ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

LORA MANAGEMENT PLATFORM

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχαήλ Γ. Ποταμιάς

Επίβλεψη : Ευστάθιος Δ. Συκάς, Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

LORA MANAGEMENT PLATFORM

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχαήλ Γ. Ποταμιάς

Επίβλεψη : Ευστάθιος Δ. Συκκάς, Καθηγητής Ε.Μ.Π

Εγκρίθηκε από τη τριμελή εξεταστική επιτροπή την ..... Οκτωβρίου 2017

.....	.....	.....
Ευστάθιος Συκάς	Γεώργιος Στασινόπουλος	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. καθηγήτρια Ε.Μ.Π.

Αθήνα, Νοέμβριος 2017

.....

Μιχαήλ Γ. Ποταμιάς
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά το ακαδημαϊκό έτος 2016-2017 υπό την επίβλεψη του κ. Ευστάθιου Συκά, καθηγητή της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ, στον οποίο οφείλω ιδιαίτερες ευχαριστίες για την ανάθεση της, δίνοντας μου την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον και επίκαιρο θέμα.

Επιπρόσθετα, θερμές ευχαριστίες οφείλω στον διδάκτορα Γεώργιο Λυμπερόπουλο, όχι μόνο για την υψηλού επιπέδου καθοδήγηση και το υλικό που μου παρείχε, αλλά και για τον χώρο που μου διέθεσε στα γραφεία του OTE-ACADEMY, δίνοντας μου την ευκαιρία να εργαστώ για την εκπόνηση της εργασίας σε έναν επαγγελματικό χώρο.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου καθώς και όλους μου τους φίλους εντός και εκτός της σχολής, οι οποίοι μου συμπαραστάθηκαν και με συντρόφευσαν στα φοιτητικά μου χρόνια. Ιδιαίτερη μνεία αξίζει στους συναδέλφους μου Αλέξανδρο, Γιώργο, Νίκο, Δημήτρη, Μιχάλη και Ιάκωβο με τους οποίους μοιραστήκαμε ένα τόσο παραγωγικό διάστημα της ζωής μας και βιώσαμε παρέα το ταξίδι.

ΠΡΟΛΟΓΟΣ

Ο όρος "Internet of Things" (ή αλλιώς Διαδίκτυο των Πραγμάτων) επινοήθηκε στα τέλη της δεκαετίας του 1990 από τον επιχειρηματία Kevin Ashton, ο οποίος ήταν μέρος μιας ομάδας που ανακάλυψε τον τρόπο σύνδεσης αντικειμένων με το διαδίκτυο μέσω μιας ετικέτας RFID. Το IoT είναι το τεχνολογικό πεδίο που αναφέρεται στη δια-δικτύωση φυσικών συσκευών, οχημάτων, κτιρίων και άλλων αντικειμένων με ενσωματωμένα ηλεκτρονικά συστήματα, λογισμικό και αισθητήρες. Οι IoT τεχνολογίες αναπτύσσονται ραγδαία τα τελευταία χρόνια και εφαρμόζονται σε μία μεγάλη γκάμα εφαρμογών, από οικιακές συσκευές και κτίρια μέχρι βιομηχανικές εγκαταστάσεις και αγροκτήματα.

Τα υπάρχοντα ευρείας χρήσης ασύρματα τηλεπικοινωνιακά δίκτυα δεν έχουν σχεδιαστεί για να εξυπηρετούν περιοχές που βρίσκονται μακριά από σταθμούς, βάσεις και υποδομές. Δεν έχουν σχεδιαστεί για μεταφορές μικρού όγκου δεδομένων και αναγκάζουν τις συσκευές να καταναλώνουν μεγάλες ποσότητες ενέργειας. Οι παραπάνω παράγοντες είναι πολύ σημαντικοί για τις συσκευές που ανήκουν στο IoT. Τη λύση για την αντιμετώπιση των παραπάνω προβλημάτων δίνουν τα δίκτυα χαμηλής ισχύος-ευρείας περιοχής (Low Power Wide Area Networks) LPWAN. Πρόκειται για τεχνολογία που εξασφαλίζει τη διασύνδεση συσκευών έχοντας ως βασική προτεραιότητα την οικονομία (χαμηλή κατανάλωση ισχύος) και την ευρεία κάλυψη. Πολλές εταιρίες έχουν αναπτύξει τα τελευταία χρόνια τη δική τους LPWAN τεχνολογία, η Sigfox, η LoRa Alliance, η Igenu και η Weightless είναι μερικές απ' αυτές.

Το LoRa (LoRa Alliance, https://LoRa-alliance.org) είναι ένα πρωτόκολλο LPWAN και αποτελεί το αντικείμενο μελέτης της παρούσας εργασίας. Η μεγάλη εμβέλεια και η οικονομία των κόμβων του δικτύου, καθιστούν τη συγκεκριμένη τεχνολογία ιδανική για εφαρμογές που αφορούν αστικές, βιομηχανικές και αγροτικές υποδομές. Η οικοδόμηση εύρωστων δικτύων δεδομένων χαμηλού κόστους για οικίες, αγροκτήματα, κοινότητες, πόλεις αλλά και ολόκληρες χώρες, μπορεί να γίνει με πολύ χαμηλό οικονομικό κόστος, καθώς ένα LoRa Gateway κοστίζει λιγότερο από έναν ηλεκτρονικό υπολογιστή.

Η παρούσα εργασία χρησιμοποιεί τη τεχνολογία LoRa για την οικοδόμηση ενός τέτοιου δικτύου χαμηλού κόστους. Στα πλαίσια της, γίνεται μελέτη των χαρακτηριστικών της LoRa τεχνολογίας και περιγράφεται η ανάπτυξη συστήματος (σε επίπεδο software) που πραγματοποιήθηκε με στόχο:

- Την αμφίδρομη επικοινωνία του με τους κόμβους
- Την αποθήκευση των δεδομένων και τη γραφική απεικόνισή τους.
- Την ανάπτυξη πλατφόρμας διαχείρισής του.

Λέξεις κλειδιά: LPWAN, LoRa, End-devices, LoRa Gateway, Backend Infrastructure, InfluxDB, Grafana, Web Interface

ABSTRACT

The term “Internet of Things” was invented by a businessman named Kevin Ashton who participated in a team that invented the way of connecting objects to the internet via a RFID tag. Lot is the field of technology that pertains to the networking of physical appliances, vehicles, buildings and objects with embedded electronic systems, software and sensors. LoT technologies are expanding rapidly in the recent years and they are applied in a wide range of applications ranging from domestic appliances and buildings to industrial installations and farmer fields.

The existing wireless telecommunication networks that are of widespread deployment have neither been designed to serve regions located far from stations, bases and infrastructure nor for the transfer of low volume data, hence resulting in appliances consuming large amounts of energy. The above-mentioned factors are significant with respect to the Lot appliances. The so called “Low Power Wide Area Network” provides for a solution in tackling the aforementioned issues. This is a technology ensuring the interface of appliances by basically prioritizing both the economy (low power consumption mode) and a broad coverage. Many companies have developed in recent years their own LPWAN technology such as Sigfox, LoRa Alliance, Igenu and Weightless, to mention but a few.

LoRa (LoRa Alliance, https://LoRa-alliance.org) is one of LPWAN protocols and is the subject matter of thisessay.The broad range and the economy of the nodes of the network make this technology ideal for the urban, industrial and rural related infrastructure appliances.Creating sound and low cost data networks not only for residential, rural, community and urban use but also for entire countries can be done in a cost- effective way since LoRa Gateway is less costly than a computer machine. This essay uses the LoRa technology for the construction of such a low-cost network. In this framework, a study is being carried out as regards the characteristics of LoRa technology and a description of the development of the system (on a software level) is being given in order to present:

♦ The two-way communication of this system with the nodes

♦ Storage of data and their graphical presentation

♦ The development of the system s management platform

Keywords: LPWAN, LoRa, End-devices, LoRa Gateway, Backend Infrastructure, InfluxDB, Grafana, Web Interface

Πίνακας Περιεχομένων

Πίνακας Διαγραμμάτων

Πίνακας Εικόνων

1. ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική έχει ως αντικείμενο την ανάπτυξη ενός Internet of Things απ' άκρη σ' άκρη συστήματος χρησιμοποιώντας τη τεχνολογία LoRa. Η ανωτέρω διαδικασία υλοποιείται με την ενοποίηση και ολοκλήρωση των ακόλουθων υποσυστημάτων:

- End devices (Dragino LoRa shields και sodaq LoRaONE) με αισθητήρες για τη μέτρηση θερμοκρασίας, υγρασίας, κίνησης και GPS.
- LoRa Gateway για την επικοινωνία των μικροελεγκτών με το backend infrastructure.

- Backend infrastructure που χρησιμοποιείται για την αποκωδικοποίηση των μηνυμάτων, την δρομολόγησή τους στην κατάλληλη εφαρμογή και την αποθήκευση τους σε βάση δεδομένων
- Web interface μέσω του οποίου ο χρήστης έχει τη δυνατότητα να διαχειρίζεται τις εφαρμογές του και να παρακολουθεί τα δεδομένα από τους αισθητήρες

Συγκεκριμένα, οι μικροελεγκτές στέλνουν μηνύματα με τα δεδομένα που αντλούν από τους αισθητήρες. Το gateway λαμβάνει και προωθεί τα μηνύματα αυτά στο backend infrastructure, το οποίο με το κατάλληλο λογισμικό λαμβάνει, αποκωδικοποιεί και αποθηκεύει τα δεδομένα σε βάση δεδομένων. Τέλος αναπτύχθηκε web interface για τη διαχείριση των συσκευών και τη παρουσίαση των δεδομένων σε γραφικές παραστάσεις.

1.2 Οργάνωση της διπλωματικής

Η περιγραφή και ανάλυση της τεχνολογίας LoRa καθώς και του LoRaWAN πρωτοκόλλου γίνεται στο κεφάλαιο 2. Επιπλέον γίνεται αναφορά και σε άλλες LPWAN τεχνολογίες.

Στο 3ο κεφάλαιο εξηγείται η λειτουργία του LoRa-gateway, περιγράφεται η διαδικασία εγκατάστασης του λογισμικού καθώς και οι περαιτέρω ρυθμίσεις.

Το κεφάλαιο 4 αφορά το backend infrastructure του συστήματος. Επεξηγείται η λειτουργία όλων των επιμέρους στοιχείων καθώς και η μεταξύ του επικοινωνία, ενώ περιγράφεται η διαδικασία και τα βήματα που ακολουθήθηκε για το στήσιμο του συστήματος. Επιπροσθέτως παρουσιάζεται η time series database influxDB που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων και το εργαλείο grafana που χρησιμοποιήσαμε για τη γραφική απεικόνιση αυτών.

Το 5ο κεφάλαιο αφορά τα end devices του συστήματος και περιγράφεται η διαδικασία ρύθμισής τους σε hardware και software επίπεδο.

Στο κεφάλαιο 6 παρουσιάζεται το web interface που αναπτύχθηκε και επεξηγείται η δομή και η λειτουργικότητά του.

Στο 7ο κεφάλαιο γίνεται η σύνοψη και προτάσεις για μελλοντικές επεκτάσεις.

2. Η τεχνολογία LoRa

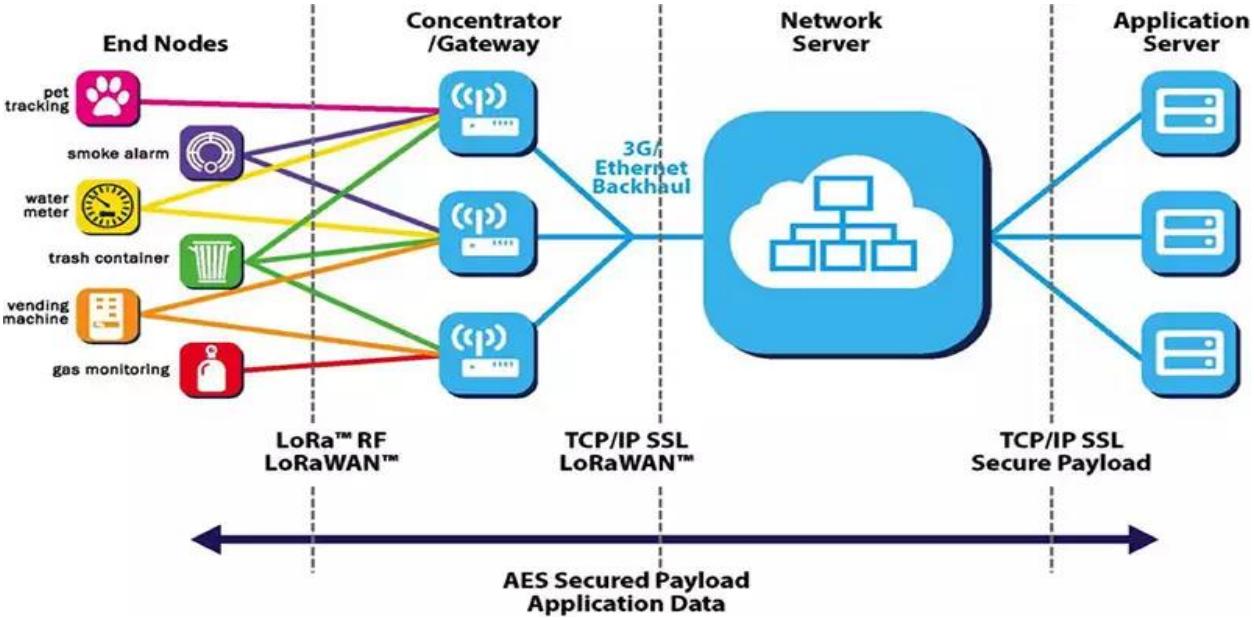
2.1 ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ LORA

Η τεχνολογία LoRa αναπτύχθηκε από την εταιρία Semtech και αποτελεί ένα νέο ασύρματο πρωτόκολλο σχεδιασμένο για επικοινωνία μεγάλης εμβέλειας και χαμηλής ισχύος.

Η συμμαχία LoRa (LoRa-Alliance) αποτελεί τον βασικό φορέα ανάπτυξης της συγκεκριμένης τεχνολογίας. Δημιουργήθηκε για να τυποποιήσει τα χαμηλής ισχύος και ευρείας περιοχής δίκτυα (Low Power Wide Area Networks) στα πλαίσια του Internet of Things.

Μεγάλες εταιρίες όπως η Cisco, actility, Microchip, IBM, STMicro, Orange mobile είναι μερικά από τα μέλη της συμμαχίας LoRa. Η συμμετοχή όλων αυτών των κολοσσών στην ανάπτυξη της LoRa τεχνολογίας όχι μόνο προμηνύει τη μεγάλη επέκταση και ανάπτυξη που θα έχει στο μέλλον αλλά εξασφαλίζει και τη διαλειτουργικότητα με άλλα μεγάλα δίκτυα.

Μία τοπολογία ενός LoRa δικτύου αποτελείται από End nodes, gateways, network server και application server όπως φαίνεται στο παρακάτω σχήμα:



ΔΙΑΓΡΑΜΜΑ 1: Ένα ολοκληρωμένο LoRa δίκτυο

Όπως φαίνεται στο σχήμα, το σενάριο επικοινωνίας έχει ως εξής:

Ο χρήστης δημιουργεί, μέσω του application server, εφαρμογές στις οποίες καταχωρεί τις συσκευές του (End Nodes). Τα End Nodes στέλνουν τα δεδομένα που αντλούν από τους αισθητήρες σε περισσότερα από ένα gateways που βρίσκονται εντός εμβέλειας. Τα gateways προωθούν τα μηνύματα προς τον network server. Ο network server εκτός από τη προώθηση των μηνυμάτων στις κατάλληλες εφαρμογές, αποκωδικοποιεί τα μηνύματα ώστε να γίνονται κατανοητά από τον χρήστη.

Αξίζει να σημειωθεί ότι κάθε LoRa Gateway έχει τη δυνατότητα να διαχειριστεί χιλιάδες κόμβους. Έτσι γίνεται σαφές ότι απαιτείται λιγότερη υποδομή άρα και μικρότερο κόστος για το χτίσιμο ενός μεγάλου και πολύπλοκου δικτύου σε σχέση με άλλα υπάρχοντα ασύρματα πρωτόκολλα.

Στον παρακάτω πίνακα παρουσιάζονται μερικά από τα βασικά χαρακτηριστικά του πρωτοκόλλου.

Προδιαγραφές	**LoRa Χαρακτηριστικά**
Εμβέλεια	2-5Km αστικό περιβάλλον 15Km μη αστικό περιβάλλον
Συχνότητα	ISM 868/915 MHz
Διαμόρφωση	Chirp Spread spectrum modulation (CSS)
Χωρητικότητα	Μεγάλη, καθώς να LoRa gateway μπορεί να δεχτεί χιλιάδες κόμβους
Μπαταρία	Μεγάλη διάρκεια ζωής μπαταρίας (μέχρι 10 χρόνια)

Είναι προφανές ότι η ασύρματη τεχνολογία LoRa, θα πρωταγωνιστήσει τα επόμενα χρόνια στην αγορά του Internet of Things. Η διασύνδεση συσκευών για τη δημιουργία έξυπνων πόλεων, βιομηχανικών και εμπορικών λύσεων θα επιτυγχάνεται μειώνοντας παράλληλα τους περιορισμούς από άλλες ασύρματες τεχνολογίες όπως η ισχύς και άλλα γενικά έξοδα.

2.2 LoRaWAN

Το LoRaWAN αποτελεί το επίπεδο δικτύου και διέπεται από τη συμμαχία LoRa. Στην ουσία με τον όρο LoRa αναφερόμαστε στο φυσικό στρώμα (το chip) ενώ με τον όρο LoRaWAN αναφερόμαστε στο MAC στρώμα (το λογισμικό που χρησιμοποιείται ώστε το chip να ενεργοποιήσει τη δικτύωση).

Το LoRaWAN σχεδιάστηκε για να βελτιστοποιήσει τη λειτουργία των LPWAN (Low Power Wide Area Networks) κυρίως ως προς τη διάρκεια ζωής της μπαταρίας, τη χωρητικότητα, το εύρος αλλά και το κόστος. Το βασικό πλεονέκτημα του LoRa βρίσκεται στην ικανότητα του να καλύπτει μεγάλο γεωγραφικό εύρος.

Το LoRa βρίσκεται στο φυσικό στρώμα ή στην ασύρματη διαμόρφωση που χρησιμοποιείται για τη δημιουργία μεγάλου εύρους επικοινωνιακής ζεύξης. Παρότι πολλά ασύρματα συστήματα χρησιμοποιούν Frequency shifting Keying (FSK) διαμόρφωση για την επίτευξη χαμηλής ισχύος, το LoRa βασίζεται σε Chirp Spread Spectrum διαμόρφωση, η οποία διατηρεί τα χαρακτηριστικά της FSK ως προς τη χαμηλή ισχύ, αλλά επιπλέον έχει το πλεονέκτημα σημαντικής αύξησης του επικοινωνιακού εύρους. Για πολλά χρόνια αυτού του είδους διαμόρφωση χρησιμοποιήθηκε για στρατιωτικές και διαστημικές επικοινωνίες κι όχι σε εμπορικές εφαρμογές. Το LoRa αποτελεί έτσι τη πρώτη εμπορική εφαρμογή που χρησιμοποιεί chirp spread spectrum modulation σε τόσο χαμηλό κόστος.

Το βασικό πλεονέκτημα του LoRa όπως αναφέρθηκε παραπάνω, είναι η μεγάλη του εμβέλεια. Φυσικά το χαρακτηριστικό αυτό αλλάζει από περιοχή σε περιοχή καθώς εξαρτάται από το περιβάλλον. Εκτιμάται ότι σε αστική περιοχή η εμβέλεια φτάνει έως 2-3 χιλιόμετρα, ενώ σε μη-αστικό περιβάλλον η εμβέλεια μπορεί να φτάσει έως 15 χιλιόμετρα.

Το LoRa ανήκει στη κατηγορία Low Power Wide Area δικτύων. Τα LPWAN networks απευθύνονται κυρίως σε εφαρμογές που απαιτούν μεγάλη διάρκεια μπαταρίας, μικρή ποσότητα δεδομένων και μεγάλη απόσταση. Άλλες μεγάλες κατηγορίες IoT τεχνολογίας είναι το Wifi και BTLE που είναι ευρέως διαδεδομένες σε εφαρμογές προσωπικών συσκευών, και η κυψελοειδής τεχνολογία (Cellular Technology) που είναι ιδανική για Εφαρμογές που χρειάζονται υψηλή απόδοση δεδομένων και έχουν πηγές ενέργειας.

Στον παρακάτω πίνακα παρουσιάζονται ορισμένα βασικά χαρακτηριστικά των τριών αυτών τεχνολογιών.

	Local Area Network Short Range Communication	**Low Power Wide Area** (LPWAN) Internet of Things	**Cellular Network** Traditional M2M
	40%	**45%**	**15%**
☺	Well established standards In building	Low power consumption Low cost Positioning	Existing coverage High data rate
☹	Battery Live Provisioning Network cost & dependencies	High data rate Emerging standards	Autonomy Total cost of ownership
	Bluetooth 4.0 · Wi Fi	LoRa	GSM · 3G+ / H+ · 4G

ΔΙΑΓΡΑΜΜΑ 2: Χαρακτηριστικά LAN, LPWAN & Cellular δικτύων

Το LoRaWAN καθορίζει το πρωτόκολλο επικοινωνίας και την αρχιτεκτονική του συστήματος για το δίκτυο, ενώ το φυσικό στρώμα του LoRa ενεργοποιεί τη ζεύξη επικοινωνίας μεγάλης εμβέλειας. Το πρωτόκολλο και η αρχιτεκτονική του δικτύου παίζουν τον πιο σημαντικό ρόλο ως προς τον καθορισμό της διάρκειας ζωής της μπαταρίας του κάθε κόμβου, τη χωρητικότητα του δικτύου, τη ποιότητα των υπηρεσιών και την ασφάλεια των εφαρμογών που προσφέρει το δίκτυο.

2.3 Τα Χαρακτηριστικά της τεχνολογίας LORA

Στην ενότητα αυτή, γίνει αναφορά στα βασικότερα χαρακτηριστικά που καθιστύν το LoRaWAN δίκτυο τόσο ενδιαφέρον

Συγκεκριμένα θα αναφερθούμε:

- Αρχιτεκτονική του δικτύου
- Η μεγάλη διάρκεια ζωής της μπαταρίας
- Η μεγάλη χωρητικότητα
- Ο διαχωρισμός των συσκευών σε κλάσεις
- Η ασφάλεια που παρέχεται

2.3.1 Αρχιτεκτονική Δικτύου

Πολλά από τα υπάρχοντα αναπτυγμένα δίκτυα χρησιμοποιούν αρχιτεκτονική πλέγματος δικτύου. Σε μία τέτοια αρχιτεκτονική, οι μεμονωμένοι τελικοί κόμβοι προωθούν τις πληροφορίες άλλων κόμβων. Αυτή η λογική έχει τα ακόλουθα αποτελέσματα:

- Αύξηση του επικοινωνιακού εύρους του δικτύου
- Αύξηση της πολυπλοκότητας του δικτύου.
- Μείωση του χρόνου ζωής της μπαταρίας του κάθε κόμβου

Επομένως για ένα δίκτυο στο οποίο οι τελικοί κόμβοι στέλνουν αενάως δεδομένα, η περιορισμένη διάρκεια μπαταρίας αποτελεί μεγάλο μειονέκτημα. Σε αντίθεση με την

αρχιτεκτονική πλέγματος δικτύου, η αρχιτεκτονική αστέρα εκτός από το μεγάλο επικοινωνιακό εύρος, εξασφαλίζει και μεγάλη διάρκεια ζωής της μπαταρίας.

Σε ένα LoRaWAN δίκτυο ο κάθε κόμβος δεν είναι συνδεδεμένος με ένα συγκεκριμένο gateway. Αντίθετα, είναι δυνατόν πολλά gateways να λάβουν τα ίδια δεδομένα από έναν κόμβο. Όταν κάποιο LoRa gateway λάβει κάποιο μήνυμα, το προωθεί μέσω κάποιου backhaul (Ethernet, wifi, satellite ή celular) στον server. Ο server αποτελεί τη καρδιά του συστήματος, καθώς είναι υπεύθυνος για τη διαχείριση του δικτύου. Η αναλυτική περιγραφή της λειτουργίας του server γίνεται στο 4ο κεφάλαιο, ωστόσο, κάποιες ενδεικτικές αρμοδιότητες του είναι οι παρακάτω:

- αποκωδικοποιεί τα πακέτα
- διασφαλίζει την ασφάλεια του δικτύου
- οργανώνει όλους τους κόμβους σε εφαρμογές, ώστε να τους διαχειρίζεται με ευκολία ο χρήστης.

2.3.2 Διάρκεια μπαταρίας

Οι κόμβοι σε ένα LoRaWAN δίκτυο είναι ασύγχρονοι καθώς επικοινωνούν όταν έχουν δεδομένα να στείλουν είτε πρόκειται για προγραμματιζόμενη επικοινωνία είτε όχι. Αυτός ο τύπος πρωτοκόλλου είναι γνωστός και ως ALOHA. Σε ένα πλεγματικό (mesh) ή κυτταρικό (celular) δίκτυο οι κόμβοι θα έπρεπε πολύ συχνά να ενεργοποιηθούν για να συγχρονιστούν με το υπόλοιπο δίκτυο, προκειμένου να προωθήσουν μηνύματα άλλων κόμβων. Αυτή η συχνή ενεργοποίηση των κόμβων θα επέφερε σημαντική μείωση στον χρόνο ζωής της μπαταρίας συγκριτικά με τους κόμβους στο LoRaWAN. Σε πρόσφατη σύγκριση που πραγματοποίησε το GSMA μεταξύ των LPWAN δικτύων, το LoRaWAN εμφανίζεται να έχει τρεις με τέσσερις φορές μεγαλύτερη διάρκεια ζωής μπαταρίας των κόμβων.

2.3.3 Χωρητικότητα δικτύου

Οι βασικοί παράγοντες που επηρεάζουν την χωρητικότητα ενός LoRa δικτύου είναι:

- Το data rate
- Το μέγεθος του μηνύματος
- Η συχνότητα εκπομπής κάθε κόμβου

Σε ένα δίκτυο τοπολογίας αστέρα, με μεγάλη εμβέλεια η χωρητικότητα του δικτύου είναι πολύ υψηλή και ένα LoRa gateway πρέπει να έχει την ικανότητα να λάβει μηνύματα από μεγάλο αριθμό κόμβων.Αυτό επιτυγχάνεται εφαρμόζοντας προσαρμοστικό data rate, αλλά και multi-channel δέκτες. Συγκεκριμένα, τα σήματα λόγω της Chirp Spread Spectrum (CSS) διαμόρφωσης μεταδίδονται σχεδόν ορθογωνικά μεταξύ τους, αν χρησιμοποιηθεί κάποιος διαφορετικός συντελεστής μετάδοσης. Έτσι όσο αλλάζει ο συντελεστής μετάδοσης, αλλάζει το data rate, και το gateway λαμβάνει όλα τα σήματα, καθώς έχει τη δυνατότητα να λάβει πολλαπλά σήματα με διαφορετικό data rate μεταξύ τους, στο ίδιο channel. Κάθε κόμβος έχει τη δυνατότητα να προσαρμόσει το data rate σε υψηλότερα επίπεδα, μειώνοντας τον χρόνο διάδοσης του σήματος, δίνοντας έτσι περισσότερο χώρο σε άλλους κόμβους να εκπέμψουν. Επιπλέον, το προσαρμοστικό data rate βελτιστοποιεί και τη διάρκεια ζωής της μπαταρίας.

2.3.4 Κλάσεις Συσκευών

Κάθε συσκευή (End-device) ενός δικτύου εξυπηρετεί διαφορετικές εφαρμογές, και έχει διαφορετικές προδιαγραφές. Το LoRaWAN, προκειμένου να βελτιστοποιήσει τις εφαρμογές αυτές, εφαρμόζει διαφορετικές κλάσεις στους κόμβους. Οι κλάσεις είναι τρείς A,B και C, και προκύπτουν από το tradeoff μεταξύ της καθυστέρησης για downlink επικοινωνία και της διάρκειας ζωής της μπαταρίας (battery lifetime).

Κλάση Α

Οι συσκευές Class A επιτρέπουν την αμφίδρομη επικοινωνία κατά την οποία κάθε μετάδοση ακολουθούν 2 μικρές λήψεις. Οι κόμβοι αυτής της κλάσης καταναλώνουν την χαμηλότερη ισχύ σε σχέση με τις συσκευές των άλλων κλάσεων, ωστόσο έχουν περιορισμό ως προς την downlink επικοινωνία.

Κλάση Β

Οι συσκευές Class B εκτός από τις λειτουργίες των συσκευών Class A, επιτρέπουν τη λήψη σε προγραμματισμένες στιγμές. Αυτό επιτυγχάνεται, λαμβάνοντας ένα σήμα συγχρονισμού από το gateway. Έτσι ο server γνωρίζει ποιες χρονικές στιγμές η συσκευή είναι διαθέσιμη να δεχτεί κάποιο μήνυμα.

Κλάση C

Οι συσκευές Class C έχουν τη δυνατότητα να δέχονται μηνύματα οποιαδήποτε στιγμή πλην των στιγμών μετάδοσης.

2.3.5 Ασφάλεια

Η ανάγκη για ασφάλεια και κρυπτογράφηση των δεδομένων είναι πλέον επιτακτική και στις IoT εφαρμογές. Το LoRaWAN εφαρμόζει δύο στρώματα ασφαλείας

1) Ασφάλεια δικτύου: εξασφαλίζει την αυθεντικότητα των κόμβων μέσα στο δίκτυο

2) Ασφάλεια εφαρμογής: εξασφαλίζει ότι ο χειριστής δικτύου δεν έχει πρόσβαση στα δεδομένα εφαρμογών του χρήστη.

Χρησιμοποιείται ο αλγόριθμος κρυπτογράφησης AES με ανταλλαγή κλειδιών χρησιμοποιώντας έναν IEEE EUI64 αναγνωριστικό.

2.4 LPWAN ΤΕΧΝΟΛΟΓΙΕΣ

Η τεχνολογία LoRa αποτελεί μία μόλις επιλογή σε μία αγορά που τα τελευταία χρόνια αναπτύσσεται με ταχείς ρυθμούς. Πολλές είναι οι εταιρίες του χώρου που δίνουν έμφαση σε εφαρμογές χαμηλής ισχύος, αναπτύσσοντας τη δική τους LPWAN τεχνολογία. Οι σημαντικότερες LPWAN τεχνολογίες καθώς και τα βασικά τους χαρακτηριστικά θα παρουσιαστούν συνοπτικά παρακάτω. Συγκεκριμένα θα αναφερθούμε στις τεχνολογίες Sigfox, LoRa, Rpma και Weightless.

2.4.1 SIGFOX :

Η γαλλική εταιρία SIGFOX ιδρύθηκε το 2009 και είναι ίσως η πιο γνωστή εταιρία στο LPWAN χώρο λόγω κυρίως του επιτυχημένου marketing που έχει αναπτύξει στην Ευρώπη. Βασικοί της προμηθευτές είναι εταιρίες κολοσσοί όπως η Texas Instruments, Silicon Labs και Axom.

Βασικά χαρακτηριστικά:

- Μεταδίδει σε ζώνη συχνοτήτων 868 MHz (EUROPE) και 902 MHz (US)
- Εύρος ζώνης 192 KHz .
- Data Rate για uplink επικοινωνία 100 bps-140 μηνύματα/μέρα.
- Υποστηρίζει αμφίδρομη επικοινωνία (uplink, downlink), με downlink data rate μέχρι 4 μηνύματα των 8 bytes/ημέρα.
- Μέγεθος μηνύματος μέχρι 12 bytes.
- Εμβέλεια 3-10 Km σε αστικικό περιβάλλον, 30-50 Km σε μη αστικό περιβάλλον.
- Συσκευές/gayeway μέχρι 1,000,000.
- Ισχύς μετάδοσης 10μW-100mW
- Διάρκεια ζωής μπαταρίας: μέχρι και 10 χρόνια.

2.4.2 LORA

Το LoRa Alliance είναι μία ανοιχτή μη κερδοσκοπική ένωση που δημιουργήθηκε με σκοπό τη προώθηση του LPWAN οικοσυστήματος. Μέχρι σήμερα απαριθμεί περισσότερα από 400 μέλη - εταιρίες που εδρεύουν στη πλειοψηφία τους στη Βόρεια Αμερική, την Ευρώπη, την Αφρική και την Ασία, ενώ τα ιδρυτικά μέλη της συμμαχίας είναι η IBM, η Microchip, η Cisco, η Semtech, η Bouygues, η Telecom, η Singtel, η KPN, η Swisscom, η Fastnet και η Belgacom.

Το LoRaWAN αποτελεί το επίπεδο δικτύου και διέπεται από τη συμμαχία LoRa. Στην ουσία με τον όρο LoRa αναφερόμαστε στο φυσικό στρώμα (το chip) ενώ με τον όρο LoRaWAN αναφερόμαστε στο MAC στρώμα (το λογισμικό που χρησιμοποιείται ώστε το chip να ενεργοποιήσει τη δικτύωση).

Η λειτουργικότητα του LoRa είναι παρόμοια με αυτήν της Sigfox και αφορά πρωτίστως την uplink επικοινωνία. Ωστόσο αντί να χρησιμοποιεί στενή ζώνη μετάδοσης, εξαπλώνει τη πληροφορία σε διαφορετικά κανάλια συχνότητας και σε διαφορετικά data rates χρησιμοποιώντας κωδικοποιημένα μηνύματα. Με τον τρόπο αυτό τα μηνύματα είναι απίθανο να συγκρουστούν μεταξύ τους, αυξάνοντας έτσι την ικανότητα του gateway.

Βασικά χαρακτηριστικά

- Μεταδίδει σε ζώνη συχνοτήτων 868 MHz (EUROPE) και 915 MHz (US)
- Εύρος ζώνης 192 KHz
- Εμβέλεια 3-5 Km σε αστικό περιβάλλον, 15 Km σε μη αστικό περιβάλλον.
- Υπάρχουν τρεις κλάσεις επικοινωνίας (class A, class B, class C), όλες υποστηρίζουν αμφίδρομη επικοινωνία.
- Data rate που ρυθμίζεται από τον χρήστη (μέχρι 50kbps)
- Μέγεθος πακέτου που ρυθμίζεται από τον χρήστη
- Συσκευές/gayeway μέχρι 1,000,000
- Ισχύς μετάδοσης 14 dBm
- Διάρκεια ζωής μπαταρίας: μέχρι και 10 χρόνια.

2.4.3 RPMA

Τα αρχικά της RPMA προκύπτουν από "Random Phase Multiple Acces" (Πολλαπλή πρόσβαση τυχαίας φάσης) και είναι τεχνολογία που αναπτύχθηκε από την εταιρία Igenu. Λόγω της αρχιτεκτονικής της, η RPMA έχει μεγαλύτερη ικανότητα για uplink και downlink επικοινωνία συγκριτικά με τις άλλες τεχνολογίες στον χώρο των LPWAN. Λειτουργεί σε φάσμα 2.4 GHz το οποίο είναι παγκοσμίως διαθέσιμο (χρησιμοποιείται για WiFi και Bluetuth), αυτό σημαίνει ότι δεν υπάρχουν αλλαγές στην αρχιτεκτονική ανά περιοχή όπως συμβαίνει με Sigfox και LoRa.

Βασικά χαρακτηριστικά

- Μεταδίδει σε ζώνη συχνοτήτων 2.4 GHz.
- Εύρος ζώνης 1 MHz
- Data Rate για uplink επικοινωνία 624 Kbps.
- Εμβέλεια 1-3 Km σε αστικό περιβάλλον, 5-10 Km σε μη αστικό περιβάλλον.
- Υποστηρίζει αμφίδρομη επικοινωνία με ικανοποιητικό data rate στα 156 kbps.
- Μέγεθος πακέτου μέχρι 10 kbytes
- Συσκευές/gayeway μέχρι 500,000
- Ισχύς μετάδοσης 20 dBm
- Διάρκεια ζωής μπαταρίας: μέχρι και 10 χρόνια.

2.4.4 WEIGHTLESS

Weightless SIG (special interest group) ιδρύθηκε το 2008 με αποστολή τη τυποποίηση των LPWANs τεχνολογιών. Ο όμιλος Weightless αποτελείται από πέντε μέλη, την Accenture, την ARM, τη M2COMM, τη Sony-Europe και τη Telensa. Ουσιαστικά αποτελεί το μοναδικό πραγματικά ανοιχτό πρότυπο που λειτουργεί με sub-1GHz φάσμα. Υπάρχουν τρεις εκδόσεις του Weightless που εξυπηρετούν διαφορετικούς σκοπούς. Η Weghtless N και P είναι οι πιο διαδεδομένες καθώς η τρίτη έκδοση, Weightless-W, υστερεί ως προς τη διάρκεια ζωής της μπαταρίας.

2.4.4.1 Weightless-N

Έχει αρκετές ομοιότητες με τη Sigfox όσο αφορά τη λειτουργικότητα, ωστόσο υπερτερεί ως προς τη MAC επιπέδου υλοποίηση καθώς χρησιμοποιεί προηγμένες τεχνικές από-διαμόρφωσης που δίνουν τη δυνατότητα στο δίκτυο να συνυπάρχει με άλλες ράδιο-τεχνολογίες χωρίς επιπλέον θόρυβο. Η έκδοση αυτή χρησιμοποιείται ως επί το πλείστων σε εφαρμογές που αφορούν τη μέτρηση θερμοκρασίας, τη μέτρηση επιπέδου δεξαμενών και άλλες τέτοιες μετρήσεις.

Βασικά χαρακτηριστικά

- Μεταδιδει σε ζώνη συχνοτήτων sub 1 GHz.
- Εύρος ζώνης 200 Hz
- Data Rate για uplink επικοινωνία 100 bps.
- Εμβέλεια που μπορεί να φτάσει μέχρι και 3 km.
- Δεν υποστηρίζει αμφίδρομη επικοινωνία, μόνο uplink.
- Μέγεθος πακέτου μέχρι 20 bytes.
- Απεριόριστες Συσκευές/gayeway
- Ισχύς μετάδοσης 17 dBm
- Διάρκεια ζωής μπαταρίας: μέχρι και 10 χρόνια.

2.4.4.2 Weightless-P

Αυτή η έκδοση χρησιμοποιεί FDMA+TDMA διαμόρφωση σε στενή ζώνη 12.5 KHz (υψηλότερη από Sigfox αλλά χαμηλότερη από LoRa). Χρησιμοποιεί προσαρμοστικό data rate (200 bps - 100 kbps). Η χρησιμοποίηση αυτής της έκδοσης συνιστάται σε πιο πολύπλοκες και εκλεπτυσμένες υλοποιήσεις στις οποίες ο έλεγχος των uplink ή downlink δεδομένων είναι σημαντικός. Σημειώνεται ότι η εμπορική ανάπτυξη τέτοιου εξοπλισμού ξεκίνησε τον τελευταίο χρόνο.

Βασικά χαρακτηριστικά

- Μεταδίδει σε ζώνη συχνοτήτων sub 1 GHz.
- Εύρος ζώνης 12.5 KHz
- Data Rate για uplink επικοινωνία 200 bps-100 Kbps.
- Εμβέλεια που μπορεί να φτάσει μέχρι και 2 km.
- Υποστηρίζει αμφίδρομη επικοινωνία με ικανοποιητικό data rate στα 100 kbps
- Μέγεθος πακέτου μεγαλύτερο από 10 bytes.
- Απεριοόριστες Συσκευές/gayeway
- Ισχύς μετάδοσης 17 dBm
- Διάρκεια ζωής μπαταρίας: μέχρι και 10 χρόνια

Στον παρακάτω πίνακα παρατίθενται τα βασικά χαρακτηριστικά των LPWAN τεχνολογιών που αναφέραμε.

	Ζώνη Συχνοτήτων	**Εύρος ζώνης**	**DATARATE**	**Εμβέλεια**	**Μέγεθος πακέτου**	**Χωρητικότητα**
SIGFOX	868 MHz (EU) 902 MHz (US)	192 KHz	• UPLINK: 100 bps-140 μηνύματα/μέρα • DOWNLINK: μέχρι 4 μηνύματα των 8 bytes/ημέρα	• Αστικό περιβάλλον: 3-10 Km • Μη αστικό περιβάλλον: 30-50 km	10 kbytes	1M
LORA	868 MHz (EU) 915 MHz (US)	192 KHz	• UPLINK: μέχρι 50 kbps • DOWNLINK: μέχρι 50 kbps	• Αστικό περιβάλλον: 3-10 Km • Μη αστικό περιβάλλον: 15 km	adaptive	1M

RPMA	2.4 GHz	1 MHz	• UPLINK: 624 kbps • DOWNLINK: 156 kbps	• Αστικό περιβάλλον: 1-3 Km • Μη αστικό περιβάλλον: 5-10 km	adaptive (μέχρι 10 kbytes)	500,000
WEIGHTLESS-N	SUB GHz	200 Hz	• UPLINK: 100 bps • DOWNLINK: δεν υποστηρίζει	3 km	μέχρι 20 bytes	απεριόριστο
WEIGHTLESS-P	SUB GHz	12.5 KHz	• UPLINK: 200 bps-100 Kbps • DOWNLINK: 100 kbps	2 km	μεγαλύτερο από 10 bytes	απεριόριστο

2.4.5 Θετικά και αρνητικά LPWAN τεχνολογιών

Στον παρακάτω πίνακα παρατίθενται τα βασικότερα πλεονεκτήματα και μειονεκτήματα για τις πέντε τεχνολογίες που παρουσιάστηκαν παραπάνω.

	ΘΕΤΙΚΑ	**ΑΡΝΗΤΙΚΑ**
SIGFOX	♦ Χαμηλή κατανάλωση ισχύος ♦ Εξαιρετικό για εφαρμογές παρακολούθησης και μέτρησης. ♦ Συνεργασία με μεγάλες εταιρίες του κλάδου (TI, Silicon Labs, Axom)	♦ Ασθενής ασφάλεια: 16bit encryption ♦ Περιορισμένες περιπτώσεις χρήσης (δεν συνιστάται για εφαρμογές που απαιτείται downlink επικοινωνία) ♦ Πολύ χαμηλό datarate
LORA	♦ Μέλη της συμμαχίας μεγάλες εταιρίες όπως CISCO, IBM, KERLINK ♦ Υψηλά επίπεδα ασφάλειας (AES 128 bit) ♦ Ευέλικτο packet size καθορισμένο από τον χρήστη ♦ Ικανοποιητικό datarate ♦ Βρίσκεται σε μεγάλη εμπορική άνοδο ♦ 3-4 φορές μεγαλύτερη διάρκεια ζωής μπαταρίας σε σχέση με τις άλλες LPWAN τεχνολογίες.	♦ Η ικανότητα downlink επικοινωνίας δεν βρίσκεται ακόμα σε ικανοποιητικά επίπεδα
RPMA	♦ Παρά τη καθυστερημένη είσοδο στην αγορά σε σχέση με τις άλλες LPWAN τεχνολογίες, έχει αποκτήσει μεγάλη εμπορική φήμη. ♦ Υποστηρίζει αμφίδρομη επικοινωνία με ικανοποιητικό datarate	♦ Υψηλή κατανάλωση ισχύος, γεγονός που δεν συνάδει με το κριτήριο Long-Battery-Life των LPWAN ♦ Πιθανή παρεμβολή από Wifi και Bluetooth λόγω του φάσματος 2.4GHz που χρησιμοποιεί
WEIGHTLESS-N	♦ Ικανοποιητική εμβέλεια σε αστικό περιβάλλον	♦ Δεν διαθέτει ικανότητα downlink επικοινωνίας ♦ Χαμηλή ταχύτητα ♦ περιορισμένη εμβέλεια σε

		περιβάλλον υπαίθρου.
WEIGHTLESS-P	♦ Προσαρμοστικό data rate γεγονός που το καθιστά ευέλικτο ♦ Υποστηρίζει downlink επικοινωνία με datarate μέχρι 100 kbps	♦ Περιορισμένη εμβέλεια σε περιβάλλον υπαίθρου, μικρότερη από τη Weightless-N.

3. LORA GATEWAY

3.1 O IC880A concentrator

Το Gateway αποτελεί ένα από τα βασικά συστατικά της εργασίας, είναι ο κόμβος που λαμβάνει και προωθεί τα μηνύματα από τις συσκευές προς τον backend infrastructure, και αντίστροφα. Στα πλαίσια της εργασίας, το gateway που χρησιμοποιήθηκε είναι το IC880A της εταιρίας Semtech.

ΕΙΚΟΝΑ 1: Ο IC880A concentrator

Στο παρακάτω σχήμα παρουσιάζεται η πλακέτα του iC880A-SPI. Τα βασικά στοιχεία της πλακέτας αυτής είναι τα pins που χρησιμοποιούνται για τη σύνδεση της πλακέτας σε κάποιο σύστημα, ένας SMA ή uFL connector που χρησιμεύει για τη σύνδεση της κατάλληλης κεραίας, καθώς και 6 led που χρησιμεύουν για να υποδείξουν την εκάστοτε λειτουργία, πιο συγκεκριμένα:

1) Backhaul packet

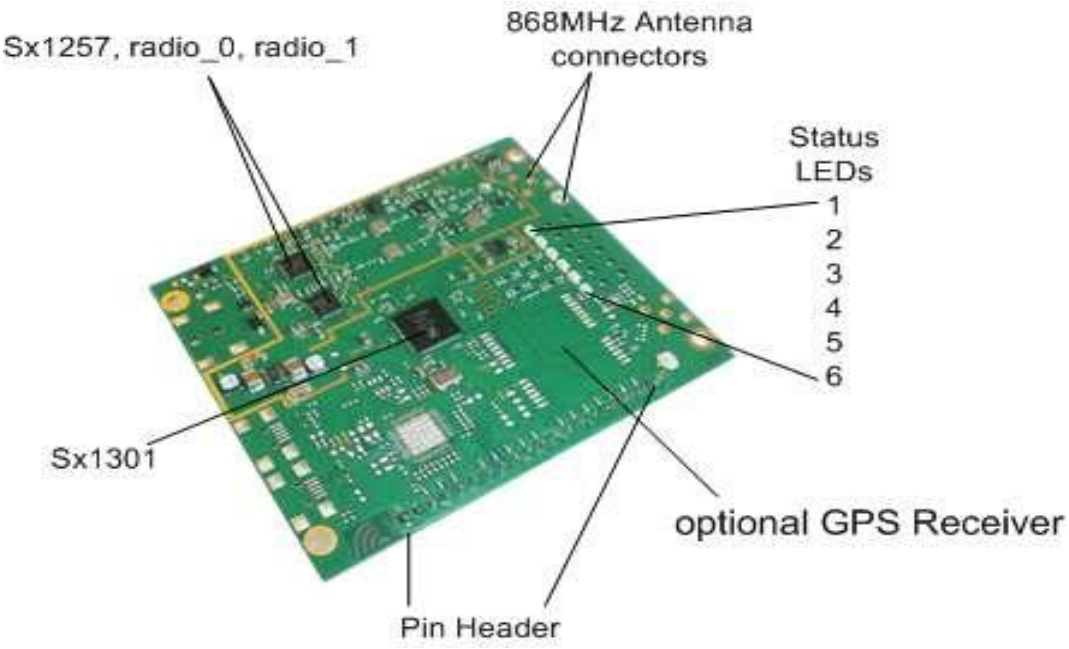
2) TX packet

3) RX Sensor packet

4) RX FSK packet

5) RX buffer not empty

6) Power



ΕΙΚΟΝΑ 2: Ο IC880a concentrator

3.2 Οδηγίες εγκατάστασης του LoRa Gateway

Για το LoRa gateway χρησιμοποιήθηκαν τα παρακάτω στοιχεία:

1) Στη παρούσα εργασία το μοντέλο που χρησιμοποιήθηκε σαν LoRa concentrator στο Gateway είναι το iC880A-SPI – LoRaWAN Concentrator 868MHz. Το προϊόν αυτό μπορεί κάποιος να το προμηθευτεί από την εταιρία IMST στον παρακάτω σύνδεσμο:

http://webshop.imst.de/ic880a-spi-LoRaWAN-concentrator-868mhz.html

τιμή: 150$

Το IC880A αποτελεί το front-end ενός LoRa gateway. Έχει την ικανότητα να λαμβάνει σήματα σε διαφορετικά κανάλια συχνοτήτων την ίδια στιγμή καθώς και να από-διαμορφώνει τα LORA σήματα χωρίς να γνωρίζει το συντελεστή επέκτασης (spreading factor) του κόμβου που τα στέλνει.

2) Μία ουρά για τη κεραία. Μπορεί να τη παραγγείλει κάποιος από το site:

http://webshop.imst.de/pigtail-for-ic880a-spi-and-ic880a-usb.html

τιμή: 6.50$

3) Ένα rasberry pi 2 το οποίο το βρίσκει κανείς στο site:

http://de.farnell.com/raspberry-pi/rpi2-modb-v1-2/sbc-raspberry-pi-2-model-b-v1/dp/2612474

τιμη: 31.5

4) Μία κεραία 868MHZ, 3DBi από το site

http://de.farnell.com/rf-solutions/ant-8whip3h-sma/ant-868mhz-peitsche-gelenk-sma/dp/2305899

τιμή:8$

5) Έναν πάροχο ρεύματος για το rasberry pi

http://de.farnell.com/stontronics/t5454dv/netzteil-raspberry-pi-5v-2a-micro/dp/2427498

τιμή: 8,00$

5) Ένα microSD card (4Gb ή παραπάνω)

http://www.ebay.com/bhp/kingston-4gb-micro-sd

τιμή: 8.00$

6)RPi to iC880a interface

https://www.tindie.com/stores/gnz/

τιμή: 5$

Η σύνδεση με το Internet μπορεί να πραγματοποιηθεί με δύο τρόπους είτε με wifi είτε με Ethernet. Ωστόσο συνιστάται η δεύτερη επιλογή αν είναι εφικτή, καθώς η ασύρματη σύνδεση ίσως αυξήσει τον θόρυβο στα εισερχόμενα σήματα.

3.2.1 Διασύνδεση στοιχείων

Προετοιμασία SD card ώστε να εισαχθεί στο rasberry pi και να κάνει boot up

1. Κατεβάζουμε το image Rasbian jessie Lite από:
   https://www.raspberrypi.org/downloads/raspbian/
2. Εκτελούμε την εντολή

   df -h
   προκειμένου να δούμε ποιες συσκευές είναι προσαρτημένες

3. Εισάγουμε την SD card. Σε περίπτωση που ο Η/Υ δεν έχει υποδοχή για sd card εισάγουμε την sd card σ' έναν sd reader τον οποίον συνδέουμε στον Η/Υ.

4. Εκτελούμε ξανά την εντολή

   df -h

Αυτήν τη φορά θα πρέπει στην έξοδο να εμφανιστεί μία καινούργια συσκευή, η συσκευή αυτή είναι η sd card που εισαγάγαμε στο βήμα 3. Σε περίπτωση που δεν προκύψει κάποια

καινούργια συσκευή, σημαίνει ότι το σύστημα δεν μπορεί να προσαρτήσει αυτόματα συσκευές και θα ακολουθηθεί μία διαφορετική μέθοδος. Εκτελούμε την εντολή

dmesg | tail

Η εντολή αυτή εκτυπώνει τα πιο πρόσφατα μηνύματα συστήματος, επομένως θα πρέπει να περιέχει και πληροφορίες για το όνομα της sd card.

5. Η αριστερή στήλη στην έξοδο της εντολής που εκτελέσαμε στο προηγούμενο βήμα προσδιορίζει το όνομα συσκευής της sd card, και πρέπει να είναι κάτι σαν /dev/mmcblk0p1 ή /dev/sdx1. Τα γράμματα p1 και 1 υποδεικνύει τον αριθμό του partition, στη προκειμένη περίπτωση όμως, το επιθυμητό είναι να γράψουμε ολόκληρη την sd card κι όχι ένα partition. Γι' αυτό το λόγο θα χρειαστεί να αφαιρεθεί αυτό το κομμάτι από την ονομασία και να έχει τη μορφή /dev/mmcblk0 ή /dev/sdX.

6. Αφού έχουμε συγκρατήσει το όνομα της συσκευής, αποσυνδέουμε την sd card.

7. Εκτελούμε την εντολή

   umount /dev/sdX1

   αντικαθιστώντας το sdX1 με το όνομα της συσκευής

8. Εκτελούμε την εντολή

   df -h

Αν το όνομα της SD card εμφανίζεται περισσότερες από μία φορά σημαίνει ότι η κάρτα έχει πολλαπλά partitions οπότε θα πρέπει να επαναληφθεί το βήμα 7 για όλα τα partitions.

9. Εκτελούμε τη παρακάτω εντολή προκειμένου να κάνουμε write το image στην SD card:

   dd bs=4M if=2017-07-05-raspbian-jessie.img of=/dev/sdX conv=fsync

αντικαθιστώντας τη μεταβλητή if= με το path στο img αρχείο, και το /dev/sdX στη μεταβλητή of= με το όνομα της συσκευής.

- Το όνομα της συσκευής πρέπει να προκύψει από τα βήμα 5
- Σε περίπτωση που το block size = 4M δεν δουλέψει, μπορεί να ορισθεί στο 1.
- Αν ο χρήστης δεν έχει συνδεθεί σαν root, θα πρέπει να προσθέσει "sudo" στην αρχή της εντολής.
- Η παραπάνω εντολή θα χρειαστεί περίπου 5 λεπτά για να ολοκληρωθεί, ενώ ένα Led θα ανάβει κατά τη διάρκεια της εκτέλεση
- Προσθέτοντας status=progress μπορούμε να δούμε τη πορεία της εκτέλεσης καθώς η εντολή dd δεν δίνει πληροφορίες για την εξέλιξη της

10. Συνδέουμε το καλώδιο ethernet στο Rpi

11. Συνδέουμε το pigtail στο IC880A

12. Συνδέουμε το καλώδιο ethernet στο Rpi

13. Δημιουργούμε τρύπες στη βάση για τη τοποθέτηση των πλακετών

14. Τοποθετούμε τις πλακέτες στη βάση αφήνοντας ένα κενό μεταξύ τους

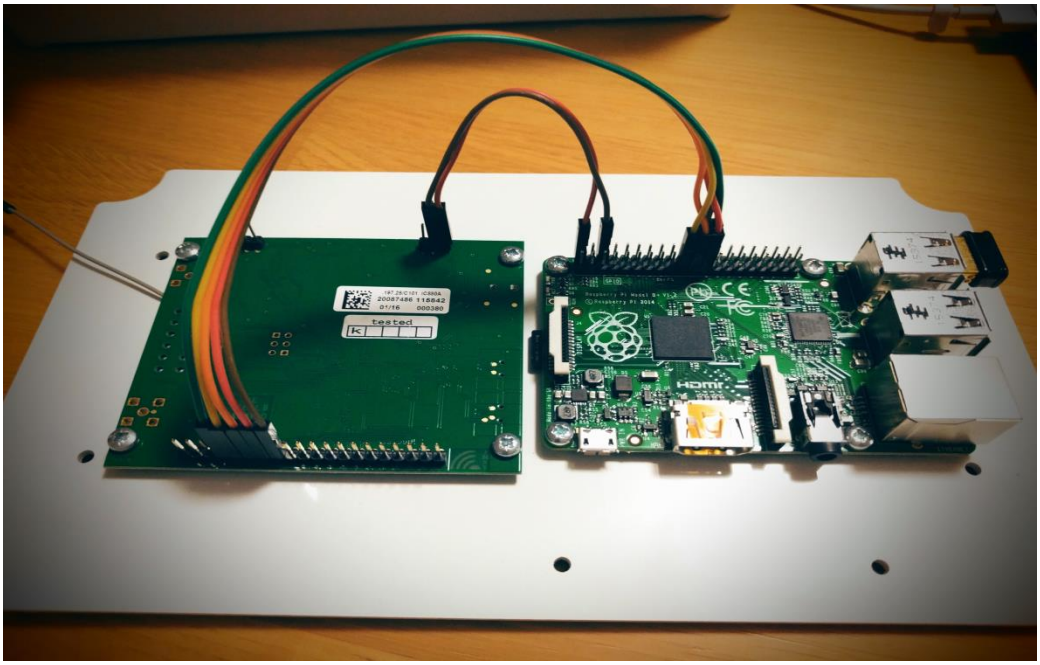
16. Τοποθετούμε τη πλάκα στήριξης στο περίβλημα

15. Τοποθετούμε τη κεραία.

16. Συνδέουμε τα καλώδια μεταξύ των δύο πλακετών σύμφωνα με τον παρακάτω πίνακα.

iC880a	Description	RPi physical pin

pin		
21	Supply 5V	2
22	GND	6
13	Reset	22
14	SPI CLK	23
15	MISO	21
16	MOSI	19
17	NSS	24



ΕΙΚΟΝΑ 3: LoRa Gateway

17. Τροφοδοτούμε με ρεύμα το Rpi, το οποίο θα τροφοδοτήσει και τον concentrator

3.2.2 Ρύθμιση του λογισμικού

Από κάποιο Host στο ίδιο LAN εκτελούμε τη παρακάτω εντολή για να συνδεθούμε στο Rpi

ssh pi@raspberrypi.local default password: raspberry

1. Χρησιμοποιούμε το raspi-config για να ενεργοποιήσουμε το SPI με την εντολή:

   $ sudo raspi-config

2. Κάνουμε reboot το σύστημα με την εντολή:

   $ sudo reboot

3. Ρυθμίζουμε τη τοποθεσία και την ώρα με τις εντολές:

   $ sudo dpkg-reconfigure locales

   $ sudo dpkg-reconfigure tzdata

4. Εγκαθιστούμε το πρόγραμμα git, αφού πρώτα κάνουμε update το σύστημα.

   $ sudo apt-get update

   $ sudo apt-get upgrade

   $ sudo apt-get install git

5. Δημιουργούμε έναν χρήστη ttn, και τον προσθέτουμε στους sudoers.

   $ sudo adduser ttn

   $ sudo adduser ttn sudo

   $ sudo visudo προσέτουμε τη γραμμή ttn ALL=(ALL) NOPASSWD: ALL

6. Κάνουμε logout, και ύστερα login, αυτήν τη φορά σαν ttn user, και αφαιρούμε τον default user pi με την εντολή:

   $ sudo userdel -rf pi

3.2.3 Ρύθμιση του wifi μέσω γραμμής εντολών

* Θα χρειαστεί ο κωδικός του wifi δικτύου που αναγράφεται στο πίσω μέρος του router

1. Για να δούμε τα διαθέσιμα δίκτυα καθώς και χρήσιμες πληροφορίες για αυτά εκτελούμε την εντολή:

   sudo iwlist wlan0 scan

   από τη λίστα που θα εμφανιστεί, αναζητούμε τα πεδία:

   α) 'ESSID:"testing"' που είναι το όνομα του δικτύου

   β) 'IE: IEEE 802.11i/WPA2 Version 1',

   που είναι η πιστοποίηση που χρησιμοποιεί. Σημειώνεται ότι ο οδηγός αυτός μπορεί να δουλέψει με WPA και WPA2, αλλά ίσως να μην μπορεί να δουλέψει με WPA2 enterprise

2. Ανοίγουμε το αρχείο ρυθμίσεων wpa-supplicant

   sudo vi /etc/wpa_supplicant/wpa_supplicant.conf

   και προσθέτουμε τις παρακάτω γραμμές στο τέλος του αρχειου:

```
network={
ssid="testing"
psk="testingPassword"
}
```

* όπου testing = το όνομα του δικτύου

* όπου testingPassword = ο κωδικός του δικτύου

εναλλακτικά, σε περίπτωση μη ασφαλούς δικτύου, χωρίς κωδικό πρόσβασης,προσθέτουμε τις ακόλουθες γραμμές:

```
network={
ssid="testing"
```

```
key_mgmt=NONE
}
```

ενναλλακτικά, σε περίπτωση κρυφού δικτύου, προσθέτουμε τις ακόλουθες γραμμές:

```
network={
ssid="yourHiddenSSID"
scan_ssid=1
psk="Your_wifi_password"
}
```

εναλλακτικά, σε περίπτωση πολλαπλών δικτύων, δίνουμε ένα id σε κάθε δίκτυο όπως στο παράδειγμα παρακάτω:

```
network={
ssid="SchoolNetworkSSID"
psk="passwordSchool"
id_str="school"
}

network={
ssid="HomeNetworkSSID"
psk="passwordHome"
id_str="home"
}
```

ενώ αν θέλουμε να δώσουμε προτεραιότητα σε ένα από τα δίκτυα, δίνουμε έναν αριθμό προτεραιότητας σε κάθε δίκτυο, όπως παρακάτω:

```
network={
ssid="HomeOneSSID"
psk="passwordOne"
priority=1
id_str="homeOne"
}
```

```
network={
ssid="HomeTwoSSID"
psk="passwordTwo"
priority=2
id_str="homeTwo"
}
```

3. Κάνουμε fork το παρακάτω repository από το github

   https://github.com/ttn-zh/ic880a-gateway/tree/spi

   και στη συνέχεια εκτελούμε τις ακόλουθες εντολές για την εγκατάσταση

```
$ git clone -b spi https://github.com/ttn-zh/ic880a-gateway.git ~/ic880a-gateway
$ cd ~/ic880a-gateway
$ sudo ./install.sh spi
```

* Ο installer αλλάζει το hosname του Rpi σε ttn-gateway

4. Αποσυνδέουμε το καλώδιο του ethernet, καθώς πλέον το LoRa gateway είναι σε λειτουργία

3.2.4 Ρύθμιση του gateway για να επικοινωνεί με τον server

1. Ανοίγουμε το αρχείο local_conf.json

```
cd /opt/ttn-gateway/bin/
sudo vi local_conf.json
```

2. Το αρχείο αυτό έχει την ακόλουθη μορφή:

```
{
"gateway_conf": {
"gateway_ID": "B827EBFFFE13BCBF",
"servers": [ { "server_address": "X.X.X.X", "serv_port_up": 1700, "serv_port_down": 1700,
"serv_enabled": true } ],
"ref_latitude": 38.049863,
"ref_longitude": 23.788187,
"ref_altitude": 6,
"contact_email": "glimperop@gmail.com",
"description": "ttn-ic880a"
}
}
```

Αντικαθιστούμε στο πεδίο "server-address" το "X.X.X.X" με τη διεύθυνση IP του server.

Αποθηκεύουμε τις αλλαγές και βγαίνουμε από το αρχείο.

3. Από τον φάκελο που βρισκόμαστε εκτελούμε την ακόλουθη εντολή

   sudo ./start.sh

   Έτσι το gateway πλέον προωθεί τα μηνύματα του στον server.

4. Το Backend infrastructure του συστήματος

4.1 The Thing Network

Το backend είναι το κεντρικό κομμάτι του συστήματος καθώς είναι υπεύθυνο για μία πληθώρα από λειτουργίες που θα αναλυθούν στη συνέχεια.

Μέχρι σήμερα έχουν αναπτυχθεί διάφορα λογισμικά που ικανοποιούν τις απαιτήσεις του LoRa δικτύου, ωστόσο στη συγκεκριμένη εργασία έγινε χρήση του ανοιχτού κώδικα του The Things Network.

Το «Δίκτυο των Πραγμάτων» ("The Things Network") είναι ένα ανοιχτό, αποκεντρωμένο, ελεύθερο δίκτυο διακίνησης Δεδομένων που ακολουθεί τη λογική του «Διαδικτύου των Πραγμάτων» ("Internet of Things" - IoT) και βασίζεται στη συνεισφορά του κοινού.

Η αποστολή του είναι να οικοδομηθεί ένα αποκεντρωμένο και τεχνολογικά ανεξάρτητο δίκτυο "IoT" το οποίο να ανήκει και να λειτουργεί από τους χρήστες του. Για το σκοπό αυτό, χτίστηκε ένα κατανεμημένο δίκτυο το οποίο υποστηρίζει όλα τα είδη συνδεσιμότητας με "IoT".

Οι βασικές αρχές που διέπουν το The things Network είναι:

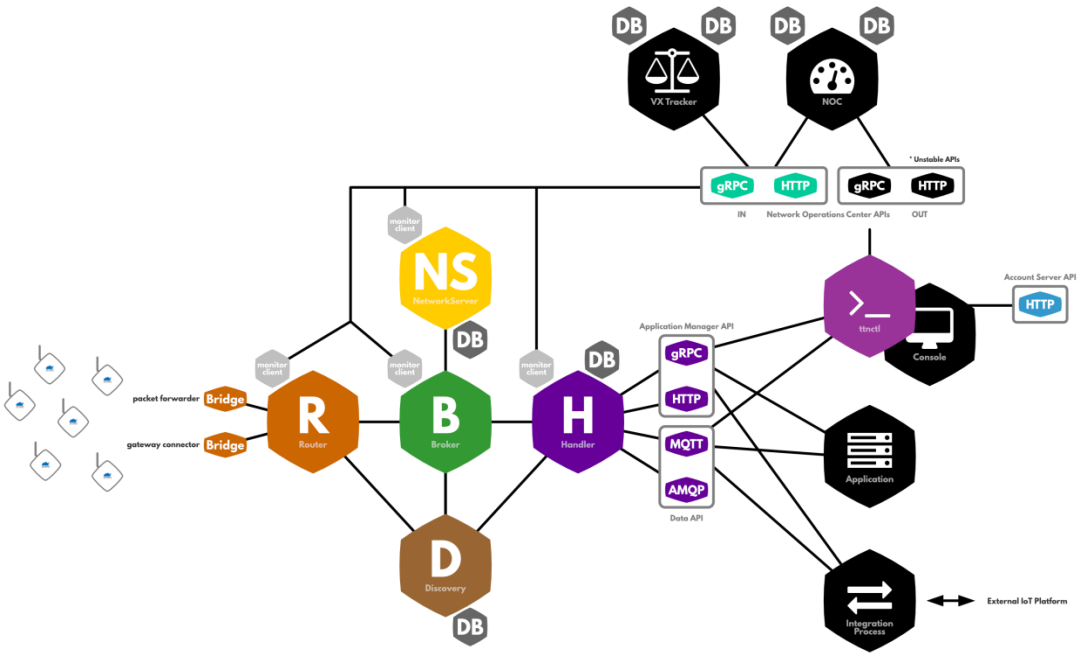
- **Τα δεδομένα σας είναι ασφαλή** – τα δεδομένα κρυπτογραφούνται σε όλα τα στάδια του δικτύου, από την αρχή έως το τέλος
- **Καθαρή ουδετερότητα** – όλα τα δεδομένα τυγχάνουν ίσης μεταχείρισης
- **Ανοιχτή πηγή** – η τεχνολογία που αναπτύχθηκε είναι και θα παραμεινει ανοιχτή

Στόχος της κοινότητας χρηστών είναι να οικοδομήσει ένα παγκόσμιο ανοικτό αποκεντρωμένο διαδίκτυο "IoT".

Η αποκέντρωση βρίσκεται στο επίκεντρο των πάντων. Είναι ενσωματωμένη στην αρχιτεκτονική και σε κάθε απόφαση σχεδιασμού έτσι ώστε να διασφαλιστεί ότι κανένας από μόνος του δεν θα μπορέσει να αποκτήσει τον έλεγχο του Δικτύου.

4.2 Αρχιτεκτονική του The things Network backend

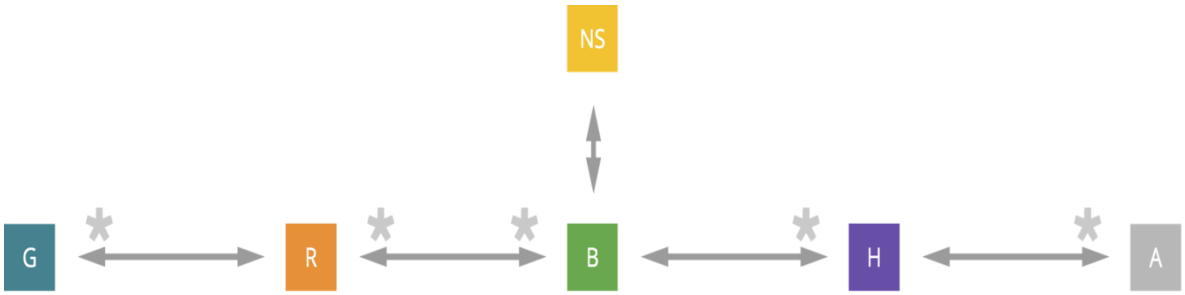
Το backend system του The things Network είναι υπεύθυνο για τη δρομολόγηση των δεδομένων μεταξύ κόμβων και εφαρμογών. Ενα τυπικό IoT δίκτυο απαιτεί τη χρησιμοποίηση gateway σαν γέφυρα μεταξύ του εκάστωτε πρωτοκόλλου και του Internet. Σε περιπτώσεις μάλιστα που οι κόμβοι υποστηρίζουν από μόνοι τους IP, ο ρόλος των gateway περιορίζεται μόνο στη προώθηση των δεδομένων. Αντίθετα, σε τεχνολογίες όπως το LoRaWAN που είναι non-IP πρωτόκολλο, απαιτείται επιπλέον επεξεργασία και δρομολόγηση για τη μεταφορά των μηνυμάτων από τους κόμβους στις εφαρμογές.



ΔΙΑΓΡΑΜΜΑ 3: Αρχιτεκτονική του The Things Network

Το backend λαμβάνει χώρα ανάμεσα από τα gateways και τις εφαρμογές, και υλοποιεί την επεξεργασία και τη δρομολόγηση των δεδομένων με τρόπο αποκεντρωμένο (decentralized) και κατανεμημένο (distributed). Τα βασικά στοιχεία του backend είναι:

- **Router (R)**
- **Broker (B**
- **Network Server (NS)**
- **Handler (H)**



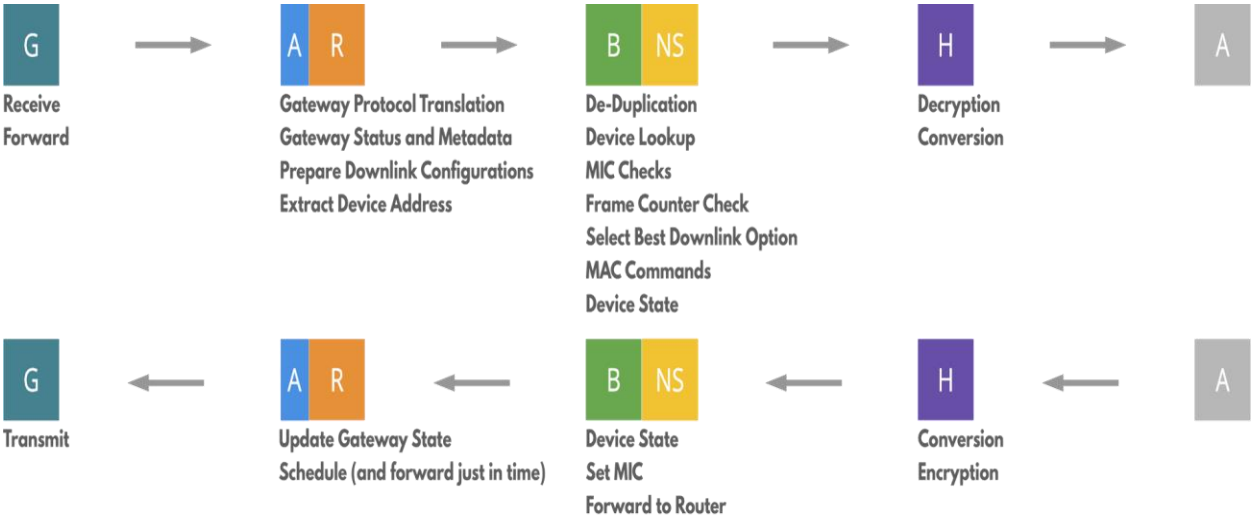
ΔΙΑΓΡΑΜΜΑ 4: Τα στοιχεία του The Things Network

Στο παραπάνω σχήμα, με (G) συμβολίζεται το Gateway, με (A) το Application, ενώ με (R,NS,B και H) τα στοιχεία του backend που αναφέρθηκαν παραπάνω.

Οι κόμβοι του δικτύου στέλνουν μηνύματα πάνω από LoRaWAN πρωτόκολλο, τα οποία λαμβάνονται από gateway που βρίσκονται εντός εμβέλειας. Τα gateways προωθούν τα μηνύματα στο backend και συγκεκριμένα στο Router (R). Το Router διαχειρίζεται τη κατάσταση των Gateways, προγραμματίζει την επόμενη μετάδοση και προωθεί τα μηνύματα στον Broker (B). Ο Broker είναι η καρδιά του συστήματος, καθώς είναι υπεύθυνος στο να βρει ποια συσκευή στέλνει το εκάστοτε μήνυμα, σε ποια εφαρμογή ανήκει η κάθε συσκευή και να προωθήσει έτσι το κάθε μήνυμα στη σωστή εφαρμογή αλλά και στο σωστό Router σε περίπτωση downlink επικοινωνίας. Ο Network Server είναι υπεύθυνος για τις λειτουργικές διαδικασίες που αφορούν το LoRaWAN πρωτόκολλο. Τέλος, ο Handler (H) συνδέται με τον Broker, διαχειρίζεται τα δεδομένα για κάθε εφαρμογή ενώ πραγματοποιεί και τη κρυπτογράφηση (downlink επικοινωνία) και την αποκρυπτογράφηση (uplink επικοινωνία) των μηνυμάτων.

4.3 Διαδικασία ροής μηνυμάτων

Στο σχήμα που ακολουθεί, παρουσιάζεται η πορεία των μηνυμάτων μέσα στο backend του The Things Network. Ακολουθεί η επεξήγηση κάθε σταδίου ξεχωριστά.

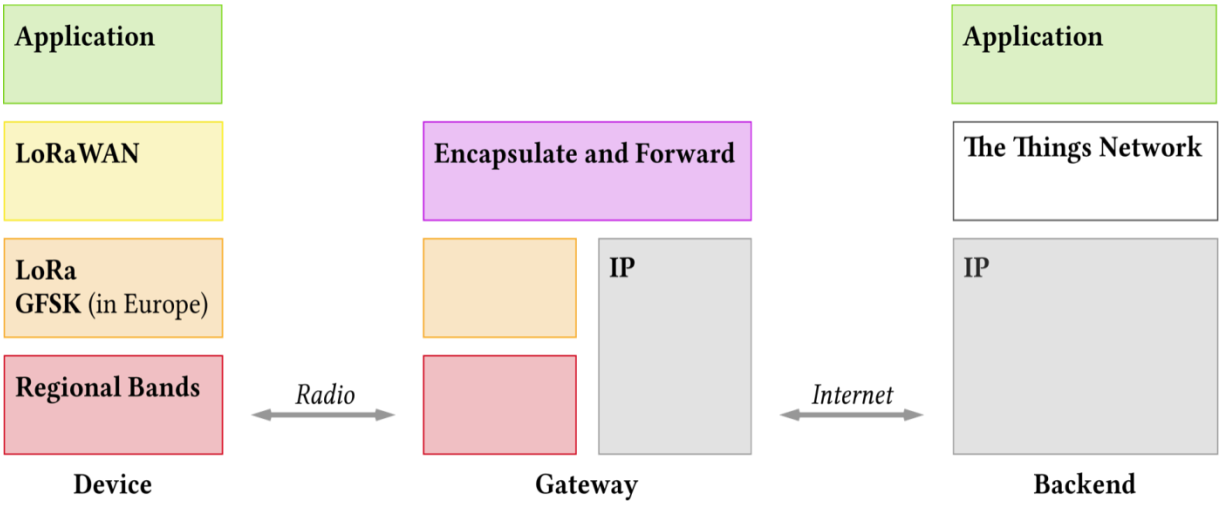


ΔΙΑΓΡΑΜΜΑ 5: Επικοινωνία των επιμέρους στοιχείων

Gateway Protocol Translation

Όταν ένα gateway λαμβάνει ένα μήνυμα το οποίο έχει μεταδοθεί πάνω από LoRa, το προωθεί στο The Things Network backend μέσω Internet. Συγκεκριμένα, όταν ένα μήνυμα λαμβάνεται, προωθείται στο backend μαζί με κάποια metadata όπως το RSSI (Received signal strength indication), το SNR (Signal-to-noise ratio),καθώς και πληροφορίες για τις συντεταγμένες του gateway (longitude, latitude, altitude).

Το The Things Network για να επιτύχει βέλτιστη επικοινωνία μεταξύ gateway και backend, ανέπτυξε το δικό της gateway protocol το οποίο ωστόσο έχει παρόμοια δομή με τα υπόλοιπα gateway protocols. Προκειμένου το backend να είναι συμβατό με οποιοδήποτε gateway protocol έχει αναπτυχθεί μία γέφυρα (LoRa-gateway bridge) για αυτόν τον σκοπό.



ΔΙΑΓΡΑΜΜΑ 6: Device-gateway-backend

Gateway Metadata (Router)

Οι γεωγραφικές συντεταγμένες του gateway, το είδος διαμόρφωσης καθώς και πληροφορίες που αφορούν το rssi (Received signal strenght indication) και το snr (signal to noise ratio) είναι πολλές φορές χρήσιμες. Για τον λόγο αυτόν το backend και συγκεκριμένα ο Router αποθηκεύει αυτές τις πληροφορίες (metadata) προκειμένου να τις προωθήσει στην συνέχεια στον broker.

Downlink επικοινωνία (Router)

Κάθε uplink μήνυμα που στέλνει μια συσκευή, ακολουθείται από ένα downlink μήνυμα - απάντηση από το router. Κάθε συσκευή έχει δύο παράθυρα λήψης, που ενεργοποιούνται ένα και δύο δευτερόλεπτα μετά τη μετάδοση του μηνύματος αντίστοιχα. Ο Router υπολογίζει ένα σκορ για κάθε παράθυρο καθώς πρέπει να διαλέξει ποια είναι η καταλληλότερη επιλογή. Το σκορ προκύπτει από:

- **Χρόνος διάδοσης :** Κάθε εκπομπή ενός μηνύματος μπλοκάρει για ένα διάστημα τους δέκτες του gateway, έτσι γίνεται σαφές ότι μηνύματα με μικρότερο χρόνο διάδοσης σε υψηλό datarate έχουν προτεραιότητα όσο αφορά τη downlink επικοινωνία.

- **Ισχύς σήματος (SNR) :** Όσο μεγαλύτερη είναι η ισχύς του σήματος, τόσο πιο πιθανό είναι ότι ο κόμβος θα λάβει το μήνυμα

- **Χρησιμοποίηση του Gateway:** Προκύπτει από το ποσοστό του χρόνου κατά το οποίο ένα gateway λαμβάνει μηνύματα. Επομένως ένα gateway με υψηλή χρησιμοποίηση προτιμάται σε περιπτώσεις που επιθυμούμε κυρίως uplink επικοινωνία ενώ gateways χαμηλής χρησιμοποίησης προτιμάται στην αντίθετη περίπτωση.

Εξαγωγή διεύθυνσης συσκευής (Router)

Το πρώτο βήμα για τη δρομολόγηση ενός πακέτου αφορά τη διεύθυνση της συσκευής (DevAddr). Η διεύθυνση αυτή είναι μία μη μοναδική 32-bit διεύθυνση. Ο κάθε Broker μέσω του Network server ανακοινώνει το πρόθεμα της διεύθυνσης για κάθε συσκευή ώστε τα υπόλοιπα στοιχεία να γνωρίζουν σε ποιον broker ταιριάζει το κάθε μήνυμα. Η διαδικασία αυτή είναι όμοια με αυτήν που ακολουθείται για την ανακοίνωση των IP range στο BGP.

Αναζήτηση συσκευής και εφαρμογής (Broker/Network Server)

Όπως αναφέρθηκε το DevAddr δεν είναι μοναδικό, για τον λόγο αυτόν είναι απαραίτητο να καθοριστεί η μοναδικότητα κάθε συσκευής, ώστε να γίνεται η σωστή προωθηση στη κατάλληλη εφαρμογή στην οποία ανήκει. Αυτό επιτυγχάνεται με τον κώδικα ακεραιότητας μηνυμάτων MIC (message integrity code). Ο έλεγχος κάθε μηνύματος πραγματοποιείται από τον Broker, επικυρώνοντας το MIC κάθε συσκευής με το Network Session Key. Αν η επικύρωση δεν πραγματοποιηθεί, το μήνυμα απορρίπτεται.

Έλεγχος πεδίου Frame Counter (Broker)

Αποτελεί ένα μέτρο ασφάλειας για την αποφυγή επιθέσεων. Μετά το Mic validation που εξηγήθηκε παραπάνω ο Broker ελέγχει την εγκυρότητα του Frame counter. Κάθε συσκευή αυξάνει το frame counter κατα ένα σε κάθε μετάδοση. Στη περίπτωση που ένα μήνυμα έχει Frame Counter μικρότερο από το σωστό, απορρίπτεται από τον Broker. Στη περίπτωση που ένα μήνυμα έχει frame Counter μεγαλύτερο από το σωστό, ο Broker απορρίπτει το μήνυμα αν Fake_Frame_Counter - Correct_Frame_counter > 16384 σε αντίθετη περίπτωση το μήνυμα γίνεται αποδεκτό.

Το LoRaWAN υποστηρίζει 16-bit και 32-bit Frame Counter, ωστόσο το μήνυμα μεταφέρει μόνο τα 16 lsb (least significant bits) για τον λόγο αυτό το backend πρέπει να παρακολουθεί το 32-bit Frame Counter που κρατάει για κάθε συσκευή.

Συλλογή Metadata (Broker)

Ο Broker αφού ολοκληρώσει τον έλεγχο και πριν προωθήσει το μήνυμα, συνενώνει τα Metadata που που είχε αποθηκεύσει ο router, με το υπόλοιπο μήνυμα.

Επιλογή για την κατάλληλη downlink επικοινωνία (Broker)

Ο Broker δεν γνωρίζει μέσω ποιου gateway έχει αποσταλεί το κάθε μήνυμα, γι' αυτό δεν μπορεί να επιλέξει τον καταλληλότερο τρόπο downlink επικοινωνίας. Ο Router όμως, όπως αναφέρθηκε παραπάνω υπολογίζει ένα σκορ για κάθε ζεύξη και το γνωστοποιεί στον Broker ο οποίος πλέον μπορεί αν αποστείλει το μήνυμα με τον βέλτιστο τρόπο.

Device State και Mac commands (Network Server)

Προτού ο Broker προωθήσει κάποιο μήνυμα στον Handler, το στέλνει πρώτα στον Network Sever ο οποίος ανανεώνει κάθε φορά το State της συσκευής. Επιπλέον προσθέτει ένα downlink template στο μήνυμα με πληροφορίες όπως το Frame counter και το message type. Με τον τρόπο αυτό, αν ο Handler θέλει να στείλει ένα μήνυμα στη συσκευή, απλά προσθέτει το payload στο μήνυμα. Εκτός από το state των συσκευών, ο Network server έχει τη δυνατότητα να προσθέσει εντολές MAC στο μήνυμα που αφορούν για παράδειγμα το snr ή το data rate.

Αποκρυπτογράφηση μηνύματος (Handler)

Ο Handler χρησιμοποιώντας το Application Session key, πραγματοποιεί αποκρυπτογράφηση του μηνύματος

Μετατροπή πεδίου Payload (Handler)

Μετά την αποκρυπτογράφηση, ο Handler μπορεί να κωδικοποιήσει ή να μετατρέψει περαιτέρω τα δεδομένα. Παραδείγματος χάρην, μία τιμή θερμοκρασίας εκφρασμένη σε Celsius μπορεί να μετατραπεί σε Fahrenheit.

MQTT (Handler)

Ο Handler αφότου λάβει κάποιο μήνυμα και το αποκρυπτογραφήσει, το κάνει publish στον MQTT broker του συστήματος. Συγκεκριμένα κάνει publish το μήνυμα σε json μορφή στο topic: **<app_name>/devices/<dev_name>/up**

Επομένως ο χρήστης κάθε εφαρμογής μπορεί να δει τα μηνύματα του κλάνοντας subscribe στο topic της εφαρμογής.

Παράδειγμα ενός μηνύματος είναι το ακόλουθο:
{payload: 'BQY=', fields:{temperature:12.86 }, port:14, counter:1234, metadata: [{ frequency: 868.1, datarate:'SF7BW125', codingrate:'4/5', ... longitude:6.55738, latitude:53.18977 }] }

Downlink (Handler)

Όπως και στην uplink επικοινωνία έτσι και στη downlink ο Handler είναι υπεύθυνος για τη κρυπτογράφηση του μηνύματος. Υπάρχουν 3 περιπτώσεις κατά τις οποίες αποστέλλεται μήνυμα το σύστημα προς κάποια συσκευή. Η πρώτη περίπτωση και πιο προφανής είναι όταν ο χρήστης της εφαρμογής θέλει να στείλει κάποιο μήνυμα προς τη συσκευή. Τότε όπως είδαμε παραπάνω, ο Handler απλώς προσθέτει το payload στο έτοιμο template που δημιούργησε ο network server. Η δεύτερη περίπτωση αφορά τα μηνύματα που απαιτούν επιβεβαίωση, στη περίπτωση αυτή ο Handler περιμένει για ένα μικρό χρονικό διάστημα μήπως προκύψει κάποιο payload που αφορά τη συγκεκριμένη συσκευή που απαιτεί επιβεβαίωση, όταν το χρονικό αυτό διάστημα λήξει στέλνει το μήνυμα επιβεβαίωσης. Τέλος, η τρίτη περίπτωση αφορά τις εντολές MAC που στέλνει ο Network server προς κάποια συσκευή.

Device State (Network Server)

Ο Broker αφού λάβει το downlink μήνυμα από τον Handler θα το προωθήσει στον Network Server, ο οποίος αφού ενημερώσει τη κατάσταση της συσκευής, όπως συμβαίνει αντίστοιχα με την uplink επικοινωνία, θα επαναπροωθήσει το μήνυμα στον Broker.

Δρομολόγηση downlink επικοινωνίας (Router)

Ο Router λαμβάνοντας ένα downlink μήνυμα από τον Broker, επιλέγει το κατάλληλο gateway για να το προωθήσει. Οπως αναφέρθηκε και στην αρχή του κεφαλαίου, ο Router είναι υπεύθυνος για τη διαχείριση και τη δρομολόγηση μηνυμάτων από και προς τα gateways.

4.4 Η Επικοινωνία μεταξύ των επιμέρους στοιχείων του συστήματος

Ένα κατανεμημένο σύστημα όπως αυτό του The Things Network απαιτεί μεγάλη ταχύτητα στην επικοινωνία των στοιχείων που το απαρτίζουν (Router, Broker, Network Server, Handler). Η απαίτηση αυτή καλύπτεται ένα σύστημα απομακρυσμένης κλήσης διαδικασιών που αναπτύχθηκε από τη google, το **gRPC**.

Το grpc δίνει τη δυνατότητα στον server να αναπτυξει και να υλοποιήσει συναρτήσεις, τις οποίες μπορεί να τις εκτελέσει ο client σαν να είναι τοπικές. Το βασικό πλεονέκτημα του gRPC είναι η ουδετερότητα του στη γλώσσα στην οποία είναι γραμμένο ένα περιβάλλον (language-neutral), έτσι οι προγραμματιστές αναπτύσσουν τις πλατφόρμες τους σε server ή client side, σε όποια γλώσσα επιθυμούν. Επιπλέον το gRPC αξιοποιεί το HTTP / 2 σε πολλαπλά αιτήματα μέσω μιας μόνο σύνδεσης TCP και εξοικονομεί εύρος ζώνης με συμπίεση κεφαλίδας (header compression).

Το gRPC χρησιμοποιεί το protocol buffer για τη συριοποίηση των δεδομένων. Είναι ένα υψηλού επιπέδου πρωτόκολλο το οποίο χρησιμοποιείται για τη μετατροπή δομών δεδομένων σε συμπαγή δυαδική μορφή.

4.5 Εγκατασταση του λογισμικού

Ο server που χρησιμοποιήθηκε για την ανάπτυξη του backend infrastructure αλλά και του web interface έχει τα παρακάτω χαρακτηριστικά:

Αρχιτεκτονική: x86_64

CPU(s): 4

Όνομα μοντέλου: Intel® Core™ i3-4010U CPU @ 1.70 GHz

Λειτουργικό σύστημα: kali GNU/Linux 2.0 (GNU/Linux 4.4.0-66-generic x86_64)

IP διεύθυνση: 94.70.239.210

Η διαδικασία που ακολουθήθηκε για την εγκατάσταση του backend είναι η εξής:

1. Go Language

- Κατεβάσαμε την έκδοση 1.8 έκδοση της Go lang χρησιμοποιώντας τη παρακάτω εντολή από ένα Linux Terminal

  curl -O https://storage.googleapis.com/golang/go1.8.linux-amd64.tar.gz

- Ελέγξαμε την αυθεντικότητα της πηγής χρησιμοποιώντας την εντολή sha256sum και ελέγχοντας το 256-bit-hash με το checksum value στη Go Download σελιδα

  sha256sum go1.7*.tar.gz

- Εγκαταστήσαμε τη Go lang

  tar xvf go1.8.linux-amd64.tar.gz

- Αλλάξαμε την ιδιοκτησία του Go σε root

  sudo chown -R root:root ./go

- Μεταφεραμε το go directory στο /usr/local

  sudo mv go /usr/local

- Ρυθμίσαμε τα Go Paths και το Go Enviroment δημιουργώντας ένα αρχείο .profile

  sudo vi ~/.profile

και αντγράφωντας τα παρακάτω

export GOROOT=$HOME/go

export GOPATH=$HOME/work

export PATH=$PATH:$GOROOT/bin:$GOPATH/bin

- Αφού αποθηκεύσουμε το αρχείο κάναμε ανανέωση του προφίλ με την εντολή

  source ~/.profile

2. Protobuf compiler

- Επιλέγοντας τη πιο πρόσφατη έκδοση του protocol buffer εκτελέσαμε την εντολή *curl -OL https://github.com/google/protobuf/releases/download/v3.2.0/protoc-3.2.0-linux-x86_64.zip*
- Στη συνέχεια κάναμε unzip το αρχείο ***unzip protoc-3.2.0-linux-x86_64.zip -d protoc3***
- Στη συνέχεια μέσα από τον φάκελο που δημιουργήθηκε εκτελέσαμε τα παρακάτω

  ./configure

  make check

  sudo make install

3. Redis

- Για την εγκατάσταση του redis εκτελέσαμε διαδοχικά τις παρακάτω εντολές:

  Wget http://download.redis.io/releases/redis-3.2.9.tar.gz

  tarxzf redis-3.2.9.tar.gz

  Cd redis-3.2.9

  make

- Για να ενεργοποιήσουμε τον redis server εκτελούμε *sudoservice redis-server start*

4. RabbitMQ

- Ενεργοποιούμε το RabbitMQ repository ***echo"deb http://www.rabbitmq.com/debian/ testing main" >>/etc/apt/sources.list***
- Προσθέτουμε το κλειδί πιστοποίησης για το πακέτο ***curl http://www.rabbitmq.com/rabbitmq-signing-key-public.asc | sudo apt-key add***
- Κάνουμε update τις πηγές που προσθέσαμε προηγουμένως ***sudo apt-get update***

- Εγκαθιστούμε το RabbitMQ server ***sudoapt-get install rabbitmq-server***
- Ρυθμίσαμε τον RabbitMQ ώστε να διαχειρίζεται τον μέγιστο αριθμό συνδέσεων ανοίγοντας το αρχείο rabbitmq-server και αφαιρέσαμε το σχόλιο (αφαιρέσαμε το #) από τη γραμμή limit. **sudo vi /etc/default/rabbitmq-server**
- Στη συνέχεια ενεργοποιήσαμε το RabbitMQ Management Console εκτελώντας: ***sudo rabbitmq-plugins enable rabbitmq_management***
- Μ' αυτόν τον τρόπο μπορούμε να διαχειριστούμε το rabbitMQ μέσω web στη διεύθυνση http://[your droplet's IP]:15672/ χρησιμοποιώντας ως username "guest" και ως κωδικό "guest".
- Τέλος δηλώσαμε την ανταλλαγή ttn.handler τύπου topic με την εντολή: ***rabbitmqadmin declare exchange name=ttn.handler type=topic auto_delete=false durable=true***

5. ttn

- Ο πηγαίος κώδικας του the things network βρίσκεται στον σύνδεσμο https://github.com/TheThingsNetwork/ttn
- Κάνουμε fork το παραπάνω repository
- Κάνουμε clone το παραπάνω fork εκτελώντας τη παρακάτω εντολή ***git clone --branch develop https://github.com/YOURUSERNAME/ttn.git $GOPATH/src/github.com/TheThingsNetwork/ttn***
- ***cd $GOPATH/src/github.com/TheThingsNetwork/ttn.***
- Εγκαταστήσαμε τις εξαρτήσεις εκτελώντας: *make test.*
- Στη συνέχεια εκτελούμε ***make build*** για να δημιουργηθεί το ttn και το ttnctl.
- Εγκαθιστούμε τα go binaries στο path $HOME/bin/ εκτελώντας ***make dev***
- Ρυθμίζουμε το εργαλείο διαχείρισης του συστήματος "ttnctl" αντιγράφοντας το αρχείο dev-example που βρίσκεται στο $HOME/src/github/Thethingsnetwork/ttn/.env/ στο μονοπάτι εργασίας μας εκτελώντας : ***cp $HOME/src/github/Thethingsnetwork/ttn/.env/ttnctl.yml.dev-example ~/.ttnctl.yml.***
- Αντιγράφουμε το πιστοποιητικό server.cert του discovery server στον φάκελο του ttnctl εκτελώντας: ***cp $HOME/src/github/Thethingsnetwork/ttn/.env/discovery/server.cert ~/.ttnctl/ca.cer***t
- Για να δούμε αν οι ρυθμισεις έχουν γίνει σωστά, εκτελούμε **ttnctl config**. Το αποτέλεσμα θα πρέπει να είναι το παρακάτω:

INFO Using config:

configfile: /home/your-user/.ttnctl.yml
datadir: /home/your-user/.ttnctl
auth-server:*https://account.thethingsnetwork.org*
discovery-address:localhost:1900
router-id:dev
handler-id:dev
mqtt-address:localhost:1883

Επιλέξαμε να λειτουργήσουμε το ttn backend χρησιμοποιώντας Docker compose. Το configuration file βρίσκεται στο$HOME/src/github/thethingsNetwork/ttn/docker-compose.yml στο οποίο μπορούμε να κάνουμε περαιτέρω ρυθμίσεις.

Όλα τα στοιχεία του συστήματος (redis, rabbitmq, discovery, network-server, router, broker, handler) ενεργοποιούνται εκτελώντας την εντολή **sudo docker-compose up**. Εναλλακτικά μπορεί κάποιος να ενεργοποιήσει το σύστημα χρησιμοποιώντας το εργαλείο Forego που παρέχει το TTN, στα πλαίσια της εργασίας όμως, επιλέξαμε να χρησιμοποιήσουμε το docker-compose.

4.6 InfluxData Time Series Database Monitoring & Analytics

Η InfluxDB χρησιμοποιείται για την αποθήκευση μεγάλου όγκου timestamped δεδομένων και είναι ιδανική επιλογή για την αποθήκευση real time δεδομένων σε IoT εφαρμογές.

4.6.1 Διαδικασία εγκατάστασης InfluxDB:

- curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add - /source /etc/lsb-release
- echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
- sudo apt-get update && sudo apt-get install influxdb
- sudo service influxdb start

Μετά την εγκατάσταση της influxDB δημιουργήσαμε τη βάση δεδομένων testLoRapot, στην οποία αποθηκεύονται τα δεδομένα, εκτελώντας τις ακόλουθες εντολές:

- Influx
- CREATE DATABASE testLoRapot

4.6.2 Απο τον server στη βάση δεδομένων

Όπως περιγράψαμε παραπάνω στο 4ο κεφάλαιο, τα μηνύματα που λαμβάνει ο server δεν αποθηκεύονται στο σύστημα. Ωστόσο, το σύστημα που αναπτύξαμε απαιτούσε την αποθήκευση των δεδομένων για την επεξεργασία και τη γραφική απεικόνισή τους. Η απαίτηση αυτή ικανοποιείται με τον κώδικα broker2influx.py. Ο κώδικας αυτός μεταφέρει τα δεδομένα όλων των μηνυμάτων που λαμβάνει ο broker του συστήματος στη βάση δεδομένων testLoRapot που δημιουργήσαμε.

Ο κώδικας **broker2influx.py**

Ο παρακάτω κώδικας είναι γραμμένος σε python, και η λειτουργία του είναι η εξής:

- Χρησιμοποιώντας τη βιβλιοθήκη **paho.mqtt.client as mqtt** κάνει subscribe στον broker του συστήματος (πόρτα 1883)
- Χρησιμοποιώντας τη βιβλιοθήκη **argparse** αναλύει τα μηνύματα που λαμβάνει (parsing) προκειμένου να συγκρατήσει την επιθυμητή πληροφορία. Σημειώνεται ότι η μορφή μηνύματος διαφέρει από συσκευή σε συσκευή, οπότε η ανάγκη για την ανάλυση του μηνύματος είναι επιτακτική προκειμένου κάθε φορά να αποθηκεύεται η επιθυμητή πληροφορία. Στη περίπτωση μας τα δεδομένα που αποθηκεύονται είναι τα εξής:
    1. Appname, το όνομα της εφαρμογής στην οποία ανήκει το μήνυμα.
    2. Devname, το όνομα της συσκευής που στέλνει το μήνυμα.
    3. Devmac, η διεύθυνση mac.
    4. Counter, ο μετρητής κάθε μηνύματος.
    5. Frequency, η συχνότητα με την οποία εκπέμπει η κάθε συσκευή.
    6. Port, η πόρτα απο την οποία στάλθηκε το μήνυμα.
    7. Rssi, μέγεθος που υποδεικνύει την ισχύ του σήματος σε DBm.
    8. Snr, εκφράζει τον λόγο της ισχύος του σήματος προς την ισχύ του θορύβου.
    9. Payload0,…,Payload6 τα δεδομένα που στέλνει κάθε συσκευή.
- Το πεδίο payload, διαφέρει απο συσκευή σε συσκευή, για παράδειγμα μία συσκευή Arduino με dragino LoRa shield στέλνει ένα πεδίο payload σε κάθε μήνυμα ενώ μία συσκευή sodaqONE έχει τη δυνατότητα να στέλνει πολλαπλές μετρήσεις σε κάθε μήνυμα. Οι μετρήσεις αυτές έχουν ένα διαχωριστικό "/" μεταξύ τους. Για τον λόγο αυτό, ο κώδικας έχει προσαρμοστεί με τέτοιο τρόπο προκειμένου να αναγνωρίζει τη κάθε περίπτωση, να αντλεί όλα τα δεδομένα του εκάστοτε μηνύματος και να τα αποθηκέυει σε

διαφορετικά πεδία (float0,float1,....,float6). Επιπλέον ο κώδικας εξετάζει αν τα πεδία που αφορούν το payload είναι string ή float τιμές πριν προχωρήσει στην αποθήκευση. Παρότι οι string τιμές αποθηκεύονται στην influxDB δεν υπάρχει η δυνατότητα επεξεργασία τους με τη Grafana, οπότε μία τέτοια επιλογή θεωρείται ανούσια.

- Χρησιμοποιώντας τη βιβλιοθήκη base64, πραγματοποιήται αποκωδικοποίηση του πεδίου payload κάθε μηνύματος.
- Χρησιμοποιώντας τη βιβλιοθήκη **from influxdb import InfluxDBClient,** αποθηκεύουμε στην influxDB τα δεδομένα που επιθυμούμε. Στη συγκεκριμένη υλοποίηση εκτός από τα τα πεδία που αφορούν το payload (μπορεί να είναι μέχρι 7 πεδία για το payload) αποθηκεύουμε και τις τιμές που αφορούν το Appname, Devname, Devmac, Counter, Frequency, Port, Rssi και Snr.

Στα πεδία που ακολουθούν παρατίθονται 4 διαφορετικά μηνύματα από 4 διαφορετικές συσκευές προκειμένου να γίνει αντιληπτή η διαφορετικότητα των μηνυμάτων. Με έντονο χρώμα είναι πεδία που αποθηκεύονται στην InfluxDB.

{"app_id":"**mylora2**","dev_id":"**pir**","hardware_serial":"**0013506610138FCD**","port":**1**,"counte
r":**1057**,"payload_raw":"**Tk9BQ1RJT04=**","metadata":{"time":"2017-06-
22T11:19:41.642673047Z","frequency":**868.1**,"modulation":"LORA","data_rate":"SF7BW125",
"coding_rate":"4/5","gateways":[{"gtw_id":"eui-
b827ebfffe13bcbf","timestamp":4045066251,"time":"2017-06-
22T11:19:41.635978Z","channel":0,"rssi":-
53,"snr":**9.8**,"rf_chain":1,"latitude":38.04986,"longitude":23.78819,"altitude":6}]}}

{"app_id":"**sodtwo**","dev_id":"**temp**","hardware_serial":"**0069F9A658CBA662**","port":**1**,"count
er":**2710**,"payload_raw":"**U09EQVE=**","metadata":{"time":"2017-06-
22T11:21:02.455674204Z","frequency":**868.1**,"modulation":"LORA","data_rate":"SF7BW125",
"coding_rate":"4/5","gateways":[{"gtw_id":"eui-
b827ebfffe13bcbf","timestamp":4125877931,"time":"2017-06-
22T11:21:02.44881Z","channel":0,"rssi**":-
105**,"snr":**8**,"rf_chain":1,"latitude":38.04986,"longitude":23.78819,"altitude":6}]}}

{"app_id":"**sodone**","dev_id":"**gpstrack**","hardware_serial":"**0051AD3554311AE7**","port":**1**,"c

ounter":**119**,"payload_raw":"**msZLWdYUV/mtFknQLQ7wAAEAugQG**","metadata":{"time":"2017-06-22T13:31:03.454184247Z","frequency":**868.3**,"modulation":"LORA","data_rate":"SF7BW125","coding_rate":"4/5","gateways":[{"gtw_id":"eui-b827ebfffe13bcbf","timestamp":3336941915,"time":"2017-06-22T13:31:03.448771Z","channel":1,"rssi**":-63**,"snr":**10.2**,"rf_chain":1,"latitude":38.04986,"longitude":23.78819,"altitude":6}]}}

{"app_id":"**thomas**","dev_id":"**sensor1**","hardware_serial":"**00DAC3D70C5053F7**","port":**3**,"counter":337,"confirmed":true,"payload_raw":"**PD0+hgMjMjMxQjNGMDU3QzEwNTQyNCNuMSM1NCNUQzoyNS4zMSNIVU06MzMuNCNQUkVTOjk5MjI1LjY0Iw==**","metadata":{"time":"2017-06-22T11:17:35.620949344Z","frequency":**868.3**,"modulation":"LORA","data_rate":"SF7BW125","coding_rate":"4/5","gateways":[{"gtw_id":"eui-b827ebfffe13bcbf","timestamp":3919039427,"time":"2017-06-22T11:17:35.614406Z","channel":1,"rssi**":-96**,"snr":**7.8**,"rf_chain":1,"latitude":38.04986,"longitude":23.78819,"altitude":6}]}}
{"payload":"YGoOACYgHgNX3epX","message":{"app_id":"thomas","dev_id":"sensor1","port":0},"gateway_id":"eui-b827ebfffe13bcbf","config":{"modulation":"LORA","data_rate":"SF7BW125","counter":798,"frequency":868300000,"power":14}}

Ακολουθεί ο κώδικας broker2influx.py

```
#!/usr/bin/env python
# COMMAND =  sudo python /home/glimperop/Grafana-InfluxDB/influxdb-vimsen.py --dbuser "mqtt" --dbpwd "@vims3n!" --dbname "mqtt_db" --mqtthost "94.70.239.217" +/+/state/#
import argparse
import sys
import signal
from influxdb import InfluxDBClient
import json
import urllib
import logging
import paho.mqtt.client as mqtt
import requests
```

```
import base64

IP = "localhost"        # The IP of the machine hosting your influxdb instance
DB = "testLoRapot"              # The database to write to, has to exist
USER = "potamias"           # The influxdb user to authenticate with
PASSWORD = "@potamias@"  # The password of that user

# This holds the parsed arguments for the entire script
parserArgs = None
# This lets the message received event handler know that the DB connection is ready
dbConn = None
# This is the MQTT client object
client = None

def processArgs():
    '''This function processes command line arguments'''
    parser = argparse.ArgumentParser(description="This script will subscribe to messages
from an MQTT server and store the data in an influxdb server. Messages are assumed to
contain just the single value data to be saved. This script does not support connecting to SSL
MQTT servers (at the moment).")
    parser.add_argument("--dbhost", help="InfluxDB server name/IP (default localhost)",
default="localhost")
    parser.add_argument("--dbport", help="InfluxDB server port (default 8086)",
default=8086, type=int)
    parser.add_argument("--dbuser", help="InfluxDB user name (default None)",
default=None)
    parser.add_argument("--dbpwd", help="InfluxDB password (default None)",
default=None)
    parser.add_argument("--dbname", help="InfluxDB database name", default="mqtt")
    parser.add_argument("--dbseries", help="InfluxDB series to store data into (default
mqtt)", default="mqtt")
    parser.add_argument("--dbcolname", help="InfluxDB column name (default reading)",
default="reading")
    parser.add_argument("--mqtthost", help="MQTT server name/IP (default localhost)",
default="localhost")
    parser.add_argument("--mqttport", help="MQTT server port (default 1883)", type=int,
default=1883)
    parser.add_argument("--mqttuser", help="MQTT user name (default None)",
default=None)
    parser.add_argument("--mqttpwd", help="MQTT password (default None)",
default=None)
    parser.add_argument("--mqttqos", help="MQTT QoS value (default: 0)", type=int,
default=0)
    parser.add_argument("--mqttclient", help="MQTT Client ID (default: auto generated, sets
```

```
clean session to false)", default=None)
        parser.add_argument("topic", help="MQTT topic to susbcribe to (required)")
        parser.add_argument("--logfile", help="If specified, will log messages to the given file
(default log to terminal)", default=None)
        parser.add_argument("-v", help="Increase logging verbosity (can be used up to 5 times)",
action="count", default=0)
        return parser.parse_args()

def _sigIntHandler(signum, frame):
        '"This function handles Ctrl+C for graceful shutdown of the programme"'
        logging.info("Received Ctrl+C. Exiting.")
        stopMQTT()
        stopInfluxDB()
        exit(0)

def setupLogging():
        '"Sets up logging"'
        global parserArgs
        if parserArgs.v > 5:
                verbosityLevel = 5
        else:
                verbosityLevel = parserArgs.v
        verbosityLevel = (5 - verbosityLevel)*10
        if parserArgs.logfile is not None:
                logging.basicConfig(filename=parserArgs.logfile, level=verbosityLevel,
format='%(asctime)s %(message)s')
        else:
                logging.basicConfig(level=verbosityLevel, format='%(asctime)s %(message)s')

def setupSigInt():
        '"Sets up our Ctrl + C handler"'
        signal.signal(signal.SIGINT, _sigIntHandler)
        logging.debug("Installed Ctrl+C handler.")

def _mqttOnConnect(client, userdata, rc):
        '"This is the event handler for when one has connected to the MQTT broker. Will exit() if
connect is not successful."'
        global parserArgs
        if rc == 0:
                logging.info("Connected to MQTT broker successfully.")
                print("Connected to MQTT broker successfully.")
                print("Topic = " + parserArgs.topic)
                client.subscribe(parserArgs.topic, qos=parserArgs.mqttqos)
                startInfluxDB()
```

```
            return
        elif rc == 1:
            logging.critical("Connection to broker refused - incorrect protocol version.")
        elif rc == 2:
            logging.critical("Connection to broker refused - invalid client identifier.")
        elif rc == 3:
            logging.critical("Connection to broker refused - server unavailable.")
        elif rc == 4:
            logging.critical("Connection to broker refused - bad username or password.")
        elif rc == 5:
            logging.critical("Connection to broker refused - not authorised.")
        elif rc >=6 :
            logging.critical("Reserved code received!")
        client.close()
        exit(1)

def _mqttOnMessage(client, userdata, message):
        '''This is the event handler for a received message from the MQTT broker.'''
        logging.debug("Received message: " + str(message.payload))
        if dbConn is not None:
            #print(message.payload)
            #print(type(message.payload))
            sendToDB(message.payload)
def isfloat(value):
  try:
    float(value)
    return True
  except ValueError:
    return False



def sendToDB(payload):
        '''This function will transmit the given payload to the InfluxDB server'''
        global parserArgs
        writeData = dict()

        y=0
        words = str(payload).split(",")
        appid2 = words[0].split(":")
        drop2 = appid2[0]
        drop1 = drop2.replace("","")
        if drop1 == '{app_id':
         appid1 = appid2[1]
```

```
appid = appid1.replace("","")
if appid != 'sodone':
 devid2 = words[1].split(":")
 devid1 = devid2[1]
 devid = devid1.replace("","")
 devmac2 = words[2].split(":")
 devmac1 = devmac2[1]
 devmac = devmac1.replace("","")
 port2 = words[3].split(":")
 port1 = port2[1]
 port = port1.replace("","")
 counter2 = words[4].split(":")
 counter1 = counter2[1]
 counter = counter1.replace("","")
 checkfield3 = words[5].split(":")
 checkfield2 = checkfield3[0]
 checkfield1 = checkfield2.replace("","")

 if checkfield1 == 'confirmed':
     checkfield = checkfield3[1]
     confirm = checkfield.replace("","")
     secondcheck3 = words[6].split(':')
     secondcheck2 = secondcheck3[0]
     secondcheck1 = secondcheck2.replace("","")

     if secondcheck1 == 'is_retry':
        secondcheck = secondcheck3[1]
        is_retry = secondcheck.replace("","")
        payload2 = words[7].split(":")
        payload1 = payload2[1]
        payloadencoded = payload1.replace("","")
        payloadx = base64.b64decode(payloadencoded)
        frequency2 = words[9].split(":")
        frequency1 =  frequency2[1]
        frequency = frequency1.replace("","")
        data_rate2 = words[11].split(":")
        data_rate1 = data_rate2[1]
        data_rate = data_rate1.replace("","")
        coding_rate2 = words[12].split(":")
        coding_rate1 = coding_rate2[1]
        coding_rate = coding_rate1.replace("","")
        channel2 = words[16].split(":")
        channel1 = channel2[1]
        channel = channel1.replace("","")
```

```
            rssi2 = words[17].split(":")
            rssi1 = rssi2[1]
            rssi = rssi1.replace("","")
            snr2 = words[18].split(":")
            snr1 = snr2[1]
            snr = snr1.replace("","")

        else:
            payload2 = words[6].split(":")
            payload1 = payload2[1]
            payloadencoded = payload1.replace("","")
            payloadx = base64.b64decode(payloadencoded)
            frequency2 = words[8].split(":")
            frequency1 =  frequency2[1]
            frequency = frequency1.replace("","")
            data_rate2 = words[10].split(":")
            data_rate1 = data_rate2[1]
            data_rate = data_rate1.replace("","")
            coding_rate2 = words[11].split(":")
            coding_rate1 = coding_rate2[1]
            coding_rate = coding_rate1.replace("","")
            channel2 = words[15].split(":")
            channel1 = channel2[1]
            channel = channel1.replace("","")
            rssi2 = words[16].split(":")
            rssi1 = rssi2[1]
            rssi = rssi1.replace("","")
            snr2 = words[17].split(":")
            snr1 = snr2[1]
            snr = snr1.replace("","")



    else:
        payload2 = checkfield3[1]
        payloadencoded = payload2.replace("","")
        payloadx = base64.b64decode(payloadencoded)
        frequency2 = words[7].split(":")
        frequency1 =  frequency2[1]
        frequency = frequency1.replace("","")
        data_rate2 = words[9].split(":")
        data_rate1 = data_rate2[1]
        data_rate = data_rate1.replace("","")
        coding_rate2 = words[10].split(":")
```

```
        coding_rate1 = coding_rate2[1]
        coding_rate = coding_rate1.replace("","")
        channel2 = words[14].split(":")
        channel1 = channel2[1]
        channel = channel1.replace("","")
        rssi2 = words[15].split(":")
        rssi1 = rssi2[1]
        rssi = rssi1.replace("","")
        snr2 = words[16].split(":")
        snr1 = snr2[1]
        snr = snr1.replace("","")
    pay = [None, None, None, None, None, None, None]
    payx = ['', '', '', '', '', '', '']
    if "#" in payloadx:
     y = payloadx.count('#')
     payloadb = payloadx.split('#')
     n = 0
     for i in range(4,y):
        pay[i-4] = payloadb[i]
        n +=1
     for i in range(0,n):
        payx1 = pay[i].split(":")
        payx[i] = payx1[1]

     payloadx = payx[0]
     payload1 = payx[1]
     payload2 = payx[2]
     payload3 = payx[3]
     payload4 = payx[4]
     payload5 = payx[5]
     payload6 = payx[6]

     json_body = [
         {
          "measurement": "LoRaWAN",
           "tags": {
                "appname": str(appid),
                "devname": str(devid),
                "devmac": str(devmac)
                  },
            "fields": {
                 "counter": float(counter),
                 "frequency": float(frequency),
                 "port": float(port),
```

```
                "rssi": float(rssi),
                "snr": float(snr),
                "float0": str(payloadx),
                "float1": str(payload1),
                "float2": str(payload2),
                "float3": str(payload3),
                "float4": str(payload4),
                "float5": str(payload5),
                "float6": str(payload6)
                }
               }
               ]
    dbConn.write_points(json_body)

   else:

    payload1 = payx[0]
    payload2 = payx[1]
    payload3 = payx[2]
    payload4 = payx[3]
    payload5 = payx[4]
    payload6 = payx[5]
    json_body = [
        {
         "measurement": "LoRaWAN",
          "tags": {
                "appname": str(appid),
                "devname": str(devid),
                "devmac": str(devmac)
                  },
           "fields": {
                "counter": float(counter),
                "frequency": float(frequency),
                "port": float(port),
                "rssi": float(rssi),
                "snr": float(snr),
                "float0": str(payloadx),
                "float1": str(payload1),
                "float2": str(payload2),
                "float3": str(payload3),
                "float4": str(payload4),
                "float5": str(payload5),
                "float6": str(payload6)
```

```
                }
                }
                ]
        dbConn.write_points(json_body)


def startInfluxDB():
        '''This function sets up our InfluxDB connection'''
        global dbConn
        global parserArgs
        try:
                dbConn = InfluxDBClient(parserArgs.dbhost, parserArgs.dbport, parserArgs.dbuser,
parserArgs.dbpwd, parserArgs.dbname)
                logging.info("Connected to InfluxDB.")
                print("Connected to InfluxDB.")
        except InfluxDBClientError as e:
                logging.critical("Could not connect to Influxdb. Message: " + e.content)
                stopMQTT()
                exit(1)
        except:
                logging.critical("Could not connect to InfluxDB.")
                stopMQTT()
                exit(1)

def stopInfluxDB():
        '''This functions closes our InfluxDB connection'''
        dbConn = None
        logging.info("Disconnected from InfluxDB.")

def startMQTT():
        '''This function starts the MQTT connection and listens for messages'''
        global client
        if parserArgs.mqttclient is not None:
                client = mqtt.Client(parserArgs.mqttclient, clean_session=False)
        else:
                client = mqtt.Client()
        client.on_connect = _mqttOnConnect
        client.on_message = _mqttOnMessage
        if parserArgs.mqttuser is not None:
                client.username_pw_set(parserArgs.mqttuser, parserArgs.mqttpwd)
        client.connect(parserArgs.mqtthost, parserArgs.mqttport, 60)
        #client.loop_start()
```

```
def stopMQTT():
	'''This function stops the MQTT client service'''
	global client
	if client is not None:
		#client.loop_stop()
		client.disconnect()
		logging.info("Disconnected from MQTT broker.")
		client = None
	else:
		logging.warning("Attempting to disconnect without first connecting to MQTT
broker.")

def main():
	# Process our command line arguments first
	global parserArgs
	parserArgs = processArgs()

	# Now setup logging
	setupLogging()

	# Setup our interrupt handler
	setupSigInt()

	# Open up a connection to our MQTT server and subscribe
	startMQTT()

	# Stay here
	global client
	while True:
		client.loop()

	# If something go wrong stop them
	stopInfluxDB()
	stopMQTT()

if __name__ == "__main__":
	main()
```

Η διαδικασία αποθήκευσης των δεδομένων ξεκινάει εκτελώντας τη παρακάτω εντολή:

sudo python /home/potamias/broker2influx.py --dbuser "potamias" --dbpwd "@potamias@" --dbname "testLoRapot" --mqtthost "94.70.239.210" "#"

4.7 GRAFANA

Το Grafana είναι μία ανοιχτή πλατφόρμα για ανάλυση δεδομένων και δημιουργία γραφικών παραστάσεων. Χρησιμοποιείται κυρίως για τη γραφική απεικόνιση δεδομένων που αφορούν εφαρμογές IoT. Το grafana έχει τη δυνατότητα σύνδεσης με μια πλειάδα πηγών δεδομένων όπως InfluxDB, Elasticsearch, Graphite κ.α. Στο σύστημα που αναπτύξαμε χρησιμοποιήσαμε την InfluxDB σαν πηγή δεδομένων όπως αναφέρθηκε παραπάνω.

4.7.1 Διαδικασία εγκατάστασης Grafana:

- Ανοίγουμε το αρχείο sources.list εκτελώντας sudo vi /etc/apt/sources.list και προσθέτουμε το παρακάτω repository
  deb https://packagecloud.io/grafana/stable/debian/ jessie main
- curl https://packagecloud.io/gpg.key | sudo apt-key add -
- sudo apt-get update
- sudo apt-get install grafana
- Τέλος, θέτουμε σε λειτουργία τη grafana sudo service grafana-server start

4.7.2 Ρύθμιση Grafana:

Η Grafana ακούει στη πόρτα 3000.

1. Χρησιμοποιούμε κάποιο browser και πλοηγούμαστε στη διεύθυνση 94.70.239.210:300 όπου 94.70.239.210 η διεύθυνση IP που φιλοξενεί τη grafana
2. Κάνουμε κλικ στο κουμπί grafana προκειμένου να ανοίξει η αριστερή στήλη του μενού.
3. Κάνουμε κλικ στο κουμπί Data sources.
4. Στη συνέχεια επιλέγουμε το κουμπί Add new.
5. Επιλέγουμε κάποιο όνομα για το dashboard και επιλέγουμε Type: InfluxDB
6. Στο πλαίσιο Http settings στο πεδίο URL πληκτρολογούμε την IP διεύθυνση του host που φιλοξενεί την Influxdb, στη περίπτωσή μας 94.70.239.210.
7. Στο πεδίο Acces επιλέγουμε proxy.
8. Στο πλαίσιο Influxdb Details συμπληρώνουμε τα πεδία Database, User, Password.
9. Επιλέγουμε test connection προκειμένου να ελεξουμε την σύνδεση με τη βάση δεδομένων και στη συνέχεια επιλέγουμε το κουμπί save.

4.7.3 Δημιουργία dashboard:

Για τη δημιουργία dashboards προκειμένου να δημιουργηθούν γραφικές παραστάσεις των δεδομένων ακολουθούμε τα παρακάτω βήματα

1. Κάνουμε κλικ στην επιλογή Home και στο παράθυρο που θα εμφανιστεί επιλέγουμε New.
2. Στη νέα σελίδα που θα μεταφερθούμε επιλέγουμε το πράσινο χρώμα και κάνουμε κλικ στην επιλογή Add Panel -> Graph
3. Στο Graph που θα δημιουργηθεί, επιλέγουμε το πλαίσιο General και συμπληρώνουμε το πεδίο Title με τίτλο της επιλογής μας.
4. Στο πλαίσιο Metrics επιλέγουμε σαν datasource τη πηγή που δημιουργήσαμε παραπάνω.
5. Τέλος δημιουργούμε το Query. Στη περίπτωσή μας το Query που δημιουργήσαμε είναι το παρακάτω :

"query": "SELECT mean(\"float0\") AS \"temperature\", mean(\"float1\") AS \"humidity\", mean(\"float2\") AS \"pressure\", mean(\"frequency\") AS \"frequency\", mean(\"rssi\") AS \"rssi\", mean(\"snr\") AS \"snr\" FROM \"LoRaWAN\" WHERE \"appname\" = 'thomas' AND $timeFilter GROUP BY time($interval) fill(null)",

Αποθηκεύοντας το dashboard, το Grafana δημιουργεί γραφικές παραστάσεις.

4.7.4 Scripted Grafana

Στα πλαίσια της εργασίας αναπτύχθηκε ένα web interface μέσω του οποίου ο χρήστης έχει τη δυνατότητα να καταχωρήσει τις συσκευές του προκειμένου να λαμβάνει μηνύματα και να παρακολουθεί τα δεδομένα μέσα από γραφικές παραστάσεις μέσω της Grafana. Η λειτουργία αυτή γίνεται αυτοματοποιημένα, χωρίς να απαιτείται από τον κάθε χρήστη να ακολουθήσει τη διαδικασία που αναφέρθηκε παραπάνω για τη δημιουργία των dashboard. Η υλοποίηση αυτή επιτεύχθηκε δημιουργώντας με τη λειτουργία scripted grafana που παρέχει η grafana.

Πρόκειτα για json scripts τα οποία λαμβάνουν χώρα σε συγγεκριμένο directory μέσα στο host που φιλοξενεί τη grafana τα οποία δημιουργούν δυναμικά dashboards με παραμέτρους που μεταφέρονται μέσω url.

Στην υλοποίηση μας διαμορφώσαμε με τέτοιο τρόπο τα Querys που πραγματοποιεί η Grafana στην Influxdb ώστε τα dashboard να ξεχωρίζουν ως προς το όνομα της εφαρμογής (Appname). Όλα τα dashboard έχουν κοινή δομή που περιγράφεται από το παρακάτω κώδικα με μοναδική διαφοροποίηση το όνομα εφαρμογής, δίνοντας έτσι στο σύστημα τη δυνατότητα να δημιουργεί δυναμικά dashboards για το χρήστη.

Ακολουθεί το javascript που ενεργοποιείται στην php σελίδα του web interface, για τη μεταφορά παραμέτρων (Appname) στο scripted grafana μέσω url:

```
<script type="text/javascript">
   document.getElementById("button4").onclick = function () {
      location.href =
"http://94.70.239.210:3001/dashboard/script/testpotamias.js?rows=1&name=<?php echo
$appnameselect;?>";
   };
    </script>
```

- * Όπου appnameselect == Appname
- ** testpotamias.js το js script για τη δημιουργία του dashboard

Ακολουθεί ο κώδικας **testpotamias.js** που βρίσκεται στο path:
/usr/share/grafana/public/dashboards

```
'use strict';


var window, document, ARGS, $, jQuery, moment, kbn;


var dashboard;

// All url parameters are available via the ARGS object
var ARGS;


dashboard = {
```

```
  rows : [],
};

// Set a title
dashboard.title = 'Scripted dash';


dashboard.time = {
  from: "now-6h",
  to: "now"
};

var rows = 1;
var seriesName = 'argName';

if(!_.isUndefined(ARGS.rows)) {
  rows = parseInt(ARGS.rows, 10);
}

if(!_.isUndefined(ARGS.name)) {
  seriesName = ARGS.name;
}

for (var i = 0; i < rows; i++) {

  dashboard.rows.push({
    title: 'Chart',
    height: '300px',
    panels: [

          {
        "aliasColors": {},
        "bars": false,
        "datasource": "testLoRapot2",
        "editable": true,
        "error": false,
        "fill": 1,
        "grid": {
          "leftLogBase": 1,
          "leftMax": null,
          "leftMin": null,
          "rightLogBase": 1,
          "rightMax": null,
          "rightMin": null,
```

```
  "threshold1": null,
  "threshold1Color": "rgba(216, 200, 27, 0.27)",
  "threshold2": null,
  "threshold2Color": "rgba(234, 112, 112, 0.22)"
},

"isNew": true,
"legend": {
  "avg": false,
  "current": false,
  "max": false,
  "min": false,
  "show": true,
  "total": false,
  "values": false
},
"lines": true,
"linewidth": 2,
"links": [],
"nullPointMode": "connected",
"percentage": false,
"pointradius": 5,
"points": false,
"renderer": "flot",
"seriesOverrides": [],
"span": 12,
"stack": false,
"steppedLine": false,
"targets": [
  {
    "dsType": "influxdb",
    "groupBy": [
      {
        "params": [
          "$interval"
        ],
        "type": "time"
      },
      {
        "params": [
          "null"
        ],
        "type": "fill"
      }
```

```
        ],
        "measurement": "LoRaWAN",
        "query": "SELECT mean(\"float0\") AS \"temperature\", mean(\"frequency\") AS
\"frequency\", mean(\"rssi\") AS \"rssi\", mean(\"snr\") AS \"snr\" FROM \"LoRaWAN\"
WHERE \"appname\" = seriesName AND $timeFilter GROUP BY time($interval) fill(null)",
        "refId": "A",
        "resultFormat": "time_series",
        "select": [
         [
          {
           "params": [
            "float0"
           ],
           "type": "field"
          },
          {
           "params": [],
           "type": "mean"
          },
          {
           "params": [
            "float0"
           ],
           "type": "alias"
          }
         ],

         [
          {
           "params": [
            "float1"
           ],
           "type": "field"
          },
          {
           "params": [],
           "type": "mean"
          },
          {
           "params": [
            "float1"
           ],
           "type": "alias"
          }
```

```
        ],
        [
          {
            "params": [
              "float2"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          },
          {
            "params": [
              "float2"
            ],
            "type": "alias"
          }
        ],

        [
          {
            "params": [
              "frequency"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          },
          {
            "params": [
              "frequency"
            ],
            "type": "alias"
         }
        ],
        [
          {
            "params": [
              "rssi"
            ],
            "type": "field"
```

```
          },
          {
            "params": [],
            "type": "mean"
          },
          {
            "params": [
              "rssi"
            ],
            "type": "alias"
          }
        ],
        [
          {
            "params": [
              "snr"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          },
          {
            "params": [
              "snr"
            ],
            "type": "alias"
          }
        ]
      ],
      "tags": [
        {
          "key": "appname",
          "operator": "=",
          "value": seriesName
        }
      ]
    }
  ],
  "timeFrom": null,
  "timeShift": null,
  "title": seriesName,
  "tooltip": {
```

```
          "shared": true,
          "value_type": "cumulative"
        },
        "type": "graph",
        "x-axis": true,
        "y-axis": true,
        "y_formats": [
          "short",
        ]
      }
    ]
  });
}


return dashboard;
```

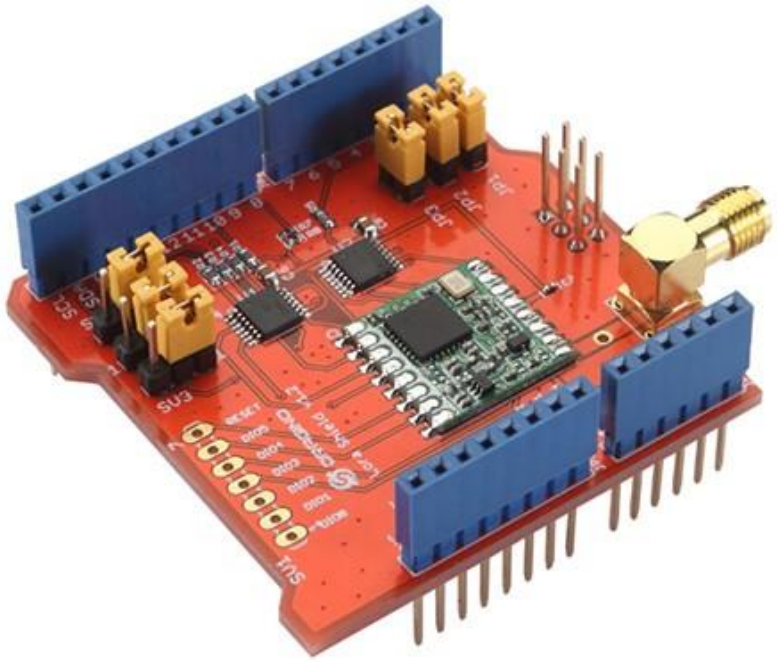
5. End Devices

Στο κεφάλαιο αυτό θα περιγράψουμε τη λειτουργία των συσκευών που μεταδίδουν τη πληροφορία που αντλούν από τους αισθητήρες. Οι συσκευές αυτές για να επικοινωνήσουν με το υπόλοιπο LoRa σύστημα πρέπει να διαθέτουν τον κατάλληλο hardware εξοπλισμό προκριμένου το σήμα που εκπέμπεται να επιτύχει τη κατάλληλη διαμόρφωση. Μέχρι σήμερα έχουν αναπτυχθεί αρκετές πλακέτες που υποστηρίζουν LORA, μερικές είναι :

- Arduino Uno LoRa shield
- Dragino LoRa shield
- Froggy Factory LoRa Shield
- Sodaq LoRabee
- Adafruit RFM95W Breakout 868Mhz
- Adafruit RFM98W Breakout 433Mhz

Στα πλαίσια της παρούσας διπλωματικής, χρησιμοποιήθηκαν 5 συσκευές. 3 Dragino LoRa και 2 Sodaq one LoRa shield.

5.1 DRAGINO LORA SHIELD



ΕΙΚΟΝΑ 4: Dragino LoRa Shield

Το Dragino LoRa shiled είναι μία πλακέτα συμβατή με arduino πλακέτα και αποτελεί πομπό μεγάλου εύρους και χαμηλού data rate βασισμένο στο chip SX1276/77/78/79 της Semtech. Τα βασικά χαρακτηριστικά του dragino LoRa shield είναι τα παρακάτω:

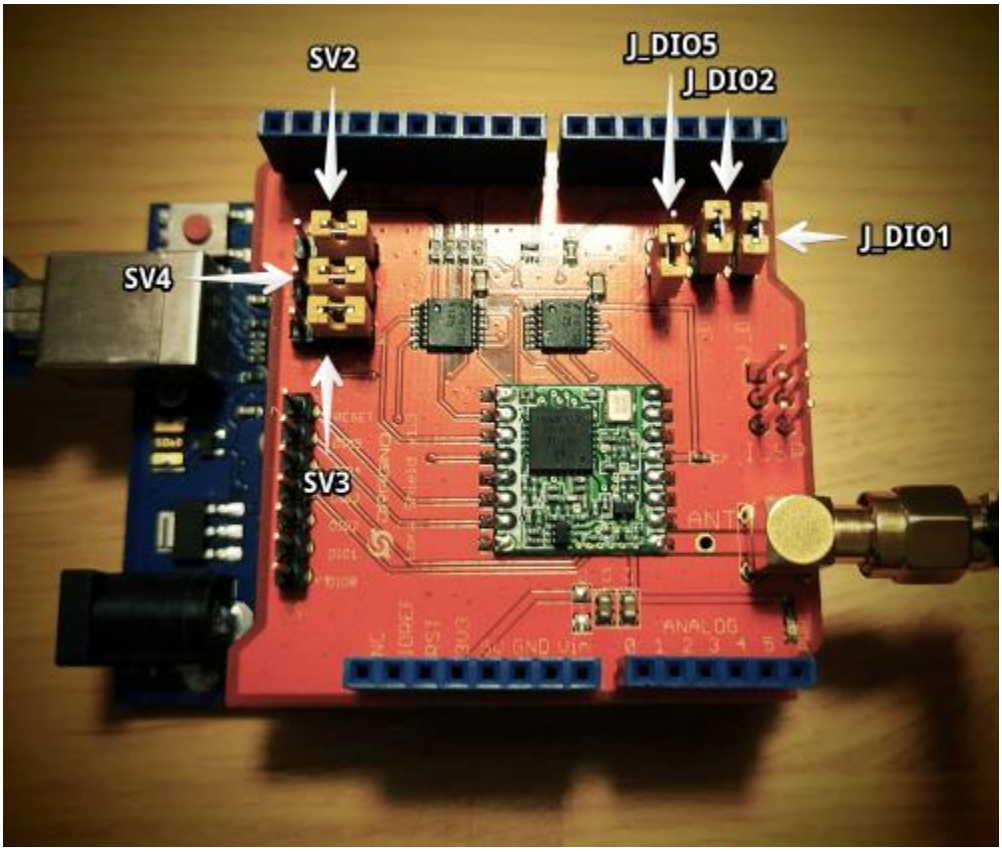
- 168 dB maximum link budget
- Είναι συμβατό με Arduino (Leonardo, UNO, MEGA, DUE)
- Ζώνη συχνοτήτων: 915MHz/868MHz (προκαθορισμένη στο εργοστάσιο)
- Bit rate μέχρι 300 kbps
- Διαστάσεις 62x43mm
- Βάρος 22g

Όπως αναφέρθηκε παραπάνω το dragino LoRa shield είναι συμβατό με τον μικροελεγκτή Arduino. Στο σύστημά μας χρησιμοποιήθηκαν 3 arduino πλακέτες οι οποίες συνδέθηκαν με dragino LoRa πλακέτες.

Η διαδικασία που ακολουθήθηκε για τη δημιουρια ενός κόμβου με dragino LoRa shield + Arduino microcontroller είναι η παρακάτω:

- Συνδέουμε τη πλακέτα LoRa shield πάνω στον μικροελεγκτή.
- Συνδέουμε τη κεραία.
- Ρυθμίζουμε τους jumpers όπως παρακάτω :

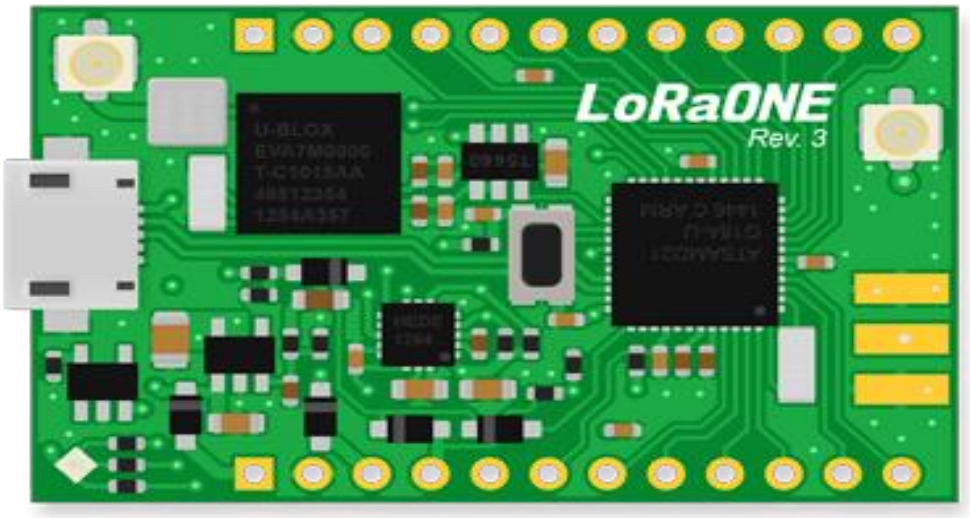
SV2, SV3 και SV4	**δεξιά θέση**
J_DIO1 και J_DIO2	**κλειστοί**
J_DIO5	**ανοιχτός**



EIKONA 5: Arduino + Dragino LoRa Shield

- Αφού ολοκληρώσουμε τη παραπάνω διαδικασία συνδέουμε με ένα USB καλώδιο τον μικροελεγκτή σε έναν υπολογιστή με εγκατεστημένο το Arduino IDE.
- Ανοίγουμε την εφαρμογή του Arduino και επιλέγουμε τη πλακέτα που χρησιμοποιούμε (π.χ Arduino/Genuino Uno) πηγαίνοντας στο tools->board.
- Επιλέγουμε τη θύρα στην οποία συνδέσαμε τη πλακέτα πηγαίνοντας tools->port
- Κατεβάζουμε την arduino-lmic βιβλιοθήκη από τον παρακάτω σύνδεσμο: https://github.com/matthijskooijman/arduino-lmic και την αποθηκεύουμε στον φάκελο Libraries που βρίσκεται μέσα στο Arduino directory. Πλέον μπορούμε να προγραμματίσουμε τη συσκευή.
- Επισκεπτόμαστε το site 94.70.239.210:8001/potamias για να δημιουργήσουμε κάποια εφαρμογή και να αποκτήσουμε τα κατάλληλα κλειδιά για την ορθή επικοινωνία της συσκευής με το υπόλοιπο σύστημα (NWSKEY, DEVADDR, APPSKEY). Αυτή η διαδικασία θα περιγραφεί αναλυτικά στο επόμενο κεφάλαιο

5.2 Sodaq LoRaONE



ΕΙΚΟΝΑ 6: Sodaq LoRaOne

Το LoRaONE της Sodaq αποτελεί μία από τις καλύτερες λύσεις για εφαρμογές LoRa καθώς. Είναι μικρό ισχυρό, οικονομικό και διαθέτει όλα τα χαρακτηριστικά που θέλει κάποιος προκειμένου να στήσει ένα LoRa Device.

Τα χαρακτηριστικά του LoRaONE:

- Microcontroller: ATSAMD21G18, 32-Bit ARM Cortex, M0+
- Είναι συμβατό με Arduino
- Διαστάσεις: 40x25 mm
- Τροφοδοσία: 3.3 V
- 14 I/O pins που χρησιμοποιούνται για αισθητήρες.
- Flash Memory: 256 Kb
- Clock speed: 48Mhz

- Διαθέτει solar charge controller (μέχρι 500mA)
- Διαθέτει GPS προκειμένου να εντοπίζεται με ακρίβεια η θέση του.
- Διαθέτει microUSB θύρα

Η διαδικασία που ακολουθήθηκε για τη δημιουργια ενός κόμβου sodaq LoRaONE είναι η παρακάτω:

- Από τον παρακάτω σύνδεσμο προμηθευόμαστε το kit με τον εξοπλισμό του sodaq LoRaOne:

https://www.kickstarter.com/projects/sodaq/LoRaone-the-LoRa-iot-development-board

ΕΙΚΟΝΑ 7: Sodaq LoRaOne kit

- Συνδέουε τη πλακέτα με τον υπολογιστή μέσω USB καλωδίου.
- Ανοίγουμε το Arduino IDE και κάνουμε κλικ στο File-> Preferences-> Additional Board Manager URLs.
- Κάνουμε επικόλληση το ακόλουθο url, *http://downloads.sodaq.net/package_sodaq_samd_index.json*
- Κάνουμε κλικ στο Tools-> Board-> Boards Manager και επιλέγουμε το SODAQ SAMD boards προκειμένου να εγκατασταθούν τα απαραίτητα αρχεία για τη πλακέτα.
- Κάνουμε κλικ στο Tools-> Board-> SODAQ ONE , πλέον μπορούμε να προγραμματίσουμε τη συσκευή.
- Επισκεπτόμαστε το site 94.70.239.210:8001/potamias για να δημιουργήσουμε κάποια εφαρμογή και να αποκτήσουμε τα κατάλληλα κλειδιά για την ορθή επικοινωνία της συσκευής με το υπόλοιπο σύστημα (NWSKEY, DEVADDR, APPSKEY). Αυτή η διαδικασία θα περιγραφεί αναλυτικά στο επόμενο κεφάλαιο

6. Web Interface

6.1 Παρουσίαση του Web Interface του συστήματος

Στο κεφάλαιο αυτό, θα περιγραφεί το web interface που αναπτύχθηκε για τη διαχείριση του συστήματος, αλλά και τη γραφικοποίηση των μετρήσεων μέσω του grafana

Ο χρήστης αφού κάνει register και login, έχει τη δυνατότητα να

- Να κάνει εγγραφή εφαρμογών.
- Να κάνει καταχώρηση συσκευών.
- Να παρακολουθήσει στη κονσόλα τη ροή των μηνυμάτων σε πραγματικό χρόνο.
- Να παρακολουθήσει τα γραφικοποιημένα δεδομένα κάθε συσκευής μέσω των scripted grafana dashboards που δημιουργούνται δυναμικά για κάθε χρήστη

Όλες οι παραπάνω λειτουργίες εκτελούνται στον server με το εργαλείο ttnctl που παρέχει το backend του The Things Network. Ο χρήστης ωστόσο, διαχειρίζεται το σύστημα με τρόπο απλοικό και κατανοητό όπως θα εξηγηθεί στη συνέχεια.

Οι php κώδικες για το web interface βρίσκονται στον server με

IP διεύθυνση: 94.70.239.210

στο path: /var/www/html/potamias/

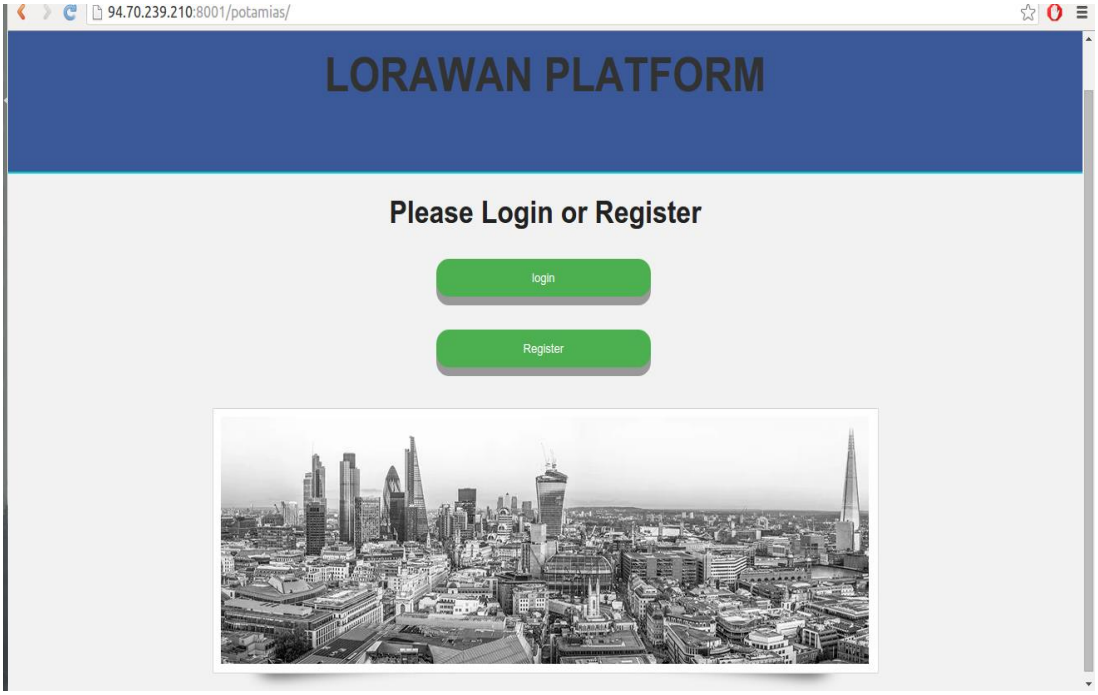
στη port: 8001

Για την υλοποίηση του web interface χρησιμοποιήθηκαν: php, html, css, javascrip και mysql

Ανοίγοντας ο χρηστης έναν οποιονδήποτε browser και πληκτρολογώντας τη διεύθυνση:

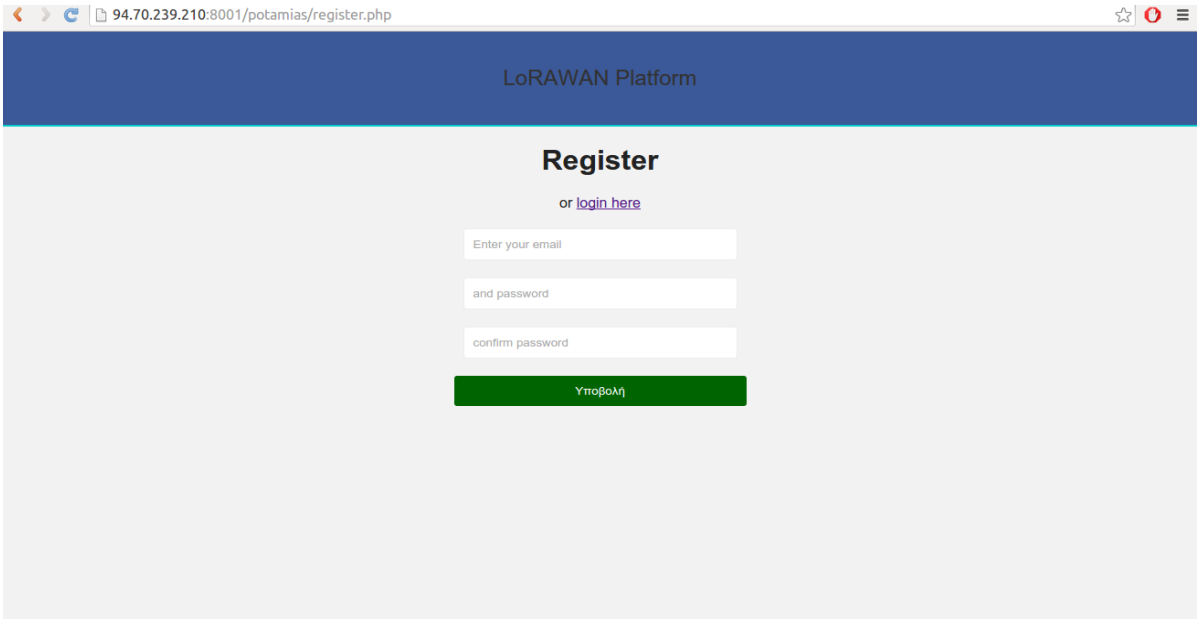
94.70.239.210:8001/potamias/

θα πλοηγηθεί στη παρακάτω σελίδα



ΕΙΚΟΝΑ 8: Σελίδα υποδοχής

Πατώντας το κουμπί register, θα μεταφερθεί σε μία φόρμα στην οποία θα κληθεί να καταχωρήσει το email του και έναν κωδικό.

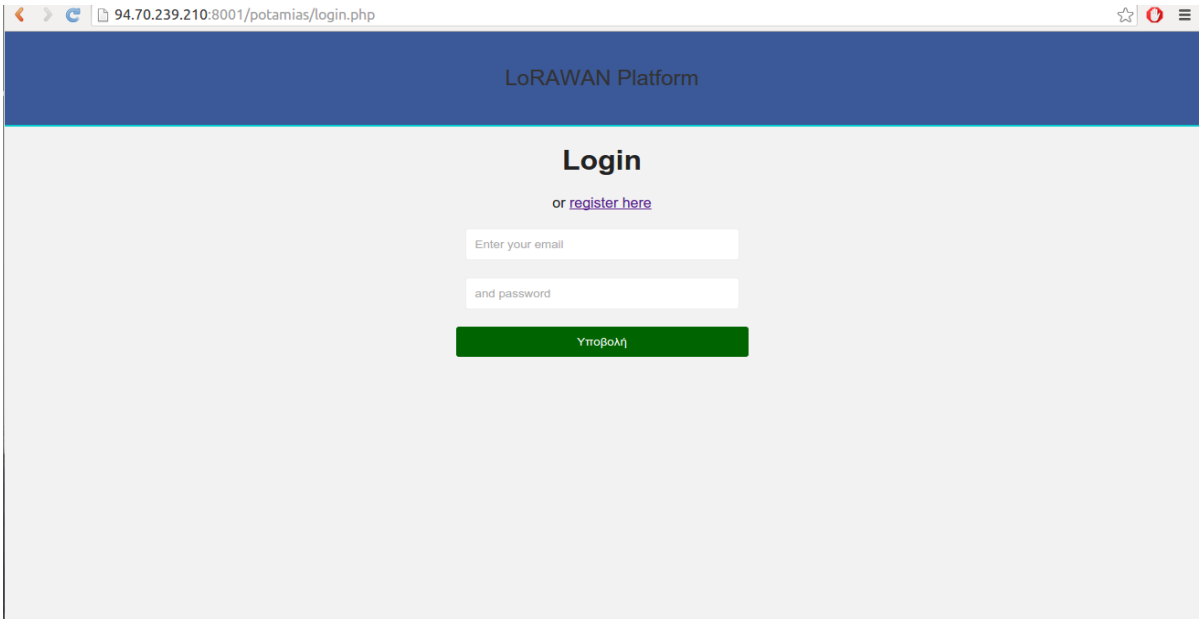


ΕΙΚΟΝΑ 9: Σελίδα register

Αφού καταχωρήσει τα στοιχεία του, και πατώντας το κουμπί login θα μεταφερθεί στη σελίδα για το login. Εκεί πρέπει να βάλει το email και το password που καταχώρησε πριν. Το σύστημα θα ελέγξει την εγκυρότητα των στοιχείων του χρήστη στη mysql βάση που αναπτύχθηκε, κι αν είναι έγκυρα θα τον εισάγει στη κεντρική σελίδα του site, αν τα στοιχεία δεν είναι έγκυρα θα εκτυπώσει το μήνυμα " Sorry, those credentials do not match".

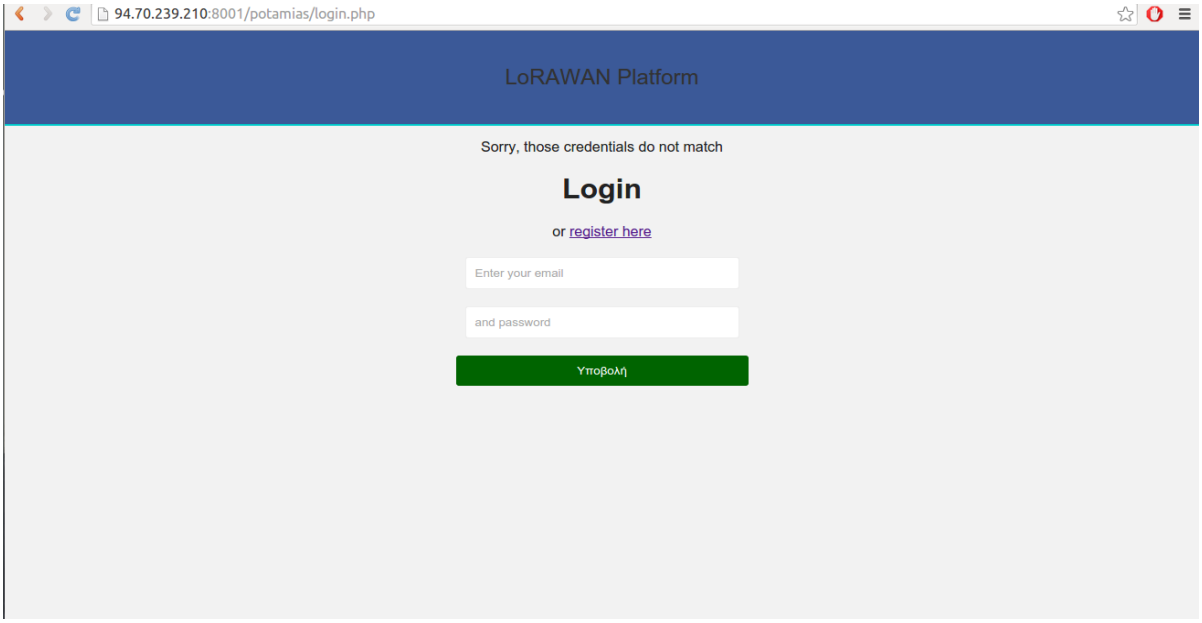
Ακολουθούν screenshots από τη σελίδα του login αρχικά, αλλά και μετά την εισαγωγή λανθασμένων στοιχείων.

Η σελίδα για το login:



ΕΙΚΟΝΑ 10: Σελίδα Login

Η σελίδα login μετά από λανθασμένη εισαγωγή στοιχείων:



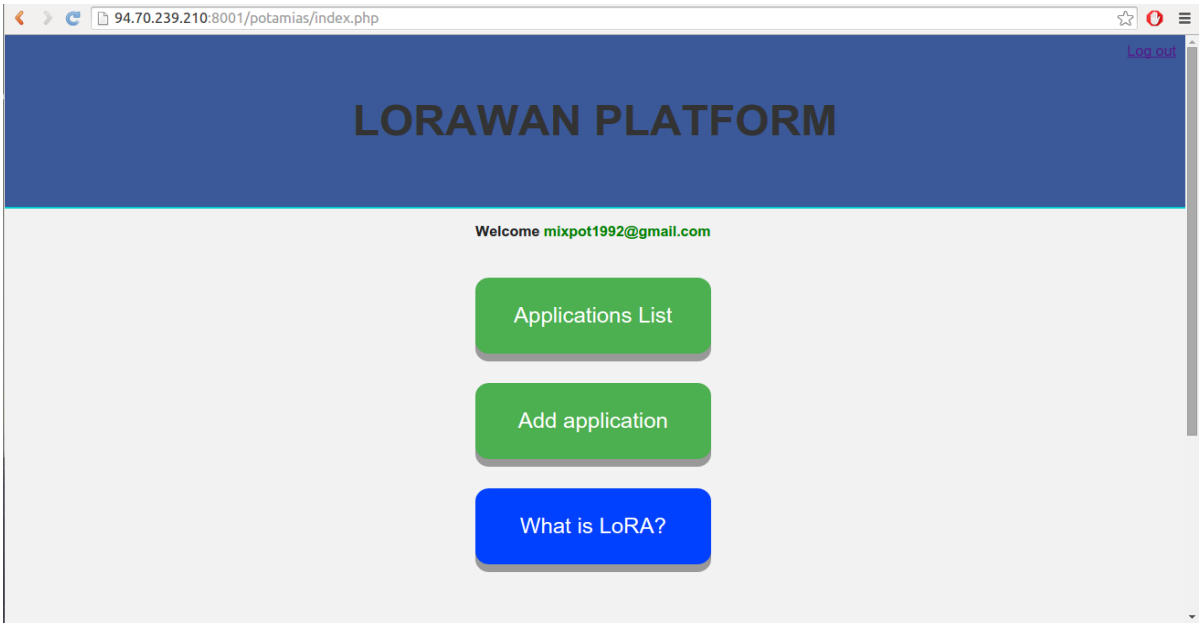
ΕΙΚΟΝΑ 11: Σελίδα λανθασμένου Login

Μετά την έγκυρη εισαγωγή στοιχείων ο χρήστης μεταφέρεται στη κεντρική σελίδα του site στην οποία του εμφανίζεται το μήνυμα "**Welcome** username@mail.com"

Στη σελίδα αυτή έχει τις εξής δυνατότητες:

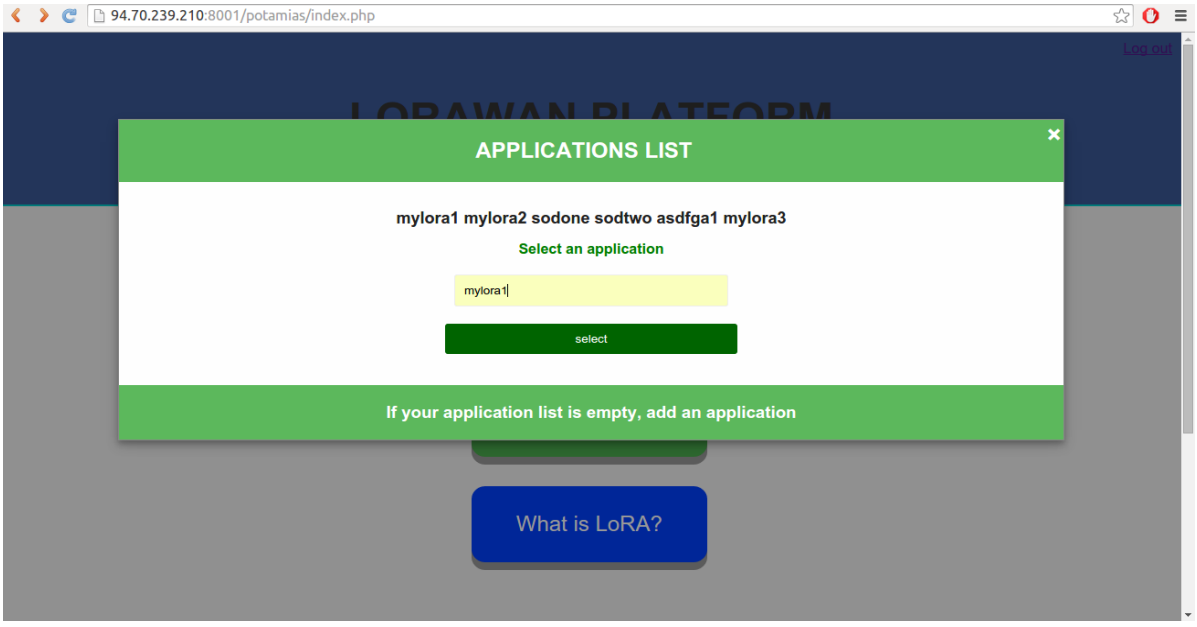
- **Applications List :** Πατώντας αυτό το κουμπί, θα εμφανιστεί ένα αναδυόμενο παράθυρο που θα εμφανίζει τα ονόματα των εφαρμογών που διαχειρίζεται ο χρήστης. Αν ο χρήστης δεν έχει ακόμα εφαρμογές τότε δεν θα εμφανίζεται κανένα όνομα στη λίστα. Επιπλέον μέσω αυτού του αναδυόμενου παραθύρου ο χρήστης μπορεί να πληκτρολογήσει στο πλαίσιο που βρίσκεται στο κάτω μέρος του παραθύρου το όνομα μιας εκ των εφαρμογών της λίστας προκειμένου να μεταφερθεί στη σελίδα διαχείρισης εφαρμογών.
- **Add Application :** Με το κουμπί αυτό ο χρήστης επιλέγει να προσθέσει κάποια εφαρμογή στη λίστα του και μεταφέρεται στη σελίδα προσθήκης εφαρμογών.
- **What is LoRa :** Η επιλογή αυτή, δίνει στον χρήστη την ευκαιρία να διαβάσει μια συνοπτική περιγραφή της τεχνολογίας LoRa αλλά και του LoRaWAN πρωτοκόλλου.
- **Logout :** το κουμπί στο πάνω δεξιά μέρος της οθόνης οδηγεί στην αποσύνδεση από το site.

Η κεντρική σελίδα του site:



ΕΙΚΟΝΑ 12: Κεντρική σελίδα

Το αναδυόμενο παράθυρο που εμφανίζεται μετά το πάτημα του κουμπιού **"Applications List"**



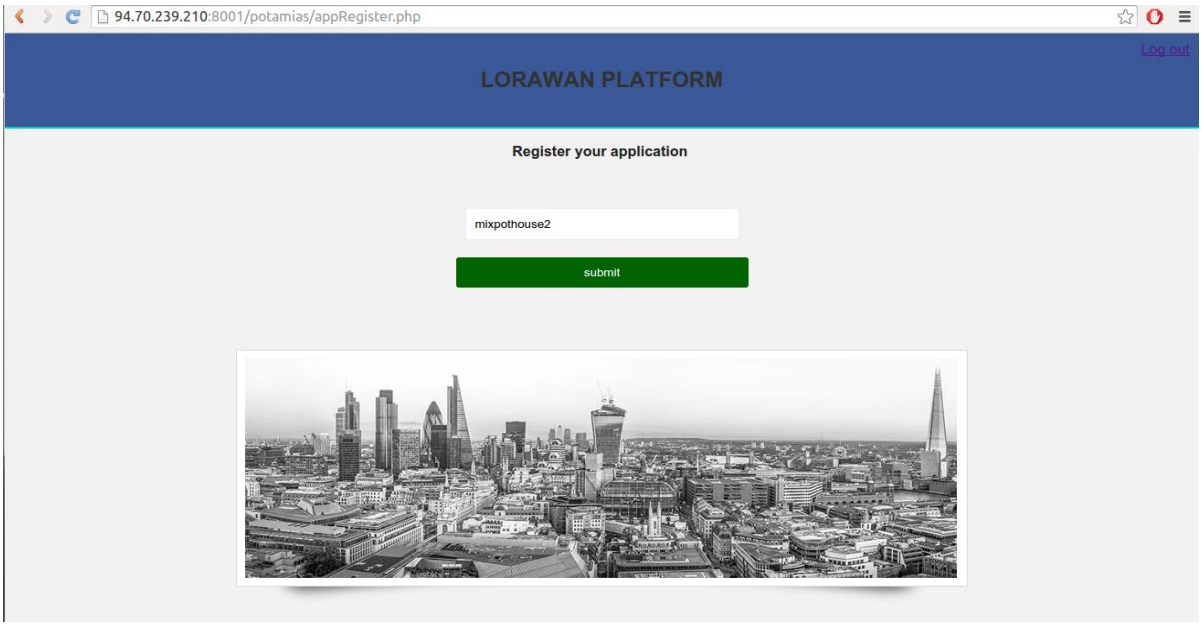
ΕΙΚΟΝΑ 13: Αναδυόμενο παράθυρο Applications List

Η σελίδα στην οποία μεταφέρεται ο χρήστης πατώντας το κουμπί **"What is LoRa?"**

ΕΙΚΟΝΑ 14: Σελίδα What is LoRa

Στη περίπτωση που ο χρήστης δεν έχει δημιουργήσει ακόμα κάποια εφαρμογή ή θελει να προσθέσει μία ακόμα θα διαλέξει την επιλογή **"Add Application"** όπως αναφέρθηκε παραπάνω. Το κουμπί αυτό θα τον οδηγήσει σε μία νέα σελίδα στην οποία θα του ζητηθεί να πληκτρολογήσει ένα όνομα για την εφαρμογή του και ύστερα να πατήσει το πλήκτρο submit.

Η σελίδα για τη προσθήκη εφαρμογή



ΕΙΚΟΝΑ 15: Σελίδα Προσθήκης εφαρμογής

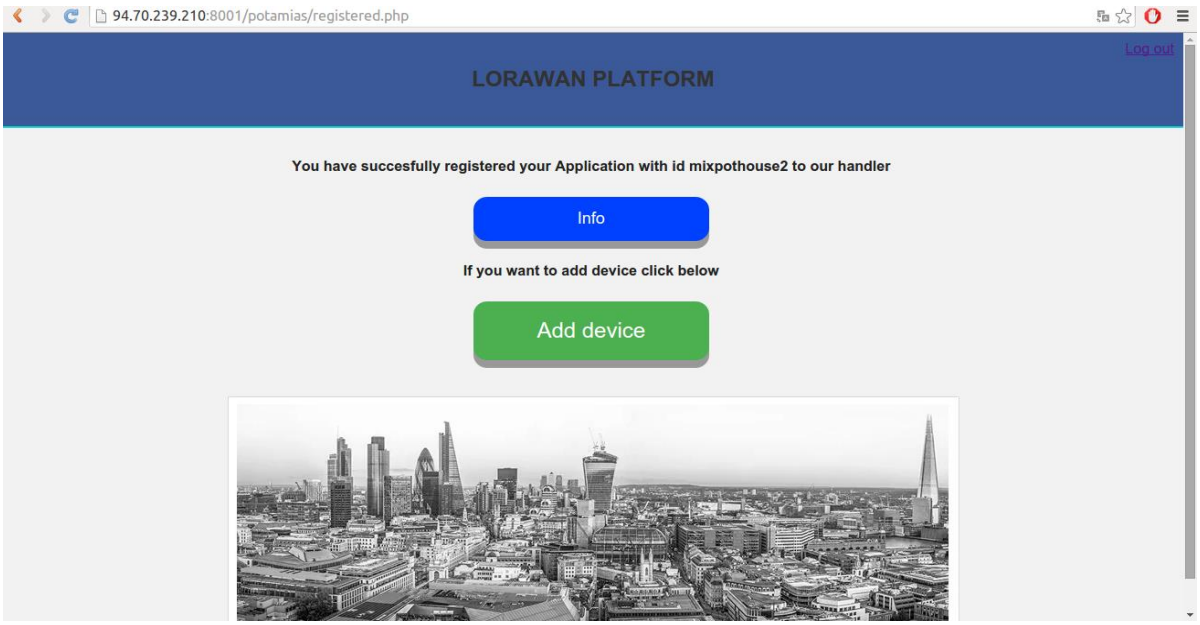
Πατώντας το πλήκτρο submit δίδεται η εντολή στο backend να εκτελέσει τη παρακάτω ακολουθία εντολών:

- **ttnctl application add mixpothouse2 “testin”**: Μία εφαρμογή με το όνομα mixpothouse2 και σχόλιο “testin”προστίθεται.
- **ttnctl application register :** Η εφαρμογή mixpothouse2 γίνεται register στον Handler του συστήματος

Ο χρήστης μεταφέρεται στην επόμενη σελίδα της νέας εφαρμογής στην οποία έχει τις παρακάτω επιλογές:

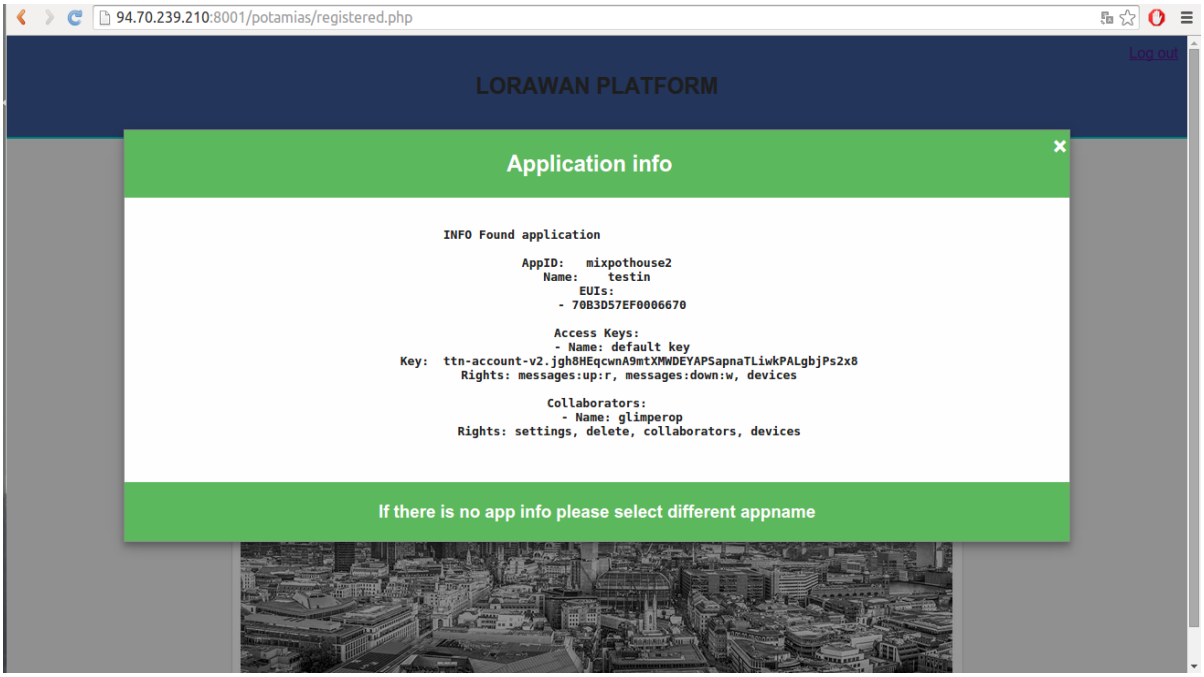
- **Application info :** Μπορεί να δει τις πληροφορίες της εφαρμογής του (Name, κλειδί EUI, acces key του ttn, rights, collaborator) και να επιβεβαιώσει ότι η εφαρμογή του έχει γίνει register σωστά. Σε περίπτωση που προκύψει κάποιο σφάλμα κατά τη προσθήκη της εφαρμογής τότε το μήνυμα σφάλματος θα εμφανιστεί στο παράθυρο. Ένα συχνό λάθος που μπορεί να προκύψει αφορά το όνομα της εφαρμογής, καθώς το backend του συστήματος απαγορεύει τη χρήση κεφαλαίων και κενών στην ονομασία των εφαρμογών, επιπλέον αποτρέπει σε κάποια εφαρμογή να έχει όνομα που ήδη χρησιμοποιείται από κάποια άλλη εφαρμογή ακόμα και άλλου χρήστη.
- **Device add :** Μετά την ορθή δημιουργία μίας εφαρμογής ο χρήστης πρέπει να προσθέσει κάποιο device στην εφαρμογή αυτή. Με το πλήκτρο αυτό ο χρήστης οδηγείται στη σελίδα προσθήκης device.

Η σελίδα της νέας εφαρμογής:



ΕΙΚΟΝΑ 16: Σελίδα νέας εφαρμογής

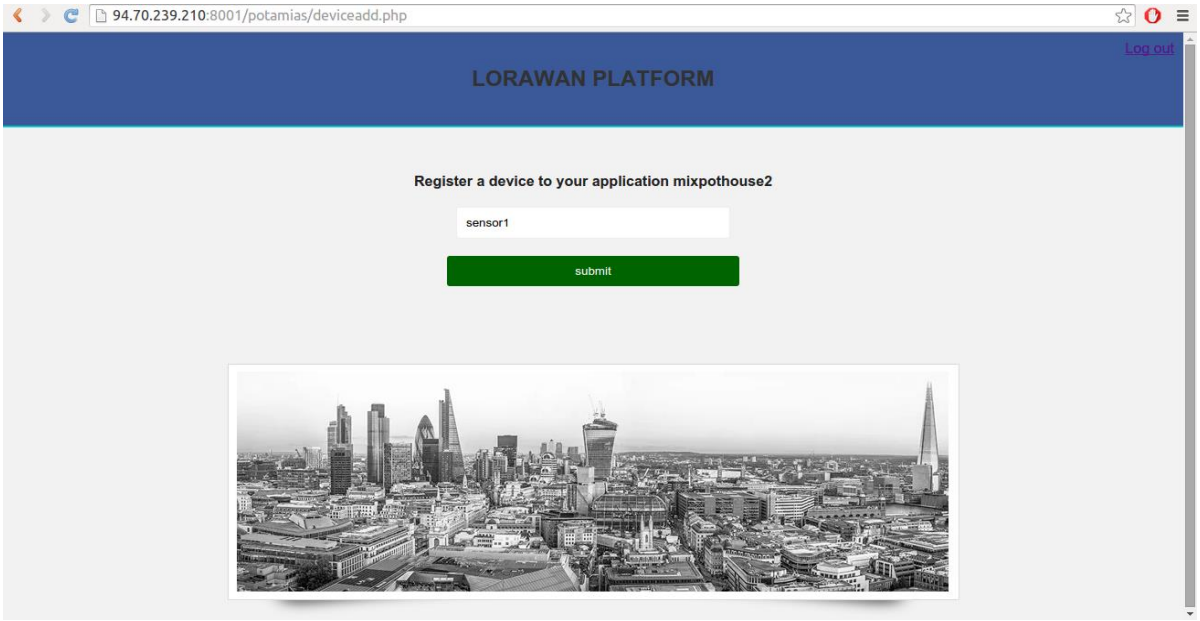
Το αναδυόμενο παράθυρο με τις πληροφορίες της εφαρμογής:



ΕΙΚΟΝΑ 17: Αναδυόμενο παράθυρο πληροφοριών εφαρμογής

Επιλέγοντας την επιλογή “Add Device” ο χρήστης μεταφέρεται στη σελίδα προσθήκης συσκευής, στην οποία καλείται να πληκτρολογήσει το όνομα που θέλει να δώσει στη συσκευή του και υστερα να πατήσει το κουμπί “submit”. Στο παρακάτω screenshot φαίνεται η προσπάθεια να προσθέσουμε ένα device με όνομα “sensor1” στο application “mixpothouse2” που δημιουργήσαμε πριν. Σημειώνεται ότι σε αντίθεση με τα ονόματα των εφαρμογών, στα ονόματα των devices υπάρχει μεγαλύτερη ελευθερία, καθώς ο χρήστης μπορεί ξνα χρησιμοποιήσει κεφαλαία αλλά και να χρησιμοποιήσει το ίδιο όνομα device σε πολλαπλές εφαρμογές. Ο μόνος περιορισμός είναι ότι δεν μπορεί να δημιουργήσει δύο συσκευές με το ίδιο όνομα στην ίδια εφαρμογή.

Σελίδα προσθήκης συσκευής:



ΕΙΚΟΝΑ 18: Σελίδα Προσθήκης εφαρμογής

Πατώντας το πλήκτρο submit δίδεται η εντολή στο backend να εκτελέσει τη παρακάτω ακολουθία εντολών:

- **ttnctl device register sensor1 :** Προστίθεται ένα device με όνομα "sensor1" στην εφαρμογή "mixpothouse2"
- **ttnctl device personalize :** Δημιουργούνται τα κελιδιά Nwskey, Appskey αλλά και το Devaddr που απαιτούνται για την επικοινωνία του device με το σύστημα.
- **ttnctl device set sensor1 - -disable-fcnt-check :** Απενεργοποιείται ο έλεγχος του fcnt counter. Σε αντίθετη περίπτωση κάθε φορά που το device έχανε τη τροφοδοσία του ή έκανε κάποιο reset δεν θα μπορούσε να επικοινωνήσει με το σύστημα καθώς το device θα μηδένιζε το fcnt counter, ενώ το σύστημα θα περίμενε fcnt counter μεγαλύτερο κατα ένα από το fcnt counter που είχε το device πριν το reset.

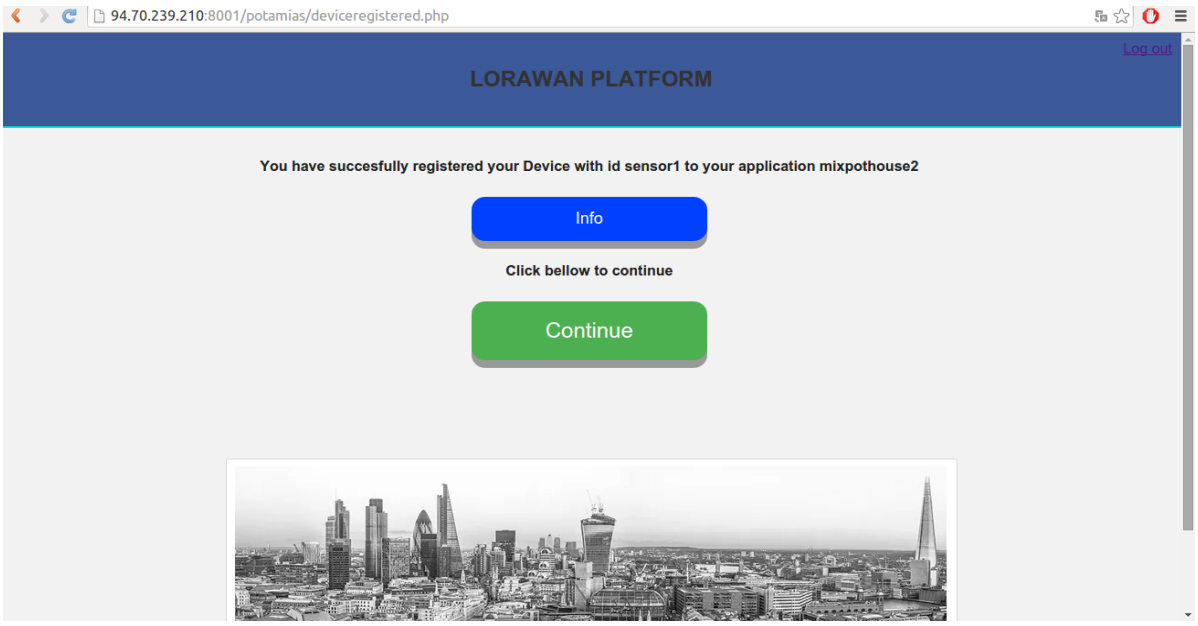
Ο χρήστης αφού κάνει "submit" το device του θα μεταφερθεί στη σελίδα του νέου device. Εκεί έχει τις παρακάτω επιλογές:

- **Info :** Με το κουμπί αυτό εμφανίζεται ένα αναδυόμενο παράθυρο μέσα στο οποίο ο χρήστης καταλαβαίνει αν το device έχει καταχωρηθεί σωστά. Σε αντίθετη περίπτωση εμφανίζεται ένα μήνυμα περιγραφής σφάλματος με το οποίο ο χρήστης θα καταλάβει τι πρόβλημα αντιμετώπισε η ενέργειά του. Αν ωστόσο δεν παρουσιαστεί κάποιο πρόβλημα, εμφανίζονται οι πληροφορίες του device που αφορούν το Appid στο οποίο

καταχωρήθηκε η συσκευή, το ID του device, πληροφορίες σχετικά με τη τελευταία φορά που έστειλε κάποιο μήνυμα η συσκευή, το κλειδί AppEUI της εφαρμογής, το DevEUI της συσκευής, το DEVaddr της συσκευής, το AppSkey, το NwkSkey, Ο fcnt counter καθώς και κάποια flags που αφορούν τη συσκευή.

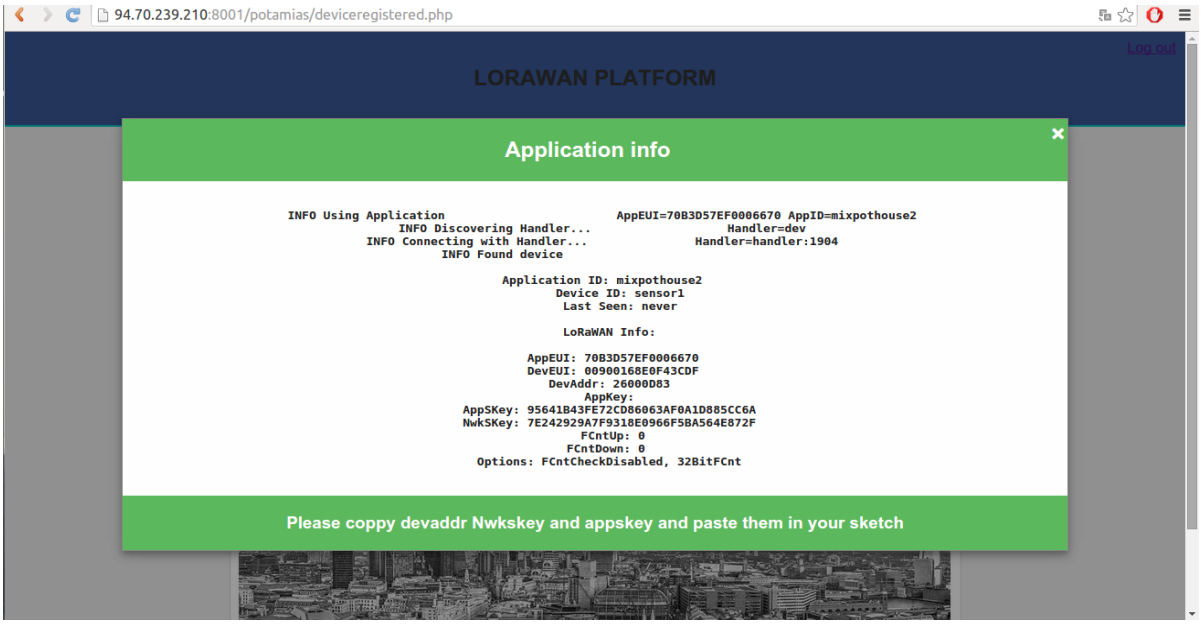
- **Continue :** Επιλέγοντας το κουμπί αυτό, ο χρήστης μεταφέρεται στη σελίδα διαχείρισης εφαρμογών.

Η σελίδα του νέου device:



ΕΙΚΟΝΑ 19: Σελίδα νέας συσκευής

Το ανδυόμενο παράθυρο για τις πληροφορίες της συσκευής:



ΕΙΚΟΝΑ 20: Αναδυόμενο παράθυρο πληροφοριών συσκευής

Πατώντας το πλήκτρο "continue" ο χρήστης μεταφέρεται στο κεντρικό σημείο του site. Στη σελίδα διαχείρησης εφαρμογών. Μέσα από αυτή τη σελίδα ο χρήστης έχει τη δυνατότητα να διαχειριστεί τα device από κάθε μία από τις εφαρμογές που έχει. Σημειώνεται ότι η σελίδα αυτή είναι προσβάσιμη κι από την αρχική σελίδα υποδοχής που παρατέθηκε παραπάνω, μέσω της επιλογής "Application list" μπορούσε να πληκτρολογήσει το όνομα μίας εκ των εφαρμογών και να πλοηγηθεί στη σελίδα διαχείρησής της.

Στη σελίδα διαχείρισης ο χρήστης έχει τις παρακάτω επιλογές

- **Devices List :** Εμφανεται ένα αναδυόμενο παράθυρο στο οποίο παρατίθεται μία λίστα με τα devices που έχει κάνει register ο χρήστης στη συγγεκριμένη εφαρμογή. Μέσα από το παράθυρο αυτό ο χρήστης έχει επιπλέον την επιλογή να ξανά εμφανίσει τους κωδικούς από κάθε device (Devaddr, Nwkskey, Appskey) πληκτρολογώντας το όνομα αυτού στο πλαίσιο που βρίσκεται στο κάτω μέρος του παραθύρου και πατώντας ύστερα το πλήκτρο select.
- **Add Device :** Ο χρήστης μέσω του κουμπιού αυτού έχει τη δυνατότητα ανα πάσα στιγμή να προσθέσει κάποιο καινούριο device στην εφαρμογή του. Με το πάτημα αυτού του κουμπιού ο χρήστης πλοηγείται στη σελίδα προσθήκης συσκευής που παρατέθηκε παραπάνω.
- **Console :** Η επιλογή αυτή μεταφέρει τον χρήστη σε μία νέα σελίδα η οποία εκτυπώνει στην οθόνη τα μηνύματα που στέλνουν οι συσκευές της εφαρμογής σε πραγματικό

χρόνο. Συγκεκριμένα, στη σελίδα εμφανίζεται αρχικά το μήνυμα " Subscribing to Appname..." που σημαίνει ότι συνδέεται με τον broker του συστήματος στο topic που αφορά τη συγκεκριμένη εφαρμογή. Επίσης εμφανίζονται με πράσινο χρώμα οι λέξεις Devaddr,devMAC, rssi, snr,float0,float1 και float2, κάτω από τις οποίες θα προστίθενται οι τιμές του εκάστοτε μηνύματος. Μόλις φτάσει το πρώτο μήνυμα από κάποιο device που ανήκει στην εφαρμογή θα εμφανιστεί μία σειρά με bold μάυρα γράμματα που θα περιέχει το όνομα της συσκευής, τη διεύθυνση MAC της συσκευής, τη τιμή RSSI, τη τιμή SNR καθώς και το payload του μηνύματος. Καθ' όλη τη διάρκεια παραμονής του χρήστη στη σελίδα console εκτελείται στον server ένα python script το οποίο κάνει mosquitto_subscribe στον broker, κάνει parsing το μήνυμα και εκτυπώνει την επιθυμητή πληροφορία. Η έξοδος του python script μεταφέρεται σε real time στη σελίδα μέσω της php συνάρτησης popen που παρατίθεται παρακάτω

```
<?php
$handle = popen('python -u /var/www/html/potamias/parse.py', 'r');
while (!feof($handle)) {
echo fgets($handle);
echo "<br>";
flush();
ob_flush();
}
pclose($handle);
?>
```

- **GRAFANA :** Με την επιλογή αυτή ο χρήστης μπορεί να δει σε μορφή γραφικών παραστάσεων τα δεδομένα που στέλνουν τα devices. Η grafana δίνει τη δυνατότητα στον προγραμματιστη να αναπτύξει JS (Java Scripts) με τα οποία μπορεί να δημιουργεί δυναμικά dashboard όπως εξηγήθηκε σε προηγούμενο κεφάλαιο. Ο χρήστης πατώντας το κουμπί αυτό πλοηγείται στη σελίδα της Grafana στο οποίο έχει δημιουργηθεί ένα dashboard με το όνομα της συσκευής. Η grafana αντλεί τα δεδομένα της όπως έχει αναλυθεί παραπάνω από την influxdb, στη συγγεκριμένη περίπτωση αντλεί τις καταχωρήσεις που αφορούν τη συγκεκριμένη εφαρμογή και γραφικοποιεί τα fields που αφορούν το rssi, snr, frequency, float0, float1, float2 εκτελώντας το ακόλουθο query:

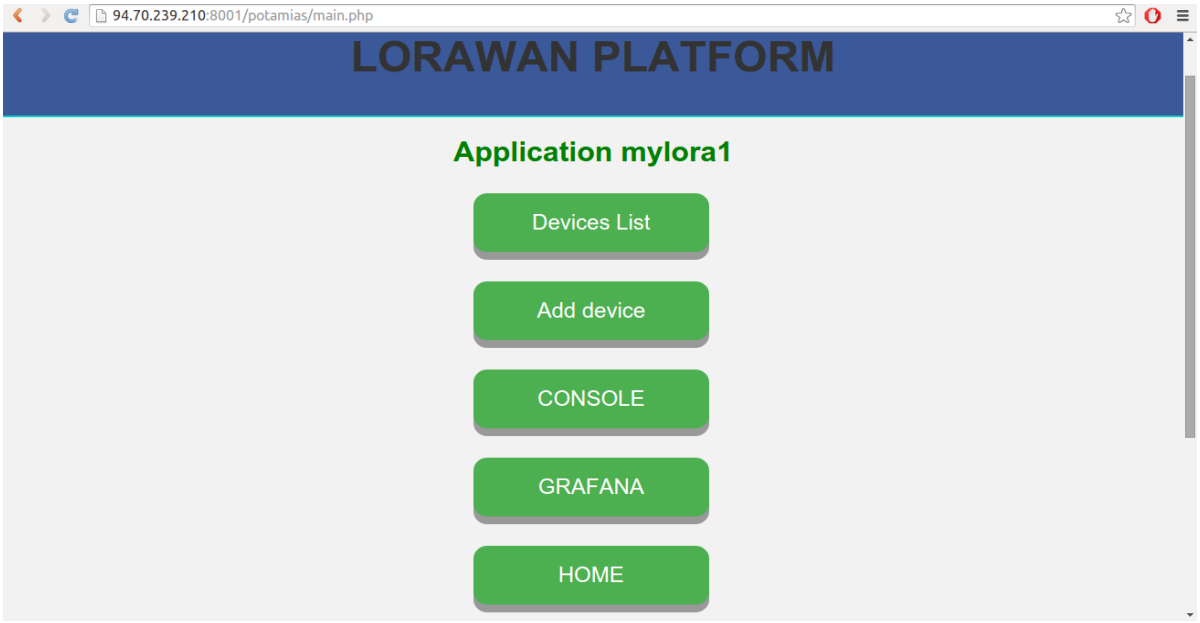
```
query": "SELECT mean(\"float0\") AS \"temperature\", mean(\"frequency\") AS
\"frequency\", mean(\"rssi\") AS \"rssi\", mean(\"snr\") AS \"snr\" FROM \"LoRaWAN\"
WHERE \"appname\" = seriesName AND $timeFilter GROUP BY time($interval)
fill(null)"
```

Όπου serriesName = Appname και μεταφέρεται σαν παράμετρος από τη σελίδα διαχείρισης της εφαρμογής στο js της Grafana.

Στη σελίδα της Grafana ο χρήστης μπορεί να δει γραφική παράσταση από τις Float τιμές που στέλνει σαν payload μία συσκευή, αλλά και πληροφορίες όπως το snr, rssi και frequency. Επιπλέον μπορεί να ρυθμίσει το χρονικό εύρος των δεδομένων καθώς υπάρχουν επιλογές από "τα τελευταία 5 λεπτά" μέχρι "Τα τελευταία 5 χρόνια".

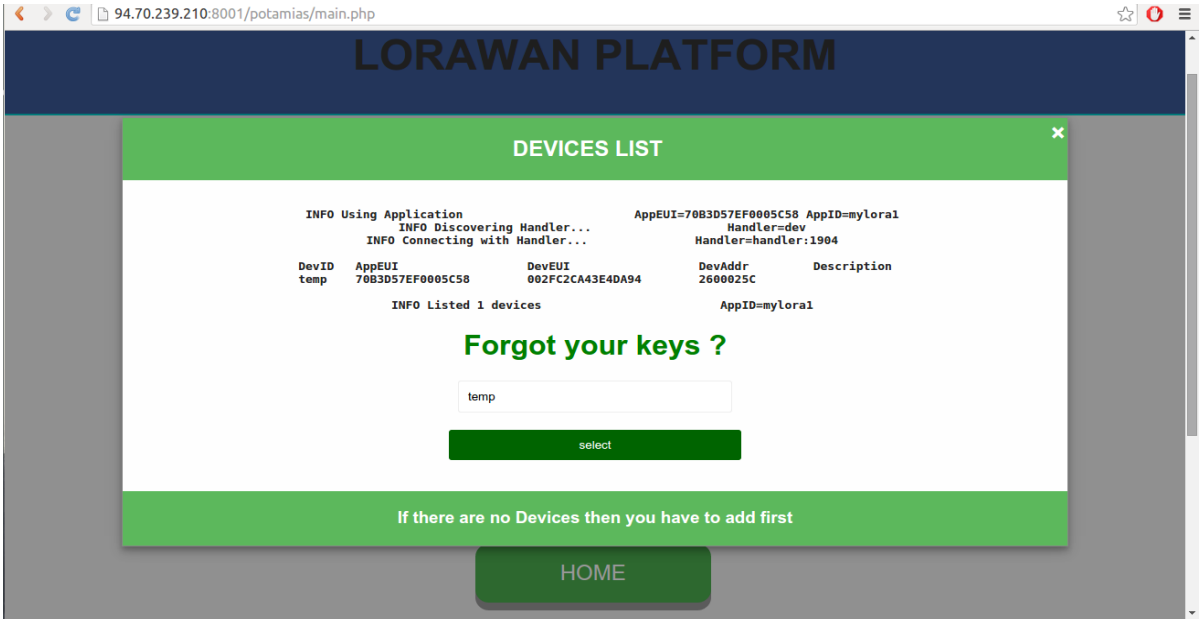
- **Home :** Με την επιλογή αυτή ο χρήστης επιστρέφει στην αρχική σελίδα του site.

Η σελίδα διαχείρισης της εφαρμογής:



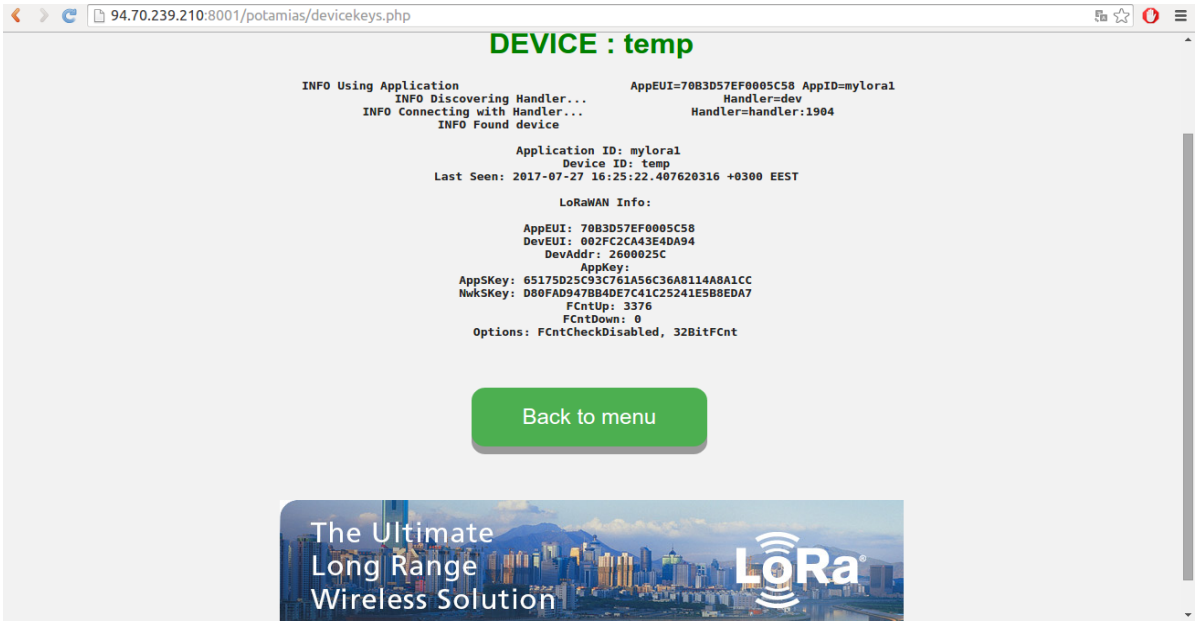
ΕΙΚΟΝΑ 21: Σελίδα Διαχείρισης εφαρμογής

Το παράθυρο με τη λίστα των συσκευών:



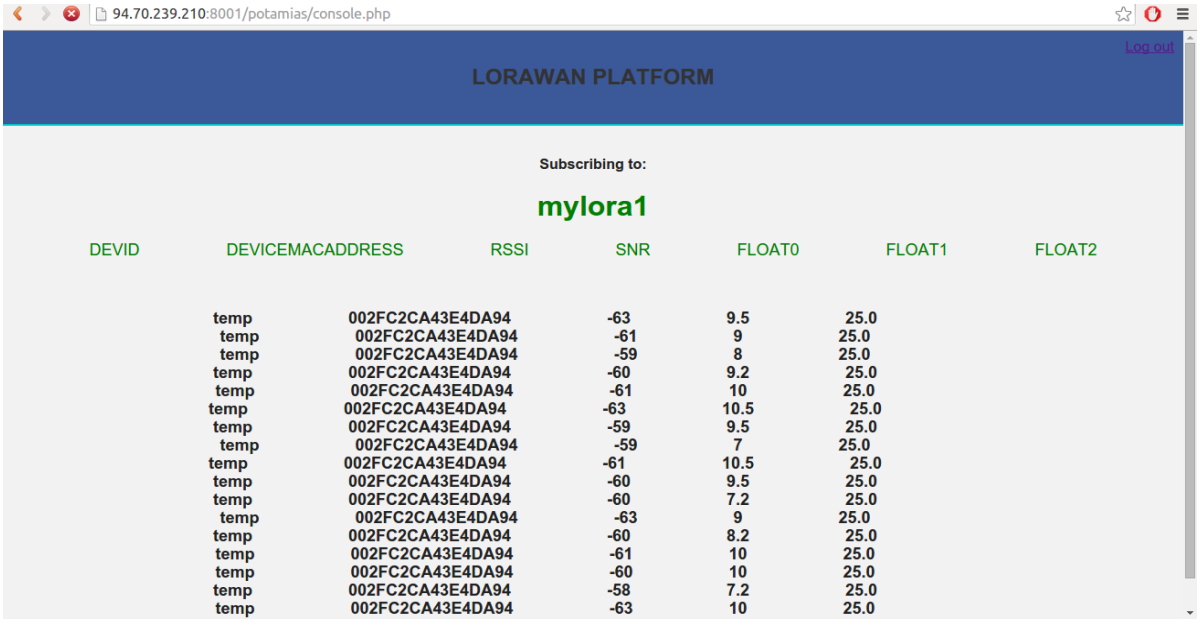
ΕΙΚΟΝΑ 22: Αναδυόμενο παράθυρο λίστας συσκευών

Η σελίδα με τους κωδικούς και τις πληροφοριες κάποιας συσκευής :



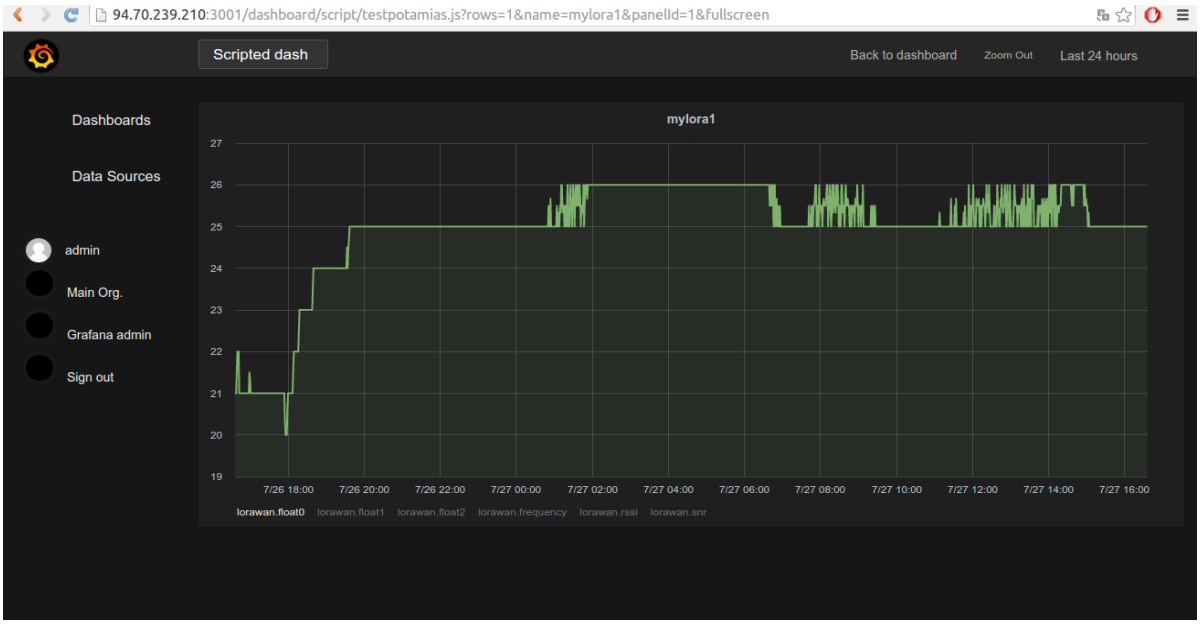
ΕΙΚΟΝΑ 23: Σελίδα πληροφοριών συσκευής

Η σελίδα μηνυμάτων των συσκευών της εφαρμογής :



ΕΙΚΟΝΑ 24: Σελίδα μηνυμάτων

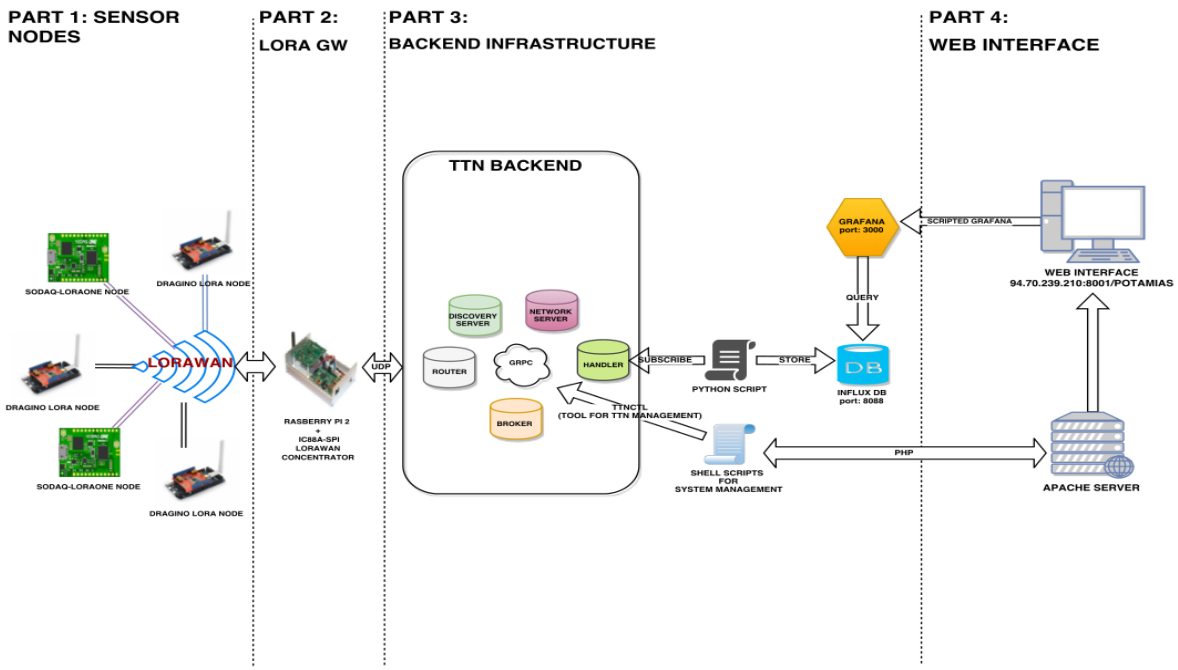
Η σελίδα με το δυναμικό dashboard της Grafana :



ΕΙΚΟΝΑ 25: Σελίδα δυναμικού dashboard της Grafana

6.2 Σχηματική απεικόνιση του συστήματος

Στο 3ο κεφάλαιο παρουσιάστηκε το LoRa gateway που χρησιμοποιείται για την αμφίδρομη επικοινωνία των end-devices (5ο κεφάλαιο) με το backend infrastructure του συστήματος (4ο κεφάλαιο), ενώ στη προηγούμενη παράγραφο έγινε περιγραφή της λειτουργίας του web interface που αναπτύχθηκε για τη διαχείριση του συστήματος. Έχοντας λοιπόν ολοκληρώσει τη περιγραφή όλων των επιμέρους τμημάτων του συστήματος, στο επόμενο σχήμα παρουσιάζεται η σχηματική απεικόνιση των τμημάτων που το απαρτίζουν, καθώς και η μεταξύ τους διασύνδεση.



ΔΙΑΓΡΑΜΜΑ 7: Σχηματική απεικόνιση του συστήματος

7. Σύνοψη

7.1 Σύνοψη και Μελλοντικές επεκτάσεις

Η παρούσα εργασία κινήθηκε σε δύο άξωνες. Στο δεύτερο κεφάλαιο έγινε η παρουσίαση των χαρακτηριστικών της τεχνολογίας LoRa, καθώς και άλλων τεχνολογιών του οικοσυστήματος LPWAN. Στα επόμενα κεφάλαια έγινε η περιγραφή του συστήματος που αναπτύχθηκε σε επίπεδο hardware και software.

Έχοντας αναπτύξει την υποδομή για τη δημιουργία ενός LoRa δικτύου, ο χρήστης μπορεί εύκολα να δημιουργήσει το δικό του δίκτυο χρησιμοποιώντας το web interface που παρουσιάστηκε στο κεφάλαιο 6, αφού πρώτα στρέψει το Gateway προς την IP διεύθυνση του Backend Infrastructure, όπως περιγράφηκε στο κεφάλαιο 3.

Η τεχνολογία LoRa που χρησιμοποιήθηκε δίνει στον χρήστη την δυνατότητα οικοδόμησης εύρωστων δικτύων με οικονομία και ασφάλεια. Μάλιστα, λόγω της μεγάλης εμβέλειάς της, η τεχνολογία LoRa εκτός των άλλων αποτελεί ιδανική λύση για γεωργικές και βιομηχανικές εφαρμογές.

Το σύστημα που αναπτύξαμε μπορεί να αποτελέσει τη βάση πάνω στην οποία να πραγματοποιηθούν μελλοντικές επεκτάσεις της εργασίας. Μία ιδέα είναι να τοποθετηθούν gps σε όλους τους κόμβους του συστήματος και να προστεθεί στο web interface χάρτης μέσα από τον οποίο ο χρήστης να εντοπίζει ανά πάσα στιγμή την ακριβή γεωγραφική θέση των συσκευών. Μία ακόμη ιδέα για μελλοντική επέκταση της εργασίας είναι να χρησιμοποιηθεί το σύστημα που αναπτύχθηκε για την αξιολόγηση του πρωτοκόλλου. Να μελετηθούν δηλαδή οι αποδόσεις των χαρακτηριστικών της τεχνολογίες σε αστικό και σε μη αστικό περιβάλλον προκειμένου να γίνει κάποια σύγκριση με κάποια ανάλογη μελέτη που έχει πραγματοποιηθεί για κάποια άλλη τεχνολογία LPWAN.

ΠΑΡΑΠΟΜΠΕΣ

- **LoRa-Alliance** https://www.LoRa-alliance.org/
- **LPWAN** https://en.wikipedia.org/wiki/LPWAN
- **Sigfox** https://www.sigfox.com/en
- **RPMA** https://www.ingenu.com/technology/rpma/
- **Weightless (N&P)** http://www.weightless.org/
- **LoRa Gateway** https://wireless-solutions.de/products/radiomodules/ic880a.html
- **Raspberry Pi 2** https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
- **The Things Network** https://www.thethingsnetwork.org/
- **TTN on Github** https://github.com/TheThingsNetwork/ttn
- **The Go Programming Language** https://golang.org/
- **Redis** https://redis.io/
- **Rabbitmq** https://www.rabbitmq.com/
- **Docker** https://www.docker.com/
- **Mosquitto** https://mosquitto.org/
- **Grafana** https://grafana.com/
- **InfluxDB** https://www.influxdata.com/
- **Arduino** https://www.arduino.cc/
- **Sodaq One** https://shop.sodaq.com/en/one-eu-rn2483-v2.html
- **Dragino LoRa shield** http://www.dragino.com/products/module/item/102-LoRa-shield.html
- **Python** https://www.python.org/
- **PHP** http://php.net/
- **JSON** http://www.json.org/
- **Javascript** https://www.javascript.com/
- **Mysql** https://www.mysql.com/
- **HTML** https://html.com/
- **CSS** https://www.w3.org/Style/CSS/Overview.en.html
- **Kali Linux** https://www.kali.org/