM69).

Έτσι, η εναλλαγή από το LoRa modem στο RFM69, γίνεται ως εξής:

```
digitalWrite(5, HIGH);
digitalWrite(7, LOW) ;
```

Ενώ με την αντίστροφη διαδικασία, πραγματοποιείται η εναλλαγή από το RFM69 modem στο LoRa:

```
digitalWrite(7, HIGH);
digitalWrite(5, LOW) ;
```

Σημειώνεται ότι είναι απαραίτητο να προηγηθεί η απενεργοποίηση του ενεργού modem πριν την ενεργοποίηση του άλλου και ως εκ τούτου, δίνεται προτεραιότητα στο να τεθεί ο σχετικός ακροδέκτης σε υψηλό δυναμικό.

7.3.2 Έναρξη Επικοινωνίας με το Δίκτυο Πλέγματος

Κατά την εκκίνηση της λειτουργίας του κόμβου, ένα από τα πρώτα πράγματα που εκτελούνται είναι η έναρξη του driver του RFM69, αλλά και η εγκαθίδρυση της λειτουργίας πλέγματος.

Η βιβλιοθήκη RadioHead μέσω των κλάσεων Manager καλεί αυτόματα τους Driver(s) που απαιτούνται. Παρόλα αυτά θα πρέπει να έχει γίνει η δημιουργία των σχετικών αντικειμένων από πριν με τους σχετικούς κατασκευαστές. Η διαδικασία αυτή υλοποιείται ως εξής:

```
RH_RF69 driver(7, 3);
RHMesh manager(driver, 0);

if ( manager.init() )
{
    driver.setFrequency( 434.0 ) ;
    driver.setTxPower( 17, true );
}
```

Στο άνωθεν παράδειγμα δημιουργείται ένα αντικείμενο RH_RF69 με ορίσματα το GPIO που έχει αντιστοιχηθεί στον ακροδέκτη CS του modem (D7), αλλά και το GPIO που έχει αντιστοιχηθεί για την κάλυψη των διακοπών που προέρχονται από αυτό (D3). Στη συνέχεια, ορίζεται η κεντρική συχνότητα στα 434 MHz, ενώ ταυτόχρονα ορίζεται και η ισχύς εκπομπής στα 17 dBm. Σημειώνεται πως, τόσο η συχνότητα εκπομπής, όσο και κεντρική συχνότητα, είναι απαραίτητο να οριστούν για την σωστή λειτουργία του συστήματος.

7.3.3 Συμμετοχή στο Δίκτυο LoRaWAN

Επόμενο στάδιο έπειτα από την εγκαθίδρυση της επικοινωνίας με το δίκτυο πλέγματος, είναι η συμμετοχή στο δίκτυο LoRaWAN. Για την επιτυχή έκβαση του αιτήματος θα απαιτηθούν ορισμένες πληροφορίες από τον εξυπηρετητή δικτύου που αφορούν τόσο την εφαρμογή, όσο και τη συσκευή που κάνει το αίτημα. Αυτές παρουσιάζονται στον (Πίνακα 4).

Πίνακας 4

Όνομα Πληροφορίας	Επεξήγηση

AppEUI	Αναγνωριστικό από το χώρο διευθύνσεων EUI64 του IEEE, που χρησιμοποιείται για να χαρακτηρίσει μοναδικά μία συσκευή.
DevEUI / JoinEUI	Αναγνωριστικό από το χώρο διευθύνσεων EUI64 του IEEE, που χρησιμοποιείται για να χαρακτηρίσει μοναδικά τον εξυπηρετητή δικτύου.
AppKey	Κλειδί κρυπτογράφησης που χρησιμοποιείται κατά τη μετάδοση μηνυμάτων ενεργοποίησης (OTAA).

Οι πληροφορίες αυτές παρέχονται στον κώδικα της πύλης υπό τη μορφή πίνακα δεκαεξαδικών αριθμών. Οι δύο πρώτες (AppEUI και DevEUI/JoinEUI) απαιτούνται σε μορφή Little-Endian, ήτοι, το λιγότερο σημαντικό byte προς τα αριστερά.

```
static const u1_t PROGMEM APPEUI[8]  = { . . . };

void os_getArtEui (u1_t* buf){memcpy_P(buf, APPEUI, 8);}

static const u1_t PROGMEM DEVEUI[8]  = { . . . };

void os_getDevEui (u1_t* buf){memcpy_P(buf, DEVEUI, 8);}

static const u1_t PROGMEM APPKEY[16] = { . . . };

void os_getDevKey (u1_t* buf){memcpy_P(buf, APPKEY, 16);}
```

Στη συνέχεια, γίνεται η ανάθεση των ακροδεκτών οι οποίοι και έχουν ανατεθεί στο LoRa modem. Η βιβλιοθήκη LMIC χρησιμοποιεί μία δομή τύπου lmic_pinmap για την ανάθεση των ακροδεκτών, ενώ οι απολύτως απαραίτητες πληροφορίες για την λειτουργία του modem, είναι ο ακροδέκτης CS (D5), αλλά και DIO0, DIO1 (D2, D6). Οι λοιποί ακροδέκτες, παρόλο που προσφέρουν επιπλέον λειτουργικότητα, στην παρούσα υλοποίηση δεν αξιοποιούνται.

```
const lmic_pinmap lmic_pins =
{
  .nss = 5,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = LMIC_UNUSED_PIN,
  .dio = {2, 6, LMIC_UNUSED_PIN},
};
```

Τέλος, για την εκκίνηση της LMIC και την αποστολή του αιτήματος συμμετοχής στο LoRaWAN, εκτελείται μία φορά ο παρακάτω κώδικας.

```
os_init();

LMIC_reset();
LMIC_startJoining();

LMIC_EXIT_FLAG = false;

do
{
     os_runloop_once();
} while ( !LMIC_EXIT_FLAG );
```

Για την διακοπή του χρονοπρογραμματιστή, χρησιμοποιείται το flag "LMIC_EXIT_FLAG", το οποίο και ενεργοποιείται από την άφιξη του σχετικού γεγονότος.

7.3.4 **Διαχείριση Εισερχόμενων Δεδομένων από το Δίκτυο Πλέγματος**

Η υλοποίηση εγγυάται ότι το δίκτυο πλέγματος θα βρίσκεται σε επικοινωνία με την πύλη για ένα παράθυρο με χρόνο —το λιγότερο- 20 δευτερόλεπτα, με βήμα ελέγχου τα 2 δευτερόλεπτα.

Σε κάθε επανάληψη της διαδικασίας εντός του «παραθύρου», το modem παραμένει ενεργό και η RadioHead ελέγχει για νέα δεδομένα. Εάν υπάρξει άφιξη δεδομένων, αυτά καταγράφονται με την μορφή που ορίζει το πρωτόκολλο, ενώ αποθηκεύονται προσωρινά.

Στη συνέχεια, πριν την ενσωμάτωση του νέου μηνύματος στα υπάρχοντα, γίνεται ένας τυπικός έλεγχος μήκους, έτσι ώστε να υπάρξει βεβαιότητα ότι δεν παραβιάζεται το πρωτόκολλο LoRaWAN. Εάν το μήκος είναι αποδεκτό, τα δεδομένα ενσωματώνονται στα προηγούμενα με την παύλα (-) να χρησιμοποιείται σαν χαρακτήρας διαχωρισμού, ενώ σε άλλη περίπτωση, απορρίπτονται.

Ακολουθεί μέρος του κώδικα που υλοποιεί τις προαναφερθείσες διαδικασίες.

```
while ( millis() < lastMeshTransmissionTimestamp + MESH_MINIMUM_ON_TIME *
1000 )
{
      if (manager.available())
      {
            uint8_t len = sizeof(buf);
            uint8_t from;

            if (manager.recvfrom(buf, &len, &from))
            {
                  String message;
                  message.concat(from);
                  message.concat("_") ;

                  message.concat(buf);
                  message.concat("_");

                  if (strlen(uplink) + message.length() < UPLINK_LENGTH)
                        strcat(uplink, message.c_str());
            }
      }
}
```

7.3.5 **Προώθηση στο Δίκτυο Κορμού**

Με τη λήξη του παραθύρου λήψης από το δίκτυο πλέγματος, εκκινείται η διαδικασία αποστολής των δεδομένων που έχουν συλλεχθεί στο δίκτυο κορμού.

Η διαδικασία ξεκινάει με την μέτρηση του μήκους των συλλεχθέντων δεδομένων, καθώς εάν αυτό είναι μηδενικό, δεν υπάρχει λόγος να πραγματοποιηθεί μετάδοση. Στη συνέχεια, απενεργοποιείται το modem του δικτύου πλέγματος, ενώ ενεργοποιείται το modem του LoRa, ενώ καλείται η LMIC για την προετοιμασία του πλαισίου αποστολής. Τέλος, εκτελείται σε επανάληψη ο χρονοπρογραμματιστής της LMIC, μέχρις ότου εμφανιστεί κάποιο γεγονός ικανό να τερματίσει τη διαδικασία.

Έπειτα της επιτυχούς αποστολής των δεδομένων, εμφανίζεται το γεγονός EV_TXCOMPLETE. Με την εμφάνισή του ορίζεται το flag εξόδου από την εκτέλεση του χρονοπρογραμματιστή, ενώ επιπλέο, διαγράφονται από την μνήμη τα –αποσταλθέντα- δεδομένα.

7.3.6 Διαχείριση Εισερχόμενων Δεδομένων από το Δίκτυο Κορμού

Με δεδομένο ότι η πύλη λειτουργεί στην τάξη Α του πρωτοκόλλου LoRaWAN, είναι γνωστό πως υφίσταται παράθυρο λήψης δεδομένων μόνο ύστερα από κάθε αποστολή. Το γεγονός αυτό απλοποιεί ιδιαίτερα τον χειρισμό των εισερχόμενων δεδομένων.

Τα κατερχόμενα δεδομένα, ακολουθούν μεν το ίδιο πρωτόκολλο (ενότητα 7.1) με τα ανερχόμενα, αν και είναι σαφές ότι το δεύτερο πεδίο σε κάθε κατερχόμενο μήνυμα θα είναι τύπου «Α» (εντολή προς τον επενεργητή).

Η πύλη, παρόλα αυτά, αναλαμβάνει μόνο την ανάγνωση του πρώτου πεδίου (διεύθυνση κόμβου προορισμού) και την προώθηση του μηνύματος στον σωστό κόμβο. Η περεταίρω ανάλυση του μηνύματος και η λήψη των αποφάσεων, πραγματοποιείται από τον ίδιο τον κόμβο, μόλις το μήνυμα παραδοθεί σε αυτόν.

7.4 Υλοποίηση και Αλγόριθμοι στους Κόμβους

Στους κόμβους του συστήματος υλοποιείται το μέρος από το πρωτόκολλο που αφορά το είδος των δεδομένων που αποστέλλονται. Επιπλέον, αναλύονται τα εισερχόμενα μηνύματα, ώστε να πραγματοποιηθεί η σωστή ενέργεια από τον κόμβο (ενεργοποίηση ή απενεργοποίηση επενεργητή). Ο κώδικας των τριών κόμβων παρουσιάζεται στα παραρτήματα (Π.2, Π.3, Π.4).

7.4.1 Χρονικός Προγραμματισμός Αποστολών Δεδομένων

Ως ένα μέτρο αποφυγής συγκρούσεων, η συχνότητα αποστολής του κάθε μηνύματος είναι τυχαία, ορισμένη όμως σε ένα διάστημα [10, 60] δευτερολέπτων.

Για να επιτευχθεί κάτι τέτοιο, δημιουργούνται ψευδοτυχαίοι αριθμοί σε αυτό το διάστημα από τη γεννήτρια αριθμών που περιλαμβάνει η πλατφόρμα του Arduino η οποία και αρχικοποιείται με μία ανάγνωση θορύβου από τον αναλογικό-ψηφιακό μετατροπέα (ADC) της πλακέτας.

7.4.2 Διαχείριση Ληφθέντων Δεδομένων

Έχει καταστεί σαφές από προηγούμενη ενότητα ότι η ανάλυση των μηνυμάτων που προέρχονται από την εφαρμογή είναι ευθύνη του κόμβου. Στην πραγματικότητα, τα μηνύματα αυτά μπορούν να πάρουν μόνον δύο (2) τιμές, μεταξύ των “A_1” και “A_0”.

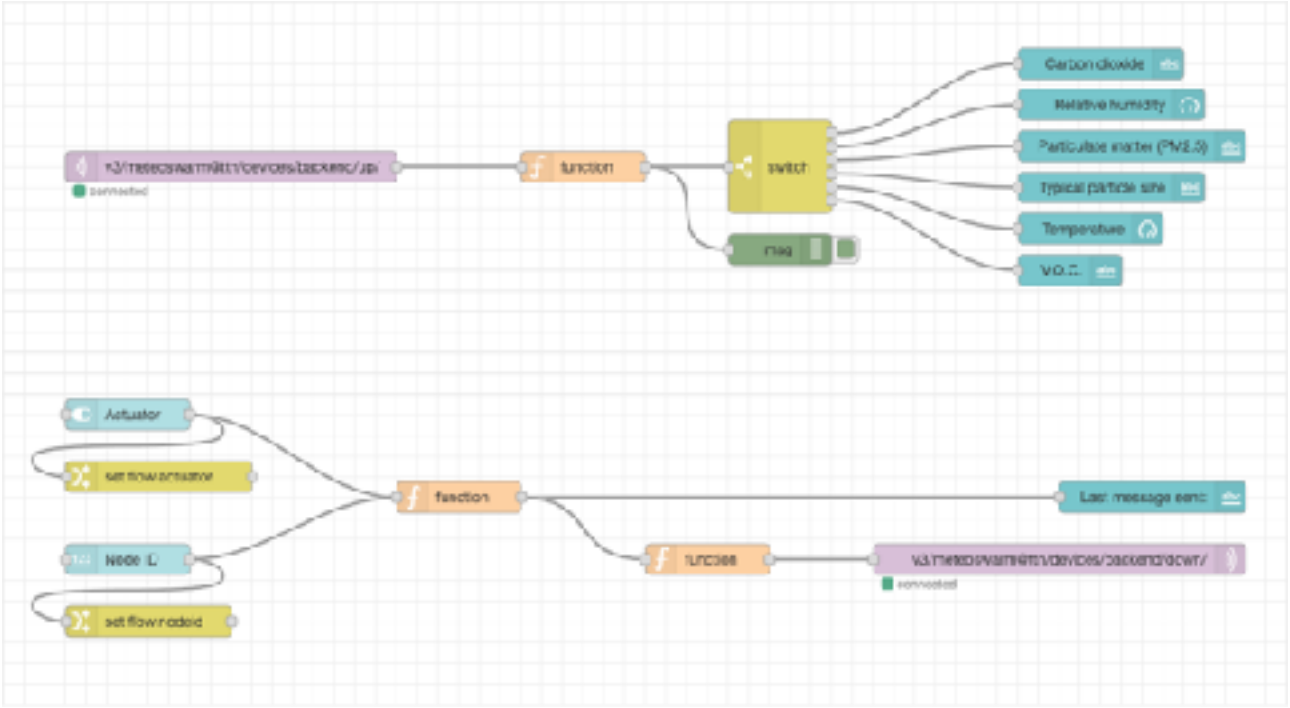
7.5 Υλοποίηση Διεπαφής Χρήστη

Τόσο ο χειρισμός του συστήματος (ενεργοποίηση και απενεργοποίηση επενεργητών) όσο και η παρουσίαση των δεδομένων, πραγματοποιούνται μέσα από το περιβάλλον οπτικού προγραμματισμού που παρέχει το Node-RED.

Η διασύνδεση με τον εξυπηρετητή εφαρμογής πραγματοποιείται μέσω πρωτοκόλλου MQTT, δυνατότητα που παρέχει από το ίδιο το The Things Stack. Έτσι, το μόνο που απαιτείται για την ανταλλαγή των δεδομένων με το Node-RED, είναι η σύνδεση με τον αντίστοιχο εξυπηρετητή, αφού πρώτα ανταλλαχθούν τα σχετικά κλειδιά.

Στο (Σχ. 9) παρουσιάζεται το σύνολο της ροής που έχει σχεδιαστεί για την εξ’ αποστάσεως αξιοποίηση του συστήματος. Το κομμάτι της παρουσίασης των δεδομένων βασίζεται στο dashboard του Node-RED, το οποίο και εγκαταστάθηκε προηγουμένως.

Σχ. 9



7.5.1 **Διαχωρισμός των Ανερχόμενων Δεδομένων**

Τα δεδομένα που προέρχονται από τους κόμβους, και άρα από το δίκτυο LoRaWAN, ακολουθούν τη μορφή του πρωτοκόλλου που αναλύθηκε στην ενότητα [7.1]. Έτσι, καθίσταται σαφές ότι πριν την παρουσίασή τους, αυτά θα πρέπει να διαχωριστούν, αλλά και να αναγνωριστούν, ώστε να αποδωθούν στα κατάλληλα λεκτικά.

Το πρώτο βήμα, είναι ο διαχωρισμός του αλφαριθμητικού σε έναν πίνακα, για την ευκολότερη διαχείρισή του. Το Node-RED δεν παρέχει κάποιο εργαλείο που να μπορεί να πραγματοποιήσει άμεσα αυτή την εργασία, οπότε και επιλέχθηκε ο είδος κόμβου "function", ο οποίος και επιτρέπει την υλοποίηση μίας λειτουργίας σε «καθαρή» JavaScript. Ο κώδικας που υλοποιεί το διαχωρισμό, είναι ο εξής:

```
var messages = [];
var msgs_str = msg.payload.split("_");

for (let i = 0; i < msgs_str.length; i++)
{
    messages.push(msgs_str[i]);
}

msg.payload = messages;
return msg;
```

Η λογική του, βασίζεται στην ιδέα του διαχωρισμού του αλφαριθμητικού με βάση το underscore ("_"). Ο χαρακτήρας αυτός αξιοποιείται από το πρωτόκολλο για το διαχωρισμό της πληροφορίας εντός του μηνύματος. Στη συνέχεια, το κάθε κομμάτι της πληροφορίας, προωθείται σε έναν πίνακα. Έτσι, για τα τρία (3) τμήματα της πληροφορίας, ο πίνακας αποτελείται –αντίστοιχα- από τρία (3) στοιχεία.

Στη συνέχεια, αξιοποιείται το είδος κόμβου μεταγωγικού διακόπτη (switch) το οποίο και ανακατευθύνει την –ήδη διαχωρισθείσα- πληροφορία σε μία από τις εξόδους του ανάλογα με τους χαρακτήρες που αυτή περιέχει.

7.5.2 **Αποστολή Δεδομένων στους Κόμβους**

Δύο είδη κόμβων –μέρη του dashboard- αξιοποιούνται για την εύκολο χειρισμό του αριθμού του κόμβου προς έλεγχο. Συγκεκριμένα, ο κόμβος "numeric" παρέχει έναν επιλογέα ο οποίος και δίνει στον χρήστη την επιλογή για μία τιμή στο διάστημα [1...254]. Ταυτόχρονα, ο κόμβος "switch" (του dashboard) δημιουργεί έναν διακόπτη εναλλαγής (toggle switch). Αλλαγή σε οποιοδήποτε από τα δύο (2) αυτά στοιχεία ελέγχου πυροδοτεί τη διαδικασία κατασκευής του μηνύματος αλλά και της αποστολής του.

Τα δεδομένα στην πλατφόρμα του Node-RED, αποθηκεύονται είτε με καθολική (global.), εντός ροής (flow.) ή εντός κόμβου (msg.) εμβέλεια. Για να είναι προσπελάσιμα όμως τα δεδομένα ενός κόμβου από έναν εξωτερικό, αυτά θα πρέπει να έχουν εμβέλεια – κατ' ελάχιστον- σε επίπεδο ροής. Για το σκοπό αυτό αξιοποιείται ένας κόμβος τύπου "change", ο οποίος και μεταφέρει το msg.payload (περιεχόμενο μηνύματος) από τον κόμβο δημιουργίας, στο επίπεδο ροής.

Στη συνέχεια, και με χρήση JavaScript, τα δεδομένα ανακτώνται από το επίπεδο ροής, ενώ χτίζεται το μήνυμα που θα κατευθυνθεί στους φυσικούς κόμβους.

```
var nid = flow.get("nodeid");
var act = flow.get("actuator");

msg.payload = nid + "_A_" + act;

return msg;
```

Τέλος, και πριν τη μετάδοσή του, το μήνυμα κωδικοποιείται κατά Base-64, μορφή η οποία και γίνεται αποδεκτή από το The Things Stack.

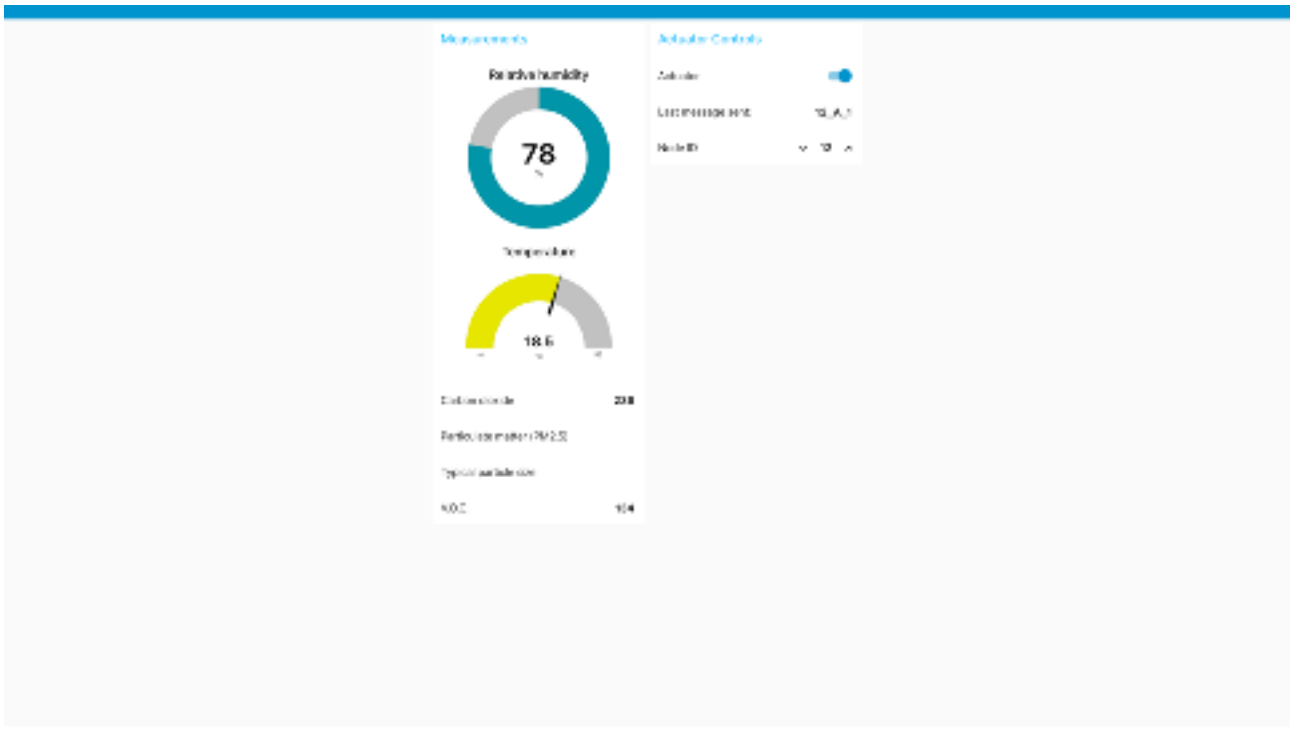
```
return {
  "payload": {
    "downlinks": [{
      "f_port": 1,
      "frm_payload": msg.payload.toString("base64"),
      "priority": "NORMAL"
    }]
  }
}
```

7.5.3 **Dashboard στο Node-RED**

Προερχόμενο από το πακέτο “dashboard”, τα στοιχεία που αναλύθηκαν μέχρι τώρα, παρέχουν έναν εύκολο και γραφικό τρόπο αλληλεπίδρασης των δεδομένων με τον χρήστη. Πιο αναλυτικά, το πακέτο dashboard παρέχει ένα σύνολο συστατικών, τα οποία μπορούν να αξιοποιηθούν για την κατασκευή μίας περιεκτικής οπτικής διεπαφής (user interface).

Η διεπαφή που δημιουργήθηκε για τις ανάγκες της παρούσας εργασίας παρουσιάζεται στην (Εικ. 15).

Εικ. 15



Η γραφική διεπαφή αποτελείται από δύο (2) στήλες, η πρώτη εκ των οποίων παρουσιάζει τα προσφάτως συλλεχθέντα δεδομένα. Συμπεριλαμβάνει ένα donut graph για τη σχετική υγρασία του ατμοσφαιρικού αέρα (στο εύρος 0-100 %RH), ένα gauge graph για τη θερμοκρασία (στο εύρος -35 - 55 °C), αλλά λεκτικές αναπαραστάσεις των λοιπών μετρήσεων, οι οποίες και δεν έχουν σαφώς ορισμένα όρια.

Στον αντίποδα η δεύτερη στήλη δίνει τη δυνατότητα ελέγχου των επενεργητών πάνω σε οποιοδήποτε από τους κόμβους του συστήματος. Η διαδικασία αυτή επιτελείται με χρήση είτε του toggle switch ή του numeric επιλογέα, ενώ οποιαδήποτε μεταβολή στα δύο αυτά χειριστήρια, θα αποστείλει αυτόματα το σχετικό μήνυμα προς τον κόμβο.

Στο κεφάλαιο αυτό, που αποτελεί επίλογο της διπλωματικής εργασίας, παρουσιάζονται τα συμπεράσματα που εξήχθησαν κατά την εκπόνηση αυτής, ενώ προτείνονται και πιθανές μελλοντικές τροποποιήσεις και επεκτάσεις του συστήματος για περεταίρω ανάδειξη των δυνατοτήτων του.

Σε κάθε περίπτωση, καθίσταται σαφές ότι σε ένα σύστημα πραγματικής λειτουργίας δύναται να υπάρχει μεγάλο πλήθος ανομοιόμορφα κατανεμημένων στο χώρο κόμβων, ή ακόμα και πολλαπλά, γειτονικά δίκτυα.

8.1 **Συμπεράσματα**

Αποτέλεσμα της διεξαγωγής της παρούσας εργασίας είναι η απόδειξη της δυνατότητας κατασκευής ενός ασύρματου δικτύου αισθητήρων με χαμηλού κόστους και ευρέως διαθέσιμα υλικά.

Αναδείχθηκε η αξία της διαμόρφωσης LoRa στα ασύρματα δίκτυα αισθητήρων, ενώ επιπλέον φάνηκαν και οι περιορισμοί του πρωτοκόλλου LoRaWAN.

8.2 **Πιθανές Μελλοντικές Εξελίξεις**

Η τρέχουσα δομή του συστήματος, ενώ επιτυγχάνει τη βασική λειτουργικότητα — που είναι η μετάδοση των τιμών των αισθητήρων— δεν επιτρέπει την ανάπτυξη μίας πλήρους και ρεαλιστικά αξιοποιήσιμης πλατφόρμας μετρήσεων. Παρακάτω παρουσιάζονται ορισμένες πιθανές επεκτάσεις της πλατφόρμας.

8.2.1 **Μηχανισμοί Δυναμικής Προσθαφαίρεσης Κόμβων**

Η τρέχουσα υλοποίηση δεν επιτρέπει τη δυναμική προσθαφαίρεση κόμβων, στοιχείο που θα έκανε το δίκτυο ικανό να αναπτυχθεί ταχύτερα, αλλά και σε πιο απαιτητικά περιβάλλοντα. Έτσι, για την ουσιαστική εκμετάλλευση του δικτύου με πλήρη αξιοποίηση της τοπολογίας πλέγματος, θεωρείται απαραίτητη η προσθήκη μηχανισμών για την άμεση και δυναμική προσθήκη ή αφαίρεση κόμβων στο ή από το δίκτυο, χωρίς να υφίσταται η ανάγκη επαναπρογραμματισμού των συσκευών.

8.2.2 **Μηχανισμοί Δυναμικής Παραμετροποίησης Κόμβων**

Η δυναμική παραμετροποίηση των χαρακτηριστικών λειτουργίας των κόμβων θα επέτρεπε της αμεσότερη επέμβαση από το διαχειριστή του δικτύου, όπου (και αν) αυτό κρινόταν απαραίτητο. Παραδείγματος χάριν, σε μία κατάσταση όπου η ατμόσφαιρα θα επιβαρυνόταν σημαντικά σε μικρό χρονικό διάστημα, θα μπορούσε ο διαχειριστής του δικτύου να τροποποιήσει τις σχετικές συσκευές, ώστε αυτές να αποστέλλουν συχνότερα μετρήσεις.

8.2.3 **Ενιαίο Στρώμα Διεπαφής Αισθητήρων**

Με δεδομένη την πολυποικιλότητα των αισθητήριων μονάδων που διατίθενται στην αγορά αλλά και τις διαφορετικές φυσικές διεπαφές που αυτοί αξιοποιούν, η σχεδίαση και κατασκευή ενός κόμβου συμβατού με το σύνολο αυτών είναι ασύμφορη, αλλά και ιδιαίτερα απαιτητική. Η ανάπτυξη ενός ενδιάμεσου στρώματος ανάμεσα στα αισθητήρια και στους κόμβους κρίνεται μονόδρομος για την εύκολη αξιοποίησή τους.

Ειδικότερα, με τη σχεδίαση και κατασκευή πλακετών οι οποίες θα κάνουν χρήση ενός χαμηλού κόστους μικροελεγκτή που να φέρει σειριακά πρωτόκολλα (λ.χ. I2C), είναι εφικτή η απόκρυψη της περιπλοκότητας του κατώτερου επιπέδου, αλλά και η ευκολότερη διασύνδεση οποιουδήποτε αισθητηρίου με τον κόμβο, μιας και σε κάθε περίπτωση ο κόμβος θα αξιοποιεί ένα ―κοινό- πρωτόκολλο.

8.2.4 **Δίκτυο Πλέγματος Δικτύων Πλέγματος**

Με την τρέχουσα υλοποίηση η τοπολογία πλέγματος αξιοποιείται μόνο στο εσωτερικό των υποδικτύων και όχι μεταξύ τους. Αυτό πρακτικά σημαίνει ότι κάθε υποδίκτυο έχει μία (1) και μοναδική διαδρομή επικοινωνίας με το δίκτυο κορμού, γεγονός που εύκολα μπορεί να οδηγήσει σε αστοχία της επικοινωνίας.

Η εφαρμογή της τοπολογίας πλέγματος ανάμεσα στις πύλες, θα προσέφερε τη δυνατότητα πολλών, εναλλακτικών διαδρομών ανάμεσα στα δίκτυα, έτσι ώστε να αποφεύγεται τόσο η συμφόρηση σε συγκεκριμένες ―μεγάλης κυκλοφορίας- διαδρομές, όσο και η ενδεχόμενη αποτυχία μετάδοσης μηνυμάτων από αυτά.

8.2.5 **Χρήση Πολλαπλών Δικτύων Κορμού**

Στην τρέχουσα υλοποίηση γίνεται χρήση —κατ' αποκλειστικότητα- του LoRaWAN ως μέσο διασύνδεσης των κόμβων με το διαδίκτυο. Όμως, από τη φύση της πλατφόρμας καθίσταται σαφές ότι θα μπορούσαν να υπάρξουν πολλαπλά δίκτυα κορμού, με διαφορετικά χαρακτηριστικά, ικανά να εξυπηρετήσουν διαφορετικές ανάγκες.

Παραδείγματος χάριν, σε ένα μεγαλύτερο, πιο πυκνό δίκτυο, ενδέχεται η δυνατότητα μεταφοράς δεδομένων που παρέχει το LoRaWAN να μην επαρκεί για την κάλυψη του πλήθους των κόμβων, ιδίως εάν αυτοί —για κάποιο λόγο- αποστέλλουν μετρήσεις σε τακτά διαστήματα.

Έτσι, ορισμένα —πιθανώς πυκνότερα- δίκτυα τα οποία βρίσκονται εντός αστικών ιστών και άρα δεν έχουν την ανάγκη μεγάλης εμβέλειας που παρέχει το LoRa, θα μπορούσαν βασιζόμενα σε εναλλακτικές τεχνολογίες με ικανότητα χειρισμού μεγαλύτερου όγκου δεδομένων (λ.χ. Ethernet, LTE) να το παρακάμπτουν.

Π.1 Κώδικας στην Πύλη

```
#include <RHMesh.h>
#include <RH_RF69.h>

#include <lmic.h>
#include <hal/hal.h>

#include <SPI.h>
#include <string.h>

char src;
void do_send();

#define SERVER_ADDRESS 0

bool LMIC_EXIT_FLAG;

char buf[RH_RF69_MAX_MESSAGE_LEN];

const unsigned DNLINK_LENGTH = 16 ;
const unsigned UPLINK_LENGTH = 255;

char dnlink[DNLINK_LENGTH];
char uplink[UPLINK_LENGTH];

String dn;
String up;

RH_RF69 driver(7, 3);

RHMesh manager(driver, SERVER_ADDRESS);

const unsigned LORA_TRANSMISSION_INTERVAL = 20;
const unsigned MESH_MINIMUM_ON_TIME =       2;

unsigned long long lastLoRaTransmissionTimestamp = 0;
unsigned long long lastMeshTransmissionTimestamp = 0;

unsigned long long lastLoRaTransmissionAttempt = 0  ;

static const u1_t PROGMEM APPEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);  }

static const u1_t PROGMEM DEVEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);  }

static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getDevKey (u1_t* buf) { memcpy_P(buf, APPKEY, 16); }
```

```
void setup()
{
  pinMode(5, OUTPUT);
  pinMode(7, OUTPUT);

  digitalWrite(5, HIGH); // Disable LoRa
  digitalWrite(7, LOW) ; // Enable RFM69

  memset(uplink, 0, UPLINK_LENGTH);
  memset(dnlink, 0, DNLINK_LENGTH);

  Serial.begin(9600);
  while (!Serial)   ;

  Serial.println(F("Starting"));

  if ( ! manager.init() )
  {
    driver.setFrequency( 434.0 ) ;
    driver.setTxPower( 17, true );
  }

  digitalWrite(7, HIGH);
  digitalWrite(5, LOW) ;

  os_init();

  LMIC_reset();
  LMIC_startJoining();

  LMIC_EXIT_FLAG = false;

  do
  {
    os_runloop_once();

  } while ( !LMIC_EXIT_FLAG );
}
```

```
void loop() {
  if ( millis() - lastLoRaTransmissionTimestamp ⩾ (LORA_TRANSMISSION_INTERVAL *
1000) )
  {
    if ( strlen(uplink) > 0 ) {
      digitalWrite(7, HIGH);
      digitalWrite(5, LOW) ;
      LMIC_EXIT_FLAG = false;
      do_send();
      do {
        os_runloop_once();
      } while ( !LMIC_EXIT_FLAG );
      memset(uplink, 0, UPLINK_LENGTH);
      lastLoRaTransmissionTimestamp = millis();
    }
  }
  digitalWrite(5, HIGH);
  digitalWrite(7, LOW) ;
  if (strlen(dnlink) > 0) {
    uint8_t addr_sep = dn.indexOf('_');
    uint8_t dest = (uint8_t) dn.substring(0, addr_sep).toInt() ;
    char*    msg = (char*)   dn.substring(addr_sep + 1).c_str();
    manager.sendtoWait(msg, strlen(buf), dest);
  }
  lastMeshTransmissionTimestamp = millis();
  while(millis() < lastMeshTransmissionTimestamp + MESH_MINIMUM_ON_TIME * 1000) {
    if (manager.available()) {
      uint8_t len = sizeof(buf);
      uint8_t from;

      if (manager.recvfrom(buf, &len, &from)) {
        String message;
        message.concat(from);
        message.concat("_") ;
        message.concat(buf);
        message.concat("-");

        if (strlen(uplink) + message.length() < UPLINK_LENGTH) {
          strcat(uplink, message.c_str());
        }
      }
    }
    delay(1);
  }
}

void do_send() {
  if (LMIC.opmode & OP_TXRXPEND)
  {
    LMIC_clrTxData();
  }
  LMIC_setTxData2(1, uplink, strlen(uplink) - 1, 0);
}
```

Π.2 Κώδικας στον Κόμβο (SVM40)

```
#include <RHMesh.h>
#include <RH_RF69.h>
#include <SensirionI2CSvm40.h>
#include <Wire.h>
#include <SPI.h>
#define CLIENT_ADDRESS 1
#define SERVER_ADDRESS 0

SensirionI2CSvm40 SVM40;
RH_RF69 driver(7, 3);
RHMesh manager(driver, CLIENT_ADDRESS);
String data[3];
char buf[RH_RF69_MAX_MESSAGE_LEN];

void setup() {
  Wire.begin();
  SVM40.begin(Wire)   ;
  SVM40.deviceReset() ;
  SVM40.startContinuousMeasurement() ;
  if (!manager.init());
  driver.setFrequency(434.0) ;
  driver.setTxPower(17, true);
}
void loop() {
  if (manager.available()) {

    uint8_t len = sizeof(buf);
    uint8_t from;

    if (manager.recvfrom(buf, &len, &from)) {
      String message = String(buf);
      digitalWrite(13, ( message.endsWith(“1”) ? HIGH : LOW ));
    }
  }
  int16_t v;
  int16_t h;
  int16_t t;

  SVM40.readMeasuredValuesAsIntegers(v, h, t);

  data[0] = "V_" + String((double) v / 10.0) ;
  data[1] = "H_" + String((double) h / 100.0);
  data[2] = "T_" + String((double) t / 200.0);


  for (uint8_t i = 0; i < 3; i++) {
    data[i].toCharArray(buf, data[i].length());
    manager.sendtoWait(buf, strlen(buf), SERVER_ADDRESS;
  }
}
```

Π.3 Κώδικας στον Κόμβο (SCD30)

```
#include <RHMesh.h>
#include <RH_RF69.h>
#include "SparkFun_SCD30_Arduino_Library.h"
SCD30 airSensor;
#include <Wire.h>
#include <SPI.h>

#define CLIENT_ADDRESS 1
#define SERVER_ADDRESS 0

RH_RF69 driver(7, 3);

RHMesh manager(driver, CLIENT_ADDRESS);

String data;
char buf[RH_RF69_MAX_MESSAGE_LEN];

void setup()
{
  Wire.begin();
  airSensor.begin();

  if (!manager.init()) {
    while (1) ;
  }

  driver.setFrequency(434.0) ;
  driver.setTxPower(17, true);
}

void loop() {
  if (manager.available()) {

    uint8_t len = sizeof(buf);
    uint8_t from;

    if (manager.recvfrom(buf, &len, &from)) {
      String message = String(buf);

      digitalWrite(13, ( message.endsWith(“1”) ? HIGH : LOW ));
    }
  }

  int16_t c = airSensor.getCO2();

  data = "C_" + String((double));
  data.toCharArray(buf, data[i].length());
  manager.sendtoWait(buf, strlen(buf), SERVER_ADDRESS);
}
```

Π.4 Κώδικας στον Κόμβο (SPS30)

```
#include <RHMesh.h>
#include <RH_RF69.h>
#include <sps30.h>
#include <Wire.h>
#include <SPI.h>

#define CLIENT_ADDRESS 1
#define SERVER_ADDRESS 0

RH_RF69 driver(7, 3);

RHMesh manager(driver, CLIENT_ADDRESS);

String data[2];
char buf[RH_RF69_MAX_MESSAGE_LEN];

void setup() {
  Wire.begin();
  sensirion_i2c_init();
  sps30_start_measurement();
  if (!manager.init()) {
    while (1) ;
  }
  driver.setFrequency(434.0) ;
  driver.setTxPower(17, true);
}

void loop() {
  if (manager.available())
  {
    uint8_t len = sizeof(buf);
    uint8_t from;

    if (manager.recvfrom(buf, &len, &from))
    {
      String message = String(buf);

      digitalWrite(13, ( message.endsWith("1") ? HIGH : LOW ));
    }
  }

  struct sps30_measurement m;
  sps30_read_measurement(&m);

  data[0] = "P_" + String((double) m.mc_2p5);
  data[1] = "S_" + String((double) m.nc_2p5);

  for (uint8_t i = 0; i < 2; i++) {
    data[i].toCharArray(buf, data[i].length());
    manager.sendtoWait(buf, strlen(buf), SERVER_ADDRESS);
  }
}
```

[1] **Τηλεπικοινωνίες και Δίκτυα Υπολογιστών**
Άρης Αλεξόπουλος - Γιώργος Λαγογιάννης, ISBN 9789609335423

[2] **A Brief History of LoRa**
[https://blog.semtech.com/a-brief-history-of-lora]-three-inventors-share-their-personal-story-at-the-things-conference]

[3,16,17] **What is LoRaWAN - A technical overview of LoRa and LoRaWAN**
[https://lora-alliance.org/resource_hub/what-is-lorawan]

[4] **Frequency Plans by Country**
[https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/]

[5] **IEEE Computer Society (August 31, 2007)**
[IEEE Standard 802.15.4a-2007]

[6] **On the Utility of Chirp Modulation for Digital Signaling**
[IEEE Transactions on Communications (Volume: 21, Issue: 6, Jun 1973)]

[7-14] **Experiencing LoRa Network Establishment on a Smart Energy Campus**
Dong-Hoon Kim, Eun-Kyu Lee and Jibum Kim
College of Information Technology, Incheon Nat'l University, Korea

[15] **What are LoRa and LoRaWAN?**
[https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/]

[18] **MCCI Catena LMIC**
[https://github.com/mcci-catena/arduino-lmic]